Embedding Kalman Filters into Reproducing Kernel Hilbert Spaces

Master-Thesis von Gregor H. W. Gebhardt
September 2014

TECHNISCHE UNIVERSITÄT DARMSTADT

Embedding Kalman Filters into Reproducing Kernel Hilbert Spaces

Vorgelegte Master-Thesis von Gregor H. W. Gebhardt

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Prof. Dr. techn. Gerhard Neumann

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 19. September 2014

(Gregor H. W. Gebhardt)

Abstract

Providing adequate models for time series of stochastic semi-observable processes is substantial for solving a wide variety of problems in machine learning. Most of the data that is observed, e.g., from robotic systems, does not depict the true states but rather noisy variates of them. Additionally, the observed state space is generally only a subset of the true state space, as the sensory equipment of most systems is limited. Hidden Markov models (HMMs) are a widely used probabilistic graphical model for time series of discrete, partially observable stochastic processes. They make the basic assumptions that a fixed number of preceding hidden states suffices to reason about the current hidden state (Markovian property) and that a certain observation is conditionally dependent solely on its corresponding hidden state. A well established concept that extends the ideas of HMMs to continuous domains is the Kalman filter (KF), which assumes linear system dynamics and represents the state as a Gaussian random variable. At its core, the Kalman filter evolves its belief over the state by advancing it according to the system dynamics (transition model) and by conditioning on new observations (observation model). In this thesis, we present an approach that extends the application of Kalman filters to non-linear dynamical systems by embedding the belief about the state into a reproducing kernel Hilbert space (RKHS). We further provide the Hilbert space analogs to the transition and the observation model. We evaluate our model on artificial data obtained from a simulated pendulum and compare the one-step and long-term prediction performances for different training setups, different initialization methods and different performance measures for the optimization of the model parameters. We further demonstrate the ability of our model to operate on high dimensional data sets by predicting artificial video frames. Finally, we compare our model to the recently presented kernel spectral algorithm (Song et al., 2010) and show that our approach outperforms the kernel spectral methods by means of generalization to previously unseen data.

Acknowledgments

First and foremost, I would like to thank Prof. Dr. Jan Peters and Prof. Dr. Gerhard Neumann for providing me the great opportunity to write my Master's thesis at the Intelligent Autonomous Systems Group in the most fascinating area of research. I am very grateful for the excellent supervision by Gerhard Neumann, who provided me with help, advice and inspiration in numerous discussions. With his extensive knowledge and his bright ideas, he always knew how to tackle even the most severe problems I was facing while working on this thesis. I would like to thank further all the members of IAS, with whom it was a great pleasure to work. They were always very kind and supportive when addressing them with all kinds of questions and problems. Finally, yet not less importantly, I would like to express my gratitude towards my parents, without whose enduring support my studies would not have been possible.

Contents

Figures, Tables and Algorithms

List of Figures

List of Tables

List of Algorithms

1 Introduction

The prediction of future states, filtering and smoothing of observed states and the estimation of latent states are an important step to solve a wide variety of problems in machine learning and robotics. Examples for these tasks are manifold: Learning forward and inverse models of the kinematics or dynamics of robotic systems, where it is not feasible to derive analytic solutions; estimation of missing quantities from incomplete sensory data, e.g., of robots equipped only with sensors measuring the joint states but not the joint velocities or torques; or smoothing noisy data from mediocre sensors, to name only a few.

The most prevalent probabilistic method to model the time series emerging from such partially observable stochastic processes is the hidden Markov model (HMM). They have been used on a wide range of applications such as recognizing human actions and gestures for imitation learning, speech processing (Rabiner, 1989) or the analysis of human genomes (Haussler and Eeckman, 1996). A hidden Markov model is a directed probabilistic graphical model that assumes a Markov chain of latent states, i.e., the conditional probability of a latent state is only dependent on a fixed number of predecessors. Though, instead of directly observing the latent states, we can only obtain partial, noisy observations. These observations are emitted from the stochastic process by a given observation probability that depends only on the corresponding latent state.

Hidden Markov models, however, have the limitation that they are typically defined for discrete latent states and discrete or Gaussian distributed observations. Moreover, the latent states are in most cases—due to their nature of being hidden—not available for learning the conditional dependencies of an HMM. Typically, such incomplete data requires resorting to expectation maximization (EM) methods, which are time consuming iterative local methods that are not guaranteed to find the optimal solution.

The spectral learning algorithm (Hsu et al., 2012) approaches the above stated learning problem under incomplete data by providing a one-shot learning method for discrete hidden Markov models. Song et al. (2010) extend the applicability of the spectral algorithm to continuous observations by embedding it into reproducing kernel Hilbert spaces (RKHS). However, both of these approaches have the shortcoming that they inherently assume discrete hidden states. This inherent assumption consequently leads to bad generalizations to hidden states which were not present in the training data. In addition, the spectral methods allow only to compute maximum-a-posteriori estimates of future and current observations, but do not provide proper probability distributions in state space.

In this work, we will present a new approach to model the time series of partially observable processes that have a continuous domain in both the observations as well as the hidden states. Our work is highly related to Kalman filters (Kalman, 1960): A state in our model is a data window that covers the current and an arbitrary number of past and future system states; We express the belief about the current data window as a Gaussian random variable (GRV) with mean and covariance; And we further assume linear system dynamics and linear relations between hidden states and observations, which we will learn from sample data.

The linear assumption is yet a strong constraint on the range of systems for which our model yields a sufficient approximation. Therefore, we embed the probability distributions of our model in a *reproducing kernel Hilbert space* (RKHS), a concept that only recently emerged (Smola et al., 2007; Sriperumbudur et al., 2008) in the area of machine learning. The basic idea behind this concept is to represent probability density functions as points in possibly infinite dimensional Hilbert spaces. A framework of linear Hilbert space operators and rules (Song et al., 2013) allows to manipulate these densities entirely in Hilbert space. Computationally, the embeddings are represented using a basis of data samples and the "kernel trick" allows to express the Hilbert space operators in terms of Gram matrices.

In contrast to most approaches that employ RKHS embeddings, we embed the distribution over a data window using both an embedding of the mean and of the covariance. By doing so, we constrain the embeddings to Gaussian distributions, which is a cutback to the diversity of probability densities that can be expressed by Hilbert space embeddings. Nevertheless, it allows us to reconstruct the fully probabilistic belief of the data window in state space. For this reconstructions we assume a linear operator from Hilbert space to state space that can be applied to the mean embedding and the covariance operator alike. The ability to project from Hilbert space into state space allows us further to condition on observations using the standard equations for conditional multivariate Gaussian distributions.

We evaluated our approach on artificial data from a simulated pendulum, where we compared the performances for different configurations of our model. We scrutinized different heuristics for the optimization of the model parameters and different initialization methods and compared the impact of noisy training data on the results to models trained with smooth data. We further evaluated the long-term prediction performances of our model by providing only a certain number of observations and predict the succeeding states by iteratively applying solely the transition model. Finally, we demonstrate the scalability of our model to high dimensional data and the ability to generalize for previously unseen data by making predictions on video frames rendered from the movements of a simulated pendulum. In both experiments we compare our model to the kernel spectral algorithm (Song et al., 2010).

1.1 Related Work

As we already noted previously, our method is highly inspired by the Kalman filter. We use a similar representation of state as Gaussian random variable and similar update methods to advance our belief over time according to the system dynamics and to update it when new observations are available.

Two widely known and applied approaches to extend the Kalman filter to nonlinear systems are the *extended Kalman filter* (EKF) and the *unscented Kalman filter* (UKF) (Wan and Van Der Merwe, 2000; Julier and Uhlmann, 1997). Equally to the standard Kalman filter, both treat the system states as Gaussian random variables (GRV) and—in contrast to our method—assume that the non-linear system functions are known. As a GRV, the system state is expressed sufficiently by a mean and a covariance matrix. Both the EKF and the UKF update the mean by applying the nonlinear system functions, which is, however, not possible for the covariance matrix. The extended Kalman filter solves this problem by linearizing the system functions at the current state. Though, this technique can lead to poor performance if the nonlinear system dynamics are not close to linear within the range of an update step. In contrast, the unscented Kalman filter describes the covariance by a set of sample-points (sigma points) and transforms each of these points individually using the non-linear system functions.

As an effective and fast method to learn hidden Markov models, Hsu et al. (2012) recently proposed the spectral learning algorithm, a one-shot learning method that avoids the tedious iterative search of the Baum-Welch algorithm. At its core, the algorithm adopts from clustering methods (Vempala and Wang, 2002) by presuming well separated distributions over the observations for each hidden state. They exploit the spectral properties of observable measures to derive an observable representation of the hidden Markov model. This observable representation is based on the observable operator model (Jaeger, 2000), which allows to approximate a joint distribution over the observations as well as a conditional distribution over future states without any notion of the hidden states.

To extend its applicability to time series of a continuous domain, the spectral learning algorithm was later embedded into reproducing kernel Hilbert spaces (Song et al., 2010). However, the algorithm still assumes the probability densities over the observations to be separable in the RKHS, which is a far more restrictive assumption for continuous valued time series than for discrete observations and hidden states. In contrast to our model, they update the state estimates entirely in the RKHS and don't provide a mapping back into the state space. Instead, they search for a sample that maximizes the a-posteriori belief within their training data set. We will describe the spectral learning algorithm in Section 2.3 and the RKHS-embedded spectral learning method in Section 3.2.

An approach to predict the state of *controlled* stochastic processes are the predictive state representations (PSRs) (Littman et al., 2002). A PSR is a set of state-action sequences (tests) that is sufficient to describe the success probability of any test sequence at the time of the PSR. Here, a test is considered successful if the execution of the actions defined by the test leads to the corresponding observations. The PSRs were initially defined on discrete actions and observations, but were only recently embedded into reproducing kernel Hilbert spaces (Boots et al., 2013) to extend their applicability to continuous domains.

Another technique for solving the problem of predicting the state of a controlled stochastic process is employed by the policy search method PILCO (probabilistic inference for learning control) (Deisenroth et al., 2014). A crucial step of policy search methods is the time consuming execution of the intermediate policy on the system to receive an estimate about how well the policy performs. PILCO alleviates the number of tedious real-world policy evaluations by optimizing the policy on simulated trajectories of the system. For the simulation they propagate a probabilistic state estimate through a learned Gaussian process (GP) model of the system dynamics. They provide two methods for this propagation, similar to the EKF and UKF, respectively. The first one uses a linearization of the Gaussian process around the current state of the system whereas the latter uses characteristic points of the state estimate that are propagated through the Gaussian process individually.

Finally, a promising concept that may be employed for modeling incomplete time series data is the recently emerging *reservoir computing* (Lukoševičius and Jaeger, 2009), a new approach to facilitate the learning of *recurrent neural networks* (RNN, a variety of neural networks that allows cyclic connections). The key idea behind the reservoir computing concept is to initialize an RNN with randomly connected nodes (*dynamic reservoir*) and train only a mapping from all nodes of the network to an output layer. Additionally, a feedback of the output into the RNN may be established. Due to its recurrent nature and the possible output feedback, the state of the reservoir is not only dependent on the input, but also memorizes the history of in- and outputs as an implicit internal state. A prominent example for reservoir computing are the *Echo State Networks* (Jaeger, 2001), which use neurons with sigmoid activation functions and a linear mapping from the network to the output. Another important example are the *Liquid State Machines* (Maass et al., 2002), which are inspired by the insights of biology in that they use concepts to mimic the mechanics found in real neurons.

1.2 Overview

In Section 2, after defining some notational conventions, we will present the standard hidden Markov models in more depth and sketch the spectral algorithm for HMMs (Hsu et al., 2012). In Section 3, we will introduce reproducing kernel Hilbert spaces (RKHS) and describe the concept of embedding distributions into an RKHS (Song et al., 2013). We will further sketch the RKHS-embedded version of the spectral algorithm (Song et al., 2010) in Section 3.2. In Section 4, we illustrate a weight-based version of the Kalman filter before we present our RKHS-embedded approach to Kalman filtering in Section 4.3. We present experimental results in which we compare our method to the RKHS-embedded spectral algorithm in Section 5 and conclude in Section 6.

2 Preliminaries

After defining some notational conventions, we explain the hidden Markov models more precisely in Section 2.2 and point out, where traditional approaches to learning HMMs, e.g., the expectation maximization (EM) algorithm, have their shortcomings. In Section 2.3, we will sketch the spectral learning algorithm (Hsu et al., 2012) for HMMs, a recently proposed one-shot learning method, which addresses most of the drawbacks of EM.

2.1 Notation

In this work, we will make use of the following notational conventions: $x_{1:t}$ denotes a set $\{x_1, \ldots, x_t\}$ of t consecutive data points x_i. Capital letters in regular font weight denote random variables, e.g., X, Y, and lower case letters their instantiations, e.g., x, y. We will make the simplification, that all random variables are of the same domain Ω. A capital P denotes the distribution of a random variable, e.g., $P(X)$, and a lowercase p stands for the probability of its instantiation, e.g., $p(x)$. Bold lower case letters denote vectors, e.g., $\mathbf{x}$, and upper case bold letters represent matrices, e.g., $\mathbf{X}$. We denote the m-dimensional identity matrix as $\mathbf{I}_m$ and an all-ones vector of the same dimension as $\mathbf{1}_m$.

2.2 Hidden Markov Models

Hidden Markov models (Barber, 2012; Murphy, 2012) are a probabilistic graphical model, widely used to formulate the conditional dependencies in time series of partially observable stochastic processes. The typical graphical structure of a hidden Markov model is shown in Figure 2.1. They are based on the assumption that the probability density over a hidden state h_t is conditionally dependent only on a fixed number L of the preceding hidden states $h_{t-L:t-1}$, which is also called the Markovian property. In the following, we will assume first-order HMMs, for which the hidden state is conditionally dependent only on its immediate predecessor, i.e., $L = 1$. This dependency is expressed by the transition probability distribution $P(H_{t+1}|H_t)$. Additionally, an observation (emission) X_t that is emitted from the process is conditionally dependent only on the corresponding hidden state H_t, which is expressed by the observation probability distribution $P(X_t|H_t)$. The initial hidden state H_1 of the system from which the observations emerge is distributed according to the distribution $\pi(H_1)$. The joint probability of an observed time series $x_{1:T}$ is then given by

$$p(x_{1:T}) = \int\limits_{h_{1:T}\in\Omega^T} p(x_1|h_1)\pi(h_1)\prod_{t=2}^{T} p(x_t|h_t)p(h_t|h_{t-1})\mathrm{d}h_{1:T}. \tag{2.1}$$

Typically, hidden Markov models are defined for a discrete and finite domain for both the hidden variables as well as the emissions. Assuming the hidden domain is of cardinality m and the observable domain is of cardinality n, where usually the number of observable states is greater than or at least equal to the number of hidden states ($n \geq m$), the transition probability distribution and the observation probability distribution can be expressed as a square matrix $\mathbf{T} \in \mathbb{R}^{m\times m}$ and as a matrix $\mathbf{O} \in \mathbb{R}^{n\times m}$, respectively. An entry $\mathbf{T}_{ij}$ of the transition matrix holds the probability $p(H_t = h_i|H_{t-1} = h_j), i, j \in \{1, \ldots, m\}$ of transitioning from the hidden state h_j to h_i. Likewise, the observation matrix has as entries $\mathbf{O}_{ij} = p(X_t = x_i|H_t = h_j), i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}$ the probability of observing the emission x_i given the hidden state h_j.

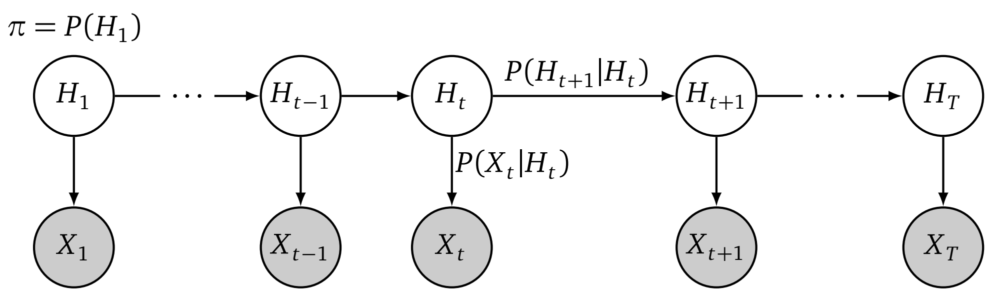


Figure 2.1.: Graphical structure of a first-order hidden Markov model. White nodes represent the latent states h_i that are internal to the underlying system. Shaded nodes represent the observations x_i that are emitted by the system.

However, in most cases, the dynamics of the underlying system and, hence, the transition and observation matrices are not known. Also, the hidden states are usually—due to their nature of being hidden—not available to the learner and, thus, the transition and observation matrices cannot be inferred directly from data. A common practice is to apply the Baum-Welch algorithm, an expectation maximization (EM) method, to iteratively compute a belief over the hidden states and, based on that, derive a maximum-likelihood estimate of the transition matrix $\mathbf{T}$, of the observation matrix $\mathbf{O}$ and of the distribution π over the initial hidden states. Nevertheless, the Baum-Welch algorithm, like other EM methods, is a time consuming iterative algorithm that relies on a good initialization to find the optimal or a near-optimal solution. Moreover, it requires the knowledge of the number of hidden states, which is not always available.

2.3 The Spectral Learning Method for Hidden Markov Models

To overcome the problems connected to the EM algorithm, Hsu et al. (2012) proposed the spectral learning algorithm for HMMs. They employ recently developed methods for clustering (Vempala and Wang, 2002) that assume distinct (i.e., separable) distributions for each cluster. Hsu et al. (2012) adopt this concept for hidden Markov models by requiring distinct distributions of the observable quantity for each of the discrete hidden states.

They base their algorithm upon the observable operator model (OOM) (Jaeger, 2000), an approach that allows to model an HMM entirely in observation space and, hence, renders the explicit modeling of the hidden states unnecessary. They define the observable operator as

$$\mathbf{A}_x = \mathbf{T}\,\mathrm{diag}(\mathbf{O}_{x1}, \ldots, \mathbf{O}_{xm}), \tag{2.2}$$

with the observation x, the transition matrix $\mathbf{T}$ and a diagonal matrix consisting of the x-th row the observation matrix $\mathbf{O}$. Thus, the observable operator combines the transitional probabilities of the hidden state with the probabilities of the current observation into a joint probability over the hidden state and the observation. Consequently, the joint probability distribution of an observed time series (c.f. Equation (2.1)) can be represented more compactly as

$$p(x_{1:T}) = \mathbf{1}_m^\mathsf{T} \mathbf{A}_{x_t} \ldots \mathbf{A}_{x_1} \pi = \mathbf{1}_m^\mathsf{T} \mathbf{A}_{x_{t:1}} \pi, \tag{2.3}$$

where π is the distribution of the initial hidden states $p(H_1)$ and the multiplication with the all-ones vector $\mathbf{1}_m$ serves as a marginalization over the hidden states. From this OOM, Hsu et al. (2012) derive an HMM representation that is computed solely on observable data. The following paragraphs briefly sketch the derivations of the spectral learning method for hidden Markov models.

Based on observation triples (x_1, x_2, x_3), Hsu et al. (2012) define the observable quantities

$$P_1 := P(X_1), \tag{2.4}$$
$$P_{2,1} := P(X_2, X_1), \tag{2.5}$$
$$P_{3,x,1} := P(X_3, X_2 = x, X_1), \tag{2.6}$$

where P_1 is a vector with the probability distribution over the first state x_1, $P_{2,1}$ is a matrix with the joint probability distribution of observing a tuple (x_1, x_2) and $P_{3,x,1}$ is a family of n matrices, with a single matrix for each instance of $x_2 = x$, holding the probabilities of observing a triple $(x_3, x_2 = x, x_1)$.

They further introduce a matrix $\mathbf{U} \in \mathbb{R}^{n \times m}$ which projects the observation matrix $\mathbf{O}$ into an m-dimensional subspace and obeys the condition that the projected observation matrix $\mathbf{U}^\intercal\mathbf{O}$ is invertible. Note that the assumption of $\mathbf{U}^\intercal\mathbf{O}$ being invertible implies linearly independent observation probabilities for each hidden state, which is not true for all systems. Therefore, it is often necessary to represent a state by data windows instead of single data points.

Hsu et al. (2012) show that the matrix $\mathbf{U}$ can be obtained from the singular value decomposition (SVD) of $P_{2,1}$ by taking the m left singular vectors corresponding to the m largest singular eigenvalues. They further state that m can be estimated by the number of eigenvalues that differ significantly from zero and that m corresponds to the number of distinct, i.e., well separable, hidden states.

Using the observable quantities (2.4) - (2.6) and the projection matrix $\mathbf{U}$, they define an observable representation of an HMM by

$$\mathbf{b}_1 := \mathbf{U}^\intercal P_1, \tag{2.7}$$
$$\mathbf{b}_\infty := \left(P_{2,1}^\intercal \mathbf{U}\right)^\dagger, \tag{2.8}$$
$$\mathbf{B}_x := \left(\mathbf{U}^\intercal P_{3,x,1}\right)\left(\mathbf{U}^\intercal P_{2,1}\right)^\dagger, \tag{2.9}$$

where † denotes the Moore-Penrose pseudo-inverse of a matrix. It can be shown that the observable representation in (2.7)-(2.9) suffices to compute not only the joint probability of a sequence of observations (Hsu et al., 2012; Hesse, 2014)

$$p(x_1, \ldots, x_T) = \mathbf{b}_\infty^\intercal \mathbf{B}_{x_t} \ldots \mathbf{B}_{x_1} \mathbf{b}_1, \tag{2.10}$$

but also to compute the conditional probability of future observations (i.e., predicting future emissions) as

$$p(x_{t+1} | x_{1:t}) = \frac{\mathbf{b}_\infty^\intercal \mathbf{B}_{x_{t+1}} \mathbf{b}_{t+1}}{\sum_x \mathbf{b}_\infty^\intercal \mathbf{B}_x \mathbf{b}_{t+1}}. \tag{2.11}$$

Here, the history of observations $x_{1:t}$ is incorporated into an *internal state* $\mathbf{b}_{t+1}$ that evolves over time by (Hsu et al., 2012)

$$\mathbf{b}_{t+1} = \frac{\mathbf{B}_{x_t} \mathbf{b}_t}{\mathbf{b}_\infty^\intercal \mathbf{B}_{x_t} \mathbf{b}_t}. \tag{2.12}$$

To extend this method to continuous domains, it can be embedded into reproducing kernel Hilbert spaces (Song et al., 2010). We will sketch this approach in Section 3.2, but beforehand, we will give an overview of the embedding of probability distributions into Hilbert spaces in the following section.

3 Reproducing Kernel Hilbert Spaces

Roughly said, Hilbert spaces are an extension of euclidean spaces, which have two or three dimensions, to vector spaces with an arbitrary, maybe infinite, number of dimensions. More formally, Hilbert spaces are inner product spaces of real or complex numbers, where inner product spaces are vector spaces with an associated inner product, which is a function that maps two elements of the vector space to a scalar. Since functions are equivalent to vectors of infinite dimensions, where the elements of the vectors are the function values, it is also possible to define a Hilbert space of functions.

Our work is based on embeddings of probability distributions into so called reproducing kernel Hilbert spaces (RKHS). An RKHS is a Hilbert space of functions $f : \Omega \to \mathbb{R}$ with a scalar product $\langle \cdot, \cdot \rangle$ that is implicitly defined by a kernel function $k : \Omega \times \Omega \to \mathbb{R}$ as $\langle \phi(x), \phi(y) \rangle := k(x, y)$. Here, $\phi(x)$ is a feature mapping into a possibly infinite dimensional space, intrinsic to the kernel function. Particularly for infinite dimensional feature spaces, $\phi(x)$ can also be regarded as a function $f(\cdot) = k(x, \cdot) = \langle \phi(x), \phi(\cdot) \rangle$ of one argument $(\cdot)$, that maps from Ω to $\mathbb{R}$. The functions f that form the elements of an RKHS are weighted sums

$$f(y) = \sum_{i=1}^{N} \alpha_i k(x_i, y), \tag{3.1}$$

of kernel evaluations of the function argument y and given samples x_i with weights $\alpha_i \in \mathbb{R}$. The kernel k has to satisfy the reproducing property (Aronszajn, 1950) that is

$$\begin{aligned} f(y) &= \langle f(\cdot), k(y, \cdot) \rangle && (3.2) \\ &= \left[\sum_{i=1}^{m} \alpha_i k(x_i, \cdot) \right]^T k(y, \cdot) \\ &= \left[\sum_{i=1}^{m} \alpha_i \phi(x_i)^T \right] \phi(y) && (3.3) \\ &= \sum_{i=1}^{m} \alpha_i \phi(x_i)^T \phi(y) \\ &= \sum_{i=1}^{m} \alpha_i k(x_i, y). \end{aligned}$$

The Moore-Aronszajn theorem (Aronszajn, 1950) proofs that every symmetric, positive definite kernel function obeys the reproducing property and, thus, defines a reproducing kernel Hilbert space.

3.1 Embedding Distributions into Reproducing Kernel Hilbert Spaces

Many probabilistic methods in machine learning are based on Gaussian distributed random variables. This has a strong background in the fact that normal distributions provide a sufficiently accurate model to most real-world probabilistic quantities. Yet, also the charming simplicity of the mathematical operations related to normal distributions and the compact representation with mean and variance contributed to a prevalent assumption of Gaussian distributed random variables.

Nevertheless, Gaussian distributions are not an appropriate model for all stochastic processes. The concept of embedding functions into reproducing kernel Hilbert spaces leverages the generalization of machine learning methods to arbitrary probability densities by providing a uniform representation of functions and, thus, probability densities as elements of an infinitely dimensional Hilbert space. Moreover, it is also possible to embed joint and conditional distributions (Song et al., 2009; Grünewälder et al., 2012) into an RKHS and manipulate the probability densities, by means of the chain, sum and Bayes' rule (Song et al., 2013), entirely in Hilbert space.

In the following sections we will sketch the methodologies of embedding probability distributions into reproducing kernel Hilbert spaces (Smola et al., 2007; Song et al., 2013).

3.1.1 Embedding of a Marginal Distribution

The kernel-embedding of a marginal distribution $P(X)$ (Smola et al., 2007) of the random variable $X \in \Omega$ is the expected feature mapping of its random variates

$$\mu_X := \mathbb{E}_X\left[\phi(X)\right] = \int_\Omega \phi(x)\,\mathrm{d}p(x). \tag{3.4}$$

Note that the resulting embedding μ_X is, as an integral over infinite dimensional feature vectors $\phi(x)$, again a function of the RKHS. Similar to the estimation of the mean of a distribution, the kernel-embedding of a marginal distribution, also called the *mean map*, can be estimated using a finite sample estimator

$$\hat{\mu}_X = \frac{1}{m}\sum_{i=1}^{m}\phi(x_i). \tag{3.5}$$

In general, given a set of feature mappings $\Phi = [\phi(x_1), \dots, \phi(x_m)]$, any distribution $q(x)$ over the domain of x_i may be embedded as a linear combination of these feature mappings

$$\hat{\mu}_q = \Phi\beta, \tag{3.6}$$

with a column weight vector β. The mean embedding of a distribution can be used to evaluate expectations over functions that are in the RKHS, i.e., $f = \Phi\alpha$, as

$$\mathbb{E}_q[f(x)] = \langle\hat{\mu}_q, f\rangle = \beta^\intercal\Phi^\intercal\Phi\alpha = \beta^\intercal\mathbf{K}\alpha, \tag{3.7}$$

where $\mathbf{K} = \Phi^\intercal\Phi$ is the Gram matrix with $\mathbf{K}_{ij} = k(x_i, x_j)$.

3.1.2 Embedding of a Joint Distribution

The kernel-embedding of a joint distribution $p(X,Y)$ (Baker, 1973; Smola et al., 2007) of two random variables $X \in \Omega_X$ and $Y \in \Omega_Y$ is defined as the expected tensor product (outer product) of the feature mappings of the random variates of X and Y:

$$\mathcal{C}_{XY} := \mathbb{E}_{XY}\left[\phi(X)\otimes\phi(Y)\right] \tag{3.8}$$

$$= \int_{\Omega_X\times\Omega_Y} \phi(x)\otimes\phi(y)\,\mathrm{d}p(x,y), \tag{3.9}$$

where $\otimes$ denotes the tensor product of two vectors with $a \otimes b = ab^\intercal$. The corresponding finite sample estimator is here

$$\hat{\mathcal{C}}_{XY} = \frac{1}{m}\sum_{i=1}^{m}\phi(x_i)\otimes\phi(y_i). \tag{3.10}$$

However, the joint distribution is not embedded into the same Hilbert space as the marginal distribution in the previous section. Instead, the joint distribution is embedded into a tensor-product RKHS, whose elements are functions $f : \Omega_X \times \Omega_Y \to \mathbb{R}$ of the form

$$f(\cdot,\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, x_i) \otimes k(y_i, \cdot). \tag{3.11}$$

3.1.3 Embedding of a Conditional Distribution

Similar to the embedding of a marginal distribution, shown in Section 3.1.1, the embedding of a conditional distribution (Song et al., 2009, 2013) is defined as

$$\mu_{Y|x} := \mathbb{E}_{Y|x}\left[\phi(Y)\right] = \int_{y\in\Omega} \phi(y)\,\mathrm{d}p(y|x). \tag{3.12}$$

Here, the embedding is not a single element of the RKHS but rather a family of elements. A particular element of the family is chosen by conditioning on a specific value of x. To obtain the conditional embedding for a specific value of x, Song et al. (2013) additionally introduced a *conditional embedding operator* $\mathcal{C}_{Y|X}$ with

$$\mu_{Y|x} = \mathcal{C}_{Y|X}\phi(x). \tag{3.13}$$

Applied to an element $\phi(x)$ of the RKHS, the conditional operator provides the embedding of the distribution $P(Y|x)$ over the variable Y conditioned on x. Given a finite set of m samples

$$D = \{(x_1, y_1), \dots, (x_m, y_m)\} \tag{3.14}$$

of the underlying distribution $P(Y,X)$, the conditional embedding operator can be estimated as

$$\hat{\mathcal{C}}_{Y|X} = \Phi(\mathbf{K} + \lambda \mathbf{I}_m)^{-1}\Upsilon^\intercal, \tag{3.15}$$

with the Gram matrix $\mathbf{K} = \Upsilon^T\Upsilon$, the matrices of the implicit feature mappings of the data $\Phi := (\phi(y_1), \dots, \phi(y_m))$ and $\Upsilon := (\phi(x_1), \dots, \phi(x_m))$, and regularization parameter λ. By applying the conditional operator to a feature vector $\phi(\bar{x})$ of a sample $\bar{x}$, it can be seen that the resulting term describes again an element of the RKHS as

$$\begin{aligned} \hat{\mu}_{Y|\bar{x}} &= \hat{\mathcal{C}}_{Y|X}\phi(\bar{x}) \\ &= \Phi(\mathbf{K} + \lambda \mathbf{I}_m)^{-1}\Upsilon^\intercal\phi(\bar{x}) && (3.16) \\ &= \Phi(\mathbf{K} + \lambda \mathbf{I}_m)^{-1}\mathbf{k}_{\bar{x}}, && (3.17) \end{aligned}$$

with kernel vector $\mathbf{k}_{\bar{x},i} = k(x_i, \bar{x})$. As an element of the RKHS, $\hat{\mu}_{Y|\bar{x}}$ is represented as a weighted sum of feature vectors, where the weights are obtained from the term $(\mathbf{K} + \lambda \mathbf{I}_m)^{-1}\mathbf{k}_{\bar{x}} \in \mathbb{R}^{m\times 1}$. Likewise, the conditional operator can also be applied to an embedding of a probability distribution, which results in the kernel version of the sum rule. Given an embedding $\hat{\mu}_Q = \Psi\beta$, with a sample basis $\Psi = \phi(z_1), \dots, \phi(z_1)$, of a distribution $Q(X) \in \Omega$, Song et al. (2013) define the kernel sum rule as

$$\begin{aligned} \hat{\mu}_{Y|Q} &= \hat{\mathcal{C}}_{Y|X}\hat{\mu}_Q \\ &= \Phi(\mathbf{K} + \lambda \mathbf{I}_m)^{-1}\Upsilon^\intercal\Psi\beta && (3.18) \\ &= \Phi(\mathbf{K} + \lambda \mathbf{I}_m)^{-1}\mathbf{G}\beta, && (3.19) \end{aligned}$$

with Gram matrix $\mathbf{G}_{ij} = k(x_i, z_j)$. Additionally, Song et al. (2013) also define kernel analogs of the chain rule and of the kernel Bayes' rule, which allows to perform Bayesian inference entirely in Hilbert space.

3.2 Embedding the Spectral Algorithm into Reproducing Kernel Hilbert Spaces

The spectral algorithm discussed in Section 2.3 has the drawback that it is only defined for hidden and observable variables with discrete domains. However, this is not applicable for most of the systems we are interested in. Song et al. (2010) presented an embedding of the spectral algorithm into reproducing kernel Hilbert spaces to evade these limitations. In the following sections, we will coarsely retrace the steps of their derivations.

3.2.1 Embedding the Observable Representation in Hilbert Space

Analog to the spectral algorithm for discrete HMMs, Song et al. (2010) begin with an observable operator representaion of the HMM. Assuming an RKHS $\mathcal{F}$ with kernel $k(x,x') = \langle \varphi(x), \varphi(x') \rangle_{\mathcal{F}}$ defined on the observations and an RKHS $\mathcal{G}$ with kernel $l(h,h') = \langle \phi(h), \phi(h') \rangle_{\mathcal{G}}$ defined on the hidden states, they propose an operator $\mathcal{A}_x : \mathcal{G} \mapsto \mathcal{G}$ such that

$$\mathcal{A}_x \phi(h_t) = p(X_t = x | h_t) \mathbb{E}_{H_{t+1}|h_t}[\phi(H_{t+1})]. \tag{3.20}$$

This equation can be read as follows: If applied to an element of the RKHS $\mathcal{G}$, which is in the equation above the embedding $\phi(h_t)$ of a sample h_t, we obtain the expected embedding of the succeeding state $\mathbb{E}_{H_{t+1}|h_t}[\phi(H_{t+1})]$ multiplied with the probability of observing the emission x given h_t.

They further show that the embedding of the predictive distribution $\mu_{X_{t+1}|x_{1:t}}$ given the preceding emissions $x_{1:t}$ can be expressed using the observable operator as

$$\mu_{X_{t+1}|x_{1:t}} = \mathcal{C}_{X_{t+1}|H_{t+1}} \left(\prod_{\tau=1}^{t} \mathcal{A}_{x_\tau} \right) \mu_{H_1} = \mathcal{O} \left(\prod_{\tau=1}^{t} \mathcal{A}_{x_\tau} \right) \mu_{H_1}, \tag{3.21}$$

with the embedding of the initial distribution μ_{H_1} and the observation operator $\mathcal{O}$. The observation operator is defined as a conditional operator $\mathcal{C}_{X_{t+1}|H_{t+1}} = \mathcal{C}_{X|H}$ that maps a distribution over the hidden states embedded into $\mathcal{G}$ to a distribution over the emissions embedded in $\mathcal{F}$. From Equation (3.20) it becomes clear that the computation of the observable operator $\mathcal{A}_x$ would require the embedded transition probability of the hidden states as well as the conditional probability of an observation given the hidden state. However, since the hidden states are generally not available, neither of these densities can be estimated from data and, thus, the same holds for the observable operator. Instead, Song et al. (2010) derive a representation for the predictive embedding $\mu_{X_{t+1}|x_{1:t}}$, similar to the spectral algorithm for discrete HMMs, that is based solely on observable data. They define

$$\mu_1 := \mathbb{E}_{X_t} \left[\varphi(X_t) \right] = \mu_{X_t}, \tag{3.22}$$

$$\mathcal{C}_{2,1} := \mathbb{E}_{X_{t+1}X_t} \left[\varphi(X_{t+1}) \otimes \varphi(X_t) \right] = \mathcal{C}_{X_{t+1}X_t}, \tag{3.23}$$

$$\begin{aligned} \mathcal{C}_{3,x,1} &:= p(X_{t+1} = x) \mathbb{E}_{X_{t+2},X_t|X_{t+1}=x} \left[\varphi(X_{t+2}) \otimes \varphi(X_t) \right] \\ &= p(X_{t+1} = x) \mathcal{C}_{3,1|2} \varphi(x), \end{aligned} \tag{3.24}$$

where they made use of the conditional operator $\mathcal{C}_{3,1|2} = \mathcal{C}_{X_{t+1}X_t|X_{t+1}}$ of the tensor product RKHS $\mathcal{F} \otimes \mathcal{F}$. This is necessary, since $P(X_{t+2}, X_{t+1} = x, X_t)$ is an improper probability distribution, which cannot be embedded into an RKHS. However, according to the chain rule, this improper distribution can be split into $P(X_{t+2}, X_t | X_{t+1}) p(X_{t+1} = x)$ and, hence, be embedded as the conditional operator $\mathcal{C}_{3,1|2}$ multiplied with the probability $p(X_{t+1} = x)$.

Analog to the discrete algorithm, they continue by defining a subspace projection $\mathbf{U}$ which obeys the condition that $\mathbf{U}^\intercal \mathcal{O}$ is invertible. This projection can again be obtained by a singular value decomposition of the joint embedding $\mathcal{C}_{2,1}$. With the projection $\mathbf{U}$ and the observable quantities defined in Equations (3.22)-(3.24) they define the embedded observable representation of an HMM as

$$\beta_1 := \mathbf{U}^\intercal \mu_1 = (\mathbf{U}^\intercal \mathcal{O})\,\mu_1, \tag{3.25}$$

$$\beta_\infty := \mathcal{C}_{2,1}\left(\mathbf{U}^\intercal \mathcal{C}_{2,1}\right)^\dagger = \mathcal{O}\,(\mathbf{U}^\intercal \mathcal{O})^{-1}, \tag{3.26}$$

$$\mathcal{B}_x := \left(\mathbf{U}^\intercal \mathcal{C}_{3,x,1}\right)\left(\mathbf{U}^\intercal \mathcal{C}_{2,1}\right)^\dagger = (\mathbf{U}^\intercal \mathcal{O})\,\mathcal{A}_x\,(\mathbf{U}^\intercal \mathcal{O})^{-1}. \tag{3.27}$$

It is straightforward to see that the embedded representation (β_1, β_∞ and $\mathcal{B}_x$) can express the embedded predictive density $\mu_{X_{t+1}|x_{1:t}}$ equivalent to Equation (3.21) as

$$\mu_{X_{t+1}|x_{1:t}} = \beta_\infty \mathcal{B}_{x_{t:1}} \beta_1. \tag{3.28}$$

Since x is of a continuous domain (i.e., there are infinitely many possible values for x), it is infeasible to estimate $\mathcal{C}_{3,x,1}$, as this would require a distinct operator $\mathcal{C}_{3,x,1}$ for each instance of x. Instead they use the conditional operator $\mathcal{C}_{3,1|2}$ applied to the embedding $\varphi(x)$, which is a multiple of $\mathcal{C}_{3,x,1}$ by the probability $p(X_{t+1} = x)$, and replace the quantity from Equation (3.27) with

$$\bar{\mathcal{B}}_x := (\mathbf{U}^\intercal \mathcal{C}_{3,1|2}\varphi(x))(\mathbf{U}^\intercal \mathcal{C}_{2,1})^\dagger. \tag{3.29}$$

Consequently, they obtain the embedded representation

$$\mu_{X_{t+1}|x_{1:t}} \propto \beta_\infty \bar{\mathcal{B}}_{x_{t:1}} \beta_1 \tag{3.30}$$

of the embedded predictive density. Note that this representation is only proportional to the predictive embedding as the probabilities $p(X_{t+1} = x)$ are in general not available and, thus, omitted in the modified quantities $\bar{\mathcal{B}}_{x_{t:1}}$.

The following section shows how to employ the “kernel-trick” for the RKHS-embedded spectral methods to derive a computationally feasible algorithm.

3.2.2 The Kernel Spectral Algorithm for Hidden Markov Models

Because the feature mapping which is implicit to the kernel function is usually a mapping into an infinitely dimensional domain, it is generally not possible to compute neither the observable quantities as stated in Equations (3.22) - (3.24) nor the observable representation in Equations (3.25), (3.26) and (3.29). Also the singular value decomposition (SVD) of $\mathcal{C}_{2,1}$ is not performable. However, by making use of a finite set of samples and employing the “kernel trick” (i.e., expressing all operations in the infinite feature space in terms of scalar products that evaluate to kernel matrices of finite (cardinality of sample set) size), the methods can be resolved to simple Gram matrix manipulations. In the following paragraphs we will sketch the derivations of the observable representation (Song et al., 2010) based on observed data and show how it can be used to predict future states.

To implement the kernel spectral algorithm, Song et al. (2010) assume a training set “of m *i.i.d.* triples $(x_1^l, x_2^l, x_3^l)_{l=1}^m$”. They define the feature matrices $\Upsilon = (\varphi(x_1^1), \ldots, \varphi(x_1^m))$, $\Phi = (\varphi(x_2^1), \ldots, \varphi(x_2^m))$ and $\Psi = (\varphi(x_3^1), \ldots, \varphi(x_3^m))$ of the first, second and third triple states respectively, where $\Upsilon, \Phi, \Psi \in \mathbb{R}^{\infty \times m}$. Using the definition from Equation (3.10) and the feature matrices Υ and Φ, the joint operator can be estimated as

$$\hat{\mathcal{C}}_{2,1} = \frac{1}{m}\Phi\Upsilon^\intercal. \tag{3.31}$$

Without loss of generality, they set the singular vectors $\mathbf{v}_i$ of the joint embedding $\hat{\mathcal{C}}_{2,1}$ equal to a linear combination $\mathbf{v}_i = \Phi\alpha_i$ of the features of the second states in the triples with a weight vector $\alpha_i \in \mathbb{R}^m$.

By exploiting the relation of the eigendecomposition to SVD, the singular values $\mathbf{v}_i$ can be obtained by solving the generalized eigenvalue problem

$$
\begin{aligned}
&& \Phi\Upsilon^\intercal\Upsilon\Phi^\intercal\mathbf{v} = w\mathbf{v} && (3.32)\\
\Leftrightarrow && \Phi\mathbf{K}\Phi^\intercal\Phi\alpha = w\Phi\alpha && (3.33)\\
\Leftrightarrow && \Phi\mathbf{K}\mathbf{L}\alpha = w\Phi\alpha && (3.34)\\
\Leftrightarrow && \Phi^\intercal\Phi\mathbf{K}\mathbf{L}\alpha = w\Phi^\intercal\Phi\alpha && (3.35)\\
\Leftrightarrow && \mathbf{L}\mathbf{K}\mathbf{L}\alpha = w\mathbf{L}\alpha, && (3.36)
\end{aligned}
$$

with $w \in \mathbb{R}$ and the kernel matrices $\mathbf{K} = \Upsilon^\intercal\Upsilon \in \mathbb{R}^{m\times m}$ and $\mathbf{L} = \Phi^\intercal\Phi \in \mathbb{R}^{m\times m}$. While, in general, it is not possible to compute the embedding $\mathcal{C}_{2,1}$, the kernel matrices can be obtained from the kernel function as $\mathbf{K}_{ij} = \left\langle \varphi(x_1^i), \varphi(x_1^j)\right\rangle = k(x_1^i, x_1^j)$. Song et al. (2010) further show that an approximation of the projection $\mathbf{U}$ can be expressed as

$$\hat{\mathbf{U}} = \Phi\mathbf{A}\mathbf{D} \in \mathbb{R}^{\infty\times N}, \tag{3.37}$$

where $\mathbf{A} := \{\alpha_1, \ldots, \alpha_N\} \in \mathbb{R}^{m\times N}$ is the matrix of the top N eigenvectors α_i and $\mathbf{D} \in \mathbb{R}^{N\times N}$ is a diagonal matrix that normalizes the singular vectors $v = \Phi\mathbf{A}$ (see Equation (A.11)). Similar to the spectral algorithm for discrete HMM, N denotes the assumed number of hidden states.

Using the definition of the marginal embedding from Equation (3.5), the embedding of the stationary density can be estimated as $\hat{\mu}_1 = \frac{1}{m}\Upsilon\mathbf{1}_m$. By employing $\hat{\mu}_1$ to Equation (3.25), the initial belief of the observable representation can be estimated as

$$
\begin{aligned}
\hat{\beta}_1 &= \hat{\mathbf{U}}^\intercal\hat{\mu}_1 && (3.38)\\
&= \frac{1}{m}\mathbf{D}^\intercal\mathbf{A}^\intercal\Phi^\intercal\Upsilon\mathbf{1}_m && (3.39)\\
&= \frac{1}{m}\mathbf{D}^\intercal\mathbf{A}^\intercal\mathbf{G}\mathbf{1}_m \in \mathbb{R}^{N\times 1}, && (3.40)
\end{aligned}
$$

with the kernel matrix $\mathbf{G} = \Phi^\intercal\Upsilon \in \mathbb{R}^{m\times m}$. The estimate of the second parameter β_∞ can be derived from Equation (3.26), with the estimates $\hat{\mathbf{U}}$ and $\hat{\mathcal{C}}_{2,1}$ as

$$
\begin{aligned}
\hat{\beta}_\infty &= \hat{\mathcal{C}}_{2,1}\left(\hat{\mathbf{U}}^\intercal\hat{\mathcal{C}}_{2,1}\right)^\dagger && (3.41)\\
&= \frac{1}{m}\Phi\Upsilon^\intercal\left(\mathbf{D}^\intercal\mathbf{A}^\intercal\Phi^\intercal\frac{1}{m}\Phi\Upsilon^\intercal\right)^\dagger && (3.42)\\
&= \Phi\mathbf{K}\mathbf{L}\mathbf{A}\mathbf{D}\Omega^{-1} \in \mathbb{R}^{\infty\times N}, && (3.43)
\end{aligned}
$$

with the diagonal matrix $\Omega = \operatorname{diag}(w_1, \ldots, w_N) \in \mathbb{R}^{N\times N}$ of the eigenvalues w_i from the generalized eigenvalue problem in Equation (3.36). A full derivation of β_∞ can be found in Appendix A.1.

By applying the definition of a conditional embedding operator for tensor product features (Song et al., 2013), they get

$$\hat{\mathcal{C}}_{3,1|2} = \Psi\operatorname{diag}\left(\left(\mathbf{L} + \lambda\mathbf{I}_m\right)^{-1}\Phi^\intercal(\cdot)\right)\Upsilon^\intercal, \tag{3.44}$$

with identity matrix $\mathbf{I}_m$ and regularization factor λ. The conditioning on an observation x is done by inserting its feature mapping for the free argument $(\cdot)$ of the conditional embedding operator. With Equation (3.44) the estimate of the third parameter of the observable representation (c.f. Equation (3.29)) becomes

$$
\begin{aligned}
\bar{\mathcal{B}}_x &= \left(\hat{\mathbf{U}}^\intercal\hat{\mathcal{C}}_{3,1|2}\varphi(x)\right)\left(\hat{\mathbf{U}}^\intercal\hat{\mathcal{C}}_{2,1}\right)^\dagger && (3.45)\\
&= \mathbf{D}^\intercal\mathbf{A}^\intercal\Phi^\intercal\Psi\operatorname{diag}\left(\left(\mathbf{L} + \lambda\mathbf{I}_m\right)^{-1}\Phi^\intercal\varphi(x)\right)\Upsilon^\intercal\left(\mathbf{D}^\intercal\mathbf{A}^\intercal\Phi^\intercal\frac{1}{m}\Phi\Upsilon^\intercal\right)^\dagger && (3.46)\\
&= \mathbf{D}^\intercal\mathbf{A}^\intercal\mathbf{F}\operatorname{diag}\left(\left(\mathbf{L} + \lambda\mathbf{I}_m\right)^{-1}\Phi^\intercal\varphi(x)\right)\mathbf{K}\mathbf{L}\mathbf{A}\mathbf{D}\Omega^{-1} \in \mathbb{R}^{N\times N}. && (3.47)
\end{aligned}
$$

with kernel matrix $\mathbf{F} = \Phi^\intercal \Psi \in \mathbb{R}^{m \times m}$. The derivation of the last equation is similar to the derivation of Equation (3.43) shown in Appendix A.1. With $\hat{\beta}_1$, $\hat{\beta}_\infty$ and $\bar{\mathcal{B}}_x$ it is possible to express the estimated conditional embedding

$$\hat{\mu}_{X_{t+1}|x_{1:t}} \propto \hat{\beta}_\infty \bar{\mathcal{B}}_{x_{t:1}} \hat{\beta}_1. \tag{3.48}$$

Yet, this provides only an embedding of an improper conditional density, proportional to $P(X_{t+1}|x_{1:t})$. It remains to find the most probable future state x_{t+1} which maximizes the function

$$f(x) = \left\langle \hat{\mu}_{X_{t+1}|x_{1:t}}, \varphi(x) \right\rangle_{\mathcal{F}}, \tag{3.49}$$

that is, we search for a state that, if interpreted as a delta distribution and embedded into the RKHS, is most similar to $\hat{\mu}_{X_{t+1}}$. Since this optimization problem is generally hard to solve, Song et al. (2010) propose to search the given training data for the sample that maximizes $f(x)$.

4 Kalman Filtering in Reproducing Kernel Hilbert Spaces

In this section, we will introduce our approach to extend hidden Markov models to continuous domains and non-linear system dynamics by embedding Kalman filters into reproducing kernel Hilbert spaces. Our model is based on data windows of the time series emitted from the underlying system. This is similar to the concept of auto-regressive hidden Markov models, which compute the belief over the next state directly from a window of the past states. Our approach, in contrast, evolves a probabilistic notion of the data window and, additionally, allows for data windows that cover also an arbitrary number of future time steps.

In Section 4.1, we will first explain the data windows in detail and show how to compactly represent them using radial basis functions. In Section 4.2, we will define a non-embedded hidden Markov model on these data windows and explain the transition and observation models required for Kalman filtering. In Section 4.3, we will finally present our non-linear extension to Kalman filters by embedding these models into reproducing kernel Hilbert spaces.

4.1 Weight-Based Representation of Data Windows

We assume a time series

$$\mathcal{D} = \{x_1, \dots, x_n\} \tag{4.1}$$

of observed states emitted from an underlying system as given and regard a subset of τ consecutive states of $\mathcal{D}$ as a data window

$$x_t^\tau = [x_{t-\tau^-}, \dots, x_t, \dots, x_{t+\tau^+}]^\intercal. \tag{4.2}$$

For a specific point in time t, the data window x_t^τ captures an arbitrary number of past and future states, τ^- and τ^+, respectively. Similar to Paraschos et al. (2013), we represent a data window as a linear combination

$$\begin{align} x_t^\tau &= \left(\sum_{i=1}^{k} \psi_i w_{t,i} \right) + \varepsilon_\Psi \tag{4.3} \\ &= \Psi \mathbf{w}_t + \varepsilon_\Psi \tag{4.4} \end{align}$$

of k basis function vectors ψ_i with the weights $w_{t,i}$. The basis functions are stationary with respect to the data window, i.e., they do only depend on the relative time within a data window. We assume an additive zero-mean Gaussian model error $\varepsilon_\Psi \sim \mathcal{N}(0, \Sigma_\Psi)$ that incorporates both the noise on the data as well as the reconstruction error of the data window from the weights. In vector-matrix notation we have the product of the basis function matrix Ψ and the weight vector $\mathbf{w}_t$. We normalize the sum of all basis functions to 1 at any given time. Thus, the entries of the basis function matrix are computed as

$$\Psi_{i,j} = \frac{b_j(i)}{\sum_k b_k(i)}. \tag{4.5}$$

Common basis functions are the radial basis functions (RBFs), for example the squared exponential basis function

$$b_i(t) = \exp\left(\frac{(t-\mu_i)^2}{l}\right), \tag{4.6}$$

with mean μ_i and length scale l. Alternatively, using the Dirac basis function

$$\delta_i(t) = \begin{cases} 1 & \text{if } t = c_i \\ 0 & \text{otherwise} \end{cases}, \tag{4.7}$$

with center c_i, is equivalent to omitting the basis transformation of the data windows x^τ to a weight-based representation, i.e., using the data windows directly as $\mathbf{w}$. Nevertheless, the transformation of the data windows into a weight space using RBFs provides not only a more compact (assuming $k < \tau$) but also a smoother, i.e., less noisy, representation of an observation window. Moreover, a weight-based representation using continuous basis functions allows for non-isochronous data streams.

Usually, robotic systems emit many measured quantities as a multi-dimensional state. For such multi-dimensional time series

$$\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\},\ \mathbf{x}_i = [x_{1,i}, \ldots, x_{m,i}]^\intercal \tag{4.8}$$

we use separate data windows $x^\tau_{d,t}, d \in 1, \ldots, m$ for each dimension d. We concatenate these windows into a big window vector

$$\mathbf{x}^\tau_t = \left[\left(x^\tau_{1,t}\right)^\intercal, \ldots, \left(x^\tau_{m,t}\right)^\intercal\right]^\intercal \tag{4.9}$$

and use a block-diagonal matrix

$$\Gamma = \begin{pmatrix} \Psi & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \Psi \end{pmatrix} \tag{4.10}$$

to obtain the weight-based representation of the window vector $\mathbf{x}^\tau_t$ analog to Equation (4.4) as

$$\mathbf{x}^\tau_t = \Gamma \mathbf{w}_t + \varepsilon_\Gamma. \tag{4.11}$$

For a given data window x^τ_t, we can obtain the weights $\mathbf{w}_t$ using the least squares method which results in a multiplication of the left Moore-Penrose pseudo-inverse of Ψ (or Γ for multidimensional data) with the data window

$$\mathbf{w}_t = (\Psi^\intercal \Psi)^{-1} \Psi^\intercal x^\tau_t. \tag{4.12}$$

We assume the weights $\mathbf{w}_t$ to be the latent states of the hidden Markov model. This assumption is based on two hypotheses. The first is that the least squares estimate filters out the noise on the data windows and, thus, provides an adequate approximation of the true data. The second is that, similar to the assumption of an auto-regressive HMM, a (noise-free) data window is sufficient to predict the next state and, hence, the next data window.

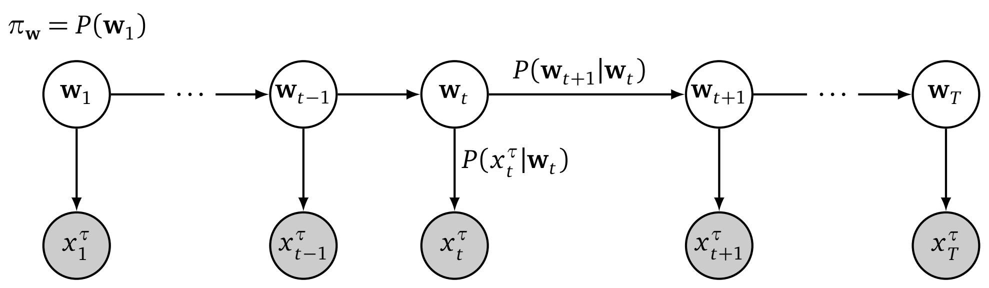


Figure 4.1.: Graphical structure of the linear weight-based hidden Markov model, with the weight vectors $\mathbf{w}_i$ as hidden states and the data windows x_i^τ as observations.

4.2 Linear Weight-Based Kalman Filter

In this section, we explain a probabilistic model that expresses the belief of the current data window x_t^τ. We will further introduce methods to advance this believe over time according to the dynamics of the underlying system and to condition the belief of a data window on a new observation x_t. In its essence, the formalism that results from the derivations in this section is basically a Kalman filter in weight space. The hidden Markov model that we assume on the weights is depicted in Figure 4.1.

From Equation (4.4), we get the conditional probability density of a data window x_t^τ given the weights $\mathbf{w}_t$ and the model error of the basis transformation Σ_Ψ, which describes the observation model as

$$p(x_t^\tau|\mathbf{w}_t) = \mathcal{N}(x_t^\tau|\Psi\mathbf{w}_t, \Sigma_\Psi). \tag{4.13}$$

Additionally, we assume the belief over the hidden state $\mathbf{w}_t$ to be Gaussian

$$p(\mathbf{w}_t|\mu_{\mathbf{w},t}, \Sigma_{\mathbf{w},t}) = \mathcal{N}(\mathbf{w}_t|\mu_{\mathbf{w},t}, \Sigma_{\mathbf{w},t}), \tag{4.14}$$

with mean $\mu_{\mathbf{w},t}$ and covariance $\Sigma_{\mathbf{w},t}$, which incorporates the current variance of the data windows. From Equations (4.14) and(4.13), we obtain the distribution over the data windows by marginalizing out the weights $\mathbf{w}$ as

$$\begin{aligned} p(x_t^\tau|\mu_{\mathbf{w},t}, \Sigma_{\mathbf{w},t}) &= \int p(x_t^\tau|\mathbf{w})p(\mathbf{w}|\mu_{\mathbf{w}_t}, \Sigma_{\mathbf{w}_t})\mathrm{d}\mathbf{w} & (4.15)\\ &= \mathcal{N}(x_t^\tau|\Psi\mu_{\mathbf{w},t}, \Psi\Sigma_{\mathbf{w},t}\Psi^\intercal + \Sigma_\Psi). & (4.16) \end{aligned}$$

Hence, the sufficient statistic to expresses the belief about the current data window x_t^τ are the mean $\mu_{\mathbf{w},t}$, the variance $\Sigma_{\mathbf{w},t}$ and the model error of the basis transformation Σ_Ψ. While the latter is stationary, the mean $\mu_{\mathbf{w},t}$ and the variance $\Sigma_{\mathbf{w},t}$ dependent on the past observations $x_{1:t-1}$ as well as on the dynamics of the underlying system. Therefore, these quantities need to be a) advanced over time by applying a transitional model

$$\mathbf{C}_\mathbf{w} : \mu_{\mathbf{w},t} \mapsto \mu_{\mathbf{w},t+1} + \varepsilon_{\mathbf{C}_\mathbf{w}}, \tag{4.17}$$

with noise $\varepsilon_{\mathbf{C}_\mathbf{w}} \sim \mathcal{N}(0, \Sigma_{\mathbf{C}_\mathbf{w}})$, which we will define in Section 4.2.1 and b) updated when a new observations is emitted by conditioning on that new observation x_t, which we will show in Section 4.2.2.

4.2.1 Transition Model

The transition model advances the belief of the weights $\mathbf{w}_t$ at time t, given by the mean $\mu_{\mathbf{w},t}$ and the variance $\Sigma_{\mathbf{w},t}$, to an *a priori* belief of the weights $\mathbf{w}_{t+1}$ at time $t+1$, expressed by the a priori mean

$\hat{\mu}_{\mathrm{w},t+1}$ and the a priori variance $\hat{\Sigma}_{\mathrm{w},t+1}$. The advanced belief is called a priori belief, since it is solely based on the transitional model, whereas the conditioning on a new observation has yet to be done. For now, we assume linear dynamics on the weights

$$\mathbf{w}_{t+1} = \mathbf{C}_\mathrm{w}\mathbf{w}_t + \varepsilon_{\mathbf{C}_\mathrm{w}}, \tag{4.18}$$

with transition matrix $\mathbf{C}_\mathrm{w}$ and zero-mean Gaussian noise $\varepsilon_{\mathbf{C}_\mathrm{w}} \sim \mathcal{N}(0, \Sigma_{\mathbf{C}_\mathrm{w}})$. Note that the linear assumption is, however, a very limiting constraint that most real-world systems do not satisfy. Based on the linear dynamics, we can formulate a transition probability density of $\mathbf{w}_{t+1}$ given $\mathbf{w}_t$ as

$$p(\mathbf{w}_{t+1}|\mathbf{w}_t) = \mathcal{N}(\mathbf{w}_{t+1}|\mathbf{C}_\mathrm{w}\mathbf{w}_t, \Sigma_{\mathbf{C}_\mathrm{w}}) \tag{4.19}$$

and obtain the a priori belief of $\mathbf{w}_{t+1}$ by applying this transition probability density to the belief of the current weights $\mathbf{w}_t$ and subsequently marginalizing over $\mathbf{w}_t$. Hence, we arrive at

$$\begin{aligned} p(\mathbf{w}_{t+1}|\hat{\mu}_{\mathrm{w},t+1}, \hat{\Sigma}_{\mathrm{w},t+1}) &= \int p(\mathbf{w}_{t+1}|\mathbf{w}_t) p(\mathbf{w}_t|\mu_{\mathrm{w},t}, \Sigma_{\mathrm{w},t}) \mathrm{d}\mathbf{w}_t \\ &= \int \mathcal{N}(\mathbf{w}_{t+1}|\mathbf{C}_\mathrm{w}\mathbf{w}_t, \Sigma_{\mathbf{C}_\mathrm{w}}) \mathcal{N}(\mathbf{w}_t|\mu_{\mathrm{w},t}, \Sigma_{\mathrm{w},t}) \mathrm{d}\mathbf{w}_t \\ &= \mathcal{N}(\mathbf{w}_{t+1}|\mathbf{C}_\mathrm{w}\mu_{\mathrm{w},t}, \mathbf{C}_\mathrm{w}\Sigma_{\mathrm{w},t}\mathbf{C}_\mathrm{w}^\mathsf{T} + \Sigma_{\mathbf{C}_\mathrm{w}}) \end{aligned} \tag{4.20}$$

with a priori mean $\hat{\mu}_{\mathrm{w},t+1} = \mathbf{C}_\mathrm{w}\mu_{\mathrm{w},t}$ and a priori variance $\hat{\Sigma}_{\mathrm{w},t+1} = \mathbf{C}_\mathrm{w}\Sigma_{\mathrm{w},t}\mathbf{C}_\mathrm{w}^\mathsf{T} + \Sigma_{\mathbf{C}_\mathrm{w}}$.

4.2.2 Conditioning on Observations

Given the a priori belief about the weights $\mathbf{w}_t$ at time t, expressed by the mean $\hat{\mu}_{\mathrm{w},t}$ and the variance $\hat{\Sigma}_{\mathrm{w},t}$, we need to update this belief by the incoming observation, i.e., infer the *a posteriori* mean $\mu_{\mathrm{w},t}$ and variance $\Sigma_{\mathrm{w},t}$, by conditioning on a new observation x_t.

At first, we formulate the probability of observing a single emission x_t given the weights $\mathbf{w}_t$. This probability density is similar to Equation (4.13), but for the conditioning of an single observation we need only the probability of a single data point of the window. Hence, we get

$$p(x_t|\mathbf{w}_t, \Sigma_{\Psi_0}) = \mathcal{N}(x_t|\Psi_0\mathbf{w}_t, \Sigma_{\Psi_0}), \tag{4.21}$$

where Ψ_0 is the row of the basis function matrix Ψ and Σ_{Ψ_0} is the entry of the diagonal of Σ_Ψ that correspond to the location of the current observation x_t in the current data window x_t^τ. Furthermore, it is also possible to condition on observations that occur between two of the predefined time spots of the basis function matrix by setting Ψ_0 to the magnitudes of each basis function at that specific point in time. With the probability density of x_t given in Equation (4.21) and the a priori probability density of the current weights

$$p(\mathbf{w}_t) = \mathcal{N}(\mathbf{w}_t|\hat{\mu}_{\mathrm{w},t}, \hat{\Sigma}_{\mathrm{w},t}), \tag{4.22}$$

we can then formulate a marginal probability density of the current observation x_t, analog to Equations (4.15) and (4.16), as

$$\begin{aligned} p(x_t|\theta) &= \int p(x_t|\mathbf{w}_t) \mathcal{N}(\mathbf{w}_t|\hat{\mu}_{\mathrm{w},t}, \hat{\Sigma}_{\mathrm{w},t}) \mathrm{d}\mathbf{w}_t \\ &= \mathcal{N}(x_t|\Psi_0\hat{\mu}_{\mathrm{w},t}, \Psi_0\hat{\Sigma}_{\mathrm{w},t}\Psi_0^\mathsf{T} + \Sigma_{\Psi_0}), \end{aligned} \tag{4.23}$$

with parameters $\theta = \{\hat{\mu}_{\mathbf{w},t}, \hat{\Sigma}_{\mathbf{w},t}, \Sigma_{\Psi_0}\}$.

Second, from Equations (4.22) and (4.23), we can derive a joint probability density of the weights $\mathbf{w}_t$ and the observation x_t as

$$\begin{aligned} p(\mathbf{w}_t, x_t|\theta) &= p(x_t|\theta)\mathcal{N}(\mathbf{w}_t|\hat{\mu}_{\mathbf{w},t}, \hat{\Sigma}_{\mathbf{w},t}) && (4.24)\\ &= \mathcal{N}\left(\begin{matrix}\mathbf{w}_t\\ x_t\end{matrix}\middle|\begin{matrix}\hat{\mu}_{\mathbf{w},t}\\ \Psi_0\hat{\mu}_{\mathbf{w},t}\end{matrix}, \begin{matrix}\hat{\Sigma}_{\mathbf{w},t} & \hat{\Sigma}_{\mathbf{w},t}\Psi_0^{\intercal}\\ \Psi_0\hat{\Sigma}_{\mathbf{w},t} & \Sigma_{\Psi_0} + \Psi_0\hat{\Sigma}_{\mathbf{w},t}\Psi_0^{\intercal}\end{matrix}\right). \end{aligned}$$

Finally, we infer the conditional density of $\mathbf{w}_t$ given x_t from Equations (4.24) and (4.23) by applying Bayes' rule and obtain

$$\begin{aligned} p(\mathbf{w}_t|x_t, \theta) &= \frac{p(\mathbf{w}_t, x_t|\theta)}{p(x_t|\theta)} && (4.25)\\ &= \mathcal{N}\left(\mathbf{w}_t|\mu_{\mathbf{w},t}, \Sigma_{\mathbf{w},t}\right), && (4.26) \end{aligned}$$

with a posteriori mean

$$\mu_{\mathbf{w},t} = \hat{\mu}_{\mathbf{w},t} + \hat{\Sigma}_{\mathbf{w},t}\Psi_0^{\intercal}(\Sigma_{\Psi_0} + \Psi_0\hat{\Sigma}_{\mathbf{w},t}\Psi_0^{\intercal})^{-1}(x_t - \Psi_0\hat{\mu}_{\mathbf{w},t}) \tag{4.27}$$

and a posteriori variance

$$\Sigma_{\mathbf{w},t} = \hat{\Sigma}_{\mathbf{w},t} - \hat{\Sigma}_{\mathbf{w},t}\Psi_0^{\intercal}(\Sigma_{\Psi_0} + \Psi_0\hat{\Sigma}_{\mathbf{w},t}\Psi_0^{\intercal})^{-1}\Psi_0\hat{\Sigma}_{\mathbf{w},t}. \tag{4.28}$$

4.3 Hilbert Space Embedded Kalman Filter

The linear model presented in the last section has two major downsides: The first disadvantage is the assumption that we made for the transition model, which is that the dynamics of the system are linear in the weight space. This clearly is a constraint with which most real systems, especially robotic systems, do not comply as the dynamics of robots are typically non-linear. The second problem of the linear model is the representation of the probability distributions. The Kalman filter uses the first and the second moment (i.e., the mean and covariance) as the representation of a probability density, which is the sufficient statistic for a Gaussian density. However, a Gaussian density may not always provide a sufficiently close approximation to the true density.

With our approach to RKHS-embedded hidden Markov models, we try to evade at least the first problem. The dynamics are now only assumed to be linear in the kernel space, which—if choosing an appropriate kernel function and a sufficient number of kernel samples—mostly all dynamical systems satisfy. (A similar assumption is also inherent to the conditional embedding.) Nevertheless, by imposing a Gaussian distribution of the kernel embeddings and by exploiting the analytic solutions to integrals over Gaussians, we are still assuming Gaussian distributed data windows. Notwithstanding these restrictions, the embedding into an RKHS yields a much more expressive hidden state, which is the Hilbert space embedding of the distribution over the weights $\mathbf{w}$. Essentially, we define a Kalman filter over the Hilbert space embedding of the Gaussian random variable representation of the state.

4.3.1 Hilbert Space Embedding of the Weight-Based Hidden Markov Model

We embed the weight-based hidden Markov model into a reproducing kernel Hilbert space $\mathcal{H}$ with a scalar product $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ that is inherent to a kernel function $k(\mathbf{w}, \mathbf{w}')$. Based on a set of N sample weights, we embed the probability density over the weights $p(\mathbf{w})$, according to the empirical estimator of a marginal distribution given in Equation (3.5), as

$$\hat{\mu}_\phi = \frac{1}{N}\sum_{i=1}^{N} \phi(\mathbf{w}_i). \tag{4.29}$$

Additionally, we embed the variance of the weight vectors using the same set of N samples into a tensor product RKHS $\mathcal{G} = \mathcal{H} \otimes \mathcal{H}$, according to the empirical estimator of a joint distribution as shown in Equation (3.10), by

$$\hat{\Sigma}_\phi = \frac{1}{N-1}\sum_{i=1}^{N} \hat{\phi}(\mathbf{w}_i)\hat{\phi}(\mathbf{w}_i)^\intercal, \tag{4.30}$$

where $\hat{\phi}(\mathbf{w}_i) = \phi(\mathbf{w}_i) - \mu_\phi$ are the centered feature mappings of the samples. With these two quantities, the mean map $\hat{\mu}_\phi$ and the covariance operator $\hat{\Sigma}_\phi$, we can express the probability density of the embeddings $\phi(\mathbf{w})$ by an infinite dimensional Gaussian

$$p(\phi(\mathbf{w})|\hat{\mu}_\phi, \hat{\Sigma}_\phi) = \mathcal{N}(\phi(\mathbf{w})|\hat{\mu}_\phi, \hat{\Sigma}_\phi). \tag{4.31}$$

To retrieve a belief over the weights from the kernel embedding, we define a conditional operator $\Lambda : \mathcal{H} \mapsto \mathbb{R}^k$ that maps an embedding $\phi(\mathbf{w})$ to the weights $\mathbf{w} \in \mathbb{R}^k$ as

$$\Lambda := \mathbf{W}(\Phi^\intercal\Phi + \lambda \mathbf{I}_N)^{-1}\Phi^\intercal \in \mathbb{R}^{k\times\infty}, \tag{4.32}$$

where $\mathbf{W} = \left[\mathbf{w}_1, \dots, \mathbf{w}_N\right]$ is the matrix of all weight samples and $\Phi = \left[\phi(w_1), \dots, \phi(w_N)\right]$ is the matrix of all sample embeddings. We assume here that the mapping

$$\mathbf{w} = \Lambda\phi(\mathbf{w}) + \varepsilon_\Lambda, \tag{4.33}$$

from $\mathcal{H}$ back into the weight space, has a zero-mean Gaussian distributed model error $\varepsilon_\Lambda \sim \mathcal{N}(0, \Sigma_\Lambda)$. With the conditional operator Λ, the mean embedding $\hat{\mu}_\phi$ and the embedding of the covariance $\hat{\Sigma}_\phi$, we can express the probability density of the weights $\mathbf{w}$ by integrating out the sample embedding $\phi(w)$ as

$$\begin{aligned} p(\mathbf{w}|\hat{\mu}_\phi, \hat{\Sigma}_\phi) &= \int p(\mathbf{w}|\phi(\mathbf{w}))p(\phi(\mathbf{w})|\hat{\mu}_\phi, \hat{\Sigma}_\phi)\mathrm{d}\phi(\mathbf{w}) && (4.34)\\ &= \int \mathcal{N}(\mathbf{w}|\Lambda\phi(\mathbf{w}), \Sigma_\Lambda)\mathcal{N}(\phi(\mathbf{w})|\hat{\mu}_\phi, \hat{\Sigma}_\phi)\mathrm{d}\phi(\mathbf{w}) && (4.35)\\ &= \mathcal{N}(\mathbf{w}|\Lambda\hat{\mu}_\phi, \Lambda\hat{\Sigma}_\phi\Lambda^\intercal + \Sigma_\Lambda). && (4.36) \end{aligned}$$

Figure 4.2 depicts the hidden Markov model that we assume on the embeddings of the weights in Hilbert space. Analog to the linear model, we have at a certain point in time t a belief over the weights $\mathbf{w}_t$. Embedded into the RKHS, this belief is expressed by the mean map $\mu_{\phi,t}$ and the covariance operator $\hat{\Sigma}_{\phi,t}$. Similar to the linear model, we will advance these quantities over time and update them when we get new observations. The embeddings are, as aforementioned, sample-based estimators of the true embeddings, but for the sake of readability we will omit the notational hint from now on. For the same reason, we will also denote a sample embedding $\phi(\mathbf{w}_t)$ as ϕ_t.

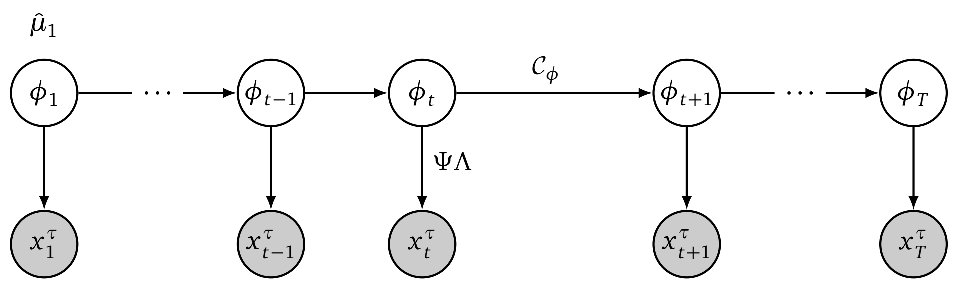


Figure 4.2.: Graphical structure of the Hilbert space embedded hidden Markov model, with the embeddings of the weight vectors ϕ_i as hidden states and the data windows w_i^τ as observations. The conditional relations between a hidden node at time t to the hidden node at time $t+1$ is modeled by the transition operator $\mathcal{C}_\phi$, defined in Equation (4.37). The conditional relations between a hidden node and an observable node is modeled by the conditional operator Λ, defined in Equation (4.32), and the basis function matrix Ψ.

4.3.2 Transition Model in Hilbert Space

The data window evolves over time according to the system dynamics. In this section, we will show how we can account for the dynamics by advancing an *a posteriori* belief of the data window in Hilbert space, expressed by the mean map $\mu_{\phi,t}$ and the covariance operator $\Sigma_{\phi,t}$, to an *a priori* belief of the succeeding data window, expressed by $\mu_{\phi,t+1}$ and $\Sigma_{\phi,t+1}$. We define a transition operator $\mathcal{C}_\phi$ that maps an embedding of the weights $\phi_t = \phi(\mathbf{w}_t)$ at time t to the successive embedding ϕ_{t+1} by

$$\phi_{t+1} = \mathcal{C}_\phi \phi_t + \varepsilon_C, \tag{4.37}$$

with a zero-mean Gaussian model error $\varepsilon_C \sim \mathcal{N}(0, \Sigma_{\mathcal{C}_\phi})$. We can obtain this embedding according to Equation (3.15) from the samples as

$$\mathcal{C}_\phi = \Phi_2(\Phi_1^\intercal \Phi_1 + \lambda I)^{-1}\Phi_1^\intercal, \tag{4.38}$$

where $\Phi_1 = [\phi(\mathbf{w}_1), \dots, \phi(\mathbf{w}_{N-1})]$ is the matrix of all feature mappings but the last and analog Φ_2 is the matrix of all feature mappings but the first, assuming that the weights $\mathbf{w}_1, \dots, \mathbf{w}_N$ belong to consecutive data windows.

The assumed Gaussian error on the transition model in Equation (4.37) implies a conditional Gaussian density over the succeeding embedding ϕ_{t+1} given the current embedding ϕ_t as $p(\phi_{t+1}|\phi_t) = \mathcal{N}(\phi_{t+1}|\mathcal{C}_\phi \phi_t, \Sigma_{\mathcal{C}_\phi})$. With the probability density over the embeddings given in Equation (4.31), we can then formulate the conditional probability density over the succeeding embedding ϕ_{t+1} given the a posteriori mean map $\mu_{\phi,t}$ and the a posteriori covariance operator $\Sigma_{\phi,t}$ by integrating out the current embedding ϕ_t as

$$p(\phi_{t+1}|\tilde{\mu}_{\phi,t+1}, \tilde{\Sigma}_{\phi,t+1}) = \int p(\phi_{t+1}|\phi_t)p(\phi_t|\mu_{\phi,t}, \Sigma_{\phi,t})\mathrm{d}\phi_t \tag{4.39}$$

$$= \int \mathcal{N}(\phi_{t+1}|\mathcal{C}_\phi \phi_t, \Sigma_{\mathcal{C}_\phi})\mathcal{N}(\phi_t|\mu_{\phi,t}, \Sigma_{\phi,t})\mathrm{d}\phi_t \tag{4.40}$$

$$= \mathcal{N}(\phi_{t+1}|\mathcal{C}_\phi \mu_{\phi,t}, \mathcal{C}_\phi \Sigma_{\phi,t} \mathcal{C}_\phi^\intercal + \Sigma_{\mathcal{C}_\phi}). \tag{4.41}$$

Hence, we obtain for the *a priori* successive mean map $\tilde{\mu}_{\phi,t+1} = \mathcal{C}_\phi \mu_{\phi,t} \in \mathbb{R}^{\infty\times 1}$ and for the *a priori* successive covariance operator $\tilde{\Sigma}_{\phi,t+1} = \mathcal{C}_\phi \Sigma_{\phi,t} \mathcal{C}_\phi^\intercal + \Sigma_{\mathcal{C}_\phi} \in \mathbb{R}^{\infty\times\infty}$.

4.3.3 Observation Model in Hilbert Space

Similar to the linear model, we update the *a priori* belief of the current state, expressed by $\tilde{\mu}_{\phi,t}$ and $\tilde{\Sigma}_{\phi,t}$, by conditioning on a new observation x_t. To perform this conditioning, we need a mapping from the RKHS-embedded representation to the mean $\mu_{x,t}$ and the variance $\sigma^2_{x,t}$ in state space, which represent our belief over the state x_t at the time of the current observation.

This mapping can be obtained as a composition of the conditional operator defined in Equation (4.32), which maps from the kernel space to the weight space, and of the basis function vector Ψ_0 that consists of the normalized magnitudes of all basis functions at the time of the new observation. As noted in Section 4.2.2, the basis function vector Ψ_0 is a row vector of the basis function matrix Ψ if the observation occurs at one of the predefined times of Ψ. We define

$$\xi := \Psi_0 \Lambda \tag{4.42}$$

as a shortcut of the mapping from the kernel space into the observation space (state space). The probability of observing an emission x_t given a feature vector ϕ_t becomes then

$$p(x_t|\phi_t) = \mathcal{N}(x_t|\xi\phi_t, \Psi_0\Sigma_\Lambda\Psi_0^\intercal + \Sigma_{\Psi_0}), \tag{4.43}$$

where Σ_{Ψ_0} is the model error of the basis function transformation at the time of the current observation. With the probability density over the embeddings formulated in Equation (4.31), we can define a joint probability density over the embeddings and the observations given the a priori mean map $\tilde{\mu}_{\phi,t}$ and the a priori covariance operator $\tilde{\Sigma}_{\phi,t}$ as well as the stationary quantities $\theta = [\Sigma_\Lambda, \Sigma_{\Psi_0}]$ by

$$\begin{align} p(\phi_t, x_t|\tilde{\mu}_{\phi,t}, \tilde{\Sigma}_{\phi,t}, \theta) &= p(x_t|\phi_t, \tilde{\mu}_{\phi,t}, \tilde{\Sigma}_{\phi,t}, \theta) p(\phi_t|\tilde{\mu}_{\phi,t}, \tilde{\Sigma}_{\phi,t}) \tag{4.44} \\ &= \mathcal{N}\left(\begin{array}{c|c} \phi_t & \tilde{\mu}_{\phi,t} \\ x_t & \xi\tilde{\mu}_{\phi,t} \end{array}, \begin{array}{cc} \tilde{\Sigma}_{\phi,t} & \tilde{\Sigma}_{\phi,t}\xi^\intercal \\ \xi\tilde{\Sigma}_{\phi,t} & \xi\tilde{\Sigma}_{\phi,t}\xi^\intercal + \Psi_0\Sigma_\Lambda\Psi_0^\intercal + \Sigma_{\Psi_0} \end{array} \right). \tag{4.45} \end{align}$$

Based on this joint distribution, we can obtain the conditional distribution of ϕ_t given the observation x_t by applying Bayes' rule and arrive at

$$\begin{align} p(\phi_t|x_t, \tilde{\mu}_{\phi,t}, \tilde{\Sigma}_{\phi,t}, \theta) &= \frac{p(\phi_t, x_t|\tilde{\mu}_{\phi,t}, \tilde{\Sigma}_{\phi,t}, \theta)}{p(x_t|\tilde{\mu}_{\phi,t}, \tilde{\Sigma}_{\phi,t}, \theta)} \tag{4.46} \\ &= \frac{p(\phi_t, x_t|\tilde{\mu}_{\phi,t}, \tilde{\Sigma}_{\phi,t}, \theta)}{\mathcal{N}(x_t|\xi\tilde{\mu}_{\phi,t}, \Psi_0\Sigma_\Lambda\Psi_0^\intercal + \Sigma_{\Psi_0})} \tag{4.47} \\ &= \mathcal{N}(\phi_t|\mu_{\phi,t}, \Sigma_{\phi,t}), \tag{4.48} \end{align}$$

with the *a posteriori* quantities $\mu_{\phi,t}$ and $\Sigma_{\phi,t}$ that are given by

$$\mu_{\phi,t} = \tilde{\mu}_{\phi,t} + \tilde{\Sigma}_{\phi,t}\xi^\intercal(\Psi_0\Sigma_\Lambda\Psi_0^\intercal + \Sigma_{\Psi_0} + \xi\tilde{\Sigma}_{\phi,t}\xi^\intercal)^{-1}(x_t - \xi\tilde{\mu}_{\phi,t}) \tag{4.49}$$

and

$$\Sigma_{\phi,t} = \tilde{\Sigma}_{\phi,t} - \tilde{\Sigma}_{\phi,t}\xi^\intercal(\Psi_0\Sigma_\Lambda\Psi_0^\intercal + \Sigma_{\Psi_0} + \xi\tilde{\Sigma}_{\phi,t}\xi^\intercal)^{-1}\xi\tilde{\Sigma}_{\phi,t}, \tag{4.50}$$

respectively.

4.3.4 Reasoning about Latent States

The model we described so far allows for predicting future states, filtering the current state or smoothing past states. However, there are many applications in which it is important to reason about latent states. Note that these latent states do not have to be the hidden states of the hidden Markov model depicted in Figure 4.2. Rather, a latent state may refer to any unobservable state of the system that correlates with the observations.

If we are given a time series labeled with latent states

$$\mathcal{D} = \begin{Bmatrix} x_1, \ldots, x_N \\ h_1, \ldots, h_N \end{Bmatrix}, \tag{4.51}$$

we can extend our model easily to provide additionally a conditional distribution over the latent states given the current belief of the data window. Similar to the conditional operator that maps from the RKHS back into the weight space (c.f. Equation (4.32)), we simply define an additional conditional operator that maps from the kernel space into the domain of the latent states by

$$\zeta := \mathbf{H}(\Phi^\intercal \Phi + \lambda I)^{-1} \Phi^\intercal, \tag{4.52}$$

where $\mathbf{H} = [h_1, \ldots, h_N]$ is the matrix of all latent states from the training data. As for the conditional operator Λ, we assume a zero-mean Gaussian error $\varepsilon_\zeta \sim \mathcal{N}(0, \Sigma_\zeta)$ that incorporates both the model error as well as the noise on the latent data. We can thus formulate a conditional probability density over the latent states given an embedding of the RKHS as

$$p(h_t | \phi_t) = \mathcal{N}(h_t | \zeta \phi_t, \Sigma_\zeta). \tag{4.53}$$

With the conditional probability density over the embeddings defined in Equation (4.31), we can then formulate a conditional probability density over the latent states given the current mean map $\mu_{\phi,t}$ and the current covariance operator $\Sigma_{\phi,t}$ by marginalizing out the embeddings ϕ_t as

$$\begin{aligned} p(h_t | \mu_{\phi,t}, \Sigma_{\phi,t}) &= \int p(h_t | \phi_t) p(\phi_t | \mu_{\phi,t}, \Sigma_{\phi,t}) \mathrm{d}\phi_t && (4.54) \\ &= \int \mathcal{N}(h_t | \zeta \phi_t, \Sigma_\zeta) \mathcal{N}(\phi_t | \mu_{\phi,t}, \Sigma_{\phi,t}) \mathrm{d}\phi_t && (4.55) \\ &= \mathcal{N}(h_t | \zeta \mu_{\phi,t}, \zeta \Sigma_{\phi,t} \zeta^\intercal + \Sigma_\zeta). && (4.56) \end{aligned}$$

A possible application of such a mapping to latent states could be, for example, to estimate a Cartesian point attractor to which a robot arm is moving from embedded data windows of joint states and velocities.

4.3.5 The RKHS Kalman Filter Algorithm

In this section, we will show how to employ the methodologies presented in the previous sections and condense them into an easy to implement algorithmic formulation. The RKHS embeddings and operators, namely the mean and covariance embedding and the conditional operators $\mathcal{C}_\phi$, Λ and ζ, are defined using feature matrices. However, the feature mappings are implicit to the kernel function and might be into an infinite domain. Hence, the methods presented so far are generally not directly computable. Nevertheless, it is possible to compute the statistics of the distributions we are interested in by employing the “kernel trick”, which results in simple Gram matrix manipulations.

The methods we described in the previous sections are all based on the application of conditional operators to embeddings. Common to all definitions of conditional operators (see Equations (4.32), (4.52) and (4.38)) is the transposed feature matrix $\Phi^\intercal$ on the right hand side. Hence, the embeddings to which the operators are applied are always multiplied with $\Phi^\intercal$ and it is sufficient to represent the embeddings in a subspace spanned by the feature mappings of the training samples. Thus, we obtain for the mean map

$$\mathbf{m} = \Phi^\intercal \mu_\phi = \frac{1}{N}\sum_{i=1}^{N} \Phi^\intercal \phi(\mathbf{w}_i) = \frac{1}{N}\Phi^\intercal \Phi \mathbf{1}_N = \frac{1}{N}\mathbf{K}\mathbf{1}_N, \tag{4.57}$$

with the kernel matrix $\mathbf{K}_{i,j} = k(\mathbf{w}_i, \mathbf{w}_j), i, j \in \{1, \ldots, N\}$ and the average kernel vector $\mathbf{m} \in \mathbb{R}^N$. The same holds for the covariance operator, where all operations involve the multiplication of $\Phi^\intercal$ from the left side and Φ from the right side (as we always apply also the transpose of the conditional operators from the right hand side). Hence, it is sufficient to represent the covariance operator by

$$\mathbf{S} = \Phi^\intercal \Sigma_\phi \Phi = \frac{1}{N}\sum_{i=1}^{N} \Phi^\intercal \hat{\phi}(\mathbf{w}_i)\hat{\phi}(\mathbf{w}_i)^\intercal \Phi \tag{4.58}$$

$$= \frac{1}{N}\Phi^\intercal \sum_{i=1}^{N} \left[(\phi(\mathbf{w}_i) - \mu_\phi)(\phi(\mathbf{w}_i) - \mu_\phi)^\intercal \right] \Phi \tag{4.59}$$

$$= \frac{1}{N}(\mathbf{K} \ominus \mathbf{m})(\mathbf{K} \ominus \mathbf{m})^\intercal, \tag{4.60}$$

where $\ominus$ denotes to column-wise subtraction. If we apply now the transition operator $\mathcal{C}_\phi$ to a mean embedding $\mu_{\phi,t}$ at time t, we get (neglecting the error term)

$$\mu_{\phi,t+1} = \mathcal{C}_\phi \mu_{\phi,t} \tag{4.61}$$

$$= \Phi_2(\Phi_1^\intercal \Phi_1 + \lambda \mathbf{I}_N)^{-1} \Phi_1^\intercal \mu_{\phi,t}. \tag{4.62}$$

By also representing the succeeding mean map $\mu_{\phi,t+1}$ in the subspace spanned by the embeddings of the training samples, we obtain a transition matrix $\mathbf{C}$ that maps $\mathbf{m}_t$ to $\mathbf{m}_{t+1}$

$$\underbrace{\Phi_1^\intercal \mu_{\phi,t+1}}_{\mathbf{m}_{t+1}} = \underbrace{\Phi_1^\intercal \Phi_2 (\Phi_1^\intercal \Phi_1 + \lambda \mathbf{I}_N)^{-1}}_{\mathbf{C}} \underbrace{\Phi_1^\intercal \mu_{\phi,t}}_{\mathbf{m}_t} \tag{4.63}$$

$$\mathbf{m}_{t+1} = \underbrace{\mathbf{K}_{1,2}(\mathbf{K} + \lambda \mathbf{I}_N)^{-1}}_{\mathbf{C}} \mathbf{m}_t. \tag{4.64}$$

Similar to the transition operator $\mathcal{C}_\phi$, the representation of the statistics using the basis spanned by the embedded training samples allows us to replace the conditional embedding operators Λ and ζ by the matrices $\mathbf{L}$ and $\mathbf{Z}$ as

$$\Lambda := \underbrace{\mathbf{W}(\Phi^\intercal \Phi + \lambda \mathbf{I}_N)^{-1}}_{\mathbf{L}} \Phi^\intercal \longrightarrow \mathbf{L} = \mathbf{W}(\mathbf{K} + \lambda \mathbf{I}_N)^{-1}, \tag{4.65}$$

$$\zeta := \underbrace{\mathbf{H}(\Phi^\intercal \Phi + \lambda \mathbf{I}_N)^{-1}}_{\mathbf{Z}} \Phi^\intercal \longrightarrow \mathbf{Z} = \mathbf{H}(\mathbf{K} + \lambda \mathbf{I}_N)^{-1}. \tag{4.66}$$

To learn our model, we consider a time series $D = [x_1, \ldots, x_N]$ as given, where we assume that the single data points are isochronous. The isochronal assumption allows us to efficiently compute the weight-based representation of the data windows with a single basis function matrix Ψ, which holds the normalized magnitudes at the regularly distributed times. It should be also possible to apply this

```
Algorithm 1: Learning the RKHS-embedded Kalman filter
input : The training data $D = [x_1, \dots, x_N]$, the window size $\tau$, the basis function matrix $\Psi$ and the
        kernel function $k$.
output: The model matrices $\mathbf{C}$, $\mathbf{L}$ and $\mathbf{Z}$ and the covariances of the model errors $\Sigma_\Psi$, $\Sigma_\mathbf{C}$, $\Sigma_\mathbf{L}$ and $\Sigma_\mathbf{Z}$.
Compute the hankel matrix with $\tau$ rows of the training inputs to obtain the data windows. This results
in $n = N - \tau + 1$ data windows.
$\mathcal{H}$ = hankel ($D$,$\tau$)
Compute the weight matrix using the basis function matrix $\Psi$.
$\mathbf{W} = (\Psi^\intercal \Psi)^{-1} \Psi^\intercal \mathcal{H}$
Estimate the model error of the basis function transformation from the reconstruction.
$\Sigma_\Psi = \mathrm{cov}\ (\mathcal{H} - \Psi \mathbf{W})$
Compute the kernel matrices $\mathbf{K}$ and $\mathbf{K}_2$.
$\mathbf{K} = k(\mathbf{W}_{1:n-1}, \mathbf{W}_{1:n-1})$,
$\mathbf{K}_{1,2} = k(\mathbf{W}_{1:n-1}, \mathbf{W}_{2:n})$,
where $\mathbf{X}_{i:j}$ denotes the sub-matrix of $\mathbf{X}$ consisting only of columns $i$ to $j$.
Compute the model matrices and estimate model errors from the reconstructions.
$\mathbf{C} = \mathbf{K}_{1,2}^\intercal (\mathbf{K} + \lambda_\mathbf{C} \mathbf{I}_n)^{-1}$          $\Sigma_\mathbf{C} = \mathrm{cov}\ (\mathbf{K}_{1,2} - \mathbf{C}\mathbf{K})$
$\mathbf{L} = \mathbf{W}_{1:n-1} (\mathbf{K} + \lambda_\mathbf{L} \mathbf{I}_n)^{-1}$       $\Sigma_\mathbf{L} = \mathrm{cov}\ (\mathbf{W}_{1:n-1} - \mathbf{L}\mathbf{K})$
if labels $\mathbf{H} = [h_1, \dots, h_n]$ are available then
    $\mathbf{Z} = \mathbf{H}_{1:n-1} (\mathbf{K} + \lambda_\mathbf{Z} \mathbf{I}_n)^{-1}$        $\Sigma_\mathbf{Z} = \mathrm{cov}\ (\mathbf{H}_{1:n-1} - \mathbf{Z}\mathbf{K})$
```

model to time series that are not sampled with a regular time interval, however, this would require a more sophisticated method to obtain the weight-based representation of the data windows since these representations have to be comparable.

In Algorithm 1, we show how to learn the RKHS-embedded Kalman filter based on a time series D, a given window size τ, a given basis function matrix Ψ and a kernel function k.

Algorithm 2 describes the computations that are necessary for the transition update and the conditioning on a new observation for a given belief in the RKHS, expressed by $\mathbf{m}_t$ and $\mathbf{S}_t$. Additionally, it depicts the mapping to the statistics of the data windows, expressed by $\mu_{x_t^\tau}$ and $\Sigma_{x_t^\tau}$, and the latent states, expressed by μ_{h_t} and Σ_{h_t}.

4.3.6 Learning with Large Training Data Sets

It is often the case that large training data sets, containing tens or hundreds of thousands data points, are available to learn a model. However, up-to-date consumer PCs do not allow for an efficient manipulation of Gram matrices built from data sets larger than a couple of thousands training points. In this section, we present two simple additions to our algorithm that allow to use the full information of a large data set, while preserving the sizes of the Gram matrices in a range such that efficient operations can be maintained. The first is the use of the *kernel data selection heuristic* to select a subset of the training data that is rich enough to describe the whole data set. The second are changes to the computation of the model matrices $\mathbf{C}$, $\mathbf{L}$ and $\mathbf{Z}$ that allow to train the model with the whole training data set.

Algorithm 2: Transition and observation update with mapping to the density over the data windows and the density over latent states

```
input  : A posteriori statistic at time $t$, expressed by $\mathbf{m}_t$ and $\mathbf{S}_t$.
output : A posteriori statistic at time $t+1$, expressed by $\mathbf{m}_{t+1}$ and $\mathbf{S}_{t+1}$, statistics in state space
         ($\mu_{x^{\tau}_{t+1}}$ and $\Sigma_{x^{\tau}_{t+1}}$), statistics in latent space ($\mu_{h_{t+1}}$ and $\Sigma_{h_{t+1}}$).
*Advancing the a posteriori belief at time t to the a priori belief at time t + 1.*
$\hat{\mathbf{m}}_{t+1} = \mathbf{C}\mathbf{m}_t$
$\hat{\mathbf{S}}_{t+1} = \mathbf{C}\mathbf{S}_t\mathbf{C}^\mathsf{T} + \Sigma_\mathbf{C}$
*Condition the a priori belief on an observation $x_{t+1}$ to obtain the a posteriori belief.*
$\mathbf{V} = \Psi_0\mathbf{L}\hat{\mathbf{S}}_{t+1}\mathbf{L}^\mathsf{T}\Psi_0^\mathsf{T} + \Psi_0\Sigma_\mathbf{L}\Psi_0^\mathsf{T} + \Sigma_{\Psi_0}$
$\mathbf{m}_{t+1} = \hat{\mathbf{m}}_{t+1} + \hat{\mathbf{S}}_{t+1}\mathbf{L}^\mathsf{T}\Psi_0^\mathsf{T}\mathbf{V}^{-1}(x_{t+1} - \Psi_0\mathbf{L}\hat{\mathbf{m}}_{t+1})$
$\mathbf{S}_{t+1} = \hat{\mathbf{S}}_{t+1} - \hat{\mathbf{S}}_{t+1}\mathbf{L}^\mathsf{T}\Psi_0^\mathsf{T}\mathbf{V}^{-1}\Psi_0\mathbf{L}\hat{\mathbf{S}}_{t+1}$
*Mapping to the density over the data windows.*
$\mu_{x^{\tau}_{t+1}} = \Psi\mathbf{L}\mathbf{m}_{t+1}$
$\Sigma_{x^{\tau}_{t+1}} = \Psi\mathbf{L}\mathbf{S}_{t+1}\mathbf{L}^\mathsf{T}\Psi^\mathsf{T} + \Psi\Sigma_\mathbf{L}\Psi^\mathsf{T} + \Sigma_\Psi$
*Mapping to the density over the latent states.*
$\mu_{h_{t+1}} = \mathbf{Z}\mathbf{m}_{t+1}$
$\Sigma_{h_{t+1}} = \mathbf{Z}\mathbf{S}_{t+1}\mathbf{Z}^\mathsf{T} + \Sigma_\mathbf{Z}$
```

Kernel Data Selection Heuristic

The *kernel data selection heuristic* is a simple heuristic to select relevant data points for the basis that we use to represent, for example, a mean mapping $\mu_{\phi,t}$ as $\mathbf{m}_t$ and a covariance operator $\Sigma_{\phi,t}$ as $\mathbf{S}_t$. We call this relevant data set the *kernel data set* in contrast to the whole *training data set*. The key idea of this heuristic is that we iteratively grow the kernel data set by adding data points that add the largest amount of information or, in other words, we add the data points about which the kernel data set that we have gathered so far can tell us least. As a measure for this lag of information we take the sum of the kernel activations of a data point with the current kernel data set. A condensed description of this heuristic is shown in Algorithm 3. We decided to always start with first element of the training data set as initialization for the kernel data set to make the heuristic deterministic, but a random initialization works as well.

Learning the Model Matrices from Large Data Sets

The goal of the second modification is to use not only the kernel data set selected by the heuristic presented in the previous section, but the whole training set with tens of thousands of data points to learn our model. To achieve this goal, we use modified kernel matrices to learn the linear mappings $\mathbf{C}$, $\mathbf{L}$ and $\mathbf{Z}$. Assuming we have the weight matrices $\mathbf{W}_\text{kernel} \in \mathbb{R}^{k\times M}$ of the kernel data and $\mathbf{W}_\text{train} \in \mathbb{R}^{k\times N}$ of the training data, instead of computing square kernel matrices solely from the kernel data, we build modified kernel matrices $\mathbf{K}'$ with $\mathbf{K}'_{i,j} = k(\mathbf{W}_\text{kernel}(i), \mathbf{W}_\text{train}(j)), i = 1,\ldots,M, j = 1,\ldots,N$, where $\mathbf{W}(i)$ is the i-th column of the respective weight matrix. Since the modified kernel matrices are not square, we resort to the Moore-Penrose pseudo-inverse for the computations of the model matrices $\mathbf{C}$, $\mathbf{L}$ and $\mathbf{Z}$. The learning of the model with these modified kernel matrices is depicted in Algorithm 4.

Algorithm 3: Kernel data selection heuristic

```
input  : Set of all weights $\mathbf{W} = \{\mathbf{w}_1, \dots, \mathbf{w}_N\}$, kernel size $M << N$, kernel function $k$.
output: Set of kernel weights $\mathbf{W}_K$.

Initialize the kernel weights set with the first weight vector.
$\mathbf{W}_K = [\mathbf{w}_1]$

Initialize a set of candidates for the kernel data set.
$\bar{\mathbf{W}} = \mathbf{W}/\mathbf{w}_1$

Grow the kernel data set until $M$ data points are gathered.
while $\mathtt{size}(\mathbf{W}_K) < M$ do
    Compute the kernel activations for all candidates.
    $\mathbf{a} = k(\mathbf{W}_K, \bar{\mathbf{W}})$

    Select the candidate with the smallest sum of kernel activations.
    $c = \mathbf{w}_i \in \bar{\mathbf{W}}$, where $\sum_j \mathbf{a}_{ji} < \sum_j \mathbf{a}_{jk} \forall k \neq i$, with $i, k \in \{1, \dots, |\bar{\mathbf{W}}|\}$ and $j = \{1, \dots, |\mathbf{W}_K|\}$

    Add the selected data point to the kernel data set and remove it from the candidates.
    $\mathbf{W}_K = \mathbf{W}_K \cap c$
    $\bar{\mathbf{W}} = \bar{\mathbf{W}}/c$
```

Algorithm 4: Learning the RKHS-embedded Kalman filter from large data sets

```
input  : Weight matrix $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_n]$, the kernel function $k$and the kernel size $M$.
output: The model matrices $\mathbf{C}$, $\mathbf{L}$ and $\mathbf{Z}$ and the covariances of the model errors $\Sigma_\Psi$, $\Sigma_\mathbf{C}$, $\Sigma_\mathbf{L}$ and $\Sigma_\mathbf{Z}$.

Select the kernel weights $\mathbf{W}_\mathbf{K}$ using the kernel data selection heuristic from Algorithm 3.
$\mathbf{W}_\mathbf{K} = \mathtt{kernel\_weights}(\mathbf{W}, k, M)$

Compute the kernel matrices $\mathbf{K}'$ and $\mathbf{K}'_2$.
$\mathbf{K}' = k(\mathbf{W}_\mathbf{K}, \mathbf{W}_{1:n-1})$,
$\mathbf{K}'_2 = k(\mathbf{W}_\mathbf{K}, \mathbf{W}_{2:n})$,
where $\mathbf{X}_{i:j}$ denotes the sub-matrix of $\mathbf{X}$ consisting only of columns $i$ to $j$.

Compute the model matrices and estimate the model errors from the reconstructions.
$\mathbf{C} = \mathbf{K}'_2\mathbf{K}'^\intercal(\mathbf{K}'\mathbf{K}'^\intercal + \lambda_\mathbf{C}\mathbf{I}_M)^{-1}$          $\Sigma_\mathbf{C} = \mathrm{cov}\,(\mathbf{K}'_2 - \mathbf{C}\mathbf{K}')$
$\mathbf{L} = \mathbf{W}_{1:n-1}\mathbf{K}'^\intercal(\mathbf{K}'\mathbf{K}'^\intercal + \lambda_\mathbf{L}\mathbf{I}_M)^{-1}$          $\Sigma_\mathbf{L} = \mathrm{cov}\,(\mathbf{W}_{1:n-1} - \mathbf{L}\mathbf{K}')$

if labels $\mathbf{H} = [h_1, \dots, h_n]$ are available then
    $\mathbf{Z} = \mathbf{H}_{1:n-1}\mathbf{K}'^\intercal(\mathbf{K}'\mathbf{K}'^\intercal + \lambda_\mathbf{Z}\mathbf{I}_M)^{-1}$          $\Sigma_\mathbf{Z} = \mathrm{cov}\,(\mathbf{H}_{1:n-1} - \mathbf{Z}\mathbf{K}')$
```

5 Experimental Results

To evaluate the performance of the RKHS-embedded Kalman filter and compare it to alternative models, we performed two experiments on artificial data from a simulated pendulum. With a first simple setup, we evaluated the performance of our model for one-step and long-term predictions of the angular position of a pendulum. For this experiment, we initialized the pendulum randomly near the upright position with a small random velocity. Subsequently, the pendulum fell either clockwise or counterclockwise and eventually came to a stop at the lower equilibrium. In a second experiment we rendered the oscillations of an undamped pendulum into video frames. For the training data, we altered the gravitational acceleration after every other swing and compared the performance of one-step predictions on evaluation data with previously seen and unseen gravitational accelerations to the performance of the Hilbert space-embedded spectral algorithm.

5.1 Pendulum Simulation

For our experiments we simulated a pendulum with unity rod length and unity mass using the well-known ordinary differential equation (ODE)

$$\ddot{\theta} = -\frac{g}{l}\sin(\theta) - a, \tag{5.1}$$

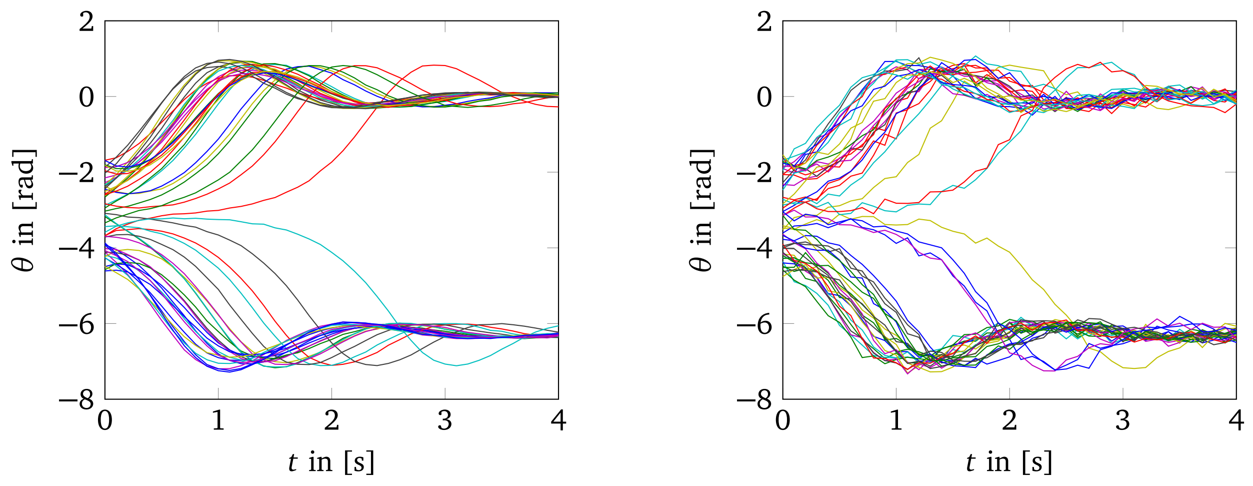
where θ is the angular displacement (in rad) with respect to the lower critical point (i.e., the pendulum hanging straight down corresponds to $\theta = 0$) and $\ddot{\theta}$ is the angular acceleration (in $\frac{\text{rad}}{\text{s}^2}$), g is the gravitational acceleration (in $\frac{\text{m}}{\text{s}^2}$) and l is the rod length (in m). We can further apply a torque to the pendulum that results in the angular acceleration a. This term allows, for example, to damp the pendulum or control it to given point attractors.

We integrated Equation (5.1) using Euler's method with a fixed time step of 0.001s. Optional system and observation noise terms were added during and after the integration, respectively. For our experiments we afterwards sampled the simulated time series down to 20 or 10 observations per second (i.e., time steps of 0.1 or 0.05 seconds).

5.2 One-Step and Long-Term Predictions of a 1D Pendulum State

To evaluate the performance of our model in doing one-step and long-term predictions, we chose a simple but profoundly non-linear pendulum setup. The pendulum was initialized randomly in the range from -1.5π to -0.5π with a random velocity of the range $-8\frac{\text{rad}}{s}$ to $+8\frac{\text{rad}}{s}$. The pendulum was damped with a damping factor of 2 and, additionally, system noise and observation noise were applied, both zero-mean Gaussian distributed with standard deviations $\sigma_{\text{sys}} = 0.05$ and σ_{obs}, respectively. We generated two data sets: a less noisy one with a standard deviation of $\sigma_{\text{obs}} = 0.01$ and a more noisy one with a standard deviation of $\sigma_{\text{obs}} = 0.1$. The data we obtained by integrating the pendulum's ODE in (5.1) consists of the pendulums angular displacement and its angular velocity. However, for the prediction tasks we reduced the data to consist only of the angular displacement.

During the simulation the pendulum fell either clockwise or counterclockwise and eventually came to a rest at $\theta = -2\pi$ or $\theta = 0$. We generated distinct data sets for learning the model, testing the performance during the optimization of the model parameters λ_C, λ_L and the kernel length scales, and for evaluating the model with the optimized parameters. Each training set consists of 50 sequences with



(a) Data set with observation noise $\sigma_{\text{obs}} = 0.01$. **(b)** Data set with observation noise $\sigma_{\text{obs}} = 0.1$.

Figure 5.1.: Training data sets for the evaluation of the RKHS-embedded Kalman filter, generated from a simulated pendulum. The pendulum was initialized randomly near the upright position with a random angular velocity. Both sets consist of 50 sequences covering 4 seconds with 41 data points per sequence. Both training data sets have the same system noise of $\sigma_{\text{sys}} = 0.05$.

41 data points per sequence. The training data sets with $\sigma_{\text{obs}} = 0.01$ and $\sigma_{\text{obs}} = 0.1$ are depicted in Figures 5.1a and 5.1b, respectively.

We learned our model with a window size of $\tau = 4$ and the Dirac basis functions, i.e., we used directly the data windows instead of a weight-based representation. As kernel function we employed the Gaussian-PCA kernel function

$$k_{\text{gaussPCA}}(\mathbf{w}, \mathbf{w}') = \exp\left(-\frac{1}{2}(\mathbf{w} - \mathbf{w}')^{\mathsf{T}}\mathbf{V}\mathbf{Q}\mathbf{V}^{\mathsf{T}}(\mathbf{w} - \mathbf{w}')\right), \tag{5.2}$$

where $\mathbf{V}$ is the basis transformation obtained by a principal component analysis (PCA) of the training weights $\mathbf{W}$ and $\mathbf{Q}$ is a diagonal matrix containing the length-scales for each dimension. We optimized the regularization parameters λ_{C} and λ_{L} as well as the length-scales in $\mathbf{Q}$ using the *Evolution Strategy with Covariance Matrix Adaptation (CMA-ES)* method (Hansen and Ostermeier, 2001).

Model evaluation heuristics for an efficient optimization of the model parameters.
As a gradient-free optimization method, CMA-ES needs to evaluate the model for numerous different parameter sets (e.g., for 8 parameters CMA-ES generates 9 parameter sets per iteration, where several hundreds of iterations are required). Since already the evaluation of a single sequence of 41 data points takes in the range of a second, evaluating the model using the update cycle presented in Sections 4.3.2 and 4.3.3 and all 50 sequences would render the optimization infeasible. Instead, we resorted to two alternative approaches for measuring the performance of our model.

The first approach, which we call *one-step evaluation*, computes the kernel-embeddings $\bar{\mathbf{m}}_{1:T-1} = k(\mathbf{W}_{\text{K}}, \bar{\mathbf{w}}_{1:T-1})$ of all but the last weights of the data windows of a full sequence, where k is the kernel function, $\mathbf{W}_{\text{K}}$ are the kernel weights from the model, and $\bar{\mathbf{w}}_i$ are the weights from the testing sequence for the optimization. The one-step approach evaluates the model then based on the one-step prediction of the embeddings $\bar{\mathbf{m}}_{2:T} = \mathbf{C}\bar{\mathbf{m}}_{1:T-1}$.

The second approach computes only the kernel-embedding $\bar{\mathbf{m}}_1 = k(\mathbf{W}_{\text{K}}, \bar{\mathbf{w}}_1)$ of the weights of the first data window of a sequence and evaluates the model then based on the long-term predictions, which are obtained by iteratively applying the transition model $\bar{\mathbf{m}}_{t+1} = \mathbf{C}\bar{\mathbf{m}}_t$. We call this approach the *long-term evaluation*. Both of these approaches leverage the optimization process with a significant reduction of processing time per evaluation. Here, the omission of large matrix multiplications, e.g., the transition update of the covariance operator (c.f. Algorithm 2), contributes the most to the increase of speed.

Initial values for the mean embedding and the covariance operator.
We further compared two different methods, A and B, to obtain an initial mean embedding $\mathbf{m}_1$ and an initial covariance operator $\mathbf{S}_1$. Both methods compute the initial covariance embedding $\mathbf{S}_1$ from the first data windows of all training sequences, similar to Equation (4.30). Method A computes also the initial mean embedding $\mathbf{m}_1$ from the first data windows of all training sequences, whereas the second method B uses the embedding $\tilde{\mathbf{m}}_1 = k(\mathbf{W}_\mathbf{K}, \tilde{\mathbf{w}}_1)$ of the weights $\tilde{\mathbf{w}}_1$ of the first data window of the evaluation sequence as initialization.

Impact of noise on the training data.
Finally, we compared the performance of our model when trained with noisy data against the performance when trained with smoothed data. We obtained the noise-free data directly from the simulation, in addition to the noisy data. However, for real environments a smoothed version of the noisy data should be a sufficient replacement.

5.2.1 Performance Results of One-Step and Long-Term Predictions

The results in Table 5.1 were obtained by performing one-step predictions, which means that we initialized the RKHS-embedded Kalman filter using one of the above proposed methods and iteratively applied the transition model and conditioned on the new observation. In each iteration, we conditioned the most recent entry of the data window on the new observation and applied the transition operator in Hilbert space to advance the belief to the next state. After the transition update, we compared the most recent entry of the data window to the corresponding true (i.e., noise-free) value.

The values in Table 5.1a were obtained by learning, optimizing and evaluating our model on data sets with an observation noise of $\sigma_{\text{obs},1} = 0.01$. The training data set is depicted in Figure 5.1a. The values in Table 5.1b were obtained by learning, optimizing and evaluating our model on data sets with an observation noise of $\sigma_{\text{obs},2} = 0.1$. The corresponding training data set is depicted in Figure 5.1b. We measured the performance by computing the root-mean-squared-error (RMSE) to the true data.

Likewise, the results in Table 5.2 were obtained by performing long-term predictions on the same evaluation data sets that we used for the one-step predictions. In contrast to the one-step prediction experiment, we observed the current emissions only during the first second of each sequence. For the last three seconds of the sequence we performed the prediction by solely applying the transition operator in Hilbert space.

Comparing one-step evaluation to long-term evaluation.
For the less noisy data sets, the models optimized with the long-term evaluation method yielded in both experiments generally better results than the models optimized with the one-step evaluation method. This result is quite intuitive since a model optimized for long-term predictions should perform at least equally well as a model optimized for one-step predictions. However, for the more noisy data set it is the other way around for the one-step prediction experiment and no obvious tendency can be encountered in the long-term prediction experiment. A possible explanation for this result is that the optimization of the model parameters often gets stuck in local optima that are, in terms of performance, far away from the global optimum.

Comparing the initialization methods.
If comparing the results obtained on the less noisy data sets, it is generally the case that initialization method B yields the better results. This is also quite intuitive, as the mean embedding of all initial training data windows is in most cases a worse approximation to the true embedding than the embedding

evaluation heuristic	long-term evaluation				one-step evaluation			
training data	noisy		smooth		noisy		smooth	
initialization method	A	B	A	B	A	B	A	B
RMSE [rad]	0.0255	0.0254	0.0247	0.0208	0.0316	0.0231	0.0301	0.0227

(a) Performance of the RKHS-KF on evaluation data with an observation noise standard deviation of $\sigma_{\text{obs}} = 0.01$.

evaluation heuristic	long-term evaluation				one-step evaluation			
training data	noisy		smooth		noisy		smooth	
initialization method	A	B	A	B	A	B	A	B
RMSE [rad]	0.1299	0.1807	0.1913	0.2121	0.1307	0.1079	0.1231	0.1008

(b) Performance of the RKHS-KF on evaluation data with an observation noise standard deviation of $\sigma_{\text{obs}} = 0.1$.

Table 5.1.: One-step prediction performance of the RKHS-embedded Kalman filter, trained with noisy and smooth training data, using the evaluation and initialization methods presented in Section 5.2, evaluated on two data sets with a different amount of observation noise. As performance measure the root-means-squared-error (RMSE) of the predictions to the noiseless data has been used.

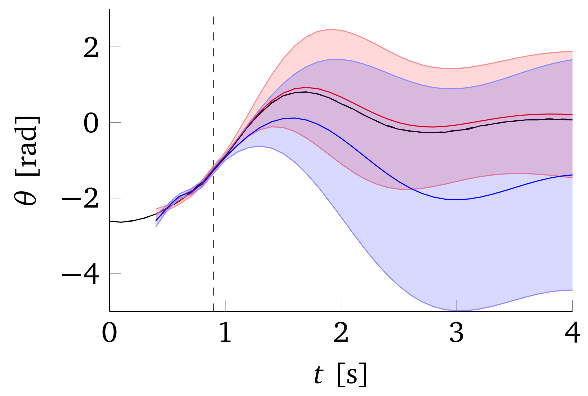
evaluation method	long-term evaluation				one-step evaluation			
training data	noisy		smooth		noisy		smooth	
initialization method	A	B	A	B	A	B	A	B
RMSE [rad]	0.6621	0.5008	0.6587	0.5272	0.9314	0.7261	0.9172	0.7145

(a) Performance of the RKHS-KF on evaluation data with an observation noise standard deviation of $\sigma_{\text{obs}} = 0.01$.

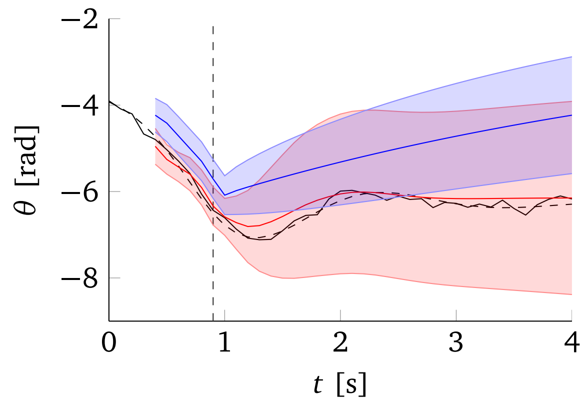
evaluation method	long-term evaluation				one-step evaluation			
training data	noisy		smooth		noisy		smooth	
initialization method	A	B	A	B	A	B	A	B
RMSE [rad]	0.9511	0.5497	1.5021	2.4974	0.9577	0.7254	0.8783	0.6232

(b) Performance of the RKHS-KF on evaluation data with an observation noise standard deviation of $\sigma_{\text{obs}} = 0.1$.

Table 5.2.: Long-term prediction performance of the RKHS-embedded Kalman filter, trained with noisy and smooth training data, using the evaluation and initialization methods presented in Section 5.2, evaluated on two data sets with a different amount of observation noise. As performance measure the root-means-squared-error (RMSE) of the predictions to the noiseless data has been used.



(a) Example of a long-term prediction experiment on the data set with $\sigma_{\text{obs}} = 0.01$.



(b) Example of a long-term prediction experiment on the data set with $\sigma_{\text{obs}} = 0.1$.

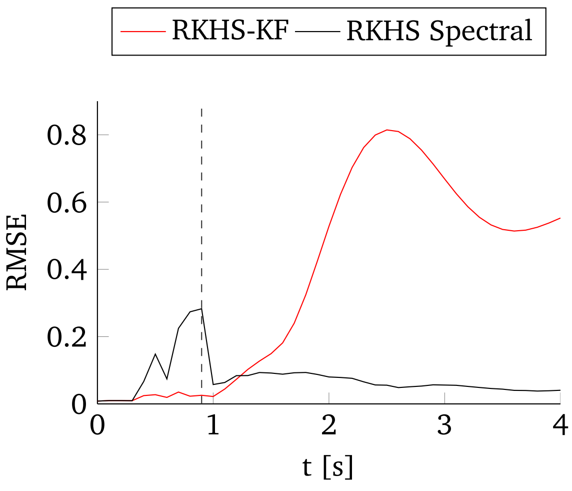
Figure 5.2.: Examples of a long-term prediction experiment. The true data is plotted as a dashed black line and the noisy data is plotted as a solid black line. The predictions of the kernel model are plotted in red with a 2σ boundary. Likewise, the predictions of the linear model are plotted in blue also with a 2σ boundary. Both models observed the noisy state until $t = 0.9$s (indicated by the gray dashed line). Afterwards the belief is updated by solely applying the transition model.

of the initial data window of the evaluation sequence. For the more noisy data, however, initialization method B provided not always a better performance than method A. This might again be caused by a poor local optimum, but also the larger difference between true and noisy data windows might contribute to this result.

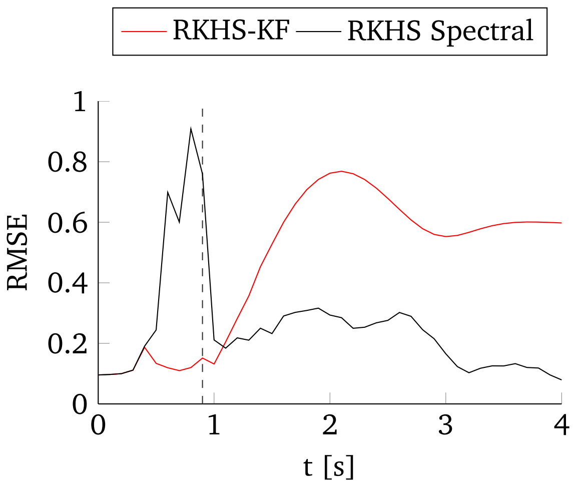
Comparing the impact of noisy training data on the performance.
Another observation is that the model generally benefits from being trained on smooth data. A reason for this might be the use of Dirac basis functions, which have, in contrast to the radial basis functions, no inherent smoothing mechanism.

Comparing the RKHS-embedded Kalman filter to the linear weight-based approach.
In Figure 5.2, two examples of long-term prediction experiments are depicted. Figure 5.2a shows an experiment on the less noisy data set with $\sigma_{\text{obs}} = 0.01$ and Figure 5.2b shows an experiment on the more noisy data set with $\sigma_{\text{obs}} = 0.1$. The models used for these examples have been learned with noisy training data and the model parameters have been optimized with the long-term evaluation heuristic. Note that these are examples on which the RKHS-embedded Kalman filter yielded fairly good results. Though, our model also clearly outperformed the linear model even on sequences on which it performed worse than in this example.

Comparing the RKHS-embedded Kalman filter to the kernel spectral algorithm.
We compared the long-term prediction performance of our model to the kernel spectral algorithm, where we used 100 eigenvectors to compose the subspace projection $\mathbf{U}$ (c.f. Section 3.2.2). We trained our model with noisy data, optimized the model parameters using the long-term evaluation heuristic and used initialization method B. Both models observed the noisy emissions until $t = 0.9$s. Afterwards, the RKHS-embedded Kalman filter updates its belief solely by applying the transition operator, whereas the kernel spectral algorithm uses the predicted state as new observation. Figure 5.3 depicts the root-mean-squared-error (RMSE) of both models to the true data. It can be seen that our model performs better when conditioned on the observations (from $t = 0$s until $t = 0.9$s), but also that the kernel spectral algorithm leads to much better results if no observations are available.



(a) RMSE of long-term predictions on the data set with $\sigma_{\text{obs}} = 0.01$.



(b) RMSE of long-term predictions on the data set with $\sigma_{\text{obs}} = 0.1$.

Figure 5.3.: Comparison of the RKHS-embedded Kalman filter to the kernel spectral algorithm. The models observe the first ten states until $t = 0.9$s (indicated by the gray dashed line) and predicted the following states from $t = 1$s until $t = 4$s. The results were averaged over 50 episodes.

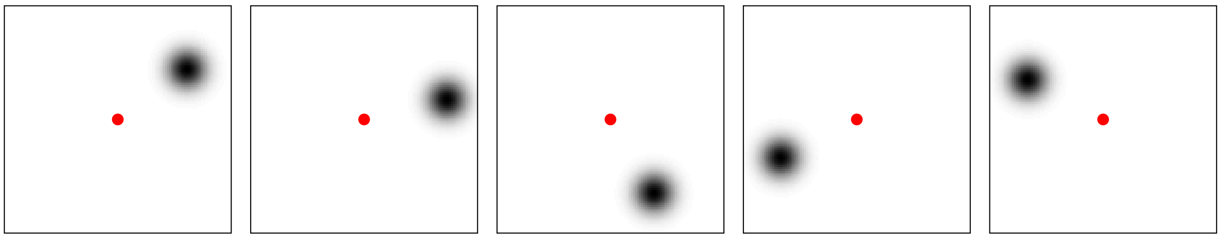

(a) $t = 1$ **(b)** $t = 6$ **(c)** $t = 11$ **(d)** $t = 16$ **(e)** $t = 21$

Figure 5.4.: Video frames generated from the pendulum data at times $t = 1, 6, 11, 16, 21$. The video frames have a width and a height of 120 pixels. The pivot point of the pendulum is in the center of the frame (red circle, not present in the video data). The pendulum has been rendered with a radial basis function around its center using a length scale of 50 pixels. The values are scaled from 0 (white) to 255 (black).

5.3 Predicting Video Frames

In a second experiment we evaluated our model on high-dimensional data. As experiment setup we chose again the pendulum presented in Section 5.1. We initialized the pendulum at a fixed angle $\theta_0 = 0.7\pi$ and simulated the undamped movements for 60 seconds with system noise $\sigma_{\text{sys}} = 0.0005$ and observations noise $\sigma_{\text{obs}} = 0.001$. We rendered the pendulum movements into square video frames with a width of 120 pixels. Five examples of these video frames are depicted in Figure 5.4. We generated two data sets that incorporated swings under the gravitational accelerations $g_1 = 5\frac{\text{m}}{\text{s}^2}$, $g_2 = 10\frac{\text{m}}{\text{s}^2}$ and $g_2 = 15\frac{\text{m}}{\text{s}^2}$, where we altered the gravitational acceleration after every other full swing. Additionally, to demonstrate the ability of our model to previously unseen data, we generated a third data set under the gravitational acceleration $g_4 = 7\frac{\text{m}}{\text{s}^2}$.

To learn our model, we treated each video frame as a vector consisting of the concatenated columns and reduced the dimensionality of these vectors from 14400 to 30 by performing a principal component analysis (PCA). We employed the Dirac basis functions and used windows of four downscaled frame-vectors. As kernel function, we used the Gaussian kernel with a kernel size of 300 samples. The

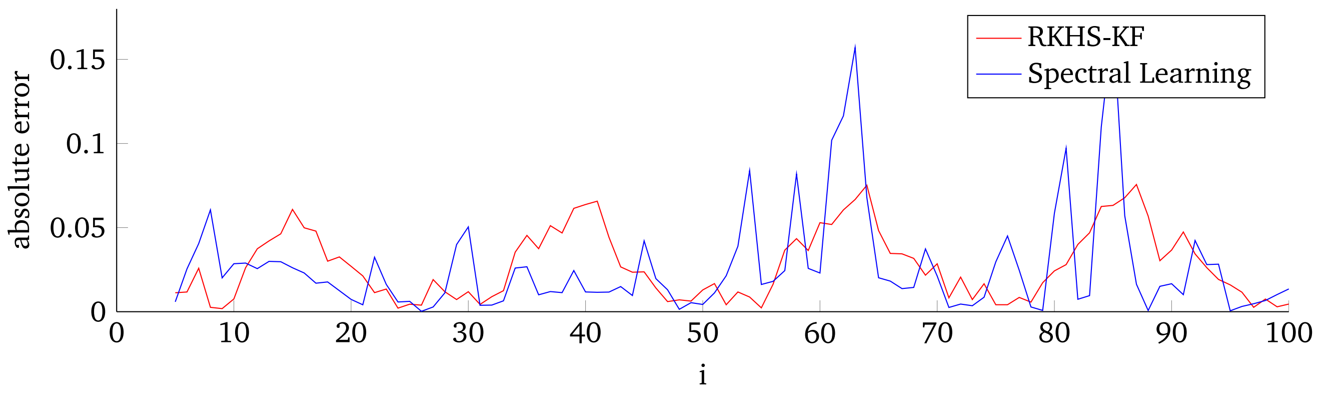


Figure 5.5.: Absolute error of one-step predictions of video frames, where the gravitational accelerations were present in the training data. The figure depicts two full swings with $g = 15\frac{\text{m}}{\text{s}^2}$. It can be seen that the RKHS-embedded Kalman filter (red) maintains a similar magnitude of the error for every swing, while the error of the RKHS-embedded spectral learning method (blue) differs significantly.

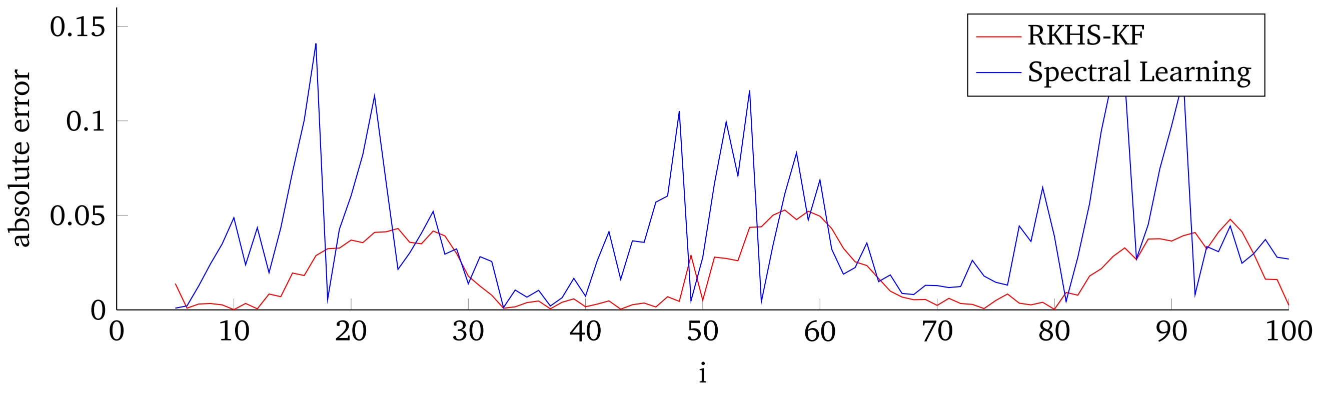


Figure 5.6.: Absolute error of one-step predictions of video frames, with a gravitational acceleration of $g = 7\frac{\text{m}}{\text{s}}$, which was not present in the training data. The figure depicts approximately 1.5 full swings. The RKHS-embedded Kalman filter (red) maintains an error magnitude similar to the evaluation on known data, whereas the RKHS-embedded spectral learning method (blue) performs worse.

regularization parameters λ_C and λ_L were each set to 10^{-4}. We tuned the length-scale parameter of the kernel function, using a multiplicative of the mean difference of successive data windows until we got reasonable kernel activations in the matrix $\mathbf{K}$.

We compared our model to the kernel spectral algorithm trained with a subspace projection $\mathbf{U}$ consisting of 100 eigenvectors. To compare the resulting video frames to the true data obtained from the pendulum simulator, we extracted the angular displacement of the pendulum from the video frames with a weighted sum of the pixel locations using the pixel intensities as weights. Figure 5.5 shows the absolute error of one step predictions on evaluation data that was simulated with a gravitational acceleration of $15\frac{\text{m}}{\text{s}^2}$, which was also present in the training data. Depicted are approximately two full swings, i.e., two swings to the left and two swings to the right. The error of both models is higher when the pendulum has a higher angular velocity. From this plot, it can be seen that our model already generalizes better for data seen during learning, as it maintains an similar error for both full swings, whereas the kernel spectral algorithm has a lower error for the first full swing and a higher error for the second one.

Figure 5.6 depicts the absolute error of one step predictions on evaluation data that was simulated with a gravitational acceleration of $7\frac{\text{m}}{\text{s}^2}$, which was not present in the training data. It can be clearly seen that the kernel spectral algorithm does not generalize well to this unknown data, while the RKHS-embedded Kalman filter has an error of the same magnitude as for the known data.

6 Conclusions

In this thesis we proposed a novel way to extend the applicability of hidden Markov models to continuous, non-linear dynamical systems. We built on the well established concept of Kalman filters, which we employed on a probabilistic representation of state, expressed by a Gaussian random variable in Hilbert space. We provided a Hilbert space analog to the transition model and showed how to condition the RKHS embedding of the Gaussian probability density over the state on new observations in state space. In addition, the Hilbert space embedded Kalman filter allows also to reason about latent states, if a labeled data set is available for training. In a condensed algorithmic formulation, our method consists mainly of simple and easy to implement Gram matrix manipulations. We further provided an extension to our algorithm that allows to train the Hilbert space embedded Kalman filter using large data sets without rendering the Gram matrix manipulations infeasible.

The evaluations show that our model scales with high dimensional data and provides a good generalization to data that was not represented in the training set. However, the evaluations also show that we still need to work on the optimization process of the model parameters, as the heuristics we used for faster evaluations are likely to emphasize on the performance of the transition model, but do not represent the conditioning on new observations properly. Likewise, the we need to address the problem of poor local optima, in which the optimization of the model parameters gets stuck, by finding better initial values or running the optimization from many different starting points.

Furthermore, embedding also the conditioning on new observations into Hilbert space by making use of the kernel Bayes' rule might also yield better performances. Moreover, performing fully Bayesian inference entirely in Hilbert space would render the Gaussian assumption in Hilbert space obsolete and, thus, allow for an extension to arbitrary probability distributions.

References

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404.

Baker, C. R. (1973). Joint measures and cross-covariance operators. *Transactions of the American Mathematical Society*, 186:273–289.

Barber, D. (2012). *Bayesian reasoning and machine learning*. Cambridge University Press, Cambridge.

Boots, B., Gretton, A., and Gordon, G. J. (2013). Hilbert space embeddings of predictive state representations. In *Proceedings of the 29th International Conference on Uncertainty in Artificial Intelligence (UAI)*.

Deisenroth, M. P., Fox, D., and Rasmussen, C. E. (2014). Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(99):1–1.

Grünewälder, S., Lever, G., Baldassarre, L., Patterson, S., Gretton, A., and Pontil, M. (2012). Conditional mean embeddings as regressors. In *Proceedings of the 29th International Conference on Machine Learning*.

Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–95.

Haussler, D. K. D. and Eeckman, M. G. R. F. H. (1996). A generalized hidden markov model for the recognition of human genes in dna. *Proc. Int. Conf. on Intelligent Systems . . .* , pages 134–142.

Hesse, T. (2014). Spectral learning of hidden markov models.

Hsu, D., Kakade, S. M., and Zhang, T. (2012). A spectral algorithm for learning hidden markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480.

Jaeger, H. (2000). Observable operator models for discrete stochastic time series. *Neural Computation*, 12(6):1371–1398.

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks. Technical report, German National Research Center for Information Technology.

Julier, S. J. and Uhlmann, J. K. (1997). A new extension of the kalman filter to nonlinear systems. In *Int. symp. aerospace/defense sensing, simul. and controls*, volume 3.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45.

Littman, M. L., Sutton, R. S., and Singh, S. (2002). Predictive representations of state. In *Advances in Neural Information Processing Systems*, pages 1555–1561. MIT Press.

Lukoševičius, M. and Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149.

Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560.

Murphy, K. P. (2012). *Machine Learning: a Probabilistic Perspective*. MIT Press.

Paraschos, A., Daniel, C., Peters, J., and Neumann, G. (2013). Probabilistic movement primitives. *Advances in Neural Information Processing Systems*, pages 2616–2624.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286.

Smola, A. J., Gretton, A., Song, L., and Schölkopf, B. (2007). A hilbert space embedding for distributions. In *Proceedings of the 18th international conference on Algorithmic Learning Theory*, volume 4754 of *Lecture Notes in Computer Science*, pages 13–31, Berlin, Heidelberg. Springer Berlin Heidelberg.

Song, L., Boots, B., Siddiqi, S. M., Gordon, G. J., and Smola, A. J. (2010). Hilbert space embeddings of hidden markov models. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 991–998.

Song, L., Fukumizu, K., and Gretton, A. (2013). Kernel embeddings of conditional distributions: A unified kernel framework for nonparametric inference in graphical models. *IEEE Signal Processing Magazine*, 30(4):98–111.

Song, L., Huang, J., Smola, A., and Fukumizu, K. (2009). Hilbert space embeddings of conditional distributions with applications to dynamical systems. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pages 1–8, New York, New York, USA. ACM Press.

Sriperumbudur, B. K., Gretton, A., Fukumizu, K., Lanckriet, G., and Schölkopf, B. (2008). Injective hilbert space embeddings of probability measures. In *Proceedings of the 21st Annual Conference on Learning Theory (COLT 2008)*, pages 111–122.

Vempala, S. and Wang, G. (2002). A spectral algorithm for learning mixtures of distributions. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 113–122. IEEE Comput. Soc.

Wan, E. and Van Der Merwe, R. (2000). The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pages 153–158. IEEE.

The plots in this thesis were generated with MATLAB R2011a and the script “matlab2tikz” by Nico Schlömer, which can be found at `https://github.com/nschloe/matlab2tikz`. For the plots in Figure 5.2 we used the MATLAB script “shadedErrorBar” by Rob Campbell, which can be found at `http://www.mathworks.com/matlabcentral/fileexchange/26311-shadederrorbar`.

A Appendix

A.1 Derivation of the Observable Parameter β_∞ in Section 3.2.2

From Equation (3.26), we obtain the definition of β_∞ in terms of the subspace projection $\mathcal{U}$ and the joint embedding $\mathcal{C}_{2,1}$ as

$$\beta_\infty := \mathcal{C}_{2,1}\left(\mathcal{U}^\intercal \mathcal{C}_{2,1}\right)^\dagger, \tag{A.1}$$

where $X^\dagger$ is the Moore-Penrose pseudo-inverse of X. We can apply the definition of the right pseudo-inverse, which is $X^\dagger = X^\intercal(XX^\intercal)^{-1}$, and obtain

$$\begin{aligned}\beta_\infty &= \mathcal{C}_{2,1}\left(\mathcal{U}^\intercal \mathcal{C}_{2,1}\right)^\intercal\left(\mathcal{U}^\intercal \mathcal{C}_{2,1}\left(\mathcal{U}^\intercal \mathcal{C}_{2,1}\right)^\intercal\right)^{-1} &\text{(A.2)}\\ &= \mathcal{C}_{2,1}\mathcal{C}_{2,1}^\intercal\mathcal{U}\left(\mathcal{U}^\intercal\mathcal{C}_{2,1}\mathcal{C}_{2,1}^\intercal\mathcal{U}\right)^{-1}. &\text{(A.3)}\end{aligned}$$

We can now put in the estimates of the joint embedding $\mathcal{C}_{2,1}$ given in Equation (3.31) and of the subspace projection given in Equation (3.37) and obtain

$$\begin{aligned}\beta_\infty &= \overbrace{\Phi\Upsilon^\intercal}^{\mathcal{C}_{2,1}}\,\overbrace{\Upsilon\Phi^\intercal}^{\mathcal{C}_{2,1}^\intercal}\,\overbrace{\Phi AD}^{\mathcal{U}}\left(\overbrace{D^\intercal A^\intercal\Phi^\intercal}^{\mathcal{U}^\intercal}\,\overbrace{\Phi\Upsilon^\intercal}^{\mathcal{C}_{2,1}}\,\overbrace{\Upsilon\Phi^\intercal}^{\mathcal{C}_{2,1}^\intercal}\,\overbrace{\Phi AD}^{\mathcal{U}}\right)^{-1} &\text{(A.4)}\\ &= \Phi\,\underbrace{\Upsilon^\intercal\Upsilon}_{K}\,\underbrace{\Phi^\intercal\Phi}_{L}\,AD\left(D^\intercal A^\intercal\,\underbrace{\Phi^\intercal\Phi}_{L}\,\underbrace{\Upsilon^\intercal\Upsilon}_{K}\,\underbrace{\Phi^\intercal\Phi}_{L}\,AD\right)^{-1} &\text{(A.5)}\\ &= \Phi KLAD\left(D^\intercal A^\intercal LKLAD\right)^{-1}. &\text{(A.6)}\end{aligned}$$

From Equation (3.36) we can use the identity $LKLA = LA\Omega$, with $\Omega := \operatorname{diag}(w_1, \ldots, w_N)$ and get

$$\beta_\infty = \Phi KLAD\left(D^\intercal A^\intercal LA\Omega D\right)^{-1}. \tag{A.7}$$

The next step is to employ the definition of the diagonal matrix D which normalizes the singular vectors v_i, that is

$$\begin{aligned}D &:= \operatorname{diag}\left((\|v_1\|_2)^{-1}, \ldots, (\|v_N\|_2)^{-1}\right) &\text{(A.8)}\\ &= \operatorname{diag}\left(\left(v_1^\intercal v_1\right)^{-\frac{1}{2}}, \ldots, \left(v_N^\intercal v_N\right)^{-\frac{1}{2}}\right) &\text{(A.9)}\\ &= \operatorname{diag}\left(\left(\alpha_1^\intercal\Phi^\intercal\Phi\alpha_1\right)^{-\frac{1}{2}}, \ldots, \left(\alpha_N^\intercal\Phi^\intercal\Phi\alpha_N\right)^{-\frac{1}{2}}\right) &\text{(A.10)}\\ &= \operatorname{diag}\left(\left(\alpha_1^\intercal L\alpha_1\right)^{-\frac{1}{2}}, \ldots, \left(\alpha_N^\intercal L\alpha_N\right)^{-\frac{1}{2}}\right) &\text{(A.11)}\\ &= (A^\intercal LA)^{-\frac{1}{2}}, &\text{(A.12)}\end{aligned}$$

where the facts that the eigenvectors α_i form an orthonormal eigen-basis, i.e., $\alpha_i^\intercal\alpha_j = 0$ for all $i, j \in \{1, \ldots, N\}$ with $i \neq j$, and that the inverse of an diagonal matrix is the inverse of its elements (and vice versa), allow the transition from (A.11) to (A.12). Equation (A.12) allows us to replace $A^\intercal LA$ with D^{-2} and together with the commutativity of diagonal matrices (i.e., $\Omega D = D\Omega$) we obtain

$$\begin{aligned}\beta_\infty &= \Phi KLAD\left(D^\intercal D^{-2}D\Omega\right)^{-1} &\text{(A.13)}\\ &= \Phi KLAD\Omega^{-1}. &\text{(A.14)}\end{aligned}$$