

Probabilistic Inference for Dynamic Networks and Complex Event Processes



Christopher Mark Lloyd

St John's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2017

To my parents,
Helen and Brian.
For always believing in me.

Acknowledgements

Personal

First and foremost I would like to thank Professor Stephen Roberts for his supervision. From the very first email I sent to Steve to enquire about the possibility of joining his group, he has been extremely supportive. I will be forever grateful to him for supporting my application and securing funding to allow me to come to Oxford. Over the last 5 years I have appreciated Steve's gentle guidance and meetings with him always gave me a great boost to keep going.

I would like to thank Mike Osborne for his help, especially editing paper submissions at the eleventh hour and for his enthusiasm for anything related to Gaussian processes, even my sillier ideas. I have also enjoyed going to quite a few conferences with Mike and discussing his work on the effect of machine learning on employment. I'm not sure whether we quite agree on the latter, suffice to say that I look forward to the vast range of new goods, services *and jobs* that will *become* economically viable thanks to machine learning.

I would like to thank everyone in the lab—I learned a lot from you guys—in particular Rory Beard and Tom Nickson and especially Tom Gunter with whom I worked so productively. I would also like to thank Rory for his excellent knowledge of 1990s television comedy that made me feel slightly less the old man in the lab, Justin Bewsher for organising nights out in Oxford and Asif Ahmad for generously hosting many lab get-togethers.

I would like to thank Frank Wood and James Hensman for agreeing to be my examiners. Furthermore, I would also like to express my gratitude to James for his occasional, but incredibly helpful advice along the way.

Finally, I would like to thank my girlfriend, Bree, for all her love, patience and encouragement, and for all her help proof reading this thesis.

Institutional

I would like to thank DSTL for funding my DPhil and Dr Gareth Brown at DSTL for his interest and support.

Abstract

This thesis introduces new unsupervised machine learning algorithms for complex event data. Event data—records of things that happen—is both ubiquitous and abundant in the modern world. Many types of event occur when two or more distinct entities interact. If these entities are capable of interacting with other entities, the result is typically a network of interactions that is dynamic and complex.

One very general statistical model of events is a Poisson process. In this work, Gaussian processes—another statistical model—are used to modulate the intensity of Poisson processes allowing dynamic patterns of interactions to be captured. This dynamic model can then be further extended using latent variables that represent the missing information needed to unpick the complex interactions present in the data. Learning Bayesian posterior distributions of the latent variables reveals previously hidden structure and also allows statistical strength to be shared across multiple distinct event streams, improving predictive performance.

Efficient inference in these models is challenging however, and a significant contribution of this thesis is an efficient Bayesian inference algorithm for Gaussian process modulated Poisson processes. This scheme scales linearly in the number of observed events and does not require discretisation of the input domain, or expensive Monte-Carlo simulation, that previous posterior inference algorithms for this model required. Latent variable adaptations of this model are presented and the resulting algorithms are applied to data from a variety of different application domains.

Contents

List of figures	xii
List of tables	xv
List of abbreviations	xv
Notation	xxi
1 Introduction	1
1.1 Motivation and applications	3
1.1.1 Computer network security	3
1.1.2 Disease and crime mapping	4
1.1.3 Social network analysis	4
1.1.4 Financial services and e-commerce	5
1.2 Thesis structure & summary of contributions	5
1.2.1 Collaborative work	7
2 Probability theory	9
2.1 Basic properties of random variables	9
2.2 Bayes' theorem	13
2.2.1 Bayesian statistics	13
2.2.2 Comparison with frequentist statistics	15
2.2.3 Specifying the prior	17
2.3 Probability distributions	19
2.3.1 The exponential family	19
2.3.2 The Gaussian distribution	23
2.3.3 The multinomial and Dirichlet distributions	25
2.3.4 Poisson, gamma and negative binomial distributions	26
2.3.5 Comparing distributions	27
2.4 Stochastic processes	28
2.4.1 Gaussian processes	29
2.4.2 Poisson processes	32
2.4.3 The Dirichlet process	37
3 Approximate inference algorithms	43

3.1	Sampling	44
3.1.1	Metropolis-Hastings	45
3.1.2	Hamiltonian Monte-Carlo	46
3.1.3	Slice sampling	48
3.1.4	Elliptical Slice Sampling	49
3.2	Variational inference	51
3.2.1	Mean field variational inference	53
3.2.2	Stochastic variational inference	55
3.2.3	Collapsed variational inference	56
3.3	Approximate Gaussian processes inference	57
3.3.1	Model approximation	58
3.3.2	Variational approximation	59
4	Latent variable models, structure learning and complex networks	63
4.1	Factor models	64
4.1.1	Principle Component Analysis	65
4.1.2	Non-negative Matrix Factorisation	66
4.2	The Gaussian Process Latent Variable Model	68
4.3	Generative models of networks	70
4.3.1	The Stochastic Block Model	71
4.3.2	The Random Function Model	75
4.3.3	Dynamic network models	76
4.3.3.1	Tensor decompositions	76
4.3.3.2	State-space models	80
4.3.4	Exchangeability and properties of networks	82
5	Convolutional point processes	87
5.1	Structured point processes	88
5.1.1	Challenges	91
5.2	Convolutional point processes	92
5.2.1	Model augmentation using thinning	92
5.2.2	Convolution processes	93
5.2.2.1	Sparse latent functions	94
5.2.2.2	Partially independent outputs	96
5.2.3	Inference algorithm	98
5.2.3.1	Re-sampling the thinning points	99
5.2.3.2	Re-sampling the output functions	101
5.2.3.3	Re-sampling the latent functions	101
5.2.4	Adaptive thinning	102
5.2.4.1	Empirical evaluation of adaptive thinning	104
5.2.5	Empirical evaluation of CPP	106
5.2.5.1	Synthetic data	107

5.2.5.2	Twitter data	107
5.2.5.3	Basketball data	109
5.3	Directions for further development	110
6	Variational inference for GP modulated Poisson processes . . .	113
6.1	Model	114
6.2	Inference	114
6.2.1	Measure theoretic aspects	115
6.2.2	Integrating over the domain	117
6.2.3	Expectations at the data points	119
6.2.4	Optimising the bound	121
6.2.5	Locating the inducing points	121
6.3	Variational bounds using other link functions	123
6.4	Predictive distribution	124
6.5	Experiments	126
6.5.1	Benchmarks	126
6.5.2	Synthetic Data	127
6.5.3	Coal mining disaster data	128
6.5.4	Twitter Data	130
6.5.5	Malaria Data	131
7	Models for complex high dimensional data	135
7.1	Kronecker structure	136
7.1.1	Relationship to tensor factorisation	139
7.1.2	Implementation	141
7.2	Latent Poisson Process Allocation	143
7.2.1	Model	145
7.2.2	Collapsed variational bound	146
7.2.3	Model identifiability	148
7.2.4	Dynamic network model	148
7.2.5	Experiments	149
7.2.5.1	Benchmark	150
7.2.5.2	Twitter data	150
7.2.5.3	Basketball data	151
7.2.5.4	Wild bird data	152
7.2.5.5	Enron data	154
7.3	Poisson Process Latent Variable Model	155
7.3.1	Tied latent variables	160
7.3.2	Stochastic inference	161
7.3.3	Back-constrained latent variables	163
7.3.4	Initialisation scheme for network data	165
7.3.5	Demonstration	168

7.3.5.1	Basketball data	168
7.3.5.2	School contact data	169
7.3.5.3	Computer network data	173
7.3.6	Discussion	177
8	Conclusion	181
8.1	Related and contemporary work	181
8.1.1	Point processes	182
8.1.2	Dynamic networks	183
8.2	Further work	185
8.2.1	Hawkes process models	185
8.2.2	Spectral Mixture Kernel	188
8.2.3	Square transformed GP noise model	193
8.2.4	Streaming inference	195
	Appendices	
A	Derivatives of the VBPP lower bound	201
A.1	Variational parameters	201
A.2	Kernel parameters	202
A.3	Kronecker structured variational covariance	203
B	Kronecker matrix implementation	205
C	Collapsed variational inference for the Poisson-IRM model . . .	209
C.1	Poisson-IRM	210
C.2	Collapsed variational lower bound	211
C.3	Conjugate natural gradient descent	213
	References	217

List of figures

2.1	Draws from a Gaussian processes.	31
2.2	Draws from a Gaussian process conditioned on one point.	31
2.3	Draws from a Gaussian process conditioned on two points.	31
2.4	Posterior distribution of a Gaussian process.	31
2.5	20 draws from a homogeneous Poisson process with parameter $\lambda = 2$	33
2.6	A draw from a Poisson process and the corresponding Poisson point process.	33
2.7	A draw from a homogeneous Poisson point process.	34
2.8	A draw from an inhomogeneous Poisson point process.	34
2.9	An inhomogeneous Poisson point process.	34
2.10	Gaussian process distributed random functions used to construct Cox process intensity functions.	35
2.11	Log-Gaussian Cox process intensity functions.	35
2.12	Sigmoid-Gaussian Cox process intensity functions.	35
2.13	Square-transformed-Gaussian Cox process intensity functions.	35
2.14	Draws from a Dirichlet process.	38
2.15	A draw from a stick breaking process.	39
2.16	Matrix representation of a draw from a Chinese Restaurant Process.	41
2.17	Matrix representation of a draw from an Indian Buffet Process.	41
3.1	Illustration of slice sampling.	50
3.2	Illustration of elliptical slice sampling.	52
4.1	The typical structure of a latent variable model.	64
4.2	An example of an adjacency matrix showing assortative and dissortative block structure.	72
4.3	Link correlation with 2 features and 1 feature per node.	74
4.4	Link correlation with 10 features and 5 features per node.	74
4.5	Link correlation with 10 features and 2 features per node.	74
4.6	Link correlation with 10 features and 9 features per node.	74
4.7	The CP and Tucker tensor decompositions.	77
4.8	The Dynamic Mixed Membership Stochastic Block Model.	81
5.1	Basketball court markings and overlaid shot data for three NBA players.	89

5.2	Point process representation of a Dynamic Interaction Network. . .	90
5.3	Draws from a convolution process.	94
5.4	Graphical model of the Convolution Point Process model	98
5.5	Illustration of adaptive thinning.	105
5.6	Synthetically generated intensity functions.	107
5.7	Learned intensity functions for synthetic data learned using CPP. .	107
5.8	The learned intensities for MP tweet data.	108
5.9	Basketball shot intensity functions learned using the CPP model. .	110
6.1	$\tilde{G}$ as a function of its argument.	120
6.2	The tightness of the variational lower bound.	125
6.3	2D Synthetic data intensities obtained from three methods.	128
6.4	The inferred intensity functions for the coal mining disaster dataset.	131
6.5	The predictive log-likelihood for the coal mining disaster dataset. .	131
6.6	The inferred intensity functions for the Twitter dataset.	132
6.7	The predictive log-likelihood for the Twitter dataset.	132
6.8	The intensity of malaria incidences in Kenya.	133
7.1	Correlated latent intensity functions constructed using a convolution process.	144
7.2	Uncorrelated square transformed GP latent intensity functions. . .	144
7.3	Bases found using LPPA and Twitter data.	151
7.4	Bases found using LPPA and basketball data.	152
7.5	Relative locations of feeding stations in Wytham Great Wood. . . .	153
7.6	Bases found using LPPA and wild bird data.	154
7.7	Bases found using N-LPPA and Enron email data.	156
7.8	Uncertainty in the location of the latent coordinates.	160
7.9	Tied latent variables in the PPLVM model.	162
7.10	A back-constrained inference model.	164
7.11	Back constrained latent variables in the PPLVM model.	165
7.12	Slices through the basketball shot intensity surface for different latent variable values.	170
7.13	Initial and optimised locations of the latent variables for each basketball player.	171
7.14	The stochastic ELBO during optimisation with basketball data. . .	172
7.15	The temporal distribution of physical interactions in the school contact dataset.	174
7.16	Slices through the node interaction surface for contact network data.	176
7.17	Single point intensities for a single node.	176
7.18	The stochastic ELBO during optimisation with contact network data.	177
7.19	Communications between hosts in a computer network.	178
7.20	Initial and optimised locations of the latent variables for computer network data.	179

7.21 Slices through the node interaction surface for selected IP ports. . .	180
8.1 Hawkes process with exponential rate function.	186
8.2 Hawkes process with attenuated Gaussian process rate function. . .	187
8.3 A draw from a Gaussian process with a spectral mixture kernel. . .	190
8.4 Power spectral density.	191
8.5 Streaming intensity function inference set-up.	196
C.1 Graphical model of the Poisson Infinite Relation Model.	210

List of tables

2.1	Natural and expectation parameters of the Gaussian distribution. . .	24
2.2	Natural and expectation parameters for the multinomial distribution and Dirichlet distributions.	25
2.3	Natural and expectation parameters for the Poisson, gamma and negative binomial distributions.	27
3.1	Summary of sparse Gaussian process approximations.	60
4.1	NMF algorithms are based on a variety of divergences measures. . .	68
4.2	Tensor decomposition algorithms.	80
5.1	Adaptive thinning results in one dimension.	106
5.2	Adaptive thinning results in two dimensions.	106
5.3	Predictive log-likelihood for held out twitter data across models. . .	109
5.4	Basketball data held-out predictive log-likelihood.	109
6.1	Test log-likelihood for 1D synthetic data.	129
6.2	RMS error and wall-clock run time for 1D synthetic data.	129
6.3	Test log-likelihood for 2D synthetic data.	129
6.4	RMS error and wall-clock run time for 2D synthetic data.	129
6.5	The predictive log-likelihood for the malaria dataset.	132
7.1	Matrix operations which can preserve or exploit Kronecker structure.	139
7.2	Twitter held out predictive log-likelihoods.	151
7.3	Basketball data held out predictive log-likelihoods.	152

List of abbreviations

ARD	Automatic Relevance Determination	66, 80, 118, 119, 159
BLAS	Basic Linear Algebra System	142
BNP	Bayesian non-parametric	28, 29, 209
BPTF	Bayesian Poisson Tensor Factorisation	79
BSMJP	Binary Semi-Markov Jump Processes	183
CP	CANDECOMP-PARAFAC	77–80, 139, 141, 148
CPP	Convolutional Point Process	6, 87, 92, 98, 106–110
CR	Common Rate	150
CRP	Chinese Restaurant Process	37, 40, 41, 73
CT	Common Topic	150
DIC	Deterministic Inducing Conditional	58–60, 76
DIN	Dynamic Interaction Network	76, 79, 80, 82, 88–90, 148, 181, 184, 210
DMMSBM	Dynamic Mixed Membership Stochastic Block Model	80, 81, 183
DP	Dirichlet Process	37, 40
DTC	Deterministic Training Conditional	59, 60
ELBO	Evidence Lower Bound	52, 53, 55, 69, 114, 116, 136, 138, 142, 157, 160, 162, 172, 177, 197, 201, 203
EM	Expectation Maximisation	66, 80, 184
EP	Expectation Propagation	43
ESS	Elliptical Slice Sampling	45, 49, 51, 99, 101, 103

FA	Factor Analysis	65
FFBS	Forward Filtering and Backward Sampling	183
FIC	Fully Independent Conditional	58–60, 96
FITC	Fully Independent Training Conditional	59, 60
GP	Gaussian process 29, 30, 33–37, 43, 57–61, 69, 75, 76, 87, 91–93, 95, 96, 100–103, 110, 111, 113–115, 117, 120–123, 136, 139, 140, 144, 155–157, 164, 182–184, 188–190, 193–195	
GPLVM	Gaussian Process Latent Variable Model 68, 69, 118, 140, 159, 183	
HMC	Hamiltonian Monte-Carlo	45, 46, 99, 101
HMM	Hidden Markov Model	33
IBP	Indian Buffet Process	40, 41, 73, 183
ICA	Independent Component Analysis	65
IFRM	Latent Feature Relational Model	73
i.i.d.	Independent and identically distributed	38, 44, 61, 83
ILAM	Infinite Latent Attribute Model	73
INLA	Integrated Nested Laplace Approximation	182
IP	Internet Protocol	174, 178–180
IRM	Infinite Relational Model 72–74, 167, 183, 184, 209, 210, 212, 214	
KL	Kullback–Leibler (divergence) 21, 27, 28, 52–55, 115, 124, 141, 157, 160, 161, 163, 172, 203	
KS	Kernel Smoothing	106, 108, 109, 126, 128, 150–152
L-BFGS	Limited-Memory Broyden–Fletcher–Goldfarb–Shanno 121, 147, 159	
LCGP	Log-Gaussian Cox Process	34, 123, 127, 182
LDA	Latent Dirichlet Allocation	148
LPPA	Latent Poisson Process Allocation 6, 143, 146, 148–152, 154, 183	

MAP	Maximum A' Posterior	19, 70, 121, 182
MCMC	Markov-Chain Monte-Carlo	44, 68, 69, 80, 93, 98, 99, 111, 126, 182, 184, 209
MH	Metropolis-Hastings	45–47, 99, 100, 103
ML	Maximum Likelihood	18, 193
MMSBM	Mixed Membership Stochastic Block Model	80, 183
MP	Member of Parliament	106
N-LPPA	Network Latent Poisson Process Allocation	148, 149, 156, 184
NMF	Non-negative Matrix Factorisation	66–68, 70, 111, 143, 148, 150–152
PCA	Principle Component Analysis	65–68
PIC	Partially Independent Conditional	58, 59, 96, 99
PITC	Partially Independent Training Conditional	59, 60
PPLVM	Poisson Process Latent Variable Model	6, 155, 156, 162, 164–166, 168, 176, 177
PUS	Polya Urn Scheme	37, 39, 41
RFID	Radio Frequency IDentification	152
RFM	Random Function Model	74–76, 84, 140, 141
RMS	Root Mean Squared	105, 106, 127–129
RT	Run Time	129
SBM	Stochastic Block Model	70–73, 79, 84, 90, 184, 209
SGCP	Sigmoid-Gaussian Cox Process	34, 104, 106, 108, 109, 113, 126–130
SGD	Stochastic Gradient Descent	162, 172
SMK	Spectral Mixture Kernel	119, 177, 188
SPFM	Semi-Parametric Factor Model	92, 143
SS	Slice Sampling	99, 101
SVD	Singular Valued Decomposition	65, 66, 80

VAE	Variational Auto Encoders	164
VB	Variational Bayes	43, 54, 80
VBPP	Variational Bayesian Poisson Processes	6, 126–130, 132, 133, 182, 183, 185, 193, 195

Notation

a, x, σ	Scalar variables are denoted using lower case italicized letters.
$\mathbf{a}, \mathbf{x}, \boldsymbol{\sigma}$	Vectors are denoted using lower case bold type face letters.
$\mathbf{A}, \mathbf{X}, \boldsymbol{\Sigma}$	Matrices are denoted using upper case bold face letters.
$\underline{\mathbf{A}}, \underline{\mathbf{X}}, \underline{\boldsymbol{\Sigma}}$	Tensors are denoted using underline bold face letters.
$a_{1:N} = a_1, \dots, a_N$. . .	Lists of variables are denoted using subscript ranges.
$f(n) = \mathcal{O}(g(n))$. . .	“Big-O”: $\exists k > 0, \exists n_0, \forall n > n_0 \bullet f(n) \leq k \cdot g(n)$.
$f(n) = \mathcal{o}(g(n))$. . .	“Little-O”: $\forall k, \exists n_0, \forall n > n_0 \bullet f(n) \geq k \cdot g(n) $.
$f(n) = \Omega(g(n))$. . .	“Big-Omega”: $\exists k > 0, \exists n_0, \forall n > n_0 \bullet f(n) \geq k \cdot g(n)$.
$\triangleq$	Equal by definition.
$\stackrel{d}{=}$	Equal in distribution.
$:=$	Set equal to (assignment).
$\otimes$	Kronecker product.
$\odot$	Hadamard or element-wise product.
$\text{vec}(\cdot)$	The vectorisation of a matrix.
$\text{vech}(\cdot)$	The vectorisation of the lower triangular matrix.
$\underline{\text{vec}}(\cdot)$	The vectorisation of a tensor.
$\text{diag}(\cdot)$	Diagonalise a matrix by zeroing off-diagonal elements.
$\text{blkdiag}(\cdot)$	Block-diagonalise a matrix by zeroing elements.
$ \cdot $	The determinant of a matrix or the size/measure of a set.
$\delta(x)$	The Dirac delta function: $+\infty$ if $x = 0$, 0 if $x \neq 0$.
$\mathbb{1}\{b\}$	The indicator function: 1 if b is true, 0 if b is false.
$\mathbb{I}$	The identity matrix.
$\mathbb{R}$	The real numbers.

$\mathbb{N}$	The natural numbers.	
$\mathbb{P}$	A probability measure	10
$\mathbb{H}[\cdot]$	The entropy of a distribution	12
$\mathbb{E}[\cdot]$	The expectation operator	12
$\text{Var}[\cdot]$	The variance operator	12
$\text{KL}(\cdot \cdot)$	The Kullback–Leibler divergence	28
$\mathcal{I}(\cdot)$	The Fisher information	21
$\mathcal{B}(\cdot)$	The Beta distribution	25
$\mathcal{D}(\cdot)$	The Dirichlet distribution	25
$\mathcal{G}(\cdot)$	The Gamma distribution	26
$\mathcal{N}(\cdot)$	The normal or Gaussian distribution	23
$\mathcal{NB}(\cdot)$	The negative binomial distribution	27
$\mathcal{P}(\cdot)$	The Poisson distribution	26
$\mathcal{M}(\cdot)$	The multinomial distribution	25
$\mathcal{U}(\mathcal{U})$	The uniform distribution: $p(X = x \in \mathcal{U}) = \frac{1}{ \mathcal{U} }$.	
$\mathcal{GP}(\cdot)$	The Gaussian process	29
$\mathcal{DP}(\cdot)$	The Dirichlet process	37
$\mathcal{PP}(\cdot)$	The Poisson process	32
$\overset{iid}{\sim}$	Independently and identically distributed (i.i.d.).	

1

Introduction

An event is simply a thing that has or will happen at a particular point in time and space. What might be deemed an event in a particular context depends on whether the happening has some significance in that context. We might think of important events in the context of our own lives for example. Any particular event is, by definition, unique; a one-off that doesn't happen twice. That said, to recognise an event as such, we must supply a definition of what counts as an event. If the definition is sufficiently expansive we will be able to find other happenings that meet our definition. On the other hand, if the definition is sufficiently contrived, perhaps we might struggle to find other examples. Cricket commentators are particularly expert at this: “This is the first time an England opening batsman has scored more than 100 runs in the second innings of a third test against the West Indies at Edgbaston—on a Tuesday!” We can construct such definitions because events are also abstract concepts that lie in a vast multidimensional space and so we can classify events along many different axes and using many different criteria.

If it is possible to see more events of the same type in the future, we can think of these events as being generated by an on-going *event process*. Some elementary questions we can ask about such an event process are:

- What is the physical system driving the process?
- How many events have occurred?
- What is the rate at which events are occurring?
- How will the rate of events change in the future?
- How is the rate of events related to other types of event?

Answering these questions may be trivial or very difficult depending on the complexity of the event process, the desired accuracy of the answer and the availability of suitable data upon which to draw our inferences.

In the modern world we can—and do—collect enormous amounts of event data. Every time we *interact* with a computer system, for example by making an online purchase or by booking a hospital appointment, we are, inadvertently, generating event data. While this data may be of little interest to the end-user, insight gleaned from it may be of great interest to people concerned with, say, optimising sales on their website or planning health care provision (Diggle et al. 2005; Lasko 2014). The specific information recorded will inevitably vary greatly depending on the purpose of the particular system in question. Event data will generally include a *time stamp* in addition to a variety of other structured and unstructured data such as location, duration, various discrete labels, free-form text etc.

Of course, *interaction events* do not necessarily occur only between a central service provider and isolated users; interactions can also occur between, for example, distributed users in a peer-to-peer network. We can think of such networks manifesting themselves as collections of interaction events in which each interaction is labelled with the connected users (or in more abstract language, *nodes*) and the time of communication. The interactions between any pair of nodes will likely influence further interactions and at any moment in time different configurations of users are communicating. Such a network is inherently dynamic and complex.

1.1 Motivation and applications

The aim of this work is to develop computationally tractable inference algorithms for the rate of interaction events in dynamic networks and other complex event processes. These models can be applied to datasets from many different application areas and the outputs can be used for a variety of purposes. Specifically, the goal is to build models that use information from one event process to inform and improve inference of the rate of other related event processes. For example, by understanding how the rate of some common event is related to the rate of another, less common event, we can use the more abundant data of the former to help estimate the rate of the latter. We would also like to allow the various types of side information associated with events to be usefully included in the inference process. This is a form of *data fusion* in which multiple data sources or data types are combined in order to produce an output that is superior to that achieved using any single data source alone.

Understanding the rate at which events are occurring is useful for many reasons. If there is an economic or social cost of particular types of event, we need to understand the rate at which they occur in order to make provisions. We may also be interested in detecting changes in the prevalence of events. Understanding how the rate of different event processes are related may give useful insight into physical systems driving event processes.

As there are many potential application areas for this work we will, by way of motivation briefly discuss several candidates.

1.1.1 Computer network security

A computer network is a rich source of event data. Computer networks are typically monitored by a variety of security systems such as virus checkers and intruder detection systems. The number of packets of data routed between host computers can be logged by the network infrastructure, for example Cisco Systems ‘Netflow’

data. Using such data, it is possible to estimate a statistical model of normal network behaviour. This model is then used to test subsequent network data for statistically unlikely deviations. This approach is used to detect sophisticated “zero-day” cyber-attacks that are not detected by simple rule based systems, especially those using well-known popular rule sets. No matter what security vulnerabilities are exploited by an attack, a network host cannot be compromised without communicating with it and this is difficult to do without changing the pattern of communication.

1.1.2 Disease and crime mapping

Epidemiologists collect large amounts of data related to the time and location of individual cases of particular diseases, for example malaria and dengue fever (*Malaria Atlas Project* 2017). Since mosquitoes transmit these diseases, their spread is greatly influenced by environmental factors that are conducive to mosquito reproduction, as well as economic deprivation that impedes effective preventative measures. Factors such as rainfall, vegetation, temperature, income and population density can all be used to improve the inference of the disease rate function. Obtaining good estimates for the prevalence of these diseases helps governments and relief organisations allocate resources in the most effective manner.

Similarly we may be interested in mapping crime over some geographic region (Jewell et al. 2015). Since the pervasiveness of similar types of crime is affected by the same social and economic factors, we may expect the rate at which these crimes occur to be strongly correlated.

1.1.3 Social network analysis

Social scientists are interested in the social structures that give rise to particular patterns of behaviours. It has long been recognised that social networks are characterised by the intensity of interactions between the participants (Granovetter 1973). Since these structures may not be directly measurable, observed data may fail to distinguish the particular social structures that generate each interaction. A

widely used notion in social network analysis is the concept of community, which is based on the idea that like-minded individuals will tend to form their own groupings. Statistical models of interaction can be used to reveal such hidden structures. The increasing use of social media, such as Twitter, has lead to an increase in the availability of social network data to which these methods can be applied. Other animals also form social networks. Using electronic means, modern zoologists are able to record animal interactions in the wild. This information can then be analysed in order to better understand the relationships they form.

1.1.4 Financial services and e-commerce

In the financial markets and other commercial market places, clients are continually bidding on, and trading in, a wide variety of products with many different service providers. These trading entities form a high dimensional complex dynamic network that is constantly adapting to new information as the participants compete to gain commercial advantage over each other. Estimating the intensity of financial transactions is an important part of predicting future price changes and therefore identifying buying or selling opportunities, as well as managing inventory and credit risk (Basu and Dassios 2002).

1.2 Thesis structure & summary of contributions

This thesis is organised in roughly two parts: Chapters 2-4 comprise mostly introductory and background material and Chapters 5-7 containing the main body of new work. Conclusions and further work are presented in Chapter 8.

Chapter 2 provides an overview of the mathematical framework used throughout—probability theory and Bayesian inference. In particular, this chapter introduces two important stochastic processes that underpin much of the new work presented herein—namely the Gaussian process and the Poisson process. Chapter 3 describes the methodologies required to perform inference in the models developed in the

later chapters. Chapter 4 outlines important related statistical models that have inspired and guided this work.

The primary contributions advanced in this thesis are as follows:

- Methods for improving the efficiency of inference of continuous Gaussian process modulated Poisson processes using:
 - Variational inference: the VBPP model, Chapter 6 & Publication 2
 - Adaptive thinning, Chapter 5 & Publication 1
- Models and inference algorithms for correlated rate processes constructed via:
 - Convolution processes: the CPP model, Chapter 5 & Publication 1
 - Sums of positive factors: the LPPA model, Chapter 7 & Publication 3
 - Latent variables: the PPLVM model, Chapter 7
- A method for improving the efficiency of variational inference in Gaussian processes using Kronecker structure, Chapter 7 & Publications 3 and 5

Most of the contributions listed have been presented in the following publications:

1. *Efficient Bayesian Non-parametric Modelling of Structured Point Processes*,
T. Gunter*, **C. Lloyd***, M. A. Osborne, S. J. Roberts,
Conference on Uncertainty in Artificial Intelligence, 2014.
2. *Variational Inference for Gaussian Process Modulated Poisson Processes*,
C. Lloyd*, T. Gunter*, M. A. Osborne, S. J. Roberts,
International Conference on Machine Learning, 2015.
3. *Latent Poisson Process Allocation*,
C. Lloyd, T. Gunter, T. Nickson, M. A. Osborne, S. J. Roberts,
Conference on Artificial Intelligence and Statistics, 2016.
4. *Inferring Dynamic Interaction Networks with N-LPPA*,
C. Lloyd, T. Gunter, M. A. Osborne, S. J. Roberts,
Networks in the social and information sciences, NIPS workshop, 2015.

*Joint first authors

5. *Blitzkriging: Kronecker-structured Stochastic Gaussian Processes*,
T. Nickson, T. Gunter, **C. Lloyd**, M. A. Osborne, S. J. Roberts,
Arxiv, 1510.07965, 2015.

Publications 1-3 are peer-reviewed conference papers, Publication 4 is a workshop paper and Publication 5 is a self-archived publication.

1.2.1 Collaborative work

Publication 1 was a joint collaboration with T. Gunter including theoretical development, software implementation and the conducting of experiments. T. Gunter wrote most of the paper.

Publication 2 was a joint collaboration with T. Gunter from a theoretical perspective. T. Gunter suggested the use of the square-transform following his previous work on Gaussian quadrature. I wrote most of the paper and software, and conducted the experiments. T. Gunter also assisted with data curation.

Publication 3: T. Gunter was involved in theoretical discussions and assisted with data curation and experiments. I wrote most of the paper, with T. Gunter providing a section on permanental point processes. T. Nickson provided some C software related to Kronecker structured matrix-vector operations.

Publication 4: T. Gunter assisted with data curation.

Publication 5: I was only involved in theoretical discussions.

In all papers M. A. Osborne and S. J. Roberts helped with background and editing.

2

Probability theory

Almost all data is random, by which we mean: subject to apparent fluctuations that we cannot account for or control. A mathematically rigorous treatment of random data must therefore provide a formalism to characterise and reason about these fluctuations. Such a treatment is provided by *probability theory*. Probability theory also provides the mathematical underpinning of Bayesian statistics, which is the philosophical view of statistics we adopt. In this chapter we will briefly review some basic concepts of probability theory and formally introduce the notation used throughout this thesis. We will then introduce several important statistical distributions and processes that are the probabilistic “building blocks” that will be used to construct models of complex event data in later chapters.

2.1 Basic properties of random variables

A formal treatment of probability theory usually begins with the consideration of a random variable, X , which is the abstract representation of an experiment. All outcomes of the experiment are assigned a numerical value. Therefore, a random variable is—strictly speaking—a function that maps from an abstract set of outcomes, Ω , to the numerical value assigned to the outcome. We assign a

probability to possible sets of outcomes of the random variable. The probability is a numerical measure of how likely the sets of outcomes are relative to each other.

A random variable, $X : \Omega \rightarrow \mathbb{R}$, is defined on a *Probability Space*, is a 3-tuple $(\Omega, \mathcal{F}, \mathbb{P})$ where:

- Ω is the abstract set of *outcomes*
- $\mathcal{F}$ is an *event space* also known as a σ -algebra
- $\mathbb{P} : \mathcal{F} \rightarrow \mathbb{R}$ is a *probability measure*.

The set $\mathcal{F}$, a collection of subsets of Ω , and the probability measure¹ $\mathbb{P}$ obey the following *Axioms of Probability*:

- $\emptyset \in \mathcal{F}, \Omega \in \mathcal{F}$
- If $A, B \in \mathcal{F}$, then $A^c \in \mathcal{F}$ and $A \cup B \in \mathcal{F}$
- If $A_i \in \mathcal{F}, \forall i \geq 1$ then $\cup_{i=1}^{\infty} A_i \in \mathcal{F}$
- $\mathbb{P}(A) \geq 0, \forall A \in \mathcal{F}$
- $\mathbb{P}(\Omega) = 1$
- If $A_i \in \mathcal{F}, \forall i > 1$ and $A_i \cap A_j = \emptyset$ for all pairs $i \neq j$ then

$$\mathbb{P}(\cup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mathbb{P}(A_i)$$

Treating random variables as functions can lead to a somewhat overloaded notation, so it is usual to drop references to Ω and replace, for example, $\mathbb{P}(\{\omega \in \Omega : X(\omega) = x\})$ with $\mathbb{P}(\{X = x\})$ or even $\mathbb{P}(X = x)$.

If the values a random variable, X , can take, i.e. $\text{Im}(X)$, is countable or countably infinite, the variable is discrete, otherwise it is continuous.

Discrete random variables have an associated *probability mass function* $P_X : \mathbb{R} \rightarrow [0, 1]$ which is short hand for $P_X(x) \triangleq \mathbb{P}(X = x)$. Continuous random variables are defined via a *cumulative probability function*: $F_X(x) \triangleq \mathbb{P}(X \leq x)$. If the probability

¹ When building models using standard probability distributions, measure theoretic considerations can usually be safely ignored. We will therefore limit discussion of measure-theoretic aspects to one occasion, in Chapter 6 (§6.2.1), where it will be necessary.

measure F_X can be written as

$$F_X(x) = \int_{-\infty}^x p_X(x) dx, \quad (2.1)$$

where dx indicates integration with respect to the *Lebesgue* measure, then p_X is called the *probability density function* of X .

Events A_1 and A_2 are *independent* if

$$\mathbb{P}(A_1 \cup A_2) = \mathbb{P}(A_1)\mathbb{P}(A_2). \quad (2.2)$$

For events A_1 and A_2 the *conditional distribution* is the probability of event A_1 occurring, given that event A_2 has occurred and is given by

$$\mathbb{P}(A_1|A_2) = \frac{\mathbb{P}(A_1 \cup A_2)}{\mathbb{P}(A_2)}. \quad (2.3)$$

Any experiment of arbitrary complexity could be considered to be a single random variable, e.g. X . However, when building probabilistic models it is usually convenient to decompose the set of outcomes into the cross product of sets of outcomes, each associated with separate random variables, $X, Y, Z, \dots$. Over these random variables there is a single *joint measure* $\mathbb{P}(X \in A, Y \in B, Z \in C, \dots)$. This case can also be extended to *multi-variate* random variables $\vec{X} = [X_1, \dots, X_N]^\top$ that take values $\mathbf{x} = [x_1, \dots, x_N]^\top$.

Random variables X and Y are independent if:

$$\mathbb{P}(X \in A, Y \in B) = \mathbb{P}(X \in A)\mathbb{P}(Y \in B), \quad (2.4)$$

and the conditional distribution of X , given the random variable Y is:

$$\mathbb{P}(X \in A|Y \in B) = \frac{\mathbb{P}(X \in A, Y \in B)}{\mathbb{P}(Y \in B)}. \quad (2.5)$$

Discrete random variables have joint probability mass functions, for example:

$$P_{X,Y}(x, y) = \mathbb{P}(X = x, Y = y), \quad (2.6)$$

while continuous random variables have joint cumulative distribution functions:

$$F_{X,Y}(x, y) = \mathbb{P}(X \leq x, Y \leq y). \quad (2.7)$$

Two random variables, X and Y , are *jointly continuous* if for some joint density function p :

$$F_{X,Y}(x, y) = \int_{-\infty}^x \int_{-\infty}^y p_{X,Y}(u, v) \, dy \, dx. \quad (2.8)$$

The *marginal* distribution of a joint distribution is obtained by integrating out all but one of the random variables covered by the joint:

$$\mathbb{P}(X = x) = \sum_{y \in \text{Im}(Y)} \mathbb{P}(X = x, Y = y). \quad (2.9)$$

The most important *statistic* of a random variable is its mean or *expected value*; it is the average value the random variable takes on under all possible outcomes. The expected value of a discrete random variable, X , is defined as:

$$\mathbb{E}[X] \triangleq \sum_{x \in \text{Im}(X)} x P_X(x). \quad (2.10)$$

The expected value of a continuous random variable, X , is defined as:

$$\mathbb{E}[X] \triangleq \int_{-\infty}^{\infty} x p_X(x) \, dx. \quad (2.11)$$

We can also compute the expected value of the random variable X transformed by some real-valued function h according to:

$$\mathbb{E}[h(X)] \triangleq \int_{-\infty}^{\infty} h(x) p_X(x) \, dx. \quad (2.12)$$

Since the expectation operator, $\mathbb{E}$, is a linear operator, we can usually compute the expected value of a random variable, X , under a linear operator, L , by exchanging the order of the operations and therefore $\mathbb{E}[L(X)] = L(\mathbb{E}[X])$.

Another statistic of great importance is the *variance* of a random variable. Variance measures the spread of the random variable around its mean and is the expected squared deviation from the mean. The variance of a random variable X , $\text{Var}[X]$ is defined as:

$$\text{Var}[X] \triangleq \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2, \quad (2.13)$$

and for two random variables the *covariance* is defined as

$$\text{Cov}[X, Y] \triangleq \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]. \quad (2.14)$$

The *entropy* of a random variable, a measure of its uncertainty, is defined as:

$$\mathbb{H}[X] = - \int p_X(x) \log p_X(x) dx. \quad (2.15)$$

2.2 Bayes' theorem

For any pair of random variables X, Y we can decompose their joint probability $\mathbb{P}(X, Y)$ as the product of conditional and marginal distributions in two ways and so $\mathbb{P}(X, Y) = \mathbb{P}(X|Y)\mathbb{P}(Y) = \mathbb{P}(Y|X)\mathbb{P}(X)$. A simple rearrangement leads to Bayes' theorem which, formally stated, asserts that if $\{A_1, A_2, \dots\}$ is a partition of Ω such that $\mathbb{P}(X \in A_i) > 0$ and $Y \in B$ is an event with $\mathbb{P}(Y \in B) > 0$, then

$$\begin{aligned}\mathbb{P}(X \in A_j|Y \in B) &= \frac{\mathbb{P}(Y \in B|X \in A_j)\mathbb{P}(X \in A_j)}{\mathbb{P}(Y \in B)} \\ &= \frac{\mathbb{P}(Y \in B|X \in A_j)\mathbb{P}(X \in A_j)}{\sum_i \mathbb{P}(Y \in B|X \in A_i)\mathbb{P}(X \in A_i)}.\end{aligned}\quad (2.16)$$

Bayes' theorem extended to continuous random variables leads to conditional densities. The conditional density of a multivariate continuous random variable $\vec{X} = \mathbf{x}$ given $\vec{Y} = \mathbf{y}$ computed using Bayes' theorem is:

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})} = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{\int p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) \, d\mathbf{x}}. \quad (2.17)$$

Note that in this second definition of Bayes' theorem we did not distinguish between the random variable and the value which it takes and we dropped the subscripts indicating which random variable the probability distribution is defined over. In the remainder of this work we will adopt these notational short cuts, provided it is obvious from the context which probability distribution we are referring to.

Bayes' theorem is central to a branch of statistics called Bayesian statistics. The qualifier "Bayesian" reflects a historical schism within statistics between Bayesian statisticians and frequentist statisticians, although modern statisticians are generally much less dogmatic than their predecessors.

2.2.1 Bayesian statistics

The definition of Bayes' rule, Equation 2.16, takes on added significance when we consider how Bayes' rule is used in the context of Bayesian analysis. The variables we are interested in are our observable data $\mathcal{D}$, and some unobservable parameter

of interest, θ , often described as a state-of-nature or world model. The quantity we wish to compute is the *posterior* distribution of the parameter given the data, $p(\theta|\mathcal{D})$. Using Bayes' rule we can compute this quantity according to:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \quad (2.18)$$

$$= \frac{p(\mathcal{D}|\theta)p(\theta)}{\int p(\mathcal{D}|\theta)p(\theta) \, d\theta}. \quad (2.19)$$

The function $p(\mathcal{D}|\theta)$ is known as the data likelihood and $p(\theta)$ is the prior distribution. The prior distribution captures our knowledge about θ , while the likelihood comes from our assumptions about how the parameter and data interact.

In principle this is the only calculation required for a Bayesian analysis. The posterior distribution over θ contains all the information that we have about θ from which we can compute any statistic of θ or, in the framework of Bayesian decision theory (Berger 1985), select an action a^* that minimises an expected loss:

$$a^* = \operatorname{argmax}_a \mathbb{E}_{p(\theta|\mathcal{D})}[L(a; \theta)], \quad (2.20)$$

where $L(a, \theta)$ is the loss associated with taking action a under the condition that world model has value θ .

In practice there is a major stumbling block: the (potentially) high dimensional integral—called the *model evidence*—that appears in the denominator of Equation 2.19. Only when the prior and likelihood form a conjugate pair, described in §2.3.1, can these integrals be computed in closed-form. Otherwise this integral can very quickly become intractable due to the *curse of dimensionality*. Historically this has been a major impediment to the application of Bayesian methods. Although approximate methods exist for the calculation of these integrals, it is only in recent decades (since ~ 1990) that these schemes have become feasible, due to increasing computational power and availability. Another issue is specifying the prior distribution, which has been seen as controversial by some frequentist statisticians.

2.2.2 Comparison with frequentist statistics

Frequentist statistics is concerned with the development of procedures which have good frequentist properties, that is, they give good answers under repeated use. This is achieved by evaluating the procedure under a range of possible datasets. Parameters on the other hand are considered to be fixed but unknown quantities.

Hypothesis testing is a typical example of frequentist analysis in which contrasting null and alternative hypothesis for some particular value (or values) of a parameter are constructed. The analysis proceeds by considering the likelihood of observing other data more extreme than the observed data under these hypothesis. This leads to the development of p-values and confidence intervals—the values of test statistic (i.e. the number produced by the procedure) at which the null-hypothesis can be rejected. Neyman Pearson theory is used to develop tests that balance Type I errors—alternative hypothesis is incorrectly accepted—and Type II errors—the null hypothesis incorrectly fails to be rejected (false positives and false negatives respectively).

In contrast, the Bayesian view is a conditional view. We condition on the particular data we observed and treat the parameters as random variables. Having observed the data, we generally don't consider other data we might have observed (although there are exceptions to this).

This distinction reflects an important philosophical difference between Bayesian and frequentist statistics: the interpretation of probability itself. A frequentist interpretation of probability is the limiting frequency of an outcome in a repeated experiment, while the Bayesian interpretation is that of *degrees of belief*.

Cox's Theorem shows that the calculus of probability theory and Bayes' theorem is the extension of logical reasoning in the presence of uncertainty (Jaynes 2003). Uncertainty may be epistemic—the uncertainty due to our own ignorance—or aleatoric—the uncertainty due to inherent randomness—however, probability theory is the appropriate framework in either case. Thus, in the Bayesian setting it is

perfectly natural to use the language of probability to formalise one's own subjective knowledge.

The Bayesian interpretation of probability is an intuitive meaning of probability familiar to bookmakers and weather forecasters alike. The Bayesian interpretation of probability is the odds that a gambler would be indifferent to a bet. It is natural for, say, a Bayesian weather forecaster to consider the probability that it will rain tomorrow even though tomorrow is not an infinitely repeatable experiment. A weather forecaster observing a red sky tonight, can legitimately update his or her posterior belief about precipitation tomorrow to incorporate the new information. Moreover, Bayes' rule provides the definitive prescription of how to achieve this.

Frequentist confidence intervals on the other hand do not necessarily correspond directly to degrees of belief, unless the procedure has a Bayesian interpretation. This can complicate the interpretation of frequentist hypothesis tests. A further advantage of the Bayesian approach is coherence. Informally, coherence means that we get the same answer, irrespective of how we ask the question. In a frequentist framework, it is possible to construct tests that give contradictory answers (i.e. they are incoherent). Bayesian methods are coherent since they are (coherence preserving) manipulations of the joint distribution—which is by definition coherent—using the calculus of probability theory.

Bayesian methods should also have good frequentist properties. Von Mises' Theorem (van der Vaart 1998) asserts that², as more information is observed, the information in the likelihood overwhelms the prior and the posterior distribution converges to a (scaled, centred) Gaussian distribution. Therefore a Bayesian point estimator (e.g. the mean or mode of the posterior) should fall within the n^{th} percentile of the true parameter n percent of the time in repeated experiments (provided sufficient data), giving valid credible intervals in a frequentist sense.

²For finite, parametric models with a prior that is smooth in the region of the true parameter.

2.2.3 Specifying the prior

In addition to distinctions from frequentist statistics, Bayesian approaches can themselves be categorised under the headings: subjective Bayesian, objective Bayesian and empirical Bayesian. These subcategories differ in their approach to specifying the prior distribution.

The subjective Bayesian is perhaps the purest form. In the subjective framework, the prior distribution is truly our best attempt to capture knowledge about the state of nature. The subjective Bayesian places a lot of emphasis on *knowledge elicitation*: working with domain experts to understand how they think about the problem domain.

The emphasis on expert domain knowledge is problematic when such knowledge is unavailable or for models in which it is inherently difficult to specify priors due to complicated interactions between parameter values and data.

Use of subjective prior distributions is considered non-scientific by those concerned that priors provide an avenue to deliberately tamper with the results of an analysis. The alternative view is that the prior distributions are a transparent and disputable means to quantify subjective knowledge and are thus more scientific (Ghahramani 2013).

Objective Bayesian methods aim to minimise or remove subjectivity in prior selection by a proscriptive procedure to determine the prior for given likelihood. Objective Bayesian priors can be thought of as priors that have the least impact on the posterior. This is not always a uniform prior over the parameter space. Several methods have been proposed for constructing objective Bayesian priors including Jeffrey's prior, reference priors and maximum entropy priors. The Jeffrey's prior is motivated by the desire to construct a prior distribution for a likelihood that is invariant to re-parametrisation. If re-parametrisation changes the prior, then the choice of parametrisation could allow subjective modification of the prior. It follows that a necessary condition for an objective prior must be invariance. The

solution to this problem is to pick the prior $p(\theta) \propto \mathcal{I}(\theta)$, where $\mathcal{I}(\theta)$ is the *Fisher Information* (§2.3.1) associated with likelihood $p(\mathcal{D}|\theta)$. Reference priors are found by finding a distribution that maximises the *mutual information* between the prior and the data, while maximum entropy priors are those which impose the weakest constraints. These methods very often produce the same result, not unusually an *improper prior*, that is, one which cannot be normalised.

In spite of the philosophical desirability of choosing a prior that best captures subjective knowledge, or alternatively which minimises subjectivity, it is still usual to pick prior distributions that are *conjugate* to the likelihood (see §2.3.1). Even so, a conjugate prior may have free *hyper-parameters* to be selected. Empirical Bayes' is the pragmatic solution in which we use the data itself to select the prior hyper-parameters; typically the ω^* which maximises the marginal-likelihood,

$$\omega^* = \operatorname{argmax}_{\omega} \int p(\mathcal{D}|\theta)p(\theta|\omega) \, d\theta \quad (2.21)$$

$$= \operatorname{argmax}_{\omega} p(\mathcal{D}|\omega), \quad (2.22)$$

is chosen. For this reason the empirical Bayes' approach is also called maximum-marginal likelihood. Having selected ω^* we then compute the posterior distribution of θ , conditioned on ω^* using Bayes' rule:

$$p(\theta|\mathcal{D}, \omega^*) = \frac{p(\mathcal{D}|\theta)p(\theta|\omega^*)}{\int p(\mathcal{D}|\theta)p(\theta|\omega^*) \, d\theta}. \quad (2.23)$$

Selection of the hyper-parameters follows the standard (frequentist) principle of Maximum Likelihood (ML) to select model parameters θ :

$$\hat{\theta}_{\text{ML}} = \operatorname{argmax}_{\theta} p(\mathcal{D}|\theta), \quad (2.24)$$

except the maximisation occurs at the hyper-parameter level, rather than at the parameter level.

This maximum-marginal-likelihood is also related to a Bayesian treatment of a hierarchical model where ω is fully integrated out:

$$p(\theta|\mathcal{D}) = \frac{\int p(\mathcal{D}|\theta)p(\theta|\omega)p(\omega) \, d\omega}{\int \int p(\mathcal{D}|\theta)p(\theta|\omega)p(\omega) \, d\theta \, d\omega}. \quad (2.25)$$

Of course, the *hyper-prior* $p(\omega)$ may also have hyper-hyper-parameters which could themselves be optimised.

Another approach is the Maximum A' Posterior (MAP) method for selecting parameters:

$$\hat{\theta}_{\text{MAP}} = \underset{\theta}{\operatorname{argmax}} p(\mathcal{D}|\theta)p(\theta). \quad (2.26)$$

where the prior is used as a *regulariser*, but not used to integrate out the parameter. MAP regularisation can be applied to hyper-parameters, although we will not employ this approach and instead use subjective, hierarchical priors and maximum-marginal methods.

2.3 Probability distributions

All the probability distributions used in this work are members of an important class of probability distributions called the *exponential family*. The exponential family has important properties that facilitate tractable inference in probabilistic models. In this section we introduce the exponential family in general and define several of members of that family, highlighting some important properties that we will make use of later chapters.

2.3.1 The exponential family

The exponential family is a class of probability distributions that contains many well-known probability distributions including the *Gaussian* or *normal* distribution, the *Poisson* distribution and the *Dirichlet* distribution. All members of the exponential family can be written in the following *canonical* form:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \exp(\boldsymbol{\theta}^\top \mathbf{t}(\mathbf{x}) + c(\mathbf{x}) - \psi(\boldsymbol{\theta})), \quad (2.27)$$

where $\boldsymbol{\theta} \in \mathbb{R}^R$ is the canonical or *natural* parameter, $\mathbf{t}(\mathbf{x})$ is a vector of *sufficient statistics*, $c(\mathbf{x})$ is called the *carrier-measure* and $h(\mathbf{x}) = \exp(c(\mathbf{x}))$ is the *base-measure*.

The function $\psi(\boldsymbol{\theta})$ is known as the *log-partition function*. Since $\int p(\mathbf{x}|\boldsymbol{\theta})d\mathbf{x} = 1$, it follows that

$$1 = \frac{\int \exp(\boldsymbol{\theta}^\top \mathbf{t}(\mathbf{x}))h(\mathbf{x}) d\mathbf{x}}{\exp(\psi(\boldsymbol{\theta}))} \quad (2.28)$$

$$\implies \psi(\boldsymbol{\theta}) = \log \int \exp(\boldsymbol{\theta}^\top \mathbf{t}(\mathbf{x}))h(\mathbf{x}) d\mathbf{x}, \quad (2.29)$$

and therefore $\psi(\boldsymbol{\theta})$ is responsible for normalising the distribution and is uniquely determined by the choice of $\mathbf{t}(\mathbf{x})$ and $c(\mathbf{x})$.

If we differentiate Equation 2.28 with respect to the natural parameter and rearrange as follows:

$$0 = \nabla_{\boldsymbol{\theta}} \int \exp(\boldsymbol{\theta}^\top \mathbf{t}(\mathbf{x}) - \psi(\boldsymbol{\theta}))h(\mathbf{x}) d\mathbf{x} \quad (2.30)$$

$$\implies 0 = \int [\mathbf{t}(\mathbf{x}) - \nabla\psi(\boldsymbol{\theta})] \exp(\boldsymbol{\theta}^\top \mathbf{t}(\mathbf{x}) - \psi(\boldsymbol{\theta}))h(\mathbf{x}) d\mathbf{x} \quad (2.31)$$

$$\implies \mathbb{E}[\mathbf{t}(\mathbf{x})] = \nabla\psi(\boldsymbol{\theta}) \quad (2.32)$$

we see the intimate relationship between the sufficient statistic and the log-partition function: the expected value of the sufficient statistic is equal to the gradient of the log-partition function.

When working with exponential family distributions, it is often more convenient to use an alternative parametrisation in terms of a new variable, $\boldsymbol{\eta}$, called the *expectation parameter*, that is equal to the expected value of the sufficient statistic:

$$\boldsymbol{\eta} = \mathbb{E}[\mathbf{t}(\mathbf{x})] = \nabla\psi(\boldsymbol{\theta}). \quad (2.33)$$

As we shall now see, the natural and expectation parameters are the convex duals of each other (Amari and Nagaoka 2007; Amari 2016).

If we differentiate Equation 2.28 with respect to $\boldsymbol{\theta}$ twice and rearrange:

$$\nabla\nabla\psi(\boldsymbol{\theta}) = \int [\mathbf{t}(\mathbf{x}) - \nabla\psi(\boldsymbol{\theta})][\mathbf{t}(\mathbf{x}) - \nabla\psi(\boldsymbol{\theta})]^\top \exp(\boldsymbol{\theta}^\top \mathbf{t}(\mathbf{x}) - \psi(\boldsymbol{\theta}))h(\mathbf{x}) d\mathbf{x} \quad (2.34)$$

$$= \mathbb{E}[(\mathbf{t}(\mathbf{x}) - \mathbb{E}[\mathbf{t}(\mathbf{x})])(\mathbf{t}(\mathbf{x}) - \mathbb{E}[\mathbf{t}(\mathbf{x})])^\top] \quad (2.35)$$

$$= \text{Cov}[\mathbf{t}(\mathbf{x}), \mathbf{t}(\mathbf{x})], \quad (2.36)$$

we see that the log-partition function, $\psi(\boldsymbol{\theta})$, must be convex, since its second derivative—the covariance of the sufficient statistic—is positive semi-definite. The Hessian of the log-partition function, $\nabla\nabla\psi(\boldsymbol{\theta})$, is the *Fisher information* of the distribution, denoted $\mathcal{I}(\boldsymbol{\theta})$.

The convex conjugate of $\psi(\boldsymbol{\theta})$ is given by the *Legendre transform*:

$$\psi^*(\boldsymbol{\theta}^*) = \sup_{\boldsymbol{\theta}} (\boldsymbol{\theta}^\top \boldsymbol{\theta}^* - \psi(\boldsymbol{\theta})). \quad (2.37)$$

Since $\psi(\boldsymbol{\theta})$ is convex, the maximum value of $\boldsymbol{\theta}^*$ occurs at $\boldsymbol{\theta}^* = \nabla\psi(\boldsymbol{\theta})$, which we recognise as $\boldsymbol{\eta}$ from Equation 2.33 and therefore:

$$\psi^*(\boldsymbol{\eta}) = \boldsymbol{\theta}^\top \boldsymbol{\eta} - \psi(\boldsymbol{\theta}). \quad (2.38)$$

Furthermore, since $\psi^*(\boldsymbol{\eta})$ is—by definition—convex, if we apply the Legendre transform to it, we find the solution $\boldsymbol{\eta}^* = \nabla\psi^*(\boldsymbol{\eta}) = \boldsymbol{\theta}$ and recover $\psi(\boldsymbol{\theta})$. Thus $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ are functions of each other, related through the gradient of the log-partition function $\psi(\boldsymbol{\theta})$ and its conjugate $\psi^*(\boldsymbol{\eta})$.

In the case $c(\mathbf{x}) = 0$, the dual log-partition function, $\psi^*(\boldsymbol{\eta})$, is equal to the *negative-entropy* of the distribution as shown by:

$$\underbrace{\mathbb{E}[\log p(\mathbf{x}|\boldsymbol{\theta})]}_{\text{negative entropy}} = \boldsymbol{\theta}^\top \mathbb{E}[\mathbf{t}(\mathbf{x})] - \psi(\boldsymbol{\theta}) + \underbrace{\mathbb{E}[c(\mathbf{x})]}_{=0 \text{ if } c(\mathbf{x})=0} \quad (2.39)$$

$$= \boldsymbol{\theta}^\top \boldsymbol{\eta} - \psi(\boldsymbol{\theta}) \quad (2.40)$$

$$= \psi^*(\boldsymbol{\eta}). \quad (2.41)$$

In the case $c(\mathbf{x}) \neq 0$, the dual log-partition function is equal to the *relative entropy* or Kullback–Leibler (KL) divergence (see §2.3.5) between the probability distribution and the normalised base-measure, plus a constant, as shown by the following:

$$\psi^*(\boldsymbol{\eta}) = \mathbb{E}[\log p(\mathbf{x}|\boldsymbol{\theta})] - \mathbb{E}[\log h(\mathbf{x})] \quad (2.42)$$

$$= \mathbb{E}[\log p(\mathbf{x}|\boldsymbol{\theta})] - \mathbb{E}[\log \bar{h}(\mathbf{x})] - \log Z \quad (2.43)$$

$$= \int p(\mathbf{x}|\boldsymbol{\theta}) \log \frac{p(\mathbf{x}|\boldsymbol{\theta})}{\bar{h}(\mathbf{x})} d\mathbf{x} - \log Z \quad (2.44)$$

$$= \text{KL}(p(\mathbf{x}|\boldsymbol{\theta}) || \bar{h}(\mathbf{x})) - \log Z \quad (2.45)$$

where $\bar{h}(\mathbf{x}) = \frac{1}{Z}h(\mathbf{x})$ is the normalised base-measure and Z is the normalising constant.

The Fisher information in the natural and expectation parameters are also related. Let g_{ij} be the $i^{\text{th}}j^{\text{th}}$ element of the matrix $\mathcal{I}(\boldsymbol{\theta})$ and let g_{jk}^* be the $j^{\text{th}}k^{\text{th}}$ element of $\mathcal{I}(\boldsymbol{\eta})$, then

$$g_{ij}g_{jk}^* = \frac{\partial^2 \psi(\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j} \frac{\partial^2 \psi^*(\boldsymbol{\eta})}{\partial \eta_j \partial \eta_k} = \frac{\cancel{\partial \eta_j}}{\partial \theta_i} \frac{\partial \theta_k}{\cancel{\partial \eta_j}} = \mathbb{1}\{i = k\}, \quad (2.46)$$

and therefore the Fisher information matrix in the natural parameters, $\mathcal{I}(\boldsymbol{\theta})$, is equal to $\mathcal{I}^{-1}(\boldsymbol{\eta})$, the inverse of the Fisher information matrix in the expectation parameters.

The log-concavity of the exponential family has benefits for parameter estimation. In particular, given data $\mathcal{D} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$ the *maximum likelihood* estimate of the expectation parameter is the mean value of the sufficient statistic

$$\hat{\boldsymbol{\eta}}_{\text{ML}} = \frac{1}{N} \sum_{n=1}^N \mathbf{t}(\mathbf{x}^{(n)}). \quad (2.47)$$

A second important property of the exponential family distribution is the existence of conjugate priors (Bernardo and Smith 1994) for the parameter $\boldsymbol{\theta}$, that allow the posterior distribution of $\boldsymbol{\theta}$ to be computed in closed-form, without the need for numerical integration. The prior $p(\boldsymbol{\theta})$ is chosen to have a similar form as the likelihood Equation 2.27,

$$p(\boldsymbol{\theta}) = \exp(\boldsymbol{\theta}^\top \boldsymbol{\omega}_0 + C(\boldsymbol{\omega}_0, n_0) - n_0 \psi(\boldsymbol{\theta})), \quad (2.48)$$

parametrised by a positive scalar value n_0 and a vector $\boldsymbol{\omega}_0$. The posterior computed using Bayes' rule is

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{\prod_{n=1}^N p(\mathbf{x}^{(n)}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \quad (2.49)$$

$$= \frac{\exp(\boldsymbol{\theta}^\top \boldsymbol{\omega}_N + C(\boldsymbol{\omega}_0, n_0) + \sum_n c(\mathbf{x}^{(n)}) + [n_0 + N]\psi(\boldsymbol{\theta}))}{p(\mathcal{D})} \quad (2.50)$$

$$= \exp\left(\boldsymbol{\theta}^\top \boldsymbol{\omega}_N + C(\boldsymbol{\omega}_N, n_0 + N) + [n_0 + N]\psi(\boldsymbol{\theta})\right). \quad (2.51)$$

The posterior has exactly the same form as the prior, except the prior parameter $\boldsymbol{\omega}_0$ is replaced with the posterior parameter $\boldsymbol{\omega}_N = \boldsymbol{\omega}_0 + \sum_n t(\mathbf{x}^{(n)})$ and n_0 is replaced with $n_0 + N$. Thus in conjugate exponential families the parameters of the prior can be interpreted as n_0 pseudo observations each with sufficient statistic $\frac{1}{n_0}\boldsymbol{\omega}_0$.

The model evidence is given by the ratio of normalising terms:

$$p(\mathcal{D}) = \frac{\exp(C(\boldsymbol{\omega}_0, n_0) + \sum_n c(\mathbf{x}^{(n)}))}{\exp(C(\boldsymbol{\omega}_N, n_0 + N))}. \quad (2.52)$$

We will now introduce a number of important probability distributions that are all members of the exponential family. Tables 2.1, 2.2 and 2.3 give the natural and expectation parameters for these distributions in terms of their more familiar *source* parameters, as well as the sufficient statistics, log-partition functions and carrier-measures for the distributions written in the canonical form of Equation 2.27.

2.3.2 The Gaussian distribution

The Gaussian distribution is, probably, the most well known of all probability distributions. The ubiquity of the Gaussian distribution is due to *central limit theorem*: the limit of sums of random variables converge to a Gaussian distribution. Therefore, many real world observable quantities that are the result of many random perturbations follow a Gaussian distribution.

The univariate Gaussian, which traces out the famous “bell-curve”, has a probability density $p(x|\mu, \sigma) = \mathcal{N}(x; \mu, \sigma^2)$, where the source parameters μ and σ^2 are the mean and variance of the distribution. The probability density function is given by:

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right), \quad x \in \mathbb{R}. \quad (2.53)$$

The multivariate Gaussian distribution, denoted $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$, has probability density function:

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{M/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right), \quad \mathbf{x} \in \mathbb{R}^R. \quad (2.54)$$

The Gaussian distribution has many important and interesting properties. Two useful properties are: (1) the Gaussian distribution is the conjugate prior for the mean of the Gaussian distribution and (2) the convolution of a Gaussian distribution with another Gaussian distribution is also a Gaussian distribution. We can condition a Gaussian distributed random variable on an affine transformation of another Gaussian random variable as follows:

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad (2.55)$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}; \mathbf{H}\mathbf{x} + \mathbf{b}, \mathbf{R}), \quad (2.56)$$

and obtain marginal and posterior probability distributions in closed-form:

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}; \mathbf{H}\boldsymbol{\mu} + \mathbf{b}, \mathbf{H}\boldsymbol{\Sigma}\mathbf{H}^\top + \mathbf{R}) \quad (2.57)$$

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}^+, \boldsymbol{\Sigma}^+) \quad (2.58)$$

$$\boldsymbol{\Sigma}^+ = \left(\boldsymbol{\Sigma}^{-1} + \mathbf{H}^\top \mathbf{R}^{-1} \mathbf{H} \right)^{-1} \quad (2.59)$$

$$\boldsymbol{\mu}^+ = (\boldsymbol{\Sigma}^+)^{-1} \mathbf{H}^\top \mathbf{R}^{-1} (\mathbf{y} - \mathbf{b}), \quad (2.60)$$

where $\boldsymbol{\mu}^+$ and $\boldsymbol{\Sigma}^+$ are the posterior mean and covariance respectively.

The marginal distribution Equation 2.57 is the result of a convolution and Equations 2.58-2.60 are the result of applying Bayes' rule. These equations are useful for developing many linear Gaussian models, for instance Equation 2.57 may be recognised as the Kalman filter prediction equation, while Equations 2.58-2.60 are the Kalman filter update equations written in *information form* (Durrant-Whyte et al. 1991).

	$\mathbf{t}(\mathbf{x})$	$\boldsymbol{\theta}$	$\boldsymbol{\eta}$	$\psi(\boldsymbol{\theta})$
$\mathcal{N}(x; \mu, \sigma^2)$	$[x, \ x^2]$	$[\frac{\mu}{\sigma^2}, \ -\frac{1}{2\sigma^2}]$	$[\mu, \ \mu^2 + \sigma^2]$	$-\frac{\theta_1^2}{4\theta_2} + \frac{1}{2} \log -\frac{\pi}{\theta_2}$
$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	$[\mathbf{x}, \ \mathbf{x}\mathbf{x}^\top]$	$[\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}, \ -\frac{1}{2}\boldsymbol{\Sigma}^{-1}]$	$[\boldsymbol{\mu}, \ \boldsymbol{\mu}\boldsymbol{\mu}^\top + \boldsymbol{\Sigma}]$	$-\frac{1}{4}\theta_1^\top \theta_2^{-1} \theta_1 + \frac{1}{2} \log -\frac{\pi^D}{ \theta_2 }$

Table 2.1: Natural and expectation parameters of the univariate and multivariate Gaussian distributions. In both cases $c(\mathbf{x}) = 0$. Note that for the multivariate Gaussian θ_1 is a vector and θ_2 is a matrix.

2.3.3 The multinomial and Dirichlet distributions

The multinomial distribution is a multivariate discrete distribution that is associated with sampling *with* replacement. The multinomial distribution has probability mass function

$$\mathcal{M}(x_{1:K}; n, p_{1:K}) = \frac{n!}{\prod_{k=1}^K x_k!} \prod_{k=1}^K p_k^{x_k} \quad \text{s.t.} \quad \sum_{k=1}^K p_k = 1 \quad \text{and} \quad \sum_{k=1}^K x_k = n. \quad (2.61)$$

The conjugate prior of the multinomial distribution is the Dirichlet distribution:

$$\mathcal{D}(x_{1:K}; \alpha_{1:K}) = \frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K x_k^{\alpha_k - 1}, \quad (2.62)$$

where α is the *shape* parameter and β is the *rate* parameter.

The multinomial distribution for two variables is known as the *binomial* distribution, $\text{Bin}(x_1, x_2; n, p) = \mathcal{M}(x_1, x_2; n, p, (1-p))$. The multinomial distribution of just one sample is known as the *categorical* distribution, $\text{Cat}(x_{1:K}; p_{1:K}) = \mathcal{M}(x_{1:K}; 1, p_{1:K})$. The binomial distribution of just one sample is known as the *Bernoulli* distribution, $\text{Bern}(x_1, x_2; p) = \text{Bin}(x_1, x_2; 1, p)$. Similarly Dirichlet distribution for two variables is known as the *Beta* distribution,

$$\mathcal{B}(x_1, x_2; \alpha, \beta) = \mathcal{D}(x_1, x_2; \alpha, \beta). \quad (2.63)$$

The categorical distribution is widely used in statistical modelling to construct *mixture models*. To draw samples from such models we first draw categorically distributed random variables that are used to index another set of distributions from which the data points are then sampled.

	$\mathbf{t}(\mathbf{x})$	$\boldsymbol{\theta}$	$\boldsymbol{\eta}$	$\psi(\boldsymbol{\theta})$	$c(\mathbf{x})$
$\mathcal{M}$	$[x_{1:K}]$	$\left[\log\left(\frac{p_{1:K}}{p_K}\right)\right]$	$n[p_{1:K}]$	$n \log\left(\sum_{k=1}^K \exp(\theta_k)\right)$	$\log(n!) - \sum_{k=1}^K \log(x_k!)$
$\mathcal{D}$	$[\log x_{1:K}]$	$[\alpha_{1:K} - 1]$	$\left[\frac{\alpha_{1:K}}{\sum_{k=1}^K \alpha_k}\right]$	$\sum_{k=1}^K \log \Gamma(\theta_k + 1) - \log \Gamma\left(\sum_{k=1}^K (\theta_k + 1)\right)$	0

Table 2.2: Natural and expectation parameters for the multinomial distribution (with known n) and Dirichlet distributions.

2.3.4 Poisson, gamma and negative binomial distributions

The *Poisson* distribution is another ubiquitous distribution. The Poisson arises as the limit of the Binomial distribution as the number of trials, n , tends to infinity and the probability of success, p , tends to 0. The Poisson distribution has probability mass function $P(x|\lambda) = \mathcal{P}(x; \lambda)$ where

$$\mathcal{P}(x; \lambda) = \frac{\lambda^x}{x!} \exp(-\lambda). \quad (2.64)$$

The parameter λ is both the mean and the variance of the distribution.

Two important properties of the Poisson distribution, which we will make use of in later chapters, are the *infinite divisibility* property and its connections to the multinomial distribution.

The infinite property states that if $X_i \sim \mathcal{P}(\lambda_i)$, for $i \in 1, \dots, N$, then $Y = \sum_{i=1}^N X_i$ is distributed $Y \sim \mathcal{P}\left(\sum_{i=1}^N \lambda_i\right)$. Thus we can represent any Poisson distributed random variable as a sum of Poisson distributed random variables.

For Poisson distributed random variables it also holds that if $X_i \sim \mathcal{P}(\lambda_i)$, for $i \in 1, \dots, N$ and $Y = \sum_{i=1}^N X_i$, then

$$X_{1:N}|Y \sim \mathcal{M}\left(Y, \frac{\lambda_{1:N}}{\sum_{i=1}^N \lambda_i}\right). \quad (2.65)$$

The conjugate prior for the Poisson distribution is the Gamma distribution:

$$\mathcal{G}(x; \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} \exp(-\beta x). \quad (2.66)$$

Given an observed set of counts $\mathcal{D} = \{x^{(1)}, \dots, x^{(N)}\}$, the posterior distribution of λ is therefore a Gamma distribution, $p(\lambda|\mathcal{D}) = \mathcal{G}(\lambda; \alpha^+, \beta^+)$, with parameters $\alpha^+ = \alpha + \sum_{n=1}^N x^{(n)}$ and $\beta^+ = \beta + N$.

If we let $\beta = \frac{1-p}{p}$ and compute the probability of a data point x under a Poisson-Gamma hierarchical prior as follows:

$$P(x; r, p) = \int_0^\infty \mathcal{P}(x; \lambda) \mathcal{G}\left(\lambda; r, \frac{1-p}{p}\right) d\lambda \quad (2.67)$$

$$= \frac{\mathcal{P}(x; \lambda) \mathcal{G}(\lambda; r, \frac{1-p}{p})}{\mathcal{G}\left(\lambda; r + x, \frac{1-p}{p} + 1\right)} \quad (2.68)$$

$$= \frac{\Gamma(r+x)}{x! \Gamma(r)} p^x (1-p)^r \quad (2.69)$$

$$= \binom{x+r-1}{x} p^x (1-p)^r \quad (2.70)$$

$$\triangleq \mathcal{NB}(x; p, r),$$

then we obtain Equation 2.70, which may be recognised as the *negative binomial* distribution. The negative-binomial distribution is a useful model for count data that is *over disbursed*, which means the variance is greater than the mean.

	$\mathbf{t}(\mathbf{x})$	$\boldsymbol{\theta}$	$\boldsymbol{\eta}$	$\psi(\boldsymbol{\theta})$	$c(\mathbf{x})$
$\mathcal{P}$	x	$\log(\lambda)$	λ	$\exp(\theta)$	$-\log(x!)$
$\mathcal{G}$	$[\log x, \quad x]$	$[\alpha - 1, \quad -\beta]$	$[\psi(\alpha) - \log(\beta), \quad -\frac{\alpha}{\beta}]$	$\log \Gamma(\theta_1 + 1) - (\theta_1 + 1) \log(-\theta_2)$	0
$\mathcal{NB}$	x	$\log p$	$\frac{pr}{1-p}$	$-r \log(1 - e^\theta)$	$\log \binom{x+r-1}{x}$

Table 2.3: Natural and expectation parameters for the Poisson, gamma and negative binomial (with known r) distributions.

2.3.5 Comparing distributions

When constructing inference algorithms, it is often useful to be able to compare probability distributions. Divergence measures allow such comparisons. Although divergence measures can be thought of as distance measures, they are not necessarily true metrics—they may not obey the triangle inequality—nor are they necessarily symmetric in their arguments. An important divergence measure we will use frequently in this work is the KL-divergence, also known as *relative entropy* amongst

other names. For continuous densities $p(x)$ and $q(x)$ it is defined as:

$$\text{KL}(p(\mathbf{x})||q(\mathbf{x})) = \int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x}. \quad (2.71)$$

KL-divergence is always non-negative and is intimately related to *information theory* (MacKay 2005). For Gaussian distributions the KL-divergence can be computed in closed-form according to:

$$\text{KL}(\mathcal{N}_1||\mathcal{N}_2) = \frac{1}{2} \left[\text{tr}(\Sigma_2^{-1}\Sigma_1) - (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \Sigma_2^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) - M + \log \frac{|\Sigma_2|}{|\Sigma_1|} \right], \quad (2.72)$$

for distributions $\mathcal{N}_1, \mathcal{N}_2$, with parameters $\boldsymbol{\mu}_1, \Sigma_1$ and $\boldsymbol{\mu}_2, \Sigma_2$ respectively.

2.4 Stochastic processes

Stochastic processes are collections of indexed random variables. Typically the index is a continuous or discrete scalar quantity, usually *time*, although more generally the index may be a vector valued quantity. Stochastic processes are a useful statistical tool for describing *dynamic systems* and allow us to construct models that capture temporal or spatiotemporal dependencies. Stochastic processes are also relevant in the context of Bayesian non-parametrics (BNP).

BNP is concerned with the development of statistical models that can *grow* as more data is observed. The term “non-parametric” means that the model has potentially an *infinite* number of parameters and not zero parameters. Stochastic processes can be adapted to allow the construction of infinite dimensional prior distributions. This is important since, in a Bayesian analysis, the prior limits what may be learned. If the prior is parametric, i.e. finite dimensional, it may not be rich enough to capture all the information in the data. By using infinite dimensional priors, BNP models have larger *coverage* than equivalent parametric models, meaning they place probability mass on a wider variety of possible parameter configurations and are therefore more likely to include configurations that are appropriate for a particular observed dataset.

2.4.1 Gaussian processes

Gaussian Processes (GPs) (Rasmussen and Williams 2006) are an important class of non-parametric priors used extensively throughout this work. The GP is the infinite dimensional extension of the (finite dimensional) Gaussian distribution. Gaussian processes are typically used as a prior for continuous functions in BNP models.

Formally, $f(t)$ is a Gaussian process $f(t) \sim \mathcal{GP}(\mu(t), \Sigma(t, t'))$ if any finite dimensional vector

$$\mathbf{f} = [f(t^{(1)}), \dots, f(t^{(N)})]^\top, \quad (2.73)$$

is distributed according to a multivariate Gaussian $\mathcal{N}(\mathbf{f}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ for all possible choices of $t^{(1)}, \dots, t^{(N)}$ and for all $N \geq 1$, where $\boldsymbol{\mu} = [\mu(t^{(1)}), \dots, \mu(t^{(N)})]^\top$ and the *Gram-matrix*, $\boldsymbol{\Sigma}$, is:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \Sigma(t^{(1)}, t^{(1)}) & \dots & \Sigma(t^{(1)}, t^{(N)}) \\ \vdots & \ddots & \vdots \\ \Sigma(t^{(N)}, t^{(1)}) & \dots & \Sigma(t^{(N)}, t^{(N)}) \end{bmatrix}. \quad (2.74)$$

In contrast to the finite mean and covariance of the Gaussian distribution, the Gaussian process has a *mean function*, μ , and a *covariance function*, Σ , also known as a *kernel-function*. A valid kernel function, K , is one for which any Gram-matrix is positive semi-definite, i.e. for any vector $\mathbf{g} \in \mathbb{R}^N$:

$$\mathbf{g}^\top \mathbf{K} \mathbf{g} = \sum_{i=1}^N \sum_{j=1}^N g_i g_j [\mathbf{K}]_{i,j} \geq 0, \quad (2.75)$$

or, more generally, for any square integrable function $g(t)$:

$$\iint g(t) K(t, t') g(t') dt dt' \geq 0. \quad (2.76)$$

A kernel function is called *stationary* if it is a function of only the difference between its inputs $K(t, t') = k(t - t')$; it is called *isotropic* if it is a function of only the *magnitude* of the difference of its inputs $K(t, t') = k(|t - t'|)$. A popular (isotropic) kernel-function is the exponentiated quadratic (also known as squared-exponential) kernel function:

$$K(t, t') = \gamma \exp \left(-\frac{(t - t')^2}{2\alpha} \right). \quad (2.77)$$

where γ is the *output-length-scale* and α is the *input-length-scale*. Increasing the input-length-scale causes points to become more highly correlated and the function values to change more slowly. A short input-length-scale (a small value of α) in contrast allows the function to adapt more quickly. Thus the input-length-scale magnifies the function in the x -axis, while the output-length-scale magnifies the function in the y -axis. Draws from a GP with the exponentiated quadratic kernel function are shown in Figure 2.1.

Gaussian processes over R -dimensional spaces, denoted $f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), \Sigma(\mathbf{x}, \mathbf{x}'))$, $\mathbf{x} \in \mathbb{R}^R$, can be constructed using multivariate kernel functions, for example the exponentiated quadratic kernel:

$$K(\mathbf{x}, \mathbf{x}') = \gamma \prod_{r=1}^R \exp\left(-\frac{(x_r - x'_r)^2}{2\alpha_r}\right). \quad (2.78)$$

A standard application of Gaussian processes is *Gaussian process regression*. A typical regression problem is to infer the (posterior) distribution of f using a dataset, $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N$, consisting of N coordinate and scalar valued pairs, where the scalar valued data points are assumed to be corrupted with isotropic Gaussian noise,

$$\mathbf{y} \sim \mathcal{N}(\mathbf{f}, \mathbb{I}\sigma^2), \quad (2.79)$$

and where $\mathbf{y} = [y^{(1)}, \dots, y^{(N)}]^\top$. Due to the conjugacy of the Gaussian distribution, the function conditioned on set of data points, $f(\mathbf{x})|\mathcal{D} \sim \mathcal{GP}(f(\mathbf{x}); \mu^+(\mathbf{x}), \Sigma^+(\mathbf{x}, \mathbf{x}'))$, is also a Gaussian process with mean and covariance functions:

$$\mu^+(\mathbf{x}) = \mu(\mathbf{x}) + \mathbf{k}_{xn}[\mathbf{K}_{nn} + \mathbb{I}\sigma^2]^{-1}[\mathbf{y} - \boldsymbol{\mu}_n] \quad (2.80)$$

$$\Sigma^+(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') - \mathbf{k}_{xn}[\mathbf{K}_{nn} + \mathbb{I}\sigma^2]^{-1}\mathbf{k}_{nx'}, \quad (2.81)$$

where $\boldsymbol{\mu}_n = [\mu(\mathbf{x}^{(1)}), \dots, \mu(\mathbf{x}^{(N)})]^\top$, $\mathbf{k}_{xn} = [K(\mathbf{x}, \mathbf{x}^{(1)}), \dots, K(\mathbf{x}, \mathbf{x}^{(N)})]$ and $\mathbf{k}_{nx'} = [K(\mathbf{x}^{(1)}, \mathbf{x}'), \dots, K(\mathbf{x}^{(N)}, \mathbf{x}')]^\top$.

Figures 2.2 and 2.3 show draws from the posterior distribution conditioned on one and two points. In these figures the observation noise, σ^2 , is 0 and therefore the

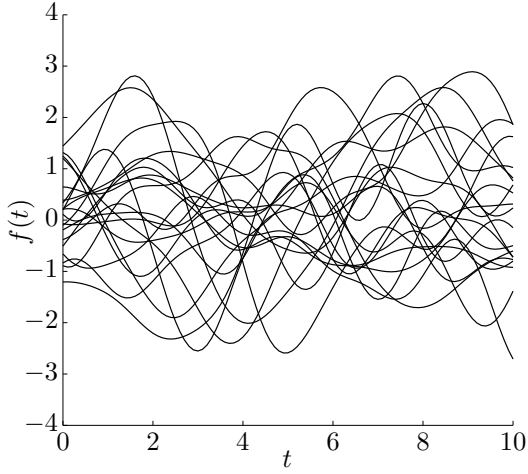


Figure 2.1: 20 draws from a Gaussian process with an Exponentiated Quadratic Kernel with parameters $\gamma = 1$ and $\alpha = 1$.

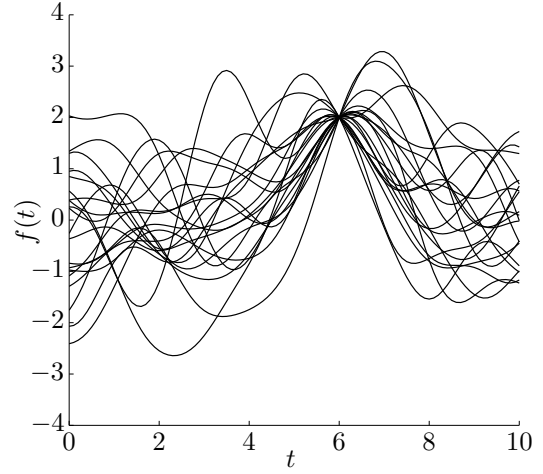


Figure 2.2: Draws from a Gaussian process conditioned on the point $(6, 2)$

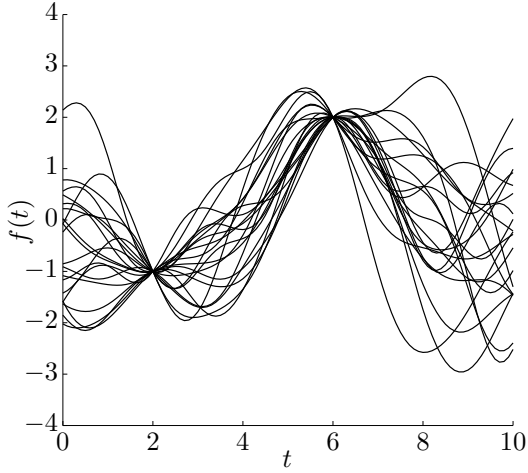


Figure 2.3: Draws from a Gaussian process conditioned on the points $(6, 2)$ and $(2, -1)$.

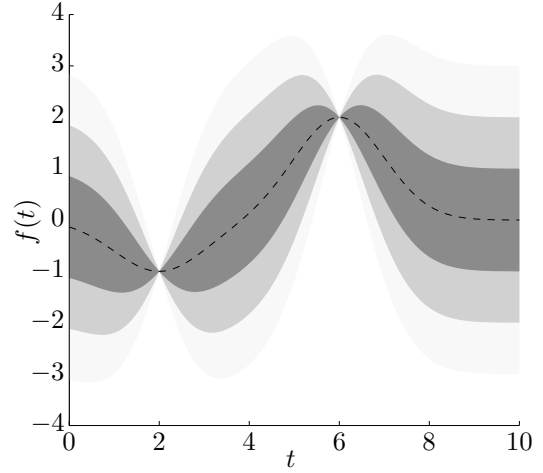


Figure 2.4: Posterior distribution of the Gaussian process conditioned on the points $(6, 2)$ and $(2, -1)$ showing the dashed mean function and shaded 1, 2 and 3 standard deviation regions.

all functions pass through the points $(6, 2)$ and $(2, -1)$ exactly. Figure 2.4 shows the posterior 1, 2 and 3 standard deviation regions. The probability that f passes through a third point in the dark, medium or light grey regions, calculated using the Gaussian cumulative distribution function, is $\sim 68.3\%$, $\sim 27.2\%$ or $\sim 4.3\%$.

2.4.2 Poisson processes

Poisson processes are counting processes. Poisson processes are used extensively in event modelling, population growth models and queuing theory. A stochastic process, $X(t)$, is a *homogeneous* Poisson process if:

- $X(0) = 0$
- $X(t) \sim \mathcal{P}(\lambda t), \forall t > 0$
- It has stationary, independent increments.

The parameter λ is the *intensity* of the process; the higher the intensity the faster the count increases. A homogeneous Poisson process with intensity λ can be constructed as a *renewal* process with holding times (i.e. the time between jumps) distributed according to an exponential distribution with parameter λ . As shown in Figure 2.5, draws from a homogeneous Poisson process are monotonically increasing piecewise continuous functions.

Instead of considering the count at time t , $X(t)$, we will consider an alternative representation of a Poisson process as set of points $\mathcal{X} = \{t^{(1)}, \dots, t^{(N)}\}$, where each point $t^{(n)}$ co-insides with a jump of $X(t)$. Such a set of points is known as *Poisson point process* although we will use Poisson process and Poisson point process interchangeably. The correspondence between the counting process and the point process is shown in Figure 2.6. The point representation of the Poisson process is more flexible and it allows us to represent point processes over higher dimensional spaces, $\mathcal{X} \subset \mathbb{R}^R$, for which there is no counting process equivalent. Figure 2.7 shows a Poisson process, $\mathcal{X} = \{\mathbf{x}^{(n)} \in \mathcal{X}\}_{n=1}^N$, on a two dimensional region $\mathcal{X} = [0, 10]^2$. For disjoint subsets $\mathcal{X}_i, \mathcal{X}_j$, the counts $N(\mathcal{X}_i), N(\mathcal{X}_j)$ are independent. This independence is due to the completely independent nature of points in a Poisson process (Kingman 1993). An inhomogeneous Poisson process differs from a homogeneous Poisson process by replacing the constant rate λ with a spatially or temporally varying function $\lambda(\mathbf{x}) : \mathcal{X} \rightarrow \mathbb{R}^+$, referred to as the

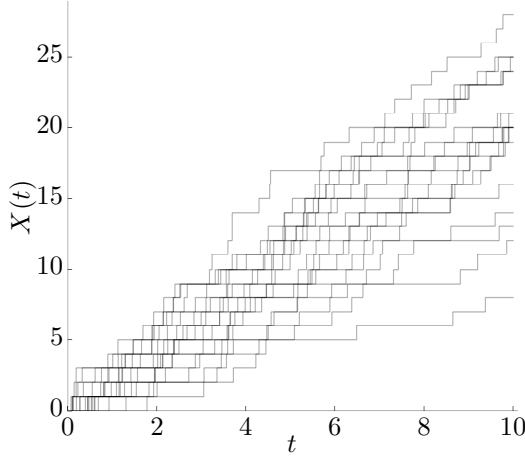


Figure 2.5: 20 draws from a homogeneous Poisson process with parameter $\lambda = 2$.

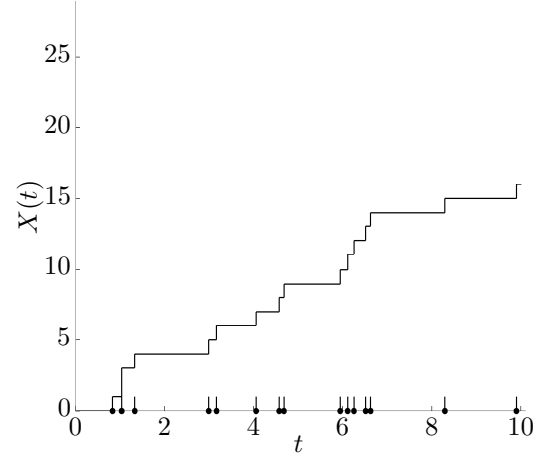


Figure 2.6: A draw from a Poisson process and the corresponding Poisson point process.

intensity or *rate* function³. The number of points, $N(\mathcal{X}_i)$, found in any subregion $\mathcal{X}_i \subset \mathbb{R}^R$ is Poisson distributed with parameter $\lambda_{\mathcal{X}_i} \triangleq \int_{\mathcal{X}_i} \lambda(\mathbf{x}) \, d\mathbf{x}$. Figure 2.8 shows an inhomogeneous Poisson process on $\mathcal{X} = [0, 10]^2$. The contour lines indicate the expected number of points per unit area over the domain.

Draws from an inhomogeneous Poisson process can be generated using a form of *rejection sampling* known as *thinning* (Lewis and Shedler 1979). First we find a value, $\lambda^* > \lambda(\mathbf{x}) \, \forall \mathbf{x} \in \mathcal{X}$, which upper bounds the intensity function over the region. We then draw $N \sim \mathcal{P}(\lambda^*|\mathcal{X}|)$ points and distribute them uniformly on $\mathcal{X}$. Next for each point, $\mathbf{x}^{(n)}$, we draw an *auxiliary* variable $r^{(n)} \sim \mathcal{U}([0, \lambda^*])$. This results in a homogeneous Poisson process on $\mathcal{X} \times [0, \lambda^*]$. Finally we form the subset of points for which $r^{(n)} < \lambda(\mathbf{x}^{(n)})$. This process is shown Figure 2.9; the red points are “thinned” and the remaining blue points form a draw from inhomogeneous Poisson process with intensity given by $\lambda(\mathbf{x})$.

An inhomogeneous Poisson process with a random intensity function is known as a doubly stochastic inhomogeneous Poisson process or *Cox process*. The stochastic process driving the intensity function is not part of the definition, however typically Hidden Markov Models (HMMs) and Gaussian processes are used. In this work

³These names are used interchangeably throughout this work.

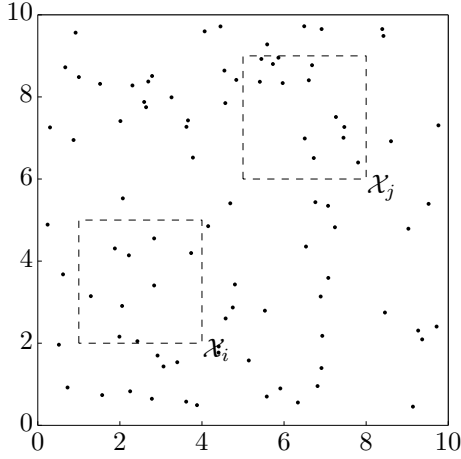


Figure 2.7: A draw from a homogeneous Poisson point process on $[0, 10]^2$ with intensity $\lambda = 1$. The counts $N(\mathcal{X}_i)$, $N(\mathcal{X}_j)$ in the regions $\mathcal{X}_i$, $\mathcal{X}_j$ are independent.

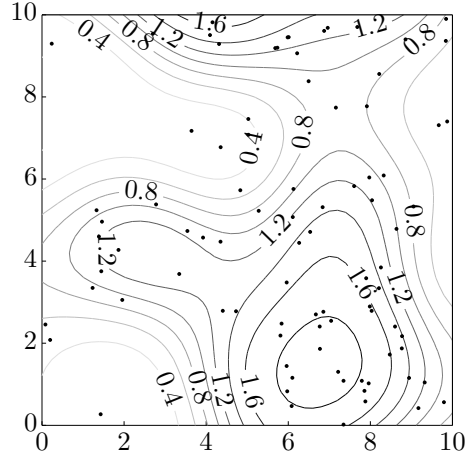


Figure 2.8: A draw from an inhomogeneous Poisson point process on $[0, 10]^2$. The contour lines indicate the intensity of the point process across the domain.

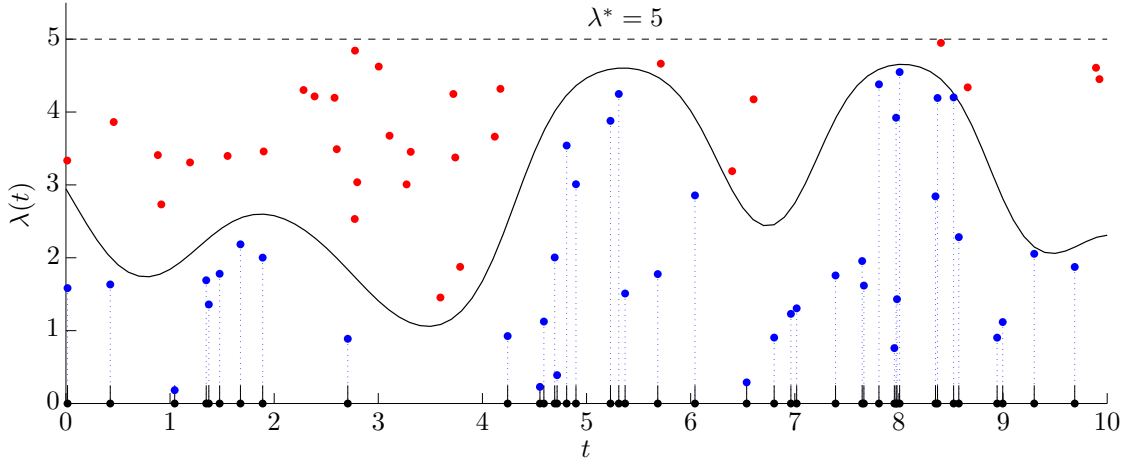


Figure 2.9: An inhomogeneous Poisson point process with intensity function constructed by “thinning” a homogeneous Poisson point process using auxiliary variables.

we will use make extensive of use of Gaussian processes to drive Cox process intensity functions. When Gaussian processes are used to modulate Poisson process intensities, a *link* function must be used to ensure the resulting intensity is positively valued. Suitable link functions include the exponential function and sigmoid-function squashing function,

$$\sigma(x) = \frac{1}{1 + \exp(-x)}, \quad (2.82)$$

resulting in the Log-Gaussian Cox Process (LGCP) and the Sigmoid-Gaussian Cox Process (SGCP) respectively. Figures 2.11 and 2.12 show the result of transforming

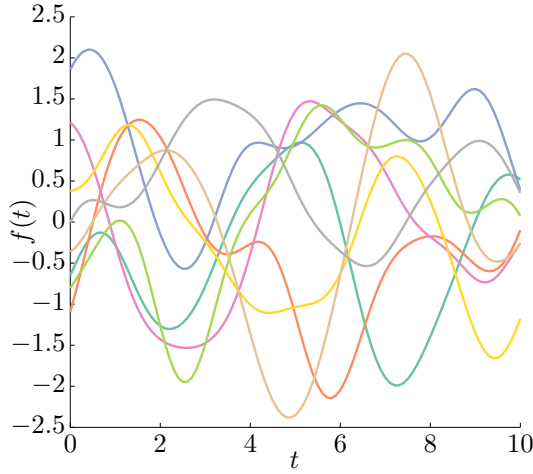


Figure 2.10: Gaussian process distributed random functions used to construct Cox process intensity functions.

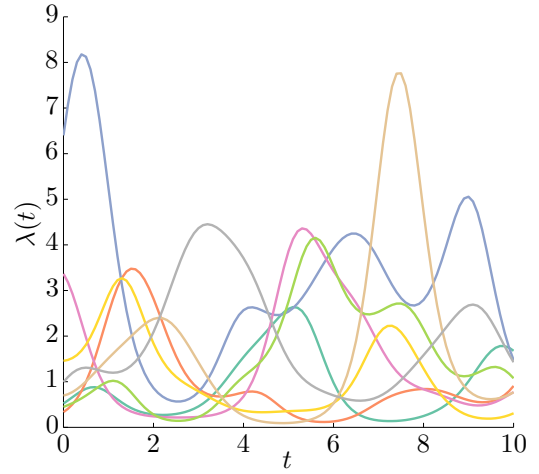


Figure 2.11: Log-Gaussian Cox process (LCGP) intensity functions:
 $\lambda(t) = \exp(f(t))$

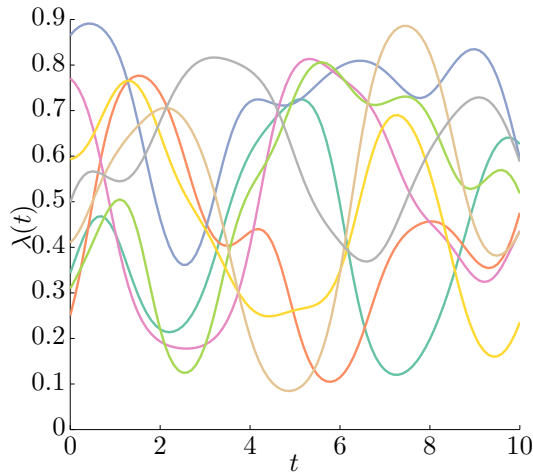


Figure 2.12: Sigmoid-Gaussian Cox process (SGCP) intensity functions:
 $\lambda(t) = \lambda^* \sigma(f(t))$, $\lambda^* = 1$.

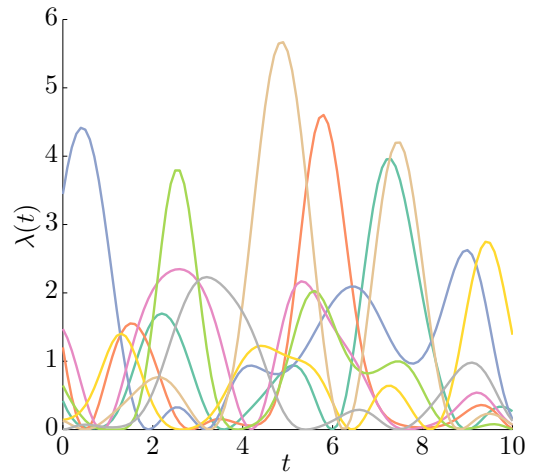


Figure 2.13: Square-transformed-Gaussian Cox process intensity functions: $\lambda(t) = f^2(t)$.

GP distributed random functions with the exponential and sigmoid functions. Another possible transformation is the square-transform shown in Figure 2.13. While the exponential and sigmoid link functions have been used extensively, for modelling both positive classification likelihoods and point process intensity functions, the square-transform has not been as widely considered (Gunter, Osborne, et al. 2014). Constructing a rate function as a zero-mean squared⁴ independent

⁴ Squared, zero-mean Gaussian distributed random variables follow a gamma distribution. As described in §2.3.4 a gamma prior integrated against a Poisson likelihood results in a negative-binomial distribution. This suggests a link between the permanental point process and the negative-binomial distribution. However this is beyond the scope of this thesis. See McCullagh

Gaussian processes (or the sum of independent GP) results in a particular sub-class of Cox process known as the *permanental* point processes; see Eisenbaum and Kaspi (2009), Hough et al. (2006) .

The square-transform has some advantages compared to the sigmoid and exponential link functions. It has a dynamic range that fits between these alternatives and can more easily model very low values, including numerical zero, as shown by Figure 2.13. Square-transformed GP Cox processes can therefore range from nearly homogeneous to strongly clustered—the latter is a characteristic of permanental point processes.

The conditional probability density of a set of points, $\mathcal{D} = \{\mathbf{x}^{(n)}\}_{n=1}^N$, given an intensity function, λ , is the following *Janossi* density (Daley and Vere-Jones 2003):

$$p(\mathcal{D}|\lambda) = \exp \left\{ - \int_{\mathcal{X}} \lambda(\mathbf{x}) \, d\mathbf{x} \right\} \prod_{n=1}^N \lambda(\mathbf{x}^{(n)}). \quad (2.83)$$

The density is arrived at by considering smaller and smaller regions $d\mathbf{x}$ until these regions contain only a single point each, and is the product of all the infinitely small regions that do not contain a point and the N regions that contain a single point.

Using Bayes' rule, the posterior distribution of the rate function conditioned on the data, is

$$p(\lambda|\mathcal{D}) = \frac{p(\lambda) \exp \left\{ - \int_{\mathcal{X}} \lambda(\mathbf{x}) \, d\mathbf{x} \right\} \prod_{n=1}^N \lambda(\mathbf{x}^{(n)})}{\int p(\lambda) \exp \left\{ - \int_{\mathcal{X}} \lambda(\mathbf{x}) \, d\mathbf{x} \right\} \prod_{n=1}^N p(\lambda(\mathbf{x}^{(n)})) \, d\lambda} \quad (2.84)$$

which is often described as *doubly-intractable* because of the nested integral in the denominator.

Models containing doubly intractable integrals, most notably Cox process models, are hard inference problems. Although the Cox process is an attractive model for many types of spatiotemporal data, the doubly-intractable integral is no insignificant obstacle to posterior inference. Efficient inference for this type of model is therefore of widespread interest and the related body of literature has been growing in recent years. A review of this literature is presented in Chapter 8. In later chapters we

and Moller (2006) for more details.

will make a significant contribution to this field by showing how square-transformed Gaussian processes can be used to construct efficient inference algorithms for Cox processes. In contrast with the previous state-of-the-art, the method presented scales linearly in the number of observations and does not require discretisation of the input domain.

2.4.3 The Dirichlet process

A Dirichlet process (Ferguson 1973), is a random probability distribution on a space Θ . If $G \sim \mathcal{DP}(\alpha, G_0)$ is a draw from a Dirichlet Process (DP) with *base measure* G_0 , support Θ and *concentration parameter* α , and if $\{S_1, \dots, S_N\}$ is a partition of Θ , then the probability mass assigned to each partition is $G(S_n)$ and the distribution over all partitions is

$$(G(S_1), \dots, G(S_N)) \sim \mathcal{D}(\alpha G_0(S_1), \dots, \alpha G_0(S_N)). \quad (2.85)$$

In other words, all finite marginal distributions on the space Θ are Dirichlet distributed. If G_0 is continuous then, with probability 1, a draw from the process will be a countably infinite collection of *atoms* $\{(\theta^{(n)}, w^{(n)})\}_{n=1}^{\infty}$, with $w^{(n)} \in [0, 1]$, $\sum_{n=1}^{\infty} w^{(n)} = 1$ and $\theta^{(n)} \in \Theta$ and G is defined by:

$$G(S) = \sum_{n=1}^{\infty} w^{(n)} \mathbb{1}\{\theta^{(n)} \in S\}. \quad (2.86)$$

Samples drawn from a DP are shown in Figure 2.14. These samples are generated by sampling values $w^{(n)}$ directly from the marginal of the DP and associating with each a value, $\theta^{(n)}$, drawn from G_0 . Constructive processes that allow samples to be drawn from this marginal distribution include the Chinese Restaurant Process (CRP), the Polya Urn Scheme (PUS) and the stick-breaking process.

The CRP is a metaphor for a procedure in which customers enter a restaurant and sit at a table. The probability that a customer sits at a particular table is proportional to the number of customers already sitting at the table and the probability that the customer sits at a new table is proportional to α . The PUS is essentially the same

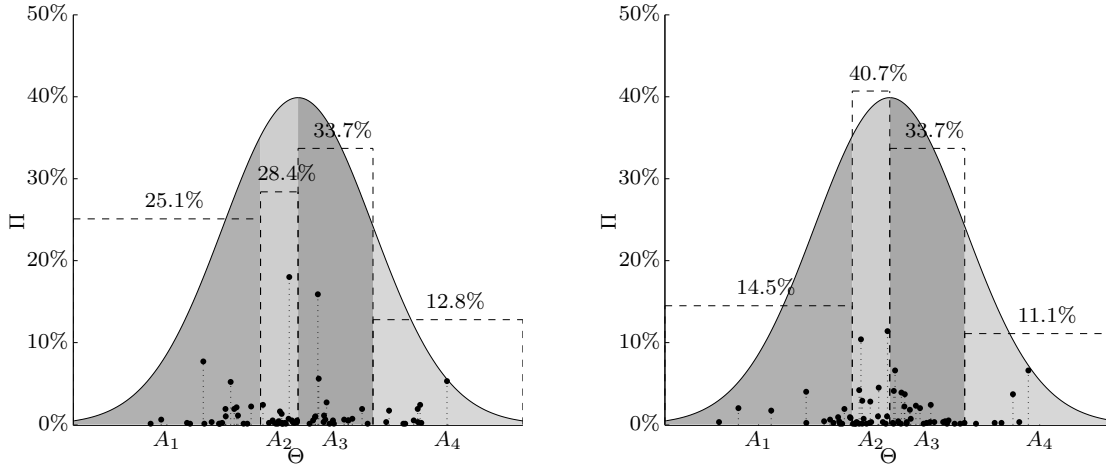


Figure 2.14: Draws from a Dirichlet process with base measure $G_0 = \mathcal{N}(0, 1)$ and concentration parameter $\alpha = 20$. The heights of dotted lines—which indicate the distribution of probability mass associated with the atoms—were sampled using a Chinese restaurant process with 1000 “customers”. The dashed lines denote the total probability mass of the atoms in each shaded region. For comparison, the measures of these regions under the G_0 are $G_0(A_1) = 30.7$, $G_0(A_2) = 19.1$, $G_0(A_3) = 34.1$ and $G_0(A_4) = 15.7$.

idea stated differently. Place α black balls in a bag. Draw a ball: if the ball is black paint a new ball with a random colour and place both balls back in the bag; if the ball is coloured, paint another ball of with the same colour and place both balls back in the bag. The stick breaking construction is different, but conceptually just as simple. Take a stick of length 1 and recursively break fractions of size $v \sim \mathcal{B}(1, \alpha)$ off the length of the stick. An example of this process is shown in Figure 2.15. In either case the fractions of balls of each colour or the fraction number of customers or the lengths of the stick pieces are the values $w^{(n)}$.

The remarkable property of these constructions is that they are *exchangeable*. A sequence of random variables $X^{(1)}, \dots, X^{(N)}$, is exchangeable if for any permutation π of the numbers $1, \dots, N$, the joint distribution, $p(X^{(1)}, \dots, X^{(N)})$ is the same as the joint distribution $p(X^{(\pi(1))}, \dots, X^{(\pi(N))})$.

Sequences of independent and identically distributed random variables are always exchangeable, however exchangeable random variables are not always independent and identically distributed. The sequence of random variables produced by these schemes are not independent and identically distributed, but they are exchangeable.

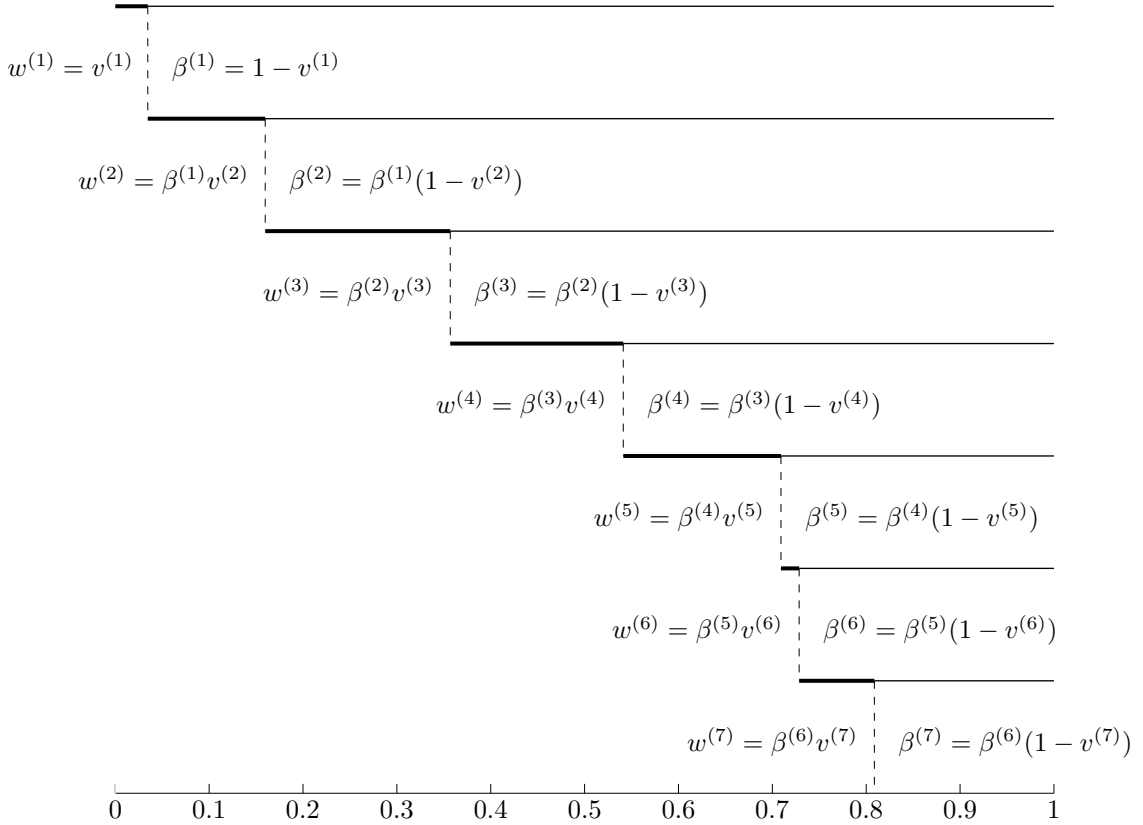


Figure 2.15: A draw from a stick breaking process with concentration parameter $\alpha = 3.0$. The fraction of the stick broken at each iteration, $w^{(n)}$, is indicated by the length of the thick black lines. $v^{(n)}$ are drawn i.i.d. from $\mathcal{B}(1, \alpha)$.

Consider the probability of observing 3 blue balls, 2 red balls and 1 green ball drawn from a PUS with α black balls to begin with. It might seem from the construction that the number of red balls drawn may be dependent on whether a red ball was drawn first. However this is not the case, as shown by the following example:

$$p(\textcolor{red}{r}, \textcolor{blue}{b}, \textcolor{red}{r}, \textcolor{green}{g}, \textcolor{blue}{b}, \textcolor{blue}{b}) = \frac{\alpha}{\alpha} \times \frac{\alpha}{\alpha+1} \times \frac{\textcolor{red}{1}}{\alpha+2} \times \frac{\alpha}{\alpha+3} \times \frac{\textcolor{blue}{1}}{\alpha+4} \times \frac{\textcolor{blue}{2}}{\alpha+5} \quad (2.87)$$

$$\begin{aligned} &= \frac{\alpha}{\alpha} \times \frac{\alpha}{\alpha+1} \times \frac{\textcolor{red}{\alpha}}{\alpha+2} \times \frac{\textcolor{blue}{1}}{\alpha+3} \times \frac{\textcolor{red}{1}}{\alpha+4} \times \frac{\textcolor{blue}{2}}{\alpha+5} \\ &= p(\textcolor{green}{g}, \textcolor{blue}{b}, \textcolor{red}{r}, \textcolor{blue}{b}, \textcolor{red}{r}, \textcolor{blue}{b}), \end{aligned} \quad (2.88)$$

and therefore the order in which the colours are observed does not affect probability of a particular mix of colours.

Exchangeability is a desirable property for statistical models, since it confirms that arbitrary labels used to specify models do not affect the underlying probability

distribution. In this work we will use the bracketed superscript $^{(n)}$ to indicate that the items to which they are attached form an exchangeable sequence with an arbitrarily assigned numbering.

The DP has a useful conjugacy property. If we observe atoms drawn from the process, $\mathcal{D} = \{\theta^{(1)}, \dots, \theta^{(N)}\}$, then we can obtain a posterior distribution over the process, which is a mixture of discrete atoms—the observed points—and the base distribution:

$$p(\theta|\mathcal{D}) = \frac{1}{N + \alpha} \sum_{n=1}^N \delta(\theta - \theta^{(n)}) + \frac{\alpha}{N + \alpha} G_0(\theta). \quad (2.89)$$

Since the Dirichlet Process is atomic, draws from it contain repeated values. While this is not a very realistic model for many types of data, the Dirichlet distribution is very useful for constructing non-parametric mixture models—mixture models with a potentially infinite number of components. In this case the values drawn from the DP are the parameters of, typically, an exponential family distribution⁵. In this case, the base measure G_0 is usually chosen to be conjugate to the distribution parametrised by θ .

An alternative representation of the CRP is a potentially infinite matrix, $\mathbf{Z}$, of 1's and 0's denoted $\mathbf{Z} \sim \text{CRP}(\alpha)$ and shown in Figure 2.16. This matrix representation is useful as it allows models to be described using simple linear algebra, for example factor models described in Chapter 4. If we allow multiple entries in each row of this matrix we obtain an infinite dimensional *feature model*. Whereas mixture models assign data points to a single cluster, feature models allow data points to be assigned to multiple clusters. A constructive process for infinite dimensional feature models is the Indian Buffet Process (IBP) (Griffiths and Ghahramani 2006), which can be viewed as an extension of the DP. The IBP metaphor refers to a restaurant in which diners choose dishes (i.e. data points are assigned features) in proportion to how many customers have already chosen the dish, plus a random number of new dishes with distribution $\mathcal{P}(\alpha/N)$. As shown in Figure 2.17, the IBP is a distribution over a binary matrix with N rows and an potentially infinite number of columns.

⁵Hence the reason we have denoted draws from the DP using θ .

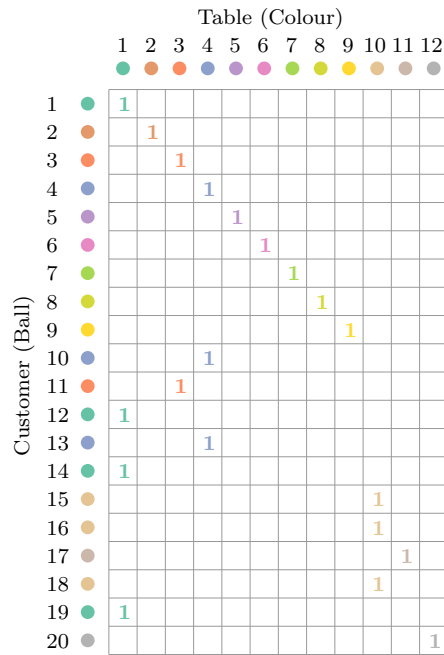


Figure 2.16: Matrix representation of a draw from a Chinese Restaurant Process. In the metaphor, the rows represent the customers and the columns represent the table. (Alternatively they are ball & colour in the PUS analogy.)

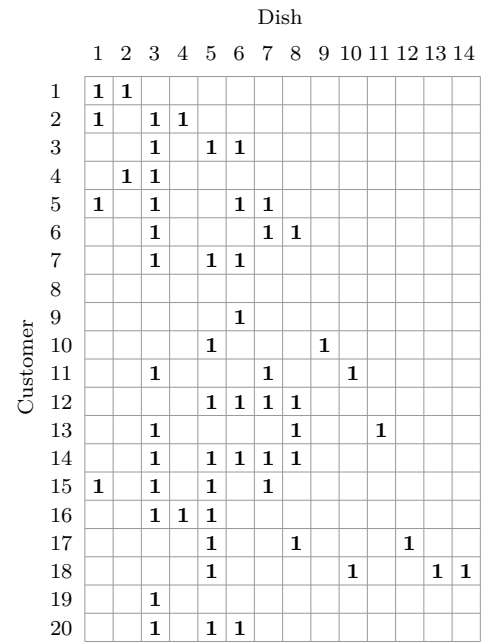


Figure 2.17: Matrix representation of a draw from an Indian Buffet Process. In the metaphor, the rows of the matrix represent customers and the columns dishes. A '1' in the matrix indicates that the customer has 'chosen' that dish.

As in the case of the CRP, draws from the IBP form an exchangeable sequence and therefore we can re-order the rows of Figure 2.17 without changing the probability of the matrix. In this thesis we will not develop models using the IBP, however they have been used to build Bayesian non-parametric models of complex networks, some of which are described in Chapter 4.

3

Approximate inference algorithms

In very simple conjugate exponential family models, posterior Bayesian distributions can be computed analytically. In more complex models this is not the case. Inference in complex models requires the numerical computation of high-dimensional integrals, which are intractable due to the *curse-of-dimensionality*. Approximate inference algorithms must therefore be used. Even when posterior inference is analytic, such as in Gaussian Process (GP) regression, approximation may be required for reasons of *computational complexity*. Approximate Bayesian inference schemes (Bishop 2007) include:

- Sampling (Neal 2010)
- Quadrature
- Variational Bayes (VB)
- Expectation Propagation (EP) (Minka 2001)
- Laplace Approximation.

In this chapter we will focus on sampling and variational inference algorithms that we will use in later chapters.

3.1 Sampling

Sampling algorithms aim to approximate the posterior distribution of the parameter(s) as a mixture of delta functions

$$p(\theta|\mathcal{D}) \approx \frac{1}{T} \sum_{t=1}^T \delta(\theta - \theta^{\{t\}}), \quad (3.1)$$

where $\theta^{\{1\}}, \dots, \theta^{\{T\}}$ are drawn from the unnormalised joint distribution $\tilde{p}(\theta)$, which is proportional to the posterior distribution:

$$\tilde{p}(\theta) \triangleq p(\mathcal{D}, \theta) = \frac{1}{Z} p(\theta|\mathcal{D}). \quad (3.2)$$

Sampling algorithms must contend with not only a potentially high dimensional θ , but must also be unaffected by the unknown scaling constant Z .

The aim of Markov-Chain Monte-Carlo (MCMC) is to generate a sequence of samples, $\theta^{\{1\}} \rightarrow \theta^{\{2\}}, \dots, \rightarrow \theta^{\{T\}}$. Each sample is generated using the previous sample using a *Markov transition kernel* $k(\theta \rightarrow \theta')$. The transition kernel defines the probability of transitioning from θ to θ' and therefore

$$\int k(\theta \rightarrow \theta') d\theta' = 1. \quad (3.3)$$

Samples generated using MCMC are not independent and identically distributed, however by iteratively sampling from the transition kernel it is possible to construct a process that is *asymptotically unbiased*, i.e. one which, if run forever, would converge to the target distribution¹. To ensure the chain has this property, it is sufficient to show the transition kernel satisfies the *detailed balance* condition,

$$p(\theta)k(\theta \rightarrow \theta') = p(\theta')k(\theta' \rightarrow \theta), \quad (3.4)$$

which can roughly be interpreted to mean the probability mass flowing into any region equals the mass flowing out of that region. Integrating both sides of 3.4 with respect to θ' gives

$$p(\theta) = \int p(\theta')k(\theta' \rightarrow \theta) d\theta', \quad (3.5)$$

¹In practice this means one which gives approximately the desired distribution given a sufficiently large number of samples.

and therefore the target distribution is *invariant* to convolution with the transition kernel.

When data is conditioned on multiple unobserved variables we can generate samples $\Theta^{\{t\}} = \{\theta_1^{\{t\}}, \dots, \theta_N^{\{t\}}\}$ using *Gibbs sampling*. In Gibbs sampling, we cycle through each variable, re-sampling each θ_n in turn, while holding the remaining variables, $\Theta_{-n} = \{\theta_1, \dots, \theta_{n-1}, \theta_{n+1}, \dots, \theta_N\}$, constant. Once we have cycled through all the variables we generate a new sample $\Theta^{\{t+1\}}$ as follows:

$$\theta_1^{\{t+1\}} \sim p(\mathcal{D}, \theta_1, \theta_2^{\{t\}}, \dots, \theta_N^{\{t\}}) \quad (3.6)$$

$$\theta_2^{\{t+1\}} \sim p(\mathcal{D}, \theta_1^{\{t+1\}}, \theta_2, \theta_3^{\{t\}}, \dots, \theta_N^{\{t\}}) \quad (3.7)$$

$$\vdots$$

$$\theta_N^{\{t+1\}} \sim p(\mathcal{D}, \theta_1^{\{t+1\}}, \dots, \theta_{N-1}^{\{t+1\}}, \theta_N). \quad (3.8)$$

The algorithm we use to sample θ_i does not need to be the same algorithm we use to sample θ_j . Where the *complete conditional* distribution of a variable θ_n given Θ_{-n} is conditionally conjugate (i.e. in the exponential family), we can compute the temporary local posterior for θ_n analytically and re-sample from the resulting distribution. Otherwise we can use a number of other re-sampling algorithms, including the Metropolis-Hastings (MH), Hamiltonian Monte-Carlo (HMC), slice sampling and Elliptical Slice Sampling (ESS) algorithms described in the following sections.

3.1.1 Metropolis-Hastings

To generate samples using Metropolis-Hastings (MH) (Metropolis et al. 1953; Neal 2010) we first obtain a new candidate value using a *proposal distribution* $q(\theta \rightarrow \theta')$ and then compute the Metropolis acceptance criteria:

$$\alpha(\theta \rightarrow \theta') = \min \left\{ 1, \frac{p(\theta')q(\theta' \rightarrow \theta)}{p(\theta)q(\theta \rightarrow \theta')} \right\}. \quad (3.9)$$

The new value, θ' , is randomly accepted or rejected with probability $\alpha(\theta \rightarrow \theta')$. If the new value is rejected it is replaced by a copy of the previous value, θ . Note

that if $p(\theta)$ is replaced by $\tilde{p}(\theta)$ in Equation 3.9, the acceptance criteria is unaffected since the normalisation constants in the numerator and denominator cancel.

The transition kernel associated with the MH algorithm is

$$k(\theta \rightarrow \theta') = q(\theta \rightarrow \theta')\alpha(\theta \rightarrow \theta') + \mathbb{1}\{\theta = \theta'\} \left[1 - \int q(\theta \rightarrow \theta'')\alpha(\theta \rightarrow \theta'') d\theta'' \right], \quad (3.10)$$

where the term in brackets is the probability that θ' will be rejected and, although it is never necessary to compute it, ensures the kernel satisfies Equation 3.3. In the $\theta = \theta'$ case Equation 3.4 is obviously satisfied, otherwise, in the $\theta \neq \theta'$ case, substituting Equation 3.10 into Equation 3.4 and re-arranging gives:

$$p(\theta)k(\theta \rightarrow \theta') = p(\theta)q(\theta \rightarrow \theta') \min \left\{ 1, \frac{p(\theta')q(\theta' \rightarrow \theta)}{p(\theta)q(\theta \rightarrow \theta')} \right\} \quad (3.11)$$

$$= \min \{p(\theta)q(\theta \rightarrow \theta'), p(\theta')q(\theta' \rightarrow \theta)\} \quad (3.12)$$

$$= p(\theta')q(\theta' \rightarrow \theta) \min \left\{ 1, \frac{p(\theta)q(\theta \rightarrow \theta')}{p(\theta')q(\theta' \rightarrow \theta)} \right\} \quad (3.13)$$

$$= p(\theta')k(\theta' \rightarrow \theta), \quad (3.14)$$

which shows the Metropolis criteria guarantees detailed balance for any suitable proposal distribution.

3.1.2 Hamiltonian Monte-Carlo

Hamiltonian Monte-Carlo (HMC) (Duane et al. 1987; Neal 2010) is an *auxiliary* method useful for sampling multivariate-valued parameters, $\boldsymbol{\theta}$. In HMC the parameter space is augmented with an additional vector $\boldsymbol{\nu}$ and the joint probability distribution is constructed to leave the marginal over $\boldsymbol{\theta}$ unchanged. This is easily achieved by defining the joint density to be factored, i.e. $p(\boldsymbol{\theta}, \boldsymbol{\nu}) \triangleq p(\boldsymbol{\theta})p(\boldsymbol{\nu})$. Sampling from the joint density automatically generates samples from the marginal distribution. Given the current value $\boldsymbol{\theta}$ and a sample $\boldsymbol{\nu} \sim p(\boldsymbol{\nu})$, the aim of HMC is to find new values $\boldsymbol{\theta}', \boldsymbol{\nu}'$ such that their joint probability is equally likely, i.e. $p(\boldsymbol{\theta}, \boldsymbol{\nu}) = p(\boldsymbol{\theta}', \boldsymbol{\nu}')$.

Changing to log space we see that preserving probability is equivalent to preserving energy in a classical mechanical system that has total energy, $H(\boldsymbol{\theta}, \boldsymbol{\nu})$, given by the

Hamiltonian:

$$H(\boldsymbol{\theta}, \boldsymbol{\nu}) \triangleq E(\boldsymbol{\theta}) + K(\boldsymbol{\nu}) \quad (3.15)$$

$$E(\boldsymbol{\theta}) \triangleq -\log p(\boldsymbol{\theta}) + \text{const} \quad (3.16)$$

$$K(\boldsymbol{\nu}) \triangleq -\log p(\boldsymbol{\nu}) + \text{const}, \quad (3.17)$$

where $\boldsymbol{\theta}$ and $\boldsymbol{\nu}$ are the position and momentum of a particle moving in a conservative potential field with potential energy $E(\boldsymbol{\theta})$ and kinetic energy $K(\boldsymbol{\nu})$. The sensible choice for $p(\boldsymbol{\nu})$ is therefore a standard multivariate normal distribution, $\mathcal{N}(\boldsymbol{\nu}; 0, \mathbb{I})$, and so $K(\boldsymbol{\nu}) = \frac{1}{2}\boldsymbol{\nu}^\top \boldsymbol{\nu}$, precisely the kinetic energy of a particle with unit mass ($m=1$) and momentum (and velocity) $\boldsymbol{\nu}$. The time evolution of this dynamic system is governed by Hamilton's equations:

$$\frac{d\boldsymbol{\nu}}{dt} = -\frac{\partial H}{\partial \boldsymbol{\theta}} \quad (3.18)$$

$$\frac{d\boldsymbol{\theta}}{dt} = \frac{\partial H}{\partial \boldsymbol{\nu}}. \quad (3.19)$$

The gradient of H with respect to $\boldsymbol{\theta}$, Equation 3.18, is the force acting particle, which is equal to the change in momentum and acceleration (since $m = 1$), and the gradient of H with respect to $\boldsymbol{\nu}$, Equation 3.19, is the velocity, i.e. the change in position.

The standard numerical algorithm for computing the evolution is the leapfrog algorithm. Starting with $\boldsymbol{\theta} = \boldsymbol{\theta}^{\{0\}}$ we generate a new sample $\boldsymbol{\theta}' = \boldsymbol{\theta}^{\{J\}}$, by taking alternate updates to position and velocity in J intermediate steps:

$$\boldsymbol{\nu}^{\{1\}} := \boldsymbol{\nu}^{\{0\}} - \nabla E(\boldsymbol{\theta}^{\{0\}}) \frac{1}{2} \Delta \quad (3.20)$$

$$\boldsymbol{\theta}^{\{j\}} := \boldsymbol{\theta}^{\{j-1\}} + \boldsymbol{\nu}^{\{j-1\}} \Delta \quad (3.21)$$

$$\boldsymbol{\nu}^{\{j\}} := \boldsymbol{\nu}^{\{j-1\}} - \nabla E(\boldsymbol{\theta}^{\{j\}}) \Delta \quad j \neq J \quad (3.22)$$

$$\boldsymbol{\nu}^{\{J\}} := \boldsymbol{\nu}^{\{J-1\}} - \nabla E(\boldsymbol{\theta}^{\{J\}}) \frac{1}{2} \Delta, \quad (3.23)$$

where the half steps, at the beginning and end of the sequence, ensure that the algorithm can be exactly reversed. Since the numerical integration is not perfect, it is usual to add a MH accept or reject step with probability,

$$a((\boldsymbol{\theta}, \boldsymbol{\nu}) \rightarrow (\boldsymbol{\theta}', \boldsymbol{\nu}')) = \min\{1, \exp(H(\boldsymbol{\theta}, \boldsymbol{\nu}) - H(\boldsymbol{\theta}', \boldsymbol{\nu}'))\}, \quad (3.24)$$

to ensure detailed balance. The important tuning parameters are the number of intermediate steps, J , and the step size, Δ ; these are usually adjusted until there is an even split between accepted and rejected proposals.

3.1.3 Slice sampling

Slice sampling (Neal 2003) is another auxiliary method. In this case the auxiliary parameter, y , is known as the slice level. Starting with the current value of $\theta^{\{n\}}$, we sample the slice level $y^{\{n+1\}} \sim \mathcal{U}([0, \tilde{p}(\theta^{\{n\}})])$. We then sample a new value $\theta^{\{n+1\}} \sim \mathcal{U}(\mathcal{Y}^{\{n+1\}})$ from a set of values, $\mathcal{Y}^{\{n+1\}} = \{\theta : \tilde{p}(\theta) > y^{\{n+1\}}\}$, for which the value of $\tilde{p}$ is greater than the slice level.

The probability of transitioning from θ to θ' via a slice level y and associated slice level set $\mathcal{Y}$ is given by the probability of the slice level, $p(y|\theta)$, multiplied by $p(\theta'|y)$, the probability of selecting θ' from the set $\mathcal{Y}$. The transition kernel is therefore

$$k(\theta \rightarrow \theta') = p(y|\theta)p(\theta'|y) = \frac{1}{\tilde{p}(\theta)} \frac{1}{|\mathcal{Y}|}, \quad (3.25)$$

which when substituted into Equation 3.4, shows that slice sampling maintains detailed balance since

$$p(\theta) \frac{Z}{|\mathcal{Y}|p(\theta)} = p(\theta') \frac{Z}{|\mathcal{Y}|p(\theta')}. \quad (3.26)$$

Slice sampling can be applied to both unimodal and multimodal densities. The main difficulty associated with slice sampling—especially sampling of multimodal densities—is the construction of the set $\mathcal{Y}$.

In most cases $\mathcal{Y}$ cannot be found analytically, so in practice we estimate a new set of values, $\hat{\mathcal{Y}}$, from which we can sample candidate values and reject those not in $\mathcal{Y}$. Furthermore, since we must ensure the algorithm is reversible, we may only sample a new value $\theta' \in \hat{\mathcal{Y}}|\theta$ if it is equally likely, starting from θ' , to sample $\theta \in \hat{\mathcal{Y}}|\theta'$.

Formally stated, to ensure detailed balance, we must sample from the set

$$\mathcal{Y}^* = \{\theta : \theta \in \mathcal{Y} \cap \hat{\mathcal{Y}} \bullet P(\hat{\mathcal{Y}}|\theta) = P(\hat{\mathcal{Y}}|\theta')\}, \quad (3.27)$$

which is the set of acceptable values which are equally likely be contained in the interval, no matter where in the interval we initiate the algorithm.

One method for constructing the set $\hat{\mathcal{Y}}$ is the *stepping-out* algorithm, Algorithm 1. As illustrated in Figure 3.1, the stepping out algorithm grows the interval $\hat{\mathcal{Y}}$ around the current value until both ends of the region lie above the slice level. The method is carefully designed to ensure that $\mathcal{Y}^* = \mathcal{Y} \cap \hat{\mathcal{Y}}$ and therefore any value $\theta' \in \hat{\mathcal{Y}} \cap \mathcal{Y}$ may be accepted. A proof of correctness of the stepping out algorithm may be found in Neal (2003). If another method for constructing $\hat{\mathcal{Y}}$ is used further tests may be required to ensure candidate points are contained in $\mathcal{Y}^*$.

Correctness of the stepping-out slice sampler requires that $\tilde{p}(\theta) > 0$ for all values θ . If this is not the case, the sampler may become stuck in one of the modes, unable “forget” its initial conditions. In this case the Markov-chain is said to be *non-ergodic*. How well a Markov-chain transitions from one region of high probability to another is referred to as *mixing*. Mixing reduces bias and improves accuracy; the more mixing the better. In slice sampling, if the distribution is multi-modal then—depending on the step size relative to the distance between modes—the slice level may need to be close to the minimum level of the function between two modes to transition between them, shown as d in Figure 3.1. If d is very small this may inhibit mixing.

3.1.4 Elliptical Slice Sampling

Elliptical Slice Sampling (ESS) is an adaptation of the slice sampling that is useful for sampling Gaussian distributed random variables with non-conjugate likelihoods (Murray et al. 2010), that is, when the joint density has a Gaussian factor,

$$\tilde{p}(\mathbf{f}) = p(\mathcal{D}, \dots | \mathbf{f}) \mathcal{N}(\mathbf{f}; 0, \Sigma), \quad (3.28)$$

which we will assume, without loss of generality, to have zero mean.

In ESS we pick the slice level $y^{\{n\}}$ *conditioned* on the current value of $\mathbf{f}^{\{n-1\}}$. We then draw a sample $\boldsymbol{\nu}^{\{n\}}$ from the Gaussian factor; together with the previous value

Algorithm 1 The stepping out algorithm

Inputs: $\tilde{p}(\theta)$, $\theta^{\{n\}}$, $y^{\{n+1\}}$, max_iterations , step_size .**Output:** $\hat{\mathcal{Y}} = [\theta_L, \theta_R]$. $u \sim \mathcal{U}([0, 1])$ $\theta_L := \theta^{\{n\}} - u \times \text{step_size}$ $\theta_R := \theta_L + \text{step_size}$ $v \sim \mathcal{U}([0, 1])$ $L = \text{floor}(\text{max_iterations} \times v)$ $R = (\text{max_iterations} - 1) - L$ **while** $L > 0 \ \& \ y^{\{n+1\}} < \tilde{p}(\theta_L)$ **do** $\theta_L := \theta_L - \text{step_size}$ $L := L - 1$ **end while****while** $R > 0 \ \& \ y^{\{n+1\}} < \tilde{p}(\theta_R)$ **do** $\theta_R := \theta_R + \text{step_size}$ $R := R - 1$ **end while**

3.29-3.32:

$$y^{\{n\}} \sim \mathcal{U}([0, p(\mathcal{D}, \dots | \mathbf{f}^{\{n-1\}})]) \quad (3.29)$$

$$a^{\{n\}} \sim \mathcal{U}([a_{\min}, a_{\max}]) \quad (3.30)$$

$$\boldsymbol{\nu}^{\{n\}} \sim \mathcal{N}(0, \boldsymbol{\Sigma}) \quad (3.31)$$

$$\mathbf{f}^{\{n\}} = \mathbf{f}^{\{n-1\}} \cos(a^{\{n\}}) + \boldsymbol{\nu}^{\{n\}} \sin(a^{\{n\}}) \quad \text{iff} \quad p(\mathcal{D}, \dots | \mathbf{f}^{\{n\}}) < y^{\{n\}}. \quad (3.32)$$

ESS can be shown to satisfy detailed balance using similar arguments to those made for slice sampling.

3.2 Variational inference

The aim of variational inference (Hinton and Camp 1993; MacKay 1995) is to find a functional approximation of the posterior, by fitting it with another distribution known as the *variational distribution*. The variational distribution is a user specified probability distribution that we are free to choose—although some choices are more sensible than others. We denote the variational distribution $q(\theta)$ or, to make the distribution's parametrisation explicit, $q(\theta; \phi)$. The optimal parameters, ϕ^* , are

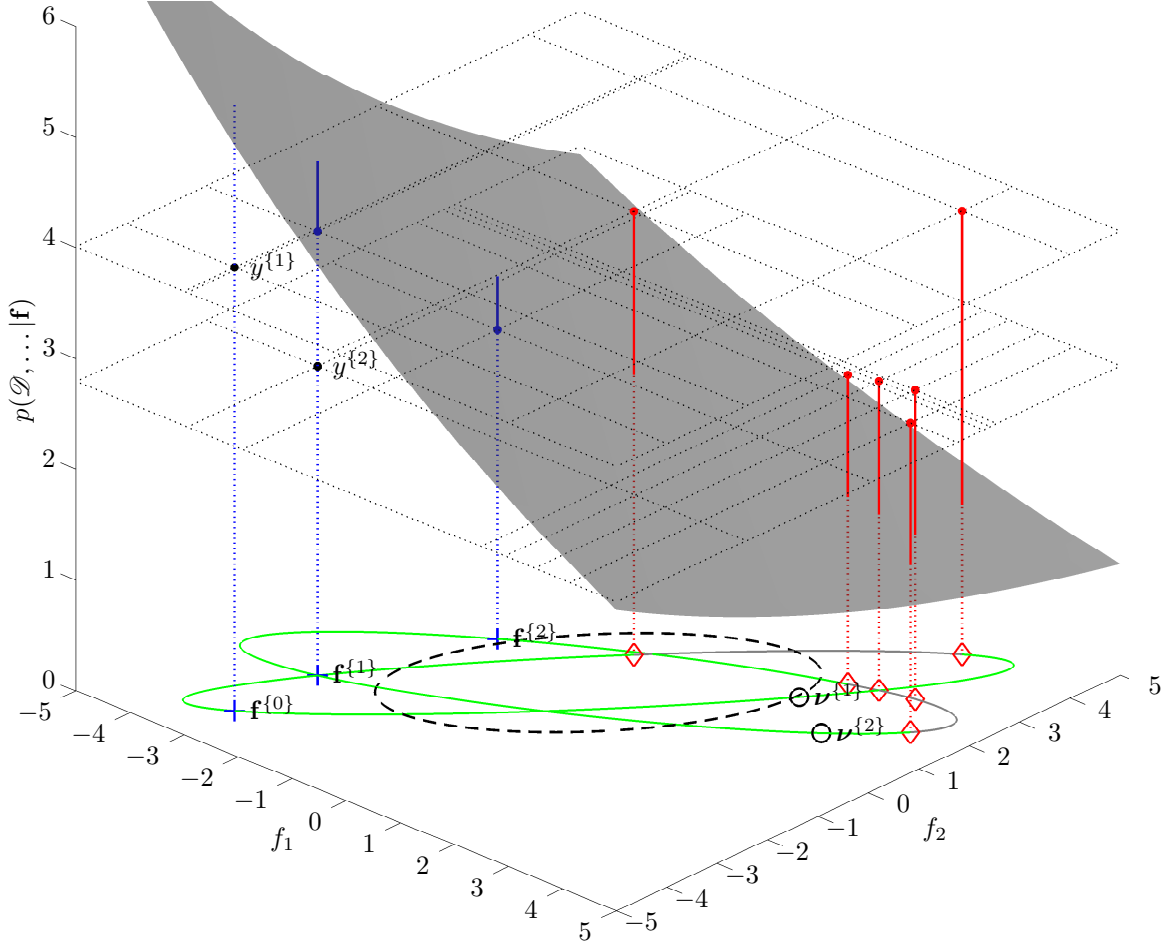


Figure 3.2: Illustration of elliptical slice sampling in two dimensions. Samples are drawn from the Gaussian distribution that has covariance, Σ , shown by the dashed line in the f_1, f_2 plane. Starting at $\mathbf{f}^{(0)}$ we draw the slice level $y^{(1)}$ and ellipse parameter $\nu^{(1)}$. Next, we sample two random angles that correspond to values of $\mathbf{f}$ where the surface, $p(\mathcal{D}, \dots | \mathbf{f})$, shown in grey, lies below the slice level, as indicated by the solid red lines. These values, shown by red diamonds, are rejected and the ellipse is shrunk between them. Eventually, we sample $\mathbf{f}^{(1)}$, which is accepted since $y^{(1)} < p(\mathcal{D}, \dots | \mathbf{f}^{(1)})$, indicated by the solid blue line. Subsequently we sample $y^{(2)}$ and $\nu^{(2)}$ and reject four more values before accepting $\mathbf{f}^{(2)}$.

found by maximising a lower bound on the model evidence. To derive the Evidence Lower Bound (ELBO), we first rearrange Equation 2.19 and then multiply and divide by the variational distribution, leaving the data likelihood unchanged:

$$p(\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\theta|\mathcal{D})} = p(\mathcal{D}|\theta) \frac{p(\theta)}{q(\theta)} \frac{q(\theta)}{p(\theta|\mathcal{D})}. \quad (3.33)$$

We then take logarithms and expectations under $q(\theta)$ of both sides. We now recognise two Kullback–Leibler (KL) divergence terms: one between the variational

and prior distributions and one between the variational and posterior distributions:

$$\log p(\mathcal{D}) = \log p(\mathcal{D}|\theta) + \log \frac{p(\theta)}{q(\theta)} + \log \frac{q(\theta)}{p(\theta|\mathcal{D})} \quad (3.34)$$

$$= \int q(\theta) p(\mathcal{D}|\theta) d\theta + \int q(\theta) \log \frac{p(\theta)}{q(\theta)} d\theta + \int q(\theta) \log \frac{q(\theta)}{p(\theta|\mathcal{D})} d\theta \quad (3.35)$$

$$= \underbrace{\mathbb{E}_{q(\theta)}[\log p(\mathcal{D}|\theta)] - \text{KL}(q(\theta)||p(\theta))}_{\text{Maximising these terms}} + \underbrace{\text{KL}(q(\theta)||p(\theta|\mathcal{D}))}_{\text{minimises this term.}} \quad (3.36)$$

We cannot compute the second of these KL-divergences, since $p(\theta|\mathcal{D})$ is precisely the quantity we are trying to compute. However, since KL-divergences are always non-negative, we can drop the latter term and obtain the ELBO, $\mathcal{L}$:

$$\log p(\mathcal{D}) \geq \mathbb{E}_{q(\theta)}[\log p(\mathcal{D}|\theta)] - \text{KL}(q(\theta)||p(\theta)) \triangleq \mathcal{L}. \quad (3.37)$$

Since the log-evidence, $\log p(\mathcal{D})$, is constant, maximising $\mathcal{L}$ will minimise the slack term, i.e. the KL-divergence between the posterior and variational distributions. In the ideal case $\text{KL}(q(\theta)||p(\theta|\mathcal{D})) = 0$ and therefore $q(\theta) = p(\theta|\mathcal{D})$, thereby recovering the posterior distribution. For this to be true $q(\theta; \phi)$ must be flexible enough to equal $p(\theta|\mathcal{D})$ for some choice of ϕ , which is generally not the case.

The lower bound given in Equation 3.37 can be derived via a more direct route using *Jensen's inequality*, which can be used to show that:

$$\log \int \frac{p(\mathcal{D}|\theta)p(\theta)q(\theta)}{q(\theta)} d\theta \geq \int q(\theta) \log \frac{p(\mathcal{D}|\theta)p(\theta)}{q(\theta)} d\theta. \quad (3.38)$$

The more direct route does not include the slack term and therefore does not make explicit that maximising $\mathcal{L}$ forces $q(\theta)$ toward the posterior distribution of interest.

3.2.1 Mean field variational inference

In most models there are multiple conditioning variables, all of which must be included in the joint variational distribution. The standard approach is to use a *mean field* approximation in which the joint variational distribution is the product of marginal distributions:

$$q(\Theta) = \prod_{n=1}^N q_n(\theta_n). \quad (3.39)$$

Typically, the variational distribution, for each variable marginalised, is chosen to be the conjugate prior of the likelihood over the variable. We can now rewrite Equation 3.36 in terms of this multivariate variational distribution:

$$\mathcal{L} = \int q(\theta_n) \int q(\Theta_{-n}) \log \frac{p(\mathcal{D}, \Theta_{-n}, \theta_n)}{q(\Theta_{-n})q(\theta_n)} d\Theta_{-n} d\theta_n \quad (3.40)$$

$$= \int q(\theta_n) \log \frac{\mathcal{L}(\theta_n; \phi_{-n})}{q(\theta_n)} d\theta_n, \quad (3.41)$$

where $\mathcal{L}(\theta_n; \phi_{-n})$ is the lower bound with all the variables except θ_n integrated out, which is parametrised by the parameters of the remaining variational distributions ϕ_{-n} . This lower bound can be thought of as an unnormalised probability distribution $\bar{p}(\theta_n) \triangleq \frac{1}{Z} \mathcal{L}(\theta_n; \phi_{-n})$. If we substitute $\bar{p}(\theta_n)$ into Equation 3.41 we find that the lower bound can be written as the KL-divergence between $q(\theta_n)$ and $\bar{p}(\theta_n)$ (Bishop 2007),

$$\mathcal{L} = -\text{KL}(q(\theta_n) || \bar{p}(\theta_n)) - \log Z. \quad (3.42)$$

Since the maximum of Equation 3.42 occurs when these two distributions are equal, the optimal variational distribution, $q^*(\theta_n) \triangleq q(\theta_n; \phi_n^*)$, must be equal to $\bar{p}(\theta_n)$ and proportional to $\mathcal{L}(\theta_n; \phi_{-n})$. In models where the likelihood of each variable is conjugate to the distribution of its parents, the lower bound $\mathcal{L}(\theta_n; \phi_{-n})$ typically has the same functional form as $q(\theta_n)$. To calculate the optimal parameter, ϕ_n^* , of $q(\theta_n)$ we simply need to map across terms in $\mathcal{L}(\theta_n; \phi_{-n})$ to those in $q(\theta_n)$ and Z becomes the appropriate normalising term for $q(\theta_n)$. This leads to the VB coordinate ascent algorithm (Ghahramani and Beal 2001) in which each variable is updated in turn

$$q^{*\{t+1\}}(\theta_1) \propto \mathcal{L}(\theta_1; \phi_2^{\{t\}}, \dots, \phi_N^{\{t\}}) \quad (3.43)$$

$$q^{*\{t+1\}}(\theta_2) \propto \mathcal{L}(\theta_2; \phi_1^{\{t+1\}}, \phi_3^{\{t\}}, \dots, \phi_N^{\{t\}}) \quad (3.44)$$

$\vdots$

$$q^{*\{t+1\}}(\theta_N) \propto \mathcal{L}(\theta_N; \phi_1^{\{t+1\}}, \dots, \phi_{N-1}^{\{t+1\}}). \quad (3.45)$$

This algorithm is very similar to the Gibbs sampling algorithm, Equations 3.6-3.8. Like the Gibbs algorithm, the VB coordinate ascent algorithm performs poorly when the model parameters are highly correlated.

3.2.2 Stochastic variational inference

Stochastic variational inference is the application of stochastic gradient descent to variational inference and allows variational inference to be scaled to very large datasets (Hoffman et al. 2010). Stochastic gradient descent is based on the insight that calculating precise gradients is often not an efficient use of computational resources. This is especially true early in the optimisation process when we are far from an optimal solution. For stochastic inference however, only unbiased gradients are necessary. We can maximise $\mathcal{L}(\phi)$ using classical gradient descent by taking a series of steps in the direction of the gradient

$$\phi^{\{t+1\}} := \phi^{\{t\}} + l \nabla_{\phi} \mathcal{L}(\phi^{\{t\}}), \quad (3.46)$$

where l is a step length that can be adjusted using line search. In stochastic gradient descent we instead use noisy gradients $\hat{\nabla}_{\phi} \mathcal{L}(\phi^{\{t\}})$ and compute the series according to

$$\phi^{\{t+1\}} := \phi^{\{t\}} + l_t \hat{\nabla}_{\phi} \mathcal{L}(\phi^{\{t\}}), \quad (3.47)$$

which is guaranteed to converge to a local or global optimum if the sequence of step sizes l_t satisfy the Robins and Monroe conditions:

$$\sum_t l_t = \infty \quad \sum_t l_t^2 < \infty. \quad (3.48)$$

Estimates of noisy gradients are readily available in statistical models simply by sub-sampling the data and so this method can be directly applied to variational inference where the function $\mathcal{L}(\phi)$ is the ELBO and ϕ is a variational parameter to be optimised. In conjugate exponential family models however, this approach is suboptimal. First, this method requires computing gradients and second, as discussed in §3.3.2, the optimal parameters, those which minimise the KL-divergence between $\mathcal{L}(\theta)$ and $q(\theta)$, can be found by inspection. However, it can be shown that the relationship between $\phi^{\{t+1\}}$ and $\phi^{\{t\}}$, where ϕ is the natural parameter of the

distribution $q(\theta)$, is equivalent to taking a unit length step of the gradient under the metric of the Fisher information

$$\phi^{\{t+1\}} := \phi^{\{t\}} + \underbrace{\mathcal{I}^{-1}(\phi^{\{t\}}) \nabla_{\phi} \mathcal{L}(\phi^{\{t\}})}_{\text{Natural gradient}}, \quad (3.49)$$

which is known as the natural gradient. Fortunately the update sequence, Equation 3.47, will still converge if the gradient is pre-multiplied by a positive definite matrix that has bounded Eigen-values (Bottou 1998; Hoffman et al. 2010). If we choose this matrix to be the inverse of the Fisher information we can rearrange Equation 3.49 and substitute it into 3.47 to obtain the update rule:

$$\phi^{\{t+1\}} := (1 - l_t) \phi^{\{t\}} + l_t \hat{\phi}^{\{t+1\}}, \quad (3.50)$$

where $\hat{\phi}^{\{t+1\}}$ is an approximation of the updated parameter obtained from sub-sampling and re-weighting the dataset².

3.2.3 Collapsed variational inference

As previously stated, the coordinate descent algorithm, Equations 3.6-3.8, may perform poorly when the model parameters are highly correlated. In this case collapsed variational inference (Hensman, Rattray, et al. 2012) may be useful. In many conditionally conjugate models, including many of the latent-variable models described in the next chapter³, the conditioning variables Θ includes two sets of variables, $\mathbf{W}$ and $\mathbf{X}$, either of which, together with the data, cause the model to be split into two conditionally independent parts. We will assume the model can be factored as $p(\mathcal{D}|\mathbf{W}, \mathbf{X})p(\mathbf{X})p(\mathbf{W})$.

We begin by lower bounding the conditional distribution on $\mathbf{W}$ and integrating out $\mathbf{X}$ using a variational distribution $q(\mathbf{Z})$ —although we could just as easily do the

²Note that this update is only applied to global parameters (see introduction to Chapter 4 for a definition). Local variables are updated in the usual fashion.

³The reader may prefer to read the introduction to Chapter 4 before returning to this section.

reverse—giving

$$\log p(\mathcal{D}|\mathbf{W}) \geq \int q(\mathbf{X}) \log \frac{p(\mathcal{D}, \mathbf{X}|\mathbf{W})}{q(\mathbf{X})} d\mathbf{X} \quad (3.51)$$

$$= \mathbb{E}_{q(\mathbf{X})}[\log p(\mathcal{D}, \mathbf{X}|\mathbf{W})] + \mathbb{H}[q(\mathbf{X})] \quad (3.52)$$

$$\triangleq \mathcal{L}(\mathbf{W}).$$

The resulting lower bound is conjugate to the prior, $p(\mathbf{W})$, which can be integrated analytically to give the collapsed lower bound:

$$\log p(\mathcal{D}) \geq \log \int p(\mathbf{W}) \exp(\mathcal{L}(\mathbf{W})) d\mathbf{W} \quad (3.53)$$

$$\triangleq \mathcal{L}_{\text{KLC}}.$$

The collapsed bound $\mathcal{L}_{\text{KLC}}$ (King and Lawrence 2006) is only a function of the variational parameters of $q(\mathbf{X})$ and any parameters of the prior distributions, $p(\mathbf{X})$ and $p(\mathbf{W})$. For any setting of the variational parameters, the collapsed bound is equivalent to the uncollapsed bound using the optimal parameters for $q(\mathbf{W})$. Performing natural gradient descent with unit length steps on this bound is equivalent to updating the parameters using the coordinate ascent algorithm, however, the collapsed bound is always a tighter bound than the uncollapsed bound and its gradient is always less than the mean field bound, potentially allowing larger steps to be taken (Hensman, Rattray, et al. 2012).

3.3 Approximate Gaussian processes inference

Due to conjugacy, the posterior distribution of a GP regression model with a linear Gaussian likelihood can be written in closed-form. In practice however, the computational cost of the required linear algebra is often unacceptably high. This is primarily due to the $\mathcal{O}(N^3)$ computational complexity of inverting size N matrices. This computational bottleneck can be addressed in a number of ways. One approach is to tackle the problem head on by developing faster or more scalable matrix inversion algorithms that exploit particular structures of Gaussian process kernel matrices (Ambikasaran and Darve 2014; Banerjee et al. 2013). Other

approaches, that fall under the heading *model approximation*, are those that involve changing the underlying modelling assumptions in order to create opportunities for more efficient linear algebra. A third approach is to derive a variational lower bound that has a lower computational complexity. In the following section we will introduce the variational approach and several model approximation methods that we will make use of in later chapters.

3.3.1 Model approximation

The model approximation methods described in this section, and the variational method described in the following section, rely on a set of *inducing points*, $\mathcal{Z} = \{\mathbf{z}^{(m)} \in \mathcal{Z}\}_{m=1}^M$ and a set of *inducing variables*, $\mathbf{u} = [u_1, \dots, u_M]^\top$, with $p(\mathbf{u}) = \mathcal{N}(\mathbf{u}; 0, \mathbf{K}_{zz})$. While the inducing points and variables are an arbitrary set of latent variables, their role in the model is as a compressed representation of the data. If we assume that $\mathbf{u}$ is part of the same Gaussian process function, f , that we aim to approximate, the mean value of f at the data points, $\mathbf{f}$, given the inducing variables is:

$$\mathbb{E}[\mathbf{f}|\mathbf{u}] = \mathbf{K}_{xz}\mathbf{K}_{zz}^{-1}\mathbf{u}. \quad (3.54)$$

Assuming the isotropic likelihood, Equation 2.79, and varying assumptions about the independence relationships between the function values at various inputs, leads us to a number of different models reported in the literature (Snelson and Ghahramani 2005). Some models also distinguish between function values at the training inputs, $\mathbf{f}$, and function values at the test inputs, $\mathbf{f}^*$. The effect of these independence assumptions is to introduce zeros into the covariance matrix of conditional distributions of $\mathbf{f}|\mathbf{u}$ and $\mathbf{f}^*|\mathbf{u}$, hence they are *sparse*-Gaussian process models. As described by Quiñonero-Candela and Rasmussen (2005) most of these models can be placed in a hierarchy of increasing sparsity: GP $\rightarrow$ PIC $\rightarrow$ FIC $\rightarrow$ DIC. In the Partially Independent Conditional (PIC) scheme, some of the values in $\mathbf{f}$ are conditionally independent of each other. In the Fully Independent Conditional (FIC) scheme, all the values in $\mathbf{f}$ are conditionally independent of each other. In

the Deterministic Inducing Conditional (DIC) scheme, all the points in $\mathbf{f}$ are fully determined by $\mathbf{u}$. If only the *training* conditional distribution is sparsified and the full Gaussian process is retained for prediction, we have the related hierarchy: GP $\rightarrow$ PITC $\rightarrow$ FITC $\rightarrow$ DTC; however these models are not strictly Gaussian processes due to the distinction between test and training points. Table 3.1 summarises the prior covariance structure of $[\mathbf{f}, \mathbf{f}^*]$ (with $\mathbf{u}$ integrated out), resulting from these approximations. To keep the notation compact, Table 3.1 uses the following definitions:

$$\mathbf{Q}_{ab} \triangleq \mathbf{K}_{az} \mathbf{K}_{zz}^{-1} \mathbf{K}_{zb} \quad (3.55)$$

$$\mathbf{J}_{ab} \triangleq \mathbf{Q}_{ab} - \mathbf{K}_{ab}. \quad (3.56)$$

The Deterministic Training Conditional (DTC) and FITC models have similar computational requirements, requiring, respectively, the calculation of $[\mathbf{Q}_{xx} + \mathbf{I}\sigma^2]^{-1}$ and $[\mathbf{Q}_{xx} - \text{diag}[\mathbf{J}_{xx}] + \mathbf{I}\sigma^2]^{-1}$, which can be computed efficiently using Woodbury matrix inversion identity. The efficiency of the PIC and Partially Independent Training Conditional (PITC) schemes depends on the degree of independence assumed. We will make use of the PIC approximation to construct dependent Poisson processes in Chapter 5, so we will defer a fuller discussion until then and for now emphasise that the PIC is part of a family of sparse approximations of the full Gaussian process.

3.3.2 Variational approximation

The inducing point framework, introduced in the previous section, improves computational efficiency in Gaussian process regression by approximating the full GP covariance structure. Since the covariance structures of the resulting models are different from the original Gaussian process, the data likelihood under the approximation is different to the original model. However, the inducing point framework also allows us to compute a computationally efficient lower bound on the original Gaussian process likelihood as we shall now see.

Model	Joint prior covariance of $[\mathbf{f}, \mathbf{f}^*]$	$\mathbf{f} \mathbf{u}$	$\mathbf{f}^* \mathbf{u}$	GP?
GP	$\begin{bmatrix} \mathbf{K}_{xx} & \mathbf{K}_{x*} \\ \mathbf{K}_{*x} & \mathbf{K}_{**} \end{bmatrix}$	exact	exact	yes
DTC	$\begin{bmatrix} \mathbf{Q}_{xx} & \mathbf{Q}_{x*} \\ \mathbf{Q}_{*x} & \mathbf{K}_{**} \end{bmatrix}$	determ.	exact	no
DIC	$\begin{bmatrix} \mathbf{Q}_{xx} & \mathbf{Q}_{x*} \\ \mathbf{Q}_{*x} & \mathbf{Q}_{**} \end{bmatrix}$	determ.	determ.	yes
FITC	$\begin{bmatrix} \mathbf{Q}_{xx} - \text{diag}[\mathbf{J}_{xx}] & \mathbf{Q}_{x*} \\ \mathbf{Q}_{*x} & \mathbf{K}_{**} \end{bmatrix}$	indep.	exact.	no
FIC	$\begin{bmatrix} \mathbf{Q}_{xx} - \text{diag}[\mathbf{J}_{xx}] & \mathbf{Q}_{x*} \\ \mathbf{Q}_{*x} & \mathbf{Q}_{**} - \text{diag}[\mathbf{J}_{**}] \end{bmatrix}$	indep.	indep.	yes
PITC	$\begin{bmatrix} \mathbf{Q}_{xx} - \text{blkdiag}[\mathbf{J}_{xx}] & \mathbf{Q}_{x*} \\ \mathbf{Q}_{*x} & \mathbf{K}_{**} \end{bmatrix}$	partially indep.	exact.	no

Table 3.1: Summary of sparse Gaussian process approximations.
(Adapted from Table 1 of Quinero-Candela and Rasmussen (2005).)

As before, we condition the distribution of $\mathbf{f}$ on, $\mathbf{u}$, the value of the Gaussian process evaluated at the inducing points locations, $\mathcal{Z}$:

$$p(\mathbf{f}|\mathbf{u}) = \mathcal{N}(\mathbf{f}; \mathbf{K}_{zx}\mathbf{K}_{zz}^{-1}\mathbf{u}, \mathbf{K}_{xx} - \mathbf{K}_{xz}\mathbf{K}_{zz}^{-1}\mathbf{K}_{zx}) \quad (3.57)$$

$$p(\mathbf{u}) = \mathcal{N}(0; \mathbf{K}_{zz}). \quad (3.58)$$

Note that this does not change the likelihood of $\mathbf{f}$, since we can integrate out $\mathbf{u}$ to recover the original distribution of $\mathbf{f}$, i.e. $\mathcal{N}(\mathbf{f}; 0, \mathbf{K}_{xx})$. The joint distribution of $\mathbf{y}$, $\mathbf{f}$ and $\mathbf{u}$ is $p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u})$ and the model evidence is:

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u}) \, d\mathbf{f} \, d\mathbf{u}. \quad (3.59)$$

We now introduce a variational distribution, $q(\mathbf{u}) = \mathcal{N}(\mathbf{u}; \mathbf{m}, \mathbf{S})$, on the inducing variables. As described in Section 3.2, we multiple and divide the joint by this distribution. We then take logarithms and obtain a lower bound using Jensen's inequality. In this case however, we will bring not just $q(\mathbf{u})$ outside the logarithm, but

also $p(\mathbf{f}|\mathbf{u})$. Once outside of the logarithm, we can integrate these two distributions analytically to give a variational distribution $q(\mathbf{f})$,

$$q(\mathbf{f}) = \int p(\mathbf{f}|\mathbf{u})q(\mathbf{u}) \, d\mathbf{u} \quad (3.60)$$

$$= \mathcal{N}(\mathbf{f}; \mathbf{K}_{xz}\mathbf{K}_{zz}^{-1}\mathbf{m}, \tilde{\mathbf{K}}). \quad (3.61)$$

where $\tilde{\mathbf{K}} = \mathbf{K}_{xx} - \mathbf{K}_{xz}\mathbf{K}_{zz}^{-1}\mathbf{K}_{zx} + \mathbf{K}_{xz}\mathbf{K}_{zz}^{-1}\mathbf{S}\mathbf{K}_{zz}^{-1}\mathbf{K}_{zx}$, leading to the general variational lower bound:

$$\log p(\mathbf{y}) = \log \iint \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u})q(\mathbf{u})}{q(\mathbf{u})} \, d\mathbf{f} \, d\mathbf{u} \quad (3.62)$$

$$\geq \int q(\mathbf{f}) \log p(\mathbf{y}|\mathbf{f}) \, d\mathbf{f} - \int q(\mathbf{u}) \log \frac{q(\mathbf{u})}{p(\mathbf{u})} \, d\mathbf{u} \quad (3.63)$$

$$= \mathbb{E}_{q(\mathbf{f})}[\log p(\mathbf{y}|\mathbf{f})] - \text{KL}(q(\mathbf{u})||p(\mathbf{u})). \quad (3.64)$$

For the standard isotropic independent and identically distributed Gaussian noise model $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}; \mathbf{f}, \sigma^2\mathbb{I})$, we can find analytically the parameters, $\mathbf{m}^*$ and $\mathbf{S}^*$, of the optimal variational distribution, $q^*(\mathbf{u}) = \mathcal{N}(\mathbf{u}; \mathbf{m}^*, \mathbf{S}^*)$. Expanding the expectation in Equation 3.64 we see that:

$$\begin{aligned} & \mathbb{E}_{q(\mathbf{f})}[\log \mathcal{N}(\mathbf{y}; \mathbf{f}, \sigma^2\mathbb{I})] \\ &= \mathbb{E}_{q(\mathbf{f})} \left[-\frac{D}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} (\mathbf{y} - \mathbf{f})^\top (\mathbf{y} - \mathbf{f}) \right] \end{aligned} \quad (3.65)$$

$$= -\frac{D}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \left\{ \mathbb{E}_{q(\mathbf{f})}[\text{tr}(\mathbf{f}\mathbf{f}^\top)] + \mathbf{y}^\top \mathbf{y} - 2\mathbf{y}^\top \mathbb{E}_{q(\mathbf{f})}[\mathbf{f}] \right\}, \quad (3.66)$$

where the statistics required to compute 3.66 are:

$$\mathbb{E}_{q(\mathbf{f})}[\text{tr}(\mathbf{f}\mathbf{f}^\top)] = \mathbb{E}_{q(\mathbf{f})}[\mathbf{f}]^\top \mathbb{E}_{q(\mathbf{f})}[\mathbf{f}] + \text{tr}(\text{Cov}_{q(\mathbf{f})}[\mathbf{f}, \mathbf{f}]) \quad (3.67)$$

$$\mathbb{E}_{q(\mathbf{f})}[\mathbf{f}] = \mathbf{K}_{xz}\mathbf{K}_{zz}^{-1}\mathbf{m} \quad (3.68)$$

$$\text{tr}(\text{Cov}_{q(\mathbf{f})}[\mathbf{f}, \mathbf{f}]) = \text{tr}(\tilde{\mathbf{K}}). \quad (3.69)$$

Substituting Equations 3.67-3.69 allows Equation 3.66 to be written compactly as

$$\mathbb{E}_{q(\mathbf{f})}[\log \mathcal{N}(\mathbf{y}; \mathbf{f}, \mathbb{I}\sigma^2)] \geq \mathcal{N}(\mathbf{y}; \mathbf{K}_{xz}\mathbf{K}_{zz}^{-1}\mathbf{m}, \sigma^2\mathbb{I}) - \frac{1}{2\sigma^2} \text{tr}(\tilde{\mathbf{K}}). \quad (3.70)$$

We will define $\mathcal{L}_{\text{GP}}$ to be the lower bound for variational GP regression with isotropic Gaussian noise:

$$\mathcal{L}_{\text{GP}} \triangleq \mathbb{E}_{q(\mathbf{f})}[\log \mathcal{N}(\mathbf{y}; \mathbf{f}, \mathbb{I}\sigma^2)] - \text{KL}(q(\mathbf{u})||p(\mathbf{u})). \quad (3.71)$$

Differentiating Equation $\mathcal{L}_{\text{GP}}$ with respect to $\mathbf{m}$ and $\mathbf{S}$ and setting equal to zero recovers the optimal parameters

$$\mathbf{S}^* = \left[\frac{1}{\sigma^2} \mathbf{K}_{zz}^{-1} \mathbf{K}_{zx} \mathbf{K}_{xz} \mathbf{K}_{zz}^{-1} + \mathbf{K}_{zz}^{-1} \right]^{-1} \quad (3.72)$$

$$\mathbf{m}^* = \frac{1}{\sigma^2} \mathbf{S}^* \mathbf{K}_{zz}^{-1} \mathbf{K}_{xz} \mathbf{y}. \quad (3.73)$$

If we substitute these optimal parameters back into $\mathcal{L}_{\text{GP}}$ we obtain the collapsed variational lower bound of Titsias (2009). For large datasets, Hensman, Fusi, et al. (2013) recommend using stochastic gradient descent.

4

Latent variable models, structure learning and complex networks

In this chapter we will introduce a number of probabilistic models described in the literature that have been developed for learning hidden structure in data—including a number of well-known unsupervised machine learning algorithms—as well as probabilistic models of complex networks. Most, if not all, of these models can be classified as *latent variable* models. The general structure of a latent variable model is shown in Figure 4.1 using a *graphical model* (Pearl 2000). A latent variable model typically consists of a set of *global* variables, $\{\mathbf{w}_d\}_{d=1}^D$, and a set of *local* variables (also referred to as the *latent* variables), $\{\mathbf{x}^{(n)}\}_{n=1}^N$, where each of the latter is typically associated with single data point $\mathbf{y}^{(n)}$. The global and local variables often have their own prior distributions $p(\mathbf{w}_d|\theta)$ and $p(\mathbf{x}^{(n)}|\phi)$. Since we could include many models under this heading, we will restrict our attention to those that have inspired and motivated this work and that provide context for the work presented in later chapters. A wider perspective on other related and contemporary work is presented in Chapter 8.

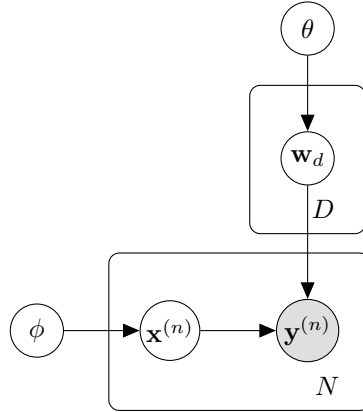


Figure 4.1: The typical structure of a latent variable model showing the global variables $\mathbf{w}_d$, the local variables $\mathbf{x}^{(n)}$ and the observed data $\mathbf{y}^{(n)}$

4.1 Factor models

The problem of determining hidden structure in data can often be reduced, conceptually if not practically, to the problem of finding a factorisation of a matrix containing the data. Factor models have a long history in statistics. Many generative models of networks and complex data are based on the concept of factor models. Given data-vectors $\{\mathbf{y}^{(n)}\}_{n=1}^N$, each with dimensionality D , we can form a stacked matrix $\mathbf{Y} \in \mathbb{R}^{D \times N}$ (the design matrix¹). A binary factorisation of the data yields two matrices $\mathbf{W} \in \mathbb{R}^{D \times R}$, $\mathbf{X} \in \mathbb{R}^{R \times N}$ such that $\mathbf{Y} \approx \mathbf{W}\mathbf{X}$, or, more explicitly:

$$\begin{bmatrix} \vdots & \dots & \dots & \vdots \\ \vdots & \dots & \dots & \vdots \\ \mathbf{y}^{(1)} & \dots & \dots & \mathbf{y}^{(N)} \\ \vdots & \dots & \dots & \vdots \\ \vdots & \dots & \dots & \vdots \end{bmatrix} \approx \begin{bmatrix} w_{1,1} & \dots & w_{1,R} \\ \vdots & \vdots & \vdots \\ w_{d,1} & \dots & w_{d,R} \\ \vdots & \vdots & \vdots \\ w_{D,1} & \dots & w_{D,R} \end{bmatrix} \begin{bmatrix} \vdots & \dots & \dots & \vdots \\ \mathbf{x}^{(1)} & \dots & \dots & \mathbf{x}^{(N)} \\ \vdots & \dots & \dots & \vdots \end{bmatrix}. \quad (4.1)$$

The columns of $\mathbf{W}$ are the global variables which form a set of *bases* and the local variables matrix $\mathbf{X}$ contains a set of stacked weights $\{\mathbf{x}^{(n)}\}_{n=1}^N$.

Each data point $\mathbf{y}^{(n)}$ is approximated by a weighted combination of the basis, with the weights contained in the corresponding latent variable $\mathbf{x}^{(n)}$. Since R is generally assumed to be less than D , the resulting decomposition is *low-rank*.

¹Note that the design matrix is often orientated in the opposite direction in many texts.

Analysing the basis may give insight into the structure of the data and we may be able to assign them “real world” interpretations. However, factorisations are in general non-unique as there are an infinite number of such *Procrustes* decompositions since

$$\mathbf{Y} \approx \mathbf{W}\mathbf{X} = \mathbf{W}\mathbf{R}\mathbf{R}^\top \mathbf{X} = \mathbf{W}'\mathbf{X}', \quad (4.2)$$

where $\mathbf{R}$ is an orthogonal rotation matrix of which there are infinitely many. The matrices $\mathbf{X}$ and $\mathbf{W}$ are therefore said to be *non-identifiable*.

Well-known algorithms for computing the factors include Principle Component Analysis (PCA), Factor Analysis (FA) and Independent Component Analysis (ICA) (Roweis and Ghahramani 1999). The solution obtained by a factorisation algorithm is determined by the error model that characterises the approximation, any *constraints* it implicitly or explicitly makes, as well as initial conditions to which the algorithm is sensitive. In a Bayesian setting, constraints may be imposed by prior distributions on the factors.

4.1.1 Principle Component Analysis

PCA solutions are constrained so that the principal components, the directions contained in the columns of $\mathbf{W}$, are orthogonal and the latent variables are uncorrelated. This results in a well-posed unique decomposition that can be found efficiently using the Singular Valued Decomposition (SVD). The normal assumption implicit in the underlying squared error loss function may not always be appropriate and the orthogonality constraint means that the basis computed by PCA extend over the entire data vector; this can make PCA hard to interpret.

Probabilistic-PCA (Tipping and Bishop 1997) is a generative model interpretation

of PCA, given by

$$p(\mathbf{X}) = \prod_{n=1}^N \mathcal{N}(\mathbf{x}^{(n)}; \mathbf{0}, \mathbb{I}) \quad (4.3)$$

$$p(\mathbf{W}|\alpha) = \prod_{d=1}^D \prod_{r=1}^R \mathcal{N}(w_{d,r}; 0, \alpha^{-1}) \quad (4.4)$$

$$p(\mathbf{Y}|\mathbf{X}, \mathbf{W}, \beta) = \prod_{n=1}^N \mathcal{N}(\mathbf{y}^{(n)}; \mathbf{W}\mathbf{x}^{(n)}, \beta^{-1}\mathbb{I}). \quad (4.5)$$

Although the maximum likelihood solution to Probabilistic-PCA is still that computed by SVD, the probabilistic interpretation is useful as the starting point for developing new algorithms. For example, under the probabilistic interpretation it becomes clear how Expectation Maximisation (EM) can be used when missing data must be handled.

In Bayesian-PCA (Bishop 1999) a hierarchical prior on the columns of $\mathbf{W}$ is used. This results in *Bayesian shrinkage*, in which elements of $\mathbf{W}$ that are not well supported by the data are driven to small values. Driving a column to zero effectively removes that column from the model and we are therefore able to infer what the effective dimensionality of the data is. This feature of Bayesian inference is known as *Automatic Relevance Determination* (ARD).

Under the standard Bayesian PCA model, the columns of $\mathbf{W}$ are normally distributed; Mohamed et al. (2009) extend this to any exponential family distribution.

4.1.2 Non-negative Matrix Factorisation

One factor model particularly relevant to work presented in later chapters is Non-negative Matrix Factorisation (NMF). NMF is a popular matrix factor model in which the data is assumed to be non-negative and the factors have non-negative constraints. The non-negative restriction is not as significant as it might seem, since real data is very often non-negative. In their seminal work Lee and Seung (2001) investigated the following divergence measure:

$$D_{\text{NMF}}(\mathbf{A}||\mathbf{B}) = \sum_i \sum_j \left(a_{i,j} \log \frac{a_{i,j}}{b_{i,j}} - a_{i,j} + b_{i,j} \right), \quad (4.6)$$

to quantify the approximation given in Equation 4.1. They used Jensen's inequality to lower bound the optimisation problem of finding values of $\mathbf{W}$ and $\mathbf{X}$ which minimise $D_{\text{NMF}}(\mathbf{Y}||\mathbf{WX})$ and are then able to derive the following set of update equations:

$$x_{r,n} := x_{r,n} \frac{\sum_{i=1}^D w_{i,r} y_{i,n} / [\mathbf{WX}]_{i,n}}{\sum_{j=1}^D w_{j,r}} \quad (4.7)$$

$$w_{d,r} := w_{d,r} \frac{\sum_{i=1}^N x_{d,i} y_{d,i} / [\mathbf{WX}]_{d,i}}{\sum_{j=1}^N x_{r,j}}, \quad (4.8)$$

which are iterated until convergence. The outstanding result is that the basis found by this simple algorithm are typically observed to be *local*, with each base vector reconstructing only part of the input vector and containing zeros elsewhere. This is due to the additive nature of the NMF decomposition. The basis found by NMF can only add features to data points and not remove them. Since features tend to extend across elements located close together in the input vector, the basis can often be interpreted. For example, when applied to face data, noses, mouths and eyes are easily recognised. Thus NMF naturally induces *sparsity* and gives rise to more easily interpretable basis vectors than other algorithms such as PCA, is easy to implement and relatively fast to compute. NMF has been successfully applied to a wide variety of problems and is particularly popular for signal processing applications such as audio and image processing. In general non-negativity is usually not sufficient to ensure a unique decomposition (Donoho and Stodden 2003). Optimisation of both $\mathbf{W}$, $\mathbf{X}$ is not simultaneously convex and iterative optimization procedures are usually employed. Since the work of Lee and Seung (2001), NMF algorithms based on a variety of divergence measures have been proposed. Some of these are listed in Table 4.1.

As described by Cemgil (2009) minimising the divergence given by Equation 4.6 is equivalent to maximising the likelihood of the Poisson model:

$$P(\mathbf{Y}|\mathbf{W}, \mathbf{X}) = \prod_{d=1}^D \prod_{n=1}^N \mathcal{P}(y_{d,n}; [\mathbf{WX}]_{d,n}). \quad (4.9)$$

Square loss	$D_2 = x - y ^2$	1
Alpha div.	$D_A^{(\beta)}(x y) = x \frac{(x/y)^{\beta-1}-1}{\beta(\beta-1)} + \frac{x-y}{\beta}$	2
Beta div.	$D_\beta(x y) = \begin{cases} \frac{1}{\beta(\beta-1)}(x^\beta + (\beta-1)y^\beta - \beta xy^{\beta-1}) & \beta \neq 0, 1 \\ x \log \frac{x}{y} - x + y & (\beta = 1) \\ \frac{x}{y} - \log xy - 1 & (\beta = 0) \end{cases}$	3
Bregman div.	$D_\varphi(x y) = \varphi(x) - \varphi(y) - \nabla_\varphi(y)(x - y)$ φ strictly convex	4

¹ Lee and Seung (2001) ; ² Cichocki et al. (2006) ; ³ Févotte and Idier (2011), Tan and Févotte (2013) ; ⁴ Dhillon and Sra (2005)

Table 4.1: NMF algorithms are based on a variety of divergences measures.

Since this parameter of the Poisson distribution in Equation 4.9 is the result of summing R values together, we can use the infinite divisibility property of the Poisson distribution to rewrite the equation as

$$P(\mathbf{Y}|\mathbf{W}, \mathbf{X}) = \prod_{d=1}^D \prod_{n=1}^N \prod_{r=1}^R \mathcal{P}(s_{d,r,n}; w_{d,r} x_{r,n}). \quad (4.10)$$

where $y_{d,n} = \sum_{r=1}^R s_{d,r,n}$. By doing so, we introduce a large number of new latent variables, $\{s_{d,r,n}\}$. Cemgil uses this new formulation to develop Bayesian-NMF algorithms based on Markov-Chain Monte-Carlo (MCMC) and variational inference; they find that these algorithms perform similarly, however the variational method is significantly faster.

4.2 The Gaussian Process Latent Variable Model

In contrast to the linear factor models described in the previous section, the Gaussian Process Latent Variable Model (GPLVM) is a non-linear latent variable model, derived from the probabilistic formulation of PCA by kernelising the latent-variables (Lawrence 2003). If we integrate out the random variable $\mathbf{W}$ in Equations 4.4-4.5 then we obtain the following log-likelihood:

$$\log p(\mathbf{Y}|\mathbf{X}, \alpha, \beta) = -\frac{DN}{2} \log(2\pi) - \frac{D}{2} \log |\mathbf{K}| - \frac{1}{2} \text{tr}(\mathbf{Y}\mathbf{K}\mathbf{Y}^\top), \quad (4.11)$$

where $\mathbf{K} = \alpha^{-1} \mathbf{X}^\top \mathbf{X} + \beta^{-1} \mathbb{I}$. The matrix $\mathbf{K}$ is a kernel matrix constructed from a simple linear kernel-function. If we replace the linear kernel matrix with a non-linear kernel, such as the exponentiated-quadratic kernel function, Equation 2.77,

we create a non-linear mapping between the data $\mathbf{Y}$ and the latent coordinate space containing the variables $\mathbf{x}^{(n)}$. The latent variables are located by optimising the Equation 4.11 using gradient-descent. Since the latent variables can live in a lower dimensional space than the data, the GPLVM provides a useful probabilistic model for non-linear dimensionality reduction.

The GPLVM can be improved by integrating out the latent coordinates, $\mathbf{X}$, against the prior of Equation 4.3. This can be achieved using MCMC or, more efficiently, using variational inference (Titsias and Lawrence 2010). The variational-GPLVM can be derived from the sparse variational Gaussian Process (GP) formulation, described in §3.3.2. Since GPLVM likelihood, Equation 4.11, factorises across the D dimensions, we can lower bound the model evidence using Jensen’s inequality after introducing a variational distribution, $q(\mathbf{X})$, and again by substituting the previously derived lower bound for the variational GP, $\mathcal{L}_{\text{VBGP}}(\mathbf{y}_d)$, given in Equation 3.71. This results in the following Evidence Lower Bound (ELBO):

$$\log p(\mathbf{Y}) \geq \mathbb{E}_{q(\mathbf{X})} \left[\log \prod_{d=1}^D p(\mathbf{y}_d | \mathbf{X}) \right] - \text{KL}(q(\mathbf{X}) || p(\mathbf{X})) \quad (4.12)$$

$$\geq \sum_{d=1}^D \mathbb{E}_{q(\mathbf{X})} [\mathcal{L}_{\text{VBGP}}(\mathbf{y}_d)] - \text{KL}(q(\mathbf{X}) || p(\mathbf{X})) \quad (4.13)$$

$$\begin{aligned} &= \sum_{d=1}^D \mathbb{E}_{q(\mathbf{X})} \mathbb{E}_{q(\mathbf{f}_d)} [\log p(\mathbf{y}_d | \mathbf{f}_d)] - \sum_{d=1}^D \text{KL}(q(\mathbf{u}_d) || p(\mathbf{u}_d)) \\ &\quad - \text{KL}(q(\mathbf{X}) || p(\mathbf{X})) \\ &\triangleq \mathcal{L}_{\text{GPLVM}}. \end{aligned} \quad (4.14)$$

Crucially, in the sparse formulation, the variables in $\mathbf{X}$ only appear in the matrices $\mathbf{K}_{xz}$ and not $\mathbf{K}_{zz}^{-1}$. Therefore—under certain conditions—we can integrate the terms in $\mathbb{E}_{q(\mathbf{f}_d)} [\log p(\mathbf{y}_d | \mathbf{f}_d)]$, Equations 3.66-3.69, by replacing terms containing $\text{tr}(\mathbf{K}_{xx})$, $\mathbf{K}_{xz}$ and $\mathbf{k}_{zx}^{(n)} \mathbf{k}_{xz}^{(n)}$ with their expectations under $q(\mathbf{X})$, respectively ψ_{xx}^0 , Ψ_{xy}^1 and $\Psi_{zz}^{2(n)}$. Assuming a mean-field factorisation of $q(\mathbf{X})$ over $\mathbf{x}^{(n)}$, these matrices—known as the “Psi-statistics”—can be calculated by integrating the kernel against $q(\mathbf{x}^{(n)})$

as follows:

$$\psi_{xx}^0 = N\gamma \quad (4.15)$$

$$[\Psi_{xz}^1]_{n,m} = \int q(\mathbf{x}^{(n)})K(\mathbf{x}^{(n)}, \mathbf{z}^{(m)}) d\mathbf{x}^{(n)} \quad (4.16)$$

$$[\Psi_{zz}^{2(n)}]_{m,m'} = \sum_{n=1}^N \int q(\mathbf{x}^{(n)})K(\mathbf{z}^{(m)}, \mathbf{x}^{(n)})K(\mathbf{x}^{(n)}, \mathbf{z}^{(m')}) d\mathbf{x}^{(n)}. \quad (4.17)$$

Having made these substitutions we can, as before, compute the optimal variational parameters $\mathbf{m}_d^*$ and $\mathbf{S}_d^*$ (of which there are now D times as many) and, in order to collapse the bound, substitute these definitions back in.

4.3 Generative models of networks

A standard mathematical description of a (static) network is as a graph given by the tuple $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{1, \dots, N\}$ is a set of nodes and $\mathcal{E} = \{(i, j) | i, j \in \mathcal{V}\}$ is a set of edges. Edges may be unordered pairs corresponding to undirected graphs or ordered pairs corresponding to directed graphs and the edge description may be extended to include, for example, edge-weights. A graph may also be represented as an *adjacency matrix*, $\mathbf{A} \in \{0, 1\}^{N \times N}$, with entries $a_{i,j} = 1 \Leftrightarrow (i, j) \in \mathcal{E}$.

Most generative probabilistic models of networks attempt to directly model the adjacency matrix. One method is to simply apply the binary matrix decomposition of Equation 4.1 directly to the adjacency matrix, for example using NMF. This method, proposed by Psorakis et al. (2011) who used the Maximum A' Posterior (MAP) NMF method of Tan and Fevotte (2009), can be used to reveal putative communities and their inter and intra-community relationships. The effective inner-rank of the decomposition indicates the number of communities and the elements of the matrix factors give the strength of attachment between nodes and communities. Other models, such as the Stochastic Block Model (SBM), are more complex matrix factorisations that allow additional constraints on the decomposition to be imposed.

There is an enormous literature on the statistical modelling and characterisation of networks—from both frequentist and Bayesian perspectives—so we will strictly limit

discussion to several important generative network models. We will consider models for both static (time-invariant) networks and for temporally inhomogeneous *dynamic* networks. We will also review Aldous-Hoover Theorem, which has important implications for the limits of many generative network models.

4.3.1 The Stochastic Block Model

The Stochastic Block Model (SBM) is a generative model of networks first introduced by Laskey and Leinhardt (1983). It has inspired a range of derivatives and the abbreviation SBM is broadly used to describe this class of model. The SBM is a tripartite factorisation of the adjacency matrix:

$$\mathbf{Y} \sim p_{\mathbf{Y}}(\mathbf{Z}\mathbf{W}\mathbf{Z}^{\top}), \quad (4.18)$$

where $p_{\mathbf{Y}}$ is usually an element-wise probability density or mass function appropriate for the type of observed data. Each element of $\mathbf{Y}$ is therefore distributed as $y_{m,n} \sim p(\sum_{r=1}^R \sum_{s=1}^R z_{m,r} w_{r,s} z_{n,s})$ where $z_{m,r}$ is the strength of attachment of the m^{th} node to the r^{th} community and $w_{r,s}$ quantifies the interaction between the r^{th} and s^{th} communities. Thus the strength of interaction between a pair of nodes is governed by the sum of inter and intra community interaction intensities of the communities the nodes are members of. For binary matrices corresponding to connection only graphs, the likelihood model is typically

$$y_{m,n} \sim \text{Bern}\left(\sigma\left(\sum_{r=1}^R \sum_{s=1}^R z_{m,r} w_{r,s} z_{n,s}\right)\right), \quad (4.19)$$

where σ is the sigmoid squashing function, Equation 2.82. The model is completed in a Bayesian setting with prior distributions over the factors $\mathbf{W}$ and $\mathbf{Z}$. For weighted matrices, the prior for $\mathbf{W}$ is typically conjugate to the likelihood $p_{\mathbf{Y}}$. In a typical SBM the rows of $\mathbf{Z}$ are categorically distributed, $\mathbf{z}_n \sim \text{Cat}(\pi_{1:R})$, and so each node is the member of precisely one community.

An example of a block structured adjacency matrix is shown in Figure 4.2. Three lines separate the apparent communities. In the first community, comprising nodes

A , B and C , most of the interaction is between nodes of the same community. In social network analysis, this is known as an *assortative* community or one exhibiting *homophily*—the tendency of people to interact with similar people. Members of the second “community”, comprising nodes D , E and F , interact mostly with members of the third community, but not with each other; this is known as a *dissortative community*. (In some definitions, a community is, by definition, a cluster that interacts more with itself than with other clusters). The third community exhibits both assortative and disassortative mixing.

	A	B	C	D	E	F	G	H	I
A	15	21	18	1	4	1	2	1	0
B	22	15	18	2	1	2	2	1	1
C	28	16	24	0	2	0	1	3	1
D	0	0	3	0	0	0	25	15	23
E	1	2	2	0	1	1	15	28	22
F	1	2	0	3	3	0	22	20	19
G	0	0	2	16	21	21	13	20	20
H	1	1	0	20	13	28	20	21	20
I	1	1	1	23	18	24	21	20	27

Figure 4.2: An example of an adjacency matrix showing assortative and dissortative block structure.

Of course, membership does not need to be restricted to just a single community. For example, the *mixed-membership* model of Airoldi et al. (2008) allows each interaction to occur within its own separate community context, with the context conditioned on per-node community distributions. An alternative approach, which achieves a similar effect, is a *feature* model that allows simultaneous membership of multiple communities. The subtle modelling difference between these approaches is that in the former, a node may be 50% in community A and 50% in community B , and in the latter, the node may be 100% in both.

The SBM can be transformed into an analogous Bayesian non-parametric equivalent, known as the Infinite Relational Model (IRM) (Kemp et al. 2006), by allowing an

infinite number of communities using the Chinese Restaurant Process (CRP). In the IRM the community memberships are distributed $\mathbf{Z} \sim \text{CRP}(\alpha)$ as described in §2.4.3. (The IRM is discussed in more detail in Appendix C, which describes collapsed variational inference (§3.2.3) in this model). An infinite dimensional feature model for dyadic (i.e. bipartite graph) data was described by Meeds et al. (2007) using the Indian Buffet Process (IBP) and a similar model including side information for networks called the Latent Feature Relational Model (IFRM) was described by Miller, Griffiths, et al. (2009). The Infinite Latent Attribute Model (ILAM) (Palla, Knowles, et al. 2012) uses both the CRP and IBP to allow each feature to have subclusters in the following generative process:

$$\mathbf{Z} \sim \text{IBP}(\alpha) \quad (4.20)$$

$$\mathbf{c}^{(m)} \sim \text{CRP}(\gamma) \quad (4.21)$$

$$w_{k,k'}^{(m)} \sim N(0, \sigma_w^2) \quad (4.22)$$

$$y_{i,j} \sim \text{Bern} \left(\sigma \left(\sum_{m=1}^{\infty} z_{i,m} z_{j,m} w_{c_i^{(m)}, c_j^{(m)}}^{(m)} \right) \right). \quad (4.23)$$

The intuition behind ILAM is as follows: while nodes (say, people) may have the same features (say, plays a sport), represented by the columns of $\mathbf{Z}$, features may have mutually exclusive sub-features (say, is a member of a particular club), which are determined by values of $c^{(m)}$. Therefore, we do not have to create new features for each sport and club combination.

Using parametric and non-parametric priors, we can therefore build increasingly complex models of networks in which the link weights are complicated sums of random variables—such as Equation 4.23—that are correlated between pairs of nodes (because they include the same features, sub-features etc). This is illustrated in Figures 4.3-4.6. Figure 4.3 shows the correlation between $\mathbf{F}_{i,j}$, the link weight between the i^{th} and j^{th} nodes, and $\mathbf{F}_{m,n}$ in a model with 2 features and one feature per node (or equivalently an SBM with 2 communities). The weights $\mathbf{W}$ are distributed $\mathcal{N}(0, 1/R^2)$ where R is the number of features. As can be seen, the link weight is either 100% correlated (if i is in the same community as m and

j is in the same community as n) or completely uncorrelated. The correlation coefficient ρ reflects how probable this is (in this case there is a 25% chance). In Figure 4.4 the correlation is roughly the same, however with 10 features and 5 per node the correlation appears more *jointly Gaussian*. If we decrease (Figure 4.5) or increase (Figure 4.6) the feature overlap we, increase or decrease the link correlation. Ultimately, it seems natural to model this correlation directly using a Gaussian process; this is the Random Function Model (RFM) for networks.

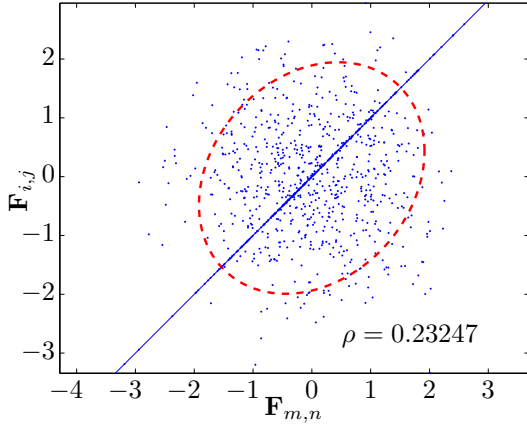


Figure 4.3: Link correlation with 2 features and 1 feature per node. As in IRM link weights are either 100% correlated (on line $y = x$) or completely uncorrelated.

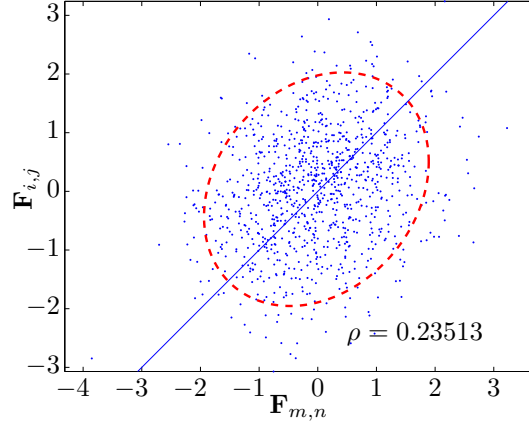


Figure 4.4: Link correlation with 10 features and 5 features per node. Moderate degree of correlation between link weights.

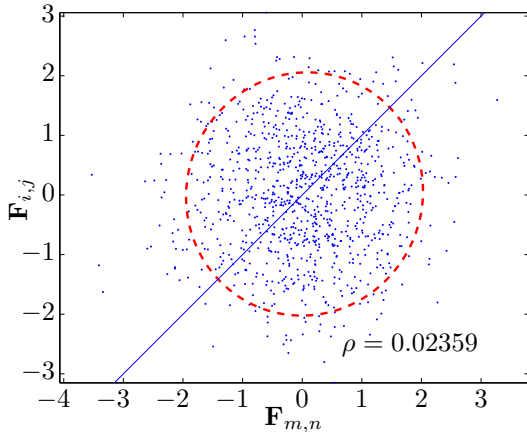


Figure 4.5: Link correlation: 10 features and 2 features per node. Nodes unlikely to share any features, very low degree of link correlation.

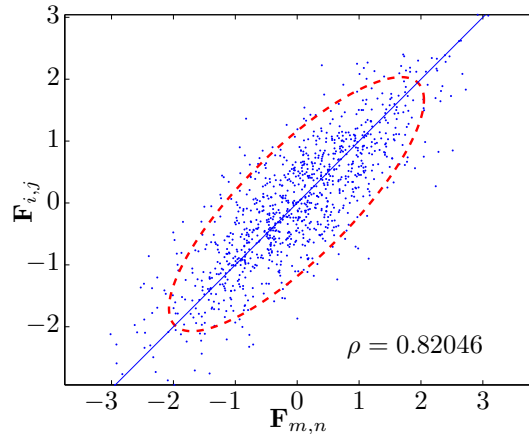


Figure 4.6: Link correlation with 10 features and 9 features per node. High degree of correlation between link weights.

4.3.2 The Random Function Model

As we saw in §4.2, by kernelising the input coordinates, we can construct a non-linear mapping between the latent variables and the data. A similar idea applied to networks is the RFM (Lloyd, Orbanz, et al. 2012). In the RFM, the network covariance structure is created using augmented latent coordinates constructed by tying latent variables together. Each node n is associated with a pair of latent coordinates $\mathbf{x}_s^{(n)}, \mathbf{x}_r^{(n)}$. We construct the augmented input vector for the link $m \rightarrow n$ from the sender, node m , to the receiver, node n , by concatenating the vectors $\mathbf{x}_s^{(m)}$ and $\mathbf{x}_r^{(n)}$. Using this set of input vectors we then draw a function from a GP in the normal way. For N nodes we obtain the N^2 length vector $\mathbf{f}$ with elements²

$$\mathbf{f} = [f([\mathbf{x}_s^{(1)}, \mathbf{x}_r^{(1)}]), f([\mathbf{x}_s^{(2)}, \mathbf{x}_r^{(1)}]), \dots, f([\mathbf{x}_s^{(N)}, \mathbf{x}_r^{(1)}]), \dots, f([\mathbf{x}_s^{(N)}, \mathbf{x}_r^{(N)}])]^\top. \quad (4.24)$$

We can reshape this vector to give the $[N \times N]$ matrix $\mathbf{F} \triangleq \text{vec}^{-1}(\mathbf{f})$. From the definition of the Gaussian process we see that the mean and covariance of the elements of $\mathbf{F}$ are:

$$\mathbb{E}[\mathbf{F}_{m,n}] = \mu([\mathbf{x}_s^{(m)}, \mathbf{x}_r^{(n)}]) \quad (4.25)$$

$$\text{Cov}[\mathbf{F}_{m,n}, \mathbf{F}_{i,j}] = \Sigma([\mathbf{x}_s^{(m)}, \mathbf{x}_r^{(n)}], [\mathbf{x}_s^{(i)}, \mathbf{x}_r^{(j)}]). \quad (4.26)$$

Thus the strength of the link $m \rightarrow n$ is similar to the link $i \rightarrow j$, if the augmented vector $[\mathbf{x}_s^{(m)}, \mathbf{x}_r^{(n)}]$ is close in the kernel input space to the augmented vector $[\mathbf{x}_s^{(i)}, \mathbf{x}_r^{(j)}]$.

The RFM can also be used to model symmetric networks. In this case we do not need to distinguish between sender and receiver latent coordinates and thus we can set $\mathbf{x}_s^{(n)} := \mathbf{x}_r^{(n)}$. Secondly, we need to ensure the covariance between the links $m \rightarrow n$ & $i \rightarrow j$ is the same as the covariance between the links $m \rightarrow n$ & $j \rightarrow i$, $n \rightarrow m$ & $i \rightarrow j$ and $n \rightarrow m$ & $j \rightarrow i$ for all pairs of links. This can be achieved using a symmetrised covariance function, such as the following sum kernel:

$$\begin{aligned} \Sigma_{\text{sym}}([\mathbf{x}_s, \mathbf{x}_r], [\mathbf{x}'_s, \mathbf{x}'_r]) = \frac{1}{4} [& K([\mathbf{x}_s, \mathbf{x}_r], [\mathbf{x}'_s, \mathbf{x}'_r]) + K([\mathbf{x}_s, \mathbf{x}_r], [\mathbf{x}'_r, \mathbf{x}'_s]) \\ & + K([\mathbf{x}_r, \mathbf{x}_s], [\mathbf{x}'_s, \mathbf{x}'_r]) + K([\mathbf{x}_r, \mathbf{x}_s], [\mathbf{x}'_r, \mathbf{x}'_s])]. \end{aligned} \quad (4.27)$$

²In practice it is only necessary for $\mathbf{f}$ to contain the links for which we have observed information.

To ensure the covariance matrix is positive definite we only evaluate the Σ_{sym} for the links $m \leq n$ and $i \leq j$ or else we may have repeated rows and columns and a rank deficient covariance. Since number of links in a network is $\mathcal{O}(N^2)$ the computational complexity of the RFM is $\mathcal{O}(N^6)$ and therefore becomes rapidly intractable. Approximate GP inference techniques, such as those described in §3.3, are therefore necessary. In their implementation Lloyd, Orbanz, et al. used the Deterministic Inducing Conditional (DIC) approximation.

The RFM bears some resemblance to latent space network models such as Hoff et al. (2002). The difference is that, in the RFM, the proximity of a pair nodes in the latent space increases the likelihood these nodes will share similar connections, but does not increase the likelihood of a connection. In contrast, in other latent space models, the likelihood of connectivity is directly governed by proximity in the latent space.

4.3.3 Dynamic network models

Dynamic networks are networks that change through time. In fact, many of the classic examples of networks—for example social networks—are dynamic networks. In many of these cases, the network is observed through interactions between pairs of nodes. Since the interactions are not simultaneous, these networks are inherently dynamic; we refer to these networks as *Dynamic Interaction Networks* (DIN). In spite of the ubiquity of such networks, probabilistic models for dynamic networks are relatively scarce.

4.3.3.1 Tensor decompositions

The simplest representation of a dynamic network is a 3-mode tensor, $\underline{\mathbf{Y}} \in \mathbb{R}^{I \times J \times K}$, whose elements, $y_{i,j,k}$, represent the existence or strength of a link from the i^{th} to the j^{th} node at the k^{th} time step.

To learn latent structure in this network we can apply tensor *decompositions* to $\underline{\mathbf{Y}}$. Unlike matrix factorisation, there is no single universally agreed definition of a tensor decomposition, however two common models are the CANDECOMP-PARAFAC (CP) and Tucker decompositions. A visualisation of these decompositions is shown in Figure 4.7. The CP decomposition is the sum of rank-1 outer products often expressed in Kronecker form. The CP decomposition is defined as:

$$\text{vec}(\underline{\mathbf{Y}}) \approx \sum_{c=1}^C \mathbf{t}_c \otimes \mathbf{r}_c \otimes \mathbf{s}_c, \quad (4.28)$$

where $\mathbf{t}_c$, $\mathbf{r}_c$ and $\mathbf{s}_c$ are column vectors and vec is the tensor vectorisation operator. In element-wise form the decomposition is:

$$y_{i,j,k} \approx \sum_{c=1}^C t_{k,c} r_{j,c} s_{i,c}. \quad (4.29)$$

These definitions can easily be extended to general P -mode tensors.

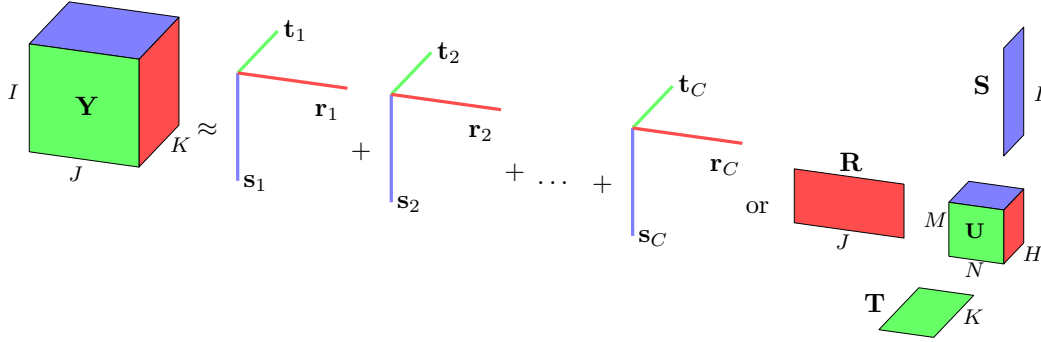


Figure 4.7: The CP (centre) and Tucker (right) tensor decompositions.

The CP decomposition is a special case of the Tucker decomposition. In the Tucker decomposition we obtain 3 matrices $\mathbf{S}$, $\mathbf{R}$ & $\mathbf{T}$, with dimensions $[I \times H]$, $[J \times M]$ & $[K \times N]$, in addition to a *core* tensor, $\underline{\mathbf{U}}$, with dimensions $[H \times M \times N]$, which we use to approximate $\underline{\mathbf{Y}}$ as

$$\text{vec}(\underline{\mathbf{Y}}) \approx [\mathbf{T} \otimes \mathbf{R} \otimes \mathbf{S}] \text{vec}(\underline{\mathbf{U}}) \quad (4.30)$$

$$\approx \sum_{h=1}^H \sum_{m=1}^M \sum_{n=1}^N [\mathbf{t}_n \otimes \mathbf{r}_m \otimes \mathbf{s}_h] u_{h,m,n}, \quad (4.31)$$

where $\mathbf{s}_h$, $\mathbf{r}_m$, $\mathbf{t}_n$ are the columns of the corresponding matrices. In element-wise form the decomposition is

$$y_{i,j,k} \approx [\bar{\mathbf{t}}_k \otimes \bar{\mathbf{r}}_j \otimes \bar{\mathbf{s}}_i] \text{vec}(\underline{\mathbf{U}}) \quad (4.32)$$

$$\approx \sum_{h=1}^H \sum_{m=1}^M \sum_{n=1}^N t_{k,n} r_{j,m} s_{i,h} u_{h,m,n}, \quad (4.33)$$

where $\bar{\mathbf{t}}_k$, $\bar{\mathbf{r}}_j$ and $\bar{\mathbf{s}}_i$ are the rows of the corresponding matrices.

The CP and Tucker decompositions are equivalent when $\underline{\mathbf{U}}$ is the identity tensor with entries $u_{h,m,n} = \mathbb{1}\{h = m = n\}$.

Kruskal (1977) showed that the CP decomposition³ “in most typical statistical situations” is unique up to permutation and scaling⁴ non-identifiabilities of the components since “the coordinate system has a special status, and free rotation is not possible.” As described by Kroonenberg (1983) “the importance of this uniqueness theorem is that if the model fits well the good fit only refers to the solution found, and not to infinitely many of them.”

The Tucker decomposition is however, non-unique in general since we can rotate any of the components and the corresponding dimension of the core matrix. For example, we can use an $[H \times H]$ orthogonal rotation matrix $\mathbf{M}$ to rotate the first component yielding a new matrix $\tilde{\mathbf{S}}$ and core tensor $\tilde{\underline{\mathbf{U}}}$. If we substitute the elements of these matrices into the element-wise definition of the tucker decomposition and rearrange as follows:

$$\begin{aligned} \sum_{h=1}^H \sum_{m=1}^M \sum_{n=1}^N t_{k,n} r_{j,m} \tilde{s}_{i,h} \tilde{u}_{h,m,n} \\ = \sum_{h=1}^H \sum_{m=1}^M \sum_{n=1}^N \left[\sum_{d=1}^H s_{i,d} m_{d,h} \right] \left[\sum_{g=1}^H m_{h,g} u_{g,m,n} \right] r_{j,m} t_{k,n} \end{aligned} \quad (4.34)$$

³Under the Frobenius norm.

⁴Specifically we can multiply $\mathbf{s}_c$, $\mathbf{r}_c$, $\mathbf{t}_c$ by scalars λ_c , μ_c , ν_c for which $\lambda_c \mu_c \nu_c = 1$ holds.

$$= \sum_{m=1}^M \sum_{n=1}^N \left[\sum_{d=1}^H \sum_{g=1}^H \underbrace{\left[\sum_{h=1}^H m_{h,d} m_{g,h} \right]}_{\mathbb{1}\{d=g\}} s_{i,d} u_{g,m,n} \right] r_{j,m} t_{k,n} \quad (4.35)$$

$$= \sum_{m=1}^M \sum_{n=1}^N \sum_{h=1}^H t_{k,n} r_{j,m} s_{i,h} u_{h,m,n}, \quad (4.36)$$

we recover the same tensor approximation of $\underline{\mathbf{Y}}$ and hence show the decomposition is indeed non-unique.

We can interpret both the Tucker and CP decompositions in the context of DIN. In the CP decompositions the vector $\mathbf{t}_c$ represents the dynamic behaviour of the c^{th} community and the vectors $\mathbf{s}_c$ and $\mathbf{r}_c$ determine the how much of this behaviour the sending and receiving nodes exhibit. In the Tucker representation the core-tensor is a set of temporally compressed inter and intra community interactions—analogueous to matrix $\mathbf{W}$ of the SBM—while the matrices $\mathbf{S}$ and $\mathbf{R}$ are the associated sender and receiver weights. Temporal compression in the Tucker decomposition allows repeated activities to be distilled. For example, if $\underline{\mathbf{Y}}$ contains the daily interactions of a social network with, say, the number of observed time steps $K = 7 \times 24 = 168$ and the corresponding dimension of $\underline{\mathbf{U}}$, $N = 24$, then the elements $u_{h,m,1}, \dots, u_{h,m,24}$ collectively *may* represent the daily pattern of interaction between the h^{th} and m^{th} communities. The matrix $\mathbf{T}$ contains the mappings that expands this weekly compressed representation back up to observed uncompressed daily representation.

A variety of tensor factorisation algorithms have been developed based on a variety of inference methods and constraints. An overview of some of these methods is given in Table 4.2. One approach to discovering structure in DIN, is to directly apply one of these decomposition algorithms. There are a number of such studies in the literature for including Kolda and Sun (2008), Sizov et al. (2010), Lin et al. (2010), Hoff (2015). In most cases the CP decomposition is used, although Kolda and Sun (2008) investigates both the CP and Tucker decompositions. Particularly noteworthy is the Bayesian Poisson Tensor Factorisation (BPTF) model of Schein et al. (2015), which was developed specifically for modelling DIN. Unlike most tensor

decompositions, Schein et al. use a Poisson likelihood to model the approximation, Equation 4.30, which is a more appropriate choice for tensors that contain count data. As a natural consequence, their model is a positive tensor decomposition. They also developed a scalable Bayesian variational inference algorithm by extending the work of Cemgil (2009) to the CP tensor decomposition. Bayesian approaches are particularly desirable for tensor decompositions because Bayesian Automatic Relevance Determination (ARD) helps to shrink away the large number of latent variables that arise in such models. This is particularly important in the case of DIN tensors, which may be very sparse.

Method	Decomp.	Error model	Constraints	Algorithmic details
HO-SVD ¹	Tucker	$ \cdot _F$	Orthogonality	Repeated SVD
PTF ²	CP	$ \cdot _F$	$\oplus$ Components	Multiplicative updates
NTF ³	CP	$ \cdot _F$	$\oplus$ Components	Gradient decent & EM
pTucker ⁴	Tucker	$ \cdot _F$	$\oplus$ Components	EM
NTFMCMC ⁵	CP	$ \cdot _F$	$\oplus$ Components	Bayesian, MCMC
BPTF ⁶	CP	Poisson	$\oplus$ Components	Bayesian, VB

¹ Lathauwer et al. (2000) ; ² Welling and Weber (2001) ; ³ Shashua and Hazan (2005) ; ⁴ Chu and Ghahramani (2009) ; ⁵ Schmidt and Mohamed (2009) ; ⁶ Schein et al. (2015)

Table 4.2: Tensor decomposition algorithms.

4.3.3.2 State-space models

While tensor decompositions have some advantages, particularly in terms of scalability, these models do not include transition models for the network state from one time step to the next. Indeed, we could reorder $\mathbf{Y}$ along its temporal axis and this would make no difference to the computed components, except that they would also be equivalently reordered. The network dynamics are implicit only in the representation of the data as a tensor and not explicitly enforced, for example, via state-space transition models that are part of most dynamic system models. A widely cited example of probabilistic generative model for dynamic networks using an explicit state-space modelling approach is the Dynamic Mixed Membership Stochastic Block Model (DMMSBM) of Fu et al. (2009). As the name suggests, it is a dynamic version of the earlier Mixed Membership Stochastic Block Model

(MMSBM) (Airoldi et al. 2008), which allows node community assignments and community-community interaction intensities to vary through time. The model is shown graphically in Figure 4.8.

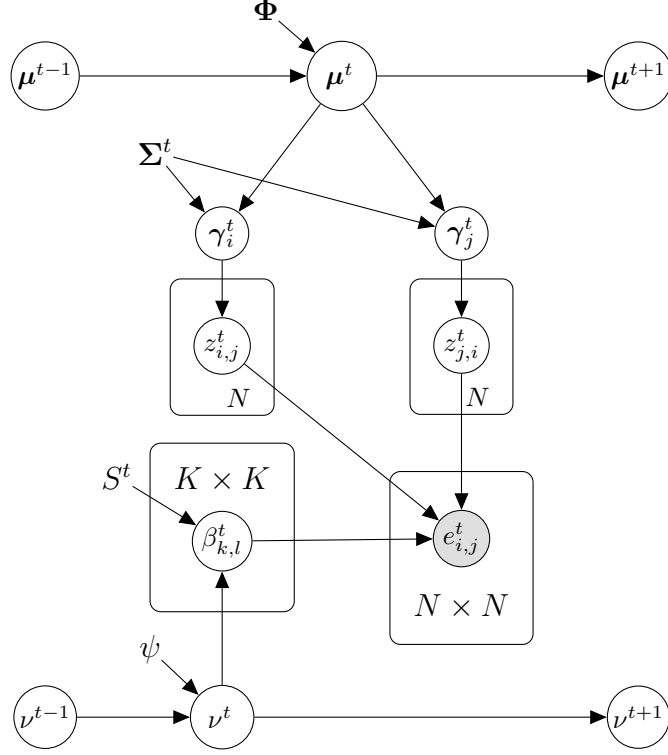


Figure 4.8: Graphical model of the Dynamic Mixed Membership Stochastic Block Model (DMMSBM) (Fu et al. 2009).

The first dynamic quantity, $\boldsymbol{\mu}^t \in \mathbb{R}^K$, is the mean of the community context vector $\gamma_i^t \sim \mathcal{N}(\boldsymbol{\mu}^t, \boldsymbol{\Sigma}^t)$ for $i \in \{1, \dots, N\}$. N is the number of nodes and K is the number of communities. The second dynamic quantity, $\nu^t \in \mathbb{R}$ is the mean of the community-community interaction weights $\beta_{k,l}^t \sim \mathcal{N}(b\nu^t, S^t)$, with $k, l \in \{1, \dots, K\}$. The variables $\boldsymbol{\mu}^t, \nu^t$ have linear Gaussian dynamic transitions $\boldsymbol{\mu}^t \sim \mathcal{N}(\mathbf{A}\boldsymbol{\mu}^{t-1}, \boldsymbol{\Phi})$ and $\nu^t \sim \mathcal{N}(b\nu^{t-1}, \psi)$, where $\mathbf{A}$ and b are the state transition matrices (b is a scalar) and $\boldsymbol{\Phi}$ and ψ are the transition (process) noise matrices (ψ is a scalar). The variables γ_i^t and $\beta_{i,j}^t$ are transformed using the softmax function⁵:

$$^*\gamma_{i,k}^t = \frac{\exp(\gamma_{i,k}^t)}{\sum_{k'=1}^K \exp(\gamma_{i,k'}^t)}, \quad ^*\beta_{k,l}^t = \frac{\exp(\beta_{k,l}^t)}{\sum_{k'=1}^K \sum_{l'=1}^K \exp(\beta_{k',l'}^t)}, \quad (4.37)$$

⁵ γ_i^t is, in effect, the natural parameter of a categorical distribution.

and are used to draw the community context for the link $i \rightarrow j$, $z_{i,j}^t \sim \text{Cat}(*\gamma_i^t)$ (similarly for $j \rightarrow i$). Finally the link states $e_{i,j}^t$ are sampled according to $\text{Bern}(*\beta_{z_{i,j}^t, z_{j,i}^t}^t)$.

In this generative process, the expected probability of a link $i \rightarrow j$ at time step t is

$$\mathbb{E}[e_{i,j}^t] = p_{i,j}^t = \sum_{k=1}^K \sum_{l=1}^K * \gamma_{i,k}^t * \beta_{k,l}^t * \gamma_{j,l}^t, \quad (4.38)$$

and therefore over time steps $1, \dots, N_t$ the expected number of interactions is $\sum_{t=1}^{N_t} p_{i,j}^t$. Due to the relationship between repeated Bernoulli trials and the Poisson distribution, as we increase the time fidelity of the model (i.e. decrease Δ_t) and allow $p_{i,j}^t$ to fall in proportion (i.e. decrease as Δ_t decrease), then the expected number of link connections, $\bar{N}$, over an interval $[0, T]$, $T = N_t \Delta_t$, will become increasingly well approximated by the integral of an inhomogeneous Poisson process intensity function. We can express this loosely as:

$$\bar{N} = \mathbb{E} \left[\sum_{t=1}^{N_t} e_{i,j}^t \right] = \sum_{t=1}^{N_t} p_{i,j}^t \approx \int_0^T \lambda_{i,j}(t) dt \text{ as } N_t \rightarrow \infty. \quad (4.39)$$

Since the interaction expectations, $p_{i,j}^t$, are correlated, their continuous time equivalents, $\lambda_{i,j}(t)$, will also be correlated. This motivates the study of dynamic networks via correlated Poisson process intensity functions. We are therefore interested in constructing correlated intensity function models, especially those that contain latent structures that reflect an underlying network topology. We will return to these questions in the following chapters.

4.3.4 Exchangeability and properties of networks

In the previous section we considered reordering the dimensions of a tensor representing a DIN. In the context of tensor factorisations, we argued that the particular labelling of a temporal dimension does not materially affect the tensor decomposition. As discussed in Chapter 2, the ability to reorder (relabel) data points and not affect the underlying probability of the data is called exchangeability. Matrices and higher ordered tensors can also be exchangeable. Exchangeability is more complicated in

these cases however, as we can distinguish between relabelling schemes that affect individual dimensions or multiple dimensions simultaneously.

If we can permute the rows and columns of a matrix of random variables, which we shall denote $(Y_{i,j})$ $i, j \in \mathbb{N}$, using separate permutations π and π' such that

$$(Y_{i,j}) \stackrel{d}{=} (Y_{\pi(i),\pi'(j)}), \quad (4.40)$$

the matrix is called *separately exchangeable*. If we must use the same permutation

$$(Y_{i,j}) \stackrel{d}{=} (Y_{\pi(i),\pi(j)}), \quad (4.41)$$

the matrix is called *jointly exchangeable*.

For adjacency matrices in which the edges connect nodes in the same set, joint exchangeability applies since, if we permute the rows and columns of an adjacency matrix independently we will, very likely, obtain a graph that will look very different. For graphs in which the edges connect nodes in difference sets, i.e. bipartite graphs, as well as other many other types of multivariate data, separate exchangeability applies. The extension of exchangeability to matrices and more generally d-arrays (p-mode tensors) is formalised by Aldous-Hoover Theorem.

Aldous-Hoover Theorem asserts that if a matrix $(Y_{i,j})$ is jointly exchangeable, there exists a random function, F , and a sequence and triangular matrix of uniform random variables, (U_i) , and $(U_{\{i,j\}})$, independent and identically distributed $\mathcal{U}([0, 1])$, that are independent of F , such that:

$$(Y_{i,j}) \stackrel{d}{=} (F(U_i, U_j, U_{\{i,j\}})). \quad (4.42)$$

The notation $U_{\{i,j\}}$ is used to indicate that $U_{\{i,j\}}$ is the same random variable as $U_{\{j,i\}}$ (since sets have no ordering). If F is symmetric in its first two arguments, the matrix is symmetric, otherwise it is asymmetric. If $(Y_{i,j})$ is separately exchangeable then the following holds:

$$(Y_{i,j}) \stackrel{d}{=} (F(U_i^{\text{row}}, U_j^{\text{col}}, U_{i,j})). \quad (4.43)$$

where (U_i^{row}) and (U_j^{col}) are two different sequences of random variables and $(U_{i,j})$ is a full matrix of random variables.

For the SBM the random matrix $\mathbf{W}$ plays the part of the random function F . The community assignments $\mathbf{Z}$ are analogous to the random variables (U_i) and the element-wise observation noise is provided by U_{ij} .

An important corollary of Aldous Hoover Theorem for networks is that statistical models of networks that are node-exchangeable—i.e. one for which the rows and columns of the adjacency matrix can be separately or jointly permuted without affecting its likelihood—are either *dense* or they are empty (with probability 1). In the networks literature the terms dense and sparse have particular meanings. A dense network is one for which the number of edges as function of the number of nodes grows at least in proportion to the total number of possible edges, i.e. $|\mathcal{E}| = \Omega(N^2)$. A sparse network on the other hand is one for which the number of links grows more slowly than the number of possible edges, i.e. $|\mathcal{E}| = o(N^2)$. This result follows directly from Equation 4.42. If $Y_{i,j} \in \{0, 1\}$ is an exchangeable array, then we have $\mathbb{E}[Y_{i,j}] = \mathbb{E}[F(U_i, U_j, U_{\{i,j\}})] = p$, for all i, j and so the expected number of edges must be pN^2 . Of course, p may be very small and so a particular adjacency matrix may be sparse in the traditional sense of containing lots of zeros, yet the corresponding network model is still dense.

Most generative models of networks, including most based on the SBM and the Gaussian process based RFM, are node-exchangeable and hence they are models of dense networks. However, many types of network are observed to be sparse. This lead Orbanz and Roy (2015) to observe “graphs representing network data typically have a finite number of edges per vertex, and exhibit properties like power-laws and ‘small-world phenomena’, which can only occur in sparse graphs. Hence, even though exchangeable graph models are widely used in network analysis, they are inherently misspecified.” Probabilistic schemes for sparse networks do exist, an example is the Degree Corrected Stochastic Block Model Karrer and Newman (2011), however such models are finitely exchangeable, but not *projective* (Caron

and Fox 2014). Projective in this context means that if a node is marginalised from a model of size N , the result does not have the same distribution as a model of size $N - 1$.

An important avenue of new research based on a different notion of exchangeability, Kallenberg-exchangeability, can yield models that are both sparse and projective (See Caron (2012) and Caron and Fox (2014), Cai et al. (2016) and the references there-in). Recent work by Herlau et al. (2016) extends block structure to the model of Caron and Fox (2014), dramatically improving link prediction performance. This suggests that block structure may often be a more important modelling consideration than sparsity and although inherently misspecified, stochastic block models remain a powerful network modelling technique. As George Box famously observed “all models are wrong, but some are useful” (Box and Draper 1987).

5

Convolutional point processes

In Chapter 1, we stated our aim of developing computationally tractable algorithms to infer the intensity of complex event processes—a task we will begin in this chapter. We were interested in constructing models that share information between similar but distinct, event processes. By building on the foundations laid out in Chapters 2-4, we will capture hidden *dependency structures* that relate multiple event processes together.

In this chapter we will develop our first *structured point processes* model for complex event data, the Convolutional Point Process (CPP) model (Gunter, Lloyd, et al. 2014). We will use *convolutional Gaussian processes*, an extension of the Gaussian Process (GP) framework introduced in Chapter 2, to connect the intensity functions of multiple related point processes. As we shall see, this improves event intensity estimation in comparison with independent modelling.

While the CPP algorithm has some strengths—it is asymptotically unbiased and can be adapted to a variety of point process model types—it leaves ample room for improvement and so we will conclude this chapter with a short review of the weaknesses of this first model. This will serve to motivate directions for the further developments that are presented in subsequent chapters.

We will start with an introduction to structured point processes, their relationship to complex event data and dynamic networks, and the challenges we face.

5.1 Structured point processes

Complex data is data that is tractable to analysis by decomposition into simpler, more easily understood structures. We can compare complex data to a complex piece of software. Although the software may solve a complicated problem, it can be easily understood if it is comprised of a series of loosely coupled modules, each of which solves a well-defined sub-problem. In Chapter 4, we saw how latent variable models, such as matrix factorisations, can be used to gain insight into complex data. Latent variables allow us to capture the hidden structures that give rise to complexity. In this chapter, we will begin to apply similar principles to continuous point processes, specifically Poisson point processes.

Figure 5.1 depicts our canonical complex spatial event process—one that we will revisit several times in this work—basketball shot data. The figure shows the position from which baskets were scored by three professional NBA players. The players’ shot profiles are not independent of each other, since after all, they are playing the same game. However, each player has their own scoring pattern that is partly a reflection of the position they play in and partly their own unique style. For example, some players show a much greater propensity to shoot from distance, outside the three-point line, while others score mainly close to the basket. For basketball, the rules of the game, team tactics and the relative ease of scoring closer to the hoop are the real world factors that influence shot location. As these factors are easy to understand, basketball shot data is a useful sanity test for our algorithms.

Dynamic Interaction Networks (DINs) are another type of complex event processes we are interested in. Networks are perhaps the quintessential complex system because of the proliferation of pairwise relationships. For the first time, in Figure

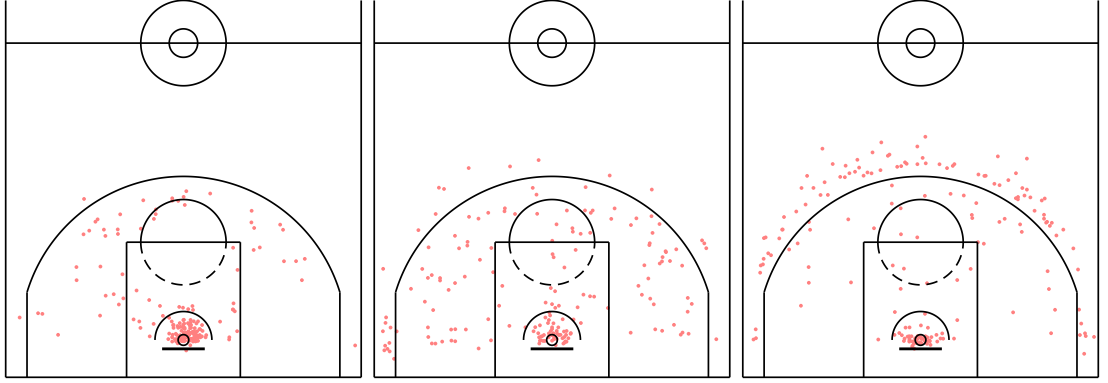


Figure 5.1: Basketball court markings and overlaid shot data for three NBA players, left-to-right: Blake Griffin, DeMar DeRozan and Damian Lillard.

5.2b, we now depict a DIN explicitly as a collection of point processes. This network data is *synthetic*, drawn from the generative distribution of a community structured Cox process network model. It is a mixed membership model with two communities: nodes 1 to 3 attach more strongly to the first community and the nodes 4 to 6 more strongly to the second community. The interaction intensity between nodes that attach more to the same community, is, on average, greater than the interaction intensity between nodes that attach more to different communities. Therefore the corresponding weighted adjacency matrix, Figure 5.2a, exhibits block structure.

Figure 5.2b reveals the hidden community labels associated with intra-community and inter-community interactions on three separate links. Figures 5.2c, 5.2d and 5.2e show the interaction intensities for the links highlighted in Figure 5.2b. The intensity for each link is the superposition of latent intensity functions, weighted by the community attachment strengths for the nodes the link connects. The inference task in such a model is therefore to learn the structures—the latent functions and the communities’ weights—that explain the complex pattern of interactions depicted.

Figure 5.2b shows another network in which inter and intra-community interactions have the same intensity on average. One set of inter-community interactions occur between the start and t_1 , the second set between t_1 and t_2 , and the intra-community interactions occur between the t_2 and the end. In this network, the community

membership is immediately apparent; nodes 1 to 3 are members of one community and nodes 4 to 6 another. However count information alone is insufficient to reveal the structure of this network, which can only be done using temporal information. On the other hand, if all interaction event processes are homogeneous Poisson processes, the use of temporal information provides no benefit in comparison to static methods, such as the Stochastic Block Model (SBM).

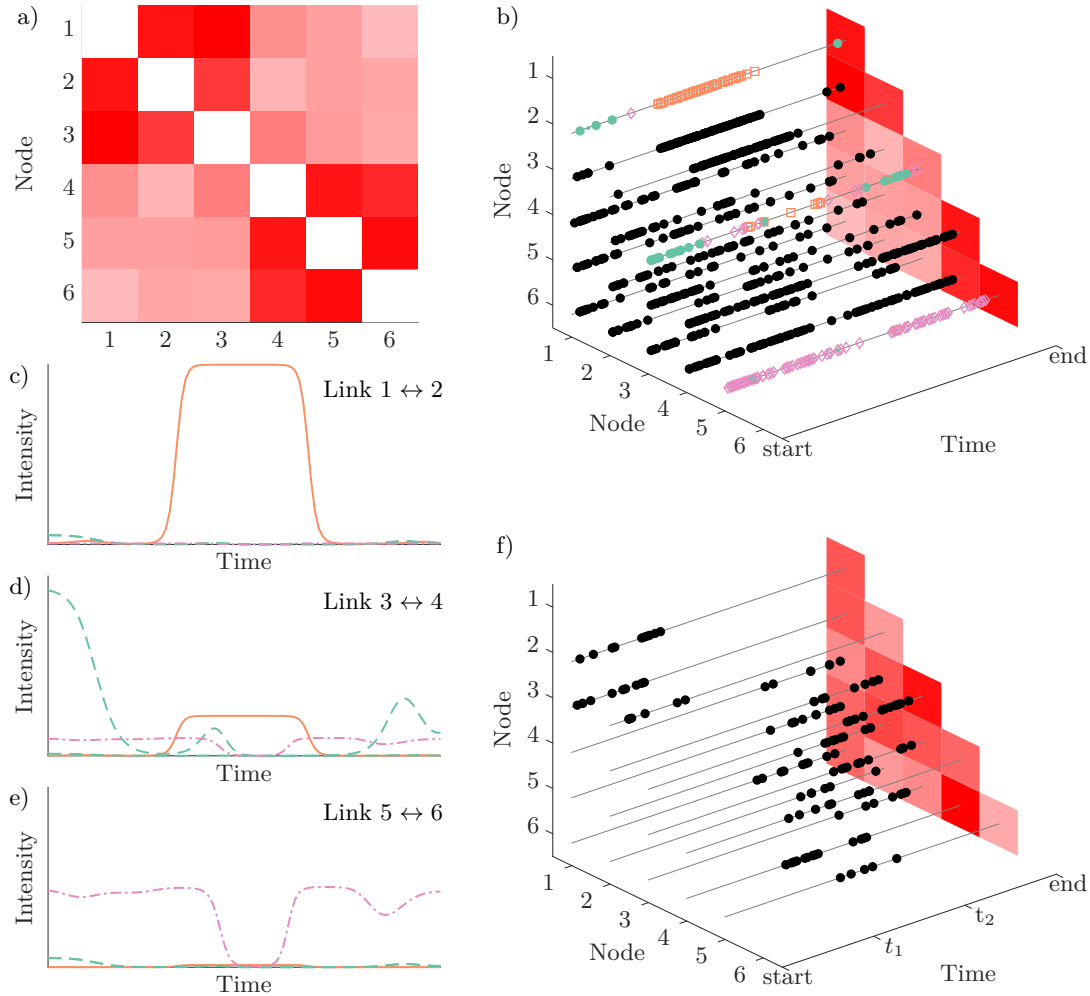


Figure 5.2: a) Adjacency matrix of interaction counts
 b) Point process representation of a Dynamic Interaction Network (DIN). Inter community interactions are show by orange squares and pink diamonds, while intra-community interactions are shown by green dots.
 c, d, e) Event intensities for highlighted links. The solid orange line corresponds to the orange squares in (b), the dash-dotted pink line to the pink squares and the dashed green lines to the green dots.
 f) A network that has structure that can only be distinguished using temporal information.

5.1.1 Challenges

Having described structured point processes and their usefulness for understanding complex event processes, we now turn our attention to technical solutions and the challenges we face. The solutions proposed in the following pages are Bayesian inference algorithms for continuous Cox processes with Gaussian process intensity functions. This framework is both appealing and challenging.

The GP approach is appealing because it allows us to impose smoothness-constraints and avoid overfitting. This is particularly important in Cox process models, since each data point provides relatively little information about the intensity function and other model parameters. The GP also allows us to model continuous intensity functions and avoid the need to pre-process the data into discrete bins or make decisions about the fidelity of the discretisation. Gaussian processes naturally extend to higher dimensional spaces where these advantages become even more pronounced. In higher dimensions, lower data densities leave us more vulnerable to overfitting and the importance of avoiding the discretisation of the input domain becomes more acute (due to the curse of dimensionality). As we shall see, the GP approach leads to options beyond linear factor models for modelling the correlations between the intensities of the multiple observed event processes.

The GP approach is challenging primarily for two reasons: the non-conjugate, doubly intractable Cox process likelihood and the poor computational scaling of GP inference. In addition to several structured Cox process models that we will introduce, the main developments to be highlighted are those that address these challenges. In this chapter, we will address the doubly intractable integral using an adaptive thinning method (§5.2.4) and manage computational complexity using sparse Gaussian processes (§5.2.2.2). Later, the combination of variational inference and a judicious link function choice will solve these challenges simultaneously. These developments transform the modelling approach from an elegant but otherwise impractical theoretical framework into a viable technical solution. It is still necessary however, to engineer efficient software implementations that make efficient use of

computational resources, for example, by identifying opportunities for parallel computation.

5.2 Convolutional point processes

In this section we will present the Convolutional Point Process (CPP) model for complex, structured event data. As previously described, a typical method of constructing models of complex data is using factor models. The extension of discrete factor models to Gaussian processes is known as a Semi-Parametric Factor Model (SPFM) (Teh, Seeger, et al. 2005). In the SPFM, each observed function is a weighted sum of base Gaussian processes. Convolutional Gaussian processes¹ generalise this idea by—as the name suggests—convolving and then summing the base functions (referred to from now on as the *latent* functions) together. An example of a convolution process is shown in Figure 5.3. The convolution blurs and/or shifts the latent functions and allows a wide range of output functions to be generated from a small number of latent functions.

5.2.1 Model augmentation using thinning

Before we see the full details of the convolution process, we will first introduce an approach to tackling the doubly intractable integral in the Cox process likelihood, Equation 2.84. As we saw in Chapter 2, one method for drawing samples from a inhomogeneous Poisson process was to “thin” a homogeneous Poisson process by rejection sampling. This insight leads to an inference algorithm for Cox processes where we treat the thinned (rejected) points as *missing data*, which we re-introduce as hidden random variables. If we do this, we find that we can write down the likelihood, Equation 2.83, in closed-form (Adams et al. 2009).

Each event process has an observed set of points $\mathcal{D}_s = \{\mathbf{x}^{(s,n)}\}_{n=1}^{N_s}$. We treat both the number, J_s , and locations, $\mathcal{T}_s = \{\tilde{\mathbf{x}}^{(s,j)}\}_{j=1}^{J_s}$, of the thinned points of each event

¹From now referred to as simply convolution processes.

process as random variables. For each event process there are $J_s + N_s$ points in total. Since the likelihood depends only on the value of the intensity function at these locations, we only need to work with finite dimensional distributions. The joint distribution of the data, the thinned points and the finite set of points $\mathbf{f}_s = [f_s(\mathbf{x}^{(s,1)}), \dots, f_s(\mathbf{x}^{(s,N_s)}), f_s(\tilde{\mathbf{x}}^{(s,1)}), \dots, f_s(\tilde{\mathbf{x}}^{(s,J_s)})]^\top$ is

$$p(\mathcal{D}_s, \mathcal{T}_s, J_s, \mathbf{f}_s | \lambda_s^*, \Omega) = (\lambda_s^*)^{N_s + J_s} \exp(-\lambda_s^* |\mathcal{X}_s|) \times \mathcal{N}(\mathbf{f}_s; \boldsymbol{\mu}_s, \mathbf{K}_s) \\ \times \prod_{n=1}^{N_s} \sigma(f_s(\mathbf{x}^{(s,n)})) \prod_{m=1}^{M_s} \sigma(-f_s(\tilde{\mathbf{x}}^{(s,j)})), \quad (5.1)$$

where $\boldsymbol{\mu}_s$ and $\mathbf{K}_s$ are determined by the convolution process prior and are yet to be defined. The intensity function for each event process is $\lambda_s = \lambda_s^* \sigma(f_s)$, where σ is the sigmoid squashing function and λ_s^* is the maximum value of the intensity. We will now define the convolution process prior for our correlated output functions f_s and then show how posterior distributions of the intensity functions and model hyper parameters can be computed by integrating out the number and locations of the thinning points using Markov-Chain Monte-Carlo (MCMC).

5.2.2 Convolution processes

We will model each output function, f_s , as the sum of the convolution of T latent functions, $u_{1:T}$, which we will assume are independently distributed Gaussian processes:

$$u_t(\mathbf{z}) | \phi_t \sim \mathcal{GP}(u_t(\mathbf{z}); 0, K_t(\mathbf{z}, \mathbf{z}')), \quad (5.2)$$

where $K_t(\mathbf{z}, \mathbf{z}') = \mathcal{N}(\mathbf{z}; \mathbf{z}', \mathbb{I}\phi_t)$ is an isotropic Gaussian kernel.

If the latent functions are GP distributed, the output functions will also be Gaussian processes (Mauricio and Lawrence 2011). The convolution is achieved using a *convolution kernel*, $G(\mathbf{x}, \mathbf{z})$, where $\mathbf{z} \in \mathcal{Z}$, the space over which the latent functions are defined and $\mathbf{x} \in \mathcal{X}$, the space over which the S dependent output functions, $f_{1:S}(\mathbf{x})$, are defined. Typically $\mathbf{z}$ is defined on the same domain as $\mathbf{x}$, i.e. $\mathcal{X} = \mathcal{Z}$, and

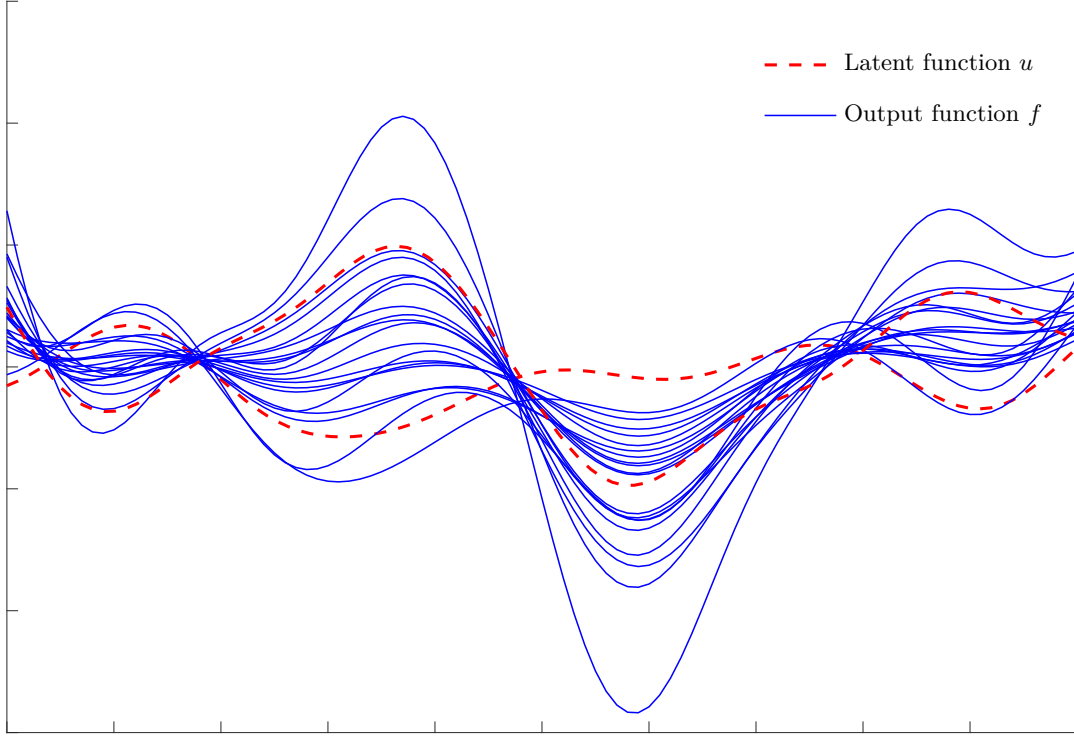


Figure 5.3: Draws from a convolution process showing two latent functions (red dashed lines), and 20 output functions (blue continuous lines).

$T < S$. The output of the s^{th} output function, $f_s(\mathbf{x})$, is therefore the convolution:

$$f_s(\mathbf{x}) = \sum_{t=1}^T \kappa_{s,t} \int_{\mathbb{R}^R} G_s(\mathbf{x}, \mathbf{z}) u_t(\mathbf{z}) \, d\mathbf{z}. \quad (5.3)$$

where $G_s(\mathbf{x}, \mathbf{z})$ is the convolution kernel associated with the s^{th} output. Since convolution is a deterministic transformation, given complete knowledge of the latent functions, the output functions are deterministic and independent.

5.2.2.1 Sparse latent functions

We cannot represent the latent functions, $u_{1:T}$, exactly since they are infinite dimensional objects, so instead we approximate them with finite dimensional samples, $\mathbf{u}_t = [u(\mathbf{z}^{(t,1)}), \dots, u(\mathbf{z}^{(t,M)})]$, where $\mathcal{Z}_t = \{\mathbf{z}^{(t,m)}\}_{m=1}^M$ is a set of inducing points. We can sample from $p(\mathbf{u}_t \mid \mathcal{Z}_t)$ since it is a conditional Gaussian distribution and then solve the convolution integral by approximating each $u_t(\mathbf{z})$ by replacing it with its

expected value, given a finite dimensional sample. Therefore Equation 5.3 becomes:

$$f_s(\mathbf{x}) \approx \sum_{t=1}^T \kappa_{s,t} \int_{\mathbb{R}^R} G_s(\mathbf{x}, \mathbf{z}) \mathbb{E}[u_t(\mathbf{z}) | \mathbf{u}_t, \mathcal{Z}_t] d\mathbf{z}. \quad (5.4)$$

This is reasonable as long as the latent functions are smooth and well approximated given the covariance function and the finite dimensional samples $\mathbf{u}_t$. For the convolution kernel we use a Gaussian kernel

$$G_s(\mathbf{x}, \mathbf{z}) = \mathcal{N}(\mathbf{x}; \mathbf{z}, \mathbb{I}\theta_s), \quad (5.5)$$

and so the covariance linking $u_t(\mathbf{z})$ to $f_s(\mathbf{x})$ is the convolution of two Gaussian distributions. As described in Chapter 2, this results in another Gaussian distribution and so the kernel function linking the latent t^{th} latent function and output s^{th} function is

$$K_{f_s, u_t}(\mathbf{x}, \mathbf{z}) = \kappa_{s,t} \int_{\mathbb{R}^R} G_s(\mathbf{x}, \mathbf{z}') K_t(\mathbf{z}', \mathbf{z}) d\mathbf{z}' \quad (5.6)$$

$$= \kappa_{s,t} \mathcal{N}(\mathbf{x}; \mathbf{z}, \mathbb{I}\theta_s + \mathbb{I}\phi_t). \quad (5.7)$$

Note that the kernel hyper-parameters θ_s and ϕ_t are both positive, therefore the output functions constrained to be blurred combinations of the latent functions.

We could also compute the convolution kernel *between* output functions, which is given by the double integral:

$$K_{f_s, f_{s'}}(\mathbf{x}, \mathbf{x}') = \sum_{t=1}^T \kappa_{s,t} \kappa_{s',t} \int_{\mathbb{R}^R} \int_{\mathbb{R}^R} G_s(\mathbf{x}, \mathbf{z}) K_t(\mathbf{z}, \mathbf{z}') G_{s'}(\mathbf{z}', \mathbf{x}') d\mathbf{z} d\mathbf{z}' \quad (5.8)$$

$$= \sum_{t=1}^T \kappa_{s,t} \kappa_{s',t} \mathcal{N}(\mathbf{x}; \mathbf{x}', \mathbb{I}\theta_s + \mathbb{I}\theta_{s'} + \mathbb{I}\phi_t), \quad (5.9)$$

and use this covariance function to construct a single large joint GP over all the intensity functions. However, the resulting inference problem scales computationally as $\mathcal{O}((N + TM)^3)$ with storage requirements of $\mathcal{O}((N + TM)^2)$, where $N = \sum_{s=1}^S (N_s + J_s)$ is the total number of observed events and thinned points across all outputs. Under this full dependence assumption, each event is linked to both inter and intra-process data and inference will soon become intractable.

5.2.2.2 Partially independent outputs

We can improve computational performance by using the Partially Independent Conditional (PIC) scheme described in Chapter 3. Under this assumption, the output functions, $f_{1:S}$, are conditionally independent given the latent processes, $u_{1:T}$. We will maintain the full GP for the output intensity functions, while summarising the latent functions using the approximation of Equation 5.4.

Importantly, this assumption allows us to reduce the computational scaling of inference to $\mathcal{O}(N_{\max}^3 + M^3 + MN_{\max}^2)$, with storage requirements of $\mathcal{O}(N_{\max}^2 + M^2 + MN_{\max})$, where $N_{\max} = \max_s(N_s + J_s)$. Further reductions may be made by using the Fully Independent Conditional (FIC) approximation. This may be useful if the number of events in a single output is still too large to be modelled jointly using a full GP.

In the PIC approximation, the covariance between outputs is neglected and therefore the kernel given by Equation 5.9 is not required. Under the PIC scheme the joint covariance between output functions and latent functions has the following block structure:

$$\left[\begin{array}{cccc|cccc} \mathbf{K}_{f_1, f_1} & \mathbf{Q}_{f_1, f_2} & \dots & \mathbf{Q}_{f_1, f_S} & \mathbf{K}_{f_1, u_1} & \mathbf{K}_{f_1, u_2} & \dots & \mathbf{K}_{f_1, u_T} \\ \mathbf{Q}_{f_2, f_1} & \mathbf{K}_{f_2, f_2} & \dots & \mathbf{Q}_{f_2, f_S} & \mathbf{K}_{f_2, u_1} & \mathbf{K}_{f_2, u_2} & \dots & \mathbf{K}_{f_2, u_T} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{Q}_{f_S, f_1} & \mathbf{Q}_{f_S, f_2} & \dots & \mathbf{K}_{f_S, f_S} & \mathbf{K}_{f_S, u_1} & \mathbf{K}_{f_S, u_2} & \dots & \mathbf{K}_{f_S, u_T} \\ \hline \mathbf{K}_{u_1, f_1} & \mathbf{K}_{u_1, f_2} & \dots & \mathbf{K}_{u_1, f_S} & \mathbf{K}_{u_1, u_1} & 0 & \dots & 0 \\ \mathbf{K}_{u_2, f_1} & \mathbf{K}_{u_2, f_2} & \dots & \mathbf{K}_{u_2, f_S} & 0 & \mathbf{K}_{u_2, u_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{K}_{u_T, f_1} & \mathbf{K}_{u_T, f_2} & \dots & \mathbf{K}_{u_T, f_S} & 0 & 0 & \dots & \mathbf{K}_{u_T, u_T} \end{array} \right],$$

where $\mathbf{K}$ denotes a covariance matrix obtained by evaluating the appropriate covariance function at all eligible pairs of data points. In the following we will use subscripts to indicate which kernel function and which inputs are used to evaluate the kernel matrices, so for example $\mathbf{K}_{\mathbf{f}_s, \mathbf{u}_t}$ denotes the cross covariance matrix between the s^{th} output and t^{th} latent function. Furthermore we will use $\mathbf{K}_{\mathbf{f}_s, \mathbf{u}_{1:T}}$ to mean stack the T $\mathbf{K}_{\mathbf{f}_s, \mathbf{u}_t}$ matrices horizontally and $\mathbf{u}_{1:T}$ to be the result of stacking the $\mathbf{u}_t$ vertically. We will use $\mathbf{K}_{\mathbf{u}_{1:T}}$ to denote the block diagonal matrix where each block is the matrix $\mathbf{K}_{\mathbf{z}_t, \mathbf{z}_t}$; this sub-block shown in the lower right quadrant above. Finally we will use $\mathbf{K}_{\mathbf{f}_{1:S}}$ to denote sub-block in the upper left quadrant above. Following Equation 3.56, we define $\mathbf{Q}_{\mathbf{f}_s, \mathbf{f}_s} = \mathbf{K}_{\mathbf{f}_s, \mathbf{u}_{1:T}} \mathbf{K}_{\mathbf{u}_{1:T}}^{-1} \mathbf{K}_{\mathbf{u}_{1:T}, \mathbf{f}_s}$. Using this notation the mean and covariance of $\mathbf{f}_s$ used in Equation 5.4 is

$$\boldsymbol{\mu}_s = \mathbf{K}_{\mathbf{f}_s, \mathbf{u}_{1:T}} \mathbf{K}_{\mathbf{u}_{1:T}}^{-1} \mathbf{u}_{1:T} \quad (5.10)$$

$$\mathbf{K}_s = \mathbf{K}_{\mathbf{f}_s, \mathbf{f}_s} - \mathbf{Q}_{\mathbf{f}_s, \mathbf{f}_s}. \quad (5.11)$$

Due to the partial independence assumption, the joint likelihood over all S intensity functions is simply the product over each output:

$$p(\mathbf{f}_{1:S} | \mathbf{u}_{1:T}, \mathcal{D}_{1:S}, \mathcal{T}_{1:S}, \mathcal{Z}_{1:T}, \kappa_{1:S, 1:T}, \theta_{1:S}, \phi_{1:T}) = \prod_{s=1}^S \mathcal{N}(\mathbf{f}_s; \boldsymbol{\mu}_s, \mathbf{K}_s). \quad (5.12)$$

Using Bayes' rule for conditional Gaussian distributions, Equations 2.58-2.60, gives the posterior over $\mathbf{u}_{1:T}$:

$$p(\mathbf{u}_{1:T} | \mathbf{f}_{1:S}, \mathcal{Z}_{1:T}, \phi_{1:T}, \kappa_{1:S, 1:T}, \theta_{1:S}) = \mathcal{N}(\mathbf{u}_{1:T}; \boldsymbol{\mu}^+, \boldsymbol{\Sigma}^+), \quad (5.13)$$

where the mean and covariance are

$$\boldsymbol{\Sigma}^+ = \left[\mathbf{K}_{\mathbf{u}_{1:T}}^{-1} + \mathbf{K}_{\mathbf{u}_{1:T}}^{-1} \mathbf{K}_{\mathbf{u}_{1:T}, \mathbf{f}_{1:S}} \mathbf{D}^{-1} \mathbf{K}_{\mathbf{f}_{1:S}, \mathbf{u}_{1:T}} \mathbf{K}_{\mathbf{u}_{1:T}}^{-1} \right]^{-1} \quad (5.14)$$

$$\boldsymbol{\mu}^+ = \boldsymbol{\Sigma}^+ (\mathbf{K}_{\mathbf{f}_{1:S}, \mathbf{u}_{1:T}} \mathbf{K}_{\mathbf{u}_{1:T}}^{-1})^T \mathbf{D}^{-1} \mathbf{f}_{1:S}, \quad (5.15)$$

and where $\mathbf{D} = \mathbf{K}_{\mathbf{f}_{1:S}} - \mathbf{K}_{\mathbf{f}_{1:S}, \mathbf{u}_{1:T}} \mathbf{K}_{\mathbf{u}_{1:T}}^{-1} \mathbf{K}_{\mathbf{f}_{1:S}, \mathbf{u}_{1:T}}^T$ is the following block diagonal

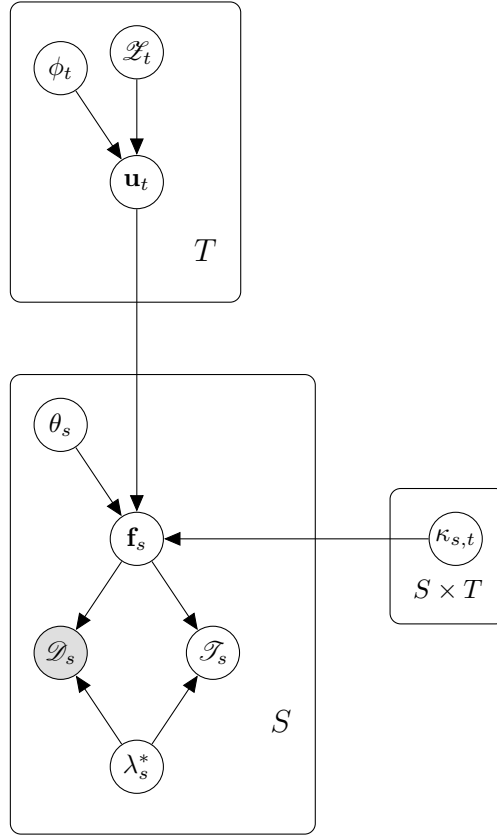


Figure 5.4: Graphical model of the Convolutional Point Process (CPP) model.

matrix

$$\mathbf{D} = \begin{bmatrix} \mathbf{K}_{\mathbf{f}_1, \mathbf{f}_1} - \mathbf{Q}_{\mathbf{f}_1, \mathbf{f}_1} & 0 & \dots & 0 \\ 0 & \mathbf{K}_{\mathbf{f}_2, \mathbf{f}_2} - \mathbf{Q}_{\mathbf{f}_2, \mathbf{f}_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathbf{K}_{\mathbf{f}_S, \mathbf{f}_S} - \mathbf{Q}_{\mathbf{f}_S, \mathbf{f}_S} \end{bmatrix}.$$

As promised, Σ^+ can be computed efficiently using the Woodbury matrix inversion lemma.

5.2.3 Inference algorithm

To perform posterior inference, we will make use of the MCMC algorithms described in Chapter 3. The posterior marginal distributions of most interest are the

output functions $f_{1:S}$ and latent functions $u_{1:T}$. We may also be interested in the distributions of the kernel parameters $\kappa_{s,t}$, θ_s , and ϕ_t , and maximum intensity λ_s^* , while the thinned points are of little interest and are simply marginalised. Since $f_{1:S}$ and $u_{1:T}$ are infinite dimensional objects, we interpolate these functions at fixed (regularly spaced) grid points in order to capture the posterior statistics. The overall procedure to perform inference in the model shown in Figure 5.4 is summarised in Algorithm 2. As indicated we use five different kinds of Markov transitions to draw samples the joint distribution. Using the PIC scheme, the likelihood over

Algorithm 2 Structured Cox process MCMC Scheme

Input: $\mathcal{D}_{1:S}$, hyperparameters.
repeat
 ParFor $s = 1, \dots, S$
 Add/remove J_{new} thinning points (MH): Equations 5.16 to 5.19
 Re-sample locations (MH): Equations 5.20 to 5.21
 Re-sample function (ESS): Equation 5.24
 Re-sample hyperparameters (HMC): Equation 5.24
 Re-sample λ_s^* (Gibbs): Equations 5.22 and 5.23
 EndParFor
 for $t = 1, \dots, T$ **do**
 Re-sample latent function values (Gibbs) $\mathbf{u}_t$: Equation 5.13
 Re-sample latent function hyperparameter ϕ_t (SS): Equation 5.25
 Sample new inducing point locations $\mathcal{Z}_t$ and recompute $\mathbf{u}_t$, Equation 5.26
 end for
until *convergence* is *true*

the point processes factorises conditioned on the latent functions. This means that, given S compute units, the updates associated with each point process may be made in parallel. This is important since the inference algorithm is computationally bottlenecked by operations associated with re-sampling the locations of the thinned points.

5.2.3.1 Re-sampling the thinning points

A number of thinned points may be added or removed and the locations of existing thinned points are re-sampled during each iteration. The insertion and deletion re-sampling steps are chosen using a proposal distribution, $b(N, K) : \mathbb{N} \times \mathbb{N} \rightarrow (0, 1)$,

which is taken to be a Bernoulli distribution with parameter set to $\frac{1}{2}$. If an insertion is required, a new point, $\tilde{\mathbf{x}}_{\text{ins}}$, is drawn uniformly over the domain, $\tilde{\mathbf{x}}_{\text{ins}} \sim \mathcal{U}(\mathcal{X})$ and a function value at that location, $f_s(\tilde{\mathbf{x}}_{\text{ins}})$, is sampled from the relevant GP. The Metropolis-Hastings (MH) acceptance criteria are then used to determine whether this new point should be added to the set or not. The insertion MH proposal and acceptance criteria for the new point are:

$$q_{\text{ins}}(\mathcal{T}_s \rightarrow \mathcal{T}_s \cup \{\tilde{\mathbf{x}}_{\text{ins}}\}) = \frac{b(N_s, J_s)}{|\mathcal{X}|} \mathcal{GP}(f_s(\tilde{\mathbf{x}}_{\text{ins}}) | \mathbf{f}_s, \mathbf{u}_{1:T}, \kappa_{s,1:T}, \theta_s) \quad (5.16)$$

$$a_{\text{ins}}(\mathcal{T}_s \rightarrow \mathcal{T}_s \cup \{\tilde{\mathbf{x}}_{\text{ins}}\}) = \frac{(1 - b(N_s, J_s + 1)) |\mathcal{X}| \lambda_s^*}{(J_s + 1) b(N_s, J_s) (1 + \exp(f_s(\tilde{\mathbf{x}}_{\text{ins}})))}. \quad (5.17)$$

A deletion results in a thinned point, $\tilde{\mathbf{x}}^{(s,j)} \in \mathcal{T}_s$, selected at random, being removed from the set of thinning points. The overall proposal distribution and Metropolis-Hastings acceptance ratios, are:

$$q_{\text{del}}(\mathcal{T}_s \rightarrow \mathcal{T}_s \setminus \tilde{\mathbf{x}}^{(s,j)}) = \frac{1 - b(K, M)}{M} \quad (5.18)$$

$$a_{\text{del}}(\mathcal{T}_s \rightarrow \mathcal{T}_s \setminus \tilde{\mathbf{x}}^{(s,j)}) = \frac{Mb(N_s, J_s - 1)(1 + \exp(f_s(\tilde{\mathbf{x}}^{(s,j)})))}{(1 - b(N_s, J_s)) |\mathcal{X}| \lambda_s^*}. \quad (5.19)$$

During each iteration, thinned points are “jittered”, i.e. moved a small distance from their previous locations. For each thinned point, $\tilde{\mathbf{x}}^{(s,j)}$, a new location, $\hat{\mathbf{x}}^{(s,j)}$, is proposed via a Gaussian proposal density. A random function value $f_s(\hat{\mathbf{x}}^{(s,j)})$ is then sampled conditioned on $\mathbf{f}_s$ with $\tilde{\mathbf{x}}^{(s,j)}$ removed. This gives the move acceptance ratio:

$$a_{\text{move}}(\tilde{\mathbf{x}}^{(s,j)} \rightarrow \hat{\mathbf{x}}^{(s,j)}) = \frac{q_{\text{move}}(\hat{\mathbf{x}}^{(s,j)} \rightarrow \tilde{\mathbf{x}}^{(s,j)})(1 + \exp(f_s(\tilde{\mathbf{x}}^{(s,j)})))}{q_{\text{move}}(\tilde{\mathbf{x}}^{(s,j)} \rightarrow \hat{\mathbf{x}}^{(s,j)})(1 + \exp(f_s(\hat{\mathbf{x}}^{(s,j)})))}, \quad (5.20)$$

where q_{move} is the proposal distribution. If we use a symmetric Gaussian proposal

$$q_{\text{move}}(\tilde{\mathbf{x}}^{(s,j)} \rightarrow \hat{\mathbf{x}}^{(s,j)}) = \mathcal{N}\left(\hat{\mathbf{x}}^{(s,j)}; \tilde{\mathbf{x}}^{(s,j)}, \frac{|\mathcal{X}|}{100}\right), \quad (5.21)$$

the q_{move} terms in Equation 5.20 cancel. With a conjugate Gamma prior, with shape α_s and rate β_s , over the maximum intensity λ_s^* , the posterior distribution, $p(\lambda_s^* | \mathcal{D}_s, \mathcal{T}_s)$, conditioned on the thinned points and observed points, has parameters

$$\alpha_s^+ = \alpha_s + N_s + J_s \quad (5.22)$$

$$\beta_s^+ = \beta_s + |\mathcal{X}|, \quad (5.23)$$

from which we can re-sample λ_s^* in a Gibbs step.

5.2.3.2 Re-sampling the output functions

Since the augmented Cox process likelihood, Equation 5.1, is not conjugate to a GP, the Elliptical Slice Sampling (ESS) algorithm, described in Section 3.1.4, is ideal for the purpose of re-sampling the output functions $\mathbf{f}_s$. The log-likelihood function we require is

$$\begin{aligned} \log p(\mathbf{f}_s | \mathcal{D}_s, \mathcal{T}_s, \theta_s, \kappa_{s,1:T}, \mathbf{u}_{1:T}) = & - \sum_{n=1}^{N_s} \log(1 + \exp(-f_s(\mathbf{x}^{(s,n)}))) \\ & - \sum_{j=1}^{J_s} \log(1 + \exp(f_s(\tilde{\mathbf{x}}^{(s,j)}))) \\ & - \frac{1}{2}(\mathbf{f}_s - \boldsymbol{\mu}_s)^\top \mathbf{K}_s(\mathbf{f}_s - \boldsymbol{\mu}_s) + \text{const}, \end{aligned} \quad (5.24)$$

where $\mathbf{K}_s$ is the covariance in Equation 5.11. To infer the posteriors over the GP hyper-parameters we use the Hamiltonian Monte-Carlo (HMC) algorithm, described in §3.1.2, with log-normal priors over each hyper-parameter.

5.2.3.3 Re-sampling the latent functions

Conditioning on the point process intensity functions, $\mathbf{f}_s$, the latent functions are dependent, with conditional posterior distribution given by Equation 5.13. Having drawn new values for each of the $\mathbf{u}_t$, we can update ϕ_t using Slice Sampling (SS) under the following log conditional posterior:

$$\log p(\phi_t | \mathbf{u}_t, \mathcal{Z}_t) = -\frac{1}{2} \mathbf{u}_t^\top \mathbf{K}_{\mathbf{u}_t, \mathbf{u}_t}^{-1} \mathbf{u}_t - \frac{1}{2} \log |\mathbf{K}_{\mathbf{u}_t, \mathbf{u}_t}| + \text{const}. \quad (5.25)$$

To re-sample the inducing points locations and values, we first sample a new set of inducing point locations, $\mathcal{Z}'_t = \{\mathbf{z}^{(m)}\}_{m=1}^M$, uniformly over their domain: $\mathbf{z}^{(m)} \sim \mathcal{U}(\mathcal{Z})$, and then sample a new set of function values $\mathbf{u}'_t$ from the predictive distribution

$$\mathbf{u}'_t \sim \mathcal{N}(\mathbf{K}_{\mathbf{u}'_t, \mathbf{u}_t} \mathbf{K}_{\mathbf{u}_t, \mathbf{u}_t}^{-1} \mathbf{u}_t, \mathbf{K}_{\mathbf{u}'_t, \mathbf{u}'_t} - \mathbf{K}_{\mathbf{u}'_t, \mathbf{u}_t} \mathbf{K}_{\mathbf{u}_t, \mathbf{u}_t}^{-1} \mathbf{K}_{\mathbf{u}_t, \mathbf{u}'_t}), \quad (5.26)$$

where $\mathbf{K}_{\mathbf{u}_t, \mathbf{u}'_t}$ has entries

$$[\mathbf{K}_{\mathbf{u}_t, \mathbf{u}'_t}]_{m, m'} = K_t(\mathbf{z}^{(m)}, \mathbf{z}^{(m')}). \quad (5.27)$$

It is important to re-sample the inducing point locations as doing so significantly improves mixing.

5.2.4 Adaptive thinning

The advantage of the thinning method described in this chapter, is that it allows the Cox-process likelihood to be written in closed-form using only finite dimensional distributions. The disadvantage of the method is the cost of marginalising these points and this problem is particularly acute when Gaussian processes are used to drive the intensity functions, due the computational scaling of GP inference. This problem becomes even worse when the data is concentrated into small, high density subdomains or in higher dimensional spaces. Under the current methodology, we must add sufficient thinning points to allow the concentration of data and thinning points to match that of the most dense subregion of the data.

In this section, we will develop a novel solution to this problem called adaptive thinning². This approach, illustrated in Figure 5.5, will model the upper bounded intensity over the space using a piece-wise constant function, where the level of each piece is some fraction of the global upper bound, λ^* . The transitions of this piece-wise constant function are determined by current state of the inferred intensity function and are set to ensure the former always upper bounds the latter. Thinning points are then added until the concentration of points matches the piecewise constant level, not the global upper bound. In doing so, we preserve the tractability of the integrals in the likelihood and allow empty regions to be uniformised with far fewer points without violating any of the properties of the point process.

The piece-wise constant function takes values from a set of L possible maximum rate ‘levels’:

$$\mathcal{L} = \{l_i \in (0, 1] | l_i < l_{i+1}, l_L = 1\}_{i=1}^L. \quad (5.28)$$

We will assume the rate levels are fixed a-priori, however a dynamic system can be envisaged. Next, we augment the variable set to include, for each thinning

²This is not the same as the method described by Favaro and Teh (2013)

point $\tilde{\mathbf{x}}^{(s,j)}$, a variable, $\tilde{r}^{(s,j)} \in \{1 \dots L\}$, that indicates which rate level is currently assigned to that point. $\tilde{r}^{(s,j)}$ is such that $\sigma(f_s(\tilde{\mathbf{x}}^{(s,j)})) \leq l_{\tilde{r}^{(s,j)}}$.

The probability of accepting a thinning point, $\tilde{\mathbf{x}}^{(s,j)}$, between an upper bound with value $l_{\tilde{r}^{(s,j)}}$, and the sigmoid GP is

$$p(\tilde{\mathbf{x}}^{(s,j)} | \tilde{r}^{(s,j)}) = \frac{l_{\tilde{r}^{(s,j)}} - \sigma(f_s(\tilde{\mathbf{x}}^{(s,j)}))}{l_{\tilde{r}^{(s,j)}}}. \quad (5.29)$$

Note that since the likelihood of the thinning point is now $p(\tilde{\mathbf{x}}^{(s,j)} | \tilde{r}^{(s,j)}) \cdot \lambda_s^* l_{\tilde{r}^{(s,j)}} \leq \lambda_s^* \sigma(-f_s(\tilde{\mathbf{x}}^{(s,j)}))$, we will use fewer thinning points as desired. The likelihood function for $\mathbf{f}_s$, Equation 5.24, is modified using the new likelihood to become

$$\begin{aligned} \log p(\mathbf{f}_s | \mathcal{D}_s, \mathcal{T}_s, \theta_s, \kappa_{s,1:T}, \mathbf{u}_{1:T}) = & - \sum_{n=1}^{N_s} \log(1 + \exp(-f_s(\mathbf{x}^{(s,n)}))) \\ & - \sum_{j=1}^{J_s} \log(1 + \exp(-f_s(\tilde{\mathbf{x}}^{(s,j)}))) \\ & + \sum_{j=1}^{J_s} \log(l_{\tilde{r}^{(s,j)}} - 1 + l_{\tilde{r}^{(s,j)}} \exp(-f_s(\tilde{\mathbf{x}}^{(s,j)}))) \\ & - \frac{1}{2}(\mathbf{f}_s - \boldsymbol{\mu}_s)^\top \mathbf{K}_s(\mathbf{f}_s - \boldsymbol{\mu}_s) + \text{const}, \end{aligned} \quad (5.30)$$

while the MH acceptance criteria, modified to use the adaptive rate levels, now becomes the following:

$$a_{\text{ins}}(\mathcal{T}_s \rightarrow \mathcal{T}_s \cup \tilde{\mathbf{x}}_{\text{ins}}) = \frac{(1 - b(N_s, J_s + 1)) |\mathcal{X}| \lambda_s^* l_{\tilde{r}_{\text{ins}}} p(\tilde{\mathbf{x}}_{\text{ins}} | \tilde{r}_{\text{ins}})}{(J_s + 1) b(N_s, J_s)} \quad (5.31)$$

$$a_{\text{del}}(\mathcal{T}_s \rightarrow \mathcal{T}_s \setminus \tilde{\mathbf{x}}^{(s,j)}) = \frac{J_s b(N_s, J_s - 1)}{(1 - b(N_s, J_s)) |\mathcal{X}| \lambda_s^* l_{\tilde{r}^{(s,j)}} p(\tilde{\mathbf{x}}^{(s,j)} | \tilde{r}^{(s,j)})}. \quad (5.32)$$

The rate levels change at each iteration during the ‘move’ step when we compute the new rate level for jittered points and so the algorithm continually adapts to the current value of the inferred intensity function. We can compose the acceptance criteria for jittered points as the product of the insertion and deletion criteria

$$a_{\text{move}} = a_{\text{ins}} \cdot a_{\text{del}}.$$

A potential drawback of this approach is that when the function values $\mathbf{f}_s$ are re-sampled using ESS, the function will be constrained to lie under the upper bound values at each point. This would inhibit the function from moving freely up and

down where the function is close to one of the upper bounds and so slow mixing of the Markov-chain. A simple solution is to ensure that there is always some slack, Δ_l , between the function evaluations and the assigned upper bound. The slack creates space for the function to move into, albeit at the cost of increasing the number of thinning points required.

The slack is incorporated by assigning the rate as follows:

$$\tilde{r}^{(s,j)} \leftarrow \begin{cases} \min \left(\operatorname{argmax}_{r \in 1, \dots, L} \left(\mathbb{1}\{\Delta_l \cdot \sigma(f_s(\tilde{\mathbf{x}}^{(s,j)}))\} < l_r\} \right) \right), & \text{if } \sigma(f_s(\tilde{\mathbf{x}}^{(s,j)})) \leq \Delta_l \\ L, & \text{otherwise.} \end{cases} \quad (5.33)$$

With a slack value of $\Delta_l = 0.9$, it was found that mixing was hardly affected.

When re-sampling λ_s^* we must compute an estimate of the total number of points under a single rate uniformisation of the space. This estimate is readily available because the number of points (including observed data points) with rate r is a Monte-Carlo integral of the proportion of space thinned to rate level l_r . Since a point in a region where the intensity is scaled by l_i in effect represents $\frac{1}{l_i}$ points with no scaling, Equation 5.22 becomes

$$\alpha_s^+ = \alpha_s + \hat{N}_s + \hat{J}_s = \alpha_s + \sum_{n=1}^{N_s} \frac{1}{l_{r^{(s,n)}}} + \sum_{j=1}^{J_s} \frac{1}{l_{\tilde{r}^{(s,j)}}}, \quad (5.34)$$

where $r^{(s,n)}$ is the notional rate of observed data computed exactly as for the thinning points.

5.2.4.1 Empirical evaluation of adaptive thinning

We will now see the results of two experiments, in one and two dimensions respectively, that were used to validate adaptive thinning. These experiments used the original Sigmoid-Gaussian Cox Process (SGCP) model modified in the manner described above. In both cases 10 known random intensity functions were used to generate event data, generating 15 random point process datasets per function in the one dimensional case and 10 in the two dimensional case. One dataset was used to learn the model and the remainder were held out for testing

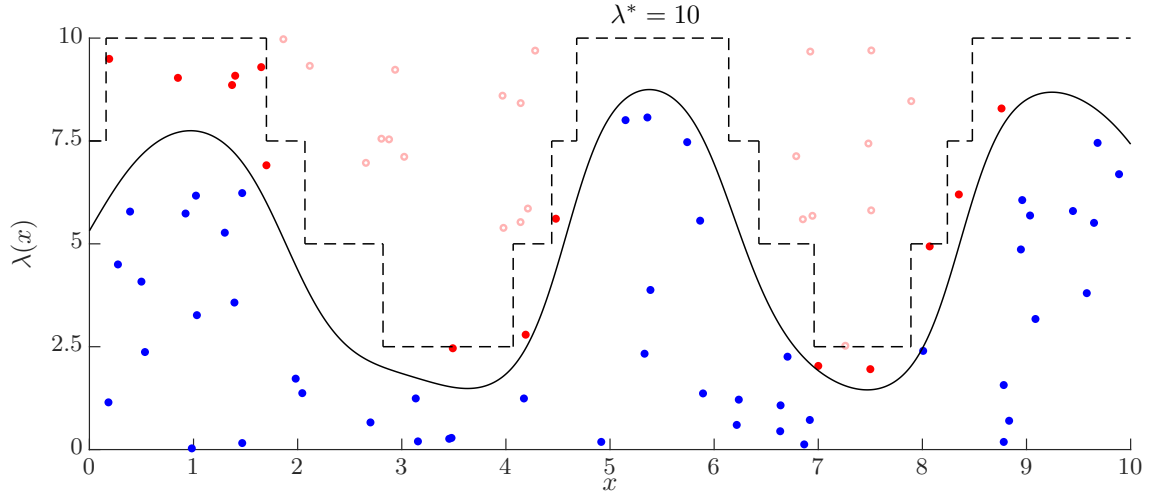


Figure 5.5: Illustration of adaptive thinning showing the data points in blue and the thinning points in red. The adaptive rate level is shown by the dashed line. Due to the slack, the adaptive rate level sits above the intensity function, but does not touch it. Only those points below the adaptive rate level must be modelled by the Gaussian process and those above it, shown by the pink o's, are no longer required.

purposes. Two measures of performance were used: the Root Mean Squared (RMS) error compared against the generating intensity function, and average predictive log-likelihood across all held out test datasets.

In one-dimension, each model was run for 6 minutes total compute time; in two dimensions, 20 minutes in total was allowed. In both cases half the time is allocated to burn-in. In one dimension, the single rate method achieved roughly one sample per second; with two rate levels, just under four samples per second was achieved and with four levels, just under 8 samples per second was achieved. Since mixing was hardly affected, this resulted in a roughly commensurate increase in the number of effective samples per second. In two dimensions the number of points required was larger in all cases, with the multi-rate approach yielding a factor of two speed-up.

The results given in Tables 5.1 and 5.2 show that in almost all cases the multi-level approaches perform better across both metrics than the original, single rate, approach.

	Predictive Log-likelihood			RMS Error		
	1 Rate	2 Rates	4 Rates	1 Rate	2 Rates	4 Rates
1	373.8	381.8	388.2	11.6	13.0	7.1
2	626.1	644.2	650.7	14.6	10.5	7.2
3	274.2	285.1	288.0	10.6	5.0	4.9
4	435.5	457.7	456.0	10.5	4.7	5.1
5	877.0	885.8	889.5	12.4	10.1	10.4
6	995.4	1006.6	1013.4	11.1	8.2	7.6
7	753.0	763.3	760.6	12.0	13.7	12.5
8	522.3	531.2	528.3	13.0	12.0	12.8
9	1840.6	1852.9	1826.0	19.6	16.4	28.8
10	2328.1	2365.8	2349.8	31.4	27.2	32.6

Table 5.1: Adaptive thinning results in one dimension.

	Predictive Log-likelihood			RMS Error		
	1 Rate	2 Rates	4 Rates	1 Rate	2 Rates	4 Rates
1	2039.6	2080.6	2203.0	13.3	13.4	11.3
2	2757.9	2758.1	2829.3	14.3	14.3	14.7
3	2753.2	2689.5	2827.2	13.5	14.7	13.5
4	2803.2	2784.0	2933.6	12.5	12.9	12.0
5	2532.4	2663.0	2572.1	17.9	16.6	17.8
6	3098.2	3040.5	3054.4	14.4	16.2	15.3
7	3157.5	3259.8	3075.2	15.1	13.7	17.5
8	2086.9	2101.0	2087.4	14.6	14.4	14.8
9	5018.1	5146.6	5185.3	18.3	17.1	15.6
10	2008.1	2205.0	2174.0	15.4	12.9	13.6

Table 5.2: Adaptive thinning results in two dimensions.

5.2.5 Empirical evaluation of CPP

We will now see an empirical evaluation of the CPP using synthetic data and two real datasets. The real data sets chosen were thought to exhibit a dependency structure that could be captured by a convolution process. The two real datasets are British Member of Parliament (MP) tweet times and NBA player shot profiles. The twitter data was obtained using the Twitter API during the period Monday 2nd to Sunday 8th December 2013, the week of Nelson Mandela's death. The basketball data for the 2013-2014 season was scraped from the NBA website. The CPP was compared to independent SGCP's (Adams et al. 2009), as well the Kernel Smoothing

(KS) method of Botev et al. (2010).

5.2.5.1 Synthetic data

A single latent function and four intensity functions were sampled using the convolution process; these are shown in Figure 5.6. Event data was then sampled from the latent functions using thinning. The CPP model was run for over 2000 iterations after it was determined that convergence had been achieved. Adaptive thinning with three rate levels, $\frac{1}{4}\lambda^*$, $\frac{1}{2}\lambda^*$ and λ^* , was used. The learned intensity functions are shown in Figure 5.7.

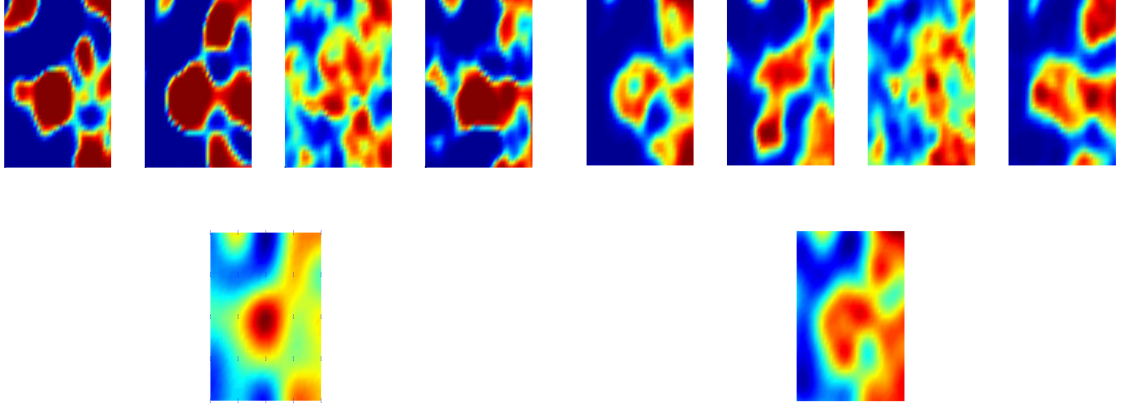


Figure 5.6: Four synthetically generated 2D intensity functions, $\sigma(f_1), \dots, \sigma(f_4)$ (above), and the single latent function, u_1 from which they were generated (below). The variety of intensity functions a single latent function can yield is evident.

Figure 5.7: Learned intensity functions corresponding to those in Figure 5.6 learned using CPP. Considering only four observed event processes were used, the latent function has been recovered well.

5.2.5.2 Twitter data

Four MPs active on twitter were selected at random; they are anonymised with the letters A, B, C, and D. Each dataset was randomly partitioned into 75% training data, with the remainder being used to evaluate predictive test log-likelihood. Figure 5.8 shows the average learned intensity functions for each MP (red line), along with the one standard deviation bars (grey shading) derived from the function samples. The bottom plot shows the learned latent driving function in the same manner.

The latent function clearly shows a strong daily period, particularly evident during the working week. Furthermore, the largest two peaks in activity occur on the 3rd and the 5th of December. Potential contributing factors to these two spikes include a public sector strike, and the death of Nelson Mandela respectively.

Table 5.3 gives predictive log-likelihood for the held out data for the CPP algorithm and the benchmarks. Both the SGCP and the CPP approach use two maximum rate levels for adaptive thinning. Compared to the independent modelling of the intensity function of the SGCP and KS benchmarks, the structured model of the CPP performs vastly better, suggesting that it is indeed appropriate for this type of data.

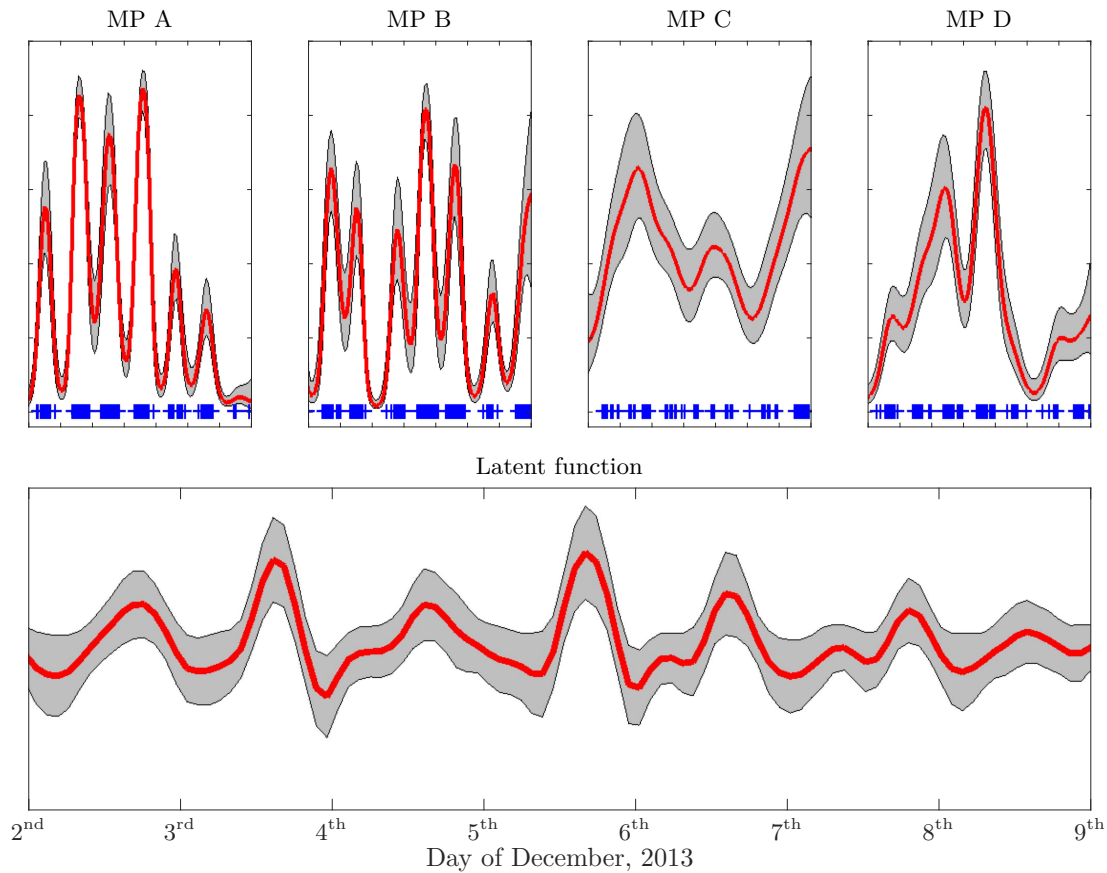


Figure 5.8: The learned intensities for MP tweet data and below, the learned latent function. The observed data is shown by the blue crosses.

MP	Predictive Log-Likelihood		
	KS	SGCP	CPP
A	177.1	176.6	469.5
B	-1.3	89.6	412.9
C	-5.4	49.6	283.4
D	-67.7	38.7	293.7

Table 5.3: Predictive log-likelihood for held out twitter data across models.

Player	Predictive Log-Likelihood		
	KS	SGCP	CPP
Blake Griffin	-121.8	335.6	374.7
Damien Lillard	-22.6	231.2	395.3
DeMer DeRozen	-2.9	253.1	410.7
Arron Affalo	-260.7	-76.7	84.2

Table 5.4: Basketball data held-out predictive log-likelihood.

5.2.5.3 Basketball data

For the basketball shot data, the approach performed particularly well. A diverse subset of four players—Blake Griffin, Damien Lillard, DeMar DeRozen, and Arron Affalo—with diverse playing styles was selected. Each player had around 600 attempted shots, of which 400 were used, holding out the rest as test data. We used three rate boundaries: $\frac{1}{4}\lambda^*$, $\frac{1}{2}\lambda^*$, λ^* , and once again averaged over 2000 samples after convergence.

Figure 5.9 depicts the resulting intensity functions and Table 5.4, gives predictive log-likelihoods on held out data for the CPP and the benchmarks. It is clear that the structured approach to modelling the basketball shot data in a hierarchical Bayesian fashion yields an improvement over both the independent SGCP as well the kernel density estimator. The adaptive thinning method was crucial for this dataset due to the high data density around the hoop.

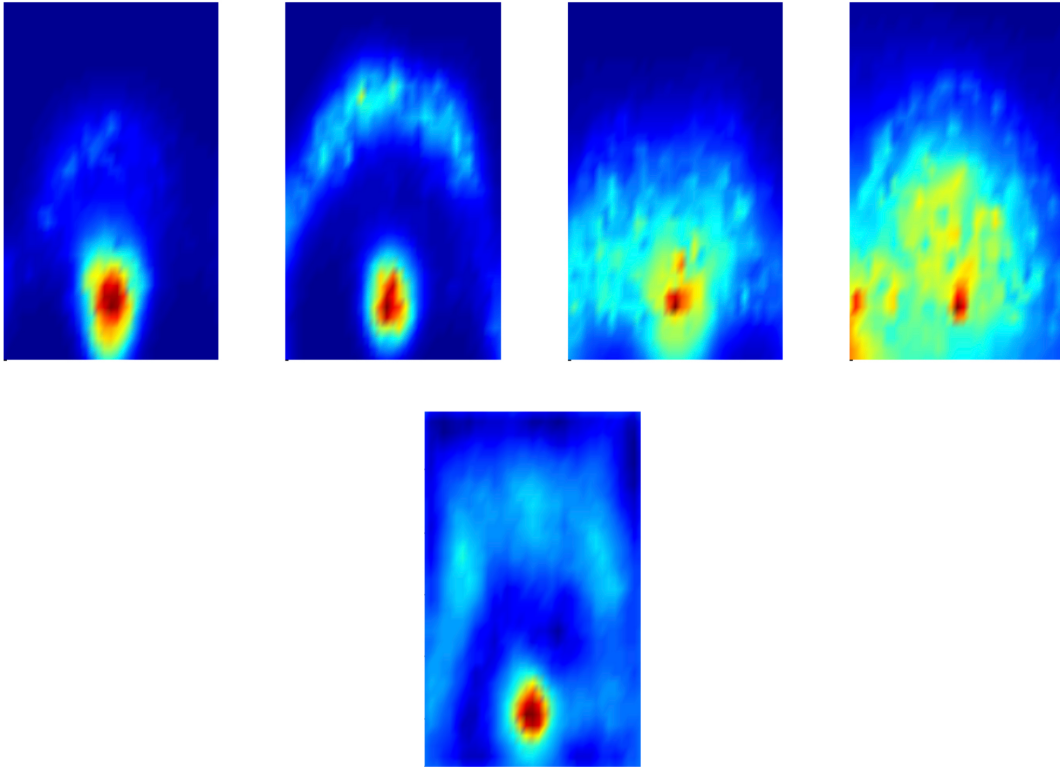


Figure 5.9: Basketball shot intensity functions learned using the CPP model. Above, intensity functions for four individual players, left to right: Blake Griffin, Damien Lillard, DeMer DeRozen and Arron Affalo; and below, a single latent function. It is clear that the latent function represents the common structure of the court hotspots—the locations of hoop and three pointer line are clearly visible—while the intensity functions for the individual players match the expected playing styles. Arron Affalo, for example, is a ‘shooting guard’ who spends the majority of his time inside the three pointer line, but has a propensity to shoot from the bottom left of the court; these effects are clearly visible on the heat map.

5.3 Directions for further development

We have introduced a fully generative model for dependent point processes based on convolution processes. The inference scheme builds on sparse Gaussian processes and adaptive-thinning, which both improve the computational efficiency of the algorithm. In order to guide further work, several aspects of this work that are worth considering in more detail.

- The relationship between inducing points and thinned points: The number of inducing points used determines the effective communication bandwidth

between the output functions. The number of inducing points used determines the amount of information the latent function can hold. However the number of thinning points required may be far larger than the number of inducing points required. Thinning therefore uses a large amount of computational effort to compress the information in the data into the inducing points. It seems more sensible therefore to use the inducing points to control the intensity function directly.

- Specification of prior hyper-parameters: Specifying the prior distributions on kernel input and output length scales from a subjective Bayesian perspective is quite difficult and an empirical Bayes' approach may be justified.
- MCMC inference: Although the inference scheme presented in the chapter goes some way to reduce the computational demands of Bayesian Cox process inference with a GP prior, it is still computationally demanding.
- Sparse latent functions: While convolution-processes allow complex interactions between latent and output functions, the sigmoid transformation occurs *after* the output functions are combined and this ordering does not yield parsimonious positive combinations of positive functions as, for example, Non-negative Matrix Factorisation (NMF) does.

In the next two chapters we will develop alternative approaches that address each of these points. While these new approaches are more efficient for Poisson processes, the work convolution process prior is still useful for constructing correlated renewal processes using the method described by Rao and Teh (2011).

6

Variational inference for Gaussian process modulated Poisson processes

In the previous chapter we went some way toward improving the tractability of the Sigmoid-Gaussian Cox Process (SGCP) by introducing a set of piecewise constant rate levels that mitigated the requirement to add sufficient thinning points to uniformise the entire space to the level of the most dense sub-region, allowing the model to scale better in higher dimensions while maintaining the desirable closed-form properties of the likelihood.

In this chapter we will develop a new model to improve Bayesian inference of Cox processes still further by making use of the sparse variational Gaussian Process (GP) inference framework introduced in §3.3.2. As we shall see, this will allow us to integrate the rate function over the domain in closed-form and therefore remove the need for computationally expensive thinning points.

This model will form the foundation for new inference algorithms for complex event data presented in later chapters.

6.1 Model

Once again we construct the prior over the intensity function using a GP. Rather than using the sigmoid link function, we will use the square-transform link function, $\lambda(\mathbf{x}) = f^2(\mathbf{x})$, $\mathbf{x} \in \mathcal{X}$ (where f is a GP distributed random function), examples of which were shown in Figure 2.13. As in the previous chapter, we will use a set of inducing variables, $\mathbf{u}$, located at the inducing points, $\mathcal{Z}$. In contrast to the convolution process model, inducing variables will be part of the same GP as f and not another latent function u and so $f|\mathbf{u} \sim \mathcal{GP}(\mu(\mathbf{x}), \Sigma(\mathbf{x}, \mathbf{x}'))$ has mean and covariance functions.

$$\mu(\mathbf{x}) = \mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \mathbf{u}, \quad (6.1)$$

$$\Sigma(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') - \mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \mathbf{k}_{zx'}, \quad (6.2)$$

where $\mathbf{u}$ is distributed $\mathcal{N}(\mathbf{u}; \vec{1}\bar{u}, \mathbf{K}_{zz})$. With this hierarchical formulation the joint distribution over $\mathcal{D}$, f , $\mathbf{u}$ and Ω is

$$p(\mathcal{D}, f, \mathbf{u}, \Omega) = p(\mathcal{D}|\lambda = f^2)p(f|\mathbf{u}, \Omega)p(\mathbf{u}|\Omega)p(\Omega), \quad (6.3)$$

where $p(\mathcal{D}|\lambda = f^2)$ is the Cox process likelihood, Equation 2.83, and $p(\Omega)$ is an optional prior on the set of kernel hyper-parameters and $\bar{u}$. For notational convenience we will often omit conditioning on Ω .

6.2 Inference

In contrast to the standard variational Gaussian process model described in Chapter 3, to bound the model evidence, $p(\mathcal{D})$, we must marginalise an infinite dimensional function, f , rather than a finite dimensional vector, $\mathbf{f}$, as was previously the case, due to the integral over the region $\mathcal{X}$ embedded in the likelihood, Equation 2.83. This raises some measure theoretic issues that we will now address. We will then go on to compute the terms in the Evidence Lower Bound (ELBO).

6.2.1 Measure theoretic aspects

In this chapter we have already used the notation $p(f|\cdot)$. From our definition of probability density, Equation 2.1, this implies that $p(f|\cdot)$ is continuous with respect to a Lebesgue measure and that we can write $p(f|\cdot) \, \mathrm{d}f$. However, since f is infinite dimensional, $\mathrm{d}f$ implies integration with respect to the infinite dimensional Lebesgue measure that—for all practical purposes—does not exist, creating some potential technical problems. However, a rigorous proof that the sparse variational framework can be extended to infinite dimensional likelihoods was provided by Matthews et al. (2016). Their approach, which we will now summarise, begins using the measure theoretic definition of Bayes’ rule which states that ratio of the data likelihood and the model evidence is the *Radon-Nikodym derivative* of the posterior measure, $\hat{P}$, and with respect to the prior measure, P :

$$\frac{\mathrm{d}\hat{P}}{\mathrm{d}P}(f) = \frac{p(\mathcal{D}|f)}{p(\mathcal{D})}. \quad (6.4)$$

In our case, P and $\hat{P}$ are Gaussian process measures on f . We now expand the measure theoretic definition of the Kullback–Leibler (KL) divergence between a sparse GP measure Q and $\hat{P}$:

$$\mathrm{KL}(Q||\hat{P}) = \int_{\mathbb{R}^\infty} \left[\log \frac{\mathrm{d}Q}{\mathrm{d}\hat{P}}(f) \right] \mathrm{d}Q(f) \quad (6.5)$$

$$= \int_{\mathbb{R}^\infty} \left[\log \frac{\mathrm{d}Q}{\mathrm{d}\hat{P}} \frac{\mathrm{d}P}{\mathrm{d}P}(f) \right] \mathrm{d}Q(f) \quad (6.6)$$

$$= \int_{\mathbb{R}^\infty} \left[\frac{\mathrm{d}Q}{\mathrm{d}P}(f) \right] \mathrm{d}Q(f) - \int_{\mathbb{R}^\infty} \left[\frac{\mathrm{d}\hat{P}}{\mathrm{d}P}(f) \right] \mathrm{d}Q(f) \quad (6.7)$$

$$= \int_{\mathbb{R}^{|\mathcal{Z}|}} \left[\log \frac{\mathrm{d}Q_{\mathcal{Z}}}{\mathrm{d}P_{\mathcal{Z}}}(f_{\mathcal{Z}}) \right] \mathrm{d}Q(f_{\mathcal{Z}}) - \int_{\mathbb{R}^{\mathcal{X}}} \left[\log \frac{p(\mathcal{D}|f_{\mathcal{X}})}{p(\mathcal{D})} \right] \mathrm{d}Q(f_{\mathcal{X}}) \quad (6.8)$$

$$= \mathrm{KL}(Q_{\mathcal{Z}}||P_{\mathcal{Z}}) - \mathbb{E}_{Q_{\mathcal{X}}}[\log p(\mathcal{D}|f_{\mathcal{X}})] + \log p(\mathcal{D}). \quad (6.9)$$

Between Equation 6.7 and Equation 6.8 we make use of Q being a measure on a sparse Gaussian process and the likelihood only depending on the region $\mathcal{X}$. Since $|\mathcal{Z}|$ is finite, the measure theoretic KL-divergence $\mathrm{KL}(Q_{\mathcal{Z}}||P_{\mathcal{Z}})$ reduces to $\mathrm{KL}(q(\mathbf{u})||p(\mathbf{u}))$, the standard definition of KL-divergence for Gaussian probability densities given by Equation 2.72 and that also appears in $\mathcal{L}_{\mathrm{GP}}$, Equation 3.64.

After rearranging and dropping $\text{KL}(Q||\hat{P})$ we arrive at the ELBO:

$$\log p(\mathcal{D}) \geq \mathbb{E}_{Q_{\mathcal{X}}} [\log p(\mathcal{D}|f_{\mathcal{X}})] - \text{KL}(q(\mathbf{u})||p(\mathbf{u})) \triangleq \mathcal{L}_{\text{PP}} \quad (6.10)$$

However $\mathbb{R}^{\mathcal{X}}$ is still an infinite dimensional space, so we use Tonelli's Theorem to reverse the ordering of the integrations over the positive integrand $f^2(\mathbf{x})$:

$$\mathbb{E}_{Q_{\mathcal{X}}} [\log p(\mathcal{D}|f)] = \mathbb{E}_{Q_{\mathcal{X}}} \left[- \int_{\mathcal{X}} f^2(\mathbf{x}) d\mathbf{x} + \sum_{n=1}^N \log f^2(\mathbf{x}^{(n)}) \right] \quad (6.11)$$

$$= - \int_{\mathcal{X}} \mathbb{E}_{Q_x} [f^2(\mathbf{x})] d\mathbf{x} + \sum_{n=1}^N \mathbb{E}_{Q_{x^{(n)}}} [\log f^2(\mathbf{x}^{(n)})]. \quad (6.12)$$

We are now only required to evaluate expectations points-wise and so we can write

$$\mathbb{E}_{Q_x} [f^2(\mathbf{x})] = \int_{-\infty}^{\infty} f^2(\mathbf{x}) dQ(f(\mathbf{x})) \quad (6.13)$$

$$= \int_{-\infty}^{\infty} f^2(\mathbf{x}) q(f(\mathbf{x})) df(\mathbf{x}). \quad (6.14)$$

where $q(f(\mathbf{x}))$ is a univariate normal density that we can write down in closed-form since $p(f(\mathbf{x})|\mathbf{u})$ is conjugate to the variational distribution $q(\mathbf{u}) = \mathcal{N}(\mathbf{u}; \mathbf{m}, \mathbf{S})$ and therefore:

$$q(f(\mathbf{x})) = \int p(f(\mathbf{x})|\mathbf{u}) q(\mathbf{u}) d\mathbf{u} \quad (6.15)$$

$$= \mathcal{N}(f(\mathbf{x}); \tilde{\mu}(\mathbf{x}), \tilde{\Sigma}(\mathbf{x}, \mathbf{x})) \quad (6.16)$$

$$\tilde{\mu}(\mathbf{x}) = \mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \mathbf{m} \quad (6.17)$$

$$\tilde{\Sigma}(\mathbf{x}, \mathbf{x}) = K(\mathbf{x}, \mathbf{x}) - \mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \mathbf{k}_{zx} + \mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \mathbf{S} \mathbf{K}_{zz}^{-1} \mathbf{k}_{zx}, \quad (6.18)$$

While derivation of Matthews et al. never requires the use of $\int(\cdot)df$, it does lead to precisely the same ELBO previously derived in Lloyd et al. (2015b), which used Jensen's inequality as follows:

$$\log p(\mathcal{D}) = \log \left[\iint p(\mathcal{D}|f) p(f|\mathbf{u}) p(\mathbf{u}) \frac{q(\mathbf{u})}{q(\mathbf{u})} d\mathbf{u} df \right] \quad (6.19)$$

$$\geq \iint p(f|\mathbf{u}) q(\mathbf{u}) d\mathbf{u} \log[p(\mathcal{D}|f)] df \quad (6.20)$$

$$+ \iint p(f|\mathbf{u}) q(\mathbf{u}) df \log \left[\frac{p(\mathbf{u})}{q(\mathbf{u})} \right] d\mathbf{u} \\ = \int q(f) \log[p(\mathcal{D}|f)] df + \int q(\mathbf{u}) \log \left[\frac{p(\mathbf{u})}{q(\mathbf{u})} \right] d\mathbf{u} \quad (6.21)$$

$$= \mathbb{E}_{q(f)} [\log p(\mathcal{D}|f)] - \text{KL}(q(\mathbf{u})||p(\mathbf{u})) \quad (6.22)$$

$$\triangleq \mathcal{L}_{\text{PP}},$$

where $p(f|\mathbf{u})$ is the “density” of the conditional GP (for which conjugacy to $q(\mathbf{u})$ is assumed), leading to $q(f)$, the “density” of the infinite dimensional variational distribution:

$$q(f) = \int p(f|\mathbf{u})q(\mathbf{u})d\mathbf{u} \quad (6.23)$$

$$= \mathcal{GP}(f; \tilde{\mu}(\mathbf{x}), \tilde{\Sigma}(\mathbf{x}, \mathbf{x}')), \quad (6.24)$$

$$\tilde{\mu}(\mathbf{x}) = \mathbf{k}_{xz}\mathbf{K}_{zz}^{-1}\mathbf{m}, \quad (6.25)$$

$$\tilde{\Sigma}(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') - \mathbf{k}_{xz}\mathbf{K}_{zz}^{-1}\mathbf{k}_{zx'} + \mathbf{k}_{xz}\mathbf{K}_{zz}^{-1}\mathbf{S}\mathbf{K}_{zz}^{-1}\mathbf{k}_{zx'}. \quad (6.26)$$

Since these two derivations give the same result, for the remainder of this thesis we will continue to use the—arguably clearer, but perhaps less technically correct—notation of Lloyd et al. and allow, for example, integration w.r.t. df . This is justified since we will never actually take expectations with respect to either $dQ(f)$ or $q(f)df$; both cases are merely place holders waiting for us to be able to take point-wise expectations.

We now have two tasks remaining: we must compute the integral over the region $\mathcal{X}$ and calculate the expectations at the data points.

6.2.2 Integrating over the domain

Our lower bound has the desirable property that we can compute in closed-form the expectation of the intensity, $f^2(\mathbf{x})$, at any point, $\mathbf{x}$, in the domain. We will now see the advantage our chosen function transformation, i.e. $\lambda(\mathbf{x}) = f^2(\mathbf{x})$, provides for the required integration over the domain.

To keep the notation concise we have introduced the following identities:

$$\begin{aligned} f_x &\triangleq f(\mathbf{x}), & \tilde{\mu}_x &\triangleq \tilde{\mu}(\mathbf{x}), & \tilde{\sigma}_x^2 &\triangleq \tilde{\Sigma}(\mathbf{x}, \mathbf{x}), \\ f_n &\triangleq f(\mathbf{x}^{(n)}), & \tilde{\mu}_n &\triangleq \tilde{\mu}(\mathbf{x}^{(n)}), & \tilde{\sigma}_n^2 &\triangleq \tilde{\Sigma}(\mathbf{x}^{(n)}, \mathbf{x}^{(n)}). \end{aligned}$$

Firstly, we can use the definition of variance, Equation 2.13, to expand $\mathbb{E}_{q(f)}[f_x^2]$ giving:

$$\mathbb{E}_{q(f)}[\log p(\mathcal{D}|f)] = - \int_{\mathcal{X}} \mathbb{E}_{q(f)}[f_x^2] d\mathbf{x} + \sum_{n=1}^N \log f_n \quad (6.27)$$

$$= - \int_{\mathcal{X}} \left\{ \mathbb{E}_{q(f)}[f_x]^2 + \text{Var}_{q(f)}[f_x] \right\} d\mathbf{x} + \sum_{n=1}^N \mathbb{E}_{q(f)} \left[\log f^2(\mathbf{x}^{(n)}) \right], \quad (6.28)$$

where the required statistics for Equation 6.28 are:

$$\mathbb{E}_{q(f)}[f_x]^2 = \tilde{\mu}_x^2 = \mathbf{m}^\top \mathbf{K}_{zz}^{-1} \mathbf{k}_{zx} \mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \mathbf{m}, \quad (6.29)$$

$$\text{Var}_{q(f)}[f_x] = \tilde{\sigma}_x^2 = K(\mathbf{x}, \mathbf{x}) - \text{tr}(\mathbf{K}_{zz}^{-1} \mathbf{k}_{zx} \mathbf{k}_{xz}) + \text{tr}(\mathbf{K}_{zz}^{-1} \mathbf{S} \mathbf{K}_{zz}^{-1} \mathbf{k}_{zx} \mathbf{k}_{xz}). \quad (6.30)$$

Secondly, it is now easy to calculate the integrals since only $\mathbf{k}_{zx} = \mathbf{k}_{xz}^\top$ is a function of $\mathbf{x}$ and $K(\mathbf{x}, \mathbf{x}) = \gamma$:

$$\int_{\mathcal{X}} \mathbb{E}_{q(f)}[f_x]^2 d\mathbf{x} = \mathbf{m}^\top \mathbf{K}_{zz}^{-1} \boldsymbol{\Psi}_{zz} \mathbf{K}_{zz}^{-1} \mathbf{m}, \quad (6.31)$$

$$\int_{\mathcal{X}} \text{Var}_{q(f)}[f_x] d\mathbf{x} = \gamma |\mathcal{X}| - \text{tr}(\mathbf{K}_{zz}^{-1} \boldsymbol{\Psi}_{zz}) + \text{tr}(\mathbf{K}_{zz}^{-1} \mathbf{S} \mathbf{K}_{zz}^{-1} \boldsymbol{\Psi}_{zz}). \quad (6.32)$$

The matrix $[\boldsymbol{\Psi}_{zz}]_{m,m'} = \Psi(\mathbf{z}^{(m)}, \mathbf{z}^{(m')})$ is constructed via the integral

$$\Psi(\mathbf{z}, \mathbf{z}') = \int_{\mathcal{X}} K(\mathbf{z}, \mathbf{x}) K(\mathbf{x}, \mathbf{z}') d\mathbf{x}. \quad (6.33)$$

For the exponentiated quadratic ARD kernel Ψ can be found by rearranging the product as a single exponentiated quadratic in $\mathbf{x}$ and $\bar{\mathbf{z}}$ as follows:

$$\Psi(\mathbf{z}, \mathbf{z}') = \int_{\mathcal{X}} \gamma^2 \prod_{r=1}^R \exp \left(-\frac{(z_r - z'_r)^2}{4\alpha_r} - \frac{(x_r - \bar{z}_r)^2}{\alpha_r} \right) d\mathbf{x} \quad (6.34)$$

$$= \gamma^2 \prod_{r=1}^R \frac{\sqrt{\pi\alpha_r}}{2} \exp \left(-\frac{(z_r - z'_r)^2}{4\alpha_r} \right) \times \left[\text{erf} \left(\frac{\bar{z}_r - \mathcal{X}_r^{\max}}{\sqrt{\alpha_r}} \right) - \text{erf} \left(\frac{\bar{z}_r - \mathcal{X}_r^{\min}}{\sqrt{\alpha_r}} \right) \right], \quad (6.35)$$

and where $\bar{\mathbf{z}} = [\bar{z}_1, \dots, \bar{z}_R]^\top$ has elements $\bar{z}_r = \frac{z_r + z'_r}{2}$.

Note that we have obtained the function Ψ by integrating $K(\mathbf{z}, \mathbf{x})K(\mathbf{x}, \mathbf{z}')$ in precisely the same way the as the $\Psi_{zz}^{2(n)}$ statistic of variational Gaussian Process Latent Variable Model (GPLVM) described in Chapter 4. The only difference in this

case being that we have integrated with respect to a uniform measure over the finite region $\mathcal{X}$ and not with respect to a Gaussian measure over the whole real line.

In addition to the exponentiated quadratic ARD kernel, the function Ψ can be computed in closed-form for other kernels, including polynomial and periodic kernels, and the Spectral Mixture Kernel (SMK) (see §8.2.2).

6.2.3 Expectations at the data points

The expectation $\mathbb{E}_{q(f)}[\log f_n^2]$ has an analytic solution that we express as¹:

$$\mathbb{E}_{q(f)}[\log f_n^2] = \int_{-\infty}^{\infty} \log(f_n^2) \mathcal{N}(f_n, \tilde{\mu}_n, \tilde{\sigma}_n^2) df_n \quad (6.36)$$

$$= -\tilde{G}\left(-\frac{\tilde{\mu}_n^2}{2\tilde{\sigma}_n^2}\right) + \log\left(\frac{\tilde{\sigma}_n^2}{2}\right) - C, \quad (6.37)$$

where $C \approx 0.5772156649$ is the Euler–Mascheroni constant and $\tilde{G}$ is defined via the confluent hyper-geometric function

$${}_1F_1(a, b, z) = \sum_{k=0}^{\infty} \frac{(a)_k z^k}{(b)_k k!}, \quad (6.38)$$

where $(\cdot)_k$ denotes the rising Pochhammer series

$$(a)_0 = 1, \quad (a)_k = a(a+1)(a+2) \dots (a+k-1). \quad (6.39)$$

Specifically $\tilde{G}$ is a specialised version of the partial derivative of ${}_1F_1$ with respect to its first argument and can be computed using the method of Ancarani and Gasaneo (2008), which has a particular solution at $a = 0$, leading to the following definition of $\tilde{G}$:

$$\tilde{G}(z) = {}_1F_1^{(1,0,0)}\left(0, \frac{1}{2}, z\right) = 2z \sum_{j=0}^{\infty} \frac{j! z^j}{(2)_j (1\frac{1}{2})_j}. \quad (6.40)$$

¹This expectation can be computed using the Mathematica symbolic mathematics package using the command: `Integrate[Log[f^2] 1/(Sqrt[2 Pi] s) Exp[-(f-m)^2/(2 s^2)], {f, -Infinity, Infinity}]`

Naive implementation of Equation 6.40 has poor numerical stability, although this can be improved slightly using an iterative scheme such as:

$$S_0 = A_0 = 1 \quad (6.41)$$

$$A_{j+1} = A_j \times \left[\left(\frac{1+j}{1\frac{1}{2}+j} \right) \left(\frac{z}{2+j} \right) \right] \quad (6.42)$$

$$S_{j+1} = S_j + A_{j+1} \quad (6.43)$$

$$\tilde{G}(z) \approx 2zS_j \quad \text{for large enough } j. \quad (6.44)$$

In practice however we use a large multi-resolution look-up table of precomputed values obtained using the Mathematica software package. As shown in Figure 6.1, this function decreases very slowly as its argument becomes increasingly negative, so we can easily compute accurate evaluations of $\tilde{G}(z)$ for any z by linear interpolation of the lookup table and, as a by-product, we also obtain its derivative, $\tilde{G}'$. Importantly,

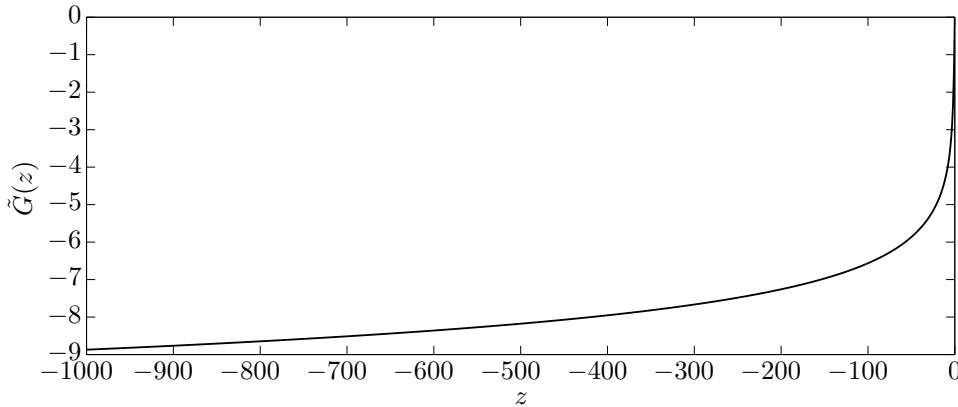


Figure 6.1: $\tilde{G}$ as a function of its argument. Accurate evaluations of $\tilde{G}$ can be obtained from a precomputed look-up table.

irrespective of the number of dimensions over which the intensity function was defined, the required integral is always of a one-dimensional function parametrised by mean and variance of the GP. Our formulation has thus allowed a potentially expensive integral over a R -dimensional latent intensity function to be computed in closed-form, in exchange for N numerical integrals. However, since the required numerical integrals are one-dimensional—and, as described, can be pre-computed—this is a happy trade-off.

6.2.4 Optimising the bound

To perform inference we find the variational parameters $\mathbf{m}^*$, $\mathbf{S}^*$ and the model hyperparameters Ω^* that maximise $\mathcal{L}_{\text{pp}}$. We could compute the Maximum A' Posterior (MAP) estimate by maximising $\mathcal{L}(\mathcal{D}; \mathbf{y}) + \log p(\Omega)$, however in the experiments presented in this chapter, the maximum-likelihood solution was found. Unlike the variational GP regression case, described in Section 3.3.2, we cannot set the derivatives equal to zero and rearrange to get closed-form solutions for the optimal variational parameters, $\mathbf{m}^*$ and $\mathbf{S}^*$, since $\tilde{\mu}_n$ and $\tilde{\sigma}_n^2$ are also functions of $\mathbf{m}$ and $\mathbf{S}$ and these terms appear in the arguments of the function $\tilde{G}'$, and so we must resort to numerical optimisation. In the experiments presented in §6.5 the L-BFGS algorithm (Liu and Nocedal 1989; Byrd et al. 1995) was used. All the parameters are optimised simultaneously by constructing an augmented vector $\mathbf{y} = [\Omega^\top, \mathbf{m}^\top, \text{vech}(\mathbf{L})^\top]^\top$, where $\text{vech}(\mathbf{L})$ is the vectorisation of the lower triangular elements of the lower triangular matrix $\mathbf{L}$, and where $\mathbf{S} = \mathbf{L}\mathbf{L}^\top$. The required gradients are given in Appendix A.

To optimise the inducing point locations we use the re-parametrisation

$$z_r^{(m)} = \frac{\mathcal{X}_r^{\min} + \mathcal{X}_r^{\max}}{2} - \frac{\mathcal{X}_r^{\min} - \mathcal{X}_r^{\max}}{2} \sin(a_r^{(m)}), \quad (6.45)$$

and optimise in $a_r^{(m)} \in [-\pi, \pi]$, which ensures the inducing points always remain within the region $\mathcal{X}$.

6.2.5 Locating the inducing points

One final part of the inference algorithm we have so far left unspecified is the number and location of inducing points. In principle, for a given set of parameters Ω , we will obtain a lower bound for the true GP likelihood for any number of inducing points in any configuration of locations. We will consider two possible approaches: firstly, treating the inducing points as optimisation parameters and, secondly, fixing them on a regular grid.

If the locations of the inducing points are optimised, this suggests—in common with other sparse GP models—that we can achieve good performance using a small number of well-placed inducing points. This is achieved at the cost of an additional set of optimisation parameters, although the size of $\mathbf{m}$ and $\mathbf{S}$ are commensurately reduced. Optimisation of the inducing points is particularly computationally expensive, however, because of the necessity to recompute $\mathbf{K}_{zz}^{-1}$ for each dimension of each inducing point ($M \times R$ in total) and since $\mathbf{K}_{zz}$ affects every term in the bound.

Regular grids, on the other hand, also have several advantages. Consider firstly that—in contrast to standard sparse GP regression—the accuracy of our solution is not only governed by the distance between the inducing points and the data points. The variance of $f(\mathbf{x})$ increases as $\mathbf{x}$ becomes further from the inducing points. However, the rate function, λ , is a function of both the mean *and* the variance of f . Since we are integrating λ over $\mathcal{X}$, we need inducing points distributed across $\mathcal{X}$ and not just in regions close to the data. An evenly spaced grid is one way to ensure this is achieved.

Regularly sampled grids also afford potential computational advantages. When the grid points are evenly spaced, the kernel matrix has Toeplitz structure, and hence allows matrix inversion (and linear solving) in $\mathcal{O}(M \log^2 M)$ time, a fact previously utilised for efficient point processes by Cunningham et al. (2008).

If the inducing point locations are restricted to the intersections of a regular or irregular grid, we can also take advantage of Kronecker structure. In Chapter 7 we will see how Kronecker structure can reduce the cost of matrix manipulations when the kernel function is separable across dimensions (as required by Equation 2.78). In the experiments described in this Chapter however, naïve inversion of the inducing point kernel matrix, $\mathbf{K}_{zz}$, was used resulting in computational complexity of $\mathcal{O}(NM^3)$. Hence the computation times reported below could potentially be improved upon.

6.3 Variational bounds using other link functions

We now consider what advantages the square-transform offers compared with the more commonly used exponential and or sigmoid link functions in the context of variational inference. Firstly, the exponential link function to construct the classic Log-Gaussian Cox Process (LCGP) $\lambda(\mathbf{x}) = \exp(f(\mathbf{x}))$. This transformation is undesirable for two reasons. The more obvious of these is that after taking expectations under $q(f)$ we are left with the integral

$$- \int_{\mathcal{X}} \exp\left(\tilde{\mu}_x + \frac{\tilde{\sigma}_x^2}{2}\right) d\mathbf{x}, \quad (6.46)$$

which cannot be computed in closed-form. We could approximate the integral using a series expansion, however this would be difficult with more than a couple of terms and furthermore, since the function is concave, this approximation would not be a lower bound.

The second—and more subtle—reason is that in using this transformation, when we take expectations under $q(f)$ of the data, we obtain

$$\mathbb{E}_{q(f)} [\log \{\exp(f_n)\}] = \tilde{\mu}_n. \quad (6.47)$$

Since the mean, $\tilde{\mu}_n$, is not a function of $\mathbf{S}$, the variance of the variational distribution $q(\mathbf{u})$, we have effectively decoupled the data from the uncertainty on our variational approximation; this is clearly undesirable.

The sigmoid function is inappropriate since we cannot take expectations of the resulting likelihood under $q(f)$. We can make some progress if we choose another squashing function, namely the probit likelihood,

$$\Phi(x) = \int_{-\infty}^x \frac{1}{2\pi} \exp(-t^2/2) dt, \quad (6.48)$$

giving rate function $\lambda(\mathbf{x}) = \Phi(f(\mathbf{x}))$. This can be integrated analytically against the GP prior, however we are again left with a difficult integral over $\mathcal{X}$ which is

$$- \int_{\mathcal{X}} \Phi\left(\frac{\tilde{\mu}_x}{\sqrt{1 + \tilde{\sigma}_x^2}}\right) d\mathbf{x}. \quad (6.49)$$

As the range of this transformation is $(0, 1)$ we would also require additional machinery to infer a scaling variable.

In contrast, the square transform allows the integral over the region $\mathcal{X}$ to be computed in closed-form and $\mathbb{E}_{q(f)}[\log f_n^2]$ can be computed quickly and accurately. Importantly this transformation also maintains the connection between the data and the variational uncertainty. Although the square transform is not a one-to-one function—any rate function λ may have been generated by f^2 or $(-f)^2$ —this sign ambiguity is integrated out in a Bayesian sense in Equation 6.36.

6.4 Predictive distribution

To form the predictive distribution we assume our optimised variational distribution $q^*(\mathbf{u}) = \mathcal{N}(\mathbf{u}; \mathbf{m}^*, \mathbf{S}^*)$ approximates the posterior $p(\mathbf{u}|\mathcal{D})$. Analogously to Equation 6.24, we next compute $q^*(f) \approx p(f|\mathcal{D})$. We can now derive a lower bound of the predictive log-likelihood on a held-out test dataset $\mathcal{H} = \{\tilde{\mathbf{x}}^{(k)}\}_{k=1}^K$:

$$\log p(\mathcal{H}|\mathcal{D}, \Omega^*) = \log \mathbb{E}_{p(f|\mathcal{D})}[p(\mathcal{H}|f)] \quad (6.50)$$

$$\approx \log \mathbb{E}_{q^*(f)}[p(\mathcal{H}|f)] \triangleq \mathcal{M}_p \quad (6.51)$$

$$\geq \mathbb{E}_{q^*(f)}[\log p(\mathcal{H}|f)] \triangleq \mathcal{L}_p. \quad (6.52)$$

The derivation of $\mathcal{L}_p$ follows Equations 6.28-6.37. The resulting bound is the same as $\mathcal{L}$ except $\mathbf{m}$ and $\mathbf{S}$ are replaced with $\mathbf{m}^*$ and $\mathbf{S}^*$, and there is no KL-divergence term. Kernel matrices are computed using Ω^* .

The tightness of the prediction distribution, i.e. the difference between $\mathcal{M}_p$ and $\mathcal{L}_p$, is related to the uncertainty associated with function f . From an information theoretic perspective, we can say that the tightness of the bound will be a function of the entropy of f : $\mathbb{H}[f] = \frac{1}{2} \log |\tilde{\Sigma}| + \text{const.}$ The variance of f , at any point, is a function of the variance of the inducing variables, $\mathbf{S}$, and the input length scales $\alpha_{1:R}$. The points of most interest are those at which training or test data points are located, where the tightness of the bound—or lack there-of—will lead to a certain

amount of slack for each data point. Figure 6.2 shows how this slack varies as a function of $\tilde{\mu}_n$ and $\tilde{\sigma}_n$. When the standard deviation of f is very small relative to the mean value, the bound is tight. As we increase the standard deviation, the lower bound actually decreases even as the upper bound increases until it is roughly equal to the mean, after which both the upper and lower bounds increase at the same rate approximately 1.28 nats apart.

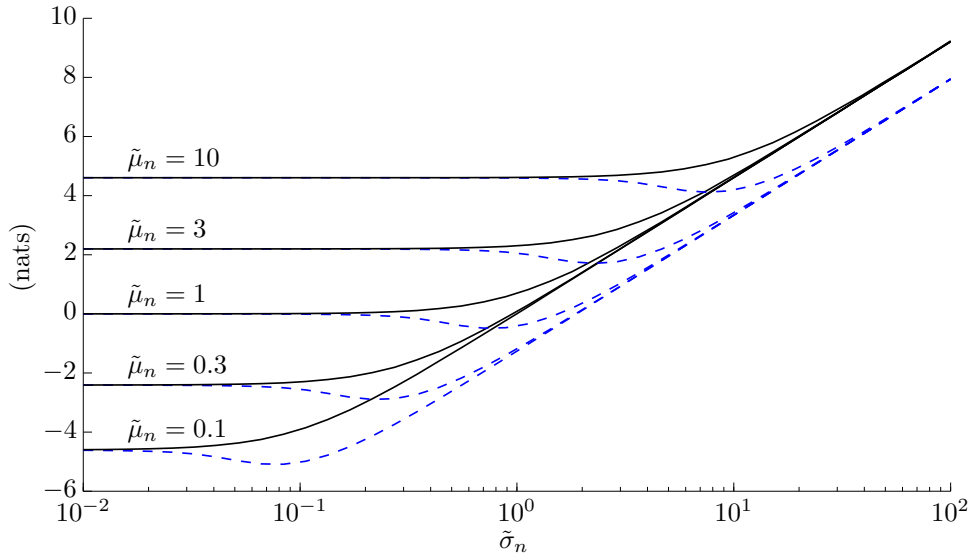


Figure 6.2: The tightness of the inequality $\log \mathbb{E}_{q(f)}[f_n^2] \geq \mathbb{E}_{q(f)}[\log f_n^2]$ as a function of $\tilde{\mu}_n$ and $\tilde{\sigma}_n$. The lower bounds, shown by blue dotted lines, are calculated using Equation 6.37.

Given this knowledge we can define a new tightened lower bound, $\mathcal{L}_0$, in which we allow the variance of function values at the inducing variables to collapse to zero by setting $\mathbf{S} := 0$ and therefore in effect $p(\mathbf{u}|\mathcal{D}) := \delta(\mathbf{u} - \mathbf{m})$. This reduces the conditional entropy of f given $\mathbf{u}$ by shrinking the final term in the definition of $\tilde{\Sigma}$ (Equation 6.24), resulting in a tighter bound, albeit for a more restricted model. Since $\mathbf{S}$ is otherwise a positive definite matrix, setting $\mathbf{S} := 0$ also reduces the integral term over the domain and this also increases the likelihood.

As it is a more restricted model, we expect the ground truth $\mathcal{M}_0 = \log \left(\mathbb{E}_{q(f|\mathbf{u}=\mathbf{m})} [p(\mathcal{H}|f)] \right)$ to be lower than the ground truth $\mathcal{M}_p$. In practice, however, because the variational $\mathcal{L}_0$ is so much tighter, it can be useful to use this bound when comparing approximate predictive likelihood with other (non-variational) approaches. This is demonstrated

empirically in Figures 6.5 and 6.7. The true predictive bounds on the held-out 50% of the data were evaluated using 10,000 Markov-Chain Monte-Carlo (MCMC) samples for a range of inducing point grid densities, both with and without optimisation of the inducing point locations. In the figures, the shaded area highlights the gap between Monte-Carlo and variational approximations as function of the number of inducing points. It should be noted that for the relatively smooth coal mining data, Figure 6.5, all bounds do not benefit from more than about 10 inducing points, while in the case of the twitter data, Figure 6.7, the faster dynamics call for increased numbers. In all of the Figures, the tightness of $\mathcal{L}_0$ compared to $\mathcal{L}_p$ is apparent.

6.5 Experiments

To evaluate the Variational Bayesian Poisson Processes (VBPP) algorithm it was benchmarked against a frequentist Kernel Smoothing (KS) approach, described below, both with and without end correction (KS+EC and KS−EC respectively), and the fully Bayesian SGCP sampler used in Chapter 5 to evaluate adaptive-thinning. The test data sets were generative data using the SGCP model and several real world data sets.

6.5.1 Benchmarks

The benchmark kernel smoothing method is similar to standard kernel density estimator, with the exception that truncated normal kernels are used to account for explicit knowledge of the domain of the intensity function—the latter is referred to as “end-correction” in some literature (Diggle 1985). The kernel smoother optimises a diagonal covariance, Σ^* , by maximising the leave-one-out training objective

$$\Sigma^* = \operatorname{argmax}_{\Sigma} \sum_{i=1}^N \log \sum_{j \neq i=1}^N \mathcal{N}_{\mathcal{X}}(\mathbf{x}^{(i)}; \mathbf{x}^{(j)}, \Sigma). \quad (6.53)$$

We can construct the predictive distribution by combining the maximum-likelihood estimates of the size and spatial location of the data. For a test data set $\mathcal{H}$ (with $K!$ permutations) this distribution is

$$p(\mathcal{H}|\mathcal{D}) = K! p(K|\mathcal{D}) \prod_{k=1}^K p(\tilde{\mathbf{x}}^{(k)}|\mathcal{D}), \quad (6.54)$$

where the location density,

$$p(\tilde{\mathbf{x}}^{(k)}|\mathcal{D}) = \frac{1}{N} \sum_{n=1}^N \mathcal{N}_{\mathcal{X}}(\tilde{\mathbf{x}}^{(k)}; \mathbf{x}^{(n)}, \Sigma^*), \quad (6.55)$$

is computed using the previously described method and the distribution of the number of points,

$$p(K|\mathcal{D}) = \frac{N^K}{K!} \exp(-N), \quad (6.56)$$

is simply a Poisson distribution with parameter N . It is straight forward to show that Equation 6.54 is equivalent to Equation 2.83 since we can interpret the rate function as $\lambda(\mathbf{x}) = \sum_{n=1}^N \mathcal{N}_{\mathcal{X}}(\mathbf{x}; \mathbf{x}^{(n)}, \Sigma^*)$ and since $\int_{\mathcal{X}} \lambda(\mathbf{x}) d\mathbf{x} = N$.

6.5.2 Synthetic Data

Synthetic LCGP data was created by first generating a log-Gaussian process on a high resolution one or two-dimensional grid and then, conditioned on the same intensity function, drawing a training dataset and multiple test datasets. Average and worst case predictive log-likelihood results for these test datasets are given in Tables 6.1 6.3 and Root Mean Squared (RMS) errors in Tables 6.2 and 6.4.

It can be seen in the 1D case that while the difference between the VBPP intensities $\mathcal{L}_p$ and $\mathcal{L}_0$ is small, the predictive log-likelihood and RMS error metrics in Tables 6.1 & 6.4 are significantly better in the $\mathcal{L}_0$ case.

Figure 6.3 visualises a two-dimensional inferred intensity conditioned on ~ 500 observations. Although the SGCP sampler gives better predictive performance than the VBPP $\mathcal{L}_0$ bound, it should be noted that the sampler uses well tuned hyper-parameters, uses the same link function as the generative process and is much more

computationally expensive. VBPP outperforms kernel smoothing in terms of both predictive likelihood and RMS error.

In general it is clear that the variational method performs approximately equivalently to the SGCP in the majority of cases, and typically surpasses the results delivered by both KS methodologies. VBPP arrives at a full posterior over the function in a fraction of the time the SGCP does, albeit typically slightly slower than the maximum-likelihood solution delivered by KS.

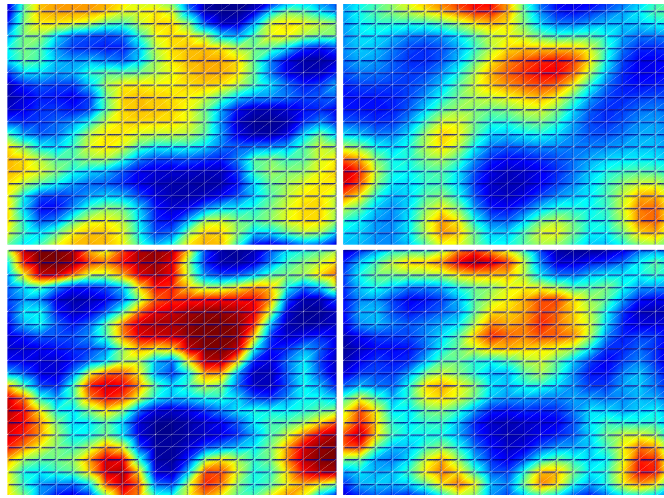


Figure 6.3: 2D Synthetic data intensities obtained from three methods (clockwise from top right), VBPP, KS+EC, SGCP. Ground truth is shown in the top left.

6.5.3 Coal mining disaster data

The first real dataset we shall investigate comprises 190 events recorded from 15th March 1851 to 22nd March 1962; each represents a coal-mining disaster that killed at least ten people in the United Kingdom. These events are indicated by the rug plot along the axis of Figure 6.4.

Since it was first analysed in this form by Jarrett (1979), this dataset has often been used to validate Cox processes models, for example Adams et al. (2009), as the rate of such disasters is expected to vary according to known historical developments (Carlin et al. 1992; Fearnhead 2006). The figure depicts a decline in the rate of such disasters throughout 1870–1890, a period that saw the British Parliament pass

	SGCP		KS-EC		KS+EC		VBPP ($\mathcal{L}_p$)		VBPP ($\mathcal{L}_0$)	
	Avg.	Min	Avg.	Min	Avg.	Min	Avg.	Min	Avg.	Min
1	209.7	148.8	202.3	147.4	207.3	151.8	199.4	144.3	207.1	151.7
2	68.3	41.0	61.1	32.8	65.4	36.4	52.0	23.0	64.9	35.8
3	4.2	-4.4	0.2	-7.9	3.5	-4.7	-0.2	-8.9	5.1	-3.7
4	-3.0	-8.7	-7.1	-13.5	-3.0	-9.9	-4.2	-9.6	-3.6	-9.0

Table 6.1: Test log-likelihood for 1D synthetic data.

	SGCP		KS-EC		KS+EC		VBPP ($\mathcal{L}_p$)		VBPP ($\mathcal{L}_0$)	
	RMS	RT(s)	RMS	RT(s)	RMS	RT(s)	RMS	RT(s)	RMS	RT(s)
1	2.6	124.7	3.9	0.022	3.3	0.016	3.2	0.7	3.2	0.7
2	1.0	120.5	3.1	0.003	2.7	0.005	2.9	0.6	2.8	0.6
3	1.3	112.2	1.4	0.002	1.5	0.003	1.2	0.6	1.1	0.6
4	0.7	108.1	1.1	0.003	0.9	0.004	0.8	0.6	0.8	0.6

Table 6.2: RMS error and wall-clock run time for 1D synthetic data.

	SGCP		KS-EC		KS+EC		VBPP ($\mathcal{L}_p$)		VBPP ($\mathcal{L}_0$)	
	Avg.	Min	Avg.	Min	Avg.	Min	Avg.	Min	Avg.	Min
1	446.1	398.1	333.8	292.8	389.8	347.5	355.9	310.4	392.9	347.2
2	-61.1	-73.6	-109.0	-123.3	-78.3	-91.7	-91.0	-102.1	-76.1	-87.3
3	122.4	73.7	43.6	-3.2	84.3	33.8	64.2	15.5	92.6	43.4
4	175.8	137.6	111.9	81.4	147.0	111.9	93.6	61.7	148.3	115.2
5	446.1	365.0	369.5	305.6	413.6	345.1	354.0	285.7	415.5	346.9

Table 6.3: Test log-likelihood for 2D synthetic data.

	SGCP		KS-EC		KS+EC		VBPP ($\mathcal{L}_p$)		VBPP ($\mathcal{L}_0$)	
	RMS	RT(s)	RMS	RT(s)	RMS	RT(s)	RMS	RT(s)	RMS	RT(s)
1	1.37	7547.83	2.01	0.10	1.48	0.34	1.21	3.26	1.21	3.26
2	0.38	1039.65	0.66	0.01	0.46	0.02	0.36	2.00	0.38	2.00
3	0.88	3173.91	1.30	0.02	1.04	0.12	0.81	2.44	0.81	2.44
4	1.71	3773.75	1.49	0.02	1.26	0.05	1.11	2.58	1.14	2.58
5	2.94	6368.44	2.29	0.05	2.02	0.21	1.82	2.83	1.81	2.83

Table 6.4: RMS error and wall-clock run time for 2D synthetic data.

several acts with the aim of improving safety for mine workers, including the Coal Mines Regulation Acts of 1872 and 1887. The inferred intensity also declines after 1950, likely related to the imposition of further safety regulation with the Mines and Quarries Act, 1954.

For real datasets, training and test subsets were created by randomly splitting the dataset in half. Since the ground truth rates for real datasets are unknown, held-out predictive likelihood were the only performance metric used, values for which are shown in Figure 6.5. The results are encouraging: VBPP outperformed kernel smoothing and SGCP with as few as 10 inducing points; more inducing points yielded no further benefit.

6.5.4 Twitter Data

In the next experiment, the tweet profile of Alistair Darling, chairman of the “Better Together Campaign”, one week either side of the Scottish independence election was modelled. Half the data, 189 tweets, was held out and a regular 31 point grid was used. The result is shown in Figure 6.6.

Figure 6.7 compares the performance of regularly spaced and optimised inducing points. The figure shows optimisation yields considerably improved performance on this dataset. It also shows that the $\mathcal{L}_0$ and $\mathcal{L}_p$ bounds become less tight as the number of inducing points increases, suggesting there is progressively less uncertainty represented in the variational parameter $\mathbf{S}$ and more uncertainty captured by a reduced kernel length scale. This transition is observed for fewer inducing points when inducing point optimisation is employed.

With or without inducing point optimisation, the Monte-Carlo sampled VBPP likelihoods $\mathcal{M}_0$ and $\mathcal{M}_p$ outperform both the SGCP and kernel smoothing by a wide margin, suggesting the square link function is an appropriate model for this data.

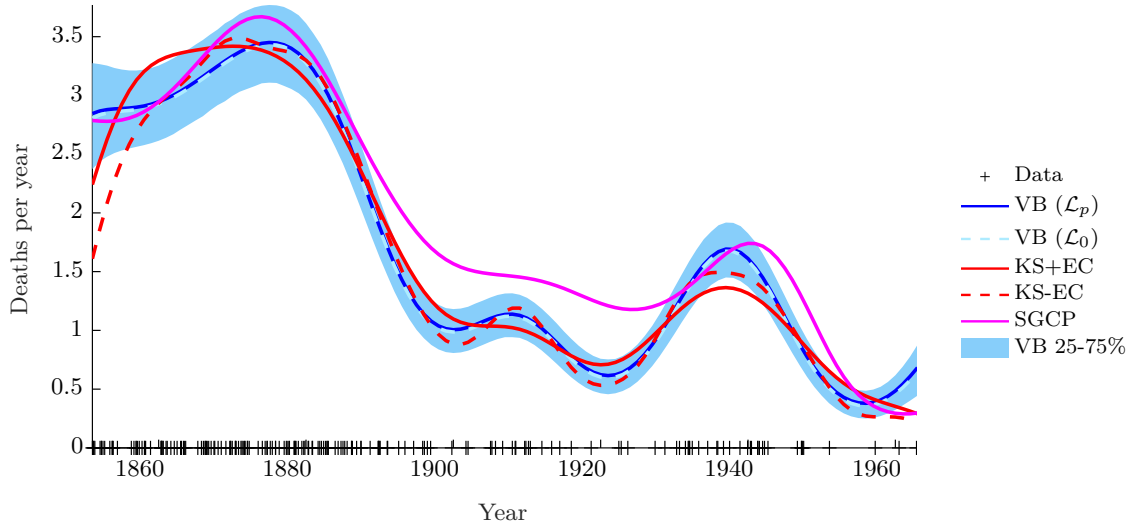


Figure 6.4: The inferred intensity functions for the coal mining disaster dataset. The inferred intensity of disasters correlates well with the historical introduction of safety regulation.

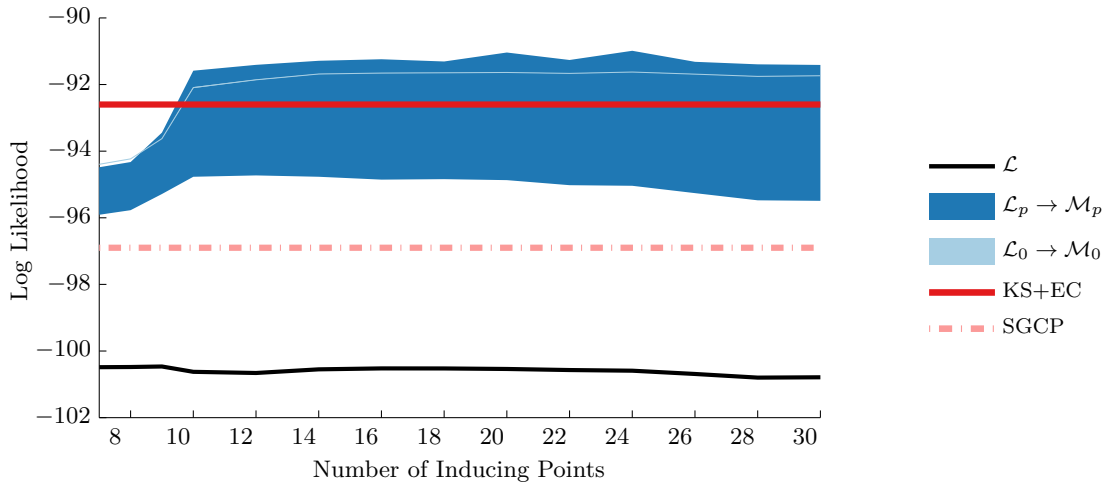


Figure 6.5: The predictive log-likelihood for the coal mining disaster dataset as a function of the number of inducing points. The difference between sampling $\mathcal{M}_0$ and $\mathcal{M}_p$, and the corresponding lower bounds, $\mathcal{L}_0$ and $\mathcal{L}_p$ (see Section 6.4) clearly demonstrates the tightness of the $\mathcal{L}_0$ bound.

6.5.5 Malaria Data

Cox process models have been used to map geographical patterns of diseases, such as malaria (*Malaria Atlas Project* 2017). There are significant potential build upon the Cox process model presented in this chapter, to develop powerful epidemiological tools allowing, for example, the joint modelling of disease incidence with correlating factors—those studying the spread of malaria often wish to use continuous rainfall

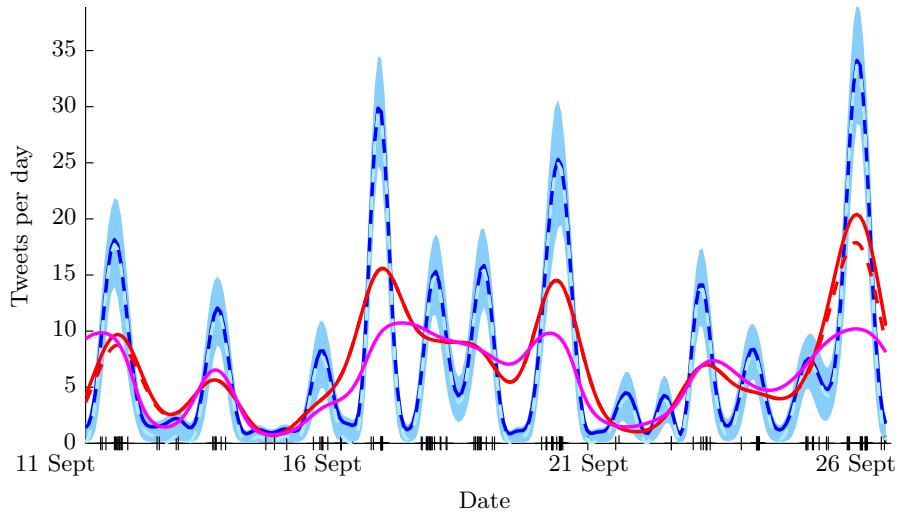


Figure 6.6: The inferred intensity functions for the Twitter dataset. (Legend as for Figure 6.4)

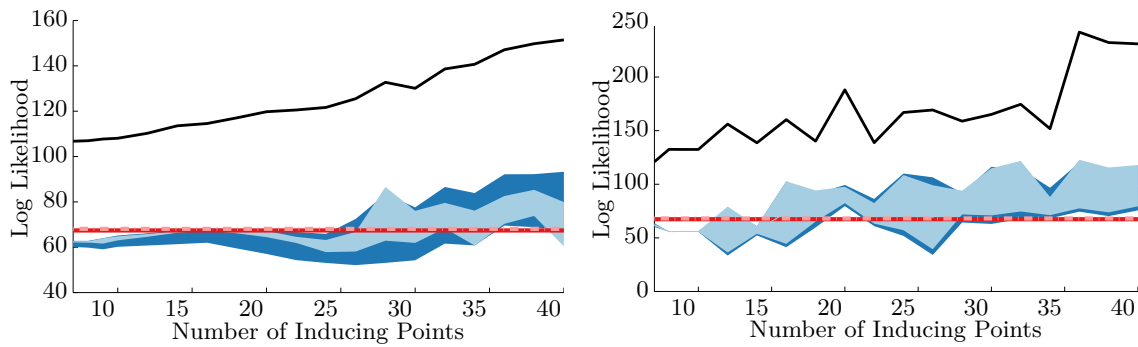


Figure 6.7: The predictive log-likelihood for the Twitter dataset as a function of the number of inducing points. Left: Inducing points located on a fixed, regular grid. Right: Optimised inducing point locations. (Legend as for Figure 6.5.)

measurements to better inform their models—in a scalable framework Bayesian framework. Figure 6.8 show the result of applying VBPP to 741 incidences of malaria outbreak documented in Kenya between 1985 and 2010. Test log-likelihood results, given in Table 6.5, show VBPP performs comparably to kernel smoothing.

KS-EC	KS+EC	VBPP ($\mathcal{M}_p$)	VBPP ($\mathcal{L}_0$)
855.0	867.2	869.7	855.9

Table 6.5: The predictive log-likelihood for the malaria dataset.

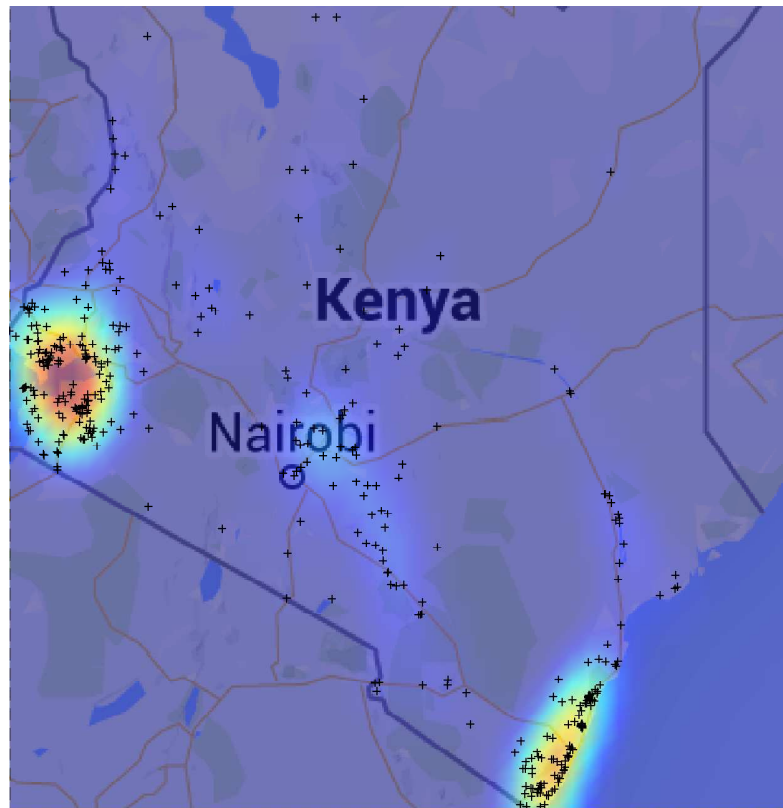


Figure 6.8: The intensity of malaria incidences in Kenya based on a sample of 741 malaria incidences that occurred between 1985 and 2010, and the associated VBPP intensity function. Inducing points located on a 20×20 grid.

7

Models for complex high dimensional data

In this chapter we will see three developments of the variational inference scheme for Gaussian process modulated Poisson processes outlined in the previous chapter. In the first development, we will see how the cost of variational Gaussian process inference (generally) can be reduced in high dimensions—compared with a naïve approach—by exploiting Kronecker structure. The following sections describe two new methods for constructing complex multi-output structured point processes. These methods are similar in spirit to, but different from, the method described in Chapter 5. However, before we move on, it is worth emphasising that the convolution process prior can be combined with the variational method since the output functions f_s are still marginally Gaussian and the convolution kernels, Equation 5.7, allow the integral over the domain, i.e. Ψ_{zz} , to be calculated. The only significant change to the model required, is to switch the link function from the sigmoid to the square transform. This prior can also be used with the Kronecker structure and positive factor model approaches described in this chapter, although preserving Kronecker structure and the associated computational benefits requires careful implementation, as we shall now see.

7.1 Kronecker structure

In order for the integrals in Equations 6.31 and 6.32 to have a closed-form, the Gaussian Process (GP) kernel must be *separable*. A separable kernel, $K_{\text{sep}}(\mathbf{x}, \mathbf{x}')$, is one that can be written as the product of kernels with no shared coordinates:

$$K_{\text{sep}}(\mathbf{x}, \mathbf{x}') = \prod_{r=1}^R K_r(x_r, x'_r). \quad (7.1)$$

Kernel matrices constructed using separable kernels, for example Equation 2.77, have implicit Kronecker structure. To take advantage of the computational efficiencies which Kronecker structure offers, we make this structuring explicit, both in our mathematical development and especially in our software implementation. Doing so dramatically reduces the headline computational complexity of both the Poisson process inference algorithm and, more generally, variational Gaussian process regression (§3.3.2).

To achieve this structuring we introduce a separate set of inducing points $\mathcal{Z}_r \triangleq \{z_r^{(m)} \in [\mathcal{X}_r^{\min}, \mathcal{X}_r^{\max}]\}_{m=1}^{M_r}$ for each dimension r , so that the overall inducing point set is the cross product of these sets, i.e. $\mathcal{Z} = \mathcal{Z}_1 \times \dots \times \mathcal{Z}_R$ and $M = \prod_r M_r$. Consequently we will never need to construct full $[M \times M]$ matrices to compute the Evidence Lower Bounds (ELBOs) $\mathcal{L}_{\text{GP}}$, $\mathcal{L}_{\text{GPLVM}}$, $\mathcal{L}_{\text{PP}}$, Equations 3.71, 4.14, 6.10, nor any of their derivatives (see §A.3).

Using these inducing points $\mathbf{K}_{zz}$ and Ψ_{zz} become the Kronecker product of R matrices:

$$\mathbf{K}_{zz} = \bigotimes_{r=1}^R \mathbf{K}_{z_r z_r} \quad (7.2)$$

$$\Psi_{zz} = \bigotimes_{r=1}^R \Psi_{z_r z_r}, \quad (7.3)$$

where $\mathbf{K}_{z_r z_r}$ and $\Psi_{z_r z_r}$ are $[M_r \times M_r]$ matrices with entries

$$[\mathbf{K}_{z_r z_r}]_{m, m'} = K_r(z_r^{(m)}, z_r^{(m')}) \quad (7.4)$$

$$[\Psi_{z_r z_r}]_{m, m'} = \Psi_r(z_r^{(m)}, z_r^{(m')}), \quad (7.5)$$

and $\Psi_r(z, z') = \int_{\mathcal{X}_r^{\min}}^{\mathcal{X}_r^{\max}} K_r(z, x) K_r(x, z') dx$. We also need to maintain Kronecker structure of the cross-kernels $\mathbf{K}_{zx} = \mathbf{K}_{xz}^\top$. Therefore for each data point, $\mathbf{x}^{(n)}$, we have a Kronecker structured vector

$$\mathbf{k}_{zx}^{(n)} = \bigotimes_{r=1}^R \mathbf{k}_{z_r x_r}^{(n)}. \quad (7.6)$$

We store the constituent matrices separately and operate on them individually to perform inference. By applying the Kronecker matrix identities for inversion, multiplication and trace, which are given in Table 7.1, we can compute, for example, $\text{tr}(\mathbf{K}_{zz}^{-1} \Psi_{zz})$ according to:

$$\text{tr} \left(\bigotimes_{r=1}^R \mathbf{K}_{z_r z_r}^{-1} \Psi_{z_r z_r} \right) = \prod_{r=1}^R \text{tr} \left(\mathbf{K}_{z_r z_r}^{-1} \Psi_{z_r z_r} \right), \quad (7.7)$$

which only requires multiplication and inversion of M_r sized matrices. As a result we reduce the computational complexity of the lower bound from $\mathcal{O}(M^2 N)$ to $\mathcal{O}(M_{\max}^2 N)$, where $M_{\max} = \max_r(M_r)$.

There are two important consequences of this structuring. The first is that inducing points cannot be moved independently. We can only control $z_r^{(m)}$, the r^{th} coordinate of M/M_r inducing points in $\mathcal{Z}$. Secondly, it is also necessary that $\mathbf{S}$, the covariance of the variational distributions $q(\mathbf{u})$, also has Kronecker structure. We can achieve this simply by defining $\mathbf{S}$ to be the Kronecker product of R positive semi-definite matrices $\mathbf{S}_r$; to be explicit:

$$\mathbf{S} = \bigotimes_{r=1}^R \mathbf{S}_r, \quad (7.8)$$

where $\mathbf{S}_r = \mathbf{L}_r \mathbf{L}_r^\top$. It is important to note that this places a restriction on the space of possible covariance matrices; hence the variational distribution in the Kronecker case is necessarily less flexible when compared with the full matrix case. Therefore, the tightest possible variational bound achievable in the Kronecker case will not be as tight as the full case and may also lead to a more complicated optimisation landscape. Recent work by Hensman, Durrande, et al. (2016) suggests that defining

$\mathbf{S}$ to be the sum of *two* Kronecker structured matrices:

$$\mathbf{S} = \tilde{\mathbf{S}} + \bar{\mathbf{S}} \quad (7.9)$$

$$\tilde{\mathbf{S}} = \bigotimes_{r=1}^R \tilde{\mathbf{S}}_r \quad (7.10)$$

$$\bar{\mathbf{S}} = \bigotimes_{r=1}^R \bar{\mathbf{S}}_r. \quad (7.11)$$

significantly improves flexibility and yields tighter bounds. Since the Kronecker product distributes over addition, $(\mathbf{A} + \mathbf{B}) \otimes \mathbf{C} = \mathbf{A} \otimes \mathbf{C} + \mathbf{B} \otimes \mathbf{C}$, we can expand all the terms in the ELBO and maintain Kronecker structure until the results of all computations have been reduced to full vectors. The exception is the computation of $\log \det(\tilde{\mathbf{S}} + \bar{\mathbf{S}})$. Hensman, Durrande, et al. recommend the method of Rakitsch et al. (2013) which is:

$$\log |\mathbf{S}| = \left[\sum_{d \in \underline{\mathbf{D}}} \log(1 + d) \right] + \log |\tilde{\mathbf{S}}|, \quad (7.12)$$

where $\underline{\mathbf{D}}$ is an R -mode tensor, with M total elements, formed from the Kronecker product of R vectors:

$$\underline{\text{vec}}(\underline{\mathbf{D}}) = \bigotimes_{r=1}^R \mathbf{d}_r, \quad (7.13)$$

where $\mathbf{d}_r$ are the Eigenvalues of $[\tilde{\mathbf{S}}_r^{-1} \bar{\mathbf{S}}_r]$ or $[\tilde{\mathbf{L}}_r^{-1} \bar{\mathbf{L}}_r][\bar{\mathbf{L}}_r^\top \tilde{\mathbf{L}}_r^{-\top}]$, the latter may be preferable due to its Hermitian structure.

We can also exploit Kronecker structure for calculations involving $\mathbf{m}$, the mean of $q(\mathbf{u})$, even when the former is represented using full vectors via the relationship:

$$[\mathbf{A}_1 \otimes \mathbf{A}_2 \otimes \dots \otimes \mathbf{A}_R] \cdot \mathbf{b}_R = [\mathbf{A}_1 \otimes \mathbf{A}_2 \otimes \dots \otimes \mathbf{A}_{R-1}] \cdot \text{vec}(\mathbf{B}_R^\top \cdot \mathbf{A}_R^\top) \quad (7.14)$$

$$= [\mathbf{A}_1 \otimes \mathbf{A}_2 \otimes \dots \otimes \mathbf{A}_{R-1}] \cdot \mathbf{b}_{R-1}. \quad (7.15)$$

The total computational cost for R applications of Equations 7.14, 7.15—assuming $\mathbf{A}_r$ are square matrices of size M_r , $\mathbf{b}_r$ are M length vectors and $\mathbf{B}_r$ are $[M_r \times \prod_{i=1}^R M_i^{1\{i \neq r\}}]$ sized matrices—is

$$\sum_{r=1}^R [M_r \times M_r \times \prod_{i=1}^R M_i^{1\{i \neq r\}}] = M \sum_{r=1}^R M_r, \quad (7.16)$$

Inverse	$[\mathbf{A}_1 \otimes, \dots, \otimes \mathbf{A}_R]^{-1}$	$=$	$\mathbf{A}_1^{-1} \otimes, \dots, \otimes \mathbf{A}_R^{-1}$
Cholesky	$\text{chol}(\mathbf{A}_1 \otimes, \dots, \otimes \mathbf{A}_R)$	$=$	$\text{chol}(\mathbf{A}_1) \otimes, \dots, \otimes \text{chol}(\mathbf{A}_R)$
Trace	$\text{tr}(\mathbf{A}_1 \otimes, \dots, \otimes \mathbf{A}_R)$	$=$	$\text{tr}(\mathbf{A}_1) \times, \dots, \times \text{tr}(\mathbf{A}_R)$
Log det.	$\log \mathbf{A}_1 \otimes, \dots, \otimes \mathbf{A}_R $	$=$	$[\frac{\log \mathbf{A}_1 }{N_1} +, \dots, + \frac{\log \mathbf{A}_R }{N_R}] \cdot \prod_r N_r$
Dot prod.	$[\mathbf{A}_1 \otimes, \dots, \otimes \mathbf{A}_R][\mathbf{B}_1 \otimes, \dots, \otimes \mathbf{B}_R]$	$=$	$\mathbf{A}_1 \mathbf{B}_1 \otimes, \dots, \otimes \mathbf{A}_R \mathbf{B}_R$
Hadamard	$[\mathbf{A}_1 \otimes, \dots, \otimes \mathbf{A}_R] \odot [\mathbf{B}_1 \otimes, \dots, \otimes \mathbf{B}_R]$	$=$	$\mathbf{A}_1 \odot \mathbf{B}_1 \otimes, \dots, \otimes \mathbf{A}_R \odot \mathbf{B}_R$

Table 7.1: Matrix operations which can preserve or exploit Kronecker structure.

which is manageable for a low number of dimensions and relatively few inducing points. In higher dimensions, M length vectors may become too large to store or compute with. However it is possible to represent $\mathbf{m}$ as the sum of Kronecker structured matrices, although, as was the case for $\mathbf{S}$, doing so will place some restrictions on $q(\mathbf{u})$. In this case we have much more flexibility to use a larger number, say C , of Kronecker vectors,

$$\mathbf{m} = \sum_{c=1}^C \mathbf{m}_{1,c} \otimes, \dots, \otimes, \mathbf{m}_{R,c}, \quad (7.17)$$

since we are not constrained by the calculation of determinants.

7.1.1 Relationship to tensor factorisation

In §4.3.3 we used the Kronecker product to define the CP and Tucker tensor decompositions. It is reasonable therefore to explore how—if at all—Kronecker structured variational Gaussian process models are related to tensor decomposition models. For the sake of clarity we will (without loss of generality) restrict our discussion to three dimensions and note that our conclusions apply equally to “ P -mode” tensors and R -dimensional GP latent spaces (where $P = R$).

Intuitively, some relationship seems reasonable since the inducing variables, $\mathbf{u}$, of a sparse or variational GP and the core tensor of the Tucker decomposition, $\underline{\mathbf{U}}$, are both compressed representations of, respectively, a function f , and a tensor $\underline{\mathbf{Y}}$. Furthermore, the dimensions of $\mathbf{u}$, which we have previously defined as $[M_1 \times M_2 \times M_3]$, and the dimensions of $\underline{\mathbf{U}}$, which we have previously defined as $[H \times M \times N]$,

are user specified parameters that determine the degree of compression in each dimension. We can therefore choose $N := M_1$, $M := M_2$, $H := M_3$.

If we expand and rearrange the expectation of f at a particular point $\mathbf{x}^{(n)}$ under the distribution $q(f)$ as follows:

$$\mathbb{E}_{q(f)}[f(\mathbf{x}^{(n)})|\mathbf{u}] = \mathbf{k}_{xz}^{(n)} \mathbf{K}_{zz}^{-1} \mathbf{m} \quad (7.18)$$

$$= [\mathbf{k}_{x_1 z_1}^{(n)} \otimes \mathbf{k}_{x_2 z_2}^{(n)} \otimes \mathbf{k}_{x_3 z_3}^{(n)}] \cdot [\mathbf{K}_{z_1 z_1} \otimes \mathbf{K}_{z_2 z_2} \otimes \mathbf{K}_{z_3 z_3}]^{-1} \cdot \mathbf{m} \quad (7.19)$$

$$= [\mathbf{k}_{x_1 z_1}^{(n)} \mathbf{K}_{z_1 z_1}^{-1} \otimes \mathbf{k}_{x_2 z_2}^{(n)} \mathbf{K}_{z_2 z_2}^{-1} \otimes \mathbf{k}_{x_3 z_3}^{(n)} \mathbf{K}_{z_3 z_3}^{-1}] \cdot \mathbf{m}, \quad (7.20)$$

and then make the following substitutions:

$$\bar{\mathbf{t}}_k := \mathbf{k}_{x_1 z_1}^{(n)} \mathbf{K}_{z_1 z_1}^{-1} \quad (7.21)$$

$$\bar{\mathbf{r}}_j := \mathbf{k}_{x_2 z_2}^{(n)} \mathbf{K}_{z_2 z_2}^{-1} \quad (7.22)$$

$$\bar{\mathbf{s}}_i := \mathbf{k}_{x_3 z_3}^{(n)} \mathbf{K}_{z_3 z_3}^{-1} \quad (7.23)$$

$$\text{vec}(\underline{\mathbf{U}}) := \mathbf{m}, \quad (7.24)$$

we recover the approximation for the $i^{\text{th}} j^{\text{th}} k^{\text{th}}$ element of the tensor $\underline{\mathbf{Y}}$ given by Equation 4.32. This isn't quite the complete story however, since in a tensor decomposition each vector, say $\bar{\mathbf{t}}_k$, is common to all entries $y_{i',j',k}$. In a general GP regression setting however, the vector $\mathbf{k}_{x_1 z_1}^{(n)}$ is not necessarily in common with any other data point $\mathbf{x}^{(n')}$. The exception to this is latent variable GP models, such as the Gaussian Process Latent Variable Model (GPLVM) (§4.2) and in particular the Random Function Model (RFM) (§4.3.2). If we consider the application of the RFM to a 3-mode tensor with elements $y_{i,j,k} = f(\mathbf{x}^{(i,j,k)}) + \epsilon_{i,j,k}$, where the latent coordinates are tied in the latent space by the relationship $\mathbf{x}^{(i,j,k)} = [x_1^{(k)}, x_2^{(j)}, x_3^{(i)}]^\top$, then the vector $\mathbf{k}_{x_1 z_1}^{(k)}$ is indeed shared by all entries $y_{i',j',k}$. In conclusion therefore, the RFM, combined with a sparse or variational approximation and a separable kernel, is a special case of the Tucker tensor decomposition.

We now consider the alternative variational mean parametrisation, Equation 7.17. As before, expanding and rearranging leads to:

$$\begin{aligned} \mathbb{E}_{q(f)}[f(\mathbf{x}^{(n)})|\mathbf{u}] &= [\mathbf{k}_{x_1 z_1}^{(n)} \mathbf{K}_{z_1 z_1}^{-1} \otimes \mathbf{k}_{x_2 z_2}^{(n)} \mathbf{K}_{z_2 z_2}^{-1} \otimes \mathbf{k}_{x_3 z_3}^{(n)} \mathbf{K}_{z_3 z_3}^{-1}] \\ &\quad \cdot \sum_{c=1}^C \mathbf{m}_{1,c} \otimes \mathbf{m}_{2,c} \otimes \mathbf{m}_{3,c} \end{aligned} \quad (7.25)$$

$$= \sum_{c=1}^C [\mathbf{k}_{x_1 z_1}^{(n)} \mathbf{K}_{z_1 z_1}^{-1} \mathbf{m}_{1,c} \otimes \mathbf{k}_{x_2 z_2}^{(n)} \mathbf{K}_{z_2 z_2}^{-1} \mathbf{m}_{2,c} \otimes \mathbf{k}_{x_3 z_3}^{(n)} \mathbf{K}_{z_3 z_3}^{-1} \mathbf{m}_{3,c}], \quad (7.26)$$

and then making the following substitutions:

$$t_{k,c} := \mathbf{k}_{x_1 z_1}^{(n)} \mathbf{K}_{z_1 z_1}^{-1} \mathbf{m}_{1,c} \quad (7.27)$$

$$r_{j,c} := \mathbf{k}_{x_2 z_2}^{(n)} \mathbf{K}_{z_2 z_2}^{-1} \mathbf{m}_{2,c} \quad (7.28)$$

$$s_{i,c} := \mathbf{k}_{x_3 z_3}^{(n)} \mathbf{K}_{z_3 z_3}^{-1} \mathbf{m}_{3,c}, \quad (7.29)$$

we recover the approximation for the $i^{\text{th}} j^{\text{th}} k^{\text{th}}$ element of a tensor using the CP method, Equation 4.29. The forgoing statements regarding the RFM model are equally applicable here and so we see that the RFM, combined with a sparse or variational approximation and separable kernels *and* a Kronecker structured mean representation, is a special case of the CP tensor decomposition.

In both these cases, elements of the tensor components are no longer free variables, but rather are constrained to be smoothly varying functions of the kernel. The kernel parameters also control $\mathbf{m}$ through the Kullback–Leibler (KL) divergence term.

7.1.2 Implementation

Taking advantage of Kronecker structure improves computational scalability at the cost of additional software complexity. To manage the additional software complexity created by the use of Kronecker matrices, it is helpful to encapsulate all Kronecker operations in a separate class that abstracts the underlying manipulations. An overview of such a class is given in Appendix B. The derivatives of $\mathcal{L}_{\text{PP}}$ (and

$\mathcal{L}_{\text{GP}}$) are not affected by Kronecker structure except for the derivative with respect to $\mathbf{S}$, details of which are given in Appendix A.

One important implementation detail is to keep all the $\mathbf{k}_{z_r x_r}^{(n)}$ vectors in contiguous memory by stacking them so that

$$\mathbf{K}_{z_r x_r}^{(1:N)} = [\mathbf{k}_{z_r x_r}^{(1)}, \dots, \mathbf{k}_{z_r x_r}^{(N)}], \quad (7.30)$$

as this allows efficient use of low-level matrix libraries. For example, to compute $\mathbf{K}_{xz}^{(1:N)} \mathbf{K}_{zz}^{-1}$ we need only R Basic Linear Algebra System (BLAS) function calls rather than NR calls.

Maintaining explicit Kronecker structure in software not only helps improve computational complexity, it can also help to improve the basic efficiency of an implementation by allowing us to more easily isolate repeated calculations. If we consider the derivative of the kernel function with respect to, say, an input length scale α_r

$$\frac{\partial K(\mathbf{z}, \mathbf{z}'; \alpha_{1:R})}{\partial \alpha_r} = \prod_{i=1}^{r-1} K_i(z_i, z'_i; \alpha_i) \cdot \frac{\partial K_r(z_r, z'_r; \alpha_r)}{\partial \alpha_r} \cdot \prod_{i=r+1}^R K_i(z_i, z'_i; \alpha_i), \quad (7.31)$$

and the associated matrices

$$\frac{\partial \mathbf{K}_{zz}}{\partial \alpha_r} = \left[\bigotimes_{i=1}^{r-1} \mathbf{K}_{z_i z_i} \right] \otimes \mathbf{K}_{z_r z_r}^{(0,0,1)} \otimes \left[\bigotimes_{i=r+1}^R \mathbf{K}_{z_i z_i} \right], \quad (7.32)$$

we see most of the matrices needed to calculate both the ELBO and the derivative(s) are common to most of these computations.

Hand-coding software to isolate all these optimisation opportunities is difficult and motivates the use of packages such as Theano that support computational graph optimisation and automatic differentiation via back-propagation. It was found that calculation of gradients using Theano back-propagation was far more efficient than hand coded derivatives. This was especially true for gradients with respect to parameters that appear in kernel matrix calculations. Efficient calculation of such gradients is particularly important for the latent variable model, described in §7.3, which requires a large number of derivatives with respect to the latent variables that

are all kernel parameters. More complex models and variational distributions, such as Equations 7.9 and 7.17, further motivate the use of automatic differentiation software.

7.2 Latent Poisson Process Allocation

In this section we will develop a linear factor model for Poisson processes we call Latent Poisson Process Allocation (LPPA) (Lloyd, Gunter, Nickson, et al. 2016). The model is conceptually a continuous time variation of Non-negative Matrix Factorisation (NMF) or a Poisson process variation of the Semi-Parametric Factor Model (SPFM) (Teh, Seeger, et al. 2005) in which the bases are (squared) Gaussian processes and the weights are a set of output length scales. To construct Poisson process factor models, we begin by considering a *marked* Poisson process. A marked Cox process over events

$$\mathcal{M} \triangleq \left\{ \left(\mathbf{x}^{(n)}, A(\mathbf{x}^{(n)}) \right) \mid \mathbf{x}^{(n)} \in \mathbb{R}^R, A(\mathbf{x}^{(n)}) \in \mathcal{T} \right\}, \quad (7.33)$$

extends a Cox process by associating with each point, $\mathbf{x}$, an additional piece of information $A(\mathbf{x}) \in \mathcal{T}$ called a mark. The form of mark itself is very general; it can be a discrete random variable, real valued random variable or indeed another point process. The mark assigned at any point can depend on the location and the value of the rate function at that location.

We consider a multivariate (discrete) mark set $\mathcal{T} \triangleq \{1, \dots, T\}$. As the marks are discrete there will be a rate function, $\lambda_t(\mathbf{x})$, associated with each mark t . Furthermore, since the set of all points, $\mathcal{M}$, is equal to the union of all sets $\mathcal{M}_t \triangleq \left\{ \left(\mathbf{x}^{(n)}, t \right) \mid \mathbf{x}^{(n)} \in \mathbb{R}^R \right\}$, the overall rate function must be the sum of the individual rate functions.

Therefore $\lambda(\mathbf{x}) = \sum_{t=1}^T \lambda_t(\mathbf{x})$ and, using the Poisson-multinomial connection, the probability of mark t at a point $\mathbf{x}$ is $p(t; \mathbf{x}) = \lambda_t(\mathbf{x}) / \lambda(\mathbf{x})$. In general the individual rate functions λ_t might not be assumed a priori independent, although in the model we develop in the next section we will assume they are.

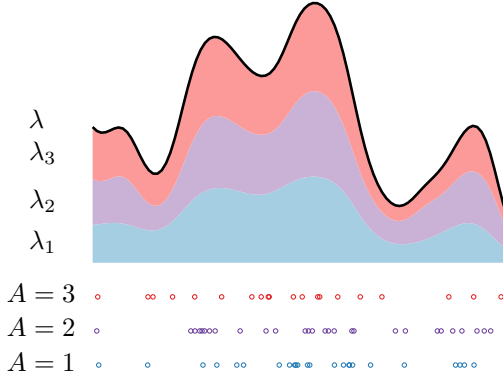


Figure 7.1: Correlated latent intensity functions constructed using a convolution process.

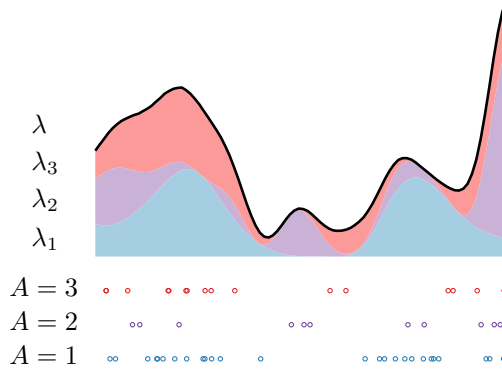


Figure 7.2: Uncorrelated square transformed GP latent intensity functions.

In this framework, the probability density of a set of N observed marked points $\mathcal{M} = \{(\mathbf{x}^{(n)}, A^{(n)})\}_{n=1}^N$, where $A^{(n)} = A(\mathbf{x}^{(n)})$, in some bounded region, $\mathcal{X}$, factorises conditioned on the rate processes:

$$p(\mathcal{M} \mid \lambda_{1:T}) = p(\{\mathbf{x}^{(n)}\}_{n=1}^N \mid \lambda_{1:T}) p(\{A^{(n)}\}_{n=1}^N \mid \{\mathbf{x}^{(n)}\}_{n=1}^N, \lambda_{1:T}). \quad (7.34)$$

Using the Poisson-multinomial connection (§2.3.4) probability of the N labels is

$$p(\{A^{(n)}\}_{n=1}^N \mid \{\mathbf{x}^{(n)}\}_{n=1}^N, \lambda_{1:T}) = \prod_{n=1}^N \frac{\lambda_{A^{(n)}}(\mathbf{x}^{(n)})}{\lambda(\mathbf{x}^{(n)})}. \quad (7.35)$$

A standard inference task for multiple point processes would be to learn the labelling distribution and unobserved point process intensity λ . For independent rate functions, this reduces to simply inferring the rates functions λ_t from subset of points marked with the corresponding mark t .

If the rate functions are not independent, then we may seek to refine our model by constructing a structural prior that allows statistical strength to be shared between the point processes associated with each label. The latter approach was used in Chapter 5, using a convolution process to tie the rate processes together, as shown in Figure 7.1.

We will assume the marks $\mathcal{T}$ are unobserved latent variables to be inferred. Due to a lack of observability, this is impossible for a single point process. To increase observability we need to observe multiple point process in which the latent rate

functions are present in linearly independent proportions. We will designate each of these point processes a separate *output* yielding a dataset $\mathcal{D}_s \triangleq \{\mathbf{x}^{(s,n)} \in \mathcal{X}\}_{n=1}^{N_s}$ and each of which will be tied to its own set of output rate functions $\lambda_{s,t}$. Since these outputs are themselves marks in $\mathcal{S} = \{1, \dots, S\}$, their superposition is a marked point process. There are therefore two distinct sets of marks: observed marks in $\mathcal{S}$ corresponding to the outputs and unobserved marks in $\mathcal{T}$ corresponding to the latent function.

7.2.1 Model

We will assume that the intensity of the t^{th} latent function in the s^{th} output is $\lambda_{s,t}(\mathbf{x}) = \gamma_{s,t} f_t^2(\mathbf{x})$, where the functions f_t are independent Gaussian process distributed random functions. The variables $\gamma_{s,t}$ are essentially the output length scales of the covariance function, however since we have more output length scales than functions we bring them outside of the covariance function. We continue to use a separable exponentiated quadratic kernel function, sans output length scale, i.e:

$$K(\mathbf{x}, \mathbf{x}') = \prod_{r=1}^R \exp\left(\frac{x_r - x'_r}{\alpha_r}\right). \quad (7.36)$$

We will assume for notational convenience that each latent function f_t has uses a common set of inducing points and input length scales, although it is entirely possible to relax this constraint, and denote the evaluation of f_t at these points $\mathbf{u}_t$.

Combining the Cox process likelihood, Equation 2.83, and mark likelihood, Equation 7.35, using indicator variables, square-transformed Gaussian process rate functions and output length scales, $\gamma_{s,t}$, gives

$$p(\mathcal{D}_s, A_s \mid f_{1:T}, \Gamma) = \prod_{t=1}^T \exp\left(-\int \gamma_{s,t} f_t^2(\mathbf{x}) d\mathbf{x}\right) \prod_{n=1}^{N_s} [\gamma_{s,t} f_t^2(\mathbf{x}^{(s,n)})]^{\mathbb{1}\{A_s^{(n)}=t\}}. \quad (7.37)$$

where $\Gamma \in \mathbb{R}_+^{S \times T}$ is a matrix of output length scales with elements $\gamma_{s,t}$. The joint distribution of $\mathcal{D}_{1:S}$, $f_{1:T}$, $\mathbf{u}_{1:T}$ and $A_{1:S}$ in this hierarchy is

$$p(\mathcal{D}_{1:S}, A_{1:S}, f_{1:T}, \mathbf{u}_{1:T} \mid \Omega) = \prod_{s=1}^S p(\mathcal{D}_s, A_s \mid f_{1:T}) \cdot \prod_{t=1}^T p(f_t \mid \mathbf{u}_t) p(\mathbf{u}_t), \quad (7.38)$$

where $\Omega \triangleq \{\Gamma, \alpha_{1:R}, \bar{u}_{1:T}\}$ is the set of model parameters. For notational convenience we will often omit conditioning on Ω .

7.2.2 Collapsed variational bound

We begin by integrating out the latent function variables $\mathbf{u}_{1:T}$ using a mean field variational distribution $q(\mathbf{u}_{1:T}) = \prod_t q(\mathbf{u}_t)$.

$$\begin{aligned} \log p(\mathcal{D}_{1:S}, A_{1:S} | \Omega) &\geq \mathbb{E}_{q(f_{1:T})} \left[\log p(\mathcal{D}_{1:S}, A_{1:S} | f_{1:T}) \right] - \sum_{t=1}^T \text{KL}(q(\mathbf{u}_t) || p(\mathbf{u}_t)) \quad (7.39) \\ &\triangleq \mathcal{L}(A_{1:S}). \end{aligned}$$

Due to the mean field assumption, we can integrate each $p(f_t | \mathbf{u}_t)$ against $q(\mathbf{u}_t)$ separately and thereby obtain

$$q(f_t) = \int p(f_t | \mathbf{u}_t) q(\mathbf{u}_t) d\mathbf{u}_t = \mathcal{GP}(f_t(\mathbf{x}); \tilde{\mu}_t(\mathbf{x}), \tilde{\Sigma}_t(\mathbf{x}, \mathbf{x}')), \quad (7.40)$$

where $\tilde{\mu}_t(\mathbf{x}) = \mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \mathbf{m}_t$ and $\tilde{\Sigma}_t(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') - \mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \mathbf{k}_{zx'} + \mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \mathbf{S}_t \mathbf{K}_{zz}^{-1} \mathbf{k}_{zx'}$ and $q(f_{1:T}) = \prod_t q(f_t)$. Using Equation the following shorthand definitions

$$\begin{aligned} f_{t,x} &\triangleq f_t(\mathbf{x}), & \tilde{\mu}_{t,x} &\triangleq \tilde{\mu}_t(\mathbf{x}), & \tilde{\sigma}_{t,x}^2 &\triangleq \tilde{\Sigma}_t(\mathbf{x}, \mathbf{x}), \\ f_{s,t,n} &\triangleq f_t(\mathbf{x}^{(s,n)}), & \tilde{\mu}_{s,t,n} &\triangleq \tilde{\mu}_t(\mathbf{x}^{(s,n)}), & \tilde{\sigma}_{s,t,n}^2 &\triangleq \tilde{\Sigma}_t(\mathbf{x}^{(s,n)}, \mathbf{x}^{(s,n)}), \end{aligned}$$

expanding Equation 7.37 yields:

$$\begin{aligned} \mathbb{E}_{q(f_{1:T})} [\log p(\mathcal{D}_{1:S}, A_{1:S} | f_{1:T})] &= \sum_s \left[- \sum_t \gamma_{s,t} \int_{\mathcal{X}} \left(\mathbb{E}_{q(f_t)} [f_{t,x}]^2 + \text{Var}_{q(f_t)} [f_{t,x}] \right) d\mathbf{x} \right. \\ &\quad \left. + \sum_n \sum_t \mathbb{1}\{A_s^{(n)} = t\} \left(\log(\gamma_{s,t}) + \mathbb{E}_{q(f_t)} [\log f_{s,t,n}^2] \right) \right]. \quad (7.41) \end{aligned}$$

Equation 7.41 contains a large number of multivariate allocation variables $A_s^{(n)}$ —hence the name Latent Poisson Process Allocation (LPPA)—that indicate which of the latent processes generated each point.

The standard variational inference approach to this problem would be to marginalise these variables using a variational distribution $q(A_{1:S})$ and to update $q(\mathbf{u}_{1:T})$, $q(A_{1:S})$ and Ω alternately using coordinate ascent. Instead we prefer to collapse out the allocation variables and update the variational parameters and model parameters in a single joint optimisation.

Collapsing the indicator variables has two primary benefits: firstly it reduces the number of variables which must be explicitly updated via a marginal gradient step

at each iteration; secondly, as we have analytically marginalised (in this case a large subset of) the unknown variables, the implicit updates of those unknown variables will occur with greater efficiency (in the sense that they will converge to a solution faster) (Hensman, Rattray, et al. 2012).

To do this we first note that we can write Equation 7.39 as the sum of a set of variables $\mathfrak{A}_{s,t,n} = \log(\gamma_{s,t}) + \mathfrak{G}_{s,t,n}$ which multiply the indicator variables $A_s^{(n)}$ and a term $\mathfrak{B}$, that does not, resulting in the compact definition:

$$\mathcal{L}(A_{1:S}) = \mathfrak{B} + \sum_{s,t,n} \mathbb{1}\{A_s^{(n)} = t\} \mathfrak{A}_{s,t,n}. \quad (7.42)$$

To collapse the bound we sum over all the possible assignments to each of the allocation variables. The set of all possible assignments to each of the allocation variables is:

$$\left\{ \{A_1^{(1)} = 1, \dots, A_S^{(N_S)} = 1\}, \dots, \{A_1^{(1)} = T, \dots, A_S^{(N_S)} = T\} \right\}, \quad (7.43)$$

and in the collapsed bound we sum over all these possible assignments to give $\mathcal{L}_{\text{LPPA}}$:

$$\begin{aligned} & \log p(\mathcal{D}_{1:S} | \Omega) \\ &= \log \sum_{A_{1:S}} p(\mathcal{D}_{1:S}, A_{1:S} | \Omega) \end{aligned} \quad (7.44)$$

$$\geq \log \sum_{A_{1:S}} \exp(\mathcal{L}(A_{1:S})) \quad (7.45)$$

$$= \log \sum_A \exp\left(\mathfrak{B} + \sum_s \sum_n \sum_t \mathbb{1}\{A_s^{(n)} = t\} \mathfrak{A}_{s,t,n}\right) \quad (7.46)$$

$$= \log \left[\exp(\mathfrak{B}) \sum_{A_1^{(1)}=1}^T \cdots \sum_{A_S^{(N_S)}=1}^T \prod_s \prod_n \exp\left(\sum_t \mathbb{1}\{A_s^{(n)} = t\} \mathfrak{A}_{s,t,n}\right) \right] \quad (7.47)$$

$$= \log \left[\exp(\mathfrak{B}) \prod_s \prod_n \sum_{A_s^{(n)}=1}^T \exp\left(\sum_t \mathbb{1}\{A_s^{(n)} = t\} \mathfrak{A}_{s,t,n}\right) \right] \quad (7.48)$$

$$= \log \left[\exp(\mathfrak{B}) \prod_s \prod_n \sum_t \exp(\mathfrak{A}_{s,t,n}) \right] \quad (7.49)$$

$$= \mathfrak{B} + \sum_s \sum_n \log \sum_t \exp \mathfrak{A}_{s,t,n} \quad (7.50)$$

$$\triangleq \mathcal{L}_{\text{LPPA}}.$$

As before, we optimise $\mathcal{L}_{\text{LPPA}}$ using L-BFGS.

7.2.3 Model identifiability

Like most factor models—NMF, Latent Dirichlet Allocation (LDA) (Blei, Ng, et al. 2003), etc.—LPPA is non-identifiable and non-unique; there may be multiple decompositions that are well supported by the data and the decomposition found is sensitive to initial conditions. The Gaussian process prior and Bayesian shrinkage will tend to reduce the non-identifiability of the solutions, by expressing a preference for smooth latent functions. Bayesian shrinkage will also tend to prevent over-fitting the number of latent functions.

The requirements for LPPA to generate unique solutions are likely to be similar to those for unique NMF solutions, i.e. complete factorial sampling of linearly independent basis (Donoho and Stodden 2003).

7.2.4 Dynamic network model

We can easily extend the LPPA model to handle Dynamic Interaction Network (DIN), described in Chapter 4, by constructing a semi-parametric tensor decomposition model that we refer to as Network Latent Poisson Process Allocation (N-LPPA) (Lloyd et al. 2015a). This semi-parametric tensor model has intensity functions

$$\lambda_{i,j} = \sum_{c_1=1}^{C_1} \sum_{c_2=1}^{C_2} \phi_{i,c_1} f_{c_1,c_2}^2 \varphi_{j,c_2}. \quad (7.51)$$

Since in our experiments the “sender” and “receiver” nodes are drawn from the same set (it is not a bipartite graph) we can reduce the number of free parameters by requiring $C_1 := C_2 := C$ and $\varphi_{c_1,c_2} := \phi_{c_1,c_2}$ where C is the number of “communities”. As the computational complexity of this algorithm increases quadratically in C , we could reduce complexity by a simpler model in which only intra-community interaction is allowed by setting $f_{c_1,c_2}^2 := 0$ for $c_1 \neq c_2$. The latter can be thought of as a semi-parametric CP tensor decomposition¹.

¹In the semi-parametric CP case setting $\varphi_{c_1,c_2} := \phi_{c_1,c_2}$ will cause the model to have symmetric intensity functions.

To implement N-LPPA we simply need to map $\mathcal{D}_{i,j}$ and $\lambda_{i,j}$ to $\mathcal{D}_s$ and λ_s , to map f_{c_1,c_2}^2 to f_t^2 and compute $\gamma_{s,t}$ from ϕ_{i,c_1} and φ_{j,c_2} . These mappings will be different depending on whether we wish to model a symmetric network with $\mathcal{D}_{i,j} = \mathcal{D}_{j,i}$, and/or a network in which reflexive interaction is by definition empty, i.e. $\mathcal{D}_{i,i} = \emptyset$, thus making no contribution to the likelihood.

Since the temporal domain is not discretised in N-LPPA, event processes with zero entries have very little computational cost, only requiring the calculation of the relevant output length scales—a fast matrix operation. In practice, the computational cost of LPPA is dominated by the number of latent functions and the total number of observations. This is useful for modelling sparse dynamic interaction networks since, very often, many pairs of nodes will not interact. In contrast, if we were to discretise the time domain and use a tensor decomposition, there may be—depending on the fidelity—many empty resolution cells, each of which adds an additional computational burden. This is an important benefit of LPPA, but one which becomes even more pronounced in the case of dynamic network data.

7.2.5 Experiments

In this section we will motivate and demonstrate the application of LPPA to several real world datasets. We do not include experiments on synthetic data with known ground truth rate functions as they are not very informative. Due to non-identifiability the learned latent rate functions may not be very similar to the generative latent functions in a mean-squared-error sense. The learned rate functions are often sparse hybrids of the generative functions. Approximate model recovery using LPPA usually requires a large number of points since the rate functions are only weakly observed, with each latent function dominating one of the outputs. However this is not representative of real world use on which we focus. Neither model recovery nor identifiability is required for the model to be of practical

value since the outputs are identifiable and, as we shall demonstrate, the model does improve predictive performance and reveal latent structure.

7.2.5.1 Benchmark

The LPPA algorithm was benchmarked against a method combining Kernel Smoothing (KS) and Poisson-NMF (Lee and Seung 2001). In this method, we first smooth each dataset using truncated normal densities as described in §6.5.1. Next we integrate the rate functions, λ_s , over D grid-cells, each of dimension $\Delta\mathbf{x}$, and denote the result of the integral over the d^{th} cell l_{sd} . We then factorise the matrix $\mathbf{L} \in \mathbb{R}_+^{S \times D}$, with entries l_{sd} , using NMF (Equations 4.7 and 4.8) as $\mathbf{L} \sim \mathcal{P}(\mathbf{AB})$, where $\mathbf{A} \in \mathbb{R}_+^{S \times T}$ is the so-called “activation” matrix, and $\mathbf{B} \in \mathbb{R}_+^{T \times D}$ is the “template” matrix. This two stage procedure—similar to that used by Miller, Bornn, et al. (2014)—results in a predictive rate function that is a positive weighted sum of piece-wise constant functions.

We can construct the test log-likelihood for the held-out datasets as:

$$\log p(\mathcal{H}_{1:S} | \mathcal{D}_{1:S}, \Sigma_{1:S}^*) = \sum_{s=1}^S \sum_{n=1}^{\tilde{N}_h} \log \sum_{t=1}^T a_{s,t} b_{t,m(h,n)} - |\Delta\mathbf{x}| \sum_{s=1}^S \sum_{t=1}^T \sum_{b=1}^B a_{s,t} b_{t,b} \quad (7.52)$$

where $m(h, n)$ is a function that maps a test data point $\tilde{\mathbf{x}}^{(h,n)}$ into the d^{th} grid-cell. For the evaluation of predictive performance, two distinct use cases were identified. In the first, Common Rate (CR), case the held out data has the same intensity as the training data; in the second, Common Topic (CT), case the same latent functions, $f_{1:T}^2$, are present, but are mixed in different proportions. The former case, the same learned output length scales, Γ , were reused for test, and the latter Γ was re-optimised, holding the remaining parameters fixed. For the CT case the weight matrix $\mathbf{A}$ is also optimised for the test data.

7.2.5.2 Twitter data

This first application is a simple one-dimensional time series from Twitter. The times at which 68 randomly selected politicians tweeted during an 84-hour time

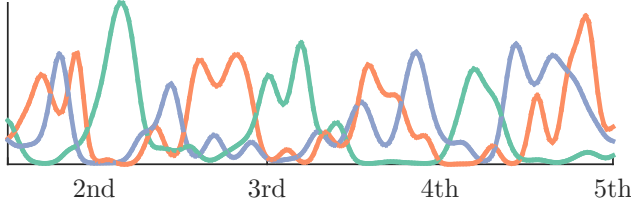


Figure 7.3: Bases found using LPPA and Twitter data.

LPPA ($\mathcal{L}_p$)	KS+NMF	IND. KS
-163.98	-379.77	-393.82

Table 7.2: Twitter data held out predictive log-likelihoods.

window from midday on 1st June 2015 through to midnight on 5th June 2015 were downloaded. Of the 68 politicians, 19 were Australian, 24 were British and 25 were from the US. Ten test/train splits of almost equal size were then created by dividing each stream in half, with odd tweets randomly assigned. Each split contained ~ 672 tweets and each politician tweeted between 3 and 61 times (in each split). Ten restarts with random initialisations were run for both LPPA and the benchmark. The run with the best training likelihood after up to 1000 iterations of gradient decent was used to compute the predictive likelihood. This was repeated for each split and the result averaged.

As shown in Figure 7.3, the three base functions inferred are periodic and it is clear that they do not fully overlap, the phase lag can be attributable to the 8-12 hour time difference between each of the three time-zones. The predictive log-likelihood performance, Table 7.2, compares well with the competing method.

7.2.5.3 Basketball data

In the next experiment we return to the problem of learning the scoring intensity of professional NBA Basketball players. Due to the positivity constraint of the latent functions, LPPA is able to learn a sparse latent factors corresponding to the key positions that influence where players are likely to score from, as shown in Figure 7.4. As described in Section 7.2.5.1, two use cases were considered: 1) the CR case, corresponds to the prediction of a held out set of shot data from players used to train the model and 2) the CT case, corresponds the construction of rate processes for players not previously used to train the model using new output length

scales, learnt for each new player independently. As before test and train splits were created and random restarts used. Inducing points were fixed on an evenly spaced grid of varying fidelity, while the benchmark used a fixed 25×30 grid.

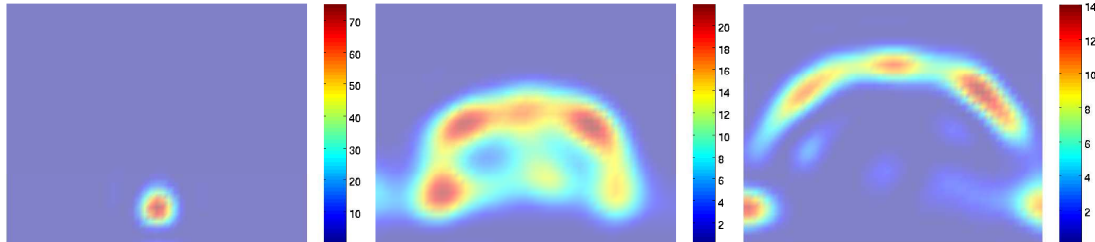


Figure 7.4: Bases found using LPPA and a basketball dataset with 20 players. Note the sparsity of the three factors, and the fact that they strongly conform to three ‘modes’ of shot typically attempted by players. From left to right: the slam-dunk, the two-pointer shot, and finally the three-point boundary line and corners.

Experimental parameters					LPPA($\mathcal{L}_p$)		KS+NMF		IND. KS
Kron	T	S	N_s	M	CR(1)	CT(2)	CR(1)	CT(2)	CR(1)
No	3	40	25	13×17	-3599.2	-3577.7	-3808.0	-4063.0	-4079.8
Yes*	3	40	25	13×17	-3701.4	-3602.8	”	”	”
No	4*	40	25	13×17	-3648.3	-3610.9	-3787.0	-3947.5	”
No	3	20*	25	13×17	-1866.6	-1832.6	-1998.1	-1806.5	-2075.0
No	3	40	50*	13×17	-5647.8	-5657.4	-6084.3	-6678.4	-6434.9
No	3	40	25	$25 \times 30^*$	-3503.9	-3523.3	-3808.0	-4063.0	-4079.8

Table 7.3: Basketball data held out predictive log-likelihoods.

7.2.5.4 Wild bird data

In this experiment the wild-bird dataset previously investigated by Ioannis et al. (2012) was used. The dataset contains the times tagged wild birds arrive at a number of Radio Frequency IDentification (RFID) equipped bird feeders distributed across Wytham Great Wood, near Oxford. The dataset contains hundreds of birds and hundreds of thousands of arrivals at dozens of locations shown Figure 7.5. A subset of the data containing 14,742 arrivals by 274 birds at 37 locations over a 7 day period was selected.

Although the kernel reflects the feeders' geographic proximity, their location identifiers are discrete. The latent coordinate space therefore contains both temporal (continuous) and feeder ID (discrete) components. The presence of this discrete dimension means that one of the integrals used to compute Equation 6.33 becomes a finite sum. The Ψ -function in the mixed coordinate space case is $\Psi(z_r, z'_r) = \sum_{x_r} K_r(z_r, x_r) K(x_r, z_r)$. When Ψ_{zz} is Kronecker structured, $\Psi_{z_2 z_2}$ becomes $\mathbf{K}_{z_2 z_2} \mathbf{K}_{z_2 z_2}$ if $\mathcal{Z}$ contains all feeding station locations and the discrete dimension is $r = 2$.

Factorisation reveals likely communities of birds since each community has a distinct arrival intensity for each feeder. The six continuous-discrete latent functions found using this analysis are shown in Figure 7.6. It was also found that most birds attached strongly to a single community, suggesting that much of the community structure has been captured.

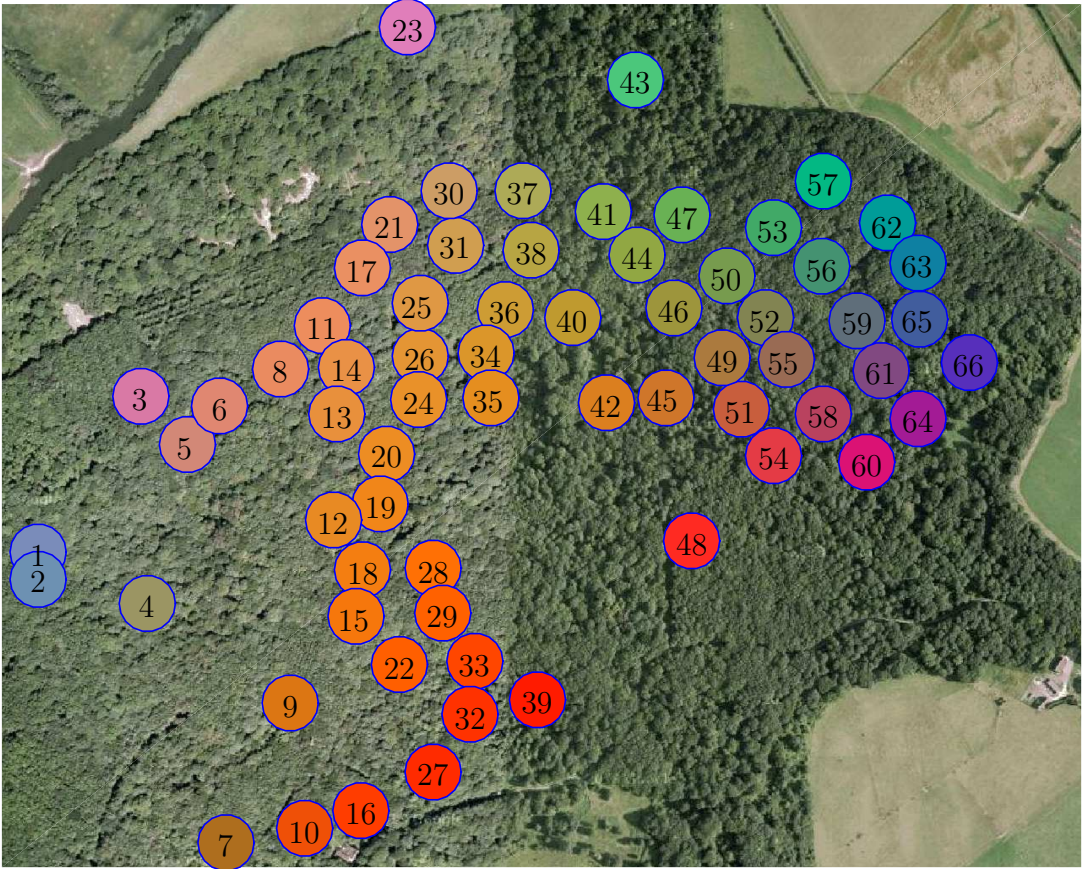


Figure 7.5: Relative locations of feeding stations in Wytham Great Wood.

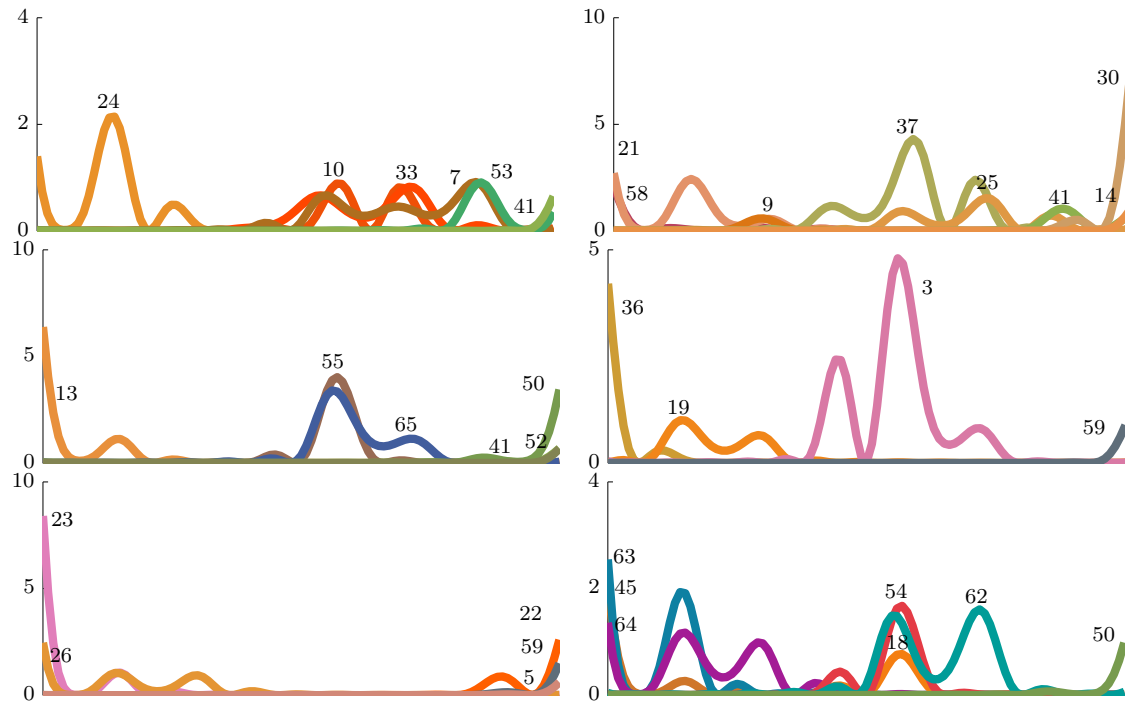


Figure 7.6: Bases found using LPPA and wild bird data. Each sub-plot represents the spatio-temporal distribution of putative communities of birds. The numbers and colour coding correspond those in Figure 7.5.

7.2.5.5 Enron data

The ENRON email dataset is a collection of emails between senior executives in the ENRON Corporation during the years leading up to its bankruptcy, amidst allegations of accountancy fraud, in what proved to be the largest case of audit failure in corporate history. Due to data protection laws, it is very rare that real email or social network data is made public and therefore the ENRON dataset has been the subject of much research by both social scientists and machine learning researchers with an interest in networks, for example Fu et al. (2009).

The email dataset contains 62208 pairwise interactions over roughly a three and half year period beginning in November 1998 and finishing in June 2002. From this a five-week period beginning at midnight on Sunday 3rd December 2000 and ending at 23 : 59 : 59 on Saturday 30th December 1999 was selected. There were 1984 emails within this period between all 149 email addresses in the database, excluding emails in which the sender and receiver was the same. (Emails to multiple

recipients were treated individually). A two-dimensional latent coordinate space with $\mathbf{x} = [\text{time}, \text{week}]$, where $\text{time} \in [1, 7)$ and $\text{week} \in \{1 \dots 5\}$ was used. There were $7 \times 4 + 1 = 29$ inducing points in the time dimension and 5 for the discrete week dimension (i.e. one per week). Both intra and inter community interactions in three latent communities were modelled.

Results of this analysis are shown in Figure 7.7. The graphs of the latent rate functions show the effect of the smooth Gaussian process prior. It is also clear how, for $C = 3$, the individuals separated into three distinct communities. It is noticeable that the first community sends emails to communities 2 and 3, but receives relatively very little back. Within communities 2 and 3 there is communication most of the time, however there are only 4 distinct days in which community 2 transmits to community 3 and only one occasion in which the reverse is true. Thus it appears that the temporal information was useful in bringing these communities into sharp relief.

A possible explanation for this may be explained by the typical behaviour of an organisational environment in which a small number of persons responsible for corporate communications will typically send a large number emails, but receive few replies.

7.3 Poisson Process Latent Variable Model

In the previous section, we saw how we can use covariate information, such as the location of bird feeders, to generate a family of correlated rate functions. In the case of the bird feeders, this covariate information was observed. Fernandez et al. (2016) used a similar construction for modelling survival data. However, by treating the covariate as missing data, we can create a new model for structured Poisson processes by reintroducing the covariate information as hidden latent variables. The result is a model we call the Poisson Process Latent Variable Model (PPLVM). As described by Pfingsten et al. (2006), latent variables also allow GP models to

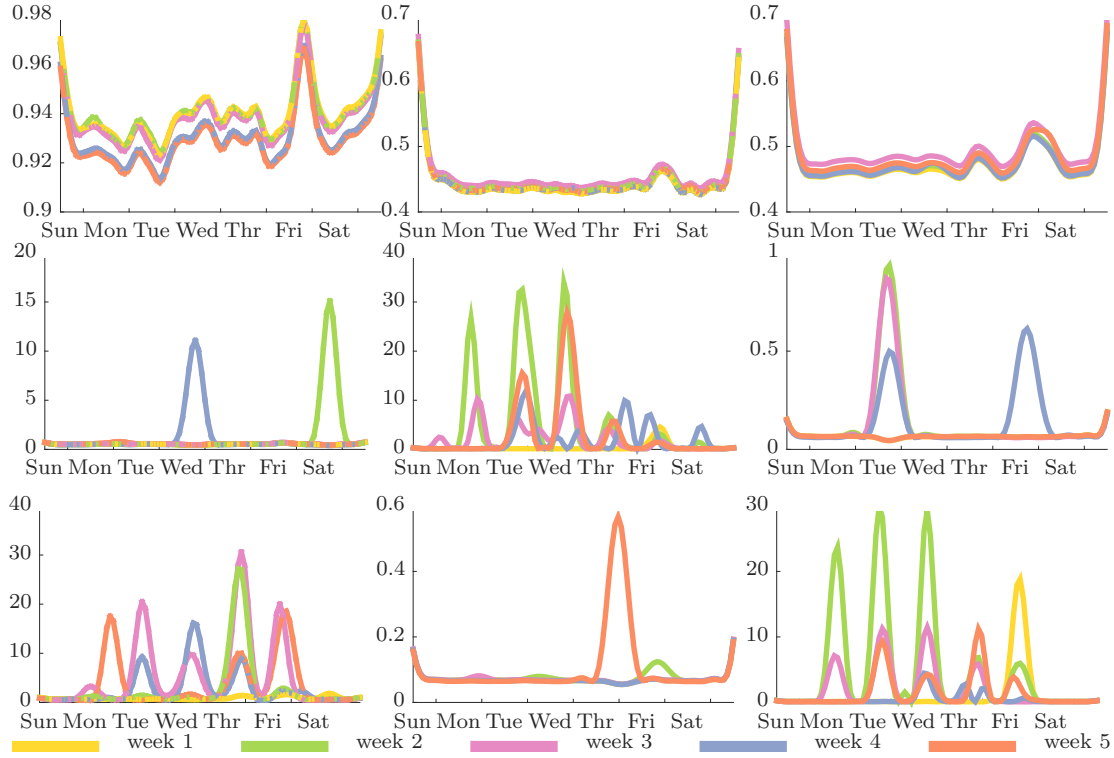


Figure 7.7: Bases found using N-LPPA and a subset of the Enron email dataset. Each sub-plot represents the weekly rate of email communication between putative communities of executives. The rows indicates the sending community and the columns the receiving community, thus the middle right graph shows the periods during which community 2 sends information to community 3.

put greater distance between data points that are close together in terms of the observed inputs, thereby “tearing apart” the input coordinate space and so the resulting GP has more flexibility to capture sudden changes in intensity.

To develop the PPLVM, we begin by assuming that each observed data point is associated with its own² set of covariate variables and therefore, in effect, its own rate function. The likelihood for a set of observed points is therefore:

$$p(\mathcal{D}|\lambda_1, \mathcal{V}) = \prod_{n=1}^N \exp\left(-\int \lambda_1(\mathbf{x}; \mathbf{v}^{(n)}) d\mathbf{x}\right) \lambda_1(\mathbf{x}^{(n)}; \mathbf{v}^{(n)}). \quad (7.53)$$

We can switch between likelihood of the single point rate function, $\lambda_1 = \frac{1}{N}\lambda$, and

² We will relax this assumption later.

Equation 7.54 since:

$$p(\mathcal{D}|\lambda) = \exp\left(-\int \lambda(\mathbf{x}) d\mathbf{x}\right) \prod_{n=1}^N \lambda(\mathbf{x}^{(n)}) \quad (7.54)$$

$$= \prod_{n=1}^N \exp\left(-\int \lambda(\mathbf{x}) d\mathbf{x}\right)^{\frac{1}{N}} \lambda(\mathbf{x}^{(n)}) \quad (7.55)$$

$$= N^N \prod_{n=1}^N \exp\left(-\int \lambda_1(\mathbf{x}) d\mathbf{x}\right) \lambda_1(\mathbf{x}^{(n)}), \quad (7.56)$$

and therefore these likelihoods differ only by a constant factor and so learning λ_1 and λ are essentially equivalent.

To construct the parametrised rate functions, $\lambda_1(\mathbf{x}; \mathbf{v})$, we use a squared GP $f^2(\mathbf{w})$ defined over the *joint* input space $\mathbf{w} = [\mathbf{x}, \mathbf{v}] \in \mathcal{W} = \mathcal{X} \times \mathcal{V}$ where $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^{R_x}$ and where $\mathbf{v} \in \mathcal{V} = \mathbb{R}^{R_v}$ are the latent coordinates. Conditioning on the set of latent variables, $\mathcal{V} = \{\mathbf{v}^{(n)}\}_{n=1}^N$, the likelihood is

$$p(\mathcal{D}|f, \mathcal{V}) = \prod_{n=1}^N \exp\left(-\int f^2([\mathbf{x}, \mathbf{v}^{(n)}]) d\mathbf{x}\right) f^2([\mathbf{x}^{(n)}, \mathbf{v}^{(n)}]). \quad (7.57)$$

As before, we condition on a set of inducing variables, $\mathbf{u}$, and inducing points, $\mathcal{Z} = \{\mathbf{z}^{(m)} \in \mathcal{W}\}_{m=1}^M$, and so $f|\mathbf{u}$ is a GP with mean and covariance functions:

$$\mu(\mathbf{w}) = \mathbf{k}_{wz} \mathbf{K}_{zz}^{-1} \mathbf{u}, \quad (7.58)$$

$$\Sigma(\mathbf{w}, \mathbf{w}') = K([\mathbf{x}, \mathbf{v}], [\mathbf{x}', \mathbf{v}']) - \mathbf{k}_{wz} \mathbf{K}_{zz}^{-1} \mathbf{k}_{zw'}, \quad (7.59)$$

We marginalise the latent variables—in addition to the inducing variables and f —using a variational distribution $q(\mathcal{V}) = \prod_{n=1}^N \mathcal{N}(\mathbf{v}^{(n)}; \mathbf{c}^{(n)}, \mathbf{L}^{(n)})$ with diagonal covariance matrix $\mathbf{L}^{(n)}$, giving the ELBO:

$$\log p(\mathcal{D}|\Omega) = \log \left[\iiint p(\mathcal{D}|f, \mathcal{V}) p(f|\mathbf{u}) p(\mathbf{u}) p(\mathcal{V}) \frac{q(\mathbf{u}) q(\mathcal{V})}{q(\mathbf{u}) q(\mathcal{V})} d\mathbf{u} d\mathcal{V} df \right] \quad (7.60)$$

$$\geq \mathbb{E}_{q(f)q(\mathcal{V})} [\log p(\mathcal{D}|f, \mathcal{V})] - \text{KL}(q(\mathbf{u})||p(\mathbf{u})) - \text{KL}(q(\mathcal{V})||p(\mathcal{V})) \quad (7.61)$$

$$\triangleq \mathcal{L}_{\text{PPLVM}},$$

where $\text{KL}(q(\mathcal{V})||p(\mathcal{V}))$ is the KL-divergence with the prior, N standard multivariate normal distributions $p(\mathbf{v}^{(n)}) = \mathcal{N}(\mathbf{v}^{(n)}; 0, \mathbb{I}_{R_v})$:

$$\text{KL}(q(\mathcal{V})||p(\mathcal{V})) = \frac{1}{2} \sum_{n=1}^N \left[\text{tr}(\mathbf{L}^{(n)}) + \mathbf{c}^{(n)\top} \mathbf{c}^{(n)} - R_v - \log |\mathbf{L}^{(n)}| \right]. \quad (7.62)$$

Expanding the expectation of log likelihood gives:

$$\begin{aligned} & \mathbb{E}_{q(f)q(\mathcal{V})} [\log p(\mathcal{D}|f_n, \mathcal{V})] \\ &= \mathbb{E}_{q(f)q(\mathcal{V})} \sum_{n=1}^N \left[- \int_{\mathcal{X}} f_n^2 d\mathbf{x} + \log[f_n^{(n)}]^2 \right] \end{aligned} \quad (7.63)$$

$$= \sum_{n=1}^N \mathbb{E}_{q(\mathbf{v}^{(n)})} \left[- \int_{\mathcal{X}} \left\{ \mathbb{E}_{q(f_n)}[f_n]^2 + \text{Var}_{q(f_n)}[f_n] \right\} d\mathbf{x} + \mathbb{E}_{q(f_n)} [\log[f_n^{(n)}]^2] \right], \quad (7.64)$$

where we use the following definitions:

$$q(f_n) = q(f([\mathbf{x}, \mathbf{v}^{(n)}])) = \mathcal{GP}(f_n; \tilde{\mu}_n, \tilde{\Sigma}_n), \quad (7.65)$$

$$f_n(\mathbf{x}) \triangleq f([\mathbf{x}, \mathbf{v}^{(n)}]) \quad (7.66)$$

$$\tilde{\mu}_n(\mathbf{x}) \triangleq \mathbf{k}_{wz}^{(n)} \mathbf{K}_{zz}^{-1} \mathbf{m} \quad (7.67)$$

$$\tilde{\Sigma}_n(\mathbf{x}, \mathbf{x}') \triangleq K([\mathbf{x}, \mathbf{v}^{(n)}], [\mathbf{x}', \mathbf{v}^{(n)}]) - \mathbf{k}_{wz}^{(n)} \mathbf{K}_{zz}^{-1} \mathbf{k}_{zw'}^{(n)} + \mathbf{k}_{wz}^{(n)} \mathbf{K}_{zz}^{-1} \mathbf{S} \mathbf{K}_{zz}^{-1} \mathbf{k}_{zw'}^{(n)}, \quad (7.68)$$

$$f_n^{(n)} \triangleq f_n(\mathbf{x}^{(n)}) \quad (7.69)$$

$$\tilde{\mu}_n^{(n)} \triangleq \tilde{\mu}_n(\mathbf{x}^{(n)}) \quad (7.70)$$

$$[\tilde{\sigma}_n^{(n)}]^2 \triangleq \tilde{\Sigma}_n(\mathbf{x}^{(n)}, \mathbf{x}^{(n)}). \quad (7.71)$$

After taking expectations of the data log-likelihood under $q(f_n)$ and $q(\mathbf{v}^{(n)})$ we have:

$$\mathbb{E}_{q(f_n)}[f_n]^2 = \mathbf{m}^\top \mathbf{K}_{zz}^{-1} \mathbf{k}_{zw}^{(n)} \mathbf{k}_{zw}^{(n)} \mathbf{K}_{zz}^{-1} \mathbf{m}, \quad (7.72)$$

$$\text{Var}_{q(f_n)}[f_n] = \gamma - \text{tr}(\mathbf{K}_{zz}^{-1} \mathbf{k}_{zw}^{(n)} \mathbf{k}_{zw}^{(n)}) + \text{tr}(\mathbf{K}_{zz}^{-1} \mathbf{S} \mathbf{K}_{zz}^{-1} \mathbf{k}_{zw}^{(n)} \mathbf{k}_{zw}^{(n)}). \quad (7.73)$$

If our kernel is separable, we can both integrate over the region, $\mathcal{X}$, and take the expectations under $q(\mathbf{v}^{(n)})$, since only $\mathbf{k}_{zw}^{(n)} = \mathbf{k}_{wz}^{(n)\top}$ are functions of $\mathbf{x}$ and $\mathbf{v}^{(n)}$ and therefore

$$\int_{\mathcal{V}} \int_{\mathcal{X}} \mathbb{E}_{q(f_n)}[f_n]^2 d\mathbf{x} q(\mathbf{v}^{(n)}) d\mathbf{v}^{(n)} = \mathbf{m}^\top \mathbf{K}_{zz}^{-1} \mathbf{\Psi}_{zz}^{(n)} \mathbf{K}_{zz}^{-1} \mathbf{m}, \quad (7.74)$$

$$\int_{\mathcal{V}} \int_{\mathcal{X}} \text{Var}_{q(f_n)}[f_n] d\mathbf{x} q(\mathbf{v}^{(n)}) d\mathbf{v}^{(n)} = \gamma |\mathcal{X}| - \text{tr}(\mathbf{K}_{zz}^{-1} \mathbf{\Psi}_{zz}^{(n)}) + \text{tr}(\mathbf{K}_{zz}^{-1} \mathbf{S} \mathbf{K}_{zz}^{-1} \mathbf{\Psi}_{zz}^{(n)}). \quad (7.75)$$

The matrices $\mathbf{\Psi}_{zz}^{(n)}$ are Kronecker structured matrices whose elements are defined by the function $\Psi^{(n)}(\mathbf{z}, \mathbf{z}')$ which is:

$$\Psi^{(n)}(\mathbf{z}, \mathbf{z}') = \int_{\mathcal{V}} \int_{\mathcal{X}} K(\mathbf{z}, [\mathbf{x}, \mathbf{v}^{(n)}]) K([\mathbf{x}, \mathbf{v}^{(n)}], \mathbf{z}') d\mathbf{x} q(\mathbf{v}^{(n)}) d\mathbf{v}^{(n)} \quad (7.76)$$

$$= \prod_{r=1}^{R_x} \Psi_{x_r}(z_r, z'_r) \cdot \prod_{r=1}^{R_v} \Psi_{v_r}^{(n)}(z_{r+R_x}, z'_{r+R_x}). \quad (7.77)$$

Note that the common factor in $\Psi_{zz}^{(n)}$ for all data points is the integral over domain, Ψ_{x_r} , given by

$$\begin{aligned} \Psi_{x_r}(z, z') &= -\frac{\sqrt{\pi\alpha_r}}{2} \exp\left(-\frac{(z - z')^2}{4\alpha_r}\right) \\ &\quad \times \left[\operatorname{erf}\left(\frac{\bar{z} - \mathcal{X}_r^{\max}}{\sqrt{\alpha_r}}\right) - \operatorname{erf}\left(\frac{\bar{z} - \mathcal{X}_r^{\min}}{\sqrt{\alpha_r}}\right) \right], \end{aligned} \quad (7.78)$$

where $\bar{z} = \frac{1}{2}(z + z')$, which is itself a factor of Equation 6.35.

For the exponentiated quadratic Automatic Relevance Determination (ARD) kernel we can compute $\Psi_{v_r}^{(n)}$ in closed-form as follows:

$$\begin{aligned} \Psi_{v_r}^{(n)}(z, z') &= \int_{-\infty}^{\infty} \exp\left(-\frac{(z - z')^2}{4\alpha_s} - \frac{(v_r^{(n)} - \bar{z})^2}{\alpha_s}\right) \\ &\quad \times \frac{1}{\sqrt{2\pi L_r^{(n)}}} \exp\left(-\frac{(v_r^{(n)} - c_r^{(n)})^2}{2 L_r^{(n)}}\right) dv_r^{(n)} \end{aligned} \quad (7.79)$$

$$= \sqrt{\frac{\alpha_s}{2L_r^{(n)} + \alpha_s}} \exp\left(-\frac{(z - z')^2}{4\alpha_s}\right) \exp\left(-\frac{(c_r^{(n)} - \bar{z})^2}{\alpha_s + 2L_r^{(n)}}\right), \quad (7.80)$$

where $s = r + R_x$. The function $\Psi_{v_r}^{(n)}$ is precisely the same as the $\Psi_2^{(n)}$ statistic used in the GPLVM (Titsias and Lawrence 2010). To take the final expectation we sample J points, $\mathbf{v}_j^{(n)} \sim q(\mathbf{v}^{(n)})$, around the latent coordinates from the variational distribution and compute the expectation under the resulting function $f_{n,j}$ at the corresponding data point $\mathbf{x}^{(n)}$ as follows:

$$\mathbb{E}_{q(f_n)q(\mathbf{v}^{(n)})}[\log[f_n^{(n)}]^2] = \sum_{j=1}^J \phi_j \int_{-\infty}^{\infty} \log([f_{n,j}^{(n)}]^2) \mathcal{N}(f_{n,j}^{(n)}, \mu_{n,j}^{(n)}, [\sigma_{n,j}^{(n)}]^2) df_{n,j}^{(n)} \quad (7.81)$$

$$= \sum_{j=1}^J \phi_j \left[-\tilde{G}\left(-\frac{[\mu_{n,j}^{(n)}]^2}{2[\sigma_{n,j}^{(n)}]^2}\right) + \log\left(\frac{[\sigma_{n,j}^{(n)}]^2}{2}\right) \right] - C, \quad (7.82)$$

where $\{\phi_j\}_{j=1}^J$ are a set of sample weights. We can generate samples from $q(\mathbf{v}^{(n)})$ using Monte-Carlo or from a deterministic sample scheme such as the Unscented Transform (Julier. and Uhlmann 1997) or other schemes such as nested Gauss-Hermite quadrature. Deterministic sampling is compatible with deterministic optimisation algorithms such as L-BFGS, while random sampling may be preferable for stochastic optimisation (§7.3.2) since repeated random sampling will be more accurate on average; this will also prevent pathological behaviour that may result from interaction with the inducing point locations.

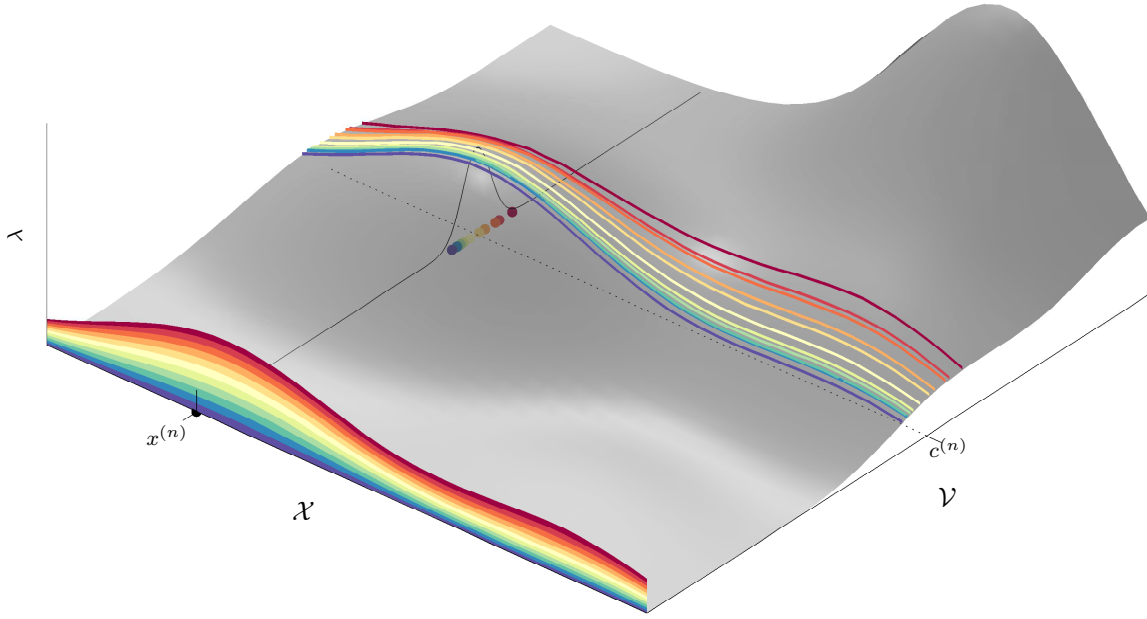


Figure 7.8: Uncertainty in the location of the latent coordinates means that in the intensity function at any point, e.g. $x^{(n)}$, is an infinite mixture of functions that lie on the manifold. To evaluate the lower bound at the data point $x^{(n)}$ we can sample the variational distribution $q(v^{(n)})$ and sum the resulting functions. In this illustration, $J = 11$ latent variables are sampled and the corresponding functions weighted and summed; the result is shown in the $v = 0$ plane.

7.3.1 Tied latent variables

So far we have assumed that each data point has its own latent variable. This was convenient when developing the ELBO Equation 7.61. However, this does not need to be the case, since multiple different data points can *share* latent variables or components of latent variables. By tying latent coordinates together, we can control which data points share the same intensity functions. For example, if we have S distinct data sources we wish to model, and a mapping function $h : \{1, \dots, N\} \rightarrow \{1, \dots, S\}$ that returns the data source to which the n^{th} data point belongs, then we can set

$$\mathbf{v}^{(n)} := \tilde{\mathbf{v}}^{(h(n))} \in \tilde{\mathcal{V}}, \quad (7.83)$$

where $\tilde{\mathcal{V}} = \{\tilde{\mathbf{v}}^{(s)}\}_{s=1}^S$ is the set of latent coordinates associated with each output. In this case we would only have S variational distributions and so the KL-divergence

term given by Equation 7.62 would become the sum of S Gaussian KL-divergences and the parameters of the variational distribution associated with $\mathbf{v}^{(n)}$ would be $\mathbf{c}^{(h(n))}$ and $\mathbf{L}^{(h(n))}$.

More generally, we can compose the latent coordinates using multiple mappings for each data point. For example, if we wish to model a network structure we can have a set of latent variables associated sending nodes, $\{\tilde{v}_1^{(s)}\}_{s=1}^{S_1}$, and a set of variables associated with receiving nodes $\{\tilde{v}_2^{(s)}\}_{s=1}^{S_2}$. We can then set construct $\mathbf{v}^{(n)}$ as

$$\mathbf{v}^{(n)} \triangleq \begin{bmatrix} v_1^{(n)} \\ v_2^{(n)} \end{bmatrix} := \begin{bmatrix} \tilde{v}_1^{(h_1(n))} \\ \tilde{v}_2^{(h_2(n))} \end{bmatrix}, \quad (7.84)$$

where $h_1 : \{1, \dots, N\} \rightarrow \{1, \dots, S_1\}$ maps to the sending node and h_2 maps to the receiving node. This modelling structure allows us to assign numerical values to categorical labels. These numerical values determine the degree of correlation between the intensity functions associated with each label. Where each data point is associated with multiple labels, as in the network case, each label type becomes a separate latent dimension.

There may be observable combinations of tied latent variables for which we have no events in the training set, but which *might* appear in a test dataset. We should include in the likelihood, the integral of the function for those combinations we wish to predict³. This suppresses the intensity function in those parts of the augmented coordinate space where we have no observed data. It is not necessary to constrain the intensity function in this way for observable combinations we do not wish to predict.

7.3.2 Stochastic inference

A further advantage of working with the single point intensity likelihood, Equation 7.37, is that it factorises over data points. This is more convenient for stochastic inference as we shall now see. We subdivide the dataset into a set of *batches* and

³ If the batch stochastic method described in §7.3.2 is used, then either the integral for these combinations should be divided by E , the number of batches in an epoch, or $1/E$ of these integrals should be randomly divided amongst each batch in the epoch.

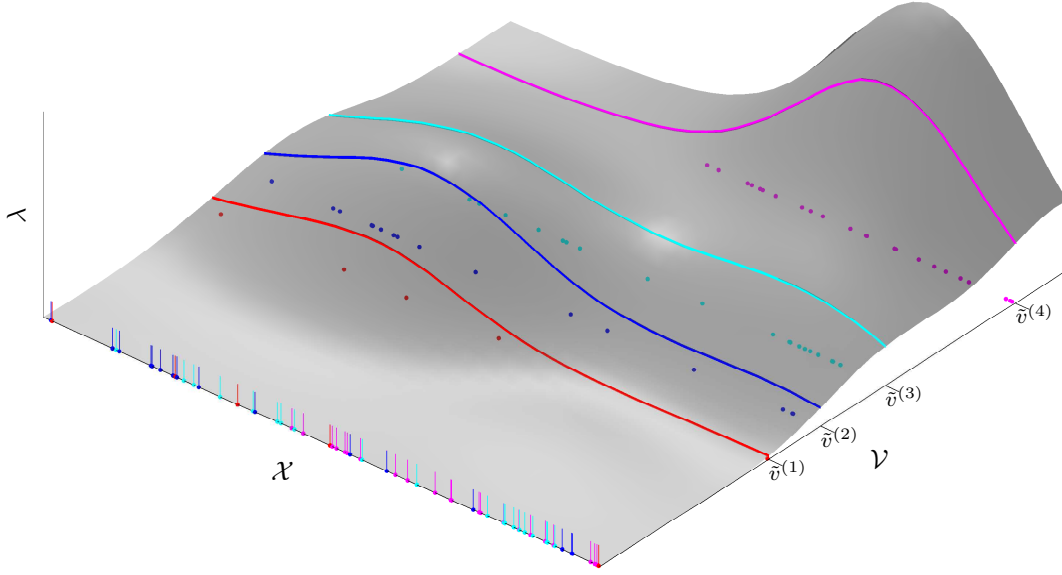


Figure 7.9: In the PPLVM model, multiple intensity functions lie on the same manifold. Data points from the same source can be tied to the same latent coordinate.

denote these batches $\mathcal{B}_1, \dots, \mathcal{B}_E$, where $\mathcal{B}_i \subset \mathcal{D}$ and $\mathcal{B}_i \cap \mathcal{B}_j = \emptyset, \forall i, j < E$. The batch size, B , is a user defined value that trades gradient noise with computational cost. Since in general B will not be divisor of N , we allow the final batch, $\mathcal{B}_E$, to contain $EB - N$ randomly re-sampled points. Each step of Stochastic Gradient Descent (SGD) uses a successive batch. A sweep through the whole dataset is called an *epoch* and E is the number of batches per epoch. At the end of the epoch we randomly reassign data points to new batches.

Since the ELBO $\mathcal{L}_{\text{PPLVM}}$ is predicated on the whole dataset, and not a subset, to apply stochastic inference we must make some minor alterations that affect the weightings of various terms in the bound. In the stochastic variational inference algorithm described in 3.2.2, this was achieved by multiplying the sufficient statistics computed using a single point by the dataset size, N , before computing the stochastic gradient. Here we take a slightly different approach in that we re-weight the terms in the bound and then differentiate. To identify the appropriate weightings, we first break up the ELBO into three separate terms $\mathcal{L}_{\text{PPLVM}} \triangleq \mathcal{L}^{[1]} + \mathcal{L}^{[2]} + \mathcal{L}^{[3]}$ that arise due to interaction with the likelihood and with the prior distributions, the latter two

being the KL-divergence terms. Specifically these terms are:

$$\mathcal{L}^{[1]} \triangleq \mathbb{E}_{q(f)q(\mathcal{V})} [\log p(\mathcal{D}|f, \mathcal{V})] \quad (7.85)$$

$$\mathcal{L}^{[2]} \triangleq -\text{KL}(q(\mathbf{u})||p(\mathbf{u})) \quad (7.86)$$

$$\mathcal{L}^{[3]} \triangleq -\text{KL}(q(\mathcal{V})||p(\mathcal{V})). \quad (7.87)$$

Since the likelihood provides the “energy” needed to overcome the prior, if each batch contains only $1/E$ of the dataset, the prior is effectively E times stronger. We can therefore either down weight the prior or up weight the likelihood; in this case we take the former approach. The batch lower bound therefore becomes

$$\mathcal{L}_{\text{PPLVM}}(\mathcal{B}_i) = \mathcal{L}^{[1]}(\mathcal{B}_i) + \frac{B}{N} \mathcal{L}^{[2]} - \sum_{r=1}^{R_w} \sum_{s=1}^{S_r} \frac{a_{r,s,i}}{b_{r,s}} \text{KL}(q(\tilde{v}_r^{(s)})||p(\tilde{v}_r^{(s)})), \quad (7.88)$$

where

$$a_{r,s,i} \triangleq \sum_{\mathbf{x}^{(n)} \in \mathcal{B}_i} \mathbb{1}\{h_r(n) = s\}, \quad (7.89)$$

$$b_{r,s} \triangleq \sum_{n=1}^N \mathbb{1}\{h_r(n) = s\}, \quad (7.90)$$

and so $\frac{a_{r,s,i}}{b_{r,s}}$ is the fraction of the data points in the i^{th} batch that are tied to s^{th} latent coordinate in the r^{th} latent coordinate dimension. Provided $EB = N$ (i.e. no re-sampled data points) then we have $\sum_{i=1}^E a_{r,s,i} = b_{r,s}$ and moreover $\mathcal{L}_{\text{PPLVM}}(\mathcal{D}) = \sum_{i=1}^E \mathcal{L}_{\text{PPLVM}}(\mathcal{B}_i)$. Thus taking a step of length 1 in $\mathcal{L}_{\text{PPLVM}}(\mathcal{B}_i)$ is an approximation of a step of length $1/E$ in $\mathcal{L}_{\text{PPLVM}}(\mathcal{D})$. If we wish to make the learning rate independent of the batch size, we can multiply the gradient by E .

7.3.3 Back-constrained latent variables

In stochastic gradient descent, a latent variable can only be updated when its associated data point appears in the current batch. If the ratio of latent variables to data points is high (i.e. S is close to N), most of the latent variables in a random batch of data will not be updated and so they will be slow to converge. We might attempt to mitigate this by using a higher learning rate for the latent variables compared to the global parameters. An alternative is to use a *back-constrained*

variational distribution (Bui and Turner 2015), also known as a *recognition model* in the context of Variational Auto Encoders (VAE) (Kingma and Welling 2013). A back-constrained variational distribution uses the data and a set of parameters to model the variational distribution. This is shown graphically in Figure 7.10. This dependency is permitted since the variational distribution is a free choice. Importantly the recognition model parameters are now shared by *all* variational distributions and are thus *global* variational parameters. Since these global parameters are updated at every stochastic update step, we *amortise* the cost of learning all the distinct variational parameters. However, the cost is a less flexible variational distribution. In the context of the PPLVM we may wish to make the distribution

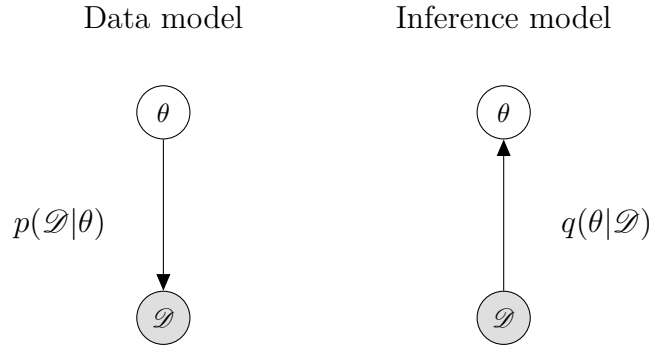


Figure 7.10: A back-constrained inference model. The variational model has a probabilistic dependency on the data.

$q(\mathcal{V})$ a function of the data. One method to construct such a function is to use a GP with a set of control inputs and control values. This is shown in Figure 7.11. The figure shows how each data point, $\mathbf{x}^{(n)}$, can be functionally mapped to its own latent variable $v^{(n)}$. Instead of using a GP, we could alternatively use a fully connected neural-network like a VAE model (Kingma and Welling 2013; Rezende et al. 2014). An example of such a formulation is given by Equations 7.91-7.94:

$$\mathbf{h}_1^{(n)} := \xi_1(\mathbf{W}_1 \mathbf{x}^{(n)} + \mathbf{b}_1) \quad (7.91)$$

$$\mathbf{h}_2^{(n)} := \xi_2(\mathbf{W}_2 \mathbf{h}_1^{(n)} + \mathbf{b}_2) \quad (7.92)$$

$$c^{(n)} := \xi_3(\mathbf{W}_3 \mathbf{h}_2^{(n)} + \mathbf{b}_3) \quad (7.93)$$

$$L^{(n)} := \xi_4(\mathbf{W}_4 \mathbf{h}_2^{(n)} + \mathbf{b}_4). \quad (7.94)$$

Here, we parametrise $c^{(n)}$ and $L^{(n)}$ using a three layer neural network, where $\xi_{1:4}$ are non-linear functions, and $\mathbf{h}_1^{(n)}$ and $\mathbf{h}_2^{(n)}$ are latent representations of the n^{th} data point in the first and second hidden layers. The weights and biases, $\mathbf{W}_{1:4}$ and $\mathbf{b}_{1:4}$, are the global variational parameters.

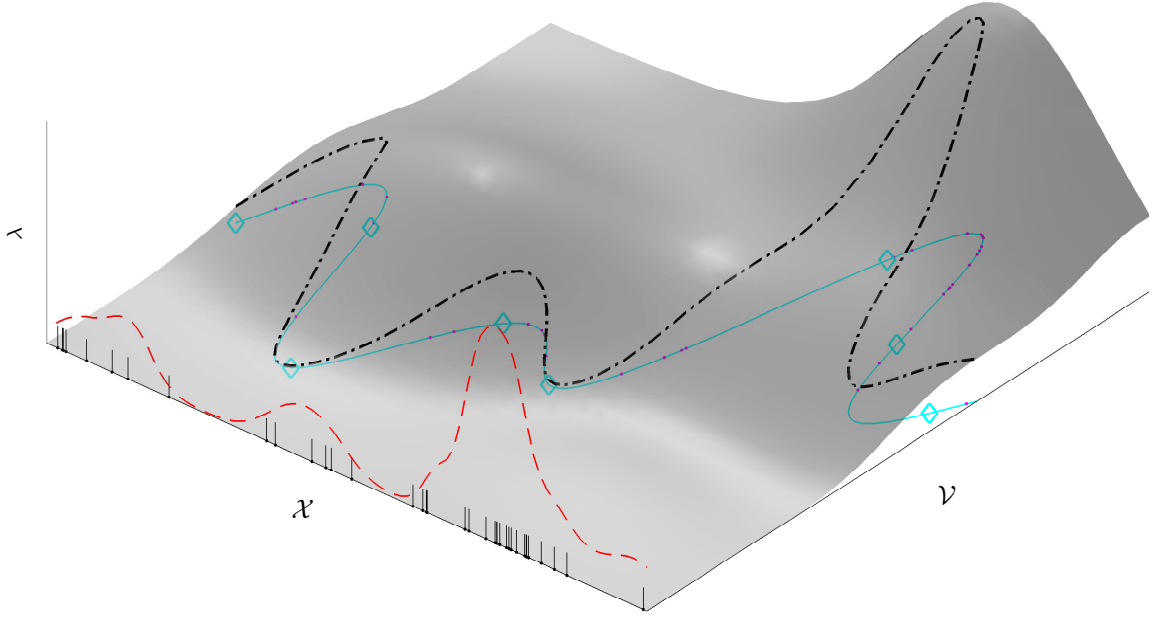


Figure 7.11: Back constrained latent variables in the PPLVM model. The correlation of the intensity function surface is controlled by the distance in both the observed input space, $\mathcal{X}$, and the distance in the latent space, $\mathcal{V}$. The latent variables are back constrained to sit on the solid cyan line, which is itself a function of $\mathbf{x}$. The latent variables for the data points are determined by latent function shown by the solid cyan line, which is parametrised by the control points shown by the diamonds. The intensity function for the points is the projection of this function onto the surface, shown by the black dash-dotted line, and the marginal rate function is shown by the dashed red line.

7.3.4 Initialisation scheme for network data

In Section 7.3.1, we saw how data points can be tied to multiple latent variables and that network models can be constructed by tying each data point to latent variables associated with the sending and receiving nodes. The result of these multiple ties is a complicated set of interactions amongst the latent variables. If the latent variables are not well initialised however, they may have to move through many

low probability configurations and the optimisation is likely to become trapped in a poor local minima. In order to make the best use of the PPLVM's explanatory power, we therefore desire a sensible initialisation scheme that will give our optimisation procedure a good chance of finding a useful configuration of the latent variables.

To begin, we will assume our covariance function is a separable exponentiated quadratic, with a kernel input length scale α_s in all latent dimensions associated with the sending nodes and α_r in all dimensions associated with receiving nodes. For a unipartite graph with V nodes, the covariance matrix between all V^2 links has V^4 entries. Excluding the covariance matrix entries between reflexive links and themselves—which are equal to the kernel output length scale—and the covariance between reflexive links and other links, this matrix contains L^2 distinct entries, where $L = \frac{V^2-V}{2}$. We can form a $[L \times L]$ matrix, Υ , of these distinct values. Υ , for sending nodes $i < m$, receiving nodes $j > n$ and all link pairs $i \rightarrow j$ & $m \rightarrow n$ is:

$$\gamma \begin{bmatrix} e^{-\frac{\alpha_s}{2}|\mathbf{v}_s^{(1)}-\mathbf{v}_s^{(2)}|^2} & \dots & e^{-\frac{\alpha_s}{2}|\mathbf{v}_s^{(1)}-\mathbf{v}_s^{(2)}|^2} & \dots & e^{-\frac{\alpha_s}{2}|\mathbf{v}_s^{(1)}-\mathbf{v}_s^{(2)}|^2} \\ \times e^{-\frac{\alpha_s}{2}|\mathbf{v}_s^{(1)}-\mathbf{v}_s^{(2)}|^2} & & \times e^{-\frac{\alpha_r}{2}|\mathbf{v}_r^{(n)}-\mathbf{v}_r^{(j)}|^2} & & \times e^{-\frac{\alpha_r}{2}|\mathbf{v}_r^{(N-1)}-\mathbf{v}_r^{(N)}|^2} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ e^{-\frac{\alpha_s}{2}|\mathbf{v}_s^{(i)}-\mathbf{v}_s^{(m)}|^2} & \dots & e^{-\frac{\alpha_s}{2}|\mathbf{v}_s^{(i)}-\mathbf{v}_s^{(m)}|^2} & \dots & e^{-\frac{\alpha_s}{2}|\mathbf{v}_s^{(i)}-\mathbf{v}_s^{(m)}|^2} \\ \times e^{-\frac{\alpha_r}{2}|\mathbf{v}_r^{(1)}-\mathbf{v}_r^{(2)}|^2} & & \times e^{-\frac{\alpha_r}{2}|\mathbf{v}_r^{(n)}-\mathbf{v}_r^{(j)}|^2} & & \times e^{-\frac{\alpha_r}{2}|\mathbf{v}_r^{(N-1)}-\mathbf{v}_r^{(N)}|^2} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ e^{-\frac{\alpha_r}{2}|\mathbf{v}_r^{(1)}-\mathbf{v}_r^{(2)}|^2} & \dots & e^{-\frac{\alpha_r}{2}|\mathbf{v}_r^{(n)}-\mathbf{v}_r^{(j)}|^2} & \dots & e^{-\frac{\alpha_r}{2}|\mathbf{v}_r^{(N-1)}-\mathbf{v}_r^{(N)}|^2} \\ \times e^{-\frac{\alpha_s}{2}|\mathbf{v}_s^{(N-1)}-\mathbf{v}_s^{(N)}|^2} & & \times e^{-\frac{\alpha_s}{2}|\mathbf{v}_s^{(N-1)}-\mathbf{v}_s^{(N)}|^2} & & \times e^{-\frac{\alpha_s}{2}|\mathbf{v}_s^{(N-1)}-\mathbf{v}_s^{(N)}|^2} \end{bmatrix},$$

where γ is the kernel output length scale and $\mathbf{v}_s^{(\cdot)}$ and $\mathbf{v}_r^{(\cdot)}$ are the latent coordinates associated with sending and receiving nodes. If the network is symmetric (with $\mathbf{v}_r^{(n)} = \mathbf{v}_s^{(n)}$ and $\alpha_s = \alpha_r$) then Υ is a symmetric matrix. More importantly, Υ has rank-1 structure and so $\Upsilon = \mathbf{v}_s \mathbf{v}_r^\top$. Now, the vector elements of the L length vector

$\mathbf{d}_s = \sqrt{\frac{\log(\mathbf{v}_s) - \frac{1}{2} \log(\gamma)}{-\frac{\alpha_s}{2}}}$ can be rearranged in a symmetric matrix, $\mathbf{D}_s$, that has the

following form:

$$\begin{bmatrix} 0 & \dots & |\mathbf{v}_s^{(1)} - \mathbf{v}_s^{(i)}| & \dots & |\mathbf{v}_s^{(1)} - \mathbf{v}_s^{(m)}| & \dots & |\mathbf{v}_s^{(1)} - \mathbf{v}_s^{(N)}| \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ |\mathbf{v}_s^{(1)} - \mathbf{v}_s^{(i)}| & \dots & 0 & \dots & |\mathbf{v}_s^{(i)} - \mathbf{v}_s^{(m)}| & \dots & |\mathbf{v}_s^{(i)} - \mathbf{v}_s^{(N)}| \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ |\mathbf{v}_s^{(1)} - \mathbf{v}_s^{(m)}| & \dots & |\mathbf{v}_s^{(i)} - \mathbf{v}_s^{(m)}| & \dots & 0 & \dots & |\mathbf{v}_s^{(m)} - \mathbf{v}_s^{(N)}| \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ |\mathbf{v}_s^{(1)} - \mathbf{v}_s^{(N)}| & \dots & |\mathbf{v}_s^{(i)} - \mathbf{v}_s^{(N)}| & \dots & |\mathbf{v}_s^{(m)} - \mathbf{v}_s^{(N)}| & \dots & 0 \end{bmatrix},$$

which is a $[V \times V]$ matrix of distances between the latent coordinates. These insights lead to the following initialisation algorithm for the latent coordinates:

1. Using a similarity measure, compare the intensity functions for each pair of links and using these measures, construct an estimation $\hat{\mathbf{\Upsilon}}$, of $\mathbf{\Upsilon}$.
2. Apply the singular valued decomposition to obtain the rank-1 factors $\hat{\mathbf{v}}_s$ and $\hat{\mathbf{v}}_r$ (for symmetric networks the latter are equal).
3. Construct the matrices $\mathbf{D}_s$ and $\mathbf{D}_r$ by taking logarithms and adding a suitable constant—this is equivalent to choosing a value of γ .
4. Apply multidimensional scaling to recover an embedding. If $\mathbf{v}_s$ ($\mathbf{v}_r$) is R_s (R_r) dimensional, recover an R_s (R_r) dimensional embedding. Scale the results to cover the space spanned by the inducing points—this is equivalent to choosing a value of α_s (α_r).
5. Use the scaled embeddings to initialise the means of the variation parameters.

When the similarity measure is the kernel function, this algorithm exactly recovers the latent coordinates up to translation and scaling.

The cost of computing the similarity matrix $\mathbf{\Upsilon}$ is $\mathcal{O}(V^4)$ and the cost of computing its largest Eigenvalue is $\mathcal{O}(V^4)$ or $\mathcal{O}(V^6)$ if naïve singular-valued decomposition is used. Therefore, this method is limited to relatively small networks. To apply the method to larger networks, we can cluster nodes statically using—for example, Infinite Relational Model (IRM) described in Appendix C—and compute the similarity

matrix for the clusters rather than the individual nodes. We compute normalised histograms for each link and compare histograms using:

$$\hat{\Upsilon}(i \rightarrow j, m \rightarrow n) = 1 - \frac{1}{2} \sum_{t=1}^T |h_{i,j,t} - h_{m,n,t}|, \quad (7.95)$$

where $h_{i,j,t}$ is the normalised count in the t^{th} bin and $\sum_t h_{i,j,t} = 1$. If the histograms overlap completely the measure is 1, if there is no overlap in the histograms, the measure is 0.

7.3.5 Demonstration

In this section we will see the results of applying the PPLVM model to three test datasets. The datasets include the now familiar basketball dataset, in addition to two new datasets. To allow inference to scale to the larger datasets used in these experiments, the batch stochastic gradient descent algorithm described in §7.3.2, in combination with the “Adam” stochastic optimisation method of Kingma and Ba (2015), was used. The alternate variance parametrisation of Equation 7.9 was also investigated.

7.3.5.1 Basketball data

For this experiment, 10545 scoring locations for 163 players were extracted. Each batch contained 528 data points and there were 20 batches per epoch. A regular 17×17 grid in the spatial domain was used, along with another regular grid with 13 points at evenly spaced intervals between -6 and 6 in the latent variable dimension. The stochastic bound was optimised for 500 epochs. Figure 7.14 shows the progression of the stochastic bound during learning on a dual-core 3.30 GHz Intel Xeon E5-2667v2 (16 cores, 32 threads).

Figure 7.12 shows slices through the latent surface. At one extreme, the shot surface is completely concentrated around the hoop. At the other extreme, there is greater intensity around the 3 point line, although scoring locations close to the hoop are still prominent. The players shot intensities are infinite mixtures of slices through

this three-dimensional surface, with the mixture weights controlled by the posterior distribution of the latent variable locations, shown in Figure 7.13. It can be seen that the players sit on a continuum between close-in scoring patterns and more balanced, all-court, scoring styles.

7.3.5.2 School contact data

The school contact dataset was created to investigate the spread of disease transmitted by physical contact (Gemmetto et al. 2014). It records instances of physical interactions between 236 primary school children in 10 classes and their 10 teachers over a period of two school days (of roughly 8 hours 45 minutes each). Interactions are recorded with 20 second accuracy without direction information. Interactions mostly occur mainly between children in the same class—except during break time—and so the network adjacency matrix is predominately block structured. For the demonstration 3632 interactions that occur between children in class 1A on the first day were extracted. The missing 20 seconds of data precision was sampled uniformly. There were 5 pairs of children that had no interactions between them.

Since this network is symmetric, we could process interactions $i \rightarrow j$ for $i < j$ only. If we take this approach, we should symmetrise the kernel along the lines of Equation 4.27 and this would allow us to construct a model in which each node (child) has only a single set of latent variables that characterise both “sending” and “receiving” behaviour, i.e. $\mathbf{v}_r^{(n)} = \mathbf{v}_s^{(n)}, \forall n$. This symmetrisation allows us to halve the number of data points and latent variables, however it requires a sum kernel that is not separable; this compromises Kronecker structure. However, the computational advantages of exploiting Kronecker structure far outweigh the benefits of exploiting network symmetry. Therefore, instead of enforcing network symmetry, we should simply treat the resulting network as asymmetric, for instance, by uniformly sampling a directed, $i \rightarrow j$ or $j \rightarrow i$, interaction for each undirected interaction. We can however tie the *variational* parameters for $\mathbf{v}_r^{(n)}$ and $\mathbf{v}_s^{(n)}$ together, i.e. set $\mathbf{c}_r^{(n)} := \mathbf{c}_s^{(n)}$ and $\mathbf{L}_r^{(n)} := \mathbf{L}_s^{(n)}$. Note that this is subtly different from assuming

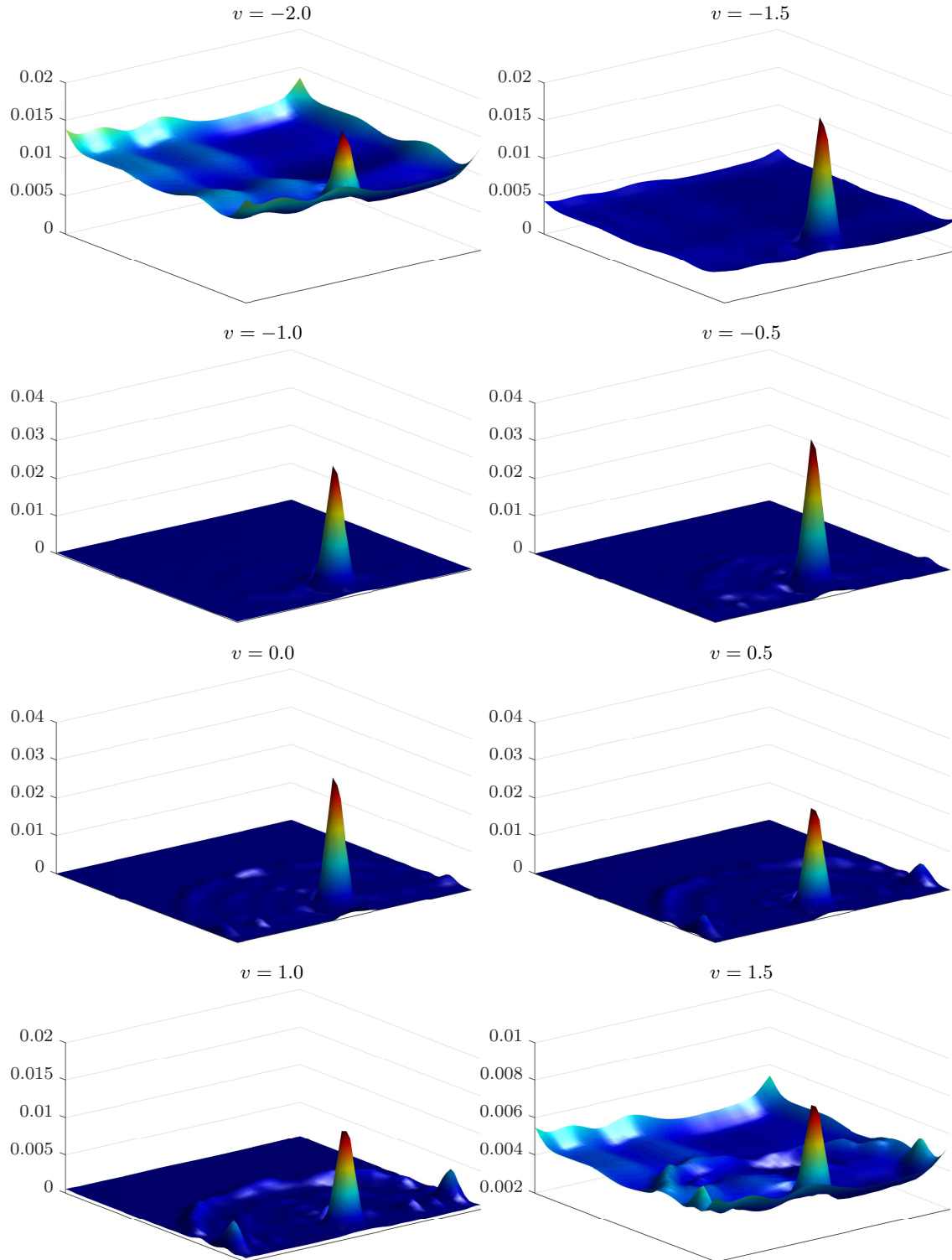


Figure 7.12: Slices through the basketball shot intensity surface for different latent variable values from $v = -2.0$ to $v = 1.5$. At $v = -1.5$ the shot intensity surface is concentrated round the hoop. As the latent variable value increases, the propensity to shoot from further from the hoop increases. At $v = -2.0$ and $v = 1.5$ the surface is reverting back toward to the uniform flat surface of the prior, however the spike around the hoop is still evident.

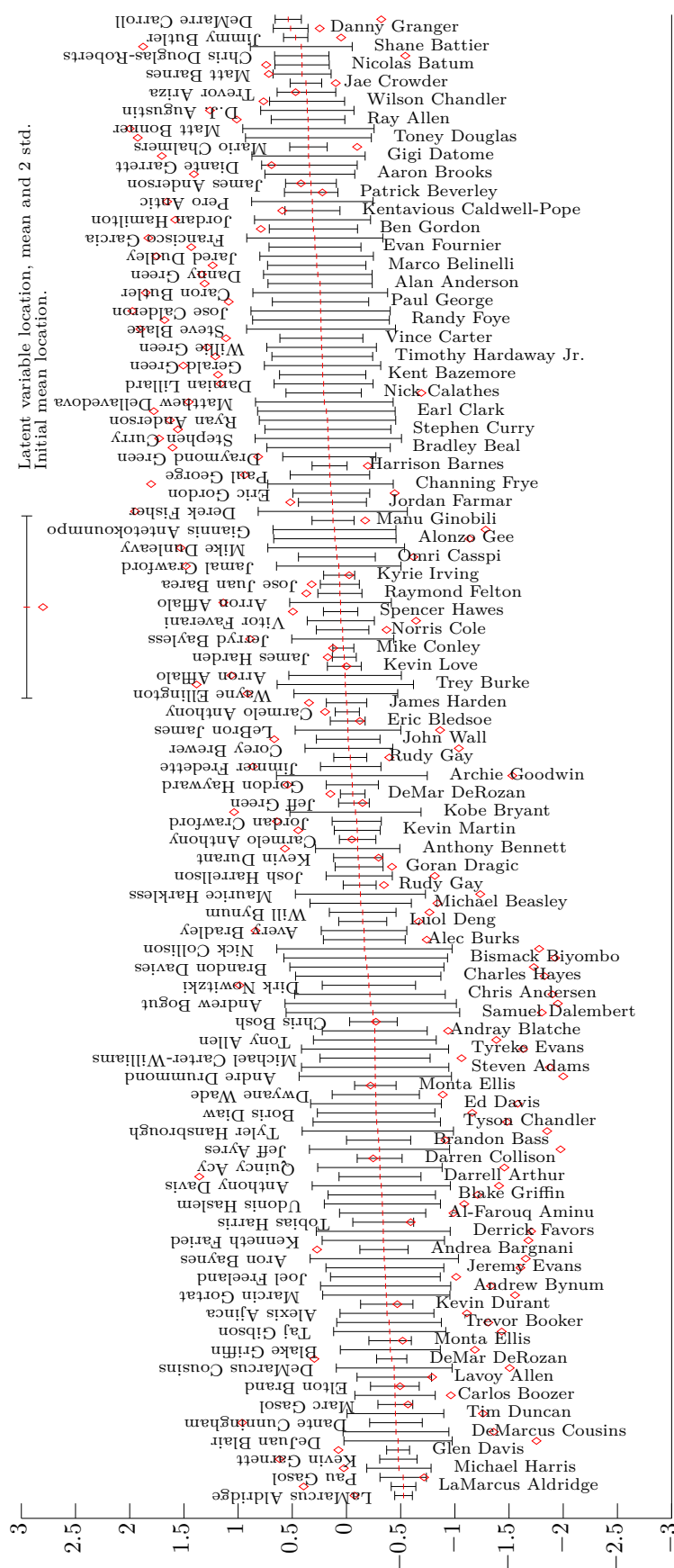


Figure 7.13: Initial and optimised locations of the latent variables for each basketball player. Means and two standard deviations are shown and the red diamonds indicate the initial location of the means.

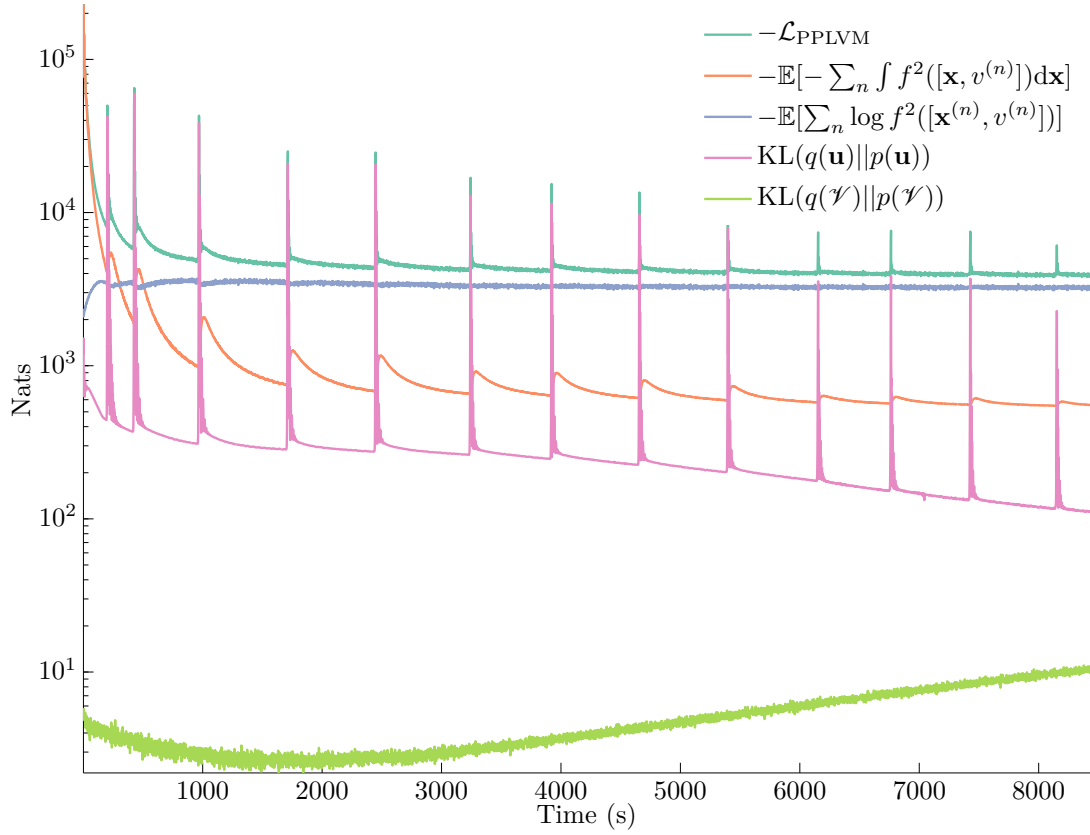


Figure 7.14: The stochastic ELBO during optimisation with basketball data showing the decomposition of the lower bound. Some random steps result in large increases in the log-determinant $\mathbf{S}$ shown by the spiking of the KL-divergence term. Note the log scale on the Y-axis.

that $\mathbf{v}_r^{(n)}$ and $\mathbf{v}_s^{(n)}$ are the *same* random variable; the former is an inference model assumption and the latter is a data model assumption.

In experiments with this dataset, learning good model parameters proved somewhat more difficult than for the basketball data. During optimisation using SGD, the $\text{KL}(q(\mathbf{u})||p(\mathbf{u}))$ term—which could be very large depending on the initial values of the variational and kernel parameters—could be minimised by allowing the latent variables to coalesce, resulting, in effect, in a one-dimensional point process with no diversity in locations of the latent variables or the corresponding rate functions. A solution to this problem of multiple local minima was proposed by Ghahramani and Hinton (2000) who used a deterministic annealing scheme to progressively up weight a KL-divergence term. In the results presented in this subsection (and the following subsection), during the first E_a epochs of SGD the $\text{KL}(q(\mathbf{u})||p(\mathbf{u}))$ term

was weighted by a factor of $\frac{i}{E_a}$ for $i \in 1 \dots E_a$ where i is the epoch number. A value of $E_a = 500$ was used for the contact network data, after which the model was trained for another 100 epochs with a weight of 1. This method allows the optimisation to first find latent locations and kernel length scales that explain the data well and only then to gradually penalise complexity.

Figure 7.16 shows slices through the 3-dimension interaction intensity surface at 1-hour intervals. The figures are very symmetrical in the line $y = x$ reflecting the network symmetry. The symmetry is not perfect however, since it is not enforced by the method described above. That interpretation of these figures is slightly complicated, as they depict the single point intensity function λ_1 that must be scaled by the number of observed points in each interacting pair. A peak in these plots should not necessarily be interpreted as corresponding to a high number of network interactions. Rather, the height of a particular point on the surface through time indicates the changing intensity of a particular event process relative to itself. This is shown more clearly in Figure 7.17 which plots the single point intensities for one network node with the remaining 21 nodes. Four pairs of interactions are highlighted and the corresponding data is plotted. Some of these pairs contain a lot more events than others, however the functions have roughly the same overall height due to the scaling.

Figure 7.18 shows the stochastic lower bound during the course of optimisation. The figure compares the bound for the sum of matrices variational covariance of Equation 7.9, with that of a single Kronecker structured matrix. The optimisation latter approach is somewhat better behaved than the former, although the sum covariance does briefly achieve a better lower bound than the single matrix covariance.

7.3.5.3 Computer network data

In this section we investigate interactions in a computer network captured by Netflow. The timestamps of the beginning of 32006 flow records were extracted along with host and port identifiers. Communications between 20 hosts on the same

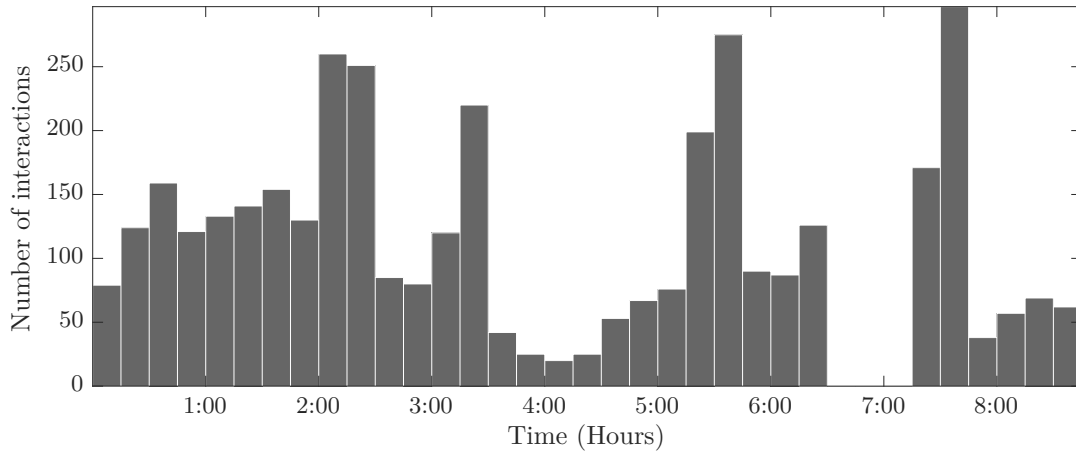


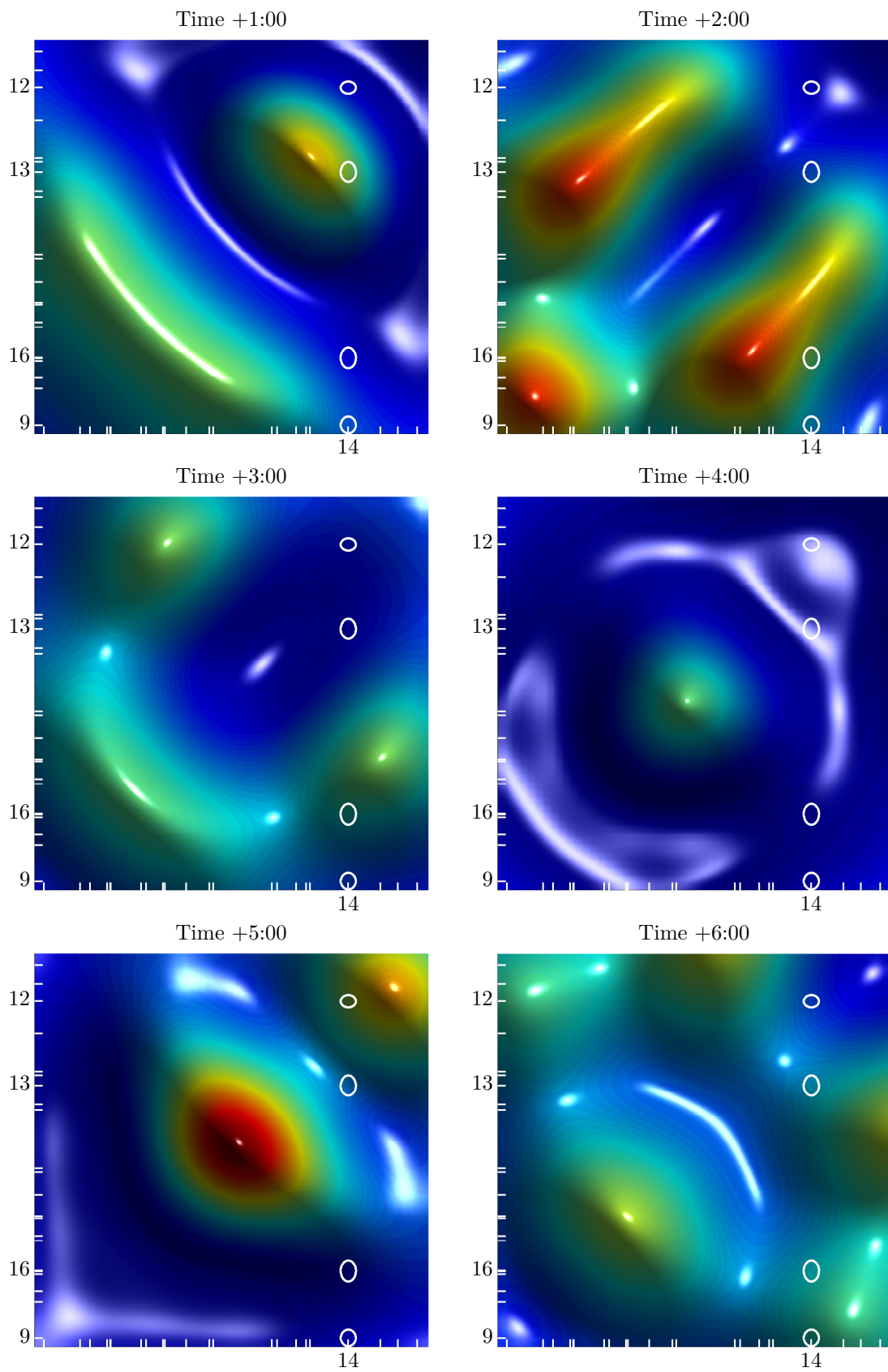
Figure 7.15: The temporal distribution of the 3632 physical interactions in the school contact dataset between the 22 children in class 1A over the course of the first day.

subnet—identified by the last 8 bits of the IP-address—on 13 low numbered port addresses over a 24 hour period were extracted. Figure 7.19 shows a visualisation of this data. As shown, the network is very sparse and most pairs of nodes (hosts) do not communicate directly. Also note that, since each Internet Protocol (IP) port is typically associated with a different internet application, and that application may be running on only a subset of hosts, even where two nodes do communicate, they may do so on only a subset of the ports.

The data was modelled as a one-dimensional point process with a three-dimensional latent variable space, with separate sender and receiver latent variables, and a third latent dimension associated with the port identifier.

Figure 7.20 shows the posterior locations of the latent variables. Note that some of the nodes sending and receiving latent variables do not move from their initial locations since empty pair combinations were excluded from the likelihood. The figure shows that the ports separated into roughly four clusters, while the receiving nodes had roughly two clusters. All the non-empty sending IP-addresses were clustered together.

Figure 7.21 shows slices through the four-dimensional surface. The surfaces are centred on the mean location of several different port latent variable locations. Communication on port 80, associated with web traffic, shows an increase in



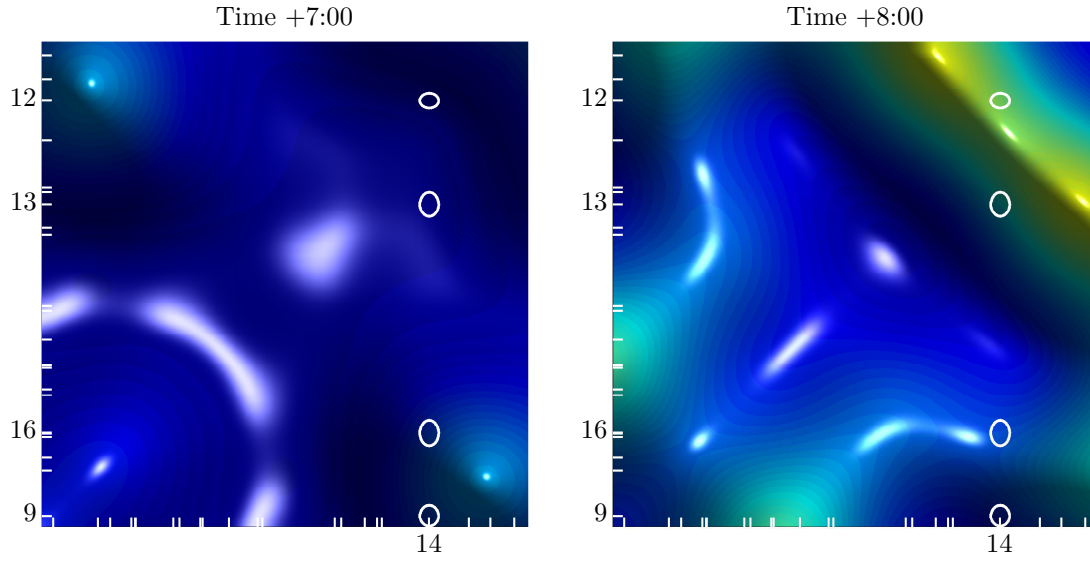


Figure 7.16: Slices through the node interaction surface for contact network data at intervals of 1 hour. The short white lines indicate the location of the posterior means of the latent variables. The circles indicate the mean and 2 standard deviation variance of the posterior locations of $\mathbf{v}_r^{(14)}$ intersecting with $\mathbf{v}_s^{(9)}$, $\mathbf{v}_s^{(12)}$, $\mathbf{v}_s^{(13)}$ and $\mathbf{v}_s^{(16)}$.

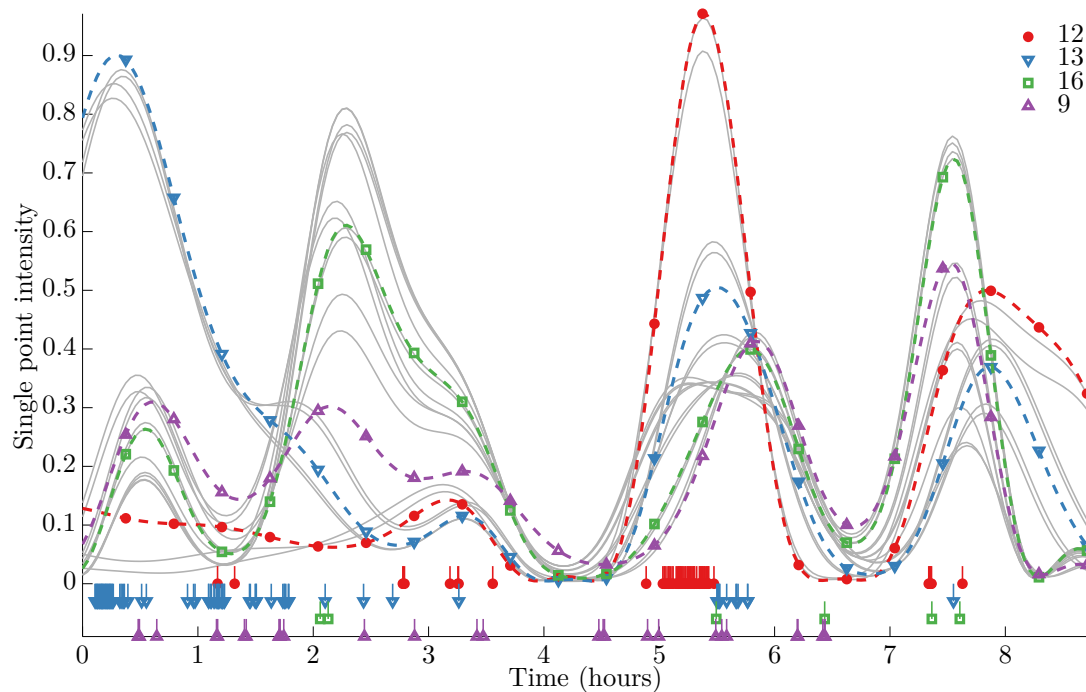


Figure 7.17: Single point intensities for node 14. The intensities for nodes 9, 12, 13 and 16 are highlighted. By placing the intensity functions on the same manifold the PPLVM is able to infer sensible intensity functions for interacting pairs that have few points, for example, between nodes 14 and 16.

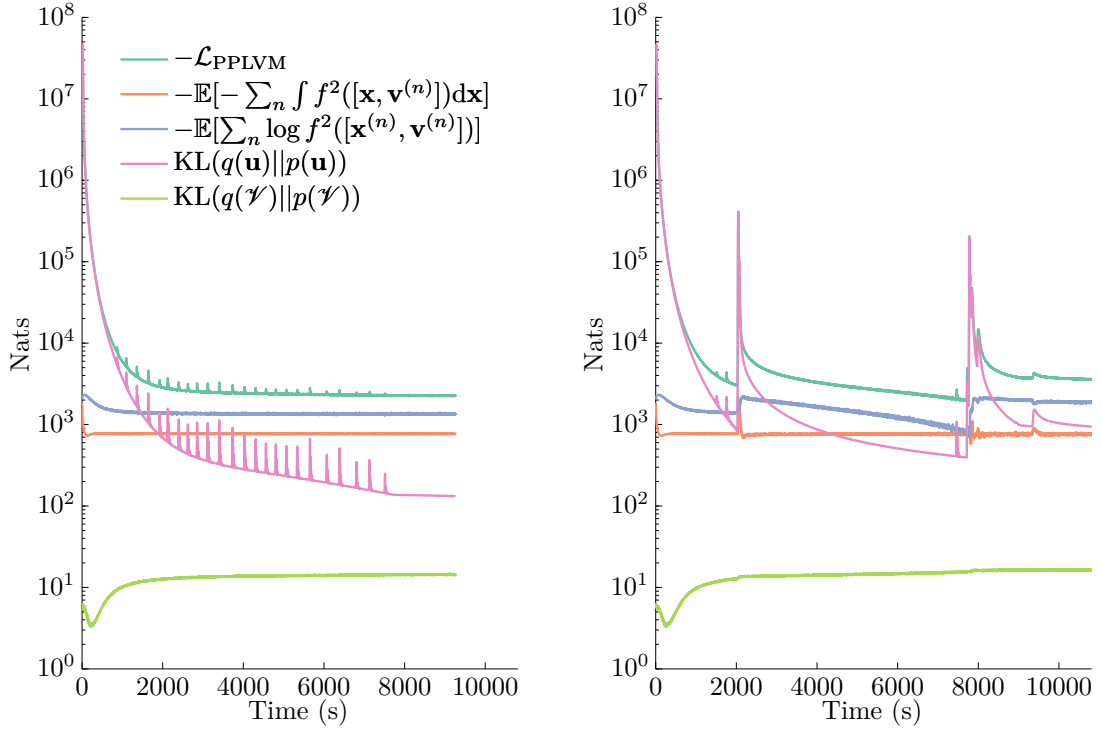


Figure 7.18: The stochastic ELBO during optimisation with contact network data. The right hand plot shows the stochastic bound for the sum of matrices covariance of Equation 7.9, while the plot on the left shows the bound using the standard single Kronecker structured matrix. The optimisation of standard approach is somewhat better behaved. The best ELBO found uses the sum covariance, although at the end of the optimisation run it is worse than that achieved using the single matrix covariance.

intensity around midday and a low intensity in the early morning. Other ports were more active throughout the day, which explains why the latent variable associated with port 80 was isolated from the others.

7.3.6 Discussion

This section demonstrated the potential of the PPLVM model. Further experiments, including an empirical study against suitable benchmarks, are needed to fully validate the approach. Another area for further work is the factored variational mean of Equation 7.17. This scales better in higher dimensions, however (latent variable model) experiments have not been very successful so far. This could be due either to the presence of local minima or the reduction of flexibility. In the case of reduced flexibility, it may be possible to increase flexibility using the Spectral

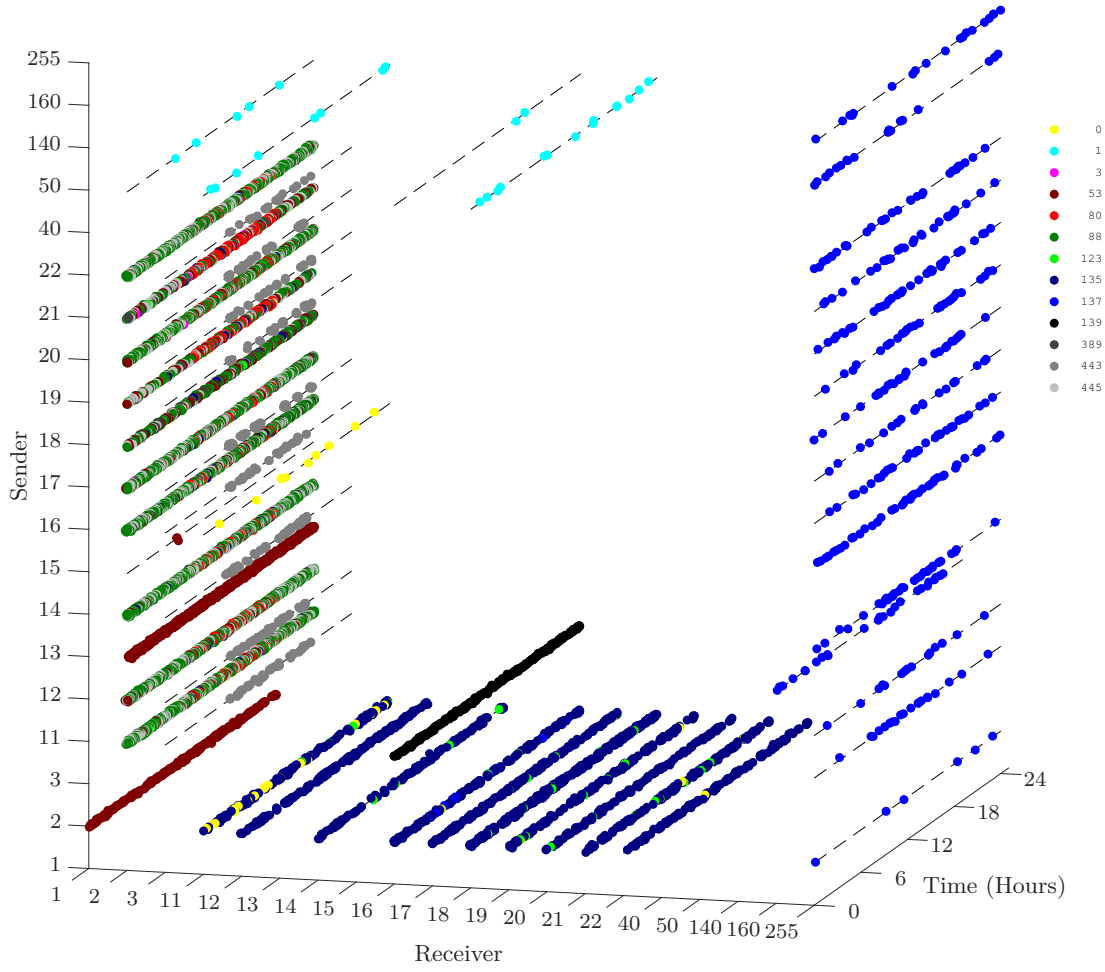


Figure 7.19: Communications between 20 hosts in a computer network over a 24 hour period. The interactions shown use 13 different IP ports. Each port is typically associated with a different internet application.

Mixture Kernel (SMK) (described in the next chapter) or by using a variational mixture distribution:

$$q(\mathbf{v}^{(n)}) = \sum_{g=1}^G \varphi_g \mathcal{N}(\mathbf{v}^{(n)}; \mathbf{c}_g^{(n)}, \mathbf{L}_g^{(n)}), \quad \sum_{g=1}^G \varphi_g = 1. \quad (7.96)$$

Another possible extension is to share the same set of latent variables between multiple conditionally independent functions given a common set of latent coordinates using the following likelihood:

$$p(\mathcal{D}|f_{1:D}, \mathcal{V}) = \prod_{d=1}^D \prod_{n=1}^{N_d} \exp \left(- \int f_d^2([\mathbf{x}, \mathbf{v}^{(h(n,d))}]) d\mathbf{x} \right) f_d^2([\mathbf{x}^{(h(n,d))}, \mathbf{v}^{(h(n,d))}]). \quad (7.97)$$

In, say, the computer network dataset, we could use the functions $f_{1:D}$ to model interactions on D ports as separate functions or alternatively model D days worth

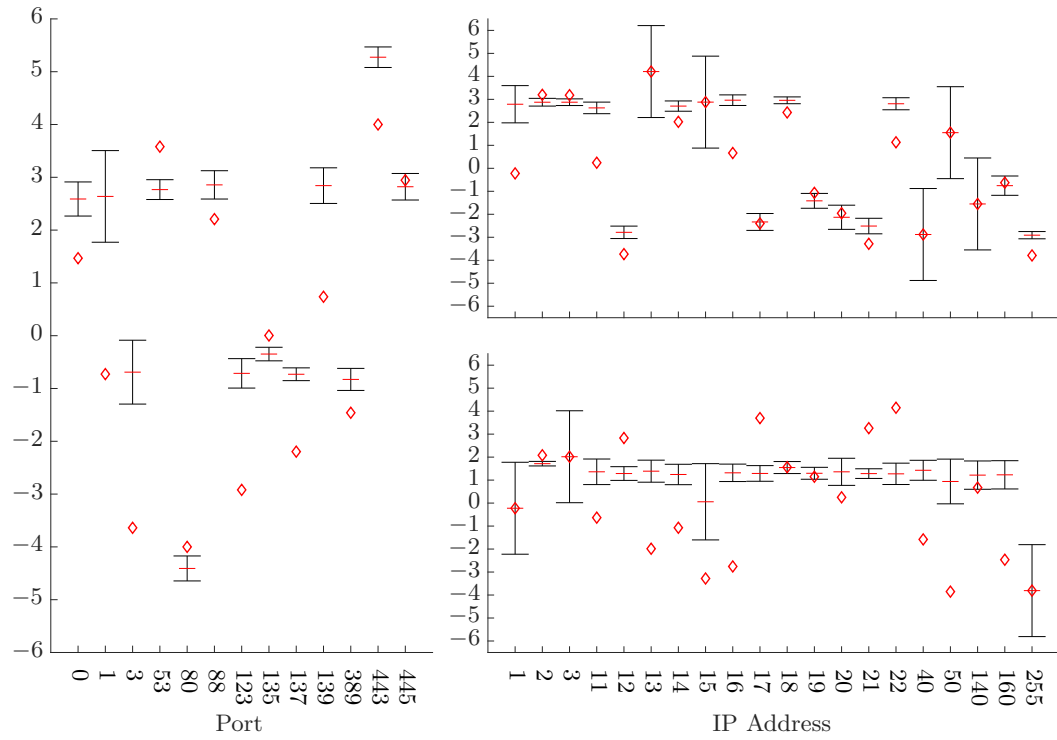


Figure 7.20: Initial and optimised locations of the latent variables for computer network data. On the left: the locations of the variables associated with the IP ports; top right: the receiving IP address; bottom right: the sending IP-address. Means and two standard deviations are shown and the initial location of the mean is indicated by the red diamond.

of data. Investigation of these ideas, along with the back-constrained latent variable model described in §7.10, is left for further work.

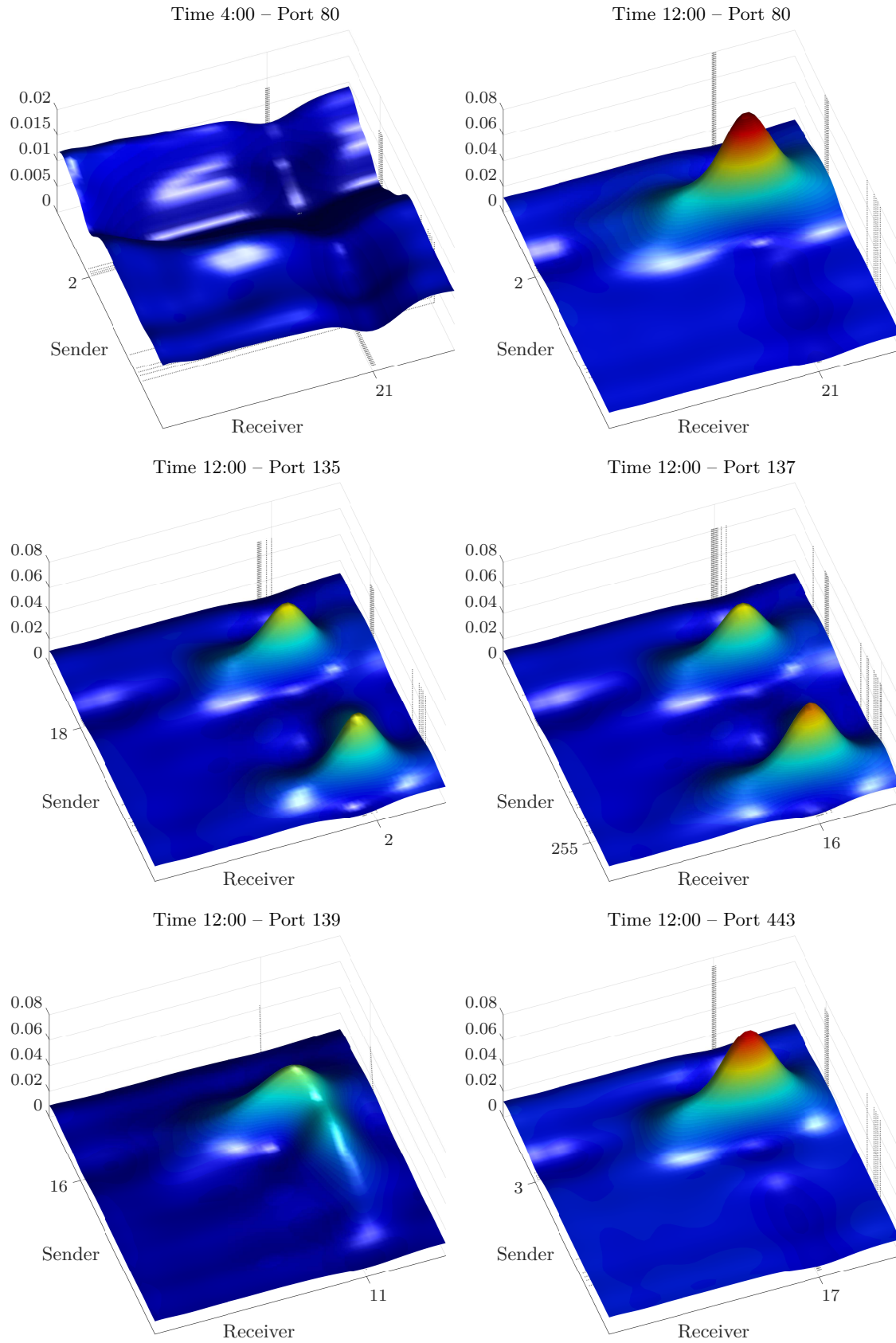


Figure 7.21: Slices through the node interaction surface for selected IP ports. The dotted lines indicate the sending and receiving latent variable locations.

8

Conclusion

This thesis was motivated by the study of correlated event processes and, in particular, Dynamic Interaction Networks (DINs), a rich source of such processes. We showed that Gaussian processes, a flexible non-parametric Bayesian prior over functions, can be used to specify the intensity functions of Cox processes. We saw how statistical dependencies between the intensity of related event processes could be captured using Gaussian processes. We developed an efficient variational inference algorithm that scales linearly in the number of data points—even in high dimensions. We showed how this algorithm can be extended to allow factor and latent variable models of dynamic networks and other complex data to be constructed. We saw that learning the hidden structure captured by these models gave both insight into the data and improved predictive performance.

8.1 Related and contemporary work

We will now take note of some related and contemporary work in point processes modelling and dynamic networks that have not already been mentioned in the text.

8.1.1 Point processes

Gaussian process modulated Poisson processes have been the subject of a number of studies. In almost all cases, the proposed methods require discretisation of the input domain. To prevent excessive information loss, such approaches—exemplified by the Log-Gaussian Cox Process (LCGP) of Moller et al. (1998)—are sensitive to the choice of discretisation and suffer from poor scaling with the dimension of the domain. In most cases, data is binned on rectangular or square grids, although Simpson et al. (2016) consider a tessellated surface. Inference algorithms for the discrete LCGP include MCMC (Moller et al. 1998) and Integrated Nested Laplace Approximation (INLA) (Rue et al. 2009; Taylor and Diggle 2014). Flaxman, Wilson, et al. (2015) presented a Laplace approximation method that exploited Kronecker structure to reduce the computational burden imposed by the large number of discretisation cells required. LCGP models were used by Girolami and Calderhead (2011) to test their Riemann Manifold MCMC algorithms and Dieng et al. (2016) to test their χ -divergence variational inference method. The LCGP was also used by Hensman, Matthews, et al. (2015) and Hensman, Durrande, et al. (2016) to test their inference schemes.

Discretisation is generally required in LCGP models since the expectation of the integral over the domain cannot be computed in closed-form¹. Kom Samo and Roberts (2015) used a sparse Gaussian approximation combined with numerical integration and a prior over the domain integral to construct a continuous model that avoided the thinning as per Adams et al. (2009), but still used MCMC.

Several methods reported in the literature used the GP prior as only a MAP regulariser, without integrating it out. One example is the method of Cunningham et al. (2008), which achieved $\mathcal{O}(n \log n)$ (n is the number of bins) performance by exploiting Toeplitz structure. However, the absence of a link function in their method meant that positivity had to be enforced somewhat inelegantly via the optimisation

¹We should note that the VBPP model did also require numerical integration, but only to compute $\tilde{G}$.

algorithm. Flaxman, Teh, et al. (2017) approximated a square transformed Gaussian process modulated Poisson point process using a finite set of basis functions based on the weight space (Mercer kernel) expansion; Walder and Bishop (2017) proposed a Bayesian variation on this theme using Laplace approximation to obtain a posterior distribution over the weights.

The LPPA model described in Chapter 7 is conceptually similar to the work of Lian et al. (2014). Both methods attempted to factor continuous Poisson processes using a positive weighted sum of latent functions. The differences are in the model used to drive the latent processes and the inference scheme, Binary Semi-Markov Jump Processes (BSMJP) and Forward Filtering and Backward Sampling (FFBS) respectively, which together would seem to limit their approach to one-dimensional data series. Saul et al. (2016) showed that a variety of likelihoods can be combined with variational Gaussian processes, one of which was a (discrete) Poisson factor model.

Bardenet and Titsias (2015) developed a variational inference algorithm for *determinantal* point processes that is related to variational GP inference. Although their development followed from a somewhat different insight, it can be considered the determinantal point processes analogue of the VBPP model for permanental point processes. Interestingly the function Ψ , which appears in the VBPP and GPLVM, also appeared in their work.

8.1.2 Dynamic networks

Ishiguro, Iwata, et al. (2010) presented a dynamic variant of the IRM in which the cluster assignments can evolve over time according to a discrete time Markov model similar to how the Mixed Membership Stochastic Block Model (MMSBM) was adapted into the Dynamic Mixed Membership Stochastic Block Model (DMMSBM). Fouldsa et al. (2011) described a comparable adaptation for the IBP; Kim and Leskovec (2013) further developed this model using the distance-dependent IBP.

Durante and Dunson (2014) developed a Bayesian logistic regression model for dynamic networks. In their model the links were modelled as $\mathbf{Y}_t \sim \text{Bern}(\sigma(\mathbf{X}_t \mathbf{X}_t^\top + \mu(t)))$ where the elements of $\mathbf{X}_t$, $x_{i,j}(t)$, were GP distributed functions. They used their algorithm to model the probability of a draw in Italian football matches based on the past 20 years of results.

Perry and Wolfe (2013) described a Poisson process model for social networks that included a variety of covariates for modelling static (e.g. homophily) and dynamic effects. They applied their model to the Enron email dataset using maximum *partial* likelihood and bootstrap to estimate the parameters.

DuBois et al. (2013) described an extension of the IRM to longitudinal networks (e.g. DIN). In contrast to Ishiguro, Iwata, et al., in their model the cluster assignments were constant and they used Poisson processes with random mixtures of deterministic user specified piecewise constant rate functions (constructed from the interaction event history statistics) to model the dynamic behaviour. They used MCMC to infer the posterior distributions of the mixing coefficients and cluster assignments. Xin et al. (2015) developed a continuous time variant of the Stochastic Block Model (SBM) for the analysis of Basketball networks in which passes between players were the interactions. Their model is similar to that of DuBois et al. (2013) except they used cubic B-splines to represent the intensity functions and they presented MCMC and Expectation Maximisation (EM) inference algorithms. In another variation on this theme, Matias et al. (2016) used kernel smoothing to model the intensity functions. Unlike Network Latent Poisson Process Allocation (N-LPPA), which is a mixed membership model, these models are single membership models where each node is a member of only one cluster or community.

Palla, Caron, et al. (2016) described a dynamic variant of sparse network model of Caron and Fox (2014) in which the model parameters could temporally adapt.

A common aspect of much work in dynamic networks is the absence of systematic comparisons between competing methods on a range of different dynamic network datasets. Future work in this direction would add greatly to the literature.

8.2 Further work

As described in the introduction to this thesis, complex event processes and dynamic networks are ubiquitous, so there are many potential datasets to which the algorithms described herein can be applied. To conclude this thesis we will focus on several potential avenues for further algorithmic development of the VBPP method described in Chapter 6. These new models can be combined in various permutations with those already described to construct a wide range of new event models suitable for numerous applications.

8.2.1 Hawkes process models

A Hawkes process is a type of *self-excitatory* Poisson point process that can be used to model social and communication networks. In a Hawkes process there are two types of points known as *offspring* and *immigrants* (Daley and Vere-Jones 2003). The preceding points produce offspring, while immigrants have no parents in the process. Immigrants arrive according to a background intensity function λ_0 . The n^{th} point in the process, $x^{(n)}$, which may be an immigrant or an offspring, creates new points with intensity $\lambda_n(x)$, which is often assumed to be an exponential distribution:

$$\lambda_n(x) = \gamma \beta \exp(-\beta(x - x^{(n)})), \quad (8.1)$$

where γ is a scaling factor that determines how many offspring each point produces on average (if $\gamma > 1$ the number of points grows exponentially fast). The intensity function of the point process at any point is therefore

$$\lambda(x) = \lambda_0(x) + \sum_{x^{(n)} \leq x} \lambda_n(x), \quad (8.2)$$

which is the sum of the intensity function of all preceding points plus intensity of immigrant arrivals. An example of a Hawkes process with the intensity function of Equation 8.1 is shown by Figure 8.1. After each new point joins the process, the intensity function spikes upward, further increasing the likelihood of new arrivals

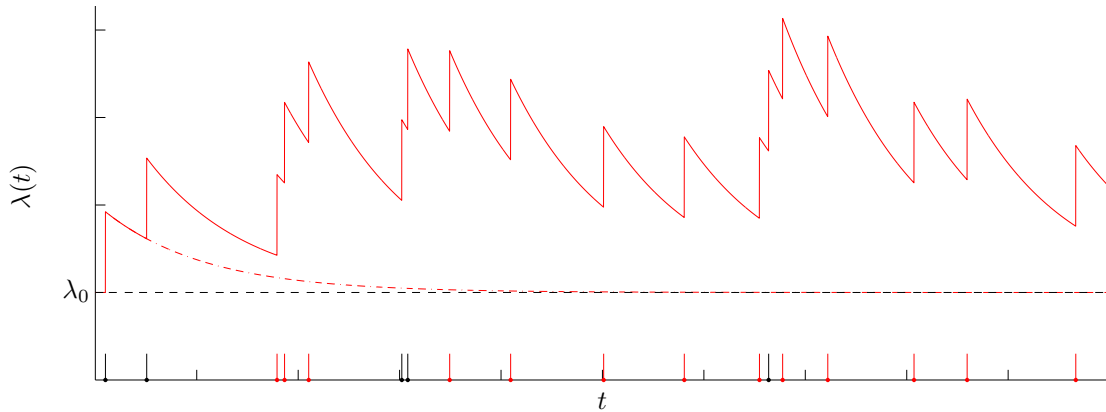


Figure 8.1: Hawkes process with exponential rate function. Immigrant points are shown in black and offspring point are shown in red. The immigrants arrive with constant intensity, λ_0 , shown by the dashed black line. The exponential rate function creates the saw tooth intensity function shown by the red line.

and hence giving rise to the self-excitatory behaviour. Inference in statistical models using Hawkes processes with exponentially decaying offspring intensity functions has been described by Simma and Jordan (2010) and others.

Whilst not unreasonable in general, the exponential intensity function is nonetheless a somewhat strong modelling assumption that may not always be appropriate. We could increase flexibility by using a mixture of distributions, a method described by Linderman and Adams (2014), or we could use a square transformed Gaussian process. Since a simple square transformed Gaussian process prior does not encode the reasonable modelling assumption that offspring are more likely to occur ‘sooner rather than later’, we will consider the exponentially attenuated square transformed Gaussian process offspring intensity function:

$$\lambda_n(x) = \gamma \beta \exp(-\beta(x - x^{(n)})) f^2(x - x^{(n)}). \quad (8.3)$$

An example of a Hawkes process generated using this intensity function is shown in Figure 8.2. In this example, the peak of the intensity function occurs somewhat after the parent point and there are several subsequent peaks. The overall intensity function is smooth and lacks the saw tooth shape of Figure 8.1. The probability of the offspring intensity functions $\lambda_{1:N}$ given a time ordered dataset $\mathcal{D} = \{x^{(n)} | x^{(n)} <$

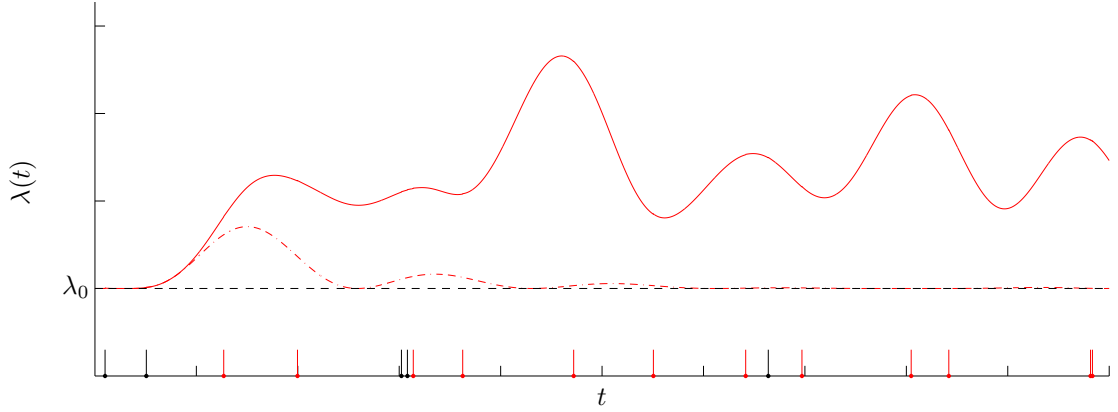


Figure 8.2: Hawkes process with attenuated Gaussian process rate function. The Gaussian process allows complex offspring intensity functions (shown by the dash-dotted red line) to be modelled.

$x^{(n+1)}\}_{n=1}^N$ is:

$$p(\lambda_{1:N}|\mathcal{D}) = \prod_{n=1}^N \exp\left(-\int_{x^{(n)}}^{\infty} \lambda_n(x) dx\right) \cdot \prod_{n=2}^N \left[\lambda_0(x^{(n)}) + \sum_{i=1}^{n-1} \lambda_i(x^{(n)})\right]. \quad (8.4)$$

To outline how we might apply the methods developed in this thesis, we will assume that all the intensity functions are the same, i.e. $\lambda_{1:N} := \Lambda \triangleq \gamma \beta \exp(-\beta x) f^2(x)$, where $f \sim \mathcal{GP}(\mu, \Sigma)$ and, in order to bound computation, impose an upper limit, $x_{\max}$, on the time between points and their potential offspring. Further assuming λ_0 is a constant, Equation 8.4 becomes:

$$p(\Lambda|\mathcal{D}) = \exp\left(-N \int_0^{x_{\max}} \Lambda(x) dx\right) \cdot \prod_{n=2}^N \left[\lambda_0 + \sum_{i=1}^{n-1} \Lambda(x^{(n)} - x^{(i)})\right]. \quad (8.5)$$

We can introduce allocation variables, A_n , as we did in Chapter 7, that assign points to potential parents:

$$p(\Lambda|\mathcal{D}) = \exp\left(-N \int_0^{x_{\max}} \Lambda(x) dx\right) \cdot \prod_{n=2}^N \left[\lambda_0^{\mathbb{1}\{A_n=0\}} \prod_{i=1}^{n-1} \Lambda(x^{(n)} - x^{(i)})^{\mathbb{1}\{A_n=i\}}\right]. \quad (8.6)$$

To apply variational inference, after expanding Λ , we can adopt an inducing point framework to marginalise f using a variational distribution $q(\mathbf{u})$ as we have done several times previously in this thesis. We can then collapse the evidence lower bound and optimise the model parameters $\{\mathbf{m}, \mathbf{S}, \gamma, \alpha, \beta, \lambda_0\}$.

In order to compute the integral we need the function Ψ for the function Equation 8.3 which is

$$\Psi(z, z') = \int_{x_0}^{x_1} \gamma \beta \exp(-\beta x) K(z, x) K(x, z') dx \quad (8.7)$$

$$= \frac{\gamma \beta \sqrt{\pi \alpha}}{2} \exp \left(\frac{1}{4} \left(\alpha \beta^2 - \frac{(z - z')^2}{\alpha} - 2\beta(z + z') \right) \right) \\ \times \left(\operatorname{erf} \left(\frac{-\alpha \beta - 2x_0 + z + z'}{2\sqrt{\alpha}} \right) - \operatorname{erf} \left(\frac{-\alpha \beta - 2x_1 + z + z'}{2\sqrt{\alpha}} \right) \right), \quad (8.8)$$

where $x_0 < x_1$ are the general integration limits that take the values $x_0 = 0$ and $x_1 = x_{\max}$ in Equation 8.5 and the kernel function, K , is as per Equation 7.36.

We can expand this framework to develop Hawkes process models for networks (Iwata et al. 2013). Given sets of interactions $\{\mathcal{D}_{i,j}\}_{i,j=1}^V$ we can require the points in set $\mathcal{D}_{i,j}$ to have parents in a set $\mathcal{D}_{k,i}$. We can use shared offspring intensity functions or learn bespoke function for each interacting pair, potentially using the latent variable model developed in Chapter 7 to share statistical strength amongst these functions.

8.2.2 Spectral Mixture Kernel

In this thesis we made extensive use of the exponentiated quadratic kernel. This kernel was convenient because it allows the integrals given by Equation 6.33 to be calculated in closed-form. The exponentiated quadratic kernel is a very popular and widely used GP kernel, although it is not ideal for all situations. Since the exponentiated quadratic kernel is infinitely differentiable, functions generated using it are infinitely smooth. However, many physical systems that we may want to model are not this smooth. Many other popular (less smooth) GP kernels, such as the rational quadratic and those in the Matern family, do not allow Ψ to be computed in closed-form. One kernel that does, is the Spectral Mixture Kernel (SMK) (Wilson and Adams 2013). This is important because the SMK is a particularly flexible and useful kernel.

As described by Wilson and Adams (2013), any stationary kernel, $K(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$, whose Fourier transform is a density, $S(\mathbf{s})$ (the *power spectral density*), can be written the dual form:

$$k(\boldsymbol{\tau}) = \int S(\mathbf{s}) \exp(2\pi i \mathbf{s}^\top \boldsymbol{\tau}) \, d\mathbf{s} \quad (8.9)$$

$$S(\mathbf{s}) = \int k(\boldsymbol{\tau}) \exp(-2\pi i \mathbf{s}^\top \boldsymbol{\tau}) \, d\boldsymbol{\tau}. \quad (8.10)$$

Since the Fourier transform of the exponentiated quadratic kernel is the Gaussian distribution, the kernel corresponds to a limited spectral density and therefore a narrow range of functions. As described by Wilson and Adams, we can approximate any spectral density—and therefore any stationary kernel—with arbitrary accuracy using mixtures of Gaussian distributions, in theory providing all the modelling flexibility we might need. If $S(\mathbf{s})$ is mixture of Gaussian densities:

$$S^{\text{sm}}(\mathbf{s}) := \prod_{r=1}^R \frac{\phi_r(s_r) + \phi_r(-s_r)}{2}, \quad (8.11)$$

where ϕ_r is the following Gaussian mixture in the r^{th} component of $\mathbf{s}$:

$$\phi_r(s_r; \mu_{r,1:A}, \sigma_{r,1:A}^2, w_{r,1:A}) \triangleq \sum_{a=1}^A w_{r,a} \frac{1}{2\pi\sigma_{r,a}^2} \exp\left(-\frac{(s_r - \mu_{r,a})^2}{2\sigma_{r,a}^2}\right), \quad (8.12)$$

and where $\{w_{r,a}\}$ are a set of weights that determine the contribution of the r^{th} , a^{th} spectral component, then the inverse Fourier transform of Equation 8.12 is a separable spectral mixture kernel,

$$k^{\text{sm}}(\boldsymbol{\tau}) = \prod_{r=1}^R \sum_{a=1}^A w_{r,a} \exp(-2\pi^2 \tau_r^2 \sigma_{r,a}^2) \cos(2\pi \tau_r \mu_{r,a}), \quad (8.13)$$

that is compatible with the Kronecker structured variational Gaussian process inference described in Chapter 7 (Wilson, Gilboa, et al. 2014). Kronecker structure is particularly important when using the spectral mixture kernel in a inducing point, or variational GP setting, as Kronecker structure allows us to have sufficient inducing points necessary to represent higher frequencies (Nickson et al. 2015).

Figure 8.3 shows an example of a draw from a squared Gaussian process with a spectral mixture covariance function. The figure illustrates that, by picking out

frequencies that allow sensible extrapolation of function, the spectral kernel can be useful for prediction beyond the region covered by the data (or in this case inducing points). The spectral density of the kernel used to construct Figure 8.3 is shown in Figure 8.4.

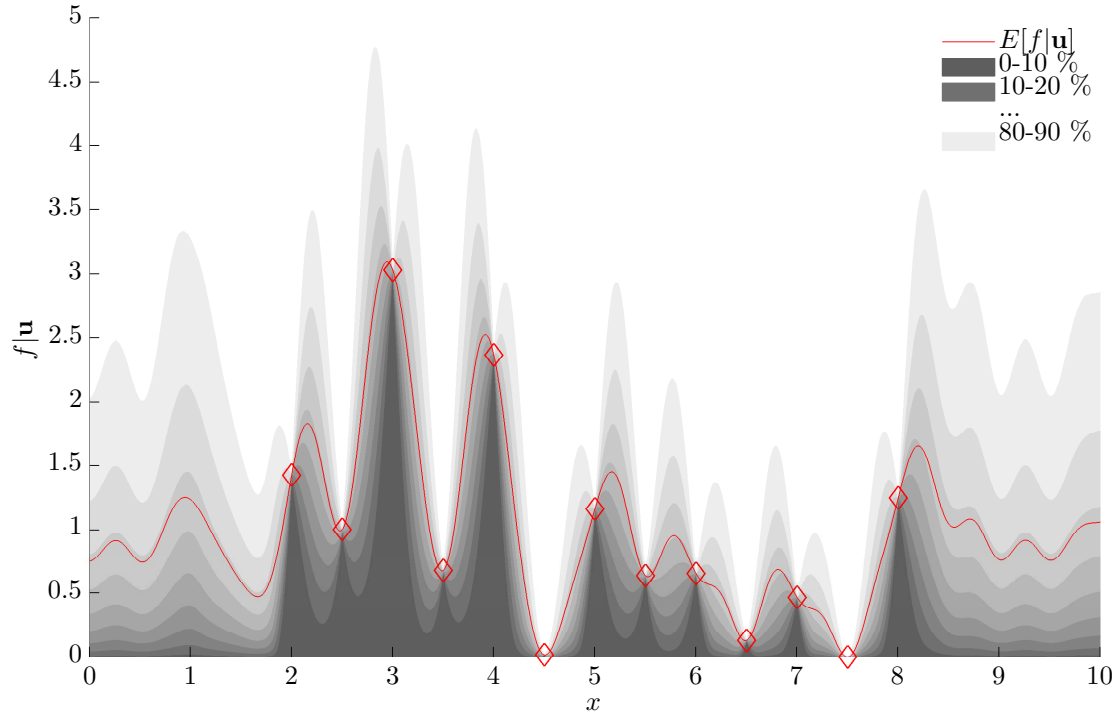


Figure 8.3: A draw from a Gaussian process with a spectral mixture kernel conditioned on a set of inducing variables indicated by the diamonds. The shaded regions indicate the 10 percentile regions and the red line the GP mean function. The integral of the function over $[0, 10]$ computed using Ψ^{sm} was 10.0299431, verified by numerical integration of $\mathbb{E}[f|\mathbf{u}]$ using 1000 trapeziums which gave 10.0299425.

To compute $\Psi(\mathbf{z}, \mathbf{z}')$ (Equation 6.33) we write $k^{\text{sm}}(\boldsymbol{\tau})$ explicitly in terms of $\mathbf{x}$ and $\mathbf{x}'$:

$$K^{\text{sm}}(\mathbf{x}, \mathbf{x}') \triangleq \prod_{r=1}^R K_r^{\text{sm}}(x_r, x'_r) \quad (8.14)$$

$$K_r^{\text{sm}}(x_r, x'_r) \triangleq \sum_{a=1}^A w_{r,a} \exp(-2\pi^2[x_r - x'_r]^2 \sigma_{r,a}^2) \cos(2\pi[x_r - x'_r] \mu_{r,a}). \quad (8.15)$$

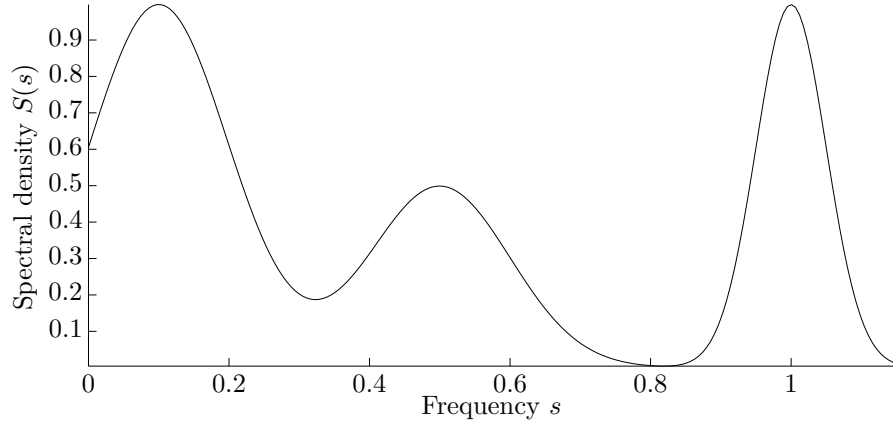


Figure 8.4: Power spectral density of Figure 8.3.

We can now introduce the inducing points $\mathbf{z}$ and $\mathbf{z}'$ and integrate out $\mathbf{x}$:

$$\int_{\mathcal{X}} K^{\text{sm}}(\mathbf{z}, \mathbf{x}) K^{\text{sm}}(\mathbf{x}, \mathbf{z}') d\mathbf{x} = \prod_{r=1}^R \int_{x_r^{\min}}^{x_r^{\max}} K_r^{\text{sm}}(z_r, x_r) K_r^{\text{sm}}(x_r, z'_r) dx_r \quad (8.16)$$

$$= \prod_{r=1}^R \sum_{a=1}^A \sum_{b=1}^A w_{r,a} w_{r,b} \Psi_{r,a,b}^{[0]} \left[\sum_{i=1}^8 \Psi_{r,a,b}^{[i]}(x_r) \right]_{x_r^{\min}}^{x_r^{\max}} \quad (8.17)$$

$$\triangleq \Psi^{\text{sm}}(\mathbf{z}, \mathbf{z}'). \quad (8.18)$$

The definition of $\Psi^{\text{sm}}(\mathbf{z}, \mathbf{z}')$ is somewhat complex. First we define the common factor

$$\Psi_{r,a,b}^{[0]} \triangleq \frac{\exp\left(\frac{-\mu_{r,a}^2 - \mu_{r,b}^2 - 4\pi^2 \sigma_{r,a}^2 \sigma_{r,b}^2 (z_r - z'_r)^2 - 8i\mu_{r,a} \pi \sigma_{r,b}^2 (z_r + z'_r) - 4i\mu_{r,b} \pi \sigma_{r,a}^2 (z_r + z'_r)}{2(\sigma_{r,a}^2 + \sigma_{r,b}^2)}\right)}{8\sqrt{2\pi(\sigma_{r,a}^2 + \sigma_{r,b}^2)}}, \quad (8.19)$$

and eight other terms, $\Psi_{r,a,b}^{[1:8]}(x_r)$,

$$\Psi_{r,a,b}^{[1]}(x_r) \triangleq -i \exp(\mathfrak{A}_{r,a,b}^{[1]} + \mathfrak{B}_{r,a,b}^{[1]}) \mathfrak{C}_{r,a,b} \operatorname{erfi}(\mathfrak{D}_{r,a,b}^{[1]}(x_r)) \quad (8.20)$$

$$\Psi_{r,a,b}^{[2]}(x_r) \triangleq -i \exp(\mathfrak{A}_{r,a,b}^{[1]} + \mathfrak{B}_{r,a,b}^{[2]}) \mathfrak{C}_{r,a,b} \operatorname{erfi}(\mathfrak{D}_{r,a,b}^{[2]}(x_r)) \quad (8.21)$$

$$\Psi_{r,a,b}^{[3]}(x_r) \triangleq -i \exp(\mathfrak{A}_{r,a,b}^{[2]} + \mathfrak{B}_{r,a,b}^{[3]}) \mathfrak{C}_{r,a,b} \operatorname{erfi}(\mathfrak{D}_{r,a,b}^{[3]}(x_r)) \quad (8.22)$$

$$\Psi_{r,a,b}^{[4]}(x_r) \triangleq -i \exp(\mathfrak{A}_{r,a,b}^{[2]} + \mathfrak{B}_{r,a,b}^{[4]}) \mathfrak{C}_{r,a,b} \operatorname{erfi}(\mathfrak{D}_{r,a,b}^{[4]}(x_r)) \quad (8.23)$$

$$\Psi_{r,a,b}^{[5]}(x_r) \triangleq -\exp(\mathfrak{A}_{r,a,b}^{[1]} + \mathfrak{B}_{r,a,b}^{[1]}) \mathfrak{S}_{r,a,b} \operatorname{erfi}(\mathfrak{D}_{r,a,b}^{[1]}(x_r)) \quad (8.24)$$

$$\Psi_{r,a,b}^{[6]}(x_r) \triangleq -\exp(\mathfrak{A}_{r,a,b}^{[1]} + \mathfrak{B}_{r,a,b}^{[2]}) \mathfrak{S}_{r,a,b} \operatorname{erfi}(\mathfrak{D}_{r,a,b}^{[2]}(x_r)) \quad (8.25)$$

$$\Psi_{r,a,b}^{[7]}(x_r) \triangleq \exp(\mathfrak{A}_{r,a,b}^{[2]} + \mathfrak{B}_{r,a,b}^{[3]}) \mathfrak{S}_{r,a,b} \operatorname{erfi}(\mathfrak{D}_{r,a,b}^{[3]}(x_r)) \quad (8.26)$$

$$\Psi_{r,a,b}^{[8]}(x_r) \triangleq \exp(\mathfrak{A}_{r,a,b}^{[2]} + \mathfrak{B}_{r,a,b}^{[4]}) \mathfrak{S}_{r,a,b} \operatorname{erfi}(\mathfrak{D}_{r,a,b}^{[4]}(x_r)), \quad (8.27)$$

each of which make further use of the following definitions:

$$\mathfrak{A}_{r,a,b}^{[1]} \triangleq \frac{4i\mu_{r,b}\pi\sigma_{r,b}^2 z_r}{\sigma_{r,a}^2 + \sigma_{r,b}^2} \quad (8.28)$$

$$\mathfrak{A}_{r,a,b}^{[2]} \triangleq \frac{4i\mu_{r,b}\pi\sigma_{r,b}^2 z'_r}{\sigma_{r,a}^2 + \sigma_{r,b}^2} \quad (8.29)$$

$$\mathfrak{B}_{r,a,b}^{[1]} \triangleq \frac{-\mu_{r,a}\mu_{r,b} + \mu_{r,a}2i\pi\sigma_{r,b}^2(3z_r + z'_r)}{\sigma_{r,a}^2 + \sigma_{r,b}^2} \quad (8.30)$$

$$\mathfrak{B}_{r,a,b}^{[2]} \triangleq \frac{\mu_{r,a}\mu_{r,b} + \mu_{r,a}2i\pi\sigma_{r,b}^2(z_r + 3z'_r)}{\sigma_{r,a}^2 + \sigma_{r,b}^2} \quad (8.31)$$

$$\mathfrak{B}_{r,a,b}^{[3]} \triangleq \frac{\mu_{r,a}\mu_{r,b} + \mu_{r,a}2i\pi\sigma_{r,b}^2(3z_r + z'_r)}{\sigma_{r,a}^2 + \sigma_{r,b}^2} \quad (8.32)$$

$$\mathfrak{B}_{r,a,b}^{[4]} \triangleq \frac{-\mu_{r,a}\mu_{r,b} + \mu_{r,a}2i\pi\sigma_{r,b}^2(z_r + 3z'_r)}{\sigma_{r,a}^2 + \sigma_{r,b}^2} \quad (8.33)$$

$$\mathfrak{C}_{r,a,b} \triangleq \cos(2\mu_{r,b}\pi[z_r - z'_r]) \quad (8.34)$$

$$\mathfrak{S}_{r,a,b} \triangleq \sin(2\mu_{r,b}\pi[z_r - z'_r]) \quad (8.35)$$

$$\mathfrak{D}_{r,a,b}^{[1]}(x_r) \triangleq \frac{-(\mu_{r,a} + \mu_{r,b}) + \mathfrak{S}_{r,a,b}(x_r)}{\mathfrak{H}_{r,a,b}} \quad (8.36)$$

$$\mathfrak{D}_{r,a,b}^{[2]}(x_r) \triangleq \frac{(\mu_{r,a} - \mu_{r,b}) + \mathfrak{S}_{r,a,b}(x_r)}{\mathfrak{H}_{r,a,b}} \quad (8.37)$$

$$\mathfrak{D}_{r,a,b}^{[3]}(x_r) \triangleq \frac{(\mu_{r,b} - \mu_{r,a}) + \mathfrak{S}_{r,a,b}(x_r)}{\mathfrak{H}_{r,a,b}} \quad (8.38)$$

$$\mathfrak{D}_{r,a,b}^{[4]}(x_r) \triangleq \frac{(\mu_{r,a} + \mu_{r,b}) + \mathfrak{S}_{r,a,b}(x_r)}{\mathfrak{H}_{r,a,b}} \quad (8.39)$$

$$\mathfrak{G}_{r,a,b}(x_r) \triangleq 2i\pi(\sigma_{r,a}^2(x_r - z_r) + \sigma_{r,b}^2(x_r - z'_r)) \quad (8.40)$$

$$\mathfrak{H}_{r,a,b} \triangleq \sqrt{2(\sigma_{r,a}^2 + \sigma_{r,b}^2)}. \quad (8.41)$$

where erfi is the imaginary error function. Although many of these terms are complex valued, Ψ^{sm} is always real valued. There are several points to take note of: first, the cost of computing Ψ^{sm} is quadratic in the number of mixture components A . Second, Equations 8.17-8.17 are arranged to be as succinct as possible and not for maximum numerical stability. To improve numerical stability, we should sum the exponents of Equation 8.19 and the exponents of the factors given in Equations 8.28-8.33 before evaluating the exponential to reduce numerical under and over flow. We can also use the *Dawson function*, $D(x)$, to implement the imaginary error

function:

$$\text{erfi}(x) = \frac{2}{\sqrt{\pi}} \exp(\underbrace{x^2}_{\text{combined exponent}}) D(x), \quad (8.42)$$

allowing more exponents to be combined.

8.2.3 Square transformed GP noise model

In some statistical applications, we may be interested in learning the maximum-likelihood (ML) parameters of a parametric Poisson process rate function in a model that uses a Cox process as simply a noise process, the parameters of which are not the main interest. Such a model may be written as

$$\lambda(\mathbf{x}) = \lambda_d(\mathbf{x}) + \lambda_n(\mathbf{x}), \quad (8.43)$$

where λ_d is the deterministic, parametric function of interest and λ_n is the random noise process. We want to use λ_d to explain as much of the observed data as possible, while using λ_n to soak up the remainder. With some minor modifications we can use the VBPP and our variational inference algorithm to construct a Bayesian squared transform Cox process noise model that incorporates λ_d into the mean function of the GP.

In order to maximise the interpretability of λ_d , it may be desirable for $\mathbb{E}[\lambda] = \lambda_d$ and therefore $\mathbb{E}[\lambda_n] = 0$. Since we require that the squared transformed GP model λ , i.e. $\lambda = f^2$, we must have $\mathbb{E}[f^2] = \lambda_d$. We can use the fact that $\mathbb{E}[f^2] = \mathbb{E}[f]^2 + \text{Var}[f]$, to find a mean function, $\bar{f} = \mathbb{E}[f]$, that ensures that $\lambda_d = \mathbb{E}[f^2]$ as follows:

$$\lambda_d(\mathbf{x}) = \bar{f}^2(\mathbf{x}) + K(\mathbf{x}, \mathbf{x}) \quad (8.44)$$

$$\Rightarrow \bar{f}(\mathbf{x}) = \sqrt{\lambda_d(\mathbf{x}) - K(\mathbf{x}, \mathbf{x})}. \quad (8.45)$$

Alternatively, instead of interpreting λ_d as the mean of λ , we may wish to use λ_n to model excess events over and above some baseline defined by λ_d , in which case we can simply define $\bar{f}(\mathbf{x}) = \sqrt{\lambda_d(\mathbf{x})}$.

We can now adapt the variational inference algorithm to incorporate $\bar{f}$. Since the inducing variables are part of the same GP as f , we have $p(\mathbf{u}) = \mathcal{N}(\bar{f}(\mathcal{Z}), \mathbf{K}_{zz})$,

where $\mathcal{Z}$ is the set of inducing points and mean of the GP conditioned on $\mathbf{u}$, $\mu(\mathbf{x})$ in Equation 6.1, now becomes

$$\mathbb{E}_{p(f|\mathbf{u})} = \mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} (\mathbf{u} - \bar{f}(\mathcal{Z})) + \bar{f}(\mathbf{x}). \quad (8.46)$$

The main change to the method presented in Chapter 6 is in the integral over the region $\mathcal{X}$. Using Equation 8.46 in that integral we now have:

$$\int_{\mathcal{X}} \mathbb{E}_{q(\mathbf{u})} \mathbb{E}_{f|\mathbf{u}}[f^2] \, d\mathbf{x} \quad (8.47)$$

$$= \int_{\mathcal{X}} \mathbb{E}_{q(\mathbf{u})} [\mathbb{E}_{p(f|\mathbf{u})}[f^2] + \text{Var}_{p(f|\mathbf{u})}[f^2]] \, d\mathbf{x} \quad (8.48)$$

$$= \int_{\mathcal{X}} \mathbb{E}_{q(\mathbf{u})} [\{\mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} (\mathbf{u} - \bar{f}(\mathcal{Z})) + \bar{f}(\mathbf{x})\}^2] + K(\mathbf{x}, \mathbf{x}) - \mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \mathbf{k}_{zx} \, d\mathbf{x}. \quad (8.49)$$

We can simplify the expansion of Equation 8.49 slightly by introducing the translated variable $\tilde{\mathbf{u}} = \mathbf{u} - \bar{f}(\mathcal{Z})$, which has a density function $p(\tilde{\mathbf{u}}) = \mathcal{N}(\tilde{\mathbf{u}}; \mathbf{0}, \mathbf{K}_{zz})$. This leads to:

$$\int_{\mathcal{X}} \mathbb{E}_{q(\tilde{\mathbf{u}})} [\{\mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \tilde{\mathbf{u}} + \bar{f}(\mathbf{x})\}^2] + K(\mathbf{x}, \mathbf{x}) - \mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \mathbf{k}_{zx} \, d\mathbf{x} \quad (8.50)$$

$$= \int_{\mathcal{X}} \mathbb{E}_{q(\tilde{\mathbf{u}})} [\mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \tilde{\mathbf{u}} \tilde{\mathbf{u}}^T \mathbf{K}_{zz}^{-1} \mathbf{k}_{zx} + 2\bar{f}(\mathbf{x}) \mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \tilde{\mathbf{u}}] + \bar{f}^2(\mathbf{x}) + K(\mathbf{x}, \mathbf{x}) - \mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \mathbf{k}_{zx} \, d\mathbf{x} \quad (8.51)$$

$$= \int_{\mathcal{X}} \mathbf{m}^T \mathbf{K}_{zz}^{-1} \mathbf{k}_{zx} \mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \mathbf{m} + 2\bar{f}(\mathbf{x}) \mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \mathbf{m} + \bar{f}^2(\mathbf{x}) + K(\mathbf{x}, \mathbf{x}) - \mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \mathbf{k}_{zx} + \mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \mathbf{S} \mathbf{K}_{zz}^{-1} \mathbf{k}_{zx} \, d\mathbf{x}, \quad (8.52)$$

where $q(\tilde{\mathbf{u}}) = \mathcal{N}(\mathbf{u}; \mathbf{m}, \mathbf{S})$ is the variational distribution. In comparison with Equations 6.31 and 6.32 we now have two extra integral terms. The first of these integrals, $\int_{\mathcal{X}} \bar{f}^2(\mathbf{x}) \, d\mathbf{x}$, should be straight forward provided λ_d is sensibly chosen. The second integral we must compute is $\int_{\mathcal{X}} 2\bar{f}(\mathbf{x}) \mathbf{k}_{xz} \mathbf{K}_{zz}^{-1} \mathbf{m} \, d\mathbf{x}$ which is unlikely to be analytically tractable unless we are very careful². Defining $\bar{f} = \sqrt{\lambda_d}$, rather than using Equation 8.45, may be slightly more advantageous in this regard, although even this is highly dependent on λ_d .

²Note that if Equation 8.45 is used we must also somehow constrain $\lambda_d(\mathbf{x}) > K(\mathbf{x}, \mathbf{x})$.

8.2.4 Streaming inference

In an *on-line* learning setting, such as the one described by Broderick et al. (2013), data arrives in a continuous stream and the challenge is to develop algorithms that can continually ingest sequential data as it arrives, ideally yielding results that are the same as—or a very close approximation of—an equivalent algorithm that has all the data available to it from the start. As described in Chapter 1, event processes typically generate precisely such data streams and therefore developing the methods previously described to work on-line is a natural direction for further work. In this section we will see how the VBPP method could be modified into an on-line algorithm.

We begin by noting that for a time ordered dataset, $\mathcal{D} = \{x^{(n)} | x^{(n)} < x^{(n+1)}\}_{n=1}^N$, the Cox process likelihood on the interval $[0, x^{(N)}]$ can be written as

$$p(\mathcal{D}|\lambda) = \exp\left(-\int_0^{x^{(1)}} \lambda(x) \, dx\right) \lambda(x^{(1)}) \prod_{n=2}^N \exp\left(-\int_{x^{(n-1)}}^{x^{(n)}} \lambda(x) \, dx\right) \lambda(x^{(n)}), \quad (8.53)$$

i.e., as the product of sequential factors that can be processed in turn. We will assume that the goal is to infer the current intensity $\lambda(x)$ given the previously observed data $x^{(1:n)} < x$ and, for the sake of keeping the exposition simple, that we will process one event at a time, although the method easily generalises to handle multiple sequential updates simultaneously.

In order to construct a recursive—Kalman filter like—algorithm, we will assume that our GP kernel is one that has the *Markov* property, for example the mean reverting Ornstein-Uhlenbeck kernel³:

$$K^{\text{OU}}(x, x') = \gamma \exp(-\alpha|x - x'|). \quad (8.54)$$

We will begin by denoting $f^{(n)} \triangleq f([x^{(n-1)}, x^{(n)}])$ as a GP distributed function between successive data points. Similarly,

$$\mathcal{Z}^{(n)} \triangleq \{z^{(m)} | x^{(n-1)} = z^{(1)} < z^{(m)} < z^{(M)} = x^{(n)}\}_{m=1}^M, \quad (8.55)$$

³The function Ψ can be computed for this kernel.

is a set of inducing points located between successive data points, and $\mathbf{u}^{(n)}$ are the corresponding inducing variables. This set-up is shown in Figure 8.5. Due

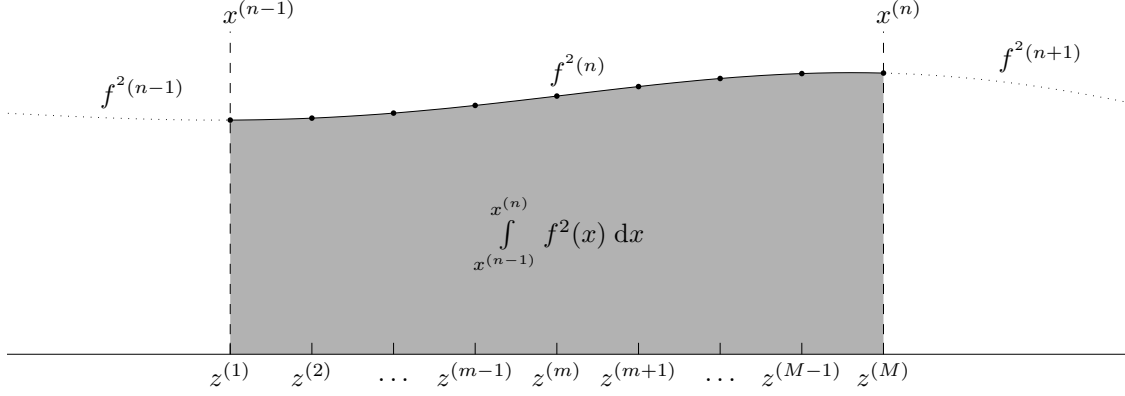


Figure 8.5: Streaming intensity function inference set-up showing the region to be integrated at each update and the locations of the inducing points and variables.

to the Markov assumption $f^{(n)}$ is completely determined by $\mathbf{u}^{(n)}$ and therefore $p(f^{(n)}|\mathbf{u}^{(n)}, \mathbf{u}^{(n-1)}, \dots, \mathbf{u}^{(1)}) = p(f^{(n)}|\mathbf{u}^{(n)})$.

At each iteration of the algorithm, we obtain a variational distribution $q(\mathbf{u}^{(n)}) = \mathcal{N}(\mathbf{m}_+^{(n)}, \mathbf{S}_+^{(n)})$ that, by inductive assumption, is approximately equal to the true posterior $p(\mathbf{u}^{(n)}|x^{(1:n)})$. To predict the filter, we construct the next set of inducing points $\mathcal{Z}^{(n+1)}$ and integrate out the current set of inducing variables $\mathbf{u}^{(n)}$ using the kernel as the filter transition density as follows:

$$p(\mathbf{u}^{(n+1)}|x^{(1:n)}) \approx \int p(\mathbf{u}^{(n+1)}|\mathbf{u}^{(n)})q(\mathbf{u}^{(n)}) d\mathbf{u}^{(n)} \quad (8.56)$$

$$= \mathcal{N}(\mathbf{u}^{(n+1)}; \mathbf{m}_-^{(n+1)}, \mathbf{S}_-^{(n+1)}), \quad (8.57)$$

where the predicted mean, $\mathbf{m}_-^{(n+1)}$, and variance, $\mathbf{S}_-^{(n+1)}$, are given by

$$\mathbf{m}_-^{(n+1)} = \mathbf{k}_{z^{(n+1)}x^{(n)}} \frac{[\mathbf{m}_+^{(n)}]_M}{K(x^{(n)}, x^{(n)})} \quad (8.58)$$

$$\mathbf{S}_-^{(n+1)} = \mathbf{K}_{z^{(n+1)}z^{(n+1)}} + \mathbf{k}_{z^{(n+1)}x^{(n)}} \left[\frac{[\mathbf{S}_+^{(n)}]_{M,M}}{K^2(x^{(n)}, x^{(n)})} - \frac{1}{K(x^{(n)}, x^{(n)})} \right] \mathbf{k}_{x^{(n)}z^{(n+1)}}. \quad (8.59)$$

In Equations 8.58 and 8.59 we have again made use of the Markov assumption and so the function value over the interval $[x^{(n)}, x^{(n+1)}]$ only depends on the distribution of a single inducing variable in $\mathbf{u}^{(n)}$, located at the last point in $\mathcal{Z}^{(n)}$, which is $x^{(n)}$.

We can then use $\mathcal{N}(\mathbf{m}_-^{(n+1)}, \mathbf{S}_-^{(n+1)})$ as the prior distribution for the next iteration, maximise the next ELBO and obtain a new posterior estimate $q(\mathbf{u}^{(n+1)})$, thereby completing the predict-update cycle.

Appendices



Derivatives of the VBPP lower bound

To define the required derivatives of $\mathcal{L}_{\text{VBPP}}$ we will first separate the Evidence Lower Bound (ELBO) into the following terms:

$$\mathcal{L}_{\text{VBPP}} = \mathcal{L}^{[1]} + \mathcal{L}^{[2]} + \mathcal{L}^{[3]} + \mathcal{L}^{[4]} + \mathcal{L}^{[5]} \quad (\text{A.1})$$

$$\mathcal{L}^{[1]} \triangleq -\mathbf{m}^\top \mathbf{K}_{zz}^{-1} \Psi_{zz} \mathbf{K}_{zz}^{-1} \mathbf{m} \quad (\text{A.2})$$

$$\mathcal{L}^{[2]} \triangleq -\gamma |\mathcal{X}| + \text{tr}(\mathbf{K}_{zz}^{-1} \Psi_{zz}) - \text{tr}(\mathbf{K}_{zz}^{-1} \mathbf{S} \mathbf{K}_{zz}^{-1} \Psi_{zz}) \quad (\text{A.3})$$

$$\mathcal{L}^{[3]} \triangleq -\sum_{n=1}^N \tilde{G} \left(-\frac{\tilde{\mu}_n^2}{2\tilde{\sigma}_n^2} \right) \quad (\text{A.4})$$

$$\mathcal{L}^{[4]} \triangleq \sum_{n=1}^N \log[\tilde{\sigma}_n^2] - (C + \log 2) \cdot N \quad (\text{A.5})$$

$$\mathcal{L}^{[5]} \triangleq -\text{KL}(q(\mathbf{u}) || p(\mathbf{u})). \quad (\text{A.6})$$

A.1 Variational parameters

Using these definitions, the non-zero derivatives of the bound with respect to the variational mean parameter, $\mathbf{m}$, are:

$$\frac{\partial \mathcal{L}^{[1]}}{\partial \mathbf{m}} = -2\mathbf{K}_{zz}^{-1} \Psi_{zz} \mathbf{K}_{zz}^{-1} \mathbf{m} \quad (\text{A.7})$$

$$\frac{\partial \mathcal{L}^{[3]}}{\partial \mathbf{m}} = \sum_{n=1}^N \left[\tilde{G}' \left(-\frac{\tilde{\mu}_n^2}{2\tilde{\sigma}_n^2} \right) \times \frac{\tilde{\mu}_n}{\tilde{\sigma}_n^2} \mathbf{K}_{zz}^{-1} \mathbf{k}_{zx}^{(n)} \right] \quad (\text{A.8})$$

$$\frac{\partial \mathcal{L}^{[5]}}{\partial \mathbf{m}} = -\mathbf{K}_{zz}^{-1} \mathbf{m} \quad (\text{A.9})$$

and the derivatives with respect to the variational covariance, $\mathbf{S} = \mathbf{L}\mathbf{L}^\top$, are:

$$\frac{\partial \mathcal{L}^{[2]}}{\partial \mathbf{S}} = -\text{tr}(\mathbf{K}_{zz}^{-1} \Psi_{zz} \mathbf{K}_{zz}^{-1}) \quad (\text{A.10})$$

$$\frac{\partial \mathcal{L}^{[4]}}{\partial \mathbf{S}} = \sum_{n=1}^N \left[\left(-\tilde{G}' \left(-\frac{\tilde{\mu}_n^2}{2\tilde{\sigma}_n^2} \right) \frac{1}{2\tilde{\sigma}_n^4} + \frac{1}{\tilde{\sigma}_n^2} \right) \times \mathbf{K}_{zz}^{-1} \mathbf{k}_{zx}^{(n)} \mathbf{k}_{xz}^{(n)} \mathbf{K}_{zz}^{-1} \right] \quad (\text{A.11})$$

$$\frac{\partial \mathcal{L}^{[5]}}{\partial \mathbf{S}} = -\frac{1}{2} [\mathbf{K}_{zz}^{-1} - \mathbf{S}^{-1}]. \quad (\text{A.12})$$

and the derivative with respect to $\text{vech}(\mathbf{L})$ is found by applying the chain rule:

$$\frac{\partial \mathcal{L}}{\partial \text{vech}(\mathbf{L})} = 2 \text{vech} \left(\frac{\partial \mathcal{L}}{\partial \mathbf{S}} \mathbf{L} \right). \quad (\text{A.13})$$

A.2 Kernel parameters

We define the following derivative kernels with respect to the kernel parameters

$\omega \in \{\gamma, \alpha_{1:R}\} \cup \{z_1^{(m)}, \dots, z_R^{(m)}\}_{m=1}^M$:

$$d\mathbf{K}_{zz} = \frac{\partial \mathbf{K}_{zz}}{\partial \omega} \quad (\text{A.14})$$

$$d\mathbf{k}_{xz}^{(n)} = \frac{\partial \mathbf{k}_{xz}^{(n)}}{\partial \omega} \quad (\text{A.15})$$

$$d\Psi_{zz} = \frac{\partial \Psi_{zz}}{\partial \omega}. \quad (\text{A.16})$$

Using these definitions, the derivatives of the terms are:

$$\frac{\partial \mathcal{L}^{[1]}}{\partial \omega} = -2 \mathbf{m}^\top [d\mathbf{K}_{zz}^{-1}] \Psi_{zz} \mathbf{K}_{zz}^{-1} \mathbf{m} - \mathbf{m}^\top \mathbf{K}_{zz}^{-1} [d\Psi_{zz}] \mathbf{K}_{zz}^{-1} \mathbf{m} \quad (\text{A.17})$$

$$\begin{aligned} \frac{\partial \mathcal{L}^{[2]}}{\partial \omega} = & -\frac{\partial \gamma}{\partial \omega} |\mathcal{X}| + \text{tr}([d\mathbf{K}_{zz}^{-1}] \Psi_{zz}) + \text{tr}(\mathbf{K}_{zz}^{-1} [d\Psi_{zz}]) \\ & - 2 \text{tr}(\mathbf{S} [d\mathbf{K}_{zz}^{-1}] \Psi_{zz} \mathbf{K}_{zz}^{-1}) - \text{tr}(\mathbf{S} \mathbf{K}_{zz}^{-1} [d\Psi_{zz}] \mathbf{K}_{zz}^{-1}) \end{aligned} \quad (\text{A.18})$$

$$\frac{\partial \mathcal{L}^{[3]}}{\partial \omega} = \sum_{n=1}^N \tilde{G}' \left(-\frac{\tilde{\mu}_n^2}{2\tilde{\sigma}_n^2} \right) \frac{2 [d\tilde{\mu}_n] \tilde{\mu}_n \tilde{\sigma}_n^2 - \tilde{\mu}_n^2 [d\tilde{\sigma}_n^2]}{2 \tilde{\sigma}_n^4} \quad (\text{A.19})$$

$$\frac{\partial \mathcal{L}^{[4]}}{\partial \omega} = \sum_{n=1}^N \frac{[d\tilde{\sigma}_n^2]}{\tilde{\sigma}_n^2} \quad (\text{A.20})$$

$$\frac{\partial \mathcal{L}^{[5]}}{\partial \omega} = -\frac{1}{2} \left[\text{tr}([d\mathbf{K}_{zz}^{-1}] \mathbf{S}) + (\mathbf{m} - \bar{u})^\top [d\mathbf{K}_{zz}^{-1}] (\mathbf{m} - \bar{u}) - \text{tr}(\mathbf{K}_{zz} [d\mathbf{K}_{zz}^{-1}]) \right], \quad (\text{A.21})$$

where

$$d\mathbf{K}_{zz}^{-1} = -\mathbf{K}_{zz}^{-1}[d\mathbf{K}_{zz}]\mathbf{K}_{zz}^{-1} \quad (\text{A.22})$$

$$d\tilde{\mu}_n = [d\mathbf{k}_{xz}^{(n)}]\mathbf{K}_{zz}^{-1}\mathbf{m} + \mathbf{k}_{xz}^{(n)}[d\mathbf{K}_{zz}^{-1}]\mathbf{m} \quad (\text{A.23})$$

$$d\tilde{\sigma}_n^2 = -2 [d\mathbf{k}_{xz}]\mathbf{K}_{zz}^{-1}\mathbf{k}_{zx}^{(n)} - \mathbf{k}_{xz}^{(n)}[d\mathbf{K}_{zz}^{-1}]\mathbf{k}_{zx}^{(n)} \quad (\text{A.24})$$

$$+ 2 [d\mathbf{k}_{xz}]\mathbf{K}_{zz}^{-1}\mathbf{S}\mathbf{K}_{zz}^{-1}\mathbf{k}_{zx}^{(n)} + 2 \mathbf{k}_{xz}^{(n)}\mathbf{K}_{zz}^{-1}\mathbf{S}[d\mathbf{K}_{zz}^{-1}]\mathbf{k}_{zx}^{(n)}, \quad (\text{A.25})$$

and

$$d\mathbf{K}_{zz} = \frac{\partial \mathbf{K}_{zz}}{\partial \omega} \quad (\text{A.26})$$

$$d\mathbf{k}_{xz}^{(n)} = \frac{\partial \mathbf{k}_{xz}^{(n)}}{\partial \omega}. \quad (\text{A.27})$$

Finally we also have the derivative of the Kullback–Leibler (KL) divergence with respect to the mean of the prior $\bar{u}$:

$$\frac{\partial \mathcal{L}^{[5]}}{\partial \bar{u}} = -\bar{u}\mathbb{1}^\top \mathbf{K}_{zz}^{-1}\mathbb{1} + \mathbb{1}^\top \mathbf{K}_{zz}^{-1}\mathbf{m}, \quad (\text{A.28})$$

where $\mathbb{1}$ is a vector of 1's. For all other terms the derivative is zero.

A.3 Kronecker structured variational covariance

To compute the derivative, $\frac{\partial \mathcal{L}}{\partial}$, of the ELBO with respect to the component matrices, $\mathbf{S}_r$, of the Kronecker structured $\mathbf{S}$ matrix¹ we first break down $\mathcal{L}_{\text{PP}}$ into more than the five terms in Equations A.2-A.6. The lower bound has $T = 2N + 3$ additive terms, $\mathcal{L}^{[1:T]}$, that involve $\mathbf{S}$. The partial derivative of $\mathcal{L}_{\text{PP}}$ with respect to $\mathbf{S}$ will therefore yield T Kronecker structured matrices:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{S}} = \sum_{t=1}^T \frac{\mathcal{L}^{[t]}}{\partial \mathbf{S}} = \sum_{t=1}^T \bigotimes_{r=1}^R \check{\mathbf{S}}_r^{[t]}. \quad (\text{A.29})$$

Since we cannot add Kronecker matrices without losing Kronecker structure we must apply the chain rule to each of these terms individually. It is helpful to vectorise $\mathbf{S}$ and $\mathbf{S}_r$ so that the Jacobian and chain rule,

$$\frac{\partial \mathcal{L}^{[t]}}{\partial \text{vec}(\mathbf{S}_r)} = \frac{\partial \text{vec}(\mathbf{S})}{\partial \text{vec}(\mathbf{S}_r)} \cdot \frac{\partial \mathcal{L}^{[t]}}{\partial \text{vec}(\mathbf{S})}, \quad (\text{A.30})$$

¹Assuming the simpler parametrisation of Equation 7.8

are well defined. If we consider the full matrix $\frac{\partial \mathcal{L}^{[t]}}{\partial \mathbf{S}}$ for a moment, its entries are of the form:

$$[\check{\mathbf{S}}_1^{[t]}]_{i_1, j_1} \times [\check{\mathbf{S}}_2^{[t]}]_{i_2, j_2} \times \dots \times [\check{\mathbf{S}}_R^{[t]}]_{i_R, j_R}. \quad (\text{A.31})$$

Now, without loss of generality, the derivative of an element of $\mathbf{S}$ with respect to $[\mathbf{S}_1]_{i_1, j_1}$ has the form

$$[\mathbf{S}_2]_{i_2, j_2} \times \dots \times [\mathbf{S}_R]_{i_R, j_R}, \quad (\text{A.32})$$

if $[\mathbf{S}_1]_{i_1, j_1}$ participates in the particular element and 0 otherwise. When we compute Equation A.30 we calculate an inner product between a vector with elements given by Equation A.31 and sparse matrix with non-zero elements of the form of Equation A.32. Calculating this inner product for $[\mathbf{S}_1]_{i_1, j_1}$ leads to

$$\frac{\partial \mathcal{L}^{[t]}}{\partial [\mathbf{S}_1]_{i_1, j_1}} = \sum_{i_2, j_2} \dots \sum_{i_R, j_R} [\check{\mathbf{S}}_1^{[t]}]_{i_1, j_1} \times [\check{\mathbf{S}}_2^{[t]}]_{i_2, j_2} [\mathbf{S}_2]_{i_2, j_2} \times \dots \times [\check{\mathbf{S}}_R^{[t]}]_{i_R, j_R} [\mathbf{S}_R]_{i_R, j_R} \quad (\text{A.33})$$

$$= [\check{\mathbf{S}}_1^{[t]}]_{i_1, j_1} \prod_{r=2}^R \left[\sum_{i_r, j_r} [\check{\mathbf{S}}_r^{[t]}]_{i_r, j_r} [\mathbf{S}_r]_{i_r, j_r} \right]. \quad (\text{A.34})$$

Tidying up and generalising Equation A.34 gives:

$$\frac{\partial \mathcal{L}^{[t]}}{\partial \mathbf{S}_r} = \check{\mathbf{S}}_r^{[t]} \cdot \prod_{i=1}^R [S_i^{[t]}]_{\mathbb{1}\{i \neq r\}} \triangleq \hat{\mathbf{S}}_r^{[t]} \quad (\text{A.35})$$

$$S_r^{[t]} \triangleq \sum_{m=1}^{M_r} \sum_{n=1}^{M_r} [\check{\mathbf{S}}_r^{[t]} \odot \mathbf{S}_r]_{m, n}. \quad (\text{A.36})$$

The desired partial derivative can now be computed by summing the partial derivatives for each term:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{S}_r} = \sum_{t=1}^T \hat{\mathbf{S}}_r^{[t]} \triangleq \mathbf{S}'_r. \quad (\text{A.37})$$

If $\mathbf{S}_r = \mathbf{L}_r \mathbf{L}_r^\top$, where $\mathbf{L}_r$ is a lower triangular matrix, we can now apply the chain rule to get the partial derivative with respect to $\mathbf{L}_r$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{L}_r} = 2 \text{vech}(\mathbf{S}'_r \cdot \mathbf{L}_r). \quad (\text{A.38})$$

B

Kronecker matrix implementation

To manage the additional software complexity created by the use of Kronecker matrices, it is helpful to encapsulate all Kronecker operations in a separate class that abstracts the underlying manipulations. For example, in MATLAB we can overload operators ‘*’ and ‘\’ by implementing the `mtimes` and `mldivide` methods, as well as standard linear algebra operations such as `chol`, as shown in the `Kronmat` class Code Listing B.1. In late binding languages like MATLAB and Python, this also allows significant reuse of previously developed software, as `Kronmat` objects are operated on completely transparently, as though they are built-in or standard types. Stacked Kronecker structured vectors are implemented using a separate `Kronvecstack` class that overloads standard operators and methods like the `Kronmat` class, albeit with fewer supported operations. A subtle difference is that Kronecker vectors are always stored column-wise since the rows and columns of `Kronvecstack` matrices are semantically different. For this reason, transposition is explicitly represented (see Code Listing B.2). One operation that cannot be computed without looping is the multiplication of a `Kronvecstack` with a full vector. Since this operation requires N loops, it is implemented in C and interfaced with MATLAB or Python via their respective C interfaces (see Code Listing B.3).

Code Listing B.1: The Kronmat class abstracts Kronecker matrix manipulation.

```

classdef Kronmat
%KRONMAT    Kronecker structured matrices.
    properties(GetAccess = public, SetAccess = protected)
        mats = {};
    end

    function r = mtimes(this, rhs)
%*   Kronmat multiplication.
        if isa(rhs, 'Kronmat')
            r = Kronmat();
            for i = 1:length(this.mats)
                r.mats{i} = this.mats{i} * rhs.mats{i};
            end
        else % Other cases ...
            end
        end

    function r = mldivide(this, rhs)
%\   Kronmat backslash.
        if isa(rhs, 'double') && size(rhs,2) == 1
            % Solve the linear system Ax = b.
            r = rhs;
            for i = length(this.mats):-1:1
                matColumns = size(this.mats{i}, 2);
                r = reshape(r, [matColumns, length(r)/matColumns]);
                r = (this.mats{i}\r)';
                r = r(:);
            end
        else % Other cases ...
            end
        end

    function r = chol(this, ul)
%CHOL    Cholesky factorization.
        if nargin < 2; ul = 'upper'; end
        r = Kronmat();
        for i = 1:length(this.mats)
            r.mats{i} = chol(this.mats{i}, ul);
        end
    end

% Other member functions...
end

```

Code Listing B.2: Multiple Kronecker vectors with the same structure are efficiently implemented using the Kronvecstack class.

```
classdef Kronvecstack
%KRONVECSTACK   Kronecker structured vectors
    properties(GetAccess = public, SetAccess = protected) {}
        mats = {};
        ctransposed = false; % Explicitly represent transposition
    end

% Rest of implementation ...
end
```

Code Listing B.3: Kronvecstack dense vector multiplication loops implemented in C99.

```
int vec_dot_kvs(size_t N, // number of Kronecker vectors
               size_t R, // number of dimensions
               size_t M, // length of vec (product of shapes)
               size_t *shapes, // input length of Kronecker vectors
               double *vec, // input vector
               double **mats, // input Kronecker vector mats
               double *ans) { // output vector
    double *A = (double*) malloc(M * sizeof(double));
    double *y = (double*) malloc(M * sizeof(double));
    for(size_t i = 0; i < N; ++i) {
        size_t n = M; // length of vector y
        memcpy(y, vec, M * sizeof(double));
        for (size_t r = R; r > 0; --r) {
            size_t m = shapes[r-1]; // rows of matrix A
            n = n / m; // columns of matrix A
            double *tmp = A; A = y; y = tmp; // y vector is new A matrix
            cblas_dgemv(CblasColMajor, // y:= alpha.A.x + beta.y
                       CblasTrans, // A is transposed
                       m, n, // rows, columns of matrix A
                       1.0, // alpha
                       A, m, // input matrix A, leading diagonal of A
                       &mats[r-1][i*m], 1, // input vector x, inc of x
                       0.0, // beta
                       y, 1); // output vector y, inc of y
        }
        ans[i] = y[0];
    }
    free(A); free(y);
    return 0;
}
```




Collapsed variational inference for the Poisson-IRM model

This appendix details the application of the collapsed variational inference, described in §3.2.3 to the Bayesian non-parametric (BNP) variant of the Stochastic Block Model (SBM) called the Infinite Relational Model (IRM), which was previously described in §4.3.1, we will refer to the resulting algorithm as Col-IRM (Lloyd 2013).

In the unipartite IRM model each link is governed two latent variables, with all latent variables interacting in a pairwise fashion. Albers et al. (2013) report that inference in this model is particularly hard using Markov-Chain Monte-Carlo (MCMC) due to these tightly coupled interactions. As has been reported in the literature (King and Lawrence 2006; Teh, Newman, et al. 2007) collapsed variational inference—where some of the model parameters are marginalised analytically—can give better performance in models that have highly correlated variables. Collapsed variational inference may therefore offer potentially benefits when applied to the IRM.

C.1 Poisson-IRM

The Poisson-IRM is a variant of the IRM appropriate for positive integer value data such as counts of interactions in Dynamic Interaction Networks (DINS) that—as the name suggests—uses a Poisson likelihood:

$$\mathbf{Y} \sim \mathcal{P}(\mathbf{Z}\mathbf{\Lambda}\mathbf{Z}^\top). \quad (\text{C.1})$$

The community assignments, $\mathbf{Z}$, is a matrix with N rows and a potentially infinite number of columns and $\mathbf{z}^{(n)}$ is the n^{th} row of this matrix. We will use $\mathbf{z}_i^{(n)}$ to denote the i^{th} entry of $\mathbf{z}^{(n)}$ that contains a single 1 located at $\tilde{z}^{(n)}$; therefore $\mathbf{z}_{\tilde{z}^{(n)}}^{(n)} = 1$ and all other entries are 0. The matrix $\mathbf{\Lambda}$ contains the intra and inter community interaction intensities, $\lambda_{i,j}$. The full generative model, including the stick breaking prior, is given by Equations C.3-C.6 and shown graphically in Figure C.1.

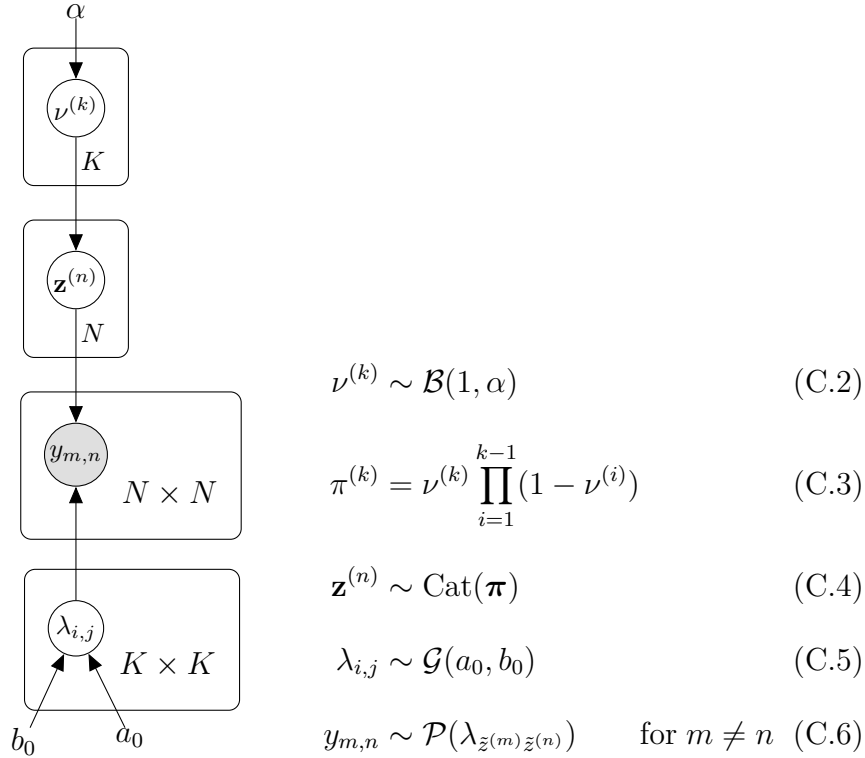


Figure C.1: Graphical model of the Poisson Infinite Relational Model.

Note we explicitly exclude symmetric links from the likelihood Equation C.6 since we cannot usually observe reflexive interactions. We also truncate the stick breaking construction at K , which is assumed to larger than the number of observed clusters.

C.2 Collapsed variational lower bound

To derive the collapsed lower bound we condition on the global variables, $\mathbf{\Lambda}$ and $\mathbf{V}$, and marginalise the local variables $\mathbf{Z}$ using a variational distribution $q(\mathbf{Z}) = \prod_{n=1}^N q(\mathbf{z}^{(n)}) = \prod_{n=1}^N \text{Cat}(\mathbf{z}; r_{n,1:K})$, as follows:

$$\begin{aligned} & \log p(\mathbf{Y}|\mathbf{V}, \mathbf{\Lambda}) \\ & \geq \mathbb{E}_{q(\mathbf{Z})} \left[\log \frac{p(\mathbf{Y}|\mathbf{Z}, \mathbf{\Lambda})p(\mathbf{Z}|\mathbf{V})}{q(\mathbf{Z})} \right] \end{aligned} \quad (\text{C.7})$$

$$\begin{aligned} & = \sum_{\substack{m=1 \\ n=1}}^N c_{m,n} \sum_{\substack{i=1 \\ j=1}}^K \left[\mathbb{E}_{q(\mathbf{z}^{(m)})}[\mathbf{z}_i^{(m)}] \cdot \mathbb{E}_{q(\mathbf{z}^{(n)})}[\mathbf{z}_j^{(n)}] \cdot (y_{m,n} \log \lambda_{i,j} - \log(y_{m,n}!) - \lambda_{i,j}) \right] \\ & + \sum_{n=1}^N \sum_{k=1}^K \mathbb{E}_{q(\mathbf{z}^{(n)})}[\mathbf{z}_k^{(n)}] \log \pi^{(k)} + \mathbb{H}_{q(\mathbf{Z})}[\mathbf{Z}] \\ & \triangleq \mathcal{L}(\mathbf{V}, \mathbf{\Lambda}). \end{aligned} \quad (\text{C.8})$$

The variable $c_{m,n} \in \{0, 1\}$ indicates whether the link $m \rightarrow n$ participates in the likelihood. Since we have explicitly excluded reflexively links we must¹ set $c_{n,n} := 0$. If the network is symmetric then we can also set $c_{m,n} := 0$ for $m > n$. We should also set $c_{m,n} := 0$ for any unobserved links and for any links that are held out for test purposes.

After replacing the random variables with their expectations, we see that the conditional bound separates into three terms, two of which are conjugate to the joint prior, $p(\mathbf{V})p(\mathbf{\Lambda})$, and an entropy term which is not a function of $\mathbf{V}$ or $\mathbf{\Lambda}$ and a constant. After integrating $\exp(\mathcal{L}(\mathbf{V}, \mathbf{\Lambda}))$ with respect to the joint prior, the form of the collapsed bound emerges as sums of the log normalisers of the prior and

¹ If we wish to include reflexive links we also need the expectation $\mathbb{E}_{q(\mathbf{z}^{(n)})}[\mathbf{z}_i^{(n)} \mathbf{z}_j^{(n)}]$ —equal to $r_{n,i}$ if $i = j$ and 0 otherwise—resulting in some terms being effectively excluded from the bound and some new terms being included.

posterior distributions of the model parameters:

$$\begin{aligned}
\log p(\mathbf{Y}) &\geq K^2 \psi_{\mathcal{G}}(a_0, b_0) - \sum_{i=1}^K \sum_{j=1}^K \psi_{\mathcal{G}}(a_{i,j}, b_{i,j}) \\
&\quad + K \psi_{\mathcal{B}}(1, \alpha) - \sum_{k=1}^K \psi_{\mathcal{B}}(\gamma_{1,k}, \gamma_{2,k}) \\
&\quad + \mathbb{H}_{q(\mathbf{Z})}[\mathbf{Z}] \\
&\quad - \sum_{m=1}^N \sum_{n=1}^N c_{m,n} \log(y_{m,n}!) \\
&\triangleq \mathcal{L}_{\text{CIRM}},
\end{aligned} \tag{C.9}$$

where the log normalisers of the gamma and beta distributions are

$$\psi_{\mathcal{G}}(a, b) = a \log b - \log \Gamma(a). \tag{C.10}$$

$$\psi_{\mathcal{B}}(s, t) = \log \Gamma(s + t) - \log \Gamma(s) - \log \Gamma(t), \tag{C.11}$$

The posterior stick breaking parameters are the same as those given by Blei and Jordan (2004).

$$a_{i,j} = a_0 + \sum_{m=1}^N \sum_{n=1}^N c_{m,n} r_{m,i} r_{n,j} y_{m,n} = [\mathbf{R}^{\top} (\mathbf{Y} \odot \mathbf{C}) \mathbf{R}]_{i,j} \tag{C.12}$$

$$b_{i,j} = b_0 + \sum_{m=1}^N \sum_{n=1}^N c_{m,n} r_{m,i} r_{n,j} = [\mathbf{R}^{\top} \mathbf{C} \mathbf{R}]_{i,j} \tag{C.13}$$

$$\gamma_{k,1} = 1 + \sum_{n=1}^N r_{n,k} \tag{C.14}$$

$$\gamma_{k,2} = \alpha + \sum_{n=1}^N \sum_{j=k+1}^K r_{n,j}. \tag{C.15}$$

Collapsed variational inference in the IRM has been investigated by Ishiguro and Sato (2014) and Ishiguro and Sato (2017), however they used the collapsed variational scheme of Teh, Newman, et al. (2007). To apply this collapsed scheme we need to compute

$$q(\mathbf{z}^{(n)}) \propto \exp \left(\mathbb{E}_{q(\mathbf{z}^{(-n)})} [\log \mathbb{E}_{p(\mathbf{V})p(\mathbf{\Lambda})} p(\mathbf{Y}, \mathbf{Z}, \mathbf{V}, \mathbf{\Lambda})] \right), \tag{C.16}$$

which is difficult unless an approximation is made. If the approximation made is $\mathbb{E}[\log x] \approx \log(\mathbb{E}[x])$, then the two collapsed bounds are the same. This method is called CVB0 in Ishiguro and Sato (2017) and they also propose an annealed

scheme in which the gradient is gradually cooled called ACVB0. Here we will use the conjugate natural gradient descent method proposed by Hensman, Rattray, et al. In this method we update the whole distribution $q(\mathbf{Z})$ in parallel, rather than each $q(\mathbf{z}^{(n)})$ in turn, as per Equation C.16.

C.3 Conjugate natural gradient descent

The only free parameters in the collapsed bound are the responsibility variables $r_{n,k}$. In principle we could perform steepest descent on these variables, however this space is not particularly convenient for optimisation since the variables are constrained to sum to 1. Instead we prefer to do optimisation in the *natural* parameter space. Let $\{r_{n,k}^*\}_{k=1}^K$ be the natural parameters of the n^{th} variational distribution, related to the expectation parameters by the *softmax* function:

$$r_{n,k} = \frac{\exp(r_{n,k}^*)}{\sum_{j=1}^K \exp(r_{n,j}^*)}. \quad (\text{C.17})$$

The gradient descent method is made much more efficient by using the Fisher Information metric giving the natural gradient. To take steps in the *natural gradient* of the *natural parameters* we simply differentiate with respect to the expectation parameters, since the Fisher information in the natural parameters is equal to a change of variables from expectation to natural parameters:

$$\tilde{\nabla}_{\mathbf{r}^*} \mathcal{L}(\mathbf{r}^*) \triangleq \mathcal{I}^{-1}(\mathbf{r}^*) \nabla_{\mathbf{r}^*} \mathcal{L}(\mathbf{r}^*) \quad (\text{C.18})$$

$$= \mathcal{I}^{-1}(\mathbf{r}^*) \left[\frac{\partial \mathbf{r}}{\partial \mathbf{r}^*} \right] \nabla_{\mathbf{r}} \mathcal{L}(\mathbf{r}) \quad (\text{C.19})$$

$$= \nabla_{\mathbf{r}} \mathcal{L}(\mathbf{r}), \quad (\text{C.20})$$

where $\tilde{\nabla}_{\mathbf{r}^*} \mathcal{L}(\mathbf{r}^*)$ is the natural gradient in the natural parameter space and $\nabla_{\mathbf{r}} \mathcal{L}(\mathbf{r})$ is the gradient in the expectation parameters. Note that $\tilde{\nabla}_{\mathbf{r}} \mathcal{L}(\mathbf{r}) = \nabla_{\mathbf{r}^*} \mathcal{L}(\mathbf{r}^*)$ also holds.

The derivatives required to implement the Collapsed IRM model are as follows:

$$\begin{aligned} \frac{\partial}{\partial r_{l,h}} \sum_{i=1}^K \sum_{j=1}^K \psi_{\mathcal{G}}(a_{i,j}, b_{i,j}) &= \sum_{i=1}^K a_{l,h}^{(0,1)} \log b_{i,h} + \frac{a_{i,h}}{b_{i,h}} b_{l,h}^{(0,1)} - \psi_1(a_{i,h}) a_{l,h}^{(0,1)} \\ &\quad + \sum_{j=1}^K a_{l,h}^{(1,0)} \log b_{h,j} + \frac{a_{h,j}}{b_{h,j}} b_{l,h}^{(1,0)} - \psi_1(a_{h,j}) a_{l,h}^{(1,0)} \end{aligned} \quad (\text{C.21})$$

$$\frac{\partial}{\partial r_{\cdot,k}} \sum_{i=1}^K \psi_{\mathcal{B}}(\gamma_{1,i}, \gamma_{2,i}) = \psi_1(\gamma_{1,k}) + \sum_{i=1}^k \psi_1(\gamma_{1,i} + \gamma_{2,i}) + \sum_{i=1}^{k-1} \psi_1(\gamma_{2,i}) \quad (\text{C.22})$$

$$\frac{\partial}{\partial r_{n,k}} \mathbb{H}_{q(\mathbf{Z})}[\mathbf{Z}] = -\log r_{n,k} - 1, \quad (\text{C.23})$$

where ψ_1 denotes the digamma function and where:

$$a_{l,h}^{(0,1)} = \sum_{m=1}^N c_{m,l} \cdot r_{m,h} \cdot y_{m,l} \quad (\text{C.24})$$

$$b_{l,h}^{(0,1)} = \sum_{m=1}^N c_{m,l} \cdot r_{m,h} \quad (\text{C.25})$$

$$a_{l,h}^{(1,0)} = \sum_{n=1}^N c_{l,n} \cdot r_{n,h} \cdot y_{l,n} \quad (\text{C.26})$$

$$b_{l,h}^{(1,0)} = \sum_{n=1}^N c_{l,m} \cdot r_{n,h}, \quad (\text{C.27})$$

To correct for the over parametrisation of the categorical distribution given by by Equation C.17 we follow Hensman (2017) and compute a correction terms. First define a new set of variables $r_{n,k}^c$ which are constrained to sum to unity:

$$r_{n,k}^c = \frac{r_{n,k}}{\sum_{j=1}^K r_{n,j}}. \quad (\text{C.28})$$

Let $\mathcal{L}_{\text{CIRM}}^{(n,k)}(r_{1,1}, \dots, r_{N,K}) \triangleq \frac{\partial \mathcal{L}_{\text{CIRM}}(r_{1,1}, \dots, r_{N,K})}{\partial r_{n,k}}$ be the $n^{\text{th}} k^{\text{th}}$ derivative of $\mathcal{L}_{\text{CIRM}}$, as a function of the variational parameters $r_{n,k}$. We now substitute the constrained variables and apply the chain rule which gives:

$$\frac{\partial \mathcal{L}_{\text{CIRM}}(r_{1,1}^c, \dots, r_{N,K}^c)}{\partial r_{n,k}} = \frac{\mathcal{L}_{\text{CIRM}}^{(n,k)}(r_{1,1}^c, \dots, r_{N,K}^c)}{\sum_{j=1}^K r_{n,j}} - \sum_{i=1}^K r_{n,i}^c \frac{\mathcal{L}_{\text{CIRM}}^{(n,i)}(r_{1,1}^c, \dots, r_{N,K}^c)}{\sum_{j=1}^K r_{n,j}}. \quad (\text{C.29})$$

If the values of the variables $r_{n,j}$ are algorithmically constrained so that $r_{n,k} := r_{n,k}^c$ then $\sum_{j=1}^K r_{n,j} = 1$ and Equation C.29 becomes:

$$\left. \frac{\partial \mathcal{L}_{\text{CIRM}}(r_{1,1}^c, \dots, r_{N,K}^c)}{\partial r_{n,k}} \right|_{r_{n,k}=r_{n,k}^c} = \mathcal{L}_{\text{CIRM}}^{(n,k)}(r_{1,1}, \dots, r_{N,K}) - \underbrace{\sum_{i=1}^K r_{n,i} \mathcal{L}_{\text{CIRM}}^{(n,i)}(r_{1,1}, \dots, r_{N,K})}_{\text{Corrects for over parameterisation}} \quad (\text{C.30})$$

which we define to be the gradient $\nabla_{\mathbf{r}}\mathcal{L}_{\text{CIRM}}(\mathbf{r}) = \tilde{\nabla}_{\mathbf{r}^*}\mathcal{L}_{\text{CIRM}}(\mathbf{r}^*)$. As recommended by Hensman, Rattray, et al. we can use non-linear conjugate gradient decent methods on the collapsed bound. In a conjugate gradient method we subtract from the current gradient the previous search direction

$$\mathbf{s}^{\{t\}} := \nabla_{\mathbf{r}}\mathcal{L}(\mathbf{r}^{\{t\}}) - \beta^{\{t\}}\mathbf{s}^{\{t-1\}}, \quad (\text{C.31})$$

weighted by β , which computed using the Fletcher–Reeves method is

$$\beta_{\text{FR}}^{\{t\}} \triangleq \frac{[\nabla_{\mathbf{r}^*}\mathcal{L}(\mathbf{r}^{*\{t\}})]^\top [\mathcal{I}(\mathbf{r}^{*\{t\}})]^{-1} \nabla_{\mathbf{r}^*}\mathcal{L}(\mathbf{r}^{*\{t\}})}{[\nabla_{\mathbf{r}^*}\mathcal{L}(\mathbf{r}^{*\{t-1\}})]^\top [\mathcal{I}(\mathbf{r}^{*\{t-1\}})]^{-1} \nabla_{\mathbf{r}^*}\mathcal{L}(\mathbf{r}^{*\{t-1\}})} \quad (\text{C.32})$$

$$\triangleq \frac{[\nabla_{\mathbf{r}^*}\mathcal{L}(\mathbf{r}^{*\{t\}})]^\top \nabla_{\mathbf{r}}\mathcal{L}(\mathbf{r}^{\{t\}})}{[\nabla_{\mathbf{r}^*}\mathcal{L}(\mathbf{r}^{*\{t-1\}})]^\top \nabla_{\mathbf{r}}\mathcal{L}(\mathbf{r}^{\{t-1\}})}. \quad (\text{C.33})$$

We can then perform a line search in the direction of $\mathbf{s}^{\{t\}}$:

$$\mathbf{r}^{*\{t\}} := \mathbf{r}^{*\{t-1\}} + l\mathbf{s}^{\{t\}}, \quad (\text{C.34})$$

where l is the step length.

References

- Adams, R., I. Murray, and D. MacKay (2009). ‘Tractable Nonparametric Bayesian Inference in Poisson Processes with Gaussian Process Intensities’. In: *International Conference on Machine Learning (ICML)*.
- Airolidi, E., D. Blei, S. Fienberg, and E. Xing (2008). ‘Mixed Membership Stochastic Blockmodels’. In: *Journal of Machine Learning Research* 9, pp. 1981–2014.
- Albers, K., A. Moth, M. Morup, and M. Schmidt (2013). ‘Large Scale Inference in the Infinite Relational Model: Gibbs Sampling is not Enough’. In: *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*.
- Amari, S. (2016). 1st ed. Vol. 194. Applied Mathematical Sciences. Springer.
- Amari, S. and H. Nagaoka (2007). *Methods of Information Geometry*. Oxford University Press.
- Ambikasaran, S. and E. Darve (2014). ‘The Inverse Fast Multipole Method’. In: *arXiv pre-print 1407.1572*.
- Ancarani, L. and G. Gasaneo (2008). ‘Derivatives of any order of the confluent hypergeometric function’. In: *Journal of Mathematical Physics* 49.6.
- Banerjee, A., J. Vogelstein, and D. Dunson (2013). ‘Parallel Inversion of Huge Covariance Matrices’. In: *arXiv pre-print 1312.1869*.
- Bardenet, R. and M. Titsias (2015). ‘Inference for Determinantal Point Processes without Spectral Knowledge’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- Basu, S. and A. Dassios (2002). ‘A Cox Process with Log-normal Intensity’. In: *Insurance: Mathematics and Economics* 31.2, pp. 297–302.
- Berger, J. (1985). *Statistical Decision Theory and Bayesian Analysis*. 2nd ed. Springer.
- Bernardo, J. M. and A. F. M. Smith (1994). *Bayesian Theory*. John Wiley & Sons, Inc.
- Bishop, C. (1999). ‘Bayesian PCA’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- (2007). *Pattern Recognition and Machine Learning*. 1st ed. Springer.
- Blei, D., A. Ng, and M. Jordan (2003). ‘Latent Dirichlet Allocation’. In: *Journal of Machine Learning Research (JMLR)* 3, pp. 993–1022.
- Blei, D. and M. Jordan (2004). ‘Variational Methods for the Dirichlet process’. In: *International Conference on Machine Learning (ICML)*. International Conference on Machine Learning (ICML).
- Botev, Z., J. Grotowski, and D. Kroese (2010). ‘Kernel Density Estimation via Diffusion’. In: *The Annals of Statistics* 38.5, pp. 2916–2957.
- Bottou, L. (1998). ‘On-line Learning and Stochastic Approximations’. In: *On-line Learning in Neural Networks*. Cambridge University Press, pp. 9–42.
- Box, G. and N. R. Draper (1987). *Empirical Model-building and Response Surfaces*. Vol. 424. Wiley New York.
- Broderick, T., N. Boyd, A. Wibisono, A. Wilson, and M. Jordan (2013). ‘Streaming Variational Bayes’. In: *Advances in Neural Information Processing Systems (NIPS)*.

- Bui, T. and R. Turner (2015). ‘Stochastic Variational Inference for Gaussian Process Latent Variable Models using Back Constraints’. In: *Black Box Learning and Inference, NIPS workshop*.
- Byrd, R., P. Lu, J. Nocedal, and C. Zhu (1995). ‘A Limited Memory Algorithm for Bound Constrained Optimization’. In: *SIAM Journal on Scientific Computing* 16.5, pp. 1190–1208.
- Cai, D., T. Campbell, and T. Broderick (2016). ‘Edge-exchangeable Graphs and Sparsity’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- Carlin, B., A. Gelfand, and A. F. Smith (1992). ‘Hierarchical Bayesian Analysis of Change-point Problems’. In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 41.2, pp. 389–405.
- Caron, F. (2012). ‘Bayesian Nonparametric Models for Bipartite Graphs’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- Caron, F. and E. Fox (2014). ‘Sparse Graphs using Exchangeable Random Measures’. In: Cemgil, A. (2009). ‘Bayesian Inference for Non-negative Matrix Factorisation Models’. In: *Computational Intelligence and Neuroscience*.
- Chu, W. and Z. Ghahramani (2009). ‘Probabilistic Models for Incomplete Multi-dimensional Arrays’. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.
- Cichocki, A., R. Zdunek, and S.-i. Amari (2006). ‘Csiszar’s Divergences for Non-negative Matrix Factorization: Family of New Algorithms’. In: *International Conference on Independent Component Analysis and Blind Signal Separation*, pp. 32–39.
- Cunningham, J., K. Shenoy, and M. Sahani (2008). ‘Fast Gaussian Process Methods for Point Process Intensity Estimation’. In: *International conference on Machine learning (AISTATS)*, pp. 192–199.
- Daley, D. J. and D. Vere-Jones (2003). *An Introduction to the Theory of Point Processes Volume I: Elementary Theory and Methods*. Second. Springer.
- Dhillon, I. and S. Sra (2005). ‘Generalized Nonnegative Matrix Approximations with Bregman Divergences’. In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 283–290.
- Dieng, A., D. Tran, R. Ranganath, J. Paisley, and D. Blei (2016). ‘The Chi-Divergence for Approximate Inference’. In: *arXiv pre-print 1611.00328*.
- Diggle, P. (1985). ‘A Kernel Method for Smoothing Point Process Data’. In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 34.2, pp. 138–147.
- Diggle, P., B. Rowlingson, and T. Su (2005). ‘Point Process Methodology for On-line Spatio-temporal Disease Surveillance’. In: *Environmetrics* 16.5, pp. 423–434.
- Donoho, D. and V. Stodden (2003). ‘When Does Non-Negative Matrix Factorization Give Correct Decomposition into Parts?’ In: *Advances in Neural Information Processing Systems (NIPS)*.
- Duane, S., A. D. Kennedy, B. J. Pendleton, and D. Roweth (1987). ‘Hybrid Monte Carlo’. In: *Physics Letters B* 195.2, pp. 216–222.
- DuBois, C., C. Butts, and P. Smyth (2013). ‘Stochastic Blockmodeling of Relational Event Dynamics’. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.
- Durante, D. and D. Dunson (2014). ‘Bayesian Logistic Gaussian Process Models for Dynamic Networks’. In: *International Conference on Machine Learning (ICML)*.

- Durrant-Whyte, H., B. Rao, and H. Hu (1991). ‘Toward a Fully Decentralized Architecture for Multi-sensor Data Fusion’. In: *IEE Colloquium on Principles and Applications of Data Fusion*.
- Eisenbaum, N. and H. Kaspi (2009). ‘On permanent processes’. In: *Stochastic Processes and their Applications* 119.5, pp. 1401–1415.
- Favaro, S. and Y. W. Teh (2013). ‘MCMC for Normalized Random Measure mixture models’. In: *Statistical Science* 28, pp. 335–359.
- Fearnhead, P. (2006). ‘Exact and Efficient Bayesian Inference for Multiple Change-point Problems’. In: *Statistics and Computing* 16.2, pp. 203–213.
- Ferguson, T. S. (1973). ‘A Bayesian Analysis of Some Nonparametric Problems’. In: *The Annals of Statistics* 1.2, pp. 209–230.
- Fernandez, T., N. Rivera, and Y. W. Teh (2016). ‘Gaussian Processes for Survival Analysis’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- Févotte, C. and J. Idier (2011). ‘Algorithms for Nonnegative Matrix Factorization with the Beta-divergence’. In: *Neural Computation* 23.9, pp. 2421–2456.
- Flaxman, S., W. Y. Teh, and D. Sejdinovic (2017). ‘Poisson Intensity Estimation with Reproducing Kernels’. In: *A Dynamic Relational Infinite Feature Model for Longitudinal Social Networks*.
- Flaxman, S., A. Wilson, D. Neill, H. Nickisch, and A. Smola (2015). ‘Fast Kronecker Inference in Gaussian Processes with non-Gaussian Likelihoods’. In: *International Conference on Machine Learning (ICML)*.
- Fouldsa, J., A. Asuncion, C. DuBois, C. Butts, and P. Smyth (2011). ‘A Dynamic Relational Infinite Feature Model for Longitudinal Social Networks’. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.
- Fu, W., L. Song, and E. Xing (2009). ‘Dynamic Mixed Membership Block Model for Evolving Networks’. In: *International Conference on Machine Learning (ICML)*.
- Gemmetto, V., A. Barrat, and C. Cattuto (2014). ‘Mitigation of Infectious Disease at School: Targeted Class Closure vs School Closure’. In: *BMC Infectious Diseases* 14.1, p. 695.
- Ghahramani, Z. and M. Beal (2001). ‘Propagation Algorithms for Variational Bayesian Learning’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- Ghahramani, Z. (2013). ‘Bayesian Non-parametrics and the Probabilistic Approach to Modelling’. In: *Philosophical Transactions of the Royal Society A* 371.1984.
- Ghahramani, Z. and G. Hinton (2000). ‘Variational Learning for Switching State-Space Models’. In: *Neural Computation* 12, pp. 831–64.
- Girolami, M. and B. Calderhead (2011). ‘Riemann Manifold Langevin and Hamiltonian Monte Carlo Methods’. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.2, pp. 123–214.
- Granovetter, M. (1973). ‘The Strength of Weak Ties’. In: *American Journal of Sociology* 78.6, pp. 1360–1380.
- Griffiths, T. L. and Z. Ghahramani (2006). ‘Infinite Latent Feature Models and the Indian Buffet Process’. In: *Advances in Neural Information Processing Systems* 18.
- Gunter, T., C. Lloyd, M. A. Osborne, and S. J. Roberts (2014). ‘Efficient Bayesian Nonparametric Modelling of Structured Point Processes’. In: *International Conference on Uncertainty in Artificial Intelligence (UAI)*.
- Gunter, T., M. A. Osborne, R. Garnett, P. Hennig, and S. Roberts (2014). ‘Sampling for Inference in Probabilistic Models with Fast Bayesian Quadrature’. In: *Advances in Neural Information Processing Systems (NIPS)*.

- Hensman, J., N. Fusi, and N. Lawrence (2013). ‘Gaussian Processes for Big Data’. In: *International Conference on Uncertainty in Artificial Intelligence (UAI)*.
- Hensman, J., M. Rattray, and N. Lawrence (2012). ‘Fast Variational Inference in the Conjugate Exponential Family’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- Hensman, J. (2017). *Github repositories*. <http://github.com/jameshensman/>. [Online; accessed 21-March-2017].
- Hensman, J., N. Durrande, and A. Solin (2016). ‘Variational Fourier Features for Gaussian Processes’. In: *arXiv pre-print 1611.06740*.
- Hensman, J., A. Matthews, M. Filippone, and Z. Ghahramani (2015). ‘MCMC for Variationally Sparse Gaussian Processes’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- Herlau, T., M. Schmidt, and M. Mørup (2016). ‘Completely Random Measures for Modelling Block-structured Sparse Networks’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- Hinton, G. E. and D. van. Camp (1993). ‘Keeping the Neural Networks Simple by Minimizing the Description Length of the Weights’. In: *Conference on Computational Learning Theory (COLT)*, pp. 5–13.
- Hoff, P., A. Raftery, and M. Handcock (2002). ‘Latent Space Approaches to Social Network Analysis’. In: *Journal of the American Statistical Association* 97, pp. 1090–1098.
- Hoff, P. (2015). ‘Multilinear Tensor Regression for Longitudinal Relational Data’. In: 3, pp. 1169–1193.
- Hoffman, M., D. Blei, and P. Cook (2010). ‘Bayesian Nonparametric Matrix Factorization for Recorded Music’. In: *International Conference on Machine Learning (ICML)*, pp. 439–446.
- Hough, J., M. Krishnapur, Y. Peres, and B. Virág (2006). ‘Determinantal Processes and Independence’. In: *Probability Surveys*.
- Ioannis, P., S. Roberts, I. Rezek, and B. Sheldon (2012). ‘Inferring Social Network Structure in Ecological Systems from Spatio-temporal Data Streams’. In: *Journal of The Royal Society Interface* 9.76, pp. 3055–3066.
- Ishiguro, K., T. Iwata, N. Ueda, and J. Tenenbaum (2010). ‘Dynamic Infinite Relational Model for Time-varying Relational Data Analysis’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- Ishiguro, K. and I. Sato (2014). ‘Collapsed Variational Bayes Inference of Infinite Relational Model’. In: *arXiv pre-print 1409.4757*.
- (2017). ‘Averaged Collapsed Variational Bayes Inference’. In: *Journal of Machine Learning Research (JMRL)* 18, pp. 1–29.
- Iwata, T., A. Shah, and Z. Ghahramani (2013). ‘Discovering Latent Influence in Online Social Activities via Shared Cascade Poisson Processes’. In: *KDD*.
- Jarrett, R. G. (1979). ‘A note on the intervals between coal-mining disasters’. In: *Biometrika* 66.1, pp. 191–193.
- Jaynes, E. T. (2003). *Probability Theory: The Logic of Science*. Cambridge University Press.
- Jewell, S., N. Spencer, and A. Bouchard-Cote (2015). ‘Atomic Spatial Processes’. In: *International Conference on Machine Learning (ICML)*.

- Julier., S. and J. Uhlmann (1997). ‘New Extension of the Kalman Filter to Nonlinear Systems’. In: *Proceedings of the 1997 SPIE Conference on Signal Processing, Sensor Fusion, and Target Recognition*.
- Karrer, B. and M. Newman (2011). ‘Stochastic Blockmodels and Community Structure in Networks’. In: *Physical Review E* 83, p. 016107.
- Kemp, C., J. Tenenbaum, T. Griffiths, T. Yamada, and N. Ueda (2006). ‘Learning Systems of Concepts with an Infinite Relational Model’. In: *21st National Conference on Artificial Intelligence*. Vol. 1, pp. 381–388.
- Kim, M. and J. Leskovec (2013). ‘Nonparametric Multi-group Membership Model for Dynamic Networks’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- King, N. J. and N. Lawrence (2006). ‘Fast Variational Inference for Gaussian Process Models Through KL-Correction’. In: *European Conference on Machine Learning (ECML)*.
- Kingma, D. and J. Ba (2015). ‘Adam: A Method for Stochastic Optimization’. In: *International Conference for Learning Representations*.
- Kingma, D. and M. Welling (2013). ‘Auto-Encoding Variational Bayes’. In: *International Conference on Learning Representations (ICLR)*.
- Kingman, J. F. C. (1993). *Poisson Processes*. Oxford University Press.
- Kolda, T. and J. Sun (2008). ‘Scalable Tensor Decompositions for Multi-aspect Data Mining’. In: *IEEE International Conference on Data Mining*, pp. 363–372.
- Kom Samo, Y. L. and S. J. Roberts (2015). ‘Scalable Nonparametric Bayesian Inference on Point Processes with Gaussian Processes’. In: *International Conference on Machine Learning (ICML)*.
- Kroonenberg, P. (1983). *Three-mode Principal Component Analysis: Theory and Applications*. DSWO Press, Leiden.
- Kruskal, J. (1977). ‘Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics’. In: *Linear Algebra and its Applications* 18.2, pp. 95–138.
- Laskey, K. B. and S. Leinhardt (1983). ‘Stochastic Blockmodels: First Steps’. In: *Social Networks* 5, pp. 109–137.
- Lasko, T. (2014). ‘Efficient Inference of Gaussian-Process-Modulated Renewal Processes with Application to Medical Event Data’. In: *International Conference on Uncertainty in Artificial Intelligence (UAI)*.
- Lathauwer, L., B. Moor, and J. Vandewalle (2000). ‘A multilinear singular value decomposition’. In: *SIAM Journal of Matrix Analysis and Applications* 21.4, pp. 1253–1278.
- Lawrence, N. (2003). ‘Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- Lee, D. and S. Seung (2001). ‘Algorithms for Non-negative Matrix Factorization’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- Lewis, P. and G. Shedler (1979). ‘Simulation of a Nonhomogeneous Poisson Process by Thinning.’ In: *Naval Research Logistics Quarterly* 26, pp. 403–413.
- Lian, W., V. Rao, B. Eriksson, and L. Carin (2014). ‘Modeling Correlated Arrival Events with Latent Semi-Markov Processes’. In: *International Conference on Machine Learning (ICML)*.
- Lin, Y.-R., H. Sundaram, and A. Kelliher (2010). ‘Multi-Relational Characterization of Dynamic Social’. In: *The Handbook of Social Network Technologies and Applications*. Ed. by B. Furht. Springer. Chap. 18, pp. 45–58.

- Linderman, S. and R. Adams (2014). ‘Discovering Latent Network Structure in Point Process Data’. In: *International Conference on Machine Learning (ICML)*.
- Liu, D. C. and J. Nocedal (1989). ‘On the Limited Memory BFGS Method for Large Scale Optimization’. In: *Mathematical Programming* 45.3, pp. 503–528.
- Lloyd, C. (2013). *Transfer of Status Report*. Tech. rep. Department of Engineering Science, University of Oxford.
- Lloyd, C., T. Gunter, T. Nickson, M. A. Osborne, and S. J. Roberts (2016). ‘Latent Poisson Process Allocation’. In: *Conference on Artificial Intelligence and Statistics (AISTATS)*.
- Lloyd, C., T. Gunter, M. A. Osborne, and S. J. Roberts (2015a). ‘Inferring Dynamic Interaction Networks with N-LPPA’. In: *Networks in the social and information sciences, NIPS workshop*.
- (2015b). ‘Variational Inference for Gaussian Process Modulated Poisson Processes’. In: *International Conference on Machine Learning (ICML)*.
- Lloyd, J., P. Orbanz, Z. Ghahramani, and D. Roy (2012). ‘Random Function Priors for Exchangeable Arrays with Applications to Graphs and Relational Data’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- MacKay, D. (1995). ‘Ensemble Learning and Evidence Maximization’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- (2005). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.
- Malaria Atlas Project (2017). <http://www.map.ox.ac.uk/explore/data-modelling/>. [Online; accessed 21-March-2017].
- Matias, C., T. Rebafka, and F. Villers (2016). ‘A Semi-parametric Extension of the Stochastic Block Model for Longitudinal Networks’. In: *Hal archives pre-print hal-01245867*.
- Matthews, A., J. Hensman, R. Turner, and Z. Ghahramani (2016). ‘On Sparse Variational Methods and the Kullback-Leibler Divergence between Stochastic Processes’. In: *Conference on Artificial Intelligence and Statistics (AISTATS)*.
- Mauricio, A. and N. Lawrence (2011). ‘Computationally Efficient Convolved Multiple Output Gaussian Processes’. In: *Journal of Machine Learning Research* 12, pp. 1459–1500.
- McCullagh, P. and J. Moller (2006). ‘The permanental process’. In: *Advances in Applied Probability* 38.4, pp. 873–888.
- Meeds, E., Z. Ghahramani, R. Neal, and S. Roweis (2007). ‘Modeling Dyadic Data with Binary Latent Factors’. In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 977–984.
- Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller (1953). ‘Equations of State Calculations by Fast Computing Machines’. In: *Journal of Chemical Physics* 24, pp. 1087–1092.
- Miller, A., L. Bornn, R. Adams, and K. Goldsberry (2014). ‘Factorized Point Process Intensities: A Spatial Analysis of Professional Basketball’. In: *International Conference on Machine Learning (ICML)*.
- Miller, K., T. Griffiths, and M. I. Jordan (2009). ‘Nonparametric Latent Feature Models for Link Prediction’. In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 1276–1284.
- Minka, T. (2001). ‘Expectation Propagation for approximate Bayesian Inference’. In: *In Advances in Neural Information Processing Systems (NIPS)*.

- Mohamed, S., K. Heller, and Z. Ghahramani (2009). ‘Bayesian Exponential Family PCA’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- Moller, J., A. Syversveen, and R. Waagepetersen (1998). ‘Log Gaussian Cox Processes’. In: *Scandinavian Journal of Statistics* 25.3, pp. 451–482.
- Murray, I., R. Adams, and D. MacKay (2010). ‘Elliptical Slice Sampling’. In: *Journal of Machine Learning Research (JMRL)* 9, pp. 541–548.
- Neal, R. (2010). *Handbook of Markov Chain Monte Carlo*. Ed. by S. Brooks, A. Gelman, G. Jones, and X. L. Meng. Chapman and Hall.
- Neal, R. (2003). ‘Slice Sampling’. In: *Annals of Statistics* 31.3, pp. 705–767.
- Nickson, T., T. Gunter, C. Lloyd, M. A. Osborne, and S. J. Roberts (2015). ‘Blitzkriging: Kronecker-structured Stochastic Gaussian Processes’. In: <http://arxiv.org/abs/1510.07965>.
- Orbanz, P. and D. Roy (2015). ‘Bayesian Models of Graphs, Arrays and Other Exchangeable Random Structures’. In: *IEEE transactions on pattern analysis and machine intelligence* 37.2, pp. 437–461.
- Palla, K., D. Knowles, and Z. Ghahramani (2012). ‘An Infinite Latent Attribute Model for Network Data’. In: *International Conference on Machine Learning (ICML)*.
- Palla, K., F. Caron, and Y. W. Teh (2016). ‘Bayesian nonparametrics for Sparse Dynamic Networks’. In: *arXiv pre-print 1607.01624*.
- Pearl, J. (2000). *Causality: Models, Reasoning, and Inference*. Cambridge University Press.
- Perry, P. and P. Wolfe (2013). ‘Point Process Modelling for Directed Interaction Networks’. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 75.5, pp. 821–849.
- Pfingsten, T., M. Kuss, and C. Rasmussen (2006). ‘Nonstationary Gaussian Process Regression using a Latent Extension of the Input Space’. In: *Eighth World Meeting on Bayesian Statistics*.
- Psorakis, I., S. J. Roberts, M. Ebden, and B. Sheldon (2011). ‘Overlapping Community Detection using Bayesian Non-negative Matrix Factorization’. In: *Physical Review E* 83.6.
- Quiñonero-Candela, J. and C. Rasmussen (2005). ‘A Unifying View of Sparse Approximate Gaussian Process Regression’. In: *Journal of Machine Learning Research (JMRL)* 6, pp. 1939–1959.
- Rakitsch, B., C. Lippert, K. Borgwardt, and O. Stegle (2013). ‘It is all in the Noise: Efficient Multi-task Gaussian Process Inference with Structured Residuals’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- Rao, V. and Y. W. Teh (2011). ‘Gaussian Process Modulated Renewal Processes’. In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 2474–2482.
- Rasmussen, C. E. and C. K. I. Williams (2006). *Gaussian Processes for Machine Learning*. 2nd. The MIT Press.
- Rezende, D., S. Mohamed, and D. Wierstra (2014). ‘Stochastic Backpropagation and Approximate Inference in Deep Generative Models’. In: *International Conference on Machine Learning (ICML)*.
- Roweis, S. and Z. Ghahramani (1999). ‘A Unifying Review of Linear Gaussian Models’. In: *Neural Computation* 11.2, pp. 305–345.
- Rue, H., S. Martino, and N. Chopin (2009). ‘Approximate Bayesian Inference for Latent Gaussian Models by Using Integrated Nested Laplace Approximations’. In: *Journal of the Royal Statistical Society, Series B* 71.2, pp. 319–392.

- Saul, A., J. Hensman, A. Vehtari, and N. Lawrence (2016). ‘Chained Gaussian Processes’. In: *Conference on Artificial Intelligence and Statistics (AISTATS)*.
- Schein, A., J. Paisley, D. Blei, and H. Wallach (2015). ‘Bayesian Poisson Tensor Factorization for Inferring Multilateral Relations from Sparse Dyadic Event Counts’. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’15)*.
- Schmidt, M. and S. Mohamed (2009). ‘Probabilistic Non-negative Tensor factorization using Markov Chain Monte Carlo’. In: *17th European Signal Processing Conference (EUSIPCO)*.
- Shashua, A. and T. Hazan (2005). ‘Non-negative tensor factorization with applications to statistics and computer vision’. In: *International Conference on Machine Learning (ICML)*, pp. 792–799.
- Simma, A. and M. Jordan (2010). ‘Modeling Events with Cascades of Poisson Processes’. In: *International Conference on Uncertainty in Artificial Intelligence (UAI)*.
- Simpson, D., J. B. Illian, F. Lindgren, S. H. Sorbye, and H. Rue (2016). ‘Going off Grid: Computationally Efficient Inference for Log-Gaussian Cox Processes’. In: *Biometrika* 103.1, pp. 49–70.
- Sizov, S., S. Staab, and T. Franz (2010). ‘Analysis of Social Networks by Tensor Decomposition’. In: *The Handbook of Social Network Technologies and Applications*. Ed. by B. Furht. Springer. Chap. 3, pp. 45–58.
- Snelson, E. and Z. Ghahramani (2005). ‘Sparse Gaussian Processes using Pseudo-inputs’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- Tan, V. and C. Fevotte (2009). ‘Automatic Relevance Determination in Nonnegative Matrix Factorization’. In: *Signal Processing with Adaptive Sparse Structured Representation (SPARS)*.
- (2013). ‘Automatic Relevance Determination in Nonnegative Matrix Factorization with the Beta-divergence’. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Taylor, B. and P. Diggle (2014). ‘INLA or MCMC?: A Tutorial and Comparative Evaluation for Spatial Prediction in Log-Gaussian Cox Processes’. In: *Journal of Statistical Computation and Simulation* 84.10, pp. 2266–2284.
- Teh, Y. W., D. Newman, and M. Welling (2007). ‘A Collapsed Variational Bayesian Inference Algorithm for Latent Dirichlet Allocation’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- Teh, Y. W., M. Seeger, and M. Jordan (2005). ‘Semiparametric Latent Factor Models’. In: *Proceedings of the International Workshop on Artificial Intelligence and Statistics*. Vol. 10.
- Tipping, M. and C. Bishop (1997). ‘Probabilistic Principle Component Analysis’. In: *Journal of the Royal Statistical Society* 61.3, pp. 611–622.
- Titsias, M. (2009). ‘Variational Learning of Inducing Variables in Sparse Gaussian Processes’. In: *Conference on Artificial Intelligence and Statistics (AISTATS)*.
- Titsias, M. and N. Lawrence (2010). ‘Bayesian Gaussian Process Latent Variable Model’. In: *Conference on Artificial Intelligence and Statistics (AISTATS)*.
- van der Vaart, A. W. (1998). *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.
- Walder, C. and A. Bishop (2017). ‘Fast Bayesian Intensity Estimation for the Permanental Process’. In: *International Conference on Machine Learning (ICML)*.

- Welling, M. and M. Weber (2001). ‘Positive Tensor Factorization’. In: *Pattern Recognition Letters* 22.12, pp. 1255–1261.
- Wilson, A., E. Gilboa, A. Nehorai, and J. Cunningham (2014). ‘Fast Kernel Learning for Multidimensional Pattern Extrapolation’. In: *Advances in Neural Information Processing Systems (NIPS)*.
- Wilson, A. and R. Adams (2013). ‘Gaussian Process Kernels for Pattern Discovery and Extrapolation’. In: *International Conference on Machine Learning (ICML)*.
- Xin, L., M. Zhu, and H. Chipman (2015). ‘A Continuous-Time Stochastic Block Model for Basketball Networks’. In: *Annals of Statistics*.