LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

INSTITUT FÜR STATISTIK

Georg Pfundstein

Hidden Markov Models with Generalised Emission Distribution for the Analysis of High-Dimensional, Non-Euclidean Data

Master Thesis

Supervision: Dr. Achim Tresch – Gene Center
Prof. Dr. Torsten Hothorn – Department of Statistics
University of Munich

19th December 2011

Abstract

Hidden Markov models (HMM) are tremendously popular for the analysis of sequential data, such as biological sequences, speech recognition as well as gesture recognition. However, since the method has got some limitations, that is mainly the restrictive emission distribution assumption in each hidden state, a generalised extension of the ordinary HMM is introduced. The method proposed in this work aims to overcome this limitation through adapting the multivariate Gaussian density so it can handle data obtained from non-Euclidean metric space. The generalised emission distribution is only dependent on the pairwise distances of all observations and no longer on a center of mass nor a variance term. We show that our method performs as good as the original HMM in many scenarios and even outperforms it in a certain non-Euclidean data situation. In addition we apply the method to ChIP-chip data in order to find out whether or not we can determine distinct gene classes that can be distinguished by different transcription state sequences.

Keywords: Hidden Markov model, ChIP-chip, L_q norm, non-Euclidean, Viterbi Algorithm, Baum-Welch Algorithm

Acknowledgment

First of all I would like to thank Dr. Achim Tresch from the Gene Center Munich for his outstanding supervision and support during my work on this thesis. I am equally grateful to Prof. Dr. Torsten Hothorn for taking over the supervision on the part of the University of Munich and for giving me the freedom on choosing the topic of this thesis. Furthermore, I send my thanks to Martina Feilke and Oliver Kühnle for proof-reading and constructive criticism. Last but definitely not least, a very big "thank you" goes to my family who always support me in what I am doing. I owe you something...

Contents

1. Introduction

Hidden Markov models (HMM) (Baum et al., 1970) are statistical models that are closely related, as the name already suggests, to Markov models. In contrast to ordinary Markov models, where the states are directly visible, the states are not directly visible in the case of HMM. However, conclusions on the hidden states can be drawn from an observable output which is depending on the states. In other words, each hidden state holds different probability distributions that produce observable output.

Hidden Markov models are particularly applicable to biological sequence data such as ChIP-chip data (Mayer et al., 2010). ChIP-chip is a technology for the identification and characterisation of the DNA binding motifs of bound transcription factors. The goal of this thesis is to identify distinct gene classes, that are characterised by their specific sequence of transcription modes. With the help of hidden Markov models we might be able to determine such gene classes, in case there are any. In terms of hidden Markov models the different transcription factor modes, which are not observable correspond to the hidden states and the expression level of these transcription factors correspond to the visible output.

When applying the hidden Markov model in practice, a multivariate Gaussian emission distribution in each hidden state is assumed most of the time. This is essentially based on the fact that there are efficient parameter estimators for this special case. This distribution assumption, however, does not hold true for many data sets, including the ChIP-chip data provided by Mayer et al. (2010). Due to this limitation we improve the HMM in order to make it more applicable to non-Gaussian data. The extension of the HMM we are going to propose in this work is based on a generalised emission density. The new derived emission distribution is expressed in terms of pairwise distances between all observations and their probabilities of belonging to a certain state and does no longer depend on a cluster center or a cluster variance. The advantage of this approach is that in this way we are able to use different distance measures which are well adjusted for the particular data situation of interest.

In the beginning of chapter two an introduction to the general hidden Markov model theory is given. Based on this, the generalised hidden Markov model (GHMM) is proposed and the corresponding parameter estimators are derived. Chapter three gives an insight into the performance of the introduced GHMM by applying the method to data generated by different simulation scenarios. It is shown that the proposed method is able to outperform the original HMM in a certain data situation. In chapter four the introduced generalised hidden Markov

model is applied to ChIP-chip data. We show that the GHMM is able to identify biologically meaningful transcription states. At the end of this thesis a discussion and a prospect for further research in this context is given.

2. Methods

2.1. Hidden Markov Models (HMM)

Ordinary Markov chains are often not flexible enough for the analysis of real world data, as the state corresponding to a specific event (observation) has to be known. However, in many problems of interest this is not given. Hidden Markov models (HMM) as originally proposed by Baum et al. (1970) can be viewed as an extension of Markov chains. The only difference compared to common Markov chains is, that the state sequence corresponding to a particular observation sequence is not observable but hidden. In other words, the observation is a probabilistic function of the state, whereas the underlying state sequence itself is a hidden stochastic process (Rabiner, 1989) (see figures 2.1 and 2.2). That means, the underlying state sequence can only be observed indirectly through another stochastic process that emits an observable output. Hidden Markov models are extremely popular when dealing with sequential data, such as speech recognition, gesture recognition (Duda et al., 2001) as well as biological sequences.

Before going into detail, let us first fix some notation used throughout this thesis. A complete overview about the notation used in this thesis is given in appendix A.

- K: Discrete number of hidden states in the model (e.g. $K = 6$).
- T: Discrete number of observations (time points) in a sequence (e.g. $T = 1000$).
- $\mathcal{S} = (s_1, s_2, ..., s_T)$: Hidden state sequence, which should be determined.
- $\mathcal{O} = (O_1, O_2, ..., O_T)$: Observation sequence.
- $\pi = (\pi_1, ..., \pi_K)$: Initial state probabilities where $\pi_i = P(s_1 = i)$ and $\sum_{i=1}^{K} \pi_i = 1$.
- $A = \{a_{ij} | i = 1, ..., K; j = 1, ..., K\}$: State transition probability where $a_{ij} = P(s_{t+1} = j | s_t = i)$ and $\sum_{j=1}^{K} a_{ij} = 1$ (see figure 2.1).
- $B = \{b_k(O_t) | k = 1, .., K; t = 1, ..., T\}$: Observation probability where $b_k(O_t) = P(O_t | s_t = k)$. Typically a multivariate Gaussian distribution

is assumed, but other distributions can be used as well.

- $\theta = \{\pi, A, B\}$: Parameter vector fully specifying a HMM.

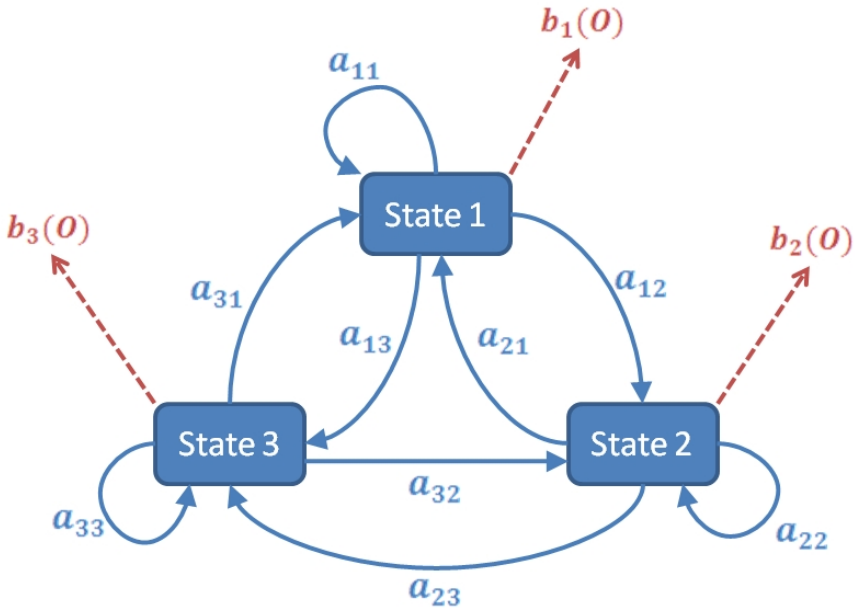

Figure 2.1.: Visualisation of the transition probabilities between three hidden states as well as the resulting probability functions (e.g. Gaussian). Every state can be reached from every other state with a specific probability. For some applications however, some transition probabilities a_{ij} might be zero. Note, that only the emissions O from the observation distributions $b_k(O)$ are observable, the states are not observable. Another representation of a HMM can be found in figure 2.2.

To give a better idea of what a hidden Markov model actually is algorithm 1 shows how an HMM observation sequence $\mathcal{O} = (O_1, O_2, ..., O_T)$ is drawn from a HMM, given a particular model θ. Note that the algorithm just generates the observation sequence $\mathcal{O}$ and not the state sequence $\mathcal{S}$ since the state sequence is hidden. Each observation could have been drawn by each hidden state with a certain probability. This leads us to another representation of Hidden Markov Models, where these models can also be viewed as a time-depending clustering task, where the Markov property holds.

Algorithm 1: Algorithm to simulate a HMM observation sequence $\mathcal{O} = (O_1, O_2, ..., O_T)$ given a particular model $\theta = \{\pi, A, B\}$.

Data: $\theta = \{\pi, A, B\}$
Result: Observation sequence $\mathcal{O} = (O_1, O_2, ..., O_T)$
Choose an initial state s_1 according to the initial state distribution π.
Set $t = 1$
for *Time* $t \in \{1, ..., T\}$ **do**
    Draw O_t from the probability distribution $b_{s_t}(\cdot)$
    Go to state s_{t+1} according to the transition probabilities $A_{s_t,\cdot}$
    Set $t = t + 1$

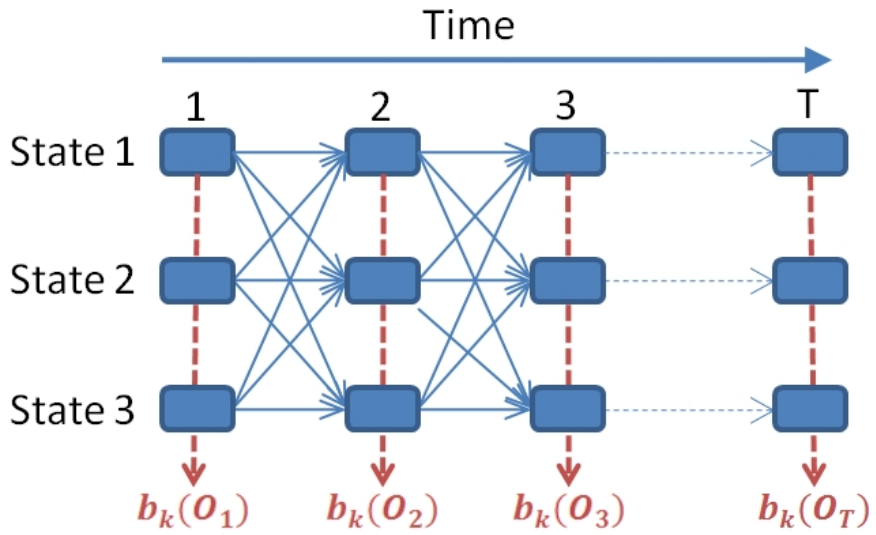

Figure 2.2.: Visualisation of the state transitions unfold over time as well as the resulting observation distributions $b_k(O_t)$ (e.g. Gaussian). Note that, since the states are not observable there are K^T possible hidden state sequences. Another representation of a HMM can be found in figure 2.1.

Given an observation sequence there are three fundamental problems to solve (Rabiner, 1989):

1. How do we compute the probability of the observation sequence given a HMM $\theta = \{\pi, A, B\}$?

2. How do we get the most likely hidden state sequence corresponding to the observation sequence given a HMM $\theta = \{\pi, A, B\}$?

3. How do we estimate the model parameters $\theta = \{\pi, A, B\}$ given one or several observation sequences?

Problem 1 can also be viewed as the problem how well a given model matches the observations. Problem 2 aims to detect the hidden part of the model, that is the state sequence supposed to be correct. However, problem 3 is the most important one as it allows us to determine optimal model parameters. Without these parameters we are not able to solve problems 1 and 2 since these problems expect a given HMM.

2.1.1. Probability of an Observation Sequence

In order to get solutions for the three problems we have to derive the probability of an observation sequence $\mathcal{O}$ given model parameters θ. The joint probability of an observation sequence and its corresponding state sequence is given by

$$P(\mathcal{O}, \mathcal{S}|\theta) = P(\mathcal{O}|\mathcal{S}, \theta) \cdot P(\mathcal{S}|\theta). \tag{2.1}$$

Hence, we can write the probability of $\mathcal{O}$ by summing 2.1 over all possible state sequences $\mathcal{S}$ as

$$P(\mathcal{O}|\theta) = \sum_{\mathcal{S}} P(\mathcal{O}|\mathcal{S}, \theta) \cdot P(\mathcal{S}|\theta),$$

in which the probability of an observation sequence given a specific state sequence $\mathcal{S}$ is

$$P(\mathcal{O}|\mathcal{S}, \theta) = \prod_{t=1}^{T} P(O_t|s_t, \theta) = \prod_{t=1}^{T} b_{s_t}(O_t)$$

and the probability of a fixed state sequence is given by

$$P(\mathcal{S}|\theta) = \pi_{s_1} \prod_{t=2}^{T} a_{s_{t-1}s_t}.$$

Altogether we can write

$$P(\mathcal{O}|\theta) = \sum_{\mathcal{S}} \pi_{s_1} \prod_{t=2}^{T} a_{s_{t-1}s_t} \prod_{t=1}^{T} b_{s_t}(O_t), \tag{2.2}$$

which is the probability of observing the data set $\mathcal{O} = (O_1, ..., O_T)$. In the basic HMM the observation probabilities $b_k(O_t)$ are usually assumed to follow a multivariate Gaussian mixture density function but can follow any other distribution as well.

The straightforward evaluation of equation 2.2 by enumerating and summing over all hidden state sequences is a computationally unfeasible problem, as there

are K possible states for each time t, resulting in K^T possible state sequences. However, this equation can also be solved using a so called forward-backward procedure (Liporace, 1982). Consider the joint probability of the incomplete observation sequence until a certain time t and state k at time t as being the forward term

$$\alpha_t(k) = P(O_1, ..., O_t; s_t = k|\theta)$$

and the backward term as

$$\beta_t(k) = P(O_{t+1}, ..., O_T|s_t = k, \theta).$$

That is the probability of the remaining observations given state k at time t. These expressions can be defined inductively as follows:

$$\begin{aligned}
\alpha_t(k) &= \sum_{j=1}^{K} \alpha_{t-1}(j) a_{jk} b_k(O_t) \\
&\quad \text{where } \alpha_1(i) = \pi_i b_i(O_1) \\
\beta_t(k) &= \sum_{j=1}^{K} \beta_{t+1}(j) a_{kj} b_j(O_{t+1}) \\
&\quad \text{where } \beta_T(i) = 1
\end{aligned}$$

The joint probability of the observation sequence $\mathcal{O}$ and state k at time t can be written as

$$\begin{aligned}
P(\mathcal{O}, s_t = k|\theta) &= P(O_1, ..., O_t; s_t = k|\theta) \cdot P(O_{t+1}, ..., O_T|O_1, ..., O_t; s_t = k, \theta) \\
&= P(O_1, ..., O_t; s_t = k|\theta) \cdot P(O_{t+1}, ..., O_T|s_t = k, \theta) \\
&= \alpha_t(k) \cdot \beta_t(k),
\end{aligned}$$

since the conditional probability of observing $O_{t+1}, ..., O_T$ given $s_t = k$ is independent of $O_1, ..., O_t$. Hence $\alpha_t(k)$ and $\beta_t(k)$ are independent and we can write the probability of a complete observation sequence, that is the solution of problem 1 as

$$\begin{aligned}
P(\mathcal{O}|\theta) &= \sum_{i=1}^{K} P(\mathcal{O}, s_t = i|\theta) \\
&= \sum_{i=1}^{K} \alpha_t(i) \cdot \beta_t(i) = \sum_{i=1}^{K} \alpha_T(i).
\end{aligned} \tag{2.3}$$

So by using the forward-backward terms we gain a more efficient way to evaluate equation 2.2 as it only requires on the order of $T \cdot K$ calculations. While the direct computation method requires on the order of K^T calculations. According

to Rabiner (1989) computing forward-backward variables leads to a saving of about 69 orders of magnitude, when computing the probability of a sequence of length $T = 100$ with $K = 5$ states. The whole procedure is illustrated in figure 2.3.

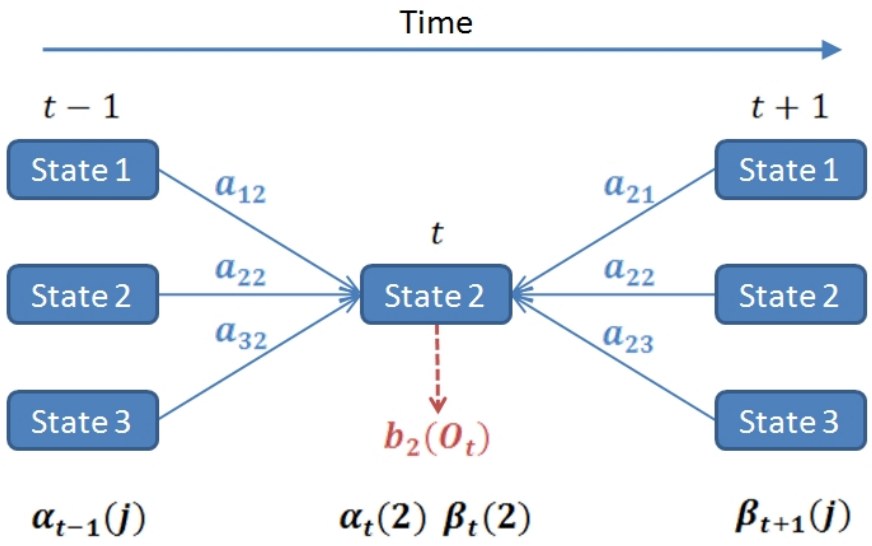

Figure 2.3.: Illustration of the required forward and backward terms in order to compute the forward-backward variables for state 2 at time t, that is $\alpha_t(2)$ and $\beta_t(2)$ respectively. Note the inductive definition of the forward-backward variables and how state 2 can be reached at time t from the 3 possible predecessors and successors respectively. The subscript j represents the states at time $t-1$ and $t+1$. The figure is based on Bishop (2006).

In order to make further computations easier let us introduce two auxiliary variables $\gamma_t(i)$ and $\xi_t(i,j)$. $\gamma_t(i) = P(s_t = i|\mathcal{O}, \theta)$ is the probability of being in state i at time t, given parameters θ as well as a sequence $\mathcal{O}$ and $\xi_t(i,j)$ is the probability of being in state i at time t and being in state j at time $t+1$, given parameters θ as well as a sequence $\mathcal{O}$. Hence, $\xi_t(i,j)$ can be interpreted as a time dependent transition probability $P(s_t = i, s_{t+1} = j|\mathcal{O}, \theta)$. These quantities can be expressed in terms of forward-backward variables as follows:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{K} \alpha_t(j)\beta_t(j)} \tag{2.4a}$$

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{K}\sum_{j=1}^{K} \alpha_t(i)a_{ij}} \tag{2.4b}$$

2.1.2. Viterbi Algorithm

Given an observation sequence and the corresponding model parameter θ we can calculate the most likely hidden state sequence $\mathcal{S}^*$ by applying the Viterbi algorithm, originally proposed by Forney (1973). Note that just looking for the most likely state for every time t could result in an invalid state sequence since some transitions could have zero probability ($a_{ij} = 0$ for some i and j). In contrast to computing the most probable state at each instant the Viterbi algorithm aims to maximize $P(\mathcal{S}|\mathcal{O}, \theta)$, that is the probability of the complete state sequence.

Let us denote

$$\delta_t(i) = \max_{s_1,...,s_{t-1}} P(s_1, ..., s_t = i; O_1, ..., O_t|\theta)$$

as the highest probability of a state sequence until time t that ends in state i. This expression can be calculated using a recursive definition

$$\delta_t(i) = \max_{1 \le j \le K} [\delta_{t-1}(j) a_{ji}]\, b_i(O_t).$$

In order to obtain the best state path the following procedure is applied (Rabiner, 1989):

1. Initialisation ($\forall i$):

$$\begin{aligned} \delta_1(i) &= \pi_i b_i(O_1) \\ \psi_1(i) &= 0 \end{aligned}$$

2. Recursion ($\forall j$):

$$\begin{aligned} \delta_t(j) &= \max_i [\delta_{t-1}(i) a_{ij}]\, b_j(O_t) \quad 2 \le t \le T \\ \psi_t(j) &= \operatorname*{argmax}_i [\delta_{t-1}(i) a_{ij}] \quad 2 \le t \le T \end{aligned}$$

3. Termination:

$$s_T^* = \operatorname*{argmax}_i [\delta_T(i)]$$

4. State sequence backtracking:

$$s_t^* = \psi_{t+1}(s_{t+1}^*) \quad t = T-1, T-2, ..., 1,$$

where $\psi_t(j)$ is an array that keeps track of the arguments which actually maximised $\delta_t(j)$ and $\mathcal{S}^* = (s_1^*, ..., s_T^*)$ is the resulting optimal state sequence. By applying the Viterbi algorithm we are able to solve problem 2, mentioned in section 2.1.

2.1.3. Baum-Welch Algorithm

The remaining and by far the most challenging problem is finding model parameters $\theta = \{\pi, A, B\}$ that maximise the probability of a given observation sequence. In other words, we want to obtain model parameters that describe how the given observation sequence comes about best. That means, we want to find a maximum-log-likelihood solution of the HMM parameters $\theta = \{\pi, A, B\}$ given a particular observation sequence $\mathcal{O}$, hence

$$\underset{\theta}{\operatorname{argmax}} \log\left(\mathcal{L}(\theta|\mathcal{O}, \mathcal{S})\right)$$

where $\mathcal{L}(\theta|\mathcal{O}, \mathcal{S})$ is given by equation 2.2.

Since there are unobserved latent variables involved this problem can not be solved analytically. However we can solve this problem using the expectation-maximisation (EM) algorithm (Dempster et al., 1977), that is also known as the Baum-Welch (Baum et al., 1970) algorithm when dealing with Hidden Markov models. The EM algorithm is an iterative method which alternates between an expectation (E)-step and a maximisation (M)-step. According to Dempster et al. (1977) the expectation of the log-likelihood using a current estimate of the parameter-vector θ

$$Q(\theta, \theta^{\text{old}}) = \mathbb{E}\left[\log\left(P(\mathcal{O}, \mathcal{S}|\theta)\right)|\mathcal{O}, \theta^{\text{old}}\right]$$

is calculated in the E-step. In the M-step the parameter-vector θ is updated in the way that the expected log-likelihood of the E-step is maximised

$$\theta^{\text{new}} = \underset{\theta}{\operatorname{argmax}}\ Q(\theta, \theta^{\text{old}}).$$

This procedure is repeated iteratively until a local maximum is reached.

Let the state sequence $\mathcal{S} = (s_1, ..., s_T)$ be the unobserved data and the observation sequence $\mathcal{O} = (O_1, ..., O_T)$ be the observed data. In other words, the incomplete data is given by the observations $\mathcal{O}$ and the complete data is given by the observations $\mathcal{O}$ including its corresponding state sequences $\mathcal{S}$. Consequently the complete data log-likelihood is given by $\log\left(\mathcal{L}(\theta|\mathcal{O}, \mathcal{S})\right)$. Therefore the Q function is given by

$$Q(\theta, \theta^{\text{old}}) = \sum_{\mathcal{S}} P(\mathcal{S}, \mathcal{O}|\theta^{\text{old}}) \cdot \log\left(\mathcal{L}(\theta|\mathcal{O}, \mathcal{S})\right). \tag{2.5}$$

By inserting $\mathcal{L}(\theta|\mathcal{O},\mathcal{S})$ into 2.5 and separating the parameters we obtain

$$\begin{aligned} Q(\theta,\theta^{\text{old}}) &= \sum_{\mathcal{S}} P(\mathcal{S},\mathcal{O}|\theta^{\text{old}}) \cdot \log\left(\pi_{s_1}\prod_{t=2}^{T} a_{s_{t-1}s_t}\prod_{l=1}^{T} b_{s_l}(O_l)\right) \\ &= \sum_{\mathcal{S}} P(\mathcal{S},\mathcal{O}|\theta^{\text{old}}) \cdot \log\left(\pi_{s_1}\right) \\ &\quad + \sum_{\mathcal{S}} P(\mathcal{S},\mathcal{O}|\theta^{\text{old}}) \cdot \left(\sum_{t=2}^{T} \log\left(a_{s_{t-1}s_t}\right)\right) \\ &\quad + \sum_{\mathcal{S}} P(\mathcal{S},\mathcal{O}|\theta^{\text{old}}) \cdot \left(\sum_{l=1}^{T} \log\left(b_{s_l}(O_l)\right)\right), \end{aligned}$$

a representation of the Q-function where each term can be optimised separately, since the parameters are independent of each other. In addition, each summand can be greatly simplified to

$$Q(\theta,\theta^{\text{old}}) = \sum_{i=1}^{K} P(s_1 = i,\mathcal{O}|\theta^{\text{old}}) \cdot \log\left(\pi_i\right) \tag{2.6a}$$

$$+ \sum_{t=2}^{T}\sum_{i=1}^{K}\sum_{j=1}^{K} P(s_{t-1} = i, s_t = j,\mathcal{O}|\theta^{\text{old}}) \cdot \log\left(a_{ij}\right) \tag{2.6b}$$

$$+ \sum_{t=1}^{T}\sum_{i=1}^{K} P(s_l = i,\mathcal{O}|\theta^{\text{old}}) \cdot \log\left(b_i(O_t)\right). \tag{2.6c}$$

The simplification is based on the fact that we sum over all state sequences $\mathcal{S}$, but in each summand we only need the state of a particular time, that is s_1 in 2.6a, s_{t-1} and s_t in 2.6b and s_l in 2.6c.

Through deriving the Q-function we gained a powerful estimation method for the computation of the model parameters π, A and B based on the EM algorithm. Now, in order to derive parameter estimates, one has to maximize equation 2.6 with respect to each parameter of interest separately. A complete derivation of the parameter update formulas can be found in section 2.2.2.

2.1.4. Limitations

Although HMMs are well tested models that "when applied properly, work very well in practice for several important applications" (Rabiner, 1989) a number of limitations exist. One of which is that the results strongly rely on a proper initialisation of the model parameters. Simulations (Rabiner, 1989) have shown that uniform initial values of π and A are suitable, but adequate initial values for the estimation of the B parameters are indispensable. Furthermore, the number

of hidden states K has to be known in advance. An inadequate number of states would result in wrong cluster representations. However, there are some methods to overcome those problems, for example using standard clustering results as initial values for B as well as the Akaike information criterion (AIC) for the identification of an appropriate state number.

There are efficient parameter update estimators for HMMs with discrete and Gaussian emission values. Thus, in practice, when dealing with continuous data a multivariate Gaussian distribution is assumed most of the time. That, however, is the strongest limitation. In many real data sets the observations do not even approximately follow a normal distribution and hence assuming normal distributed emissions could result in "useless" model parameters. Of course one could replace the Gaussian distribution by a distribution adapted to the situation of interest. But in this case, it is not certain whether the distribution can be included in the Baum-Welch algorithm and hence whether efficient parameter estimators exist. In addition, there are situations where no known distribution fits the data in an reasonable extent. Consider the situation pictured in figure 2.4 where the observations are arranged in non-spherical clusters. In addition the situation in which a cluster center can not be defined in a proper way may arise. This could be the case when the observation cannot be represented by a p-dimensional vector, thus has no location in space, in other words when the observation sample is obtained from a non-Euclidean metric space. It should be clear that a normal distribution is not appropriate in this case as well. Moreover, datasets where the observations belong to more than one state (cluster) at a time with a specific probability cannot be handled with a normal hidden Markov model. Consider the case where no explicit boundary between the clusters can be determined. In this case soft cluster assignments as in fuzzy k-Means (Dunn, 1973) would be more appropriate.

Given these limitations one has to think about how to adjust the well established HMM framework in order to get rid of some restrictions.

2.2. Generalised HMM (GHMM)

To overcome some of the mentioned limitations of the standard hidden Markov model we suggest to use a generalised emission density that is able to deal with samples obtained from non-Euclidean metric space. In the following sections we are going to introduce a density that is only dependent on the pairwise distances of all observations and no longer on a center of mass nor a variance term. Furthermore the HMM parameter estimates for all included parameters are derived.

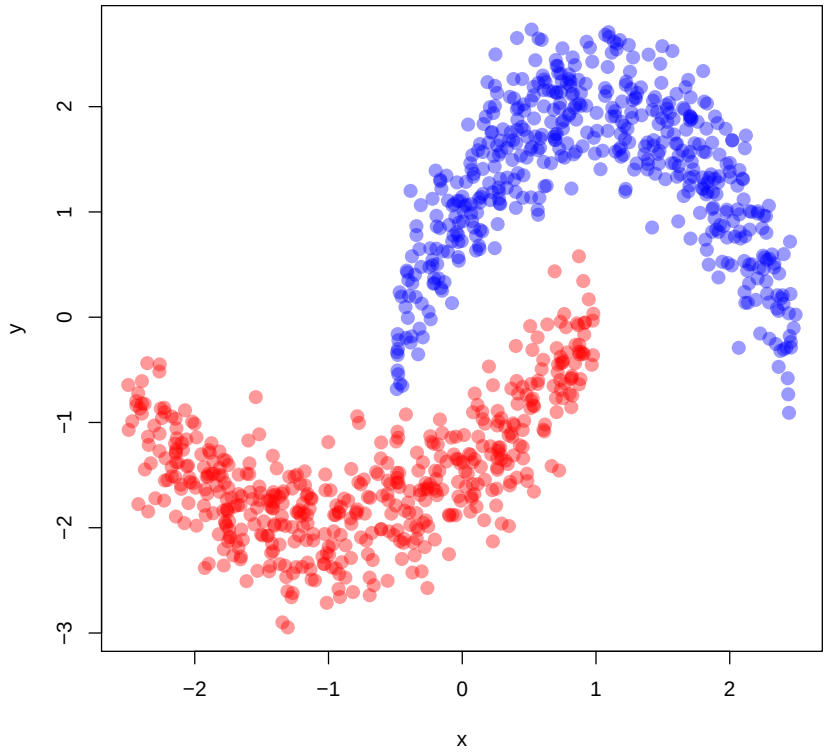

Figure 2.4.: Dataset with non-spherical clusters where the application of Gaussian emission distributions has limited results. The figure was generated using R (R Development Core Team, 2009).

2.2.1. Generalised Emission Density

The origin of our generalised emission density is the normal distribution density with zero covariance and equal variance in all dimensions:

$$f_{\mathcal{O}}(O_i) = \frac{1}{(2\pi\sigma_k^2)^{\frac{p}{2}}} \exp\left(-\frac{1}{2\sigma_k^2}(O_i - \mu_k)^2\right) \tag{2.7}$$

where p is the dimension of observation O_i and k a given cluster. This probability density function however is not applicable when working with data observed from non-Euclidean metric space. In order to overcome this weakness we suggest to replace the variance term σ_k^2 and the center μ_k in equation 2.7 with expressions only depending on the pairwise distances of all observations.

The basis of our improvements is formed by the fact that observations can be

assigned to clusters $k = \{1, ..., K\}$ in a fuzzy way. Instead of a hard assignment of an observation $i = \{1, ..., T\}$ to a cluster k we use soft assignments, referred to as responsibilities $r_{ik} \in [0, 1]\ \forall i, k$ in the further work. That means, each observation i could have been emitted from each cluster k with a certain probability r_{ik}. We introduce a matrix

$$R = \{r_{ik} | i = 1, ..., T; k = 1, ..., K\},\ r_{ik} \in [0, 1]\ \forall i, k$$

where each row sum equals one, that is $\sum_{k=1}^{K} r_{ik} = 1\ \forall i$. Hence R contains the probability of each sample i belonging to a certain cluster k. In the following sections R_k will refer to the sum over all observations given a specific cluster k, thus $R_k = \sum_{i=1}^{T} r_{ik}$.

Now, in order to derive the generalised emission distribution we can replace the variance σ_k^2 by its empirical estimate

$$\hat{\sigma}_k^2 = \frac{1}{R_k} \sum_{i=1}^{T} (O_i - \mu_k)^2$$

that can be expressed in terms of responsibilities r_{ik} and pairwise distances of all observations according to Lemma 1.

Lemma 1.

$$\sigma_k^2 = \frac{1}{2R_k^2} \sum_{i=1}^{T} \sum_{j=1}^{T} r_{ik} r_{jk} ||O_i - O_j||^2$$

A proof for Lemma 1 can be found in Müller (2011). By plugging Lemma 1 into the normal probability distribution function 2.7 we get rid of the variance term σ_k^2. In a second step we aim to remove the cluster center μ_k. According to Lemma 2 the distance $(O_i - \mu_k)^2$ in 2.7 can also be expressed in terms of responsibilities r_{ik} and pairwise distances of all observations.

Lemma 2.

$$(O_i - \mu_k)^2 = \frac{1}{R_k} \left(\sum_{j=1}^{T} r_{jk} ||O_j - O_i||^2 - \frac{1}{2R_k} \sum_{j=1}^{T} \sum_{l=1}^{T} r_{jk} r_{lk} ||O_j - O_l||^2 \right)$$

Lemma 2 states that the distance of a point O_i to the corresponding center of mass μ_k can be written as the sum of distances of point O_i to all other points plus the sum of all pairwise distances. Again, a proof of this Lemma can be found in Müller (2011). Inserting Lemma 1 and 2 into the normal distribution density function 2.7 yields a new empirical probability density of observing O_i in state

k, that is $b_k(O_i)$ in terms of hidden markov models.

$$
\begin{aligned}
b_k(O_i) &= P(O_i|k) = \frac{1}{(2\pi\sigma_k^2)^{\frac{p}{2}}} \exp\left(-\frac{1}{2\sigma_k^2}(O_i-\mu_k)^2\right) \\
&= \left(\frac{2\pi}{2R_k^2}\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lk}r_{jk}d_{lj}\right)^{-\frac{p}{2}} \exp\left[-\frac{1}{2}\left(\frac{1}{2R_k^2}\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lk}r_{jk}d_{lj}\right)^{-1}\right. \\
&\qquad \left.\cdot\frac{1}{R_k}\left(\sum_{j=1}^{T} r_{jk}d_{ji} - \frac{1}{2R_k}\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lk}r_{jk}d_{lj}\right)\right] \\
&= \left(\frac{\pi}{R_k^2}\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lk}r_{jk}d_{lj}\right)^{-\frac{p}{2}} \exp\left[-\frac{1}{2R_k}\left(\frac{\sum_{j=1}^{T} r_{jk}d_{ji}}{\frac{1}{2R_k^2}\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lk}r_{jk}d_{lj}}\right.\right. \\
&\qquad \left.\left. - \frac{\frac{1}{2R_k}\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lk}r_{jk}d_{lj}}{\frac{1}{2R_k^2}\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lk}r_{jk}d_{lj}}\right)\right] \\
&= \left(\frac{\pi}{R_k^2}\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lk}r_{jk}d_{lj}\right)^{-\frac{p}{2}} \exp\left[-R_k\frac{\sum_{j=1}^{T} r_{jk}d_{ji}}{\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lk}r_{jk}d_{lj}} + \frac{1}{2}\right] \quad (2.8)
\end{aligned}
$$

The complete derivation of the generalised emission distribution can be found above, where d_{ij} is some measure for the distance between observation O_i and observation O_j, e.g. the euclidean distance $||O_i - O_j||^2$. Note, that this function is only expressed in terms of pairwise distances between all observations and its responsibilities belonging to a certain cluster k and does no longer depend on a cluster center or a cluster variance. A graphical comparison between the derived generalised emission distribution and the normal distribution can be found in figure 4.1. The plots illustrate that the two distributions match each other perfectly when using the squared Euclidean distance as distance measure d_{ij}. However, the advantage in comparison to the original HMM is that we are able to use different distance measures which are well adjusted for the particular data situation of interest. An example of an alternative distance measure could be the Manhattan norm. One could also think about using a weighted distance measure, where the distances are weighted according to some prior knowledge of the data.

2.2.2. Derivation of the Parameter Estimations

The parameter estimates of $\theta = \{\pi, A, R\}$ can be derived according to the Baum-Welch algorithm described in section 2.1.3. As already mentioned we have to maximize equation 2.6 with respect to each parameter of interest separately in order to get updates of the parameters.

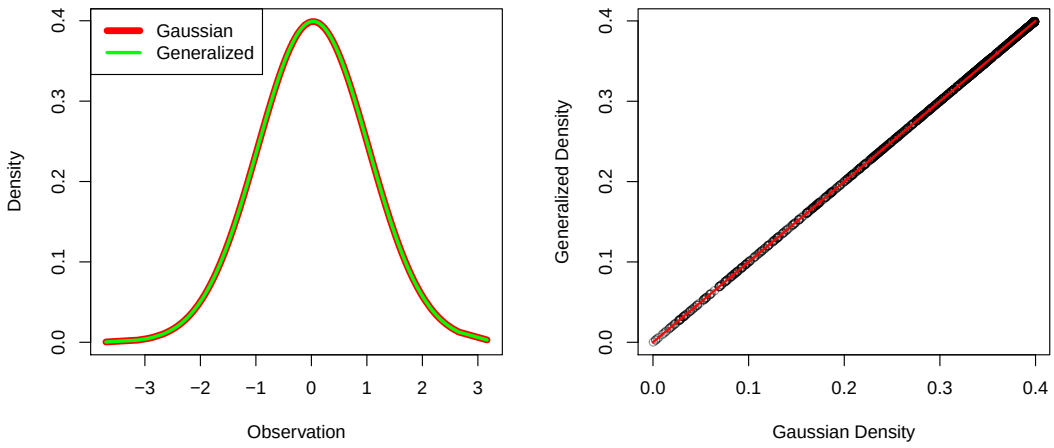

Figure 2.5.: A comparison of an empirical gaussian distribution and the corresponding generalised distribution (distance d_{ij} is the squared euclidean distance) introduced in section 2.2.1 based on $n = 1000$ samples from a standard normal density. The plot on the left shows two perfectly matched density functions and the plot on the right shows a plot of the generalised density against the empirical normal density that lies on the bisecting line. The figure was generated using R (R Development Core Team, 2009).

2.2.2.1. Initial State Probabilities

Taking the partial derivative of $Q(\theta, \theta^{\text{old}})$ with respect to π_i under the constraint $\sum_{l=1}^{K} \pi_l = 1$ leads to

$$\begin{aligned} 0 &= \frac{\partial}{\partial \pi_i} \left[Q(\theta, \theta^{\text{old}}) - \lambda \left(\sum_{l=1}^{K} \pi_l - 1 \right) \right] \\ &= \frac{1}{\pi_i} \cdot P(s_1 = i, \mathcal{O} | \theta^{\text{old}}) - \lambda \end{aligned} \tag{2.9}$$

where λ is the Lagrange multiplier (Luenberger and Ye, 2008). Multiplying 2.9 by π_i and summing over i leads to

$$\lambda = P(\mathcal{O} | \theta^{\text{old}}) \tag{2.10}$$

By inserting equation 2.10 into equation 2.9 we obtain

$$\begin{aligned} 0 &= \frac{1}{\pi_i} \cdot P(s_1 = i, \mathcal{O}|\theta^{\text{old}}) - P(\mathcal{O}|\theta^{\text{old}}) \\ \pi_i &= \frac{P(s_1 = i, \mathcal{O}|\theta^{\text{old}})}{P(\mathcal{O}|\theta^{\text{old}})}, \end{aligned} \tag{2.11}$$

the estimated probability for state i at time 1. Note, that 2.11 can also be expressed in terms of the forward-backward variables described in section 2.1, as it exactly meets the definition of the auxiliary variable $\gamma_t(i)$ given in equation 2.4a:

$$\pi_i = \frac{\alpha_1(i)\beta_1(i)}{\sum_{j=1}^{K} \alpha_1(j)\beta_1(j)} = \gamma_1(i) \tag{2.12}$$

2.2.2.2. Transition Probabilities

In order to obtain the transition probabilties a_{ij} we take the partial derivative of $Q(\theta, \theta^{\text{old}})$ under the constraint $\sum_{l=1}^{K} a_{il} = 1$:

$$\begin{aligned} 0 &= \frac{\partial}{\partial a_{ij}} \left[Q(\theta, \theta^{\text{old}}) - \lambda \left(\sum_{l=1}^{K} a_{il} - 1 \right) \right] \\ &= \frac{1}{a_{ij}} \sum_{t=2}^{T} P(s_{t-1} = i, s_t = j, \mathcal{O}|\theta^{\text{old}}) - \lambda \end{aligned} \tag{2.13}$$

Multiplying 2.13 by a_{ij} and summing over j yields to:

$$\lambda = \sum_{t=2}^{T} P(s_{t-1} = i, \mathcal{O}|\theta^{\text{old}}) \tag{2.14}$$

Again, substituting 2.14 into 2.13 yields

$$\begin{aligned} 0 &= \frac{1}{a_{ij}} \sum_{t=2}^{T} P(s_{t-1} = i, s_t = j, \mathcal{O}|\theta^{\text{old}}) - \sum_{t=2}^{T} P(s_{t-1} = i, \mathcal{O}|\theta^{\text{old}}) \\ a_{ij} &= \frac{\sum_{t=2}^{T} P(s_{t-1} = i, s_t = j, \mathcal{O}|\theta^{\text{old}})}{\sum_{t=2}^{T} P(s_{t-1} = i, \mathcal{O}|\theta^{\text{old}})} \\ &= \frac{\sum_{t=2}^{T} \alpha_{t-1}(i) a_{ij} b_j(O_t) \beta_t(j)}{\sum_{t=2}^{T} \alpha_{t-1}(i)\beta_{t-1}(i)} \\ &= \frac{\sum_{t=2}^{T} \xi_{t-1}(i,j)}{\sum_{t=2}^{T} \gamma_{t-1}(i)} \end{aligned}$$

which can again be expressed in terms of the auxiliary variables $\gamma_t(i)$ and $\xi_t(i)$. Note that the estimator of a_{ij} is the quotient of the expected number of transitions from state i to state j and the expected number of transitions from state i (Rabiner, 1989).

2.2.2.3. Responsibilities

Computing an update for the responsibilites R is more complicated since deriving a closed form solution of the constrained global non-linear optimisation problem is not possible. So one has to think about an adequate optimisation strategy to find the maximum

$$R^{\max} = \underset{R}{\operatorname{argmax}}\, Q(\{\pi^{\text{old}}, A^{\text{old}}, R\}, \{\pi^{\text{old}}, A^{\text{old}}, R^{\text{old}}\})$$

under the constraints

$$\sum_{k=1}^{K} r_{ik} = 1 \ \forall i$$

and

$$r_{ik} \in [0, 1] \ \forall i, k.$$

This task can be solved using analytical optimisation methods such as gradient ascent. This iterative method aims to optimise a function $f(x)$ by taking steps τ proportional to the direction of f at a point z, that is the gradient $\nabla f = f'(z)$. A new point $z^{\text{new}} = z + \tau \cdot f'(z)$ is determined at each iteration until the algorithm reaches a local/global maximum. Disadvantages of the algorithm are that it can not overcome local maxima plus the results crucially depend on the step size τ. Too small steps will result in a very long computing time until convergence and too large values of τ may lead to missed maxima. One way to overcome the problem associated with the step size is to combine the gradient ascent with a line search that finds an optimal step size at every iteration. However, the original algorithm does not take the linear and box constraints into account, that is why we have to adjust the method such that it can handle the constraints. The problem associated with the constraints is illustrated in figure 2.6A, where an optimisation of $r_{t\cdot}$ proportional to ∇f might violate the linear and box constraints.

In order to modify the gradient method to suit our needs we first have to derive the partial derivative of $Q(\theta, \theta^{\text{old}})$ with respect to r_{ab}. In doing so, the closed form

solution of the gradient is given by

$$\begin{aligned}\frac{\partial}{\partial r_{ab}}Q(\theta,\theta^{\text{old}}) =& \textcircled{1} - \textcircled{2} \\ =& \sum_{t=1}^{T} P(s_t = b, \mathcal{O}|\theta^{\text{old}}) \left[\frac{p}{R_b} - \frac{p \cdot \sum_{l=1}^{T} r_{lb} d_{la} - \sum_{j=1}^{T} r_{jb} d_{jt}}{\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lb} r_{jb} d_{lj}} \right. \\ & \left. - R_b \frac{d_{at} \cdot \sum_{l=1}^{T}\sum_{j=1}^{T} r_{lb} r_{jb} d_{lj} - \sum_{j=1}^{T} r_{jb} d_{jt} \cdot 2 \sum_{l=1}^{T} r_{lb} d_{la}}{(\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lb} r_{jb} d_{lj})^2} \right]\end{aligned}$$

where expressions (1) and (2) are derived according to pages 21 and 22. For each observation $a = 1, ..., T$ we have a set of dependant responsibilities $r_{a\cdot} = \{r_{a1}, r_{a2}, ..., r_{aK}\}$ that have to be updated all at once in each iteration. Thus the direction information for an responsibility update of a single observation a is given by

$$\nabla\mathcal{L}(r_{a\cdot}) = \left(\frac{\partial}{\partial r_{a1}}Q(\theta,\theta^{\text{old}}), \frac{\partial}{\partial r_{a2}}Q(\theta,\theta^{\text{old}}), ..., \frac{\partial}{\partial r_{aK}}Q(\theta,\theta^{\text{old}}) \right). \tag{2.15}$$

As already mentioned updating responsibilities $r_{a\cdot}$ in direction of $\nabla\mathcal{L}(r_{a\cdot})$ might violate the linear constraint $\sum_{k=1}^{K} r_{ak} = 1$. We deal with this problem by calculating the projection of the direction information 2.15 onto the line of allowed solutions, that is the simplex spanned by the corners of the box of feasible solutions $[0,1]^K$ (see figure 2.6). However the box constraint $r_{ab} \in [0,1]$ remains an unsolved problem since an update of the responsibilities might point out of the box of valid solutions, especially when r_{ab} is already close to the borders zero or one.

To overcome both constraints we proceed as follows: First we are looking for responsibilities that are sufficiently far away from the box corners, hence that can be increased or decreased without the risk of leaving the box. We do this by creating two non disjoint sets

$$\begin{aligned} C_1 &= \{k : r_{ak} < 1 - \epsilon\},\ \forall k = 1, ..., K \\ C_2 &= \{k : r_{ak} > \epsilon\},\ \forall k = 1, ..., K \end{aligned}$$

where C_1 includes states whose responsibilities may be increased and C_2 whose responsibilities may be decreased. The threshold ϵ is set to 0.005 here. In a second step we discretize the direction information by choosing the responsibility with the largest partial derivative from the set of states which can be increased

$$m = \underset{k}{\text{argmax}}\, \nabla\mathcal{L}(r_{ak}),\ \forall k \in C_1$$

as well as the responsibility with the smallest partial derivative from the set of

states that can be decreased

$$g = \underset{k}{\operatorname{argmin}} \nabla \mathcal{L}(r_{ak}),\ \forall k \in C_2.$$

Thus we get r_{am} that should be increased and r_{ag} that should be decreased. Note that the sets C_1 and C_2 are not empty and that m and g are never equal. In order to take the linear constraint into account we introduce a weight vector w which is set to 1 at position m, to -1 at position g and to zero in the remaining positions. More formally, the weight vector is given by $w = (e_m - e_g)$, where e_i is the $i-$th unit vector. The responsibilites $r_{ak}, k \notin \{m, g\}$ are kept constant. By using the weights w we are able to update the responsibilites $r_{a\cdot}$ for each observation $a = 1, ...T$ by applying the following update step:

$$r_{a\cdot}^{\text{new}} = r_{a\cdot} + \tau \cdot w$$

Note that the linear constraint is met, since we go $-\tau$ in direction g and τ in direction m and hence the sum over all responsibilities for a given observation a remains the same.

$$
\begin{aligned}
\frac{\partial}{\partial r_{ab}} Q(\theta, \theta^{\text{old}}) =& \frac{\partial}{\partial r_{ab}} \sum_{k=1}^{K} \sum_{t=1}^{T} P(s_t = k, \mathcal{O}|\theta^{\text{old}}) \cdot \log\left(b_k(O_t)\right) \\
=& \frac{\partial}{\partial r_{ab}} \sum_{k=1}^{K} \sum_{t=1}^{T} P(s_t = k, \mathcal{O}|\theta^{\text{old}}) \cdot \left[-\frac{p}{2} \log\left(\frac{\pi}{R_k^2} \sum_{l=1}^{T} \sum_{j=1}^{T} r_{lk} r_{jk} d_{lj} \right) - R_k \frac{\sum_{j=1}^{T} r_{jk} d_{jt}}{\sum_{l=1}^{T} \sum_{j=1}^{T} r_{lk} r_{jk} d_{lj}} + \frac{1}{2} \right] \\
=& \underbrace{\frac{\partial}{\partial r_{ab}} \sum_{k=1}^{K} \sum_{t=1}^{T} -P(s_t = k, \mathcal{O}|\theta^{\text{old}}) \frac{p}{2} \log\left(\frac{\pi}{R_k^2} \sum_{l=1}^{T} \sum_{j=1}^{T} r_{lk} r_{jk} d_{lj} \right)}_{\text{①}} - \frac{\partial}{\partial r_{ab}} \sum_{k=1}^{K} \sum_{t=1}^{T} P(s_t = k, \mathcal{O}|\theta^{\text{old}}) R_k \frac{\sum_{j=1}^{T} r_{jk} d_{jt}}{\sum_{l=1}^{T} \sum_{j=1}^{T} r_{lk} r_{jk} d_{lj}} \\
=& \underbrace{\frac{\partial}{\partial r_{ab}} \sum_{t=1}^{T} -P(s_t = b, \mathcal{O}|\theta^{\text{old}}) \frac{p}{2} \log\left(\frac{\pi}{R_b^2} \sum_{l=1}^{T} \sum_{j=1}^{T} r_{lb} r_{jb} d_{lj} \right)}_{\text{①}} - \underbrace{\frac{\partial}{\partial r_{ab}} \sum_{t=1}^{T} P(s_t = b, \mathcal{O}|\theta^{\text{old}}) R_b \frac{\sum_{j=1}^{T} r_{jb} d_{jt}}{\sum_{l=1}^{T} \sum_{j=1}^{T} r_{lb} r_{jb} d_{lj}}}_{\text{②}}
\end{aligned}
$$

$$
\begin{aligned}
\textcircled{1} &= -\frac{p}{2}\sum_{t=1}^{T} P(s_t = b, \mathcal{O}|\theta^{\text{old}}) \cdot \frac{\partial}{\partial r_{ab}} \log\left(\frac{\pi}{R_b^2}\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lb} r_{jb} d_{lj}\right) \\
&= -\frac{p}{2}\sum_{t=1}^{T} P(s_t = b, \mathcal{O}|\theta^{\text{old}}) \cdot \left[\left(\frac{\pi}{R_b^2}\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lb} r_{jb} d_{lj}\right)^{-1} \cdot \underbrace{\frac{\partial}{\partial r_{ab}}\left(\frac{\pi}{R_b^2}\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lb} r_{jb} d_{lj}\right)}_{\begin{cases} 2r_{ab}d_{aa} = 0 & \text{for } l = a \;\&\; j = a \\ \sum_{j=1}^{T} r_{jb} d_{aj} & \text{for } l = a \;\&\; j \neq a \\ \sum_{l=1}^{T} r_{lb} d_{la} & \text{for } l \neq a \;\&\; j = a \\ 0 & \text{otherwise} \end{cases}}\right] \\
&\qquad -\frac{2\pi}{R_b^3}\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lb} r_{jb} d_{lj} + \frac{\pi}{R_b^2}\begin{cases} \sum_{j=1}^{T} r_{jb} d_{aj} & \text{for } l = a \;\&\; j \neq a \\ \sum_{l=1}^{T} r_{lb} d_{la} & \text{for } l \neq a \;\&\; j = a \\ 0 & \text{otherwise} \end{cases} \\
&= -\frac{p}{2}\sum_{t=1}^{T} P(s_t = b, \mathcal{O}|\theta^{\text{old}}) \cdot \left[\left(\frac{\pi}{R_b^2}\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lb} r_{jb} d_{lj}\right)^{-1} \cdot \left(-\frac{2\pi}{R_b^3}\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lb} r_{jb} d_{lj} + \frac{2\pi}{R_b^2}\sum_{l=1}^{T} r_{lb} d_{la}\right)\right] \\
&= \sum_{t=1}^{T} P(s_t = b, \mathcal{O}|\theta^{\text{old}}) \cdot \left[\frac{p}{R_b} - p \cdot \frac{\sum_{l=1}^{T} r_{lb} d_{la}}{\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lb} r_{jb} d_{lj}}\right] \\
\textcircled{2} &= \sum_{t=1}^{T} P(s_t = b, \mathcal{O}|\theta^{\text{old}}) \cdot \frac{\partial}{\partial r_{ab}}\left[R_b \frac{\sum_{j=1}^{T} r_{jb} d_{jt}}{\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lb} r_{jb} d_{lj}}\right] \\
&= \sum_{t=1}^{T} P(s_t = b, \mathcal{O}|\theta^{\text{old}}) \cdot \left[\frac{\sum_{j=1}^{T} r_{jb} d_{jt}}{\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lb} r_{jb} d_{lj}} + R_b \cdot \frac{\partial}{\partial r_{ab}}\left(\frac{\sum_{j=1}^{T} r_{jb} d_{jt}}{\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lb} r_{jb} d_{lj}}\right)\right] \\
&= \sum_{t=1}^{T} P(s_t = b, \mathcal{O}|\theta^{\text{old}}) \cdot \left[\frac{\sum_{j=1}^{T} r_{jb} d_{jt}}{\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lb} r_{jb} d_{lj}} + R_b \frac{d_{at} \cdot \sum_{l=1}^{T}\sum_{j=1}^{T} r_{lb} r_{jb} d_{lj} - \sum_{j=1}^{T} r_{jb} d_{jt} \cdot 2\sum_{l=1}^{T} r_{lb} d_{la}}{(\sum_{l=1}^{T}\sum_{j=1}^{T} r_{lb} r_{jb} d_{lj})^2}\right]
\end{aligned}
$$

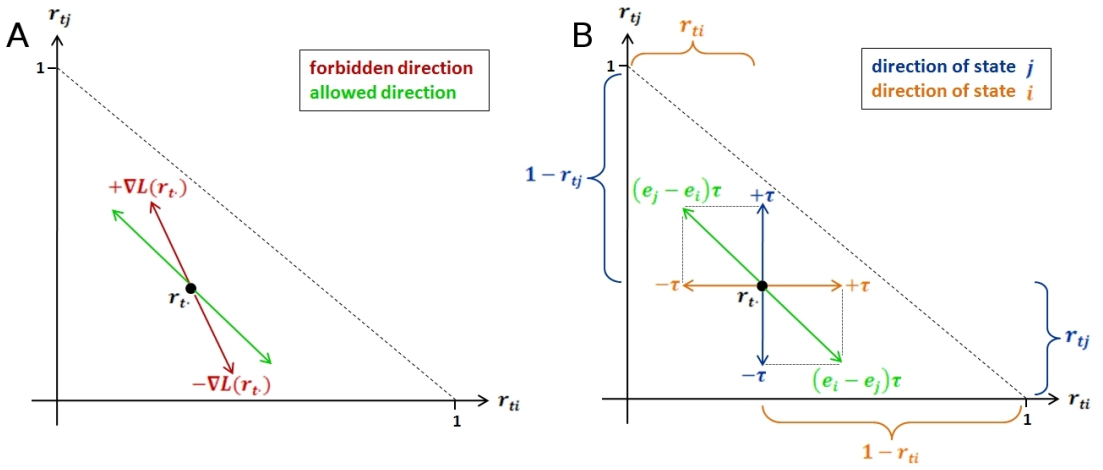

Figure 2.6.: Update step for the responsibilities. For the sake of simplicity, only the responsibilities r_{tj} and r_{ti} are shown. Using the standard update formula of the gradient ascent method might violate the linear- and box constraints (A). To solve this problem $\nabla\mathcal{L}(r_{t\cdot})$ has to be projected onto the line of allowed directions, which are located on the parallel to the simplex (A). This projection is applied by taking step τ in direction of the largest partial derivative $\text{argmax}_k(\nabla\mathcal{L}(r_{ak}))$ and $-\tau$ in direction of the smallest partial derivative $\text{argmin}_k(\nabla\mathcal{L}(r_{ak}))$ (B). The weight vector w is given by $(e_i - e_j)$, respectively $(e_j - e_i)$, where e_i is the $i-$th unit vector (B). The natural interval for the step size τ is depending on the current update direction and is given by $[0, \min(1 - r_{tj}, r_{ti})]$, respectively $[0, \min(1 - r_{ti}, r_{tj})]$ (B).

As mentioned in the beginning of this section we perform a line search in order to determine an optimal step size τ. The line search algorithm detects an optimal τ for each observation a such that $Q(\{\pi^{\text{old}}, A^{\text{old}}, R\}, \{\pi^{\text{old}}, A^{\text{old}}, R^{\text{old}}\})$ is maximised. There are some restrictions on the feasible values for τ, whereas the minimum is set to 0 and the maximum to $\min(1 - r_{am}, r_{ag}, 0.1)$. The reason for the lower bound is that going in the negative direction is already controlled by the weight vector w and should not be changed by the step size. The maximum is because $0 \leq r_{ak} + \tau \cdot w_k \leq 1$ must hold for all k. As only the responsibilities for states m and g change it is sufficient to take

$$\begin{aligned} 0 \leq r_{am} + \tau \leq 1 \quad &\Rightarrow \quad r_{am} + \tau \leq 1 \\ 0 \leq r_{ag} - \tau \leq 1 \quad &\Rightarrow \quad 0 \leq r_{ag} - \tau \end{aligned}$$

into account. It follows that $\tau \leq 1 - r_{am}$ and $\tau \leq r_{ag}$, which is valid for all $\tau_{\max} \leq \min(1 - r_{am}, r_{ag})$. Besides, an artificial restriction of the maximal step size of 0.1 is included in order to ensure that the responsibilities do not change to a

large extent in a single step. A graphical illustration of the proposed responsibility update step and the corresponding natural step size boundary is given in figure 2.6.

In each iteration of the EM algorithm the responsibilities $r_{a\cdot}$ for all observations O_a, $a = 1, ..., T$ are updated according to the procedure described above. Note that the convergence of the responsibilities is reached through the EM iterations and not within each iteration since we just take one step at a time. A summary of the proposed optimisation method is given in Algorithm 2.

Algorithm 2: Algorithm to obtain new responsibility matrices R^{new}

Data: For each sequence $s = 1, ..., S$: Samples $\mathcal{O}_s = \{O_1, ..., O_T\} \subset \mathbb{R}^p$ and Responsibility matrix $R_s \in [0,1]^{T \times K}$. Transition matrix $A \in [0,1]^{K \times K}$, start probabilites $\pi \in [0,1]^{K \times 1}$

Result: Updated responsibility matrices $R_s^{\text{new}} \in [0,1]^{T \times K}$

for *Sequence* $s \in \{1, ..., S\}$ **do**
  for *Time* $t \in \{1, ..., T\}$ **do**
    $dt = \nabla \mathcal{L}(r_{t\cdot}^{(s)})$
    $let\_up = \text{which}(r_{t\cdot}^{(s)} < 1 - \epsilon)$
    $let\_down = \text{which}(r_{t\cdot}^{(s)} > \epsilon)$
    $up = \text{argmax}\,(dt[let\_up]))$
    $down = \text{argmin}\,(dt[let\_down]))$
    $w = \text{zeros}(K)$
    $w[up] = 1$
    $w[down] = -1$
    $taumin = 0$
    $taumax = \min\,(1 - r_{t,up}^{(s)}, r_{t,down}^{(s)}, 0.1)$
    $\tau = \text{optimizeStep}(\mathcal{O}_s, R_s, A, \pi, taumin, taumax, w)$
    $r_{t\cdot}^{\text{new}(s)} = r_{t\cdot}^{(s)} + \tau \cdot w$

2.2.3. Method Outline

This section aims to give a short outline of the proposed estimation method that is based on the expectation-maximisation algorithm. The goal is to maximise equation 2.6 with respect to the parameters θ where $b_i(O_t)$ is given by our generalised emission distribution 2.8.

The EM algorithm starts with some initial values of θ, denoted by θ^{old}, where π and A are often initialised uniformly. The responsibilities R however have to be initialised adequate. Note, that the initial values have to meet the constrained conditions mentioned in section 2.1. Then we alternate between the expectation step, in that we evaluate the quantities $\gamma_t(i)$ and $\xi_t(i,j)$ given current values of

θ and the maximisation step, in that we compute updates for the parameters A, π and R according to the formulas and procedures presented in section 2.2.2. In some cases the proposed estimation method might lead to an oscillation between two parameter settings. For this reason we include a post processing step that determines the parameter estimates responsible for the maximum likelihood out of all iterations. The complete estimation method is summarised in Algorithm 3. Figure 2.7 illustrates an example of use. An example of the responsibility updates during the EM algorithm is given in figures B.1 and B.2 in appendix B.

Algorithm 3: EM (Baum-Welch) algorithm with post processing step to obtain updated parameter estimates θ.

Data: Initial parameters $\theta^{\text{old}} = \{\pi^{\text{old}}, A^{\text{old}}, R^{\text{old}}\}$, Observation sequence $\mathcal{O} = (O_1, ..., O_T)$, Threshold ϵ.
Result: Updated parameters $\theta = \{\pi, A, R\}$

begin
  Set $\mathcal{L}(\theta^{\text{new}}|\mathcal{O}, \mathcal{S}) = \infty$
  Calculate $\mathcal{L}(\theta^{\text{old}}|\mathcal{O}, \mathcal{S})$
  Set $iter = 0$
  while $\mathcal{L}(\theta^{new}|\mathcal{O}, \mathcal{S}) - \mathcal{L}(\theta^{old}|\mathcal{O}, \mathcal{S}) > \epsilon$ **do**
    Set $\theta^{\text{old}} = \theta^{\text{new}}$
    Set $iter = iter + 1$
    E step:
    Calculate $\forall t, i, j$
      $\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{K} \alpha_t(j)\beta_t(j)}$
      $\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{K}\sum_{j=1}^{K} \alpha_t(i)a_{ij}}$
    M step:
    Calculate $\forall t, i, j$
      $\pi_i^{\text{new}} = \gamma_1(i)$
      $a_{ij}^{\text{new}} = \frac{\sum_{t=2}^{T} \xi_{t-1}(i,j)}{\sum_{t=2}^{T} \gamma_{t-1}(i)}$
      R^{new} using algorithm 2
    $\Theta_{iter} = \theta^{\text{new}}$
  $\theta = \text{argmax}\left(\mathcal{L}(\Theta_i|\mathcal{O}, \mathcal{S})\right)$
end

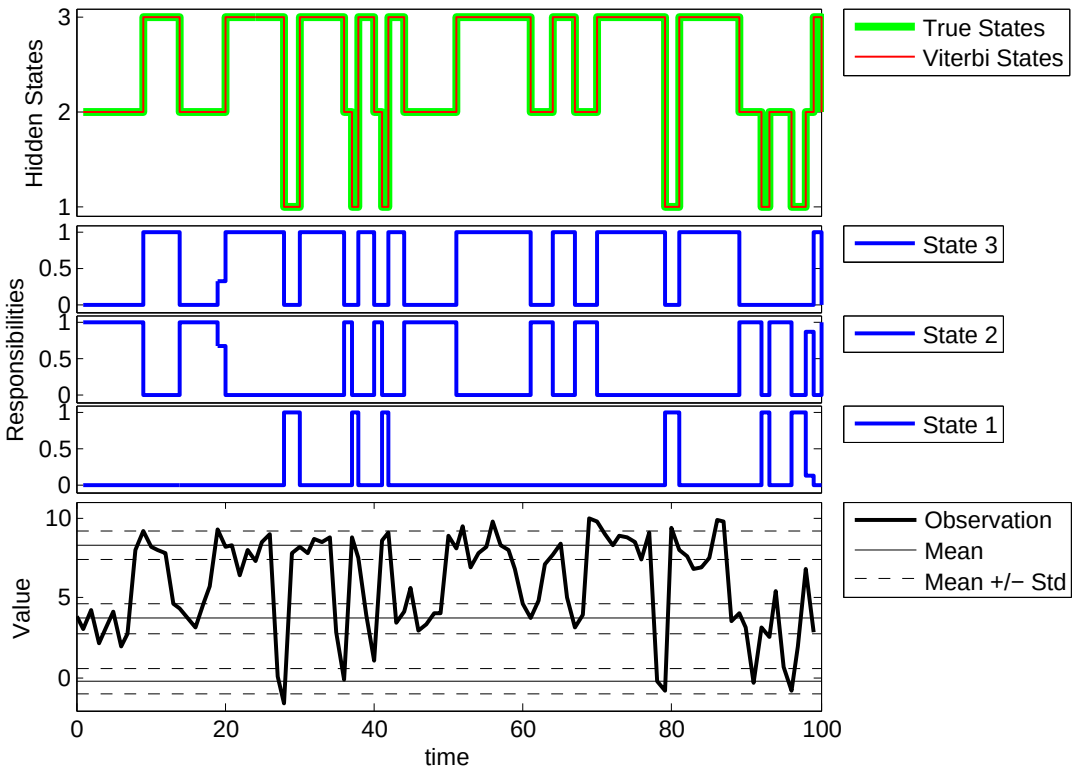

Figure 2.7.: An illustration of a hidden Markov model with generalised emission distributions. The observation sequence $\mathcal{O}$ that was generated by a HMM with $K = 3$ hidden states and transition probabilities $a_{11} = 0.2$, $a_{12} = 0.3$, $a_{13} = 0.5$, $a_{21} = 0.1$, $a_{22} = 0.7$, $a_{23} = 0.2$, $a_{31} = 0.1$, $a_{32} = 0.1$ and $a_{33} = 0.8$ can be viewed at the bottom. The responsibilities $r_{\cdot 1}$, $r_{\cdot 2}$ and $r_{\cdot 3}$ computed through the method introduced in section 2.2.2.3 are shown in the middle. Note that some responsibilities are not equal to zero or one, which is the case when the observation can not clearly be attributed to any of the states. Observation 18, which lies between state two and three can be mentioned as an example of this effect. In the top diagram one can see the corresponding true hidden state sequence $\mathcal{S}$ as well as the estimated hidden state sequence $\mathcal{S}^*$. As it can be seen, the proposed method is able to identify all hidden states perfectly.

2.2.4. Implementation

An implementation of the proposed method is available for MATLAB (Mat, R2011a). The implementation is based on a free MATLAB toolbox written by Murphy (2005), that provides the original Baum-Welch algorithm as well as the Viterbi algorithm. Due to the computational complexity it is not recommended to apply the method to more than 10000 observations in one run. For more details on the running time and for an improvement approach see section 3.6.

3. Simulation

This chapter aims to give an insight into the performance of the proposed generalised hidden Markov model. In order to see whether the proposed method is able to analyse data generated by a hidden Markov model we simulate data based on different HMM parameters and scenarios. As a benchmark we use the normal hidden Markov model introduced in section 2.1. To be able to compare the results we introduce a performance measure.

3.1. Performance Measure

In order to compare the results of the common and the generalised hidden Markov model (GHMM) we use a performance measure that is based on the state sequences.

The advantage of simulation studies is that we know the true model parameters, as well as the true state sequence $\mathcal{S}$. However, since a hidden Markov model depends on many parameters, which is especially true for the generalised model, it is not meaningful to use the discrepancies of the true and the estimated parameters as a performance measure. Furthermore the emission distribution of the original HMM has different parameters than the emission distribution of the generalised model, which means that we can not compare the performance of those models by simply comparing the parameter estimates. Another natural approach would be to compare the likelihood of each model. This however, can neither be applied since the generalised model does not have a real probability distribution.

Though we can use the discrepancy between the true hidden state sequence and the estimated hidden state sequence $\mathcal{S}^*$ as a measure of model goodness. Where the estimated state sequence is calculated by means of the Viterbi algorithm, already introduced in section 2.1.2. In this way we are able to compare models with different parameters, as it is the case with the original and generalised HMM. Moreover, all model parameters incorporate into the score since the Viterbi algorithm makes use of them. The proposed score is given by

$$P = \frac{|\mathcal{S} \cap \mathcal{S}^*|}{T}, \tag{3.1}$$

where T is the number of states in the sequence of interest and $|\mathcal{S} \cap \mathcal{S}^*|$ is the number of correctly identified states. Thus the score is defined in the interval

$[0, 1]$ and is just the percentage of correct classified states in a sequence.

3.2. Scenario 1: Initialisation

At first we want to get an idea of how the proposed GHMM performs under consideration of different initial values. Therefore, we simulate data based on 10 sequences per $T = 100$ observations with $p = 1$ dimensions and $K = 2$ states according to the simulation procedure described in algorithm 1. In doing so, we want to generate data where the time depending structure of the HMM is highly pronounced. Therefore we generate the transition probability matrix A according to

$$A = (1 - \lambda) \cdot \begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{pmatrix} + \lambda \cdot \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix},$$

where λ is a value on the interval $[0, 0.2]$ drawn from a uniform distribution. The reason for this diagonally dominant matrix is because high probabilities for staying in a particular state result in data strongly depending on the time information. In contrast, uniformly distributed transition probabilities would result in data simulated for a standard clustering task without any time dependencies. The initial state probability π is drawn from a uniform distribution. The samples are drawn from a Gaussian distribution with $\mu = (0, 4)$ and $\sigma^2 = (1, 1)$. That means that there is just a small overlap between the two states and hence the method should practically be able to identify all hidden states.

As already mentioned in section 2.1.4 the results of the original HMM mainly depend on the initialisation of the emission parameter and not that much on the initial values of the transition- and initial-state probabilities. For that reason we use uniformly distributed initial values for the transitions A as well as for the initial-state probabilities π. As initialisation for the responsibilities R we use

$$R^{\lambda}_{\text{init}} = (1 - \lambda) \cdot \Gamma + \lambda \cdot F, \tag{3.2}$$

where F is a $T \times K$ matrix with all entries equal to $\frac{1}{K}$ and $\Gamma = \{\gamma_t(i) | t = 1, ..., T; i = 1, ..., K\}$ is given through the forward-backward terms 2.4a computed by a standard hidden Markov model. Remember that $\gamma_t(i)$ is the probability of being in state i at time t given model parameters and observations. So this seams to be a suitable initialisation for the responsibility r_{ti}. Note that it is also possible to use a standard fuzzy clustering technique, for example, the non-Euclidean fuzzy k-means method proposed by Müller (2011) to initialise the responsibilities.

However, we use equation 3.2 with λ values that vary between $\{0, 0.2, 0.4, 0.6, 0.8\}$ within each simulation for the initialisation. Thus, in each step we are moving further away from an optimal initialisation in order to observe a difference in the performance of the GHMM. In addition to see whether the proposed method is able to find the hidden states under a "worst-case" initialisation we generate

random initial responsibilities. In total we compute generalised hidden Markov models based on six different responsibility initialisations. The score of each method, which is given by 3.1 is then compared to the score of the standard HMM.

3.2.1. Results of Scenario 1

The results of the simulation described above are illustrated in figure 3.1. Due to the computationally expensive nature of the generalised HMM we were only able to simulate 20 datasets. We used the original hidden Markov model as a benchmark model which is indicated through the red colored boxplot. As one can see, the method is able to find almost all hidden states and there is just little difference between the different responsibility initialisations. What is more,

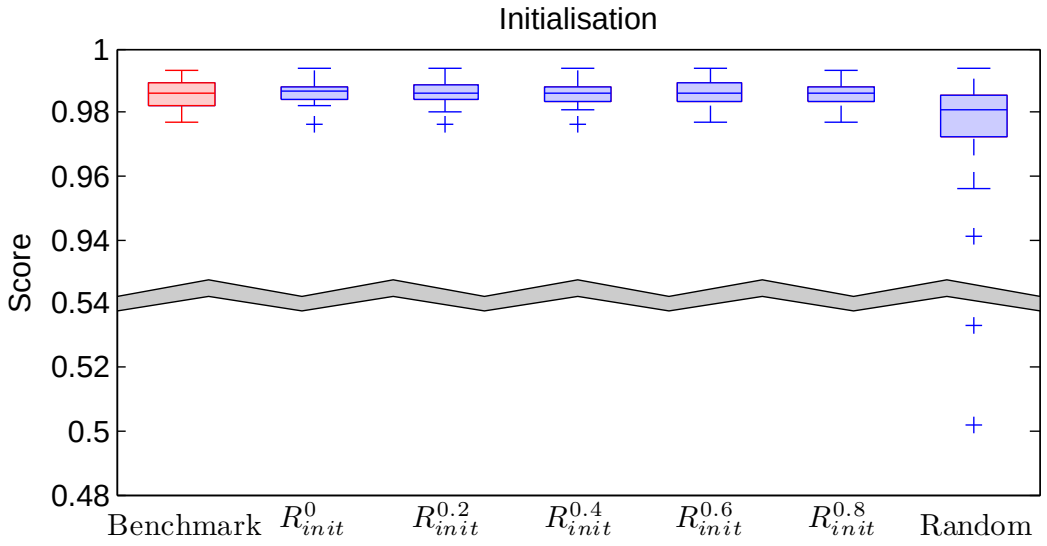

Figure 3.1.: Performance of the generalised hidden Markov model under consideration of different responsibility initial values. Where $R^{\lambda}_{\text{init}}$ indicates an initialisation of R according to 3.2. The results are based on 20 datasets with 10 sequences per 100 observations, one dimension and two clusters with mean $\mu = (0, 4)$. The boxplots show the distribution of the mean score for each dataset, whereas the benchmark model is given by the original HMM. Note that the scale on the Y axis is interrupted in order to provide a better illustration of the outliers at the random initialisation.

the proposed method performs at least as good as the original method. In some cases the GHMM even performs slightly better than the original HMM. However

there are two huge outliers for the random initialisation. This suggests that the random initialisation may not always work as good as other initialisations. Note that this result is based on two well separated clusters with $\mu = (0, 4)$. The same conclusion, however, also holds true for observations drawn from clusters with greater overlap. See figure B.3 in appendix B for the corresponding boxplot with cluster centers $\mu = (0, 2)$. To summarise, one can say that the method does not as strongly depend on the initialisation as it is the case with the original HMM. Nevertheless we suggest using responsibility initial values that are based on $\gamma_t(i)$, since our conclusions are based on merely two rather artificial scenarios. In the ongoing work we use responsibility initial values of R^0_{init} since this initialisation, in some cases, performs slightly better than others.

3.3. Scenario 2: Gaussian Distribution

To get a deeper insight into the performance of the GHMM we simulate data with various difficulties. Again we sample data based on 10 sequences per $T = 100$ according to the simulation procedure described in algorithm 1. But now the cluster centers μ are chosen randomly from the interval $[0, 5]$ and the variance σ^2 is fixed to 1. In this way we get clusters with varying overlap and hence various difficulty. Moreover, we either increase the dimension in each step by two ($d = d + 2$) or the number of hidden states by two ($K = K + 2$) starting with an initial K and p of 2. The maximum number of clusters and dimensions was set to 10, what leads to five different scenarios within each simulation set-up. This approach aims to show if the results of the proposed method are depending on the data dimension as well as on the number of states. The transition probabilities and initial state probabilities were again generated according to the procedure described in section 3.2.

3.3.1. Results of Scenario 2

The results of scenario 2 are based on 20 datasets in each case and are illustrated in figures 3.2 and 3.3. We used the original HMM as a benchmark model again.

Figure 3.2 shows the performance when a variation in the number of clusters is taken into account, whereas the number of dimensions is fixed to $d = 2$. As one can see, the median score of the GHMM is either equal or lower compared to the corresponding score of the original HMM. Taking a random fluctuation due to the small number of simulations into account we can say that the general performance levels of both methods are quite similar. With an increasing number of clusters the performance of both methods decreases. That is obvious since the more cluster, the higher the probability that some clusters show the same or similar characteristics. Thus with a higher number of clusters it is much more difficult to distinguish between them.

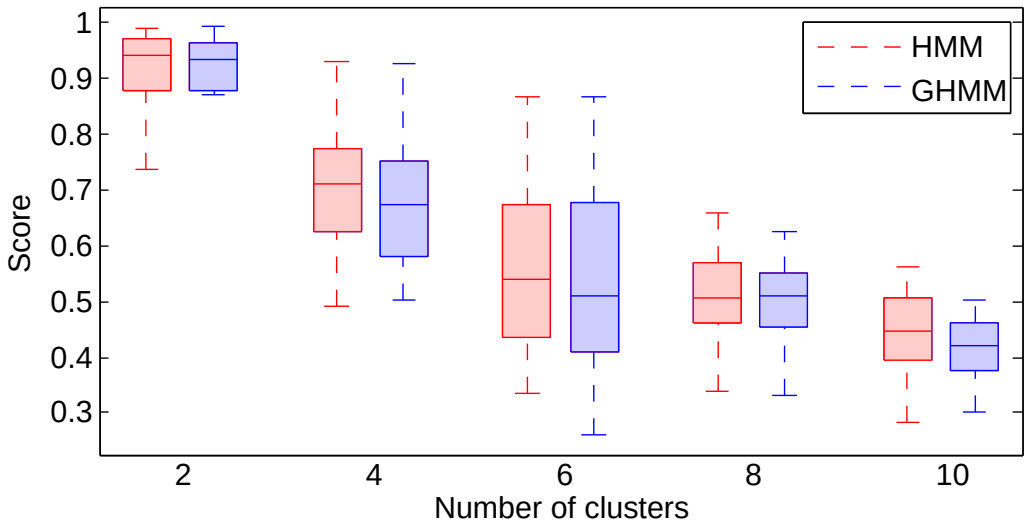

Figure 3.2.: Performance of the generalised hidden Markov model compared to the original HMM under consideration of different numbers of clusters. The results are based on 20 datasets with 10 sequences per 100 observations and the cluster centers are drawn randomly from the interval $[0, 5]$. Each boxplot tuple shows the distribution of the mean score for each dataset for both algorithms. The number of clusters increases from one tuple to the next while the number of dimensions is kept constant ($p = 2$).

In contrast, figure 3.3 shows the results of the algorithms with an increasing number of dimensions. In this case the number of clusters is fixed to $K = 2$. We can see that the original hidden Markov model is superior to the generalised model in almost all cases. What is more, the variance of the GHMM is higher than the corresponding variance of the HMM, at least for the cases with more than two dimensions. However, we can also observe that an increasing number of dimensions comes along with an increase of the mean score in both models. This is quite intuitive since the more dimensions, the easier it is to separate the clusters from each other.

In total, one can say that the number of clusters does not seem to have a huge effect on the performance of the GHMM compared to the HMM. The number of dimensions does, however, have an effect on the performance in the way that the original HMM provides better results than the generalised HMM. A reason why the original hidden Markov model outperforms the generalised model in some cases might be the higher number of parameters that have to be estimated for the GHMM. Another reason is that the original HMM is predestinated for

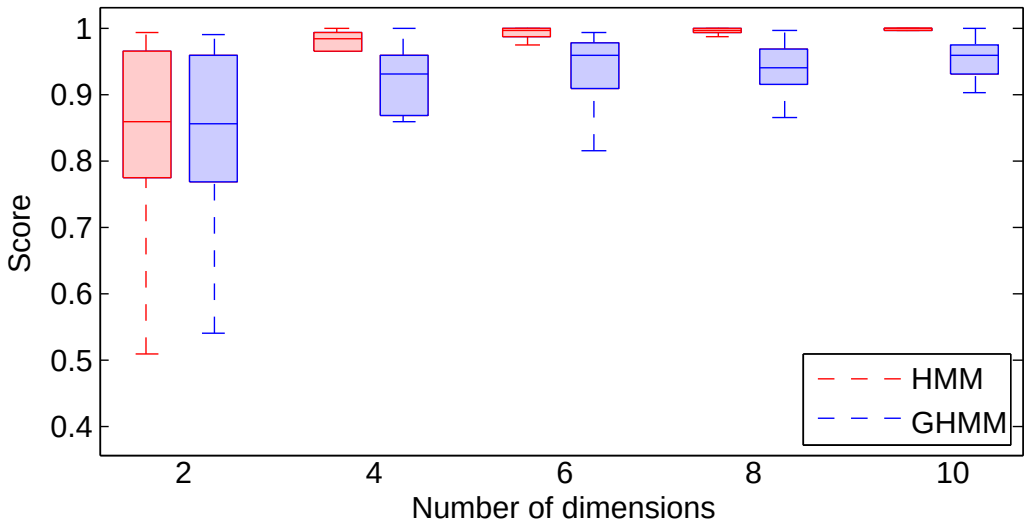

Figure 3.3.: Performance of the generalised hidden Markov model compared to the original HMM under consideration of different data dimensions. The results are again based on 20 datasets with 10 sequences per 100 observations and the cluster centers are drawn randomly from the interval $[0, 5]$. Each boxplot tuple shows the distribution of the mean score of each dataset for both algorithms. The number of dimensions increases from one tuple to the next while the number of clusters is kept constant by $K = 2$.

the analysis of Gaussian data as simulated in this scenario since a multivariate Gaussian distribution is used as emission distribution. For this reason we simulate non-Gaussian data predestinated for the GHMM in the following section.

3.4. Scenario 3: L_q Norm

So far we have only evaluated the performance of GHMM on spherical data drawn from a Gaussian HMM, and therefore we could not expect to outperform the original HMM model. However, as mentioned before with our generalised emission distribution we are able to use other distance measures respectively other emission distributions. Suppose we have prior knowledge about the non-spherical cluster structure and the distance measure that fits the data best. We want to see whether the prior information has an influence on the results or not.

We use a similar simulation as in scenario 2 but this time we use observations obtained from a distribution having a non-spherical shaped structure. This leaves

us with the question of how we are able to simulate data from such a distribution. In contrast to spherical data, which we can easily sample from a Gaussian distribution, it is more complicated to sample non-spherical data. However, the shape of non-spherical clusters meets in many cases some special distance measure condition. Therefore we have the advantage that we can draw conclusions on the optimal distance measure d_{ij} from the cluster structure. For this reason we would like to sample data whose structure meets the shape of some particular norm. A distance measure which is known as the L_q norm, is defined as

$$||x||^q = \left(\sum_{i=1}^{n} |x_i|^q \right)^{1/q}, \tag{3.3}$$

where $x = (x_1, ..., x_n)$ is a vector and q indicates the norm. The L_1 norm is known as the Manhattan norm, L_2 is known as the Euclidean norm and L_∞ is known as the Maximum norm. Figure 3.4 gives a graphical illustration of the three norms. In order to simulate data with different norm shapes we make use of a

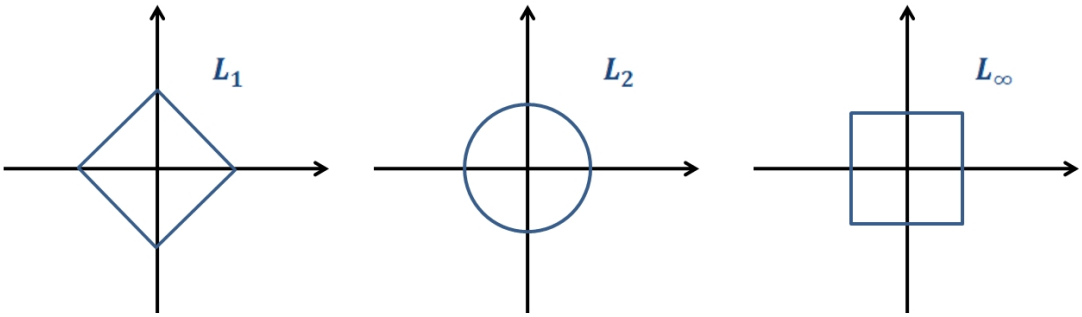

Figure 3.4.: A graphical illustration of three different L_q norms and their corresponding unit circles. The distance from the origin is calculated according to equation 3.3.

rejection sampling (Robert and Casella, 1999), that is described in the following subsection.

3.4.1. Rejection Sampling

The rejection sampling is based on the idea, that for sampling from the distribution of interest we can draw samples from a simpler distribution than the distribution of interest and reject or accept them with a certain probability, depending on the distribution of interest.

Assume we want to draw samples from a difficult distribution $f(x)$, which is called the target distribution. Instead of directly drawing from $f(x)$, which is

difficult for some reasons, we can sample from a simpler distribution $g(x)$, which is called the proposal distribution. We accept samples from $g(x)$ with some probability

$$p = \frac{f(x)}{M \cdot g(x)},$$

where $M \in \mathbb{R}$ is a rejection constant such that $f(x) < M \cdot g(x)$ for all x. Thus $M \cdot g(x)$ forms an envelope over the target distribution $f(x)$. We then accept a sample x from $g(x)$ as a random realisation of $f(x)$ if $u < p$, where u is a random sample drawn from a uniform distribution on the unit interval. Otherwise, if $u > p$ we reject x and repeat the sampling process by drawing a new x from $g(x)$.

Applied to our problem of sampling data from different L_q norms we can generate L_2 norm shaped clusters as a proposal by drawing from a Gaussian distribution. The proposal distribution is then given through the L_2 norm by

$$g(x) = \frac{\exp\left(-||x||^2\right)}{2\pi}.$$

If we would like to have a random sample of data following a particular L_q norm shape we can set the target distribution to

$$f(x) = \frac{\exp\left(-||x||^q\right)}{2\pi}.$$

Realisations from a distribution preferring a L_q norm shape are then obtained by proceeding according to the rejection sampling described in the last paragraph. Examples of clusters drawn from different L_q norms are given in figure 3.5.

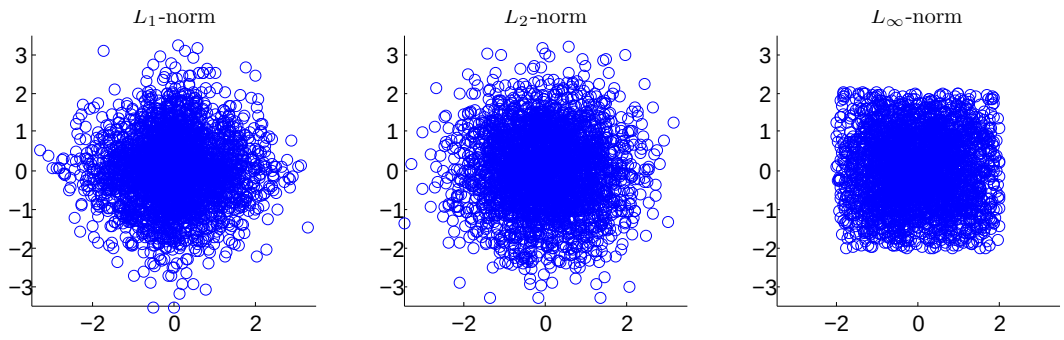

Figure 3.5.: Examples of 2-dimensional clusters drawn from different L_q norms. Note that the clusters follow the form of the corresponding unit circles given in figure 3.4.

3.4.2. Results of Scenario 3

The results of scenario 3 are shown in figure 3.6. The simulations are based on 20 datasets with 10 sequences per $T = 100$ observations with a fixed number of dimensions ($d = 2$) and a fixed number of clusters ($K = 2$). The data follows a L_∞ norm shaped structure and is sampled according to the rejection sampling described in section 3.4.1. The cluster center is set to the origin $(0, 0)$ in one dimension and drawn randomly from the interval $[0, max\_dist]$ in the other dimension. We decrease the maximal distance $max\_dist = \{4, 3, 2, 1, 0.5\}$ in each step. In this way we get clustering tasks with various difficulty since the smaller the maximum distance, the more overlap between the clusters and thus the more difficult to distinguish between them.

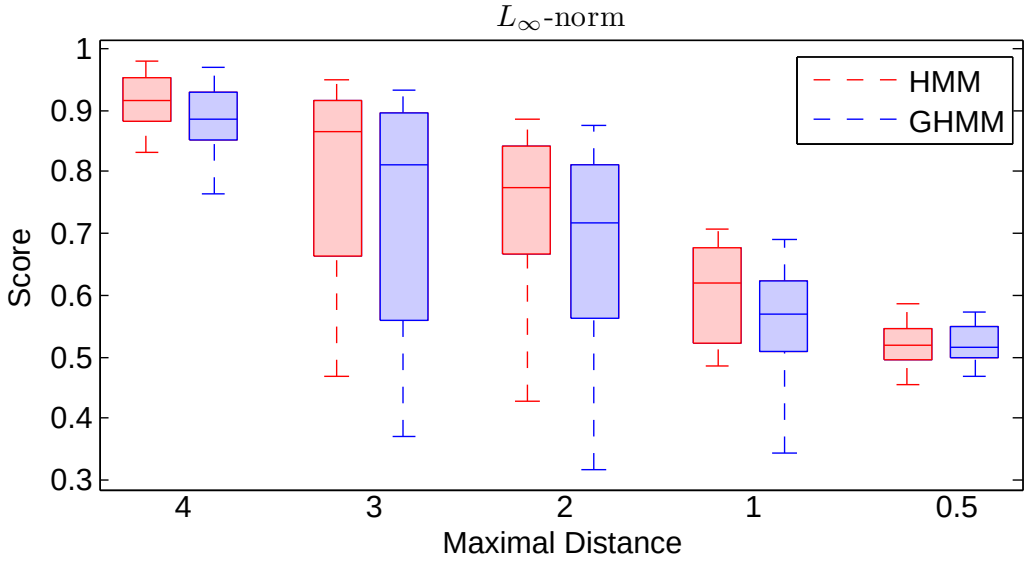

Figure 3.6.: Performance of the generalised hidden Markov model on L_∞ norm shaped data. Each boxplot is based on 20 datasets (10 sequences, $T = 100$), where the maximal distance between the cluster centers is decreased from one boxplot double to the next in order to increase the difficulty. The L_∞ norm is used as distance measure for the GHMM as it is the most adapted measure to the existing data situation. The number of dimensions ($d = 2$) and the number of clusters ($K = 2$) is fixed.

In contrast to scenarios 1 and 2 where we used the L_2 norm as distance measure we now use the L_∞ norm. This is appropriate as we have prior knowledge about the non-spherical data structure. However, as one can see in figure 3.6, we are not able to gain considerable advantage compared to the original HMM. The

generalised model performs worse than the ordinary hidden Markov model in all considered cases. It can be concluded that in this scenario adapting the distance measure does not lead to a superior model.

3.5. Scenario 4: Nominal Data

In this section we want to demonstrate that our method is able to deal with a wider range of data than the ordinary HMM. For this reason, we aim to investigate how the proposed model performs when we are faced with nominal data. Nominal data is given when we are dealing with categorical data where the order of the categories is arbitrary. Note that certain statistical concepts, for example mean or variance are meaningless for nominal data. However, keep in mind that the generalised observation density as proposed in section 2.2.1 is not depending on an expected value or a variance. For this reason we might be able to gain advantage compared to the original HMM.

We simulate data based on 10 sequences per $T = 100$ observations with $p = 3$ dimensions and $K = \{2, 3, 4\}$ states. Each state emits a three-dimensional nominal observation where the probability of each category varies amongst the states. The number of categories is equivalent to the number of states and each category is preferred by another state. The transition probabilities and initial state probabilities were again generated according to the procedure described in section 3.2.

It has to be mentioned that one-dimensional nominal data, where each state is linked to exactly one category could easily be analysed using a normal Markov model without hidden states, as there is no variance in the output of each state and hence the true states are known in advance. However, higher dimensional nominal data, as in the present case cannot be analysed using a normal Markov model, since the number of possible states is getting too big for a finite state model.

It is obvious that in the present case it is not meaningful to use some L_q norm as distance measure as introduced in section 3.4, because of the unordered structure of the data. Therefore we use the Hamming distance (Hamming, 1950), which is the percentage of vector positions that differ:

$$d_{ij} = \frac{\sum_{p=1}^{n} \mathbb{1}_{\{i_p \neq j_p\}}}{n}$$

Using the generalised HMM we are thus able to work with nominal data without violating the unordered structure of such data. Strictly speaking, the standard hidden Markov model is not able to deal with such data, since the Gaussian emission distribution is dependent on a mean and a variance term. Both, however can not be defined in a proper way for nominal data.

3.5.1. Results of Scenario 4

The results illustrated in figure 3.7 are based on scenario 4 and 20 datasets in each case. Again we use the original HMM as a benchmark model which, admittedly is not an appropriate model for the present data situation as already described in the last section. Nevertheless, it serves as a benchmark model in order to show that our method is a more flexible approach and can be adapted to different data situations. As one can see the generalised model is superior to the original model in almost all cases. In the case of two and three clusters the GHMM was able to find the hidden states without any errors in nearly 50 percent of the cases. Regarding the case of four clusters, the variance of the GHMM is larger and the performance declines, but is still better than the ordinary HMM in most cases. We can conclude that when dealing with nominal data our method in general outperforms the HMM. The reason for the superioritiy is the adjusted distance measure, which is adapted to the particular data situation.

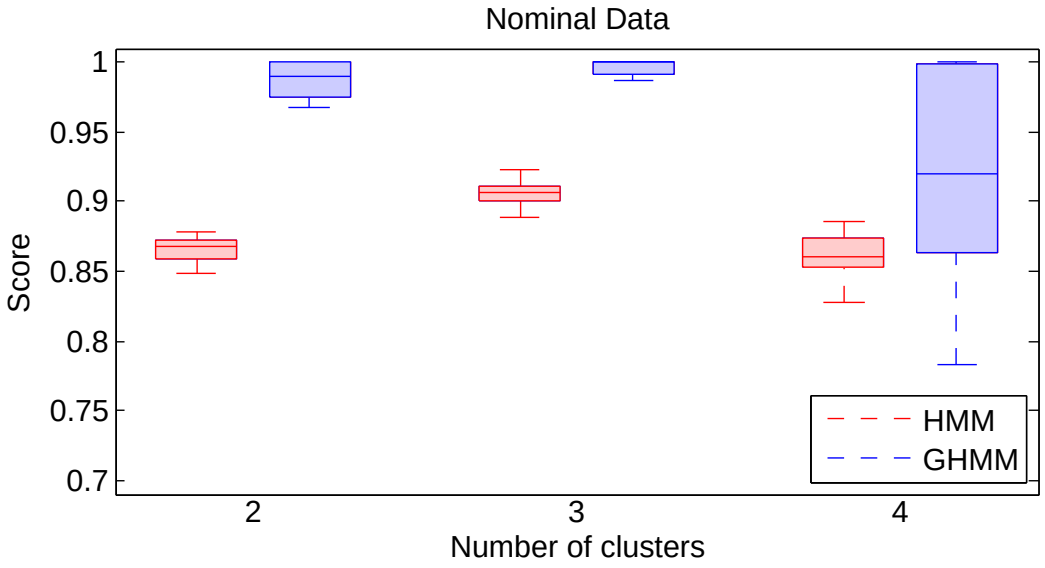

Figure 3.7.: Performance of the generalised hidden Markov model on nominal emission data. The results are based on 20 datasets with 10 sequences per 100 observations. The number of clusters, which is equivalent to the number of categories is increased from one tuple to the next and the number of dimensions is kept constant ($p = 3$). The GHMM is superior to the HMM in this case because of the modified distance measure which is adapted to the particular data situation.

3.6. Scenario 5: Running Time

The simulations in scenarios 1 - 4 have shown that the greatest weakness of the GHMM is the long running time. Compared to the standard HMM we are faced with an enormous increase in the running time. The increase on the one hand is based on the fact that the generalised model depends on considerable more parameters than the ordinary model. On the other hand the responsibility updates are derived through an analytical optimisation strategy and not through a closed form. The most time consuming part however is the calculation of the generalised emission distribution as well as its gradient. Note that the generalised emission distribution is defined by the sum over all pairwise observation distances.

It should be clear that a distribution based for example on 10000 observations does not change in a serious way when only a random subset of for example 5000 observations is taken into account. So in cases with many observations it is not necessary to include all observations in each update step as the distribution should already be stable when based on only a fraction of the observations.

So in order to speed up the estimation of GHMM parameters we want to investigate what amount of observations is sufficient to observe the same results as when using all observations. Furthermore we compare the running time of all simulations to see if the exclusion of observations leads to a decrease in running time.

In each iteration of the EM algorithm (see also algorithm 3) we choose a different random subset of observations for which the parameter updates are calculated. In this way all observations are included by chance and artificial effects are avoided. However, choosing the subset totally random could result in underrepresented states. Consider the case when the subset includes no observations with highly pronounced responsibilities in a specific state. In this case the responsibility updates for all observations associated with this particular state would be underrepresented in the affected state. Therefore we make use of a more restrictive sampling strategy. Besides sampling randomly we also include 10 of the most representative observations for each state. Thus we avoid subsamples with "missing" states.

The procedure described above is evaluated on one-dimensional datasets based on 10 sequences per $T = 100$ observations from two different states. The cluster centers μ are again chosen randomly from an interval between 0 and 5 and the variance σ^2 is set to 1. In addition to the original HMM and the generalised HMM based on all observations we compute GHMMs based on different fractions of $\{80, 60, 40, 20, 10, 5\}$ percent of all observations. Note that we have 1000 observations in total, since we have 10 sequences per 100 observations. Furthermore we stopped the elapsed time for each single computation.

3.6.1. Results of Scenario 5

As one can see in figure 3.8 the score of the GHMM does not change to a great extent when only a subset of observations is considered. Even in the case when a fraction of only 5% of the observations is taken into account the distribution of the score is as good as when all observations are considered. Also in comparison to the ordinary HMM, the GHMM performs equally well under all considered observation fractions. If we look at the running time, that is indicated in minutes,

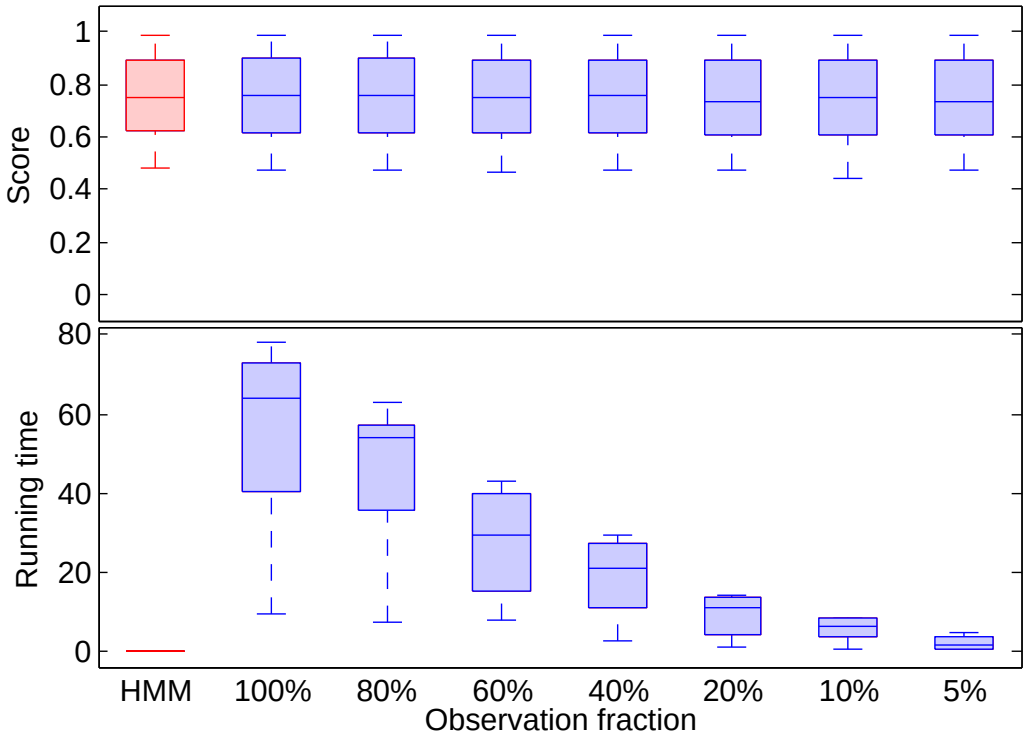

Figure 3.8.: Performance of the generalised hidden Markov model under consideration of different observation subsets. Whereas the benchmark model, indicated in red, is given by the original HMM. The corresponding distribution of the running time in minutes is shown on the bottom. The results are based on 20 datasets with 10 sequences per 100 observations, which means that an observation fraction of 100% is equivalent to 1000 observations.

of both methods we observe a huge difference. The computation of an original hidden Markov model takes on average about 9 seconds whereas the computation of an GHMM can take up to 3600 times longer. Nevertheless, the running time of the generalised model decreases with decreasing number of considered obser-

vations. In comparison to the computation of a GHMM with all observations that takes in average about 54.14 minutes we are faced with a mean running time of about 2.19 minutes when only 5% percent of all observations are taken into account. This means that we are able to achieve an improvement in running time by at least a factor of around 24 at a constant performance level. However, there might be a trade-off between the number of necessary iterations and the running time of one iteration. The less observations are taken into account, the shorter the running time of one iteration, but the more iterations are potentially necessary. An indication of this might be that in some cases the running time when all observations are considered is almost as low as when only 20% of the observations are taken into account. Nevertheless, the proposed running time optimisation seems to work very well. A reason for this result might be the good initialisation. Remember that the responsibilities are initialised according to R^0_{init} in 3.2. Thus the responsibilities are almost perfect at the beginning of the iterations in many cases and do not have to change anymore. For this reason we run the same simulations with a responsibility initialisation of $R^{0.6}_{\text{init}}$. In this way we can see whether the results are just a matter of the good initialisation or if the results also hold true for weaker initialisations. It turns out that they also hold true for a responsibility initialisation of $R^{0.6}_{\text{init}}$. See figure B.4 in appendix B for the corresponding boxplots. Here, the trade-off mentioned above is clarified by the lower median running time in case of 100% observations compared to 60% observations.

3.7. Conclusions

Through the scenarios 1-5 we have gained an insight into the performance of the generalised hidden Markov model. It has to be noted that the scenarios are only based on 20 datasets and therefore a random fluctuation of the results due to the small number of simulations has to be considered in each case.

First we have shown that our method is not highly dependent on the responsibility initialisation. However, we suggest to initialise the responsiblities with the forward-backward terms 2.4a computed by a standard hidden Markov model. The same initialisation has been used throughout the remaining scenarios.

Furthermore, when applied to Gaussian data we have shown that the performance decreases with an increasing number of clusters and increases with an increasing number of dimensions. This effect however can also be observed when the original HMM is applied. Although both models behave similar under the considered Gaussian scenarios, the original model provides better results than the generalised model when an increasing dimensionality is taken into account. In contrast there is no difference between both models when the number of clusters is increased.

The results in scenario 3 in general do not meet our expectations. We expected

that the generalised hidden Markov model is superior to the original model when applied to L_∞ norm shaped data because of the adapted distance measure used. The original model, however, outperforms the generalised model in this scenario. Thus, having prior information about the data is not beneficial in this particular case.

For nominal data we have shown that the proposed GHMM provides better results than the original HMM. The reason for this is the modified distance measure, which is adapted to the particular data situation. Besides providing better results we have also achieved a method which can deal with nominal data, without violating some statistical concepts such as variance or mean. This does not apply to the original HMM. In scenario 4 it becomes clear, that having prior knowledge about the data and adjusting the distance measure to the needs of this data structure can lead to GHMM results that are better than the HMM results.

In the last scenario we have shown that we are able to achieve a great running time reduction at a constant performance level. The improvement is based on an adjusted parameter estimation procedure in which in each iteration only a random subset of observations is taken into account. This approach enables us to apply the generalised hidden Markov model to larger amounts of data such as for example sequential biological DNA data.

To summarise the results of the investigated scenarios we can say that the performance of the original HMM is superior to the performance of the GHMM in cases with more than two-dimensions. However, when looking at two dimensional data our method performs as good as the original model. Besides we were also able to show the potential improvement of an adjusted distance measure in the case of nominal data. In this case the information about the data leads to results which are superior to the results of the original HMM.

4. Application to ChIP-chip Data

In the last chapter we have shown that the proposed GHMM is able to analyse data generated by a hidden Markov process. Now, in order to see how the method performs when applied to real data, we apply the GHMM to ChIP-chip data provided by the lab of Patrick Cramer (Mayer et al., 2010). A description of the data and the related biological question is given in the following section.

4.1. Data Description

The ChIP-chip (also known as ChIP-on-chip) method is a technique for the determination and characterisation of the DNA binding behavior of a particular protein of interest (POI), which combines chromatin immunoprecipitation (ChIP) and microarray technology (chip). Chromatin immunoprecipitation is a technology that isolates the DNA fragments that specifically bind to that particular POI and microarray is a method to measure the expression levels of these DNA fragments.

Proteins that are bound to the DNA of a cell at a certain time are cross-linked to the DNA. Afterwards the DNA is fragmented into smaller units and the protein of interest along with the bound DNA fragments is extracted using an antibody specific for the POI. The resulting DNA fragments are then separated from the POI and hybridized to a microarray. In this way we are able to determine the binding position and the corresponding binding intensity of the protein of interest. This provides valuable insight into the DNA binding motifs of a transcription factor.

Mayer et al. (2010) investigated the DNA binding behavior of Polymerase II (PolII) and several transcription factors during the transcription (Müller, 2011). The goal was to discover, characterise and quantify the different modes (states) of the PolII. It is well known that the binding behaviour changes during the transcription and that it shows a different phosphorylation pattern at the beginning than at the end of the transcription (Mayer et al., 2010). However, it is not yet clear whether there are gene specific transcription state sequences or if there is a general state sequence for all genes. In case the state sequences vary among genes the question arises if there are distinct gene classes that can be clearly identified by the different state sequences.

In order to get an insight into these questions we propose to estimate a hidden Markov model, where the hidden states correspond to the different transcription

factor modes and the observations correspond to the ChIP-chip occupancies of the transcription factors. So one might be able to identify different gene classes, characterised by different transcription states and patterns. As already mentioned in section 2.1.4 the HMM in the most common case assumes a multivariate Gaussian emission distribution in each hidden state. Since the ChIP-chip data provided by Mayer et al. (2010) do not follow a Gaussian distribution (see figure 4.1) we make use of the generalised hidden Markov model introduced in this thesis.

Mayer et al. (2010) were able to identify a general transcription initiation factor complex (TFIIB, Kin28, Tfg1 and Cet1), an elongation factor complex (Spt4, Spt5, Spt6, Elf1, Spn1, Bur1, Ctk1, Paf1 and Spt16) as well as a termination factor complex (Pcf11). The elongation factors could be separated into three groups, depending on the time at which the particular group is present during the transcription (Müller, 2011). This means that Mayer et al. (2010) were able to identify five different factor groups with 14 transcription factors in total. Given the already mentioned computational complexity of our method we were only able to use the normalised ChIP-chip profiles of a subset of ten genes out of 200 genes, whereas a gene is equivalent to a sequence in terms of hidden Markov models. For

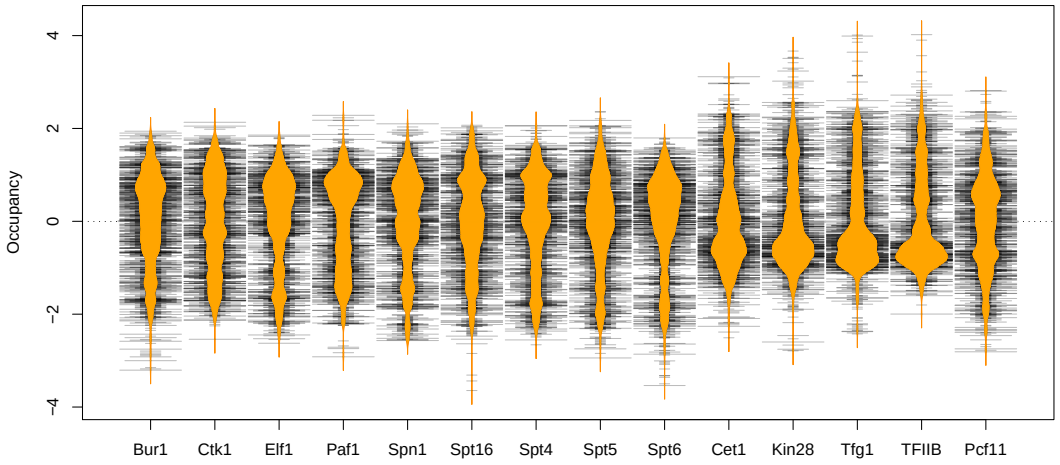

Figure 4.1.: The occupancy distribution of each transcription factor is shown in orange, the individual observations are shown in grey bars. The distribution plots were generated using the beanplot (Kampstra, 2008) R package.

each gene, we have $p = 14$ transcription factors (dimensions) as well as $T = 1350$ positions (observations). The distribution of each transcription factor is shown in figure 4.1. As we can see we can clearly distinguish between the initiation

factor distributions and the elongation factor distributions. If we take a look at the occupancies of all transcription factors at each position of the gene we can also see differences between factors that are present at the transcription start site (TSS), during the transcription and around the polyadenylation (pA) site. An example of the mean transcription factor modifications during the transcription can be found in figure 4.2.

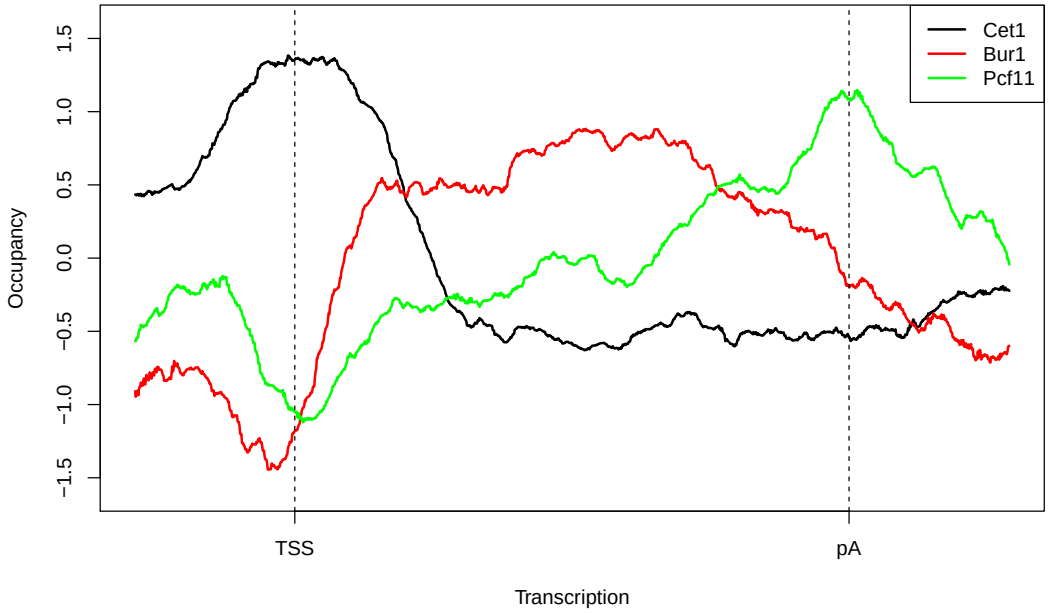

Figure 4.2.: The mean occupancy of selected transcription factors during the transcription. The dashed black lines indicate the transcription start site (TSS) and polyadenylation (pA) site, respectively. Cet1 is one of the initiation factors, Bur1 one of the elongation factors and Pcf11 is a termination factor.

4.2. Results

We applied our generalised hidden Markov model with $K = 6$ hidden states to the dataset described in section 4.1. The transition and initial probabilities of the model were initialised uniformly. For the initalisation of the responsibilities we used

$$R_{\text{init}} = \{r_{ti} = \gamma_t(i) | t = 1, ..., T; i = 1, ..., K\},$$

that is given through the common HMM as described in section 3.2. The common HMM, in turn, was initialised through a k-means clustering. In order to accelerate the estimation of GHMM parameters we used the speed optimised procedure introduced in section 3.6 with a subset of 200 observations.

The matrix of hidden state sequences determined through the Viterbi algorithm (see also section 2.1.2) is shown in figure 4.3. As one can see the most dominating state is the yellow one (state 4), while the blue state (state 3) is hardly visible and state 6 is not present in any gene at any time. The majority of genes show the same sequential pattern at the beginning of the transcription, where the red state is followed by the green state, which in turn is followed by the yellow state. The pink state mostly appears at the end of the transcription. To get

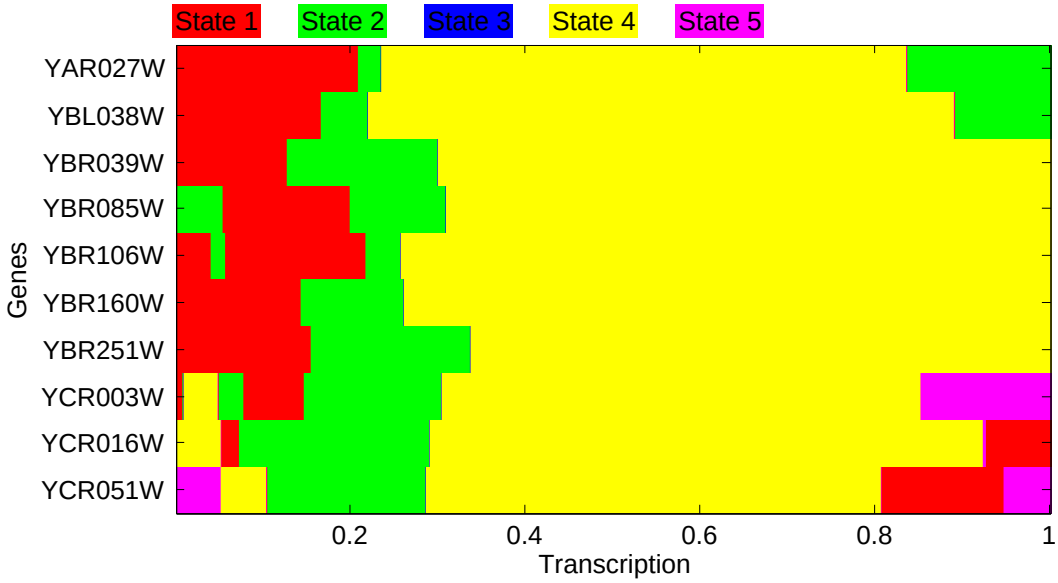

Figure 4.3.: Hidden state sequences of the ten examined genes. The sequences were generated using the Viterbi algorithm (see also section 2.1.2). Most genes show a sequential pattern of red (state 1), green (state 2) and yellow (state 4) at the beginning of the transcription. Note that state 3 is rarely present and that state 6 is not present at all.

another idea of the state appearance during the transcription, the histogram in figure 4.4 shows the frequency of each hidden state for several time points. This plot emphasises the dominance of the sequential pattern of red, green and yellow states. A reason why state 6 completely disappeared might be that it is occupied by the remaining states and their large variability. One approach to tackle this problem would be to regularise the pseudo-variance term in the denominator of

the emission distribution 2.8. However, further research is required in this area. The decreasing frequency of the yellow state and the increasing frequency of the pink state at the end of the transcription suggests that the pink state in general might be associated with the transcription termination.

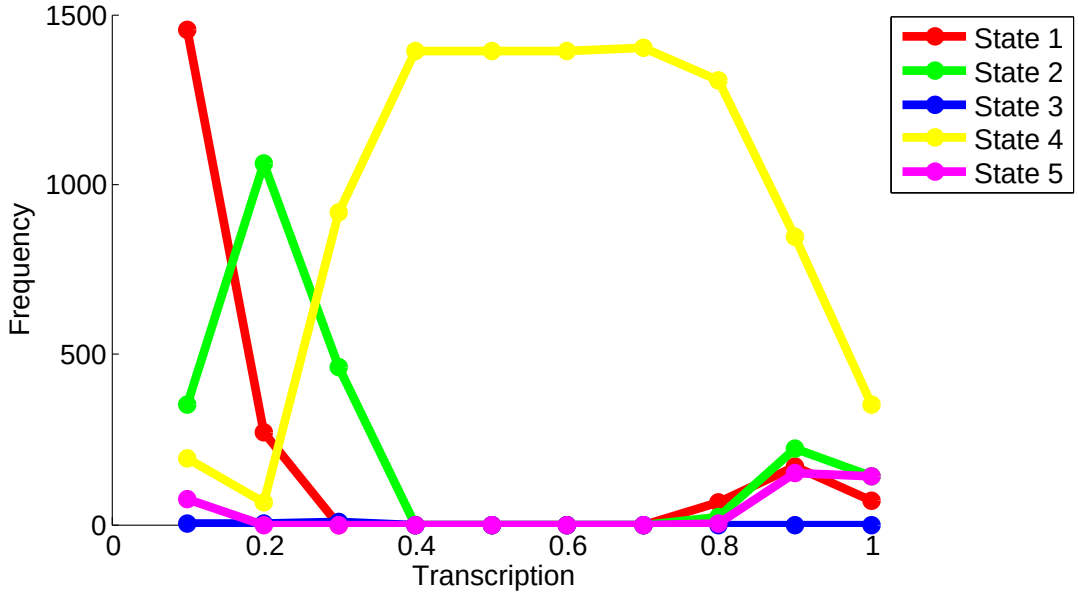

Figure 4.4.: Frequency of each hidden state during the transcription. As one can see, the red and the green states are mainly present at the beginning of the transcription and the yellow state can be found most of the remaining time. The figure is based on a script from Failmezger (2011).

When looking at the distribution of the transcription factor occupancies in each state, which is given in figure 4.5, we can observe considerable differences between the states. The red and green states show high values in the initiation factors (Cet1, Kin28, Tfg1 and TFIIB) and low values in the elongation factors. The yellow state on the other hand shows characteristics contrary to these states. This outcome was to be expected, since the red and green states are mainly present at the start of the transcription and so are the initiation factors. The same applies to the elongation factors, since the yellow state corresponds to clusters with high values in the elongation factors, which are mainly present after the transcription initiation. According to Mayer et al. (2010), initiation factors are exchanged for elongation factors. This pattern can also be found in our state assignments, since the green state is always followed by a yellow state.

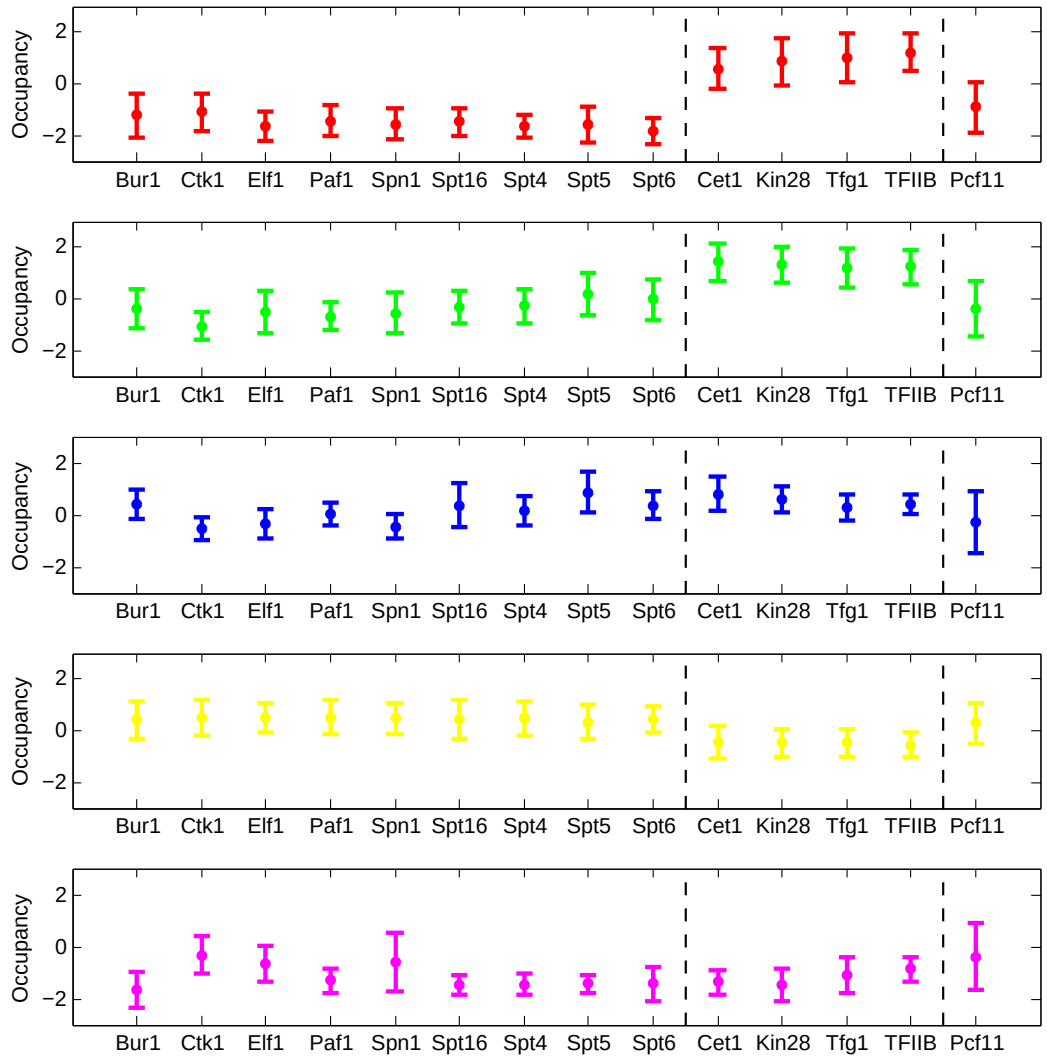

Figure 4.5.: Distribution (mean ± standard deviation) of the transcription factor occupancies in each state. Each state color matches the one used in figures 4.3 and 4.4. The dashed black lines separate the elongation (Bur1, Ctk1, Elf1, Paf1, Spn1, Spt16, Spt4, Spt5 and Spt6), initiation (Cet1, Kin28, Tfg1 and TFIIB) and termination (Pcf11) factors. One can clearly distinguish between states which are present at the beginning of the transcription (red and green) and states which are present at the end of the transcription (yellow and pink). Note that state 6 is not shown at all since it appears at no time during transcription.

The outcome described above suggests that we have determined biologically meaningful transcription states and patterns in the examined genes. However, in order to find distinct gene classes we would have to apply the proposed method to more than ten genes, which was not possible in the scope of this thesis due to the computational complexity of our model. Besides possibly finding distinct gene classes, increasing the number of considered genes would also lead to an increase of the accuracy of the observed transcription states. Nevertheless, the application of our method to ChIP-chip data has proven the ability of our approach to analyse sequential data and to get meaningful results.

5. Conclusions and Discussion

In this thesis we proposed a generalised modification of the well established hidden Markov model, which we call the GHMM. We derived a non-Euclidean generalisation of the emission distribution which is only expressed through pairwise distances between all observations and makes the explicit calculation of a cluster center or a cluster variance unnecessary. This allows us to deal with samples obtained from non-Euclidean metric space, provided that a proper distance measure is defined on the samples. Given prior knowledge of a particular distance measure that fits the non-Euclidean data structure best we are able to gain advantage over the ordinary hidden Markov model. However, due to the definition, the generalised emission distribution provides no characteristic cluster properties such as a mean or a variance term, which might be considered a drawback.

The proposed method is based on a fuzzy cluster assignment, that means on parameters, representing the probability of each observation belonging to a certain state. Thus, compared to the original HMM, we have to handle considerably more parameters. This becomes particularly apparent when calculating the parameter update during the GHMM estimation procedure. Since there is no closed form solution for the responsibility update we had to make use of a modified gradient ascent method. This is the reason why we are faced with an enormous increase in the running time compared to the common HMM. Apart from the running time optimisations made in this thesis further improvements are necessary in order to make the method applicable to more complex problems such as finding distinct gene classes that can be characterised by different transcription state sequences.

We were able to show that the GHMM performs as good as the original HMM in two-dimensional cases were the observations are drawn from a Gaussian distribution. However, it performs worse than the ordinary model the higher the number of dimensions gets. In addition, we showed the superiority of our model in cases of nominal emission data. We were able to translate the prior information about the data structure into an appropriate distance measure, which leads to generalised models that are superior to the normal HMM.

Applied to ChIP-chip data we have shown the capability of our method to analyse real data sets. We have identified biologically meaningful transcription states and their corresponding sequential pattern. Our method was able to detect states that are linked to transcription initiation factors, as well as states that are linked to elongation factors. However, it was not able to identify states associated with the transcription termination. One reason for this might be that there is only one factor identified as a termination factor and that the signal is too weak

to distinguish a termination state from the remaining states.

In conclusion, we can say that we have introduced a generalisation of the common hidden Markov model that can improve the performance of the standard HMM when applied to non-Euclidean data.

A. Notation Summary

Symbol	Explanation
K	Number of hidden states in the model.
T	Number of observations (time points) in a sequence.
p	Number of dimensions.
$\mathcal{S} = (s_1, s_2, ..., s_T)$	Hidden state sequence.
$\mathcal{S}^* = (s_1^*, s_2^*, ..., s_T^*)$	Most likely hidden state sequence, estimated with the Viterbi algorithm.
$\mathcal{O} = (O_1, O_2, ..., O_T)$	Observation sequence.
$\pi = (\pi_1, ..., \pi_K)$	Initial state probabilities where $\pi_i = P(s_1 = i)$ and $\sum_{i=1}^{K} \pi_i = 1$.
A	Matrix of transition probabilities where $a_{ij} = P(s_{t+1} = j \mid s_t = i)\ \forall i = 1, ..., K; j = 1, ..., K$ and $\sum_{j=1}^{K} a_{ij} = 1$.
$b_k(O_t)$	Observation probability where $b_k(O_t) = P(O_t \mid s_t = k)$.
R	Matrix of responsibilities where $r_{ij} = P(s_i = j \mid O_i)$ $\forall i = 1, ..., T; j = 1, ..., K$ and $\sum_{j=1}^{K} r_{ij} = 1$.
$R_k = \sum_{t=1}^{T} r_{tk}$	Sum over all observations for a given state k.
$R_{\text{init}}^{\lambda}$	Matrix of initial responsibilities where $r_{ij}^{\text{init},\lambda} = (1 - \lambda) \cdot \gamma_i(j) + \lambda \cdot 1/K$
$\theta = \{\pi, A, R\}$	Parameters of a HMM with generalised emission distribution.
Θ	Set of parameter estimates θ determined during the estimation procedure.
$\alpha_t(j)$	Forward variable of the forward-backward procedure.
$\beta_t(j)$	Backward variable of the forward-backward procedure.
$\gamma_t(j)$	Probability of being in state j at time t
Γ	Matrix of forward-backward terms $\gamma_t(i)$
$\xi_t(i, j)$	Probability of being in state i at time t and state j at time $t + 1$.

Table A.1.: Explanation of the notation used in the thesis (1/2).

Symbol	Explanation
$\delta_t(j)$	Auxiliary variable of the Viterbi algorithm.
$\psi_t(j)$	Values which maximize $\delta_t(j)$ in the Viterbi algorithm.
d_{ij}	Distance measurement between observation i and j e.g. the Euclidean distance $\|\|O_i - O_j\|\|^2$
$\nabla\mathcal{L}(r_{a\cdot})$	Gradient $\frac{\partial}{\partial r_{a\cdot}} Q(\theta, \theta^{\text{old}})$
τ	Step size of gradient ascent (determined through a line search).
C_1	States $k : r_{ak} < 1 - \epsilon,\ \forall k = 1, ..., K$ that can be increased.
C_2	States $k : r_{ak} > \epsilon,\ \forall k = 1, ..., K$ that can be decreased.
w	Binary weight vector of length K that indicates which responsibility should be increased and which should be decrease.
m	State that maximises the partial derivative with respect to $r_{a\cdot}$.
g	State that minimises the partial derivative with respect to $r_{a\cdot}$.
P	Performance measure that is equivalent to the percentage of correctly classified states in a sequence.

Table A.2.: Explanation of the notation used in the thesis (2/2).

B. Additional Figures

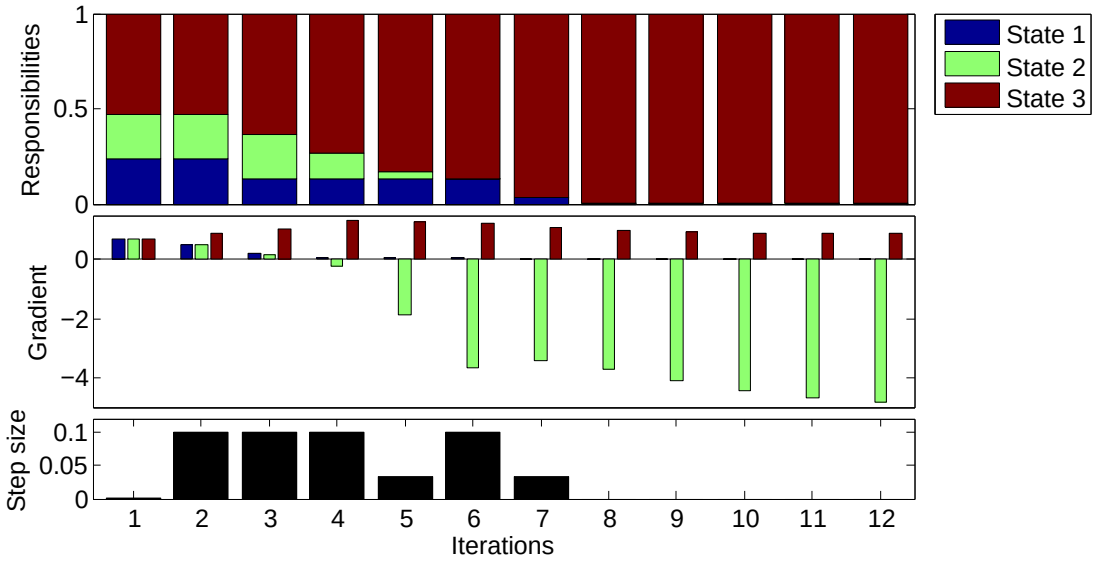

Figure B.1.: Visualisation of the responsibilities $r_{t\cdot}$ (top), the corresponding gradient $\nabla\mathcal{L}(r_{t\cdot})$ (middle) and the optimal step size τ (bottom) for each iteration. The responsibilities are updated in each iteration and converge to one for state three after eight iterations. The remaining iterations are required in order to get converged transition probabilities A and initial state probabilities π. The gradient information in the middle suggests the direction in which the responsibilities should be updated (see also section 2.2.2.3). On the bottom one can see the step size computed with a line search. Note that besides the natural step size restrictions the maximal step is limited to 0.1.

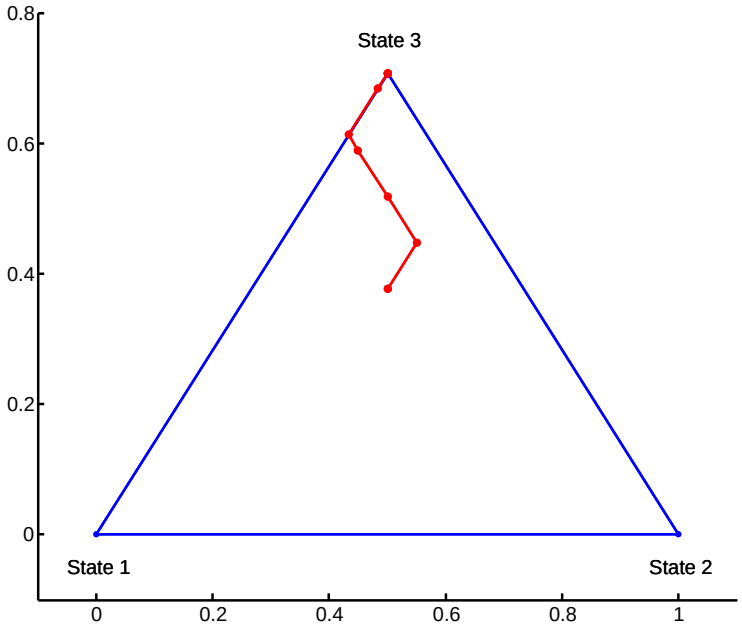

Figure B.2.: Visualisation of the trajectory of the responsibilities $r_{t\cdot}$ shown in figure B.1. The responsibilities are updated during the iterations and are moving from the initialisation towards state three, which is the true state. Note that because of the linear constraint the responsibilities are only updated along the parallel to the simplex boundaries (see also figure 2.6).

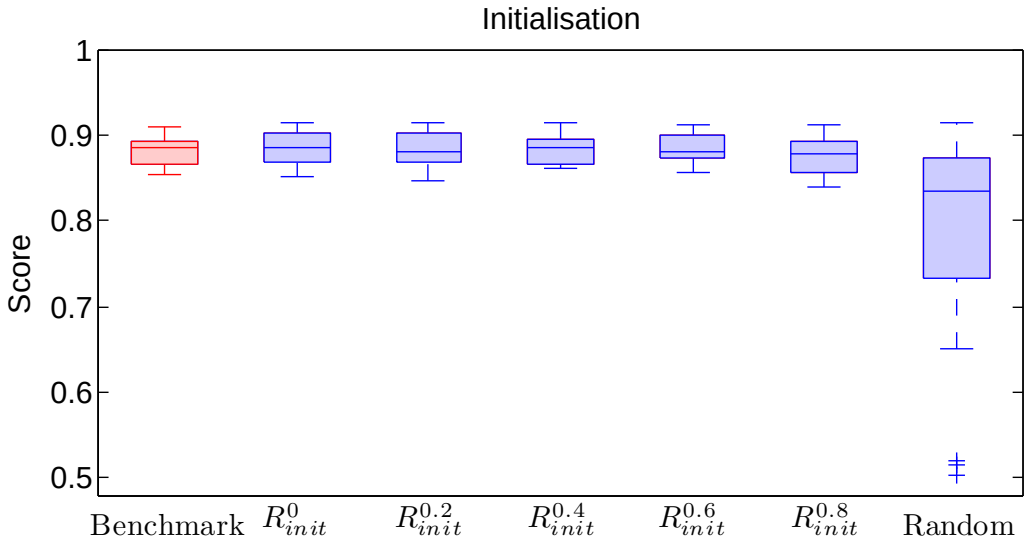

Figure B.3.: Performance of the generalised hidden Markov model under consideration of different responsibility initial values. The simulation is based on scenario 1 but in contrast to the results illustrated in figure 3.1 we used data with cluster centers $\mu = (0, 2)$, in other words with a greater overlap. $R^{\lambda}_{\text{init}}$ again indicates an initialisation of R according to 3.2. The results are based on 20 datasets with 10 sequences per 100 observations, one dimension and two clusters.

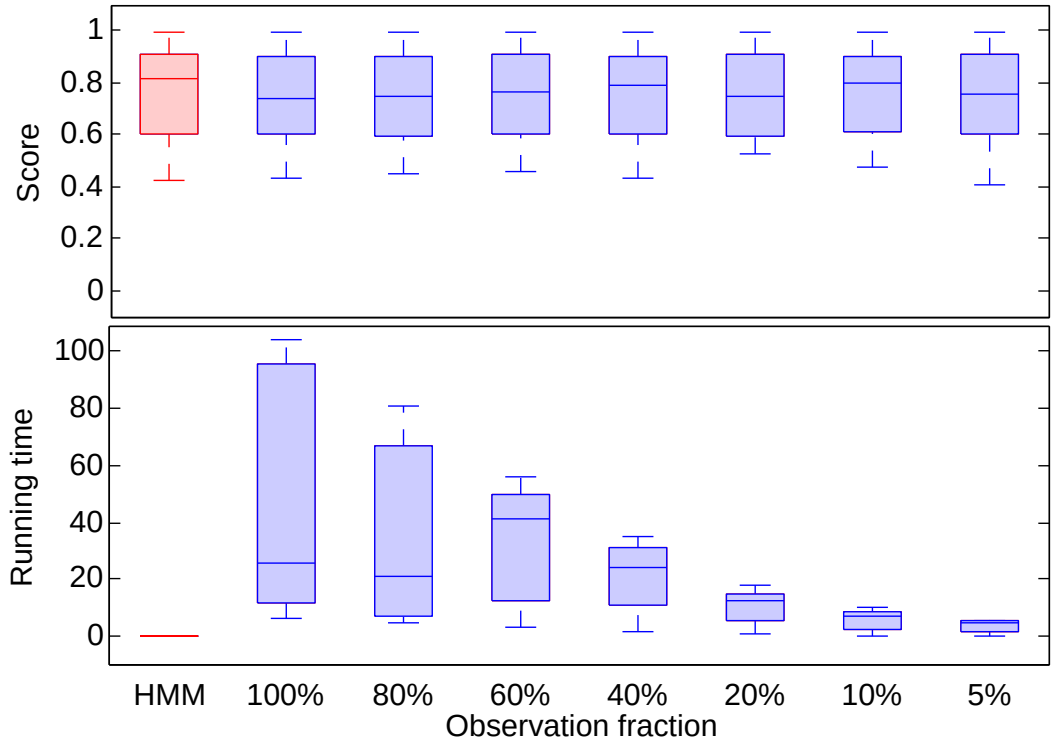

Figure B.4.: Performance of the generalised hidden Markov model under consideration of different observation subsets. The corresponding distribution of the running time in minutes is shown on the bottom. In contrast to figure B.4, the results are based on a responsibility initialisation of $R_{\text{init}}^{0.6}$. The rest of the simulation settings however, are similar to the scenario described in section 3.6.

C. CD-ROM Content

The attached CD-ROM contains the whole Matlab- and R-Code used in this thesis, as well as the ChIP-chip data set, the resulting .mat files, the generated graphics and a digital version of the thesis in hand. A small overview over the content of the included folders is given below:

- **images:** All generated graphics in .pdf and .jpg format.
- **Matlab-Script:**
  - **ChipChip:** Matlab files to analyse the ChIP-chip data.
  - **HMMall:** Matlab functions provided by Murphy (2005).
  - **HMMspecial:** Matlab functions for the GHMM.
  - **Simulations:** Matlab files for the simulations.
  - **workspace:** Results of the simulations.
- **R-Script:** R files and input data.

▷ Readme file in .txt format.

▷ This thesis in .pdf format.

Bibliography

Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occuring in the statistical analysis of probabilistic functions of markov chains. *The Annuals of Mathematical Statistics*, 41(1):164 – 171, 1970.

Jeff A. Bilmes. A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. *International Computer Science Institute*, TR-97-021, 1998.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC, 2006.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., 2001.

Rakesh Dugad and U. B. Desai. A Tutorial On Hidden Markov Models. Technical report, Indian Institute of Technology - Bombay, 1996.

J. C. Dunn. A Fuzzy Relative of the ISODATA Process and its Use in Detecting Compact Well-Seperated Clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

Henrik Failmezger. Automated, statistical analysis of dynamic cellular behaviour for high throughput microscopic time-lapse imaging. Master's thesis, LMU München - TU München - Genzentrum, 2011.

Gernot A. Fink. *Mustererkennung mit Markov-Modellen*. B.G. Teubner, 2003.

G. David Forney. The Viterbi Algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

R.W. Hamming. Error Detecting and Error Correcting Codes. *The Bell System Technical Journal*, 26(2), 1950.

Peter Kampstra. Beanplot: A Boxplot Alternative for Visual Comparison of Distributions. *Journal of Statistical Software, Code Snippets*, 28(1):1–9, 2008. URL `http://www.jstatsoft.org/v28/c01/`.

Louis A. Liporace. Maximum Likelihood Estimation for Multivariate Observations of Markov Sources. *IEEE Transactions on Information Theory*, IT-28(5): 729–734, 1982.

David G. Luenberger and Yinyu Ye. *Linear and Nonlinear Programming*. Springer Verlag, 2008.

MATLAB. The MathWorks Inc., Natick, Massachusetts, R2011a. version 7.12.0.

Andreas Mayer, Michael Lidschreiber, Matthias Siebert, Kristin Leike, Johannes Söding, and Patrick Cramer. Uniform transitions of the general RNA polymerase II transcription complex. *Nature Structural & Molecular Biology*, 17 (10):1272–1278, 2010.

Sören Müller. Improved minimization strategies for the k-means loss function and its non-Euclidean generalizations. Master's thesis, LMU München - TU München - Genzentrum, 2011.

Kevin Murphy. *Hidden Markov Model (HMM) Toolbox for Matlab*, 2005. URL `http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html`.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2009. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

Lawrence R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer Verlag, 1999.

Holger Wunsch. Der Baum-Welch Algorithmus für Hidden Markov Models, ein Spezialfall des EM-Algorithmus. Master's thesis, Universität Tübingen, 2001.

Ming Zheng, Leah O. Barrera, Bing Ren, and Ying Nian Wu. ChIP-chip: Data, Model, and Analysis. Technical report, Ludwig Institute for Cancer Research, UCSD, 2005.

List of Figures

Affidavit

I, Georg Pfundstein, hereby declare that this master-thesis in question was written single-handed and no further as the denounced resources and sources were employed.

Munich, 19th December 2011

Georg Pfundstein

Eidesstattliche Erklärung

Hiermit erkläre ich, Georg Pfundstein, dass es sich bei der vorliegenden Masterarbeit um eine selbständig verfasste Arbeit handelt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt wurden.

München, den 19. Dezember 2011

Georg Pfundstein