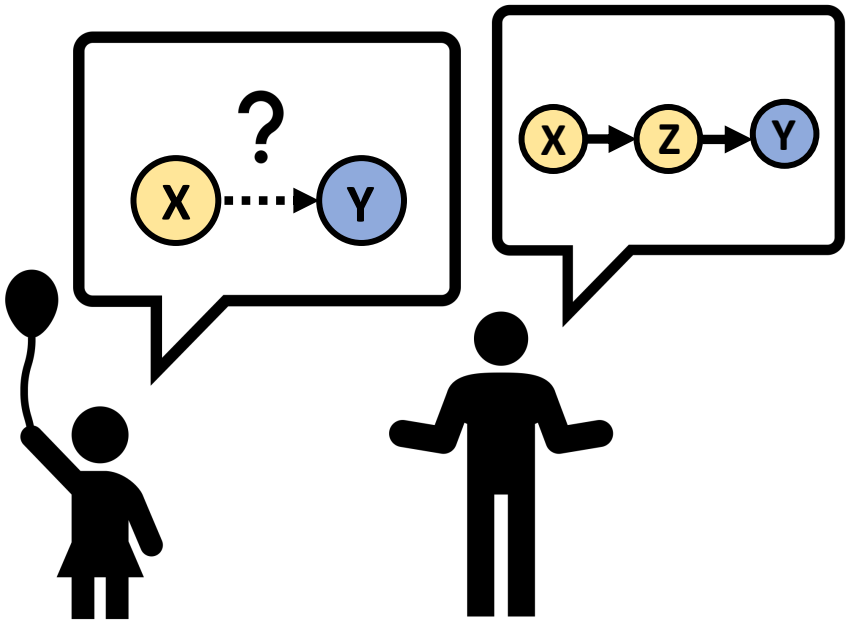
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



Learning using Privileged Time Series

Theory and applications of long-term prediction with Markov-Gaussian-linear dynamical systems

Master's thesis in Computer science and engineering

Rickard Karlsson & Martin Willbo

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021

Master's thesis 2021

Learning using Privileged Time Series

Theory and applications of long-term prediction with Markov-Gaussian-linear dynamical systems

Rickard Karlsson & Martin Willbo

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2021

Learning using Privileged Time Series
Theory and applications of long-term prediction with
Markov-Gaussian-linear dynamical systems
Rickard Karlsson & Martin Willbo

Supervisor: Fredrik D. Johansson, Department of Computer Science and Engineering
Examiner: Devdatt Dubhashi, Department of Computer Science and Engineering

Master's Thesis 2021
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A teacher providing additional privileged information (Z) to their pupil, which explains the answer (Y) to their question (X).

Typeset in LaTeX
Gothenburg, Sweden 2021

Learning using Privileged Time Series
Theory and applications of long-term prediction with
Markov-Gaussian-linear dynamical systems
Rickard Karlsson & Martin Willbo
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

In this thesis, we study the impact of having privileged information, in the form of intermediate time series, available during the training of models for long-term prediction. An algorithm which incorporates these time series is presented, and we prove that it is more efficient when the time series are drawn from a Gaussian-Linear dynamical system in comparison to a linear baseline model without access to the privileged information. Notably, the main analysis tool which we use is Rao-Blackwell's theorem. Using synthetically generated data, we validate the theoretical results and characterize the algorithm's behavior, but also test the limits of the algorithm by evaluating it on synthetic data where the assumptions of the theoretical analysis are violated. Furthermore, the applicability of the algorithm is investigated on real-world datasets for forecasting air quality in Chinese cities and predicting Alzheimer's disease progression. We show that our approach is preferable to classical learning in most settings, especially when data is scarce. Furthermore, empirical results are used to discuss the trade-off between bias and variance when using the proposed algorithm. We conclude that learning with privileged information in time series data for long-term prediction is beneficial and a highly interesting subject for further research. Proposals are given for future work, such as extensions to non-linear models or investigating how privileged time series could be beneficial for latent variable systems.

Keywords: Machine learning, linear dynamical systems, privileged information, time series, Alzheimer's disease, Rao-Blackwellization.

Acknowledgements

First and foremost, we wish to aim our biggest gratitude towards our supervisor Fredrik for his engagement and encouragement during our work, who has constantly exceeded our expectations for what the role of a supervisor needs to be. We also want to thank the rest of the Healthy AI lab group for your feedback during our presentations and for sharing your interesting research in the weekly meetings. Additionally, another thanks go to Zeshan, Rahul and David for their helpful discussions when working with us on turning our thesis into a scientific paper. We also want to acknowledge The Alzheimer's Neuroimaging Initiative (ADNI) for collecting and providing the data used in this project. Last but not least, we show our appreciation for our family and friends who have been supportive during our past years at Chalmers.

Rickard & Martin, Gothenburg, June 2021

Contents

List of Figures

List of Tables

1

Introduction

The amount of available data and computing resources has grown immensely over the last decade, which has lead to an increased interest and new advances in data-driven algorithms in many domains. There are, however, still many areas where sample sizes are low due to the collection of data being expensive or non-trivial. Healthcare and medicine are good examples of this. On top of small datasets, typical characteristics of healthcare data are high dimensionality coupled with other issues such as missing values that can make the construction and training of predictive models difficult [1]. Therefore, it is of great interest to use algorithms that are sample efficient, i.e learn from few examples but still have a high predictive capacity. In this thesis, we study a way to increase the sample efficiency of algorithms for long-term prediction tasks in particular.

A long-term prediction task adheres to the setting where we are interested in predicting an outcome happening in the future, and there are many applications where the ability to such predictions is relevant. Going back to the healthcare example; in medicine prediction, diagnostic, or prognostic tasks are usually addressed after some preliminary tests taken at the time of a patient's admission. The results of these tests make up a patient's so-called baseline data, and the accuracy of predictive models based on the baseline data can be crucial when it comes to assessing and controlling patients' mortality rates and overall well-being long term. But notably, for patients who are admitted to the hospital, or otherwise scheduled for examinations following a medical procedure, additional measurements are performed. Their progress is measured in medical records that contain information regarding their state at multiple points in time, so-called post-baseline data. This information is not known in advance for newly admitted patients. Nonetheless, finding ways to use the post-baseline data during the training of a predictive model could potentially lead to better sample efficiency.

Having access to post-baseline data during the training of a predictive model is a case of *learning using privileged information* (LuPI) [2, 3] or *side information* [4]. The post-baseline data is labeled as privileged information since it is not available during the prediction stage. In this thesis, we investigate the described setting in a formal manner. More specifically, we study if it is possible to use privileged information in time series to train a predictive model with better performance than

a model that does not have access to privileged information during training. This is done by theoretical and empirical studies. The main tool for the theoretical studies is the Rao-Blackwell theorem [5, 6].

Our main contributions are the following: (1) We prove theoretical guarantees that privileged information in time series is always desirable under certain conditions regarding the data distribution; (2) to the best of our knowledge, we showcase a novel approach of using so-called Rao-Blackwellization for statistical analysis of a predictive model; (3) we perform an extensive set of experiments on both synthetic and real-world datasets, namely prediction of Alzheimer's disease progression and air quality forecasting, where we show that including privileged information when training a model is beneficial in most cases, but we also provide insights when this approach fails; and (4) we give a proposal for interesting future work on this topic.

1.1 Context

Improving sample efficiency focuses on getting the most out of every sample during training. Multiple ways have been proposed to improve this aspect of machine learning algorithms. For example, data augmentation has become standard in image classification, where synthetic images are created by applying transforms that could rotate or distort the image without changing its inherent meaning and label [7].

An alternative way for improving sample efficiency is to include additional information in the training data that is relevant to a particular learning task. This idea constitutes the learning using privileged information (LuPI) framework [2, 3, 4]. An example from image classification which uses privileged information is a dataset containing images of birds that were accompanied by texts describing the attributes of the birds. Then, an algorithm which made use of these additional texts during training learned faster than the baseline algorithm with no access to the privileged information [8]. Another example is if knowledge is available about the similarity between different training samples, then this can be exploited during training to achieve a better sample complexity by matching such samples with each other [3, 9].

In this thesis we focus on long-term prediction. That is, given the past and present observations of a time series we perform predictions for outcomes into the future, where there is a clear chronological order between the input and outcome in the data. There are several challenges with long-term prediction such as rising uncertainties and accumulating errors as the prediction horizon increases. Two approaches of how to perform long-term prediction are direct and recursive approaches [10]. An example of a recursive approach is using a Recurrent Neural Network (RNN) for the prediction of Alzheimer's disease progression. Learning a model for disease progression from available data and using predicted values as input make predictions about points arbitrarily far out into the future possible [11]. A direct approach in contrast does not use predicted values during prediction. Instead one model per wanted prediction horizon need to be trained. It can be shown that this approach is preferable when using least-squares support-vector machines [10].

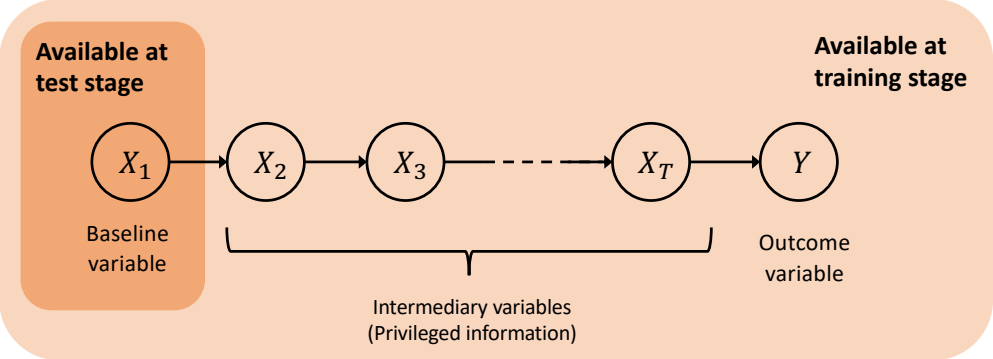


Figure 1.1: A visualization of the data in the setting with privileged time series.

Another difficulty with long-term prediction is that it limits the amount of data that can be used from a given data set for classical supervised learning. For example, consider a data set with points collected over ten years. If a seven year prediction horizon is wanted, only the first three years of the set can be matched with a target. The data between input and output can however be used as privileged information. If the intermediate data is used to create *soft targets*, with the objective of the learning algorithm to minimize a convex combination of the real and soft target losses, one can show that the sample efficiency can be improved in certain scenarios [12].

A special case of model-based reinforcement learning can be seen as a recursive approach to long-term prediction with privileged information. The special case being one available action and rewards being given only at a fixed future time point. The intermediate data points is then used to estimate the state-transition model. This approach is viable if the transitions can be simulated with sufficient fidelity [13, 14].

1.2 Problem

We formalize the problem of learning a model $f(X_1) \approx Y$ using the post-baseline data as an instance of *learning using privileged information* (LuPI) [2, 3], also known as *side information* [4]. Hence, we will refer to the post-baseline data as *privileged time series data* from now on.

At training time, we observe a discrete time series of length T with measurements written as $(X_1, \ldots, X_T)$ where X_t is measured at a time step $t = 1, \ldots, T$. Each sample can be seen as a trajectory of length T with a future outcome Y_{training}. At test time, we have a different set of samples containing only the initial time step X_1 and we wish to predict the corresponding outcome Y_{test}. In other words, we do not have access to any information into the future for $t > 1$ when the model is being evaluated at test time. This setting is visualized in Figure 1.1.

Given an algorithm that utilizes privileged information as described above the problem is to compare the algorithm's predictive accuracy and sample complexity com-

pared to a baseline model which observes the same number of training samples but do not have access to data for $t = 2, \ldots, T$ during training. More specifically, we attempt to answer the following questions in this thesis:

1. Can we use privileged time series for long-term predictions to achieve better sample efficiency when training a predictive model compared to a model which does not have access to the privileged information?

2. Specifically, under what conditions can we expect, and potentially guarantee, that the privileged information improves a model's predictive performance in terms of it's risk?

3. What is the predictive performance of the algorithm, which uses this privileged information, on both synthetic and real-world data sets for long-term prediction problems?

1.3 Limitations

The aim of this thesis is to compare the performance of algorithms that either have or do not have access to privileged information in addition to baseline data during training time. However, learning using privileged information is a general concept that encompasses many different tasks. Hence, we only consider the setting with a particular and explicit relationship between the input X_1, output Y and the privileged information $X_2, \ldots, X_T$. More specifically, when the data represents states at multiple points in time where each state at time t only depends on the previous state at time $t-1$. This is also known as the Markov property and will be described in the next chapter in more detail.

To study when privileged information is beneficial from a theoretical point of view, we will make further assumptions about the problem setting to simplify the analysis. For instance, we will assume linear transitions in the time series which enables the use of theory from linear statistical models that have been studied rigorously in the past. However, this means that the theoretical analysis does not necessarily apply to less restrictive settings. This limitation is difficult to circumvent, although it motivates why experiments on real-world data sets will be important to evaluate the validity of those assumptions.

Regarding the empirical study, there are two main lines of work. Firstly, as already mentioned, we will use real-world data sets to evaluate the models which we want to study. However, the goal is not to make comparisons to state-of-the-art applications since this thesis do not directly work on more practical applications as it requires a different type of effort. Instead, we emphasize that the experiments will aid with an analysis of the proposed algorithms. Secondly, we will use synthetic data to strengthen this analysis. In particular, such experiments can mainly be used to validate the results from the theoretical analysis since it possible to enforce any assumptions about the underlying data distribution.

1.4 Outline

In this chapter, we have introduced and motivated the problem that is studied in our thesis. Following in Chapter 2, we will define the problem formulation mathematically and provide the necessary background to understand our attempt to solve it. In Chapter 3, we present the main theoretical results from our work. Afterwards, Chapter 4 and 5 present the experimental setup and results, respectively, to support our findings from Chapter 3 as well as answer our research questions. Finally, Chapter 6 and 7 discuss our results, give suggestions for future work and provide a conclusion for our thesis.

1.5 Notation

- If X is a random variable then a lower letter indicates a value taken by that random variable. Furthermore, datasets are written with bold letters, i.e. $\mathbf{X}$, which can also be a random variable. It represents a matrix where samples are listed as row entries and each column corresponds to a feature.
- We denote n and d as the number of samples and input features, respectively.
- Subscript indices, such as X_t, usually denote time, unless otherwise stated.
- The transpose of a vector or matrix X is denoted with $X^\top$.
- $X \perp\!\!\!\perp Y|Z$ means that X is independent of Y given Z which is a set of random variables that may be empty.
- If we have some value we wish to estimate a parameter θ, then $\hat{\theta}$ denotes the estimator.
- $X \to Y$ means that Y is seen as a function of X, which can be non-deterministic.
- If X is a vector or a matrix, then $||X||_2$ indicates the L^2-norm of X.

2

Background

In this chapter we introduce the background for the thesis. It starts by stating the problem that we investigate formally. Then, the necessary theory is described in detail.

2.1 Problem formulation

Let us define the target variable $Y \in \mathbb{R}^k$ with k features, and the explanatory variable $X_1 \in \mathbb{R}^d$ having d features. Note that $k = 1$ unless stated otherwise. The task of interest is to learn a function $f(X_1) = \mathbb{E}[Y \mid X_1]$ where X_1 is an observation at time $t = 1$ and Y is a future outcome for some $t > 1$. We assume that there is a distribution D such that $(X_1, Y) \sim D(X_1, Y)$. In traditional supervised learning, we could select the function based on n samples of (X_1, Y) from the distribution that minimizes some loss $\mathcal{L}(\hat{f}(X_1), Y)$. But we are considering a scenario where there is additional data $(X_2, X_3, \ldots, X_T) \sim D(X_2, X_3, \ldots, X_T)$ for up until a final time step $t = T$. Without loss of generality, we assume that $X_t \in \mathbb{R}^d$ for $t = 1, \ldots, T$. However, the data are neither the input or output to the function f which we want to learn. In addition, it is not available during test time since the future is unknown when we make a prediction from time $t = 1$. This type of data is called privileged information and can contain valuable clues to improve the learning of f.

The goal is still to learn a function $f(X_1) = \mathbb{E}[Y \mid X_1]$ when we have access to privileged time series datasets $\forall t : \mathbf{X}_t = [X_t^{(1)}, \ldots, X_t^{(n)}]^\top \in \mathbb{R}^{n \times d}$ and $\mathbf{Y} = [Y^{(1)}, \ldots, Y^{(n)}]^\top \in \mathbb{R}^{n \times 1}$. These represent n trajectories of length T with a corresponding outcome Y, and we wish to find a function that minimizes the same loss $\mathcal{L}(\hat{f}(X_1), Y)$ as before. However, whether the privileged information is beneficial or not depends how it relates to the input, X_1, and the output, Y. This can vary greatly depending on the application. Consequently, we have to assume the structure of the privileged information a priori.

To define the underlying distribution D that generates the privileged time series data, we will define it as a Markov chain. This can be seen as a graph,

$$X_1 \to X_2 \to \cdots \to X_t \to \cdots \to X_T \to Y \tag{2.1}$$

where variables are represented as nodes and the arrows between pairs of nodes indicate conditional dependencies.

For a Markov chain, we have the Markov assumption that any state at time step t is only dependent on the state of the previous time step $t-1$, i.e.

$$p(X_t|X_{t-1}, X_{t-2}, \ldots, X_1) = p(X_t|X_{t-1}) \ .$$

Consequently, $D = p(X_1, X_2, \ldots, X_T, Y)$ can be factorized into

$$D = p(Y|X_T) \prod_{t=2}^{T} p(X_t|X_{t-1}) p(X_1) \ .$$

This modularity shows the intuition for how to sample from the distribution D. We would start by sampling X_1 according to $p(X_1)$ and, then, sample X_t iteratively based on the previous value X_{t-1} according to $p(X_t|X_{t-1})$. Afterwards, when we have sampled up until X_T, the outcome depends only on X_T according to $p(Y|X_T)$. In other words, if we know X_T that means that the target variable only depends on X_T. Lastly, note that this implies also that X_{t_c} is independent of future values where $t > t_c$.

Although the Markov chain describes the system well from a probabilistic point-of-view, we are also interested in the functional relationships between the variables $(X_1, \ldots, X_T, Y)$. Hence, it is helpful to complement the graph with a set of so-called structural equations [15]. This set can be defined in the following way:

$$\begin{cases} X_1 := f_1(U_1) \\ X_2 := f_2(X_1, U_2) \\ \vdots \\ X_t := f_t(X_{t-1}, U_t) \\ \vdots \\ X_{T-1} := f_{T-1}(X_{T-2}, U_{T-1}) \\ X_T := f_T(X_{T-1}, U_T) \\ Y := f_Y(X_T) + U_Y \end{cases} \tag{2.2}$$

where the set of functions $F = \{f_1, \ldots, f_T, f_Y\}$ belong to some functional class $\mathcal{F}$. Note that we have added explicit noise variables $U_1, \ldots, U_T$, and U_Y which are independent and have distributions $D_{U_1}, \ldots, D_{U_T}$ and D_{U_Y} respectively.

2.2 Learning using privileged information

Supervised learning in the classical machine learning paradigm is concerned with learning a model $f : \mathcal{X} \to \mathcal{Y}$ predicting an outcome $Y \in \mathcal{Y}$ given explanatory variables $X \in \mathcal{X}$, which are distributed according to a distribution $\mathcal{X} \times \mathcal{Y} \sim D$, where there is a target concept $c : \mathcal{X} \to \mathcal{Y}$, which is considered the true function.

The goal is to select a function $f \in \mathcal{F}$ that approximates c, which is done by minimizing the generalization error given some loss $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$,

$$R(f) = \mathbb{E}_{X \sim D} \left[\mathcal{L}(f(X), c(X)) \right] \ .$$

In many situations the distribution D is unknown which makes a direct minimization of the generalization error infeasible. However, we have often access to data which are assumed to be sampled from this distribution. Let $\mathbf{X} = [X^{(1)}, \dots X^{(n)}]^\top$ and $\mathbf{Y} = [Y^{(1)}, \dots Y^{(n)}]^\top$ be matrices that contain n samples from D, such that $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\mathbf{Y} \in \mathbb{R}^{n \times 1}$. Then, a quantity that is available to the learner in a supervised setting is the empirical error, or the empirical risk, which can be thought of as an approximation to the generalization error,

$$R_{\text{emp}}(f) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(X^{(i)}), Y^{(i)}) \ . \tag{2.3}$$

Hence, the task for a learning algorithm is to select a hypothesis f that minimizes the empirical error (2.3), formulated as

$$\hat{f} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \, R_{\text{emp}}(f) \ . \tag{2.4}$$

Learning with privileged information, also known as side information, involves having access to additional information during training that is relevant to the mapping one wish to learn [2, 3, 4]. This practice can be likened to a teacher-student setting where the student's goal is to master a problem with help from a teacher, in machine learning this is usually a function. For this purpose the student is given examples, but the student can ask the teacher for aid and the teacher can bestow the student with tools and help the student to better learn from the examples. The teacher can indicate which examples are similar to each other, or gives strong clues that lead to the correct answer. In machine learning this could be for the algorithm to better generalize to out of sample data. During a test, however, the student has to solve the problems without the additional clues from the teacher*. Hence the name *privileged information*, it is only available during the training stage.

Similar to the classical paradigm of supervised machine learning, learning with privileged information shares the goal of finding a hypothesis f that minimizes an objective function $\mathcal{L}$ given the dataset $(\mathbf{X}, \mathbf{Y})$. This problem is alternatively stated to (2.4) in (2.5) to ease the notation of learning with privileged information but the expressions are equivalent,

$$\underset{f \in \mathcal{F}}{\operatorname{argmin}} \ \mathcal{L}_f \left(f \mid (\mathbf{X}, \mathbf{Y}) \right) \ . \tag{2.5}$$

*The attentive reader might notice that this is what the front cover picture refers to.

Instead of the tuple $(\mathbf{X}, \mathbf{Y})$, the learning algorithm have access to a triple $(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$ available during training time where $\mathbf{Z}$ contains $Z \in \mathcal{Z}$—the privileged information—which is only available during training time. It optionally results in a second optimization problem (2.6). Note that here there is one sample of privileged information per sample of the explanatory variables X, but this does not necessarily have to be the case.

$$\underset{f \in \mathcal{F}, g \in \mathcal{G}}{\operatorname{argmin}} \; \mathcal{L}_Z \left(f, g \mid (\mathbf{X}, \mathbf{Z}, \mathbf{Y}) \right) \tag{2.6}$$

In the classical machine learning paradigm for a supervised learning setting the objective $\mathcal{L}_f$ is chosen as to suit the task at hand. For example, mean-squared error is a typical choice for the objective function for a regression task. This is no different for the exact form of the objective $\mathcal{L}_Z$ and auxiliary function g. These objects are partly informed by the privileged information Z and prior assumptions about how the data are related to X and Y.

For the problem presented in Section 2.1, the structure of the time series data $(X_1, \ldots, X_T, Y)$ is described with a Markov chain, see equation (2.1). In this case, a *direct pattern* prior is reasonable, as described by [4]. To simplify an example of how learning with privileged information can be used for the problem in Section 2.1, we denote the time series data with (X, Z, Y) where Z is the privileged information such that

$$X \to Z \to Y \ .$$

The direct pattern prior assumes that the privileged information Z represents intermediate results from the computations of the function $f : \mathcal{X} \to \mathcal{Y}$. In other words, one can see the function f as being a composite function of two functions $\phi : \mathcal{X} \to \mathcal{Z}$ and $\psi : \mathcal{Z} \to \mathcal{Y}$. That is, $Y = f(X) = \psi(\phi(X)) = \psi(Z)$. The second optimization problem (2.6) can then be expressed as $\mathcal{L}_Z = \mathcal{L}(\phi \mid (\mathbf{X}, \mathbf{Z}))$ where the loss $\mathcal{L}$ is suitably chosen. The primary objective (2.5) is then to minimize $\mathcal{L}(\psi \mid (\mathbf{Z}, \mathbf{Y}))$. where again the loss $\mathcal{L}$ is suitably chosen.

Example 1. To understand how privileged information can be helpful, we give an example to build intuition for it, inspired by [4]. Consider that we have the variables (X, Z, Y) such that Z is a function of X and Y is a function of Z. These relations are assumed to not be known in advance. Hence, we want to learn the mapping $f(X) = Y$ from some examples as seen in Table 2.1. Without knowing Z, we could have many hypotheses for what the function f is. However, if Z is known, then it might become easier to guess. In that case, we might suspect that $Z = X^2$ and $Y = Z + 3$ based on the examples. This would mean that $Y = X^2 + 3$, which happens to be correct. Had we only observed samples from X and Y, this relationship may not have been as easy to catch.

Table 2.1: Training data in Example 1 where Z is privileged information in a system $X \to Z \to Y$.

X	Z	Y
1	1	4
2	4	7
3	9	12
4	16	19

2.3 Linear regression

Regression is one of the main tasks in supervised machine learning. The goal is to predict the real-valued target $Y \in \mathbb{R}^1$ given a particular value of features $X \in \mathbb{R}^d$. This can be formulated as

$$Y = \mathbb{E}[Y \mid X = x] + \varepsilon \tag{2.7}$$

where we have some additive noise $\varepsilon \sim p(\varepsilon)$. Then, the task is to infer $\mathbb{E}[Y \mid X = x]$ for any x given these an observed dataset $(\mathbf{X}, \mathbf{Y})$.

Generally when considering equation (2.7), assumptions are made about the functional class of $E[Y|X = x]$ and the distribution $p(\varepsilon)$. This leads to the most common variant of regression, namely linear regression. For this case we make the following assumptions:

1. Linear sufficiency: It is sufficient to model Y from X with a linear relationship[†].
2. Zero-mean noise: $\mathbb{E}[\varepsilon] = 0$
3. Uncorrelated errors: $\mathrm{Cov}(\varepsilon^{(i)}, \varepsilon^{(j)}) = 0$ for any observations $i, j = 1, \ldots, n$ such that $i \neq j$.
4. Homoscedastic noise: The noise is independent of X, meaning that $\mathrm{Var}(\varepsilon|X) = \sigma^2 \mathrm{I}_n$ where I_n is the n-dimensional identity matrix.
5. No measurement error in X.

In this case, we say that $E[Y|X = x] = \theta^\top x$ where $\theta \in \mathbb{R}^d$ is the unknown model parameter. The problem can be re-formulated as

$$Y = \theta^\top X + \varepsilon \ .$$

Additionally, although the assumptions does not state this, it is common to assume that the noise is Gaussian, i.e. that we have $\varepsilon \sim \mathrm{N}(0, \sigma^2 \mathrm{I}_n)$ [‡].

[†]This is not the same as saying that X is linear. Instead. we say that Y is linearly dependent on some vector representation of X.

[‡]This is usually motivated by the fact that the combined sources of noise in nature are Gaussian due to the Central Limit Theorem.

Although the assumptions for linear regressions with Gaussian noise are relatively restrictive, there are several benefits with using linear statistical models. They have closed-form solutions that helps theoretical analyses. In addition, they can still be extended to other noise distributions, which is known as generalized linear models [16]. Lastly, they can also be applied to nonlinear problems with the so-called kernel trick that utilizes a kernel space so that the linear model learns a nonlinear relation in that space instead [16]. However, we will not study in-depth any of these extensions to non-linear models.

2.3.1 Ordinary least squares estimator

To infer the parameter θ given a dataset $(\mathbf{X}, \mathbf{Y})$, we want to find the maximum likelihood estimator (MLE) defined as

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, L(\theta; \mathbf{X}, \mathbf{Y})$$

where $L(\theta; \mathbf{X}, \mathbf{Y}) = p(\mathbf{X}, \mathbf{Y}|\theta)$ is the likelihood of the observed data given the unknown parameter. With Normal-distributed noise, the likelihood is

$$L(\theta; \mathbf{X}, \mathbf{Y}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{Y} - \mathbf{X}\theta)^\top \Sigma^{-1}(\mathbf{Y} - \mathbf{X}\theta)\right) .$$

To maximize this, it is convenient to instead consider minimizing the negative log-likelihood $-\log L(\theta; \mathbf{X}, \mathbf{Y})$. Note that this does not change the optimum due to the monotonicity of the logarithmic function. Writing out the log-likelihood and disregarding unnecessary constants which do not depend on θ, we get the following

$$\begin{aligned}
-\log L(\theta; \mathbf{X}, \mathbf{Y}) &\propto_\theta \frac{1}{2}(\mathbf{Y} - \mathbf{X}\theta)^\top \Sigma^{-1}(\mathbf{Y} - \mathbf{X}\theta) \\
&= \frac{1}{2}(\mathbf{Y} - \mathbf{X}\theta)^\top \frac{1}{\sigma}\mathrm{I}_n(\mathbf{Y} - \mathbf{X}\theta) \qquad \text{(assuming } \Sigma = \sigma\,\mathrm{I}_n\text{)} \\
&\propto_\theta \frac{1}{2}(\mathbf{Y} - \mathbf{X}\theta)^\top (Y - X\theta) \\
&= \frac{1}{2}||\mathbf{Y} - \mathbf{X}\theta||_2^2 .
\end{aligned}$$

Hence, we want to solve $\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, ||\mathbf{Y} - \mathbf{X}\theta||_2^2$, which can be done by setting the derivative to zero with respect to θ,

$$\frac{\delta}{\delta\theta}\left(\frac{1}{2}||\mathbf{Y} - \mathbf{X}\theta||_2^2\right) = \mathbf{X}^\top(\mathbf{Y} - \mathbf{X}\theta) = 0 .$$

This equation is also known as the normal equation. Since this is a strictly convex problem, there is a unique solution that returns the MLE estimator, also known as the Ordinary Least Squares (OLS) estimator,

$$\hat{\theta} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y} . \tag{2.8}$$

Fixed design setting

The target Y is a random variable with the Normal distribution $\mathrm{N}(X\theta, \sigma^2 \mathrm{I}_n)$ if we have assumed Gaussian noise. Meanwhile, it is common to assume that $\mathbf{X}$ is non-random and fixed which is known as the fixed design setting, where $\mathbf{X}$ is also known as the design matrix. This becomes important, among other things, when considering the statistical properties of the OLS estimator.

An important property of the OLS estimator is unbiasedness, which can be shown in the following way,

$$\begin{aligned}\mathbb{E}[\hat{\theta}] &= \mathbb{E}[(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}] \\ &= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\,\mathbb{E}[\mathbf{Y}] \\ &= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{X}\theta \\ &= \theta\ .\end{aligned}$$

We used that X is not random in the second equality and $E[\mathbf{Y}] = \mathbf{X}\theta$ in the third equality.

In addition, the OLS estimator has the variance

$$\begin{aligned}\mathrm{Var}(\hat{\theta}) &= \mathrm{Var}\left((\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}\right) \\ &= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\,\mathrm{Var}(Y)((\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top)^\top \\ &= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\sigma^2\,\mathrm{I}_n((\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top)^\top \\ &= \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}\end{aligned}$$

since $\mathrm{Var}(\mathbf{Y}) = \sigma^2\,\mathrm{I}_n$.

Random design setting

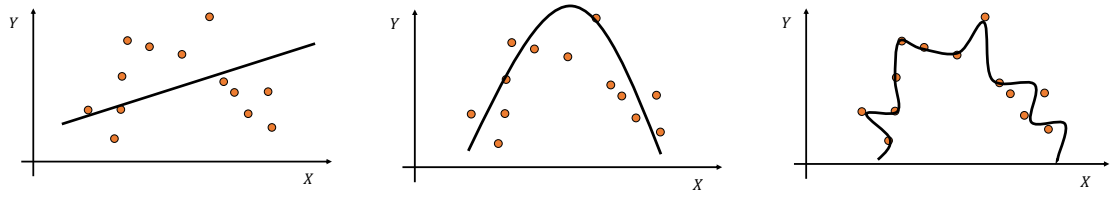
In certain settings we might not be able to assume that X is non-random. For example, let X be the age of a person and Y their height. If we want to find out how age and heights are related we have to randomly select a group from the general population and record both these attributes. Then, since we pick the people at random, the recorded age X in our data have been randomly selected as a consequence.

When X is random, which is known as the random design setting, the OLS estimator is still unbiased but its variance is different. In this setting, the variance of the OLS estimator is

$$\mathrm{Var}(\hat{\theta}) = \sigma^2\,\mathbb{E}[(\mathbf{X}^\top\mathbf{X})^{-1}] \quad \text{(random design setting)}\ .$$

We refer the reader to [17] for proofs of these statements.

An important difference between the fixed design and random design setting is that the former restricts an analysis of the prediction error of $\hat{\theta}^\top X$ to the randomness from the noise ε. This is known as the in-sample error, which is different from the generalized risk $\mathbb{E}_{X,Y}[\mathcal{L}(\hat{\theta}^\top X, Y)]$ where the randomness of X is also taken into account.



(a) Underfitted model **(b)** Correct model **(c)** Overfitted model

Figure 2.1: Example of where the middle figures shows the correct model $f(X)$ and the red dots are the training sample. The left-most model is too inflexible (underfitting) while the right-most model is too flexible (overfitting).

2.4 Bias-variance decomposition

In machine learning and statistics when investigating the generalization of a trained model $\hat{f}$ onto unseen data the sources of errors can be attributed to either the variance or the bias of the model. The former refers to error that comes from significant fluctuations due to overfitting to spurious noise, while the latter refers to systematic errors that could be due to an inflexible or misspecified model, also known as underfitting, see Figure 2.1. When fitting a model $\hat{f}$ given a data set from some distribution D with mean squared error (MSE) as the loss function, it is possible to decompose the prediction error for some, potentially unseen, point x_0 of the learned model into bias and variance error terms which make these sources of errors explicit.

Assuming that $Y = f(X) + \varepsilon_\varepsilon$ where $\mathbb{E}[\varepsilon] = 0$ and $\text{Var}(\varepsilon) = \sigma^2$. Additionally, if considering the MSE loss, we can derive an expression of the expected prediction error consisting of three terms for any input point x_0,

$$\begin{aligned}
\text{MSE}\left(\hat{f}(X = x_0)\right) &= \mathbb{E}_{Y|X=x_0}\left[\left(Y - \hat{f}\left(x_0\right)\right)^2\right] \\
&= \sigma_\varepsilon^2 + \left[\mathbb{E}_{Y|X=x_0}\left[\hat{f}\left(x_0\right)\right] - f\left(x_0\right)\right]^2 + \mathbb{E}_{Y|X=x_0}\left[\hat{f}\left(x_0\right) - \mathbb{E}_{Y|X=x_0}\left[\hat{f}\left(x_0\right)\right]\right]^2 \\
&= \sigma_\varepsilon^2 + \text{Bias}^2\left(\hat{f}(x_0)\right) + \text{Var}\left(\hat{f}(x_0)\right) \\
&= \text{ Irreducible Error } + \text{Bias}^2 + \text{ Variance } .
\end{aligned} \tag{2.9}$$

The first term is the random variation of the target Y around its true mean $\mathbb{E}[Y \mid X = x_0] = f(x_0)$. This is sometimes called the *Irreducible Error* since it is not possible to avoid this term no matter how well we estimate $f(X = x_0)$. The second term is the squared bias, this is a measure of how the average of the estimation differs from the true mean. The third and last term is the variance. This is the square of the expected deviation of an estimate $\hat{f}(X = x_0)$ from its mean [18].

2.5 Gaussian-linear dynamical system

A dynamical system is a time-dependent system consisting of a transition model g_t that describes how a state variable X_t evolves over time. In addition, if X_t is not observed directly, then there may exist an observation model h_t which describes another variable Z_t serving as an observation X_t. This is formulated as

$$\begin{aligned} X_t &= g_t(X_{t-1}, \varepsilon_t, V_t) \\ Z_t &= h_t(X_t, \delta_t, V_t) \end{aligned} \tag{2.10}$$

where ε_t and δ_t are noise variables. Meanwhile, V_t can be some external input into the system, which often appears in e.g. control theory where such models are commonly used. Lastly, note that the expression for X_t share many similarities with the set of structural equations in (2.2), because f_t can be seen as a transition model as well.

We will describe a simpler variant of the state-space model known as the Gaussian-linear dynamical system, as described by [16]. In this case, equation (2.10) can be re-formulated as

$$\begin{aligned} X_t &= A_t^\top X_{t-1} + B_t^\top V_t + \varepsilon_t \\ Z_t &= C_t^\top X_t + D_t^\top V_t + \delta_t \end{aligned}$$

where $\varepsilon_t \sim \mathrm{N}(0, Q_t)$ and $\delta_t \sim \mathrm{N}(0, R_t)$. The parameters A_t, B_t, C_t and D_t are linear mappings in the transition and observation models, while Q_t and R_t are the covariance matrices for the noise variables. These parameters can either be dependent or independent of time, which is called a non-stationary or stationary system respectively.

Learning the parameters $(A_t, B_t, C_t, D_t, Q_t, R_t)$ from observations $(Z_1, \ldots, Z_T)$, which is called system identification [19], has been studied extensively. One of the most common methods for this task is to use a Kalman filter [16, 19].

In certain settings some of the parameters might already be known which makes the inference problem easier. We will consider a case where $B_t = 0$, $C_t = I$, $D_t = 0$ and $R_t = 0$. This means that the state $X_t = Z_t$ is fully observable and that there is no external input signal[§]. Hence, A_t and Q_t are the only unknown parameters. However, we can set $Q_t = I$ without loss of generality since we can learn a modified A_t that captures information about the noise's covariance [16].

For a fully observable system with observations $(\mathbf{X}_1, \ldots, \mathbf{X}_T)$, the parameter A_t can be inferred by computing the maximum likelihood estimator

$$\hat{A}_t = \underset{A_t}{\operatorname{argmax}}\, L(A_t; \mathbf{X}_{t-1}, \mathbf{X}_t) \quad t = 2, \ldots, T$$

which equates to a linear regression problem. With Gaussian noise, $\hat{A}_t$ is given by the OLS estimator, see equation (2.8).

[§]It is not necessary to remove the external signal, but it makes the notation more convienent. Still, the special-case with external signals is briefly discussed in Chapter 3.

Meanwhile, if A_t is time-independent, we can compute

$$\hat{A} = \underset{A_t}{\operatorname{argmax}}\, L(A; \mathbf{X}_1, \ldots, \mathbf{X}_T)$$

which is the same as minimizing the sum of residuals

$$\hat{A} = \underset{A}{\operatorname{argmin}} \sum_{t=1}^{T} ||\mathbf{X}_t - \mathbf{X}_{t-1}A||_2^2$$

once again due to the assumption about Gaussian noise [16].

Stability of system

A system can be characterized as stable or unstable depending on how the system evolves as t grows very large. One way to define stability is by saying that if $\lim_{t\to\infty} ||X_t|| = \infty$ then the system is unstable. This is closely related to the properties of the transition parameter A and its eigenvalues as we will demonstrate.

If we ignore the system noise ε and write X_t as a function of X_1, we get that

$$X_t = (A^{t-1})^\top X_1 \ . \tag{2.11}$$

Assuming that $A \in R^{d\times d}$ is diagonalizable, A can be decomposed as $A = U\Lambda U^{-1}$ where U is a matrix with its eigenvectors as columns and Λ is a diagonal matrix with the eigenvalues as the entries along the diagonal, i.e. $\Lambda_{ii} = \lambda_i$ for $i = 1, \ldots, d$. We insert this decomposition into equation (2.11) and notice that

$$X_t = ((U\Lambda U^{-1})^{t-1})^\top X_1 = (U\Lambda^{t-1}U^{-1})^\top X_1 \ .$$

If $\lambda_i > 1$ for some i, then $||X_t||$ will become very large as t grows meaning that the system is unstable. Similarly, we can also see that if $\lambda_i < 1$ for some i, then $||X_t||$ will keep shrinking as t increases [16].

2.6 The Rao-Blackwell theorem

The Rao-Blackwell theorem, named after the mathematicians Calyampudi Radhakrishna Rao and David Blackwell, is a well-known theorem in statistics that despite its simplicity leads to surprisingly powerful results [5, 6].

We consider to have a parameter θ for the distribution of a dataset $\mathbf{X} \sim P(\mathbf{X}; \theta)$ that we wish to estimate with an estimator $\hat{\theta}$. Furthermore, we have another statistic $T(\mathbf{X})$ which is function of the data. If the statistic $T(\mathbf{X})$ is sufficient, meaning that $\hat{\theta} \perp\!\!\!\perp \mathbf{X}|T(\mathbf{X})$, then it is possible to find an estimator $\hat{\theta}_{RB}$ which is uniformly better than $\hat{\theta}$ with respect to any convex loss. Moreover, the estimator has the closed form expression $\hat{\theta}_{RB} = \mathbb{E}[\hat{\theta}|T(\mathbf{X})]$. Finding such an estimator with better loss is also known as Rao-Blackwellization.

The Rao-Blackwell theorem is proven with the help of Jensen's inequality which is stated in the following lemma, a proof can be found in [5].

Lemma 1 (Jensen's inequality)**.** *For any convex function $f(X)$ where X is a random variable with expectation $\mathbb{E}[X] = \mu$, then*

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$$

with equality only if X is a constant equal to μ, i.e. X is a degenerate random variable.

Then, [5] describes the following theorem known as the Rao-Blackwell theorem.

Theorem 2 (Rao-Blackwell)**.** *If we have an estimator $\hat{\theta}$ for a parameter θ and there is a sufficient statistic $T(\mathbf{X})$ for θ. Then, there exists an estimator $\hat{\theta}_{RB} = \mathbb{E}[\hat{\theta}|T(\mathbf{X})]$ such that*

$$\mathbb{E}[\mathcal{L}(\hat{\theta}_{RB}, \theta)] \leq \mathbb{E}[\mathcal{L}(\hat{\theta}, \theta)] \tag{2.12}$$

for any convex loss $\mathcal{L}$.

There will only be an equality if $\hat{\theta}_{RB}$ is constantly equal to $\hat{\theta}$.

Proof. Since $\mathcal{L}$ is a convex function we can use Lemma 1 and write

$$\mathcal{L}(\hat{\theta}_{RB}, \theta) = \mathcal{L}\left(\mathbb{E}[\hat{\theta}|T(\mathbf{X})], \theta\right) \leq \mathbb{E}\left[\mathcal{L}(\hat{\theta}, \theta)|T(\mathbf{X})\right]$$

where θ is constant and thus can be moved inside the expectation as well. Then, we take the expectation over $T(\mathbf{X})$ of both the right and left sides above. We get that

$$\mathbb{E}[\mathcal{L}(\hat{\theta}_{RB}, \theta)] \leq \mathbb{E}\left[\mathbb{E}\left[\mathcal{L}(\hat{\theta}, \theta)|T(\mathbf{X})\right]\right] = \mathbb{E}[\mathcal{L}(\hat{\theta}, \theta)]$$

where the second step comes from the fact that $E_X[X] = E_Y[E_X[X|Y]]$ for two random variables X and Y. Thus, we have shown that the inequality in equation (2.12) holds. Equality happens only if the distribution of $\hat{\theta}_{RB}$ is degenerate. ■

Rao-Blackwellization has been used in various contexts, for example to reduce variance in Markov Chain Monte Carl sampling procedurs [20], which can be applied to state inference in dynamical systems [21]. It has also been used to improve goal percentage estimates in basketball [22].

3

Learning using Privileged Time Series

This chapter describes the Learning using Privileged Time Series (LuPTS) prediction algorithm, which uses privileged information in the time series setting that has been presented. We begin with the assumptions that are made to construct the algorithm, then, we present a theoretical analysis of the algorithm.

3.1 Assumptions

In the previous chapter we formulated the underlying process for $(X_1, \ldots, X_T, Y) \sim p(X_1, \ldots, X_T, Y)$ as a Markov chain which defines the conditional probabilities between the variables. In addition, we stated in equation (2.2) that there is a functional relationship between them. We will now *explicitly state all the assumptions* that are made to construct an algorithm for learning the conditional expectation $\mathbb{E}[Y|X_1]$ with privileged time series.

Assumption 1 (Markov property)**.** *The conditional probability of state X_t given the states from previous time steps only depends on the most recent state X_{t-1}, i.e.*

$$P(X_t \mid X_{t-1}, X_{t-2}, \ldots, X_1) = P(X_t \mid X_{t-1}) \ .$$

Furthermore, the outcome Y only depends on the state at time T,

$$P(Y \mid X_T, X_{T-1}, \ldots, X_1) = P(Y \mid X_T) \ .$$

Assumption 2 (Linearity with additive isotropic Gaussian noise)**.** *The relationship between X_t and X_{t-1}, for $t = 2, \ldots, T$, is*

$$X_t = A_t^\top X_{t-1} + \varepsilon_t$$

where $\varepsilon_t \perp\!\!\!\perp X_{t-1}, X_t$ is a random variable independent of X_{t-1}. Similarly, we have

$$Y = \beta^\top X_T + \varepsilon_Y$$

where $\varepsilon_Y \perp\!\!\!\perp Y, X_T$. In addition, ε_t and ε_Y are distributed according to $N(0, \sigma_t^2 \, \mathrm{I}_d)$ and $N(0, \sigma_Y^2)$ respectively.

Note that these two assumptions corresponds to a Gaussian-linear dynamical system. Additionally, the isotropic Gaussian noise fulfills the necessary conditions for ordinary least squares regression, which are stationarity, mean-zero and homoscedasticity. Given these assumptions, we can state the following set of equations that describe the variables in the privileged time series setting. This is a re-formulation of equation (2.2).

$$\begin{cases} X_1 := U \\ X_2 := A_2^\top X_1 + \varepsilon_2 \\ \vdots \\ X_t := A_t^\top X_{t-1} + \varepsilon_t \\ \vdots \\ X_{T-1} := A_{T-1}^\top X_{T-2} + \varepsilon_{T-1} \\ X_T := A_T^\top X_{T-1} + \varepsilon_T \\ Y := X_T\beta + \varepsilon_Y \end{cases} \tag{3.1}$$

3.2 Algorithm

All the necessary groundwork regarding the distribution of the data $p(X_1, \ldots, X_T, Y)$ has been laid down in previous sections. We will now construct an algorithm, called the Learning using Privileged Time Series (LuPTS) algorithm, which uses the privileged time series data $(X_2, \ldots, X_T)$ during training time, to predict $\hat{f}(X_1) \approx Y$ at test time without access to the privileged information.

The task of the LuPTS algorithm is to infer the set of parameters $\Theta = \{A_2, \ldots, A_T, \beta\}$ that describes the linear relationships found in equation (3.1). Before explaining how to infer these parameters, we will first explain the algorithm step by step in less technical terms.

During the training stage, the algorithm is given a dataset $(\mathbf{X}_1, \ldots, \mathbf{X}_T, \mathbf{Y})$ and outputs an estimation $\hat{\Theta}$ of the parameters. The estimated parameters are selected by minimizing an empirical loss

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} \sum_{i=1}^{n} \mathcal{L}\left(f_\Theta | (\mathbf{X}_1, \ldots, \mathbf{X}_T, \mathbf{Y})\right)$$

where f_Θ is a linear function parameterized by Θ. Then, during the prediction stage, given new input X_1, the algorithm first predict $\hat{X}_2$. Then, given $\hat{X}_2$, it predicts $\hat{X}_3$ and so on. Finally, it is able to predict Y by nesting the functions. For the LuPTS algorithm, the prediction becomes

$$f_{\hat{\Theta}}(X_1) = (\hat{A}_2\hat{A}_3 \ldots \hat{A}_T\hat{\beta})^\top X_1 = \hat{Y} \ .$$

In our work, we will consider the mean squared error as the loss function, which is motivated by the assumption about Gaussian noise. The procedure in inferring $\hat{\Theta}$ will be explained further in the next section. The resulting Algorithm 1, however, is presented here.

Algorithm 1 Learning using Privileged Time series (LuPTS)

Require: Dataset $(\mathbf{X}_1, \ldots, \mathbf{X}_T, \mathbf{Y})$
1: **for** $t = 2, \ldots, T$ **do**
2: $\quad \hat{A}_t = (\mathbf{X}_{t-1}^\top \mathbf{X}_{t-1})^{-1} \mathbf{X}_{t-1}^\top \mathbf{X}_t$
3: **end for**
4: $\hat{\beta} = (\mathbf{X}_T^\top \mathbf{X}_T)^{-1} \mathbf{X}_T^\top \mathbf{Y}$
5: **return** $\{\hat{A}_1, \hat{A}_2, \ldots, \hat{A}_T, \hat{\beta}\}$

3.2.1 Inferring model parameters

The parameters are inferred by computing the MLE estimator for each parameter. As shown before, the MLE estimator can be computed by minimizing the negative log-likelihood. Due to the Markov property assumption, we can factorize the joint probability density which means that the negative log likelihood can be expressed as

$$\begin{aligned} -\log p(\mathbf{X}_1, \ldots, \mathbf{X}_T, \mathbf{Y}|\Theta) &= -\log \left(p(\mathbf{Y}|\mathbf{X}_T, \beta) \prod_{t=2}^{T} p(\mathbf{X}_t|\mathbf{X}_{t-1}, A_t) p(\mathbf{X}_1) \right) \\ &= -\log p(\mathbf{Y}|\mathbf{X}_T, \beta) - \sum_{t=2}^{T} \log p(\mathbf{X}_t|\mathbf{X}_{t-1}, A_t) - \log p(\mathbf{X}_1) \ . \end{aligned}$$

The first equality comes from the Markov property, while the second one comes from the properties of the logarithmic function. This means that the MLE estimator is given by computing

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} \ -\log p(\mathbf{Y}|\mathbf{X}_T, \beta) - \sum_{t=2}^{T} \log p(\mathbf{X}_t|\mathbf{X}_{t-1}, A_t) - \log p(\mathbf{X}_1) \ .$$

Note that the likelihood can be seen as several components and that $p(\mathbf{X}_1)$ can be ignored when maximizing this as it does not depend on any distribution parameter. The terms inside the sum correspond to the state transitions where the transition X_{t-1} to X_t only depends on the transition matrix A_t. Hence, we infer A_t by minimizing each term $-\log p(\mathbf{X}_t|\mathbf{X}_{t-1}, A_t)$ respectively. Similarly, $-\log p(\mathbf{Y}|\mathbf{X}_T, \beta)$ corresponds to the mapping from the last state X_T to the outcome Y, hence we select $\hat{\beta}$ that solely minimizes this term.

Due to the assumption about linearity and Gaussian noise, the MLE estimators are the same as the OLS estimator introduced in the previous chapter,

$$\begin{aligned} \hat{A}_t &= (\mathbf{X}_{t-1}^\top \mathbf{X}_{t-1})^{-1} \mathbf{X}_{t-1}^\top \mathbf{X}_t, \quad t = 2, \ldots, T \\ \hat{\beta} &= (\mathbf{X}_T^\top \mathbf{X}_T)^{-1} \mathbf{X}_T^\top \mathbf{Y} \ . \end{aligned}$$

Note the similarity to the *direct mapping* of learning with side information. The optimization problem for finding the state transitions matrices A_t could correspond to the side objective $\mathcal{L}_Z$ and the optimization problem for finding the linear mapping between X_T and Y would be the primary objective $\mathcal{L}_f$.

3.2.2 Introducing an additional assumption: Stationary state transitions

So far we have assumed that the transition parameters A_t can change over time. However, we will now consider a special-case for the LuPTS algorithm where we assume that A_t is independent of the time, meaning it is stationary.

Assumption 3 (Stationarity)**.** *For any time steps $t, \tilde{t} = 2, \ldots, T$ where $t \neq \tilde{t}$ we have that $p(X_t|X_{t-1}) = p(X_{\tilde{t}}|X_{\tilde{t}-1})$, implying that A_t is time-independent.*

There are two main motivations for why this assumption is of interest. First, the stationarity assumption is commonly considered in literature regarding time series or linear dynamical systems. As mentioned by [19], time-invariant systems are usually justified for practical applications due to their convenience and applicability. Hence, assuming stationarity is reasonable to consider in this line of work as well. Second, and more interestingly, if the assumption holds then the statistics for estimating the single transition matrix is expected to improve. In the non-stationary case, we have n samples to estimate each A_t for $t = 2, \ldots, T$, and the number of parameters we need to estimate grows with T. But if there is a constant A for each transition, we have $n \times (T-1)$ samples instead to estimate a single parameter. Note that these would not be independent samples, still the amount of available samples for A increases with the sequence length T which could be beneficial.

Referring back to Section 2.5, we showed that the transition matrix in a stationary Gaussian-linear dynamical model can be estimated by maximizing the likelihood from data. This lead the following equation:

$$\hat{A} = \underset{A}{\operatorname{argmin}} \sum_{t=2}^{T} ||\mathbf{X}_t - \mathbf{X}_{t-1}A||_2^2 \ .$$

Note that problem is convex, hence we can set the derivative with respect to A to zero to find the MLE estimator,

$$\begin{aligned} \sum_{t=1}^{T} 2\mathbf{X}_t^\top(\mathbf{X}_{t+1} - \mathbf{X}_t A) &= 0 \\ \Rightarrow \sum_{t=1}^{T} \mathbf{X}_t^\top \mathbf{X}_{t+1} &= \sum_{t=1}^{T} \mathbf{X}_t^\top \mathbf{X}_t A \ . \end{aligned}$$

Then, we compute the inverse of the left-hand side in the above equation and perform left-multiplication with it to both sides, which gives the following closed form expression for the MLE,

$$\hat{A} = \left(\sum_{t=1}^{T} \mathbf{X}_t^\top \mathbf{X}_t \right)^{-1} \sum_{t=1}^{T} \mathbf{X}_t^\top \mathbf{X}_{t+1} \ .$$

A stationary variant of the LuPTS algorithm is described by Algorithm 2. The

prediction made by this algorithm becomes

$$f_{\hat{\Theta}}(X_1) = (\hat{A}^{T-1}\hat{\beta})^\top X_1 = \hat{Y} \ .$$

Algorithm 2 Stationary variant of Learning using Privileged Time series (LuPTS)

Require: Dataset $(\mathbf{X}_1, \ldots, \mathbf{X}_T, \mathbf{Y})$

1: $\hat{A} = \left(\sum_{t=1}^{T} \mathbf{X}_t^\top \mathbf{X}_t\right)^{-1} \sum_{t=1}^{T} \mathbf{X}_t^\top \mathbf{X}_{t+1}$

2: $\hat{\beta} = (\mathbf{X}_T^\top \mathbf{X}_T)^{-1} \mathbf{X}_T^\top \mathbf{Y}$

3: **return** $\{\hat{A}, \hat{\beta}\}$

3.3 Statistical analysis

As the outline of the LuPTS algorithm has been explained, we will now study it theoretically, but before that, we will characterize the variance in the distribution $p(X_1, \ldots, Y)$. Then, we will show our main results that the LuPTS algorithm is guaranteed to have improved parameter recovery and prediction error on average than the baseline algorithm under the assumptions we have made. Lastly, we discuss various remarks on the theory and briefly touch upon how a more general and non-linear variant of the LuPTS algorithm could work.

3.3.1 Characterization of system variance

So far, we have considered regression from X_1 to Y. But one interesting question to also answer is whether it is better to predict Y based on a state X_t further into the future, i.e. $t > 1$ compared to from state X_1. Even if the LuPTS algorithm does not perform a direct prediction from X_t to Y, but instead it predicts Y based on an estimation $\hat{X}_t$, this gives us an indication of how helpful it is to make a prediction from a state which is "closer" to Y in time.

Given the assumptions outlined in Section 3.1, the LuPTS algorithm (and any OLS estimator) is unbiased. This means that any prediction error stems from its variance. Hence, predicting from a state further into the future is easier if the variance $\text{Var}(Y \mid X_t)$ is less than $\text{Var}(Y \mid X_{t-1})$.

We show this to be true by first outlining how Y can be described as a function of X_t, for any $t = 1, \ldots, T$, in Lemma 3. We get a characterization of the irreducible error and therefore an expression of the variance in Y given $X_t, t = 1, \ldots, T$. Finally we give an indication of how this variance depends on system stability.

Lemma 3. *Given the system described in Section 3.1, we have that,*

$$Y = (\prod_{k=t+1}^{T} A_k \beta)^\top X_t + \tilde{\varepsilon}_t, \ t = 1, \ldots, T$$

where

$$\tilde{\varepsilon}_t = \beta^\top C(t) + \varepsilon_Y$$

and

$$C(t) = \begin{cases} 0, & t = T. \\ \sum_{j=t+1}^{T} \left[\prod_{k=j+1}^{T} A_k \right]^\top \varepsilon_j, & 1 \leq t < T. \end{cases}, \quad t = 1, \ldots, T \ .$$

Note, $\tilde{\varepsilon}_t$ is a scalar which should not be confused with ε_t which is generally a vector.

We were interested in showing that $\mathrm{Var}(Y \mid X_t) \leq \mathrm{Var}(Y \mid X_{t-1})$. From Lemma 3, we have

$$\mathrm{Var}(Y \mid X_t) = \mathrm{Var}\left(\left(\prod_{k=t+1}^{T} A_k \beta \right)^\top X_t + \tilde{\varepsilon}_t \mid X_t \right) = \mathrm{Var}(\tilde{\varepsilon}_t)$$

where the second equality is because X_t is constant and independent of the noise. Furthermore, we can show that

$$\mathrm{Var}(\tilde{\varepsilon}_t) = \mathrm{Var}(\tilde{\varepsilon}_{t-1}) - \mathrm{Var}\left(\left(\prod_{k=t}^{T} A_k \beta \right)^\top \varepsilon_t \right) \leq \mathrm{Var}(\tilde{\varepsilon}_{t-1})) \ .$$

The inequality is possible since variances are always non-negative. Since $\mathrm{Var}(\tilde{\varepsilon}_{t-1}) = \mathrm{Var}(Y \mid X_{t-1})$, we have shown that

$$\mathrm{Var}(Y \mid X_t) \leq \mathrm{Var}(Y \mid X_{t-1}) \ .$$

Note that equality only holds if

$$\mathrm{Var}\left(\left(\prod_{k=t}^{T} A_k \beta \right)^\top \varepsilon_t \right) = 0$$

which happens if the noise ε_t has zero variance. If the variance is non-zero, we can conclude that it is strictly better to predict Y from X_t than from X_{t-1} for all $t = 2, \ldots, T$.

Remark. Note that these arguments also hold for the stationary case. The only difference being that $A_k = A_{k'}$ for $k, k' = 2 \ldots, T$.

Furthermore, given the characterization of stable and unstable systems presented in Section 2.5, one can reason about how these systems properties affect the variance of Y given X_1. In Corollary 4, this is outlined for the case of a stationary system. When the spectral radius of the transition matrix is larger than unity the spectral radius of the powers of A increases with the exponent, leading to larger variance in Y given X_1.

Corollary 4. *Given $C(t)$ in Lemma 3 and that we can decompose A into $U\Lambda U^{-1}$ one can for stationary systems write:*

$$C(t = 1) = (I + \sum_{j=1}^{T-2} A^j) = (I + \sum_{k=j}^{T-2} U\Lambda^j U^{-1}) \ .$$

3.3.2 Variance reduction using Rao-Blackwellization

In this section, we will consider the setting where the model specification is correct, meaning that we have data from a Markov Gaussian-linear dynamical system, as described in Section 3.1. Using arguments from the Rao-Blackwell theorem [5, 6], we will show that the non-stationary LuPTS estimator $\hat{\theta}_{\text{LuPTS}}$, which is the output from Algorithm 1, has improved statistical properties compared to the baseline estimator $\hat{\theta}_{\text{Baseline}} = (\mathbf{X}_1^\top \mathbf{X}_1)^{-1}\mathbf{X}_1^\top \mathbf{Y}$. To do this, we go step by step through the necessary lemmas which lead to our main results in Theorem 5.

We define the expected mean squared loss for the parameter recovery $\text{MSE}(\hat{\theta}) := \mathbb{E}_D[||\hat{\theta} - \theta||_2^2]$ and the expected risk $R(\hat{\theta}) = \mathbb{E}_{X_1,Y}[(\hat{\theta}^\top X_1 - Y)^2]$ which is the mean squared prediction error. Then, we state the following.

Theorem 5. *Let $D = (\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_T, \mathbf{Y})$ be a random dataset where $\hat{\theta}_{\text{Baseline}} := (\mathbf{X}_1^\top \mathbf{X}_1)^{-1}\mathbf{X}_1^\top \mathbf{Y}$ and $\hat{\theta}_{\text{LuPTS}} = \hat{A}_2 \dots \hat{A}_T \hat{\beta}$ is the output of Algorithm 1 (non-stationary). If the data is generated from the Markov-Gaussian-linear system described in Section 3.1, $\hat{\theta}_{\text{LuPTS}}$ is unbiased, and*

$$\text{MSE}(\hat{\theta}_{\text{LuPTS}}) = \text{MSE}(\hat{\theta}_{\text{Baseline}}) - \mathbb{E}_D[\text{Tr}(\text{Cov}(\hat{\theta}_{\text{Baseline}} \mid \hat{\theta}_{\text{LuPTS}}))] \ , \tag{3.2}$$

where the expectation is taken over random datasets D, since both estimators are functions of them. Further, it holds for the expected risk that over new, unseen samples (X_1, Y),

$$\mathbb{E}_D[R(\hat{\theta}_{\text{LuPTS}})] = \mathbb{E}_D[R(\hat{\theta}_{\text{Baseline}})] - \mathbb{E}_{D,X_1}[\text{Var}_{\hat{\theta}_{\text{Baseline}}}(\langle \hat{\theta}_{\text{Baseline}}, X_1 \rangle \mid \hat{\theta}_{\text{LuPTS}})] \ . \tag{3.3}$$

Since the variance and covariance are non-negative, $\hat{\theta}_{\text{LuPTS}}$ is at least as good as $\hat{\theta}_{\text{Baseline}}$ in both metrics. These statements holds for any input distribution $p(X_1)$.

The proof of Theorem 5 is similar to the proof of Theorem 2. In our case, we use the fact that the output of Algorithm 1 is a Rao-Blackwell estimator, which is stated as $\mathbb{E}[\hat{\theta}_{\text{Baseline}} \mid \hat{A}_2, \dots, \hat{A}_T, \hat{\beta}] = A_2 \dots \hat{A}_T \hat{\beta}$. However, this is not a trivial fact and we need to prove it first. We give a full proof to Theorem 5 later on in this section, after having showed the necessary building blocks in form of two lemmas.

Theorem 5 states that LuPTS is expected to better in comparison to baseline on average across datasets of the same size, in terms of parameter recovery and prediction error. Specifically, LuPTS will be better than baseline when there is uncertainty in $\hat{\theta}_{\text{Baseline}}$ given $\hat{\theta}_{\text{LuPTS}}$. In other words, if there are multiple datasets that lead to the same estimator $\hat{\theta}_{\text{LuPTS}}$ but different $\hat{\theta}_{\text{Baseline}}$, then we can expect $\hat{\theta}_{\text{LuPTS}}$ to perform better.

Note that due to both parameter estimates being unbiased, equation (3.4) states that the variance in the estimation by LuPTS will always be equal to or lower than the baseline estimation. This is clear from equation (2.9), since the difference in mean squared error comes from a difference in the variance when the bias term is zero as the irreducible error is the same for both algorithms.

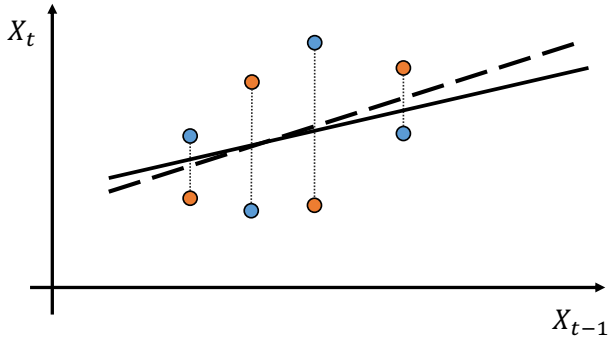


Figure 3.1: A display of the main idea behind the proof of Lemma 6. The wide dotted line is $\hat{A}_t^\top X_{t-1}$ and the solid line is $A_t^\top X_{t-1}$, the colored markers are different samples from two separate datasets. Note that the two different colors are mirrored along the line of best fit. Both datasets return the same OLS estimator $\hat{A}_t$, and we claim that they have the same probability to occur.

Building blocks for Theorem 5

As stated already, to prove Theorem 5, we need to prove two lemmas. The first one is a general statement for any Markov-Gaussian-linear dynamical system with isotropic noise.

Lemma 6. *Let $D = (\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_T, \mathbf{Y})$ be a random dataset generated from the Markov-Gaussian-linear system described in Section 3.1 and let $K = (\hat{A}_2, \ldots, \hat{A}_T, \hat{\beta})$ be the output from Algorithm 1 (non-stationarity). Then, for any $t = 2, \ldots, T$, we have that*

$$\mathbb{E}[\mathbf{X}_t | \mathbf{X}_{t-1}, K] = \mathbf{X}_{t-1}\hat{A}_t \quad \textit{and} \quad \mathbb{E}[\mathbf{Y} | \mathbf{X}_T, K] = \mathbf{X}_T\hat{\beta} \ .$$

The entire proof is found in Appendix A.2, but we present a proof sketch below.

Proof sketch. We will talk about the left statement in the lemma, but the same arguments applies to the right. Due to the isotropic noise, we can argue that, for any residual R_t, there exists two different datasets $\mathbf{X}_t = \mathbf{X}_{t-1}\hat{A}_t + R_t$ and $\tilde{\mathbf{X}}_t = \mathbf{X}_{t-1}\hat{A}_t - R_t$ which lead to the same OLS estimator $\hat{A}_t$. In addition, we show that these have the same probability of occuring. Hence, due to symmetry, the expected value lie exactly in between them, which is $\mathbf{X}_{t-1}\hat{A}_t$. Figure 3.1 shows what this looks like for the one-dimensional case where the differently colored samples belong to two separate datasets.

Lemma 6 says that the expected state at t, across datasets of the same size for which Algorithm 1 returns $\hat{A}_2\hat{A}_3 \ldots \hat{A}_T\hat{\beta}$, is equal to the estimated state at t given the previous state at $t-1$. It can be used to prove a second lemma stating that the output of LuPTS (Algorithm 1) is a sufficient statistic and, indeed, Rao-Blackwell estimator.

Lemma 7. *Let $D = (\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_T, \mathbf{Y})$ be a random dataset generated from the Markov-Gaussian-linear system described in Section 3.1, $(\hat{A}_2, \dots, \hat{A}_T, \hat{\beta})$ be the output from Algorithm 1 (non-stationarity), and $\hat{\theta}_{\text{Baseline}} \coloneqq (\mathbf{X}_1^\top \mathbf{X}_1)^{-1} \mathbf{X}_1^\top \mathbf{Y}$. It holds that,*

$$\mathbb{E}_D[\hat{\theta} \mid \hat{A}_2, \dots, \hat{A}_T, \hat{\beta}] = \hat{A}_2 \dots \hat{A}_T \hat{\beta} \ .$$

Proof. Let smaller letters $(\mathbf{x}_1, \dots, \mathbf{x}_T, \mathbf{y})$ indicate a value of the random dataset D and $B = (\mathbf{x}_1^\top \mathbf{x}_1)^{-1} \mathbf{x}_1^\top$, then we have,

$$\begin{aligned}
\mathbb{E}[\hat{\theta}|K] &= \int p(\mathbf{x}_1, \dots, \mathbf{x}_T, \mathbf{y}|K) \hat{\theta} d\mathbf{X}_1 \dots d\mathbf{X}_T d\mathbf{Y} \\
&\overset{(a)}{=} \int p(\mathbf{y}|\mathbf{x}_T, K) \prod_{t=2}^{T} p(\mathbf{x}_t|\mathbf{x}_{t-1}, K) p(\mathbf{x}_1|K) \hat{\theta} d\mathbf{X}_1 \dots d\mathbf{X}_T d\mathbf{Y} \\
&\overset{(b)}{=} \int p(\mathbf{y}|\mathbf{x}_T, K) \prod_{t=2}^{T} p(\mathbf{x}_t|\mathbf{x}_{t-1}, K) p(\mathbf{x}_1|K) \underbrace{B\mathbf{y}}_{=\hat{\theta}} d\mathbf{X}_1 \dots d\mathbf{X}_T d\mathbf{Y} \\
&\overset{(c)}{=} \int \prod_{t=2}^{T} p(\mathbf{x}_t|\mathbf{x}_{t-1}, K) p(\mathbf{x}_1|K) B \underbrace{\left[\int \mathbf{y} p(\mathbf{y}|\mathbf{x}_T, K) d\mathbf{Y} \right]}_{=\mathbb{E}[\mathbf{Y}|\mathbf{x}_T, K] = \mathbf{x}_T \hat{\beta}} d\mathbf{X}_1 \dots d\mathbf{X}_T \\
&= \int \prod_{t=2}^{T} p(\mathbf{x}_t|\mathbf{x}_{t-1}, K) p(\mathbf{x}_1|K) B \mathbf{x}_T \hat{\beta} d\mathbf{X}_1 \dots d\mathbf{X}_T \\
&\overset{(c)}{=} \int \prod_{t=2}^{T-1} p(\mathbf{x}_t|\mathbf{x}_{t-1}, K) p(\mathbf{x}_1|K) B \underbrace{\left[\int \mathbf{x}_T p(\mathbf{x}_T|\mathbf{x}_{T-1}, K) d\mathbf{X}_T \right]}_{=\mathbb{E}[\mathbf{X}_T|\mathbf{x}_{T-1}, K] = \mathbf{x}_{T-1} \hat{A}_{T-1}} \hat{\beta} d\mathbf{X}_1 \dots d\mathbf{X}_{T-1} \\
&= \int \prod_{t=2}^{T-1} p(\mathbf{x}_t|\mathbf{x}_{t-1}, K) p(\mathbf{x}_1|K) B \mathbf{x}_{T-1} \hat{A}_{T-1} \hat{\beta} d\mathbf{X}_1 \dots d\mathbf{X}_{T-1} \\
&\overset{(c)}{=} \int \prod_{t=2}^{T-2} p(\mathbf{x}_t|\mathbf{x}_{t-1}, K) p(\mathbf{x}_1|K) B \underbrace{\left[\int \mathbf{x}_{T-1} p(\mathbf{x}_{T-1}|\mathbf{x}_{T-2}, K) d\mathbf{X}_{T-1} \right]}_{=\mathbb{E}[\mathbf{X}_{T-1}|\mathbf{x}_{T-2}, K] = \mathbf{x}_{T-2} \hat{A}_{T-2}} \hat{A}_{T-1} \hat{\beta} d\mathbf{X}_1 \dots d\mathbf{X}_{T-2} \\
&= \dots = \quad \text{(recursively)} \\
&= \int p(\mathbf{x}_1|K) \underbrace{B\mathbf{x}_1}_{=I} \hat{A}_1 \dots \hat{A}_{T-1} \hat{\beta} d\mathbf{X}_1 \\
&\overset{(d)}{=} \hat{A}_1 \dots \hat{A}_{T-1} \hat{\beta} \int p(\mathbf{x}_1|K) d\mathbf{X}_1 = \hat{A}_1 \dots \hat{A}_{T-1} \hat{\beta}
\end{aligned}$$

where we have used the Markov assumption in (a), the definition of the OLS estimator $\hat{\theta}$ in (b) and then recursively utilize Lemma 6 in (c). Finally, in (d), we can move the estimators outside the integral as we have conditioned on them. ■

Proof of Theorem 5

Now, having showed Lemma 6 and 7, we are ready to present the proof of Theorem 5. As a reminder, the statement is the following.

Let $\mathrm{MSE}(\hat{\theta}) := \mathbb{E}_D[||\hat{\theta} - \theta||_2^2]$ be the expected mean squared loss for the parameter recovery, and $R(\hat{\theta}) = \mathbb{E}_{X_1,Y}[(\hat{\theta}^\top X_1 - Y)^2]$ the expected risk, which is the mean squared prediction error.

Theorem 5. *Let $D = (\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_T, \mathbf{Y})$ be a random dataset where $\hat{\theta}_{\text{Baseline}} := (\mathbf{X}_1^\top \mathbf{X}_1)^{-1}\mathbf{X}_1^\top \mathbf{Y}$ and $\hat{\theta}_{\text{LuPTS}} = \hat{A}_2 \dots \hat{A}_T \hat{\beta}$ is the output of Algorithm 1 (non-stationary). If the data is generated from the Markov-Gaussian-linear system described in Section 3.1, $\hat{\theta}_{\text{LuPTS}}$ is unbiased, and*

$$\mathrm{MSE}(\hat{\theta}_{\text{LuPTS}}) = \mathrm{MSE}(\hat{\theta}_{\text{Baseline}}) - \mathbb{E}_D[\mathrm{Tr}(\mathrm{Cov}(\hat{\theta}_{\text{Baseline}} \mid \hat{\theta}_{\text{LuPTS}}))] \ , \tag{3.4}$$

where the expectation is taken over random datasets D, since both estimators are functions of them. Further, it holds for the expected risk that over new, unseen samples (X_1, Y),

$$\mathbb{E}_D[R(\hat{\theta}_{\text{LuPTS}})] = \mathbb{E}_D[R(\hat{\theta}_{\text{Baseline}})] - \mathbb{E}_{D,X_1}[\mathrm{Var}_{\hat{\theta}_{\text{Baseline}}}(\langle \hat{\theta}_{\text{Baseline}}, X_1 \rangle \mid \hat{\theta}_{\text{LuPTS}})] \ . \tag{3.5}$$

Since the variance and covariance are non-negative, $\hat{\theta}_{\text{LuPTS}}$ is at least as good as $\hat{\theta}_{\text{Baseline}}$ in both metrics. These statements holds for any input distribution $p(X_1)$.

Proof. Unbiasedness of $\hat{\theta}_{\text{LuPTS}}$ follows from Lemma 7 since $\hat{\theta}_{\text{Baseline}}$ is also unbiased. The remaining result follows from Lemma 7 and standard Rao-Blackwell arguments [5, 6],

$$\begin{aligned}
\mathrm{MSE}(\hat{\theta}_{\text{LuPTS}}) &= \mathbb{E}_D[\|\hat{\theta}_{\text{LuPTS}} - \theta\|^2] \\
&= \mathbb{E}_D[\| \mathbb{E}[\hat{\theta}_{\text{Baseline}} \mid \hat{\theta}_{\text{LuPTS}}] - \theta\|^2] \\
&= \mathbb{E}_D[\| \mathbb{E}[\hat{\theta}_{\text{Baseline}} - \theta \mid \hat{\theta}_{\text{LuPTS}}]\|^2] \\
&= \mathbb{E}_D\left[\sum_{j=1}^{d} (\mathbb{E}[\hat{\theta}_{j,\text{Baseline}} - \theta_j \mid \hat{\theta}_{\text{LuPTS}}])^2\right] \\
&= \mathbb{E}_D\left[\sum_{j=1}^{d} \left(\mathbb{E}[(\hat{\theta}_{j,\text{Baseline}} - \theta_j)^2 \mid \hat{\theta}_{\text{LuPTS}}] - \mathrm{Var}\left[\hat{\theta}_{j,\text{Baseline}} \mid \hat{\theta}_{\text{LuPTS}}\right]\right)\right] \\
&= \mathbb{E}_D[\mathbb{E}[\|\hat{\theta}_{\text{Baseline}} - \theta\|^2 \mid \hat{\theta}_{\text{LuPTS}}]] - \mathbb{E}_D\left[\sum_{j=1}^{d} \mathrm{Var}\left[\hat{\theta}_{j,\text{Baseline}} \mid \hat{\theta}_{\text{LuPTS}}\right]\right] \\
&= \mathrm{MSE}(\hat{\theta}_{\text{Baseline}}) - \mathbb{E}_D\left[\left(\mathrm{Cov}\left[\hat{\theta}_{\text{Baseline}} \mid \hat{\theta}_{\text{LuPTS}}\right]\right)\right] \ .
\end{aligned}$$

Recall that X_1 represents a new test point, independent of the dataset D. For the second result, we note that for any estimator $\hat{\theta}$,

$$\mathbb{E}_D[R(\hat{\theta})] = \mathbb{E}_D[\mathbb{E}_{X_1,Y}[(\hat{\theta}^\top X_1 - Y)^2]] = \mathbb{E}_{X_1}[\mathbb{E}_D[\mathbb{E}_{Y|X_1}(\hat{\theta}^\top X_1 - Y)^2 \mid X_1]] \ .$$

Then, if $\hat{\theta}$ is unbiased, for the Gaussian linear model $Y = \theta^\top X_1 + \epsilon$ with $\epsilon \sim \mathrm{N}(0, \sigma^2)$,

$$\mathbb{E}_D[\mathbb{E}_{Y|X_1}[(\hat{\theta}^\top x_1 - Y)^2 \mid X_1 = x_1]] = \underbrace{\mathbb{E}_D[(\hat{\theta}^\top x_1 - \mathbb{E}_D[\hat{\theta}]^\top x_1)^2]}_{=\text{variance}} + \underbrace{(\mathbb{E}_D[\hat{\theta}]^\top x_1 - \theta^\top x_1)^2}_{=\text{bias}^2=0} + \sigma^2 \ .$$

The variance term can then be rewritten as, since $\mathbb{E}_D[\hat{\theta}] = \theta$,

$$\mathbb{E}_D[(\hat{\theta}^\top x_1 - \mathbb{E}_D[\hat{\theta}]^\top x_1)^2] = \mathbb{E}_D[\langle \hat{\theta} - \theta, x_1 \rangle^2] \ .$$

Then, since $\hat{\theta}_{\text{LuPTS}}$ is an unbiased estimator of θ_{Baseline},

$$\begin{aligned}
\mathbb{E}_D[R(\hat{\theta}_{\text{LuPTS}})] &= \mathbb{E}_{X_1}[\mathbb{E}_D[\langle \hat{\theta}_{\text{LuPTS}} - \theta, X_1 \rangle^2]] + \sigma^2 \\
&= \mathbb{E}_{X_1}[\mathbb{E}_D[\langle \mathbb{E}[\hat{\theta}_{\text{Baseline}} \mid \hat{\theta}_{\text{LuPTS}}] - \theta, X_1 \rangle^2]] + \sigma^2 \\
&= \mathbb{E}_{X_1}[\mathbb{E}_D[\mathbb{E}[\langle \hat{\theta}_{\text{Baseline}} - \theta, X_1 \rangle \mid \hat{\theta}_{\text{LuPTS}}]^2]] + \sigma^2 \\
&= \mathbb{E}_{X_1}[\mathbb{E}_D[\mathbb{E}[\langle \hat{\theta}_{\text{Baseline}} - \theta, X_1 \rangle^2 \mid \hat{\theta}_{\text{LuPTS}}] \\
&\qquad - \mathrm{Var}(\langle \hat{\theta}_{\text{Baseline}} - \theta, X_1 \rangle \mid \hat{\theta}_{\text{LuPTS}})]] + \sigma^2 \\
&= \mathbb{E}_D[R(\hat{\theta}_{\text{Baseline}})] - \mathbb{E}_{D,X_1}[\mathrm{Var}(\langle \hat{\theta}_{\text{Baseline}} - \theta, X_1 \rangle \mid \hat{\theta}_{\text{LuPTS}})] \ .
\end{aligned}$$

In the last step, we make use of the fact that $\hat{\theta}_{\text{Baseline}}$ is unbiased and

$$\mathbb{E}_D[R(\hat{\theta})] = \mathbb{E}_{X_1}[\mathbb{E}_D[\langle \hat{\theta} - \theta, X_1 \rangle^2] + \sigma^2 \ .$$

■

As pointed out already, the proof uses standard arguments from literature [5, 6]. Worthwhile to also mention is that it should be possible to reformulate Theorem 5 to other convex losses, as touched upon in Section 2.6. For example, we could consider the mean absolute error $||\hat{\theta} - \theta||_2$, or potentially the logistic loss function if extending the theorem for classification tasks. Note that we can not necessarily guarantee, however, that changing the loss function would lead to a gap between LuPTS and baseline in the equations in Theorem 5. And even if it exists, it might look different.

In the following subsections, we will present a set of remarks and comments which relates to Theorem 5, discussing various aspects and interpretations of it.

Remarks on the error gap in Theorem 5

As a reminder,

$$\mathbb{E}_D[\mathrm{Tr}(\mathrm{Cov}(\hat{\theta}_{\text{Baseline}} \mid \hat{\theta}_{\text{LuPTS}}))] \quad \text{and} \quad \mathbb{E}_{D,X_1}[\mathrm{Var}_{\hat{\theta}_{\text{Baseline}}}(\langle \hat{\theta}_{\text{Baseline}}, X_1 \rangle \mid \hat{\theta}_{\text{LuPTS}})]$$

are the gaps between the error of the baseline and LuPTS models. Although we have already discussed how these depend on the uncertainty in $\hat{\theta}_{\text{Baseline}}$ given $\hat{\theta}_{\text{LuPTS}}$, in this section, we will give two remarks to build further intuition for these.

Connection to second moment. To give an example for the two-step case with $T = 2$, i.e. $\hat{\theta}_{\text{LuPTS}} = \hat{A}_2\hat{\beta}$ we can show another way to interpret the covariance $\text{Cov}(\hat{\theta}_{\text{Baseline}} \mid \hat{\theta}_{\text{LuPTS}})$.

Let $B = (\mathbf{X}_1^\top \mathbf{X}_1)^{-1}\mathbf{X}_1$, then we have,

$$
\begin{aligned}
&\text{Cov}_D\left(\hat{\theta}_{\text{Baseline}} | \hat{\theta}_{\text{LuPTS}}\right) = \\
&\quad \overset{(a)}{=} \mathbb{E}_D\left[(\hat{\theta}_{\text{Baseline}} - \mathbb{E}[\hat{\theta}_{\text{Baseline}}|\hat{\theta}_{\text{LuPTS}}])(\hat{\theta}_{\text{Baseline}} - \mathbb{E}[\hat{\theta}_{\text{Baseline}}|\hat{\theta}_{\text{LuPTS}}])^\top | \hat{\theta}_{\text{LuPTS}}\right] \\
&\quad \overset{(b)}{=} \mathbb{E}_D\left[(\hat{\theta}_{\text{Baseline}} - \hat{\theta}_{\text{LuPTS}})(\hat{\theta}_{\text{Baseline}} - \hat{\theta}_{\text{LuPTS}})^\top | \hat{\theta}_{\text{LuPTS}}\right] \\
&\quad \overset{(c)}{=} \mathbb{E}_D\left[(B(\mathbf{Y} - \mathbf{X}_2\hat{\beta}))(B(\mathbf{Y} - \mathbf{X}_2\hat{\beta}))^\top | \hat{\theta}_{\text{LuPTS}}\right] \\
&\quad \overset{(d)}{=} \mathbb{E}_{\mathbf{X}_1,\mathbf{X}_2}\left[B\,\mathbb{E}_{\mathbf{Y}}\left[(\mathbf{Y} - \mathbf{X}_2\hat{\beta})(\mathbf{Y} - \mathbf{X}_2\hat{\beta})^\top | \hat{\theta}_{\text{LuPTS}}, \mathbf{X}_1, \mathbf{X}_2\right] B^\top | \hat{\theta}_{\text{LuPTS}}\right] \\
&\quad \overset{(e)}{=} \mathbb{E}_{\mathbf{X}_1,\mathbf{X}_2}\left[B\,\mathbb{E}_{\mathbf{Y}}\left[(\mathbf{Y} - \mathbf{X}_2\hat{\beta})(\mathbf{Y} - \mathbf{X}_2\hat{\beta})^\top | \hat{\beta}, \mathbf{X}_2\right] B^\top | \hat{\theta}_{\text{LuPTS}}\right] \\
&\quad \overset{(f)}{=} \mathbb{E}_{\mathbf{X}_1,\mathbf{X}_2}\left[B\,\text{Var}_{\mathbf{Y}}\left(\mathbf{Y} | \hat{\beta}, \mathbf{X}_2\right) B^\top | \hat{\theta}_{\text{LuPTS}}\right]
\end{aligned}
$$

where (a) comes from the definition of conditional variance; (b) uses Lemma 7; in (c) we use the definition of the OLS estimators; (d) is iterated expectations; (e) is possible due to the Markov assumption since $\mathbf{Y} \perp\!\!\!\perp \hat{A}_2, \mathbf{X}_1 | \hat{\beta}, \mathbf{X}_2$; and, finally, (f) uses, once again, the definition of conditional variance since we know $\mathbb{E}[\mathbf{Y}|\mathbf{X}_2, \hat{\beta}] = \mathbf{X}_2\hat{\beta}$ from Lemma 6.

We see that the gap depends on $\text{Var}_{\mathbf{Y}}\left(\mathbf{Y}|\hat{\beta}, \mathbf{X}_2\right)$, which is related to the second moment of Y conditioned on $\hat{\beta}$ and $\mathbf{X}_2$. Even though we can not show how this term depends on sample size, dimensionality or other problem parameters, an analogy can perhaps be made to Lemma 6 where we showed what the first moment is. Hence, learning more about the second moment seems to be of interest to understand the gap better. This remains as an open question to be answered.

Dependence on system and outcome noise Using a different lens, we can show how the system and outcome noise are key components to the difference between the LuPTS and baseline estimators, $\hat{\theta}_{\text{LuPTS}}$ and $\hat{\theta}_{\text{Baseline}}$. We can express the difference between the estimators explicitly in the case where $T = 2$. With $H_1 = (\mathbf{X}_1^\top \mathbf{X}_1)^{-1}\mathbf{X}_1^\top$ and $H_2 = (\mathbf{X}_2^\top \mathbf{X}_2)^{-1}\mathbf{X}_2^\top$, we have,

$$
\begin{aligned}
\hat{\theta}_{\text{LuPTS}} - \hat{\theta}_{\text{Baseline}} = &\underbrace{H_1\mathbf{X}_1 A H_2\mathbf{X}_2\beta}_{\theta} + (H_1\mathbf{X}_1 A H_2\boldsymbol{\varepsilon}_Y - H_1\boldsymbol{\varepsilon}_Y) + \underbrace{(H_1\boldsymbol{\varepsilon}_2 H_2\mathbf{X}_2\beta - H_1\boldsymbol{\varepsilon}_2\beta)}_{=0} \\
&- \underbrace{H_1\mathbf{X}_1 A\beta}_{\theta} + H_1\boldsymbol{\varepsilon}_2 H_2\boldsymbol{\varepsilon}_Y = (AH_2 - H_1)\boldsymbol{\varepsilon}_Y + H_1\boldsymbol{\varepsilon}_2 H_2\boldsymbol{\varepsilon}_Y \ .
\end{aligned}
$$

Whenever $\boldsymbol{\varepsilon}_Y = 0$, both remaining terms are 0, irrespective of other factors. If $\boldsymbol{\varepsilon}_2 = 0$ and A is invertible, both terms are 0 as well (when $\boldsymbol{\varepsilon}_2 = 0$, $\mathbf{X}_2 = \mathbf{X}_1 A$ and $AH_2 = H_1$). In other words, in the case where either the dynamics or the outcome are noiseless, the LuPTS estimator reduces to the OLS estimator. More importantly, if neither noise term is 0, the difference will generally not be 0 . As a consequence, $\text{Cov}(\hat{\theta}_{\text{Baseline}} \mid \hat{\theta}_{\text{LuPTS}}) > 0$, and LuPTS is strictly better than baseline.

Remarks on limitations and potential extensions of Theorem 5

The main limitation of Theorem 5 is that we make strong assumptions on the structure of the time series, i.e. Markov and linear transitions with additive, isotropic Gaussian noise. Even though we will see that the LuPTS algorithm works well on real-world tasks where these assumptions necessarily do not hold in Chapter 5, we will make a few remarks on possible ways to extend the theory.

Anisotropic noise The isotropic noise assumption, i.e. all covariances of the noise can be written as $\text{Cov}(\varepsilon_t) = \sigma_t^2 \, \text{I}_d$, is necessary to prove Lemma 6. However, the assumption is mostly made to make the analysis easier. For the anistropic case with $\text{Cov}[\varepsilon_t] = \text{diag}(\sigma_1^{(t)}, \ldots, \sigma_d^{(t)})$. Then, Lemma 6 is still feasible using a similar analysis. We have made a remark regarding this under the proof of the isotropic case in Appendix A.2.

Non-linear models It is not trivial to generalize Theorem 5 to a non-linear setting as we rely heavily on the definition and theory for OLS estimators. Hence, we do not see a straightforward way to apply the Rao-Blackwell arguments in the non-linear case. Still, it is possible to reason about non-linear models in the context of time series with privileged information, which we discuss further in Section 3.3.3 and Chapter 6.

Stationary dynamics As pointed out already, Theorem 5 only applies to Algorithm 1. Although the stationarity assumption is a further restriction, it is of interest to still understand how the stationary variant of LuPTS would compare to both the non-stationary variant and baseline. Potentially, it could be better than both of them if the stationary assumptions holds. However, the dependence between samples at each time step makes Lemma 6 more difficult to prove for the stationary case. Currently, we do not have a clear idea of how to approach it differently, but it is left as an interesting direction for future work.

Introducing interventions As mentioned in Section 2.5, it is common when studying linear dynamical systems to include an externally controlled signal V_{t-1} such that

$$X_t = A_t^\top X_{t-1} + B_t^\top V_{t-1} \varepsilon_t$$

where B_t is the linear parameter which accounts for the control signal. This could, for example, be used to model a physician who intervenes and changes the treatment dose for a patient for some time step $t > 1$ after baseline. In our analysis, we have considered that $B_t = 0$. However, it is trivial to extend Theorem 5 to this setting if V_t is known. We can introduce

$$\tilde{X}_t = \begin{bmatrix} X_t \\ V_t \end{bmatrix} \quad \text{and} \quad \tilde{A}_t = \begin{bmatrix} A_t \\ B_t \end{bmatrix} .$$

Then, we have

$$\tilde{X}_t = \tilde{A}_t^\top \tilde{X}_{t-1} + \varepsilon_t$$

which is formulated precisely as those linear dynamical systems that we are studying. Instead of inferring A_t, we learn $\tilde{A}_t$ but the same theory still applies.

Adversarial example The OLS estimator $\hat{\theta}_{\text{Baseline}}$ is the optimal unbiased linear estimator between the samples (X_1, Y) with respect to variance, as it is the minimum-variance mean-unbiased estimator of θ [17]. However, when having privileged information, this result do not necessarily apply as we have the intermediate variables $(X_2, \ldots, X_T)$. Still, evaluated using only (X_1, Y), baseline will always have a better or equal training error compared to the LuPTS algorithm (LuPTS is, however, expected to perform better on unseen data). This comes from the fact that the OLS estimator is the best solution to the least squares loss on the observed data. Consequently, it has an interesting implication. Although highly unlikely, if the test set is very similar or identical to the training set, then baseline will probably perform better (or at least equally well) on that particular example. This also gives the hint that when the sample size n is very large, we should expect that the difference between LuPTS and baseline should be very small since there are few unseen data points that could appear.

Algorithm 3 Generalization of Learning using Privileged Time series (LuPTS)

Require: Dataset $\{X_1^{(i)}, \ldots, X_T^{(i)}, Y^{(i)}\}_{i=1}^n$. Function classes $\mathcal{F}$, $\mathcal{G}$. Loss function L.

1: **for** $t = 1, \ldots, T-1$ **do**
2: $\quad \hat{f}_t = \underset{f_t \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^m L(f_t(x_{i,t}), x_{i,t+1})$
3: **end for**
4: $\hat{g} = \underset{g \in \mathcal{G}}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^m L(g(x_{i,T}), y_i)$
5: **return** $\hat{h} = \hat{g} \circ \hat{f}_{T-1} \circ \ldots \circ \hat{f}_1$

3.3.3 Extensions to non-linear models

The theory which has been presented up until this point has been for linear estimators. However, non-linear models are also of great interest due to their greater flexibility. But, unfortunately, the variance reduction using Rao-Blackwellization is not trivial to generalize to non-linear estimators. With that said, we can still reason about a more general variant of LuPTS.

Under the Markov assumption, defined in Section 3.1, it is natural to consider the following generalization of the LuPTS algorithm to non-linear estimators: (a) For each time step t in the series, fit an estimator of the transition from X_t to X_{t+1}, denoted $\hat{f}_t$; (b) fit an outcome model predicting Y from X_T, denoted $\hat{g}$; lastly (c) return $\hat{h} = \hat{g} \circ \hat{f}_{T-1} \circ \cdots \circ \hat{f}_1$ as the hypothesis. Predictions by the algorithm becomes $\hat{h}(X_1) \approx Y$. The steps described in this paragraph are outlined in Algorithm 3, where it is necessary to define the loss function and the functional classes $\forall t : f_t \in \mathcal{F}$, $g \in \mathcal{G}$ that we use to model the dynamics and outcome respectively.

In the general case, without assumptions on the data-generating process or the hypothesis classes $\mathcal{F}$ and $\mathcal{G}$, we may bound the expected risk of the LuPTS algorithm in terms of the risk accumulated in simulating the system dynamics through $\hat{f}$, and that of the outcome model $\hat{g}$. This is stated by the following theorem.

Theorem 8 (Error decomposition)**.** *Let $\hat{f} = \hat{f}_T \circ \hat{f}_{T-1} \circ \cdots \circ \hat{f}_2$ be the estimated system dynamics and $\hat{g}$ is the the estimated outcome model which maps from X_T to Y, then we have*

$$\begin{aligned}\mathbb{E}\left[\left(\hat{g}(\hat{f}(X_1)) - Y\right)^2\right] \leq \mathbb{E}\left[\left(\hat{g}(\hat{f}(X_1)) - \hat{g}(X_T)\right)^2\right] + \mathbb{E}\left[(Y - \hat{g}(X_T))^2\right] \\ + 2\sqrt{\mathbb{E}\left[\left(\hat{g}(\hat{f}(X_1)) - \hat{g}(X_T)\right)^2\right]\mathbb{E}\left[(Y - \hat{g}(X_T))^2\right]}\end{aligned} \tag{3.6}$$

where $\mathbb{E}\left[\left(\hat{g}(\hat{f}(X_1)) - \hat{g}(X_T)\right)^2\right]$ is the mean squared error in Y that stems from errors in the learned dynamical system while $\mathbb{E}\left[(Y - \hat{g}(X_T))^2\right]$ is due to the error in the model $\hat{g}$.

Proof. Proof as seen in [23].

$$\begin{aligned}\mathbb{E}\left[\left(\hat{g}(\hat{f}(X_1)) - Y\right)^2\right] &= \mathbb{E}\left[\left(\hat{g}(\hat{f}(X_1)) - Y + \hat{g}(X_T) - \hat{g}(X_T)\right)^2\right] \\ &= \mathbb{E}\left[\left((\hat{g}(\hat{f}(X_1)) - \hat{g}(X_T)) - (Y - \hat{g}(X_T))\right)^2\right] \\ &= \mathbb{E}\left[\left(\hat{g}(\hat{f}(X_1)) - \hat{g}(X_T)\right)^2\right] + \mathbb{E}\left[(Y - \hat{g}(X_T))^2\right] \\ &\quad - 2\,\mathbb{E}\left[\left(\hat{g}(\hat{f}(X_1)) - \hat{g}(X_T)\right)(Y - \hat{g}(X_T))\right] \\ &\leq \mathbb{E}\left[\left(\hat{g}(\hat{f}(X_1)) - \hat{g}(X_T)\right)^2\right] + \mathbb{E}\left[(Y - \hat{g}(X_T))^2\right] \\ &\quad + 2\sqrt{\mathbb{E}\left[\left(\hat{g}(\hat{f}(X_1)) - \hat{g}(X_T)\right)^2\right]\mathbb{E}\left[(Y - \hat{g}(X_T))^2\right]}\end{aligned}$$

The first equalities follow from simple algebra, and the inequality in the last step is possible because Cauchy-Schwarz inequality for random variables. ■

Although the bound on the risk for the LuPTS can not directly be compared to that of the baseline model, it does, however, give an indication of how the risk of the LuPTS model depends on the estimation of the dynamics $\hat{f}_{T-1} \circ \cdots \circ \hat{f}_1$ and outcome $\hat{g}$. In cases where the outcome model is estimated perfectly, the LuPTS model can still have an arbitrarily bad prediction error if the estimation of the dynamics is poor. In addition, we could anticipate that the expected risk associated with the estimated dynamics grows with T as the problem becomes more difficult. Meanwhile, the outcome model $\hat{g}$ does not depend on sequence length T as it always consists of one estimated function.

4

Experimental setup

This chapter describes the setup of the experiments which are performed. In Section 4.1, the implementation is outlined and in subsequent sections, Section 4.2.1, 4.2.2 and 4.2.3 the datasets used in the experiments are presented and it is described how the datasets were pre-processed for the experiment in Chapter 5.

4.1 Implementation

The LuPTS algorithm outputs several OLS estimates. To be exact, one for each state transition A_t using variables from the time series as appropriate, and one for the outcome model β. The stationary variant of LuPTS is similar, the difference being that there is only one estimate for the state transition A. The baseline model used for comparison is also an OLS estimate, given only the baseline and outcome variable. All OLS estimates were implemented using the LinearRegression class found in the Python module scikit-learn [24]. The LinearRegression implementation used for both of the LuPTS algorithms and the baseline model were unregularized.

When extending the algorithm to binary classification tasks the outcome model in the LuPTS algorithm and the baseline model used for comparison are replaced with the LogisticRegressionCV implementation in scikit-learn. The L_2 regularization parameter for these were tuned, for values from 1×10^{-4} to 1×10^{4}, using 5-fold cross validation on the training portion. Regularization was used as unregularized logistic regression does not converge if the problem happens to be linearly separable [25].

For regression tasks the models were evaluated with the *coefficient of determination* (R^2) and for classification tasks the models were evaluated with the the *Area Under the Receiver Operating Characteristic Curve* (AUC).

For the real-world datasets presented in Section 4.2, pre-processing steps were performed in preparation to use the data for training and evaluating the algorithms. Generally, these steps covered selection of time points from the time series in the datasets. That is; which time point was to be treated as the baseline features (X_1), what were the time points that are to be considered privileged during training ($X_2, \ldots, X_T$), and what would represent the outcome (Y). Additionally, missing

values in the training portion were imputed by mean imputation and the training portion was then standardized. These steps were implemented using the SimpleImputer and RobustScaler classes found in scikit-learn [24]. Note that for the synthetic data pre-processing was not necessary by construction.

All procedures for pre-processing real-world datasets, generating synthetic datasets, training and evaluating models were implemented in Python with the help of standard scientific modules such as NumPy and scikit-learn. The experiments were run on mid-tier laptops generally utilizing one CPU core. Running times for all experiments under this setup rarely exceeded a couple of minutes. The source code is available in a GitHub repository*.

4.2 Datasets

The following section provides a description for the datasets used for experiments. First, the generative procedure for the synthetic data is explained and, then, the real-word datasets are presented. In addition, an explanation about the pre-processing, training and evaluation is given.

4.2.1 Synthetic data

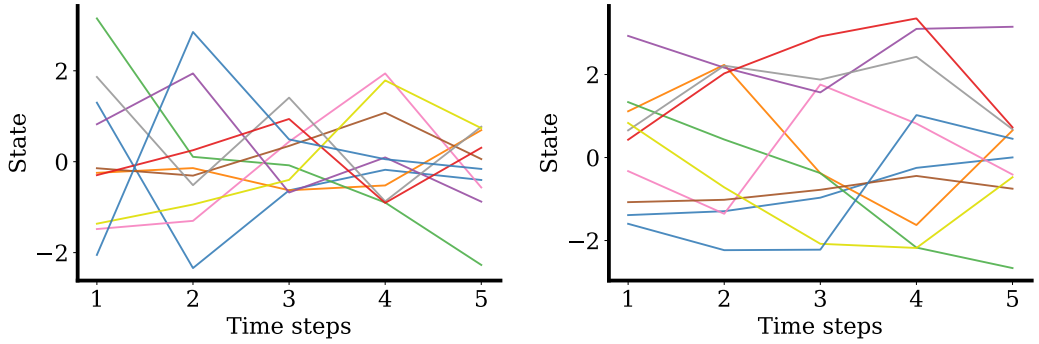
As a reminder, the Gaussian-linear dynamical system of interest is

$$\begin{aligned} X_t &= A_t^\top X_{t-1} + \varepsilon_t, \quad \text{for } t = 2, \dots, T \\ Y &= \beta^\top X_T + \varepsilon_Y \ . \end{aligned} \tag{4.1}$$

To verify and further investigate the theoretical results which have been presented, a sample procedure was used to generate a synthetic dynamical system on this form where the assumption under consideration holds, namely the Markov property and linearity with additive Gaussian noise. The parameter $A_t \in \mathbb{R}^{d \times d}$ and $\beta \in \mathbb{R}^{d \times 1}$ were generated in the following way: For each $t = 2, \dots, T$, all elements in A_t are sampled independently from a Normal distribution $\mathrm{N}(\mu = 0, \sigma = 0.2)$, except for the diagonal elements of A_t which were set to 1. If the system was stationary, the sampling procedure for A_t had to only be performed once. The linear parameter for the outcome model β was sampled from the same distribution, i.e. $\beta_j \sim \mathrm{N}(\mu = 0, \sigma = 0.2)$ for $j = 1, \dots, d$.

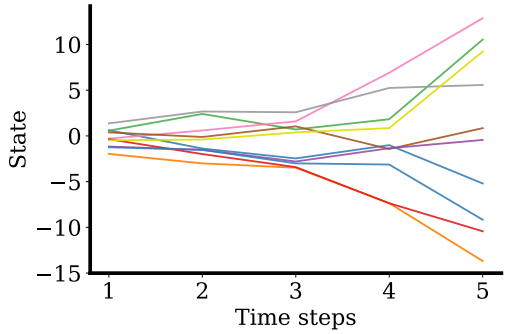
The eigenvalues $(\lambda_1, \dots, \lambda_d)$ of A_t has an influence on the systems behavior and stability. To ensure a desirable behavior of the system, the spectral radius $\rho(A_t)$ was enforced to a specific value after the sampling procedure. This was done by factorizing A_t into its spectral decomposition $U_t \Lambda_t U_t^{-1}$ and computing $\Lambda_t^{(new)} = \frac{\kappa}{\rho(A_t)} \Lambda_t$. Then, an update $A_t = U_t \Lambda_t^{(new)} U_t^{-1}$ was performed where κ is the new spectral radius of A_t. If $\kappa = 1$ then a stable system is achieved, but it is also possible to get dampening ($\kappa < 1$) and unstable ($\kappa > 1$) systems. Generally, $\kappa = 1.5$ was used since unstable systems are easier to learn [26].

*https://github.com/RickardKarl/LearningUsingPrivilegedTimeSeries



(a) Dampening system; spectral radius $\kappa = 0.2$

(b) Stable system; spectral radius $\kappa = 1.0$



(c) Unstable system; spectral radius $\kappa = 2.0$

Figure 4.1: Three different instances of linear dynamical systems synthetically generated by the procedure described in Section 4.2.1, but with various spectral radii. The outcome is not included in these figures, and note the different scales on the y-axis.

In addition to the parameters A_t and β, another key characteristics of the synthetic system is the distribution of the input X_1 and Gaussian noise variables $\varepsilon_t, \varepsilon_Y$. All were Normal distributed, $p(X_1) = N(\mu = 0, \sigma^2 = 5\,\mathrm{I}_d)$; the system noise variables $p(\varepsilon_t) = N(\mu = 0, \sigma^2 = 1)$ for $t = 2, \ldots, T$; and the outcome noise variable $p(\varepsilon_Y) = N(\mu = 0, \sigma^2 = 1)$.

In Figure 4.1, a dampening, stable and unstable system are generated with the procedure described here. Note that the variances in the final time step vary greatly depending on the stability of the system.

Breaking the Markov assumptions

Another similar system to (4.1) was considered where Y has a direct dependence on X_1, which breaks the Markov assumption. In this case, a parameter δ was introduced such that $Y = X_T\beta + X_1\delta + \varepsilon_Y$ where δ is generated in the same way as β.

4.2.2 Alzheimer's disease progression modeling

Alzheimer's is the most common cause of dementia and it is a progressive disease which worsens over time [27]. Predicting the progression of Alzheimer's is both helpful for the patient and clinicians studying the disease. For the patient it can aid in planning for the future, and for clinicians studying and working on treating the disease it is beneficial at an early stage to be able to predict if a patient is likely to develop Alzheimer's in the future. A treatment and the study of treatment effect is anticipated to be most fruitful if applied early in the disease progression [28].

The Alzheimer's Disease Neuroimaging Initiative (ADNI)[†] database was launched in 2003 as a public-private partnership, led by Principal Investigator Michael W. Weiner, MD. The primary goal of ADNI has been to test whether serial magnetic resonance imaging (MRI), positron emission tomography (PET), other biological markers, and clinical and neuro psychological assessment can be combined to measure the progression of mild cognitive impairment (MCI) and early Alzheimer's disease (AD). Subjects in ADNI are followed over several years with measurements taken at three months intervals.

The potential performance increase of having privileged information during training was investigated for disease progression prediction measured by two subject outcomes. The first was to predict a patient's Mini Mental State examination (MMSE) score four years from baseline measurements, the second was to predict a patients AD diagnostic status four years from the same baseline. To be more specific, outcomes (Y) are observed at a fixed follow-up time (48 months) after baseline (X_1) were taken in the ADNI study. Privileged information was collected at intermediate follow-up measurements, these were restricted to follow-ups at 12 months, 24 months and 36 months after baseline. Patient features were selected from three sets of demographic, cognitive test scores and bio-marker features present in the ADNI dataset. Examples of such are age, ADAS11 score and APOE4 gene expression. We refer the reader to Appendix B.1 for a full list of the features used for the Alzheimer's disease progression modeling tasks.

Pre-processing

There is a significant amount of missing values in the observations from the ADNI dataset, which is demonstrated by Figure 4.2. The missingness varies with the time of measurement, as does which subjects are present at certain follow-ups. Hence, a subset of the subjects in the study needed to be selected in order to carry out the experiments. This meant that subjects without an observation of the target outcome at the follow-up at 48 months were excluded. Furthermore, it was required that the subjects with an observation of the target at this time point also were present at the intermediate follow-ups used as privileged information, which were 12 months, 24 months and 36 months after baseline. Categorical features, such as biological sex and APOE4 gene expression, were one-hot encoded. Finally, if any feature had more than 70% of the observations missing for the selected subjects at any of the

[†]adni.loni.usc.edu

time points in consideration they were excluded. The features excluded as a result of this were FDG, ABETA, TAU and PTAU.

Data exploration

The selection criteria described in the pre-processing drastically reduced the number of subjects in the study which could be used for training and evaluation. The number of subjects present at the follow-up at 48 months after baseline were relatively small compared to the total number of subjects in the study, this number is then further decreased when selecting subjects with a measurement of the target outcome. For the MMSE prediction task, it resulted in a dataset with $n = 502$. Similarly, for the AD diagnosis prediction task, it resulted in a dataset with $n = 504$, out of which 133 had progressed into AD. This is also seen in Figure 4.3.

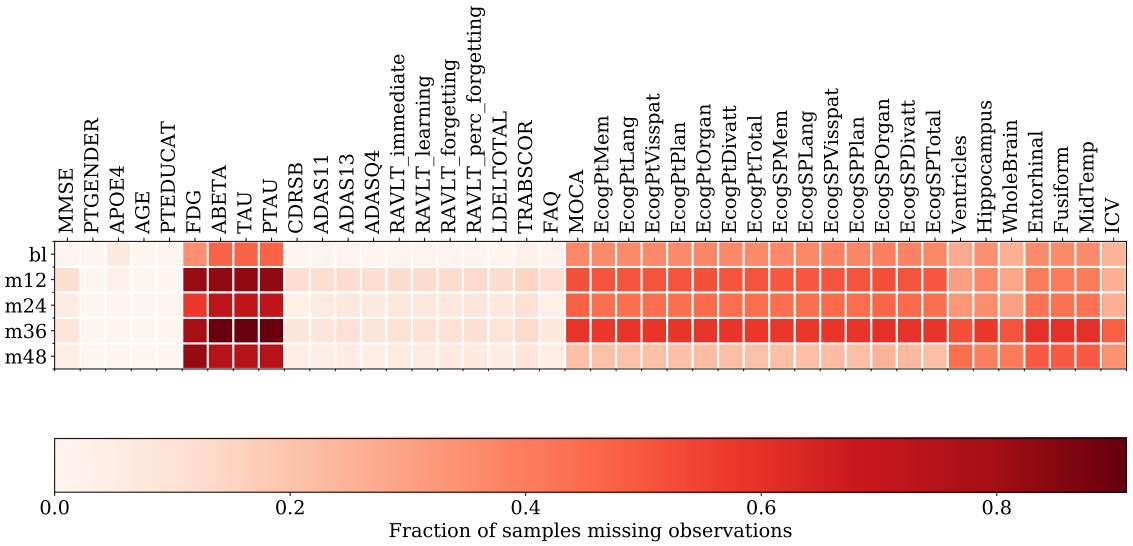


Figure 4.2: Ticks on the y-axis represent different times of measurement with *bl* for the baseline measurement and *m12* the follow-up 12 months after baseline etc. The fraction of observations with missing values per feature for the different time points of the selected subjects.

As previously mentioned there are some missingness in the observations. In Figure 4.2, the fraction of observations missing values per feature is shown for the different time points. The figure shows for the MMSE prediction dataset, but is representative for AD diagnosis task as well since there was a substantial overlap of subjects between them.

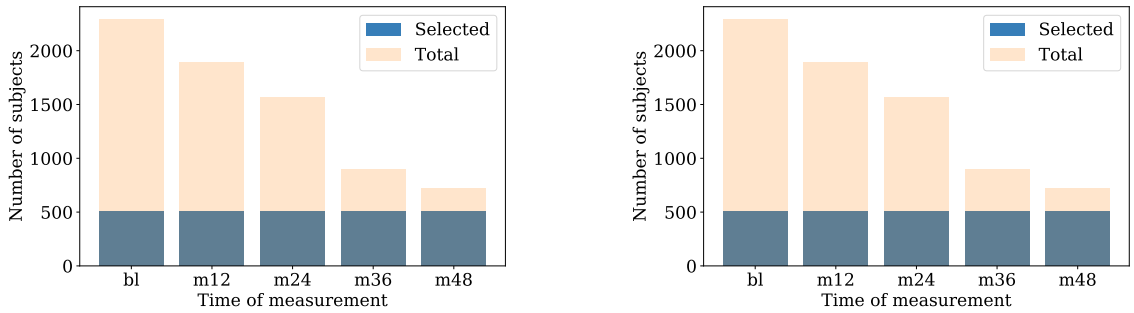


Figure 4.3: Ticks on the x-axis represent different times of measurement with *bl* for the baseline measurement and *m12* the follow-up 12 months after baseline etc. The total number of subjects are outlined at any given time point in the study and the number of those which were selected during pre-processing. **Left**: MMSE as target outcome. **Right**: AD diagnosis as target outcome.

4.2.3 Forecasting air quality in Chinese cities

Megacities in China such as Beijing and Shanghai are experiencing chronic air pollution with large concentrations of fine particles [29]. Due to the serious health risks caused by the pollution, it is of great interest to monitor the air quality and predict how it might change in the very near future [30]. This experiment was based on the task to forecast the $PM_{2.5}$ concentration, that is particles with a diameter of less than 2.5 μm, in different Chinese cities for the next couple of hours into the future. The $PM_{2.5}$ dataset contains hourly meteorological recordings between the years 2012 and 2014 of the fine particle pollution in Beijing, Chengdu, Guangzhou, Shanghai and Shenyang [31]. In addition to the $PM_{2.5}$ concentration levels in the air, the data contains 7 features such as temperature, humidity and combined wind direction, see Table 4.1 for the complete list.

The task can be described as following: At time $t = 1$ the features X_1 were observed, which also contained the current $PM_{2.5}$ air concentration, in one of the five cities. Based on this information, it is of interest to predict what the $PM_{2.5}$ concentration will be $T + 1$ hours into the future. The privileged information in the training data corresponded to the intermediate measurements $X_2, \ldots, X_T$ where the spacing between any adjacent time steps t and $t+1$ was one hour. This task was performed separately for each city, as their data look different from each other due to, among other things, geographical differences.

Pre-processing

Due to the prevalence of missing values for the $PM_{2.5}$ concentration levels in the dataset, the first pre-processing step was to extract all non-overlapping sequences of length T that had no missing values for the $PM_{2.5}$ concentration. In addition, a rule was enforced that there must be at least a gap of six hours between adjacent sequences to decrease correlations between them. Finally, dummy encoding was

Table 4.1: Features in the $PM_{2.5}$ dataset.

Feature	Type
Temperature	Numerical
Humidity	Numerial
Dew Point	Numerical
Pressure	Numerical
Cumulated wind speed	Numerical
PM2.5 concentration	Numerical
Season	Categorical
Combined wind direction	Categorical

used for the categorical features in the dataset. Standardization and imputation were applied during the training stage in the experiments.

Data exploration

To understand the $PM_{2.5}$ dataset better, a plot is included with correlations between the set of features contained in (X_1, Y). This is seen in Figure 4.4. Interestingly, there seems to be mainly three strongly correlated features with the target concentration at $t = 24$, which are the dew point, temperature and the baseline concentration level at $t = 1$.

With the pre-processing described in the previous subsection, the number of available samples will depend on the chosen time horizon T and also be different for each city as they have a varying degree of missingness in the target variable. The number of samples that were obtained for different settings are found in Table 4.2. Naturally, the number of available samples decreases when T increases. Additionally, in Figure 4.5, a few examples of the $PM_{2.5}$ concentration levels are displayed over the span of 24 hours in Beijing.

Table 4.2: Each entry contain the number of total samples for a particular city when pre-processing the data with respect to a fixed time horizon T.

City	$T = 5$ hours	$T = 11$ hours	$T = 23$ hours
Beijing	3543	2442	1466
Chengdu	2022	1392	844
Shanghai	2343	1609	961
Shenyang	1487	1008	594
Guangzhou	2246	1530	923

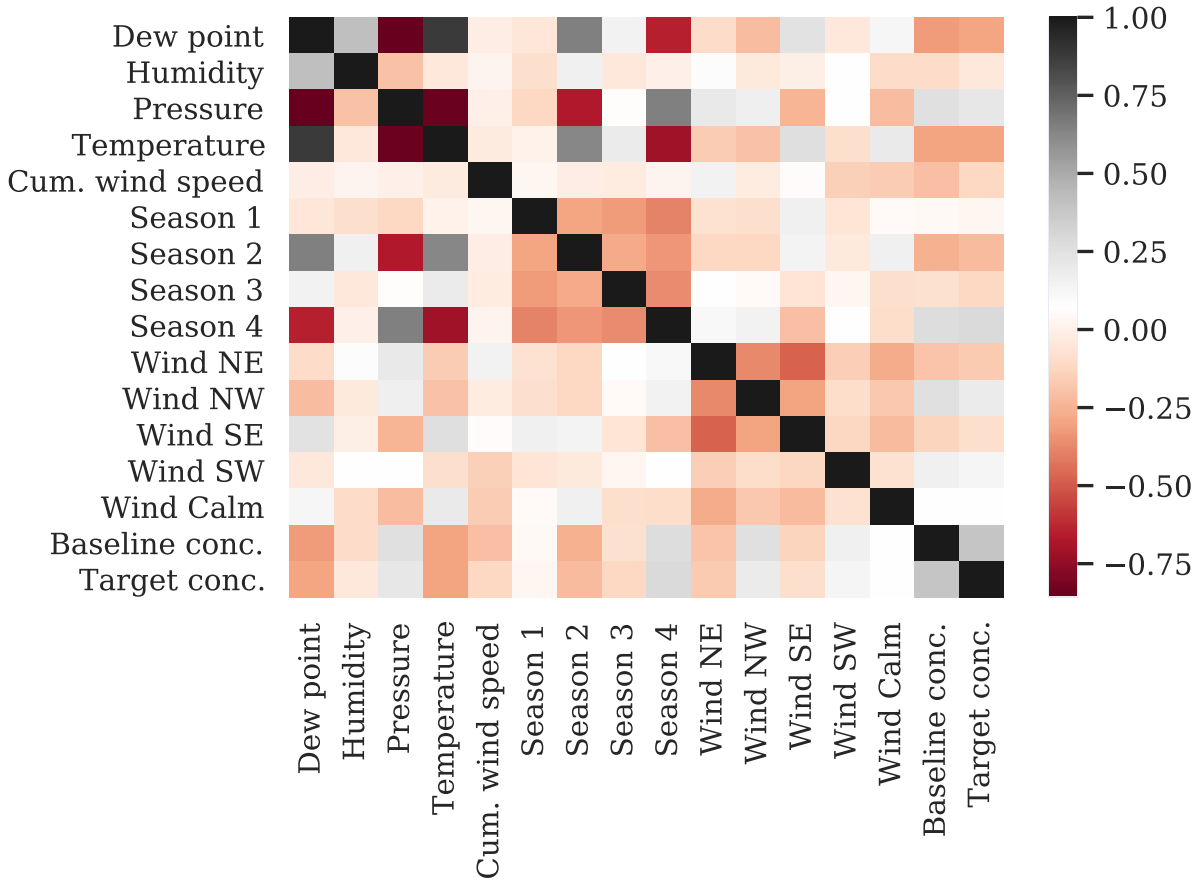


Figure 4.4: Heatmap with correlations between the features in Table 4.1, including the target concentration level at $T = 24$. The data is taken from Shanghai. The season and wind variables are binary.

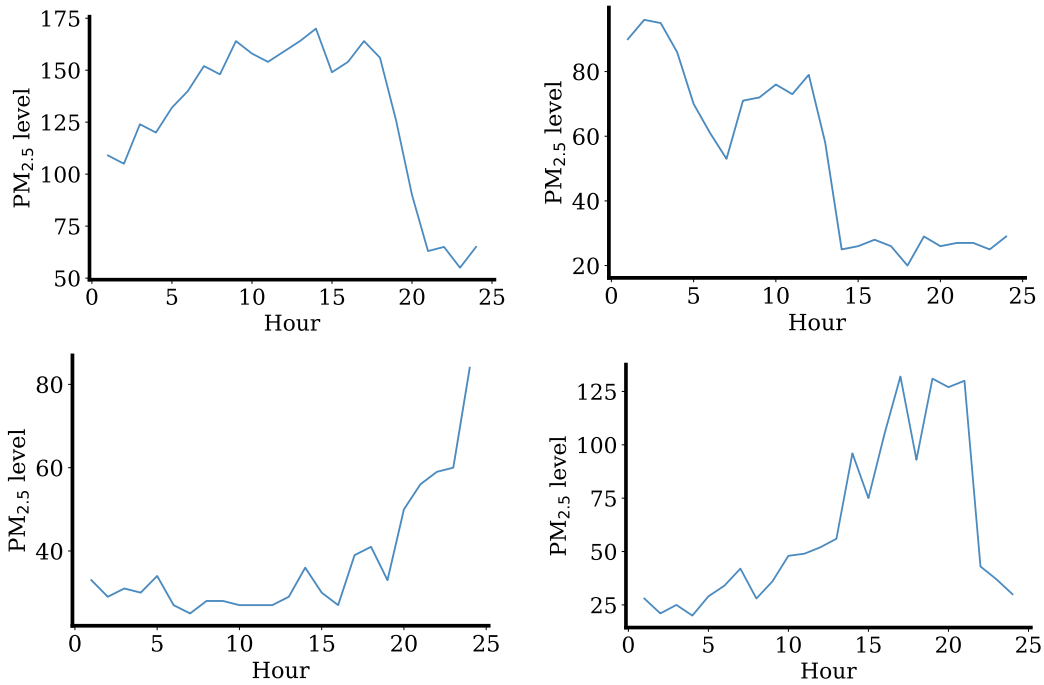


Figure 4.5: Examples of $PM_{2.5}$ concentration levels for Beijing over 24 hours; measurement unit $\mu g/m^3$.

5

Results

The results presented in this chapter are structured in the following way: First, the synthetic experiments are presented in Section 5.1, and then the results for the real-world tasks in Section 5.2.

In all plots, Baseline refers to OLS or Logistic Regression (depending on the task), LuPTS refers to the output of Algorithm 1 (non-stationarity) and Stat-LuPTS to the output of Algorithm 2 (stationarity).

5.1 Synthetic experiments

The main purpose of the synthetic experiments is to answer two questions: (1) Can we validate the theory presented in Chapter 3 and investigate the gap between baseline and LuPTS and how it varies depending on problem parameters? (2) What happens to the LuPTS algorithm when the underlying assumptions do not hold? Section 5.1.1 and Section 5.1.2 addresses question (1) and (2), respectively.

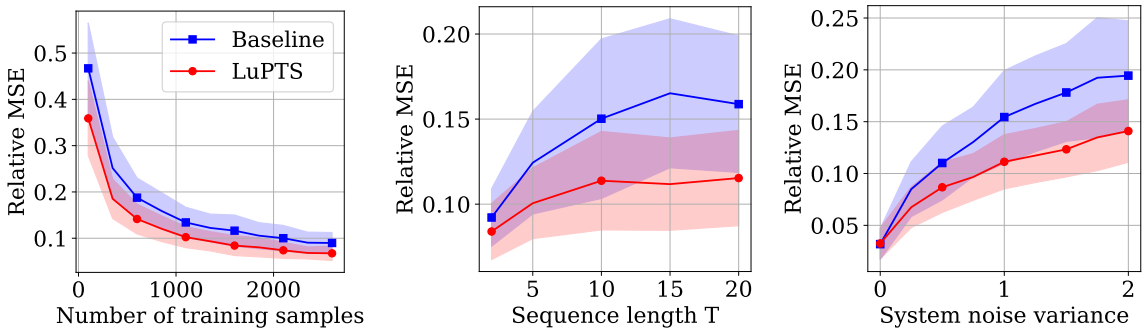
For all experiments, the following default values were used unless otherwise stated: $n = 1000$, $T = 10$, $d = 25$, $\kappa = 1.5$, and $\text{Var}(\epsilon_t) = \text{Var}(\epsilon_Y) = 1$ for $t = 1, \ldots, T-1$. Finally, the input distribution was $p(X_1) = \text{N}(\mu = 0, \sigma^2 = 5\,\text{I}_d)$. 200 iterations were performed in each experiment, and a new system was generated in each iteration. The size of the test set was equal to the size of the training set.

5.1.1 Validation of theory

In this section, experiments are presented that investigate the theoretical results from Chapter 3. In particular, they verify that Theorem 5 holds, and investigate how the gap between the baseline and LuPTS depends on problem parameters.

Parameter recovery

Figures 5.1a, 5.1b and 5.1c present the relative mean squared error $||\theta - \hat{\theta}||_2^2 / ||\theta||_2^2$ across multiple iterations of the parameter estimates for the baseline and LuPTS on a synthetic system. The impact of varying the number of training samples n,



(a) Varying number of training samples n.

(b) Varying sequence length T.

(c) Varying variance of system noise $\mathrm{Var}(\varepsilon_t)$.

Figure 5.1: Parameter recovery when varying one parameter at the time, while the others remained fixed to their default values. Relative MSE used as metric (lower is better); shaded region corresponds to one standard deviation over 200 iterations.

sequence length T, and variance of system noise on the MSE while keeping the other two fixed were investigated. It was noted that the LuPTS parameter estimates were closer to the true parameter values (indicated by a lower relative MSE) compared to the baseline estimator. The gap size between the methods seemed to vary greatly with respect to any of the problem parameters shown. In Figure 5.1b, the difference between the algorithms increased for larger T, which can be explained by the fact that there will be more uncertainty between X_1 and Y as T gets larger. Notably, when the variance of the system noise was set to zero (Figure 5.1c), no difference between LuPTS and the baseline method was observed. This result agrees with how the MSE gap in Theorem 5 depends on $\mathrm{Var}(\hat{\theta}_{\text{Baseline}} \mid \hat{\theta}_{\text{LuPTS}})$, which is only non-zero if there is uncertainty in the baseline estimate for a given estimate by the LuPTS algorithm.

Impact of system's stability

An investigation was done on how both the baseline and LuPTS algorithm performed on different linear dynamical system by varying the spectral radius. It affects the stability of the system, and can impact the algorithms capability to learn the dynamics as will be seen.

Let the spectral radius $\forall t : \rho(A_t) = \kappa$. A stable system has $\kappa = 1$, while an unstable/dampening system has $\kappa < 1$ and $\kappa > 1$, respectively. Figure 5.2 presents the relative MSE (y-axis) for parameter recovery when κ was varied (x-axis). LuPTS performed better than baseline for all values of the spectral radius, but they both got better when the radius increased. The observation that unstable systems were easier to learn overall has also been pointed out by [26].

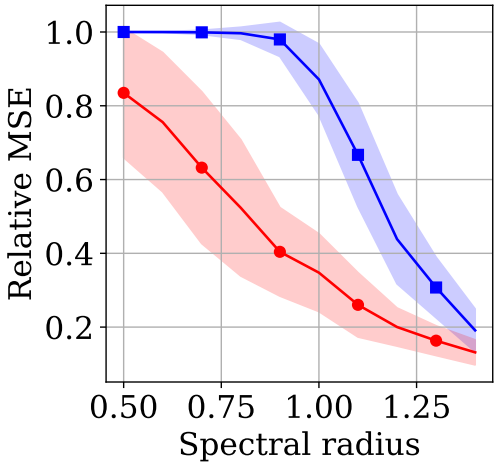


Figure 5.2: Parameter recovery (relative MSE) when varying spectral radius $\rho(A_t) = \kappa$. A stable system has $\kappa = 1$, while an unstable/dampening system has $\kappa > 1$ and $\kappa < 1$, respectively. Baseline (—■—) and LuPTS (—●—); shaded region corresponds to one standard deviation over 200 iterations. Lower is better.

5.1.2 Violation of assumptions

In the following section experiments are shown that investigate what happens when breaking one of the following assumptions¨: the Markov property, linearity or stationarity.

Breaking the Markov property assumption

Another linear parameter δ was introduced such that $Y = X_T\beta + X_1\delta$. The response became directly dependent on X_1. This means that Y is dependent of another variable beyond the previous state X_T, hence the Markov property does not hold. To investigate the impact of this violation, the ratio of the norms $\frac{||\delta||_2}{||\beta||_2}$ was varied. A larger ratio would lead to a larger deviation from the assumption while if $\delta = 0$ then the ratio is zero and the Markov property holds. Figure 5.3 shows the results that were obtained.

It was observed that predictions from LuPTS get worse in terms of R^2 score (y-axis) as the ratio (x-axis) grew. Albeit the bias, however, for small positive ratios it could be seen that LuPTS still performed as well as baseline. This might be explained by that the privileged information still contains useful knowledge which offsets the bias when it is small. However, the conclusion is that if the assumption about the Markov property is not true, then LuPTS should likely not be used.

LuPTS and non-linear dynamics

For investigating how the LuPTS algorithms performed when the assumption of linearity is broken, simulations of the non-linear Lorenz equations (5.1) were used. To compute the state $S_t = (u, v, w)_t,\ t = 1, .., T$ in the system, *odeint* from SciPy [32] was used with 0.01 time increments. Hence, state S_0 was observed at time 0, state

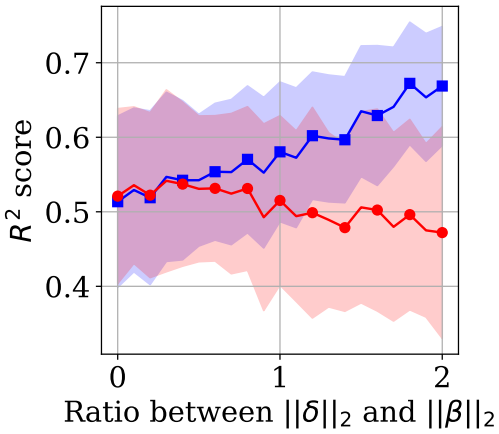


Figure 5.3: Adding a direct linear relationship between X_1 and Y, larger value on x-axis leads to more substantial violation of the Markov assumption. Baseline (—■—) and LuPTS (—●—). R^2 used as metric (higher is better); shaded region corresponds to one standard deviation over 200 iterations.

S_1 was observed at time 0.01 and so forth. In Figure 5.4 an example of a trajectory of the system is shown, note that the x-axis represents state index t and not the time increments. For the experiments the parameters of the Lorenz system were set to $\sigma = 10$, $\rho = 28$ and $\delta = \frac{8}{3}$.

$$\begin{aligned} \frac{\mathrm{d}u}{\mathrm{d}t} &= \sigma(v - u) \\ \frac{\mathrm{d}v}{\mathrm{d}t} &= u(\rho - w) - v \\ \frac{\mathrm{d}w}{\mathrm{d}t} &= uv - \delta w \end{aligned} \tag{5.1}$$

The outcome Y remained a linear function of the last state with additive Gaussian noise as in Section 4.2.1, i.e. $Y = \beta^\top S_T + \varepsilon_Y$ where β and ε_Y were sampled from the same distributions as the other synthetic experiments. The same was true for how the initial state S_0 was sampled as well.

Figure 5.4 shows the results of the baseline and LuPTS models for the Lorentz system. Note, a training size of 200 samples were used to produce the results. As can be seen, the baseline and both LuPTS variants had equal predictive strength but as the sequence lengths grew the performance of both LuPTS variants worsened, especially for Stat-LuPTS. The conclusion is that assuming linear estimators for the state transitions in this given situation was not ideal, and the experiments highlight a weakness of the LuPTS algorithm.

Investigating stationarity LuPTS

All synthetic experiments presented up until this subsection have been with non-stationary systems. Even though there are not the same theoretical guarantees for the stationary variant of LuPTS as with the non-stationary algorithm, an empir-

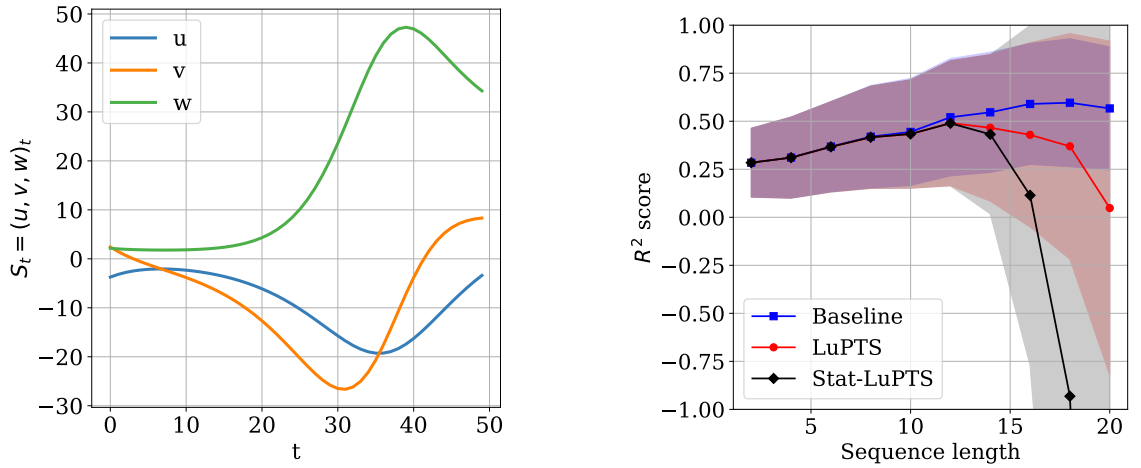


Figure 5.4: Left: Example trajectory from the Lorenz system. **Right**: R^2 score for the baseline and LuPTS models for varying sequence lengths. Shaded region corresponds to one standard deviation over 200 iterations.

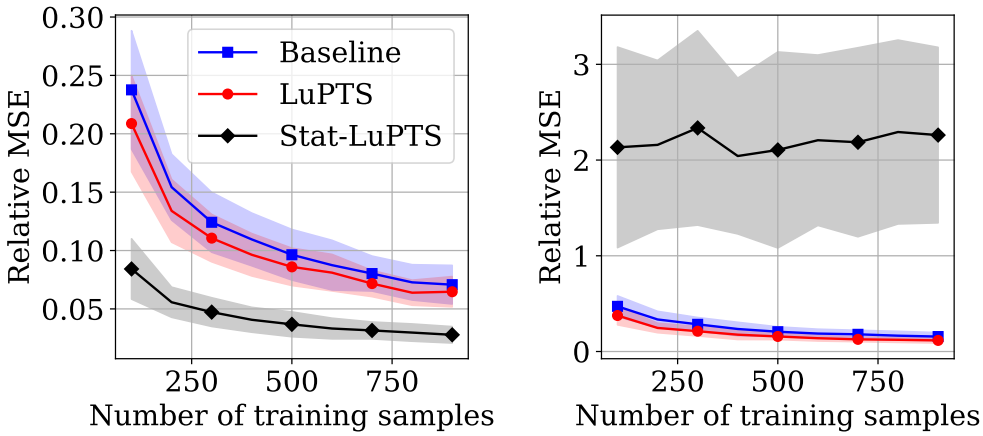
ical investigation was done to observe how it compared to the other algorithms. Two experiments were performed, one experiment where the stationary assumption held and one when it was violated, while observing the parameter recovery of each algorithm.

When the stationary assumptions held (Figure 5.5a), LuPTS did better than baseline as before. But more importantly, Stat-LuPTS was closer to the true parameter estimate than both of them. Meanwhile, as expected, when breaking the stationary assumption (Figure 5.5b), Stat-LuPTS performed significantly worse while LuPTS and baseline remained about the same.

These experiments indicate that the stationary variant of LuPTS is preferable when the stationarity assumption is true.

5.2 Real-world tasks

In this section, results are presented from the experiments with the Alzheimer's disease progression dataset (Section 5.2.1) and the air quality $PM_{2.5}$ dataset (Section 5.2.2). These experiments are motivated by a set of questions. Firstly, to verify that the LuPTS algorithm also works on real-world tasks that stretches outside the theoretical scope that has been considered so far. Secondly, to investigate how the amount of available privileged information in a particular task affects the LuPTS algorithm, in particular for low-sample settings. Thirdly, for the Alzheimer's disease progression task, testing an extension of the LuPTS algorithm to a classification task. Lastly, to study the bias-variance trade-off in the LuPTS algorithm, which is done mainly with the air quality forecasting task.



(a) Stationarity was true. **(b)** Stationarity was violated.

Figure 5.5: Parameter recovery with or without stationarity when varying the number of training samples n. R^2 used as metric; shaded region corresponds to one standard deviation over 200 iterations.

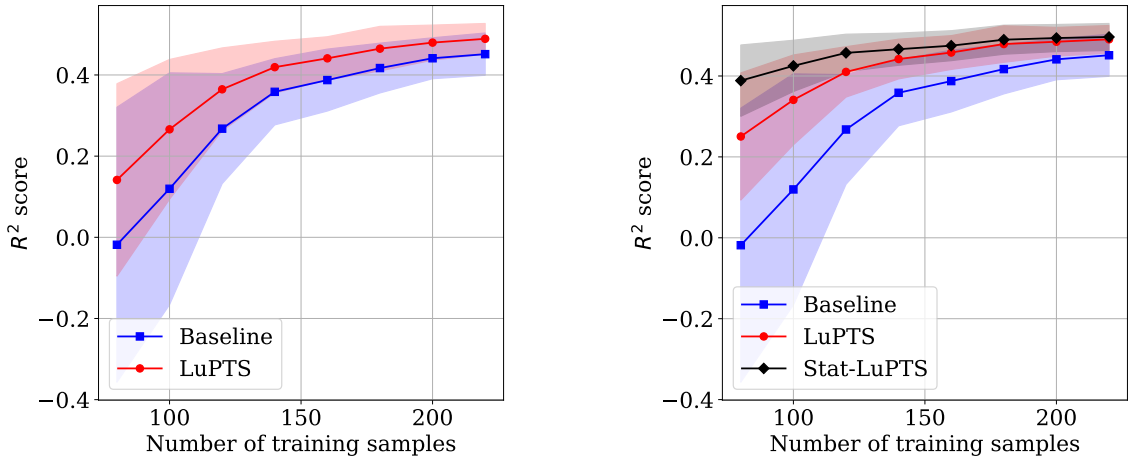
5.2.1 Alzheimer's disease progression modeling

Results are shown for the two tasks outlined in Section 4.2.2. For both the regression and classification task, an increase in predictive performance was observed when using LuPTS compared to the Baseline. This increase was further nuanced in the case where more intermediate time points are used as privileged information, especially for the LuPTS algorithm that assumes stationarity.

The datasets for each task were split into a training and test sets using 2-fold repeated cross validation with 50 repetitions resulting in 100 iterations total. For each iteration the baseline and LuPTS models were trained using a differing number of training samples sampled from the training set constituting the training portion. The models were then evaluated on the held-out test set and results were averaged over all iterations per training portion sample size.

Predicting Mini Mental State Examination score

For the Mini Mental State Examination (MMSE) score prediction task, LuPTS improved predictive performance over the baseline model for all training set sizes, as shown in Figure 5.6a and 5.6b, with the most noticeable improvement for small sizes. In addition to improved mean R^2 score the LuPTS model also had the added benefit of lower variance, with similar observations for how this depended on the training set size. This improvement is further increased when including more time points as privileged information during training. When using more than one intermediate time point as privileged information, using Stat-LUPTS resulted in even more of an improvement, over both the baseline model and non-stationary LuPTS. This can be seen in Figure 5.6b where the improvement in predictive performance was substantial, both in mean R^2 score and variance reduction.



(a) One privileged time point. m24.

(b) Three privileged time points. m12, m24, m36.

Figure 5.6: MMSE prediction task. Follow-up measurements used as privileged information per experiment outlined in subfigure captions, m12 corresponding to follow-up 12 months after baseline etc. R^2 used as metric (higher is better); shaded region corresponds to one standard deviation over 100 iterations.

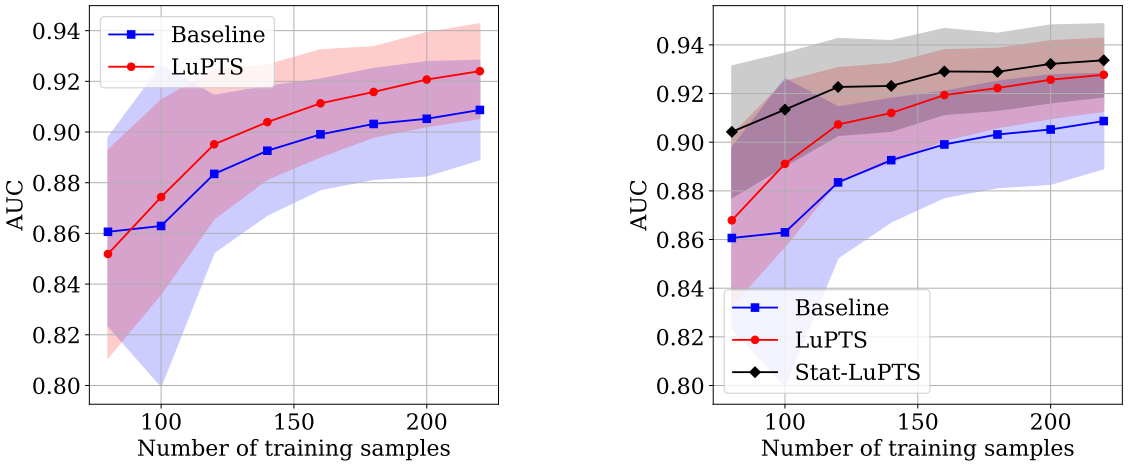
Predicting Alzheimer's disease diagnosis

The observations from the results of the Alzheimer's disease (AD) diagnosis prediction task, shown in Figure 5.7a and 5.7b, were similar to the ones of the MMSE prediction task. Using LuPTS improved the predictive performance over the baseline model. In contrast, the improvement was most apparent for large training set sizes. As for the MMSE prediction task; using more than one intermediate time point as privileged information further increased the improvement, especially for Stat-LuPTS. Notably, the LuPTS algorithm also had a substantial performance increase for small training set sizes.

Qualitative trajectory investigation

In Figure 5.6 and 5.7, it was shown that using LuPTS resulted in increased predictive performance over the baseline model for the Alzheimer's disease progression tasks. Given that the outcomes are predicted from the last time point of an estimated time series, $\hat{X}_T$, one question of interest is how well the LuPTS algorithm models the time series.

In Figure 5.8, the actual time series data (ground truth) and the corresponding predicted trajectory for all subjects in the test set are shown for the MMSE feature. As before, only half of the dataset was used for training. The LuPTS model managed to roughly capture the characteristics of the subjects MMSE progression over time. Additionally, the LuPTS seemed to capture the decline in subjects' MMSE score over time with the exception for subjects with a large deviation between baseline and follow-up measurement at 36 months. In Figure 5.9, the corresponding trajectories



(a) One privileged time point. m24.

(b) Three privileged time points. m12, m24, m36.

Figure 5.7: AD diagnosis prediction task. Follow-up measurements used as privileged information per experiment outlined in subfigure captions, m12 corresponding to follow-up 12 months after baseline etc. AUC used as metric (higher is better); shaded region corresponds to one standard deviation over 100 iterations.

are shown for the ADAS11 feature. The results are similar in nature to the MMSE predicted trajectories with the LuPTS models being largely able to capture the dynamics of the time series with the same exception.

The MMSE and ADAS11 trajectories are examples of when the LuPTS models captured the dynamics of the data relatively well. In Figure 5.10 a trajectory of the EcogPtMem feature is shown and compared to the LuPTS models predictions, note that the feature value range is a result that mean imputation was performed before standardization in tandem with a large amount of missingness. It is clear from this example that the LuPTS model did not learn the dynamics of the data, all features considered, with high fidelity. It can be seen that the subject progression (green) differed significantly from the predicted trajectories (red and black).

Note that the trajectories shown in Figure 5.8 and 5.9 indicate that the LuPTS model has qualitative limitations as a generative model. In particular, the actual scores shown in the figures are constrained to a particular range of values. Meanwhile, the predicted trajectory from the LuPTS is not bounded. For example, if the prediction would continue beyond the 36 months mark and further into the future, then, some subjects' MMSE score could potentially reach far outside the defined range of that clinical test. This would not have any semantic meaning for a working clinician.

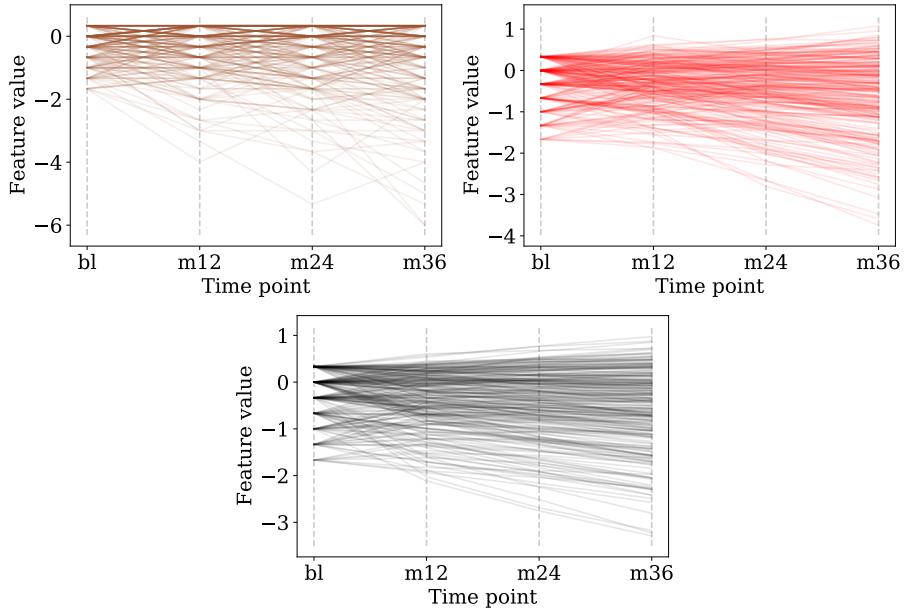


Figure 5.8: MMSE feature progression. bl corresponds to the baseline measurement, m12 to the follow-up 12 months after baseline etc. Top left: Subject progression. Top right: Predicted progression LuPTS. Bottom: Predicted progression LuPTS assuming stationarity.

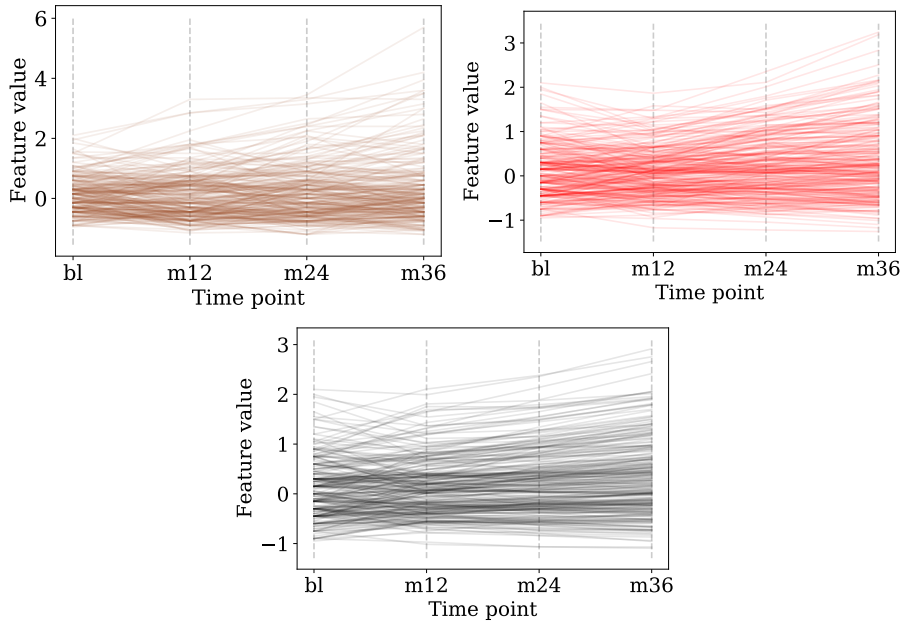


Figure 5.9: ADAS11 feature progression. bl corresponds to the baseline measurement, m12 to the follow-up 12 months after baseline etc. Top left: Subject progression. Top right: Predicted progression LuPTS. Bottom: Predicted progression LuPTS assuming stationarity.

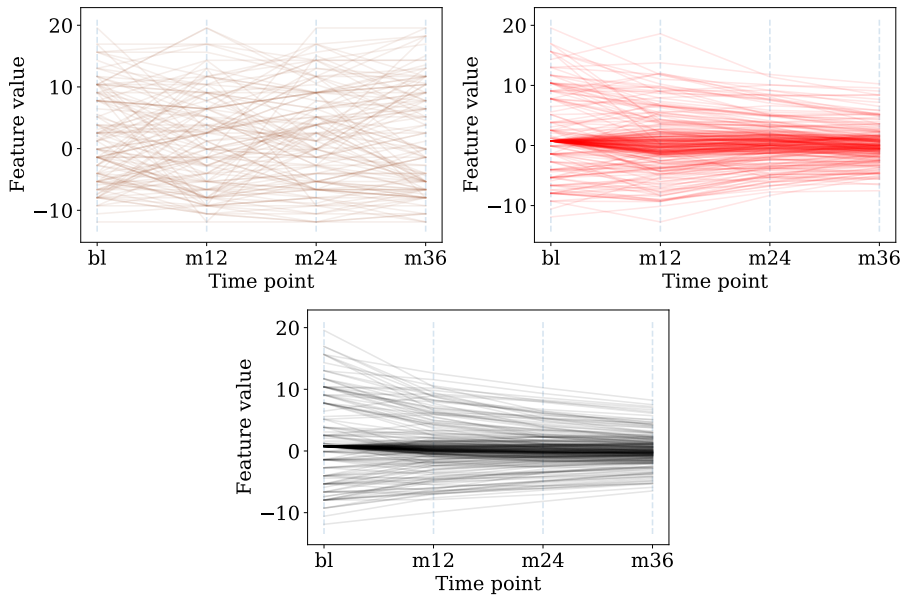


Figure 5.10: EcogPtMem feature progression. bl corresponds to the baseline measurement, m12 to the follow-up 12 months after baseline etc. Top left: Subject progression. Top right: Predicted progression LuPTS. Bottom: Predicted progression LuPTS assuming stationarity.

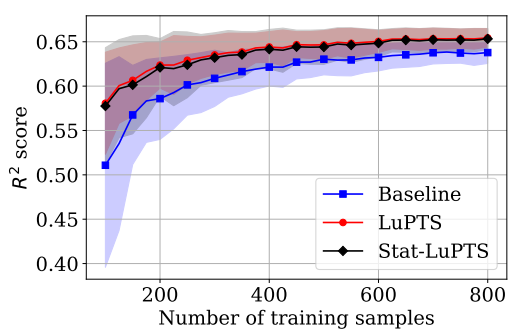
5.2.2 Forecasting air quality in Chinese cities

The dataset was split into a training and test set portion consisting of 80% and 20% of the data respectively. The training procedure on the dataset for the forecasting task was the following: The number of training samples were varied, and for each sample size, data points was randomly sampled from the training set without replacement. Then, before training the algorithms on this set, standardization and mean imputation were applied when applicable. Each algorithm were evaluated after training on a held-out test set which was the same for every run, and the corresponding R^2 score was noted. This process was iterated 200 times per sample size.

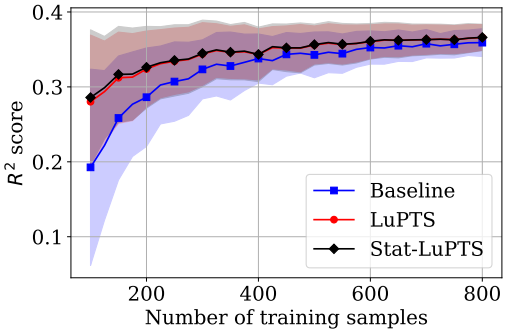
The $PM_{2.5}$ concentration was forecasted for several time horizons, mainly due to two reasons. First, predictions further into the future are more challenging. Second, for longer horizons, more time points can be used as privileged information. The included time horizons were 6, 12, or 24 hours into the future. At time $t = 1$, the features X_1 were observed, which also contained the current $PM_{2.5}$ concentration. Based on this information, $PM_{2.5}$ concentration was to be predicted $T+1$ hours into the future. The intermediate measurements $X_2, \ldots, X_T$ was considered privileged information where the spacing between any adjacent time steps t and $t+1$ was one hour.

In the following paragraphs, results are presented from the forecasting task. For each experiment, results are showed for the different cities contained in the dataset. However, to emphasize the observations that answers our questions, as well as to

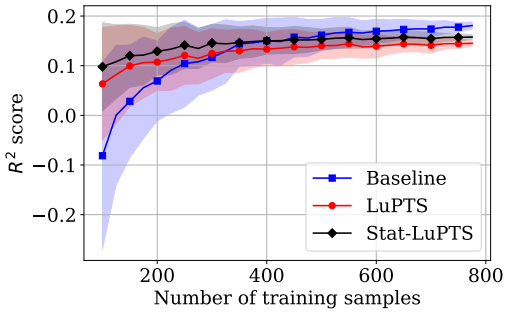
improve the reading experience, some figures have been moved to Appendix B.2. However, the most important findings are found in this section.



(a) Forecasting 6 hours into the future



(b) Forecasting 12 hours into the future

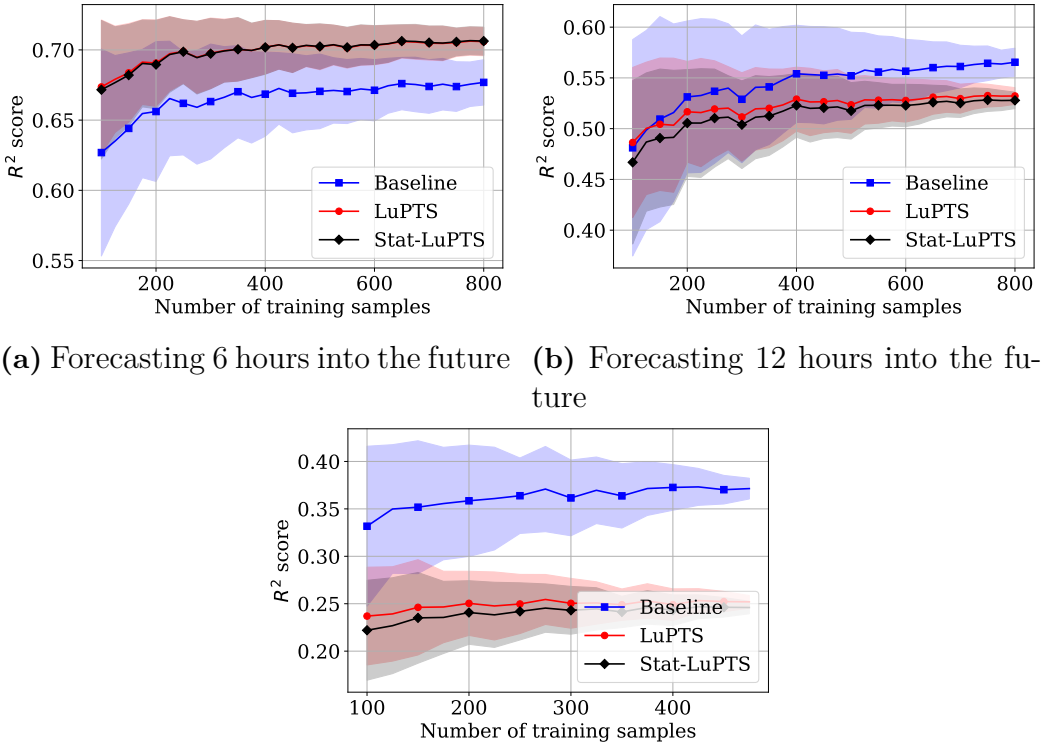


(c) Forecasting 24 hours into the future

Figure 5.11: Shanghai: Changing the time horizon to predict the $PM_{2.5}$ concentration levels. R^2 used as metric (higher is better); shaded region indicates one standard deviation across 200 iterations.

The first results are found in Figure 5.11 and 5.12, which display the R^2 score for the different time horizons for Shanghai and Shenyang, respectively. In all experiments, the maximum amount of privileged information were included, i.e. all intermediate hours in between the baseline and outcome measurement. For the 6 hour forecast (Figure 5.11a and 5.12a), improved performance was observed when using both LuPTS variants for all sample sizes. However, there seemed to be no difference between Stat-LuPTS and LuPTS. Also, note that both of them performed particularly well in the low-sample regime. For the 12 hour and 24 forecast, the results depicted another picture of both LuPTS and Stat-LuPTS. For Shanghai (Figure 5.11b and 5.11c, respectively), the results were very similar to what was observed for the 6 hour forecasting task. The only difference is that LuPTS actually performed slightly worse on the 24 hour forecast when the number of training samples was very large, which will be discussed why in a subsequent paragraph. For Shenyang, however, both Stat-LuPTS and LuPTS performed significantly worse on the task with all, except for very small, tested number of training samples (Figure 5.12b and 5.12c).

In addition to the above results, results are displayed in Table 5.1 for all cities with



(a) Forecasting 6 hours into the future

(b) Forecasting 12 hours into the future

(c) Forecasting 24 hours into the future

Figure 5.12: Shenyang: Changing the time horizon to predict the $PM_{2.5}$ concentration levels. R^2 used as metric (higher is better); shaded region indicates one standard deviation across 200 iterations.

experiments performed in a low-sample regime of $n = 100$. Foremost, note that both LuPTS variants performed better in terms of R^2 score, and there was a large variance reduction for many of the cities compared to baseline. Also, XGBoost [33], a tree-based gradient boosting algorithm, was included as a reference which can be compared to the linear models. For all cities, XGBoost performed significantly worse than both baseline and the LuPTS variants.

As observed before, the LuPTS variants performed slightly worse on the 24 hour forecast when the number of training samples was large (Figure 5.11c). This stems from a trade-off between bias and variance for the LuPTS algorithm. In the next experiments that shall be presented, this will be highlighted further.

Bias-variance trade-off when varying sequence length and privileged information

The time horizon have already been adjusted in the previous experiments, but these experiments will investigate what happens when adjusting the available amount of privileged time points in between the baseline and outcome measurements. For example, it would be possible to only include measurements from every third hour, or even less often. In Figure 5.13, the results are for the three forecast horizons

Table 5.1: Results from evaluation in the low-sample regime $n = 100$ with $T = 6$ for each Chinese city. Average R^2 score with standard deviation in parenthesis from 200 iterations. Higher is better.

Method	Beijing	Chengdu	Guangzhou	Shanghai	Shenyang
Baseline	0.61 (0.04)	0.56 (0.09)	0.36 (0.10)	0.50 (0.12)	0.62 (0.08)
LuPTS	0.62 (0.04)	0.61 (0.06)	0.44 (0.07)	0.58 (0.06)	0.67 (0.04)
Stat-LuPTS	0.62 (0.04)	0.61 (0.06)	0.44 (0.07)	0.57 (0.07)	0.67 (0.05)
SelectBest	0.61 (0.04)	0.58 (0.09)	0.40 (0.09)	0.56 (0.07)	0.64 (0.08)
XGBoost	0.48 (0.10)	0.43 (0.17)	0.29 (0.17)	0.38 (0.35)	0.38 (0.11)

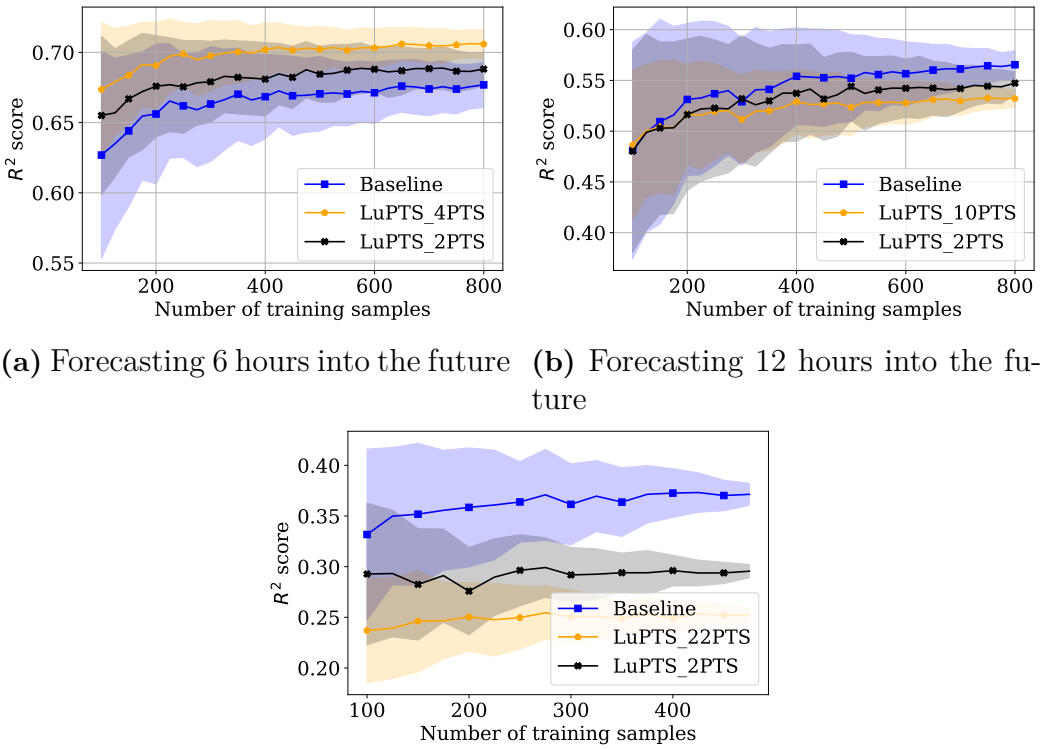
in Shenyang with different numbers of evenly spaced time points used as privileged information.

As noted before, for the 6 hour forecasting task (Figure 5.13a), LuPTS performed better. But more importantly, adding more privileged time points was beneficial. However, the results for the longer time horizons were different (Figure5.13b and 5.13c). LuPTS performed worse than baseline but, in addition, increasing the number of privileged time points had a negative impact on the performance. This may be due to the learned dynamical system being biased, and consequently, the bias compounds when the predicted "roll-out" is longer. This argument also explains why using more privileged time points would be subpar if the bias was large already. On the other hand, adding more privileged information reduced the variance.

Testing a model selection procedure

As seen in Figure 5.11b and 5.11c, it is possible that LuPTS can perform worse on certain tasks. Hence, it would be of interest to be able to decide on beforehand whether to use baseline or LuPTS. One way to approach this is to split the training set into two parts, one which is used to train both the baseline and LuPTS algorithm while the other part is held-out as a validation set. Then, select the algorithm which performed best on the validation set and re-train that algorithm on the entire training set again. This is precisely what the SelectBest algorithm did in the experiments shown in Figure 5.14.

In the experiments, SelectBest closed some of the gap between baseline and LuPTS. However, for the more extreme case with Guangzhou (Figure 5.14b), the variance became very large. This was also noticeable for Beijing, although not as discernible (Figure 5.14a). One reason for why SelectBest had such a large variance could be due to the small sample sizes, as it limits the size of the validation set. To conclude, it appears that the model selection procedure performed by SelectBest could bring some benefits, but that, in its current form, it should not be considered as an robust alternative to alleviate the problem of deciding which algorithm to use.

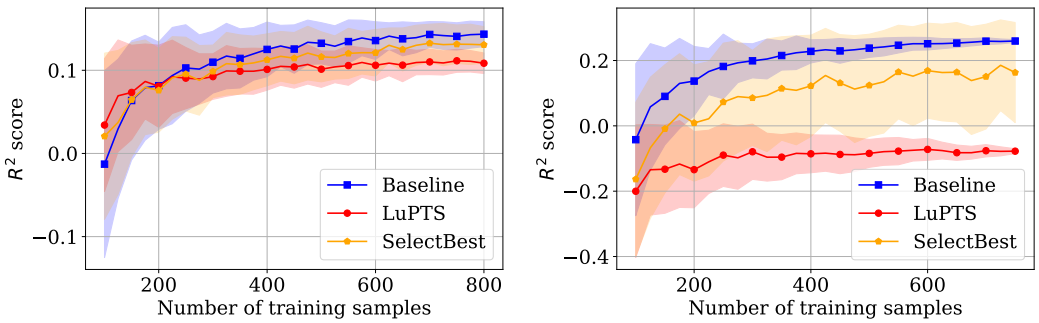


(a) Forecasting 6 hours into the future

(b) Forecasting 12 hours into the future

(c) Forecasting 24 hours into the future

Figure 5.13: Shenyang: Changing the amount of privileged information for the LuPTS for different time horizons, where **X** in LuPTS_**X**PTS indicates the number of privileged time points. R^2 used as metric (higher is better); shaded region indicates one standard deviation across 200 iterations.



(a) Beijing

(b) Guangzhou

Figure 5.14: Forecasting 24 hours into the future, where SelectBest is the best-performing of the two algorithms based on a held-out validation set. R^2 used as metric (higher is better); shaded region indicates one standard deviation across 200 iterations.

6

Discussion

In this chapter, we shall briefly discuss the things that we necessarily have not mentioned or inspected thoroughly due to our limited scope, but which could still be of interest. Throughout this chapter, we also propose some directions for future work which we believe are relevant and compelling to pursue.

6.1 Limitations with the theory and algorithm

Our theory relies on a set of key assumptions, mainly that we have a Markov-Gaussian-linear dynamical system. These assumptions have made the theoretical analysis more convenient, and we have proved guarantees under these conditions. However, we acknowledge that these are also very strong assumptions that necessarily do not match complex processes in the real-world. From Chapter 5 we showed that LuPTS can perform well even if the assumptions are not enforced, but it is still important to remember that we are limited in what we can claim about the algorithm overall.

Linear models, although an idealization of the real-world, are not necessarily bad. Particularly in the a low-sample setting, they can work efficiently. In Section 5.2.2, we showed how a tree-based model did not performs as well as the linear baseline. Still, considering non-linear and non-parametric models is a relevant topic as they have many benefits as well. Theorem 5 is, however, not easily extended to these types of methods since the Rao-Blackwellization relies heavily on the fact that we can compute the expectation over a closed-form expression of the estimator. This is generally not possibly for all models. Because of this, other theoretical tools may have to be used to study a non-linear variant of the LuPTS algorithm. There is extensive theory on deriving generalization bounds on the expected risk for non-linear classes [34]. Another approach is to use kernel methods which can be combined with linear estimators to learn non-linear relationships [16]. Ideally, we would still be able to answer the question whether privileged information is beneficial to improve the sample efficiency, but for a broader class of functions.

Although the Markov assumption has been one of the main restrictions under consideration, loosening it could be possible. An interesting extension in particular is

to generalize our results to any type of graph structure, i.e. going beyond a chained sequence, and consider general Bayesian networks instead. However, going in this direction would also raise the question about in which scenarios it is both useful and reasonable to assume that the data have a more complicated graph structure.

Finally, we have restrictions of the noise in the theory, which is assumed to be isotropic and Gaussian. In Chapter 3, we discussed the former and concluded that it is possible to introduce anisotropic noise instead, but that it is inconvenient to deal with. Regarding other variants of noise, we could formalize the problem with generalized linear models [16] which can incorporate correlated noise. For non-Gaussian noise, however, it does not appear trivial to extend the theory. Mainly, the proof of lemma 6 depends on this assumption in multiple places, e.g. that the distribution of the OLS estimators are Gaussian as well due to Gaussian noise.

6.2 Extensions of the theory and algorithm

In our work, we have mainly focused on answering the question whether privileged information is beneficial to improve the sample efficiency when compared to not having it. We formulated this as finding out what the gap was between these two methods. But there are many more interesting questions which could be asked with respect to using privileged information for time series data.

Related to the gap in Theorem 5, we could try to understand it further by characterizing it with respect to the the number of training samples or problem dimension.

We could also characterize the LuPTS algorithm's performance when using privileged information by finding a lower bound and upper bound on its risk, i.e.

$$l_{\text{LuPTS}} \leq R(\hat{f}_{\text{LuPTS}}) \leq u_{\text{LuPTS}}$$

where l_{LuPTS} and u_{LuPTS} could be functions of the training sample size, number of features and more. Continuing with this, we could compare these bounds to other bounds for a baseline algorithm without access to the privileged information,

$$l_{\text{Baseline}} \leq R(\hat{f}_{\text{Baseline}}) \leq u_{\text{Baseline}} \ .$$

The bounds l_{Baseline} and u_{Baseline} would also be functions of the same problem parameters. Then, ideally, we would be interested in knowing if $u_{\text{LuPTS}} \leq l_{\text{Baseline}}$ holds for some given functional class that f_{LuPTS} and f_{Baseline} belong to.

Another example could be to use the LuPTS algorithm for causal questions. Especially in the case where $T = 2$ for LuPTS, i.e. we have a two-step regression problem, similarities can be made to instrumental variable (IV) regression, also known as two-stage least squares [35]. In this case, the assumed structure of the data closely resembles what we have been studying. In literature, however, IV is commonly used to correct for bias when estimating effects from causal variables, e.g. the treatment effect from a particular medication. Meanwhile, questions regarding

LuPTS have been mainly revolved around variance reduction. Still, there are potentially interesting links which can be explored between these two areas. Lastly, it is also interesting to consider if using privileged information in time series can improve the identification of causal effects in general.

6.3 Bias-variance trade-off

For the analysis and theory presented in Chapter 3, the underlying distribution for the data followed the assumptions, i.e. Markovian and linear with additive, isotropic and Gaussian noise. In this well-specified setting, the improvement in performance using the LuPTS model is realized by variance reduction only, since the baseline model and LuPTS are both unbiased in this setting and share the irreducible error emanating from noise in the dynamics and outcome.

In misspecified settings, however, we could argue that the LuPTS model is more sensitive to bias in contrast to the baseline model. More specifically, it is exposed to bias from a misspecified dynamical system which the baseline model is not. This was demonstrated when we performed, among other things, an experiment breaking the Markov assumption in Section 5.1.2. Furthermore, an additional example of how LuPTS could suffer from bias was observed for the air quality forecasting task (Section 5.2.2). In that case, more privileged information seemed to increase the bias of the model, while at the same time reducing the variance. Interestingly, this depicts a picture that the usefulness of privileged information can vary, which leads us to propose the following conjecture: If the LuPTS is well-suited for a particular task where the chance of bias is small, then, adding more privileged information is better. Otherwise, privileged information is necessarily not useful.

Related to the bias-variance trade-off, a connection can be made to regularization where the same type of trade-off is well-known when using e.g. lasso and ridge regression [16]. It is therefore interesting to consider the usage of privileged information as a form of regularization, but additionally, it would also be of interest to investigate how these traditional forms of regularization could be combined with LuPTS. This would likely lead to an even larger bias in the algorithm, while also leading to a further reduction in variance. But potentially, for the right tasks, this could reduce the prediction error overall.

6.4 LuPTS for practical usage

An aspect that has been out of the scope of this thesis is how to make the concept of learning using privileged time series more applicable. Our focus has been on a theoretical and empirical understanding of asking whether privileged information is beneficial or not, but we have not dedicated enough effort to build an algorithm with the highest achievable predictive performance. However, this question could be relevant if we would wish to deploy the algorithm for a real-world task. A natural extension is to use other functional classes, such as non-linear models that might

prove more flexible. In particular, recurrent neural networks [36] can be used to simulate a trajectory, and there is work on using another variant of deep neural networks to model differential equations [37] which encompass a wider range of dynamical systems. With that said, we should still emphasize that a linear model can still be useful. In fact, the ease of implementing and training linear models could be seen as a strength with respect to practical usage.

On top of that, ensuring safe usage of the algorithm is crucial. As we have seen in some experiments, LuPTS can fail, which in high-stakes applications, such as in healthcare, would not be acceptable. Hence, an important question to address is to have a robust approach to deciding when privileged information should be used and when it should not, as we showed that this is not trivial with the SelectBest algorithm in Section 5.2.2. Another idea that could be tested is to have a regularized model with a feature vector containing both the the baseline variables X_1 and the predicted variables $\hat{X}_t$, and we could apply a feature selection procedure to select the most predictive features.

Additionally, another important aspect is how to deal optimally with missing values in the data, in particular if it is the privileged information which is mostly missing. For example, it is common in longitudinal datasets to have censoring, where there is time periods with partially unobserved data points but the outcome is still known. An interesting question is how to use the intermediary privileged information to learn the dynamics despite of partial censoring.

Lastly, it is of further interest to investigate how the LuPTS generalizes to classification tasks as we noted that it could work for predicting Alzheimer's disease progression.

6.5 Ethical aspects

Although the LuPTS algorithm presented in this thesis shows promise and, in many cases, improvements in predictive performance for the real-world tasks, caution should still be taken before deploying this algorithm in high-stakes settings without a thorough investigation of its efficacy. Implementations can have a wide-reaching effect depending on where it is applied. Especially for the clinical task presented in this thesis since around 35 million people worldwide suffer from Alzheimer's disease [27].

6.6 Future work

We briefly mention a couple of areas, some of which have been discussed in this chapter, for future work.

There are many possibilities to extending the theory presented in this thesis, but also more generally for Learning using Privileged Time Series. As already proposed

earlier in this chapter, investigating theory for non-linear models could be studied in this context and would be promising for complex real-world tasks. Other extensions such as non-Gaussian noise or different graph structures than a Markov chain are also of interest.

Investigating the idea of privileged information with time series in more practical applications would be a relevant topic as well. For example, how to better deal with missingness or implementing non-linear models in this framework.

Another direction of research is to investigate a mixture of linear dynamical systems that interacts with a latent variable. For example, a latent variable could be the unobserved variable representing the sub-type of a particular disease, which lead to different paths of disease progression (i.e. "dynamics") within a group of people affected by the disease. The latent variable may be easier to infer with the help of the privileged time series. In addition, there might still be a potential gain in sample efficiency when learning the actual predictive model using LuPTS.

7

Conclusion

In our thesis, we began by asking the question whether privileged information can be used in a context with time series data. In particular, we wanted to investigate an algorithm for *Learning using Privileged Time Series* (LuPTS) that utilizes privileged information and decide if it performs better than an algorithm without access to this information. To answer the question, we set out two main lines of work: (1) to study the algorithm theoretically to understand if we could find appropriate conditions where privileged information is guaranteed to help; (2) investigate how the algorithm performs on real-world tasks.

To summarize what we have done, we have first proved, in the case for Markov-Gaussian-linear dynamical systems, that LuPTS is better than ordinary least squares regression without access to privileged information, both in terms of expected parameter recovery and prediction risk. Importantly, we use an approach based on Rao-Blackwellization to study a predictive learning algorithm, which to the best of our knowledge is novel. In addition, we have performed an extensive set of experiments on both synthetic and two real-world datasets. As demonstrated throughout Chapter 5, LuPTS performs better than baseline in most cases, particularly in the low-sample regime, but we have also highlighted when and why it fails. We can conclude that the concept of learning using privileged time series is a promising and continually interesting topic to explore further.

Bibliography

[1] Choong Ho Lee and Hyung Jin Yoon. Medical big data: Promise and challenges. *Kidney Research and Clinical Practice*, 36(1):3–11, Mar 2017.

[2] Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5-6):544–557, 2009.

[3] Vladimir Vapnik, Rauf Izmailov, Alex Gammerman, and Vladimir Vovk. Journal of machine learning research. 16:2023–2049, 2015.

[4] Rico Jonschkowski, Sebastian Höfer, and Oliver Brock. Patterns for learning with side information. *CoRR*, abs/1511.06429, 2015.

[5] C. Radhakrishna Rao. *Linear statistical inference and its applications*. Wiley series in probability and mathematical statistics. Wiley, 2nd edition, 1973.

[6] David Blackwell. Conditional Expectation and Unbiased Sequential Estimation. *The Annals of Mathematical Statistics*, 18(1):105 – 110, 1947.

[7] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, Jul 2019.

[8] Roman Visotsky, Yuval Atzmon, and Gal Chechik. Learning with per-sample side information. In Patrick Hammer, Pulin Agrawal, Ben Goertzel, and Matthew Iklé, editors, *Artificial General Intelligence*, pages 209–219, Cham, 2019. Springer International Publishing.

[9] Fengyi Tang, Cao Xiao, Fei Wang, Jiayu Zhou, and Li-wei H. Lehman. Retaining privileged information for multi-task learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '19, page 1369–1377, New York, NY, USA, 2019. Association for Computing Machinery.

[10] Antti Sorjamaa, Jin Hao, Nima Reyhani, Yongnan Ji, and Amaury Lendasse. Methodology for long-term prediction of time series. *Neurocomputing*, 70(16-18):2861–2869, 2007.

[11] Minh Nguyen, Tong He, Lijun An, Daniel C. Alexander, Jiashi Feng, and B.T. Thomas Yeo. Predicting alzheimer's disease progression using deep recurrent neural networks. *NeuroImage*, 222:117203, 2020.

[12] Shogo Hayashi, Akira Tanimoto, and Hisashi Kashima. Long-term prediction of small time-series data using generalized distillation. *Transactions of the Japanese Society for Artificial Intelligence*, 35(5), 2020.

[13] Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, July 1991.

[14] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[15] Jonas Peters, Dominik Janzing, and Bernhard Schlkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. The MIT Press, 2017.

[16] Kevin Patrick Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[17] John A. Rice. *Mathematical Statistics and Data Analysis*. Belmont, CA: Duxbury Press., 3rd edition, 2006.

[18] Jerome Friedman Trevor Hastie, Robert Tibshirani. *The Elements of Statistical Learning*. Springer, New York, NY, 2016.

[19] Lennart Ljung. *System Identification*. Birkhäuser Boston, 1998.

[20] George Casella and Christian P. Robert. Rao-blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

[21] Fredrik Lindsten. *Rao-Blackwellised particle methods for inference and identification*. PhD thesis, Jun 2011.

[22] Daniel Daly-Grafstein and Luke Bornn. Rao-blackwellizing field goal percentage. *Journal of Quantitative Analysis in Sports*, 15(2):85–95, 2019.

[23] Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I. Jordan, Joseph E. Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning, 2018.

[24] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-

learn: Machine learning in python. *J. Mach. Learn. Res.*, 12(null):2825–2830, November 2011.

[25] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[26] Max Simchowitz, Horia Mania, Stephen Tu, Michael I. Jordan, and Benjamin Recht. Learning without mixing: Towards A sharp analysis of linear system identification. *CoRR*, abs/1802.08334, 2018.

[27] Joseph Gaugler, Tricia Johnson, Jessica Reimer, and Jennifer Weuve. 2020 alzheimer's disease facts and figures. *Alzheimer's & Dementia*, 16(3):391–460, 2020.

[28] The Alzheimer's Disease Prediction Of Longitudinal Evolution (TADPOLE) challenge. URL: `https://tadpole.grand-challenge.org/Home/`. Last time accessed: 2021-04-08.

[29] Shuxiao Wang and Jiming Hao. Air quality management in china: Issues, challenges, and options. *Journal of Environmental Sciences*, 24(1):2–13, 2012.

[30] Frank J. Kelly, Gary W. Fuller, Heather A. Walton, and Julia C. Fussell. Monitoring air pollution: Use of early warning systems for public health. *Respirology*, 17(1):7–19, 2012.

[31] Xuan Liang, Shuo Li, Shuyi Zhang, Hui Huang, and Song Xi Chen. Pm2.5 data reliability, consistency, and air quality assessment in five chinese cities. *Journal of Geophysical Research: Atmospheres*, 121(17):10,220–10,236, 2016.

[32] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[33] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[34] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning.*

From theory to algorithms. Jan 2013.

[35] M. A. Hernán and J. M. Robins. Instruments for causal inference: an epidemiologist's dream? *Epidemiology*, 17(4):360–372, Jul 2006.

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

[37] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

A

Proofs of theorems and lemmas

A.1 Proof of lemma 3

Lemma 3. *Given the system described in Section 3.1, we have that,*

$$Y = (\prod_{k=t+1}^{T} A_k\beta)^\top X_t + \tilde{\varepsilon}_t,\ t = 1, \dots, T$$

where

$$\tilde{\varepsilon}_t = \beta^\top C(t) + \varepsilon_Y$$

and

$$C(t) = \begin{cases} 0, & t = T. \\ \sum_{j=t+1}^{T} \left[\prod_{k=j+1}^{T} A_k\right]^\top \varepsilon_j, & 1 \leq t < T. \end{cases}, \quad t = 1, \dots, T$$

Proof. The statement follows from the structural equations in 3.1.

$$\begin{aligned} Y &= \beta^\top X_T + \varepsilon_Y \\ &= \beta^\top (A_T^\top X_{T-1} + \varepsilon_X) + \varepsilon_Y \\ &= \dots = \\ &= (\Big[\prod_{k=t+1}^{T} A_k\Big]^\top X_t + \beta^\top \underbrace{(\varepsilon_T + A_T^\top \varepsilon_{T-1} + \dots + (A_{t+2} \dots A_T)^\top \varepsilon_{t+1})}_{C(t)} + \varepsilon_Y \\ &= \Big[\prod_{k=t+1}^{T} A_k\beta\Big]^\top X_t + \beta^\top C(t) + \varepsilon_Y \\ &= \Big[\prod_{k=t+1}^{T} A_k\beta\Big]^\top X_t + \tilde{\varepsilon}_t \end{aligned}$$

■

A.2 Proof of lemma 6

Lemma 6. *Let* $D = (\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_T, \mathbf{Y})$ *be a random dataset generated from the Markov-Gaussian-linear system described in Section 3.1 and* $K = (\hat{A}_2, \dots, \hat{A}_T, \hat{\beta})$

be the output from Algorithm 1 (non-stationarity). Then, for any $t = 2, \ldots, T$, we have that

$$\mathbb{E}[\mathbf{X}_t|\mathbf{X}_{t-1}, K] = \mathbf{X}_{t-1}\hat{A}_t \quad \textit{and} \quad \mathbb{E}[\mathbf{Y}|\mathbf{X}_T, K] = \mathbf{X}_T\hat{\beta} \ .$$

Proof. We will first talk about how to show $\mathbb{E}[\mathbf{X}_t|\mathbf{X}_{t-1}, K] = \mathbf{X}_{t-1}\hat{A}_t$ and then explain how the same arguments are applied to proving $\mathbb{E}[\mathbf{Y}|\mathbf{X}_T, K] = \mathbf{X}_T\hat{\beta}$.

Let $\mathbf{R}_t = \mathbf{X}_t - \mathbf{X}_{t-1}\hat{A}_t$ be the residual of the OLS estimate $\hat{A}_t$. We will show that for all R_t, we have that $p(\mathbf{X}_t = \mathbf{X}_{t-1}\hat{A}_t + \mathbf{R}_t|\mathbf{X}_{t-1}, K) = p(\mathbf{X}'_t = \mathbf{X}_{t-1}\hat{A}_t - \mathbf{R}_t|\mathbf{X}_{t-1}, K)$, which implies the statement in the lemma if we assume isotropic Gaussian noise.

To show this, we first use Bayes formula:

$$\begin{aligned} p(\mathbf{X}_t|\mathbf{X}_{t-1}, K) &= \frac{p(K|\mathbf{X}_t, \mathbf{X}_{t-1})p(\mathbf{X}_t|\mathbf{X}_{t-1})}{p(K|\mathbf{X}_{t-1})} \\ &= \frac{p(\hat{\beta}, \hat{A}_T, \ldots, \hat{A}_{t+1}|\mathbf{X}_t)p(\hat{A}_t|\mathbf{X}_t, \mathbf{X}_{t-1})p(\hat{A}_1, \ldots, \hat{A}_{t-1}|\mathbf{X}_{t-1})p(\mathbf{X}_t|\mathbf{X}_{t-1})}{p(K|\mathbf{X}_{t-1})} \end{aligned}$$

where we have used the Markov property in the second equality which implies the following statements:

$$\begin{aligned} \hat{A}_1, \ldots, \hat{A}_{t-1} &\perp\!\!\!\perp \mathbf{X}_t|\mathbf{X}_{t-1} \\ \hat{A}_{t+1}, \ldots, \hat{A}_T, \hat{\beta} &\perp\!\!\!\perp \mathbf{X}_{t-1}|\mathbf{X}_t \\ \hat{A}_t \perp\!\!\!\perp \hat{A}_1, \ldots, \hat{A}_{t-1}, \hat{A}_{t+1}, \ldots, \hat{A}_T\hat{\beta}|\mathbf{X}_t, \mathbf{X}_{t-1}. \end{aligned}$$

For $p(\mathbf{X}_t|\mathbf{X}_{t-1}, K) = p(\mathbf{X}'_t|\mathbf{X}_{t-1}, K)$ to hold, we look at the factors that depend on $\mathbf{X}_t$. This tells us that we need to prove the following three statements:

(a) $p(\mathbf{X}_t|\mathbf{X}_{t-1}) = p(\mathbf{X}'_t|\mathbf{X}_{t-1})$

(b) $p(\hat{A}_t|\mathbf{X}_t, \mathbf{X}_{t-1}) = p(\hat{A}_t|\mathbf{X}'_t, \mathbf{X}_{t-1})$

(c) $p(\hat{\beta}, \hat{A}_T, \ldots, \hat{A}_{t+1}|\mathbf{X}_t) = p(\hat{\beta}, \hat{A}_T, \ldots, \hat{A}_{t+1}|\mathbf{X}'_t)$

We will now prove each of these statements:

Statement (a): We first define $\boldsymbol{\varepsilon}'_t = \boldsymbol{\varepsilon} - 2R_t$ where we have that $\mathbf{X}_t = \mathbf{X}_{t-1}\hat{A}_t + \boldsymbol{\varepsilon}$. Then, note that $\mathbf{X}_{t-1}A_t + \boldsymbol{\varepsilon}'_t = \mathbf{X}_{t-1}\hat{A}_t - R_t = \mathbf{X}'_t$ which means that showing (a) equates to showing that $p(\boldsymbol{\varepsilon}) = p(\boldsymbol{\varepsilon}'_t)$, since the noise is independent of $\mathbf{X}_{t-1}$. For Gaussian noise, these probabilities are determined by the inner product of the noise, hence it is sufficient to prove

$$\boldsymbol{\varepsilon}^\top\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}'^\top\boldsymbol{\varepsilon}'$$

We have that

$$\boldsymbol{\varepsilon}'^\top\boldsymbol{\varepsilon}' = \boldsymbol{\varepsilon}^\top\boldsymbol{\varepsilon} - 4\boldsymbol{\varepsilon}^\top\mathbf{R}_t + 4\mathbf{R}_t^\top\mathbf{R}_t$$

and thus, we need to show

$$\mathbf{R}_t^\top(\boldsymbol{\varepsilon} - \mathbf{R}_t) = 0 \ .$$

By definition,

$$\mathbf{R}_t = \mathbf{X}_t - \mathbf{X}_{t-1}\hat{A}_t$$

and so

$$\mathbf{R}_t^\top(\boldsymbol{\varepsilon} - \mathbf{R}_t) = \mathbf{R}_t^\top(\mathbf{X}_t - \mathbf{X}_{t-1}A_t - (\mathbf{X}_t - \mathbf{X}_{t-1}\hat{A}_t)) = -\mathbf{R}_t^\top(\mathbf{X}_{t-1}(A_t - \hat{A}_t)) = 0$$

since $\mathbf{R}_t^\top\mathbf{X}_{t-1} = 0$ is a property of the OLS estimator. This proves statement (a).

Statement (b): Now, we see that

$$\begin{aligned}\hat{A}'_t &= (\mathbf{X}_{t-1}^\top\mathbf{X}_{t-1})^{-1}\mathbf{X}_{t-1}^\top\mathbf{X}'_t \\ &= (\mathbf{X}_{t-1}^\top\mathbf{X}_{t-1})^{-1}\mathbf{X}_{t-1}^\top(\mathbf{X}_t - 2\mathbf{R}_t) \\ &= \hat{A}_t - 2(\mathbf{X}_{t-1}^\top\mathbf{X}_{t-1})^{-1}\mathbf{X}_{t-1}^\top\mathbf{R}_t = \hat{A}_t \ .\end{aligned}$$

since, again, $\mathbf{X}_{t-1}^\top\mathbf{R}_t = 0$. This implies that the distribution of $\hat{A}_t$ is the same if conditioned on $\mathbf{X}_t$ or $\mathbf{X}'_t$, which proves statement (b).

Statement (c): We can factorize $p(\hat{\beta}, \hat{A}_T, \ldots, \hat{A}_{t+1}|\mathbf{X}_t)$ as

$$p(\hat{\beta}|\hat{A}_T, \ldots, \hat{A}_{t+1}, \mathbf{X}_t)p(\hat{A}_T|\hat{A}_{T-1}, \ldots, \hat{A}_{t+1}, \mathbf{X}_t)\ldots p(\hat{A}_{t+2}|\hat{A}_{t+1}, \mathbf{X}_t)p(\hat{A}_{t+1}|\mathbf{X}_t)$$

Let $C_k = (\hat{A}_k, \hat{A}_{k-1}, \ldots, \hat{A}_{t+1})$ for $k = t+1, \ldots, T$. Then, we can summarize the problem as the following: we need to show that each factor in the above equation are the same for both $\mathbf{X}_t$ and $\mathbf{X}'_t$, i.e.,

$$\begin{aligned}&p(\hat{\beta}|C_T, \mathbf{X}_t) = p(\hat{\beta}|C_T, \mathbf{X}'_t) \\ &p(\hat{A}_k|C_{k-1}, \mathbf{X}_t) = p(\hat{A}_k|C_{k-1}, \mathbf{X}'_t), \quad k = t+2, \ldots, T \\ &p(\hat{A}_{t+1}|\mathbf{X}_t) = p(\hat{A}_{t+1}|\mathbf{X}'_t)\end{aligned}$$

These equations could be seen as the distributions of OLS estimators with a (conditional) random design or, for the third one, with a fixed design matrix. As we assume mean-zero and uncorrelated Gaussian noise $\varepsilon_t \sim N(0, \sigma_t^2 I)$ and $\varepsilon_Y \sim N(0, \sigma_Y^2)$, the distribution of OLS estimators (see Chapter 14 in [17]) is known as,

$$\begin{aligned}\hat{\beta}|C_T, \mathbf{X}_t \sim &N\left(\beta, \sigma_Y^2\,\mathbb{E}\left[\left(\mathbf{X}_T^\top\mathbf{X}_T\right)^{-1}|C_T, \mathbf{X}_t\right]\right) \\ \hat{A}_k^{(\text{row } i)}|C_{k-1}, \mathbf{X}_t \sim &N\left(A_k^{(\text{row } i)}, \sigma_k^2\,\mathbb{E}\left[\left(\mathbf{X}_{k-1}^\top\mathbf{X}_{k-1}\right)^{-1}|C_{k-1}, \mathbf{X}_t\right]\right), \quad k = t+2, \ldots, T \\ \hat{A}_{t+1}^{(\text{row } i)}|\mathbf{X}_t \sim &N\left(A_{t+1}^{(\text{row } i)}, \sigma_{t+1}^2\left(\mathbf{X}_t^\top\mathbf{X}_t\right)^{-1}\right)\end{aligned}$$

where $i = 1, \ldots, d$ represent the rows of $\hat{A}_k$, this corresponds to the different features which are predicted. Hence, it is sufficient to show that

$$\mathbb{E}\left[\left(\mathbf{X}_{k-1}^\top\mathbf{X}_{k-1}\right)^{-1}|C_{k-1}, \mathbf{X}_t\right] = \mathbb{E}\left[\left(\mathbf{X}_{k-1}^\top\mathbf{X}_{k-1}\right)^{-1}|C_{k-1}, \mathbf{X}'_t\right], \quad k = t+2, \ldots, T$$

and the special case where $\mathbf{X}_t^\top \mathbf{X}_t = \mathbf{X'}_t^\top \mathbf{X}'_t$. For the latter, we have

$$\begin{aligned}(\mathbf{X}_{t-1}\hat{A}_t \pm \mathbf{R}_t)^\top (\mathbf{X}_{t-1}\hat{A}_t \pm \mathbf{R}_t) = \\ &= (\mathbf{X}_{t-1}\hat{A}_t)^\top (\mathbf{X}_{t-1}\hat{A}_t) \pm 2(\mathbf{X}_{t-1}\hat{A}_t)^\top \mathbf{R}_t + \mathbf{R}_t^\top \mathbf{R}_t \\ &= (\mathbf{X}_{t-1}\hat{A}_t)^\top (\mathbf{X}_{t-1}\hat{A}_t) + \mathbf{R}_t^\top \mathbf{R}_t\end{aligned}$$

where we have used that the cross term $(\mathbf{X}_{t-1}\hat{A}_t)^\top \mathbf{R}_t = \hat{A}_t^\top \mathbf{X}_{t-1}^\top \mathbf{R}_t = 0$ because $\mathbf{X}_{t-1}^\top \mathbf{R}_t = 0$. As the cross term is the only thing which differs in $\mathbf{X}_t^\top \mathbf{X}_t$ and $\mathbf{X'}_t^\top \mathbf{X}'_t$, it implies that they must be equal.

For $\mathbb{E}\left[\left(\mathbf{X}_{k-1}^\top \mathbf{X}_{k-1}\right)^{-1} | C_{k-1}, \mathbf{X}_t\right]$ with $k = t+2, \ldots, T$ we use the same expression as before but observe the following recursive relationship between the inner product of $\mathbf{X}_k$ and $\mathbf{X}_{k-1}$:

$$\mathbf{X}_k^\top \mathbf{X}_k = (\mathbf{X}_{k-1}\hat{A}_k)^\top \mathbf{X}_{k-1}\hat{A}_k + \mathbf{R}_k^\top \mathbf{R}_k = \hat{A}_k^\top \underbrace{\mathbf{X}_{k-1}^\top \mathbf{X}_{k-1}}_{\text{Inner product}} \hat{A}_k + \mathbf{R}_k^\top \mathbf{R}_k$$

Hence, we get that

$$\mathbf{X}_{k-1}^\top \mathbf{X}_{k-1} = \prod_{i=t+1}^{k-1} \hat{A}_i^\top (\mathbf{X}_t^\top \mathbf{X}_t) \prod_{i=t+1}^{k-1} \hat{A}_i + \sum_{j=t+1}^{k-2} \mathbf{R}_j^\top \mathbf{R}_j \prod_{i=j+1}^{k-1} \hat{A}_i + \mathbf{R}_{k-1}^\top \mathbf{R}_{k-1}$$

We see that $\mathbf{X}_{k-1}^\top \mathbf{X}_{k-1}$ is directly dependent on $\mathbf{X}_t^\top \mathbf{X}_t$ where the residuals and OLS estimators are fixed as we condition upon them. Since we already have shown that $\mathbf{X}_t^\top \mathbf{X}_t = \mathbf{X'}_t^\top \mathbf{X}'_t$, that means that

$$\mathbb{E}\left[\left(\mathbf{X}_{k-1}^\top \mathbf{X}_{k-1}\right)^{-1} | C_{k-1}, \mathbf{X}_t\right] = \mathbb{E}\left[\left(\mathbf{X}_{k-1}^\top \mathbf{X}_{k-1}\right)^{-1} | C_{k-1}, \mathbf{X}'_t\right]$$

for $k = t+2, \ldots, T$. This proves statement (c).

Since we have proven all three statements that were presented in the beginning of this proof, we have shown that $p(\mathbf{X}_t = \mathbf{X}_{t-1}\hat{A}_t + \mathbf{R}_t | \mathbf{X}_{t-1}, K) = p(\mathbf{X}'_t = \mathbf{X}_{t-1}\hat{A}_t - \mathbf{R}_t | \mathbf{X}_{t-1}, K)$.

Finally, to show that $\mathbb{E}[\mathbf{Y}|\mathbf{X}_T, K] = \mathbf{X}_T\hat{\beta}$, we can use the same arguments as before to show $p(\mathbf{Y} = \mathbf{X}_T\hat{\beta} + \mathbf{R}_Y | \mathbf{X}_T, K) = p(\mathbf{Y'} = \mathbf{X}_T\hat{\beta} - \mathbf{R}_Y | \mathbf{X}_T, K)$ although only statements (a) and (b) are necessary for this case. ■

Remark on anisotropic noise For the anisotropic case, the analysis becomes slightly different. The noise in the data $\mathbf{X}_t$ is $\boldsymbol{\varepsilon}_t = [\varepsilon_{t,1}, \ldots \varepsilon_{t,n}]^\top \in \mathbb{R}^{n \times d}$. The rows corresponds the the noise in a particular sample, while the columns are for the different features. Furthermore, the covariance of the ith feature is $\mathrm{Cov}(\boldsymbol{\varepsilon}_t^{(\text{column } i)}) = \sigma_{t,i}^2 I_n$ for $i = 1, \ldots, d$ where I_n is the n-dimensional identity matrix. With anisotropic

noise, we have $\sigma_{t,i} \neq \sigma_{t,i'}$ for $i, i' = 1, \ldots, d$. Then, the above lemma will be feasible using a similar analysis as we can show that,

$$\hat{A}_k^{(\text{row } i)} | C_{k-1}, \mathbf{X}_t \sim \mathrm{N}\left(A_k^{(\text{row } i)}, \sigma_{k,i}^2 \ \mathbb{E}\left[\left(\mathbf{X}_{k-1}^\top \mathbf{X}_{k-1}\right)^{-1} \mid C_{k-1}, \mathbf{X}_t\right]\right) \quad k = t{+}1, \ldots, T$$

Then, the analysis follows as in Lemma 6.

B

Additional experimental results

B.1 Alzheimer's disease progression modeling

The dataset used for the experiments on Alzheimer's disease progression was the ADNIMERGE set from the Alzheimer's Disease Neuroimaging Initiative (ADNI). All the features used for the experiments are found in Table B.1. Note that it is the tags used in the dataset that are listed. In Table B.2 and B.3 are the results from the experiments presented in Section 5.2.1 listed numerically, rounded to two decimal points.

Table B.1: Features used for the ADNI experiments

Feature tags		
AGE	PTGENDER	PTEDUCAT
APOE4	FDG	AV45
ABETA	TAU	PTAU
CDRSB	ADAS11	ADAS13
ADASQ4	MMSE	RAVLT_immediate
RAVLT_learning	RAVLT_forgetting	RAVLT_perc_forgetting
LDELTOTAL	TRABSCOR	FAQ
MOCA	EcogPtMem	EcogPtLang
EcogPtVisspat	EcogPtPlan	EcogPtOrgan
EcogPtDivatt	EcogPtTotal	EcogSPMem
EcogSPLang	EcogSPVisspat	EcogSPPlan
EcogSPOrgan	EcogSPDivatt	EcogSPTotal
Ventricles	Hippocampus	WholeBrain
Entorhinal	Fusiform	MidTemp
ICV		

Table B.2: MMSE prediction experiment results, average R^2 score with one standard deviation in parenthesis from 100 iterations. **Left**: One privileged time point used. **Right**: three privileged time points used.

Samples	Baseline	LuPTS
80	-0.02 (0.34)	0.14 (0.24)
100	0.12 (0.29)	0.27 (0.17)
120	0.27 (0.14)	0.36 (0.1)
140	0.36 (0.08)	0.42 (0.06)
160	0.39 (0.08)	0.44 (0.05)
180	0.42 (0.06)	0.46 (0.05)
200	0.44 (0.05)	0.48 (0.04)
220	0.45 (0.05)	0.49 (0.04)

Samples	Baseline	LuPTS	Stat-LuPTS
80	-0.02 (0.34)	0.25 (0.16)	0.39 (0.09)
100	0.12 (0.29)	0.34 (0.11)	0.43 (0.06)
120	0.27 (0.14)	0.41 (0.06)	0.46 (0.05)
140	0.36 (0.08)	0.44 (0.05)	0.47 (0.04)
160	0.39 (0.08)	0.46 (0.04)	0.47 (0.04)
180	0.42 (0.06)	0.48 (0.05)	0.49 (0.04)
200	0.44 (0.05)	0.48 (0.04)	0.49 (0.03)
220	0.45 (0.05)	0.49 (0.03)	0.5 (0.03)

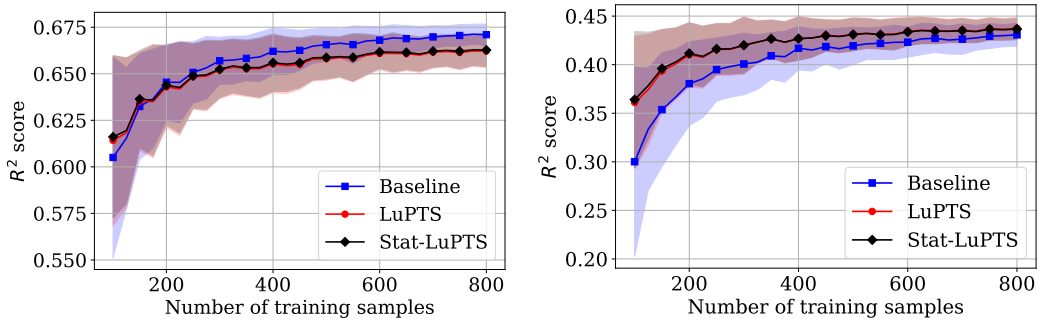
Table B.3: AD prediction experiment results, average AUC with one standard deviation in parenthesis from 100 iterations **Left**: one privileged time point used. **Right**: three privileged time points used.

Samples	Baseline	LuPTS
80	0.86 (0.04)	0.85 (0.04)
100	0.86 (0.06)	0.87 (0.04)
120	0.88 (0.03)	0.9 (0.03)
140	0.89 (0.03)	0.9 (0.02)
160	0.9 (0.02)	0.91 (0.02)
180	0.9 (0.02)	0.92 (0.02)
200	0.91 (0.02)	0.92 (0.02)
220	0.91 (0.02)	0.92 (0.02)

Samples	Baseline	LuPTS	Stat-LuPTS
80	0.86 (0.04)	0.87 (0.04)	0.9 (0.03)
100	0.86 (0.06)	0.89 (0.03)	0.91 (0.02)
120	0.88 (0.03)	0.91 (0.02)	0.92 (0.02)
140	0.89 (0.03)	0.91 (0.02)	0.92 (0.02)
160	0.9 (0.02)	0.92 (0.02)	0.93 (0.02)
180	0.9 (0.02)	0.92 (0.02)	0.93 (0.02)
200	0.91 (0.02)	0.93 (0.02)	0.93 (0.02)
220	0.91 (0.02)	0.93 (0.02)	0.93 (0.02)

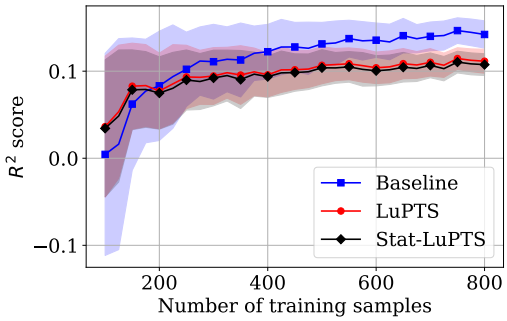
B.2 Forecasting air quality in Chinese cities

In Section 5.2.2, we showed the experiments with different time horizons for Shanghai and Shenyang. The same experiments for the other cities are provided here, see Figure B.1, B.2 and B.3.



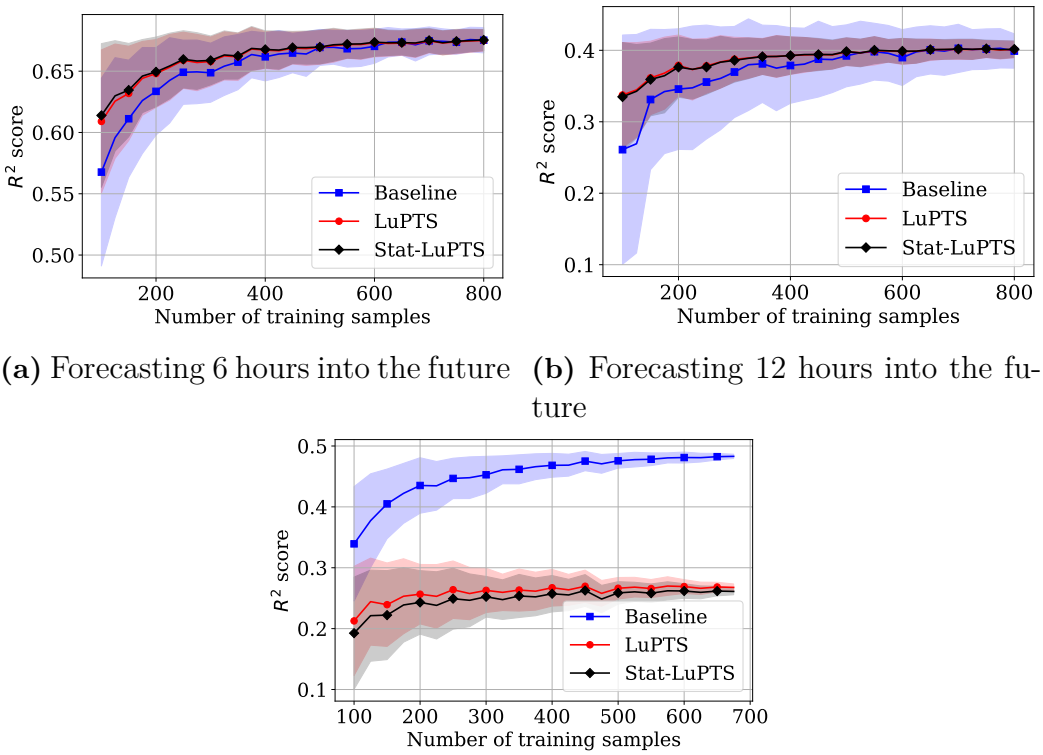
(a) Forecasting 6 hours into the future

(b) Forecasting 12 hours into the future



(c) Forecasting 24 hours into the future

Figure B.1: Beijing: Changing the time horizon to predict the $PM_{2.5}$ concentration levels. R^2 used as metric (higher is better); shaded region indicates one standard deviation across 200 iterations.

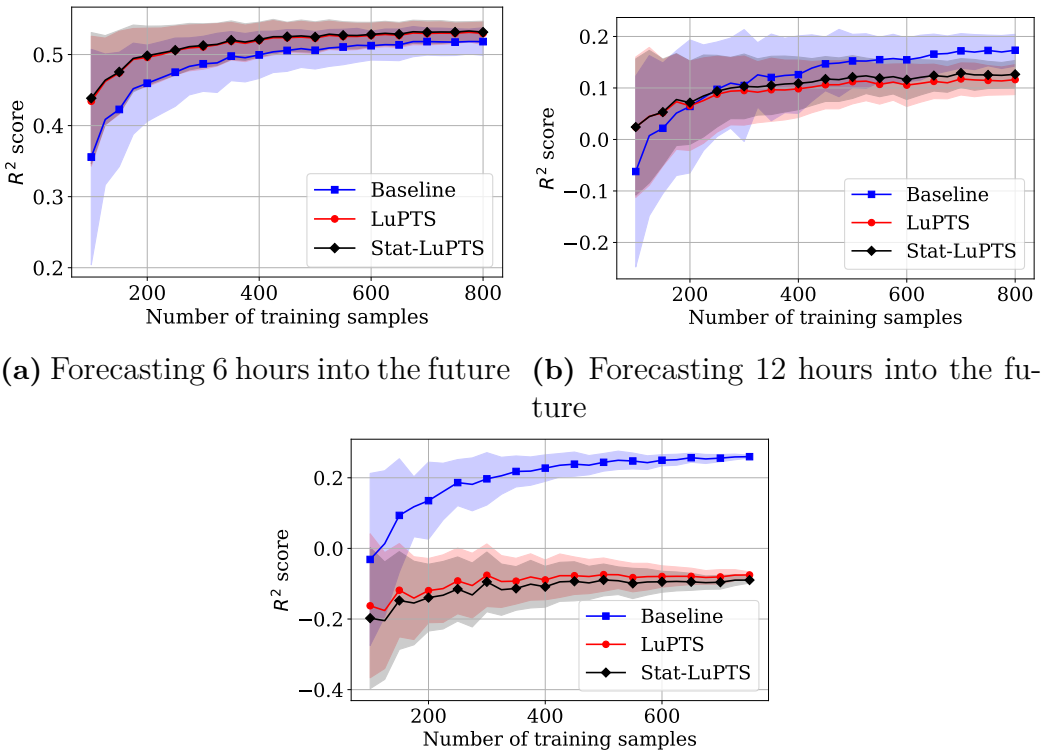


(a) Forecasting 6 hours into the future

(b) Forecasting 12 hours into the future

(c) Forecasting 24 hours into the future

Figure B.2: Chengdu: Changing the time horizon to predict the $PM_{2.5}$ concentration levels. R^2 used as metric (higher is better); shaded region indicates one standard deviation across 200 iterations.



(a) Forecasting 6 hours into the future

(b) Forecasting 12 hours into the future

(c) Forecasting 24 hours into the future

Figure B.3: Guangzhou: Changing the time horizon to predict the $PM_{2.5}$ concentration levels. R^2 used as metric (higher is better); shaded region indicates one standard deviation across 200 iterations.