Nonlinear Dimensionality Reduction for Motion Synthesis and Control

Sebastian Bitzer

Doctor of Philosophy

Institute of Perception, Action and Behaviour

School of Informatics

University of Edinburgh

2010

Abstract

Synthesising motion of human character animations or humanoid robots is vastly complicated by the large number of degrees of freedom in their kinematics. Control spaces become so large, that automated methods designed to adaptively generate movements become computationally infeasible or fail to find acceptable solutions.

In this thesis we investigate how demonstrations of previously successful movements can be used to inform the production of new movements that are adapted to new situations. In particular, we evaluate the use of nonlinear dimensionality reduction techniques to find compact representations of demonstrations, and investigate how these can simplify the synthesis of new movements.

Our focus lies on the Gaussian Process Latent Variable Model (GPLVM), because it has proven to capture the nonlinearities present in the kinematics of robots and humans. We present an in-depth analysis of the underlying theory which results in an alternative approach to initialise the GPLVM based on Multidimensional Scaling. We show that the new initialisation is better suited than PCA for nonlinear, synthetic data, but have to note that its advantage shrinks on motion data.

Subsequently we show that the incorporation of additional structure constraints leads to low-dimensional representations which are sufficiently regular so that once learned dynamic movement primitives can be adapted to new situations without need for relearning. Finally, we demonstrate in a number of experiments where movements are generated for bimanual reaching, that, through the use of nonlinear dimensionality reduction, reinforcement learning can be scaled up to optimise humanoid movements.

Acknowledgements

I am the product of the influences of the people I met. Consequently, I want to thank again my family, my friends and teachers of my former lifes who brought me into the position to go on the endeavour of pursuing a PhD in the first place. I particularly thank Kevin Spellman and Johanna Pagels who continued to be good friends in Edinburgh and who, no matter how long we had not met, always gave me the feeling that it was yesterday. However, it is the people who guided, supported and motivated me throughout the sometimes painful undertaking of PhD research who I want to thank specifically here.

I am grateful to my supervisor, Sethu Vijayakumar, for always providing me with the opportunities necessary to advance my project, for pointing me into the right direction, for encouraging me to engage with the scientific community and for bringing me back down to earth when my ambitions threatened to take me on paths inaccessible during a period of PhD study. I thank our postdoc, Stefan Klanke, for guiding me through the practicalities of research, for many useful down-to-earth discussions about technical problems, for getting things efficiently done when needed for my project and for always lending me his ear for my problems, in general, whether as a friend, or as a colleague. I am grateful to Chris Williams for sharing his brilliant ideas and insights with me, for having the right intuitions when interpreting results and for being sufficiently persitent to carry me through a valley of negative results.

I thank my office mates and fellow PhD students. Adrian Haith for his sharp, polite analyses and for introducing me to golf. Matthew Howard for his confident, relaxed judgements and his robotics view on my work. Djordje Mitrovic for making the office to a lively place, for challenging my views and for his never dwindling well of creativeness. Ioannis Havoutis for piloting some of my work and for including me into the Greek family. Hannes Saal for motivating us through his own, silent drive to success. Ian Saunders for surprising everyone with fun props and innovative experiments (and his tendency to get hurt - no wonder he investigates prosthetics). Konrad Rawlik for his unifying mathematical perspective and for the spontaneous oddities that made me laugh.

I also thank the remaining regulars of IPub (Toby Collins, Rowland Silito, Finlay Stewart, Tom Larkworthy, Michael Mangan, Tim Lukins, Jan Wessnitzer, Mark Payne, Theophile Gonos, Aroosha Laghaee, Georgios Petrou, Aris Valtazanos, Benjamin Rosman, Evelina Overlingaite) for making Friday evenings an eagerly anticipated event. Similarly, I thank Andi Winterboer, Martin Tietze, Sebastian Andersson,

Michael Kaisser, Ivan Meza-Ruiz, Jens Apel, Marc-Andre Martel, Joao Cabral and Marc Heise for many valuable non-robotic evenings in Bannerman's and the occasional football viewing.

During my PhD I found in volleyball the perfect way to balance the intellectual challenges of my studies with the physical activities and required mental strength of the game. I am grateful to everyone in Scottish volleyball who made my experience so exciting, fun and in general worthwhile. I thank Gerry MacDonald and Andrew Slavin as representatives of NUVOC for taking me on in the team, developing me further and eventually recognising what I can do. I thank the players of Portobello beach (Mauricio Lopez, Graham Riddle, Lars Jeppesen, Mel Coutts, Barry McGuigan, Ulrich Heitzlhofer, Carlo Zachau, George Sevastos, Colin Macnab, Gavin Watt) for being patient with me, for teaching me sooo many things, for enduring the Scottish weather with me and for simply being a fun crowd. Last, but not least, I thank the students of EUVC (Chris Moultrie, Paul Johnson, Jamie Evenden, Michal Bury, Colin Macnab, Jan Domozilov, Nathan Agee, Ulrich Heitzlhofer, Findlay Haddow, Boudewijn Grievink, George Sevastos, Stuart Parker, Paolo Puggioni, Abi O'Connor, Fiona Macpherson, Lorraine Steel and many more) for being the best team mates for who you always want to give everything, for listening to and trusting me and for throwing outstanding parties.

I also want to thank my former flatmates Veronica Wolf, Natalia Issaeva, You-Ying Chau, Rosemary Apple, Lindsay Egan, Ralph Lumby, Graeme Jarvie, Lea Zielke, Diane Le Cottier, Stephanie Monk and Simon Trüb for providing a home that you always enjoy returning to.

Finally, I am deeply grateful to Alex Reddaway, Verena Rieser, Wendy van der Neut, Sophia Perie and Florine Hiersemenzel for making me feel alive more than anyone, or anything else in the world.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Sebastian Bitzer*)

Table of Contents

1 Introduction **1**

2 Nonlinear Dimensionality Reduction Methods on Motion Data **11**

2.1 A Motion Capture Example 12

2.2 Dimensionality Reduction Methods 14

2.2.1 Non-Generative Methods 15

2.2.2 Generative Methods 18

2.3 Motion Interpolation in Latent Spaces 26

2.3.1 Robotic Motion Data (DLR arm) 26

2.3.2 DR Method Details 28

2.3.3 Results 29

2.4 Discussion 32

3 GPLVM-MDS **35**

3.1 Relating MDS, PCA and the GPLVM 36

3.1.1 Relating Classical MDS and PCA 36

3.1.2 Relating probabilistic PCA and the GPLVM 38

3.1.3 Relating the GPLVM and metric MDS 42

3.2 Metric MDS with Missing Data 43

3.2.1 Iterative Minimisation of Stress 43

3.2.2 Probabilistic Matrix Factorisation 44

3.2.3 Robustness Experiments 45

3.3 Variability of Covariance Estimates 49

3.3.1 Generating Synthetic Data from the GPLVM 49

3.3.2 Distribution of the Sample Covariance Matrix 51

3.3.3 Sample Covariance and Reconstruction Quality 52

3.4 Solving the GPLVM 56

3.4.1 Comparison to PCA on Synthetic Data 56
3.4.2 Results of GPLVM Optimisation 60
3.4.3 Comparison to Isomap 66
3.5 Motion Capture Data 68
3.5.1 Latent space dimensionality 70
3.5.2 Normalisation of data 71
3.5.3 Results 72
3.5.4 Conclusion 76
3.6 Discussion 76

4 Dynamic Movement Primitives in Latent Spaces 79
4.1 Introduction 79
4.2 Dynamic Movement Primitives 81
4.2.1 Formulation 82
4.2.2 Dynamic Movement Primitives in Joint Space 85
4.2.3 Dynamic Movement Primitives in Latent Space 87
4.3 Latent Spaces for Dynamic Movement Primitives 89
4.4 Results 91
4.4.1 DLR 91
4.4.2 Human Motion Capture Data 94
4.5 Discussion 97

5 Reinforcement Learning in Latent Spaces 99
5.1 Introduction 100
5.2 Learning the State Abstraction from Demonstrated Examples 101
5.2.1 Example: Constrained Bimanual Manipulation 102
5.2.2 Finding Suitable Latent Spaces 104
5.3 Out-of-Sample Mappings for the GPLVM 106
5.3.1 Out-of-Sample Mapping Methods 107
5.3.2 Evaluation of Out-of-Sample Mappings 109
5.3.3 Ill-Defined Inverses 112
5.3.4 Conclusion 115
5.4 Reinforcement Learning in Latent Space 116
5.4.1 Preliminaries 116
5.4.2 TD(0) V-Learning 116
5.4.3 Incorporating the Latent Space State Representation 117

5.5 Experiments 118
5.5.1 Bimanual Reaching in End-effector Space 119
5.5.2 Bimanual Reaching in Joint space 123
5.5.3 A Planar DLR-Arm Problem 126
5.5.4 Full-Body Humanoid Reaching 127
5.6 Discussion 129

6 Conclusion **133**

A Scale of Probabilistic PCA Latent Variables **135**

B Calculation of Gradients **137**
B.1 Likelihood Gradients for Out-of-sample GPLVM Mapping 137

Bibliography **139**

Chapter 1

Introduction

As we design robots to become more anthropomorphic with an aim for them to co-exist in human friendly environments, the number of degrees of freedom and consequently the variety of movements that they can execute have grown significantly. This raises many issues concerning the control and planning in these robots: Whose task is it to define such a rich, complicated movement set for every new robot? How do you make those movements look natural? How do you cope with the large degree of redundancy?

A promising way out of this dilemma is for the robot (student) to learn the desired movements from a teacher (e.g., human demonstrator) through imitation. Early variants of this approach have already been proposed in the late 1970s (Grossman, 1977). The terms "teaching by showing" and *"guiding"* were then used in the context of industrial robots to describe the practice of demonstrating robot movements as a series of a few target points of the end-effector at which additional commands, e.g. weld, could be executed (Lozano-Perez, 1983). Since then, more general, continuous representations for demonstrated movements, supported by machine learning algorithms, have been developed. Recently several reviews of the field have been published. Billard et al. (2008) provide a comprehensive, but very concise overview while Argall et al. (2009) focus on the problem of extracting policies (mapping states to actions) from demonstrations. The field has been called "robot programming by demonstration", "robot learning from demonstration", "imitation learning" and probably any sensible combination of these. "Imitation learning" is the most recent term and it highlights the connection to imitation in biology. It also stands for a widening of the field towards human-robot interaction which additionally addresses the more fundamental questions of *who*, *when* and *what* to imitate, in contrast to just *how* to imitate (Billard et al., 2008).

In contrast to the former practice of teaching a robot through guiding, more general imitation learning approaches have to additionally solve the correspondence problem (Nehaniv and Dautenhahn, 2002), i.e., the problem of relating the movement as demonstrated by the teacher to the body of the student. For example, it is unclear how a movement recorded from a human with 57 degrees of freedom (DOF) should be translated to a humanoid robot with only 19 DOF. While ad-hoc correspondences between the 19 robot DOF and a subset of 19 DOF of the human may be found, the limits of movement in each of the 19 DOF may still be different (for example, a DOF of the human may range from 0° to 90° while the corresponding DOF of the robot only ranges from 0° to 70°). Even if this problem can be solved, the resulting movement may not be dynamically stable on the robot, because of a different distribution of weights across body parts. Although there are suggestions for filtering human motion to make it dynamically stable on a humanoid motion (Naksuk et al., 2005; Yamane and Nakamura, 2003), their assumptions, or computational constraints make general applicability questionable. Shon et al. (2006) suggested to learn a common space between teacher and student poses for which correspondences are known, but they do not answer where the matched poses should come from in the first place. One possibility is through optimisation of an approximately matched initial movement on the robot. Grimes et al. (2006) and Chalodhorn et al. (2010) proposed such procedures in which stability is defined as minimum differences in pressure sensors at the feet, or minimum, predicted gyroscope readings from the torso of a humanoid robot. This definition of stability is a very crude heuristic. Alternatively it can be attempted to optimise an initial trajectory with reinforcement learning in more general settings (e.g. Peters and Schaal, 2008b; Guenter et al., 2007).

The correspondence problem continues to be one of the biggest hurdles in imitation learning. Hence, some studies (e.g. Ito et al., 2006; Calinon et al., 2007; Calinon and Billard, 2009; Peters and Schaal, 2008b; Guenter et al., 2007) in robot learning from demonstration bypass it by returning to the old practice of guiding which is now called kinaesthetic teaching and in contrast to guiding also includes demonstrations of continuous trajectories. Hence, kinaesthetic teaching now also includes the style and timing of demonstrated movements.[1] Suitable continuous representations for the recorded demonstrations have been proposed. Ijspeert et al. (2003) suggested dynamic movement primitives (cf. Chapter 4) which represent a demonstrated trajectory as the attractor of a set of dynamical systems which has advantages for control, but other ap-

[1] Also teleoperation can be seen as a form of kinaesthetic teaching (Argall et al., 2009).

proaches have been used such as autoregressive models (Wang et al., 2008), or hidden Markov models (Inamura et al., 2004; Calinon and Billard, 2005; Kulic et al., 2008; Lee and Nakamura, 2010), or regression on time using nonparametric (Lawrence and Moore, 2007), or Gaussian mixture models (Calinon et al., 2007; Calinon and Billard, 2009).

While kinaesthetic teaching naturally overcomes the correspondence problem between demonstrator and imitator bodies and is a very intuitive way of interacting with a robot, it is limited to robots with a small number of DOF which can simultaneously be controlled by the demonstrator. In Chapter 5 we show how kinaesthetic teaching can be extended to full-body motions of humanoids by only requiring discrete samples from the relevant postures of the motion instead of the complete continuous motion. This allows demonstrators to move all joints in a single posture to desired positions in several steps. This, then, is very similar to guiding. The key difference is that in guiding the demonstrator only controlled the position and orientation of the end-effector of a non-redundant, industrial robot whereas we now consider robots with redundant DOF and it is our aim to not only learn the parts of the motion directly related to the task, but also how redundancies in the robot kinematics are resolved.

In robotics we differentiate between *joint space, task space* and *null space* (Liégeois, 1977; Nakamura, 1991; Siciliano and Khatib, 2008). In *joint space* each coordinate corresponds to one DOF (joint) of the robot and any particular configuration of joints (a pose, posture) is represented as one point in joint space. The *task space* is the space in which the task is defined. Most frequently this is just the 3D Cartesian space of the real world which may, however, have its origin at various places (e.g. base of robot, or target). For example, any posture of a traditional, 6 DOF industrial robot corresponds to a point in its 6D joint space. Its task space is defined as the 3D position of its end-effector together with its 3D rotation in the 3D Cartesian space centred at the robot base. The task space, therefore, also is 6D and constrains all 6 DOF of the robot. So this robot is not redundant and no null space exists where *null space* is a theoretical construct controlling the remaining DOF of the robot which are not constrained through the task. The 7 DOF DLR arm as described in Section 2.3.1 has effectively one redundant DOF, when a 6 DOF task space is prescribed. A point in task space (also defined as position and orientation of end-effector), therefore, determines a line in joint space rather than a point. To select one particular joint angle configuration a point in null space has to be chosen additionally. Note that the DOF in null space often do not directly correspond to a subset of the robot DOF (cf. Fig. 2.4(d) where for a

change in 1D null space several joints of the robot change).

The mapping from joint space to task space is called *forward kinematics*. It is a many-to-one mapping and can be easily computed, if the details of the robot body are known. The mapping from task space (e.g., a desired end-effector position) to joint space is called *inverse kinematics*. It is one-to-many and hence cannot be computed without knowledge of how redundancy is resolved. Several approaches to redundancy resolution have been suggested from a theoretical (D'Souza et al., 2001; Tevatia and Schaal, 2000; Whitney, 1969) and an engineering perspective (Chiaverini et al., 2008), but in a learning from demonstration setting it can also be learnt from the demonstrations. In particular, Calinon and Billard (2009) and in an earlier version Calinon et al. (2007) include demonstrations as constraints in task and joint space into an optimisation which synthesises new movements.

Through an appropriate definition of the task space a lot of information about the demonstrated movements is already given to the learning from demonstration system. For example, Calinon and Billard (2009) defined the task space centred on the initial position of an object that is to be relocated. Consequently, moving towards that object will simply be moving to the origin in the defined task space, irrespective of where the movement started. One step away from incorporating this information independent from the demonstrations was suggested by Muehlig et al. (2009) who designed a system which selects a task space from a pool of candidate task spaces according to a set of predefined criteria which are evaluated on demonstrated movements.

In this thesis, we investigate the case where, apart from the kinaesthetically taught demonstrations themselves, no other information about the movements is known. This situation may occur when a new movement skill is taught to the robot by a demonstrator who, while competent in the skill, is not an expert in robotics. In this scenario, the only information available comes from the statistics of the demonstrated data points. However, we make further basic assumptions about the data: a) All demonstrations belong to the execution of a single task, so there exists an underlying (unobserved) task space which is lower dimensional than the joint space, i.e., the robot is redundant. b) Redundancy is resolved in a consistent way, for example, by optimisation of a continuous criterion, such that the variability in the data is predominantly related to the task (subject to noise). Furthermore, we assume to know the dimensionality of the task space in our experiments and we give justification for our choices in experiment descriptions. Note that task spaces can also be smaller than the available range of movement, if they are defined on end-effector positions. For example, in the task

of wiping a windscreen with one hand of a 27 DOF humanoid, the task space is constrained to the 2D surface of the windscreen. Howard et al. (2009) use this example amongst others to evaluate their method for extracting a general, unconstrained policy from a set of differently constrained demonstrations. Our focus here is different: From demonstrations, we try to capture implicit constraints present in the data such that subsequently generated movements also adhere to the learnt constraints.

Our approach to this problem is based on dimensionality reduction (DR) which is the task of recovering meaningful low-dimensional structures hidden in high-dimensional data (de Silva and Tenenbaum, 2003). In the example above, as a consequence of our assumptions, we expect that successful demonstrations of windscreen wiping lie on a 2D manifold in the 27D joint space of the humanoid (subject to noise in actuation and sensing). Our aim then, is to recover a 2D latent representation of that manifold from a data set in 27D with DR and generate new movements from the latent representation such that the constraints in the data are automatically fulfilled. Therefore, the DR method employed should meet two main requirements: 1) It needs to be nonlinear, because we know that task and joint spaces are usually nonlinearly related. 2) It needs to define a generative mapping which can be used to obtain points in joint space from points in the low-dimensional representation.

So far, nonlinear DR has rarely been used to represent movements in robotics. (See next chapter for a description of the mentioned methods.) Tatani and Nakamura (2003) applied an autoassociative neural network to motion patterns of a humanoid robot, but only investigated the ability to reproduce the original patterns without generation of new movements. Chalodhorn et al. (2009) also used autoassociative networks to segment and reproduce periodic motions of a humanoid robot. Steffen et al. (2009) adapted unsupervised kernel regression for a data set of hand movements. None of these studies compared their results with other methods. Linear DR, in particular principal components analysis (PCA), is often embedded into learning from demonstration systems, although merely as an intermediate, computationally more efficient representation without claim to represent the task space particularly well (Grimes et al., 2006; Chalodhorn et al., 2010; Calinon and Billard, 2005; Calinon et al., 2007).

The learning from demonstration framework in robotics is also similar to motion synthesis in animation when new motions are generated from motion capture data. Impressive results can be obtained, when a large database of motion capture data is available, by organising the database in a suitable way (Kovar et al., 2002; Lee et al., 2002). However, in our scenario we focus on learning one particular skill from very

few demonstrations when extensive and exhaustive motion generation or capture is either expensive or infeasible. If two similar motions are available, linear interpolation between these works surprisingly well when they are represented as absolute positions and rotations of body parts in a global coordinate system (Wiley and Hahn, 1997). Also linear combination of motion sequences has been shown to work reasonably well (Giese and Poggio, 2000; Safonova and Hodgins, 2005). In this context, dimensionality reduction can be recognised as a particular way of interpolating between poses when the low-dimensional space is used to generate new poses. In particular, nonlinear DR then is a basis for nonlinear interpolation[2] although the low dimensionality of the representation certainly has additional benefits such as computational efficiency for subsequent applications. For example, Urtasun et al. (2006) made use of the low-dimensional representation and interpolation capabilities of a Gaussian process dynamical model (Wang et al., 2008) for tracking movements of people from video. This model is an extension of the Gaussian process latent variable model (Lawrence, 2005) which had before been suggested as a tool for animators by providing inverse kinematics based on motion capture data (Grochow et al., 2004). The model has been further adapted for hierarchies of motions (Lawrence and Moore, 2007) and for periodic motions (Urtasun et al., 2008). Of course, PCA has also been applied for motion synthesis in the animation community. For example, Urtasun et al. (2004) concatenated all frames of a motion to one data vector and used PCA to find a low-dimensional representation of a set of motions (walking, running, jumping) which allowed to transfer a motion style to the motion of a person for which that style had not been observed. Torresani et al. (2007) utilised PCA as an intermediate representation for labelled motion fragments in a motion synthesis system based on dynamic time warping. Finally, Safonova et al. (2004) used PCA on individual frames to enable a highly engineered optimisation of motions under constraints given by an animator.

The main difference between learning from demonstration in robotics and motion synthesis in animation is the focus on online, continuous control in robotics which is necessitated by the ongoing interaction of a robot with the physical world while animations are typically prepared for a fixed setting and then just replayed. Although we particularly have control applications in mind (see Chapters 4 and 5), our results apply to compact representations of motions in general and are therefore also of interest for motion synthesis in animation.

Our aims in this thesis are twofold. On the one hand, we suggest DR as a tool for

[2]These considerations have inspired the experiment in Section 2.3.

learning from demonstration, because it potentially provides representations of movements which simplify the generation of new movements from demonstrations. On the other hand, we endorse in particular the use of nonlinear DR methods to be able to approximate the nonlinear relation between recorded joint angle motions and the physical variables underlying the considered task. At the same time we aim to provide strict quantitative evaluations of the considered methods on widely varying platforms (toy simulations, DLR arm, humanoid robot, human motion capture). In the following we present a concise overview of the chapters in this thesis and list their main contributions.

In **Chapter 2** we introduce and compare the most common nonlinear DR methods and in particular review generative methods available at the time of writing.

Original Contributions

- comparison of a large set of generative DR methods on a motion capture example
- evaluation of quality of motion interpolation in latent spaces

Papers

- Bitzer, S., Klanke, S. and Vijayakumar, S. (2009). Does Dimensionality Reduction Improve the Quality of Motion Interpolation? *Proceedings of the 17th European Symposium on Artificial Neural Networks (ESANN)*, pages 71–76.

In **Chapter 3** we analyse our preferred method for nonlinear DR, the Gaussian process latent variable model (GPLVM), and relate it to multidimensional scaling (MDS). As a consequence we gain an initialisation for the GPLVM which outperforms PCA.

Original Contributions

- establishing theoretical relation between GPLVM and MDS
- comparison of methods for executing MDS on a distance matrix with missing entries (including application of probabilistic matrix factorisation)
- analysis of GPLVM-MDS showing that ability to reconstruct underlying latent configuration increases with dimensionality of observations
- comparison of GPLVM-MDS and PCA as initialisations for the GPLVM on synthetic and motion capture data

Papers

- Bitzer, S. and Williams, C. (2010). GPLVM-MDS: Kick-starting GPLVM Optimization via a Connection to Metric MDS. *Advances in Neural Information Processing Systems 23*, (submitted).

In **Chapter 4** we show that nonlinear DR can be used to extend the generalisation capabilities of dynamic movement primitives used to represent a set of demonstrated movements. As this application makes high demands on the regularity of the latent space, we introduce an adaptation of the GPLVM which allows to incorporate structural information about the demonstrations known a priori.

Original Contributions

- showing that latent variables learnt with nonlinear DR can relate to task variables governing demonstrated movements
- template based prior for GPLVM allowing to flexibly incorporate prior information

Papers

- Bitzer, S., Havoutis, I. and Vijayakumar, S. (2008). Synthesising Novel Movements through Latent Space Modulation of Scalable Control Policies. In *From Animals to Animats 10*, pages 199–209. Springer.
- Bitzer, S. and Vijayakumar, S. (2009). Latent Spaces for Dynamic Movement Primitives. In *Proceedings of 9th IEEE RAS International Conference on Humanoid Robots (Humanoids 09)*.

In **Chapter 5** we demonstrate the use of nonlinear DR in an online control setting in the framework of reinforcement learning. We also show that by using DR to represent constraints present in demonstrations, reinforcement learning can be scaled to problems which are otherwise intractable.

Original Contributions

- evaluation of out-of-sample mappings for the GPLVM
- extension of kinaesthetic teaching to robots with more DOF than can be handled by a single person

- successful reinforcement learning of full-body humanoid movements starting from a randomly initialised policy

Papers

- Bitzer, S., Howard, M. and Vijayakumar, S. (2010). Using Dimensionality Reduction to Exploit Constraints in Reinforcement Learning. To appear in Proceedings of *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Chapter 2

Nonlinear Dimensionality Reduction Methods on Motion Data

A tremendous wealth of dimensionality reduction (DR) methods has been proposed in the literature and the number of available methods grows yearly. This is 1) due to differing amounts of prior information available in different settings and 2) due to different assumptions about the relationship between high-dimensional data and low-dimensional representation[1]. For example, regarding 1), taking class membership of data points into account for a classification problem should result in a different low-dimensional configuration of points compared to when only the covariances between data points are considered. This additionally highlights that the purpose of the extracted low-dimensional representation determines the criteria that we use to evaluate it and so can also influence it. In this thesis, we have several objectives driving the choice of DR methods. In the context of motion we want to use demonstrated movements or postures to generate new movements by interpolation and we evaluate existing methods with respect to that aim in Section 2.3 while we adapt an existing method in Chapter 4 to improve interpolation of dynamic movement primitives, and show how low-dimensional representations which interpolate between a few, single postures can be used to learn online control in Chapter 5.

The main criterion to categorise DR methods is according to their assumptions about the relationship between high-dimensional data and low-dimensional representation (point 2 above). While these assumptions are made explicit in probabilistic modelling they are often less clear in alternative approaches. Below, we review some of the most common DR methods and roughly categorise them in non-generative (sec.

[1] and, of course, due to the fact that in a true unsupervised setting, nobody knows the correct solution

2.2.1) and generative methods (sec. 2.2.2). Since we are not only interested in visualisation of the low-dimensional structure, but also want to generate new movements based on it, our focus lies on generative methods which explicitly model the mapping between low-dimensional and high-dimensional space. Other methods are mentioned, because they can be used to provide initialisations for the iterative algorithms often employed to learn generative models. We also introduce the Gaussian Process Latent Variable Model in Section 2.2.2.1 which, due to its Gaussian Process mapping between low- and high-dimensional space, is a particularly powerful and flexible method and is thus at the core of our investigations.

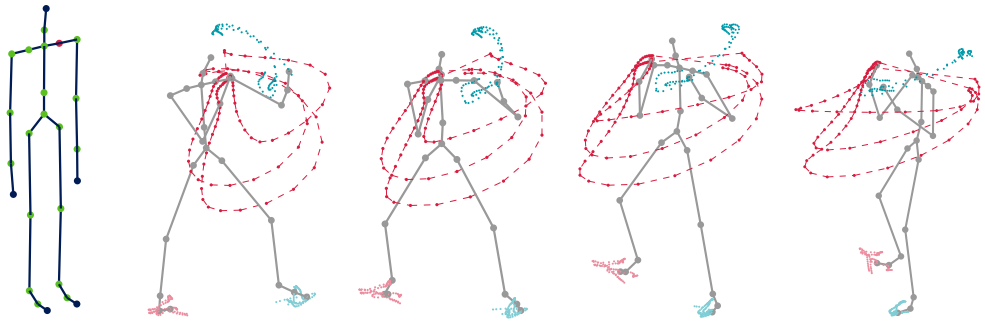
In the modelling literature the low-dimensional representation is usually defined in terms of latent (unobserved) variables. We adopt this terminology and use latent points/representation interchangeably with low-dimensional points/representation as well as latent space interchangeably with low-dimensional or reduced space. When we speak of a latent configuration, we mean the configuration of points in latent space which is also equivalent to simply "latent points" which, in turn, are the representations of the high-dimensional data points in low-dimensional space.

2.1 A Motion Capture Example

Before we describe the DR methods we present the details of a motion capture data set which we use to show exemplary latent configurations produced by selected methods.

This data set consists of three punches recorded from a human using a marker based motion capture (mocap) system[2]. Marker positions were translated into joint angles using a kinematic model of the human. This model contained 19 joints plus a root node which is used to encode global position and rotation of the body. All rotations were represented with 3 Euler angles in degrees which are resolved by rotation around z-, x- and y-axis in this order where the xyz coordinate system follows the conventions of the BVH file format for mocap data. Therefore, the movements were encoded with 63 degrees of freedom (DOF), 60 rotations plus 3D position of the root in Euclidean space. In terms of control, however, the position of the root (defined at the hip of the skeleton), i.e. the global translation of the body in space, is merely an effect of the movements in the joints. Therefore we removed the translation of the root from this data set. Additionally we removed the rotations of a joint placed on the left collarbone (red dot in Fig. 2.1(a)), because they could not be recorded and were always 0. As this

[2]Data provided by Taku Komura, School of Informatics, University of Edinburgh.



(a) DOF (b) Trajectories of hand and feet.

Figure 2.1: Motion capture Punches. (a) Degrees of freedom of the motion capture model. Green dots: recorded joints (each allows full 3D rotation), red dot: joint fixed in recordings. (b) Trajectories of the four limbs for the three punches. Red dots: right extremities, blue dots: left. Points of right hand connected by dashed line for clarity. Skeleton plotted in same pose (1st frame of high punch) for the 4 different views as reference. The largest movement is clearly in the right hand (punch hand) compared to other extremities.

joint naturally contributes little to the overall movement, missing its changes in joint angles did not severely impair the quality of the recording. The remaining number of DOF, thus, was 57.

We selected the three punches in the data set, because visually they mainly differ in the height of the punch. There is one high, one low and one very low punch, where there is not much difference in height between low and very low punches (see Fig. 2.1(b)). The original punches contained 76, 107 and 84 frames for high, low and very low punches, respectively, with frames recorded at 60 Hz. As well as the actual punch, they also contained the return to the initial position. We cut the movements further so that they only contained the punches and the movements all start after the first frame, leaving us with 27, 33, and 37 frames. Because our application in Chapter 4 requires sequences with equal length we subsampled the high and very low punches with cubic spline interpolation to 33 poses. This had no impact on the visual quality of the motions. Therefore, the final data set on which we applied DR had 99 data points with 57 dimensions. We normalised this data such that data points were centred and had unit standard deviation in all dimensions. The main features of these movements were the punch height and the state within the punch (related to time). We thus fix the dimensionality of the latent space to 2 for this data set, but it potentially contains

additional variability.

We use this data to illustrate in the next section that the latent configurations resulting from DR on motion data can differ greatly across methods.

2.2 Dimensionality Reduction Methods

The categorisation of DR methods in this section is based on the importance of the ability to determine the so-called pre-image[3] of a latent point in the original, high-dimensional space which is necessary for reconstructing movements from latent space. We therefore categorise methods according to whether they explicitly model this generative mapping. In probabilistic modelling this corresponds to the distinction between generative and discriminative models. While generative models try to learn the joint distribution of the latent variables z and the observed variables y, $P(z,y) \propto P(y|z)P(z)$, and aim to determine the posterior $P(z|y)$ by inference, discriminative models learn the posterior directly and do not define the conditional $P(y|z)$ which can be used to determine pre-images. Because the DR methods without explicit generative mapping are typically not probabilistic models, we chose the more general category non-generative instead of discriminative below.

In the following descriptions we adhere to the convention that N data points in a D dimensional space are stored in matrix $\mathbf{Y} \in \mathbb{R}^{N \times D}$ and the corresponding latent points in a space with M dimensions are stored in matrix $\mathbf{Z} \in \mathbb{R}^{N \times M}$ where $M < D$.

PCA Principal Components Analysis is probably the most commonly used DR technique. Several other methods can be related to PCA and we, therefore, provide a concise introduction of the method before considering the categorisation of other methods. It is based on the assumption that the latent and observed variables are linearly related and it can be derived as a projection which maximises the variance of the projected data, or as a projection which minimises the average projection cost (Bishop, 2006, ch. 12.1). In both cases the latent points are computed by projecting the centred data points onto the eigenvectors of the data covariance matrix with the M largest eigenvalues: $\mathbf{Z} = \mathbf{Y}\mathbf{U}_M$ where $\mathbf{U}_M \in \mathbb{R}^{D \times M}$. In return, the reconstruction of a latent point in data space can be obtained as a linear combination of the found eigenvectors: $\mathbf{y} = \mathbf{U}_M\mathbf{z}$. For

[3] This terminology is adopted from Kernel PCA (Schölkopf et al., 1998). In particular, the pre-image of a latent space point $\mathbf{z}$ is its representation in data space $\mathbf{y}$ for which $\mathbf{y} = f^{-1}(\mathbf{z})$ where f is the dimensionality reduction mapping.

increasing number of used eigenvectors M the latent points $\mathbf{Z}$ then explain an increasing amount of the data variance[4] and therefore the reconstruction of $\mathbf{Y}$ becomes more accurate. Although even nonlinear data sets can thus be represented with PCA with low error, this has to be traded against the benefits of having a truly low-dimensional representation with small M. PCA can also be formulated as a linear probabilistic model and can be considered as a special case of factor analysis where the variance of the noise in each of the D observed variables is set to the same value (Bishop, 2006, ch. 12.2). Note, however, that the scale of the latent variables differs between PCA and probabilistic PCA (see Section 3.4.1 and Fig. 2.2(a,b)).

2.2.1 Non-Generative Methods

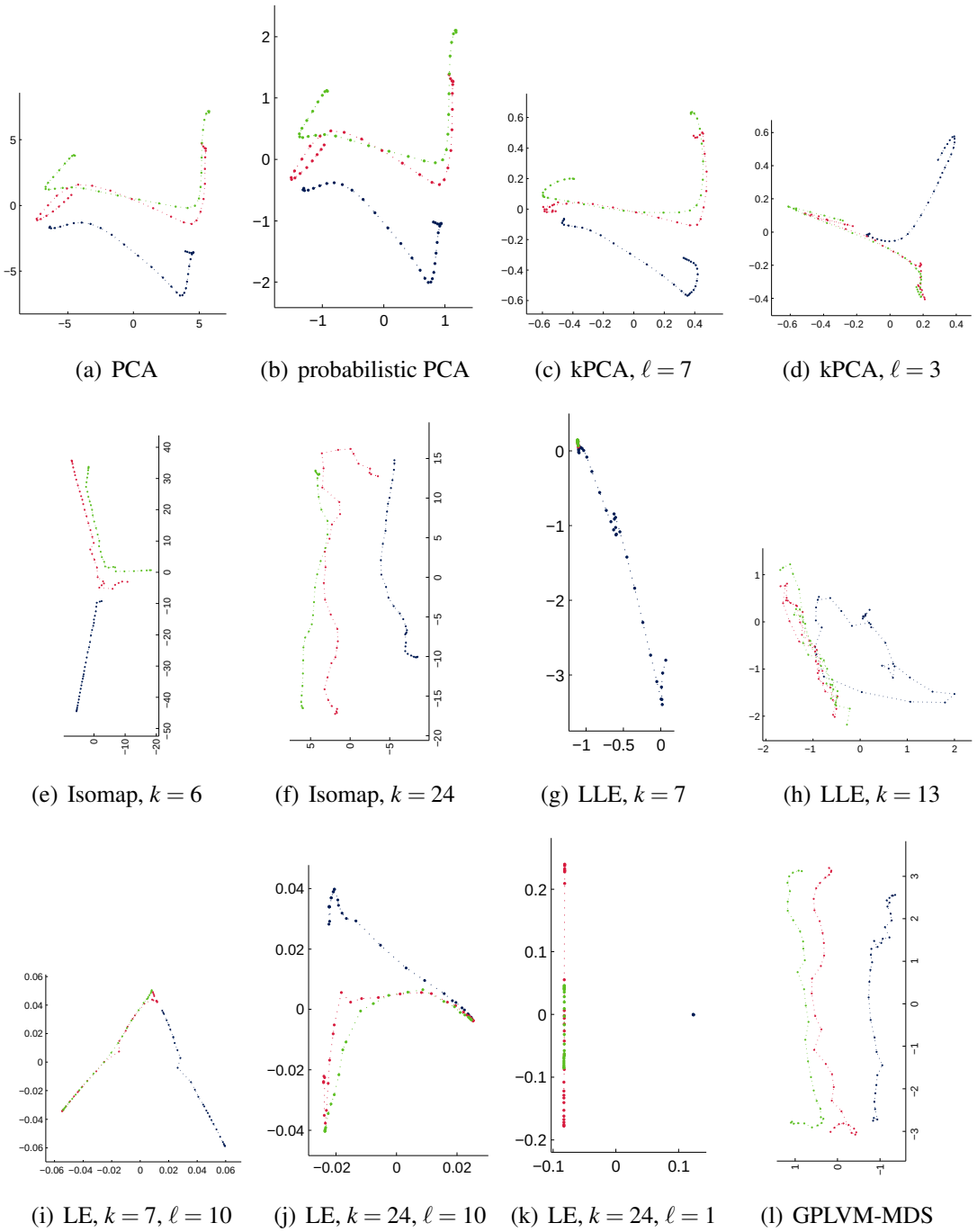
Due to the abundant number of DR methods and their variants it shall not be our aim to give a comprehensive overview of all of them. We refer the reader, for example, to van der Maaten et al. (2009) for a more exhaustive review of (mostly) non-generative DR methods. Instead we here concisely introduce some of the most commonly used methods to exemplify the breadth of available techniques.

kPCA Kernel PCA (Schölkopf et al., 1998) is a nonlinear extension to PCA. It maps the data points into a feature space using a nonlinear map ϕ: $\mathbf{g} = \phi(\mathbf{y})$, and then applies PCA on the feature points $\mathbf{G}$, the assumption being that the data is more linear in feature space. By using the kernel trick the mapping ϕ never actually has to be computed. Instead, all computations can be done using the kernel function which indirectly computes the inner product of points $\mathbf{y}_i$ and $\mathbf{y}_j$ in feature space: $k(\mathbf{y}_i, \mathbf{y}_j) = \mathbf{g}_i^T \mathbf{g}_j$ by means of an equivalent function which does not explicitly use ϕ and may be much more efficient to compute[5]. The most common choice of kernel is the squared exponential[6] (cf. eq. 3.11): $k(\mathbf{y}_i, \mathbf{y}_j) = \exp(-\frac{1}{2\ell^2}|\mathbf{y}_i - \mathbf{y}_j|^2)$. With this choice of kernel, the mapping ϕ becomes smoother and more linear for increasing kernel width ℓ, i.e., kPCA becomes more similar to PCA for increasing ℓ (cf. Fig. 2.2(a-d)). The kPCA optimisation problem is convex and can be solved efficiently, but the kernel parameters (e.g. width ℓ) have to be chosen by hand or have to be optimised with respect to an independent criterion. Even though approximations for determining pre-images from latent space have been proposed (Kwok and Tsang, 2004; Bakir et al., 2004), getting pre-images

[4] The sum of the corresponding eigenvalues indicates the amount of explained variance

[5] Kernel functions here and covariance functions in Gaussian Processes are equivalent.

[6] Also sometimes called Gaussian or radial basis function kernel.



(a) PCA (b) probabilistic PCA (c) kPCA, $\ell = 7$ (d) kPCA, $\ell = 3$

(e) Isomap, $k = 6$ (f) Isomap, $k = 24$ (g) LLE, $k = 7$ (h) LLE, $k = 13$

(i) LE, $k = 7$, $\ell = 10$ (j) LE, $k = 24$, $\ell = 10$ (k) LE, $k = 24$, $\ell = 1$ (l) GPLVM-MDS

Figure 2.2: Latent spaces of non-generative methods for motion capture punches. Dark blue: high punch, light green: low punch, red: very low punch. x-axis: 1st latent dimension, y-axis: 2nd latent dimension. (e,f,l) have been rotated by 90° to better fit into figure.

is a problem, because the feature space is usually higher dimensional than the original data space and pre-images might simply not exist.

Isomap Isomap (Tenenbaum et al., 2000) is a special form of metric MDS (see Chapter 3 for a description) in which the dissimilarities between data points correspond to estimated geodesic distances along the assumed low-dimensional manifold. Geodesic distances are estimated by the shortest path in a graph which only connects data points to their k nearest neighbours. The usual MDS/PCA mechanism can then be used to determine the amount of variance explained by the resulting latent variables which may serve to decide the cardinality of the latent space or act as a heuristic for the choice of k. Note, however, that the variance refered to here is not the variance of the actual data, but rather a variance defined by the estimated geodesic distances. Therefore, if the geodesic distances are not estimated well, the estimate of the residual variances will not be good either. Fig. 2.2(e,f) shows examples for a 2D latent space on the mocap data with $k = 6$ and $k = 24$, respectively, corresponding to residual variances of $v_1 = 0.013, v_2 = 0.005$ and $v_1 = 0.093, v_2 = 0.019$ where v_1 is the residual variance not explained by the first latent variable and v_2 is the residual variance after the first and second latent variable have been taken into account.

LLE Locally Linear Embedding (Roweis and Saul, 2000) is a nonlinear DR method which aims to preserve the local linear structure of the data points. In particular, it determines the k nearest neighbours for each data point $\mathbf{y}_i$ and computes the weights of the linear combination of these neighbours such that the error between $\mathbf{y}_i$ and the result of the linear neighbour combination is minimised. The latent points are then the result of minimising the equivalent reconstruction error in latent space using the same weights. The important insight of the method is that this is equivalent to computing the M eigenvectors with the lowest nonzero eigenvalues of the matrix $(\mathbf{I}-\mathbf{W})^T(\mathbf{I}-\mathbf{W})$ where $\mathbf{I} \in \mathbb{R}^{N\times N}$ is the identity matrix and $\mathbf{W} \in \mathbb{R}^{N\times N}$ is the sparse matrix of weights. As, in contrast to PCA, MDS and Isomap, the lowest eigenvalues of a matrix need to be computed, LLE is more numerically unstable and produces widely varying results for changing k (Fig. 2.2(g,h)).

LE Laplacian Eigenmaps (Belkin and Niyogi, 2003) is another nonlinear DR method based on graph theory. It computes the k nearest neighbours for each data point and then minimises a weighted sum of distances between these neighbours in latent

space. To be precise, it minimises $\sum_{ij} w_{ij}|\mathbf{z}_i - \mathbf{z}_j|^2$ where the weights are determined as $w_{ij} = \exp(-\frac{1}{2\ell^2}|\mathbf{y}_i - \mathbf{y}_j|^2)$[7], if i and j are nearest neighbours, otherwise $w_{ij} = 0$. As the weights are large for close data points, LE focuses on preserving the local distances in latent space. Again, the optimal solution to this problem can be found by solving a (generalised) eigenvalue problem on a sparse matrix defined in terms of the weights where the eigenvectors with the lowest eigenvalues define the latent configuration. Compared to the previous methods LE has an additional, free parameter ℓ which controls the influence of the distances in data space on the distances in latent space (weights are almost equal for all nearest neighbours for very large ℓ and for very small ℓ weights rapidly decrease for more distant nearest neighbours). Some resulting latent spaces can be seen in Fig. 2.2(i-l).

For any given parameter setting the presented non-generative methods have in common that they define an optimisation for which the global optimal solution can be found rather efficiently. The example latent spaces in Fig. 2.2 show, however, that these solutions are highly dependent on the choice of the parameters and the methods leave that choice entirely up to the user. In this figure we also show a latent configuration computed by GPLVM-MDS (see Chapter 3) for comparison which does not require additional parameters, if the data is appropriately normalised, but which is harder to optimise.

2.2.2 Generative Methods

Generative methods define a probabilistic model based on latent variables which is usually learnt by maximising the likelihood of the observed data. Models often require further model selection choices, but offer the advantage that latent variables and model parameters can be optimised together using the same objective. However, the resulting optimisations are predominantly nonlinear and only local optima may be found.

We have already mentioned that PCA can also be derived from a linear, probabilistic model and that factor analysis (FA) is a generalisation of that model to differing amounts of noise on the different observed variables. While probabilistic PCA can still be solved using eigendecomposition of the covariance matrix, we need to employ, for example, the EM algorithm (Bishop, 2006, ch. 9) to find an optimal solution for FA. Independent Component Analysis (e.g. Hyvärinen and Oja, 2000) is also based on

[7] Other forms of decay, or even $w_{ij} = 1/k$ are also possible.

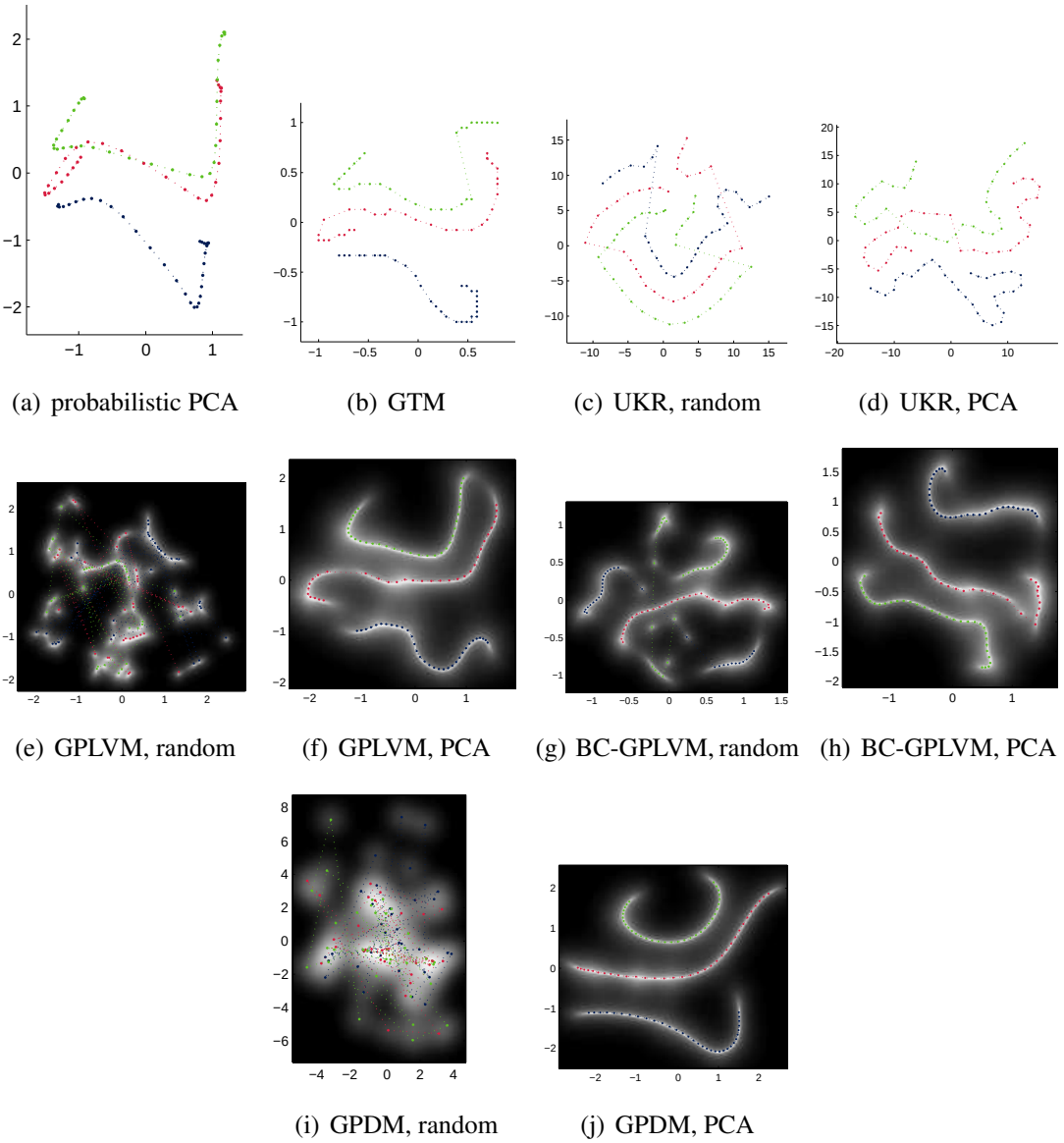
UKR, random	UKR, PCA	GPLVM, random	GPLVM, PCA
$0.636 \cdot 10^{-3}$	$0.167 \cdot 10^{-3}$	$0.0018 \cdot 10^{-3}$	$0.0003 \cdot 10^{-3}$
BC-GPLVM, random	BC-GPLVM, PCA	GPDM, random	GPDM, PCA
$0.0004 \cdot 10^{-3}$	$0.0002 \cdot 10^{-3}$	31	$0.0011 \cdot 10^{-3}$
GTM			
$8 \cdot 10^{-3}$			

Table 2.1: Comparison of nSSEs between data and its reconstruction from latent space for different generative methods and the results shown in Fig. 2.3.

the same linear model as PCA. It was not part of our comparisons, but is potentially worth investigating as it is optimising an objective very different from PCA.

GTM The generative topographic mapping (Bishop et al., 1998) is a probabilistic extension of the more heuristic self-organising map (Kohonen, 1982). The model consists of two components: a generalised linear regression model with radial basis functions (rbfs) whose centres are distributed on a uniform grid in latent space ($f(\mathbf{z};\mathbf{W}) = \mathbf{W}\phi(\mathbf{z})$) and a Gaussian conditional probability of the data given latent points and the regression model ($P(\mathbf{Y}|\mathbf{Z}, f(\cdot;\mathbf{W}))$). To obtain the data likelihood the latent points need to be integrated out, but this integration is not tractable in general. Therefore, a prior over latent points is defined which is a sum of delta functions again located at a (potentially different) regular grid in latent space, in effect selecting regular samples in latent space. Under this prior integration is equivalent to a sum over the samples and thus becomes tractable. The EM algorithm can then be used to efficiently estimate the parameters $\mathbf{W}$. The posterior over latent points given a data point $\mathbf{y}_i$, $P(\mathbf{z}|\mathbf{y}_i, \mathbf{W})$, is again a sum of delta functions, i.e., all the probability mass is distributed at the samples selected by the prior. The posterior mean may also lie between the samples as it is a weighted sum of them, but the GTM tends to assign most probability mass to one of them and then also the means follow the grid closely.

Computationally the GTM optimisation is very robust and efficient compared to other generative methods. However, it is highly dependent on the user's choice of parameters. For example, the number of rbfs in the regression model determines the quality of the generative mapping, but the main pitfall of the GTM are the latent samples defined by the prior. We clearly want as many as possible, but we quickly reach feasible memory limits when increasing their number and are usually confined to 2D



(a) probabilistic PCA (b) GTM (c) UKR, random (d) UKR, PCA

(e) GPLVM, random (f) GPLVM, PCA (g) BC-GPLVM, random (h) BC-GPLVM, PCA

(i) GPDM, random (j) GPDM, PCA

Figure 2.3: Latent spaces of generative methods for motion capture punches. Dark blue: high punch, light green: low punch, red: very low punch. x-axis: 1st latent dimension, y-axis: 2nd latent dimension. The shading in (e-j) visualises the confidence in the GPLVM predictions: $\log_{10}(1/\sigma_d^2)$ where σ_d^2 is from eq. (2.2).

latent spaces for reasonable grid densities. Fig. 2.3(b) shows a GTM latent space obtained with a grid of 10×10 rbfs with a width of 0.444 and latent samples on a grid of 40×40 [8]. To evaluate the quality of the regression model we compute the normalised sum of squared errors (nSSE)[9] between the data and its reconstruction from latent space, nSSE = $8 \cdot 10^{-3}$.

UKR Unsupervised Kernel Regression (UKR) (Meinicke et al., 2005) is the unsupervised counterpart to the Nadaraya-Watson regression estimator (Bishop, 2006, ch. 6.3). The regression model is a weighted sum of the data points where the weights correspond to normalised distances in some feature space defined by a kernel function. UKR generalises kernel regression by including the (unobserved) inputs into the optimisation problem. In order to control the complexity, or smoothness, respectively, of the latent manifold in observed space the optimisation is defined using leave-one-out cross-validation which can be implemented without additional computational cost. The resulting optimisation is highly nonlinear and is solved using gradient descent techniques on a suitable initialisation of the latent points. Model selection choices include the kernel and the loss function (Klanke and Ritter, 2007), but the initialisation of the latent points has the largest influence on the resulting latent configuration. In Fig. 2.3(c,d) we depict UKR results for a random initialisation (nSSE = $0.636 \cdot 10^{-3}$) and one using PCA (nSSE = $0.167 \cdot 10^{-3}$). The kernel was a Gaussian and the loss was the standard squared loss function with leave-one-out cross-validation.

LELVM/DRUR The Laplacian Eigenmaps Latent Variable Model (Carreira-Perpinan and Lu, 2007) adds out-of-sample and generative mappings to LE without changing the LE latent configuration. As this is done using kernel regression, the model is a special case of UKR in which the latent configuration is initialised with LE and no optimisation is executed. For the LELVM it is therefore critical to set the kernel parameters correctly which is entirely left to the user. Of course, it is also restricted to the use of LE as DR method.

Dimensionality reduction by unsupervised regression (Carreira-Perpinan and Lu, 2008) was suggested to overcome the shortcomings of the LELVM. Instead of regression based on kernel density estimation it defines out-of-sample and generative mappings as rbf networks in which the rbfs are centred at the data and latent points,

[8] We also used a Gaussian zero mean prior with variance 2.5 on the regression weights.

[9] $\text{nSSE}(\mathbf{X}, \hat{\mathbf{X}}) = \sum_{ij}(\hat{x}_{ij} - x_{ij})^2 / \|\mathbf{X}_c\|_F^2$ where $\mathbf{X}_c$ is $\mathbf{X}$ with centred columns

respectively. The network weights are then learnt together with the latent points to minimise the mapping errors of the rbf networks. Similar to an EM algorithm optimisation of network weights (efficient, global solution) is alternated with optimisation of latent points (computationally demanding, local minima), but the model is not probabilistic and therefore does not take the uncertainty of the latent representation into account. To avoid overfitting the objective function also contains regularisation terms for the network weights. The overall method has several free parameters that need to be set by the user (rbf widths, regularisation parameters), is computationally very demanding and has significant problems with local minima.

Autoencoders Linear autoassociative neural networks can also be used as a DR method and it can be shown that the optimal weights of a suitably defined multilayer perceptron correspond to the principal components of the data (Bishop, 2006, ch. 12.4.2). Consequently, PCA can be generalised to the nonlinear case by using a nonlinear autoassociative network with additional hidden layers (Kramer, 1991). These nonlinear networks are very hard to optimise and often get stuck in suboptimal local minima. Even though Hinton and Salakhutdinov (2006) proposed a method which alleviates this problem to some extent (note that a similar approach has already been used by Tatani and Nakamura (2003)), the user is still left with many design choices which can tune the network performance, but are difficult to interpret (e.g. number of hidden units).

2.2.2.1 GPLVM

The Gaussian Process Latent Variable Model (Lawrence, 2005) can be derived as a nonlinear extension to a dual formulation of probabilistic PCA (repeated in Section 3.1.2). In the resulting model each dimension is drawn independently from a common Gaussian Process (Rasmussen and Williams, 2006). The conditional density of the observed data then is a product of Gaussians with common parameters (for which it is the likelihood)

$$P(\mathbf{Y}|\mathbf{Z},\beta) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{y}_{:,d}|\mathbf{0},\mathbf{K}) \tag{2.1}$$

where $\mathbf{y}_{:,d} \in \mathbb{R}^N$ contains the data from the d-th observed variable, $\mathbf{0} \in \mathbb{R}^N$ is a vector of zeros and $\mathbf{K} \in \mathbb{R}^{N\times N}$ is a covariance matrix defined by the covariance function of the underlying GP as $K_{ij} = k(\mathbf{z}_i, \mathbf{z}_j)$ and β is a vector of covariance function parameters.

The GP model can be learnt by maximising the likelihood (2.1) with respect to the covariance parameters using a gradient descent algorithm. This optimisation is computationally very demanding as $\mathbf{K}$ needs to be inverted in every gradient step[10]. GPs are also known to run into local maxima of the likelihood when covariance parameters are learnt. Compared to a GP, the GPLVM adds another NM parameters due to the inclusion of $\mathbf{Z}$ to the optimisation of (2.1). This has the potential to aggravate local optima problems and makes initialisation of latent points $\mathbf{Z}$ and covariance parameters β even more important. Fig. 2.3(e,f) demonstrate these problems using PCA and a random initialisation of the latent points. The covariance function was the standard squared exponential (SE) with added white noise (eq. (3.19)) and the parameters were initialised as $\phi = 1, \ell = 1, \sigma^2 = 0.1$. 500 scaled conjugate gradient steps were taken. The nSSE between data and its reconstruction from latent space were nSSE = $0.0003 \cdot 10^{-3}$ for PCA initialisation and nSSE = $0.0018 \cdot 10^{-3}$ for random initialisation.

Despite the problems of optimising the GPLVM it is our method of choice, because it also inherits all benefits of a GP: it is extremely powerful and flexible, but the well-defined probabilistic framework automatically controls the complexity of the model during learning and therefore is less prone to overfitting compared to other methods when only a limited amount of data is available. Additionally, covariance parameters can typically be interpreted in terms of properties of the observed data which simplifies interpretation of results and can guide initialisation of the optimisation. Once the parameters of the GPLVM have been learnt, prediction from latent space to data space can be done efficiently using GP prediction. In particular, given an arbitrary point in latent space $\mathbf{z}^*$ the GP prediction is Gaussian distributed, $y_d^* \sim \mathcal{N}(\mu_d, \sigma_d^2)$, with parameters

$$\begin{aligned} \mu_d &= \mathbf{k}^{*T} \mathbf{K}^{-1} \mathbf{y}_{:,d} \\ \sigma_d^2 &= k^* - \mathbf{k}^{*T} \mathbf{K}^{-1} \mathbf{k}^* \end{aligned} \tag{2.2}$$

where $\mathbf{K}^{-1}$ is the inverted covariance matrix which can be precomputed, because it is independent of the test point, $\mathbf{k}^*$ is a vector with components $(\mathbf{k}^*)_i = k(\mathbf{z}^*, \mathbf{z}_i)$ and $k^* = k(\mathbf{z}^*, \mathbf{z}^*)$. Note that σ_d^2 is equal for all d.

Finally, the probabilistic model allows to flexibly include prior information about the latent points by introduction of a corresponding prior. In the next section we summarise extensions to the GPLVM that have been proposed in the literature, some of which are based on this idea.

[10] Although more efficient approximations have been suggested, their success is limited.

2.2.2.2 Variants of the GPLVM

Equation (2.1) defines a likelihood for all the parameters of the GPLVM. It is therefore straightforward to loosen some of the assumptions of the GPLVM and simply include the additional parameters into the optimisation of (2.1). For example, the original formulation of the GPLVM assumes that the scale of the observed variables is equal. So it has been proposed (Grochow et al., 2004) to add output scale variables (one for each dimension) to the model. Similarly, different covariance parameters could be allowed for different output variables, but this would, in effect, change the model from one common, underlying GP to several underlying GPs and we expect that this makes learning of the GPLVM impossible (cf. the results on the variability of covariance estimates in Chapter 3.3). For our small data sets we believe that the original GPLVM optimisation already suffers from too many degrees of freedom and we prefer to normalise the data appropriately rather than including more free parameters into the optimisation.

An important insight used for many of the non-generative methods in Section 2.2.1 is that the local distances need to be preserved, especially when the data is highly nonlinear. In other words, if two points are close in data space, we also want them close in latent space while large distances in data space, which may be distorted through nonlinear interactions, should not be enforced in latent space to the same extent. The GPLVM, however, only ensures that close points in latent space are close in data space, i.e. it preserves local distances only in the reverse direction. Consequently, Lawrence and Quinonero-Candela (2006) have introduced the back-constrained **(BC-)GPLVM** which, instead of optimising the latent points $\mathbf{Z}$ directly, represents them as a smooth parametric function of the data and optimises the parameters of that function. As parametric functions, for example, the multi-layer perceptron (MLP) and kernel based regression (KBR[11]) have been proposed. While we had great problems successfully learning the MLP, the KBR back constraints are often successful in producing smoothed latent configurations and sometimes also reduce the dependence on the initialisation (see Fig. 2.3(g,h), width of rbf kernel was $\ell = 2.236$, nSSE = $0.0002 \cdot 10^{-3}$ for PCA and nSSE = $0.0004 \cdot 10^{-3}$ for random initialisation).

A more direct way of biasing the latent configuration towards exhibiting a particular property is to introduce a prior over latent points. Wang et al. (2008) have suggested a dynamics prior to take the sequential nature of, for example, motion data sets into account. They derive a nonparametric, autoregressive model, the Gaussian Process

[11] Because we use the standard rbf kernel, this is equivalent to a radial basis function network in which the rbfs are centred on the data points.

Dynamical Model (**GPDM**). Even though it is derived in the same way as GPs, the resulting distribution over latent points is not Gaussian and that makes, for example, prediction of sequences complicated. However, it is still simple to include the prior into the gradient based optimisation of eq. (2.1) as it is just another additive term in the resulting log-likelihood. Like the original covariance parameters the covariance parameters of the GPDM can also be included in the optimisation, but this exasperates the local optima problems and often does not lead to expected results. We were more successful fixing GPDM parameters by hand. In particular we used a compound covariance function consisting of a SE, a linear and a noise part where the lengthscale of the SE was $\ell = 2.236$ the variance of the SE and linear part were equal at $\phi = 0.01$ and the noise variance was $\sigma^2 = 10^{-6}$. The GPDM was defined on the differences of the latent points rather than the points themselves. Fig. 2.3(i,j) depicts resulting GPDM latent spaces for PCA (nSSE = $0.0011 \cdot 10^{-3}$) and random (nSSE = 31) initialisations. The GPDM prior successfully smoothed the initialisation, when it already roughly contained the right sequential structure, but it did not help to overcome local optima issues.

Other priors have been suggested. Lawrence and Moore (2007) demonstrated the use of hierarchical priors for motion modelling. These introduce additional latent variables which are organised in a hierarchy chosen by the user. However, the hierarchy is difficult to learn and great care has to be taken to avoid overfitting to the kind of small data sets we consider. They also suggested a time-indexed GP dynamics which allows forking of trajectories in latent space, but has the disadvantage of being tied to one particular roll-out of time. Urtasun and Darrell (2007) devised a prior which helps separating data points according to class membership in classification tasks.

Urtasun et al. (2008) have adapted the GPLVM particularly for periodic movements, like walking and running, with several cycles which may vary in style of execution. Similar to our suggestion for discrete movements in Chapter 4 they aim to introduce more structure in latent space. In particular they constrain latent variables to be periodic and introduce an additional term in the optimisation of the log-likelihood which corresponds to the LLE objective of reproducing locally linear reconstructions of the data in latent space. Additionally "transition" points are identified which are forced to lie close in latent space. The resulting latent spaces indeed exhibit the desired structure, but this is achieved through outsourcing the design of the latent space to the user. A study with equivalent aims was presented by Wang et al. (2007) who defined a Multifactor GPLVM together with a very strict circular dynamics in latent space. Their

idea was to let different factors represent different aspects of the data (typically movement style, or subject of movement). However, each factor has to be selected by hand and introduces additional latent variables which are nonlinearly related to the existing variables. Then not only learning, but already initialisation becomes a problem.

2.3 Evaluating the Quality of Motion Interpolation in Latent Spaces

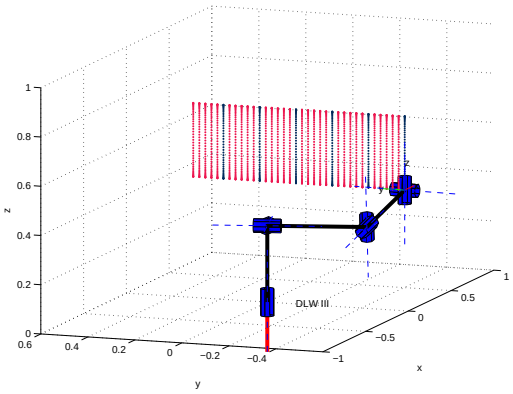
If we were only interested in visualisation of a high-dimensional data set, any of the latent spaces in Figs. 2.2 and 2.3 may be sufficient, but then we still were not able to say whether any one was better than another (we may exclude some as they do not make sense on a very superficial level). So the purpose of DR is critical for the evaluation of its quality and we can only evaluate DR in the context of the application that it is embedded in. Some applications would not be computationally feasible without DR, for others DR is intended to improve performance in terms of accuracy or efficiency. Of course new methods are published together with a claim for such performance improvements, but often i) evaluations are only done on particular, restricted data sets, ii) it is not clear whether the performance improvement is due to the dimensionality reduction or another part of the suggested system, or iii) no quantitative evaluation is done at all. The domain of motion synthesis from human motion capture is particularly prone to insufficient evaluations, because we do not know the ground truth in these applications. Therefore we cannot conclusively say when dimensionality reduced motion representations are beneficial and when they are, why this is the case.

In order to have a more systematic evaluation, we consider robotic motion data, for which we have complete control of the underlying generating principle and which is sufficiently complex to exclude trivial solutions. If dimensionality reduction succeeds, it is very easy to generate new motions which follow these principles. Consequently we use interpolation of motions to evaluate DR results which additionally allows us to compare to the case without DR.

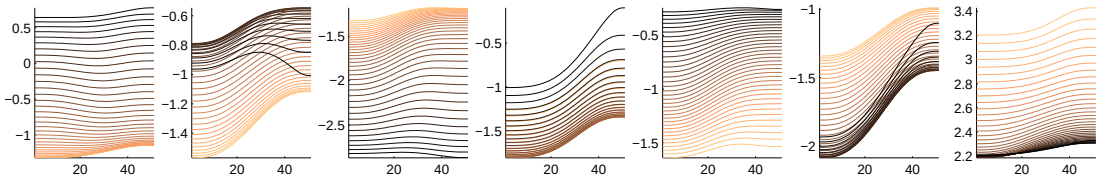
2.3.1 Robotic Motion Data (DLR arm)

We use the kinematics of the 7 DOF DLR Light-Weight Robot III arm (Fig. 2.4(a)) for our experiments. Setting the position and orientation of its end-point constrains 6 of the 7 DOF. The 7th DOF is resolved as described in Dahm and Joublin (1997) by

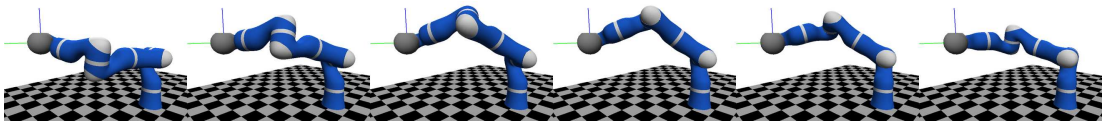
(a) DLR arm with Schunk hand



(b) position set in simulation



(c) 36 trajectories of position-set in joints 1 to 7 (in this order). x: time, y: joint angle



(d) DLR arm poses with fixed end-point but varying α (2.2, 2.8, 3.4, 4, 4.6, 5.2)

Figure 2.4: (a) DLR Light-Weight Robot III with attached Schunk hand. (b) 36 trajectories of position set shown in Cartesian space of the robot (task space). Colour coding corresponds to interpolation width 5 (column 5 in Fig. 2.5), i.e., blue points: data trajectories, red points: trajectories which need to be interpolated. (c) Joint angle trajectories for the position set. (d) Visualisation of the effect of α on the robot posture.

setting a "redundancy angle" α (see Fig. 2.4(d) for visualisation). For our first data set (α-set) we define a single, straight line, upward movement of the arm end-point with fixed orientation and let α vary in steps of 0.1 from 2.2 to 5.7 yielding 36 different arm movements. For the second data set (position-set) we fix $\alpha = 4.2$ and vary the position of the upward movement along a line from $[0.48, -0.41]$ to $[0.48, 0.49]$ in the robot's base plane such that we also obtain 36 different arm movements (see Fig. 2.4(b,c)). Within each data set any pose can be uniquely identified with 2 parameters: the height of the end-point and either α or the position along the line (y-value of base plane). Furthermore all poses are smoothly connected along those dimensions. Therefore these data sets are inherently 2 dimensional and ideally dimensionality reduction would align a 2 dimensional latent space along these directions.

We evaluate interpolation quality of whole trajectories for increasing distances between the trajectories used for interpolation. This is done by increasing the number of trajectories left out for evaluation. We start by leaving out every 2nd trajectory as targets for interpolation and end by leaving out all 34 trajectories between the 1st and last trajectories in the data set. We call the number of left out trajectories the "interpolation width". The data sets for each interpolation width were centred, but the standard deviation was left unchanged[12]. A generated trajectory is evaluated against its original with respect to the average root mean squared error (RMSE) between corresponding poses. This can be computed for the joints (in joint space) and for the position of the robot end-point (in task space). Furthermore we define a trajectory to be successfully interpolated, if the task space RMSE between interpolated and original poses is smaller than 0.0032 and the RMSE between corresponding α-values and its standard deviation is smaller than 0.01. These criteria ensure that the trajectory exhibits the correct values along the principled directions identified above.

2.3.2 DR Method Details

We compare interpolation directly in joint space against interpolation in latent spaces resulting from PCA, the GTM, the GPLVM, the BC-GPLVM, the GPDM and UKR. For the GTM we chose a grid of 10×10 rbfs with width set to twice the distance between two neighbouring rbf centres, sample points were placed at the nodes of a 30×30 grid, we included a mild Gaussian prior ($\sigma^2 = 2.5$) on the regression weights and optimised for maximally 100 steps. The GPLVM had the SE+noise+bias covariance

[12] The standard deviations of the full data set for the 7 joints were: [0.561, 0.292, 0.456, 0.281, 0.406, 0.305, 0.375].

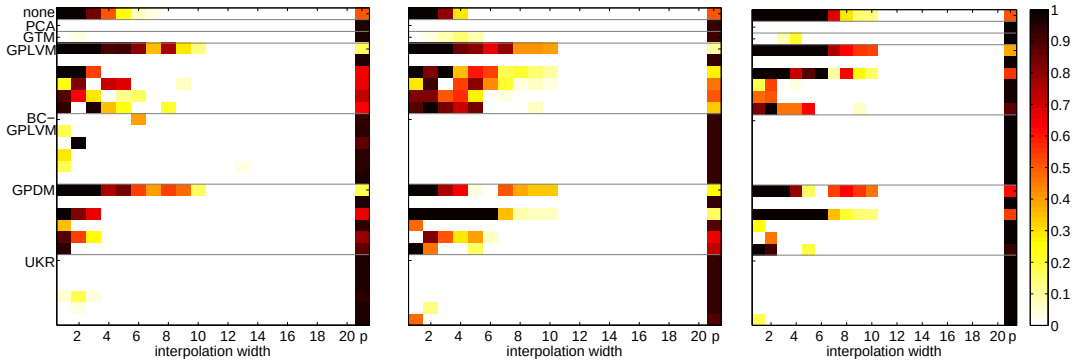
function where the SE+noise part is equal to eq. 3.19 and the bias allows for a change in mean common to all data dimensions. The covariance parameters were initialised as $\ell = 1, \phi = 1, \sigma^2 = 0.135$ and variance of the bias term also 0.135. We included a standard normal prior on the latent points. The optimisation was run for 500 steps of conjugate gradient descent. The BC-GPLVM and GPDM inherited the GPLVM parameters. The BC-GPLVM had KBR back constraints with rbf kernels with width $\ell = 3.905$ and their weights were initialised at random. The GPDM was again defined on the differences between latent points and had a SE+linear+noise covariance function with $\ell = 3.905$ and variances set as above. For UKR we also use a standard rbf kernel with squared loss function and leave-one-out cross-validation.

For the GPLVM approaches und UKR it is required to initialise the latent points for optimisation. We used 6 different initialisations. 1) Ad-hoc parallel, diagonal lines. The points in the n-th line corresponded to the points in the n-th data trajectory. The lines were arranged with equal distance to each other such that the length of the lines was 10 times larger than the distance to the next line. This initialisation therefore already incorporates a large amount of prior information and should be seen as a competitive baseline. 2) Uniformly distributed random points. 3) Probabilistic PCA. 4) Isomap with $k = 12$. 5) LLE with $k = 12$. 6) LE with $k = 12$ and $\ell = 0.1$.

2.3.3 Results

Our analysis is centred around the question: "Does Dimensionality Reduction Improve the Quality of Motion Interpolation?" Consequently, we evaluate the interpolation quality directly in joint space and compare it to interpolation in latent spaces resulting from dimensionality reduction. More specifically we interpolate in the latent space, map to joint space using the generative mapping of the DR method and compare the resulting joint space trajectories with the originals that have been left out when doing the dimensionality reduction. The interpolation itself is done pose by pose for any desired trajectory from corresponding poses of the given trajectories (the naive interpolation is done in the same way in joint space rather than in latent space). Either linear or spline interpolation is used. In addition to the investigation into quality improvement, these experiments are designed to give an insight into how variable DR results are for different choices of data sets and parameter settings.

In Fig. 2.5 we see that overall results are roughly similar for the two data sets. First we note that PCA has no successful interpolations at all even though the two-

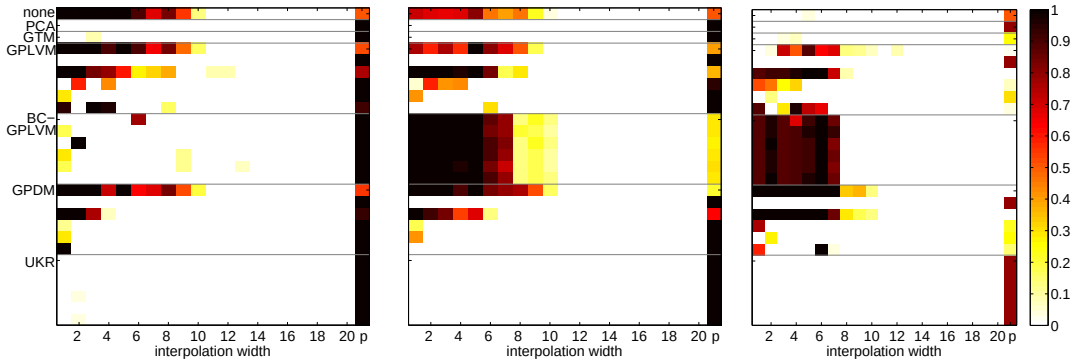


(a) position-set, linear (b) α-set, linear (c) α-set, spline

Figure 2.5: Ratio of successful interpolations for the two data sets. x-axis: interpolation width, y-axis: dimensionality reduction method. Values range from 0 (white, no successful interpolations) to 1 (black, all successful). For interpolation widths > 20 (not shown) also all ratios $= 0$. First line: naive, joint space interpolation. For GPLVM variants and UKR 6 different initialisations of latent points are tested (shown in this order): ad-hoc parallel lines, random, PCA, Isomap, LLE, Laplacian Eigenmaps. Last column: p-value for hypothesis 'The mean fraction of successful interpolations across interpolation widths is smaller or equal to the corresponding mean of the naive interpolation' (for small p we accept with high confidence that dimensionality reduction is advantageous, we used a two-sample t-test).

dimensional PCA space already captures 97% or 94% of the data variance. Also UKR and GTM fail to produce successful interpolations. This might not necessarily be a problem of the latent representation, but could be due to the weakness of the generative mapping. For the GPLVM approaches we see that results are highly dependent on the chosen initialisation and data set. PCA initialisation, which is standard in the literature, is consistently outperformed by initialisation with ad-hoc, parallel lines. Note that no DR method significantly outperforms spline interpolation in joint space (see Fig. 2.5(c), Fig. 2.6(a)). This is because in the given setting this form of nonlinear interpolation already performs close to the limit of what can be achieved. The lowest p-value of 0.05 (see Fig. 2.5 for explanation) is achieved for linear interpolation on the α-set with the standard GPLVM and ad-hoc lines initialisation. This finding suggests that, with a suitable choice of parameters and initialisation, DR can make a nonlinear motion interpolation problem into a linear one.

These experiments help to understand whether and how well DR methods can un-



(a) no noise, position-set (b) noise, position-set (c) noise, α-set

Figure 2.6: Ratio of successful spline interpolations for noisy data sets. Gaussian noise with standard deviation equal to 1/100 of the data standard deviation. Interpretation as in Fig. 2.5

cover principles underlying movement data, but, because the data is noise free, they neglect an important feature of these methods. Fig. 2.6 shows results when a small amount of noise was added to the data (1/100 of the standard deviation of the data in each joint). In the presence of noise joint space interpolation produces fewer successful interpolations, because the variance introduced by the noise is sufficient to increase the pose errors such that interpolated trajectories in joint space do not fulfil the success criteria anymore even though the general shape and position of the interpolated trajectories roughly fit the data. The GPLVM approaches, on the other hand, smooth out some of the noise variance and therefore maintain their interpolation quality to a greater extent. Furthermore, the BC-GPLVM only produces successful interpolations in the presence of noise, because the noise prevents premature convergence of the BC-GPLVM optimisation that is occurring otherwise. These experiments indicate that the GPDM is the most robust of the tested methods, if a good initialisation is given (GPDM with lines initialisation gives a higher fraction of successful interpolations for more interpolation widths than any other method and initialisation). This is no surprise, because it is the only tested method that models temporal coherence of the data which is used to smooth the data over time. However, this is only beneficial, if the initial guess for the latent configurations reflects the temporal structure of the data reasonably well.

2.4 Discussion

In this chapter we have introduced several nonlinear DR methods. With the help of a motion capture data set typical for our learning by demonstration setting we have demonstrated that latent configurations resulting from nonlinear DR are not only highly variable across different methods, but also within methods across different parameter settings. We attribute this variance to the general problem that nonlinear DR is ill-posed: given a high-dimensional data set there are infinitely many combinations of latent spaces and corresponding nonlinear mappings that could have generated the data. Non-generative methods therefore predominantly constrain latent spaces to reproduce local distances observed in the data. This necessitates a dense sampling of data points so that local distances are well approximated. However, the motion data sets that we consider for learning by demonstration often do not follow this assumption: while data points are densely sampled within a motion (recorded as joint trajectories) they are often more sparse across recorded movements.

In an experiment with robotic movement data we have addressed this problem by investigating the quality of interpolations in latent spaces as the distances between movements increases. As expected, we observed a drop in interpolation quality for all methods. Comparing interpolation directly in joint space with interpolation in latent space after dimensionality reduction we conclude

- Interpolation in latent space could not outperform nonlinear interpolation in joint space, but it could achieve comparable accuracy (with the best initialisations). Consequently, dimensionality reduction potentially offers computational advantages as interpolation may be done in fewer dimensions.

- Interpolation in latent space was more robust against noise. Therefore, dimensionality reduction can denoise data.

- With the right initialisation (resulting in a suitable latent space) linear interpolation in latent space was as good as spline interpolation in joint space and clearly outperformed linear interpolation in joint space. We conclude that dimensionality reduction can turn a nonlinear problem into a linear one by absorbing the nonlinearity in the generative mapping. We exploit this property in Chapter 4 to extend the applicability of dynamic movement primitives.

Of the tested generative methods only those based on the GPLVM allowed accurate interpolations. We believe that this is due to the very powerful GP mappings. The re-

dundancy between latent representation and generative mapping in a nonlinear setting means that, even when the latent representation does not correspond to the true, underlying configuration of points, nonlinear mappings exist which reconstruct the observed data. The GP framework, as a nonparametric model, lets us always easily determine a GP mapping which reconstructs the data very well. At the same time the use of a SE covariance function and the automatic regularisation introduced by the probabilistic model make sure that the data is smoothly interpolated. However, also a GP cannot magically recover the desired mapping when data points are too far from each other, or the latent configuration is too different from the true one (and thus requiring, for example, jumps in the mapping to connect neighbouring points). Furthermore, the gradient based optimisation of the GPLVM is in general not able to find the true latent configuration, or even just a good approximation, because the optimisation landscape is too complex with many local optima. In the next chapter we address this issue by proposing a new initialisation for the GPLVM which is directly derived from it and does not require any additional parameters. Even with an improved initialisation GPLVM latent spaces sometimes do not fulfil the requirements with respect to a chosen application. We present motion interpolation with dynamic movement primitives as an example in Chapter 4 and show there how we can incorporate additional prior information into the GPLVM to improve results in that application. Finally, we demonstrate in Chapter 5 the use of the GPLVM embedded into reinforcement learning as a procedure to nonlinearly interpolate between only a few demonstrated postures while simultaneously providing an efficient state representation. This allows us for the first time to learn accurate, full-body movements for a humanoid robot from demonstration.

Chapter 3

GPLVM-MDS: Relating the GPLVM to Metric MDS

In this chapter we explore the GPLVM from a theoretical perspective. In Section 3.1 we show that we can replace the optimisation of the GPLVM likelihood with a metric Multidimensional Scaling (MDS) problem when the GPLVM covariance function is isotropic. In the ideal case we can, therefore, replace a difficult, nonlinear optimisation with a simple eigenvalue problem. However, as the underlying covariance matrix is only approximated by the sample covariance matrix, an iterative MDS procedure has to be used. On synthetic data we demonstrate in Sections 3.3 and 3.4 that the MDS result is a better initialisation for the GPLVM than PCA and also compares favourably to Isomap. In Section 3.5 we show that on real-world, motion capture data this advantage is maintained, but to a lesser extent, and conjecture that this is due to a discrepancy between data and model.

3.1 Relating MDS, PCA and the GPLVM

In this section we derive the relationship between MDS and the GPLVM. First, we recapitulate the close ties between classical MDS and PCA (Section 3.1.1) and the derivation of the GPLVM as a nonlinear extension of probabilistic PCA (Section 3.1.2). We close the circle by showing that MDS can be used to find a GPLVM solution under the condition that the underlying GP has an isotropic covariance function (Section 3.1.3).

3.1.1 Relating Classical MDS and PCA

Here we restrict ourselves to the description of classical MDS. For a more complete treatment of MDS see, e.g., Cox and Cox (2000). In classical MDS we are given a matrix of distances $\mathbf{D} \in \mathbb{R}^{N\times N}$ between unknown (latent) points in a Euclidean space $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_N)^T, \mathbf{y}_i \in \mathbb{R}^D$. Our aim then is to reconstruct $\mathbf{Y}$ directly from the distances d_{ij}. In particular, we first need to find the inner product matrix $\mathbf{B}$ with $[\mathbf{B}]_{ij} = b_{ij} = \mathbf{y}_i^T \mathbf{y}_j$. Given that $\mathbf{Y}$ is centred [1] and $d_{ij}^2 = (\mathbf{y}_i - \mathbf{y}_j)^T(\mathbf{y}_i - \mathbf{y}_j)$ it can be shown (Cox and Cox, 2000, ch. 2.2.1) that

$$\mathbf{B} = \mathbf{HAH} \tag{3.1}$$

where $[\mathbf{A}]_{ij} = -\frac{1}{2}d_{ij}^2$ and $\mathbf{H}$ is the centring matrix $\mathbf{H} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T$ with $\mathbf{1} = (1, \ldots, 1)^T$ a vector of N ones. $\mathbf{B}$ is symmetric, positive semi-definite and can also be written in terms of its eigendecomposition

$$\mathbf{B} = \mathbf{V}\Gamma\mathbf{V}^T$$

where $\mathbf{V}$ is the matrix of unit eigenvectors and $\mathrm{diag}(\Gamma)$ are the corresponding eigenvalues. By noting that also $\mathbf{B} = \mathbf{YY}^T$ it is clear then that

$$\mathbf{Y} = \mathbf{V}\Gamma^{\frac{1}{2}}\mathbf{R}^T \tag{3.2}$$

where $\Gamma^{\frac{1}{2}} = \mathrm{diag}(\gamma_1^{\frac{1}{2}}, \ldots, \gamma_N^{\frac{1}{2}})$ and $\mathbf{R}$ is an arbitrary rotation matrix (this is the singular value decomposition of $\mathbf{Y}$). The solution, therefore, is invariant to rotation of $\mathbf{Y}$. It is also invariant with respect to reflection in the origin, because of the arbitrary sign of the eigenvectors.

For reducing the dimensionality of the latent positions we note that $\mathbf{y}_i = \Gamma^{\frac{1}{2}}\mathbf{v}_{i,:}^T$ and therefore

$$d_{ij}^2 = (\mathbf{y}_i - \mathbf{y}_j)^T(\mathbf{y}_i - \mathbf{y}_j) = \sum_{n=1}^{N} \gamma_n (v_{in} - v_{jn})^2.$$

[1] The distances are invariant to a change of mean of the latent positions. We therefore choose their mean to be $\mathbf{0}$.

To approximate $\mathbf{Y}$ with a lower dimensional representation $\mathbf{Z}$ with dimension $M < D$ we, thus, should choose the eigenvectors with the M largest eigenvalues. It can then be shown that the distances $\hat{d}_{ij}$ between points in $\mathbf{Z}$ minimise $\sum_{ij}(d_{ij}^2 - \hat{d}_{ij}^2)$.

From a very similar point of view we can derive PCA (Bishop, 2006, ch. 12.1). There, our aim is to represent the data set $\acute{\mathbf{Y}} = (\acute{\mathbf{y}}_1, \ldots, \acute{\mathbf{y}}_N)^T, \acute{\mathbf{y}}_n \in \mathbb{R}^D$ in an alternative, orthonormal basis of dimensionality $M < D$ such that the average squared distance between points $\acute{\mathbf{y}}_n$ in the original space and their reconstruction from the basis representation $\tilde{\mathbf{y}}_n$

$$J = \frac{1}{N}\sum_{n=1}^{N} \|\acute{\mathbf{y}}_n - \tilde{\mathbf{y}}_n\|^2 \tag{3.3}$$

is minimised. In particular, we have

$$\tilde{\mathbf{y}}_n = \sum_{d=1}^{M} z_{nd}\mathbf{u}_d + \sum_{d=M+1}^{D} b_d \mathbf{u}_d$$

where $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_D)$ is the matrix of orthonormal basis vectors, $\mathbf{Z}$ is the representation of $\acute{\mathbf{Y}}$ in the space spanned by the first M basis vectors of $\mathbf{U}$ and $\mathbf{b}$ are biases in the remaining dimensions. Setting derivatives of eq. (3.3) with respect to $\mathbf{Z}$ and $\mathbf{b}$ to zero gives $z_{nd} = \acute{\mathbf{y}}_n^T \mathbf{u}_d$ and $b_d = \bar{\mathbf{y}}^T \mathbf{u_d}$ where $\bar{\mathbf{y}} = \frac{1}{N}\sum_n \acute{\mathbf{y}}_n$ is the mean of the data. If we note that $\acute{\mathbf{y}}_n = \sum_{d=1}^{D}(\acute{\mathbf{y}}_n^T \mathbf{u}_d)\mathbf{u}_d$ and plug these results into $\acute{\mathbf{y}}_n - \tilde{\mathbf{y}}_n$, we get

$$\acute{\mathbf{y}}_n - \tilde{\mathbf{y}}_n = \sum_{d=M+1}^{D} \left[(\acute{\mathbf{y}}_n - \bar{\mathbf{y}}_n)^T \mathbf{u}_d\right] \mathbf{u}_d$$

and see that $\acute{\mathbf{y}}_n - \tilde{\mathbf{y}}_n$ lies in the space orthogonal to the subspace spanned by the first M basis vectors. The optimal representation of $\acute{\mathbf{Y}}$ in the first M components of the basis defined by $\mathbf{U}$, therefore, is the orthogonal projection of $\acute{\mathbf{Y}}$ onto that space. Finally, we can reformulate J as a function of only $\mathbf{U}$

$$J = \frac{1}{N}\sum_{n=1}^{N}\sum_{d=M+1}^{D} \left(\acute{\mathbf{y}}_n^T \mathbf{u}_d - \bar{\mathbf{y}}^T \mathbf{u}_d\right)^2 = \sum_{d=M+1}^{D} \mathbf{u}_d^T \mathbf{S}_D \mathbf{u}_d \tag{3.4}$$

where

$$\mathbf{S}_D = \frac{1}{N}\sum_{n=1}^{N}(\acute{\mathbf{y}}_n - \bar{\mathbf{y}})(\acute{\mathbf{y}}_n - \bar{\mathbf{y}})^T = \frac{1}{N}(\mathbf{H}\acute{\mathbf{Y}})^T(\mathbf{H}\acute{\mathbf{Y}}) = \frac{1}{N}\mathbf{Y}\mathbf{Y}^T$$

is the sample covariance matrix of the centred data variables. Defining a Lagrange function with the orthonormality constraint of $\mathbf{U}$ and setting its derivative with respect to $\mathbf{u}_d$ to 0 we find

$$\mathbf{S}_D \mathbf{u}_d = \lambda_d \mathbf{u}_d.$$

Therefore basis vectors $\mathbf{u}_d$ must be the eigenvectors of the covariance matrix $\mathbf{S}_D$ and $J = \sum_{d=M+1}^{D} \lambda_d$. In order to minimise J we then need to select the eigenvectors of $\mathbf{S}_D$ with the M largest eigenvalues as our basis. These are called the principal components.

At this point we recognise that classical MDS and PCA are equivalent (as already noted in the original publication Gower (1966)). While in MDS we select the M largest eigenvalues of the inner product matrix $\mathbf{B} = \mathbf{Y}\mathbf{Y}^T$, in PCA we select the M largest eigenvalues of the sample covariance matrix $\mathbf{S}_D$. To see that the solutions are equivalent, we note that $\mathbf{S}_D = \frac{1}{N}\mathbf{Y}^T\mathbf{Y}$ where $\mathbf{Y}$ is centred (as assumed for MDS). We then have

$$\begin{aligned} \tfrac{1}{N}\mathbf{Y}^T\mathbf{Y}\mathbf{u}_d &= \lambda_d \mathbf{u}_d \\ \mathbf{Y}\mathbf{Y}^T\mathbf{Y}\mathbf{u}_d &= N\lambda_d \mathbf{Y}\mathbf{u}_d \\ \mathbf{Y}\mathbf{Y}^T\hat{\mathbf{v}}_d &= \gamma_d \hat{\mathbf{v}}_d \end{aligned}$$

and see that the eigenvalues of MDS and PCA are related as $\gamma_d = N\lambda_d$ (order remains). Note that the eigenvectors $\hat{\mathbf{v}}_d$ need not have unit length. The correct normalisation is

$$\|\hat{\mathbf{v}}_d\| = \sqrt{\mathbf{u}_d^T\mathbf{Y}^T\mathbf{Y}\mathbf{u}_d} = \sqrt{N\lambda_d}$$

Consequently, we also recognise that the resulting low-dimensional representations of PCA and MDS are equal:

$$\mathbf{z}_{:,d}^{PCA} = \mathbf{Y}\mathbf{u}_d = \hat{\mathbf{v}}_d = (N\lambda_d)^{\frac{1}{2}}\mathbf{v}_d = \gamma_d^{\frac{1}{2}}\mathbf{v}_d = \mathbf{z}_{:,d}^{MDS}.$$

This result clearly only applies to classical MDS in which we know that the given dissimilarities are true, Euclidean distances. In this case, however, we can now use the duality between the PCA and MDS formulations to compute results more efficiently depending on whether $\mathbf{S}_D \in \mathbb{R}^{D\times D}$ or $\mathbf{B} \in \mathbb{R}^{N\times N}$ is the smaller matrix. Next, we see that from the dual of a probabilistic formulation of PCA we can derive the GPLVM.

3.1.2 Relating probabilistic PCA and the GPLVM

PCA can also be derived from a generative, probabilistic model. In this model the data $\mathbf{y}$ is a linear function of the latent variables $\mathbf{z}$ plus added Gaussian noise

$$\mathbf{y}_n = \mathbf{W}\mathbf{z}_n + \varepsilon_n \qquad \mathbf{z}_n \sim \mathcal{N}(\mathbf{0},\mathbf{I}) \qquad \varepsilon_n \sim \mathcal{N}(\mathbf{0},\sigma^2\mathbf{I}). \tag{3.5}$$

Therefore the conditional distribution of $\mathbf{y}_n$ given $\mathbf{z}_n$ is

$$p(\mathbf{y}_n|\mathbf{z}_n) = \mathcal{N}(\mathbf{y}_n|\mathbf{W}\mathbf{z}_n,\sigma^2\mathbf{I}).$$

In the standard formulation of probabilistic PCA we define a Gaussian prior over $\mathbf{z}_n$, integrate out $\mathbf{z}_n$ and maximise the resulting likelihood with respect to the parameters of the model, $\mathbf{W}$ and σ. It can then be shown analytically that the maximum likelihood solution for $\mathbf{W}$ are the eigenvectors of the sample covariance matrix $\mathbf{S}_D$ as before (Tipping and Bishop, 1999)[2]. Lawrence (2005) alternatively derived the dual formulation of PCA by defining a prior over $\mathbf{W}$, integrating out $\mathbf{W}$ and maximising the likelihood of the latent positions $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_N)^T$. In particular, we define

$$p(\mathbf{W}) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{w}_d|\mathbf{0}, \mathbf{I})$$

$$p(\mathbf{Y}|\mathbf{Z}, \mathbf{W}, \mu, \sigma) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{y}_{:,d}|\mathbf{Z}\mathbf{w}_d, \sigma^2\mathbf{I}).$$

Note that the interpretation of the model has changed slightly. While the likelihood term is the same as in standard probabilistic PCA, the prior over latent positions $\mathbf{Z}$ has been replaced with a prior over the parameters $\mathbf{W}$. As a result, the assumption that data points are independent has been dropped (more about this below) and the model can now also be written as

$$\mathbf{y}_{:,d} = \mathbf{Z}\mathbf{w}_d + \varepsilon_d \qquad \mathbf{w}_d \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \qquad \varepsilon_d \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}) \tag{3.6}$$

where $\mathbf{y}_{:,d} \in \mathbb{R}^N$ is the d-th column of $\mathbf{Y}$ containing observations of variable d for all data points, $\mathbf{w}_d = \mathbf{w}_{d,:}^T \in \mathbb{R}^M$ contains the elements of the d-th row of matrix $\mathbf{W}$ of the original formulation and $\varepsilon_d \in \mathbb{R}^N$ is independent, Gaussian noise. Now we can integrate out $\mathbf{W}$ and get the likelihood for the latent positions and the variance of the noise

$$p(\mathbf{Y}|\mathbf{Z}, \sigma^2) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{y}_{:,d}|\mathbf{0}, \mathbf{Z}\mathbf{Z}^T + \sigma^2\mathbf{I}) \tag{3.7}$$

We want to maximise the log-likelihood which is

$$\begin{aligned} L &= -\frac{DN}{2}\log 2\pi - \frac{D}{2}\log|\mathbf{K}| - \frac{1}{2}\mathrm{tr}(\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^T) \\ &= -\frac{DN}{2}\log 2\pi - \frac{D}{2}\log|\mathbf{K}| - \frac{D}{2}\mathrm{tr}(\mathbf{K}^{-1}\mathbf{S}_N) \end{aligned} \tag{3.8}$$

where we have defined the kernel or covariance matrix $\mathbf{K} = \mathbf{Z}\mathbf{Z}^T + \sigma^2\mathbf{I}$ and the normalised inner product matrix of the data points $\mathbf{S}_N = \frac{1}{D}\mathbf{Y}\mathbf{Y}^T$ [3]. To solve for $\mathbf{Z}$ we set

[2] the ML solution for σ^2 is the average variance associated with the discarded dimensions and the ML solution for μ is the sample mean, see also Appendix A for analysis of the resulting latent points

[3] Because of its similarity with a sample covariance matrix we also use the symbol $\mathbf{S}$ for this matrix, but note that it is technically no sample covariance matrix, because we do not subtract the mean over data points.

the derivative of the log-likelihood L to 0 and obtain

$$\mathbf{S}_N[\mathbf{Z}\mathbf{Z}^T + \sigma^2\mathbf{I}]^{-1}\mathbf{Z} = \mathbf{Z}.$$

After replacing $\mathbf{Z}$ with its singular value decomposition $\mathbf{Z} = \mathbf{U}\mathbf{L}\mathbf{V}^T$ and right-multiplying by $\mathbf{V}$ we see that, again, we arrive at an eigenvalue problem of the inner product matrix similar to classical MDS before

$$\mathbf{S}_N\mathbf{U} = \mathbf{U}(\mathbf{L}^2 + \sigma^2\mathbf{I}). \tag{3.9}$$

The solution for $\mathbf{Z}$ therefore is

$$\mathbf{Z} = \hat{\mathbf{U}}(\hat{\Lambda}_N - \sigma^2\mathbf{I})^{\frac{1}{2}} \tag{3.10}$$

where $\hat{\mathbf{U}}$ are M eigenvectors of $\mathbf{S}_N$ and $\hat{\Lambda}_N$ their corresponding eigenvalues. Lawrence (2005) further shows that L is maximised when the M eigenvectors with the largest eigenvalues are chosen. For small variance σ^2 this solution is almost equal to classical MDS eq. (3.2). The difference is that here the eigenvectors Λ_N are additionally scaled by $1/D$, removing the dependence of the scale of $\mathbf{Z}$ on the dimensionality of the observed space D.

With the given formulation of dual, probabilistic PCA the GPLVM is a straightforward nonlinear extension of it (Lawrence, 2005). We only have to note that the defined likelihood eq. (3.7) is equivalent to the likelihood defined by D independent draws from a common Gaussian Process with inputs $\mathbf{Z}$, outputs $\mathbf{Y}$, mean function $m(\mathbf{z}) = 0$ and linear covariance function $k(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i^T\mathbf{z}_j + \sigma^2\delta_{ij}$ (see e.g. Rasmussen and Williams, 2006). We can then allow nonlinear relationships between $\mathbf{Z}$ and $\mathbf{Y}$ by replacing the linear covariance function with a nonlinear covariance function, for example, with the squared exponential (SE)

$$k(\mathbf{z}_i, \mathbf{z}_j) = \exp(-\frac{1}{2\ell^2}|\mathbf{z}_i - \mathbf{z}_j|^2). \tag{3.11}$$

The solution for $\mathbf{Z}$ eq. (3.10), that was derived under the linear model, obviously does not apply to the GPLVM anymore. Instead, we have to maximise the log-likelihood eq. (3.8) using an iterative optimisation procedure such as scaled conjugate gradients. This optimisation has proven to be difficult in experiments and tends to run into local optima. The remaining part of this chapter explores an approximation to this optimisation based on metric MDS, but before we go on we reconsider the corresponding generative model.

3.1.2.1 Observations with an offset

In eq. 3.6 we assumed that our observations do not have an offset. A linear, generative model with corresponding offset is

$$\mathbf{y}_{:,d} = \mathbf{Z}\mathbf{w}_d + \mu_d \mathbf{1} + \varepsilon_d$$

where $\mathbf{1} \in \mathbb{R}^N$ is a vector of ones. The model leads to the likelihood

$$\begin{aligned} p(\mathbf{Y}|\mathbf{Z},\mu,\sigma^2) &= \prod_{d=1}^{D} \mathcal{N}(\mathbf{y}_{:,d}|\mu_d\mathbf{1}, \mathbf{Z}\mathbf{Z}^T + \sigma^2\mathbf{I}) \\ &= \prod_{d=1}^{D} \mathcal{N}(\mathbf{y}_{:,d}|\mu_d\mathbf{1}, \mathbf{K}). \end{aligned}$$

By setting the derivative of the log likelihood with respect to μ_d to 0 and solving for μ_d we get

$$\mu_d = \frac{\mathbf{1}^T\mathbf{K}^{-1}\mathbf{y}_{:,d}}{\mathbf{1}^T\mathbf{K}^{-1}\mathbf{1}}. \tag{3.12}$$

Therefore the maximum likelihood solution for μ is only equal to the sample mean, if $\mathbf{K} = \mathbf{I}$, i.e. all data points are independent. This somewhat surprising result can be understood when we remember that, in contrast to the original probabilistic PCA formulation, we do not assume that the observed data points are independent[4]. Because data points can be correlated, their correlation needs to be taken into account when computing the offset. In particular, eq. (3.12) weights the contributions of single data points by how much they covary with other data points. For the remainder of our discussion we will continue to assume that there is no measurement offset, but note that the standard way of centring observations with respect to the sample mean does not strictly apply to this model unless covariances between all data points are small. Notice also that for highly correlated data points, for example, for data points which lie close together relative to the lengthscale of the SE covariance function, it is not possible to differentiate between the sampled function and a potential offset in the measurements.

But notice that the offset μ_d may also be integrated out (see Section 4.3 for an analogous example in which we use an offset to allow translations of a template sequence). Then the covariance function contains an additional constant term which corresponds to the variance of the prior over μ_d (assuming that this is a Gaussian prior with zero mean).

[4]This is equivalent to the problem of Generalised Least Squares, see e.g. Kariya and Kurata (2004).

3.1.3 Relating the GPLVM and metric MDS

Above we have seen that for linear $\mathbf{K}$ the GPLVM (dual pPCA) is up to a constant scaling equivalent to classical MDS. For general, isotropic $\mathbf{K}$ we can relate the GPLVM to the more general metric MDS: The free-form maximisation of the likelihood in eq. (3.8) over $\mathbf{K}$ is obtained by setting $\mathbf{K} = \mathbf{S}_N$. Of course, as $\mathbf{K}$ is parametrised by $\mathbf{Z}$ it will not in general be possible to find locations $\mathbf{Z}$ so as to make this happen. However, it does suggest that we might try setting $k(\mathbf{z}_i, \mathbf{z}_j) \simeq s_{ij}$ for all $i,\ j$. If the kernel function k is isotropic, i.e. it is a function of $d_{ij}^2 = |\mathbf{z}_i - \mathbf{z}_j|^2$ so that $k(\mathbf{z}_i, \mathbf{z}_j) = f(d_{ij}^2)$, then we have

$$d_{ij}^2 \simeq f^{-1}(s_{ij}) \qquad \forall\ i,\ j.$$

For example, the SE covariance function sets $k_{ij} = \exp(-d_{ij}^2/2\ell^2)$, so that

$$d_{ij}^2 \simeq -2\ell^2 \log(s_{ij}). \tag{3.13}$$

Given an $N \times N$ matrix of distances with entries d_{ij}^2 it is straightforward to solve for the best M-dimensional Euclidean configuration using classical scaling as described above. The attraction of this method is that it provides a *direct* algorithm for estimating $\mathbf{Z}$, rather than the usual iterative solutions. Alternatively it can be used to initialise $\mathbf{Z}$ before iterative optimisation of L, instead of (say) PCA initialisation of $\mathbf{Z}$ based on an eigendecomposition of $\mathbf{S}_N$.

Scaling $\mathbf{S}_N$: It is sensible to impose the constraint that the diagonal entries in $\mathbf{S}_N$ are such that $d_{ii}^2 = 0$ for all $i = 1, \ldots, N$. Assuming that $f^{-1}(1) = 0$ (which holds e.g. for the SE covariance function), then this can be achieved by replacing $\mathbf{S}_N$ by its rescaled version $\mathbf{R}$, where

$$r_{ij} = \frac{s_{ij}}{\sqrt{s_{ii} s_{jj}}}, \tag{3.14}$$

so that $r_{ii} = 1$ for $i = 1, \ldots, N$. (This is similar to the construction of the correlation matrix from a covariance matrix, except that here the notions of sample and variable are interchanged so that $\mathbf{S}$ and $\mathbf{R}$ are $N \times N$, not $D \times D$.) We assume below that $\mathbf{R}$ is used in place of $\mathbf{S}_N$.

We can now relate this construction to metric MDS, a generalisation of classical scaling. In metric MDS the $\mathbf{z}$-space interpoint distances d_{ij} are related to dissimilarities δ_{ij} in $\mathbf{y}$-space by the relationship $d_{ij} \simeq g(\delta_{ij})$ for some specified, monotonic function g (Kruskal and Wish, 1978, p. 22) (for $g(x) = x$ we recover classical MDS). As r_{ij} is a measure of similarity, it is natural to set $\delta_{ij}^2 = r_{ii} + r_{jj} - 2r_{ij} = 2(1 - r_{ij})$. Combining this with $f(d_{ij}^2) \simeq r_{ij}$ we see indeed that $d_{ij}^2 \simeq f^{-1}(1 - \delta_{ij}^2/2)$. For the SE covariance

function f^{-1} is the logarithm, a monotonically increasing function. Therefore, with a suitable covariance function, GPLVM-MDS is a special form of metric MDS.

A problem: It may happen that there is no d_{ij} corresponding to values of r_{ij} in a certain range. For example, with the SE kernel we cannot find d_{ij}'s corresponding to $r_{ij} \leq 0$, but such values may well arise in practice. Indeed, due to sampling fluctuations, negative empirical r_{ij}'s could occur even if the "true" value were positive, but they could also arise through model mis-specification. A simple approach in this case is to treat the entries with $r_{ij} \leq 0$ as missing, and apply an MDS algorithm that handles missing data as described below. However, note that small r_{ij} corresponds to large d_{ij} for the SE kernel, so there is an expectation that these missing distances in $\mathbf{z}$-space will be large.

3.2 Metric MDS with Missing Data

When we have missing entries in the matrix of dissimilarities we cannot compute the eigendecomposition of it anymore and have to resort to other techniques. We consider two alternatives: the iterative minimisation of a metric MDS objective function called stress (Buja et al., 2001), and the filling in of missing values using probabilistic matrix factorisation (Salakhutdinov and Mnih, 2008). We show below that the former is quite robust on synthetic data while the latter is more sensitive to missing large distances.

3.2.1 Iterative Minimisation of Stress

Several algorithms for metric MDS have been proposed. One of them is the iterative minimisation of the Stress criterion (e.g. Buja et al., 2001)

$$\text{Stress}(\mathbf{Z}) = \left(\frac{\sum_{i,j} w_{ij} (d_{ij} - |\mathbf{z}_i - \mathbf{z}_j|)^2}{\sum_{i,j} w_{ij} d_{ij}^2} \right)^{\frac{1}{2}} \tag{3.15}$$

which minimises the normalised squared error between the given dissimilarities and the distances between the estimated latent points. The weights w_{ij} can then be used to accommodate missing values. In particular we set

$$w_{ij} = \begin{cases} 0 & \text{if } r_{ij} < 0 \\ 1 & \text{otherwise} \end{cases}$$

where r_{ij} are the correlations computed in eq. (3.14). Stress can then be minimised with gradient descent. We employ the routine provided in the Matlab statistics toolbox

(mdscale). We initialise $\mathbf{Z}$ randomly, repeat the optimisation with new initialisation R times and use the Stress value to select the best of the resulting solutions.

The experiments by Graef and Spence (1979) suggest that MDS is particularly sensitive to missing large distances which correspond to small r_{ij}. We investigate this problem for our setup in Section 3.2.3.

3.2.2 Filling in Missing Values with Probabilistic Matrix Factorisation

Recently probabilistic matrix factorisation (PMF) has been proposed (Salakhutdinov and Mnih, 2008) as a method for filling in missing values in a matrix which, we know, has a low-rank factorisation. Based on its construction we can argue that the assumptions of PMF also apply to the squared distance matrix $\mathbf{D}$ with elements $[\mathbf{D}]_{ij} = d_{ij}^2$.

Proposition 1 *If the latent points* $\mathbf{Z}$ *lie in a* M *dimensional Euclidean space, then the matrix of squared distances* $\mathbf{D}$ *has rank* $\leq M+2$.

Proof: Let $\mathbf{A} = -\frac{1}{2}\mathbf{D}$, and $\mathbf{B} = \mathbf{HAH}$, where $\mathbf{H}$ is the centring matrix $\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T$ as in eq. (3.1). Then $\mathbf{B}$ has rank $\leq M$ (and equal to M if there are $N \geq M+1$ points lying in "general position"). Now $d_{ij}^2 = b_{ii} - 2b_{ij} + b_{jj}$, or in matrix notation $\mathbf{D} = \mathbf{b}\mathbf{1}^T + \mathbf{1}\mathbf{b}^T - 2\mathbf{B}$, where $\mathbf{b} = \mathrm{diag}(\mathbf{B})$. As the first two terms are of rank 1, overall $\mathbf{D}$ has rank $\leq M+2$.

As a consequence there must be a factorisation of $\mathbf{D}$ with rank at most $M+2$. PMF defines a probabilistic model for this factorisation and uses maximum likelihood optimisation to find the corresponding factors. The resulting optimisation problem is equivalent to a regularised least-squares problem with objective function

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} I_{ij}\left(d_{ij}^2 - (\mathbf{U}^T\mathbf{V})_{ij}\right)^2 + \frac{\lambda_u}{2}\|\mathbf{U}\|_F^2 + \frac{\lambda_v}{2}\|\mathbf{V}\|_F^2 \tag{3.16}$$

where $\mathbf{U} \in \mathbb{R}^{(M+2)\times N}$ and $\mathbf{V} \in \mathbb{R}^{(M+2)\times N}$ are the factor matrices to be found, $\|\mathbf{M}\|_F$ is the Frobenius norm of matrix $\mathbf{M}$, I_{ij} is 0 if d_{ij}^2 is missing and otherwise 1 and λ_u, λ_v are regularisation parameters related to the amount of noise in the probabilistic model. We initialise $\mathbf{U}$ and $\mathbf{V}$ at random and then find a local minimum of E with gradient descent. In our experiments we set $\lambda_u = \lambda_v = 1$ and restart optimisation with different initialisation 5 times although the optimisation converges to very similar values of E in all of them.

Remark: D is a (squared) distance matrix. The PMF model neither takes into account the symmetry of $\mathbf{D}$, nor its positiveness, nor the zeros in its diagonal. The resulting matrix $\hat{\mathbf{D}} = \mathbf{U}^T\mathbf{V}$ will be close to symmetric and positive, but it may still have deviations from the properties of a distance matrix. Therefore we make $\hat{\mathbf{D}}$ symmetric with $\hat{\mathbf{D}} = \frac{1}{2}(\hat{\mathbf{D}}^T + \hat{\mathbf{D}})$, reverse the sign of all negative entries (usually around 2% of all entries in $\hat{\mathbf{D}}$) and set the diagonal to 0. We have also tried to incorporate the symmetry into the model by directly setting $\hat{\mathbf{D}} = \frac{1}{2}(\mathbf{U}^T\mathbf{V} + \mathbf{V}^T\mathbf{U})$ in eq. (3.16), but this lead to worse reconstruction accuracies of the latent configurations (see next section) except for distance matrices which have only been disturbed with a small amount of noise (up to $\eta = 0.78$ in eq. (3.17)).

Once we have a full distance matrix $\hat{\mathbf{D}}$ we use classical MDS to find latent points $\mathbf{Z}$ without further iterative optimisations.

3.2.3 Experiments for Robustness against Missing Data and Noise

In this section we investigate the robustness of the two presented methods against missing large distances, as expected for GPLVM-MDS, and against noise on the covariance matrix. The experiments indicate that Stress-MDS is to be preferred over PMF.

First, we focus on large, missing distances only. We generated 100 random, $\mathcal{N}(0, \mathbf{I})$, data points in 2 dimensions and computed their squared distances. We then removed all distances above a certain threshold (4.09, 3.88, 3.65, 3.42, 3.16, 2.88, 2.57, 2.22, 1.79) and applied Stress-MDS ($R = 200$) and PMF-MDS to it. The resulting latent points were subjected to a Procrustes analysis to account for the invariances inherent in MDS. We used the Procrustes analysis of the Matlab statistics toolbox (function procrustes) and allowed translation, rotation, reflection and scaling of the latent points. We compare the residual Procrustes errors, which correspond to the normalised sum of squared errors (nSSE) between the original and reconstructed latent points, between Stress-MDS and PMF-MDS for the different distance thresholds. A nSSE of 0.1 means that the error is roughly 1/10 of the spread of the original data points[5]. All experiments were repeated 12 times with different, initial latent points. Fig. 3.1 shows the mean error over the 12 repetitions together with its two standard deviation error bars. As more and more distances are removed the error increases as expected. Stress-MDS stays surprisingly robust while PMF-MDS has larger error and variance.

It is more interesting to look at the error dependent on the number of missing

[5]Definition: $\text{nSSE}(\mathbf{X}, \hat{\mathbf{X}}) = \sum_{ij}(\hat{x}_{ij} - x_{ij})^2 / \|\mathbf{X}_c\|_F^2$ where $\mathbf{X}_c$ is $\mathbf{X}$ with centred columns

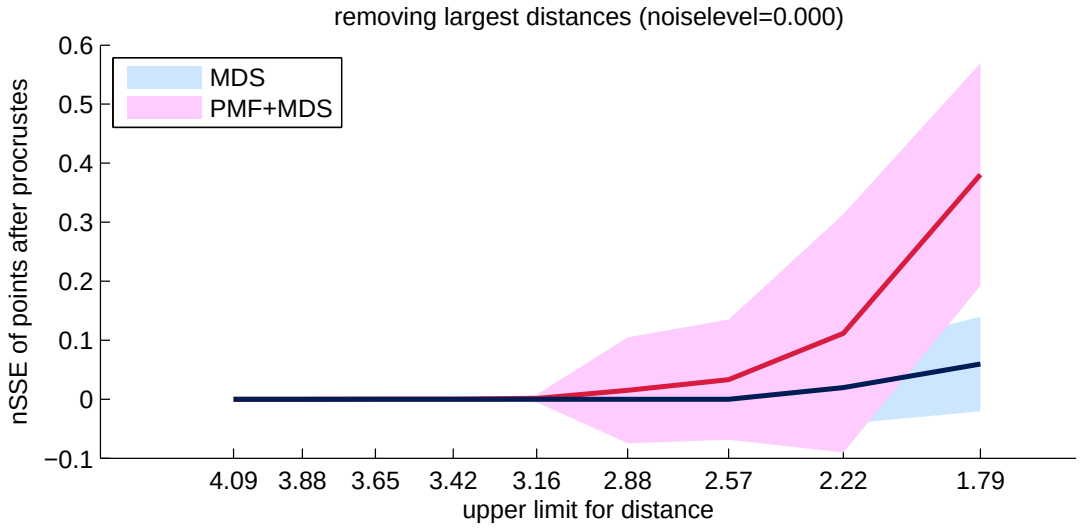


Figure 3.1: Residual Procrustes errors of resulting reconstruction of latent points for decreasing threshold of large distances which are removed before MDS procedure. Shown are mean and twice the standard deviation of 12 repetitions.

distances, because for any given configuration of latent points the number of missing distances might be different for a fixed distance threshold. Fig. 3.2 shows the error as before, but dependent on the fraction of missing entries in the distance matrix as a scatter plot containing all 12 repetitions of the experiment. In all experiments Stress-MDS gives a lower error than PMF-MDS. Up to about 20% missing distances Stress-MDS can perfectly reconstruct the underlying latent points.

In these experiments we only removed distances. The remaining distances were equal to the correct ones and did not contain noise. So we repeated these experiments (generated latent points were the same), but, instead of removing distances dependent on their size, we added an increasing amount of noise to the covariance matrix which in turn also resulted in an increasing fraction of missing distances. To be precise, we computed the covariance matrix from the squared distances using the SE covariance function eq. (3.11) with $\ell = 1$ and added to it a rank one update of noise

$$\tilde{\mathbf{S}}_N = \mathbf{K} + \varepsilon\varepsilon^T \qquad \varepsilon_i \sim U[-0.5\eta, 0.5\eta] \quad i = 1,\ldots,N \tag{3.17}$$

where ε_i is uniformly distributed in $[-0.5\eta, 0.5\eta]$. This form of noise maintains the symmetry of the covariance matrix, but introduces negative entries depending on the size of η. In our experiments we used $\eta = \{0,\ 0.15,\ 0.23,\ 0.34,\ 0.52,\ 0.78,\ 1.17,\ 1.76,\ 2.65,\ 4.00\}$. From $\tilde{\mathbf{S}}_N$ we computed the correlations $\mathbf{R}$ and discarded all negative correlations before reconstructing the squared distances $\mathbf{D}$ using the inverse of the SE

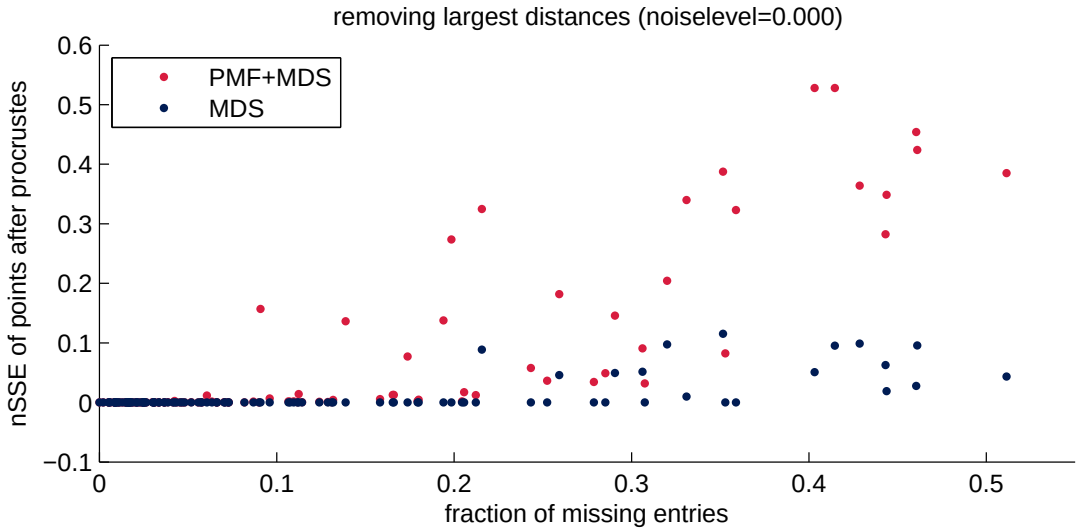


Figure 3.2: Residual Procrustes errors of resulting reconstruction of latent points for increasing number of missing entries in distance matrix (shown as fraction of total number of entries). Points always occurr as pairs on one particular fraction of missing entries.

covariance function and applying Stress-MDS or PMF-MDS to find the latent points. The resulting Procrustes errors can be seen in Fig. 3.3. Stress-MDS also gives slightly more accurate results than PMF-MDS in the presence of noise, but, most importantly, we have to note that the error increases very quickly with the amount of noise in the data. We can compare this to the situation without noise by plotting the error again dependent on the number of missing distances, as done in Fig. 3.4. We notice that Stress-MDS reaches an error level of 0.1 at about 10-15% missing distances while without noise it barely goes above an error of 0.1 over the whole range of tested fractions up to 50%. We mention particularly the error of 0.1, because this is the level from which it becomes subjectively difficult to see the relationship between original and reconstructed configurations in 2D due to the error in the reconstructions.

In conclusion, these experiments show that for small amounts of missing distances or noise both Stress-MDS and PMF-MDS are robust and able to reconstruct the latent points. For increasing distortion of the distances, however, Stress-MDS exhibits an advantage over PMF-MDS and, therefore, is our preferred choice below. The noise in our experiments had a strong negative effect on the accuracy of the point reconstructions, but its form was rather simplistic. In the next section we investigate the sources of noise in the GPLVM directly and quantify their effect on the accuracy of resulting latent spaces.

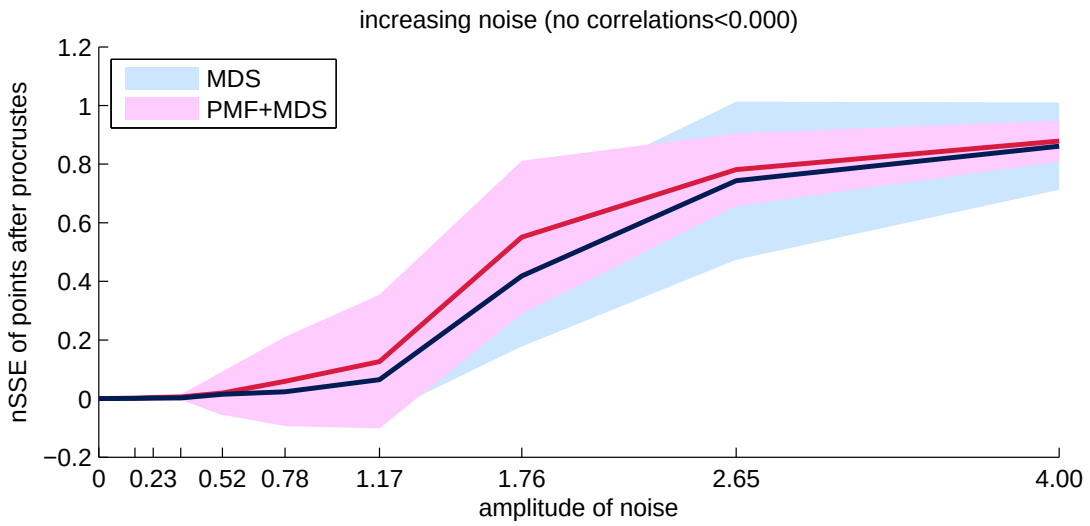


Figure 3.3: Residual Procrustes errors of resulting reconstruction of latent points for increasing amount of noise. Shown are mean and twice the standard deviation of 12 repetitions.

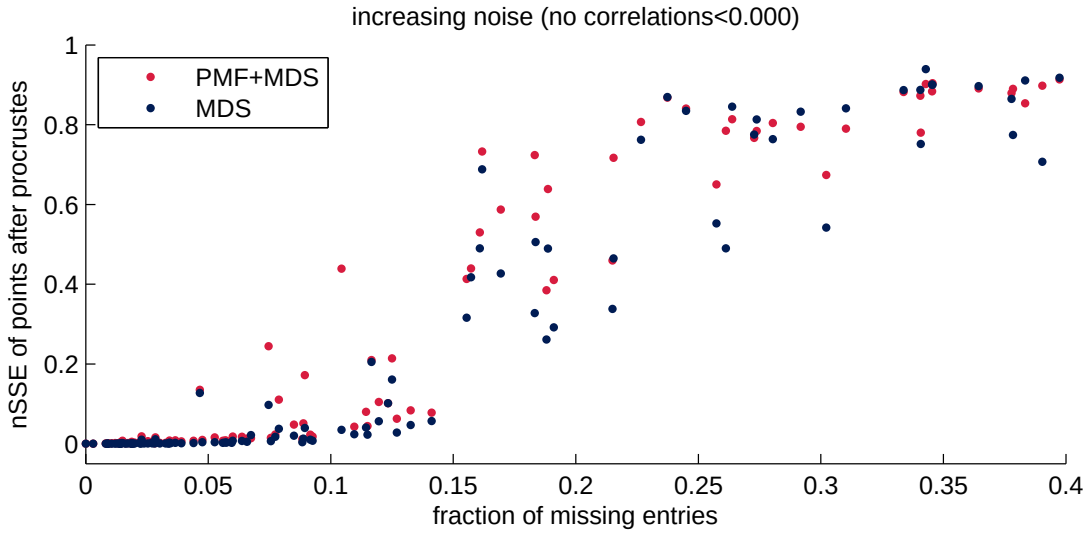


Figure 3.4: Residual Procrustes errors of resulting reconstruction of latent points for increasing number of missing entries in distance matrix (shown as fraction of total number of entries) under increasing amount of noise.

3.3 Variability of Covariance Estimates

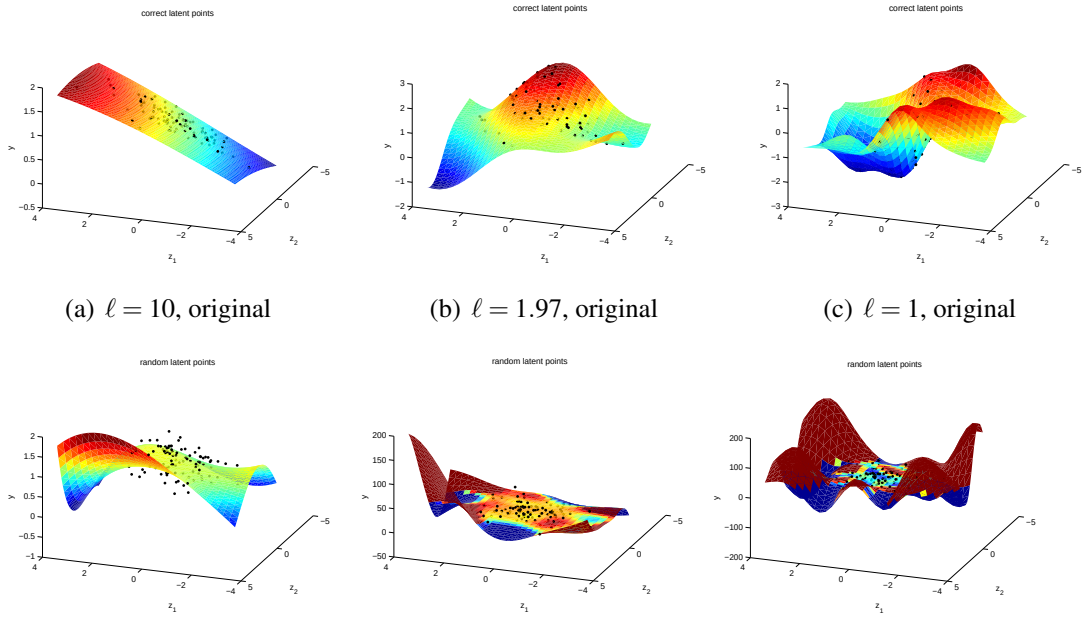
Even if we specify the correct model, sampling fluctuations can lead to missing entries in the squared distance matrix. In this section we investigate the variance of the normalised inner product matrix $\mathbf{S}_N$. Both theoretically and on synthetic data, we show that this variance scales with the inverse of the dimensionality of the observed data D. Furthermore, we demonstrate on synthetic data that, in contrast to PCA, GPLVM-MDS can reconstruct the true underlying latent points even when the generative process is highly nonlinear as long as D is large, i.e., the covariance matrix is well approximated.

3.3.1 Generating Synthetic Data from the GPLVM

The experiments following in this section are based on synthetic data which we generated directly from the model. Consequently, the conclusions that we draw apply to the model itself. Any deviation of actual data from the model assumptions introduce additional errors.

The first step of the generation of data is to draw $N = 100$ data points $\mathbf{Z} \in \mathbb{R}^{100\times 2}$ from a 2D Gaussian $\mathcal{N}(0,\mathbf{I})$ distribution as before. From these data points we computed the covariance matrices $\mathbf{K}$ using the SE covariance function with different lengthscales $\ell \in \{10.00, 1.97, 1.41, 1.15, 1.00\}$. According to the GPLVM the data points in each of D output dimensions then must be drawn from a Gaussian distribution with covariance $\mathbf{K}$: $\mathbf{y}_{:,d} \sim \mathcal{N}(0,\mathbf{K})$. To draw from a multivariate Gaussian distribution with dependent variables we used the method described in Rasmussen and Williams (2006, appendix A.2) which means sampling from a standard, normal distribution and then correlating the samples by multiplying them with the Cholesky decomposition of $\mathbf{K}$. We repeated this procedure for all of the D output dimensions to obtain the full data sets $\mathbf{Y} \in \mathbb{R}^{100\times D}$ where we also varied $D \in \{3,5,8,14,23,39,65,108,180,300\}$. All experiments below were repeated 12 times for each setting of D and ℓ by sampling new $\mathbf{Y}$ from a given $\mathbf{Z}$ (resulting in $12 \cdot 10 \cdot 5 = 600$ different data sets based on one draw of $\mathbf{Z}$).

Fig. 3.5 (a-c) shows examples of the generated data for 3 different lengthscales. For $\ell = 10$ all points covary strongly and the data is almost linear. For decreasing lengthscales the data becomes more nonlinear. These plots use the real locations of the latent points which are unknown to us in the GPLVM setting. Fig. 3.5 (d-f) exemplifies how dramatically the functions implemented by the GPLVM change when the latent points associated with the data are changed. In these plots the y-values corresponding

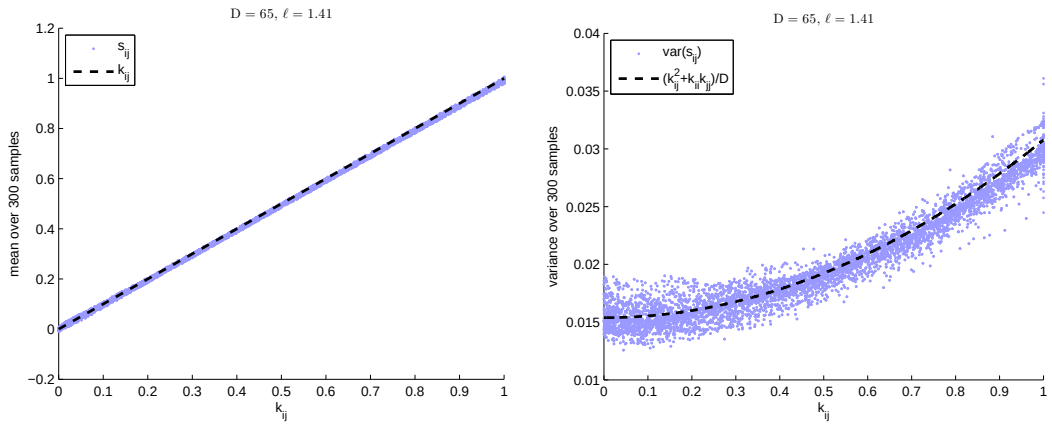


(a) $\ell = 10$, original (b) $\ell = 1.97$, original (c) $\ell = 1$, original

(d) $\ell = 10$, random (e) $\ell = 1.97$, random (f) $\ell = 1$, random

Figure 3.5: Example draws of synthetic data for one output dimension and different lengthscales. (a-c) x-y-axis corresponds to latent positions used to draw the data (z_1, z_2), z-axis shows a single output (y), dots are the data set $\{\mathbf{Z}, \mathbf{Y}\}$, surface is output of the corresponding GP. (d-f) x-y-axis corresponds to new random draw of latent positions $(\hat{z}_1, \hat{z}_2)$, z-axis as before, dots are the data set $\{\hat{\mathbf{Z}}, \mathbf{Y}\}$, surface is output of the corresponding GP. Note that the colour coding of height is equal in the two rows, but that the scale of y is much larger in (e,f).

to the output of the GPs are the same as in (a-c), but the z_1 and z_2 coordinates have been randomised. We observe that the functions implemented by the mean prediction of the new GPs (surfaces) completely changed and that the data points (dots) do not lie on the surfaces anymore which results in very low likelihoods. Consequently, maximum likelihood optimisation of the GPLVM aims to locate latent points $\mathbf{Z}$ such that $\{\mathbf{Z}, \mathbf{Y}\}$ can be well approximated with the GPs in the GPLVM. The situation is complicated by the simultaneous optimisation of the GP hyperparameters, such as ℓ, but before we return to that point in Section 3.4 we investigate the type of noise that we have to expect in the data.



(a) mean (b) variance

Figure 3.6: Mean and variance of entries in the sample covariance matrix. x-axis: value in the true covariance matrix (k_{ij}), y-axis: mean (a) or variance (b) of the corresponding entry in sample covariance matrix (s_{ij}). Dashed line: theoretical result, dots: empirical mean and variance over 300 data sets. Data sets had lengthscale of $\ell = 1.41$ and number of output dimensions $D = 65$.

3.3.2 Distribution of the Sample Covariance Matrix

The GPLVM model in eq. (3.7) states that the columns of the matrix of observations $\mathbf{Y}$ are independent samples from a multivariate Gaussian distribution with mean $\mathbf{0}$ and covariance $\mathbf{K}$. Thus, the matrix $\mathbf{Y}\mathbf{Y}^T$ is Wishart distributed (Wishart, 1928) with parameter $\mathbf{K}$ and D degrees of freedom: $\mathbf{Y}\mathbf{Y}^T \sim W_N(\mathbf{K}, D)$. The Wishart distribution has mean $E(\mathbf{Y}\mathbf{Y}^T) = D\mathbf{K}$ and the elements of $\mathbf{Y}\mathbf{Y}^T$ have variance $V(\mathbf{y}_i^T \mathbf{y}_j) = D(k_{ij}^2 + k_{ii}k_{jj})$ (see e.g. Muirhead (2005, ch. 3.2)). Consequently, the mean and variance for the sample covariance are

$$E(\mathbf{S}_N) = \frac{1}{D}E(\mathbf{Y}\mathbf{Y}^T) = \mathbf{K} \qquad V(s_{ij}) = \frac{1}{D^2}V(\mathbf{y}_i^T \mathbf{y}_j) = \frac{1}{D}(k_{ij}^2 + k_{ii}k_{jj}) \tag{3.18}$$

Therefore, $\mathbf{S}_N$ is an unbiased estimator of $\mathbf{K}$ and estimates improve with increasing dimensionality of the observation space.

We confirmed this finding experimentally by repeatedly sampling data points $\mathbf{Y}$ from a given covariance matrix $\mathbf{K}$ and computing the sample covariance matrix from each of the resulting data sets. In Fig. 3.6 we plot the mean and variance of the entries in the sample covariance matrices over 300 such draws together with the theoretical result. Theory and experiment correspond well to each other.

We obtained this result by averaging over many data sets. For every single data set

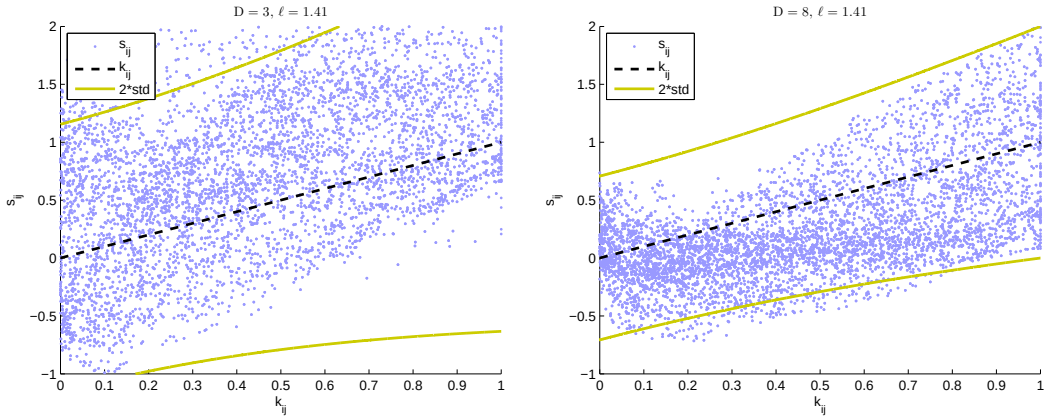
the variability of the entries in the sample covariance matrix may deviate from the predicted values, but we still expect that the overall variability decreases with the number of output dimensions, because more information about the underlying covariances is available. We again confirmed this empirically and plot results in Fig. 3.7. For this figure we only drew single data sets and plot the true covariances against those computed in the sample covariance matrix. Eventually we are not interested in the sample covariance matrix directly, but in the correlation matrix $\mathbf{R}$ as computed in eq. (3.14). Fig. 3.8 demonstrates that it is indeed beneficial to normalise the sample covariance matrix in this way, because this procedure makes the estimates of large covariances very accurate while the variance of small covariances stays similar to before.

The theoretical result and the experiments clearly state that the estimates of the covariances between data points improve for increasing dimensionality of the data. Consequently, with less noise on the distances computed from $\mathbf{R}$ the reconstructed configurations of latent points should also become more accurate. This is the topic of the next section.

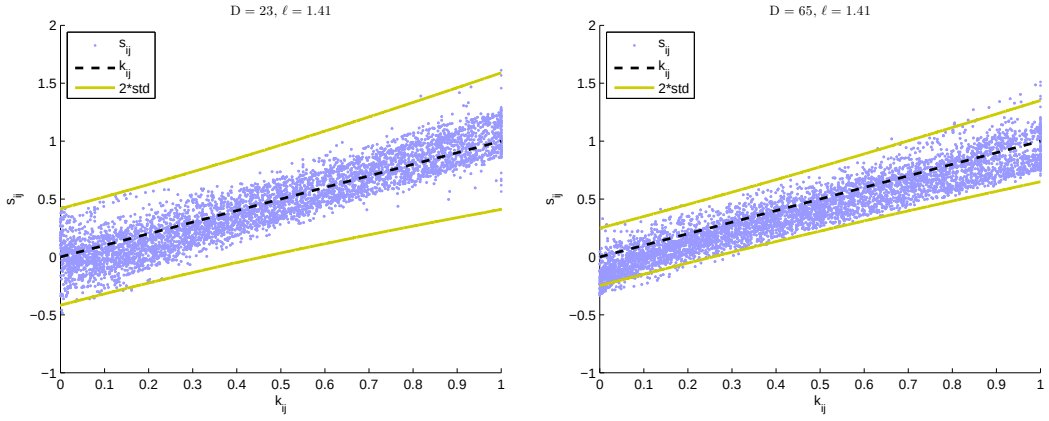
3.3.3 Effects of Variability of Sample Covariance on Reconstruction Quality

In Section 3.2.3 we have seen that noise can heavily disturb the reconstruction of the latent points from a covariance matrix and we now know that we have to expect a large amount of noise on sample covariance matrices which have been computed from a data set with only a few dimensions. How many independent dimensions does a data set need before we can reconstruct the underlying latent points with high accuracy?

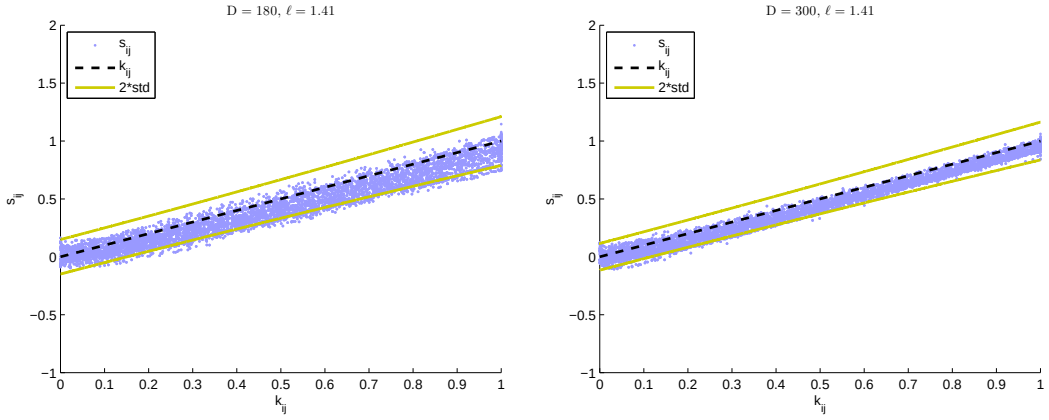
We used GPLVM-MDS to reconstruct latent points for the various data sets that we generated with different lengthscales and different number of output dimensions. Again, we performed Procrustes analysis of the results with respect to the true underlying latent points to account for the invariances of MDS and report the resulting nSSE. The mean nSSE over the 12 repetitions of the experiments is plotted in Fig. 3.9. We draw two conclusions from these results: 1) as data points become less correlated with decreasing lengthscale (and the underlying function becomes more variable) the accuracy of reconstruction reduces. 2) For increasing number of output dimensions the accuracy of reconstruction increases as expected. The values in table 3.1 indicate that we can expect to achieve an error below 0.1 for lengthscales down to $\ell = 1.41$ already from 65 independent output dimensions.



(a) D=3 (b) D=8

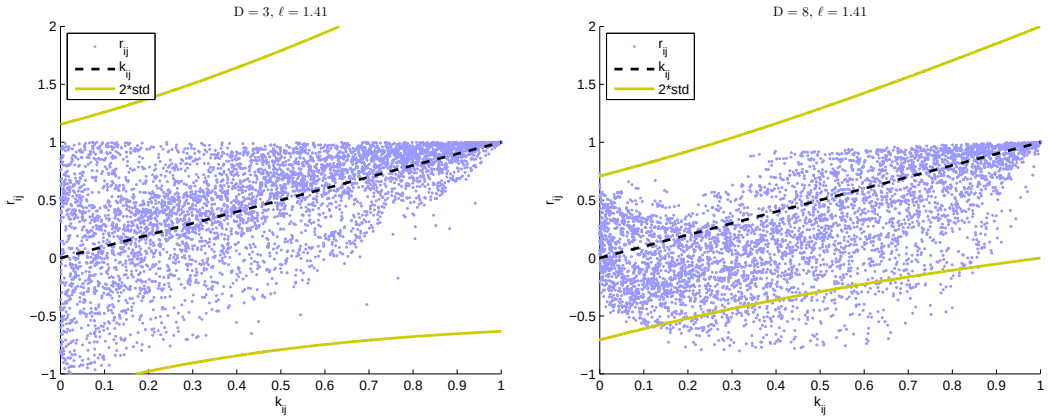


(c) D=23 (d) D=65



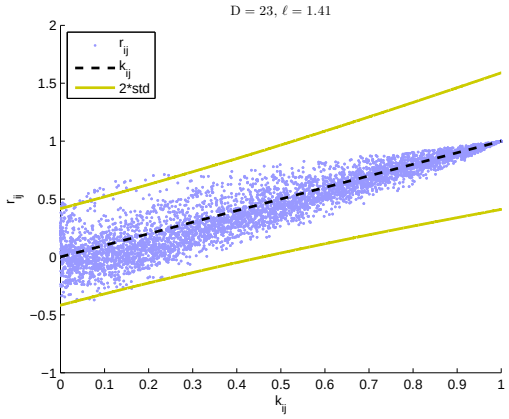
(e) D=180 (f) D=300

Figure 3.7: Correspondence between true covariances and sample covariance of a single data set for increasing number of output dimensions. x-axis: value in the true covariance matrix (k_{ij}), y-axis: value of the corresponding entry in sample covariance matrix (s_{ij}). Yellow lines: twice the standard deviation according to Wishart distribution of covariance matrix. Data sets had lengthscale of $\ell = 1.41$.

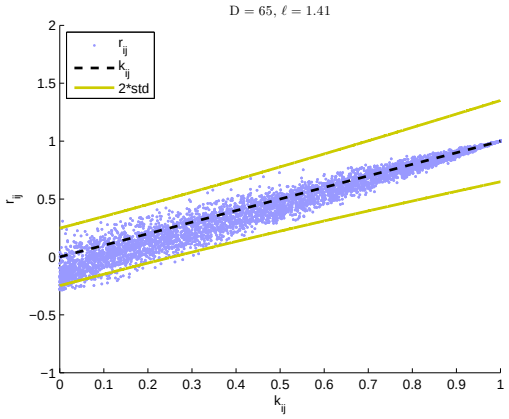


(a) D=3

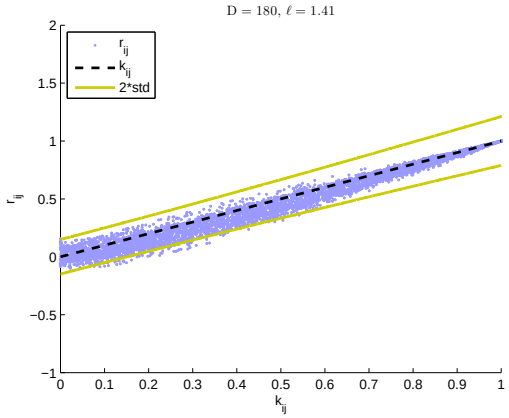
(b) D=8



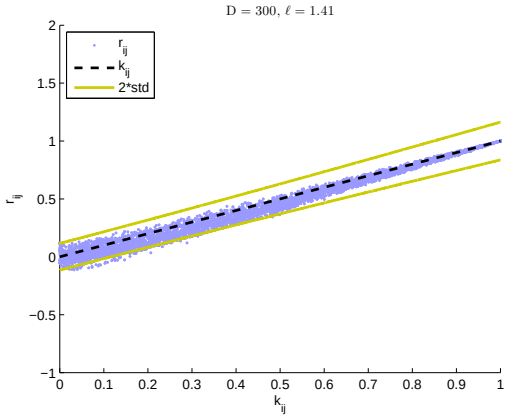
(c) D=23



(d) D=65



(e) D=180



(f) D=300

Figure 3.8: Correspondence between true covariances, k_ij and sample correlations, r_ij, of a single data set for increasing number of output dimensions. x-axis: value in the true covariance matrix (k_{ij}), y-axis: value of the corresponding entry in sample correlation matrix (r_{ij}). Yellow lines are repeated from Fig. 3.7 to ease visual comparison. Data sets had lengthscale of $\ell = 1.41$.

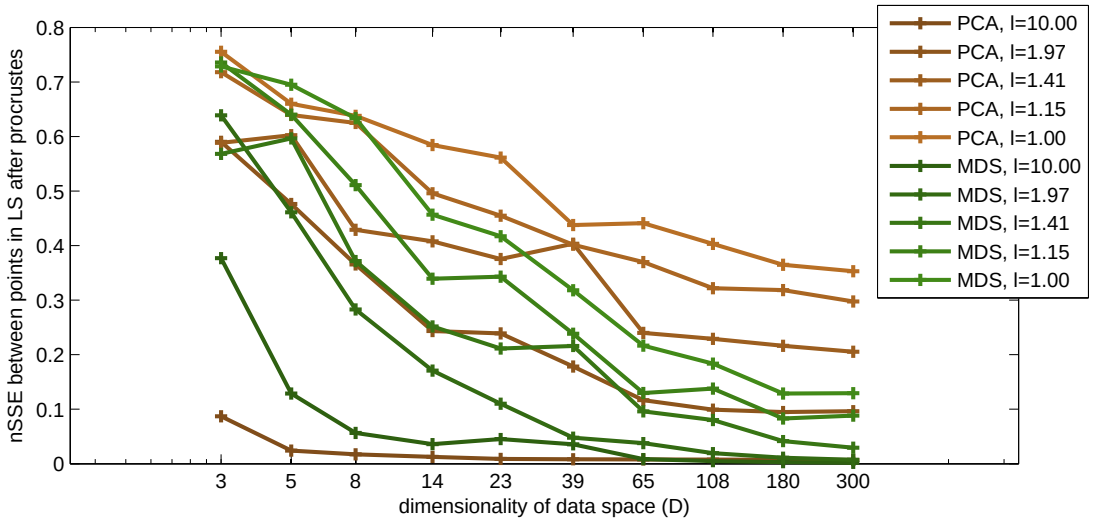


Figure 3.9: Mean error between original and reconstructed latent points after Procrustes analysis. Green: GPLVM-MDS, brown: pPCA. Colour darkness codes for lengthscale (light: $\ell = 1$, dark: $\ell = 10$). x-axis: number of output dimensions.

ℓ	3	5	8	14	23	39	65	108	180	300
10.00	0.377	0.129	0.056	0.036	0.045	0.036	0.009	0.005	0.003	0.002
1.97	0.639	0.461	0.283	0.171	0.110	0.048	0.038	0.020	0.011	0.007
1.41	0.568	0.596	0.372	0.252	0.211	0.216	0.096	0.080	0.042	0.030
1.15	0.736	0.640	0.511	0.339	0.343	0.238	0.130	0.138	0.083	0.088
1.00	0.729	0.695	0.634	0.457	0.417	0.318	0.217	0.184	0.129	0.130

Table 3.1: Mean error between original and reconstructed latent points after Procrustes analysis as depicted in Fig. 3.9. Columns correspond to different number of output dimensions (D) as indicated.

In order to get a qualitative impression of these errors we depict example configurations of latent points in Fig. 3.10 together with the true latent points. The 4 plots show solutions for 4 different draws of $\mathbf{Y}$ with varying lengthscale, but equal $\mathbf{Z}$ and output dimensionality ($D = 65$). The achieved nSSEs were 0.004, 0.068, 0.186 and 0.255. The most important observation in these plots is that most of the error is contributed from points lying at the margin of the data set while points in the centre are reconstructed with low error. This matches our finding that large correlations towards $r_{ij} = 1$ are good estimators of the true covariances k_{ij} while small correlations are more noisy (remember that small covariances correspond to large distances). Because for decreasing lengthscales the number of small covariances increases, the reconstruction becomes worse from the outside of the data set towards the centre. This also makes intuitive sense: the method is based on information about the covariances between data points, but for small lengthscales the data points will covary very little and will be almost independent. Then it is not possible to reconstruct the original configuration of points, because there is not sufficient information about the relationship between points in the data.

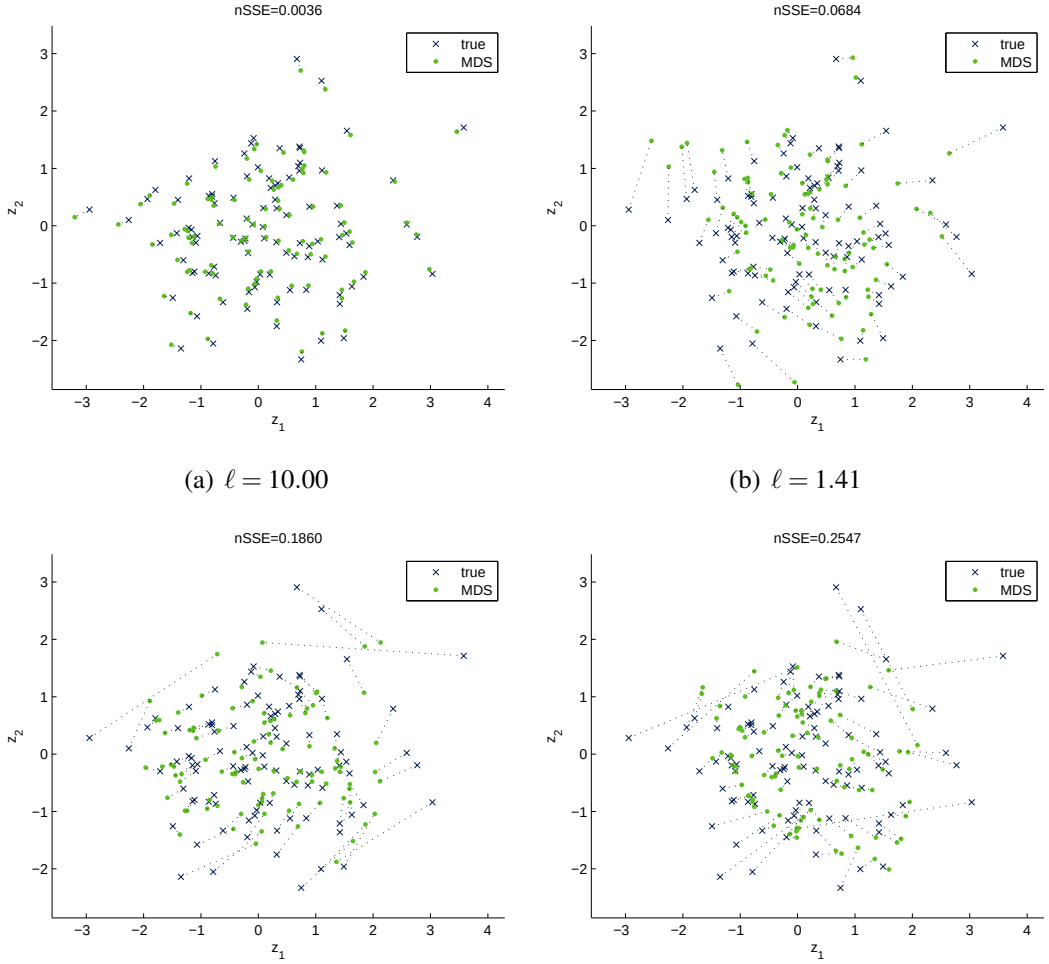
3.4 Solving the GPLVM

GPLVM-MDS can be seen as a stand-alone dimensionality reduction technique in the flavour of Isomap[6] (Tenenbaum et al., 2000) or LLE (Roweis and Saul, 2000), but its real purpose is to shortcut the expensive optimisation of the GPLVM. In the ideal case the GPLVM-MDS solution is the global optimum of the GPLVM likelihood. Because it is an approximation and because of the presence of noise, this may not be the case. Then it should at least lie close to an optimum and be a good initialisation for the GPLVM. In this section we explore how good GPLVM-MDS is as an initialisation for the GPLVM and see that it has clear advantages over PCA on the synthetic data, but loses its advantage on motion capture data.

3.4.1 Comparison to PCA on Synthetic Data

Remark: Comparing to which version of PCA? All versions of PCA are based on the eigenvectors of the sample covariance matrix $\mathbf{S}_D$ as principal components. However, standard PCA, probabilistic PCA based on eq. (3.5) and dual probabilistic PCA

[6] Note the relationship between Isomap and GPLVM-MDS: both use MDS to find a configuration of points from distances, but differ in how the distances are computed.



(a) $\ell = 10.00$ (b) $\ell = 1.41$

(c) $\ell = 1.15$ (d) $\ell = 1.00$

Figure 3.10: Example reconstructions with true latent points. Configuration of true points and number of output dimensions is same in all plots, plots differ in lengthscale used to compute true covariances before generating data $\mathbf{Y}$ on which GPLVM-MDS was applied.

based on eq. (3.6) differ in the scaling of the principal components. While the latent points produced by standard PCA scale with the dimensionality of the data, in dual probabilistic PCA this dependency is removed (see Section 3.1.2). In probabilistic PCA the latent variables are scaled by the square root of the inverse of the corresponding eigenvalues of $\mathbf{S}_D$, so the standard deviation of the latent points becomes 1 in all dimensions as defined in their prior (see appendix A).

The GPLVM is derived from dual probabilistic PCA. Hence, comparing to dual probabilistic PCA makes sense, because it is the linear form of our model. However, there are two reasons for comparing against probabilistic PCA instead. 1) Probabilistic PCA is the standard choice of initialisation for the GPLVM. This may also be motivated by the additional standard normal prior on the latent variables in Neil Lawrence's implementation of the GPLVM, the use of which is also motivated below. 2) The latent variables in our synthetic data are actually standard normal distributed. Therefore the scale of the probabilistic PCA latent points will be approximately correct which we observe in an increased GPLVM likelihood. However, the difference in GPLVM likelihood between dual and original probabilistic PCA did not have an effect on conclusions drawn between PCA and GPLVM-MDS. Below we report only results for probabilistic PCA.

For our experiments on synthetic data Fig. 3.9 shows that latent points reconstructed with GPLVM-MDS are closer to the true latent points than those reconstructed with PCA (all experiments repeated 12 times as before, number of repetitions of Stress-MDS $R = 200$). Fig. 3.11 repeats the data in Fig. 3.9 but as the difference between GPLVM-MDS and PCA where this result becomes even clearer. It is then also apparent that the advantage of GPLVM-MDS is most pronounced for output spaces with many independent dimensions, in particular with respect to the variance of this difference across several trials. These findings are also reflected in the corresponding values of the GPLVM log-likelihood. To compute the log-likelihood of the GPLVM for a given reconstruction of latent points we compute the covariance matrix based on it and use eq. (3.8). The results are presented in Fig. 3.12. Note that the GPLVM log-likelihood scales with D. We remove this dependency and divide the log-likelihood by D in these plots to see whether it improves for increasing D. We call the result normalised log-likelihood. From about $D = 10$ we can say with high confidence that GPLVM-MDS is a better initialisation of the GPLVM than PCA, because its reconstruction of latent points leads to higher likelihoods. Also in terms of absolute log-likelihoods GPLVM-

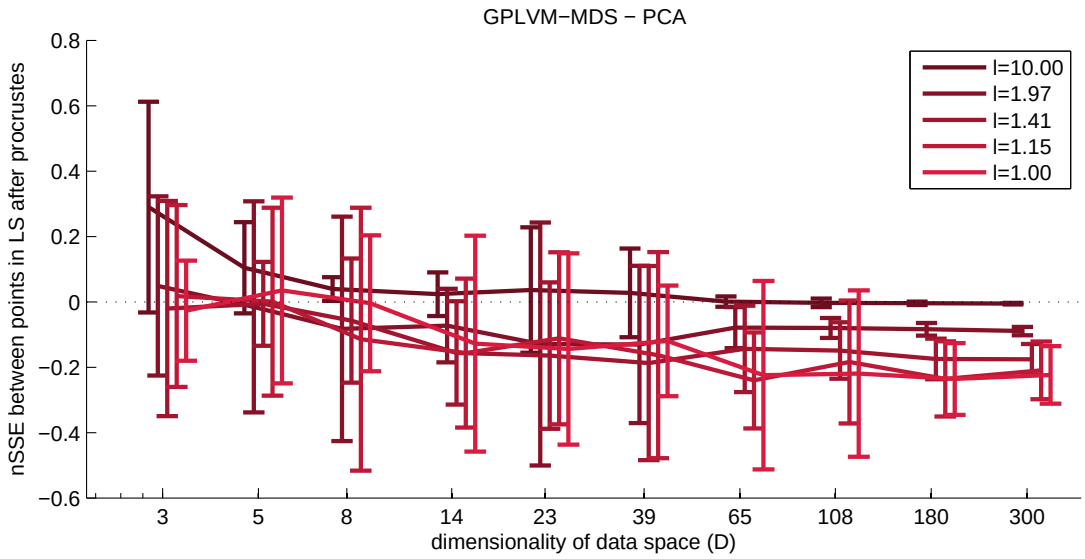


Figure 3.11: Difference between mean errors between original and reconstructed latent points after Procrustes analysis of GPLVM-MDS and PCA (difference below 0 means PCA has larger error). Error bars: twice standard deviation over 12 repetitions. Colour darkness codes for lengthscale (light: $\ell = 1$, dark: $\ell = 10$). x-axis: number of output dimensions. For each D points for different lengthscales have been plotted with an offset for better visibility.

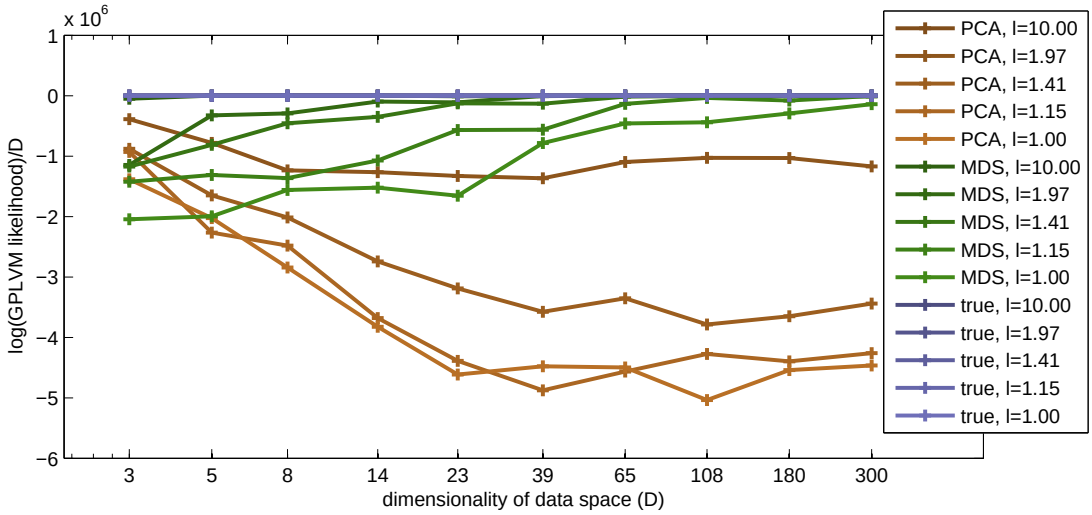
MDS achieves much better than PCA and improves for increasing D even though it remains far from the log-likelihoods of the true configuration of points for lengthscales close to 1. Notice, on the other hand, that both, GPLVM-MDS and PCA, do very well for the large lengthscale $\ell = 10$. So, for almost linear data the linear method PCA is sufficient, otherwise GPLVM-MDS is able to account for the nonlinearities in the data until data points become independent through too small lengthscale.

3.4.2 Results of GPLVM Optimisation

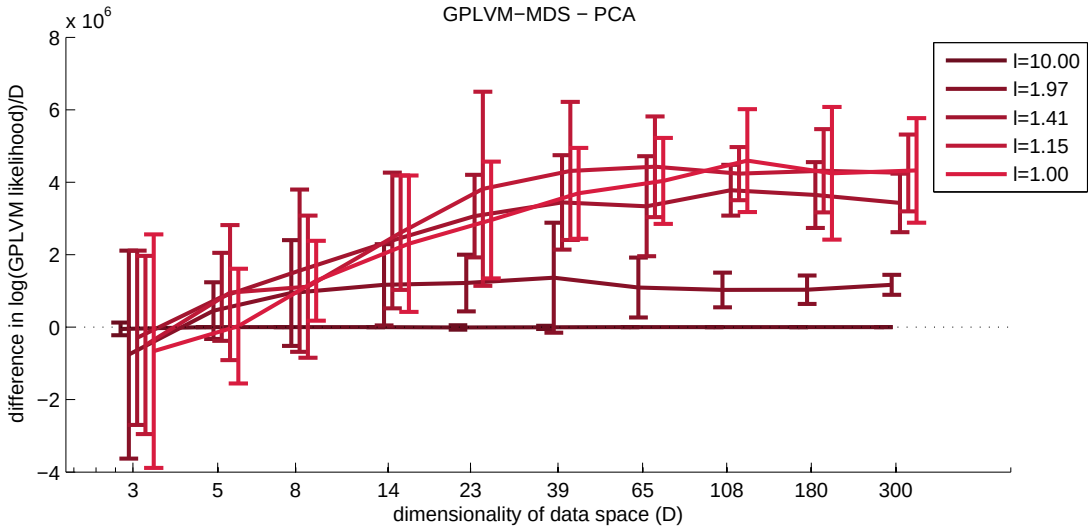
In this section we look at the effects of optimising the GPLVM likelihood on the configurations found by PCA and GPLVM-MDS on the synthetic data. It is well known (but seldom mentioned) that optimising the GPLVM is a nightmare. The likelihood has many local minima and initialisation is key. The reason for this is that the GPLVM has many parameters and they influence each other nonlinearly. Additionally to the latent points (NM parameters) the parameters of the covariance function (e.g. lengthscale and output scale) are optimised simultaneously. First, we focus only on the latent points and keep covariance parameters fixed to the true value which was used to generate the data.

Fig. 3.13 presents the normalised GPLVM log-likelihood after 200 steps of conjugate gradient descent with fixed covariance function parameters. When the GPLVM is initialised with the true latent points, the likelihood still improves marginally, because of sampling errors in the data. However, it improves tremendously (notice the difference in scale compared to Fig. 3.12) for the PCA and GPLVM-MDS latent points which come close to the values for the true latent points. For GPLVM-MDS the log-likelihoods even almost reach those for the true latent points for number of output dimensions greater than 100. This is good news and suggests that the GPLVM optimisation indeed improves on the initialisation and drives latent points towards the true configuration, but looking at the resulting latent points directly gives a different picture.

Fig. 3.14(a) shows the change in nSSE of Procrustes aligned reconstructions of latent points after optimisation for GPLVM-MDS. We observe 2 behaviours: either the errors stay approximately equal, or there is a significant increase. For PCA (not shown) almost all configurations of latent points with $\ell > 10$ were much worse after optimisation. How can likelihoods improve while latent point configurations worsen? The answer lies in the scale of the latent points. Before optimisation the standard



(a) normalised GPLVM log-likelihood



(b) difference in normalised GPLVM log-likelihood: GPLVM-MDS−PCA

Figure 3.12: Comparison between GPLVM-MDS and PCA based on normalised GPLVM log-likelihood. x-axis: number of output dimensions. Colour darkness codes for lengthscale (light: $\ell = 1$, dark: $\ell = 10$). (a) Mean over 12 trials of absolute normalised log-likelihood for GPLVM-MDS, PCA and the true configuration of points. (b) Difference between GPLVM-MDS and PCA of data presented in (a) (difference above 0 means GPLVM-MDS has larger likelihood), error bars: twice standard deviation. In (b) for each D points for different lengthscales have been plotted with an offset for better visibility.

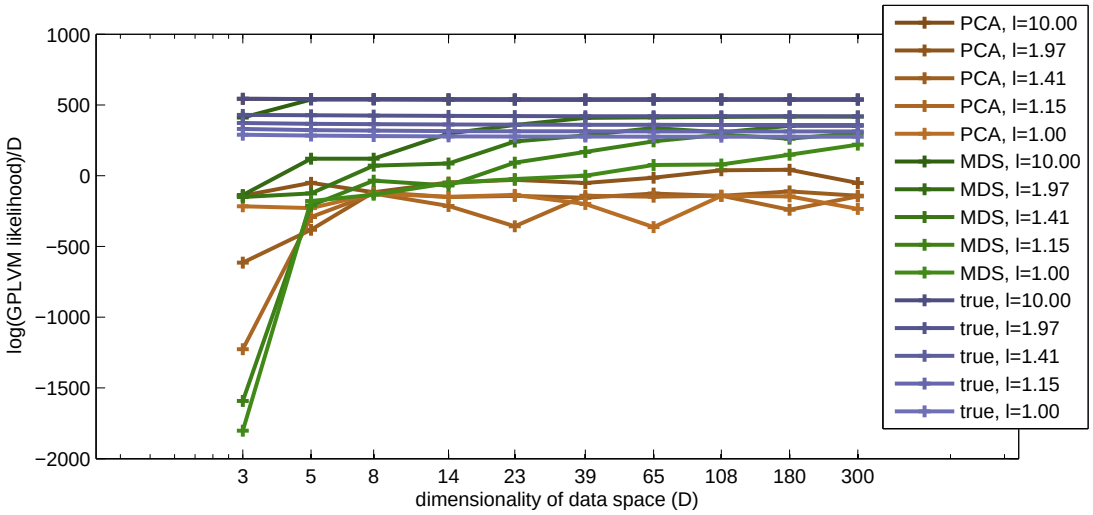
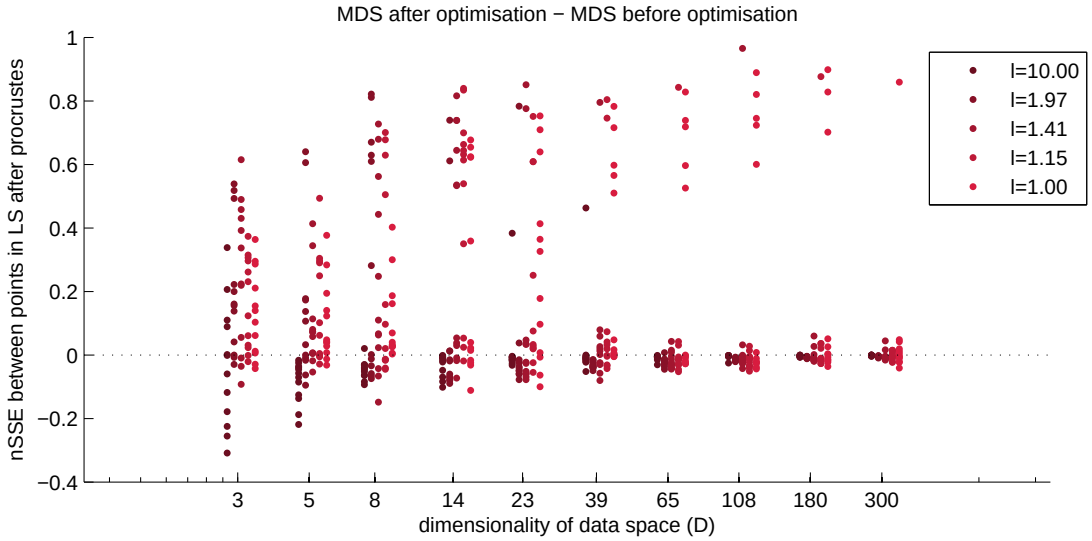


Figure 3.13: Normalised GPLVM log-likelihood after optimisation (mean over 12 repetitions) for GPLVM-MDS, PCA and the true configuration of points. x-axis: number of output dimensions. Colour darkness codes for lengthscale (light: $\ell = 1$, dark: $\ell = 10$).

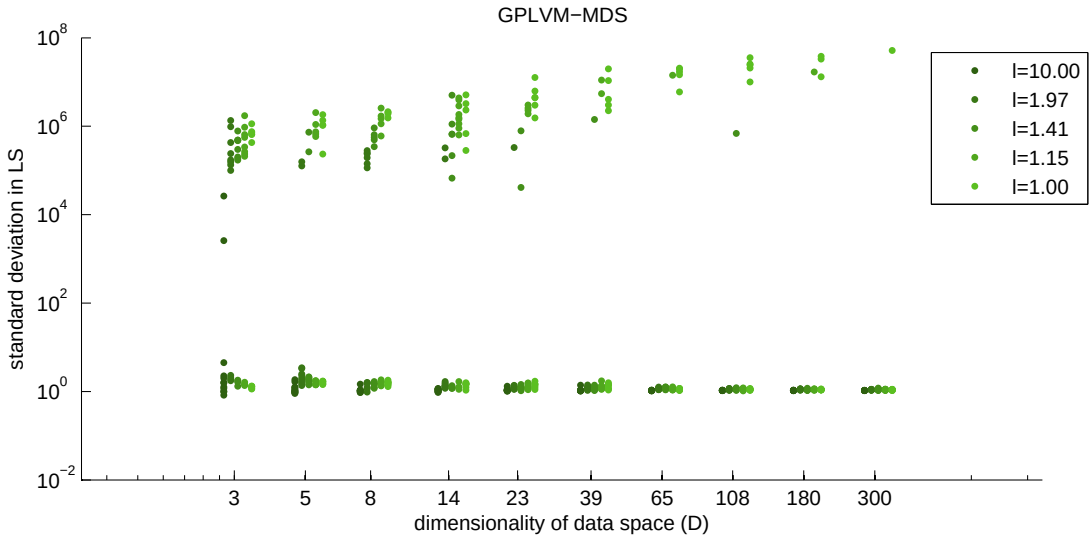
deviations of latent points were around 1 as expected, but, as Fig. 3.14(b) shows, after optimisation it increased up to 10^8 for some runs.

For latent points $\mathbf{Z}$ which result in covariances that do not match the covariances of the observed data $\mathbf{S}_N$ one way of increasing the likelihood is to make latent points independent by spreading them in latent space. When this happens, the model essentially discards the covariances in the data as sampling errors of the covariance matrix. In this way the model will not achieve a likelihood as good as with the true latent points, but this solution, unfortunately, still is a local maximum of the likelihood in which the optimisation then gets caught. In this case, the optimisation can even move latent points away from a reasonable solution. An effective solution for preventing this behaviour is to introduce a Gaussian prior over the latent points $\mathbf{z}_i \sim \mathcal{N}(0, \mathbf{I})$ which penalises a large scale (it prevents most of the large increases of nSSE in Fig. 3.14(a), but also does not lead to improvements in nSSE). This prior is used in Neil Lawrence's standard GPLVM implementation and we also employed it henceforth unless stated otherwise.

In the trials in which the optimisation did not spread out the latent points we still see a significant improvement of the likelihoods while the reconstruction accuracy after Procrustes analysis stayed close to the values before optimisation. This means, on the one hand, that the optimisation adapted the latent points such that the model fits the high-dimensional data, but, on the other hand, that the optimisation converged



(a) nSSE difference after−before optimisation



(b) standard deviation of latent points after optimisation

Figure 3.14: Analysis of latent points after optimisation for GPLVM-MDS as scatterplots including data of the 12 trials. x-axis: number of output dimensions (note that for each D points for different lengthscales have been plotted with a small offset for better visibility). Colour darkness codes for lengthscale (light: $\ell = 1$, dark: $\ell = 10$).

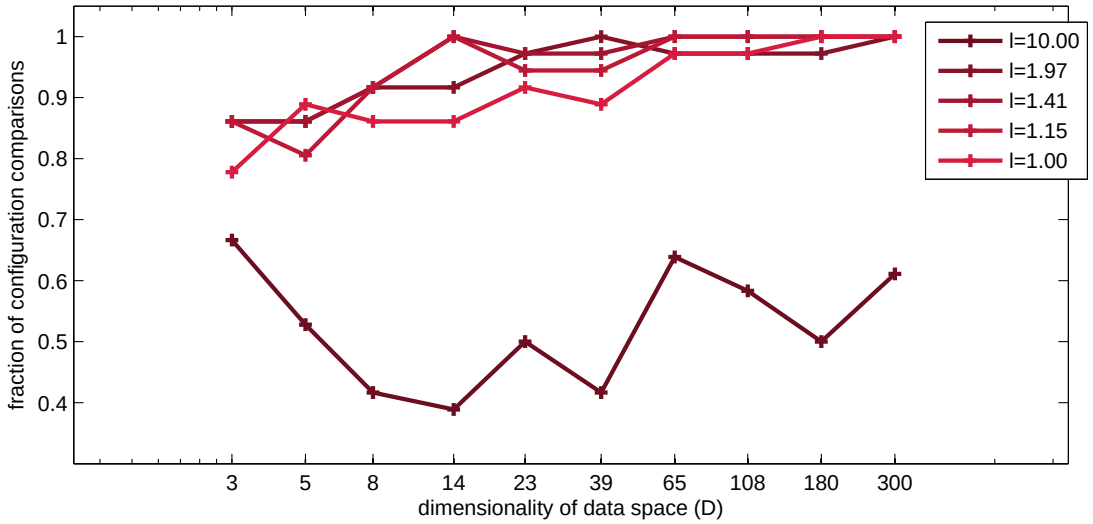


Figure 3.15: Fraction of latent configuration comparisons for which a higher likelihood also meant a lower reconstruction error. x-axis: number of output dimensions. Colour darkness codes for lengthscale (light: $\ell = 1$, dark: $\ell = 10$).

to a local optimum close to the initialisation, but not necessarily closer to the true configuration. While an improvement in likelihood after optimisation is not necessarily related to an improvement in reconstruction accuracy, we at least found that if the likelihood was larger for one of two configurations of latent points after optimisation, it usually also had a better reconstruction accuracy.

In particular, we considered the configurations resulting from optimisation when the GPLVM had been initialised with either the true, PCA or GPLVM-MDS configurations. For each of them we computed the resulting log-likelihood and nSSE to the true configuration of points after procrustes analysis[7]. In pairwise comparisons we then recorded if one of the configurations had a larger likelihood, whether then it also had a smaller nSSE to the true configuration of latent points. Fig. 3.15 shows the fraction of comparisons for which this was true over the 12 repetitions of our experiments. For data sets from 14 independent output dimensions upwards this was the case for 90 or more per cent of the comparisons except for the very smooth data with lengthscale $\ell = 10$ for which log-likelihoods were very close together. Therefore, we can take a larger likelihood after optimisation as an indicator for the quality of the found latent points. This is in line with the findings in Harmeling (2007) who compared model selection techniques for nonlinear dimensionality reduction problems.

[7]The configuration resulting from GPLVM optimisation when initialised with the true configuration is usually slightly different from the true configuration depending on the amount of noise.

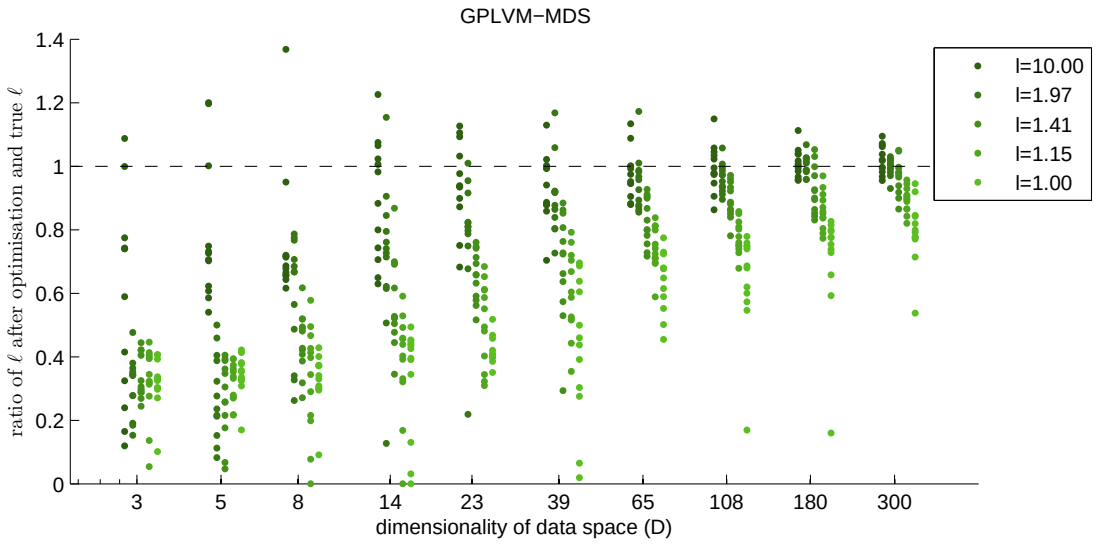


Figure 3.16: Ratio of optimised and true lengthscales ($\ell_{\text{opt}}/\ell_{\text{true}}$) with GPLVM-MDS initialisation. x-axis: number of output dimensions. Colour darkness codes for true lengthscale of corresponding data set (light: $\ell = 1$, dark: $\ell = 10$).

In these experiments the parameters of the covariance function were fixed to the values used during generation of the data. In practice we do not know these parameters and optimise them together with the latent points. We repeated the experiments above, but included the parameters of the covariance function in the optimisation. The exact covariance function that we used was

$$k(\mathbf{z}_i, \mathbf{z}_j) = \phi \exp\left(-\frac{1}{2\ell^2}|\mathbf{z}_i - \mathbf{z}_j|^2\right) + \delta_{ij}\sigma^2 \tag{3.19}$$

where δ_{ij} is the Kronecker delta. The parameters are ϕ, the output scale, ℓ, the lengthscale as before, and σ^2, the variance of independent Gaussian noise on the outputs.

We initialised the parameters with the true values which were $\phi = 1$, $\sigma^2 = 1e-7$ and ℓ as before. Ideally the parameters, therefore, will not change during optimisation. However, this is not the case. Especially the lengthscale is variable during optimisation and tends to be underestimated (cf. Fig. 3.16). Notice that changing the lengthscale has the same effect on the covariance matrix as scaling all latent points $\mathbf{Z}$ by a common value. Therefore, we expect that optimisation results in extremely small lengthscales in some cases, equivalent to the increase in scale of latent points above. The results in Fig. 3.16 confirmed that this occasionally happens (points close to 0), but far less frequently than the explosion of latent point scale in Fig. 3.14(b). These effects can also be prevented by introducing a prior on the lengthscale which encourages larger

lengthscales[8], but this does not prevent the lengthscale from being underestimated (results not shown). Fig. 3.16 also shows that the optimised lengthscales get closer to the true lengthscales for increasing D for GPLVM-MDS. We think that this is an effect of the increased accuracy of reconstructed configurations. For PCA, on the other hand, lengthscales are also underestimated with many output dimensions ($\ell_{\text{opt}}/\ell_{\text{true}}$ around 0.3 in most experiments with $\ell_{\text{true}} < 1.5$) due to a lack of improvement of found latent configurations.

The increased number of degrees of freedom in the optimisation also led to improved log-likelihoods, but again without moving latent points closer to the true configuration (compare Fig. 3.17 to Fig. 3.13 and Fig. 3.14(a)).

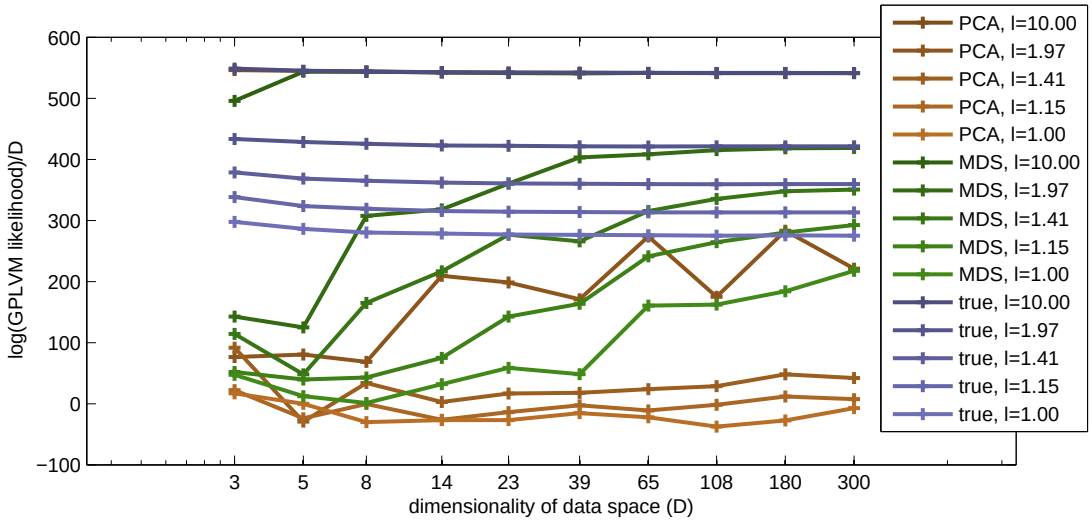
In summary, only when the initialisation was very close to the true configuration of latent points, the optimisation of the GPLVM likelihood moved the latent points towards the true configuration. By using simple Gaussian priors on latent points and lengthscale the optimisation could be prevented to find a trivial, globally non-optimal, solution, but these priors also did not help in finding the true configuration of points.

Optimisation increased log-likelihoods by several orders of magnitude such that their resulting scale was comparable to the log-likelihood of the true configuration. While this was independent of an improvement in reconstruction accuracy of the true latent configuration (likelihood may improve without accuracy improving), we still found that there is a good correspondence between the optimised log-likelihood and how close the found latent points were to the true configuration. This means that we cannot draw conclusions about the quality of found latent points from the improvement in log-likelihoods from before to after optimisation, but that the log-likelihoods after optimisation give us an indication for which configuration of latent points is better (as long as they are not too close together).

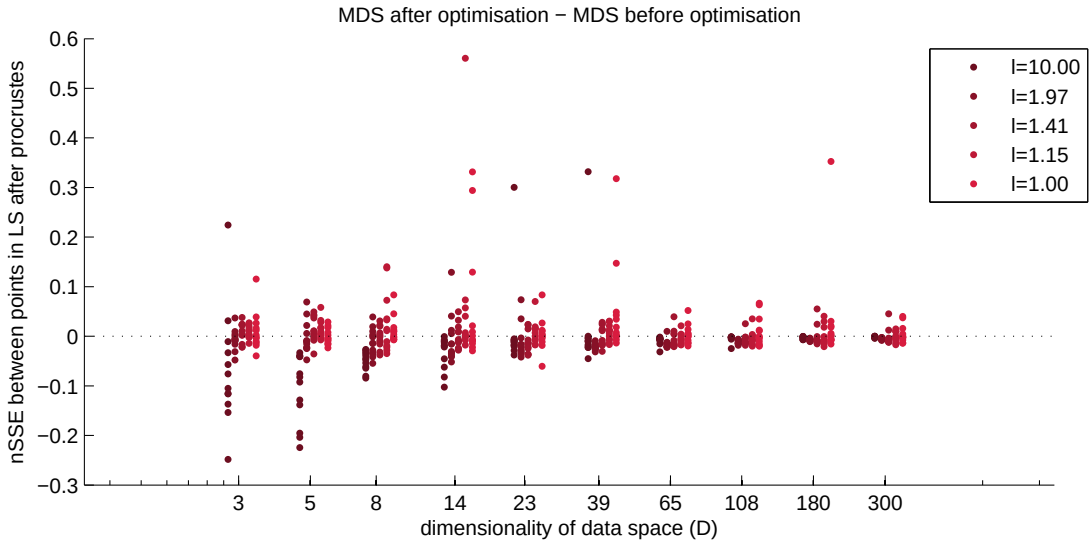
3.4.3 Comparison to Isomap

Isomap (Tenenbaum et al., 2000) has previously been suggested as an alternative initialization when PCA fails to uncover the principal structure of a nonlinear data set (Lawrence, 2005). Here we briefly compare GPLVM-MDS to Isomap on the synthetic data. Isomap also employs MDS in its final step, but the distances are approximated from a neighborhood graph which depends on the number of nearest neighbors k, a free parameter. In our experiments we computed solutions for $k \in \{1,\ldots,50\}$ and selected

[8] We implemented this as Gaussian prior on ℓ^{-2}: $P(\ell^{-2}) \sim N(0,1)$, but note that $\ell > 0$ and that this is therefore no valid prior. However, it can still be seen as an ad-hoc regulariser with a similar effect.



(a) normalised GPLVM log-likelihood



(b) nSSE difference after−before optimisation

Figure 3.17: Results of GPLVM optimisation of latent points and covariance function parameters. x-axis: number of output dimensions. Colour darkness codes for lengthscale (light: $\ell = 1$, dark: $\ell = 10$).

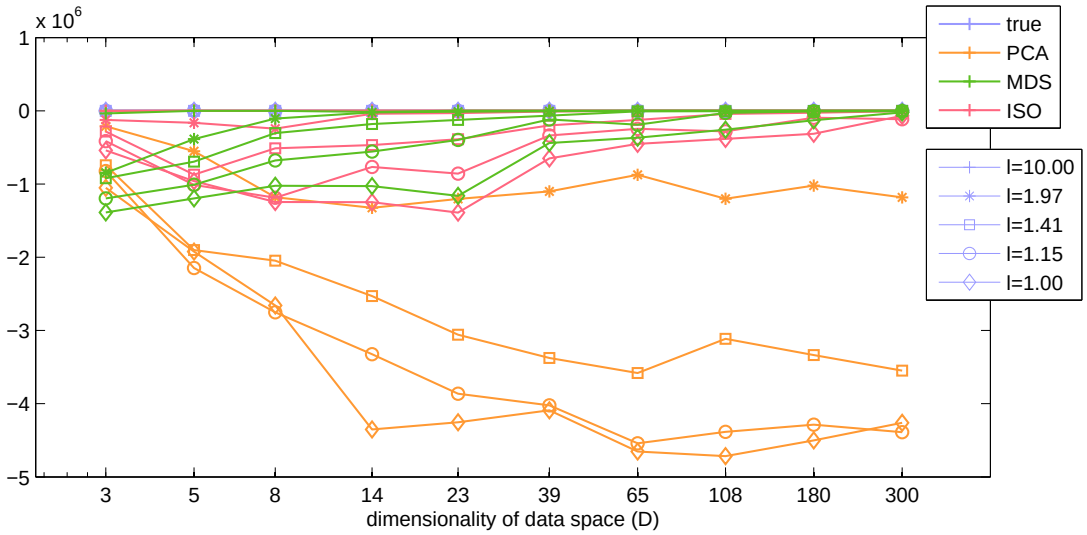
the one with the highest GPLVM log-likelihood as initialization for the GPLVM. To allow for a fairer comparison between the two methods we here also used the GPLVM log-likelihood to select the best repetition of iterative MDS optimisation for GPLVM-MDS instead of the stress criterion[9]. Resulting normalised log-likelihoods are shown in Fig. 3.18. On many data sets GPLVM-MDS and Isomap gave very similar results. However, there were some data sets on which the likelihood of the GPLVM-MDS solution was considerably larger than that of the Isomap solution leading to a slight advantage of GPLVM-MDS on average across data sets, as can be seen in the figure. Even though only means are shown in the figure, we found that GPLVM-MDS had larger log-likelihoods in the majority of cases, although these were not as frequent as with PCA (71% of data sets with $D \geq 39$ and $\ell < 10$ before and 60% after optimisation). Although GPLVM-MDS also maintained an advantage over Isomap, we found that Isomap provides similar benefits over PCA as GPLVM-MDS.

3.5 Motion Capture Data

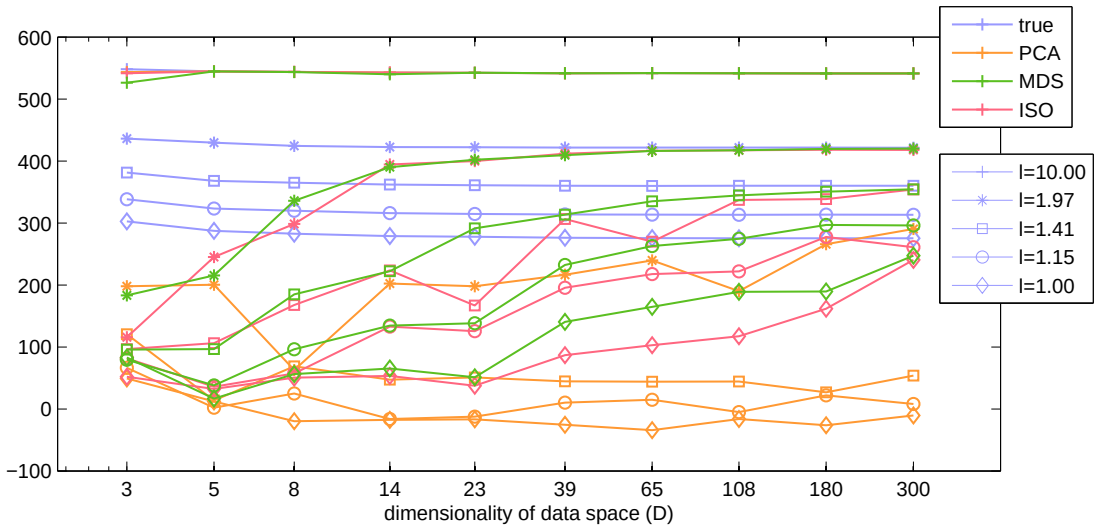
The analysis with synthetic data had two advantages: we knew 1) ground truth for the latent points and 2) that the model was correct for the data. In this section we test GPLVM-MDS on motion capture data where we do not know ground truth and cannot be sure that our model applies. The only quantitative evaluation available to us is based on the likelihood of the GPLVM. We then have to infer the quality of the found latent representations from the achieved likelihoods for the different initialisations. The results on synthetic data suggest that we can infer from the likelihood the quality of the found latent spaces, but as these results are based on data which matched the model, results with data which do not necessarily match the model have to be interpreted with care.

We tested GPLVM-MDS on 6 different motion capture data sets. The first two are our own and represent punches of a single person. The first are the full, recorded movements of three punches while in the second we cut out the retraction at the end of the punches (see Section 2.1 for details). The remaining data sets have been used in other publications to demonstrate the working of the GPLVM and its variants. In particular, data set 3 (running) has been used in Lawrence and Quinonero-Candela (2006) and data sets 4 (walking of a single person), 5 (walking of 4 different people)

[9] In contrast to experiments before we chose $R = 100$ to save computational time, but this did not reduce performance of GPLVM-MDS.



(a) before optimization



(b) after optimization

Figure 3.18: Comparison of initialisations based on normalised GPLVM log-likelihood. x-axis: number of output dimensions. y-axis: Mean over 12 trials of normalised log-likelihood for GPLVM-MDS, Isomap, PCA and the true configuration of points. Color codes for method of initialization, marker symbols for lengthscale ℓ of data as shown in figure legends.

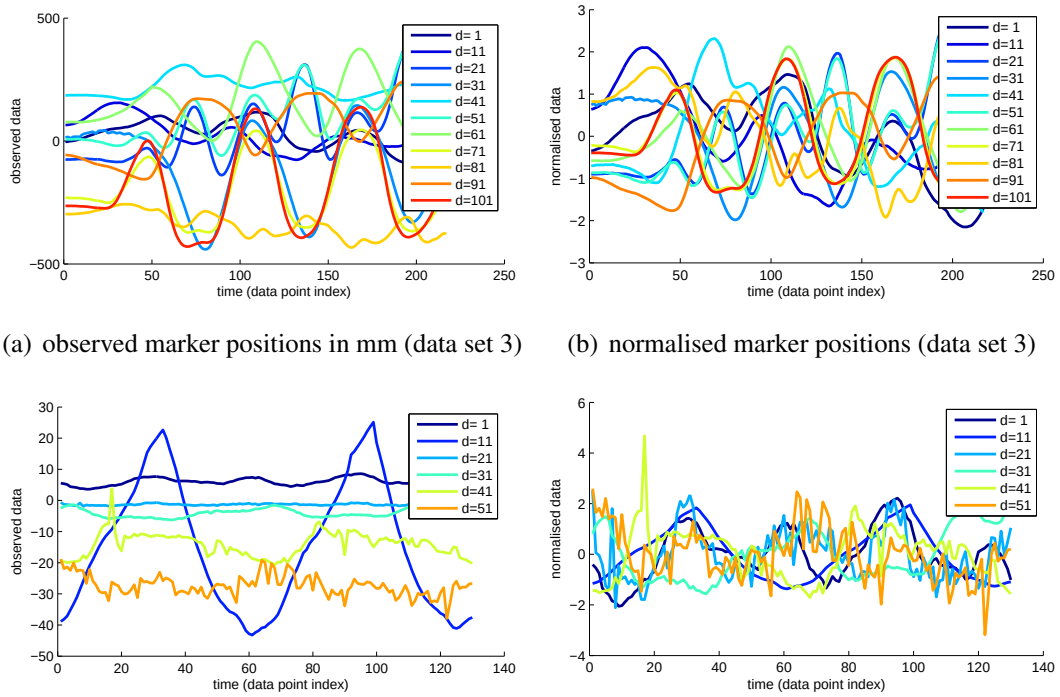
and 6 (4 golf swings of a single person) in Wang et al. (2008). The motion in data set 3 was stored as the raw xyz-positions of 34 motion capture markers and therefore has 102 dimensions. In this type of representation a lot of variation in the data is contributed from the spatial translation of the body in Cartesian space which is more an effect of the motion than a determining feature. We, therefore, followed the authors in Lawrence and Quinonero-Candela (2006) and removed the mean of the data in each frame what, approximately, corresponds to removing the translation of the body. To be precise, we subtracted the mean of the 34 markers within each frame separately for the x, y and z coordinates. Motions in all other data sets were represented as 3D Euler rotations between consecutive links of an associated skeleton[10] and we also removed the position of the root node determining the position of the body in Cartesian space in each frame. The remaining number of dimensions was 57 for data sets 1 and 2, 53 for data sets 4 and 5, and 52 for data set 6 while the number of data points in each data set was 264, 97, 217, 130, 288 and 255, respectively.

3.5.1 Latent space dimensionality

The method presented here leaves the choice of latent dimensionality M to the user. In our experiments on motion capture data we chose M according to our prior beliefs about the intrinsic dimensionality of the data. We justify our choices as follows.

The 3 punches in data set 2 are the same style of punch, but differ mainly in the height of the punching hand. Therefore, we expect a 2-dimensional latent space to be sufficient in which one dimension represents the state within the punch, i.e., time and the other the height of the punch. In data set 1 we additionally included the return to the starting posture after the punch and therefore add a third dimension to account for the cyclic nature of the movement. In data set 3 the recorded person breaks into a run from standing. Running is a cyclic motion and the person goes through roughly two cycles of running. For a cyclic motion with no other differences between parts of the motion a 2D latent space is sufficient. Because the person goes over into running from standing, however, the motion in data set 3 has an additional component which is, for example, expressed in different degrees of leaning forward when decreasing speed. We therefore set $M = 3$ for data set 3. The walk in data set 4, as a cyclic motion, may be represented in two dimensions only, but because the walks in data set 5 are from different people, an additional 3rd dimension has to be introduced which

[10] Angles ranged in $[-0.87, 0.57]\pi, [-0.87, 0.56]\pi, [-0.52, 0.54]\pi, [-0.47, 0.49]\pi$ and $[-1.12, 0.65]\pi$ for data sets 1,2,4,5 and 6, respectively.



(a) observed marker positions in mm (data set 3)

(b) normalised marker positions (data set 3)

(c) observed joint angles in degrees (data set 4)

(d) normalised joint angles (data set 4)

Figure 3.19: Example data trajectories before and after normalisation. (a,b) Data set 3 (running), each line corresponds to a marker trajectory in either x, y, or z of Cartesian space. (c,d) Data set 4 (walking), each line corresponds to Euler rotation around either the x, y, or z axis in one joint.

allows differences between subjects to be represented. The golf swings in data set 6 are executed by the same person in the same style, but because after the backswing the body goes through similar postures as the starting posture, the difference being movement in the hip, we allow for $M = 2$.

3.5.2 Normalisation of data

We did not need to normalise the synthetic data, because we knew that its distribution was correct. However, the motion capture data does not fulfil the assumptions of the model and needs to be normalised. In particular, normalisation can potentially rectify the violation of two assumptions of the GPLVM: 1) there is no offset on the observed variables and 2) all observed variables have the same scale, but normalisation can also introduce additional problems.

We have discussed in Section 3.1.2.1 that under the GPLVM it is not possible to differentiate between an offset in an observed variable and a genuine sample of a func-

tion with consistently large values when the lengthscale of the GP is large relative to the spread of the latent points. If we have data with a large lengthscale *and* an offset, removing the mean of the observed variables removes the offset, but also changes the covariances between data points and leads to a wrong estimate of the covariance matrix, similar to not removing the offset. In motion capture data we frequently observe rotations with an offset, because for any given motion the centre of a rotation in a single joint might be different. Without knowledge of the centre of rotation, the mean is our best guess, but for the small data sets used here and in particular for non-cyclic movements such as golf swings, it is expected that removing the mean still leaves considerable inaccuracies.

The parameter ϕ of the used covariance function (eq. 3.19) allows for output scales different from 1. However, the model still assumes that all observed variables have the same scale. Unfortunately this is not the case for motions represented with joint angles as for these motions the amplitude of rotations often varies considerably across joints. We rectify this by normalising the standard deviation to 1 for each observed variable after centring. Even though by taking this step we can subsequently model the data with a GPLVM, the normalised data (see e.g. Fig. 3.19) indicates that other assumptions of the model are subsequently violated. For example, the measurement noise of the motion capture system is scaled together with the observations, i.e., the assumption is violated that the noise for each observed variable is equal, but already in the raw joint angle data the trajectories appear to have varying amounts of noise, or different lengthscales (Fig. 3.19(d) is an example where this is particularly apparent).

3.5.3 Results

After normalisation we applied PCA, Isomap, or GPLVM-MDS[11] to the motion capture data sets and initialised a GPLVM[12] with the resulting latent points and covariance function parameters $\ell = 1, \phi = 1$ and $\sigma^2 = 0.01$. This allowed us to compute the GPLVM log-likelihood of the PCA, Isomap and GPLVM-MDS solutions as defined in eq. (3.8). Subsequently, we optimised the GPLVM using scaled conjugate gradients for 500 steps and recorded the log-likelihood of the result.

For GPLVM-MDS we first note that more than 50% of the entries in the normalised inner product matrix $\mathbf{S}_N$ were negative in all data sets (52,56,58,56,55 and 52% for the

[11] Setup for Isomap and GPLVM-MDS as in Section 3.4.3.

[12] The GPLVM had a Gaussian prior on latent points, but not on covariance function parameters. Introduction of this prior has no large effect on log-likelihoods.

(a) before optimization

	DS1	DS2	DS3	DS4	DS5	DS6
PCA	−458	−365	6	−569	−733	−141
Isomap	45	−54	**185**	−468	−597	23
GPLVM-MDS	**66**	**−5**	173	**−342**	**−426**	**43**

(b) after optimization

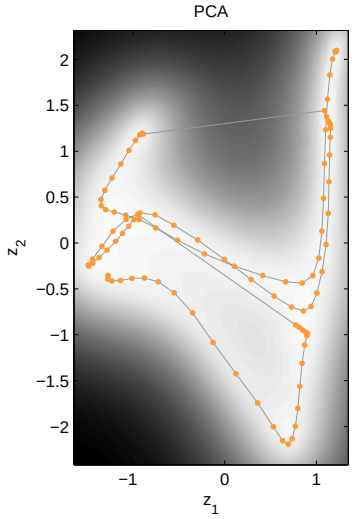
	DS1	DS2	DS3	DS4	DS5	DS6
GPLVM-PCA	635	200	587	−22	106	276
GPLVM-ISOMAP	658	204	**597**	**−19**	**109**	276
GPLVM-MDS	**689**	**208**	589	−21	107	**281**

Table 3.2: Normalised GPLVM log-likelihoods (L/D in eq. 3.8) for the 6 mocap data sets (1-uncut punches, 2-cut punches, 3-run, 4-walk, 5-walks of 4 people, 6-golf swings)

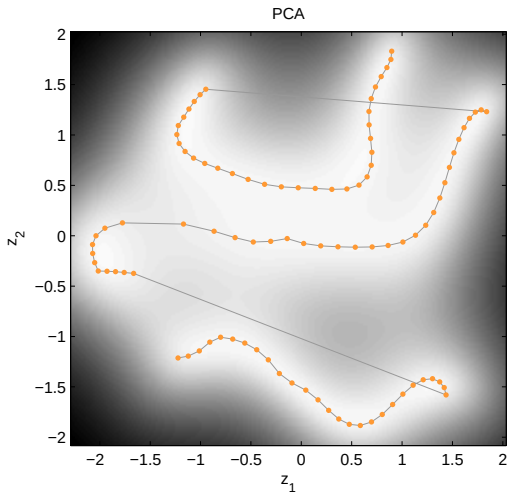
6 data sets). This indicates that either our model is wrong[13], or there is a large amount of noise in the data, as anticipated from the discussion of the normalisation step. From our experience with the synthetic data we, therefore, did not expect GPLVM-MDS to perform well on these data sets. Nevertheless, it consistently achieved better GPLVM likelihoods than PCA before and after optimization, although likelihoods after optimization were quite similar on three data sets (see Table 3.2). Apart from data set 3 GPLVM-MDS also gave larger likelihoods than Isomap before optimization and maintained this advantage on three data sets after optimization.

In Figures 3.20 and 3.21 we show the resulting 2D latent spaces for data sets 2 (punches without retraction) and 4 (walking). For the punches GPLVM-MDS had the largest GPLVM likelihood before and after optimisation. We see that the optimisation smoothed the latent configuration, separated the single (different) punches and in the case of PCA made the latent configuration more similar to the structure found by GPLVM-MDS. On the other hand, for the walk cycles in data set 4 the optimisation disrupted the periodic structure found to different degrees by the different initialisations. Also note that the scale of the data shrank. We interpret this as an indication for a discrepancy between model and data as already discussed in the context of normalisation in Section 3.5.2 (cf. Fig. 3.19). One particular influential assumption of the model is the dimensionality of the latent space. While we motivated our choice

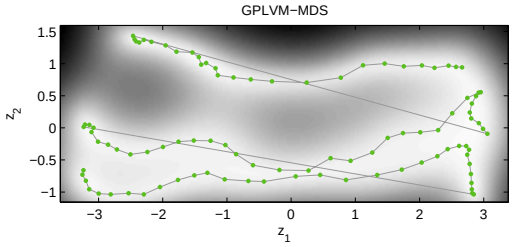
[13] It is, for example, conceivable that the motion capture data contains joints for which data points are genuinely anti-correlated in time, i.e. have negative covariance, but this cannot be modelled with the SE covariance function.



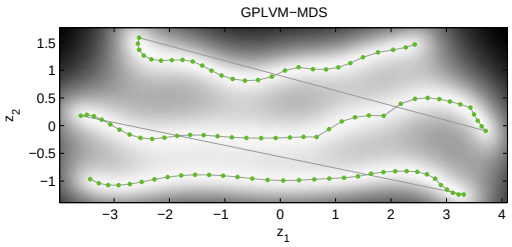
(a) PCA, before optimisation (-365)



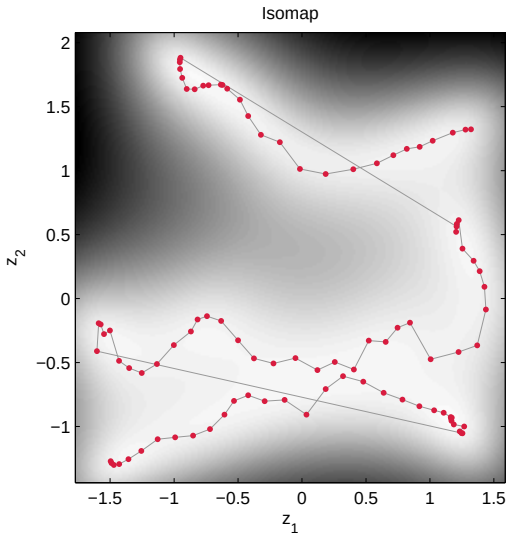
(b) PCA, after optimisation (200)



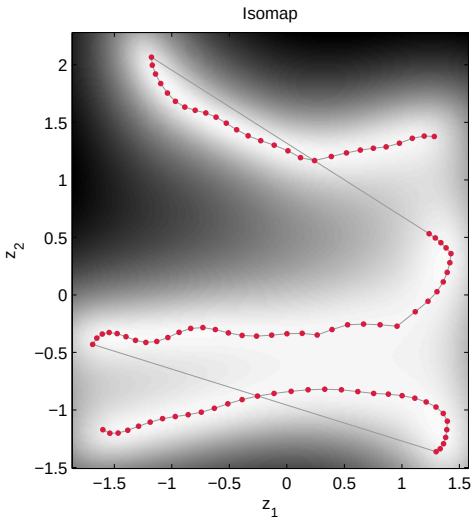
(c) GPLVM-MDS, before optimisation (-5)



(d) GPLVM-MDS, after optimisation (208)

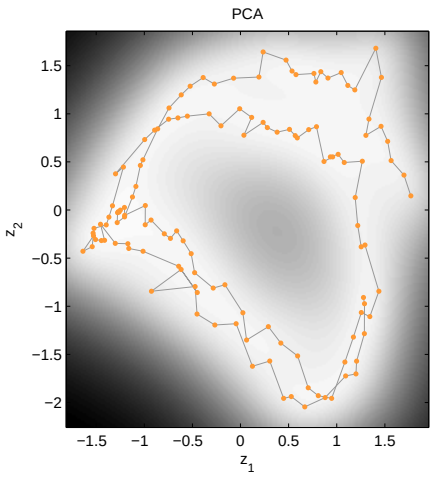


(e) Isomap, before optimisation (-54)

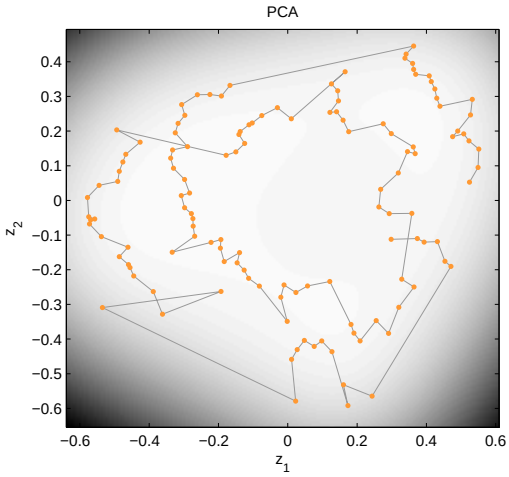


(f) Isomap, after optimisation (204)

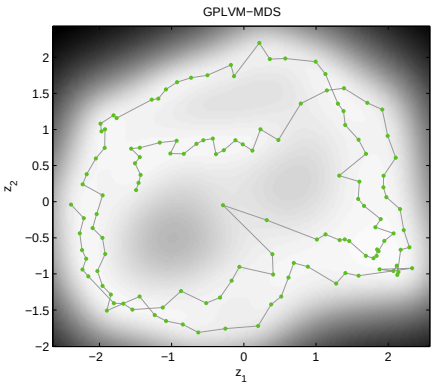
Figure 3.20: Latent points and log GPLVM confidences before and after GPLVM optimisation for data set 2 (3 punches without retraction). Numbers in parantheses are normalised log-likelihoods repeated from Table 3.2.



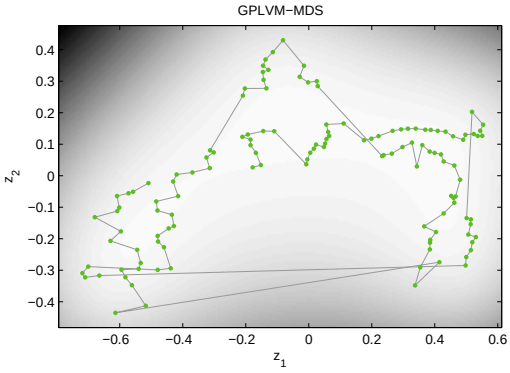
(a) PCA, before optimisation (-569)



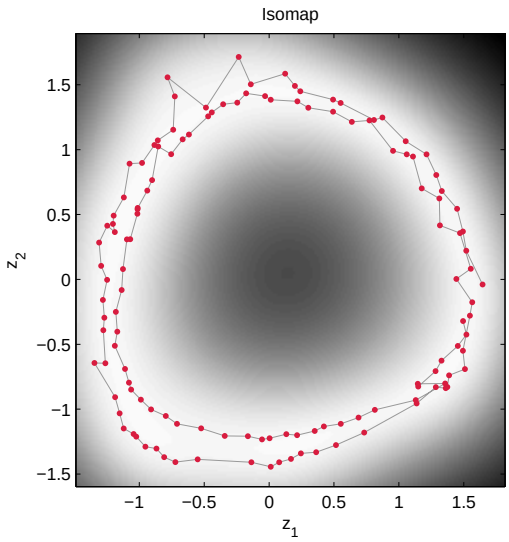
(b) PCA, after optimisation (-22)



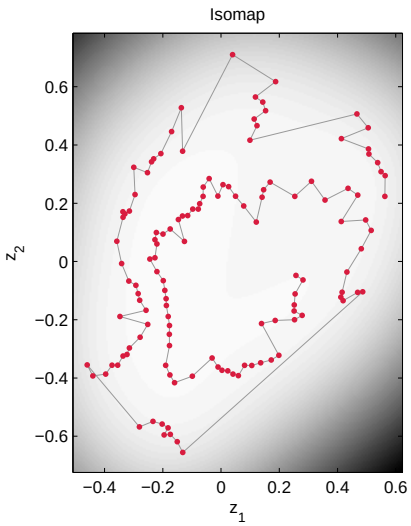
(c) GPLVM-MDS, before optimisation (-342)



(d) GPLVM-MDS, after optimisation (-21)



(e) Isomap, before optimisation (-468)



(f) Isomap, after optimisation (-19)

Figure 3.21: Latent points and log GPLVM confidences (shading) before and after GPLVM optimisation for data set 4 (walking). Temporal order of data points indicated with grey connecting lines. Numbers in parantheses are normalised log-likelihoods repeated from Table 3.2.

$M = 2$ it could still be that unexpected features in the data necessitate larger M. Indeed, choosing $M = 3$ prevented optimisation from disrupting the periodic structure of the data and increased the likelihoods. Although the latter is not surprising in general, we also observed that GPLVM-MDS maintained its advantage over Isomap for $M = 3$ (normalised log-likelihoods after optimisation: 112, 112, 114 for PCA, Isomap and GPLVM-MDS, respectively).[14]

3.5.4 Conclusion

Given that more than 50% of distances were missing in all GPLVM-MDS experiments and having our results on the synthetic data in mind, it is surprising that GPLVM-MDS actually produced interpretable latent configurations. We believe that this is due to the sequential nature of the motion capture data sets which means that every data point always has neighbours with which it has a large covariance. This is important for the reconstruction accuracy, because we have seen that large covariances are better estimated in the sample covariance matrix than small covariances and, therefore, lead to better distance estimates.

We saw a clear advantage of GPLVM-MDS over PCA and Isomap before GPLVM optimisation (except for Isomap on data set 3). The latent configurations found by GPLVM-MDS also better reflect our intuitions of the underlying structure of the tested motions, although the same could be argued for Isomap and it remains a subjective view open to discussion. After optimisation PCA and in particular Isomap caught up in terms of likelihood. While PCA likelihoods stayed below those of GPLVM-MDS, Isomap overtook GPLVM-MDS on two data sets. It is difficult to judge the workings of an optimisation process with a large number of nonlinearly related parameters and it is, therefore, not clear what the reason for this is. Our analysis shows, however, that some problems on the tested motion capture data sets can be attributed to the GPLVM itself, i.e. the mismatch between model assumptions and observed data.

3.6 Discussion

In this chapter we have derived a relationship between metric MDS and the GPLVM which allows us to shortcut the optimisation of the GPLVM likelihood using an iterative MDS procedure. The resulting method is a particular instance of metric MDS

[14] In 3D it becomes clear that the smooth circle in Fig. 3.21(e) is actually a projection of two perpendicular half-circles to a plane.

based on the inverse of an isotropic covariance function. Our experiments on synthetic data have shown that, in contrast to naive GPLVM optimisation of a random, initial configuration[15], GPLVM-MDS can indeed find a close approximation of the true configuration of latent points. GPLVM-MDS configurations were also closer to the true configurations than the PCA results.

We have to pay for the increased performance achieved by GPLVM-MDS with its increased computational cost compared to PCA initialisation. The missing entries in the reconstructed distance matrix force us to use an iterative MDS optimisation which also needs to be initialised at random. While PCA adds just one more $O(N^3)$ step to the GPLVM optimisation, each iteration of MDS costs $O(N^2)$ and MDS optimisation is repeated R times. With the values of R used in our experiments GPLVM-MDS can be up to twice as slow as GPLVM-PCA. In our experience, R needs to be increased for problems with many missing distances which makes these problems also computationally demanding. Isomap, on the other hand, needs to be repeated a fixed number of times for different neighborhood sizes k which may depend on the number of data points N, but not on the underlying difficulty of the data. Thus, GPLVM-Isomap may also have a slight computational advantage over GPLVM-MDS on difficult data sets with many missing distances, like the motion capture data, even though their computational complexities are in general very similar. Nevertheless, we find that in an offline setting the improved accuracy of the results, in particular compared to PCA, justifies the higher computational cost.

Before optimisation the GPLVM-MDS configurations clearly led to better likelihoods, but the gap between GPLVM-MDS, Isomap and PCA shrank after optimisation. A likely reason for this is that some aspects of the GPLVM assumptions were not fulfilled, in particular, the assumption that all observed variables are generated from a single, underlying process. Even though this is a fundamental problem for the GPLVM, the learnt model could still generate the motion capture data used for training with low error even from only 2 or 3 latent dimensions. However, simple reconstruction of observed data points is not sufficient for the smooth interpolation between them and consequently the generation of new motions. To obtain smoother latent representations Wang et al. (2008) introduced an autoregressive prior on the latent points. If additional constraints about the modelled motions are known, more structured priors should be employed. In the next chapter we introduce representations of motion which

[15] Experiments not shown, but even after GPLVM optimisation the average nSSE to the true configuration was just under 1.

are particularly suited for control and motivate the template-based prior presented in Section 4.3.

Chapter 4

Dynamic Movement Primitives in Latent Spaces

Dynamic movement primitives (DMPs) have been proposed as a powerful, robust and adaptive tool for planning robot trajectories based on demonstrated example movements. Adaptation of DMPs to new task requirements becomes difficult when demonstrated trajectories are only available in joint space, because their parameters do not in general correspond to variables meaningful for the task. This problem becomes more severe with increasing number of degrees of freedom and, hence, is particularly an issue for humanoid movements. We show in Section 4.2.3 that DMP parameters *can* directly relate to task variables, when DMPs are learnt in latent spaces resulting from dimensionality reduction of demonstrated trajectories. In general, however, standard dimensionality reduction techniques do not provide adequate latent spaces, as they need to be highly regular.

In this work we concentrate on learning discrete (point-to-point) movements and propose a modification of the GPLVM in Section 4.3 which makes it more suitable for the use of DMPs by favouring latent spaces with highly regular structure. We motivate our approach on data from the 7-DOF DLR arm in Section 4.4.1 and demonstrate its feasibility on high-dimensional human motion capture data in Section 4.4.2.

4.1 Introduction

The Dynamic Movement Primitive (DMP) framework (Ijspeert et al., 2002) provides representations particularly suited for robot programming by demonstration. DMPs are nonlinear dynamical systems which are learnt such that an example movement is the

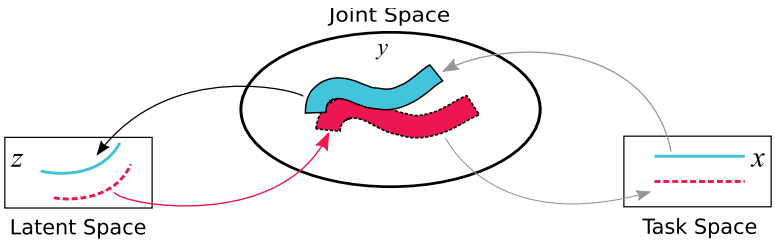


Figure 4.1: Schematic of experimental setup: Given are demonstrations in joint space (blue). They have corresponding trajectories in task space, but we do not in general know them. We infer a latent space through DR in which DMPs are learnt. Parameters of DMPs are changed to generate new trajectories (red) which are translated to joint space through the generative DR mapping.

attractor of the system. Hence, DMPs are control policies which can *robustly* replicate demonstrations. Additionally DMPs have parameters which allow to change the speed of the learnt dynamics and shift and stretch it in its state space. These parameters can potentially be used to adapt a learnt movement to new situations. However, when DMPs are learnt in joint space, as originally proposed, it is not in general the case that a change in DMP parameters has an effect meaningful for the given task (see Section 4.2.2), or maintains other desirable properties of the demonstrated movements such as resolution of null space in a naturally looking way. For low-DOF (degree of freedom) robots, for which inverse kinematics is easily solvable, a solution is to learn DMPs in task space (Pastor et al., 2009) where DMP parameters directly relate to task variables. For full-body humanoid movements, however, learning how to resolve redundant degrees of freedom in a natural way is an important aspect of learning from demonstration.

Here we suggest to use dimensionality reduction (DR) to infer spaces from movement trajectories demonstrated in joint space, in which DMP parameters can be related to task variables. Fig. 4.1 shows a schematic of the different spaces involved. The task space positions ($\mathbf{x}$) are only known in control experiments. In general we observe joint angles ($\mathbf{y}$) and infer latent points ($\mathbf{z}$) with dimensionality reduction. We learn DMPs in latent space (blue, solid line) and change their parameters to generate new motion (red, dashed line) which should have a correspondent in task space.

This allows us to generate new motion which fulfils different task goals by a simple change of DMP parameters while maintaining the overall style of the demonstrated movements. While we show that application of standard dimensionality reduction methods can lead to acceptable results in Section 4.2.3, further experiments also re-

vealed that in many cases resulting latent spaces are not sufficiently regular to be used in combination with DMPs. This is, in particular, an issue with noisy, high-dimensional data from human motion capture, but it can already be observed for motion of a redundant 7-DoF robotic arm. In Section 2.3.1 we used such a system (the DLR light-weight robot arm) to extensively evaluate the capability of a range of dimensionality reduction methods to produce latent spaces which simplify point-by-point interpolation of robot poses - a problem less demanding than the one we consider here. These experiments showed that the usefulness of the latent space strongly depends on the data set, the method used and its parameters. Here we extend this analysis to the case in which we aim to interpolate whole trajectories by only changing parameters of a learnt DMP and show that no standard DR method consistently gives acceptable results. After introducing DMPs in the next section, we, consequently, suggest a modification of the GPLVM which favours the use of DMPs in latent space by requiring it to be more regular. We present results on the DLR data set in Section 4.4.1 and finally evaluate our approach on human motion capture data in Section 4.4.2.

4.2 Dynamic Movement Primitives

The dynamic movement primitive framework is based on work presented by Ijspeert (Ijspeert et al., 2002, 2003). A similar formulation with dynamical systems has been used to let a humanoid perform a drumming task (Degallier et al., 2006) as a combination of discrete and periodic DMPs. However, there was no robot learning involved in this study and parameters for the DMPs had to be found by hand. Gams et al. (2009) extended Ijspeert's framework to simultaneously learning the frequency in the periodic case. DMPs have also been used as compact representations of movements for reinforcement learning (Peters and Schaal, 2006; Guenter et al., 2007). These approaches also aim to adapt a demonstrated movement to a new situation, but in order to do so they change the representation of the dynamical systems directly which leaves their parameters non-interpretable. This means that for each new situation the representation has to be tediously learnt anew. Recently a reformulation of DMPs has been proposed (Hoffmann et al., 2009; Pastor et al., 2009) which is suited for applying DMPs in task space and allows for automatic obstacle avoidance. Potentially this formulation can equally be applied to latent spaces, but for this study we have chosen to use the formulation presented in the following section.

4.2.1 Formulation

Our formulation is based on that by Ijspeert et al. (2003) who presented solutions for discrete, goal-directed movements (e.g., reaching) as well as periodic movements. It can be argued that these two types of movements are inherently different (Schaal et al., 2004). Here we concentrate on the former: discrete movements.

Discrete DMPs are characterised by a start state, y_0 [1], a trajectory of state variable y and a goal state, g. The state trajectory is represented by the nonlinear, second order dynamical system

$$\begin{aligned} \frac{1}{\tau}\dot{v} &= \alpha_v(\beta_v(g-y)-v) + \frac{g-y_0}{g^*-y_0^*}f(\xi) \\ \frac{1}{\tau}\dot{y} &= v \end{aligned} \tag{4.1}$$

Ignoring the modulating function f, this is a linear, two-dimensional dynamical system with a single, attracting stable point at $[g,0]$ (for appropriate setting of α_v and β_v). The nonlinearity is introduced through f which is used to shape the trajectory of the dynamical system between y_0 and g. It can be represented as a weighted sum of RBF basis functions

$$f(\xi) = \xi\frac{\sum_{i=1}^{n}\Psi_i(\xi)w_i}{\sum_i \Psi_i(\xi)} \qquad \Psi_i(\xi) = \exp\left(-h_i(\xi-c_i)^2\right) \tag{4.2}$$

which depend on the state, ξ, of a canonical system that converges to 0

$$\frac{1}{\tau}\dot{\xi} = -\alpha_\xi\xi. \tag{4.3}$$

At the beginning of a trajectory we always set $\xi = 1$. The influence of f on the dynamical system, therefore, vanishes towards the end of the trajectory. ξ thereby ensures that the goal, as a point attractor of the underlying linear system, is always reached. The number of basis functions, n, their width and centres, $h_i, c_i \in [0,1]$, and the parameters of the linear dynamical systems, $\alpha_v, \beta_v, \alpha_\xi > 0$, are chosen a priori[2]. Given a complete movement $[\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}}]$, the weights, w_i, of the nonlinear component are learnt. Once the movement is learnt, or, in other words, encoded as a DMP with start state y_0^* and goal g^*, we can replicate the demonstrated trajectory by resetting $\xi = 1$, setting $y_0 = y_0^*$ and $g = g^*$ and evolving the dynamical system over time until $\xi = 0$. Since the learnt trajectory is an attractor of the dynamical system, it is robust against perturbations (see

[1] y_0 is actually always accompanied by $\dot{y}_0$, because this is a second order dynamical system. We determine $\dot{y}_0$ from the first two states in the data sequence and keep it fixed thereafter.

[2] If trajectories are smooth, our settings $n = 40$, c_i equally spaced in [0,1], $\alpha_v = 25, \beta_v = 25/4, \alpha_\xi = 25/3$ are sufficient. All our experiments were conducted with these settings.

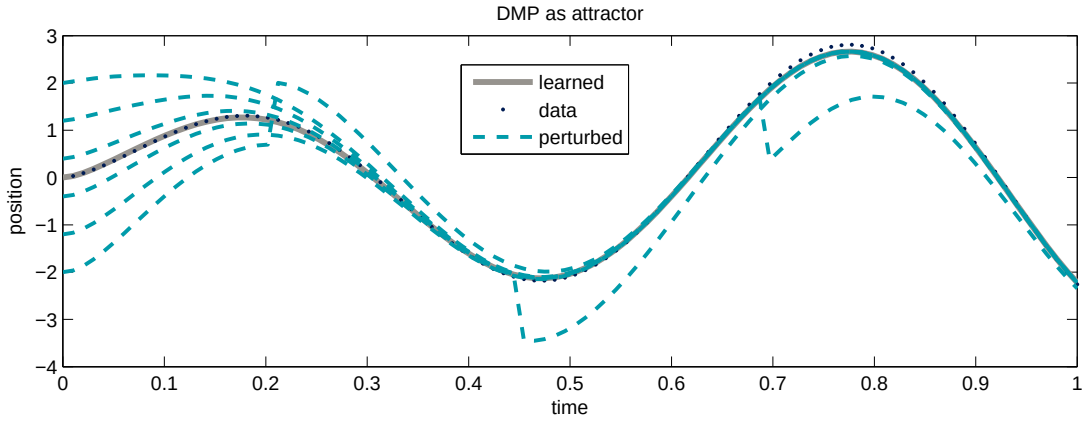
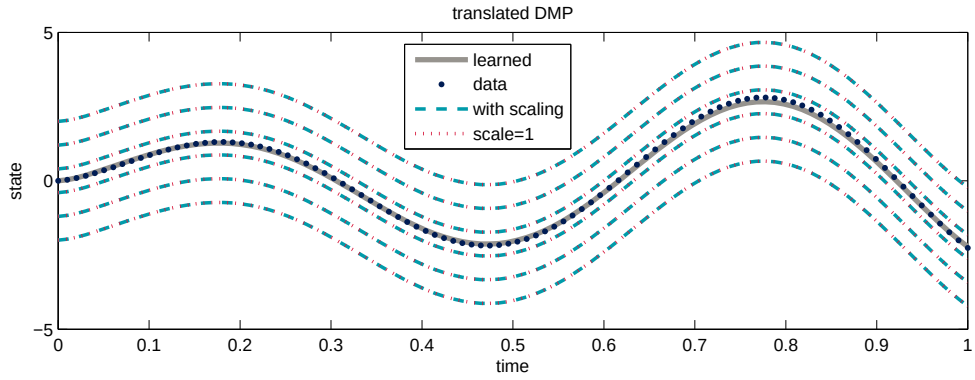


Figure 4.2: DMP as an attractor: dots are data, thick gray line is the learnt DMP and dashed lines are 6 different runs of the DMP which have been perturbed at the start and randomly during the movement.

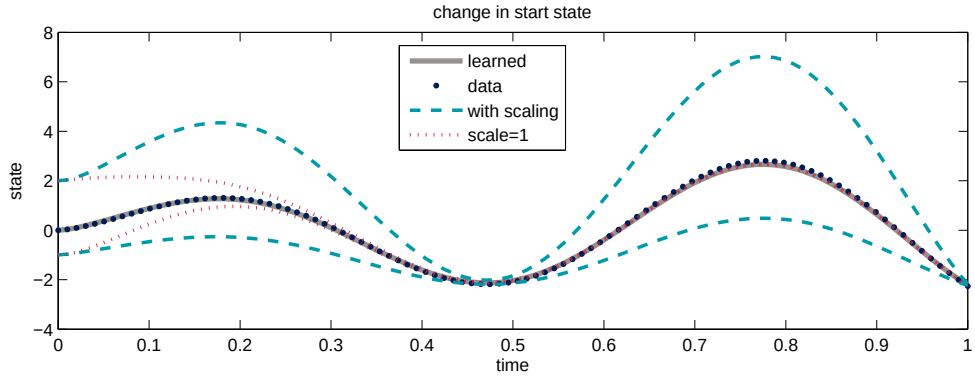
Fig. 4.2). We can modulate the DMP without having to update the learnt weights of f by changing parameters τ, y_0 and g. With τ we can speed up or slow down the motion and with y_0 and g we can produce qualitatively equivalent dynamics of motion in different parts of the state space.

We chose $\frac{g-y_0}{g^*-y_0^*}$ as scaling for f in eq. (4.1), because it allows for scaling and translation of the trajectory in state space. To be precise, if g and y_0 are translated by the same amount, the scaling of the trajectory is maintained while, if the difference between g and y_0 increases, the amplitude of the trajectory increases simultaneously (correspondingly for decrease). Division by $g^* - y_0^*$ is a convenience, setting the scale of the demonstrated trajectory to 1 and therefore simplifying potential manual adjustments of the scaling, if that is necessary (setting scale to 2 then doubles the amplitude of the trajectory).

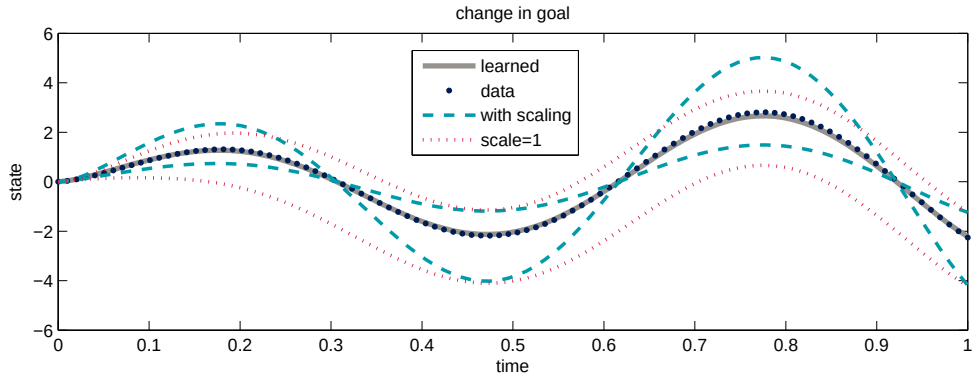
This is one possible choice for how a learnt trajectory can be generalised to different settings of g and y_0. A disadvantage of this choice is that, if the start state and goal of the demonstrated trajectory, g^* and y_0^*, are very close, already small changes in $g - y_0$ will have an extreme effect on the amplitude of the trajectory. The simplest solution in this case is to fix the scale at 1. Pastor et al. (2009) suggested an alternative DMP formulation which also sets a constant scaling. In general, however, this is not a satisfactory solution either, because a constant scale does not maintain the shape of the demonstrated trajectory at its start (see Fig. 4.3). In fact, with constant scale a change in start state of the DMP is equivalent to perturbing it at the start. A change in start state thus has no special meaning in this formulation.



(a) translation



(b) change in start state with fixed goal



(c) change in goal with fixed start state

Figure 4.3: Comparison of formulation with changing scale of DMP (eq. 4.1, blue, dashed lines) with formulation with fixed scale (red, dotted lines). (a) Common translation of start state and goal: both formulations translate the whole trajectory accordingly. (b) Translation only of start state: equivalent to perturbation for fixed scale, for changing scale trajectory is scaled. (c) Translation only of goal: for fixed scale trajectories translated according to goal and start state perturbed to original start state, for changing scale trajectories are also translated, but additionally scaled equivalently to (b).

The question, how demonstrated trajectories should be generalised for changing start and goal state, can ultimately not be answered without knowing the aim of the generalisation. In our experiments we, therefore, simply keep the formulation in eq. (4.1) and note that the GPLVM with isotropic covariance functions is invariant with respect to rotations of the latent space. Consequently, found latent spaces can be rotated in a suitable way to avoid small $g^* - y_0^*$ without changing GPLVM predictions.

4.2.2 Dynamic Movement Primitives in Joint Space

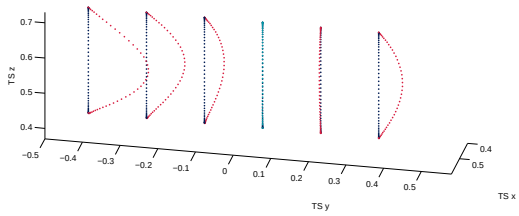
The formulation in eq. (4.1) is only defined for a single degree of freedom. To represent movements of robots or humans with several joints different dynamical systems can be learnt for each joint (note that we call the set of dynamical systems DMP, not the single dynamical systems). While a single, demonstrated movement can be represented in this way, it is unclear what effect modulation of the DMP through changing start state and goal in joint space has and, in particular, whether it is useful for the task and preserves constraints present in the demonstration (for example, how redundancy of plants with many DOF is resolved).

To illustrate this problem we revisit the DLR arm position data of Section 2.3.1 with interpolation width 5. Remember that the data consists of joint space trajectories which correspond to parallel, upwards trajectories in a plane in task space and that the redundancy of the DLR arm has been resolved consistently for all of them. In this experiment we selected one of the joint space trajectories and learnt a DMP[3] on it. We then modulated the DMP by changing its start state $\mathbf{y}_0$ and goal $\mathbf{g}$ to those of the other trajectories in the selected data set.

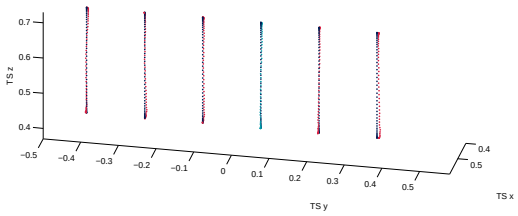
Exemplary results are depicted in Fig. 4.4. Modulation in joint space leads to large errors in task space (4.4(a)) as well as in the way redundant DOF are resolved[4]. Especially, the error of the modulated trajectories increases with the distance to the learnt trajectory. This is no surprise, because the single DMP has no information about the other trajectories which increasingly change their shape in joint space due to the nonlinearities in the inverse kinematics as the distance to the learnt trajectory increases.

[3] As there were 7 joints we learnt 7 dynamical systems.

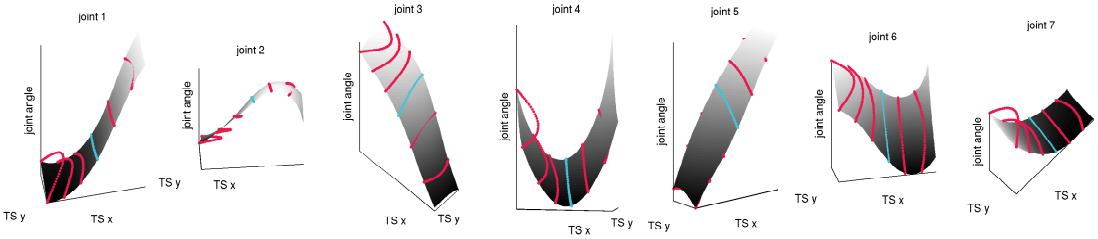
[4] Because the trajectories of the modulated DMP do not lie on the same task space plane as the data, Fig. 4.4(c) actually shows the combination of task space and redundancy resolution errors.



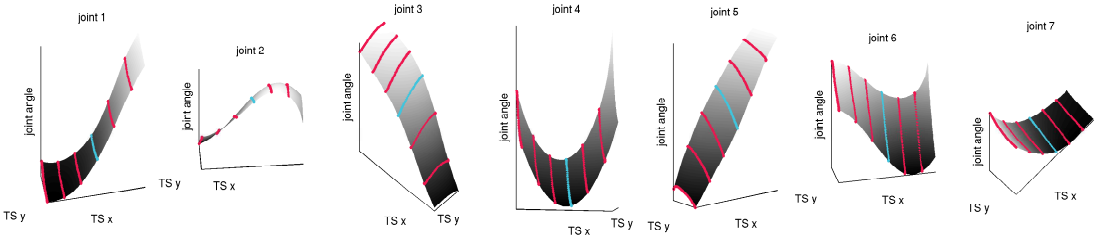
(a) task space, DMP in joint space



(b) task space, DMP in TmP-GPLVM latent space



(c) inverse kinematics, DMP in joint space



(d) inverse kinematics, DMP in TmP-GPLVM latent space

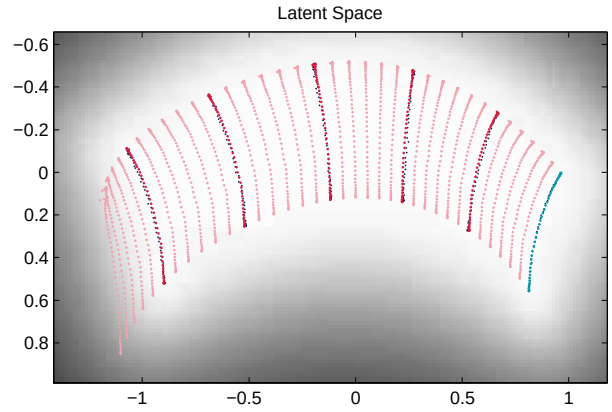
Figure 4.4: Modulating DMPs in joint space vs. modulating DMPs in latent space. (a,b) show task space trajectories (3D end-effector position, original data points lie in plane of that space). (c,d) depict the inverse kinematics function used to generate the DLR arm data together with trajectories generated from DMPs in joint space. These trajectories are visualised by evaluating the forward kinematics of the joint trajectories and plotting the resulting task space values against the selected joint value. (a,c) DMP learnt and modulated in joint space. (b,d) DMP learnt and modulated in TmP-GPLVM latent space as described in Section 4.3. In all plots: black dots - actual data, blue dots - DMP with learnt parameters, red dots - DMP with modulated start state and goal.

4.2.3 Dynamic Movement Primitives in Latent Space

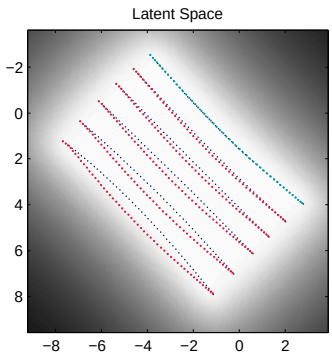
The results of the previous section clearly indicate that we need to account for the variability of the demonstrated trajectories, if we want to be able to generate meaningful motions by simply modulating the parameters of a DMP. In Chapter 2 we have seen that direct spline interpolation between postures of several demonstrated motion sequences already can produce accurate new sequences. So we could just interpolate a whole sequence pose by pose with splines and then learn a new DMP on this sequence directly in joint space. Even though this would give us a robust representation of the new sequence useful for control, we would still not be able to modulate the DMP online and would have to relearn the DMP for each new sequence. We propose, alternatively, to use DR to provide us with a space close enough to the underlying task space such that similar trajectories in task space are similar in the found latent space and can therefore be approximated by modulating a single DMP.

We reconsider the noisy DLR arm position data set with interpolation width 5 from Section 2.3. Fig. 2.6(b) suggests that, for example, the GPLVM with PCA initialisation and the GPDM with lines initialisation provide latent spaces which allow very accurate interpolation of trajectories, but are they also suited for DMP modulation? Fig. 4.5(a,b) shows the latent spaces for these methods together with the learnt and modulated DMP trajectories and Fig. 4.5(c,d) shows the corresponding task space trajectories. The representation of the demonstrated movements in the latent space of the GPLVM with PCA initialisation was conducive to the modulation of a DMP in this case. The changing shape of the demonstrated trajectories could be replicated to a large extent by modulation of start state and goal of a DMP learnt on one of them. So here the resultant latent space fitted well to the type of generalisation selected with our DMP formulation. Note, however, that there were small temporal shifts between the demonstrated and DMP trajectories which introduced errors in the directions of the trajectories and made them just fail our quality criteria defined in Section 2.3.1 (hence the low ratio of successfully interpolated trajectories in Fig. 4.7(a)).

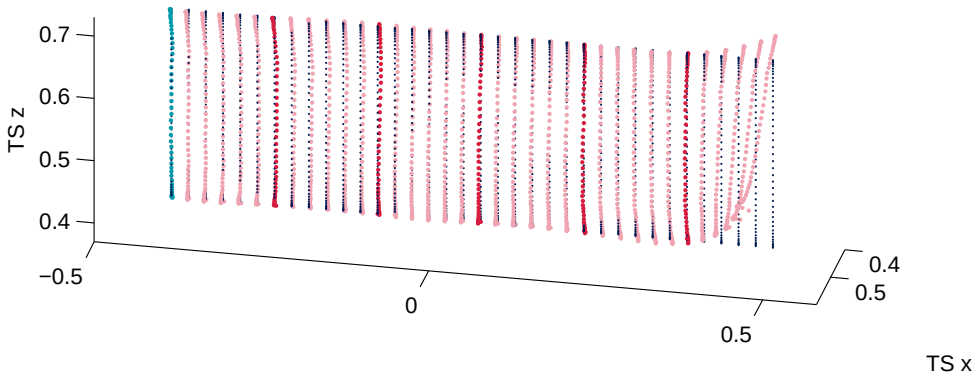
The GPDM with lines initialisation is an example in which DMP modulation clearly failed to reproduce the desired trajectories even though they could be accurately reconstructed with point-by-point spline interpolation. The reason is that the trajectory shapes in latent space changed differently to the way the DMP was generalised for changing parameters.



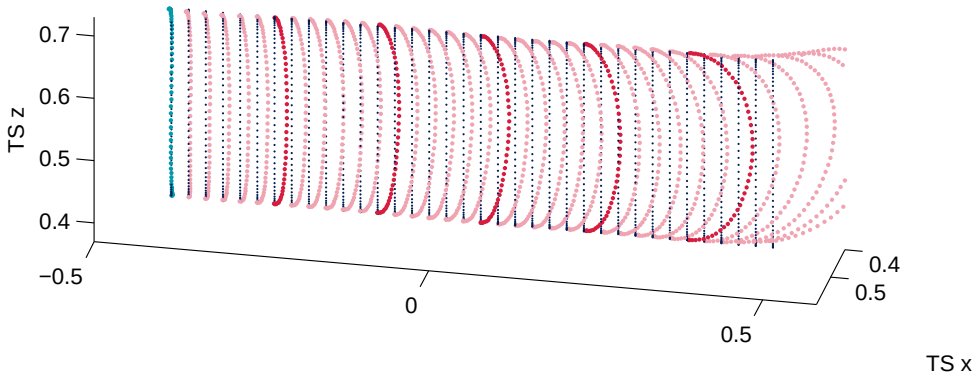
(a) GPLVM with PCA, LS



(b) GPDM with lines, LS

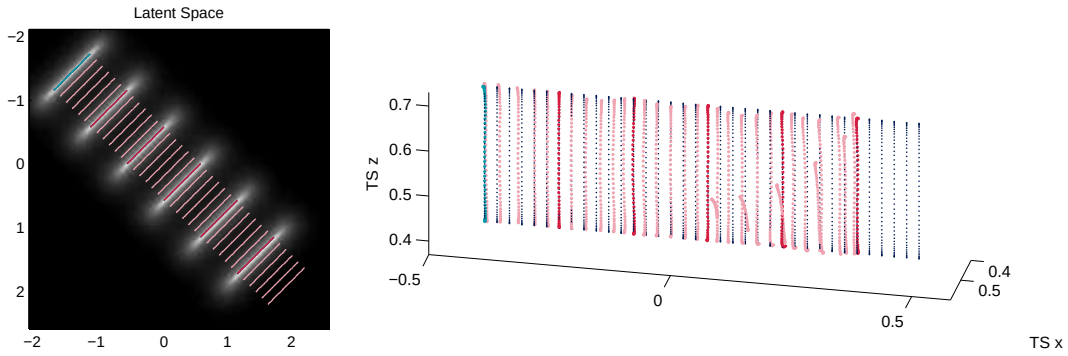


(c) GPLVM with PCA, TS



(d) GPDM with lines, TS

Figure 4.5: DMP applied in latent space of previously proposed methods (noisy DLR position data set). (a) Latent space of standard GPLVM with PCA initialisation. (b) Latent space of GPDM initialised with lines. (c) Task space corresponding to (a). (d) Task space corresponding to (b). Dark blue dots: data, light blue: learnt DMP, red: modulated DMP, pink: interpolated DMP.



(a) LS (b) TS

Figure 4.6: Completely predetermined, highly structured latent space for noisy DLR position data. Only GP parameters of the generative mapping have been learnt. (a) Latent space with DMP. (b) Corresponding task space. Dark blue dots: data, light blue: learnt DMP, red: modulated DMP, pink: interpolated DMP.

4.3 Latent Spaces for Dynamic Movement Primitives

A straight forward way to ensure that the latent trajectories of demonstrated sequences fit to the generalisation properties of the DMP is to ensure that the latent trajectories are just translated copies of a template (cf. Fig. 4.3(a)). An example of such a latent space for the DLR data is shown in Fig. 4.6. In this experiment we fixed the latent points to a set of parallel lines and learnt a GP mapping from these points to the joint space data. The predictive variance in Fig. 4.6 indicates that generalisation in this latent space was poor. The plotted task space trajectories confirm this: while the trajectories of the DMP fit the data well when modulated to the trajectories in the training data, DMP modulation for interpolation of trajectories produced inaccurate results. While this latent space was perfect for DMP modulation, it did not a good job in representing the data. So we need a method that can automatically determine good compromises between the strict structure of the latent space desired for DMPs and the freedom needed to represent the data well. Simultaneously we want flexibility to incorporate varying amounts of prior information into the latent space representation. For example, we might want to set the shape of the template by hand, but let the method determine the arrangement of the sequences. In the following we present a prior over latent points which incorporates these ideas in the GPLVM framework.

The idea is to let the data sequences in latent space be translations of a template $\bar{\mathbf{Z}} \in \mathbf{R}^{N \times M}$ where N is the number of data points in the template sequence and M is the

dimensionality of the latent space. In general the prior in latent space then is

$$P(\mathbf{Z}|\bar{\mathbf{Z}}) \sim GP(\mathrm{m}(\bar{\mathbf{z}}), \mathrm{k}(\bar{\mathbf{z}}, \bar{\mathbf{z}}')) \tag{4.4}$$

where *GP* stands for a Gaussian Process which has the template sequence as input. With S sequences in the data instead of only one the formulation becomes

$$P(\mathbf{Z}^D|\bar{\mathbf{Z}}) = \prod_{s=1}^{S} P(\mathbf{Z} = \mathbf{Z}_s^D|\bar{\mathbf{Z}}). \tag{4.5}$$

We want to restrict the GP to be linear and especially only allow translations. To achieve this we make use of the weight space view of a GP:

$$\mathbf{z} = \mathbf{A}\bar{\mathbf{z}} + \mathbf{b} = \mathbf{W}\hat{\mathbf{z}} \qquad \hat{\mathbf{z}} = [\bar{\mathbf{z}}, 1]^T. \tag{4.6}$$

To restrict the linear transformation $\mathbf{A}\bar{\mathbf{z}} + \mathbf{b}$ to only translations we set $\mathbf{A} = \mathbf{I}$ and let only $\mathbf{b}$ vary. So you have M GPs for which the linear weights should be set as

$$\mathbf{w}_1 = [1, 0, \ldots, 0, b_1], \mathbf{w}_i = [0, \ldots, 0, 1, 0, \ldots, 0, b_i], \mathbf{w}_M = [0, \ldots, 0, 1, b_M].$$

To achieve this we have to choose appropriate priors over the weights which are Gaussian with mean $\mu_i = [0, \ldots, 0, 1, 0, \ldots, 0, 0]^T$ and covariance Σ_i being all zero except for the lower left corner which is 1 (the variance of b_i is 1, all other parameters are fixed). The resulting mean and covariance function of the corresponding GP then is

$$\mathrm{m}_i(\hat{\mathbf{z}}) = \hat{\mathbf{z}}^T \mu_i = \hat{z}_i \qquad \mathrm{k}_i(\hat{\mathbf{z}}, \hat{\mathbf{z}}') = \hat{\mathbf{z}}^T \Sigma_i \hat{\mathbf{z}}' + \delta(\hat{\mathbf{z}}, \hat{\mathbf{z}}')\sigma_T^2 = 1 + \delta(\hat{\mathbf{z}}, \hat{\mathbf{z}}')\sigma_T^2. \tag{4.7}$$

So the conditional part of the prior for the whole data becomes

$$\begin{aligned} P(\mathbf{Z}^D|\bar{\mathbf{Z}}) &= \prod_{s=1}^{S} P(\mathbf{Z} = \mathbf{Z}_s^D|\bar{\mathbf{Z}}) \\ &= \prod_{s=1}^{S}\prod_{i=1}^{M} \frac{1}{\sqrt{|2\pi\hat{\mathbf{K}}|}} \exp\left[-\frac{1}{2}(\mathbf{z}_{si} - \hat{\mathbf{z}}_i)^T \hat{\mathbf{K}}^{-1}(\mathbf{z}_{si} - \hat{\mathbf{z}}_i)\right] \end{aligned} \tag{4.8}$$

$$\hat{\mathbf{K}} = \hat{\mathbf{K}}_i = \begin{bmatrix} 1+\sigma_T^2 & 1 & \ldots & 1 \\ 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 1 & \ldots & 1 & 1+\sigma_T^2 \end{bmatrix} \quad \hat{\mathbf{z}}_i = \begin{bmatrix} \hat{z}_{1i} \\ \vdots \\ \hat{z}_{Ni} \end{bmatrix} \quad \mathbf{z}_{si} = \begin{bmatrix} z_{1si} \\ \vdots \\ z_{Nsi} \end{bmatrix}$$

and we see that it simplifies to a Gaussian for each sequence in each latent dimension which has the corresponding template sequence as mean and equal covariance between

all data points in the sequence. The noise parameter σ_T controls by how much individual data points may diverge from the template sequence and, therefore, how successful DMP modulation will be. We can add both, the variables for the template $\hat{\mathbf{Z}}$ and the noise σ_T, to the GPLVM optimisation to let it automatically determine a good compromise between faithful representation of the data and our need for highly structured latent spaces. To avoid local optima problems it is recommended to also define priors over the additional variables. For the template sequence a Gaussian Process dynamics model as proposed by Wang et al. (2008) suggests itself such that the prior becomes

$$P(\mathbf{Z}^D, \bar{\mathbf{Z}}) = P(\mathbf{Z}^D|\bar{\mathbf{Z}})P(\bar{\mathbf{Z}}) \sim GP(\mathrm{m}(\bar{\mathbf{z}}), \mathrm{k}(\bar{\mathbf{z}}, \bar{\mathbf{z}}'))GPDM(\bar{\mathbf{Z}}).$$

We can initialise the template, for example, with the mean of the initialisations of the observed sequences.

Even with the GPDM prior on the template, however, the optimisation still did not converge to solutions which were suitable for DMP modulation in first experiments. So we increase the amount of prior information in the model, by fixing σ_T and $\bar{\mathbf{Z}}$ to predetermined values[5]. Typically we have to choose σ_T very small so that the latent sequences closely follow the template. In this case the optimisation then still has to determine an arrangement of the sequences in latent space which optimally predicts the observed data. Because the new optimisation problem is much more constrained than the original GPLVM problem, it usually converges to good solutions even when the latent sequences $\mathbf{Z}^D$ are initialised at random.

4.4 Results

We first evaluate the template prior on the DLR arm position data set which allows us to make quantitative judgements about the quality of DMP modulation in the found latent spaces. In Section 4.4.2 we then demonstrate the proposed method on a real-world motion capture data set.

4.4.1 DLR

These experiments are an extension of the experiments presented in Section 2.3.1, please see there for details of the setup. In short, we use the position data set with

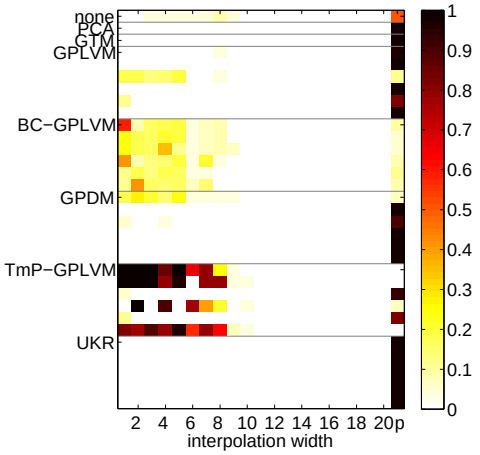
[5]Note that a similar effect can be achieved by choosing narrow prior distributions for these variables.

noise which consists of straight upwards movements in a plane of the DLR arm end-effector. The redundancy is resolved in a consistent way. Therefore, the position of the end-effector in the 2D-plane uniquely determines the 7 joint angles of the robot and justifies use of 2D latent spaces. We evaluate generated movements according to whether they follow the expected trajectory in the task space (plane) and whether they resolve redundancy in the defined way. If they do within a strict error margin, we say that they are successfully generated. In contrast to Section 2.3.1 interpolations are not done pose-for-pose, but by interpolating $\mathbf{z}_0$ and $\mathbf{g}$ of a learnt DMP[6] between the demonstrated trajectories. We gradually increase the distance between two demonstrated movements. This is implemented by skipping an increasing number of movements during selection of the data set. We call the number of skipped movements the interpolation width, as before.

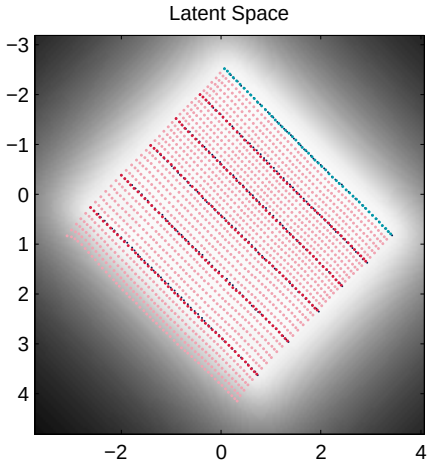
Fig. 4.7(a) summarises the accumulated results of our experiments. We tested learning and interpolation of DMPs in joint space (1st row) and in latent spaces resulting from different dimensionality reduction techniques (subsequent rows). Except for PCA and the GTM all tested techniques depend on the initialisation of latent points before optimisation. We used 6 initialisations (in that order): parallel lines, random, PCA, Isomap, LLE and Laplacian Eigenmaps. Our results show that no previously proposed DR technique produces latent spaces in which interpolation of DMP parameters consistently generates trajectories which correspond to the desired trajectories in the underlying task space. Our GPLVM with template prior ($\bar{\mathbf{Z}}$ fixed to a diagonal line, $\sigma_T^2 = 0.001$), on the other hand, reaches a high percentage of successful interpolations up to an interpolation width of 5 and partially beyond. We stress that this is a measure for how well the found latent spaces generalise to new movements of the same kind which we can only evaluate in this controlled robotic setup.

Fig. 4.4(c,d) illustrates the problems involved in this data set. Depicted is the inverse kinematics function for the 7 DLR joints that would need to be learnt, if the task space coordinates were known. Fig. 4.4(c) shows the result of learning a DMP and varying its parameters in joint space. Due to the nonlinearities involved newly generated movements do not follow the task constraints and exhibit different dynamics. If the DMP is learnt and modulated in a TmP-GPLVM latent space as in Figs. 4.4(b,d) and 4.7(b,c) (interpolation width 5), however, generated movements fit well to the desired.

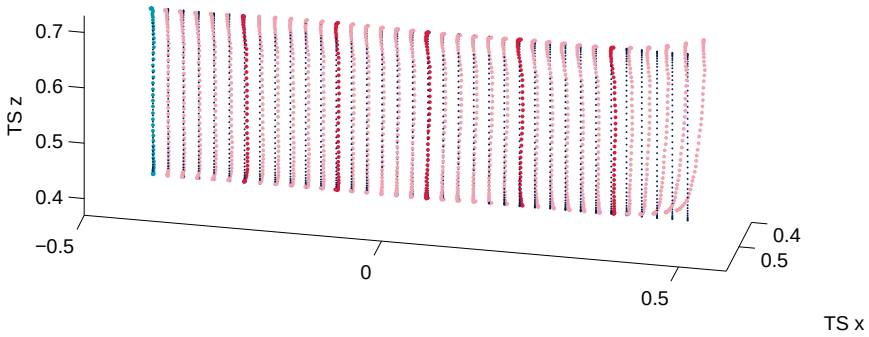
[6]The DMP is always learnt on the 1st demonstrated trajectory in these experiments. Spline interpolation of start state and goal on first and last data points of sequences in data set.



(a) quantitative summary



(b) TmP-GPLVM LS for interpolation width 5



(c) TmP-GPLVM TS for interpolation width 5

Figure 4.7: (a) Ratio of successfully interpolated trajectories using DMPs. 1 means all trajectories successfully interpolated. First line: DMPs operated in joint space, no dimensionality reduction. For GPLVM variants and UKR 6 different initialisations of latent points are tested (shown in this order): ad-hoc parallel lines, random, PCA, Isomap, LLE, LE. Last column: p-value for hypothesis that 'The mean ratio of successful interpolations is smaller or equal to the corresponding mean of the joint space interpolation'. (b) Latent space of GPLVM with suggested template prior for interpolation width 5. (c) Corresponding task space trajectories.

4.4.2 Human Motion Capture Data

We use the punch data set of Section 2.1 to evaluate our method on a real-world motion capture data set which fits the assumptions of the method. In particular, the sequences in the data are movements of one kind (punch) which predominantly vary with respect to a prominent feature of that movement (punch height). However, the punches have not been recorded with this application in mind and exhibit more variation apart from the punch heights. We apply dimensionality reduction on this data set (99 57-dimensional data points, centred and scaled to standard deviation of 1 in each dimension).

Again we want to interpolate between demonstrated movements. In contrast to the DLR data, however, we do not have ground truth in this case. We still can evaluate: a) The accuracy of the DR generative mapping, that is the error between a data point in joint space and its reconstruction from the corresponding latent point by means of the generative mapping (red arrow in Fig. 4.1). We call this the *reconstruction error* e_r. b) The discrepancy between a data sequence and the corresponding sequence generated from a modulated DMP, that is, the DMP is learnt on a different sequence, but its parameters $\mathbf{z}_0$ and $\mathbf{g}$ are set to the values of the target sequence. An error occurs, when the dynamics (shape) of the learnt sequence differs from the dynamics of the target sequence. We call this error the *modulation error* and it can be evaluated in latent space (e_m^{LS}) or joint space (e_m^{JS}) after making use of the DR generative mapping. e_m^{JS} is a combined measure of e_r and e_m^{LS}. Both, the reconstruction and modulation errors, only measure the fit of the model to the training data, but they give us a lower bound on the interpolation performance that we expect. In other words, if the modulation error is too high, interpolation will not give desired results either. All errors are given as normalised sum of squared errors (nSSE)[7].

4.4.2.1 Existing dimensionality reduction techniques

First we evaluate the use of DMPs in joint space. The modulation error was $e_m^{\mathrm{JS}} = 1.498$. This is unacceptably high and, for example, means that some joints rotate for several periods. The resulting movement is completely unnatural.

Because of its ease of use, PCA is a popular tool for pre-processing of motion capture data, but its representational power on this kind of data is actually weak. The data reconstruction error for a 2D latent space was very large at $e_r = 0.609$. By increasing

[7] Definition: $\mathrm{nSSE}(\mathbf{X}, \hat{\mathbf{X}}) = \sum_{ij}(\hat{x}_{ij} - x_{ij})^2 / \|\mathbf{X}_c\|_F^2$ where $\mathbf{X}_c$ is $\mathbf{X}$ with centred columns

init	GPLVM	BC-GPLVM	GPDM	TmP-GPLVM
random	≥ 1.000	≥ 1.000	≥ 1.000	**0.004**
lines	0.381	≥ 1.000	0.519	**0.004**
pPCA	≥ 1.000	0.659	≥ 1.000	**0.004**
GPLVM-MDS	0.980	≥ 1.000	0.675	**0.004**
Isomap	0.719	≥ 1.000	≥ 1.000	**0.004**
LLE	≥ 1.000	0.608	≥ 1.000	**0.007**

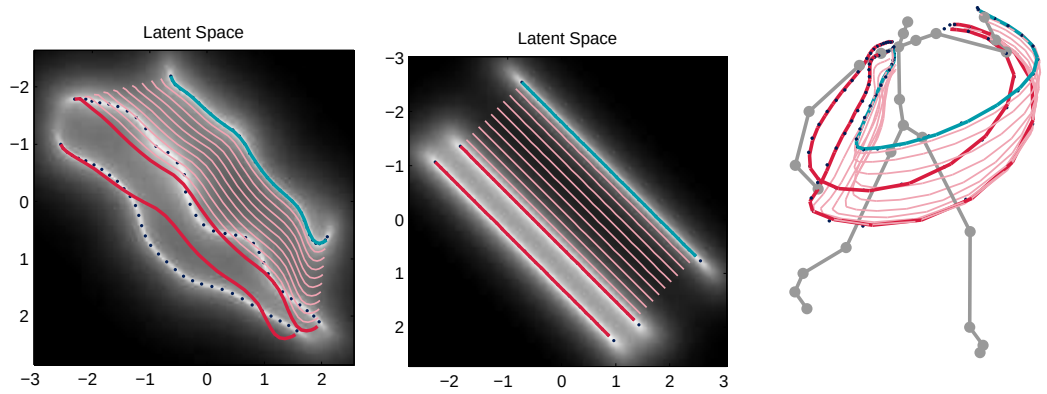
Table 4.1: Summary of results (modulation error in joint space: e_m^{JS}=nSSE of trajectories generated from DMPs modulated in latent space). For comparison, without DR: 1.498, with PCA: 2.464

the number of latent dimensions the PCA representation became more accurate and e_r decreased. For a 10D latent space e_r dropped to an acceptable 0.022 while explaining 97.9% of the data variance. The modulation error, however, stayed high for all latent dimensions (for 10D $e_m^{\text{JS}} = 1.023$) meaning that PCA is no suitable DR method for our application.

PCA is the standard method for initialisation of the GPLVM. Additionally we used random, lines, GPLVM-MDS, Isomap (k=6) and LLE (k=7) as initialisations on this data set. Apart from the GPLVM we also tested other variants of it as before. We defined the GPDM dynamics on the differences of the latent points and used a SE+linear kernel with inverse width 0.2, variances 0.01 and white noise 1e-6. We also tested a back constrained GPLVM for which we used kernel based regression (kbr) back constraints with rbf kernels with width 0.2. For all GPLVM variants we used a strong prior on the lengthscale of the SE covariance function of the latent space to data mapping to avoid overfitting[8]. We did not use sparse approximations and set the maximum number of gradient descent steps to 500 which is usually reached. We compare modulation errors e_m^{JS} for the different initialisations and variants of the GPLVM in Table 4.1.

Although the data reconstruction error for GPLVM based approaches was usually very small (in the order of 0.001 or smaller), Table 4.1 shows that neither a standard GPLVM, a GPLVM with back constraints, or a GPDM could produce latent spaces sufficiently regular to allow for the representation of all punches with a single DMP. As an example, we plot the best latent space (GPLVM with lines initialisation) together with the learnt and modulated DMPs in Fig. 4.8(a).

[8]Gaussian prior with $\sigma^2 = 0.001$ defined over $1/\ell^2$



(a) GPLVM, lines init (b) TmP-GPLVM, random init (c) Interpolation

Figure 4.8: (a) Latent space of GPLVM with lines initialisation after optimisation. (b) Latent space of GPLVM with template prior and random initialisation after optimisation. (c) Trajectories of punch hand for movements generated from the DMP in (b). Dark blue dots: data points, light blue line: learnt DMP, red lines: DMP modulated to other trajectories in data set, pink lines: interpolated DMP trajectories.

4.4.2.2 Template Prior

Table 4.1 also shows results for the GPLVM augmented with the template prior. It clearly outperformed all other methods in terms of joint space modulation error. Good results were largely independent of initialisation used. The movements generated from the modulated DMPs were almost identical to the original movements. For these results we fixed the template $\bar{\mathbf{Z}}$ to a diagonal line and set $\sigma_T^2 = 1e-7$. The data reconstruction error was low at 0.001.

A low modulation error e_m^{JS} only guarantees that we can use modulation of a DMP to accurately reconstruct all demonstrated movements, but it does not guarantee that a DMP with interpolated parameters also produces a movement that we would regard as interpolated with respect to the desired task. In particular, in our punch example, a movement interpolated between a high and low punch should result in a punch with intermediate height. Fig. 4.8(c) illustrates the result of such an interpolation experiment. We linearly interpolated $\mathbf{z}_0$ and $\mathbf{g}$ between the high and very low punches and generated new trajectories in the latent space shown in Fig. 4.8(b). The figure depicts the punch hand trajectories of 9 interpolated punches together with the original data points and the learnt and modulated DMP trajectories. The DMP was learnt on the high punch. We note that the order of the punch sequences in latent space did not correspond to the

punch heights. In particular, the very low punch was closer to the high punch than the low punch. Also none of the interpolations directly fit to the low punch. We believe that this is a consequence of the data resulting from differences in style of execution of the punches. Nevertheless, we were able to generate smooth interpolations between the punches with the DMP from latent space.

4.5 Discussion

In this chapter we have pointed out that the parameters of discrete DMPs which are learnt in joint space, as originally suggested, do not have a correspondence with underlying task variables. Indeed, we have shown that a change of these parameters produced movements which neither followed the constraints set by the task, nor the consistent resolution of redundant degrees of freedom. Following previous work we have suggested to use dimensionality reduction to provide a representation of demonstrated movements which overcomes these problems by making parameters of DMPs interpretable with respect to task variables while at the same time capturing regular structure in the data. Experiments, however, have shown that application of standard dimensionality reduction methods was not sufficient to generate meaningful new movements by only an adaptation of DMP parameters. Hence we have suggested to modify the GPLVM by incorporating a template prior which strongly favours latent spaces reflecting the temporal and spatial structure of point-to-point movements (TmP-GPLVM).

In a control experiment on data from a 7-DoF robot arm we have demonstrated that only DMPs modulated in a TmP-GPLVM latent space generated movements which accurately followed the expected trajectories as defined by task and null space constraints. As the results have been obtained for interpolated DMP parameters, we can conclude that the proposed method also generalises to unseen movements which were not present in the data.

For human motion capture data we have observed very similar results: a change of DMP parameters lead to unrealistic movements when the DMP was learnt directly in joint space or a latent space resulting from standard dimensionality reduction methods. However, when the DMP was applied in a TmP-GPLVM latent space, the original movements could be reconstructed just by changing DMP parameters. As we were missing ground truth in this data set, it was not possible to quantitatively evaluate movements generated through interpolation of DMP parameters. Instead we have

presented interpolated trajectories which at least qualitatively corresponded to our expectations.

For both our examples it can be argued that the highly structured latent spaces of the TmP-GPLVM did not correspond to the true, underlying latent spaces. We accept the different representation in favour of the improved generalisation capabilities of the DMP. The imposition of such structure did not impair the ability to successfully transfer the interpolated DMP trajectories from latent space to the original joint space representation. We believe that this is an example of the representational power of GPs. It may well be that for other, richer data sets the strict constraints on the structure in latent space prevent a faithful representation of the data. In this case the TmP-GPLVM provides principled ways of loosening these constraints (e.g. by increase of σ_T, or allowing to learn the template $\bar{\mathbf{Z}}$), but it should in principle also be possible to allow a larger set of linear transformations of the template by adaptation of $\mathbf{A}$ in eq. (4.6). Another useful extension of this work would allow varying numbers of data points for each sequence by loosening the point-by-point coupling between template and data sequences. Again we note, however, that there is a tradeoff between the ease of DMP modulation and the complexity of the resulting latent space.

In conclusion, we have presented evidence that the TmP-GPLVM provides latent spaces that allow fast, online generation of a continuum of new movements by a simple change of DMP parameters based on a small number of similar demonstrations. It is therefore a useful tool for robot programming by demonstration.

The greatest strength of DMPs, compared to other methods of generating movements from examples, is their properties in an online control setting. Then we benefit from their attractor properties which ensure that the generated trajectory is tracked also under the influence of noise and perturbations. The next chapter presents how we can implement online control from GPLVM latent spaces using the example of reinforcement learning.

Chapter 5

Reinforcement Learning in Latent Spaces

In the last chapter we have shown how latent spaces can be used to generate novel movements with dynamic movement primitives. In this chapter we go one step further and investigate how dimensionality reduction can help reinforcement learning which tries to find movements which optimally achieve an abstract goal without being given explicit feedback on how to improve the current movement. Our work is based on the knowledge that reinforcement learning in high-dimensional, continuous spaces, such as those found in many tasks in robotics, remains a challenging problem. To overcome this challenge, a popular approach has been to use successful demonstrations of the task to find an appropriate initialisation of the policy in an attempt to reduce the number of iterations needed to find a solution. Here, we present an alternative way to incorporate prior knowledge from demonstrations of individual postures into learning, by extracting the inherent structure of the problem in order to find an efficient state representation. In particular, we use the GPLVM to capture latent constraints present in the data. By learning policies in the learnt latent space, we are able to solve the planning problem in a reduced space that automatically satisfies the task constraints. As shown in our experiments, this reduces the exploration needed and greatly accelerates the learning. We demonstrate the feasibility of our approach for learning a complex reaching task on the 7 DOF KUKA light-weight arm and the 19 DOF KHR-1HV humanoid.

5.1 Introduction

The application of reinforcement learning (RL) to continuous, high-dimensional systems such as humanoid robots (Fig. 5.14(a)) remains a challenging problem. While a large variety of RL algorithms have been developed for solving complex planning problems (Sutton and Barto, 1998), typically the scalability of these approaches is limited to applications involving small, discrete worlds. Continuous state spaces necessitate discretisation, or the use of continuous function approximators, but both are affected by the curse of dimensionality, which states that the resources needed to achieve a certain accuracy scale exponentially with the number of dimensions of the state space.

In this context, recent attention in the robotics community has focused on addressing the issue of making RL feasible for high-dimensional continuous problems. Several approaches have been suggested to allow RL to achieve acceptable results with limited resources despite working with large state spaces. In a programming by demonstration framework, demonstrated trajectories can be used to initialise a parametrised policy (Guenter et al., 2007; Peters and Schaal, 2008a). Because such an initial policy is assumed to be close to the optimal policy, only a limited number of local updates of the policy parameters may be needed to find an acceptable solution. Hierarchical RL (Barto and Mahadevan, 2003) is a more general approach in which the RL problem is broken down into a hierarchy of sub-problems. Solutions of sub-problems are combined to solve the high-level problem. This divide and conquer approach is intuitively plausible, but the difficulty is then shifted towards selection and learning of the hierarchy. Despite recent advances (Konidaris and Barto, 2009), problems remain, in particular with large state spaces. Abstractions (Li et al., 2006) have been suggested as a general term describing a mapping of state space to a more compact, abstract space which benefits learning. The simplest abstraction, for example, just selects a subset of the state dimensions, but any transformation of the state space is possible. If an insight into the control problem exists a priori, an abstraction can be chosen by hand (Morimoto and Atkeson, 2009), but ideally we would like to learn suitable abstractions from experience.

In this chapter we investigate the suitability of dimensionality reduction (DR) as a method for automatically determining an abstraction for RL from demonstrated postures. In other words, we answer: “Does RL benefit from state representations found by DR from demonstrated examples?” and give conditions under which our answer is

valid. The idea of using DR to aid RL has recently been explored by Morimoto et al. (2008). They use Kernel dimensionality reduction to find a low dimensional state representation which maintains the relation to the reward function as much as possible. Unfortunately, their state representation only uses a linear mapping from the original state space which, in many problems, is not sufficient to represent the state space faithfully (see Sec. 5.5). Gaussian Processes (GPs) (Rasmussen and Williams, 2006), on the other hand, are powerful nonlinear function approximators and have been used within RL to approximate the problem dynamics or value functions (Rasmussen and Kuss, 2004; Morimoto and Atkeson, 2009). Our contribution shows that the GPLVM, as a DR method based on GPs, can produce faithful state representations that simplify the learning problem and make RL feasible for robots with redundant degrees of freedom even when no initialisation of the policy is available.

5.2 Learning the State Abstraction from Demonstrated Examples

The idea of our approach is to use posture data acquired from kinaesthetic demonstrations to extract a non-linear manifold, which can then be used as state representation for RL.

Using kinaesthetic demonstrations, in which the robot's movements are manually guided, has several benefits, for example, (i) it ensures that all demonstrated postures are feasible for the robot, (ii) the demonstrator can directly see that task constraints are satisfied within the demonstrations and (iii) it avoids correspondence issues that may arise due to differences in embodiment between the demonstrator and imitator (since the demonstrations are already performed on the robotic plant). However, it also becomes increasingly difficult to demonstrate successful movements (i.e., continuous trajectories) with increasing number of degrees of freedom of the robot plant. We, therefore, do not rely on demonstrations being continuous movement trajectories and rather allow *discrete demonstrations*, where desired postures are demonstrated individually (similar to 'keyframing' in animation). Recording demonstrations then consists of moving all robot joints in a desired posture, recording the joint angles and repeating the procedure for a different desired posture until the space of postures needed to fulfil the given task has been sampled to the satisfaction of the demonstrator. In this way even a single person can demonstrate a set of postures of a high-DOF humanoid which

allow generation of full-body movements.

In this context DR corresponds to a nonlinear interpolation procedure which allows us to generate continuous movements from a discrete set of samples. At the same time it provides us with a state representation which makes RL feasible even when the dimensionality of the original state space is very high. If demonstrations of continuous movements are available instead of discrete samples, DR can still be applied to find a compact state representation beneficial to RL. In this case the sample density will just be higher and potentially a method should be used which takes the sequential structure of the demonstrations into account (see e.g. Chapter 4). A schematic of our approach of learning state abstractions is shown in Fig. 5.1.

We assume that the demonstrated postures fulfil constraints, when they are necessary for the achievement of the given task (see example below). By introducing this invariance into the demonstrations, this latent structure (i.e., the constraint manifold) can be incorporated into the state space model learnt by the DR[1]. This in turn benefits RL by restricting exploration to parts of the space in which (in the eyes of the demonstrator) a feasible solution to the task exists.

Our notion of constraint here is not as strict as the word might suggest. In general it stands for a desired property of postures or movements which we would like to be true at all times. However, we have to assume that demonstrations are noisy and constraints are even in the training data only approximately fulfilled. Therefore, we define an error margin within which we regard constraints to be true for each example problem below in which we know the desired, underlying constraints. Furthermore, we note that we do not extract an explicit representation of the constraints, rather, they are implicitly represented in the latent space resulting from DR on the demonstrations. Consequently, it is not possible to distinguish regions not included in the demonstrations with respect to whether they fulfil the constraint, or not. While this would be useful to know, extrapolation from the demonstrations is outwith the scope of this work.

5.2.1 Example: Constrained Bimanual Manipulation

As a simple example, consider a bimanual manipulation task in which we want to move an object with two hands from one place to another (see Fig. 5.2). If the full

[1]This has interesting parallels with the idea of looking for *generalised coordinates* in analytical dynamics (e.g., see Udwadia and Kalaba (1996)) where, under a holonomic constraint (i.e., an *equality constraint*), it is possible to find a coordinate system in which the constraint is automatically satisfied. This greatly simplifies the problem of solving the equations of motion of the system.

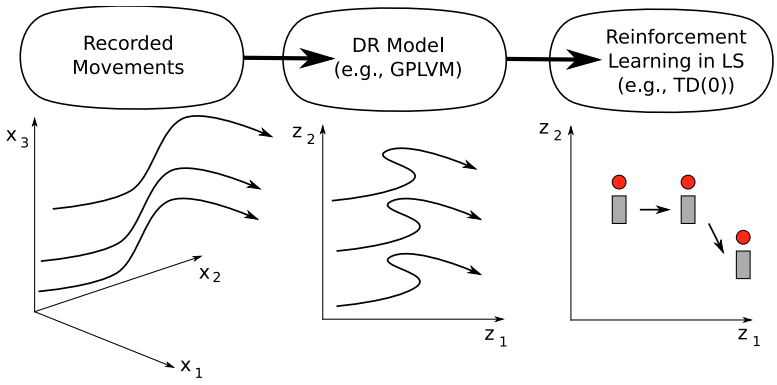
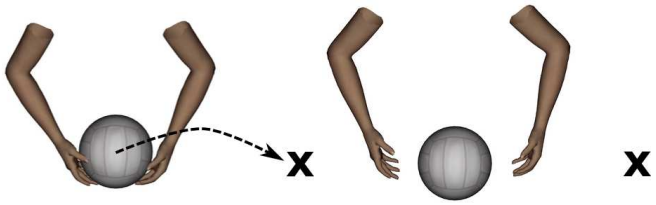


Figure 5.1: Schematic of our approach. Given a set of demonstrated movements, dimensionality reduction is used to find a low-dimensional manifold on which the demonstrations lie (the latent space). The space defined by this manifold can then be used as the state-space representation within the reinforcement learning.



(a) (b)

Figure 5.2: (a) In order to move the ball to the target ($\mathbf{x}$), the movement must be constrained so that the hands remain a fixed distance apart. (b) If the constraint is broken, the ball is dropped.

state of the two arms is defined by the positions of the shoulder and elbow joints, then total dimensionality of the system is four. However, for the movement to succeed, the condition that the two hands must remain a fixed distance apart must be fulfilled throughout the movement (see Fig. 5.2(a)), otherwise the object will be dropped. In effect, this constrains the possible movements that can be used to solve the task: 1 degree of freedom is eliminated by constraining the distance between the two hands, and a second is eliminated if we do not allow rotations of the object. In other words, for this problem, any successful movement (fulfilling these task constraints) must lie on a 2-D manifold embedded in the full 4-D state space.

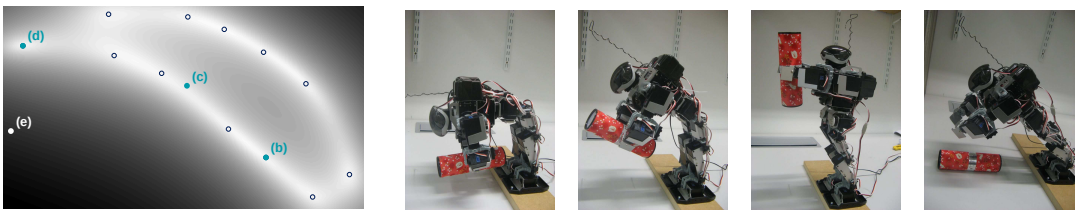
To exploit the existence of this manifold by restricting exploration in RL to the space in which the task constraints are satisfied we have to find an appropriate representation of that space for use in learning. In some cases, this could be derived from

an expert analysis of the task (here, involving derivation of kinematics of the plant and the constraints) resulting in an analytical model of the manifold. However, while this is a valid approach, it greatly increases the complexity of finding the appropriate representation for the non-expert user. Instead, in our framework we propose to *learn the manifold by demonstration*. That is, we rely on the demonstrator to provide appropriate example postures that (i) satisfy the task constraints (here, postures in which the hands are in an appropriate position to grasp the object), (ii) have sufficient coverage to make a reasonable approximation of the underlying manifold, and (iii) define a space in which a feasible solution to the task (here, a path to the target) exists. In other words, by taking appropriate care in selecting example postures, a non-expert demonstrator can use our framework to automatically learn a state representation that captures structural elements of the task (in this example, as an implicit model of the constraints) without the need to define them formally by hand.

While this is a simple example, similar arguments also apply to more complicated situations, in particular for natural movements with many degrees of freedom such as that of humans (Grochow et al., 2004) or humanoid robots (Chalodhorn et al., 2009) subject to more complex environmental or task constraints (Howard et al., 2009). For systems such as these, using DR is even more appealing since formal definition of the underlying task structure is increasingly difficult as the system dimensionality increases.

5.2.2 Finding Suitable Latent Spaces with Dimensionality Reduction

As in all other dimensionality reduction problems, PCA is our first choice of method, because of its simplicity and yet often useful results. Again, our experiments show, however, (see Section 5.5) that the error of pure PCA reconstructions is in many cases too large for our purposes, when a small number of latent dimensions is used. On the other hand, the GPLVM can represent motion data accurately also in very few dimensions. Therefore, the potential range of reinforcement learning problems that we can apply the GPLVM to is considerably greater than that for PCA. The computational cost of finding a latent space is heavily increased through the GPLVM optimisation, but in our approach we can run the optimisation offline before reinforcement learning and it is therefore not time critical. For an application of the GPLVM to RL it is more important that the generative mapping defined by the GPLVM is cheap. This is



(a) LS (b) (c) (d) (e)

Figure 5.3: GPLVM learnt from 12 demonstrated poses of KHR-1HV humanoid on a lifting task. (a) shows the resulting latent space. (b-e) show the robot in poses generated from the depicted points in latent space. In (e), the robot drops the object, because hands move apart.

standard GP prediction and is done efficiently in $O(N)$ for each test point, because the covariance matrix $\mathbf{K}^{-1}$ is fixed after learning and can be pre-computed (see Section 2.2.2.1, eq. 2.2). The returned variance is equal in all data dimensions and gives a measure for the confidence in the mean prediction.

Although it is possible that the mean prediction fulfils desired constraints in regions of high predictive variance, usually this is a good indicator for poor generalisation away from the demonstrated postures (see, e.g., Section 5.5.3). In absence of other information, this relationship can be exploited in the RL to prevent the expensive evaluation of states for which the predictive variance indicates that the generated posture is a poor generalisation anyway (see Section 5.5.4). We illustrate this point on a GPLVM latent space learnt from kinaesthetic demonstrations of the KHR-1HV in Fig. 5.3. The robot has to lift an object which can be done successfully while moving through the poses within the region of low predictive variance (Fig. 5.3 (b-d)), but fails when regions with high variance are visited (Fig. 5.3 (e)), because the robot leans too much forward and hands of the robot move apart.

In our experiments below we used the standard squared exponential (SE) covariance function with added white noise (eq. (3.19)) and initialised the latent points $\mathbf{Z}$ with points found by PCA. The RL relies on two major properties of the resultant latent space:

1. successful, continuous trajectories in state space must be represented as continuous trajectories in latent space.

2. Points generated from latent space must obey the given constraints.

These properties assure that data points lie in connected regions in latent space between

which the RL can smoothly transition without breaking the constraints. Our key insight in applying the GPLVM to this problem is that without any further information 1) the PCA initialisation of the GPLVM together with the SE covariance function provide the desired smoothness of the latent space while 2) the learnt GP mappings provide the necessary accuracy. Previously introduced variants of the GPLVM, (e.g. Lawrence and Quinonero-Candela, 2006; Wang et al., 2008; Urtasun et al., 2008) and ch. 4, use additional prior information about the data to further restrict and therefore simplify the DR problem. If the assumptions they make about the data apply to the problem at hand, it is worth investigating their use for RL, too. In our experience, however, these variants tradeoff improved smoothness (1) against reduced accuracy (2).

In order to incorporate our DR model into RL, the DR technique must provide both a generative mapping and its inverse (i.e., generative mapping from latent space to joint space, and the inverse mapping back to latent space). In the next section we evaluate alternative ways of equiping the GPLVM with this inverse mapping.

5.3 Out-of-Sample Mappings for the GPLVM

In actual control experiments, including reinforcement learning, we have to incorporate the feedback from the high-dimensional state of the real plant (data space) into the state in latent space. However, the GPLVM only learns the generative mapping from latent to data space and does not provide a mapping from data to latent space. Many DR methods have the same problem and various out-of-sample extensions have been suggested which work within the framework of the given method (e.g. de Silva and Tenenbaum, 2003; Bengio et al., 2004; Li et al., 2005).

In particular for control applications, we have to take care that the generative and its inverse mapping correspond well to each other to prevent unexpected effects and deadlocks. For example, undesired loops can occur when a policy in latent space results in a step which is reversed by errors in the inverse mapping which in turn would lead to a repitition of the step by the policy and so on.

For the GPLVM we consider 4 main approaches for realising an out-of-sample extension:

1. [ll] Using a locally, linear approximation of the mapping based on the nearest neighbours in data space.

2. [gp] Learning the mapping as nonlinear regression from observed data to GPLVM latent points.

3. [bc] Approximating the mapping with the learnt back-constraints of a BC-GPLVM.

4. [ml] Maximising the probability of the test point under the learnt model with respect to its latent representation.

1. and 2. are generic approaches which can be used with any DR technique once the latent points have been found, 3. is particular to the BC-GPLVM and 4. is applicable to all generative methods which define a probabilistic model of the observed data.

In the following we present details of these approaches and evaluate the quality of the resulting inverse mappings according to the error on random samples in latent space. The evaluation tests how closely the approximation of the inverse mapping with the chosen approach comes to the real inverse of the generative DR mapping which can not be derived analytically from the GPLVM. In general, however, there is no unique inverse mapping between observed and latent space and we discuss this issue in Section 5.3.3.

5.3.1 Out-of-Sample Mapping Methods

5.3.1.1 Locally Linear Approximation

This approach is based on LLE (Roweis and Saul, 2000). For a given test point $\mathbf{y}$ in data space we find its k nearest neighbours $\tilde{\mathbf{Y}} = [\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_k]$ from the set of training points and compute weights of their linear combination $\hat{\mathbf{y}} = \tilde{\mathbf{Y}}\mathbf{w}$ which minimise the squared error between the test point and its reconstruction $\|\hat{\mathbf{y}} - \mathbf{y}\|^2$ [2]. In latent space we then use the corresponding set of nearest neighbour points $\tilde{\mathbf{Z}}$ to compute the latent representation of the test point with the same linear combination: $\mathbf{z} = \tilde{\mathbf{Z}}\mathbf{w}$. This method depends on the parameter k which determines how well $\mathbf{y}$ can be approximated by the linear combination of neighbouring vectors. To represent $\mathbf{y}$ perfectly we require D linearly independent basis vectors. This can theoretically be achieved with $k = D+1$. Because in a nonlinear setting this linear approximation can at most be locally valid, there is a tradeoff between high reconstruction accuracy and generalisation which is controlled by k [3].

[2] We implemented this using right matrix division in Matlab

[3] In the following experiments we used $k = 8$.

5.3.1.2 Nonlinear Regression

The second approach is a straightforward application of nonlinear regression techniques. Given the data $\mathbf{Y}$ and its latent representation as found by DR $\mathbf{Z}$ we just learn a regressor $f(\mathbf{y}) = \mathbf{z}$. Any regression model can be used. Because of its power and simplicity we learn another set of GPs (number of GPs equal to dimensionality of latent space). In line with the GPLVM these GPs share the same covariance function. While this is an arbitrary, simplifying choice our experiments indicate that it has no large effect on the quality of the learnt mapping.

5.3.1.3 Back-constraints

In the back-constraint GPLVM the function $f(\mathbf{y}) = \mathbf{z}$ is already part of the model. In fact, the latent points are actually optimised to be consistent with the conjectured function. We therefore expect that the BC-GPLVM particularly favours accurate inverse DR mappings.

5.3.1.4 Maximum Likelihood

Analogous to learning the GPLVM we can use maximum likelihood optimisation of its predictive distribution to find a latent space representation $\mathbf{z}^*$ of a test point $\mathbf{y}^*$ which maximises the probability of observing the test point given $\mathbf{z}^*$ and the learnt GPLVM. In particular, we have the following likelihood defined by the predictive distribution of the GPLVM (cf. eq. (2.2))

$$P(\mathbf{y}^*|\mathbf{z}^*,\mathbf{Z},\mathbf{Y},\gamma) = \frac{1}{Z}\exp\left(-\frac{1}{2}(\mathbf{y}^*-\mu(\mathbf{z}^*))^T\Sigma^{-1}(\mathbf{z}^*)(\mathbf{y}^*-\mu(\mathbf{z}^*))\right) \tag{5.1}$$

where $\mu(\mathbf{z}^*)$ and $\Sigma(\mathbf{z}^*)$ are the predictive mean and covariance of a GPLVM learnt on data set $\mathbf{Z},\mathbf{Y}$ with parameters γ. The formulas for the predictive mean and covariance are as follows.

$$\mu(\mathbf{z}^*) = (\mathbf{k}^{*T}(\mathbf{z}^*)\mathbf{K}^{-1}\mathbf{Y}\mathbf{S})^T \tag{5.2}$$

$\mathbf{k}^{*T}(\mathbf{z}^*)$ is a kernel vector containing values of the covariance function evaluated between $\mathbf{Z}$ and $\mathbf{z}^*$ with dimensions $N \times 1$. $\mathbf{K}^{-1}$ is the inverted kernel matrix of the learnt GP with dimensions $N \times N$. $\mathbf{S}$ is a diagonal matrix containing optional scaling for each of the output dimensions, dimensions $D \times D$. In our experiments we usually choose $\mathbf{S} = \mathbf{I}$.

$$\Sigma(\mathbf{z}^*) = \left(k^*(\mathbf{z}^*) - \mathbf{k}^{*T}(\mathbf{z}^*)\mathbf{K}^{-1}\mathbf{k}^*(\mathbf{z}^*)\right)\mathbf{S}\mathbf{S} \tag{5.3}$$

Here $k^*(\mathbf{z}^*)$ is the covariance function evaluated between $\mathbf{z}^*$ and $\mathbf{z}^*$ only. Note that Σ is isotropic and instead of using its full representation we can replace it with $\sigma^2(\mathbf{z}^*) = \Sigma_{ii}$ below, in particular when $\mathbf{S} = \mathbf{I}$.

We want to maximise the probability of the test point $\mathbf{y}^*$ by adapting the latent position $\mathbf{z}^*$. We do this by minimising the negative log-likelihood

$$L(\mathbf{z}^*) \propto \frac{1}{2\sigma^2(\mathbf{z}^*)}(\mathbf{y}^* - \mu(\mathbf{z}^*))^T(\mathbf{y}^* - \mu(\mathbf{z}^*)) \tag{5.4}$$

We derive the partial gradients of this function with respect to the latent point $\mathbf{z}^*$: $\partial L(\mathbf{z}^*)/\partial \mathbf{z}^*$ in appendix B.1. These can be used in a gradient descent algorithm to find a desired solution for $\mathbf{z}^*$. From the structure of the formula we see that intuitively we are trying to set $\mathbf{y}^* = \mu(\mathbf{z}^*)$ taking the learnt model into account. We therefore initialise $\mathbf{z}^*$ with the latent representation of the nearest training data point $\tilde{\mathbf{z}}$.

5.3.1.4.1 Peculiarities of Predictive Variance The likelihood of a test point $\mathbf{y}^*$ can be increased by moving the mean prediction closer to $\mathbf{y}^*$, or by increasing the predictive variance which is independent of $\mathbf{y}^*$. So, it happens that at the initialisation $\tilde{\mathbf{z}}$ the contribution to the gradient of the predictive variance outweighs the contribution of the mean prediction and essentially drives $\mathbf{z}^*$ into regions of high predictive variance, away from the data and ignoring the test point $\mathbf{y}^*$. This was reflected in first experiments with this method in which optimised latent representations of some test points fitted very closely to the expected positions in latent space while others lay very far from the original data. Clearly, this is not a desired result and questions the utility of the predictive variance when projecting $\mathbf{y}^*$ into latent space. A simple and effective solution to this problem is to exclude the predictive variance from the computation of the gradient. We then only need the gradient in eq. (B.3) and save computations needed for computing the gradient related to the predictive variance.

5.3.2 Evaluation of Out-of-Sample Mappings

We start with the simplifying assumption that the inverse of the generative GPLVM mapping exists and is well defined. In our first experiment we evaluated how well the chosen out-of-sample mapping approximates this inverse. The precise experimental procedure was as such: points were randomly sampled in latent space, they were mapped to observed space using the generative mapping, these points were mapped back into latent space using the chosen inverse mapping and the error between the original samples and their mapped version was computed (see Fig. 5.4).

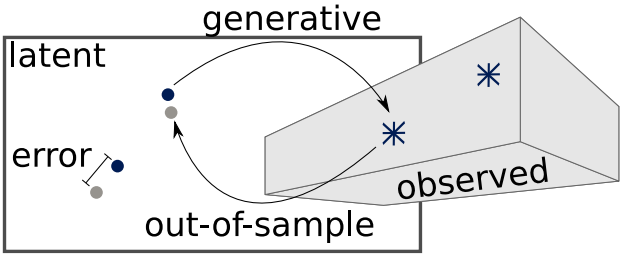


Figure 5.4: Evaluation procedure for out-of-sample mappings: generate random latent points, map to observed space with generative DR mapping, map to latent space with out-of-sample mapping, compute error.

(a) GPLVM

	punch long	punch short	planar arms	DLR arm up
ML	≥ 10.000	**0.495**	**0.000**	**0.000**
Lin	≥ 10.000	≥ 10.000	0.070	≥ 10.000
GP	**0.871**	0.735	0.002	**0.000**

(b) BC-GPLVM

	punch long	punch short	planar arms	DLR arm up
ML	≥ 10.000	3.434	**0.000**	0.038
Lin	≥ 10.000	**0.084**	0.024	≥ 10.000
GP	**0.898**	0.115	**0.000**	**0.003**
BC	1.020	0.096	**0.000**	0.005

Table 5.1: nMSE for out-of-sample mappings on four different data sets (mocap punches long and shortened, two 2DOF planar arms and DLR arm up movements). (a) DR with GPLVM and PCA initialisation. (b) DR with BC-GPLVM and PCA initialisation.

(a) GPLVM

	punch long	punch short	planar arms	DLR arm up
ML	**0.584**	**0.288**	**0.000**	**0.000**
Lin	0.894	0.828	0.370	0.978
GP	0.982	0.938	0.168	0.002

(b) BC-GPLVM

	punch long	punch short	planar arms	DLR arm up
ML	**0.644**	**0.150**	**0.000**	**0.012**
Lin	0.904	0.682	0.274	0.912
GP	0.982	0.644	0.006	0.154
BC	1.000	0.644	0.002	0.202

Table 5.2: Fraction of test points for which nMSE exceeded 0.001 in results of Table 5.1. (a) DR with GPLVM and PCA initialisation. (b) DR with BC-GPLVM and PCA initialisation.

Table 5.1 presents the normalised mean squared error computed over 500 data points for four different data sets. In all cases we ran experiments for the standard GPLVM and the BC-GPLVM with PCA initialisation[4]. The data sets were the original and cut mocap punch data sets from Section 2.1, a data set from 2 planar 2-link arms as explained below and the DLR arm up position set introduced in Section 2.3.1. These results suggest that the performance of the different methods is very variable. We see that for the BC-GPLVM smaller errors can be expected from any chosen out-of-sample method. [ml] is good on the robot data, but very bad on the punches. [gp] has comparably low errors on all data sets. The absolute size of the errors, however, is discouraging and does not allow to make conclusive judgements.

We present an alternative summary statistic in Table 5.2. Instead of looking at the errors directly we evaluated for how many test points the inverse mapping produced errors exceeding a sufficiently small limit (nMSE > 0.001). The table depicts the fraction of test points for which the nMSE exceeds the limit. Therefore, the lower the number in the table, the more points have a negligibly small error meaning that the out-of-sample mapping is more precise. This statistic reveals that the pure nMSE in Table 5.1 was distorted by averaging effects. Actually, [ml] provided the most accurate inverse mapping for all data sets. We observed the largest difference to the other meth-

[4]We ran experiments also with the lines initialisation (not presented). Results were very similar.

ods for the cut punches where the fraction of test points with large error was only 0.15 for [ml] in the BC-GPLVM while all other methods had fractions larger than 0.64. This is the opposite to what Table 5.1(b) suggests, indicating that the nMSE for the 15% of badly reconstructed test points had very large errors for [ml] while the error in the other methods was moderately high for all test points. We visualise this point in Fig. 5.5 which shows the latent space learnt with the BC-GPLVM (PCA initialisation) on the cut punches together with the test points and their reconstructions from [ml] and [gp]. While most test points which lay close to the original data could be reconstructed very well with [ml], [gp] also had significant errors in the high confidence regions. On the other hand, in Fig. 5.5(a) 38 of 500 points are not plotted, because they lay outside the limits of the plot (in the order of latent space coordinates (5,-3)), while Fig. 5.5(b) contains all points reconstructed with [gp]. This explains why [ml] had larger nMSE even though more points were better reconstructed.

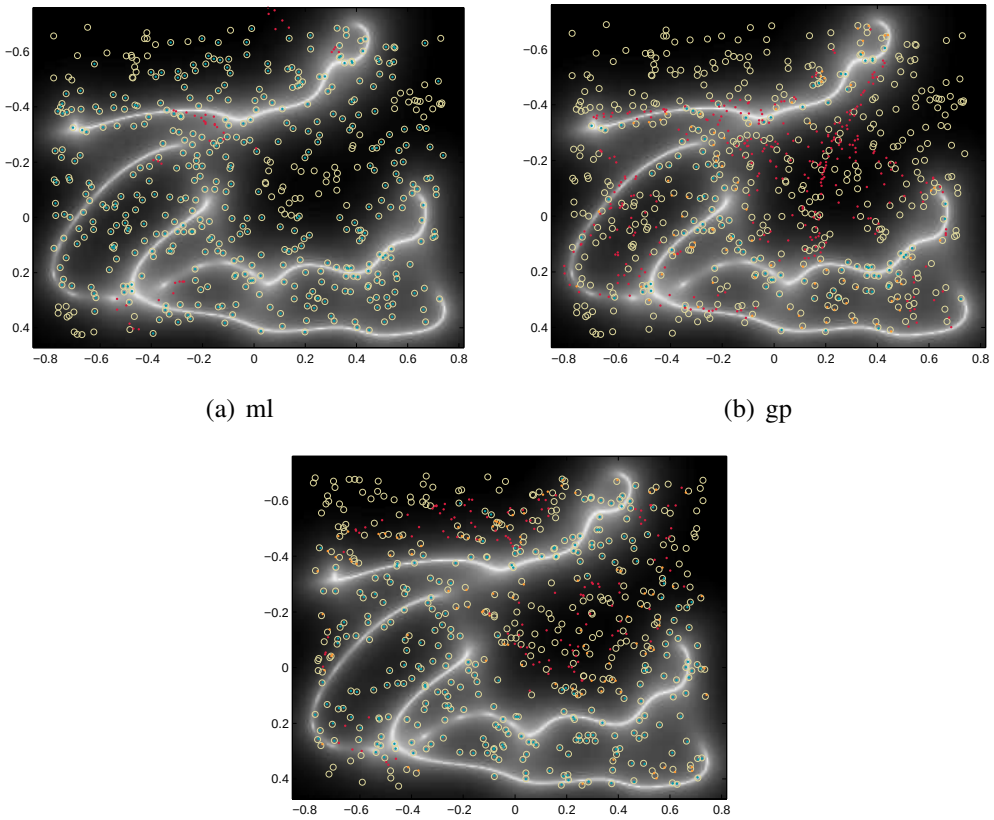
The results in Table 5.2 also suggest that there was an improvement in the accuracy of the inverse mappings, if the BC-GPLVM was used instead of a standard GPLVM, although the effect was small. [bc] and [gp] did not significantly differ in their performance. This was due to the similarity of the methods used: kernel-based regression back-constraints with squared exponential kernels and GPs with squared exponential kernels, respectively. As expected, [ll] as a linear method could not accurately approximate the inverse mapping.

5.3.3 Ill-Defined Inverses

The last section neglected a major issue with the out-of-sample mapping: potentially it is many-to-many, not one-to-one. This results from the properties of the generative mapping:

a) It defines a low-dimensional manifold in high-dimensional observation space of the DR problem, i.e., points off the manifold need to be projected into latent space, making the out-of-sample mapping many-to-one.

b) Potentially many points in latent space map to the same point in observed space. This happens when the manifold intersects itself, or neighbouring parts of latent space are collapsed into one point in observed space, making the out-of-sample mapping one-to-many.



(a) ml (b) gp

(c) ml+gp

Figure 5.5: Example latent space evaluating inverse mappings. Circles: 500 test points, dots: reconstructions using either [ml] (a), [gp] (b), or bootstrapping [gp] with [ml] (c). Colour coding: blue: $\mathrm{nMSE} <= 0.0001$, orange: $0.0001 < \mathrm{nMSE} <= 0.001$, red: $\mathrm{nMSE} > 0.001$. In (a) 38 points lie outside of limits of plot.

We consider b) first. It presents a major problem for all presented out-of-sample approaches, because they do not handle several solutions to one given point. In order to determine the significance of the problem to our application we need to investigate when it occurs. In the absence of data the mean prediction of a GP is the mean function. For the GPLVM this means that the further you move away from the original data in latent space, the closer the mean prediction gets to the mean of the data in observed space. This is independent of the direction in which you move away from the data. Consequently, there are infinitely many positions to which the out-of-sample mapping could map the mean of the data. This is particularly visible for [ml] in Fig. 5.5(a). In regions of low-predictive variance, close to the data, points were perfectly reconstructed with the out-of-sample mapping, but reconstructions of points further away from the data ended up in completely different areas of latent space. We have previously seen that the GPLVM, and nonlinear DR methods in general, do not extrapolate well away from the data. Therefore, this result is expected and does not restrict the application of the GPLVM to control problems beyond the limits set by the GPLVM in the first place.

While here the latent space collapses to a single point away from the data, a one-to-many out-of-sample mapping can also result from an intersecting manifold within the data. This, however, is a degenerate case which needs special attention already during dimensionality reduction. For example, periodic manifolds like spheres or tori fall in that category (see e.g. Urtasun et al., 2008). Again, the problem for the out-of-sample methods is tied to a problem for DR in general. We conclude that one-to-many out-of-sample mappings introduce no additional difficulties into our problem, but tests as the ones described in the previous section should be used to diagnose potential degenerate cases.

The out-of-sample mapping also needs to be many-to-one (a). All data points generated from latent space necessarily lie on the manifold defined by the generative mapping. In a control context, however, noise in the execution of a command can lead us off that manifold. The out-of-sample mapping then needs to project the current position in joint space to the point in latent space which corresponds to the closest point on the manifold. [ml] is exactly such a projection method as it optimises the position of a point in latent space such that its corresponding point in joint space has minimal distance to the test point. Therefore we can regard [ml] as the optimal solution to this problem. [gp], on the other hand, suffers from the exact same decay-to-mean effect as we have discussed for the generative mapping: the further you go away from the data

manifold, the closer your out-of-sample estimate will get to the mean in latent space. This effect is already visible in Fig. 5.5(b) where test points outside the data in the upper, left corner were reconstructed around the mean at (0.04, -0.16).

5.3.4 Conclusion

Our results indicate clearly that [ml] gives us the most accurate out-of-sample mapping and should be the method of choice. It has, however, two main disadvantages:

1. because it is an iterative procedure, it is slow compared to the other methods

2. its performance does not degrade gracefully: when errors occur, they are catastrophic introducing unpredictable jumps in the out-of-sample mapping

Both properties are highly undesirable in an online control setting. [gp] does not have these disadvantages and should be used when it gives acceptable accuracy, as it is the case for the two robotic data sets in our evaluation. In cases in which the accuracy of [gp] is insufficient, as in the motion capture data sets, we may improve its performance by bootstrapping with the [ml] results: when learning the out-of-sample GPs add data points to the training set which are reconstructed well with [ml]. In this way we can extend the range in which data is present for the GPs without getting into regions for which the out-of-sample mapping is ambiguous. Additionally, we can add further data points to which noise has been added in joint space together with their [ml] latent space reconstructions to improve accuracy off the latent space manifold. Note that computational costs of [gp] increase with the number of training data points. Therefore, it suffers from a speed-accuracy tradeoff, but maintains the advantage over [ml] of defining a continuous out-of-sample mapping. Fig. 5.5(c) presents results of the bootstrapping procedure in which we added 100 random and 100 random, noisy data points to the GP training set. The fraction of points with nMSE > 0.0001 decreased from 0.79 to 0.46 and with nMSE > 0.001 from 0.63 to 0.26 for the shown data.

In conclusion, we suggest to use the [gp] out-of-sample mapping for the GPLVM in control applications, because of its continuity also away from the data. We note, however, that the accuracy of the mapping needs to be monitored and potentially improved by adding more data to the training set.

5.4 Reinforcement Learning in Latent Space

As mentioned in the introduction, a large number of RL techniques are available for planning and optimising movements given an appropriate representation of the state. For the experiments in this paper, we restrict ourselves to the popular class of methods known as temporal difference learning (TD(0)-learning) (Sutton and Barto, 1998). We find these to perform robustly without the need for careful initialisation of the model parameters. In the following we first briefly describe TD(0) learning of the value function in general and subsequently explain its application in reduced dimensional spaces.

5.4.1 Preliminaries

The general goal of learning is to find a policy $\pi(\mathbf{u}|\mathbf{y})$ that maximises the expectation of the discounted accumulated reward (Sutton and Barto, 1998)

$$V^{\pi}(\mathbf{y}) = E_{\pi}\left\{\sum_{t=0}^{\infty}\gamma^{t} r_t | \mathbf{y}_0 = \mathbf{y}\right\} \tag{5.5}$$

under the state dynamics

$$\mathbf{y}_{t+1} = \mathbf{y}_t + \delta t\, \mathbf{f}(\mathbf{y}_t, \mathbf{u}_t). \tag{5.6}$$

Here, $\mathbf{y} \in \mathbb{R}^D$ denotes the (continuous) state, $\mathbf{u} \in \mathbb{R}^A$ the action and δt is a time constant. $V^{\pi}(\mathbf{y})$ is the expected return accumulated by the agent when following the policy π starting from state $\mathbf{y}_0$, γ is a discount factor and r_t denotes the instantaneous reward collected at time t. $V^{\pi}(\mathbf{y})$ can also be identified as the value function of the policy π.

5.4.2 TD(0) V-Learning

Temporal difference learning methods update the estimate of the value function or Q-function based on the one-step temporal difference (also known as the Bellman error) (Sutton and Barto, 1998). In our experiments we used the variant of TD-learning that uses a function approximator to learn the value function in continuous space (Doya, 2000). For this, we used a parametric model of the form

$$\hat{V}(\mathbf{y}) = \mathbf{w}^T \mathbf{b}(\mathbf{y}) \tag{5.7}$$

where $\mathbf{w} \in \mathbb{R}^M$ is a vector of weights, and $\mathbf{b}(\mathbf{y}) \in \mathbb{R}^M$ is a vector of fixed basis functions. For the basis functions we used normalised radial basis functions (RBFs) $b_i(\mathbf{x}) = \frac{K(\mathbf{x}-\mathbf{c}_i)}{\sum_{j=1}^{M} K(\mathbf{x}-\mathbf{c}_j)}$ calculated from squared exponential kernels $K(\cdot)$ around M pre-determined centres $\mathbf{c}_i$, $i = 1 \ldots M$.

During episodes, the estimate of the value function is learnt online by calculating the temporal difference

$$\delta_t = r_t + \gamma \hat{V}(\mathbf{y}_{t+1}) - \hat{V}(\mathbf{y}_t) \tag{5.8}$$

after every step and updating the value function according to the learning rule

$$\hat{V}(\mathbf{y}_{t+1}) = \hat{V}(\mathbf{y}_t) + \alpha\delta_t \tag{5.9}$$

where α is the learning rate. For our parametric model, this means the parameters of the value function are updated according to

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha\delta_t \mathbf{b}(\mathbf{y}_t). \tag{5.10}$$

Finally, using the learnt value function, actions are selected according to a soft-max policy that provided directed exploration during episodes. For this, actions are drawn from a discrete set of $|U|$ continuous actions $\mathbf{u}_i \in \mathbb{R}^d$, $i \in \{1, \ldots |U|\}$ according to a Boltzmann distribution

$$p(\mathbf{u}_i|\mathbf{y}) = \frac{\exp(\beta \hat{Q}(\mathbf{y}, \mathbf{u}_i))}{\sum_{j=0}^{|U|} \beta \hat{Q}(\mathbf{y}, \mathbf{u}_j)} \tag{5.11}$$

where β controls the rate of exploration and $Q(\mathbf{y}, \mathbf{u}_i)$ is the state-action value for action $\mathbf{u}_i$. The latter is calculated from the learnt value function using the state transition function $\mathbf{f}(\mathbf{y}, \mathbf{u})$ to perform a one-step look ahead:

$$\hat{Q}(\mathbf{y}, \mathbf{u}_i) = \hat{V}(\mathbf{y} + \delta t\, \mathbf{f}(\mathbf{y}, \mathbf{u}_i)). \tag{5.12}$$

For full details of the implementation of TD(0)-learning used in our experiments we refer the reader to (Neumann, 2005).

5.4.3 Incorporating the Latent Space State Representation

For including the state representation learnt with DR into our RL framework, we replace the high-dimensional state $\mathbf{y}$ with its DR representation $\mathbf{z}$, and modify the state update equations accordingly.

Fig. 5.6 depicts a diagram of a RL step in latent space. In particular, starting from a state in latent space, $\mathbf{z}_1$, RL selects and executes an action $\mathbf{a}$ according to its current policy, leading to a new latent space state $\mathbf{z}'_2$. The latter is then used to generate a target in the environment $\mathbf{y}'_2$ by mapping through the generative GPLVM model, which can be reached, for example, using a simple PD controller. In general, (due to tracking errors, noise, etc.) we will not reach $\mathbf{y}'_2$ but instead a slightly different state $\mathbf{y}_2$ which we must

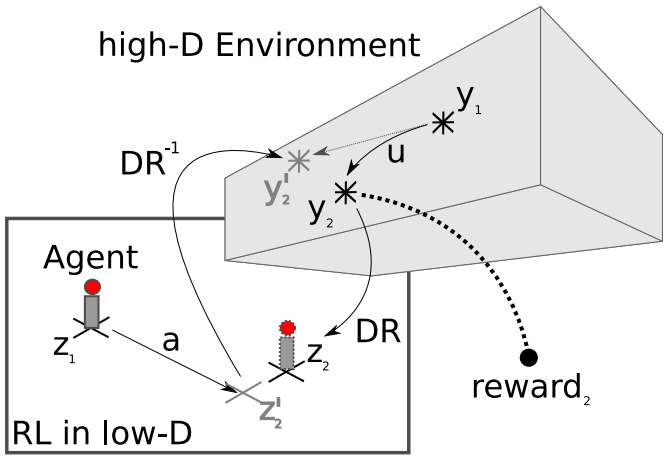


Figure 5.6: Exploiting the latent dimensionality of demonstrations for reinforcement learning.

estimate (e.g., by taking a sensor reading). It is at this point that we receive a reward (i.e., based on the true environmental state). Finally, we return to latent space via the inverse mapping (out-of-sample GP) to estimate the corresponding reduced state $\mathbf{z}_2$, which is then used to select the next action. Note that,

1. Due to the nonlinearity of the DR mapping, the same action executed in different locations of latent space may correspond to different actions in the environment.

This, however, is not a problem, if a suitably flexible policy is chosen which selects the best actions locally (i.e. dependent solely on the current state).

2. Since the RL is restricted to the smaller latent space, it may not be possible to find globally optimal, or even feasible solutions if they do not lie on this manifold.

In practice, however, this is easily rectified by the demonstrator by, for example, adjusting one or more example poses and re-learning the DR model.

5.5 Experiments

In this section we report experiments exploring the performance of learning for systems of varying complexity and size. First, in order to illustrate the concepts involved, we apply our method to a simulated 4-D toy system with linear state dynamics. We then test the scalability of the method to (i) more complex non-linear state dynamics and (ii) higher dimensional systems. For the latter we test our approach on a reaching experiment using the 7-DOF KUKA lightweight arm (Fig. 5.11) as well as the 19-DOF KHR-1HV humanoid (Fig. 5.14(a)).

5.5.1 Bimanual Reaching in End-effector Space

The goal of our first experiment is to assess the efficiency of the proposed approach compared to that of standard RL for a simple toy system. For this, as an intuitive example, we chose to investigate the problem of carrying a ball to a target using a bimanual strategy, similar to the example described in Section 5.2.1. Note that, this task can be framed in either (i) end-effector space, or (ii) joint space of a pair of robotic manipulators. In this section we first investigate the problem in end-effector space, before looking at the more complex joint space problem in Section 5.5.2.

We assume that the full state of the system can be described by the horizontal-planar coordinates of the two hands, $\mathbf{y} \in \mathbb{R}^4$. We further assume that the two hands can be moved independently with actions $\mathbf{u} \in \mathbb{R}^4$, resulting in state transitions with linear dynamics, i.e.,

$$\mathbf{y}_{t+1} = \mathbf{y}_t + \delta t\, \mathbf{B}\mathbf{u} \tag{5.13}$$

where for simplicity we chose $\mathbf{B} = \mathbf{I}$ and the time step was fixed at $\delta t = 0.1$.

In this experiment, successful carrying requires that the two hands remain either side of the ball, at a fixed distance apart throughout the trajectories. This can be expressed as the pair of constraints

$$x_1 = x_3 + \delta \qquad x_2 = x_4 \tag{5.14}$$

where $\delta = 0.1m$ is the width of the ball. Note, however, that this information is *not explicitly available* to the learner and therefore must be learnt either (i) from experience (i.e. exploration), or (ii) from the examples given to the learner as demonstrations.

The task of the learner is to find a movement that brings the hands to a target $\mathbf{x}^* = (0,0,0.1,0)^T$ without dropping the ball. For this, the learner was rewarded according to a Gaussian reward function defined over the full 4-D state-space:

$$R(\mathbf{y}) = \exp\left\{-\theta \|\mathbf{y} - \mathbf{y}^*\|^2\right\} \tag{5.15}$$

where $\theta = 0.75$ and $\mathbf{y}^* = (0,0,0.1,0)$ denotes the target state. Under this reward function the learner receives very little reward in most parts of the space, but this rapidly increases as the hands approach the target. To increase the difficulty of this problem, we also placed a large obstacle ($0.5 \times 0.5m$) in the environment between the mean start state and the target (see Fig. 5.8). Episodes terminated prematurely and a penalty of $R_0 = -1$ was added to the accumulated reward, when the learner (i) hit an obstacle, (ii) hit the boundaries of the state space or (iii) broke the constraint (5.14) and dropped the ball.

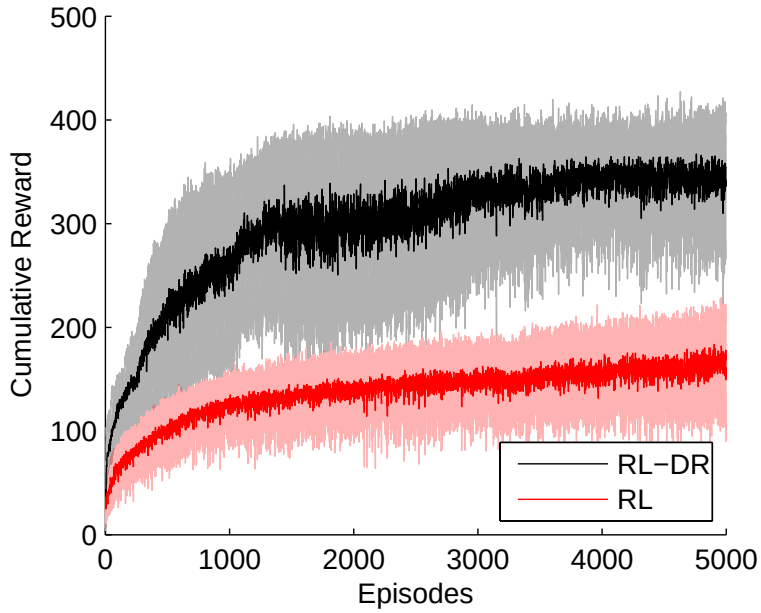


Figure 5.7: Cumulative reward against number of episodes for the two-hand problem when learning in the full 4D state space (red) and the reduced 2D space (black). The mean±s.d. over 25 trials are shown.

We compared TD(0)-learning in the full 4-D state space $\mathbf{y} \in \mathbb{R}^4$ against the proposed framework in which a reduced state representation $\mathbf{z} \in \mathbb{R}^2$ is learnt from demonstrations, prior to applying RL (TD(0)) in the resulting space. For the latter, to simulate demonstration of successful postures, we randomly sampled 200 points across the space in which the constraint (5.14) was satisfied. These were used to train a GPLVM.

For the reinforcement learning, we chose a learning rate of $\alpha = 0.9$ and discount factor $\gamma = 0.95$. We used a Gaussian Radial Basis Function (RBF) network to approximate the value function in continuous space as described in Section 5.4.2. For learning in the latent space, the centres of the RBF network were placed evenly on a 20×20 grid; for learning in the 4-D space the grid was scaled up to $20 \times 20 \times 20 \times 20$ grid. In both cases the width of the RBFs was chosen to yield a suitable overlap. The weights of the RBF network were initialised randomly in every trial. Training was conducted for 5000 episodes, with each episode lasting 500 steps ($50\,s$). The start state of each episode was drawn from a Gaussian distribution $\mathcal{N}(\mathbf{y}_0, 0.1)$ around the point $\mathbf{y}_0 = (1, 1, 1.1, 1)^T$.

The agent used a soft-max policy (5.11) with discrete actions to navigate the state-space, where we set $\beta = 10.0$. The action set contained actions that simply moved the agent forward and backward in each dimension. For example, for learning in

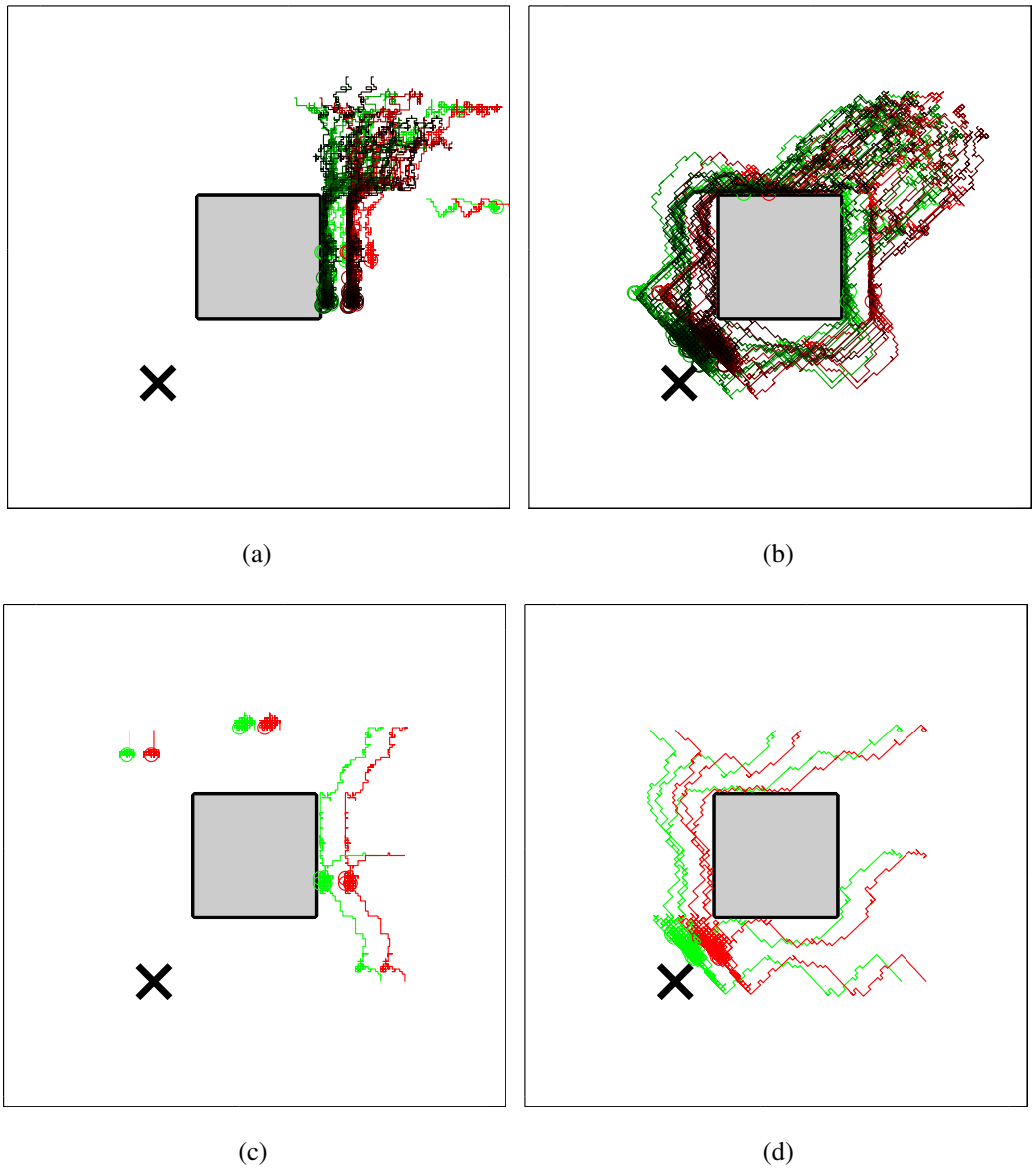
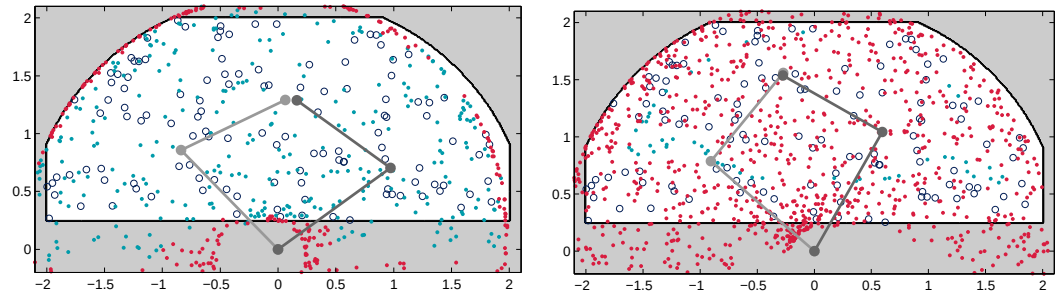


Figure 5.8: (a,b) Left (green) and right (red) hand trajectories generated at equal intervals throughout 5000 episodes of training in 4D (a) and 2D (b). Darker colours indicate later phases of learning. (c,d) 5 test trajectories sampled from different starting points after 5000 episodes of training with the two approaches (c-4D, d-2D). The grey area indicates the location of the obstacle and the target is marked with an 'x'.

the 2-D latent space the action set consisted of $U = \{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4\} = \{(0.1, 0)^T, (-0.1, 0)^T, (0, 0.1)^T, (0, -0.1)^T\}$. The 4-D agent was similarly given actions enabling it to move forward and backward in each of the 4 state space dimensions, but with an additional 4 actions corresponding to coordinated movement of the two hands $U' = \{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4\} = \{(0.1, 0, 0.1, 0)^T, (-0.1, 0, -0.1, 0)^T, (0, 0.1, 0, 0.1)^T, (0, -0.1, 0, -0.1)^T\}$. The latter were provided to ensure that, if the policy is optimal, the 4-D agent could reach the goal in the same number of steps as that of the 2-D agent accumulating the same amount of reward.

The experiment was repeated for 25 trials and the reward, accumulated in each episode of learning, was recorded. The results are shown in Fig. 5.7. As can be seen, in the initial phase of learning, the average reward accumulated by the two learners increases rapidly. However, when learning in the full 4-D space, beyond the first 500 episodes the average accumulated reward starts to level out and the progress of learning is slow. In contrast, learning in the reduced dimensional space proceeds much quicker, with convergence already after approximately 3000 episodes.



(a) GPLVM (b) PCA

Figure 5.9: Evaluating DR for the planar 2-link arms: Do points generated from latent space keep a distance of 0.1 between the two end-effectors? Light blue dots: end-effector position of left arm (right arm not shown) for which the distance between end-effectors lies within 0.005 of 0.1, red dots: distance differs by more than 0.005 from 0.1, circles: demonstrations used for DR. Top: GPLVM, bottom: PCA. In both figures the configuration of the arms is plotted for one example point.

The reason for the performance difference becomes clear when looking at the trajectories generated during training. In Fig. 5.8(a)-(b) we show examples of trajectories generated at regular intervals during training with the two approaches. Clearly, due to the higher dimensionality learning in the full 4-D state space requires far more exploration to cover the same proportion of state-space. The trajectories generated during

	Bimanual TS	Bimanual JS	KUKA arm
PCA	0.05 ± 0.01	3.25 ± 0.41	5.74 ± 0.15
GPLVM	0.24 ± 0.18	0.80 ± 0.70	0.14 ± 0.05
PCA	100.00 ± 0.00	4.92 ± 0.81	0.94 ± 0.42
GPLVM	94.50 ± 5.61	61.03 ± 6.16	68.16 ± 9.01

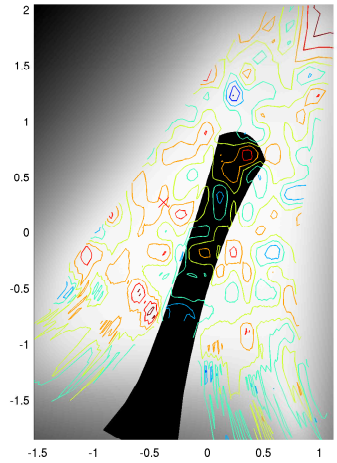
Table 5.3: Top: Reconstruction error (RMSE $\times 10^2$) of learnt DR model for 1000 random points in end-effector space. Bottom: Percentage of 1000 randomly sampled points in latent space which fulfill constraints (cf. Figures 5.9, 5.13). Shown are mean$\pm$s.d. over 20 trials, repeating DR with different random points.

training also appear to be shorter than those generated with the DR model, despite both having to avoid the same obstacle and state-space boundaries. The difference however, is that the trajectories generated in the latent space of the DR model *automatically satisfy the constraint on the hands* (5.14). This means that exploration is focused only on the reduced part of the space in which possible solutions lie, and exploration of actions that lead to the ball being dropped is avoided. As a result, the learner that uses the DR model rapidly learns a policy that allows it to satisfy the constraints and reach the goal from a larger range of the state space (compare example trajectories in Fig. 5.8(c) and (d)).

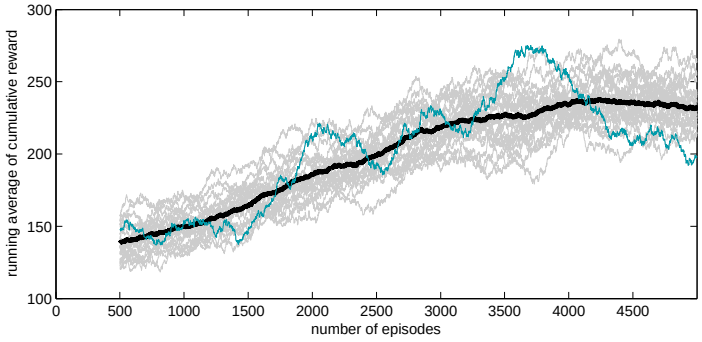
It is not given a priori that the DR actually represents the constraints correctly. In this example there exists a simple linear translation between the 2D constrained space and the 4D space of the hand positions. Consequently already linear PCA gives good results and even outperforms the GPLVM as shown in Table 5.3. In our next experiment, however, the nonlinear relationship between the spaces introduced through the kinematics necessitates the use of nonlinear DR techniques.

5.5.2 Bimanual Reaching in Joint space

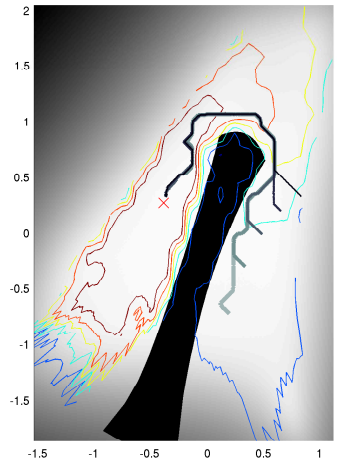
Our second experiment is formulated in joint space of the toy problem of the previous section. We define 2 planar 2-link arms which consist of an upper arm of length $1.2\,m$ and a lower arm of length $1\,m$, rooted at the origin of our space (see Fig. 5.9). The constraints of the problem are the same as above and the full state space still is 4D, but instead of end-effector positions, the state is described by the joint angles of the robot. Therefore, the kinematics of the arms introduces nonlinearities into the problem which can not be handled by linear DR techniques. To illustrate this point we first compare



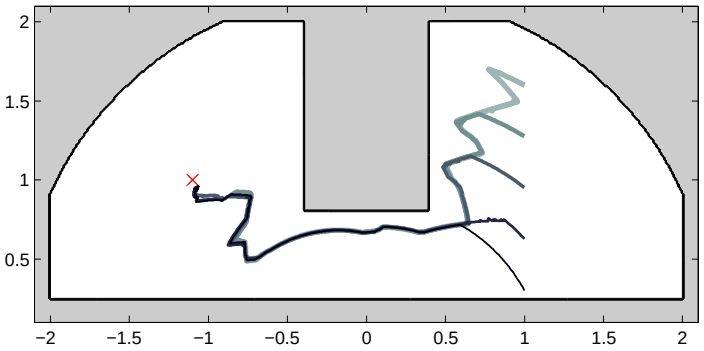
(a) initial value function



(b) rewards during learning, thick black line: average over the plotted trials, blue line: trial for which results are plotted below



(c) learnt value function with example trajectories



(d) example trajectories in end-effector space (left hand only), red cross: goal

Figure 5.10: RL results for the planar 2-link arms. Shading in (b) and (c) visualises predictive variance of GPLVM generative mapping (white means low variance and high confidence). Black object is latent space representation of the obstacle after mapping through out-of-sample GP of GPLVM.

the GPLVM with PCA and evaluate their ability to represent the constraint that the two end-effectors keep a fixed distance.

Fig. 5.9 shows a visualisation of our simulation in which several data sets are plotted. For clarity all data points shown are from the left hand of the robot only. The circles depict 123 randomly sampled data points which fulfil the constraints. We executed dimensionality reduction on their joint space representation. We then drew uniform samples from the resulting latent space and mapped them to joint angles of the robot using the generative DR mapping. The dots are the corresponding hand positions as computed with the forward kinematics of the robot. They are colour coded according to whether they fulfil the constraint within a small error margin (blue for fulfilling, red for breaking the constraint). The results clearly show that PCA (Fig. 5.9(b)) can only correctly represent the constraint in a very small region of end-effector space while the GPLVM (Fig. 5.9(a)) covers almost the complete work space. Table 5.3 further documents this result.

Having established that the constraints are correctly represented by the GPLVM we ran RL in its latent space. We used the above setup with the following changes: we set the width of the Gaussian reward to $\theta = 0.35$, learning rate to $\alpha = 0.8$, discount factor to $\gamma = 0.99$, time constant to $\delta t = 0.05$, soft-max policy to $\beta = 20$, extended the action set to also include diagonal actions and ran the learning for 5000 episodes with 1000 steps each (or the episode was stopped prematurely under the conditions given above). We again introduced an obstacle which this time only allowed successful trajectories to pass through a corridor in end-effector space (Fig. 5.10(d)). Start states were drawn uniformly across latent space.

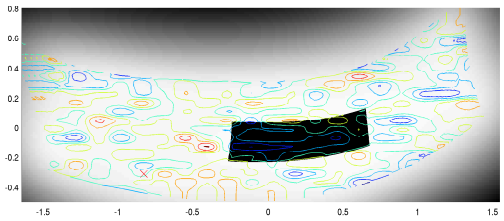
Results of the learning are presented in Fig. 5.10. Fig. 5.10(b) shows running averages over a window of 500 episodes of the cumulative reward per episode for 25 runs of RL (trials). We plot running averages, because the random start states mean that the cumulative reward per episode is highly variable. The accumulated reward clearly increases with learning. For the trial highlighted as the blue line we present the initial and learnt value functions in latent space in Figures 5.10(a) and 5.10(c), respectively. As demonstrated by the drawn sample trajectories in Fig. 5.10(c) the learnt policy successfuly solves the task by leading trajectories around the obstacle into the goal. Fig. 5.10(d) depicts the resulting trajectories in end-effector space. For clarity reasons we only plot the trajectories of the left hand, but right hand trajectories follow with the desired distance of $0.1\,m$ behind the left hand.

Figure 5.11: Anthropomorphic DLR/KUKA light-weight arm (LWR-III) in reaching experiment.

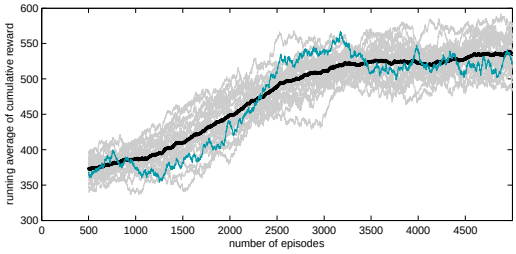
5.5.3 A Planar DLR-Arm Problem

We also tested the suitability of dimensionality reduction for reinforcement learning with the anthropomorphic, 7-DOF KUKA-arm (Fig. 5.11). Setting the position and orientation of its end-effector constrains 6 of the 7 DOF. The remaining redundant DOF is resolved in a consistent way in the inverse kinematics of the robot (see Section 2.3.1 for details). The task for the robot was to learn to reach around an obstacle while its end-effector is constrained to move on a plane (see Fig. 5.13). We additionally fixed the orientation of the end-effector on the plane which left only 2 remaining DOF suggesting to use a 2D latent space for the DR. We ran the DR on joint space points which were computed from 100 randomly sampled positions in end-effector space which fulfilled the described constraints. Again, our results show that PCA is not sufficient to represent the constraints in a low-dimensional space while the GPLVM faithfully generates correctly constrained points (see Fig. 5.13 analogous to Fig. 5.9 and Table 5.3). For this example we also plot the corresponding latent spaces (Figures 5.13(c) and 5.13(d)) and note that there is a good correspondence between the predictive variance of a point generated with the GPLVM and whether this point fulfils the constraints, or not. This means that low-confidence latent points can be discarded by the RL without expensive evaluations of these points in the real world.

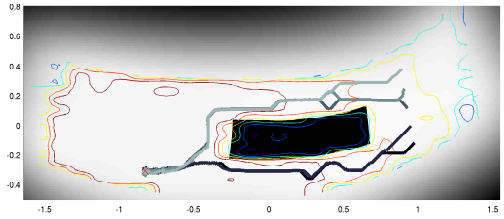
We conducted RL experiments in the GPLVM latent space using the same parameters as for the planar 2-link arms except for a change of the reward parameter to $\theta = 4$, because of the reduced scale of the end-effector space. Fig. 5.12 presents results analogous to Fig. 5.10 for the planar 2-link arms. The RL consistently found policies which allow the arm to reach the target while avoiding the obstacle and staying on the constraint plane. We successfuly tracked the planned trajectories with our real arm using a PD-controller (see accompanying video).



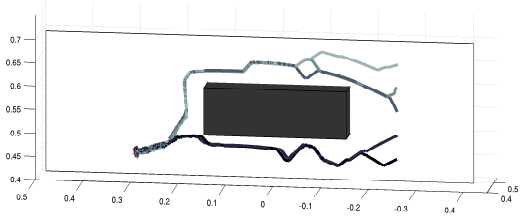
(a) initial value function



(b) rewards during learning, thick black line: average over the plotted trials, blue line: trial for which results are plotted below



(c) learnt value function with example trajectories

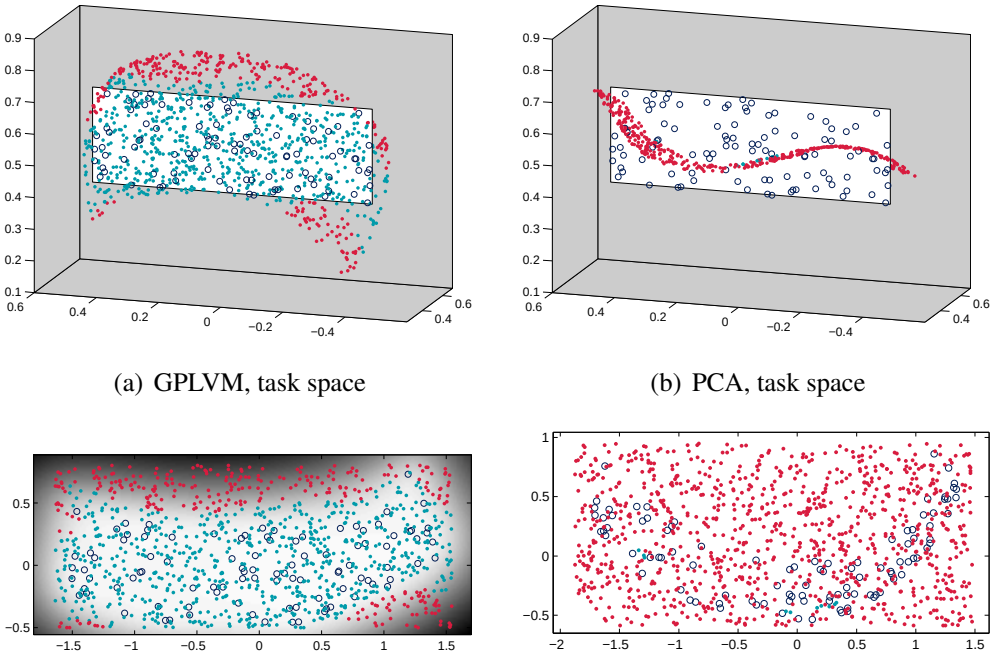


(d) example trajectories in end-effector space

Figure 5.12: RL results for the KUKA arm. Shading in (a) and (c) visualises predictive variance of GPLVM generative mapping (white means low variance and high confidence). Black object is latent space representation of the obstacle after mapping through out-of-sample GP of GPLVM.

5.5.4 Full-Body Humanoid Reaching

In our final experiment we demonstrate the complete approach on the 19 DOF humanoid robot KHR-1HV (Fig. 5.14) which task it is to lift an object while avoiding potential obstacles. Instead of devising an inverse kinematics for this task by hand we demonstrated individual poses of two alternative ways of lifting an object (7 poses in total). Of the 19 DOF 10 were major contributors to the changes in postures, the remaining 9 changed by less than 10 degrees across different postures (see Fig. 5.14(b)). The realised postures all lay on a central y-z-plane of the robot, i.e., the hands of the robot did not move sidewards (subject to noise originating from the manual demonstrations). Therefore, the space of related movements was inherently 2D which motivated the use of 2D latent spaces for dimensionality reduction. Compared to the previous examples, PCA already performed remarkably well in reconstructing and interpolating the demonstrated postures. The remaining inaccuracies, however, meant that the robot excessively leaned backwards (see Fig. 5.14(c)) which caused it to fall after transition between relevant poses. With the GPLVM, on the other hand, we were able to learn



(a) GPLVM, task space (b) PCA, task space

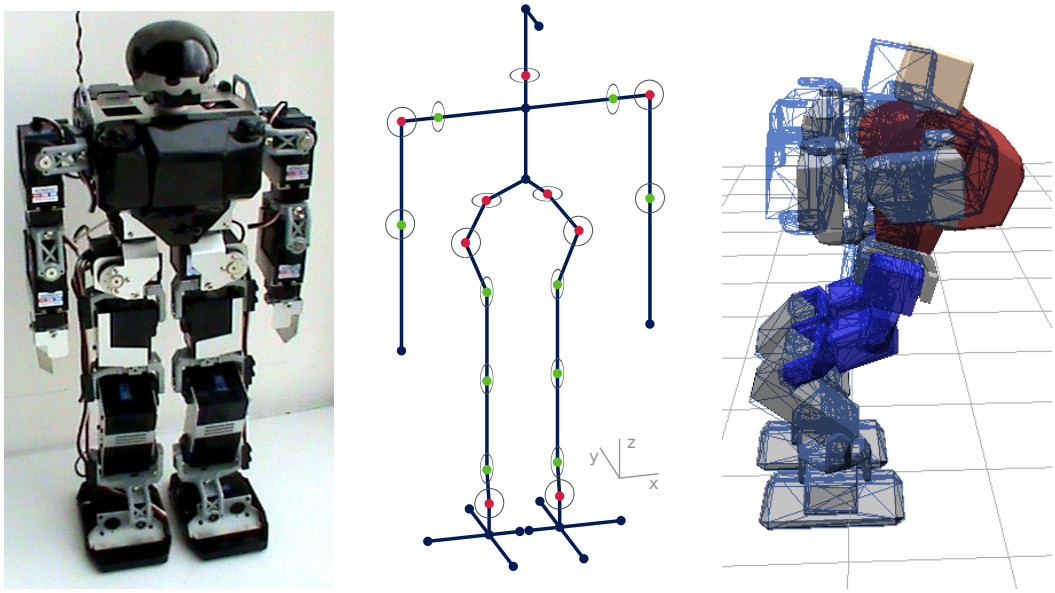
(c) GPLVM, latent space (d) PCA, latent space

Figure 5.13: Evaluating DR for the DLR arm: Do points generated from latent space lie on the constraint plane in task space? Light blue dots lie within a distance of 0.003 (3mm) from the plane, red dots further away, dark blue circles show data points used for DR. GPLVM (left) is contrasted against PCA (right). Task space (top, end-effector position in m) and latent space (bottom) are shown. Shading in 5.13(c) corresponds to confidence in generative GPLVM mapping.

a 2D latent space which almost perfectly reconstructed the demonstrated postures and stayed away from unstable positions in high confidence regions between the demonstrations.

We applied RL in the learnt latent space with some changes to the parameter settings of Section 5.5.2. In particular, we replaced the Gaussian reward function in eq. (5.15) with a more pointed Laplacian $R(\mathbf{x}) = \exp\{-\theta\|\mathbf{x}-\mathbf{x}^*\|\}$ with $\theta = 20$ [5], increased the discount factor to $\gamma = 0.995$, set $\delta t = 0.5$, allowed for more stochastic action selection $\beta = 10$ and reduced the number of steps per episode to 200. An episode was aborted when an obstacle was hit, or when the predictive variance of a latent point was larger than 0.0004 (corresponding to standard deviation of 1.15 degrees in each joint). The latter criterion is an indirect measure of constraint fulfillment

[5] The reward was defined over the positions of the hands. We used the forward kinematics to evaluate the generated movements in simulation.



(a) KHR-1HV (b) skeleton (c) PCA error

Figure 5.14: KHR-1HV and its degrees of freedom. The skeleton in (b) shows the 19 DOF of the robot. The 10 joints shown in green contribute most to the movement. The remaining 9 joints (red) change by less than 10 degrees. Orientation of circles indicates axis of rotation joints: x-verticle ellipse, y-circle, z-horizontal ellipse. (c) shows the robot in a demonstrated pose reconstructed by the GPLVM (wire frame) and PCA (solid). There is no visual difference between the GPLVM pose and the original demonstration.

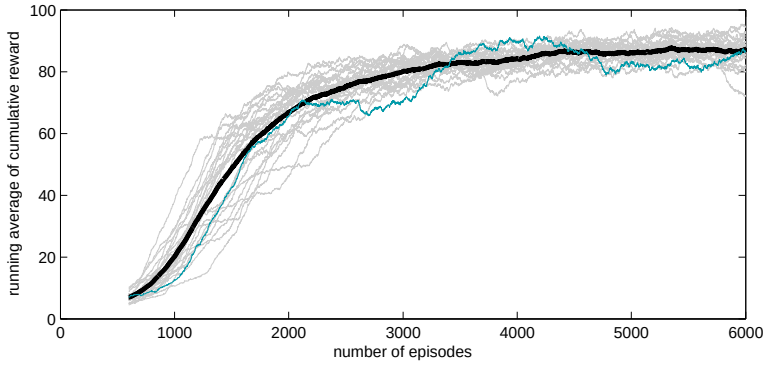
and replaces the direct measures from the previous experiments as they are unavailable in this completely unsupervised setting where the only information about the task is given indirectly by the demonstrations themselves.

In Fig. 5.15 we present results of RL on this problem. As in the previous examples RL consistently learnt good approximations of the value function and resulting policies moved the hands of the robot to the target while avoiding the obstacle.

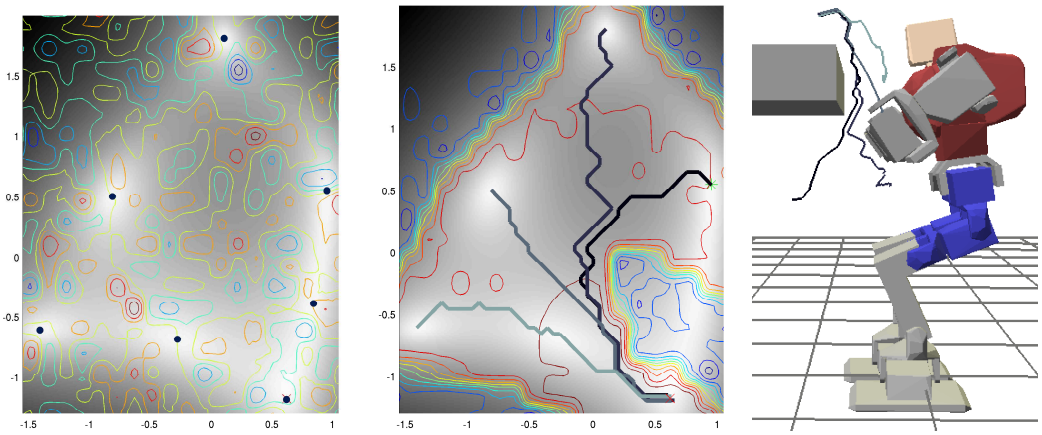
In the accompanying video we present these results on the real robot. We show an example demonstration, explore the resulting latent space online and execute trajectories of the learnt policy (see also Fig. 5.16).

5.6 Discussion

In this chapter we have explored the use of nonlinear DR to learn state abstractions for RL from demonstrated postures. This approach is based on the assumption that the



(a) Running average (over 600 episodes) of rewards for each of the 25 trials. Thick black line: average over the plotted trials. Blue line: trial for which results are plotted below.



(b) initial value function, dots: demonstrated postures

(c) learnt value function with example trajectories

(d) example trajectories in end-effector space

Figure 5.15: RL results for the KHR-1HV. Shading in (b) and (c) visualises predictive variance of GPLVM generative mapping (white means low variance and high confidence).

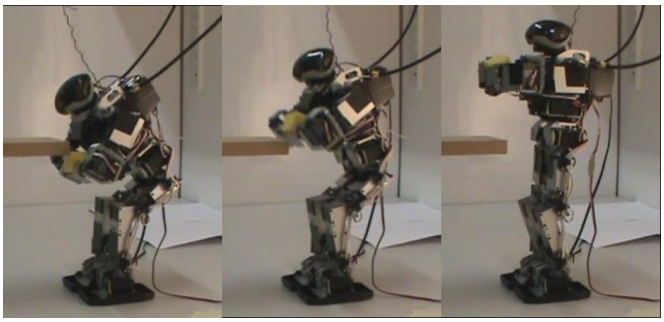


Figure 5.16: Video frames of successful, obstacle avoiding trajectory executed on KHR-1HV after reinforcement learning.

considered tasks introduce constraints on the kinematics of the robot which define hidden low-dimensional structure in the robot's joint space. DR then exploits this structure to automatically determine low-dimensional state representations for RL which make RL feasible also for humanoid bodies with many DOF.

While PCA, as a linear method, often finds reasonably good approximations of the underlying structure, we have shown here that the GPLVM, as a nonlinear, nonparametric method, can provide superior accuracy in the representation of the underlying constraints. Additionally we have seen that already from as little as 7 individually demonstrated postures the GPLVM provides latent spaces which allow smooth interpolation between them and so allows to extend the procedure of kinaesthetic demonstration to full-body humanoid movements.

In previous chapters we were concerned about a particular structure in latent space which simplifies interpolation of postures. This is not critical in this application, because RL (depending on the method used) is sufficiently flexible to account for distortions in the state representation as long as the state representation roughly maintains the relative configuration of latent points desired for the task, i.e., the order of postures is not shuffled in latent space, and it is possible to smoothly interpolate between postures in latent space. In most cases it is, therefore, sufficient to use PCA as initialisation for the GPLVM (true in all our examples) as it provides the desired continuity.

Yet, this is no guarantee that the latent spaces resulting from GPLVM optimisation have the necessary properties. In particular, it can happen that after GPLVM optimisation the latent space is disrupted in groups of postures which are separated through regions of high predictive variance. In this situation it is often fruitful to experiment with different initialisations of the optimisation (both of latent points, see ch. 3, and values of covariance parameters, especially noise), or to increase the dimensionality of the latent space.

If the order of latent points deviates severely from the underlying logical order of postures with respect to task space, it might be impossible for RL to find a solution to the given problem. This situation can occur when the demonstrated postures exhibit variation which is unrelated to the task. In general, this corresponds to the situation in which the variance of what we regard as noise in the demonstrations is larger or equal to the variance of the desired, task related changes across demonstrated postures. Without additional information it is impossible for DR to resolve this problem on its own. If we know more about the underlying structure of postures and have a structure hypothesis, we can use similar methods as described in Section 4.3 to use the structure

hypothesis to guide the GPLVM optimisation.

In an RL setting the reward of a given posture is a natural source of information about the significance of a posture for the task. Morimoto et al. (2008) have shown how the reward can be used to guide *linear* DR. In the light of the inaccuracies of linear DR presented here, extending their work to the nonlinear case in our framework presents itself as a promising direction for future research. However, also this approach is limited. For example, in RL the reward is often defined only locally for the situation when a task has been achieved and then does not give any information about postures for which the task was not achieved, but which potentially contributed to its achievement. So ideally we would use the value function to guide the DR, but this is what is learnt in RL and so we would end up in a catch-22 scenario as we then would need RL to do DR and vice-versa.

Another disadvantage of using the reward in DR is that DR then is optimised for one particular setting of the reward while the approach presented here tries to capture structure in task space more generally and not tied to one particular instance of that task. For example, in reaching we try to capture the space of end-effector positions without taking the goal into account to be able to reuse the learnt latent space for different start and goal positions. This is related to the general tradeoff between compact representations and freedom of exploration. The computational advantage of DR as state abstraction for RL is based on constraining exploration to the manifold defined by the demonstrations. Conversely, this, of course, also means that RL will not be able to find solutions off that manifold, for example, when the task changes compared with the demonstrations.

The RL method employed in this chapter is very basic. Yet, we were able to show that in our framework, even using this method, RL can successfully learn full-body humanoid movement tasks. However, the large number of episodes needed before converging to an acceptable solution still necessitated the use of simulation in our experiments. For online experiments with the real robots a more efficient RL method has to be found which takes a smaller number of episodes until convergence. The framework presented in this chapter is modular (DR and RL are decoupled). Therefore, it is simple to replace either one as soon as a better method becomes available.

Chapter 6

Conclusion

In this thesis we have investigated the use of nonlinear dimensionality reduction for learning of movements from demonstrations. We have compared generative DR methods and selected the Gaussian process latent variable model for its power and flexibility in representing the nonlinear relationships between latent and observed variables even when only a small amount of data is available. We have provided an interpretation of the GPLVM which allowed us to 1) understand the dependence of a successful application of the GPLVM on a good approximation of the underlying covariance matrix and 2) devise an initialisation for the GPLVM which outperformed PCA. We have further suggested a modification of the GPLVM which let us constrain the structure in latent space and so allowed us to represent a set of related movements with a single dynamic movement primitive. Finally, we have demonstrated the utility of GPLVM latent spaces also during online control using the example of reinforcement learning.

In all our experiments we contrasted the results of linear (PCA) and nonlinear (GPLVM) DR and hence could show that linear DR is not sufficient to represent movements in latent spaces with sufficiently low dimensionality to relate latent to task variables while this can be achieved with nonlinear DR. Apart from this advantage in interpretability of nonlinear latent variables, the lower dimensionality achieved for a given accuracy level also means reduced computational cost for applications in which DR is embedded. For example, it makes a large difference in reinforcement learning, if the value function only needs to be defined in two rather than four or more dimensions.

The representational advantage of nonlinear over linear DR is achieved at the cost of higher computational and model selection demands during DR. In contrast to PCA, the optimisation of the GPLVM consequently poses two challenges: 1) $O(N^3)$ complexity of each gradient step and 2) local optima. We approached 2) by proposing

the GPLVM-MDS which again lead to increased computational costs. Fortunately, our learning from demonstration setting allows us to safely assume a sufficiently small size of data sets such that 1) is no major issue in our case. Nevertheless, sparse approximations for GPs and the GPLVM have been proposed (Lawrence, 2007) and could be used on larger data sets (although successful applications of these have not frequently been reported).

Even though the GPLVM is a nonparametric model in which the necessary functional forms are chosen automatically dependent on the requirements of the data, the user still has to make model selection choices by deciding for a particular form of covariance function. In our work we followed the most common practice of using the squared exponential covariance function. Through this choice we make the implicit assumption that task variables have a particularly smooth relation to joint angles. Given the predominantly linear and trigonometric nature of the kinematics this assumption is valid in most cases, but care has to be taken to avoid singularities in the data (e.g. joints jumping from π to $-\pi$[1]). Despite the basic fit of smoothness assumptions the GPLVM still requires sufficiently dense samples of data points to model it well. For the small learning from demonstration data sets it is therefore beneficial that the GPLVM allows to easily incorporate prior information into the model as exemplified in chapter 4. Similarly, such an approach may be attempted, when the data does not entirely fit our assumptions and exhibits variability unrelated to the task which may result in a considerably distorted model after learning. Then, even when including prior information about the problem, the GPLVM may learn a different combination of latent configuration and generative mapping than defined by the underlying task. In this case interpolated movements, although potentially smooth, may not entirely fit the expectations of the demonstrator, but given only limited information about the demonstrated movements in our setting this is unavoidable.

In conclusion, we have demonstrated the benefits of nonlinear dimensionality reduction in a learning from demonstration scenario in which, apart from the joint angles of the demonstrations themselves, minimal information is available. While we obtained promising results, our experiments also indicated that the lack of information about the task has to be compensated with stronger assumptions about the data.

[1]Note that joints extremely rarely move through full cycles. Therefore it is safe in most cases to correct a jump from π to $-\pi$ by adding 2π to the negative values

Appendix A

Scale of Probabilistic PCA Latent Variables

The maximum likelihood solution for parameters in the probabilistic PCA model as defined in eq. 3.5 is (see e.g. Bishop, 2006, Section 12.2.1)

$$\mathbf{W} = \mathbf{U}(\mathbf{L} - \sigma^2\mathbf{I})^{\frac{1}{2}}\mathbf{R} \tag{A.1}$$

where $\mathbf{U}$ and $\mathbf{L}$ are the eigenvectors and eigenvalues, respectively, of the sample covariance matrix $\mathbf{S}_D = \frac{1}{N}\mathbf{Y}^T\mathbf{Y}$ and $\mathbf{R}$ is an arbitrary rotation matrix which we set $\mathbf{R} = \mathbf{I}$ from here on without loss of generality. The posterior mean of a latent point $\mathbf{z}$ is

$$\mathbf{z} = (\mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I})^{-1}\mathbf{W}^T\mathbf{y}.$$

As under eq. (A.1) $\mathbf{W}^T\mathbf{W} = \mathbf{L} - \sigma^2\mathbf{I}$, substituting the ML solution for $\mathbf{W}$ yields

$$\mathbf{z} = \mathbf{L}^{-1}(\mathbf{L} - \sigma^2\mathbf{I})^{\frac{1}{2}}\mathbf{U}^T\mathbf{y}.$$

In the limit $\sigma^2 \to 0$ we therefore see that the latent variables are scaled by the square root of the inverse of the eigenvalues of $\mathbf{S}_D$

$$\mathbf{z} = \mathbf{L}^{-\frac{1}{2}}\mathbf{U}^T\mathbf{y}$$

and for the full data set in matrix notation

$$\mathbf{Z} = \mathbf{Y}\mathbf{U}\mathbf{L}^{-\frac{1}{2}}. \tag{A.2}$$

Proposition 2 *The mean of latent points* $\mathbf{Z}$ *found by probabilistic PCA in the limit* $\sigma^2 \to 0$ *is 0 in all dimensions and their standard deviation is 1.*

Proof: Let $\mathbf{z}_{:,i}$ be the latent values in dimension i, i.e., the i-th column of $\mathbf{Z}$. From eq. (A.2) we then have

$$\mathbf{z}_{:,i} = \mathbf{Y}\mathbf{u}_i l_i^{-\frac{1}{2}}$$

where $\mathbf{u}_i$ and l_i are the i-th eigenvector and eigenvalue, respectively, of $\mathbf{S}_D$. The mean of $\mathbf{z}_{:,i}$ then is

$$\bar{\mathbf{z}}_{:,i} = \frac{1}{N}\mathbf{1}_N^T \mathbf{z}_{:,i} = \frac{1}{N}\mathbf{1}_N^T \mathbf{Y}\mathbf{u}_i l_i^{-\frac{1}{2}} = \frac{1}{N}\mathbf{0}_D^T \mathbf{u}_i l_i^{-\frac{1}{2}} = 0,$$

because $\mathbf{Y}$ is centred ($\mathbf{1}_N$ is a vector of N ones and $\mathbf{0}_D$ is a vector of D zeros). As $\mathbf{z}_{:,i}$ has mean 0, its variance can be written as

$$\text{var}(\mathbf{z}_{:,i}) = \frac{1}{N}\mathbf{z}_{:,i}^T \mathbf{z}_{:,i} = \frac{1}{N} l_i^{-1} \mathbf{u}_i^T \mathbf{Y}^T \mathbf{Y}\mathbf{u}_i = l_i^{-1}\mathbf{u}_i^T \mathbf{S}_D \mathbf{u}_i = l_i^{-1}\mathbf{u}_i^T \mathbf{u}_i l_i = 1.$$

Consequently, the standard deviation of $\mathbf{z}_{:,i}$ also is 1 in all dimensions i.

Proposition 2 shows that in the limit of no noise the latent variables follow their Gaussian prior. In dual probabilistic PCA no prior over latent variables is defined. Consequently, their scale differs from original probabilistic PCA. The solutions of the two models are still related through the matrices $\mathbf{Y}^T\mathbf{Y}$ and $\mathbf{Y}\mathbf{Y}^T$, though. In particular, the latent variables of dual probabilistic PCA can be written in terms of the probabilistic PCA latent variables

$$\mathbf{z}_{:,i}^{\text{dual}} = \left(\frac{l_i}{D}\right)^{\frac{1}{2}} \mathbf{z}_{:,i} = D^{-\frac{1}{2}} \mathbf{Y}\mathbf{u}_i.$$

Appendix B

Calculation of Gradients

B.1 Likelihood Gradients for Out-of-sample GPLVM Mapping

$$\frac{\partial L(\mathbf{z}^*)}{\partial \mathbf{z}^*} \propto \frac{1}{2} \frac{\frac{\partial (\mathbf{y}^* - \mu(\mathbf{z}^*))^\top (\mathbf{y}^* - \mu(\mathbf{z}^*))}{\partial \mathbf{z}^*} \sigma^2(\mathbf{z}^*) - (\mathbf{y}^* - \mu(\mathbf{z}^*))^\top (\mathbf{y}^* - \mu(\mathbf{z}^*)) \frac{\partial \sigma^2(\mathbf{z}^*)}{\partial \mathbf{z}^*}}{\sigma^4(\mathbf{z}^*)} \tag{B.1}$$

$$\begin{aligned} \frac{\partial \sigma^2(\mathbf{z}^*)}{\partial \mathbf{z}^*} &= \frac{\partial k^*(\mathbf{z}^*)}{\partial \mathbf{z}^*} - \frac{\partial \mathbf{k}^{*\top}(\mathbf{z}^*) \mathbf{K}^{-1} \mathbf{k}^*(\mathbf{z}^*)}{\partial \mathbf{z}^*} \\ &= \frac{\partial k^*(\mathbf{z}^*)}{\partial \mathbf{z}^*} - 2 \left[\frac{\partial \mathbf{k}^*(\mathbf{z}^*)}{\partial \mathbf{z}^*} \right]^\top \mathbf{K}^{-1} \mathbf{k}^*(\mathbf{z}^*) \end{aligned} \tag{B.2}$$

$$\begin{aligned} \frac{\partial (\mathbf{y}^* - \mu(\mathbf{z}^*))^\top (\mathbf{y}^* - \mu(\mathbf{z}^*))}{\partial \mathbf{z}^*} &= -2 \left[\frac{\partial \mu(\mathbf{z}^*)}{\partial \mathbf{z}^*} \right]^\top \mathbf{y}^* + 2 \left[\frac{\partial \mu(\mathbf{z}^*)}{\partial \mathbf{z}^*} \right]^\top \mu(\mathbf{z}^*) \\ &= 2 \left[\frac{\partial \mu(\mathbf{z}^*)}{\partial \mathbf{z}^*} \right]^\top (\mu(\mathbf{z}^*) - \mathbf{y}^*) \end{aligned} \tag{B.3}$$

$$\frac{\partial \mu(\mathbf{z}^*)}{\partial \mathbf{z}^*} = \mathbf{S}\mathbf{Y}^\top \mathbf{K}^{-1} \frac{\partial \mathbf{k}^*(\mathbf{z}^*)}{\partial \mathbf{z}^*} \tag{B.4}$$

The remaining gradients to be calculated are

$$\frac{\partial \mathbf{k}^*(\mathbf{z}^*)}{\partial \mathbf{z}^*} \quad \text{and} \quad \frac{\partial k^*(\mathbf{z}^*)}{\partial \mathbf{z}^*}$$

which are kernel gradients and are left to derive by the reader for their choice of kernel.

Bibliography

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*.

Bakir, G. H., Weston, J., and Schölkopf, B. (2004). Learning to find pre-images. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

Barto, A. G. and Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):341–379.

Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373–1396.

Bengio, Y., Paiement, J., Vincent, P., Delalleau, O., Roux, N. L., and Ouimet, M. (2004). Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems, NIPS, 16*. MIT Press, Cambridge, MA.

Billard, A., Calinon, S., Dillmann, R., and Schaal, S. (2008). Chapter 59: Robot programming by demonstration. In Siciliano, B. and Khatib, O., editors, *Handbook of Robotics*. Springer.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer.

Bishop, C. M., Svensen, M., and Williams, C. K. I. (1998). GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–234.

Buja, A., Swayne, D. F., Littman, M. L., Dean, N., and Hofmann, H. (2001). Xgvis: Interactive data visualization with multidimensional scaling. Technical report, AT&T Labs.

Calinon, S. and Billard, A. (2005). Recognition and Reproduction of Gestures using a Probabilistic Framework combining PCA, ICA and HMM. In *22nd International Conference on Machine Learning*, pages 105–112.

Calinon, S. and Billard, A. (2009). Statistical learning by imitation of competing constraints in joint space and task space. *Advanced Robotics*, 23:2059–2076.

Calinon, S., Guenter, F., and Billard, A. (2007). On learning, representing, and generalizing a task in a humanoid robot. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(2):286 –298.

Carreira-Perpinan, M. A. and Lu, Z. (2007). The laplacian eigenmaps latent variable model. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics, AISTATS*. poster.

Carreira-Perpinan, M. A. and Lu, Z. (2008). Dimensionality reduction by unsupervised regression. In *Proceedings of 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8.

Chalodhorn, R., Grimes, D. B., Grochow, K., and Rao, R. P. (2010). Learning to walk by imitation in low-dimensional subspaces. *Advanced Robotics*, 24:207–232(26).

Chalodhorn, R., MacDorman, K. F., and Asada, M. (2009). Humanoid robot motion recognition and reproduction. *Advanced Robotics*, 23:349–366(18).

Chiaverini, S., Oriolo, G., and Walker, I. (2008). Kinematically redundant manipulators. In Siciliano, B. and Khatib, O., editors, *Handbook of Robotics*, pages 245–268. Springer.

Cox, T. F. and Cox, M. A. A. (2000). *Multidimensional Scaling*. Chapman & Hall/CRC, 2nd edition.

Dahm, P. and Joublin, F. (1997). Closed form solution for the inverse kinematics of a redundant robot arm. Technical Report IR-INI 97-08, Ruhr-Universität Bochum, Institut für Neuroinformatik.

de Silva, V. and Tenenbaum, J. B. (2003). Global versus local methods in nonlinear dimensionality reduction. In S. Becker, S. T. and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 705–712. MIT Press, Cambridge, MA.

Degallier, S., Santos, C. P., Righetti, L., and Ijspeert, A. (2006). Movement generation using dynamical systems: a humanoid robot performing a drumming task. In *IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS06)*.

Doya, K. (2000). Reinforcement learning in continuous time and space. *Neural Computation*, 21:219–245.

D'Souza, A., Vijayakumar, S., and Schaal, S. (2001). Learning inverse kinematics. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Gams, A., Ijspeert, A. J., Schaal, S., and Lenarcic, J. (2009). On-line learning and modulation of periodic movements with nonlinear dynamical systems. *Autonomous Robots*, 27(1):3–23.

Giese, M. A. and Poggio, T. (2000). Morphable models for the analysis and synthesis of complex motion patterns. *International Journal of Computer Vision*, 38(1):59–73.

Gower, J. C. (1966). Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika*, 53(3-4):325–338.

Graef, J. and Spence, I. (1979). Using distance information in the design of large multidimensional scaling experiments. *Psychological Bulletin*, 86(1):60–66.

Grimes, D. B., Chalodhorn, R., and Rao, R. P. N. (2006). Dynamic imitation in a humanoid robot through nonparametric probabilistic inference. In *Proceedings of Robotics: Science and Systems (RSS'06)*, Cambridge, MA. MIT Press.

Grochow, K., Martin, S. L., Hertzmann, A., and Popovic, Z. (2004). Style-based inverse kinematics. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*.

Grossman, D. D. (1977). Programming a computer controlled manipulator by guiding through the motions. Research Report RC6393, IBM T. J. Watson Research Center. Declassified 1981.

Guenter, F., Hersch, M., Calinon, S., and Billard, A. (2007). Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics*, 21(13):1521–1544.

Harmeling, S. (2007). Exploring model selection techniques for nonlinear dimensionality reduction. School of Informatics Research Report EDI-INF-RR-0960, University of Edinburgh.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hoffmann, H., Pastor, P., Park, D.-H., and Schaal, S. (2009). Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Howard, M., Klanke, S., Gienger, M., Goerick, C., and Vijayakumar, S. (2009). A novel method for learning policies from variable constraint data. *Autonomous Robots*, 27(2):105–121.

Hyvärinen, A. and Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5):411 – 430.

Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1398–1403. ICRA2002 best paper award.

Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2003). Learning attractor landscapes for learning motor primitives. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems, NIPS, 15*, pages 1523–1530, Cambridge, MA. MIT Press.

Inamura, T., Toshima, I., Tanie, H., and Nakamura, Y. (2004). Embodied symbol emergence based on mimesis theory. *The International Journal of Robotics Research*, 23(4-5):363–377.

Ito, M., Noda, K., Hoshino, Y., and Tani, J. (2006). Dynamic and interactive generation of object handling behaviors by a small humanoid robot using a dynamic neural network model. *Neural Networks*, 19(3):323–337.

Kariya, T. and Kurata, H. (2004). *Generalized Least Squares*. Wiley Series in Probability and Statistics. Wiley.

Klanke, S. and Ritter, H. (2007). Variants of unsupervised kernel regression: General cost functions. *Neurocomputing*, 70(7-9):1289–1303. Advances in Computational Intelligence and Learning - 14th European Symposium on Artificial Neural Networks 2006.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69.

Konidaris, G. and Barto, A. (2009). Efficient skill learning using abstraction selection. In *Proceedings of the Twenty First International Joint Conference on Artificial Intelligence (IJCAI)*. preprint.

Kovar, L., Gleicher, M., and Pighin, F. (2002). Motion graphs. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 473–482, New York, NY, USA. ACM.

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243.

Kruskal, J. B. and Wish, M. (1978). *Multidimensional Scaling*. Sage Publications.

Kulic, D., Takano, W., and Nakamura, Y. (2008). Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains. *International Journal of Robotics Research*, 27(7):761–784.

Kwok, J.-Y. and Tsang, I.-H. (2004). The pre-image problem in kernel methods. *Neural Networks, IEEE Transactions on*, 15(6):1517 –1525.

Lawrence, N. (2005). Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of Machine Learning Research*, 6:1783–1816.

Lawrence, N. D. (2007). Learning for larger datasets with the gaussian process latent variable model. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics, AISTATS*.

Lawrence, N. D. and Moore, A. J. (2007). Hierarchical gaussian process latent variable models. In *International Conference on Machine Learning, ICML*.

Lawrence, N. D. and Quinonero-Candela, J. (2006). Local distance preservation in the GP-LVM through back constraints. In *Proceedings of the International Conference in Machine Learning (ICML)*.

Lee, D. and Nakamura, Y. (2010). Mimesis model from partial observations for a humanoid robot. *The International Journal of Robotics Research*, 29(1):60–80.

Lee, J., Chai, J., Reitsma, P. S. A., Hodgins, J. K., and Pollard, N. S. (2002). Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics (TOG)*, 21(3):491–500.

Li, H., Teng, L., Chen, W., and Shen, I.-F. (2005). Supervised learning on local tangent space. In *Advances in Neural Networks - ISNN 2005*, volume 3495/2005 of *Lecture Notes in Computer Science*, pages 546–551.

Li, L., Walsh, T. J., and Littman, M. L. (2006). Towards a unified theory of state abstraction for mdps. In *In Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*, pages 531–539.

Liégeois, A. (1977). Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Trans. Sys., Man and Cybernetics*, 7:868–871.

Lozano-Perez, T. (1983). Robot programming. *Proceedings of the IEEE*, 71:821–841.

Meinicke, P., Klanke, S., Memisevic, R., and Ritter, H. (2005). Principal surfaces from unsupervised kernel regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(9):1379–1391.

Morimoto, J. and Atkeson, C. G. (2009). Nonparametric representation of an approximated poincar map for learning biped locomotion. *Autonomous Robots*, 27(2):131–144.

Morimoto, J., Hyon, S.-H., Atkeson, C. G., and Cheng, G. (2008). Low-dimensional feature extraction for humanoid locomotion using kernel dimension reduction. In *Proc. IEEE International Conference on Robotics and Automation ICRA 2008*, pages 2711–2716.

Muehlig, M., Gienger, M., Steil, J. J., and Goerick, C. (2009). Automatic selection of task spaces for imitation learning. In *IEEE International Conference on Intelligent Robots and Systems*, St. Louis, MO, USA.

Muirhead, R. J. (2005). *Aspects of multivariate statistical theory*. Wiley, 2nd edition.

Nakamura, Y. (1991). *Advanced Robotics: Redundancy and Optimization*. Addison Wesley, Reading, MA.

Naksuk, N., Lee, C. S. G., and Rietdyk, S. (2005). Whole-body human-to-humanoid motion transfer. In *Proc. 5th IEEE-RAS International Conference on Humanoid Robots*, pages 104–109.

Nehaniv, C. L. and Dautenhahn, K. (2002). The correspondence problem. In Dautenhahn, K. and Nehaniv, C. L., editors, *Imitation in Animals and Artifacts*, pages 41–64. MIT Press.

Neumann, G. (2005). The reinforcement learning toolbox, reinforcement learning for optimal control tasks. Master's thesis, Graz University of Technology.

Pastor, P., Hoffmann, H., Asfour, T., and Schaal, S. (2009). Learning and generalization of motor skills by learning from demonstration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Peters, J. and Schaal, S. (2006). Reinforcement learning for parameterized motor primitives. In *2006 International Joint Conference on Neural Networks, IJCNN*, pages 73–80.

Peters, J. and Schaal, S. (2008a). Natural actor-critic. *Neurocomputing*, 71(7-9):1180 – 1190. Progress in Modeling, Theory, and Application of Computational Intelligenc - 15th European Symposium on Artificial Neural Networks 2007, 15th European Symposium on Artificial Neural Networks 2007.

Peters, J. and Schaal, S. (2008b). Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697.

Rasmussen, C. E. and Kuss, M. (2004). Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems 16 (NIPS 2003)*. MIT Press.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

Safonova, A. and Hodgins, J. K. (2005). Analyzing the physical correctness of interpolated human motion. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 171–180, New York, NY, USA. ACM.

Safonova, A., Hodgins, J. K., and Pollard, N. S. (2004). Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In *ACM Transactions on Graphics (TOG), Proceedings of SIGGRAPH*, volume 23, pages 514–521, New York, NY, USA. ACM.

Salakhutdinov, R. and Mnih, A. (2008). Probabilistic matrix factorization. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems 20*, pages 1257–1264. MIT Press, Cambridge, MA.

Schaal, S., Sternad, D., Osu, R., and Kawato, M. (2004). Rhythmic arm movement is not discrete. *Nature Neuroscience*, 7(10):1136–1143.

Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319.

Shon, A. P., Grochow, K., Hertzmann, A., and Rao, R. P. N. (2006). Learning shared latent structure for image synthesis and robotic imitation. In *Advances in Neural Information Processing Systems, NIPS, 18*.

Siciliano, B. and Khatib, O., editors (2008). *Handbook of Robotics*. Springer.

Steffen, J., Klanke, S., Vijayakumar, S., and Ritter, H. J. (2009). Realising dextrous manipulation with structured manifolds using unsupervised kernel regression with structural hints. In *ICRA 2009 Workshop: Approaches to Sensorimotor Learning on Humanoid Robots*, Kobe, Japan.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Tatani, K. and Nakamura, Y. (2003). Dimensionality reduction and reproduction with hierarchical nlpca neural networks - extracting common space of multiple humanoid motion patterns. In Nakamura, Y., editor, *Proc. IEEE International Conference on Robotics and Automation ICRA '03*, volume 2, pages 1927–1932 vol.2.

Tenenbaum, J. B., de Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

Tevatia, G. and Schaal, S. (2000). Inverse kinematics for humanoid robots. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*.

Tipping, M. E. and Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 61(3):611–622.

Torresani, L., Hackney, P., and Bregler, C. (2007). Learning motion style synthesis from perceptual observations. In Schölkopf, B., Platt, J., and Hoffman, T., editors, *Advances in Neural Information Processing Systems 19*, pages 1393–1400. MIT Press, Cambridge, MA.

Udwadia, F. E. and Kalaba, R. E. (1996). *Analytical Dynamics: A New Approach*. Cambridge University Press.

Urtasun, R. and Darrell, T. (2007). Discriminative gaussian process latent variable model for classification. In *International Conference on Machine Learning, ICML*.

Urtasun, R., Fleet, D. J., and Fua, P. (2006). 3d people tracking with gaussian process dynamical models. In *Conference on Computer Vision and Pattern Recognition, CVPR*.

Urtasun, R., Fleet, D. J., Geiger, A., Popovic, J., Darrell, T. J., and Lawrence, N. D. (2008). Topologically-constrained latent variable models. In McCallum, A. and Roweis, S., editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 1080–1087. Omnipress.

Urtasun, R., Glardon, P., Boulic, R., Thalmann, D., and Fua, P. (2004). Style-based motion synthesis. *Computer Graphics Forum*, 23(4):799–812.

van der Maaten, L. J. P., Postma, E. O., and van den Herik, H. J. (2009). Dimensionality reduction: A comparative review. TiCC-TR 2009-005, Tilburg University.

Wang, J. M., Fleet, D. J., and Hertzmann, A. (2007). Multifactor gaussian process models for style-content separation. In *International Conference on Machine Learning, ICML*.

Wang, J. M., Fleet, D. J., and Hertzmann, A. (2008). Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):283–298.

Whitney, D. E. (1969). Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems*, 10(2):47–53.

Wiley, D. J. and Hahn, J. K. (1997). Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Applications*, 17(6):39–45.

Wishart, J. (1928). The generalised product moment distribution in samples from a normal multivariate population. *Biometrika*, 20A(1-2):32–52.

Yamane, K. and Nakamura, Y. (2003). Dynamics filter - concept and implementation of online motion generator for human figures. *IEEE Transactions on Robotics and Automation*, 19(3):421–432.