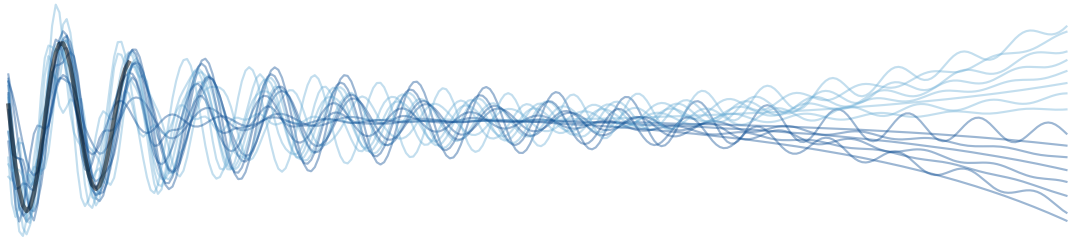
THE UNIVERSITY
of EDINBURGH

Multi-Task Dynamical Systems:

Customising Time Series Models



Alex Bird

Doctor of Philosophy

School of Informatics

University of Edinburgh

2020

Abstract

Time series datasets are usually composed of a variety of sequences from the same domain, but from different entities, such as individuals, products, or organizations. We are interested in how time series models can be specialized to individual sequences (capturing the specific characteristics) while still retaining statistical power by sharing commonalities across the sequences.

The major contribution of this thesis is the multi-task dynamical system (MTDS); a general methodology for extending multi-task learning (MTL) to time series models. Our approach endows dynamical systems with a set of hierarchical latent variables which can modulate *all* model parameters. To our knowledge, this is a novel development of MTL, and applies to time series both with and without control inputs. This thesis provides a formulation of three example MTDS models: a multi-task linear dynamical system, a multi-task recurrent neural network (RNN) and a multi-task pharmacodynamic model.

The use of MTDS models is demonstrated in detail on three datasets. For a synthetic physical dataset, we demonstrate that our learning and inference algorithms are able to recover a model that is indistinguishable from the optimal one, and provide substantial improvements in prediction over gated RNN models. We apply the MTDS to motion-capture data of people walking in various styles, which results in improved performance, style transfer and the ability to *morph* between walking styles. Finally, using patient drug-response data, we show that our MTDS approach can result in significant improvements over modern pharmacodynamic (PD) models without deviating from the trusted PD model form.

Acknowledgements

A PhD is not an easy journey for most of us, and I would like to take the opportunity to thank the many people who have made it possible for me.

Most importantly, I owe a great debt of gratitude to my supervisor, Chris Williams, whose intuition, insight and ideas have been pivotal in the formulation of this thesis. Indeed it is Chris' keen eye for useful application of ML methods which provided the impetus for this thesis' topic. I would like to note in particular the generosity of his time throughout my studies, especially given our geographical separation. I also extend my thanks to Chris Hawthorne, who generously 'inducted' me into the field of anaesthetics, and provided the data for our pharmacodynamics project as well as much helpful discussion.

The Alan Turing Institute proved an exciting and stimulating environment to complete my PhD, and importantly, provided the funding for this work. I am enormously grateful for this, and all who made it happen, particularly at the nascent stage of the institution. My time there exposed me to a far wider variety of ideas and perspectives in the area of 'data science' than would plausibly have happened in a traditional academic setting. Despite a wide range of long and stimulating discussions with many members of the institute, I must unfortunately refrain from thanking any by name on account of the sheer numbers involved.

And last, but not least, my deepest gratitude goes to those who have provided community over the course of my PhD, not least my parents, my brother and sister-in-law, and the TJ's. It would be very hard to keep going over these years without the social counterbalance and support of wonderful people. Thanks most of all to my parents who have always been supportive, whatever I ended up doing, and for bailing me out halfway through my MSc work, when no bank would give me a loan. Finally, in contrast to the subject of this work, I dedicate it to a constant, *et lux in tenebris lucet*.

Declaration

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

(Alex Bird)

Contents

Abbreviations

This document contains many abbreviations (initialisms and acronyms), most of which are standard in the literature. A table of important abbreviations is provided below for completeness:

Abbreviation	**Definition**
AdaIS	Adaptive importance sampling
ADF	Assumed density filtering
AIC	Akaike information criterion
AMIS	Adaptive mixture importance sampling
AR(X)	Autoregressive (with eXogenous inputs)
ARMA	Autoregressive moving average
BIC	Bayesian information criterion
BIS	Bispectral index (vital sign)
BPTT	Back-propagation through time
CEC	Constant error carousel (an aspect of LSTMs)
DHO	Damped harmonic oscillator
ELBO	Evidence lower bound
EM	Expectation maximization
ESS	Effective sample size
FA	Factor analysis
FK	Forward kinematics (animation/robotics)
GAN	Generative adversarial network
GLME	Generalized linear mixed effects
GMM	Gaussian mixture model
GP	Gaussian process
GPLVM	Gaussian process latent variable model
GRU	Gated recurrent unit (a form of RNN)
HMC	Hamiltonian Monte Carlo
IS	Importance sampling
IQR	Inter-quartile range
IWAE	Importance weighted autoencoder
KL	Kullback-Leibler
LDS	Linear dynamical system
LSTM	Long short term memory (a form of RNN)
MA	Moving average
MAP	Maximum a posteriori
MC	Monte Carlo
MCMC	Markov chain Monte Carlo

MCO	Monte Carlo objective
ME	Mixed effects
MLP	Multi-layer perceptron
MLE	Maximum likelihood estimation
MSE	Mean squared error
MT(L)	Multi-task (learning)
MTDS	Multi-task dynamical system
NLL	Negative log likelihood
NUTS	No U-turn sampler
ODE	Ordinary differential equation
PCA	Principal components analysis
PK	Pharmacokinetic
PD	Pharmacodynamic
ReLU	Rectified linear unit
RMSE	Root mean squared error
RNN	Recurrent neural network
RTS	Rauch-Tung-Striebel (algorithm for state smoothing)
SGD	Stochastic gradient descent
SSM	State space model
ST(L)	Single task (learning)
SVD	Singular value decomposition
TL	Transfer learning
VAE	Variational autoencoder
VAR	Vector auto-regressive
VAR(X)	Vector auto-regression (with eXogenous inputs)
VGG	Visual Geometry Group (Oxford), refers to convnet architecture
VI	Variational inference
SMC	Sequential Monte Carlo

CHAPTER 1

Introduction

This thesis concerns the personalisation or customisation of models in the context of time series data. A time series dataset often consists of multiple sequences which are similar, but with certain characteristic differences between them, forming a related 'family' of sequences. Examples include the differing styles of handwritten text (Graves, 2013), the differences in human walking style between individuals (Ghosh et al., 2017), or the differing response of patients to an anaesthetic agent (Bird et al., 2019). In all these cases, the individual sequences share common features, but also exhibit some idiosyncratic characteristics. The central question of this thesis is how such variation can be captured successfully and efficiently.

This inter-sequence variation is not limited purely to human individuality; it can arise from a great many sources, and is common to most collections of time series data. For instance, there may be high level similarities in data derived from different hospitals or care homes, but there will also be inter-organisational differences based on geography, demographics, specialism and staffing. Different product categories in retail data may react to seasonal or promotional events differently. Longitudinal surveys of animals may reveal differences in population growth of predators and prey based on species and location. Even mechanical objects, such as wind turbines, can often exhibit unique operational characteristics.

A concrete example of such 'similar-yet-distinct' time series data is provided in Figure 1-1. The panels show five days of electricity consumption from 3 different clients from Dua and Graff (2017). There is some commonality between all clients (such as a daily seasonality, with the lowest consumption at night), but also major differences, such as the overall magnitude of usage, different weekday shapes (such as spikes in the morning and/or evening) and weekend usage. No side information (such as demographics) is provided with this dataset, and hence a time series model cannot use external supervision to 'explain' these differences.

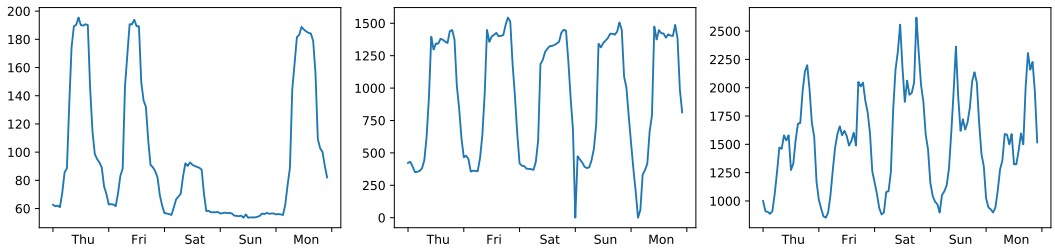


Figure 1-1: Electricity usage (kilowatt hours) of three clients in Q1 2014 from Dua and Graff (2017).

1.1 Modelling time series data

How can this inter-sequence variation be modelled? It is difficult to provide an answer to this question in the abstract, due to the wide variety of models available for time series data. In order to give the necessary context, let us first provide a brief (non-exhaustive) history of time series modelling.

Systematic analysis of time series data did not perhaps appear until Graunt's analysis of the Bills of Mortality in London in 1662 (Glass, 1964), but records of time series data can be found as far back as the Zhou dynasty (1046 – 771 BCE; see Xu, 1989). The most prominent examples of early time series models are of physical data, beginning in earnest after the advent of calculus in the 17th century. Well-known examples include Newton's application of differential equations to Kepler's three laws (Hirsch et al., 1974), or Verhulst's logistic model of population growth (Bacaër, 2011). Formalising such mathematical modelling into the modern field of continuous-time dynamical systems owes much to Poincaré's work in the late 19th century (Aubin and Dalmedico, 2002).

However, most sources of time series data follow no obvious (or at least tractable) natural law. In answer to this problem, modelling of *stochastic* time series data began to gain traction in the 1920s via the autoregressive theory developed by Yule. ARMA models were proposed later that same decade by both Yule and Slutsky, although it was Wold who later proved their theoretical validity (for the history of this period, see Makridakis, 1976). From the 1950s onwards a variety of other dynamic models proliferated, such as forms of exponential smoothing (see §1.4, West and Harrison, 2006 for further details), unified in part by Harrison and Stevens (1971) via use of a general dynamic linear model. This 'state space' approach was enabled by use of the Kalman filter (Kalman, 1960).

The last third of the 20th century saw further development of 'state space models' (SSMs), including discrete latent variables (e.g. Rabiner, 1989) and mixed continuous and discrete variables (e.g. Shumway and Stoffer, 1991). SSMs are a convenient form of time series model, encompassing many of the above methods, and can even be used as an approximation to continuous-time dynamical systems (see e.g. Section 2.2.2.7). More recently, modern 'neural' methods have facilitated the modelling of highly structured complex sequences, such as natural language, speech, music and video (see Goodfellow et al., 2016,

chs. 10, 12).

For many of the above model types (including SSMs), it is common to use a 'one-size-fits-all' model applied to all sequences in the dataset, but this fails to specialise to the individual-level characteristics. Alternatively, one might learn one model per sequence. In the case of the data from Figure 1-1, the former approach results in inappropriate averaging over many orders of magnitude, and the latter approach may perform poorly due to lack of data. In contrast, modern 'black box' models (such as a recurrent neural network, or RNN) may be able to adapt to the inter-sequence variation. But this is an unsatisfying solution, which limits the *types of model* that can be used, as well as our understanding of *how* the sequences are being modelled. Delegating the responsibility of modelling to a 'black box' is undesirable in many circumstances, and we wish to explore a more generally applicable approach.

1.2 A multi-task approach

Our observation is that the modelling of various related sequences is a similar problem to that of *multi-task learning* (MTL), usually encountered in the iid context.[1][2] MTL is an approach for improving the performance of multiple related *tasks*, which are most commonly standard supervised learning problems. This improvement is achieved by sharing information about the learned relationships between the task models. The models are encouraged to use similar relationships to each other, resulting in a reduction in overfitting – especially for tasks with small datasets.

A standard iid supervised learning scenario can be described as follows. The goal is to learn a model f from a dataset $\mathcal{D} = (\{\mathbf{u}_1, \ldots, \mathbf{u}_T\}, \{\mathbf{y}_1, \ldots, \mathbf{y}_T\})$ consisting of inputs and outputs respectively.[3] This is performed such that f approximates the relationship between inputs and outputs, i.e.:

$$\mathbf{y}_t \approx f(\mathbf{u}_t), \tag{1.1}$$

for all $t = 1, \ldots, T$. The model f is usually learned via empirical risk minimisation (ERM), where a loss function $\mathcal{L}$ measuring the mismatch between $f(\mathbf{u}_t)$ and $\mathbf{y}_t$ is minimised over all training examples in $\mathcal{D}$, i.e.

$$f^\star = \arg\min_{f \in \Omega} \frac{1}{T} \sum_{t=1}^{T} \mathcal{L}\left(f(\mathbf{u}_t), \mathbf{y}_t\right), \tag{1.2}$$

[1] iid refers to 'independent and identically distributed' random variables.

[2] For clarity, the phrase 'iid context/setting' in this thesis refers to modelling assumptions which include mutual independence of observations within a dataset, and iid model residuals.

[3] It is conventional to use $\mathbf{x}$ for the input variable in supervised learning; we follow the convention from the time series literature where $\mathbf{x}$ is reserved for the latent state, and $\mathbf{u}$ is used for control inputs. For similar reasons we will use the index set $t = 1, \ldots, T$ for the datapoints.

for some model class Ω.

If there are N tasks, described by datasets, $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(N)}$, the ERM approach may learn a model $f^{(i)}$ for each dataset $\mathcal{D}^{(i)}$. However, when the N tasks are related in some way, this is not an efficient use of the data: each model is learned in isolation and similarities between the tasks are ignored. The contribution of MTL is to couple the model training such that the $\{f^{(i)}\}_{i=1}^{N}$ are in some sense 'similar'. Most approaches achieve this by learning a restricted model class $\mathcal{H}$ such that $f^{(i)} \in \mathcal{H}$ for all $i = 1, \ldots, N$ (see Section 2.3 for some common approaches). The aim is to make $\mathcal{H}$ small enough to reduce overfitting, but large enough to capture the differences between tasks.

In this thesis we extend MTL to time series models. In this case, each time series is now considered as a 'task dataset', requiring its own bespoke model. By sharing information between the tasks (analogously to the iid case), we can learn powerful models despite the limited data of each time series. This is nevertheless more challenging than the iid case, notably because time series models must learn a sequence of models: $\mathbf{y}_1 \approx f_1(\mathbf{u}_1)$, $\mathbf{y}_2 \approx f_2(\mathbf{u}_{1:2}, \mathbf{y}_1)$, $\ldots$, $\mathbf{y}_T \approx f_T(\mathbf{u}_{1:T}, \mathbf{y}_{1:T-1})$ rather than a single model $f(\mathbf{u}_t), t = 1, \ldots, T$. This requires us to share relationships between models *within* each time series as well as *between* each time series. We present our methodology for the application of MTL to general classes of dynamical systems in Chapter 3. Naturally, we call such models 'multi-task dynamical systems' (MTDSs).

We must note that this is not the first attempt to apply MTL to time series models, but other approaches tend to be limited or specialised to a particular model class. Perhaps the best known example is for use with RNNs: one can append a unique index for each sequence to the input (perhaps via *one-hot* encoding). However, this is not a general purpose approach, cannot generalise to new sequences, and obtains mixed results in practice (see e.g. Spieckermann et al., 2015; Martinez et al., 2017). For other existing work, see Section 3.7.

1.3 Online customisation

Application of MTL may result in good performance on the data in the training set, but it does not immediately apply to novel sequences encountered at test time. Novel sequences are to be expected: for instance, new patients at a hospital, new customers using an online platform, or new clients using an energy provider. How can these benefit from the MTL approach?

As in the case of iid data, application of MTL to time series data results in learning a model class $\mathcal{H}$, containing at minimum all of the multi-task models in the training set. Consider this to be an estimate of the *sequence family* underlying the data generating process; we may then hope that any novel time series data can be well approximated by some model $f' \in \mathcal{H}$. Hence, for a novel dataset $\mathcal{D}'$, we search $\mathcal{H}$ for the most appropriate model.

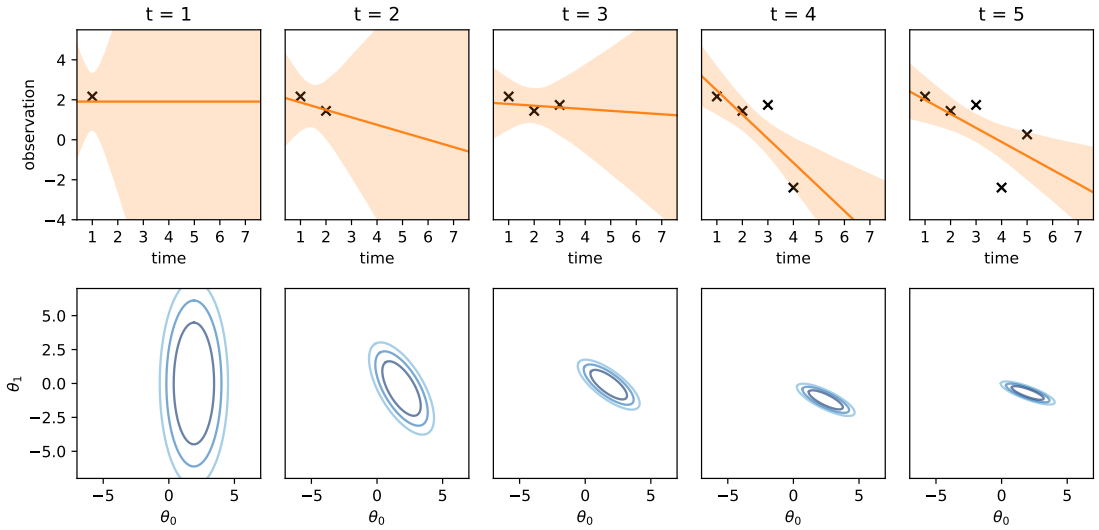


Figure 1-2: (*top*) Bayesian linear regression of observations at $t = 0, \ldots, 4$, showing 90% credible intervals, units are kWh; (*bottom*) posterior over the slope and intercept, with the level curves showing the central 80%, 95% and 99% credible interval.

To avoid overfitting $f' \in \mathcal{H}$, one can use Bayesian inference, which further presents the opportunity of *online updates*, where f' achieves increasing customisation to the sequence as the length T of the test data increases.

This online customisation is somewhat similar to the well-known online approach to Bayesian linear regression. A simple example is the linear model $y_t = \theta_0 + \theta_1 u_t + \epsilon_t$, where a target y_t is predicted from an affine function of u_t in the presence of Gaussian noise (ϵ_t). By considering $\boldsymbol{\theta} = (\theta_0, \theta_1)$ as a latent variable with prior $p(\boldsymbol{\theta})$, one can iteratively improve one's estimate of the parameters via the update:

$$p(\boldsymbol{\theta} \,|\, \mathbf{u}_{1:t}, \mathbf{y}_{1:t}) \;\propto\; p(\mathbf{y}_t \,|\, \mathbf{u}_t, \boldsymbol{\theta})\, p(\boldsymbol{\theta} \,|\, \mathbf{u}_{1:t-1}, \mathbf{y}_{1:t-1}). \tag{1.3}$$

This update can be calculated in closed form for all t due to the linear Gaussian assumptions. An illustration is given in Figure 1-2 for $u_t = t$, where the top panel shows for increasing t the model fit and predictions for $\{y_{t+k} : k \geq 1\}$ with 90% credible intervals. With more observations, the belief state of $\boldsymbol{\theta}$ contracts around the optimal values as shown in the bottom row of Figure 1-2.

How can we perform online customisation for multi-task dynamical models? We assume a parametric form, where the model has parameters $\boldsymbol{\theta}$, together with a prior, $p(\boldsymbol{\theta})$, which is commonly available from the MTL training. Similar to the above, we can recursively update the belief state of the parameters by incorporating the observations at time t via:

$$p(\boldsymbol{\theta} \,|\, \mathbf{u}_{1:t}, \mathbf{y}_{1:t}) \;\propto\; p(\mathbf{y}_t \,|\, \mathbf{u}_{1:t}, \mathbf{y}_{1:t-1}, \boldsymbol{\theta})\, p(\boldsymbol{\theta} \,|\, \mathbf{u}_{1:t-1}, \mathbf{y}_{1:t-1}). \tag{1.4}$$

However, due to the more complex dependency structure and generally a nonlinear/non-

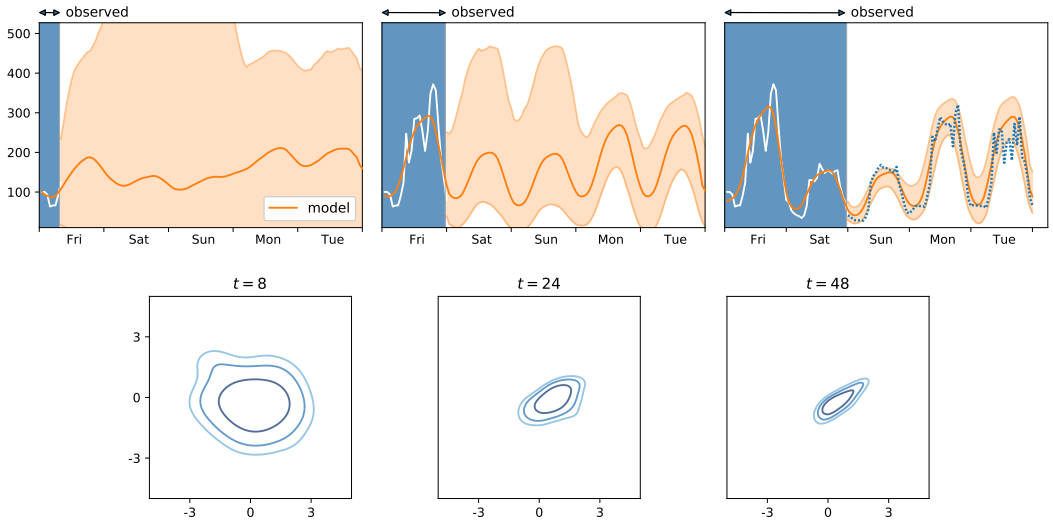


Figure 1-3: (*top*) Online inference of the electrical consumption at $t = 8, 24, 48$ hours. The observed section is highlighted in blue, and the true sequence continuation is shown in dotted blue in the final panel. (*bottom*) Posterior distribution of the dynamical parameters (kernel density estimate of the first 2 principal components). As per Figure 1-2, the level curves show the 80%, 95%, 99% CIs.

Gaussian model form, Equation (1.4) is much more challenging to compute than in the case of linear regression. We present a practical approach for its computation in Chapter 3.

Figure 1-3 provides an example of online customisation for a multi-task RNN with $\boldsymbol{\theta} \in \mathbb{R}^{3360}$ trained on the electricity dataset of Dua and Graff (2017). The input data consists of the day of the week, and day of the month. After the first 8 hours ($t = 8$), there is high uncertainty about the parameters, and hence the forecast (top left). However, after 24 hours, we can see the presence of a daily oscillation, and an approximate idea of its magnitude. After 48 hours (top right), there is enough information to refine this further, including information about the weekend consumption. The model is able to capture both the weekday and weekend level, and the asymmetric nature of the daily shape, which peaks late into the evening. The belief state over the model parameters is shown in the bottom panels, which shows increased certainty with increasing t. (Note that only the first two principal components of this distribution are shown due to the higher dimensional nature of $\boldsymbol{\theta}$ here: these encompass 64% of the posterior variation.)

This is a fundamentally different approach to handling uncertainty than existing probabilistic dynamical models such as state space models and Gaussian Processes (for more on which, see Section 2.2). These models are well known for 'data fusion', incorporating input and observational data over time, but this typically results only in local adjustments to the fit. The unique characteristics of a sequence are assigned to local aberration rather than consistent idiosyncrasy, and the long term prediction will usually return to a simple mean function. Our approach retains the inferred customisation for arbitrarily long forecasts by capturing it as part of the model f (a *static* quantity) rather than a *dynamic latent state*.

1.4 Contributions and benefits

This major contribution of this thesis is applying MTL to the class of dynamical systems. In detail, our contributions split into three areas: methodology, applications, and empirical evaluation of benefits.

Methodology We provide the methodological basis for learning and forecasting from MTDS models, including:

- **Learning and inference methods** We provide two practical methods for learning general MTDS models. For inference, our online approach is (to our knowledge) a novel and efficient approach to Monte Carlo inference in a wider class of static problems (Chopin, 2002), and may have further application elsewhere.
- **Specific model parameterisations** Some care must be taken when applying MTL to dynamical systems. We provide examples of SSMs, RNNs and a pharmacodynamic model within Chapters 3-6 together with discussion of their motivation.

Applications The benefits of the MTDS depend especially on the *amount of data*, the *comparison point* (i.e. allowable alternative models), and *domain-specific* concerns. Our three detailed studies, outlined below, provide insight into these on a variety of applications:

- **1-D (synthetic) physical data** using linear dynamical systems (Chapter 4). This enables us to understand how to apply the MTDS approach to SSMs, how this compares to existing methods (including RNN approaches), and whether our inference and learning algorithms can recover an approximately optimal representation of a sequence family.
- **64-D motion capture data** of differing walking styles using RNNs (Chapter 5). This provides a study of how to apply the MTDS to flexible neural models, and the advantages of doing so. In particular, we demonstrate that the MTDS can perform style transfer, and style morphing over time.
- **3-D drug response data** using pharmacodynamic models (Chapter 6). In this example, we show that we can achieve state-of-the-art results for a healthcare problem while remaining within the model class of existing medical practice. This is a practical step along the journey to personalised drug infusions.

Benefits The specific benefits are elaborated in the relevant chapters, and an overall summary is given in Chapter 7. A brief summary is provided below.

- **Performance gains** over SSMs, and RNNs under limited data (Chapters 4, 5, 6).

- **Increased interpretability over RNNs** (via use of standard dynamical systems) while retaining similar or better performance (Chapters 4, 6).
- A **low-dimensional representation** of sequences via their inferred parameters. Instead of focusing on high variance components in data space, this representation captures *high-level* sequence characteristics, such as oscillation frequency or walking style (Chapters 4, 5).
- **Controllable predictions**, allowing a user to consider different predictions for the same data by changing the specialisation type. This may be of particular interest to artists using sequence models, e.g. for human motion (Chapter 5) or physical modelling (Chapter 4). Alternatively, controlling the predictions can be used for 'what-if' planning or gauging uncertainty of a forecast.

See Section 3.3 for further details on the uses of MTDSs.

Papers Some sections of this thesis are available in the following papers:

- A. Bird, C. K. I. Williams, and C. Hawthorne. Multi-Task Time Series Analysis applied to Drug Response Modelling. In *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019 (Chapter 6).
- A. Bird and C. K. I. Williams. Customizing Sequence Generation with Multi-Task Dynamical Systems. *arXiv preprint arXiv:1607.06450*, 2020 (Chapters 3-5).

Code The code for a variety of MTDS models has been refactored into a Julia library, which can be found at https://github.com/ornithos/MTDS.jl. This includes a notebook indicating the intended use.

1.5 Thesis structure

The remainder of this thesis is structured as follows: Chapter 2 provides an overview of the background material required for this thesis. This includes latent variable models, time series modelling, and multi-task learning. The methodological contributions can primarily be found in Chapter 3, which includes an overview of the MTDS, its specialisation to some specific model types, learning and inference algorithms, model uses, and related work. Chapter 4 provides a thorough exploration of the MTDS on synthetic physical data, serving both as a validation of the methodological work of the prior chapter, and an exploration of the model properties. Chapter 5 applies the MTDS to motion capture data, which provides an insight into the application to RNNs and its performance on challenging, highly-structured data. Our experimental work concludes in Chapter 6 with an application to drug response data; we demonstrate that the MTDS can substantially improve existing clinical models without the need for black-box approaches. Finally, Chapter 7 summarises the conclusions in the preceding chapters and includes suggestions for future work.

CHAPTER 2

Background

This chapter provides a targeted overview of the areas on which our contributions build. In what follows, we assume some prior knowledge of probability distributions, graphical models, and neural networks. The first Section (2.1) introduces the key concept of a latent variable model and provides some examples. Section 2.2 provides an overview of some classical and modern time series methods, including autoregressive moving average (ARMA), linear dynamical system (LDS) and recurrent neural network (RNN) models. Section 2.3 introduces multi-task learning (MTL) and surveys some common approaches and related paradigms.

2.1 Latent variable models

In many modelling problems, some aspects of the data generating process are considered unknown. For instance, answers to a questionnaire may exclude questions of political affiliation, or patients may respond differently to treatment based on unavailable genetic information. In such cases, these *latent* properties can induce a rich *structure* of responses, such as a correlation between answers to politically relevant questions, or correlations between treatment response and its side effects. This section will introduce some basic forms of latent variable models which can be used as building blocks in more complex scenarios.

We restrict our attention to the simple iid case where the observation $\mathbf{y} \in \mathcal{Y}$ depends only on the latent variable $\mathbf{z} \in \mathcal{Z}$ (Figure 2-1a). A good example of this setup are images of handwritten digits (see Figure 2-1b). While the images are 784-dimensional, the variation can be mostly explained using just a few latent attributes, such as the intended digit and the handwriting style. Notably, most 784-D vectors generated uniformly at random will *not* correspond to valid handwritten digits, implying that such images are confined to a small – and hopefully structured – subset of $\mathcal{Y}$.

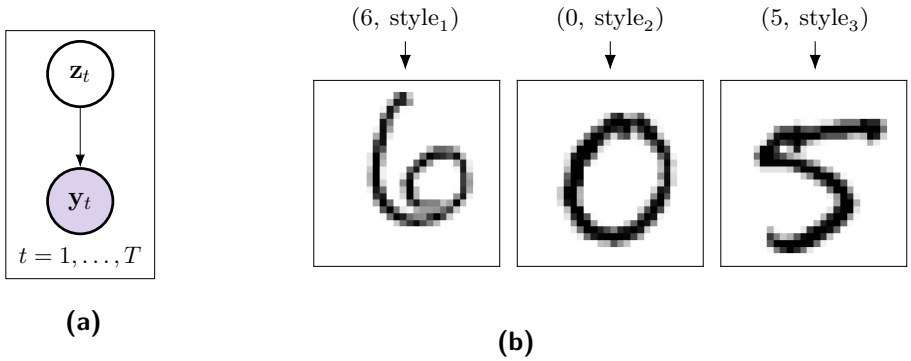


Figure 2-1: (a) Graphical model of a basic iid latent variable model, observed node shaded purple. (b) Example of iid latent variable model: 784 pixel images generated from two latent attributes (shown above).

A simple iid latent variable model is specified with the two distributions $p(\mathbf{z})$ and $p(\mathbf{y} \,|\, \mathbf{z})$, called the *generative model*, which results in the distribution over observations $p(\mathbf{y}) = \int p(\mathbf{y} \,|\, \mathbf{z})\, p(\mathbf{z})\, \mathrm{d}\mathbf{z}$. Given an observation $\mathbf{y}$ from such a model, it is often useful to determine the most likely values of $\mathbf{z}$ which gave rise to it. This may be of interest for a variety of unsupervised tasks (such as interpretative analysis, imputation and de-noising) as well as for downstream supervised tasks. The answer is achieved through probabilistic *inference* which proceeds via Bayes' rule:

$$p(\mathbf{z} \,|\, \mathbf{y}) \;=\; \frac{p(\mathbf{y} \,|\, \mathbf{z})\, p(\mathbf{z})}{p(\mathbf{y})}. \tag{2.1}$$

In many cases inference is difficult, and typically no closed form solution exists. However, there are certain choices of generative model which admit closed form solutions (see Section 2.1.2 for an example); otherwise various approximate answers may be obtained.

2.1.1 Learning latent variable models

If a generative model is not known *a priori*, one may choose a parametric form for the model, and learn the parameters from data. In the case of the handwritten digits, assuming $\mathcal{Z} = \mathbb{R}^{n_z}$, $n_z < 784$, a parametric latent variable model learns a low-dimensional *manifold* embedded in $\mathcal{Y}$ which lies close to the observed images. More generally we seek a high probability over areas in $\mathcal{Y}$ near the data, and a low probability over areas far from the data. We can achieve this by maximising the marginal likelihood (see e.g. Berger, 2013, §3). For a model indexed by a parameter $\boldsymbol{\theta}$, and datapoints $\mathbf{y}_1, \ldots, \mathbf{y}_T$, the marginal likelihood is defined by:

$$\prod_{t=1}^{T} p(\mathbf{y}_t; \boldsymbol{\theta}) \;=\; \prod_{t=1}^{T} \int p(\mathbf{y}_t \,|\, \mathbf{z}_t; \boldsymbol{\theta})\, p(\mathbf{z}_t; \boldsymbol{\theta})\, \mathrm{d}\mathbf{z}_t. \tag{2.2}$$

For complicated models, $p(\mathbf{y}_t; \boldsymbol{\theta})$ is generally not available in closed form. In this case,

while we may not directly optimise the marginal likelihood, we may consider a *variational lower bound* (see e.g. Jordan et al., 1999), sometimes called the *evidence lower bound*, or ELBO:

$$\log p(\mathbf{y}) \;=\; \log \int p(\mathbf{y}|\mathbf{z})p(\mathbf{z})\;\mathrm{d}\mathbf{z} \;=\; \log \int \frac{p(\mathbf{y}|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z})}\,q(\mathbf{z})\,\mathrm{d}\mathbf{z} \tag{2.3}$$

$$\geq \int \left[\log \frac{p(\mathbf{y},\mathbf{z})}{q(\mathbf{z})}\right] q(\mathbf{z})\,\mathrm{d}\mathbf{z} \;=:\; \mathcal{L}, \tag{2.4}$$

where we introduce an auxiliary probability distribution $q(\mathbf{z})$ and make use of Jensen's inequality in Equation (2.4). The bound is tight when $q(\mathbf{z}) = p(\mathbf{z}\,|\,\mathbf{y})$, since Equation (2.4) may be rearranged as:

$$\mathcal{L} \;=\; \log p(\mathbf{y}) - \mathrm{KL}\left(q(\mathbf{z}) \,\|\, p(\mathbf{z}\,|\,\mathbf{y})\right). \tag{2.5}$$

The integral in Equation (2.4) is still generally intractable (for exceptions see Wainwright et al., 2008), but low-variance estimates of its gradient may be estimated via Monte Carlo (see Kingma and Welling, 2014; Rezende et al., 2014).

2.1.2 Factor analysis

Our first example of a latent variable model is factor analysis (FA). FA assumes that the latent variable $\mathbf{z} \in \mathbb{R}^{n_z}$ is drawn from a unit Gaussian, and observed noisily after a linear transformation:

$$p(\mathbf{z}_t) \;=\; \mathcal{N}(0, I)\,, \tag{2.6}$$

$$p(\mathbf{y}_t\,|\,\mathbf{z}_t) \;=\; \mathcal{N}(C\,\mathbf{z}_t, R)\,, \tag{2.7}$$

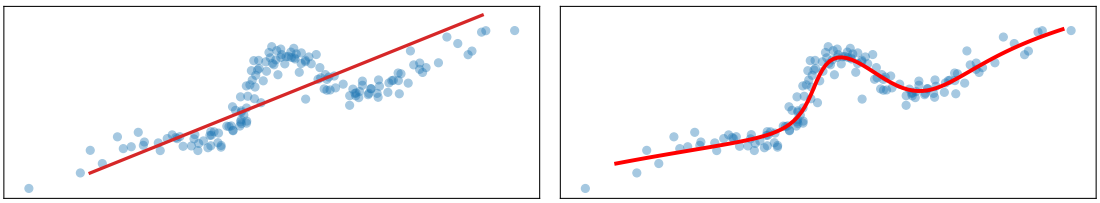
for observations $\mathbf{y}_t \in \mathbb{R}^{n_y}$, $t = 1, \ldots, T$, where $R \in \mathbb{R}^{n_y \times n_y}$ is a non-negative diagonal matrix and $C \in \mathbb{R}^{n_z \times n_y}$ is the factor loading matrix. (We assume that the data are centered, and hence have zero-mean.) For this model, the marginal likelihood can be calculated in closed form:

$$p(\mathbf{y}) \;=\; \int p(\mathbf{y}\,|\,\mathbf{z})\,p(\mathbf{z})\,\mathrm{d}\mathbf{z} \;=\; \mathcal{N}\left(0,\, CC^{\mathsf{T}} + R\right), \tag{2.8}$$

with the resulting distribution over the data also being Gaussian.

The geometric interpretation of factor analysis is that data are generated in a lower-dimensional subspace (determined by the matrix C), then perturbed by independent Gaussian noise in $\mathcal{Y}$. The goal is to locate this lower dimensional subspace which captures the important correlations in the data. An example of this is given in Figure 2-2(a) for 2-D data. Notice that most of the datapoints lie near the 1-D (red) manifold, and hence this manifold 'explains' most of the structure of the data.

Factor analysis is useful for dimensionality reduction, and we briefly remark upon its



(a) Factor analysis **(b)** Variational autoencoder

Figure 2-2: Manifolds (red line) learned by latent variable models on an example 2-D dataset (blue). Only the central 99% of manifold is shown (according to $p(\mathbf{z})$).

relation to principal components analysis (PCA). Unlike PCA, FA allows varying noise levels across the dimensions of the data. However, in setting $R = \sigma^2 I$ for some $\sigma \in \mathbb{R}$ (known as probabilistic PCA, see Tipping and Bishop, 1999) and taking the limit $\sigma \to 0$, we recover PCA. Hence PCA can yield a good initialisation for C, and can be a reasonable approximation to FA if the noise level is approximately uniform across dimensions.

Finding the most likely projection of each datapoint in the manifold learned by FA (and its uncertainty) can be achieved via posterior inference. For FA, this can be performed in closed form[1]:

$$p(\mathbf{z} \,|\, \mathbf{y}) \;=\; \mathcal{N}\left(C^{\mathsf{T}} \Sigma_{yy}^{-1} \mathbf{y},\;\; I - C^{\mathsf{T}} \Sigma_{yy}^{-1} C\right), \qquad \text{where } \Sigma_{yy} = (CC^{\mathsf{T}} + R). \tag{2.9}$$

A straight-forward approach to learning the parameters of the model is to use the Expectation Maximisation algorithm (Dempster et al., 1977) with the posterior distribution given in Equation (2.9).

2.1.3 Variational autoencoders

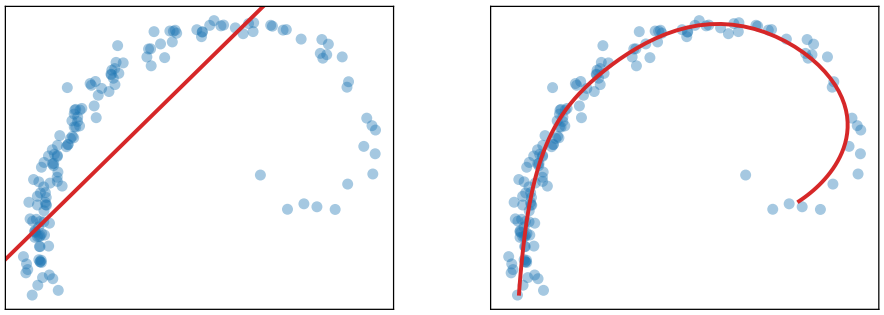
Factor analysis is frequently useful, but sometimes limited due to the assumption that the data lie close to a *linear* subspace. Figure 2-3 gives an example dataset where these assumptions are inappropriate. Variational autoencoders (VAEs) (Kingma and Welling, 2014; Rezende et al., 2014) are a generalisation of FA to *nonlinear manifolds*. The latent variable $\mathbf{z} \in \mathbb{R}^{n_z}$ is again drawn from a unit Gaussian, but its transformation is now via a neural network $\mathbf{g}_{\boldsymbol{\theta}}(\cdot)$. Where observations are real-valued, a common forward model is:

$$p(\mathbf{z}_t) \;=\; \mathcal{N}(0, I) \tag{2.10}$$

$$p(\mathbf{y}_t \,|\, \mathbf{z}_t) \;=\; \mathcal{N}\left(g_{\boldsymbol{\theta}}(\mathbf{z}_t), R\right) \tag{2.11}$$

where again, R is a non-negative diagonal matrix. Examples of fitted VAEs are given in Figures 2-2(b), 2-3(b) using a *multilayer perceptron* (MLP) for $\mathbf{g}_{\boldsymbol{\theta}}$ with one hidden layer.

[1] See the Gaussian identities in appendix A.1.



(a) Factor analysis **(b)** Variational autoencoder

Figure 2-3: Manifolds (red line) learned by latent variable models on a second example dataset. Only the central 99% of manifold is shown (according to $p(\mathbf{z})$).

In both cases the VAE has learned a one-dimensional manifold (shown in red) which captures the structure of the data well.

Unlike factor analysis, there is generally no closed form distribution for $p(\mathbf{y})$, or $p(\mathbf{z}\,|\,\mathbf{y})$, and we instead optimise the evidence lower bound (described above). To complete the specification of the VAE, we seek a $q(\mathbf{z})$ which achieves a tight bound of the ELBO. The VAE learns a parametric approximation $q_{\boldsymbol{\lambda}}(\mathbf{z}\,|\,\mathbf{y})$ to the posterior $p(\mathbf{z}\,|\,\mathbf{y})$ termed the 'variational posterior'. A common choice is a diagonal Gaussian for each observation $\mathbf{y}_t$:

$$q_{\boldsymbol{\lambda}}(\mathbf{z}\,|\,\mathbf{y}_t) = \mathcal{N}\left(\boldsymbol{\mu}_t, \operatorname{diag}(\mathbf{s}_t)\right), \tag{2.12}$$

where $\boldsymbol{\lambda} = \{\boldsymbol{\mu}_t, \mathbf{s}_t\}_{t=1}^{T}$. The variational parameters, $\boldsymbol{\lambda}$, and the model parameters, $\boldsymbol{\theta}$, are learned jointly via stochastic gradient ascent on Equation (2.4). Where T is reasonably large, it can be sensible to use *amortised* inference, where the parameters $\boldsymbol{\lambda}_t$ for each $\mathbf{y}_t$ are predicted by a neural network (as proposed in Kingma and Welling, 2014; Rezende et al., 2014).

Various other generalisations of factor analysis to nonlinear manifolds are available, including kernelised PCA, principal curves/surfaces (Hastie et al., 2009), Gaussian process latent variable models (GPLVM, Lawrence, 2005), generative topological mappings (GTM, Bishop et al., 1998). Many further methods exist for nonlinear dimensionality reduction which do not admit a clear latent variable interpretation. Such methods are elaborated upon elsewhere, and are not of primary importance in what follows.

2.2 Time series models

Many datasets possess an important temporal structure, from areas as diverse as natural language processing (e.g. Cho et al., 2014), electronic health data (e.g. Schulam and Saria, 2015), financial data (e.g Stock and Watson, 1999), and human locomotion (e.g. Fragkiadaki et al., 2015). Time series models move beyond iid assumptions and are specialised to exploit this temporal structure. We discuss three well-known classes of these models: ARMA-type models (Section 2.2.2), linear dynamical systems (Section 2.2.3) and recurrent neural networks (Section 2.2.4).

We define a *time series* as a collection of data $\{\mathbf{y}(t)\colon t \in \mathcal{T}\}$, where each observation is recorded at a specific time t from an *index set* $\mathcal{T}$. Mathematically, a distribution over a time series is specified by a *stochastic process*, a collection of random variables indexed by $\mathcal{T}$. If the distribution is time invariant it is called a *stationary* stochastic process. In this thesis we consider *discrete-time* time series where the index set $\mathcal{T}$ is the set of integers $1, 2, \ldots, T$ (possibly with missing values), and hence write $\mathbf{y}_t := \mathbf{y}(t)$, and $\mathbf{y}_{1:\tau} := \{\mathbf{y}_1, \ldots, \mathbf{y}_\tau\}$. We will also assume in what follows that the *observation space is a real vector space*, i.e. $\mathbf{y}_t \in \mathbb{R}^{n_y}$.

2.2.1 Introduction

There are two common approaches to time series modelling. Firstly, the *autoregressive* approach, where observations $\mathbf{y}_t$ are a (stochastic) function of a *fixed-size subset* of the previous observations $\mathbf{y}_{1:t-1}$. Secondly, the *state space* approach, where observations are a function of a *fixed-size vector* variable (called the 'state'), which is updated at each timestep, and attempts to summarise the relevant information from the entire $\mathbf{y}_{1:t-1}$.[2] Both are attempts to model a causal process with a variable-sized history using a finite set of parameters. Sections 2.2.2 – 2.2.4 explore many important time series models in use today, all of which can be interpreted as *dynamical systems*, i.e. state space models (SSMs).

Dynamical systems take the following form, where the hidden state $\mathbf{x}_t$ follows a (possibly stochastic) dynamical model, and optionally takes control inputs $\mathbf{u}_t$:

$$\mathbf{x}_t \sim p(\mathbf{x} \mid \mathbf{x}_{t-1}, \mathbf{u}_t ; \boldsymbol{\theta}), \tag{2.13}$$

$$\mathbf{y}_t \sim p(\mathbf{y} \mid \mathbf{x}_t, \mathbf{u}_t ; \boldsymbol{\theta}), \tag{2.14}$$

for $t = 1, \ldots, T$, with parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$. Dynamical systems have a number of desirable properties in general, such as time-invariant feature extraction, linear complexity in T, and in principle, an unbounded length of temporal dependence.

[2] This definition follows a predictive rather than a generative perspective, which has the advantage of applying even under model misspecification.

In practice, time series models can be used in a number of different ways. A common task is *prediction* (or *forecasting*), for instance via the quantity $\mathbb{E}[\mathbf{y}_{t+1:t+k} \,|\, \mathbf{y}_{1:t}]$ for some window length k. This is notoriously difficult for large k due to incomplete information and changepoints in the underlying data generating process. However, where the data are of physical and/or highly structured phenomena, long term prediction may be possible. Alternatively, models may be used for *smoothing* or *imputation* where data are noisy or missing, a form of 'post'-diction. A generative time series model may also be used for *anomaly detection* by monitoring the likelihood of a sequence over time (for example Quinn et al., 2009, in the context of healthcare).

2.2.2 Linear ARMA-type models

In 1970, Box et al. popularised a cluster of related models, including the well-known autoregressive moving average (ARMA) model, which are still in wide use today. We will briefly describe a number of these below, including AR, MA, VAR, VARX and ARMA models. For further information see e.g. Box et al. (2008); Hamilton (1994); Shumway and Stoffer (2017).

2.2.2.1 Autoregressive models

Linear autoregressive (AR) models are among the simplest non-trivial time series models. An order-p AR model, denoted AR(p), assumes each observation is a linear function of the p previous observations[3], perturbed by white noise[4] (with variance σ^2):

$$y_t = \sum_{j=1}^{p} a_j y_{t-j} + \epsilon_t, \qquad \epsilon_t \sim \mathcal{WN}(0, \sigma^2). \tag{2.15}$$

Figure 2-4 shows the graphical model for an AR(1) process. For ease of exposition, our focus will be on Gaussian white noise, in which case the model defines a zero-mean Gaussian process (Rasmussen and Williams, 2006).

Not every set of parameters $\{a_j\}_{j=1}^{p}$ will result in a sensible model. To see this, the definition in Equation (2.15) may be recursively unrolled to present the AR(p) purely in terms of the noise process. For the AR(1) model (we assume for convenience that $y_0 = 0$), this gives:

$$y_t = \sum_{j=1}^{t} a_1^{t-j} \epsilon_j \tag{2.16}$$

[3] Extensions to non-consecutive / dynamically selected previous observations are possible, as e.g. in the 'method of analogues'. See e.g. Moore and Little (2014); Viboud et al. (2003) and refs. therein.

[4] Extensions beyond white noise assumptions include Gaussian mixtures (e.g. Roberts and Penny, 2002), (generalized) autoregressive conditionally heteroscedastic models ((G)ARCH models, e.g. Brockwell et al., 2002, §7) and bilinear models (e.g. Douc et al., 2014, §3.4).

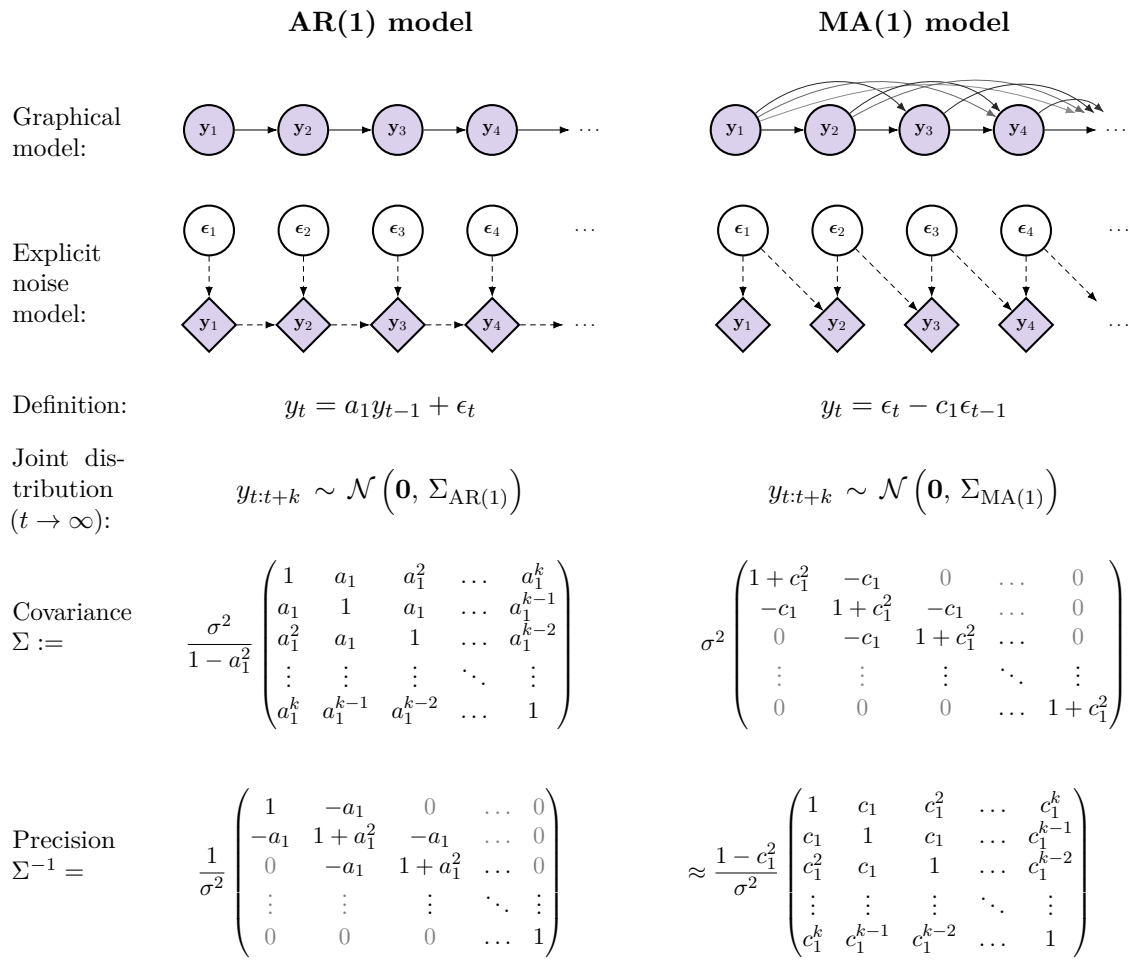


Figure 2-4: A comparison of order-1 autoregressive and moving average models. Here we assume $\epsilon_t \sim \mathcal{N}\left(0, \sigma^2\right)$. The definitions are given under the graphical models (note the negative coefficient in the MA model), and both models result in a Gaussian joint distribution over sequences. For large t, these approach the stationary distribution, where the covariance over consecutive observations is given. The inverse covariance (precision) is also shown to highlight the relationship between the models. Note that the precision for the MA(1) model is not exact due to the first and last entries on the diagonal of the covariance. For an exact inverse, see Hu and O'Connell (1996).

for all t. We see that each y_t is a linear combination of *all* previous $\{\epsilon_j : j \leq t\}$. Since the coefficients of each noise term grow geometrically with t, we must employ the constraint that $|a_1| \leq 1$ in order to avoid the process 'blowing up'. If the inequality holds strictly, the sum in Equation (2.16) converges for $t \to \infty$ to some stationary distribution. For such coefficients, the impact of any impulses or 'shocks' observed in the data will decay geometrically over time. See e.g. Box et al. (2008) §3.2 for the conditions on general AR(p) processes.

2.2.2.2 Moving average models

Moving average (MA) models employ a different but equally simple relationship. The MA(q) model is defined as:

$$y_t = \epsilon_t + \sum_{j=1}^{q} c_j \epsilon_{t-j}, \qquad \epsilon_t \sim \mathcal{WN}(0, \sigma^2) \tag{2.17}$$

for all t, see Figure 2-4 for a graphical model. Here the observations are a linear function of the q most recent noise variates. Unlike an AR model, the impact of impulses are bounded in time, for instance in the MA(1) process, $\mathrm{Cov}(y_t, y_{t-2}) = 0$. For this reason, there need be no restriction on the parameters $\{c_j\}_{j=1}^{q}$.

2.2.2.3 ARMA models

Figure 2-4 provides a side-by-side comparison of order-1 AR and MA models. When t is large enough, the distribution implied by both models is a stationary zero-mean Gaussian process (assuming Gaussian white noise). As shown in Figure 2-4, the MA(1) model naturally parameterises the covariance, and the AR(1) model similarly parameterises the inverse covariance (precision). This correspondence is formalised in the well-known duality between AR and MA processes: a (weakly) stationary AR(p) process may be expressed as an MA(∞) process, and an MA(q) process may be expressed as an AR(∞) process[5] (see e.g. Box et al., 2008, §3.3.5).

For a parsimonious model, one may wish to combine both representations, which is known as an ARMA model. The ARMA(p, q) model can be written for all t:

$$y_t - \sum_{j=1}^{p} a_j y_{t-j} = \epsilon_t - \sum_{j=1}^{q} c_j \epsilon_{t-j}, \qquad \epsilon_t \sim \mathcal{WN}(0, \sigma^2). \tag{2.18}$$

Generalisations beyond white noise processes for $\{\epsilon_t\}$ are possible, as in footnote 4 (page 15), but we will not consider such models further in this thesis.

[5] The MA process must be invertible, see Box et al. (2008), §3.1.3.

2.2.2.4 Vector AR models

The AR model can be readily extended to vector-valued observations $\mathbf{y}_t \in \mathbb{R}^{n_y}$. A vector autoregressive (VAR) model of order 1 is defined as:

$$\mathbf{y}_t = A\mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t, \tag{2.19}$$

$t = 1, \ldots, T$ for some matrix $A \in \mathbb{R}^{n_y \times n_y}$ and noise process $\{\boldsymbol{\epsilon}_t\}$. It is convenient to assume that $\boldsymbol{\epsilon}_t \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \Omega)$. The parameter A must be chosen carefully (as for the AR model) to avoid a geometric explosion in $\|\mathbf{y}_t\|_2$ over time. Rewriting the VAR(1) model in terms of the noise process gives $\mathbf{y}_t = \sum_{j=1}^{t} A^{t-j}\boldsymbol{\epsilon}_j$, and the expected 2-norm of the observation $\mathbf{y}_t$ can be bounded as follows:

$$\begin{aligned}
\mathbb{E}\,\|\mathbf{y}_t\|_2 &\leq \mathbb{E}\sum_{j=1}^{t} \|A\|_2^{t-j} \|\boldsymbol{\epsilon}_j\|_2 && \text{(triangle inequality)} && (2.20)\\
&= \sum_{j=1}^{t} \|A\|_2^{t-j}\, \mathbb{E}\left(\boldsymbol{\epsilon}_j^{\mathsf{T}}\boldsymbol{\epsilon}_j\right)^{1/2} && && (2.21)\\
&\leq \mathrm{Tr}(\Omega)^{1/2} \sum_{j=1}^{t} \|A\|_2^{t-j}. && \text{(Jensen's inequality)} && (2.22)
\end{aligned}$$

Here, $\|\cdot\|_2$ is the induced 2-norm (i.e. the largest singular value), and hence Equation (2.20) follows by definition and use of the triangle inequality. Geometric growth of the $\mathbf{y}_t$ can thus be avoided by constraining the spectral radius of A to be bounded by unity. Such models will be denoted as 'stable'.[6]

An additional 'control input' or 'exogenous' variable $\mathbf{u}_t \in \mathbb{R}^{n_u}$ may often be available alongside each $\mathbf{y}_t$. A VARX model (a VAR model with eXogenous inputs) incorporates these as follows:

$$\mathbf{y}_t = A\mathbf{y}_{t-1} + B\mathbf{u}_t + \epsilon_t, \qquad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \Omega), \tag{2.23}$$

for some matrix $B \in \mathbb{R}^{n_u \times n_y}$. An order-$p$ VARX model will generally include p autoregressive matrices $A_1, \ldots, A_p$ as well as p regression matrices B_j, $j = 1, \ldots, p$ for lagged inputs $\mathbf{u}_{t-1}, \ldots, \mathbf{u}_{t-p}$.

2.2.2.5 Connection between VAR(1) and AR(p) models

Any AR(p) model may be represented by a VAR(1) model with $\mathbf{y} \in \mathbb{R}^p$. The AR($p$) model may be written:

$$y_{t+1} = \left\langle (a_1, a_2, \ldots, a_p),\ (y_t, y_{t-1}, \ldots, y_{t-p+1}) \right\rangle + \epsilon_t. \tag{2.24}$$

[6] These models are Lyapunov stable *in expectation* (cf. e.g. Aström and Murray, 2010, §4.3).

Define an augmented observation vector $\mathbf{y}_t$ as follows to yield a VAR(1) model:

$$\underbrace{\begin{pmatrix} y_{t+1} \\ y_t \\ y_{t-1} \\ \vdots \\ y_{t-p+2} \end{pmatrix}}_{\mathbf{y}_{t+1}} = \underbrace{\begin{pmatrix} a_1 & a_2 & \dots & a_{p-1} & a_p \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ddots & 1 & 0 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} y_t \\ y_{t-1} \\ y_{t-2} \\ \vdots \\ y_{t-p+1} \end{pmatrix}}_{\mathbf{y}_t} + \underbrace{\begin{pmatrix} \epsilon_t \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}}_{\boldsymbol{\epsilon}_t}, \tag{2.25}$$

with a degenerate Ω. Thus a VAR(1) model in $\mathbb{R}^p$ is a more expressive model than an AR(p) model. This idea can also be applied to writing a VAR(p) model in $\mathbb{R}^{n_y}$ as a VAR(1) model in $\mathbb{R}^{n_y \times p}$.

2.2.2.6 What does a sample from a VAR process look like?

Let us consider the qualitative features of data generated from a VAR model. Consider first the noiseless case with $\Omega \to 0$ which starts from a known $\mathbf{y}_0 \neq \mathbf{0}$. Let $\{\lambda_i\}$ and $\{\mathbf{v}_i\}$ be the eigenvalues and eigenvectors of A, and $\mathbf{y}_0 = \sum_{i=1}^{n_y} b_i \mathbf{v}_i$. Then:

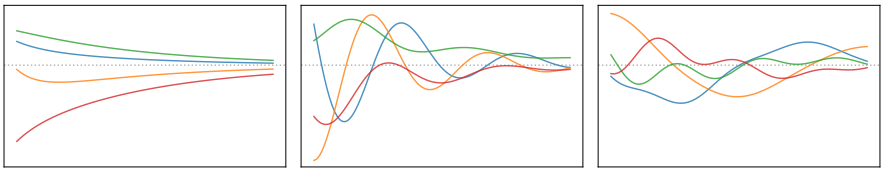
$$\mathbf{y}_t = A^t \mathbf{y}_0 = \sum_{i=1}^{n_y} b_i A^t \mathbf{v}_i = \sum_{i=1}^{n_y} b_i \lambda_i^t \mathbf{v}_i. \tag{2.26}$$

Assuming a stable VAR process, the spectral radius of A is bounded by 1, and so $|\lambda_i| \leq 1$ for all i. We can categorise the action of A on $\mathbf{y}_0$ in the subspaces spanned by each $\mathbf{v}_i$ according to their eigenvalue λ_i:

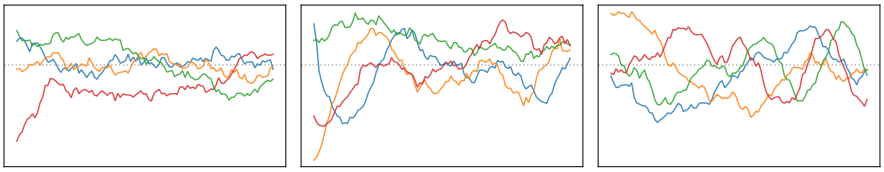
- $\lambda_i \in \{-1, +1\}$. b_i is unchanged over time ($\lambda_i = 1$) or oscillates with period $t = 2$ about the origin ($\lambda_i = -1$).
- $\lambda_i \in (-1, 1)$. $|b_i|$ shrinks geometrically over time, and if $\lambda < 0$, oscillates about the origin with period $t = 2$.
- $\lambda_i \in \mathbb{C} \setminus \mathbb{R}$. In this case there will exist a corresponding complex conjugate, say λ_{i+1}. The components $\{b_i, b_{i+1}\}$ will decay geometrically and rotate with constant angular velocity $\arctan(\mathrm{Im}(\lambda_i)/\mathrm{Re}(\lambda_i))$ about the origin of the subspace spanned by $\mathbf{v}_i$, $\mathbf{v}_{i+1}$.

Hence for an initial shock, $\mathbf{y}_t$ will follow a linear combination of various geometric decay and damped harmonic functions. See Figure 2-5a for examples. Since VAR(p) models may be rewritten as (larger) VAR(1) models, higher order models differ only in the number of components; the qualitative action of A remains the same.

As the noise covariance Ω increases, the sequence will appear more complex, but these qualitative phenomena are still present in expectation, and hence in the autocorrelation function. See Figure 2-5b for samples from stochastic VAR(1) models. A VARX model



(a) Deterministic VAR(1)



(b) Stochastic VAR(1)

Figure 2-5: Example samples from different VAR(1) processes with $n_y = 4$, each dimension plotted in a different colour. Each column in (a) and (b) has the same parameters and initial conditions.

introduces what might be called 'supervised disturbances', $\mathbf{w}_t(\mathbf{u}_t) := B\mathbf{u}_t + \boldsymbol{\epsilon}_t$, which follow the same dynamics as the VAR process. Hence the functional relationship to the inputs can also be described by geometric decay and damped harmonic oscillation.

2.2.2.7 Connection between VAR models and ODEs

Deterministic VARX models are the discrete time analog of forced linear ordinary differential equations (ODEs). Consider the following order-1 multivariate ODE:

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = \tilde{A}\mathbf{y}(t) + \tilde{B}\mathbf{u}(t) \tag{2.27}$$

with parameters $\tilde{A} \in \mathbb{R}^{n_y \times n_y}$ and $\tilde{B} \in \mathbb{R}^{n_x \times n_u}$. Recall that order-$p$ differential equations may also be written in this form, see e.g. Gelb (1974, §3.1). Suppose that $\mathbf{u}(t)$ is a piecewise constant function with changepoints in the set $\mathcal{T} = \{\Delta t,\, t \in \mathbb{Z}\}$ and values $\{\ldots, \mathbf{u}_{-1}, \mathbf{u}_0, \mathbf{u}_1, \ldots\}$. Equation (2.27) may be discretised to $\mathcal{T}$ via:

$$\mathbf{y}_{t+1} = A\,\mathbf{y}_t + B\,\mathbf{u}_t, \tag{2.28}$$

$$A := e^{\tilde{A}\Delta}, \tag{2.29}$$

$$B := \left(\int_0^\Delta e^{\tilde{A}s}\,\mathrm{d}s\right)\tilde{B} \;=\; \tilde{A}^{-1}\left(e^{\tilde{A}\Delta} - I\right)\tilde{B}, \tag{2.30}$$

where $e^{\tilde{A}\Delta}$ is a matrix exponential (see Aström and Murray, 2010, §5.3 for more details). Hence a (forced) linear ODE can be approximated in discrete time by a VARX(1) model. As we have seen, the expected samples of VAR(1) models mirror those of linear ODE solutions (see e.g. Strogatz, 2018, §5). An equivalent identification can be made in the stochastic case (Särkkä and Solin, 2019, §6). For more details on stochastic differential

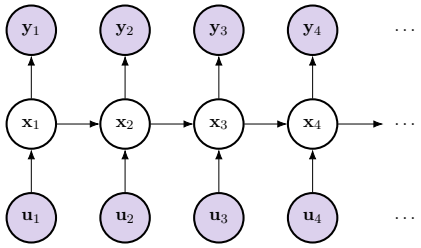


Figure 2-6: Graphical model of a linear dynamical system: observed variables shaded purple.

equations and their relationship to Gaussian time series models, see Särkkä and Solin (2019) and Rasmussen and Williams (2006, appendix B).

2.2.3 Linear dynamical systems

We now move to a different philosophy of modelling, which involves a *state space* approach. A 'state', $\mathbf{x}_t \in \mathbb{R}^{n_x}$, is a time-varying unobserved quantity which, from a predictive perspective, functions as an approximately sufficient statistic of the past. From a generative perspective, the latent state defines the true dynamic process, which is observed only indirectly via $\mathbf{y}_t$. In this section we introduce the linear dynamical system (LDS), leaving examples of nonlinear systems to Section 2.2.4.

The LDS is well-known for having early application during the Apollo missions. For navigation, the crucial quantities of position and velocity of the spacecraft (the 'state') could not be observed, but orientation, acceleration, and relative position of the stars were available. These quantities could be resolved into an estimate of position and velocity via an LDS, which became a critical part of the navigation system.[7] LDS models are now used in a wide variety of domains: they are useful as general purpose time series models, which provide interpretable states, and which allow embedding of domain knowledge. In this section, we introduce LDSs, their relationship to ARMA models, and inference of their latent state.

2.2.3.1 Definition

The latent states $\mathbf{x}_t \in \mathbb{R}^{n_x}$ in an LDS evolve according to VARX(1) dynamics, observed indirectly by $\mathbf{y}_t$ via a noisy linear transformation. As in the previous section, we assume Gaussian noise, yielding:

$$\mathbf{x}_t = A\,\mathbf{x}_{t-1} + B\,\mathbf{u}_t + \mathbf{b} + \mathbf{v}_t, \qquad \mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, R), \tag{2.31}$$

$$\mathbf{y}_t = C\,\mathbf{x}_t + D\,\mathbf{u}_t + \mathbf{d} + \mathbf{w}_t, \qquad \mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, S). \tag{2.32}$$

[7] While this problem is not linear, early work used linearisation, for which LDS assumptions then applied. For further details of this application see e.g. McGee and Schmidt (1985); Hoag (1969); Murtagh (1967).

for all t. Where control inputs $\mathbf{u}_t \in \mathbb{R}^{n_u}$ are available, they can enter the LDS both within the latent VARX(1) process and as a regression within the emission equation. See Figure 2-6 for a graphical model. One may also interpret the LDS as a dynamic factor analysis model (cf. Section 2.1.2) where the latent variable is now correlated in time (e.g. Roweis and Ghahramani, 1999). This model is again jointly Gaussian in $\{\mathbf{y}_t\}$ due to the linear Gaussian relationships.

2.2.3.2 Stability and identifiability

Since the system equation (Equation 2.31) is a VARX(1) process, we require the same stability requirement as in Section 2.2.2.4, namely that the spectral radius of A is bounded by unity. In some cases, learning the model from data will rule out an unstable matrix, but it is often useful to enforce the constraint, see e.g. Siddiqi et al. (2008).

The LDS is also non-identifiable up to an arbitrary invertible matrix. The latent space used to define $\mathbf{x}_t$ may be squashed and/or rotated arbitrarily, since these (linear) operations may be undone by the emission matrix, C. Specifically, for any invertible matrix G, the distribution over the observations $\mathbf{y}_{1:T}$ is invariant to the following parameter transformations:

$$A \leftarrow G^{-1}AG, \quad B \leftarrow G^{-1}B, \quad \mathbf{b} \leftarrow G^{-1}\mathbf{b}, \quad R \leftarrow G^{-1}RG^{-\mathsf{T}}, \tag{2.33}$$

$$C \leftarrow CG, \quad D \leftarrow D, \quad \mathbf{d} \leftarrow \mathbf{d}, \quad S \leftarrow S. \tag{2.34}$$

We provide a proof in appendix A.2.1 for completeness. This unfortunately complicates the issue of interpreting the parameters and states of learned LDS models, and impedes comparisons between models.

2.2.3.3 Equivalence of LDS and ARMA models

It can be shown that for any ARMA model there is an equivalent LDS and vice versa. This might be expected since the LDS defines a Gaussian distribution over the $\{\mathbf{y}_t\}$ and ARMA models are dense in Gaussian processes (see e.g. Brockwell et al., 1991, §13.4). The LDS approach is hence not more expressive in principle than ARMA models, but is often more parsimonious and interpretable in practice. We provide some brief intuition of the formal equivalence below; for ease of exposition, we assume that the LDS has no control inputs, and $\mathbf{x}_0 = 0$.

To show that an ARMA(p, q) model can be written as an LDS, we begin with the representation of an AR(p) model represented as a VAR(1) model (Section 2.2.2.5). Following Hamilton (1994, §13.1) we extend this construction with the emission matrix:

$$y_t = \underbrace{\begin{bmatrix} 1 & c_1 & c_2 & \dots & c_r \end{bmatrix}}_{C_{\text{ARMA}}} \mathbf{x}_t, \tag{2.35}$$

with $r = \min(p, q+1)$. This is equivalent to an ARMA(p, q) model where the MA model has parameters $\{c_1, \ldots, c_r\}$, some of which may be zero (proof in appendix A.2.2). This is easily extended to vector-valued ARMA models.

On the other hand, any LDS can be written in the form of a MA(∞) model. From (2.31), (2.32), we can write the observations purely in terms of the noise processes:

$$\mathbf{y}_t = \mathbf{w}_t + \sum_{i=1}^{t} CA^{t-i}\mathbf{v}_i \tag{2.36}$$

$$= \begin{bmatrix} 0_{n_y \times n_x} & I_{n_y} \end{bmatrix} \boldsymbol{\epsilon}_t + \sum_{i=1}^{t} CA^{t-i} \begin{bmatrix} I_{n_x} & 0_{n_x \times n_y} \end{bmatrix} \boldsymbol{\epsilon}_i \tag{2.37}$$

where $\boldsymbol{\epsilon}_t := \begin{bmatrix} \mathbf{v}_t^\mathsf{T} & \mathbf{w}_t^\mathsf{T} \end{bmatrix}^\mathsf{T}$, which is the form of an MA(∞) model. When $\|A\|_2 \ll 1$, this can be well approximated by a finite order MA model. See Gilbert (1993), Hamilton (1994, §13.5), for a formal proof and further discussion.

2.2.3.4 Inference

To perform inference over $\mathbf{x}_t$, we can use the well-known Kalman filter (Kalman, 1960). For predictive tasks, the crucial quantity is the *filtering distribution* $p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t})$ which summarises our belief state over $\mathbf{x}_t$ for the observations up to time t. Since the joint distribution over $(\mathbf{x}_{1:T}, \mathbf{y}_{1:T})$ for an LDS is Gaussian, this distribution is also Gaussian and available via recursive computation. We recapitulate the derivation in appendix A.2.3, but a summary of the algorithm is given below, following Särkkä (2013). For convenience, we assume that no inputs are available. More details can be found e.g. in Barber (2012, §24.4), and a general presentation including extensions can be found in Särkkä (2013).

Kalman Filter The filtering distribution may be computed using the well-known recursion introduced by Kalman (1960). The filter is usually initialised from the prior distribution over $\mathbf{x}_1$, and the filtering distribution for each time t,

$$p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t}) = \mathcal{N}(\mathbf{m}_t, P_t), \tag{2.38}$$

is calculated recursively via the below calculations:

$$\begin{aligned}
\mathbf{m}_t^- &= A\mathbf{m}_{t-1}, \\
P_t^- &= AP_{t-1}A^\mathsf{T} + R. \\
K_t &= P_t^- C^\mathsf{T}(CP_t^- C^\mathsf{T} + S)^{-1}. \\
\mathbf{m}_t &= \mathbf{m}_t^- + K_t(\mathbf{y}_t - C\mathbf{m}_t^-), \\
P_t &= P_t^- - K_t C P_t^-.
\end{aligned}$$

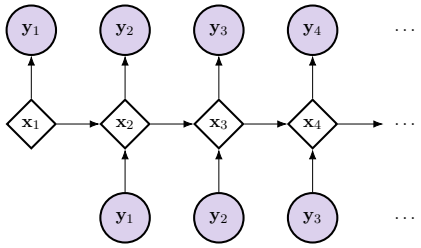


Figure 2-7: One-step-ahead prediction of an LDS shown as a deterministic graphical model. Control inputs are suppressed for clarity.

These updates may be implemented in a variety of other forms for computational and numerical reasons (see e.g. Durbin and Koopman, 2012, §4.3, §6.3, Barber, 2012, §24.4) but the underlying idea remains the same.

The Kalman Filter can also be used to calculate the likelihood of an LDS:

$$p(\mathbf{y}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{y}_t \,|\, \mathbf{y}_{1:t-1}) \tag{2.39}$$

where

$$p(\mathbf{y}_t \,|\, \mathbf{y}_{1:t-1}) = \mathcal{N}\left(C\mathbf{m}_t^-, CP_t^-C^\mathsf{T} + S\right). \tag{2.40}$$

See appendix A.2.3, especially Equation (A.26) for further details.

Rauch-Tung-Striebel (RTS) Smoother One may also be interested in using future information to obtain greater precision for the latent variable:

$$p(\mathbf{x}_t \,|\, \mathbf{y}_{1:T}) = \mathcal{N}\left(\mathbf{m}_t^s, P_t^s\right). \tag{2.41}$$

This is known as the *smoothed* distribution, and can be obtained via the backward RTS recusion (note that one must first perform the forward recursion above):

$$\begin{aligned} G_t &= P_t A^\mathsf{T} \left[P_{t+1}^-\right]^{-1}, \\ \mathbf{m}_t^s &= m_t + G_t\left(\mathbf{m}_{t+1}^s - \mathbf{m}_{t+1}^-\right), \\ P_t^s &= P_t + G_t\left(P_{t+1}^s - P_{t+1}^-\right)G_t^\mathsf{T}. \end{aligned}$$

The recursion is initialised from the distribution $p(\mathbf{x}_T \,|\, \mathbf{y}_{1:T}) = \mathcal{N}\left(\mathbf{m}_T^s, P_T^s\right)$ obtained from the Kalman Filter, and the quantities $\mathbf{m}_{t+1}^-$, P_{t+1}^- are from the one-step predictive distributions also from the Kalman Filter (above). For a proof, see Särkkä (2013, §8.2).

2.2.3.5 Steady state inference

As may be observed from the Kalman equations, the posterior covariance matrices of an LDS, $\{P_t\}$, are independent of the data, and are often known to converge quickly to a fixed point:

$$P = (APA^\mathsf{T} + R) - (APA^\mathsf{T} + R)\, C^\mathsf{T} \left(C(APA^\mathsf{T} + R)C^\mathsf{T} + S\right)^{-1} C\, (APA^\mathsf{T} + R), \quad (2.42)$$

which we call the *steady state* of the Kalman filter. This is derived by setting $P_t = P_{t-1}$ (see the relevant quantities in Equation 2.38); further discussion can be found in Harvey (1990, §3.3.3). The Kalman gain (K_t) and 'smoother gain' (G_t) depend on time only through P_t, so these quantities also converge to a fixed point if P_t does. Let us refer to these steady state quantities as K and G respectively. Pre-calculating P, K and G by solving Equation (2.42) either by spectral methods (e.g. Laub, 1979) or simple iteration can allow for dramatic improvement in efficiency of both filtering and smoothing, since it reduces all necessary calculations in eqs. (2.38, 2.41) to linear operations.

2.2.3.6 Steady state prediction via a deterministic model

Using the above quantities, the one-step likelihood (Equation 2.40) at steady state may be written in the following simplified form:

$$\mathbf{m}_t = A\mathbf{m}_{t-1} + K(\mathbf{y}_t - CA\mathbf{m}_{t-1}) \quad (2.43)$$

$$\mathbf{y}_{t+1} \,|\, \mathbf{y}_{1:t} \sim \mathcal{N}\left(CA\mathbf{m}_t,\, C(APA^\mathsf{T} + R)C^\mathsf{T} + S\right). \quad (2.44)$$

This follows directly from setting $K_t \leftarrow K$ and $P_t \leftarrow P$ in eqs. (2.38, 2.40). It is notable that the algorithm for *calculating the likelihood* of an LDS itself forms a *deterministic* LDS (Figure 2-7). Writing eqs. (2.43 - 2.44) as $\mathbf{m}_t = \tilde{A}\mathbf{m}_{t-1} + \tilde{B}\mathbf{y}_t$ and $\mathbf{y}_{t+1}|\mathbf{y}_t \sim \mathcal{N}\left(\tilde{C}\mathbf{m}_t,\, \tilde{S}\right)$, and learning the matrices $\tilde{A}, \tilde{B}, \tilde{C}, \tilde{S}$ directly from the data, we see that for every *stochastic* LDS model at steady state, there exists an equivalent *deterministic* LDS with the same likelihood over $\mathbf{y}_{1:T}$. Hence one may simply learn this deterministic version instead of the more challenging stochastic variant. This may be seen as an analogy of the 'teacher forcing' criterion used to train recurrent neural networks (see e.g. Goodfellow et al., 2016, §10.2.1).

This idea has some deficiencies: it is only strictly applicable at steady state, and ignores any increased uncertainty at the beginning of the sequence. It also fails to incorporate the state uncertainty for k-step ahead predictions $\mathbf{y}_{t+k} \,|\, \mathbf{y}_{1:t}$, $k > 1$. Tasks such as smoothing, imputation, and anomaly detection are also not immediately possible, and it is arguably less interpretable. Nevertheless, such a model is mathematically and computationally much simpler both for learning and inference if it is to be used solely for prediction. It also permits a variety of extensions, such as more expressive dynamics or emissions, or hierarchical models of LDSs, which efficient inference would normally preclude.

Algorithm 1: Expectation Maximisation of a LDS

Result: Optimised parameter $\hat{\boldsymbol{\theta}}$
Inputs: $\boldsymbol{\theta}^{(0)}$, n_{iters} ;
for $n = 1 : n_{iters}$ **do**
  $p(\mathbf{x}_{1:T} \,|\, \mathbf{y}_{1:T}, \boldsymbol{\theta}^{(n-1)}) \leftarrow \text{RTSSmoother}(\boldsymbol{\theta}^{(n-1)}, \mathbf{y}_{1:T})$;
  $\boldsymbol{\theta}^{(n)} \leftarrow \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{p(\mathbf{x}_{1:T} \,|\, \mathbf{y}_{1:T}, \boldsymbol{\theta}^{(n-1)})} \sum_{t=1}^{T} \log p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}; \boldsymbol{\theta}) + \log p(\mathbf{y}_t \,|\, \mathbf{x}_t; \boldsymbol{\theta})$;
end

2.2.3.7 Learning

Learning a LDS usually proceeds via maximum likelihood estimation (MLE), defined as:

$$\hat{\boldsymbol{\theta}} := \arg\max_{\boldsymbol{\theta}} \; p(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T; \boldsymbol{\theta}) \;=\; \arg\max_{\boldsymbol{\theta}} \; p(\mathbf{y}_1; \boldsymbol{\theta}) \prod_{t=1}^{T} p(\mathbf{y}_t \,|\, \mathbf{y}_{1:t-1}; \boldsymbol{\theta}), \tag{2.45}$$

with $\boldsymbol{\theta} := \{A, B, \mathbf{b}, R, C, D, \mathbf{d}, S\}$. Each of the one-step conditional distributions on the RHS can be obtained from the Kalman filter using Equation (A.26), or equivalently Equation (2.44). However, taking the derivatives of the Kalman filter updates is laborious and error-prone. It is more common to use expectation maximisation (EM, Dempster et al., 1977), an alternative optimisation strategy which makes use of the lower bound in Equation (2.4). The EM algorithm for LDS models proceeds iteratively from an initial $\boldsymbol{\theta}^{(0)}$ following the high-level steps in algorithm 1. The optimal parameters of the inner optimisation loop can be found in closed form for an LDS, for which details may be found e.g. in Särkkä (2013) §12.3.

Expectation maximisation is often observed to converge quickly to the vicinity of an optimum, but slowly thereafter. Second order techniques (e.g. Shumway and Stoffer, 2017, §6.3) may be preferred if convergence to optima is important, but for machine learning applications this is rarely required (see e.g. Prechelt, 1998).[8] A more serious problem with EM is convergence to suboptimal local optima, similarly to use of gradient methods. In order to avoid such local optima, subspace identification methods may be used, which approximate a global solution via spectral methods (see e.g. Van Overschee and De Moor, 2012).

2.2.4 Recurrent neural networks

Recurrent neural networks (RNNs) are a class of nonlinear state space models with deterministic state. RNNs can respond to data with far greater flexibility than LDS or ARMA models, but cannot calibrate uncertainty well out-of-the-box. In this section we will introduce basic RNNs, discuss optimisation issues, and present two common architectures: long short term memory (LSTM) and gated recurrent units (GRUs).

[8] In a nutshell, the optimisation objective of minimising training error is different to the true objective, that of minimising generalisation error.

2.2.4.1 A basic RNN

We begin with one of the simplest architectures of RNNs (Elman, 1990), which can be seen as a deterministic generalisation of the LDS. The basic RNN is described by the following dynamics:

$$\mathbf{x}_t \;=\; \mathbf{f}\left(A\mathbf{x}_{t-1} + B\mathbf{u}_t + \mathbf{b}\right) \tag{2.46}$$

for $t = 1, \ldots, T$, with parameters $\mathbf{x}_0, A, B, \mathbf{b}$. The vector-valued $\mathbf{f}$ is an 'activation function' which acts elementwise on its inputs; common choices include an elementwise tanh or ReLU[9]. Usually a RNN defined by Equation (2.46) is coupled with an emission model, for which the most straightforward is a linear transformation of the state, i.e.:

$$\mathbf{y}_t \;=\; C\,\mathbf{x}_t + \boldsymbol{\epsilon}_t, \qquad \boldsymbol{\epsilon}_t \overset{\text{iid}}{\sim} \mathcal{N}\left(0, \sigma^2 I\right). \tag{2.47}$$

In the same way that an LDS can be seen as a dynamic factor analysis, the RNN is sometimes motivated by augmenting an artificial neural network (e.g. Goodfellow et al., 2016, §6) with 'memory' or 'state', i.e. where $\mathbf{x}_{t-1}$ represents the neuron activations after previous computation. Such a model is surprisingly expressive; with a sufficiently large state dimension, it is universal, in the sense that it can approximate any function that can be computed by a Turing machine (e.g. Siegelmann and Sontag, 1991; Hyötyniemi, 1996).

2.2.4.2 Do RNNs need a stochastic state?

While RNNs have a deterministic state, they can approximate a stochastic state via use of a *teacher forcing* setup. This is motivated by the decomposition of a joint probabilistic distribution:

$$p(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T) \;=\; p(\mathbf{y}_1) \prod_{t=1}^{T} p(\mathbf{y}_t \,|\, \mathbf{y}_{1:t-1}), \tag{2.48}$$

where each factor $p(\mathbf{y}_t \,|\, \mathbf{y}_{1:t-1})$ is approximated by the RNN. This approximation requires only the previous observation $\mathbf{y}_{t-1}$ as an input for each factor; the information from previous ones can be kept in the hidden state, $\mathbf{x}_{t-1}$. This is a nonlinear version of the deterministic model of $p(\mathbf{y}_t \,|\, \mathbf{y}_{1:t-1})$ for LDSs at steady state in Section 2.2.3.6. However, implicitly learning a stochastic model in this way is data intensive, may not be robust to shifts in the data generating process, and can be difficult to regularise (Chung et al., 2015, §2.1). Explicitly stochastic versions of RNNs have also been explored, where the dynamics are subject to a noise process. Practical architectures and learning algorithms have been proposed, which include variational Bayes approaches (e.g. Fraccaro et al., 2016; Goyal et al., 2017) or Monte Carlo objectives, with inference performed by sequential Monte

[9] The rectified linear unit is defined as $\text{ReLU}(x) := \max(x, 0)$.

Carlo (SMC, see e.g. Maddison et al., 2017; Naesseth et al., 2018; Le et al., 2018). However, in many important examples, the implicit stochasticity learned by a deterministic model appears to be adequate.

2.2.4.3 Optimisation of RNNs

Learning a RNN is generally performed by gradient methods, such as those used for feedforward neural networks. However, unlike in the feedforward case, backpropagating gradients does not occur in a fixed-size computational graph, but effectively traverses the dynamical system in reverse. This is avoided in the LDS via use of EM (information is propagated backwards via the RTS smoother instead), but this is not possible with RNNs. The resulting backpropagation through time (BPTT) is notorious for several reasons. Firstly the gradient signal itself undergoes an autoregressive process backward in time, and after a number of time-steps may either decay to zero (the 'vanishing gradient problem') or explode (Hochreiter, 1991; Bengio et al., 1994). Hence determining the optimal step size is challenging. A common related problem is of poor conditioning of the optimisation landscape of RNNs (e.g. Pascanu et al., 2013).[10] Thirdly, the length of the dynamical chain is variable, which prohibits certain computational optimisations, and the memory needed to store the intermediate computation for a backward pass grows linearly with time. With $T = 100$ timesteps (for example), a RNN requires the same computation per gradient-update as a 100-layer neural network.

There have been various proposals to alleviate these problems; we can only provide a review of some highlights here. Williams and Peng (1990) propose *truncated* BPTT (TBPTT), where the gradient signal is truncated after some pre-specified window of time-steps to reduce the computation and memory burden for large T. Pascanu et al. (2013) demonstrate that the highly nonlinear nature of the RNN loss surface can benefit from 'gradient clipping', where the gradient is projected onto a L2 ball with some pre-specified radius. Martens and Sutskever (2011) propose an optimisation method which can incorporate second-order information without explicitly calculating the Hessian. Sutskever et al. (2013) demonstrate that one can achieve similar results to Martens and Sutskever (2011) by use of momentum and careful initialisation, avoiding the need for second-order information.

The initialisation of RNN parameters is a crucial choice for effective optimisation. Sutskever et al. (2013), following the Echo State Network of Jaeger and Haas (2004) suggested initialising the transition matrix A to have a sparse initialisation with a spectral radius of 1.1. They also required a hyperparameter for the scale of the input matrix B. In the case of feedforward neural networks, Glorot and Bengio (2010); He et al. (2015) proposed initialising weights from a distribution designed to ensure the variance of the activations (and gradients) remained unchanged during the forward (resp. backward) pass. These ideas have been applied (without modification) to RNNs and appear to be the current

[10] i.e. some of the Hessians evaluated during the optimisation trajectory are poorly conditioned.

de facto standard. Layer normalisation (Ba et al., 2016) may also be performed, which standardises the activations within each layer at runtime, reducing the need for careful initialisation.

We now introduce two important model architectures that are easier to optimise by design. It is often easier to modify an optimisation problem than improve an optimisation algorithm (cf. *preconditioning*, see Nocedal and Wright, 2006, §5.1), and this is especially crucial for neural architectures, as argued in Goodfellow et al. (2016), §8.7.5.

2.2.4.4 Long Short-Term Memory (LSTM) networks

LSTM networks were introduced in Hochreiter and Schmidhuber (1997), motivated by avoiding the vanishing/exploding gradient problem described above. Learning long-range dependencies with RNNs was deemed difficult at the time, for which this problem appeared fundamental. The key feature of this architecture was the so-called Constant Error Carousel (CEC) which simply copied the state from the previous timestep, thereby ensuring a unit norm gradient which would neither decay nor explode. Hochreiter and Schmidhuber showed that very few other choices of architecture could satisfy this requirement. However, since the state could no longer 'forget' information, the input weights were responsible both for passing through critical information and blocking irrelevant information. Due to the apparent difficulties of learning such weights (and related difficulties with the output transformation), Hochreiter and Schmidhuber introduced input and output 'gates', which provided an additional mechanism to block information flow. Together with the 'forget' gate introduced by Gers et al. (2000), the modern LSTM cell can be described by the following equations:

$$\mathbf{g}_t^i = \mathbf{f}\left(W^i \mathbf{o}_{t-1} + B^i \mathbf{u}_t + \mathbf{b}^i\right) \quad \text{(input gate)} \tag{2.49}$$

$$\mathbf{g}_t^o = \mathbf{f}\left(W^o \mathbf{o}_{t-1} + B^o \mathbf{u}_t + \mathbf{b}^o\right) \quad \text{(output gate)} \tag{2.50}$$

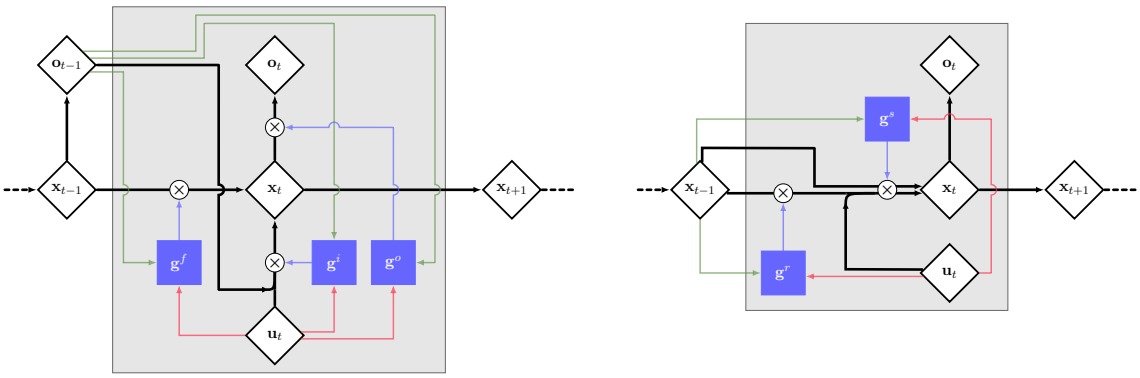
$$\mathbf{g}_t^f = \mathbf{f}\left(W^f \mathbf{o}_{t-1} + B^f \mathbf{u}_t + \mathbf{b}^f\right) \quad \text{(forget gate)} \tag{2.51}$$

$$\mathbf{x}_t = \mathbf{g}_t^f \odot \mathbf{x}_{t-1} + \mathbf{g}_t^i \odot \tanh\left(W^s \mathbf{o}_{t-1} + B^s \mathbf{u}_t + \mathbf{b}^s\right) \quad \text{(state update)} \tag{2.52}$$

$$\mathbf{o}_t = \mathbf{g}_t^o \odot \tanh(\mathbf{x}_t) \quad \text{(output)} \tag{2.53}$$

where we assume $\mathbf{y}_t = \mathbf{o}_t + \boldsymbol{\epsilon}_t$ for some $\boldsymbol{\epsilon}_t \sim \mathcal{N}\left(0, \sigma^2\right)$. The parameters are $\{W^i, W^o, W^f, W^s, B^i, B^o, B^f, B^s, \mathbf{b}^i, \mathbf{b}^o, \mathbf{b}^f, \mathbf{b}^s\}$ and the gates use an elementwise logistic sigmoid $\mathbf{f}$. A visualisation is provided in Figure 2-8a. There are many variations upon this basic definition including different choices of gates and different quantities used in their calculation. For a fairly comprehensive survey, see Greff et al. (2016).

LSTMs are known to substantially outperform basic RNNs on many sequential tasks, such as speech recognition (Graves et al., 2013) and machine translation (Sutskever et al., 2014; Bahdanau et al., 2015). While the LSTM was motivated by avoiding the autoregressive challenges with gradient flow, its impressive performance may in part be explained by the



(a) LSTM cell.

(b) GRU cell. Gate $\mathbf{g}^s$ takes a convex combination of its two lines.

Figure 2-8: Schematic of LSTM and GRU cells. Gates are highlighted in blue, with their multiplicative action marked by the $\otimes$ symbol. The aim is to highlight the relationship to a standard dynamical system; transformations are not shown to reduce clutter.

gates, allowing multiplicative relationships between inputs, outputs and state which are unavailable in basic RNNs (cf. Jayakumar et al., 2020). We will encounter an example in Chapter 5 with *short*-term relationships which performs better with a gated architecture; and Martens and Sutskever (2011) found that with appropriate optimisation methods, basic RNNs could achieve similar (or improved) performance to LSTMs on tasks with *long-term* dependencies.

2.2.4.5 Gated Recurrent Units

Another popular recurrent cell is the GRU, developed by Cho et al. (2014), which has only two gates. This simpler model retains an analogy to the CEC of the LSTM, whereby information (and gradients) can be retained in principle over long time intervals. The GRU is defined by the following equations:

$$\mathbf{g}_t^r = \mathbf{f}\left(A^r \mathbf{x}_{t-1} + B^r \mathbf{u}_t + \mathbf{b}^r\right) \quad \text{(reset gate)} \tag{2.54}$$

$$\mathbf{g}_t^s = \mathbf{f}\left(A^s \mathbf{x}_{t-1} + B^s \mathbf{u}_t + \mathbf{b}^s\right) \quad \text{(update gate)} \tag{2.55}$$

$$\hat{\mathbf{x}}_t = \tanh\left(A^{\mathbf{x}}(\mathbf{g}_t^r \odot \mathbf{x}_{t-1}) + B^{\mathbf{x}} \mathbf{u}_t + \mathbf{b}^{\mathbf{x}}\right) \quad \text{(new state proposal)} \tag{2.56}$$

$$\mathbf{x}_t = (1 - \mathbf{g}_t^s) \odot \mathbf{x}_{t-1} + \mathbf{g}_t^s \odot \hat{\mathbf{x}}_t \quad \text{(state update)} \tag{2.57}$$

where $\mathbf{f}$ is an elementwise logistic sigmoid. See Figure 2-8b for a visualisation. This model has fewer parameters than the LSTM, but often enjoys similar or better performance to an LSTM (see e.g. Chung et al., 2014). The LSTM and GRU share the idea of a CEC paired with a forget gate, which appears to be a powerful combination (van der Westhuizen and Lasenby, 2018). Nevertheless, the GRU is fundamentally different to the LSTM, and was

derived from a different motivation.[11] Crucially, the GRU couples the input and forget gates via convex combination, and allows the new state proposal (Equation 2.56) to 'forget' some of the current state. To the best of our knowledge, there is no principled reason for choosing one architecture over the other, both appear to have advantages for different datasets, however the GRU is sometimes preferred due to its lower parameter count.

2.2.5 A note on continuous time models

The aim of the section above is to provide an overview of discrete-time models where the index set $\mathcal{T} \subset \mathbb{Z}$. An extensive presentation of *continuous-time* models i.e. where $\mathcal{T} \subset \mathbb{R}$ is also possible, but out of scope for this thesis. The choice of continuous- vs discrete-time can divide opinion into camps. In principle continuous-time models dominate discrete-time models in terms of generality, since the former can be immediately applied to irregularly sampled time series while the latter cannot. However, continuous-time models can suffer from increased complexity of their derivation and/or implementation, and often an increase in computational complexity. This can limit their applicability in practice, if not in principle. It is also possible to coerce discrete-time models to apply to irregularly sampled time series by increasing the model's effective sampling frequency, handling missing values appropriately, and/or using nonparametric or semiparametric interpolation methods (e.g. Shukla and Marlin, 2019; Li and Marlin, 2016).

A classic approach to continuous-time modelling is via Gaussian processes (see e.g. Rasmussen and Williams, 2006, and especially Appendix B on Gaussian Markov Processes). Some relevant work in this area which pertains to the contributions of this thesis is given in Section 3.7. Gaussian processes have previously been considered problematic for longer time series due to a complexity of $\mathcal{O}(|\mathcal{T}|^3)$ for vanilla implementations. We will simply note that more efficient methods are possible (see Liu et al., 2020, for a recent review), especially for time series models (Hartikainen and Särkkä, 2010). Furthermore, flexible 'neural' models for continuous time series data have also been proposed recently (Chen et al., 2018; Rubanova et al., 2019), including architectures which embed physical priors (Greydanus et al., 2019; Cranmer et al., 2020), which can be useful for modelling physical time series data.

[11] GRUs were apparently developed fairly independently of LSTMs (Cho, 2015, §4.2.3). They are motivated by considering the read and write access patterns from/to a CPU cache. Whereas a basic RNN CPU 'reads' and 'writes' to the entire memory, $\mathbf{x}_t$, at each time t, a CPU typically reads and writes from a sparse subset of memory locations, motivating the use of gated updates.

$p(\mathbf{u})$	$\mathcal{U}$	$p(\mathbf{y}\|\mathbf{u})$	$\mathcal{Y}$	Paradigm
✓	✓	✓	✓	single task
✓	✓	✗	·	multi-output task
✗	✓	·	·	homogeneous feature MTL
✗	✓	✓	✓	– covariate shift tasks
✗	✓	✗	✓	– domain adaptative tasks
✗	✗	·	·	heterogeneous feature MTL

Table 2.1: A classification of MTL according to which quantities are shared between tasks. From left-to-right, (✓) indicates sharing of $p(\mathbf{u})$ (input distribution), $\mathcal{U}$ (input domain), $p(\mathbf{y}|\mathbf{u})$ (output conditional distribution), $\mathcal{Y}$ (output domain); an orange dot (·) indicates that the quantity can be shared or unshared. Two well-known special cases of homogeneous feature MTL are shown below it; other cases are possible. This table draws from multiple sources (Zhang and Yang, 2017; Sugiyama and Kawanabe, 2012; Redko et al., 2019).

2.3 Multi-task learning

Machine learning is commonly engaged with a single *learning task*. In the context of supervised learning (which is our focus in this thesis), a task $\mathcal{T}$ is a set of input-output pairs[12] $(\mathbf{u}, \mathbf{y}) \in \mathcal{U} \times \mathcal{Y}$ with density $p_{\text{true}}(\mathbf{u}, \mathbf{y})$ for which the goal is to learn an approximation to the conditional density $p_{\text{true}}(\mathbf{y} \,|\, \mathbf{u})$. Where data are not plentiful, the mapping $\mathcal{U} \to \mathcal{Y}$ can be severely underdetermined, and the learned model generalises poorly. *Multi-task learning* (MTL) seeks to *share information between multiple tasks*, $\{\mathcal{T}_i\}_{i=1}^N$, and thereby improve the model estimates for each $\mathcal{T}_i$. MTL can thus allow learning of a more complex model than each task's data can reliably support.

There are a variety of MTL setups under various different names, and not every set of tasks is considered to result in MTL. Table 2.1 provides an overview of some cases, including 'covariate shift' and 'domain adaptive' types (Sugiyama and Kawanabe, 2012; Redko et al., 2019, §2).[13] Our focus will be on the 'homogeneous feature' class of MTL, where the support of the inputs, $\mathcal{U}$, is the same across all tasks. The complement of this class, 'heterogeneous feature MTL' (Zhang and Yang, 2017) is more challenging and is less well studied. We will also focus this survey on iid problems; one of our core contributions is to extend a general form of MTL to time-structured problems.

In the below, we provide an introduction to MTL (Section 2.3.1) along with some common approaches (Section 2.3.2). We note that this is a vast area of machine learning; for a more thorough review, see the survey paper by Zhang and Yang (2017). Due to the proliferation of related paradigms, we provide a clarification of the relationship of MTL to transfer learning, meta learning and random effects models (among others) in Section 2.3.3.

[12] It is conventional to use $\mathbf{x}$ for the input variable in supervised learning; we follow the convention from the time series literature where $\mathbf{x}$ is reserved for the latent state, and $\mathbf{u}$ is used for control inputs. For similar reasons we will use the index set $t = 1, \ldots, T$ for the datapoints.

[13] We borrow this terminology from transfer learning, more on which in Section 2.3.3.

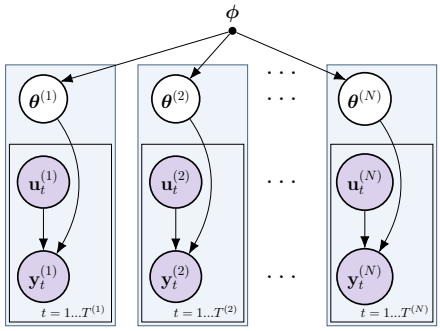


Figure 2-9: Graphical model of MTL for conditionally iid data. Each task for $i = 1, \ldots, N$ is represented in blue, where $\boldsymbol{\theta}^{(i)}$ governs the task relationship.

2.3.1 Introduction

Intelligence in the natural world is marked by its ability to learn efficiently from a series of tasks (see e.g. Kemp et al., 2007). In contrast, most applications of ML are in a *tabula rasa* setting, where relationships are learned from scratch from a potentially large set of hypotheses. Finding those hypotheses that generalise well usually requires a large amount of data, resulting in a high sample complexity for each task. In order to avoid this requirement, one may specify an *inductive bias*, $\mathcal{H}$, a probability measure which specifies a priori more likely hypotheses (Mitchell, 1991; Baxter, 2000). An example of a useful inductive bias for image classification might be to remove all hypotheses which are not functions of predefined features such as Gabor filters, face detectors, wheel detectors etc.

Multi-task learning seeks to *learn* an inductive bias, $\mathcal{H}$, by finding one which performs well for all tasks in a training set. We are usually interested in learning a $\mathcal{H}$ which improves the generalisation error of the training tasks, as well as novel tasks. However, without constraints, such a procedure may simply return the union of the single task models. The key question is how to define the search space for $\mathcal{H}$; what are the admissable inductive biases? We will provide a number of common answers in Section 2.3.2.

Our presentation will follow a probabilistic approach where each task $\mathcal{T}_i$ is defined by a likelihood $p(\mathbf{y} \,|\, \mathbf{u}; \boldsymbol{\theta}^{(i)})$, parameterised by $\boldsymbol{\theta}^{(i)} \in \Theta$. MTL is then defined as *learning a density over* Θ *from the tasks* $\{\mathcal{T}_i\}_{i=1}^N$*, resulting in an inductive bias that improves the average performance of the tasks.* This is a hierarchical model (see Figure 2-9) defined by:

$$\boldsymbol{\theta}^{(i)} \sim p(\boldsymbol{\theta}\,; \boldsymbol{\phi}) \tag{2.58}$$

$$\mathbf{y}_t^{(i)} \sim p(\mathbf{y} \,|\, \mathbf{u}_t^{(i)}; \boldsymbol{\theta}^{(i)}), \qquad t = 1, \ldots, T^{(i)}, \tag{2.59}$$

for some prior density $p(\boldsymbol{\theta}\,; \boldsymbol{\phi})$ parameterised by $\boldsymbol{\phi}$, and $T^{(i)}$ is the number of observations. This model may be learned via MAP optimisation (with point estimates for $\{\boldsymbol{\theta}^{(i)}\}$) or in a fully Bayesian manner. This description is easily extended to generative models by specification of some parametric distribution $p(\mathbf{u}_t^{(i)}; \boldsymbol{\theta})$ common to all i. If a vector of task-specific attributes is available for each task (known variously as 'task descriptors',

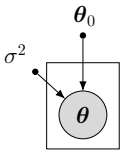
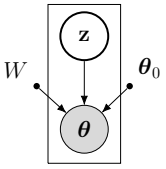
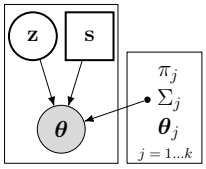
	(a) Shared	(b) Shrinkage	(c) Subspace	(d) Cluster
Graphical model				
Marginal prior	$\delta(\boldsymbol{\theta} - \boldsymbol{\theta}_0)$	$\mathcal{N}\left(\boldsymbol{\theta} \mid \boldsymbol{\theta}_0,\, \sigma^2 I\right)$	$\mathcal{N}\left(\boldsymbol{\theta} \mid \boldsymbol{\theta}_0,\, WW^\mathsf{T}\right)$	$\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\boldsymbol{\theta} \mid \boldsymbol{\theta}_j,\, \Sigma_j\right)$

Figure 2-10: Implicit prior structure of various MTL models as graphical models. MT parameters are generated probabilistically, possibly via a continuous latent variable $\mathbf{z} \in \mathbb{R}^{n_z}$ and a discrete random variable $s \in \{1, \ldots, K\}$.

'metadata', or 'side information'), these can be used to augment the task parameters, see e.g. Bonilla et al. (2008).

2.3.2 Common approaches

While we have suggested an abstract formulation of MTL above, it remains to consider: (i) what classes of prior $p(\boldsymbol{\theta}\,;\boldsymbol{\phi})$ may be appropriate; (ii) which collections of tasks may be improved by MTL. We will treat each point in turn in the below. Our treatment here is applicable to all types of MTL in Table 2.1; further specialisations may be possible in individual cases.

2.3.2.1 Types of inductive bias

In many cases, each task has some task-specific parameters, tuned without reference to the other tasks. Partitioning the parameter space into task-specific and multi-task parameters yields $\Theta = \Theta_{\text{task}} \times \Theta_{\text{MT}}$, where the inductive bias is limited to $\boldsymbol{\theta}_{\text{MT}} \in \Theta_{\text{MT}}$. In principle a wide variety of choices of $p(\boldsymbol{\theta}_{\text{MT}};\, \boldsymbol{\phi})$ may be used (see the survey paper by Zhang and Yang, 2017). However, there are a few common choices in practice:

(a) Learning *shared representations* across tasks. This approach is often applied in deep models (often called a 'multi-head' architecture), where the $\boldsymbol{\theta}_{\text{MT}}$ are the parameters of the lower layer(s), fixed to some $\boldsymbol{\theta}_0$ across tasks (Caruana, 1993). (This is similar to use of pre-trained nets such as VGG or Inception for image classification.) We may interpret this model probabilistically by giving the MT parameters $\boldsymbol{\theta}_{\text{MT}}$ the hard 'shared' prior of Figure 2-10(a).

(b) *Shrinking* parameter estimates towards a common value $\boldsymbol{\theta}_0$. It is common to use $\boldsymbol{\theta}_0 = \mathbf{0}$ and a sparsity inducing distribution (e.g. Laplace) to perform feature selection (Argyriou et al., 2007). This is a softer version of the above approach, see Figure

2-10(b). A wide variety of penalties (or prior distributions) may be used here, see §2.1, Zhang and Yang (2017) for more details.

(c) Learning a *low rank structure* over the task parameters (Ando and Zhang, 2005). This approach assumes that the $\{\boldsymbol{\theta}_{\text{MT}}{}^{(i)}\}_{i=1}^{N}$ can be found in a common low-dimensional subspace (see Figure 2-10(c)). This approach has the alternative dual interpretation of finding a low-dimensional feature subspace common to all models.

(d) *Clustering* the tasks via the parameters (Bakker and Heskes, 2003), see Figure 2-10(d). This approach partitions the tasks, and only borrows strength across tasks which appear to be similar. One may use either hard-clustering (as in K-means) or soft clustering (as in a Gaussian mixture model). Where the number of clusters is not known, one may use a nonparametric approach such as in Xue et al. (2007).

More generally we may use any latent variable model in parameter space, such as the nonlinear VAE described in Section 2.1.3. Furthermore, we may use various approaches in combination since they have complementary properties. The 'shared representation' prior is used to find a common (possibly large and possibly nonlinear) feature space, which can be used for all tasks. The 'subspace' prior restricts to a low-dimensional subspace of this feature space which forces tasks to use similar combinations of features. The 'cluster' prior provides some automation of selection of similar tasks, grouping relevant tasks together.

2.3.2.2 Which tasks are related?

If tasks are unrelated, a learned inductive bias $\mathcal{H}$ will either provide little benefit, or be overfit to the peculiarities of the given tasks $\{\mathcal{T}_i\}_{i=1}^{N}$. Such overfitting can result in *negative transfer* between tasks (Pan and Yang, 2010). However, to the best of our knowledge, no generally accepted definition of 'related tasks' is available to aid this choice.[14] Tasks are usually chosen via human intuition; see Caruana (1998), §3, for some heuristics and further discussion. Examples of related tasks include different image classification problems (e.g. Lu et al., 2017), predicting treatment success across different hospitals (Caruana, 1998, §4), or sentiment analysis of different products (e.g. Zhang and Yeung, 2010).

Tasks which intuitively share some aspects of information processing are likely to obtain some benefit when learned together. However, as tasks become less similar to each other, it may become more difficult to learn their commonalities; a feature shared by humans (e.g. Hume and Pazzani, 1996). Furthermore it may be unclear how to partition the parameter space into Θ_{task} and Θ_{MT}. Since task selection, and choosing the right type of inductive bias (Section 2.3.2.1) is difficult, it will often be necessary to perform model selection and hyperparameter tuning to avoid negative transfer and capitalise on task commonalities. If sufficient computational resource is available, a sensible approach to task selection is

[14] Attempts to define task relatedness include Ben-David and Schuller (2003) via PAC bounds (who require existence of a function that can transform one task into another) and Mahmud and Ray (2008) via compression (using Kolmogorov complexity). A heuristic but easily computable metric is given in Thrun and O'Sullivan (1996) via calculation of all pairwise task performances.

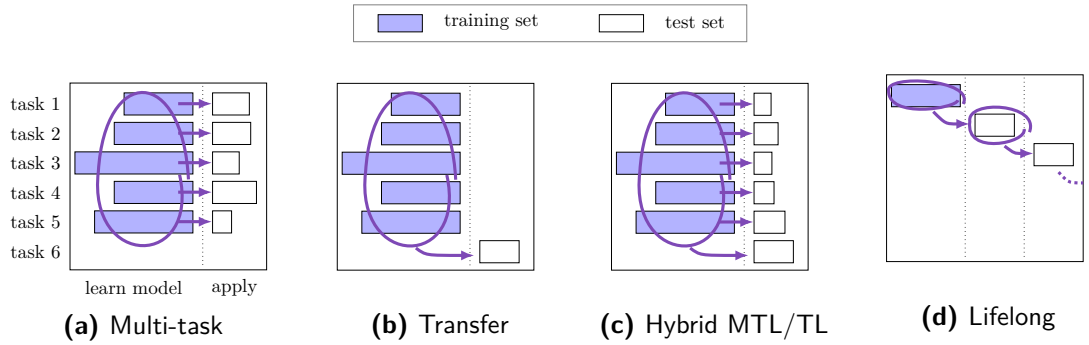


Figure 2-11: Various learning paradigms compared to MTL. Model is learned on available tasks, shown before dotted line (purple), and applied to unseen data / tasks, shown after (white).

to provide a large candidate pool of tasks, and allow such techniques to choose a useful subset.

2.3.3 Related paradigms

A number of paradigms related to MTL are defined based on availability and importance given to the different tasks. Below we discuss transfer learning, meta-learning and random-effects models.

Transfer learning. Whereas MTL is concerned (in principle) with improving performance on *all* the tasks in the training set, *transfer learning* (TL) is concerned primarily with future as-yet-unknown tasks; see Pan and Yang (2010) for a useful overview. The goal is to use the information extracted from the source tasks at training time in order to learn the target task more efficiently. See Figures 2-11a, 2-11b for a schematic illustration of this comparison. Special cases include *domain adaptation* and *covariate shift*, defined in Table 2.1. Where strong performance is expected after only a small number of examples of each class, TL may be called 'one-shot learning' or 'few-shot learning' (see e.g. Fei-Fei et al., 2006; Lake et al., 2015).

Models learned by MTL may also be intended for use on unseen tasks (e.g. Xue et al., 2007), which blurs the line somewhat. We consider a *hybrid* MTL and TL approach to require strong performance on both the training tasks *and* the unseen test task(s) (see Figure 2-11c). Another distinction is *lifelong machine learning* (Thrun, 1996; Chen and Liu, 2016) which may be seen as a repeated application of TL (Figure 2-11d) where previous task data cannot be revisited.[15] Lifelong learning without 'catastrophic forgetting' of previous information (McCloskey and Cohen, 1989) is an important challenge on the path to creating intelligent agents.

[15] This is sometimes also known as *continual learning*, usually (not always) in the context of reinforcement learning.

Meta learning. *Meta-learning*, or *learning to learn* (e.g. Schmidhuber, 1987; Bengio et al., 1992) is a field pertaining to models that can 'improve themselves' which has recently seen renewed interest. In contrast to standard MTL and TL approaches, meta-learning emphasises learning the entire process of model learning, including the optimiser.[16] However, due to the well-known problem of induction (Mitchell, 1991), statistical efficiency must be gained through extracting knowledge from the training set, and despite some philosophical differences, meta-learning and transfer learning appear to be practically equivalent. Indeed, if we are to take the definition of 'learning to learn' in Thrun and Pratt (1998) seriously, it is equivalent to TL, or a hybrid MTL / TL paradigm. With that said, framing the problem with a different narrative appears to have resulted in some significant advances (e.g. Santoro et al., 2016; Vinyals et al., 2016; Ravi and Larochelle, 2017; Finn et al., 2017).

Mixed effects models. Mixed effects (ME) models (which can be traced back at least to Fisher, 1919) allow different experimental units, such as people or organisations, to use related but distinct models. (It is assumed that each unit produces multiple observations.) These models share the same architecture, but some parameters are considered *latent*, specific to each unit, known as *random effects*. The other parameters are shared and known as *fixed effects*; a *mixed effect* model has both kinds of effect. Clearly when each unit corresponds to a task, mixed effects models are a form of MTL. This is equivalent to the 'shrinkage' approach discussed in 2.3.2.1, where $\boldsymbol{\theta}_{\text{MT}}$ has the prior of Figure 2-10(b). However, due to their frequentist interpretation, ME models *integrate out* the latent variable, attributing it to 'nuisance variation' rather than considering it of direct interest (see e.g. Pinheiro and Bates, 2000, §1.1). Furthermore, due in part to the requirement of deriving various statistical quantities for parameters (such as confidence intervals and hypothesis tests), ME models tend to have a fairly simple structure. It is relatively uncommon to find complex base models or random effects with non-Gaussian distributions.

Most importantly, the goal is not to learn an inductive bias over parameters, but to induce a task-aware covariance structure between observations, in order to estimate underlying parameters better. To the best of our knowledge, gaining statistical strength across tasks by use of correlation, low rank structure or nonlinear dependency between latent variables does not feature much, if at all in the ME literature. Hence for the purposes of MTL, ME models appear in practice to be limited. The difference is mostly one of philosophy and intention, however. ME models may also be interpreted in a Bayesian manner (see e.g. Gelman and Hill, 2007) as hierarchical models, and hence may be readily extended to more powerful MTL models.

[16] Note that 'meta-learning' is sometimes (but less frequently) used to refer to architecture optimisation and other hyper-parameter prediction (e.g. Vanschoren, 2018).

CHAPTER 3

Multi-Task Dynamical Systems

Perhaps the most important class of time series models today are dynamical systems (introduced in Chapter 2), which encompass a wide variety of models. Applications may be found in domains as diverse as physical modelling (Linderman et al., 2017), activity monitoring (Nweke et al., 2018), drug response (White et al., 2008), system failure logs (Zhang et al., 2016), public transport demand forecasting (Toqué et al., 2016), motion capture (Martinez et al., 2017), retail sales data (Rangapuram et al., 2018), automatic speech recognition (ASR, Chiu et al., 2018), and natural language processing (NLP, Wu et al., 2016). Since such time series data can arise from various sources (such as different people, systems, locations or organisations), each dataset often comprises a variety of sequences with different characteristics. For instance, motion capture data might include different styles of walking, and healthcare data might exhibit a variety of personalised responses to the same drug. These characteristic differences often require different dynamics, which a single dynamical system is unable to provide – at least explicitly.

The goal of this thesis is to allow dynamical systems to take advantage of this inter-sequence variation, enabling 'personalisation' or 'customisation' of models in a given context. The present chapter introduces our proposed approach: the multi-task dynamical system (MTDS), together with the central methodological details. Section 3.1 provides an introduction to the model, and Section 3.2 examines two special cases (the multi-task LDS, and multi-task RNN). Section 3.3 provides further motivation for the MTDS, discussing a variety of ways it may be used. Section 3.4 provides methods to learn the model and Section 3.5 discusses task inference and proposes an efficient approach. Section 3.6 discusses some adaptations to the learning procedure that we have found helpful in learning these models. Finally, Section 3.7 closes with a discussion of related work.

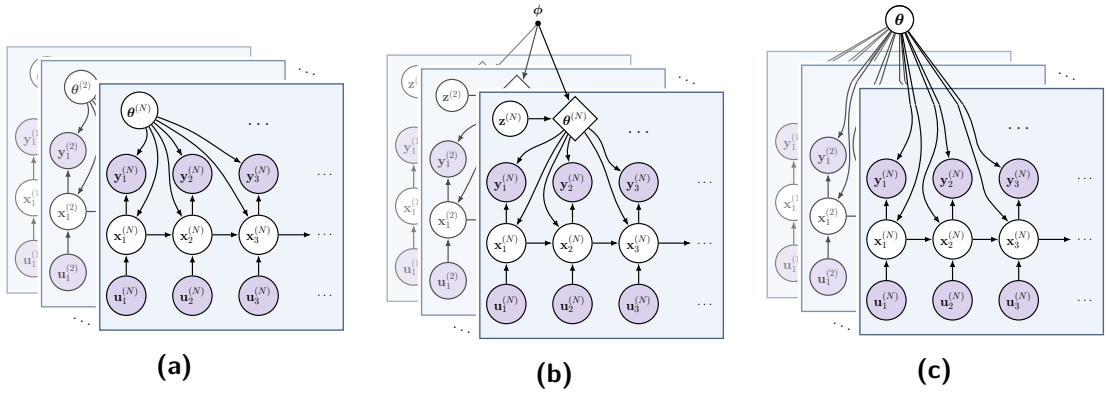


Figure 3-1: Comparison of approaches for modelling multiple sequences. (a) single task approach, where each sequence is learned separately; (b) the multi-task dynamical system; (c) the pooled approach.

3.1 The Multi-Task Dynamical System (MTDS)

There are two common approaches[1] to modelling this inter-sequence variation: the most flexible option is to train an individual model per sequence (as per the graphical model in Figure 3-1a). An individual model can in principle capture these features, but suffers from overfitting and fails to exploit the regularities between the sequences in the training set. More commonly, the different sequences are *pooled* together to train a single dynamical system, despite the inter-sequence variation (a one-size-fits-all approach, Figure 3-1c). This may fail to capture the idiosyncratic features at all; a simple model such as a linear dynamical system (LDS) will learn only an average effect.

In contrast to these approaches, we aim to learn a *family* of dynamical systems that is consistent with the sequences in the training set. Each training sequence is given a bespoke model from the 'sequence family', but this family exploits the regularities observed across all sequences, which will often substantially reduce overfitting. We achieve this via use of a set of hierarchical latent variables, similar to the approach used in multi-task learning for iid models, now with each *sequence* treated as a task. We hence call this approach the *multi-task dynamical system* (MTDS). The MTDS learns a low dimensional manifold in parameter space, indexed by a latent code $\mathbf{z} \in \mathcal{Z}$, which corresponds to the specialisation of each sequence model. See Figure 3-1b for a graphical model.

The choice of a low dimensional manifold enables the MTDS to determine directions in parameter space with respect to which the parameters $\boldsymbol{\theta} \in \Theta$ need not change, sharing strength across sequences. This avoids considering directions in Θ which result in unlikely or uncharacteristic predictions, as well as ignoring so-called 'sloppy directions' (Transtrum et al., 2011), which can add great complexity to inference, with little benefit in terms of fit or generalisation. But it can also find the key direction(s) of variability in Θ between training sequences. As a simple example, consider sequences generated from the family

[1] We address more sophisticated approaches in the related work in Section 3.7.

of dth order linear ODEs. Such sequences may vary in terms of oscillation frequencies, magnitude, half life to an input impulse, etc. As discussed in Section 2.2.2.7, this sequence family may be approximated by a LDS, but while the ODE has d+1 degrees of freedom, the LDS has $\mathcal{O}(d^2)$. An MTDS approach can *learn* the relevant d+1 degrees of freedom of the LDS in Θ simply by training on example sequences, as well as learning the relative probabilities of each configuration (for an example of this, see Chapter 4).

Unlike previous efforts to customise time series models, our MTDS construction allows application to general classes of dynamical systems. This is available due to our choice of learning an *arbitrary manifold* over *all* the model parameters (both system and emission), rather than being constrained to pre-determined parameter sets (see related work, Section 3.7). This allows the MTDS to model variability in observation space (such as magnitude or offset), as well as different responses to inputs, different sensitivity to initial conditions, and differences in dynamic evolution. The MTDS can thus be applied to classical ARMA models, state space models, and modern recurrent neural networks (RNNs).

The MTDS provides user visibility of the task specialisation, as well as the ability to control it directly if desired via use of the hierarchical latent variable, $\mathbf{z}$. This stands in contrast to a RNN approach, which also has the flexibility to model inter-sequence variation, but does so in an implicit and opaque 'black-box' manner. This prohibits interpretability and end-user control, but also can suffer from mode drift where the personalisation erroneously changes over time. For example, in Ghosh et al. (2017) a RNN generates a mocap sequence which performs an unprompted transition from walking to drinking. Our contributions with respect to such end-user control will be explored further in the empirical work, especially in Chapter 5.

The following material will define and develop the necessary technical aspects for learning and inference of such a model. Discussion of the practical uses of the MTDS is given in Section 3.3 in advance of demonstrating its benefits in our experimental work in Chapters 4 - 6.

3.1.1 Model definition

Consider a collection of input-output sequences $\mathcal{D} = \left\{ (\mathbf{u}^{(i)}_{1:T_i},\ \mathbf{y}^{(i)}_{1:T_i}) \right\}_{i=1}^{N}$ consisting of inputs and outputs respectively, where T_i is the length of sequence i. Each sequence i is described by a different dynamical system, whose parameter $\boldsymbol{\theta}^{(i)} \in \Theta$ depends on the hierarchical latent variable $\mathbf{z}^{(i)} \in \mathcal{Z}$. See Figure 3-1b for a graphical model, where the state variables are denoted $\mathbf{x}_t \in \mathcal{X}$, $t = 1, \ldots, T$. The MTDS is defined by the equations:

$$\boldsymbol{\theta}^{(i)} = \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z}^{(i)}), \quad \mathbf{z}^{(i)} \sim p(\mathbf{z}) \tag{3.1}$$

$$\mathbf{x}^{(i)}_t \sim p(\mathbf{x} \mid \mathbf{x}^{(i)}_{t-1},\ \mathbf{u}^{(i)}_t,\ \boldsymbol{\theta}^{(i)}), \tag{3.2}$$

$$\mathbf{y}^{(i)}_t \sim p(\mathbf{y} \mid \mathbf{x}^{(i)}_t,\ \mathbf{u}^{(i)}_t,\ \boldsymbol{\theta}^{(i)}), \tag{3.3}$$

for $t = 1, \ldots, T_i$ for each $i = 1, \ldots, N$. (Unless otherwise specified, we assume $\mathbf{x}_0 := \mathbf{0}$, but other choices are possible). In this thesis we further assume $\mathcal{Z} = \mathbb{R}^k$ which the vector-valued function $\mathbf{h}_{\boldsymbol{\phi}}(\cdot)$ transforms to conformable model parameters $\boldsymbol{\theta} \in \mathbb{R}^d$. By restricting the parameter manifold to $k \ll d$ dimensions, eqs. (3.1-3.3) result in a multi-task model rather than simply a hierarchical model.

In order to complete the specification of the MTDS model we must specify three key quantities:

1. The base model (eqs. 3.2-3.3), such as a LDS or RNN.
2. The dimensions of $\boldsymbol{\theta}$ that depend on the latent variable $\mathbf{z}$ (for example, one might choose the emissions (Equation 3.3) to be constant wrt. $\mathbf{z}$, or to modulate only the offset/bias terms of eqs. (3.2) and (3.3) wrt. $\mathbf{z}$).
3. The choice of prior $p(\boldsymbol{\theta})$, that is, the choice of distribution $p(\mathbf{z})$ and transformation $\mathbf{h}_{\boldsymbol{\phi}}$.

The choice of quantities (1) and (2) will be problem-specific (we give examples in Section 3.2), but the choice of prior (3) can be discussed in a more general context.

3.1.1.1 Choice of prior

A general purpose choice for the prior over $\boldsymbol{\theta}$ is a *nonlinear factor analysis model*, described by:

$$\boldsymbol{\theta} = \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z}), \qquad p(\mathbf{z}) = \mathcal{N}(0, I), \tag{3.4}$$

where $\mathbf{h}_{\boldsymbol{\phi}}$ is some deterministic function, such as a multilayer perceptron (MLP), (cf. Kingma and Welling, 2014; Rezende et al., 2014). This permits the parameter density to be non-Gaussian (via use of $\mathbf{h}_{\boldsymbol{\phi}}$) and/or lie on a nonlinear manifold. Other choices are possible, but this choice can make use of a vast literature, and a variety of implementations and generalisations. (Clearly if conjugate distributions (see e.g. Bernardo and Smith, 2009, §5.2) are available for a given time series model, a conjugate prior may be a better choice.)

Unlike the standard usage of such architectures (VAEs, Section 2.1.3), the dimension d of the output space $\Theta = \mathbb{R}^d$, corresponding to the system and emission parameters, may be very large, e.g. $\mathcal{O}(10^6)$ for RNNs. The parameter $\boldsymbol{\phi}$ will therefore be even larger; in the case of an affine $\mathbf{h}_{\boldsymbol{\phi}}$, the parameter will have $(k+1)\times d$ dimensions, and for an MLP, $\boldsymbol{\phi}$ could be orders of magnitude larger even than this. Practical choices of prior will restrict the final layer of such an MLP to be relatively small, reducing the flexibility of this nonlinear approach. Nevertheless, use of an MLP may still be advantageous since, for example:

(i) The MLP can result in non-Gaussian densities in parameter space, even if the resulting manifold is approximately linear.

(ii) A linear space of recurrent model parameters can yield highly non-linear changes even to simple dynamical systems via bifurcation (see e.g. Strogatz, 2018, §8). We speculate it might be advantageous to curve the manifold to avoid such phenomena.

(iii) More expressive choices may help utilisation of the latent space (see e.g. Chen et al., 2017).

Nonetheless, it may often be reasonable to use a linear factor analysis model (i.e. $\mathbf{h}_\phi$ is affine) when $\boldsymbol{\theta}$ is large. Empirically we have observed higher marginal likelihoods for smaller state space models when using a nonlinear manifold (see Section 4.2.2), but affine and nonlinear manifolds may work equally well for larger RNN models (see Section 5.1.2.1). We note finally that when the number of tasks is fairly small, a Gaussian Process Latent Variable Model (GPLVM, Lawrence, 2005) may also be appropriate.

3.2 Examples

In order to make the framework described more concrete, we present two general choices of the base model and their dependence on $\mathbf{z}$: a multi task linear dynamical system (MT-LDS) and a multi-task RNN (MT-RNN). In order to reduce the notational clutter, dependence on $\mathbf{z}$ is indicated with a subscript (e.g. $A_\mathbf{z} := A(\mathbf{z})$).

3.2.1 Multi-task linear dynamical systems (MT-LDS)

We first show how the linear dynamical system (LDS), introduced in Section 2.2.3, may be extended to learn a family of sequence models via a multi-task construction. While powerful, LDS models are unable to perform much in the way of customisation to inter-sequence variation; the multi-task LDS (MT-LDS) offers far greater flexibility. For a given $\mathbf{z}$, a MT-LDS can be described by:

$$\mathbf{x}_t = A_\mathbf{z}\mathbf{x}_{t-1} + B_\mathbf{z}\mathbf{u}_t + \mathbf{b}_\mathbf{z} + \mathbf{w}_t, \tag{3.5}$$

$$\mathbf{y}_t = C_\mathbf{z}\mathbf{x}_t + D_\mathbf{z}\mathbf{u}_t + \mathbf{d}_\mathbf{z} + \boldsymbol{\epsilon}_t, \tag{3.6}$$

for $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, R_\mathbf{z}), \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, S_\mathbf{z})$, with $\boldsymbol{\theta}_\mathbf{z} = \{A_\mathbf{z}, B_\mathbf{z}, \mathbf{b}_\mathbf{z}, C_\mathbf{z}, D_\mathbf{z}, \mathbf{d}_\mathbf{z}, R_\mathbf{z}, S_\mathbf{z}\} = \mathbf{h}_\phi(\mathbf{z})$. The parameterisation of $\boldsymbol{\theta}_\mathbf{z}$ must satisfy the constraints of positive definite $R_\mathbf{z}$ and $S_\mathbf{z}$ and usually a stable $A_\mathbf{z}$ (i.e. $\|A_\mathbf{z}\|_2 \leq 1$) for all $\mathbf{z}$. We discuss parameterisations of each of the matrices A, B, R, S below. We will make use of the fact that the same model over the observations can be achieved with different configurations of $A, B, \mathbf{b}, C, Q$ (i.e. LDS models are overparameterised, see Section 2.2.3.2).

Parameterisation of A. For the vast majority of applications, we require that the matrix A satisfies the stability constraint:

$$\|A\|_2 \leq 1, \tag{3.7}$$

(see Section 2.2.2.4). However, for sequences with medium/long-term dependencies, we require $\|A\|_2 \approx 1$, and hence $A_{\mathbf{z}}$ is likely to violate the stability constraint for some $\mathbf{z}$. This is an example of a bifurcation (Strogatz, 2018, §8), where a small change in the parameters can yield a qualitatively different behaviour of a system. Since Equation (3.7) must be satisfied for *all* $\mathbf{z}$, we cannot rely on projection methods, such as Siddiqi et al. (2008) to enforce the constraint. For univariate autoregressive time series models, one may consider a partial autocorrelation parameterisation and constrain the parameters within the interval $[-1, 1]$. We instead choose a general parameterisation that respects the constraint by construction.

Equation (3.7) is equivalent to ensuring that the singular values of A lie within the unit hypercube (since singular values are non-negative). Consider the singular value decomposition (SVD), $A = U\Sigma V^{\mathsf{T}}$. Instead of learning A, we can learn each of these factors, learning two orthogonal matrices U and V, and enforcing the constraint on Σ using an elementwise nonlinearity. The orthogonal matrices can be parameterised in a number of ways (see e.g. Khuri et al., 1989). A straight-forward choice is the *Cayley transform*. If Q is an orthogonal matrix that does not have the eigenvalue -1, then it may be written in Cayley's form:

$$Q = (I - \mathcal{S})(I + \mathcal{S})^{-1}, \tag{3.8}$$

where $\mathcal{S}$ is skew-symmetric (Khuri et al., 1989). In order to permit negative eigenvalues, we can pre-multiply each orthogonal matrix by a diagonal matrix E with elements in $\{-1, +1\}$. We then have $A = UE_U\Sigma E_V V^{\mathsf{T}}$, with two such diagonal matrices E_U, E_V corresponding to the two orthogonal matrices, and these may be absorbed into Σ. In our implementation we have used $\Sigma = \mathrm{diag}\{\tanh(\boldsymbol{\upsilon})\}$ for some vector $\boldsymbol{\upsilon}$, which constrains the diagonal elements within $[-1, 1]$ as required. As a final point of implementation, we have found it necessary to reduce the scale of the inputs to the Cayley transformation relative to the other parameters, otherwise learning can become unstable. See appendix A.3.1 for further discussion of the Cayley transformation and other learning-related issues.

However, a further simplification is possible. Due to the degeneracy of the LDS, we may transform any set of LDS parameters with an arbitrary invertible G, without affecting the distribution over $\mathbf{y}_{1:T}$ (see Section 2.2.3.2, eqs. 2.33, 2.34). Under the transformation $A \leftarrow G^{-1}AG$,[2] choosing $G = U$, i.e. the left singular values of A, we have:

$$A = U^{-1}U\Sigma V^{\mathsf{T}}U = \Sigma V^{\mathsf{T}}U =: \Sigma Q \tag{3.9}$$

[2]The parameters B, $\mathbf{b}$, R, and C are also affected by the transformation, but it is not relevant here.

for some orthogonal matrix Q. This follows from the closure of the orthogonal group under multiplication, which is easily verified. This can be performed without loss of generality; application of $G = U$ corresponds merely to a change of basis; the effect on the other parameters in the LDS (Section 2.2.3.2) can be safely ignored. It thus suffices to parameterise A by a diagonal matrix with entries in $[-1, 1]$ and only a single orthogonal matrix Q.

The parameterisation $A = \Sigma Q$ results in only $n_x(n_x + 1)/2$ degrees of freedom; less than the n_x^2 of the original A in Equation (3.5). This is in fact a desirable result: the standard LDS is highly overparameterised (see appendix A.2.1). Reducing the parameter count helps inference and parameter identification, while retaining straight-forward optimisation. This is not merely a *mathematical* convenience. Preliminary experiments parameterising $A = U\Sigma V^\mathsf{T}$ (i.e. with two orthogonal matrices) demonstrated slower and at times highly unstable optimisation.

Parameterisation of B. Consider changing the basis of the system equation by $G = \kappa^{-1} I$ (again via eqs. 2.33, 2.34). Due to the similarity transform applied to A, the overall scale of the latent system, κ, is independent of the choice of A. This is a further degeneracy that we wish to avoid; a hierarchical model may otherwise waste statistical strength and computation on learning equivalent representations. One solution is to fix the scale of B.[3]

A straight-forward approach, which appears to work in practice, is to upper bound the magnitude of each element of B. Let $\tilde{B}$ be the relevant elements of $\boldsymbol{\theta}$, arranged in a conformable $n_x \times n_u$ matrix. We might, for example, choose the transformation $B = \tanh(\tilde{B})$ where tanh acts element-wise. This does not entirely remove the scale degeneracy, but we have found it to work well in practice. A direct fixed-norm approach appears more difficult to optimise. If a sparse B is desired, one may wish to use an over-parameterisation, using two matrices $\tilde{B}_1, \tilde{B}_2$, with $B = \sigma(\tilde{B}_1) \circ \tanh(\tilde{B}_2)$, where $\circ$ is element-wise multiplication, and σ a logistic sigmoid. This is motivated by the fact that the gradient of tanh is greatest at 0, and hence the former parameterisation is unlikely to result in a sparse representation.

Parameterisation of R, S. The covariance matrices R, S must be in the positive definite cone. Where a diagonal covariance will suffice, any parameterisation for enforcing positivity can be used, such as exponentiation, squaring or softplus. A number of parameterisations are available for full covariance matrices (see Pinheiro and Bates, 1996). A simple choice is to use a product of 'square root' matrices, $R = LL^\mathsf{T}$, e.g. where L is a lower triangular Cholseky factor. As before, it is useful to enforce uniqueness, which can be done by ensuring the diagonal is positive.

Further implementation details can be found in appendix A.2.2.

[3] One may instead fix the scale of C or Q, although the latter may require more work to achieve.

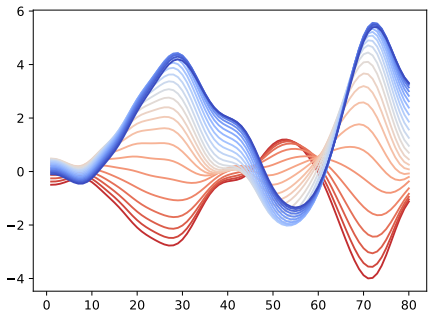


Figure 3-2: Interpolating between two sequences (red/blue) along the sequence manifold learned by an MTDS. Data are for illustrative purposes only.

3.2.2 Multi-task recurrent neural network (MT-RNN)

The dynamics of a basic multi-task RNN (MT-RNN) are described by:

$$\mathbf{x}_{t+1} = \tanh\left(A_{\mathbf{z}}\mathbf{x}_t + B_{\mathbf{z}}\mathbf{u}_t + \mathbf{b}_{\mathbf{z}}\right), \tag{3.10}$$

$$\mathbf{y}_t = C_{\mathbf{z}}\mathbf{x}_t + D_{\mathbf{z}}\mathbf{u}_t + \mathbf{d}_{\mathbf{z}} + \boldsymbol{\epsilon}_t, \tag{3.11}$$

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, S_{\mathbf{z}})$. We use an affine emission model for simplicity (extensions are straight-forward), resulting in $\boldsymbol{\theta}_{\mathbf{z}} = \{A_{\mathbf{z}}, B_{\mathbf{z}}, \mathbf{b}_{\mathbf{z}}, C_{\mathbf{z}}, D_{\mathbf{z}}, \mathbf{d}_{\mathbf{z}}, S_{\mathbf{z}}\} = \mathbf{h}_{\phi}(\mathbf{z})$. Due to the nonlinearity of an RNN, enforcing stability of $A_{\mathbf{z}}$ is not required (see e.g. Miller and Hardt, 2019, §4.4) and empirically, learning appears to be more straight-forward than an MT-LDS. Where long-term dependencies are present, bounding the spectral radius may be important (see e.g. Pascanu et al., 2013), in which case use of orthogonal matrices, as in the MT-LDS, may be appropriate. (cf. Helfrich et al., 2018).

RNNs are observed to utilise different regions of the state for different dynamic regimes, depending on the context. Sussillo and Barak (2013) demonstrate an RNN trained on a simple 3-bit memory task learns 8 fixed points corresponding to each of the 2^3 possible memory states, with saddle points functioning as junctions, directing a state x_t towards the appropriate fixed point. Changing the offset term $\mathbf{b}$ can then result in bifurcations in behaviour, resulting in a 'jump' to a different fixed point. Hence smooth variation of the bias parameter may result in 'jumps' between the dynamic regimes. If we wish to ensure a smooth interpolation between dynamics, it can be useful to make the bias constant wrt. $\mathbf{z}$ (see also Section 5.1.2.1).

3.3 Some uses of the MTDS

A trained MTDS model can be used in many ways depending on the context. While motivated by providing customisation to individual sequences, the hierarchical construction confers other benefits too. Chapters 4-6 provide detailed investigations into some of these aspects, but we collect a number of possible uses here for motivation and reference.

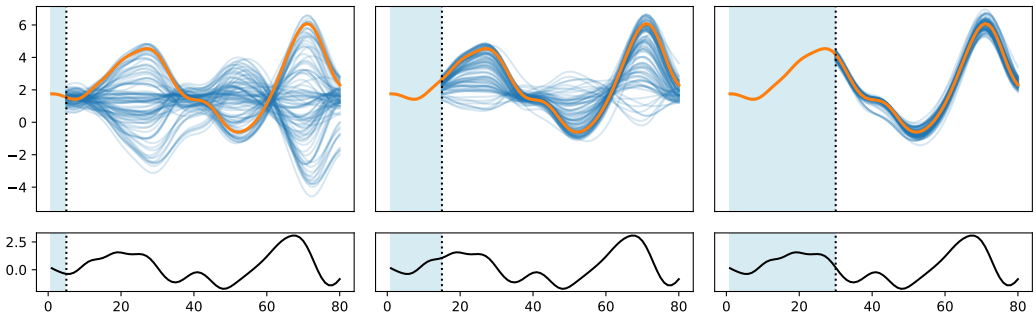


Figure 3-3: Improving predictive accuracy as more data are seen. Top panel shows observation space, with samples from the predictive posterior (shown in blue) and ground truth (shown in orange); bottom panel shows the input. Observed data are marked by the shaded area. Data are for illustrative purposes only.

Customised dynamical system modelling This was the original motivation for the MTDS. We may learn individualised sequence parameters for each sequence in the training set, even when the sequences are short. This allows an improved model fit and predictive accuracy even for AR or simple state space models (which may be important for interpretability, especially for use in scientific contexts).

Improved data efficiency When applied to a flexible time series model on limited data, the MTDS can often result in improved performance. Conditioned on an input sequence, the MTDS learns a low dimensional manifold embedded in sequence space. See Figure 3-2 for an example of travelling along the manifold between two sequences. Optimising the marginal likelihood encourages such interpolations to lie close to the training set, and hence provides an important regularisation effect, favouring a simpler hypothesis class. Training individual models via MLE will generally not favour simple hypotheses, and the usual regularisation strategies (such as penalising the L2 norm of the parameters) may reduce performance generally, and capacity to learn long-term relationships in particular. As well as calibrating the regularisation more carefully, the MTDS also supports the use of Bayesian inference over $\mathbf{z}$ at test time to reduce the effect of overfitting.[4]

Unsupervised adaptation to novel tasks When predictions are required for a never-before-seen task, we can infer its parameters online. The prediction defaults to using the prior $p(\mathbf{z})$ where no data are available, but we can update the model whenever new data become available. See Figure 3-3 for an illustration of how this can work. Suppose we

[4]Bayesian inference over the parameters of any time series model is always possible in principle, but a sensible prior may not be available, and it can be highly impractical for complex models. See Section 4.3.2 for a failure case in the context of LDS models.

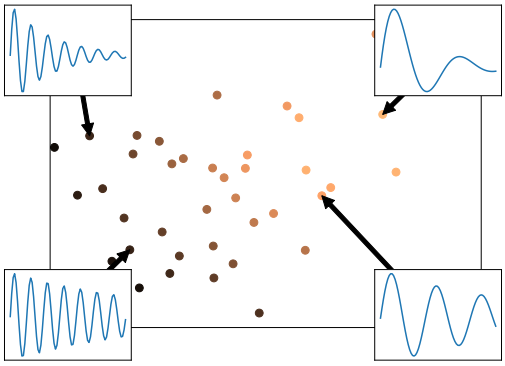


Figure 3-4: Visualisation of latent embedding of damped harmonic oscillation data using MTDS, coloured by the true frequency.

have seen observations $\mathbf{y}_{1:t}$ up to time t, and a control sequence $\mathbf{u}_{1:T}$. Then:

$$p(\mathbf{y}_{t+1:T} \,|\, \mathbf{u}_{1:T}, \mathbf{y}_{1:t}) = \int p(\mathbf{y}_{t+1:T} \,|\, \mathbf{u}_{1:T}, \boldsymbol{\theta})\, p(\boldsymbol{\theta} \,|\, \mathbf{y}_{1:t}, \mathbf{u}_{1:t})\, \mathrm{d}\boldsymbol{\theta} \tag{3.12}$$

$$= \int p(\mathbf{y}_{t+1:T} \,|\, \mathbf{u}_{1:T}, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z}))\, p(\mathbf{z} \,|\, \mathbf{y}_{1:t}, \mathbf{u}_{1:t})\, \mathrm{d}\mathbf{z} \tag{3.13}$$

$$\approx \frac{1}{M} \sum_{m=1}^{M} p(\mathbf{y}_{t+1:T} \,|\, \mathbf{u}_{1:T}, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z}_m)), \tag{3.14}$$

for $\mathbf{z}_m \sim p(\mathbf{z} \,|\, \mathbf{y}_{1:t}, \mathbf{u}_{1:t})$. The approximated predictive posterior (Equation 3.14) should result in improved predictions as t increases. If the MTDS model has learned the appropriate degrees of freedom, and the time series is stationary, this procedure should concentrate on the true model for large t. To this end, we demonstrate in Section 4.2.2 that with sufficient data, the correct degrees of freedom can indeed be learned by the MTDS. Even if the MTDS does not learn all the required degrees of freedom, the inductive bias for $\boldsymbol{\theta}$ will often result in much better performance on novel sequences than a *tabula rasa* model due to data efficiency, even though the latter might be consistent. See Section 4.2.2 for an example. As always, we must keep in mind the bias-variance trade-off.

Improved predictive uncertainty The integral $\int p(\mathbf{y}_{1:T} \,|\, \mathbf{u}_{1:T}, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z}))\, p(\mathbf{z})\, \mathrm{d}\mathbf{z}$ induces a long term correlation structure across $\mathbf{y}_{1:T}$ (see e.g. Figure 3-3). A simple pooled state space model (SSM) must account for any such correlation within localised noise variables, and hence may achieve a lower likelihood than an MTDS. Since the MTDS thereby reduces the amount of variance explained by the state, the variance of the predictive posterior will be reduced compared to a standard SSM as $p(\mathbf{z} \,|\, \mathbf{y}_{1:t}, \mathbf{u}_{1:t})$ becomes more concentrated. In some circumstances, the uncertainty from standard SSM predictions can quickly become vacuous, and unlike the MTDS, observing more data will not reduce this substantially, if at all.

Anomaly detection Substantial deviations from expected trajectories of a SSM may be tolerated due to larger system or emission variances learned by a pooled approach. By

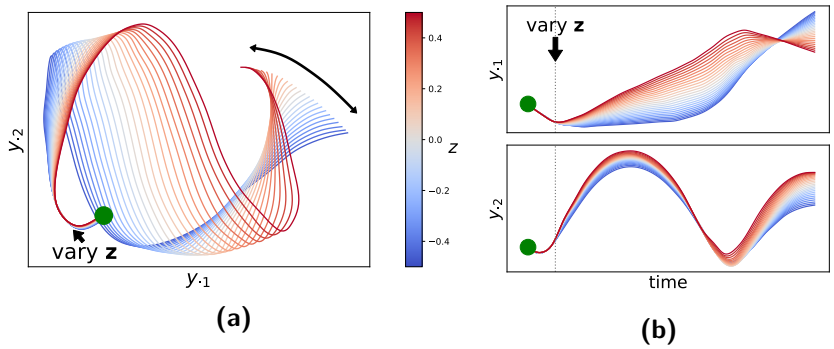


Figure 3-5: (a) Adjusting the prediction of a double pendulum bob by use of latent $\mathbf{z}$. Prediction begins at green dot; a variety of predictions from different $\mathbf{z}$ (changed after 'vary $\mathbf{z}$') are shown in different colours. (b) Co-ordinate-wise view of the previous panel over time.

learning a customised model, anomalies may be identified faster or more reliably. The MTDS does not generally admit a closed form for this likelihood, but a relatively efficient estimate can be made using an importance sampling method (we suppress inputs for simplicity):

$$p(\mathbf{y}_{1:T}) = \int p(\mathbf{y}_{1:T}, \mathbf{z})\,\mathrm{d}\mathbf{z} = \int \frac{p(\mathbf{z})}{q(\mathbf{z}\,|\,\mathbf{y}_{1:T})}\, p(\mathbf{y}_{1:T}\,|\,\mathbf{z})\, q(\mathbf{z}\,|\,\mathbf{y}_{1:T})\,\mathrm{d}\mathbf{z} \tag{3.15}$$

$$\approx \frac{1}{M}\sum_{m=1}^{M} \frac{w_m}{\sum_{j=1}^{M} w_j}\, p(\mathbf{y}_{1:T}\,|\,\mathbf{z}_m) \tag{3.16}$$

where $q(\mathbf{z}\,|\,\mathbf{y}_{1:T})$ is the (approximate) posterior, $\mathbf{z}_m \sim q(\mathbf{z}\,|\,\mathbf{y}_{1:T})$ and $w_m \propto \frac{p(\mathbf{z}_m)}{p(\mathbf{y}_{1:T}, \mathbf{z}_m)}$ are the (un-normalised) importance weights.

Visualisation / dimensionality reduction Each sequence in a dataset is represented by its task code $\mathbf{z} \in \mathbb{R}^k$, which can be useful for visualisation, especially if $k \in \{2, 3\}$. Note that this visualisation does not require the sequences be of the same length, unlike many alternatives. This can be a useful way of summarising the variability in the dataset, discovering structure within the dataset and relationships between the sequences. See Figure 3-4 for an example for sequences generated by a damped harmonic oscillator (see Chapter 4). Here, a trained MTDS has successfully learned that frequency and decay rate are the major axes of inter-sequence variation for this dataset.

Controllable predictions An MTDS model may be considered an (infinite) ensemble of time series models which are consistent with the training set, and the choice of model is controlled by the latent $\mathbf{z}$. This in turn can be used to control the trajectory of the predictions. Further, since the model learns the relevant degrees of freedom in parameter space, implausible predictions can largely be avoided.

In the case that predictions are highly uncertain (the posterior over $\mathbf{z}$ has high variance), it

may be helpful to adjust the latent code to visualise different scenarios and select a more plausible point forecast than the mean, e.g. via use of prior knowledge. Alternatively, models used for artistic purposes may be adapted freely by an artist. For example, Figure 3-5 shows the trajectory of a bob of a double pendulum. Unlike a standard sequence model, the user can control this trajectory – the figure shows a variety of available trajectories starting from the point marked 'vary $\mathbf{z}$'. The learned manifold can further be useful for *morphing* such predictions over time via a continuously and continually changing value of $\mathbf{z}$.

3.4 Model learning

An MTDS can be learned according to a variety of objectives; our focus in this thesis will be the marginal likelihood, introduced in Section 3.4.1. Since this is intractable, Sections 3.4.2 and 3.4.3 provide two options for approximating the optimisation objective.

3.4.1 Objective

The parameters $\boldsymbol{\phi}$ of an MTDS can be learned from a dataset $\mathcal{D} := \{Y^{(i)}, U^{(i)}\}_{i=1}^{N}$, defining $Y := \mathbf{y}_{1:T}$, $U := \mathbf{u}_{1:T}$ to reduce the notational burden. We can learn the parameters via maximum marginal likelihood:

$$\boldsymbol{\phi}^* \;=\; \arg\max_{\boldsymbol{\phi}} \sum_{i=1}^{N} \log p(Y^{(i)} \,|\, U^{(i)}, \boldsymbol{\phi}), \tag{3.17}$$

where

$$\log p(Y \,|\, U, \boldsymbol{\phi}) \;=\; \log \int_{\mathcal{Z}} p(Y \,|\, U, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z}))\, p(\mathbf{z})\, \mathrm{d}\mathbf{z}. \tag{3.18}$$

The first term in this integrand,

$$p(Y \,|\, U, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z})) \;=\; \int_{\mathcal{X}^T} p(Y \,|\, X, U, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z}))\, p(X|U, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z}))\, \mathrm{d}X, \tag{3.19}$$

is generally intractable for stochastic dynamics (with notable exceptions of discrete and linear-Gaussian models). A common approach is to use a variational *evidence lower bound* (ELBO) see e.g. Goyal et al. (2017); Miladinović et al. (2019) or a *Monte Carlo objective* (MCO) e.g. Maddison et al. (2017); Le et al. (2018); Naesseth et al. (2018). We assume that a low variance estimate is possible using existing techniques, and refer the reader to the references for further details. More simply, the use of deterministic state models (as in this thesis, see discussion in Section 3.6.3) avoids the integral in Equation (3.19).

We then turn to the integral in Equation (3.18). Generally, this also cannot be computed in closed form, and must be approximated. In the following sections we describe two

practical approaches. Firstly a variational approach optimising the ELBO, which benefits from recent work and the widespread usage of VAE-like architectures (Section 2.1.3). Secondly, we describe a MCO approach which we believe to be somewhat novel; while it is generally slower, it can form an arbitrarily tight bound and appears to be better at avoiding poor local minima. We note finally that other approaches could be used, such as a MAP approximation of Equation (3.18), or a Monte Carlo EM approach (McLachlan and Krishnan, 2007, §6), using e.g. HMC for the posterior approximation at each iteration.

3.4.2 Variational approach

We may bound $\log p(Y \,|\, U, \boldsymbol{\phi})$ for an individual sequence using the ELBO (derived in Section 2.1.1). Note that we may be using a stochastic estimate of the ELBO (where only a stochastic estimate of $\log p(Y \,|\, U, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z}))$ is available). The ELBO is:

$$\mathcal{L}(Y, U; \boldsymbol{\phi}, \boldsymbol{\lambda}) \;=\; \mathbb{E}_{q_{\boldsymbol{\lambda}}(\mathbf{z} \,|\, Y, U)} \Big[\log p(Y \,|\, U, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z})) \Big] - \mathrm{KL} \Big(q_{\boldsymbol{\lambda}}(\mathbf{z} \,|\, Y, U) \;\Big\|\; p(\mathbf{z}) \Big), \tag{3.20}$$

where KL is the Kullback-Leibler divergence and $q_{\boldsymbol{\lambda}}(\mathbf{z} \,|\, Y, U)$ is an approximate posterior for $\mathbf{z}$. The lower bound $\sum_{i=1}^{N} \mathcal{L}(Y^{(i)}, U^{(i)}; \boldsymbol{\phi}, \boldsymbol{\lambda})$ may then be optimised via reparameterisation (Kingma and Welling, 2014; Rezende et al., 2014), with minibatches of size $N_{\text{batch}} < N$. We will generally choose $q_{\boldsymbol{\lambda}}(\mathbf{z} \,|\, Y, U) = \mathcal{N}\Big(\boldsymbol{\mu}_{\boldsymbol{\lambda}}(Y, U),\, \mathbf{s}_{\boldsymbol{\lambda}}(Y, U)\Big)$, where $\boldsymbol{\mu}_{\boldsymbol{\lambda}}$, $\mathbf{s}_{\boldsymbol{\lambda}}$ are inference networks (as e.g. Fabius and van Amersfoort, 2015) or learned directly as parameters. These variational parameters may of direct interest (e.g. for visualisation), but may alternatively be an auxiliary artifact to be discarded after optimisation. In all our experiments, we chose the $\mathbf{s}_{\boldsymbol{\lambda}}(Y, U)$ to be diagonal matrices. If $p(Y|U, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z}))$ is a powerful base model such as an RNN, the ELBO is well-known to exhibit latent collapse: a higher value of the ELBO can be achieved if the model is able to avoid using the latent $\mathbf{z}$ (e.g. Chen et al., 2017). In order to avoid this, we used a form of KL annealing (Bowman et al., 2016).

3.4.3 Monte Carlo objective

Monte Carlo objectives (MCOs, Mnih and Rezende, 2016) construct a lower bound for marginal likelihoods via a transformation of an appropriate Monte Carlo estimator. Our MCO approach considers the logarithmic transformation of the approximation:

$$p(Y) \;=\; \frac{1}{M} \sum_{m=1}^{M} \mathbb{E}_{p(\mathbf{z}_m)} \Big[p(Y \,|\, \mathbf{z}_m) \Big] \tag{3.21}$$

$$\approx\; \frac{1}{M} \sum_{m=1}^{M} p(Y \,|\, \mathbf{z}_m) \qquad \text{for} \;\; \mathbf{z}_m \sim p(\mathbf{z}), \tag{3.22}$$

$m = 1, \ldots, M$, which uses M replicas $\{\mathbf{z}_m\}_{m=1}^M$. Then:

$$\log p(Y) = \log \mathbb{E}_{p(\mathbf{z}_{1:M})}\left[\frac{1}{M}\sum_{m=1}^{M} p(Y \mid \mathbf{z}_m)\right] \tag{3.23}$$

$$\geq \mathbb{E}_{p(\mathbf{z}_{1:M})}\left[\log \frac{1}{M}\sum_{m=1}^{M} p(Y \mid \mathbf{z}_m)\right] =: \mathcal{L}_{\text{MCO}} \tag{3.24}$$

where $p(\mathbf{z}_{1:M}) := p(\mathbf{z}_1)\ldots p(\mathbf{z}_M)$. The tightness of the bound can be increased by increasing the number of replicas M (Burda et al., 2016), and an unbiased estimate of $\mathcal{L}_{\text{MCO}}$ can be made by sampling from $p(\mathbf{z}_{1:M})$.

Assuming $p(\mathbf{z})$ is parameter-free (perhaps via re-parameterisation), we can easily calculate the gradient (if not, see Mnih and Rezende, 2016). By exchanging integration and differentiation, the gradient is:

$$\nabla_\phi \mathcal{L}_{\text{MCO}} = \int \cdots \int_{\mathcal{Z}} \nabla_\phi \log \left[\frac{1}{M}\sum_{m=1}^{M} p(Y \mid \mathbf{z}_m)\right] p(\mathbf{z}_{1:M})\, \mathrm{d}\mathbf{z}_{1:M} \tag{3.25}$$

$$= \mathbb{E}_{p(\mathbf{z}_{1:M})}\left[\frac{\nabla_\phi \sum_{m=1}^{M} p(Y \mid \mathbf{z}_m)}{\sum_{m=1}^{M} p(Y \mid \mathbf{z}_m)}\right] \tag{3.26}$$

$$= \mathbb{E}_{p(\mathbf{z}_{1:M})}\left[\frac{\sum_{m=1}^{M} p(Y \mid \mathbf{z}_m)\nabla_\phi \log p(Y \mid \mathbf{z}_m)}{\sum_{m'=1}^{M} p(Y \mid \mathbf{z}_{m'})}\right] \tag{3.27}$$

$$= \mathbb{E}_{p(\mathbf{z}_{1:M})}\left[\sum_{m=1}^{M} \tilde{w}_m \nabla_\phi \log p(Y \mid \mathbf{z}_m)\right], \tag{3.28}$$

with the equivalent of 'self-normalised importance weights':

$$\tilde{w}_m := \frac{p(Y \mid \mathbf{z}_m)}{\sum_{m'=1}^{M} p(Y \mid \mathbf{z}_{m'})} \tag{3.29}$$

for each of the M replicas. An unbiased estimate of the gradient in Equation (3.28) can be calculated by drawing a small number of samples from each $p(\mathbf{z}_m)$; in our experiments, a single sample for each replica worked sufficiently well.

Note that Equation (3.28) is an importance sampled version of the Fisher identity (see e.g. Douc et al., 2014, §D.2):

$$\nabla \log p(Y) = \int p(\mathbf{z} \mid Y)\ \nabla \log p(Y, \mathbf{z})\, \mathrm{d}\mathbf{z}. \tag{3.30}$$

This follows from our assumption that $p(\mathbf{z})$ is parameter free. The Fisher identity can be derived (for instance) via use of similar arguments to EM (McLachlan and Krishnan, 2007, §3.4), and was the original motivation for our approach here. The use of *self-normalised* weights as in Equation (3.29) accounts for why Equation (3.30) is the gradient of the log likelihood, and Equation (3.28) is the gradient of a lower bound.

Algorithm 2: Importance Sampled Optimisation for Uncontrolled MTDS

Result: Optimised parameter $\hat{\phi}$
Inputs: $\{Y^{(n)}\}_{i=1}^{N}, \phi, M, M_{\text{rsmp}}, \text{nepochs}, \text{optimiser}$;
for *epoch = 1:nepochs* **do**
  for *minibatch* S *in* $\{1, \ldots, N\}$ **do**
    // calculate posterior samples;
    $\mathbf{z}_m \overset{\text{Sobol}}{\sim} p(\mathbf{z}),\ m = 1, \ldots, M$;
    // calculate importance weights (Equation 3.29);
    $W \leftarrow \textit{construct\_weights}\left(\log p(Y = \cdot \mid \mathbf{h}_\phi(\cdot)), \{\mathbf{z}_m\}_{m=1}^{M}, \{Y^{(n)}\}_{i \in S}\right)$;
    // compute gradient;
    $\mathbf{g} \leftarrow \mathbf{0}$;
    for i *in* S **do**
      for _ *in* $\{1, \ldots, M_{rsmp}\}$ **do**
        $j \sim \textit{Categorical}\left(W_{(i)}\right)$;
        $\mathbf{g} \mathrel{+}= \nabla_\phi \log p(Y^{(i)} \mid \mathbf{h}_\phi(\mathbf{z}_j))$;
      end
    end
    Optimise(optimiser, ϕ, $\frac{1}{M_{\text{rsmp}}}\mathbf{g}$);
  end
end

The approximation of Equation (3.24) via Monte Carlo might be expected to suffer from high variance due to the fact that the prior is usually a poor proposal for the posterior.[5] Nevertheless, it should not be a poor proposal for the *aggregate* posterior, i.e. $\frac{1}{N}\sum_{i=1}^{N} p(\mathbf{z} \mid Y^{(i)})$. One should expect the divergence between the prior and aggregate posterior (i.e. $\text{KL}\left(\frac{1}{N}\sum_{i=1}^{N} p(\mathbf{z} \mid Y^{(i)}) \,\middle\|\, p(\mathbf{z})\right)$) to be small after optimising the marginal likelihood, provided the model is well-matched to the data. Further details and related observations can be found e.g. in Hoffman and Johnson (2016); Seth et al. (2019); Tomczak and Welling (2018).

This suggests an idea in the case that the number of distinct tasks is relatively small (perhaps for $k \leq 100$), and when the likelihood does not contain any exogenous inputs (i.e. $U = \varnothing$). In this case, we can sample M latent variables across a batch of N_b tasks, and from the above argument, we can expect the prior to be fairly well-matched to the aggregate posterior $\frac{1}{N_b}\sum_{i=1}^{N_b} p(\mathbf{z} \mid Y^{(i)})$. The model need only be run forward M times, since the corresponding $\hat{Y}_m$ contain no task-specific inputs $\{U^{(i)}\}_{i=1}^{N_b}$ by assumption. The likelihood *for all tasks* $(Y^{(i)},\ i = 1, \ldots, N_b)$ can therefore be calculated inexpensively from the $\{Y_m\}_{m=1}^{M}$, and the importance-weighted samples can thus result in a efficient and relatively low-variance estimate of the gradient. We can further take advantage of low-discrepancy random variates such as Sobol sequences (e.g. Lemieux, 2009) to reduce the variance.

[5] Using the prior as a proposal for the posterior has nonetheless seen successful use in Tang and Salakhutdinov (2013) and implicitly in Dauphin and Grangier (2016).

Algorithm 2 provides our implementation of these ideas. Note that when calculating the gradient, we resample a small number $M_{\text{rsmp}} \leq 5$ of particles from the importance weights (for each task i) to reduce the cost of backpropagation (a similar resampling scheme is suggested in Burda et al., 2016). We anticipate an objection that the prior may not be a good proposal for the aggregate posterior during the earlier stages of optimisation – the above argument holds only close to convergence. However our practical experience in Chapter 4 suggests that it often converges faster than use of the variational approach of Section 3.4.2. Importance sampling from the prior may in fact serve as a useful bias earlier in the optimisation, drawing in posterior distributions which are far from the prior, and expanding those which are overly concentrated.

In the case where tasks include exogenous inputs (i.e. where each observation $Y^{(i)}$ has a different input $U^{(i)}$), running the dynamics forward from a particle $\mathbf{z}_m$ can no longer be amortised over all $\{Y^{(i)}\}_{i=1}^{N_b}$ since the prediction $\hat{Y}^{(i)}$ depends on $U^{(i)}$. In this case, the importance weighted autoencoder (IWAE) of Burda et al. (2016) may be preferred. Our observation is that Algorithm 2 converges faster than the IWAE[6] where no inputs exist, and often finds better solutions. Unlike the variational $q_{\boldsymbol{\lambda}}$, this sampling-based approach cannot get 'stuck' in a local optimum, there are no constraints on the shape of the posteriors, and there are fewer parameters to learn.

3.5 Inference

A key quantity for prediction is the posterior predictive distribution. For an unseen test sequence $\{Y', U'\}$, the posterior predictive distribution is $p(\mathbf{y}'_{t+1:T} \,|\, \mathbf{y}'_{1:t}, \mathbf{u}'_{1:T}) = \int_{\mathcal{Z}} p(\mathbf{y}'_{t+1:T} \,|\, \mathbf{u}'_{1:T}, \mathbf{z})\, p(\mathbf{z} \,|\, \mathbf{y}'_{1:t}, \mathbf{u}'_{1:t})\, \mathrm{d}\mathbf{z}$, usually estimated via Monte Carlo. We must therefore have access to the posterior over $\mathbf{z}$. When we use variational methods to learn the model, inference for training examples may be performed using the variational distributions $q_{\boldsymbol{\lambda}}$, but these may not generalise to $t > T$ or to out-of-sample sequences. Furthermore, such $q_{\boldsymbol{\lambda}}$ may not be available if using an MCO approach. We turn instead to a more general method of inference.

Inference of $\mathbf{z}$ can be performed by any number of variational or Monte Carlo approaches. However, given the sequential nature of the model, it is natural to consider exploiting the posterior at time t for calculating the posterior at time $t+1$. Bayes' rule implies an update of the following form:

$$p(\mathbf{z} \,|\, \mathbf{y}'_{1:t+1}, \mathbf{u}'_{1:t+1}) \;\propto\; p(\mathbf{y}'_{t+1} \,|\, \mathbf{y}'_{1:t}, \mathbf{u}'_{1:t+1}, \mathbf{h}_\phi(\mathbf{z}))\, p(\mathbf{z} \,|\, \mathbf{y}'_{1:t}, \mathbf{u}'_{1:t}), \tag{3.31}$$

following the conditional independence assumptions of the MTDS. This update (in principle) incorporates the information learned at time t in an optimal way. We are interested in inferential methods which can exploit this prior information efficiently. Below we discuss

[6] Which includes, as a special case, the VAE.

existing work using both Monte Carlo (MC) and variational inference, before discussing our preferred approach in Section 3.5.1 using adaptive importance sampling.

Monte Carlo inference A gold standard of inference over $\mathbf{z}$ may be the No U-Turn Sampler (NUTS) of Hoffman and Gelman (2014) (a form of Hamiltonian Monte Carlo), provided k is not too large and efficiency is not a concern. However, Equation (3.31) casts doubt on the use of Markov Chain Monte Carlo (MCMC) methods, since it is not obvious how to incorporate at time $t+1$ the samples of $\mathbf{z}$ obtained at time t. Perhaps a more relevant approach is Sequential Monte Carlo (SMC, e.g. Doucet et al., 2001) which is designed for use in a sequential context. Unfortunately, naïve use of SMC (particle filtering) will result in severe particle depletion over time. To see this, let the posterior after time t be $p(\mathbf{z} \,|\, \mathbf{y}'_{1:t}, \mathbf{u}'_{1:t}) = \frac{1}{M} \sum_{m=1}^{M} w_m \delta(\mathbf{z} - \mathbf{z}_m)$. Then the updated posterior at time $t+1$ will be:

$$p(\mathbf{z} \,|\, \mathbf{y}'_{1:t+1}, \mathbf{u}'_{1:t+1}) \;\propto\; \frac{1}{M} \sum_{m=1}^{M} w_m p(\mathbf{y}'_{t+1} \,|\, \mathbf{y}'_{1:t}, \mathbf{u}'_{t+1}, \mathbf{h}_\phi(\mathbf{z})) \delta(\mathbf{z} - \mathbf{z}_m), \tag{3.32}$$

$$\Rightarrow p(\mathbf{z} \,|\, \mathbf{y}'_{1:t+1}, \mathbf{u}'_{1:t+1}) \;=\; \frac{1}{M} \sum_{m=1}^{M} \tilde{w}_m \delta(\mathbf{z} - \mathbf{z}_m), \tag{3.33}$$

where $\tilde{w}_m = \frac{w_m p(\mathbf{y}'_{t+1} \,|\, \mathbf{u}'_{t+1}, \mathbf{h}_\phi(\mathbf{z}_m))}{\sum_{j=1}^{M} w_j p(\mathbf{y}'_{t+1} \,|\, \mathbf{u}'_{t+1}, \mathbf{h}_\phi(\mathbf{z}_j))}$, simply a *re-weighting* of existing particles. The number of particles with significant weights w_m will reduce quickly over time. But since the model is static with respect to $\mathbf{z}$ (see Chopin, 2002), there is no dynamic process to 'jitter' the $\{\mathbf{z}_m\}$ as in a typical particle filter, and hence a resampling step cannot improve diversity.

Chopin (2002) discusses two related solutions: firstly using 'rejuvenation steps' (cf. Gilks and Berzuini, 2001) which applies a Markov transition kernel to each particle. The downside to this approach is the requirement to run until convergence; and the diagnosis thereof, which can result in substantial extra computation. The second alternative given is to sample from a 'fixed' (or global) proposal distribution (accepting a move with the usual Metropolis-Hastings probability) for which convergence is more easily monitored. This introduces a further difficulty, however, of appropriately tuning the proposal distribution. Neither option appears practical as an efficient inner step for iterations of Equation (3.31).

Variational inference Variational inference (VI) considers a *parametric* approximation to the posterior $p(\mathbf{z} \,|\, \mathbf{y}_{1:t}, \mathbf{u}_{1:t})$; variational approaches may not be statistically consistent, but they can be much faster than MC methods. A well-known approach to problems with the structure of Equation (3.31) is assumed density filtering (ADF, see e.g. Opper and Winther, 1998). For each t, ADF performs the Bayesian update and the projects the posterior into a parametric family $\mathcal{Q}$. The projection is done with respect to the reverse KL Divergence, i.e. $q_{t+1} = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}\left(p(\mathbf{z} \,|\, \mathbf{y}'_{1:t+1}, \mathbf{u}'_{1:t+1}) \;\middle\|\; q\right)$. Intuitively, the projection

finds an 'outer approximation' of the true posterior, avoiding the 'mode seeking' behaviour of the usual forward KL, which is particularly problematic if it attaches to the wrong mode.

Clearly the performance of ADF depends crucially on the choice of $\mathcal{Q}$. Unfortunately, where $\mathcal{Q}$ is expressive enough to capture a good approximation, the optimisation problem will usually be challenging, typically requiring a stochastic gradient approach, resulting in an expensive inner loop. Furthermore, when the changes from q_t to q_{t+1} are relatively small, the gradient signal will be weak, resulting perhaps in misdiagnosed convergence and hence accumulation of error over increasing t. Some improvements are possible, such as re-use of stale gradients (Tomasetti et al., 2019) or standard variance reduction techniques. Nevertheless, given the possible inefficiencies of stochastic gradient approaches, compounded errors, and inaccuracies derived from $\mathcal{Q}$, ADF may be considered unreliable for general use.

Computational complexity of inference Bayesian updating as in Equation (3.31) suggests that a constant time update is possible wrt. t (at least at first glance). However, by explicitly considering the latent state:

$$p(\mathbf{z} \,|\, \mathbf{y}'_{1:t+1}, \mathbf{u}'_{1:t+1}) \;\propto\; \int p(\mathbf{y}'_{t+1} \,|\, \mathbf{x}_{t+1},\, \mathbf{u}'_{t+1}, \mathbf{h}_\phi(\mathbf{z}))\, p(\mathbf{x}_{t+1} \,|\, \mathbf{x}_t)\, p(\mathbf{z},\, \mathbf{x}_t \,|\, \mathbf{y}'_{1:t}, \mathbf{u}'_{1:t})\, \mathrm{d}\mathbf{x}_{t+1}, \tag{3.34}$$

we see that a constant time algorithm will require the inference of the *joint* posterior $p(\mathbf{z},\, \mathbf{x}_t \,|\, \mathbf{y}'_{1:t}, \mathbf{u}'_{1:t})$ which may dramatically increase the dimensionality of the inference problem. MC methods cannot obtain this constant time performance except by reweighting particles as in Equation (3.33), which should be avoided. (Use of MCMC-type methods would require the calculation of the full joint density up to time t for the acceptance probability.) In principle, variational methods can still obtain constant time updates, but in exchange for a more challenging optimisation problem, and (most likely) poorer posterior approximation. It is also challenging for deterministic state problems since the joint posterior will be highly degenerate in $(\mathbf{z}, \mathbf{x})$. In many practical applications, it may be easier and more efficient to accept the linear time complexity of each update, due to running the dynamics forward for $p(\mathbf{y}'_{t+1} \,|\, \mathbf{u}'_{1:t+1}, \mathbf{h}_\phi(\mathbf{z}))$ (and its derivative) for each $\mathbf{z}$. In practice, sequential incorporation of previous information will perform a kind of annealing (Chopin, 2002) which reduces the difficulty, and hence hopefully the runtime of inference at each stage.

3.5.1 An adaptive importance sampling approach

Having discussed an overview of the possible options, we can now introduce our proposed method: a sequential application of adaptive importance sampling (IS). This blends the advantages of both MC and VI methods by using a parametric posterior approximation q_{prop}

which is fitted via Monte Carlo methods, specifically, adaptive IS (see below). The parametric posterior generates the required diversity in samples without resorting to MCMC moves, and the use of IS accelerates convergence and avoids the compounded errors of ADF. The key quantity for IS is the proposal distribution q_{prop}: IS will not perform well unless q_{prop} is well-matched to the target distribution. Our observation is that the natural annealing properties of the filtering distributions (Equation 3.31) allow a slow and reliable adaptation of the proposal distribution. To the best of our knowledge, this is a novel approach to inference in static sequential problems such as Equation (3.31).

The target distribution (the posterior) can be multimodal and highly non-Gaussian, but in practice can usually be well approximated by a Gaussian mixture model (GMM). We have found this to be a good choice for q_{prop} in our experiments. At each time t, the proposal distribution q_{prop} is improved over N_{AdaIS} iterations using adaptive importance sampling (AdaIS), described for mixture models in Cappé et al. (2008). We briefly review the methodology for a target distribution p_*. Let the AdaIS procedure at the nth iteration fit the proposal:

$$q^n_{\text{prop}}(\mathbf{z}) \;:=\; \sum_{j=1}^{J} \alpha^n_j \mathcal{N}\left(\mathbf{z} \mid \boldsymbol{\mu}^n_j,\, \Sigma^n_j\right), \tag{3.35}$$

with $\alpha_j \in \mathbb{R}_+$ such that $\sum_{j=1}^{J} \alpha_j = 1$. For each iteration n, perform:

$$\mathbf{z}_m \;\sim\; q^{n-1}_{\text{prop}}, \qquad m = 1, \ldots, M \qquad \text{(sample)} \tag{3.36}$$

$$\tilde{w}_m \;\propto\; p_*(\mathbf{z}_m)/q^{n-1}_{\text{prop}}(\mathbf{z}_m), \quad m = 1, \ldots, M \qquad \text{(calculate weights)} \tag{3.37}$$

(where q^0_{prop} is the prior). The nth proposal distribution q^n_{prop} is then fitted to the resulting empirical distribution $\sum_{m=1}^{M} \tilde{w}_m \delta(\mathbf{z} - \mathbf{z}_m)$, estimating $\{\alpha^n_j, \boldsymbol{\mu}^n_j, \Sigma^n_j\}_{j=1}^{J}$ via weighted Expectation Maximisation (see Cappé et al., 2008, for details). We can monitor the effective sample size (ESS, see Ch. 9, Owen, 2013) every iteration to understand the quality of q_{prop}, and stop once the ESS has reached a certain threshold M_{ess}, or when $n = N_{\text{AdaIS}}$; see Algorithm 3.

In our sequential setting, we fit a sequence of proposal distributions $q_1, q_2, \ldots, q_T$, with each proposal q_t being tuned via AdaIS (using up to N_{AdaIS} adaptations), and q_{t-1} forming the prior for q_t. (We can define $q_0 := p(\boldsymbol{\theta})$ for the MTDS.) The method is thus able to make use of the previous posterior without suffering from accumulated errors, and the AdaIS updates benefit from the fast initial convergence of the EM algorithm (see e.g. Xu and Jordan, 1996). The method is a little more challenging in the case where p_* is intractable, but it can be achieved by integrating ideas from Chopin et al. (2013), for example. Typically only a small number of AdaIS iterations are required (usually $N_{\text{AdaIS}} \leq 5$ for our problems), rendering the procedure substantially faster than MCMC moves—and by avoiding stochastic gradients—substantially faster than variational approaches. Finally, we have found the use of low-discrepancy MC samples such as Sobol sequences helpful in

Algorithm 3: Filtered inference via Iterated AdaIS.

Result: Approximate posteriors $\{q_t\}_{t=1}^T$
Inputs: $\mathbf{y}_{1:T}$, $\mathbf{u}_{1:T}$, $\boldsymbol{\phi}$, M, M_{ess}, N_{AdaIS}, J;
$q_0 \leftarrow p(\mathbf{z})$;
for $t = 1 : T$ **do**
  ess $\leftarrow 0$;
  $q^0_{\text{prop}} \leftarrow q_{t-1}$;
  for $n = 1 : N_{AdaIS}$ **do**
    for m *= 1:M* **do**
      $\mathbf{z}_m \sim q^{n-1}_{\text{prop}}$;
      $w_m \leftarrow \frac{p(\mathbf{y}_{1:t} \mid \mathbf{u}_{1:t}, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z}_m))p(\mathbf{z})}{q^{n-1}_{\text{prop}}(\mathbf{z}_m)}$;
    end
    $\tilde{w}_m \leftarrow \frac{w_m}{\sum_{\ell=1}^M w_\ell}, \quad m = 1, \ldots, M$;
    $q^n_{\text{prop}} \leftarrow \text{WgtExpectationMaximisation}\left(\{\mathbf{z}_m\}_{m=1}^M, \{\tilde{w}_m\}_{m=1}^M, J;\ \text{init} = q^{n-1}_{\text{prop}}\right)$;
    ess $\leftarrow \text{EffectiveSampleSize}\left(\{\tilde{w}_m\}_{m=1}^M\right)$;
    if *ess* $> M_{ess}$ **then**
      break;
    end
  end
  $q_t \leftarrow q^n_{\text{prop}}$
end

reducing sampling variance and further speeding convergence.

Application of adaptive importance sampling in a sequential context is a novel development as far as we are aware; and it is has proved superior in our experiments (in terms of speed, accuracy and robustness) to the alternative approaches discussed above. The viability of this approach is perhaps questionable for larger values of k (we use $k \leq 10$), but it is unclear whether larger latent spaces will be commonplace. Recall that k refers to the degrees of freedom of the *model family* rather than the models themselves, which can have many orders of magnitude more parameters. For relatively small deterministic LDS models, and $k = 4$, inference has taken $\mathcal{O}(100\text{ms})$ on a standard laptop per posterior q_t, and one may choose to thin the sequence of posteriors to multiples of ω steps ($t = \omega, 2\omega, 3\omega$) rather than every step.

We observed in these experiments that posteriors are often multimodal for $t \leq 20$ and sometimes beyond (especially when using the MCO objective), motivating the GMM parameterisation (for example, see Figure A-6 in the appendix). In these experiments, the GMM appears to capture the salient characteristics of the target distribution well. The runtime is usually dominated by the data generating process, so there is little disadvantage in using a GMM over a simple Gaussian proposal.

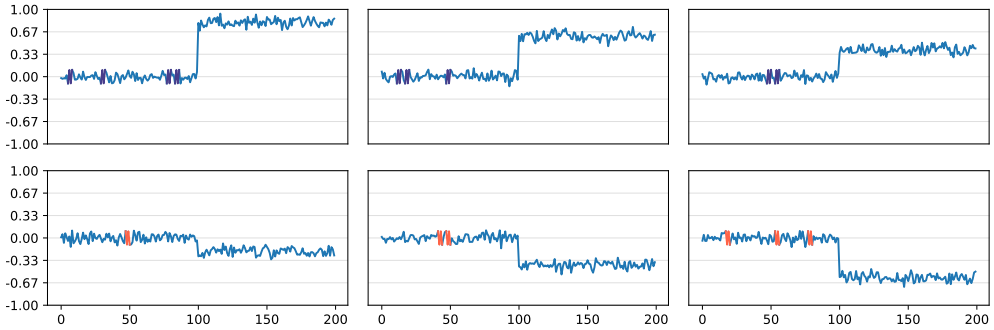


Figure 3-6: Example training sequences for which information about the long term trajectory will typically be ignored by a standard MTDS approach. The initial section of each sequence contains a code (highlighted in blue/red) which indicates the level shift expected after the halfway point (see text). Most base models used will consider this to be 'noise', and fail to adapt the latent $\mathbf{z}$ accordingly.

3.6 Some modifications used in practice

A well-known adage in machine learning is that the objective function used during training should be well tailored to a model's use at test time. The MTDS may be used for a number of downstream tasks, some of which are given in Section 3.3. In order to obtain better task-specific performance, changes in objective function and heuristics may be helpful. In this section we detail three such modifications that we have found useful in experiments: discriminative adaptation of the latent $\mathbf{z}$, improving variance estimation via regularisation, k-step objectives for long term predictions.

3.6.1 Posterior inference using a discriminative approach

We have seen in Section 3.3 that one can use the MTDS for prediction via the predictive posterior $p(\mathbf{y}_{t+1:T} \,|\, \mathbf{y}_{1:t}) = \int p(\mathbf{y}_{t+1:T} \,|\, \mathbf{z})\, p(\mathbf{z} \,|\, \mathbf{y}_{1:t})\, \mathrm{d}\mathbf{z}$ (suppressing inputs for simplicity). Predictions $\hat{\mathbf{y}}_{t+1:T}$ are tailored based on the observations so far ($\mathbf{y}_{1:t}$) via use of the generative model. However, if the model family represented by the MTDS does not contain the true model, predictions may be arbitrarily poor, even for $t \to \infty$. Worse, this may not be obvious from the fit of the training data.

A demonstration of this is provided in the data in Figure 3-6. The sequences approximately follow zero mean Gaussian white noise up until $t = 100$, but then receive an input impulse (of unit magnitude) which changes the process mean. This new level differs between sequence instances, and an MT-LDS will model the level change as entirely stochastic via the latent $\mathbf{z}$. However, the interval $t \in [1, 100]$ in fact contains a code which indicates the level change. The encoding is via short signatures highlighted in Figure 3-6: blue segments raise the level shift by 0.2 units, and red segments lower the level shift by 0.2 units. Modelling such signatures would require use of a very complex generative model, and would not substantially improve the model fit over a MT-LDS during training. But in

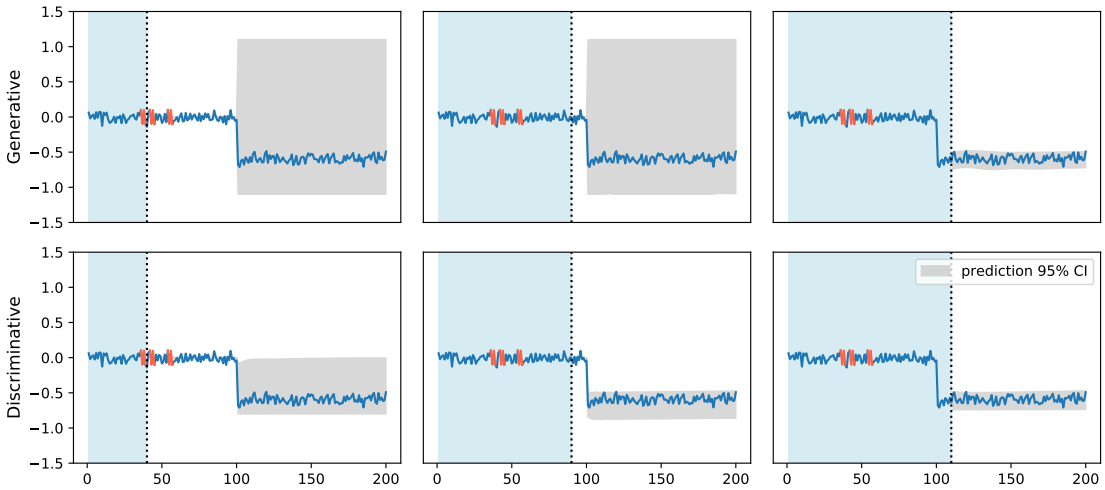


Figure 3-7: 95% predictive credible interval using generative approach (top) and discriminative approach (bottom) on an example test sequence. Observed data are marked by the blue shaded area.

ignoring these minor departures from LDS assumptions, the model will fail to exploit the information at test time when calculating $p(\mathbf{y}_{t+1:T} \,|\, \mathbf{y}_{1:t})$. The predictive posterior would then be entirely agnostic before the level shift occurs.

For the purposes of illustration, we learn an MT-LDS on this data, using the MCO approach of Section 3.4.3. The predictive posterior of the MT-LDS (using the sequential AdaIS inference approach, Section 3.5.1) is displayed in the top row of Figure 3-7 after $t = 40, 90, 120$ with 95% credible intervals. As expected, no information is gained from observing the first 90 observations, and the prediction is entirely agnostic. The appropriate level shift is only captured *after* it has occurred (top right). But using a discriminative approach we can do better. In this case, instead of learning a generative model $p(\mathbf{y}_{1:T}, \mathbf{z})$ which is inverted at test time to calculate $p(\mathbf{z} \,|\, \mathbf{y}_{1:t})$, we choose to *learn* a distribution $q_{\boldsymbol{\lambda}}(\mathbf{z} \,|\, \mathbf{y}_{1:t})$ on the training set, using a parametric model. See the bottom row of Figure 3-7, where $\mathbf{z}$ is inferred by a trained RNN.

How might such a discriminative network be learned? One can consider this to be a sequence-to-sequence (seq2seq) learning problem with a learnable encoder $q_{\boldsymbol{\lambda}}(\mathbf{z} \,|\, \mathbf{y}_{1:t})$ and fixed decoder $p(\mathbf{y}_{t+1:T} \,|\, \mathbf{y}_{1:t}, \mathbf{z})$, where the latter is learned via the methods in Section 3.4. However, a naïve implementation will generally result in a collapse of the variance over $\mathbf{z}$. As an alternative, consider the following objective, justified by the ELBO:

$$\log p(\mathbf{y}_{t+1:T} \,|\, \mathbf{y}_{1:t}) \geq \mathbb{E}_{q_{\boldsymbol{\lambda}}(\mathbf{z} \,|\, \mathbf{y}_{1:t})} \left[\log p(\mathbf{y}_{t+1:T} \,|\, \mathbf{y}_{1:t}, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z})) \right] - \mathrm{KL} \left(q_{\boldsymbol{\lambda}}(\mathbf{z} \,|\, \mathbf{y}_{1:t}) \,\middle\|\, p(\mathbf{z} \,|\, \mathbf{y}_{1:t}) \right). \tag{3.38}$$

The posterior $p(\mathbf{z} \,|\, \mathbf{y}_{1:t})$ can be obtained either from the initial MTDS optimisation, or as a separate inference problem.[7] There are two notable distinctions of Equation (3.38)

[7] The 'conditional VAE' of Sohn et al. (2015) *learns* this posterior, but I believe this to be an unsound

compared to standard variational objectives. Firstly, only the variational parameters ($\boldsymbol{\lambda}$) are optimised; secondly only the first t observations, $\mathbf{y}_{1:t}$, are used for conditioning $q_{\boldsymbol{\lambda}}$. The objective in Equation (3.38) ostensibly requires learning separate models for each time step t. One might circumvent this via implementing $q_{\boldsymbol{\lambda}}$ via a RNN and summing the criterion of Equation (3.38) over multiple t, and iterating the $q_{\boldsymbol{\lambda}}$ RNN in each case for the relevant number of steps. Other approaches include that of Ivanov et al. (2019) which supports arbitrary conditioning.

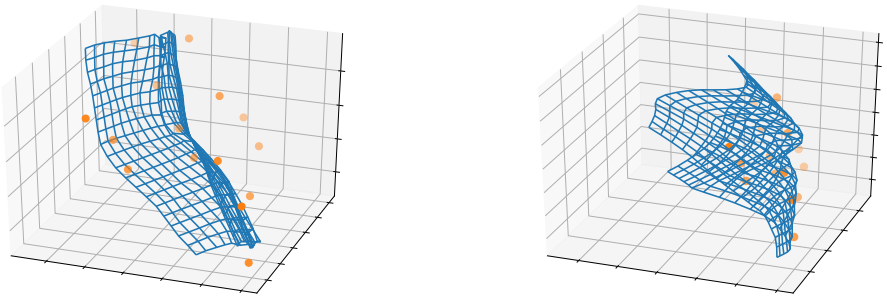
In general, generative and discriminative approaches will extract different information. Where the MTDS has learned the true model, the generative approach is optimal, but otherwise the discriminative approach has the potential to perform better. However, the discriminative approach may still perform worse if there are insufficient data to learn the conditional distribution reliably. We do not advocate for one approach over another in this thesis – the generative approach is a safer choice since it requires no learning, but possibly poorer where data are plentiful. We note in closing that this example is quite extreme, and the examples in Chapters 4 - 6 did not appear to benefit from a discriminative approach.

3.6.2 Learning the emission variance

It is relatively common to use an unweighted mean squared error (MSE) objective when learning time series models, which implicitly assumes isotropic Gaussian noise for the emission distribution. This can be reasonable if the observations are standardised prior to learning (autocorrelations can be modelled by the state when using SSMs). Learning the magnitude of the variance can then be ignored during learning and calculated post-hoc. However, for a MTDS, the concentration of the posterior will depend directly on this variance, and hence so will the latent representation. A model with small variance may obtain 'sharper' reconstructions but poorer interpolation between training sequences; a high variance model may achieve the opposite. The principled approach is therefore to learn the variance along with the model parameters.

However, learning the variance introduces its own challenges when the model can obtain a low variance fit of the observations. This may occur either via overfitting of the sequence, or in the case that the true emission variance is low. In the extreme case of noiseless observations, the learned variance may become arbitrarily small, and the reconstruction term (the first term in the ELBO, Equation 3.20) can become very large, dwarfing the impact of any prior regularisation (e.g. the second term in the ELBO, Equation 3.20). The modelling implications of this problem are serious. In this case, the model manifold can interpolate the observed sequences almost exactly, reducing the posterior covariance towards zero, and effectively removing the regularisation of the model manifold between the training sequences. An example of this phenomenon for a VAE learned on 3-D iid data is shown in Figure 3-8. (Visualising this problem in sequence space is much harder.)

approach, equivalent to introducing a latent variable without a prior, and would therefore suffer from collapsing variance. Their qualitative results support this.



(a) **(b)**

Figure 3-8: Comparison of VAE with 2-D latent space (embedding shown in blue) fitted on 3-D data (orange). (a) Model fit with correctly specified variance; (b) model fit with low emission variance. Data are for illustrative purposes only.

A possible remedy is to inflate the emission variance via use of a prior distribution. This inflates the posteriors, and allows them to cover more of the model manifold, resulting in greater regularisation. A similar observation is made in (Svénsen, 1998, §3) in the context of latent variable models for iid data.

3.6.3 Long term prediction

The final modification detailed in this section is not specific to MTDS models, but it is an important consideration for learning a sequence family. MTDS models may often be used for sequence generation or long term prediction (Section 3.3) which requires greater care in choosing the objective function. Perhaps counter-intuitively, the use of maximum (marginal) likelihood estimation (MLE) can perform much worse than a direct k-step objective, such as:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\Theta} \sum_{t=1}^{T-k} \mathbb{E}_{p_{\text{true}}(\mathbf{y}_{1:t})} \mathrm{KL}\left(p_{\text{true}}(\mathbf{y}_{t+1:t+k} \,|\, \mathbf{y}_{1:t}) \,\middle\|\, q(\mathbf{y}_{t+1:t+k} \,|\, \mathbf{y}_{1:t}; \boldsymbol{\theta})\right) \tag{3.39}$$

where p_{true} is the true distribution and $q(\mathbf{y}_{1:T}; \boldsymbol{\theta})$ is the model, parameterised by $\boldsymbol{\theta}$. An extended discussion on this point is provided in Appendix A.7. To be more precise, using an objective such as Equation (3.39) is likely to be beneficial where the true model is not contained in the model class $\{q(\mathbf{y}_{1:T}; \boldsymbol{\theta}) : \boldsymbol{\theta} \in \Theta\}$.

In practice, calculating the likelihood terms $q(\mathbf{y}_{t+1:t+k} \,|\, \mathbf{y}_{1:t}; \boldsymbol{\theta})$ in Equation (3.39) can be computationally expensive, perhaps requiring Monte Carlo approximation. This can be circumvented by using deterministic state models. This model class is not as limiting as it may first appear, as we have discussed in Sections 2.2.3.6 and 2.2.4.2. Deterministic state models are capable of *learning* the inference procedure which is hard-coded into stochastic state models.[8] An alternative may be to use locally linear state space models, such as

[8] This is especially noticeable for RNNs. Results even in the few domains where stochastic RNNs have

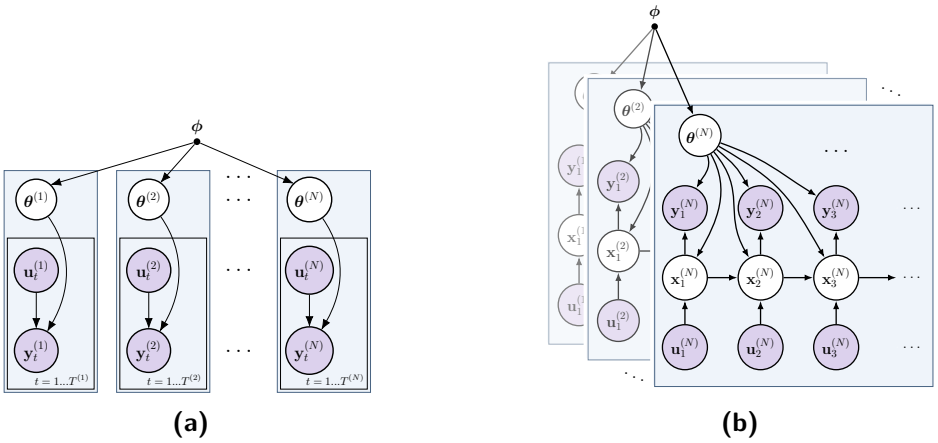


Figure 3-9: Comparison of MTL for an (a) iid model; and (b) dynamical system.

those in Fraccaro et al. (2017); Karl et al. (2017)

3.7 Related work

There are a variety of ideas related to the MTDS across the fields of machine learning and statistics. This is indicative of the number of circumstances in which such problems arise. Therefore while we hope to provide a helpful overview of this area, we can make no claim to an exhaustive summary. We note that there is relatively little which is explicitly multi-task, and none with the kind of generality we have discussed in this chapter. One of our core contributions is to extend ideas from MTL (derived from an iid context) to the time series context (see Figure 3-9). Other uses of latent variables in a RNN context are discussed in the recent review paper of Girin et al. (2020).

Supervised adaptation Probably the simplest form of MTL for time series is when side information is available about the sequences (in the form of a label or 'task descriptor' per sequence). For RNNs, this is commonly exploited in two ways: the label can be used to customise the $\mathbf{x}_0$ ('initial state customisation') or appended to the inputs for each t ('bias customisation'), see Goodfellow et al. (2016, §10.2.4). In the former approach, we may expect that the customisation will be lost for large t, whereas the latter at least ensures the customisation is not forgotten. However, this 'bias customisation' (so-called because it implicitly provides a customised bias to the dynamics) is limited in its expressiveness, and regardless, assumes that the available descriptors contain the information needed to customise the model fully. We show in Chapter 5 an example of where a much richer description is possible than the labels suggest. We can accommodate labels within the MTDS in a more expressive way by augmenting the latent variable $\mathbf{z}$.

demonstrated advantage over deterministic models (e.g. speech modelling) have recently been challenged (Lai et al., 2019).

Unsupervised bias customisation A number of dynamical models following an *unsupervised* 'bias customisation' approach have been proposed recently. Miladinović et al. (2019) and Hsu et al. (2017) propose models where the biases of an LSTM cell depend on a (hierarchical) latent variable. Yingzhen and Mandt (2018) propose a dynamical system where the latent dynamics are concatenated with a time-constant latent variable prior to the emission. In contrast, our MTDS model performs full parameter customisation, both on the dynamics and emission distributions. Note in particular that bias-customisation is a highly inflexible strategy for simpler models such as the LDS.

Modulating linear dynamical systems Various approaches modulate the parameters of linear dynamical systems for the purpose of local linear approximation. This is similar to the idea of *switching models* (see e.g. Chang and Athans, 1978; Murphy, 1998; Fox et al., 2009) which switch over time between parameters from a discrete set. Such models have a hierarchical structure, like the MTDS, but are far more limited in their customisation and base model, and are used to capture *intra*-sequence rather than *inter*-sequence variation. Luttinen et al. (2014) propose a local linear model for use in non-stationary environments, with a latent variable selecting the parameters within the convex hull of a learned set of basis matrices (sometimes called 'three-way weights'). Karl et al. (2017) extend this idea to nonlinear emission models, using inference networks for the latent variables. A different approach is taken by Rangapuram et al. (2018), who apply linear models in a MTL context, using an RNN to predict time-varying parameters. However, unlike the previous models and MTDS, this prediction is supervised, inferring the parameters entirely from the inputs U.

Adjusting transition weights in nonlinear models In the context of autoregressive or 'gated' Boltzmann machines, Memisevic and Hinton (2007) propose a transition based on the sum of transition matrices, weighted by a (binary-valued vector) latent variable. This idea is extended in Memisevic and Hinton (2010) to be a sum of *low rank* transition matrices, which substantially lowers the parameter count. Moving closer to the MTDS framework, Spieckermann et al. (2015) uses a similar idea for the transition matrix of an RNN, although the latent variable $\mathbf{z}$ is a real-valued vector and optimised as a parameter. The MTDS goes beyond these models by modulating *all* weights via *nonlinear* manifolds in parameter space. Further, we use a theoretically justified probabilistic objective, and develop its application for a more general class of models.

Clustering via time series dynamics A related idea is to cluster a collection of time series via their dynamics. This is useful in a variety of circumstances including scientific discovery, exploratory analysis and model building (we note the MTDS may also be used for these purposes). Generally such clusters must be learned jointly with the model; post-hoc analysis of learned parameters is problematic due to non-identifiability (see Section

2.2.3.2). This can be performed in a number of ways: for instance with each task associated with a single dimension of a multivariate latent process (e.g. Inoue et al., 2007), different projections of one of many independent latent processes (e.g. Chiappa and Barber, 2007), or independent dynamical systems which merely share parameters, similar to the MTDS (e.g. Lin et al., 2019). A different approach is proposed by Saad and Mansinghka (2018) who use a nonparametric model with *discrete* (non-Markov) latent dynamics, where the latent dynamics are shared within each cluster to gain statistical strength. The latent $\mathbf{z}$ used in the MTDS can be seen as a relaxation of the cluster assumption, which we apply to more general multivariate models.

Multi-task GPs (MTGPs) Multi-task GPs (Bonilla et al., 2008) are commonly used for sequence prediction. Examples include those in Osborne et al. (2008); Titsias and Lázaro-Gredilla (2011); Álvarez et al. (2012); Roberts et al. (2013). MTGPs however can only be *linear* combinations of (a small number of) latent functions (see Figure 3-10). Since the sharing of information is mediated via the latent process, an MTGP can only provide *different projections of the same underlying phenomena*. Distinct sequences (i.e. those that are not co-located in time or space) cannot obviously benefit from this approach. In contrast, an MTDS can allow entirely different realisations of the latent process, with a soft constraint of sharing dynamics. Furthermore, inputs are not easily integrated into the MTGP and predictions depend critically upon often unknown mean functions. The MT GP dynamical system of Korkinof and Demiris (2017) mitigates some of these limitations, but it retains a simple linear combination of latent dynamics with a Gaussian prior over the combination.

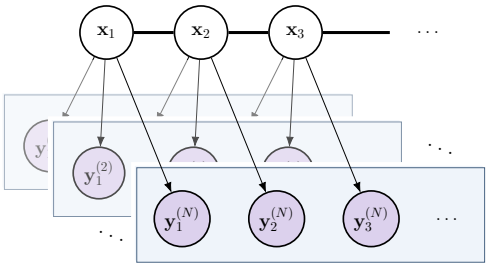


Figure 3-10: Multi-task GP following the linear model of coregionalisation (LMC). Other configurations of MTGP are possible (see Álvarez et al., 2012), but our conclusions remain the same. A thick horizontal bar represents a set of fully connected nodes.

Application to video data Controlling and customising sequence prediction has received much attention in the case of video data. As in the MTDS, these approaches can learn features that are constant (or slowly varying) within a subsequence. Denton and Birodkar (2017) propose a method to decompose time-varying and static features of a video by using an adversarial loss. Villegas et al. (2017) also decompose frames into time-varying and static features, but use architectural constraints to enforce the separation. Hsieh et al. (2018) extends these approaches by learning a parts-based scene decomposition, and applying such a latent decomposition to each part. Tulyakov et al. (2018) use a GAN architecture where $\mathbf{z}$ factorises into content and motion components. However, as before, the dynamic evolution cannot be easily customised with these methods; the

contribution of the MTDS is orthogonal and could be usefully combined with the dynamic learning in these methods.

Multiplicative RNN architectures When applied to an RNN, the MTDS architecture results in multiplicative interactions between $\mathbf{z}$ and the hidden state. Sutskever et al. (2011) propose a RNN architecture where the inputs $\mathbf{u}_t$ interact multiplicatively with the hidden state $\mathbf{x}_t$, modulating the transition matrix via tensor factorisation. The 'Hypernetworks' of Ha et al. (2016) 'predict' the parameters of networks (including RNNs) using a further network. Unlike the MTDS, the purpose is primarily to modulate the parameters over time, which can result in fewer parameters. The prediction is either conditioned on some function of the inputs $\mathbf{x}_t$ and the hidden state or some *learned* embedding $\mathbf{z}$, and neither paper provides experiments to demonstrate the effectiveness of estimating $\mathbf{z}$ in this way.

Mixed effects and hierarchical models We have already noted the connection between MTL and mixed effects models in Section 2.3.3. Mixed effects models have been extended for use in time series models for the purposes of gaining strength over parameters. However, such models are usually designed in a bespoke way, or use slow MCMC methods to perform inference. Random effects are often specified in simple ways to avoid mathematical difficulties (e.g. Tsimikas and Ledolter, 1997). Where more general application is proposed (e.g. Zhou et al., 2013), bespoke and complex learning algorithms are utilised which do not generalise easily. The Bayesian ('hierarchical') version of these models are also widely used, e.g. in healthcare (Aguilar and West, 1999), ecology (Royle and Dorazio, 2008) and retail (Chapados, 2014), although efficient inference and learning is often challenging. As discussed in Section 2.3, our philosophy is different: we are looking to exploit relationships between parameters to facilitate faster adaptation and improved prediction in an MTL sense. To this end, the MTDS learns a low dimensional manifold over the parameters; mixed-effects or hierarchical models usually learn a simple Gaussian density over (a pre-specified subset of) parameter space.

CHAPTER 4

Application to Physical Synthetic Data

Our first investigation with the MTDS aims to demonstrate that we can learn an inductive bias that is almost indistinguishable from that of the true sequence family. This will provide evidence that the methodology can also capture unknown sequence families, and shed light on any difficulties that might be encountered. For this purpose, we consider data generated by (the sum of) damped harmonic oscillators (DHOs). A DHO models an oscillating particle losing energy to forces such as heat and friction, a phenomenon encountered widely in nature. The sum of two DHOs may occur, for example, in a simple acoustical system.

This data exhibits many of the properties that motivated this project. While each sequence is relatively simple, the relationship between the sequences in the family is quite complex (for examples, see Figure 4-1). This is likely to be the case for any physical or clinical data where models need to generalise across different dynamics, such as harmonic frequencies or decay profiles. The MT-LDS method investigated in this chapter may be used in such cases provided individual sequences can be modelled via a LDS (or linear ODEs of arbitrary order). In contrast to standard mathematical modelling, the MT-LDS automatically learns the degrees of freedom, and parameter estimation can be performed efficiently online. We will investigate extensions to more complex nonlinear dynamical systems in Chapters 5 and 6.

This chapter considers two experiments. The first experiment investigates how closely the true model can be approximated via the procedures detailed in Chapter 3. Further experiments are performed to understand the sample complexity of learning such a model, and the impacts of various architectural changes. The second experiment seeks to understand how effective such a learned model is in the predictive setting: including the impact of training set size, and the comparative performance vs. existing approaches. Our conclusions are that the MT-LDS can learn a good approximation of the true sequence family from a practical number of instances, performing on par with the true physical model. By

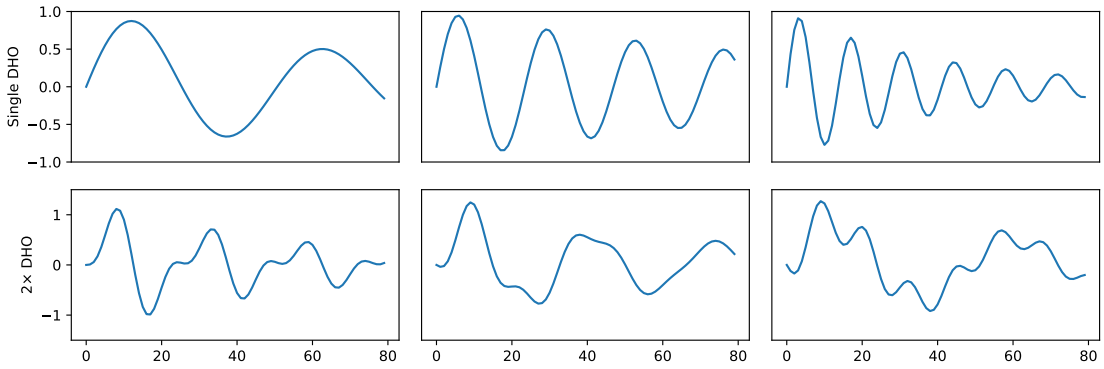


Figure 4-1: Example traces of damped harmonic oscillation with different angular frequencies and damping: (above) single DHO; (below) superposition of two DHOs.

comparison with existing sequence models, we show that this can facilitate state-of-the-art predictive performance for this dataset.

The data and basic MT-LDS model are introduced in Section 4.1. The details of the first experiment (learning the true model) follows in Section 4.2 with results given in Section 4.2.2. The details of the second experiment (predictive performance) are given in Section 4.3 with results in Section 4.3.2.

4.1 Common experimental setup

This section describes the data generating process (Sections 4.1.1 - 4.1.2) and the form of the MT-LDS to be used in all experiments (Section 4.1.3).

4.1.1 Damped harmonic oscillation

A particle undergoing damped harmonic oscillation follows an ordinary differential equation (ODE) for the distance from the origin, y, which may be written as:

$$\frac{\mathrm{d}^2 y}{\mathrm{d}t^2} + b\frac{\mathrm{d}y}{\mathrm{d}t} + cy = 0. \tag{4.1}$$

for parameters $b \geq 0$ and $c > 0$. The RHS of Equation (4.1) is zero since we are not considering a driven system. The ODE may be solved in closed form as:

$$y_t = A\rho^t \sin(\omega t + B), \tag{4.2}$$

where ρ is the decay constant ($|\rho| \leq 1$), ω is the angular increment, and A, B are the magnitude and phase of the motion. The parameters $\{\rho, \omega, A, B\}$ are functions of b, c and

the initial conditions. In our experiments we use two sets of initial conditions:

$$\text{(i)} \quad y(0) = 0, \quad \dot{y}(0) = \omega \qquad \Rightarrow A = 1, B = 0 \tag{4.3}$$

$$\text{(ii)} \quad y(0) = 0, \quad \dot{y}(0) = -\frac{1}{2}\omega \qquad \Rightarrow A = \frac{1}{2}, B = \pi. \tag{4.4}$$

These may be interpreted as striking the particles in with a force proportional to the frequency; the latter strikes the particle with a smaller force in the opposite direction to the former.

Instead of the nonlinear DHO equation (4.2), the DHO may be sampled at $t = 1, \ldots, T$ and written as a two-dimensional linear dynamical system:

$$\mathbf{x}_t = \rho \begin{bmatrix} \cos(\omega) & \sin(\omega) \\ -\sin(\omega) & \cos(\omega) \end{bmatrix} \mathbf{x}_{t-1} \tag{4.5a}$$

$$y_t = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}_t \tag{4.5b}$$

with initial state (i) $x_0 = [0 \;\; 1]^T$; (ii) $x_0 = [0 \;\; -0.5]^T$. It is easily shown that the representations of eqs. (4.2) and (4.5) are equivalent for $t = 1, \ldots, T$.[1] This state space formulation demonstrates that the sequences can be learned with an LDS with a stable transition matrix for all ω, $|\rho| \leq 1$, as per the MT-LDS assumptions.

4.1.2 Data generating process

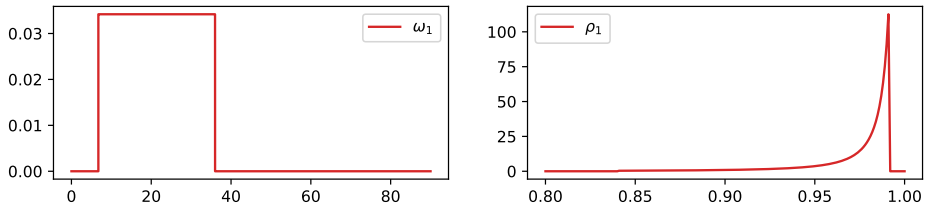
Defining a sequence family requires us to specify distributions over the parameters. We will define two sequence families of length $T = 80$ for our experiments: a simple DHO, with a prior over the parameters ω, ρ, and a double DHO family with parameters $\omega_1, \rho_1, \omega_2, \rho_2$. Some examples from each family are shown in Figure 4-1. We propose that angular frequencies be generated in the interval $\left[\frac{6.75}{180}\pi, \frac{36}{180}\pi\right]$, ensuring that 1.5 to 8 cycles are performed during the sequence. This avoids very high and low frequencies, which may be problematic under noisy measurement due to aliasing or insufficient variation. For the decay constant, the values of ρ which yield interesting time series are heavily skewed towards 1, for example, any $\rho \in [0, 0.5]$ will decay the magnitude below 0.01 by $t = 7$. A more useful representation of ρ is its *half-life*, defined as the ν which satisfies $\rho^\nu = 0.5$.[2] For our data, we propose generating a half life in the interval $[4, 80]$.

The angular frequency and half-life values are generated uniformly at random from their support.[3] For a single DHO, we use the intervals discussed above; for the double DHO data, we assume the first DHO has a lower frequency, and the second DHO has a higher frequency

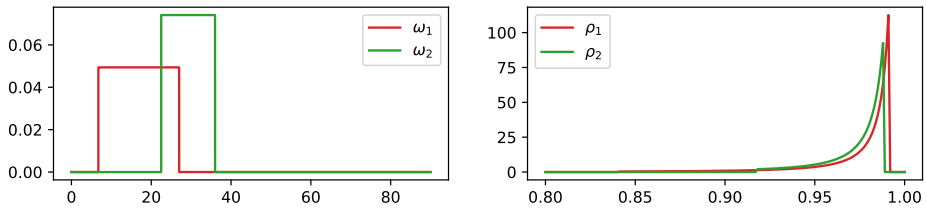
[1] For example, via use of the double angle formulae $\sin(\alpha + \beta) = \sin(\alpha)\cos(\beta) + \sin(\beta)\cos(\alpha)$ and $\cos(\alpha + \beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta)$.

[2] The decay constant can be recovered from ν via $\rho = 2^{-1/\nu}$.

[3] The resulting density over ρ for the above interval is $p(\rho) = \frac{\log(2)}{76} \frac{1}{\rho \log^2(\rho)}$, with support $[0.841, 0.991]$ to 3 decimal places. See Appendix A.4.1.



(a) Single DHO.



(b) Double DHO.

Figure 4-2: Prior densities of ω (shown in degrees) and ρ chosen for the DHO experiments.

Generating Process	ω_1	ν_1	ω_2	ν_2
Single DHO	U[6.75, 36]	U[4, 80]	-	-
Double DHO	U[6.75, 27]	U[4, 80]	U[22.5, 36]	U[8, 60]

Table 4.1: Priors for generating process for linear DHO experiments. ν is the half-life of ρ.

with a smaller range for half-life. The choices used in our data generation are shown in Table 4.1 and the resulting density functions visualised in Figure 4-2. In both cases for each training set size $N = 2^1, 2^2, \ldots, 2^7$, we generate datasets $\mathcal{D}_N = \{Y^{(1)}, \ldots Y^{(N)}\}$, with each sequence $Y^{(i)} := y^{(i)}_{1:T}$ of length $T = 80$. Each sequence i is generated using Equation (4.5) with additive white noise ($\sigma = 0.05$) on the y_t. The parameters are drawn from the priors in Table 4.1. Hence the true distribution for the single DHO is defined as:

$$\hat{y}^{\text{single}}_t(\nu, \omega) = 2^{-t/\nu} \sin(\omega t), \qquad t = 1, \ldots, T \tag{4.6a}$$

$$p_{\text{single}}(Y) = \iint \mathcal{N}\left(Y \mid \hat{y}^{\text{single}}_{1:T}(\nu, \omega), \sigma^2 I\right) p_{\text{single}}(\nu)\, p_{\text{single}}(\omega)\, \mathrm{d}\boldsymbol{\theta}_1 \tag{4.6b}$$

where $\boldsymbol{\theta}_1 := \{\nu, \omega\}$. The double DHO is similarly defined as:

$$\hat{y}^{\text{double}}_t(\nu_1, \omega_1, \nu_2, \omega_2) = 2^{-t/\nu_1} \sin(\omega_1 t) - 0.5 \times 2^{-t/\nu_2} \sin(\omega_2 t), \qquad t = 1, \ldots, T \tag{4.7a}$$

$$p_{\text{double}}(Y) = \int \cdots \int \mathcal{N}\left(Y \mid \hat{y}^{\text{double}}_{1:T}(\nu_1, \omega_1, \nu_2, \omega_2), \sigma^2 I\right) p_{\text{double}}(\nu_1) \times p_{\text{double}}(\nu_2)\, p_{\text{double}}(\omega_1)\, p_{\text{double}}(\omega_2)\, \mathrm{d}\boldsymbol{\theta}_2, \tag{4.7b}$$

where $\boldsymbol{\theta}_2 := \{\nu_1, \omega_1, \nu_2, \omega_2\}$. In both cases, the priors are defined in Table 4.1.

4.1.3 MTDS model

We will use the same MT-LDS model architecture for the MTDS in all experiments, parameterised as in Section 3.2.1 (with $B = 0, D = 0$). For a given latent code $\mathbf{z}$, the LDS model is:

$$\mathbf{x}_t = A_{\mathbf{z}}\,\mathbf{x}_{t-1} \tag{4.8a}$$

$$\mathbf{y}_t = C_{\mathbf{z}}\,\mathbf{x}_t + d_{\mathbf{z}} + \epsilon_t, \tag{4.8b}$$

$t = 1, \ldots, T$, with $\epsilon_t \sim \mathcal{N}\left(0, \sigma^2\right)$. For generality, we make $\mathbf{x}_0$ a function of $\mathbf{z}$, hence $\boldsymbol{\theta}_{\mathbf{z}} = \{A_{\mathbf{z}}, \mathbf{x}_{0\mathbf{z}}, C_{\mathbf{z}}, d_{\mathbf{z}}\}$. The transition matrix $A_{\mathbf{z}}$ uses a two-matrix Cayley parameterisation (Equation 3.9) with a scaled input ($\gamma = 0.1$) to the skew-symmetric matrix for a better conditioned optimisation problem (see appendix A.3.1, also Section 3.2.1). $\mathbf{x}_{0,\mathbf{z}}$ is parameterised in the same way as $B_{\mathbf{z}}$ in Section 3.2.1 to avoid a scale degeneracy in the latent dynamics.

For the 'single DHO' dataset, the model will use a state dimension $d = 2$; for the 'double DHO' data, we assign the state dimension $d = 4$. A standard LDS model following Equation (4.8) with the Cayley parameterisation of A has $\frac{1}{2}n_x(n_x + 5) + 2$ degrees of freedom (d.o.f.). This results in 9 d.o.f. for the single DHO model and 20 d.o.f. for the double DHO model, but by construction each dataset has only 2 and 4 d.o.f.s respectively (eqs. 4.6 - 4.7). Hence we assign latent 'task variable' dimensions of $k = 2$ and $k = 4$ respectively. The parameters $\boldsymbol{\theta}_{\mathbf{z}}$ are generated via a nonlinear factor analysis ('VAE-like') prior:

$$\boldsymbol{\theta}_{\mathbf{z}} = \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z}) \quad \mathbf{z}, \sim \mathcal{N}\left(0, I_k\right), \tag{4.9}$$

where $\mathbf{h}_{\boldsymbol{\phi}}(\cdot)$ is an MLP with one hidden layer of 64 units, and a logistic sigmoid activation function, parameterised by $\boldsymbol{\phi}$. We choose a nonlinear generative model due to its greater flexibility, but note also that the parameters ω and ν enter nonlinearly into eqs. (4.6-4.7). The learnable parameters are $\{\boldsymbol{\phi}, \sigma\}$, with the latter from the noise model in Equation (4.8b). The model is learned via maximum marginal likelihood (ML-II) using the Monte Carlo Objective MCO of Section 3.4.3. Each model is optimised using a batch size of 20 and 800 prior samples[4], increased to 3000 samples at the end of training for fine tuning. Further details can be found in appendix A.4.2.

There was evidence of some overfitting when training the models, especially for smaller datasets. We have discussed this phenomenon in Section 3.6.2 (see Figure 3-8 for an example in the iid case). Preliminary experiments suggest that a sensible emission variance – allowing smooth interpolation between examples in the training set – could often be achieved using $\sigma = 0.2$. The standard deviation of all models is set to this value during the final stages of optimisation.

[4]The prior samples are Quasi Monte Carlo samples which make use of a low-discrepancy Sobol sequence.

Table 4.2: Interpretation of Bayes Factors given by Kass and Raftery (1995)

Bayes Factor (K)	$\log K$	Interpretation
1 to 3	0 to 1	Barely worth mentioning
3 to 20	1 to 3	Positive evidence
20 to 150	3 to 5	Strong evidence
> 150	> 5	Very strong evidence

4.2 Experiment 1: Learning the true model

Our first objective is to understand how well the MT-LDS has captured the true sequence family implied by the above generating process. This cannot be easily ascertained via comparison of the parameters, since the same distribution over sequences Y permits various transformations of the latent state, and aliased versions of the cycle. Let $\boldsymbol{\phi}^*$ be the parameter of the MT-LDS which corresponds to the true generating process[5], and the parameter learned via ML-II on the training set $\mathcal{D}_N$ be $\hat{\boldsymbol{\phi}}_N$. We expect asymptotically for $N \to \infty$ that the Bayes factor,

$$K = \frac{p(Y \mid \boldsymbol{\phi}^*)}{p(Y \mid \hat{\boldsymbol{\phi}}_N)} \to 1 \tag{4.10}$$

for Y drawn from a test set, provided that one of the global optima can be found. For finite N, we can expect $K > 1$, but at what value might we say that the ratio is 'close enough' to 1?

A widely used interpretation of the log of the ratio in Equation (4.10) is given in Kass and Raftery (1995), see Table 4.2. This provides us with a heuristic for determining whether our learned model is 'close enough' to the true model; specifically for Y drawn from the true model, if $\log p(Y \mid \hat{\boldsymbol{\phi}}_N) \geq \log p(Y \mid \boldsymbol{\phi}^*) - 1$, then the difference is 'barely worth mentioning'. We will report the average $\log K$ for a large number of test examples, N_{test}, in order to estimate convergence to the entire distribution $p(Y)$. This may alternatively be interpreted as estimating the Kullback-Liebler (KL) Divergence between the true distribution p, and the model $\hat{p} = p(Y \mid \hat{\boldsymbol{\phi}}_N)$:

$$D_{KL}(p \,||\, \hat{p}) \approx \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \log \frac{p(Y^{(i)} \mid \boldsymbol{\phi}^*)}{p(Y^{(i)} \mid \hat{\boldsymbol{\phi}}_N)}, \qquad Y^{(i)} \sim p(Y^{(i)} \mid \boldsymbol{\phi}^*); \tag{4.11}$$

the above argument implies $D_{KL}(p \,||\, \hat{p}) \leq 1$ is 'close enough'.

4.2.1 Experimental setup

For each training set size $N = 2^1, 2^2, \ldots, 2^7$ we generate 10 training sets on which MT-LDS models are trained from scratch. This allows us to estimate the distribution of the

[5] If the MLP construction of the prior does not admit such a $\boldsymbol{\phi}^*$, a close approximation will be possible.

test marginal log likelihood due to sampling variance and model initialisation. We are interested in how this approaches its true value over increasing N. The chosen model has been described in Section 4.1.3, but we will also investigate the impact of alternative model configurations (size of system dimension, latent dimension and the prior $p(\boldsymbol{\theta})$).

To estimate the marginal log likelihood of the model we average over $N_{\text{test}} = 10\,000$ test examples, and the marginal log likelihood w.r.t. the true process is calculated via Monte Carlo integration over the distribution of the parameters (using 10 000 Sobol samples). Since we have used a standard deviation of $\sigma = 0.2$ for the MT-LDS models to reduce overfitting, we estimate the true marginal log likelihood using this value too.

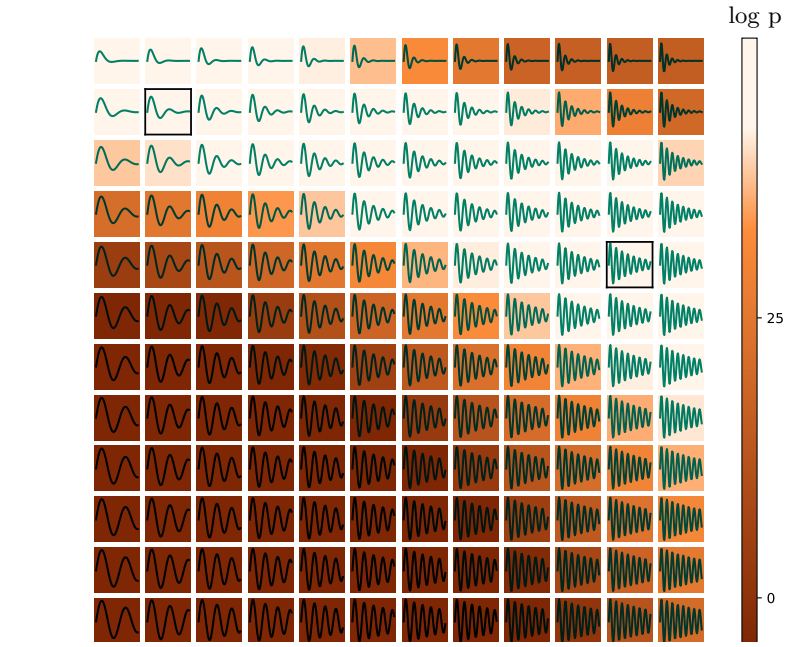
4.2.2 Results

Visualising the simple DHO model We begin by visualising the results from a p_{single} model. This permits easy visualisation since both the data and the model have 2 d.o.f.s. We may hope that the resulting intution transfers to more complex models.

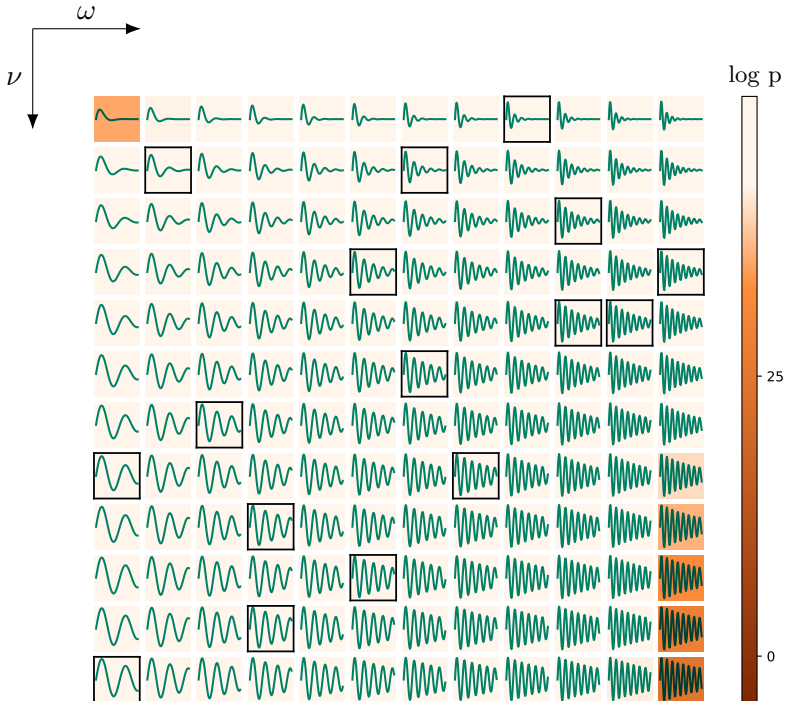
The coverage of the p_{single} distribution learned by an MT-LDS model from $N = 2$ and $N = 16$ examples is visualised in Figure 4-3. The figures show grid samples of p_{single} with increasing frequency on the horizontal axis and increasing decay rate on the vertical axis, coloured by their log probability under the learned model. Figure 4-3(a) shows that a model trained on just two sequences can still perform sensible interpolation and extrapolation, giving high probability to approx. one-quarter of the true sequence space. Crucially, this generalisation happens in *latent* space, with respect to frequency and decay rate, rather than *observation* space, as per e.g. a Multi-task GP. After $N = 16$ examples, Figure 4-3(b) shows that the model manifold grows to fill almost the entire sequence space. These figures also provide a suggestion of when convergence to the true model can be slow. The learned model is much better at interpolation than extrapolation, and hence benefits more from training examples marking out the extremes of the distribution than typical cases.

A visualisation of the learned latent space is given in Figure 4-4, with the embedding of the training set coloured by the true frequency and half-life. Remarkably, for this example, z_1 broadly captures the frequency of the sequence, and z_2 captures the decay rate. While the model does not always learn a 'disentangled' relationship, such disentangled results are typical of the representations with a higher marginal likelihood. No methods/heuristics from the disentanglement literature (e.g. Makhzani et al., 2015; Higgins et al., 2017) have been used here.

Does the learned model converge to the true model? Figure 4-5 shows the distribution of the average test marginal log likelihood for both single and double DHO datasets, for each training set size $N = 2^1, \ldots, 2^7$. The right hand panel is a 'zoomed-in' view near the true average log marginal likelihood with the interpretations of Kass and



(a) $N = 2$



(b) $N = 16$

Figure 4-3: Grid-sampled visualisation of the true distribution over sequences (horizontal axis shows increasing ω, vertical axis shows decreasing half-life ν). Samples coloured according to log probability under the learned model. The closest samples to sequences in the training set are highlighted with a black border. Learned model shown for two different sizes of training set, $N = 2$ and $N = 16$.

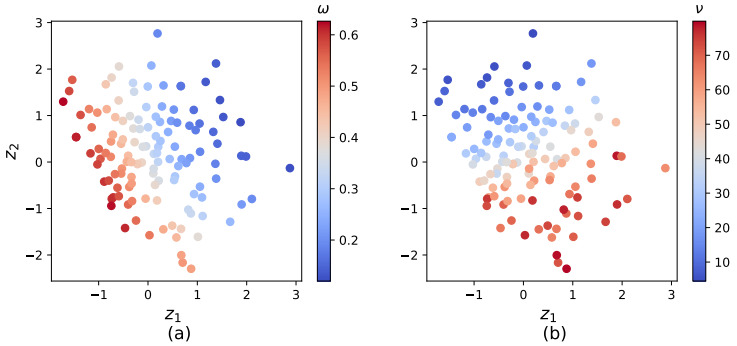
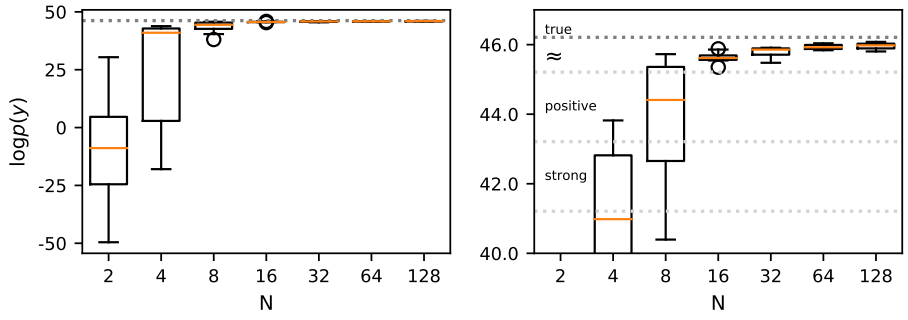
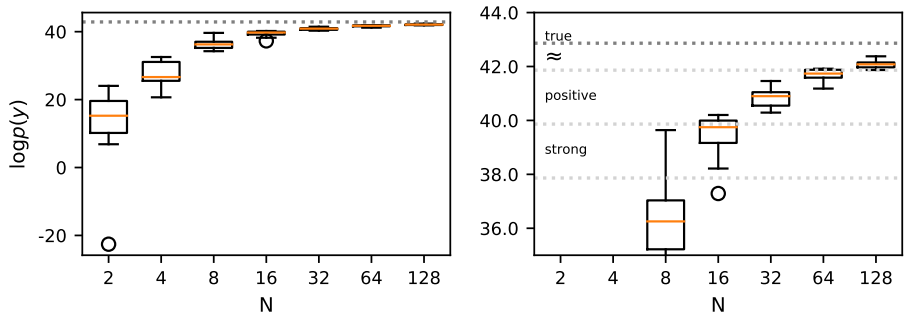


Figure 4-4: MAP Embedding of DHO training sequences coloured by (a) true frequency ω and (b) true half-life ν.



(a) Single DHO



(b) Double DHO

Figure 4-5: Average marginal likelihood of true data distribution under the DHO model learned from N examples, plotted against N (x-axis, log scale). Boxes show the distribution over 10 repetitions of each experiment, with training sets drawn randomly (low discrepancy) from the true distribution. Boxes show median, inter-quartile range (IQR) and whiskers show most extreme point within $1.5\times$ IQR above/below each box. Results for (a) the Single DHO dataset and (b) the Double DHO dataset are shown. In each case, the RHS panels are rescaled to show the upper end of the plot with Bayes Factor interpretations overlaid.

				N		
model	prior	n_x	k	4	32	128
original	MLP	4	4	27.4 (4.0)	40.9 (0.4)	42.1 (0.2)
linear prior	Linear	4	4	19.1 (5.0)	38.1 (0.4)	39.3 (0.3)
2× state dim.	MLP	8	4	15.0 (7.4)	39.7 (0.6)	42.0 (0.2)
4× state dim.	MLP	16	4	15.1 (9.1)	39.5 (0.7)	42.0 (0.1)
2× latent dim.	MLP	4	8	25.1 (6.0)	40.9 (0.4)	41.8 (0.1)
2× both dims.	MLP	8	8	21.7 (3.6)	40.8 (0.5)	42.0 (0.1)

Table 4.3: Average test marginal log likelihood of double DHO data for different MT-LDS architectures. Values are the mean (standard deviation) over 5 random training sets of size N. Average marginal loglikelihood under true model ≈ 42.9.

Raftery (1995) overlaid. As the size of the training set increases, the objective appears to converge to the true value. The single DHO is 'close enough' by $N = 16$; the double DHO appears to need $N = 128$ examples to reach a similar level. These results give us confidence that our model and training procedure can learn the true model given sufficient data, and the quantities required are by no means unfeasibly large.

4.2.3 Sensitivity to hyperparameters

The above results provide evidence that an appropriately chosen MTDS architecture can learn a good approximation to a true sequence family of interest. While this is encouraging, it is not yet clear whether we require the architecture to match the true underlying model (as in Section 4.1.3). In this section, we consider the sensitivity to the choice of $\mathbf{h}_{\phi}(\cdot)$, state dimension, and latent dimension.

The test marginal log likelihood for various hyperparameter choices for p_{double} are given in Table 4.3. We first wish to address the question: is a nonlinear/non-Gaussian prior over $\boldsymbol{\theta}$ necessary to obtain this performance? The second row of Table 4.3 shows the results from using a linear $\mathbf{h}_{\phi}$; i.e. the use of a factor-analysis-style prior. The model performs significantly worse[6] for all N. After $N = 128$ examples, the MT-LDS model with the nonlinear prior is close to the true model, but the linear-Gaussian prior variant still has 'positive'/'strong' evidence against it.

We next consider the choice of state dimension, n_x: will the model overfit as n_x increases? For small values of N, there is some evidence of overfitting (see rows 3, 4 in Table 4.3), but with sufficient data, and certainly by $N = 128$, there is very little difference in performance even with a fourfold increase in n_x. Finally turning to the choice of k, will the model overfit if the latent dimension k is increased by a factor of two? The results are not significantly different (see penultimate row), except perhaps for $N = 128$. We may expect this result since $\mathbf{z}$ is (approximately) integrated out. This equivalence in performance is largely sustained even when doubling the state dimension n_x (bottom row).

[6] A two-sample t-test with $\alpha = 0.05$ is used for all significance statements.

We conclude that at least for this problem, the performance is not especially sensitive to the model architecture, provided we use a MLP for $\mathbf{h}_{\boldsymbol{\phi}}$.

4.3 Experiment 2: Prediction

The trained MTDS models (or rather, families of models) can be used to provide personalised or 'customised' predictions at test time. The following experiments will demonstrate the benefit of this customisation compared to standard time-series approaches.

4.3.1 Experimental setup

Suppose we have a novel test sequence $y_{1:T}$ for which we have seen the first t observations. The predictive posterior (following Equation 3.14) can be approximated as:

$$p(y_{t+1:T} \mid y_{1:t}, \hat{\boldsymbol{\phi}}) \approx \frac{1}{M} \sum_{m=1}^{M} p(y_{t+1:T} \mid \mathbf{z}_m \hat{\boldsymbol{\phi}}), \qquad \mathbf{z}_m \sim p(\mathbf{z} \mid y_{1:t} \hat{\boldsymbol{\phi}}), \tag{4.12}$$

assuming we have estimated the posterior over $\mathbf{z}$ as e.g. in Section 3.5.1. The performance of this prediction for $t = 5, 10, \ldots, 80$ on novel sequences will be quantified using Root Mean Squared Error (RMSE), averaged over the $T - t$ timesteps of the predictive mean. For brevity, we only report the results for the more challenging double DHO sequence family.

We report the results for training set sizes $N = 4, 32, 128$, each averaged over 10 randomly generated training sets. The RMSE is averaged over 20 test examples drawn from the true distribution in each case. An example of this experimental setup is shown in Figure 4-6 on page 79; the entire training set (in this case $N = 8$) is shown in the top panel, and the bottom panels show predictions after $t = 15, 25, 35$ for two example test sequences.

4.3.1.1 Benchmark models

We are interested in two classes of benchmarks for these experiments. Firstly, ablation tests, comparing the MT-LDS against other LDS configurations. We will consider: (i) the 'pooled' approach, which is a popular choice for state space models (Section 3.1, i.e. using a *single model for all sequences*); (ii) a *bias-customised* model (where only $\mathbf{b}$ depends on $\mathbf{z}$); and (iii) a *single task* (STL) approach, where a LDS is inferred from scratch for each sequence. Secondly, we provide a comparison against modern RNN models. We will also consider the recent extension to bias-customised variants (Miladinović et al., 2019, see Section 3.7), which is a limited form of MT-RNN. Each benchmark model is detailed below.

Pooled LDS This is a linear dynamical system using essentially the same architecture as the MT-LDS except the parameters $\boldsymbol{\theta}$ are the same for all sequences in the training set. We allow a standard deviation σ_t that depends on t, since the magnitude of all sequences decays over time, which can affect the effective weighting of each datapoint during training. The model is:

$$\mathbf{x}_t = A\,\mathbf{x}_{t-1} + \mathbf{b} \tag{4.13a}$$

$$\mathbf{y}_t = C\,\mathbf{x}_t\, + d\, + \sigma_t \epsilon_t, \tag{4.13b}$$

$\epsilon_t \sim \mathcal{N}(0,1)$, with parameters $\boldsymbol{\theta} = \{A, \mathbf{x}_0, \mathbf{b}, C, \{\sigma_t\}_{t=1}^T\}$. This model was initialised using spectral methods and fine tuned using Adam.

MT-Bias LDS This is a restricted form of the MT-LDS model where only the dynamics bias $\mathbf{b}$ depends on $\mathbf{z}$ (cf. Miladinović et al., 2019). The model is as per eqs. (4.13a,b) but with a latent variable $\mathbf{z} \sim \mathcal{N}(0, I)$ which only affects the latent bias, i.e. $\mathbf{b}_z = \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z})$. The learned parameters are $\{A, \mathbf{x}_0, C, \{\sigma_t\}, \boldsymbol{\phi}\}$. As per the MT-LDS, we have $\mathbf{z} \in \mathbb{R}^k$ with $k = 4$. This model is trained via the MCO approach, in the same way as the MT-LDS.

Single task ST-LDS The ST benchmark uses an individual LDS model per sequence, learning the parameters online. In order to avoid overfitting, we use Bayesian inference for all parameters, $A^{(i)} \in \mathbb{R}^{4\times4}$, $\mathbf{x}_0^{(i)} \in \mathbb{R}^4$, $C^{(i)} \in \mathbb{R}^{1\times4}$, $\sigma^{(i)} \in \mathbb{R}$ for each model i. We use the same two-matrix Cayley construction for $A^{(i)}$ as the MT-LDS in order to avoid unstable transition matrices. In order to simplify inference, we remove the biases ($\mathbf{b}$, d) since these are not needed to fit the data, and A already has reduced degrees of freedom (10) due to the Cayley construction. The resulting parameter vector has 19 dimensions[7] and is given a prior of $\mathcal{N}\left(\mathbf{0}, 10^2 I\right)$, which achieves a good trade-off between an uninformative prior and stable inference.

Pooled RNN We use a standard RNN architecture (similar to Section 2.2.4.1) with affine emissions:

$$\mathbf{x}_t = \tanh\left(A\mathbf{x}_{t-1} + B\,\mathbf{u}_t\, + \mathbf{b}\right) \tag{4.14a}$$

$$\mathbf{y}_t = C\,\mathbf{x}_t\, + \mathbf{d}\, + \epsilon_t, \tag{4.14b}$$

with parameters $\{A, \mathbf{x}_0, B, \mathbf{b}, C, \mathbf{d}\}$. The size of the hidden state is chosen from $n_x \in \{32, 64, 128\}$ and L2 regularisation of all parameters with the coefficient chosen from $\{0, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ by cross validation on an additional validation set of size $N_{\text{valid}} = 20$. The model was trained using the Adam optimiser.

If the RNN has no access to inputs, it will not be able to perform much better than the Pooled LDS. In order to take advantage of the model's known ability to adapt to

[7] The parameters $A, \mathbf{x}_0, C, \sigma$ have $10, 4, 4, 1$ degrees of freedom respectively.

sequence characteristics, it will be trained via a *teacher forcing* setup (e.g. Goodfellow et al., 2016, §10.2.1) where for all t, the observation y_{t-1} is given as input for predicting y_t. At test time, long term predictions can be obtained via ancestral sampling. This is a more advantageous training regime than for the LDS models, but we wish to use RNNs in their most common form.

Pooled GRU The GRU architecture follows the specification in Section 2.2.4.5, with linear emissions as in Equation (4.14b). The dynamics parameters are $\{A^r, A^s, A^{\mathbf{x}}, \mathbf{x}_0, B^r, B^s, B^{\mathbf{x}}, \mathbf{b}^r, \mathbf{b}^s, \mathbf{b}^{\mathbf{x}}\}$ where the superscripts refer to the reset gate, state update gate and new state proposal respectively. Cross validation for the optimal hyperparameters was performed as above, and the model was trained via teacher forcing using the Adam optimiser.

MT-Bias RNN This is a restricted form of MT-RNN where only the dynamic bias $\mathbf{b}$ depends on a latent variable $\mathbf{z} \sim \mathcal{N}(0, I)$ (similar to Miladinović et al., 2019). The base model is the same as the Pooled RNN in eqs. (4.14a,b) and as above, we use $\mathbf{z} \in \mathbb{R}^k$, $k = 4$. The model is trained using the variational criterion in Section 3.4.2, using a separate posterior per training example rather than an inference network, and optimised via Adam. Unlike the pooled variant, we do not provide observations y_{t-1} as inputs at time t, as otherwise the model broadly ignored the task variable $\mathbf{z}$.

MT-Bias GRU This is a restricted form of MT-GRU, in analogy to the MT-Bias RNN. Here, all of the dynamic biases $\mathbf{b}^r, \mathbf{b}^s, \mathbf{b}^{\mathbf{x}}$ depend on a $\mathbf{z} \sim \mathcal{N}(0, I)$ with $k = 4$. The model is trained as per the MT-Bias RNN.

4.3.1.2 Inference

The MT-LDS, ST-LDS and various MT-Bias models require inference over $\{\mathbf{z}, \log \sigma\}$ for $t = 5, 10, \ldots, 80$ to calculate the predictive posterior (Equation 4.12). We chose to infer $\log \sigma$ as it is held artificially high ($\sigma = 0.2$) for learning to avoid overfitting (see above). For the MT-LDS and MT-Bias models (LDS, RNN, GRU) we used the sequential adaptive importance sampling (AdaIS) method detailed in Section 3.5.1. The approximate posterior for each t uses a Gaussian mixture model (GMM) with $J = 3$ components. An example of a posterior for the double DHO data is given in appendix A.4.5, demonstrating the need for a multi-modal approximation.

The AdaIS approach cannot be used for the ST-LDS since the posteriors tend to lie close to a low dimensional manifold, causing numerical issues. We therefore found it necessary to use Hamiltonian Monte Carlo (HMC, Neal et al. 2011). The sampler was initialised from a (local) MAP value, 1000 samples were used for burn-in/tuning, and 600 samples were used for inference, for which the autocorrelation-based sample size (Gelman et al., 2013, Ch. 11.5) typically exceeds 100 for each parameter. Since we cannot easily or efficiently perform

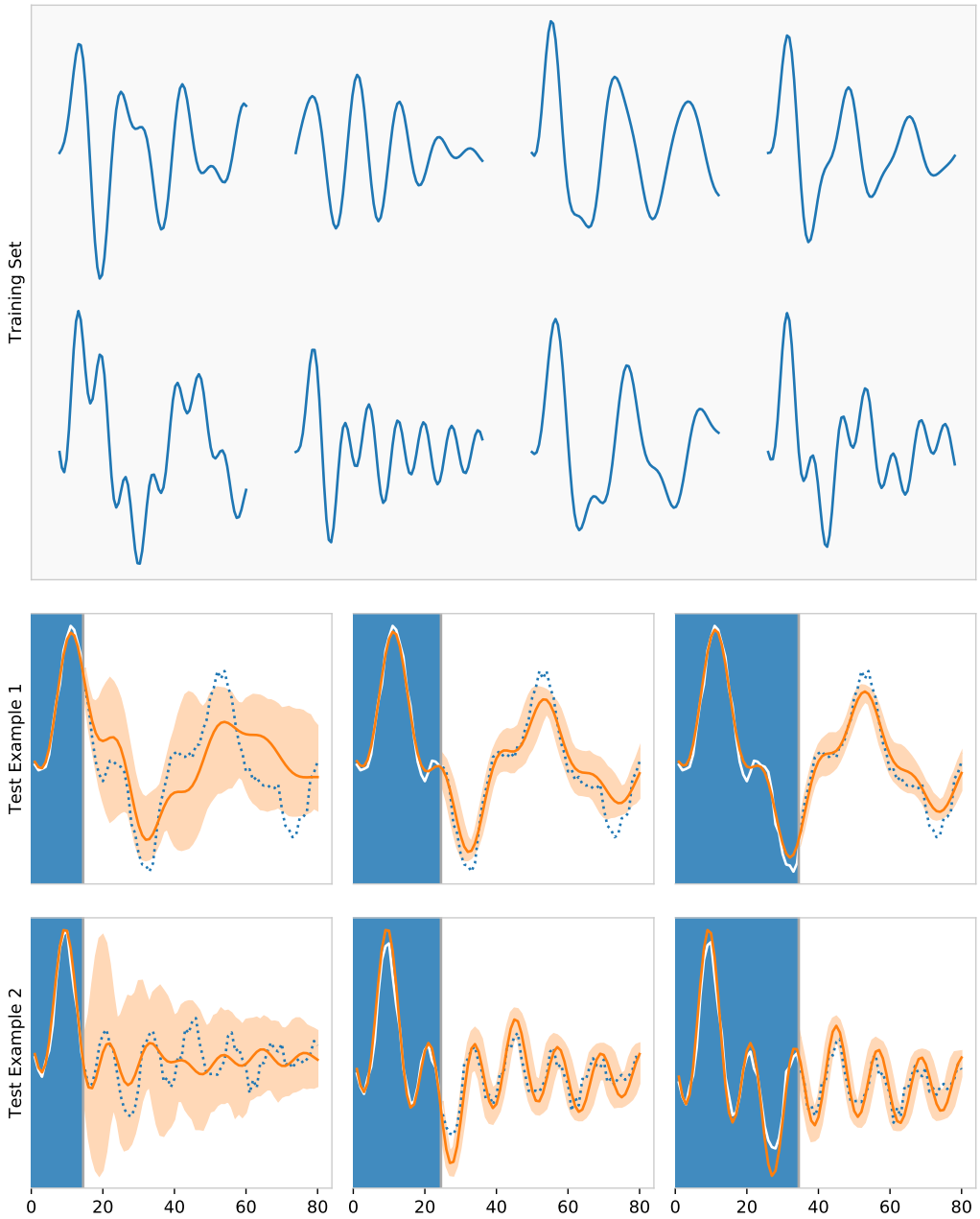


Figure 4-6: Multi-DHO Example Task. Top panel: training set for MTDS ($N = 8$). Bottom panels: example predictions (95% C.I.s) shown in orange for new sequences at $t = 15, 25, 35$. Observed values shaded blue, future sequence values (unseen) shown in dotted blue.

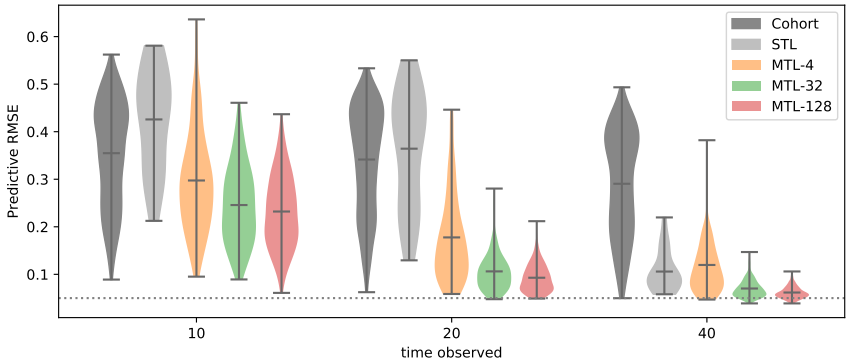


Figure 4-7: Predictive performance for $t = 10, 20, 40$ (x-axis) on the double DHO data. Violin plots show a kernel density estimate of the RMSE over sequences in the test set; horizontal bars show min, median, max scores.

sequential inference using MCMC (see Section 3.5), inference at each time t proceeds from scratch. For further details on all inference methods, see appendix A.4.3.

4.3.2 Results

An example of the prediction task is given in Figure 4-6 for $N = 8$ training examples. Two test sequences are shown at the bottom, with the blue area demonstrating the observations seen so far. MT-LDS predictions $p(y_{t+1:T} \mid y_{1:t}, \mathcal{D})$ are shown in orange with 95% pointwise credible intervals shaded. We can see that even with a relatively small number of training sequences, the model is able to learn a helpful inductive bias for predicting novel sequences. Note that neither test sequence is especially similar to any sequence in the training set, and useful personalised predictions are available even by $t = 15$. However in the upper sequence, the magnitude of the peaks/troughs falls slightly short in the predictions for $t = 15, 25, 35$, and in the bottom panel, the model is unable to account for the dip at $t \approx 27$. This demonstrates that these features are not available within the sequence family learned by the MT-LDS; it is not able to extrapolate far enough with $N = 8$ training sequences.

4.3.2.1 Standard LDS approaches

Figure 4-7 provides a violin plot comparing the RMSE of the Pooled, Multi-Task and Single-Task LDS approaches at various time intervals ($t = 10, 20, 40$) for the 20 test examples. The Pooled LDS, trained on $N = 128$ sequences, is unable to improve upon its fairly poor predictions even as more data are seen. The ST-LDS performs very poorly initially, but does improve with t. Three MT-LDS models are shown for training set sizes $N = 4, 32, 128$, with the latter performing close to optimally after $t = 40$. (The dashed line shows the optimal performance, since the test set have a noise level of $\sigma = 0.05$.) The MT-LDS performance compares favourably to the Pooled approach even for a very small

Model	Time to $t = 40$	Per prediction
Pooled	< 0.01	< 0.01
MT-LDS	1.93	0.24
ST-LDS	2776.28	347.03

Table 4.4: Inference CPU time (seconds) per approach. Inference follows a sequential schedule for $t = 5, 10, \ldots, 40$, 'per prediction' times are the average time per prediction.

training set size of $N = 4$. Results for more combinations (t, N) of the MT-LDS are given in Appendix A.4.4.

The STL approach performs worse than the Pooled approach at $t = 10$ (as may be expected), but by $t = 40$ it has adapted with some success to the data. It is nevertheless not much better than the $N = 4$ MT-LDS. Moreover, inference is extremely expensive compared to the inference required in an MT-LDS (see Table 4.4). Posterior inference of the STL approach is extremely poorly conditioned, requiring the use of expensive methods (HMC, see Section 4.3.1.2) and careful tuning, and could not obviously benefit from the sequential structure of the problem. The difference in runtime is partly attributable to the inference method, but on the other hand, the MT-LDS model *learns* a low-dimensional representation which allows more efficient inference than HMC. In particular, it permitted the development of an efficient sequential approach described in Section 3.5.1.

4.3.2.2 Comparison to RNN approaches

We now turn to the comparison with more flexible RNN models. Table 4.5 provides results for all benchmark models for training sets of size $N = 4, 32, 128$ and predictive posteriors at time $t = 20, 40$. The best performing model for each (N, t) combination is highlighted, together with any model which does not score significantly worse than this (according to a one-sided $\alpha = 0.05$ t-test). The standard Pooled RNN/GRU models have significantly worse performance than the MT-LDS for all N and t. These are data intensive models and there is no mechanism to impose interpolation between the dynamics seen in the training data. Nevertheless, these models are able to perform better than the Pooled LDS due both to the 'teacher forcing' setup and their increased flexibility.

The bias-customised versions of these models are more interesting. With $N = 128$ examples, the MT-Bias RNN is able to show improvement over the Pooled RNN, but struggles to achieve the same performance as the MT-LDS; an RNN cannot easily change dynamic frequencies by changing the bias. However, with a training set size of $N = 128$ the MT-Bias GRU can perform almost on par with the MT-LDS. This is likely because the GRU *can* change frequencies via the bias due to the multiplicative gates; see Section 4.5 for a demonstration.[8] Nevertheless, the MT-Bias GRU has a greater sample complexity

[8] This contrasts with the usual motivation of GRU/LSTM architectures of permitting long-term

	DHO RMSE					
	$N = 4$		$N = 32$		$N = 128$	
Model	$t = 20$	$t = 40$	$t = 20$	$t = 40$	$t = 20$	$t = 40$
ST-LDS*	0.36	0.11	-	-	-	-
Pooled models						
Pooled LDS	0.37	0.31	0.36	0.31	0.36	0.31
Pooled RNN	0.35	0.25	0.23	0.17	0.24	0.16
Pooled GRU	0.30	0.33	0.27	0.19	0.23	0.17
Bias-customisation						
MT-Bias LDS	0.41	0.33	0.37	0.30	0.33	0.27
MT-Bias RNN	0.37	0.36	0.26	0.24	0.16	0.12
MT-Bias GRU	0.36	0.25	0.16	0.13	0.11	**0.06**
Fully customised MT-LDS						
MT-LDS (original)	0.18	**0.12**	**0.11**	**0.07**	**0.09**	**0.06**
MT-LDS (Linear $\mathbf{h}_\phi$)	0.21	**0.15**	**0.13**	**0.08**	0.12	0.07
MT-LDS ($n_x = 8$)	0.22	**0.15**	**0.11**	**0.08**	**0.09**	**0.06**
MT-LDS ($n_x = 16$)	0.22	**0.15**	**0.11**	**0.08**	**0.09**	**0.06**
MT-LDS ($k = 4$)	0.18	**0.13**	**0.11**	**0.07**	**0.10**	**0.06**
MT-LDS ($n_x = 8, k = 8$)	0.19	**0.12**	**0.10**	**0.07**	**0.10**	**0.06**

Table 4.5: DHO results using LDS models, with training set size N shown. Predictive RMSE after $t = 20, 40$. (*) The ST-LDS trains from scratch, and hence on $N = 0$ sequences. The best performing models (t-test with $\alpha = 0.05$, see main text) for each (N, t) are highlighted in bold, except for $(N = 4, t = 20)$ due to high variance of the results.

than the MT-LDS, and is much less interpretable. At the other end of the spectrum, the MT-Bias LDS can perform very little useful adaptation simply by varying its dynamical bias, motivating the need for more flexible adaptation for SSMs than the simple bias customisation of Miladinović et al. (2019).

The performance of the variants of MT-LDS models discussed in Section 4.2.3 is also given in Table 4.5. Regardless of the state dimension n_x or the latent dimension k, all variants perform close to optimally with $N = 128$ by $t = 40$, demonstrating again that the MT-LDS is robust to over-parameterisation. The slightly poorer performance of the linear-Gaussian prior version reflects the suboptimal model learning in Section 4.2.3, although it still improves upon most RNN/GRU approaches.

4.4 Conclusion

This chapter has provided a thorough review of the performance of MT-LDS models on a univariate physical time series dataset. The learning algorithm proposed in Section 3.4.3 has been shown to be effective in closely approximating the true sequence family. For the double DHO dataset, this can be performed with $N \leq 128$ examples, moreover the MT-LDS can outperform most Pooled and MT-Bias RNN/GRU approaches for predictive

dependencies or writing to a 'CPU cache' (Section 2.2.4). The inputs of such gated models can perform bias modulation, facilitating customisation (which can further be 'locked in' via the gates).

tasks with as little as $N = 4$ examples. This strong performance may be explained by the MT-LDS learning to approximate the true degrees of freedom of the DHO (as exemplified in Figure 4-4 for the case of a single DHO). This permits interpolation to never-before-seen sequence examples. However, Figure 4-3 suggests that extrapolation may be more challenging. We should therefore be careful of using a learned MT-LDS on highly novel sequences, but this form of extrapolation is difficult for most modelling approaches.

4.5 Appendix: MT-Bias GRUs can approximate adjustable latent autoregressive systems

It is interesting that the MT-Bias GRU can perform on par with an MT-LDS for the 'double DHO' data with $N = 128$ examples. In this section we show that the MT-Bias GRU can learn an equivalent model to an MT-LDS for this data. Section 4.5.1 provides a detailed demonstration for the case of AR(2) data. The same method can be used to show that the MT-Bias GRU can approximate *any* adjustable latent VAR system, as discussed in Section 4.5.2. We have nevertheless seen the advantages of using an MT-LDS approach: greater data efficiency, interpretability, and smaller model size.

4.5.1 Approximating an adjustable AR(2) system by a GRU

For a standard RNN, the state dynamics are updated via the equation:

$$\text{(RNN)} \qquad \mathbf{x}_t \;=\; \tanh\left(A\mathbf{x}_{t-1} + \mathbf{b}(\mathbf{u}_t, \mathbf{z})\right) \tag{4.15}$$

where we write $\mathbf{b}(\mathbf{u}_t, \mathbf{z}) := B\begin{bmatrix}\mathbf{u}_t & \mathbf{z}\end{bmatrix}$ which encompasses both MT-Bias RNNs and standard RNNs (where $\mathbf{z} = 1$). Contrast this with the system dynamics of a LSTM cell:

$$\text{(LSTM)} \qquad \mathbf{x}_t \;=\; \mathbf{i}_t \odot \tanh\left(A(\mathbf{r}_t \odot \mathbf{x}_{t-1}) + \mathbf{b}(\mathbf{u}_t, \mathbf{z})\right) \;+\; \mathbf{f}_t \odot \mathbf{x}_{t-1} \tag{4.16}$$

where $\mathbf{i}_t$, $\mathbf{r}_t$, $\mathbf{f}_t$ are the *input*, *reset* and *forget* gates respectively; each is a function of $(\mathbf{u}_t, \mathbf{z}, \mathbf{x}_{t-1})$ and has support $[0, 1]^{n_x}$. The implicit LSTM transition matrix at time t is:

$$\tilde{A}_{u_t,z} \;:=\; A\,\mathrm{diag}(\mathbf{r}_t) \tag{4.17}$$

which is a function of $\mathbf{r}_t$ and hence $(\mathbf{u}_t, \mathbf{z})$. Implementations differ in exactly where the reset gate applies in Equation (4.16) both for GRUs and LSTMs (see Section 2.2.4), but the representative power is broadly equivalent.

The comparison of eqs. (4.15-4.16) suggests that modulation of the system's dynamical properties (e.g. frequency, decay rate etc.) by external inputs is much easier for gated architectures than basic RNNs. However, it is not clear exactly what degrees of freedom are possible from Equation (4.16) (with the transition matrix of Equation 4.17) by changing

the inputs $(\mathbf{u}_t, \mathbf{z})$. We will show that there exist a single set of parameters such that Equation (4.16) can model all deterministic AR(2) processes, with the particular process selected by the inputs. This includes all DHO sequences as per this chapter, but also includes non-oscillating sequences. In what follows, we restrict our attention to time-independent $\mathbf{b}(\mathbf{u}_t, \mathbf{z}) = B\mathbf{z}$, i.e. $\mathbf{u}_t = 0$ for all t; and a GRU architecture, hence $\mathbf{i}_t := \mathbf{1} - \mathbf{f}_t$. This is due to our particular interest in MT-Bias GRU models, but our discussion thus also applies to the more general case without these restrictions, extending to standard GRUs and LSTMs with $u_t \neq 0$.

4.5.1.1 Constructing the approximation

Consider a GRU where the input gate is fully open, i.e. $\mathbf{i}_t = \mathbf{1}$ for all t and hence $\mathbf{f}_t = \mathbf{0}$ in Equation (4.16). This simplified model can be written

$$\text{('reset' gate)} \qquad \mathbf{r}(\mathbf{z}) = \boldsymbol{\sigma}(H\mathbf{x}_{t-1} + L\mathbf{z} + \mathbf{d}), \tag{4.18a}$$

$$\text{(system dynamics)} \qquad \mathbf{x}_t = \tanh\left(\underbrace{A\,\mathrm{diag}\left(\mathbf{r}(\mathbf{z})\right)}_{\tilde{A}_{\mathbf{z}}}\mathbf{x}_{t-1} + B\mathbf{z}\right), \tag{4.18b}$$

where $\boldsymbol{\sigma}$ is a logistic sigmoid applied elementwise. We compare this to an AR(2) system in VAR(1) form, which can be written for $\mathbf{x}_t \in \mathbb{R}^2$ as:

$$\mathbf{x}_t = A_{\mathrm{AR}(2)}\mathbf{x}_{t-1} \tag{4.19a}$$

with the transition matrix,

$$A_{\mathrm{AR}(2)} = \begin{bmatrix} \alpha_1 & \alpha_2 \\ 1 & 0 \end{bmatrix}. \tag{4.19b}$$

The d.o.f.s are the parameters α_1, α_2; the lag-1 and lag-2 coefficients respectively. We seek to show that with a state dimension of $n_x = 5$, the matrix $\tilde{A}_{\mathbf{z}}$ can achieve an equivalent system where the range of $\mathbf{z} \in \mathbb{R}^2$ includes all stable combinations of (α_1, α_2). The required region is shown in Figure 4-8 (see Box et al. 2008, §3.2.4 for a proof). It is sufficient to find a system that approximates all AR(2) processes with values $(\alpha_1, \alpha_2) \in [-2, 2] \times [-1, 1]$.

Proof. By comparison of Equation (4.18b) with Equation (4.19a), we set $B = 0$ and assume (wlog) that $|\mathbf{x}_t| \ll 1$ for all t, hence tanh is well approximated by the identity. Since $\mathbf{r}(\mathbf{z})$ is implicitly a function of $\mathbf{x}_{t-1}$, we set $H = 0$ which removes any multiplicative relationships in $\mathbf{x}_{t-1}$ in Equation (4.18b). This results in Equation (4.18) describing a

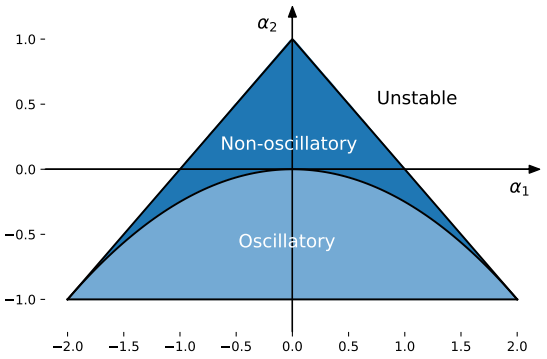


Figure 4-8: Values of (α_1, α_2) resulting in stable AR(2) processes. All values outside the shaded triangle are unstable.

VAR(1) model for every $\mathbf{z}$:

$$\mathbf{r}(\mathbf{z}) = \sigma(L\mathbf{z} + \mathbf{d}) \tag{4.20a}$$

$$\mathbf{x}_t \approx \tilde{A}_{\mathbf{z}} \mathbf{x}_{t-1}. \tag{4.20b}$$

Here $\tilde{A}$ is a function of $\mathbf{r}(\mathbf{z})$ as defined in Equation (4.17), and the approximation in Equation (4.20b) can hold arbitrarily closely. It suffices to find parameters $A, L, \mathbf{d}$ such that for all values $(\alpha_1, \alpha_2) \in [-2, 2] \times [-1, 1]$ in Equation (4.19), there exists an equivalent system in Equation (4.20) by choice of an appropriate $\mathbf{z} \in \mathbb{R}^2$.

To this end, consider the following form of $\tilde{A}_{\mathbf{z}}$:

$$\tilde{A}_{\mathbf{z}}\, \mathbf{x}_{t-1} = A \operatorname{diag}\left(\mathbf{r}(\mathbf{z})\right) \mathbf{x}_{t-1} \tag{4.21}$$

$$= \underbrace{\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{A} \operatorname{diag}(\mathbf{r}(\mathbf{z}))\mathbf{x}_{t-1} \tag{4.22}$$

$$= \underbrace{\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}}_{E} \underbrace{\begin{bmatrix} 2r_1(\mathbf{z}) & -2r_2(\mathbf{z}) & r_3(\mathbf{z}) & -r_4(\mathbf{z}) & 0 \\ 0 & 0 & 0 & 0 & r_5(\mathbf{z}) \end{bmatrix}}_{\text{value matrix}} \mathbf{x}_{t-1} \tag{4.23}$$

$$= E \begin{bmatrix} 2r_1(\mathbf{z})x_1^{t-1} - 2r_2(\mathbf{z})x_2^{t-1} + r_3(\mathbf{z})x_3^{t-1} - r_4(\mathbf{z})x_4^{t-1} \\ r_5(\mathbf{z})x_5^{t-1} \end{bmatrix} \tag{4.24}$$

where $r_j \in [0, 1]$ is the jth element of $\mathbf{r}$, and (with a small abuse of notation) x_j^t is the jth

element of $\mathbf{x}_t$. The 'copy matrix', E takes three copies of the first element in the value matrix and two copies of the second element. This means that $x_1^{t-1} = x_2^{t-1} = x_5^{t-1}$ and $x_3^{t-1} = x_4^{t-1}$, hence:

$$\tilde{A}\,\mathbf{x}_{t-1} \;=\; E \begin{bmatrix} 2(r_1(\mathbf{z}) - r_2(\mathbf{z}))x_1^{t-1} + (r_3(\mathbf{z}) - r_4(\mathbf{z}))x_3^{t-1} \\ r_5(\mathbf{z})x_1^{t-1} \end{bmatrix}. \tag{4.25}$$

Now let us turn to the form of $\mathbf{r}$. We set $L, \mathbf{d}$ as follows:

$$\mathbf{r}(\mathbf{z}) \;=\; \sigma\left(\underbrace{\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 0 & 0 \end{bmatrix}}_{L} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \gamma \end{bmatrix}}_{\mathbf{d}} \right), \tag{4.26}$$

with $\gamma \to \infty$. Using these values of L and $\mathbf{d}$, define a_1 as follows:

$$\begin{aligned} a_1 \;:=\; r_1(\mathbf{z}) - r_2(\mathbf{z}) \;&=\; \sigma(z_1) - \sigma(-z_1) && (4.27)\\ &=\; \sigma(z_1) - (1 - \sigma(z_1)) && (4.28)\\ &=\; 2\sigma(z_1) - 1 \in [-1, 1]. && (4.29) \end{aligned}$$

Similarly define $a_2 := r_3(\mathbf{z}) - r_4(\mathbf{z}) \in [-1, 1]$. Finally $r_5(\mathbf{z}) = \sigma(\gamma) \to 1$. Therefore substituting Equation (4.26) in Equation (4.25) gives us:

$$\tilde{A}\,\mathbf{x}_{t-1} \;=\; E \begin{bmatrix} 2a_1 x_1^{t-1} + a_2 x_3^{t-1} \\ 1 \times x_1^{t-1} \end{bmatrix}. \tag{4.30}$$

where $a_1 \in [-1, 1]$ and $a_2 \in [-1, 1]$. Apart from the duplication of state variables, and up to re-indexing, this is exactly the same form of update as given in the AR(2) equation in eqs. (4.19), with the lag-1 and lag-2 coefficients given by $(2a_1, a_2) \in [-2, 2] \times [-1, 1]$ chosen by $\mathbf{z} \in \mathbb{R}^2$ as required. □

4.5.2 Discussion

As a direct corollary of this demonstration, a MT-Bias GRU with $n_x = 10$ latent states can in principle model the 'double DHO' data with high accuracy. (However it may require a large amount of data, and perhaps a large number of optimisation epochs; this construction requires the relevant gates to saturate.) This goes some way to explain the strong results of the MT-Bias GRU with 128 examples.

In fact, a GRU can approximate an arbitrary transition matrix A for a VAR(1) model. We have presented a demonstration for the AR(2) model (written in VAR(1) form) for

ease of exposition, and due to its relevance to this chapter. However, the same method of approximation can be applied *mutatis mutandis* to any transition matrix. Each real-valued parameter of A requires two columns in the 'value matrix' in Equation (4.22), and hence a fully-adjustable transition matrix of size $n_x \times n_x$ can be approximated by a GRU with $2n_x^2$ hidden units. A lower bound on the requirement is n_x^2 hidden units, due to the required degrees of freedom. Our MTDS method assumes that the d.o.f.s between sequences will not be very large, hence our use of a low-dimensional manifold.

In the (typical) case where LSTM and GRU models are provided with inputs $\{\mathbf{u}_t\}$, these are able to learn the equivalent of a deterministic LDS with customisable and *time-dependent* parameters. Using their gates, LSTM and GRU models may be able to 'lock in' the customisation for indefinite periods of time. So far as we are aware, this is a novel observation, and provides evidence that the benefits of gates in RNNs may in part be attributed to their capability of customising the system dynamics, or performing 'task inference'. Standard RNNs in contrast cannot easily perform any such modulation; they are restricted to changes in the biases. This is an alternative and complementary perspective to the common understanding of gated architectures, which includes the benefits of learning long term dependencies, fixing 'vanishing gradient' issues, and 'reading'/'writing' to a 'CPU cache'.

CHAPTER 5

Modelling the Style of Human Locomotion

We now turn to real-world applications of the multi-task dynamical system. This chapter details the first of two such applications: modelling human locomotion. This is a clear example of a sequence family: each person has a particular walking style (e.g. Lee and Grimson, 2002), yet by its very nature, there is an intrinsic similarity between all human locomotion. We can therefore expect the majority of such differences to be captured with relatively few degrees of freedom. Unlike in Chapter 4, the data are multivariate, with a rich structure between dimensions and nonlinear input-output relationships.

We will be using motion capture (mocap) data for the purposes of modelling – converting to or from video data may be considered separate problems (see e.g. Rogez and Schmid, 2016 for the latter). The MTDS can help to address several outstanding problems for mocap data: training dynamical models in the presence of limited data, robustness to dataset shift, and style transfer for generative models. Furthermore, learning the sequence family via a latent manifold allows us to interpolate or morph style over time (see Figure 5-1), which, to the best of our knowledge, is an entirely novel contribution.

In this chapter, Section 5.1.1 introduces the data representation and preprocessing, and Section 5.1.2 discusses the choice of MTDS model and experimental setup. Section 5.2

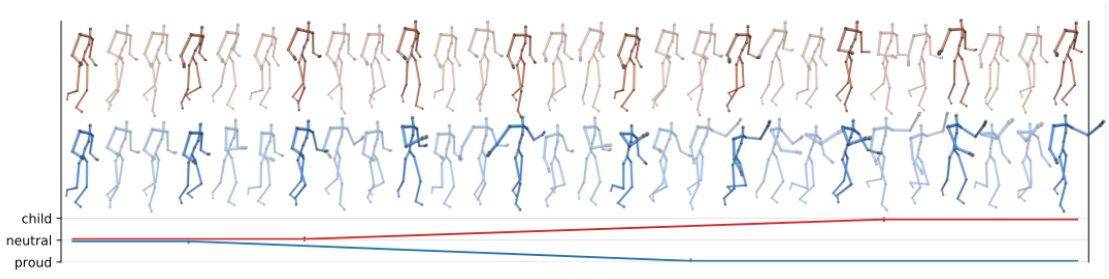
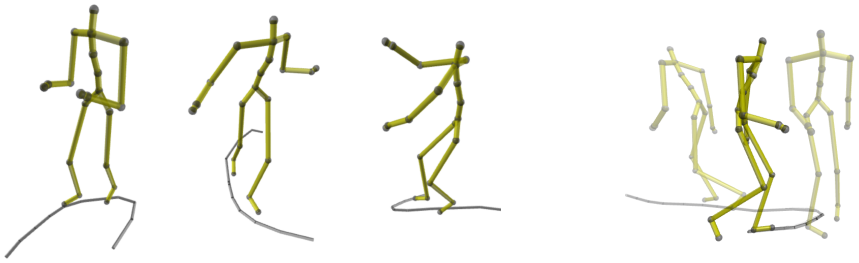


Figure 5-1: Morphing the style of a human locomotion sequence over time: (red) from neutral to childlike; (blue) from neutral to proud; (bottom) interpolation schedule.



(a) (b)

Figure 5-2: (a) Example frames from mocap dataset, together with ± 1 second ground trajectory shown in grey. (b) Example subsequence where the skeleton rotates in the opposite direction to the corner.

gives a brief overview of the related work for mocap modelling. Section 5.3 provides the results of a variety of experiments, looking at latent representations, data efficiency, dataset shift and style transfer. Section 5.4 concludes the chapter.

5.1 Experimental setup

This section provides an introduction to the mocap data (Section 5.1.1), modelling choices and competitor models (Section 5.1.2).

5.1.1 Mocap data

The data are obtained from Mason et al. (2018) which consists of planar walking and running motions in 8 styles recorded with a motion capture (mocap) suit. These styles are: 'angry', 'childlike', 'depressed', 'neutral', 'old', 'proud', 'sexy', 'strutting'. The path (or 'trajectory') along which the skeleton is walking is provided as an input, in order to avoid modelling the random/complex choices of the actor. See Figure 5-2 for examples. In this case the sequence family corresponds to the possible walking styles humans exhibit while following a pre-determined path.

The original data is recorded at 120 fps; we downsample to 30 fps as per Martinez et al. (2017); Pavllo et al. (2018). Each style has 3 - 5 individual sequences of varying length, totalling ca. 2000 frames per style. The data are mapped to a 21-joint skeleton used in the codebase of Holden et al. (2016), shown in Figure 5-3a, which is a subset of the CMU skeleton (De la Torre et al., 2009). Unlike Mason et al. (2018), we do not perform any data augmentation such as mirroring. Our interests are primarily in the contribution of the MTDS in modelling style, rather than in producing high fidelity computer graphics. For inputs to the model, we provide the ground trajectory, the gait cycle (a value in $[0, 2\pi)$ corresponding to the phase of the leg motion), and a Boolean indicator for the rotational direction while traversing a corner (see below for further details).

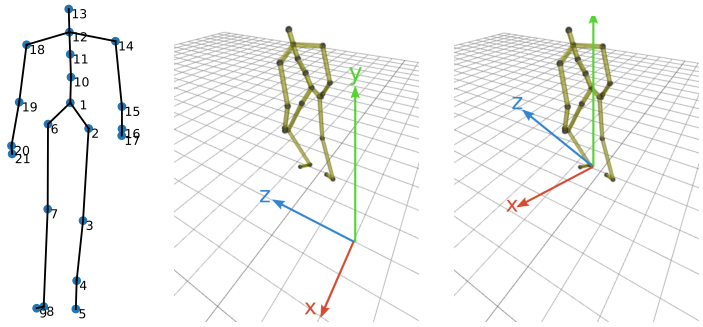


Figure 5-3: (a) The 21-joint skeleton. (b) Eulerian representation. (c) Lagrangian representation.

5.1.1.1 Data representation

We choose a Lagrangian representation (Figure 5-3c) where the coordinate frame is centered at the *root joint* of the skeleton (joint 1 in Figure 5-3a, the pelvis), projected onto the ground. The frame is rotated such that the z-axis points in the 'forward' direction, roughly normal to the body.[1] This is in contrast to the Eulerian frame, which has a fixed position for all t (Figure 5-3b). In the Lagrangian frame, the joint positions are always relative to the root joint, which avoids confusing the overall *trajectory*/*rotation* of the skeleton (captured primarily by the root joint) with the local motions of the other joints.

The relative joint positions within a Lagrangian frame may be represented either by spatial position or joint angle. For the latter, the spatial positions of all joints can be recovered from the angle made with their parent joint via use of forward kinematics (FK). This construction enforces the constraint of constant bone length for the skeleton over time, which is a desirable property. However, it also substantially increases the sensitivity of internal joints. For instance, the rotation of the trunk will disproportionately affect the error of the joints in both arms. For this reason, as in Mason et al. (2018), we have chosen to model the spatial position of joints, which may result in violations of bone length, but avoids these sensitivity issues. See also §2.1, Pavllo et al. (2018).

A further decision is whether to encode the joint positions via velocity (i.e. differencing) which is purported to result in smoother predictions. We encode the root joint (i.e. the position/rotation of the co-ordinate frame) in this way, but not joints 2 to 21 (relative joint motion) due to the problem of accumulating errors over time. Modelling the root joint via velocity is standard in mocap models, not least in order to provide greater control to animators: one can then amend the trajectory of a mocap model with ease. Hence our per-frame representation consists of the 'velocity' $\dot{x}, \dot{z}, \dot{\omega}$ of the co-ordinate frame, the vertical position of the root joint, and 3-d position of the remaining 20 joints, which

[1] The 'forward' direction of a skeleton contains some ambiguity especially due to the small-scale rotations induced by the walking style. We use the direction normal to the shoulder blades (under an 8 frame, or 250ms moving average) similarly to Holden et al. (2016).

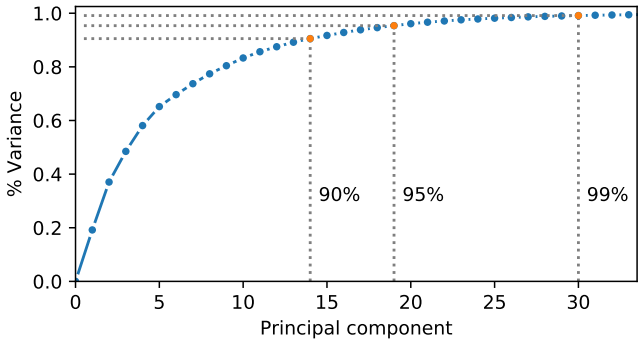


Figure 5-4: Principal components analysis of processed data $\{\mathbf{y}_t\}$ for all styles

gives $\mathbf{y}_t \in R^{64}$. Finally, the $\{\mathbf{y}_t\}$ are standardised to have zero mean and unit variance per observation channel. A principal components analysis indicates that at least 19 dimensions are required in order to retain 95% of the variation (Figure 5-4).

5.1.1.2 Model inputs

Our choice of inputs reflects controls that an animator may wish to manipulate. The first input is the trajectory that the skeleton is to follow. As in Holden et al. (2017), we provide the trajectory over the next second (30 frames), sampled every 5 frames. Unlike previous work, the trajectory history of the prior 30 frames is omitted, since it can be retained in the dynamic state. The (2-d) trajectory co-ordinates are given wrt. the current (Lagrangian) co-ordinate frame, and hence can vary rapidly during a tight corner. In order to provide some continuity in the inputs, we also provide a first difference of the trajectory in absolute (Eulerian) co-ordinates.[2]

The velocity implied by the trajectory does not disambiguate the gait frequency vs. stride length. The same motion may be achieved with fast short steps, or slow long strides. We therefore provide the gait frequency via a phasor (as in Holden et al., 2017), whose frequency may be externally controlled. This is provided by sine and cosine components to avoid the discontinuity at 2π. This may also be useful to an animator. A final ambiguity exists from the trajectory at tight corners: the skeleton can rotate either towards the focus of the corner, or in the opposite direction. Figure 5-2b demonstrates the latter, which appears not infrequently in the data. We provide a boolean indicator for each of the 6 sampled trajectory timesteps, indicating corners for which this happens.

Altogether we have $\mathbf{u}_t \in \mathbb{R}^{32}$: 12 inputs each for the Lagrangian trajectory and differenced Eulerian trajectory, 2 inputs for the gait phase and 6 inputs for the turning indicators. These inputs $\{\mathbf{u}_t\}$ are standardised to have zero mean and unit variance. Constructing these inputs was not straight-forward, and we provide details of the most significant

[2]Here we are using a simple first-difference operator on the raw values. It may be noted that this operation amplifies high frequency components in the signal (e.g. Oppenheim et al., 1999, §2.9), but smoother alternatives appeared not to be necessary for these experiments.

pre-processing steps in Appendix A.5.1. A particular problem was the stylistic swaying/movement of the root joint which had to be removed from the trajectory. This leaked information about the style, and moreover prevented customisation of the pelvic movement when performing style transfer. Future work might consider attempting to automate this process. For instance one could try to learn an intermediate representation $\mathbf{u}_t \rightarrow \tilde{\mathbf{u}}_t$ which functions as the input to the mocap model, but with an adversarial objective to predict the style from the $\{\tilde{\mathbf{u}}_t\}$.

5.1.2 Models

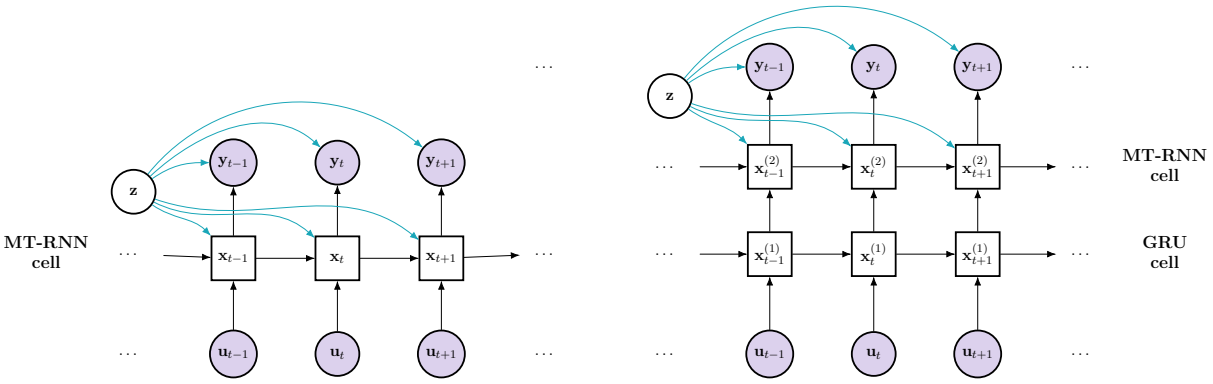
Our goal is to demonstrate that the style of human locomotion can be modelled independently of its trajectory via use of the latent variable $\mathbf{z}$. This should enable a variety of benefits discussed above, not least end user control. This section will motivate and describe our MTDS model, and provide a brief description of the competitor models.

5.1.2.1 MTDS model

In contrast to Chapter 4, a linear dynamical system (LDS) is not well suited to modelling mocap data. Our preliminary work demonstrated empirically poorer performance for LDS models compared to RNNs even when trained on a single style. A particular problem is that the trajectory and limb positions appear to be highly nonlinear functions of the inputs when turning corners. Unfortunately, since the physical space used by Mason et al. (2018) was relatively small, turning corners is a very common feature in the data. The data also exhibit a variety of asymmetric quasi-periodic sequences which cannot be modelled efficiently using an LDS. See Figure A-9 in the Appendix for an example. Such problems are compounded by the fact that classic subspace identification methods for LDSs (e.g. Van Overschee and De Moor, 2012) scale poorly with the size of the data.

A natural choice is therefore a MT-RNN (or gated variant such as MT-GRU or MT-LSTM) as shown in Figure 5-5a. Preliminary MT-RNN experiments showed strong performance on the training data, but on changing the value of $\mathbf{z}$, predictions either did not change, or became implausible. This appears to be because the input trajectories are not independent of the style (even after removing the information leakage described above). This resulted in style inference being partially performed by the RNN rather than the latent $\mathbf{z}$. Furthermore, since certain features such as corner types, or extreme values of speed, and gait frequency may be found only within a single style (or group of styles), the response to such unique inputs was only learned for certain values of $\mathbf{z}$.

In order to capture the global input-output relationships across all styles, we propose a two-layer RNN structure, where the first layer is *not* multi-task and is able to learn a shared representation across all tasks. The second layer is a MT-RNN which performs specialisation to the style using the latent variable $\mathbf{z}$. The separation of responsibility between layers is assisted by three important refinements. The first layer passes information



(a) Standard MT-RNN **(b)** Proposed MTDS model

Figure 5-5: Graphical model of a standard MT-RNN and the proposed MTDS, for a single style i. For clarity, the superscript i is omitted, and the dependence on $\mathbf{z}$ is shown in a different colour.

to the second layer via a bottleneck of ℓ units, encouraging the first layer to focus on the globally most important features. Secondly, we omit a skip connection from the first layer, which would undermine the bottleneck. Finally, the MT-RNN is given a relatively higher learning rate, providing a preference to model greater variation than the first layer, but it only has access to the inputs via the shared representation from the first layer, forcing $\mathbf{z}$ to explain more of the variance.

In detail, our proposed MTDS model is a 2 hidden-layer recurrent model where the first hidden layer is a 1024 unit GRU and the second hidden layer is a 128 unit MT-RNN, followed by a linear decoding layer. Explicitly, omitting index i, the model for a style represented by the task variable $\mathbf{z}$ is:

$$[\boldsymbol{\psi}_2(\mathbf{z}), C(\mathbf{z}), \mathbf{d}(\mathbf{z})] = \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z}), \tag{5.1}$$

$$\mathbf{x}_{1,t} = \text{GRUCell}_{1024}\{\mathbf{x}_{1,t-1},\ \mathbf{u}_t;\ \ \boldsymbol{\psi}_1\}, \tag{5.2}$$

$$\mathbf{x}_{2,t} = \text{RNNCell}_{128}\{\mathbf{x}_{2,t-1},\ H\mathbf{x}_{1,t-1};\ \boldsymbol{\psi}_2(\mathbf{z})\}, \tag{5.3}$$

$$\hat{\mathbf{y}}_t = C(\mathbf{z})\mathbf{x}_{2,t} + \mathbf{d}(\mathbf{z}), \tag{5.4}$$

for $t = 1, \ldots, T$. See Figure 5-5b for a graphical model. The parameters which depend on $\mathbf{z}$ are $\boldsymbol{\theta} = \{\boldsymbol{\psi}_2, C, \mathbf{d}\} \in \mathbb{R}^p$ and the learnable parameters are thus $\{\boldsymbol{\psi}_1, H, \boldsymbol{\phi}\}$.[3] In our experiments, we use $\ell = 24$ units for the bottleneck matrix H.[4] The first layer GRU uses 1024 units primarily for qualitative reasons, since it was observed to produce smoother animations than smaller networks.[5] Using a MT-GRU instead of a MT-RNN in layer two produced worse style transfer; changing the value of $\mathbf{z}$ in this case often made little

[3] The parameters which depend on $\mathbf{z}$ have the following dimensions; $\boldsymbol{\psi}_2$: $128 \times 128 + 128$; C_2: 128×64; and $\mathbf{d}$: 64. This results in a total of $p = 24,768$ dimensions for the base parameters of the MT-RNN.

[4] The number of bottleneck units ℓ was the first one that was tried, motivated in part by the effective degrees of freedom of the inputs found by PCA (Figure 5-4).

[5] Use of 512 instead of 1024 units in initial experiments did not significantly affect the held-out performance during training, but the animations were noticeably more jerky.

difference to the style. This may be related to the fact that GRUs can innately perform style inference via use of gates (see Section 4.5).

The choice of $\mathbf{h}_\phi$ appeared to be less important than in Chapter 4. We used an affine approach (which calculates the parameters $\boldsymbol{\theta} \in \mathbb{R}^p$ using a $k \times p$ matrix and p-dimensional bias) instead of an MLP approach (for which see Appendix A.5.2) since the performance was similar, inference of the latent $\mathbf{z}$ was often faster, and the parameters were apparently more robust to choice of initialisation. A nonlinear $\mathbf{h}_\phi$ may be more important when the base model is simpler as in e.g. Section 4.1.3. In contrast to a MT-LDS, the increased flexibility of a MT-RNN may permit the model to more easily find a linear subspace which contains the required sequence family parameters.

In order to ensure smooth variation of the dynamics wrt. $\mathbf{z}$, it proved important to fix the dynamical bias of the MT-RNN layer to a single point estimate (i.e. no dependence on $\mathbf{z}$). Smooth variation of this parameter can otherwise result in 'jumps' while interpolating between sequences. A particularly stark example is given in Figure 5-6 which visualises various sequence predictions of a given joint while interpolating the latent $\mathbf{z}$ from a 'strutting' (blue) to 'angry' (red) style. The top figure shows the model output for the MT-RNN model where the bias is fixed wrt. $\mathbf{z}$, the bottom shows the output for the model where *only* the bias depends on $\mathbf{z}$. We speculate that modulating the bias induces bifurcations in the state space, whereas adapting the transition matrix allows for smooth interpolation (as discussed in Section 3.2.2).

5.1.2.2 Benchmark models

For comparison, we implement a standard 1024-unit GRU pooled over all styles, which serves both as a competitor model (Martinez et al., 2017) and an ablation test. Unlike the pooled LDS of Section 4, this pooled model can perform some 'multi-task'-like customisation, but in an implicit and black-box manner. Style inference follows the approach in Martinez et al. (2017) who pass an initial seed sequence $\mathbf{y}_{1:T_{\text{enc}}}$ to the network before prediction. In our experiments we use $T_{\text{enc}} = 64$ frames. This is a form of sequence-to-sequence (seq2seq) learning with shared weights between the encoder and decoder.

Secondly, we provide a restricted form of MT-RNN, following recent work from Miladinović et al. (2019), where only the bias/offset of the RNN is a function of $\mathbf{z}$ (named 'MT-Bias'). The architecture is otherwise the same as the MTDS model above. We also provide constant baseline predictions of (i) the training set mean and (ii) the last observed frame of the seed sequence ('zero-velocity' prediction). We do not compare against any other models listed in the related work of Section 3.7 since any advantage would be derived from an orthogonal contribution to our own. We wish to isolate the benefits due to the latent $\mathbf{z}$. See Section 5.2 below for further discussion.

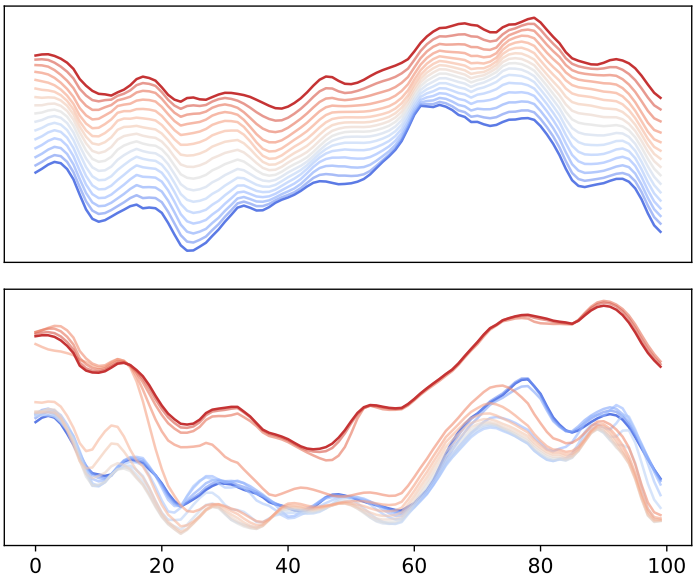


Figure 5-6: Sequence interpolation for model of joint 11 (x direction) over time $t = 1, \ldots, 100$ with latent $\mathbf{z}$ varied from 'strutting' (blue) to 'angry' (red) style. (top) MTDS model; (bottom) MT-Bias model.

5.1.2.3 Learning and Inference

All models were learned using an 'open-loop' objective, i.e. the $\mathbf{y}_t$ are not appended to the inputs during prediction as in 'teacher forcing' or standard MLE training ('closed loop').[6] This forces the model to recover from its mistakes and was first introduced in this context in Martinez et al. (2017). To demonstrate the advantage of this, we also provide results for the standard GRU models trained using a closed loop criterion (i.e. via teacher forcing). We give some model-specific training details below; more can be found in Appendix A.5.2.1.

MTDS, MT-Bias models Each sequence was broken into overlapping segments of length 64 (approx. two second intervals), with a different $\mathbf{z}$ per segment. Unlike open-loop training in prior work, we do not append the model predictions $\hat{\mathbf{y}}_t$ to the inputs at the following time step ($\mathbf{u}_{t+1}$). This is performed in previous work since observations $\mathbf{y}_{1:T_{\text{enc}}}$ are required at the encoding stage, and hence predictions are used in their place for the decoding stage. By use of an explicit latent $\mathbf{z}$, we avoid the need to perform encoding via the sequence model, and hence avoid the need to append the predictions to the inputs; the information from the inputs is already contained in the recurrent state.

The model was optimised using the variational procedure in Section 3.4.2, using a slower learning rate (by $\mathcal{O}(10)$) for the first layer parameters (i.e. $\boldsymbol{\psi}_1, H, C_1$) to obtain a more descriptive $\mathbf{z}$. We used standard variational inference for each $\mathbf{z}^{(i)}$ (i.e. each posterior is

[6] We borrow the terms 'open-' and 'closed-loop' from control theory, e.g. Aström and Murray (2010), §1.

parameterised directly), which worked better in general than amortised inference using an inference network. A form of KL annealing (Bowman et al., 2016) was also used for improving the quality of the latent description. Hyperparameters were chosen both via reference to the ELBO and the *qualitative* training set fit, together with plausible style transfer capabilities. We report results for latent dimensions $k = 3, 5, 7$ to provide the reader with an intuition of this hyperparameter's importance. See Appendix A.5.2.1 for further details.

Benchmark models The models are trained on predicting length 64 sequences (chosen at random at each iteration), using the encoding from the previous $T_{\text{enc}} = 64$ frames. The hyper-parameters were chosen using a grid search over learning rate, regularisation, and optimiser {Adam, SGD}. We perform the search over the pooled data for all 8 styles, with a stratified sample of 12.5% held out for a validation set. The models for each experiment were trained using early stopping, with the model chosen on a 12.5% validation set.

Inference The sequential AdaIS approach of Section 3.5.1 was used to understand the nature of the MTDS posteriors over $\mathbf{z}$. Each observation sequence has 64 frames each of dimension 64 and the posterior is fairly concentrated. Since forward simulation is relatively expensive for our chosen MTDS model, AdaIS is slower than for the DHO experiments (Chapter 4). For example, our $k = 3$ experiments took approx. 24 seconds for inference per 64-frame sequence. During preliminary investigation, each posterior was unimodal and approximately Gaussian, and for prediction, the MAP value performed similarly to the posterior predictive mean, perhaps due to the posterior concentration. Our experiments hence used a MAP estimate of $\mathbf{z}$ at test time to approximate the posterior.

5.2 Related work

Many generative models have been proposed for mocap data. Earlier approaches include a nonlinear dynamical system parameterised by Gaussian processes (Wang et al., 2008) and a recurrent Restricted Boltzmann Machine (Taylor et al., 2010). Holden et al. (2016) propose a nonlinear autoregressive model which is extended in Holden et al. (2017) by making the network weights a function of the gait phase. Competitive RNN-based approaches are introduced in Fragkiadaki et al. (2015), and Martinez et al. (2017) introduce the idea of *open loop* or 'sampled' training (cf. Bengio et al., 2015) in order to avoid predictions converging to a mean pose. Pavllo et al. (2018) include quaternion arithmetic within a recurrent model, but due to a wide variety of pre-processing changes, the value of these improvements is not clear.

Style transfer has been widely explored for mocap data e.g. Hsu et al. (2005); Min et al. (2010); Xia et al. (2015), to name only a few. However, there has been comparatively little work for style transfer in generative models. Recently Mason et al. (2018) proposed style

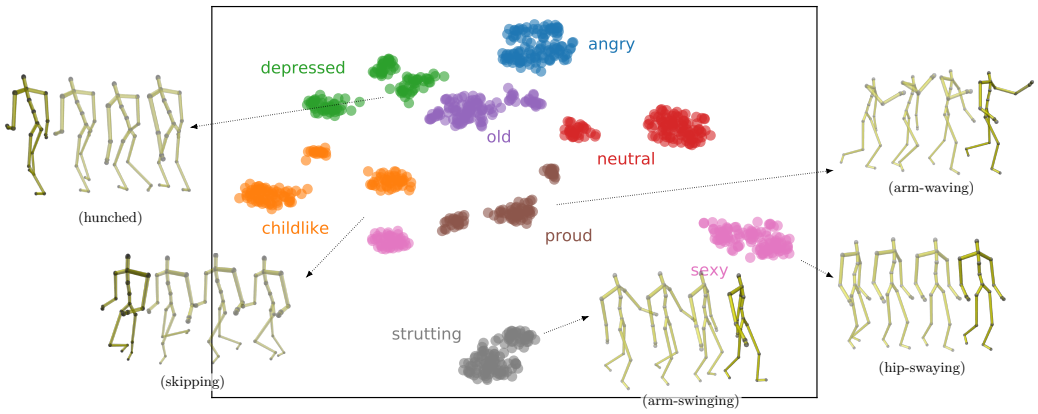


Figure 5-7: $k = 8$ mean embedding of each sequence segment, visualised via t-SNE, coloured by its (unseen) task label.

transfer via residual adapters to the approach of Holden et al. (2017), but does not provide quantitative results. We are not aware of any work which applies generally to dynamical systems, much less of any existing work that permits style *interpolation*.

Some inspiration for style transfer in RNNs may be taken from other applications (see Section 3.7). Miladinović et al. (2019); Hsu et al. (2017) propose modulating dynamical biases in order to disentangle sources of variation. Yingzhen and Mandt (2018) modulate an auxiliary input to the emission network. Other authors have proposed architectural changes or adversarial objectives to disentangle static and dynamic features in video data. Generally applicable examples include Tulyakov et al. (2018); Hsieh et al. (2018); Kosiorek et al. (2018), however in all cases, such features are modulated simply via changing the bias or input to the recurrent generation network. This motivates our decision in 5.1.2.2 to include bias-customised variants as our primary point of comparison, isolating the contribution of our latent $\mathbf{z}$.

5.3 Results

Figure 5-7 shows a t-SNE plot (Van der Maaten and Hinton, 2008) of the mean embedding of $\mathbf{z}$ for each of the 64-frame segments in the dataset, coloured by the true style label.[7] (We remind the reader that this label is unavailable during training.) Our MTDS model can successfully disambiguate the styles without supervision, and moreover provides a more fine-grained representation than the original labels. Many styles have at least two sub-styles (e.g. 'childlike' comprises both skipping and juggling motions). This visualisation suggests that our MTDS model can indeed learn a useful manifold over sequence styles; further investigation in Section 5.3.4 validates the intermediate points on this manifold, and demonstrates the potential of style interpolation.

[7] A direct plot for a $k = 2$ MTDS is also provided in Appendix A.5.3.1; while there are insufficient degrees of freedom in this case, styles are merged in sensible places.

This granular description of sequence style, and customisation of predictions is unavailable from any of our competitor models. In the case of standard GRUs, some customisation is occurring, but it is entangled in the various hidden units, and unavailable to an end user. Moreover, it cannot be controlled should a different style be desired. In the case of bias-customisation (MT-Bias models, related to e.g. Miladinović et al., 2019, Hsieh et al., 2018 and Kosiorek et al., 2018), the latent representation appears to be much poorer. Appendix A.5.3.1 provides a comparison of latent representation between the MTDS and MT-Bias models, where the MT-Bias shows substantial conflation of styles. This results in significantly less control over style than our MTDS model (see style transfer experiments in Section 5.3.3), alongside the aforementioned problems with interpolation (as e.g. in Figure 5-6).

To demonstrate the benefit of the MTDS approach quantitatively, we provide results from three experiments, considering (i) data efficiency, (ii) performance on unseen walking styles, and (iii) style transfer.

5.3.1 Data efficiency

We test the conventional advantage of MTL by considering reduced subsets of the original dataset. We use six training sets with 2^8 to 2^{13} frames per style (logarithmically spaced), with sampling stratified carefully across major variations of all styles. A 'single-task' 1024-unit GRU model is also included for comparison, which is trained and tested on a single style. In all cases, the test performance (MSE) is calculated from 4 held-out sequences from each style (64 frames each), and averaged over all styles.

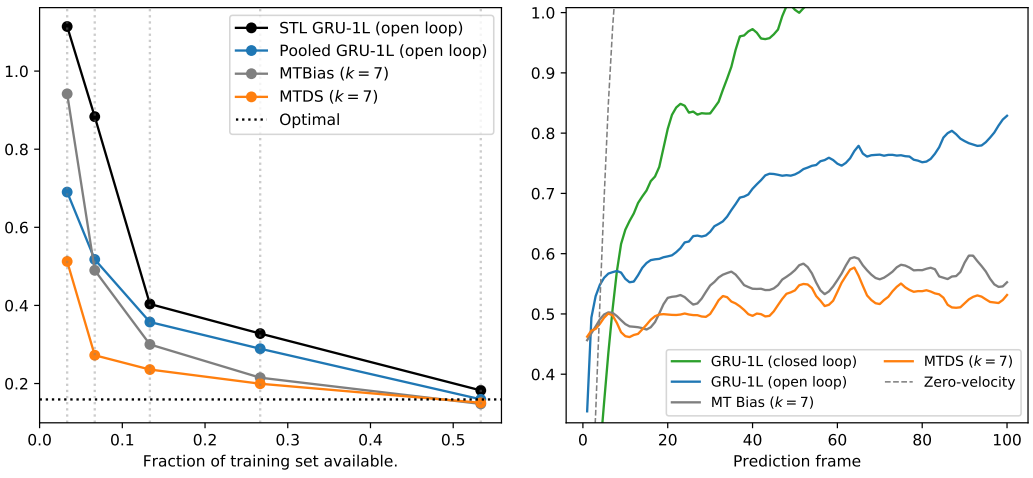
The results for the six dataset sizes are shown in Table 5.1. Some example animations can be found from the linked video in Section 5.3.4. The open-loop GRU (Martinez et al., 2017) performs far better than the standard closed-loop variant, and we will omit the latter from further discussion. The results show strong performances for the MTDS approach, which tends to perform best for $k > 3$.[8] For small datasets ($< 50\%$ of the dataset), we observe advantages from all multi-task models (including the Pooled GRU) over a STL approach. However, the MTDS demonstrates far superior performance than the other MT approaches for smaller datasets; achieving 0.27 MSE after only 7% of the dataset. The MT-Bias model requires more than twice this amount of data to obtain the same performance, and the Pooled GRU requires more than four times this amount. These results are plotted graphically in Figure 5-8a.[9] Note that for dataset sizes up to (and including) 13% of the original, the MTDS is equal or better across *all* styles and dataset sizes except for the {'angry', 'sexy'} styles for the smallest dataset. See Appendix A.5.3.2 for a per-style breakdown.

[8] Results for $k = 5$ and $k = 7$ are statistically equivalent, using a paired t-test with significance level $\alpha = 0.05$.

[9] Standard errors are omitted from this graph for clarity; but see Table 5.1 where the best performing model for each training set size is highlighted, together with any models that did not perform significantly worse according to a one-sided paired $\alpha = 0.05$ t-test.

	MSE					
	Training set size					
Model	3%	7%	13%	27%	53%	97%
Training mean	0.76	0.76	0.72	0.73	0.73	0.73
Zero-velocity	1.23	1.23	1.23	1.23	1.23	1.23
STL GRU (open loop)	1.11	0.88	0.40	0.33	**0.18**	**0.18**
Pooled GRU (closed loop)	0.79	0.61	0.82	0.87	0.76	1.21
Pooled GRU (open loop)	0.69	0.52	0.36	0.29	**0.16**	**0.16**
MT Bias ($k = 3$)	0.93	0.44	0.30	**0.21**	**0.14**	**0.16**
MT Bias ($k = 5$)	0.98	0.44	0.30	**0.20**	**0.14**	**0.16**
MT Bias ($k = 7$)	0.94	0.49	0.30	**0.21**	**0.15**	**0.16**
MTDS ($k = 3$)	0.62	0.34	0.35	**0.21**	0.21	0.19
MTDS ($k = 5$)	**0.53**	**0.29**	**0.22**	**0.19**	**0.15**	**0.16**
MTDS ($k = 7$)	**0.51**	**0.27**	**0.24**	**0.20**	**0.16**	**0.18**

Table 5.1: Experiment 1 (data efficiency): MSE for length-64 predictions where training set size is expressed as a fraction of the original dataset. The best performances for each training set size (up to $\alpha = 0.05$ significance) are highlighted.



(a) **(b)**

Figure 5-8: (a) Experiment 1 (data efficiency): out-of-sample MSE by % of training set seen. The performance achieved by the GRU models for the entire training set is shown as the 'Optimal' dashed line. (b) Experiment 2 (novel test data): MSE performance (avg over folds).

5.3.2 Novel test data

Our second experiment investigates how well the MTDS can generalise to novel sequence styles. This is similar to a domain adaptation or zero-shot learning task. We consider a leave-one-out (LOO) procedure with eight folds, where each fold has a training set comprising 7 styles, and a test set comprising the held-out style. We consider the deterioration of predictive MSE over a large time window ($\tau \leq 200$, ca. 7 seconds). It can be relatively easy to predict τ-steps ahead for $\tau \leq 10$ (see Martinez et al., 2017) even for novel sequences, but the error usually deteriorates with increasing τ. Our results report the predictive MSE for each τ averaged over the 8 folds, and over 32 different starting locations within each fold. The competitor models are as above (excluding the STL model), but we also include a 2-layer GRU (with 1024 units in each layer) as the training set is generally larger.

A summary of results is shown in Figure 5-8b, where the axes are truncated for clarity (for the full range see Appendix A.5.3.3). The standard Pooled GRUs work well for small values of τ but degrade very quickly. The closed-loop variants perform better for $\tau \leq 5$ but degrade even more rapidly than the open-loop approach. The deteriorating results for these models are consistent with the inputs moving the state into an area where the dynamics have not been trained. In contrast, the MTDS and MT-Bias models find a better customisation which evidences very little worsening over the predictive interval. Importantly, these models are able to 'remember' their customisation over long intervals via the latent $\mathbf{z}$.

A per-style breakdown comparing the MTDS to the pooled GRU approach can be found in Appendix A.5.3.3. The $k = 7$ MTDS shows equal or better performance to the pooled GRU on all styles for $\tau \geq 10$, and its initial performance may perhaps be improved via interpolation from the seed sequence given the performance of the 'zero-velocity' baseline for $\tau \leq 5$. The results of the 2-layer competitors are shown in Table 5.2, but they achieve similar performance to the 1-layer models on aggregate (a similar result is suggested in Martinez et al., 2017).

These experiments demonstrate that it can be crucial to retain control over the task inference for novel data, rather than delegating it to a black-box procedure; the implicit inference of standard GRU networks can perform very poorly when presented with unexpected inputs. For this experiment, while the full MTDS consistently outperforms the MT-Bias approach, the difference is not large. In practice, perhaps either could be used. We note that all models struggle to capture the arm movements of novel styles, since these are often entirely novel. Customisation to the legs and trunk is easier since less extrapolation is required (see animation videos linked in Section 5.3.4).

5.3.3 Style transfer

Finally, we investigate how much control is available via the latent code, $\mathbf{z}$. Style transfer is a further property of MTDS-type models available by virtue of the latent variable. It is

Model	MSE					
	$\tau = 5$	$\tau = 10$	$\tau = 20$	$\tau = 50$	$\tau = 100$	$\tau = 200$
Training mean	1.04	1.04	1.05	1.04	1.06	1.07
Zero-velocity	0.69	1.20	1.37	1.21	1.35	1.48
Pooled 1-layer GRU (closed loop)	**0.35**	0.64	0.81	1.00	1.45	7.28
Pooled 2-layer GRU (closed loop)	**0.34**	0.61	0.79	0.97	1.41	6.34
Pooled 1-layer GRU (open loop)	0.56	0.56	0.60	0.73	0.83	0.92
Pooled 2-layer GRU (open loop)	0.53	0.55	0.59	0.73	0.85	0.94
MT Bias ($k = 3$)	0.60	0.60	0.58	0.59	0.64	0.63
MT Bias ($k = 7$)	0.50	**0.48**	0.53	0.57	**0.55**	**0.63**
MTDS ($k = 3$)	0.61	0.62	0.59	0.61	0.63	0.63
MTDS ($k = 7$)	0.49	**0.46**	**0.50**	**0.54**	**0.53**	**0.61**

Table 5.2: Experiment 2 (novel test data): average predictive MSE at $\tau = 5, 10, 20, 50, 100, 200$. The best performing model(s) (up to $\alpha = 0.05$ significance) for each t is highlighted in bold.

unavailable from standard pooled GRU approaches, and hence in this section we can only compare the full MTDS with the restricted MT-Bias model. For each pair of styles (s_1, s_2) we investigate style transfer from a source sequence of style s_1 to a target style s_2. Due to the *within*-style variation, we use four different source sequences for each pair (s_1, s_2), and choose the latent encoding $\mathbf{z}$ of s_2 from the posterior mean $(\boldsymbol{\mu}_{\boldsymbol{\lambda}})$ of an appropriate sequence in $\mathcal{D}$. Evaluation is performed via use of a classifier, for which we use a 512-unit GRU to encode the sequence followed by a 300-unit hidden layer MLP with multinomial emissions.[10] The classifier is trained on the complete data using the (previously unused) labels from Mason et al. (2018). Qualitative results are available via the videos linked in Section 5.3.4, and further experimental details are given in Appendix A.5.3.4.

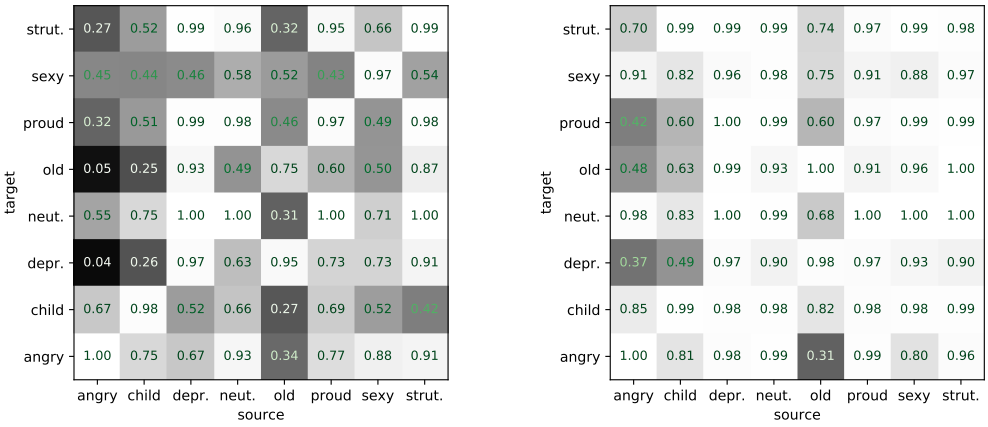
The results are summarised in Table 5.3, which provides the average 'probability' assigned by the classifier for each *target* style s_2, averaged over all the input sequences where the source $s_1 \neq s_2$. (The best performing model for each style is highlighted in bold; standard errors are omitted, but see below for a per-style breakdown.) Successful style transfer should result in the classifier assigning a high probability to the target style. The results suggest that the style can generally be well controlled by $\mathbf{z}$ in the case of the full MTDS, but the MT-Bias model exhibits reduced control for some (source, target) pairs.

Target	MT Bias	MTDS
Angry	0.78	**0.86**
Child	0.59	**0.95**
Depr.	0.65	**0.81**
Neut.	0.79	**0.93**
Old	0.55	**0.86**
Proud	0.71	**0.82**
Sexy	0.55	**0.90**
Strut	0.71	**0.92**

Table 5.3: Experiment 3 (style transfer): classifier probability of target style averaged over all input styles.

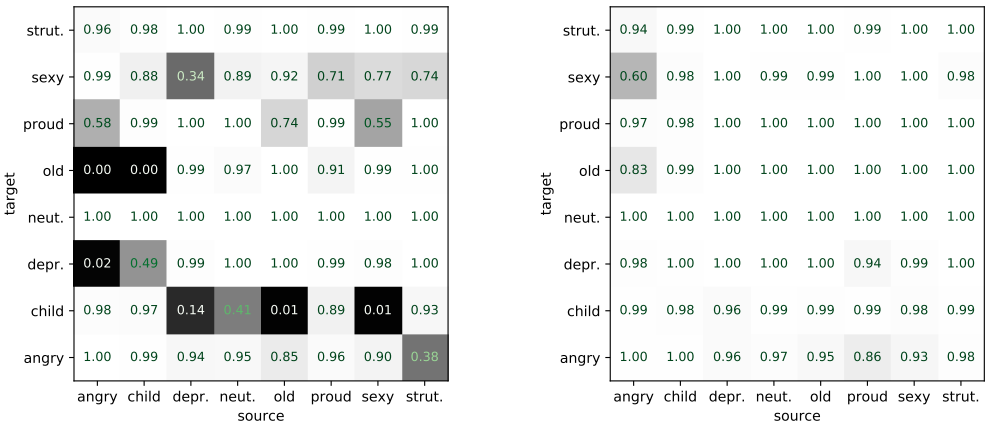
The breakdown of these results into their (source, target) pairs are given in Figure 5-9. The cells provide the average classifier probability for the target style over each combination (averaged over the four source sequences). Success-

[10] The classifier did not need much architectural tuning. The hyperparameters were chosen here as sensible defaults, and after training achieved greater than 99% accuracy on held-out data across all styles.



(a) MT-Bias **(b)** MTDS

Figure 5-9: Average classification accuracy for style transfer using inputs from source style (columns) and latent code $\mathbf{z}$ from target style (rows). There is no style transfer on the diagonal.



(a) MT-Bias **(b)** MTDS

Figure 5-10: Average classification accuracy for style transfer where only a single source input is used for each (source, target) pair. The configuration of the matrix is the same as Figure 5-9.

ful style transfer should result in a high score in every cell. For most (source, target) pairs, the full MTDS model substantially outperforms the MT-Bias model, resulting in superior user control in the majority of cases. We can gain some insight into the MT-Bias performance via its latent representation (Appendix A.5.3.1), where we see it has conflated a variety of styles, which are disambiguated therefore only via the inputs. It is therefore not surprising that changing the latent $\mathbf{z}$ often results in unrecognisable changes.

It is notable that both models exhibit worse results when styles are associated with extremes of the input distribution. Specifically, both the 'childlike' and 'angry' styles have unusually fast trajectories, and the 'old' style has unusually slow ones. The lowest scores tend to involve these styles as either the source or the target. This suggests that the first layer (GRU) fails to provide a coherent shared representation of these behaviours, and/or there is some information leakage from the inputs. Further improvements may be available, perhaps by using an adversarial loss, or applying domain knowledge to the model. Since our goal is to demonstrate the contributions of the MTDS on generic model architectures, we leave this to future work.

Providing style transfer from a wide variety of source styles is a challenging task. We are attempting to find a single latent code ($\mathbf{z}^{(s_2)}$) for each target style s_2 which can reliably transfer style from 28 different source sequences, many of which are mismatched to the target style. We consider a more pragmatic experiment where the variety of source styles is reduced to a single example each. Note nevertheless that the same $\mathbf{z}^{(s_2)}$ is used across all sources s_1. The results of this secondary experiment are provided in Figure 5-10. In this case, the MTDS achieves successful style transfer for almost all (source, target) combinations. The MT-Bias model still has many notable failures.

5.3.4 Qualitative results

Qualitatively, the MTDS appears to learn a sensible manifold of walking styles. Figure 5-7 (at the beginning of Section 5.3) provides a visualisation of this latent space where similar motions are placed close to each other. For instance, the neighboring 'old' and 'depressed' styles both involve leaning over, and the neighboring 'childlike' and 'sexy' clusters are both 'skipping'-type motions. This suggests that the embedding is more informative than the original labels.

Visualisations of sequence interpolation, such as Figure 5-6 (top) suggest that smooth interpolation is available through these styles in latent space, but we must animate the results for verification. This is indeed the case, and evidence is provided below. Moreover, we can take advantage of this by morphing styles dynamically, allowing smooth changes over time for a mocap character between any of the eight styles in the dataset.

Smooth interpolation between styles is also available from the full MTDS model as suggested by Figure 5-1 (page 88), and Figure 5-6 (top, page 95); this can be verified in the animations linked below. Interpolation of the style manifold of the MT-Bias model

tends to result in 'jumps', as suggested by Figure 5-6 (bottom). The full MTDS model can hence be used to morph styles dynamically, allowing smooth changes over time for a mocap character between any of the eight styles in the dataset.

Animations The animations for all experiments have been collected into a project webpage[11]. They form a crucial part of the model evaluation, which cannot be adequately summarised in a non-dynamic medium such as this thesis. A brief description of all animations is provided below.

1. **In-sample predictions** https://vimeo.com/362069486. The goal is to showcase the best possible performance of the models by visualising their performance on the training set.
2. **MTL examples** https://vimeo.com/362122944. Examples from Section 5.3.1. We compare the quality of animations and fit to the ground truth for two limited training set sizes (6.7% and 13.3% of the full data). For both models, MSE to the ground truth is given, averaged over the entire predictive window (length 256). This is different to the experimental setup, which uses only the first 64 frames.
3. **Novel test examples** https://vimeo.com/362068342. Examples from Section 5.3.2. We show the adaptions obtained by each model to novel sequences. Again, MSE is given averaged over the predictive window (length 256).
4. **Style morphing** https://vimeo.com/361910646. This animation demonstrates the effect of changing the latent code over time. This also provides evidence of style transfer and style interpolation from Section 5.3.3.

The animations provide a comparison between the ground truth, the relevant MTDS model, and the (1 layer, open loop) pooled GRU model (where applicable). In all cases, animations are a complete predictive rollout with no access to the ground truth.

5.4 Conclusion

This chapter has investigated the use of MTDS models on a rich high-dimensional time-series with highly nonlinear relationships. This has corroborated an observation in Chapter 4 that GRUs (and to a lesser extent RNNs) are able to perform some implicit style inference, and hence the construction of MTDS models requires care when using these base models. Appropriately defined MTDS models are able to learn a highly informative latent representation of *dynamic* differences between sequences (which is unavailable from existing latent variable models). Furthermore, this representation can be used to control the model predictions, and interpolate between members of the sequence family (i.e. style of human locomotion).

[11] https://sites.google.com/view/mtds-customized-predictions/home

Individual experiments have shown that the MTDS can substantially outperform existing methods in small data settings, as well as avoid performance deterioration under dataset shift. The same model can also be used to perform highly effective style transfer. A simpler MTDS variant which only customises the dynamic bias (MT-Bias, proposed most notably by Miladinović et al., 2019) provides an interesting comparison point. This model is similarly robust to dataset shift, but at least for these data, cannot capture such a rich latent representation, permit reliable style transfer or morphing, and performs substantially worse when data are scarce.

CHAPTER 6

Unsupervised Personalization of Pharmacodynamic Models for Anaesthetic Induction

Our final application of multi-task dynamical systems is in the medical domain. In this context (unlike many high profile applications of machine learning) sample sizes are small, and mistakes can have severe consequences. To avoid making serious mistakes, existing models tend to be simple, inflexible and predict average effects. *Personalising* or improving these models is often thought to require larger samples and more covariates (e.g. genomic and proteomic data). In this case it may be many years before the required datasets are widely available to researchers, and it is not clear which features will be sufficient to effect the desired personalisation.

Our notion of a sequence family, modelled by a multi-task dynamical system (MTDS), allows us to take a step towards personalisation without any additional data. This chapter presents an example of predicting patient response to an anaesthetic agent. We will use a MTDS to model the family of possible responses to the drug, and personalise the prediction over time according to the observed response. We can apply this method to any base model class, but our goal is to personalise the one that is currently used in clinical practice, with a view to improving predictions while maintaining trust by anaesthetists. The effective personalisation of such models presents the possibility of a control system which can safely induce and maintain anaesthesia automatically, at least in the absence of surgical stimuli.

We discuss the modelling background in Section 6.1, our proposed base model and MTDS variant in Section 6.2, and related work in Section 6.3. The experimental setup is detailed in Section 6.4 with results in Section 6.5. A conclusion is given in Section 6.6 followed in Section 6.7 by an appendix regarding the need for experimental design.

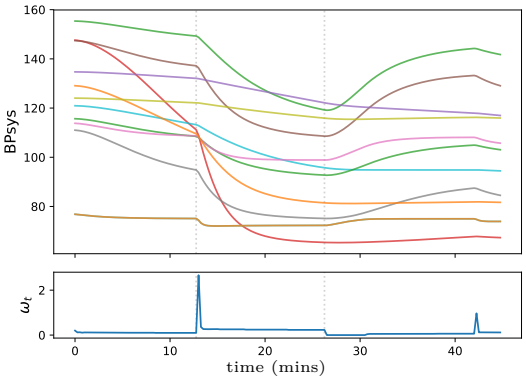


Figure 6-1: Differing systolic blood pressure (mmHg) responses to a drug infusion (mg/s) shown in the bottom panel. The responses follow individual PD models trained offline.

6.1 Background

In this section, we provide an introduction to the task in Section 6.1.1, an overview of the most common modelling paradigm (PK/PD models) and some current methods of personalisation (Section 6.1.2).

6.1.1 The modelling task

In order to sedate a patient, an anaesthetist initially targets a certain blood concentration of an anaesthetic agent. In the case of propofol, this may be between 2.5-5.0 μg/ml. The drug is administered via use of an intravenous infusion pump, which uses an internal model to provide the desired concentration. The response of the patient is quantified via vital signs, providing an important feedback loop to anaesthetists; in our case the vital signs are systolic and diastolic blood pressure and BIS[1], a measure of consciousness. BIS is known to be somewhat unreliable (see e.g. Lobo and Schraag, 2011; Schuller et al., 2015), but is nevertheless commonly used in practice, and perhaps the best quantitative proxy for consciousness in general use today.

The patient response to the drug infusion depends on their physiology, resulting in substantial inter-patient variation. Some examples of modelled responses to the same infusion sequence are shown in Figure 6-1 for systolic blood pressure; real data exhibit similar inter-patient differences. The initial dosage targets a BIS value in the range 40-60, but the vital signs must be monitored on a continual basis to ensure the patient stays within the therapeutic window. This task is made substantially harder due to the lag between dose and response.

[1]The Bispectral Index (BIS) of Myles et al. (2004) is a proprietary scalar-valued transformation of EEG signals which attempts to quantify the level of consiousness. BIS incorporates time-domain, frequency-domain, and bispectral analysis of the EEG to obtain a scalar between 0 (deep anaesthesia) and 100 (awake).

When a patient is in the operating theatre, the anaesthetist's job can be much more complex, requiring additional drugs and further adjustments in anticipation of and response to the various surgical operations on the patient. This is perhaps necessarily an ad-hoc procedure, and may require human attention for many years to come. The scope of our work here will be predicting patient response to a single drug in the absence of such external stimuli. This is an important first step towards a control system for steady state anaesthetic maintenance, with the potential to free up considerable time from practicing anaesthetists.

6.1.2 PK/PD models

The dominant paradigm for modelling drug response is the pharmacokinetic/pharmacodynamic (PK/PD) approach. This decomposes the problems into two sub-tasks:

- **Pharmacokinetics (PK)** is the study of drug concentrations in the tissues of the body. A PK model describes the distribution of a drug bolus[2] or infusion within an organism, including major blood vessels, under effects such as absorption, metabolism and elimination.
- **Pharmacodynamics (PD)** studies the relationship between drug concentration and physiological effect. The classical approach to pharmacodynamics assumes that drug molecules bind reversibly to receptors at some *effect site* in order to exert the desired effect.

There are a variety of reasons for modelling these effects separately, of which we mention two. Firstly, a drug is usually not administered at the required effect site, and we must model the lag induced by this discrepancy, as well as the elimination process. Secondly, PK mechanisms are better understood and the models more easily fitted (via blood samples); PD relationships are complex, highly variable and further obscured by disease progression and measurement difficulty (Lin, 2007, §3). The following sections discuss common modelling approaches for both PK (Section 6.1.2.1) and PD (Section 6.1.2.2) processes.

6.1.2.1 PK models

The most common PK model in anaesthesia is the three-compartment model (3CM). This is a vector-valued ordinary differential equation (ODE) describing drug concentrations for differently perfused physiological compartments under a continuous time infusion (see e.g. Bailey and Haddad, 2005). These compartments, denoted C_1, C_2, C_3, are sometimes interpreted as blood, muscle and fat, and can be conceptualised with the latter two as 'peripheral' compartments each connected only to the *central compartment*, blood. See Figure 6-2 for a pictorial representation.

[2] A bolus is the administration of a discrete amount of a drug, often given via an injection.

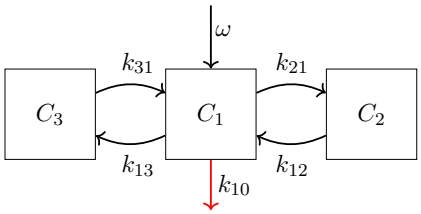


Figure 6-2: A three-compartment model for propofol pharmacokinetics.

The drug in-flow from a Target Controlled Infusion (TCI) pump, $\omega(t)$ enters the central compartment and the concentrations $\mathbf{c}(t) = (c_1(t), c_2(t), c_3(t))^\mathsf{T}$ of all compartments evolve as:

$$\frac{\mathrm{d}\mathbf{c}}{\mathrm{d}t} = A\mathbf{c}(t) + \mathbf{e}_1\omega(t) \tag{6.1a}$$

with

$$A = \begin{bmatrix} -(k_{10}+k_{12}+k_{13}) & k_{21} & k_{31} \\ k_{12} & -k_{21} & 0 \\ k_{13} & 0 & -k_{31} \end{bmatrix}, \qquad \mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. \tag{6.1b}$$

It is easy to see that when $\omega(t) = 0$, the total concentration $\sum_{j=1}^{3} c_j(t)$ is conserved iff $k_{10} = 0$. Hence $k_{10} \geq 0$ is called the 'elimination rate'. For a given set of parameters, Equation (6.1a) under a bolus infusion (delta function) can be solved analytically to yield a linear combination of three exponential decay functions.

Statistical modelling of pharmacokinetic effects has been an active field for over 50 years, dating back at least as far as Jelliffe (1967). While the interpretation of compartmental models may be disputed, the model is established in clinical practice; its continued use and development is indicative of its surprising effectiveness. These models are trusted by anaesthetists, and the rate constants appear now to be part of standard anaesthetic vocabulary.

6.1.2.2 PD models

Most PD models propose that the physiological effect can be determined directly (up to random noise) from the drug concentration at some effect site.[3] Let us denote this concentration as $x(t)$. The effect site (C_e) is assumed to be distinct from any of the PK compartments but connected to the central compartment (C_1); see Figure 6-3. In practice, the compartment C_e need not exist in any discrete physiological sense, but it is an important degree of freedom in fitting the data, allowing for a lag between $c_1(t)$ and the observed effect. This lag may be caused by multiple factors such as distribution time, receptor binding time and effects relating to intermediate substances – for further details,

[3] For a wider variety of PD models see the review in Mager et al. (2003).

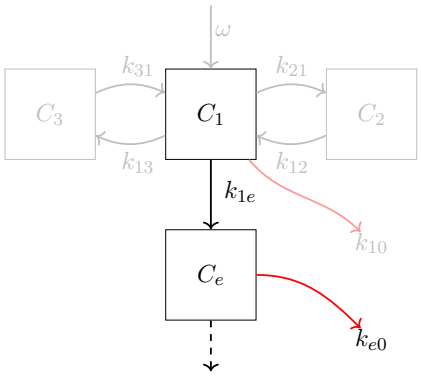


Figure 6-3: A PD model in the context of a PK/PD approach. The effect compartment C_e attaches to the central compartment of a PK model. The calculated concentration in C_e is assumed to be directly related to the pharmacodynamic effect.

see e.g. Holford (2018).

Denoting the rate of the in-flow to the effect site as k_{1e} and the elimination rate as k_{e0}, the effect site concentration $x(t)$ is modelled by the equilibrating process:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = k_{1e}c_1(t) - k_{e0}x(t); \tag{6.2}$$

i.e. the effect site concentration increases/decays exponentially towards that of the central compartment. The volume at the effect site is assumed to be negligible, so no adjustment is made to $c_1(t)$ due to the out-flow to $x(t)$. Where multiple effects are observed simultaneously (e.g. BPsys, BPdia, BIS), it is common to use one effect site per observation channel, resulting in effect site concentrations $x_j(t)$ for $j = 1, \ldots, n_y$.

The relationship of $\mathbf{x}(t)$ to the observations $\mathbf{y}(t)$ is usually modelled by some nonlinear transformation $g_{\boldsymbol{\eta}}$ plus white (Gaussian) noise, i.e. for a given time t,

$$y_j(t) = \mathcal{N}\left(g_{\boldsymbol{\eta}_j}(x_j(t)),\, \nu_j^{-1}\right) \tag{6.3}$$

for $j = 1, \ldots, n_y$ with parameters $\boldsymbol{\eta}, \boldsymbol{\nu}$. Most common choices of $g_{\boldsymbol{\eta}}(\cdot)$ are sigmoidal in nature and include the Hill function (Wagner, 1968) and generalised logistic sigmoid (Georgatzis et al., 2016). In practice, the assumption of white noise seems to be violated – one can observe correlated noise processes (such as artifacts, dropouts and unexplained deviations from steady-state) even in the relatively 'clean' research data.[4]

In practice, drugs act through an interplay of numerous factors, and it is not possible to analytically relate the drug effect to blood concentration or the number of bound receptors (see e.g. Bailey and Haddad, 2005, and refs. therein). Our focus on such PD models stems not from a strong belief in the existing paradigm, but a pragmatic concern to avoid

[4]The impact of correlated noise is reduced by our sampling rate of 1/15 Hz. There is relatively little current work which accounts for the noise processes mentioned above, but since such concerns are orthogonal to our interests in this Chapter, we leave this to future work.

substantial change from trusted models; better results may be possible with more flexible models.

6.1.2.3 Personalisation

The models discussed above take a pooled, or one-size-fits-all approach, and fail to take into consideration inter-individual variation. We provide here a very brief overview of existing work relating to personalisation.

PK models A number of studies have proposed 3CM models which depend on commonly available patient covariates such as age, gender, height or weight. These include Marsh et al. (1991); Schnider et al. (1998); White et al. (2008); see Eleveld et al. (2018) for a combined study. These models regress the rate constants in Equation (6.1) on the covariates, and are fitted using a nonlinear mixed effects approach[5] using the software package `NONMEM` (Sheiner and Beal, 1981). Details of the parameterisation proposed by these authors may be found in Appendix A.6.1. A number of validation studies exist (see e.g. Glen and Servin, 2009; Masui et al., 2010; Glen and White, 2014; Hüppe et al., 2019) which compare the predictive performance of such models in clinical practice. These studies have confirmed a degree of bias and inaccuracy of the models but overall their performance is considered by most clinicians to be adequate for clinical use (at least within the populations in which they were developed).

PD models There is comparatively little work to personalise the rate constants of the PD model. In most commercially available implementations of the Marsh and Schnider models, fixed k_{1e}, k_{e0} parameters are used, as well as constant parameters in the emission function $g_{\boldsymbol{\eta}}(\cdot)$. It is widely accepted by practicing anaesthetists that there is a significant amount of inter-individual variability in PD response to propofol which is largely unaccounted for in existing models. Eleveld et al. (2018) adjust the PD parameters based on age, but this results in less than 10% improvement compared to the original pooled model.

Discussion Unfortunately, personalisation based on patient covariates appears to be somewhat limited. Notwithstanding the many advances of the last 40 years, the following words of Hull (1979) retain some relevance:

> the clinical anaesthetist may ... be forgiven for feeling the sources of variation are so legion that in the case of many drugs used in anaesthetic practice it is better to titrate dose against response than to attempt to predict the correct dose on theoretical groups. He is, of course, right.

[5] A brief discussion of mixed-effects models and their relationship with MTL can be found in Section 2.3.3.

Forty years later, the retrospective study of Hüppe et al. (2019) suggests (even in the case of well-studied PK models) that existing covariates will not result in a MDAPE[6] below 20%. The metabolism of propofol is known to be subject to various factors for which data are unlikely to be available for the foreseeable future, such as individual genetic influence, alcohol consumption, and drug utilisation. A practical personalised model must hence make use of the only other available data: the observed response. The following section introduces our proposal for this using the MTDS.

6.2 Proposed model

In this section, we consider learning a *sequence family* of PD responses, conditioned on a propofol infusion sequence. In learning this sequence family (via an MTDS), we can adapt the predictions based on the posterior probability of possible responses. This automates the adjustment of model parameters as more observations are seen. The parameterisation of the base PD model is described in Section 6.2.1 followed by the MTDS version in Section 6.2.2 which enables online personalisation.

In this work, we choose not to personalise the PK component beyond the existing work of White et al. (2008). Our preliminary work showed that substantial gains were possible via effective personalisation of the PD model alone, and retaining the PK component results in a reduced change to current clinical practice. While we have judged this to be a favourable trade-off between performance and clinical confidence, we leave the evaluation of this decision to future work.

6.2.1 Base PD model

Our proposed base model is a relaxation of the PD model in Section 6.1.2.2 in discrete time. We assume access to the PK model prediction applied to each TCI infusion sequence. Specifically, let the inputs $\{u_t\}$ be the modelled central compartment discretised on the unit grid $t = 1, \ldots, T$, using the parameters of White et al. (2008). Let the effect site concentration at time t be denoted x_t, then Equation (6.2) may be discretised as:

$$x_t = \beta_1 x_{t-1} + \beta_2 u_{t-1} \tag{6.4}$$

for some β_1, β_2, with no loss of generality if $c_1(t)$ is constant in each interval $(t-1, t]$. These coefficients are related to the rate constants as:

$$\beta_1 = e^{-k_{e0}} \qquad \Rightarrow \beta_1 \in (0, 1) \tag{6.5}$$

$$\beta_2 = \frac{k_{1e}}{k_{e0}} \left(1 - e^{-k_{e0}}\right) \qquad \Rightarrow \beta_2 > 0 \tag{6.6}$$

[6] MDAPE is an initialism of median average prediction error.

since the rate constants are positive. These relationships were derived via use of Laplace transforms and the convolution theorem (see Appendix A.6.3.1 for further details). Since $\beta_1 \in (0, 1)$, the ARX(1) process in Equation (6.4) is stable and non-oscillating.

The nonlinear emission is modelled via a function $g_{\boldsymbol{\eta}}(\cdot)$ with parameter vector $\boldsymbol{\eta}$. We have found the choices in previous work (generalised sigmoid, Hill function) to be numerically unstable for gradient-based optimisation, or insufficiently flexible. Instead we use a basis of logistic sigmoid (σ) functions and express:

$$g_{\boldsymbol{\eta}}(x) \;=\; \sum_{r=1}^{L} \eta_r \, \sigma(\, a_r(x - b_r)), \tag{6.7}$$

with constants $a_r < 0$ and coefficients $\eta_r \geq 0$ for all r. These constraints enforce the desired monotonicity that as concentration increases, the observations (blood pressure etc.) are non-increasing. We have fitted an 8 dimensional basis with pre-selected constants $\{a_r, b_r\}_{r=1}^{8}$ chosen by optimising the fit to the learned generalised sigmoid functions used by Georgatzis et al. (2016). One may choose to interpret Equation (6.7) as a constrained 2-layer MLP with one input and one output, but its motivation is quite different.

To complete the model, we introduce additional parameters $\boldsymbol{\beta}_3$ and $\boldsymbol{\alpha}$ which provide personalised offsets to the values of the effect site dynamics and the emission respectively. These are degrees of freedom one might expect in a dynamical system, but are not present in the usual PD formulation. The full model for a given patient can be written with $\mathbf{x}_t \in \mathbb{R}^{n_y}$ and $\mathbf{y}_t \in \mathbb{R}^{n_y}$ as:

$$\mathbf{x}_t \;=\; \boldsymbol{\beta}_1 \odot \mathbf{x}_{t-1} \;+\; \boldsymbol{\beta}_2 u_t \tag{6.8a}$$

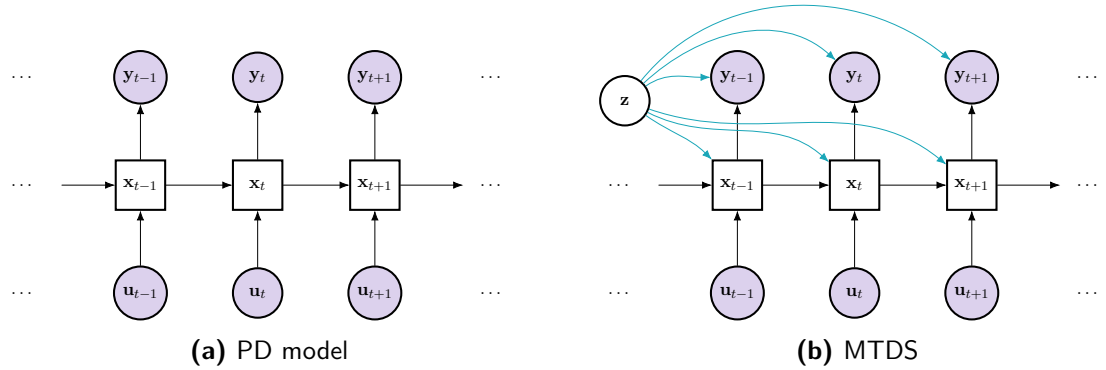
$$y_{tj} \;=\; g_{\boldsymbol{\eta}_j}(x_{tj} + \beta_{3j}) \;+\; \alpha_j \;+\; \epsilon_{tj}, \tag{6.8b}$$

$\epsilon_{tj} \sim \mathcal{N}\left(0,\, \nu_j^{-1}\right)$ for $j = 1, \ldots, n_y$ and $t = 1, \ldots, T$.

This results in a nonlinear deterministic state dynamical system (see Figure 6-4a), where each dimension is independent. A stochastic state might be considered as an extension to the standard PD approach, but preliminary investigation showed superior predictions with the deterministic approach (see also Appendix A.7). The parameters of the model are $\boldsymbol{\nu}$ and $\boldsymbol{\theta} = \{\boldsymbol{\alpha},\, \boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{n_y},\, \boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_{n_y}\} \in \mathbb{R}^d$, $d = 36$, while $\mathbf{a}, \mathbf{b}$ are constants. In principle, $\boldsymbol{\alpha}$ may be estimated prior to anaesthetic induction since it relates to pre-infusion patient-specific vitals levels.

6.2.2 MTDS model

We now discuss a version of eqs. (6.8) which can perform increasing personalisation of the model over time. A naïve approach would be to place uninformative priors over all dimensions of $\boldsymbol{\theta}$ and perform Bayesian inference online. We have previously called this the 'single task' (STL) approach, and it is deficient in at least two ways. Firstly, the inference



(a) PD model **(b)** MTDS

Figure 6-4: Graphical model of (a) the base PD model and (b) the MTDS variant for an individual patient; the independence of the channels is not shown. For clarity, the superscript i is omitted, and the dependence on $\mathbf{z}$ is shown in a different colour.

will be poorly conditioned and very expensive. Secondly, it fails to take advantage of the inductive bias available from the training data. The MTDS approach avoids both of these limitations; we describe its implementation below.

We assume that each patient i can be described by the PD model with a different $\boldsymbol{\theta}$. Let the parameters required by patient i be denoted as $\boldsymbol{\theta}^{(i)}$ corresponding to a latent code $\mathbf{z}^{(i)} \in \mathbb{R}^k$. The parameters for this patient are calculated from $\mathbf{z}^{(i)}$ via:

$$\boldsymbol{\theta}^{(i)} = [\boldsymbol{\alpha}^{(i)}, \boldsymbol{\beta}_1^{(i)}, \boldsymbol{\beta}_2^{(i)}, \boldsymbol{\beta}_3^{(i)}, \boldsymbol{\eta}_1^{(i)}, \ldots, \boldsymbol{\eta}_{n_y}^{(i)}] = \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z}^{(i)}), \tag{6.9}$$

where the latent code has the prior:

$$\mathbf{z}^{(i)} \sim \mathcal{N}(0, I_k), \tag{6.10}$$

with $\mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z}) = \mathbf{f}(\Phi\mathbf{z} + \mathbf{c})$ for some 'loading matrix' Φ, offset $\mathbf{c}$ and elementwise transformation function $\mathbf{f}$ (see below). The resulting parameters are used in the PD model in eqs. (6.8), and the resulting MTDS is shown as a graphical model in Figure 6-4b. The improvements over the STL approach are made possible by virtue of learning an appropriate Φ and $\mathbf{c}$ from the training data.

The function $\mathbf{f}$ consists of elementwise univariate transformations which ensure that each parameter satisfies the required constraints. For example, the unit interval constraints for $\boldsymbol{\beta}_1^{(i)}$ can be enforced via a logistic sigmoid, and the non-negativity constraints for $\boldsymbol{\beta}_2^{(i)}$ by $\text{softplus}(x) = \log(1 + e^x)$ etc. If parameters are unconstrained, no nonlinearity is applied. This choice of $\mathbf{h}_{\boldsymbol{\phi}}$ is a constrained version of the affine approach investigated in Chapters 4 and 5, which may result in a more interpretable latent code for clinical practice. The meaning of each dimension of $\mathbf{z}$ can be obtained via inspection of the matrix $\Phi \in \mathbb{R}^{d \times k}$.

We have formulated this model for an unknown $\mathbf{z}$ which is inferred over time. However, some information may be gleaned from covariates $\boldsymbol{\zeta}$ such as age, height, weight etc. To the extent that these covariates 'describe' the differences between patient responses, we

can set $\mathbf{z} \leftarrow \boldsymbol{\zeta}$ which we call a *task-descriptor* approach. In this case, test time inference is not required, but the model cannot adapt to the response. A hybrid approach is also possible, which performs inference only on a subset of the dimensions of $\mathbf{z}$.

6.3 Related work

There has been a variety of recent work considering personalised treatment response modelling. We provide some examples here, excluding deep learning methods, since we are considering data from clinical trials with small sample sizes. The work in this chapter uses the same dataset as Georgatzis et al. (2016) who demonstrate that relaxing the PK/PD model class to a SSM can result in improved model fit and in-sample prediction. Our work is orthogonal to this, as the goal is to demonstrate how to generalise or adapt to *out-of-sample* patients.

Multi-task GPs (MTGPs) have been proposed on a number of occasions for healthcare modelling. For instance, Dürichen et al. (2015) use MTGPs in the context of condition monitoring, but the multi-task application is with respect to observation channels, not patients. This is to be expected given our argument in Section 3.7 that MTGPs are limited to differing views of the same underlying phenomena. Alaa et al. (2018) consider a mixture of GPs, which can be applied to multiple patients. But here, personalisation is limited to use of covariates, restricted to a fixed set of subtypes, and not applicable for data with control inputs.

Schulam and Saria (2015) propose a form of generalised linear mixed effects (GLME) model, which assumes an *additive* decomposition of population, individual and (GP-based) noise components. This can adjust models to an individual, but only via linear coefficients and is unable to customise dynamical parameters. Some extensions are proposed in Xu et al. (2016); Futoma et al. (2016), but these limitations remain. The most similar work is perhaps that of Soleimani et al. (2017), who extend the GLME approach to include convolutions (or equivalently linear ODEs) of control inputs. Nevertheless, these still relate *additively* to the observations (exploiting the linearity of GPs); extensions to nonlinear dynamical systems (e.g. PK/PD models) are not straight-forward. Furthermore, no multi-task ideas are used; adaptivity is restricted to simple random effects.

6.4 Experimental setup

This section describes the experimental setup for the evaluation of our model. Section 6.4.1 describes the data, Section 6.4.2 describes the form of evaluation, and the model details and benchmarks are given in Section 6.4.3.

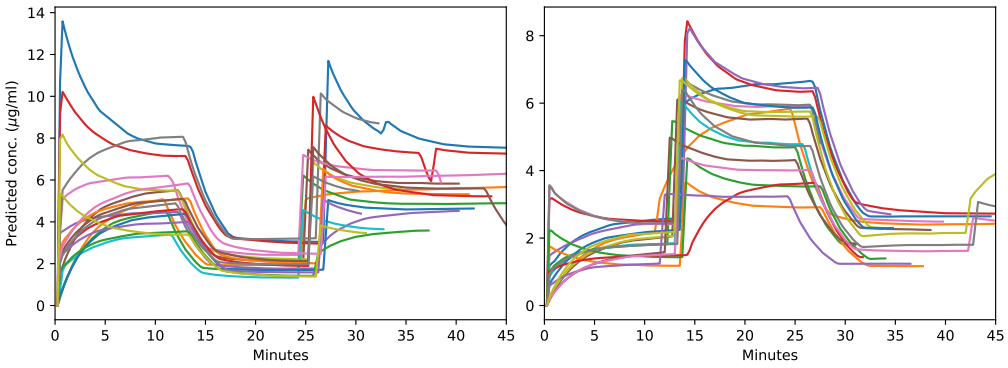


Figure 6-5: Predictions of White et al. (2008) PK model for each of the 40 patients, split by infusion schedule type: (*left*) high-low-high, (*right*) low-high-low. The inter-individual differences seen here are primarily due to the raw infusion chosen by the anaesthetist, rather than the modelled inter-patient variation.

6.4.1 Data

The data were obtained from an anaesthesia study carried out at the Golden Jubilee National Hospital in Glasgow, Scotland, as described in Georgatzis et al. (2016). These consist of $N = 40$ time series of Caucasian patients; the median length is 36 minutes (range approx. 27 - 50 mins) and the data are subsampled to 15 second intervals. Each patient was assigned to one of two pre-operative infusion schedules of propofol. The schedules split the time into three consecutive segments of 10-15 minutes each, within which the TCI pump targeted a high-low-high, or a low-high-low concentration of propofol. The two schedules are visualised in Figure 6-5 using the propofol concentration predicted via the White et al. (2008) model. Each patient has additional covariates of age, gender, height, weight (and BMI). Some pre-processing details are provided in Appendix A.6.2.

There are usually $n_y = 3$ channels comprising the vital signs: systolic blood pressure (BPsys), diastolic blood pressure (BPdia) and BIS[7]. BIS is the most informative metric for consciousness, but is only available for 27 of the patients. A variety of noise processes corrupt the underlying signal, including those derived from:

- anxiety prior to the operation,
- movement by the patient,
- clinical interventions,

as well as other unknown deviations. We leave the modelling of such features (e.g. via methods similar to Quinn et al., 2009) to future work.

An example time series is shown in Figure 6-6. The data for this patient consists of the three vital signs (top) and a drug infusion input (bottom). The schedule is indicated via the vertical dotted lines with the middle section targeting a higher propofol concentration. One can observe many of the discussed features here including lagged and nonlinear responses, and missing values.

[7]For the definition of BIS, see the footnote on page 107.

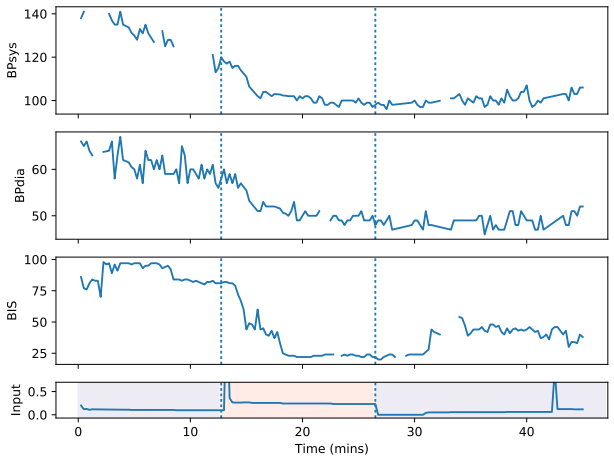


Figure 6-6: Vital signs BPsys, BPdia (mmHg), BIS, and drug infusion (mg/s) for an example patient

6.4.2 Evaluation

A clinically useful PK/PD model will be able to provide accurate predictions over a 5-10 minute forecast window; the response to a propofol bolus dose is typically evident within this time. We therefore evaluate the out-of-sample predictions of the MTDS and all competitor models (see below) within a 20 and 40-step ahead window (5, 10 mins). The performance is measured via Root Mean Squared Error[8] (RMSE) – we believe this to be more appropriate for personalised predictions than Mean Average Error (as used in various other studies such as Eleveld et al., 2018; we want to ensure strong penalisation of poor predictions for unusual patients). We report these scores both after 12 minutes and 24 minutes of observations, which we can use to understand if a model is improving over time.

We design the experiment in a leave-one-out (LOO) manner due to the relatively small number of patients (in machine learning terms). For each of 40 folds, a model is learned on 39 patients and tested on the held-out patient, and the results are averaged. During training, the RMSE is weighted such that each patient has equal contribution to the objective despite differing sequence lengths, to avoid a bias towards patients with longer sequences.

6.4.3 Model details

This section provides a review of the MTDS model and benchmarks used in our experiments. Section 6.4.3.1 describes the various PD models compared in our experiments, including the MTDS approach. An LSTM benchmark is described in Section 6.4.3.2. All models are fitted using stochastic gradient methods unless specified.

[8]More clinically relevant metrics, perhaps via assigning different penalties to different regions of BIS or BPsys/BPdia scores may be considered in future work.

Name	Parameters	Adaptive $\boldsymbol{\alpha}$	Details
Pooled	$\boldsymbol{\theta} = \boldsymbol{\theta}_0$	✗	One-size-fits-all model.
Pooled-α	$\boldsymbol{\theta} = \boldsymbol{\theta}_0$	✓	As above, but with adaptive $\boldsymbol{\alpha}$.
Task-ID	$\boldsymbol{\theta} = \mathbf{h}_{\boldsymbol{\phi}}(\boldsymbol{\zeta})$	✗	Customised using patient covariates.
Task-ID-α	$\boldsymbol{\theta} = \mathbf{h}_{\boldsymbol{\phi}}(\boldsymbol{\zeta})$	✓	As above, but with adaptive $\boldsymbol{\alpha}$.
MTPD-k	$\boldsymbol{\theta} = \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z})$, $\mathbf{z} \sim \mathcal{N}(0, I_k)$	✓	$k = 5, 7$ chosen by AIC, BIC on preliminary experiments.
Single Task	$\boldsymbol{\theta} \sim \mathcal{N}\left(0, 100^2 I_{33}\right)$	✓	(Relatively) uninformative Gaussian prior on all dimensions of $\boldsymbol{\theta}$.

Table 6.1: Versions of the pharmacodynamic model in considered in our experiments.

6.4.3.1 PD models

We consider a number of variants of the PD model described in Section 6.2.1. See Table 6.1 for an overview. The most basic benchmark will be a one-size-fits-all Pooled model, and a task-descriptor ('Task-ID') version. The Task-ID model adapts $\boldsymbol{\theta}$ from known covariates or 'task-descriptors' of the patients ($\boldsymbol{\zeta}$); this approach resulted in a small improvement on the training set, but regularisation of the parameters of $\mathbf{h}_{\boldsymbol{\phi}}$ was essential to to avoid poor performance on the validation set.[9] These models estimate parameters only via use of the training set, and perform no online adaptation. This provides a proxy for state-of-the-art models such as Jeleazcov et al. (2015); Eleveld et al. (2018). We also provide variants of these benchmarks which adapt the 'offset' or 'level' online, denoted 'Pooled-α' and 'Task-ID-α' respectively. This is motivated by the importance of the parameter $\boldsymbol{\alpha}$ in fitting patient data (see Figure 6-1).

The MTDS model is implemented according to eqs. (6.8) and (6.9) in Section 6.2. In order to choose k (the dimension of $\mathbf{z}$), preliminary experiments following the LOO protocol above were performed, and the values $k = 5$ and $k = 7$ were chosen by information criteria (AIC, BIC; for more details of the model selection see Appendix A.6.4.1). We refer to these two models as MTDS-k for $k = 5, 7$.[10] The single-task version of the PD model is the most flexible variant, and requires no learning. Instead, a relatively uninformative prior is placed on each parameter (parameters are constrained to their support via sigmoidal or softplus transformations where relevant, cf. Section 6.2.2.) Due to the difficulty of the inference problem, use of HMC (via Stan, Carpenter et al., 2016) proved essential here.

The MTDS model was trained via a MAP approximation of the objective in Section 3.4.1. These experiments were performed prior to the development of much of Chapter 3 and published in Bird et al. (2019). Later experiments using the variational criterion of Section 3.4.2 saw no significant improvement upon the published results; the procedural difference compared to previous chapters appears not to be especially important for this

[9] Regularisation hyperparameters were tuned on a validation set of size $N = 4$.

[10] The degrees of freedom of $\boldsymbol{\alpha}$ are not included in k, which is adapted separately from the MT parameters in order to compare all models like-for-like.

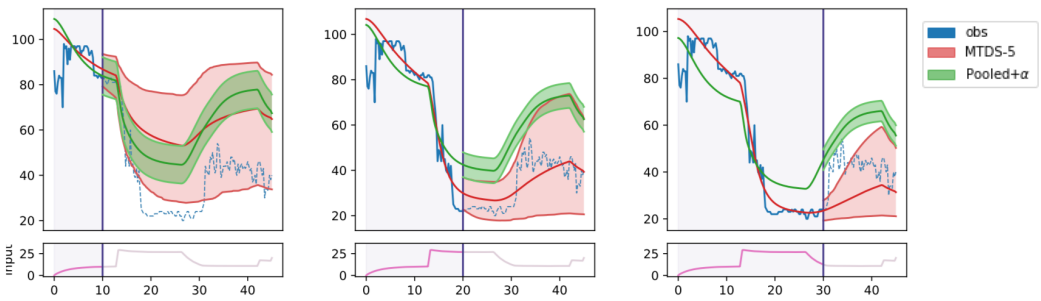


Figure 6-7: Example predictions (mean and 90% CI) for BIS channel of a patient at $t = 10, 20, 30$ minutes using Pooled and MTDS-5 models. The PK central compartment concentration (μg/ml) is shown in the bottom panel. Retrospective fits are shown without intervals for clarity.

set of tasks. Inference was performed via the approaches discussed in Section 3.5. Further details regarding learning and inference can be found in Appendix A.6.5.

6.4.3.2 LSTM Benchmark

It is unlikely that more complex/'neural' models such as RNNs will be accepted by practicing anaesthetists in the near future for a variety of reasons. The sample complexity of a RNN is poorly matched to the typical sample size of a clinical trial, predictions may perform very poorly under dataset shift (see Section 5.3.2), and the model is inscrutable, which precludes both an understanding of the prediction, and the ability to alter it. Nevertheless, it is still useful to provide a 'neural' benchmark to help us understand the opportunity cost of using simpler models. Note that if black box models are permissible, we might also expect improvements to RNNs using an MTDS approach (as in Chapter 5).

For the benchmark, we use a one-layer LSTM, with a hidden layer size of 32 and L2 regularisation coefficient 10^{-3} chosen by grid search from $\{16, 32, 128\} \times \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$, and fitted via use of the Adam optimiser. As in Chapter 5 we train the model in an an open-loop (or seq2seq) fashion, encoding 40 timesteps of inputs and outputs prior to a 40-step prediction. In each training iteration, the starting time t is randomised. Observations with missing values required transformation for the 'encoder' section of the LSTM: this were handled by zero-imputation, concatenated with a one-hot encoding of the pattern of missingness.

6.5 Results

The results are split into three sections. Firstly, an introduction (Section 6.5.1), which presents an example MTDS prediction over time, and provides the best possible results (in-sample) that can be achieved from our PD class. The main out-of-sample results for all models are then presented in Section 6.5.2. Finally Section 6.5.3 provides a closer look at the performance over time for three selected models.

	$t = 12$ m		$t = 24$ m	
	RMSE 20-ahead	RMSE 40-ahead	RMSE 20-ahead	RMSE 40-ahead
BPsys	5.40	5.31	5.43	5.19
BPdia	3.63	3.79	3.55	3.75
BIS	7.42	7.35	7.68	7.81

Table 6.2: Optimal PD fit: The 20 and 40-step predictive RMSE calculated *in-sample* for an individual model fitted to each patient after $t = 12, 24$.

6.5.1 Introduction

Example predictions from both the MTDS-5 and Pooled-α models can be seen in Figure 6-7 for the BIS channel. One can see the models adapting over time at $t = 10, 20, 30$ minutes, where the credible intervals show the predictive posterior for the *underlying PD function*. While the adaptive Pooled-α model (green) is fixed in shape and only updates its offset, the MTDS permits much greater flexibility, providing continual adaption over increasing t. The MTDS can further share information between the 3 channels via inference on $\mathbf{z}$, unlike the Pooled model. Further examples can be seen in Appendix A.6.6.

The best performance that the PD model class described in Section 6.2.1 can achieve (for this dataset) is shown in Table 6.2. This is the average *in-sample* error for 20-step and 40-step ahead predictions at $t = 12, 24$ minutes for each channel trained on a *per-patient* basis. The RMSE of the BIS channel is relatively high, in part due to noise processes, but also since the PD model class is insufficiently flexible for this channel.[11] For an example of BIS violating the PD model assumptions, see the step-change in the BIS channel observations at $t = 30$ in Figure 6-7 which may perhaps correspond to a phase change in patient state (see e.g. Mukamel et al. 2014) or some form of hysteresis.

6.5.2 LOO results for all models

The *out-of-sample* performance (LOO average) relative to the optimal performance (per Table 6.2) is shown in Table 6.3 for all models. This is reported as Standardised RMSE (SRMSE), which is a ratio of the model RMSE to the optimal fit in Table 6.2. A value of 1.00 indicates the same level of performance as the optimal PD fit, and a higher value indicates a worse fit. Results for the three channels are listed separately. One can see that the non-adaptive approaches often proposed in the literature (both Pooled and Task-ID) are highly suboptimal; for the blood pressure channels, the RMSE is more than twice that of the optimal fit.

By $t = 24$ there is a clear win for the MTDS models over all other PD approaches. The

[11] BIS is known not to follow PD assumptions: it is a composition of many signals and suffers from the difficulty of defining consciousness (especially as a scalar value); see e.g. Lobo and Schraag (2011); Schuller et al. (2015).

20-step performance is essentially optimal for the BPsys and BPdia channels, and further, the performance is substantially better than the flexible black-box LSTM model. For BIS, an optimal performance is not achieved by any approach within the class of PD models, but the MTDS results are a promising step forward (for the LSTM results, see below). For a discussion of the important degrees of freedom and a visualisation of a learned Φ, see Appendix A.6.6.2.

The STL approach performs relatively poorly for all channels, and takes 5 - 10× longer for inference, although its performance is expected to improve with increasing t. The adaptive Pooled-α model performs better initially, but it is not flexible enough to provide much customisation, and shows no improvement over time. Adapting the offset $\boldsymbol{\alpha}$ of the Pooled-α model nevertheless allows a substantial improvement over the non-adaptive version; the level is among the most important degrees of freedom (see also Figure 6-1). The performance of the patient covariate model Task-ID (both adaptive and non) is similar to or worse than the Pooled version. While Task-ID models clearly suffer from overfitting, the in-sample improvement is fairly small, suggesting the available patient covariates may lack the required information to attain meaningful improvement.

There appears to be some performance advantage for using the LSTM model for BIS (although the MTDS is able to reduce the gap by $t = 24$). This may be due to the particularly serious mis-specification of the PD model for this channel, as mentioned above. The SRMSE of 0.88 for the BIS prediction at $t = 24$ indicates that the LSTM can fit the data better than the *in-sample* PD model. In particular, the 1-d linear dynamics of the PD model cannot fit the stepped level changes and hysteresis sometimes observed. The latter results in correlated noise to the fit, which further misleads online inference for the MTDS.

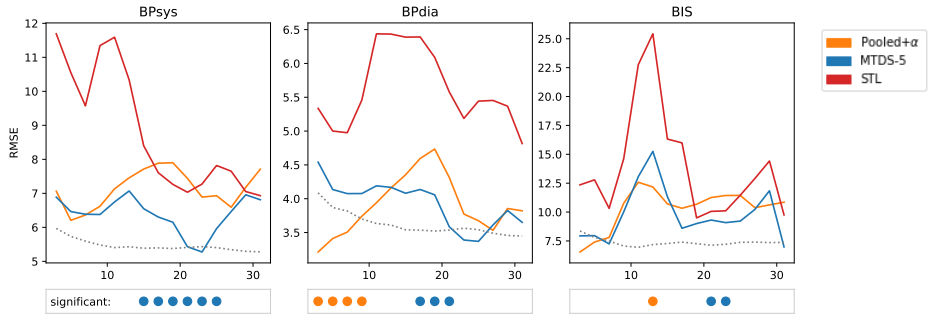
6.5.3 Improvement over time

The average 20-step-ahead predictive RMSE for $t = 3, \ldots, 31$ is provided in Figure 6-8a for the Pooled-α, MTDS-5 and STL models. The optimal PD performance is shown by the dotted black line for each channel, and the coloured dots under the graph denote where the respective model is significantly better than its competitors[12]. The benefit of the MTDS over the STL approach is sustained across all time points for the given period, although the STL model appears to be 'closing the gap' for BPsys near the 30 minute mark. We see that the advantage of the MTDS-5 over the Pooled-α model opens up after $t \geq 13$ minutes for the BPsys and BPdia channels, but that the error has some dependence on time, notably at 12-14 and 27-29 minutes, which is when the infusion sequences change (see Figure 6-5). Section 6.7 argues that this performance degradation is in part inevitable, since multiple changepoints are required to determine the required model parameters, motivating improved experimental design of the $\{u_t\}$.

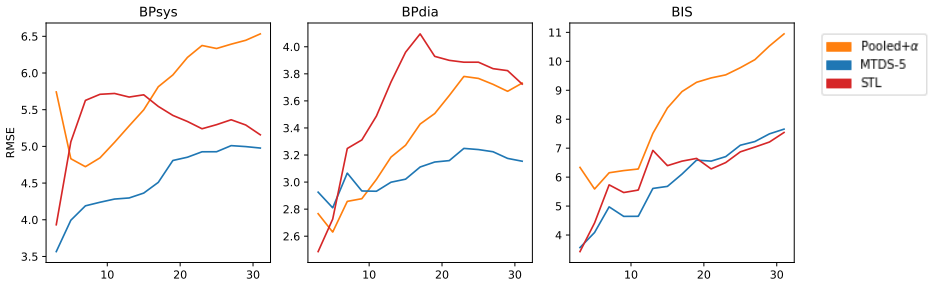
[12] Significance is at the $\alpha = 0.05$ level using a nonparametric test (Wilcoxon signed-rank test, see Scheff, 2016, §8). Note that no adjustment has been made for multiple testing, e.g. via a Bonferroni correction.

		$t = 12$ m		$t = 24$ m	
		SRMSE 20-ahead	SRMSE 40-ahead	SRMSE 20-ahead	SRMSE 40-ahead
BPsys	Pooled	2.86	2.65	3.25	3.11
	Task-ID	2.84	2.64	3.23	3.10
	Pooled+α	1.23	1.31	1.37	1.41
	Task-ID+α	1.26	1.37	1.55	1.67
	STL	2.10	2.29	1.29	1.60
	MTDS-5	**1.18**	**1.30**	**1.00**	1.15
	MTDS-7	1.19	1.31	**1.00**	**1.12**
	LSTM (open loop)	1.59	1.76	1.46	1.48
BPdia	Pooled	2.42	2.37	2.59	2.35
	Task-ID	2.40	2.35	2.57	2.34
	Pooled+α	**1.03**	**1.13**	1.21	1.15
	Task-ID+α	1.16	1.30	1.23	1.24
	STL	1.50	1.74	1.57	1.58
	MTDS-5	1.12	1.24	**1.01**	1.03
	MTDS-7	1.15	1.25	1.02	**0.99**
	LSTM (open loop)	1.61	1.80	1.26	1.41
BIS	Pooled	1.66	1.87	2.08	1.82
	Task-ID	1.69	1.90	2.11	1.84
	Pooled+α	1.45	**1.67**	1.47	1.49
	Task-ID+α	1.66	2.01	1.65	1.69
	STL	1.97	3.16	1.31	1.61
	MTDS-5	1.35	1.81	1.21	1.29
	MTDS-7	**1.32**	1.85	**1.19**	**1.28**
	LSTM (open loop)	1.19	1.55	0.88	1.28

Table 6.3: The out-of-sample performance of each model standardised by the optimal PD results in Table 6.2 (lower is better). For each channel, models are split into standard approaches, adaptive (MTDS) approaches, and non-PD approaches. As per Table 6.2, the results give the 20 and 40-step predictive RMSE after $t = 12, 24$. The results highlighted in bold are the best performance from a PD model.



(a) 20-step ahead predictive RMSE



(b) Retrospective RMSE (all preceding datapoints)

Figure 6-8: Performance over time (in mins). In 6-8a, the lower dotted line indicates the avg. error for the retrospective fit of the optimal PD function over each task. If the MTL or Pooled model is significantly better than the other (see text), it is shown as the relevant coloured dot below. Best viewed in colour.

Figure 6-8b shows the retrospective RMSE as a function of time for the different models, based on "post-dicting" the the data seen up to time t given the inferences for $\mathbf{z}$ at time t. These plots show that the MTDS-5 model is substantially better at this task than the Pooled-α and STL models for both BP channels. The fact that the Pooled-α model does not fit well retrospectively indicates that it cannot capture the inter-patient variation well, and hence little performance improvement is expected over time.

6.6 Conclusion

The application of the MTDS framework to PK/PD modelling problem provides a novel approach to personalised medicine, which (for our propofol dataset) shows substantial promise over traditional approaches using patient covariates. Clinicians often perform mental adjustments of models, titrating the dose to the response. The MTDS facilitates the automation of this practice. This may be a necessary direction for PK/PD models for many years to come, since common patient covariates appear to have insufficient information for full personalisation as noted by Hüppe et al. (2019) for PK models, and observed in our own results for PD models.

The experiments have highlighted a number of areas for further improvement. Firstly, the BIS channel may benefit from a more flexible PD model class. This can be seen from the

relatively poor performance of optimal PD fits, visual inspection, and comparison with LSTM models. Possible directions include a higher dimensional state, nonlinear dynamics, and/or switching latent state variables, but the design should be carefully constructed alongside domain experts. Secondly, model predictions can likely be improved via better experimental design of the infusion sequence, and avoid the performance degradation at 27-29 minutes (see also Section 6.7 below). Finally, incorporating artefact models should reduce the sensitivity of the inference to correlated noise and artefacts, reducing suboptimal predictions.

6.7 Appendix: The need for experimental design

In this section we show that the information required to predict the response after the second changepoint ($t \approx 27$ minutes for this dataset) may not be contained in the observations preceding it. This suggests the need for a careful design of the infusion sequence in the pre-operative window to extract maximum information.

Consider the following nonlinear dynamical system, which is a generalisation of the PD model:

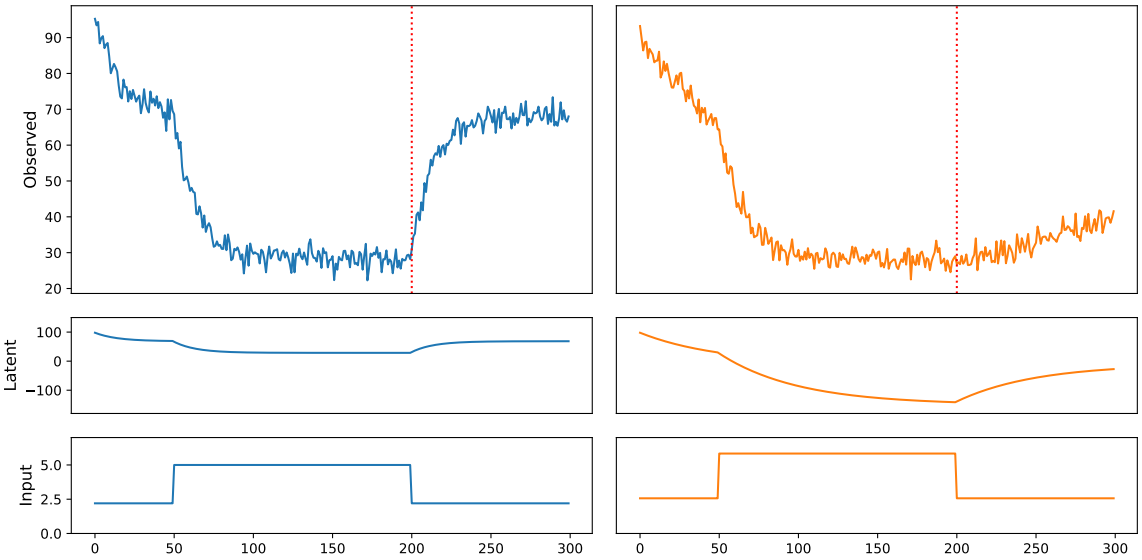
$$\mathbf{x}_t = A\mathbf{x}_{t-1} + Bu_t \tag{6.11a}$$

$$y_{tj} = g_{\boldsymbol{\eta}_j}(x_{tj} + \beta_j) + \alpha_j + \epsilon_{tj}, \tag{6.11b}$$

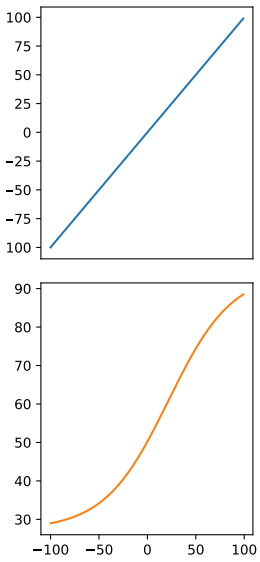
$\epsilon_{tj} \sim \mathcal{N}\left(0, \nu_j^{-1}\right)$ for $t = 1, \ldots, T$ and $j = 1, \ldots, n_y$. It is well-known that state space models are non-identifiable (see Section 2.2.3.2). In the case of eqs. (6.11), it is easy to construct an example where two different parameter settings can generate the same observations to time t but result in quite divergent forecasts.

We provide an example of this phenomenon in Figure 6-9a, with inputs $u_t \in \mathbb{R}$ (bottom), state $x_t \in \mathbb{R}$ (middle), and outputs $y_t \in \mathbb{R}$ (top). The emission functions $g_{\boldsymbol{\eta}}$ are shown in Figure 6-9b; the first is linear, the second sigmoidal. The observed y_t's for $t \leq 200$ are almost identical up to the noise process, but after the changepoint at $t = 200$, the systems evolve very differently. This is because the nonlinear function obscures the different latent states. The first system (blue) in Figure 6-9a converges to the low value of $y_t \approx 30$ via the AR process; the second system (orange) converges to $y_t \approx 30$ due to the sigmoidal $g_{\boldsymbol{\eta}}$. The best prediction we can make when x_t is unknown is shown in red in Figure 6-9c (although this will depend on the prior probability of each case).

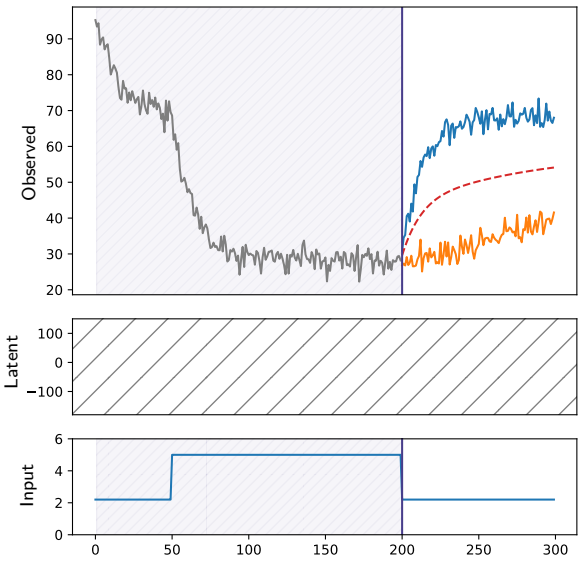
A similar property exists in the propofol dataset when the vital signs have converged to some level for the first time. We do not know whether the steady-state response is finely balanced, or whether the response is 'saturated', i.e. the drug infusion exceeds the amount needed to keep the patient in the current state. This ambiguity can only be resolved via observing a further changepoint in the u_t's. For our data, this changepoint happens at $t \approx 28$ and results in suboptimal predictions at this time. We emphasise this is related



(a) Two sequences generated from different dynamical systems. (*top*) observations $\{y_t\}$; (*middle*) latent state $\{x_t\}$; (*bottom*) input $\{u_t\}$.



(b) The nonlinear emission functions corr. to panel (a).



(c) Possible forecasts after $t = 200$ when system parameters and latents are unknown.

Figure 6-9: Synthetic data generated from two different dynamical systems. See text for more details.

not to the absolute value of t, but rather the number of changepoints that have been seen. Changepoints in the infusion sequence must therefore be carefully planned in order to gain the maximum information while minimising the amount of time t required to obtain it. We suggest that future work employs the use of optimal experimental design (see e.g. Huan and Marzouk, 2013) in order to improve the accuracy of forecasts for smaller values of t.

CHAPTER 7

Conclusion

In this thesis, we have shown how to construct, train, and perform inference in a new class of time series model: the multi-task dynamical system. Through empirical work we have shown that this hierarchical model can learn an embedding of sequence characteristics, which can be used to improve performance and/or modulate time series forecasts depending on the goals of an end user. In this work we have successfully applied the MTDS to linear dynamical systems, recurrent neural networks, and standard pharmacodynamic models, and investigated the concomitant benefits.

We have claimed that time series data often arise in 'families' of related sequences, and Chapters 4-6 have investigated some examples of this. However, almost all datasets which contain multiple time series manifest such inter-sequence differences. We can find further examples in the context of natural language or video data; other examples might regularly be found in a business context, albeit rarely in the public domain. Given the truth of this claim, predicting time series data accurately must require task inference (in the sense of this thesis), unless one can use a separate model for each time series. The only existing models which appear to perform task inference are neural models (such as RNNs), which perform this implicitly. Our MTDS approach provides an explicit alternative to the task inference in RNNs, and an explicit approach may in many circumstances be preferred. A discussion of these circumstances is provided in Section 7.1; possible further extensions to the model are discussed in Section 7.2.

7.1 When to use a MTDS

We have provided a fairly comprehensive demonstration of the properties and benefits of the MTDS in Chapters 4-6. The MTDS has been shown to yield improvements in predictive accuracy in limited data scenarios, but standard GRUs have sometimes shown comparable or better performance when the amount of data is larger. Due to the increased

Dataset size	In-sample sequences small	In-sample sequences large	Novel sequences small	Novel sequences large
vs SSMs	↑	↑	↑	↑
vs RNNs	↑	↓	↑	↑/↓

Table 7.1: Expected performance benefit of using an MTDS compared to both SSMs and RNNs, where ↑ indicates improved performance for an MTDS approach, and ↓ indicates worse performance.

complexity of training and using MTDS models, it may be helpful to summarise the situations which appear to benefit from an MTDS approach. We first address predictive accuracy, which is often the key consideration for academic work. However, there are a variety of other reasons that an MTDS may be of use in practice such as interpretability or control, which are also discussed below.

Predictive accuracy A qualitative summary of the situations where the MTDS can be expected to yield improvement is shown in Table 7.1. The table summarises the performance seen in our experimental work, split by dataset size (small and large datasets), and whether the predictions are for sequences in the training set, or entirely novel sequences. If one is restricted to using SSMs (e.g. for reasons of interpretability), one can usually expect a MTDS approach to perform better, or at least not worse than a SSM. This has been the case both in Chapter 4 and Chapter 6. In the case where a black-box model such as a GRU is permitted, a MTDS approach will likely be beneficial under limited data; but has the potential to result in reduced accuracy with larger datasets. However, the MTDS is more robust to dataset shift, and may perform better on novel out-of-sample data.

Interpretable modelling In a variety of applications, such as healthcare, science or business, interpretable models (such as state space models) are very important. These not only provide a window into the process underlying the data, but also allow domain knowledge and constraints to be built into the model. The MTDS approach allows improved performance of such models and may be considered a much more favourable trade-off between interpretability and performance than a black-box approach.

Low-dimensional representation Sequence data is (in principle) of unbounded dimension, and collections of sequence data are often not aligned in time. Standard dimensionality reduction techniques will cope poorly with these problems. Furthermore, standard approaches such as PCA focus on high variance discrepancy in *data space* such as differences in magnitude and struggle to incorporate control inputs $\mathbf{u}_{1:T}$. The MTDS approach in contrast learns a *generative model* of the sequence, capturing an estimate of the true degrees of freedom. This avoids the above problems, resulting in a low dimensional representation which captures intuitive qualitative differences between sequences.

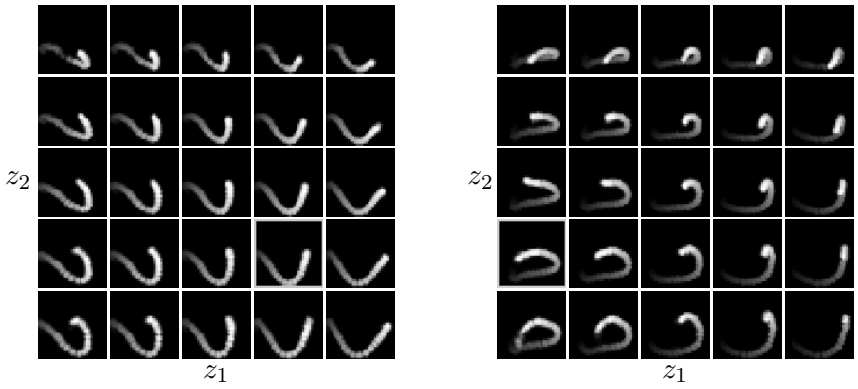


Figure 7-1: Effect of latent adjustment on generation of double pendulum data. The two panels represent two different initial conditions, within which the predictions for different values of (z_1, z_2) are shown. The increasing brightness shows the flow of time; only the second bob is shown for clarity.

Controlling long-term predictions Animators require fine-grained control over models, which is not possible with most existing approaches. The MTDS supports a large number of different possible roll-outs which are consistent with a sequence so far. An example is given in Figure 7-1 of a video generation model of a double pendulum, where the trajectory is modulated by the latent variable $\mathbf{z}$. The two panels shows two different initial conditions; the horizontal and vertical positions within each panel correspond to the value of z_1 and z_2 respectively. z_1 relates to the relative kinetic energy between the bobs, and z_2 corresponds to the total energy of the system. Other domains may also benefit from this control, especially where domain knowledge can be used in order to determine the value of $\mathbf{z}$.

7.2 Future work

There are many extensions available to the work described in this thesis. The MTDS already incorporates ideas from fields including MTL, (deep) generative models, time series models, recurrent neural networks, variational inference, Monte Carlo integration; and the applications have required further background still. The following suggestions are ideas which we have been unable to explore due to time constraints, as well as avoiding a further increase in scope.

Stochastic state models Stochastic state models have been avoided in this work for two reasons: firstly because the available datasets did not demand it, and secondly since long term predictions appear to benefit from the deterministic state (see Section 2.2.3.6 and Appendix A.7 for further discussion). Additional challenges for learning and inference exist for stochastic state MTDSs, since the latent dynamical state must be integrated out. This is less problematic for LDS models, which allow closed-form integration, and

this benefit may in principle be extended to nonlinear models too as in e.g. Karl et al. (2017). More generally, Sequential Monte Carlo (SMC, e.g. Doucet et al., 2001) might be considered for approximate integration. Stochastic state models may be particularly important for anomaly detection tasks; the MTDS allows us to tailor the model to an individual person, organisation, hospital, machine etc. which may reduce false positives.

Dynamic customisation In this thesis, we have assumed that each time series has a customisation which remains static in time. This is not necessarily the case: for instance, mocap data can comprise a variety of styles and activities over time, or the patient response to a drug infusion may undergo a phase change. This might be handled by a time-dependent $\mathbf{z}_t$ which varies smoothly over time, or makes use of a switching model. One could use this (for instance) to perform activity recognition for mocap data in a fully unsupervised manner.

'Closed loop' amortised inference There are many advantages to the Monte Carlo inference routine proposed in this thesis. Most importantly, we are guaranteed that the inference routine cannot overfit, and it is robust to dataset shift. On the other hand, real-world applications often require faster response times than are possible using a MC approach. This may be facilitated by an amortised approach, but use of standard encoders (such as RNNs, as discussed in Chapter 3) may perform poorly online and suffer from overfitting to the training data. While standard amortised approaches use an open-loop discriminative approach (i.e. they are 'blind' to the appropriateness of their output), one might consider a *closed-loop* variant. For instance, one could 'learn' an optimiser (cf. Andrychowicz et al., 2016) for accelerating standard variational inference as per Marino et al. (2018), making use of gradient and loss information.

Lifelong learning The MTDS is necessarily limited by the sequence family $\mathcal{H}$ that is learned from the training set. Where $\mathcal{H}$ does not capture the characteristics of a new test sequence, the MTDS is unable to extrapolate effectively. From Equation (3.18), we have for a given test sequence $\mathbf{y}'_{1:t}$ with inputs $\mathbf{u}'_{1:t}$:

$$\log p(\mathbf{y}'_{1:t} \,|\, \mathbf{u}'_{1:t},\, \boldsymbol{\phi}) \;=\; \log \int_{\mathcal{Z}} p(\mathbf{y}'_{1:t} | \mathbf{u}'_{1:t},\, \mathbf{h}_{\boldsymbol{\phi}}(\mathbf{z}))\, p(\mathbf{z})\, \mathrm{d}\mathbf{z}. \tag{7.1}$$

If this log probability falls below some threshold, we might consider updating the parameters $\boldsymbol{\phi}$ to provide a better prediction. However, to avoid catastrophic forgetting (McCloskey and Cohen, 1989), it will be important to ensure that the training set sequences retain a high log likelihood. This might be achieved by taking gradient steps in the direction of maximising Equation (7.1), while also maximising the log likelihood of samples (or 'hallucinations') generated from the existing model (as per e.g. Shin et al., 2017).

MTPD model extensions We have seen in Chapter 6 that the existing pharmacodynamic (PD) model class is unable to model various aspects of the observations such as phase changes or hysteresis. This suggests expanding the model class to incorporate a larger, stochastic, and possibly nonlinear latent state. This must be constructed with domain experts in order to retain interpretability where possible, as well as obey physiological constraints. In order to avoid being misled by artefacts in the observations, integrating the model with the Factorial Switching Linear Dynamical System (FSLDS, Quinn et al., 2009) may yield substantial improvements, especially in test time inference. Finally, results of this model may be substantially improved by better experimental design, as argued in the conclusion of Chapter 6.

Bibliography

O. Aguilar and M. West. Analysis of Hospital Quality Monitors Using Hierarchical Time Series Models. In *Case Studies in Bayesian Statistics*, pages 287–302. Springer, 1999.

A. M. Alaa, J. Yoon, S. Hu, and M. van der Schaar. Personalized Risk Scoring for Critical Care Prognosis using Mixtures of Gaussian Processes. *IEEE Transactions on Biomedical Engineering*, 65(1):207–218, 2018.

M. A. Álvarez, L. Rosasco, and N. D. Lawrence. Kernels for Vector-Valued Functions: A Review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012.

R. K. Ando and T. Zhang. A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. *Journal of Machine Learning Research*, 6(Nov):1817–1853, 2005.

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas. Learning to Learn by Gradient Descent by Gradient Descent. In *Advances in Neural Information Processing Systems 30*, pages 3981–3989, 2016.

A. Argyriou, T. Evgeniou, and M. Pontil. Multi-Task Feature Learning. In *Advances in Neural Information Processing Systems 21*, pages 41–48. MIT Press, 2007.

K. J. Aström and R. M. Murray. *Feedback Systems: an Introduction for Scientists and Engineers*. Princeton University Press, 2010.

D. Aubin and A. D. Dalmedico. Writing the History of Dynamical Systems and Chaos: Longue Durée and Revolution, Disciplines and Cultures. *Historia Mathematica*, 29(3):273–339, 2002.

M. Auger-Méthé, C. Field, C. M. Albertsen, A. E. Derocher, M. A. Lewis, I. D. Jonsen, and J. M. Flemming. State-Space Models' Dirty Little Secrets: Even Simple Linear Gaussian Models Can Have Estimation Problems. *Scientific Reports*, 6:26677, 2016.

J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer Normalization. *CoRR*, 2016.

N. Bacaër. *A Short History of Mathematical Population Dynamics*. Springer Science & Business Media, 2011.

D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations*, 2015.

J. M. Bailey and W. M. Haddad. Drug Dosing Control in Clinical Pharmacology. *IEEE Control Systems*, 25(2):35–51, 2005.

B. Bakker and T. Heskes. Task Clustering and Gating for Bayesian Multitask Learning. *Journal of Machine Learning Research*, 4(May):83–99, 2003.

L. Balles and P. Hennig. Dissecting Adam: The Sign, Magnitude and Variance of Stochastic Gradients. In *Proceedings of the 35th International Conference on Machine Learning*, pages 413–422, 2018.

D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

J. Baxter. A Model of Inductive Bias Learning. *Journal of Artificial Intelligence Research*, 12: 149–198, 2000.

J. Bayer and C. Osendorfer. Learning Stochastic Recurrent Networks. *arXiv preprint arXiv:1411.7610*, 2014.

S. Ben-David and R. Schuller. Exploiting Task Relatedness for Multiple Task Learning. In *Learning Theory and Kernel Machines*, pages 567–580. Springer, 2003.

S. Bengio, Y. Bengio, J. Cloutier, and J. Gecsei. On the Optimization of a Synaptic Learning Rule. In *Optimality in Artificial and Biological Neural Networks*, pages 6–8, 1992.

S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In *Advances in Neural Information Processing Systems 29*, pages 1171–1179, 2015.

Y. Bengio, P. Simard, and P. Frasconi. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

J. O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer Science & Business Media, 2013.

J. M. Bernardo and A. F. Smith. *Bayesian Theory*, volume 405. John Wiley & Sons, 2009.

A. Bird and C. K. I. Williams. Customizing Sequence Generation with Multi-Task Dynamical Systems. *arXiv preprint arXiv:1607.06450*, 2020.

A. Bird, C. K. I. Williams, and C. Hawthorne. Multi-Task Time Series Analysis applied to Drug Response Modelling. In *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019.

C. M. Bishop. *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer, 2007.

C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: The Generative Topographic Mapping. *Neural Computation*, 10(1):215–234, 1998.

E. Bonilla, K. M. A. Chai, and C. K. I. Williams. Multi-task Gaussian Process Prediction. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 21*. MIT Press, Cambridge, MA, 2008.

S. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating Sentences from a Continuous Space. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning (CoNLL)*, 2016.

G. E. Box, G. M. Jenkins, and G. C. Reinsel. Time Series Analysis: Forecasting and Control. 2008.

P. J. Brockwell, R. A. Davis, and S. E. Fienberg. *Time Series: Theory and Methods*. Springer Science & Business Media, 1991.

P. J. Brockwell, R. A. Davis, and M. V. Calder. *Introduction to Time Series and Forecasting*, volume 2. Springer, 2002.

Y. Burda, R. Grosse, and R. Salakhutdinov. Importance Weighted Autoencoders. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

O. Cappé, R. Douc, A. Guillin, J.-M. Marin, and C. P. Robert. Adaptive Importance Sampling in General Mixture Classes. *Statistics and Computing*, 18(4):447–459, 2008.

B. Carpenter, A. Gelman, M. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. A. Brubaker, J. Guo, P. Li, A. Riddell, et al. Stan: A Probabilistic Programming Language. *Journal of Statistical Software*, 20(2):1–37, 2016.

R. Caruana. Multitask Learning: A Knowledge-Based Source of Inductive Bias. In *Proceedings of the 10th International Conference on Machine Learning (ICML)*, pages 41–48, 1993.

R. Caruana. *Multitask Learning*. PhD thesis, Carnegie Mellon University, 1998.

G. Casella. An Introduction to Empirical Bayes Data Analysis. *The American Statistician*, 39(2): 83–87, 1985.

C.-B. Chang and M. Athans. State Estimation for Discrete Systems with Switching Parameters. *IEEE Transactions on Aerospace and Electronic Systems*, 1978.

N. Chapados. Effective Bayesian Modeling of Groups of Related Count Time Series. In *International Conference on Machine Learning*, 2014.

T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems 31*, 2018.

X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. Variational Lossy Autoencoder. In *Proceedings of the International Conference on Learning Representations*, 2017.

Z. Chen and B. Liu. Lifelong Machine Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 10(3):1–145, 2016.

S. Chiappa and D. Barber. Output Grouping using Dirichlet Mixtures of Linear Gaussian State-Space Models. In *2007 5th International Symposium on Image and Signal Processing and Analysis*. IEEE, 2007.

S. Chiappa, S. Racaniere, D. Wierstra, and S. Mohamed. Recurrent Environment Simulators. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina, et al. State-of-the-Art Speech Recognition with Sequence-to-Sequence Models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

K. Cho. Natural Language Understanding with Distributed Representation. *arXiv preprint arXiv:1511.07916*, 2015.

K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

N. Chopin. A Sequential Particle Filter Method for Static Models. *Biometrika*, 89(3):539–552, 2002.

N. Chopin, P. E. Jacob, and O. Papaspiliopoulos. SMC2: an Efficient Algorithm for Sequential Analysis of State Space Models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3):397–426, 2013.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In *NIPS 2014 Workshop on Deep Learning*, 2014.

J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio. A Recurrent Latent Variable Model for Sequential Data. In *Advances in Neural Information Processing Systems 29*, pages 2980–2988, 2015.

G. Claeskens, N. L. Hjort, et al. *Model Selection and Model Averaging*. Cambridge University Press, 2008.

J.-M. Corneuet, J.-M. Marin, A. Mira, and C. P. Robert. Adaptive Multiple Importance Sampling. *Scandinavian Journal of Statistics*, 39(4):798–812, 2012.

M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. Lagrangian Neural Networks. *arXiv preprint arXiv:2003.04630*, 2020.

Y. N. Dauphin and D. Grangier. Predicting Distributions with Linearizing Belief Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

F. De la Torre, J. Hodgins, A. Bargteil, X. Martin, J. Macey, A. Collado, and P. Beltran. Guide to the Carnegie Mellon University Multimodal Activity (CMU-MMAC) Database. 2009.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1): 1–22, 1977.

B. Dennis, J. M. Ponciano, S. R. Lele, M. L. Taper, and D. F. Staples. Estimating Density Dependence, Process Noise, and Observation Error. *Ecological Monographs*, 76(3):323–341, 2006.

W. Dent and A. Min. A Monte Carlo study of Autoregressive Integrated Moving Average Processes. *Journal of Econometrics*, 7(1):23–55, 1978.

E. L. Denton and V. Birodkar. Unsupervised Learning of Disentangled Representations from Video. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4414–4423. 2017.

R. Douc, E. Moulines, and D. Stoffer. *Nonlinear Time Series: Theory, Methods and Applications with R Examples*. Chapman and Hall/CRC, 2014.

A. Doucet, N. De Freitas, and N. Gordon. An introduction to sequential Monte Carlo methods. In A. Doucet, N. De Freitas, and N. Gordon, editors, *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

D. Dua and C. Graff. UCI Machine Learning Repository, 2017.

J. Durbin and S. J. Koopman. *Time Series Analysis by State Space Methods*. Oxford University Press, 2012.

R. Dürichen, M. A. Pimentel, L. Clifton, A. Schweikard, and D. A. Clifton. Multitask Gaussian Processes for Multivariate Physiological Time-Series Analysis. *IEEE Transactions on Biomedical Engineering*, 62(1):314–322, 2015.

D. Eleveld, P. Colin, A. Absalom, and M. Struys. Pharmacokinetic–Pharmacodynamic Model for Propofol for Broad Application in Anaesthesia and Sedation. *British Journal of Anaesthesia*, 120(5):942–959, 2018.

J. L. Elman. Finding Structure in Time. *Cognitive Science*, 14(2):179–211, 1990.

R. Everson and S. Roberts. Independent Component Analysis: A Flexible Nonlinearity and Decorrelating Manifold Approach. *Neural Computation*, 1999.

O. Fabius and J. R. van Amersfoort. Variational Recurrent Auto-encoders. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

L. Fei-Fei, R. Fergus, and P. Perona. One-Shot Learning of Object Categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

C. Finn, P. Abbeel, and S. Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

R. A. Fisher. The Correlation between Relatives on the Supposition of Mendelian Inheritance. *Transactions of the Royal Society of Edinburgh*, 52, 1919.

E. Fox, M. I. Jordan, E. B. Sudderth, and A. S. Willsky. Sharing Features Among Dynamical Systems with Beta Processes. In *Advances in Neural Information Processing Systems 22*, pages 549–557, 2009.

M. Fraccaro, S. K. Sønderby, U. Paquet, and O. Winther. Sequential Neural Models with Stochastic Layers. In *Advances in Neural Information Processing Systems 30*, pages 2199–2207, 2016.

M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther. A Disentangled Recognition and Nonlinear Dynamics Model for Unsupervised Learning. In *Advances in Neural Information Processing Systems 31*, pages 3601–3610, 2017.

K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent Network Models for Human Dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4346–4354, 2015.

J. Futoma, M. Sendak, B. Cameron, and K. Heller. Predicting Disease Progression with a Model for Multivariate Longitudinal Clinical Data. In *Machine Learning for Healthcare Conference*, 2016.

A. Gelb. *Applied Optimal Estimation*. MIT press, 1974.

A. Gelman and J. Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, 2007.

A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian Data Analysis*. CRC Press, 3rd edition, 2013.

K. Georgatzis, C. K. I. Williams, and C. Hawthorne. Input-Output Non-Linear Dynamical Systems applied to Physiological Condition Monitoring. *Machine Learning for Healthcare*, 2016.

E. Gepts, F. Camu, I. Cockshott, and E. Douglas. Disposition of Propofol Administered as Constant Rate Intravenous Infusions in Humans. *Anesthesia & Analgesia*, 66(12):1256–1263, 1987.

F. Gers, J. Schmidhuber, and F. Cummins. Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, 12(10):2451, 2000.

P. Ghosh, J. Song, E. Aksan, and O. Hilliges. Learning Human Motion Models for Long-Term Predictions. In *2017 International Conference on 3D Vision (3DV)*, pages 458–466. IEEE, 2017.

P. D. Gilbert. *State Space and ARMA Models: An Overview of the Equivalence*. Bank of Canada, 1993.

W. R. Gilks and C. Berzuini. Following a Moving Target – Monte Carlo Inference for Dynamic Bayesian Models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(1):127–146, 2001.

L. Girin, S. Leglaive, X. Bie, J. Diard, T. Hueber, and X. Alameda-Pineda. Dynamical Variational Autoencoders: A Comprehensive Review. *arXiv preprint arXiv:2008.12595*, 2020.

D. V. Glass. John Graunt and His Natural and Political Observations. *Notes and Records of the Royal Society of London*, 19:63–100, 1964.

J. Glen and F. Servin. Evaluation of the Predictive Performance of Four Pharmacokinetic Models for Propofol. *British Journal of Anaesthesia*, 102(5):626–632, 2009.

J. Glen and M. White. A Comparison of the Predictive Performance of Three Pharmacokinetic Models for Propofol Using Measured Values Obtained During Target-Controlled Infusion. *Anaesthesia*, 69(6):550–557, 2014.

X. Glorot and Y. Bengio. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

A. Goyal, A. Sordoni, M.-A. Côté, N. R. Ke, and Y. Bengio. Z-Forcing: Training Stochastic Recurrent Networks. In *Advances in Neural Information Processing Systems 31*, pages 6713–6723, 2017.

A. Graves. Generating Sequences with Recurrent Neural Networks. *arXiv preprint arXiv:1308.0850*, 2013.

A. Graves, A.-r. Mohamed, and G. Hinton. Speech Recognition with Deep Recurrent Neural Networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649. IEEE, 2013.

K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2016.

S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian Neural Networks. In *Advances in Neural Information Processing Systems 32*, 2019.

D. Ha, A. Dai, and Q. Le. Hypernetworks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

J. D. Hamilton. *Time Series Analysis*, volume 2. Princeton University Press, Princeton, NJ, 1994.

P. Harrison and C. Stevens. A Bayesian Approach to Short-Term Forecasting. *Journal of the Operational Research Society*, 22(4):341–362, 1971.

J. Hartikainen and S. Särkkä. Kalman Filtering and Smoothing Solutions to Temporal Gaussian Process Regression Models. In *2010 IEEE International Workshop on Machine Learning for Signal Processing*, 2010.

A. C. Harvey. *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press, 1990.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media, 2009.

K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

K. Helfrich, D. Willmott, and Q. Ye. Orthogonal Recurrent Neural Networks with Scaled Cayley Transform. In *International Conference on Machine Learning*, pages 1974–1983, 2018.

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. β-vae: Learning Basic Visual Concepts with a Constrained Variational Framework. 2017.

M. W. Hirsch, R. L. Devaney, and S. Smale. *Differential Equations, Dynamical systems, and Linear Algebra*, volume 60. Academic Press, 1974.

D. G. Hoag. *Apollo Navigation, Guidance, and Control Systems: A Progress Report*. MIT Instrumentation Laboratory, 1969.

S. Hochreiter. Untersuchungen zu Dynamischen Neuronalen Netzen. *Diploma Thesis, Technische Universität München*, 91(1), 1991.

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

M. D. Hoffman and A. Gelman. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

M. D. Hoffman and M. J. Johnson. ELBO Surgery: Yet Another Way to Carve up the Variational Evidence Lower Bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, 2016.

D. Holden, J. Saito, and T. Komura. A Deep Learning Framework for Character Motion Synthesis and Editing. *ACM Transactions on Graphics (TOG)*, 35(4):138, 2016.

D. Holden, T. Komura, and J. Saito. Phase-Functioned Neural Networks for Character Control. *ACM Transactions on Graphics (TOG)*, 36(4):42, 2017.

N. Holford. Pharmacodynamic Principles and the Time Course of Delayed and Cumulative Drug Effects. *Translational and Clinical Pharmacology*, 26(2):56–59, 2018.

J.-T. Hsieh, B. Liu, D.-A. Huang, L. F. Fei-Fei, and J. C. Niebles. Learning to Decompose and Disentangle Representations for Video Prediction. In *Advances in Neural Information Processing Systems 31*, pages 517–526, 2018.

E. Hsu, K. Pulli, and J. Popović. Style Translation for Human Motion. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 1082–1089. ACM, 2005.

W.-N. Hsu, Y. Zhang, and J. Glass. Unsupervised Learning of Disentangled and Interpretable Representations from Sequential Data. In *Advances in Neural Information Processing Systems 30*, pages 1876–1887. 2017.

G. Hu and R. F. O'Connell. Analytical Inversion of Symmetric Tridiagonal Matrices. *Journal of Physics A: Mathematical and General*, 29(7):1511, 1996.

X. Huan and Y. M. Marzouk. Simulation-Based Optimal Bayesian Experimental Design for Nonlinear Systems. *Journal of Computational Physics*, 232(1):288–317, 2013.

C. Hull. Pharmacokinetics and Pharmacodynamics. *British Journal of Anaesthesia*, 51(7):579–594, 1979.

T. Hume and M. Pazzani. Learning Sets of Related Concepts: A Shared Task Model. In *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society*, 1996.

T. Hüppe, F. Maurer, D. I. Sessler, T. Volk, and S. Kreuer. Retrospective comparison of Eleveld, Marsh, and Schnider propofol pharmacokinetic models in 50 patients. *British Journal of Anaesthesia*, 2019.

F. Huszár. How (Not) to Train Your Generative Model: Scheduled Sampling, Likelihood, Adversary? *arXiv preprint arXiv:1511.05101*, 2015.

H. Hyötyniemi. Turing Machines are Recurrent Neural Networks. *Proceedings of STeP*, 96, 1996.

L. Y. Inoue, M. Neira, C. Nelson, M. Gleave, and R. Etzioni. Cluster-Based Network Model for Time-Course Gene Expression Data. *Biostatistics*, 2007.

O. Ivanov, M. Figurnov, and D. P. Vetrov. Variational Autoencoder with Arbitrary Conditioning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

H. Jaeger and H. Haas. Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. *Science*, 304(5667):78–80, 2004.

S. M. Jayakumar, W. M. Czarnecki, J. Menick, J. Schwarz, J. Rae, S. Osindero, Y. W. Teh, T. Harley, and R. Pascanu. Multiplicative Interactions and Where to Find Them. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

C. Jeleazcov, M. Lavielle, J. Schüttler, and H. Ihmsen. Pharmacodynamic Response Modelling of Arterial Blood Pressure in Adult Volunteers During Propofol Anaesthesia. *British Journal of Anaesthesia*, 115(2):213–226, 2015.

R. W. Jelliffe. A Mathematical Analysis of Digitalis Kinetics in Patients with Normal and Reduced Renal Function. *Mathematical Biosciences*, 1(2):305–325, 1967.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An Introduction to Variational Methods for Graphical Models. *Machine learning*, 37(2):183–233, 1999.

R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

M. Karl, M. Soelch, J. Bayer, and P. van der Smagt. Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data. In *International Conference on Learning Representations (ICLR)*, 2017.

R. E. Kass and A. E. Raftery. Bayes Factors. *Journal of the American Statistical Association*, 90 (430):773–795, 1995.

C. Kemp, A. Perfors, and J. B. Tenenbaum. Learning Overhypotheses with Hierarchical Bayesian Models. *Developmental Science*, 10(3):307–321, 2007.

I. Khuri, I. Andre, et al. The Parameterization of Orthogonal Matrices: A Review Mainly for Statisticians. *South African Statistical Journal*, 23(2):231–250, 1989.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.

D. P. Kingma and M. Welling. Stochastic Gradient VB and the Variational Auto-Encoder. In *Second International Conference on Learning Representations, ICLR*, 2014.

D. Korkinof and Y. Demiris. Multi-Task and Multi-Kernel Gaussian Process Dynamical Systems. *Pattern Recognition*, 66:190–201, 2017.

A. Kosiorek, H. Kim, Y. W. Teh, and I. Posner. Sequential Attend, Infer, Repeat: Generative Modelling of Moving Objects. In *Advances in Neural Information Processing Systems*, pages 8606–8616, 2018.

G. Lai, Z. Dai, Y. Yang, and S. Yoo. Re-examination of the Role of Latent Variables in Sequence Modeling. In *Advances in Neural Information Processing Systems 33*, pages 7812–7822, 2019.

B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-Level Concept Learning Through Probabilistic Program Induction. *Science*, 350(6266):1332–1338, 2015.

A. M. Lamb, A. G. A. P. Goyal, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio. Professor Forcing: A New Algorithm for Training Recurrent Networks. In *Advances In Neural Information Processing Systems 30*, pages 4601–4609, 2016.

A. Laub. A Schur Method for Solving Algebraic Riccati Equations. *IEEE Transactions on Automatic Control*, 1979.

N. Lawrence. Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models. *Journal of Machine Learning Research*, 6(Nov):1783–1816, 2005.

T. A. Le, M. Igl, T. Rainforth, T. Jin, and F. Wood. Auto-Encoding Sequential Monte Carlo. In *International Conference on Learning Representations*, 2018.

L. Lee and W. E. L. Grimson. Gait Analysis for Recognition and Classification. In *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*. IEEE, 2002.

C. Lemieux. *Monte Carlo and Quasi-Monte Carlo Sampling*. Springer, 2009.

S. C. Li and B. M. Marlin. A Scalable End-to-End Gaussian Process Adapter for Irregularly Sampled Time Series Classification. In *Advances in Neural Information Processing Systems 29*, 2016.

A. Lin, Y. Zhang, J. Heng, S. A. Allsop, K. M. Tye, P. E. Jacob, and D. Ba. Clustering Time Series with Nonlinear Dynamics: A Bayesian Non-Parametric and Particle-Based Approach. In *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019.

J. H. Lin. Pharmacokinetic and Pharmacodynamic Variability: A Daunting Challenge in Drug Therapy. *Current Drug Metabolism*, 8(2):109–136, 2007.

S. Linderman, M. Johnson, A. Miller, R. Adams, D. Blei, and L. Paninski. Bayesian Learning and Inference in Recurrent Switching Linear Dynamical Systems. In *Artificial Intelligence and Statistics*, pages 914–922, 2017.

H. Liu, Y.-S. Ong, X. Shen, and J. Cai. When Gaussian Process Meets Big Data: A Review of Scalable GPs. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

F. A. Lobo and S. Schraag. Limitations of Anaesthesia Depth Monitoring. *Current Opinion in Anesthesiology*, 24(6):657–664, 2011.

Y. Lu, A. Kumar, S. Zhai, Y. Cheng, T. Javidi, and R. Feris. Fully-Adaptive Feature Sharing in Multi-Task Networks with Applications in Person Attribute Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5334–5343, 2017.

J. Luttinen, T. Raiko, and A. Ilin. Linear State-Space Model with Time-Varying Dynamics. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 338–353. Springer, 2014.

D. J. MacKay. Bayesian Interpolation. *Neural Computation*, 1992.

C. J. Maddison, J. Lawson, G. Tucker, N. Heess, M. Norouzi, A. Mnih, A. Doucet, and Y. Teh. Filtering Variational Objectives. In *Advances in Neural Information Processing Systems, 31*, pages 6573–6583, 2017.

D. E. Mager, E. Wyska, and W. J. Jusko. Diversity of Mechanism-Based Pharmacodynamic Models. *Drug Metabolism and Disposition*, 31(5):510–518, 2003.

M. Mahmud and S. Ray. Transfer Learning Using Kolmogorov Complexity: Basic Theory and Empirical Evaluations. In *Advances in Neural Information Processing Systems 22*, pages 985–992, 2008.

A. Makhzani, J. Shlens, N. Jaitly, and I. J. Goodfellow. Adversarial Autoencoders. *CoRR*, abs/1511.05644, 2015.

S. Makridakis. A Survey of Time Series. *International Statistical Review/Revue Internationale de Statistique*, pages 29–70, 1976.

J. Marino, M. Cvitkovic, and Y. Yue. A General Method for Amortizing Variational Filtering. In *Advances in Neural Information Processing Systems 32*, pages 7857–7868, 2018.

B. Marsh, M. White, N. Morton, and G. Kenny. Pharmacokinetic Model Driven Infusion of Propofol in Children. *British Journal of Anaesthesia*, 67(1):41–48, 1991.

J. Martens and I. Sutskever. Learning Recurrent Neural Networks with Hessian-Free Optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 1033–1040. Citeseer, 2011.

J. Martens, J. Ba, and M. Johnson. Kronecker-factored Curvature Approximations for Recurrent Neural Networks. In *International Conference on Learning Representations*, 2018.

J. Martinez, M. J. Black, and J. Romero. On Human Motion Prediction Using Recurrent Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2891–2900, 2017.

I. Mason, S. Starke, H. Zhang, H. Bilen, and T. Komura. Few-Shot Learning of Homogeneous Human Locomotion Styles. In *Computer Graphics Forum*, volume 37, pages 143–153. Wiley Online Library, 2018.

K. Masui, R. N. Upton, A. G. Doufas, J. F. Coetzee, T. Kazama, E. P. Mortier, and M. M. Struys. The Performance of Compartmental and Physiologically Based Recirculatory Pharmacokinetic Models for Propofol: a Comparison Using Bolus, Continuous, and Target-Controlled Infusion Data. *Anesthesia & Analgesia*, 111(2):368–379, 2010.

M. McCloskey and N. J. Cohen. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165. Elsevier, 1989.

L. A. McGee and S. F. Schmidt. Discovery of the Kalman Filter as a Practical Tool for Aerospace and Industry. 1985.

G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*, volume 382. John Wiley & Sons, 2007.

R. Memisevic and G. Hinton. Unsupervised Learning of Image Transformations. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

R. Memisevic and G. E. Hinton. Learning to Represent Spatial Transformations with Factored Higher-Order Boltzmann Machines. *Neural Computation*, 22(6):1473–1492, 2010.

Đ. Miladinović, M. Waleed Gondal, B. Schölkopf, J. M. Buhmann, and S. Bauer. Disentangled State Space Representations. *arXiv e-prints*, art. 1906.03255, Jun 2019.

J. Miller and M. Hardt. Stable Recurrent Models. In *International Conference on Learning Representations (ICLR)*, 2019.

J. Min, H. Liu, and J. Chai. Synthesis and Editing of Personalized Stylistic Human Motion. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. ACM, 2010.

T. M. Mitchell. The Need for Biases in Learning Generalizations. In T. G. Dietterich and J. Shavlik, editors, *Readings in Machine Learning*. Morgan Kaufmann, 1991.

A. Mnih and D. Rezende. Variational Inference for Monte Carlo Objectives. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2188–2196, 2016.

P. J. Moore and M. A. Little. Enhancements to a Method of Analogues Forecasting Algorithm. In *International Symposium on Nonlinear Theory and its Applications*, 2014.

E. A. Mukamel, E. Pirondini, B. Babadi, K. F. K. Wong, E. T. Pierce, P. G. Harrell, J. L. Walsh, A. F. Salazar-Gomez, S. S. Cash, E. N. Eskandar, et al. A Transition in Brain State During Propofol-Induced Unconsciousness. *Journal of Neuroscience*, 34(3):839–845, 2014.

K. P. Murphy. Switching Kalman Filters. Technical report, UC Berkeley, 1998.

T. B. Murtagh. Analysis of Sextant Navigation Measurements During Lunar Module Rendezvous. 1967.

P. Myles, K. Leslie, J. McNeil, A. Forbes, M. Chan, B.-A. T. Group, et al. Bispectral Index Monitoring to Prevent Awareness During Anaesthesia: the B-Aware Randomised Controlled Trial. *The Lancet*, 363(9423):1757–1763, 2004.

C. Naesseth, S. Linderman, R. Ranganath, and D. Blei. Variational Sequential Monte Carlo. In *International Conference on Artificial Intelligence and Statistics*, pages 968–977, 2018.

R. M. Neal et al. MCMC Using Hamiltonian Dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11), 2011.

J. Nocedal and S. Wright. *Numerical Optimization*. Springer Science & Business Media, 2006.

H. F. Nweke, Y. W. Teh, M. A. Al-Garadi, and U. R. Alo. Deep Learning Algorithms for Human Activity Recognition using Mobile and Wearable Sensor Networks: State of the Art and Research Challenges. *Expert Systems with Applications*, 2018.

A. V. Oppenheim, R. W. Schafer, and J. R. Buck. *Discrete-Time Signal Processing*. Prentice-hall Englewood Cliffs, 2nd edition, 1999.

M. Opper and O. Winther. A Bayesian Approach to On-line Learning. pages 363–378. 1998.

M. A. Osborne, S. J. Roberts, A. Rogers, S. D. Ramchurn, and N. R. Jennings. Towards Real-Time Information Processing of Sensor Network Data Using Computationally Efficient Multi-Output Gaussian Processes. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*, pages 109–120. IEEE Computer Society, 2008.

A. B. Owen. *Monte Carlo Theory, Methods and Examples.* 2013.

S. J. Pan and Q. Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

R. Pascanu, T. Mikolov, and Y. Bengio. On the Difficulty of Training Recurrent Neural Networks. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1310–1318, 2013.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic Differentiation in PyTorch. 2017.

D. Pavllo, D. Grangier, and M. Auli. Quaternet: A Quaternion-Based Recurrent Model for Human Motion. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2018.

J. C. Pinheiro and D. M. Bates. Unconstrained Parametrizations for Variance-Covariance Matrices. *Statistics and Computing*, 6(3):289–296, 1996.

J. C. Pinheiro and D. M. Bates. *Mixed-Effects Models in S and S-Plus.* Springer, 2000.

L. Prechelt. Automatic Early Stopping Using Cross Validation: Quantifying the Criteria. *Neural Networks*, 1998.

S. J. Prince. *Computer Vision: Models Learning and Inference.* Cambridge University Press, 2012.

J. A. Quinn, C. K. I. Williams, and N. McIntosh. Factorial Switching Linear Dynamical Systems applied to Physiological Condition Monitoring. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(9):1537–1551, 2009.

L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77:257–286, 1989.

U. Ramer. An Iterative Procedure for the Polygonal Approximation of Plane Curves. *Computer Graphics and Image Processing*, 1(3):244–256, 1972.

S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. Deep State Space Models for Time Series Forecasting. In *Advances in Neural Information Processing Systems 31*, pages 7785–7794, 2018.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*, volume 2. MIT press Cambridge, MA, 2006.

S. Ravi and H. Larochelle. Optimization as a Model for Few-Shot Learning. In *5th International Conference on Learning Representations*, 2017.

I. Redko, E. Morvant, A. Habrard, M. Sebban, and Y. Bennani. *Advances in Domain Adaptation Theory.* ISTE Press - Elsevier, 2019.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1278–1286, 2014.

S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain. Gaussian Processes for Time-Series Modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2013.

S. J. Roberts and W. D. Penny. Variational Bayes for Generalized Autoregressive Models. *IEEE Transactions on Signal Processing*, 2002.

G. Rogez and C. Schmid. Mocap-Guided Data Augmentation for 3D Pose Estimation in the Wild. In *Advances in Neural Information Processing Systems*, pages 3108–3116, 2016.

S. Roweis and Z. Ghahramani. A Unifying Review of Linear Gaussian Models. *Neural Computation*, 11(2):305–345, 1999.

J. A. Royle and R. M. Dorazio. *Hierarchical Modeling and Inference in Ecology: the Analysis of Data from Populations, Metapopulations and Communities.* Elsevier, 2008.

Y. Rubanova, R. T. Q. Chen, and D. K. Duvenaud. Latent Ordinary Differential Equations for Irregularly-Sampled Time Series. In *Advances in Neural Information Processing Systems 32*, 2019.

F. Saad and V. Mansinghka. Temporally-Reweighted Chinese Restaurant Process Mixtures for Clustering, Imputing, and Forecasting Multivariate Time Series. In *International Conference on Artificial Intelligence and Statistics*, pages 755–764, 2018.

A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-Learning with Memory-Augmented Neural Networks. In *International Conference on Machine Learning*, pages 1842–1850, 2016.

S. Särkkä. *Bayesian Filtering and Smoothing*, volume 3. Cambridge University Press, 2013.

S. Särkkä and A. Solin. *Applied Stochastic Differential Equations*, volume 10. Cambridge University Press, 2019.

S. W. Scheff. *Fundamental Statistical Principles for the Neurobiologist: A Survival Guide.* Academic Press, 2016.

J. Schmidhuber. Evolutionary Principles in Self-Referential Learning. On Learning Now to Learn: The Meta-Meta-Meta...-Hook. *Diploma Thesis, Technische Universität München*, 1987.

T. W. Schnider, C. F. Minto, P. L. Gambus, C. Andresen, D. B. Goodale, S. L. Shafer, and E. J. Youngs. The Influence of Method of Administration and Covariates on the Pharmacokinetics of Propofol in Adult Volunteers. *The Journal of the American Society of Anesthesiologists*, 88 (5):1170–1182, 1998.

P. Schulam and S. Saria. A Framework for Individualizing Predictions of Disease Trajectories by Exploiting Multi-Resolution Structure. In *Advances in Neural Information Processing Systems 29*, pages 748–756, 2015.

P. Schuller, S. Newell, P. Strickland, and J. Barry. Response of Bispectral Index to Neuromuscular Block in Awake Volunteers. *British Journal of Anaesthesia*, 115, 2015.

S. Seth, I. Murray, and C. K. I. Williams. Model Criticism in Latent Space. *Bayesian Analysis*, 14(3):703–725, 2019.

L. B. Sheiner and S. L. Beal. Evaluation of Methods for Estimating Population Pharmacokinetic Parameters II. Biexponential Model and Experimental Pharmacokinetic Data. *Journal of Pharmacokinetics and Pharmacodynamics*, 9(5):635–651, 1981.

H. Shin, J. K. Lee, J. Kim, and J. Kim. Continual Learning with Deep Generative Replay. In *Advances in Neural Information Processing Systems 31*, pages 2990–2999, 2017.

S. N. Shukla and B. M. Marlin. Interpolation-Prediction Networks for Irregularly Sampled Time Series. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

R. H. Shumway and D. S. Stoffer. Dynamic Linear Models with Switching. *Journal of the American Statistical Association*, 86(415):763–769, 1991.

R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications: with R Examples*. Springer, 2017.

S. M. Siddiqi, B. Boots, and G. J. Gordon. A Constraint Generation Approach to Learning Stable Linear Dynamical Systems. Technical report, DTIC Document, 2008.

H. T. Siegelmann and E. D. Sontag. Turing Computability with Neural Nets. *Applied Mathematics Letters*, 4(6):77–80, 1991.

K. Sohn, H. Lee, and X. Yan. Learning Structured Output Representation Using Deep Conditional Generative Models. In *Advances in Neural Information Processing Systems 29*, pages 3483–3491, 2015.

H. Soleimani, A. Subbaswamy, and S. Saria. Treatment-Response Models for Counterfactual Reasoning with Continuous-Time, Continuous-Valued Interventions. *arXiv preprint arXiv:1704.02038*, 2017.

S. Spieckermann, S. Düll, S. Udluft, A. Hentschel, and T. Runkler. Exploiting Similarity in System Identification Tasks with Recurrent Neural Networks. *Neurocomputing*, 169:343–349, 2015.

J. H. Stock and M. W. Watson. Forecasting Inflation. *Journal of Monetary Economics*, 44(2): 293–335, 1999.

S. H. Strogatz. *Nonlinear Dynamics and Chaos: with Applications to Physics, Biology, Chemistry, and Engineering*. CRC Press, 2018.

M. Sugiyama and M. Kawanabe. *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation*. MIT Press, 2012.

D. Sussillo and O. Barak. Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks. *Neural Computation*, 2013.

I. Sutskever, J. Martens, and G. E. Hinton. Generating Text with Recurrent Neural Networks. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the Importance of Initialization and Momentum in Deep Learning. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1139–1147, 2013.

I. Sutskever, O. Vinyals, and Q. Le. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems, 28*, 2014.

J. F. Svénsen. *GTM: the Generative Topographic Mapping.* PhD thesis, Aston University, 1998.

Y. Tang and R. R. Salakhutdinov. Learning Stochastic Feedforward Neural Networks. In *Advances in Neural Information Processing Systems 27*, pages 530–538, 2013.

G. W. Taylor, L. Sigal, D. J. Fleet, and G. E. Hinton. Dynamical Binary Latent Variable Models for 3D Human Pose Tracking. In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

S. Thrun. Is Learning the n-th Thing Any Easier than Learning the First? In *Advances in Neural Information Processing Systems 12*, pages 640–646, 1996.

S. Thrun and J. O'Sullivan. Discovering Structure in Multiple Learning Tasks: The TC Algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, volume 96, pages 489–497, 1996.

S. Thrun and L. Pratt. *Learning to Learn.* Springer Science & Business Media, 1998.

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

M. K. Titsias and M. Lázaro-Gredilla. Spike and Slab Variational Inference for Multi-Task and Multiple Kernel Learning. In *Advances in Neural Information Processing Systems 24*, pages 2339–2347, 2011.

N. Tomasetti, C. Forbes, A. Panagiotelis, et al. Updating Variational Bayes: Fast Sequential Posterior Inference. Technical report, Monash University, Department of Econometrics and Business Statistics, 2019.

J. Tomczak and M. Welling. VAE with a VampPrior. In *International Conference on Artificial Intelligence and Statistics*, 2018.

F. Toqué, E. Côme, M. K. El Mahrsi, and L. Oukhellou. Forecasting Dynamic Public Transport Origin-Destination Matrices with Long-Short Term Memory Recurrent Neural Networks. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016.

M. K. Transtrum, B. B. Machta, and J. P. Sethna. The Geometry of Nonlinear Least Squares with Applications to Sloppy Models and Optimization. *Physical Review E*, 83(3):036701, 2011.

J. V. Tsimikas and J. Ledolter. Mixed Model Representation of State Space Models: New Smoothing Results and their Application to REML Estimation. *Statistica Sinica*, pages 973–991, 1997.

S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. MoCoGAN: Decomposing Motion and Content for Video Generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1526–1535, 2018.

R. E. Turner and M. Sahani. Two Problems with Variational Expectation Maximisation for Time-Series Models. *Bayesian Time Series Models*, pages 115–138, 2011.

L. Van der Maaten and G. Hinton. Visualizing Data Using t-SNE. *Journal of Machine Learning Research*, 9(11), 2008.

J. van der Westhuizen and J. Lasenby. The Unreasonable Effectiveness of the Forget Gate. *arXiv preprint arXiv:1804.04849*, 2018.

P. Van Overschee and B. De Moor. *Subspace Identification for Linear Systems: Theory – Implementation – Applications*. Springer Science & Business Media, 2012.

J. Vanschoren. Meta-Learning: A Survey. *arXiv preprint arXiv:1810.03548*, 2018.

C. Viboud, P.-Y. Boëlle, F. Carrat, A.-J. Valleron, and A. Flahault. Prediction of the Spread of Influenza Epidemics by the Method of Analogues. *American Journal of Epidemiology*, 2003.

R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee. Decomposing Motion and Content for Natural Video Sequence Prediction. In *International Conference on Learning Representations*, 2017.

O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching Networks for One Shot Learning. In *Advances in Neural Information Processing Systems 30*, pages 3630–3638, 2016.

J. Wagner. Kinetics of Pharmacologic Response I. Proposed Relationships Between Response and Drug Concentration in the Intact Animal and Man. *Journal of Theoretical Biology*, 20(2): 173–201, 1968.

M. J. Wainwright, M. I. Jordan, et al. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian Process Dynamical Models for Human Motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):283–298, 2008.

M. West and J. Harrison. *Bayesian Forecasting and Dynamic Models*. Springer Science & Business Media, 2006.

M. White, G. N. Kenny, and S. Schraag. Use of Target Controlled Infusion to Derive Age and Gender Covariates for Propofol Clearance. *Clinical Pharmacokinetics*, 47(2):119–127, 2008.

R. J. Williams and J. Peng. An Efficient Gradient-Based Algorithm for Online Training of Recurrent Network Trajectories. *Neural computation*, 2(4):490–501, 1990.

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144*, 2016.

S. Xia, C. Wang, J. Chai, and J. Hodgins. Realtime Style Transfer for Unlabeled Heterogeneous Human Motion. *ACM Transactions on Graphics (TOG)*, 34(4):119, 2015.

L. Xu and M. I. Jordan. On Convergence Properties of the EM Algorithm for Gaussian Mixtures. *Neural Computation*, 8(1):129–151, 1996.

Y. Xu, Y. Xu, and S. Saria. A Bayesian Nonparametric Approach for Estimating Individualized Treatment-Response Curves. In *Machine Learning for Healthcare Conference*, 2016.

Z. Xu. The Basic Forms of Ancient Chinese Sunspot Records. *Chinese Science*, 9:19–28, 1989.

Y. Xue, X. Liao, L. Carin, and B. Krishnapuram. Multi-Task Learning for Classification with Dirichlet Process Priors. *Journal of Machine Learning Research*, 8(Jan):35–63, 2007.

L. Yingzhen and S. Mandt. Disentangled Sequential Autoencoder. In *International Conference on Machine Learning*, pages 5656–5665, 2018.

K. Zhang, J. Xu, M. R. Min, G. Jiang, K. Pelechrinis, and H. Zhang. Automated IT System Failure Prediction: A Deep Learning Approach. In *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016.

Y. Zhang and Q. Yang. A Survey on Multi-Task Learning. *arXiv preprint arXiv:1707.08114*, 2017.

Y. Zhang and D. Yeung. A Convex Formulation for Learning Task Relationships in Multi-Task Learning. In *UAI 2010, Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, 2010.

J. Zhou, L. Han, and S. Liu. Nonlinear Mixed-Effects State Space Models with Applications to HIV Dynamics. *Statistics & Probability Letters*, 83(5):1448–1456, 2013.

APPENDIX A

Appendix

A.1 Gaussian identities

In this thesis, we make use of the following Gaussian conditioning formulae:

1. **Gaussian conditional → joint distribution**. Let $\mathbf{y}$ depend on $\mathbf{x}$ through some affine transformation with Gaussian noise, and $\mathbf{x}$ have a Gaussian prior distribution, i.e.

$$p(\mathbf{y} \,|\, \mathbf{x}) = \mathcal{N}\left(H\mathbf{x} + \mathbf{u}, R\right), \qquad p(\mathbf{x}) = \mathcal{N}\left(\mathbf{m}, P\right). \tag{A.1}$$

   Then the joint distribution over $\mathbf{x}$ and y is:

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{m} \\ H\mathbf{m} + \mathbf{u} \end{pmatrix}, \begin{pmatrix} P & PH^{\mathsf{T}} \\ HP & HPH^{\mathsf{T}} + R \end{pmatrix}\right). \tag{A.2}$$

2. **Gaussian joint → conditional distribution**. Let:

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu}_{\mathbf{x}} \\ \boldsymbol{\mu}_{\mathbf{y}} \end{pmatrix}, \begin{pmatrix} \Sigma_{\mathbf{xx}} & \Sigma_{\mathbf{xy}} \\ \Sigma_{\mathbf{yx}} & \Sigma_{\mathbf{yy}} \end{pmatrix}\right). \tag{A.3}$$

   Then the conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ is:

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}\left(\boldsymbol{\mu}_x + \Sigma_{\mathbf{xy}}\Sigma_{\mathbf{yy}}^{-1}(\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}}), \Sigma_{\mathbf{xx}} - \Sigma_{\mathbf{xy}}\Sigma_{\mathbf{yy}}^{-1}\Sigma_{\mathbf{yx}}\right) \tag{A.4}$$

The first identity (eq. A.2) follows from various applications of the law of total expectation. The second identity requires a bit of extra work, but a proof can be found e.g. in Bishop (2007), §2.3.1.

A.2 Background

This section contains three proofs regarding linear dynamical systems omitted from the time series section of the background material (Section 2.2) in the interests of brevity.

A.2.1 Non-identifiability of an LDS

The linear dynamical system with state dimension $n_x > 1$ is non-identifiable due to over-parameterisation. That is, there are infinitely many ways of specifying the parameters while maintaining the same distribution over observations. As discussed in Section 2.2.3.2, this is due to the basis of the hidden state (and directions thereof) being unspecified by the usual formulation. As a result we can rotate/reflect the hidden state by any invertible linear transformation G and obtain the same distribution over Y.

Proof. Consider the LDS $p_{\mathbf{y},\mathbf{x}}(\mathbf{y}_{1:T}, \mathbf{x}_{1:T})$ as parameterised in Equation (2.31)-(2.32), and the change of basis $\mathbf{v}_t = G^{-1}\mathbf{x}_t$ for an invertible matrix G:

$$p(\mathbf{y}_{1:T}) = \int p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}_{1:T} \,|\, \mathbf{x}_{1:T}) p_{\mathbf{x}}(\mathbf{x}_{1:T}) \,\mathrm{d}\mathbf{x}_{1:T} \tag{A.5}$$

$$= \int \underbrace{p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}_{1:T} \,|\, G\mathbf{v}_1, \ldots, G\mathbf{v}_T)}_{(i)} \underbrace{|G|^T p_{\mathbf{x}}(G\mathbf{v}_1, \ldots, G\mathbf{v}_T)}_{(ii)} \,\mathrm{d}\mathbf{v}_{1:T} \tag{A.6}$$

which follows from the change of variables formula. To remove ambiguity, we subscript each density by the variables which define them. By Equation (A.6), the implied distributions wrt. $\mathbf{v}$ in the integrand, $p_{\mathbf{y}|\mathbf{v}}(\mathbf{y}_{1:T} \,|\mathbf{v}_{1:T})$, $p_{\mathbf{v}}(\mathbf{v}_{1:T})$ have the same marginal density over $\mathbf{y}_{1:T}$. We will show below that these distributions remain LDSs and have the parameterisation given in Equation (2.33)-(2.34) of Section 2.2.3.2. We can therefore conclude that this reparameterisation defines the same distribution over $\mathbf{y}_{1:T}$ and holds for any invertible G.

To demonstrate this, we consider the terms in the integrand in turn, working in log space for convenience.

(i) The emission distribution:

$$\log p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}_{1:T} \,|\, G\mathbf{v}_1, \ldots, G\mathbf{v}_T) = \sum_{t=1}^{T} \log p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}_t \,|\, G\mathbf{v}_t) = \sum_{t=1}^{T} \log \mathcal{N}\left(\mathbf{y}_t \,|\, CG\mathbf{v}_t,\, S\right) \tag{A.7}$$

$$=: \sum_{t=1}^{T} \log p_{\mathbf{y}|\mathbf{v}}(\mathbf{y}_t \,|\, \mathbf{v}_t) =: \log p_{\mathbf{y}|\mathbf{v}}(\mathbf{y}_{1:T} \,|\, \mathbf{v}_{1:T}) \tag{A.8}$$

The emission distribution $p_{\mathbf{y}|\mathbf{v}}(\mathbf{y}_t \,|\, \mathbf{v}_t)$ hence uses the parameters of Equation (2.34).

(ii) The transition distribution:

$$\log p_{\mathbf{x}}(G\mathbf{v}_1, \ldots, G\mathbf{v}_T) + T \log |G| \tag{A.9}$$

$$= \sum_{t=1}^{T} \log p_{\mathbf{x}}(G\mathbf{v}_t \,|\, G\mathbf{v}_{t-1}) + \log |G| \tag{A.10}$$

$$= \sum_{t=1}^{T} \log \mathcal{N}\left(G\mathbf{v}_t \,|\, AG\mathbf{v}_{t-1},\, R\right) + \log |G| \tag{A.11}$$

$$= -\frac{1}{2} \sum_{t=1}^{T} \left[\log |R| - 2 \log |G| + \text{const} + (G\mathbf{v}_t - AG\mathbf{v}_{t-1})^{\mathsf{T}} R^{-1} (G\mathbf{v}_t - AG\mathbf{v}_{t-1})\right] \tag{A.12}$$

$$= -\frac{1}{2} \sum_{t=1}^{T} \Big[\log |G^{-1} R G^{-\mathsf{T}}| + \text{const} + \tag{A.13}$$

$$(\mathbf{v}_t - G^{-1} AG\mathbf{v}_{t-1})^{\mathsf{T}} \left(G^{-1} R G^{-\mathsf{T}}\right)^{-1} (\mathbf{v}_t - G^{-1} AG\mathbf{v}_{t-1})\Big] \tag{A.14}$$

$$= \sum_{t=1}^{T} \log \mathcal{N}\left(\mathbf{v}_t \,|\, G^{-1} AG\mathbf{v}_{t-1},\, G^{-1} R G^{-\mathsf{T}}\right) =: p_{\mathbf{v}}(\mathbf{v}_{1:T}). \tag{A.15}$$

Hence the emission distribution $p_{\mathbf{v}}(\mathbf{v}_{1:T})$ is a VAR(1) system using the parameters of Equation (2.33).

We therefore conclude that the integrand does define a LDS wrt. $\mathbf{v}$ with the parameters of Equation (2.33)-(2.34). □

A.2.2 An ARMA(p, q) process can be written as an LDS.

The proof here broadly follows Hamilton (1994), §13.1. We will use the same transition matrix for the ARMA(p, q) model as when writing an AR(p) model as a VAR(1), i.e. as in Equation (2.25), but now with $n_x = r = \min(p, q+1)$, using $a_j = 0$ where $j > r$.

We will first need to rewrite the ARMA model in Equation (2.18) using the lag operator L, defined such that $Lz_t = z_{t-1}$, $L^2 z_t = z_{t-2}$, …, etc. applied to any variable z_t. Then the ARMA(p, q) model in Equation (2.18) may be written:

$$\left(1 - \sum_{j=1}^{p} a_j L^j\right) y_t \;=\; \left(1 - \sum_{j=1}^{q} c_j L^j\right) \epsilon_t, \qquad \epsilon_t \sim \mathcal{N}\left(0, \sigma^2\right), \tag{A.16}$$

where $c_j = 0$ for $j > q$. We will find this form useful for algebraic manipulation in what follows.

Proof. Consider the observation process:

$$y_t = \underbrace{\begin{bmatrix} 1 & c_1 & c_2 & \dots & c_r \end{bmatrix}}_{C_{\text{ARMA}}} \mathbf{x}_t, \tag{A.17}$$

observed without noise. Explicitly, writing x_{jt} for the jth element of $\mathbf{x}_t$, we have:

$$y_t \;=\; x_{1t} + \sum_{j=2}^{r} c_{j-1}\, x_{jt} \;=\; x_{1t} + \sum_{j=2}^{r} c_{j-1}\, x_{1,t-j+1}, \tag{A.18}$$

since by the choice of matrix A we have $x_{jt} = x_{j-1,t-1}$ which can be recursively applied for $j > 1$. Using the lag-operator form of the model from Equation (A.16), we can write (A.18) as:

$$y_t \;=\; \left(1 + \sum_{j=1}^{r-1} c_j L^j\right) x_{1t}, \tag{A.19}$$

where L is the lag operator. Now multiplying through by the AR polynomial $1 - \sum_{j=1}^{r} a_j L^j$ (cf. eq. A.16) gives:

$$\left(1 - \sum_{j=1}^{r} a_j L^j\right) y_t \;=\; \left(1 + \sum_{j=1}^{r-1} c_j L^j\right)\left(1 - \sum_{j=1}^{r} a_j L^j\right) x_{1t} \tag{A.20}$$

since the polynomials commute. The structure of the matrix A gives us that $x_{1t} = \sum_{j=1}^{r} a_j L^j\, x_{1t} + \epsilon_t$ and hence $(1 - \sum_{j=1}^{r} a_j L^j)\, x_{1t} = \epsilon_t$. Therefore,

$$\left(1 - \sum_{j=1}^{r} a_j L^j\right) y_t \;=\; \left(1 + \sum_{j=1}^{r-1} c_j L^j\right) \epsilon_t \tag{A.21}$$

which is the form of the ARMA model in Equation (A.16). □

A.2.3 Derivation of the Kalman Filter.

The Kalman Filter is a recursive procedure for calculating $p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t})$ in an LDS. This can be derived fairly simply using the Gaussian identities in A.1. Using Bayes' rule, we have:

$$p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t}) \propto p(\mathbf{x}_t,\, \mathbf{y}_t \,|\, \mathbf{y}_{1:t-1}) \tag{A.22}$$

$$= p(\mathbf{y}_t \,|\, \mathbf{x}_t, \mathbf{y}_{1:t-1})\, p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t-1}) \tag{A.23}$$

$$= p(\mathbf{y}_t \,|\, \mathbf{x}_t) \underbrace{\int p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1} \,|\, \mathbf{y}_{1:t-1})\, \mathrm{d}\mathbf{x}_{t-1}}_{\text{prediction}} \tag{A.24}$$

where $p(\mathbf{y}_t|\mathbf{x}_t)$ in Equation (A.24) follows from the conditional independence assumptions of eqs. (2.31 - 2.32). Let the previous filtering distribution $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{m}_{t-1}, P_{t-1})$,

calculated from the previous recursion. The recursive update follows two steps: firstly the 'prediction' step (highlighted in eq. A.24), and then the 'update' step, which incorporates the information from the observation and renormalises the result.

Prediction step Using Gaussian identity (A.2), the 'prediction' of the latent $\mathbf{x}_t$ given $\mathbf{y}_{1:t-1}$ is:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{m}_t^-, P_t^-), \tag{A.25}$$

where

$$\begin{aligned} \mathbf{m}_t^- &= A\mathbf{m}_{t-1}, \\ P_t^- &= AP_{t-1}A^\mathsf{T} + R. \end{aligned}$$

Update step We now incorporate the information from the emission process into the prediction and renormalise. This follows two steps. Firstly we combine the emission distribution $p(\mathbf{y}_t \,|\, \mathbf{x}_t)$ with the predictive state equation above (eq. A.25), using Gaussian identity (A.2):

$$p\left(\begin{pmatrix}\mathbf{x}_t \\ \mathbf{y}_t\end{pmatrix} \middle|\, \mathbf{y}_{1:t-1}\right) = \mathcal{N}\left(\begin{pmatrix}\mathbf{m}_t^- \\ C\mathbf{m}_t^-\end{pmatrix}, \begin{pmatrix}P_t^- & P_t^-C^\mathsf{T} \\ CP_t^- & CP_t^-C^\mathsf{T} + S\end{pmatrix}\right). \tag{A.26}$$

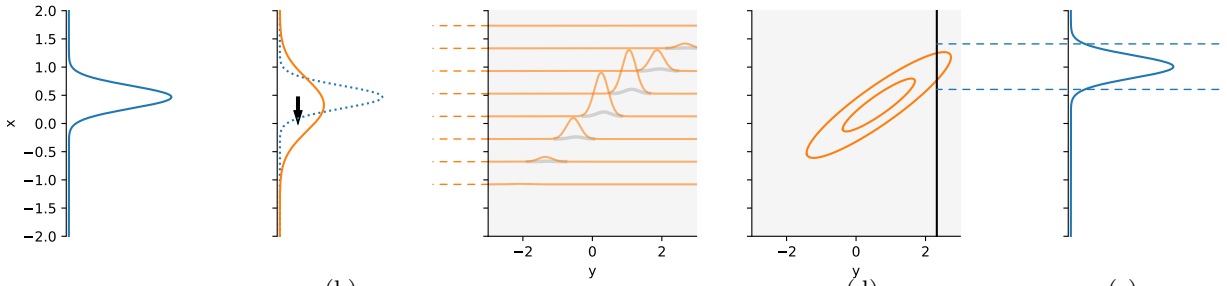
We then condition on the observed $\mathbf{y}_t$ using identity (A.4), which yields:

$$p(\mathbf{x}_t \,|\, \mathbf{y}_{1:t}) = \mathcal{N}(\mathbf{m}_t, P_t) \tag{A.27}$$

where

$$\begin{aligned} \mathbf{m}_t &= \mathbf{m}_t^- + K_t(\mathbf{y}_t - C\mathbf{m}_t^-), \\ P_t &= P_t^- - K_tCP_t^-, \\ K_t &= P_t^-C^\mathsf{T}(CP_t^-C^\mathsf{T} + S)^{-1}. \end{aligned}$$

Equations (A.25) and (A.27) define the Kalman filter updates, where K_t is the Kalman gain. These updates are often implemented in other forms to provide computational and numerical benefits (see e.g. Durbin and Koopman, 2012, §4.3, §6.3, Barber, 2012, §24.4) but the underlying idea remains the same. A visualisation of these steps is provided in Figure A-1 for 1-D data. The prediction step in Figure A-1(b) (orange) moves $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$ (blue) towards the origin, adding Gaussian noise. Some sample distributions for $\mathbf{y}_t$ conditioned on different values of $\mathbf{x}_t$ are given in A-1(c), with their peak height proportional to the prior probability. The level curves of the complete joint distribution of $p(\mathbf{x}_t, \mathbf{y}_t|\mathbf{y}_{1:t-1})$ are given in Figure A-1(d). The update step is shown in Figure A-1(d)-(e), where the



(a) $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$ **Previous filter.**

(b) $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$ **Prediction.** (A.25)

(c) $p(\mathbf{y}_t \mid \mathbf{x}_t)$ $\mathbf{y}_t$ **predictions given** $\mathbf{x}_t$.

(d) $p(\mathbf{x}_t, \mathbf{y}_t \mid \mathbf{y}_{1:t-1})$ **Joint.** (A.26)

(e) $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ **Update.** (A.27)

Figure A-1: Example of Kalman Filter update for 1D state and emission. Joint distribution shows 1,2,3 s.d. level curves; dashed blue lines show the values ± 2 s.d.

joint distribution (A.26) is conditioned on the observed $\mathbf{y}_t$ (shown with the vertical black line), resulting in the updated distribution of Figure A-1(e).

A.3 Multi-Task Dynamical Systems

This section contains some additional material primarily regarding the development of the MT-LDS parameterisation.

A.3.1 Parameterising using the Cayley transformation

We investigate the properties of the Cayley transformation[1] for 2-dimensional orthogonal matrices, since they are easily represented as rotation matrices (note the positive determinant implied by the Cayley transform). Let a 2-D skew-symmetric matrix S be parameterised by an input ψ:

$$\mathcal{S}(\psi) = \begin{bmatrix} 0 & -\psi \\ \psi & 0 \end{bmatrix}. \tag{A.28}$$

How does the input ψ affect the resulting matrix? Solving for ω in the orthogonal matrix Q using the Cayley parameterisation (eq. 3.8):

$$Q_{\text{rotation}}(\omega) = Q_{\text{cayley}}(\psi) \tag{A.29}$$

$$\Rightarrow \begin{bmatrix} \cos(\omega) & \sin(\omega) \\ -\sin(\omega) & \cos(\omega) \end{bmatrix} = (I - \mathcal{S}(\psi))(I + \mathcal{S}(\psi))^{-1} \tag{A.30}$$

$$\Rightarrow \omega = 2\arctan(\psi) \tag{A.31}$$

[1] While we have used the Cayley transformation to construct orthogonal matrices, alternative approaches are possible, such as the matrix exponential. See Everson and Roberts (1999) for an example application.

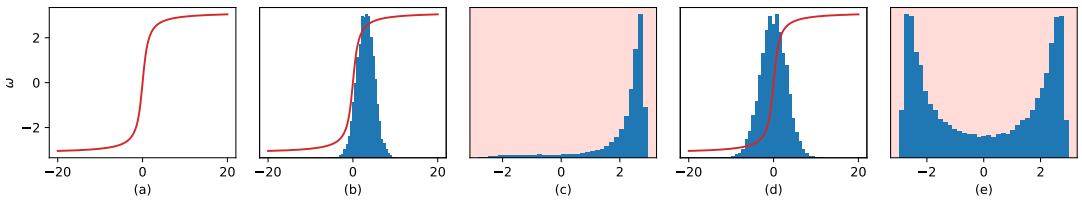


Figure A-2: (*a*) Relationship between input to 2D Cayley form (ψ) and resulting rotation ω in radians. (*b*) Example prior distribution in ψ – passing through nonlinearity (red) results in skewed distribution (*c*). (*d*) Example prior distribution in ψ – passing through nonlinearity (red) results in bimodal distribution (*e*).

(see appendix A.3.2 for a proof). Thus the relationship between the input value to the Cayley form and the implied rotation is sigmoidal in nature. For small angles, $|\omega| < \pi/4$ (i.e. less than 45 degrees) the function is fairly well approximated by the first order Maclaurin expansion $\omega = 2\psi$, and hence:

$$Q_{\text{cayley}}(\psi) \approx \begin{bmatrix} \cos(2\psi) & \sin(2\psi) \\ -\sin(2\psi) & \cos(2\psi) \end{bmatrix} \quad \text{for} \quad \psi \in [-\pi/8, \pi/8]. \tag{A.32}$$

Changing the frequency of oscillation is a global (not local) event, and hence the resulting transition matrix is extremely sensitive to small changes in ψ. (This assumes the model is using oscillations in the latent system to explain the observations as illustrated in Section 2.2.2.6.) Furthermore, density estimation in sequence space will be challenging, since larger values of ψ will begin to saturate the sigmoid (arctan) which can result in skewed or bimodal results (see Figure A-2 for an example).

It is therefore common that we should want the inputs to the Cayley transformation, ψ, to be small and have a tightly concentrated distribution. On the other hand, the diagonal matrix $\Sigma = \text{diag}\{\tanh(\boldsymbol{\upsilon})\}$ will often require the elements of $\boldsymbol{\upsilon}$ to be close to the interval $[3, 5]$ in order to ensure the magnitude of the latent space does not decay too fast (see Section 2.2.2.6). Due to the gradient of the sigmoid in this interval, the distribution over inputs may be quite wide. The result is a badly scaled optimisation problem, resulting in highly unstable / non-monotonic learning curves when using the methods detailed in Section 3.4.

An example of the learning curve of the (negative) marginal log-likelihood obtained while optimising a DHO model (Section 4.1.3, using the MCO objective) is shown in Figure A-3b. The blue curve shows the instabilities encountered during learning. Figure A-3a (bottom) shows the value of the gradient per iteration for one of the weights associated with the Cayley input variable. The weight experiences high magnitude oscillation in value, where large deviations appear to correlate with large divergences in the objective value. A simple (and effective) remedy has been to employ a constant scalar γ to shrink the inputs to the skew-symmetric $\mathcal{S}$, i.e. $\mathcal{S}(\gamma\psi)$. The value $\gamma = 0.1$ has worked well in practice (see for example the gradient values in Figure A-3a (top) and the optimisation

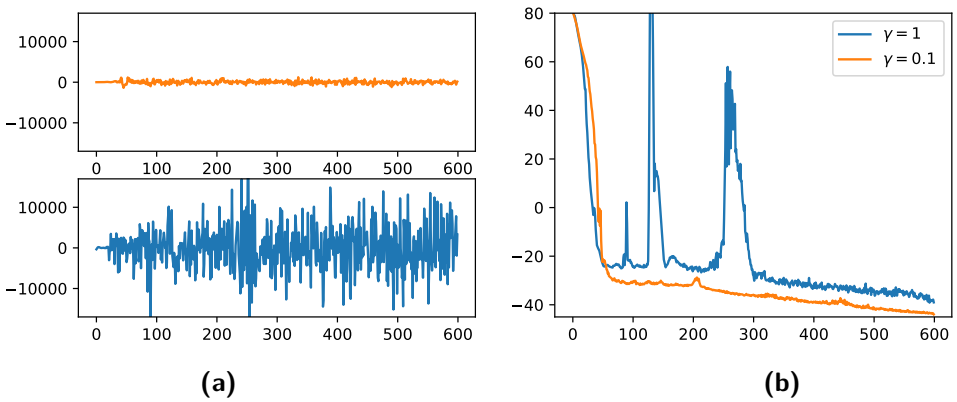


Figure A-3: Optimisation of a 2-D MT-LDS on a simple DHO problem (see Chapter 4), epochs on x axis. (a) The gradient of a single weight in the network responsible for ψ, bottom panel shows the values without the rescaling strategy ($\gamma = 1$), the top panel uses $\gamma = 0.1$. (b) The (negative) marginal log likelihood during optimisation both with ($\gamma = 0.1$) and without ($\gamma = 1$) the rescaling strategy.

trace for $\gamma = 0.1$ in Figure A-3b).

In all our experiments we use Adam (Kingma and Ba, 2014), which does not obviously adapt to the poor scaling of the optimisation problem.[2] One might instead consider using second-order optimisation methods. However, it is not straight-forward to use these together with stochastic gradient estimates, and it may also hurt generalisation performance. This is made more challenging due to the sequential nature of the problem (for an example method, see Martens et al. 2018). In principle one could instead reduce the overall learning rate to mitigate this problem, if one is willing to accept the increase in optimisation time. Nevertheless, even in this case, we have observed examples where diffusion sets in a long way from any optimum, and one might erroneously diagnose convergence. An example is given in Appendix A.3.3.

A.3.2 Proof of 2-D Cayley parameterisation of rotation matrix

The Cayley transformation of a 2D skew-symmetric matrix

$$\mathcal{S}(\psi) = \begin{bmatrix} 0 & -\psi \\ \psi & 0 \end{bmatrix} \tag{A.33}$$

results in a rotation matrix of angle $\omega = 2\arctan(\psi)$ for a skew-symmetric matrix, as stated in the previous section. (Note that both matrices only have one degree of freedom.)

[2] Note that while some have considered the use of squared gradients in Adam as an estimate of the (diagonal) inverse Fisher Matrix, it may be better understood as a variance adaptive trust region. Indeed, Balles and Hennig (2018) argue that Adam uses the *sign* of the gradient as the direction as opposed to any first or second order approximation.

Proof. The Cayley transformation of $\mathcal{S}$ is (eq. 3.8):

$$Q_{\text{cayley}}(\psi) = (I - S(\psi))(I + S(\psi))^{-1} = \begin{bmatrix} 1 & \psi \\ -\psi & 1 \end{bmatrix} \begin{bmatrix} 1 & -\psi \\ \psi & 1 \end{bmatrix}^{-1} \tag{A.34}$$

$$= \frac{1}{1+\psi^2} \begin{bmatrix} 1 & \psi \\ -\psi & 1 \end{bmatrix} \begin{bmatrix} 1 & \psi \\ -\psi & 1 \end{bmatrix} = \frac{1}{1+\psi^2} \begin{bmatrix} 1-\psi^2 & 2\psi \\ -2\psi & 1-\psi^2 \end{bmatrix} \tag{A.35}$$

Comparing to the rotation matrix

$$Q_{\text{rotation}}(\omega) = \begin{bmatrix} \cos(\omega) & \sin(\omega) \\ -\sin(\omega) & \cos(\omega) \end{bmatrix} \Rightarrow \omega = \arccos\left(\frac{1-\psi^2}{1+\psi^2}\right) \tag{A.36}$$

Using the identity $\arccos(z) = 2\arctan\left(\frac{\sqrt{1-z^2}}{1+z^2}\right)$, we have:

$$\omega = 2\arctan\left(\frac{\sqrt{1 - \frac{(1-\psi^2)^2}{(1+\psi^2)^2}}}{1 + \frac{1-\psi^2}{1+\psi^2}}\right) = 2\arctan\left(\sqrt{\frac{(1+\psi^2)^2 - (1-\psi^2)^2}{(1+\psi^2)^2}} \frac{1+\psi^2}{1+\psi^2+1-\psi^2}\right) \tag{A.37}$$

$$= 2\arctan(\psi). \tag{A.38}$$

□

A.3.3 Reducing Adam's step size to improve optimisation stability

We have noted in Section A.3.1 that scale differences can cause problems in optimisation. An obvious (if slow) solution is to reduce the step size to one which suits the smallest scale parameter in the model. We will consider this approach *instead* of scaling the inputs to the Cayley matrix as in Section A.3.1. The below explores a toy experiment that suggests this may be a poor strategy.

We choose a simple MT-LDS model class where the true generating process is known. Let ψ be the input to the orthogonal Cayley matrix, and s be the damping factor of the latent system. Following the Cayley parameterisation of eq. (3.8) gives us:

$$A(\psi, s) = (sI_2)\, Q(\psi) \tag{A.39}$$

$$\mathbf{x}_t = A(\psi, s)\, \mathbf{x}_{t-1}, \tag{A.40}$$

$$y_t = C\, \mathbf{x}_t \tag{A.41}$$

for $t = 1, \ldots, T$ where the parameters ψ, s are generated by:

$$\mathbf{z} \sim \mathcal{N}(0, I_2) \tag{A.42}$$

$$\psi = \tan(\omega/2); \qquad \omega = \left(10\,\sigma(wz_1) + 4\right)\frac{2\pi}{360} \tag{A.43}$$

$$s = \sigma(z_2 + b). \tag{A.44}$$

$\boldsymbol{\phi} = \{w, b\}$ are the learnable parameters, and C is considered fixed. We set the true parameters as $w^* = 1.7$ and $b^* = 5.0$.

The training set consists of $N = 100$ sequences of length $T = 80$ from the true parameters $\boldsymbol{\phi}^*$ with additive white noise of standard deviation 0.05. A model is initialised with parameters w_0, b_0 and optimised using the MCO method. We are interested in how the iterates $\boldsymbol{\phi}_k = (w_k, b_k)$ – the value of the parameters after the kth optimisation iteration – approach $\boldsymbol{\phi}^*$.

A.3.3.1 Experimental Results

Figure A-4 shows the progress of various optimisation runs using Adam for $\arg\max_{\boldsymbol{\phi}} \log p(Y; \boldsymbol{\phi})$ with the same initialisation. The different panels show step sizes $\eta \in \{10^{-3}, 5{\cdot}10^{-3}, 10^{-2}, 10^{-1}\}$ respectively. The only learning rate which reaches the optimum in less than 1 000 steps by Adam is $\eta = 10^{-1}$.[3] Table A.1 shows the average effective step size within each 200 step "chunk" of the 1000 step optimisation. We can see that for $\eta = 5 \cdot 10^{-3}$: the effective step size taken in the direction of b has reduced by a factor of 5 by $k = 800$, and one might easily imagine that it has converged. The learning rate $\eta = 10^{-2}$ performs better, but convergence to the optimum is still extremely slow. The fast convergence of $\eta = 10^{-1}$ suggests that large step sizes can be highly beneficial, and extremely slow convergence can be expected for small learning rates. Use of (Nesterov) momentum in preliminary experiments appeared not to help, perhaps because steep valleys and large gradients can be expected close to an optimum in dynamical models (Bengio et al., 1994; Pascanu et al., 2013). The re-scaling proposed in Section A.3.1 appears to be a superior strategy.

A.3.4 MT-LDS parameterisation

In this section, we will review how each of the parameterised matrices has been constructed in practice. We first discuss construction of the component parts and conclude with the implementation of the matrices A, B, R, S (we assume C and D are unconstrained).

A **skew-symmetric** matrix of order d requires a $d(d-1)/2$ dimensional vector of parameters $\mathbf{v}$. We reshape $\mathbf{v}$ into a strictly lower triangular matrix, an operation we will refer

[3] While $\eta = 10^{-3}$ is considered a good default choice, it would take 4150 steps to achieve the optimum even if all step sizes were of maximum length – which will not be the case. Default settings cannot be used blindly.

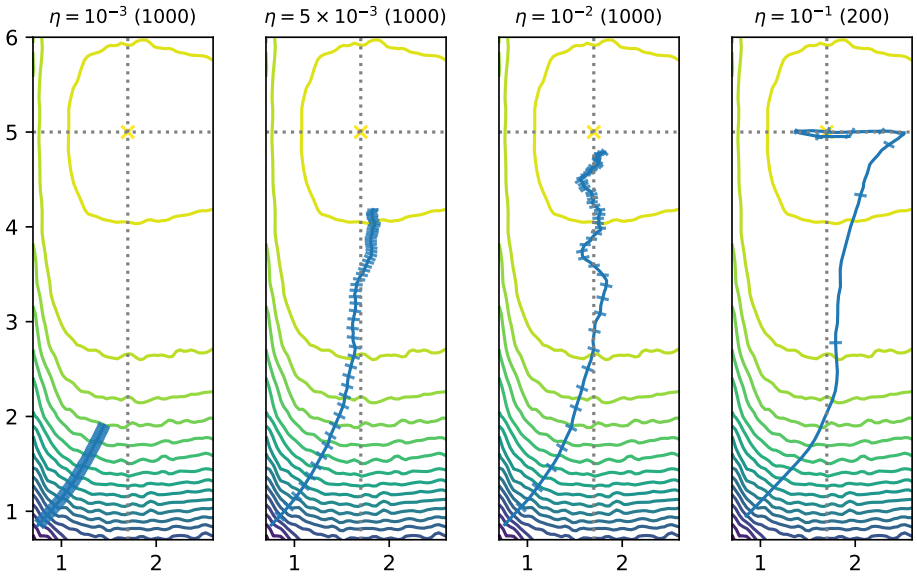


Figure A-4: Progress towards known optimum using Adam for simple model (eq. A.41). All panels show the contours of the objective function $\log p(Y)$ with w on the x-axis, and b on the y-axis, and the progress of an optimisation routine. The optimum is shown at (1.7, 5.0). The step size of each optimisation trajectory is given in the title, along with the number of steps taken. Function contours are estimated via Monte Carlo and smoothed using Gaussian convolution.

	Average $\hat{\eta}_b$ in chunk				
η	200	400	600	800	1000
10^{-3}	0.0011	0.0011	0.0011	0.0011	0.0010
$5 \cdot 10^{-3}$	0.0053	0.0051	0.0033	0.0018	0.0012
10^{-2}	0.0103	0.0051	0.0022	0.0013	0.0008
10^{-1}	0.0211	0.0022	0.0032	0.0038	0.0043

Table A.1: Average effective step size for b within the iteration chunks $k \in (0, 200], (200, 400], \ldots$ etc. for Adam optimiser with max. step size η shown.

to as 'tril', and add the resulting matrix to its negative transpose, i.e.

$$S(\mathbf{v}) = -\operatorname{tril}(\mathbf{v}) + \operatorname{tril}(\mathbf{v})^{\mathsf{T}}. \tag{A.45}$$

A **box-constrained diagonal** matrix of order d requires a d dimensional vector $\mathbf{v}$. Pass $\mathbf{v}$ through a sigmoidal function g such as σ, tanh or softsign[4] (Glorot and Bengio, 2010) to enforce the box constraints $\mathcal{B}$,[5] then project the vector onto the diagonal of a $d \times d$ matrix, i.e.

$$D_{\mathcal{B}}(\mathbf{v}) = \operatorname{diag}\left(g(\mathbf{v})\right). \tag{A.46}$$

Different sigmoid functions g will have different scale/sparseness properties due to different saturation characteristics.

A **diagonal covariance matrix** may be parameterised as above, but now with a diverging nonlinear function g such as exp or softplus.[6]

A **full covariance matrix** may be parameterised in various ways. We choose to parameterise via Cholesky factors, i.e. using the fact that a positive definite matrix C_{PD} may be decomposed as $C_{\mathrm{PD}} = LL^{\mathsf{T}}$ for lower triangular L. The decomposition can be made unique by enforcing a positive diagonal of the L. Hence for a $\frac{1}{2}d(d+1)$ dimensional parameter vector $\mathbf{v}$ we construct

$$L(\mathbf{v}) = \operatorname{tril}\left(\mathbf{v}_{1:d(d-1)/2}\right) + D_{[0,1]}(\mathbf{v}_{d(d-1)/2+1:d(d+1)/2)}), \tag{A.47}$$

$$C_{\mathrm{PD}}(\mathbf{v}) = L(\mathbf{v})L(\mathbf{v})^{\mathsf{T}}. \tag{A.48}$$

A.3.4.1 Implementation of MT-LDS parameters

A parameter vector $\boldsymbol{\theta}$ can be instantiated as a vector of parameters for various operations in a given model. We will define $\boldsymbol{\theta}_{\psi}$ to be the subset of the elements corresponding to a model parameter $\boldsymbol{\psi}$. For instance, in an MT-LDS with a full covariance matrix, $\boldsymbol{\theta}_R$ will be the $\frac{1}{2}m(m+1)$ elements corresponding to the positive definite covariance matrix R. We will also define the projection $\Pi^{(r,s)}$ to reshape a length rs vector into a $r \times s$ matrix.

Transition Matrix A**.** The vector $\boldsymbol{\theta}_A$ contains elements for the diagonal matrix and the orthogonal matrix which we denote $\boldsymbol{\theta}_{A_1}$ and $\boldsymbol{\theta}_{A_2}$ respectively.

$$A(\boldsymbol{\theta}) = D_{[0,1]}(\boldsymbol{\theta}_{A_1})(I - S(\boldsymbol{\theta}_{A_2}))(I + S(\boldsymbol{\theta}_{A_2}))^{-1}. \tag{A.49}$$

If not stated otherwise, we use $g = \sigma$, the logistic sigmoid, within $D_{[0,1]}(\cdot)$.

[4] $\operatorname{softsign}(x) := x/(1+|x|)$

[5] Since box constraints in d-dimensions are isomorphic to each other, we will assume that the necessary affine transformations are applied to the image of the chosen sigmoid function.

[6] $\operatorname{softplus}(x) := \log(1+\exp(x))$

Input Matrix B. The vector $\boldsymbol{\theta}_B$ contains elements for the sigmoid and tanh operations which we denote $\boldsymbol{\theta}_{B_1}$ and $\boldsymbol{\theta}_{B_2}$ respectively.

$$B(\boldsymbol{\theta}) = \sigma\left(\Pi^{(d,s)}\boldsymbol{\theta}_{B_1}\right)\tanh\left(\Pi^{(d,s)}\boldsymbol{\theta}_{B_2}\right), \tag{A.50}$$

where σ and tanh operate elementwise on matrix inputs.

Covariance Matrix R. This follows the standard positive definite construction of eq.s (A.47), (A.48):

$$R(\boldsymbol{\theta}) = L(\boldsymbol{\theta}_R)L(\boldsymbol{\theta}_R)^{\mathsf{T}}. \tag{A.51}$$

Covariance matrix S is implemented equivalently.

A.4 Application to Damped Harmonic Oscillation

A.4.1 Prior density assumed for ρ

The explicit density of ρ can be calculated analytically. Let the uniform distribution over ν be $p_\nu(\nu) = \mathrm{U}[a,b]$ and the change of variables be $\rho = g(\nu) = \exp\{-\log(2)/\nu\}$. We have via change of variables (see e.g. Casella, 1985, Ch. 2) that $p(\rho) = \left|\frac{\mathrm{d}}{\mathrm{d}\rho}g^{-1}(\rho)\right| p_\nu(g^{-1}(\rho))$. Since $g^{-1}(\rho) = -\log(2)\log^{-1}(\rho)$ for $\rho > 0$, and $\frac{\mathrm{d}}{\mathrm{d}y}g^{-1}(\rho) = \frac{\log(2)}{\rho\log^2(\rho)}$, we have

$$\begin{aligned} p(\rho) &= \left|\frac{\log(2)}{\rho\log^2(\rho)}\right| \frac{1}{b-a}\mathbb{1}\left\{-\log(2)\log^{-1}(\rho) \in [a,b]\right\} \\ &= \frac{\log(2)}{b-a}\frac{1}{\rho\log^2(\rho)}\mathbb{1}\left\{\rho \in \left[\frac{1}{2^{1/a}}, \frac{1}{2^{1/b}}\right]\right\} \end{aligned}$$

since $\frac{\log(2)}{\rho\log^2(\rho)}$ is always positive for $\rho \in (0,1)$. We presume we can exclude the endpoints of the interval since they correspond to a half-life of zero and infinity respectively. For our specific choice of (half-life) prior, $p_\nu(\nu) = \mathrm{U}[4,80]$, we have

$$\approx \frac{\log(2)}{76}\frac{1}{\rho\log^2(\rho)}\mathbb{1}\left\{\rho \in [0.8409, 0.9914]\right\}.$$

A.4.2 Optimisation parameters

We use the Adam optimiser (Kingma and Ba, 2014) initially with a max step size of 3×10^{-3}, with the default moving average parameters of $\beta_1 = 0.9, \beta_2 = 0.99$. In each epoch, $M = 800$ samples were taken for the (shared) MCO proposal. Each epoch contained 20 iterations (one per sequence, although all are updated in each iteration), and the learning rate is decayed by 0.99 at the end of each one in a geometric decay schedule. Each sequence

Epochs	η	decay	$\log\sigma$ mean	M	M_{rsmp}
250	3×10^{-3}	0.99	n/a	800	1
50	$2.4 \times 10^{-4}(*)$	0.99	-2.0	2 000	3

Table A.2: Double DHO schedule. (*) Effective learning rate after 250 epochs of the decay schedule.

initially resamples a single posterior sample ($M_{rsmp} = 1$) from the prior to estimate its gradient. After 250 epochs, the model is usually within a useful basin of convergence, and we switch to 50 epochs of a more intensive schedule (larger M, larger M_{rsmp}) to ensure lower variance updates. It is at this stage where it has proved important to enforce the (soft) constraint of $\sigma = 0.2$ via a prior. These details are summarised in Table A.2. The single DHO models are easier to train, and in particular, require fewer MCO samples. In all cases, this training schedule has proved highly reliable.

A.4.3 Inference parameters for DHO Predictions

MT-LDS We use the AdaIS scheme described in Section 3.5.1. The proposal distribution q_{prop} is a $k = 3$ component Gaussian mixture model (GMM) refined over up to 7 iterations for each t. All algorithm parameters are given in Table A.3. As stated in the main text, we additionally inferred the emission standard deviation σ. However, inferring σ close to its true value of 0.05 is extremely challenging for any method (including MCMC methods) since the posterior of a sequence often concentrates close to a lower dimensional manifold. This results in high condition numbers / numerical issues for any covariance calculations (required for estimating the GMM during AdaIS or for the mass matrix in HMC). We imposed a prior of $\log\sigma \sim \mathcal{N}\left(-2.0, 0.1^2\right)$ (note that $\sigma = \exp(-2) \approx 0.13$) to stabilise the problem. We have found the procedure to be robust in practice, only twice failing to produce samples with ESS > 100 out of almost 1000 runs.[7]. These two failures were alleviated by choosing a slightly higher prior mean of $\log\sigma$. We also experimented with a more sophisticated variant: Adaptive *Multiple* Importance Sampling (Corneuet et al., 2012) but did not see any improvement.

ST-LDS Inference of the single task models is performed using the NUTS criterion (Hoffman and Gelman, 2014) for Hamiltonian Monte Carlo (HMC). Tuning is performed using ideas from Hoffman and Gelman (2014, Algorithm 4, 5), and the mass matrix is estimated from the warmup phase.[8] We initialised samplers from the MAP value, obtained via optimisation with 10 random restarts. The burn-in/tuning phase used 1000 samples and the subsequent 600 samples were used for inference. For both MAP optimisation and sampling, we found it essential to enforce a low standard deviation (we used $\sigma = 0.2$

[7] The inner loop of AdaIS is used $20 \times 3 \times 16 = 960$ times (# sequences × # models × # timepoints per sequence).

[8] Implementation `https://github.com/tpapp/DynamicHMC.jl`, author Tamas K. Papp.

Parameter	Description	Value
k	Num. mixture components	3
iters	Max num. of adaptive IS iterations	7
nsmp	Num. samples per IS iteration	1 000
init_nsmp	Num. samples for first proposal	3 000
final_nsmp	Num. samples for final proposal	3 000
tilt	Exponential tilt of proposal	2.0
min_ESS	Minimum eff. sample size	100
n_retry	Num. retries if ESS $<$ min_ESS	2
EM_iters	Num. EM iters in GMM fit	3
kmeans_iters	Max. kmeans iters for init	100

Table A.3: DHO Inference parameters

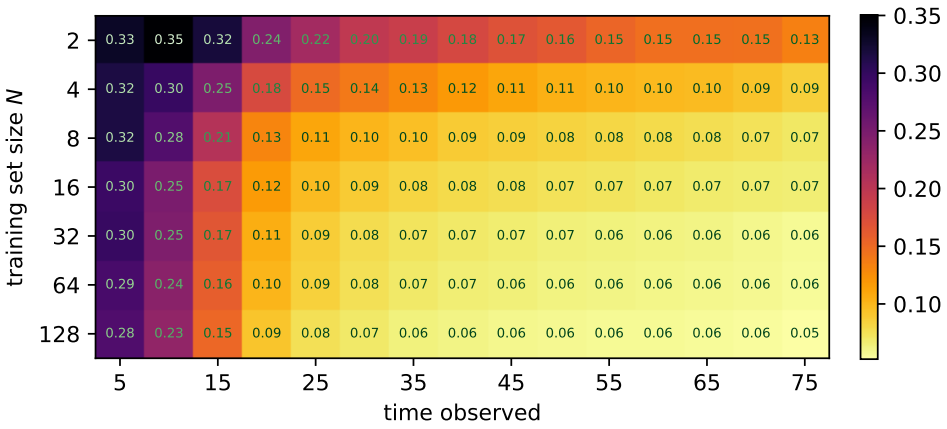
and $\log \sigma \sim \mathcal{N}\left(-2, 0.2^2\right)$ respectively) similarly to the MTL experiments. The choice of 600 samples represents a trade-off between accuracy and runtime reduction. (We wish to benchmark the runtime as well as the performance of each method.)

A.4.4 Results for prediction tasks

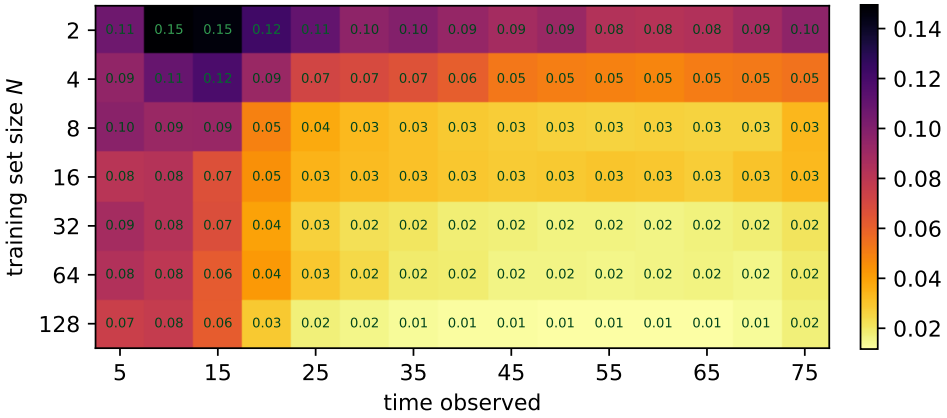
Additional experiments were run for the MT-LDS on the double DHO data (beyond those stated in the main text). These were to understand the sensitivity on the choice of training set size $N \in \{2, 4, 8, 16, 32, 64, 128\}$ and the amount of time observed $t \in \{5, 10, 15, \ldots, 75\}$. The RMSE of all combinations of these values is shown in A-5a and the standard deviation across the 5 replications is shown in Figure A-5b. It is perhaps impressive that the MT-LDS can perform close to optimally on 20 test examples having seen only 8 examples in training.

A.4.5 Necessity of GMM proposal for inference

Learning an MTDS generally results in much simpler posteriors than one would find from performing inference on the parameters of the base model directly (e.g. on a small LDS model). Both MCO and variational methods of learning regularise the posteriors on the training set – with the variational method placing even greater restriction on the form of posterior. Nevertheless, the resulting posteriors are often non-Gaussian – especially on novel test sequences – and using a Gaussian posterior or proposal distribution may perform poorly at test time. A fairly typical example from the double DHO experiments (using an MT-LDS) is shown in Figure A-6.



(a) Average of predictive RMSE by training dataset size and time observed.



(b) Standard deviation of predictive RMSE by training dataset size and time observed.

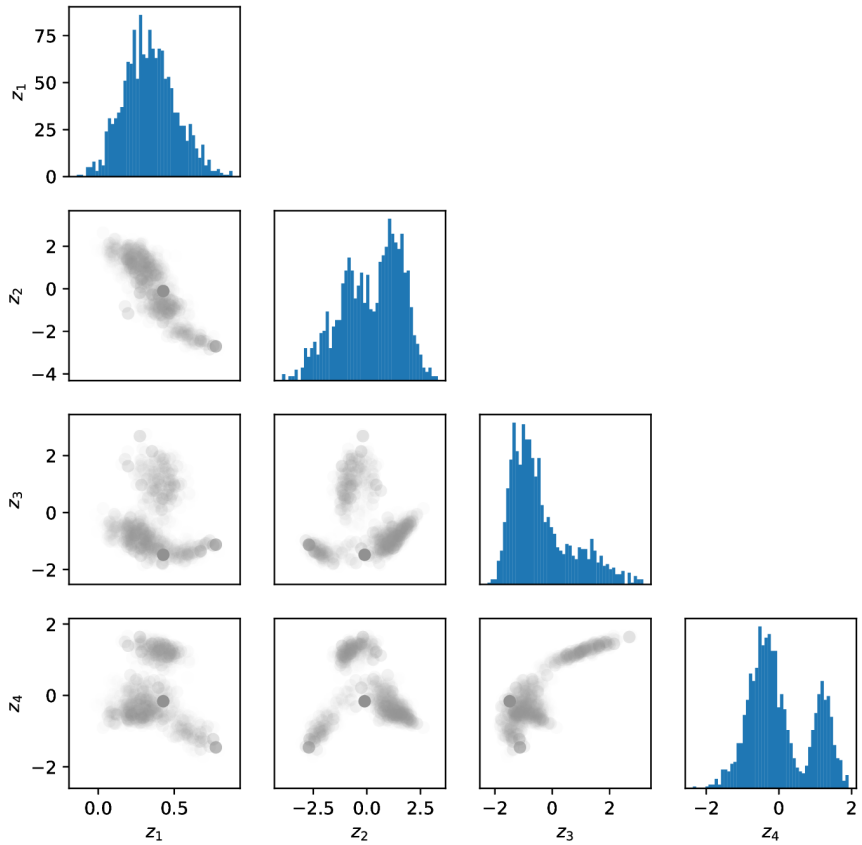


Figure A-6: Pairwise scatter plot of posterior samples for double DHO data using an MT-LDS model trained by MCO (at $t = 30$). The importance weight is indicated by transparency. The diagonal shows a marginal histogram of the samples for each dimension of $\mathbf{z}$.

A.5 Application to Human Locomotion

A.5.1 Data preprocessing

There was a substantial amount of pre-processing for the mocap data. The observations $\{\mathbf{y}_t\}$ broadly followed the ideas in Mason et al. (2018). However a proportion of this is implemented in a large C# codebase written for the Unity game engine, which makes use of various internal representations that are poorly documented. The implementation used here (currently at https://github.com/ornithos/mocap-mtds) attempts to perform the geometric and forward/inverse kinematic transformations based on the available descriptions in various papers/code including Holden et al. (2016), Mason et al. (2018), De la Torre et al. (2009).

Calculating the inputs $\{\mathbf{u}_t\}$ was especially difficult, and involved solving some issues re-

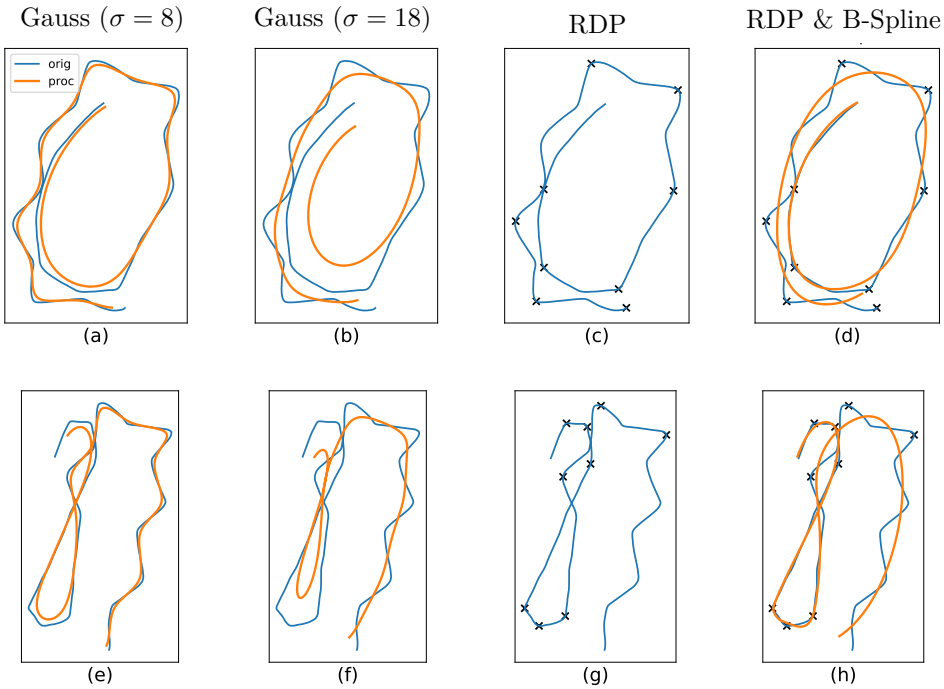


Figure A-7: Comparison of trajectory smoothing methods. Each row (a-d), (e-h) shows a different trajectory segment in blue, with the results of different methods (Gaussian filtering with bandwidths $\sigma = 8, 18$; RDP, and RDP+spline) shown in orange.

garding information leakage, as well as automatic annotation of foot placement and reverse corner rotation. Details of each are given below.

A.5.1.1 Removing information leakage from the root trajectory

The root trajectory is computed by projecting the root joint onto the ground. However, this projection leaks information about the style of locomotion, for instance via swaying. We wish to remove all such information, since a model can otherwise learn the style from the $\{\mathbf{u}_t\}$ and ignore the latent $\mathbf{z}$. Our goal is to find an *appropriately smoothed* version of the extracted trajectory $\mathcal{T}$.

Figure A-7 provides two example trajectories (blue) which contain information about the locomotion style. In this case the low frequency swaying suggests that the trajectory is from the 'sexy' style. In order to remove this information leakage, we must smooth over this trajectory, without destroying any information about the underlying path. The example shown here is particularly challenging, since the swaying motion is of a similar magnitude to some of the real features of the data. We cannot use simple smoothing methods; for instance a Gaussian filter[9] results either in undersmoothed corners (Figure A-7 (a), (e) using $\sigma = 8$) or oversmoothed corners (Figure A-7 (b), (f) using $\sigma = 18$).

[9] i.e. convolving with a Gaussian kernel (see e.g. Prince, 2012, §13.1.3).

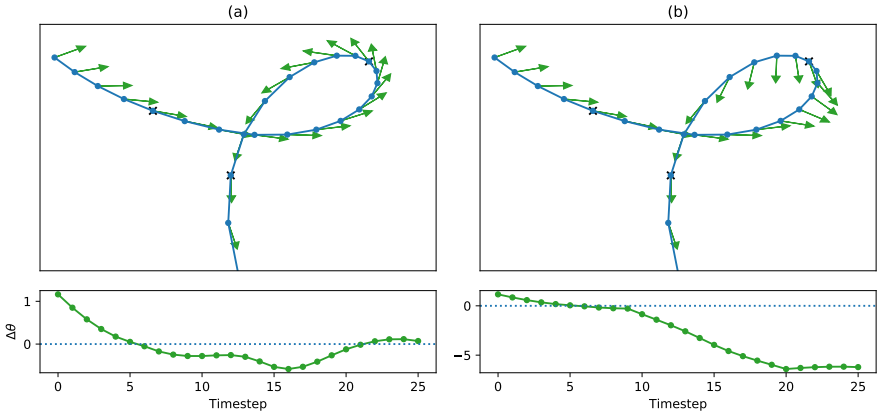


Figure A-8: Comparison of (a) normal corner rotation vs. (b) reverse rotation. The trajectory is shown in blue in the top panel in the xz plane, with the body direction shown in green. The bottom panel shows the difference between the angle of the body and the angle of the trajectory.

We instead use a heuristic approach which first attempts to find the true corners of the motion, and then uses these as control points in a cubic B-spline fit. In order to find these 'corners', we fit a polygonal approximation to $\mathcal{T}$ using the Ramer-Douglas-Peucker algorithm (RDP, e.g. Ramer, 1972). The RDP algorithm seeks a polygonal approximation to a curve via a divide-and-conquer approach which greedily chooses points that minimise the Hausdorff distance. Figure A-7, panels (c), (g) shows the control points chosen by the RDP algorithm for two different trajectories. A B-spline is then fitted to $\mathcal{T}$ with these control points using a least squares criterion. Some per-style tuning of the RDP parameter ϵ, and a small number of manually added control points rendered this a semi-automatic process.

A.5.1.2 Extracting the gait phase

Foot contacts are calculated via the code used by Holden et al. (2017), which is based on thresholding of vertical position and velocity of each foot. As in Mason et al. (2018) we check visually for outliers and correct misclassified foot contacts manually. The leading edge of each foot contact is taken to represent 0 (left) and π (right), and the gait phase is calculated by interpolation.

A.5.1.3 Annotating reverse rotations

To establish whether the body has turned the reverse direction around a corner, one can investigate the relative angle the body makes with the root trajectory. The direction of the body is calculated via the normal of the plane defined by the two shoulders (smoothed over time). See Figure A-8a, where the green arrows indicate the body direction, and the relative angle ($\Delta\theta$) is the angle the body direction makes with the trajectory.

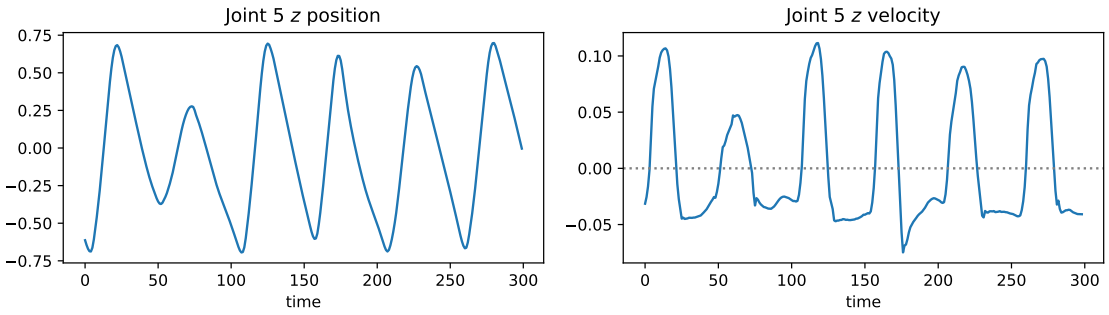


Figure A-9: Left toe joint position and velocity (first difference) for subset of "angry" style.

When the body turns *with* the corner, this relative angle, $\Delta\theta$, will be relatively low. Figure A-8a demonstrates this case, where the relative angle is shown in the bottom panel. On the other hand, if the body turns in the reverse direction, this ultimately results in a 2π radian relative rotation. This scenario is shown in Figure A-8b. In this case we highlight the entire section preceding the corner with the boolean indicator, where the RDP algorithm is used to partition the trajectory into segments.

A.5.2 Models

This section includes some additional details regarding the MTDS and competitor models used in the experiments.

Difficulty of using LDS The main text mentions problems of highly nonlinear relationships with the trajectory when turning corners, as well as sequences which cannot be efficiently modelled using an LDS. As an example of the latter, see Figure A-9, the z position of the left toe, which follows an almost sawtooth-like pattern. The first difference is shown on the right which demonstrates the difficulty of modelling via a combination of linear functions.

MTDS $\mathbf{h}_\phi$ The MLP variant of $\mathbf{h}_\phi$ (which was used for comparison in preliminary experiments) used one hidden layer with 300 hidden units and tanh activations. The final affine layer used a rank 30 matrix, since $[\boldsymbol{\psi}_2, C_2, \mathbf{d}]$ comprise almost 25 000 dimensions (see calculation in the main text), and a full rank final layer would thus require over 7 000 000 tunable parameters (cf. Section 3.6).

A.5.2.1 Learning

All models use a sequence prediction length of $L = 64$ and a batch size of $N_{\text{batch}} = 16$. Since the max. sequence length $L = 64$ is relatively short, truncated backpropagation through time did not prove necessary.

Model	Optimiser	η	Multi-task η	Regularisation
MT-RNN	Adam	3×10^{-5}	1×10^{-3}	1×10^{-2}
GRU-1L (closed loop)	Adam	5×10^{-4}	-	5×10^{-4}
GRU-2L (closed loop)	Adam	1×10^{-4}	-	5×10^{-4}
GRU-1L (open loop)	Adam	5×10^{-4}	-	0
GRU-2L (open loop)	Adam	1×10^{-4}	-	0

Table A.4: Hyper-parameters of mocap models. η denotes the learning rate.

MTDS The qualitative investigation into learning smooth animations and style transfer motivated split learning rates between shared and multi-task networks (Section 5.1.2.1), and the amount of L2 regularisation. See Table A.4 for the chosen values. The main learning rate η applies to the fixed parameters wrt. $\mathbf{z}$ (i.e. $\boldsymbol{\psi}_1, H$), and the multi-task learning rate applies to the parameter generation parameters $\boldsymbol{\phi}$ and inference parameters $\boldsymbol{\lambda}$. Standard variational inference proved more reliable than amortised inference: we used a Gaussian with diagonal covariance (parameterised using softplus) for the variational posterior over each $\mathbf{z}$. L2 regularisation was applied to $\boldsymbol{\phi}, \boldsymbol{\psi}_1, H$.

Each model was optimised for 20 000 iterations. The ELBO had often reached a plateau by this time, and training longer often resulted in a worse latent representation (as evidenced through poor style transfer). As noted in the main text, we perform a form of KL annealing in order to learn a descriptive latent $\mathbf{z}$. This is performed by removing the KL penalty of the ELBO ($\mathrm{KL}\left(p(\mathbf{z}|\mathbf{y}) \,\|\, p(\mathbf{z})\right)$) for the initial 2 000 iterations, and enforcing a small posterior standard deviation ($\mathbf{s}_{\boldsymbol{\lambda}} = 10^{-3}$) for the same duration. This is similar to finding a MAP estimate for the $\{\mathbf{z}\}$. For the remaining iterations, the original ELBO criterion is used, and $\mathbf{s}_{\boldsymbol{\lambda}}$ is initialised from 10^{-3}. The model is implemented in PyTorch (Paszke et al., 2017) and trained on a GPU.

GRU benchmarks These models (in contrast to the MTDS) predict the difference (or 'velocity') from the previous frame via a residual architecture, as this performs better in Martinez et al. (2017). Hyperparameters were chosen via grid search over learning rate ($\eta \in \{1 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}, 5 \times 10^{-3}, 1 \times 10^{-2}\}$), regularisation ($\gamma \in \{1 \times 10^{-5}, 1 \times 10^{-4}, 1 \times 10^{-3}, 1 \times 10^{-2}, 1 \times 10^{-1}\}$), and optimisers {Adam, SGD}. Once the hyperparameters were chosen (Table A.4), the benchmark models were trained for 20 000 iterations, recording the validation error every 1 000 iterations on a stratified 12.5% held out sample. The model with the lowest validation error during optimisation was chosen.

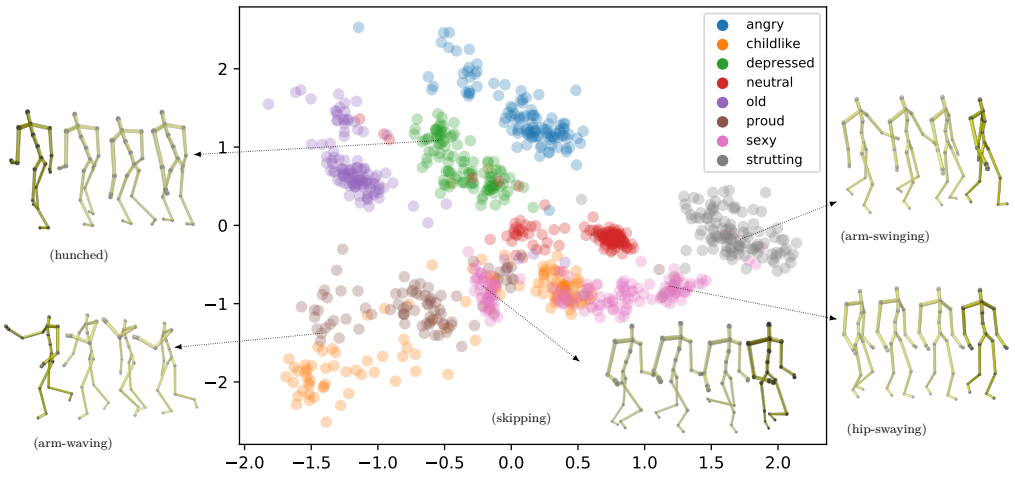


Figure A-10: $k = 2$ mean embedding of each sequence segment, coloured by its (unseen) task label.

A.5.3 Experiments

A.5.3.1 Qualitative Results

We have seen a t-SNE plot of the $k = 8$ embedding in the main text, which shows that the MTDS can separate all sub-styles into different clusters. Figure A-10 provides a raw embedding for when $k = 2$. There are apparently insufficient degrees of freedom to separate all styles[10], so some (in particular, some sub-styles of 'sexy' and 'childlike') are amalgamated. Nevertheless, given the similarity of the styles, this is broadly sensible.

Figure A-11 compares the t-SNE embeddings between the MTDS and MT-Bias models for a variety of perplexity parameter values. Clusters only become visible by perplexity $p = 5$; the MTDS maintains well-defined homogeneous clusters throughout, but in the MT-Bias case there are many clusters which 'bleed into each other'. This means that some differing styles are closer to each other in latent space than some instances of the same style. This problem is visible across all larger values of perplexity, where 'childlike', 'sexy' and 'proud' have substantial overlap, likewise do 'depressed', 'old' and 'angry'.

A.5.3.2 Data efficiency

Data The training data for each style uses 4 subsequences chosen carefully to represent the inter-style variation. Obviously it is important that frames are consecutive rather than randomly sampled. Over the increasing size training sets, each of these subsequences is a superset of the previous one. The 6 training set sizes ($2^8, 2^9, 2^{10}, 2^{11}, 2^{12}, 2^{13}$ frames per style) are not exact since short subsequences are discarded (e.g. at file boundaries), and the largest set contains all the training data except the test set[11], where data are not

[10] Every model run using $k = 8$ produced distinct clusters for all styles, every run using $k = 2$ resulted in amalgamating some styles.

[11] This final set averages $7680 \approx 2^{12.9}$ frames per style.

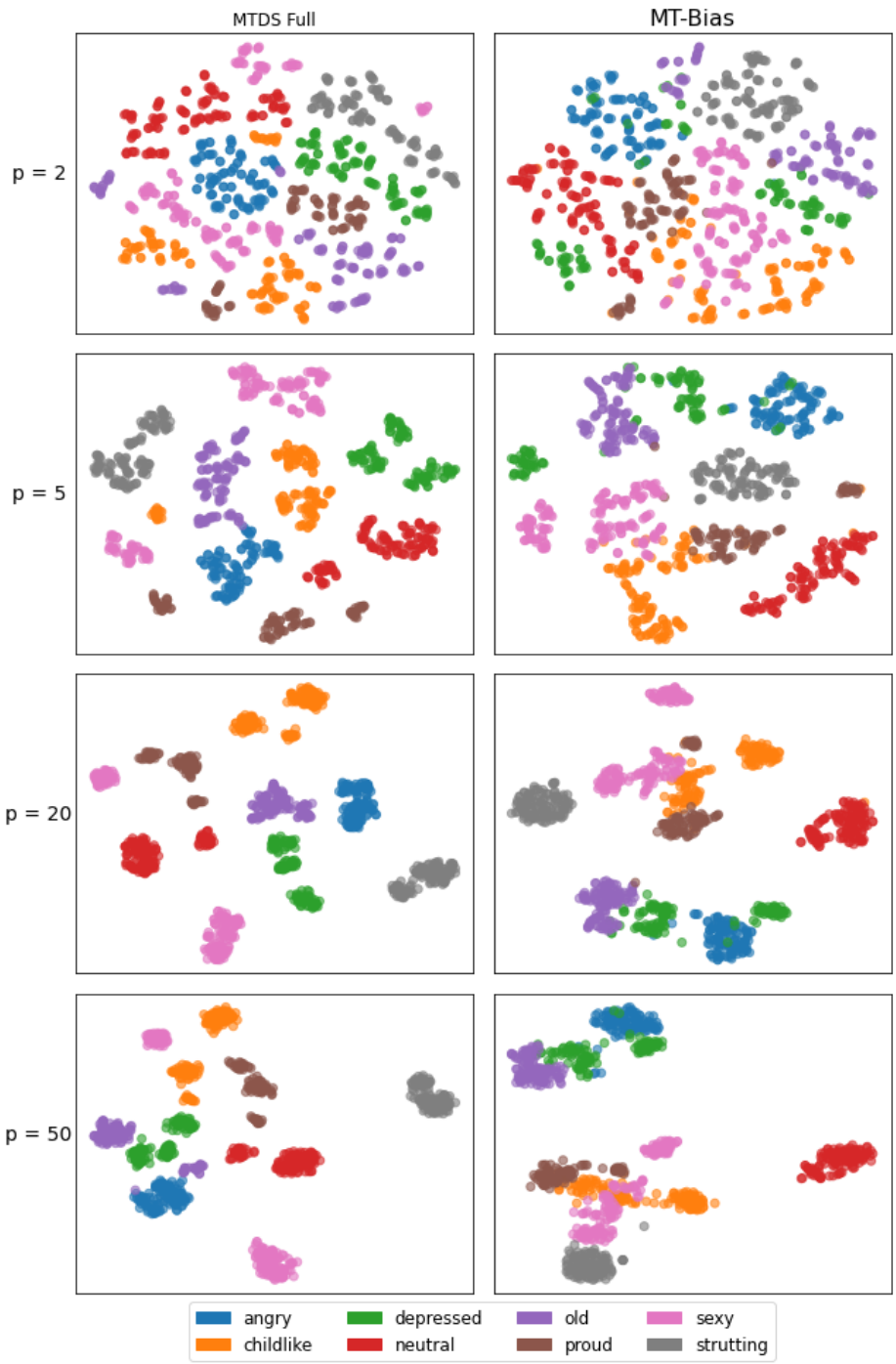


Figure A-11: Mean embedding of latent $\mathbf{z}$ coloured by true style of the proposed MTDS model vs. a MT-Bias approach. Figure shows t-SNE plot for perplexity 2,5,20,50. Original data have $k = 8$ dimensions and standardised to have zero mean, unit variance.

evenly distributed over styles. The test set comprises 4 sequences from each style, each of length 64, and is the same for all experiments. A length-64 seed sequence immediately preceding each test sequence was used for inference for all models.

Single task models The STL models use an identical architecture to the pooled 1-layer GRU models, except they are trained only on the data for their style. Since there is less data for these models, we train them for a maximum of 5 000 iterations. We omit 2-layer GRUs for this experiment since the amount of data is mostly small.

Per-style results The results in the main text are averaged over all eight styles. Figure A-12 provides graphs of results for each individual style. The graphs here also provide the result for the final training set size (97% of the data) for transparency, but we are most interested in the results for the smaller datasets. For the initial three datapoints (3%, 7%, 13%), the MTDS model provides equal[12] or better results for *all* styles. Note that the 'angry' and 'childlike' styles appear to be harder than the others, most likely due to their relatively high speed.

A.5.3.3 Novel test data

Results for all τ-step ahead errors Figure A-13 shows the τ-step errors for all $\tau \in 1, \ldots, 320$. We can see that the predictions of Pooled GRUs (only implicitly multi-task) deteriorate rapidly as τ increases, although the closed loop variant performs markedly worse, with an almost exponentially increasing error over τ. The open-loop models deteriorate roughly linearly with τ, and hence are fairly well represented by the truncated graph in the main text.

The improvement of the MTDS over the pooled GRU approaches is sustained over almost all styles, except at the very beginning, as discussed in the main text, and can be seen in Figure A-14. The only exception is style 8 ('strutting') for which the MTDS is not clearly better until after about $\tau = 75$ prediction steps.

A.5.3.4 Style transfer

The MTDS and MT-Bias models used for these experiments use a $k = 8$ latent code (due to the 8 classes) and are trained on all styles.

Classifying the style The classifier is learned on the original observations to distinguish between the 8 styles. We use a 512-unit GRU to encode an observation sequence (usually of 64 frames), and transform the final state via a 300-unit hidden layer MLP with sigmoid activations into multinomial emissions. The model is trained via cross-entropy with 20% of

[12] In the sense of statistically indistinguishable at an $\alpha = 0.05$ level.

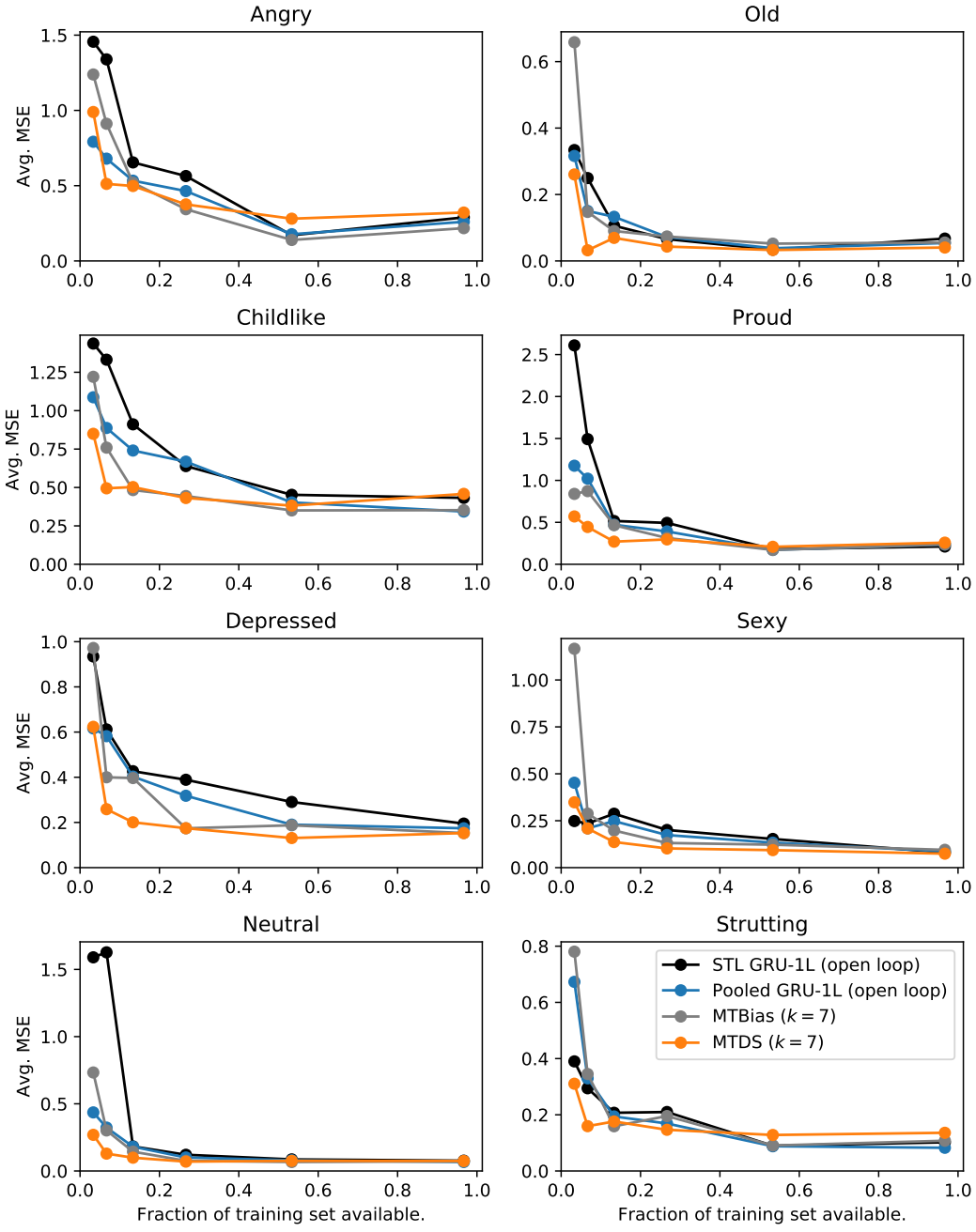
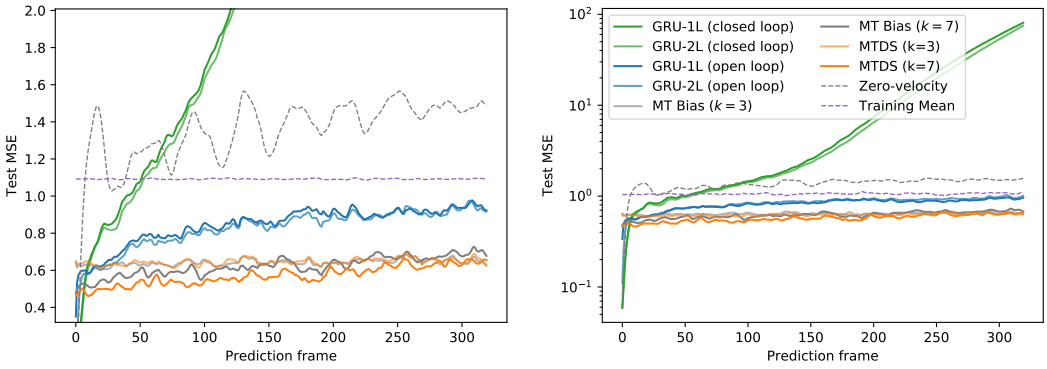


Figure A-12: Per style MSE of Experiment 1 plotted against the fraction of training data available.



(a) **(b)**

Figure A-13: Results for Experiment 2 for all models on (a) truncated scale, (b) log scale.

the training data held out as a validation set; training was stopped as the validation error approached 0. We perform a standardisation of the gait frequency across all styles, since some styles can be identified purely by calculating the frequency. The mean frequency across all styles (1 cycle per 33 frames) is applied to all sequences via linear interpolation. In this we make use of the instantaneous phase given in the inputs.

Experimental setup The experimental setup is as follows. We carefully choose four segments of length 64 for each of the styles $s_1 = 1, \ldots, 8$ which represent the variability within each style. These correspond to the inputs $U_j^{(s_1)}$ for each source style s_1, with examples $j = 1, \ldots, 4$. We next seek the 'archetypal' latent code $\mathbf{z}$ associated with each target style s_2. For each s_2, we optimise $\mathbf{z}$ over 20 candidate values, for which we use the posterior means of examples of style s_2 in the training set. Data are generated from all $\{U_j^{(s_1)}\}$ and the $\mathbf{z}$ which provides the greatest success in style transfer is chosen. The 32 highly varied input sequences guard against overfitting – the 'archetypal' codes for each style must perform well across much of the variety of the original dataset. We provide a scalar measurement of the 'success' of style transfer for each pair (s_1, s_2) by using the resulting 'probability' that the classifier assigns the target style s_2, averaged across the four input sequences for the source s_1.

A.5.4 Visualisation tools

Unlike standard work in 3D Graphics, we used a more lightweight solution to visualisation than large frameworks such as e.g. Blender or Unity. The visualisations here are via use of three.js via the interface provided by MeshCat.jl and WebIO.jl. This allowed visualisations to be piped to a local web browser directly from Julia code and investigated interactively. The skeleton was a custom implementation consisting of cylinders and lines programmed to follow the model output. Due to the interface provided by MeshCat.jl, each shape had

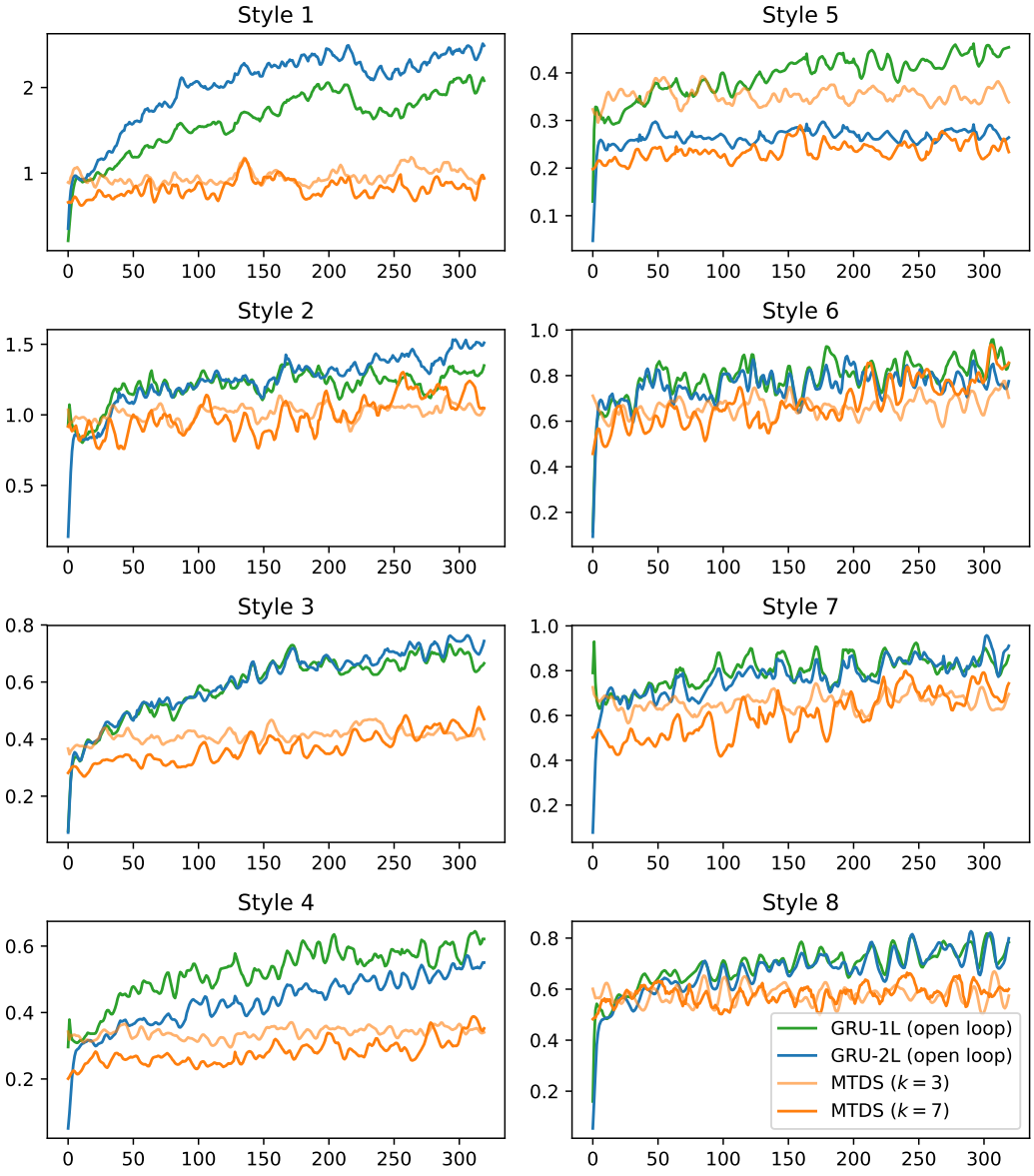


Figure A-14: Per style MSE of Experiment 2 plotted against time.

to be mapped via 3D transformation from the origin for each frame. Finally, a camera was programmed to follow a smoothed path behind the modelled trajectory.

A.6 Application to pharmacodynamic modelling

A.6.1 Background - personalisation of PK models

In this section, we review the modelling details of three important propofol PK studies. The resulting parameter estimates in each case are shown in Table A.5.

- **Marsh model**. The Marsh model (Gepts et al., 1987; Marsh et al., 1991) was an early clinical study of 18 Caucasian adults (5 female, 13 male) estimating the volumes and clearances within the 3 compartment model. The data were fitted by a triexponential model using a nonlinear least squares fit. Calculated *volumes* were found to have a significant relationship with weight, however, individual pharmacokinetic parameters did not correlate with weight, and no other comparisons were mentioned in the original paper. The effect of volume parameters in this model affects only the normalisation of the infusion rate $u(t)$.
- **Schnider model**. The Schnider Model (Schnider et al., 1998) was the first study to systematically estimate Propofol PK parameters from patient covariates. The clinical study involved 23 patients (11 female, 12 male), stratified within ages 18-34, 35-65, 66+; no details of race are given. Compartment models (with two and three compartments) were fitted to each patient via `NONMEM`, using three volume and three clearance parameters, each of which had lognormal random effects per patient. The fitted (inferred) parameter values per patient were then extracted and predicted using a Generalised Additive Model (GAM) from the patient covariates, and after variable selection, the resultant features were used in a final model. `NONMEM` was used again to calculate these coefficients directly.
- **White model**. The White model (White et al., 2008) is a larger study of 113 patients (59 female, 54 male; no details of race were given) which provides a further personalisation of the Marsh Model. Based on a heuristic threshold from the diminishing gains of maximising model likelihood, only the central compartment volume and clearance rate were optimised.

A.6.2 Data processing

We use the data from Georgatzis et al. (2016), but nonetheless, the data required substantial processing. The following sections describe the munging and pre-processing performed; this will unfortunately not be exhaustive, but we provide a write-up where we have records. We believe this is a useful practice for future work.

A.6.2.1 Data munging

In this section we make a record of the data sources and any pre-processing involving *extracting, aligning, and merging.*

Parameter	Marsh	Schnider	White
v_1	$0.228\ l\,\mathbf{kg^{-1}}$	$4.27\ l$	$0.176 + 4.6 \times 10^{-5} \times \mathbf{age} \times \mathbf{male} + 0.192 + 6.7 \times 10^{-4} \times \mathbf{age} \times \mathbf{female}\ l\,\mathbf{kg^{-1}}$
v_2	$0.463\ l\,\mathbf{kg^{-1}}$	$18.9 - 0.391\,(\mathbf{age} - 53)\ l$	$0.463\ l\,\mathbf{kg^{-1}}$
v_3	$2.893\ l\,\mathbf{kg^{-1}}$	$238\ l$	$2.893\ l\,\mathbf{kg^{-1}}$
k_{10} (min^{-1})	0.119	$0.443 + 0.0107\,(\mathbf{weight} - 77) - 0.0159\,(\mathbf{LBM} - 59) + 0.0062\,(\mathbf{height} - 177)$	$(26.9 - 0.029 \times \mathbf{age} \times \mathbf{male} + 37.9 - 0.198 \times \mathbf{age} \times \mathbf{female})/v_1$
k_{12} (min^{-1})	0.114	$0.302 - 0.0056\,(\mathbf{age} - 53)$	0.114
k_{13} (min^{-1})	0.042	0.196	0.042
k_{21} (min^{-1})	0.055	$[1.29 - 0.024\,(\mathbf{age} - 53)]/[18.9 - 0.391\,(\mathbf{age} - 53)]$	0.055
k_{31} (min^{-1})	0.0033	0.0035	0.0033

Table A.5: Compartment model parameters by study. Covariates in **bold**. 'male'/'female' factors should be interpreted as indicators in $\{0, 1\}$. Age is given in years, height in cm, LBM = Lean Body Mass, calculated via the *James formula*. As a consequence, the model is invalid for patients with a high (> 37) Body Mass Index (BMI).

Data sources The data were not provided in a single merged source, and some partial pre-processing had been performed. Not all details of the provenance or preprocessing of the data were clear, but that which was ascertained is recorded here. The data were provided in a variety of text files, spreadsheets and MATLAB datafiles; we list each source below:

1. **Infusion data** From 40× Agilia pump data files. Cumulative volume and pump target and calculations given. Files have some (apparently) free text and sometimes different numbers of columns.
2. **BIS data** From 27× BIS data files. Recording started earlier than in other files, but start time of experiment usually given on the second sheet in cell $(3, 2)$. This start time is used for an offset in the extracted vector.
3. **Blood samples** Sparse blood samples taken, often both arterial and venous recorded in 40× spreadsheets.
4. **Patient covariates** Given in a single spreadsheet. The patient ID is coded under a different system to that used in the rest of the files.
5. **Vital Signs** MATLAB file with each patient vital signs saved as a cell of time series objects. Some pre-processing done, apparently this is primarily from a script of Martin Shaw (not verified).
6. **Administered drug records** A spreadsheet records the various drugs, including Propofol that were administered, and the time. In principle, alignment could be ensured by matching the Propofol protocol times with those in the Agilia files. However, it is not always obvious exactly how to match these since the spreadsheet is over a superset (in both directions).

These different data sources caused some issues when merging due to different file structures, different clocks, and different instruments.

Inconsistent data file structure The raw data files do not conform to a single standard layout. The working assumption is that different instruments were used for different patients, or that data were poorly tokenised by an initial ingestion. The infusion files are particularly problematic. Columns do not always contain a single variable type and data are sometimes interspersed by rows of free text (for instance warnings). Different files may contain different variables/columns. Each file was therefore manually inspected to ensure the correct information is collected; it is not easy to discover this programatically.

Alignment of BIS data We have had to assume the vital signs data and infusion data is aligned to the same clock. Presumably this was done during pre-processing, although it is not possible to verify this due to the absence of the raw blood pressure files. It has been discovered that patient 72 has a different set of columns for BIS, which was manually handled in the ingestion script. The original scripts obtained from Georgatzis et al. (2016) did not appear to align these correctly, although this perhaps is due to versioning problems. We avoid using the processed MATLAB data files and use the raw data in order to obtain the correct start time – this provides a superior alignment between BIS and the propofol infusion.

A.6.2.2 Data pre-processing

In this section we record any pre-processing which involves *changing* the data from the raw sources.

Infusion Dropout The infusion rate is calculated by differencing the cumulative amount infused. For 3 patients (patient 8, 34, 35), instrument dropout caused an under-reporting of Propofol infusion near the start of the sequence, and some unexpected values as the amount administered was recalculated. An example (patient 8) is shown in Figure A-15 (top left). To correct this, we assume that the cumulative amount after the corruption is correct (green section, middle right) and extrapolate the gradient of the cumulative infusion back to the origin (green section, bottom right). We then use this to replace the corrupted infusion rate in the original differenced form, and transform the offset at the origin into an initial bolus dose (green section, bottom left). The result appeared to be similar to sequences for patients with uncorrupted infusion sequences.

Unexpected changes in infusion Some patients exhibit unusual spikes or reductions in infusion rate after about 40 minutes. All of these features correspond to unusual effects in blood pressure. It has been presumed that these unexpected changes correspond to

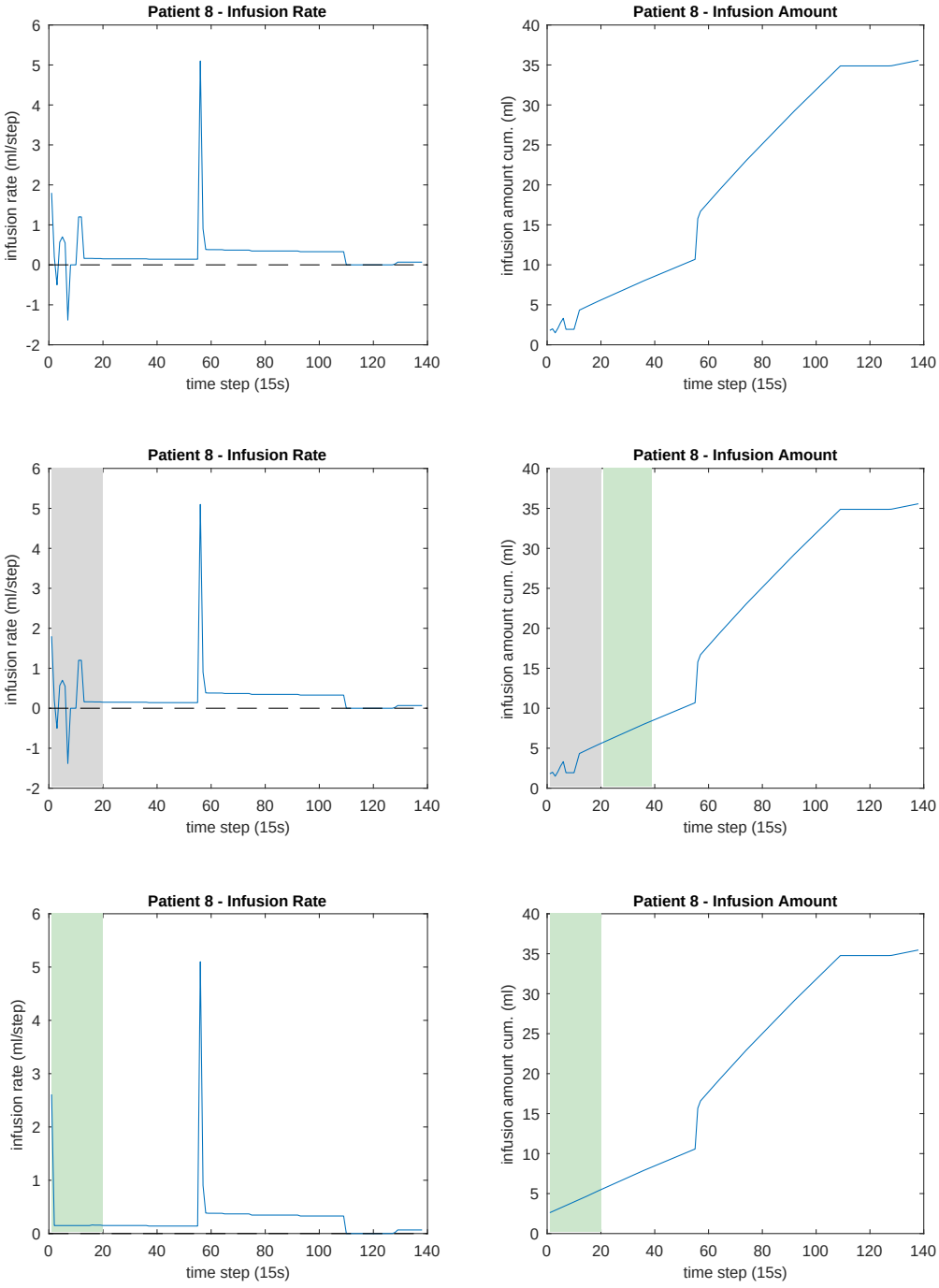


Figure A-15: Correcting the infusion data for patient 8 (see text). Left hand column is the infusion rate rime series, right hand column is the cumulative sum. (Top row) shows the raw time series. (Middle row) identifies the problematic region (red), and a 'clean' region in the cumulative graph in green. (Bottom row) applies the 'clean' trend identified above to the time series; the offset in the cumulative data is applied as a Bolus dosage at $t = 1$.

medically appropriate responses to the patient as they are approaching surgery, and hence all data at and subsequent to these events has been removed.

Elevated vitals near $t = 0$ Many patients had elevated vital signs at the beginning of their observations (some had systolic BP above 200). Our clinical collaborator suggested this can often be because a patient is anxious prior to the operation. Since we have no record of the steady state blood pressure for each patient, but the MTL fit showed strong evidence of downwards bias in the first two minutes, we discarded these initial datapoints during inference at test time.

A.6.3 Model

This section provides some additional details about our PD base model and MTDS approach in Section 6.2: in particular, derivation of the discrete time approximation (Section A.6.3.1); and model selection details (Section A.6.4.1). Section A.6.4 provides further information regarding information criteria and parameter counting for model selection.

A.6.3.1 Derivation of discrete-time relationship for piecewise constant $\mathbf{u}(t)$

Equation (2) can be solved by integration using Laplace transformations. Let $U(s)$ be the Laplace transform of $u(t)$, and $X(s)$ the Laplace transform of $x(t)$. Define further a piecewise $u(t) = \sum_{i=0}^{T-1} R(t-i)u_i$ with a slight abuse of notation for u, and for $R(\cdot)$ the the unit rectangle function. Then we have:

$$sX(s) = k_{1e}U(s) - k_{e0}X(s) \tag{A.52}$$

$$\Rightarrow X(s) = \frac{k_{1e}U(s)}{s + k_{e0}}. \tag{A.53}$$

Recognising the RHS as the product of two known Laplace transformations gives:

$$\mathcal{L}(x(t)) = \mathcal{L}(k_{1e}u(t)) \cdot \mathcal{L}(e^{-k_{e0}t}H(t)) \tag{A.54}$$

for the Heaviside step function $H(t)$. Then using the convolution theorem:

$$\Rightarrow x(t) = k_{1e} \int_0^t e^{-k_{e0}\tau} u(t-\tau)\,\mathrm{d}\tau \tag{A.55}$$

$$= k_{1e} \left[\sum_{i=0}^{t-1} u_i \int_0^t e^{-k_{e0}\tau} R(t-\tau-i)\,\mathrm{d}\tau \right] \tag{A.56}$$

$$= k_{1e} \left[\sum_{i=0}^{t-1} u_i \int_{t-i-1}^{t-i} e^{-k_{e0}\tau}\,\mathrm{d}\tau \right] \tag{A.57}$$

$$= k_{1e} \sum_{i=0}^{t-1} u_i \frac{1}{k_{e0}} \left[e^{-k_{e0}(t-i-1)} - e^{-k_{e0}(t-i)} \right] \tag{A.58}$$

$$= \sum_{i=0}^{t-1} \frac{k_{1e}}{k_{e0}} u_i \left(1 - e^{-k_{e0}}\right) e^{-k_{e0}(t-i-1)} \tag{A.59}$$

$$= \frac{k_{1e}}{k_{e0}} \left(1 - e^{-k_{e0}}\right) \sum_{i=0}^{t-1} u_i (e^{-k_{e0}})^{t-i-1}. \tag{A.60}$$

Equation (3) can be written explicitly as $x_{tj} = \beta_{1j} \sum_{i=0}^{t-1} \beta_{2j}^{t-i-1} u_i$ by unrolling the recursion. The relationships given in Section 3.1 are then derived by comparison with eq. (A.60).

A.6.4 Choosing the value of k

Since we do not know a priori the number of degrees of freedom needed to model the data, we perform a series of experiments to ascertain the optimal value of k. For each $k \in 1, \ldots, 20$, a 40-fold LOO experiment is performed where we calculate the RMSE of the optimal fit to the remaining patient. This section describes initial exploratory work that was undertaken prior to the development of the full MTDS. In the full Bayesian framework, the optimal value of k may have been obtained via use of shrinkage priors / automatic relevance determination (for a classic introduction, see MacKay, 1992).

We expect to see the performance decrease monotonically for increasing k until the subspace of parameter variation is found. Once this subspace is captured by span(Φ), the error is expected to remain constant. A number of model selection criteria are available to locate this optimal value of k, such as the Akaike Information Criteria (AIC) and Bayesian Information Criteria (BIC) (see e.g. Claeskens et al., 2008). However, these are not strictly valid for time series models. Sophisticated adaptations are required to deal with the time-structured nature of the observations and the non-identifiability of the model from the matrix factorisation, ΦZ.[13] We may nevertheless use them as a heuristic for penalising overly complex models.

In order to understand how well these criteria perform as heuristics, we generate synthetic data according to our model assumptions with a known value of $k = 3$. After performing

[13] Z is defined as the matrix where each column j corresponds to the latent code $\mathbf{z}^{(j)}$.

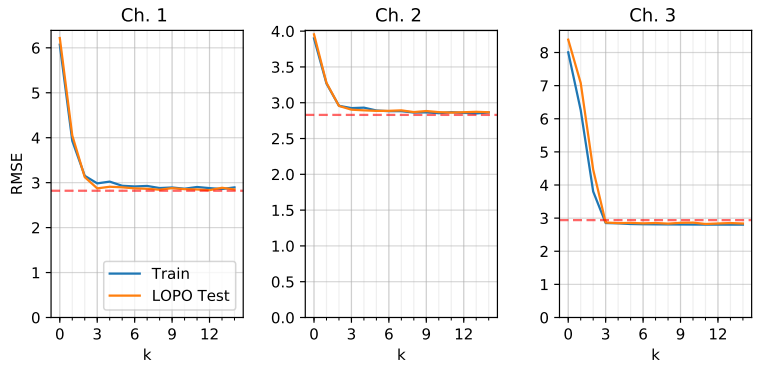


Figure A-16: Optimised RMSE with increasing k – synthetic data. Training and LOPO test set error are shown. The best objective achieved by individual models is shown by the dashed red line.

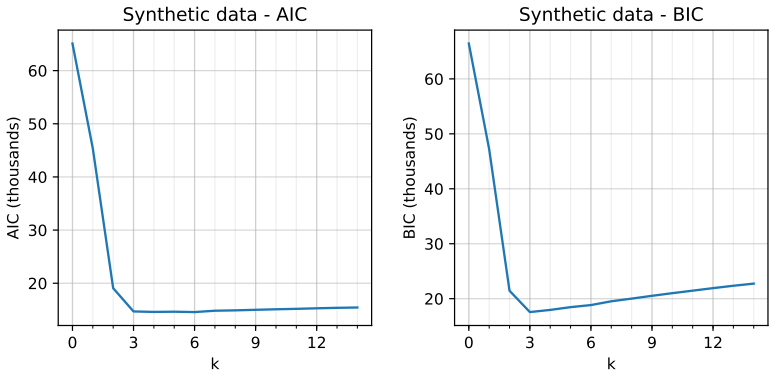


Figure A-17: Model selection information criteria for synthetic data (smaller is better).

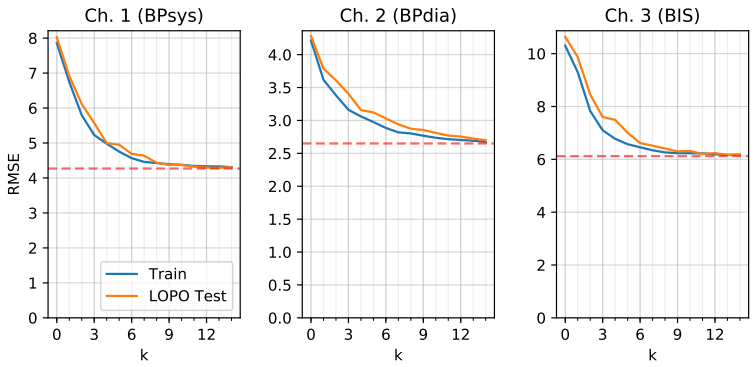


Figure A-18: Optimised RMSE with increasing k – patient data. Training and LOPO test set error are shown. The best objective achieved by individual models is shown by the dashed red line.

the 40-fold experiments, the RMSE over increasing values of k per channel is shown in Figure A-16. This follows the expected behaviour of monotonically decreasing error until the required subspace ($k = 3$) is found, and a constant error thereafter, corresponding to the irreducible noise in the data. The AIC and BIC criteria are shown in Figure A-17. Observe that in both cases, $k = 3$ appears to be a minimum (i.e. has the 'greatest evidence'), but higher values of k are not penalised as strongly for the AIC. In this case, the BIC correctly determines that $k = 3$ corresponds to the true model. Details of the AIC and BIC calculations are given in Section A.6.4.1.

We now perform the same experiment for the real data, as shown in Figures A-18, A-19. Figure A-18 demonstrates that the associated parameter subspace for the patient data is of higher dimension than for the synthetic data. It requires between 7 - 12 degrees of freedom to perfectly match the individual model performance, and perhaps subsequently, the test set performance suffers a little due to overfitting the subspace. Nevertheless, optimality is seldom required, and the gap between the global and individual models is substantially reduced after 3 dimensions. Using the same information criteria heuristics as before, we see in Figure A-19 that BIC chooses a latent dimension of $k = 5$ whereas AIC is more equivocal, with the minimum value giving evidence for a latent dimension of $k = 8$.

A.6.4.1 Details of the information criteria

We use the following definitions of the model selection criteria:

$$\text{AIC} = 2d - 2\ell_{n,d} \tag{A.61}$$

$$\text{BIC} = \log(n)d - 2\ell_{n,d} \tag{A.62}$$

where n is the number of observations, d is the number of parameters and $\ell_{n,d}$ is the log-likelihood. Since the factors Φ and Z are non-identifiable up to any invertible $d \times d$ matrix, we need to reduce the parameter count by these degrees of freedom.

We have seen in Section 6.2.1 that the number of parameters in an individual model is

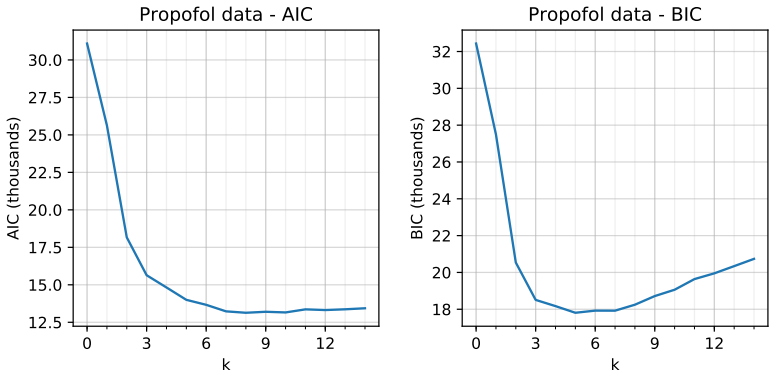


Figure A-19: Model selection information criteria for patient data (smaller is better).

$d = 36$. We will exclude the individual offsets fitted to each model since these are not shared, which reduces the number by 3, so we have:

$$d = \underbrace{33 \times (k+1)}_{\Phi,\mathbf{c}} + \underbrace{k \times N}_{Z} - \underbrace{k \times k}_{\text{over-param}} + \underbrace{\sum_{i=1}^{N} n_y^{(i)}}_{\boldsymbol{\alpha}} \tag{A.63a}$$

$$= (33 + N - k)k + 33 + \sum_{i=1}^{N} n_y^{(i)} \tag{A.63b}$$

where $n_y^{(i)}$ is the observation dimension of patient i. The contribution of each parameter in the PD model is shown underneath. It is not sensible to have $k > \min(N, 33)$ since the latter is the rank of the original matrix before factorisation. Thus (A.63b) shows us that the parameter count interpolates between the number of parameters in the global model ($k = 0, d = 33 + \sum_{i=1}^{N} n_y(i)$) and approximately the number of parameters in the individual models ($k = 33, d = 33 + 33N + \sum_{i=1}^{N} n_y^{(i)}$), although it cannot achieve equality if constant terms are included.

A.6.5 Experimental details

In this section we provide additional details regarding learning and inference.

A.6.5.1 Learning

The models were implemented in PyTorch (MTDS, STL, Pooled, TaskID) and Julia (LSTM); the gradients were calculated via backpropagation. Since we used the MAP approximation for the MTDS, all gradients could be calculated exactly, and we used the Adam optimiser in all cases. The gradients for the LSTM were not exact since we were optimising for 40-step prediction over all t, and the starting t was randomly sampled within each iteration. The grid search for the LSTM was performed using a nested cross-

validation procedure with a validation set of size 2 in each fold. Missing values for the observations in the log likelihood were handled by zeroing both the relevant observed and predicted values.

A.6.5.2 Inference of the MTDS and STL models

We use the Stan implementation of HMC for inference where required. This was especially important for the STL models in order to accurately capture the poorly conditioned and highly non-Gaussian posteriors. In all cases we used two chains of 3000 samples each. Each chain was initialised at a MAP value and the mass matrix was estimated during a burn-in period of 1500 samples. These settings usually resulted in over 100 effective samples (using split-$\hat{R}$, Gelman et al. 2013, §11.4). For some STL experiments, the estimate of the mass matrix essentially failed, resulting in very poor samples. In these cases, we overrode Stan's mass matrix estimate with a diagonal matrix; the per-dimension scales were calculated from the average mass matrices from experiments with sensible performance.

In practice the iid modelling assumptions are violated significantly. This was not especially problematic during training, but caused substantial problems for inference on short sequences at test time. On a number of occasions, un-modelled noise processes in the data resulted in implausible predictions with high confidence. In order to avoid this situation, one cannot simply increase the noise standard deviation; by the time it is high enough to ignore these noise processes, very little attention is paid to the underlying trend. A simple approach that appears to alleviate this problem is to downsample the observation data given to the *inference* algorithm by a factor of 4. This removes all significant partial autocorrelations seen in the residuals; predictions are still made on the full data.

A.6.6 Results

In this section we provide some additional examples of MTDS vs (adaptive) Pooled model predictions (Section A.6.6.1) and an example of the 'loading matrix' Φ of a learned MTDS-5 model (Section A.6.6.2.

A.6.6.1 Example prediction plots

Some example plots are provided in Figure A-20 to give the reader an intuition for the multi-task model. The examples were chosen to demonstrate the benefit of the MTDS over the (adaptive) Pooled model, and also the situations in which the model can perform worse. The first two examples (patient 12 and 19) demonstrate the model successfully adapting to patient dynamics for different channels. The example of patient 34 is one of a significant violation to modelling assumptions; no PD model is able to account for the uplift at 20 minutes. Patient 36 does not obviously follow the pattern of expected vital signs; the MTDS predicts an uplift at $T = 30$ as expected from a reduction in dose,

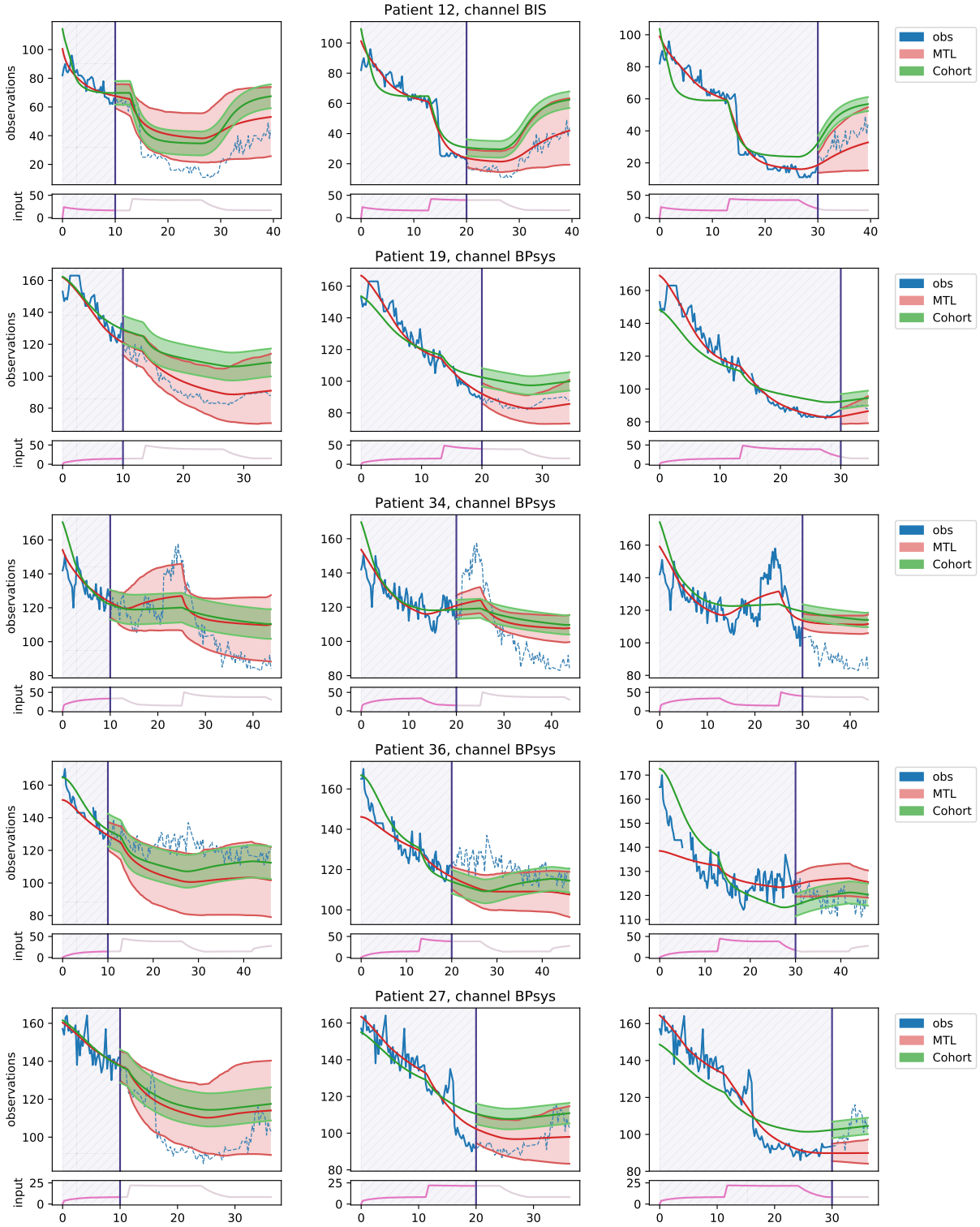


Figure A-20: Example predictions for various patients and channels. Each row is a given (patient, channel) combination; columns show 90% predictive intervals at increasing points in time ($T = 10, 20, 30$ minutes).

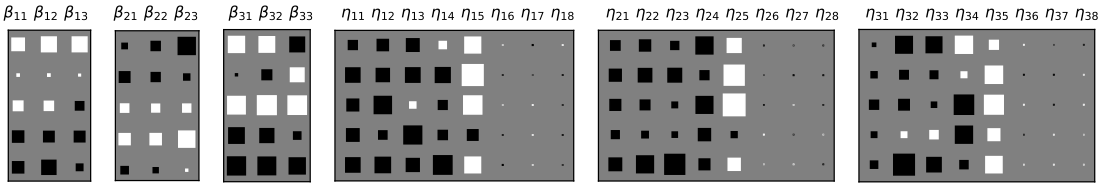


Figure A-21: Hinton plot of the $k \times d$ weight matrix Φ^{T} split by the corresponding parameter sets (positive values are shown in white). The $k = 5$ degrees of freedom corresponding to $\mathbf{z}$ are shown as rows. The relevant parameters are superimposed, but correspond in general to an elementwise nonlinear transformation of this affine space.

but the vitals instead reduce. Patient 27 shows an example where the MTDS is a little over-confident after 30 minutes; the Pooled model out-performs the MTDS at this stage by predicting a flatter curve.

A.6.6.2 Interpreting the latent code

An example of the weight matrix Φ within the multi-task network $\mathbf{h}_\phi$ is shown via Hinton plots in Figure A-21 for a trained $k = 5$ MTDS model. This was achieved using L1 regularisation to obtain a fairly sparse solution. The dynamics parameters $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ for channels 1 and 2 (BPsys and BPdia) appear to be highly correlated, as might be expected. z_3 is an important degree of freedom which allows BIS to have a different AR coefficient to the blood pressure channels. The effect of $\mathbf{z}$ on the emission function is more difficult to interpret: see Figure A-22 overleaf, which shows the effect of changing each dimension in isolation. Broadly speaking, increasing the value for any dimension of $\mathbf{z}$ results in a flatter and lower emission function, except for dimension 4 where the effect is relatively smaller.

A broad summary of the effect of each latent dimension is given in Table A.6. (We allow for some over-generalisation to permit a more parsimonious description – ideas from the disentangled literature, such as Makhzani et al. (2015); Higgins et al. (2017) should be used if a more easily interpretable model is desired). We interpret $\boldsymbol{\beta}_2$ as the sensitivity to propofol (magnitude of the latent x_t), and $\boldsymbol{\beta}_3$ as the 'level', since it determines the offset to $g_{\boldsymbol{\eta}}$ (a higher $\boldsymbol{\beta}_3$ results in lower observations). Unfortunately, the observed level is a function of $\boldsymbol{\beta}_3$, $\boldsymbol{\eta}$ and the inferred $\boldsymbol{\alpha}$, all of which affect the emissions in different ways, and a direct interpretation is difficult.

In general, we should not expect the same interpretations of the latent dimensions (up to permutation) even when trained on the same data. Different combinations of features might be equally valid. Nevertheless, it is gratifying that some interpretation can be placed upon the latent code.

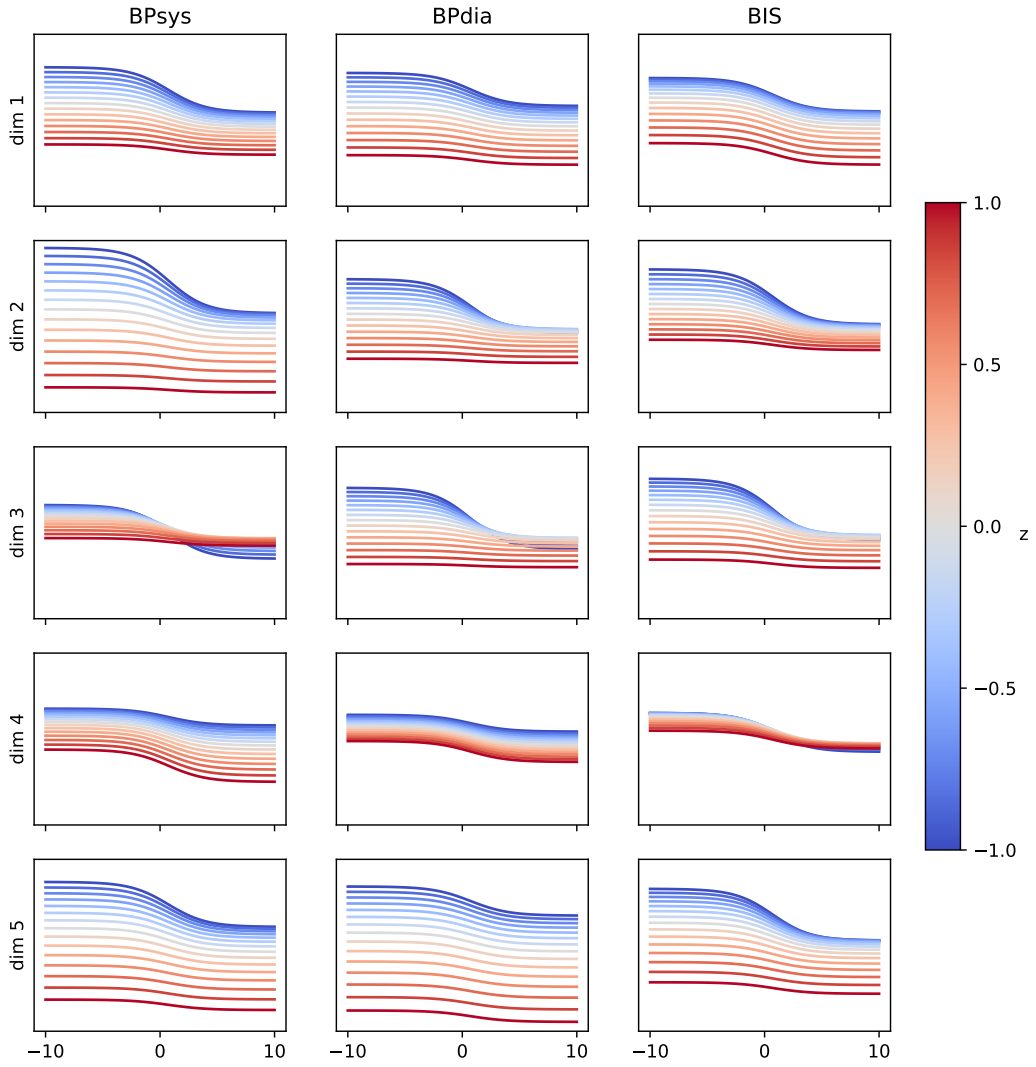


Figure A-22: Impact of each dimension of $\mathbf{z}$ on emission function with Φ given as in Figure A-21.

dim	lag/smooth	sensitivity to propofol	level	emission transform
1	↑all	↓BIS	↓BP	flatter/lower
2	-	-	-	flatter/lower
3	↑BP ↓BIS	↓all	↓all	flatter/lower
4	↓all	↑all	↑BP	-
5	↓all	-	↑all	flatter/lower

Table A.6: Summary (coarse-grained) of the effect of each dimension of $\mathbf{z}$ on the PD model (from an example MTDS).

A.7 Objectives for long term prediction

The various time series models discussed in this thesis (e.g. ARMA, LDS, RNN) are most often trained via maximum likelihood estimation (MLE), i.e. maximising the probability of the data, $p(\mathbf{y}_{1:T}; \boldsymbol{\theta})$. This often results in sensible short term predictions. However, in our empirical work (chapters 4 - 6) we have found that using MLE can produce poor long term predictions. Other authors, such as Bengio et al. (2015) and Chiappa et al. (2017) have made similar observations. In what follows, we will discuss the nature of the problem and some alternative objectives (Section A.7.1), present some examples of the problem (Section A.7.2), and a discussion (Section A.7.3) including related work.

A.7.1 Suboptimal prediction via MLE

A joint distribution can be decomposed into the product of 1-step ahead conditionals via the 'chain rule' of probability: $p(\mathbf{y}_{1:T}; \boldsymbol{\theta}) = p(\mathbf{y}_1; \boldsymbol{\theta}) p(\mathbf{y}_2 | \mathbf{y}_1; \boldsymbol{\theta}) \dots p(\mathbf{y}_T | \mathbf{y}_{1:T-1}; \boldsymbol{\theta})$. Any other causal (and hence valid) decomposition can be reduced to this form. To see this consider three cases:

- There exists a term $p(\mathbf{y}_{t+1} \,|\, \mathbf{y}_{1:t} \setminus \mathbf{y}_j)$ for some $j \leq t$. In this case we require another term $p(\mathbf{y}_j \,|\, \mathbf{y}_{1:t+1} \setminus \mathbf{y}_j)$ which is not causal.
- There exists a term $p(\mathbf{y}_{t+j} \,|\, \mathbf{y}_{1:t})$ for some $j > 1$. Similarly to the above, this will ultimately require non-causal terms, such as $p(\mathbf{y}_{t+1} \,|\, \mathbf{y}_{1:t}, \mathbf{y}_{t+j})$.
- There exists a term $p(\mathbf{y}_{t+1:t+j} \,|\, \mathbf{y}_{1:t})$ for some $j > 1$. But this itself may be reduced to one step terms, which satisfies the above via induction.

Hence for time series, MLE is equivalent to maximising all of the 1-step probabilities within the dataset. This fact is exploited by dynamical systems, which seek a parametric form of the one-step conditional $p(\mathbf{y}_{t+1} \,|\, \mathbf{y}_{1:t})$ for all t.

If one wishes to predict over a longer time horizon, the key quantity is the k-step predictive distribution $p(\mathbf{y}_{t+k} \,|\, \mathbf{y}_{1:t}; \boldsymbol{\theta})$, for $k > 1$. While this is not one of the terms in the one-step decomposition, it can be computed as:

$$p(\mathbf{y}_{t+k} \,|\, \mathbf{y}_{1:t}; \boldsymbol{\theta}) = \int p(\mathbf{y}_{t+k} \,|\, \mathbf{y}_{1:t+k-1}; \boldsymbol{\theta}) \dots p(\mathbf{y}_{t+1} \,|\, \mathbf{y}_{1:t}; \boldsymbol{\theta}) \, \mathrm{d}\mathbf{y}_{t+1:t+k-1}. \tag{A.64}$$

This uses the component one-step distributions from the joint distribution $p(\mathbf{y}_{1:T}; \boldsymbol{\theta})$ and hence it is often argued that optimal k-step predictions follow directly from learning the optimal joint distribution via MLE. In this section, we claim that this conclusion is well-founded only if the true distribution $p_{\text{true}}(\mathbf{y}_{1:T})$ is contained in the model class under consideration. If this is not the case, better predictions are often available through direct minimisation of the k-step error.

To elaborate upon this argument, it is helpful to maintain the distinction between the true distribution $p_{\text{true}}(\mathbf{y}_{1:T})$ and the parametric model, which for clarity we denote $q(\mathbf{y}_{1:T}; \boldsymbol{\theta})$.

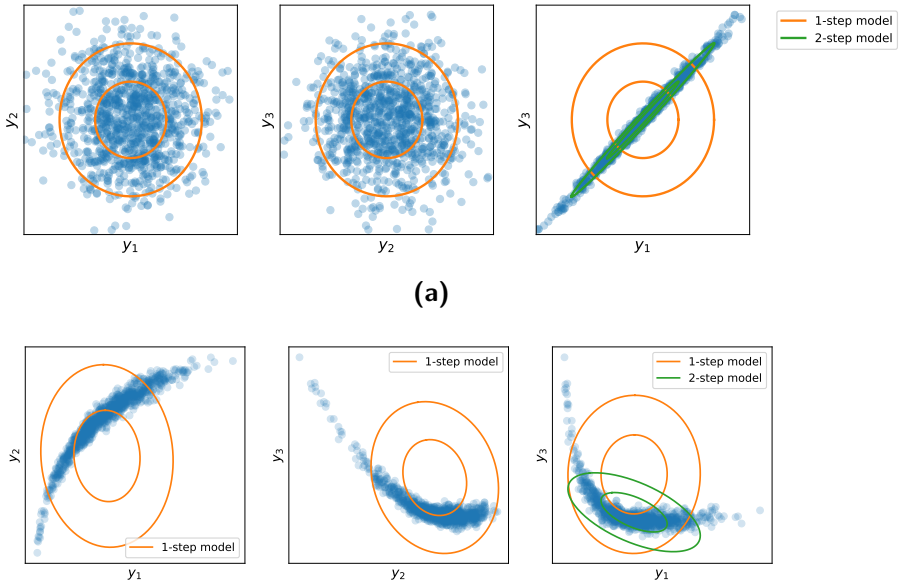


Figure A-23: Example distributions $p_{\text{true}}(y_{1:3}) \notin \mathcal{Q}$ where $\hat{\boldsymbol{\theta}}_{\text{MLE}}$ results in poor 2-step predictions. $\mathcal{Q}$ is the class of order-1 linear autoregressions. Each figure (a), (b) shows samples from p_{true} via the 3 bivariate marginal distributions (y_1, y_2), (y_2, y_3), (y_1, y_3) and the Gaussian models $q(y_t|y_{t-1}; \hat{\boldsymbol{\theta}}_{\text{MLE}})$, $q(y_3|y_1; \hat{\boldsymbol{\theta}}_{\text{2-step}})$ are visualised via their level curves (central $p = 0.90, 0.95$).

MLE can be formulated as the empirical (approximate) minimisation of:

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \arg\min_{\boldsymbol{\theta}} \text{KL}\left(p_{\text{true}}(\mathbf{y}_{1:T}) \,\middle\|\, q(\mathbf{y}_1; \boldsymbol{\theta}) \prod_{t=1}^{T-1} q(\mathbf{y}_{t+1} \,|\, \mathbf{y}_{1:t}; \boldsymbol{\theta})\right) \tag{A.65}$$

for a model class $\mathcal{Q} := \{q(\mathbf{y}_{1:T}; \boldsymbol{\theta}) \colon \boldsymbol{\theta} \in \Theta\}$ with parameters in Θ. Suppose we are really interested in optimal 2-step predictions $q(\mathbf{y}_{t+2} \,|\, \mathbf{y}_{1:t}; \boldsymbol{\theta})$. A natural way to express this objective is:

$$\hat{\boldsymbol{\theta}}_{\text{2-step}} = \arg\min_{\boldsymbol{\theta}} \sum_{t=1}^{T-2} \mathbb{E}_{p_{\text{true}}(\mathbf{y}_{1:t})}\left[\text{KL}\left(p_{\text{true}}(\mathbf{y}_{t+2} \,|\, \mathbf{y}_{1:t}) \,\middle\|\, q(\mathbf{y}_{t+2} \,|\, \mathbf{y}_{1:t}; \boldsymbol{\theta})\right)\right]. \tag{A.66}$$

Now in the case that $p_{\text{true}} \in \mathcal{Q}$, this second objective is minimised by setting $\hat{\boldsymbol{\theta}}_{\text{2-step}}$ such that $q(\mathbf{y}_{t+2} \,|\, \mathbf{y}_{1:t}; \boldsymbol{\theta}) = p_{\text{true}}(\mathbf{y}_{t+2} \,|\, \mathbf{y}_{1:t})$ for all t,[14] and hence via Equation (A.64), $\hat{\boldsymbol{\theta}}_{\text{MLE}}$ is a minimiser of Equation (A.66). However there is no reason to believe that $\hat{\boldsymbol{\theta}}_{\text{MLE}}$ is a minimiser of Equation (A.66) if $p_{\text{true}} \notin \mathcal{Q}$, due to the compounded errors of the suboptimal one-step components in Equation (A.64).

We provide two basic examples for $T = 3$ in Figure A-23 where MLE performs worse than a 2-step objective. In each row, the empirical distribution $p_{\text{true}}(y_{1:3})$ is visualised by its bivariate marginals, and $\mathcal{Q}$ is the class of AR(1) models defined by $q(y_{t+1} \,|\, y_{1:t}; \boldsymbol{\theta}) =$

[14] This follows since $\text{KL}(p \,\|\, q) \geq 0$ and $\text{KL}(p \,\|\, p) = 0$.

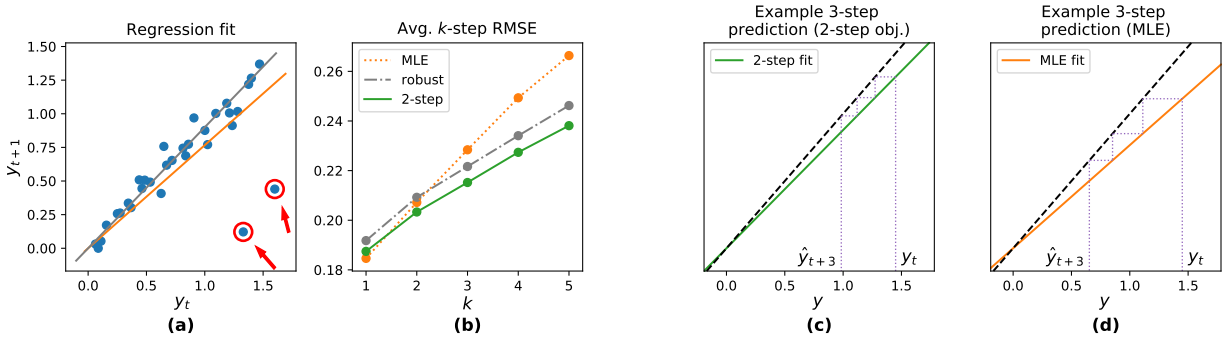


Figure A-24: (a) Linear autoregressive fit with and without outliers (highlighted in red). (b) RMSE for k-step predictions with autoregression fitted by MLE, robust MLE, 2-step objective (eq. A.66). (c), (d) Example 3-step cobweb plot (iterated prediction) from y_t using $\hat{\boldsymbol{\theta}}_{\text{2-step}}$ and $\hat{\boldsymbol{\theta}}_{\text{MLE}}$; black dashed line is the identity function.

$\mathcal{N}\left(ay_t, s^2\right)$. In both cases, clearly $p_{\text{true}} \notin \mathcal{Q}$. The marginal (Gaussian) distributions of the model fits are shown by level curves, and we see in both cases that the distribution $q(y_3 \,|\, y_1; \hat{\boldsymbol{\theta}}_{\text{MLE}})$ (orange), fit by MLE, substantially overestimates the variance, and models y_1 and y_3 as almost uncorrelated, resulting in highly suboptimal predictions compared to $\hat{\boldsymbol{\theta}}_{\text{2-step}}$.

A.7.2 Examples of time series with outliers

This section investigates the impact of MLE on two time series examples where the misspecification is limited to outliers.

AR model with outliers Our first example uses an order-1 AR model class for $\mathcal{Q}$, and the true distribution is given by:

$$p_{\text{true}}(x_t|x_{1:t-1}) = \mathcal{N}\left(0.9\, x_{t-1}, 0.1^2\right) \tag{A.67}$$

$$p_{\text{true}}(y_t|x_t) = \begin{cases} x_t & \text{with} \quad p = 0.95 \\ \mathcal{N}\left(0.4, 0.1^2\right) & \text{with} \quad p = 0.05. \end{cases} \tag{A.68}$$

Consecutive pairs (y_t, y_{t+1}) generated from this model are shown in Figure A-24(a). The basic autoregressive relationship (eq. A.67) is shown by the grey regression line, but the MLE fit, using $\hat{\boldsymbol{\theta}}_{\text{MLE}}$, is shown by the orange regression line, which takes into account the outliers. Figure A-24(b) compares the k-step performance of $\hat{\boldsymbol{\theta}}_{\text{MLE}}$ versus a robust fit which excludes the outliers, as well as the two-step objective (using $\hat{\boldsymbol{\theta}}_{\text{2-step}}$) of Equation (A.66). The 1-step error is smallest for the MLE fit, as expected, but the performance of the MLE fit quickly degrades for larger k; the 2-step fit performs the best for $k > 1$.

It is instructive to consider why the MLE performance degrades with k, via a closer look at the 3-step predictions, shown as a cobweb diagram in Figures A-24(c,d), which compare

the predictions when using $\hat{\boldsymbol{\theta}}_{\text{2-step}}$ and $\hat{\boldsymbol{\theta}}_{\text{MLE}}$. The estimate of the autoregressive coefficient a (shown by the coloured line) is much smaller for $\hat{\boldsymbol{\theta}}_{\text{MLE}}$ than for $\hat{\boldsymbol{\theta}}_{\text{2-step}}$. The former is the optimal trade-off for the outliers in the *first* step – but when performing a k-step prediction, this same trade-off is applied to *every intermediate step* of the prediction via Equation (A.64), shrinking the prediction towards zero, as in Figure A-24(d). A direct k-step objective such as Equation (A.66) is more robust to mis-specification, as it pays more attention to the long term impact of the trade-offs that are made.

ARMA model with outliers Our second example is an ARMA(3,2) model defined by:

$$p_{\text{true}}(x_t \,|\, x_{t-1}) = \mathcal{N}\left(0.8x_{t-1}, 0.15\right) \tag{A.69}$$

$$p_{\text{true}}(y_t \,|\, x_t) = \cos\left(\frac{2\pi t}{45}\right) + x_t. \tag{A.70}$$

visualised in Figure A-25 (top left), with the deterministic sinusoid shown in dashed black.[15] The bottom row shows the same sequence, but corrupted by outliers between $t = 105$ and 108. We will choose $\mathcal{Q}$ to be a LDS with $n_x = 3$ dimensional latent state, and hence $p_{\text{true}} \in \mathcal{Q}$, but the sequence with outliers in the bottom row is not. In both cases, we train the model on the first 150 datapoints, and test on the final 50. In the first case, where $p_{\text{true}} \in \mathcal{Q}$, MLE provides a sensible forecast (top middle). In the bottom row (bottom middle) where $p_{\text{true}} \notin \mathcal{Q}$, the MLE fit results in a poor long term prediction, due to a myopic focus on the 1-step fit.

We compare these results to the prediction of an LDS trained via a k-step objective, optimising all predictions in a $k = 50$-step ahead window according to the objective:

$$\hat{\boldsymbol{\theta}}^{\text{k-step}}_{\text{1-step}} = \arg\min_{\boldsymbol{\theta}} \sum_{t=1}^{T-k} \mathbb{E}_{p_{\text{true}}(\mathbf{y}_{1:t})} \text{KL}\left(p_{\text{true}}(\mathbf{y}_{t+1:t+k} \,|\, \mathbf{y}_{1:t}) \,\middle\|\, q(\mathbf{y}_{t+1:t+k} \,|\, \mathbf{y}_{1:t};\, \boldsymbol{\theta})\right). \tag{A.71}$$

The predictions from the optimised $\hat{\boldsymbol{\theta}}^{\text{50-step}}_{\text{1-step}}$ are shown in the final column in Figure A-25. The resulting model is able to perform close to optimal for the predictive window even when the model class is misspecified. The training set likelihood of the $\hat{\boldsymbol{\theta}}_{\text{MLE}}$ model is 25% higher *per observation* than that of the $\hat{\boldsymbol{\theta}}^{\text{50-step}}_{\text{1-step}}$ model, which underlines the problem of relying upon likelihood for long term predictive accuracy.

A.7.3 Discussion

There have been a number of recent papers that have observed the poor performance of MLE for long term prediction. Some of these have proposed objectives similar to Equation (A.71), including Bengio et al. (2015), Martinez et al. (2017) and Chiappa et al.

[15] Note that sinusoidal motion can be written as a deterministic AR(2) process, and the sum of an AR(1) and AR(2) model can be written as ARMA(3,2), see §4.7, p108, Hamilton, 1994.

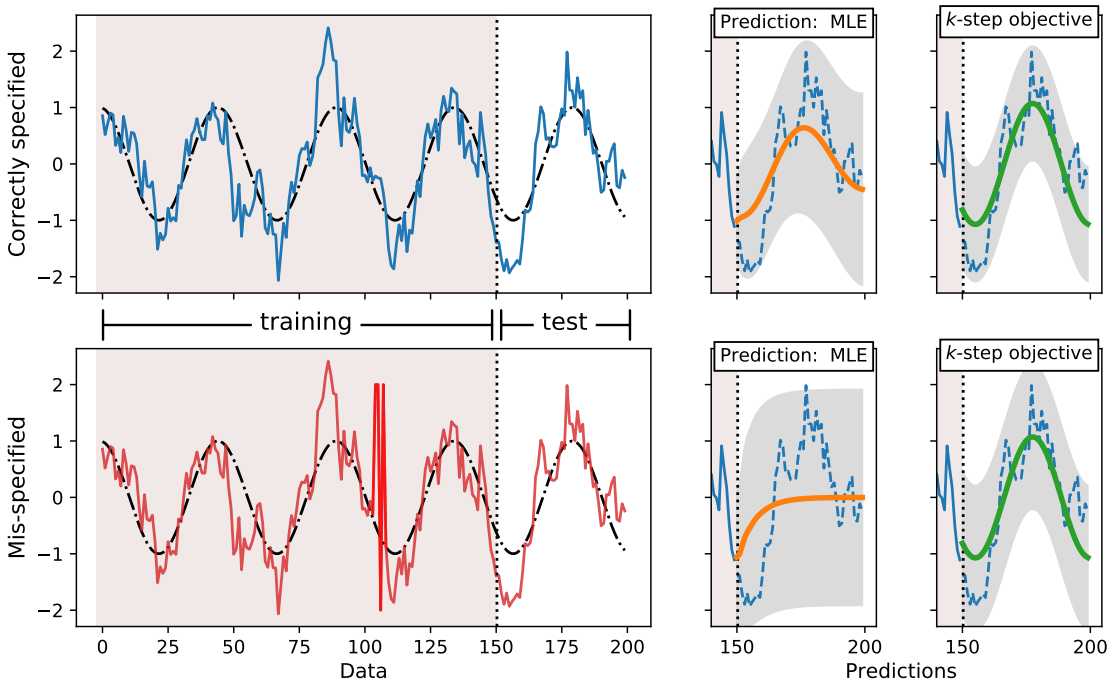


Figure A-25: Example ARMA(3,2) dataset (top), and with additional noise at $t = 105$ (bottom). The underlying sinusoidal motion is shown in black. The second column shows predictions on the test set with a stochastic state LDS, and the final column shows predictions for a deterministic state LDS.

(2017). Karl et al. (2017) pursue an alternative approach, changing the model and using regularisation to encourage better use of the dynamic transition. Lamb et al. (2016) address the problem adversarially by using a classifier to enforce similar distributions of hidden states for k-step predictions vs. 1-step predictions. Our main contribution here is to provide intuition for why MLE performs poorly for long term predictions.

In the context of generative models, Huszár (2015) suggests that MLE might perform poorly due to the mode-averaging nature of minimising $\mathrm{KL}\,(p \,\|\, q)$ wrt. q. In our context, it is more helpful to consider this as the spreading of probability mass over a diffuse area. This is typical under mis-specification, since the MLE objective places a large penalty on observations that are poorly modelled. In the previous example, this manifests in an inflated estimate of the noise variance, and a consequent poor fit of the underlying dynamic evolution.

However, Huszár argues against the proposed k-step objective of Equation (A.71), showing that it is inconsistent; i.e. in the case where $p_{\text{true}} \in \mathcal{Q}$, predictions from the learned model converge to $\prod_{j=1}^{k} p_{\text{true}}(\mathbf{y}_{t+j} \,|\, \mathbf{y}_{1:t})$ rather than the predictive joint $p_{\text{true}}(\mathbf{y}_{t+1:t+k} \,|\, \mathbf{y}_{1:t})$. While this may be a problem for generating plausible samples in generative models, the mean and marginal variance are often the only moments used for time series prediction, in which case we consider Equation (A.71) to be an appropriate objective. Furthermore, the product distribution $\prod_{j=1}^{k} p_{\text{true}}(\mathbf{y}_{t+j} \,|\, \mathbf{y}_{1:t})$ may be more reliably estimated from a sub-optimal model class $\mathcal{Q}$. The proposed solution of Huszár is to instead use a generalised

Jensen-Shannon divergence $\mathrm{JS}_\pi\left(p_{\text{true}}(\mathbf{y}_{1:T}) \,||\, q(\mathbf{y}_{1:T};\boldsymbol{\theta})\right)$ where

$$\mathrm{JS}_\pi\left(p \,||\, q\right) = \pi\,\mathrm{KL}\left(p \,\middle\|\, \pi p + (1-\pi)q\right) + (1-\pi)\,\mathrm{KL}\left(q \,\middle\|\, \pi p + (1-\pi)q\right). \tag{A.72}$$

However, it is not necessarily obvious that this performs better than a direct k-step prediction in the time series context, and no empirical results are given in its support.

A further cause of poor predictive performance may be the numerical optimisation of Equation (A.65). Optimisation may suffer from overfitting to small samples, or converging to poor optima. Even simple LDSs are known to have parameter estimation problems for practical sample sizes (e.g. $T \approx 100$, see e.g. Dent and Min; Auger-Méthé et al., 1978; 2016). System noise may be confused with emission noise with a dramatic impact upon predictive accuracy (e.g. Dennis et al., 2006; Auger-Méthé et al., 2016). Where the objective is intractable, standard approximations may not converge to an optimum (Turner and Sahani, 2011).

In the case of long term predictions, it may often be more practical to use a deterministic state model. We have seen that deterministic state models can often approximate stochastic state models (e.g. Section 2.2.3.6, Section 4.3.2). Integrating the system dynamics in stochastic nonlinear models is often impractical, due to the computational requirements of Monte Carlo sampling, which is further exacerbated by requiring the calculation of the k-step predictive distribution. Finally, a deterministic state appears to improve parameter identification substantially, since stochasticity cannot 'explain' deviations from the model under suboptimal parameter estimates. While the deterministic approach rules out a *diversity* of predictions, stochastic models will not necessarily provide useful diversity either (see e.g. Bayer and Osendorfer, 2014), by modelling correlations in the system dynamics with iid noise. Such diversity is of central interest to our work; the MTDS is able to model latent correlations via use of deterministic dynamical systems. See Chapters 3 - 6.