Investigating Variational Gaussian Process State-Space Models with Gaussian Likelihood

Christopher J. Cundy
Christ's College

UNIVERSITY OF CAMBRIDGE

A dissertation submitted to the University of Cambridge in partial fulfilment of the requirements for part III of the Computer Science Tripos

University of Cambridge
Computer Laboratory
William Gates Building
15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom

Email: cjc208@cl.cam.ac.uk

June 27, 2017

Declaration

I Christopher J. Cundy of Christ's College, being a candidate for Part III of the Computer Science Tripos, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: -

Signed:

Date:

Abstract

Investigating variational Gaussian process state-space models with Gaussian likelihood

We have demonstrated a principled method to infer the properties of dynamical systems, the doubly variational Gaussian process state-space model (GP-SSM): combining previous work on variational approaches to GP-SSMs with a further variational assumption which allows a fully analytic lower bound on the marginal likelihood.

Previously unpublished work on this topic is put into context, and extended to deal with testing and prediction, which can be achieved fully analytically. We show that the variational GP-SSM is able to learn the dynamics of complicated systems such as the cart-pole, and is also able to infer underlying processes, such as integrating an impulse twice to attain a position, and finding some underlying structure in the Bouc-Wen nonlinear system identification dataset.

In a comparision to an autoregressive (AR) model, the GP-SSM was not much better than the AR in a 5D cartpole problem, whilst being much more costly to train and test. However, the system fared better than the AR model when two of the underlying variables were hidden: it was able to infer their existence and beat the AR model in initial predictions. While both the AR and GP-SSM model performed badly after a dozen or so predictions, the GP-SSM also drastically underestimated its own uncertainty (as is common in variational approaches to Bayesian learning). As in previous work with VFE approximations, we find that progress is stymied by technical issues in performing the optimisation, including the model getting stuck in local minima, and sensitivity to initial conditions. Future work will focus on ensuring robustness of this optimisation, and clarifying the ability of the model to infer entirely unseen latent processes.

Word Count: 9,988.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many interesting problems can be formulated as dynamical systems, where noisy glimpses are observed from some underlying process. Existing approaches to these problems are very data-inefficient, requiring many thousands of samples to obtain any idea of the system's dynamics. This hinders their usefulness in settings where we have restrictions on the amount of data we can observe: such as computationally intensive simulations, or when limited by latency with the outside world. We wish to develop a principled method to infer characteristics of dynamical systems from observations, trying to use all the relevant information optimally. Furthermore, we would prefer if such a method were able to deal with complicated nonlinear systems, such as turbulent aircraft aerodynamics.

In this dissertation, such a method is developed, using a Gaussian process to model the complicated nonlinear functions governing systems. Previous work on variational Gaussian process state space models from R. Frigola is described, and C. Rasmussen's (unpublished) implementation of a doubly variational GP-SSM is described. This implementation is then extended to fully cover the range of training, testing, and prediction; comparison to the state of the art is reported. We describe successes in inferring underlying high-dimensional processes when presented with low-dimensional representations of a dynamical system, and give a principled estimation of the di-

mensionality of the latent states.

The variational GP-SSM serves as a useful tool in the quest to develop general-purpose problem-solving algorithms, especially where we have a large amount of computing power or time and we wish to extract all the information from a small amount of data.

Chapter 2

Background

2.1 Dynamical Systems

A dynamical system is a very general class of mathematical problems in which a system is represented as a vector $\boldsymbol{x}_t$ at time t in an n-dimensional space. The dynamics of the system are encoded in a transition function which maps a state at time t to the state at time $t + 1$.

As a simple example, we can consider the system of an ideal pendulum under gravity, where given the state of the system at time t, we can find the state of the system at a later time t'. We require two independent pieces of information to characterise the state of the system, such as the angular displacement and the angular velocity.

The least number of parameters required to uniquely define a dynamical system's state is known as the dimensionality of the system, and determines many of its properties. For instance, a theorem by Poincare and Bendixson 1901, shows that chaotic behaviour only occurs for continuous dynamical systems of dimension 3 or greater.

Although some systems, such as the pendulum, may be solved analytically, finding exact expressions for the state in the future, more general systems are

analytically impossible to solve and so must be approached in an approximate manner.

Dynamical systems are a very general framework in which to explore the evolution of systems, but they have a definite structure which can be exploited. That is, instead of attempting to directly predict future trajectories of the system from all of the past observations, we can try to learn the underlying transition function from the past observations, predicting future observations by using this inferred function.

However, typically transition functions of dynamical systems are highly nonlinear. For instance, the simple pendulum example described above cannot be solved algebraically when the angles involved become large.

We can deal with such systems in several different ways. One approach is to find a set of nonlinear transformations which simplify the function, i.e. learn a convenient representation. This is the approach taken by deep learning systems. In the pendulum example above, we can represent the state transformation more conveniently in momentum-energy phase space.

We can also use an approach that can explicitly model complicated nonlinear functions. In the following work we use Gaussian Processes as a powerful tool to directly handle these functions.

2.2 Multivariate Gaussian Distributions

Since we will make heavy use of Gaussian distributions, it is worth reiterating several key results.

We will use the result of conditional dependence of a Gaussian, i.e. given that a vector $\boldsymbol{x} \sim \mathcal{N}_{\boldsymbol{x}}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix}, \qquad \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \qquad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{12}^\top & \boldsymbol{\Sigma}_{22} \end{pmatrix}, \tag{2.1}$$

the conditional probability $p(\boldsymbol{x}_1|\boldsymbol{x}_2)$ also follows a normal distribution parameterised with $\boldsymbol{\mu}', \boldsymbol{\Sigma}'$, where

$$\boldsymbol{\mu}' = \boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{12}^\top \boldsymbol{\Sigma}_{11}^{-1}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1) \qquad \text{and} \qquad \boldsymbol{\Sigma}' = \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{12}^\top \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\Sigma}_{12}. \quad (2.2)$$

We will come across a few complicated expressions involving kernel matrices that are actually denoting a very simple conditional dependence when refactored into this form, so this equation is worth keeping in mind.

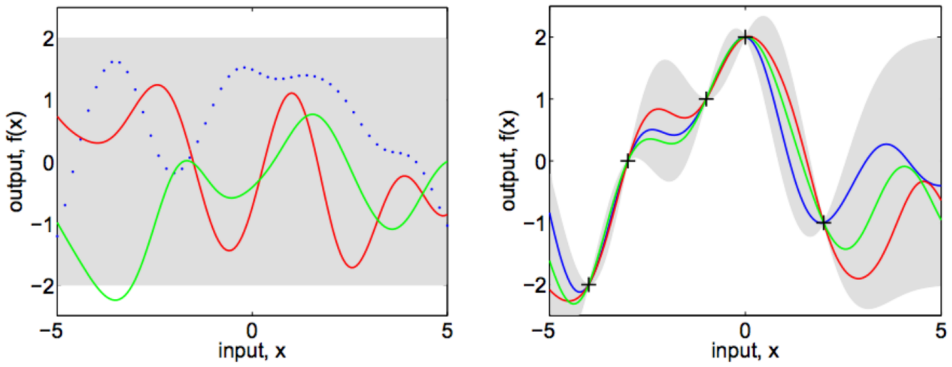
2.3 Gaussian Processes

A Gaussian Process is a collection of variables, of which any finite linear combination has a joint Gaussian distribution: see C. E. Rasmussen and Williams 2005 for a comprehensive description. The most natural motivation for their use in machine learning may be to view them as a generalisation of a radial basis function (RBF) model. In an RBF model, the predicted value of a function at a point is computed from a linear combination of a finite number of Gaussians, centred at points in the prediction space. In a Gaussian process, we have such a contribution from every point in the space, and our vector of weights in the RBF case is replaced with its infinite-dimensional analogue - the covariance function $K(\boldsymbol{x}_1, \boldsymbol{x}_2)$ giving the contribution to the prediction of $\boldsymbol{x}_1$ from $\boldsymbol{x}_2$. Excitingly, after marginalising out the predictions from all points in space, we are left with a Gaussian describing the distribution of the predicted value at the point we are interested in. This allows us to naturally capture our uncertainty in point estimates. Furthermore, if we try to learn a function producing data with a GP, as a nonparametric method, they do not make a maximum-likelihood estimation of any parameters of the function. Instead we get a distribution over all possible functions, conditioned on the points that we have observed. The choice of covariance function K allows us to encode very general assumptions about the properties of the function we are trying to predict, such as differentiability, periodicity, etc. GPs are a very good choice for our representation of the state space model

transition function, as they provide a full description of the uncertainty in the estimation of the transition function, instead of a single maximum likelihood estimation.

See figure 2.1 for an illustration of Gaussian process regression and the intuitive way which a GP naturally captures the increasing uncertainty of the prediction as we head further from the observations.

Figure 2.1: A plot from C. E. Rasmussen and Williams 2005 showing Gaussian Process regression. On the left, the prior distribution over functions is shown, along with several examples of functions drawn from that distribution. On the right, the posterior over functions after several data points have been observed. Intuitively, the posterior is tighter near the observations (as we know the function must have a value near the observed value), and looser further away from the observations.



(a), prior (b), posterior

However, GPs do have considerable disadvantages. The main obstacle to their application is the computational complexity of prediction, which involves inversion of a matrix and goes as $\mathcal{O}(n^3)$ for n observed data points. Recent work has provided many different sparse approximations to a full GP, which introduce M inducing inputs. The output of the GP is conditioned on these inputs instead of the full number of observations, reducing the complexity of prediction to $\mathcal{O}(nM^2)$.

Recent work such as Titsias 2009 has provided a variety of techniques to choose these inducing inputs while preserving as much information as possible.

2.4 Sparse GPs

The main drawback of Gaussian Process methods is their computational complexity, which can be reduced with sparse methods. Central to these approaches is the question of how the set of inducing inputs $\boldsymbol{z}$ is chosen. Although early approaches to sparse methods included obvious choices such as choosing a simple subset of the data points, more sophisticated methods allowed the inducing inputs themselves to be treated as parameters for the GP, and jointly optimised along with the hyperparameters of the GPs. The two predominant sparse methods are the Fully Independent Training Conditional (FITC), introduced by Snelson and Ghahramani 2006, and the Variational Free Energy (VFE) approach, introduced by Titsias 2009. We can summarise the key properties of the two approximations in table 2.1.

Table 2.1: Comparison of the two leading sparse approximations to Gaussian Processes, Variational Free Energy (VFE), and Fully Independent Training Conditional (FITC), with comments sourced from Bauer, Wilk, and C. E. Rasmussen 2016.

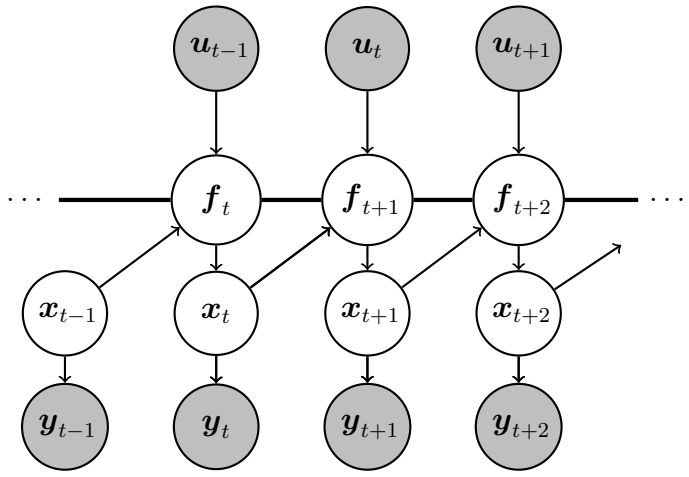
	VFE	**FITC**
Noise Variance	Can Severely Underestimate	Generally Overestimates
Additional Inducing Inputs	Can Ignore	Always Improves Performance
Many Inducing Inputs	True GP is Global Minimum	True GP is not Global Minimum
Optimisation	Can be Difficult to Optimise	Easy to Optimise

While the FITC method often performs well in practice, if aggressively optimised it can give somewhat pathological results with wildly oscillating amounts of noise. In this work, we follow the approach of Frigola in using a VFE approximation, as the guarantee that the approximation to the true GP always gets better if we add more inducing inputs provides a straightforward method to trade off increased computation for increased accuracy. Of course, knowing that the sparse approximation is close to the exact answer if we use enough inducing inputs doesn't let us know if we have used enough inputs or not. Along with the technical difficulties that can occur in optimising the VFE bound, we should keep in mind the danger of using too few inducing inputs.

2.5 Gaussian Process State-Space Models

Knowing which variables depend on each other in a model is crucially important. This can be summarised concisely by a graphical model (see Bishop 2006 for more details). A Gaussian process state-space model is described by the model in figure 2.8. We follow the convention in Dietz 2010, with shaded circles representing observed variables, unshaded circles representing latent variables, and a thick black line representing variables drawn from the same Gaussian process. Note that the thick line implies that functions drawn from the GP are conditioned on all the latent $\boldsymbol{x}$ states, not just the immediately preceding one. The diagram shows that the central object in our model is the

Figure 2.2: A graphical model for the conditional dependencies in the Gaussian Process State Space Model.



inferred transition function $\boldsymbol{f}$, which is described by a distribution over functions according to a Gaussian process. The latent state at a time t is given by evaluating the transition function on the state at time $t-1$, (along with some deterministic control inputs $\boldsymbol{u}$ and adding some noise (process noise)). The observations are then obtained from the likelihood function (generally expressed as a parametric function $p(\boldsymbol{y}_t|\boldsymbol{x}_t, \theta_y)$.)

Crucially, the latent process and state is entirely unobserved, and inferred from the observations.

The model is then described with the following conditional dependencies, as described by Frigola 2015

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m_f(\boldsymbol{x}), k_f(\boldsymbol{x}, \boldsymbol{x}')), \tag{2.3}$$

$$\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0), \tag{2.4}$$

$$\boldsymbol{f}_t = f(\boldsymbol{x}_{t-1}), \tag{2.5}$$

$$\boldsymbol{x}_t | f_t \sim \mathcal{N}(\boldsymbol{f}_t, Q), \tag{2.6}$$

$$\boldsymbol{y}_t | x_t \sim p(\boldsymbol{y}_t | x_t, \theta_y). \tag{2.7}$$

In this formulation, we clearly separate out the uncertainty due to the lack of knowledge of the dynamics of the system, which is given by the uncertainty in the predictions of f from the GP in equation 2.3; the uncertainty due to measurement error, which is given by the parameters θ of the likelihood function in equation 2.7; and the uncertainty due to stochasticity in the process itself, which is described by the covariance matrix Q in equation 2.6. Specifying the conditional dependencies of the variables in the model tells us exactly how we should invert the generative process of the model to infer the underlying states using Bayes' theorem.

2.6 The Variational GP-SSM

We can read off diagram 2.8 to see that the joint probability of the model is

$$p(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{f}) = \prod_{t=1}^{T} p(\boldsymbol{y}_t, |\boldsymbol{x}_t) \prod_{t=2}^{T} p(\boldsymbol{x}_t, |\boldsymbol{f}_t) p(\boldsymbol{f}_t | \boldsymbol{f}_{1:t-1}, \boldsymbol{x}_{1:t-1}). \tag{2.8}$$

In order to speed up inference when there are many data points, augmented inputs $\boldsymbol{z}$ and targets $\boldsymbol{v} = f(\boldsymbol{z})$ are chosen so that the augmented joint probability is

$$p(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{f}, \boldsymbol{v}) = p(\boldsymbol{y}|\boldsymbol{x}) p(\boldsymbol{x}, \boldsymbol{f}|\boldsymbol{v}) p(\boldsymbol{v}). \tag{2.9}$$

Unfortunately exact inference in this model is completely intractable. A variety of methods for overcoming this have been proposed, including using sampling methods, or variational approaches.

2.7 Variational Optimisation of a GP-SSM

Methods to perform approximate Bayesian inference generally fall into two camps: numerical sampling techniques, and algebraic variational techniques. Whilst sampling techniques are typically guaranteed to result in the exact posterior, given enough time to converge, variational approaches rely on a specific assumption about an approximate distribution (such as factorisation properties or a chosen functional form), which limit the accuracy of the approximation: if the variational assumption is not good, then the variational approach will not do very well. However, variational approaches result in an analytic solution which often gives more insight than a numerical solution, even if the numerical solution is more accurate. In Frigola 2015, the derivation of the variational GP-SSM is described, which we reproduce in this section.

In the variational approach, we wish to find the most probable model (i.e. the values of the inducing inputs and latent parameters) given the observations we have of the system. Although we are not able to compute the probability of the model directly, the standard variational approach (see Bishop 2006) allows us to derive a lower bound for its value, namely

$$\log p(\boldsymbol{y}|\boldsymbol{\theta}) \geq \left\langle \log \frac{p(\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{v}, \boldsymbol{y})}{q(\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{v})} \right\rangle_{q(\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{v})}, \tag{2.10}$$

where $\boldsymbol{v}$ are our chosen inducing targets. Note that we have chosen q as an approximating distribution to the real probability $p(\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{v})$.

Substituting in the expression for the joint probability of the model, this is

$$\left\langle \log \frac{p(\boldsymbol{v})p(\boldsymbol{x}_0)\prod_{t=1}^{T} p(\boldsymbol{y}_t|\boldsymbol{f}_{1:t-1}, \boldsymbol{x}_{0:t-1}, \boldsymbol{u})p(\boldsymbol{y}_t|x_t)p(\boldsymbol{x}_t|\boldsymbol{f}_t)}{q(\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{v})} \right\rangle_{q(\boldsymbol{x},\boldsymbol{f},\boldsymbol{v}).} \tag{2.11}$$

In order to make the following equations tractable, a suitable choice of $q(\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{v})$ is needed. We choose

$$q(\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{v}) = q(\boldsymbol{v})q(\boldsymbol{x}) \prod_{t=1}^{T} p(\boldsymbol{f}_t|\boldsymbol{f}_{1:t-1}, \boldsymbol{x}_{0:t-1}, \boldsymbol{v}). \tag{2.12}$$

The terms which are dependant only on $\boldsymbol{f}$ cancel out leaving us with a much simpler lower bound,

$$\left\langle \log \frac{p(\boldsymbol{v})p(\boldsymbol{x}_0)\prod_{t-1}^{T} p(\boldsymbol{y}_t\boldsymbol{x}_t)p(\boldsymbol{x}_t|\boldsymbol{f}_t)}{q(\boldsymbol{v})q(\boldsymbol{x})} \right\rangle_{q(\boldsymbol{x},\boldsymbol{f},\boldsymbol{v}).} \tag{2.13}$$

Our assumption of the factorisation of $p(\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{v})$ is the variational assumption, and is the key assumption made in the whole derivation. What is the content of the assumption? We are assuming that the probability of a function $\boldsymbol{f}$ is conditional only on previous function draws, the inducing inputs, and previous latent states. We are also assuming that the probability of $\boldsymbol{v}$ doesn't depend on the latent state $\boldsymbol{x}$. This seems to be a very reasonable assumption, especially when there are sufficent inducing inputs, given that the $\boldsymbol{z}$s are simply there to support the GP.

Substituting in our choice, we find that the evidence lower bound (sometimes known as the ELBO) is given by a sum of different terms,

$$\mathcal{L}(q(\boldsymbol{v}), q(\boldsymbol{x}), \boldsymbol{\theta}) = -\mathrm{K}L(q(\boldsymbol{v})||p(\boldsymbol{v})) + \mathcal{H}(q(\boldsymbol{x})) + \langle \log p(\boldsymbol{x}_0) \rangle_{q(\boldsymbol{x}_0)} \tag{2.14}$$

$$+ \sum_{t=1}^{T} \left\langle \langle \log p(\boldsymbol{x}_t|\boldsymbol{f}_t) \rangle_{p(\boldsymbol{f}_t|\boldsymbol{x}_{t-1},\boldsymbol{v})} \right\rangle_{q(\boldsymbol{x})q(\boldsymbol{v})} + \langle \log p(\boldsymbol{y}_t|\boldsymbol{x}_t) \rangle_{q(\boldsymbol{x})} \,, \tag{2.15}$$

where $\mathrm{K}L(p,q)$ is the Killback-Leibler divergence between two probability distributions p, q, and $\mathcal{H}$ is the entropy.

It is possible to prove that the functional form of the distribution for $\boldsymbol{v}$ which maximises the lower bound is a multivariate Gaussian. This extremely fortunate result doesn't require any asumptions further to the factorisation of $q(\boldsymbol{v})$ made above. Furthermore, the mean and covariance of $q(\boldsymbol{v})$, $\boldsymbol{\mu}, \Sigma$, are both functions of the two sufficient statistics

$$\begin{aligned} \boldsymbol{\Psi}_1 &= \sum_{t=1}^{T} \left\langle \boldsymbol{K}_{\boldsymbol{x}_{t-1},\boldsymbol{z}}^{\top} \boldsymbol{Q}^{-1} \boldsymbol{x}_t \right\rangle_{q(\boldsymbol{x}_{t-1:t}),} \\ \boldsymbol{\Psi}_2 &= \sum_{t=1}^{T} \left\langle \boldsymbol{K}_{\boldsymbol{x}_{t-1},\boldsymbol{z}}^{\top} \boldsymbol{Q}^{-1} \boldsymbol{K}_{x_{t-1},\boldsymbol{z}} \right\rangle_{q(\boldsymbol{x}_{t-1}),} \end{aligned} \tag{2.16}$$

If we compute the expectations over $q(\boldsymbol{v})$, we find that the lower bound on the evidence is

$$\mathcal{L}(q(\boldsymbol{v}), q(\boldsymbol{x}), \boldsymbol{\theta}) = -\,\mathrm{K}L(q(\boldsymbol{v})||q(\boldsymbol{v})) + \mathcal{H}(q(\boldsymbol{x})) + \left\langle \log p(\boldsymbol{x}_0) \right\rangle_{q(\boldsymbol{x}_0)} \tag{2.17}$$

$$+ \sum_{t=1}^{T} \left(-\frac{1}{2} \left\langle (\boldsymbol{Q}^{-1}(\boldsymbol{B}_{t-1} + \boldsymbol{A}_{t-1} \boldsymbol{\Sigma} \boldsymbol{A}_{t-1}^{\top})) \right\rangle_{q(\boldsymbol{x}_{t-1})} \right) \tag{2.18}$$

$$+ \left\langle \log \mathcal{N}(\boldsymbol{x}_t | \boldsymbol{A}_{t-1} \boldsymbol{\mu}, \boldsymbol{Q}) \right\rangle_{q(\boldsymbol{x}_{t-1:t})} \tag{2.19}$$

$$+ \left\langle \log \left(p(\boldsymbol{y} | \boldsymbol{x}_t) \right) \right\rangle_{q(\boldsymbol{x}_t)}, \tag{2.20}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are given by

$$\boldsymbol{A}_{t-1} = \boldsymbol{K}_{\boldsymbol{x}_{t-1},\boldsymbol{z}} \boldsymbol{K}_{\boldsymbol{z}}^{-1} \tag{2.21}$$

$$\boldsymbol{B}_{t-1} = \boldsymbol{K}_{\boldsymbol{x}_{t-1},\boldsymbol{x}_{t-1}} - \boldsymbol{K}_{\boldsymbol{x}_{t-1},\boldsymbol{z}} \boldsymbol{K}_{\boldsymbol{z}}^{-1} \boldsymbol{K}_{\boldsymbol{z},\boldsymbol{x}_{t-1}}, \tag{2.22}$$

and $\boldsymbol{K}_{\boldsymbol{x}_t,\boldsymbol{z}}$ is the Gram matrix $(\boldsymbol{K}_{\boldsymbol{x},\boldsymbol{z}})_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{z}_j)$. Referring to equation 2.2, we can interpret the term $\boldsymbol{A}_{t-1}\boldsymbol{\mu}$ as the mean prediction of the state at time t, extrapolating from the state at time $t-1$, conditioning on the inducing inputs $\boldsymbol{z}$. We can also see that the predictive covariance at time t is given by the expression $\boldsymbol{B}_{t-1}$.

So if we wish to carry out a variational approximation to a GP-SSM model such as the one described in this chapter, we know the optimal distribution for $q(\boldsymbol{v})$, but we are left with an open question as to what distribution to use for $q(\boldsymbol{x})$. Luckily, we are able to find an explicit optimal form of this distribution. We find that the optimal distribution is

$$q^*(\boldsymbol{x}) \propto p(\boldsymbol{x}_0) \prod_{t=1}^{T} p(\boldsymbol{y}_t|\boldsymbol{x}_t) \exp\left(-\frac{1}{2}\mathrm{tr}(\boldsymbol{Q}^{-1}(\boldsymbol{B}_{t-1} + \boldsymbol{A}_{t-1}\boldsymbol{\Sigma}\boldsymbol{A}_{t-1}^T))\right) \mathcal{N}(\boldsymbol{x}_T|\boldsymbol{A}_{t-1}\boldsymbol{\mu}, \boldsymbol{Q}). \tag{2.23}$$

This is a somewhat simpler functional dependence, which can actually be interpreted as a state space model itself. However, in this case the model factorises in a Markovian fashion: the probability of a state $\boldsymbol{x}_t$ depends only on the previous state $\boldsymbol{x}_{t-1}$. The state transition function is clearly not Gaussian, as it has a complicated dependence on the $\boldsymbol{B}_{t-1}$ and $\boldsymbol{A}_{t-1}$ terms (which are nonlinear, but Markovian as the only latent state they depend on is $\boldsymbol{x}_{t-1}$). In Frigola 2015, evaluating the posterior through this nonlinear transformation was achieved with Monte-Carlo Markov chain methods. In the present work, we deliberately choose to use a non-optimal functional form for $q(\boldsymbol{x})$. However, our choice of $q(\boldsymbol{x})$ to be Gaussian does allow us to sidestep any use of approximate sampling methods and let us study the maximum likelihood model analytically.

2.8 Autoregressive Models

Predictions on dynamical systems can be made using autogressive (AR) models. An nth order linear AR model, denoted $AR(n)$, is given by the equation predicting an observation at time t,

$$X_t = \sum_{i=1}^{N} a_i X_{t-i} + Z, \tag{2.24}$$

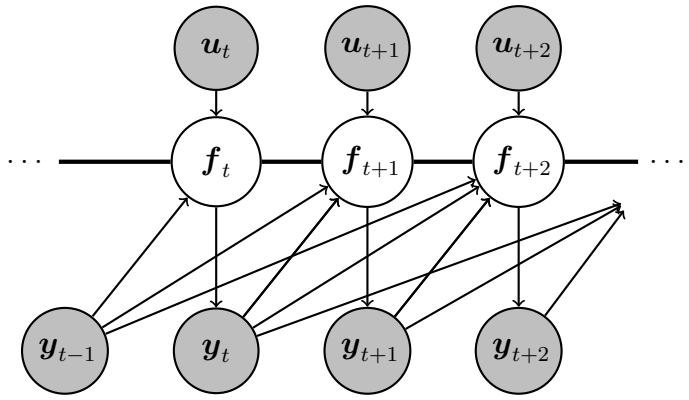
Where Z is a random variable, $Z \sim \mathcal{N}(0, 1)$. We might expect such a model to perform very badly, as it doesn't take into account a latent process at all, and there isn't any distinction between observation and process noise. This latter point is particularly troubling, as a small amount of noise into the inputs of a function can result in very wrong outputs, as extensively studied in McHutchon 2014. Despite these downsides, AR models are widely used, and are reasonably successful.

More recent work has introduced nonlinear AR models, such as the GP-FNARX setup. In this framework, we drop the linearity condition, and simply predict the nth observation as a function of the preceding observations, using a Gaussian process to model our function.

Using a Gaussian process to represent our function also allows us to propagate uncertainty through our predictions analytically. We would certainly expect that our predictions will become less accurate the further forward in time we look, so having a directly estimation of the uncertainty associated with each prediction is very useful. A Gaussian process over the autoregressive model gives us a very natural approach to this, with the prediction given by

$$\boldsymbol{y}_t = f(\boldsymbol{y}_{t-1}, \ldots, \boldsymbol{y}_{t-N}, \boldsymbol{u}_{t-1}, \ldots, \boldsymbol{u}_{t-N}). \tag{2.25}$$

Figure 2.3: A graphical model for the conditional dependencies in the 3rd order Gaussian Process Autoregressive Model, in training.



Chapter 3

Related Work

As described by Roweis and Ghahramani 2000, the idea of using the underlying probabilistic structure of a dynamical system to interpret observations has a long history. The first widely-used application of the dynamical system framework was the Kalman filter, introduced in Kalman 1960, which corresponds to a state-space model with a linear transition function and Gaussian likelihood. These restrictions on the form of the dynamics allow the Kalman filter's equations to be solved cheaply and exactly: key reasons why it is used widely to aggregate sensor readings in fields as disparate as aircraft autopilots, econometrics, and signal processing.

Dropping the linear assumption increases the complexity of the problem to a huge degree. The extended Kalman filter tries to handle nonlinear systems by linearising the system around the current point in state space. Although this ad-hoc extension of the Kalman filter does tend to work well with certain nonlinearities, this depends to a large extent on knowing the accurate dynamics of the system, as Jacobians are required to calculate the required linearisation. See Ljung 1979 for more details. More recent work on the unscented Kalman filter, introduced in Wan and Van Der Merwe 2000 have achieved remarkably large gains in accuracy for the same computational complexity.

In the related field of system identification, the early work relied on assumptions of a certain parametric models, the parameters of which were then inferred from recorded data. More recent work has attempted to relax the restrictions on specific parametric forms by introducing nonparametric methods such as neural networks or Gaussian processes. The recent focus on deep learning arises from its remarkable ability to learn some underlying lower-dimensional set of latent variables describing e.g. image classes in a high-dimensional space of pixel intensities (see Goodfellow, Bengio, and Courville 2016) for more detail. Similarly, by not relying on a parametric form for the transition function in a dynamical system and fitting a Gaussian Process for the proposed transition function, the GP autoregressive model described in section 2.8 can give very good results. Since we assume our inputs will be corrupted by some observation noise, we can do even better by smoothing the data before we feed it into our regression. Although this approach does not attempt to understand the structure of the dynamical system, in practice it serves very well.

Another approach to controlling systems, Reinforcement Learning, again sidesteps the problem of learning an accurate model of the dynamics of a complex system and instead attempts to learn an accurate approximation to the value function, which maps actions onto the expected rewards obtained by taking that action. Extremely impressive results have been obtained by deep reinforcement learning algorithms on diverse problems such as Go or Atari games (see Silver et al. 2016). However the representation of the environment that deep learning systems create is very opaque to interpret, being nonparametrically described by thousands or millions of neuron weights. Furthermore, RL systems are data-inefficient, requiring very many training periods to perform well.

In recent years Gaussian process latent variable models have started to be widely used. The variational GP-SSM approach described in the previous section describes the reasoning behind their use. Along with the variational approach, many attempts have been made to simplify the GP-SSM model to make a solution tractable. These include assuming a parametric form for the

GP based on R. Turner, Deisenroth, and C. E. Rasmussen 2010, and much work involving sampling methods in Wang, Fleet, and Hertzmann 2008.

There are a wide variety of approaches to solving dynamical systems. The approach outlined in this thesis has the advantage of being fully analytic and not relying on sampling methods, and extends a long history of Gaussian process methods used to comprehend dynamical systems.

Chapter 4

Design and Implementation

Building on the previous work in the area, we discuss the (previously unpublished) implementation of a doubly-variational GP-SSM by C. Rasmussen and describe how we can extend this framework to derive maximum-likelihood predictions.

In order to allow tractability, and to remove any need for sampling methods, several further assumptions are introduced on top of those discussed in the previous chapter.

Squared Exponential Covariance Function We choose the covariance function in the Gaussian Process over the transition function to be a squared exponential kernel as described in C. E. Rasmussen and Williams 2005. In order to eliminate any sampling, we are forced to choose a kernel which can be analytically integrated against a Gaussian density. This introduces one hyperparameter per latent dimension per GP, the characteristic length scale ℓ. As discussed in C. E. Rasmussen and Williams 2005, the SE covariance function has been the standard choice of covariance function for Gaussian processes for decades.

Gaussian Likelihood We assume that $p(\boldsymbol{y}_t|\boldsymbol{x}_t)$ is Gaussian. This allows us to solve for an optimal mapping from the latent space to the observation space. Assuming that errors have a Gaussian form is a very common

assumption in physical science, and is often justified by appeal to the central limit theorem (if the errors are generated by many interactions of independent errors, the total observed error will be approximately Gaussian). If the likelihood was in actual fact not Gaussian, we would expect that our method would still be able to form a good model if we allowed it more latent states, as it could 'absorb' the likelihood into the state-space transition function.

Gaussian Latent Variable Distribution The crucial assumption made in this implementation is that the distribution $q(\boldsymbol{x})$ is Gaussian. Equation 2.23 shows that the optimal $q(\boldsymbol{x})$ has a Markovian structure, but is complicated by non-Gaussian terms involving $\boldsymbol{A}$ and $\boldsymbol{B}$, which are ignored. Although we can prove that our choice of $q(\boldsymbol{x})$ is non-optimal, we hope that the increase in speed from the fully analytic treatment of the optimisation allows for better overall performance than a more optimal choice for $q(\boldsymbol{x})$.

While the first two assumptions may be the best assumptions in certain cases, such as if we know our transition function is infinitely differentiable with Gaussian noise, the final assumption is a deliberate choice for a suboptimal distribution, in a trade-off in order to achieve an analytic solution.

When evaluating a model that tries to learn the dynamics of a system, the most obvious approach is to ask the model to predict a new sequence and measure the accuracy of this generated sequence. However, we need to specify some sort of initial conditions for the generated sequence so that we can compare to a specific real sequence. The natural way to provide these initial conditions is to truncate a generated sequence, showing the first few observations to the model, then evaluating its predictions against the remaining observations. This means we have to develop the algorithm for three cases:

Training Given some valid training observations and control inputs, jointly find the nonlinear transition functions and underlying latent states corresponding to the maximum likelihood system.

Inference on a test seed set Given some observations, control inputs, and

a trained model, find the latent states corresponding to the observations, while fixing the transition function and underlying mapping from latent states to observations.

Prediction Given a set of latent states corresponding to the output of the previous case, and control inputs, output future predictions.

The first case is the focus of an unpublished note (C. Rasmussen 2016). In this project we developed the second and third cases. In practice, the output in the first two cases was the negative log marginal likelihood and derivatives, allowing a range of different algorithms to be used to solve the gradient descent problem to find the optimal set of parameters.

4.1 Implementation

Table 4.1: Showing the key variables in the implementation.

Variable	Meaning
E	Dimensionality of the latent space
U	Dimensionality of the control inputs
M	Number of inducing inputs
N	Number of timeseries
T(n)	Length of the timeseries

Since we wish to train our model by gradient descent, we must calculate the lower bound on the log likelihood, and provide derivatives. The lower bound follows from equation 2.17, where we additionally explicitly compute the expectations over $q(\boldsymbol{x})$. We introduce the following notation for the distributions $q(\boldsymbol{x})$,

$$p\begin{pmatrix}\boldsymbol{x}_t\\ \boldsymbol{x}_{t+1}\end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix}\boldsymbol{\mu}_t\\ \boldsymbol{\mu}_{t+1}\end{pmatrix}, \begin{pmatrix}\boldsymbol{\Sigma}_{t,t} & \boldsymbol{\Sigma}_{t,t+1}\\ \boldsymbol{\Sigma}_{t,t+1}^\top & \boldsymbol{\Sigma}_{t+1,t+1}\end{pmatrix}\right). \tag{4.1}$$

Which identifies the joint probability of a particular value of the underlying state at t and $t+1$ as a Gaussian with a block-matrix form for the covari-

ance matrix. Due to the marginalisation property of a Gaussian, we can quickly identify the marginal distribution $p(\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{\mu}_t, \Sigma_{t,t})$, and similarly for $p(\boldsymbol{x}_{t+1})$. For an illustration of how these distributions look in the simpler case when the Σ matrices are scalars, see figure 5.1.1.

The ELBO arises from several sets of terms, which can be classified into transition function terms, which govern the interaction of the latent space points with the underlying Gaussian process transition function; entropy terms which penalise overly uncertain $q(\boldsymbol{x})$ distributions; and likelihood terms which penalise latent space points which correspond to observations differing from those actually observed.

The transition function and likelihood terms are different in the testing case, as we are using a fixed transition function and latent space mapping determined in prior training, instead of calculating this based on the latent points. This means that the expression obtained for the ELBO is quite different in the training and testing case.

4.1.1 Transition Function Terms

The functions $\Psi_{1,2}$ completely describe the interaction of the latent space points with the inferred transition function f. Ψ is defined in equation 2.16. We run into Ψ in two ways, as it is used in parameterising the optimal $q(\boldsymbol{v})$ Gaussian distribution, but it also appears when computing the expectations in equation 2.17. In the training case this leads to considerable simplification, but in the testing (latent state inference) case we define Ψ^* as the expectation over test data, leaving Ψ to stand for the expectations over (fixed) training data.

In the training case, we find that the expectations over $q(\boldsymbol{x})$ can be expressed

through the terms $\boldsymbol{\Psi}_{1e}, \boldsymbol{\Psi}_{2e}, \boldsymbol{K}_e$ as

$$\begin{aligned} &\tfrac{1}{2}\sum_{e=1}^{E}\log|(\boldsymbol{K}_e+\boldsymbol{\Psi}_{2e})^{-1}\boldsymbol{K}_e| + \mathrm{tr}\boldsymbol{K}_e^{-1}\boldsymbol{\Psi}_{2e} + +\boldsymbol{\Psi}_{1e}^{\top}(\boldsymbol{K}_e+\boldsymbol{\Psi}_{2e})^{-1}\boldsymbol{\Psi}_{1e} \\ &-\tfrac{1}{2}\mathrm{tr}Q^{-1}\sum_{t=2}^{T}(I+\boldsymbol{\mu}_t^{\top}\boldsymbol{\mu}_t+\Sigma_{t,t}) - \tfrac{T-1}{2}\log|Q| - \tfrac{(T-1)E}{2}\log(2\pi). \end{aligned} \tag{4.2}$$

In the testing case, the relevant terms are

$$\begin{aligned} &\tfrac{1}{2}\sum_{e=1}^{E}\mathrm{tr}\boldsymbol{K}_e^{-1}\boldsymbol{\Psi}_{2e}(K_e+\Psi_{2e})^{-1}\Psi_{2e}^{*} - \mathrm{tr}(K_e+\Psi_{2e})^{-1}\Psi_{1e}\Psi_{1e}^{\top}(K_e+\Psi_{2e})^{-1}\Psi_{2e}^{*} \\ &+\boldsymbol{\Psi}_{1e}^{\top}(\boldsymbol{K}_e+\boldsymbol{\Psi}_{2e})^{-1}\boldsymbol{\Psi}_{1e}^{*} - \tfrac{1}{2}\mathrm{tr}Q^{-1}\sum_{t=2}^{T}(I+\boldsymbol{\mu}_t^{\top}\boldsymbol{\mu}_t+\Sigma_{t,t}) \\ &-\tfrac{T-1}{2}\log|Q| - \tfrac{(T-1)E}{2}\log(2\pi). \end{aligned} \tag{4.3}$$

We also need to provide the relevant derivatives - with respect to Ψ in the training case, and with respect to Ψ^* while Ψ is held constant in the testing case. These are found by repeated application of the chain rule.

Explicit Calculation

Explicitly calculating Ψ involves computing the expectation of $\boldsymbol{K}$ over $q(\boldsymbol{x})$ described in eq. 2.16. We need to fully detail this computation, as it will be used in finding the optimal forward prediction in the test case.

To compute these expectations, we multiply the joint $q(\boldsymbol{x}_t, \boldsymbol{x}_{t-1})$ by the covariance function, which is written as a joint Gaussian itself,

$$k_e(\mathbf{x}_{t-1}, \mathbf{z}_{ie}) \;=\; \exp\Big(-\tfrac{1}{2}\begin{bmatrix}\mathbf{x}_{t-1}-\mathbf{z}_{ie}\\ \mathbf{x}_t\end{bmatrix}^{\top}\begin{bmatrix}\Lambda_e^{-1} & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\mathbf{x}_{t-1}-\mathbf{z}_{ie}\\ \mathbf{x}_t\end{bmatrix}\Big),$$

so that

$$Q\Psi_{1,i,e} = \int \mathbf{x}_t k_e(\mathbf{x}_{t-1}, \mathbf{z}_{ie}) q(\mathbf{x}_{t-1:t}) d\mathbf{x}_{t-1} d\mathbf{x}_t \;=\; \big(\boldsymbol{\mu}_t + \Sigma_{t,t-1}[\Lambda_e + \Sigma_{t-1,t-1}]^{-1}(\mathbf{z}_{ie} - \boldsymbol{\mu}_{t-1})\big)$$
$$\times |I + \Lambda_e^{-1}\Sigma_{t-1,t-1}|^{-1/2} \exp\big(-\tfrac{1}{2}(\boldsymbol{\mu}_{t-1} - \mathbf{z}_{ie})[\Lambda_e + \Sigma_{t-1,t-1}]^{-1}(\boldsymbol{\mu}_{t-1} - \mathbf{z}_{ie})\big). \tag{4.4}$$

$$Q\Psi_{2,i,j} = \int k_e(\mathbf{z}_{ie}, \mathbf{x}_{t-1}) k_e(\mathbf{x}_{t-1}, \mathbf{z}_{je}) q(\mathbf{x}_{t-1}) d\mathbf{x}_{t-1} \;=\; \exp(-(\mathbf{z}_{ie} - \mathbf{z}_{je})\Lambda_e^{-1}(\mathbf{z}_{ie} - \mathbf{z}_{je})/4)$$
$$\times |I + 2\Lambda_e^{-1}\Sigma_{t-1,t-1}|^{-1/2} \exp(-(\tfrac{\mathbf{z}_{ie}+\mathbf{z}_{je}}{2} - \boldsymbol{\mu}_{t-1})[\Lambda_e/2 + \Sigma_{t-1,t-1}]^{-1}(\tfrac{\mathbf{z}_{ie}+\mathbf{z}_{je}}{2} - \boldsymbol{\mu}_{t-1})/2) \tag{4.5}$$

4.1.2 Entropy

The entropy of the Gaussian $q(\boldsymbol{x})$ distributions is simply given by

$$\mathcal{H}(q(\boldsymbol{x})) = \tfrac{TE}{2}(1 + \log(2\pi)) + \tfrac{1}{2}\sum_{t=2}^{T} \log \left| \begin{pmatrix} \boldsymbol{\Sigma}_{t,t} & \boldsymbol{\Sigma}_{t,t+1} \\ \boldsymbol{\Sigma}_{t,t+1}^\top & \boldsymbol{\Sigma}_{t+1,t+1} \end{pmatrix} \right| - \tfrac{1}{2}\sum_{t=2}^{T-1} \log |\Sigma_t|, \tag{4.6}$$

and is the same in the training and testing case.

4.1.3 Likelihood

The terms relating to the likelihood are slightly more complex, as they tie the model to observations.

The latent states map to the observed values through a linear mapping, so that

$$p(\boldsymbol{y}_t|\boldsymbol{x}_t) = \mathcal{N}(C\boldsymbol{x}_t + C', R). \tag{4.7}$$

When computing the expectation over $q(\boldsymbol{x}_t)$, which itself has a Gaussian distribution, the $\boldsymbol{x}_t$ marginal covariance Σ_t is transformed through the mapping

to give

$$\sum_{t=1}^{T}\langle \log p(\boldsymbol{y}_t|\boldsymbol{x}_t)\rangle_{q(\boldsymbol{x}_t)} \;=\; -\tfrac{DT}{2}\log(2\pi) - \tfrac{T}{2}\log|R| \\ -\tfrac{1}{2}\mathrm{tr}R^{-1}\sum_{t=1}^{T}\big((\boldsymbol{y}_t - C\boldsymbol{\mu}_t)(\boldsymbol{y}_t - C\boldsymbol{\mu}_t)^\top + C\Sigma_t C^\top\big). \tag{4.8}$$

It is possible to find the mapping R^* and C^* that maximises the likelihood:

$$R^* \;=\; \tfrac{1}{T}\Big[\sum_{t=1}^{T}\boldsymbol{y}_t\boldsymbol{y}_t^\top - C^*\sum_{t=1}^{T}\boldsymbol{\mu}_t\boldsymbol{y}^\top\Big], \quad \text{and} \quad C^* \;=\; \sum_{t=1}^{T}\boldsymbol{y}_t\boldsymbol{\mu}_t^\top\Big[\sum_{t=1}^{T}\boldsymbol{\mu}_t\boldsymbol{\mu}_t^\top + \Sigma_{t,t}\Big]^{-1}. \tag{4.9}$$

In the training case, we wish to vary the $q(\boldsymbol{x})$ distribution in order to find the most likely latent states, which causes a corresponding change in the maximum likelihood values of the parameters R^*, C^*. In the training case, we can substitute these back to equation 4.9, giving a simplified likelihood,

$$\mathcal{L}^* = -\tfrac{DT}{2}(1 + \log(2\pi)) - \tfrac{T}{2}\log|R^*|. \tag{4.10}$$

However, in the testing case we use the latent space mapping that was optimised in the training case: we don't want to change our model when it's looking at testing data. Hence we don't get a simplified form of the likelihood, and must calculate it in full from equation 4.8. In both the training and test cases, we find that the derivatives are

$$\frac{\mathcal{L}^*}{\partial\boldsymbol{\mu}_t} \;=\; C^\top R^{-1}(\boldsymbol{y}_t - C\boldsymbol{\mu}_t)\Big|_{R=R^*,C=C^*,} \quad \text{and} \quad \frac{\mathcal{L}^*}{\partial\Sigma_{t,t}} \;=\; -\tfrac{1}{2}C^\top R^{-1}C\Big|_{R=R^*,C=C^*,} \tag{4.11}$$

4.1.4 Complexity

The number of variables to be optimised by the minimiser is $E^2(M + 2TN + 1) + E(TN + MU + U + 1)$. This limits our choice of optimisation routine -

Algorithm 1: Overview of the variational Gaussian process state-space model algorithm. In the training and testing cases, a gradient descent algorithm needs to be used to find the optimal setting of all the parameters.

Input :

- In the training case:
  - A set of inducing inputs
  - Parameters describing the mean, $\boldsymbol{\mu}_t$ and marginal and pairwise covariances $\Sigma_{t,t}$, $\Sigma_{t,t-1}$ for each latent time point.
  - A set of observed data and control inputs
- In the testing (inference) case:
  - A (fixed) set of inducing inputs
  - Parameters describing the mean, $\boldsymbol{\mu}_t$ and marginal and pairwise covariances $\Sigma_{t,t}$, $\Sigma_{t,t-1}$ for each latent time point.
  - A set of observed data and control inputs
- In the testing (prediction) case:
  - Parameters describing the mean, $\boldsymbol{\mu}_t$ and marginal and pairwise covariances $\Sigma_{t,t}$, $\Sigma_{t,t-1}$ for each latent time point, corresponding to an optimised latent state
  - A set of control inputs

Output:

- In the training case:
  - The negative log marginal likelihood $\mathcal{L}$ and derivatives with respect to inducing inputs, hyperparameters, and latent states.
- In the testing (inference) case:
  - The negative log marginal likelihood $\mathcal{L}$ and derivatives with respect to latent states.
- In the testing (prediction) case:
  - Predictions for the time points after the end of the optimised timeseries, and the joint covariance matrix of these predictions.

function Variational Gaussian Process State Space Model

if *Predicting* **then**

  Predict observations using analytic expression

else

  if *Testing (inference)* **then**

    Using fixed Ψ, K C, R,

    Calculate $\mathcal{L}$ and derivatives with respect to latent state parameters.

  else

    Update Ψ, K, C, R

    Calculate $\mathcal{L}$ and derivatives with respect to latent state parameters, hyperparameters and inducing input locations.

using a full BFGS solver requires $\mathcal{O}(V^2)$ memory to store the Hessian for V variables, which quickly becomes too much to fit in RAM. However, methods such as conjugate gradients can scale to over 100,000 variables, so this is not an insurmountable problem.

The algorithm itself takes $\mathcal{O}(E^2TNM^2)$ time in calculating the Ψ derivatives, which is the most itensive part of the algorithm. Although this rules out training models with exceptionally large E, we are able to comfortably train a model with 7 latent states, 50 inducing inputs and several hundred observations on a mid-range laptop. Further optimisation, particularly investigating stochastic gradient descent, may be able to give significant speedup.

4.2 Analytic Prediction

In the testing case, we have a seed timeseries for which we have inferred the latent states. We then wish to compute a predicted future trajectory of the system. It turns out that we can do this entirely analytically.

Predicting one point into the future is equivalent to increasing the length of the test timeseries by one, and trying to find the parameters $\boldsymbol{\mu}_T^*$, $\Sigma_{T,T}^*$ and $\Sigma_{T-1,T}^*$ which maximise the likelihood. Due to the Markovian factorisation property of $q(\boldsymbol{x})$, the parameters must only depend on the previous timepoint's mean $\boldsymbol{\mu}_T$ and marginal covariance $\Sigma_{T-1,T-1}$.

Since almost all of the NLML terms do not depend on $\boldsymbol{\mu}_T$, we have that

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_t} = \sum_{e=1}^{E} \Psi_{1e}^\top \left(K_e + \Psi_{2e}^*\right)^{-1} \frac{\partial \Psi_{1e}^*}{\partial \boldsymbol{\mu}_t} - \mathrm{tr}(Q^{-1})\boldsymbol{\mu}_t, \tag{4.12}$$

and substituting in the explicit form for Ψ_{1e}^*, we find that

$$\begin{aligned} \boldsymbol{\mu}_{T,e}^* =& \Psi_{1e}^\top \left(K_e + \Psi_{2e}\right)^{-1} |I + \Lambda_e^{-1}\Sigma_{T-1,T-1}|^{-1/2} \times \\ & \exp\left(-\frac{1}{2}(\boldsymbol{\mu}_{T-1} - \mathbf{z}_{ie})[\Lambda_e + \Sigma_{T-1,T-1}]^{-1}(\boldsymbol{\mu}_{T-1} - \mathbf{z}_{ie})\right), \end{aligned} \tag{4.13}$$

which corresponds to the mean prediction of the GP, conditioned on the inducing inputs. For the marginal $\Sigma_{T,T}$, we find that

$$\frac{\partial \mathcal{L}}{\partial \Sigma_{T,T}} = -\frac{1}{2}\mathrm{tr}Q^{-1} + \frac{1}{2}\frac{\partial}{\partial \Sigma_{T,T}}|\log \Sigma_{T-1:T,T-1:T}|. \tag{4.14}$$

Setting the derivative to zero gives us

$$\Sigma^*_{T,T} = Q + \Sigma^{*\top}_{T-1,T}\Sigma^{-1}_{T-1,T-1}\Sigma^*_{T-1,T} \tag{4.15}$$

For the pair-wise marginal,

$$\frac{\partial \mathcal{L}}{\partial \Sigma_{T-1,T}} = \sum_{e=1}^{E} \Psi_{1e}^{\top}\left(K_e + \Psi_{2e}\right)^{-1}\frac{\partial \Psi^*_{1e}}{\partial \Sigma_{T-1,T}} + \frac{1}{2}\frac{\partial}{\partial \Sigma_{T-1,T}}|\log \Sigma_{T-1:T,T-1:T}|. \tag{4.16}$$

This gives us

$$0 = \sum_{e=1}^{E} \Psi_{1e}^{\top}\left(K_e + \Psi_{2e}\right)^{-1}\frac{\partial \Psi^*_{1e}}{\partial \Sigma_{T-1,T}} - \Sigma^{-1}_{T-1,T-1}\Sigma^*_{T-1,T}\left(\Sigma^*_{T,T} - \Sigma^{*\top}_{T-1,T}\Sigma^{-1}_{T-1,T-1}\Sigma^*_{T-1,T}\right)^{-1}. \tag{4.17}$$

Substituting, we find that

$$\begin{aligned}\Sigma^*_{T-1,T} =& \Sigma_{T-1,T-1}\sum_{e=1}^{E} \Psi_{1e}^{\top}\left(K_e + \Psi_{2e}\right)^{-1}\left(\Lambda_e + \Sigma_{T-1,T-1}\right)^{-1}\left(\boldsymbol{z} - \boldsymbol{\mu}_{T-1}\right) \\ &\times |I + \Lambda_e^{-1}\Sigma_{T-1,T-1}|^{-1/2}\exp\Big(-\frac{1}{2}(\boldsymbol{\mu}_{T-1} - \mathbf{z}_{ie})[\Lambda_e + \Sigma_{T-1,T-1}]^{-1}(\boldsymbol{\mu}_{T-1} - \mathbf{z}_{ie})\Big),\end{aligned} \tag{4.18}$$

which we can then substitute in to equation 4.15 to find $\Sigma^*_{T,T}$.

We can motivate our choice of $\Sigma^*_{T,T}$ by considering the prediction of the value of $\boldsymbol{x}_T$ conditional on the value of $\boldsymbol{x}_{T-1}$. By equation 2.2 we have a conditional distribution $q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1}) = \mathcal{N}(\boldsymbol{\mu}', \Sigma')$, with $\Sigma' = \Sigma^*_{T,T} - \Sigma^{\top *}_{T-1,T}\Sigma^{-1}_{T-1,T-1}\Sigma^*_{T-1,T-1} = Q$. We can clearly see how the derived $\Sigma^*_{T,T}$ is exactly the variance for which the extra uncertainty added to the prediction of the next state is Q.

The intuition for equation 4.18 is that we are directly calculating the covari-

ance $\mathbb{E}((z_i - \boldsymbol{\mu}_{T-1})(z_j - \boldsymbol{\mu}_T)) = \mathbb{E}(z_i(z_j - \boldsymbol{\mu}_{T-1}))$, from the inducing points z. We rescale the individual contributions by $(\Lambda_e + \Sigma_{T-1,T-1})^{-1}$, and then scale up by a factor of $\Sigma_{T-1,T-1}$ after collecting the contributions from all z.

4.2.1 Predicting

4.3 Datasets

We worked on several different datasets, spanning a range of applications, number of latent dimensions, and noise regimes. The success of the implementation in a large number of different settings shows that the method is broadly applicable in many different situations, due to its weak assumptions (that the underlying process is a dynamical system), and its robustness to noise. We hope that if the process has too much noise, and it isn't possible to pin down the dynamics of the system, then the model will return an inunformative probability distribution. However, if the underlying process cannot be described as a dynamical process, then it the model may return nonsensical results.

4.3.1 1D Example

In order to explain the model in action, an introductory toy model of a nonlinear system will be used. This has a latent dimensionality of 1, and a transition function

$$f(x) = \frac{4}{5} + \left(x + \frac{1}{5}\right)\left(1 - \frac{5}{1 + \exp(-2x)}\right), \tag{4.19}$$

chosen so that as x goes to $+\infty$, the function is a straight line with slope -4, while as x goes to $-\infty$ the function is a straight line with slope 1. In the generative process we can add a certain amount of Gaussian noise, and

we can add a certain amount of observation noise. A plot of the transition function is shown in figure 4.3.1.

A plot of three simulated timeseries is shown in figure 4.3.1. The plot shows the clear difference between process and observation noise. The 1D case is complex enough to expose the real necessity of using a principled model that takes into account the difference between the two types of noise.

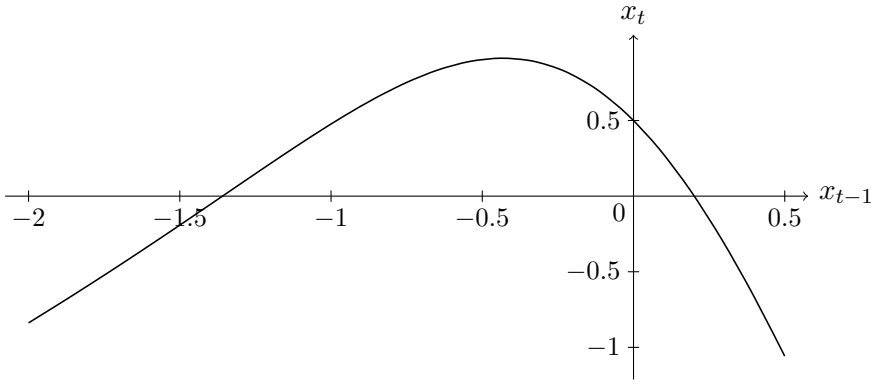


Figure 4.1: Showing the transition function used in the 1D case. Gaussian noise was added to the transition function so that the process was partially schochastic

4.3.2 Cart Pole

The cart-pole system is a very common benchmark in Reinforcement Learning (RL). It consists of a box that runs on a 1D track, which is attached to a pendulum that is allowed to swing freely. The state can be represented with four parameters: θ, the angle of the pendulum (measured downwards) and rate of change $\dot{\theta}$, along with the position of the cart x and the velocity $\dot{x}$. The transition function is extremely complicated, and so this is a good benchmark to try to learn the dynamics.

In order to make the learning process slightly easier, the angle is not represented directly with θ, as the fact that we measure the angle from 0 to 2π introduces an artificial nonlinearity in the dynamics when moving past the angular origin. Instead we expand the state space by working with the sin

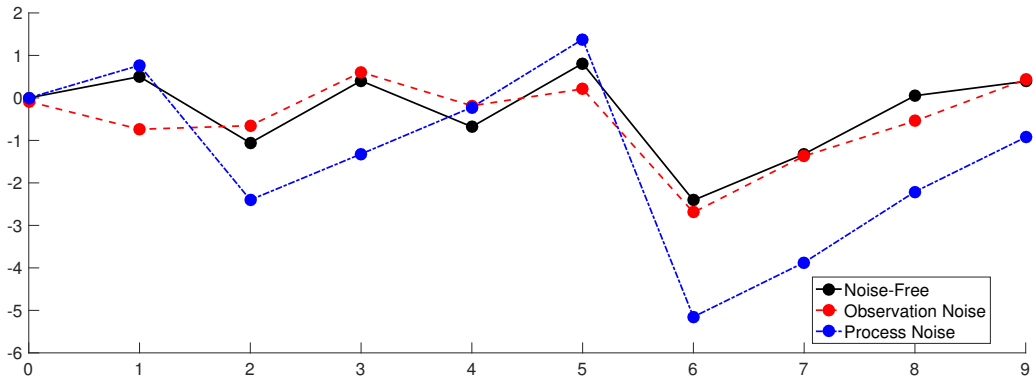


Figure 4.2: Some simulated trajectories in the 1D case. The three trajectories have the same initial condition, and have added process or observation noise, which are both Gaussian with mean zero and standard deviation one-half. The two noises have very different effects, with process noise pushing the timeseries onto completely different trajectories while the observation noise simply obscures the real state. We clearly need our models to distinguish the two noises

and cosine of the angle instead of the angle directly. The input space to the model consists of horizontal impulses applied to the cart, which results in a model with a 5D state space for the latent state, augmented with a 1D state for the inputs. However, in order to give an accurate description of the state, we need to equip the model not only with the current input, but also the previous force. This is because in the generative model, the forces are applied in a finite time which overlaps with the timesteps, such that the previous force is still being applied during the next timestep. So overall, we have a 7D problem, counting 5 dimensions from the latent space, and two from the inputs.

The complexity of this example does mean that we are unable to analytically confirm that the model has identified the correct dynamics, as we are able to do in the 1D case. However, we are able to compare predictions of the model to the results we obtain.

4.3.3 Bouc-Wen

The Bouc-Wen model is a commonly-used framework to examine many of the key features of systems exhibiting hysteresis. A Bouc-Wen oscillator in one dimension has an equation of motion

$$m_L\ddot{y} + r(y, \dot{y}) + z(\dot{y}) = u(t), \tag{4.20}$$

where

$$r = k_L y + c_L \dot{y}. \tag{4.21}$$

If the z term is neglected, the Bouc-Wen system is a straightforward damped oscillator. Adding in the z term results in a wide range of rich phenomena. Although z is not directly observable, it obeys a differential equation

$$\dot{z} = \alpha\dot{y} - \beta\left(\gamma|\dot{y}||z|^{\nu-1}z + \delta\dot{y}|z|^{\nu}\right). \tag{4.22}$$

Including z quickly turns a simple physical system into a challenging nonlinear identification task. At the 2016 workshop on nonlinear system identification benchmarks (see Noel and Schoukens 2016), the Bouc-Wen system was chosen as a new model system to test upcoming methods against. We used the dataset prepared there. Of particular importance was the addition of a small amount of noise to the results: Gaussian noise with magnitude 8×10^{-4} mm. Since the typical amplitude of the system was around 1 mm, this may not seem like a very large amount of noise. However, since we wish to infer several hidden states (the $\dot{y}$, z and $\dot{z}$ states), this could be a problem.

Furthermore, the characteristic frequency of the system was 36 Hz, and so we need a sampling frequency faster than this to capture any accurate picture of the dynamics. A frequency of 750 Hz was recommended to capture the nonlinear dynamics of the system. However, a very rapid sampling frequency introduces a new problem - if y doesn't change very much over one time step, then the majority of the observed change in the position will actually be due to noise. The model will need to learn the necessity of smoothing over the

previous few position readings to get an accurate measure in the change in position.

In fact, we found that z varied too rapidly at a sampling frequency of 750 Hz to be picked up, so a sampling frequency of 4,000 Hz was used. At this level, the typical change in the position variable was around 5×10^{-3} mm, which was only around seven times the noise level. We were stuck in a tricky balancing act of choosing a sampling rate that was fast enough to gain some idea of the dynamics whilst not so fast it was overwhelmed by noise.

Chapter 5

Implementation and Evaluation

5.1 Learning

5.1.1 1D Toy Model

Many of the challenges and achievements studied in the more complicated datasets are also present in the 1D case. The 1D case is significantly easier to study, both because of the reduced computational complexity, and because it is possible to visualise the transition function in a 2-dimensional plot. Before training the model, it's necessary to choose an initialisation. Since the model relies on a gradient-descent optimisation of thousands of parameters, an exhaustive search of this space is obviously impossible. Therefore the model is likely to be much more likely to find a good solution if initialised with a good first guess. The initialisations lie in three different areas:

Inducing Inputs Chosen to be randomly-distributed points, $\boldsymbol{v} \sim \mathcal{N}(\mathbf{0}, I)$.

Distribution of $\boldsymbol{x}$ The means were chosen to be equal to the observed data, and the variances were chosen to be isotropic Gaussians,

$$q(\boldsymbol{x}_t, \boldsymbol{x}_{t+1}) = \mathcal{N}\left((\boldsymbol{y}_t, \boldsymbol{y}_{t+1}), \begin{pmatrix} \alpha I & 0 \\ 0 & \alpha I \end{pmatrix}\right).$$

In practice, the optimisation did not seem very sensitive to the value of α.

Hyperparameters We chose the initial length-scales as 0, and the initial process noise to be 0, corresponding to unit length scale and noise level. The length-scales corresponding to the deterministic inputs were set to the log standard deviation of the inputs.

These choices for the initialisations generally scale to higher dimensions, although the initialisation of $q(\boldsymbol{x})$ is not so straightforward when there are more latent states than observed states. In that case, we have to consider if we wish to engineer some initialisations, or try to get the model to learn the latent states from scratch.

Figures 5.1.1 and 5.1.1 shows some plots of models that have been trained on the 1D system, showing the transition function, as well as the $q(\boldsymbol{x})$ distributions and the inferred states for a section of the timeseries. We can see that the model learns the correct dynamics of the system, even under quite challenging noise conditions.

After learning the dynamics, we are able to make principled forwards predictions, using the method described in section 4.2.

After a model was trained, it was used to make predictions on a previously-unseen test sequence. Five initial observations from the test sequence were shown, and these were used to infer the latent state. The final latent state was used to make a prediction, which was used as an input to the next prediction. Figure 5.1.1 shows a prediction in the low-noise regime, inluding displaying the $q(\boldsymbol{x})$ distribution. This shows one of the limitations of our variational assumption for the form of $q(\boldsymbol{x})$, as the most uncertain points are forced by their Gaussian shape to spread probability density over areas of the joint space that are not near the transition function at all. In the limit of predictions with high amounts of observation noise, we would expect that the shape of the joint probability distribution would become spread over the whole extent of the transition function. However, the Gaussian nature of the distributions means the joint distribution cannot curve to follow the region

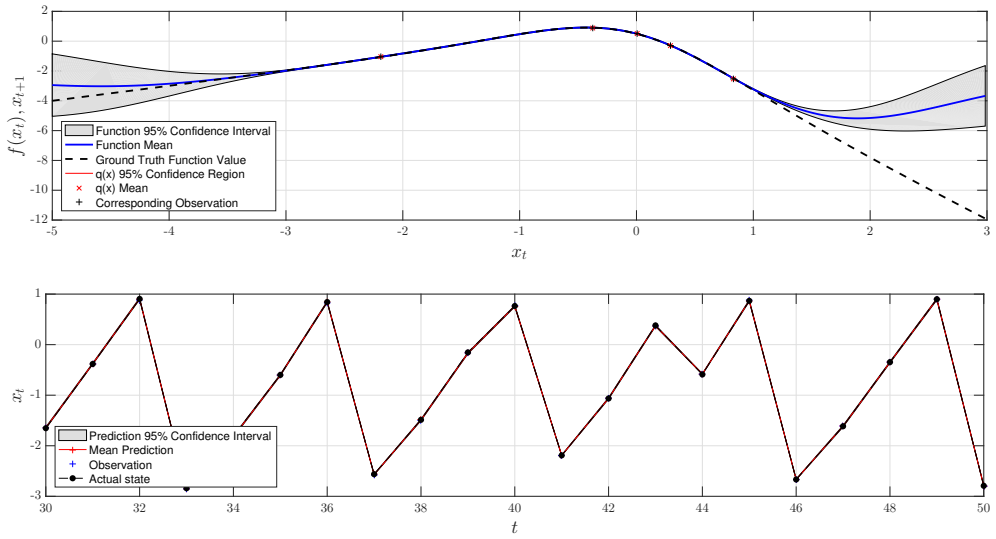


Figure 5.1: A trained model in the almost complete absence of noise ($|Q| \sim 10^{-4}, R \sim 10^{-4}$). Trained with 50 data points and 20 inducing inputs, the model learns the correct shape of the transition function. The upper panel shows the resulting Gaussian process over the transition function, as well as a few of the individual $q(\boldsymbol{x}_t, \boldsymbol{x}_{t+1})$ joint distributions. The lower panel shows the observed timeseries.

near the nonlinear transition function. Underestimating the uncertainty in its own predictions is a common hallmark of variational approaches to Bayesian learning (see R. E. Turner and Sahani 2011), and it seems like this model is no different.

We expect the advantage of the GP-SSM compared to the AR method to be the ability to distinguish between process noise and observation noise. This allows the GP-SSM to get a much more accurate inference on the initial state of the system, and so make more accurate predictions forward into the future. Given this, we would expect that the AR model would fare most badly in a regime with high amounts of observation noise, and comparatively small amounts of data. This suspicion is confirmed in table 5.1.1, summarising the performance of the methods' predictive capacity in various settings.

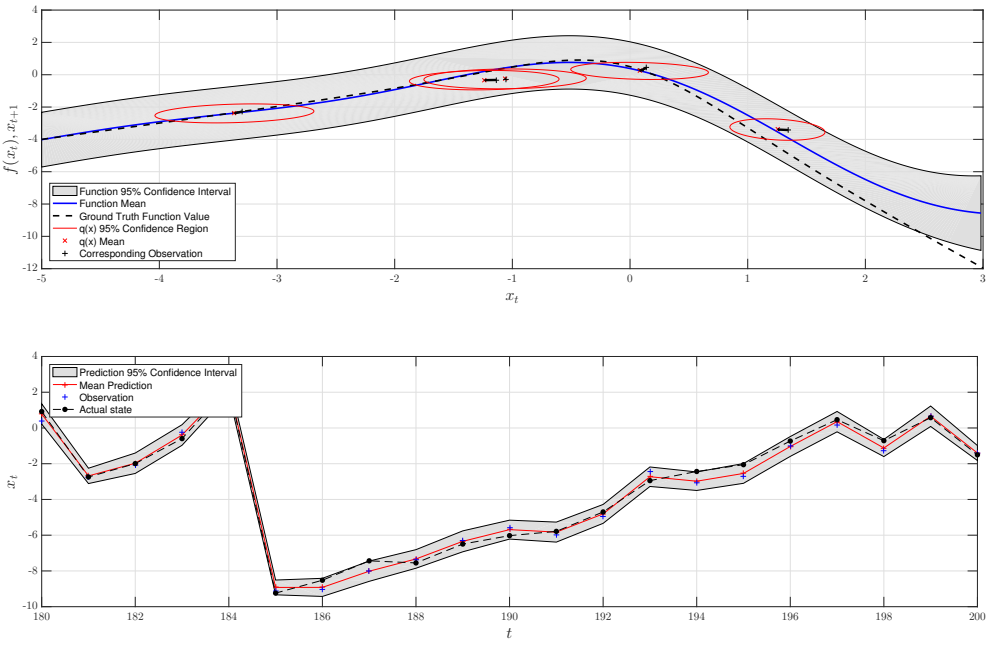


Figure 5.2: A trained model with high process noise and observation noise ($R = |Q| = 0.25$). Here the trained states are less able to pick out the correct states. However, the inferred states are generally closer to the actual state than the observation. The model includes the true transition function in the 95% confidence interval of possible dynamics. We can see that the right-hand side of the transition function is more incorrect, which we can understand by the steeper transition function leading to more variation in the output for a given variation in the input.

Evaluation

We evaluated the predictions with two approaches. We report the root mean square error (RMSE) between the observations y and the predictions $\hat{y}$, which is given by

$$e_{\mathrm{RMS}} = \sqrt{\frac{1}{N_t}\sum_{t=1}^{N_t}(\hat{y}(t) - y(t))^2}. \tag{5.1}$$

Table 5.1: Comparison of the variational GP-SSM to the order two autoregressive model in the 1D case, with process or observation noise. In both cases, the given noise had a standard deviation of 0.5 and the other noise was kept at a standard deviation of 0.01. We see that the GP-SSM is able to do well in the presence of observation noise, and not too badly in the presence of process noise. We report both the root mean square error and the negative joint log probability of the observations.

		Train Time	Test Time	RMSE (5 Ahead)	NJLP (5 Ahead)
Process Noise					
	VGPSSM	120s	6s	2.7	0.79
	GP-AR(2)	2s	<1s	10.9	1.4
Observation Noise					
	VGPSSM	115s	5s	6.0	0.90
	GP-AR(2)	2s	<1s	71	2.5

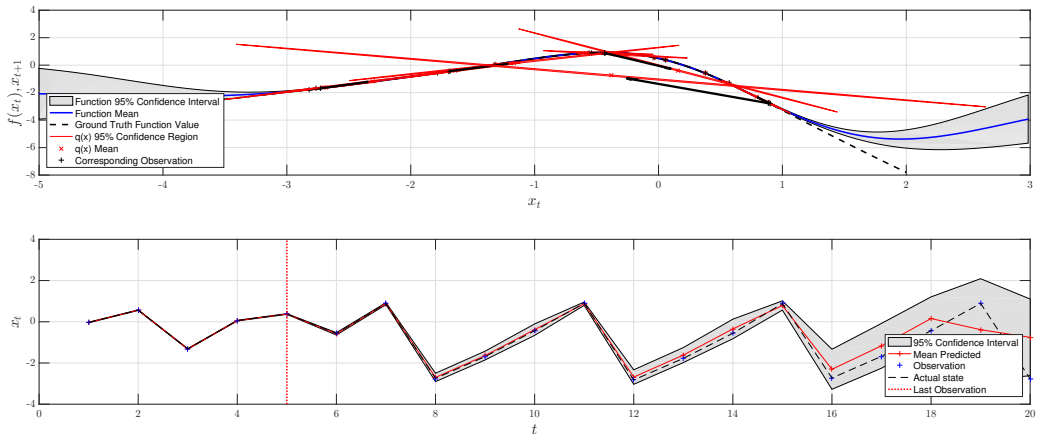


Figure 5.3: Predictions from a low-noise regime. We can see that the uncertainty in the predicted states increases as the points go into the future. Also evident is the inaccuracy made by the assumption of Gaussian errors: the distributions are forced to spread their uncertainty over areas which are far from the transition function, and so extremely unlikely for the state to ever reach those regions of the state space. However, the model does have a consistent handle on its own uncertainty, with e.g. its uncertainty decreasing dramatically at time points 7 and 11. We can see this is because the previous states are at points where the transition function is flat, so any states in the same region will map map to the same points, reducing the error.

We also report the negative log joint probability of the observed data $\boldsymbol{y}_{T_N:T}$ given the seed data in the initial timeseries $\boldsymbol{y}_{1:T_N}$.

$$-\ln\left(p(\boldsymbol{y}_T, \boldsymbol{y}_{T-1}, \boldsymbol{y}_{T-2}, \ldots | \boldsymbol{y}_1, \boldsymbol{y}_2, \ldots)\right). \tag{5.2}$$

In the case of our GP-SSM, we have to do some work to compute this, noting that we can factorise the joint probability due due to our assumption of the Markovian structure:

$$-\ln\left(p(\boldsymbol{y}_T, \boldsymbol{y}_{T-1}, \boldsymbol{y}_{T-2}, \ldots | \boldsymbol{y}_1, \boldsymbol{y}_2, \ldots)\right) = \ln p(\boldsymbol{y}_0) + \ln p(\boldsymbol{y}_1|\boldsymbol{y}_0) + \ln p(\boldsymbol{y}_2|\boldsymbol{y}_1) + \ldots. \tag{5.3}$$

Finally, we need to relate the probability of finding a point in the latent space to the observation space. We have that

$$\begin{pmatrix} \boldsymbol{x}_t \\ \boldsymbol{x}_{t+1} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu}_t \\ \boldsymbol{\mu}_{t+1} \end{pmatrix}, \begin{pmatrix} \Sigma_{t,t} & \Sigma_{t,t+1} \\ \Sigma_{t,t+1}^\top & \Sigma_{t+1,t+1} \end{pmatrix}\right). \tag{5.4}$$

The linear mapping to the observation space results in $q(\boldsymbol{y}_t) = \mathcal{N}(C\boldsymbol{\mu}_t + C', C\Sigma_t C^\top + R)$. So the action of the mapping in the joint space results in a joint probability density

$$\begin{pmatrix} \boldsymbol{y}_t \\ \boldsymbol{y}_{t+1} \end{pmatrix} = \mathcal{N}\left(\begin{pmatrix} C\boldsymbol{\mu}_t + C' \\ C\boldsymbol{\mu}_{t+1} + C' \end{pmatrix}, \begin{pmatrix} C\Sigma_{t,t}C^\top + R & C\Sigma_{t,t+1}C^\top \\ C\Sigma_{t,t+1}^\top C^\top & C\Sigma_{t+1,t+1}C^\top + R \end{pmatrix}\right). \tag{5.5}$$

We can then use equation 2.2 to get a factorisation of the joint probability density. When using the AR(2) model, we can factorise the joint probability density too, but we don't have to worry about the mapping from a latent space to the observation space, as the predictions are made directly in the observation space.

Note that we want our negative log joint probability to be as large negative as possible, corresponding to a model which is certain of the correct answer. A final point to consider is that the joint log probability gives a joint proba-

bility over all dimensions of the output, while we report the RMSE over each dimension separately. While this seems the most principled method to gauging how accurately the model's uncertainty relates to the actual error, it is not particularly easy to visualise the probability density over E-dimensional Gaussian distributions, hence the use of RMSE in addition.

The question of when to stop is an interesting point to consider. If we're attempting to predict an entire timeseries ahead, then if there is any process noise at all (which we assumed in this model), then at some point it simply won't be possible to predict ahead, as the noise compounds over time. So we should report the RMSE and joint error as we increase the number of points predicted forwards, as otherwise the total error will be dominated by the completely ignorant final few predictions.

5.1.2 Inference of Hidden States

One of the attractive features of the GP-SSM model is the possibility of inferring hidden states. The 1D system provides a very simple setting to expose some of the challenges that are likely to pose problems when extending this work to higher dimensionalities.

The lack of initialisation is an obvious problem when trying to infer the existence of unseen states. Although we may often have hints as to how we should initialise our hidden states, we would prefer to be able to not have to tweak initialisations in order to infer the correct underlying dynamics. Unfortunately in practice, without the 'hints' provided by the initialisation, the minimizer was not able to discern the dynamics of the system, and found itself in a local minimum with high process noise and observation noise. By applying a 'barrier function' (see Bartholomew-Biggs 2005 for more detail), roughly equivalent to assuming a strong prior over the magnitudes, we can not allow the model to consider the case where the process noise is too large. The results of limiting the process noise in this way are shown in table 5.2. We can see that the model correctly identifies the most likely configuration to be one with a single hidden state.

Table 5.2: Showing the effect of imposing a restriction on the process noise when optimising the negative log marginal likelihood (NLML). Here we use 100 data points, moderate noise with $|Q| = |R| = 10^{-2}$, and 30 inducing inputs. Examining the length-scales in the third case shows that the model has shut down one of the dimensions, with the C matrix dominated by the element mapping the active dimension to the likelihood.

E	Process Noise	NLML	Test $-\ln p(\boldsymbol{y}_{5:10})$
1	Unconstrained	168	16.1
1	Constrained	-202	-11.8
2	Constrained	-197	0.6

5.2 Cart Pole

A model was trained on the cart-pole system described in section 4.3.2. As in the 1D case, we show the trained model and compare the predictions to an autoregressive model. Initialisations were exactly as in the 1D case, with normalised observations for the latent points. The model is able to make good predictions, as can be seen in figure 5.2. The RMSE shows that the mean predictions are pretty similar compared to the AR model, and perhaps very slighly better than the AR model for the initial time points. However, the joint probability shows that the predictions are much more certain for the GP-SSM for the initial timesteps and then become inaccurate. This fits with what we would expect: the model structure allows the GP-SSM to work out which latent states are there, giving better predictions in the initial time steps, whilst the variational assumptions give the characteristic underestimation of uncertainty leading to overconfident erroneous elections after around 10 forward predictions.

5.2.1 Reduced Cart Pole

We can also consider the situation where we hide some of the variables, and see if the model is able to learn these hidden states. In particular, we will look at the situation where we hide the velocity and angular velocity, leaving

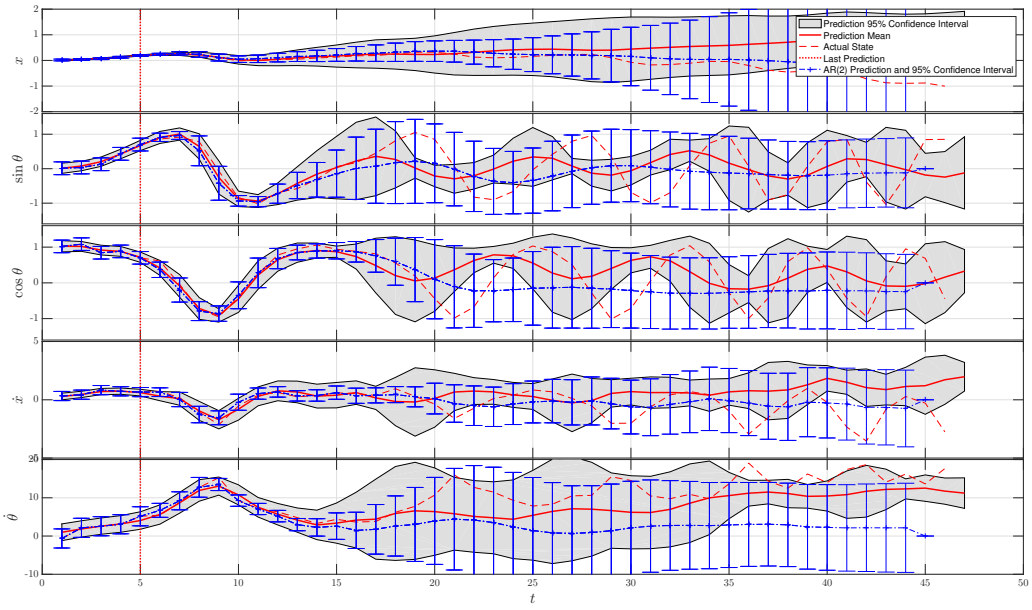


Figure 5.4: Predictions from the variational GP-SSM and an order-2 AR model. We see that both models are able to accurately capture the dynamics of the system, able to predict several points forward into the future. We also see that both methods are able to give a fairly consistent estimation of the error in their estimates. Figure 5.2 gives a quantitative account of the accuracy of both predictions

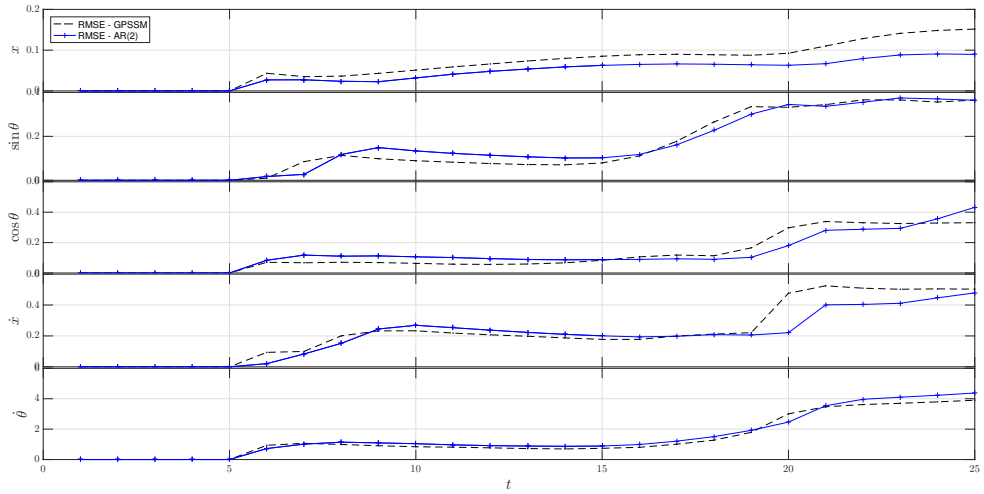


Figure 5.5: RMSE for predictions $t-5$ steps ahead on the 5d cart-pole dataset, averaged over forwards predictions from each time point in 10 test sequences. The two RMSEs are very similar - as we would perhaps expect as the AR model is in some sense the GP-SSM without latent variables.

the position, and sin and cosine of the pendulum angle.

Models with varying numbers of latent states were used, with each hidden state initialised to the difference between observed states and previously observed states: with the aim to get the model to recognise these 'velocity' terms. Results can be seen in table 5.3.

Table 5.3: Models with differing numbers of latent states on the 5-dimensional cart-pole dataset, and the optimised NLML found. We find that the model identifes three hidden states, presumably corresponding to the velocities of the three observed states.

Latent Dimension	Train $\mathcal{L}$
3	-300
4	-700
5	-1700
6	-1900
7	-1750

We had a particular focus on a model with six latent states, with the first three states initialised to the normalized observations, and the remaining

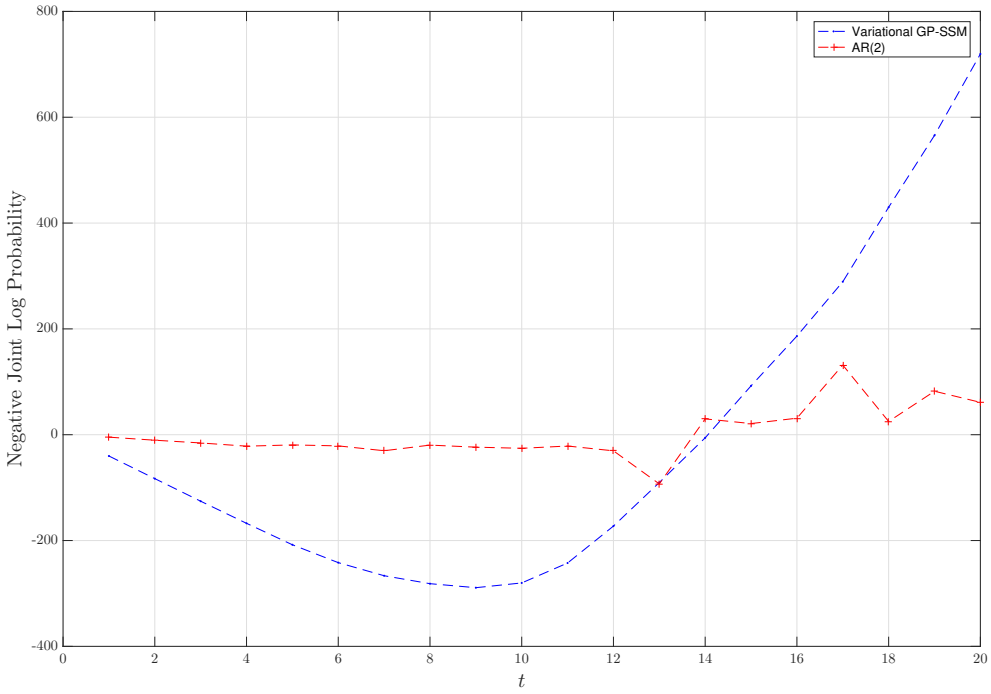


Figure 5.6: Predictive power of the two competing methods on the 5D cart-pole set. Here we see that the GP-SSM is able to make better predictions for the first few time points, but that it quickly reverses track and makes bad predictions, whilst the AR model makes fairly noncommittal predictions for all future predictions. Shown are the joint log probability t steps ahead on the 3d cart-pole dataset, averaged over forwards predictions from each time point in 10 test sequences.

three states initialised to the normalized difference between the observations and previous observations. After much optimisation, the model trained to deliver a model that used all six Gaussian Processes. We would like to be able to analyse what the model is inferring from the data, which we can do in an approximate fashion by looking at the length-scales for each Gaussian process. The model can 'shut down' an input to one of the GPs by increasing the length-scale in that dimension of the state space. We can see a plot of a badly optimised model that hit a local optimum with a completely inactive GP in figure 5.2.1.

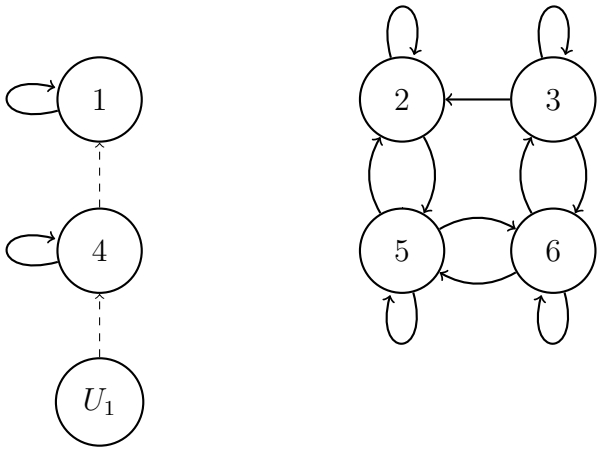
By looking at which GPs' length scales are small, we can see the logical structure of the model the GP-SSM has constructed. The logical structure of the derived model is given in figure 5.2.1.

Since some of the functions depend only one two inputs, we can plot these functions in a 3D figure. Some of those are shown in figure 5.2.1 The model seems to have essentially decoupled the system into two disjoint parts, with one GP modelling the position of the centre of mass of the cart-pole system by integrating in the applied force. This actually allows the 6D system to make better long-range predictions of x than the model with all five observables, at the expense being unable to make more confident shorter-term estimates.

The remaining four GPs are left to model the dynamics of the pendulum and any interactions with the cart. As we might anticipate, to a large degree the model's derived dynamics are not very easily understandable due to mixing between the different GPs. The states 4, 5 and 6 are not coupled to the likelihood to any degree, so they are merely acting as supporting information for the predictions of the states 1, 2 and 3.

In any case, the model is able to give good predictions going into the future, although it does not do any better than an AR(2) model except in the long range prediction of the position.

Figure 5.7: A graphical model for the conditional dependencies in the GP-SSM trained on the 3d cart-pole dataset, with 6 latent states. Looking at the mapping from latent to observation space, we see that states 1,2,3 are coupled to the corresponding observation states 1,2,3, whilst states 4,5,6 are not coupled to the observation to any degree.



5.3 Bouc-Wen

Inferring the hidden states in the Bouc-Wen model was a much more significant challenge. In the 3D cart-pole case, it is easy to understand the physical meaning of all of the observations. In the Bouc-Wen system, the governing differential equation is more complicated. Furthermore, in the Bouc-Wen case we wish to infer several unobserved states ($\dot{y}$, z, $\dot{z}$) from a single observed state, y. However, some limited sucess was achieved.

Several models were fitted, with varying numbers of latent states. For the case with one underlying dimension, the states were initialised to the observed values of y. For increasing latent dimension, the states were initialised to the difference in y values, with the aim to get the model to recognize this state as the velocity; to a rough estimate of z, obtained by finding the extra acceleration due to the z term; and to an empirical $\dot{z}$ found by taking the difference between the observed acceleration and the forces arising from the non-z terms.

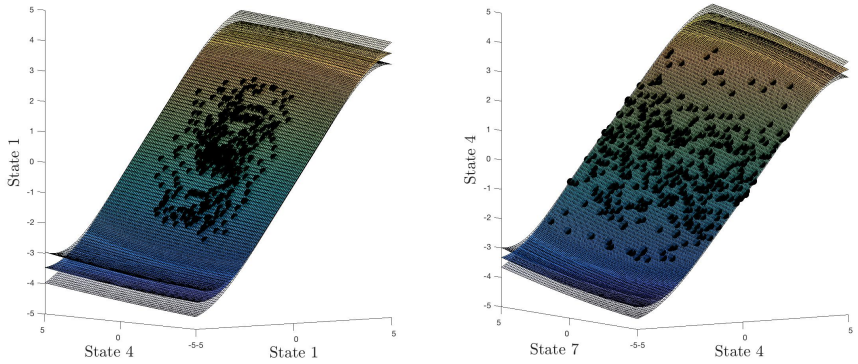


Figure 5.8: The transition functions for states 1 and 4, for the variational GP-SSM trained on the reduced cart pole dataset. These two states essentially function as an integrator for the force (which is the input state 7), encoding state 1 as the position of the center of mass. The function is known with great certainty, as the error bars (shown in faint black) are very tight around the function.

After extensive experimentation, we found that pre-training the hyperparameters while fixing the inducing inputs, then optimising the inducing inputs separately, then finally optimising all the variables jointly could ensure that the model did not quickly write off all the extra latent dimensions as noise. The results in table 5.4 were obtained with different number of latent dimensions. The most likely dimensionality is the dimensionality we would expect. Referring back to the governing equation 4.22, we can see that knowledge of both z and $\dot{y}$ is needed to calculate $\dot{z}$, which tallies with the fact that the GP-SSM only gets a substantial increase in understanding when four latent states are used, and not much of an increase with one, two or three.

The structure of the dependencies in the trained model is shown in figure 5.3. Interestingly, it does appear to be able to pick up some of the latent state, with a state 4 that doesn't depend on y. This is exactly as we'd expect from equation 4.22.

However, we wouldn't necessarily expect to see the state 3 depend on state 1, as we should be able to obtain z by integrating up state 3, without needing to use state 1 directly at all. However, since only one state is observed, some

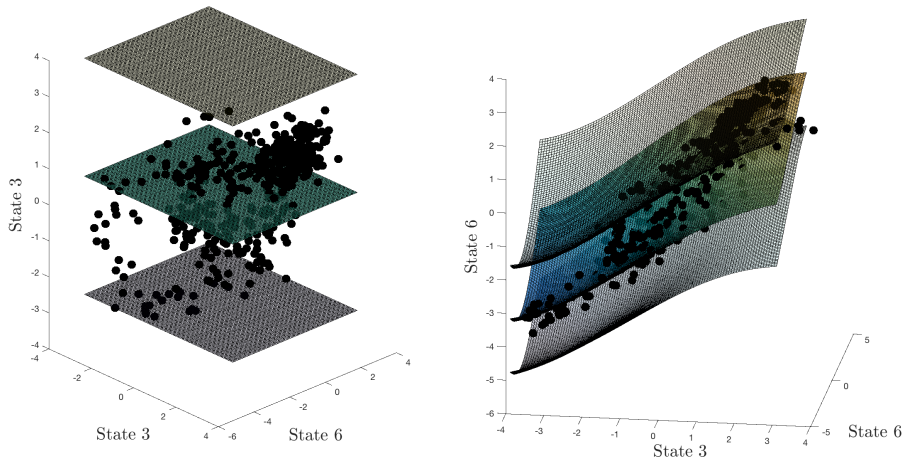


Figure 5.9: An example of a transition function for a badly optimised model of the reduced cart-pole. The transition function here has very high error bars, which imply that the function merely adds noise to the latent states. We can see how the model could shut down a GP in such a way if it wasn't providing any predictive power. In such a case we should note that the 'observation noise' in the system could be made from contributions from the R matrix and the Q matrix in the dead GP.

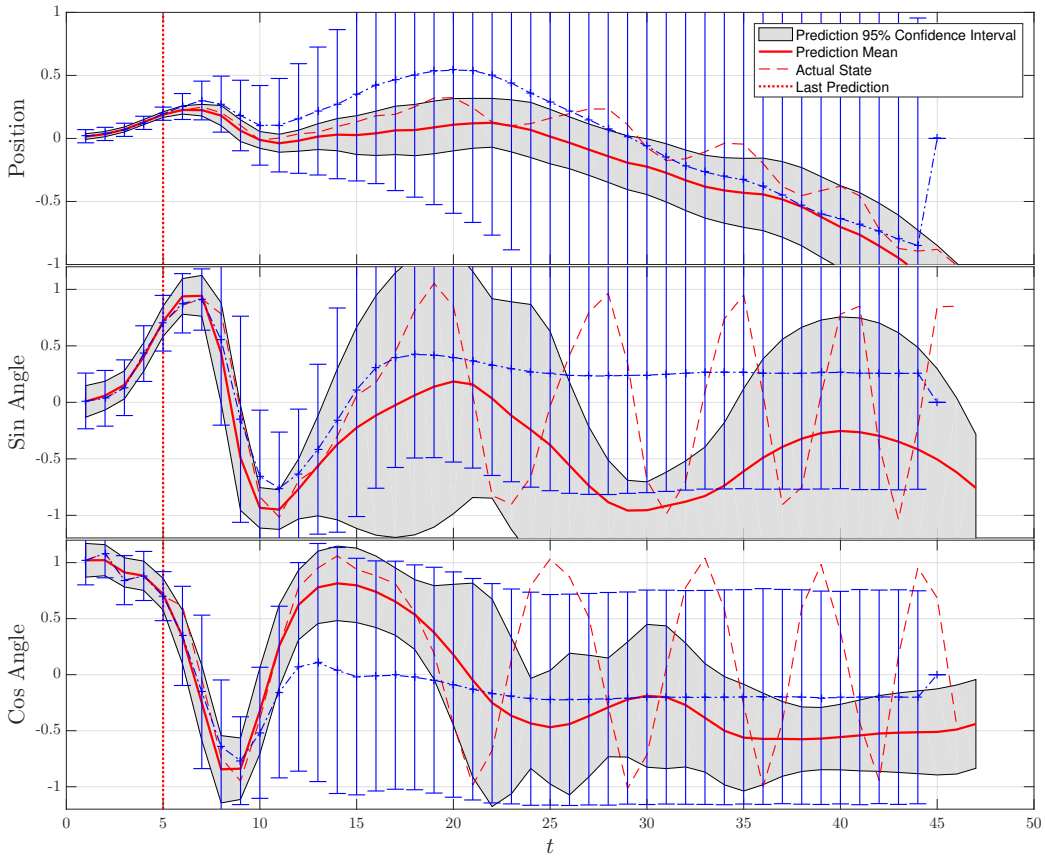


Figure 5.10: Predictions from the variational GP-SSM and an order-2 AR model. We see that both models are able to accurately capture the dynamics of the system, able to predict several points forward into the future. We also see that both methods are able to give a fairly consistent estimation of the error in their estimates. The following two plots show a quantitative analysis of the errors.

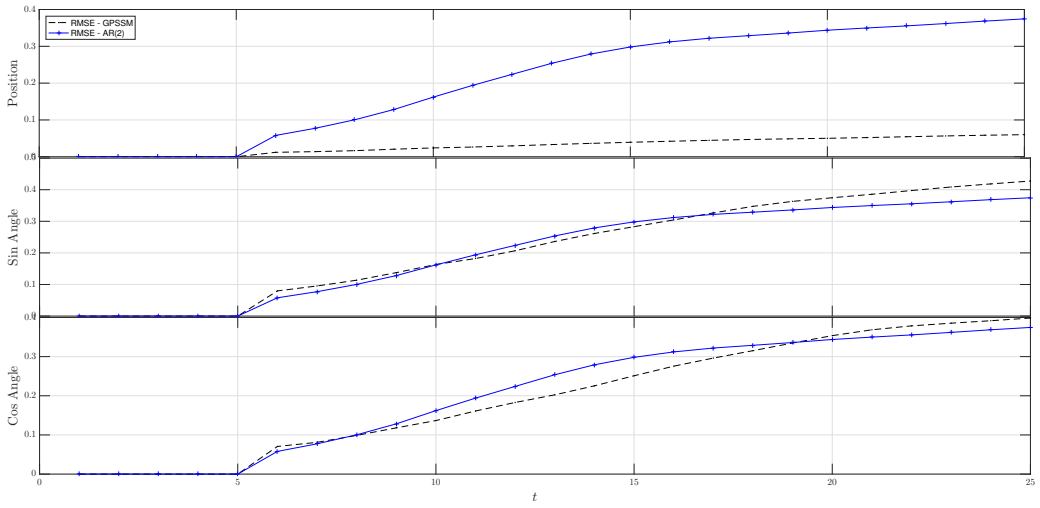


Figure 5.11: RMSE for the predictions on the 3d cart-pole dataset, averaging over forwards predictions from each time point in 10 test sequences. We see that the GP-SSM has a clear advantage over the AR(2) in predicting the position variable, but is comparable in the sin and cosine predicitons.

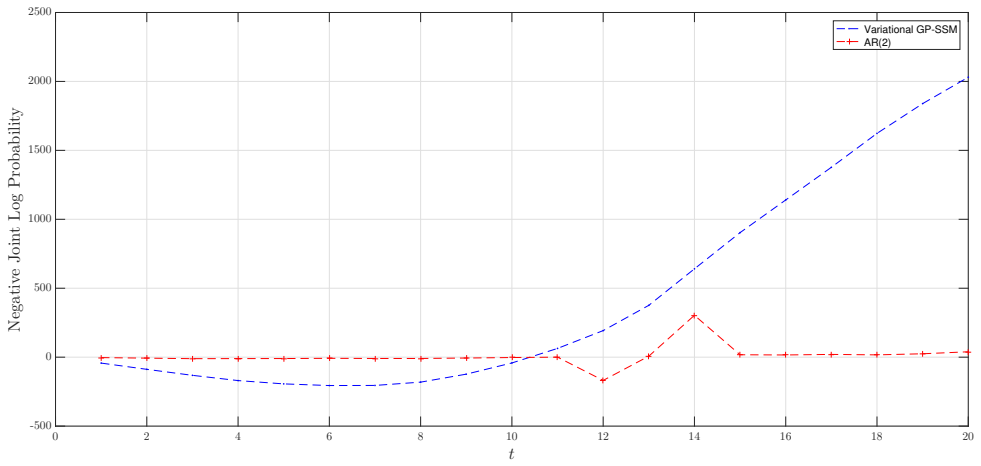


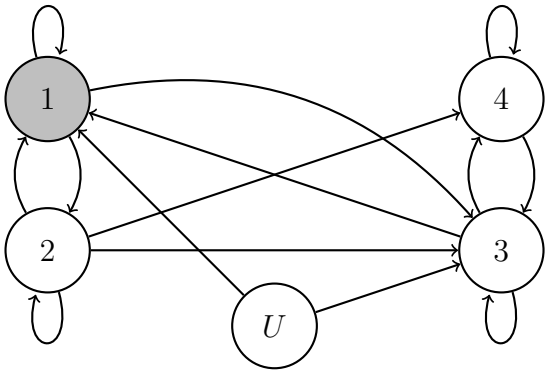
Figure 5.12: Negative joint log probability for the two predictive methods, averaging over forwards predictions from each time point in 10 test sequences, predicting t points into the future. We see that the GP-SSM's predictions are more precise and accurate at first, but that they start to become inaccurate whilst underestimating the uncertainty in the predictions. The AR(2) model does not make very precise predictions, but has a well-calibrated grasp of its own certainty.

Table 5.4: Successive models trained on a 400-timepoint subset of the BoucWen dataset, with varying underlying latent dimension. We are able to clearly see the model pick out the 4-dimensional case as the most likely. This gives us further reason to think that the GP-SSM is picking out the dynamics.

Latent Dimension	Train $\mathcal{L}$
1	-2280
2	-2285
3	-2300
4	-3790

'mixing' of the latent states is possible, and even expected. This could be what is occuring here.

Figure 5.13: The structure of the model found by the GP-SSM when trained on the Bouc-Wen dataset. The only variable that is coupled to the likelihood is state 1. Although somewhat hard to interpret, it seems as if the model is picking up the latent z state, with the state 4 (initialised with an estimate of $\dot{z}$), not depending on state 1, y, at all.



Unfortunately, forward predictions on this data set were not very good, with the predictions diverging from the truth rapidly after a few timesteps forwards. This is shown in figure 5.3.

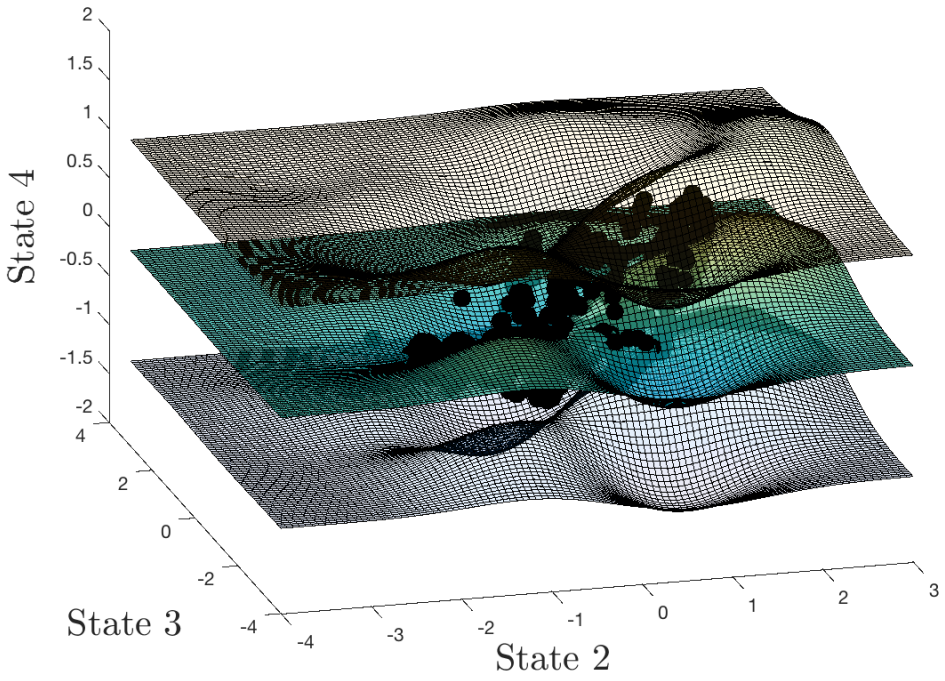


Figure 5.14: The transition function for state 4 in the Bouc-Wen dataset. The GP-SSM picks up a nonlinear transition function, albeit with a large amount of uncertainty.

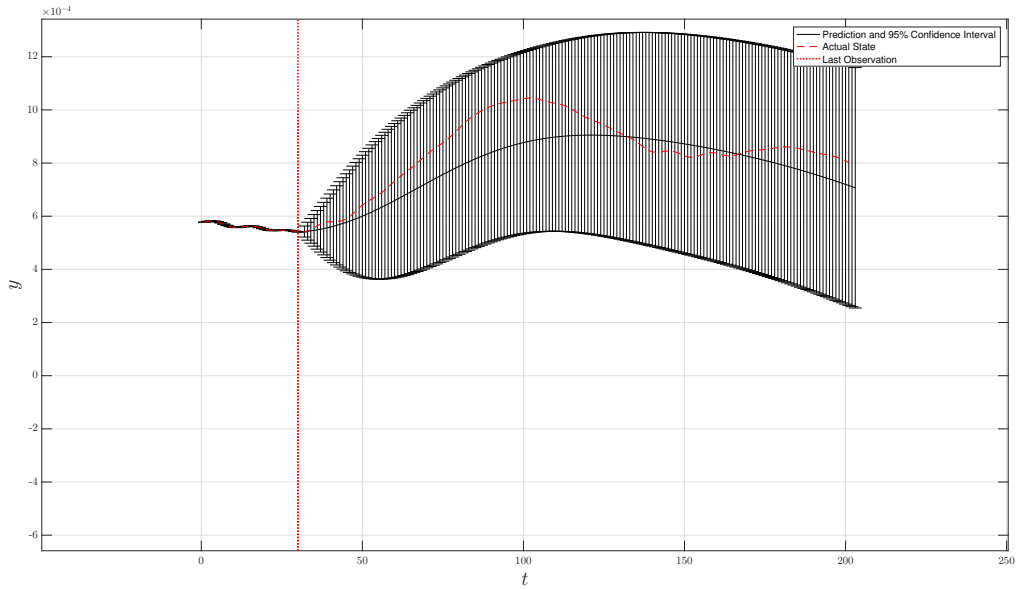


Figure 5.15: Predictions from the Bouc-Wen dataset. For the 'seed' time-series, 30 points were shown to the model, due to worries about the model being able to pick up the states of the three latent variables with only a handful of observations. Unforunately the predicitions are not very good, with the values rapidly diverging from the observations. The model does retain knowledge of its own uncertainty, which is very high.

Chapter 6

Summary and Conclusions

We have demonstrated a principled method to infer the properties of dynamical systems, the doubly variational Gaussian process state-space model (GP-SSM): combining previous work on variational approaches to GP-SSMs with a further assumption of a Gaussian latent variable distribution which allows a fully analytic lower bound on the marginal likelihood.

Previously unpublished work on this topic is put into context, and extended to deal with testing and prediction, which can be achieved fully analytically. We have shown that the variational GP-SSM is able to learn the dynamics of complicated systems such as the cart-pole, and is also able to infer underlying processes, such as integrating an impulse twice to attain a position, and finding some underlying structure in the Bouc-Wen nonlinear system identification dataset.

In a comparision to an autoregressive (AR) model, the GP-SSM was not much better than the AR in a 5D cartpole problem, whilst being much more costly to train and test. However, the system fared better than the AR model when two of the underlying variables were hidden: it was able to infer their existence and beat the AR model in initial predictions. While both the AR and GP-SSM model performed badly after a dozen or so predictions, the GP-SSM also drastically underestimated its own uncertainty (as is common

in variational approaches to Bayesian learning). As in previous work with VFE approximations, we find that progress is stymied by technical issues in performing the optimisation, including the model getting stuck in local minima, and sensitivity to initial conditions. However, some work with barrier functions seems to show promise for slowing the training rate and helping find a global minimum, even when initialisations are not specially chosen.

Future work will focus on ensuring robustness of the gradient descent optimisation of model parameters, hopefully developing a set of best practices of barrier functions and training rates that work for all data sets, reducing reliance on intuition and 'tricks' in training the model. We will also clarify the ability of the GP-SSM to learn underlying latent states, particularly looking at the complexity of the model on very high-dimensional data, which we didn't consider at all. If the model can reduce dimensionality with the same success as it can infer hidden states, the doubly variational GP-SSM could succeed as a very generally applicable, interpretable and flexible learning algorithm.

Bibliography

[1] Michael Bartholomew-Biggs. "Barrier Function Methods". In: *Nonlinear Optimization with Financial Applications*. Boston, MA: Springer US, 2005. Chap. 19, pp. 211–218. ISBN: 978-0-387-24149-4.

[2] Matthias Bauer, Mark van der Wilk, and Carl Edward Rasmussen. "Understanding Probabilistic Sparse Gaussian Process Approximations". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 2016, pp. 1525–1533. URL: http://papers.nips.cc/paper/6477-understanding-probabilistic-sparse-gaussian-process-approximations.

[3] Ivar Bendixson. "Sur les courbes definies par des equations differentielles". In: *Acta Mathematica* 24 (1901), pp. 1–88.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] Laura Dietz. *Directed Factor Graph Notation for Generative Models*. Tech. rep. Saarbrücken, Germany: Max Planck Institute for Informatics, 2010. URL: http://www.mpi-inf.mpg.de/%5C~%7B%7Ddietz/dirfactor-notation.pdf.

[6] Roger Frigola. "Bayesian Time Series Learning with Gaussian Processes". PhD thesis. University of Cambridge, 2015.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[8] Rudolph Emil Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D (1960), pp. 35–45.

[9] L. Ljung. "Asymptotic behavior of the extended Kalman filter as a parameter estimator for linear systems". In: *IEEE Transactions on Automatic Control* 24.1 (Feb. 1979), pp. 36–50. ISSN: 0018-9286. DOI: 10.1109/TAC.1979.1101943.

[10] Andrew McHutchon. "Nonlinear Modelling and Control using Gaussian Processes". PhD thesis. Cambridge, UK: University of Cambridge, Department of Engineering, 2014. URL: ..

[11] J.P. Noel and M. Schoukens. "Hysteretic benchmark with a dynamic nonlinearity". In: *Workshop on Nonlinear System Identification Benchmarks, Brussels, Belgium, April 25-27, 2016*. 2016, pp. 7–14. URL: http://homepages.vub.ac.be/~mschouke/benchmarkBoucWen.html.

[12] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN: 026218253X.

[13] C.E. Rasmussen. "Variational Gaussian Process Timeseries Inference". Jan. 2016.

[14] Sam Roweis and Zoubin Ghahramani. *An EM algorithm for identification of nonlinear dynamical systems*. 2000.

[15] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587 (2016), pp. 484–489.

[16] Edward Snelson and Zoubin Ghahramani. "Sparse Gaussian Processes using Pseudo-inputs". In: *Advances in Neural Information Processing Systems 18*. Ed. by Y. Weiss, P. B. Schölkopf, and J. C. Platt. MIT Press, 2006, pp. 1257–1264. URL: http://papers.nips.cc/paper/2857-sparse-gaussian-processes-using-pseudo-inputs.pdf.

[17] Michalis K. Titsias. "Variational Learning of Inducing Variables in Sparse Gaussian Processes". In: *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009*. 2009, pp. 567–574. URL: http://www.jmlr.org/proceedings/papers/v5/titsias09a.html.

[18] Richard E Turner and Maneesh Sahani. "Two problems with variational expectation maximisation for time-series models". In: *Bayesian Time series models* (2011), pp. 115–138.

[19] Ryan Turner, Marc Peter Deisenroth, and Carl Edward Rasmussen. "State-Space Inference and Learning with Gaussian Processes". In: ed. by Yee Whye Teh and Mike Titterington. Vol. 9. W & CP. Chia Laguna, Sardinia, Italy, May 2010, pp. 868–875. URL: ..

[20] Eric A Wan and Rudolph Van Der Merwe. "The unscented Kalman filter for nonlinear estimation". In: *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*. Ieee. 2000, pp. 153–158.

[21] Jack M. Wang, David J. Fleet, and Aaron Hertzmann. “Gaussian Process Dynamical Models for Human Motion”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 30.2 (Feb. 2008), pp. 283–298. ISSN: 0162-8828.