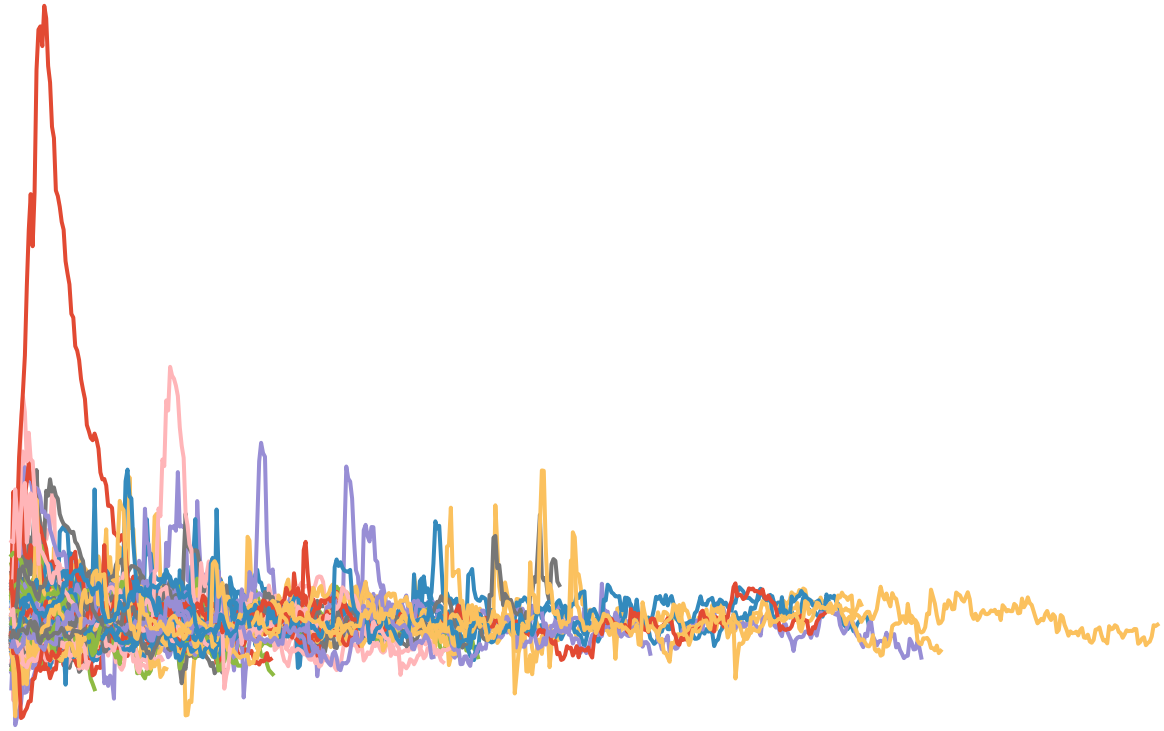
CHALMERS
UNIVERSITY OF TECHNOLOGY



Probabilistic Modelling of Sensors in Autonomous Vehicles

Autoregressive Input/Output Hidden Markov Models for Time Series Analysis

Master's thesis in Engineering Mathematics and Computational Science

Edvin Listo Zec

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Master's thesis 2017

Probabilistic Modelling of Sensors in Autonomous Vehicles

Autoregressive Input/Output Hidden Markov Models for Time Series Analysis

Edvin Listo Zec

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2017

Probabilistic Modelling of Sensors in Autonomous Vehicles
Autoregressive Input/Output Hidden Markov Models for Time Series Analysis
Edvin Listo Zec

Supervisor: Alexander Schliep, Department of Computer Science and Engineering
Advisor: Nasser Mohammadiha, Volvo Car Group
Examiner: Devdatt Dubhashi, Department of Computer Science and Engineering

Master's Thesis 2017
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Probabilistic graphical model visualising the autoregressive input/output hidden markov model.

Typeset in LaTeX
Gothenburg, Sweden 2017

Probabilistic Modelling of Sensors in Autonomous Vehicles
Autoregressive Input/Output Hidden Markov Models for Time Series Analysis
Edvin Listo Zec
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

Testing the quality of sensors in autonomous vehicles is crucial for safety verification. This is usually done by collecting a lot of data in many different settings. However, this can be very time consuming and expensive. Therefore, one is interested in virtual verification methods that simulate these situations, so many scenarios can be tested in parallel without actual hazards. In this thesis a generative model is created for the longitudinal errors in the sensors and an extension to the hidden Markov model, called autoregressive input/output hidden Markov model (AIOHMM) is implemented. In this extension the transition probabilities are conditioned on an input vector and the emissions are conditioned with the emissions at previous time steps, making it better suited for modelling long-term dependencies. We show that conditioning on the previous error is not enough to capture the behaviour of the errors, and that conditioning the transitions on an input is an important aspect of the model.

Keywords: Generative Model, Autonomous vehicle, Autoregressive, Input Output, Hidden Markov Model, Sensor Modelling, Time Series.

Acknowledgements

I would like to express my special thanks to my Volvo supervisor Nasser Mohammadiha for giving me the wonderful opportunity to write the thesis on this topic and for his guidance throughout this project. Thank you for the time spent on invaluable discussions and feedback. I would also like to thank my Chalmers supervisor Alexander Schliep for great feedback throughout the process from which many new ideas were created.

I give my thanks to Volvo Car Group for giving me the opportunity to write my thesis with them and for providing me with the necessary tools and the data in order to perform the project. Further I would like to give my thanks to the other master thesis students: Christopher, Henrik, Donal and Lucia. Thank you for all the discussions and especially for always wanting to have a tre-fika.

Edvin Listo Zec, Gothenburg, September, 2017

Contents

1
Introduction

During the last decades the development of machine learning algorithms has increased tremendously in a variety of different research areas. These algorithms are also becoming increasingly more important in the automotive industry in the development of advanced driver assistance systems (ADAS) and autonomous driving (AD) [1]. In order to monitor the current status of an environment, sensors are often used. For example, a sensor system is needed for detecting and identifying objects of interest around the vehicle, and thus the system plays a crucial role by monitoring the environment in order to make quick and safe decisions.

Volvo Cars' active safety department works with autonomous driving (AD) and highly automated driving (HAD) for cars. The cars have sensors that gather a lot of raw data that is processed and transformed into a list of objects with relevant information such as speed and position, but also weather conditions like temperature and current wind velocity etc. Functions are then designed using these parameters in order to perform adaptive cruise control or implement collision avoidance systems.

Volvo is interested in testing the quality of the sensors since they have to be evaluated before being deployed. This is usually done by collecting a lot of data in many different settings. However, this can be very time consuming and expensive. Therefore one is interested in virtual verification methods that simulate these situations, so many scenarios can be tested in parallel without actual hazards. In particular, it is of interest to look at the errors in the sensor data. Simulating the errors is of great interest since it could be used to better model the sensors in a virtual environment.

1.1 Problem overview

The aim of this thesis is to build a generative model of sensor data given by Volvo Car Group in order to model the error in the sensors. In particular, it is of interest to model the error of the longitudinal position of tracked objects from a host car as seen in figure 1.1. By generating artificial data from a probability distribution resembling the empirical data as good as possible, one is able to save both time and resources since driving and collecting data is an expensive task to perform. The artificial data can be used together with the real collected data to perform further analysis, for example establishing with statistical certainty that autonomous driv-

ing is at least as safe as human driving. In this project we will use empirical data and build a generative model from it, with the goal being to have the generative distribution as close to the empirical one as possible.

We consider an off-the-shelf sensor from Volvo Cars and propose a method to model it. The sensor is referred to as the production sensor in this thesis. The vehicle is also equipped with a light detection and ranging (Lidar) sensor which is used as the ground truth. Thus the goal of the thesis is to model the difference between the production sensor and the ground truth readings, with respect to longitudinal position. We define the longitudinal error as

$$lgtErr = lgtPos_{prod.sensor} - lgtPos_{groundtruth}.$$

In this project, Hidden Markov models (HMMs) and extensions thereof will be learned in an unsupervised fashion. HMMs in themselves can be viewed as an extension of simple Markov models, where the states are hidden (unobserved) while the output is visible (observed). Each state has a probability distribution over the output, so the generated output sequence from an HMM reveals information about the state sequences. The extended HMM implemented in this thesis is called Autoregressive Input/Output HMM (AIOHMM). In this model the observation probabilities are conditioned both on previous observations and some input feature. Further a time inhomogeneous transition matrix in incorporated. After having trained the AIOHMM on data given by Volvo, we can generate artificial data of the errors which can be used in a virtual environment for further analysis.

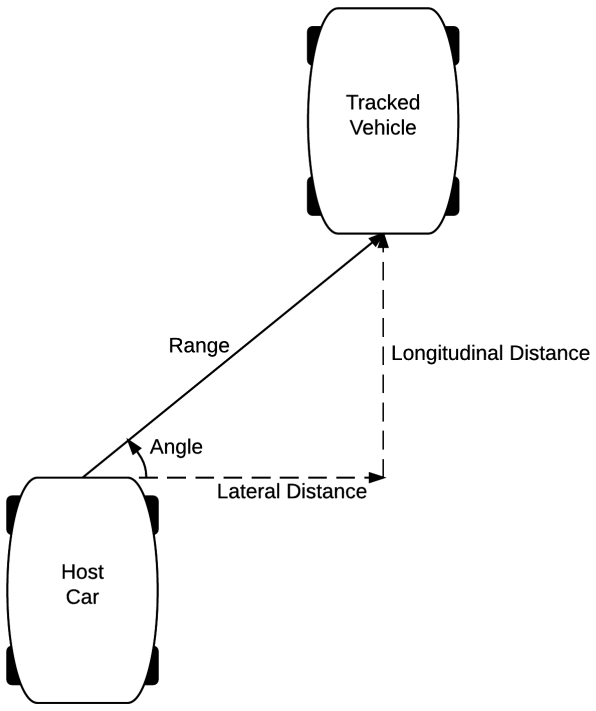


Figure 1.1: Illustration describing the setup and the variables.

1.2 Thesis outline

This thesis is composed into four main chapters. It begins with **Theory**, describing all necessary theory and algorithms required to understand the implemented model. Next is **Methods**, where the implemented methods and the general work process is described. The **Results and Discussion** chapter visualises the different generative models created and we discuss the obtained results. Lastly, the **Conclusions and future work** chapter summarises the main conclusions of the thesis, and we reflect over possible future works.

2
Theory

"All models are wrong; some models are useful".

— George Box, *Statistics for Experimenters*

This chapter contains all the theory required to understand the implemented model of the thesis. We start by presenting the expectation-maximisation (EM) algorithm in general, and continue to describe hidden Markov models and how to train them with EM. Lastly, an extended version of an HMM is presented.

2.1 The expectation-maximisation algorithm

The expectation-maximisation algorithm uses an iterative method in order to maximise the likelihood for parameters in a general statistical model, and is often used in order to train an HMM. The algorithm is usually summarised in two steps: the expectation (E) step (1-3) and the maximisation (M) step (4-5). Below follows a general overview of how the algorithm looks as described by [2], and we will later see how to apply it to HMMs.

Let $\boldsymbol{y}$ be some observed data, $\boldsymbol{x}$ the complete data and $p(\boldsymbol{y}|\boldsymbol{\theta})$ and $p(\boldsymbol{x}|\boldsymbol{\theta})$ two parametric densities. It is assumed that the whole data $\boldsymbol{x}$ can be modelled as a continuous random vector $\boldsymbol{X}$ with density $p(\boldsymbol{x}|\boldsymbol{\theta})$. Given that one only has the observed data $\boldsymbol{y}$, the goal is to find the maximum likelihood estimate (MLE) of $\boldsymbol{\theta}$, which is often maximised using log-likelihood of $\boldsymbol{y}$.

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{y}|\boldsymbol{\theta}).$$

This can however often times be difficult to solve for, and that is where the EM algorithm enters. With the EM we first make a guess about X and solve for the $\boldsymbol{\theta}$ that maximises the expected log-likelihood of X. Then we iteratively update $\boldsymbol{\theta}$ until a stopping criteria is reached.

Step 1: Set $k = 0$ and initialise $\boldsymbol{\theta}^{(k)}$.

Step 2: Given the observed data $\boldsymbol{y}$, calculate the conditional probability distribution $p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}^{(k)})$.

Step 3: Calculate the conditional expected log-likelihood Q:

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(k)}) = \int_{\mathcal{X}(\boldsymbol{y})} \log p(\boldsymbol{x}|\boldsymbol{\theta}) p(\boldsymbol{x}|\boldsymbol{y}, \theta^{(k)}) \, \mathrm{d}\boldsymbol{x} = E_{\boldsymbol{X}|\boldsymbol{y},\theta^{(k)}}[\log p(\boldsymbol{X}|\boldsymbol{\theta})].$$

Here $\mathcal{X}(\boldsymbol{y})$ denotes the closure of the set $\{\boldsymbol{x} : p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}) > 0\}$, which we assume does not depend on $\boldsymbol{\theta}$.

Step 4: Calculate $\boldsymbol{\theta}^{(k+1)} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(k)})$.

Step 5: Update $k := k + 1$ and return to Step 2. Stop criteria can be defined as needed, usually one chooses to stop when $\|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^{k}\| < \varepsilon$ for some $\varepsilon > 0$.

The EM algorithm guarantees convergence and that the next iteration is at least as good as the previous one. This means that one stops in a local maximum, and thus one usually initialises the EM algorithm by randomisation several times, and then chooses the $\boldsymbol{\theta}$ with the largest likelihood in order to get a model that yields the best local maximum.

The complete data $\boldsymbol{X}$ can in many cases be made up from the observed data $\boldsymbol{Y}$ added by some latent (hidden) data $\boldsymbol{Z}$ such that $(\boldsymbol{X}, \boldsymbol{Y}) = \boldsymbol{Z}$. In this case the conditional expected log-likelihood can be expressed as an integral over the $\boldsymbol{Z}$-domain $\mathcal{Z}$ since the randomness of the complete data will only come from the latent data $\boldsymbol{Z}$. Thus for $\boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{z})$ we get

$$\begin{aligned} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(k)}) &= \int_{\mathcal{X}} \log p(\boldsymbol{x}|\boldsymbol{\theta}) p(\boldsymbol{x}|\boldsymbol{y}, \theta^{(k)}) \, \mathrm{d}\boldsymbol{x} \\ &= \int_{\mathcal{X}} \log p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{\theta}) p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{y}, \boldsymbol{\theta}^{(k)}) \, \mathrm{d}\boldsymbol{x} \\ &= \int_{\mathcal{Z}} \log p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{\theta}) p(\boldsymbol{z}|\boldsymbol{y}, \boldsymbol{\theta}^{(k)}) \, \mathrm{d}\boldsymbol{z} \\ &= E_{\boldsymbol{Z}|\boldsymbol{y}, \boldsymbol{\theta}^{(k)}}[\log p(\boldsymbol{y}, \boldsymbol{Z}|\boldsymbol{\theta})]. \end{aligned}$$

2.2 Hidden Markov models

2.2.1 Introduction

Hidden Markov models (HMMs) have historically been one of the most popular models used when dealing with random sequences [3]. Let $(y_t)_{t=1}^T$ be sequential observations from some random sequence Y. An HMM assumes that there exists some underlying hidden sequence of states $(z_t)_{t=1}^T$ and that the elements of Y are conditionally independent given the state sequence $(z_t)_{t=1}^T$. The hidden states can take on one of N fixed values such that $Z_t \in \{1, 2, \ldots, N\}$.

An HMM works under two main assumptions.

1. The Markov property: that the conditional probability distribution of each hidden state Z_{t+1} given all its previous states is equal to its conditional probability distribution given only its previous state z_t.

$$p(Z_{t+1} = n|z_1, z_2, \ldots, z_t) = p(Z_t = n|z_t).$$

2. The observation y_{t+1} is independent of other observations and states, given the hidden state z_t

$$p(y_{t+1}|y_1, y_2, \ldots, y_{t-1}, z_1, z_2, \ldots, z_{t-1}) = p(y_t|z_t).$$

The parameter set of an HMM consists of

1. The initial probability $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_N]$, where $\pi_n = p(Z_1 = n)$. For the case with continuous state space, the state spaces have a corresponding initial probability density $\boldsymbol{\pi}$.
2. The state transition probability matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ that gives the probability of transitioning from state i to state j, $a_{ij} = p(Z_t = j|Z_{t-1} = i)$.
3. The probability distribution for the observations Y given the hidden state n.

$$b_n(y_t) = p(Y_t = y_t|Z_t = n).$$

We can thus summarise the parameter set for an HMM as $\boldsymbol{\theta} = \{\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{\pi}\}$

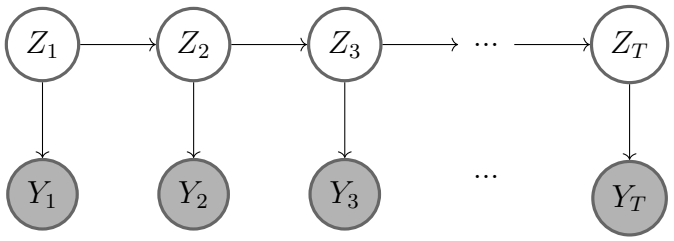


Figure 2.1: A graphical representation of an HMM.

2.2.2 The three problems

In order for the hidden Markov model to be applicable, three main problems have to be addressed as described by Rabiner [3].

1. Given the observation sequence $(y_t)_{t=1}^T$ and a model with parameter set $\boldsymbol{\theta} = \{\mathbf{A}, \mathbf{b}, \boldsymbol{\pi}\}$, how does one efficiently compute $p(\boldsymbol{y}|\boldsymbol{\theta})$? In other words, how does one compute the probability of the observation sequence given the model?
2. Given $(y_t)_{t=1}^T$ and $\boldsymbol{\theta}$, how does one choose a state sequence $(z_t)_{t=1}^T$ that in some best sense explains the observations?
3. How does one adjust the model parameters $\boldsymbol{\theta}$ to maximise $p(\boldsymbol{y}|\boldsymbol{\theta})$?

2.2.3 Solution 1: The forward-backward algorithm

The naive way of calculating $p(\boldsymbol{y}|\boldsymbol{\theta})$ would be by listing all possible state sequences of length equal to the number of observations, T. If we consider just one such state sequence $\boldsymbol{z} = z_1, z_2, \ldots, z_T$ we could calculate $p(\boldsymbol{y}|\boldsymbol{\theta})$ by summing the joint probability of $\boldsymbol{y}$ and $\boldsymbol{z}$, over all $\boldsymbol{z}$, i.e.

$$\sum_z p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{\theta}) = \sum_z p(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{\theta}) p(\boldsymbol{z}, \boldsymbol{\theta}).$$

However, by computing all this one would end up with a time complexity of $\mathcal{O}(TN^T)$. This can be realised by noting that at every time step we have N different state

sequences, resulting in a total of N^T different state sequences for which we have to do an order of $\mathcal{O}(T)$ calculations. In other words, this does not scale well even with small models. Enter the forward-backward algorithm (in actuality, only the forward algorithm will be used to solve the first problem).

Define the forward variable

$$\alpha_t(i) = P(y_1, y_2, \ldots, y_t, Z_t = i|\boldsymbol{\theta}), \tag{2.1}$$

i.e. the probability of observing the sequence $y_1, y_2, ..., y_t$ at time t and state i given the model $\boldsymbol{\theta}$. We call this the forward probability and can recursively calculate it by

1. Initialise $\alpha_1(i) = \pi_i b_i(y_1),\ i = 1, \ldots, N$
2. Calculate $\alpha_{t+1}(j) = b_j(y_{t+1}) \sum_{i=1}^{N} \alpha_t(i) a_{ij},\ t = 1, \ldots, T-1,\ \ j = 1, \ldots, N.$
3. Let $p(\boldsymbol{y}|\boldsymbol{\theta}) = \sum_{i=1}^{N} \alpha_T(i)$.

By performing time complexity analysis on this algorithm, we see that we have a complexity of $\mathcal{O}(N^2T)$ when calculating $\alpha_t(j)$, which is considerably smaller than $\mathcal{O}(TN^T)$.

We now introduce the backward part of the algorithm, which will be used in solution two and three. In the same fashion as above, we define the backward probability as

$$\beta_t(i) = p(y_{t+1}, y_{t+2}, \ldots, y_T|Z_t = i, \boldsymbol{\theta}). \tag{2.2}$$

In other words, $\beta_t(i)$ describes the probability of a partial sequence given the state and the model at time t. Just as before, we solve for $\beta_t(i)$ recursively.

1. Initialise (arbitrarily) $\beta_T(i) = 1,\ i = 1, \ldots, N.$
2. Calculate $\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(y_{t+1}) \beta_{t+1}(j),\ t = T-1, T-2, \ldots, 1,\ \ i = 1, \ldots, N.$

Just as with the forward probability, the time complexity of calculating the backwards probability is of order $\mathcal{O}(N^2T)$.

2.2.4 Solution 2: The Viterbi algorithm

Problem two is about finding the in some way "best" sequence of states $(z_t)_{t=1}^{T}$, and there are different ways of defining this problem. One way to define it is to find the individually most likely sequence by choosing the most likely state for each t. However, a problem arises if we have state transition probabilities $a_{ij} = 0$, since then an infeasible sequence could be chosen as the "optimal".

The most used criterion finds the single best sequence of states by maximising $p(\boldsymbol{z}|\boldsymbol{y}, \boldsymbol{\theta})$ with the Viterbi algorithm [3]. We have that $p(\boldsymbol{z}|\boldsymbol{y}, \boldsymbol{\theta})$ is proportional (as a function of $\boldsymbol{z}$) to $p(\boldsymbol{z}, \boldsymbol{y}|\boldsymbol{\theta})$ and it is therefore equivalent to maximise the latter. In order to fulfil this task, we define the quantity

$$\delta_t(i) = \max_{z_1, z_2, \ldots, z_{t-1}} p(z_1, z_2, \ldots, z_t = i, y_1, y_2, \ldots, y_t|\boldsymbol{\theta}),$$

which is the probability at time t of the most likely state sequence which ends in state i, that generates the observation sequence $\boldsymbol{y}$. We can calculate it with recursion

as

$$\delta_{t+1}(i) = \left[\max_i \delta_t(i)a_{ij}\right] b_j(y_{t+1}). \tag{2.3}$$

We create an array $\psi_t(i)$ that saves the argument that is maximised in (2.3) for each t and j in order to be able to retrieve the state sequence. In conclusion, the Viterbi algorithm returns the most probable state sequence $\boldsymbol{z}^*$ given an observations sequence $\boldsymbol{y}$. The Viterbi algorithm is summarised in Algorithm 1 and has a time complexity of $\mathcal{O}(N^2T)$.

Algorithm 1: Viterbi

input : Sequence of observations $\boldsymbol{y}$, state space $\boldsymbol{S} = \{1, \ldots, N\}$, initial probabilities $\boldsymbol{\pi}$, transition matrix $\boldsymbol{A}$, emission probabilities $\boldsymbol{b}$
output: Optimal (most likely) state sequence z_t^*
for *each state* $i \in \boldsymbol{S}$ **do**
  $\delta_1(i) = \pi_i b_i(y_1)$
  $\psi_1(i) = 0$
end
for *each time step* $t \in \{2, \ldots, T\}$ **do**
  for *each state* $j \in \boldsymbol{S}$ **do**
    $\delta_t(j) = \left[\max_i \delta_{t-1}(i)a_{ij}\right] b_j(y_t)$
    $\psi_t(j) = \arg\max_i \delta_{t-1}(i)a_{ij}$
  end
end
$P^* = \max_i \delta_T(i)$
$z_T^* = \arg\max_i \delta_T(i)$
for *each time step* $t \in \{T-1, T-2, \ldots, 1\}$ **do**
  $z_t^* = \psi_{t+1} z_{t+1}^*$
end
return z_t^*

2.2.5 Solution 3: The Baum-Welch method

The third problem is to find a way to adjust the model parameters $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{b}\}$ in order to maximise $p(\boldsymbol{y}|\boldsymbol{\theta})$. There is no known way of doing this analytically [3], however the Baum-Welch method has been developed and guarantees to find a local maximum. Further it is equivalent to the EM algorithm for learning the parameters of an HMM. We start with showing the update rules for the case with discrete emissions y_t, and then compose constraints so they are valid for the continuous case as well.

2.2.6 HMM with discrete emissions

In order to write down the full method, we start by defining

$$\gamma_t(i) = p(z_t = i|\boldsymbol{y}, \boldsymbol{\theta})$$

as the probability of being in state i at time t given the observation sequence $\boldsymbol{y}$ and the model $\boldsymbol{\theta}$. This can be rewritten using the forward and backward probabilities (2.1) and (2.2) and applying Bayes' theorem. We then get

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}.$$

We further define

$$\xi_t(i,j) = p(z_t = i, z_{t+1} = j|\boldsymbol{y}, \boldsymbol{\theta})$$

as the probability of being in state i at time t and state j and time $t+1$, given the observation sequence and the model. This can in the same fashion as before be written as

$$\begin{aligned}\xi_t(i,j) &= \frac{\alpha_t(i)a_{ij}\beta_{t+1}(j)b_j(y_{t+1})}{p(\boldsymbol{y}|\boldsymbol{\theta})} \\ &= \frac{\alpha_t(i)a_{ij}\beta_{t+1}(j)b_j(y_{t+1})}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i)a_{ij}\beta_{t+1}(j)b_j(y_{t+1})}.\end{aligned}$$

We can relate $\gamma_t(i)$ with $\xi_t(i,j)$ by summing over j.

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j).$$

Further, if we sum $\gamma_t(i)$ over the time steps t (leaving out the last step T) we get the over time expected number of times that state i is visited (equivalent to the number of times we transition from state i). To summarise, we have

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{ expected number of transitions from state } i.$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{ expected number of transitions from state } i \text{ to state } j.$$

We now have the tools to update the model parameters for the HMM. We thus get the expected frequency in state i at time $t = 1$,

$$\pi_i^* = \gamma_1(i).$$

The expected number of transitions from state i to state j divided by the expected number of transitions from state i,

$$a_{ij}^* = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}. \tag{2.4}$$

And finally, the expected number of times in state j and observing $y_t = v_k$ divided by the expected number of times in state j,

$$b_j^*(k) = \frac{\sum_{t=1}^{T-1} \mathbb{1}_{y_t = v_k}\gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}. \tag{2.5}$$

2.2.7 HMM with continous emissions

In equation (2.5) we assumed that the emission y_t from the HMM was discrete. Oftentimes however it is of interest to model continuous outputs. This extension of the HMM lets us use a probability density function (pdf) to model the emission probabilities and we can formulate update rules for the parameters of the pdf, given certain restrictions. Let the general representation of the pdf be a finite mixture of the form

$$b_j(Y) = \sum_{m=1}^{M} c_{jm}\mathcal{P}(y_t, \mu_{jm}, \Sigma_{jm}),\ j = 1, \ldots, N,$$

where Y is the vector being modelled. Here c_{jm} is the mixture coefficient for the mth mixture in state j and $\mathcal{P}$ is any log-concave or elliptically symmetric density (like the Gaussian) with mean μ_{jm} and covariance matrix Σ_{jm}. We must restrict the mixture coefficients c_{jm} to satisfy

$$\sum_{m=1}^{M} c_{jm} = 1, \quad j = 1, \ldots, N$$

$$c_{jm} \geq 0, \quad j = 1, \ldots, N,\ m = 1, \ldots, M.$$

Further, the pdf must be normalised such that

$$\int_{-\infty}^{\infty} b_j(x)\,\mathrm{d}x = 1, \quad j = 1, \ldots, N.$$

By modelling the outputs with a pdf of this kind, we can in practice approximate any finite, continuous density function arbitrarily closely. This makes this model applicable to a wide range of problems.

The following closed form update rules can be derived for the parameters c_{jm}, μ_{jm} and Σ_{jm}. [3]

$$c_{jm}^* = \frac{\sum_{t=1}^{T} \gamma_t(j,m)}{\sum_{t=1}^{T}\sum_{m=1}^{M} \gamma_t(j,m)},$$

$$\mu_{jm}^* = \frac{\sum_{t=1}^{T} \gamma_t(j,m) Y_t}{\sum_{t=1}^{T} \gamma_t(j,m)},$$

$$\Sigma_{jm}^* = \frac{\sum_{t=1}^{T} \gamma_t(j,m)(Y_t - \mu_{jm})(Y_t - \mu_{jm})^\intercal}{\sum_{t=1}^{T} \gamma_t(j,m)}.$$

Here $\gamma_t(j,m)$ is the probability of being in state j at time t with the mth mixture component accounting for Y_t

$$\gamma_t(j,m) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}\right]\left[\frac{c_{jm}\mathcal{P}(Y_t, \mu_{jm}, \Sigma_{jm})}{\sum_{k=1}^{M} c_{jk}\mathcal{P}(Y_t, \mu_{jk}, \Sigma_{jk})}\right]$$

The transition probabilities are updated the same as before, according to equation (2.4).

Baum et al. [4] have shown that one of two things will happen by updating the parameter set $\boldsymbol{\theta}$ with the Baum-Welch method.

1. The current model $\boldsymbol{\theta}$ is in a critical point of the likelihood function.
2. The updated model $\boldsymbol{\theta}^*$ is more likely than $\boldsymbol{\theta}$, i.e.

$$p(\boldsymbol{y}|\boldsymbol{\theta}^*) > p(\boldsymbol{y}|\boldsymbol{\theta}).$$

The Baum-Welch method is used iteratively, until a stopping criterion is met and the end result is the maximum likelihood estimate of the HMM. Often one chooses to stop when the difference between the current and previous model is small. Just as with the EM algorithm, only a local maximum is guaranteed. Thus it is common to initialise at random several times, then performing the Baum-Welch method for all initialisations and lastly choose the best model.

2.2.8 HMMs with multiple sequences

All the previous equations assume that the parameter updates are performed with a single observation sequence. However, one would oftentimes want to train an HMM based on multiple sequences, $\boldsymbol{Y} = \{\boldsymbol{Y^{(1)}}, \boldsymbol{Y^{(2)}}, \ldots, \boldsymbol{Y^{(k)}}\}$, in order to get good parameter updates. In practice, no additional complexity is added to the algorithm by introducing multiple sequences [5], [3]. The forward and backward probabilities as well as the observation probability are calculated in the same fashion as before, for each training sequence $\boldsymbol{Y^{(k)}}$.

2.3 Autoregressive Input/Output HMM

In this section we will look at an extension of the simple HMM, called autoregressive input/output HMM (AIOHMM). In an AIOHMM the emissions and the transitions are in addition to a regular HMM also conditioned on some input data and previous outputs. The sensor data we work with (longitudinal and latitudinal error of the sensors) could perhaps be better modelled by taking into account external inputs of the vehicle, like the speed or acceleration. Also, the data indicates that the error is autoregressive, so conditioning it on previous errors in time should result in a better model. A Gaussian is used to model the emissions, with a time dependent mean in order to better model the longterm dependencies.

Bengio and Frasconi were likely first with describing an input-output HMM (IOHMM) [6]. Their architecture conditions the probabilities on some input vector in order to give a time dependence for both the state transitions and the emissions. However, the architecture can be further extended to capture the autoregressiveness of the data by also conditioning the emission and transition probabilities on previous emissions. This results in the Autoregressive Input-Output HMM, which we implement as proposed by Jain et al. [7].

Let X_t be the input at time step t. We use a different notation than that in their paper for the states and outputs: let Z_t be the hidden state and Y_t be the output. Analogous independence assumptions as for an HMM are made, and we have the following equations.

- The transition probability is parameterized with a log-linear function.

$$\psi_{ijt} = p(Z_t = j | Z_{t-1} = i, X_t; w_{ij}) = \frac{\exp(w_{ij} X_t)}{\sum_{\ell \in S} \exp(w_{i\ell} X_t)}$$

- The emission probabilities are parameterized with the normal distribution.

$$b(Y_t) = p(Y_t | Z_t = i, X_t, Z_{t-1}; \mu_{it}, \Sigma_i) \sim \mathcal{N}(Y_t | \mu_{it}, \Sigma_i).$$

In order to capture the autoregressiveness of the data, the mean of the Gaussian is further modelled as

$$\mu_{it} = \mu_i(1 + a_i X_t + b_i Y_{t-1}),$$

where a_i and b_i are learnt parameters for all states. Thus we can summarise the learning parameters of the AIOHMM as

$$\theta_i = \{\mu_i, a_i, b_i, \Sigma_i, w_{ij} | j \in S\}.$$

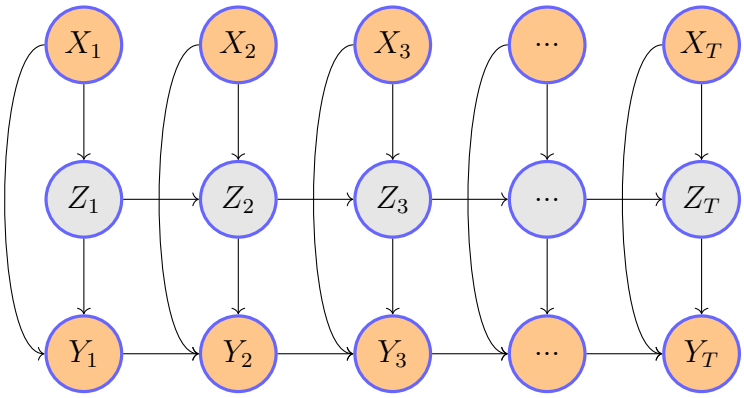


Figure 2.2: Graphical representation of an AIOHMM

2.3.1 Learning the parameters

The EM algorithm is used in order to learn the parameters, as described by Jain et al [7]. In the E-step the lower bound of the data log-likelihood is calculated. Let $\ell_c(\theta; \mathcal{D}_c)$ be the log-likelihood of the complete data

$$\mathcal{D}_c = \{X_{1,n}^{K_n}, Y_{1,n}^{K_n}, Z_{1,n}^{K_n} | n = 1, \ldots, N\},$$

consisting of N different sequences each of lengths K_n. We have that

$$\begin{aligned}
\ell_c(\theta; \mathcal{D}_c) &= \sum_{n=1}^{N} \log P(Y_{1,n}^{K_n}, Z_{1,n}^{K_n} | X_{1,n}^{K_n}; \theta) = \\
&= \sum_{n=1}^{N} \sum_{t=1}^{K_n} \sum_{j \in S} \mathbb{1}(Z_t = j) \log P(Y_t | Z_t = j, Y_{t-1}, X_t) \\
&+ \sum_{n=1}^{N} \sum_{t=1}^{K_n} \sum_{i,j \in S} \mathbb{1}(Z_t = j, Z_{t-1} = i) \log P(Z_t = j | Z_{t-1} = i, X_t).
\end{aligned}$$

Thus the E-step consists of calculating

$$Q(\theta, \hat{\theta}) = E[\ell_c(\theta, \mathcal{D}_c|\hat{\theta}, \mathcal{D}]. \tag{2.6}$$

To do this we compute

$$E[\mathbb{1}(Z_t = j)|\hat{\theta}, \mathcal{D}] = P(Z_t = j|Y_1^K, X_1^K; \hat{\theta}) = \gamma_{jt},$$
$$E[\mathbb{1}(Z_t = j, Z_{t-1} = i)|\hat{\theta}, \mathcal{D}] = P(Z_t = j, Z_{t-1} = i|Y_1^K, X_1^K; \hat{\theta}) = \xi_{ijt}.$$

Both γ_{jt} and ξ_{ijt} are calculated using the forward-backward algorithm as described in Section 2.2.6. We thus can expand equation (2.6) as

$$\begin{aligned} Q(\theta, \hat{\theta}) = &\sum_{n=1}^{N}\sum_{t=1}^{K_n}\sum_{j\in S}\gamma_{jt}\log\mathcal{N}(Y_t|\mu_{jt}, \Sigma_j) \\ &+ \sum_{n=1}^{N}\sum_{t=1}^{K_n}\sum_{i,j\in S}\xi_{ijt}\log\psi_{ijt}. \end{aligned}$$

In the M-step we maximise equation (2.6), and update the parameter set as

$$\theta = \arg\max_{\boldsymbol{\theta}} Q(\theta, \hat{\theta}).$$

All parameters except for w_{ij} are optimised by deriving their closed form updates. The transition parameter w_{ij} is updated numerically with gradient descent. We define the objective function as

$$L = -\sum_{n=1}^{N}\sum_{t=1}^{K_n}\sum_{i,j\in S}\xi_{ijt}\log\psi_{ijt}. \tag{2.7}$$

This can be recognised as the cross-entropy error function for multi-class output [8]. The gradient of L with respect to w_{ij} is known as

$$\frac{\partial L}{\partial w_{ij}} = X_t\left(-\xi_{ijt} + \psi_{ijt}\sum_{n=1}^{N}\sum_{t=1}^{K_n}\sum_{i,j\in S}\xi_{ijt}\right) = X_t\left(-\xi_{ijt} + \psi_{ijt}\sum_{n=1}^{N}\sum_{t=1}^{K_n}\sum_{j\in S}\gamma_{jt}\right). \tag{2.8}$$

The parameter w_{ij} is thus updated with gradient descent using equations (2.7) and (2.8). The update rules for the other parameters are described below. Let $c_{it} = 1 + a_i X_t + b_i Y_{t-1}$ such that $\mu_{it} = c_{it}\mu_i$. We then get the following.

$$a_i = \left[\sum_{n=1}^{N}\sum_{t=1}^{K_n}\gamma_{it}X_t X_t^\intercal\right]^{-1}\sum_{n=1}^{N}\sum_{t=1}^{K_n}\gamma_{it}\left[\frac{X_t Y_t^\intercal \Sigma_i^{-1}\mu_i}{\mu_i^\intercal\Sigma_i^{-1}\mu_i} - X_t - X_t Y_{t-1}^\intercal b_i\right]$$

$$b_i = \left[\sum_{n=1}^{N}\sum_{t=1}^{K_n}\gamma_{it}Y_{t-1}Y_{t-1}^\intercal\right]^{-1}\sum_{n=1}^{N}\sum_{t=1}^{K_n}\gamma_{it}\left[\frac{Y_{t-1} Y_t^\intercal \Sigma_i^{-1}\mu_i}{\mu_i^\intercal\Sigma_i^{-1}\mu_i} - Y_{t-1} - Y_{t-1}X_t^\intercal a_i\right]$$

$$\mu_i = \frac{\sum_{n=1}^{N}\sum_{t=1}^{K_n} c_{it}\gamma_{it}Y_t}{\sum_{n=1}^{N}\sum_{t=1}^{K_n} c_{it}^2\gamma_{it}}$$

$$\Sigma_i = \frac{\sum_{n=1}^{N}\sum_{t=1}^{K_n}\gamma_{it}(Y_t Y_t^\intercal + c_{it}^2\mu_i\mu_i^\intercal - c_{it}Y_t\mu_i^\intercal - c_{it}\mu_i Y_t^\intercal)}{\sum_{n=1}^{N}\sum_{t=1}^{K_n}\gamma_{it}}$$

3

Methods

3.1 Implementation

The main model implementation was done in Python 3.6 and the Autoregressive Input/Output HMM was based on Matlab code by Jain et al [7]. Three main model types were implemented.

1. A simple HMM with Gaussian emissions.
2. An AIOHMM with a homogeneous transition matrix (h-AIOHMM), where only the emissions were conditioned on an input.
3. A full AIOHMM where both the emissions and the state transitions were conditioned on an input.

All models were trained using a training data set containing 11845 sequences of tracked objects, with a maximum sequence length of approximately 900. Several models were trained for each model type, where the initial parameters and number of states as well as training sequences differed between models. Popular model selection methods are the Akaike information criterion (AIC) and the Bayesian information criterion (BIC). However, it can be misleading to use these criteria for HMMs since the k parameter that penalises the model size is increasing quadratically with the number of states in an HMM. This can lead to a stronger bias towards smaller models than is necessary. Thus the choosing of the number of states becomes more of an art than a science. In our case, the hidden states have no physical interpretation and different number of states were tested arbitrarily.

The input variable X_t was chosen by first calculating the pairwise correlations between all variables and then removing all column variables that correlated more than 60%. The input X_t was then selected as the variable that correlated highest with the output variable Y_t. Two inputs were chosen from this correlation analysis. The first one being the absolute angle between the ego vehicle and the tracked object. The second one being the lifetime of the object (the duration of which it was being tracked). All sequences of length less than 20 were filtered out before training since it is of more interest to model longer time series.

3.2 Model evaluation

After training, all models are evaluated and compared. The goal is to sample from the models, and yield a distribution as close to the empirical data as possible. A validation set that has never been seen by the model is used as the empirical distribution. Comparing two probability distributions can be done in a lot of different ways and extensive work has been done in this field [9]. In this thesis we discuss two measures, namely the Kullback-Leibler divergence and the Jensen-Shannon divergence. The goal of the model evaluation is to measure how close the generative model is to empirical data, in order to find the best model as possible.

3.2.1 Kullback-Leibler divergence

The Kullback-Leibler (KL) divergence can be used in order to evaluate the different models. The Kullback-Leibler divergence between to probability distributions P and Q is defined as

$$K(P\|Q) = \sum_x \log\left(\frac{p(x)}{q(x)}\right) p(x),$$

and can intuitively be thought of as a distance between the two distributions [10]. However, it is important to note that it in actuality is not a true metric as it is not symmetric, $K(P\|Q) \neq K(Q\|P)$, and as the triangle inequality does not hold.

The KL divergence has its deficiencies. The distribution P could generate samples that have zero probability for the Q distribution, which results in an infinite KL divergence.

3.2.2 Jensen–Shannon divergence

It is possible to smooth and symmetrize the KL divergence to yield the Jensen–Shannon divergence [11]. Let $M = \frac{1}{2}(P+Q)$. The Jensen-Shannon divergence is then defined as

$$JSD(P\|Q) = \frac{1}{2}K(P\|M) + \frac{1}{2}K(Q\|M).$$

It is symmetric, $JSD(P\|Q) = JSD(Q\|P)$, and is guaranteed to be finite. Further, its square root is a true metric called the Jensen-Shannon distance (JSd).

The Jensen-Shannon distance together with the log-likelihood will be used in order to evaluate models. A low JSd together with a high log-likelihood will indicate that a model is good. However, we won't be able to only rely on these two metrics for model evaluation since a bad model could theoretically have a small JSd and a high log-likelihood. Thus we will also have to perform visual inspection of the generated data in order to guarantee that the model is appropriate.

4

Results and discussion

In this chapter we present the results from different implemented models. We will compare HMMs with AIOHMMs with and without a homogeneous transition matrix and discuss the results. Histograms of the generated data is presented and compared to a validation data set never seen by the model. Further, we visualise samples from some chosen models.

Note that all numbers in the histograms and time series plots have been scaled due to confidentiality.

4.1 Comparison of models

In figure 4.1 we see 100 samples of real data. Each colour represent a tracked object. Note that the error is large in the beginning, when the sensors just have identified an object, an goes towards zero with time. We can also observe that the error seems to increase at the end of a sequence, i.e. when a tracked object is escaping from the sensor range.

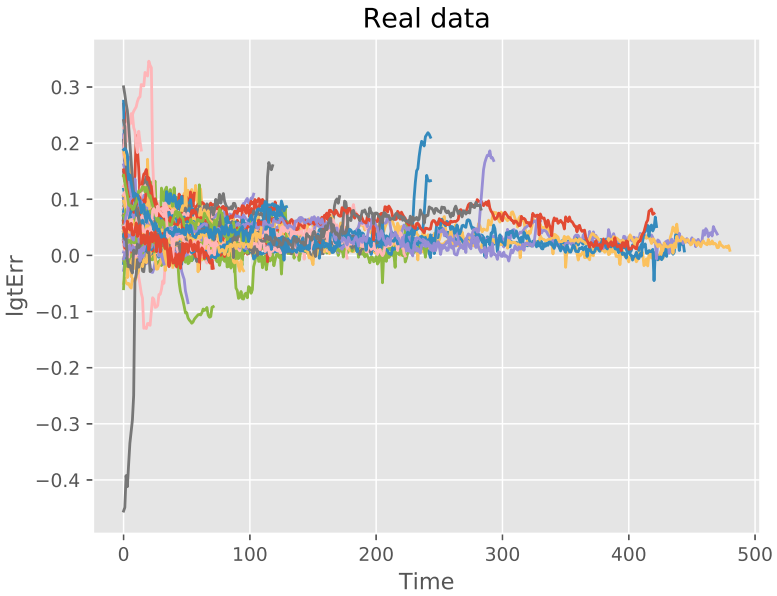
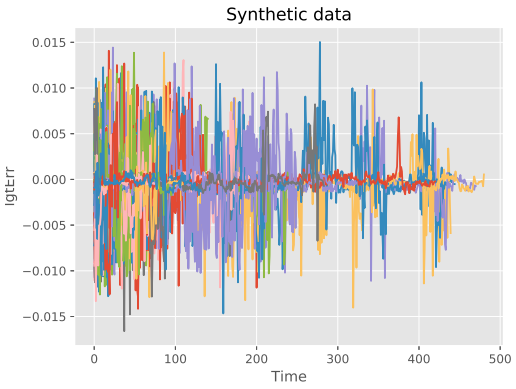


Figure 4.1: 100 samples of the data. Each colour represents a tracked object.

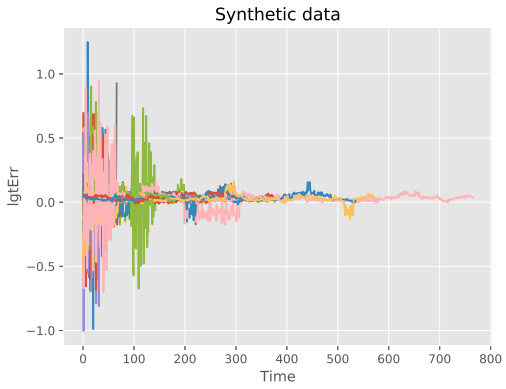
4.1.1 Simple HMMs

In figure 4.2 we see samples from four different simple HMMs. All four models were trained on the whole data set, with different number of states. From visual inspection we see that none of the models capture the true nature of the errors. However, in figure 4.2b and to some extent in figure 4.2d we see the 10 and 20 state HMMs respectively capturing the high variance in the beginning and decreases towards the end. Not too surprisingly the simple HMMs do not capture the autoregressiveness of the data due to the conditional independence given the states of the emissions. In figure 4.3 we see histograms of the validation data set (red) and the respective samples from the different models (blue).

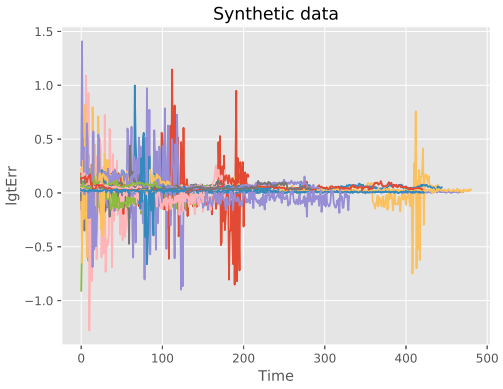
Note the distribution seen in figure 4.3b, where the 10 state HMM captures the validation distribution with a Jensen-Shannon distance of 0.17. This is a drawback of the evaluation method since it is permutation invariant, meaning that we will get the same distribution by rearranging the generated values (i.e. loosing the time dimension). Thus a good model will always have a low JSd, however a low JSd does not imply a good model as seen in this case. It is therefore still important to look at the log-likelihood and perform visual inspection of the generated data.



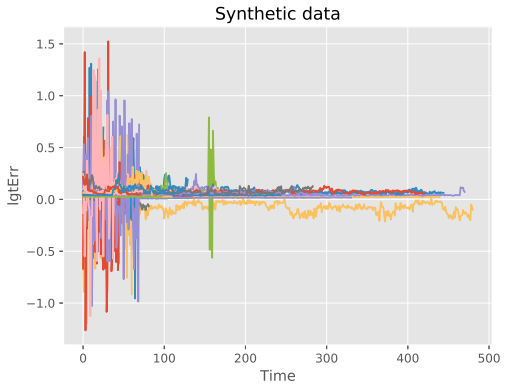
(a) HMM with 5 states.



(b) HMM with 10 states.

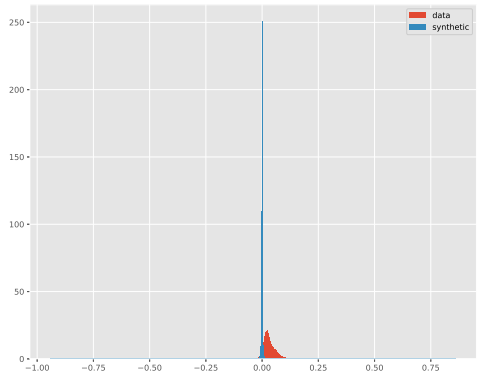


(c) HMM with 15 states.

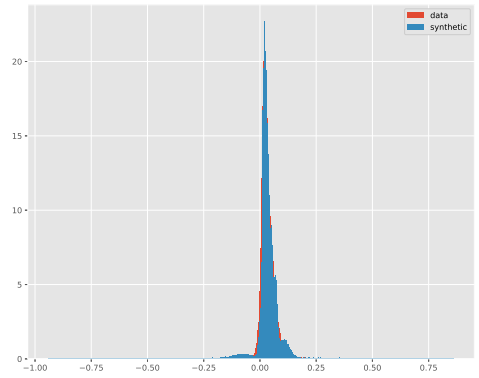


(d) HMM with 20 states.

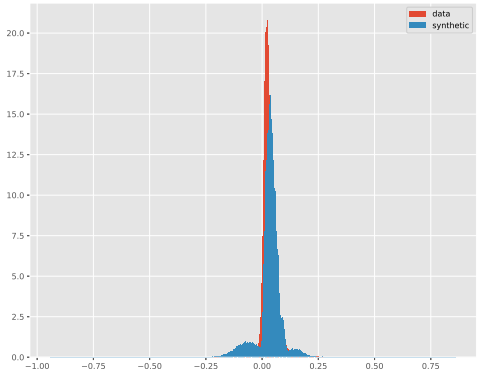
Figure 4.2: 100 samples from simple HMMs with different number of states.



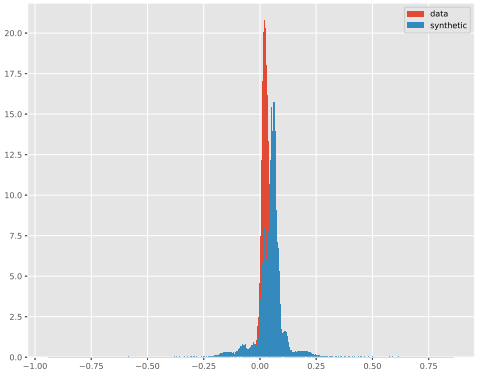
(a) HMM with 5 states, JSd: 0.90.



(b) HMM with 10 states, JSd: 0.17.



(c) HMM with 15 states, JSd: 0.27.



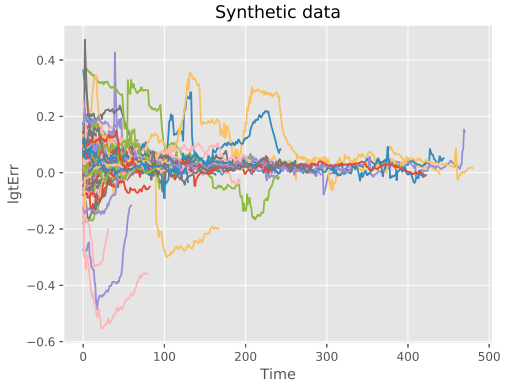
(d) HMM with 20 states. JSd: 0.37

Figure 4.3: Histogram of validation data set (red) and samples from HMMs blue).

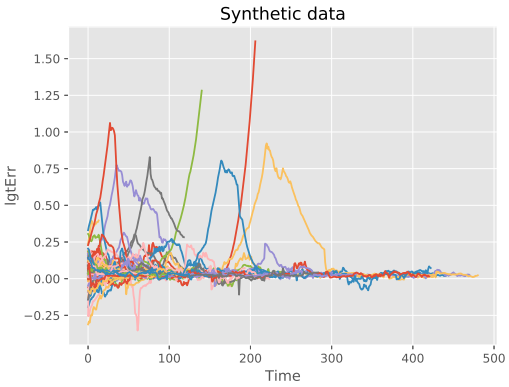
4.1.2 Homogeneous AIOHMMs

In figure 4.4 we see samples from different h-AIOHMMs (time-homogeneous transition matrices). From visual inspection we can see that the smoothness, autoregressiveness and long-term dependencies of the true data are much better captured than in the case of the simple HMMs. The distributions as seen in figure 4.5 are close to the true validation distribution, with a lowest Jensen-Shannon distance of 0.17 with a 9-state model with 2 inputs. It can be seen that the models are consistently having problems with the tails of the distributions.

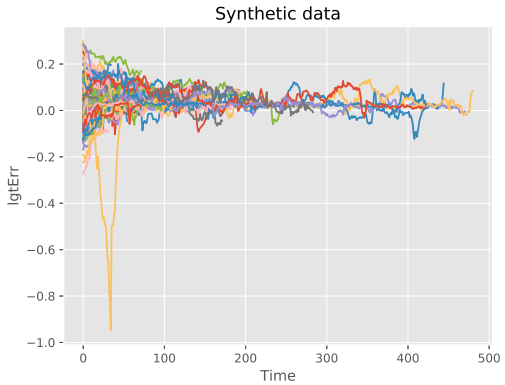
As with the simple HMMs, all models here were trained on the whole data set with not much longer computational time needed. Thus we see that by adding the error at the previous time step together with an input, the model is able to capture the behaviour of the true data well.



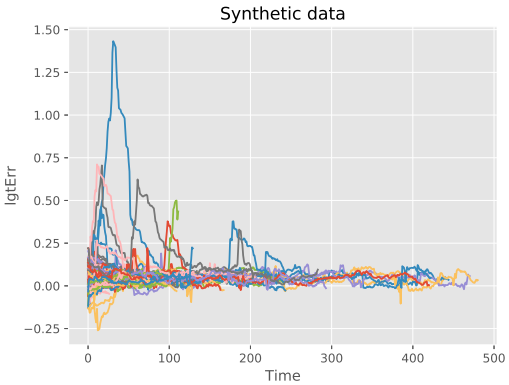
(a) 9 states with 2 inputs. JSd: 0.17.



(b) 7 states with 1 input (lifetime). JSd: 0.18.

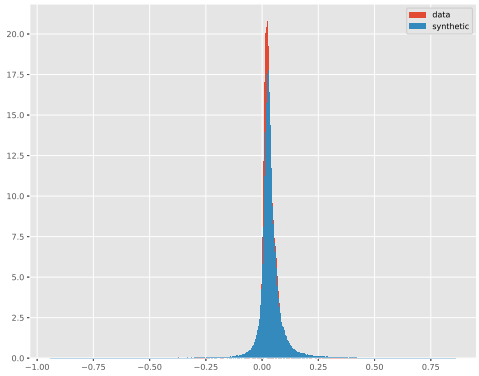


(c) 6 states with 1 input (lifetime). JSd: 0.24.

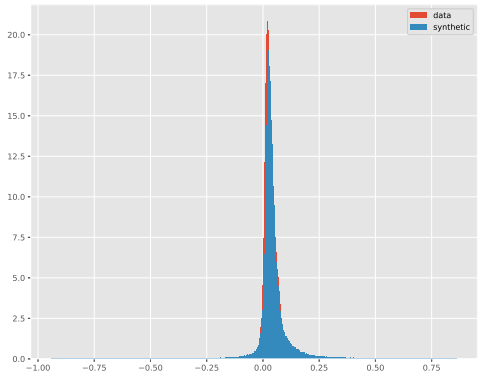


(d) 5 states with 1 input (absAng). JSd: 0.26.

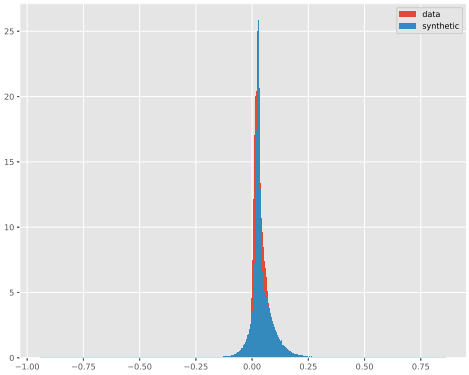
Figure 4.4: 100 samples from AIOHMMs with homogeneous transition matrices, different number of states and inputs.



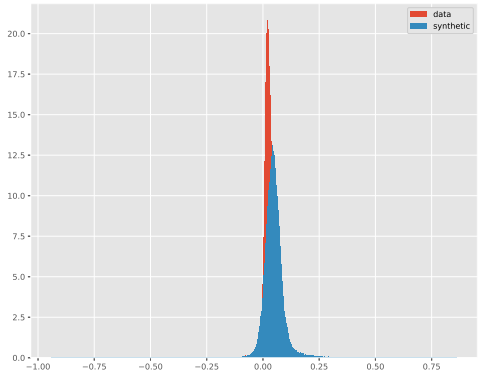
(a) 9 states with 2 inputs. JSd: 0.17.



(b) 7 states with 1 input (lifetime). JSd: 0.18.



(c) 6 states with 1 input (lifetime). JSd: 0.24.



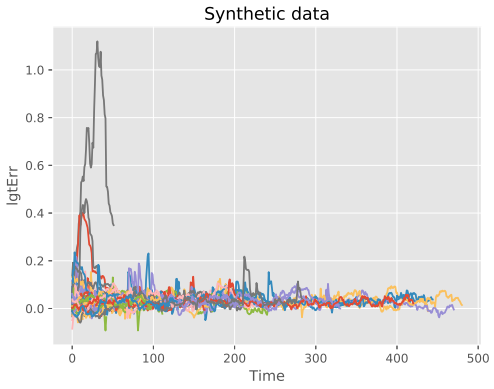
(d) 5 states with 1 input (absAng). JSd: 0.26

Figure 4.5: Histogram of validation data set (red) and samples from different h-AIOHMMs (blue).

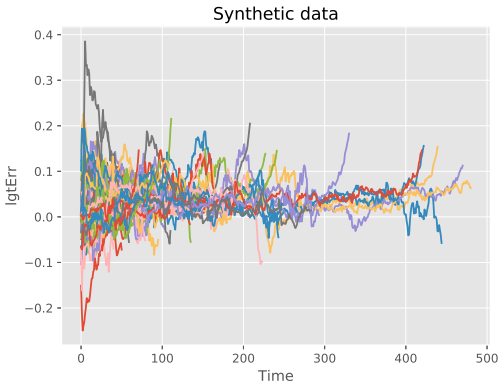
4.1.3 AIOHMMs

In figure 4.6 we see samples from different AIOHMMs (inhomogeneous transition matrices). Here the models were only trained on about 200 sequences out of the 11000 in the whole data set due to the gradient descent being computationally heavy. The h-AIOHMMs converged in about 30-60 minutes depending on the number of states when running 40 iterations of EM, whereas the AIOHMMs took several days to converge for only 7 iterations of EM. In figure **??** we see the histograms of the validation data set as compared to the respective models.

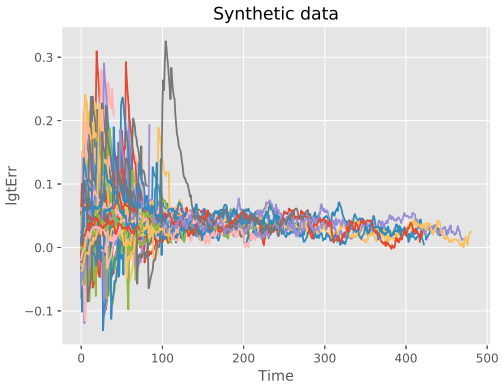
We see that including a transition matrix that changes over time is in most cases able to capture the smoothness of the data, even tough much less data were used. In terms of Jensen-Shannon distance, the 5 state AIOHMM with absolute angle as input in figure 4.6a performed the best. By visual inspection we see that it shares aspects of the true data in a smooth fashion, capturing the important parts such as high error in the beginning and then going towards zero.



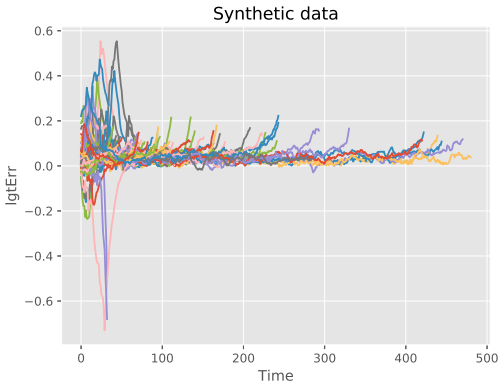
(a) AIOHMM with 5 states and 1 input (absAng). JSd: 0.13.



(b) AIOHMM with 4 states and 2 inputs. JSd: 0.19.

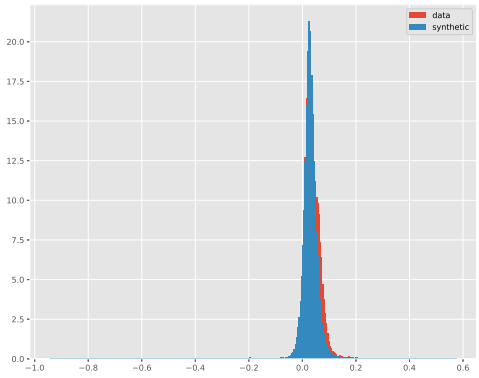


(c) AIOHMM with 3 states and 1 input (lifetime). JSd: 0.21.

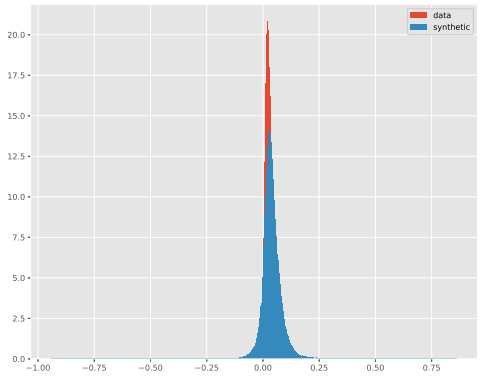


(d) AIOHMM with 5 states and 2 inputs. JSd: 0.22.

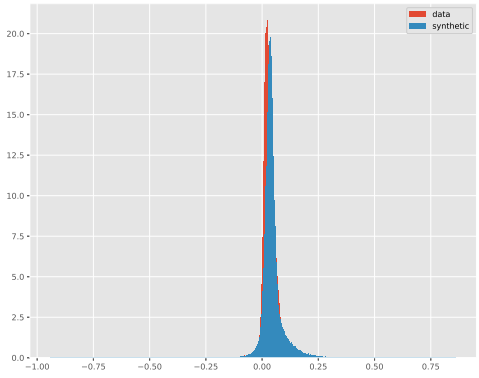
Figure 4.6: 100 samples from AIOHMMs with inhomogeneous transition matrices, different number of states and inputs.



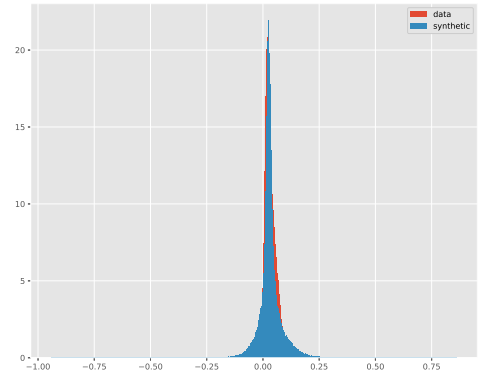
(a) AIOHMM with 5 states and 1 input (absAng). JSd: 0.13.



(b) AIOHMM with 4 states and 2 inputs. JSd: 0.19..



(c) AIOHMM with 3 states and 1 input (lifetime). JSd: 0.21.



(d) AIOHMM with 5 states and 2 inputs. JSd:: 0.22.

Figure 4.7: Histogram of validation data set (red) and samples from different AIOHMMs (blue).

4.1.4 Summary of statistics

In table 4.1 we see a summary of different trained models. We can see that the 5 state AIOHMM has a high log-likelihood together with a low Jensen-Shannon distance as compared to the h-AIOHMMs. It manages to capture a better behaviour compared to the homogeneous models even though fewer states and a smaller part of the data were used. In figure 4.8 we see the likelihood-iteration plots for the best three models from each model type.

In table 4.3 we see the a and b parameter for the 5 state AIOHMM for each state. To recall, the mean for each state was modelled as

$$\mu_{it} = \mu_i(1 + a_i X_t + b_i Y_{t-1}).$$

The ratio of the means of the absolute angle and the longitudinal error is approximately 20 and the medians 13. By multiplying the a parameter with these numbers we can get a sense of how much the models weighs each variable. In most states we still see that the model weighs the error more than the input which is not too surprising since the error correlates about 98% with the previous error. We observe that in state 4 the model weighs the input slightly more than the previous error.

To further investigate what impact the a and b parameters have on the model, we compared the 5 state AIOHMM with itself when setting all a_i to zero in one model and all b_i to zero in another. The results are summarised in table 4.2. We see that by setting a_i to zero in all states i, we get a slightly higher JSd and no change in log-likelihood. This implies that the input does not affect the model that much when conditioning it on the emissions. However, we see that the previous error has a large impact on the model. The JSd drastically increases to 0.67 and the log-likelihood decreases to $-23 \cdot 10^4$ by setting all $b_i = 0$. For this model, we can conclude that the inputs are important for the transitions and that they do not have a large impact on the emissions.

Table 4.1: A summary of different models.

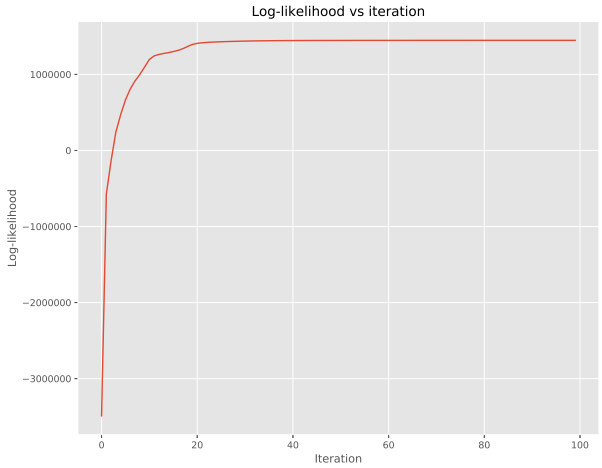
Model	States	JSd	Loglik (10^4)	Input
HMM	10	0.17	1.1	N/A
HMM	15	0.27	2.4	N/A
HMM	20	0.37	3.1	N/A
HMM	5	0.9	-170	N/A
h-AIOHMM	9	0.17	14	Absolute angle, Lifetime
h-AIOHMM	7	0.18	12	Lifetime
h-AIOHMM	15	0.18	13	Absolute angle
h-AIOHMM	10	0.20	13	Absolute angle
h-AIOHMM	7	0.21	13	Absolute angle, Lifetime
h-AIOHMM	6	0.24	13	Lifetime
h-AIOHMM	5	0.26	12	Absolute angle
h-AIOHMM	5	0.32	12	Absolute angle, Lifetime
h-AIOHMM	3	0.38	12	Absolute angle, Lifetime
AIOHMM	5	0.13	9.9	Absolute angle
AIOHMM	4	0.19	1.7	Absolute angle, Lifetime
AIOHMM	3	0.21	5.7	Life time
AIOHMM	5	0.22	12.0	Absolute angle, Lifetime
AIOHMM	10	0.24	11.0	Absolute angle
AIOHMM	5	0.24	9.4	Life time
AIOHMM	3	0.28	12.0	Absolute angle, Lifetime

Table 4.2: Parameter values for the 5 state AIOHMM.

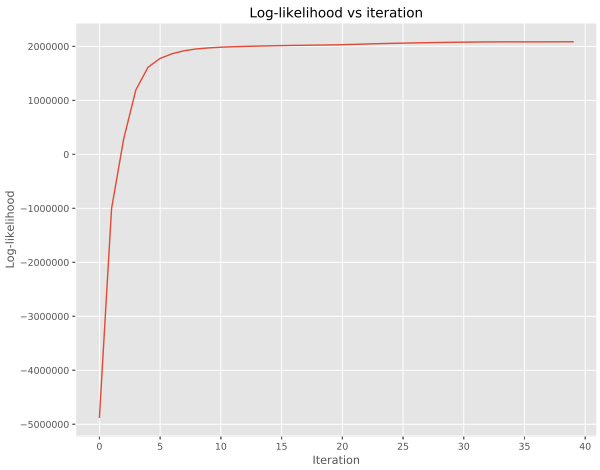
State	1	2	3	4	5
a	-0.0146	-0.0342	-0.0142	0.4176	-0.0646
b	3.7407	49.8227	19.3975	4.1241	4.8731
μ	0.2700	0.0195	0.0474	0.1172	0.1328

Table 4.3: Comparison of the 5 state AIOHMM with itself, setting parameters to zero.

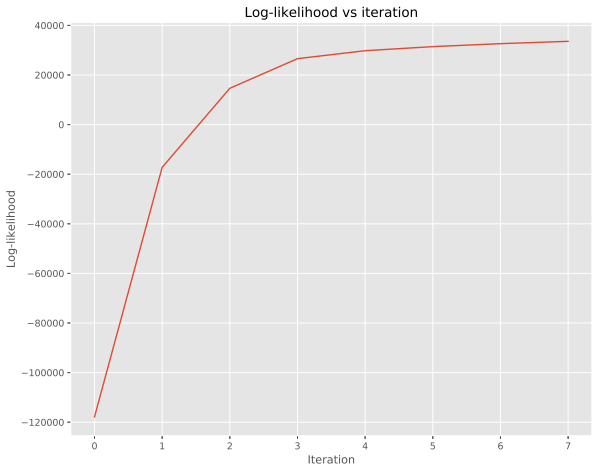
Model	JSd	Loglik (10^4)
Original	0.13	9.9
$a_i = 0$	0.14	9.9
$b_i = 0$	0.67	-23



(a) HMM with 10 states.



(b) h-AIOHMM with 9 states and 2 inputs.



(c) AIOHMM with 5 states and 1 input (absAng).

Figure 4.8: Log-likelihood vs iteration plots for the best model from each model type.

5
Conclusions and future work

5.1 Conclusions

To conclude, we can safely state that simple HMMs do not work for this type of modelling due to too strong assumptions. Not many long-term dependencies are captured because of this. Mainly the conditional independence assumptions between emissions severely limit the HMMs.

The h-AIOHMMs perform much better than simple HMMs and do not take much longer time to train with respect to the size of the data. It manages to capture the autoregressiveness of the data due to conditioning the emission probabilities on previous errors as seen in figure 4.4. Further, by adding an input vector the models manage to learn true structures from the data better.

Lastly, the AIOHMM with inhomogeneous transition matrices manage to perform even better. We note that using an input and conditioning the transition probabilities with it, the models learn the true nate of the data in a better fashion. At the same time, we noticed that conditioning the emissions with an input had a very small impact and that the previous emissions were more important. The down-side of the AIOHMMs is that training takes much longer time, due to the gradient descent when training the w parameter for the inhomogeneous transition probabilities.

5.2 Future work

For future work the implemented models in this thesis can be used in order to model other time series, for example the lateral position error of tracked vehicles. Further it would be interesting to optimise the gradient descent in the AIOHMM training in order to yield faster training. By doing this we could train several different AIOHMMs and perform further analysis, for example how the number of states effect the performance of the models.

Moreover, it would be of great interest to improve the model evaluation, so one does not need to rely on visual inspection of the generated data for evaluation. It would be a great addition to the the model if we could develop a metric that tells us how good it is. A great limitation of the current model evaluation is that it is invariant under permutation, resulting in that bad models can yield good evaluation values. An interesting approach would be to implement a robust classifier (e.g. a

neural network or a support vector machine) in order to distinguish true data from generated one. If implemented well, the robust classifier should have problem separating true data from generated one for a good AIOHMM.

Finally, comparing AIOHMMs with other generative models (e.g. recurrent neural networks, deep belief networks or generative adversarial networks) would be of great interest in order to analyse the strengths and weaknesses of the AIOHMMs with respect to other models.

Bibliography

[1] Andre Luckow et al. "Deep learning in the automotive industry: Applications and tools". In: *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE. 2016, pp. 3759–3768.

[2] Maya R. Gupta and Yihua Chen. "Theory and Use of the EM Algorithm". In: *Foundations and Trends® in Signal Processing* 4.3 (2011), pp. 223–296. ISSN: 1932-8346. DOI: 10.1561/2000000034. URL: http://dx.doi.org/10.1561/2000000034.

[3] Lawrence R. Rabiner. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition". In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286. URL: http://www.cs.ubc.ca/~murphyk/Bayes/rabiner.pdf.

[4] Leonard E. Baum and George R. Sell. "Growth transformations for functions on manifolds." In: *Pacific J. Math.* 27.2 (1968), pp. 211–227. URL: http://projecteuclid.org/euclid.pjm/1102983899.

[5] Steve J. Young et al. *The HTK Book Version 3.4*. Cambridge University Press, 2006.

[6] Yoshua Bengio and Paolo Frasconi. "Input-Output HMM's for Sequence Processing". In: *Annalen der Physik* 7.5 (1996), pp. 1231–1249. URL: http://www.dsi.unifi.it/~paolo/ps/tnn-96-IOHMMs.pdf.

[7] Ashesh Jain et al. "Know Before You Do: Anticipating Maneuvers via Learning Temporal Driving Models". In: *CoRR* abs/1504.02789 (2015). URL: http://arxiv.org/abs/1504.02789.

[8] Peter Sadowski. "Notes on backpropagation". In: (2016). URL: https://www.ics.uci.edu/~pjsadows/notes.pdf.

[9] Sung-Hyuk Cha. "Comprehensive survey on distance/similarity measures between probability density functions". In: *City* 1.2 (2007), p. 1.

[10] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[11] Jop Briët and Peter Harremoës. "Properties of classical and quantum Jensen-Shannon divergence". In: *Physical review A* 79.5 (2009), p. 052311.