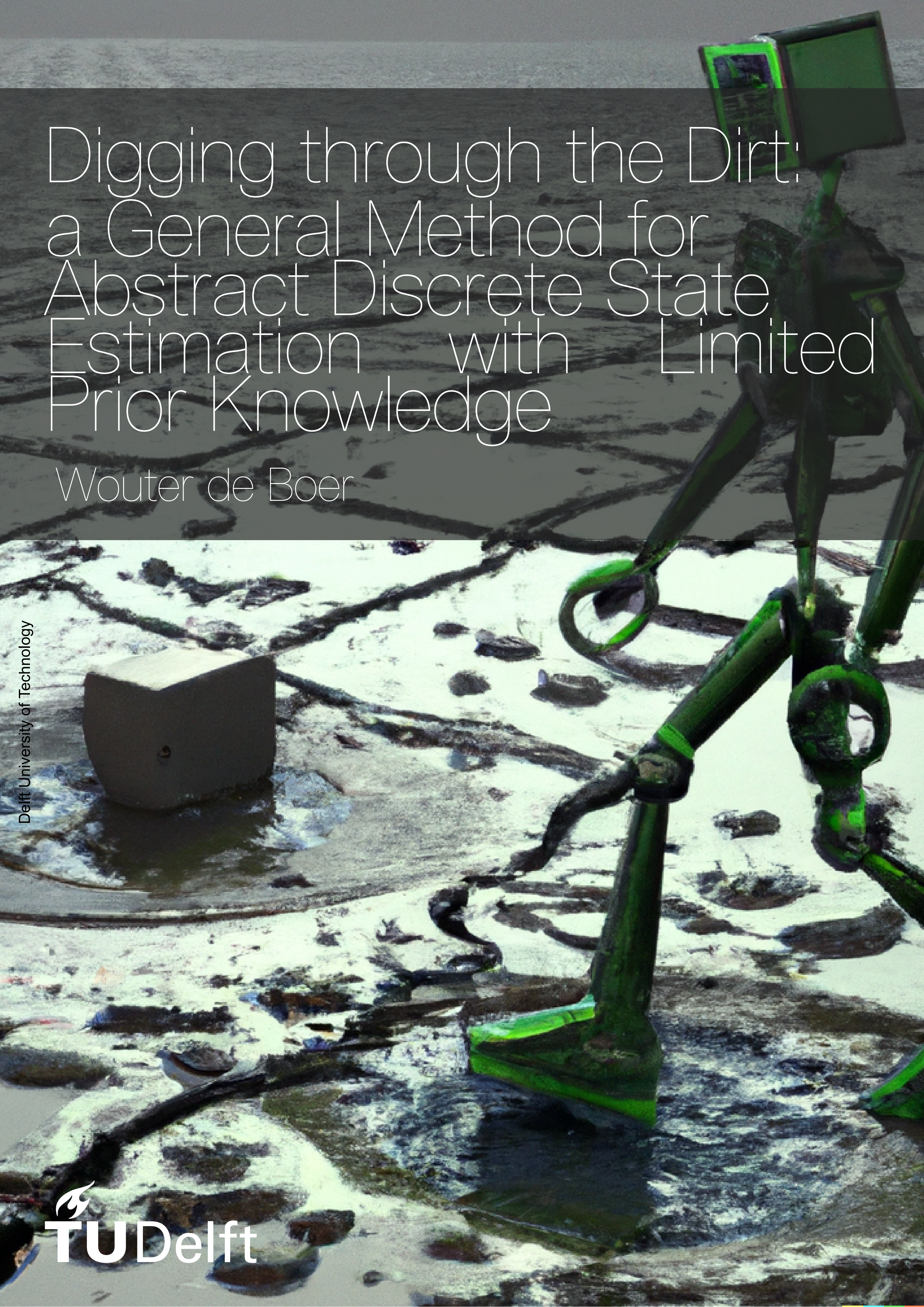


Digging through the Dirt: a General Method for Abstract Discrete State Estimation with Limited Prior Knowledge

Wouter de Boer

Delft University of Technology

Digging through the Dirt: a General Method for Abstract Discrete State Estimation with Limited Prior Knowledge

by

Wouter de Boer

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday April 26, 2023 at 14:00.

Student number:	4384059	
Project duration:	October 1, 2022 – April 26, 2023	
Thesis committee:	Prof. dr. ir. M. Wisse,	TU Delft, supervisor
	Dr. R. Kraaij,	TU Delft
	Ir. C. Pezzato,	TU Delft

Cover: Generated with the assistance of DALL-E 2

TU Delft

Preface

Before you lies my thesis ”Digging through the Dirt: a General Method for Abstract Discrete State Estimation with limited prior Knowledge”. With this thesis, I hope to graduate from the TU Delft, receiving a Master’s degree in Robotics.

What fascinates me the most about the field of robotics is operation under uncertainty: how can we program robots to operate in situations we ourselves do not expect? The subject of this thesis worked perfectly to feed that interest. I had to ask questions to myself that helped me gain insight into the fundamentals of information; what information do I possess? How can I use given information with what I already know? In turn, this thesis taught me valuable lessons about working independently. In the end, a thesis is something to finish by yourself. Of course, I had a lot of help on the way.

I would like to thank my supervisor, Prof. dr. ir. Martijn Wisse, for his excellent guidance and support. I thoroughly enjoyed our discussions going into depth about the (seemingly) simplest problems. I am grateful for your enthusiasm and interest, and that you had the patience to help form the thesis subject to my interests. I also would like to thank Dr. Richard Kraaij for his elaborate insights into the mathematical part of my thesis.

I am also grateful to Max and Thijs for their company in the 3ME penthouse and for providing advice and an open ear. I would like to thank Hoang as well for providing support on the scientific process. Furthermore, I would like to thank Jesse and Alex for the long coffee breaks. And of course, I want to thank Luka for her endless support and care.

Finally, I want to thank my family, friends, huize Sowieso, and my rugby team for providing a welcome distraction at times necessary. I would also like to thank the reader for his interest: I hope you enjoy your reading.

Wouter de Boer
Delft, April 2023

Contents

Abstract

Autonomous robots are often successfully deployed in controlled environments. Operation in uncontrolled situations remains challenging; it is hypothesized that the detection of *abstract discrete states* (ADS) can improve operation in these circumstances. ADS are high-level system states that are not directly detectable and influence system dynamics. An example of a typical ADS problem that is used in this thesis is that of a wheeled robot driving through puddles of mud that, when entered, alters the velocity of the robot. When the robot is in such a puddle, it is in an ADS 'mud', and when it is not, it is in an ADS 'free'. ADS can be indirectly inferred through the analysis of lower-level data such as the velocity of the robot. The goal of this thesis is to design a general *abstract discrete state estimator* (ADSE) operating with limited prior knowledge. An ADSE is a hierarchical system for detecting changes in ADS. The ADSE should be *general*; applicable to multiple ADSE problems. The ADSE should further operate under *limited prior knowledge*; only assuming that the amount of ADS and the ADS that describes the regular operation are known. The basis for the ADSE designed in this thesis is a *Gaussian hidden Markov model* (GHMM), a hidden Markov model enhanced with Gaussian emissions. Randomly generated experiments are done on a simple but general ADSE problem. Two unsupervised learning methods derived from Expectation Maximization are evaluated, namely *Baum-Welch* (BW) and *forward extraction* (FWE). FWE is introduced in this thesis and is a simpler implementation of Viterbi extraction, leveraging assumptions of ADSE to theoretically gain computational efficiency. We found that both BW and FWE exhibit superior performance compared to a likelihood-based baseline estimator when the maximum score of the learning curve is considered. When the final score is considered, in some cases, FWE displays a deteriorating learning curve, resulting in worse final scores compared to the baseline. Furthermore, it was found that there exists a relation between the overlap coefficient of the Gaussian emissions of each state and the maximum reached score, suggesting the existence of a theoretical limit: the lower the overlap coefficient (therefore the less similar the ADS), the higher the maximum reached score. It was further shown that BW exhibits better convergence than FWE to the true model parameters. Besides this, FWE obtained comparable or in some cases even superior scores compared to BW. In general, from the results, the diversity of the experiments conducted, and the assumptions made we can conclude that the GHMM can be a *general* method for an ADSE under *limited prior knowledge*. To quantify the suitability of the GHMM for ADSE, further research should include the evaluation of different ADSE methods on the same problem. There exists a tradeoff between the lower computational cost FWE and the more stable but more computationally intensive BW learning. Therefore, future research can include a combination of these methods. Other extensions include extending the GHMM to a Gaussian mixture hidden Markov model to allow for the modeling of more complex distributions, or the application to multiple states or a changing environment.

List of Acronyms

Acronym	Definition
ADS	abstract discrete state
ADSE	abstract discrete state estimator/estimation
BW	Baum-Welch
EM	expectation maximization
FWE	forward extraction
GMM	Gaussian mixture model
GHMM	Gaussian hidden Markov model
HMM	hidden Markov model
KLD	Kullback-Leibler divergence
LE	likelihood estimator
MCC	Matthews’ correlation coefficient
OVL	overlap coefficient
VBE	Viterbi extraction

1 Introduction

Autonomous robots are a vastly researched topic and have the potential to forward industry and enhance lives. Most successful applications of autonomous robots include controlled environments, such as warehouses and factories. However, the real world is uncontrolled and complex which produces challenges for the deployment of autonomous robots. The circumstances the robot is in can affect how to control it; if a robot encounters for instance component failure, obstacles, or rough terrain the dynamics can differ, meaning that regular control input will not produce the expected results.

Modeling and in turn detection of these events can prove useful for the operation of robots in uncontrolled environments; when such a change in what in this work will be referred to as *abstract discrete states* (ADS) occurs, a robot could adapt its dynamic model accordingly. The problem is that ADS are often not directly detectable: for example, the presence of slippery terrain can be indirectly detected by an increase in motor torque needed to maintain a certain speed or acceleration. In addition, the sensors used are often not fully accurate, creating uncertainty about the ADS the system currently is in.

In order to accurately estimate these ADS and in turn improve robustness in uncertain environments, the goal of this thesis is to create what will be coined as an *abstract discrete state estimator* (ADSE). This system can be seen as hierarchical: the ADSE estimates a higher-level state by analyzing data from a lower-level state, creating a level of abstraction by classifying this data into discrete states. Control is not within the scope of the ADSE; the system will not handle nor consider in what manner an input should differ for each ADS, acting more as an observer.

Readers with a background in control might find similarities with hybrid or switching-state systems. Hybrid systems describe continuous systems that operate in 'modes' with different dynamics, between which can be switched based on conditions such as control input [7]. A difference is that hybrid systems generally assume that the conditions for switching states and the dynamics of these states are known. An additional difference is that the hybrid systems approach focuses on the dynamics and control of the system, including properties such as reachability or stability. In a way, ADSE is typically applied on hybrid systems where these switching conditions are unknown. Another similarity can be found in outlier detection. The main workflow of outlier detection (or anomaly detection) is to specify normal behavior, and then classify behavior differing from this standard as an anomaly. Typically these outliers are removed from the data, but how to exactly handle these is dependent on the situation [34]. It is important to note that handling outliers is not the main focus of the field of outlier detection, which is on identifying these outliers. The difference between ADSE and outlier detection is that outliers are typically sparse in outlier detection, while in ADSE ADS are considered part of the system and not an anomaly. A subfield of outlier detection, called novelty detection, is the most similar to the concept of ADSE to our knowledge. The focus, like outlier detection, lies on detecting events that are not present in training data [24] [17]. Typically for novelty detection, these events are subsumed into the model, in contrast to outlier detection. What is different from ADSE is that the focus lies in developing measures for how different the novelty (outlier) is from normal data and what thresholds to conduct for classification [20]. Compared to novelty detection the idea of ADSE is more general; it is not necessarily needed that ADS are absent in training data. During runtime it should adapt when these ADS *are* present, and when a new ADS appears, this should not pose problems as well. Furthermore, ADSE fixates on online state estimation, and not on developing measures for dissimilarity.

As a method for ADSE, the *hidden Markov model* (HMM) is possibly a suitable method. The HMM is a stochastic model typically used in analyzing time series or sequences, such as biomolecular sequences in computational biology [3], finance[19], and speech recognition [27] and synthesis [32]. An HMM assumes that the current state of the system is *hidden*, or in other words that it cannot be directly observed. Through observations, a hint about the current state is obtained. Combined with an internal model of the expected evolution of states, the HMM is able to estimate the current state while taking uncertainty into account. In its basic form, the HMM handles a discrete system. Since for ADSE a continuous space must be mapped into an abstract discrete space, it is necessary to adjust the HMM to allow for the handling of continuous spaces. A possible extension for this problem is called the *Gaussian hidden Markov model* (GHMM) and is the method that will be used in this thesis.

The goal of this thesis is to create a *general* ADSE with *limited prior knowledge*: the system should be applicable to multiple types of ADSE problems without knowledge of the ADS to be classified. This will be done using the GHMM framework, therefore this thesis will in turn answer the question: can the GHMM be used for a general ADSE with limited prior knowledge? In order to do so, the performance of two learning methods for GHMMs, Baum-Welch and the novel forward extraction is compared to a baseline estimator on a typical ADSE problem.

In chapter 2, the HMM framework and related concepts to apply GHMMs to ADSE are explained. Then, in chapter 3, a typical ADSE problem is defined using GHMMs. Furthermore, in this chapter forward extraction is introduced and the details of the experiments are defined. Afterward, the results are discussed in chapter 4. Finally, the implications of these results for ADSE and future work are discussed in chapter 5.

2 Background Knowledge

This chapter serves as an in-depth reference to the concepts used in this thesis. In section 2.1, the necessary basic concepts are explained, which includes the Markov process. Then, in section 2.2 the key concept of this thesis, the hidden Markov model, is described by extending the Markov process. Then, expectation maximization, a general unsupervised learning algorithm is explained in section 2.3.1 with the help of an example on Gaussian mixture models. Finally, in section 2.3, learning methods for a hidden Markov model are explained. These expectation maximization-based methods are Baum-Welch and a lower-cost alternative Viterbi extraction.

2.1. Prerequisite Knowledge

This section serves as a recap for some basic concepts crucial for understanding the hidden Markov model

2.1.1. Bayes' Rule

The following subsection is taken from the literature research [4]

Bayes' rule is an important concept in probability devised by Thomas Bayes (1701-1761), which enables the use of evidence to calculate the likelihood of a phenomenon. Bayes' rule is used for numerous applications, including statistics, machine learning, AI, and medical diagnosis. Formally, the rule describes the chance of a *hypothesis* H (the phenomenon) being true, *given* the chance of observing *evidence* E. The rule is written as in Equation 2.1:

$$P(H|E) = \frac{P(H)P(E|H)}{P(E)} \tag{2.1}$$

The components of Equation 2.1 are formally defined as:

- $P(H|E)$: *posterior*, the chance of H being true given that E is observed.
- $P(H)$: *prior*, the total chance of H being true.
- $P(E|H)$: *likelihood*, the probability of observing E given that H is true.
- $P(E)$: *marginal*, the total probability of observing E.

2.1.2. Markov Process

Parts of this section are retrieved from the literature research [5]

A *Markov process*, also known as a Markov chain, describes a stochastic process and is named after the Russian mathematician Andrey Markov (1856-1922).
In its discrete form, the Markov process consists of a sequence of length n containing random variables $(S_0, S_1, ...S_n)$ where n is the number of timesteps in the process. Furthermore, there exists a finite state space, which consists of the possible values the random variables can assume. In a Markov process,

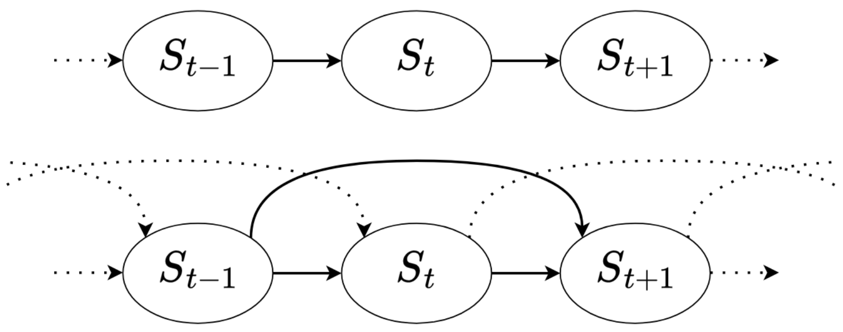


Figure 2.1: A schematic diagram of a Markov process. A 1st order system is shown on top, a 2nd order system at the bottom.

each state only depends on a finite fixed amount of previous states, which is more commonly known as the *Markov property* [25]. Formally, the Markov property can be written as in Equation 2.2.

$$P(S_t|S_0,...,S_{t-2},S_{t-1}) = P(S_t|S_{t-1}) \tag{2.2}$$

To be able to describe a Markov process, the relation between states needs to be defined; the probability of ending up in a specific state given the previous state. This relation is called the *transition probability*. The probability of S_t being in state j given that S_{t-1} is in state i is the following conditional probability: $P(S_t = j|S_{t-1} = i)$ (or shorter: $P(j|i)$). The value of each transition probability needs to be defined beforehand, and all these probabilities can be rearranged in a matrix called the *transition model* or *transition probability matrix*. The transition model has a size of (N, N) where N is the size of the set of states S. This is showcased in Equation 2.3 for a state space $S = (i, j)$.

$$\mathbf{T}(S_t, S_{t-1}) = \begin{pmatrix} P(S_t = i|S_{t-1} = i) & P(S_t = i|S_{t-1} = j) \\ P(S_t = j|S_{t-1} = i) & P(S_t = j|S_{t-1} = j) \end{pmatrix} \tag{2.3}$$

Summarizing, a Markov process can be described with the following model:

- $S \ni (s_a, s_b, ...)$, the set of possible system states or the state space
- $\mathbf{T}(S_t, S_{t-1})$, the transition model (the transition probability matrix)

A Markov process of 1st and 2nd order is described schematically in Figure 2.1.

As a simple example, consider a wheeled robot driving across a path. This path contains puddles of mud: let there be two states, $S_t = mud$ for when the robot is driving through such a puddle at time t and $S_t = free$ when it is not. Therefore, the state space is given as $S = (mud, free)$. Alternatively, this can be considered a one-state system that can take on the values of true and false: $S = (mud, \neg mud)$. From inspection of the path, the following transition model is derived:

$$\mathbf{T}(S_t, S_{t-1}) = \begin{bmatrix} P(S_t = mud|S_{t-1} = mud) & P(S_t = mud|S_{t-1} = free) \\ P(S_t = free|S_{t-1} = mud) & P(S_t = free|S_{t-1} = free) \end{bmatrix} = \begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix} \tag{2.4}$$

This is also showcased in Figure 2.2.

Now, the probability of sequences can be calculated. The sequence $(mud, free, mud, free, mud)$ can be calculated by taking the product of the transition probabilities:

$$P(mud|free) * P(free|mud) * P(mud|free) * P(free|mud) = 0.2 * 0.3 * 0.2 * 0.3 = 0.0036 \tag{2.5}$$

Similarly, the chance of four consecutive timesteps in a mud puddle given that initially the robot is in a puddle of mud can be calculated by simply raising the relevant transition probability to the power of four:

$$P(mud|mud) * P(mud|mud) * P(mud|mud) * P(mud|mud) = 0.7^4 = 0.24 \tag{2.6}$$

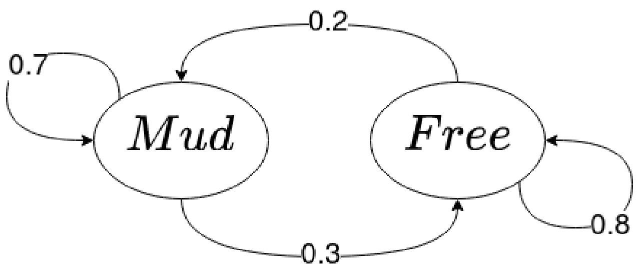


Figure 2.2: A diagram of the mud example. Arrows denote probabilities of state transitions.

2.2. Hidden Markov Model

Some parts from this section are taken from the literature review [6].

2.2.1. Definition

The *hidden Markov model* (HMM) is an extension of the Markov process where the system state is not observable, but can only be observed through another set of stochastic processes that produce the sequence of observed symbols [26]. In other words, the state is not directly observable but 'hidden', but can be inquired about by observations which themselves are generated stochastically.

To integrate the hidden state in the Markov process, some additional components must be added to the model. First, a set of possible observations (sometimes called *evidence* [28]) needs to be defined which provides information about the hidden state. To relate these to the hidden states, it is necessary to define the probability of obtaining a specific observation for a specific hidden state. This probability is often called the *observation* or *(evidence) emission* probability. Similarly to the transition model, these can be arranged in a matrix, which will be referred to as the *observation model* (sometimes referred to as the *sensor model* [28]). The observation model is of size N, M, where N is the size of the set of states S and M is the size of the set of observations Z. The observation model can be constructed similarly to the transition model, which is showcased in Equation 2.7 for an observation space $Z = (a, b, c)$ and a state space $S = (i, j)$

$$\mathbf{O}(S_t, Z_t) = \begin{bmatrix} P(Z_t = a|S_t = i) & P(Z_t = a|S_t = j) \\ P(Z_t = b|S_t = i) & P(Z_t = b|S_t = j) \\ P(Z_t = c|S_t = i) & P(Z_t = c|S_t = j) \end{bmatrix} \tag{2.7}$$

To illustrate this, the mud example from section 2.1.2 is extended with uncertain observations. This extension is in the form of an inaccurate 'mud sensor': a color sensor that can determine if the color of the ground the robot is on is 'dark' or 'light'. These discrete *observations* can be described with two states, 'dark' or 'light', resulting in an observation space $Z = (dark, light)$. Again, this can alternatively be considered as a single state variable that can take on the values of true and false: $Z = (dark, \neg dark)$. These readings 'hint' about the state of the path, where it is more likely that a mud puddle is dark and that a free path is light. This is reflected in the following observation model:

$$\mathbf{O}(S_t, Z_t) = \begin{bmatrix} P(Z_t = dark|S_t = mud) & P(Z_t = dark|S_t = free) \\ P(Z_t = light|S_t = mud) & P(Z_t = light|S_t = free) \end{bmatrix} = \begin{bmatrix} 0.9 & 0.2 \\ 0.1 & 0.8 \end{bmatrix} \tag{2.8}$$

Lastly, due to the unobservability of the HMM, it is necessary to define an initial state distribution. This allows for tracking the likelihood of the system being in each possible state. The estimated state distribution at any point in time is referred to as the *belief state*, therefore the initial state distribution will be referred to as the *initial belief*.

Summarizing, the model for a discrete-time HMM will consist of:

- $S \ni (s_a, s_b, ...)$, the set of possible system states or the state space
- $\mathbf{T}(S_t, S_{t-1})$, the transition model
- $Z \ni (z_a, z_b, ...)$, the set of possible observations

- **O**(S_t, Z_t), the observation model (the observation probability matrix)
- b_0, the initial belief

A 1st and 2nd-order HMM are shown schematically in Figure 2.3.

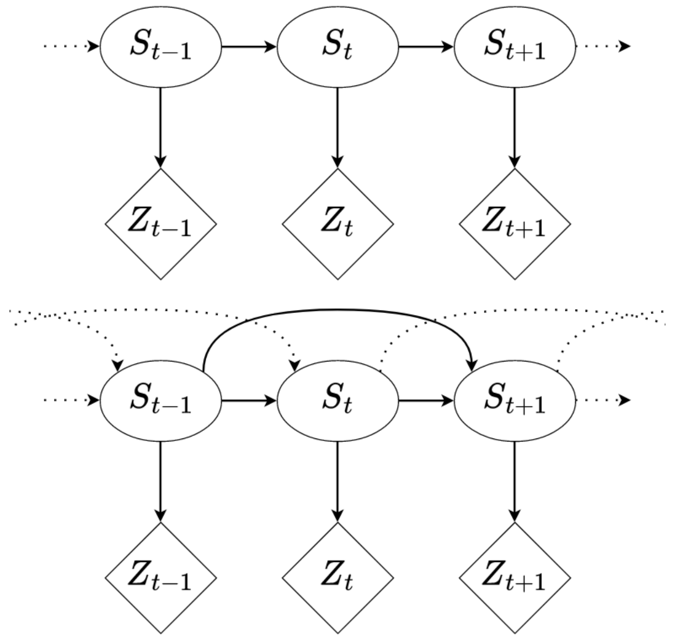


Figure 2.3: A schematic diagram of a hidden Markov model. The above shows a 1st-order model, below a 2nd-order.

Problems
In literature, typically three problems are mentioned in regard to the HMM: 1) the *likelihood* problem, 2) the *decoding* problem, and 3) the *learning* problem. In [12], these are described as follows:

- 1) Likelihood. Given a model λ, what is the probability of occurrence of an observation sequence $Z = Z_0, Z_1, ..., Z_T$: $P(Z|\lambda)$?
- 2) Decoding. What is a state sequence $S = S_0, S_1, ..., S_T$ so that the joint probability of the observation sequence Z and S, $P(Z, S|\lambda)$ is maximized?
- 3) Learning. How can the parameters of an HMM model be arranged so that $P(Z|\lambda)$ is maximized?

The first problem of likelihood can be solved by the forward algorithm, which is described in the next section. Solving the decoding problem means finding an optimal state sequence from a sequence of observations and an HMM model. This is done by the Viterbi algorithm, which is briefly explained in section 2.3.3. In turn, the learning problem tries to find the HMM model parameters (transition model and observation model) such that the likelihood of an observation sequence is maximized. This is described in section 2.3.2.

2.2.2. Forward Algorithm

To maintain a state estimate as the process continues, it is useful to condense the history in a single vector, the belief state. The belief state is updated at every timestep with new information (observations) with a process called the *forward algorithm* (also referred to as *filtering* or *state estimation*) [29]. This process is shown in Equation 2.9. Here, $\mathcal{F}_t(S)$ denotes the belief (or 'forward probability') of state S at time t. In other words, the probability of the system being in a state given all observations ($Z_{0:t}$) upon that point. Furthermore, both the transition probability $P(S_t|S_{t-1})$ and observation probability $P(Z_t|S_t)$ are used, along with a normalizing factor η so that the probabilities sum to one.

$$\mathcal{F}_t(S) = P(S_t|Z_{0:t}) = \eta \sum_{s_{t-1}} P(S_t|S_{t-1})P(Z_t|S_t)\mathcal{F}_{t-1}(S_{t-1}) \tag{2.9}$$

In essence, at each timestep, the current belief is 'propagated' forward one timestep with the transition model. Then, the belief is updated with the new information with the observation model. This recursion is displayed in Equation 2.9; the previous forward probability is used for the next timestep.

Now, the forward algorithm will be applied to the mud example. The same path is used as in section 2.1.2, therefore the transition model will be the same as in Equation 2.4. In other words, the underlying dynamics of the world remain the same. For simplicity, the state space will be abbreviated as follows: $S = (m, f)$. The observation model is the same as in the previous section, Equation 2.8. Lastly, since the robot has just got updated with knowledge of HMMs, the initial belief is 'flat': $b_0 = \mathcal{F}_0(\boldsymbol{S}) = [P(S_0 = mud), P(S_0 = free)] = (0.5, 0.5)$.

Now suppose that the sensor reads 'light' at the first timestep. Then, the probability of being in a mud puddle is equal to:

$$\begin{aligned} \mathcal{F}_1(m) &= \eta(P(m|m) * \mathcal{F}_0(m) + P(m|f) * \mathcal{F}_0(f)) * P(light|m) \\ \mathcal{F}_1(m) &= \eta(0.7 * 0.5 + 0.2 * 0.5) * 0.1 = 0.045\eta \end{aligned} \tag{2.10}$$

And the probability of the path being free:

$$\begin{aligned} \mathcal{F}_1(f) &= \eta(P(f|m) * \mathcal{F}_0(m) + P(f|f) * \mathcal{F}_0(f)) * P(light|f) \\ \mathcal{F}_1(f) &= \eta(0.3 * 0.5 + 0.8 * 0.5) * 0.8 = 0.44\eta \end{aligned} \tag{2.11}$$

Then, calculating the normalizing constant η and multiplying gives the following forward probabilities:

$$\begin{aligned} \eta &= \frac{1}{0.045 + 0.44} \\ \mathcal{F}_1(\boldsymbol{S}) &= [\mathcal{F}_1(m), \mathcal{F}_1(f)] = [0.093, 0.907] \end{aligned} \tag{2.12}$$

As Equation 2.12 shows, it is very likely that the robot is in state free at this timestep.

This belief can further be used for prediction: this is the same process as filtering but without an observation, only propagating the belief forward with the transition model. Since only the transition model is used, this is equal to calculating the probability of a sequence in a Markov process (Equation 2.5).

Readers with a control background might find similarities with the *Kalman filter* (An extensive description can be found in [35]). In essence, a Kalman filter is an HMM where the hidden variables exist in a continuous state space in addition to that all observed and hidden variables are described by Gaussian distribution [14]. Welling [36] notes that HMMs can deal with nonlinear (or stochastic) evolution of states and observations, while a Kalman filter is restricted to linear evolution and observation. In addition, he explains that an HMM is limited to being in one state at a time.

2.2.3. Forward-Backward Algorithm

The forward algorithm is useful to maintain an estimate of the current system state with a small amount of memory. But, when the process has advanced, earlier estimates can be revisited with the information gained by new observations. This results in less variance in the belief state chain. This process is called *smoothing* or the *forward-backward algorithm*. Note that for the forward-backward algorithm, the observations need to be stored in memory.

What is of interest here is the probability of the system being in a state at time t given all observations until the end of the chain T. More specifically: $P(S_t|Z_{0:T})$. In order to calculate this, we need the *backward* probability: the chance of observing the observations from the next timestep $t+1$ until the end of the chain T given the system being in a state at the current time t. This probability, $P(Z_{t+1:T}|S_t)$, is given by the *backward algorithm* and is shown in Equation 2.13 [30]. Note that the 'first' backwards probability at time T is initialized as 1: the observation $Z_{T+1:T}$ is empty, therefore the probability of observing it is 1.

$$\beta_t(S) = P(Z_{t+1:T}|S_t) = \sum_{S_{t+1}} P(S_{t+1}|S_t)P(Z_{t+1}|S_{t+1})\beta_{t+1}(S_{t+1}) \tag{2.13}$$

This equation is similar to the forward algorithm, but then 'going back' in time. Also, similar to the forward probability, this is a recursive equation: the backward probability of the next timestep is used.

The forward-backward algorithm is then just the forward and backward probability multiplied and normalized. This is shown Equation 2.14, where η is again a normalizing factor [30].

$$\gamma_t(S) = P(S_t|Z_{0:T}) = \frac{\mathcal{F}_t(S)\beta_t(S)}{\sum_S \mathcal{F}_t(S)\beta_t(S)} = \eta\,\mathcal{F}_t(S)\beta_t(S) \tag{2.14}$$

Now, consider the same example as used for the forward algorithm in section 2.2.2. The next two timesteps are considered, which give two observations: $Z_2 = dark$ and $Z_3 = light$. Then, the forward probabilities are calculated as follows:

$$\begin{aligned}
\mathcal{F}_2(m) &= \eta(0.7 * 0.093 + 0.2 * 0.907) * 0.9 = 0.222\eta \\
\mathcal{F}_2(f) &= \eta(0.3 * 0.093 + 0.8 * 0.907) * 0.2 = 0.151\eta \\
\mathcal{F}_2(\boldsymbol{S}) &= [0.595, 0.405] \\
\\
\mathcal{F}_3(m) &= \eta(0.7 * 0.595 + 0.2 * 0.405) * 0.1 = 0.050\eta \\
\mathcal{F}_3(f) &= \eta(0.3 * 0.595 + 0.8 * 0.405) * 0.8 = 0.402\eta \\
\mathcal{F}_3(\boldsymbol{S}) &= [0.091, 0.909]
\end{aligned} \tag{2.15}$$

We see that the forward algorithm gives a larger chance of the system being in a mud puddle at timestep 2. To calculate the backward probabilities, first initialize the backward probability at time T, $\beta_3(\boldsymbol{S}) = [1, 1]$. Then the backward probabilities for the other timesteps are derived as follows:

$$\begin{aligned}
\beta_2(m) &= P(m|m) * P(Z_3|m) * \beta_3(m) + P(f|m) * P(Z_3|f) * \beta_3(f) \\
\beta_2(m) &= 0.7 * 0.1 * 1 + 0.3 * 0.8 * 1 = 0.31 \\
\\
\beta_2(f) &= P(m|f) * P(Z_3|m) * \beta_3(m) + P(f|f) * P(Z_3|f) * \beta_3(f) \\
\beta_2(f) &= 0.2 * 0.1 * 1 + 0.8 * 0.8 * 1 = 0.66 \\
\\
\beta_1(m) &= 0.7 * 0.9 * 0.33 + 0.3 * 0.2 * 0.66 = 0.245 \\
\beta_1(f) &= 0.2 * 0.9 * 0.33 + 0.8 * 0.2 * 0.66 = 0.165 \\
\\
\beta_2(\boldsymbol{S}) &= [0.31, 0.66] \\
\beta_1(\boldsymbol{S}) &= [0.245, 0.165]
\end{aligned} \tag{2.16}$$

Which, when completing the forward-backward algorithm, gives the following *smoothed* probabilities:

$$\begin{aligned}
\gamma_3(\boldsymbol{S}) &= \mathcal{F}_3(\boldsymbol{S}) = [0.091, 0.909] \\
\gamma_2(\boldsymbol{S}) &= \eta[0.595 * 0.31, 0.495 * 0.66] = [0.361, 0.639] \\
\gamma_1(\boldsymbol{S}) &= \eta[0.093 * 0.245, 0.907 * 0.165] = [0.132, 0.868]
\end{aligned} \tag{2.17}$$

The results in Equation 2.17 show that the smoothed probability of the system being in state mud at timestep 2 is now lower than that of the system being in state free. The effect of the forward-backward algorithm is displayed when these results are compared to the forward probabilities: the smoothed probabilities are less susceptible to variance in the observations.

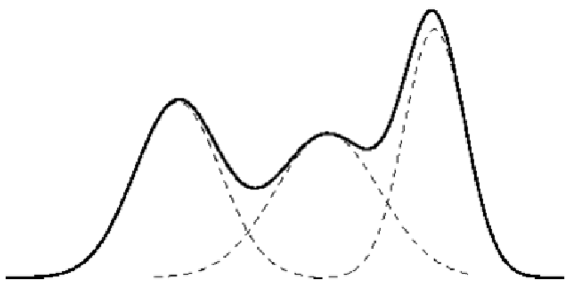


Figure 2.4: A schematic depiction of a Gaussian mixture model [23]. It can be seen that a Gaussian mixture model (full line) consists of a weighted sum of a number of univariate Gaussians (dashed lines).

Gaussian Hidden Markov model
Observations in an HMM do not have to be constricted to discrete values. These can be extended in a case where every state emits a value drawn from a Gaussian distribution: a *Gaussian (emission) hidden Markov model* (GHMM). For a GHMM, the set of observations is $Z = (f_a, f_b, ...)$, where $f_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$. For determining the observation model of a GHMM, the reader is referred to section 3.3.2.

2.3. Learning

For learning the parameters of an HMM, a commonly used method is Baum-Welch (BW) learning. This will be explained in this section by first laying out the concept of expectation maximization (EM). Then, after describing BW, a lower-cost alternative called Viterbi extraction (VBE) is explained.

2.3.1. Expectation Maximization

Expectation maximization (EM) is an unsupervised learning method to find the parameters of a statistical model that is dependent on hidden variables: "It produces maximum-likelihood estimates of parameters when there is a many-to-one mapping from an underlying distribution to the distribution governing the observation" [22]. In other words, it is assumed that there exist multiple latent variables on which the data is dependent. When learning, the parameters of the model are set so that the likelihood of this data is maximized.

The algorithm consists of two steps: the *E-step* (expectation), which is followed by the *M-step* (maximization) [11]. These steps are then iteratively applied. In the E-step, the probabilities are estimated for which latent variable generated what data point in what magnitude given the current model. Then in the M-step, the model parameters are updated by maximizing the likelihood, assuming the assignments estimated during the E-step are the true values.

The EM algorithm is guaranteed to converge to a local maximum: the likelihood function increases at each iteration. The local maximum is not guaranteed to be the global maximum, however; if the function has multiple maxima, the local maximum to which the algorithm converges is dependent on the initial parameter estimates [22].

Gaussian Mixture Model
A common application and an intuitive example of EM is learning of *Gaussian mixture models* (GMM). A GMM is a distribution consisting of a weighted combination of K Gaussian distributions and is used for modeling probability distributions. The principle is shown schematically in Figure 2.4, taken from [23].

Learning a GMM works as follows: first, the mean and variance of K Gaussian distributions are initialized, in addition to the mixing coefficient (weights). Then in the E-step, given a collection of data points, for each data point the probability of it belonging to each single of the K Gaussian distributions is computed. Finally, in the M-step, the parameters of each Gaussian distribution are updated to maximize the expectations of the data points calculated in the E-step. In simpler terms, data points are assigned to a Gaussian distribution, after which the distribution is fitted to best match the data. For a Gaussian distribution f_k (component) with likelihood function $\mathcal{L}_k(x_i)$, mean μ_k, variance σ_k^2 and weight w_k, the E-step is given by Equation 2.18 [31] Here, $\kappa_{i,k} = P(f_k|x_i, w_k, \mu_k, \sigma_k^2)$, the probability that GMM component f_k generated data point x_i.

$$\kappa_{i,k} = \frac{w_k \mathcal{L}_k(x_i)}{\sum_{j=1}^{K} w_j \mathcal{L}_j(x_i)} \tag{2.18}$$

Then, the parameters of a component are computed at the M-step, in the following order [31]:

$$\begin{aligned} \tilde{w_k} &= \sum_{i=1}^{N} \frac{\kappa_{i,k}}{N} \\ \tilde{\mu_k} &= \frac{\sum_{i=1}^{N} \kappa_{i,k} x_i}{\sum_{i=1}^{N} \kappa_{i,k}} \\ \tilde{\sigma^2}_k &= \frac{\sum_{i=1}^{N} \kappa_{i,k}(x_i - \tilde{\mu_k})^2}{\sum_{i=1}^{N} \kappa_{i,k}} \end{aligned} \tag{2.19}$$

Here, N is the total amount of data points.

When more closely examining Equation 2.18 and Equation 2.19, a clear intuition can be found on the EM algorithm. One can see that the weight is simply the probability of assigning a data point to a component averaged over all data points. Similar for the mean and variance of the component, which are a weighted average of κ and the data points; data points that are more probable to belong to the component are weighted heavier.

2.3.2. Baum-Welch algorithm

The *Baum-Welch* (BW) algorithm is a learning algorithm for HMMs and can be considered a special case of the EM algorithm [10][33]. Therefore, the algorithm also maximizes a likelihood function and enjoys the same convergence properties. BW utilizes the forward-backward algorithm (section 2.2.3) to gain an estimate of the system parameters. In this case, the transition model and the observation model of the HMM. One more probability is necessary to gain these parameters: let us define the ξ-function, the probability of a transition from i to j occurring at time t given all observations and the model parameters θ: $P(S_t = i, S_{t-1} = j|Z_{0:T}, \theta)$. When omitting θ for simplicity, this is given by [2]:

$$\begin{aligned} \xi_t(S_t, S_{t-1}) = P(S_t, S_{t-1}|Z_{0:T}) &= \frac{\mathcal{F}_{t-1}(S_{t-1})\beta_t(S_t)P(S_t|S_{t-1})P(Z_t|S_t)}{\sum_{s_{t+1}} \mathcal{F}_{t-1}(S_{t-1})\beta_t(S_t)P(S_t|S_{t-1})P(Z_t|S_t)} \\ \xi_t(S_t, S_{t-1}) &= \eta\, \mathcal{F}_{t-1}(S_{t-1})\beta_t(S_t)P(S_t|S_{t-1})P(Z_t|S_t) \end{aligned} \tag{2.20}$$

Here, again η is a normalizing factor. The combination of a forward-backward pass through the system data and the calculation of the ξ-function is considered the E-step of the algorithm: according to the current parameters (transition model and observation model), probabilities are estimated of which latent variable (hidden state) generated what data point (observation). Then, the parameters can be re-estimated in the M-step (for a GHMM):

$$\begin{aligned} P(S_t = j|S_{t-1} = i) &= \frac{P(S_t = j, S_{t-1} = i|Z_{0:T})}{P(S_{t-1} = i|Z_{0:T})} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_{t-1}(i)} \\ \mu_S &= \frac{\sum_{t=1}^{T} \gamma_t(S) Z_t}{\sum_{t=1}^{T} \gamma_t(S)} \\ \sigma_S &= \sqrt{\frac{\sum_{t=1}^{T} \gamma_t(S)(Z_t - \mu_S)^2}{\sum_{t=1}^{T} \gamma_t(S)}} \end{aligned} \tag{2.21}$$

For a 'regular' HMM, the equations for the transition model are similar, but logically a difference exists for the now discrete observation model. Here, $\hat{z}_t$ denotes the observation at timestep t:

$$\begin{aligned} O(S_t = i, Z_t = a) &= \frac{\sum_{t=1}^{T} \gamma_t(i) 1_{\hat{z}_t=a}}{\sum_{t=1}^{T} \gamma_t(i)} \\ 1_{\hat{z}_t=a} &\begin{cases} 1 & \hat{z}_t = a \\ 0 & otherwise \end{cases} \end{aligned} \tag{2.22}$$

Finally, when only the transition model is to be learned, there can be a computational saving by only computing the ξ-function, which is related to the smoothed probabilities. This is shown by using that all transitions from one state must sum to 1:

$$\begin{aligned} \sum_{j=1}^{M} \xi_t(S_t = j | S_{t-1} = i) = P(S_t = j, S_{t-1} = i | Z_{0:T}) \\ &+ P(S_t = i, S_{t-1} = i | Z_{0:T}) + ... + P(S_t = M, S_{t-1} = i | Z_{0:T}) \\ &= P(S_{t-1} = i | Z_{0:T}) \\ &\quad * [P(S_t = j, S_{t-1} = i | Z_{0:T}) + P(S_t = i, S_{t-1} = i | Z_{0:T}) + ... \\ &\quad + P(S_t = M, S_{t-1} = i | Z_{0:T})] \\ &= P(S_{t-1} = i | Z_{0:T}) * 1 = \gamma_{t-1}(i) \end{aligned} \tag{2.23}$$

Therefore, it is shown that the ξ-function summed over every possible transition from one state i is equal to the smoothed probability of the same state i. This has the following implications for the M-step of BW:

$$P(S_t = j | S_{t-1} = i) = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_{t-1}(i)} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{M} \xi_t(i,j)} \tag{2.24}$$

2.3.3. Viterbi Extraction

While the Baum-Welch algorithm is the standard referred learning algorithm for HMMs, alternatives exist. One of these is the *Baum-Viterbi* algorithm or *Viterbi extraction* (VBE) [13][18]. Here, the learning procedure is the same as for BW, but instead of a 'soft' assignment, a binary assignment is assumed: the most likely state is assigned a 1 and the rest of the states a 0. In VBE, this most likely state is determined by the *Viterbi algorithm* [15]. This algorithm solves the decoding problem (section 2.2.1) and gives the most likely state sequence given a sequence of observations.

The Viterbi algorithm is similar to the forward algorithm, but with 'backtracking' [27]: the Viterbi algorithm considers the full sequence, and finds the best state sequence given that this sequence is valid. For example, when the transition probability between two states s_a and s_b is 0, the forward algorithm could return a sequence physically impossible. This is because the forward algorithm considers a state estimation at one instant. The Viterbi algorithm overcomes this issue by including 'backtracking' which is keeping track of the best path.

An upside compared to BW is that the Viterbi algorithm requires just one pass over the system data compared to the two passes the forward-backward algorithm uses. Therefore, VBE has the potential of being a faster learning method than BW.

For VBE, the estimated transition is determined by the optimal state sequence decoded by the Viterbi algorithm. this concludes the E-step of VBE. Then, for the first part of the M-step, a transition model can be derived by a frequentist approach: by dividing the amount of one type of transition from a state by the total amount of transitions from that same state. This is shown in Equation 2.25, where $n_{a \to b}$ denotes the total amount of transitions estimated from state a to b and where M is the number of states in the model.

$$P(S_t = j|S_{t-1} = i) = \frac{n_{i\to j}}{\sum_{j=1}^{M} n_{i\to j}} \tag{2.25}$$

For estimating the observation model, the process is equal to BW, but with the 'hard' count obtained by the Viterbi algorithm. The M-step is given similarly as BW for a GHMM in Equation 2.26. Here $V_t(S_t)$ is equal to 1 if the optimal state sequence V at timestep t, decoded by the Viterbi algorithm, is equal to the argument of S_t.

$$\begin{aligned} V_t(S_t) &= \begin{cases} 1 & S_t = V_t \\ 0 & else \end{cases} \\ \mu_S &= \frac{\sum_{t=1}^{T} V_t(S)Z_t}{\sum_{t=1}^{T} V_t(S)} \\ \sigma_S &= \sqrt{\frac{\sum_{t=1}^{T} V_t(S)(Z_t - \mu_S)^2}{\sum_{t=1}^{T} f_{t_{max}}(S)}} \end{aligned} \tag{2.26}$$

And similarly to Equation 2.22 for a regular HMM:

$$\begin{aligned} O(S_t = i, Z_t = a) &= \frac{\sum_{t=1}^{T} 1_{\hat{z}_t=a} V_t(i)}{\sum_{t=1}^{T} V_t(i)} \\ 1_{\hat{z}_t=a} & \begin{cases} 1 & \hat{z}_t = a \\ 0 & otherwise \end{cases} \end{aligned} \tag{2.27}$$

Similar to the GMM, upon closer inspection of the M-step there is a clear intuition: only the 'relevant' estimations are weighted in the calculation of the parameters.

3 Methods

In this chapter, the concept of abstract discrete states will be defined further in section 3.1. A typical toy problem for abstract discrete states is defined in section 3.2. In addition, this section includes a description of how experimental data can be generated from this toy problem Then, an ADSE method based on a Gaussian hidden Markov model will be described in section 3.3. Afterward in section 3.4, a practical learning algorithm coined forward extraction, derived from Viterbi extraction, is defined. Finally, experiments for validating this model, evaluating the difference between two learning methods, and the relation between abstract discrete states characteristics and performance are described in section 3.5.

3.1. Abstract Discrete States

Abstract discrete states (ADS) are high-level system states that influence system dynamics; discrete events or system statuses that have an influence on the behavior of a system. An ADS is typically not directly observable but can be inferred through the analysis of simpler system data, such as velocity or temperature. Therefore, ADS are hierarchical in a way; estimations about a higher abstract state are made by the analysis of a lower-level state. A typical ADS problem is depicted in Figure 3.1. Examples of ADS include rough terrain or a foreign object causing excessive axle friction for a wheeled robot, changing the relation between input and velocity. Or, an improper cable connection causing an increase in resistance and therefore resulting in a higher input signal needed to allow the same movement of a robotic manipulator. An example of a high-abstraction ADS is part malfunction of a factory robot causing anomalies in the end product. This thesis only considers one level of abstraction. Still, it is expected that for future work more levels of abstraction can be hierarchically built on each subsequent level similar to the methods described in this work.

The general, robust inference of unknown ADS is the goal of this thesis, that is the creation of a *general abstract discrete state estimator* (ADSE) operating under *limited prior knowledge*; the nature of the ADS should not be important for the functioning of the ADSE and the ADSE assumes only the amount of ADS and that the behavior of the ADS that is 'normal' operation is known. The behavior of the other ADS is to be learned in an unsupervised manner. In order for the problem to be solvable, it is assumed that an ADS exerts influence on at least one measurable system state. To capture the essence of ADS, a general toy problem is defined in section 3.2.

3.2. Problem Description and Data Generation

For the toy problem we will use a 1D, endless world referred to as the 'mudworld'. A wheeled robot with Gaussian movement is driving across this world in one direction. This Gaussian movement is dependent on the current true ADS, which in this world can be either 'mud' or 'free'; the world consists of randomly generated patches of 'mud' which, when entered, alter the mean and variance of the Gaussian velocity of the robot. If the robot enters such a patch, the true ADS changes from 'free' to 'mud'. When the patch is left, the true ADS changes to 'free' again. When the robot is in state 'free' (f), the velocity is drawn from a probability distribution $f_f \sim \mathcal{N}(\mu_{f_f}, \sigma_{f_f})$. When the state 'mud' ($m$) is entered,

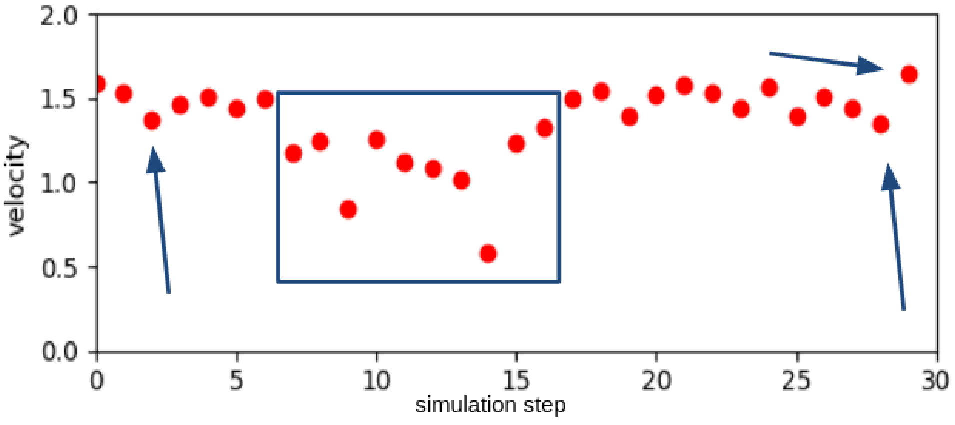
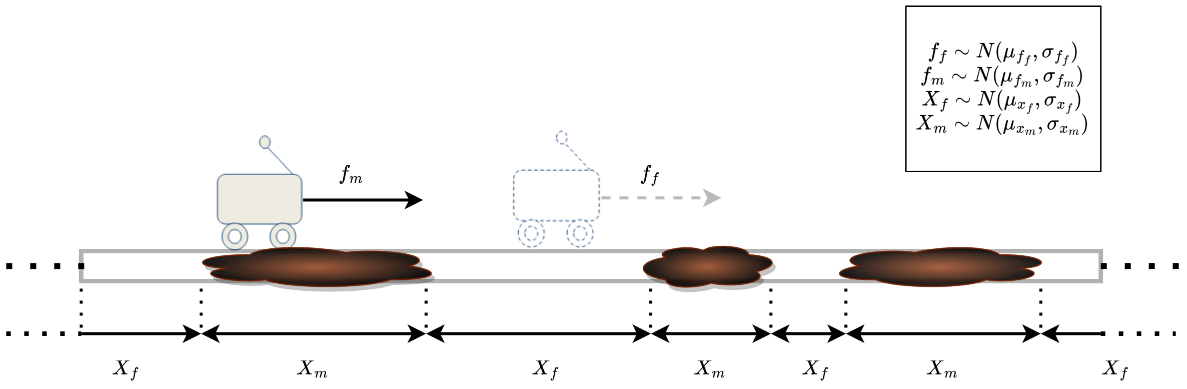


Figure 3.1: Schematic depiction of a typical abstract discrete state problem. The box denotes a possible different abstract discrete state. The challenge is to determine where to draw this box, and additionally, whether values such as those pointed out by the arrows are to be classified as another abstract discrete state.

the velocity is drawn from another probability distribution $f_m \sim \mathcal{N}(\mu_{f_m}, \sigma_{f_m})$; now, the same input results in a different mean velocity and variance. Note that generally in this thesis $dt = 1$: therefore, the distance moved after a timestep is equal to the velocity at the start of that timestep.

In turn, the size of patches of mud and mud-free patches are described by two Gaussian distributions, $X_m \sim \mathcal{N}(\mu_{x_m}, \sigma_{x_m})$ and $X_f \sim \mathcal{N}(\mu_{x_f}, \sigma_{x_f})$ respectively. A mud patch is always followed by a free patch and vice-versa. One exception exists: in the event that a patch is sized below 0, the patch size is set to 0, meaning that the same patch type can follow up. It is randomly chosen if the starting patch is a mud patch or a free patch. A schematic depiction of the world model is shown in Figure 3.2.



$f_f \sim N(\mu_{f_f}, \sigma_{f_f})$
$f_m \sim N(\mu_{f_m}, \sigma_{f_m})$
$X_f \sim N(\mu_{x_f}, \sigma_{x_f})$
$X_m \sim N(\mu_{x_m}, \sigma_{x_m})$
f_m
f_f
X_f
X_m
X_f
X_m
X_f
X_m
X_f

Figure 3.2: Schematic depiction of the world model with 'mud patches' used in the experiments.

As an example, consider such an environment generated by a Gaussian distribution for free patches $X_f \sim \mathcal{N}(2, 1)$ and a Gaussian distribution for mud patches $X_m \sim \mathcal{N}(1, 1)$. This results in an environment consisting of one free patch $X_{f_0} = 1$, followed by a mud patch $X_{m_0} = 0.5$ and then a free patch $X_{f_1} = 2$ again. The Gaussian distributions determining movement are $f_f \sim \mathcal{N}(1.0, 0.05)$ for free patches, and $f_m \sim \mathcal{N}(0.75, 0.2)$ for mud patches.

The agent starts at position $x_0 = 0$, therefore the ADS at $t = 0$ is f. The robot receives an input of 1; since the true ADS is f, the velocity will be drawn from f_f. This gives a velocity of 1.1, therefore the position $x_1 = 1.1$. Now, the true ADS is m: $1.0 \leq x_1 < 1.5$. Therefore, the next input of 1 results in a velocity drawn from f_m, which gives 0.7. Now, the position is $x_2 = 1.8$. The current ADS changes to f again: $1.5 \leq x_2 < 3.5$. This gives a velocity of 0.95, and so on. These velocities are used as observation in the ADSE; an example can be found in section 3.3.3.

This section described, apart from the general idea of the toy problem, how the data on which ADSE will be performed is generated. This data generation part and the ADSE part of the problem described in the next section are in fact separate: performing ADSE has no influence on the data generated. In other words, the ADSE is essentially an observer of the dynamical system. In order to perform ADSE in this problem, a *mapping* is needed from the continuous world to a discrete space; from the velocity of the robot the current ADS, 'mud' or 'free', needs to be derived. This is done by approximating the world dynamics by a GHMM, which is explained in the next section.

3.3. Model Formulation

The first step in creating an HMM is to formally define all parameters, as in section 2.2.1. When considering this problem, intuitively it differs from a classical HMM representation in two ways. First, since the observation is a velocity that is observed after a timestep, it gives information about the state a system was in only after the system has transitioned to the next state. In runtime, the current belief of the system is lagging behind the true state by one timestep. Conceptually, this does not alter the HMM, but it is useful to keep this in mind if the system is to be used for real-time operation. A schematic representation of the resulting HMM is shown in Figure 3.3. Secondly, the observations emitted are Gaussian, transforming the HMM into a GHMM (See section 2.3.1). More detail on how this is translated to the model can be found in section 3.3.2.

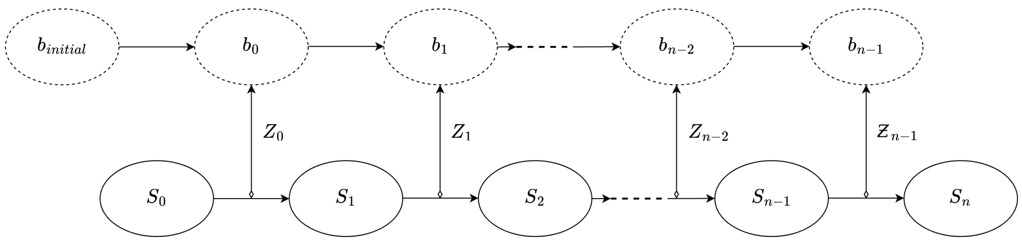


Figure 3.3: Schematic depiction of the hidden Markov model used. The length of the chain is n, b denotes the belief, Z an observation, and S the state. The subscripts of these variables denote the timestep.

3.3.1. Transition Model

The transition model is defined as the probability of entering a state at the following timestep given a state: $P(S_t|S_{t-1})$. The complete transition model $T(S_t, S_{t-1})$ is defined as a square matrix of size (N, N) where N is the size of the set of states S:

$$S = \{Mud(m), Free(f)\} \tag{3.1}$$

Therefore, the transition model can be defined globally as:

$$T(S_t, S_{t-1}) = \begin{bmatrix} P(S_t = m|S_{t-1} = m) & P(S_t = m|S_{t-1} = f) \\ P(S_t = f|S_{t-1} = m) & P(S_t = f|S_{t-1} = f) \end{bmatrix} \tag{3.2}$$

Which will be referred to more compactly as:

$$T(S_t, S_{t-1}) = \begin{bmatrix} P(m|m) & P(m|f) \\ P(f|m) & P(f|f) \end{bmatrix} \tag{3.3}$$

Logically follows that the columns in the transition model sum to 1.

3.3.2. Observation Model

Since the observations are drawn from a Gaussian distribution the observation model is continuous, it is not possible to fit the model in a table such as the transition model. A function has to be defined that captures the relation between observations and states: $P(Z_t|S_t)$. We assume that every state emits observations from a distinct Gaussian distribution.

We are interested in the probability of the system being in a state given a data point (observation) Z_t and the Gaussians distributions belonging to each state, $f_{mud} = f_m \sim \mathcal{N}(\mu_{f_m}, \sigma_{f_m})$ and $f_{free} = f_f \sim \mathcal{N}(\mu_{f_f}, \sigma_{f_f})$. In other words, we are interested in which Gaussian distribution is more likely and by what magnitude. This problem is described by the *law of likelihood*, where ϕ is the ratio of the two likelihood functions:

$$\phi = \frac{P(f_f|Z_t)}{P(f_m|Z_t)} \tag{3.4}$$

Since the probability density function $f(x)$ of a Gaussian distribution is defined on the interval $[-\infty, \infty]$, we can assume that both components from Equation 3.4 are possible for any value of Z_t. In other words, we can assume that all possible observations can be generated by any Gaussian distribution. Then, assuming the models have equal prior probability (that is $P(f_f) = P(f_m)$), follows from Bayes' rule:

$$\phi = \frac{P(Z_t|f_f)P(f_f)}{P(Z_t)} * \frac{P(Z_t)}{P(Z_t|f_m)P(f_m)} = \frac{P(Z_t|f_f)}{P(Z_t|f_m)} \tag{3.5}$$

This is equal to the ratio of the probability density functions, which for a Gaussian distribution is described by:

$$f(z_t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{z_t-\mu}{\sigma})^2} \tag{3.6}$$

This is further displayed in Figure 3.4.

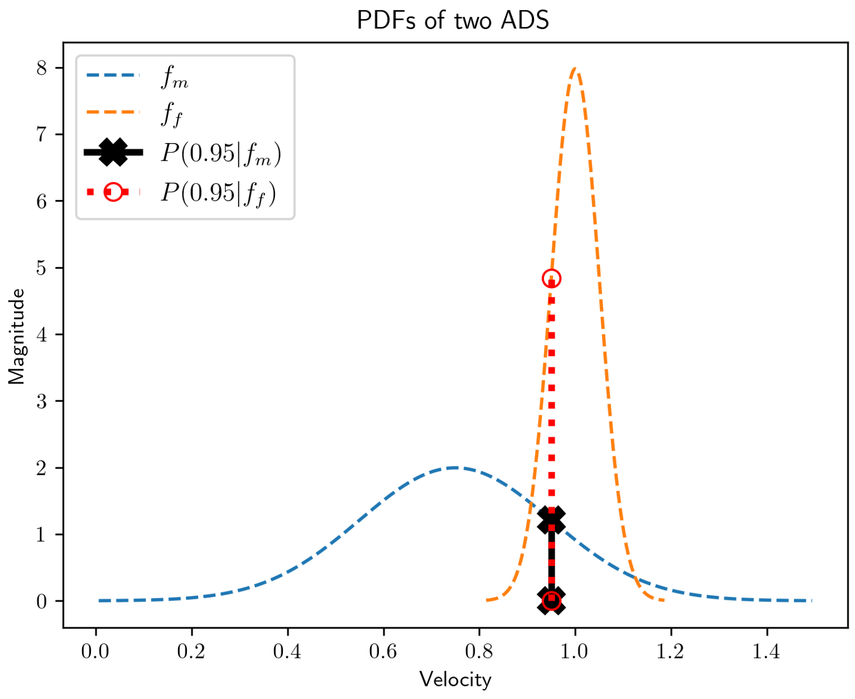


Figure 3.4: Probability density functions for two normally distributed abstract discrete states with $f_f \sim \mathcal{N}(1, 0.05)$ and $f_m \sim \mathcal{N}(0.75, 0.2)$. The vertical lines denote the magnitude of the probability density function at a velocity of 0.95. The normalized ratio of these magnitudes gives the observation model, see Figure 3.5.

The observation probabilities can then be described as the normalized ratio between probability density functions:

$$
\begin{aligned}
P(Z_t|S_t = f) &= \frac{\phi}{\phi+1} \\
P(Z_t|S_t = m) &= 1 - P(Z_t|S_t = f) = 1 - \frac{\phi}{\phi+1}
\end{aligned}
\tag{3.7}
$$

This gives a continuous observation function that can be directly used as in section 2.2. The function is shown graphically in Figure 3.5.

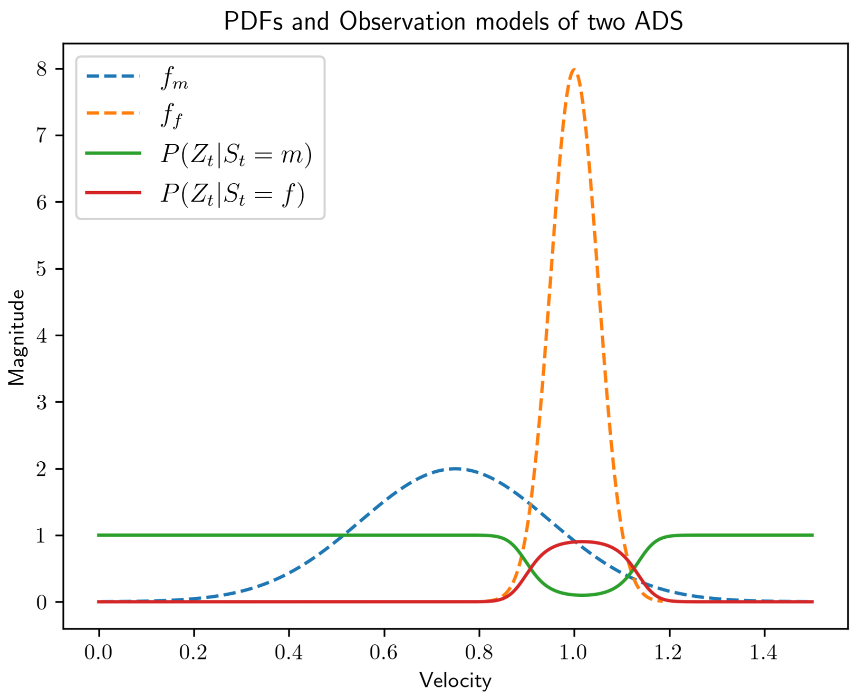


Figure 3.5: Observation model and probability density functions for two normally distributed abstract discrete states with $f_f \sim \mathcal{N}(1, 0.05)$ and $f_m \sim \mathcal{N}(0.75, 0.2)$.

3.3.3. Example

For an example of how ADSE with this model works, consider a similar setting as in section 3.2: one free patch $X_{f_0} = 1$, followed by a mud patch $X_{m_0} = 0.5$ and then a free patch $X_{f_1} = 2$ again.
Three timesteps are considered, which result in a true ADS sequence of $\{f, m, f\}$ and corresponding velocities of $\{1.1, 0.7, 0.9\}$. Assuming that true distributions determining movement f_f and f_m are learned as $f_f \sim \mathcal{N}(1.0, 0.06)$ and $f_m \sim \mathcal{N}(0.7, 0.15)$ and applying Equation 3.7 gives the following observation probabilities for each timestep: $P(1.1|S = f) = 0.96$, $P(0.7|S = f) = 0$, $P(0.95|S = f) = 0.88$.

The following transition model is used in the ADSE:

$$
T(S_t, S_{t-1}) = \begin{bmatrix} 0.6 & 0.3 \\ 0.4 & 0.7 \end{bmatrix}
\tag{3.8}
$$

Starting from an initial belief $b_{init}(\boldsymbol{S}) = [P(S_{init} = m), P(S_{init} = f)] = (0.5, 0.5)$ and applying the forward algorithm gives the following estimates for the ADS at the first timestep:

$$
\begin{aligned}
&\mathcal{F}_0(m) = \eta(P(m|m) * f_0(m) + P(m|f) * f_0(f)) * P(1.1|m)\\
&\mathcal{F}_0(m) = \eta(0.6 * 0.5 + 0.3 * 0.5) * 0.04 = 0.018\eta\\
&\mathcal{F}_0(f) = \eta(0.4 * 0.5 + 0.7 * 0.5) * 0.96 = 0.528\eta\\
&b_0(\boldsymbol{S}) = (0.033, 0.967)
\end{aligned}
\tag{3.9}
$$

Then, using these probabilities as the current belief and using the forward algorithm two more times yields: $b_1(\boldsymbol{S}) = (1, 0)$ and $b_2(\boldsymbol{S}) = (0.17, 0.83)$. When retrieving the maximum of each belief, we can see that in this example the true ADS sequence is correctly estimated by the ADSE.

3.4. Forward Extraction

For the ADSE, state estimation will be done in realtime. Therefore, at every timestep a forward pass is necessary, resulting in a current belief. Furthermore, since the ADSE represents a physical system and ADS changes are by definition possible, we assume that the path generated by the forward pass is always valid. In other words, we assume the HMM to be fully connected or *ergodic*. To leverage this assumption, we propose a VBE (section 2.3.3) inspired learning method called *Forward extraction* (FWE). For this simplified algorithm, only the estimated transition is necessary to track: for every estimated transition, one count is added. The estimated transition is determined by the states with the maximum belief value, instead of the most likely sequence determined by the Viterbi algorithm. The Viterbi algorithm assumes that the HMM is not necessarily ergodic and therefore requires access to the whole history. FWE spreads the computational load by reusing the calculations that are necessary anyway for ADSE at every timestep. This results in a learning method theoretically computationally more efficient and more practical for ADSE compared to VBE. Since VBE is guaranteed to be more efficient than BW, the same applies to FWE.

The E-step for FWE is just maximizing the belief retrieved from a forward pass, returning a belief. This is similar to VBE, where instead of a forward pass the Viterbi algorithm is used. An example of the application of Equation 2.25 is shown in Equation 3.12.

A transition model can be derived similarly to VBE, by a frequentist approach. This is shown in Equation 2.25.

For estimating the observation model, the process is equal to VBE, but with the count obtained by maximizing the forward probabilities instead of the Viterbi algorithm. The M-step is given similarly as VBE for a GHMM in Equation 3.10, here $\mathcal{F}_{t_{max}}(S_t)$ is equal to 1 if the maximum argument of the forward probability at timestep t is equal to the argument of S_t.

$$
\begin{aligned}
\mathcal{F}_{t_{max}}(S_t) &= \begin{cases} 1 & S_t = argmax(\mathcal{F}_t) \\ 0 & else \end{cases}\\
\mu_S &= \frac{\sum_{t=1}^{T} \mathcal{F}_{t_{max}}(S) Z_t}{\sum_{t=1}^{T} \mathcal{F}_{t_{max}}(S)}\\
\sigma_S &= \sqrt{\frac{\sum_{t=1}^{T} \mathcal{F}_{t_{max}}(S)(Z_t - \mu_S)^2}{\sum_{t=1}^{T} \mathcal{F}_{t_{max}}(S)}}
\end{aligned}
\tag{3.10}
$$

And similarly to Equation 2.27 for a regular HMM:

$$\begin{aligned} O(S_t = i, Z_t = a) &= \frac{\sum_{t=1}^{T} 1_{\hat{z}_t = a} \mathcal{F}_{t_{max}}(i)}{\sum_{t=1}^{T} \mathcal{F}_{t_{max}}(i)} \\ 1_{\hat{z}_t = a} &\begin{cases} 1 & \hat{z}_t = a \\ 0 & otherwise \end{cases} \end{aligned} \tag{3.11}$$

Similar to the learning methods described in section 2.3, upon closer inspection of the M-step there is a clear intuition: only the 'relevant' estimations are weighted in the calculation of the parameters.

Supervised Learning
FWE lends perfectly as a simple method for supervised learning, in this context that is learning a model for a (provided) true state sequence. Conceptually, this is equal to learning a model for fully accurate state estimation. Thus, the parameters of the HMM can be derived by Equation 3.10 just by inserting the true state sequence instead of the maximized forward probabilities.

3.5. Experiments

The goal of this thesis is to create a *general* ADSE with *limited prior knowledge*. To assess whether the GHMM suits these requirements, experiments will be done in the mudworld (section 3.2). The agent in the mudworld will be inferring ADS with a learning GHMM-based ADSE. The ADSE will be evaluated on classification performance and how accurately the true parameters are learned. Both BW learning (section 2.3.2) and FWE will be evaluated. To assess the effectiveness of learning and the addition of a GHMM, these methods will be compared to a non-learning likelihood-based estimator with perfect knowledge of the Gaussian distributions determining the agent's movement ($f_f \sim \mathcal{N}(\mu_{f_f}, \sigma_{f_f})$ and $f_m \sim \mathcal{N}(\mu_{f_m}, \sigma_{f_m})$). Apart from perfect knowledge of f_f and f_m, this baseline is similar to the ADSE system with a transition model of $[0.5, 0.5]$, and therefore will not take any sequence into account. This estimator is referred to as the likelihood estimator (LE).

Training is done on a training set of 5000 transitions, and evaluation after each training iteration is done on a test set of 1000 transitions. The world of the training and test set are different due to a differing random seed, but are built in the same manner; the Gaussian distributions of the patch sizes X_m and X_f are the same. Thus, the training data is representative of the test set. Furthermore, it is important to note that due to the random seed the velocities at each timestep are also differing from the training and test set. For each scenario, 30 training iterations will be done, resulting in 30 train scores and 31 test scores when the initial score is included.

To research if the ADSE is able to operate with *limited prior knowledge*, the assumption will be made that only the regular movement is known, meaning that the Gaussian distribution f_f that is fed into the model is equal to the true Gaussian. For f_m an initial Gaussian distribution $\mathcal{N}(0.1, 1.0)$ is used: therefore, the only prior knowledge that is fed into the ADSE is that there exist a total of two ADS, both with Gaussian emissions, of which one is known. Note that the possibility to adjust f_f during the learning process still exists.

In order to determine if the GHMM is a *general* method for ADSE, three scenarios will be evaluated. These scenarios exhibit different distributions of ADS; effectively, this corresponds to different practical problems. For example, an uncommon 'normal' state or on the contrary when 'mud' states are sparse. These different problems can be condensed into three types of scenarios: 1) where mud patches are significantly smaller than free patches ('Free', $\mu_{X_f} > \mu_{X_m}$), 2) where free patches are significantly smaller than mud patches ('Mud', $\mu_{X_f} < \mu_{X_m}$) and 3) where patch sizes are equal ('Equal', $\mu_{X_f} = \mu_{X_m}$). X_f and X_m are chosen rather arbitrarily; it is expected that the ratio between patches contributes to meaningful results more than the specific values. For all scenarios, a low variance, as well as a high variance case, will be assessed to simulate the randomness of an environment, with low variance simulating a more predictive environment. See Table 3.1.

Table 3.1: Parameters chosen for scenario generation, with movement in m generated by a distribution $\mathcal{N}(1.0, 0.2)$. The mud patches are scaled with μ_{f_m}.

	High variance		Low variance	
Scenario	Free patch (X_f)	Mud patch (X_m)	Free patch (X_f)	Mud patch (X_m)
1: Free	$\mu = 20.4, \sigma = 3.0$	$\mu = 3.6\mu_{f_m}, \sigma = 2.0\mu_{f_m}$	$\mu = 20.4, \sigma = 0.3$	$\mu = 3.6\mu_{f_m}, \sigma = 0.2\mu_{f_m}$
2: Mud	$\mu = 3.6, \sigma = 2.0$	$\mu = 20.4\mu_{f_m}, \sigma = 3.0\mu_{f_m}$	$\mu = 3.6, \sigma = 0.2$	$\mu = 20.4\mu_{f_m}, \sigma = 0.3\mu_{f_m}$
3: Equal	$\mu = 8.7, \sigma = 3.0$	$\mu = 8.7\mu_{f_m}, \sigma = 3.0\mu_{f_m}$	$\mu = 8.7, \sigma = 0.3$	$\mu = 8.7\mu_{f_m}, \sigma = 0.3\mu_{f_m}$

For each scenario the mean and variance of f_m are altered to research the effect of similarity between f_f and f_m on the difficulty of learning and state estimation. In addition, every combination is essentially another ADS problem in itself. Therefore, experimenting with these variations can aid in evaluating if the ADSE is *general* as well. The means (μ_{f_m}) used are $\{0.5, 0.75, 0.9, 1.0, 1.1, 1.25, 1.5\}$ and the variances (σ_{f_m}) used are $\{0.1, 0.2, 0.3\}$. This brings the total amount of experiments done on $3*2*2*7*3 = 252$: all combinations of 3 scenarios, 2 variances, 2 learning methods, 7 means, and 3 variances of f_m.

The size of mud patches are scaled according to a factor equal to μ_{f_m}: for example, when in the equal scenario with high variance $\mu_{f_m} = 0.5$, then the patch sizes are $X_f \sim \mathcal{N}(8.6, 3.0)$ and $X_m \sim \mathcal{N}(4.3, 1.5)$. This results in roughly the same amount of timesteps spent in each state on average for each experiment. Therefore, when different means and variances for f_m are used for an experiment, the only changing variable is the mean and variance of f_m.

A description of the programs used to simulate the environments, use the ADSE and run the experiments can be found in Appendix A (A link to the code can be found here). It is recommended to briefly examine this section for a deeper understanding of the differences between the data generation part and the actual ADSE part of the simulation.

Determining the transition model

It is favorable for assessing learning effectiveness to have an indication of what transition model describes the world; then it can be considered if the ADSE learns the true transition model. Analytically, it is hard to derive a transition model of the world; the patch sizes and the velocities are both drawn from Gaussian distributions. A much easier way is a learning approach as in section 3.4. If this is done supervised (by injecting the true ADS sequence), an accurate transition model can be generated. For this HMM, using Equation 2.25 gives the following equations:

$$\begin{aligned} P(m|m) &= \frac{n_{m\to m}}{n_{m\to m} + n_{m\to f}} \\ P(m|f) &= \frac{n_{f\to m}}{n_{f\to f} + n_{f\to m}} \\ P(f|m) &= \frac{n_{m\to f}}{n_{m\to m} + n_{m\to f}} = 1 - P(m|m) \\ P(f|f) &= \frac{n_{f\to f}}{n_{f\to f} + n_{f\to m}} = 1 - P(m|f) \end{aligned} \tag{3.12}$$

Measures

To determine whether the goal of this thesis is reached, that is the creation of a general ADSE with limited prior knowledge, it is important to quantify the performance. Besides judging the classification performance to an extent, such a measure is necessary to evaluate the learning process. Then, it is possible to track improvements in performance across the learning process. A problem with performance is that it is related to the difficulty of a problem. For a meaningful answer on how good a classifier is, the difficulty of such a problem must be quantified; normally, this is done by comparison with other systems on the same problem. This is not within the scope of this thesis. Therefore, in the context of this thesis, a high classification score alone does not necessarily imply a competent classifier. The only conclusions that can then be drawn from the performance in itself are 1) if the system improves upon the baseline LE, 2) if the system performance increases due to learning, and 3) which learning method performs better. Therefore, to answer if the GHMM can be used as a general method for ADSE with limited prior knowledge it is crucial to consider how the learned parameters approach the true parameters as well. This can answer the question if the score earned is in fact the optimal score the ADSE can reach for the problem; then, by assessing both the classification and learning performance, it can be answered to what extent the GHMM can be used for an ADSE.

The classification performance of the ADSE will be evaluated by means of the *Matthew's correlation coefficient* (MCC). The MCC is a weighted measure of classifier performance and is generally considered more useful to assess performance on imbalanced datasets [9]. The score ranges from -1, a reversed classifier, to 0, equal to a 50/50 random guess, to 1, a perfect classifier. The MCC only generates a high score if the sensitivity, precision, specificity, and negative predictive value are high [8]. In practice, this means that in an imbalanced dataset, only a high score can be achieved when both uncommon and common states are classified correctly by a high degree. Since the 'patch' scenarios can show imbalance, this is considered a suitable performance measure for the ADSE. The MCC is given by Equation 3.13 [9], where TP is a true positive, TN a false negative, FP a false positive, and FN a false negative.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}} \tag{3.13}$$

The learned parameters of the transition model will be compared to that derived by a frequentist approach on the true sequence of states of the training set, as described in section 3.5. Additionally, the learned parameters of f_m will be compared with the true values. This will be evaluated by the closed-form expression of the *Kullback-Leibler divergence* (KLD) for two Gaussian distributions f_1 and f_2 as described in Equation 3.14 [1]. Intuitively, the KLD (sometimes referred to as relative entropy) describes the degree of similarity between two probability distributions.

$$KL(f_1||f_0) = \frac{1}{2}(ln\frac{\sigma_0^2}{\sigma_1^2} + \frac{\sigma_1^2}{\sigma_0^2} + \frac{(\mu_1 - \mu_0)^2}{\sigma_0^2} - 1) \tag{3.14}$$

Since the observation model is related to the overlap between the two probability density functions, the hypothesis is that this overlap determines the difficulty of such an ADS problem with two univariate Gaussian distributions. Therefore, the similarity will be evaluated by the *overlap coefficient* (OVL). The OVL is a measure of 'agreement' between two probability distributions or populations and is given by Equation 3.15 [16]. Here, Φ denotes the cumulative distribution function of the standard normal distribution ($\mathcal{N}(0, 1)$) and X denotes the points of intersection, where X_1 is the smaller of the two points. The OVL is equal to 1 when the two distributions are exactly the same, and approaches 0 when the overlap diminishes. The Python standard library 'statistics' contains this function and will be used to calculate the OVL.

$$OVL = \Phi(\frac{X_1 - \mu_1}{\sigma_1}) + \Phi(\frac{X_2 - \mu_2}{\sigma_2}) - \Phi(\frac{X_1 - \mu_2}{\sigma_2}) - \Phi(\frac{X_2 - \mu_1}{\sigma_1}) + 1 \tag{3.15}$$

3.6. Conclusion

ADS are high-level discrete system states that influence system dynamics. Therefore, these states can typically be inferred indirectly through lower-level system data. An ADS problem was introduced: the 1D 'mudworld'. This is a randomly generated world containing patches of mud. A wheeled robot drives through this world with Gaussian velocity. Two ADS exist in this world: 'free', where the Gaussian distributed velocity is not impaired, and 'mud', where the mean and variance of this velocity differs. An ADSE method based on GHMMs was described. Furthermore, a practical algorithm for ADSE, FWE, was introduced. For the experiments, three scenarios are considered; three different patterns of 'patches'. Patches are areas in the world that contain a single ADS. The size of these patches is described by a Gaussian distribution, found in Table 3.1. Experiments for validating this model, evaluating the difference between the two learning methods, and the relation between ADS characteristics and performance were described. These experiments consist of trials in each scenario and for a set of means and variances for the Gaussian velocity in the mud ADS, creating a total amount of 252 different experiments.

4 Results

In this chapter, first the results of the experiments described in section 3.5 will be evaluated in section 4.1. This is done by two measures: by considering the Matthews' correlation coefficient of the learned model and by considering the convergence to the true parameters of the transition model and of the Gaussian distributions determining movement, f_f and f_m. Additionally, the overlap coefficient will be utilized to assess the correlation between the Matthews' correlation coefficient and classification performance. Then, the two learning methods, Baum-Welch (section 2.3.2) and forward extraction (section 3.4) will be compared in section 4.2.

4.1. Results

To evaluate the performance of the ADSE, the system was evaluated on the mudworld described in section 3.2. Summarizing chapter 3, experiments were done by letting the robot 'travel' through this 1D world. This world contained 'patches' consisting of one of the two possible ADS: 'mud' and 'free'. These patches were configured according to six different scenarios, found in Table 3.1. Every timestep, the agent received its current velocity. This normally distributed velocity was dependent on the current ADS of the system: f_m for 'mud', and f_f for 'free'. Based on this observation, an estimate was made on the current ADS. An example of this process can be found in section 3.3.3. After each pass through the complete simulation environment, a performance score was calculated. After this, the agent adjusted its parameters according to the classifications it made and the set of observations, the *training set*. This concluded one *training iteration*. Then, the updated agent was evaluated on a representative *test set* which was generated through the same principle as the training set. The learning was done by either of the two learning methods: BW and FWE. Further details about the experiments can be found in section 3.5.

Tables containing key points of the resulting learning curves can be found in Table 4.1 and Table 4.2 for test and train scores respectively. In this table, the maximum, initial, and final scores of the learning curve of both BW and FWE can be found. The initial score is the score obtained before the first training iteration is completed, and the final score is the score obtained after the last training iteration. In addition, the baseline score of the LE is shown in this table. All these values are for the case when $\mu_{f_m} = 0.75$ and $\sigma_{f_m} = 0.2$. The full learning curves for this case, in addition to the convergence to the true parameters of the transition model and to the true Gaussian distributions determining movement, can be found in Appendix B.

In Table 4.1, the maximum score for both learning methods exceeds the baseline score in all scenarios and for both levels of scenario variance. For BW learning, this is also the case for the final scores. However, in the mud scenario, FWE obtained a final score lower than the baseline. The full learning curves for this scenario are shown in Figure 4.1, in addition to the convergence to the transition model and the convergence of the parameters of the learned f_m to the parameters of the true f_m. For comparison, the learning curves of the same case for BW are displayed in Figure 4.2. From the learning curves, we see that after reaching a maximum, improvement degraded until the classifier ended up worse than the

Table 4.1: Summary of the maximum, first and final iteration **test** Matthews' correlation coefficient. Baseline score of the likelihood estimator with perfect knowledge of the Gaussian distributions determining movement ($\mathcal{N} \sim (0.75, 0.2)$) is included. The highest score of the two methods is shown in bold.

		Maximum		Final			
Scenario	Variance	Baum-Welch	FWE	Baum-Welch	FWE	Initial	Baseline
1: Free	High	0.877	**0.881**	0.877	**0.881**	0.841	0.778
	Low	**0.858**	0.852	**0.857**	0.852	0.848	0.761
2: Mud	High	**0.804**	0.719	**0.782**	0.556	0.614	0.655
	Low	**0.754**	0.692	**0.744**	0.510	0.588	0.633
3: Equal	High	**0.828**	**0.828**	0.824	**0.828**	0.787	0.797
	Low	**0.835**	0.817	**0.832**	0.816	0.780	0.790

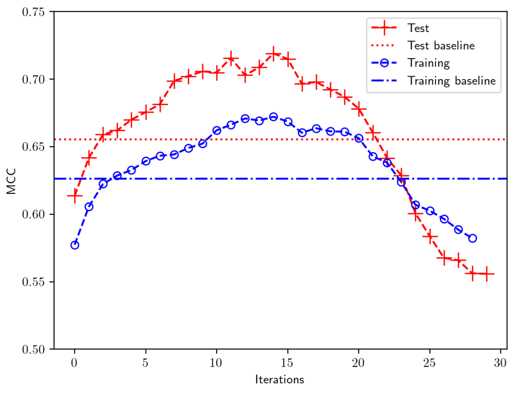
Table 4.2: Summary of the maximum, first and final iteration **train** Matthews' correlation coefficient. Baseline score of the likelihood estimator with perfect knowledge of the Gaussian distributions determining movement ($\mathcal{N} \sim (0.75, 0.2)$) is included. The highest score of the two methods is shown in bold.

		Maximum		Final			
Scenario	Variance	Baum-Welch	FWE	Baum-Welch	FWE	Initial	Baseline
1: Free	High	**0.863**	0.860	**0.861**	0.853	0.829	0.778
	Low	**0.859**	0.853	**0.857**	0.853	0.834	0.777
2: Mud	High	**0.750**	0.672	**0.750**	0.582	0.577	0.626
	Low	**0.728**	0.660	**0.721**	0.571	0.574	0.621
3: Equal	High	**0.848**	0.822	**0.848**	0.822	0.771	0.794
	Low	**0.836**	0.814	**0.835**	0.813	0.770	0.794

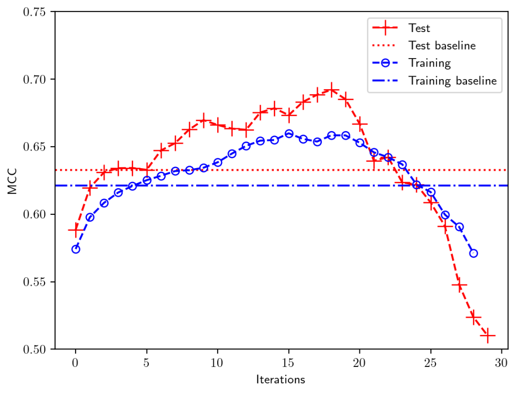
baseline. A similar result for the training scores, even while exhibiting less steep degradation, makes it hard to appoint this to overfitting: in that case, the training score should keep improving while the test score degrades. When comparing the learning curves with the convergence to the transition model and the convergence of the parameters of the learned f_m to the parameters of the true f_m in Figure 4.1, remarkable is that the transition model estimate for $P(m|f)$ eventually moved away from the actual value. In turn for the Gaussian distributions, the estimates of the parameters of f_m moved towards those belonging to f_f. This can be appointed to more data being classified as mud, since both $P(m|m)$ and $P(m|f)$ converge towards 1, explaining why the parameters of f_f are deviating as well; only even higher values will be appointed to the free state, resulting in a higher mean. With this, the variance grows lower too, which is a result of the remaining population of emissions classified as free being less divided among the velocity space. Therefore, the convergence of these parameters explains the drop in performance: the model starts to overclassify the mud state, entering a vicious circle until only the most distinctive observations are classified as a free state.

Considering the baseline scores in Table 4.1 and Table 4.2, in the mud and equal scenarios the initial scores were generally lower than the baseline score. This was expected since the initial f_m was very different from the actual f_m: one would predict a lower score for the same transition model. Surprisingly, in the free scenarios, this did not hold; here, the initial score was generally higher than the baseline score. A likely explanation is that free states are relatively easy to distinguish due to the lower variance. This is proven by the fact that the highest scores were all achieved within the free scenario, where the total amount of free states was the highest. From this follows that if mud is an uncommon state it can be preferable to use a significantly different Gaussian as initial f_m. Then, only significantly different observations will be classified as mud, while similar values are more likely to be appointed as a free state. In a way, this can be seen as a certain bias or low-pass filter against the uncommon mud state. As the parameters of the HMM were learned, the score improved. This suggests that the transition model gradually compensated for this bias while the observation model converged to the true values.

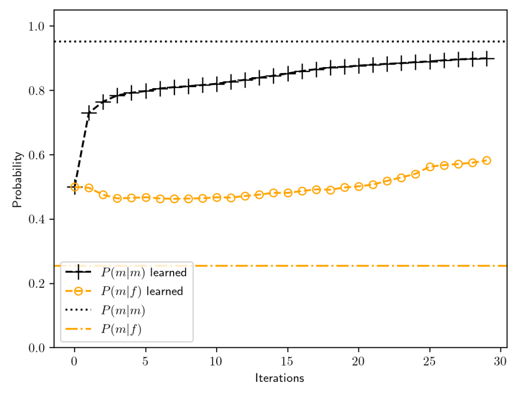
When comparing variances in Table 4.1 and Table 4.2, in general, a higher variance gave a larger relative difference between the test and train score. An explanation is that low variance results in more similar test and train data; in some of these cases, such as the free scenario with low variances, the



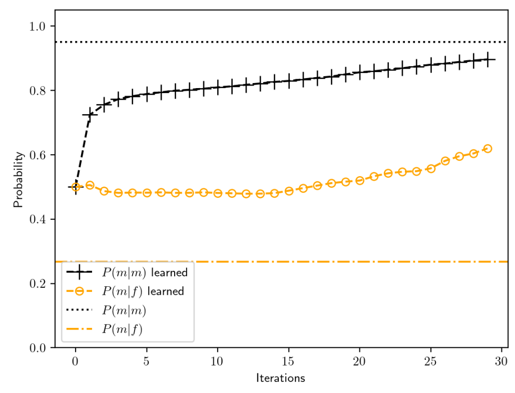
(a) Scores, high variance case



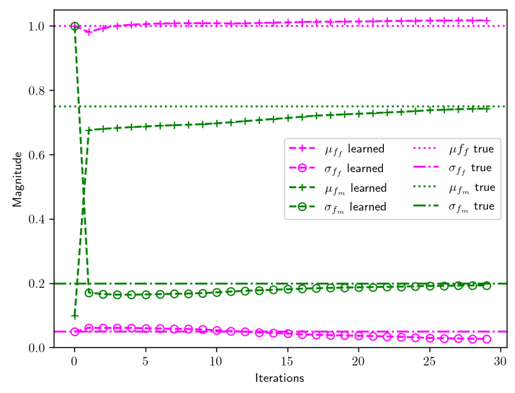
(b) Scores, low variance case



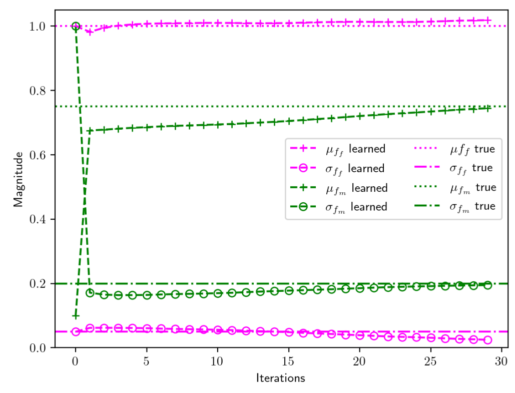
(c) Learned transition model parameters, high variance case



(d) Learned transition model parameters, low variance case



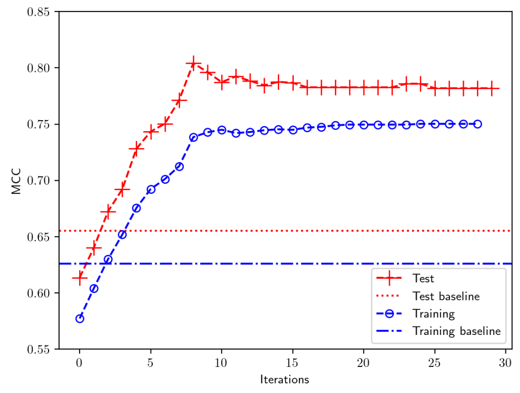
(e) Learned parameters of Gaussian distributions, high variance case



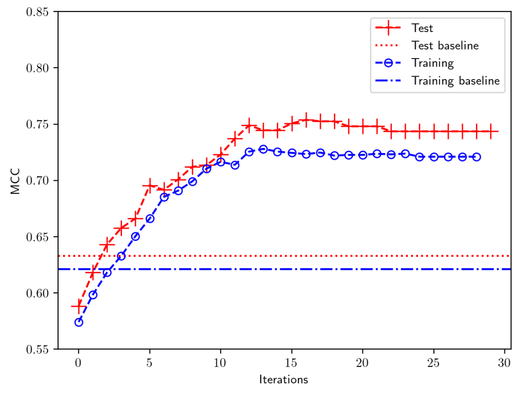
(f) Learned parameters of Gaussian distributions, low variance case

Figure 4.1: Matthews' correlation coefficients, learned probabilities for the transition model and magnitude of means and variances of learned Gaussian distributions when using **Forward Extraction** for the mud scenario. Here, $\mu_{f_m} = 0.75$ and $\sigma_{f_m} = 0.2$. For the Matthews' correlation coefficients, the score of the baseline, the likelihood estimator with perfect knowledge of the Gaussian distributions determining movement, is included.

test and train scores are almost identical. The detailed learning curves in Appendix B display this more clearly.



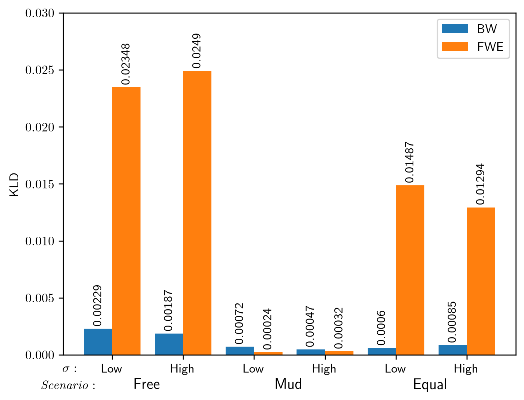
(a) Scores, high variance case



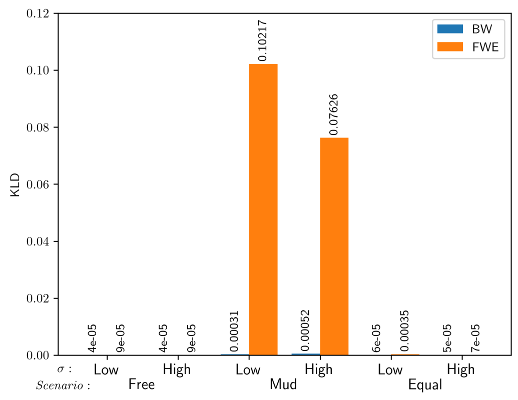
(b) Scores, low variance case

Figure 4.2: Matthews' correlation coefficients scores when using **Baum-Welch** for the mud scenario. Here, $\mu_{f_m} = 0.75$ and $\sigma_{f_m} = 0.2$. The score of the baseline, the likelihood estimator with perfect knowledge of the Gaussian distributions determining movement, is included.

Figure 4.3a shows the KLD of the learned f_m and f_f relative to the true f_m and f_f. For each case, the estimate at the final iteration was used. The estimates at every iteration can be found in Figure B.5 and Figure B.6 for BW and FWE respectively. A correlation can be found between the total amount of mud states and the KLD. When the total amount of mud states was higher, such as in the mud scenario, the KLD tends to be lower. This is an indication that the Gaussian distribution is approximated more closely. A simple explanation is that a higher amount of mud states gives more data on f_m, thus increasing the accuracy of the expectation maximization algorithm.



(a) Kullback-Leibler divergence of the learned f_m relative to the actual f_m.



(b) Kullback-Leibler divergence of the learned f_f relative to the actual f_f.

Figure 4.3: Kullback-Leibler divergence of learned Gaussian distribution relative to the actual Gaussian distribution. The value of the final iteration is used to calculate the Kullback-Leibler divergence, the last value in the graphs of Figure B.5 and Figure B.6.

In Table 4.3 the learned values of the transition model can be found (for additional insights, Figure B.3 and Figure B.4 contain curves showing the transition model estimate at every timestep). The table contains the learned transition model at the final training iteration, in addition to the true transition model. Furthermore, these values were for the case when $\mu_{f_m} = 0.75$ and $\sigma_{f_m} = 0.2$. From the table, it becomes clear that BW was able to converge close to the true value in all cases, while FWE was further off from the true estimate, only obtaining the best estimate in one case. Furthermore, there seems to exist a correlation between the amount of mud or free states and the estimate for $P(m|m)$ and $P(m|f)$ respectively; the more states, the more accurate the respective estimate. This is likely because more data exists on a state, improving the accuracy of classifying that state: this is the same principle as was described above for the convergence to the true Gaussian parameters in Figure 4.3. This is presumably no coincidence: better estimates of f_m and f_f produce more accurate ADS estimations, therefore in turn producing more accurate transition model estimations. In other words, the transition model estimate is indirectly dependent on the accuracy of the learned Gaussian parameters.

It is further notable in Table 4.3 that in none of the cases, the true transition model was exactly reached. This is likely because a single misclassification 'within' a patch results in a transition from m to f and a transition from f to m. Therefore, this number will always have some sort of inflation in a two-state case with a non-perfect score; when the states are estimated perfectly, the transition model is by definition equal to the learned model (section 3.4). This is further displayed by the fact that in the free and equal scenarios (which can be considered easier to classify considering the maximum scores achieved), the true transition model was approached more closely. This is the result of fewer misclassifications. The expectation is that in a case with more than two states this would have less of an effect. In a two-state case, a misclassification automatically results in an estimate for the 'opposite' state, always degrading the estimate for both transition probabilities. When there are more states, it is still possible to approach the transition probabilities between some states even when the overall score is poor, as long as these transitions are properly estimated.

Table 4.3: The learned transition models of both learning methods. The true transition model is included. The absolute difference between the learned and the true probabilities are shown within brackets; the smallest difference is shown in bold.

		Baum-Welch		FWE		True	
Scenario	Variance	$P(m\|m)$	$P(m\|f)$	$P(m\|m)$	$P(m\|f)$	$P(m\|m)$	$P(m\|f)$
1: Free	High	0.706 (**0.040**)	0.049 (**0.004**)	0.599 (0.147)	0.053 (0.008)	0.746	0.045
	Low	0.673 (**0.044**)	0.053 (0.004)	0.611 (0.106)	0.051 (**0.002**)	0.717	0.049
2: Mud	High	0.934 (**0.019**)	0.307 (**0.052**)	0.900 (0.053)	0.583 (0.327)	0.953	0.255
	Low	0.917 (**0.033**)	0.367 (**0.100**)	0.897 (0.054)	0.612 (0.353)	0.950	0.267
3: Equal	High	0.856 (**0.031**)	0.127 (**0.018**)	0.750 (0.136)	0.173 (0.064)	0.887	0.109
	Low	0.849 (**0.034**)	0.134 (**0.022**)	0.734 (0.149)	0.178 (0.066)	0.883	0.112

In Figure 4.5 and Figure 4.6 the learning curves from the test data of all μ_{f_m} and for $\sigma_{f_m} = 0.2$ are shown. When closely examining these figures it can be found that most learning curves, especially in the free and equal scenarios, eventually converge to a single score. One might expect small differences, even when converged. An explanation is that classification is evaluated as a discrete value, meaning that tiny changes in the model will not have any substantial effect on the number of correct classifications and therefore the MCC. In turn, the MCC is based on discrete amounts, meaning that there is a set amount of values the MCC can assume.

Curious is that when considering the maximum score, we found that for all cases and both learning methods the baseline score was beaten, generalizing the conclusions drawn from Table 4.1. When considering the final score, this applies to almost all cases. Furthermore, in the mud scenario for FWE, seen in Figure 4.6c and Figure 4.6d, the same problem of overclassifying the mud state appeared for some μ_{f_m}. Notable is that this effect diminishes for a higher absolute distance from μ_{f_f} to μ_{f_m}. In addition, there seems to exist a pattern in that scores of $\mu_{f_m} > 1$ are lower than the scores of $\mu_{f_m} < 1$ when considering the same absolute distance. This is likely the result of the initial $\mu_{f_m} = 0.1$ being closer to the latter, giving a better initial estimate. In general, there is a relation between this absolute distance and the score obtained, with 0.25 always outperforming 0.1, and 0.5 always outperforming 0.25.

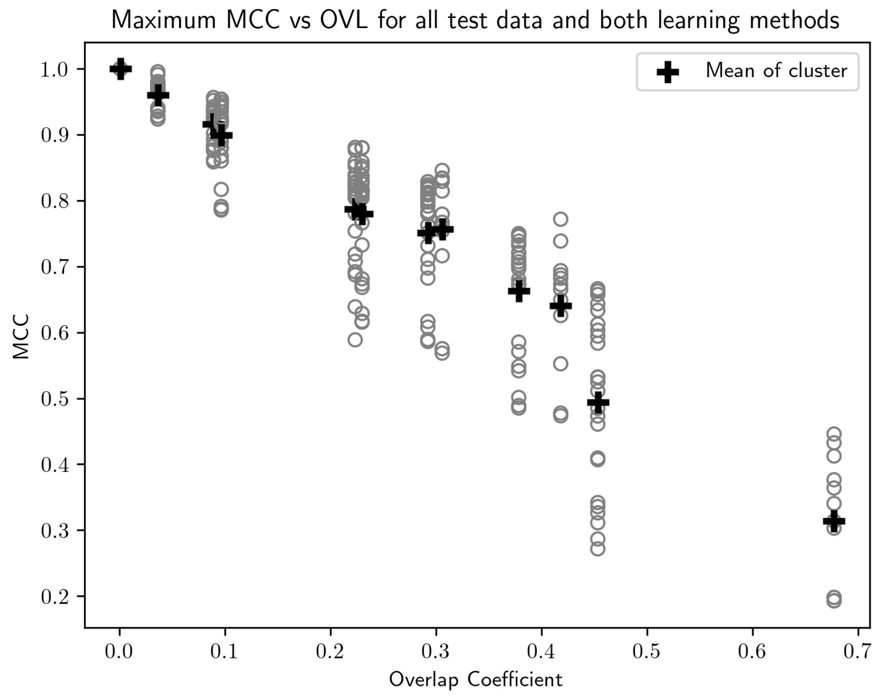


Figure 4.4: Matthews' correlation coefficient versus overlap coefficient for the maximum scores of all test data, both learning methods and all scenarios combined. The grey circles denote a single maximum score and a black cross denotes the mean of a cluster; there are 12 clusters in total, 3 different variances times 4 unique absolute distances to μ_{f_m}.

To further prove this point, Figure 4.4 displays the maximum MCC of all test data and all scenarios: this data includes both learning methods, all patch variances and all values of μ_{f_m} and σ_{f_m}, producing 252 maximum scores. In total 12 clusters of data grouped by the OVL exist. When considering the absolute distance from μ_{f_f} to μ_{f_m}, 4 unique values exist. Then, with 3 unique variances $3 * 4 = 12$ values of the OVL exist. The means of these clusters are taken to evaluate the effect of the OVL on the classifier performance. Interestingly enough these results show that, even for the results of a large range of experiment parameters combined in one graph, there exists a negative correlation between the OVL and the MCC.

When interpreting the results it is important to take the input of the system into account: in the experiments, the input was set to 1 and is kept equal at each timestep, resulting in $\mu_{f_f} = 1$. This was done for simplicity. In the simulation, the mean and variance of f_f and f_m would scale accordingly. But, varying the input is arbitrary for classification: the same relative effects apply to the velocity. This is effectively the same as changing the size of the patches. Therefore, the conclusions drawn from the results remain general for all inputs as long as it is assumed constant. Varying the input across the training iteration is a different case: this results in a more complex problem, as the ADS sequence is then dependent on the variation of the input as well.

4.2. Comparison

Apart from the overclassification problem of FWE, no direct comparison between the two learning methods is made yet, which is the goal of this section. First, it is important to stress the fact that for each experiment the test and training data for both learning methods are equal, making direct comparison possible.

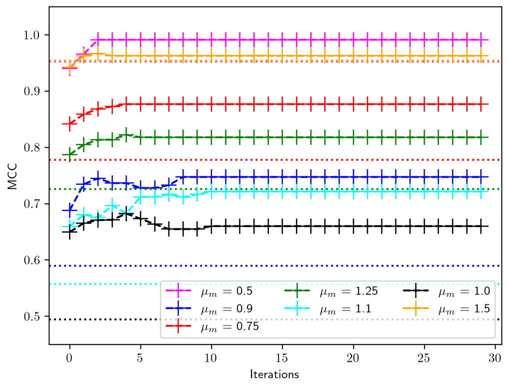
When considering Table 4.1, BW demonstrated superior results for the maximum scores apart from two cases. In these cases, FWE obtained a better final score as well. In general, in the free and equal scenario, FWE did approach the score obtained by BW very closely. In the mud scenario, FWE suffered from the overclassification problem as described in the previous section. When comparing Figure 4.5 and Figure 4.6 (containing the scores for all true μ_{f_m}) as additional evidence, BW can be considered the more stable method of the two learning methods, at least for the mud scenario. In none of the evaluated μ_{f_m}, BW shows as much of a performance drop as FWE does. In addition, the final score obtained by BW is in all of the cases better than the baseline, which is not the case for FWE in the mud scenario for most μ_{f_m}. It has to be noted however that in not all cases in the mud scenario, total convergence was reached yet. Therefore, for BW in particular, it is not certain that the final score remains better than the baseline when the number of training iterations is further increased.

For the case of convergence to the true parameters of f_m and f_f in Figure 4.3, we can see that in general BW exhibited superior performance. While FWE seemed to produce a better estimate to f_m in the mud scenario when considering Figure 4.3a, it is shown in Figure 4.3b that this was at the cost of a worse estimate of f_f.

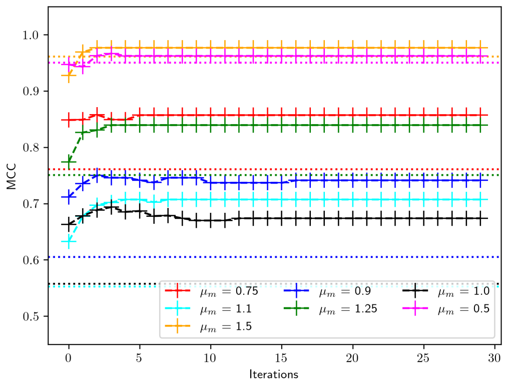
Comparing the convergence to the true transition model in Table 4.3, BW was able to outperform FWE in most of the cases. Only one probability was better estimated by FWE (free scenario, low variance), but when we consider the other estimated probability as well, the sum of the differences is still larger than those of the estimates found by BW.

Surprisingly, learning the exact values of the transition model and the Gaussian distributions did not seem to be crucial for a high score: in the free and equal scenarios with high variance, where FWE exhibited respectively a slightly better and equal score compared to BW, the estimates of FWE for the transition model and Gaussian distributions are substantially more off from the true values than BW.

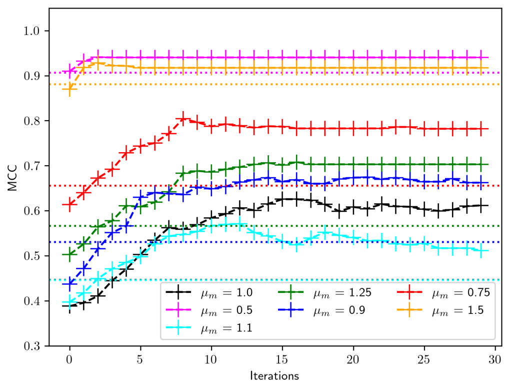
In summary, it was found that while BW showed consistent performance, FWE suffered from degrading performance in some cases. Essentially, BW was better able to approach the true values of both the transition model and the Gaussian parameters. However, FWE can attain comparable or even superior scores than BW in instances where its performance remains stable. Besides this, in all cases, both BW and FWE outperformed the baseline.



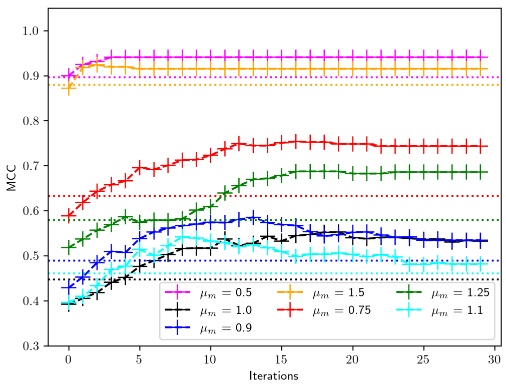
(a) Free scenario, high variance



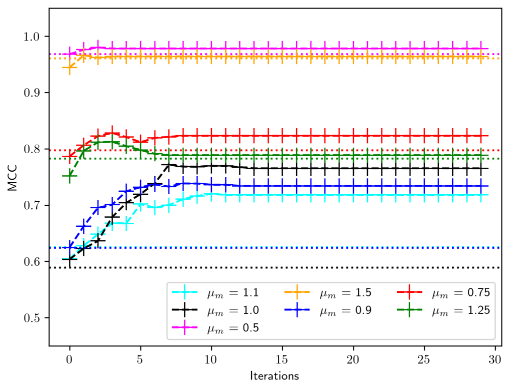
(b) Free scenario, low variance



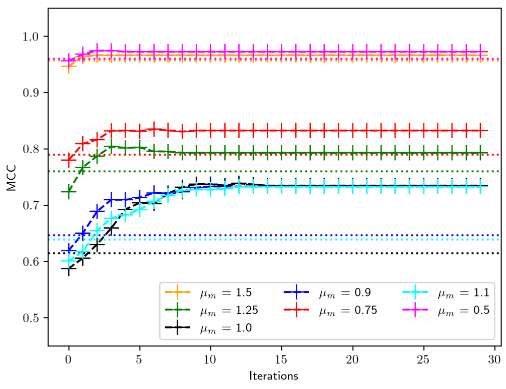
(c) Mud scenario, high variance



(d) Mud scenario, low variance

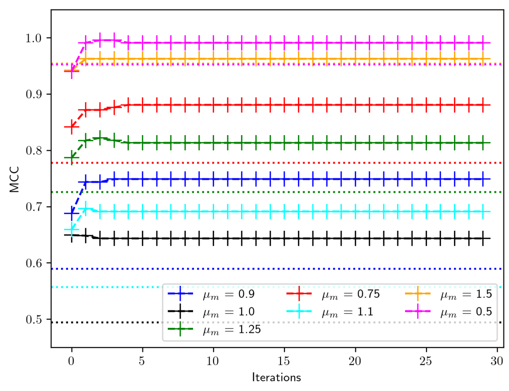


(e) Equal scenario, high variance

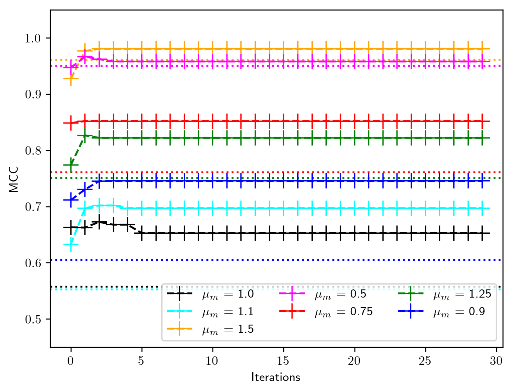


(f) Equal scenario, low variance

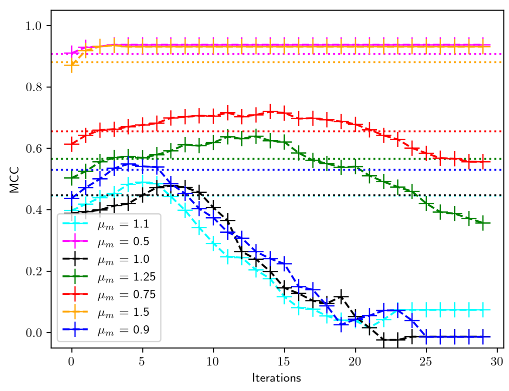
Figure 4.5: Test set Matthews' correlation coefficient for all mean scales when using **Baum-Welch** as learning method for the three scenarios and a mud variance of 0.2. The scores of the baselines, the likelihood estimator with perfect knowledge of the Gaussian distributions determining movement, are included as dotted lines.



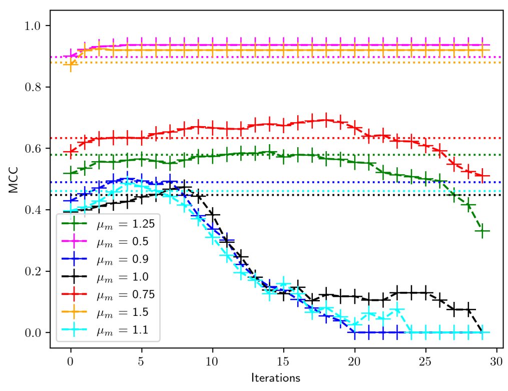
(a) Free scenario, high variance



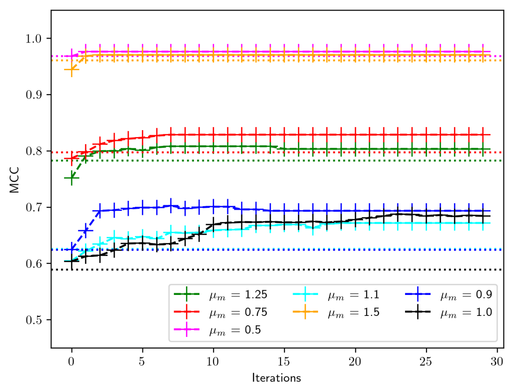
(b) Free scenario, low variance



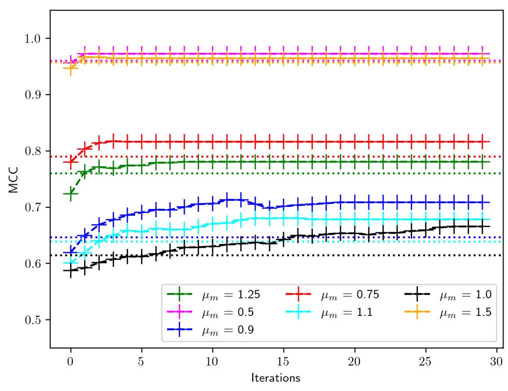
(c) Mud scenario, high variance



(d) Mud scenario, low variance



(e) Equal scenario, high variance



(f) Equal scenario, low variance

Figure 4.6: Test set Matthews' correlation coefficient for all mean scales when using **forward extraction** as learning method for the three scenarios and a mud variance of 0.2. The scores of the baselines, the likelihood estimator with perfect knowledge of the Gaussian distributions determining movement, are included as dotted lines.

4.3. Conclusion

In this chapter, the results of the experiments described in section 3.5 were evaluated. This was done by considering the classification scores of the learned model and the estimates of the true parameters of the GHMM. The scores were compared with a non-learning baseline estimator, that is a GHMM with perfect knowledge of the Gaussians determining movement in free and mud but with a flat transition model, meaning that the transition model has no influence on classification. It was found that in all cases the maximum score of the two learning methods, BW and FWE, exceeds the baseline score. For the score at the final training iteration, BW outperformed the baseline in every case and FWE in the ’free’ and ’equal’ scenarios. In some cases of the mud scenario, FWE displayed a gradual increase in performance until a turning point, where the score deteriorated. Inspection with the convergence charts showed that this problem was caused by an overclassification of the ’mud’ state. Furthermore, it was shown that there is a relation between the maximum reached score and the OVL: the lower the OVL (thus the less similar the ADS), the higher the maximum score. Considering the two learning methods, it was found that BW displayed better convergence to the true Gaussian parameters than FWE in addition to better scores. However, when the performance of FWE remained stable, it was able to attain comparable or even superior scores compared to BW.

5 Discussion

Foundings
The goal of this thesis was to design a *general* ADSE system with limited *prior knowledge*. It was found that a possible method for this is the GHMM. The results indicate that the GHMM can be used as ADSE given the basic description of an ADSE: it is able to detect an ADS change from indirectly related lower-level system data. Furthermore, from the results it was concluded that the GHMM improves upon the baseline, the LE, as an ADSE method. The results further suggested that there exists a correlation between the performance and the OVL of the two ADS; the less similar the ADS, the higher the classification score. Considering the learning methods, it was found that BW learns closer estimates of the true GHMM parameters. Besides this, FWE was still able to attain comparable or even superior scores compared to BW. Furthermore, BW was found to be a more stable method than FWE; FWE is theoretically much faster but prone to degrading performance in some cases. In some instances, such as the mud scenario, the ADSE ends up performing worse than the baseline if the training process continues.

Additionally, the GHMM outperforms the LE with *limited prior knowledge*: while the LE has access to the true Gaussian emissions of the ADS, only the emission of the 'normal' ADS is known for the GHMM-based ADSE. It was found, especially for BW, that the learned distributions of the GHMM converge closely to the true Gaussian emissions. In addition, the ADSE can learn an accurate representation of an unknown environment by learning a transition model. Therefore, it can be concluded that it is possible to learn unknown ADS, given that the number of ADS is known. In other words: we can conclude that the GHMM is suitable for an ADSE problem with *limited prior knowledge* on the ADS.

While the ADS of the problem is the concept of mud, it is safe to say that the same method applies to other ADS as well; from the definition of ADS in section 3.1, all ADS which can be described by a Gaussian distribution are in essence some form of the mudworld. For example, with a higher variance or a lower mean. From the results, it was found that the ADSE can successfully handle these cases as well, confirming that the GHMM can be considered a *general* method for this type of ADSE problem.

It remains difficult, however, to give an answer to how well exactly this system performs, or how *suitable* the system is for ADSE. It is problematic to connect the absolute difficulty of the problems to a score value: for example, can an MCC of 0.9 for a particular problem be considered good? Or would an MCC of 0.7 for that same problem be sufficient as well? For that, alternative ADSE systems have to be evaluated on the same problems, which is out of the scope of this thesis.

Concluding, the main takeaway is that the GHMM is suitable as a general ADSE with limited prior knowledge. The question remains to what extent the GHMM is suitable. For that, further research on the difficulty of this ADS problem is necessary. This includes the application of other ADSE systems on the same problems. Besides that, there exist some assumptions and further limitations to this study, as described in the next section. After that, this chapter will be concluded with recommendations for future work.

Limitations
Considering the simulation, there are a few limitations that have to be taken into account when interpreting the results. The first one is that the experiments are not run until convergence, therefore no general conclusion can be made on the convergence of this method. In addition, no multiple runs with a different random seed of the same case are done. The expectation is, however, that the conclusions still hold: since every set of training data will be generated from the same Gaussian distributions, adding more runs with a different random seed is in essence equal to extending the amount of evaluation data. And one could argue that since no quantitative evaluation is possible, the additional value of multiple runs is arbitrary. Another limitation is that one set amount of training data is used. It can be interesting to experiment for each case on what the minimum amount of training data could be to obtain similar results.

As described in section 3.4, FWE assumes a fully connected HMM. Therefore, it is assumed that this is the case in this thesis. While this can be considered a valid assumption for the mudworld, this has to be taken into account for different domains where ADS are unrelated; in this case, VBE is expected to be a more effective method.

Another assumption is that the Gaussian determining own movement is known. While in most realistic cases this applies and the system is able to work without this assumption, the performance may differ. Convergence will most likely be slower or to a value further from the true values. Regardless, this must be taken into account when applying the methods for a different study. A simple solution could be to use the first or mean of a first group of observations as mean for an ADS: this will take only one or a few timesteps and because a realistic initial guess is made for one of the ADS it is expected that less training iterations will be needed to approximate the Gaussian distributions.

Another limitation is that it is assumed that the mean of two ADS scales in the same manner: the ratio between the means of the two ADS remains constant. In these experiments, the same input of 1 is used for every timestep. This is something to consider for other dynamical systems or when a varying input is used, as not necessarily all ADS will scale linearly.

Recommendations for Future Work
The first and simplest recommendation for future work is to ’lock’ the parameters of the Gaussian of the known ADS. While the true parameters are injected into the model at the start, these can be adjusted during the learning process; this is contradictory to the assumption that the own ADS is known. From the results it was found that especially for FWE this distribution varied, possibly impacting the performance. It is possible that disabling learning for these parameters can improve the performance; essentially, the assumption was not fully leveraged.

An interesting piece of future work is the application to multiple ADS, firstmost where the number of ADS is known. While it is expected that the GHMM can handle this without problems, the current code would have to be extended. For further extending this to a setting where the amount of ADS is unknown, it is recommended to train multiple models in parallel and to subsequently use a model selection method such as the *Bayesian information criterion* [21] to assess which model is the best fit. A related extension could be to incorporate information from multiple dimensions, as now the system only has been tested on a 1D simulation. It is expected that the method is general enough to allow for this, as the dimensionality of the observations can be reduced to 1. Obviously, the computational needs will be greater as the world increases in complexity.

As an alternative method, the Gaussian mixture model (GMM, see section 2.3.1) was considered as an observation model for an HMM: instead of comparing the likelihood of two univariate Gaussians, a GMM was trained on the observation data. Then, samples were classified by the GMM and this classification was directly used as an observation model. It was found however that the problem is better described by a GHMM, as the emissions for each state are generated by univariate Gaussians and not a mixture. Using a GMM for this problem disables the knowledge that multiple ADS exist. Furthermore, since a GMM uses weights to describe the occurrence of each Gaussian in the mixture, the GMM implicitly represents the frequency of states, but not the sequence. An interesting extension related to the GMM is a *Gaussian mixture hidden Markov model*, where the emissions of each state are described by a GMM. Since GMMs can approximate many types of distributions given enough components, this can be an interesting extension to further generalize the ADSE. Simultaneously, this allows the ADSE to handle more complex ADS. This would not require many additions to the method; how to learn the parameters of a Gaussian mixture hidden Markov model is described in [2]. Note that in theory, Gaussian mixture emissions are already supported by the ADSE method as is, but the representing power of a univariate Gaussian is likely worse compared to that of a GMM for this kind of emission.

Within the methods evaluated, it is expected that a combination of these can be more computationally efficient than a single one while not sacrificing performance. If knowledge of the domain is present, this can be used to determine such a combination. For example, the low-computational cost LE is likely sufficient for cases where the OVL between the two ADS is small. For cases where the OVL is larger, a combination of the faster FWE and the more stable BW can prove useful: FWE could be used for the first training iterations to quickly gain a reasonable estimate, after which BW can be used to further improve the estimate and assure stable convergence. While it is apparent that FWE has the potential to be faster than BW since it requires only one pass versus two passes over the training data, this is not proven in this thesis: the focus of this thesis lies not within assessing the efficiency of an algorithm or implementing the algorithm in a computationally efficient manner. Therefore, the recommendation is to perform experiments on the computational speeds of these algorithms. This knowledge can then be used to further tune a LE, FWE, and BW combined ADSE.

In this thesis, FWE is used only on the forward data. The binary labels are determined during runtime after each timestep. This makes the algorithm very efficient, as a counter for each transition can be maintained at the same time. An interesting study could be to apply FWE on smoothed data; in that case, FWE will require more computational resources. But, then FWE is still slightly more efficient than BW since it does not require the calculation of the ξ function (Equation 2.20). The hypothesis is that, compared to only using forward data, this can improve results as well as aid in the performance drop problem due to the smoothed data being less sensitive to misclassified observations.

Considering FWE, some potential improvements to improve stability were conceived but not executed. This includes the introduction of a dynamic learning rate that changes the amount of data used for the learning process depending on the current phase in the learning process or the current convergence. Potentially, this could improve the stability by allowing quick learning of a reasonable estimate at the start of the learning process, which will then be updated less frequently as the learning process further continues. Another method is to apply an Exponential Weighted Moving Average or other similar measures to the learned parameters, weighing earlier learned parameters in the calculation for parameters learned further in the process. The expectation is that this reduces variance in the learning curve, therefore potentially overcoming the stability issues of FWE.

Another potential improvement that is not researched yet is to utilize early stopping. Since for most cases, the optimal score is reached within the first few training iterations, it can prove useful to stop the training process before the planned amount of iterations. The problem is that the training process is unsupervised, therefore no knowledge is available to the ADSE on the performance of the estimator during the training. What can be utilized is the convergence of parameters; if no change above a certain threshold occurs, the training process can be stopped. In another case, when domain knowledge or previous experience is available, this could be leveraged: for example, if the OVL is small, Figure 4.5 and Figure 4.6 show that only two or three training iterations can suffice.

Research on applying the system to a changing environment can be promising. A changing environment consists of different ADS that disappear or change over time. Essential for the application to a changing environment is to realize a threshold for keeping an ADS or rejecting it. This will additionally reduce the amount of prior information needed and further generalize the system. The KLD can be a suitable measure for such a threshold, as displayed in an intrusion detection paper using an HMM-based novelty detection method [37].

A final suggestion for future work includes the application of a different model to the ADSE problem. A *Recurrent Neural Network* (RNN) or *Long Short-term Memory* (LSTM) network seems fitting due to the ability to learn sequence-dependency within data. These networks can contain more parameters than an HMM, which is limited to the amount of pre-determined states. Therefore, these types of methods could potentially allow for more accurate ADSE. Care has to be taken in the case of limited prior knowledge, however: typically, RNNs and LSTMs are trained in a supervised manner, requiring data on the ADS. Furthermore, the simplicity of the HMM allows for interpretability, which is a classic shortcoming of deep learning methods.

6 Conclusion

For continuous dynamical systems, it can prove useful to construct a hierarchical model that detects changes in the *abstract discrete state* (ADS) of the system. ADS are high-level discrete system states that influence system dynamics. These are often not directly observable but can be inferred through analysis of simpler system data such as velocity or temperature. Knowledge of what ADS a system is in can aid in system control. For example, consider a wheeled robot in rough terrains, such as mud. Mud can indirectly be detected by a lower velocity for the same motor inputs due to slip. When entering a patch of mud, the system ADS changes from 'free' to 'mud'; control of the system can then change accordingly. The goal of this thesis was to create a *general abstract discrete state estimator* (ADSE) operating under *limited prior knowledge*: The ADSE acts as a system observer and estimates the current ADS of a system. Furthermore, the ADSE should be *general* as it should be applicable to multiple ADS problems and should be able to do so without *prior knowledge* of the ADS to be classified.

In order to reach this goal, the *Gaussian hidden Markov model* (GHMM) framework was found to be a suitable model for the ADS problem. The GHMM is an extension of the hidden Markov model, a stochastic modeling framework used for sequences and time series. This framework contains the assumption that the system state is not directly observable, fitting into the philosophy of the ADSE; the same assumption applies to the ADS problem. The parameters of a GHMM can be learned in an unsupervised manner by the Baum-Welch (BW) algorithm or by forward extraction (FWE). FWE was introduced in this thesis and is a simpler, theoretically more computationally efficient method than BW and is inspired by Viterbi extraction. FWE attains this efficiency by leveraging the assumptions of ADSE. Both learning methods are based on expectation maximization.

In order to research the applicability of the GHMM to the ADSE problem, experiments were done for a characteristic ADSE problem. Here, the observed data was generated by two Gaussian distributions, each belonging to one ADS. In this problem, three scenarios were generated, producing consecutive sequences of ADS of which the lengths are in turn generated by Gaussian distributions. Furthermore, the mean and variance for the Gaussian distribution belonging to the unknown ADS were varied, creating a total of 126 experiments per learning method. Both learning methods were compared to a simple likelihood-based baseline estimator. The baseline operated similarly to a GHMM with perfect knowledge of the Gaussian distributions of the ADS. Additionally, the baseline used a flat transition model, therefore not taking sequence into account.

The results of these experiments displayed superior performance for both learning methods compared to the baseline when the maximum test score is considered. When the test score of the final training iteration is considered, BW outperformed the baseline in every case, while FWE did not; a degrading trend after a gradually increasing performance was shown in one of the scenarios. This was found to be related to the overclassification of one ADS. Furthermore, it was shown that there exists a relation between the maximum reached score and the overlap coefficient of the two ADS. This suggests that the theoretical limit of this method is related to this measure. When comparing BW and FWE, BW shows no overclassification issues like FWE, resulting in more stable usage. In addition, BW shows better convergence to the true parameters of the GHMM. Besides this, FWE was able to obtain scores comparable to or even surpassing the scores obtained by BW.

The results suggest that the GHMM is a viable method for ADSE: besides consistently outperforming the baseline, globally the GHMM is able to detect the current ADS of the system. It remains problematic to judge the exact performance of the system since no quantitative measure for the problem difficulty is known. For this, different ADSE systems need to be examined on the same problem. Furthermore, the ADSE works for different variances and means belonging to the ADS, suggesting that different concepts of ADS can also be estimated. While the scenarios in the experiments are defined within the concept of 'mud', the concepts of ADS for other cases will be similar: data generated by differing latent dynamics. Therefore, the GHMM can be considered a *general* method for ADSE. Since the system is able to estimate ADS with only the assumptions that the total number of ADS and the behavior of the 'normal' ADS are known, it is safe to say that the GHMM can be considered a method for ADSE under *limited prior knowledge* as well.

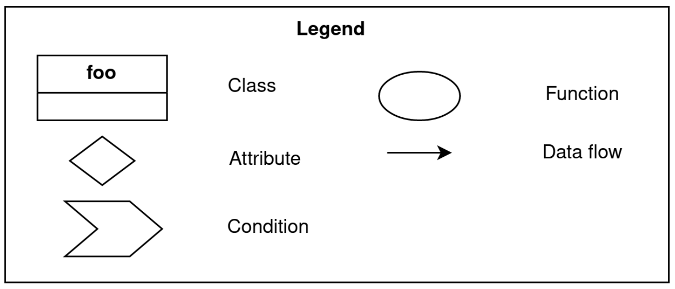
It is assumed that the observations emitted by an ADS are normally distributed, limiting the representation power of the ADSE. Therefore, a recommendation for future work is to extend the univariate Gaussian assumption to a Gaussian mixture model. Then, the problem can be described by a Gaussian mixture midden Markov model. With this extension, it is expected that the ADSE will be able to accurately portray more complex ADS, as a GMM is able to model more complex distributions. Likely, this can be built upon the current ADSE without many fundamental changes. Another suggestion for future work is to combine BW and FWE during training. FWE is theoretically a much faster method than BW, but unstable in some cases. This characteristic suggests that a combination of these methods can be a viable option: the initial few training iterations can be done by FWE, after which BW can be used for stable convergence. Further future work can include the extension to multiple states or dimensions, or application to a changing environment.

A
Software Architecture

A large part of the work done in this thesis was the development of custom programs in Python to create and validate the ADSE. The full code can be found here. A schematic, simplified depiction of the software can be found in Figure A.1. The classes used are:

- *Dynamics*, a class that describes the environment and handles the dynamics (in other words, the movement of the robot). This includes generating patches of mud.
- *ADSE*, the ADSE handles the classification of ADS from velocity data: the function 'Classify' is essentially a forward pass.
- *Learner*, this class handles the learning process of the ADSE for either BW or FWE learning. For BW this includes the backward pass, smoothing, and the calculation of the ξ-function. The class is built on an ADSE object.
- *Noise*, a utility class building on SciPy (scipy.stats.norm). This class contains functions for easier manipulation of scipy.stats.norm objects. This class is used as an attribute in both Dynamics and ADSE.
- *Simulation/Visualisation*, either a Simulation or Visualisation object can be used. This is a driver class for Dynamics and the ADSE. It handles the input, which can be directly from a keyboard or from data. A Visualisation adds various real-time plots to a Simulation, including a world plot with the robot movement, histogram and current ADS estimate. Furthermore, the Simulation/Visualisation calls the ADSE to classify on the velocity data on each timestep coming from the Dynamics.
- *Experiment*, a class containing and connecting a Simulation, Learner and ADSE object. It is used for easy running of multiple experiments with multiple settings. Furthermore, this contains functions to evaluate the performance of the ADSE and functions for handling and plotting data.

Globally, an experiment is run as follows: first, all classes are initiated. This includes defining the Gaussian distributions for generating patches and movement, defining the initial ADSE conditions and initializing the train and test dynamics. A pass through the test data is done with the initial model, providing a score described by the MCC. Then, a simulation run is done by iteratively applying the input data to the dynamics. In each timestep, the forward probabilities are calculated. When the simulation run is complete, performance on the train data is evaluated. Then, the preferred learning method is called and the new transition model and Gaussian estimates are injected into the learning ADSE. This updated model is then passed through the test data to assess the performance. This process is repeated until the desired number of training iterations is reached.



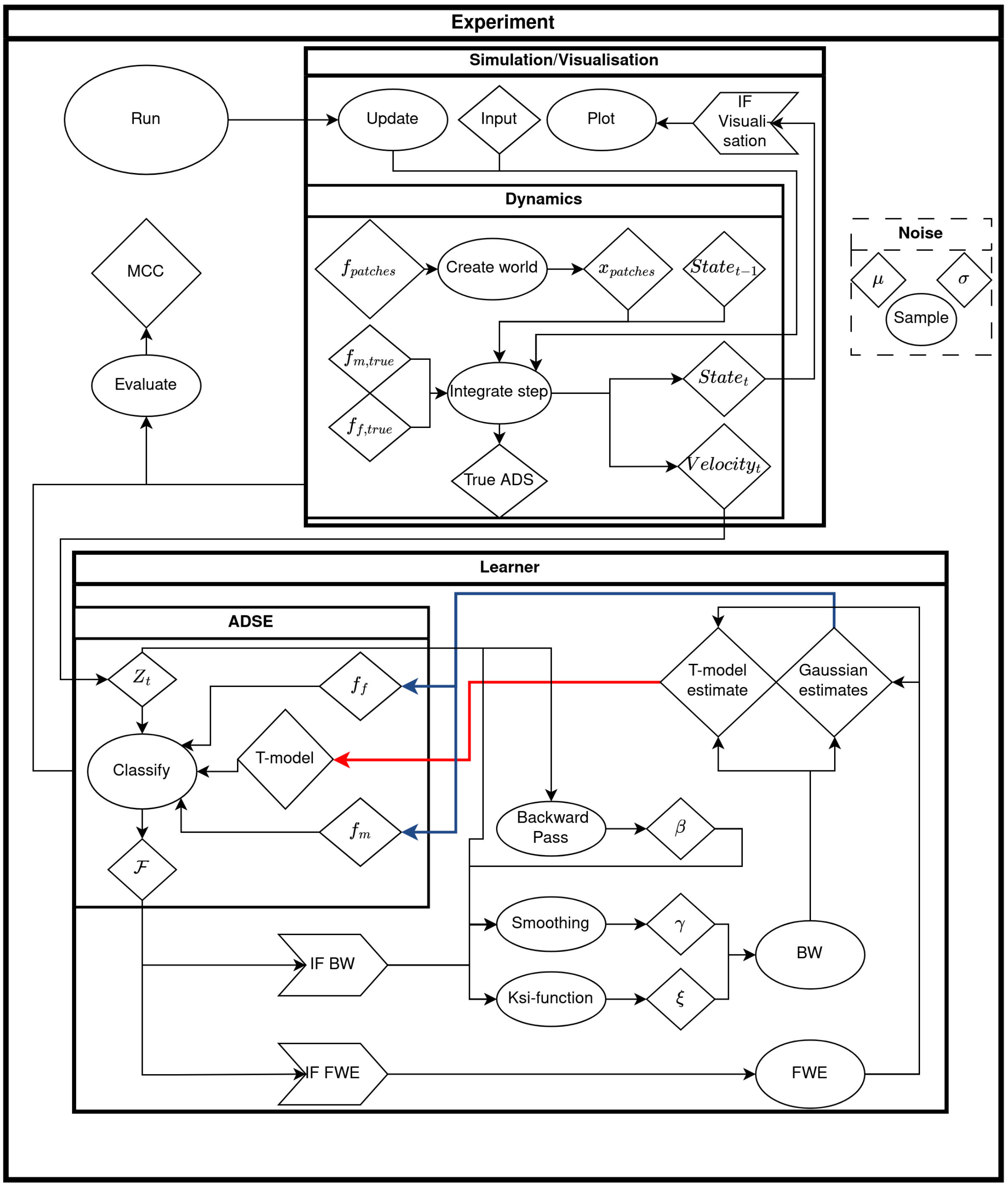
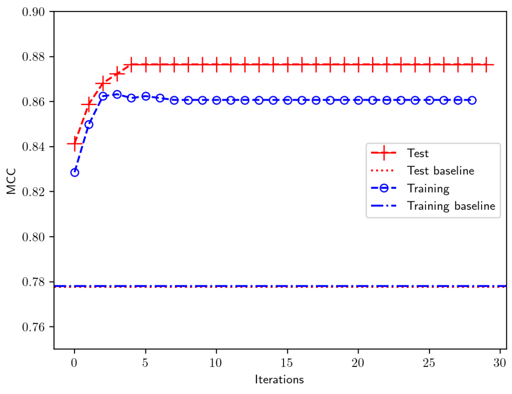


Figure A.1: A schematic depiction of the software developed for this thesis.

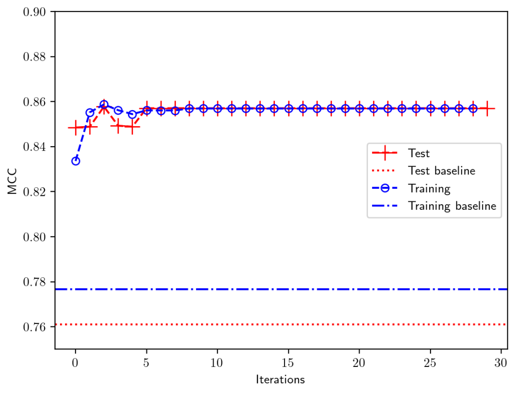
B
Detailed Results

In this appendix additional figures for all scenarios can be found for the single variance and mean case ($\mu_{f_m} = 0.75$, $\sigma_{f_m} = 0.75$), which were evaluated in more detail in section 4.1. From these figures, the values in Table 4.1, Table 4.2, Table 4.3, and Figure 4.3 are retrieved. More specifically, the scores coincide with the learning curves in Figure B.1 and Figure B.2, the transition model values withFigure B.3 and Figure B.4, and the KLD values with Figure B.5 and Figure B.6, all for BW and FWE respectively. All these figures contain the evolution of their respective quantity throughout the learning process.

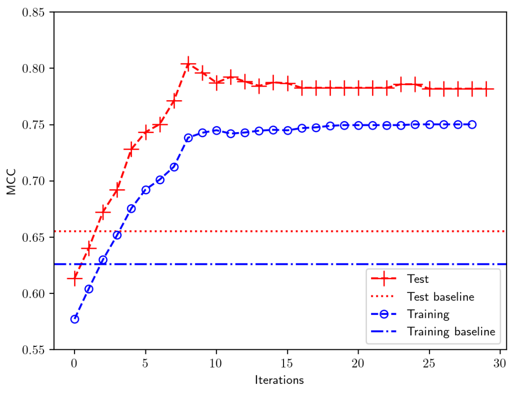
Note that some duplicate figures exist, which are in order of appearance in section 4.1: Figure B.6c, Figure B.6d, Figure B.4c, Figure B.4d, Figure B.6c, Figure B.6d, Figure B.5c, Figure B.5d. These are duplicated in this chapter to allow for a more direct comparison between each specific scenario.



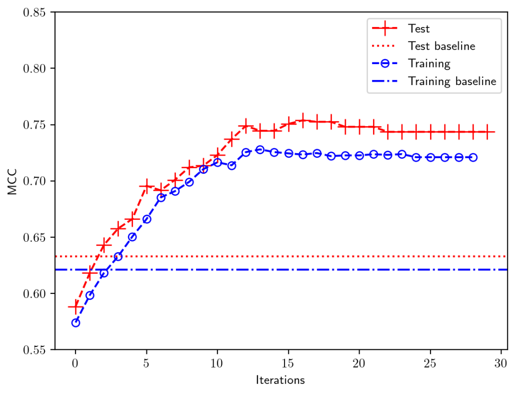
(a) Free scenario, high variance



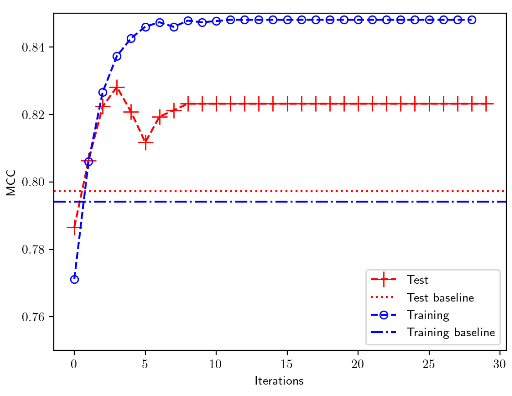
(b) Free scenario, low variance



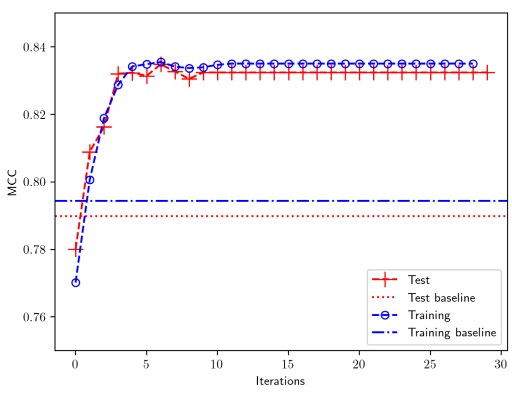
(c) Mud scenario, high variance



(d) Mud scenario, low variance

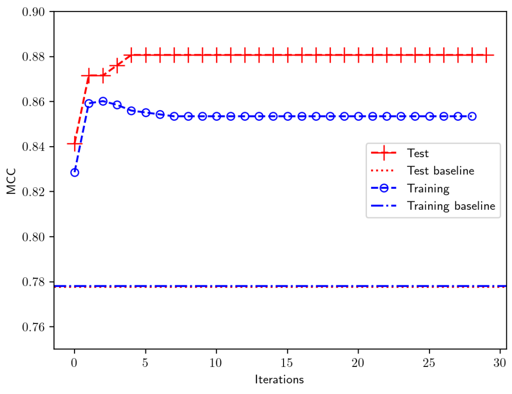


(e) Equal scenario, high variance

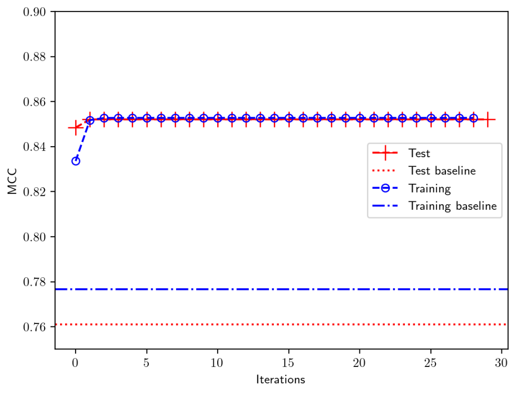


(f) Equal scenario, low variance

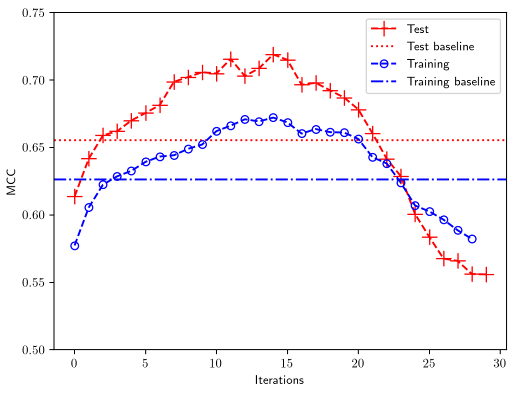
Figure B.1: Matthews' correlation coefficient when using **Baum-Welch** learning for the three scenarios and a mean scale of 0.75. The scores of the baselines, the likelihood estimator with perfect knowledge of the Gaussian distributions determining movement, are included as dotted lines.



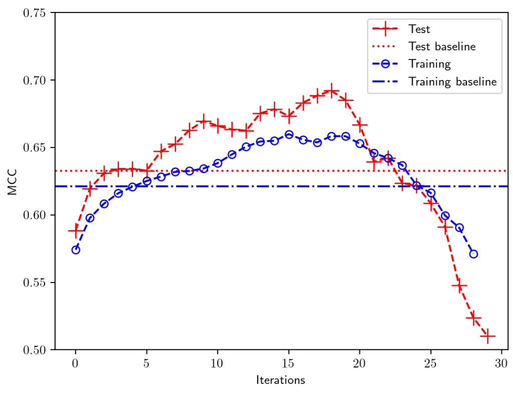
(a) Free scenario, high variance



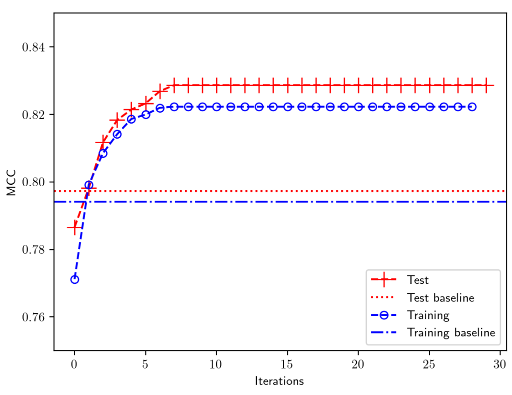
(b) Free scenario, low variance



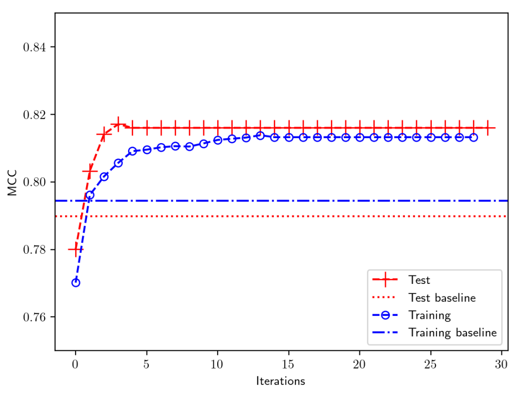
(c) Mud scenario, high variance



(d) Mud scenario, low variance

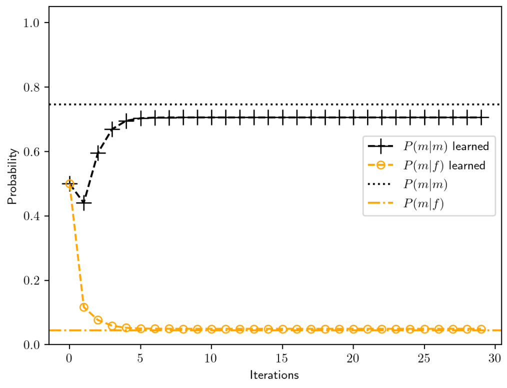


(e) Equal scenario, high variance

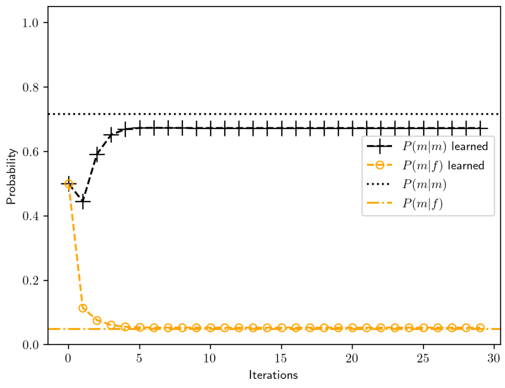


(f) Equal scenario, low variance

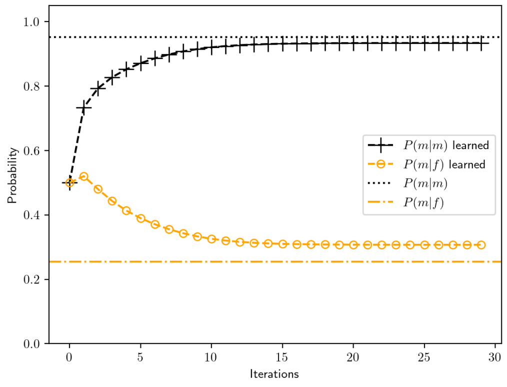
Figure B.2: Matthews' correlation coefficient when using **Forward Extraction** as learning method for the three scenarios and a mean scale of 0.75. The scores of the baselines, the likelihood estimator with perfect knowledge of the Gaussian distributions determining movement, are included as dotted lines.



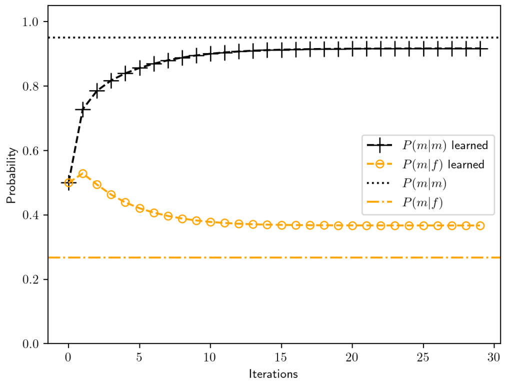
(a) Free scenario, high variance



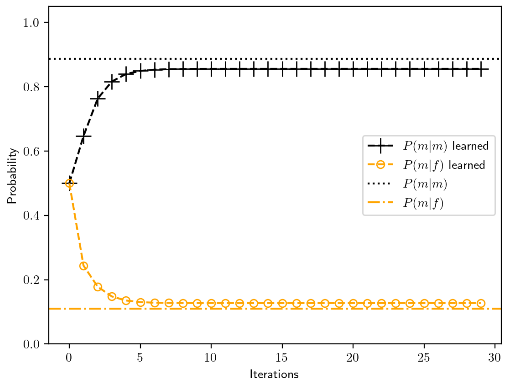
(b) Free scenario, low variance



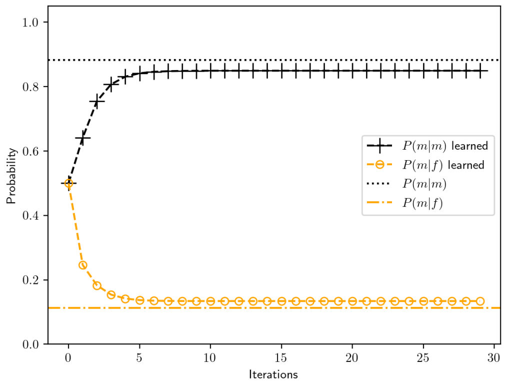
(c) Mud scenario, high variance



(d) Mud scenario, low variance

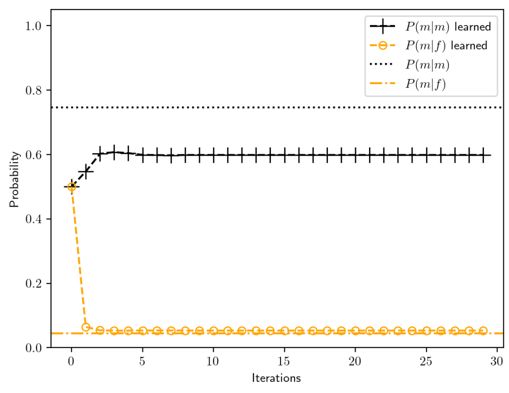


(e) Equal scenario, high variance

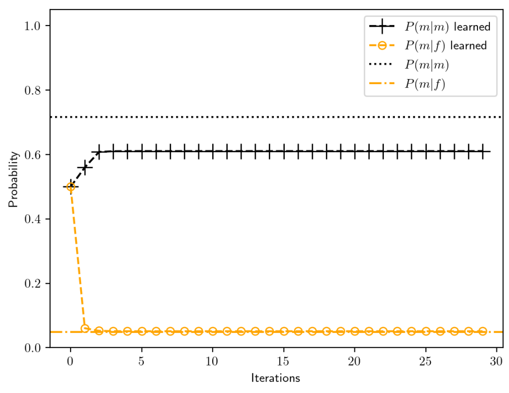


(f) Equal scenario, low variance

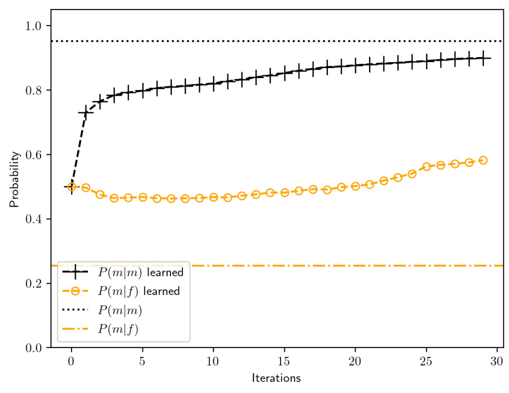
Figure B.3: Probabilities of learned **transition model** when using **Baum-Welch** learning for the three scenarios and a mean scale of 0.75. The horizontal lines denote the probabilities determined by supervised learning.



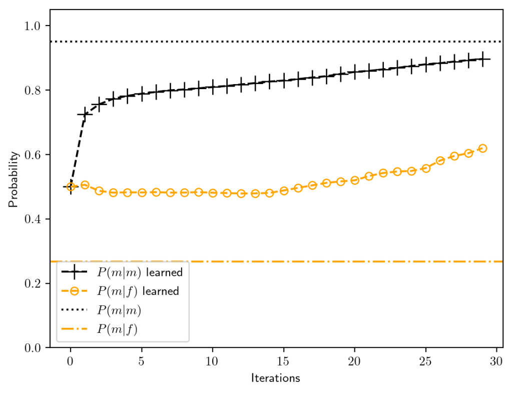
(a) Free scenario, high variance



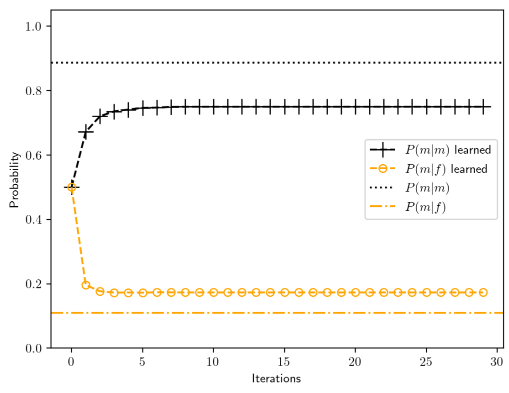
(b) Free scenario, low variance



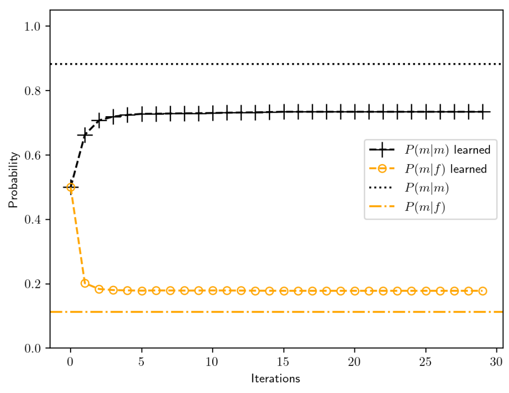
(c) Mud scenario, high variance



(d) Mud scenario, low variance

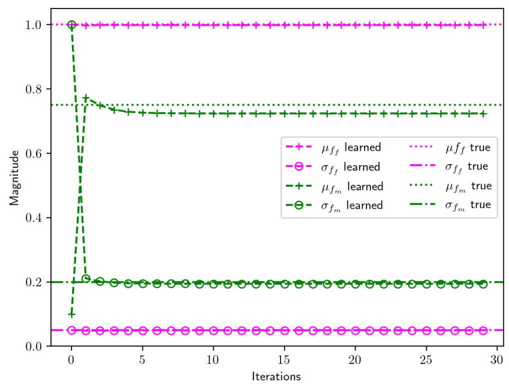


(e) Equal scenario, high variance

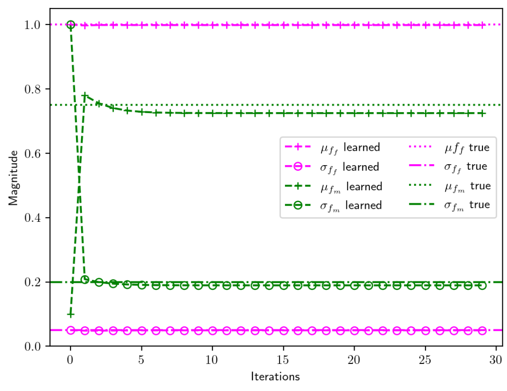


(f) Equal scenario, low variance

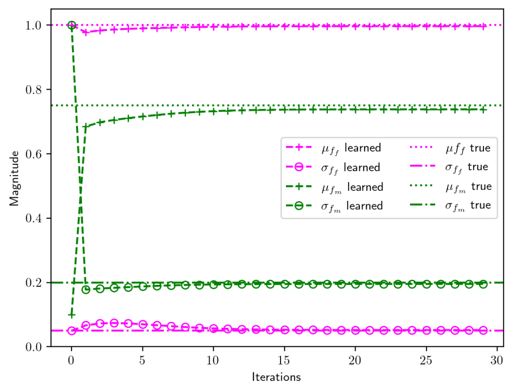
Figure B.4: Probabilities of learned **transition model** when using **Forward Extraction** for the three scenarios and a mean scale of 0.75. The horizontal lines denote the probabilities determined by supervised learning.



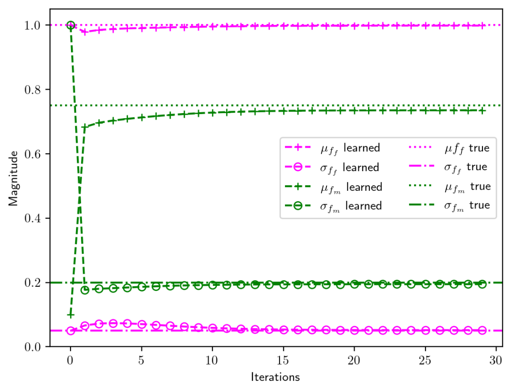
(a) Free scenario, high variance



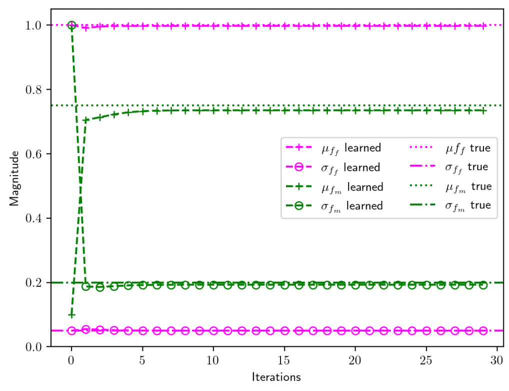
(b) Free scenario, low variance



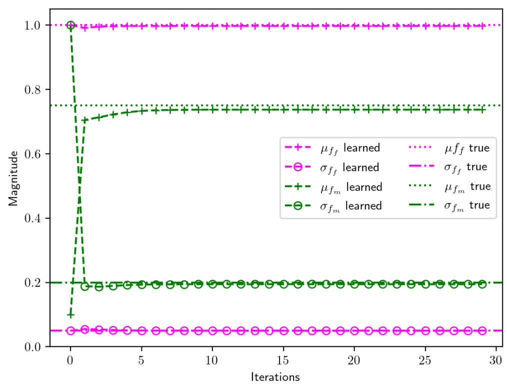
(c) Mud scenario, high variance



(d) Mud scenario, low variance

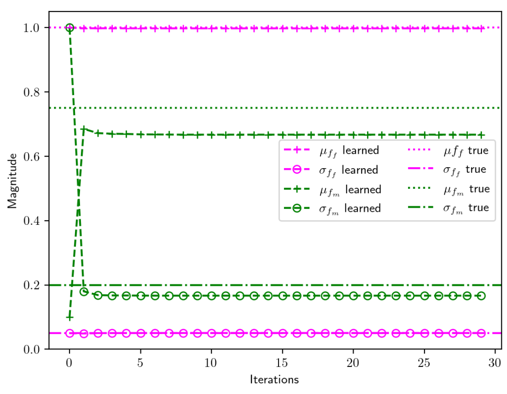


(e) Equal scenario, high variance

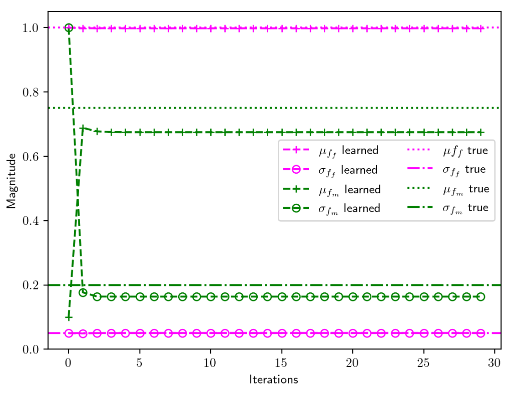


(f) Equal scenario, low variance

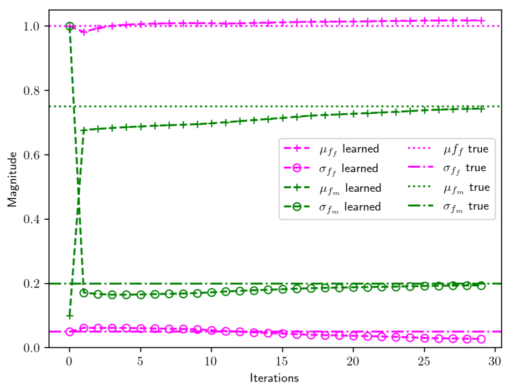
Figure B.5: Magnitude of means and variances of learned **Gaussian distributions** when using **Baum-Welch** learning for the three scenarios and a mean scale of 0.75. The horizontal lines denote the true mean and variance of the noise attempted to learn.



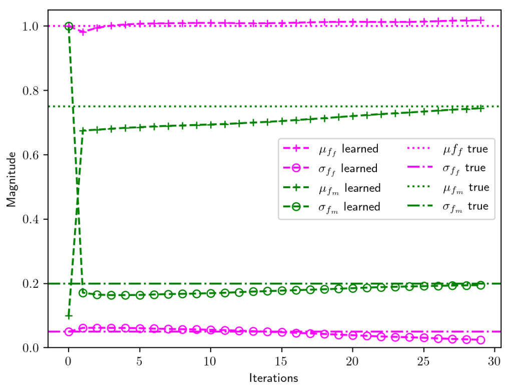
(a) Free scenario, high variance



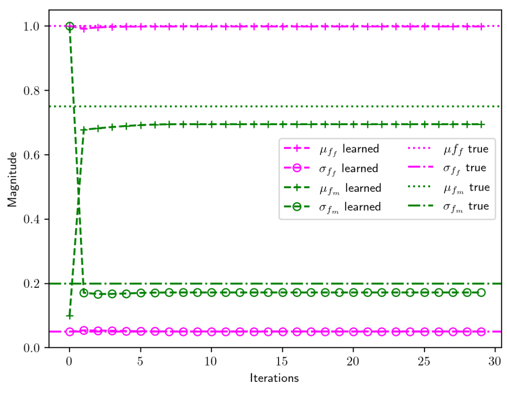
(b) Free scenario, low variance



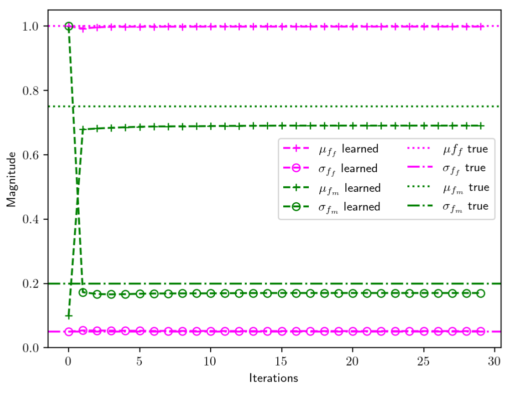
(c) Mud scenario, high variance



(d) Mud scenario, low variance



(e) Equal scenario, high variance



(f) Equal scenario, low variance

Figure B.6: Magnitude of means and variances of learned **Gaussian distributions** when using **Forward Extraction** for the three scenarios and a mean scale of 0.75. The horizontal lines denote the true mean and variance of the noise attempted to learn.

References

[1] Dmitry I Belov and Ronald D Armstrong. “Distributions of the Kullback–Leibler divergence with applications”. In: *British Journal of Mathematical and Statistical Psychology* 64.2 (2011), pp. 291–309.

[2] Jeff A Bilmes et al. “A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models”. In: *International computer science institute* 4.510 (1998), p. 126.

[3] Ewan Birney. “Hidden Markov models in biological sequence analysis”. In: *IBM journal of Research and Development* 45.3.4 (2001), pp. 449–454.

[4] Wouter de Boer. *On the Applicability of POMDPs for Abstract Discrete State Estimation [Unpublished Literature Review]*. Sept. 2022, p. 2.

[5] Wouter de Boer. *On the Applicability of POMDPs for Abstract Discrete State Estimation [Unpublished Literature Review]*. Sept. 2022, pp. 2–4.

[6] Wouter de Boer. *On the Applicability of POMDPs for Abstract Discrete State Estimation [Unpublished Literature Review]*. Sept. 2022, pp. 4–5.

[7] Michael S Branicky. “Introduction to hybrid systems”. In: *Handbook of networked and embedded control systems* (2005), pp. 91–116.

[8] Davide Chicco and Giuseppe Jurman. “An invitation to greater use of Matthews correlation coefficient (MCC) in robotics and artificial intelligence”. In: *Frontiers in Robotics and AI* (2022), p. 78.

[9] Davide Chicco and Giuseppe Jurman. “The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation”. In: *BMC genomics* 21 (2020), pp. 1–13.

[10] Arthur P Dempster, Nan M Laird, and Donald B Rubin. “Maximum likelihood from incomplete data via the EM algorithm”. In: *Journal of the royal statistical society: series B (methodological)* 39.1 (1977), pp. 1–22.

[11] Chuong B Do and Serafim Batzoglou. “What is the expectation maximization algorithm?” In: *Nature biotechnology* 26.8 (2008), pp. 897–899.

[12] Rakesh Dugad and UDAY B Desai. “A tutorial on hidden Markov models”. In: *Signal Processing and Artifical Neural Networks Laboratory, Dept of Electrical Engineering, Indian Institute of Technology, Bombay Technical Report No.: SPANN-96.1* (1996).

[13] Yariv Ephraim and Neri Merhav. “Hidden markov processes”. In: *IEEE Transactions on information theory* 48.6 (2002), pp. 1518–1569.

[14] Ramsey Faragher. “Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes]”. In: *IEEE Signal processing magazine* 29.5 (2012), pp. 128–132.

[15] G David Forney. “The viterbi algorithm”. In: *Proceedings of the IEEE* 61.3 (1973), pp. 268–278.

[16] Henry F Inman and Edwin L Bradley Jr. “The overlapping coefficient as a measure of agreement between probability distributions and point estimation of the overlap of two normal densities”. In: *Communications in Statistics-theory and Methods* 18.10 (1989), pp. 3851–3874.

[17] Nathalie Japkowicz, Catherine Myers, Mark Gluck, et al. “A novelty detection approach to classification”. In: *IJCAI*. Vol. 1. Citeseer. 1995, pp. 518–523.

[18] Frederick Jelinek. “Continuous speech recognition by statistical methods”. In: *Proceedings of the IEEE* 64.4 (1976), pp. 532–556.

[19] Rogemar S Mamon and Robert James Elliott. *Hidden Markov models in finance*. Vol. 4. Springer, 2007.

[20] Markos Markou and Sameer Singh. “Novelty detection: a review—part 1: statistical approaches”. In: *Signal processing* 83.12 (2003), pp. 2481–2497.

[21] Geoffrey J McLachlan and Suren Rathnayake. “On the number of components in a Gaussian mixture model”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 4.5 (2014), pp. 341–355.

[22] Todd K Moon. “The expectation-maximization algorithm”. In: *IEEE Signal processing magazine* 13.6 (1996), pp. 47–60.

[23] Stephanie Pancoast, Murat Akbacak, and Michelle Sanchez. “Supervised acoustic concept extraction for multimedia event detection”. In: Nov. 2012, pp. 9–14. DOI: `10.1145/2390214.2390219`.

[24] Marco AF Pimentel et al. “A review of novelty detection”. In: *Signal processing* 99 (2014), pp. 215–249.

[25] Mark Pinsky and Samuel Karlin. *An introduction to stochastic modeling*. Academic press, 2010.

[26] Lawrence Rabiner and Biinghwang Juang. “An introduction to hidden Markov models”. In: *ieee assp magazine* 3.1 (1986), pp. 4–16.

[27] 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.

[28] Stuart Jonathan Russell and Peter Norvig. *Artificial Intelligence: A modern approach*. Pearson, 2022, pp. 566–567.

[29] Stuart Jonathan Russell and Peter Norvig. *Artificial Intelligence: A modern approach*. Pearson, 2022, pp. 571–573.

[30] Stuart Jonathan Russell and Peter Norvig. *Artificial Intelligence: A modern approach*. Pearson, 2022, pp. 574–576.

[31] Stuart Jonathan Russell and Peter Norvig. *Artificial Intelligence: A modern approach*. Pearson, 2022, pp. 817–820.

[32] Keiichi Tokuda et al. “Speech synthesis based on hidden Markov models”. In: *Proceedings of the IEEE* 101.5 (2013), pp. 1234–1252.

[33] Stephen Tu. “Derivation of baum-welch algorithm for hidden markov models”. In: *URL: https://people. eecs. berkeley. edu/~ stephentu/writeups/hmm-baum-welch-derivation. pdf* (2015).

[34] Hongzhi Wang, Mohamed Jaward Bah, and Mohamed Hammad. “Progress in outlier detection techniques: A survey”. In: *Ieee Access* 7 (2019), pp. 107964–108000.

[35] Greg Welch, Gary Bishop, et al. “An introduction to the Kalman filter”. In: (1995).

[36] Max Welling. “Hidden Markov Models”. In: (Nov. 2000).

[37] Dit-Yan Yeung and Yuxin Ding. “Host-based intrusion detection using dynamic and static behavioral models”. In: *Pattern recognition* 36.1 (2003), pp. 229–243.