Gaussian Process Dynamical Models for Human Motion

by

Jack Meng-Chieh Wang

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Gaussian Process Dynamical Models for Human Motion

Jack Meng-Chieh Wang

Master of Science

Graduate Department of Computer Science

University of Toronto

2005

This thesis introduces Gaussian process dynamical models (GPDMs) for nonlinear time series analysis. A GPDM comprises a low-dimensional latent space with associated dynamics, and a map from the latent space to an observation space. We marginalize out the model parameters in closed-form, which leads to modeling both dynamics and observation mappings as Gaussian processes. This results in a nonparametric model for dynamical systems that accounts for uncertainty in the model. We train the model on human motion capture data in which each pose is 62-dimensional, and synthesize new motions by sampling from the posterior distribution. A comparison of forecasting results between different covariance functions and sampling methods is provided, and we demonstrate a simple application of GPDM on filling in missing data. Finally, to account for latent space uncertainty, we explore different priors settings on hyperparameters and show some preliminary GPDM learning results using a Monte Carlo expectation-maximization algorithm.

Acknowledgements

First and foremost, this research would never have happened without the technical contributions from my co-supervisors Aaron Hertzmann and David Fleet. Aaron's work on style-based inverse kinematics piqued my interest in machine learning techniques, and David's course on visual motion analysis is where this work first started. They were responsible for motivating this project, and spent countless hours with me to refine the research into a publishable form. I have learned a great deal from working with both of them.

This thesis is influenced by insightful comments from Allan Jepson, brief discussions with Radford Neal, and talks given by Geoff Hinton. In addition, thanks to Raquel Urtasun for explaining to me the issues in applying this research to human tracking, which became one of the motivations for Chapter 5.

Many thanks to my fellow students at the DGP: Abhishek, Alex, and Eron for co-founding DAG. Anand and Patricio for providing non-stop comic relief. Gonzalo, Jacky, Mike M., Mike P., Nigel, Noah, and Patrick for being great targets to distract without having to stand up. Faisal, Joe, and Mike N. for many actual research conversations. Azeem, Bowen, Irene, Kevin, Mike W., Shahzad, and Winnie for many conversations on subjects ranging from adopting puppies to world domination. Naiqi and Qinxin for many conversations no one else in the lab understands. Dan V., Sam, Tovi, and Tristan for regular food and drink excursions. Anastasia and Marge for keeping the lab sane. Thanks to the DGP research staff, especially John Hancock for year-round technical support.

Last but not least, thanks to all the professors at the DGP, especially Ravin, Aaron, and Karan for having good taste in food.

Contents

1 Introduction **1**

2 Related Work **4**

2.1 Dimensionality reduction 4

2.1.1 Linear methods 5

2.1.2 Geometrically-motivated manifold learning 6

2.1.3 Nonlinear latent variable models 9

2.2 Dynamical systems 13

2.2.1 Linear dynamical systems 15

2.2.2 Nonlinear dynamical systems — state estimation 16

2.2.3 Nonlinear dynamical systems — model estimation 18

2.3 Applications 19

2.3.1 Monocular human tracking 19

2.3.2 Computer animation 20

3 Gaussian Process Dynamics **22**

3.1 Multiple sequences 25

3.2 Higher-order features 26

3.3 In relation to GP regression and GPLVM 27

4 Properties of the GPDM and Algorithms **29**

4.1 Visualizations 30
4.2 Mean prediction sequences 31
4.3 Optimization 32
4.4 Forecasting 33
4.5 Missing data 34
4.6 Multiple subjects 34

5 Accounting for Latent Space Uncertainty **46**
5.1 Priors on hyperparameters 47
5.2 Learning hyperparameters with uncertainty 51
5.2.1 A Monte Carlo EM algorithm for GPDM 53
5.2.2 Preliminary results 55

6 Discussion and extensions **67**

A Prior Distribution of Latent Variables **69**

B Gradients **71**

Bibliography **72**

List of Figures

2.1 Visualization of NLDR on the "Swiss roll" data set, taken from [49]. (a) Illustrating the difference between geodesic distance (solid line) and Euclidean distance (dotted line) in observation space. (b) The recovered embedding using Isomap, red line indicates the approximate geodesic distance. 9

3.1 Time series graphical models. (a) Nonlinear latent-variable model for time series. (Hyperparameters $\bar{\alpha}$ and $\bar{\beta}$ are not shown.) (b) GPDM model. Because the mapping parameters $\mathbf{A}$ and $\mathbf{B}$ have been marginalized over, all latent coordinates $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]^T$ are jointly correlated, as are all poses $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_N]^T$. 23

4.1 Models learned from a walking sequence comprising 2.5 gait cycles. The latent coordinates learned with a GPDM (a) and GPLVM (b) are shown in blue. Vectors depict the temporal sequence. 36

4.2 Visualizations of GPDM learned from a walking sequence. (a) Random trajectories drawn from the model using hybrid Monte Carlo are green. (b) negative log variance for reconstruction shows positions in latent space that are reconstructed with high confidence. 37

4.3 A GPDM of walk data at 120Hz learned with RBF+linear kernel dynamics. The simulation (red) was started far from the training data, and then optimized (green). The poses depicted were reconstructed from latent points along the optimized (simulation) trajectory. 38

4.4 Animation of samples from hybrid Monte Carlo. The trajectory length is 200 frames, four selected frames of each sample are shown here. 39

4.5 Two GPDMs and mean predictions. The first is that from the previous figure. The second was learned with a linear kernel. 40

4.6 GPDM models of walking in a 2D latent space. As above, latent coordinates of data are shown in blue, and mean-prediction simulations are red. (a) GPDM with a linear+RBF kernel. The nonlinear kernel produces a closed limit cycle. (b) GPDM with a linear kernel does not produce a limit cycle for long simulations. 41

4.7 GPDM models of walking in a 2D latent space, with double the number of data points compared to Figure 4.6. (a) GPDM with a linear+RBF kernel. (b) GPDM with a linear kernel. 42

4.8 The GPDM model was learned from three swings of a golf club, using a 2^{nd} order RBF kernel for dynamics. The two plots show 2D orthogonal projections of the 3D latent space. 43

4.9 GPDM from walk sequence with missing data learned with (a) a RBF+linear kernel for dynamics, and (b) a linear kernel for dynamics. Blue curves depict original data. Green curves are the reconstructed, missing data. . . 44

4.10 A GPDM learned with six walkers, one cycle each. (a) Inverse reconstruction variance plot, notice the regions with high reconstruction certainty are separated into small clusters. (b) Simulation results. 45

5.1 Effect of hyperparameter settings in GP regression (a) A conservative hyperparameter setting leads to large uncertainty in prediction. (b) The same data set with a more aggressive hyperparameter setting, which shows overfitting. 58

5.2 Inverse reconstruction variance plot of walk data in 2D with RBF kernel (a) Latent space trajectory given by $M = 1$, notice large discontinuities and variations on reconstruction variance. (b) Latent space trajectory given by $M = 1000$. 59

5.3 Inverse reconstruction variance plot of 120Hz walk data in 2D with RBF kernel (a) Setting $M = 1000$ does not remove discontinuities in data. (b) Latent space trajectory given by $M = 3000$. 60

5.4 Inverse reconstruction variance plot of the first 157 frames of the 120Hz walk data (316 frames in total). (a) Latent space trajectories given by $M = 1$. (b) Latent space trajectory given by $M = 1000$. 61

5.5 Simulations from walker models. (a) Latent space trajectories given by $M = 1$. (b) Latent space trajectory given by $M = 1000$, the simulations follows training data very closely. 62

5.6 Inverse reconstruction variance plot of three golf swing data in 2D with RBF kernel. (a) Latent space trajectories given by $M = 1$. (b) Latent space trajectory given by $M = 1500$. 63

5.7 2D GPDM walking model with RBF dynamics learned using MCEM. (a) Inverse variance plot with a smooth trajectory. (b) Simulation of dynamics, note that the simulation does not lead to a limit cycle. 64

5.8 2D GPDM golf model with RBF dynamics learned using MCEM. (a) Inverse variance plot. (b) Simulation of dynamics. 65

5.9 Samples (green trajectories) from the posterior of a 2D GPDM golf model during the t^{th} iteration of Algorithm 1. The red trajectories represents the posterior mean. 66

Chapter 1

Introduction

Good statistical models for human motion are important for many applications in vision and graphics, namely visual tracking, activity recognition, and computer animation. In computer vision, the highly ambiguous estimation of 3D human pose and motion from monocular video can be improved greatly by incorporating strong prior knowledge of likely states. Specific activities could be classified and recognized by evaluating the likelihood of the observation given the prior model. In computer animation, instead of having animators specify all degrees of freedom in a human-like character, the task of posing characters in animation can be simplified by finding the most likely pose given relatively sparse constraints.

While these models could be designed by hand, it is often impractical for all but the simplest problems. Since many factors interact with each other to generate the observed motion, such as body type, mood, and style, the designer of the model must either account for an enormous amount of complexity or design separate models for many different subjects and motions. For example, a good walking model for a male is unlikely to be a good walking model for a female. Likewise, a good running model is unlikely to be a good jumping model.

The model designing task can be greatly simplified if the designer is only required to

specify a small set of abstract parameters that describe the data generation process (i.e., dependencies between factors), and allow specific parameters to be extracted automatically from data. In other words, we seek to build *generative models* of human motion and acquire estimations of model parameters directly from training data. As described here, the problem can be viewed as one of *machine learning* — creating computer programs by the analysis of data sets.

A useful generative model of motion needs to assign a probability distribution over the space of all possible motions. In particular, higher probabilities should be assigned to motions "closer to" the training data. The current machine learning literature offers a number of generative models for time series data, but each has its own advantages and disadvantages for modeling human motion. Simple models such as hidden Markov models (HMMs) and linear dynamical systems (LDS) are efficient and exact for learning purposes, but are limited in their expressiveness for complex motions. On the other hand, more powerful models such as switching linear dynamical systems (SLDS) and nonlinear dynamical systems (NLDS) require many parameters that need to be hand-tuned.

In this thesis, we investigate a Bayesian approach to learning NLDS, averaging over model parameters rather than estimating them.[1] Inspired by the fact that averaging over nonlinear regression models leads to a Gaussian process (GP) model, we show that integrating over NLDS parameters can also be performed in closed-form. The resulting Gaussian process dynamical model (GPDM) is fully defined by a set of low-dimensional representations of the training data, with both observation and dynamics mappings learned from GP regression. As a natural consequence of GP regression, the GPDM removes the need to select many parameters associated with function approximators while retaining the full power of nonlinear dynamics and observation.

Supervised learning with GP regression has been used to model dynamics for a variety

[1] A version of this work (with David J. Fleet and Aaron Hertzmann) has been accepted for publication in the 2005 Neural Information Processing Systems (NIPS) conference.

of applications [17, 29, 40]. Our approach is most directly inspired by the unsupervised Gaussian process latent variable model (GPLVM) [25, 26], which models the joint distribution of the observed data and their corresponding representation in a low dimensional latent space. This distribution can then be used as a prior for inference from new measurements. However, the GPLVM is not a dynamical model; it assumes that data are generated independently. Accordingly it does not respect temporal continuity of the data, nor does it model the dynamics in the latent space. Here we augment the GPLVM with a latent dynamical model. The result is a Bayesian generalization of subspace dynamical models to nonlinear latent mappings and dynamics.

We show that, given proper initialization, the maximum *a posteriori* (MAP) estimation of latent variable locations with GP dynamics prior results in smoother latent coordinates than learning the latent space and dynamics separately. For motion synthesis, we generate fair samples from the posterior distribution using hybrid Monte Carlo (HMC) [33], and two approximations to the posterior mean: mean-prediction and optimization. A comparison of forecasting results between different covariance functions and sampling methods is provided, and we demonstrate a simple application of GPDM on filling in missing data. Finally, we experiment with priors and sampling methods for hyperparameter estimation to better account for the uncertainty in latent coordinates.

Chapter 2

Related Work

Dimensionality reduction and dynamical model estimation are two essential tools for modeling high-dimensional motion data. The former is often necessary before density estimation methods can be applied to learn a probability model over the data, whereas the latter models the time dependence. Closely-related to both are latent variable models, which often can be used for dimensionality reduction and density estimation at the same time. In this chapter, we review work from both areas along with previous applications of human prior models in vision and graphics.

2.1 Dimensionality reduction

Many typical tasks in statistics and machine learning suffer from the "curse of dimensionality." More specifically, the number of samples required to adequately cover a hypervolume increases exponentially as a function of dimensionality. Performance in various algorithms, both in terms of speed and accuracy, can often by improved by first obtaining a lower-dimensional representation of the data via dimensionality reduction techniques. One important problem that often benefits from dimensionality reduction is density estimation: the problem of estimating an underlying probability density function based on observed data. Solutions addressing the two problems are not disjoint, however, as

dimensionality reduction methods based on latent variable models provide density estimations at the same time. We will use $\mathbb{R}^D$ to denote the observation space and $\mathbb{R}^d$ for the lower dimensional space, where $D > d$. In the most general formulation, let $g : \mathbb{R}^d \to \mathbb{R}^D$, be the data generation process

$$\mathbf{y} = g(\mathbf{x}; \mathbf{B}) + \mathbf{n}, \tag{2.1}$$

where $\mathbf{y} \in \mathbb{R}^D$, $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{n}$ is a random noise vector, $\mathbf{B}$ are parameters of g.

2.1.1 Linear methods

A natural way of approaching the dimensionality reduction problem is to represent the data in a linear subspace of the original observation space. The main question is then selecting an appropriate basis for the subspace. Principal component analysis (PCA) is perhaps the most well-known linear dimensionality reduction technique. It finds a basis for the projected subspace such that the variance of the projected data is maximized. However, the basic form of PCA does not provide any probabilistic information about the data. Probabilistic PCA (PPCA), introduced by Roweis [42] and Tipping and Bishop [51], shows finding the basis by maximizing variance is equivalent to maximum-likelihood (ML) estimation of parameters. PPCA is a latent variable model which assumes the observed data ($\mathbf{y}$) is generated by lower-dimensional data ($\mathbf{x}$). The prior on $\mathbf{x}$ is assumed to be a spherical Gaussian centred around the origin.

PPCA is a special case of a more general class of latent variable models known as factor analyzers, where $\mathbf{y}$ is related to $\mathbf{x}$ via a linear mapping with Gaussian process noise:

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b} + \mathbf{n}, \tag{2.2}$$

where $\mathbf{x} \sim \mathcal{N}(0; \mathbf{R})$ and $\mathbf{n} \sim \mathcal{N}(0; \mathbf{Q})$. In PPCA, $\mathbf{R} = \mathbf{I}$ and $\mathbf{Q} = \sigma^2\mathbf{I}$.

Linear methods are generally easy to implement and are very efficient, but are not ideal for many data sets. The "Swiss roll" data set (Figure 2.1) is a popular test example for nonlinear dimensional reduction algorithms, where we can easily identify a 2D

coordinate frame by unrolling the data on a plane. A linear map which only allows for rotation, scaling, and translation is clearly insufficient to map data points on a plane back to a Swiss roll. Typically, when restricted to linear mappings, a subspace with larger number of dimensions than the intrinsic dimensionality of the data is needed to adequately represent variations in the data.

From a density estimation point of view, another problem is the assumption of Gaussian prior distributions. Consider a walking data set, which usually maps to a curve that circumnavigates the origin in latent space under PPCA. The probability model assumes that the latent coordinates are samples from a Gaussian distribution centred at the origin, consequently points near the origin are assigned higher probability. Such a model of walking assigns the mean pose (mapped to the origin in latent space), which is never observed in the data set, the highest probability.

2.1.2 Geometrically-motivated manifold learning

In applications such as data visualization, it is often sufficient to recover an embedding of the data without a mapping back into observation space. Recent advances in nonlinear dimensional reduction (NLDR) model the structure of the data generating manifold without providing mappings between observation and embedding spaces. Depending on the way geometric information is used, these algorithms can be broadly divided into two groups: local and global. Local techniques preserve geometric structure around neighbourhoods of samples, without global information. On the other hand, global techniques attempt to preserve geometry on all scales [9].

Locally linear embedding (LLE) [41] is a well-known local NLDR technique that is geometrically-motivated. The crucial observation is that all smooth manifolds are locally linear with respect to sufficiently small neighbourhoods on the manifold. In particular, given a well-sampled manifold $\mathcal{M} \subset \mathbb{R}^n$, and let $\mathbf{y}_i$ denote samples from $\mathcal{M}$, the local linearity assumption implies that there exists $w_{ij} \in \mathbb{R}$ such that $\mathbf{y}_i = \sum_j w_{ij}\mathbf{y}_j$, where

$\mathbf{y}_j$ are samples in the locally linear neighbourhood of $\mathbf{y}_i$. The total reconstruction error of a given set of weights $\mathbf{w} = \{w_{ij}\}$ is then given by

$$\epsilon(\mathbf{w}) = \sum_i \|\mathbf{y_i} - \sum_j w_{ij}\mathbf{y}_j\|^2 \tag{2.3}$$

which is then minimized in closed-form with respect to $\mathbf{w}$, subject to the constraints that $w_{ij} = 0$ for $\mathbf{y}_j$ outside the neighbourhood of $\mathbf{y}_i$, and that $\sum_j w_{ij} = 1$.

To compute $\mathbf{x}_i$, the embedding of $\mathbf{y}_i$ in a lower dimensional subspace, it is assumed that the optimal reconstruction weights in observation space are preserved. In other words, $\mathbf{x}_i = \sum_j w_{ij}\mathbf{x}_j$, for w_{ij} minimizing (2.3). The assumption effectively says that the mapping from high-dimensional observation space to the low-dimensional embedding space that we seek is locally linear, which is true for any continuous mapping with a sufficiently small local neighbourhood. Consequently, the embedding cost function is defined by

$$\phi(\mathbf{x}) = \sum_i \|\mathbf{x_i} - \sum_j w_{ij}\mathbf{x}_j\|^2, \tag{2.4}$$

where $\mathbf{x} = \{x_i\}$. The above optimization problem can be formulated as a sparse eigenvalue problem and solved in closed-form [41].

Another local technique similar in spirit to LLE is the Laplacian eigenmap algorithm [2]. Consider a function $\psi : \mathcal{M} \to \mathbb{R}$, if $\|\nabla\psi\|$ is small, points near each other in $\mathcal{M}$ will be mapped to points near each other in $\mathbb{R}$. In particular, finding ψ such that $\int_{\mathcal{M}} \|\nabla\psi(\mathbf{x})\|^2$ is minimized produces an embedding that best preserves the average locality. The minimization problem has been shown to be equivalent to finding eigenfunctions of the Laplace-Beltrami operator, defined on differentiable functions on a manifold.

The Laplace-Beltrami operator for functions on continuous manifolds have a parallel in graph theory called the Laplacian matrix. The Laplacian eigenmap algorithm constructs a graph of data points in observation space, using weighted edges to encode proximity information. The graph serves as a discrete approximation of a continuous manifold. The embedding is obtained by solving for the eigenvectors of the Laplacian

matrix, which has the property of preserving locality as indicated by the continuous case.

The Isomap algorithm [49] and its variants C-Isomap, L-Isomap [9], and ST-Isomap [21] are global techniques which extend multidimensional scaling (MDS). In MDS, a measure of "dissimilarity" between pairs of data is given in matrix form, and the goal is to recover a lower dimensional representation of the data such that the separations between pairs of projected data in latent space is related to their dissimilarity in observation space. In classical MDS, which can be shown to be equivalent to PCA, the separations in latent space is set to be equal to the dissimilarities. By setting different measurements for dissimilarity, embeddings that reflect different properties of the data can be recovered. Particularly, the measurement of dissimilarity can be set to account for the structure of a underlying manifold that generated the data.

Geodesic distance on the manifold is used as the measurement of dissimilarity in Isomap. The distance between points on the recovered embedding reflects their separation on the manifold, not the distance in observation space (See Figure 2.1). The approximate geodesic distance is obtained by first constructing a weighted graph, using edge weights to indicate distances between points in observation space, then computing the shortest path distances between all pairs of points. The final embedding is recovered by applying classical MDS to the dissimilarity matrix formed by the shortest path distances.

Geometrically-motivated NLDR methods discussed in this section never explicitly model the function g or g^{-1} (2.1) in the process of finding an embedding for the data. Consequently, there are no simple methods of mapping data outside of the training set to the lower-dimensional space or back. One solution is to use nonlinear regression algorithms such as neural networks, RBF networks, or Gaussian processes to learn g as a post-process. However, modeling g^{-1} using nonlinear regression is more difficult, as the domain of g^{-1} typically has a much higher dimensionality. Alternatively, geometrically-motivated generalizations to new data have recently been investigated [3], and are capable of recovering embeddings of new data in observation space without explicitly modeling

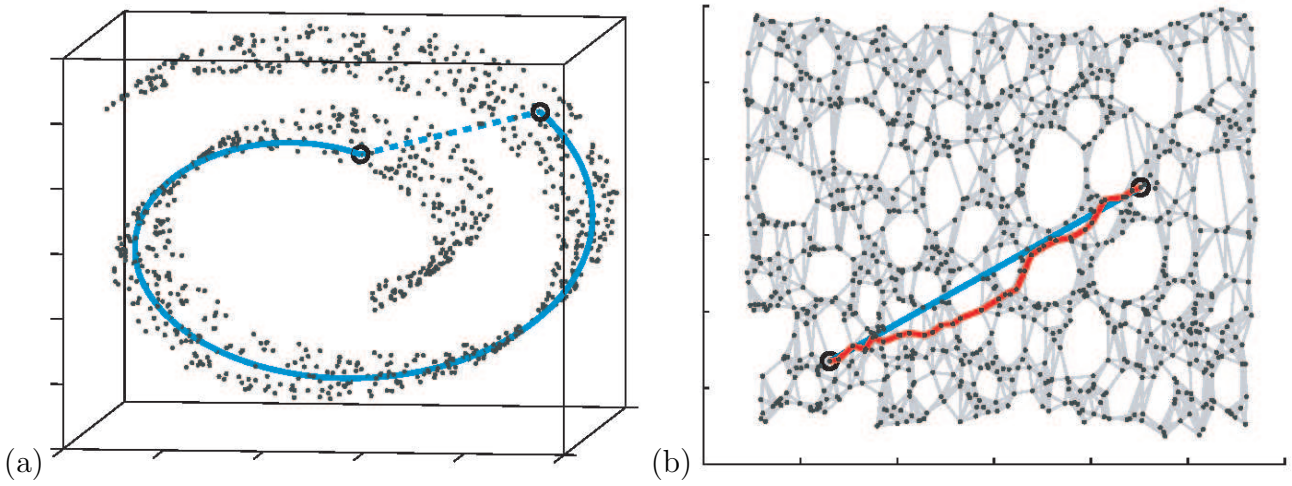


Figure 2.1: Visualization of NLDR on the "Swiss roll" data set, taken from [49]. (a) Illustrating the difference between geodesic distance (solid line) and Euclidean distance (dotted line) in observation space. (b) The recovered embedding using Isomap, red line indicates the approximate geodesic distance.

g^{-1}.

While NLDR methods can be augmented with mappings, they do not provide a probability distribution over data. Typical density estimation techniques such as mixtures-of-Gaussians (MoG) can be used to learn a probability model [19] in the lower dimensional space, which can be used as a probability model for data as long as new data in observation space can be mapped to the lower dimensional space. However, as observed in [18] with human pose data, MoG density estimation is prone to overfitting and requires tuning a large number of parameters in practice. For LLE and Isomap, an additional problem is that they assume the observed data is generated from a densely sampled manifold, which is typically not true for human motion data.

2.1.3 Nonlinear latent variable models

Nonlinear latent variable models (NLVM) are latent variable models that remove the linearity assumption on g, and are capable of modeling data generated from a nonlinear

manifold. NLVM methods treat the embedding and mapping as parameters in a generative model. The parameters are typically estimated using optimization or Monte Carlo simulation when needed. In this section, we discuss three representative algorithms: density networks [32], generative topographic mappings (GTM) [4] , and Gaussian process latent variable models (GPLVM) [25, 26].

The density network model [32] is an extension of Bayesian supervised neural networks to the unsupervised problem of density estimation. The function $g(\mathbf{x};\mathbf{B})$ is modeled using a neural network with $\mathbf{B}$ as weights in [32], but the model allows for general parametric function approximators. The distribution $p(\mathbf{y}_i|\mathbf{x}_i,\mathbf{B})$ is defined by the noise model in observation space, which is typically Gaussian. The probability of all observed data $\mathbf{Y} = \{\mathbf{y}_i\}$ is given by

$$p(\mathbf{Y}|\mathbf{B}) = \prod_{i=1}^{N} p(\mathbf{y}_i|\mathbf{B}), \tag{2.5}$$

where the probability of $\mathbf{y}_i$ is

$$p(\mathbf{y}_i|\mathbf{B}) = \int p(\mathbf{y}_i|\mathbf{x}_i,\mathbf{B})p(\mathbf{x}_i)d\mathbf{x}_i. \tag{2.6}$$

For the typical spherical Gaussian prior on $\mathbf{x}$, the integral over $\mathbf{x}$ can be evaluated by Monte Carlo sampling. The value of $\mathbf{B}$ is estimated by performing gradient based minimization of the negative log of (2.5).

Once $\mathbf{B}$ is learned, inference can be performed on the posterior distribution of $\mathbf{x}_i$

$$p(\mathbf{x}_i|\mathbf{y}_i,\mathbf{B}) = \frac{p(\mathbf{y}_i|\mathbf{x}_i,\mathbf{B})p(\mathbf{x}_i)}{p(\mathbf{y}_i|\mathbf{B})}. \tag{2.7}$$

Specific values of $\mathbf{x}_i$ can be estimated by averaging Monte Carlo simulation results on the posterior (posterior mean), which is useful for visualization [32]. Monte Carlo simulations can also be used to compute (2.6), which gives a density estimate in observation space given a particular set of mapping parameters $\mathbf{B}$.

While the density network model is very general, it is also computationally expensive. Moreover, the optimization of (2.5) with respect to $\mathbf{B}$ is nondeterministic due to the use of Monte Carlo integration.

The GTM algorithm [4] can be viewed as a special instance of the density network framework, with a regularly sampled discrete prior on $\mathbf{x}_i$ and $g(\mathbf{x}_i; \mathbf{B})$ approximated by a RBF network. Particularly, the prior on $\mathbf{x}_i$ along with a Gaussian noise model induce a mixture of Gaussians density in observation space:

$$p(\mathbf{x}_i) = \frac{1}{K} \sum_{j=1}^{K} \delta(\mathbf{x}_i - \mathbf{c}_j) \tag{2.8}$$

$$p(\mathbf{y}_i|\mathbf{B}) = \frac{1}{K} \sum_{j=1}^{K} p(\mathbf{y}_i|\mathbf{x}_i = \mathbf{c}_j, \mathbf{B}), \tag{2.9}$$

where the $\mathbf{c}_j$ terms are regularly sampled, fixed locations in latent space.

The estimation of model parameters in a mixture model (2.9) is a well-known problem in machine learning, and can be solved efficiently using the expectation-maximization (EM) algorithm [10]. In the context of mixture models, the posterior probabilities, or responsibilities, of each Gaussian component j can be evaluated by applying Bayes' theorem in the E-step

$$p(\mathbf{x}_i = \mathbf{c}_j|\mathbf{y}_i, \mathbf{B}) = \frac{p(\mathbf{y}_i|\mathbf{x}_i = \mathbf{c}_j, \mathbf{B})}{\sum_{k=1}^{K} p(\mathbf{y}_i|\mathbf{x}_i = \mathbf{c}_k, \mathbf{B})}, \tag{2.10}$$

where $\mathbf{B}$ is assumed to be fixed from the previous iteration. In the M-step, solving for $\mathbf{B}$ reduces to solving for a set of linear equations. The details of the GTM learning algorithm can be found in [4].

The posterior mean can be used if we wish to find a single $\mathbf{x}_i$ to represent $\mathbf{y}_i$. Unlike in the density networks model, Monte Carlo simulation is not necessary. Taking advantage of the regularly sampled prior on $\mathbf{x}_i$, we get

$$< \mathbf{x}_i >_{p(\mathbf{x}_i|\mathbf{y}_i, \mathbf{B})} = \int p(\mathbf{x}_i|\mathbf{y}_i, \mathbf{B}) \mathbf{x}_i d\mathbf{x}_i = \sum_{j=1}^{K} p(\mathbf{x}_j = \mathbf{c}_j|\mathbf{y}_i, \mathbf{B}) \mathbf{c}_j. \tag{2.11}$$

Both density networks and GTMs provide a density estimate over observation space and explicitly models $g : \mathbb{R}^d \to \mathbb{R}^N$. Model parameter estimation in density networks require Monte Carlo techniques for integrating over hidden variables, which is inefficient

and might be impractical depending on the dimensionality of the latent space. The GTM has a nice property that model parameters can be estimated with the EM algorithm, but regular sampling in latent space creates undesirable discrete visualization artefacts [25].

The GPLVM [25, 26] takes a different approach to density estimation by marginalizing over parameters instead of latent variables. If we assume g takes the form of a linear combination of basis functions

$$g(\mathbf{x}_i; \mathbf{B}) \;=\; \sum_j \mathbf{b}_j \; \psi_j(\mathbf{x}_i), \tag{2.12}$$

where $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, ...]$. We can see that while g is nonlinear in $\mathbf{x}_i$, it is linear in $\mathbf{B}$. Even though we cannot analytically evaluate (2.6) by marginalizing over $\mathbf{x}_i$, we can analytically marginalize over $\mathbf{B}$ with an isotropic prior on the columns of $\mathbf{B}$. Suppose we use RBFs for ψ, and take the number of basis functions to infinity, the marginalization results in

$$p(\mathbf{Y}|\mathbf{X}, \bar{\beta}) \;=\; \frac{1}{\sqrt{(2\pi)^{ND}|\mathbf{K}|^D}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^T\right)\right) \;, \tag{2.13}$$

where $\mathbf{K}$ is a kernel matrix with hyperparameters $\bar{\beta} = \{\beta_1, \beta_2, \beta_3\}$. The elements of the kernel matrix are defined by a RBF kernel function, $(\mathbf{K})_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$.

$$k(\mathbf{x}, \mathbf{x}') \;=\; \beta_1 \exp\left(-\frac{\beta_2}{2}||\mathbf{x} - \mathbf{x}'||^2\right) + \beta_3^{-1}\delta_{\mathbf{x},\mathbf{x}'}. \tag{2.14}$$

See Chapter 3 for more on the RBF kernel hyperparameters. It should be noted that the RBF kernel function only represents a subset of all possible GPLVMs, any function that results in a nonnegative definite kernel matrix for all possible $\mathbf{X}$ can be used as the kernel function [33]. Intuitively, different kernel functions correspond to different representations of g, and their selection should depend on the application.

The GPLVM estimates the joint density of the data points ($\mathbf{Y}$) and their latent space representations ($\mathbf{X}$). The MAP estimates of $\mathbf{X}$ are used to represent a learned model, and can be directly used for data visualization [25, 26]. GPLVM has the attractive property of generalizing reasonably well from small data sets in high dimensional observation spaces [18], and a fast learning algorithm based on active sets is available. A drawback of this

approach is that the likelihood of a new data point given a learned model cannot be evaluated in closed-form, as a latent space position for the new data point is needed to evaluate (2.13).

Except for ST-Isomap, neither geometrically-motivated NLDR nor nonlinear latent variable methods discussed in this section are designed to model data with temporal coherence. For applications such as activity manifold learning [12], the training data are typically video and motion capture sequences, where temporal information is significant. For example, if we try to learn a manifold for a particular person walking using several walk trajectories, the sampling rate along each trajectory will typically be much greater than between trajectories. The result is a sparsely and nonuniformly sampled manifold, which creates problems for manifold learning algorithms. Incorporating temporal information also allows more information to be extracted from the data; Namely, we could compute the likelihood of motion trajectories, and perform forecasting: calculating the probability distribution over the next pose given the last few poses.

2.2 Dynamical systems

The analysis of time series data has been studied extensively in various fields such as control engineering, finance, and economics. We are particularly interested in modeling processes that are not directly observable, corresponding to the graphical model in Figure 3.1(a). The $\mathbf{x}_t$ terms represent hidden states of the system at time t, while $\mathbf{y}_t$ terms represent the observed outputs of the system at time t. We also assume that there exists a dynamical process that governs the evolution of $\mathbf{x}_t$ with respect to time, and a separate observation process that maps $\mathbf{x}_t$ to $\mathbf{y}_t$. The former is usually assumed to be a Markov process. In engineering literatures, parameter estimation in such models is referred to as system identification. Note that dynamical systems can also have input signals $\mathbf{u}_t$ at each time step, which is useful for modeling control systems. We focus on the fully

unsupervised case in this thesis, where there are no inputs to the system.

Although the observations could be simply a noisy version of the hidden states, the main motivation of analyzing the dynamical process in a hidden state space instead of the observation space is the dimensionality of typical observations. Consider a video of a rotating cube as an example, the observation space could be a 512 by 512 image, while the rotation itself could be described by a quaternion. Suppose we are interested in predicting the next frame of the video given the current one, learning a mapping between consecutive frames directly means solving a 262144-dimensional regression problem. On the other hand, if we could estimate the quaternion which describes the rotation from the image, we could simply predict the next quaternion from the current one, which is a four dimensional regression problem. The idea also applies to analysing human motion capture data, where the observation space consist of all the joint angles in a skeleton.

There are three distributions of interest for our applications in this type of dynamical systems. Let $\mathbf{x}_{1:t} = \{\mathbf{x}_1, \cdots, \mathbf{x}_t\}$. The Markov assumption states,

$$p(\mathbf{x}_t|\mathbf{x}_{1:t-1}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{2.15}$$

which is given by the stochastic dynamical process. This distribution predicts the next state of the system given the current one. It is entirely dependent on the noise distribution for the dynamical process, which is typically Gaussian.

The second distribution of interest is $p(\mathbf{y}_t|\mathbf{x}_t)$, which is given by the observation process. This is also typically Gaussian. Moreover, since the $\mathbf{y}_t$ terms are independent given the $\mathbf{x}_t$ terms, we have

$$p(\mathbf{y}_{1:t}|\mathbf{x}_{1:t}) = \prod_{i=1}^{t} p(\mathbf{y}_i|\mathbf{x}_i), \tag{2.16}$$

where $\mathbf{y}_{1:t} = \{\mathbf{y}_1, \cdots, \mathbf{y}_t\}$. We could think of this distribution as the result of density estimation in observation space in a latent variable model, such as the GPLVM.

Finally, we are interested in $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ in order to estimate the hidden state at time t. In a Markov process, this distribution can be propagated through time in a recur-

sive manner. Using Bayes' rule, the first-order Markov assumption, and the conditional independence of observations, it can be shown that

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = kp(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1}), \tag{2.17}$$

where

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})d\mathbf{x}_{t-1}. \tag{2.18}$$

For a detailed discussion, see [14]. Note that for certain applications, it is feasible to estimate the value of $\mathbf{x}_t$ using information from both the past and the future. The operations are called filtering and smoothing respectively, and Equations (2.17) and (2.18) are referred to as the filtering equations.

Suppose both the dynamical and observation processes (model parameters) are known, then the first two distributions are defined. We then need obtain a state estimation given an observation sequence. On the other hand, if the hidden trajectory corresponding to an observation sequence is known (i.e., the third distribution is known with certainty), we have a function fitting problem. In the general case, both model parameters and hidden state trajectories are unknown.

2.2.1 Linear dynamical systems

The simplest, and most studied type of dynamical system is the linear dynamical system (LDS). Various solutions for LDS can be found in both the engineering and statistics literature. Mathematically, a LDS can be expressed by

$$\mathbf{x}_t = \mathbf{M}\mathbf{x}_{t-1} + \mathbf{n}_{x,t} \tag{2.19}$$

$$\mathbf{y}_t = \mathbf{N}\mathbf{x}_t + \mathbf{n}_{y,t}, \tag{2.20}$$

where $\mathbf{M}$ and $\mathbf{N}$ are matrix operations corresponding to linear dynamical and observation mappings respectively, $\mathbf{n}_{x,t}$ and $\mathbf{n}_{y,t}$ are zero mean, general Gaussian noise processes.

Using a combination of Kalman smoothing and gradient based optimization, the problem can be formulated as maximum-likelihood learning (2.21) [11], and solved using the EM algorithm [44, 15]. Closed-form solutions to the problem do exist in the system identification theory literature. The N4SID algorithm, introduced by Van Overschee and De Moor [54] is available as a Matlab toolbox.

The key observation for linear systems is that the linear transformation of a Gaussian is still Gaussian. If $p(\mathbf{x}_1|\mathbf{y}_1)$ is Gaussian, it can be shown that the filtering equations are also Gaussian for all t. The distributions can then be fully represented by first- and second-order statistics, which can be estimated using the Kalman smoothing algorithm. These distributions are used to compute the expected log likelihood required in the E-step of the algorithm. The linear mappings allow the likelihood function to be integrated analytically. Model parameters $\mathbf{M}$, $\mathbf{N}$, along with covariance matrices of the noise processes can then be found by optimization in the M-step.

While computations in LDS are efficient and are relatively easy to analyze, the model is not suitable for a large class of problems. By definition, nonlinear variations in the state space are treated as noise in an LDS model, resulting in overly smoothed motion during simulation. The linear observation function suffers from the same shortcomings as linear latent variable models, which are discussed in Section 2.1.1.

2.2.2 Nonlinear dynamical systems — state estimation

A natural way of increasing the expressiveness of the model is to remove the linearity assumption on the mappings, but learning such general nonlinear models is difficult. The main difficulty arises from the fact that the filtering equations are now generally non-Gaussian. So even if we assume the model parameters are known, the state estimation problem alone is nontrivial. Various methods have been proposed for this type of state estimation, most of which centers around approximating the filtering distribution with a Gaussian.

Extended Kalman filtering (EKF) approximates the nonlinear dynamics and observations mapping with locally linear mappings around the current state. The effect is that filtering equations in this linearized system are Gaussian, assuming the initial state distribution is Gaussian. EKF performs well for systems with smooth mappings that are local linear. However, for motions with sudden acceleration, such as the middle of a golf or baseball swing, local linearization could produce highly inaccurate estimation. Moreover, local linearization requires computing the Jacobian of the mapping, which can be expensive in practice.

The unscented Kalman filter (UKF) [22] is an alternative method that attempts to approximate the arbitrary state distribution with a Gaussian, but makes no simplifying assumptions about the nonlinear mapping. A deterministically-chosen set of points (so that their mean and covariance is equal to the current state approximation) is sent through a nonlinear mapping. The mean and covariance of the outputs, are then used to approximate the output distribution as a Gaussian. While [22] reported improved estimation results compared to EKF, it does not address problems where the true state distribution is highly non-Gaussian, or multi-modal.

A number of Monte Carlo sampling methods have also been proposed for the state estimation problem [7, 31, 20, 53, 8]. The basic idea is to represent the filtering distribution $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$ at any given time with a number of weighted discrete samples. At each time step, the samples are propagated through the dynamics process $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and are reweighed through the observation likelihood $p(\mathbf{y}_t|\mathbf{x}_t)$, which then becomes a weighted discrete representation of $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. These methods are theoretically capable of representing arbitrary state distributions, but are highly sensitive to dimensionality and are expensive to compute.

2.2.3 Nonlinear dynamical systems — model estimation

Complementary to state estimation is the model estimation problem, which defines the distributions in (2.15) and (2.16). Suppose we have point estimates of the $\mathbf{x}_t$ terms (i.e., assume the hidden states are fully observed), then the problem reduces to nonlinear regression. Methods such as artificial neural networks (ANN), radial basis function networks (RBFN), or Gaussian processes (GP) can then be used to learn both the dynamics mapping and the observation mapping.

However, ideal estimates of the posterior distribution should not be point estimates. Since the state values are never observed, our uncertainty about them should be reflected during estimation. As discussed previously, results from EKF and UKF are Gaussians, whereas Monte Carlo methods give discrete representations of arbitrary distributions. Fitting functions to uncertain inputs (and outputs of the dynamics function) can no longer be viewed as a simple regression problem.

Another way of viewing model estimation is as to find a set of parameters θ such that the likelihood $p(\mathbf{y}_{1:t}|\theta)$ is maximized. Using point estimates of $\mathbf{x}_{1:t}$ for nonlinear regression is equivalent to maximizing the *complete likelihood* $p(\mathbf{y}_{1:t}, \mathbf{x}_{1:t}|\theta)$. On the other hand, keeping the uncertainty in $\mathbf{x}_{1:t}$, we should maximize the *incomplete likelihood*

$$p(\mathbf{y}_{1:t}|\theta) = \int_{\mathbf{x}_{1:t}} p(\mathbf{y}_{1:t}, \mathbf{x}_{1:t}|\theta) d\mathbf{x}_{1:t}, \tag{2.21}$$

integrating out $\mathbf{x}_{1:t}$.

The above integral is generally intractable, and suggests that model estimation should not be treated separately from state estimation when the states are truly hidden. While Monte Carlo approximations can be used, they are expensive and induce a difficult stochastic optimization problem. The EM algorithm maximizes a lower bound of the incomplete likelihood, and can be applied in a straightforward fashion to linear systems as discussed previously. We discuss the EM algorithm in more detail in Chapter 5.

Ghahramani and Roweis [16] showed that nonlinear systems can also be learned using

the EM algorithm, by combining EKF and RBF regression. Alternatively, linear dynamical systems can be augmented with switching states to approximate a nonlinear system, which has been applied to model human motion [38]. Both approaches require sufficient amounts of training data that one can learn the parameters of the switching or basis functions. Determining the appropriate number of basis functions is also difficult.

To summarize, a central difficulty in modeling time series data is in determining a model that can capture the nonlinearities of the data without overfitting. Linear autoregressive models require relatively few parameters and allow closed-form analysis, but can only model a limited range of systems. In contrast, existing nonlinear models can model complex dynamics, but may require many training data points to accurately learn MAP models.

2.3 Applications

Our work is motivated by modeling human motion for video-based people tracking and data-driven animation. An individual human pose is typically parameterized with more than 60 parameters. Despite the large state space, the space of activity-specific human poses and motions has a much smaller intrinsic dimensionality; in our experiments with walking and golf swings, 3 dimensions often suffice. Bayesian people tracking requires dynamical models in the form of transition densities in order to specify prediction distributions over new poses at each time instant (e.g., [38, 45, 37, 46, 12, 52]); similarly, data-driven computer animation requires prior distributions over poses and motion (e.g., [5, 24, 39, 30, 27, 18]).

2.3.1 Monocular human tracking

Despite the difficulties with linear subspace models mentioned above, PCA has been applied to visual tracking of humans and other vision applications [55, 37, 45, 52]. The

typical data representation in this approach is to concatenate the entire trajectory of a walk cycle or a golf swing as one vector in the data space. The lower dimensional PCA space is then used as the state space. Nonlinearity of the data distribution is no longer a problem here, since the training data lie in trajectory space instead of pose space. In addition, there is no need to learn dynamics from the locations of data points. The introduction of a phase parameter which propagates forward in time can serve as an index to the prior distribution of poses. While the results of tracking are reasonable, this prior model is highly specific to the activity being captured. It is not clear how it can be extended to transition between multiple activities due to the explicit dependence on phase.

Nonlinear dimensionality reduction techniques such as LLE have also been used in the context of human pose analysis. Elgammal and Lee [12] use LLE to learn activity based manifolds from silhouette data. They then use nonlinear regression methods to learn mappings from manifolds back to silhouette space and to 3D data. Jenkins and Matarić [21] use ST-Isomap to learn embeddings of multi-activity human motion data and robot teleoperation data. Sminchisescu and Jepson [46] used spectral embedding techniques to learn an embedding of 3D motion capture data. They also learn a mapping back to pose space separately, and a large amount of training data is needed. None of the above approaches learn a dynamics function explicitly, and no density model in the embedding space is learned in [21]. In general, the need to learn an embedding, mappings, and a density function separately is undesirable.

2.3.2 Computer animation

The applications of prior models for animation include style-content separation [5], motion synthesis subject to sparse user constraints [24, 39, 30, 18], and interactive avatar control [27].

Brand and Hertzmann [5] model human motion by augmenting an HMM with stylis-

tic parameters. They applied the model to the problem of style-content separation in human motion and were able to achieve effects such as replacing poorly performed dance sequences from a novice actor by ones performed by an expert actor. However, the discrete latent space of an HMM is not suited to model continuous human motion, and artefacts such as feet sliding are often observed in the results.

A number of authors have constructed prior models by combining large motion databases with simple probabilistic models based on distance between poses in both position and velocity [24, 39, 30, 27]. This approach is designed for real time control and synthesis of animation, but a large amount of data is needed and the ability to generalize is minimal. Although the quality of synthesis is relatively high, the model is only capable of synthesizing poses in or near poses the database.

The Gaussian process latent variable model (GPLVM), recently developed by Lawrence [25, 26], learns a lower dimensional representation of the data as well as a probabilistic mapping from latent space back to data space. A scaled version of GPLVM (SGPLVM) was applied to human pose data in computer animation by Grochow et al. [18], where the density function given by the probabilistic mapping was used to determine the most likely pose given kinematics constraints. The function prefers poses close to the training data and falls off smoothly as distance increases. However, the SGPLVM did not explicitly model dynamics in the pose data.

On the other hand, the GPLVM initializes an embedding by PCA. A likelihood of the data is then defined by GP regression, treating PCA coordinates as inputs and pose as outputs. As a result, both a mapping to pose space and a density function over latent space is obtained. The input coordinates can be moved to locations given by maximum likelihood estimation. We introduce an additional prior on the locations in latent space with dynamics.

Chapter 3

Gaussian Process Dynamics

The Gaussian process dynamical model (GPDM) comprises a mapping from a latent space to the data space, and a dynamical model in the latent space (Figure 1). These mappings are typically nonlinear. The GPDM is obtained by marginalizing out the parameters of the two mappings, and optimizing the latent coordinates of training data.

More precisely, our goal is to model the probability density of a sequence of vector-valued states $\mathbf{y}_1 ..., \mathbf{y}_t, ..., \mathbf{y}_N$, with discrete-time index t and $\mathbf{y}_t \in \mathbb{R}^D$. As a basic model, consider a latent-variable mapping with first-order Markov dynamics:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}; \mathbf{A}) + \mathbf{n}_{x,t} \tag{3.1}$$

$$\mathbf{y}_t = g(\mathbf{x}_t; \mathbf{B}) + \mathbf{n}_{y,t} \tag{3.2}$$

Here, $\mathbf{x}_t \in \mathbb{R}^d$ denotes the d-dimensional latent coordinates at time t, $\mathbf{n}_{x,t}$ and $\mathbf{n}_{y,t}$ are zero-mean, white Gaussian noise processes, f and g are (nonlinear) mappings parameterized by $\mathbf{A}$ and $\mathbf{B}$, respectively. Figure 3.1(a) depicts the graphical model.

While linear mappings have been used extensively in autoregressive models, here we consider the nonlinear case for which f and g are linear combinations of basis functions:

$$f(\mathbf{x}; \mathbf{A}) = \sum_i \mathbf{a}_i \, \phi_i(\mathbf{x}) \tag{3.3}$$

$$g(\mathbf{x}; \mathbf{B}) = \sum_j \mathbf{b}_j \, \psi_j(\mathbf{x}) \tag{3.4}$$

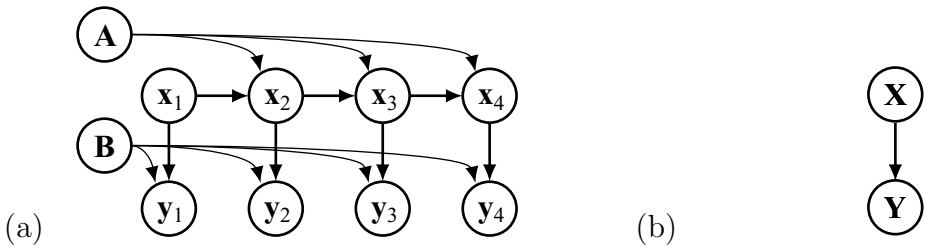


Figure 3.1: Time series graphical models. (a) Nonlinear latent-variable model for time series. (Hyperparameters $\bar{\alpha}$ and $\bar{\beta}$ are not shown.) (b) GPDM model. Because the mapping parameters $\mathbf{A}$ and $\mathbf{B}$ have been marginalized over, all latent coordinates $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]^T$ are jointly correlated, as are all poses $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_N]^T$.

for weights $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, ...]$ and $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, ...]$, and basis functions ϕ_i and ψ_j. In order to fit the parameters of this model to training data, one must select an appropriate number of basis functions, and one must ensure that there is enough data to constrain the shape of each basis function. Ensuring both of these conditions can be very difficult in practice.

However, from a Bayesian perspective, the specific forms of f and g — including the numbers of basis functions — are incidental, and should therefore be marginalized out. With an isotropic Gaussian prior on the columns of $\mathbf{B}$, marginalizing over g can be done in closed form [33, 35] to yield

$$p(\mathbf{Y} \,|\, \mathbf{X}, \bar{\beta}) \;=\; \frac{|\mathbf{W}|^N}{\sqrt{(2\pi)^{ND}|\mathbf{K}_Y|^D}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{K}_Y^{-1}\mathbf{Y}\mathbf{W}^2\mathbf{Y}^T\right)\right) \;, \tag{3.5}$$

where $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_N]^T$, $\mathbf{K}_Y$ is a kernel matrix, and $\bar{\beta} = \{\beta_1, \beta_2, ..., \mathbf{W}\}$ comprises the kernel hyperparameters. The elements of kernel matrix are defined by a kernel function, $(\mathbf{K}_Y)_{i,j} = k_Y(\mathbf{x}_i, \mathbf{x}_j)$. For the latent mapping, $\mathbf{X} \rightarrow \mathbf{Y}$, we currently use the RBF kernel

$$k_Y(\mathbf{x}, \mathbf{x}') \;=\; \beta_1 \exp\left(-\frac{\beta_2}{2}||\mathbf{x} - \mathbf{x}'||^2\right) + \beta_3^{-1}\delta_{\mathbf{x},\mathbf{x}'} \;. \tag{3.6}$$

As in the SGPLVM [18], we use a scaling matrix $\mathbf{W} \equiv \mathrm{diag}(w_1, ..., w_D)$ to account for differing variances in the different data dimensions. This is equivalent to a GP with kernel function $k(\mathbf{x}, \mathbf{x}')/w_m^2$ for dimension m, or a warped GP [47] with warping $\mathbf{YW}$.

The form of the kernel function, which represents entries in the covariance matrix of $p(\mathbf{Y} \mid \mathbf{X}, \bar{\beta})$, is closely related to the function approximator used for (3.4). In this case, the RBF kernel corresponds to using an infinite number of RBFs to represent f [33]. Hyperparameter β_1 represents the overall scale of the output function, while β_2 corresponds to the inverse width of the RBFs. The variance of the process noise term $\mathbf{n}_{y,t}$ is given by β_3^{-1}.

The dynamic mapping on the latent coordinates $\mathbf{X}$ is conceptually similar, but more subtle. Conceptually, we would like to model each pair $(\mathbf{x}_t, \mathbf{x}_{t+1})$ as a training pair for regression with g. However, we cannot simply substitute them directly into the GP model of (3.5) as this leads to the nonsensical expression $p(\mathbf{x}_2, ..., \mathbf{x}_N \mid \mathbf{x}_1, ..., \mathbf{x}_{N-1})$.

As above, we form the joint probability density over the latent coordinates and the dynamics weights $\mathbf{A}$ in (3.3). We then marginalize over the weights $\mathbf{A}$, i.e.,

$$p(\mathbf{X} \mid \bar{\alpha}) \;=\; \int p(\mathbf{X}, \mathbf{A} \mid \bar{\alpha})\, d\mathbf{A} \;=\; \int p(\mathbf{X} \mid \mathbf{A}, \bar{\alpha})\, p(\mathbf{A} \mid \bar{\alpha})\, d\mathbf{A} \;. \tag{3.7}$$

Incorporating the Markov property (3.1) gives:

$$p(\mathbf{X} \mid \bar{\alpha}) \;=\; p(\mathbf{x}_1) \int \prod_{t=2}^{N} p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{A}, \bar{\alpha})\; p(\mathbf{A} \mid \bar{\alpha})\, d\mathbf{A} \;, \tag{3.8}$$

where $\bar{\alpha}$ is a vector of kernel hyperparameters. Assuming an isotropic Gaussian prior on the columns of $\mathbf{A}$, it can be shown that this expression simplifies to:

$$p(\mathbf{X} \mid \bar{\alpha}) \;=\; p(\mathbf{x}_1) \frac{1}{\sqrt{(2\pi)^{(N-1)d} |\mathbf{K}_X|^d}} \exp\left(-\frac{1}{2} \mathrm{tr}\left(\mathbf{K}_X^{-1} \mathbf{X}_{out} \mathbf{X}_{out}^T \right) \right) \;, \tag{3.9}$$

where $\mathbf{X}_{out} = [\mathbf{x}_2, ..., \mathbf{x}_N]^T$, $\mathbf{K}_X$ is the $(N-1) \times (N-1)$ kernel matrix constructed from $\{\mathbf{x}_1, ..., \mathbf{x}_{N-1}\}$, and $\mathbf{x}_1$ is assumed to be have an isotropic Gaussian prior. For a derivation, see Appendix A.

We model dynamics using both the RBF kernel of the form of (3.6), as well as the following "linear + RBF" kernel:

$$k_X(\mathbf{x}, \mathbf{x}') \;=\; \alpha_1 \exp\left(-\frac{\alpha_2}{2} ||\mathbf{x} - \mathbf{x}'||^2 \right) + \alpha_3 \mathbf{x}^T \mathbf{x}' + \alpha_4^{-1} \delta_{\mathbf{x},\mathbf{x}'} \;. \tag{3.10}$$

The kernel corresponds to representing g as the sum of a linear term and RBF terms. The inclusion of the linear term is motivated by the fact that linear dynamical models, such as first and second-order autoregressive models, are useful for many systems. Hyperparameters α_1, α_2 represent the output scale and the inverse width of the RBF terms, and α_3 represents the output scale of the linear term. Together, they control the relative weighting between the terms, while α_4^{-1} represents the variance of the noise term $\mathbf{n}_{x,t}$.

It should be noted that, due to the nonlinear dynamical mapping in (3.3), the joint distribution of the latent coordinates is *not* Gaussian. Moreover, while the density over the initial state may be Gaussian, it will not remain Gaussian once propagated through the nonlinear dynamics. One can also see this in (3.9) since $\mathbf{x}_t$ terms occur inside the kernel matrix, as well as outside of it. The log likelihood is therefore not quadratic in $\mathbf{x}_t$.

Finally, we also place simple priors on the hyperparameters, i.e., $p(\bar{\alpha}) \propto \prod_i \alpha_i^{-1}$, and $p(\bar{\beta}) \propto \prod_i \beta_i^{-1}$. Note that the priors prefer small output scale (small $\alpha_1, \alpha_3, \beta_1$), large width for the RBFs (small α_2, β_2), and large noise variance (small α_4, β_3) to discourage overfitting. Together, the priors, the latent mapping, and the dynamics define a generative model for time series observations:

$$p(\mathbf{X}, \mathbf{Y}, \bar{\alpha}, \bar{\beta}) \;=\; p(\mathbf{Y}|\mathbf{X}, \bar{\beta})\, p(\mathbf{X}|\bar{\alpha})\, p(\bar{\alpha})\, p(\bar{\beta}) \;. \tag{3.11}$$

The corresponding graphical model is shown in Figure 3.1(b).

3.1 Multiple sequences

This model extends naturally to multiple sequences $\mathbf{Y}_1, ..., \mathbf{Y}_M$. Each sequence has associated latent coordinates $\mathbf{X}_1, ..., \mathbf{X}_M$ within a shared latent space. For the latent mapping g we can conceptually concatenate all sequences within the GP likelihood (3.5). A similar concatenation applies for the dynamics, but omitting the first frame of each sequence from $\mathbf{X}_{out}$, and omitting the final frame of each sequence from the kernel matrix $\mathbf{K}_X$. The same structure applies whether we are learning from multiple sequences, or learning

from one sequence and inferring another. That is, if we learn from a sequence $\mathbf{Y}_1$, and then infer the latent coordinates for a new sequence $\mathbf{Y}_2$, then the joint likelihood entails full kernel matrices $\mathbf{K}_X$ and $\mathbf{K}_Y$ formed from both sequences.

3.2 Higher-order features

The GPDM can be extended to model higher-order Markov chains, and to model velocity and acceleration in inputs and outputs. For example, a second-order dynamical model,

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{x}_{t-2}; \mathbf{A}) + \mathbf{n}_{x,t} \tag{3.12}$$

may be used to explicitly model the dependence of the prediction on two past frames (or on velocity). In the GPDM framework, the equivalent model entails defining the kernel function as a function of the current and previous time step:

$$\begin{aligned} k_X(\,[\mathbf{x}_t, \mathbf{x}_{t-1}],\,[\mathbf{x}_\tau, \mathbf{x}_{\tau-1}]\,) \;&=\; \alpha_1 \exp\left(-\frac{\alpha_2}{2}||\mathbf{x}_t - \mathbf{x}_\tau||^2 - \frac{\alpha_3}{2}||\mathbf{x}_{t-1} - \mathbf{x}_{\tau-1}||^2\right) \\ &\quad + \alpha_4\, \mathbf{x}_t^T \mathbf{x}_\tau \;+\; \alpha_5\, \mathbf{x}_{t-1}^T \mathbf{x}_{\tau-1} \;+\; \alpha_6^{-1} \delta_{t,\tau} \end{aligned} \tag{3.13}$$

Similarly, the dynamics can be formulated to predict velocity:

$$\mathbf{v}_{t-1} = f(\mathbf{x}_{t-1}; \mathbf{A}) + \mathbf{n}_{x,t} \tag{3.14}$$

Velocity prediction may be more appropriate for modeling smoothly motion trajectories. Using Euler integration with time step Δt, we have $\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{v}_{t-1}\Delta t$. The dynamics likelihood $p(\mathbf{X} \,|\, \bar{\alpha})$ can then be written by redefining $\mathbf{X}_{out} = [\mathbf{x}_2 - \mathbf{x}_1, ..., \mathbf{x}_N - \mathbf{x}_{N-1}]^T / \Delta t$ in (3.9). In this thesis, we use a fixed time step of $\Delta t = 1$. This is analogous to using $\mathbf{x}_{t-1}$ as a "mean function."

Higher-order features can also be fused together with position information to reduce the Gaussian process prediction variance [48, 34].

3.3 In relation to GP regression and GPLVM

We have motivated the development of GPDM through the marginalization of model parameters in a subspace dynamical system. Equivalently, the model can be thought of as using one set of state estimations to define both the observation and dynamics mapping.

In Gaussian process regression [33], the learned nonlinear function is not represented by a set of specific parameters, but by the training data (input/output pairs) itself and a few hyperparameters. Suppose the states $\mathbf{X}$ are given in addition to $\mathbf{Y}$, then the observation mapping is fully defined by inputs $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_t, \cdots, \mathbf{x}_N\}$ and outputs $\mathbf{Y} = \{\mathbf{y}_1, \cdots, \mathbf{y}_t, \cdots, \mathbf{y}_N\}$. The k^{th} order dynamics mapping is defined by inputs $\mathbf{X}_{in} = \{\mathbf{x}_1, \cdots, \mathbf{x}_t, \cdots, \mathbf{x}_{N-k}\}$ and outputs $\mathbf{X}_{out} = \{\mathbf{x}_{k+1}, \cdots, \mathbf{x}_t, \cdots, \mathbf{x}_N\}$. It can be shown that both prediction distributions $p(\tilde{\mathbf{y}}|\tilde{\mathbf{x}}, \mathbf{X}, \mathbf{Y}, \bar{\beta})$ and $p(\tilde{\mathbf{x}}_{t+1}|\tilde{\mathbf{x}}_t, \mathbf{X}_{in}, \mathbf{X}_{out}, \bar{\alpha})$ are Gaussian, and can be computed analytically (4.4), (4.5). Our model corresponds to describing both mappings using Gaussian processes, where the state estimation $\mathbf{X}$ is obtained by maximizing the joint likelihood (3.11).

As we alluded to earlier, the GPDM is closely related to the GPLVM [25]. In the GPLVM, a unit Gaussian prior is used on $\mathbf{X}$ during estimation, which is sensible when the data is assumed to be generated independently. The GPDM can be thought of as replacing the Gaussian prior on $\mathbf{X}$ with (3.9), which reflects our prior belief that the observed data is generated by an autoregressive dynamical process. The general nonlinear dynamics mapping we use gives rise to a relatively loose preference for smooth trajectories (i.e., nearby points in latent space should map to nearby points). Stronger domain knowledge can be incorporated by modifying the kernel function. Namely, the linear kernel leads to a prior that discourages nonlinearities in general. Using multiple sequences and higher order models simply correspond to changing the expression for $p(\mathbf{X}|\bar{\alpha})$.

From a practical point of view, our goal is to augment GPLVM with latent space

dynamics. The state estimation process in GPLVM has the tendency to place data points that are far apart in observation space, far apart in latent space (even more so than PCA). While this property is useful for data visualization, it leads to trajectories with large jumps when applied to human motion data. The obvious solution of learning dynamics on such trajectories typically leads to underfitting, as large jumps are treated as outliers. Furthermore, we can see from the graphical model that the hidden states are constrained by the data as well as the dynamics function. This is not the case if dynamics is learned after learning GPLVM, where state estimates are not influenced by the dynamics function at all.

Chapter 4

Properties of the GPDM and Algorithms

Learning the GPDM from measurements $\mathbf{Y}$ entails minimizing the negative log-posterior:

$$\begin{aligned} \mathcal{L} &= -\ln p(\mathbf{X}, \bar{\alpha}, \bar{\beta} \,|\, \mathbf{Y}) && (4.1) \\ &= \frac{d}{2} \ln |\mathbf{K}_X| + \frac{1}{2} \mathrm{tr}\left(\mathbf{K}_X^{-1} \mathbf{X}_{out} \mathbf{X}_{out}^T\right) + \sum_j \ln \alpha_j && (4.2) \\ &\quad - N \ln |\mathbf{W}| + \frac{D}{2} \ln |\mathbf{K}_Y| + \frac{1}{2} \mathrm{tr}\left(\mathbf{K}_Y^{-1} \mathbf{Y} \mathbf{W}^2 \mathbf{Y}^T\right) + \sum_j \ln \beta_j \end{aligned}$$

up to an additive constant. We minimize $\mathcal{L}$ with respect to $\mathbf{X}$, $\bar{\alpha}$, and $\bar{\beta}$ numerically. Our implementation is based on Neil Lawrence's GPLVM code [1]. Optimization is performed with the scaled conjugate gradient (SCG) implementation in NETLAB. Minimization with respect to $\mathbf{W}$ is done in closed-form, setting

$$w_k \Leftarrow \sqrt{N(\mathbf{y}_{:,k}^T \mathbf{K}_Y^{-1} \mathbf{y}_{:,k})^{-1}}, \qquad (4.3)$$

once every ten iterations. Gradients are given in Appendix B.

[1] http://www.dcs.shef.ac.uk/~neil/gplvm/

4.1 Visualizations

Figures 4.1 and 4.2 show a GPDM 3D latent space learned from a human motion capture data comprising three walk cycles. The human motion capture sequences were obtained from the CMU motion capture database (http://mocap.cs.cmu.edu). Each pose comprised 56 Euler angles for joints, three global (torso) pose angles, and three global (torso) translational velocities. For learning, the data was mean-subtracted, and the latent coordinates were initialized with PCA. The GPDMs were learned by minimizing $\mathcal{L}$ in (4.2).

Figure 4.1(b) shows a 3D SGPLVM learned from the walking pose data. Notice that the sequence of latent coordinates is not particularly smooth; there are numerous cases where consecutive poses in the walking sequence are relatively far apart in the latent space. By contrast, Figure 4.1(a) shows that the GPDM produces a much smoother configuration of latent positions. Here the GPDM arranges the latent positions roughly in the shape of a saddle.

In order to further visualize properties of the GPDM, Figure 4.2(a) shows 25 fair samples from the latent dynamics of the GPDM. All samples are conditioned on the same initial state, $\mathbf{x}_0$, and each has a length of 60 time steps.[2] As noted above, because we marginalize over the weights of the dynamic mapping, $\mathbf{A}$, the dynamical model is somewhat complex; that is, the joint distribution over a sequence of poses cannot be factored according to the underlying Markov model (Figure 3.1(a)) into a sequence of low-order Markov transitions. As a consequence, one cannot properly draw samples from the model in a causal fashion, one state at a time from the effective transition density, $p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1})$. Instead, here the fair samples, $\{\tilde{\mathbf{X}}_{1:60}^{(j)}\}_{j=1}^{25}$, with $\tilde{\mathbf{X}}_{1:60}^{(j)} \sim p(\tilde{\mathbf{X}}_{1:60} \,|\, \mathbf{x}_0, \mathbf{X}, \mathbf{Y}, \bar{\alpha})$, were generated with hybrid Monte Carlo [33] (HMC). The resulting trajectories shown in Figure 4.2(a) are smooth and generally follow the path of the training motions. Figure 4.4 shows

[2]The length was chosen to be just less than a full gait cycle for ease of visualization.

example animations from HMC samples.

Another visualization of the GPDM is given in Figure 4.2(b). Following [25], this is a volume visualization of the inverse reconstruction variance, i.e., $-\ln \sigma^2_{\mathbf{y}|\mathbf{x},\mathbf{X},\mathbf{Y},\bar{\beta}}$. In effect this shows the confidence with which the model reconstructs the pose from the latent position; in effect, the GPDM models a high probability "tube" around the data.

Figure 4.6 shows 2D models. In theory, a walk cycle can be represented as a single closed curve in 2D. However, notice that the 2D GPDM produces large "jumps" in the latent trajectories. As a result, the dynamics are over-smoothed and simulation is unreliable (e.g., see Figure 4.7). These "jumps" appear when the 2D trajectories initialized by PCA intersect. While the 2D PCA projections are similar, the reconstructions and dynamics at intersections are often different. The learning typically leads to a local minima which breaks the curves to establish latent coordinates that yield better reconstruction, but without removing the discontinuities. Many of these problems disappear in 3D where intersections are not common, resulting in smoother models (see Figure 4.1b).

Figure 4.8 shows GPLVM models learned from human motion capture data of golf swings, which show a 3D GPDM learned from three swings of a golf club. The plots show the same latent space from two viewpoints looking parallel to two of the coordinate axes. The learning aligns the sequences and nicely accounts for variations in speed during the club trajectory.

4.2 Mean prediction sequences

For both 3D people tracking and computer animation, it is desirable to generate new motions efficiently. Here we consider a simple online method for generating a new motion, called *mean-prediction*, which avoids the relatively expensive Monte Carlo sampling used above. In mean-prediction, we consider the next timestep $\tilde{\mathbf{x}}_t$ conditioned on $\tilde{\mathbf{x}}_{t-1}$ from

the Gaussian prediction [33]:

$$\tilde{\mathbf{x}}_t \sim \mathcal{N}(\mu_X(\tilde{\mathbf{x}}_{t-1}); \sigma_X^2(\tilde{\mathbf{x}}_{t-1})\mathbf{I}) \tag{4.4}$$

$$\mu_X(\mathbf{x}) = \mathbf{X}_{out}^T \mathbf{K}_X^{-1} \mathbf{k}_X(\mathbf{x}) \ , \qquad \sigma_X^2(\mathbf{x}) = k_X(\mathbf{x}, \mathbf{x}) - \mathbf{k}_X(\mathbf{x})^T \mathbf{K}_X^{-1} \mathbf{k}_X(\mathbf{x}) \tag{4.5}$$

where $\mathbf{k}_X(\mathbf{x})$ is a vector containing $k_X(\mathbf{x}, \mathbf{x}_i)$ in the i-th entry and $\mathbf{x}_i$ is the i^{th} training vector. In particular, we set the latent position at each timestep to be the most likely (mean) point given the previous step: $\mathbf{x}_t = \mu_X(\mathbf{x}_{t-1})$. In this way we ignore the process noise that one might normally add. Compared to including the random process noise at each time step, we find that this mean-prediction often generates motions that are more like the fair samples shown in Figure 4.1(a). Similarly, new poses are given by $\mathbf{y}_t = \mu_Y(\mathbf{x}_t)$.

Depending on the dataset and the choice of kernels, long sequences generated by mean-prediction can diverge from the data. On our data sets, mean-prediction trajectories from the GPDM with an RBF or linear+RBF kernel for dynamics usually produce sequences that roughly follow the training data (e.g., see the red curves in Figures 4.5 and 4.8b,c). This usually means producing closed limit cycles with walking data. We also found that mean-prediction motions are often very close to the mean obtained from the HMC sampler; by initializing HMC with mean-prediction, we find that the sampler reaches equilibrium in a small number of interactions. Compared to the RBF kernels, mean-prediction motions generated from GPDMs with the linear kernel often deviate from the original data (e.g., see Figs 4.5b, 4.6b, and 4.7b), and lead to over-smoothed animation.

4.3 Optimization

While mean-prediction is efficient, there is nothing in the algorithm that prevents trajectories from drifting away from the training data. It is sometimes desirable to optimize a particular motion under the GPDM, which often reduce drift of the mean-prediction

motions. To optimize a new sequence, we first select a starting point $\tilde{\mathbf{x}}_1$ and a number of time steps. The likelihood $p(\tilde{\mathbf{X}} \,|\, \mathbf{X}, \bar{\alpha})$ of the new sequence $\tilde{\mathbf{X}}$ is then optimized directly (holding the latent positions of the previously learned latent positions, $\mathbf{X}$, and hyperparameters, $\bar{\alpha}$, fixed). To see why optimization generates motion close to the training data, note that the variance of pose $\mathbf{x}_{t+1}$ is determined by $\sigma_X^2(\mathbf{x}_t)$, which will be lower when $\mathbf{x}_t$ is nearer the training data. Consequently, the likelihood of $\mathbf{x}_{t+1}$ can be increased by moving $\mathbf{x}_t$ closer to the training data. This generalizes the preference of the SGPLVM for poses similar to the examples [18], and is a natural consequence of the Bayesian approach. As an example, Figure 4.3 shows an optimized walk sequence initialized from the mean-prediction.

4.4 Forecasting

We performed a simple experiment to compare the predictive power of the GPDM to a linear dynamical system, implemented as a GPDM with linear kernel in the latent space and RBF latent mapping. We trained each model on the first 130 frames of the 60Hz walking sequence (corresponding to 2 cycles), and tested on the remaining 23 frames. From each test frame mean-prediction was used to predict the pose 8 frames ahead, and then the RMS pose error was computed against ground truth. The test was repeated using mean-prediction and optimization for three kernels and first-order Markov dynamics:

	Linear	RBF	Linear+RBF
mean-prediction	59.69	48.72	36.74
optimization	58.32	45.89	31.97

Due to the nonlinear nature of the walking dynamics in latent space, the RBF and Linear+RBF kernels outperform the linear kernel. Moreover, optimization (initialized by mean-prediction) improves the result in all cases, for reasons explained above.

4.5 Missing data

The GPDM model can also handle incomplete data (a common problem with human motion capture sequences). The GPDM is learned by minimizing $\mathcal{L}$ (4.2), but with the reconstruction terms corresponding to missing frames removed. The latent coordinates for missing data are initialized by cubic spline interpolation from the 3D PCA initialization of the observed points.

While this produces good results for short missing segments (e.g., 10–15 frames of the 157 frame walk sequence used in Figure 4.1), it fails on long missing segments. The problem lies with the difficulty in initializing the missing latent positions sufficiently close to the training data. To solve the problem we first learn a model with a subsampled data sequence. Reducing sampling density effectively increases uncertainty in the reconstruction process so that the probability density over the latent space falls off more smoothly from the data. We then restart the learning with the entire data set, but with the kernel hyperparameters fixed. In doing so, the dynamics terms in the objective function exert more influence over the latent coordinates of the training data, and a smooth model is learned.

With 50 missing frames of the 157 walk sequence, this optimization produces models (Figure 4.9) that are much smoother than those in Figure 4.1. The linear kernel is able to pull the latent coordinates onto a cylinder (Figure 4.9b), and thereby provides an accurate a dynamical model. Both models shown in Figure 4.9 produce estimates of the missing poses that are visually indistinguishable from the ground truth.

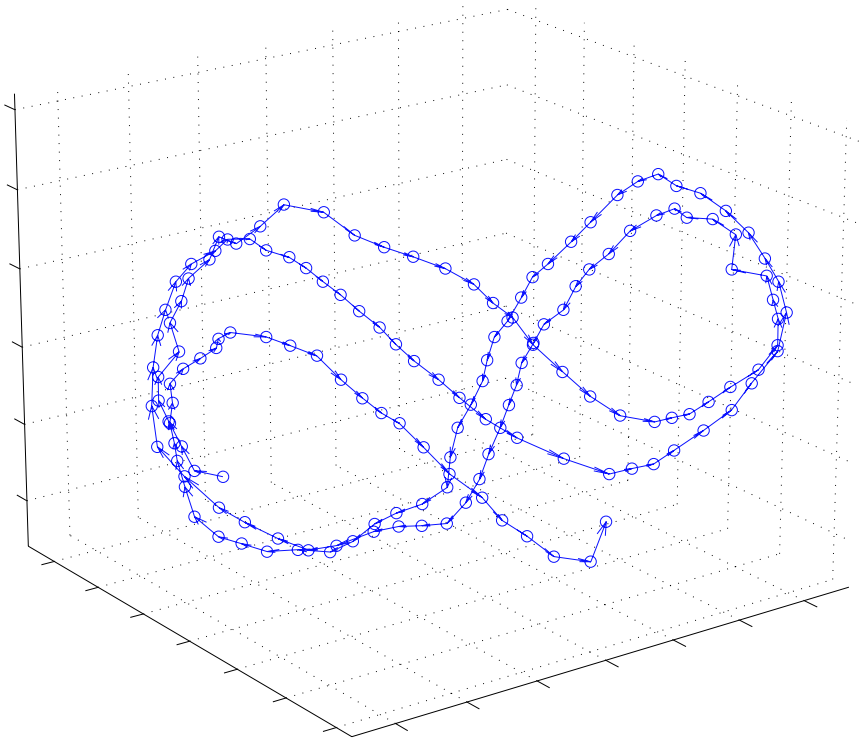
4.6 Multiple subjects

The GPDMs discussed so far have all been trained with one single subject doing one activity in one particular style. However, for practical applications, the prior motion model must be general enough to make sensible predictions on motion data from different

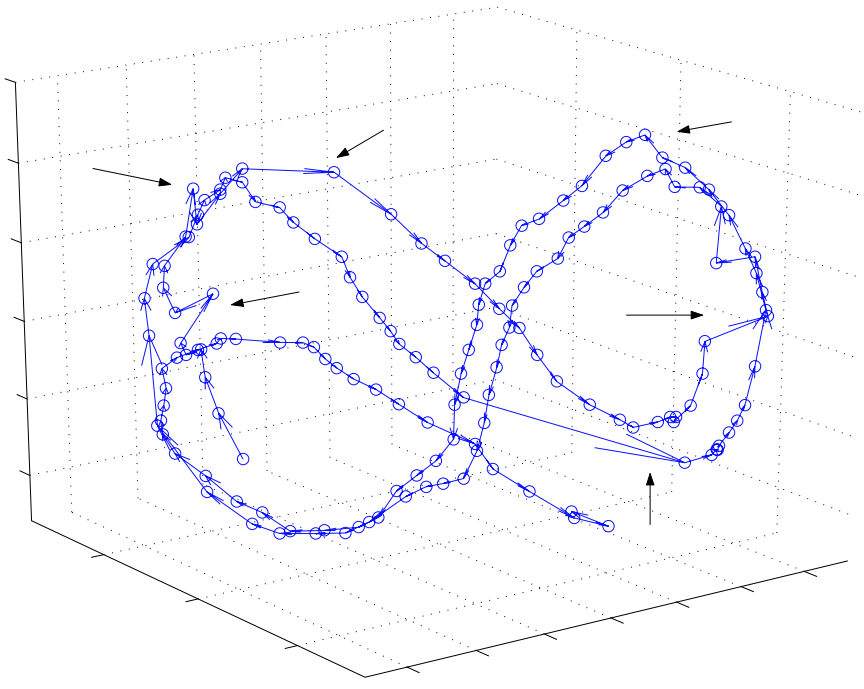
subjects. We find that GPDMs learned from the training sets described so far tend to be overly specific to the observed motion and subject. For example, a human walker tracking application may forecast future poses by first estimating the most likely latent space position for the current pose. One can only expect the predictions to be accurate if the estimated position is near the training data in latent space, which does not consistently happen if the new pose is not from the same subject performing the same walking style.

One potential solution to the above problem is to use richer sets of training data. In the ideal case, the model should learn a high-probability tube region in the latent space, representing all likely poses of a particular activity regardless of the subject and style. Figure 4.10 shows results from learning with six walk cycles, each from a different person. The inverse reconstruction variance plot shows regions with high reconstruction certainty separated into small clusters. In constrast with Figure 4.10(a), Figure 4.2(b) shows a tube-shaped region of high reconstruction certainty from the GPDM with a single walker. The simulation results from the latter tend to generate points within the tube, which leads to good reconstruction. Not surprisingly, animation results corresponding to the red curve in Figure 4.10(b) look awkward and unsmooth. Notice that the simulation trajectory contain points outside of regions with high reconstruction certainty in latent space. These points do not have enough neighbours around them for good reconstruction, and are typically reconstructed to the mean pose in observation space.

We would like to see a GPDM trained with multiple subjects performing the same activity to have a inverse reconstruction variance plot similar to Figure 4.2(b). Unfortunately, the learning algorithm proposed in this section does not produce the desired result.

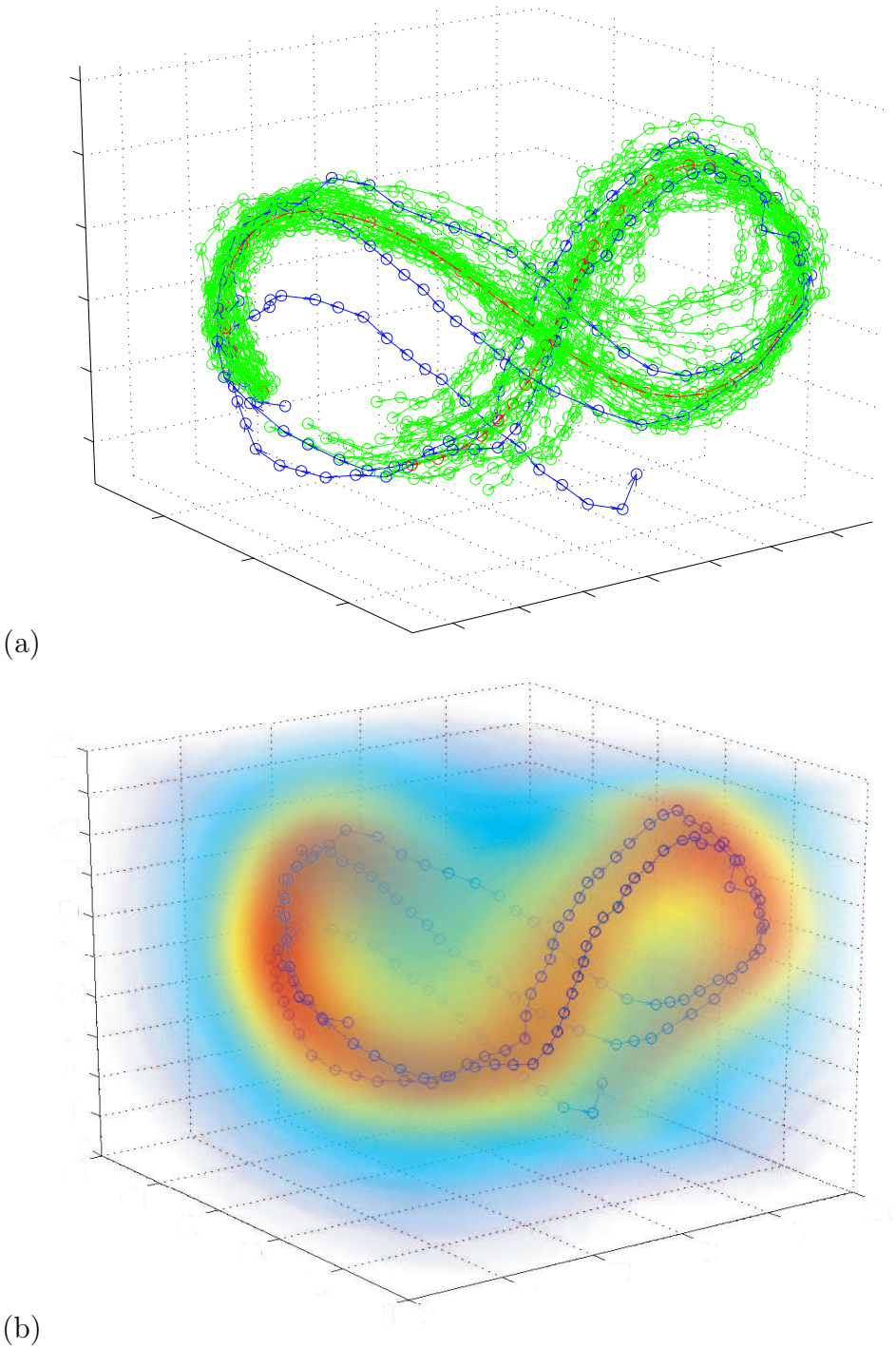


(a)



(b)

Figure 4.1: Models learned from a walking sequence comprising 2.5 gait cycles. The latent coordinates learned with a GPDM (a) and GPLVM (b) are shown in blue. Vectors depict the temporal sequence.



(a)

(b)

Figure 4.2: Visualizations of GPDM learned from a walking sequence. (a) Random trajectories drawn from the model using hybrid Monte Carlo are green. (b) negative log variance for reconstruction shows positions in latent space that are reconstructed with high confidence.

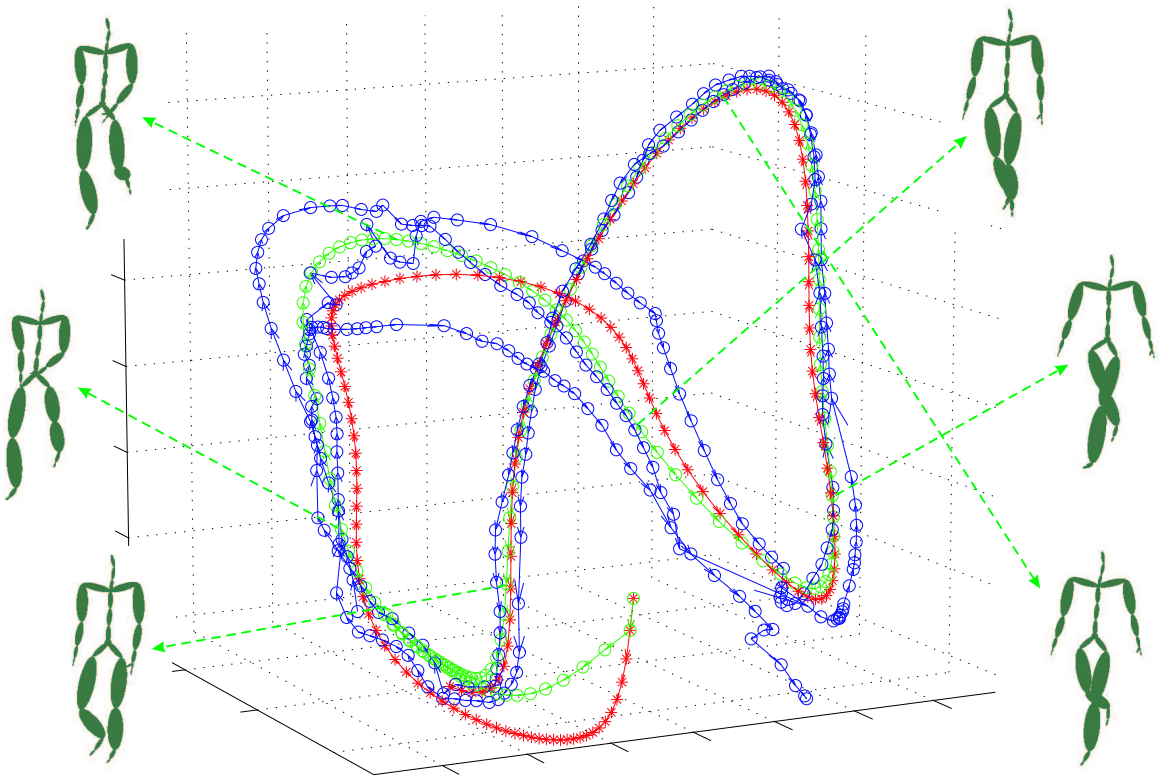


Figure 4.3: A GPDM of walk data at 120Hz learned with RBF+linear kernel dynamics. The simulation (red) was started far from the training data, and then optimized (green). The poses depicted were reconstructed from latent points along the optimized (simulation) trajectory.

Figure 4.4: Animation of samples from hybrid Monte Carlo. The trajectory length is 200 frames, four selected frames of each sample are shown here.

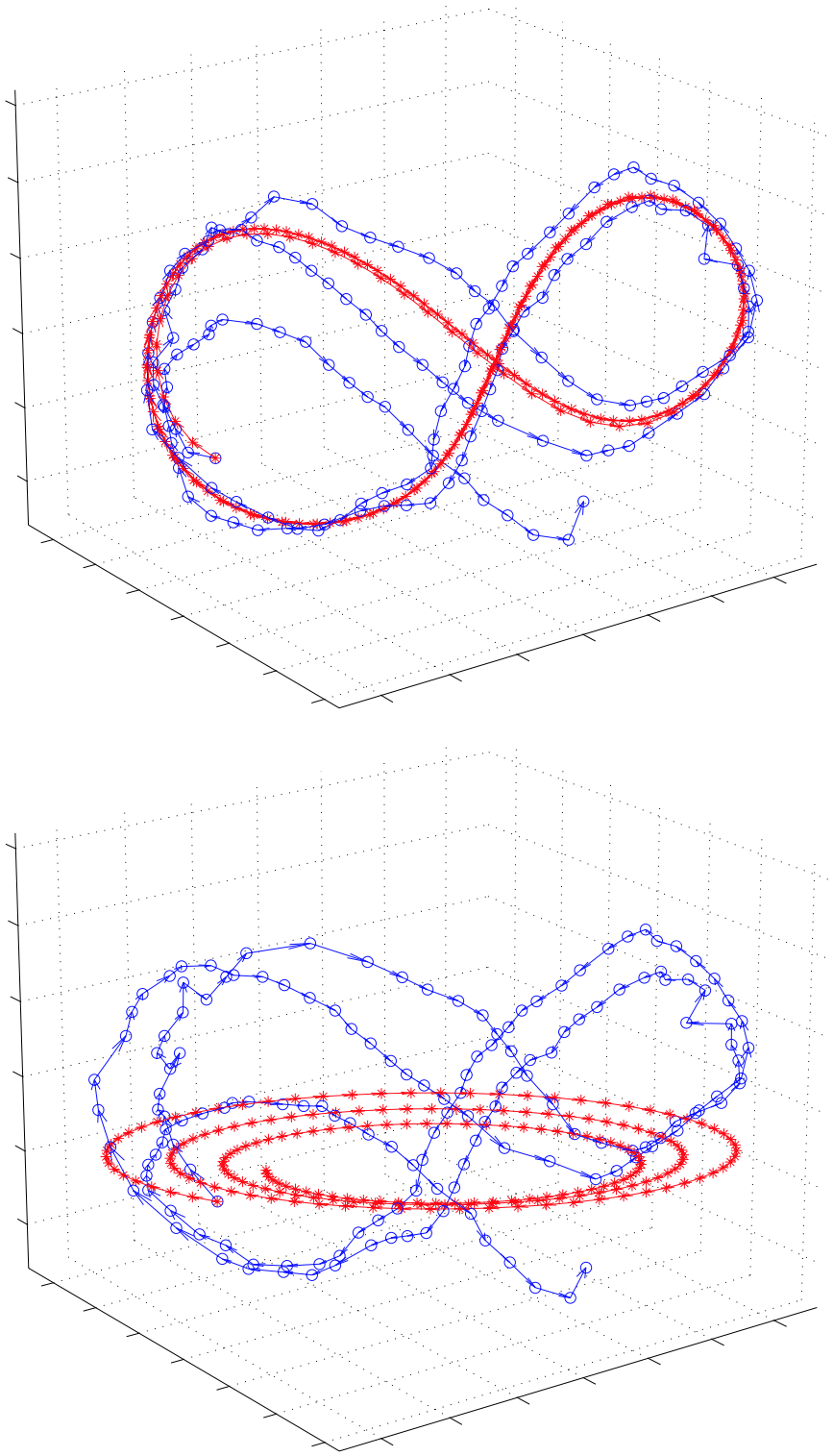


Figure 4.5: Two GPDMs and mean predictions. The first is that from the previous figure. The second was learned with a linear kernel.

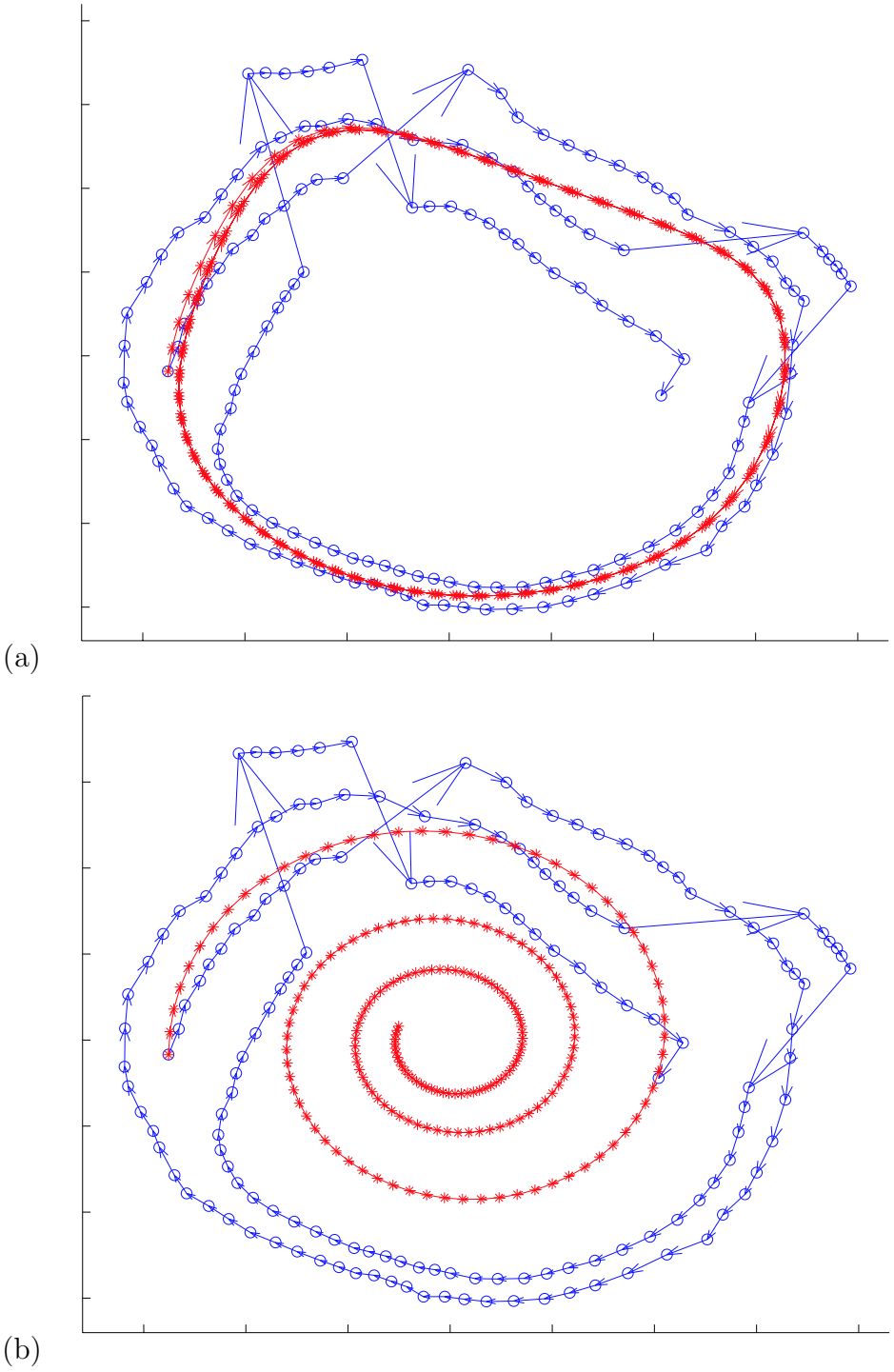
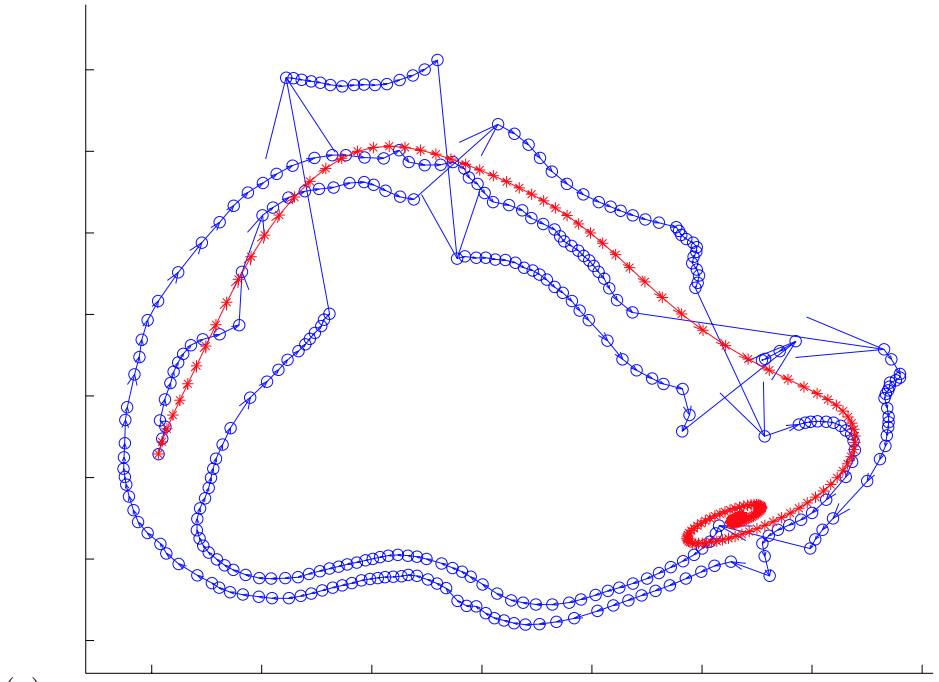
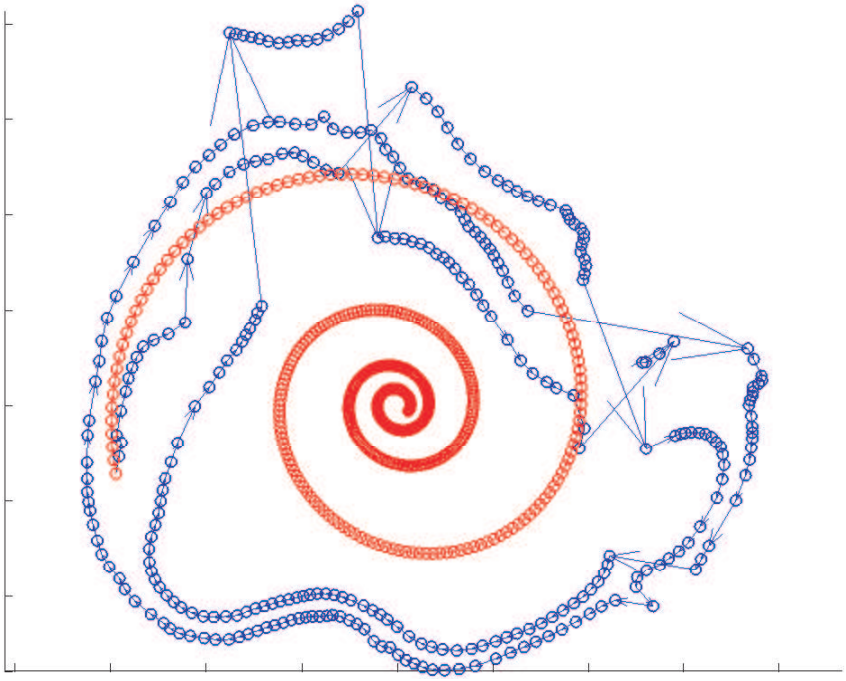


Figure 4.6: GPDM models of walking in a 2D latent space. As above, latent coordinates of data are shown in blue, and mean-prediction simulations are red. (a) GPDM with a linear+RBF kernel. The nonlinear kernel produces a closed limit cycle. (b) GPDM with a linear kernel does not produce a limit cycle for long simulations.



(a)



(b)

Figure 4.7: GPDM models of walking in a 2D latent space, with double the number of data points compared to Figure 4.6. (a) GPDM with a linear+RBF kernel. (b) GPDM with a linear kernel.

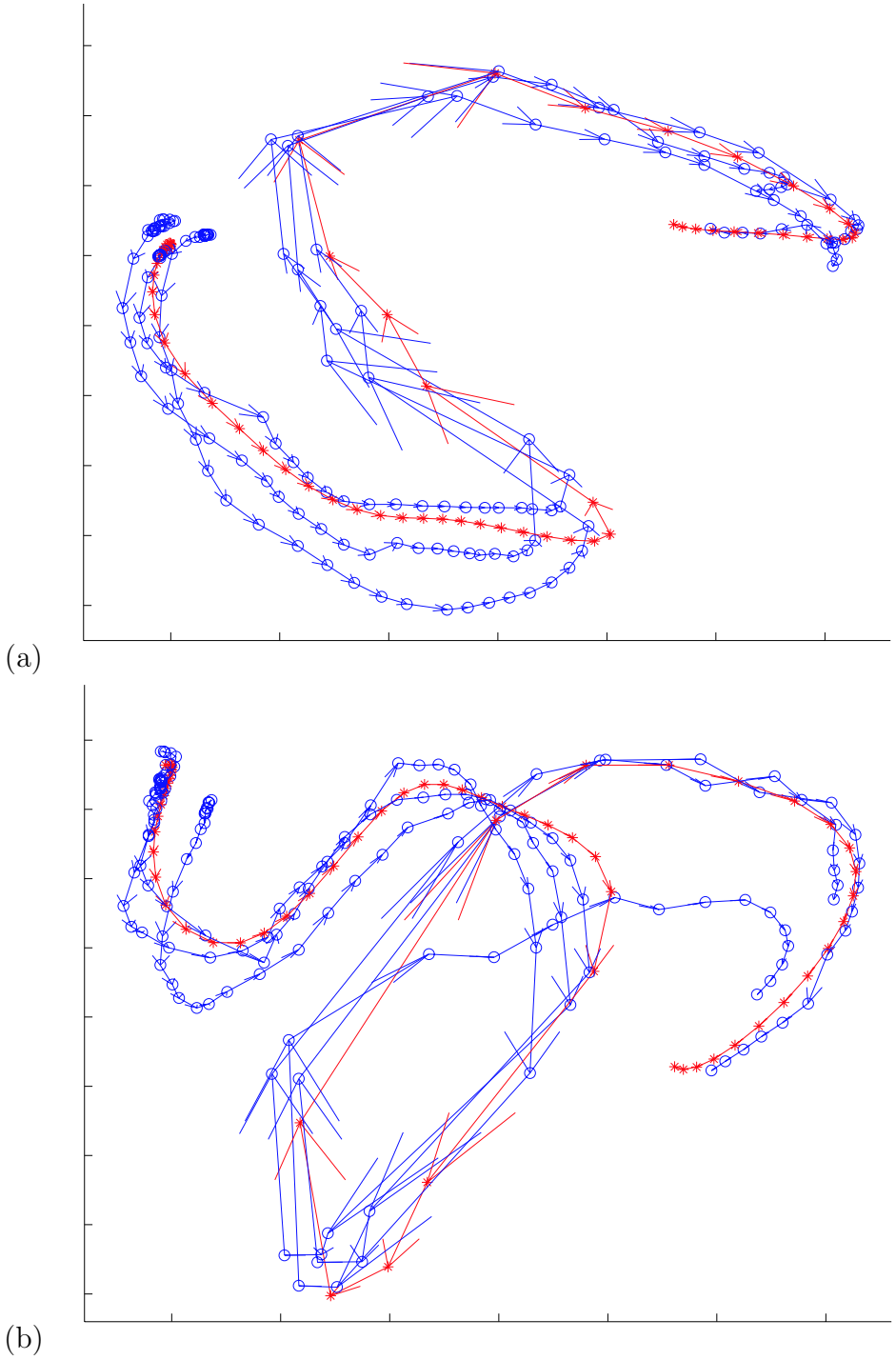


Figure 4.8: The GPDM model was learned from three swings of a golf club, using a 2^{nd} order RBF kernel for dynamics. The two plots show 2D orthogonal projections of the 3D latent space.

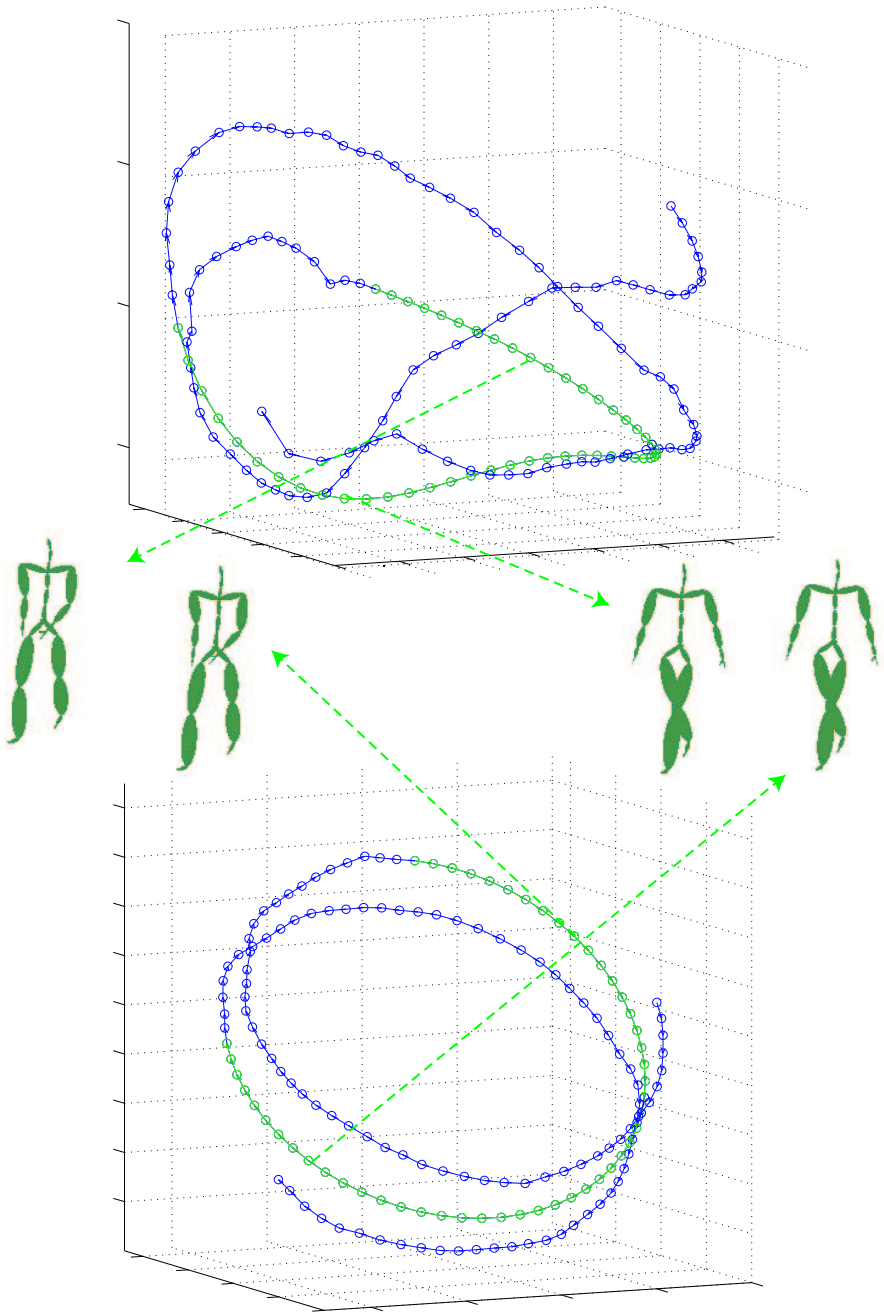
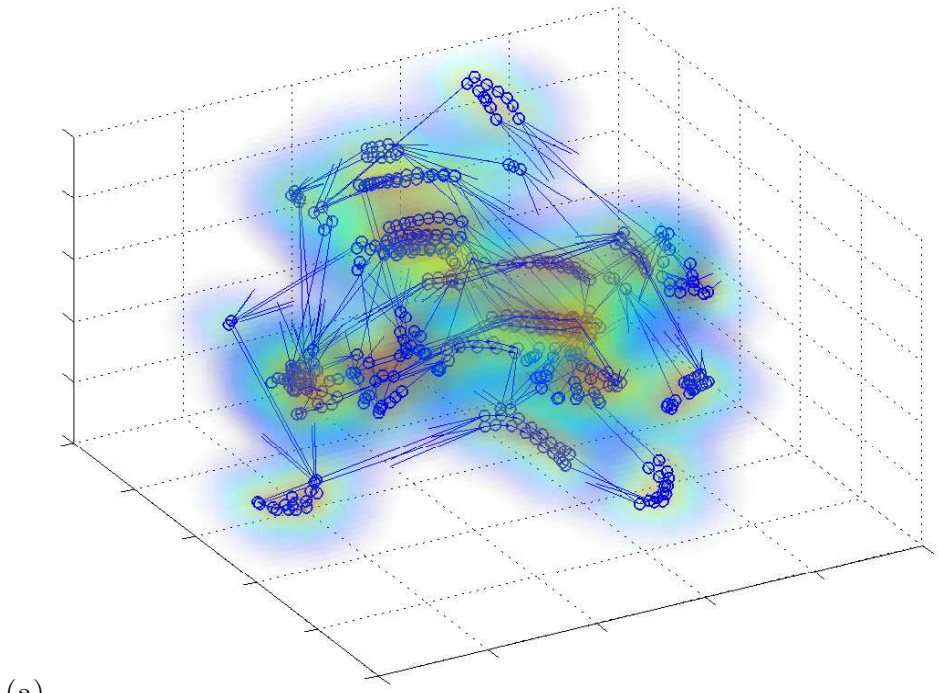
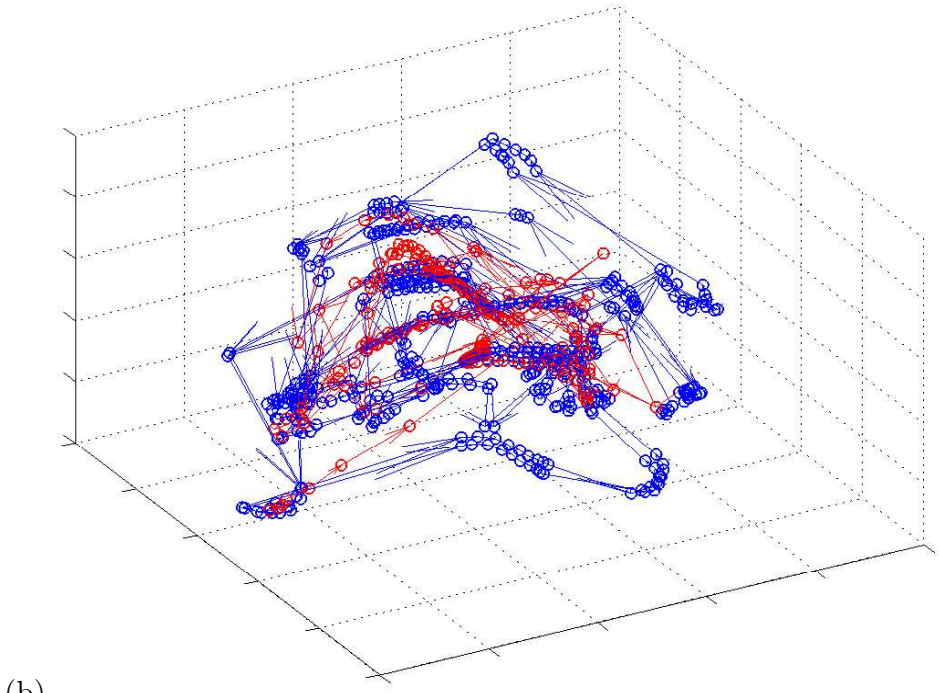


Figure 4.9: GPDM from walk sequence with missing data learned with (a) a RBF+linear kernel for dynamics, and (b) a linear kernel for dynamics. Blue curves depict original data. Green curves are the reconstructed, missing data.



(a)



(b)

Figure 4.10: A GPDM learned with six walkers, one cycle each. (a) Inverse reconstruction variance plot, notice the regions with high reconstruction certainty are separated into small clusters. (b) Simulation results.

Chapter 5

Accounting for Latent Space Uncertainty

The shape of the regions with high reconstruction certainty (warm-coloured regions in the volume visualizations) discussed in Section 4.6 is completely determined by the placement of training poses in latent space and the hyperparameters for reconstruction. For example, the clumpy warm-coloured regions in Figure 4.10(a) is partially a consequence of the unsmooth arrangement of training data in latent space. On the other hand, smooth latent space trajectories tend to define a tube-shaped warm-coloured region around the training data, as seen in Figure 4.2(b). As discussed previously, having a tube-shaped region with high reconstruction certainty leads to dynamic simulations that generate poses within the tube, which is good for reconstruction.

One of the motivations for GPDM is to augment GPLVM with temporal information so that smooth latent space trajectories can be recovered from smooth motion data. To this end, we have seen the effectiveness of GPDM on walk cycles from a single subject projected into 3D (Figure 4.1). In 2D, or 3D with multiple walkers, neither GPLVM nor GPDM produce particularly smooth trajectories (Figure 4.6, Figure 4.10). We discuss methods to produce smoother trajectory estimates in this chapter.

5.1 Priors on hyperparameters

The MAP learning algorithm introduced in the previous chapter relies on point estimates of the latent coordinates to estimate the hyperparameters and vice versa. The loss of uncertainty in latent space likely results in overfitted hyperparameters in both reconstruction and dynamics. The estimation of $\mathbf{X}$ depends on the sum of $-\ln p(\mathbf{Y}|\mathbf{X}, \bar{\beta})$ and $-\ln p(\mathbf{X}|\bar{\alpha})$. Since there are no explicit scaling factors between the terms, the influence each term has on the value of $\mathbf{X}$ depends on hyperparameters $\bar{\alpha}$ and $\bar{\beta}$. Moreover, large jumps in the trajectory estimation are typically caused by the reconstruction term exerting more influence than the dynamics term, as the latter penalizes large jumps in $\mathbf{X}$.

Using the RBF kernel for reconstruction as an example, the hyperparameter setting with large RBF width, small output scale, and large noise variance corresponds to small values of β_1, β_2, and β_3. Such a setting is "conservative", or more uncertain, in the sense that the GP predictions have higher variance, and prefers to underfit the data instead of overfitting it. Intuitively, when the hyperparameters are set conservatively (Figure 5.1a), the likelihood function is smoother and less sensitive to changes in data points. In the extreme case, if the hyperparameters consider the entire data set to be noise, then the likelihood is independent of the data set. On the other hand, if the hyperparameters are set more aggressively and overfit the data (Figure 5.1b), small changes to the input data could have a significant impact on the likelihood function. In our learning algorithm, the reconstruction hyperparameters ($\bar{\beta}$), typically overfit the data more than the hyperparameters for dynamics ($\bar{\alpha}$), resulting in unsmooth trajectories. It is not clear at this moment why the algorithm seems to favour reducing reconstruction error over dynamics prediction error. The difference in dimensionality between the observation space and the latent space is one possible explanation, and the solution may depend on finding a sensible way of adjusting distance measurements based on dimensionality.

It is tempting to set values of the hyperparameters by hand, keeping them fixed

so that only $\mathbf{X}$ is learned in the model. Although we can trade off the smoothness of the trajectories with reconstruction error on the training data this way, it is difficult in practice. The scales of hyperparameters are closely intertwined with each other [18], and are highly dependent on the data set.

A more principled way of influencing values of parameters is by attaching a prior distribution on them so that certain values are preferred to others before any data is seen. Following the GPLVM, the priors on hyperparameters in the GPDM used in Chapter 3 are simply $p(\bar{\alpha}) \propto \prod_i \alpha_i^{-1}$, and $p(\bar{\beta}) \propto \prod_i \beta_i^{-1}$. As discussed in previous chapters, these priors assign high probabilities to small values of $\bar{\alpha}$ and $\bar{\beta}$, which encourages the hyperparameters to be conservative.

To incorporate a stronger preference for smooth trajectories, we could employ a prior belief that there is more uncertainty (small $\beta_1, \beta_2, \beta_3$) in the reconstruction mapping than the dynamics mapping by increasing the strength of the prior on $\bar{\beta}$. In particular, $p(\bar{\beta}) \propto \prod_i \beta_i^{-M}$, where $M > 1$. Recall that we optimize the negative log posterior:

$$\begin{aligned} \mathcal{L} &= -\ln p(\mathbf{X}, \bar{\alpha}, \bar{\beta} \,|\, \mathbf{Y}) && (5.1) \\ &= \frac{d}{2} \ln |\mathbf{K}_X| + \frac{1}{2} \mathrm{tr}\left(\mathbf{K}_X^{-1} \mathbf{X}_{out} \mathbf{X}_{out}^T\right) + \sum_j \ln \alpha_j && (5.2) \\ &\quad - N \ln |\mathbf{W}| + \frac{D}{2} \ln |\mathbf{K}_Y| + \frac{1}{2} \mathrm{tr}\left(\mathbf{K}_Y^{-1} \mathbf{Y} \mathbf{W}^2 \mathbf{Y}^T\right) + M \sum_j \ln \beta_j \end{aligned}$$

up to an additive constant. Note that M becomes a scaling factor for the reconstruction prior.

Figure 5.2 illustrates the effects of M on the latent space trajectories. With $M = 1$, the optimized $\mathbf{X}$ depends largely on the reconstruction term, which pulls points apart to reduce reconstruction error. A much smoother trajectory is obtained with $M = 1000$, where the dynamics term dominates.

Since the motivation for introducing M is to influence the optimized values of $\bar{\alpha}$ and $\bar{\beta}$, we want to examine their values more closely. Both mappings use RBF kernels here, so both sets of hyperparameters consist of output scale (α_1, β_1), inverse width (α_2, β_2), and

Table 5.1: Hyperparameter Values for 60Hz Walking Data

	$M = 1$	$M = 10$	$M = 500$	$M = 1000$	MCEM
rec. RBF width β_2^{-1}	0.379	0.380	0.66	6.36	0.0034
rec. output scale β_1	0.00175	0.00155	0.000547	0.000371	0.00127
rec. noise variance β_3^{-1}	0.00000351	0.00000324	0.00000554	0.0000162	0.00000435
$\sum_j \ln \beta_j$	7.18	7.14	5.25	1.28	11.4
training rec. error	2.45	2.45	2.74	4.82	2.73
dyn. RBF width α_2^{-1}	2.99	3.06	1.17	10.04	14.9
dyn. output scale α_1	0.799	0.7501	1.17	16.1	2.98
dyn. noise variance α_3^{-1}	0.0142	0.0142	0.0419	0.0000528	0.0000152
$\sum_j \ln \alpha_j$	2.94	2.91	1.67	10.32	9.49

inverse noise variance (α_3, β_3). One way to get an idea of the certainty of the mapping represented by a particular set of hyperparameters is to consider $\sum_j \ln \alpha_j$ and $\sum_j \ln \beta_j$. The higher these values are, the more certain the mapping is, and the more influence it has on latent coordinates. The hyperparameter values after 500 iterations versus M values are shown in Table 5.1. Notice that for low values of M, $\sum_j \ln \alpha_j$ is much less than $\sum_j \ln \beta_j$. Examining the latent space coordinates in these models, we find large jumps as in Figure 5.2(a). On the other hand, the smooth trajectory in Figure 5.2(b) corresponds to a model with $\sum_j \ln \alpha_j$ much greater than $\sum_j \ln \beta_j$, as indicated in the $M = 1000$ column. The decrease in reconstruction certainty can also be seen from the increase in training reconstruction error as M increases. Figure 5.6 shows learning results on the golf swing data set, which consists of 174 data points in total, setting $M = 1500$ results in reasonably smooth trajectories.

It would be ideal if the effect of M on hyperparameters is independent from the

Table 5.2: Hyperparameter Values for 120Hz Walking Data

	$M = 500$	$M = 1000$	$M = 3000$
rec. RBF width β_2^{-1}	0.195	0.285	1.74
rec. output scale β_1	0.000886	0.000395	0.000399
rec. noise variance β_3^{-1}	0.00000143	0.00000196	0.000047
$\sum_j \ln \beta_j$	8.07	6.56	1.59
training rec. error	1.35	1.87	5.56
dyn. RBF width α_2^{-1}	2.02	2.08	1.53
dyn. output scale α_1	0.7534	0.7563	1.25
dyn. noise variance α_3^{-1}	0.0193	0.0222	0.00000313
$\sum_j \ln \alpha_j$	3.06	2.93	12.5

data set. Unfortunately, this is not the case, as seen in Figure 5.3 with a more densely sampled walking data. We see that the model learned with $M = 1000$ still produce large jumps and uneven prediction variances. To obtain comparable smooth trajectories with the previous data set, the model needed to be learned with $M = 3000$. The general behaviour of hyperparameters with respect to M remains similar to the smaller data set from before, as indicated in Table 5.2. However, the trade-off between $\sum_j \ln \alpha_j$ and $\sum_j \ln \beta_j$ occurs more slowly with respect to M.

Despite the dependency on the data set, one could try different settings of M until a smooth trajectory is learned. A heuristic that seems to work well in practice is picking M to be approximately 10 times the number of data points and adjust its value up or down by observing the hyperparameter values. Note that when M is too large, the algorithm gives up on trying to identify patterns in the data and treats the entire data set as noise. In this case, the RBF width of the reconstruction function tend to grow very quickly, which is noticeable even in the first few iterations.

The temporal sampling rate does not seem to influence the choice of M for our walking data set. In Figure 5.4, the first 157 frames from a 120Hz walking data set are used to learn a model. Setting $M = 1000$ results in a smooth latent trajectory, whereas the learning algorithm fails to recover any meaningful reconstruction function for $M = 3000$.

Figure 5.5 shows simulations from the walking data set with both $M = 1$ and $M = 1000$. In both cases, the simulation stays reasonably close to the training data. The animation corresponding to the Figure 5.5(a) contains noticeable jerkiness at the character's right foot upon ground contact, whereas the animation corresponding to Figure 5.5(b) look visually identical to simulations from 3D GPDMs.

We have focused our discussion on generating smooth trajectories so far, but it is not clear that this would be the top priority in all applications. In fact, judging from the $\sum_j \ln \alpha_j$ and $\sum_j \ln \beta_j$ values for models with smooth trajectories, we are essentially trading-off uncertainty in the two mappings by incorporating a strong prior preference for uncertainty in reconstruction. Naturally, other types of prior beliefs can be incorporated into the learning process in a similar fashion. For example, if we assume that $\bar{\alpha}$ and $\bar{\beta}$ are not independent, the priors

$$p(\bar{\beta}) \propto \prod_i \beta_i^{-1} \tag{5.3}$$

$$p(\bar{\alpha}|\bar{\beta}) \propto (\prod_i \beta_i - \prod_i \alpha_i)^{-M} \tag{5.4}$$

encourage the two mappings to be similarly uncertain, and therefore exert a similar amount of influence on $\mathbf{X}$.

5.2 Learning hyperparameters with uncertainty

The heuristics discussed in the previous section demonstrates the need for strong priors in the GPDM to learn smooth trajectories in 2D, but it is not clear why a smooth 2D representation of the smooth human motion data cannot be learned purely from the

Algorithm 1 Learning GPDM with MCEM

Require: A number of iterations, T, A number of samples R.

Initialize $\mathbf{X}$ through PCA.

for $t = 1$ to T **do**

$\mathbf{X}_{opt} \Leftarrow$ optimize $\ln p(\mathbf{X}|\mathbf{Y}, \bar{\alpha}^{t-1}, \bar{\beta}^{t-1})$ with respect to $\mathbf{X}$ using SCG.

Initialize HMC with $\mathbf{X}_{opt}$.

$\{\mathbf{X}^{(r)}\}_{r=1}^{R} \Leftarrow$ generate R samples from $p(\mathbf{X}|\mathbf{Y}, \bar{\alpha}^{t-1}, \bar{\beta}^{t-1})$ using HMC.

$\{\bar{\alpha}^t, \bar{\beta}^t\} \Leftarrow$ optimize (5.11) with respect to $\{\bar{\alpha}, \bar{\beta}\}$ using SCG.

for $k = 1$ to d **do**

$w_k \Leftarrow \sqrt{N(\mathbf{y}_{:,k}^T(\frac{1}{R}\sum_{r=1}^{R}\mathbf{K}_Y^{-1})\mathbf{y}_{:,k})^{-1}}$

end for

end for

data set itself. Since our heuristics revolve around influencing hyperparameter values, it is worthwhile revisiting the way these values are estimated in the GPDM. In particular, our MAP learning algorithm ignores latent space uncertainty. Our current estimates for $\bar{\alpha}, \bar{\beta}$ are obtained by maximizing the *log complete likelihood* — $\ln p(\mathbf{Y}, \mathbf{X}|\bar{\alpha}, \bar{\beta})$, which assumes both $\mathbf{Y}$ and $\mathbf{X}$ are observed. Although the values of $\mathbf{X}$ are updated at each iteration with MAP estimates of $\mathbf{X}$, the hyperparameters are still estimated with respect to only one set of values for the hidden states at a time. Instead of learning all of $\mathbf{X}, \bar{\alpha}, \bar{\beta}$ simultaneously, one might consider marginalizing over $\mathbf{X}$ in the same fashion as algorithms for NLDS discussed in Section 2.2.3 to account for the uncertainty in $\mathbf{X}$.

An objective function for parameter estimation with missing data (treating the hidden states as missing data) is the *log incomplete likelihood*

$$\ln p(\mathbf{Y}|\bar{\alpha}, \bar{\beta}) = \ln \int_{\mathbf{X}} p(\mathbf{Y}, \mathbf{X}|\bar{\alpha}, \bar{\beta}) d\mathbf{X}, \tag{5.5}$$

where uncertainties in $\mathbf{X}$ are marginalized out.

The integral above is intractable and has to be approximated numerically. If we

generate fair samples $\{\mathbf{X}^{(r)}\}_{r=1}^{R}$ from the distribution $p(\mathbf{X}|\bar{\alpha})$, the integral in (5.5) can be approximated by:

$$\begin{aligned} \ln \int_{\mathbf{X}} p(\mathbf{Y}, \mathbf{X}|\bar{\alpha}, \bar{\beta}) d\mathbf{X} &= \ln \int_{\mathbf{X}} p(\mathbf{Y}|\mathbf{X}|\bar{\beta}) p(\mathbf{X}|\bar{\alpha}) d\mathbf{X} \\ &\approx \ln \frac{1}{R} \sum_{r=1}^{R} p(\mathbf{Y}|\mathbf{X}^{(r)}, \bar{\beta}), \end{aligned} \tag{5.6}$$

where $p(\mathbf{Y}|\mathbf{X}|\bar{\beta})$ and $p(\mathbf{X}|\bar{\alpha})$ are defined in Chapter 3. We could sample from $p(\mathbf{X}|\bar{\alpha})$ using HMC, and it is easy to show that

$$\frac{\partial}{\partial \bar{\beta}} \ln p(\mathbf{Y}|\bar{\alpha}, \bar{\beta}) \approx \frac{1}{\sum_{r=1}^{R} p(\mathbf{Y}|\mathbf{X}^{(r)}, \bar{\beta})} \sum_{r=1}^{R} \frac{\partial}{\partial \bar{\beta}} p(\mathbf{Y}|\mathbf{X}^{(r)}, \bar{\beta}). \tag{5.7}$$

However, we need to evaluate the exponential function in (5.7), which tends to create numerical issues. An additional problem is that we cannot find the derivative with respect to $\bar{\alpha}$ analytically, since the samples depends on $\bar{\alpha}$. Moreover, the HMC sampling procedure is expensive and makes finite-difference approximations impractical.

5.2.1 A Monte Carlo EM algorithm for GPDM

As discussed in Chapter 2, the EM algorithm is a well-known solution to the problem of parameter estimation with latent space uncertainty. The basic idea of EM is to maximize a lower bound of the log incomplete likelihood instead of the quantity itself. More precisely, we seek to maximize the *free energy* [36]

$$F(q, \{\bar{\alpha}, \bar{\beta}\}) = \int_{\mathbf{X}} q(\mathbf{X}|\mathbf{Y}) \ln p(\mathbf{Y}, \mathbf{X}|\bar{\alpha}, \bar{\beta}) d\mathbf{X} - \int_{\mathbf{X}} q(\mathbf{X}|\mathbf{Y}) \ln q(\mathbf{X}|\mathbf{Y}) d\mathbf{X}, \tag{5.8}$$

which is a lower bound of the log incomplete likelihood for any distribution q over $\mathbf{X}$. The first term on the right hand side is the *expected complete log likelihood*, while the second term is called the *entropy*.

In the E-step of the i^{th} iteration, we compute the distribution $q^{i+1}(\mathbf{X}|\mathbf{Y})$ that maximizes F given $\{\bar{\alpha}^i, \bar{\beta}^i\}$. It can be shown that the optimal solution is given by

$$q^{i+1}(\mathbf{X}|\mathbf{Y}) = p(\mathbf{X}|\mathbf{Y}, \bar{\alpha}^i, \bar{\beta}^i), \tag{5.9}$$

the posterior distribution given the current estimation of hyperparameters.

In the M-step, we seek a set of parameters $\{\bar{\alpha}^{i+1}, \bar{\beta}^{i+1}\}$, that maximizes F given $q^{i+1}(\mathbf{X}|\mathbf{Y})$. Since the entropy term is independent of parameters, we only need to maximize the expected complete log likelihood

$$\begin{aligned}\mathcal{L}_{\mathcal{E}} &= \int_{\mathbf{X}} q(\mathbf{X}|\mathbf{Y}) \ln p(\mathbf{Y}, \mathbf{X}|\bar{\alpha}, \bar{\beta}) d\mathbf{X} \\ &= \int_{\mathbf{X}} p(\mathbf{X}|\mathbf{Y}, \bar{\alpha}^i, \bar{\beta}^i) \ln p(\mathbf{Y}, \mathbf{X}|\bar{\alpha}, \bar{\beta}) d\mathbf{X}. \end{aligned} \tag{5.10}$$

For our problem, we can sample from the required posterior in the E-step using (3.11) with HMC and use the samples to approximate the integral

$$\int_{\mathbf{X}} p(\mathbf{X}|\mathbf{Y}, \bar{\alpha}^i, \bar{\beta}^i) \ln p(\mathbf{Y}, \mathbf{X}|\bar{\alpha}, \bar{\beta}) d\mathbf{X} \approx \frac{1}{R} \sum_{r=1}^{R} \ln p(\mathbf{Y}, \mathbf{X}^{(r)}|\bar{\alpha}, \bar{\beta}), \tag{5.11}$$

where $\{\mathbf{X}^{(r)}\}_{r=1}^{R}$ are samples from $p(\mathbf{X}|\mathbf{Y}, \bar{\alpha}^i, \bar{\beta}^i)$. The derivative with respect to the hyperparameters is given by

$$\begin{aligned}\frac{\partial \mathcal{L}_{\mathcal{E}}}{\partial \bar{\beta}} &\approx \frac{\partial}{\partial \bar{\beta}} \frac{1}{R} \sum_{r=1}^{R} \ln p(\mathbf{Y}, \mathbf{X}^{(r)}|\bar{\alpha}, \bar{\beta}) \\ &= \frac{1}{R} \sum_{r=1}^{R} \frac{\partial}{\partial \bar{\beta}} \ln p(\mathbf{Y}, \mathbf{X}^{(r)}|\bar{\alpha}, \bar{\beta}) \end{aligned} \tag{5.12}$$

$$\frac{\partial \mathcal{L}_{\mathcal{E}}}{\partial \bar{\alpha}} \approx \frac{1}{R} \sum_{r=1}^{R} \frac{\partial}{\partial \bar{\alpha}} \ln p(\mathbf{Y}, \mathbf{X}^{(r)}|\bar{\alpha}, \bar{\beta}). \tag{5.13}$$

The approximations are simply sums of the derivatives of the complete log likelihood, which we used for optimizing (3.11).

It should be noted that the samples $\{\mathbf{X}^{(r)}\}_{r=1}^{R}$ are drawn from the posterior given the hyperparameters in the previous iteration. Since we only seek to maximize the expected complete log likelihood in the M-step, the samples do not need to be altered as we evaluate different values of $\bar{\alpha}$ and $\bar{\beta}$. This is a significant difference between maximizing the free energy and maximizing the incomplete log likelihood (5.5), as we cannot easily approximate the gradients of the latter. Algorithm 1 describes GPDM learning with uncertain $\mathbf{X}$ in pseudocode.

The variation of EM algorithm used in this section is generally referred to as Monte Carlo EM (MCEM). Since the expected complete log likelihood is not computed exactly in the M-step, MCEM does not guarantee an increase in the incomplete log likelihood in every iteration. On the other hand, it is still an improvement over not accounting for uncertainties in $\mathbf{X}$ at all, and should give us an idea of what GPDM hyperparameter estimations look like when optimized against an approximation of the correct objective function.

5.2.2 Preliminary results

The HMC sampling step required in each iteration of MCEM is computationally very expensive. The dimensionality of the sampling space is the number of training data times the number of latent space dimensions, typically over 1000. Furthermore, the HMC parameters such as the number of leapfrog steps and the step size, need to be set for each EM iteration, as the posterior distribution changes according to the hyperparameters. We leave a proper implementation of MCEM for GPDM as future work, but examine some preliminary results obtained from very crude sampling procedures in the rest of the section. It should be noted that the reliability of observations from results in this section depends on having samples that approximate the posterior distribution well, which we make no effort to ensure here.

Figure 5.7(a) shows a GPDM of walking learned with MCEM. We run the algorithm for 50 iterations, with 10 HMC samples per iteration. We initialize the HMC sampler with an optimal trajectory with no burn-in samples, and adjust the parameters so that the rejection rate is around 20% for the initial posterior distribution. The hyperparameters are initialized all to ones. The inverse variance plot is very similar to the one learned from maximizing (5.2) with $M = 1000$. In both cases, the region with low prediction variance is ring-shaped, as the variance smoothly increases away from the data points. Moreover, we can see from the grey scale plots that the prediction variance in the model

learned from MCEM falls off more quickly as one moves further from the data.

The dynamic simulation in Figure 5.7(b) learned using the MCEM algorithm look surprisingly similar to Figures 4.6(b) and 4.7(b), where linear kernels are used. Although the simulation results in a circular spiral towards the origin in both cases, the cause is very different. In the case with linear kernels, the placements of latent space coordinates are fairly noisy, which would describe a relatively complex dynamics mapping. However, the expressive power of the linear kernel is limited and underfits the data as a result. We can see that Figures 4.6(a) and 4.7(a) shows similar latent space placements, but with more complex dynamic mappings learned from nonlinear kernels. On the other hand, the latent space coordinates learned using MCEM do resemble parts of a smooth spiral pointing towards the centre of the ring. The dynamics simulation is not a result of underfitting, but an accurate reflection of the data itself. Despite the lack of a limit cycle, the first 200 frames of animation generated from this model look smooth and are comparable to animations corresponding to Figure 5.5(b).

The estimated hyperparameter values are listed in Table 5.1. Unlike the model learned with $M = 1000$, the MCEM model learns both the reconstruction and dynamics mapping with high certainty. Moreover, the values $\sum_j \ln \alpha_j$ and $\sum_j \ln \beta_j$ are similar, which means the optimization of latent space coordinates does not heavily favour one mapping over another.

Applying Algorithm 1 to the golf data set results in a curious latent space trajectory (Figure 5.8). The poses at the beginning of the swing and at the end of the swing are separated into two groups in latent space. The simulated dynamics (Figure 5.8b) consists of a large jump in the middle of the swing. Although poses during the swing differs from each other more than the rest of the poses due to rapid movement, it seems that points should increase their separation more gradually than what is observed in the result. However, the simulation does not contain poses that are far away from the training data. The animation of the golf swing corresponding to the simulation looks visually identical

to the simulations obtained from the 3D GPDM golf models.

One possible explanation for the latent space trajectory is our crude approximation to the posterior distribution. A closer examination reveals that the set of parameters for HMC that results in 20% rejection rate for the first few iterations generates nothing but rejections later on in the learning process. As a result, for a significant number of iterations (last 20 or so), the algorithm reduces to coordinate descent on our original objective function in Chapter 4.

The only way to ensure a reasonable number of samples are generated at every iteration is to adapt the HMC parameters according to the posterior at every iteration. Figure 5.9 shows the evolution of the posterior distribution during Algorithm 1. To obtain a slightly better approximation for the posterior (but still with 10 samples per iteration), we use a down-sampled version of the golf data to shorten the computation time, the HMC sampler is hand-tuned at each iteration to achieve a rejection rate of approximately 20%. We see that the samples deviate less and less from the mean as the number of iteration increases, and that the mean trajectory is not separated into two groups at the 50th iteration.

Although the idea of marginalizing out $\mathbf{X}$ while estimating $\bar{\alpha}, \bar{\beta}$ is conceptually appealing, the computational overhead becomes unbearable for large data sets. The models described in this section, which only use 10 HMC samples per iteration, take over 24 hours to learn. The fact that our preliminary results seem to give balanced values of $\sum_j \ln \alpha_j$ and $\sum_j \ln \beta_j$ suggest (5.3) and (5.4) could be sensible priors to use if we have to estimate both $\mathbf{X}$ and the hyperparameters. However, the M value would still be dependent on the data set, and could be troublesome to set.

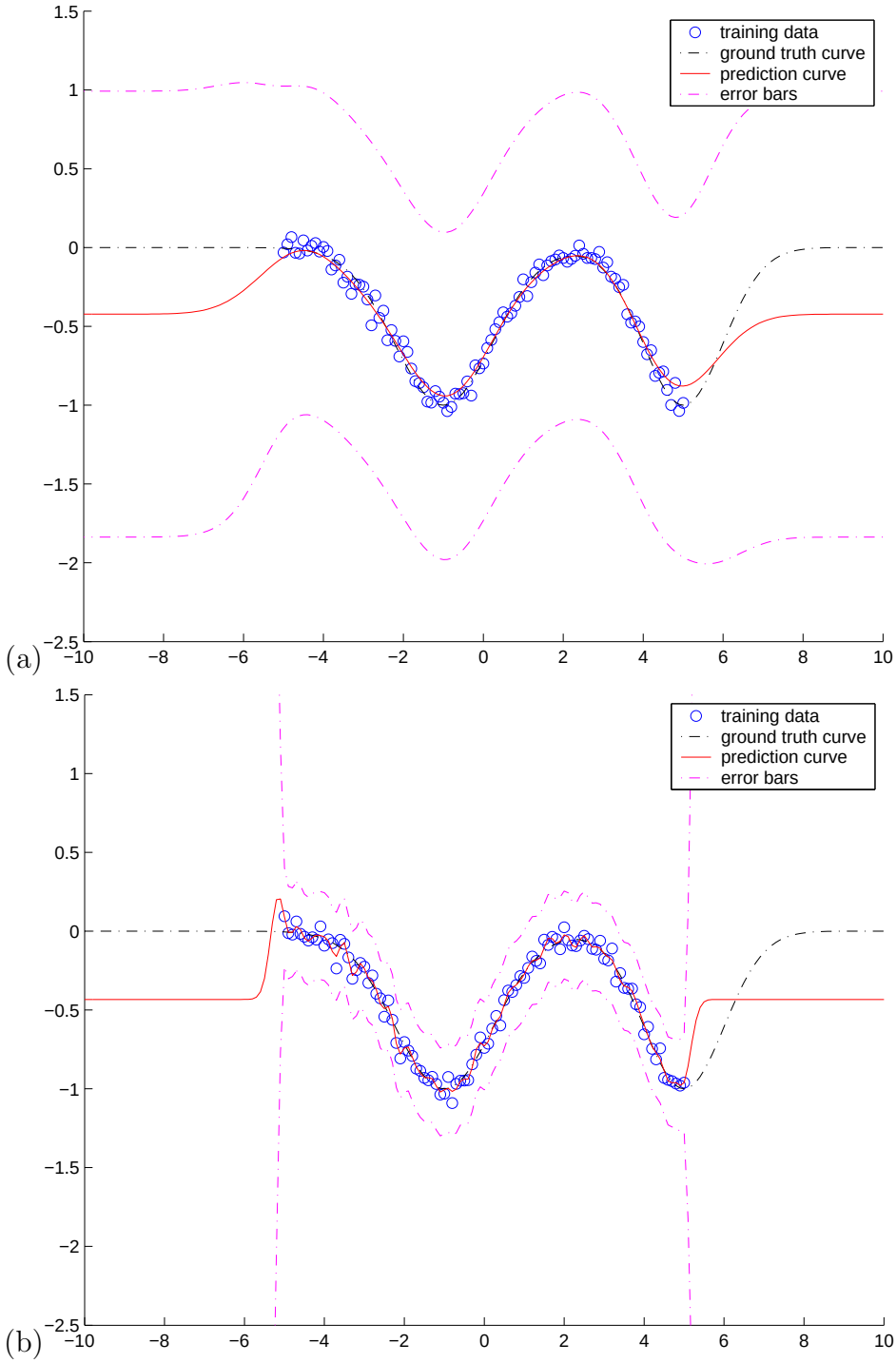


Figure 5.1: Effect of hyperparameter settings in GP regression (a) A conservative hyperparameter setting leads to large uncertainty in prediction. (b) The same data set with a more aggressive hyperparameter setting, which shows overfitting.

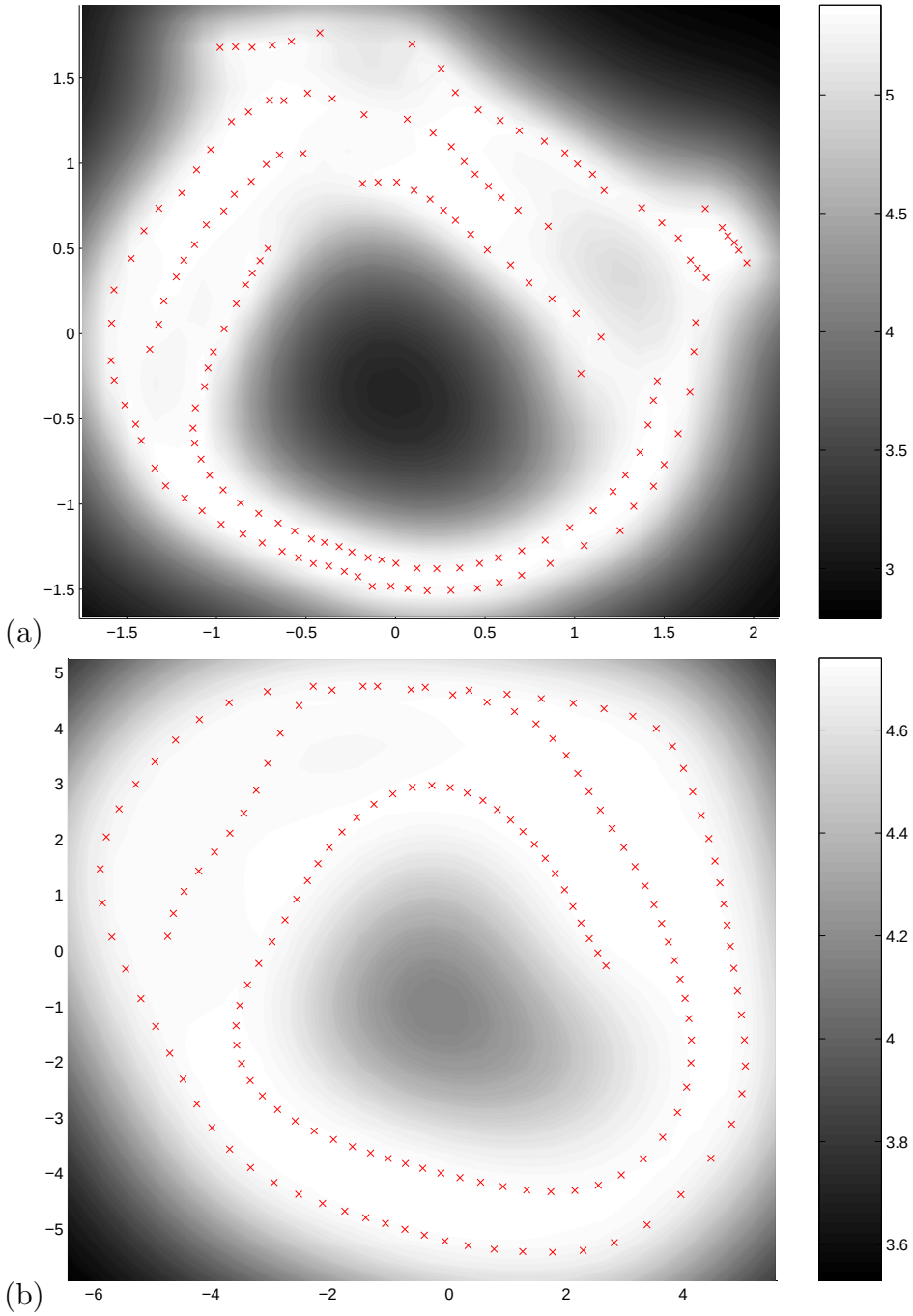


Figure 5.2: Inverse reconstruction variance plot of walk data in 2D with RBF kernel (a) Latent space trajectory given by $M = 1$, notice large discontinuities and variations on reconstruction variance. (b) Latent space trajectory given by $M = 1000$.

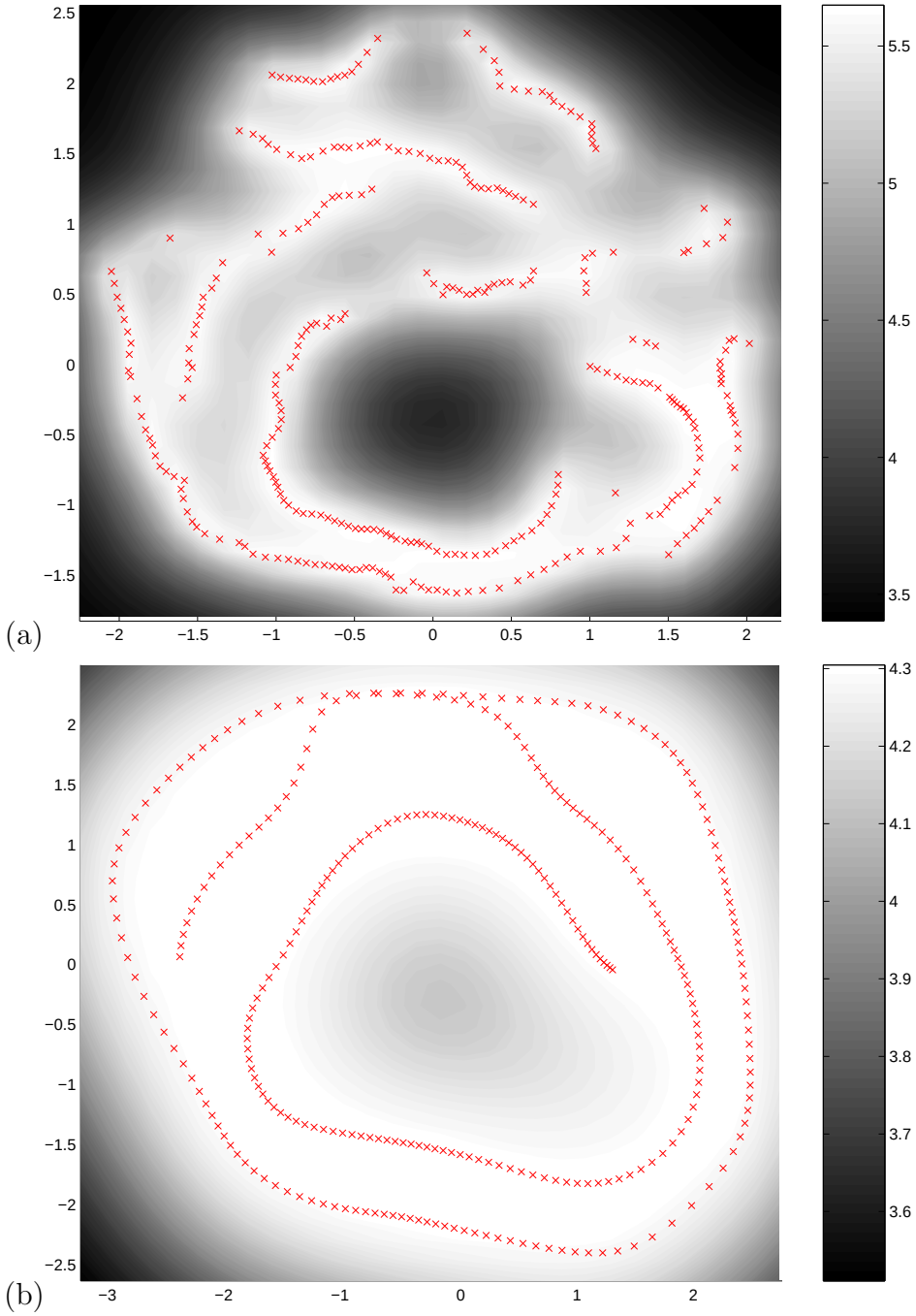


Figure 5.3: Inverse reconstruction variance plot of 120Hz walk data in 2D with RBF kernel (a) Setting $M = 1000$ does not remove discontinuities in data. (b) Latent space trajectory given by $M = 3000$.

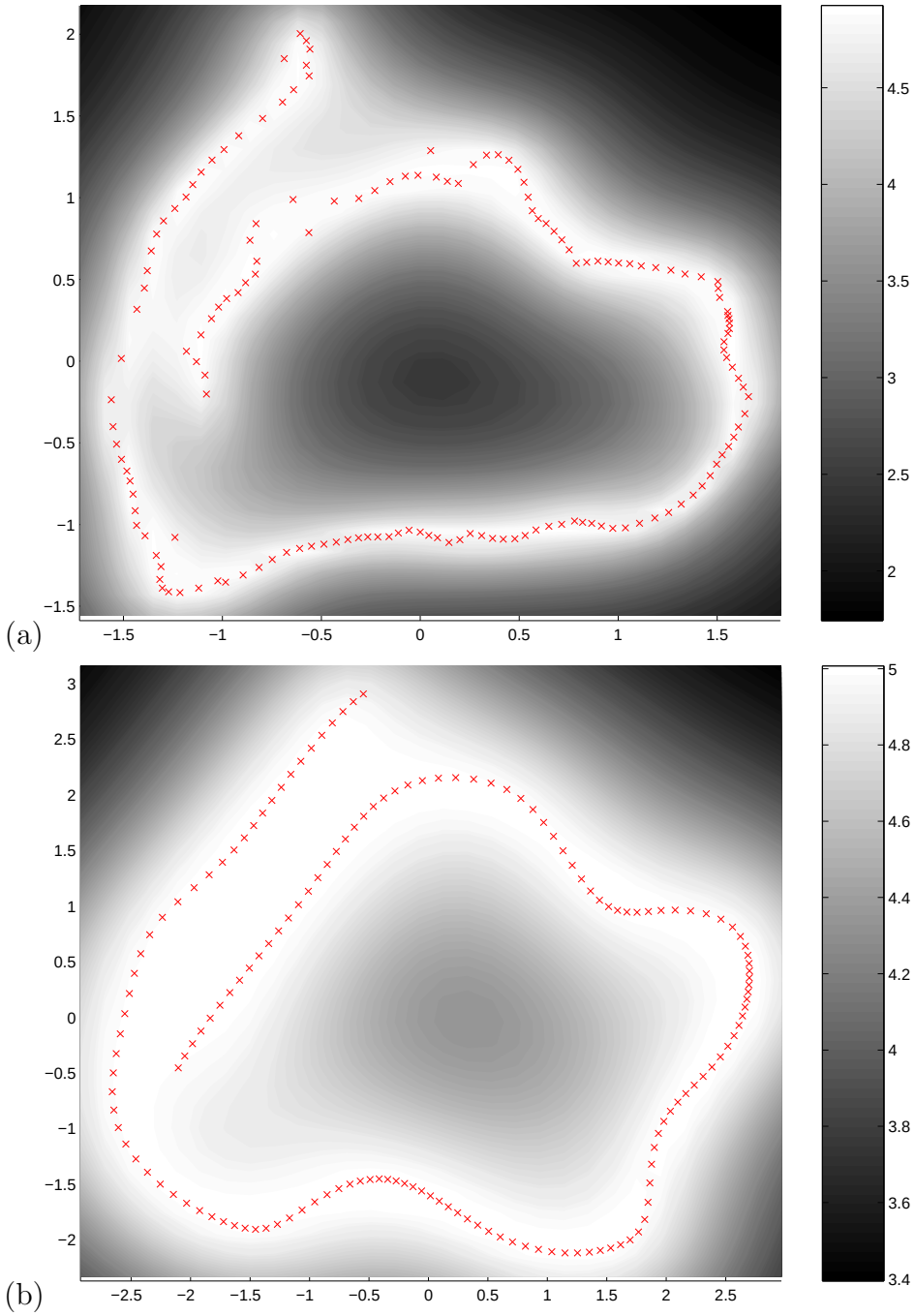


Figure 5.4: Inverse reconstruction variance plot of the first 157 frames of the 120Hz walk data (316 frames in total). (a) Latent space trajectories given by $M = 1$. (b) Latent space trajectory given by $M = 1000$.

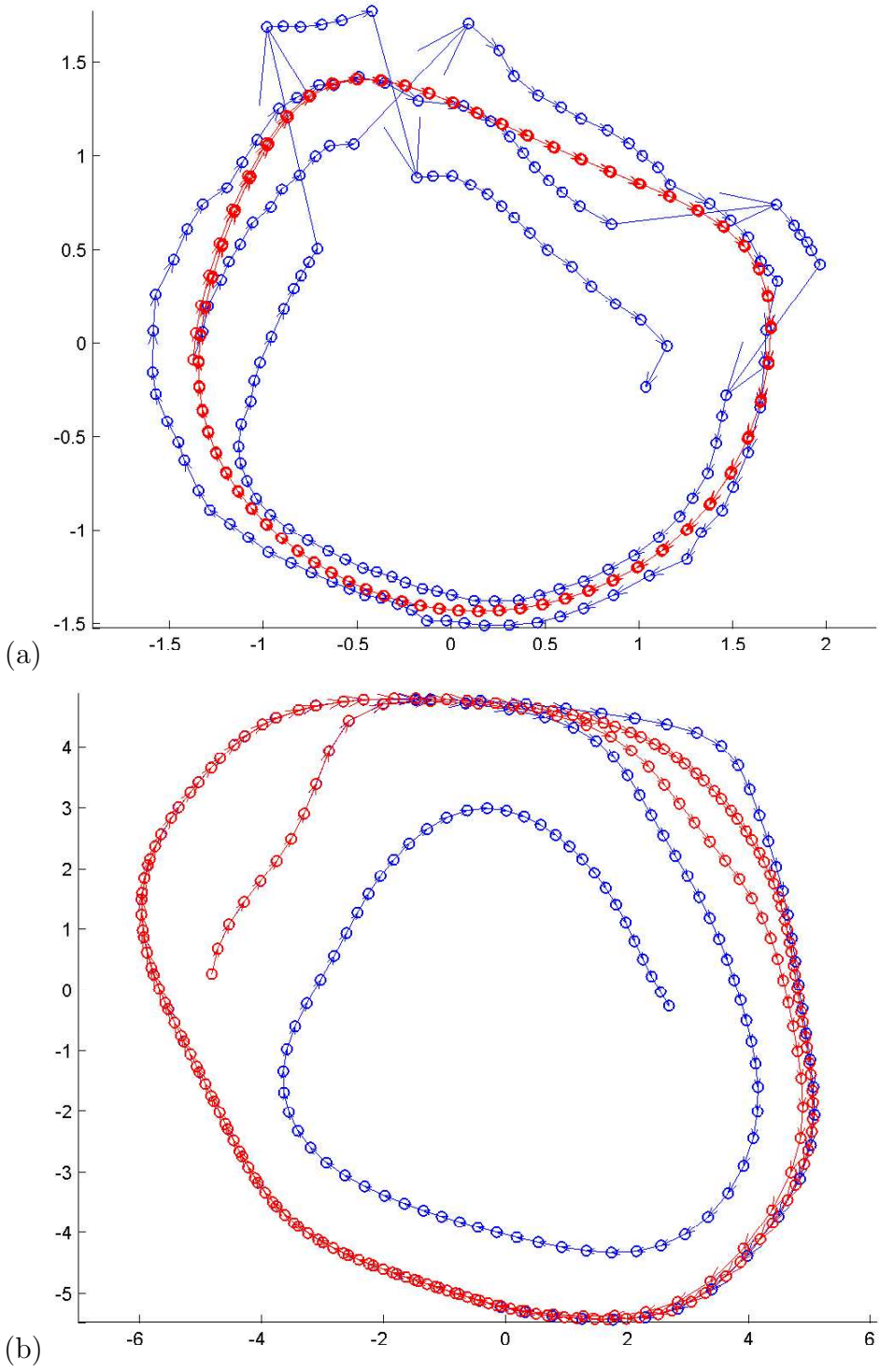


Figure 5.5: Simulations from walker models. (a) Latent space trajectories given by $M = 1$. (b) Latent space trajectory given by $M = 1000$, the simulations follows training data very closely.

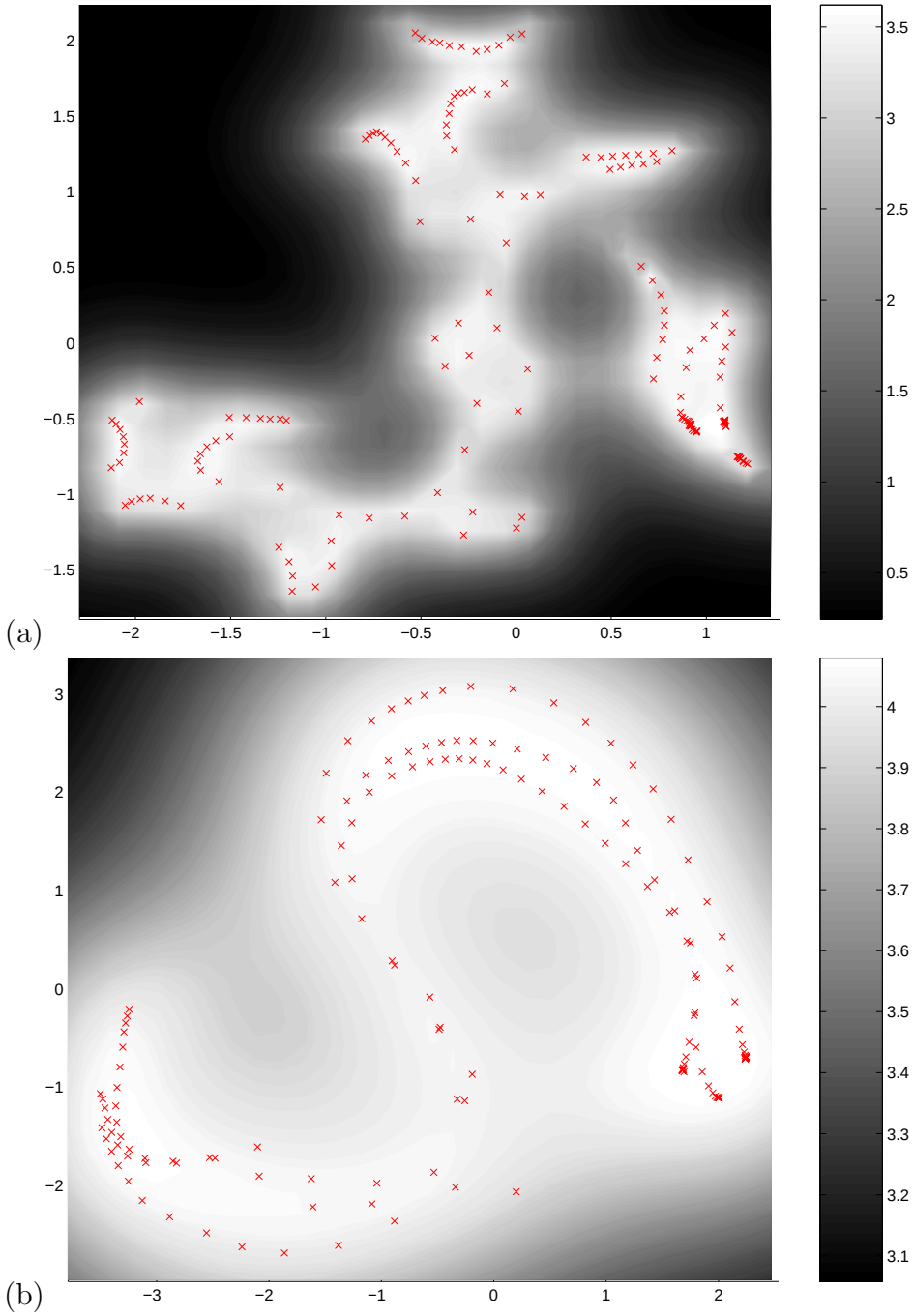


Figure 5.6: Inverse reconstruction variance plot of three golf swing data in 2D with RBF kernel. (a) Latent space trajectories given by $M = 1$. (b) Latent space trajectory given by $M = 1500$.

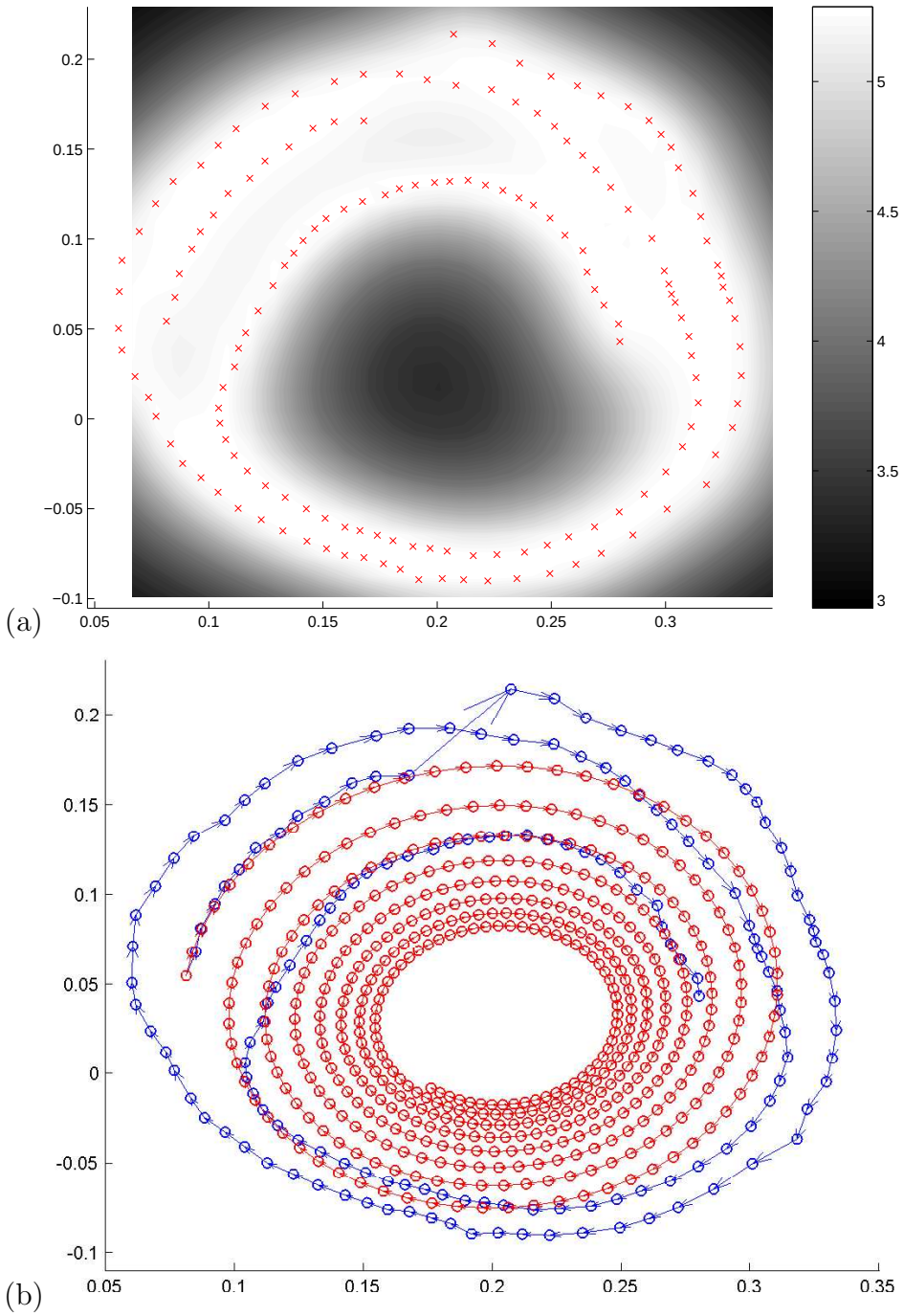


Figure 5.7: 2D GPDM walking model with RBF dynamics learned using MCEM. (a) Inverse variance plot with a smooth trajectory. (b) Simulation of dynamics, note that the simulation does not lead to a limit cycle.

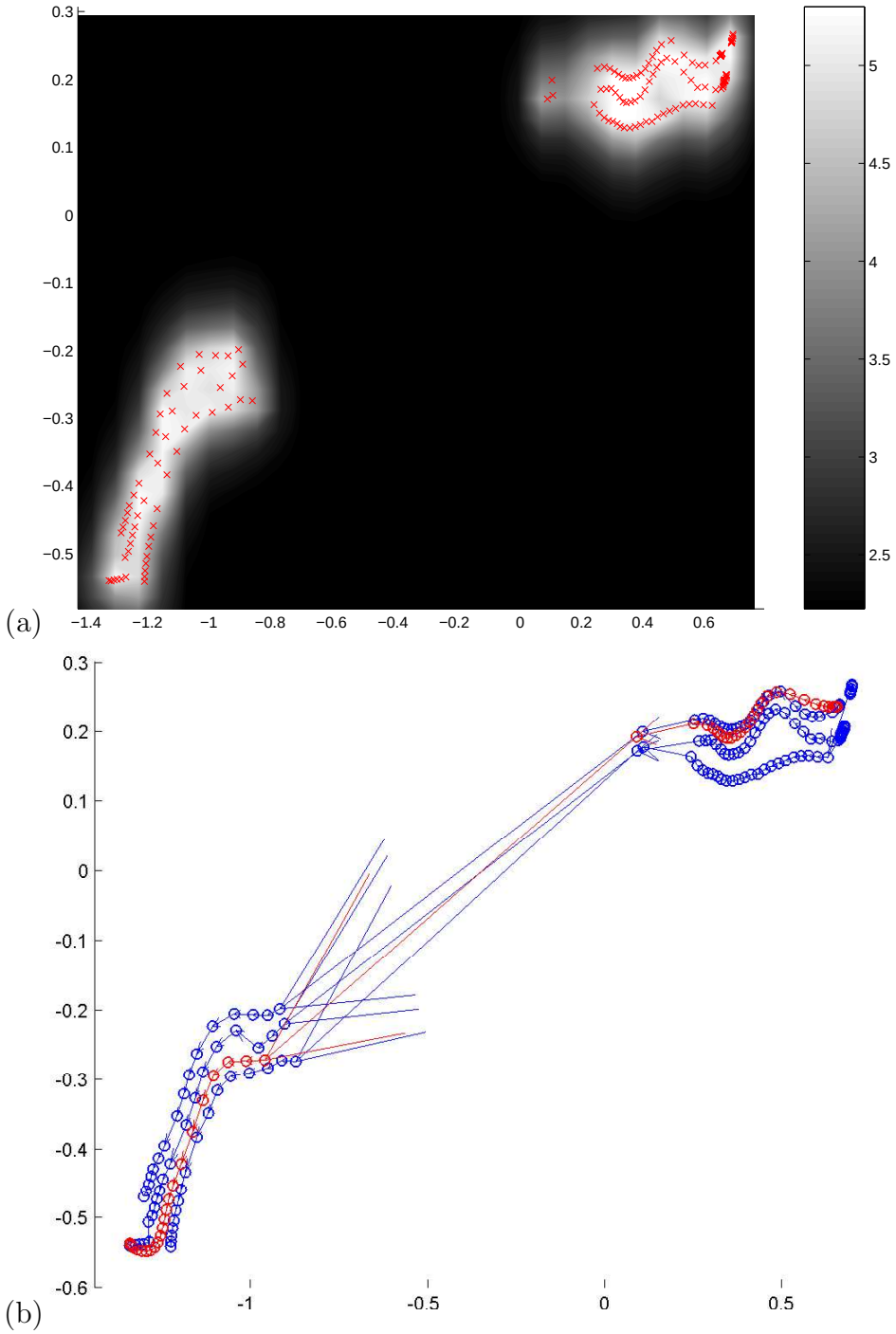
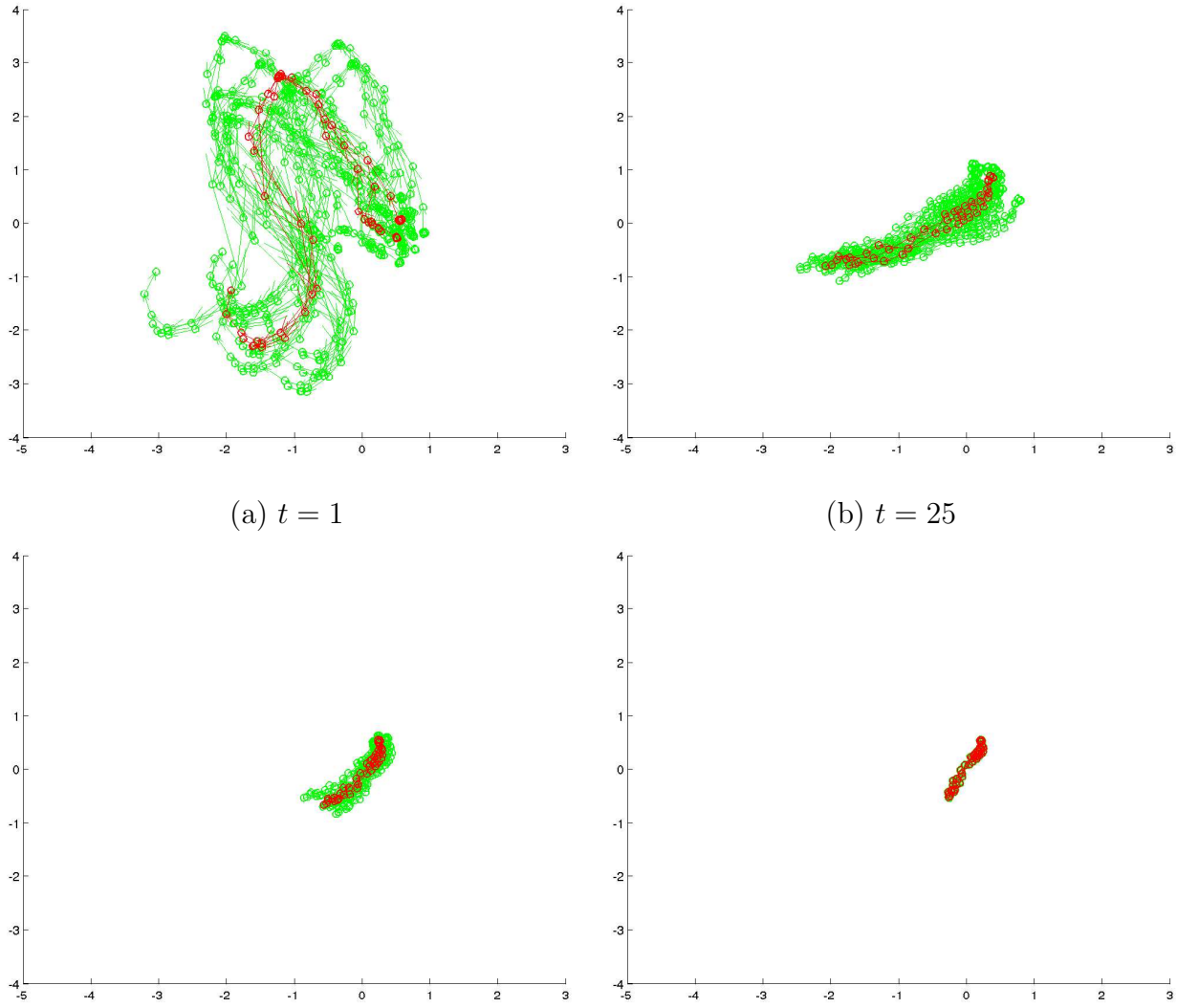


Figure 5.8: 2D GPDM golf model with RBF dynamics learned using MCEM. (a) Inverse variance plot. (b) Simulation of dynamics.



(a) $t = 1$ (b) $t = 25$

(c) $t = 40$ (d) $t = 50$

Figure 5.9: Samples (green trajectories) from the posterior of a 2D GPDM golf model during the t^{th} iteration of Algorithm 1. The red trajectories represents the posterior mean.

Chapter 6

Discussion and extensions

One of the main strengths of the GPDM model is the ability to generalize well from small datasets. Conversely, performance is a major issue in applying GP methods to larger datasets. Previous approaches prune uninformative vectors from the training data [25]. This is not straightforward when learning a GPDM, however, because each timestep is highly correlated with the steps before and after it. For example, if we hold $\mathbf{x}_t$ fixed during optimization, then it is unlikely that the optimizer will make much adjustment to $\mathbf{x}_{t+1}$ or $\mathbf{x}_{t-1}$. The use of higher-order features [48, 34] provides a possible solution to this problem. Specifically, consider a dynamical model of the form $\mathbf{v}_t = f(\mathbf{x}_{t-1}, \mathbf{v}_{t-1})$. Since adjacent time-steps are related only by the velocity $\mathbf{v}_t \approx (\mathbf{x}_t - \mathbf{x}_{t-1})/\Delta t$, we can handle irregularly-sampled data points by adjusting the timestep Δt, possibly using a different Δt at each step. Another intriguing approach for using GPDM with large datasets is by incorporating a hierarchical mixture of GPs [43].

A number of further extensions to the GPDM model are possible. It would be straightforward to model dynamics $f(\mathbf{x}_t, \mathbf{u}_t)$ by including a control signal $\mathbf{u}_t$ in the kernel function. The kernels would have to be augmented with an extra hyperparameter that weighs the contribution of the control signal. The prior distribution over latent variables (3.9) is then conditioned on the $\mathbf{u}_t$ terms. The dynamics function then defines a mapping from

$\{\mathbf{x}_t, \mathbf{u}_t\}$ to $\mathbf{x}_{t+1}$, which could potentially be incorporated into existing frameworks for GPs in reinforcement learning [40, 13].

In physical simulations, one typically uses dynamics to generate accelerations (forces). State is represented by position and velocity, and updated by numerical integration. One question is whether this numerical integration should be made explicit in the model (i.e., using velocity instead of position as outputs of the dynamics function), or whether it should be learned as part of the model. While an explicit representation might help the model and prevent perverse physical models, it might also prevent more expressive representations, particularly when the dynamics model is inaccurate. Learning the integration might help improve numerical accuracy in simulation, avoiding unlikely sequences that would normally arise due to numerical drift.

It would also be interesting to explore other means of integrating out $\mathbf{X}$ for hyperparameter estimation. The MCEM algorithm currently used is slow and impractical for large data sets, more sophisticated methods that makes use of the importance sampling theory can potentially speed up the computation [23].

For applications in animation, animator constraints could be specified in pose space to synthesize entire motion sequences by constrained optimization. It would be a generalization of the interactive posing application presented by Grochow et al. [18]. However, speed would certainly be an issue due to the size of the search space. In human tracking, GPDM could be used to predict future poses, but care must be taken to avoid overfitting. Possible solutions include accounting for latent space uncertainty, model annealing [18] or learning with a large data set. Naturally, the latter hinges on overcoming the efficiency issues mentioned earlier.

Appendix A

Prior Distribution of Latent Variables

To define GPDM, we need an expression for the prior distribution over latent variables $p(\mathbf{X}|\bar{\alpha})$. We first show that the joint distribution of $\mathbf{X}$ can be written as a product of Gaussians by the Markov assumption, the integration is then a special case of the marginalization in the GPLVM. Equation (3.9) can then be derived by appropriate substitution of variables.

Since each output dimension is assumed to be independent, we can write

$$\begin{aligned} p(\mathbf{X}|\bar{\alpha}) &= p(\mathbf{x}_1)p(\hat{\mathbf{X}}|\bar{\alpha}) \\ &= \prod_{q=1}^{d} p(\mathbf{x}_{1,q})p(\hat{\mathbf{x}}_{:,q}|\bar{\alpha}), \end{aligned} \tag{A.1}$$

where $\hat{\mathbf{X}} = \mathbf{X}_{out}$.

Applying rules of probability for each dimension, we have

$$\begin{aligned} p(\hat{\mathbf{x}}_{:,q}|\bar{\alpha}) &= p(\mathbf{x}_{2,q} \cdots \mathbf{x}_{N,q}|\bar{\alpha}) \\ &= \int_{\mathbf{a}_q} p(\mathbf{x}_{2,q} \cdots \mathbf{x}_{N,q}, \mathbf{a}_q|\bar{\alpha}) d\mathbf{a}_q \\ &= \int_{\mathbf{a}_q} p(\mathbf{x}_{2,q} \cdots \mathbf{x}_{N,q}|\mathbf{a}_q, \bar{\alpha}) p(\mathbf{a}_q) d\mathbf{a}_q. \end{aligned} \tag{A.2}$$

By the Markov assumption,

$$\begin{aligned} p(\hat{\mathbf{x}}_{:,q}|\bar{\alpha}) &= \int_{\mathbf{a}_q} \prod_{t=1}^{N-1} p(\mathbf{x}_{t+1,q}|\mathbf{x}_{t,:}, \mathbf{a}_q, \bar{\alpha}) p(\mathbf{a}_q) d\mathbf{a}_q \\ &= \int_{\mathbf{a}_q} \left(\prod_{t=1}^{N-1} \frac{1}{\sigma_X \sqrt{2\pi}} \exp\left(\frac{-\|\mathbf{x}_{t+1,q} - \mathbf{x}_{t,:}\mathbf{a}_q\|^2}{2\sigma_X^2} \right) \right) \frac{1}{\sqrt{2\pi}^d} \exp\left(\frac{-\|\mathbf{a}_q\|^2}{2} \right) d\mathbf{a}_q, \end{aligned}$$

where σ_X is the variance of the noise term in the dynamics mapping.

The above integral is a special case of the integral for parameter marginalization in the GPLVM [25, 26]:

$$\begin{aligned} p(\mathbf{y}_{:,q}|\mathbf{X}, \bar{\beta}) &= \int_{\mathbf{b}_q} \prod_{t=1}^{N} p(\mathbf{y}_{t,q}|\mathbf{x}_{t,:}, \mathbf{b}_q, \bar{\beta}) p(\mathbf{b}_q) d\mathbf{b}_q \\ &= \int_{\mathbf{b}_q} \left(\prod_{t=1}^{N} \frac{1}{\sigma_Y \sqrt{2\pi}} \exp\left(\frac{-\|\mathbf{y}_{t,q} - \mathbf{x}_{t,:}\mathbf{b}_q\|^2}{2\sigma_Y^2} \right) \right) \frac{1}{\sqrt{2\pi}^d} \exp\left(\frac{-\|\mathbf{b}_q\|^2}{2} \right) d\mathbf{b}_q \\ &= \frac{1}{\sqrt{(2\pi)^N |\mathbf{K}|}} \exp\left(-\frac{1}{2} \mathbf{y}_{:,q} \mathbf{K}^{-1} \mathbf{y}_{:,q}^T \right), \end{aligned} \tag{A.3}$$

where $\mathbf{K} = \mathbf{X}\mathbf{X}^T + \sigma_Y \mathbf{I}$.

Substituting $\mathbf{Y}, \mathbf{X}$ with $\mathbf{X}_{out}, \mathbf{X}_{in}$ in (A.3) does not change the integration operator, therefore

$$\begin{aligned} p(\hat{\mathbf{x}}_{:,q}|\bar{\alpha}) &= \int_{\mathbf{a}_q} \left(\prod_{t=1}^{N-1} \frac{1}{\sigma_X \sqrt{2\pi}} \exp\left(\frac{-\|\mathbf{x}_{t+1,q} - \mathbf{x}_{t,:}\mathbf{a}_q\|^2}{2\sigma_X^2} \right) \right) \frac{1}{\sqrt{2\pi}^d} \exp\left(\frac{-\|\mathbf{a}_q\|^2}{2} \right) d\mathbf{a}_q, \\ &= \frac{1}{\sqrt{(2\pi)^{N-1} |\mathbf{K}_X|}} \exp\left(-\frac{1}{2} \hat{\mathbf{x}}_{:,q} \mathbf{K}_X^{-1} \hat{\mathbf{x}}_{:,q}^T \right), \end{aligned} \tag{A.4}$$

where $\mathbf{K}_X = \mathbf{X}_{in}\mathbf{X}_{in}^T + \sigma_X \mathbf{I}$.

Combining (A.1) and (A.4) we get

$$p(\mathbf{X} \mid \bar{\alpha}) = p(\mathbf{x}_1) \frac{1}{\sqrt{(2\pi)^{(N-1)d} |\mathbf{K}_X|^d}} \exp\left(-\frac{1}{2} \mathrm{tr}\left(\mathbf{K}_X^{-1} \mathbf{X}_{out} \mathbf{X}_{out}^T \right) \right). \tag{A.5}$$

Finally, following [25, 26], we introduce nonlinearity to the dynamics mapping by replacing $\mathbf{K}_X$ with RBF or linear+RBF kernels.

Appendix B

Gradients

$$\frac{\partial \mathcal{L}}{\partial \mathbf{X}} = \frac{\partial \mathcal{L}}{\partial \mathbf{K}_Y} \circ \frac{\partial \mathbf{K}_Y}{\partial \mathbf{X}} + \frac{\partial \mathcal{L}}{\partial \mathbf{K}_X} \circ \frac{\partial \mathbf{K}_X}{\partial \mathbf{X}} - \mathbf{K}_X^{-1}\mathbf{X}_{out} \circ \frac{\partial \mathbf{X}_{out}}{\partial \mathbf{X}} \tag{B.1}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{K}_Y} = \frac{1}{2}\mathbf{K}_Y^{-1}\mathbf{W}\mathbf{Y}\mathbf{Y}^T\mathbf{W}^T\mathbf{K}_Y^{-1} - \frac{D}{2}\mathbf{K}_Y^{-1} \tag{B.2}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{K}_X} = \frac{1}{2}\mathbf{K}_X^{-1}\mathbf{X}_{out}\mathbf{X}_{out}^T\mathbf{K}_X^{-1} - \frac{d}{2}\mathbf{K}_X^{-1} \tag{B.3}$$

$$\frac{\partial k_Y(\mathbf{x}, \mathbf{x}')}{\partial \mathbf{x}} = -\beta_2(\mathbf{x} - \mathbf{x}')k_Y(\mathbf{x}, \mathbf{x}') \tag{B.4}$$

$$\frac{\partial k_Y(\mathbf{x}, \mathbf{x}')}{\partial \beta_1} = \exp\left(-\frac{\beta_2}{2}||\mathbf{x} - \mathbf{x}'||^2\right) \tag{B.5}$$

$$\frac{\partial k_Y(\mathbf{x}, \mathbf{x}')}{\partial \beta_2} = -\frac{\beta_1}{2}\|\mathbf{x} - \mathbf{x}'\|^2 \exp\left(-\frac{\beta_2}{2}||\mathbf{x} - \mathbf{x}'||^2\right) \tag{B.6}$$

$$\frac{\partial k_Y(\mathbf{x}, \mathbf{x}')}{\partial \beta_3} = -\frac{\delta_{\mathbf{x},\mathbf{x}'}}{\beta_3^2} \tag{B.7}$$

$$\frac{\partial k_X(\mathbf{x}, \mathbf{x}')}{\partial \mathbf{x}} = -\alpha_1\alpha_2(\mathbf{x} - \mathbf{x}') \exp\left(-\frac{\alpha_2}{2}||\mathbf{x} - \mathbf{x}'||^2\right) + \alpha_3\mathbf{x}' \tag{B.8}$$

$$\frac{\partial k_X(\mathbf{x}, \mathbf{x}')}{\partial \alpha_1} = \exp\left(-\frac{\alpha_2}{2}||\mathbf{x} - \mathbf{x}'||^2\right) \tag{B.9}$$

$$\frac{\partial k_X(\mathbf{x}, \mathbf{x}')}{\partial \alpha_2} = -\frac{\alpha_1}{2}\|\mathbf{x} - \mathbf{x}'\|^2 \exp\left(-\frac{\alpha_2}{2}||\mathbf{x} - \mathbf{x}'||^2\right) \tag{B.10}$$

$$\frac{\partial k_X(\mathbf{x}, \mathbf{x}')}{\partial \alpha_3} = \mathbf{x}^T\mathbf{x}' \tag{B.11}$$

$$\frac{\partial k_X(\mathbf{x}, \mathbf{x}')}{\partial \alpha_4} = -\frac{\delta_{\mathbf{x},\mathbf{x}'}}{\alpha_4^2} \tag{B.12}$$

Bibliography

[1] Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors. *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*. MIT Press, 2003.

[2] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *NIPS*, pages 585–591. MIT Press, 2001.

[3] Yoshua Bengio, Jean-François Paiement, Pascal Vincent, Olivier Delalleau, Nicolas Le Roux, and Marie Ouimet. Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering. In Thrun et al. [50].

[4] Christopher M. Bishop, Markus Svensén, and Christopher K. I. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998.

[5] Matthew Brand and Aaron Hertzmann. Style machines. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 183–192, July 2000.

[6] Carla E. Brodley, editor. *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*. ACM, 2004.

[7] J. Carpenter, P. Clifford, and P. Fernhead. An improved particle filter for non-linear problems. In *IEE Proceedings on Radar, Sonar and Navigation*, 1999.

[8] Kiam Choo and David J. Fleet. People tracking using hybrid Monte Carlo filtering. In *ICCV*, pages 321–328, 2001.

[9] Vin de Silva and Joshua B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In Becker et al. [1], pages 705–712.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[11] Gianfranco Doretto, Alessandro Chiuso, Ying Nian Wu, and Stefano Soatto. Dynamic textures. *International Journal of Computer Vision*, 51(2):91–109, 2003.

[12] Ahmed Elgammal and Chan-Su Lee. Inferring 3D body pose from silhouettes using activity manifold learning. In *CVPR (2)*, pages 681–688, 2004.

[13] Yaakov Engel, Shie Mannor, and Ron Meir. Bayes meets Bellman: The Gaussian process approach to temporal difference learning. In Tom Fawcett and Nina Mishra, editors, *ICML*, pages 154–161. AAAI Press, 2003.

[14] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[15] Zoubin Ghahramani and Geoffrey E. Hinton. Parameter estimation for linear dynamical systems. Technical Report CRG-TR-96-2, Department of Computer Science, University of Toronto, February 1996.

[16] Zoubin Ghahramani and Sam T. Roweis. Learning nonlinear dynamical systems using an EM algorithm. In Michael J. Kearns, Sara A. Solla, and David A. Cohn, editors, *NIPS*, pages 431–437. The MIT Press, 1998.

[17] Agathe Girard, Carl Edward Rasmussen, Joaquin Quiñonero Candela, and Roderick Murray-Smith. Gaussian process priors with uncertain inputs - application to multiple-step ahead time series forecasting. In Becker et al. [1], pages 529–536.

[18] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. *ACM Transactions on Graphics*, 23(3):522–531, August 2004.

[19] Nicholas R. Howe, Michael E. Leventon, and William T. Freeman. Bayesian reconstruction of 3D human motion from single-camera video. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller, editors, *NIPS*, pages 820–826. The MIT Press, 1999.

[20] Michael Isard and Andrew Blake. CONDENSATION – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, August 1998.

[21] Odest Chadwicke Jenkins and Maja J. Matarić. A spatio-temporal extension to Isomap nonlinear dimension reduction. In Brodley [6].

[22] Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Control, Orlando*, 1997.

[23] W. Kendall, O. Barndorff-Nielsen, and M. N. M. van Lieshout, editors. *Stochastic Geometry: Likelihood and Computation*, chapter 3, pages 141–172. London: Chapman and Hall/CRC, 1999.

[24] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482, July 2002.

[25] Neil D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In Thrun et al. [50].

[26] Neil D. Lawrence. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. Technical Report CS-04-08, Department of Computer Science, University of Sheffield, August 2004.

[27] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics*, 21(3):491–500, July 2002.

[28] Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors. *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*. MIT Press, 2001.

[29] W. E. Leithead and E. Solak andD. J. Leith. Direct identification of nonlinear structure using Gaussian process prior models. In *Proceedings of the European Control Conference*, 2003.

[30] Yan Li, Tianshu Wang, and Heung-Yeung Shum. Motion texture: A two-level statistical model for character motion synthesis. *ACM Transactions on Graphics*, 21(3):465–472, July 2002.

[31] Jun S. Liu and Rong Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998.

[32] D. J. C. MacKay. Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research, Section A*, 354(1):73–80, 1995.

[33] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[34] Roderick Murray-Smith and Barak A. Pearlmutter. Transformations of Gaussian process priors. Technical Report TR-2003-149, Department of Computer Science, Glasgow University, June 2003.

[35] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.

[36] Radford M. Neal and Geoffrey E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in graphical models*, pages 355–368, 1999.

[37] Dirk Ormoneit, Hedvig Sidenbladh, Michael J. Black, and Trevor Hastie. Learning and tracking cyclic human motion. In Leen et al. [28], pages 894–900.

[38] Vladimir Pavlovic, James M. Rehg, and John MacCormick. Learning switching linear models of human motion. In Leen et al. [28], pages 981–987.

[39] Katherine Pullen and Christoph Bregler. Motion capture assisted animation: Texturing and synthesis. *ACM Transactions on Graphics*, 21(3):501–508, July 2002.

[40] Carl Edward Rasmussen and Malte Kuss. Gaussian processes in reinforcement learning. In Thrun et al. [50].

[41] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, December 2000.

[42] Sam T. Roweis. EM algorithms for PCA and SPCA. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *NIPS*. The MIT Press, 1997.

[43] J. Q. Shi, R. Murray-Smith, and D. M. Titterington. Hierarchical Gaussian process mixtures for regression. *Statistics and Computing*, 15:31–41, 2005.

[44] R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*, 3(4):253–264, 1982.

[45] Hedvig Sidenbladh, Michael J. Black, and David J. Fleet. Stochastic tracking of 3D human figures using 2D image motion. In *ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part II*, pages 702–718, London, UK, 2000. Springer-Verlag.

[46] Cristian Sminchisescu and Allan D. Jepson. Generative modeling for continuous non-linearly embedded visual inference. In Brodley [6].

[47] Edward Snelson, Carl Edward Rasmussen, and Zoubin Ghahramani. Warped Gaussian processes. In Thrun et al. [50].

[48] E. Solak, Roderick Murray-Smith, William E. Leithead, Douglas J. Leith, and C. E. Rasmussen. Derivative observations in Gaussian process models of dynamic systems. In Becker et al. [1], pages 1033–1040.

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

[50] Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors. *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*. MIT Press, 2004.

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

[52] Raquel Urtasun, David J. Fleet, and Pascal Fua. Monocular 3-D tracking of the golf swing. In *CVPR (2)*, pages 932–938. IEEE Computer Society, 2005.

[53] Rudolph van der Merwe, Arnaud Doucet, Nando de Freitas, and Eric A. Wan. The unscented particle filter. In Leen et al. [28], pages 584–590.

[54] P. Van Overschee and B. De Moor. N4SID : Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica, Special Issue on Trends in System Identification*, 30(1):75–93, 1994.

[55] Yaser Yacoob and Michael J. Black. Parameterized modeling and recognition of activities. In *ICCV*, pages 120–127, 1998.