Spectral Learning of Hidden Markov Models

Spektrales Lernen von Hidden Markov Modellen
Bachelor-Thesis von Thomas Hesse aus Frankfurt am Main
März 2014

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Spectral Learning of Hidden Markov Models
Spektrales Lernen von Hidden Markov Modellen

Vorgelegte Bachelor-Thesis von Thomas Hesse aus Frankfurt am Main

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Dr. Heni Ben Amor
3. Gutachten: Dr. Gerhard Neumann

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter und 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.
In der abgegebenen Thesis stimmen die schriftliche und elektronische Fassung überein.

Darmstadt, den 4. März 2014

(Thomas Hesse)

Thesis Statement

I herewith formally declare that I have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.
In the submitted thesis the written copies and the electronic version are identical in content.

Darmstadt, March 4, 2014

(Thomas Hesse)

Abstract

The Hidden Markov Model is the most fundamental model of partially observable uncontrolled systems in Machine Learning. The *state of the art* over the last years was the Baum-Welch algorithm which derives from the Expectation Maximization algorithm. However, this algorithm is very computationally intensive and is only guaranteed to find a locally optimal solution. An alternative algorithm to the Baum-Welch algorithm is the so called Spectral Learning algorithm. This algorithm uses spectral methods to exploit the hidden relations between successive observable output. The Spectral Learning algorithm does not learn the Hidden Markov Model (HMM) directly for this purpose but rather a transformed spectral model based on an observable representation. Finding a global solution is guaranteed by the algorithm. Spectral Learning works best in sceanrios with a lot of data. This work evaluates Spectral Learning by learning a grid world modeled by a HMM. The grid world is supposed to be an abstract highway. We successfully modeled the driving behavior of a car driver in this grid world. Finally, we successfully imitated one learned behavior for a scenario that is common for the behavior to imitate.

Zusammenfassung

Das Hidden Markov Model ist ein fundamentales Modell für teilweise beobachtbare und unkontrollierte Systeme in Machine Learning. Der *Stand der Technik* der letzten Jahre ist der Baum-Welch Algorithmus gewesen, welcher sich von dem Expectation Maximization Algorithmus ableitet. Dieser Algorithmus ist jedoch sehr rechenintensiv und garantiert keine globale Lösung, da er nur gegen lokale Lösungen konvergiert. Eine Alternative zu dem Baum-Welch Algorithmus ist der sogenannte Spectral Learning Algorithmus. Dieser benutzt spektrale Methoden um die versteckten Relationen zwischen aufeinanderfolgenden Ausgaben zu identifizieren. Der Spectral Learning Algorithmus lernt für diesen Zweck nicht das echte Hidden Markov Modell, sondern eine transformierte Darstellung des eigentlichen Hidden Markov Modell. Außerdem ist es garantiert eine globale Lösung zu finden. Spektrales Lernen kommt am besten zum Einsatz in Szenarien in denen viele Daten vorliegen. In dieser Arbeit wird untersucht wie sich der Spectral Learning Algorithmus für das Lernen einer Gitterwelt verhält. Hierbei wird die Gitterwelt als abstraktes System für eine Autobahn angenommen. Anschließend wurde erfolgreich versucht mehrere Fahrverhalten eines Autofahrers in diesem abstraktem System zu lernen. Schlussendlich wurde erfolgreich eines der gelernten Verhalten in einem dafür passendem Szenario imitiert.

Acknowledgments

Foremost, I would like to express my sincere gratitude to my first supervisor Dr. Heni Ben Amor who supported me passionately and motivated me. I would also like to thank my co-supervisor Dr. Gerhard Neumann for all the time he spent giving me advices and input on my work. Your advices have been priceless.

A special thanks to professor Jan Peters for giving me the opportunity to work on a highly interesting and actual research topic.

At this point, I also want to thank my family who made this study possible to me. Words cannot express my gratefulness. At the end I want to thank all my friends who supported and incited me during the time writing this thesis.

Contents

Figures and Tables

List of Figures

List of Tables

Abbreviations, Symbols and Operators

List of Abbreviations

CCA	Canonical Correlation Analysis
EM	Expectation Maximization
HMM	Hidden Markov Model
i.i.d.	independently and identically distributed
IL	Imitation Learning
KL	Kullback-Leibler divergence
ML	Machine Learning
MP	Markov Process
NLP	natural language processing
OOM	Observable Operator Model
PSR	Predictive State Representation
Spectral HMM	Spectral Hidden Markov Model
SVD	Singular Value Decomposition

List of Symbols

Notation	Description
$\mathbf{1}$	the all ones vector
V	set of emissions of a HMM
π	the initial state distribution vector of a HMM
$\mathbf{O}$	the observation matrix of a HMM
$\overline{\pi}$	the stationary state distribution vector of a HMM
λ	describes a HMM as a system λ assembled by $\mathbf{T}$, $\mathbf{O}$ and π
$\mathbf{T}$	the transition matrix of a HMM
H	set of hidden states of a HMM
$\mathbf{X}$	sequence of observations which occurs when executing a HMM λ
$\boldsymbol{\theta}$	parameters of a probability distribution
s_2	the size of the emission set V
s_1	the size of the hidden state set H
$\mathbf{b}_t$	belief state to time t of the Spectral Hidden Markov Model (Spectral HMM)
$\mathbf{B}_x$	spectral canonical basis for each observation symbol in the subspace of the Spectral HMM
$\mathbf{b}_1$	initial belief state of the Spectral HMM

$\mathbf{b}_\infty$	multiplcative neutral element for the belief states of the Spectral HMM
$\mathbf{p}_1$	probalitiy vector of singletons formed by the given observation sequence
$\hat{\mathbf{p}}_1$	empirical probalitiy vector of singletons formed by the given observation sequence
$\mathbf{P}_{2,1}$	probalitiy matrix of pairs formed by the given observation sequence
$\hat{\mathbf{P}}_{2,1}$	empirical probalitiy matrix of pairs formed by the given observation sequence
$\mathbf{P}_{3,x,1}$	probalitiy tensor of triples formed by the given observation sequence
$\hat{\mathbf{P}}_{3,x,1}$	empirical probalitiy tensor of triples formed by the given observation sequence
$\mathbf{U}$	orthonormal basis won by $\text{SVD}(\mathbf{P}_{2,1})$ that preserves state dynamics
$\hat{\mathbf{U}}$	empirical orthonormal basis won by $\text{SVD}(\hat{\mathbf{P}}_{2,1})$ that preserves state dynamics
$\forall$	the universal quantifier

List of Operators

Notation	Description	Operator
$*$	the conjugate transpose of a complex matrix, behaves like the normal transpose for a real matrix	$(\bullet)^*$
diag	the diagonal function that creates a diagonal matrix from a vector or a diagonal matrix from a vector	$\text{diag}(\bullet)$
KL	the Kullback-Leibler divergence (2.4) for measuring the distance between probability distributions	$\text{KL}(\bullet \| \bullet)$
range	the matrix range operator	$\text{range}(\bullet)$
$\top$	the matrix transpose	$(\bullet)^\top$
$+$	the Moore–Penrose pseudoinverse	$(\bullet)^+$
ln	the natural logarithm	$\ln(\bullet)$
$A(x)$	the observable operator model according to [1]	$A(\bullet)$
p	the probability function	$p(\bullet)$
$\otimes$	the tensor product	$(\bullet \otimes \bullet)$

1 Introduction

Our world is determined by discrete and continuous signals of different nature. These signals of real world processes produce an observable output which can be measured and used to generate a signal model. Once such a model is extracted, we can change its parameters, enhance the observed signal values, or perform other kinds of signal modifications. One broad class of signal models are the statistical signal models where the signal is assumed to be a stochastic process [2]. An example for a statistical signal model is the Hidden Markov Model. Hidden Markov Models have been applied to a wide range of applications, e.g. in speech recognition, natural language processing and genomic sequence modeling [3]. They are a fundamental tool for modeling stochastic systems that obey the Markov property. As most other signal models, HMMs depend on a set of parameters that need to be determined based on the observed data [2, 4]. Typically, this is realized using supervised learning. First, training data is recorded from the system at hand. Given this training data, a machine learning method is used to automatically determine the HMM parameters that produce a similar behavior as observed in the training data. A good parameterization of the HMM leads to an accurate signal model that can be used to analyse and emulate the system at hand, while a bad parameterization leads to a poor model that does not properly reflect the behavior of the system.

Training a HMM is generally done using the Expectation Maximization (EM) algorithm, which maximizes the lower bound of the log likelihood of the training data [4, 5]. However, the EM algorithm for HMMs suffers from two major drawbacks [5]. EM-based training is an iterative and computationally expensive method that requires a substantial number of iterations to converge. In addition, since it is only a local method, it is prone to getting stuck in local optima. Proper initialization before training or repeated execution of training is therefore in practice needed when using HMMs. A solution to these problems has recently been proposed through the introduction of Spectral Learning methods for HMMs [3]. Spectral Learning uses methods from linear algebra to extract the parameters of an HMM from training data. Since many of the used methods have global convergence guarantees, the learned HMMs will most likely have a better performance than those learned using EM. Addtionally, it was shown that spectral approaches are particularly well suited for problems with many observations [3, 6]. These results have made Spectral Learning a viable and promising alternative to traditional EM-based training.

1.1 Related Work

The Spectral Learning algorithm composes from two different ideas. The first idea is called *subspace identification* and is located in the field of system theory [7]. The subspace identification was used in the seventies to identify a state space model from impulse responses, i.e. markov parameters. In general, subspace identification algorithms are used to identify *input-state-output* models. To do so, they learn the linear relation between hidden states and observable output. Spectral Learning exploits this idea of finding the linear relationship between hidden states and observable output data to learn an alternative representation for HMMs. This step of Spectral Learning is based on the tensor decomposition of the defined tensor moments for the HMM [8]. To do so, spectral methods are used on a correlation matrix. This matrix contains correlation information about past and future observations. This method is very similiar to the idea of Canonical Correlation Analysis (CCA) proposed by [9]. However, subspace identification algorithms are used to learn linear dynamical systems such as Kalman Filters and assume additive noise therefore.

The second idea that is fundamental to HMMs are the Observable Operator Models (OOMs). As Spectral Learning cannot directly estimate the latent states of HMMs, the OOM is used to compute all necessary probabilities directly in the space of observable symbols. However, Predictive State Representations (PSRs) can model HMMs and OOMs are HMMs. Hence, they are applicable to additive noise models [6, 1, 3]. There are also latent tree models that can be used in conjunction with the Spectral Learning algorithm [10]. The latent tree models behave very similiar to the HMMs in the context of Spectral Learning. Spectral HMMs have also been extended to a continuous setting. This idea is called Hilbert Space Embeddings of HMMs [11] and allows HMMs to infer in continuous space with the Spectral Learning algorithm proposed by [3]. Likewise the discrete case, the original HMM parameters do not need to be restored for the continuous case [11].

1.2 Outline

Chapter 2 introduces the Expectation Maximization (EM) algorithm, the Hidden Markov Model (HMM) and conluding from these definitions the Baum-Welch algorithm. The EM algorithm is introduced because it is the *state of the art* of Imitation Learning (IL) in Machine Learning (ML) for learning the parameters of a model. The chosen model in this work is the HMM and hence it is introduced together with the EM algorithm for HMMs, called the Baum-Welch algorithm.

Chapter 3 introduces the Spectral Learning algorithm. The Spectral Learning algorithm is introduced for HMMs in particular. Hence, this chapter is based on the definition of the HMM.

Chapter 4 extends Chapter 3 by using the presented spectral algorithm and evaluating results in the context of an experiment.

Chapter 5 summarizes the results of this work and provides an outlook for future work on the topic on Spectral Learning.

2 Hidden Markov Model

The Hidden Markov Model (HMM) is a widely used statistical tool for modeling discrete time series. The theory behind HMMs are that they consist out of hidden states internally. Hence, the hidden states cannot be seen from outside but we can see the observations generated from the hidden states. In the following, an insight at how to use HMMs according to [12] is given. There are three main goals in [12] to acheive with Hidden Markov Models:

1. Given an observation sequence, what is the probability of an optimal observation sequence?
2. Given an observation sequence, what are the hidden states that produced that observable sequence?
3. Given an observation sequence, what HMM does fit the given observation sequence best?

We will have a closer look at the goals of the three issues for which a solution with Hidden Markov Models is needed. In the oncoming chapters, an introduction to the Hidden Markov Model, the Expectation Maximization (EM) algorithm and the Baum-Welch algorithm is given as well as there are examples for basic understanding of these mechanisms.

2.1 The Expectation Maximization Algorithm

Since HMMs are the most fundamental models used for learning in partially observable systems [12], we use them here for the explanation of algorithms that were used prior to Spectral Learning. Basically this chapter focuses on giving an insight on how learning algorithms prior to Spectral Learning were used to learn the parameterization of a model. To start with, we will have a look at the general form of the EM algorithm according to [4]. In our case the EM algorithm in conjunction with the HMM will be shown later after the definition of the HMM.

Assume a data set X where values Z are missing. The data set X is then assumed to be the observable sequence and the data set Z is assumed to be the hidden sequence. Furthermore, let the set of parameters of a model be defined by $\boldsymbol{\theta}$. Our goal now is to compute the new parameter set of the model. The EM algorithm [4] starts with an initial guess for the parameters denoted by $\boldsymbol{\theta}^{\text{old}}$. From the definitions above follow the log likelihood function

$$\ln\{p(X|\boldsymbol{\theta})\} = \ln\left\{\int_Z p(X,Z|\boldsymbol{\theta})\,dZ\right\}. \tag{2.1}$$

The goal of the EM algorithm is to maximize the log likelihood, more precisely the marginal log likelihood, given by the Equation 2.1 with respect to the parameters $\boldsymbol{\theta}$. This problem is hard due to non-convex optimization on the lower bound of the log likelihood. The EM algorithm is split into two steps. The E-step is used to estimate the distribution of the hidden state and the M-step is used to estimate the new parameters of the model. The first step of the EM algorithm is the expectation step or E-step. We estimate the posterior distribution for the hidden data set Z defined as

$$p\left(Z\middle|X,\boldsymbol{\theta}^{\text{old}}\right). \tag{2.2}$$

The expectation of the marginal log likelihood in Equation 2.1 can be found with the posterior distribution in Equation 2.2 and is denoted as $\mathcal{Q}\left(\boldsymbol{\theta},\boldsymbol{\theta}^{\text{old}}\right)$ given by

$$\mathcal{Q}\left(\boldsymbol{\theta},\boldsymbol{\theta}^{\text{old}}\right) = \int_Z p\left(Z\middle|X,\boldsymbol{\theta}^{\text{old}}\right)\cdot\ln\{p(X,Z|\boldsymbol{\theta})\}\,dZ. \tag{2.3}$$

The posterior distribution of the hidden data set was used in Equation 2.3 because the marginal distribution of the hidden set $p(Z)$ can be chosen freely. To explain why the marginal distribution of the hidden data set is replaced by the posterior distribution of the hidden data set, let us further introduce the Kullback-Leibler divergence for this purpose. The Kullback-Leibler divergence is used to measure the distance between two probability distributions. For two probability distributions p and q, the Kullback-Leibler divergence is defined as

$$\mathrm{KL}(p\|q) = -\int_x p(x)\cdot\ln\left\{\frac{q(x)}{p(x)}\right\}dx \tag{2.4}$$

where p is the true distribution and q is the estimated distribution. Furthermore it holds that $\mathrm{KL}\left(p\|q\right) \neq \mathrm{KL}\left(q\|p\right)$ and $\mathrm{KL}\left(p\|q\right) \geq 0$ with the equality if $p(x) = q(x)$. Assuming that $\mathrm{KL}\left(p\left(Z\right)\|\, p\left(Z\middle|X,\boldsymbol{\theta}^{\mathrm{old}}\right)\right) = 0$, the marginal probability of the hidden data set can be replaced by the posterior distribution of the hidden data set. By doing so, we just shift the bound as can be verified by [4]. The decomposition is explained detailed in [4]. Afterwards, the results of the E-step are used for the maximization step or M-step of the EM algorithm. The M-step is the second step of the EM algorithm that follows after the E-step. The new parameters $\boldsymbol{\theta}^{new}$ are estimated in the M-step of the EM algorithm.

$$\begin{aligned} \boldsymbol{\theta}^{new} &= \underset{\boldsymbol{\theta}}{\mathrm{arg}max}\left\{\mathcal{Q}\left(\boldsymbol{\theta},\boldsymbol{\theta}^{\mathrm{old}}\right)\right\} \\ &= \underset{\boldsymbol{\theta}}{\mathrm{arg}max}\left\{\int_Z p\left(Z\middle|X,\boldsymbol{\theta}^{\mathrm{old}}\right)\cdot \ln\left\{p\left(X,Z|\boldsymbol{\theta}\right)\right\}dZ\right\} \qquad \text{with Equation 2.3} \end{aligned} \tag{2.5}$$

The *argmax* operator in the M-step is evaluated using the derivative with respect to the parameters $\boldsymbol{\theta}$ and setting the derivative to zero. The obtained new parameters $\boldsymbol{\theta}^{new}$ are then set to $\boldsymbol{\theta}^{\mathrm{old}}$ at the end of the current iteration for the next iteration, thus $\boldsymbol{\theta}^{\mathrm{old}} = \boldsymbol{\theta}^{new}$ at the end of the current iteration. The E-step, M-step and update of the new parameters define one recursion for the EM algorithm. The EM algorithm iterates until convergence, i.e. until the difference of the log likelihood falls below a lower bound ε. Important attributes of the EM algorithm are that the algorithm does not have any guarantees on finding a good solution and only converges locally. Hence the EM algorithm is a non convex optimization problem.

2.2 Formal Defintion

To introduced the Hidden Markov Model (HMM), important variables for the definition of a HMM are explained similiar to [12] with a slightly different notation in this work. Primarily the Hidden Markov Model is introduced as a state based graph and after that, time series will be added to the process so that the HMM unfolds over time. This explanation should give a good overview of what a HMM is and how it works.

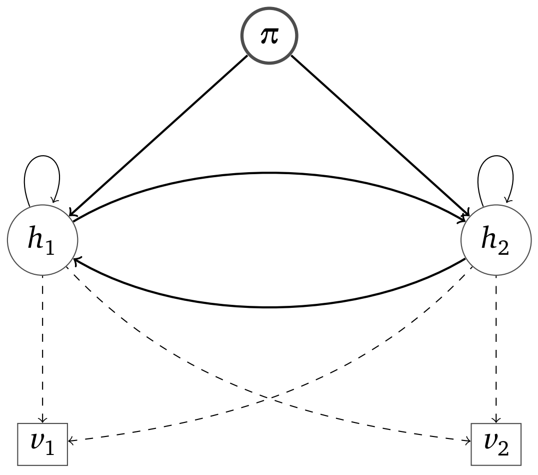


Figure 2.1: A basic Hidden Markov Model with the initial state π and two hidden states h_1 and h_2 which can emit two discrete observation symbols v_1 or v_2 each. The transitions between the hidden states are visualized by the solid line and the emission of observable symbols are visualized by the dashed lines.

A particular visual representations of a HMM is depicted in Figure 2.1. There are hidden states denoted as $h_i \in H$ for $i = 1, 2, \ldots, s_1$ where s_1 describes the length of the hidden state set H. The Figure 2.1 shows two hidden states transitions to themselves and Figure 2.1 also shows the observable symbols, the so called emissions that can be observed over time. These observable symbols are denoted with $v_j \in V$ for $j = 1, 2, \ldots, s_2$ where s_2 describes the length of the observable symbol set V. At last π describes the initial state distribution from which the *first* initial state h_0 is independently and identically distributed (i.i.d.) sampled defined as the marginal probability

$$[\pi]_i = p\left(h_i\right). \tag{2.6}$$

Now, let us define the matrix quantities $\mathbf{T}$ and $\mathbf{O}$ which derive from the hidden state set H and the observable symbol set V. The $\mathbf{T}$ and $\mathbf{O}$ matrices can be understood as a linear operator for the transition between hidden states h_t and observable symbols v_t over time. The $\mathbf{T}$ matrix is defined as

$$[\mathbf{T}]_{j,i} = p\left(h_j\middle|h_i\right) \tag{2.7}$$

and describes the probability distribution of the transitions between each hidden states. Similiar, the **O** matrix is defined as

$$[\mathbf{O}]_{k,j} = p\left(v_k \middle| h_j\right) \tag{2.8}$$

and represents the probability distribution to sample an observable symbol from a hidden state. The columns of the transition matrix **T** and the observation matrix **O** should **always** sum up to 1 because these matrices and vector are probability distributions. Now, a HMM can be defined as a three tupel λ where

$$\lambda = (\mathbf{T},\ \mathbf{O},\ \pi)$$

given Equations 2.6, 2.7 and 2.8. This Hidden Markov Model λ obeys the following *Markov Condition*.

Markov condition.
The hidden state of time step t is only dependent on the last hidden state and not on the whole history, thus

$$\forall t\ :\ p\left(h_t \middle| h_{1:t-1}\right) = p\left(h_t \middle| h_{t-1}\right). \tag{2.9}$$

Likewise it must hold for the observable symbols that the observation probability of time step t only depends on the hidden state of time step t. This behavior means that

$$\forall t\ :\ p\left(v_t \middle| h_t\right).$$

where $b_j(v_k)$ is the probability to emit an observable symbol v_k in state h_j, both to time t where $k = 1,\ 2,\ \ldots,\ s_2$. The variables which describe the Hidden Markov Model are listed below as a short overview.

- $\mathbf{T} \in \mathbb{R}^{s_1 \times s_1}$ is the matrix of state transition probabilities and is used to make transitions between hidden states of a HMM
- $\mathbf{O} \in \mathbb{R}^{s_2 \times s_1}$ is the observation probability matrix and produces observations from a given hidden state
- $\pi = (\pi_1\ \pi_2 \ldots \pi_{s_1})^\top$ is the initial state distribution
- $H = \{h_1, h_2, \ldots, h_{s_1}\}$ is a distinct set of hidden states of the Markov Process (MP)
- $V = \{1, 2, \ldots, s_2\}$ is the set of possible observations (these observation symbols are often also called emissions)
- s_1 is the size of the set H (i.e. the amount of hidden states)
- s_2 is the size of the set V

The simplified notation $\{1,\ 2,\ \ldots,\ s_2\}$ of the observation symbols comes without loss of generality as suggested in [12] and will therefore be used throughout. The number could easily map to an actual observable symbol. This mapping of observations to numbers should be kept in mind for the various following examples on Hidden Markov Models.

Another intuitive way of understanding a HMM is to look at the HMM over time. To do so, the HMM is *unfolded* into a Trellis diagram.

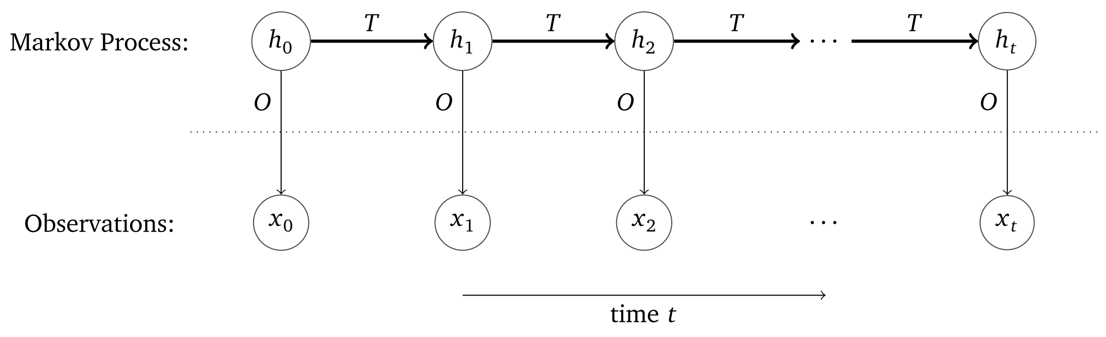


Figure 2.2: A Trellis diagram of a generic Hidden Markov Model (HMM) representation. This diagram gives an intuition of the mechanics of HMMs over time series. Observable symbols v_t are generated by the O matrices after each transition of hidden states h_t done with the **T** matrix. The dotted line in the Trellis diagram visually seperates the hidden states h_t from the observations x_t.

2.3 Example on Hidden Markov Model

In this section, we want to give an example for a more fundamental understanding of HMMs for further proceeding in this work. The particular example is visualized in Figure 2.3.

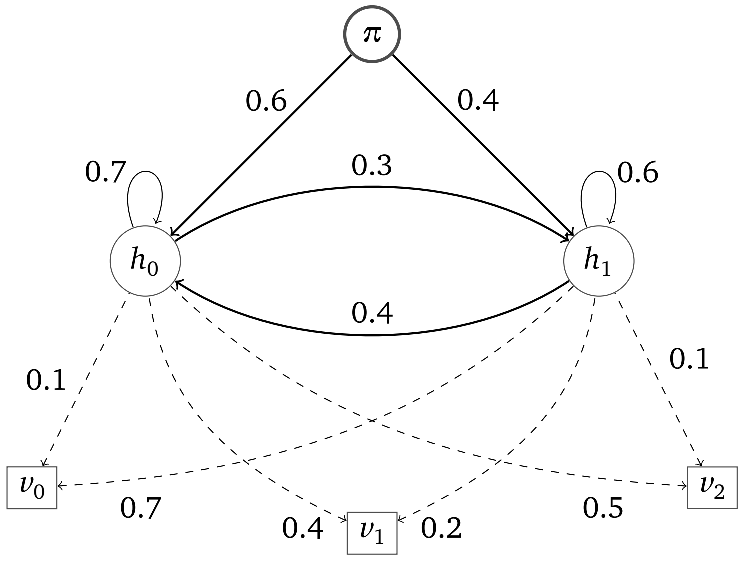


Figure 2.3: A concrete HMM with two states h_0 and h_1 which can emit three discrete observation symbols v_0, v_1 or v_2.

Assuming a HMM similiar to the one in Figure 2.1 with the following transition, observation and initial state probability distributions

$$\mathbf{T} = \begin{pmatrix} 0.7 & 0.4 \\ 0.3 & 0.6 \end{pmatrix} \qquad \mathbf{O} = \begin{pmatrix} 0.1 & 0.7 \\ 0.4 & 0.2 \\ 0.5 & 0.1 \end{pmatrix} \qquad \pi = \begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} \tag{2.10}$$

where the columns should always sum up to 1 since this is a probability distribution.
For the purpose of this example we consider the following model. The given data for learning is a sequence containing weather information at different days. The days in this model represent the different time steps t of the HMM. We also define a set of actions $\{walk, shop, clean\}$ and recall the simplified notation from Section 2.2 for observable symbols. Therefore we map the observable symbols to $walk = 1$, $shop = 2$ and $clean = 3$. The problem state of this task is now to guess the most likeli sequence of weather based on an input sequence of actions. Thus to compute the probability whether it was a sunny day or a rainy day for a given action or a sequence of actions. The probability of observing an action in a specific weather situation are summarized in Table 2.1. The hidden states represent the weather here.

	rainy	sunny
rainy	0.7	0.4
sunny	0.3	0.6

	rainy	sunny
walk	0.1	0.7
shop	0.4	0.2
clean	0.5	0.1

Table 2.1: Transition and observation probabilities for a simple example of a Hidden Markov Model

Lets assume the observation sequence $\mathbf{X} = (3, 1, 2)$ and let S and R denote the hidden states for *sunny* and *rainy*. The joint probability distribution for a complete data sequence given the parameters $\boldsymbol{\theta} = (\pi, \mathbf{T}, \mathbf{O})$ of the HMM can generally be computed as

$$p\left(x_{1:t_f}, h_{1:t_f} \middle| \boldsymbol{\theta}\right) = \pi\left(h_0 \middle| \pi\right) \cdot \left(\prod_{t=1}^{t_f-1} p\left(h_{t+1} \middle| h_t, \mathbf{T}\right)\right) \cdot \left(\prod_{t=0}^{t_f-1} p\left(x_t \middle| h_t, \mathbf{O}\right)\right) \tag{2.11}$$

where t_f is the final time and describes the length of the sequence. The probability for a permutated sequence, e.g. *RSR*, will be computed detailed for better understanding. Using Equation 2.11, we get

$$\begin{aligned} p(3,1,2\,,\,R,S,R) &= \pi(R)\cdot p(3,R)\cdot p(S,R)\cdot p(1,S)\cdot p(R,S)\cdot p(2,R) \\ &= 0.6\cdot 0.5\cdot 0.3\cdot 0.7\cdot 0.4\cdot 0.4 \\ &= 0.01008. \end{aligned} \tag{2.12}$$

The probabilities of the other permutations of sequences are computed similiarly. For calculation of those, the matrices in Equation 2.10 were used which are summarized in Table 2.1. And hence this will not be computed detailed anymore. These probabilities of the different hidden state sequences H are listed in Table 2.2. The normalized values are the conditional probabilities that sum up to one in contrast to the computed joint probabilities that lack the normalization. The values were simply normalized by divison with the sum of the joint probabilities.

H	RRR	RRS	RSS	RSR	SRS	SRR	SSR	SSS
$p(H,X)$	0.00588	0.00126	0.00756	0.01008	0.000168	0.000784	0.001344	0.001008
$p(H\|X)$	0.209372	0.044865	0.269192	0.358923	0.005982	0.027916	0.047856	0.035892

Table 2.2: Resulting permutated hidden state sequence H and its joint probabilities $p(H,X)$ and normalized probabilities $p(H|X)$ of the particular Hidden Markov Model for the given action sequence $\mathbf{X}=(3,1,2)$.

Now, we are able to compute the conditional probability of the hidden state sequence of each time step $t = 0,1,2$. The marginal probability of the hidden states R and S of each time step t is computed by the sum of the conditional probabilities as shown in Table 2.2 starting with the corresponding hidden state sequence of each time step t. As a result, the marginal probabilities for the hidden states R and S in each time step t of the sequence are obtained. The results of this procedure are listed in the Table 2.3. With the given results from Table 2.3, we can see the most likeli choice for that sequence in this example. The most likeli hidden state sequence is *RSR* given Table 2.2 and Table 2.3.

time step	t_0	t_1	t_2
$p(\mathrm{R})$	0.882352	0.288135	0.644067
$p(\mathrm{S})$	0.117648	0.711865	0.355933

Table 2.3: Resulting marginal probabilities of each hidden state in time step $t = 0,1,2$ in the particular Hidden Markov Model.

2.4 Baum-Welch Algorithm

The Baum-Welch algorithm is the EM algorithm variant for HMMs with a modified E-step of the original EM algorithm [4]. Recall the definition of the EM algorithm from Section 2.1 and the definition of the HMM from Section 2.2. Likewise the EM algorithm, the Baum-Welch algorithm tries to maximize the log likelihood function in HMMs [4]. Let us start with the definition of an initial parameter set of the HMM defined by $\boldsymbol{\theta}^{old}$ and assume an observable data set X with some missing values which we assume to be the hidden data set Z. Now, we can find the posterior distribution of the hidden data set defined by

$$p\left(Z\middle|X,\boldsymbol{\theta}^{old}\right), \tag{2.13}$$

instead of using the marginal probability distribution by minimizing the KL. We already applied this trick in Section 2.1. Prior to Equation 2.3 of Section 2.1, we reformulate the expected complete-data log likelihood as

$$\mathcal{Q}\left(\boldsymbol{\theta},\boldsymbol{\theta}^{old}\right) = \sum_Z p\left(Z\middle|X,\boldsymbol{\theta}^{old}\right)\cdot \ln\left\{p\left(X,Z|\boldsymbol{\theta}\right)\right\}. \tag{2.14}$$

For further proceeding, we introduce the marginal posterior distribution of a hidden variable and joint posterior distribution of two successive hidden variables denoted as $\gamma(z_n)$ given by Equation 2.15 and $\xi(z_{n-1},z_n)$ given by Equation 2.16 respectively [4]. The marginal posterior distribution of a hidden variable is defined as

$$\gamma(z_n) = p\left(z_n\middle|X,\boldsymbol{\theta}^{old}\right) \tag{2.15}$$

and the joint posterior distribution of two succesive hidden variables is defined as

$$\xi\left(z_{n-1}, z_n\right) = p\left(z_{n-1}, z_n \middle| X, \boldsymbol{\theta}^{old}\right). \tag{2.16}$$

The marginal posterior distribution of a hidden variable and the joint posterior distribution of two successive hidden variables are estimated in the E-step of the Baum-Welch algorithm and are estimated very efficiently using the forward-backward algorithm [4]. It holds that the expectation for Equation 2.15 and Equation 2.16 sums up to 1 for all hidden variables and successive hidden variables. Now we want to use the joint probability distribution of the complete data given by Equation 2.11 in Section 2.3 in conjunction with the marginal posterior distribution of a hidden variable in Equation 2.15 and the joint posterior distribution of two successive hidden variables in Equation 2.16. For this purpose, we extend the marginal probability z_n and joint probabilities z_{n-1}, z_n to a conditional probability form z_{nk} and $z_{n-1,j}, z_{nk}$. The sum of the conditional probability always sums up to 1. We plug the Equations 2.15, 2.16 and 2.11 into Equation 2.14 and receive

$$\begin{aligned}\mathcal{Q}\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}\right) = &\sum_k \gamma\left(z_{1k}\right) \cdot \ln\left(p\left(z_{1k}\right)\right) + \sum_n \sum_j \sum_k \xi\left(z_{n-1,j}, z_{nk}\right) \cdot \ln\left(p\left(z_{nj} \middle| z_{n-1,k}\right)\right) \\ &+ \sum_n \sum_k \gamma\left(z_{nk}\right) \cdot \ln\left(p\left(x_n \middle| z_k\right)\right).\end{aligned} \tag{2.17}$$

The Equation 2.17 is then maximizied in the M-step of the Baum-Welch algorithm [4]. From this point, we can proceed with the Baum-Welch algorithm starting as in Equation 2.5. To do so, we use the argmax operator again to maximize the log likelihood function for the HMM. We do this by deriving Equation 2.17 with respect to the parameters $\boldsymbol{\theta}$ and setting the derivative to zero. We then solve this equation for $\boldsymbol{\theta}$ and set them, i.e. $\boldsymbol{\theta}^{new} = \boldsymbol{\theta}^{old}$.

2.4.1 The Forward-Backward Algorithm

The forward-backward algorithm is used to obtain the marginal posterior of the hidden variables for conditional probability in Equation 2.15 and the joint posterior of two successive hidden variables for conditional probability in Equation 2.16. We will derive the formulars for the probabilities which can also be written as the sum of the conditional probabilities [4]. The graph for the HMM can be represented as a tree [4] and therefore a two state message passing algorithm can be used. Let $\mathbf{X} = \{x_1, \ldots, x_N\}$ define our observable data set. It holds that

$$\gamma\left(z_n\right) = p\left(z_n \middle| \mathbf{X}, \boldsymbol{\theta}^{old}\right) = \frac{p\left(\mathbf{X}, \boldsymbol{\theta}^{old} \middle| z_n\right) \cdot p\left(z_n, \boldsymbol{\theta}^{old}\right)}{p\left(\mathbf{X}, \boldsymbol{\theta}^{old}\right)} \tag{2.18}$$

using Bayes' theorem. Because $\boldsymbol{\theta}^{old}$ is fixed, we leave it out for simplicity of the equations. With the conditional independence property and the product rule of probability, the Equation 2.18 decomposes into

$$\gamma\left(z_n\right) = \frac{p\left(x_1, \ldots, x_n, z_n\right) \cdot p\left(x_{n+1}, \ldots, x_N \middle| z_n\right)}{p\left(\mathbf{X}\right)}. \tag{2.19}$$

Now let us define the forward propagation

$$\begin{aligned}\alpha\left(z_n\right) &= p\left(x_1, \ldots, x_n, z_n\right) \\ &= p\left(x_n \middle| z_n\right) \cdot \sum_{z_{n-1}} \alpha\left(z_{n-1}\right) \cdot p\left(z_n \middle| z_{n-1}\right)\end{aligned} \tag{2.20}$$

and the backward propagation

$$\begin{aligned}\beta\left(z_n\right) &= p\left(x_{n+1}, \ldots, x_N\right) \\ &= \sum_{z_{n+1}} \beta\left(z_{n+1}\right) \cdot p\left(x_{n+1} \middle| z_{n+1}\right) \cdot p\left(z_{n+1} \middle| z_n\right)\end{aligned} \tag{2.21}$$

according to [4]. The forward propagation 2.20 and the backward propagation 2.21 are plugged into the Equation 2.19 which yields to

$$\begin{aligned}\gamma\left(z_n\right) &= \frac{p\left(x_1, \ldots, x_n, z_n\right) \cdot p\left(x_{n+1}, \ldots, x_N \middle| z_n\right)}{p\left(\mathbf{X}\right)} \\ &= \frac{\alpha\left(z_n\right) \cdot \beta\left(z_n\right)}{p\left(\mathbf{X}\right)} = \frac{\alpha\left(z_n\right) \cdot \beta\left(z_n\right)}{\sum_{z_n} \alpha\left(z_n\right) \cdot \beta\left(z_n\right)}.\end{aligned}$$

For the joint posterior distribution of two successive hidden variables follows similiarly [4]

$$\begin{aligned}\xi\left(z_{n-1}, z_n\right) &= p\left(z_{n-1}, z_n \middle| \mathbf{X}\right) \\ &= \frac{\alpha\left(z_{n-1}\right) \cdot p\left(x_n \middle| z_n\right) \cdot p\left(z_n \middle| z_{n-1}\right) \cdot \beta\left(z_n\right)}{p\left(\mathbf{X}\right)} = \frac{\alpha\left(z_{n-1}\right) \cdot p\left(x_n \middle| z_n\right) \cdot p\left(z_n \middle| z_{n-1}\right) \cdot \beta\left(z_n\right)}{\sum_{z_n} \alpha\left(z_n\right) \cdot \beta\left(z_n\right)}\end{aligned}$$

2.5 Example on Baum-Welch Algorithm

The Baum-Welch algorithm is the EM variant for learning a HMM as we learned in the former section. Hence, we want to have a look how the Baum-Welch algorithm works in practice. The Section 2.3 showed how to compute the sequence probability for a HMM analytically given the parameters of the HMM. However, typically we do not know the parameters of a HMM, and hence, these parameters need to be learned. We use the same HMM from Section 2.3 to sample three different observation sequences. The samples are drawn i.i.d. from the HMM that is found in Equation 2.10. Afterwards, the Baum-Welch algorithm is used to learn the parameters of the HMM. Three test runs were made where the Baum-Welch algorithm tried to learned the parameters of the HMM. The results of the three runs are depicted in Figure 2.4.

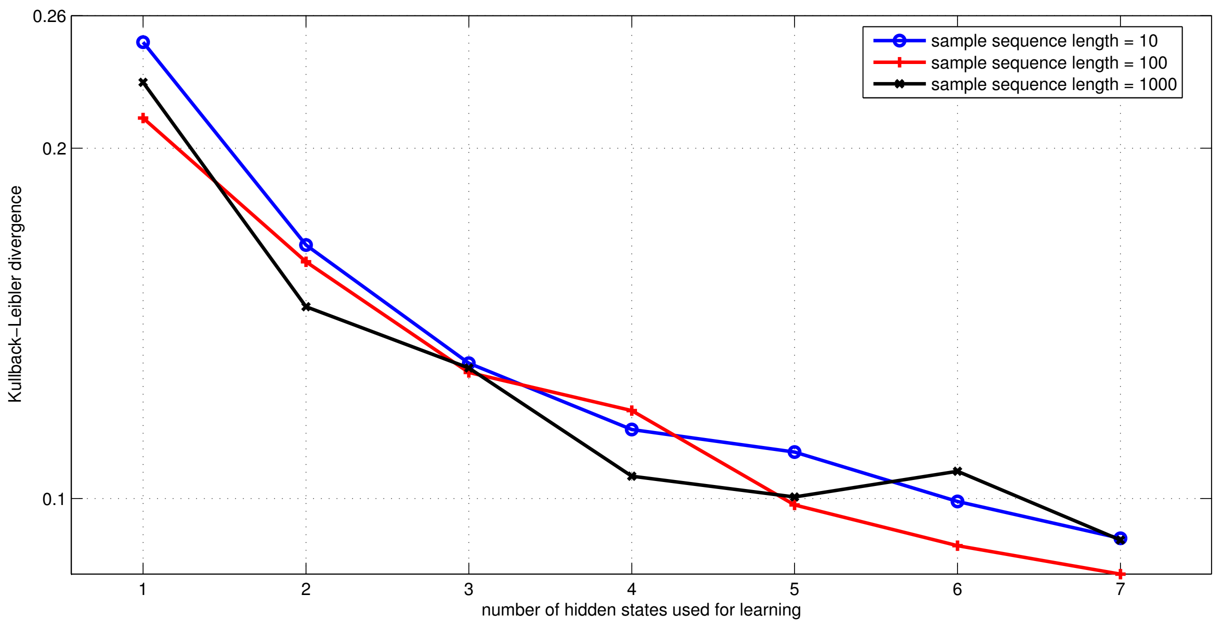


Figure 2.4: Evaluation of the Baum-Welch algorithm for three different sample sequence lengths. The KL on the y-axis is plotted against the number of hidden states used for learning on the x-axis.

Seven different HMMs were learned for the three different runs. The results of the Kullback-Leibler divergence for the each run and learned HMM are listed in the Table 2.4.

number of hidden states	1	2	3	4	5	6	7
KL for 10 samples	0.2467	0.1651	0.1307	0.1146	0.1096	0.09941	0.09242
KL for 100 samples	0.2123	0.1597	0.1283	0.119	0.09875	0.09108	0.08611
KL for 1000 samples	0.2279	0.1462	0.1295	0.1045	0.1003	0.1055	0.09211

Table 2.4: The Kullback-Leibler divergence for each run and each learned HMM. The HMMs were learned with the Baum-Welch algorithm.

The best HMM is learned for a sample sequence of length 100 and is assumed to have 7 hidden states. Now we want to use the best learned HMM to compute the probability of the observable sequence $(3,1,2)$ from the Section 2.3. The probabilitiy for the observable sequence $(3,1,2)$ of the original HMM is *32.40%* and the probability for the learned HMM is *33.20%*.

3 Spectral Learning of Hidden Markov Model

An online Spectral Learning algorithm is proposed in this chapter which makes use of spectral methods such as the Singular Value Decomposition (SVD). The Spectral Learning algorithm depends exclusively on observable sequences. The algorithm is a so called ***one-shot*** method which uses linear algebra for the learning instead of iteratively approximating the "true" parameters of the original model as shown in the Expectation Maximization (EM) algorithm.

3.1 Observable Operator Models

OOMs were firstly introduced by [1]. The name is influenced by the fact that the OOM transitions between the observable states given only observations. Thus the observable operator is based on observations which also depends on the HMM. It is defined as

$$A(x) = T \cdot \operatorname{diag}\left(\mathbf{O}_{x,1}, \ldots, \mathbf{O}_{x,s_1}\right) \tag{3.1}$$

such that the following holds

$$p\left(x_t, \ldots, x_1\right) = \mathbf{1}^\top \cdot A\left(x_t\right) \cdot \ldots \cdot A\left(x_1\right) \cdot \pi.$$

The OOM is used in this context here so that we do not have to represent the hidden state explicitly anymore. Hence, it is used for the definition of the observable representation in Section 3.2.

3.2 Observable Representation

To start with, we use the just introduced OOMs so that we have a representation of the HMM that is directly learnable from observations, i.e. we do not need to know the transition model. The conditional independence property of the HMM is important because it guarantees that the HMM fully describes the probability distribution of a sampled sequence. The sampled sequences will be used to define an empirical estimate of the observable representation which we will denote with a hat. Let us also define t as the current time step and t_f as the final time step, i.e. the last time step, throughout this section. It is also required that the following condition holds for the HMM.

Non-degeneracy condition.
The $\mathbf{O}$ *and* $\mathbf{T}$ *matrices have full column rank and* $\pi > 0$ *element-wise.*

This condition is needed to let the Spectral Learning algorithm seperate between an output distribution and a mixture of output distributions. The observable representation itself is based on probabilities of singletons, pairs and triples where these are only based on the given observations. These probabilities form the first three moments of the HMM. These form the following vector $\mathbf{p}_1$, matrix $\mathbf{P}_{2,1}$ and matrices $\mathbf{P}_{3,x,1}$ (or set of matrices). The vector of singletons is defined as follows

$$\left[\mathbf{p}_1\right]_i = p\left(x_1 = i\right) = \mathbf{O} \cdot \overline{\pi}. \tag{3.2}$$

The $\mathbf{p}_1$ vector describes the marginal probability of an observation in the observation sequence with $\mathbf{p}_1 \in \mathbb{R}^{s_2}$. In addition the matrix $\mathbf{P}_{2,1} \in \mathbb{R}^{s_2 \times s_2}$ consists out of the pairwise correlation of two successive observations. The matrix $\mathbf{P}_{2,1}$ is formed as shown in Equation 3.3 and is defined as

$$\begin{aligned}\left[\mathbf{P}_{2,1}\right]_{i,j} &= p\left(x_2 = i, x_1 = j\right) \\ &= \mathbf{O} \cdot \mathbf{T} \cdot \operatorname{diag}\left(\overline{\pi}\right) \cdot \mathbf{O}^\top.\end{aligned} \tag{3.3}$$

This matrix is also very important for the Spectral Learning algorithm since it has the meaning of the correlation between future and past observations. This matrix will be of greater use in the Section 3.3 for the core step of the Spectral Learning algorithm because it is used to identify a projection $\mathbf{U}$ between the subspace in which we define the observable representation and our current space. The set of matrices $\mathbf{P}_{3,x,1} \in \mathbb{R}^{s_2 \times s_2}$ is formed in Equation 3.4 and describes the parametrized correlation of the previous observation and the successive observation for some given observation x_t. We may notice that the form of the $\mathbf{P}_{3,x,1}$ set of matrices is very similiar to the form of the $\mathbf{P}_{2,1}$ matrix in Equation 3.3 except for the OOM (right before a new observation is generated). OOMs are used to generate an observable sequence or for prediction. The OOMs used here were presented in Section 2.3.

$$\begin{aligned}\left[\mathbf{P}_{3,x,1}\right]_{i,j} &= p\left(x_3 = i, x_2 = v, x_1 = j\right) && \forall v \in V \\ &= \mathbf{O} \cdot A(x) \cdot \mathbf{T} \cdot \operatorname{diag}\left(\overline{\pi}\right) \cdot \mathbf{O}^\top && \forall x \in \mathbf{X}\end{aligned} \tag{3.4}$$

There are multiple matrices given with $\mathbf{P}_{3,x,1}$ which are put together as a third grade tensor matrix. We provide a short introduction of the tensor product. An exhausting definition of the tensor product can be found in [8] which also contributes to Spectral Learning. A real tensor of order d is denoted as a d-way array $\left[M_{i_1,i_2,\ldots,i_d}\right]$ of real numbers $i_1 = i_2 = \ldots = i_d = n$ for a tensor $M \in \bigotimes^d \mathbb{R}^n$ in this context. The $\mathbf{P}_{3,x,1}$ tensor of third grade is then written as

$$\begin{aligned}\left[\mathbf{P}_{3,x,1}\right]_{i,k,j} &= p\left(x_3 = i, x_2 = k, x_1 = j\right) \\ &= \left(\mathbf{O} \cdot A\left(x_k\right) \cdot \mathbf{T} \cdot \operatorname{diag}\left(\overline{\pi}\right) \cdot \mathbf{O}^\top\right) \otimes e_k && \forall x \in \mathbf{X}\end{aligned} \tag{3.5}$$

where e_k is the k-th column vector of the canonical basis and the observation symbol v_k is proportional to x_2 in Equation 3.5. The next Section 3.3 focuses on how to obtain the unitary matrix $\mathbf{U}$ we use here and how to understand it. The matrix $\mathbf{U}$ is the basis of the subspace obtained from the Singular Value Decomposition (SVD) of the matrix of pairs $\mathbf{P}_{2,1}$. This matrix is needed to create the final observable representation in the form of *belief states*.

These *belief states* are defined in the following. Let us start with the initial belief state defined by the Equation 3.6. The basic of of the formulation of an initial belief state is closely related to the HMM. We start from the initial belief state and transition through the belief states as we would transition in a HMM.

$$\mathbf{b}_1 = \mathbf{U}^\top \cdot \mathbf{p}_1 = \mathbf{U}^\top \cdot \mathbf{O} \cdot \overline{\pi} \tag{3.6}$$

This initial belief state vector behaves similiar to the stationary state distribution from the HMM. In fact, [3] proofed that this is only a linear relation. Furthermore there exists a canonical all ones vector of the observable representation expressed as a belief transition state shown in Equation 3.7. This belief state behaves as the neutral element for multiplication in the subspace and sums up the rows of our current belief state. Note, that the observable representation is defined and operates in the same subspace.

$$\begin{aligned}\mathbf{b}_\infty{}^\top &= \mathbf{p}_1{}^\top \cdot \left(\mathbf{U}^\top \cdot \mathbf{P}_{2,1}\right)^+ \\ &= \pi^\top \cdot \mathbf{O}^\top \cdot \left(\mathbf{U}^\top \cdot \mathbf{P}_{2,1}\right)^+ \\ &= \mathbf{1}^\top \cdot \mathbf{T} \cdot \operatorname{diag}\left(\overline{\pi}\right) \cdot \mathbf{O}^\top \cdot \left(\mathbf{U}^\top \cdot \mathbf{P}_{2,1}\right)^+ \\ &= \mathbf{1}^\top \cdot \left(\mathbf{U}^\top \cdot \mathbf{O}\right)^{-1} \cdot \left(\mathbf{U}^\top \cdot \mathbf{O}\right) \cdot \mathbf{T} \cdot \operatorname{diag}\left(\overline{\pi}\right) \cdot \mathbf{O}^\top \cdot \left(\mathbf{U}^\top \cdot \mathbf{P}_{2,1}\right)^+ \\ &= \mathbf{1}^\top \cdot \left(\mathbf{U}^\top \cdot \mathbf{O}\right)^{-1} \cdot \left(\mathbf{U}^\top \cdot \mathbf{P}_{2,1}\right) \cdot \left(\mathbf{U}^\top \cdot \mathbf{P}_{2,1}\right)^+ \\ &= \mathbf{1}^\top \cdot \left(\mathbf{U}^\top \cdot \mathbf{O}\right)^{-1}\end{aligned} \tag{3.7}$$

Now that we have an initial belief state $\mathbf{b}_1$ and the belief state $\mathbf{b}_\infty$, we want to transition between observations in our belief state. Hence, we define a tensor of canonical bases for the observations in Equation 3.8.

$$\begin{aligned}\mathbf{B}_x &= \left(\mathbf{U}^\top \cdot \mathbf{P}_{3,x,1}\right) \cdot \left(\mathbf{U}^\top \cdot \mathbf{P}_{2,1}\right)^+ \\ &= \left(\mathbf{U}^\top \cdot \mathbf{O} \cdot A(x) \cdot \mathbf{T} \cdot \operatorname{diag}\left(\overline{\pi}\right) \cdot \mathbf{O}^\top\right) \cdot \left(\mathbf{U}^\top \cdot \mathbf{P}_{2,1}\right)^+ \\ &= \left(\mathbf{U}^\top \cdot \mathbf{O}\right) \cdot A(x) \cdot \mathbf{T} \cdot \operatorname{diag}\left(\overline{\pi}\right) \cdot \mathbf{O}^\top \cdot \left(\mathbf{U}^\top \cdot \mathbf{P}_{2,1}\right)^+ \\ &= \left(\mathbf{U}^\top \cdot \mathbf{O}\right) \cdot A(x) \cdot \left(\mathbf{U}^\top \cdot \mathbf{O}\right)^{-1} \cdot \left(\mathbf{U}^\top \cdot \mathbf{O}\right) \cdot \mathbf{T} \cdot \operatorname{diag}\left(\overline{\pi}\right) \cdot \mathbf{O}^\top \cdot \left(\mathbf{U}^\top \cdot \mathbf{P}_{2,1}\right)^+ \\ &= \left(\mathbf{U}^\top \cdot \mathbf{O}\right) \cdot A(x) \cdot \left(\mathbf{U}^\top \cdot \mathbf{O}\right)^{-1} \cdot \left(\mathbf{U}^\top \cdot \mathbf{P}_{2,1}\right) \cdot \left(\mathbf{U}^\top \cdot \mathbf{P}_{2,1}\right)^+ \\ &= \left(\mathbf{U}^\top \cdot \mathbf{O}\right) \cdot A(x) \cdot \left(\mathbf{U}^\top \cdot \mathbf{O}\right)^{-1}\end{aligned} \tag{3.8}$$

The term $\left(\mathbf{U}^\top \cdot \mathbf{O}\right)$, e.g. given in Equation 3.8, cancels out with its inverse when multiplying the $\mathbf{B}_x$ matrices.

Finally, let us have a look how to estimate the moments of the HMM. In practice, we can obtain the empirical vector of singletons $\hat{\mathbf{p}}_1$ from Equation 3.2 as shown in Figure 3.1 where the occurence of each observation symbol $v \in V$ is counted from the observation sequence $\mathbf{X}$ and normalized with the sum of all occurences of the observable symbols afterwards. The beginning and the end of the sequence in Figure 3.1 is left out, just to give the idea on how to proceed on the given sequence for the empirical singleton vector $\hat{\mathbf{p}}_1$.

$$\cdots \quad x_{i-3} \quad x_{i-2} \quad \boxed{x_{i-1}} \quad x_i \quad x_{i+1} \quad x_{i+2} \quad x_{i+3} \quad \cdots$$

$$\cdots \quad x_{i-3} \quad x_{i-2} \quad x_{i-1} \quad \boxed{x_i} \quad x_{i+1} \quad x_{i+2} \quad x_{i+3} \quad \cdots$$

$$\cdots \quad x_{i-3} \quad x_{i-2} \quad x_{i-1} \quad x_i \quad \boxed{x_{i+1}} \quad x_{i+2} \quad x_{i+3} \quad \cdots$$

Figure 3.1: Computation of the first empirical moment, the $\hat{\mathbf{p}}_1$ vector, by counting the observable symbols $v \in V$ from the observation sequence for $t = 1, \ldots, t_f$.

Basically, the observation sequence $\mathbf{X}$ of length $|\mathbf{X}|$ will be splitted in $|\mathbf{X}|$ many observation sequences of length one in Figure 3.1. It is very important to keep in mind that this presupposes that the observations were all drawn i.i.d. from the same stationary state distribution $\overline{\pi}$. In practive we remove the start of a sequence and assume that we start from the stationary state distribution $\overline{\pi}$. We could get poor results due to learning a mixture of distributions if we would not delete the begin of a sequence. For completeness, we also show how to estimate the pairs and triples in Figure 3.2 and Figure 3.3 respectively.

$$\cdots \quad x_{i-3} \quad x_{i-2} \quad \boxed{x_{i-1} \quad x_i} \quad x_{i+1} \quad x_{i+2} \quad x_{i+3} \quad \cdots$$

$$\cdots \quad x_{i-3} \quad x_{i-2} \quad x_{i-1} \quad \boxed{x_i \quad x_{i+1}} \quad x_{i+2} \quad x_{i+3} \quad \cdots$$

$$\cdots \quad x_{i-3} \quad x_{i-2} \quad x_{i-1} \quad x_i \quad \boxed{x_{i+1} \quad x_{i+2}} \quad x_{i+3} \quad \cdots$$

Figure 3.2: Computation of the second empirical moment, the $\hat{\mathbf{P}}_{2,1}$ matrix, by counting the observable symbols $v \in V$ from the observation sequence for $t = 1, \ldots, t_f$.

$$\cdots \quad x_{i-3} \quad \boxed{x_{i-2} \quad \boxed{x_{i-1}} \quad x_i} \quad x_{i+1} \quad x_{i+2} \quad x_{i+3} \quad \cdots$$

$$\cdots \quad x_{i-3} \quad x_{i-2} \quad \boxed{x_{i-1} \quad \boxed{x_i} \quad x_{i+1}} \quad x_{i+2} \quad x_{i+3} \quad \cdots$$

$$\cdots \quad x_{i-3} \quad x_{i-2} \quad x_{i-1} \quad \boxed{x_i \quad \boxed{x_{i+1}} \quad x_{i+2}} \quad x_{i+3} \quad \cdots$$

Figure 3.3: Computation of the third empirical moment, the $\hat{\mathbf{P}}_{3,x,1}$ tensor, by counting the observable symbols $v \in V$ from the observation sequence for $t = 1, \ldots, t_f$.

With these empircal estimates, we are able to reformulate the belief states in an empirical context given the linear projection $\hat{\mathbf{U}}$.

3.3 The Core of Spectral Learning

The core of the Spectral Learning algorithm is the subspace identification [7] given by the Singular Value Decomposition of the $\mathbf{P}_{2,1}$ matrix [3]. The idea of this is closely related to the Canonical Correlation Analysis (CCA) from [9] which exploits the linear correlating relationship between multidimensional variables. The $\mathbf{P}_{2,1}$ matrix in Equation 3.3 has the following structure

$$\mathbf{O} \cdot \mathbf{T} \cdot \operatorname{diag}\left(\overline{\pi}\right) \cdot \mathbf{O}^\top$$

which dissects into the parameters of the HMM, the observation matrix $\mathbf{O}$ at the left and its transpose at the right. The foundation of Spectral Learning seeks a factorization that cancels out the hidden state in the middle. The basic idea is now to obtain an unitary correlation matrix of the observation that may be of lower dimension and preserves the state dynamics of the original system. This unitary basis can be found via the *thin* SVD of the $\mathbf{P}_{2,1}$ matrix from Equation 3.3 such that the following *invertibility condition* holds. The SVD is given by the following equation

$$SVD\left(\mathbf{P}_{2,1}\right) = \mathbf{U} \cdot \mathbf{D} \cdot \mathbf{W}^{*} \tag{3.9}$$

The unitary matrix $\mathbf{U}$ consists out of the non-zero left-singular vectors because then $\text{range}(\mathbf{U}) = \text{range}(\mathbf{O})$ such that the *invertibility condition* holds as in [3].

Invertibility condition.
The matrix $\left(\mathbf{U}^{\top} \cdot \mathbf{O}\right)$ *is invertible where* $\mathbf{U}$ *is found by the thin SVD of* $\mathbf{P}_{2,1}$.

The correlation matrix $\mathbf{P}_{2,1}$ holds correlation information of two views in each matrix component. The *thin* SVD basically tries to find a linear projection that mutually maximizes the correlation between two views. For this purpose, the OOMs presented in Section 2.3 are used which are natural in the context of subspace identification [3].

Then, after definition of the observable representation from Equations 3.6, 3.7 and 3.8, we can estimate the belief state update, the sequence probability and the conditional probability. The update of a belief state transitions between observations in the belief space and is defined as

$$\mathbf{b}_{t+1} = \frac{\mathbf{B}_{x_t} \cdot \mathbf{b}_t}{\sum_x \mathbf{b}_{\infty}{}^{\top} \cdot \mathbf{B}_x \cdot \mathbf{b}_t} \tag{3.10}$$

The estimation of the probability of a sequence given the belief states from the observable representation is recursively defined with the state update in Equation 3.10 as

$$\begin{aligned} p\left(x_1, \ldots, x_t\right) &= \mathbf{b}_{\infty}{}^{\top} \cdot \mathbf{B}_{x_t} \cdot \ldots \cdot \mathbf{B}_{x_1} \cdot \mathbf{b}_1 \\ &= \mathbf{1}^{\top} \cdot \left(\mathbf{U}^{\top} \cdot \mathbf{O}\right)^{-1} \cdot \left(\mathbf{U}^{\top} \cdot \mathbf{O}\right) \cdot A\left(x_t\right) \cdot \left(\mathbf{U}^{\top} \cdot \mathbf{O}\right)^{-1} \cdot \ldots \\ &\quad \cdot \left(\mathbf{U}^{\top} \cdot \mathbf{O}\right) \cdot A\left(x_1\right) \cdot \left(\mathbf{U}^{\top} \cdot \mathbf{O}\right)^{-1} \cdot \left(\mathbf{U}^{\top} \cdot \mathbf{O}\right) \cdot \pi \\ &= \mathbf{1}^{\top} \cdot A\left(x_t\right) \cdot \ldots \cdot A\left(x_1\right) \cdot \pi \end{aligned} \tag{3.11}$$

where the normalization from Equation 3.10 is left out of each multiplication for visibility. The conditional probability given the the belief states from the observable representation is defined as follows.

$$p\left(x_t \middle| x_1, \ldots, x_{t-1}\right) = \frac{\mathbf{B}_{x_t} \cdot \mathbf{b}_t}{\sum_x \mathbf{b}_{\infty}{}^{\top} \cdot \mathbf{B}_x \cdot \mathbf{b}_t} \tag{3.12}$$

The belief states from the observable representation given by Equations 3.6, 3.7 and 3.8 and the unitary matrix $\mathbf{U}$ form the Spectral HMM.

3.4 Outline of Spectral Learning

The Spectral Learning algorithm is an ***one-shot*** method that uses algebraic methods for learning. The algorithm uses an alternative way of inference and does not rely on maximizing the (log) likelihood as the EM algorithm does. The method used here is based on exploiting the latent structure between successive observations from future and past to form an observable representation. This approach is also related to the CCA from [9]. Here, in this context a Singular Value Decomposition (SVD) is used to exploit the linear related latent structure. Spectral Learning then uses an alternative belief state representation that implicitly contains information about the hidden states and is only based on observations and hance called observable representation. The very basic idea is to exploit the latent structure via subspace identification from [7] with a *thin* SVD. The *thin* SVD is used to obtain an orthonormal matrix which projects into the subspace. We need this subspace transformation because we assume that $s_2 > s_1$. For this exploitation of the latent structure the (*thin*) SVD is used because the HMM only has low order state sequences. Thus the name of the Spectral Learning algorithm derives from the inference procedure. According to the idea of subspace identification by [7], the matrix $\mathbf{U}$ spans a s_2-by-r subspace base of the correlation between past and future observations with $r \leq s_2$. From this point an observable representation is formed as explained in Section 3.1. The observable representation is then used for predicting the probability of a sequence or the conditional probability as shown in Section 3.2. This method is also proved to be consistent. The known downsides are the formation of the inverse probability matrix of the hidden states and the empirical estimates which require a lot of data.

```
Data: observation triples $(x_1, x_2, x_3)$, reduced rank $r$
Result: Spectral Hidden Markov Model parameterized by the observable representation $\mathbf{b}_1$,$\mathbf{b}_\infty$ and $\mathbf{B}_x$
Step 1.
    form empirical estimates $\hat{\mathbf{p}}_1$, $\hat{\mathbf{P}}_{2,1}$, $\hat{\mathbf{P}}_{3,x,1}$ of $\mathbf{p}_1$, $\mathbf{P}_{2,1}$, $\mathbf{P}_{3,x,1}$ from input data
Step 2.
    perform thin SVD of reduced rank $r$ on $\hat{\mathbf{P}}_{2,1}$ to obtain the matrix of left singular vectors corresponding to the $r$-largest singular values denoted by $\hat{\mathbf{U}}$
Step 3.
    compute model parameters $\mathbf{b}_1$,$\mathbf{b}_\infty$ and $\mathbf{B}_x$ from empirical estimates
```

Algorithm 1: Pseudocode of learning a Spectral Hidden Markov Model with the Spectral Learning algorithm.

3.5 Example on Spectral Learning

In this section the Spectral Learning algorithm will be tested with the small toy example from Section 2.3. The Spectral Learning algorithm for HMMs may look quite easy at a first glance but, in practice, is difficult to apply due to the many assumptions we made on the HMM. The Section 3.5.1 will reveal advantages and disadvantages of the spectral algorithm with an additional comparison to the Baum-Welch algorithm.

3.5.1 Positive and negative Examples

Lets consider the same setup for the HMM as in Section 2.3 for the correct use of the Spectral Learning algorithm. The observation sequence that was used for learning the Spectral HMM is the same as the EM algorithm introduced in Section 2.5. Thus, the Spectral Learning algorithm can be compared to the Baum-Welch algorithm for the particular example. The spectral algorithm was executed three times where the dimension for the learned Spectral HMM was reduced more in each run. The Kullback-Leibler divergence was computed to the true HMM for the resulting Spectral HMMs of different dimensions. They are depicted in Figure 3.4.

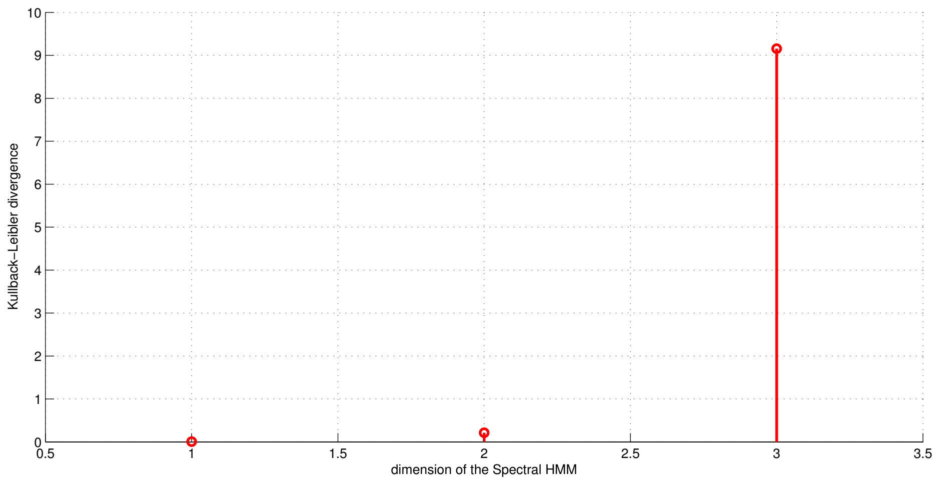


Figure 3.4: Evaluation of the Baum-Welch algorithm for three different sample sequence lengths. The KL on the y-axis is plotted against the number of hidden states used for learning on the x-axis.

Before going into detail of the results from Figure 3.4, lets have a look at the computed probabilitiy sequence for $(3,1,2)$. The results of the computed sequence probability with the corresponding KL are listed in the Table 3.1.

Spectral HMM dimension	1	2	3
sequence prob.: $p(3,1,2)$	0.2917	0.4043	**15.3359**
KL(Spectral HMM||HMM)	0.006869	0.121128	3.445319

Table 3.1: The probability for the sequence $(3,1,2)$ with the Kullback-Leibler divergence of each learned Spectral Hidden Markov Model.

Now we can see the major disadvantage of the Spectral Learning algorithm. We used 100 samples for learning and we do not get a probabilitiy out of the sequence $(3,1,2)$ for the Spectral HMM which has no reduced dimension, i.e. of dimension three. In such cases the Spectral HMM tends to produce values greater one or lower zero. We know that in original HMM the non-degeneracy condition holds. Therefore, due to the problematic case is the sampled sequence we used to learn. The unexpected behavior of the Spectral HMM appears because of the small amount of data used for learning such a relative high dimensional Spectral HMM. Not enough data means that the original HMM may have not reached its stationary distributed state and therefore is not ergodic. For this reason, we should consider longer sequences from which we remove the start of the sequence which could be non ergodic.

We redo the example with two sequences of length 500 where the first 50 observations are clipped off of each sequence.

The probabilitiy for the sequence $(3, 1, 2)$ and the KL for each Spectral HMM are listed in the Table 3.2. The results indicate that there is a lot more data needed when learning a higher dimension of the Spectral HMM. We can also see that more data results in better results and an increased stability of the Spectral Learning algorithm.

Spectral HMM dimension	1	2	3
sequence prob.: $p(3, 1, 2)$	0.3175	0.3341	0.3163
KL(HMM||Spectral HMM)	0.000992	0.000965	2.620611

Table 3.2: The probability for the sequence $(3, 1, 2)$ with the Kullback-Leibler divergence of each learned Spectral Hidden Markov Model learned from more data.

As an addition, [3] showed sample complexity bounds on the joint probability and conditional probability for a desired accuracy of the Spectral HMM. The proofs can also be found there.

4 Experiment and Result

The experiment of this work is based on discrete time series setting. The here observed environment is a 4-by-3 grid world that is supposed to be a highway in an abstract sense. An observable symbol is one complete instance of this grid to time step t. For this grid, the black fields indicate no occurence of cars and the gray fields indicate foreign cars, i.e. cars that are not controlled by the player. The white field represents the own car that was controlled throughout the process of gathering observable sequences. The whole grid environment is also shown in Figure 4.1. The goal of this experiment is to learn two behaviors of the own car solely with the Spectral Learning algorithm. However, the Spectral Learning algorithm should be able to seperate clearly between the two learned behaviors.

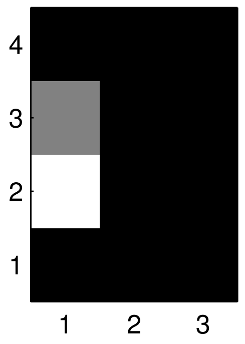
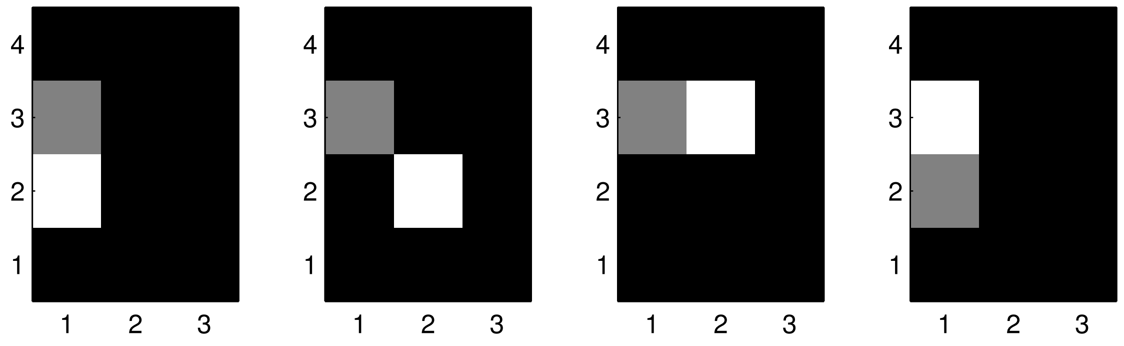


Figure 4.1: The grid environment of the experiment setup showing each of its elements that define the various observable symbols given by the various permutations of the elements. The white field is defined as the controlled car, the gray fields are foreign cars and the black fields are the street.

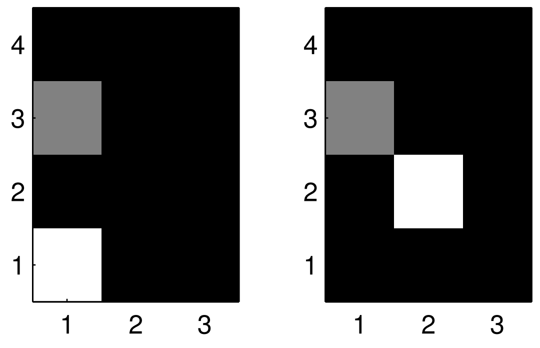
The data was collected with the car that was controlled by the player, i.e. the white field. In the following, we will try to correctly predict a sequence of the aggressive behavior which is one of the learned behaviors. The three step sequence predicted by the Spectral Learning algorithm should look like Figure 4.2. Here the first, second and third step is depicted in the Figures 4.2a, 4.2b and 4.2c respectively. The initial environment depicted in Figure 4.1 was chosen because it is supposed to trigger the aggressive behavior. The Spectral HMM was trained for two big sets of passive behavior and three rather small sets of aggressive behavior. The aggressive training sets had a length of 569, 740 and 400 observations and the passive training sets both had a length of 1950 observations. The behaviors are assumed to be a latent substructure of the observations and ,thus, we should be able to implicitly model by the Spectral Learning algorithm. The aggressive behavior made the controlled car, i.e. the white field, to stay near the upper left corner throughout and to overtake *any* car that blocks the way to the upper left corner. The neutral or passive behavior forced the controlled car to stay on the right lane throughout and to evade cars coming from the top.
We clipped off 100 observations from the start of each sequence of the training data to reach a stationary state distribution $\overline{\pi}$ as explained in Chapter 3. The collected data was used to learn a Spectral HMM with five dimensions. Such a low dimension was chosen to get clear results on the choice of the action. If we learn the Spectral HMM in a lower dimensional subspace, we have less variance in each dimension because the Spectral Learning algorithm picks the subspace domains with the most information and with the least noise. After learning the Spectral HMM, it was used



(a) The first most reasonably action of the aggressive behavior.
(b) The second most reasonably action of the aggressive behavior.
(c) The third most reasonably action of the aggressive behavior.
(d) The fourth most reasonably action of the aggressive behavior.

Figure 4.2: The observable sequence we want to predict successfully with Spectral Learning.

to check for the most likeli action to perform given the joint probability of the current state and each possible action to perform. We note here, that the Spectral HMM can only compute the joint probability for sequences that were passed to the Spectral Learning algorithm. Therefore, we assume that the collected data for learning a five dimensional subspace suffices as we will see in the evaluation. We start with the initial state shown in Figure 4.1 and denote it as x_{init}. Each executable action depicted in Figure 4.3 of the player car was performed. The most reasonably action should force the



(a) The executed action *down*. **(b)** The executed action *right*.

Figure 4.3: The two evaluated actions {*right*, *down*} starting from the initial position depicted in Figure 4.1.

car to go left because the aggressive behavior is defined by moving the own car to the upper left to the position (3,1) of the grid as shown in Figure 4.2d. The aggressive behavior also demands to overtake every car that is blocking the way. We proceed now by computing the joint probability of the initial position x_{init} depicted in Figure 4.1 with each executable action. These actions are denoted as x_{1_1} depicted in Figure 4.3a and x_{1_2} depicted in Figure 4.3b. The results of the joint probabilities $p(x_{1_i}, x_{init})$ from this first run are collected in the Table 4.1 where x_{1_i} denotes each executable action.

observation	action	probability	normalized probability
x_{1_1}	*down*	$4.876107 \cdot 10^{-16}$	0.0
x_{1_2}	*right*	$4.215054 \cdot 10^{-9}$	1.0

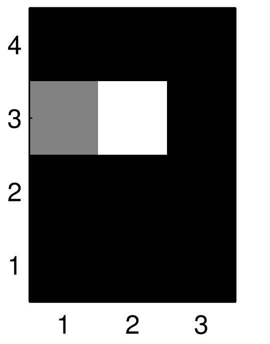
Table 4.1: The joint probability and normalized probability from the first run.

After the first run, the position of the car was updated to the state x_{1_2} because this was chosen as the most reasonably action. After this, we will evaluate all executable actions again starting from x_{1_2} with the same learned Spectral HMM. The possible actions are depicted in the Figure 4.4. Again the most reasonably action according to the current state x_{1_2} would be the aggressive behavior depicted in Figure 4.4a because the car came from the left and now has the possibility to overtake the other car. Please note, that this scenario occured very often while collecting the data for the aggressive behavior. For testing purposes on our assumption, we calculated the joint probability of $p\left(x_{2_i}, x_{1_2}\right)$ where x_{2_i} are the four actions depicted in Figure 4.4. The results of the joint probability for all four actions are listed in the Table 4.2.

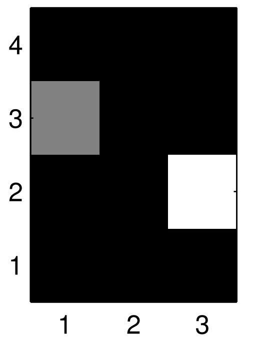
observation	action	probability	normalized probability
x_{2_1}	*up*	$6.877512 \cdot 10^{-4}$	1.0
x_{2_2}	*right*	$9.333473 \cdot 10^{-13}$	0.0
x_{2_3}	*down*	$7.070393 \cdot 10^{-10}$	0.0
x_{2_4}	*left*	$9.704818 \cdot 10^{-9}$	0.0

Table 4.2: The joint probability and normalized probability from the second run.

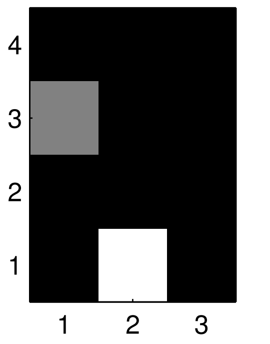
Now the controlled car cannot move up anymore because this was no allowed action. The car would now wait for the other car to move down so that it should be able to move left according to the learned aggressive behavior. After the car moved down, we start our last evaluation to find the last action from the set of executable actions depicted in Figure 4.5.



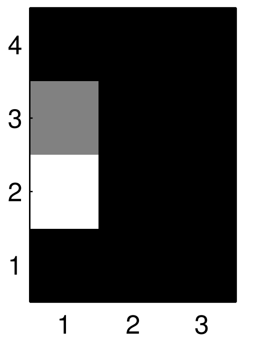
(a) The executed action *up*.



(b) The executed action *right*.



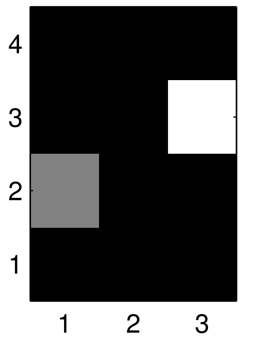
(c) The executed action *down*.



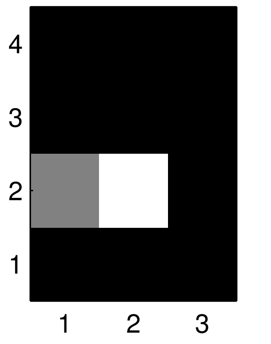
(d) The executed action *left*.

Figure 4.4: All evaluated actions {*up, right, left, down*} during the second run.

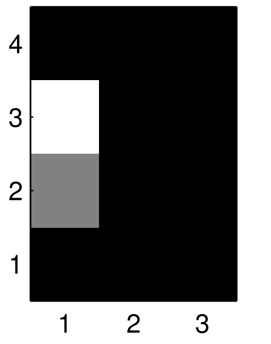
In this case, we would assume the action left denoted as x_{3_3} to be executed with the highest probability. This action



(a) The executed action *right*.



(b) The executed action *down*.



(c) The executed action *left*.

Figure 4.5: All evaluated actions {*right, down, left*} during the third run.

would match the aggressive behavior best which we want to imitate. The results of the joint probability $p\left(x_{3_i}, x_{2_1}\right)$ of each executable action denoted as x_{3_i} respectively can be found in Table 4.3.

observation	action	probability	normalized probability
x_{3_1}	*right*	$1.679045 \cdot 10^{-3}$	0.0382
x_{3_2}	*down*	$3.728679 \cdot 10^{-2}$	0.8483
x_{3_3}	*left*	$4.988087 \cdot 10^{-3}$	0.1135

Table 4.3: The joint probability and normalized probability from the third run.

The most likeli action of the third run is to go downwards according to the Table 4.3. We will have a discussion on these results in the next chapter.

5 Conclusion and Future Work

We start with a discussion and a small recapof the results and continue with the conclusion of this work. In our experiment, we tried to imitate one of two learned behaviors. The environment in Figure 4.1 provided was assumed to trigger the desired behavior. Given our assumptions on this behavior, we formulated a possible outcome in Figure 4.2 of the desired behavior and started to evaluate the most likeli actions in each scenario. We observed if the most likeli action would match the most reasonably action. The experiment nearly succeeded. The assumption on the last action in Figure 4.2d was violated. The discussion should reveal possible causes of why the last action in Figure 4.2d was violated. Before the last action in Figure 4.2d was evaluated the controlled car waited for the environment to change due to constraints. The controlled car, i.e. the white field, was not allowed to enter the top row and therefore had to wait to overtake the other car. This change of environment could have led to a change in the behavior regarding the Markov Property. Another possible source of the unexpected change in the behavior could be the repudiation of the non-degeneracy condition. Hence, the Spectral Learning algorithm was not able to seperate between the two behaviors and learned a mixture of the two behaviors, the aggressive driving behavior and passive driving behavior. The violation of the non-degeneracy condition is the most likeli cause for this behavior still assuming that there was enough data used for learning.
However, the Spectral Learning algorithm was able to predict all the prior sequence according to our assumptions. The results were learned with few dimensions to get very precise results. We can assume the results to be correct because the Spectral Learning algorithm learns globally by infering with moment estimation of the HMM. The Spectral Learning exploits the low rank structure of the HMM by assuming marginal independence. The low rank structure is exploited using a SVD on the empirical second moment which encodes the correlation between past and future observations. For learning the parameters of the HMM, we use OOMs as an observable factorization. By doing so, we can learn the parameters directly from observations but the downside is that we can get degenerated probabilities. The problem of the degenerated probabilities come from the alternative observable representation which contains the inverse of an estimate of a probability matrix. Thus if there is enough data for the Spectral Learning algorithm to learn, we get better empirical estimations on the moments, i.e. the second moment. This effect leads to better results and more stability.

The Spectral Learning algorithm is in general a very powerful algorithm for Imitation Learning (IL) in Machine Learning (ML). It is not only applicable to HMMs but also to latent tree models [10]. The current *state of the art* is to use Spectral Learning as a preparation step for the EM or Baum-Welch algorithm when there is little data provided. Spectral Learning looks very promising and is not limited to the time discrete case because HMMs can be extended to a continuous case with Hilbert space embeddings [11]. There is left a lot of space for the work on Spectral Learning itself including the relaxation of the very hard conditions of Spectral Learning and increased stability of the algorithm.

Bibliography

[1] H. Jaeger, “Observable operator models for discrete stochastic time series,” *Neural Computation*, vol. 12, no. 6, pp. 1371–1398, 2000.

[2] L. R. Rabiner, “A tutorial on hidden markov models and selected applications in speech recognition,” in *Proceedings of the IEEE*, pp. 257–286, 1989.

[3] D. Hsu, S. M. Kakade, and T. Zhang, “A spectral algorithm for learning hidden markov models,” *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1460–1480, 2012.

[4] C. M. Bishop and N. M. Nasrabadi, “Pattern recognition and machine learning,” 2006.

[5] T. K. Moon, “The expectation-maximization algorithm,” *Signal processing magazine, IEEE*, vol. 13, no. 6, pp. 47–60, 1996.

[6] B. Boots and G. J. Gordon, “An online spectral learning algorithm for partially observable nonlinear dynamical systems.,” in *AAAI*, 2011.

[7] P. Van Overschee and B. De Moor, “Subspace identification for linear systems: theory, implementation, applications,” *status: published*, 1996.

[8] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky, “Tensor decompositions for learning latent variable models,” *arXiv preprint arXiv:1210.7559*, 2012.

[9] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor, “Canonical correlation analysis: An overview with application to learning methods,” *Neural Computation*, vol. 16, no. 12, pp. 2639–2664, 2004.

[10] A. Anandkumar, K. Chaudhuri, D. Hsu, S. M. Kakade, L. Song, and T. Zhang, “Spectral methods for learning multivariate latent tree structure.,” in *NIPS*, pp. 2025–2033, 2011.

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

[12] M. Stamp, “A revealing introduction to hidden markov models,” *Department of Computer Science San Jose State University*, 2004.