Department of Electrical and Electronic Engineering
Imperial College London

DIMITRIOS KORKINOF

PROBABILISTIC LEARNING BY DEMONSTRATION FROM COMPLETE AND INCOMPLETE DATA

Applications in motion trajectory learning and sequence completion.

Submitted in part fulfilment of the requirements for the degree of Doctor of Philosophy in Electrical and Electronic Engineering of Imperial College London, September 2014.

Probabilistic Learning by Demonstration from Complete and Incomplete Data
Applications in motion trajectory learning and sequence completion.

Dedicated to my parents Vangelitsa and Miroslav...

DECLARATION

I hereby declare that I composed this thesis entirely by myself and that it describes my own research.

London , September 2014

Dimitrios Korkinof

ABSTRACT

In recent years we have observed a convergence of the fields of robotics and machine learning initiated by technological advances bringing AI closer to the physical world. A prerequisite, however, for successful applications is to formulate reliable and precise offline algorithms, requiring minimal tuning, fast and adaptive online algorithms and finally effective ways of rectifying corrupt demonstrations. In this work we aim to address some of those challenges.

We begin by employing two offline algorithms for the purpose of Learning by Demonstration (LbD). A Bayesian non-parametric approach, able to infer the optimal model size without compromising the model's descriptive power and a Quantum Statistical extension to the mixture model able to achieve high precision for a given model size. We explore the efficacy of those algorithms in several one- and multi-shot LbD application achieving very promising results in terms of speed and and accuracy.

Acknowledging that more realistic robotic applications also require more adaptive algorithmic approaches, we then introduce an online learning algorithm for quantum mixtures based on the online EM. The method exhibits high stability and precision, outperforming well-established online algorithms, as demonstrated for several regression benchmark datasets and a multi-shot trajectory LbD case study.

Finally, aiming to account for data corruption due to sensor failures or occlusions, we propose a model for automatically rectifying damaged sequences in an unsupervised manner. In our approach we take into account the sequential nature of the data, the redundancy manifesting itself among repetitions of the same task and the potential of knowledge transfer across different tasks. We have devised a temporal factor

model, with each factor modelling a single basic pattern in time and collectively forming a dictionary of fundamental trajectories shared across sequences. We have evaluated our method in a number of real-life datasets.

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude and respect for my advisor Dr. Yiannis Demiris. He was the one who introduced me to Machine Learning and I believe that this single event irreversibly changed the course of my life as it led me to discover the field that I would later become so passionate about. Moreover, his guidance, advice and personal support were absolutely fundamental for this work, but more importantly for the skill-set and experience I gained during the course of my Ph.D. and which will remain with me for the rest of my life.

I would like to thank my parents Vangelitsa and Miroslav, for their unconditional love and support, without which none of this would be possible. Thank you for always being there for me, in good and in difficult times. Thank you for your psychological and financial support. Finally, thank you for enduring my relentless pessimism.

To Venetia, thank you for your love, care and support. Thank you for trying to make sense of what I'm talking about most of the time. Thank you for your patience, as I am not the easiest person in the world to get along with.

To my colleagues and friends in the Personal Robotics laboratory: It is rare privilege to work with people that you can also call friends, but I have been very fortunate in that regard. I feel richer for knowing you all.

So thank you Ayse for being so supportive, for listening to crazy concepts and for your valuable advice. Thank you Harold for our amazing discussions, thank you Yan and Miguel for "the good old times and some new". Thank you Yixing and Theo for all the jokes and laughs we had together, which always cheered me up, even in the gloomiest and rainiest of days. Finally, a special thanks to Kyuhwa for

generously sharing his motion capture dataset with me. Also, thank you Arturo, Martina, Hyung Jin, Yanyu, Alex, Eris, Maxime, Jonathan, Dimitri, Simon, Balint, Murilo and Raquel.

Last but not least, to my friends, thank you all for the coffees and beers we drank together; thank you for the time we spent together and the interesting conversations about everything and nothing.

This work was supported by the State Scholarship Foundation of Greece (IKY) through the EU NSRF 2007-2013.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AI Artificial Intelligence

AIC Akaike Information Criterion

ARD Automatic Relevance Determination

AR Auto-regressive

BIC Bayesian Information Criterion

BP Beta Process

CD Canonical Decomposition

DILN Discrete Infinite Logistic Normal

DoF Degrees of Freedom

DP Dirichlet Process

DPM Dirichlet Process Mixture

DPMR Dirichlet Process Mixture Regression

DTW Dynamic Time Warping

DS Dynamical Systems

EM Expectation Maximisation

EP Expectation Propagation

ESS Elliptical Slice Sampling

FA Factor Analyser

FIM Fisher Information Matrix

FPC Fixed Point Completion

GMM Gaussian Mixture Model

GMR Gaussian Mixture Regression

GP Gaussian Process

GPC Gaussian Process Classification

GPDS Gaussian Process Dynamical Systems

GPDM Gaussian Process Dynamical Models

GPLVM Gaussian Process Latent Variable Model

GPR Gaussian Process Regression

HDP Hierarchical Dirichlet Process

HMC Hamiltonian Monte Carlo

HMM Hidden Markov Model

HMRF Hidden Markov Random Field

IS Importance Sampling

IBP Indian Buffet Process

ibpGP Indian buffet process Gaussian Process

KL Kullback-Leibler

LbD Learning by Demonstration

LDA Latent Dirichlet Allocation

LDS Linear Dynamical Systems

LGPR local Gaussian Process Regression

LFM Latent Force Models

LHR Left Hand Rotation

LWPR Locally Weighted Projection Regression

MAP Maximum a-Posteriori

MCMC Markov Chain Monte Carlo

MFCC Mel-frequency Cepstrum Coefficients

MGP Multiplicative Gamma Process

mGP multi-kernel Gaussian Process

MLE Maximum Likelihood Estimation

MoCap Motion Capture

MRF Markov Random Field

MSE Mean Square Error

MTL Milti-task learning

NGN Normalised Gaussian Network

ODE Ordinary Differential Equation

oGMM online Gaussian Mixture Model

oGP online Gaussian Process

oQMM online Quantum Mixture Model

oSMM online Student-t Mixture Model

PF Particle Filtering

PYP Pitman-Yor Process

QMM Quantum Mixture Model

QMR Quantum Mixture Regression

RMSE Root Mean Square Error

SMC Sequential Monte Carlo

STL Single-task learning

SVT Singular Value Thresholding

SVM Support Vector Machine

SS-GP spike and slab Gaussian Process

SMM Student-t Mixture Model

SLDS Switching Linear Dynamical Systems

RL Reinforcement Learning

VB Variational Bayesian

VBMC Variational Bayesian Matrix Completion

1

INTRODUCTION

The purpose of this chapter is threefold: We first wish to broadly introduce the fields of robotics and machine learning, as well as their coevolution during the past few years. This part may be safely skipped by readers already familiar with both fields. Second, we wish to present the motivating factors that initiated this research and the choices we made along the way. Finally, we present an outline of this thesis and its contributions from both a theoretical and a practical perspective, with the purpose of facilitating the reading of this thesis.

1.1 INTRODUCTION

During the past two decades we have seen a technological revolution in intelligent systems, artificial intelligence and machine learning, three terms that literally speaking refer to very similar but nevertheless diverse fields. The need for intelligent algorithms was recognised many years ago, when the foundations of several relevant mathematical fields were laid; it was made practically possible only after the advent of computer systems and has now (in the past decade or so) infiltrated our everyday lives attracting a huge amount of interest by the scientific community.

Not surprisingly, the advent of computer intelligence began in the "virtual world" and, more importantly, online. Google's search engine revolutionised the web

by employing a simple notion that lies at the core of the concept of intelligent machines, learning by example; more specifically, harvesting human knowledge in the web and thus helping us find the webpages we are looking for. More recently, we have seen many examples of computer vision, speech recognition and recommender systems (i.e. opponent matching) used in a large variety of computer applications, smartphones, cameras and gaming consoles.

The field of robotics is the receptor of all those technologies aiming to translate them into the physical world. Robotics is the ultimate step for machine intelligence, breaching the limits of the virtual world and becoming an active physical entity. Of course, the theoretical and practical adversities of such an endeavour are daunting and for many years the technological advances were simply insufficient to make it possible.

Prior to addressing the challenges posed by robotic agents, scientists need to utilise solutions to problems that are subject to rigorous research, stem from a variety of disciplines and, even today, remain open. More specifically, a physical AI would most certainly rely upon:

- Advanced computer computer vision capabilities.
- A variety of reliable sensors.
- Accurate and reliable mechanical systems.
- Compact, but powerful enough hardware.

Most of those requirements remain challenging, but just a few years ago used to pose an almost impossible to breach barrier for robotics. As an example we could refer to Boston Dynamics, who have recently managed, with significant effort, increase the mean-time-between-failures of their mechanical systems from 3 minutes to around 3 hours.

We can speculate that these challenges may be the reason that robotics research became decoupled from research in machine intelligence, although they are funda-

mentally so profoundly inseparable. However, the technological gap between the two fields was so large that it used to be practically impossible to bridge, leading researchers to consider the two problems separately.

As is usually the case, research precedes technology and we are now very fortunate to live in an exciting time where the science fiction is gradually becoming everyday reality. The simplest intelligent robotic applications are already widely available and affordable, such as toy robots and robotic vacuum cleaners. But more excitingly, huge advances are at the verge of becoming commercially available. For instance, Google's fully autonomous car and Boston Dynamics quadruped robots would look like fiction just a few years ago.

From an academic perspective, the aforementioned technological advances have triggered a profound change. They have gradually began to reestablish the long lost link between machine intelligence and robotics, in the sense that modern Machine Learning algorithms find use in real-life applications with physical agents.

1.2 MOTIVATION

In this section we briefly present the sources of our inspiration for the research presented in this thesis.

WHY MACHINE LEARNING? It is peculiar that computers are so brilliant in playing chess, even beating the world's best chess player, and yet they are unable to recognise what we are saying; a toddler is better in recognising everyday objects than the world's fastest supercomputer. The fundamental reason that computers are so effective in chess is that we can find a solution to the optimal next move by applying a set of logical rules. This solution is exact and precise, it just requires raw computational power.

Logic has played an important part in computer intelligence throughout the years. However, it is not applicable to all situations. For instance, let us consider a basic pattern recognition problem, such as distinguishing cats and dogs from some given images. The first approaches to solving this problem in the 70's were to employ expert systems consisted of a set of logical rules in order to reach a decision. That is indeed a natural choice, attempting to imprint human expertise into a machine. For instance, we can argue that a cat is usually smaller than a dog, or a cat usually has shorter fur. Scientists, however, quickly realised that for every rule they could think of, there was an exception and on top of that the whole process of hard-coding, hand-crafted rules into a machine was a tedious and time consuming task. The problem was simply too complicated to be modelled by a set of rules and that is the reason that rule-based systems, although performing very well in some tasks, have been largely abandoned for pattern recognition applications. Machine learning techniques gave a solution to this problem, by accounting for the innate uncertainty and enabling the computer to extract the underlying pattern from the example data and then being able to generalise the acquired knowledge in difference circumstances.

Our world is ever-changing, imprecise, full of uncertainty and exceptions. There are phenomena that we cannot fully explain in analytical form or with rule-based systems, because they are simply beyond our reach in terms of complexity.

WHY LEARNING BY DEMONSTRATION? Another instance of this vast complexity is the so called "curse of dimensionality", defined in [Bellman, 1961] as "*the severe difficulty that can arise in spaces of many dimensions*".

For instance, let us consider a NAO robot academic edition with 25 degrees-of-freedom [Gouaillier et al., 2009] and for simplicity we shall assume that each joint motor requires just 3 different commands (forwards, backwards and zero). There are $3^{25} \simeq 847 \cdot 10^9$ possible actions at every moment and for an iCub with 54 degrees-of-freedom [Sandini et al., 2007] the number of possible actions is even larger, $3^{54} \simeq 5.8 \cdot 10^{25}$. Of course, searching such vast action spaces for a good

solution, by means of Reinforcement Learning (RL) for instance, may be a very challenging and time-consuming problem (for a broader discussion on this subject we refer to Chapter 3).

In this case, one or more demonstrations might facilitate our effort, by providing the areas of interest in this vast multidimensional space and the constrains of the skill we wish to learn. For example, with multiple demonstrations in our possession we would typically be able to identify areas of motion where precision is crucial and others where we need not be so precise. We would also be able identify acceptable variations of the motion in question.

WHY MACHINE LEARNING IN ROBOTICS? As we have mentioned before, robotics became decoupled from AI due technological adversities. For the same reason Machine Learning, as a relatively new field within AI, evolved as an autonomous field with sparse connections to the fields of potential application, including but not limited to robotics.

The field has revolved for a long time around a limited set of benchmark datasets and as is maturing in the past few years has received a great deal of criticism for neglecting the importance of practical applicability. On that account, we believe that impactful contributions can only be achieved by closing the feedback loop between theory and application. We also believe that this is a viable aspiration given the latest technological developments.

WHY MIXTURE MODELS? Gaussian Mixture Model (GMM), along with HMMs, were two of the first statistical models to be used in trajectory LbD applications, i.e. in [Calinon & Billard, 2007; Calinon, 2008, 2007]. Throughout the years the model has proven its resilience by being used:

- In a large variety of different settings [Malekzadeh et al., 2013; Calinon et al., 2014; Kormushev et al., 2011; Rozo et al., 2014]

- In interesting medical applications [Mylonas et al., 2013]
- In conjunction with Reinforcement Learning (RL) [Calinon et al., 2012b, 2013]
- In conjunction with Dynamical Systems [Pistillo et al., 2011]

As a result, the GMM has become a classical approach in trajectory LbD and despite the fact that it may not be the best suited model for sequential data, it is nevertheless widely used, in one form or another, sometimes as a component to extensions that significantly enhance its efficacy. The wide adoption of GMMs by the robotics community and its widespread use means that the benefit of improvements to the model performance is multiplied across the multitude of practical applications.

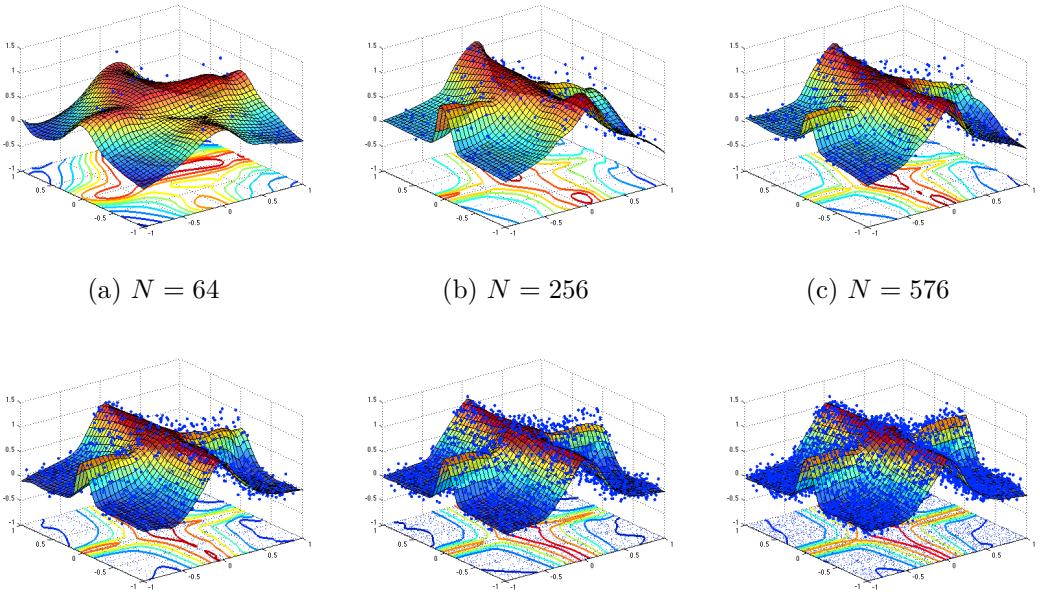
WHY ONLINE METHODS? In an academic setting, it is usually sufficient to provide proof of concept solutions and it is beyond the scope of academic research to design commercially deployable products. As the problems addressed in academia are highly challenging, researchers are often forced to make certain assumptions, simplifications and compromises in order to be able to successfully tackle them. Some of those assumptions, for instance, may be the following:

- Controlled and restricted laboratory environment.
  It is common to setup a controlled laboratory environment for conducting experiments, as it is difficult to address unrestricted generic problems.
- Phenomena that do not evolve in time (static).
  We may for instance assume that the ability of a player remains constant over time, as taking this into account may significantly complicate our methods.
- Loose time contains.
  A commercial solution usually requires strict time constraints in order to be viable, as user patience cannot be guaranteed.

On the contrary, in real-life scenarios, we often encounter highly diverse and/or dynamic environments, time-evolving phenomena and long delays may render our algorithms inapplicable. As a result, in order to integrate robotic applications more effectively and naturally into human everyday life, it may be preferable to compromise in terms of representation accuracy, but achieve a more dynamic, adaptive algorithmic behaviours.

For the aforementioned reasons, we envision adaptive, online training algorithms, able to refine their estimates over time. An example can be seen in fig. 1.1, where the estimate of a noisy surface is gradually refined as more and more training data-points become available to us. The same concept can be employed for learning dynamic phenomena that change over time.

WHY MISSING DATA? Accurate offline and dynamic online algorithms lie at the core of robot LbD. However, the assumption we implicitly adopt is that we are able to successfully acquire enough example data in order to perform inference.



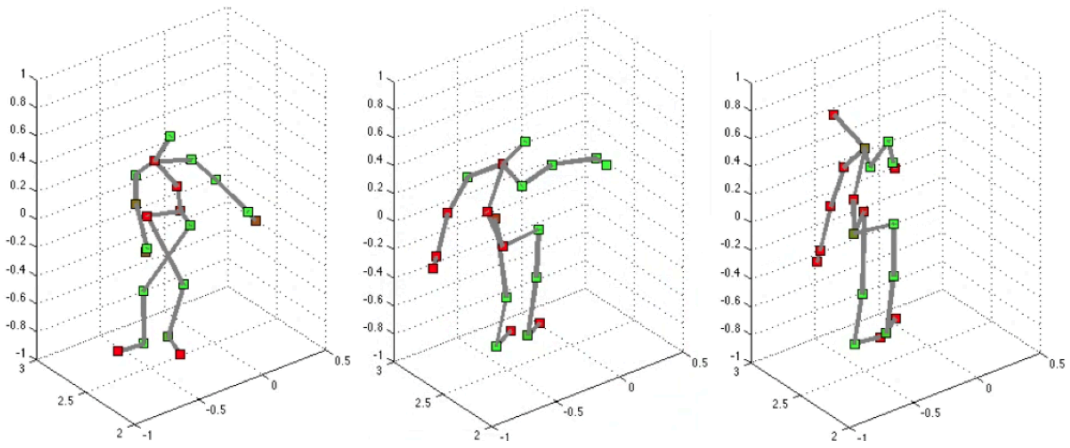
(a) $N = 64$ (b) $N = 256$ (c) $N = 576$

(d) $N = 1024$ (e) $N = 2304$ (f) $N = 4096$

Figure 1.1: Online learning of a non-linear function from noisy data. The surfaces we depict are calculated on a regular grid of test points that do not appear in the training set. As N we denote the number of data-points used for model training, prior to the testing phase. This classic benchmark first appears in [Schaal & Atkeson, 1998].

Data collection does not pose a significant problem itself. Nowadays, it is easier than ever, especially as affordable sensor equipment becomes increasingly available. As a result, there is a large amount of motion capture data of human demonstrations freely available online.

The main issue, however, lies in the quality of those data. Despite the advances in sensor equipment, it is sometimes impossible to avoid corruption. Even when using highly expensive and precise motion capture systems in a highly controlled environment, there is still a lot of effort required in manually rectify corrupted sequences or even discard data sequences that are beyond repair. The problem is



(a) (b) (c)

Figure 1.2: Microsoft Research Action Recognition 3D Dataset. The dataset consists of 3D joint positions and a metric showing the confidence of tracking for each joint. With blackgreen and blackred we depict joints tracked with high and low confidence respectively. A severe issue with this dataset is that the confidence metric not always accurate itself. Source: [Li et al., 2010]

further intensified in case we only have low-cost equipment in our disposal, such as the popular Kinect sensor.

Technological advances are not expected to eradicate this problem in the near future for two main reasons. First, the demand for user-friendly and/or portable tracking equipment does not allow for multi view tracking, which is necessary to alleviate the detrimental effects of occlusions. Second, as popular sensors become more precise and reliable (i.e. Kinect 2), demand automatically shifts towards even smaller, portable and affordable sensors (i.e. Leap) which are expected to suffer from similar reliability issues.

A specific instance that triggered our interest in getting involved into this issue is the Microsoft Research Action Recognition 3D Dataset [Li et al., 2010], which is captured with a Kinect sensor. As we can see in fig. 1.2, skeleton tracking is often very problematic and this particular dataset suffers from extensive corruption.

1.3 OUTLINE

1.3.1 *Contributions*

The application domain of robotics is irrefutably interesting, impactful and useful. On the other hand, the field of machine learning, has triggered impressive advances in machine intelligence in the past decade, it has however been less successful in infiltrating into more practical disciplines that could directly benefit from those advances, such as robotics.

It is our belief that inadequate attention has been paid towards formulating new methods or tailoring existing ones to the unique needs of robotic applications, rather than simply reusing decade-old models for novel purposes. Motivated by this realisation, we have chosen, in this work, to focus and contribute towards introducing improvements to 4 main areas of importance for robotic applications:

- Improving the accuracy of trajectory representation.
- Automatically determining the optimal model size.
- Formulating accurate online and adaptive learning algorithms.
- Handling the corruption of demonstration data.

The aforementioned contributions span across the full scale of main issues faced by researchers in Learning by Demonstration (LbD) and thus we believe their collective and cumulative impact is maximal.

Our main hypothesis is that it is possible to use recent advances in Machine Learning theory, in order to introduce improvements across the spectrum and consequently facilitate future advances and application in LbD. This is the single unifying theme that dominates this work.

From a purely practical point of view, roboticists interested in applying the methods developed in this work can benefit from the following contributions:

- We demonstrate the facility in robotics of a non-parametric Bayesian model, able to infer the optimal model size in a purely data driven manner. This approach can substitute the Bayesian Information Criterion (BIC) and it has been since then used by several other reserchers in robotics applications.
- We introduce a novel Gaussian Mixture Regression (GMR) method, inspired from quantum statistics, able to achieve state-of-the-art representation accuracy, without significant increase in computational costs. This method can substitute the conventional Gaussian Mixture Model (GMM) and introduce performance gains in most typical applications.
- We propose a novel, online learning algorithm for Quantum Mixture Regression (QMR), thus we extend it's applicability to large datasets and adaptive learning scenarios. Again, this approach can substitute the online Gaussian Mixture Model (oGMM) in most applications.
- Finally, we propose a novel method for unsupervised motion sequence completion. This approach is able to rectify severely damaged sequences, without requiring any human intervention.

Our motivations for this work are discussed in detail in Section 1.2 of this chapter.

1.3.2 *Publications*

The main publications stemming from the work presented in this thesis are listed as follows.

> [4] Korkinof & Demiris, Multi-task and Multi-kernel Gaussian Process Dynamical Systems, *IEEE Transactions Signal Processing*, under review

[3] Korkinof & Demiris, Online Quantum Mixture Regression for Trajectory Learning by Demonstration, *IROS*, 2013

[2] Chatzis, Korkinof & Demiris, A Quantum-Statistical Approach Towards Robot Learning by Demonstration. *IEEE Transactions on Robotics*, 2012

[1] Chatzis, Korkinof & Demiris, *A Nonparametric Bayesian Approach Towards Robot Learning by Demonstration.* Robotics and Autonomous Systems 2012

1.3.3 *Roadmap*

In this section, we present a rough outline of this thesis along with advice on how to read it more efficiently.

- CHAPTER 1 - INTRODUCTION

  Chapter 1 presents a general introduction to the thesis, along with the motivation that inspired this research and an outline.

- PART I - BACKGROUND

  - CHAPTER 2 - MACHINE LEARNING: contains a brief introduction to Statistical Machine Learning and the mathematical concepts encountered in the rest of this thesis. It is meant to act as a tutorial and expert readers may safely skip this chapter. It begins with the motivation that led to the advent of statistical modelling and continues with a tutorial-level introduction to important inference methods, models and recent developments. The contents of this chapter are a prerequisite for the rest of the thesis.

  - CHAPTER 3 - LBD: presents a brief review of Learning by Demonstration (LbD), the main open issues and approaches pertaining to Machine Learning.

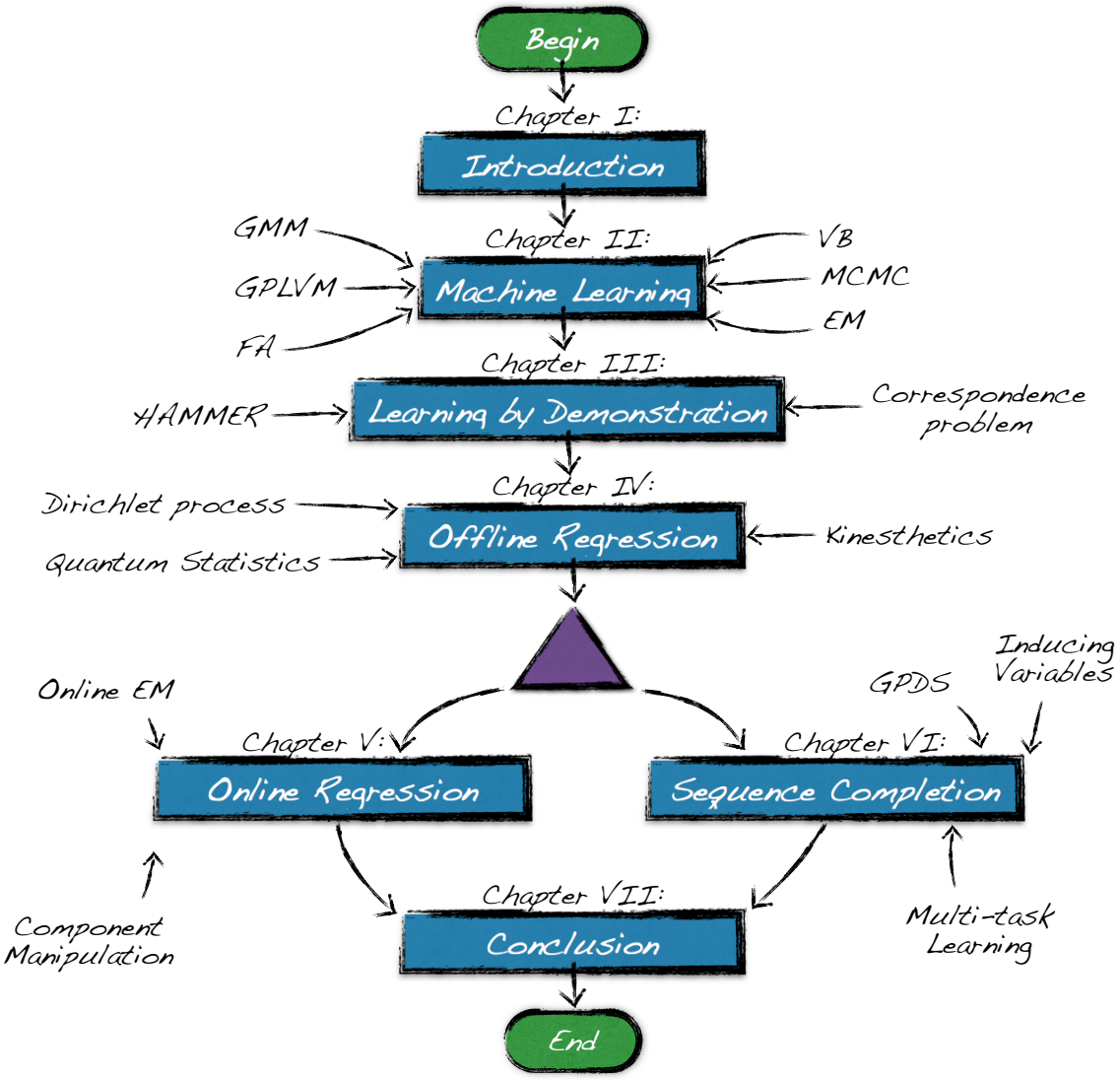


Figure 1.3: **Thesis roadmap**: Each node in this flow chart denotes a chapter in this thesis. We also add a few title keywords for the main concepts in each chapter.

- PART II - LEARNING FROM COMPLETE DATA
  - CHAPTER 4 - OFFLINE LEARNING: In this chapter we present the empirical evaluation of two models, the Dirichlet Process Mixture Regression (DPMR) and the Quantum Mixture Regression (QMR), presented in [Chatzis, Korkinof, and Demiris, 2012a] and [Chatzis, Korkinof, and Demiris, 2012c] respectively. We experimentally investigate the efficacy of those methods in robot LbD applications, considering a number of realistic scenarios.
  - CHAPTER 5 - ONLINE LEARNING: In this chapter, we present the online Quantum Mixture Model (oQMM), a method proposed in [Korkinof & Demiris, 2013], which combines the merits of quantum mechanics and stochastic optimisation. Our method is suitable for complex robotic applications, where data is abundant or where we wish to iteratively refine our model and conduct predictions during the course of learning. We demonstrate its efficacy in a number of benchmark and LbD applications.
- PART III - LEARNING FROM INCOMPLETE DATA
  - CHAPTER 6 - MULTI-KERNEL GPDS: In this chapter, we propose a model for rectifying damaged sequences in an unsupervised manner. We have devised a factor model consists of Gaussian Process Dynamical Systems (GPDS), each one modelling a single basic pattern in time and able to represent their sequential nature. Factors collectively form a dictionary of fundamental trajectories shared among different sequences and thus able to capture redundancies and allow for optimal knowledge transfer among similar tasks.
- CHAPTER 7 - CONCLUSION

  This chapter concludes the thesis and provides a brief overview of its contributions. We also discuss, in detail, the limitations of the proposed methods,

along with directions for future research that we wish to pursue in order to overcome these limitations. We consider this chapter very important, especially so for researchers or professionals interested in using the proposed algorithms in practice.

- PART IV - APPENDICES
  - APPENDIX A - MULTI-GPDS SUPPLEMENTARY RESULTS: In this appendix we present some supplementary results pertaining to the Multi-GPDS, the method proposed in Chapter 6.
  - APPENDIX B - TENSOR DECOMPOSITION: We present some background relevant to tensor algebra and decomposition. We also conduct a brief review of relevant tensor completion approaches in the literature.
  - APPENDIX C - BCD-LDS: We present some preliminary results for the Bayesian Canonical Decomposition Linear Dynamical Systems (BCD-LDS) method, which is outlined in the conclusion of this thesis and is part of our ongoing work.
  - APPENDIX D - BCD-LDS: We present the Gibbs sampler equations for the BCD-LDS for readers interested in implementing the method.

Part I

BACKGROUND

2

STATISTICAL MACHINE LEARNING

In this chapter we introduce Bayesian Machine Learning and explain its significance, as well as the main concepts and inference methods associated with it. We also provide a tutorial-level introduction into inference methods that are used throughout this thesis.

2.1 BAYESIAN INFERENCE

Bayesian statistics is one of many aspects of modern machine learning research. There are, of course, many successful models that are not, strictly speaking, Bayesian and still enjoy excellent performance. We could refer to the prior art in Deep Learning and stochastic Neural Networks for relevant examples of popular models employing optimisation rather than inference [Hinton & Salakhutdinov, 2006; Salakhutdinov & Hinton, 2009].

However, Bayesian inference exhibits traits that make it especially appealing both to the academia and industry. For that reason rigorous research was initiated towards formulating new such models [Griffiths & Ghahramani, 2006] or translating already existing ones into a Bayesian framework [Salakhutdinov & Mnih, 2008; Oh et al., 2005].

The main idea is to impose a prior distribution over the model parameters reflecting our a-priori beliefs for their values. Note that we make the choice of priors before conducting any estimates based on our observations, thus familiarity with the underlying phenomena could facilitate our choice; otherwise in case we are unable to make an educated guess, we can take an agnostic approach by simply choosing non-informative priors[1].

After obtaining some training data, we can refine our prior estimate and calculate a probability distribution reflecting our a-posteriori knowledge. The advantages of such an approach can be summarised as follows:

- It is consistent with the concept of learning-by-example and thus able to extract patterns from the data unassisted, as opposed to logic-based systems where a human would typically encode a set of rules.
- It enjoys a simple and elegant mathematical description, namely the Bayes law, which is at the epicentre of the field. Although for most models applying direct inference is intractable and for that reason we resort to complicated approximations, the simplicity of the underlying concept remains.
- We can estimate a full posterior distribution of the parameters and predictions. This is advantageous compared to conducting point-wise estimates as a full posterior can first be multimodal and, second, it allows us to assess the uncertainty of our estimates. This is definitely more informative and may be especially valuable in case we wish to apply active learning [Chatzis, Korkinof, and Demiris, 2011; Settles, 2010], novelty detection [Chandola et al., 2009], etc.
- It enjoys a very advantageous asymptotical convergence property, according to which the inference is guaranteed to asymptotically converge to the true data distribution regardless of the possibly poor choice of priors. That holds

1 The term "non-informative prior" refers to distributions that assume no prior knowledge of the parameter space. This is achieved by assigning either equal density to discrete parameter spaces or zero-mean and high variance to continuous parameter spaces.

as long as the prior does not allocate zero probability to any part of the parameter space.

- Finally, Bayesian models are less prone to overfitting and in some cases can automatically detect the optimal number of parameters required, either by reducing them in case of Automatic Relevance Determination (ARD) or by increasing them indefinitely in non-parametric models.

Let us now see how we progressed from the deceivingly simple concept formulated in Bayes' law, to a whole new field of statistics with such major contribution in almost every aspect of modern science and technology.

Let us denote as $\mathcal{D}$ a general data-set of observations and $\mathcal{M}$ a postulated model described using some parameter vector $\boldsymbol{\vartheta}$. Our main objective is to estimate the underlying model responsible for the generation of the dataset.

In a Bayesian setting and prior to observing any actual data we need to choose a prior probability distribution over the model parameters $\boldsymbol{\vartheta}$, denoted as $p(\boldsymbol{\vartheta})$, which would reflect our prior beliefs and uncertainties about parameter values. For instance, if we feel unsure about the possible values of our parameters, we might choose a broad and not especially informative prior distribution over the parameter space. We can then utilise the information in the observed data $\mathcal{D}$ in order to come up with a rather more educated guess, which will be reflected in the posterior distribution $p(\boldsymbol{\vartheta}|\mathcal{D})$. On the contrary, the more strongly biased the prior distribution we choose, the more evidence (conclusive data) will be required to overturn this initial prejudice. A simple example is a prior distribution placed over an ordinary six-sided dice (example taken from [Resnik & Hardisty, 2010]). Real dice are not exactly uniformly weighted, due to the laws of physics and the reality of manufacturing. A bag of dice manufactured using a crude process 100 years ago will likely have probabilities that deviate wildly from the uniform distribution, whereas a bag of state-of-the-art dice used by Las Vegas casinos may have barely perceptible imperfections. If we place a prior that reflects a strong bias towards a

highly unfair dice, a large number of observations will be needed to contradict this prior belief.

Quantities of interest in Bayesian inference include the following:

- Posterior distribution $p(\boldsymbol{\vartheta}|\mathcal{D})$.

  The posterior distribution for any given model may be obtained by simply applying Bayes rule for the aforementioned probabilities as follows:

  $$p(\boldsymbol{\vartheta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\vartheta})p(\boldsymbol{\vartheta})}{p(\mathcal{D})} \tag{2.1}$$

  where $p(\mathcal{D}|\boldsymbol{\vartheta})$ is the likelihood of the data given the model parameters and $p(\boldsymbol{\vartheta})$ is the prior placed over the model parameters.

- Evidence of model $\mathcal{M}$:

  The evidence of the model, which appears in the denominator of eq. (2.1) acting as the normalisation constant, is given as follows:

  $$p(\mathcal{D}|\mathcal{M}) = \int p(\mathcal{D}, \boldsymbol{\vartheta}|\mathcal{M})d\boldsymbol{\vartheta} = \int p(\mathcal{D}|\boldsymbol{\vartheta})p(\boldsymbol{\vartheta}|\mathcal{M})d\boldsymbol{\vartheta} \tag{2.2}$$

- Predictive distribution

  For a new data point $\mathbf{x}^*$,

  $$p(\mathbf{x}^*|\mathcal{D}) = \int p(\mathbf{x}^*, \boldsymbol{\vartheta}|\mathcal{D})d\boldsymbol{\vartheta} = \int p(\mathbf{x}^*|\boldsymbol{\vartheta})p(\boldsymbol{\vartheta}|\mathcal{D})d\boldsymbol{\vartheta} \tag{2.3}$$

  Notice that the posterior distribution of the parameters $p(\boldsymbol{\vartheta}|\mathcal{D})$ is needed to calculate the predictive distribution.

A frequently encountered issue is that, in all but the simplest of models, the posterior distribution of eq. (2.1) is not analytically tractable. This is certainly the case in most interesting models, with the most notable exception being the Gaussian Process (GP). The reason is that the integral required for obtaining marginal likelihood $p(\mathcal{D})$ can rarely be calculated analytically. Thus, we are usually able to calculate the posterior distribution family up to the normalisation constant.

To overcome this issue, we resort to approximate inference methods, with the main ones being:

- Expectation Maximisation (EM) algorithm [McLachlan & Krishnan, 2000]
- Variational Bayesian (VB) inference [Bishop, 2007]
- Expectation Propagation (EP) [Minka, 2001]
- Sampling methods, including but not limited to:
  - Importance Sampling (IS) [Murphy, 2012]
  - Gibbs sampling [Resnik & Hardisty, 2010]
  - Markov Chain Monte Carlo (MCMC) [Andrieu et al., 2003]
  - Sequential Monte Carlo (SMC) [Doucet et al., 2001a]

It is also important at this point to distinguish between parameter learning and state-space variable learning. From a theoretical perspective, we would be able to apply most inference methods to both cases, however, state-space variables pose unique difficulties mainly due to their spatio-temporal dependencies, and have triggered the advent of several inference method tailored to effectively addressing those challenges. Some examples of state-space models are the Gaussian Mixture Model (GMM), Factor Analyser (FA), Hidden Markov Random Field (HMRF), Hidden Markov Model (HMM), and Dynamical Systems (DS).

For the scope of this thesis we will mainly focus on the EM, VB and Gibbs sampling; however for reasons of brevity we present a brief introduction into other popular methods as well.

2.2 APPROXIMATE INFERENCE

2.2.1 *Maximum Likelihood Estimation (MLE)*

Strictly speaking, we cannot consider Maximum Likelihood Estimation (MLE) as one of the Bayesian inference methods. Nevertheless, we mention it here as it is frequently used in conjunction with Bayesian inference for estimating intractable parameters, an approach referred to as Type-II maximum likelihood. We can also use MLE as a standalone optimisation scheme.

The concept is fairly simple and focuses on the direct optimisation of the model evidence (likelihood) with respect to all parameters of interest. We would typically first attempt to find the stationary points of the likelihood by equating its derivatives to zero. However, in case the solution is not analytical we can resort to gradient ascend methods. In that perspective, MLE is applicable even for the most cumbersome of models as long as the computational cost of calculating the likelihood derivatives is not forbidding.

2.2.2 *Expectation Maximisation (EM)*

The Expectation Maximisation (EM) algorithm is perhaps one of the most widely used optimisation algorithm in Machine Learning and comprises a very powerful framework applicable in a variety of models employing latent variables. The algorithm was originally introduced for fitting mixture models, a task previous very difficult in the general formulation of the problem. The EM has also been used for inference in another established model, namely the Hidden Markov Model (HMM), although a similar solution known as the Baum-Welch algorithm was proposed before the emergence of the EM. For a comprehensive analysis of the EM and

its extensions for finite mixture models and hidden Markov models we refer to [McLachlan & Krishnan, 2000].

As we mentioned, the EM algorithm is particularly useful for models with implicit or explicit hidden variables. For instance, the Gaussian Mixture Model (GMM), in its original formulation of eq. (2.4), does not explicitly define any latent variables and it used to be notoriously difficult to obtain an estimate of the model parameters $\boldsymbol{\vartheta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$. However, if we augment the dataset with latent cluster indicators $\boldsymbol{z} = \{z_n\}_{n=1}^N$, the existence of which is implied in the model, the problem can be formulated in a much more manageable form as in eq. (2.5).

$$p\left(\boldsymbol{Y}|\left\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right\}_{k=1}^K\right) = \prod_{n=1}^N \sum_{k=1}^K \pi_k p\left(\boldsymbol{y}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) \tag{2.4}$$

$$p\left(\boldsymbol{Y}, \boldsymbol{z}|\left\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right\}_{k=1}^K\right) = \prod_{n=1}^N \prod_{k=1}^K \left[\pi_k p(\boldsymbol{y}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right]^{\delta(z_n=k)} \tag{2.5}$$

where the prior distribution over the latent cluster indicators is $p(z_n = k|\boldsymbol{\pi}) = \pi_k$ and $p(\boldsymbol{y}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \mathcal{N}(\boldsymbol{y}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$.

Given the complete data model likelihood $p\left(\boldsymbol{Y}, \boldsymbol{z}|\boldsymbol{\vartheta}\right)$, our aim is to optimise the marginal likelihood $p\left(\boldsymbol{Y}|\boldsymbol{\vartheta}\right)$ with respect to the parameter vector $\boldsymbol{\vartheta}$. The EM is able to achieve that by assuming a two-step optimisation as follows [Bishop, 2007]:

E-step:	Evaluate the posterior of the hidden indicators, given the parameters: $p(z_n = k\|\boldsymbol{Y}, \boldsymbol{\vartheta}^{old}) = \gamma_{nk}$
M-step:	Maximise the posterior expectation of the complete log-likelihood with respect to the parameters: $\boldsymbol{\vartheta}^{new} = \arg\max_{\boldsymbol{\vartheta}} Q\left(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{old}\right)$ where $Q\left(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{old}\right) = \mathbb{E}_{p(\boldsymbol{z}\|\boldsymbol{Y}, \boldsymbol{\vartheta}^{old})}\left[\ln p(\boldsymbol{Y}, \boldsymbol{z}|\boldsymbol{\vartheta})\right]$

In the example of the GMM we are considering, the posterior of the latent indicators is:

$$p(z_n = k|\boldsymbol{Y}, \boldsymbol{\vartheta}^{old}) \propto p\left(z_n = k|\boldsymbol{\pi}^{old}\right) p\left(\boldsymbol{y}_n|z_n = k, \boldsymbol{\vartheta}^{old}\right) = \frac{\pi_k \mathcal{N}\left(\boldsymbol{y}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\boldsymbol{y}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)} \tag{2.6}$$

and the posterior expectation of the complete log-likelihood that is to be maximised with respect of the parameters can be shown to be as follows:

$$Q\left(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{old}\right) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} \left[\ln \pi_k + \ln p(\boldsymbol{y}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right] \tag{2.7}$$

The E- and M-steps are repeated iteratively and convergence is theoretically guaranteed, thus at every iteration the marginal likelihood is bound to increase $p(\boldsymbol{Y}|\boldsymbol{\vartheta}^{new}) \geq p(\boldsymbol{Y}|\boldsymbol{\vartheta}^{old})$.

2.2.3 *Variational Bayesian (VB)*

We can trace variational methods' origins in physics, statistics and control theory in the form of the calculus of variations. In the recent past, variational methods have been employed for approximate inference and estimation in probabilistic models commonly found in machine learning.

Variational Mean Field Theory approximates the calculation of a complex posterior distribution based on the assumption that it may be fully factorised into individual independent component distributions. We can then optimise with respect to those functionals according to the calculus of variations.

As an example and without loss of generality, let us consider the GMM again but from a fully Bayesian perspective this time, defined as follows:

$$p\left(\boldsymbol{y}_n|z_n = k\right) = \mathcal{N}\left(\boldsymbol{y}_n|\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1}\right) \tag{2.8}$$

$$p\left(z_n = k\right) = \pi_k \tag{2.9}$$

$$p\left(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k\right) = \mathcal{N}\left(\boldsymbol{\mu}_k|\boldsymbol{m}_k, \left(\beta_k \boldsymbol{\Lambda}_k\right)^{-1}\right) \mathcal{W}\left(\boldsymbol{\Lambda}_k|\boldsymbol{W}_k, \nu_k\right) \tag{2.10}$$

$$p\left(\boldsymbol{\pi}\right) = Dir\left(\boldsymbol{\pi}|\boldsymbol{a}\right) \tag{2.11}$$

Bernoulli	$\pi^z(1-\pi)^{1-z}$	Beta	$\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\pi^{\alpha-1}(1-\pi)^{\beta-1}$
Multinomial	$\frac{(\sum_k z_k)!}{\prod_k z_k!}\prod_{k=1}^K \pi_k^{z_k}$	Dirichlet	$\frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)}\prod_{k=1}^K \pi_k^{\alpha_k-1}$
Univariate Normal	$\mathcal{N}\left(x\|\mu,\lambda^{-1}\right)$	Normal Gamma	$\mathcal{N}\left(\mu\|\mu_0,\lambda_0^{-1}\right)\mathcal{G}\left(\lambda\|\alpha,\beta\right)$
Multivariate Normal	$\mathcal{N}\left(\boldsymbol{x}\|\boldsymbol{\mu},\boldsymbol{\Lambda}^{-1}\right)$	Normal Wishart	$\mathcal{NW}\left(\boldsymbol{\mu},\boldsymbol{\Lambda}\|\boldsymbol{m}_0,\beta,\boldsymbol{\Lambda},\nu,\boldsymbol{W}\right)$

Table 2.1: Examples of frequently used conjugate distribution pairs.

The difference to the EM algorithm is that we are imposing priors with respect to all models parameters and consequently inference yields full posterior distributions rather than point estimates for $\boldsymbol{\vartheta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k\}_{k=1}^K$.

Let us now denote the vector $\mathcal{W} = \{\boldsymbol{z}, \pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k\}_{k=1}^K$, containing all model parameters and latent variables, upon which prior distributions have been imposed. The selection of prior distributions has to be done such as to facilitate inference, which can be achieved by selecting conjugate priors [2] to the parameter distributions.

Let us also define $\Theta = \{a_k, \boldsymbol{m}_k, \boldsymbol{W}_k, \nu_k, \beta_k\}_{k=1}^K$ to be the set of the hyper-parameters of all imposed priors.

In order to conduct variational inference for this model, we will introduce an arbitrary distribution denoted $q(\mathcal{W})$ to approximate the real but intractable posterior $p(\mathcal{W}|\Theta, \mathbf{Y})$. Under this assumption, the marginal log-likelihood (log-evidence) $\ln p(\boldsymbol{Y})$ of the model can be analysed as follows [Jordan et al., 1999]:

$$\ln p(\boldsymbol{Y}) = \ln \int p(\mathcal{W}|\boldsymbol{Y},\Theta)p(\boldsymbol{Y},\mathcal{W}|\Theta)d\mathcal{W} = \tag{2.12}$$

$$= \ln \int q(\mathcal{W})\frac{p(\mathcal{W}|\boldsymbol{Y},\Theta)p(\boldsymbol{Y},\mathcal{W}|\Theta)}{q(\mathcal{W})}d\mathcal{W} = \tag{2.13}$$

$$\geq \int q(\mathcal{W}) \ln \frac{p(\mathcal{W}|\boldsymbol{Y},\Theta)p(\boldsymbol{Y},\mathcal{W}|\Theta)}{q(\mathcal{W})}d\mathcal{W} = \tag{2.14}$$

$$= \int q(\mathcal{W}) \ln \frac{p(\boldsymbol{Y},\mathcal{W}|\Theta)}{q(\mathcal{W})}d\mathcal{W} + \int q(\mathcal{W}) \ln \frac{p(\mathcal{W}|\boldsymbol{Y},\Theta)}{q(\mathcal{W})}d\mathcal{W} \tag{2.15}$$

2 The term "conjugate prior" refers to a prior distribution selected in such a manner, that after applying Bayes rule (eq. 2.1) the functional form of the posterior is expected to remain the same as the prior. Conjugacy also depends on the likelihood structure and its existance cannot be guaranteed for all models. Some examples of useful conjugate distribution pairs are given in Table 2.1.

where by transitioning from eq. (2.13) to eq. (2.14) we have made use of the Jensen's inequality [3].

This analysis finally yields the following result:

$$\ln p(\boldsymbol{Y}) \geq \mathcal{L}(q) + KL(q\|p) \tag{2.16}$$

where $\mathcal{L}(q) = \int q(\mathcal{W}) \ln \frac{p(\boldsymbol{Y},\mathcal{W}|\Theta)}{q(\mathcal{W})} d\mathcal{W}$ and $KL(q\|p) = -\int q(\mathcal{W}) \ln \frac{p(\mathcal{W}|\boldsymbol{Y},\Theta)}{q(\mathcal{W})} d\mathcal{W}$ stand for the Lower Bound and the Kullback-Leibler (KL) divergence respectively. The latter is a measure of discrepancy between the (approximate) variational posterior $q(\mathcal{W})$ and the actual posterior $p(\mathcal{W}|\Theta, \boldsymbol{Y})$.

Since the KL divergence is non-negative, $\mathcal{L}(q)$ forms a strict lower bound of the log-evidence and would become exact if the approximate posterior coincides with the true posterior $q(\mathcal{W}) = p(\mathcal{W}|\Theta, \boldsymbol{Y})$.

$$\ln p(\boldsymbol{Y}) \geq \mathcal{L}(q) \tag{2.17}$$

Consequently, by maximising this lower bound $\mathcal{L}(q)$ (variational free energy) so that it becomes as tight as possible, we minimise the KL-divergence between the true and the variational posterior.

By imposing conjugate priors over the parameters, the variational posterior $q(\mathcal{W})$ is expected to take the same functional form as the prior $p(\mathcal{W})$, thus is expected to factorise for example as:

$$q(\mathcal{W}) = \prod_{k=1}^{K} q\left(a_k\right) \prod_{n=1}^{N} \prod_{k=1}^{K} q\left(z_{nk}\right) \prod_{k=1}^{K} q\left(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k\right) \tag{2.18}$$

It should be noted that no further assumptions have been made as to what the form of the approximate posterior distribution $q(\mathcal{W})$ might be, other than that it factorises w.r.t. its individual parameters. We will now proceed by optimising the lower bound with respect to the factorised approximate posterior.

3 Given that X is a random variable and $f(\cdot)$ a convex function, Jensen's inequality states that: $f\left(\mathbb{E}\left[X\right]\right) \leq \mathbb{E}\left[f\left(X\right)\right]$.

The lower bound of the GMM can be found as follows:

$$\begin{aligned}
\mathcal{L}(q) &= \int q(\mathcal{W}) \ln \frac{p(\boldsymbol{Y}, \mathcal{W}|\Theta)}{q(\mathcal{W})} d\mathcal{W} = && (2.19)\\
&= \sum_{k=1}^{K} \int q\left(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k\right) \ln \frac{\mathcal{NW}\left(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k\right)}{q\left(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k\right)} d\boldsymbol{\mu}_k d\boldsymbol{\Lambda}_k + && (2.20)\\
&+ \int q\left(\boldsymbol{\pi}\right) \ln \frac{Dir\left(\boldsymbol{\pi}|\boldsymbol{a}\right)}{q\left(\boldsymbol{\pi}\right)} d\boldsymbol{\pi} + && (2.21)\\
&+ \sum_{n=1}^{N} \sum_{k=1}^{K} q\left(z_n = k\right) \int q\left(\boldsymbol{\pi}\right) \ln \frac{p\left(z_n = k|\boldsymbol{\pi}\right)}{q\left(z_n = k\right)} d\boldsymbol{\pi} + && (2.22)\\
&+ \sum_{n=1}^{N} \sum_{k=1}^{K} \int q\left(z_n = k\right) q\left(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k\right) \ln \mathcal{N}\left(\boldsymbol{y}_n|\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k\right) d\boldsymbol{\mu}_k d\boldsymbol{\Lambda}_k && (2.23)
\end{aligned}$$

At this point we typically proceed by differentiating the lower bound with respect to each approximate posterior functional and equating the derivatives to zero; thus obtaining the optimal variational posteriors. This procedure yields analytical updates for the model parameters in cases of conjugate models.

VB is nowadays very widely used in Machine Learning, mainly due to the fact that it enjoys faster convergence than most sampling methods for instance. Typically, a VB and a Gibbs iteration would have a complexity of the same order of magnitude, however, we would need to execute many more Gibbs draws until convergence, compared to VB iterations.

2.2.3.1 *Non-conjugate Models*

Despite its merits, a severe limitation of VB is that it is not applicable to models exhibiting non-conjugacies. In that perspective, we may be restricted in our choice of priors or likelihoods to distribution families that yield conjugate variational posteriors. As a consequence, we would often need to compromise with respect to the model descriptive power.

During the years, many researchers have tried to circumvent this restriction of VB and device methods to extend it for inference in non-conjugate models. This is achieved by introducing further approximations that however, apart from be-

ing cumbersome, could introduce further (significant) computational cost and convergence issues.

The relevant methods can be summarised as follows:

LOWER BOUNDING A possible solution to the manifestation of analytically intractable terms is to replace them with tractable lower bounds. Note that we are already optimising with respect to the log-evidence's lower bound and this is not expected to significantly hamper inference as long as the bound of the intractable term is tight enough.

In [Griffiths & Ghahramani, 2006] for example, this approach is used in order to derive variational inference for the Indian Buffet Process (IBP). However, it is not always possible or easy to discover tight enough lower bounds.

TYPE-II MAXIMUM LIKELIHOOD Another approach to non-conjugacy is to exclude from the Bayesian inference parameters for which we are unable to formulate analytical updates. In this case, we would typically choose not to impose priors over those parameters and would use the derivatives of the lower bound to perform gradient-based optimisation. This approach is usually referred in the literature as Type-II Maximum Likelihood and is widely used in models employing Gaussian Process (GP) or Gaussian Process Latent Variable Model (GPLVM) priors to estimate the kernel parameters.

We would typically try performing Type-II MLE for a small enough number of model parameters, as we are otherwise expected to encounter convergence issues and high computational costs.

STOCHASTIC VARIATIONAL INFERENCE The stochastic variational framework, is a more general treatment of non-conjugacy [Wang & Blei, 2013]. The authors propose either a Laplace or a Taylor approximation of the non-conjugate terms with comparable performance.

A similar approach can be found in [Paisley et al., 2012], where the non-conjugate term is approximated by sampling and the sampler variance is reduced by means of a control variate without introducing any bias.

Another very recent approach is the [Titsias & Lázaro-Gredilla, 2014].

STOCHASTIC RELAXATION According to stochastic relaxation, we may choose to disregard the fluctuations of some variables causing non-conjugacy. Therefore, we choose to simplify the problem by fixing those variables to their expected values during part of the inference. This is an approach similar to Maximum a-Posteriori (MAP) and has proven especially useful for Markov Random Field (MRF) priors, such as [Chatzis, Korkinof, and Demiris, 2012b].

2.2.4 *Sampling Methods*

Inference using sampling methods is an approximation, but of different kind to the one employed in VB. In this particular case the objective is not to approximate the true posterior with another density, but rather to draw sufficiently many samples from the true posterior in order to obtain an estimate of its density. Theoretically speaking, using sampling methods we are able to approximate the true posterior with arbitrary precision. However, this is guaranteed in the asymptotical case, where we are able to draw an infinite number of samples. We typically need to compromise with respect to the number of sample we can draw from the posterior.

The most generic of the sampling algorithms is the Markov Chain Monte Carlo (MCMC), which allows sampling from a large class of distributions and scales well with the dimensionality of the sample space. We can trace the MCMC methods origin in physics [Metropolis & Ulam, 1949], and it was only towards the end of the 1980s that they started having a significant impact in the field of statistics and Machine Learning.

A large number of extensions and variations have emerged, including the Metropolis, Metropolis-Hastings algorithms [Bishop, 2007], sequential MCMC [Doucet et al., 2001b], Elliptical Slice Sampling (ESS) [Murray et al., 2010], Hamiltonian Monte Carlo (HMC) [Girolami & Calderhead, 2011] as well as Gibbs sampling [Resnik & Hardisty, 2010], to mention just a few. An exhaustive analysis of those methods is beyond the scope of this thesis, thus we shall briefly describe the basic idea and will focus on Gibbs sampling. For more information we refer to [Bishop, 2007].

Let us consider an integral that is intractable to solve analytically:

$$p(y|\mathcal{D}) = \int p(y|\boldsymbol{\vartheta})p(\boldsymbol{\vartheta}|\mathcal{D})d\boldsymbol{\vartheta} = \int f(\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z} = \mathbb{E}_{p(\boldsymbol{z})}[f(\boldsymbol{z})] \tag{2.24}$$

We have simplified the notation with the aid of a general function $f(\boldsymbol{z})$ and the corresponding probability distribution $p(\boldsymbol{z})$, so the above integral can also be seen as the expectation of function $f(\boldsymbol{z})$ with respect to the distribution $p(\boldsymbol{z})$. As $\boldsymbol{z}$ we assume a vector of random variables.

If we suppose that we have a mechanism of producing samples $\boldsymbol{z}^{(t)}$ from $p(\boldsymbol{z})$, then the expectation in question can be written as an infinite sum as follows:

$$\mathbb{E}_{p(\boldsymbol{z})}[f(\boldsymbol{z})] = \lim_{N\to+\infty} \frac{1}{N}\sum_{t=1}^{N} f(\boldsymbol{z}^{(t)}) \approx \frac{1}{N}\sum_{t=1}^{N} f(\boldsymbol{z}^{(t)}) \tag{2.25}$$

This infinite sum may be numerically approximated by a finite one, given that N is sufficiently large, otherwise we would not be able to retrieve the desired result.

For our purposes, the key concept is to perceive $\boldsymbol{z}$'s as points in a multidimensional state space and find ways to "walk around" the space - going from $\boldsymbol{z}^{(0)}$ to $\boldsymbol{z}^{(1)}$ to $\boldsymbol{z}^{(2)}$ and so forth - so that the likelihood of visiting any given point $\boldsymbol{z}$ is proportional to $p(\boldsymbol{z})$. We would prefer to spend our time adding values of $f(\boldsymbol{z})$ to the sum where $p(\boldsymbol{z})$ is large. The transition from $\boldsymbol{z}^{(t)}$ to the next state $\boldsymbol{z}^{(t+1)}$ is done in accordance with a transition probability distribution $P_{trans}(\boldsymbol{z}^{(t+1)}|\boldsymbol{z}^{(0)}, \boldsymbol{z}^{(1)}, \cdots, \boldsymbol{z}^{(t)})$, which needs to ensure that this stochastic process exhibits ergodicity and reversibility. More specifically ergodicity is ensured iff the algorithm is able to "forget" the initial starting point $\boldsymbol{z}^{(0)}$ and reversibility iff each transition $\boldsymbol{z}^{(t)} \to \boldsymbol{z}^{(t+1)}$ is possible to be reversed $\boldsymbol{z}^{(t+1)} \to \boldsymbol{z}^{(t)}$.

The fact that transitions are probabilistic makes this a Monte Carlo approach. What will make it a Markov Chain Monte Carlo algorithm is defining the transition probabilities so that the next state visited $\boldsymbol{z}^{(t+1)}$ depends solely on the current one $\boldsymbol{z}^{(t)}$. That is:

$$P_{trans}(\boldsymbol{z}^{(t+1)}|\boldsymbol{z}^{(0)}, \boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(t)}) = P_{trans}(\boldsymbol{z}^{(t+1)}|\boldsymbol{z}^{(t)}) \tag{2.26}$$

Gibbs sampling is a special case of MCMC, where the transition probability is of a specific form. Let us assume $\boldsymbol{z} = [z_1, z_2, \ldots, z_k]$, with $k > 1$. The basic concept is that, rather than probabilistically choosing the next state all at once and then deciding to accept or reject it, we make a separate partial transition choice for each of the z_k, where each choice depends on the state of all other $k-1$ random variables in the vector:

$$z_i^{(t+1)} \sim P(Z_i|z_1^{(t+1)}, \ldots, z_{i-1}^{(t+1)}, z_{i+1}^{(t)}, \ldots, z_k^{(t)}) \tag{2.27}$$

with

$$\begin{aligned} &P(Z_i|z_1^{(t+1)}, \ldots, z_{i-1}^{(t+1)}, z_{i+1}^{(t)}, \ldots, z_k^{(t)}) = \\ &= \frac{P(z_1^{(t+1)}, \ldots, z_{i-1}^{(t+1)}, z_i^{(t)}, z_{i+1}^{(t)}, \ldots, z_k^{(t)})}{P(z_1^{(t+1)}, \ldots, z_{i-1}^{(t+1)}, z_{i+1}^{(t)}, \ldots, z_k^{(t)})} \end{aligned} \tag{2.28}$$

Once we have obtained a new state $\boldsymbol{z}^{(t+1)}$, following the aforementioned transition scheme, this state is guaranteed to be a valid sample drawn from $p(\boldsymbol{z})$ and thus the transition $\boldsymbol{z}^{(t)} \to \boldsymbol{z}^{(t+1)}$ is always accepted. Note that this transition scheme does not involve rejection of samples.

Let us now derive some of the conditional posteriors required to perform Gibbs sampling for our Bayesian GMM example given by eq. (2.8)-(2.11):

$$p(z_n = k|\cdot) \quad \propto p(z_n = k|\boldsymbol{\pi})\mathcal{N}\left(\boldsymbol{y}_n|\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1}\right) \tag{2.29}$$

$$\begin{aligned} p(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k|\cdot) \quad &\propto \mathcal{N}\left(\boldsymbol{\mu}_k|\boldsymbol{m}_k, (\beta_k\boldsymbol{\Lambda}_k)^{-1}\right)\mathcal{W}\left(\boldsymbol{\Lambda}_k|\boldsymbol{W}_k, \nu_k\right) \\ &\prod_{n=1}^{N}\left[\mathcal{N}\left(\boldsymbol{y}_n|\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1}\right)\right]^{\delta(z_n=k)} \end{aligned} \tag{2.30}$$

$$p(\boldsymbol{\pi}|\cdot) \quad \propto Dir\left(\boldsymbol{\pi}|\boldsymbol{a}\right)\prod_{n=1}^{N}\prod_{k=1}^{K}\left[p\left(z_n = k|\boldsymbol{\pi}\right)\right]^{\delta(z_n=k)} \tag{2.31}$$

We iteratively draw samples from the above conditionals until convergence. We would typically need many more samples until convergence, compared to VB iterations,.

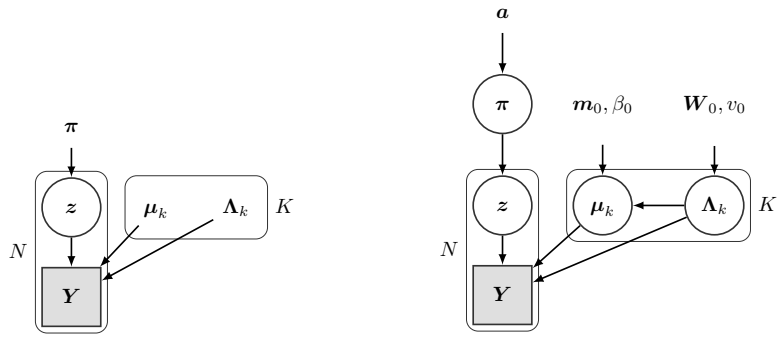
It should be noted that conjugacy is also a prerequisite for employing Gibbs sampling. However, we can easily handle non-conjugacies by applying an MCMC step along with the conjugate Gibbs updates; this is typically easier and more natural than handling non-conjugacies in VB inference, although the performance of the MCMC step may not always be satisfactory.

2.3 STATISTICAL MODELS

The purpose of this section is twofold. First we wish to provide a brief introductory-level overview of well-established models, so as to set a common reference frame for the rest of the thesis, where we may refer to those models. Second, we aim to provide a very brief review of important developments, highlighting approaches we consider relevant for gaining an understanding of the current prior art. However, this review however is by no means meant to be exhaustive or representative, but rather of introductory nature. Also note that the prior-art pertaining to each of the novel approaches proposed in this thesis is more formally presented in the beginning of each of the corresponding chapters in Part II and III.

2.3.1 *Mixture Models*

The finite mixture model has been used in many occasions over the years with Gaussian or other emission densities and has received numerous extensions and modifications. A factor that renders it especially appealing is that a sufficiently large Gaussian Mixture Model (GMM) can approximate any probability density with arbitrary precision.



(a) EM GMM (b) Bayesian GMM

Figure 2.1: Plate diagrams of the GMM from the perspective of the EM algorithm and from a Bayesian perspective. Filled square and circle nodes illustrate observed and hidden random variables respectively.

We have mentioned the model in Chapter 2, introducing the complete eq. (2.5) and incomplete eq. (2.4) data representation, as well as the Bayesian formulation of eq. (2.8)-(2.11). We have also used this basic model as an example solution under the EM algorithm, VB inference and Gibbs sampling. In fig. 4.5 we present plate diagrams of the GMM. More specifically, fig. 2.1a shows the model from the perspective of the EM algorithm, where there is no prior distribution imposed on the cluster parameters $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k\}_{k=1}^K$. On the contrary, in the fully Bayesian formulation those parameters are treated as random variables and we are able to infer their posterior distributions.

2.3.1.1 *Recent Developments*

There have been several approaches to augment conventional mixture models in order to achieve two main objectives (fig. 2.2) : 1) Enhancing the conventional model or 2) Taking advantage of the data structure. Some approaches address those two objectives simultaneously.

ENHANCING MIXTURE MODELS :

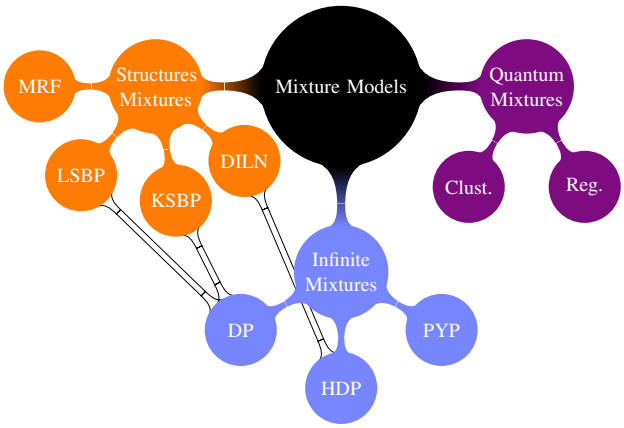


Figure 2.2: An illustration of different developments in Mixture Modelling and the relationships between them.

We can identify two main directions in enhancing conventional mixtures. One is the imposition of non-parametric priors [Antoniak, 1974; Walker et al., 1999; Neal, 2000] and the other is introducing quantum hidden states [Tanaka and Tsuda, 2008; Chatzis, Korkinof, and Demiris, 2012c; Korkinof and Demiris, 2013].

The non-parametric formulations of mixture models are based on the postulation of infinite states and they address the well documented difficulty of selecting the optimal number of states. The number of components would previous be estimated with ad-hoc methods, such as the Bayesian Information Criterion (BIC) [Schwarz et al., 1978] or the Akaike Information Criterion (AIC) [Posada & Buckley, 2004], which however yield notoriously noisy estimates.

On the contrary, using non-parametric priors is equivalent to imposing a prior distribution and inferring the optimal number of mixtures for a certain realisation of the dataset. Another comparative advantage to parametric models is that as we obtain more observations the number of mixtures will typically grow, rather than remain fixed and constrain the descriptive power of the model.

The most widely used prior for infinite mixture models is the Dirichlet Process (DP) and its hierarchical counterpart the Hierarchical Dirichlet Process (HDP) [Teh et al., 2006]. An alternative for data exhibiting power law behaviours is the Pitman-Yor Process (PYP) [Pitman & Yor, 1997; Teh, 2006].

It should be noted that non-parametric priors are not fundamentally new mathematical concepts. It is rather that they have only recently been employed in Bayesian Machine Learning. The focus on such models has also given rise to alternative representations (stick-breaking), facilitating VB [Kurihara et al., 2007; Teh et al., 2007b] and MCMC inference.

On the other hand, quantum inspired approaches attempt to capture the innate uncertainty in attributing data to clusters. Quantum clustering extends the principle of soft clustering (in mixture models) in the same manner as soft clustering extends the principle of hard-clustering (k-means algorithm). A more detailed explanation of quantum mixture models is provided in Chapter 4.

EXPLOITING DATA STRUCTURE :

Conventional mixture models have been also extended to take into account possible structure in the data. Extensive research has focused on formulating finite or infinite mixture models with neighbourhood preserving properties. Such models have proven especially useful in image and speech segmentation.

One of the most notable examples is the Hidden Markov Random Field (HMRF) [Forbes & Peyrard, 2003] which is based on the assumption that the cluster of the current datapoint is dependent on the clusters of its neighbours. This assumption is especially useful for data consisted of concise segments.

The notion has been pursued in a wide variety of models extending the DP and other similar stick-breaking processes to incorporate spatial information through MRF priors [Chatzis, Korkinof, and Demiris, 2012b], kernels [Dunson

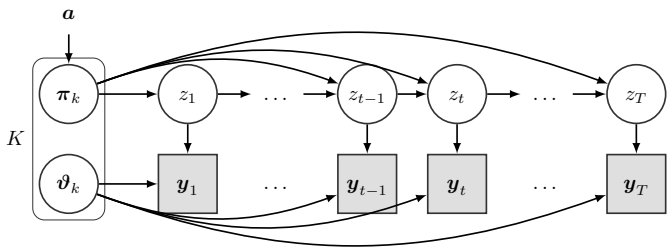


Figure 2.3: Plate diagram of the Hidden Markov Model (HMM). Note that $\boldsymbol{\vartheta}_k = \{\boldsymbol{\mu_k}, \boldsymbol{\Lambda}_k\}$ and $\boldsymbol{\pi}_k = [\pi_{k1}, \pi_{k2}, \ldots, \pi_{kK}]$ is a vector of transition probabilities from state k to any other model state.

& Park, 2008; Ren et al., 2011] or explicit spatial information [Rodrıguez & Dunson, 2011]. It is also interesting to observe models incorporating latent spatial information such as the Discrete Infinite Logistic Normal (DILN) distribution [Paisley et al., 2011].

2.3.2 *Hidden Markov Model*

The Hidden Markov Model (HMM) [Rabiner & Juang, 1986] can be simply viewed as a straightforward extension of the mixture model using a markovian prior on the cluster components. More specifically we can immediately retrieve the HMM from eqs. (2.8)-(2.11) by simply replacing eq. (2.9) with a markovian prior such as: $p(z_t = k|z_{t-1} = m) = \pi_{mk}$. The model's plate diagram is given in fig. 2.3.

However, the HMM has been of great significance to Statistical Machine Learning over the years, as it has not only opened new possibilities but has also posed new challenges. For instance, we employ dynamic programming methods to infer the optimal sequence of states, with algorithms referred to as forward-backward and Vitterbi.

2.3.2.1 *Recent Developments*

During the years the HMM has become an industry standard and has been in the epicentre of vigorous research, giving rise to numerous improvements and extentions.

We can briefly mention the segmental and semi- HMM [Yu, 2010], where each state is also associated with a duration in time. The factorial HMM [Ghahramani & Jordan, 1997] consisted of multiple markov chains and finally coupled [Brand et al., 1997] and fused [Pan et al., 2004] HMMs formulated to incorporate information from multiple data-streams.

With the advent of Bayesian non-parametrics the attention of the machine learning community has shifted to the infinite counterparts of the aforementioned models. Work towards this direction has given rise to the infinite HMM [Beal et al., 2001], HDP HMM [Teh et al., 2006], infinite semi-HMM [Johnson & Willsky, 2013], the infinite factorial HMM [Gael et al., 2009] and the sticky-HDP-HMM [Fox et al., 2008] among others.

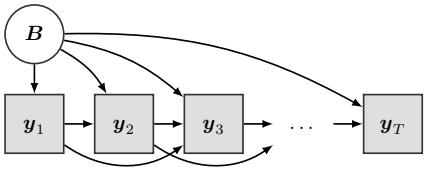
2.3.3 *Linear Dynamical Systems*

Linear Dynamical Systems (LDS) (fig. 2.4b) have been widely used in time-series modelling as they are able to effectively capture data dynamics in time. Let us postulate the following time-invariant state-space (SS) system [Fox et al., 2009]:

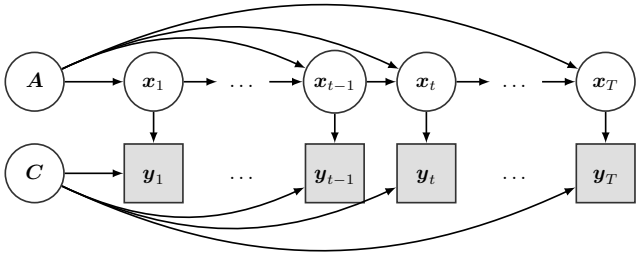
$$\boldsymbol{x}_t = \boldsymbol{A}\boldsymbol{x}_{t-1} + \boldsymbol{\epsilon}_t, \quad \boldsymbol{y}_t = \boldsymbol{C}\boldsymbol{x}_t + \boldsymbol{w}_t \tag{2.32}$$

where $\boldsymbol{y}_t \in \mathbb{R}^D$ is a multi-dimensional output in time, $\boldsymbol{x}_t \in \mathbb{R}^M$ represents the state variable, $\boldsymbol{\epsilon}_t$ and $\boldsymbol{w}_t$ represent Gaussian noise and $\boldsymbol{A}$, $\boldsymbol{C}$ are the system loading matrices.

The general LDS is a system of discrete-time differential equations, which, depending on the loading matrices' structure, represents a differential equation of up to M^{th}



(a) AR(m) process



(b) Linear Dynamical Systems (LDS)

Figure 2.4: Plate diagrams of the AR_m process and the Linear Dynamical Systems (LDS). Note that the in 2.4b $\boldsymbol{x}_t$ represents a multidimensional continuous latent variable, rather than a discrete state as in the case of the HMM.

order. In that perspective, an LDS is able to model a wide variety of complex dynamical phenomena.

Auto-regressive (AR) processes (fig. 2.4a) are closely related and are in fact a special case of the LDS we have previously described. More specifically an m^{th} order autoregressive process, referred to as AR_m, is defined as follows:

$$\boldsymbol{y}_t = \sum_{k=1}^{m} \boldsymbol{B}_k \boldsymbol{y}_{t-k} + \boldsymbol{\epsilon}_t \tag{2.33}$$

The AR_m can directly arise as a special case of the LDS in eq. (2.32), if we set the loading matrices as follows:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{B}_1 & \boldsymbol{B}_2 & \cdots & \boldsymbol{B}_M \\ 0 & \boldsymbol{I} & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{I} \end{bmatrix}, \quad \boldsymbol{C} = [\boldsymbol{I}, 0, \cdots, 0] \tag{2.34}$$

Every AR_m model can be represented as an LDS, but not vice-versa.

2.3.3.1 *Recent Developments*

The literature in LDS and AR processes has been prolific. Most notably we have had the advent of Switching Linear Dynamical Systems (SLDS) [Oh et al., 2005] and segmental SLDS [Oh et al., 2008], which are a combination of discrete and continuous latent variables usually dependent upon time. The main concept is that observations are emitted from different discrete states in time, where each state is modelled by a LDS or AR process thus representing different signal dynamics. SLDS have been extended in [Fox et al., 2009] using the sticky-HDP to form HMMs with emission probabilities based on LDS priors and have been successfully used in motion segmentation and modelling.

2.3.4 *Factor Analyser*

Factor Analyser (FA) is another standard model in Machine Learning. The assumption is that the observations $\{\boldsymbol{y}_n\}_{n=1}^N$ are governed by a set of latent factors $\{\boldsymbol{f}_n\}_{n=1}^N$, which are linearly combined according to a loading matrix $\boldsymbol{W}$:

$$\boldsymbol{y}_n = \boldsymbol{W}\boldsymbol{f}_n + \boldsymbol{\epsilon} \tag{2.35}$$

The concept has been extensively used for modelling purposes, as well as for dimensionality reduction, visualisation and descriptive statistics. A typical Bayesian FA would be described by the following priors on the model parameters:

$$\boldsymbol{y}_n \sim \mathcal{N}\left(\boldsymbol{W}\boldsymbol{f}_n + \boldsymbol{\mu}_y, \boldsymbol{\Lambda}_y^{-1}\right) \tag{2.36}$$

$$\boldsymbol{f}_n \sim \mathcal{N}\left(\boldsymbol{\mu}_f, \boldsymbol{\Lambda}_f^{-1}\right) \tag{2.37}$$

$$\boldsymbol{W} \sim \mathcal{MN}\left(\boldsymbol{M}_w, \boldsymbol{V}_w, \boldsymbol{K}_w\right) \tag{2.38}$$

where $\boldsymbol{y}_n$ is a $D \times 1$ vector of the n^{th} observation, $\boldsymbol{f}_n$ a $K \times 1$ vector containing the n^{th} factor and $\boldsymbol{W}$ is a $D \times K$ loading matrix. As $\boldsymbol{\mu}_y$ we denote a bias and $\boldsymbol{\Lambda}_y$

is the noise precision matrix. Finally, $\mathcal{MN}(\cdot)$ is the matrix-Normal distribution defined as follows:

$$\mathcal{MN}(\boldsymbol{W}|\boldsymbol{M},\boldsymbol{V},\boldsymbol{K}) = \frac{|\boldsymbol{K}|^{\frac{D}{2}}}{|2\pi\boldsymbol{V}|^{\frac{K}{2}}} e^{-\frac{1}{2}tr\left\{(\boldsymbol{W}-\boldsymbol{M})^T\boldsymbol{V}^{-1}(\boldsymbol{W}-\boldsymbol{M})\boldsymbol{K}\right\}} \tag{2.39}$$

where $\boldsymbol{V}$ and $\boldsymbol{K}$ are row- and column-wise covariance matrices.

For a simplified model we could relax the dependancy structure of the prior over the loading matrix and factorise the distribution either row- or column-wise depending on the interdependencies we wish to capture in the data. Thus eq. (2.38) could be replaced by either:

$$p(\boldsymbol{W}|\boldsymbol{\mu}_w,\boldsymbol{\Lambda}_w) = \prod_{k=1}^{K} \mathcal{N}\left(\boldsymbol{w}_k|\boldsymbol{\mu}_w,\boldsymbol{\Lambda}_w^{-1}\right) \tag{2.40}$$

or

$$p(\boldsymbol{W}|\boldsymbol{\mu}_w,\boldsymbol{\Lambda}_w) = \prod_{d=1}^{D} \mathcal{N}\left(\boldsymbol{w}_d|\boldsymbol{\mu}_w,\boldsymbol{\Lambda}_w^{-1}\right) \tag{2.41}$$

Factor models can also be viewed as a factorisation of the data matrix $\boldsymbol{Y}$. It should be noted that eq. (2.35) shows only one of many possible decompositions. More specifically, it shows a decomposition where factors vary across observations, but share common loadings. We could alternatively factorise the data into a set (dictionary) of static factors, shared among all data-points as follows:

$$\boldsymbol{y}_n = \boldsymbol{W}_n\boldsymbol{f} + \boldsymbol{\epsilon} \tag{2.42}$$

The above formulation is used in Chapter 6 of this thesis.

2.3.4.1 *Recent Developments*

Recent developments in factor modelling focus predominantly on finding low-rank, sparse representations of the data matrix. This effect may be achieved by either using an appropriate sparse prior on the factor loadings $\boldsymbol{W}$ or by postulating binary activation variables $\boldsymbol{Z}$ in which case eq. (2.35) takes the following form:

$$\boldsymbol{Y} = (\boldsymbol{W} \circ \boldsymbol{Z})\,\boldsymbol{F} + \boldsymbol{\epsilon} \tag{2.43}$$

where $\boldsymbol{Z}$ is a binary activation matrix on which we could impose a parametric or non-parametric prior and as $\circ$ we denote the element-wise or Hadamard product.

PARAMETRIC PRIORS :

The alternatives in the relevant literature are the following two:

1. The Automatic Relevance Determination (ARD) prior:

$$\boldsymbol{w}_k \sim \mathcal{N}\left(0, a_k^{-1}\boldsymbol{I}\right), \quad a_k \sim \mathcal{G}\left(a, \frac{1}{b}\right) \tag{2.44}$$

   where we have imposed a factor-wise zero-mean Normal distribution on the factor loadings. As the precision grows $a_k \to \infty$, the loadings of factor k diminish $\boldsymbol{w}_k \to 0$.

2. The "spike & slab" prior [Titsias & Lázaro-Gredilla, 2011]:

$$\boldsymbol{w}_k \sim \pi_k \mathcal{N}\left(\boldsymbol{w}_k|0, \sigma_w^2\boldsymbol{I}\right) + (1-\pi_k)\delta(\boldsymbol{w}_k) \tag{2.45}$$

   or equivalently

$$\boldsymbol{w}_k = z_k\hat{\boldsymbol{w}}_k \sim \pi_k^{z_k}(1-\pi_k)^{1-z_k}\mathcal{N}\left(\hat{\boldsymbol{w}}_k|0, \sigma_w^2\boldsymbol{I}\right) \tag{2.46}$$

   This prior allows factor loadings to diminish, when the binary indicator variable z_k is inactive.

NON-PARAMETRIC PRIORS :

In the case of non-parametric factor models, we stipulate an infinite possible number of factors, only a finite subset of which is active. This effect may be achieved by three main prior distributions:

1. The Indian Buffet Process (IBP) [Griffiths & Ghahramani, 2006, 2011] and more specifically its stick-breaking representation [Teh et al., 2007a]:

$$z_j \sim Bernoulli(\pi_j), \quad \pi_j = \prod_{k=1}^{j} u_k, \quad u_k \sim \mathcal{B}(a, 1) \tag{2.47}$$

2. The Beta Process (BP) [Paisley & Carin, 2009] and its stick-breaking representation [Paisley et al., 2010a]:

$$z_j \sim Bernoulli(\pi_j), \quad \pi_j \sim \mathcal{B}\left(\frac{a}{K}, \frac{K-1}{K}b\right) \tag{2.48}$$

where K is the truncation level on the number factors; we usually impose a high truncation level so as to achieve a good approximation.

3. And more recently the Multiplicative Gamma Process (MGP) [Bhattacharya & Dunson, 2011]:

$$\boldsymbol{w}_j \sim \mathcal{N}\left(0, a_j^{-1}\boldsymbol{I}\right), \quad a_j = \prod_{k=1}^{j} \xi_k, \quad \xi_k \sim \mathcal{G}\left(a, \frac{1}{b}\right) \tag{2.49}$$

2.3.5 *Gaussian Processes*

A Gaussian process is defined as a possibly infinite collection of random variables $[f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), ...]$, any finite subset of which follows a joint Gaussian distribution [Rasmussen & Williams, 2004].

More specifically, the training data $\mathcal{D} = (\boldsymbol{y}, \boldsymbol{X})$ are consisted of an observation vector $\boldsymbol{y} = [y_n]_{n=1}^N$ and a covariate matrix $\boldsymbol{X} = [\boldsymbol{x}_n]_{n=1}^N$, where, in general, $\boldsymbol{x}_n$ lies in a D-dimensional space, $\boldsymbol{x}_n \in \mathbb{R}^D$. The observations are modelled as follows:

$$y_n \sim \mathcal{N}\left(f(\boldsymbol{x}_n), \sigma^2\right) \tag{2.50}$$

$$\boldsymbol{f} \sim \mathcal{N}\left(0, \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})\right) \tag{2.51}$$

where $\boldsymbol{f} = [f_n]_{n=1}^N$ and each random variable $f_n = f(\boldsymbol{x}_n)$ represents the value of the function given input $\boldsymbol{x}_n$.

In that perspective, a Gaussian process can be viewed as a prior distribution over the set of all possible functions $f(\boldsymbol{x})$, with some characteristics dictated by the functional form of the kernel matrix $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$.

For a set of test points $\boldsymbol{X}_*$, direct inference is tractable for this model as follows:

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma^2\boldsymbol{I} & \boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}_*) \\ \boldsymbol{K}(\boldsymbol{X}_*,\boldsymbol{X}) & \boldsymbol{K}(\boldsymbol{X}_*,\boldsymbol{X}_*) \end{bmatrix}\right) \tag{2.52}$$

$$\mathbb{E}\left[\boldsymbol{f}_*|\boldsymbol{\mathcal{D}},\boldsymbol{X}_*\right] = \boldsymbol{K}(\boldsymbol{X}_*,\boldsymbol{X})\left[\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma^2\boldsymbol{I}\right]^{-1}\boldsymbol{y} \tag{2.53}$$

$$\mathbb{V}\left[\boldsymbol{f}_*|\boldsymbol{\mathcal{D}},\boldsymbol{X}_*\right] = \boldsymbol{K}(\boldsymbol{X}_*,\boldsymbol{X}_*) - \left[\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma^2\boldsymbol{I}\right]^{-1}\boldsymbol{K}(\boldsymbol{X}_*,\boldsymbol{X}) \tag{2.54}$$

This is a very convenient property, although inference does not scale well with the size of the training set as it requires the inversion of an $N \times N$ matrix with complexity $\mathcal{O}\left(N^3\right)$.

In order to account for multiple dimensions, we could impose either a common or independent GP priors per dimension, with the latter being preferable to the former [Rasmussen & Williams, 2004].

2.3.6 *Gaussian Process Latent Variable Model*

Let us now suppose a multi-dimensional set of observed data $\boldsymbol{Y} \in \mathbb{R}^{N\times D}$ and no known corresponding covariates $\boldsymbol{X}$. We may assume that the observed data can be accurately represented by a latent covariate space $\boldsymbol{X} \in \mathbb{R}^{N\times Q}$ of lower dimensionality $Q \ll D$ through a non-linear relationship, such as a GP kernel.

Given the latent covariate space $\boldsymbol{X}$, the GPs are postulated to be independent across dimensions, giving rise of the following factorisation.

$$p(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{d=1}^{D} p\left(\boldsymbol{Y}_{:d}|\boldsymbol{X}\right) \tag{2.55}$$

where as $\boldsymbol{Y}_{:d}$ we denote a vector consisted of the d^{th} column of $\boldsymbol{Y}$ and its corresponding probability distribution given by:

$$p\left(\boldsymbol{Y}_{:d}|\boldsymbol{X}\right) = \mathcal{N}\left(\boldsymbol{Y}_{:d}|\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma^2\boldsymbol{I}\right) \tag{2.56}$$

Note that the GPLVM is the generative counterpart of the GP. It has been successfully applied for the purpose of data visualisation [Lawrence, 2003], dimensionality reduction [Lawrence, 2005], classification [Urtasun & Darrell, 2007] and time-series modelling [Lawrence & Moore, 2007].

Estimating the latent covariates $\boldsymbol{X}$ may be achieved by directly optimising the likelihood in eq. (2.56), yielding point estimates (MLE) [Lawrence, 2003]. A Bayesian treatment of the model eqs. (2.57)-(2.59) was later proposed in [Titsias & Lawrence, 2010] by means of VB.

Variational inference for the model is of course intractable, due to the non-linear relationship between $\boldsymbol{X}$ and $\boldsymbol{f}_d$ in eq. (2.58). However, tractable inference can be achieved by introducing the concept of "inducing variables" [Titsias & Lawrence, 2010].

$$\boldsymbol{Y}_{:d} \sim \mathcal{N}\left(\boldsymbol{f}_d, \sigma^2 \boldsymbol{I}\right) \tag{2.57}$$

$$\boldsymbol{f}_d \sim \mathcal{N}\left(0, \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})\right) \tag{2.58}$$

$$\boldsymbol{x}_q \sim \mathcal{N}\left(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q\right) \tag{2.59}$$

More specifically, we can augment our original problem by introducing $M \ll N$ extra auxiliary sample points $\boldsymbol{u}$ drawn from the same prior distribution as the original sample $\boldsymbol{f}$, but located on a different set of latent locations $\boldsymbol{Z}$:

$$\boldsymbol{u} \sim \mathcal{N}(0, \boldsymbol{K}(\boldsymbol{Z}, \boldsymbol{Z})) \tag{2.60}$$

By doing so we do not need impose an explicit posterior on the original GP sample $\boldsymbol{f}$, but rather we can simply retrieve it it through the auxiliary sample $\boldsymbol{u}$, making use of eq. (2.61).

$$p\left(\boldsymbol{f}|\boldsymbol{u}, \boldsymbol{X}, \boldsymbol{Z}\right) = \mathcal{N}\left(\boldsymbol{f}|\boldsymbol{B}\boldsymbol{u}, \boldsymbol{S}\right) \tag{2.61}$$

$$\boldsymbol{B} = \boldsymbol{K}_{NM}\boldsymbol{K}_{MM}^{-1} \tag{2.62}$$

$$\boldsymbol{S} = \boldsymbol{K}_{NN} - \boldsymbol{K}_{NM}\boldsymbol{K}_{MM}^{-1}\boldsymbol{K}_{MN} \tag{2.63}$$

where $\boldsymbol{K}_{NN} = \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$, $\boldsymbol{K}_{NM} = \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{Z})$ and $\boldsymbol{K}_{MM} = \boldsymbol{K}(\boldsymbol{Z}, \boldsymbol{Z})$.

Type-II maximum likelihood is now only necessary for the much smaller set of still intractable latent locations $\boldsymbol{Z}$, while inference becomes tractable for $\boldsymbol{X}$.

3

LEARNING BY DEMONSTRATION

HISTORICALLY, the first robotic systems were designed for industrial purposes and more specifically in order to execute actions exhibiting high repetitiveness, requiring extremely high precision or dexterity. These industrial robots are deployed in static, controlled environments and therefore could be deterministically programmed to performed desired tasks.

Although the field of industrial robotics is thriving today, recent technological advances have been fuelled by our desire to utilise robotic assistance in our everyday life. However, the leap from static, controlled environments to a dynamic, uncertain and imprecise world is significant and requires alternative methods of skill acquisition, other than manual programming.

Modern ways of skill acquisition can be categorised as follows [Kormushev et al., 2013]:

- **Learning by Imitation**
  According to this concept, we can encode new skills by utilising human demonstrations, acquired by means of:
  - **Kinesthetics**
    In this case, a human demonstrator is required to manually move the robot joints to the desired positions. The demonstration is encoded in the form of joint angle positions in time and recently also by means of

force sensors located at the points of contact between human and robot. Although the method has been proven highly effective and is also extensively used in the present thesis, it nevertheless exhibits certain limitation. There are for instance actions that would be very difficult or sometimes impossible to demonstrate by kinesthetics, especially the ones requiring balance, such as walking.

- **Teleoperation**

  Teleoperation is an alternative way of introducing demonstrations and highly popular in medical/surgical robotics. This way we can obtain accurate demonstrations, provided that an appropriate teleoperation interface exists.

  Such approaches, however, are usually limited to end-effector or end-position motion and it is usually difficult to directly teleoperate high Degrees of Freedom (DoF).

- **Observation**

  Finally, it is possible to encode skills acquired by direct observation of the actions performed by a human demonstrator. Although purely vision-based observation is still very challenging, it is possible to record demonstrations using Motion Capture (MoCap) systems. This has become increasingly accessible with the emergence of low-cost depth sensors, such as the Microsoft Kinect, which nowadays enjoy high popularity in the scientific community.

  Learning by observation, however, remains challenging for two main reasons. First of all, transferring actions among different embodiments, also known as the "correspondence problem" is certainly non-trivial. Second, even when using highly expensive MoCap systems it is sometimes impossible to avoid corruption due to occlusions. This problem is only intensified when using single-view Kinect sensors.

- **Policy-based Learning**

  Policy-based learning is a way to learn new tasks automatically, without

necessarily providing any demonstration data and is usually performed by means of Reinforcement Learning (RL). The main concept is to search the action space for areas that comply with a certain reward function. The method is very useful, especially in cases when human demonstration is difficult or impossible.

Although the RL is powerful and widely used in robotics it suffers from very slow convergence in high dimensional action spaces.

Recent research attempts to narrow the search space of Reinforcement Learning (RL) algorithms utilising human demonstration and thus effectively combining RL and LbD [Calinon et al., 2012b, 2013].

For the purposes of this thesis we shall primarily focus on kinesthetics Learning by Demonstration (LbD) and handling data corruption in learning by observation.

3.1 IMITATION LEARNING

After two decades of rigorous research, LbD is still a popular concept today, maybe more than ever before, due to the advent of more precise and reliable robotic platforms. The challenges however faced by researchers in realistic applications still remain formidable and often overwhelming. The reason is that the underlying problem in question entails severe practical adversities, stemming from theoretical and technical challenges in several domains. To overcome these challenges researchers have utilised various methodologies stemming from diverse research areas such as machine learning, computer vision [Lopes & Santos-Victor, 2005], and human-robot interaction [Demiris & Khadhouri, 2006].

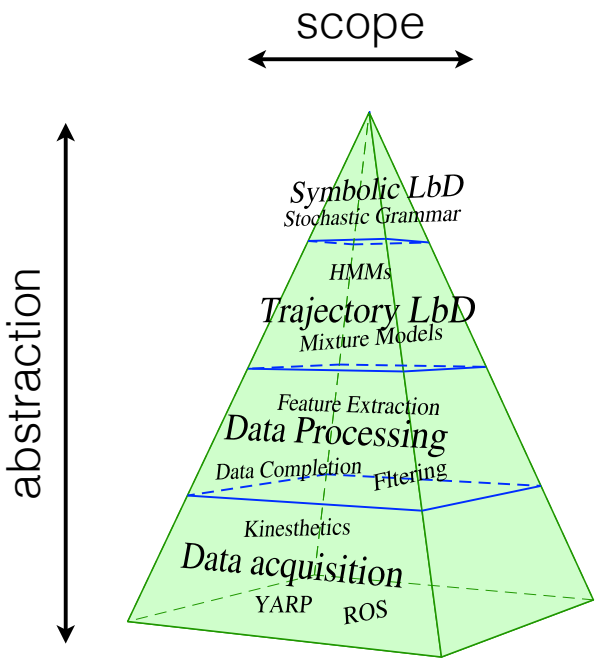


Figure 3.1: A pyramid illustration of Learning by Demonstration (LbD). Higher levels in the pyramid, indicate higher levels of abstraction. Broader pyramid layers indicate broader scope, in terms of more application- and platform-specific solutions.

3.2 STATE-ACTION MAPPING

In order to tackle the problem, researchers usually decompose the problem into simpler sub-problems with different levels of abstraction. Each level receives as input the results of the one below it and expresses the problem in a semantically higher level representation. In that sense, state-action mapping LbD can be graphically represented as a pyramid with ascending levels of abstraction fig. 3.1. The lower we are in the pyramid, the more application- and hardware-specific the issues we encounter. The scope of the issues is also broader in the sense that they are less platform and/or solution independent. On the contrary, as we ascend, we can deal with higher-level semantic information, relevant to learning sequences of actions for instance. The scope also becomes narrower in terms of application independence.

3.2.1 *Data acquisition and processing*

At the bottom level of abstraction lies the Data Acquisition. The process is very platform and application specific as it is highly dependent on the robotic platform, operating system and type of demonstration we wish to record.

It is followed by a Data Processing step, at which point our aim is to achieve the following:

- Denoise the data, by filtering-out excessive noise.
- Extract features appropriate for the application in question.
- Fill in missing segments caused by occlusions or sensor failures.

It should be noted that the data processing does not necessarily have to be a separate step, as incomplete or imperfect measurements can be dealt with in conjunction with modelling the trajectory data.

3.2.2 *Trajectory LbD*

Trajectory LbD is learning the motion trajectories of simple tasks or subtasks within a demonstration. It is a low-level skill representing. This may sometimes require to simplify the trajectories we wish to model by either segmenting a complex task into simpler actions or decomposing it into a linear superposition of simpler trajectories. This can be done as a separate step and there are many examples in the literature of either manual segmentation or decomposition, as well as machine learning method able to accomplish that automatically. However, many established Machine Learning models are implicitly segmenting or decomposing the data, as is the case for instance with GMMs, HMMs and FAs.

3.2.3 *Symbolic level LbD*

A successful segmentation and modelling of a demonstration could allow us to focus on a higher, symbolic level of abstraction. In this case we are predominantly interested in learning the sequence of actions and the order in which they occur. A significant amount of research in non-related areas is almost directly applicable to this problem. We can indicatively mention that logic, word sequence modelling [Wood et al., 2011] and stochastic grammars [Lee et al., 2012, 2013] are methods directly applicable to symbolic-level LbD. Alternatively, we could use algorithms formulated for natural language modelling, which, at present, are quite mature.

A prerequisite for applying symbolic LbD in the first place, is to be able to successfully segment large sequences into basic trajectories in order to generate their symbolic representation. In that sense, symbolic level LbD is reliant on the success of methods pertaining to the lower levels of abstraction.

There are two alternative approaches to achieving symbol generation stemming from research in trajectory LbD:

- **Unsupervised segmentation**: In this purely unsupervised approach, we could use statistical models that postulate a sequence of hidden states, such as HMMs or switching Linear Dynamical Systems (LDS) [Fox et al., 2009; Oh et al., 2005]. These models would be able to automatically generate a sequence of symbols, which however would not necessarily be semantically meaningful. This approach is used in [Lee et al., 2012].

- **Segmentation and recognition**: In case we wish to decompose our demonstration in semantically meaningful primitives, we need to first define the desired actions, then segment our sequence and finally recognised the action performed in each segment. As shown in [Lee et al., 2013], this can be done by segmenting the sequence using the method proposed in [**?**] and then applying standart classification algorithms such as an Support Vector Machine (SVM).

In all of the above cases, we are usually required to restrict ourselves to an a-priori fixed number of primitive actions, which is one of the most significant limitations of this approach. This requirement may be relaxed in the case of unsupervised motion segmentation by using modern non-parametric sequence segmentation methods, such as the ones proposed in [Oh et al., 2005].

3.2.4 *Task reproduction*

Both trajectory and symbolic LbD representations have their comparative merits and shortcomings [Billard et al., 2008]. The symbolic representation allows hierarchical learning, but requires a predefined set of basic actions, which renders it not especially versatile or adaptive. The trajectory-based method is a generic skill representation, thus it can handle different types of motion, however it does not allow for the reproduction of high level skills.

A fundamental issue that arises when attempting to reproduce a learned skill is transferring it across different embodiments. This is widely known as the "correspondence problem". The term was first introduced in [Nehaniv & Dautenhahn, 2000], although the issue was well-known long before. In the same work, the authors also present an algebraic foundation for the correspondence along with some necessary properties, however the formulation contributes little to none towards an actual solution.

3.2.5 *Machine Learning for LbD*

Several attempts have been made to handle the correspondence problem in the past. In [Alissandrakis et al., 2002], the authors' approach is to simplify the problem in a game of chess. Each piece is considered an agent with different embodiment. Another attempt on robotic imitation this time is presented in [Alissandrakis et al.,

2007]. From a biological perspective mapping between different embodiments was proposed in [Johnson & Demiris, 2004] using the HAMMER architecture [Demiris & Khadhouri, 2006], a powerful framework proven very effective in action recognition.

Several researchers have considered statistical machine learning algorithms as an effective means to extract trajectory patterns underlying a set of demonstrated skills. Indeed, previous approaches in trajectory-based robot LbD focus on GMMs [Ghahramani & Jordan, 1994; Calinon & Billard, 2009], HMMs [Calinon et al., 2010; Lee & Nakamura, 2007; Calinon et al., 2007], Locally Weighted Projection Regression (LWPR) [Vijayakumar et al., 2005; Peters & Schaal, 2008], GPs [Grimes et al., 2006; Nguyen-Tuong & Peters, 2008] and online Gaussian Process Regression (GPR) [Soh & Demiris, 2013].

A comprehensive approach towards the introduction of statistical Machine Learning in LbD using trajectory based skill representation, was presented in the Ph.D. thesis of Sylvain Calinon [Calinon, 2007]. Simple and well established models were used in this case for both learning and reproduction of tasks. In most cases these are GMMs and HMMs. More statistical machine learning approaches in robotics can be seen in the work presented in the following theses: [Ting, 2009] and [Plagemann, 2008].

For comprehensive reviews of Learning by Demonstration (LbD) in robotics we refer to [Schaal, 1999] and [Billard et al., 2008].

3.2.5.1 *Gaussian Mixture Regression*

Gaussian Mixture Regression (GMR) has been a statistical method especially popular in modern LbD either in conjunction with GMMs or HMMs. We briefly introduce the concept in this section.

Let us suppose that we have a set of training trajectories in our possession pertaining to motion in either end-effector or joint angle space. Let us also suppose each trajectory is expressed by means of its corresponding positions and velocities in

time $\mathcal{D} = \{\boldsymbol{x}_t, \boldsymbol{v}_t\}_{t=1}^{T}$. At each time-step, we may perceive the position $\boldsymbol{x}_t$ as a predictor variable and the corresponding velocity as a response variable. Under this assumption, we would be able to predict the desired velocity, given the current position and using that information we may retrieve the next time-step position $\boldsymbol{x}_{t+1}$. We should note that this setting is not exclusively used in LbD, but is also adopted in other unrelated application, such as traffic flow modelling [Kim et al., 2011].

Under a GMR setting we may jointly learn a generative model from the set of position and velocities, then use it to make predictions regarding the response variable, given a value of the predictor variable by means of the model conditional predictive distribution. Note that contrary to [Kim et al., 2011] or other discriminative methods, we do not make any direct assumptions regarding the form for the regression function.

The so obtained GMM can be given as follows:

$$p([\boldsymbol{x}_t, \boldsymbol{v}_t]|\boldsymbol{\pi}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}) = \sum_{k=1}^{K} \pi_k \mathcal{N}([\boldsymbol{x}_t, \boldsymbol{v}_t]|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{3.1}$$

where $\boldsymbol{\pi} = (\pi_k)_{k=1}^{K}$ are the prior weights of the mixture components, and $\mathcal{N}(\cdot|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is a Gaussian density with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$. We can typically obtain parameter estimates for this model by directly applying the Expectation Maximisation (EM) algorithm [McLachlan & Krishnan, 2000].

After obtaining the parameters of the density $p([\boldsymbol{x}_t, \boldsymbol{v}_t]|\boldsymbol{\pi}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K})$, we may retrieve a trajectory by obtaining, at each time-step, an estimate of the velocities, given the current position with the following conditional expectation:

$$\bar{\boldsymbol{v}}_t = \mathbb{E}\left[\boldsymbol{v}_t|\boldsymbol{x}_t, \boldsymbol{\pi}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}\right] \tag{3.2}$$

More specifically, following the analysis of [Bishop, 2007] the required expectation can be obtained by first expressing the means $\boldsymbol{\mu}_k$ and covariance matrices $\boldsymbol{\Sigma}_k$ of the component densities as $\boldsymbol{\mu}_k = \begin{bmatrix} \boldsymbol{\mu}_k^x \\ \boldsymbol{\mu}_k^v \end{bmatrix}$ and $\boldsymbol{\Sigma}_k = \begin{bmatrix} \boldsymbol{\Sigma}_k^{xx} & \boldsymbol{\Sigma}_k^{xv} \\ \boldsymbol{\Sigma}_k^{vx} & \boldsymbol{\Sigma}_k^{vv} \end{bmatrix}$.

Then the conditional probability $p\left(\boldsymbol{v}_t|\boldsymbol{x}_t, \boldsymbol{\pi}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K\right)$ of $\boldsymbol{v}_t$ given $\boldsymbol{x}_t$ is as follows [Bishop, 2007]:

$$p\left(\boldsymbol{v}|\boldsymbol{x}, \boldsymbol{\pi}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K\right) = \mathcal{N}(\boldsymbol{v}|\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}) \tag{3.3}$$

$$\tilde{\boldsymbol{\mu}} = \sum_{k=1}^{K} \phi_k(\boldsymbol{x}) \left[\boldsymbol{\mu}_k^v + \boldsymbol{\Sigma}_k^{vx} \left(\boldsymbol{\Sigma}_k^{xx}\right)^{-1} \left(\boldsymbol{x} - \boldsymbol{\mu}_k^x\right)\right] \tag{3.4}$$

$$\tilde{\boldsymbol{\Sigma}} = \sum_{k=1}^{K} \phi_k^2(\boldsymbol{x}) \left[\boldsymbol{\Sigma}_k^{vv} - \boldsymbol{\Sigma}_k^{vx} \left(\boldsymbol{\Sigma}_k^{x}\right)^{-1} \boldsymbol{\Sigma}_k^{xv}\right] \tag{3.5}$$

$$\phi_i(\boldsymbol{x}) = \frac{\pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k^x, \boldsymbol{\Sigma}_k^{xx})}{\sum_{m=1}^{K} \pi_m \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_m^x, \boldsymbol{\Sigma}_m^{xx})} \tag{3.6}$$

The required conditional expectation is the mean of the above distribution:

$$\bar{\boldsymbol{v}}_t = \mathbb{E}\left[\boldsymbol{v}_t|\boldsymbol{x}_t, \boldsymbol{\pi}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K\right] = \tilde{\boldsymbol{\mu}} \tag{3.7}$$

And the predictive variance:

$$\tilde{\boldsymbol{\Sigma}} = \mathbb{V}\left[\boldsymbol{v}_t|\boldsymbol{x}_t, \boldsymbol{\pi}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K\right] \tag{3.8}$$

This quantity can be used to assess prediction uncertainty continuously along the generated trajectories.

3.2.5.2 *Models with dynamics and kinematics*

Although GMR is a very popular and general method, applicable in a large variety of applications, however it does not account for motion dynamics or kinematics constraints. There are however several approaches in the Machine Learning literature that do and they range from simple to highly complicated ones.

One of the simplest approaches is to use Linear Dynamical Systems (LDS), and more specifically switching LDSs. Although inference becomes slightly more complicated, such methods exhibit high infiltration in robotics [Calinon et al., 2012a] and more complex non-parametric variants have been successfully employed for the purpose of motion segmentation [Fox et al., 2009; Oh et al., 2005].

Assuming linear dynamics may be restrictive in many occasions and that has motivated the advent of non-linear dynamical models that are frequently used

in conjunction with GPs. There are several such models, as, for instance, the following:

- Gaussian Process Dynamical Systems (GPDS) [Damianou et al., 2011]: This is a simple non-linear dynamical system, dependent on time and modelled by means of a Gaussian Process Latent Variable Model (GPLVM). The model is explained and extended in Chapter 6 of this thesis.

- Gaussian Process Dynamical Models (GPDM) [Wang et al., 2005, 2008]: This is again a dynamical system, where the non-linear transfer function of the state-space variable is modelled by means of a GP.

- Latent Force Models (LFM) [Alvarez et al., 2009a, 2010]: The assumption in this case is that the system is modelled through a first or second order Ordinary Differential Equation (ODE) with some latent input governed by a GP. Inference for the model has also been formulated by means of SMC [Hartikainen & Sarkka, 2012] and it has been successfully utilised in motion, as well financial applications.

Although the aforementioned models are more powerful and tailored for modelling motion, they are also rather cumbersome and pose significant challenges in terms of inference.

Part II

LEARNING FROM COMPLETE DATA

4

OFFLINE LEARNING

MODERN approaches towards trajectory LbD focus on utilising probabilistic models in order to encode demonstrations, extract their underlying constraints and reproduce smooth generalised motor trajectories. Popular model choices include either generative (GMM [Ghahramani & Jordan, 1994], HMM [Billard et al., 2006]) or discriminative (GPR [Argall et al., 2009]) regression models and their extensions [Nguyen-Tuong & Peters, 2008; Vijayakumar et al., 2005].

The GMM in particular, has been shown to be very effective in trajectory LbD [Lee & Nakamura, 2007; Billard et al., 2008] and is especially popular due to one appealing property according to which it is guaranteed that every probability density can be approximated with arbitrary precision by a Gaussian Mixture Model (GMM) of a sufficient number of components.

There are two main aspects of any statistical algorithm that may be identified as being critical for practical applications and especially so in the field of robotics. These factors are the accuracy of representation and the computational complexity induced. The former specifies the quality of learning and the latter the cost in time or computational resources. However, achieving a good compromise between the two is difficult and notoriously elusive. Increasing the number of model parameters is expected to increase accuracy, but induce higher computational burden. Moreover, after a certain threshold, the benefit of adding more parameters would diminish

and model generalisation is expected to suffer due to overfitting. In the case of GMR, both in terms of computational complexity and accuracy, it is imperative to choose the optimal number of mixture components for a given task.

In this chapter we present the empirical evaluation of the following two offline LbD methods:

DPMR : The Dirichlet Process Mixture Regression (DPMR) method is a Bayesian non-parametric model proposed in [Chatzis, Korkinof, and Demiris, 2012a], with two key attributes. First, it constitutes a fully Bayesian treatment of mixture regression using Variational Bayesian (VB) posterior inference. Second, it employs a non-parametric Dirichlet Process (DP) prior (such as in [Walker et al., 1999; Neal, 2000]) with a component shrinkage property for the purpose of automatically determining the optimal number of states. The main advantage of this method is that, by dynamically inferring the optimal number of states, it enjoys low computational complexity without compromising the model descriptive power.

QMR : The Quantum Mixture Regression (QMR) method is an extension of the conventional Gaussian Mixture Regression (GMR) inspired by quantum statistics, proposed in [Chatzis, Korkinof, and Demiris, 2012c]. The Quantum Mixture Model (QMM) in general is first encountered in the literature in [Tanaka & Tsuda, 2008], where the method has shown performance gains in image segmentation applications. The main concept is the introduction of composite, quantum states as a superposition of pure system states. It can be viewed as an extension to soft-clusting as we shall explain later in this chapter.

The remainder of this chapter is organised as follows: In Section 4.1, we present the relevant theoretical background on model selection, Dirichlet processes, infinite mixtures, quantum statistics and quantum mixtures. In Section 4.2, we provide the empirical evaluation for both models, including a comprehensive analysis of

their comparative advantages and limitations in one- and multi-shot LbD scenario applications. The final section contains a conclusion and future directions.

4.1 BACKGROUND

4.1.1 *Model Selection*

It is crucial in most typical applications to select the right model that represents the observed dataset. Too simple models will suffer from poor accuracy of representation and too complex ones will exhibit overfitting and poor generalisation on the test set.

To clarify the term "model complexity", it usually refers to the number of model parameters. Learning more model parameters, leads to higher algorithmic complexity within the same model family. Note however, that, although we usually quantify the model complexity in terms of the number of model parameters, this does not always apply across model families. In this case model complexity can only be qualitatively assessed. For instance, we could state that Linear Dynamical Systems (LDS) are less complex than non-linear Dynamical Systems with the same number of parameters, it would be difficult however to quantify this assessment.

In pursuit of an educated guess regarding the utility of a model for a particular dataset, we may use the marginal likelihood (or model evidence) as a selection criterion:

$$p(\mathcal{D}|\mathcal{M}) = \int p(\mathcal{D}|\boldsymbol{\vartheta};\mathcal{M})p(\boldsymbol{\vartheta}|\mathcal{M})d\boldsymbol{\vartheta} \tag{4.1}$$

where as $p(\mathcal{D}|\boldsymbol{\vartheta};\mathcal{M})$ is the likelihood of a dataset $\mathcal{D}$ given a model $\mathcal{M}$ with a set of parameters $\boldsymbol{\vartheta}$ and $p(\boldsymbol{\vartheta}|\mathcal{M})$ is the prior distribution of that set of parameters.

Therefore, the marginal likelihood is effectively providing information regarding the suitability of a particular model, regardless of the quality of parameter estimation.

In that sense it may be interpreted as the probability of the data, given the model $\mathcal{M}$, but averaging across all possible model parameter configurations.

Although the marginal likelihood can be very useful, it is unfortunately rarely tractable for any but the simplest of models. For that reason, alternative approximations have been proposed in the literature and used for model selection, most notably the Bayesian Information Criterion (BIC) [Schwarz et al., 1978].

More specifically, instead of solving the integral in eq. 4.1, which may be problematic, we could obtain the marginal likelihood from Bayes rule:

$$p(\mathcal{D}|\mathcal{M}) = \frac{p(\mathcal{D}|\boldsymbol{\vartheta};\mathcal{M}))p(\boldsymbol{\vartheta}|\mathcal{M}))}{p(\boldsymbol{\vartheta}|\mathcal{D};\mathcal{M})} \tag{4.2}$$

This itself is not any easier, as it requires the posterior distribution $p(\boldsymbol{\vartheta}|\mathcal{D};\mathcal{M})$.

In the case of the BIC, we first approximate the posterior using the Laplace approximation:

$$p(\boldsymbol{\vartheta}|\mathcal{D};\mathcal{M}) \backsimeq \mathcal{N}\left(\boldsymbol{\vartheta}|\boldsymbol{\vartheta}^*, \boldsymbol{H}^{-1}\right) \tag{4.3}$$

where $\boldsymbol{\vartheta}^*$ is the MLE solution w.r.t. the parameters and $\boldsymbol{H} = \nabla^2_{\boldsymbol{\vartheta}} p(\boldsymbol{\vartheta}|\mathcal{D};\mathcal{M})|_{\boldsymbol{\vartheta}=\boldsymbol{\vartheta}^*}$ is the Hessian of the posterior evaluated at the MLE solution $\boldsymbol{\vartheta}^*$.

We may then substitute eq. 4.3 in eq. 4.2 and evaluating the result at the stationary point $\boldsymbol{\vartheta}^*$ yields the following result:

$$\ln p(\mathcal{D}|\mathcal{M}) \backsimeq \ln p\left(\mathcal{D}|\boldsymbol{\vartheta}^*;\mathcal{M}\right) + \ln p\left(\boldsymbol{\vartheta}^*|\mathcal{M}\right) + \frac{D}{2}\ln 2\pi - \frac{1}{2}\ln|\boldsymbol{H}| \tag{4.4}$$

where D is the dimensionality of $\boldsymbol{\vartheta}$.

In the case of the BIC, we also adopt the assumption that the Hessian is full rank, which results to the following gross approximation of the marginal likelihood, penalising models with large number of parameters:

$$\ln p(\mathcal{D}|\mathcal{M}) \backsimeq \ln p\left(\mathcal{D}|\boldsymbol{\vartheta}^*;\mathcal{M}\right) - \frac{D}{2}\ln N \tag{4.5}$$

In brief, the main underlying concept of BIC is to penalise the likelihood of models with a large number of parameters. In practice, the BIC requires certain conditions

[Leroux et al., 1992], which are not necessarily met in all occasions [McLachlan & Krishnan, 2000].

In the case of GMR, the model complexity/number of parameters is directly analogous to the number of mixture components. Also note that we shall henceforth use the following three equivalent terms when referring to the number of mixture components of a Gaussian Mixture Model (GMM): a) mixture components, model components or just components, b) model states or states and c) mixtures.

4.1.2 *Non-parametric models*

Non-parametric models, such as the DPM described in this chapter, provide an alternative, more principled, view to the model selection problem.

More specifically, an a-priori fixed number of parameters acts as a bottleneck when we increase the number of available training data, restricting our models' descriptive power and not allowing them to capture the additional information that becomes available. Addressing this key issue, non-parametric models assume an a-priori infinite number of parameters, out of which, only a finite set manifests itself given a particular dataset.

Most of the highly successful models in the literature are, in fact, non-parametric; for instance, we may mention the Dirichlet process, the Gaussian process, kNN etc.

4.1.3 *Dirichlet Process*

The Dirichlet process was first introduced in [Ferguson, 1973] and is a non-parametric prior over distributions, in the same sense that the Dirichlet distribution is a parametric prior over polynomial distributions. The process is also dubbed the

"Chinese restaurant" in the literature, referring to an analogy with the selection of popular dishes in a Chinese restaurant.

A DP is denoted as $\mathrm{DP}(G_0, \alpha)$, where G_0 is a base distribution and α is a positive scalar refered to as the innovation parameter. We can draw parameters $\boldsymbol{\vartheta}_n \sim G_0$ pertaining to a parametric density. In the case of Gaussians $\boldsymbol{\vartheta}_n = \{\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n\}$ and then G_0 is the normal-Wishart.

We can draw samples from a DP as follows:

- The first sample $\boldsymbol{\vartheta}_1^* \sim G_0$ drawn from the base measure with probability 1.
- For drawing the N^{th} sample let us suppose that we have already drawn K unique parameter vectors $\{\boldsymbol{\vartheta}_k\}_{k=1}^K$ henceforth referred to as atoms. Then our N^{th} draw will be either selected from the set of existing atoms with discrete probabilities proportional to each atom's popularity or will be a new draw from G_0 with probability proportional to the innovation parameter α.

This procedure can be formally expressed as follows:

$$p(\boldsymbol{\vartheta}_N^* | \{\boldsymbol{\vartheta}_n^*\}_{n=1}^{N-1}, G_0, \alpha) = \frac{\alpha}{\alpha+N-1} G_0 + \sum_{k=1}^K \frac{m_k^{N-1}}{\alpha+N-1} \delta_{\boldsymbol{\vartheta}_k} \tag{4.6}$$

where with $\delta_{\boldsymbol{\vartheta}_k}$ we denote the distribution centred at a single unique atom $\boldsymbol{\vartheta}_k$ and $m_k^{N-1} = \sum_{n=1}^{N-1} \mathbb{I}(\boldsymbol{\vartheta}_n^* = \boldsymbol{\vartheta}_k)$ with $\mathbb{I}(\cdot)$ being 1 iff the input condition is true and 0 otherwise.

Note that the DP does not a-priori limit the number of atoms drawn $\{\boldsymbol{\vartheta}_k\}_{k=1}^K$, but rather postulates that, after a finite number of draws from the process, we will have a finite number of unique atoms that would additionally exhibit a clustering effect. In other words, more popular atoms will tend to be drawn more often.

An alternative formulation of the DP, which facilitates inference is provided by the stick-breaking construction of Sethuraman [Sethuraman, 1991]. Let us consider two infinite collections of independent random variables $\{u_k\}_{k=1}^\infty$, $\{\boldsymbol{\vartheta}_k\}_{k=1}^\infty$, where u_k is drawn from the Beta distribution $\mathcal{B}(1, \alpha)$ and $\boldsymbol{\vartheta}_k$ from the base distribution

G_0. The stick-breaking representation of the DP is then given by [Sethuraman, 1991] as:

$$DP = \sum_{k=1}^{\infty} \pi_k(\boldsymbol{u})\delta_{\boldsymbol{\vartheta}_k}, \quad \pi_k(\boldsymbol{u}) = u_k \prod_{j=1}^{k-1}(1-u_j) \tag{4.7}$$

This representation of the DP makes clear that the parameter support consists of a countably infinite sum of discrete atoms $\boldsymbol{\vartheta}_k$, drawn independently from G_0. It can also be observed that the innovation parameter α controls the mean value of the stick variables, u_k, as a hyper-parameter of their prior distribution; hence, it regulates the effective number of atoms [Sethuraman, 1991].

Under the stick-breaking representation of eq. (4.7) of the Dirichlet process, the atoms $\boldsymbol{\vartheta}_k$ can be seen as the parameters of the component distributions of a mixture model consists of an unbounded number of component densities, with mixing proportions $\pi_k(\boldsymbol{v})$. In this way, DP mixture models are formulated [Antoniak, 1974].

4.1.4 *Infinite Mixtures*

Let us suppose a multidimensional set of observations $\mathcal{D} = \{\boldsymbol{y}_n\}_{n=1}^N$ modelled by a DP mixture. Then, each observation $\boldsymbol{y}_n$ is assumed to be drawn from its own probability density function $p(\boldsymbol{y}_n|\boldsymbol{\vartheta}_n^*)$ parametrised by a parameter vector $\boldsymbol{\vartheta}_n^*$, with $\boldsymbol{\vartheta}_n^*$ drawn from a common DP prior. Observations may be assigned to a unique atom $\boldsymbol{\vartheta}_k$ with probability $\pi_k(\boldsymbol{u})$. Introducing indicator variables $\boldsymbol{z} = \{z_n\}_{n=1}^N$, with $z_n = k$ denoting that $\boldsymbol{\vartheta}_n^* = \boldsymbol{\vartheta}_k$ the model likelihood and prior specifications are given as follows [Blei & Jordan, 2006]:

$$\boldsymbol{y}_n|z_n = k, \boldsymbol{\vartheta}_k \sim p(\boldsymbol{y}_n|\boldsymbol{\vartheta}_k) \tag{4.8}$$

$$p(z_n = k|\boldsymbol{\pi}(\boldsymbol{u})) = \pi_k(\boldsymbol{u}) \tag{4.9}$$

$$u_k|\alpha \sim \mathcal{B}(1,\alpha) \tag{4.10}$$

$$\boldsymbol{\vartheta}_k|G_0 \sim G_0 \tag{4.11}$$

where $\boldsymbol{\pi}(\boldsymbol{u}) = \{\pi_k(\boldsymbol{u})\}_{k=1}^{\infty}$ is given in eq. (4.7).

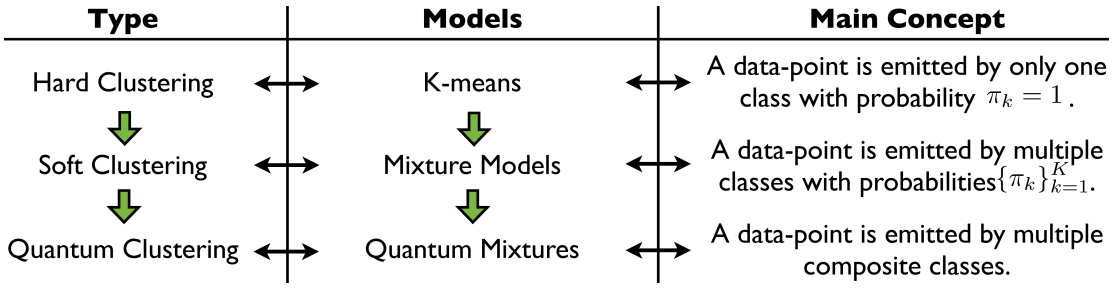


Figure 4.1: Model evolution in time (vertical arrows) and the main underlying concepts associated with each one.

4.1.4.1 *Posterior Inference*

As mentioned in Chapter 2, inference for Bayesian models can typically be conducted by means of Variational Bayesian (VB) or Markov Chain Monte Carlo (MCMC) methods. In this chapter, we have used VB inference, but we could equivalently employ Gibbs sampling as described in Section 2.2.4. For an introduction to VB inference we refer to Section 2.2.3 of this thesis. For detailed parameter updates of the aforementioned model we refer to [Chatzis, Korkinof, and Demiris, 2012a], a work which follows exactly the derivations first appearing in [Blei & Jordan, 2006].

4.1.5 *Quantum Statistics*

The generalisation of conventional probability theory has given rise to a whole new field of mathematics with particular applicability in physics, namely the field of quantum statistics. According to the concept of quantum probability, a classical probability density can be generalised by a density matrix, let us denote it as $\boldsymbol{F}$, with the following properties:

- $\boldsymbol{x}^T \boldsymbol{F} \boldsymbol{x} \geq 0$, $\forall \boldsymbol{x}$: Positive semi-definite.
- $\boldsymbol{F} = \boldsymbol{F}^{\dagger}$: Hermitian (or self-adjoint)[1].

1 With † we denote the conjugate transpose of a matrix.

- $tr\{\boldsymbol{F}\} = 1$: Normalized.

For instance, a conventional probability density of an event u having K distinct outcomes with probabilities $p(u = k) = \pi_k, \forall k = 1, \ldots, K$ can be described by the following diagonal probability matrix:

$$\boldsymbol{F} = diag\left([\pi_1, ..., \pi_k]\right) = \sum_{k=1}^{K} \pi_k \boldsymbol{e}_k \boldsymbol{e}_k^T \tag{4.12}$$

with $\{\boldsymbol{e}_k\}_{k=1}^{K}$ being a set of basis vectors of pure states, such as:

$$[\boldsymbol{e}_k]_i = \begin{cases} 1, & i = k \\ 0, & i \neq k \end{cases} \tag{4.13}$$

and $[\boldsymbol{e}_k]_i$ is the i^{th}element of vector $\boldsymbol{e}_k$.

In quantum statistics we are able to extend the probability matrix $\boldsymbol{F}$ so as to allow the manifestation of non-diagonal elements. This, in turn, gives rise to composite states, formed as a superposition of the system's pure states:

$$\boldsymbol{F} = \sum_{k=1}^{K} \pi_k \boldsymbol{u}_k \boldsymbol{u}_k^T \tag{4.14}$$

where a basis vector $\boldsymbol{u}_k = \begin{bmatrix} \frac{\sqrt{2}}{3} & \frac{2}{3} & \frac{\sqrt{3}}{3} \end{bmatrix}^T$, would mean that this state corresponds to a mixture of the three system pure states with probabilities $\frac{2}{9}$, $\frac{4}{9}$ and $\frac{3}{9}$ respectively. More specifically, the square of elements of the normalised basis vector for each direction corresponds to the membership probabilities of each state.

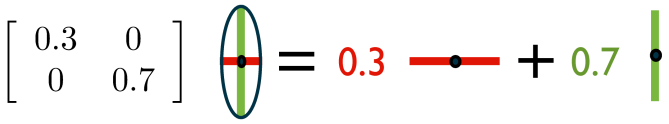
4.1.5.1 *Numerical Example*

Let us suppose a conventional mixture of two components with probability vector:

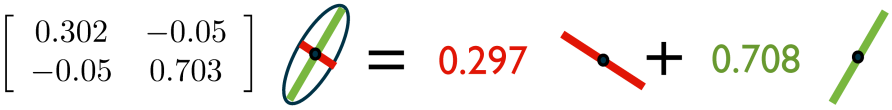
$$\begin{bmatrix} \pi_1 & \pi_2 \end{bmatrix}^T = \begin{bmatrix} 0.3 & 0.7 \end{bmatrix}^T \tag{4.15}$$

Under a quantum statistical perspective the system's probability matrix is expressed as follows:

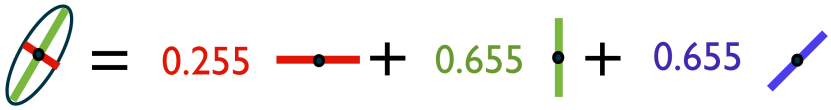
$$\begin{bmatrix} \pi_1 & 0 \\ 0 & \pi_2 \end{bmatrix} = \pi_1 \boldsymbol{e}_1 \boldsymbol{e}_1^T + \pi_2 \boldsymbol{e}_2 \boldsymbol{e}_2^T \tag{4.16}$$



(a) Pure probability state-space system consists of a mixture of two distributions with probabilities 0.3 and 0.7.



(b) Quantum probability state-space system constructed by applying a quantum disturbance $\gamma = 0.1$ to the pure state system. We have expressed the non-diagonal probability matrix using it's eigenvectors.



(c) Alternative view of the same non-diagonal probability matrix, in this case not by means of it's eigenvectors. It should be noted that the number of alternative representations is unbounded.

Figure 4.2: Illustrative numerical example of the relation between quantum and conventional statistics.

where $\boldsymbol{e}_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$ and $\boldsymbol{e}_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$.

We shall now "disturb" this conventional probability space, in order to transform it to, an almost equivalent, quantum probability space, by introducing a non-diagonal coefficient γ, which we dub a "quantum disturbance". We add the quantum disturbance to the off-diagonal elements of the log probability matrix of the pure system and exponentiating the result yields a quantum system that is able to capture richer data dependancies, as we show later in this chapter.

$$A = -\begin{bmatrix} 1.204 & 0.1 \\ 0.1 & 0.357 \end{bmatrix} \Rightarrow e^A = \begin{bmatrix} 0.302 & -0.05 \\ -0.05 & -0.703 \end{bmatrix} \tag{4.17}$$

It should be noted that adding non-diagonal elements in this manner always results in positive semi-definite probability matrices, due to the fact that A is symmetric.

The latter quantum state system can be analyzed with respect to the probability matrix eigenvectors (fig. 4.2b) as follows:

$$\begin{aligned} e^A &= 0.297 \begin{bmatrix} -0.993 \\ -0.116 \end{bmatrix} \begin{bmatrix} -0.993 \\ -0.116 \end{bmatrix}^T \\ &+ 0.708 \begin{bmatrix} 0.115 \\ -0.993 \end{bmatrix} \begin{bmatrix} 0.115 \\ -0.993 \end{bmatrix}^T \end{aligned}$$

The resulting system consists of two quantum classes, with probabilities 0.297 and 0.708. Each quantum class is composite and formed by linear superposition of the system's pure classes with probability vectors $\begin{bmatrix} 0.987 & 0.013 \end{bmatrix}^T$ and $\begin{bmatrix} 0.013 & 0.987 \end{bmatrix}^T$ (squares of elements of the normalised basis vector). Of course, the probability matrix can also be expressed in any other alternative way (fig. 4.2c).

Concluding the quantum statistical extension of the conventional mixture model can also be viewed as an elegant mixture of mixtures. Succeeding the concepts of hard clustering (k-means algorithm) and the soft clustering (mixture model), the quantum extension can be viewed as the next evolutionary step in mixture modeling (fig. 4.1).

4.1.6 *Quantum Mixtures*

The following formal introduction to quantum mixture models is a summary of the analysis first presented in [Tanaka & Tsuda, 2008]:

We first define the following probability density matrices:

$$\boldsymbol{F} \triangleq - \begin{bmatrix} \ln \pi_1 & 0 & \dots & 0 \\ 0 & \ln \pi_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \ln \pi_K \end{bmatrix} \tag{4.18}$$

and

$$\boldsymbol{G}(\boldsymbol{y}_n) \triangleq \begin{bmatrix} \ln p\left(\boldsymbol{y}_n|\boldsymbol{\vartheta}_1\right) & 0 & \dots & 0 \\ 0 & \ln p\left(\boldsymbol{y}_n|\boldsymbol{\vartheta}_2\right) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \ln p\left(\boldsymbol{y}_n|\boldsymbol{\vartheta}_K\right) \end{bmatrix} \tag{4.19}$$

where $\{\pi_k\}_{k=1}^K$ is a normalised set of discrete probabilities, $\boldsymbol{Y} = \{\boldsymbol{y}_n\}_{n=1}^N,\ \boldsymbol{y}_n \in \mathbb{R}^D$ is a set of multidimensional observations and $p(\cdot)$ is some set of appropriate density functions, which without loss of generality shall be henceforth assumed Gaussians, but could be any other density:

$$p\left(\boldsymbol{y}_n|\boldsymbol{\vartheta}_k\right) = \mathcal{N}\left(\boldsymbol{y}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) \tag{4.20}$$

with $\boldsymbol{\vartheta}_k = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ being its parameter vector. Under this scheme, the model log-likelihood is simply:

$$\boldsymbol{H}(\boldsymbol{y}_n) = \boldsymbol{F} - \boldsymbol{G}(\boldsymbol{y}_n) \tag{4.21}$$

The above construction is a quantum statistical equivalent to the conventional GMM. We have effectively expressed the GMM likelihood as a function of a quantum density matrix of a special form, namely diagonal. Introducing quantum effects to the density matrix $\boldsymbol{F}$ can be achieved by adding a quantum perturbation to the conventional system in the form of non-diagonal elements γ. This formulation yields the following generalisation towards a quantum mixture model:

$$\boldsymbol{F} = -\begin{bmatrix} \ln \pi_1 & \cdots & \gamma \\ \vdots & \ddots & \vdots \\ \gamma & \cdots & \ln \pi_K \end{bmatrix} \tag{4.22}$$

$$\boldsymbol{H}(\boldsymbol{y}_n) = -\sum_{k=1}^{K}\sum_{k'=1}^{K} B_{kk'}^{(n)} \boldsymbol{X}_{kk'} = -\begin{bmatrix} \ln\left(\pi_1 p\left(\boldsymbol{y}_n|\boldsymbol{\vartheta}_1\right)\right) & \cdots & \gamma \\ \vdots & \ddots & \vdots \\ \gamma & \cdots & \ln\left(\pi_K p\left(\boldsymbol{y}_n|\boldsymbol{\vartheta}_K\right)\right) \end{bmatrix} \tag{4.23}$$

$$B_{kk'}^{(n)} = \begin{cases} \ln\left(\pi_k p\left(\boldsymbol{y}_n|\boldsymbol{\vartheta}_k\right)\right) & , k = k' \\ \gamma & , otherwise \end{cases} \tag{4.24}$$

$$\left(\boldsymbol{X}_{kk'}\right)_{ij} = \begin{cases} 1 & , i = j \\ 0 & , otherwise \end{cases} \tag{4.25}$$

The likelihood of the data under the aforementioned model can be formulated as follows:

$$\mathcal{L}(\boldsymbol{Y}) = p\left(\boldsymbol{Y}|\left\{\pi_k, \boldsymbol{\vartheta}_k\right\}_{k=1}^{K}\right) \backsimeq \prod_{n=1}^{N} \frac{tr\left\{e^{-\boldsymbol{H}(\boldsymbol{y}_n)}\right\}}{tr\left\{e^{-\boldsymbol{F}}\right\}} \tag{4.26}$$

where the exponential $\exp(\boldsymbol{A})$ of a matrix $\boldsymbol{A}$ is defined as:

$$\exp(\boldsymbol{A}) \triangleq \sum_{\rho=0}^{\infty} \frac{1}{\rho!} \boldsymbol{A}^{\rho} \tag{4.27}$$

and the logarithm $\log(\boldsymbol{A})$ is given by:

$$\log(\boldsymbol{A}) \triangleq -\sum_{\rho=1}^{\infty} \frac{1}{\rho} (\boldsymbol{I} - \boldsymbol{A})^{\rho} \tag{4.28}$$

The quantum perturbation γ is considered as a model hyper-parameter related to the prior probability of a composite model state comprising of pairs of pure model states. We propose a gradient-based scheme to optimise with respect to this parameter in one of the following sections.

A GMM with likelihood expression of the form of eq. (4.26), includes quantum effects and is based on states constructed by superposing the model states corresponding to the mixture components of a classical GMR model.

Ultimately the introduction of the quantum disturbance achieves a more elaborate model structure, accounting for richer data dependencies. As a result, the QMM has been shown to outperform conventional mixture models in image segmentation applications [Tanaka & Tsuda, 2008] and in regression as presented later in this chapter.

4.1.6.1 *Model Inference*

Considering a dataset consisting of N samples $\boldsymbol{Y} = \{\boldsymbol{y}_n\}_{n=1}^N$, model parameter estimation is possible for this model by means of MLE. More specifically, by directly optimising the model likelihood of eq. (4.26) with respect to its parameters. The necessary extrema conditions are given as follows [Tanaka & Tsuda, 2008]:

$$\pi_k \backsimeq \frac{1}{N}\sum_{n=1}^{N}\phi_{nk}, \quad \boldsymbol{\mu}_k = \frac{\sum_{n=1}^N \phi_{nk}\boldsymbol{y}_n}{\sum_{n=1}^N \phi_{nk}} \tag{4.29}$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_{n=1}^N \phi_{nk}\left(\boldsymbol{y}_n - \boldsymbol{\mu}_k\right)\left(\boldsymbol{y}_n - \boldsymbol{\mu}_k\right)^T}{\sum_{n=1}^N \phi_{nk}} \tag{4.30}$$

where

$$\phi_{nk} = \frac{\partial \ln tr\left\{e^{-\boldsymbol{H}(\boldsymbol{y}_n)}\right\}}{\partial B_{kk}} = \frac{tr\left\{\boldsymbol{X}_{kk}e^{-\boldsymbol{H}(\boldsymbol{y}_n)}\right\}}{tr\left\{e^{-\boldsymbol{H}(\boldsymbol{y}_n)}\right\}} \tag{4.31}$$

The above derivative follows directly due to linear response theory [Hertel, 2001] and the matrices $\boldsymbol{X}_{kk}$ are given by eq. (4.25).

Although during the course of this thesis we have frequently advocated in favour of Bayesian solutions (rather than MLE), that is nevertheless not always an achievable goal. The MLE, although not expected to be the optimal solution, is our only choice in this case. Note that many other successful models employ MLE and more notably the Boltzmann Machine.

4.1.6.2 *Estimating the quantum disturbance γ*

Estimating the quantum disturbance is very important for the algorithm's performance. Although in our experiments we have not employed optimisation of this parameter, we propose a gradient descent estimation for future applications of this method. The necessary likelihood derivative can be found as follows:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \gamma} &= \sum_{k,k'} \left[\sum_{n=1}^{N} \frac{\partial}{\partial B_{kk'}^{(n)}} \ln tr\left\{ e^{-\boldsymbol{H}(\boldsymbol{y}_n)} \right\} \frac{\partial B_{kk'}^{(n)}}{\partial \gamma} - N \frac{\partial}{\partial D_{kk'}} \ln tr\left\{ e^{-\boldsymbol{F}} \right\} \frac{\partial D_{kk'}}{\partial \gamma} \right] \\ &= \sum_{k,k'} \left[\sum_{n=1}^{N} \frac{tr\left\{ \boldsymbol{X}_{kk'} e^{-\boldsymbol{H}(\boldsymbol{y}_n)} \right\}}{tr\left\{ e^{-\boldsymbol{H}(\boldsymbol{y}_n)} \right\}} \frac{\partial B_{kk'}^{(n)}}{\partial \gamma} - N \frac{tr\left\{ \boldsymbol{X}_{kk'} e^{-\boldsymbol{F}} \right\}}{tr\left\{ e^{-\boldsymbol{F}} \right\}} \frac{\partial D_{kk'}}{\partial \gamma} \right] = \\ &= \sum_{k \neq k'} \left[\sum_{n=1}^{N} \frac{tr\left\{ \boldsymbol{X}_{kk'} e^{-\boldsymbol{H}(\boldsymbol{y}_n)} \right\}}{tr\left\{ e^{-\boldsymbol{H}(\boldsymbol{y}_n)} \right\}} - N \frac{tr\left\{ \boldsymbol{X}_{kk'} e^{-\boldsymbol{F}} \right\}}{tr\left\{ e^{-\boldsymbol{F}} \right\}} \right] \end{aligned} \tag{4.32}$$

with

$$D_{kk'} = \begin{cases} \ln \pi_k & , k = k' \\ \gamma & , otherwise \end{cases} \tag{4.33}$$

Note that the quantum disturbance γ is a positive small number and has a profound effect on the model, thus it is necessary to employ constrained gradient descent with a small learning rate.

4.1.6.3 *Predictive Density*

Following model training and for the purpose of utilising the model for predictions in a LbD setting, it is required to also obtaine a predictive probability density. For the predictive density of conventional GMMs we refer to Section 3.2.5.1 and for Quantum Mixture Regression (QMR) to [Chatzis, Korkinof, and Demiris, 2012c].

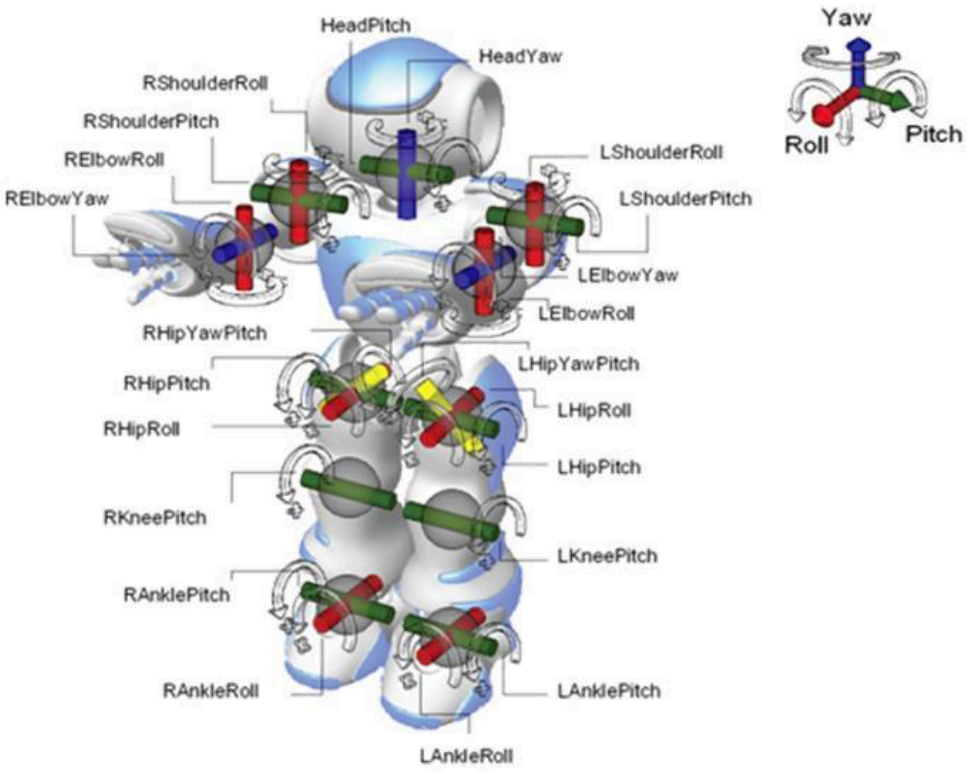
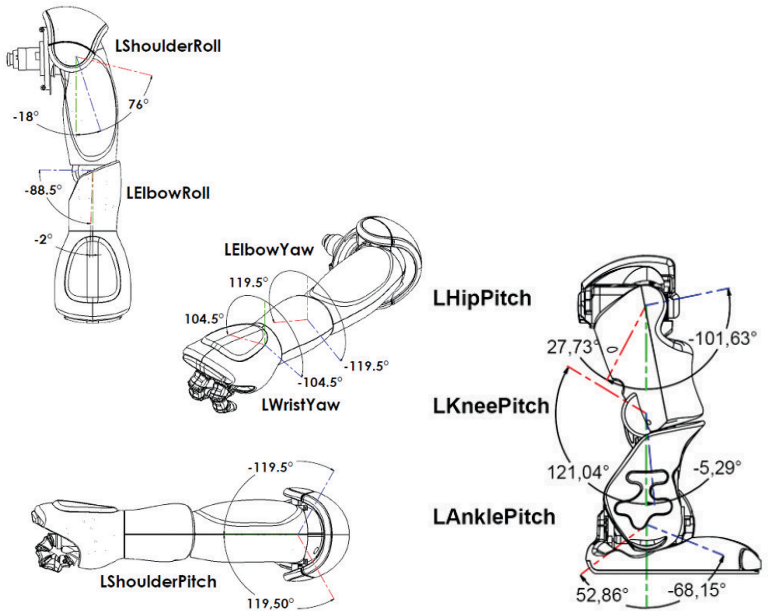


Figure 4.3: Aldebaran's NAO robotic platform academic edition. It has 27 DoF and is equipped with two cameras, microphones, speakers, ultrasound and touch sensors.

4.2 EMPIRICAL EVALUATION

In this section, we present an empirical evaluation of the models presented in this chapter, namely the Dirichlet Process Mixture (DPM) model and the Quantum Mixture Model (QMM), in a series of applications pertaining to robot LbD. We compare both approaches to state-of-the-art methods in the field of robotics, such as EM-GMR [Ghahramani & Jordan, 1994] and local Gaussian Process Regression (LGPR) [Nguyen-Tuong & Peters, 2008]. We have considered three distinct application scenarios with potential practical applicability under a one-



(a) Left arm joint specifications. (b) Left leg joint specifications.

Figure 4.4: NAO robot joint specifications and corresponding ranges of movement, presented in degrees. (See also nao) and Table 4.2

and multi-shot learning setting. In all experiments, we have utilised joint angle data, which present a greater challenge for learning algorithms (compared, e.g., to end-effector data). Our source codes have been developed in Matlab and run on a PC with an Intel Core i7 3.4 GHz CPU, 16 GB RAM, running Ubuntu Linux.

ROBOTIC PLATFORM: For the purposes of our experimental evaluation, we have employed the NAO robotic platform academic edition, a humanoid robot with 27 DoF [Gouaillier et al., 2009] (fig. 4.3). The robot is equipped with two cameras, one looking downwards and one forwards. There are 3 microphones located asymmetrically on the head and two speakers, as well as ultrasound and touch sensors. It is small and quite portable and able to walk. The robot is not especially well-suited for manipulation as it only has 1-DoF grippers on each hand.

The training trajectories were presented to the robot by means of kinesthetics; that is manually moving the robot's arms and recording the joint angle positions.

During this procedure, joint position sampling was conducted, with the sampling rate set to 20 Hz. The robot joints actively participating in each experiment varied according to the specification of the performed motion types (for details see Table 4.2 and fig. 4.4). The aforementioned joint angle data were collected using a fully threaded NAO-Matlab communication protocol developed in Python.

	One-shot		Multi-shot		Units
Task	#Points	#Dims	#Points	#Dims	
Block	445	11	2175	8	rad
Ph.Ed.	355	5	849	5	rad
LF8s	242	6	717	5	rad

Table 4.1: Datasets of the utilised datasets for one- and multi-shop LbD.

4.2.1 *Experimental Setup*

In our experiments, joint angle positions serve as the predictor variable $\boldsymbol{x}_n$, whereas as a response variable we have used the velocity vector $\boldsymbol{v}_n$ that should be imposed on the robot joints so as to remain on the learnt trajectory.

MULTI-SHOT LEARNING :

In this experiment, we used multiple demonstrations of each task, so as to capture the variability of human action and evaluate our model's ability to generalise learned trajectories. Training was conducted using three out of the four available sequences, and the generalisation capabilities of the compared methods were evaluated using the fourth demonstration for testing.

Due to the temporal misalignment across demonstrations, we pre-processed the sequences using Dynamic Time Warping (DTW) [Myers & Rabiner, 1981],

a method first used in speech recognition for signal alignment. Subsequently, we used a low-pass filter to smooth out anomalies resulting from the alignment.

ONE-SHOT LEARNING :

In the case of the one-shot learning scenario, the aim is to evaluate our approach under a sparser setting. Testing was conducted by adding uniformly distributed noise $\mathcal{U}(0,1)$ to the initial points of the training data sequence and running the algorithms so as to regenerate the learnt trajectories. Considering the maximum joint ranges presented in Table 4.2, it becomes apparent that the induced noise levels give rise to a substantial deviation from the initial trajectory starting points, thus constitute a significant disturbance. For the purpose of this experiment we have regarded one demonstration from each task prior to applying temporal alignment.

In Table 4.1, we present some details regarding the number of data-points and the dimensionality of the datasets utilised.

In order to measure the performance of the evaluated algorithms, we utilise the Mean Square Error (MSE) along the entire sequence length as our error metric. We have excluded the time component from the MSE calculation, due to its trivial form.

We have repeated our experiments for various number of model states K to examine how models' performance is affected by this selection. We experimented with values of K greater than 25, and low enough to ensure that the number of estimated model parameters (i.e. $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$, and the π_k) does not exceed the number of training data points. This is done so as to avoid the possibility of overfitting for the GMM, as suggested in [McLachlan & Krishnan, 2000]. Note that in the case of DPMR, K represents the truncation threshold, thus the maximum number of model states, rather than the final number of states which is inferred. We refer to the final number of states inferred by the DPM as "Active Components".

Joints/Tasks	Lazy figure 8	Ph.Education	Blocking	Range (rads)
LShoulderPitch	√			[−2.0857, +2.0857]
LShoulderRoll	√		√	[−0.3142, +1.3265]
LElbowYaw	√		√	[−2.0857, +2.0857]
LElbowRoll	√		√	[+1.5446, +0.0349]
RShoulderPitch			√	[−2.0857, +2.0857]
RShoulderRoll			√	[−0.3142, +1.3265]
RElbowYaw			√	[−2.0857, +2.0857]
RElbowRoll			√	[+1.5446, +0.0349]
LHipPitch		√		[−1.7739, +0.4841]
LAnklePitch		√		[−1.1895, +0.9228]
RHipPitch		√		[−1.7739, +0.4841]
RAnklePitch		√		[−1.1895, +0.9228]

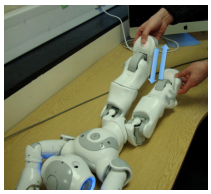
Table 4.2: NAO robot active joints in each experiment and corresponding range of movement.

In order to account for the effect of random initialisations on the model performance and to accumulate statistics, we repeat training and testing multiple times for each number of model states K. When possible the evaluated algorithms receive common initialisation at the beginning of each execution. We present means and standard deviations of the MSE accumulated over 100 runs for one-shot and 50 runs for multi-shot for each K.

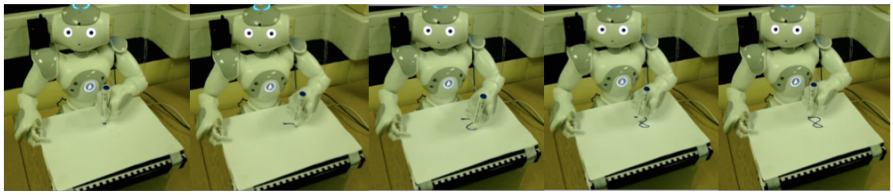
We briefly describe the conducted experiments below:

LAZY FIGURE 8S :

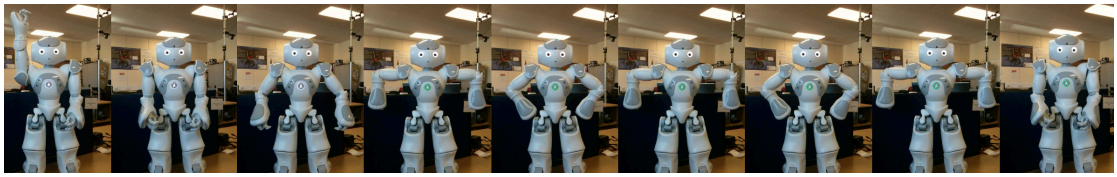
In this experiment, we evaluate the considered methods in terms of their applicability to teaching a robot by demonstration how to draw a complex figure. The considered figure comprises a lazy figure 8 (fig. 4.5b), a task which



(a) Ph.E.



(b) Drawing lazy figure 8's.



(c) Communicative gesture for the violation "Blocking"

Figure 4.5: Illustration of the motion trajectories utilised.

is perceived as a classical benchmark for pattern generation methodologies [Pearlmutter, 1995; Zegers & Sundareshan, 2003]. Each demonstration consists of joint angle data from drawing 3 consecutive L8s.

UPPER BODY MOTION :

In the case of upper body motion, our experiments involve a higher number of joints, thus further increasing the dimensionality and, consequently, the complexity of the addressed problem. We examine learning and reproduction of a communicative gesture used by Basketball officials, with potential applicability in the case of a robotic referee. We have chosen a gesture that poses a challenge to the LbD algorithm in terms of the implied motion complexity, namely the sign concerning the violation "blocking" (fig. 4.5c).

LOWER BODY MOTION :

Finally, we examine an experimental case involving movement of the lower robot body, simulating a lower abdominal muscle exercise (fig. 4.5a). This is one of the scenarios under investigation of the ALIZ-E EU FP7 project (http://www.aliz-e.org/), where robots are used as companions to diabetic and/or obese children in paediatric ward settings over extended time periods,

and learn along with the children various sensorimotor activities (e.g. dance, games, and physical exercises) so that they can practice and improve together.

4.2.2 *Model Accuracy Results*

Examining the accuracy of representation for all evaluated methods we present analytical results for different choices of the number of component densities K in figs. 4.6 and 4.7 for the one and multi-shot scenarios respectively. In Tables 4.3 and 4.4 we present the best achieved MSE results for all evaluated methods. Results were accumulated over 100 for the one-shot and 50 repetitions for the multi-shot scenario with different random initialisation across repetitions but common among methods.

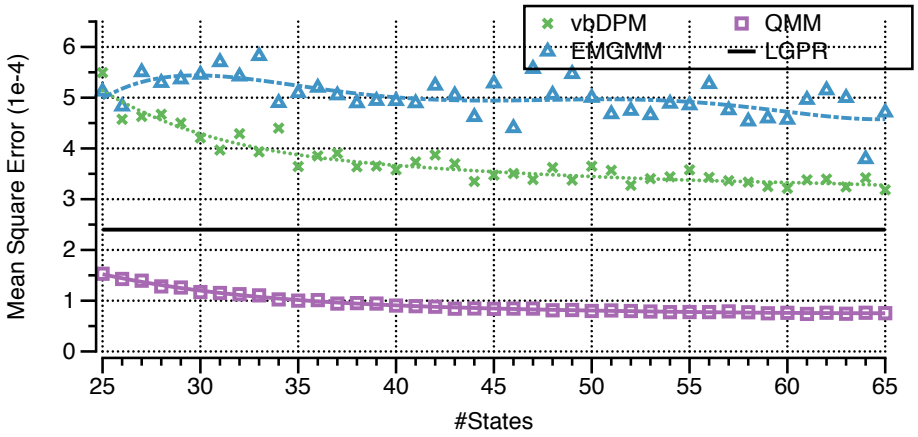
ONE-SHOT LBD :

Regarding the one-shot learning experiment, we can observe that the QMM and LGPR are the best performing methods. Comparing the two methods, LGPR achieves the best performance in the "Blocking" experiment, around 10% better than the QMM. In the Ph. Education experiment the QMM achieves approximately 20% higher accuracy. Finally, the difference between the two methods is significant in the case of the Lazy Figure 8s experiment, where the QMM performs around 3 times better than the LGPR.

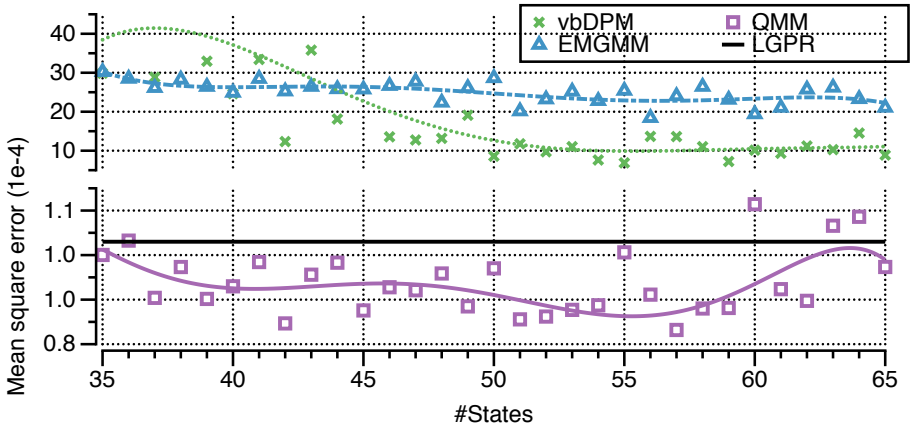
As far as the GMM and DPM are concerned, we can observe that the latter generally outperforms the former achieving as much as 2 times better accuracy. An exception is the "Blocking" experiment, where the GMM performs better. Both methods are outperformed by the LGPR and QMM by up to an order of magnitude.

MULTI-SHOT LBD :

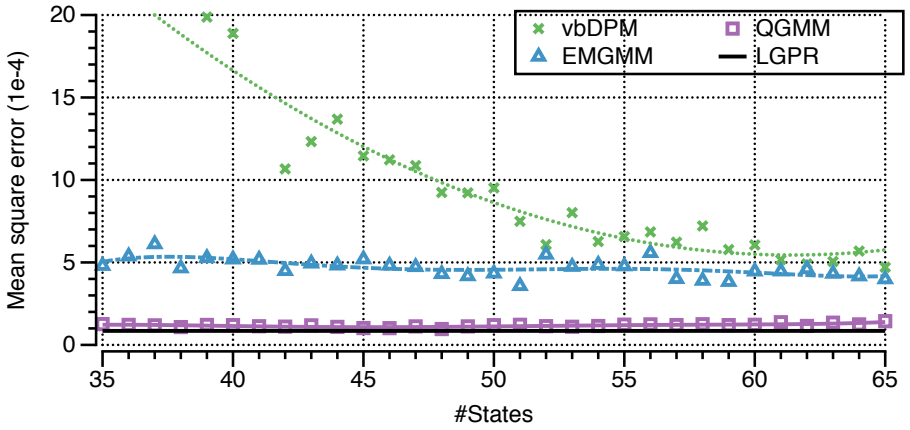
In the multi-shot LbD scenario, the QMM achieves again the best performance in two out of the three experiments, followed this time by the DPM which



(a) Lazy figure 8s



(b) Ph. Ed. exercise



(c) Blocking gesture

Figure 4.6: **One-shot LbD**: MSE plots of all evaluated methods against the number of components. Note that in the case of the DPMR the x-axis represents the truncation threshold as the number of components is determined dynamically.

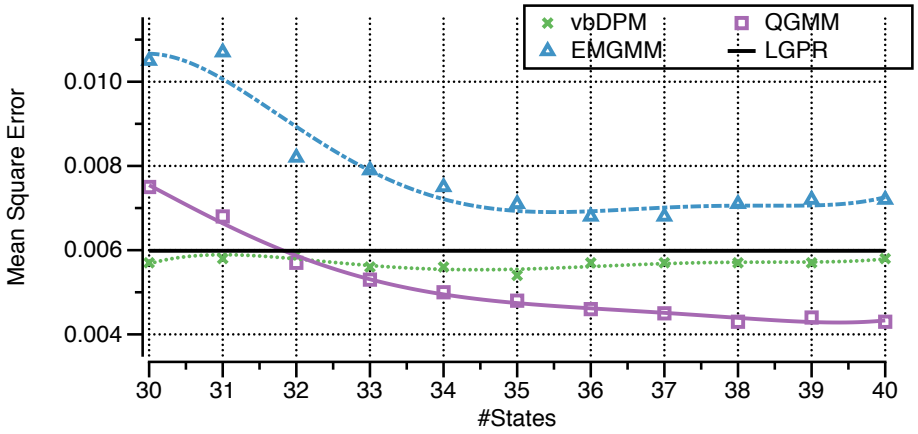
performs very well under this setting. We should note that the DPM achieves the best performance in the "Blocking" dataset.

As far as the other evaluated methods are concerned, LGPR performs generally better than the GMM in all experiments, apart from the "Blocking" where we have a reversal of roles.

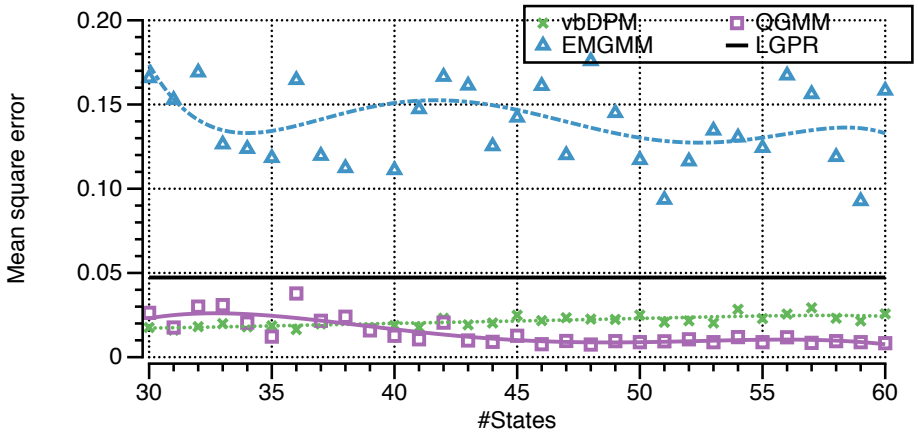
From a theoretical perspective, the DPM's higher accuracy compared to the GMM can be attributed to the fully Bayesian treatment of the mixture model employed in this case. It should be noted that the Variational Bayesian (VB) inference yields full posterior estimates for all model parameters and some hyper-parameters, which should be a significant advantage over the Expectation Maximisation (EM) algorithm. The use of a Dirichlet Process (DP) prior on the other hand, is not theoretically expected have a significant impact on the accuracy, but is rather employed for its component shrinkage property and would result in reduced computational complexity. The aforementioned theoretical considerations are supported by our findings in most experiments.

Commenting on the performance of the LGPR, we observed better performance under the sparser one-shot LbD setting. This could be explained by the fact that LGPR is effectively a model employing simpler inference and as such it may be more effective in data deprived settings.

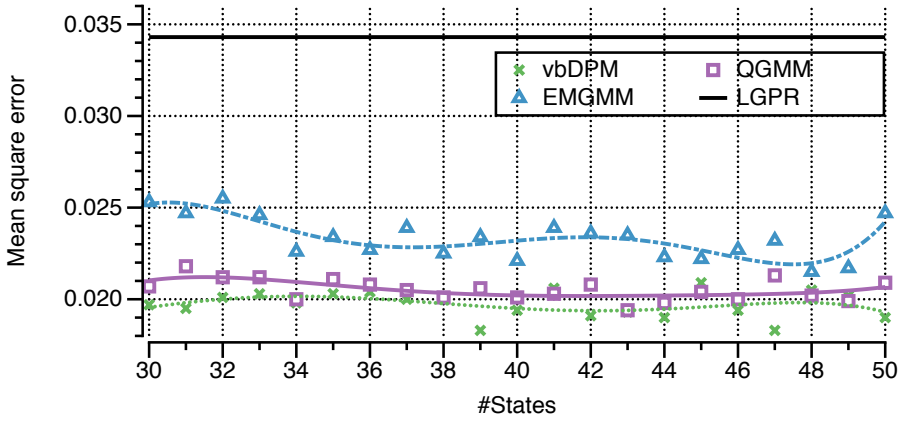
Finally, the introduction of quantum states is highly beneficial as illustrated by the consistent performance of the QMM under both one- and multi-shot settings. We can observe that the QMM provides a significant improvement to the conventional GMM and also outperforms the fully Bayesian DPM in most occasions. We can therefore conclude that the method's robustness and reliability in a variety of settings, renders it especially well-suited and appealing for robotic applications.



(a) Lazy figure 8s.



(b) Ph. E. exercise.



(c) Blocking gesture.

Figure 4.7: **Multi-shot LbD**: MSE plots of all evaluated methods against the number of components. Note that in the case of the DPMR the x-axis represents the truncation threshold as the number of components is determined dynamically.

	One-shot LbD MSE $\left(10^{-4}\right)$			
Task/Method	EMGMM	vbDPM	LGPR	QMM
Blocking	3.83 ± 2.6	4.71 ± 1.0	**0.85**	0.96 ± 0.12
Ph. Education	18.44 ± 10.2	6.88 ± 3.6	1.03	$\mathbf{0.83 \pm 0.24}$
Lazy figure 8s	3.79 ± 0.26	3.19 ± 0.88	2.40	$\mathbf{0.74 \pm 0.04}$

Table 4.3: **One-shot LbD**: Best mean and std of MSE results for all evaluated methods.

	Multi-shot learning MSE $\left(10^{-2}\right)$			
Task/Method	EMGMM	vbDPM	LGPR	QMM
Blocking	2.15 ± 0.24	$\mathbf{1.83 \pm 0.07}$	3.43	1.94 ± 0.19
Ph. Education	9.27 ± 6.71	1.60 ± 0.032	4.73	$\mathbf{0.75 \pm 0.029}$
Lazy figure 8s	0.64 ± 0.098	0.56 ± 0.013	0.60	$\mathbf{0.36 \pm 0.013}$

Table 4.4: **Multi-shot LbD**: Best mean and std of MSE results for all evaluated methods.

4.2.3 *Generalisation*

In this section we aim at evaluating the robustness of the proposed approaches to training data deprivation and their knowledge generalisation capabilities on wider testing sets. We do so by increasing the number of demonstrations available for testing and correspondingly decreasing the number of training demonstrations.

More specifically, we have first segmented the demonstrated trajectories to include one L8 draw per demonstration, rather than 3 as in the previous section. This gives rise to a dataset which consists of 12 demonstrations containing a single L8 draw each, in contrast to 4 demonstrations containing 3 L8s each. Note that the resulting dataset is much simpler than the one we have utilised in the previous section.

We then gradually decrease the amount of available training information and simultaneously increase the testing information. This setting is meant to assess

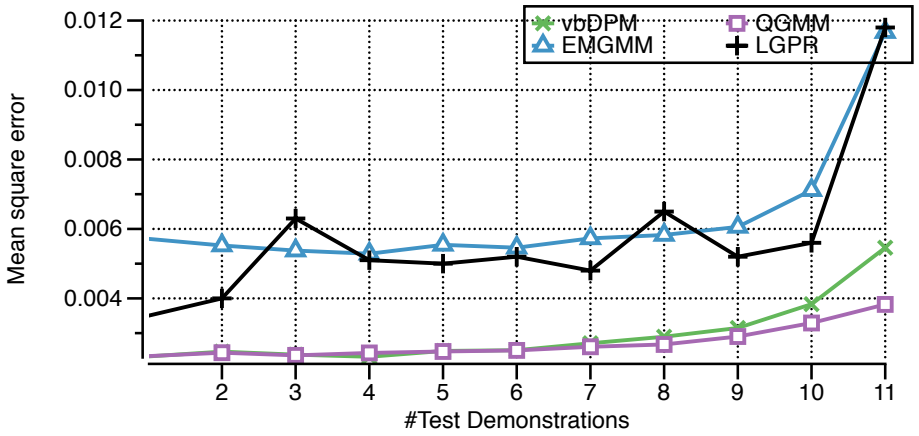


Figure 4.8: **Multi-shot L8s:** In this figure we depict the test MSE, as we gradually increase the testing set (and decrease the training set). For the purpose of this experiment, we have separated the draws of the L8s from 3 to 1 per demonstration giving rise to simpler trajectories.

the robustness of the evaluated algorithms under training data deprivation and their generalisation ability on a wider testing set at the same time.

Our results are presented in fig. 4.8. As we can observe, increasing the testing set results in a slow, but gradual, increase of the test MSE, up to the point where the training set becomes insufficient for the algorithms to learn from. The DPM and QMM perform similarly and we do not perceive the performance differences of the previous section due to the simplicity of the task. However, the QMM has a clear advantage under severe training data deprivation. This may be attributed to the large number of parameters of the Bayesian DPM, which makes it especially data savvy. Finally, the conventional GMM performs slightly worse, but comparably, to the LGPR, and both methods perform significantly worse that the DPM and the QMM.

4.2.4 *Computational Costs*

In the following paragraphs we aim to outline the computational costs introduced by the algorithms we have considered in this chapter. This analysis is by no means meant to be strict or accurate, but rather to serve as a rough guideline on the expected cost of each algorithm.

The exact complexity and computational cost is heavily dependent on the particular implementation choices of Matlab and other libraries we have used during the development of our source code. For instance, matrix multiplication or inversion has a nominal complexity of $\mathcal{O}(N^3)$, but can be reduced to $\mathcal{O}(N^{2.373})$ by using highly optimised alternative numerical algorithms. As Matlab is not an open-source software we would not be able to know such implementation details.

In the analysis that follows we also distinguish between training and testing cost. The former is an indication of the time spent for training and reflects an offline use of resources. The latter, on the contrary, affects the applicability of our algorithms after they have been deployed in a robotic system with limited on-board resources.

TRAINING COST

- **LGPR**:
  LGPR is the model with the lowest number of parameters in this case. Updating the local GPs involves the computation of a kernel similarity with respect to each cluster center and subsequently adding this point to the Gram matrix of the most likely local model. A computationally efficient way to perform this action is proposed in [Nguyen-Tuong & Peters, 2008] and is dominated by a Cholesky decomposition of complexity $\mathcal{O}(N_k^2)$, where N_k is the number of data-points assigned to the k^{th} local model.

- **EM-GMM**:
  The EM parameter update equations for the GMM are given in eq. 4.29-4.30.

They are as well dominated by a vector outer product per datapoint of complexity $\mathcal{O}(D^2)$, where however D is the dimensionality of the modelled data-point. However, the parameter vector $\boldsymbol{\vartheta}_k = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ contains $(1 + D + D^2 + N)$ elements per mixture component, which is typically higher than LGPR with the same number of components. This in brief means that in typical situations we expect the EM-GMM to induce higher computational cost during training than LGPR. This however may be reversed if $N_k >> D$.

- **EM-QMM**:
  The EM updates for the QMM, exhibit almost identical complexity to the ones for conventional GMMs. The difference is an extra matrix exponential that adds a moderate computational cost to the algorithm. In the case of very efficient matrix exponential calculations this cost is expected to be minimised. In practice, we have observed a slightly higher ($\sim 25\%$) training time for the QMM, compared to the GMM.

- **DPM**:
  In contrast to the aforementioned models the DPM is a fully Bayesian model and we employ VB rather than EM. As a result, the DPM typically enjoys higher accuracy, which however comes at a higher computational cost. This is because the number of parameters of the DPM is typically larger.
  More specifically, the VB update equations for the DPM are given in [Chatzis, Korkinof, and Demiris, 2012a]. The corresponding parameter updates are dominated again by a vector outer product of complexity $\mathcal{O}(D^2)$, but the parameter vector has the following dimensionality: $3 + K(4 + 2D + 2D^2 + N)$. For the aforementioned reason the DPM is typically expected to exhibit higher computational costs, roughly twice as high as the EM-GMM (also depending on other factors).

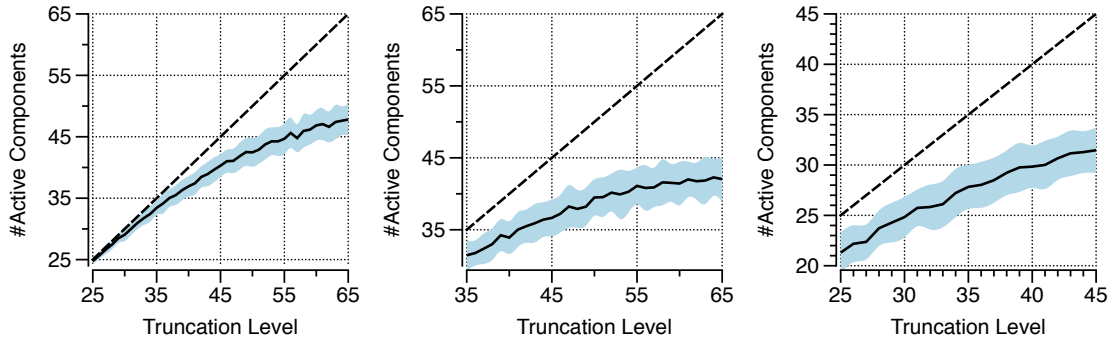
TESTING COST During testing, we perform an estimate of the response, given the predictor variable. More specifically we use the conditional predictive distribution of the response variable, given the predictor variable.

- **EM-GMR & DPMR**:
  For GMR, the predictive distribution is given in eq. 3.3-3.6 and for the DPM in [Chatzis, Korkinof, and Demiris, 2012a]. In both cases, the computational cost of the predictive distribution is almost identical and dominated by a matrix inversion, with nominal complexity $\mathcal{O}(KD_p^3)$, where D_p is the dimensionality of the predictor variable and K is the number of mixture components.

- **EM-QMR**:
  Similarly to the case of training, the testing cost of the QMR is identical to the one of conventional GMR with the addition of a matrix exponent per mixture component. This slightly increases the computational time we experience during training again by $\sim 25\%$.

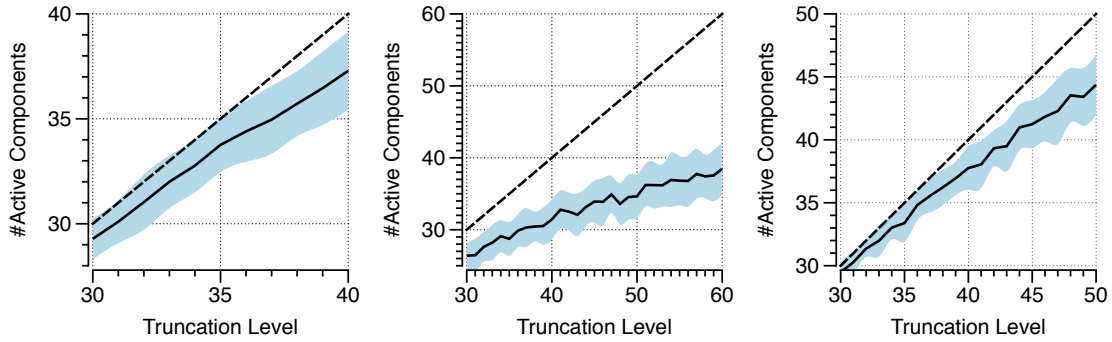
- **LGPR**:
  The use of the LGPR during testing is not as favourable from a computational point of view as in training. Direct inference is performed, which however does not scale well with the number of data-points per local model. More specifically, prediction requires a kernel computation and a matrix inversion per local model. Complexity is dominated by the matrix inversion of $\mathcal{O}(N_k^3)$ and scales cubically with the number of data-points assigned to each local GP N_k, which would typically be higher than the dimensionality of the predictor variable D_p of GMR. Thus the model is expected to require more resources during testing than GMR.

BENEFIT FROM USING THE DPM The benefit of using the DPM stems from the shrinkage property of the Dirichlet process prior, which achieves automatic model selection and effectively reduces the number of components. More specifically, after training the model, we can easily identify "active" and "inactive" components based on their component membership posteriors. The resulting "active" components are dynamically estimated in a data-driven manner, as we have previously mentioned.



(a) Lazy figure 8s. (b) Ph. E. exercise. (c) Blocking gesture.

Figure 4.9: **One-Shot LbD**: Number of DPMR active components, with their corresponding standard deviation.



(a) Lazy figure 8s. (b) Ph. E. exercise. (c) Blocking gesture.

Figure 4.10: **Multi-Shot LbD**: Number of DPMR active components, with their corresponding standard deviation.

In fig. 4.9 and 4.10, we present the inferred number of active components for the one- and multi-shot learning cases respectively. We can observe that rising the VB truncation threshold (x-axis) results in an increase of the number "active" components only up to a certain extent, beyond which their number exhibits saturation.

The benefit in terms of the testing computational cost is exactly analogous to the decrease in the number of model components achieved, as depicted in fig. 4.11, where we have plotted both quantities in common axis.

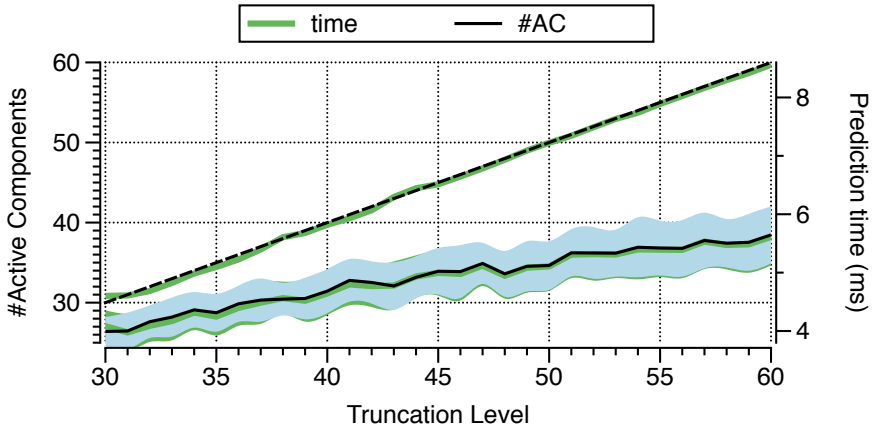


Figure 4.11: Prediction time plotted in common scale with the number of active components. As expected, the quantities are exactly analogous.

4.2.5 *Accuracy VS computational time*

As we have previously detailed, the DPM achieves good accuracy in generally lower computational time during testing. For that reason it is to be preferred in applications where the execution time is a critical factor.

It is worth, however, to consider in more detail the benefits of QMR against the computational costs it induces compared to conventional GMR. Let us, for that purpose, consider the setting employed in Section 4.2.3. We have recorded the mean testing time resulting from 50 runs of GMR and QMR, as presented in fig. 4.12, using a MacBook Pro, dual-core Intel Core i7 2.8GHz, 8GB of RAM, running Matlab 2014b under OSX 10.10.

We can observe that the extra computational time of QMR is ranging from 22%-25% compared to GMR. Increasing the size of the testing set, amounts to increased test time, however the percentage of difference between QMR and GMR remains the constant.

The 25% extra computational cost, compared to the roughly twice better performance, results in around 2% increase in accuracy for every 1% of additional computational cost.

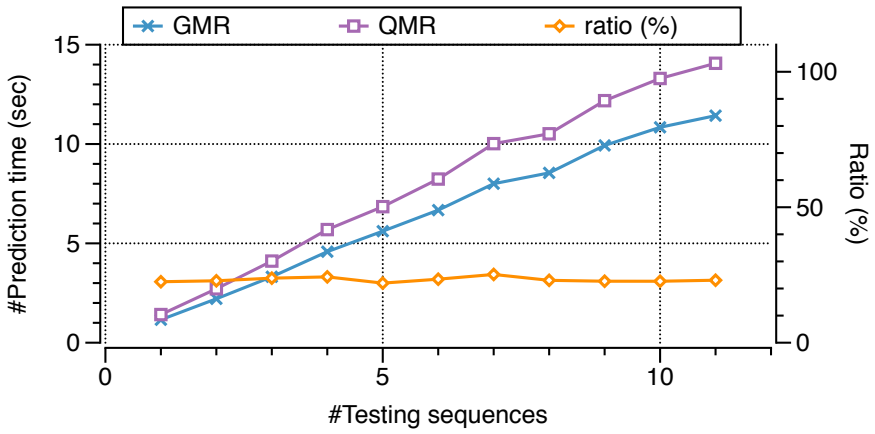


Figure 4.12: Comparison of prediction (testing) computational time for GMR and QMR.

4.3 CONCLUSION

In this chapter, we have empirically evaluated two promising approaches that aim at more effective real-life applications of robot LbD by addressing two key issues, namely the model size selection and the accuracy of representation. We have compared the models with established methods frequently employed in similar settings. We are able, on one hand using the DPM to achieve a more parsimonious task representation, without compromising results in terms of the model predictive power. On the other hand, with the QMM we achieve state-of-the-art reconstruction accuracy for an insignificantly higher computational burden. We demonstrated those attributes in a variety of one- and multi-shot Learning by Demonstration (LbD) scenarios using the NAO robot academic edition and on tasks with potential practical applicability.

5

ONLINE LEARNING

Online and big-data solutions have become increasingly popular in the past few years, with a rekindled interest in stochastic optimisation [Bottou, 1998, 2010]. This may be attributed to the huge abundance of data triggered by advances in sensor equipment, surveillance techniques and the logging of human behaviour, i.e. social media. Pattern extraction from big amounts of data necessitates a more adaptive approach to conventional offline learning algorithms. Thus, rather than regarding the available data in its entirety, which would induce significant computational costs and long delays, we need inference algorithms able to progressively refine the model estimates by processing a small subset of the data at a time. The benefit of such an approach is twofold. We are first able to use the so obtained model almost immediately to perform predictions and we expect that those predictions will become more accurate with time. Second, we are able to capture time evolving phenomena with a more dynamically adaptive approach. An important trade-off with online approaches is that we are frequently forced to compromise in terms of accuracy.

In the field of robotics, in particular, where the amount of available data is usually manageable, the importance of online approaches lies into their dynamic nature. Indeed, there is a strong demand in the field for algorithms that would adaptively learn and refine their model of reality so as to be able to integrate robotic application more successfully and naturally into human life.

Nevertheless, online versions of complex algorithms still remain a formidable challenge, as well as a field of vibrant research. In this regard, we believe that robotics could highly benefit from recent advances in stochastic optimisation and that online methods could provide the momentum needed towards more effective real-life robotic applications. This belief is reinforced by an increasing amount of work towards formulating online algorithms for robotic applications, such as learning robot dynamics [De La Cruz et al., 2012] and kinematics [Damas & Santos-Victor, 2012; Droniou et al., 2012].

Motivated by the aforementioned points, we present a novel online training algorithm for the Quantum Mixture Model (QMM). The proposed approach builds upon recent advances in several fields, such as machine learning, quantum statistics and stochastic optimisation, to yield a powerful framework for incremental learning and prediction from multiple demonstrations. We also present an effective component production and pruning scheme to facilitate learning, in cases when the data depart form the iid[1] assumption.

We demonstrate the efficacy of our algorithm in a synthetic example, a series of benchmark datasets and a learning by demonstration task.

5.1 ONLINE LEARNING FOR QUANTUM MODELS

We envision an online training algorithm for the Quantum Mixture Model (QMM) with the following key traits:

- Capacity to processes data on-the-fly.
- No need to store processed data-points.
- No need to iterate through the dataset.

1 iid: independent and identically distributed.

To achieve this purpose we shall orient ourselves towards stochastic gradient descent methods, known to posses the desirable traits mentioned. More specifically, averaged and second-order stochastic gradient algorithms have been shown to be asymptotically efficient, even after a single pass through the training set [Bottou, 1998] and data-points may be discarded as soon as they have been used for training.

The basic notion is that we move towards the direction of steepest ascend of the objective function $\mathcal{L}$ on the parameters manifold, and this is captured by the following updates:

$$\boldsymbol{\vartheta}^{(t+1)} = \boldsymbol{\vartheta}^{(t)} + \eta_t \nabla_{\boldsymbol{\vartheta}} \mathcal{L}(\boldsymbol{y}_t, \boldsymbol{\vartheta}^{(t)}) \tag{5.1}$$

where as $\boldsymbol{\vartheta}^{(t)}$ and $\boldsymbol{y}_t$ we denote the parameter and data vectors at time t respectively, and $\mathcal{L}(\cdot)$ is the model log-likelihood in the case of probabilistic models, but can also be any other objective function with respect to which we wish to optimise. The algorithm is guaranteed to asymptotically converge, provided that the stochastic weights satisfy the following necessary conditions [Bottou, 1998]:

$$\sum_{t}^{+\infty} \eta_t = +\infty \tag{5.2}$$

$$\sum_{t}^{+\infty} \eta_t^2 < +\infty \tag{5.3}$$

As the gradient of the objective function is evaluated at a single data point, it constitutes a noisy estimate of direction. For that reason, second-order stochastic gradient ascend algorithms are employed to speed-up convergence. They utilise higher order information regarding the parameter space manifold. A popular choice is to normalise the gradient with the Fisher Information Matrix (FIM) [McLachlan & Krishnan, 2000], a quantity also known as the natural gradient [Amari, 1998]:

$$\boldsymbol{\vartheta}^{(t+1)} = \boldsymbol{\vartheta}^{(t)} + \eta_t \mathcal{I}_{\boldsymbol{\vartheta}}^{-1}(\boldsymbol{\vartheta}^{(t)}) \nabla_{\boldsymbol{\vartheta}} \mathcal{L}(\boldsymbol{y}_t, \boldsymbol{\vartheta}^{(t)}) \tag{5.4}$$

where the FIM is defined as follows:

$$\mathcal{I}_{\boldsymbol{\vartheta}}(\boldsymbol{\vartheta}) \triangleq \mathbb{E}_{p(\boldsymbol{y}_t)} \left[\left(\nabla_{\boldsymbol{\vartheta}} \mathcal{L}(\boldsymbol{y}_t, \boldsymbol{\vartheta}) \right) \left(\nabla_{\boldsymbol{\vartheta}} \mathcal{L}(\boldsymbol{y}_t, \boldsymbol{\vartheta}) \right)^T | \boldsymbol{\vartheta} \right] \tag{5.5}$$

or under certain regularity conditions coincides with the expectation of the Hessian:

$$\mathcal{I}_{\boldsymbol{\vartheta}}(\boldsymbol{\vartheta}) \triangleq -\mathbb{E}_{p(\boldsymbol{y}_t)} \left[\nabla^2_{\boldsymbol{\vartheta}} \mathcal{L}(\boldsymbol{y}_t, \boldsymbol{\vartheta}) | \boldsymbol{\vartheta} \right] \tag{5.6}$$

Natural gradient ascend algorithms have been popular lately and successfully employed in formulating online training algorithms for a few well-established statistical models, such as the Latent Dirichlet Allocation (LDA) [Hoffman et al., 2010] and the Hierarchical Dirichlet Process (HDP) [Wang et al., 2011]. The reason is that the natural gradient captures the Riemannian structure of the parameter space of a statistical model and the FIM is the only invariant measure of that structure.

The online EM outlined in the next section yields updates of similar form to the above, which have been proven to asymptotically converge to the natural gradient descent updates.

5.1.1 *Proposed Approach*

5.1.1.1 *Online EM*

However, a shortcoming of stochastic gradient ascend is that the updates may not necessarily meet all constraints and enforcing certain constraints may be very difficult. The main problem is focused on the covariance matrix, whose positive semi-definiteness is not guaranteed at each time-step and is very difficult to enforce through regularisation, a fact that severely impairs learning.

For that reason we shall employ a similar stochastic update algorithm, based on the EM algorithm, which guarantees that all constraints imposed are in fact met at each time step. The online EM was first introduced in [Sato & Ishii, 2000] for the NGN. It was subsequently generalised in [Cappé & Moulines, 2009], where the authors present a proof convergence and asymptotic equivalence to natural gradient ascend. Moreover, the algorithm performs simple and efficient updates, utilising the sufficient statistics of the exponential family so as to avoid redundant calculations.

The online EM for conventional GMMs is summarised by the following set of equations:

$$E-step: \begin{cases} s_t^{(k,1)} = \varphi_{kt} = p(k|\boldsymbol{y}_t; \boldsymbol{\Theta}^{(t)}) \\ \boldsymbol{s}_t^{(k,2)} = \varphi_{kt}\boldsymbol{y}_t & ,\forall k \\ \boldsymbol{s}_t^{(k,2)} = \varphi_{kt}\boldsymbol{y}_t\boldsymbol{y}_t^T \end{cases} \tag{5.7}$$

$$Updates: \begin{cases} S_t^{(k,p)} = (1-\eta_t)S_{t-1}^{(k,p)} + \eta_t s_t^{(k,p)} & ,\forall k,p \end{cases} \tag{5.8}$$

$$M-step: \begin{cases} \pi_k = S_t^{(k,1)}, \quad \boldsymbol{\mu}_k = \frac{\boldsymbol{S}_t^{(k,2)}}{S_t^{(k,1)}} \\ \boldsymbol{\Sigma}_k = \frac{\boldsymbol{S}_t^{(k,3)} - S_t^{(k,1)^{-1}}\boldsymbol{S}_t^{(k,2)}\boldsymbol{S}_t^{(k,2)^T}}{S_t^{(k,1)}} \end{cases} \tag{5.9}$$

where as $p(k|\boldsymbol{y}_t, \boldsymbol{\Theta}^{(t)})$ we denote the responsibility of cluster k for data-point $\boldsymbol{y}_t$.

The algorithm is guaranteed to asymptotically converge, provided that the stochastic weights η_t satisfy the conditions in eqs. 5.2 and 5.3.

During our empirical evaluation we have also compared against the more robust Student-t mixture model, for which we have adapted the online EM algorithm accordingly. The formulation is presented in eq. 5.11 for brevity and since it does not, to the best of our knowledge, appear elsewhere in the literature.

$$E-step: \begin{cases} s_t^{(k,1)} = \varphi_{kt} = \frac{\pi_k t(\boldsymbol{y}_t|\boldsymbol{\vartheta}_k)}{\sum_{j=1}^{K}\pi_j t(\boldsymbol{y}_t|\boldsymbol{\vartheta}_j)} \\ u_{kt} = \frac{\nu_k+D}{\nu_i+\Delta(\boldsymbol{y}_t;\boldsymbol{\mu}_k,\Sigma_k)} \\ s_t^{(k,2)} = \varphi_{ti}u_{kt} \\ \boldsymbol{s}_t^{(k,3)} = \varphi_{kt}u_{kt}\boldsymbol{y}_t \\ s_t^{(k,4)} = \varphi_{kt}\ln u_{kt} \\ \boldsymbol{s}_t^{(k,5)} = \varphi_{kt}u_{kt}\boldsymbol{y}_t\boldsymbol{y}_t^T \end{cases} \tag{5.10}$$

$$Updates: \begin{cases} S_t^{(k,p)} = (1-\eta_t)S_{t-1}^{(k,p)} + \eta_t s_t^{(k,p)} \end{cases} \tag{5.11}$$

$$M-step: \begin{cases} \pi_k = S_t^{(k,1)}, \quad \boldsymbol{\mu}_k = \frac{S_t^{(k,3)}}{S_t^{(k,2)}} \\ \boldsymbol{\Sigma}_k = \frac{S_t^{(k,5)} - S_t^{(k,2)^{-1}} S_t^{(k,3)} S_t^{(k,3)^T}}{S_t^{(k,1)}} \\ \nu_i: \ \ln\left(\frac{\nu_k}{2}\right) - \psi\left(\frac{\nu_k}{2}\right) + \psi\left(\frac{\nu_k^{n-1}+D}{2}\right) \\ -\ln\left(\frac{\nu_k^{n-1}+D}{2}\right) + \frac{S_t^{(k,4)} - S_t^{(k,2)}}{S_t^{(k,1)}} + 1 = 0 \end{cases} \tag{5.12}$$

where

$$t(\boldsymbol{y}_t|\boldsymbol{\vartheta}_k) = \frac{\Gamma\left(\frac{\nu_k+D}{2}\right)|\boldsymbol{\Sigma}_k|^{-\frac{1}{2}}}{(\pi\nu_k)^{\frac{D}{2}}\Gamma\left(\frac{\nu_k}{2}\right)\left[1+\nu_k^{-1}\Delta(\boldsymbol{y}_t;\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)\right]^{\frac{\nu_k+D}{2}}} \tag{5.13}$$

and ν_k are the degrees of freedom, D the dimensionality of the observations and $\Delta(\cdot)$ the Mahalanobis distance.

A proof of convergence for the online EM is provided in [Cappé & Moulines, 2009], which covers all non-quantum models, namely the oGMM and oSMM.

In the case of the online Quantum Mixture Model (oQMM), we propose for the point-wise responsibilities φ_{tk} to be approximately updated using the derivative shown Chapter 4, eq. 4.31. We have observed that this proposal works very well in practice, empirically establishing that the algorithm converges at least as fast and outperforms conventional models, in most occasions.

Concluding, all algorithms require an inhibition phase, equivalent to bootstrapping or warm-up. A period during which we update the global statistics, but suppress the M-step until sufficient information has been accumulated. The stability of each subsequent update can also be facilitated by processing the data in mini-batches, which accounts for a notable speed-up and is also reported to lead to performance gains [Liang & Klein, 2009].

5.1.2 *Component Manipulation (cm)*

Manipulating the number of components is important in online algorithms, especially so, when we depart from the assumption that the data are presented independently and identically distributed (i.i.d.) to the algorithm. An effective mechanism of component birth and decay can not only account for a substantial speed-up, by pruning unneeded components, but also for a performance boost, by adding components in areas misrepresented by the model.

Following the analysis in [Sato & Ishii, 2000], we propose the following mechanism.

5.1.2.1 *Component Pruning and Reset*

In case the mass of a component diminishes, we need to act to remove it or reset it.

Deleting components is straightforward, however we note that it is necessary to re-normalise global statistics $S^{(k,1)}$.

Resetting a component involves resetting it's mixing coefficient and covariance matrix. The mixing coefficient is set to $\frac{1}{K}$, where K is the number of components, and the covariance is set broad enough. We have randomly initialised the covariances to diagonal matrices drawn from a uniform distribution $\mathcal{U}[0, 0.1]$, however this is expected to differ according to the data variance.

In this case, we also perform an inverse E-step to update global statistics as follows:

$$inverse\ E-step : \begin{cases} S_t^{(k,1)} & = \pi_k^{(new)}, \forall k \\ \boldsymbol{S}_t^{(k,2)} & = S_t^{(k,1)} \boldsymbol{\mu}_k \\ \boldsymbol{S}_t^{(k,3)} & = S_t^{(k,1)} \boldsymbol{\Sigma}_k + \\ & \quad + S_t^{(k,1)^{-1}} \boldsymbol{S}_t^{(k,2)} \boldsymbol{S}_t^{(k,2)^T} \end{cases} \tag{5.14}$$

5.1.2.2 *Component Generation*

If one, or more, data-points are not represented with sufficiently large likelihood by any of the existing components, we create a new one centred at the mean of the misrepresented points and with a sufficiently large default covariance.

To be more specific, a data-point is considered as misrepresented by the current mixture model, when its likelihood, under all current components, is below a certain threshold. Similar to the previous case, the threshold has to be chosen wisely and depends on the model. We have used a likelihood threshold of 0.001 for Gaussian densities and 1 for Student-t densities.

The initial mixing coefficient is set to $\frac{1}{K}$.

5.2 EMPIRICAL EVALUATION

5.2.1 *Implementation and Setting*

Among the models evaluated, we have implemented the online versions of the GMM and SMM, LWPR and the oGP. For the Locally Weighted Projection Regression (LWPR) [Vijayakumar et al., 2005], we have used the authors' LWPR library [Klanke et al., 2008; Vijayakumar & Klanke]. For the online Gaussian

	M	N_m	α	γ	N_i
Synthetic	10	1	0.6	0.09	100
S.-A. [Schaal & Atkeson, 1998]	30	1	0.8	0.09	50
`pumadyn` [Corke, 1996]	30	1	0.8	0.09	50
L8s (t, x)	30	30	0.8	0.09	170
L8s $(t, x) + cm$	-	30	0.85	0.09	30
L8s $(\dot{x}, x)$	30	20	0.8	0.09	170

Table 5.1: Main hyper-parameters used in all experiments.

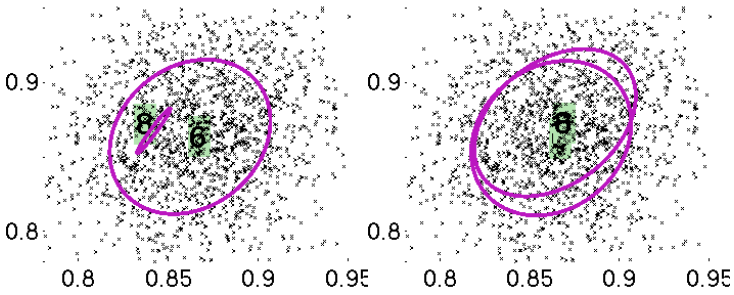
Process (oGP), we utilise a Matlab version of the OTL library that appears in [Soh & Demiris, 2012; Soh et al., 2012]. Testing was conducted in Matlab 2012b on an Ubuntu Linux PC, i7 3.4GHz, 16GB RAM.

Parameters for LWPR and the oGP were fixed to the recommended defaults. For the oGP, we have chosen a moderate number of 50 basis vectors for all experiments. All other methods are run with common parametrisation and initialisation for impartial comparisons. The main hyper-parameters for each experiment are presented in Table 5.1, where M is the number of components, N_m is the mini-batch size, α is the parameter of the stochastic coefficients $\eta_t = n^{-\alpha}$, γ is the quantum parameter and N_i is the length of the inhibition phase.

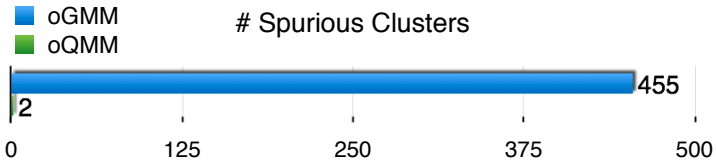
5.2.2 *Synthetic Experiment*

During our experimental evaluation and besides the performance gains achieved by the oQMM, we have also observed consistently higher numerical stability. This could be attributed to the more balanced component weighting induced by the quantum effects, as illustrated in the following synthetic example.

For the purpose of the experiment, we have randomly generated 20 datasets of 5000, 2D data-points, sampled from an equally weighted mixture of 3 Gaussians



(a) Online GMM. (b) Online QMM.



(c) Number of spurious clusters.

Figure 5.1: **Synthetic**: Component #8 has a low rank covariance matrix leading to numerical instabilities in higher dimensional datasets.

Method/Dataset	Schaal-Atkeson
LWPR	5.44 ± 0.72
oGP	40.7 ± 7.96
oGMM	4.55 ± 1.34
oSMM	5.20 ± 0.99
oQMM	$\mathbf{4.02 \pm 1.32}$

Table 5.2: **Function approximation**: Test-set MSE for the Schaal-Atkeson [Schaal & Atkeson, 1998] benchmark. Scale of results is 10^{-3}.

with randomly chosen covariances and deterministically chosen means sufficiently far apart. We have repeated training over 10 independent runs for each dataset to accumulate statistics and reduce variance attributed to random initialisations. The model was purposely over-specified with the number of components set to $K = 10$.

Method/Dataset	pumadyn [Corke, 1996] (10^{-2})			
	8fm	8fh	8nm	8nh
LWPR	**1.36**	8.87	9.71	12.8
oGP	2.34	13.77	5.65	14.32
oGMM	4.59 ± 1.000	9.20 ± 0.76	8.79 ± 3.16	12.33 ± 1.77
oSMM	3.64 ± 0.980	9.37 ± 0.91	7.31 ± 1.50	10.86 ± 1.07
oQMM	2.31 ± 0.280	$\mathbf{8.38 \pm 0.27}$	$\mathbf{5.33 \pm 1.0}$	$\mathbf{9.77 \pm 0.65}$

Table 5.3: **Robot dynamics**: Test-set mean square error for the pumadyn [Corke, 1996] dataset.

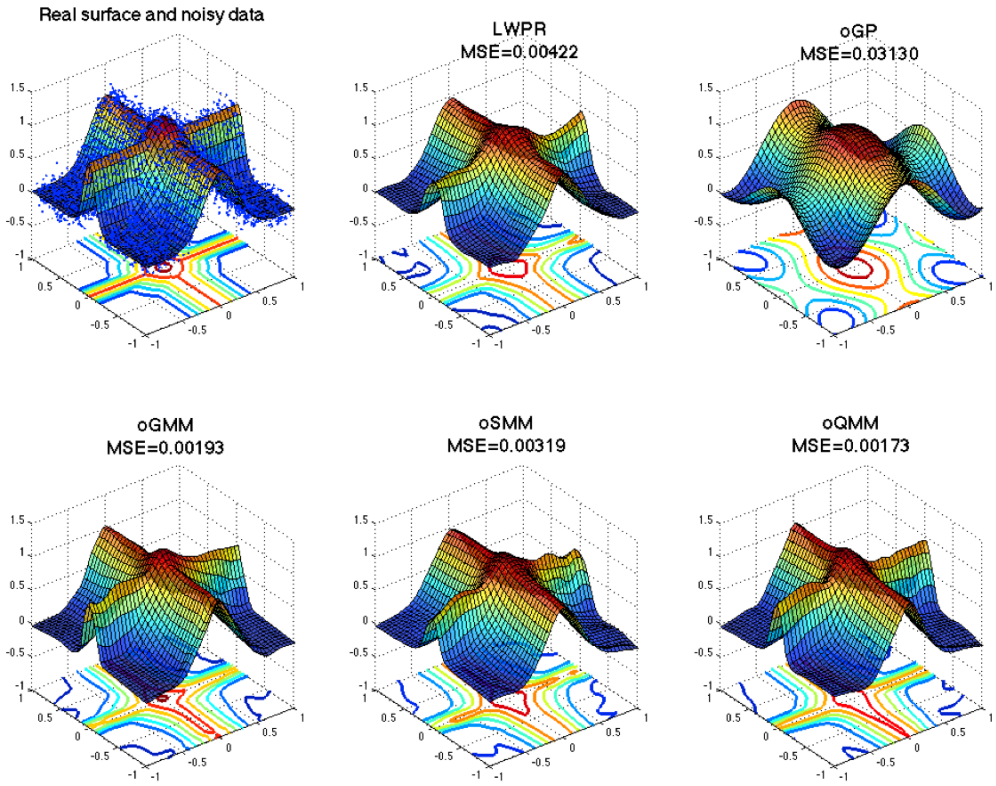


Figure 5.2: **Function approximation**: Best surfaces for all evaluated methods.

Under this setting, we observe that the oGMM tends to be more prone to producing unstable, spurious clusters with a typical case shown in fig. 5.1. Examining the components with regard to the largest eigenvalue of their covariance, we found that the oGMM produced 445 components with maximum eigenvalue below 10^{-4},

while the oQMM produced only 2. This effect becomes more intense in higher dimensional datasets and can lead the oGMM to numerical instability, whereas the oQMM remains less affected.

5.2.3 *Benchmark Data*

5.2.3.1 *Noisy Function Approximation*

The first benchmark is a non-linear function approximation problem under the presence of noise, proposed in [Schaal & Atkeson, 1998] and also adopted in [Sato & Ishii, 2000] and [Vijayakumar et al., 2005].

$$y = \max\left\{e^{-10x_1^2}, e^{-50x_2^2}, e^{-5(x_1^2+x_2^2)}\right\} + \mathcal{N}(0, 0.01) \tag{5.15}$$

Points drawn from the noisy function, the form of the noisy surface and the original surface can be seen in the first 3 subplots of fig. 5.2 respectively.

We have randomly generated 20 training datasets using eq. 5.15, by drawing 5000 points $\boldsymbol{x}$ from a uniform distribution $\mathcal{U}[-1, 1]$. In order to account for different random initialisations, we execute each of the oGMM, oSMM and oQMM 10 times for each dataset with common parametrisation and initialisation. LWPR and the oGP are executed once for every dataset, as they exhibit an almost deterministic behaviour.

The accumulated statistics are presented in Table 5.2. By applying the student-t test we have established that the differences of all presented results are statistically significant at the confidence level of 5% or less. The best predicted surface for each evaluated method is presented in fig. 5.2.

The quantum mixture achieves the best results in this experiment, around 13% better than the conventional mixture, 29% better than the Student-t mixture and over 35% better than LWPR. The online Gaussian process performs poorly in this experiment.

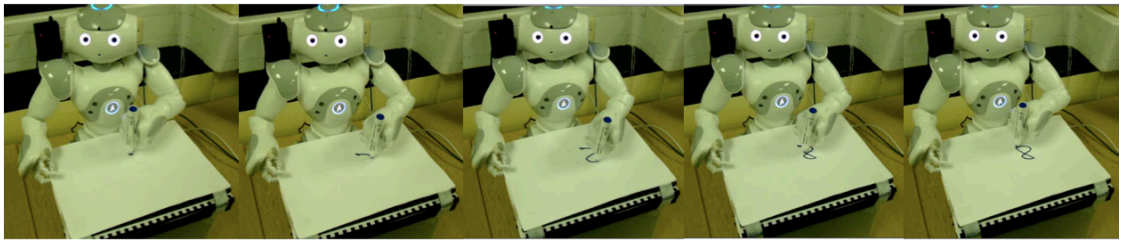


Figure 5.3: Experimental setup of NAO robot drawing *lazy figure 8s*.

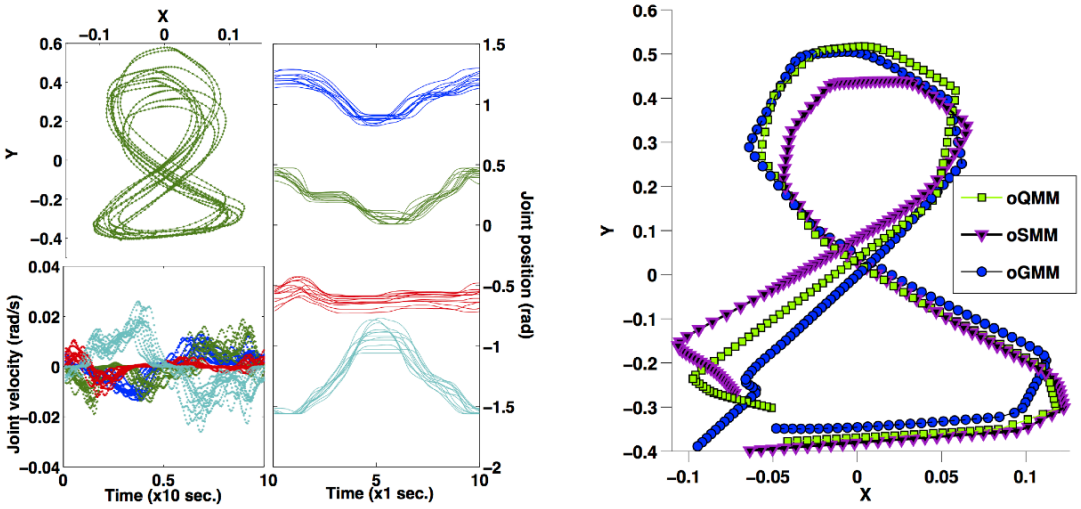
5.2.3.2 *Puma Robot Dynamics*

For our second experiment we consider 4, 9-dimensional datasets from the well-known Puma robot dynamics benchmark (`pumadyn`) [Corke, 1996]. The task is to learn the simulated forward dynamics of a Puma 560 robot arm.

Specifically, the 8 first dimensions are used as inputs, consisting of joint angular positions and velocities for 3 links and torque values for two joints. The last dimension is the target variable and represents the angular acceleration of the third link. The name of each dataset starts with an integer indicating dimensionality, followed by two letters denoting the non-linearity and noise levels respectively (with `f` standing for "fairly linear", `n` "non-linear", `m` "medium noise" and `h` "high noise").

Out of 8192 available data-points, we use the first 7192 for training and the rest for testing. We have executed 50 repetitions of the oGMM, oSMM and oQMM to accumulate statistics and account for random initialisation. LWPR and the oGP were executed once for each dataset, as their performance is almost deterministic.

Our results can be seen in Table 5.3. We can observe that LWPR proves especially effective in fairly linear datasets, regardless of their noise levels, while the oGP is effective in cases of moderate noise, regardless of the level of non-linearity. On the contrary, the oQMM seems to perform well in both aforementioned cases, exhibiting invariant performance regardless the non-linearity or noise level,thus constituting a more generally applicable method. It should be noted that, at the least challenging dataset `8fm`, LWPR performs better than the oQMM. Despite



(a) Training data: **Upper left**: Joint positions with respect to the experiment's drawing plane. **Lower left**: Angular velocities (rad/s) for each joint. **Right**: Angular positions (rad) for each joint. (figure best viewed in colour)

(b) Predicted trajectories obtained by all evaluated methods. X-Y is the drawing plane of our experiment. (figure best viewed in colour)

Figure 5.4: **Multi-shot LbD**: Case-study training dataset of NAO drawing lazy figure 8s.

that, in all remaining datasets the oQMM achieves equivalent or superior results to all rival methods.

5.2.4 *Case Study: Multi-Shot Trajectory LbD*

Our case study is a multi-shot LbD task, namely drawing *lazy figure 8s*, as shown in fig. 5.3. The task might appear trivial, however in the high dimensional joint space it entails severe challenges for learning algorithms and for that reason is regarded a classical benchmark [Vijayakumar et al., 2005].

In our experiments, we make use of the NAO robotic platform (academic edition); a humanoid robot with 27 DoF, a subset of which is employed in this case. We have obtained 12 distinct demonstrations of *lazy figure 8s* presented to the NAO robot

	(t, x)	$(t, x) + cm$	(x, v)
LWPR	4.221		3.558
oGP	5.947		4.774
oGMM	2.902 ± 0.6	2.517 ± 0.6	2.850 ± 1.7
oSMM	2.986 ± 0.7	4.240 ± 0.1	2.214 ± 0.7
oQMM	$\mathbf{2.688 \pm 0.4}$	$\mathbf{2.296 \pm 0.4}$	$\mathbf{2.173 \pm 0.8}$

Table 5.4: **Multi-shot LbD**: Test-set trajectory reconstruction MSE. (10^{-3})

by means of kinesthetics. Extra care has to be devoted so as each demonstration to be performed in consistent speed or alternatively the trajectories could be subjected to online time warping. Each demonstration is consisted of 170, 5-dimensional data-points: 4 joint angle positions $\boldsymbol{x}$ and the time component t. We have used the first 11 demonstrations for training and the last for testing. As can be seen in fig. 5.4a, the dataset is severely ridden with noise. Furthermore, each demonstration has different points of origin and relatively few data-points. All those characteristics combined constitute a formidable challenge for any algorithm.

We consider two different learning scenarios. According to the first, the predictor variable is the time component of the demonstrated task t and joint angle positions $\boldsymbol{x}$ serve as response variables. This setting is frequently employed with Gaussian processes, where time is considered the free variable of the experiment. The second scenario is a more challenging one and consists of predictions regarding the next step velocities $\boldsymbol{v}$, given previous step joint angle positions $\boldsymbol{x}$. This is because the velocities are considerably noisier (fig. 5.4a) and task reproduction requires predictions of higher precision. In this case the error metric we employ is the trajectory reconstruction error, calculated from the predicted velocities.

The results, as shown in Table 5.4, reveal that for the first scenario (t, x), the oQMM performs better than both oSMM and oGMM from around $8\% - 11\%$ and much better than LWPR and oGP. Regarding the component manipulation (cm) scheme, we have found that it performs reasonably well, achieving around 17% higher accuracy with lower computational costs.

In the case of the second scenario (x, v), the best method is still the oQMM, with the oSMM also performing at the same level. In fig. 5.4b we can see an example of the best fit of reconstructed trajectories for the oGMM, oSMM and oQMM. We can see that the shape of the trajectory yielded by the oQMM is considerably better. Although generally achieving low reconstruction errors, the component manipulation scheme was not consistent enough in this scenario.

5.2.5 *Generalisation*

Following a similar setting to Section 4.2.3, we aim at assessing the proposed algorithms under data deprivation and on a wider testing set. However, this is more difficult here for the following reasons:

- The online algorithms evaluated in this section are not designed to train from small dataset, rather the oposite.
- All evaluated algorithms do not iterate through the datasets, but consider each datapoint only once. This fact is expected to severely diminish their effectiveness under the presence of very few training examples.
- In the present setting the dataset is already much smaller than required by online algorithms and further reducing it is expected to cause stability issues.

Nevertheless, we shall attempt to push our algorithms to the limit and try to tentatively draw some conclusions.

Our results are presented in fig. 5.5. We have only reached up to 4 sequences for testing and 8 for training, until we observed severe instability of the evaluated algorithms. From fig. 5.5 we may observe that all methods disintegrate rapidly when less than 10 demonstrations are available for training. More specifically, the oGP is the most sensitive to the reduction of the training set. The rest of the methods perform similarly, with the LWPR being quite stable for small training

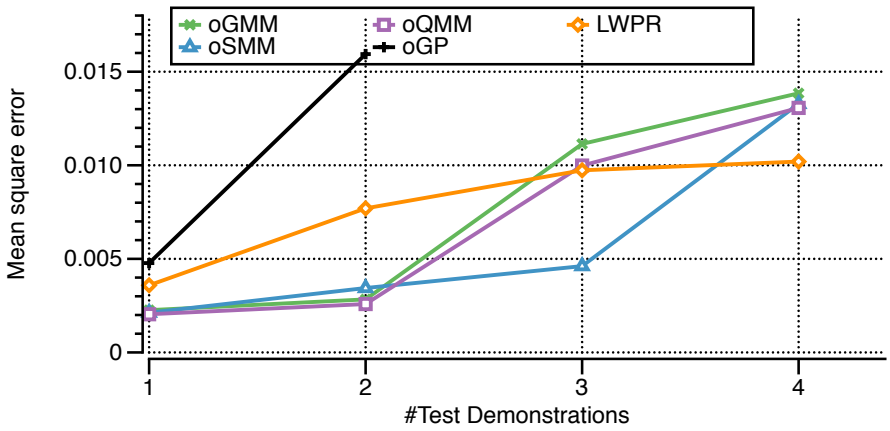


Figure 5.5: **Multi-shot LbD:** In this figure we depict the median test MSE, as we gradually increase the testing set (and decrease the training set).

sets. The oSMM also performs quite well and finally the oGMM and oQMM achieve average performance.

5.2.6 *Note on Computational Cost*

The fastest method is the oGP as it enjoys simple gradient update equations. It is followed by the conventional mixture model, whose cost is dominated by a vector outer product with complexity $\mathcal{O}(D^2)$, where as D we denote the dimensionality of the data. The oQMM is more computationally intensive than the oGMM, due to the fact that it requires one extra matrix exponential. Finally, LWPR and especially the oSMM are the most computationally intensive. In the former this can be due to the fact that a large number of local linear models is needed in order to effectively model data exhibiting high degree of non-linearity. In the case of the oSMM, we cannot analytically solve w.r.t. the Degrees of Freedom (DoF) of the Student-t distribution and thus we resort to costly numerical methods, which introduce high computation burden.

5.2.7 *Accuracy VS computational time*

In this section, following the similar discussion of Section 4.2.5, we shall comment upon the benefits of the oQMM in contrast with the computational costs it induces compared to the conventional mixture model (oGMM). More specifically, we focus on the velocity-position experiment of Section 5.2.4. We have recorded the mean model update and prediction time resulting from 50 runs as follows:

- GMR: 9.66 ± 0.45
- QMR: 12.1 ± 0.66

In order to obtain the aforementioned timings, we have used a MacBook Pro, dual-core Intel Core i7 2.8GHz, 8GB of RAM, running Matlab 2014b under OSX 10.10.

We can observe that the extra computational time of the oQMM is 30.8%, while the oGMM error is 31.1% higher. This means that 1% of error reduction comes at a cost of approximately 1% increase in computational time.

5.3 CONCLUSION

In this chapter, we have presented the online quantum mixture model, a powerful framework for robot learning by demonstration. Our approach is based on quantum mixture regression and recent advances in stochastic optimisation. We also provide a component manipulation scheme, which can result in higher performance and lower computational costs.

Our method is especially suited for large, complex datasets. It exhibits higher numerical stability and is generally applicable regardless the noise or the non-linearity level of the data. We have also shown that it performs very well in a

demanding multi-shot learning by demonstration application, where it enjoys higher accuracy of prediction and trajectory reconstruction.

Part III

LEARNING FROM INCOMPLETE DATA

6

MULTI-KERNEL GPDS

ALTHOUGH sensor equipment is becoming increasingly precise and reliable nowadays, it is sometimes impossible to avoid the manifestation of missing data. Even when using highly expensive motion capture systems or surveillance equipment, the data may suffer from severe corruption and/or may require manual rectification. The problem is intensified in case we only have low-cost equipment in our disposal, such as the popular Kinect sensors. As technological advances facilitate the data collection process, it becomes progressively easier to collect large amounts of data and make them publicly available. Very few of those datasets however are free of corruption and manually rectifying them may be costly or even impossible.

In data completion problems the main approaches in the literature employ either supervised or unsupervised learning. In the case of supervised methods, learning is performed on a set of complete training sequences and rectification of incomplete sequences is performed by means of the posterior predictive distribution of the trained model. This is rather more restrictive scenario, as it may be difficult to obtain a sufficiently large set of complete training sequences. There are many relevant example in the literature employing a plethora of models, such as the Gaussian Process (GP), the Hidden Markov Model (HMM), Linear Dynamical Systems (LDS) etc.

On the contrary, unsupervised learning, which is performed solely on the incomplete data, pertains to a more realistic situation for practical applications. In this case missing data prediction may also be achieved by means of the model predictive distribution, however this frequently intractable. In order to overcome the intractability of the predictive posterior, we may resort to approximations such as Variational Bayesian (VB) [Opper & Archambeau, 2009], sampling methods [Brooks et al., 2011] or Maximum a-Posteriori (MAP) predictions [Zhou et al., 2011]. The latter, although being a rough approximation, is applicable in a wider range of models than VB, it is less computationally intensive than sampling and often provides a good enough estimate. Similar approaches are widely adopted in image processing, such as inpainting[1] [Paisley et al., 2010b].

In the non-sequential unsupervised case, the missing data problem in question has previously been approached from a matrix completion perspective [Ma et al., 2014]. A variety of algorithms have been proposed over the years, the great majority of which focus on point-wise optimisation imposing various types of regularisation constraints for more effective learning [Marjanovic & Solo, 2012; Candès & Tao, 2010] and with many successful applications [Kalogerias & Petropulu, 2014; Parhizkar et al., 2013]. Statistical approaches, on the contrary, enjoy a number of comparative merits stemming mainly from the fact that inference, rather than optimisation, is employed. As a result, such methods are less prone to overfitting and have also been reported to achieved superior performance [Babacan et al., 2012].

Modern statistical methods usually focus on image processing [Zhou et al., 2010, 2012] or collaborative filtering [Yang et al., 2012; Sun et al., 2014] and few have been formulated to explicitly model temporal data. Furthermore, although unsupervised missing data completion is common in image inpainting, this concept has not been extensively employed for sequential data, where learning is usually performed in a supervised manner.

1 Interpolation of the pixels removed from random locations in the image.

A model that is widely adopted for the purpose of data completion in general is the GP [Rasmussen & Williams, 2004]. This is mainly due to the fact that GPs are powerful non-parametric models, allowing us to impose high level data characteristics, such as smoothness, trend or periodicity. This is possible by selecting the appropriate functional form of the kernel matrix. This convenient property of the GP has been successfully used, for example, in [Duvenaud et al., 2013] for the purpose of creating the "automated statistician" by selecting and interpreting the characteristics of the most probable kernel structure for a given set of data. Alternatively to using single GPs, some researchers have explored models consisting of linear superpositions of GPs [Titsias & Lázaro-Gredilla, 2011], which is shown to increase the model's descriptive power, with further relevant examples found in [Wilson et al., 2012] and [Luttinen & Ilin, 2009].

An interesting generalisation to the conventional GP for the unsupervised case (where no observed covariates are available) is the Gaussian Process Latent Variable Model (GPLVM). The model assumes lower-dimensional latent input covariates, giving rise to a powerful non-linear factor model. The GPLVM has been applied with considerable success in data visualisation [Lawrence, 2003], dimensionality reduction [Lawrence, 2005], classification [Urtasun & Darrell, 2007] and time-series modelling [Lawrence & Moore, 2007]. Its efficacy has also been previously shown in data completion applications [Zhou et al., 2010].

The underlying concept of the GPLVM has initiated vigorous research in the field of Gaussian processes, giving rise to a variety of models able to capture data dynamics in motion and financial time-series [Alvarez et al., 2009b; Hartikainen et al., 2012]. Recently a Bayesian formulation of the GPLVM has been devised in [Titsias & Lawrence, 2010] by using the concept of inducing variables in order to ameliorate some of the intractabilities encountered during inference.

Further extending the notion of latent covariates towards modelling temporal data, Damianou et. al. [Damianou et al., 2011] introduced a hierarchical model with an intermediate latent layer and a kernel governed by time, dubbed the Gaussian

Process Dynamical Systems (GPDS) and later developed the deep GP [Damianou & Lawrence, 2013] by introducing multiple such intermediate latent layers.

In this chapter, we propose a multi-task, multi-kernel model consisting of GPDS factors that we shall dub the Multi-GPDS. We form a dictionary of non-linear temporal factors shared among multiple tasks in order to achieve information transfer across tasks with significant merits in terms of reconstruction accuracy. We have observed considerable performance gains compared to popular methods in the literature. More specifically, we compare against the following methods: Interpolation, SVT [Cai et al., 2010], FPC [Ma et al., 2011], VBMC [Babacan et al., 2012], SS-GP [Titsias & Lázaro-Gredilla, 2011], as well as some variations of the aforementioned models. For more information we refer to the experimental section of this chapter.

6.1 THEORETICAL BACKGROUND

6.1.1 *Gaussian Processes*

A Gaussian process is formally defined as a possibly infinite collection of random variables $[f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), ...]$, any finite subset of which follows a joint Gaussian distribution [Rasmussen & Williams, 2004].

More specifically, the training data $\mathcal{D} = (\boldsymbol{y}, \boldsymbol{X})$ consist of an observation vector $\boldsymbol{y} = [y_t]_{t=1}^T$ and a covariate matrix $\boldsymbol{X} = [\boldsymbol{x}_t]_{t=1}^T$, where, in general, $\boldsymbol{x}_t$ lies in a Q-dimensional space, $\boldsymbol{x}_t \in \mathbb{R}^Q$. The observations are modeled as follows:

$$y_t \sim \mathcal{N}\left(f(\boldsymbol{x}_t), \sigma^2\right) \tag{6.1}$$

$$\boldsymbol{f} \sim \mathcal{N}\left(0, \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})\right) \tag{6.2}$$

where $\boldsymbol{f} = [f_t]_{t=1}^T$ and each random variable $f_t = f(\boldsymbol{x}_t)$ represents the value of the function given input $\boldsymbol{x}_t$.

In that perspective, a Gaussian Process (GP) can be viewed as a prior distribution over the set of all possible functions $f(\boldsymbol{x})$, with some characteristics dictated by the functional form of the kernel matrix $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$.

For a set of test points $\boldsymbol{X}_*$, direct inference is tractable for this model as follows:

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I} & \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*) \\ \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) & \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*) \end{bmatrix}\right) \tag{6.3}$$

$$\mathbb{E}\left[\boldsymbol{f}_* | \boldsymbol{\mathcal{D}}, \boldsymbol{X}_*\right] = \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) \left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I}\right]^{-1} \boldsymbol{y} \tag{6.4}$$

$$\begin{aligned} \mathbb{V}\left[\boldsymbol{f}_* | \boldsymbol{\mathcal{D}}, \boldsymbol{X}_*\right] &= \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*) - \\ &\quad - \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) \left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I}\right]^{-1} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*) \end{aligned}$$

The ability to conduct direct inference is very convenient, although it does not scale well with the size of the training set as it is dominated by the inversion of an $T \times T$ matrix with complexity $\mathcal{O}\left(T^3\right)$.

In order to account for multiple dimensions, we could either impose a common or independent GP priors per dimension, with the latter being preferable to the former [Rasmussen & Williams, 2004].

6.1.2 *Gaussian Process Latent Variable Models*

Let us now consider a multi-dimensional set of observed data $\boldsymbol{Y} \in \mathbb{R}^{T \times D}$ and no known corresponding covariates $\boldsymbol{X}$. We may still assume that the observed data can be accurately represented by a latent covariate space $\boldsymbol{X} \in \mathbb{R}^{T \times Q}$ of lower dimensionality $Q \ll D$ through a non-linear relationship.

Given the latent covariate space $\boldsymbol{X}$, the GPs are postulated to be independent across dimensions, giving rise of the following factorisation.

$$p(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{i=1}^{D} p\left(\boldsymbol{y}_i | \boldsymbol{X}\right) \tag{6.5}$$

where as $\boldsymbol{y}_i$ we denote a vector consisting of the i^{th} column of $\boldsymbol{Y}$. Note that we follow a completely analogous notation for the rest of this chapter.

The corresponding probability distribution of $\boldsymbol{y}_i$ is given by:

$$p\left(\boldsymbol{y}_i|\boldsymbol{X}\right) = \mathcal{N}\left(\boldsymbol{y}_i|\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma^2\boldsymbol{I}\right) \tag{6.6}$$

6.2 PROPOSED APPROACH

6.2.1 *Motivation*

Let us now consider the case we have multiple sequences in our disposal, but perhaps none of which are complete. Each sequence $\boldsymbol{Y}_n \in \mathbb{R}^{T\times D}$ consists of multi-dimensional observations and we constrain them to be of fixed length T. The sequences may be of either the same or different tasks, with the assumption that there is at least some redundancy to be harnessed among them.

We envision a model able to take into account the sequential nature of the data and the redundancies that manifest themselves among repetitions of a single task as well as those across different tasks. This could be achieved by supposing that our sequences are a linear superposition of a finite dictionary of not necessarily linear basis functions, shared among all instances. Indeed, this notion is frequently encountered in the literature for the purpose of modeling both image [Zhou et al., 2011, 2012] and motion data [Vollmer et al., 2012].

In the past, researchers have considered imposing either spherical Gaussian priors [Babacan et al., 2012] or full GP priors [Titsias & Lázaro-Gredilla, 2011] in the context of matrix completion and collaborative filtering with great success. Indeed, it has been well documented that certain datasets, such as motion and sound, may lie in much lower dimensional manifolds than those they are presented in [Vijayakumar et al., 2005]. Nevertheless, the assumption that this manifold coincides with the

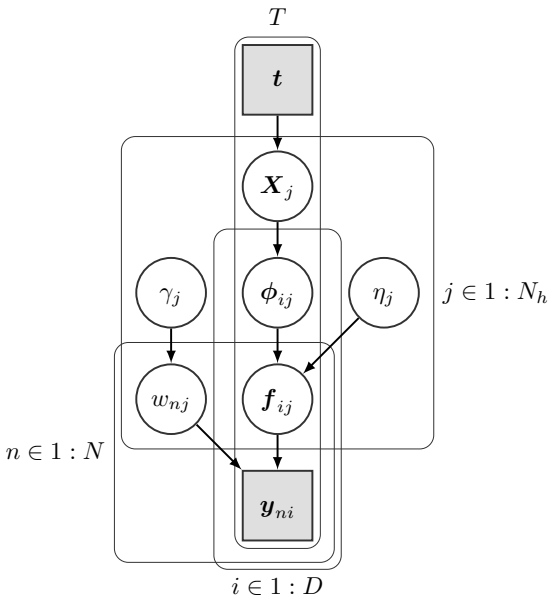


Figure 6.1: Simplified plate diagram of the proposed model (Multi-GPDS), showing roughly the dependence between variables. Note that $\boldsymbol{y}_{ni}, \boldsymbol{f}_{ij}, \boldsymbol{\phi}_{ij} \in \mathbb{R}^{T\times 1}$ and $\boldsymbol{X}_j \in \mathbb{R}^{T\times Q}$, with Q being the dimensionality of the latent covariate space.

time component could be a bit restrictive. In fact, it is quite possible that the optimal manifold is not one dimensional and, even if it is, it might be represented by more complex one-dimensional trajectories.

6.2.2 *Model Formulation*

As we have described in the previous section, the training set $\mathcal{D} = \{\boldsymbol{Y}_n\}_{n=1}^N$ consists of N sequences of D-dimensional data in time, denoted as $\boldsymbol{Y}_n \in \mathbb{R}^{T\times D}$. We postulate that they can be represented by a dictionary of non-linear basis factors denoted as $\boldsymbol{f}_{ij} \in \mathbb{R}^{T\times 1}$ shared among all sequences as follows:

$$\boldsymbol{y}_{ni} \sim \mathcal{N}\left(\boldsymbol{y}_{ni} | \sum_{j=1}^{N_h} w_{nj}\boldsymbol{f}_{ij}, \beta^{-1}\boldsymbol{I}\right) \tag{6.7}$$

We define the aggregate loadings matrix $\boldsymbol{W} \in \mathbb{R}^{N \times N_h}$, with $[\boldsymbol{W}]_{nj} \triangleq w_{nj}$ on which we impose a spherical Gaussian prior with column-wise sparsity as follows:

$$\boldsymbol{w}_j \sim \mathcal{N}(\boldsymbol{w}_j|0, \gamma_j^{-1}\boldsymbol{I}) \tag{6.8}$$

where as $\boldsymbol{w}_j$ we denote the j^{th} column of the loading matrix $\boldsymbol{W}$.

As for the basis functions, we choose a Gaussian Process Dynamical Systems (GPDS) prior distributions, one for each latent basis matrix and with a different latent set of covariates $\boldsymbol{X}_j \triangleq \{\boldsymbol{x}_{tj}\}_{t=1}^{T}$, with $\boldsymbol{x}_{tj} \in \mathbb{R}^Q$. This choice was made so as to achieve a rich non-linear structure.

$$\boldsymbol{f}_{ij}|\boldsymbol{\phi}_{ij} \sim \mathcal{N}\left(\boldsymbol{f}_{ij}|\boldsymbol{\phi}_{ij}, \eta_j^{-1}\boldsymbol{I}\right) \tag{6.9}$$

$$\boldsymbol{\phi}_{ij}|\boldsymbol{X}_j \sim \mathcal{N}\left(\boldsymbol{\phi}_{ij}|0, \boldsymbol{K}_{xx}^{(j)}(\boldsymbol{X}_j, \boldsymbol{X}_j)\right) \tag{6.10}$$

$$\boldsymbol{x}_{qj}|\boldsymbol{t} \sim \mathcal{N}\left(\boldsymbol{x}_{qj}|0, \boldsymbol{K}_{tt}^{(j)}(\boldsymbol{t}, \boldsymbol{t})\right) \tag{6.11}$$

Finally, we impose Gamma priors on the precision parameters $\{\gamma_j, \eta_j\}_{j=1}^{N_h}$:

$$\gamma_j \sim \mathcal{G}\left(a_\gamma, \frac{1}{b_\gamma}\right) \tag{6.12}$$

$$\eta_j \sim \mathcal{G}\left(a_\eta, \frac{1}{b_\eta}\right) \tag{6.13}$$

and a standard Jeffrey's prior for the precision of the likelihood β:

$$p(\beta) = \beta^{-1} \tag{6.14}$$

A simplified plate diagram of the proposed model is presented in fig. 6.1.

6.2.3 *Variational Posterior Inference*

6.2.3.1 *Inducing Variables*

Variational inference is, of course, intractable for this model due to the non-linear relationship between $\boldsymbol{X}_j$ and $\boldsymbol{\phi}_{ij}$ in eq. 6.10. However, tractable inference can be

achieved by introducing the concept of "inducing variables" [Titsias & Lawrence, 2010], as shown in [Damianou et al., 2011] for the conventional GPDS. This concept is directly applicable to our case as well. More specifically, we can augment our original problem by introducing $M \ll T$ extra auxiliary sample points drawn from the same prior distribution as the original sample $\boldsymbol{\phi}_{ij}$, but located on a different set of latent locations $\boldsymbol{Z}_j$:

$$\boldsymbol{u}_{ij} \sim \mathcal{N}(0, \boldsymbol{K}_{xx}^{(j)}(\boldsymbol{Z}_j, \boldsymbol{Z}_j)) \tag{6.15}$$

For notational simplicity and in order to explicitly indicate the dimensionality of the corresponding kernel covariance matrices, we shall make the following notational replacements:

- $\boldsymbol{K}_{xx}^{(j)}(\boldsymbol{X}_j, \boldsymbol{X}_j) \rightarrow \boldsymbol{K}_{TT}$
- $\boldsymbol{K}_{xx}^{(j)}(\boldsymbol{X}_j, \boldsymbol{Z}_j) \rightarrow \boldsymbol{K}_{TM}$
- $\boldsymbol{K}_{xx}^{(j)}(\boldsymbol{Z}_j, \boldsymbol{Z}_j) \rightarrow \boldsymbol{K}_{MM}$.

Note that we have also dropped the factor indices from the kernel matrices $\{\boldsymbol{K}_{TT}, \boldsymbol{K}_{TM}, \boldsymbol{K}_{MM}\}$ and will henceforth assume they refer to factor j although not explicitly indicated.

Given the inducing variables $\boldsymbol{u}_{dj}$, our initial sample can be directly retrieved as follows:

$$p(\boldsymbol{\phi}_{ij}|\boldsymbol{u}_{ij}, \boldsymbol{X}_j, \boldsymbol{Z}_j) = \mathcal{N}(\boldsymbol{\phi}_{ij}|\boldsymbol{B}_j\boldsymbol{u}_j, \boldsymbol{S}_j) \tag{6.16}$$

$$\boldsymbol{B}_j = \boldsymbol{K}_{TM}\boldsymbol{K}_{MM}^{-1} \tag{6.17}$$

$$\boldsymbol{S}_j = \boldsymbol{K}_{TT} - \boldsymbol{K}_{TM}\boldsymbol{K}_{MM}^{-1}\boldsymbol{K}_{MT} \tag{6.18}$$

by doing so, we do not need to impose an explicit posterior on the original GP sample $\boldsymbol{\phi}_{ij}$, but rather we can simply update it through the auxiliary sample $\boldsymbol{u}_{ij}$. Type-II maximum likelihood is now only necessary for the much smaller set of still intractable latent locations $\boldsymbol{Z}_j$, while inference becomes tractable for $\boldsymbol{X}_j$.

6.2.3.2 *Variational Factorisation*

Given the aforementioned augmented set of variables, the variational posterior for our model is factorised as follows:

$$
\begin{aligned}
q(\boldsymbol{W}, \boldsymbol{H}, \boldsymbol{F}, \boldsymbol{\Phi}, \boldsymbol{U}) = & \prod_{n=1}^{N} q(\boldsymbol{w}_n) \prod_{t=1}^{T} \prod_{i=1}^{D} q(\boldsymbol{f}_{ti}) \\
& \cdot \prod_{i=1}^{D} \prod_{j=1}^{N_h} p(\boldsymbol{\phi}_{ij} | \boldsymbol{u}_{ij}, \boldsymbol{X}_j) q(\boldsymbol{u}_{ij}) \\
& \cdot \prod_{j=1}^{N_h} \prod_{q=1}^{Q} q\left(\boldsymbol{x}_{qj}\right) q\left(\gamma_j\right) q\left(\eta_j\right) q\left(\beta\right)
\end{aligned}
$$

It should be noted that, similar to [Babacan et al., 2012], we have chosen a row-wise factorisation for the loadings $\boldsymbol{W}$ and factors $\boldsymbol{F}_i$, different from the column-wise factorisation used in the prior. This would not be the natural choice since we normally aim to capture dependancies among data-points in time (rows), rather than among factors (columns) [Luttinen & Ilin, 2009]. However, variational inference for the column-wise factorisation yields diagonal posterior covariance matrices and fails to represent the intended relationships, while a row-wise factorisation allows for a richer dependency structure.

6.2.3.3 *Variational Posterior Distributions*

FACTORS AND LOADINGS POSTERIORS Optimising the lower bound yields analytical updates for most of the random variables in our model. We present our results in this section.

The posterior of the factor loadings $\boldsymbol{w}_n$ is formulated as follows:

$$q(\boldsymbol{w}_n) = \mathcal{N}\left(\boldsymbol{w}_n | \boldsymbol{\mu}_{w_n}, \boldsymbol{\Sigma}_{w_n}\right) \tag{6.19}$$

$$\boldsymbol{\Sigma}_{w_n} = \left[\boldsymbol{\Gamma} + \boldsymbol{V}_n\right]^{-1} \tag{6.20}$$

$$\boldsymbol{\mu}_{w_n} = \boldsymbol{\Sigma}_{w_n} \boldsymbol{c}_n \tag{6.21}$$

$$\boldsymbol{V}_n = \langle \beta \rangle \sum_{(t,i) \in \mathcal{O}_n} \left[\langle \boldsymbol{f}_{ti} \rangle \langle \boldsymbol{f}_{ti} \rangle^T + \boldsymbol{\Sigma}_{f_{ti}} \right] \tag{6.22}$$

$$\boldsymbol{c}_n = \langle \beta \rangle \sum_{(t,i) \in \mathcal{O}_n} y_{nit} \langle \boldsymbol{f}_{ti} \rangle \tag{6.23}$$

with $\boldsymbol{\Gamma} = diag\left[\langle \boldsymbol{\gamma} \rangle\right]$.

As $\mathcal{O}_n$ we denote the set consisting of the dyads of indices (t, i) which are observed in sequence $\boldsymbol{Y}_n$.

Note that $\boldsymbol{V}_n$ is a full $N_h \times N_h$, rather than diagonal, matrix and consequently so is the posterior covariance $\boldsymbol{\Sigma}_{w_n}$.

The posterior of the intermediate factor variables $\boldsymbol{f}_{ti}$ can be formulated as follows:

$$q(\boldsymbol{f}_{ti}) = \mathcal{N}\left(\boldsymbol{f}_{ti} | \boldsymbol{\mu}_{f_{ti}}, \boldsymbol{\Sigma}_{f_{ti}}\right) \tag{6.24}$$

$$\boldsymbol{\Sigma}_{f_{ti}} = \left[\boldsymbol{H} + \boldsymbol{L}_n\right]^{-1} \tag{6.25}$$

$$\boldsymbol{\mu}_{f_{ti}} = \boldsymbol{\Sigma}_{f_{ti}} \left[\boldsymbol{H} \langle \boldsymbol{\phi}_{ti} \rangle + \boldsymbol{z}_n\right] \tag{6.26}$$

$$\boldsymbol{L}_{ti} = \langle \beta \rangle \sum_{n \in \mathcal{O}_{ti}} \left[\langle \boldsymbol{w}_n \rangle \langle \boldsymbol{w}_n \rangle^T + \boldsymbol{\Sigma}_{w_n} \right] \tag{6.27}$$

$$\boldsymbol{z}_{ti} = \langle \beta \rangle \sum_{n \in \mathcal{O}_{ti}} y_{nit} \langle \boldsymbol{w}_n \rangle \tag{6.28}$$

with $\boldsymbol{H} = diag\left[\langle \boldsymbol{\eta} \rangle\right]$.

Equivalently, as $\mathcal{O}_{td}$ we denote the set consisting of the indices of sequences $\boldsymbol{Y}_n$ for which the time-step t of dimension d is observed.

GAUSSIAN PROCESS DYNAMICAL SYSTEM POSTERIORS The posteriors of the GPDS variables are slightly more complicated to formulate. An efficient solution for the conventional GPDS is presented in [Damianou et al., 2011], where the authors are able to re-parametrise the problem according to [Titsias & Lawrence,

2010]. We follow similar principles in devising our own solution, which is outlined later on, in the section describing hyper-parameter optimisation.

More specifically, we only need to formulate the posterior of the inducing variables $\boldsymbol{u}_{dj}$, through which we are able to retrieve a posterior estimate for the $\boldsymbol{\phi}_{ij}$'s. Following that, we may then completely disregard the $\boldsymbol{u}_{ij}$'s.

Let us denote the following expectations of the various kernel matrices w.r.t. the latent locations:

$$\begin{aligned}\psi_0 &= tr\left\{\langle \boldsymbol{K}_{TT}\rangle_{q(\boldsymbol{X}_j)}\right\}\\ \boldsymbol{\Psi}_1 &= \langle \boldsymbol{K}_{TM}\rangle_{q(\boldsymbol{X}_j)}\\ \boldsymbol{\Psi}_2 &= \langle \boldsymbol{K}_{MT}\boldsymbol{K}_{TM}\rangle_{q(\boldsymbol{X}_j)}\end{aligned}$$

As shown in [Titsias & Lawrence, 2010], these expectations are tractable for certain kernel functions. For notational simplicity we have dropped the factor index from quantities $\{\psi_0, \boldsymbol{\Psi}_1, \boldsymbol{\Psi}_2\}$ and assume they all refer to factor j although not explicitly indicated.

Optimising the lower bound yields the following solution for the posterior of the inducing variables:

$$q\left(\boldsymbol{u}_{ij}\right) = \mathcal{N}\left(\boldsymbol{u}_{ij}|\boldsymbol{\mu}_{u_{ij}}, \boldsymbol{\Sigma}_{u_{ij}}\right) \tag{6.29}$$

$$\boldsymbol{\Sigma}_{u_{ij}} = \boldsymbol{K}_{MM}^T \boldsymbol{A}_j \boldsymbol{K}_{MM} \tag{6.30}$$

$$\boldsymbol{\mu}_{u_{ij}} = \boldsymbol{K}_{MM}^T \boldsymbol{A}_j \boldsymbol{\Psi}_1^T \boldsymbol{\mu}_{f_{ij}} \tag{6.31}$$

Based on the posterior of $\boldsymbol{u}_{ij}$, we obtain the mean of $\boldsymbol{\phi}_{ij}$, as follows:

$$\boldsymbol{\mu}_{\phi_{ij}} = \langle \eta_j \rangle \boldsymbol{\Psi}_1 \boldsymbol{A}_j \boldsymbol{\Psi}_1^T \boldsymbol{\mu}_{f_{ij}} \tag{6.32}$$

with $\boldsymbol{A}_j = \left(\boldsymbol{K}_{MM} + \langle \eta_j \rangle \boldsymbol{\Psi}_2\right)^{-1}$.

Another quantity necessary for the rest of the inference is the full posterior expectation of $\left\langle \boldsymbol{\phi}_{ij}^T \boldsymbol{\phi}_{ij} \right\rangle$, which is as follows:

$$\begin{aligned}\left\langle \boldsymbol{\phi}_{ij}^T \boldsymbol{\phi}_{ij} \right\rangle = \;& \psi_0 - tr\left\{\boldsymbol{K}_{MM}^{-1}\boldsymbol{\Psi}_2\right\} + tr\left\{\boldsymbol{\Psi}_2 \boldsymbol{A}_j\right\}\\ &+ \langle \eta_j \rangle^2 \boldsymbol{\mu}_{f_{ij}}^T \boldsymbol{\Psi}_1 A_j^T \boldsymbol{\Psi}_2 A_j \boldsymbol{\Psi}_1^T \boldsymbol{\mu}_{f_{ij}}\end{aligned}$$

For the above result we have used eq. 6.16 as well as the following:

$$\langle \boldsymbol{B}_j \rangle = \boldsymbol{\Psi}_1 \boldsymbol{K}_{MM}^{-1} \tag{6.33}$$

$$\left\langle \boldsymbol{B}_j^T \boldsymbol{B}_j \right\rangle = \boldsymbol{K}_{MM}^{-T} \boldsymbol{\Psi}_2 \boldsymbol{K}_{MM}^{-1} \tag{6.34}$$

$$tr\left\{ \langle \boldsymbol{S}_j \rangle \right\} = \psi_0 - tr\left\{ \boldsymbol{K}_{MM}^{-1} \boldsymbol{\Psi}_2 \right\} \tag{6.35}$$

PRECISION VARIABLES POSTERIORS The posterior expectations of the precision variables β, $\{\gamma_j\}_{j=1}^{N_h}$ and $\{\eta_j\}_{j=1}^{N_h}$ are given as follows:

$$\langle \beta \rangle = \frac{\#\mathcal{O}}{\sum_{(n,t,i)\in\mathcal{O}} \left\langle \left(y_{nit} - \boldsymbol{w}_n^T \boldsymbol{f}_{ti} \right)^2 \right\rangle} \tag{6.36}$$

where $\mathcal{O}$ is the set of triplets (n, t, i) of observed data and $\#\mathcal{O}$ is the cardinality of this set.

$$\langle \gamma_j \rangle = \frac{2a + N}{2b + \left\langle \boldsymbol{w}_j^T \boldsymbol{w}_j \right\rangle} \tag{6.37}$$

and

$$\langle \eta_j \rangle = \frac{2a + TD}{2b + \sum_{i=1}^{D} \left\langle \left(\boldsymbol{f}_{ij} - \boldsymbol{\phi}_{ij} \right)^T \left(\boldsymbol{f}_{ij} - \boldsymbol{\phi}_{ij} \right) \right\rangle} \tag{6.38}$$

The posterior expectations required for the above updates are given eq. (33)-(35).

$$\left\langle \left(y_{nit} - \boldsymbol{w}_n^T \boldsymbol{f}_{ti} \right)^2 \right\rangle = \left(y_{nit} - \langle \boldsymbol{w}_n \rangle^T \langle \boldsymbol{f}_{ti} \rangle \right)^2 + \tag{6.39}$$

$$tr\left\{ \boldsymbol{\mu}_{w_n} \boldsymbol{\mu}_{w_n}^T \boldsymbol{\Sigma}_{f_{ti}} + \boldsymbol{\Sigma}_{w_n} \boldsymbol{\mu}_{f_{ti}} \boldsymbol{\mu}_{f_{ti}}^T + \boldsymbol{\Sigma}_{w_n} \boldsymbol{\Sigma}_{f_{ti}} \right\} \tag{6.40}$$

$$\left\langle \boldsymbol{w}_j^T \boldsymbol{w}_j \right\rangle = \sum_{n=1}^{N} \left(\left[\boldsymbol{\mu}_{w_n} \right]_j^2 + \left[\boldsymbol{\Sigma}_{w_n} \right]_{jj} \right) \tag{6.41}$$

$$\left\langle \boldsymbol{f}_{ij}^T \boldsymbol{f}_{ij} \right\rangle = \sum_{t=1}^{T} \left(\left[\boldsymbol{\mu}_{f_{ti}} \right]_j^2 + \left[\boldsymbol{\Sigma}_{f_{ti}} \right]_{jj} \right) \tag{6.42}$$

6.2.3.4 *Hyper-parameter Optimisation*

The hyper-parameters that are optimised rather than inference are the ones pertaining to the individual GPDS factors. In order to estimate them, we follow closely the approach introduced in [Damianou et al., 2011].

The optimisation itself involves Type-II maximum likelihood with respect to:

1. The kernel hyper-parameters of latent kernel matrices $\boldsymbol{K}_{TT}$, $\boldsymbol{K}_{TM}$ and $\boldsymbol{K}_{MM}$.
2. The time kernel matrix $\boldsymbol{K}_{tt}$.
3. The inducing locations $\boldsymbol{Z}_j$.
4. The parameters of the latent locations posterior distribution $q(\boldsymbol{X}_j) = \mathcal{N}(\boldsymbol{X}_j|\cdot)$.

As the posteriors $q(\boldsymbol{X}_j)$ are also Gaussian, we typically need to estimate means and covariances, which involves $\mathcal{O}\left(T^2\right)$ number of parameters per factor. Due to the high number of parameters this is both computationally intensive and slow to converge. For that reason, the authors in [Damianou et al., 2011] have re-parametrised the problem using a Gaussian approximation [Opper & Archambeau, 2009], involving $\mathcal{O}(T)$ number of parameters and as a result the approach scales much better with the number of training data.

In brief, the parameters to be estimated by gradient ascent methods are $\boldsymbol{\Theta}_j = \left\{\boldsymbol{\theta}_j^{(x)}, \boldsymbol{\theta}_j^{(t)}, \boldsymbol{Z}_j, \boldsymbol{\lambda}_j, \boldsymbol{\mu}_{X_j}\right\}$. The likelihood terms dependent on those parameters can be computed by integrating out all other random variables analytically.

To be more specific we begin by considering only the terms containing $\boldsymbol{\Theta}_j$. Then we compute the expectation w.r.t. $\boldsymbol{f}_{ij}$, after which we can integrate out the rest of the variables yielding the likelihood to be optimised in eq. 6.44. Finally, the Gaussian integral I_{ij} is given in eq. 6.45.

$$\mathcal{L}(\boldsymbol{\Theta}_j) \propto \tag{6.43}$$

$$\begin{aligned}
&\propto \sum_{i=1}^{N_y} \int q(\boldsymbol{\phi}_{ij}, \boldsymbol{u}_{ij}, \boldsymbol{X}_j) \ln \frac{\mathcal{N}\left(\boldsymbol{f}_{ij}|\boldsymbol{\phi}_{ij}, \eta_j^{-1}\boldsymbol{I}\right) p(\boldsymbol{f}_{ij}, \boldsymbol{u}_{ij}, \boldsymbol{X}_j|\boldsymbol{t})}{q(\boldsymbol{\phi}_{ij}, \boldsymbol{u}_{ij}, X_j)} d\boldsymbol{f} d\boldsymbol{\phi} d\boldsymbol{u} d\boldsymbol{X} \\
&= \sum_{i=1}^{N_y} \ln I_{ij} - \frac{1}{2}\langle\eta_j\rangle \sum_{\forall(t,i)} \left[\boldsymbol{\Sigma}_{f_{ti}}\right]_{jj} - \frac{1}{2}\langle\eta_j\rangle \psi_0 + \frac{1}{2}\langle\eta_j\rangle tr\left\{\boldsymbol{K}_{MM}^{-1}\boldsymbol{\Psi}_2\right\} \\
&- KL(q(\boldsymbol{X}_j) \parallel p(\boldsymbol{X}_j))
\end{aligned} \tag{6.44}$$

	#Sequences	#Actions	#Repetitions	#Timepoints	#Dimensions
Drawing 8s	24	1	24	121	4
Dance Primitives	449	6	{57,83,84}	73	69
HDM05	168	8	{16,20,52}	70	93

Table 6.1: Dataset details.

$$I_{ij} = \int e^{\left\langle \ln \mathcal{N}\left(\boldsymbol{\mu}_{f_{ij}} | \boldsymbol{B}_j \boldsymbol{u}_{ij}, \langle \eta_j \rangle^{-1} \boldsymbol{I}\right)\right\rangle} \mathcal{N}(\boldsymbol{u}_{ij} | 0, \boldsymbol{K}_{MM}) d\boldsymbol{u}_{ij} = \\ = \frac{\langle \eta_j \rangle^{\frac{N}{2}} |\boldsymbol{K}_{MM}|^{\frac{1}{2}}}{|\boldsymbol{K}_{MM} + \langle \eta_j \rangle \boldsymbol{\Psi}_2|^{\frac{1}{2}}} e^{-\frac{1}{2}\boldsymbol{\mu}_{f_{ij}}^T \left[\langle \eta_j \rangle \boldsymbol{I} - \langle \eta_j \rangle^2 \boldsymbol{\Psi}_1 \left(\boldsymbol{K}_{MM} + \langle \eta_j \rangle \boldsymbol{\Psi}_2\right)^{-1} \boldsymbol{\Psi}_1^T\right] \boldsymbol{\mu}_{f_{ij}}} \quad (6.45)$$

6.3 EMPIRICAL EVALUATION

6.3.1 *Experimental Setting*

6.3.1.1 *Comparisons*

For the purpose of empirically evaluating our approach we have compared against popular models in the literature, some used as originally proposed and others adapted to unsupervised sequence completion.

To be more specific, we aim at quantitatively assessing the efficacy of two main aspects of our model:

- The imposition of GPDS priors for dictionary components
- The coupled prior on factors and loadings (section 6.2.3.2)

For that purpose we compare against the following methods:

- **Interpolation**: The simplest way to conduct missing data reconstruction. This method is aimed to serve as a baseline for all evaluated methods and

highlight the necessity of more sophisticated approaches under the current setting. The particular interpolation method we have used involves solving an approximation to one of several partial differential equations in order to interpolate or extrapolate the missing segments.

- **SVT** [Cai et al., 2010] and **FPC** [Ma et al., 2011]: The methods are dubbed singular Singular Value Thresholding (SVT) and Fixed Point Completion (FPC). They are both non-Bayesian and non-sequential models employing point-wise optimisation for the purpose of matrix completion.

- **SS-GP** [Titsias & Lázaro-Gredilla, 2011]: This model relies on a likelihood of the same functional form as the Multi-GPDS (eq. 6.7), with the difference of a spike-and-slab prior on the loading weights and simple GPs in time on the factors. Note that this method is the first one to employ a coupled variational posterior of the type $q(\hat{w}_{nj}, h_{nj}) = q(h_{nj})q(\hat{w}_{nj}|h_{nj})$, which according to the authors in [Titsias & Lázaro-Gredilla, 2011] and our own experience introduces performance benefits. Note also that this model employs a fully factorized variational posterior of the type $\prod_n \prod_j q(w_{nj})$ rather than a partially factorised one $\prod_n q(\boldsymbol{w}_n)$ as in the case of the VBMC, Multi-GP and Multi-GPDS. For that reason this method is expected to capture less dependancies, which may in some cases translate to lower performance.

$$
\begin{aligned}
w_{nj} &= \hat{w}_{nj} h_{nj} \\
\hat{w}_{nj} &= \mathcal{N}(w_{nj}|0, \gamma_j^{-1}) \\
h_{nj} &= Bernoulli(\pi_j) \\
\boldsymbol{f}_{ij} &\sim \mathcal{GP}(\boldsymbol{f}_{ij}|0, \boldsymbol{K}(\boldsymbol{t}, \boldsymbol{t}))
\end{aligned}
$$

- **IBP-GP**: A model otherwise identical to the SS-GP with the difference that it assumes an Indian Buffet Process (IBP) prior [Griffiths & Ghahramani, 2006] on the factor loadings to induce sparsity. We have not found an equivalent method in the bibliography (using GP priors on the factors), however the extension is straightforward. Note that we adopt the same principle to the SS-GP during inference by adopting a coupled variational posterior, which

resulted in some performance gains. Due to the non-parametric IBP prior we expect this method to perform better compared to its parametric counterpart (SS-GP).

$$
\begin{aligned}
&w_{nj} = h_{nj}\hat{w}_{nj} \\
&h_{nj}|\boldsymbol{u} \sim Bernoulli(\pi_j(\boldsymbol{u})) \\
&\pi_j(\boldsymbol{u}) = \prod_{k=1}^{j} u_k \\
&u_k \sim Beta(\gamma, 1) \\
&\hat{w}_{nj} \sim \mathcal{N}(\hat{w}_{nj}|0, \gamma_j^{-1})
\end{aligned}
$$

- **VBMC** [Babacan et al., 2012]: Variational Bayesian Matrix Completion (VBMC) is a model assuming spherical Gaussian priors for the purpose of general matrix completion. In order to use this method in our setting we have concatenated along the time and joints dimensions. This model is able to capture richer dependency structures by coupling the factors and loadings in the posterior as discusses in Section III. This is achieved by factorising the prior row-wise and the posterior column-wise, leading to full, instead of diagonal, posterior covariance matrices. Note that although this is not an explicitly sequential model, it implicitly models all dependancies by means of a joint posterior distribution over time and joints. We expect it to work well, but not better than the Multi-GPDS, due to the fact that spherical Gaussian priors are generally less expressive that GPDS priors.

- **Multi-GP**: A model otherwise the same as the Multi-GPDS with the difference that it assumes GP, instead of GPDS priors, on the factors. We have compared against this model in order to illustrate the benefits of introducing a more complex GPDS prior on the factors.

- **AR1**: A model otherwise the same as the Multi-GPDS with the difference that it assumes an auto-regressive prior AR_1 on the factors. We also impose a joint Normal-Gamma prior on the mean and precision of the autoregressive

prior. This model has been chosen for comparison in order to highlight the necessity of non-linear GPDS priors on the factor components.

$$
\begin{aligned}
f_{1ij} &\sim \mathcal{N}\left(f_{1ij}|\mu_{ij}, \lambda_{ij}^{-1}\right) \\
f_{tij} &\sim \mathcal{N}\left(f_{tij}|f_{(t-1)ij}, \lambda_{ij}^{-1}\right) \\
\mu_{ij}, \lambda_{ij} &\sim \mathcal{N}\left(\mu_{ij}|\mu_0, (\beta_0\lambda_{ij})^{-1}\right)\mathcal{G}\left(\lambda_{ij}|a_0, \frac{1}{b_0}\right)
\end{aligned}
$$

6.3.1.2 *Data Removal Scheme*

Throughout this chapter we have utilised complete data sequences from which we remove consecutive segments in a manner that is designed to be as realistic and challenging as possible. Our sole purpose in using complete sequences is to have access to the corresponding ground truth necessary for quantitatively assessing the reconstruction accuracy.

In the data removal scheme we aim at removing a variable random number of segments, of variable random length. More specifically, throughout our experimental evaluation, we perform data removal according to the scheme presented in Algorithm 1.

DESCRIPTION OF ALGORITHM 1

- `[line 2]` Calculate the total number of data-points N_r to be remove from sequence n according to the desired missing data ratio r.
- `[line 3]` Draw the number of missing segments q_n according to a truncated (strictly positive) Poisson distribution with intensity λ.
- `[line 4]` Draw a vector $\boldsymbol{\ell}$ of length q_n of uniformly distributed numbers.
- `[line 5]` Normalize $\boldsymbol{\ell}$ so that it sums to one: $\sum_{k=1}^{q_n} \ell_k = 1$. ℓ_k now represents the proportion of N_r that will be removed.

Algorithm 1 Random data removal algorithm.

Inputs:

- N: Number of sequences.
- $r \in \mathbb{R}$: Desired missing data ratio.
- $\lambda \in \mathbb{N}$: Intensity of Poisson distribution for the number of missing data segments.
- $\boldsymbol{T} \in \mathbb{N}^N$: Vector of length N, where each element T_n containing the number of datapoints in the n^{th} data sequence.

Outputs:

- $\boldsymbol{q} \in \mathbb{N}^N$: Vector, each element q_n of which, contains the number of missing segments in each sequence.
- $\{\boldsymbol{m}_n \in \mathbb{N}^{q_n}\}_{n=1}^N$: Vectors containing the lengths of missing data segments.
- $\{\boldsymbol{s}_n \in \mathbb{N}^{q_n}\}_{n=1}^N$: Vectors containing the starting points of the missing data segments.

Begin:

for n=1...N **do**
  $N_r = rT_n$
  $p(q_n|\lambda) \propto \delta(q_n > 0)Poisson(\lambda)$
  $\boldsymbol{\ell} \sim \mathcal{U}([0, 1])$
  $\boldsymbol{\ell} = normalise(\boldsymbol{\ell})$
  for k=1...q_n **do**
    $m_{nk} = \lfloor N_r \ell_k \rfloor$
    $b_k = \lceil T_n \ell_k \rceil$
    $s_k \sim \mathcal{U}[0, b_k - m_{nk}]$
  end for
end for

End

Return: $\boldsymbol{q}$,$\{\boldsymbol{m}_n\}_{n=1}^N$,$\{\boldsymbol{s}_n\}_{n=1}^N$

- `[line 7]` Calculate the length of the missing data segment m_{nk}. Note that as $\lfloor \cdot \rfloor$ we denote the 'floor' operator.
- `[line 8]` Estimate the length b_k of a bounding box for the missing segment. Note that as $\lceil \cdot \rceil$ we denote the 'ceiling' operator and that both b_k and m_{nk} represent the same proportion of T_n and N_r respectively: $\frac{b_k}{T_n} = \frac{m_{nk}}{N_r} = \ell_k$.
- `[line 9]` Draw a starting position s_k, uniformly at random, within the bounding box of length b_k to place the k^{th} missing segment within it. In order to guarantee the non-overlapping placement of all segments we enforce the following constraint on the starting points: $0 \leq s_k < b_k - m_{nk}$.

6.3.1.3 *Data Processing*

In order to ensure a realistic experimental scenario we conduct processing of the sequences only after the data removal.

All evaluated algorithms require sequences of fixed length, hence we fix the sequences to their average length by subsampling the longer or interpolating the shorter ones. During the subsampling we simply remove data on regular intervals in order to shorten the sequence to the required length. In the case of short sequences, we infuse values in regular intervals by means of interpolation so as to achieve the desired length. In case the interpolation is not possible due to missing data, we simply flag the new datapoint as missing. Note that this procedure achieves an effect similar to demonstration speed normalisation, but does not align the sequences in time.

Finally, prior to presenting the data to each algorithm, we have normalised the incomplete sequences by removing their mean per dimension and scaling according to their total standard deviation. The normalisation is performed based only on the observed data-points.

	Param.	SVT	FPC	VBMC	AR1	SS-GP	Multi-GP	IBP-GP	Multi-GPDS
Robotic	λ	2							
	N_h	32							16
	N_{GD}	1000		NA		10			
	N_{freq}	NA				10			
	$\boldsymbol{K}_{tt}^{(j)}$	NA				compound(RBF,White,Bias)			
	$\boldsymbol{K}_{xx}^{(j)}$	NA							SE
	Q	NA							2
MoCap	λ	2							
	N_h	64							
	N_{GD}	1000		NA		10			
	N_{freq}	NA				5			
	$\boldsymbol{K}_{tt}^{(j)}$	NA				compound(RBF,White,Bias)			
	$\boldsymbol{K}_{xx}^{(j)}$	NA							SE
	Q	NA							1

Table 6.2: Main hyper-parameters needed to reproduce all experiments: λ: Poisson intensity, N_h: Number of latent factors or initial rank, Q: Dimensionality of the GPDS latent covariate space, N_{GD}: Maximum number of gradient descent iterations, N_{freq}: Kernel update frequency. As 'NA' we have denoted parameters that are 'Not Applicable' for a certain algorithm.

6.3.1.4 *Implementation and Setting*

In our implementation we attempt to ensure consistency, impartiality and reproducibility by following a number of principles as outlined below.

For the purpose of consistency, we have used the same GP library[2] and the same kernel structure in time for all methods (given in Table A.1).

In order to account for the influence of random initialisation, all presented results are calculated by accumulating statistics over 30 independent repetitions, with different random initialisations and random data removal per repetition. In order

2 The GP library used is the one developed and maintained by Prof. Neil Lawrence and his team [Lawrence, 2003]. We have modified their code as well as the visualisation tools provided for the purpose of our experiments.

to ensure impartiality, we seed the random number generators identically across methods and use the same or completely analogous random initialisation schemes.

For all experiments we present our results for missing data ratios ranging from 50% to 80% with increments of 5%. With bold lettering we mark the best statistically significant results, determined by applying the Student-t test.

We provide all hyper-parameters necessary for reproducing our experiments in Table A.1, which may briefly be described as follows:

- λ: Poisson intensity, or else the expected number of missing data segments.
- N_h: Number of latent factors or initial rank for the VBMC, SVT and FPC methods.
- Q: Dimensionality of the GPDS latent covariate space $\boldsymbol{X}$. Not applicable for other algorithms.
- N_{GD}: Maximum number of gradient descent iterations. This is applicable for all gradient-based algorithms (SVT and FPC) and those employing gradient descent for kernel hyper-parameter optimisation (SS-GP, IBP-GP, Multi-GP and Multi-GPDS).
- N_{freq}: Kernel update frequency. We update the kernel hyper-parameters every N_{freq} iterations in order to reduce computational costs.
- As 'NA' we denote parameters that are 'Not Applicable'. 'RBF' stands for radial basis function kernel, 'White' for white Gaussian noise, 'Bias' for a constant term and 'SE' for square exponential kernel.

Note that the in the case of the robotic experiment, we observed that the evaluated methods performed better when optimising the kernel parameters more rarely, every 10 instead of every 5 VB iterations. Also, due to the relatively simpler nature of the robotic data, less factors were sufficient and most methods achieved optimal performance for 32 factors, whereas the Multi-GPDS performed best with 16 factors.

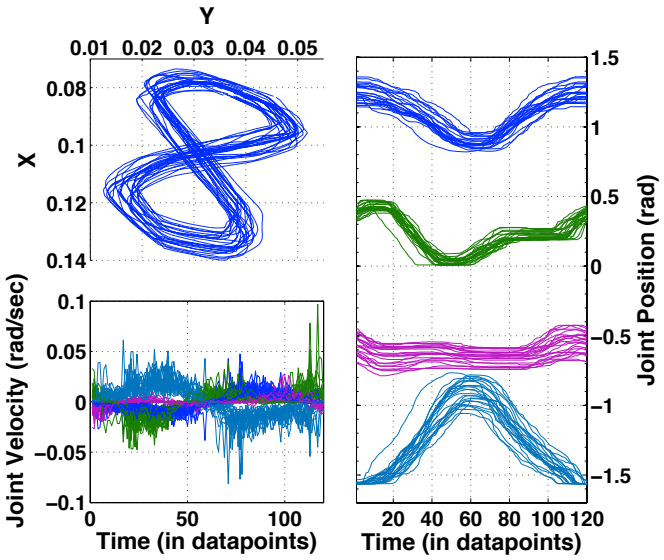


Figure 6.2: **Drawing *lazy figure* 8s**: Upper left: End effector position on the experiment's drawing plane. Lower left: Angular velocities (rad/s) per joint. Right: Angular positions (rad) per joint. (figure best viewed in colour)

Finally, our results and source code necessary to reproduce all experiments are available through the first author's website `http://www.korkinof.com` and the Personal Robotics Lab website `http://www3.imperial.ac.uk/personalrobotics`.

6.3.2 *Robotic Data*

This dataset consists of joint angle positions of a NAO humanoid robot (academic edition) recorded while drawing lazy figure 8s. The dataset includes 24 distinct demonstrations presented to the NAO robot by means of kinesthetics[3]. As can be seen in fig. 6.2, the dataset is noisy, each demonstration has different points of origin and there is significant variation among demonstrations. The task itself entails challenges for learning algorithms and for that reason is widely considered a classical benchmark [Vijayakumar et al., 2005].

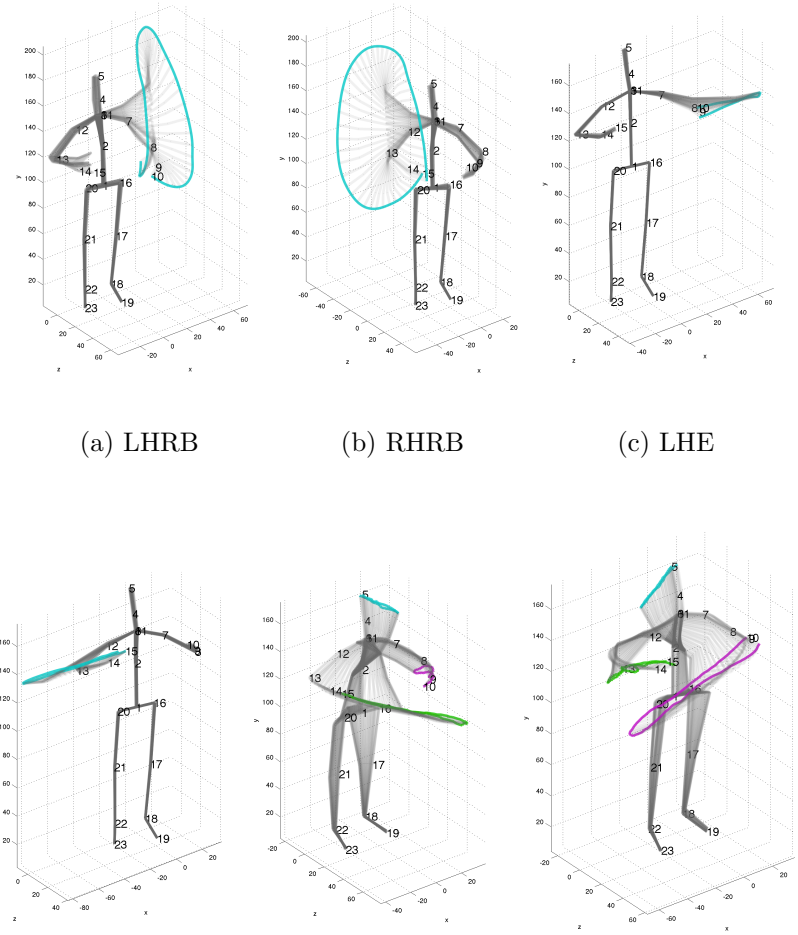
3 Manual movement of the joints in the desired positions during the demonstration.

	50%	**55%**	**60%**	**65%**	**70%**	**75%**	**80%**
Interp.	5.85(0.24)	7.08(0.36)	9.71(0.52)	10.7(0.50)	11.1(0.63)	12.3(0.96)	13.1(1.18)
SVT	4.24(0.55)	4.95(0.75)	5.56(0.78)	6.98(1.21)	7.86(1.26)	8.49(1.81)	9.54(1.30)
FPC	4.29(0.55)	4.97(0.73)	5.54(0.76)	6.92(1.16)	7.80(1.24)	8.44(1.82)	9.54(1.27)
SS-GP	2.73(0.24)	3.06(0.36)	3.47(0.52)	4.20(0.50)	4.79(0.63)	5.29(0.96)	6.56(1.19)
IBP-GP	2.92(0.54)	3.53(0.90)	3.73(0.79)	4.37(0.68)	4.87(0.75)	5.24(0.51)	5.79(0.81)
Multi-GP	2.66(0.30)	2.96(0.28)	3.29(0.37)	3.69(0.37)	4.07(0.42)	4.29(0.34)	4.66(0.32)
VBMC	2.89(0.28)	3.29(0.31)	3.71(0.37)	4.41(0.47)	5.30(0.98)	6.02(0.84)	7.05(0.90)
AR1	2.50(0.25)	2.79(0.28)	3.00(0.31)	3.38(0.37)	3.64(0.43)	3.90(0.38)	**4.24(0.31)**
Multi-GPDS	**2.20(0.24)**	**2.44(0.18)**	**2.59(0.28)**	**2.91(0.31)**	**3.32(0.49)**	**3.64(0.47)**	**4.28(0.42)**

Table 6.3: **Drawing lazy figure 8s**: Reconstruction root mean square error for all evaluated methods. Each column represents results for a different missing data ratio. The scale of the results is 10^{-2}.

The results pertaining to this experiment are presented in Table A.4. As may be observed, the Multi-GPDS achieves overall the best reconstruction accuracy, with the exception of the highest missing data ratio (80%) where its performance is statistically equivalent to the AR1. In all other cases the Multi-GPDS outperforms the AR1 by 10%-14%.

With regard to the rest of the evaluated algorithms, we may observe that the Multi-GP performs quite well in this experiment, followed by the SS-GP and the VBMC. The IBP-GP under-performs for this dataset, as the number of factors selected appears to be too low for the non-parametric prior. Finally, the SVT and FPC perform almost identically and significantly better than interpolation, but worse that all other methods.



(a) LHRB (b) RHRB (c) LHE

(d) RHE (e) ATR (f) CTR

Figure 6.3: **Dance primitives**: Figures (a)-(f) show each separate action, which include the following: (a): Left hand rotation (LHR), (b): Right hand rotation (RHR), (c): Left hand extension (LHE) (d): Right hand extension (RHE) (e): Anti-clockwise torso rotation (ATR) and (f): Clockwise torso rotation (CTR) (figures best seen in colour)

6.3.3 *Motion Capture Data*

6.3.3.1 *Dance Primitives*

This dataset consists of 6 human dance motion primitives and has previously been used in learning dance sequences by means of context-free stochastic grammars

	50%	55%	60%	65%	70%	75%	80%
Interp.	2.75(0.20)	3.31(0.28)	4.06(0.36)	4.73(0.35)	5.63(0.32)	6.44(0.43)	7.01(0.31)
SVT	2.17(0.05)	2.38(0.05)	2.73(0.06)	3.09(0.06)	3.58(0.08)	4.20(0.06)	4.90(0.11)
FPC	1.45(0.05)	1.62(0.05)	1.93(0.07)	2.24(0.06)	2.72(0.08)	3.36(0.09)	4.18(0.12)
SS-GP	1.10(0.03)	1.19(0.04)	1.34(0.04)	1.49(0.04)	1.71(0.06)	1.99(0.09)	2.47(0.18)
IBP-GP	1.07(0.03)	1.15(0.03)	1.30(0.04)	1.44(0.05)	1.62(0.06)	1.85(0.07)	2.30(0.19)
Multi-GP	1.08(0.03)	1.20(0.04)	1.43(0.05)	1.65(0.06)	1.99(0.06)	2.48(0.09)	3.11(0.10)
VBMC	**1.02(0.02)**	1.11(0.03)	1.27(0.04)	1.44(0.06)	1.66(0.05)	1.99(0.06)	2.50(0.07)
AR1	1.08(0.02)	1.19(0.03)	1.38(0.04)	1.56(0.05)	1.78(0.05)	2.04(0.06)	2.34(0.07)
Multi-GPDS	**1.00(0.02)**	**1.07(0.03)**	**1.20(0.04)**	**1.32(0.05)**	**1.47(0.05)**	**1.65(0.08)**	**1.84(0.08)**

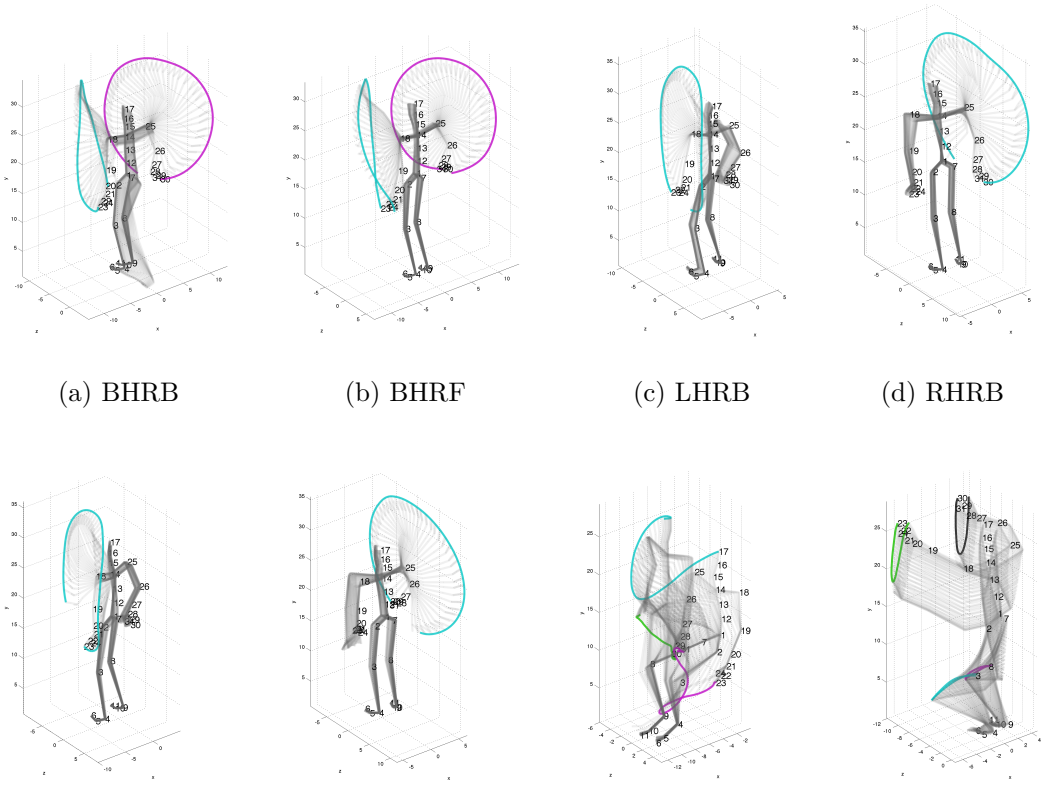
Table 6.4: **Dance Primitives Dataset**: Reconstruction RMSE for all evaluated methods. Column represents results for a different missing data ratios.

[Lee et al., 2012, 2013]. The data is recorded as time-series using an OptiTrack 8-camera motion capture system.

There are 449 sequences in total, belonging to 8 distinct actions with not necessarily the same number of repetition per action class. The data is given in raw 3D positions in time and includes only upper body motion. We illustrate the actions in fig. 6.3, where we also provide a short task description and corresponding abbreviation for quick reference. For a better understanding of the dataset we refer to the following video shared by the authors in [Lee et al., 2013]: `http://www.youtube.com/watch?v=S99ViThK050`.

Reconstruction errors for all evaluated methods are presented in Table 6.4. We observe that the differences among algorithms tend to be more prominent in higher missing data ratios, whereas their performance almost converges in lower ratios. Compared to the second best performing method, the Multi-GPDS achieves the globally best reconstruction accuracy, with differences ranging from 2.5% in lower missing data ratios, up to 20% for the highest ratio of 80%.

As far as the other methods are concerned, the IBP-GP and SS-GP perform very well in this experiment; their performance is being matched and surpassed in lower missing data ratios by the VBMC. It becomes obvious that the GP prior has an



(a) BHRB (b) BHRF (c) LHRB (d) RHRB

(e) LHRF (f) RHRF (g) SoC (h) SHE

Figure 6.4: **HDM05 Dataset**: Figures (a)-(h) show each separate action, which include the following: (a): Both hand rotation backwards (BHRB), (b): Both hand rotation forwards (BHRF), (c): Left hand rotation backwards (LHRB), (d): Right hand rotation backwards (RHRB), (e): Left hand rotation forwards (LHRF), (f): Right hand rotation forwards (RHRF), (g): Sitting down on chair (SoC) and (h) Squat with both hands extended (SHE) (figures best seen in color)

advantage in more severely corrupt sequences. Finally, the gradient-based methods SVT and FPC perform considerably worse, with the FPC outperforming the SVT by a large margin.

	50%	55%	60%	65%	70%	75%	80%
Interp.	0.71(0.05)	0.88(0.05)	1.04(0.06)	1.26(0.06)	1.46(0.08)	1.73(0.07)	2.00(0.11)
SVT	0.70(0.01)	0.80(0.01)	0.92(0.01)	1.07(0.02)	1.27(0.03)	1.56(0.02)	1.90(0.03)
FPC	0.67(0.01)	0.78(0.01)	0.90(0.02)	1.06(0.02)	1.27(0.03)	1.57(0.02)	1.92(0.03)
SS-GP	0.46(0.01)	0.51(0.01)	0.55(0.02)	0.61(0.02)	0.69(0.02)	0.81(0.02)	1.04(0.05)
IBP-GP	0.45(0.01)	0.48(0.01)	0.53(0.02)	0.58(0.02)	0.64(0.02)	0.76(0.02)	0.96(0.04)
Multi-GP	0.44(0.004)	0.49(0.01)	0.55(0.01)	0.62(0.01)	0.71(0.01)	0.85(0.01)	1.03(0.02)
VBMC	0.52(0.004)	0.59(0.01)	0.67(0.01)	0.78(0.01)	0.94(0.02)	1.20(0.02)	1.53(0.03)
AR1	0.52(0.01)	0.57(0.01)	0.61(0.01)	0.66(0.01)	0.71(0.01)	0.78(0.01)	0.85(0.01)
Multi-GPDS	**0.39(0.01)**	**0.42(0.01)**	**0.45(0.01)**	**0.49(0.01)**	**0.53(0.01)**	**0.59(0.02)**	**0.71(0.04)**

Table 6.5: **HDM05 Dataset**: Reconstruction RMSE for all evaluated methods. Each column represents results for a different missing data ratio.

6.3.3.2 *HDM05 Dataset*

The HDM05 database [Müller et al., 2007] contains motion capture data recorded at the Hochschule der Medien (HDM) in the year 2005 under the supervision of Bernhard Eberhardt. It consists of around 70 motions classes, with up to 50 repetitions of each action from a variety of actors and recorded using a Vicon system. In its entirety, the database contains over 1500 systematically recorded and documented motion sequences with a total duration of over 3 hours.

For the purpose of this experiment we have selected the 8 actions illustrated in fig. 6.4. We have included primitive actions closely related to the ones used in our "dance primitives" experiment, such as LHRB, RHRB, LHRF and RHRF (for these abbreviations we refer to fig. 6.4), as well as more complex actions consisting of a combination of those primitive actions and involving both hands, such as BHRB and BHRF. Finally, two actions involving full body motion (SoC and SHE) have been included in order to increase the difficulty of the experiment. Additionally, as tasks (g) and (h) are relatively unrelated to tasks (a)-(f), we expect limited factor sharing to take place across those two groups.

We present our results in Table 6.5. We may observe that the Multi-GPDS is able to achieve the best reconstruction accuracy by a margin ranging from 11%-23% compared to the second best performing method.

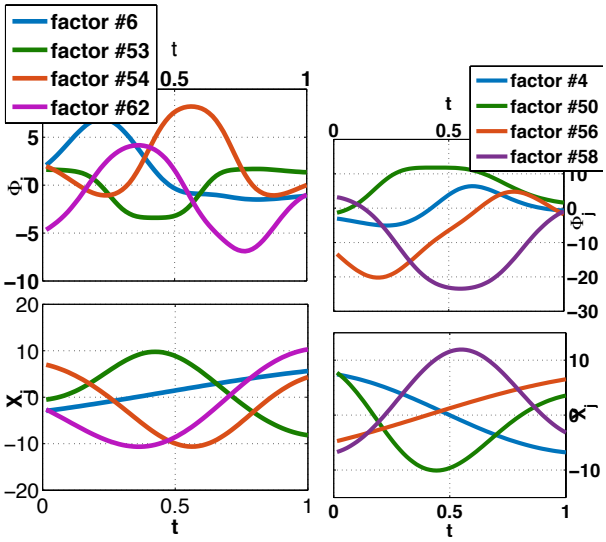
As for the other methods, the autoregressive model (AR1) performs best in highest missing data ratio, but is outperformed by the IBP-GP for lower ratios. The IBP-GP exhibits consistently better performance than the SS-GP, revealing the benefits of the non-parametric prior employed. Finally, the Multi-GP is also consistently better than the VBMC in this experiment indicating the importance of the GP prior. Finally, the gradient-based methods, SVT and FPC, achieved similar but considerably worse performance.

6.3.4 *Discussion*

6.3.4.1 *Model Properties*

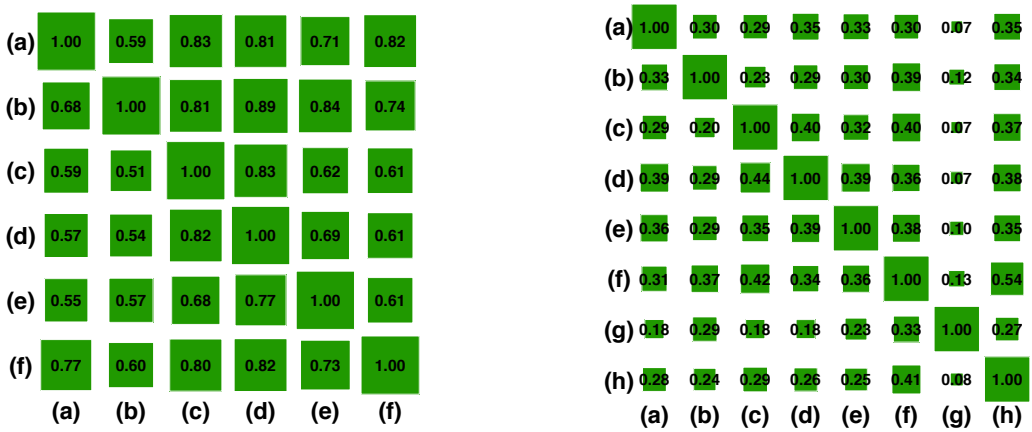
In this section we comment upon the properties of the Multi-GPDS, as they manifest themselves during our empirical evaluation, and investigate the extent to which our model meets its aims.

TEMPORAL FACTORS The purpose of employing GPDS priors on the model factors was to achieve a non-linear sequential structure with high descriptive power. In order to illustrate the effect of the GPDS prior, we have formed a set $\mathcal{F}$ consisting of the single most dominant factor per sequence. That means that factor index $j \in \mathcal{F}$ iff $\exists n$ such that $j = \arg\max_k \left[w_{nk}^2\right]$. We have then randomly chosen 4 factors from $\mathcal{F}$ to depict them in fig. 6.5. We observe that the latent covariates $\boldsymbol{X}_j$ form smooth trajectories generally more complex than linear. The resulting temporal factors $\boldsymbol{\phi}_{ij}$ are considerably more non-linear and complex. Their functional form resembles the one achieved by time-varying length-scales.



(a) Dance primitives dataset. (b) HDM05 dataset.

Figure 6.5: Illustration of several dominant posterior Multi-GPDS covariates and factors in time. We can see that the GPDS prior yields highly non-linear factor and is ultimately a significantly more descriptive prior on the temporal factors.



(a) Dance primitives dataset. (b) HDM05 dataset.

Figure 6.6: Factor sharing confusion matrices. In order to avoid cluttering these two subfigures, we have numbered the action classes in the order they appear in figures 6.3 and 6.4.

FACTOR SHARING AND MULTI-TASK LEARNING Our aim of utilising the redundancies that manifest themselves among repetitions of the same task, as

	50%	55%	60%	65%	70%	75%	80%
	fig. 6.4a: Both hand rotation backward (BHRB)						
STL	0.63(0.08)	0.70(0.07)	0.81(0.09)	0.93(0.10)	1.09(0.15)	1.13(0.15)	1.22(0.19)
MTL	**0.12(0.01)**	**0.15(0.01)**	**0.17(0.01)**	**0.20(0.01)**	**0.24(0.02)**	**0.30(0.02)**	**0.45(0.07)**
	fig. 6.4g: Sit-down on chair (SoC)						
STL	0.78(0.04)	0.85(0.04)	0.93(0.05)	1.03(0.07)	1.15(0.10)	1.44(0.17)	**1.63(0.12)**
MTL	**0.49(0.04)**	**0.57(0.04)**	**0.67(0.04)**	**0.77(0.06)**	**0.87(0.06)**	**1.11(0.10)**	**1.61(0.22)**

Table 6.6: **HDM05 Dataset**: Reconstruction RMSE per action class for the Multi-GPDS trained under a Single-task learning (STL) and Milti-task learning (MTL) settings.

well as across different tasks, are addresses by learning a dictionary shared among all sequences. This choice allows for multi-task learning and ad-hoc component sharing determined by the magnitude of the factor loading weights. Note that the same property is shared by all evaluated multi-task models.

In this section we attempt to quantify the factor sharing achieved by the Multi-GPDS in specific and present our results in fig. 6.6. We do so by measuring the mean square error of the absolute factor loadings $\boldsymbol{w}_n$ between tasks. More specifically the factor sharing between task i and task j, denoted as $1/d_{ij}$ is quantified as follows:

$$d_{ij} = \frac{1}{|\mathcal{I}_i|\,|\mathcal{I}_j|} \sum_{n \in \mathcal{I}_i} \sum_{n' \in \mathcal{I}_j} ||\boldsymbol{w}_n| - |\boldsymbol{w}_{n'}|| \tag{6.46}$$

where $\mathcal{I}_i$ is the set of all sequence indices belonging to task i and $|\mathcal{I}_i|$ is the cardinality of this set. The similarities are then normalised by dividing with their maximum value and are presented in the form of a confusion matrix in fig. 6.6.

The factor sharing property the Multi-GPDS, not only leads to smaller, more efficient models taking advantage of commonalities, but also to significant performance gains as may be seen in Table 6.6. More specifically, we observe an impressive 3-5 times better reconstruction accuracy in task BHRB (fig. 6.4a) and up to 40% in task SoC (fig. 6.4g).

Tasks for which we present results were purposefully chosen in order to highlight the fact that those with higher level of commonality to others (such as BHRB, fig. 6.6b) have naturally benefitted more by the multi-task learning employed. In contrast, for relatively more unique tasks (such as SoC, fig. 6.6b) this benefit diminishes, but is nevertheless still significant.

6.4 CONCLUSION

In this chapter we propose a novel multi-kernel GPDS factor model for rectifying severely damaged sequences. Our aims are threefold: 1) To utilise the redundancy manifesting itself among different demonstrations of the same task, 2) To allow multi-task learning among similar tasks and 3) To take into account the sequential nature of the data. We have devised efficient variational Bayesian inference and have subjected our model to rigorous empirical evaluation on one robotic and two motion capture datasets. We achieve significant improvements in terms of reconstruction accuracy compared to popular methods in the literature. Finally we investigate the properties of out model as they arise from the empirical evaluation.

7

CONCLUSION

This chapter concludes the current thesis. We begin with a quick overview including the main achievements of this work. This is followed by a discussion on the limitations of the presented methods, along with a description of our current work towards extending the proposed methods and directions for future research.

7.1 OVERVIEW

As previously mentioned, the main sources of motivation and inspiration for this research have been the following:

- The convergence of the fields of robotics and machine learning initiated by technological advances bringing AI closer to the physical world.
- The belief that the opportunity for more impactful research lies in the intersection of fields.
- Fine-tayloring widely utilised methods can multiply the impact of this research.

- Closing the feedback loop between theory and practice, something that the field of machine learning has neglected for years, can only be beneficial in terms of formulating algorithms that are more effective in practice.

We began by focusing on the following offline LbD algorithms:

- **DPM** [Chatzis, Korkinof, and Demiris, 2012a]: A fully Bayesian approach towards LbD, which inherits all the merits of conducting inference rather than optimisation (i.e. minimal overfitting). The approach also employs a non-parametric DP prior, which allows for minimal tuning and automatic data-driven model size estimation.

- **QMM** [Chatzis, Korkinof, and Demiris, 2012c]: In this case we are able to achieve significantly enhanced reconstruction accuracy for a given model size.

We demonstrated the efficacy of both in one- and multi-shot learning by demonstration scenarios, such as drawing Lazy Figure 8s and other motions. With the DPM we are able to achieve smaller models without compromising the model descriptive power. With the QMM, we observe state-of-the-art reconstruction accuracy, outperforming the GMM, LGPR and DPM in most occasions.

However, we acknowledged that more promising practical applications in robotics require a more adaptive and flexible learning algorithms, able to refine and adapt their estimates of the phenomena they model. For that purpose, we introduce an online learning algorithm for quantum mixtures based on the online EM. In this case we are able to formulate a stable online algorithm, that outperforms many of its widely used counterparts, such as the LWPR, the oGP, oGMM and oSMM in a series of benchmark datasets, as well as a LbD case study involving a NAO robot drawing Lazy Figure 8s.

Still, LbD is not possible without a sufficient amount of example data from which to learn. That is not an issue in itself as data collection is nowadays relatively effortless, cheap and fast. There is a multitude of datasets demonstrating a variety of different human behaviours freely available online; a prolific source of information

to be harvested for the purpose of machine intelligence. What is problematic, however, is that few of those datasets are "clean" and readily usable. It is, in fact, not the cost or effort of collecting the data that inhibits our progress, but rather the effort of "cleaning". For that purpose we devised a sequential factor model to rectify severely damaged sequences in a multi-task, multi-kernel, totally unsupervised manner. We demonstrated the efficacy of the proposed method in a series of applications including human motion (Motion Capture) and robotic joint data. We have shown that the proposed approach can perform very well even with severely damaged sequences and outperforms many relevant gradient- or inference-based methods, such as the SVT, FPC, SS-GP, IBP-GP, Multi-GP and VBMC.

7.2 MODEL LIMITATIONS

In this section we discuss some of the limitations associated with the methods presented in this thesis.

To begin with, we should acknowledge that there is no one-solution-fits-all in machine learning; each method is formulated with certain assumptions and postulations in mind, which happen to be often mutually exclusive. This does not mean that general and multi-purpose methods do not exist. However, such approaches usually represent a family or a combination of methods and/or employ appropriate structure learning or model selection in order to accommodate many possible data attributes.

Second, we should note that the purpose of the experimental evaluations presented in this thesis is comparative. We have not necessarily chosen our experiments with regard to their suitability to the evaluated methods or the evaluated methods with regard to their suitability to the experiments. Most experiments were chosen so as to provide a common, but challenging and realistic benchmark allowing us to reveal comparative merits and expose shortcomings in the evaluated algorithms.

Thus, the models considered do not necessarily provide the best possible solution to the settings in which they were evaluated, but rather the experimental settings pose a significant challenge for our models

7.2.1 *Limitations of Gaussian Mixtures*

The most important limitation of Gaussian mixtures, regardless of whether they are infinite or quantum or spatially constrained, stems from the assumption of i.i.d. data emission. This attribute makes all relevant models less suitable for sequential data, than HMMs for instance, and/or less suitable for position-velocity trajectories than Dynamical Systems. This innate limitation is intensified in the case of the online versions of those algorithms.

However, this well-documented limitation should be considered in the right perspective. More specifically:

- Our contributions do not exclude, but rather pave the way towards quantum mixtures of Dynamical Systems or HMMs with quantum mixture emission densities for instance.
- In the online case, inference for HMMs and/or Dynamical Systems would require Particle Filtering (PF) or Sequential Monte Carlo approaches which are not guaranteed to converge and would also require a significant warm-up period, also referred to as bootstrapping.
- There is a plethora of LbD approaches where the i.i.d. property is desirable.
- There are instances where non-sequential models were effectively used in situations where sequential models would be more appropriate, for example in [Vijayakumar et al., 2005] or this interesting medical application [Mylonas et al., 2013].

As with the quantum mixtures, we do observe significant performance gains accompanied by higher numerical stability, which may be attributed to smoother mixing weights as a result of the quantum disturbance. A restrictive factor for those models is that they cannot be easily extended due to non-conjugacy issues.

7.2.2 *Limitations of GPDS factor models*

We can identify three main limitations of using GPDS in dynamic factor models. First of all, they are expected to work well in situations where the data may be analysed into a low or a moderate number of relatively complex non-linear factors (as in the MoCap dataset). This means that in the case of simpler trajectories (Spoken Arabic Digits dataset) or when the data can be represented as a Linear Dynamical Systems (LDS) (Lazy Figure 8s dataset) the benefit of the model is minimised. This is reflected in our experiments, where however the Multi-GPDS still outperforms simpler methods, as the trajectories are complex enough in order to benefit from the model's complexity.

Second, the computational complexity of the approach, although manageable due to its efficient formulation using inducing variables, is nevertheless still significant. This means the computational burden induced would be forbidding in case we wish to increase the size of the model to more than 200 factors for instance.

Third, using GPs limits us to VB inference and the problem is mainly focused on the process of learning the kernel hyper-parameters. Although methods for sampling the GP hyper-parameters do exist in the literature, i.e. Hamiltonian Monte Carlo (HMC) and Elliptical Slice Sampling (ESS), type-II maximum likelihood within VB remains by far the most effective learning strategy, according to our experence. This means that pursuing alternative decompositions remains problematic.

7.2.3 *Limitations of VB inference*

Variational inference has been very popular in recent years, mainly due to the fact that it is an elegant approximate inference method yielding analytical, iterative updates for model parameters. One of its biggest advantages is that it scales well in terms of complexity. More specifically the computational costs induced are of the same order of magnitude with Gibbs, but VB typically requires much fewer iterations to converge. More generic sampling schemes pose a significantly higher computational burden.

The main shortcoming of VB is that it requires posterior conjugacy and it is very awkward to handle non-conjugate models within this framework, if at all possible. For that purpose there have been numerous extensions to VB for non-conjugate models. The proposed approaches tend to induce significant computational burden and require a great deal of fine tuning inhibiting their widespread adoption.

Sampling algorithms on the contrary tend to deal with non-conjugate models more naturally, but more generic methods may suffer from slow mixing and convergence issues.

7.3 WORK IN PROGRESS

For the purpose of sequence completion, we are currently exploring alternative tensor factorisations, such as the Canonical Decomposition (CD) and the Tucker decomposition in combination with temporal factors consisting of general Linear Dynamical Systems (LDS). Our aim is to alleviate the computational complexity of the GPDS factors.

In this case we have observed that VB is not an effective inference method, especially so for the CD, but is also not to be preferred in learning LDS parameters. Consequently, we employ Gibbs sampling, which enjoys rapid mixing and rela-

tively small computational complexity. Preliminary results show very promising performance in terms of the quality of sequence reconstruction.

7.3.1 *Proposed Approach*

Motivated by relevant approaches in the literature, we proceed to formulate a model with the following main attributes:

- A fully Bayesian approach to temporal tensor decomposition.
- A more scalable and compact representation, using the Canonical Decomposition (CD).
- A deeper hierarchical structure allowing us to achieve a better representation of our data.

For more information on tensor algebra and the related approaches in the literature, we refer to Appendix B.

7.3.2 *Model Formulation*

Let us assume a tensor viewpoint of our dataset, which we denote as $\mathcal{Y} \in \mathbb{R}^{N \times N_T \times N_y}$. We begin by applying a Canonical Decomposition (CD), such as the one suggested in [Xiong et al., 2010], with the difference that we assume heteroscedastic noise, with different precision per demonstration which as we have observed amounts for performance gains:

$$y_{nit} \sim \prod_{n=1}^{N} \prod_{t=1}^{N_T} \prod_{i=1}^{N_y} \mathcal{N}\left(y_{nit} | \sum_{j=1}^{N_h} w_{nj} g_{ij} z_{tj}, \beta_n^{-1}\right) \tag{7.1}$$

with the following prior specifications:

$$\boldsymbol{w}_n \sim \mathcal{N}\left(\boldsymbol{w}_n|\boldsymbol{\mu}_w, \boldsymbol{\Lambda}_w^{-1}\right) \tag{7.2}$$

$$\boldsymbol{g}_i \sim \mathcal{N}\left(\boldsymbol{g}_i|\boldsymbol{\mu}_g, \boldsymbol{\Lambda}_g^{-1}\right) \tag{7.3}$$

$$\boldsymbol{\mu}_w, \boldsymbol{\Lambda}_w \sim \mathcal{N}\left(\boldsymbol{\mu}_w|\boldsymbol{\mu}_0, \left(\beta_0\boldsymbol{\Lambda}_w\right)^{-1}\right)\mathcal{W}\left(\boldsymbol{\Lambda}_w|\boldsymbol{W}_0, \nu_0\right) \tag{7.4}$$

$$\boldsymbol{\mu}_g, \boldsymbol{\Lambda}_g \sim \mathcal{N}\left(\boldsymbol{\mu}_g|\boldsymbol{\mu}_0, \left(\beta_0\boldsymbol{\Lambda}_g\right)^{-1}\right)\mathcal{W}\left(\boldsymbol{\Lambda}_g|\boldsymbol{W}_0, \nu_0\right) \tag{7.5}$$

where $\mathcal{W}(\cdot)$ is the Wishart distribution, defined as follows:

$$\mathcal{W}\left(\boldsymbol{\Lambda}|\boldsymbol{W}_0, \nu_0\right) = \frac{1}{\mathcal{C}}\left|\boldsymbol{\Lambda}\right|^{\frac{1}{2}(\nu_0 - D - 1)} e^{-\frac{1}{2}tr\left\{\boldsymbol{W}_0^{-1}\boldsymbol{\Lambda}\right\}} \tag{7.6}$$

As for the temporal loadings $\boldsymbol{z}_t$, we choose to extend the approach in [Xiong et al., 2010] and place a full LDS prior on them as follows:

$$\boldsymbol{z}_t \sim \mathcal{N}\left(\boldsymbol{z}_t|\boldsymbol{C}\boldsymbol{x}_t, \boldsymbol{\Lambda}_z^{-1}\right) \tag{7.7}$$

$$\boldsymbol{x}_1 \sim \mathcal{N}\left(\boldsymbol{x}_1|\boldsymbol{\mu}_x, \boldsymbol{\Lambda}_x^{-1}\right) \tag{7.8}$$

$$\boldsymbol{x}_t \sim \mathcal{N}\left(\boldsymbol{x}_t|\boldsymbol{A}\boldsymbol{x}_{t-1}, \boldsymbol{\Lambda}_x^{-1}\right) \tag{7.9}$$

$$\boldsymbol{C} \sim \mathcal{MN}\left(\boldsymbol{C}|\boldsymbol{M}_C, \boldsymbol{\Lambda}_z, \boldsymbol{K}_C\right) \tag{7.10}$$

$$\boldsymbol{A} \sim \mathcal{MN}\left(\boldsymbol{A}|\boldsymbol{M}_A, \boldsymbol{\Lambda}_x, \boldsymbol{K}_A\right) \tag{7.11}$$

$$\boldsymbol{\Lambda}_z \sim \mathcal{W}\left(\boldsymbol{\Lambda}_z|\boldsymbol{W}_0, \nu_0\right) \tag{7.12}$$

$$\boldsymbol{\mu}_x, \boldsymbol{\Lambda}_x \sim \mathcal{N}\left(\boldsymbol{\mu}_x|\boldsymbol{\rho}_0, \left(\beta_0\boldsymbol{\Lambda}_x\right)^{-1}\right)\mathcal{W}\left(\boldsymbol{\Lambda}_x|\boldsymbol{W}_0, \nu_0\right) \tag{7.13}$$

where as $\mathcal{MN}(\cdot)$ we denote the matrix-Normal distribution defined as follows:

$$\mathcal{MN}\left(\boldsymbol{A}|\boldsymbol{M}, \boldsymbol{\Lambda}, \boldsymbol{K}\right) = \frac{\left|\boldsymbol{K}\right|^{\frac{d}{2}}\left|\boldsymbol{\Lambda}\right|^{\frac{m}{2}}}{\left|2\pi\right|^{\frac{m}{2}}} e^{-\frac{1}{2}tr\left\{(\boldsymbol{A}-\boldsymbol{M})^T\boldsymbol{\Lambda}(\boldsymbol{A}-\boldsymbol{M})\boldsymbol{K}\right\}} \tag{7.14}$$

Note that with the aforementioned formulation we have achieved the following objectives:

- We have placed a general LDS prior on the temporal CD loadings which can express any AR_m process.
- We achieve a deeper level representation than the one in [Rogers et al., 2013], allowing us to progressive decrease the dimensionality of the temporal loadings.

- We employ a full matrix prior for the transfer matrices $\boldsymbol{A}$ and $\boldsymbol{C}$ able to capture dependencies among rows and columns.
- We formulate a fully conjugate, fully Bayesian model for which Gibbs inference is tractable.
- As posterior calculations for $\boldsymbol{z}_t$ can be done in parallel and $\boldsymbol{x}_t$ is a variable of much lower dimensionality, thus inference scales very well with the size of the dataset.

We present our preliminary results in Appendix C and the conditional posteriors necessary to implement the Gibbs sampler in Appendix D.

7.4 FUTURE WORK

7.4.1 *Quantum Mixtures*

As we have previously mentioned, one of the main shortcomings of the QMM is that its awkward mathematical form is an inhibiting factor for applying the concept to different settings. Gradient based methods are applicable in this case, but yield point estimates w.r.t. the parameters, instead of full posterior distributions.

Consequently, we shall orient our future work towards MCMC sampling algorithms, which are becoming widely popular with the advent of progressively more complex models. Although the general Metropolis-Hastings algorithm could be used to sample from the QMM, we plan to also consider the Hamiltonian Monte Carlo (HMC), which could be readily applied by using the MLE derivatives shown in Chapter 4. For the oQMM we would like to explore the promising sequential MCMC approaches, such as particle filtering, which are known to be able to handle model non-conjugacies.

7.4.2 *Completing Misaligned Sequences*

One issue that we have frequently encountered is the difficulty of machine learning algorithms to handle (temporally or spatially) misaligned demonstrations. In that case models that usually appear in the literature, would not be able to capture reoccurring patters, ultimately leading to poor utilisation of redundant information. To ameliorate this issue we frequently resort to patio-temporal sequence alignment as a preprocessing step.

We plan to address the issue of temporally misaligned sequences in our future research by employing shift-invariant convolutional factor models. This approach could lead to very powerful models, as shown in convolutional Boltzmann Machines, able to extract recurring or misaligned factors. A severe disadvantage, in the Bayesian case, is the multi-modality of the corresponding posterior distributions. For that purpose, the only relevant Bayesian approach towards convolutional FAs [Chen et al., 2013], employs a scheme known as max-pooling.

Finally, handling spatial misalignment is slightly easier, as we need only learn a static translation and rotation for each sequence. However, in a Bayesian setting, it is non-trivial to ensure that the constrains associated with the rotation matrix are met, especially so in the case of MCMC. Constraints, may however be introduced in VB by employing a gradient-ascent step, similar to the approach in [Luttinen & Ilin, 2010], but for a different purpose.

7.4.3 *Action Recognition from Incomplete Data*

Further extending the concept of missing data, we wish to integrate action recognition into our sequence completion approach. The main concept is that the demonstration-specific loading weights of the decomposition are representative of each action/task and can be utilised in order to perform classification on severely

damaged sequences. For that purpose we have formulated a max-margin dynamical tensor CD for simultaneous sequence completion and classification. We believe this model could provide an automated and integrated approach towards the problem in question.

7.5 CONCLUDING REMARKS

Despite the limitations we have outlined here, we believe that the significance of this work lies in the following achievements stemming from this thesis:

- We have introduced offline methods that address important issues in trajectory LbD and can be beneficial for enhancing the methods' practical applicability; such as representation accuracy and optimal model selection.
- We have formulated an agile and highly adaptive algorithm for online learning, which achieves state-of-the-art performance and can readily be applied in a plethora of practical applications replacing popular GMM-based methods.
- We have proposed an approach towards completing severely damaged sequences in an unsupervised manner, which can effortlessly be used to "clean" large amounts of recorded motion data from corruption.

Overall, we believe that this work paves the way for numerous extensions and useful applications of the proposed methods. We hope that it is going to provide new directions for us and other researchers to pursue in the future and will initiate interesting developments in both robotics and machine learning.

Part IV

APPENDICES

A

MULTI-GPDS SUPPLEMENTARY RESULTS

In this appendix we present some supplementary results pertaining to the Multi-GPDS, the method proposed in Chapter 6.

More specifically, two of the experiments presented here are the same as the ones presented in Chapter 6 (Robotics dataset and Dance Primitives dataset). The third experiment presented (speech dataset) is not included in the main body of this thesis.

It should be noted that all results presented here use a different data removal scheme to the one presented in Algorithm 1 of Section 6.3.1.2. This alternative scheme, outlined in the next section, is simpler, but provides less control on the final percentage of removed data and for that reason it was not preferred in the main body of this thesis. Nevertheless, we choose to include those results here for the purpose of brevity.

A.1 DATA REMOVAL

The data removal scheme we followed in order to obtain the results presented in this Appendix is outlined in pseudocode in Algorithm 2. In more detail data removal takes place as follows:

DESCRIPTION OF ALGORITHM 2

- `[line 1]` Draw the number of missing segments q according to a Poisson distribution with intensity λ.
- `[line 2]` Calculate the mean length of the missing data segments μ, based on the ration r, the length of the sequence T and the number of missing segments q.
- `[line 3]` Calculate the variance of the noise we are going to add to the segment length σ^2.
- `[line 4]` Draw a vector $\boldsymbol{m}$ of dimension q of normally distributed segment lengths, with mean μ and variance σ^2.
- `[line 5]` Search for a non-overlapping segment of length m_k to remove, in a random order.
- `[line 8]` If found, then remove the corresponding data points.
- `[line 10]` Otherwise remove a random overlapping segment.

A.2 RESULTS

A.2.1 *Speech Data: Spoken Arabic Digits*

For the purpose of this experiment, we have used a dataset containing short time series of Mel-frequency Cepstrum Coefficients (MFCC) corresponding to spoken Arabic digits, which has previously been used in [Hammami & Sellam, 2009; Hammami & Bedda, 2010]. It includes a total of 8800 (10 digits x 10 repetitions x 88 speakers) time-series of 13 cepstral coefficients. The sound data are collected from 44 male and 44 female native Arabic speakers between the ages of 18 and 40.

Algorithm 2 Random data removal algorithm.

Inputs:

- $r \in \mathbb{R}$: Desired missing data ratio.
- $\lambda \in \mathbb{N}$: Intensity of Poisson distribution for the number of missing data segments.
- $T \in \mathbb{N}$: The number of data-points in the data sequence.

Outputs:

- $q \in \mathbb{N}$: The number of missing segments in the sequence.
- $\boldsymbol{m} \in \mathbb{N}^q$: Vector containing the lengths of missing data segments.
- $\boldsymbol{s} \in \mathbb{N}^q$: Vector containing the starting points of the missing data segments.

Begin:

$q|\lambda \sim Poisson(\lambda)$
$\mu = \frac{rT}{q}$
$\sigma^2 = 0.1\mu$
$\boldsymbol{m} \sim round\left(\mathcal{N}(\mu\mathbf{1}, \sigma^2\boldsymbol{I})\right)$
for k=1...q **do**
  Search randomly for any non-overlapping segment of length m_k to remove
  if found **then**
    Remove data-points
  else
    Remove data-point from a random overlapping segment
  end if
end for

End

Return: q,$\boldsymbol{m}$,$\boldsymbol{s}$

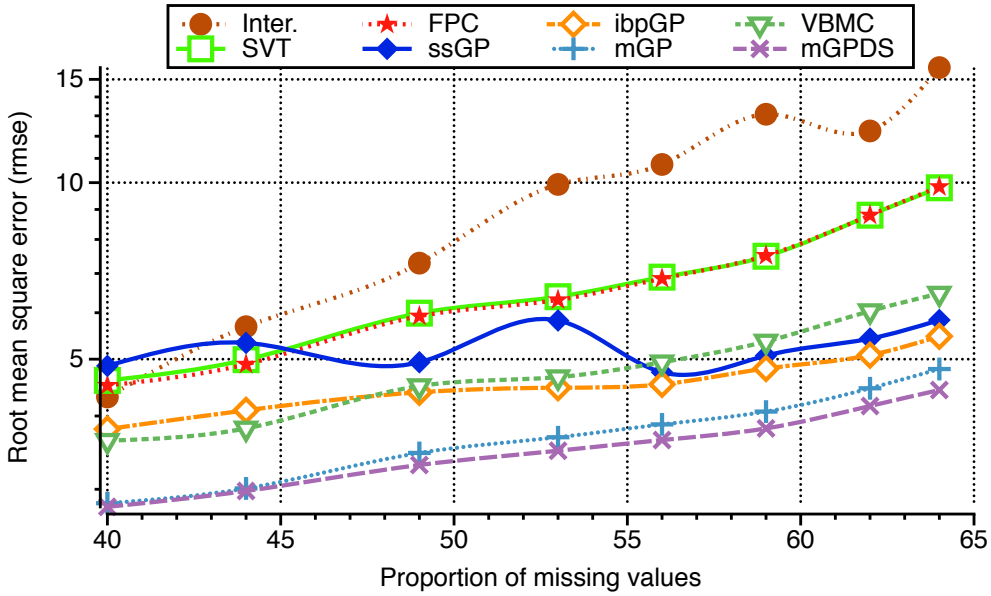


Figure A.1: **Drawing *lazy figure* 8s**: Reconstruction root mean square error for all evaluated methods. The scale of the results in this graph is 10^{-2}.

We have discarded sequences containing 20 data-points or less and have subjected the remaining 5320 series to mild preprocessing. More specifically, we have applied a low-pass filter across time, with cut-off frequency set at 25%, and have fixed the sequence length without synchronising them in time[1]. Finally, we have reduced the number of sequences to 280 (200 training + 80 testing) by taking one every 20 and have respected the training-testing segmentation provided by the authors.

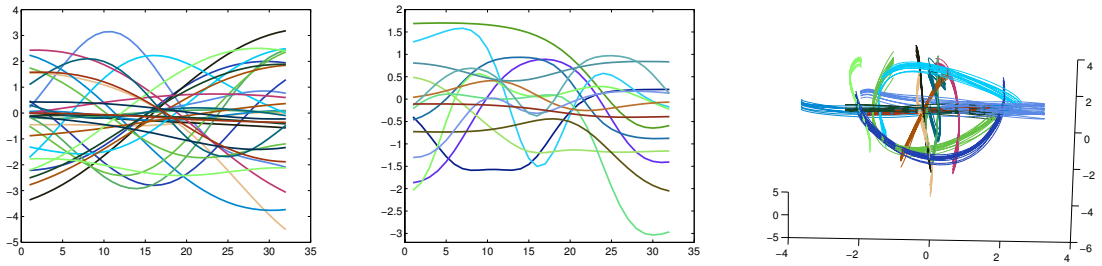
In fig. A.3, we present a mean across repetitions of each digit and for the groups of male and female speakers separately. As we may observe, the frequency footprint is significantly different for each digit. Across male and female subjects it appears to be similar, but with the peaks and troughs being generally more prominent in male speakers. Finally, it should be noted that there is significant variability across demonstrations of the same digits and for the same speaker gender, something that is not depicted in the diagrams we present.

1 We do so, by first calculating the mean length of all sequences, then we under-sample the longer and interpolate the shorter ones.

		SVT	FPC	ssGP!	ibpGP	mGP	VBMC	mGPDS!
Robotic	λ	2	2	2	2	2	2	2
	N_h	32	32	32	32	32	32	16
	N_L	-	-	-	-	-	-	1
	N_{GD}	500	500	50	50	50	50	50
Speech	λ	4	4	4	4	4	4	4
	N_h	32	32	32	32	32	32	32
	N_L	-	-	-	-	-	-	2
	N_{GD}	500	500	10	10	10	10	10
MoCap	λ	5	5	5	5	5	5	5
	N_h	-	-	-	-	-	-	1
	N_L	64	64	64	64	64	64	64
	N_{GD}	500	500	10	10	10	10	10

Table A.1: Main hyper-parameters used for each experiment: λ: Poisson mean number of gaps. N_h: number of latent factors or initial rank. N_{GD}: Maximum number of gradient iterations. N_L: Latent covariate dimensionality (GPDS).

Our purpose in this experiment is twofold, first we wish to evaluate the reconstruction error for each method and second to establish whether trajectory reconstruction also translates into higher classification rates. For that reason, we have used two classifiers, namely a k-nearest neighbors (k-NN) and a Gaussian Process Classification (GPC). Applying a 1-NN classifier to the full preprocessed dataset performed best with accuracy 92.63%, not far below the accuracy achieved by the authors' proposed methods [Hammami & Sellam, 2009; Hammami & Bedda, 2010]. As for the reduced dataset of 280 sequences, classification accuracy is 90% for a 3-NN and 91.25% for the GPC. It should be noted that evaluating the methods according to the misclassification error of reconstructed data, introduces further random fluctuations and for that reason we have accumulated statistics over 100 random initialisations. For the same reason we have preferred a 3-NN classifier, which performed more consistently in this case.



(a) Latent covariates with time $\boldsymbol{X}_j(t)$.

(b) Latent factor with time $\boldsymbol{\phi}_{1j}(t)$.

(c) Latent factors with $\boldsymbol{X}_j$: $\boldsymbol{\phi}_{ij}(\boldsymbol{X}_j)$.

Figure A.2: **Spoken Arabic Digits**: Graphical representation of some of the hidden space inferred using the mGPDS. We have isolated the prevalent factors for each sequence and plot the following: **(a)** Latent covariate against time: $\boldsymbol{X}_j(t)$, **(b)-(c)** Latent factors against time for the first two spectral dimensions: $\boldsymbol{\phi}_{ij}(t)$ and **(d)** All spectral dimensions against the latent covariate: $\boldsymbol{\phi}_{ij}(\boldsymbol{X}_j)$. (figure best seen in colour)

Our results are presented in Tables A.2 and A.3 for the root mean square error (RMSE) and misclassification error respectively. The results of Table A.3 in particular are presented as the excess error above the baseline performance of the classifier achieved on fully observed bata. In fig. A.2, we present the latent space inferred by the Multi-GPDS. We can observe that the latent covariates $\boldsymbol{X}_j$ exhibit complex, but regular and symmetric trajectories. The actual GPDS factors however are more complex in order to accommodate the data.

Regarding the reconstruction errors, we can observe that all methods perform consistently, with results varying only slightly. The Multi-GPDS outperforms other methods in all missing data ratios and the perceived difference is statistically significant at the confidence level of 5% or lower. We can also observe that the point-wise methods, namely the SVT and FPC, outperform the dictionary based methods with "spike and slab" and IBP priors. Performance gains are generally limited in this experiment, which may be attributed to the limited redundancy and high variability of the data, however they translate into notable gains in classification accuracy.

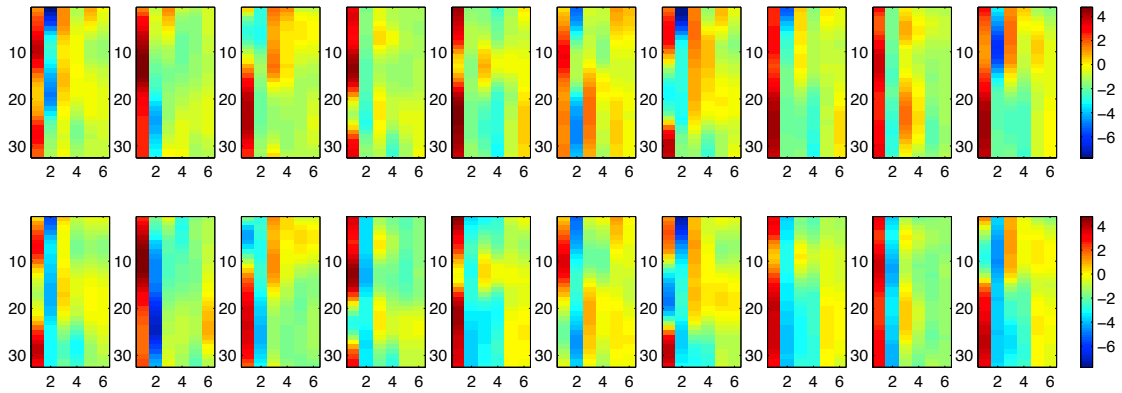


Figure A.3: **Spoken Arabic digits**: Mean of the first 5 out of 13 MFCC spectral dimensions in time for male (line one) and female (line two) speakers separately. Columns from left to right represent spoken digits 0-9. (figure best seen in colour)

As far as the misclassification error of the reconstructed sequences is concerned, we can observe that the GPC performs better than the k-NN, both in terms performance and consistently. For brevity, we present both results in our analysis. We can see in Table A.3, that the Multi-GPDS reconstructed trajectories enjoy generally higher classification accuracy, which indicates that more precise reconstruction of the missing data can also translate into classification performance gains.

	59%	**64%**	**68%**	**71%**	**74%**
Interp.	0.703(0.060)	0.845(0.077)	1.004(0.090)	1.182(0.119)	2.759(0.368)
SVT	0.616(0.009)	0.672(0.009)	0.730(0.012)	0.778(0.011)	0.860(0.012)
FPC	0.608(0.009)	0.663(0.009)	0.722(0.012)	0.771(0.011)	0.852(0.012)
SS-GP	0.616(0.008)	0.678(0.009)	0.749(0.012)	0.808(0.012)	0.900(0.016)
IBP-GP	0.587(0.007)	0.644(0.008)	0.715(0.012)	0.785(0.013)	0.883(0.014)
Multi-GP	0.564(0.008)	0.604(0.008)	0.650(0.010)	0.692(0.014)	0.788(0.017)
VBMC	0.568(0.006)	0.607(0.007)	0.652(0.008)	0.696(0.011)	0.779(0.013)
Multi-GPDS	**0.541(0.007)**	**0.572(0.010)**	**0.610(0.0152)**	**0.644(0.012)**	**0.737(0.013)**

Table A.2: **Spoken Arabic Digits**: Reconstruction root mean square error for each evaluated method.

	3-NN Classifier (baseline: 10%)				
	59%	**64%**	**68%**	**71%**	**74%**
None	+15.7(3.3)	+18.6(3.9)	+21.2(3.7)	+23.9(4.2)	+28.3(5.0)
Interp.	+8.04(3.8)	+11.3(3.3)	+15.9(4.2)	+21.9(4.7)	+39.2(4.7)
SVT	+7.39(2.7)	+8.39(3.3)	+9.9(3.6)	+12.6(4.0)	+17.9(4.0)
FPC	+7.33(2.7)	+8.34(3.4)	+10.1(3.6)	+12.5(3.6)	+17.5(4.3)
SS-GP	+5.91(2.6)	+8.71(3.7)	+11.2(3.3)	+14.9(3.8)	+23.9(4.5)
IBP-GP	+5.72(2.5)	+7.39(3.0)	+10.3(3.8)	+14.5(4.3)	+23.0(4.2)
Multi-GP	+5.51(2.4)	**+6.66(3.2)**	+8.22(3.4)	+9.22(3.3)	+14.5(4.3)
VBMC	+5.44(2.6)	+6.85(3.4)	+8.21(3.3)	+9.94(2.8)	+15.1(4.0)
Multi-GPDS	**+5.17(2.7)**	+6.98(2.7)	**+7.47(3.4)**	**+9.23(3.1)**	**+13.7(4.0)**
	GP Classifier (baseline: 8.75%)				
	59%	**64%**	**68%**	**71%**	**74%**
None	-	-	-	-	-
Interp.	+4.66(3.0)	+6.17(2.9)	+9.54(3.1)	+12.9(3.7)	+27.8(4.7)
SVT	**+2.96(1.9)**	+3.61(2.6)	+5.64(2.7)	+6.72(2.9)	+10.8(3.4)
FPC	+3.03(2.0)	+3.64(2.4)	+5.35(2.6)	+6.76(2.9)	+10.1(3.1)
SS-GP	+3.42(2.5)	+4.95(3.0)	+7.70(2.9)	+11.0(3.7)	+17.6(4.0)
IBP-GP	+3.44(2.3)	+4.22(2.7)	+6.58(3.3)	+10.2(3.3)	+17.1(4.1)
Multi-GP	+3.11(2.4)	+3.84(2.6)	+5.03(2.8)	+5.98(2.6)	+10.0(3.7)
VBMC	+3.35(2.0)	+3.60(2.2)	+4.70(2.6)	+6.40(2.4)	+9.80(3.6)
Multi-GPDS	+3.21(2.0)	**+3.26(2.3)**	**+4.39(2.5)**	**+5.49(2.5)**	**+9.06(3.0)**

Table A.3: **Spoken Arabic Digits**: Misclassification error achieved for reconstructed data with each evaluated method and under different missing data ratios. We present the results as the excess error above the baseline classification achieved for fully observed data.

A.2.2 *Robotics Dataset*

The setting of this experiment is identical to the one shown in Section 6.3.2 of this this thesis. As we have mentioned the main difference is that we have followed a different data removal scheme.

	40%	**44%**	**49%**	**53%**
Interp.	4.31(1.10)	5.68(1.96)	7.29(3.33)	9.93(5.28)
SVT	4.60(0.76)	4.99(0.91)	5.99(1.55)	6.39(1.63)
FPC	4.51(0.76)	4.90(0.92)	5.92(1.56)	6.31(1.64)
ssGP	4.87(2.36)	5.33(2.63)	4.94(1.49)	5.82(3.41)
ibpGP	3.80(0.74)	4.09(0.56)	4.39(0.52)	4.47(0.55)
mGP	2.84(0.43)	3.01(0.35)	3.46(0.39)	3.68(0.43)
VBMC	3.63(0.51)	3.81(0.45)	4.50(0.83)	4.66(1.23)
mGPDS	**2.80(0.44)**	**2.98(0.32)**	**3.30(0.42)**	**3.49(0.34)**
	56%	**59%**	**62%**	**64%**
Interp.	10.7(3.8)	13.1(5.8)	13.3(4.2)	15.7(3.9)
SVT	6.90(1.74)	7.49(1.93)	8.80(2.30)	9.80(3.06)
FPC	6.86(1.77)	7.51(2.00)	8.80(2.33)	9.84(3.09)
ssGP	4.75(0.36)	5.07(0.70)	5.42(1.82)	5.83(2.07)
ibpGP	4.53(0.54)	4.82(0.48)	5.08(0.64)	5.47(1.43)
mGP	3.87(0.36)	4.07(0.41)	4.46(0.64)	4.81(0.84)
VBMC	4.94(0.99)	5.36(1.20)	6.04(1.98)	6.47(2.72)
mGPDS	**3.64(0.32)**	**3.81(0.46)**	**4.16(0.45)**	**4.43(0.53)**

Table A.4: **Drawing *lazy figure* 8s**: Reconstruction root mean square error for all evaluated methods. Each column represents results for a different missing data ratio. The scale of the results is 10^{-2}.

A.2.3 *Dance Primitives Dataset*

The setting of this experiment is identical to the one shown in Section 6.3.3.1 of this this thesis. Again, here we have followed a different data removal scheme.

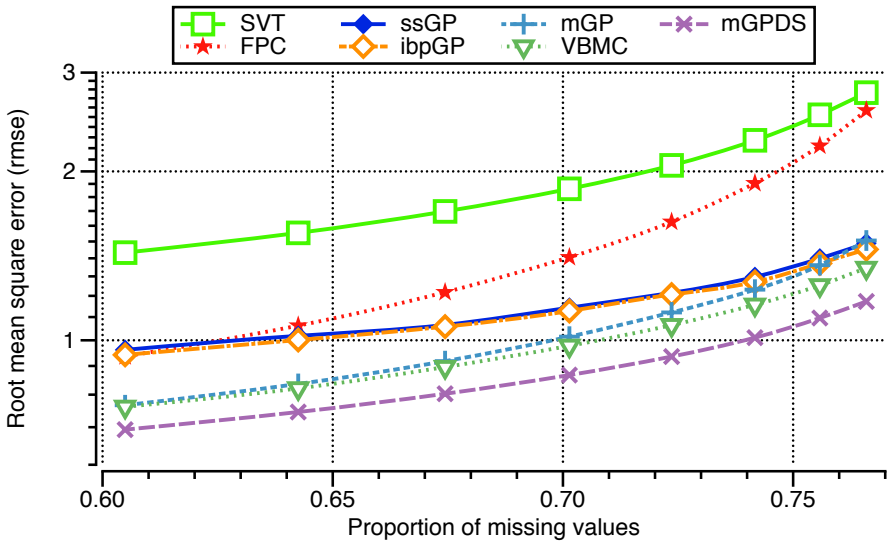


Figure A.4: **Dance primitives**: Root mean square errors for all evaluated methods.

	60%	**64%**	**67%**	**70%**
STL	1.270(0.069)	1.395(0.075)	1.555(0.078)	1.664(0.073)
MTL	**0.837(0.030)**	**0.911(0.050)**	**1.008(0.052)**	**1.098(0.069)**
	72%	**74%**	**76%**	**77%**
STL	1.824(0.098)	1.941(0.103)	2.064(0.107)	2.161(0.131)
MTL	**1.192(0.059)**	**1.296(0.078)**	**1.403(0.078)**	**1.491(0.062)**

Table A.5: **Dance primitives**: Reconstruction RMSE for action LHR (fig. 6.3a) achieved by the Multi-GPDS when trained jointly with all available tasks and separately with data from task LHR only.

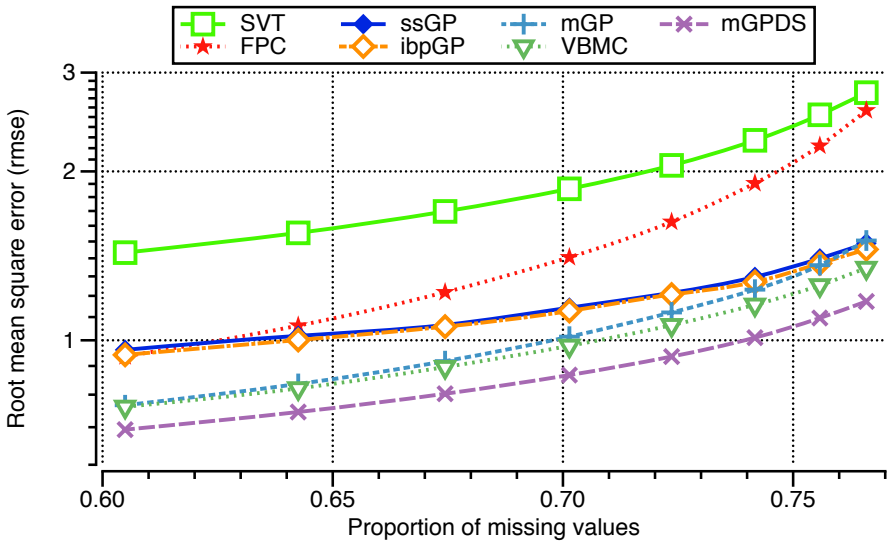


Figure A.5: **Dance primitives**: Root mean square errors for all evaluated methods.

	60%	**64%**	**67%**	**70%**
Interp.	3.35(0.29)	4.58(1.19)	6.05(2.04)	9.47(3.55)
SVT	1.433(0.015)	1.554(0.018)	1.697(0.023)	1.862(0.026)
FPC	0.936(0.014)	1.062(0.019)	1.218(0.025)	1.406(0.029)
SS-GP	0.962(0.030)	1.018(0.029)	1.065(0.024)	1.143(0.033)
IBP-GP	0.942(0.029)	1.001(0.041)	1.058(0.039)	1.128(0.036)
Multi-GP	0.766(0.008)	0.836(0.010)	0.918(0.012)	1.015(0.016)
VBMC	0.761(0.008)	0.821(0.010)	0.896(0.010)	0.978(0.013)
Multi-GPDS	**0.693(0.007)**	**0.745(0.010)**	**0.803(0.012)**	**0.867(0.013)**
	72%	**74%**	**76%**	**77%**
Interp.	11.35(4.29)	14.15(4.27)	17.98(4.45)	21.31(5.05)
SVT	2.050(0.041)	2.272(0.039)	2.523(0.060)	2.763(0.084)
FPC	1.626(0.065)	1.905(0.055)	2.222(0.075)	2.569(0.153)
SS-GP	1.215(0.025)	1.297(0.028)	1.399(0.028)	1.489(0.025)
IBP-GP	1.206(0.031)	1.270(0.028)	1.371(0.042)	1.452(0.031)
Multi-GP	1.121(0.018)	1.231(0.027)	1.361(0.032)	1.506(0.051)
VBMC	1.064(0.014)	1.157(0.015)	1.253(0.019)	1.346(0.020)
Multi-GPDS	**0.935(0.015)**	**1.011(0.018)**	**1.096(0.021)**	**1.173(0.025)**

Table A.6: **Dance primitives**: Reconstruction root mean square error for all evaluated methods. Each column represents results for a different missing data ratio.

B

TENSOR DESOMPOSITION

In this Appendix we briefly present some theoretical background on tensor algebra and decomposition. We also include a brief overview of the relevant methods that appear in the literature.

B.1 TENSOR ALGEBRA

Following a similar notation to [Rogers et al., 2013] let us denote as $\mathbb{N}$ and $\mathbb{R}$ the sets of natural and real numbers respectively. Let us also suppose that $\mathcal{I} \in \mathbb{N}^M$ denotes an index vector space such that $\mathcal{I} \equiv [i_1, \ldots, i_M]^T : i_k \in \mathcal{I}_k \subseteq \mathbb{N},\ \forall k \in [1, M]$. A tensor space on the index space we have defined shall be denoted as $\mathcal{Y} \in \mathbb{R}^{\mathcal{I}}$ or with respect to the individual index spaces as $\mathcal{Y} \in \mathbb{R}^{\mathcal{I}_1 \times \cdots \times \mathcal{I}_M}$, also referred to as an M-way array.

We shall refer to a single element of tensor $\mathcal{Y}$ using a set of indices as $y_{i_1 i_2 \cdots i_M} \in \mathbb{R}$. Sub-tensors constructed by fixing the index of the k^{th} dimension will be refereed to as $\mathcal{Y}_{i_k} \in \mathbb{R}^{\mathcal{I}_1 \times \cdots \times \mathcal{I}_{k-1} \times \mathcal{I}_{k+1} \times \cdots \times \mathcal{I}_M}$. Equivalently, by omitting the index of the k^{th} dimension, we denote a vector containing tensor elements along the corresponding slice $\boldsymbol{y}_{i_1 \cdots i_{k-1} i_{k+1} \cdots i_M} \in \mathbb{R}^{\mathcal{I}_k}$.

Finally, we shall use two main reshaping operators to "vectorise" or "tensorise" equivalent to wrapping and unwrapping the corresponding data structures. More specifically as $\boldsymbol{y} = vect(\mathcal{Y})$ we denote the operation of unwraping tensor $\mathcal{Y} \in \mathbb{R}^{\mathcal{I}_1 \times \cdots \times \mathcal{I}_M}$ into a vector $\boldsymbol{y} \in \mathbb{R}^{\mathcal{I}_1 \mathcal{I}_2 \cdots \mathcal{I}_M}$. "Tensorisation", on the contrary, $\mathcal{Y} = ten(\boldsymbol{y})$ is precisely the inverse of the previous operation so that applying it to vector $\boldsymbol{y} \in \mathbb{R}^{\mathcal{I}_1 \mathcal{I}_2 \cdots \mathcal{I}_M}$ yields the original tensor data structure $\mathcal{Y} \in \mathbb{R}^{\mathcal{I}_1 \times \cdots \times \mathcal{I}_M}$.

B.1.1 *Multilinear Tensor Decompositions*

Tensor decomposition may be regarded as a generalisation of its matrix counterpart. In the most general case, we refer to a multilinear decomposition by denoting it as follows:

$$\mathcal{Y} = \mathcal{A} \circledast \mathcal{F} + \mathcal{Q} \tag{B.1}$$

where $\mathcal{Y}$ is the observations tensor, $\mathcal{A}$ a loading tensor, $\mathcal{F}$ is a core tensor, the multilinear equivalent to a factor matrix and $\mathcal{Q}$ is a noise tensor. As $\circledast$ we denote the generalised tensor multiplication operator with the following effect:

$$y_{i_1 \cdots i_M} = \sum_{j_1=1}^{J_1} \cdots \sum_{j_M=1}^{J_M} a_{i_1 \cdots i_M, j_1 \cdots j_M} f_{j_1 \cdots j_M} + \epsilon_{i_1 \cdots i_M} \tag{B.2}$$

In practice we would typically resort to some necessary simplifications to the general case. For instance by assuming homoscedastic noise and a loading tensor $\mathcal{A}$ decomposable as $\mathcal{A} = \boldsymbol{A}^{(1)} \times \boldsymbol{A}^{(2)} \times \cdots \times \boldsymbol{A}^{(M)}$ eq. (B.2) yields the following formulation:

$$y_{i_1 \cdots i_M} = \sum_{j_1=1}^{J_1} \cdots \sum_{j_M=1}^{J_M} a^{(1)}_{i_1 j_1} \cdots a^{(M)}_{i_M j_M} f_{j_1 \cdots j_M} + \epsilon \tag{B.3}$$

One of the first methods proposed was the Canonical Decomposition (CD) of an M-way tensor $\mathcal{Y}$ defined as follows:

$$y_{i_1 \cdots i_M} = \sum_{j=1}^{J} a^{(1)}_{i_1 j} a^{(2)}_{i_2 j} \cdots a^{(M)}_{i_M j} + \epsilon \tag{B.4}$$

It constitutes an analysis into individual $rank-1$ components or factors. This reconstruction is suitable for expressing tensors of rank up to J.

A generalisation to the CD was proposed in [Tucker, 1966] and is also considered as the generalisation of factor analysis in tensor spaces. A full-rank Tucker decomposition can be defined as in eq. (B.3).

As we may observe this formulation involves M factor loading matrices and a single core tensor factor $\mathcal{F} \in \mathbb{R}^{J_1 \times \cdots \times J_M}$ of typically much lower dimensionality compared to $\mathcal{Y}$. The main incentive is that the factor tensor $\mathcal{F}$ acts as a compact latent representation of the original tensor space.

Note that vectorising eq. (B.3) yields the following equivalent representation:

$$vect(\mathcal{Y}) = \boldsymbol{A} vect(\mathcal{F}) + \boldsymbol{\epsilon} \tag{B.5}$$

where $\boldsymbol{A} = \boldsymbol{A}^{(1)} \otimes \boldsymbol{A}^{(2)} \cdots \boldsymbol{A}^{(M)}$ and as $\otimes$ we denote the Kronecker product.

The full Tucker decomposition is also referred to as $Tucker_M$, to indicate that we decompose our original tensor into $M-way$ factors. In that sense the CD may also also be referred to as $Tucker_1$ and is a special case of the Tucker decomposition by setting the core factor tensor to a super-diagonal identity matrix. Along the same lines, using a super-diagonal core tensor gives rise to the following decomposition to which we shall refer later in this chapter:

$$y_{i_1 \cdots i_M} = \sum_{j=1}^{J} r_j a_{i_1 j}^{(1)} a_{i_2 j}^{(2)} \cdots a_{i_M j}^{(M)} + \epsilon \tag{B.6}$$

The above formulation conveniently allows us to control the rank of our tensor decomposition by imposing prior with component shrinkage on $\boldsymbol{r}$, such as an ARD, IBP or BP priors.

Nevertheless, depending on the structure of the data we wish to model it may be more appropriate to use one of the intermediate decompositions lying between between $Tucker_1$ and $Tucker_M$. For instance if we consider tensor time-series it

may be beneficial to choose a time-varying core factor tensor and a $Tucker_{(M-1)}$ decomposition as follows:

$$y^{(t)}_{i_1 \cdots i_{M-1}} = \sum_{j_1=1}^{J_1} \cdots \sum_{j_{M-1}=1}^{J_{M-1}} a^{(1)}_{i_1 j_1} \cdots a^{(M-1)}_{i_{M-1} j_{M-1}} f^{(t)}_{j_1 \cdots j_{M-1}} \tag{B.7}$$

B.1.2 *Related Approaches*

Various tensor-based models have recently emerged in the literature addressing issues such as automatically selecting the optimal tensor rank or dealing with temporal data. They all seem to agree however that manipulating the data in tensor form is beneficial to aggregating across dimensions and using matrix decomposition algorithms; which may be attributed to the fact that we respect the innate structure of the data.

In tensor decomposition in particular, estimating the rank in a data driven manner becomes even more important than in matrix decomposition problems. That is due to the fact that finding the tensor rank is an ill-posed problem [De Silva & Lim, 2008]. For an $[N \times M]$ matrix, its rank is $r \leq \min(N, M)$, whereas in the case of a tensor $[J_1 \times J_2 \times \cdots \times J_M]$ the rank is always $r > \min(J_1, J_2, \ldots, J_M)$, which does not help set an upper rank bound in our algorithms.

Similar to the matrix factorisation literature, tensor factorisation approaches can be categorised according to the parameter estimation method utilised. More specifically whether parameter estimation is conducted through optimisation [Cichocki et al., 2014; London et al., 2013] or inference [Xiong et al., 2010; Rai et al.]. There have been numerous optimisation-based methods that explore almost the full scale of possible tensor decompositions. Especially interesting and thorough is the analysis in [Phan & Cichocki, 2010], where the authors explore decompositions that also facilitate multidimensional data classification.

Again, as in the matrix decomposition case, to which we have referred in Chapter 6, authors have reported that a probabilistic [Mnih & Salakhutdinov, 2007] and

full Bayesian [Mnih & Salakhutdinov, 2007] treatment is preferable to point-wise optimisation [Zhao et al., 2014; Xiong et al., 2010; Rai et al.].

Extending the concept of Bayesian matrix factorisation the authors in [Tao et al., 2008] present a Bayesian tensor decomposition based on the CD. Subsequent publications address the issues of automatically inferring the optimal tensor rank by decomposing as in eq. (B.6) and utilising either ARD eq. (B.9) [Zhao et al., 2014] or a Multiplicative Gamma Process (MGP) prior eq. (B.10) [Rai et al.] respectively. Similarly to using the MGP prior, one could use other non-parametric priors for automatic rank estimation, such as the Indian Buffet Process (IBP) or the Beta Process (BP).

$$y_{i_1 \cdots i_M} = \sum_{j=1}^{J} r_j a_{i_1 j}^{(1)} a_{i_2 j}^{(2)} \cdots a_{i_M j}^{(M)} \tag{B.8}$$

$$r_j \sim \mathcal{N}\left(0, \alpha_j^{-1}\right), \quad \alpha_j \sim \mathcal{G}\left(a, \frac{1}{b}\right) \tag{B.9}$$

$$r_j \sim \mathcal{N}\left(0, \alpha_j^{-1}\right), \; \alpha_j = \prod_{k=1}^{j} \xi_k, \; \xi_k \sim \mathcal{G}\left(a, \frac{1}{b}\right) \tag{B.10}$$

Although, rank estimation is valuable for finding sparse representations of the data, we have observed that they lack in performance compared to approaches which do not employ model reduction, even when extending those approached for temporal data.

Bayesian tensors for time-series data are first introduced in [Xiong et al., 2010] by placing an autoregressive prior AR_1 of a special form on the temporal loadings of the Canonical Decomposition (CD). (this special form AR_1 can be retrieved from eq. (2.33) by setting $m = 1$ and $\boldsymbol{B}_1 = \boldsymbol{I}$).

The concept is then extended for the $Tucker_M$ decomposition in [Rogers et al., 2013]. In this case the authors in have considered a general decomposition, more specifically a $Tucker_{(M-1)}$ decomposition for temporal data and have modelled the output tensor $\mathcal{Y}_t$ as a multi-way Linear Dynamical Systems (LDS) as follows:

$$\mathcal{Y}_t = \mathcal{C} \circledast \mathcal{Z}_t + \mathcal{Q} \tag{B.11}$$

$$\mathcal{Z}_t = \mathcal{A} \circledast \mathcal{Z}_{t-1} + \mathcal{R} \tag{B.12}$$

Now, let us take a closer look at the aforementioned formulation. From the viewpoint of $\mathcal{Y}_t$, the construction constitutes a general multilinear dynamical system as mentioned in [Rogers et al., 2013]. However it could also be viewed as a $Tucker_{(M-1)}$ decomposition, where we have set an AR_1 prior on the temporal core factor tensor. Note also that it is a probabilistic, but not a Bayesian method as it employs Kalman EM to estimate $\mathcal{Z}_t$, $\mathcal{Q}$ and $\mathcal{R}$, along with gradient ascent MLE estimation of $\mathcal{A}$ and $\mathcal{C}$. Finally, note that in eq. (B.11) we make the following transition $\mathbb{R}^{\mathcal{I}} \rightarrow \mathbb{R}^{\mathcal{J}}$, through $\mathcal{C}$, which allows for (substantial) dimensionality reduction. This, however is not the case, in the hierarchically second layer of eq. (B.12), where we transfer within a space of the same dimensionality $\mathbb{R}^{\mathcal{J}} \rightarrow \mathbb{R}^{\mathcal{J}}$.

C

BCD-LDS RESULTS

In this Section we present some preliminary results pertaining to our ongoing work on tensor completion, outlined in Section 7.3.

C.1 PRELIMINARY EVALUATION

For the purpose of empirically evaluating the proposed approach we employ the exact same setting used in Appendix A. More specifically, in addition to Section A, we introduce comparisons with the following methods:

- **PCD-AR1**: Probabilistic Canonical Decomposition (CD) with AR_1 prior on the time-dependent loadings. This practically corresponds to the method proposed in [Xiong et al., 2010] where we employ spherical Gaussian priors, instead of full Gaussians for all continuous random variables. This approach is probabilistic, but not fully Bayesian.
- **BCD-AR1**: Bayesian Canonical Decomposition (CD) with AR_1 prior on the time-dependent loadings. This corresponds exactly to the method proposed in [Xiong et al., 2010].

- **BCD-LDS**: Bayesian Canonical Decomposition (CD) with LDS prior on the time-dependent loadings. This is the method proposed in this work.

In order to achieve impartial comparability among the evaluated methods, we have run all new algorithms with the same number of factors as the Multi-GPDS, namely 64. We have generated 2000 Gibbs samples from the posteriors of the sampling-based methods and for our results we have considered the 500 last samples. Finally we have repeated model training for 20 random initialisations to accumulate statistics.

Our preliminary results are presented in Table C.1. By inspecting them, we can observe that the BCD-LDS appears to be the most effective in reconstructing the 4 most severely damaged sequences, closely followed by the Multi-GPDS. In moderately damaged sequences the main methods we consider, namely the Multi-GPDS, PCD-AR1, BCD-AR1 and BCD-LSD are statistically equivalent. And finally the Multi-GPDS achieves the best reconstruction for the 2 least damaged sequences.

	60%	**64%**	**67%**	**70%**
Interp.	3.35(0.29)	4.58(1.19)	6.05(2.04)	9.47(3.55)
SVT	1.433(0.015)	1.554(0.018)	1.697(0.023)	1.862(0.026)
FPC	0.936(0.014)	1.062(0.019)	1.218(0.025)	1.406(0.029)
SS-GP	0.962(0.030)	1.018(0.029)	1.065(0.024)	1.143(0.033)
IBP-GP	0.942(0.029)	1.001(0.041)	1.058(0.039)	1.128(0.036)
Multi-GP	0.766(0.008)	0.836(0.010)	0.918(0.012)	1.015(0.016)
VBMC	0.761(0.008)	0.821(0.010)	0.896(0.010)	0.978(0.013)
Multi-GPDS	**0.693(0.007)**	**0.745(0.010)**	**0.803(0.012)**	**0.867(0.013)**
PCD-AR1	0.710(0.008)	0.752(0.008)	**0.804(0.012)**	**0.869(0.015)**
BCD-AR1	0.713(0.008)	0.752(0.009)	**0.803(0.010)**	**0.865(0.013)**
BCD-LDS	0.752(0.008)	0.781(0.007)	0.822(0.010)	**0.866(0.011)**

	72%	**74%**	**76%**	**77%**
Interp.	11.35(4.29)	14.15(4.27)	17.98(4.45)	21.31(5.05)
SVT	2.050(0.041)	2.272(0.039)	2.523(0.060)	2.763(0.084)
FPC	1.626(0.065)	1.905(0.055)	2.222(0.075)	2.569(0.153)
SS-GP	1.215(0.025)	1.297(0.028)	1.399(0.028)	1.489(0.025)
IBP-GP	1.206(0.031)	1.270(0.028)	1.371(0.042)	1.452(0.031)
Multi-GP	1.121(0.018)	1.231(0.027)	1.361(0.032)	1.506(0.051)
VBMC	1.064(0.014)	1.157(0.015)	1.253(0.019)	1.346(0.020)
MultiGPDS	0.935(0.015)	1.011(0.018)	1.096(0.021)	1.173(0.025)
PCD-AR1	0.945(0.016)	1.032(0.019)	1.159(0.030)	1.294(0.043)
BCD-AR1	0.934(0.016)	1.017(0.017)	1.115(0.023)	1.231(0.027)
BCD-LDS	**0.924(0.013)**	**0.988(0.012)**	**1.057(0.016)**	**1.137(0.019)**

Table C.1: **Dance primitives**: Reconstruction root mean square error for all evaluated methods. Each column represents results for a different missing data ratio.

D

BCD-LDS GIBBS SAMPLER

In this section we present the conditional posteriors necessary for implementing the Gibbs sampler and performing posterior inference for the proposed approach, part of our ongoing work, outlined in Section 7.3.

The derivation are more or less standard, but do not necessarily appear elsewhere in the literature. We present out final results here for brevity. Thus readers not interested in implementing the sampler may safely skip this Appendix.

D.1 GIBBS SAMPLER

D.1.1 *Sampling Loadings* $\boldsymbol{w}_n$

$$\boldsymbol{w}_n \sim \mathcal{N}\left(\boldsymbol{w}_n | \tilde{\boldsymbol{\mu}}_{w_n}, \tilde{\boldsymbol{\Lambda}}_{w_n}^{-1}\right) \tag{D.1}$$

$$\tilde{\boldsymbol{\Lambda}}_{w_n} = \boldsymbol{\Lambda}_w + \boldsymbol{V}_n \tag{D.2}$$

$$\tilde{\boldsymbol{\mu}}_{w_n} = \tilde{\boldsymbol{\Lambda}}_{w_n}^{-1}\left(\boldsymbol{\Lambda}_w \boldsymbol{\mu}_w + \boldsymbol{b}_n\right) \tag{D.3}$$

$$\boldsymbol{V}_n = \sum_{(t,i)\in\mathcal{O}_n} \beta_n \left(\boldsymbol{z}_t \circ \boldsymbol{g}_i\right)\left(\boldsymbol{z}_t \circ \boldsymbol{g}_i\right)^T \tag{D.4}$$

$$\boldsymbol{b}_n = \sum_{(t,i)\in\mathcal{O}_n} \beta_n y_{nti} \left(\boldsymbol{z}_t \circ \boldsymbol{g}_i\right) \tag{D.5}$$

D.1.2 *Sampling Time-dependant Factors* $\boldsymbol{z}_t$

$$\begin{aligned}
\boldsymbol{z}_t &\sim \mathcal{N}\left(\boldsymbol{z}_t|\tilde{\boldsymbol{\mu}}_{z_t}, \tilde{\boldsymbol{\Lambda}}_{z_t}^{-1}\right) && \text{(D.6)}\\
\tilde{\boldsymbol{\Lambda}}_{z_t} &= \boldsymbol{\Lambda}_z + \boldsymbol{V}_t && \text{(D.7)}\\
\tilde{\boldsymbol{\mu}}_{z_t} &= \tilde{\boldsymbol{\Lambda}}_{z_t}^{-1}\left(\boldsymbol{\Lambda}_z \boldsymbol{C}\boldsymbol{x}_t + \boldsymbol{b}_t\right) && \text{(D.8)}\\
\boldsymbol{V}_t &= \sum_{(n,i)\in\mathcal{O}_t} \beta_n \left(\boldsymbol{g}_i \circ \boldsymbol{w}_n\right)\left(\boldsymbol{g}_i \circ \boldsymbol{w}_n\right)^T && \text{(D.9)}\\
\boldsymbol{b}_t &= \sum_{(n,i)\in\mathcal{O}_t} \beta_n y_{nti} \left(\boldsymbol{g}_i \circ \boldsymbol{w}_n\right) && \text{(D.10)}
\end{aligned}$$

D.1.3 *Sampling State-space Variable* $\boldsymbol{x}_t$

$$\begin{aligned}
\boldsymbol{x}_t &\sim \mathcal{N}\left(\boldsymbol{x}_t|\tilde{\boldsymbol{\mu}}_{x_t}, \tilde{\boldsymbol{\Lambda}}_{x_t}^{-1}\right) && \text{(D.11)}\\
\tilde{\boldsymbol{\Lambda}}_{x_t} &= \boldsymbol{\Lambda}_x + \boldsymbol{V}_t && \text{(D.12)}\\
\tilde{\boldsymbol{\mu}}_{x_t} &= \tilde{\boldsymbol{\Lambda}}_{x_t}^{-1}\boldsymbol{b}_t && \text{(D.13)}\\
\boldsymbol{V}_t &= \begin{cases} \boldsymbol{A}^T\boldsymbol{\Lambda}_x\boldsymbol{A} + \boldsymbol{C}^T\boldsymbol{\Lambda}_z\boldsymbol{C} & , \forall t \in [1, N_T) \\ \boldsymbol{C}^T\boldsymbol{\Lambda}_z\boldsymbol{C} & , t = N_T \end{cases} && \text{(D.14)}\\
\boldsymbol{b}_t &= \begin{cases} \boldsymbol{\Lambda}_x\boldsymbol{\mu}_x + \boldsymbol{A}^T\boldsymbol{\Lambda}_x\boldsymbol{x}_2 + \boldsymbol{C}^T\boldsymbol{\Lambda}_z\boldsymbol{z}_1 & , t = 1 \\ \boldsymbol{\Lambda}_x\boldsymbol{A}\boldsymbol{x}_{N_T-1} + \boldsymbol{C}^T\boldsymbol{\Lambda}_z\boldsymbol{z}_{N_T} & , t = N_T \\ \boldsymbol{\Lambda}_x\boldsymbol{A}\boldsymbol{x}_{t-1} + \boldsymbol{A}^T\boldsymbol{\Lambda}_x\boldsymbol{x}_{t+1} + \boldsymbol{C}^T\boldsymbol{\Lambda}_z\boldsymbol{z}_t & , t \in (1, N_T) \end{cases} && \text{(D.15)}
\end{aligned}$$

D.1.4 *Sampling Loadings* $\boldsymbol{g}_i$

$$\begin{aligned}
\boldsymbol{g}_i &\sim \mathcal{N}\left(\boldsymbol{g}_i|\tilde{\boldsymbol{\mu}}_{g_i}, \tilde{\boldsymbol{\Lambda}}_{g_i}^{-1}\right) && \text{(D.16)}\\
\tilde{\boldsymbol{\Lambda}}_{g_i} &= \boldsymbol{\Lambda}_g + \boldsymbol{V}_i && \text{(D.17)}\\
\tilde{\boldsymbol{\mu}}_{g_i} &= \tilde{\boldsymbol{\Lambda}}_{g_i}^{-1}\left(\boldsymbol{\Lambda}_g\boldsymbol{\mu}_g + \boldsymbol{b}_i\right) && \text{(D.18)}\\
\boldsymbol{V}_i &= \sum_{(n,t)\in\mathcal{O}_i} \beta_n \left(\boldsymbol{w}_n \circ \boldsymbol{z}_t\right)\left(\boldsymbol{w}_n \circ \boldsymbol{z}_t\right)^T && \text{(D.19)}\\
\boldsymbol{b}_i &= \sum_{(n,t)\in\mathcal{O}_i} \beta_n y_{nti}\left(\boldsymbol{w}_n \circ \boldsymbol{z}_t\right) && \text{(D.20)}
\end{aligned}$$

D.1.5 *Sampling Parameters* $\boldsymbol{\mu}_w, \boldsymbol{\Lambda}_w$ *and* $\boldsymbol{\mu}_g, \boldsymbol{\Lambda}_g$

Same as the ones in [Xiong et al., 2010].

D.1.6 *Sampling Parameters* $\boldsymbol{C}, \boldsymbol{\Lambda}_z$

$$\begin{aligned}
\boldsymbol{C}, \boldsymbol{\Lambda}_z &\sim \mathcal{MN}\left(\boldsymbol{C}|\tilde{\boldsymbol{M}}_C, \boldsymbol{\Lambda}_z^{-1}, \tilde{\boldsymbol{K}}_C\right)\mathcal{W}\left(\boldsymbol{\Lambda}_z|\tilde{\boldsymbol{W}}_0, \tilde{\nu}_0\right) && \text{(D.21)}\\
\tilde{\boldsymbol{K}}_C &= \boldsymbol{K}_C + \sum_{t=1}^{N_T} \boldsymbol{x}_t\boldsymbol{x}_t^T && \text{(D.22)}\\
\tilde{\boldsymbol{M}}_C &= \left(\boldsymbol{M}_C\boldsymbol{K}_C + \sum_{t=1}^{N_T} \boldsymbol{z}_t\boldsymbol{x}_t\right)\tilde{\boldsymbol{K}}_C^{-1} && \text{(D.23)}\\
\tilde{\nu}_0 &= \nu_0 + N_T && \text{(D.24)}\\
\tilde{\boldsymbol{W}}_0^{-1} &= \boldsymbol{W}_0^{-1} + \boldsymbol{M}_C^T\boldsymbol{K}_C\boldsymbol{M}_C - \tilde{\boldsymbol{M}}_C^T\tilde{\boldsymbol{K}}_C\tilde{\boldsymbol{M}}_C + \sum_{t=1}^{N_T} \boldsymbol{z}_t\boldsymbol{z}_t^T && \text{(D.25)}
\end{aligned}$$

D.1.7 *Sampling Parameters* $\boldsymbol{A}, \boldsymbol{\mu}_x, \boldsymbol{\Lambda}_x$

$$\boldsymbol{A}, \boldsymbol{\mu}_x, \boldsymbol{\Lambda}_x \sim \mathcal{MN}\left(\boldsymbol{C}|\tilde{\boldsymbol{M}}_C, \boldsymbol{\Lambda}_z^{-1}, \tilde{\boldsymbol{K}}_C\right) \mathcal{N}\left(\boldsymbol{\mu}_x|\tilde{\boldsymbol{\rho}}_0, \left(\tilde{\beta}_0 \boldsymbol{\Lambda}_x\right)^{-1}\right) \tag{D.26}$$

$$\mathcal{W}\left(\boldsymbol{\Lambda}_z|\tilde{\boldsymbol{W}}_0, \tilde{\nu}_0\right) \tag{D.27}$$

$$\tilde{\boldsymbol{K}}_A = \boldsymbol{K}_A + \sum_{t=2}^{N_T} \boldsymbol{x}_{t-1}\boldsymbol{x}_{t-1}^T \tag{D.28}$$

$$\tilde{\boldsymbol{M}}_A = \left(\boldsymbol{M}_A \boldsymbol{K}_A + \sum_{t=2}^{N_T} \boldsymbol{x}_t \boldsymbol{x}_{t-1}^T\right) \tilde{\boldsymbol{K}}_A^{-1} \tag{D.29}$$

$$\tilde{\beta}_0 = \beta_0 + 1 \tag{D.30}$$

$$\tilde{\boldsymbol{\rho}}_0 = \frac{\beta_0 \boldsymbol{\rho}_0 + \boldsymbol{x}_1}{\tilde{\beta}_0} \tag{D.31}$$

$$\tilde{\nu}_0 = \nu_0 + N_T \tag{D.32}$$

$$\tilde{\boldsymbol{W}}_0^{-1} = \boldsymbol{W}_0^{-1} + \boldsymbol{M}_A^T \boldsymbol{K}_A \boldsymbol{M}_A - \tilde{\boldsymbol{M}}_A^T \tilde{\boldsymbol{K}}_A \tilde{\boldsymbol{M}}_A + \sum_{t=1}^{N_T} \boldsymbol{x}_t \boldsymbol{x}_t^T \tag{D.33}$$

$$+\beta_0 \boldsymbol{\rho}_0 \boldsymbol{\rho}_0^T - \tilde{\beta}_0 \tilde{\boldsymbol{\rho}}_0 \tilde{\boldsymbol{\rho}}_0^T \tag{D.34}$$

BIBLIOGRAPHY

A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn. Imitation with ALICE: learning to imitate corresponding actions across dissimilar embodiments. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 32(4):482–496, 2002. (Cited on page 74.)

A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn. Correspondence mapping induced state and action metrics for robotic imitation. *IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics*, 37(2):299–307, 2007. (Cited on page 74.)

M. Alvarez, J. R. Peters, N. D. Lawrence, and B. Schölkopf. Switched latent force models for movement segmentation. In *Advances in neural information processing systems*, pages 55–63, 2010. (Cited on page 78.)

M. A. Alvarez, D. Luengo, and N. D. Lawrence. Latent force models. In *International Conference on Artificial Intelligence and Statistics*, pages 9–16, 2009a. (Cited on page 78.)

M. A. Alvarez, D. Luengo, and N. D. Lawrence. Latent force models. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 9–16, 2009b. (Cited on page 135.)

S.-I. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10:251–276, 1998. (Cited on page 115.)

C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1-2):5–43, 2003. (Cited on page 44.)

C. E. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174, 1974. (Cited on pages 57 and 86.)

B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009. (Cited on page 80.)

D. S. Babacan, M. Luessi, R. Molina, and A. K. Katsaggelos. Sparse bayesian methods for low-rank matrix estimation. *IEEE Transactions on Signal Processing*, 60(8):3964–3977, 2012. (Cited on pages 134, 136, 138, 142, and 149.)

M. J. Beal, Z. Ghahramani, and C. E. Rasmussen. The infinite hidden markov model. In *Advances in Neural Information Processing Systems (NIPS)*, pages 577–584, 2001. (Cited on page 60.)

R. E. Bellman. *Adaptive control processes: A guided tour*. Princeton University Press (Princeton, N.J), 1961. (Cited on page 27.)

A. Bhattacharya and D. B. Dunson. Sparse bayesian infinite factor models. *Biometrika*, 98(2):291–306, 2011. (Cited on page 65.)

A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. *Springer Handbook of Robotics*, pages 1371–1394, 2008. (Cited on pages 74, 75, and 80.)

A. G. Billard, S. Calinon, and F. Guenter. Discriminative and adaptive imitation in uni-manual and bi-manual tasks. *Robotics and Autonomous Systems*, 54(5): 370–384, 2006. (Cited on page 80.)

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 1st ed. 2006. corr. 2nd printing edition, October 2007. ISBN 0387310738. (Cited on pages 44, 46, 53, 76, and 77.)

D. M. Blei and M. I. Jordan. Variational inference for dirichlet process mixtures. *Bayesian analysis*, 1(1):121–143, 2006. (Cited on pages 86 and 87.)

L. Bottou. *Online learning and stochastic approximations.* Cambridge University Press, Cambridge, UK, 1998. (Cited on pages 113 and 115.)

L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Internationl Conference on Computational Statistics (COMPSTAT)*, 2010. (Cited on page 113.)

M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (ICCV)*, pages 994–999. IEEE, 1997. (Cited on page 60.)

S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. *Handbook of Markov Chain Monte Carlo*. CRC Press, 2011. (Cited on page 134.)

J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010. (Cited on pages 136 and 148.)

S. Calinon. *Continuous Extraction of Task Constraints in a Robot Programming by Demonstration Framework*. PhD thesis, Ecole Polytechnique Federale de Lausanne, May 2007. (Cited on pages 28 and 75.)

S. Calinon. Robot programming by demonstration. In *Springer handbook of robotics*, pages 1371–1394. Springer, 2008. (Cited on page 28.)

S. Calinon and A. Billard. Active teaching in robot programming by demonstration. In *IEEE International Symposium on Robot and Human interactive Communication (ROMAN)*, pages 702–707. IEEE, 2007. (Cited on page 28.)

S. Calinon and A. Billard. Statistical learning by imitation of competing constraints in joint space and task space. *Adv. Robot.*, 23(15):2059–2076, 2009. (Cited on page 75.)

S. Calinon, F. Guenter, and A. Billard. On learning, representing and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 37:286–298, 2007. (Cited on page 75.)

S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard. Learning and reproduction of gestures by imitation: An approach based on hidden Markov model and Gaussian mixture regression. *IEEE Robotics and Automation Magazine*, 17(2):44–54, 2010. (Cited on page 75.)

S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell. Statistical dynamical systems for skills acquisition in humanoids. In *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*, pages 323–329. IEEE, 2012a. (Cited on page 77.)

S. Calinon, A. Pervez, and D. G. Caldwell. Multi-optima exploration with adaptive gaussian mixture model. In *IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pages 1–6. IEEE, 2012b. (Cited on pages 29 and 70.)

S. Calinon, P. Kormushev, and D. G. Caldwell. Compliant skills acquisition and multi-optima policy search with em-based reinforcement learning. *Robotics and Autonomous Systems*, 61(4):369–379, 2013. (Cited on pages 29 and 70.)

S. Calinon, D. Bruno, and D. G. Caldwell. A task-parameterized probabilistic model with minimal intervention control. In *IEEE International Conference on Robotics and Automation (ICRA)*, number EPFL-CONF-198848, 2014. (Cited on page 28.)

E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Trans. on Information Theory*, 56(5):2053–2080, 2010. (Cited on page 134.)

O. Cappé and E. Moulines. On-line expectation-maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613, 2009. (Cited on pages 116 and 118.)

V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009. (Cited on page 41.)

S. Chatzis, D. Korkinof, and Y. Demiris. The one-hidden layer non-parametric bayesian kernel machine. In *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, November 2011. (Cited on page 41.)

S. Chatzis, D. Korkinof, and Y. Demiris. A nonparametric bayesian approach toward robot learning by demonstration. *Robotics and Autonomous Systems*, 60 (6):789 – 802, 2012a. (Cited on pages 37, 81, 87, 108, 109, and 166.)

S. Chatzis, D. Korkinof, and Y. Demiris. A spatially-constrained normalized gamma process prior. *Expert Systems with Applications*, 39(17):13019 – 13025, 2012b. (Cited on pages 52 and 58.)

S. Chatzis, D. Korkinof, and Y. Demiris. A quantum-statistical approach toward robot learning by demonstration. *IEEE Transactions on Robotics*, 28(6):1371–1381, 2012c. (Cited on pages 37, 57, 81, 94, and 166.)

B. Chen, G. Polatkan, G. Sapiro, D. Blei, D. Dunson, and L. Carin. Deep learning with hierarchical convolutional factor analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(8), 2013. (Cited on page 174.)

A. Cichocki, C. Mandic, A. Phan, C. Caifa, G. Zhou, Q. Zhao, and L. De Lathauwer. Tensor decompositions for signal processing applications. from two-way to multiway component analysis. *IEEE Signal Processing Magazine*, (accepted), 2014. (Cited on page 193.)

P. I. Corke. A robotics toolbox for matlab. *IEEE Robotics & Automation Magazine*, 3(1):24–32, 1996. (Cited on pages 18, 121, 123, and 125.)

B. Damas and J. Santos-Victor. An online algorithm for simultaneously learning forward and inverse kinematics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1499–1506. IEEE, 2012. (Cited on page 114.)

A. Damianou and N. Lawrence. Deep gaussian processes. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 207–215, 2013. (Cited on page 136.)

A. C. Damianou, M. K. Titsias, and N. D. Lawrence. Variational gaussian process dynamical systems. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2510–2518, 2011. (Cited on pages 78, 135, 141, 143, 145, and 146.)

J. S. De La Cruz, W. Owen, and D. Kulic. Online learning of inverse dynamics via gaussian process regression. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012. (Cited on page 114.)

V. De Silva and L.-H. Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, 30 (3):1084–1127, 2008. (Cited on page 193.)

Y. Demiris and B. Khadhouri. Hierarchical attentive multiple models for execution and recognition of actions. *Robotics and Autonomous Systems*, 54(5):361–369, May 2006. (Cited on pages 70 and 75.)

A. Doucet, N. De Freitas, and N. Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001a. (Cited on page 44.)

A. Doucet, N. De Freitas, and N. Gordon. *Sequential Monte Carlo methods in practice*. Springer, 2001b. (Cited on page 53.)

A. Droniou, S. Ivaldi, V. Padois, and O. Sigaud. Autonomous online learning of velocity kinematics on the icub: A comparative study. In *IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, 2012. (Cited on page 114.)

D. B. Dunson and J.-H. Park. Kernel stick-breaking processes. *Biometrika*, 95(2): 307–323, 2008. (Cited on page 58.)

D. Duvenaud, J. Lloyd, R. Grosse, J. Tenenbaum, and G. Zoubin. Structure discovery in nonparametric regression through compositional kernel search. In *Internation Conference on Machine Learning (ICML)*, pages 1166–1174, 2013. (Cited on page 135.)

T. S. Ferguson. A bayesian analysis of some nonparametric problems. *The annals of statistics*, pages 209–230, 1973. (Cited on page 84.)

F. Forbes and N. Peyrard. Hidden markov random field model selection criteria based on mean field-like approximations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1089–1101, 2003. (Cited on page 58.)

A. S. Fox, Emily B andWillsky, E. B. Sudderth, and M. I. Jordan. Nonparametric bayesian learning of switching linear dynamical systems. In *Advances in Neural Information Processing Systems (NIPS)*, pages 457–464, 2009. (Cited on pages 60, 62, 73, and 77.)

E. B. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky. An hdp-hmm for systems with state persistence. In *International conference on Machine learning (ICML)*, pages 312–319. ACM, 2008. (Cited on page 60.)

J. V. Gael, Y. W. Teh, and Z. Ghahramani. The infinite factorial hidden markov model. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1697–1704, 2009. (Cited on page 60.)

Z. Ghahramani and M. I. Jordan. Factorial hidden markov models. *Machine learning*, 29(2-3):245–273, 1997. (Cited on page 60.)

Z. Ghahramani and M. I. Jordan. Supervised learning from incomplete data via an EM approach. In *Advances in Neural Information Processing Systems (NIPS)*, volume 6, pages 120–127, 1994. (Cited on pages 75, 80, and 95.)

M. Girolami and B. Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011. (Cited on page 53.)

D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier. Mechatronic design of nao humanoid. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 769–774. IEEE, 2009. (Cited on pages 27 and 96.)

T. Griffiths and Z. Ghahramani. Infinite latent feature models and the indian buffet process. In *Advances in Neural Information Processing Systems (NIPS)*,

volume 18, pages 475–482. Vancouver, Canada, December 2006. (Cited on pages 40, 51, 64, and 148.)

T. L. Griffiths and Z. Ghahramani. The indian buffet process: An introduction and review. *The Journal of Machine Learning Research*, 12:1185–1224, 2011. (Cited on page 64.)

D. B. Grimes, R. Chalodhorn, and R. P. Rao. Dynamic imitation in a humanoid robot through nonparametric probabilistic inference. In *Robotics: science and systems*, pages 199–206. Cambridge, MA, 2006. (Cited on page 75.)

N. Hammami and M. Bedda. Improved tree model for arabic speech recognition. In *IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, volume 5, pages 521–526. IEEE, 2010. (Cited on pages 178 and 181.)

N. Hammami and M. Sellam. Tree distribution classifier for automatic spoken arabic digit recognition. In *International Confonference for Internet Technology and Secured Transactions*, pages 1–4, 2009. (Cited on pages 178 and 181.)

J. Hartikainen and S. Sarkka. Sequential inference for latent force models. *arXiv preprint arXiv:1202.3730*, 2012. (Cited on page 78.)

J. Hartikainen, M. Seppänen, and S. Särkkä. State-space inference for non-linear latent force models with application to satellite orbit prediction. In *International Conference on Machine Learning (ICML)*, pages 903–910, 2012. (Cited on page 135.)

P. Hertel. Lectures on theoretical physics: Linear response theory. Technical report, University of Osnabruck, Germany, 2001. (Cited on page 93.)

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. (Cited on page 40.)

M. Hoffman, F. R. Bach, and D. M. Blei. Online learning for latent dirichlet allocation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 856–864, 2010. (Cited on page 116.)

M. Johnson and Y. Demiris. Abstraction in Recognition to Solve the Correspondence Problem for Robot Imitation. In *Proceedings of TAROS 2004*, pages 63–70. Essex, 2004. (Cited on page 75.)

M. J. Johnson and A. S. Willsky. Bayesian nonparametric hidden semi-markov models. *The Journal of Machine Learning Research*, 14(1):673–701, 2013. (Cited on page 60.)

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An Introduction to Variational Methods for Graphical Models. *Springer Machine Learning*, 37(2): 183–233, 1999. (Cited on page 48.)

D. S. Kalogerias and A. P. Petropulu. Matrix completion in colocated mimo radar: Recoverability, bounds & theoretical guarantees. *IEEE Transactions on Signal Processing*, 62(2):309–321, 2014. (Cited on page 134.)

K. Kim, D. Lee, and I. Essa. Gaussian process regression flow for analysis of motion trajectories. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1164–1171. IEEE, 2011. (Cited on page 76.)

S. Klanke, S. Vijayakumar, and S. Schaal. A library for locally weighted projection regression. *Journal of Machine Learning Research (JMLR)*, 9:623–626, 2008. (Cited on page 120.)

D. Korkinof and Y. Demiris. Online quantum mixture regression for trajectory learning by demonstration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3222 – 3229, November 2013. (Cited on pages 37 and 57.)

P. Kormushev, S. Calinon, and D. G. Caldwell. Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input. *Advanced Robotics*, 25(5):581–603, 2011. (Cited on page 28.)

P. Kormushev, S. Calinon, and D. G. Caldwell. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, 2(3):122–148, 2013. (Cited on page 68.)

K. Kurihara, M. Welling, and Y. W. Teh. Collapsed variational dirichlet process mixture models. In *IJCAI*, volume 7, pages 2796–2801, 2007. (Cited on page 58.)

N. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *The Journal of Machine Learning Research (JMLR)*, 6:1783–1816, 2005. (Cited on pages 67 and 135.)

N. D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in Neural Information Processing Systems (NIPS)*, volume 2, page 5, 2003. (Cited on pages 67, 135, and 153.)

N. D. Lawrence and A. J. Moore. Hierarchical gaussian process latent variable models. In *International Conference on Machine Learning (ICML)*, pages 481–488, 2007. (Cited on pages 67 and 135.)

D. Lee and Y. Nakamura. Mimesis scheme using a monocular vision system on a humanoid robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2162–2168, 2007. (Cited on pages 75 and 80.)

K. Lee, T. K. Kim, and Y. Demiris. Learning action symbols for hierarchical grammar induction. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3778–3782, 2012. (Cited on pages 73 and 158.)

K. Lee, Y. Su, T.-K. Kim, and Y. Demiris. A syntactic approach to robot imitation learning using probabilistic activity grammars. *Robotics and Autonomous Systems*, 61(12):1323–1334, 2013. (Cited on pages 73 and 158.)

B. G. Leroux et al. Consistent estimation of a mixing distribution. *The Annals of Statistics*, 20(3):1350–1360, 1992. (Cited on page 84.)

W. Li, Z. Zhang, and Z. Liu. Action recognition based on a bag of 3d points. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 9–14. IEEE, 2010. (Cited on pages 13 and 32.)

P. Liang and D. Klein. Online EM for unsupervised models. In *Annual Conference of the North American Association for Computational Linguistics (NAACL)*, 2009. (Cited on page 119.)

B. London, T. Rekatsinas, B. Huang, and L. Getoor. Multi-relational learning using weighted tensor decomposition with modular loss. *arXiv preprint arXiv:1303.1733*, 2013. (Cited on page 193.)

M. Lopes and J. Santos-Victor. Visual learning by imitation with motor representations. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, 35(3):438–449, 2005. (Cited on page 70.)

J. Luttinen and A. Ilin. Variational gaussian-process factor analysis for modeling spatio-temporal data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1177–1185, 2009. (Cited on pages 135 and 142.)

J. Luttinen and A. Ilin. Transformations in variational bayesian factor analysis to speed up learning. *Neurocomputing*, 73(7):1093–1102, 2010. (Cited on page 174.)

R. Ma, N. Barzigar, A. Roozgard, and S. Cheng. Decomposition approach for low-rank matrix completion and its applications. *IEEE Transactions on Signal Processing*, 62(7):1671–1683, 2014. (Cited on page 134.)

S. Ma, D. Goldfarb, and L. Chen. Fixed point and bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128(1-2):321–353, 2011. (Cited on pages 136 and 148.)

M. S. Malekzadeh, D. Bruno, S. Calinon, T. Nanayakkara, and D. G. Caldwell. Skills transfer across dissimilar robots by learning context-dependent rewards. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1746–1751. IEEE, 2013. (Cited on page 28.)

G. Marjanovic and V. Solo. On lq optimization and matrix completion. *IEEE Trans. on Signal Processing*, 60(11):5714–5724, 2012. (Cited on page 134.)

G. McLachlan and T. Krishnan. *Finite Mixture Models*. Wiley series in Probability and Statistics, 2000. (Cited on pages 44, 46, 76, 84, 98, and 115.)

N. Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, September 1949. (Cited on page 52.)

T. P. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001. (Cited on page 44.)

A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1257–1264, 2007. (Cited on pages 193 and 194.)

M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, and A. Weber. Documentation mocap database hdm05. 2007. (Cited on page 160.)

K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012. (Cited on page 44.)

I. Murray, R. P. Adams, and D. Mackay. Elliptical slice sampling. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 541–548, 2010. (Cited on page 53.)

C. S. Myers and L. R. Rabiner. A comparative study of several dynamic time-warping algorithms for connected-word recognition. *Bell System Technical Journal*, 60(7):1389–1409, 1981. (Cited on page 97.)

G. P. Mylonas, P. Giataganas, M. Chaudery, V. Vitiello, A. Darzi, and G.-Z. Yang. Autonomous efast ultrasound scanning by a robotic manipulator using learning from demonstrations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3251–3256. IEEE, 2013. (Cited on pages 29 and 168.)

R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *J. Comput. Graph. Statist.*, 9:249–265, 2000. (Cited on pages 57 and 81.)

C. L. Nehaniv and K. Dautenhahn. Of Hummingbirds and Helocopters: An algebraic framework for interdisciplinary studies of imitation and its applications. In Y. Demiris and A. Birk, editors, *Interdisciplinary Approaches to Robot Learning*, volume 24 of *Series in Robotics and Intelligent Systems*, pages 136–161. World Scientific, 2000. (Cited on page 74.)

D. Nguyen-Tuong and J. Peters. Local gaussian process regression for real-time model-based robot control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 380–385, 2008. (Cited on pages 75, 80, 95, and 107.)

S. M. Oh, J. M. Rehg, T. Balch, and F. Dellaert. Learning and inference in parametric switching linear dynamic systems. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1161–1168. IEEE, 2005. (Cited on pages 40, 62, 73, 74, and 77.)

S. M. Oh, J. M. Rehg, T. Balch, and F. Dellaert. Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. *International Journal of Computer Vision*, 77(1-3):103–124, 2008. (Cited on page 62.)

M. Opper and C. Archambeau. The variational gaussian approximation revisited. *Neural computation*, 21(3):786–792, 2009. (Cited on pages 134 and 146.)

J. W. Paisley and L. Carin. Nonparametric factor analysis with beta process priors. In *International Conference on Machine Learning (ICML)*, pages 777–784. ACM, 2009. (Cited on page 65.)

J. W. Paisley, A. K. Zaas, C. W. Woods, G. S. Ginsburg, and L. Carin. A stick-breaking construction of the beta process. In *International Conference on Machine Learning (ICML)*, pages 847–854, 2010a. (Cited on page 65.)

J. W. Paisley, M. Zhou, G. Sapiro, and L. Carin. Nonparametric image interpolation and dictionary learning using spatially-dependent dirichlet and beta process priors. In *IEEE International Conference on Image Processing (ICIP)*, pages 1869–1872. IEEE, 2010b. (Cited on page 134.)

J. W. Paisley, C. Wang, and D. Blei. The discrete infinite logistic normal distribution for mixed-membership modeling. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011. (Cited on page 59.)

J. W. Paisley, D. M. Blei, and M. I. Jordan. Variational bayesian inference with stochastic search. In *International Conference on Machine Learning (ICML)*, pages 1367–1374, 2012. (Cited on page 52.)

H. Pan, S. E. Levinson, T. S. Huang, and Z.-P. Liang. A fused hidden markov model with application to bimodal speech processing. *IEEE Transactions on Signal Processing*, 52(3):573–581, 2004. (Cited on page 60.)

R. Parhizkar, A. Karbasi, S. Oh, and M. Vetterli. Calibration using matrix completion with application to ultrasound tomography. *IEEE Transactions on Signal Processing*, 61(20):4923–4933, 2013. (Cited on page 134.)

B. A. Pearlmutter. Gradient calculations for dynamic recurrent neural networks: A survey. *IEEE Transactions on Neural Networks*, 6(5):1212–1228, 1995. (Cited on page 100.)

J. Peters and S. Schaal. Learning to control in operational space. *The International Journal of Robotics Research (IJRR)*, 27(2):197–212, 2008. (Cited on page 75.)

A. H. Phan and A. Cichocki. Tensor decompositions for feature extraction and classification of high dimensional datasets. *Nonlinear theory and its applications, IEICE*, 1(1):37–68, 2010. (Cited on page 193.)

A. Pistillo, S. Calinon, and D. G. Caldwell. Bilateral physical interaction with a robot manipulator through a weighted combination of flow fields. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3047–3052. IEEE, 2011. (Cited on page 29.)

J. Pitman and M. Yor. The Two-Parameter Poisson-Dirichlet Distribution Derived from a Stable Subordinator. *The Annals of Probability*, 25(2):855–900, April 1997. (Cited on page 58.)

C. Plagemann. *Gaussian Processes for Flexible Robot Learning*. PhD thesis, University of Freiburg, 2008. (Cited on page 75.)

D. Posada and T. R. Buckley. Model selection and model averaging in phylogenetics: advantages of akaike information criterion and bayesian approaches over likelihood ratio tests. *Systematic biology*, 53(5):793–808, 2004. (Cited on page 57.)

L. Rabiner and B.-H. Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986. (Cited on page 59.)

P. Rai, Y. Wang, S. Guo, and G. Chen. Scalable bayesian low-rank decomposition of incomplete multiway tensors. (Cited on pages 193 and 194.)

C. E. Rasmussen and C. K. Williams. *Gaussian processes for machine learning.*, volume 14. The MIT Press, 2004. (Cited on pages 65, 66, 135, 136, and 137.)

L. Ren, L. Du, L. Carin, and D. Dunson. Logistic stick-breaking process. *The Journal of Machine Learning Research*, 12:203–239, 2011. (Cited on page 59.)

P. Resnik and E. Hardisty. Gibbs Sampling for the Uninitiated. *Tutorial*, (June), 2010. (Cited on pages 42, 44, and 53.)

A. Rodrıguez and D. B. Dunson. Nonparametric bayesian models through probit stick-breaking processes. *Bayesian Analysis*, 6(1):145–178, 2011. (Cited on page 59.)

M. Rogers, L. Li, and S. Russell. Multilinear dynamical systems for tensor time series. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2634–2642, 2013. (Cited on pages 172, 190, 194, and 195.)

L. Rozo, S. Calinon, and D. G. Caldwell. Learning force and position constraints. *IEEE Human-robot Cooperative Transportation*, 2014. (Cited on page 28.)

R. Salakhutdinov and G. E. Hinton. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 448–455, 2009. (Cited on page 40.)

R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *International Conference on Machine Learning (ICML)*, pages 880–887. ACM, 2008. (Cited on page 40.)

G. Sandini, G. Metta, and D. Vernon. *The icub cognitive humanoid robot: An open-system research platform for enactive cognition*, volume 4850 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2007. (Cited on page 27.)

M.-A. Sato and S. Ishii. On-line em algorithm for the normalized gaussian network. *Neural computation*, 12(2):407–432, 2000. (Cited on pages 116, 119, and 124.)

S. Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999. ISSN 1879-307X. (Cited on page 75.)

S. Schaal and C. G. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10(8):2047–2084, 1998. (Cited on pages 13, 18, 31, 121, 122, and 124.)

G. Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6 (2):461–464, 1978. (Cited on pages 57 and 83.)

J. Sethuraman. A constructive definition of dirichlet priors. Technical report, DTIC Document, 1991. (Cited on pages 85 and 86.)

B. Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52: 55–66, 2010. (Cited on page 41.)

H. Soh and Y. Demiris. Iterative temporal learning and prediction with the sparse online echo state gaussian process. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2012. (Cited on page 121.)

H. Soh and Y. Demiris. When and how to help: An iterative probabilistic model for learning assistance by demonstration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3230–3236. IEEE, 2013. (Cited on page 75.)

H. Soh, Y. Su, and Y. Demiris. Online spatio-temporal gaussian process experts with application to tactile classification. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4489–4496. IEEE, 2012. (Cited on page 121.)

J. Sun, D. Parthasarathy, and K. Varshney. Collaborative kalman filtering for dynamic matrix factorization. *Signal Processing, IEEE Transactions on*, 62(14): 3499–3509, July 2014. ISSN 1053-587X. doi: 10.1109/TSP.2014.2326618. (Cited on page 134.)

K. Tanaka and K. Tsuda. A quantum-statistical-mechanical extension of gaussian mixture model. In *Journal of Physics: Conference Series*, volume 95, page 012023. IOP Publishing, 2008. (Cited on pages 57, 81, 90, 92, and 93.)

D. Tao, M. Song, X. Li, J. Shen, J. Sun, X. Wu, C. Faloutsos, and S. J. Maybank. Bayesian tensor approach for 3-d face modeling. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(10):1397–1410, 2008. (Cited on page 194.)

Y. W. Teh. A hierarchical Bayesian language model based on Pitman-Yor processes. *International Conference on Computational Linguistics*, pages 985–992, Morristown, NJ, USA, July 2006. (Cited on page 58.)

Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical dirichlet processes. *Journal of the american statistical association*, 101(476), 2006. (Cited on pages 58 and 60.)

Y. W. Teh, D. Görür, and Z. Ghahramani. Stick-breaking construction for the indian buffet process. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 556–563, 2007a. (Cited on page 64.)

Y. W. Teh, K. Kurihara, and M. Welling. Collapsed variational inference for hdp. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1481–1488, 2007b. (Cited on page 58.)

J.-A. S. Y. Ting. *Bayesian methods for autoomous learning systems*. PhD thesis, University of Southern California, 2009. (Cited on page 75.)

M. Titsias and M. Lázaro-Gredilla. Doubly stochastic variational bayes for non-conjugate inference. In *International Conference on Machine Learning (ICML)*, pages 1971–1979, 2014. (Cited on page 52.)

M. K. Titsias and N. D. Lawrence. Bayesian gaussian process latent variable model. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 844–851, 2010. (Cited on pages 67, 135, 141, 143, and 144.)

M. K. Titsias and M. Lázaro-Gredilla. Spike and slab variational inference for multi-task and multiple kernel learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 24, pages 2339–2347, 2011. (Cited on pages 64, 135, 136, 138, and 148.)

L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966. (Cited on page 192.)

R. Urtasun and T. Darrell. Discriminative gaussian process latent variable model for classification. In *International Conference on Machine Learning (ICML)*, pages 927–934. ACM, 2007. (Cited on pages 67 and 135.)

S. Vijayakumar and S. Klanke. *A library for Locally Weighted Projection Regression - Supplementary Documentation*. (Cited on page 120.)

S. Vijayakumar, A. D'souza, and S. Schaal. Incremental online learning in high dimensions. *Neural computation*, 17(12):2602–2634, 2005. (Cited on pages 75, 80, 120, 124, 126, 138, 155, and 168.)

C. Vollmer, J. P. Eggert, and H.-M. Gross. Modeling human motion trajectories by sparse activation of motion primitives learned from unpartitioned data. In *Advances in Artificial Intelligence*, pages 168–179. 2012. (Cited on page 138.)

S. G. Walker, P. Damien, P. W. Laud, and A. F. Smith. Bayesian nonparametric inference for random distributions and related functions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):485–527, 1999. (Cited on pages 57 and 81.)

C. Wang and D. M. Blei. Variational inference in nonconjugate models. *The Journal of Machine Learning Research*, 14(1):1005–1031, 2013. (Cited on page 51.)

C. Wang, J. W. Paisley, and D. M. Blei. Online variational inference for the hierarchical dirichlet process. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 752–760, 2011. (Cited on page 116.)

J. Wang, A. Hertzmann, and D. M. Blei. Gaussian process dynamical models. In *Advances in neural information processing systems*, pages 1441–1448, 2005. (Cited on page 78.)

J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(2):283–298, 2008. (Cited on page 78.)

A. Wilson, Z. Ghahramani, and D. A. Knowles. Gaussian process regression networks. In *International Conference on Machine Learning (ICML)*, pages 599–606, 2012. (Cited on page 135.)

F. Wood, J. Gasthaus, C. Archambeau, L. James, and Y. W. Teh. The sequence memoizer. *Communications of the ACM*, 54(2):91–98, 2011. (Cited on page 73.)

L. Xiong, X. Chen, T.-K. Huang, J. G. Schneider, and J. G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*, volume 10, pages 211–222. SIAM, 2010. (Cited on pages 171, 172, 193, 194, 196, and 201.)

Z. Yang, G.-A. Levow, and H. Meng. Predicting user satisfaction in spoken dialog system evaluation with collaborative filtering. *Selected Topics in Signal Processing, IEEE Journal of*, 6(8):971–981, Dec 2012. ISSN 1932-4553. doi: 10.1109/JSTSP.2012.2229965. (Cited on page 134.)

S.-Z. Yu. Hidden semi-markov models. *Artificial Intelligence*, 174(2):215–243, 2010. (Cited on page 60.)

P. Zegers and M. K. Sundareshan. Trajectory generation and modulation using dynamic neural networks. *IEEE Transactions on Neural Networks*, 14(3):520–533, 2003. (Cited on page 100.)

Q. Zhao, L. Zhang, and A. Cichocki. Bayesian cp factorization of incomplete tensors with automatic rank determination. *arXiv preprint arXiv:1401.6497*, 2014. (Cited on page 194.)

M. Zhou, C. Wang, M. Chen, et al. Nonparametric bayesian matrix completion. In *IEEE Sensor Array and Multichannel Signal Processing Workshop*, pages 213–216, 2010. (Cited on pages 134 and 135.)

M. Zhou, H. Yang, G. Sapiro, D. B. Dunson, and L. Carin. Dependent hierarchical beta process for image interpolation and denoising. In *International Conference on Artificial Intelligence and Itatistics (AISTATS)*, pages 883–891, 2011. (Cited on pages 134 and 138.)

M. Zhou, H. Chen, J. W. Paisley, L. Ren, L. Li, Z. Xing, D. Dunson, G. Sapiro, and L. Carin. Nonparametric bayesian dictionary learning for analysis of noisy and incomplete images. *IEEE Transactions on Image Processing*, 21(1):130–144, 2012. (Cited on pages 134 and 138.)