Modeling Event-driven Time Series with Generalized Hidden Semi-Markov Models*

Felix Salfner
Department of Computer Science
Humboldt-Universität zu Berlin
salfner@informatik.hu-berlin.de

November 21, 2006

Abstract

This report introduces a new model for event-driven temporal sequence processing: Generalized Hidden Semi-Markov Models (GHSMMs). GHSMMs are an extension of hidden Markov models to continuous time that builds on turning the stochastic process of hidden state traversals into a semi-Markov process. A large variety of probability distributions can be used to specify transition durations.

It is shown how GHSMMs can be used to address the principle problems of temporal sequence processing: sequence generation, sequence recognition and sequence prediction. Additionally, an algorithm is described how the parameters of GHSMMs can be determined from a set of training data: The Baum-Welch algorithm is extended by an embedded expectation-maximization algorithm. Under some conditions the procedure can be simplified to the estimation of distribution moments. A proof of convergence and a complexity assessment are provided.

1 Introduction

Modeling of time series, or temporal sequence processing, is typically accomplished if one of four problems need to be solved[1]:

1. Sequence generation: Having specified a model, generate samples of time series.
2. Sequence recognition: Does some given sequence belong to the typical behavior of the underlying stochastic process or not? More precisely: What is the probability for it?

*This work was supported in part by Intel Corporation and Deutsche Forschungsgemeinschaft DFG project MA 1773 / 3-1

3. Sequence prediction: Assess the probability of the next observation (or state) of the time series.

4. Sequential decision making: Select a sequence of actions in order to achieve some goal or to optimize some cost function.

The majority of models for temporal sequence processing deal with series whose values occur equidistantly (See, e.g., [2] for an overview). However, there are applications that have to deal with time series consisting of values that occur at random points in time. These time series are sometimes also referred to as *event-driven* time series. Examples of applications resulting in event-driven time series include service execution, traffic control applications, customer behavior or traces of events in computer systems. This report introduces a new model for event-driven time series in order to solve the questions stated above: Generalized Hidden Semi Markov Models (GHSMMs).

In this report, only time series consisting of values from a countable finite set (referred to as *symbols*) are considered but the theory can be extended easily to multidimensional real values. Fig. 1 shows

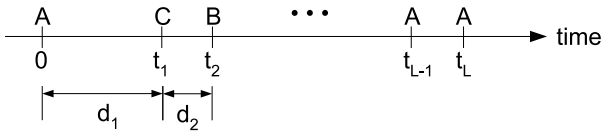

Figure 1: A non-equidistant time series with discrete symbols. The series consists of symbols A, B and C that occur at time $0, \ldots, t_L$. The delay between two successive symbols is denoted by d_k.

an example for an event-driven time series and introduces some basic notation:

- $\{o_1, \ldots, o_M\}$ denotes the set of symbols, which is $\{A, B, C\}$ in the example.
- O_k denotes the symbol that has occurred at time t_k and
- d_k denotes the length of the time interval between two successive symbols O_{k-1} and O_k .

1.1 Three approaches to time handling

Several ways exist to incorporate time of symbol occurrence into models for time series. This report presents three approaches: (1) dividing time into equidistant slots, (2) defining delay symbols, and (3) extending models to continuous time.

Time slots. A straightforward solution to time handling is to define time intervals of fixed length. If there are no events within one time slot, a special symbol denoting "silence" is included. The time series shown in Figure. 2 would translate into the equidistant series "$A\,C\,B\,S\,S\,S\,F$" where S denotes the symbol indicating silence. One advantage of this approach is that any model for equidistant time series can be used since time can be computed from the number of steps. However, there are two problems with this approach:

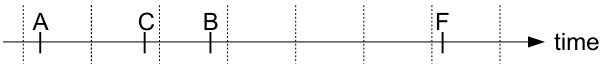

Figure 2: Handling event-driven time series by division of time into slots of fixed length.

1. In most cases, there is some probability that more than one event occurs within one time slot. There are several solutions to this situation including the definition of additional symbols representing combined events, dropping of events or assignment to the next "free" slot – but none of the solutions is really convincing. It is obvious that a reduction of slot width reduces the probability of co-occurrence. On the other hand, small time steps cannot represent long delays appropriately. In general, if the length of inter-symbol intervals varies greatly, this leads to long chains of silence symbols, which deteriorates model quality for most of the models.

2. Time resolution is reduced since it is no longer known when exactly an event has occurred within the time slot. This is true especially for the case of wide time slots.

Delay symbols. A second approach to modeling of event-driven time series is to define a set of delay symbols representing delays of different length. The sequence shown in Fig. 2 could then be represented by "$A\,S_1\,C\,S_1\,B\,S_3\,F$". In comparison to the approach described above, representation of time is improved since chains of silence symbols are avoided. Especially, if silence symbols are representing delays on a logarithmic scale, a wide range of inter-symbol delays can be handled appropriately. Nonetheless, two other disadvantages appear:

1. In most cases the model must be adapted to handle the two different types of values: symbols and delays.

2. Time resolution is worse due to the fact that one symbol accounts for even longer time intervals.

Continuous Time. The third and most elegant approach to handle non-equidistant time intervals in time series is obviously to incorporate *continuous time* into the model itself. A prominent example of such is the extension of discrete time Markov chains (DTMCs) to continuous time Markov chains (CTMCs): Instead of stepping from one state to another, CTMCs can handle transitions of arbitrary durations since transition probabilities are defined by exponential probability distributions $P(t)$.

1.2 Focus of this report

As a result of their extended modeling capabilities, hidden Markov models (HMMs) are becoming more and more frequent in time series modeling. Examples include, e.g., detection of intrusion into computer systems [3], fault diagnosis [4], network traffic modeling [5], speech recognition [6], part-of-speech tagging [7], and genetic sequence finding applications [8]. However, HMMs are based on discrete-time Markov chains (DTMCs) and therefore need to be extended in order to process

event-driven time series. Ignoring the first two approaches to time handling, this report focuses on extending HMMs to continuous time. Specifically, it introduces a new continuous-time HMM that allows to use a large variety of transition distributions and their mixtures. The model is called Generalized Hidden Semi-Markov Model (GHSMM).

The report is organized as follows: In order to provide some background, traditional discrete-time HMMs are briefly introduced in Section 2. In Section 3 HMM extensions that have been published hitherto are recaptured and their properties are discussed. The GHSMM model is introduced in Section 4, and its application to temporal sequence processing is described in Sections 5-7. Deriving model parameters from a set of training data is described in Section 8, followed by a proof of convergence in Section 9 and complexity considerations in Section 10.

2 A Short Introduction to Discrete-time HMMs

Traditional HMMs are based on discrete-time Markov chains (DTMC), which consist of a set $\mathscr{S} = \{s_i\}$ of N states and a square matrix $\boldsymbol{A} = [a_{ij}]$ defining transition probabilities between the states. $\boldsymbol{A}$ is a stochastic matrix, from which follows that

$$\sum_{j=1}^{N} a_{ij} = 1 \tag{1}$$

Additionally, a vector of initial state probabilities $\boldsymbol{\pi} = [\pi_i]$ has to be specified. Again,

$$\sum_{i=1}^{N} \pi_i = 1 \tag{2}$$

must be fulfilled.

The stochastic process of DTMCs can be described as follows: An initial state is chosen according to the probability distribution defined by $\boldsymbol{\pi}$. Starting from the initial state, the process transits from one state to the next according to the transition probabilities defined by $\boldsymbol{A}$. This notion corresponds to the so-called *Markov assumptions*:

1. The process is memoryless: a transition's destination is dependent only on the current state irrespective of the states that have been visited previously.
2. The process is time-homogeneous: transition probabilities $\boldsymbol{A}$ stay the same regardless of the time that has already elapsed ($\boldsymbol{A}$ is not depending on time t)

More formally, both assumptions can be expressed by the following equation:

$$P(S_{t+1} = s_j \mid S_t = s_i, \ldots, S_0) = P(S_1 = s_j \mid S_0 = s_i) \tag{3}$$

Loss of memory is expressed by the fact that all previous states $S_0, \ldots, S_{t-1}$ of the left-hand side are ignored and time-homogeneity is reflected by the fact that the transition probability are equal for time $t \rightarrow t+1$ and $0 \rightarrow 1$.

HMMs extend the concept of DTMCs. At each time step an output (or observation) is generated according to a probability distribution. The key notion is that this output probability distribution depends on the state the stochastic process of the DTMC is in. Two types of HMMs can be distinguished regarding the types of their outputs. If the output is chosen from some finite countable set, outputs are called *symbols*. If the output is continuous, e.g, a vector of real numbers, the model is called continuous HMM.[1] As stated in the introduction, this report deals only with discrete symbols and does not consider continuous HMMs.

To formalize the extension, HMMs additionally define a finite countable set of symbols $\mathscr{O} = \{o_i\}$ of M different symbols and a stochastic matrix $\boldsymbol{B} = [b_{ij}]$. Each row i of $\boldsymbol{B}$ defines a probability distribution for state s_i such that b_{ij} is the probability for emitting symbol o_j given that the stochastic process is in state s_i. Hence, $\boldsymbol{B}$ has dimensions $N \times M$ and

$$\sum_{j=1}^{M} b_{ij} = 1 \tag{4}$$

Please note that for readability reasons, b_{ij} will sometimes be denoted by $b_i(o_j)$. Figure 3 shows a simple discrete-time HMM.

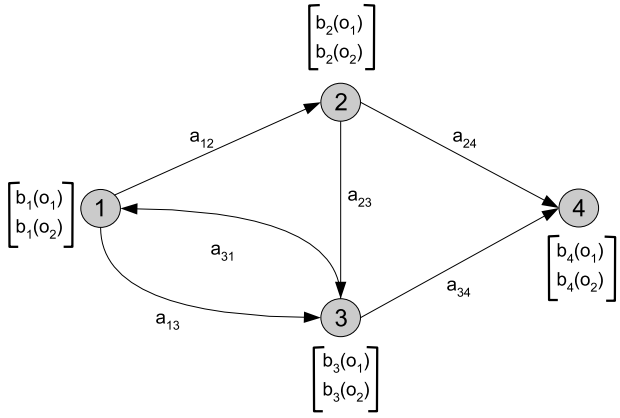

Figure 3: A simple discrete-time HMM with $N = 4$ states and $M = 2$ output symbols.

The fact that HMMs are called "hidden" refers to the notion that only the outputs can be measured from outside and the state of the stochastic DTMC process is hidden from the observer. The majority of theory about HMMs is concerned with this property. For example, efficient algorithms have been developed to find the most probable sequence of DTMC states when some sequence of output symbols has been observed. Other algorithms address the problem of finding optimal HMM parameters $\lambda = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi})$ when a set of observation sequences is given (and nothing is known about the states the DTMC has travelled through). The most well-known algorithms will be described in the following sections.

[1] Not to be confused with continuous-**time** HMMs as defined in the next section

2.1 Forward-backward algorithm

The probability that a given HMM with parameters $\lambda = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi})$ has generated some observation sequence $\boldsymbol{o} = (O_t)$ is called *sequence likelihood* and is denoted by $P(\boldsymbol{o}|\lambda)$. If the sequence of hidden states that generated the observation sequence was known, the likelihood could be computed by:

$$P(\boldsymbol{o}, \boldsymbol{s}|\lambda) = \pi_{S_0}\, b_{S_0}(O_0) \prod_{t=1}^{L} a_{S_{t-1}\,S_t}\, b_{S_t}(O_t) \tag{5}$$

where $\boldsymbol{s} = [S_t]$ denotes the sequence of hidden states and L is the length of the sequence. However, as only $\boldsymbol{o}$ is known, all possible state sequences $\boldsymbol{s}$ have to be considered:

$$P(\boldsymbol{o}|\lambda) = \sum_{\boldsymbol{s}} \pi_{S_0}\, b_{S_0}(O_0) \prod_{t=1}^{L} a_{S_{t-1}\,S_t}\, b_{S_t}(O_t) \tag{6}$$

which results in intractable complexity. Nevertheless, an efficient reformulation has been found exploiting the Markov assumption that the transition probability only depends on the current state. Using this property, Equation 6 can be rearranged such that repetitive computations can be grouped together. From this rearrangement it is only a small step to a recursive formulation, which is also known as dynamic programming approach. The resulting algorithm is called *forward algorithm*.

Forward algorithm. The algorithm is based on a forward variable $\alpha_t(i)$ denoting the probability of subsequence $(O_0 \dots O_t)$ and the fact that the stochastic process is in state i at time t:

$$\alpha_t(i) = P(O_0\, O_1\, \dots\, O_t, S_t = s_i|\lambda) \tag{7}$$

$\alpha_t(i)$ can be computed by the following recursive computation scheme:

$$\alpha_0(i) = \pi_i\, b_i(O_0) \tag{8}$$

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i)\, a_{ij}\, b_j(O_t); \quad 1 \le t \le L \tag{9}$$

The algorithm can be visualized by a so-called *Trellis* structure as shown in Figure 4. Each node is representing one $\alpha_t(i)$ while edges visualize the terms of the sum in Equation 9.

As $\alpha_L(i)$ is the probability of the entire sequence and the fact that the stochastic process is in state i at the end of the sequence, sequence likelihood $P(\boldsymbol{o}|\lambda)$ can be computed by:

$$P(\boldsymbol{o}|\lambda) = \sum_{i=1}^{N} \alpha_L(i) \tag{10}$$

Backward algorithm. A backward variable $\beta_t(i)$ can be defined in a similar way, denoting the probability of the rest of the sequence $(O_{t+1} \dots O_L)$ given the fact that the stochastic process is in state i at time t:

$$\beta_t(i) = P(O_{t+1}\, \dots\, O_L|S_t = s_i, \lambda) \tag{11}$$

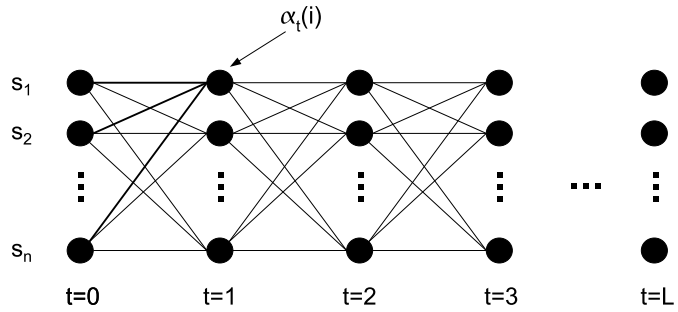

Figure 4: A trellis structure visualizing $\alpha_t(i)$. Edges indicate the terms that have to be summed up in the forward algorithm (see Equation 9).

$\beta_t(i)$ can be computed in a similar recursive way by:

$$\beta_L(i) = 1 \tag{12}$$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j); \quad 1 \le t \le L \tag{13}$$

Forward-backward algorithm. Combining both $\alpha_t(j)$ and $\beta_t(i)$ leads to another probability that is of interest in time series modeling: the probability of being in state s_i at time t given an observation sequence $\boldsymbol{o}$. This probability is denoted by

$$\gamma_t(i) = P(S_t = s_i|\boldsymbol{o}, \lambda) \tag{14}$$

Some computations yield:

$$P(S_t = s_i|O_0 \ldots O_L, \lambda) = \frac{P(S_t = s_i, O_0 \ldots O_L \,|\, \lambda)}{P(O_0 \ldots O_L \,|\lambda)} \tag{15}$$

$$= \frac{P(S_t = s_i, O_0 \ldots O_t\, O_{t+1} \ldots O_L \,|\, \lambda)}{P(O_0 \ldots O_L \,|\lambda)} \tag{16}$$

$$= \frac{P(S_t = s_i, O_0 \ldots O_t \,|\, \lambda)\; P(O_{t+1} \ldots O_L \,|S_t = s_i, \lambda)}{P(O_0 \ldots O_L \,|\lambda)} \tag{17}$$

$$= \frac{\alpha_t(i)\; \beta_t(i)}{P(O_0 \ldots O_L \,|\lambda)} \tag{18}$$

Hence $\gamma_t(i)$ can be computed by:

$$\gamma_t(i) = \frac{\alpha_t(i)\, \beta_t(i)}{P(\boldsymbol{o} \,|\, \lambda)} = \frac{\alpha_t(i)\, \beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\, \beta_t(i)} \tag{19}$$

Viterbi algorithm. In order to find the most probable *sequence* of states the straightforward solution would be to select the most probable state at each time step t:

$$S_{max}(t) = \arg\max_i \gamma_t(i) \tag{20}$$

However, it turns out that models exist for which one of the transitions from $S_{max}(t)$ to $S_{max}(t+1)$ is not possible (the transition probability a_{ij} equals zero). This is due to the fact that α and β both sum up all possible paths between the states of the DTMC – and γ is only the product of α and β.

The solution to compute the most probable state sequence is called *Viterbi algorithm*. Very similar to $\alpha_t(i)$, let $\delta_t(i)$ denote the probability of the most probable state sequence for the subsequence of observations $(O_0 \dots O_t)$ that ends in state s_i:

$$\delta_t(i) = \max_{S_0 \dots S_{t-1}} P(O_0 \dots O_t,\ S_0 \dots S_{t-1}, S_t = s_i|\lambda) \tag{21}$$

$\delta_t(i)$ can be computed by a slight modification of the forward algorithm using maximum values instead of the sum over all states:

$$\delta_0(i) = \pi_i\, b_i(O_0) \tag{22}$$

$$\delta_t(j) = \max_{1 \le i \le N} \delta_{t-1}(i)\, a_{ij}\, b_j(O_t); \quad 1 \le t \le L \tag{23}$$

In order to find the most sequence of probable states each state selected by the maximum operator has to be stored in a separate array. The sequence can then be found by tracing backwards through the array starting from state $\arg\max_i \delta_L(i)$.

2.2 Training of discrete-time HMMs

In the forward-backward algorithm, the model's parameters λ were assumed to be fixed and known. However, in the majority of applications, λ cannot be inferred analytically but need to be estimated from recorded sample data. In the machine learning community, such a procedure is called *training*. Several algorithms exist for HMM training, of which the *Baum-Welch* algorithm is most prominent. This sections introduces the standard Baum-Welch algorithm.

Let's first introduce formulas for parameter estimation if the sequence of "hidden" states is known. This scenario occurs, e.g., in part-of-speech tagging applications:

- Initial state probabilities π_i are determined by the relative frequency of sequences starting in state s_i:

$$\overline{\pi}_i = \frac{\text{number of sequences starting in } s_i}{\text{total number of sequences}} \tag{24}$$

- Transition probabilities a_{ij} are determined by the number of times the process went from state s_i to state s_j divided by the number of times, the process left state s_i to anywhere:

$$\overline{a_{ij}} = \frac{\text{number of transitions } s_i \to s_j}{\text{number of transitions } s_i \to ?} \tag{25}$$

- Emission probabilities $b_i(o_j)$ are determined by the number of times the process has generated symbol o_j in state s_i compared to the number of times the process has been in state s_i:

$$\overline{b_i(o_j)} = \frac{\text{number of times symbol } o_j \text{ has been emitted in state } s_i}{\text{number of times the process has been in state } s_i} \tag{26}$$

However, in many applications the hidden data is not known. The solution found by Baum and Welch introduced expectation values for the unknown quantities. The algorithm belongs to the class of Expectation-Maximization (EM) algorithms. EM algorithms employ an iterative scheme for lower bound maximization of the overall data likelihood. The rationale behind the algorithm is described in detail in section 9. At this point it is sufficient to state that the algorithm consists of two major steps:

1. E-step: Compute estimations for the hidden data given a fixed set of model parameters.
2. M-step: Adjust model parameters to maximize data likelihood.

This scheme is repeated until data-likelihood converges to some optimum value. Convergence to a local optimum can be proven (see section 9).

E-Step In order to compute expectation values for the E-step of the algorithm, another quantity has to be introduced: the probability of a transition from state s_i to s_j at time t, given some observation sequence $\boldsymbol{o}$:

$$\xi_t(i,j) = P(S_t = s_i, S_{t+1} = s_j | \boldsymbol{o}, \lambda) \tag{27}$$

Starting from $\gamma_t(i)$, it can be derived similar to Equations 15–18 by interposing the transition from s_i to s_j between α and β:

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}{P(\boldsymbol{o}|\lambda)} \tag{28}$$

$$= \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)} \tag{29}$$

This approach is visualized in Figure 5.

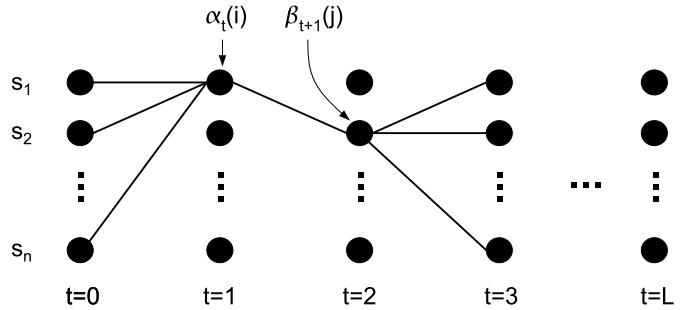

Figure 5: A trellis structure visualizing the computation of $\xi_t(i,j)$ (see Equation 29).

The expected number of transitions from state s_i to s_j can be obtained by summing over time t:

$$\sum_{t=0}^{L-1} \xi_t(i,j) \tag{30}$$

M-step The second step of the Baum-Welch algorithm is a maximum likelihood optimization of HMM parameters that uses expected values for the unknown data. As it is unknown which state the process has travelled through, expectation values are substituted into Equations 24–26:

$$\bar{\pi}_i \equiv \frac{\text{expected number of sequences starting in state } s_i}{\text{total number of sequences}} \equiv \gamma_0(i) \tag{31}$$

$$\bar{a}_{ij} \equiv \frac{\text{expected number of transitions } s_i \to s_j}{\text{expected number of transitions } s_i \to ?} \equiv \frac{\sum_{t=0}^{L-1} \xi_t(i,j)}{\sum_{t=0}^{L-1} \sum_{j=1}^{N} \xi_t(i,j)} \tag{32}$$

$$\bar{b}_i(k) \equiv \frac{\text{expected number of times observing } o_k \text{ in state } s_i}{\text{expected number of times in state } s_i} \equiv \frac{\sum_{t=0}^{L-1}{}_{s.t.\ O_t=o_k} \gamma_t(i)}{\sum_{t=0}^{L-1} \gamma_t(i)} \tag{33}$$

Please note that summing up $\xi_t(i,j)$ over all destination states s_j yields the probability for the source state s_i at time t. Hence, $\xi_t(i,j)$ can be related to $\gamma_t(i)$:

$$\sum_{j=1}^{N} \xi_t(i,j) = \gamma_t(i) \tag{34}$$

Using this Equation, Equation 32 could be simplified. But since this relation does not hold for the extension to continuous time, substitution has been omitted for the sake of comparability to equations presented in Section 8.

EM algorithm. The overall EM algorithm starts from an initialized and hence completely defined HMM. As much a-priori knowledge as possible should be used to get reasonable initial values for the parameters. If this is not possible, simply random initialization can be used. After initialization, E and M-steps are executed repeatedly until some level of convergence has been reached.[2] In each E-step the parameter estimations of the previous M-step are used.

2.3 Multiple Training Sequences

The formulas presented here have only considered one single observation sequence. In most applications, there will be a large set of training sequences. The main idea of multiple sequence training

[2]For implementation, a maximum number of iterations is often used as an additional stopping criterion.

is that nominators and denominators of Equations 31 to 33 are transformed into a weighted sum over sequences. The weight for each sequence $\boldsymbol{o}_k$ is determined by its sequence likelihood $P(\boldsymbol{o}_k \mid \lambda)$ computed along with the E-step of the algorithm.

3 Previous Extensions to Continuous Time

Discrete-time HMMs have been applied very successfully to several application areas in which time is not considered (e.g., gene sequence analysis) or to areas in which measurements/symbols occur periodically (e.g., symbols acquired by periodic probing). Discrete-time HMMs have also been applied to application domains in which the temporal behavior of the underlying stochastic process is important, but the need for a computationally feasible solution forced the usage of discrete-time models. For example in speech recognition, phoneme durations vary statistically, which should be reflected by the model in such a way that the underlying stochastic process stays longer in some states than in others. First modeling approaches have ignored this fact and have used discrete-time HMMs due to the fact that speech-recognition is a real-time task and computers were not powerful enough in the early 80s to perform computations quick enough. However, in [9] Russell and Cook showed that a switch to continuous time models can improve modeling performance significantly. Despite speech recognition, continuous-time HMMs have been successfully applied to other applications as well (see, e.g., [10]). This section briefly introduces various extensions to continuous time that have been published and spells out what makes GHSMM a unique approach.

Self transitions in discrete-time HMMs. As described in Section 1, the simplest way to handle event-driven time series is to descretize time by defining time slots of length Δ and to handle time variance (i.e., delays) as multiples of Δ. When applying this approach to discrete-time HMMs, delays are represented by self-transitions: In each time step, there is some probability that the stochastic process transits to itself and hence stays in the state (see Figure 6).

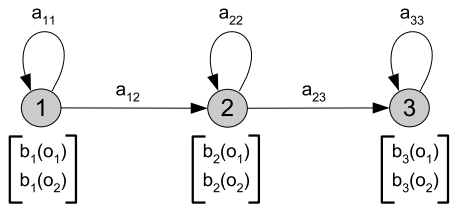

Figure 6: Duration modeling by a discrete-time HMM with self-transitions.

This approach leads to a geometric distribution for state sojourn times, since the probability to stay in state s_i for d time-steps of duration Δ equals

$$P_i(D = d) = a_{ii}^{d-1} \, (1 - a_{ii}) \tag{35}$$

CT-HMMs. A similar approach has been proposed in [5]. Here, the embedded time-discrete Markov chain is replaced by a continuous-time Markov chain (CTMC). The resulting model is abbreviated by CT-HMM and may not be confused with continuous HMMs (CHMMs), which are discrete-time HMMs with *continuous output* probability densities. CTMCs are determined by an initial distribution and an infinitesimal generator matrix Q. Determination of the infinitesimal generator matrix Q follows a two-step approach: First, a transition matrix $P(\Delta)$ and the initial distribution are estimated by Baum-Welch training from the training data and then Q is obtained by Taylor expansion of the equation

$$Q = \frac{1}{\Delta} \ln(P) \tag{36}$$

which is directly derived from Kolmogorov's equations (see, e.g., [11]). Δ denotes again some minimal delay (a time step) as in the previous approach.

Models such as CT-HMMs still imply strong assumptions about the underlying stochastic process since CTMCs are based on time-homogeneous, memoryless, exponential distributions. A more powerful approach towards continuous-time HMMs is to substitute the underlying DTMC by a semi-Markov process (SMP). Resulting models are called Hidden Semi-Markov Models (HSMMs). SMPs are described in more detail along with the introduction of GHSMMs in Section 4. For the time being it is sufficient to remember that SMPs allow to use arbitrary probability distributions in order to characterize state transition behavior.

A first approach to HSMMs is to substitute the self-transitions as in Figure 6 by state durations that follow a state-specific probability distribution $p_i(d)$ as depicted in Figure 7. Several solutions

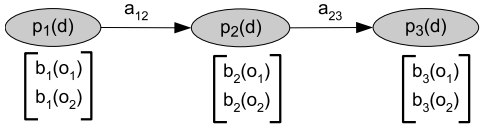

Figure 7: Duration modeling by explicit modeling of state durations.

have been developed to explicitly specify and determine $p_i(d)$ from training data along with the Baum-Welch algorithm.

Ferguson's model. One of the first approaches to explicit state duration modeling was proposed by Ferguson in [12][3]. The idea was to use a discrete probability distribution for $p_i(d)$. While the approach was very flexible, it showed three disadvantages: first, it is a discrete-time model requiring the definition of a time step Δ and a maximum delay D, second, convergence of the training algorithm was insufficiently slow, and third, much more training data was needed for training. The last two drawbacks result from a dramatically increased number of parameters that have to be estimated from the training data: The number of parameters increases from N self-transitions to $N \times D$ duration probabilities.

[3] A crisp overview can be found in [13].

HSMMs with Poisson-distributed durations. Ferguson proposed to use parametric distributions instead of discrete ones. Russell and Moore [14] have used Poisson distributions. A comparison of both models showed that the models perform better if there is an insufficient amount of training data available [9].

HSMMs with gamma-distributed durations. Levinson provided a maximum likelihood estimation for parameters of gamma-distributed durations [15]. As it is the case with most maximum likelihood procedures, optimal parameters are obtained by derivation of the likelihood function. However, this derivative cannot be computed explicitly and numerical approximation has to be applied.

HSMMs with durations from the exponential family. Mitchell and Jamieson [16] extended the spectrum of available distributions for explicit duration modeling to all distributions of the exponential family, which includes gamma distributions. Their work is also founded on a direct computation of maximum likelihood involving numerical approximation of the maximum.

HSMMs with Viterbi path constrained uniform distributions. The authors of [17] present an approach where transition durations are assumed to be uniformly distributed. Their key idea is that first parameters $\boldsymbol{\pi}$, $\boldsymbol{A}$ and $\boldsymbol{B}$ are obtained by the discrete-time HMM reestimation procedure. A subsequent step involves computation of Viterbi paths for the training data in order to identify minimum and maximum durations for each transition: this defines a uniform duration distribution for each transition.

Expanded State HMMs (ESHMMs). In parallel to the development of HSMMs with parameterized probability distributions, it has been found that Ferguson's model can be implemented in a much easier way by a series-parallel topology of the hidden states [18]. To be precise, each state of the HMM is replaced by a DTMC sharing the same emission probability distribution. State durations are then expressed by the probabilities of the DTMC (Figure 8 shows a small example for a HMM with left-to-right topology). Those models are referred to by Expanded State HMMs (ESHMMs).

ESHMMs allow to use implementations of standard discrete-time HMMs. Furthermore, the idea to represent state durations by state chains led to several variants of the duration structure, that extend Ferguson's model. For example, the duration structure may have self-transitions that allow to model durations of arbitrary length instead of a fixed maximum duration D. Some structures have been proposed in [19] and [20] and a comparison of two extended structures is provided in [9]. More elaborate training algorithms for ESHMMs have been proposed in [21] and [22].

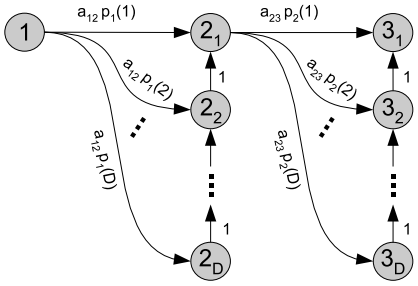

Figure 8: Topology of an Expanded State HMM (ESHMM). Emission probabilities $b_i(o_j)$ have been omitted.

Inhomogeneous HMMs (IHMMs). Ramesh and Wilpon have developed another variant of HMMs, called Inhomogeneous HMM (IHMM) [23]. Time homogeneity of stochastic processes refers to the property that the behavior (i.e., the probability distributions) do not change over time. In terms of Markov chains, it means that the transition probabilities a_{ij} are constant and not a function of time. However, the authors abandon this assumption and define:

$$a_{ij}(d) = P(S_{t+1} = j | S_t = i, d_t(i) = d); \quad 1 \leq d \leq D \tag{37}$$

which is the transition probability from state s_i to state s_j given that the duration in state s_i at time $t - d_t(i) = d$ – equals d. In order to define a proper stochastic process, the transition probabilities must satisfy:

$$\forall d \in \{1, \ldots, D\} : \quad \sum_{j=1}^{N} a_{ij}(d) = 1 \tag{38}$$

As can be seen from the formulas, Ramesh and Wilpon also assume discretized time and a maximum state duration D.

What is new about GHSMM. Generalized hidden semi-Markov models extend the models presented above in four aspects:

1. GHSMMs allow to use *a variety of parametric transition probability distributions*. More specifically, each transition can be modeled by a mixture of kernel distributions. Almost any type of parametric probability distribution can be used for each kernel. However, since parameter estimation is performed on a maximum likelihood basis, distributions should be preferred for which the maximum likelihood equations can be solved analytically or for which good analytical approximations exist. This includes well-known distributions such as Gaussian, exponential, gamma, Pareto, or log-normal.

2. GHSMMs model *transition durations* rather than state durations. Widely used state durations are a special case where all transitions are equally distributed. However, GHSMMs are not the first model that use transition durations, they have also been proposed in, e.g., [24].

3. GHSMMs operate on true *continuous time* instead of multiples of a minimum delay Δ. This feature is especially important for event-driven time series modeling.

4. There is no *no maximum duration D* in GHSMMs.

The first extension is based on the usage of an embedded EM algorithm for estimation of transition probability parameter estimation. Since EM algorithms perform lower bound maximization (see [25]), more complex problems can be solved which translates into a solution for more complex transition distributions.

The second extension results from a strict application of the theory of semi-Markov processes. As main developments for HMMs have been achieved in the area of speech recognition, state duration modeling has been sufficient. However, for other applications in temporal sequence processing, this is not always the case.

Regarding the last two extensions, many approaches have introduced time discretization together with some maximum delay. Both restrictions were used to enable a dynamic programming approach, i.e, forward-backward and Viterbi algorithm. Although the mathematical restrictions are inescapable, GHSMM apply efficient algorithms by a stricter enforcement of the Markov assumptions.

4 Generalized Hidden Semi-Markov Models (GHSMMs)

GHSMMs are HMMs that are based on a continuous-time semi-Markov process. They allow to define durations for each transition of the embedded Markov chain and to use a large variety of duration distributions. In this section, the theory of semi-Markov processes is introduced briefly and applied to HMMs. Formulas for the forward-backward algorithm, Viterbi algorithm and training are developed along with the principle tasks of temporal sequence processing in Sections 5-8.

4.1 Wrap-up of Semi-Markov Processes

Semi-Markov processes (SMP) are a continuous-time extension to Markov renewal sequences, which are defined as follows (see, [26]):
A sequence of bivariate random variables $\{(Y_n, T_n)\}$ is called a **Markov renewal sequence** if

1. $T_0 = 0,\ T_{n+1} \geq T_n; \quad Y_n \in \mathscr{S}, \qquad$ and
2. $\forall n \geq 0 :\ P(Y_{n+1} = j,\ T_{n+1} - T_n \leq t | Y_n = i,\ T_n,\ \ldots,\ Y_0,\ T_0) = P(Y_1 = j,\ T_1 \leq t | Y_0 = i) \quad (39)$

where random variables T_n denote time and Y_n denote the state of the n-th element in the Markov renewal sequence. Please note that T_n refer to instants on a continuous time scale and t is the length of the interval between T_{n+1} and T_n. Furthermore, Equation (39) expresses that Markov renewal sequences are memoryless and time-homogeneous: As the transition probability is only depending on the immediate predecessor, it has no memory of the states the process has travelled

through, and since transition probability at time n is equal to the probability at time 0, the process is time-homogeneous.

Let $g_{ij}(t)$ denote the conditional probability that state s_j follows s_i after time t as defined by Equation (39). Then the matrix $\boldsymbol{G}(t) := [g_{ij}(t)]$ is called *the kernel of the Markov renewal sequence.* Please note that $g_{ij}(t)$ has all properties of a cumulative probability distribution except that the limiting probability p_{ij} can be less than 1:

$$p_{ij} := \lim_{t\to\infty} g_{ij}(t) = P(Y_1 = j | Y_0 = i) \leq 1 \tag{40}$$

Even if Markov renewal sequences are defined on a continuous time scale, they form a discrete sequence of points. If the gaps between the points of a Markov renewal sequence are filled, a Semi-Markov process (SMP) is obtained. More formally:
A continuous-time stochastic process $\{X(t), t \geq 0\}$ with countable state space $\mathscr{S}$ is said to be a **semi-Markov process** if

1. it has piecewise constant, right continuous sample paths, and
2. $\{(Y_n, T_n), n \geq 0\}$ is a Markov renewal sequence, where T_n is the n-th jump epoch and $Y_n = X(T_n+)$

which means that the state X of the SMP at any time t is defined by the state Y_n of the embedded Markov renewal sequence where n is determined such that it is the largest index for which $T_n \leq t$. Figure 9 visualizes the concept.

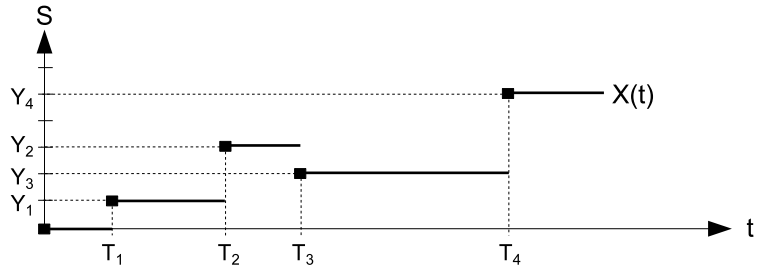

Figure 9: A semi-Markov process $X(t)$ defined by a Markov renewal sequence $\{(Y_n, T_n)\}$

The term "semi-Markov" stems from the fact that the process X itself is not a Markovian process since properties of the T_n process depend on the state of Y_n. Yet, the associated process (Y_n, T_n) is Markovian.

A SMP is called *regular*, if it only performs a finite number of transitions in a finite amount of time. As this report only considers regular SMPs, the term "regular" will be omitted from now on.

Please note that the limits defined in Equation 40 "eliminate" time behavior. By this, p_{ij} define a DTMC that is said to be *embedded* in the SMP. From this analogy it is clear that the following

property holds for each transient state s_i:

$$\sum_{j=1}^{N} p_{ij} = 1 \tag{41}$$

expressing the fact that it is sure that the SMP leaves state s_i if time d approaches infinity.

In addition to the notion of the embedded DTMC, the limiting probabilities p_{ij} can be used to define a quantity that helps to understand the way how SMPs operate. Let $d_{ij}(t)$ denote a probability distribution for the duration of a transition from state s_i to state s_j:

$$d_{ij}(t) = P(T_1 \leq d \,|\, Y_0 = i,\, Y_1 = j) \tag{42}$$

Using the limiting probabilities, $d_{ij}(t)$ can be computed from $g_{ij}(t)$ the following way:

$$d_{ij}(t) = \begin{cases} \frac{g_{ij}(t)}{p_{ij}} & \text{if } p_{ij} > 0 \\ 1 & \text{if } p_{ij} = 0 \end{cases} \tag{43}$$

Therefore, $g_{ij}(t)$ can be split into a transition probability and a transition duration distribution:

$$g_{ij}(t) = p_{ij}\ d_{ij}(t) \tag{44}$$

which leads to an intuitive description of the behavior of SMPs: Assume that at time 0 the system enters state i. It then chooses the next state to visit to be j according to probability p_{ij}. Having decided upon the next state to be j, it stays in state i for a random amount of time sampled from distribution $d_{ij}(t)$ before it enters state j. Once the SMP enters state j it looses all memory of the history and behaves as before, starting from state j. Note that the theory of SMPs allow $p_{ii} \neq 0$, i.e., the SMP may return to state i immediately after leaving it. However, for simplicity reasons, it will be assumed further on that $p_{ii} = 0$.

Finally, it should be noted that SMPs are fully specified by two quantities:

1. the initial distribution $\boldsymbol{\pi} = [\pi_i] = [P(X(0) = i)]$
2. the kernel $\boldsymbol{G}(t)$ of the underlying Markov renewal sequence. Alternatively, $\boldsymbol{G}(t)$ can be specified by $\boldsymbol{P} = [p_{ij}]$, which is a transition matrix for the embedded DTMC, and a matrix $\boldsymbol{D}(t) = [d_{ij}(t)]$ defining probability distributions for the duration of each transition from s_i to s_j.

Be aware that the matrix multiplication $\boldsymbol{G}(t) = \boldsymbol{P} \times \boldsymbol{D}(t)$ is not correct since Equation 44 only holds for each $g_{ij}(t)$ separately.

4.2 GHSMMs: Combining Semi-Markov Processes with Hidden Markov Models

Generalized hidden semi-Markov models (GHSMMs) are hidden Markov models that use a semi-Markov process (SMP) instead of a discrete-time Markov chain (DTMC) to model the stochastic process of hidden state transitions. A simple example is provided in Figure 10.

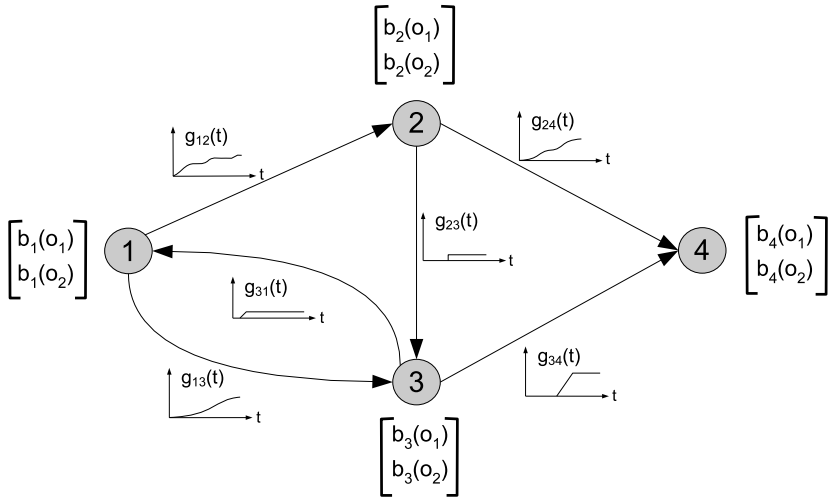

Figure 10: A simple GHSMM example.

The example shown in Figure 10 is identical to the discrete-time example shown in Figure 3 except that the the transition probabilities a_{ij} have been replaced by cumulative probability distributions $g_{ij}(t)$. From Equation 41 follows that for each transient state the limiting values p_{ij} – which are the upper bounds of each $g_{ij}(t)$ as defined in Equation 40 – of all outgoing edges sum up to 1.

As shown by Equation 44, $g_{ij}(t)$ can be represented as a product of p_{ij} and a distribution characterizing the duration of the transition from state s_i to s_j. In order to specify transitions, GHSMMs use a mixture of probability distributions:

$$d_{ij}(t) = \sum_{r=0}^{R} w_{ij,r}\, \kappa_{ij,r}(t|\theta_{ij,r}) \tag{45}$$

$$\text{s.t.} \sum_{r=0}^{R} w_{ij,r} = 1 \tag{46}$$

where $w_{ij,r}$ are weights that sum up to 1 and $\kappa_{ij,\,r}$ are so-called *kernels*. Each kernel $\kappa_{ij,\,r}(t|\theta_{ij,r})$ is a cumulative probability distribution with parameters $\theta_{ij,r}$. For example, if $\kappa_{ij,\,r}$ is a Gaussian kernel, $\theta_{ij,r}$ consists of mean $\mu_{ij,r}$ and variance $\sigma^2_{ij,r}$. Furthermore, one kernel, say $\kappa_{ij,0}$, can model some *background distribution* (e.g., a uniform distribution) in order to account for "delay noise" meaning delays that do not follow the characteristics modeled by $d_{ij}(t)$. For kernels $\kappa_{ij,0}$, any type of parametric probability distribution can be used. However, since parameter estimation is performed on a maximum likelihood basis, distributions should be preferred for which the maximum likelihood equations can be solved analytically. This includes well-known distributions such as Gaussian, exponential, Pareto, or log-normal.

As it is the case for the entire report, only GHSMMs with discrete emission probabilities $\boldsymbol{B}$ are considered here. Nevertheless, the approach could be extended easily to continuous, multimodal outputs (see [13, 27, 28] for a summary how it is done for discrete-time HMMs). Additionally as stated above, it is assumed, that $p_{ii} = 0$, expressing the fact that there are no self transitions in the

model.

In summary, a GHSMM is completely defined by

- The set of states $\mathscr{S} = \{s_1, \ldots, s_N\}$
- The set of symbols $\mathscr{O} = \{o_1, \ldots, o_M\}$
- The N-dimensional initial state probability vector $\boldsymbol{\pi}$
- The $N \times M$ matrix of emission probabilities $\boldsymbol{B}$
- The $N \times N$ limiting transition probability matrix $\boldsymbol{P}$
- The $N \times N$ matrix of cumulative transition duration distributions $\boldsymbol{D}(t)$

For better readability of formulas, define $\lambda = (\boldsymbol{\pi}, \boldsymbol{B}, \boldsymbol{P}, \boldsymbol{D}(t))$ to be the set of parameters. The number of states and the set of symbols are not included since the first is not altered by the Baum-Welch training algorithm and the second is application specific and is hence fixed and not further addressed here.

5 Generating Temporal Sequences with GHSMMs

The easiest way to explain the mode of operation of GHSMMs is to look at the first problem of temporal sequence processing, which refers to the generation of a time series. The goal of time series generation is to obtain a random sequence that obeys to some properties characterizing some typical behavior. This translates into a specification of a GHSMM and then to simulate a random run through the model. The emissions that are generated during the run form a random temporal sequence. To illustrate this, the model shown in Figure 10 is used and one exemplary time series is generated as follows:

1. The initial state is chosen according to probability distribution $\boldsymbol{\pi}$, say state s_2.
2. The first emission symbol is drawn according to probability distribution $b_2(o_k)$, say o_2.
3. The second state is determined according to probability distribution p_{2j}, which is the second row in $\boldsymbol{P}$. Say the second state was s_1.
4. Having decided about the successor state, a duration for the transition is sampled from distribution $d_{21}(t)$. Assume the duration was 13 seconds.
5. The second emission symbol is chosen according to $b_1(o_k)$, say o_1.
6. The procedure now repeats from the third step. Assume that the third state was s_4, the transition's duration was 7 seconds and that the third symbol emitted is o_1.

7. Since s_4 is an absorbing state, sequence generation ends.

The described example results in a time series as shown in Figure 11.

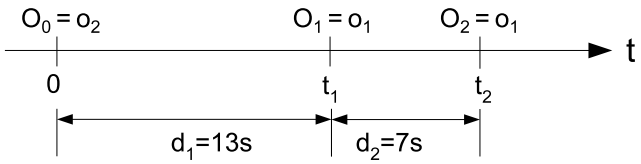

Figure 11: A temporal sequence that could have been generated by the model shown in Figure 10.

The figure also introduces some notations that will be used in the following: $O_i = o_k$ denotes that the i-th element of a sequence is symbol o_k. Therefore, the sequence of the example is represented by $O_0 = o_2$, $O_1 = o_1$, $O_2 = o_1$. The same applies to the sequence of hidden states the process has traversed. The sequence of states in the example is $S_0 = s_2$, $S_1 = s_1$, $S_2 = s_4$. Furthermore, d_i denotes the delay between symbols O_i and O_{i-1}, which in the example corresponds to $d_1 = 13s$, $d_2 = 7s$.

6 Recognition of Temporal Sequences: The Forward Algorithm

The term "sequence recognition" refers to the task of deciding whether some sequence (that has been observed before) belongs to some typical pattern or not. In the case of soft classification not a binary decision but a probability of class membership is desired. Sequence recognition with GHSMMs is achieved by computing the probability that the sequence were generated by the GHSMM. This probability is called sequence likelihood and can be computed by the forward algorithm.

In order to elucidate properties and limitations of the GHSMM approach, previous extensions of HMMs to continuous-time are analyzed in more detail, first. The focus is on duration modeling in speech recognition applications, since most extensions have been developed in this area. It is then illustrated why the assumptions of previous extensions are not appropriate for event-driven time series and the approach taken by GHSMMs is explained. The last part of the section concentrates on a formal derivation of the forward algorithm.

6.1 Duration Modeling in Speech Recognition

In the area of speech recognition, one of the tasks is to assign phonemes to a series of sound samples. The procedure is quite complex and will only be explained in a simplified version, here. However, there are a few things that need to be mentioned in order to understand the difference between speech recognition and temporal sequence processing of event-driven time series.

The process of phoneme recognition is sketched in Figure 12. Starting from the bottom of the figure,

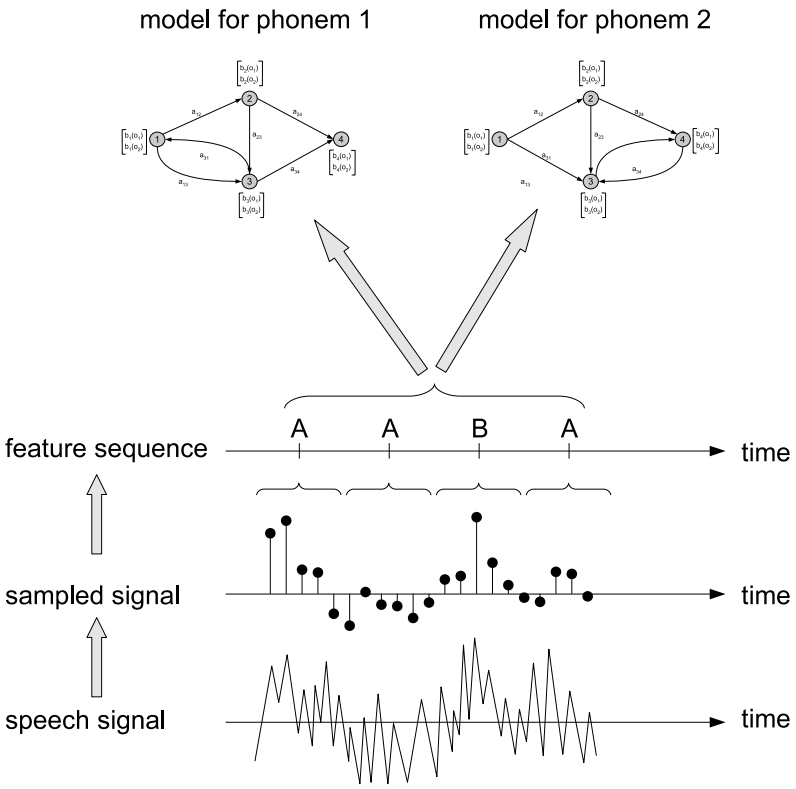

Figure 12: A simplified sketch of phoneme assignment to a speech signal.

---

the analog sound signal is sampled and converted into a digital signal. Portions of the sampled signal are then analyzed in order to extract features of the signal. Feature extraction involves, e.g., a short-time Fourier transform and various other computations. Since this report focuses on discrete emissions only, assume that the result of feature extraction is one symbol out of a discrete set[4]. Then, the sequence is analyzed by several HMMs: Each HMM is representing one phoneme and sequence likelihood is computed for each using the forward or Viterbi algorithm. In order to assign phonemes to the sequence of features, phonemes having maximum sequence likelihood are selected.

As it was pointed out by several authors, the quality of assignment can be improved by introducing the notion of state duration: Traversing the hidden states should not be absolutely synchronous to the sequence of features. It should rather be possible that the stochastic process resides in one state generating several emissions before transitioning to another state. This is visualized in Figure 13. Figure 13 a) shows the trivial case where each feature symbol corresponds to a state transition. Introducing the notion of state duration, the process of state transitions is decoupled from the occurrence of observation symbols since the process can stay in one state producing several subsequent feature symbols. However, this flexibility results in an increased number of potential state sequences, as can be seen from Figures 13 b) to d). The increased number of potential

[4]Usually, it is a feature vector containing both discrete and continuous values

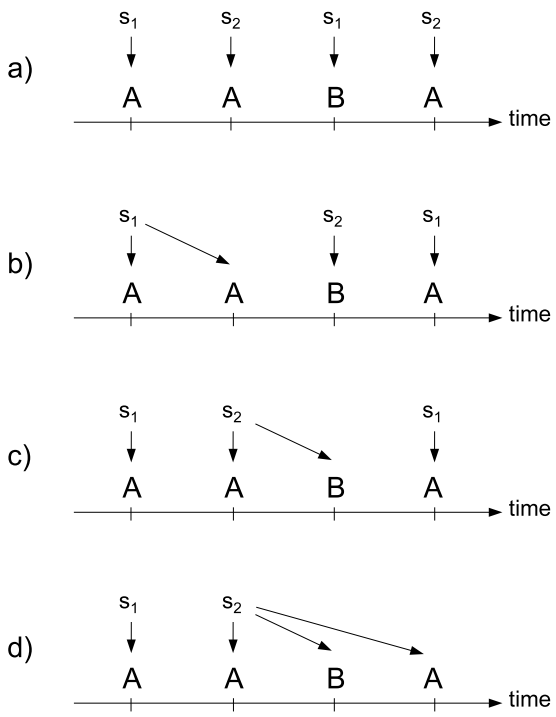

Figure 13: Assigning states s_i to feature symbols A or B.

---

state sequences obviously increases computational complexity to compute sequence likelihood as all possible paths have to be summed up (c.f., Equation 6). To be precise, the number of potential paths increase from N^L to

$$\sum_{k=0}^{L-1} \binom{L-1}{k} N\,(N-1)^k \tag{47}$$

where L denotes the length of the sequence and k the number of state transitions that take place during sequence generation, and $\binom{x}{0}$ is assumed to be equal to 1. There is no chance to apply dynamic programming in order to get to an efficient algorithm such as the forward algorithm (c.f., Section 2.1). This is because the Markov assumptions do not apply and the necessary condition that all the information needed to compute $\alpha_t(j)$ must be included in α's of the previous time step is not fulfilled for variable state durations.

Concrete models that were used in speech recognition have typically applied one restriction in order to come up with an algorithm that is close to the original forward algorithm: They included an upper bound for state durations (denoted by D). This leads to the following forward-like algorithm (see, e.g., [16]):

$$\alpha_t(j) = \sum_{i=1}^{N} \sum_{\tau=1}^{\min(D,t)} \alpha_{t-\tau}(i)\, a_{ij}\, d_j(\tau) \prod_{m=0}^{\tau-1} b_j(O_{t-m}) \tag{48}$$

$\alpha_t(j)$ denotes the probability of the observation sequence for all state sequences where state s_j ends at time t. The algorithm includes an additional sum over τ, which is the duration how long the process stays in state s_j. $d_j(\tau)$ specifies the probability for such a duration. The product over $b_j(\cdot)$

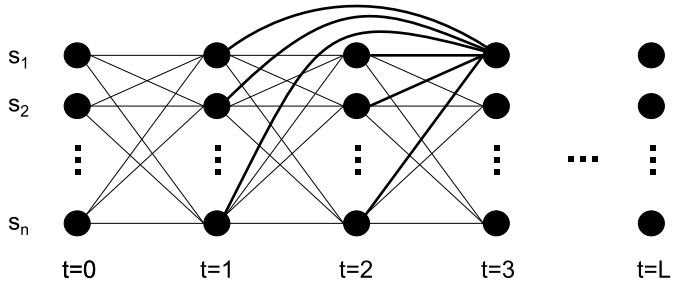

Figure 14: The trellis structure showing the effect of duration modeling for $D = 2$ (c.f., Figure 4 and Equation 48). Thick lines highlight terms involved in computation of $\alpha_3(1)$

---

results from the fact that during its stay, state s_j has to produce all the emission symbols. Similar to the standard forward algorithm, the approach can be visualized by a Trellis structure, as shown in Figure 14.

The major drawback of the algorithm is its computational complexity – it increases by a factor of $\frac{D^2}{2}$ (c.f., [23]).

The various models that have been listed in Section 3 take different approaches to specify the duration distribution $d_j(\tau)$ and to estimate the distribution from training data.

6.2 Duration Modeling for Event-driven Time Series

The essential difference between speech recognition and modeling of event-driven time series is that in speech recognition the sequence of symbols occur equidistantly (periodically) which is not the case in event-driven series.

The first conclusion from this difference is that delays are not integer steps anymore, but some interval on the real scale. Moreover, delays may cover a wide range of values ranging from very short to very long time spans. Therefore, discrete delay models such as Ferguson's are not well-suited.

The second conclusion is that time variability is already included in the observation sequence itself. Therefore, it is not necessary anymore that the stochastic process of hidden state traversals is independent of the occurrence of observation symbols and the forward algorithm has to investigate various state durations that cannot be inferred from the observed data. Moreover, a close relation between hidden state transition and occurrence of observation symbol can be assumed. Specifically, GHSMMs build on a one-to-one mapping of hidden states and observation symbols, as shown in Figure 15.

The one-to-one mapping of state transitions and observation symbol occurrence has two advantages:

1. It enforces the Markov assumption and hence leads to a forward algorithm that is very similar to the standard forward algorithm of discrete-time HMMs. Specifically, the sum over τ in

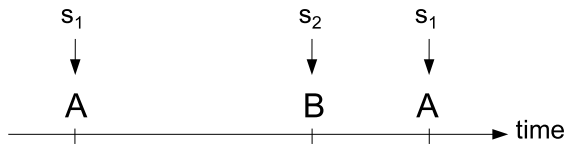

Figure 15: Time variation is included in the observation sequence. Therefore, GHSMMs assume a one-to-one mapping of states and symbols, enforcing the Markov assumption. This leads to efficient algorithms.

Equation 48 is not necessary anymore and the GHSMM forward algorithm hence belongs to the same complexity class as the standard forward algorithm (see Section 10).

2. It allows to assign durations to transitions rather than to states, which increases modeling flexibility and expressiveness. Obviously, state durations are a special case of transition durations, where all outgoing transitions have the same duration distribution.

6.3 The Forward Algorithm for GHSMMs

The forward algorithm of GHSMMs is derived from the discrete-time equivalent as described by Equations 7–9. The fact that event-driven observation sequences are considered leads to a change in time indexing: t_k denotes the time when symbol O_k has occurred instead of t denoting a time step in discrete-time equivalents.

In standard discrete-time HMMs, the transition probability is simply a_{ij}. In GHSMMs, the transition probability is replaced by $g_{ij}(t)$ specifying the probability that the transition has taken place at a time less or equal to t. However, in order to derive the forward algorithm for GHSMMs, a one-to-one replacement of a_{ij} with $g_{ij}(t)$ is not sufficient. This follows from the following considerations:

1. Assume that at time t_{k-1} the stochastic process has just entered state s_i and has emitted observation symbol o_{k-1}.

2. As the goal of the forward algorithm is to compute probability of a given time series $\boldsymbol{o}$ and GHSMMs assume that there is a state transition occurring at each observation symbol, it is known when the transition to the next state has taken place and the duration of the transition is $d_k := t_k - t_{k-1}$.

3. Knowing d_k we are able to compute the transition probability to each successor state s_j by $g_{ij}(d_k)$.

4. The subsequent symbol o_k is then emitted by state s_j with probability $b_j(o_k)$.

5. However, from Equations 40 and 41 follows that for the sum over all successor states the

following inequality holds[5]:

$$\sum_{j=1}^{N} g_{ij}(d_k) \;\leq\; 1 \tag{49}$$

expressing the fact that by time $d_k < \infty$ some fraction of the probability mass may not be distributed among the successor states. The explanation for this is natural: as d_k is finite, there may be some probability that the stochastic process still resides in state s_i after time d_k. The probability for this is

$$1 - \sum_{j=1}^{N} g_{ij}(d_k) \tag{50}$$

In this case it must be assumed that state s_i has generated symbol o_k

6. Applying Markov assumptions, the stochastic process looses all memory and considerations for the following sequence start from 1. Please note that this is the reason why the efficient dynamic programming approach of the forward algorithm can be applied.

In order to formalize these considerations, probability $v_{ij}(d_k)$ is defined as

$$v_{ij}(d_k) \;=\; P(S_k = s_j,\, d_k = t_k - t_{k-1} \mid S_{k-1} = s_i) \tag{51}$$

$$= \begin{cases} g_{ij}(d_k) & \text{if } i \neq j \\ 1 - \sum_{h=1}^{N} g_{ih}(d_k) & \text{if } i = j \end{cases} \tag{52}$$

This relates the GHSMM approach to IHMMs because for GHSMMs also holds that

$$\forall\, d,\, i: \quad \sum_{j=1}^{N} v_{ij}(d) = 1 \tag{53}$$

However, the process must still be called homogeneous since probabilities $v_{ij}(d)$ stay the same regardless of time t_k. Additionally in contrast to IHMMs, GHSMMs use continuous distributions rather than discrete ones.

Similar to the case of discrete-time HMMs, the forward variable for GHSMMs denotes the probability of the observation sequence up to time t_k for all state sequences that end in state s_i (at time t_k):

$$\alpha_k(i) = P(O_0\, O_1\, \ldots\, O_k,\, S_k = s_i|\lambda) \tag{54}$$

By replacing a_{ij} by $v_{ij}(t)$ and changing time indexing, the following recursive computation scheme for $\alpha_k(i)$ is derived:

$$\begin{aligned} \alpha_0(i) &= \pi_i\, b_i(O_0) \\ \alpha_k(j) &= \sum_{i=1}^{N} \alpha_{k-1}(i)\, v_{ij}(t_k - t_{k-1})\, b_j(O_k); \quad 1 \leq k \leq L \end{aligned} \tag{55}$$

[5]Remember that $g_{ii}\,(t) \equiv 0$ is assumed

By analogy with discrete-time HMMs, sequence likelihood $P(\boldsymbol{o} \mid \lambda)$ is the sum over the last column of the Trellis structure:

$$P(\boldsymbol{o} \mid \lambda) = \sum_{i=1}^{N} \alpha_L(i) \tag{56}$$

Again, sequence likelihood $P(\boldsymbol{o} \mid \lambda)$ is the probability that model λ can generate observation sequence $\boldsymbol{o}$. This probability can be used as an indicator if the sequence is exhibiting properties that are formalized by the GHSMM λ.

6.4 Finding the Most Probable Sequence of States: The Viterbi Algorithm

The forward algorithm incorporates all possible state sequences. However, in some applications this is it not desired and only the most probable sequence of states should be considered. This is computed by the Viterbi algorithm.

In analogy with discrete-time HMMs, the Viterbi algorithm for GHSMMs is derived by replacing the sum over all previous states with the maximum operator:

$$\delta_k(i) = \max_{S_0\, S_1 \ldots S_{k-1}} P(O_0\, O_1 \ \ldots\ O_k,\, S_0,\, S_1,\, \ldots,\, S_{k-1},\, S_k = s_i \mid \lambda) \tag{57}$$

$$\delta_0(i) = \pi_i\, b_i(O_0) \tag{58}$$

$$\delta_k(j) = \max_{1 \le i \le N} \delta_{k-1}(i)\, v_{ij}(t_k - t_{k-1})\, b_j(O_k) \tag{59}$$

Hence $\max_i \delta_L(i)$ is the maximum probability of any state sequence generating observation sequence $\boldsymbol{o}$. The sequence of states itself can again be obtained by storing which state was selected by the maximum operator and then tracing back through the array starting from state $\arg\max_i \delta_L(i)$.

7 Sequence Prediction

Sequence prediction deals with the estimation of the future behavior of a temporal sequence. Given a model and a time series, the question is how the time series will evolve in the near future based on the characteristics expressed by the underlying model. More precisely, two different types of sequence prediction can be distinguished:

1. What is the probability for the next observation or observations of the sequence?
2. What is the probability that the underlying stochastic process will reach a certain distinguished state within some time interval?

7.1 Probability of the Next Observation

In order to estimate the probability of next observations, the following probability is defined:

$$\eta_t(o_k) = P(O_{L+1} = o_k,\, T \leq t \,|\, t_L,\, O_0 \,\ldots\, O_L,\, \lambda); \quad t \geq t_L \tag{60}$$

$\eta_t(o_k)$ is the probability that the next emitted observation symbol is o_k, given the time of symbol occurrence being less or equal to t, a GHSMM λ and the beginning of an event-driven time series $\boldsymbol{o} = O_0 \,\ldots\, O_L$. For $\eta_t(o_k)$ the following equations hold:

$$\begin{aligned} \eta_t(o_k) &= \sum_{j=1}^{N} P(S_{L+1} = s_j,\, O_{L+1} = o_k,\, T \leq t \,|\, t_L,\, \boldsymbol{o},\, \lambda) && (61) \\ &= \sum_{j=1}^{N} P(O_{L+1} = o_k,\, T \leq t \,|\, S_{L+1} = s_j,\, t_L,\, \boldsymbol{o},\, \lambda)\, P(S_{L+1} = s_j \,|\, t_L,\, \boldsymbol{o},\, \lambda) && (62) \end{aligned}$$

For the first probability in Equation 62 holds:

$$P(O_{L+1} = o_k,\, T \leq t \,|\, S_{L+1} = s_j,\, t_L,\, \boldsymbol{o},\, \lambda) \;= b_j(o_k) \tag{63}$$

whereas the second probability in Equation 62 can be split up further:

$$\begin{aligned} & P(S_{L+1} = s_j,\, T \leq t \,|\, t_L,\, \boldsymbol{o},\, \lambda) && (64) \\ &= \sum_{i=1}^{N} P(S_{L+1} = s_j,\, S_L = s_i,\, T \leq t \,|\, t_L,\, \boldsymbol{o},\, \lambda) && (65) \\ &= \sum_{i=1}^{N} P(S_{L+1} = s_j,\, T \leq t \,|\, S_L = s_i,\, t_L,\, \boldsymbol{o},\, \lambda)\, P(S_L = s_i \,|\, t_L,\, \boldsymbol{o},\, \lambda) && (66) \\ &= \sum_{i=1}^{N} v_{ij}(t - t_L)\, P(S_L = s_i \,|\, t_L,\, \boldsymbol{o},\, \lambda) && (67) \end{aligned}$$

and by use of the forward algorithm:

$$P(S_L = s_i \,|\, t_L,\, \boldsymbol{o},\, \lambda) \;=\; \frac{\alpha_L(i)}{P(\boldsymbol{o} \,|\, \lambda)} = \frac{\alpha_L(i)}{\sum_{j=1}^{N} \alpha_L(j)} \tag{68}$$

or in case of the Viterbi algorithm:

$$P(S_L = s_i \,|\, \boldsymbol{o},\, \lambda) \;=\; \frac{\delta_L(i)}{\sum_{j=1}^{N} \delta_L(j)} \tag{69}$$

Summarizing Equations 61–68, the probability that observation symbol o_k will occur up to time t in the future can be computed by

$$\eta_t(o_k) = \sum_{j=1}^{N} b_j(o_k) \sum_{i=1}^{N} v_{ij}(t - t_L)\, \frac{\alpha_L(i)}{P(\boldsymbol{o} \,|\, \lambda)} \tag{70}$$

7.2 Probability to Reach a Distinguished State

Computing probabilities for the next observation symbol involved a probability distribution for the next hidden state S_{L+1} (see Equation 67). However, if not the next observation symbol is of interest but the probability distribution to reach a distinguished state, computation of the first-step successor is not sufficient. Moreover, the general probability to reach the distinguished state s_d for the first time by time t is desired:

$$P(S_d = s_d,\, T_d \leq t \,|\, \boldsymbol{o}, \lambda); \quad T_d = \min(\, t : S_t = s_d) \tag{71}$$

The procedure to compute this probability involves two steps:

1. Based on the given observation sequence $\boldsymbol{o}$ and the GHSMM λ, compute the probability distribution for the last hidden state in the sequence $P(S_L = s_i \,|\, \boldsymbol{o},\, \lambda)$ using Equation 68 or 69.

2. As future observations are unknown, all possible sequences of observation symbols have to be taken into account. Due to the fact that $\sum_{k=1}^{M} b_i(o_k) = 1$ for any state s_i, summing over all output symbols can be omitted and only the semi-Markov process of hidden state transitions has to be analyzed. In semi-Markov theory, the desired quantity is called "first passage time distribution"

In order to compute the first passage time distribution, so-called *first step analysis* is applied (see, e.g., [26]). The essence of first step analysis is as follows:

> The first step of the stochastic process reaches the designated state directly or the process transits to an intermediate state. In the latter case, the designated state is then reached directly from the intermediate state or via another intermediate state. This establishes a recursive computation scheme.

Let T_d denote the time to first reach the designated state s_d, then

$$F_{id}(t) = P(T_d \leq t | S_L = s_i) \tag{72}$$

is the probability to reach s_d by time t given that the semi-Markov process of hidden state traversals is in state s_i at the end of the observation sequence. Then

$$F_{id}(t) = g_{id}(t) + \sum_{j \neq d} \int_0^t d\, g_{ij}(\tau)\, F_{jd}(t - \tau) \tag{73}$$

where $g_{id}(t)$ is the cumulative probability distribution as defined in Section 4.1 and

$$\int_0^t d\, g_{ij}(\tau)\, F_{jd}(t - \tau)$$

denotes the Riemann integral.

In order to solve the equation system of (73), a recursive scheme can be defined:

$$F_{ij}^{(0)}(t) = 0 \tag{74}$$

$$F_{ij}^{(n+1)}(t) = g_{id}(t) + \sum_{j \neq d} \int_0^t d\, g_{ij}(\tau)\, F_{jd}^{(n)}(t-\tau) \tag{75}$$

having the property that for convergence of the approximation holds:

$$\sup_{0 \leq x \leq t} \left| F_{ij}^{(n)}(x) - F_{ij}(x) \right| \leq \mu^{\left[\frac{n}{r}\right]} \tag{76}$$

since for any fixed $t \geq 0$, an integer r and real number $0 < \mu < 1$ exists such that:

$$\sum_j g_{ij}^{*r}(t) \leq \mu \tag{77}$$

where $g_{ij}^{*r}(t)$ denotes the r-th convolution of $g_{ij}(t)$ with itself.

Since $F_{id}(t)$ assumes the stochastic process to be initially in state s_i, we have to sum up all states S_L in order to compute the probability to reach state s_d within time t:

$$P(S_d = s_d,\, T_d \leq t \mid \boldsymbol{o}, \lambda) = \sum_i F_{id}(t)\, P(S_L = s_i \mid \boldsymbol{o},\, \lambda) \tag{78}$$

Computation of Equation 75 can be quite costly, depending on n, which is the maximum number of transitions up to time t that are considered in the approximation. Additionally, each step involves solution of the Riemann integral which must in many cases be solved numerically as there are many distributions for which there's no analytical representation (e.g., the cumulative distribution of a Gaussian random variable). However, computational complexity can be limited since the maximum number of transitions is commonly limited by the application (in most applications, there is a minimum delay between successive observations) which also limits the number of time points for which the Riemann integral has to be approximated.

A second important note is related to real-time or online applications. $F_{id}(t)$ is only depending on the parameters of the GHSMM and not on the observation sequence: it can hence be precomputed. Online evaluation of Equation 78 only has to compute Equations 68 or 69 for each state, multiply with precomputed $F_{id}(t)$ and sum up the products.

8 Extracting Features from Training data: The Baum-Welch Algorithm

Previous sections assumed that a GHSMM was given. This section deals with the task to estimate the parameters λ of a GHSMM from training sequences that have been observed prior to modeling. For this purpose, the Baum-Welch algorithm for discrete-time HMMs (as described in Section 2.2) is adapted to GHSMMs.

8.1 The Backward Variable, Xi and Gamma

In addition to the forward variable $\alpha_k(i)$, reestimation formulas for discrete-time HMMs were based on a backward variable $\beta_t(i)$, a state probability $\gamma_t(i)$, and a transition probability $\xi_t(i,j)$. The same applies to reestimation formulas for GHSMMs. The GHSMM equivalents $\beta_k(i)$, $\gamma_k(i)$ and $\xi_k(i,j)$ are defined as follows.

The backward variable $\beta_k(i)$ is the probability of the rest of the observation sequence $O_{k+1} \ldots O_L$ given that the process is in state s_i at time t_k and a GHSMM. $\beta_k(i)$ is computed backwards starting from time t_L:

$$\beta_k(i) = P(O_{k+1} \ldots O_L \mid S_k = s_i, \lambda) \tag{79}$$

$$\beta_L(i) = 1$$

$$\beta_k(i) = \sum_{j=1}^{N} v_{ij}(d_k)\, b_j(O_{k+1})\, \beta_{k+1}(j) \tag{80}$$

$\gamma_k(i)$ is the probability that the stochastic process is in state i at the time when the k-th observation occurs. It can be computed from $\alpha_k(i)$ and $\beta_k(i)$ following the same scheme as presented in Equations 15–18:

$$\gamma_k(i) = \frac{\alpha_k(i)\, \beta_k(i)}{\sum_{i=1}^{N} \alpha_k(i)\, \beta_k(i)} \tag{81}$$

$\xi_k(i,j)$ is the probability that the stochastic process is in state s_i at time t_k and transits to state s_j at time t_{k+1}:

$$\xi_k(i,j) = P(S_k = s_i, S_{k+1} = s_j \mid \boldsymbol{o}, \lambda) \tag{82}$$

$$\xi_k(i,j) = \frac{\alpha_k(i)\, g_{ij}(d_{k+1})\, b_j(O_{k+1})\, \beta_{k+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_k(i)\, g_{ij}(d_{k+1})\, b_j(O_{t+1})\, \beta_{k+1}(j)} \tag{83}$$

Please note that $\xi_k(i,j)$ uses $g_{ij}(t)$ instead of $v_{ij}(t)$. This is due to the fact that $\xi_{ij}(t)$ is about *transitions* but $v_{ij}(t)$ also contains the probability that the process has stayed in state s_i, and self-transitions are not considered ($p_{ii} = 0$).

As was the case for discrete-time HMMs, the expected number of transitions from state s_i to state s_j is the sum over time

$$\sum_{k=0}^{L-1} \xi_k(i,j). \tag{84}$$

However, the relation between $\gamma_t(i)$ and $\xi_t(i,j)$, as given in Equation 34 (page 10), does not hold for GHSMMs: The sum of transition probabilities to all successor states does not include the probability that the process remains in state s_i (c.f., Equation 49) and is hence not equal to $\gamma(i)$.

8.2 Reestimation Formulas

Expectedly, the so-called reestimation formulas for GHSMMs are very similar to their discrete-time HMM equivalents, which are described in Section 2.2. In analogy with discrete-time HMMs, reestimation formulas are only provided for one single training sequence. In the common case of multiple sequences, the same procedure as in Section 2.3 must be applied.

Initial probabilities $\boldsymbol{\pi}$ are computed in the same way as for discrete-time HMMs:

$$\bar{\pi}_i \equiv \frac{\text{expected number of series starting in state } s_i}{\text{total number of sequence}} \equiv \gamma_0(i) \tag{85}$$

Emission probabilities $\boldsymbol{b_i(o_j)}$ are as well identical to their discrete-time equivalents:

$$\bar{b}_i(o_j) \equiv \frac{\text{expected number of times observing } o_j \text{ in state } s_i}{\text{expected number of times in state } s_i} \equiv \frac{\sum\limits_{k=0}^{L} \gamma_k(i)}{\sum\limits_{k=0}^{L} \gamma_k(i)} \tag{86}$$

Transition distributions $\boldsymbol{g_{ij}(t)}$. According to Equation 44 (page 17), $g_{ij}(t)$ is the product of the limiting transition probability p_{ij} of the embedded DTMC and the duration distribution $d_{ij}(t)$.

Limiting transition probabilities $\boldsymbol{p_{ij}}$ are independent of time since they are obtained by letting $t \to \infty$ and p_{ij} simply denotes the probability that a transition from state s_i to s_j has taken place in the training sequence. Hence, p_{ij} can be identified with transition probabilities a_{ij} of HMMs and the same formulas apply. In the Baum-Welch algorithm, a_{ij} are estimated by the relative frequency of $s_i \to s_j$ transitions, and from Equation 84 follows:

$$\bar{p}_{ij} \equiv \frac{\text{expected number of transitions } s_i \to s_j}{\text{expected number of transitions } s_i \to ?} \equiv \frac{\sum\limits_{k=0}^{L-1} \xi_k(i,j)}{\sum\limits_{j=1}^{N} \sum\limits_{k=0}^{L-1} \xi_k(i,j)} \tag{87}$$

Transition durations $\boldsymbol{d_{ij}(t)}$. The procedure to estimate $d_{ij}(t)$ is more complex since GHSMMs allow to use a great variety of duration distributions and an "embedded" expectation-maximization algorithm is used. Additionally, if only a subset of transition distributions d_{ij} are used, a more efficient maximization algorithm can be applied. The next section describes both variants in detail.

8.3 Maximizing Duration Likelihood of Transitions

Transition durations $d_{ij}(t)$ are cumulative probability distributions specifying the duration of transitions from state s_i to s_j last. The problem is, that for any training sequence $\boldsymbol{o}$ it is not known exactly what path of hidden states the stochastic process has traversed. Instead, the only two things known about each transition that took place from observation symbol O_{k-1} to O_k are:

1. the duration of the transition: d_k
2. the probability that a transition from hidden state s_i to s_j took place: $\xi_{k-1}(i,j)$

Combining both quantities for each pair of observation symbols $O_{k-1} \rightarrow O_k$; $1 \leq k \leq L$ having a delay of d_k, a weight can be assigned to each d_k for each pair of states $s_i \rightarrow s_j$ by $\xi_{k-1}(i,j)$. This results in a set of weighted durations as shown in Figure 16. The goal is to find transition duration distribution densities $d_{ij}^{'}(t)$ such that they best represent the weighted sample.

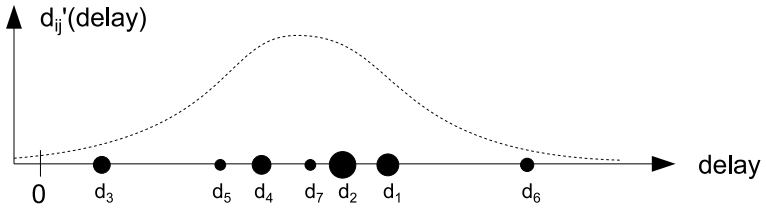

Figure 16: Weighted distribution of transition durations and estimated Gaussian duration distribution density. Points visualize the delay values d_k that occur in the training sequence, their size corresponding to the weights $\xi_{k-1}(i,j)$. The dashed curve symbolizes an estimated probability density of the duration distribution $d_{ij}^{'}(t)$.

Recall that transition duration distributions in GHSMMs are defined as:

$$d_{ij}(t) = \sum_{r=0}^{R} w_{ij,r}\, \kappa_{ij,r}(t|\theta_{ij,r}) \tag{88}$$

$$\text{s.t.} \sum_{r=0}^{R} w_{ij,r} = 1 \tag{89}$$

Hence more specifically, the goal of transition duration optimization is to adjust the weights $w_{ij,r}$ and parameters $\theta_{ij,r}$ for each transition $s_i \rightarrow s_j$ such that the likelihood of the weighted delays of the training sequence is maximized. Due to the Markov assumptions it is possible to estimate each $d_{ij}(t)$ independently for each transition $s_i \rightarrow s_j$ and further considerations will focus on evaluation of one single transition. Therefore, the index ij will be omitted in cases where only one transition is addressed.

GHSMMs use an expectation maximization (EM) approach to achieve this. EM algorithms repetitively apply maximum likelihood estimation (see, e.g., [29]). Since this is the most complex part of EM algorithms, the formulas are provided here.

EM algorithms typically do not maximize the data likelihood directly but maximize a so-called Q-function (Section 9 provides more details about this procedure). In the case of transition duration optimization, the following Q-function has to be maximized:

$$Q_d = \sum_{k=1}^{L} \sum_{r=0}^{R} P(d_k)\, \log\left[P(d_k \,|\, r)\, P(r)\right]\, P\big(r \,|\, d_k,\, \theta^{old}\big) \tag{90}$$

where

- $P(d_k)$ is the probability of the data point (delay between O_{k-1} and O_k)
- $P(r)$ is the probability of kernel r
- $P(d_k \,|\, r)$ is the probability that delay d_k is generated by kernel r
- $P\left(r \,|\, d_k,\, \theta^{old}\right)$ is the probability of kernel r given the data point and previous set of parameters.

$$\begin{aligned}
P(d_k) &= \xi_{k-1} && (91)\\
P(r) &= w_r && (92)\\
P(d_k \,|\, r) &= \kappa'_r(d_k|\theta_r) && (93)\\
P\left(r \,|\, d_k,\, \theta^{old}\right) &= \frac{P(d_k \,|\, r,\, \theta^{old})\, P(r \,|\, \theta^{old})}{\sum_r P(d_k \,|\, r,\, \theta^{old})\, P(r \,|\, \theta^{old})} = \frac{\kappa'_r(d_k|\theta_r^{old})\, w_r^{old}}{\sum_r \kappa'_r(d_k|\theta_r^{old})\, w_r^{old}} && (94)
\end{aligned}$$

with $\kappa'_r(\cdot)$ denoting the probability density of $\kappa_r(\cdot)$, which are defined to be cumulative distributions, and superscripts ^{old} referring to parameter values of the previous reestimation step.

The logarithm of Equation 90 can be split into a sum

$$\begin{aligned}
Q_d = &\sum_{k=1}^{L} \sum_{r=0}^{R} P(d_k)\, \log\left[P(d_k \,|\, r)\right]\, P\big(r \,|\, d_k,\, \theta^{old}\big)\\
&+ \sum_{k=1}^{L} \sum_{r=0}^{R} P(d_k)\, \log\left[P(r)\right]\, P\big(r \,|\, d_k,\, \theta^{old}\big)
\end{aligned} \tag{95}$$

It can be observed that the two parts of Equation 95 are independent of each other from which follows that weights w_r and kernel parameters θ_r can be optimized separately.

As will be derived in Section 9, reestimation of the kernel parameters θ_r follows from partial differentiation of the first summand of Equation 95 with respect to θ_r. Since this eliminates the sum over the kernels $\sum_r$, each kernel's parameters can be optimized separately by solving:

$$\begin{aligned}
\frac{\partial\, Q_d}{\partial\, \theta_r} &= \sum_{k=1}^{L} \frac{1}{P(d_k \,|\, r,\, \theta_r)}\, \frac{\partial\, P(d_k \,|\, r,\, \theta_r)}{\partial\, \theta_r}\, P(d_k)\, P\left(r \,|\, d_k,\, \theta^{old}\right)\\
&= \sum_{k=1}^{L} \frac{1}{\kappa'_r(d_k, \theta_r)}\, \frac{\partial\, \kappa'_r}{\partial\, \theta_r}\, \xi_{k-1}\, \frac{\kappa'_r(d_k|\theta_k^{old})\, w_r^{old}}{\sum_r \kappa'_r(d_k|\theta_k^{old})\, w_r^{old}}\\
&\stackrel{!}{=} 0
\end{aligned} \tag{96}$$

Since this is maximum likelihood parameter estimation, it is recommended to use kernels for which a formal solution to Equation 96 exists. Examples include Gaussian, exponential, Pareto, or log-normal distributions. Appendix A provides equations for some distributions. However, if no analytical solution exists (e.g., for the gamma distribution), Equation 96 can be solved numerically, which of course increases computational complexity. However since the EM algorithm is an iterative procedure, it is sufficient to use a good estimate that increases Q_d (such an estimate for the gamma distribution is provided in Appendix A). This approach is called generalized expectation maximization (GEM) algorithm.

Reestimation of the kernels' weights can be obtained from the second summand of Equation 95. Due to the constraint given in Equation 89, Lagrange optimization is applied yielding:

$$w_r = P(r) = \frac{\sum_{k=1}^{L} \xi_{k-1}\, P\left(r \mid d_k,\, \theta^{old}\right)}{\sum_{k=1}^{L} \xi_{k-1}} \tag{97}$$

The derivation of the formula is again provided in Section 9.

The special case of single kernel durations. If only one kernel κ_{ij} is used for each transition duration, Equation 90 turns into the standard objective function for maximum likelihood (ML) estimation. It can be shown that for special distributions ML estimation is equal to the methods of moments, where the first n moments are estimated from observation samples and the distribution's parameters are computed from them.[6] The Normal distribution $\mathcal{N}(\mu, \sigma^2)$ is one prominent example (see, e.g., [30] for more). In general, the n-th moment for durations of a transition from state s_i to s_j is:

$$E\left[D^n\right] = \int_{-\infty}^{\infty} d^{\,n}\, P(d) \tag{98}$$

which in in our case boils down to estimation from the finite sample of durations occurring in the observation sequence:

$$E\left[D^n\right] = \frac{\sum_{k=1}^{L} d_k^n\ \xi_{k-1}(i,j)}{\sum_{k=1}^{L} \xi_{k-1}(i,j)} \tag{99}$$

8.4 A Summary of the Training Algorithm

Since a lot of formulas have been developed in the last sections, the entire training procedure is reiterated here. The goal of the training procedure is to adjust the GHSMM parameters λ such that the likelihood of a given training sequence $\boldsymbol{o}$ is maximized. However, the Baum-Welch training

[6] This is known as the method of moments.

algorithm does only affect $\boldsymbol{\pi}$, $\boldsymbol{B}$, $\boldsymbol{P}$, and $\boldsymbol{D}(t)$, but not the *structure* of the GHSMM. The structure consists of

- the set of states $\mathscr{S} = \{s_1, \ldots, s_N\}$,
- the set of symbols $\mathscr{O} = \{o_1, \ldots, o_M\}$,
- the topology of the model. It defines, which of the N states can be initial states, which of the potentially $N \times N$ transitions can be traversed by the stochastic process, and which of the potentially $N \times M$ emissions are available. Technically, a transition $s_i \to s_j$ can be "removed" by setting $p_{ij} = 0$. The same holds for the initial state distribution $\boldsymbol{\pi}$ and the emission probabilities: if $b_i(o_k)$ is set to zero, state s_i cannot generate observation symbol o_k. Since the Baum-Welch algorithm can never assign a non-zero value to probabilities that are equal to zero, Baum-Welch does not change the structure of the GHSMM.
- specification of the transition duration kernels $\boldsymbol{D}(t)$. This includes the *number* and *types* of kernels for each existing transition. It may also comprise specification of additional parameters that are not adjusted by maximum likelihood estimation. For example, upper and lower bounds for uniform background distributions need to be set up before training starts.

Having specified the GHSMM structure, the Baum-Welch algorithm performs the steps shown in Figure 17 in order to adjust the parameters λ such that sequence likelihood of $P(\boldsymbol{o} \,|\, \lambda)$ reaches at least a local maximum.

In analogy with discrete-time HMMs, the procedure was outlined only for training with one single sequence. If multiple sequences are used, computations are performed for each sequence and the results are combined as described in Section 2.3.

When executing the algorithms on computers, probabilities quickly approach the limit of computational accuracy, even with double precision floating point numbers. Therefore, a technique called *scaling* has been developed (see, e.g., [13]). The same technique can be applied to GHSMMs without restrictions.

9 Proving Convergence

Both, discrete-time HMMs and GHSMMs use the Baum-Welch algorithm for parameter estimation. Its main characteristic is that sequence likelihood is increased by an iterative reestimation procedure until some local maximum is reached. While convergence of the algorithm was originally proven by Baum et al. in [31], simpler proofs exploit the fact that the algorithm is in fact an instance of expectation maximization (EM) algorithms [32]. This section will therefore sketch the latter approach and adapt it to GHSMMs. Since the proof of convergence relies on a so-called Q-function, it will be introduced first, followed by the application to GHSMMs.

1. Initialize the model by assigning random values to $\boldsymbol{\pi}$, $\boldsymbol{B}$, $\boldsymbol{P}$, and $\boldsymbol{D}(t)$ for all entries that are existing in the structure. This is λ^{old}.

2. Compute $\alpha_k(i)$ by Equation 55, $\beta_k(i)$ by Equation 80, $\gamma_k(i)$ by Equation 81, and $\xi_k(i,j)$ by Equation 83 using λ^{old} and observation sequence $\boldsymbol{o}$.

3. Compute sequence likelihood $P(\boldsymbol{o} \,|\, \lambda^{old})$ by Equation 56.

4. Adjust $\boldsymbol{\pi}$ by Equation 85, $\boldsymbol{B}$ by Equation 86, and $\boldsymbol{P}$ by Equation 87.

5. Reestimate the parameters of $\boldsymbol{D}(t)$ by the embedded EM algorithm for each $d_{ij}(t)$:

   (a) Compute delay log-likelihood

   $$P(\boldsymbol{d} \,|\, \theta_{ij}^{old}) = \sum_{k=1}^{L} \xi_k(i,j) \, \log \left[d_{ij}(d_k) \right]$$

   where $d_{ij}(\cdot)$ is the mixture of kernels as in Equation 88.

   (b) Adjust the kernel parameters $\theta_{ij,r}$ by the solution to Equation 96 for each kernel $\kappa_{ij,r}$ of $d_{ij}(\cdot)$.

   (c) Adjust the kernel weights $w_{ij,r}$ by Equation 97.

   (d) Set $\theta_{ij}^{old} := \theta_{ij}^{new}$ and repeat steps 5a to 5d until the difference of delay log-likelihoods $P(\boldsymbol{d} \,|\, \theta_{ij})$ is less than some bound ϵ_d.

6. Set $\lambda^{old} := \lambda^{new}$ and repeat steps 2 to 6 until the difference in observation sequence likelihood $P(\boldsymbol{o} \,|\, \lambda)$ is less than some bound ϵ.

Figure 17: The complete Baum-Welch training algorithm for GHSMMs.

9.1 Derivation of the Q-Function

EM algorithms are one form of maximum-a-posteriori (MAP) estimators and hence rely on the probability of some data that has been observed, which in this case refers to the observation sequence $\boldsymbol{o}$ forming dataset $\mathcal{O}$. Assuming that observations are independent and identically distributed, the goal is to maximize data likelihood $P(\boldsymbol{o}|\lambda)$.

The potential of EM algorithms and their wide range of application stems from two properties:

1. EM algorithms build on *lower bound optimization*. This allows to optimize complex objective functions by maximizing a lower bound that is much easier to handle.

2. EM algorithms can handle *incomplete/unobservable data*.

Lower bound optimization The first property of EM algorithms is that they employ lower bound optimization [33], which is also called *primal-dual method* [34]. In lower bound optimization, an objective function, which is computationally intractable, is optimized by repetitive maximization of some lower bound that is easier to compute. If the lower bound is defined such that it equals the objective function at the point of current estimation, a maximization of the lower bound automatically leads to an increased value of the objective function – except for the case when the derivative of the objective function equals zero, which is a local optimum. (see Figure 18).

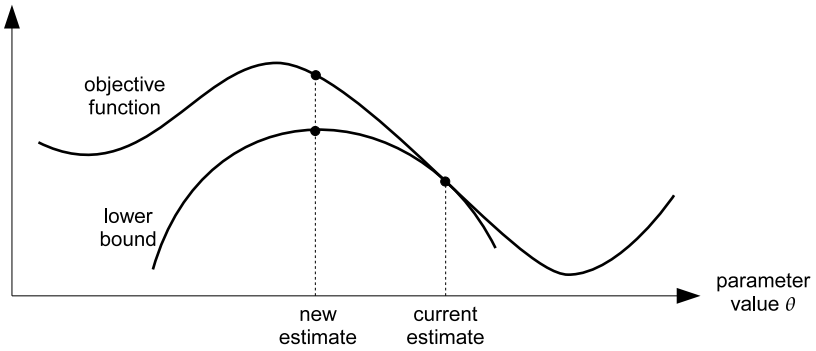

Figure 18: Lower bound optimization. Maximizing a lower bound that equals the objective function at the point of current estimation of parameter λ leads to an increased value of the objective function.

From this consideration, the following iterative optimization scheme can be derived:

1. Determine a lower bound that equals the objective function at the current estimate of parameter λ. This is the E-step of the algorithm.
2. Determine the maximum of the lower bound yielding the next estimation of λ. This is the M-step of the algorithm.
3. Repeat until the increase of the objective function is below some threshold.

In comparison to lower bound maximization, gradient-based optimization approaches build on a linear approximation at the point of the current estimate for θ and move along that line for some distance to obtain the new estimate.

Handling of unobservable data In the case of HMMs / GHSMMs, handling of unknown data refers to the fact that the sequence of hidden states $\boldsymbol{s}$, which the stochastic process has traversed, cannot be observed. Therefore, two data sets must be distinguished: the *complete dataset* $\mathcal{Z} = (\mathcal{O}, \mathcal{S})$ includes both observed and unknown data, while the *incomplete dataset* $\mathcal{O}$ only consists of observed data.

The way how EM algorithms deal with unknown data is that the incomplete data likelihood is

assumed to be the *marginal* of the complete data set. Hence,

$$P(\boldsymbol{o}|\lambda) = \int_{\boldsymbol{s}} P(\boldsymbol{o}, \boldsymbol{s}|\lambda)\, d\boldsymbol{s} \tag{100}$$

The Q-Function. In order to find a lower bound to data likelihood Jensen's inequality [35] can be used:

$$\sum_j g(j)\, a_j \geq \prod_j g_j^{a_j}(j); \quad a_j \geq 0, \sum_j a_j = 1,\, g(j) \geq 0 \tag{101}$$

stating that the arithmetic mean is never less than the geometric mean. Application to Equation 100 requires extension by some arbitrary function $q(\boldsymbol{s})$ as follows (see [33]):

$$P(\boldsymbol{o}|\lambda) = \int_{\boldsymbol{s}} \frac{P(\boldsymbol{o}, \boldsymbol{s}|\lambda)}{q(\boldsymbol{s})}\, q(\boldsymbol{s})\, d\boldsymbol{s} \;\geq\; \prod_{\boldsymbol{s}} \left(\frac{P(\boldsymbol{o}, \boldsymbol{s}|\lambda)}{q(\boldsymbol{s})} \right)^{q(\boldsymbol{s})\, d\boldsymbol{s}} = f(\lambda, q(\boldsymbol{s})) \tag{102}$$

where

$$\int_{\boldsymbol{s}} q(\boldsymbol{s})\, d\boldsymbol{s} \stackrel{!}{=} 1 \tag{103}$$

$f(\lambda, q(\boldsymbol{s}))$ is the lower bound and $q(\boldsymbol{s})$ is some arbitrary cumulative probability distribution over $\boldsymbol{s}$.

The arbitrary function f needs to be chosen such that the lower bound touches the objective function at the current estimate of parameters λ^{old}. It can be shown ([33]) that setting

$$q(\boldsymbol{s}) = P(\boldsymbol{s} \,|\, \boldsymbol{o},\, \lambda^{old}) \tag{104}$$

fulfills the requirement.

Maximization of the lower bound is performed by maximizing its logarithm. Substituting Equation 104 into the logarithm of $f(\lambda, q(\boldsymbol{s}))$ and dropping terms that are not depending on λ yields the so-called Q-function:

$$\begin{aligned} Q(\lambda, \lambda^{old}) &= \int_{\boldsymbol{s}} \log \big[P(\boldsymbol{o}, \boldsymbol{s}|\lambda) \big]\; P(\boldsymbol{s} \,|\, \boldsymbol{o},\, \lambda^{old})\, d\boldsymbol{s} && (105) \\ &= E_{\boldsymbol{s}} \big[\log P(\boldsymbol{o}, \boldsymbol{s}|\lambda) \big] && (106) \end{aligned}$$

which is in fact the expected value over $\boldsymbol{s}$ of the log-likelihood of the complete data set. Since likelihood of the complete data set is in many cases easier to optimize than the one of the incomplete data set, EM algorithms can solve more complex optimization problems.

EM algorithms. To summarize the functioning of EM algorithms, the following two steps are performed iteratively until some convergence with respect to training data likelihood is reached:

- **E-step:** Compute the Q-function based on parameters λ^{old} obtained by initialization or the previous M-step.
- **M-step:** Compute the next estimation for λ by maximizing the Q-function:

$$\lambda^{new} = \arg\max_{\lambda} Q(\lambda,\, \lambda^{old}) \tag{107}$$

Maximization is in most cases achieved by partial derivation of the Q-function and solving the equation

$$\frac{\partial Q}{\partial \lambda} \stackrel{!}{=} 0 \tag{108}$$

If there are additional constraints on the parameters (e.g., the sum of parameters has to be equal to 1), Lagrange multipliers are used. In the case that there is no analytical solution for the optimum, it is even sufficient to find some new parameter values λ for which the lower bound is greater than for λ^{old}. This approach is called "Generalized EM algorithm"

9.2 The Q-function for GHSMMs

In terms of GHSMMs, the complete dataset $\mathcal{Z} = (\mathcal{O}, \mathcal{S})$ consists of the observation sequence $\boldsymbol{o}$ and the sequence of hidden states $\boldsymbol{s}$ the stochastic process has traveled through. If both the sequence of hidden states and observation sequence are known, (complete) data likelihood is

$$\begin{aligned} P(\boldsymbol{o},\, \boldsymbol{s} \,|\, \lambda) &= \pi_{s_0}\, b_{s_0}(O_0) \prod_{k=1}^{L} v_{s_{k-1}\, s_k}(d_k)\, b_{s_k}(O_k) && (109)\\ &= \pi_{s_0} \prod_{k=0}^{L} b_{s_k}(O_k) \prod_{k=1}^{L} v_{s_{k-1}\, s_k}(d_k) && (110) \end{aligned}$$

and hence the Q-function for GHSMMs is

$$\begin{aligned} Q(\lambda, \lambda^{old}) &= \sum_{\boldsymbol{s}\in\boldsymbol{S}} \log\left[P(\boldsymbol{o}, \boldsymbol{s}|\lambda)\right] P(\boldsymbol{s} \,|\, \boldsymbol{o},\, \lambda^{old}) && (111)\\ &= \sum_{\boldsymbol{s}\in\boldsymbol{S}} \log\left[\pi_{s_0}\right] P(\boldsymbol{s} \,|\, \boldsymbol{o},\, \lambda^{old}) && (112)\\ &\quad + \sum_{\boldsymbol{s}\in\boldsymbol{S}} \sum_{k=0}^{L} \log\left[b_{s_k}(O_k)\right] P(\boldsymbol{s} \,|\, \boldsymbol{o},\, \lambda^{old}) && (113)\\ &\quad + \sum_{\boldsymbol{s}\in\boldsymbol{S}} \sum_{k=1}^{L} \log\left[v_{s_{k-1}\, s_k}(d_k)\right] P(\boldsymbol{s} \,|\, \boldsymbol{o},\, \lambda^{old}) && (114) \end{aligned}$$

where $\boldsymbol{S}$ denotes the set of all possible state sequences $\boldsymbol{s}$.

Some papers (e.g., [36]) use $P(\boldsymbol{s},\, \boldsymbol{o} \,|\, \lambda^{old})$ instead of $P(\boldsymbol{s} \,|\, \boldsymbol{o},\, \lambda^{old})$. In fact, this does not matter since

$$P(\boldsymbol{s},\, \boldsymbol{o} \,|\, \lambda^{old}) = P(\boldsymbol{s} \,|\, \boldsymbol{o},\, \lambda^{old}) \times P(\boldsymbol{o} \,|\, \lambda^{old}) \tag{115}$$

and since $P(\boldsymbol{o} \,|\, \lambda^{old})$ is independent of λ it does not affect the arg max operator used for maximization (see Equation 107).

9.3 Maximizing the Q-function

The important thing to note about the Equation for $Q(\lambda, \lambda^{old})$ is that terms 112 to 114 are independent of each other with respect to $\lambda = (\boldsymbol{\pi},\, \boldsymbol{B},\, \boldsymbol{G})$. Since maximization involves partial derivation,

the terms can be maximized separately.

Maximizing $\boldsymbol{\pi}$. The first term (112) can be further simplified:

$$\sum_{\boldsymbol{s}\in\boldsymbol{S}} \log\left[\pi_{s_0}\right] P(\boldsymbol{s}\,|\,\boldsymbol{o},\,\lambda^{old}) = \sum_{i=1}^{N} \log\left[\pi_i\right] P(S_0 = s_i\,|\,\boldsymbol{o},\,\lambda^{old}) \tag{116}$$

since for each $\boldsymbol{s} \in \boldsymbol{S}$, only the first state is of importance and $P(S_0 = s_i\,|\,\boldsymbol{o},\,\lambda^{old})$ contains all state sequences starting with state s_i.

In order to determine the new estimation for $\boldsymbol{\pi}$, the following constrained maximization problem has to be solved:

$$\pi_i = \arg\max_{\pi_i} Q(\lambda,\,\lambda^{old}); \quad s.t. \sum_{i=1}^{N} \pi_i = 1 \tag{117}$$

which can be done using a Lagrange multiplier φ:

$$\frac{\partial}{\partial \pi_i}\left(\sum_{i=1}^{N} \log\left[\pi_i\right] P(S_0 = s_i\,|\,\boldsymbol{o},\,\lambda^{old}) - \varphi\Big(\sum_{i=1}^{N} \pi_i - 1\Big)\right) \stackrel{!}{=} 0 \tag{118}$$

$$\Leftrightarrow \qquad P(S_0 = s_i\,|\,\boldsymbol{o},\,\lambda^{old})\,\frac{1}{\pi_i} - \varphi = 0 \tag{119}$$

$$\Leftrightarrow \qquad \pi_i = \frac{P(S_0 = s_i\,|\,\boldsymbol{o},\,\lambda^{old})}{\varphi} \tag{120}$$

Summing Equation 120 over i yields φ:

$$\sum_{i=1}^{N} \frac{P(S_0 = s_i\,|\,\boldsymbol{o},\,\lambda^{old})}{\varphi} \stackrel{!}{=} 1 \tag{121}$$

$$\Leftrightarrow \qquad \varphi = \sum_{i=1}^{N} P(S_0 = s_i\,|\,\boldsymbol{o},\,\lambda^{old}) = 1 \tag{122}$$

Substituting φ back into Equation 120 yields the solution

$$\pi_i \;=\; P(S_0 = s_i\,|\,\boldsymbol{o},\,\lambda^{old}) \;=\; \gamma_0(i) \tag{123}$$

Maximizing $\boldsymbol{b_i(o_j)}$. The second term (113) can be maximized in a similar way. The first step is to simplify the log-likelihood by the same arguments as before:

$$\sum_{\boldsymbol{s}\in\boldsymbol{S}} \sum_{k=0}^{L} \log\left[b_{s_k}(O_k)\right] P(\boldsymbol{s}\,|\,\boldsymbol{o},\,\lambda^{old}) = \sum_{i=1}^{N} \sum_{k=0}^{L} \log\left[b_i(O_k)\right] P(S_k = s_i\,|\,\boldsymbol{o},\,\lambda^{old}) \tag{124}$$

For readability reasons, $b_i(o_j)$ will be denoted by b_{ij}. Then the maximization problem becomes:

$$b_{ij} = \arg\max_{b_{ij}} Q(\lambda,\,\lambda^{old}); \quad s.t.\ \forall\, i:\ \sum_{j=1}^{M} b_{ij} \stackrel{!}{=} 1 \tag{125}$$

leading to

$$\frac{\partial}{\partial b_{ij}} \left(\sum_{i=1}^{N} \sum_{k=0}^{L} \log\left[b_{ik}\right] P(S_k = s_i \,|\, \boldsymbol{o},\, \lambda^{old}) - \sum_{i=1}^{N} \varphi_i \Big(\sum_{j=1}^{M} b_{ij} - 1 \Big) \right) = 0 \tag{126}$$

$$\Leftrightarrow \qquad \sum_{\substack{k=0;\\ O_k = o_j}}^{L} P(S_k = s_i \,|\, \boldsymbol{o},\, \lambda^{old}) \, \frac{1}{b_{ij}} - \varphi_i = 0 \tag{127}$$

$$\Leftrightarrow \qquad b_{ij} = \frac{\displaystyle\sum_{\substack{k=0;\\ O_k = o_j}}^{L} P(S_k = s_i \,|\, \boldsymbol{o},\, \lambda^{old})}{\varphi_i} \tag{128}$$

Summing Equation 128 over j yields φ_i:

$$\sum_{j=1}^{M} \frac{\displaystyle\sum_{\substack{k=0;\\ O_k = o_j}}^{L} P(S_k = s_i \,|\, \boldsymbol{o},\, \lambda^{old})}{\varphi_i} \stackrel{!}{=} 1 \tag{129}$$

$$\Leftrightarrow \qquad \varphi_i = \sum_{j=1}^{M} \sum_{\substack{k=0;\\ O_k = o_j}}^{L} P(S_0 = s_i \,|\, \boldsymbol{o},\, \lambda^{old}) = \sum_{k=0}^{L} P(S_0 = s_i \,|\, \boldsymbol{o},\, \lambda^{old}) \tag{130}$$

and hence

$$b_{ij} \;=\; \frac{\displaystyle\sum_{\substack{k=0;\\ O_k = o_j}}^{L} P(S_k = s_i \,|\, \boldsymbol{o},\, \lambda^{old})}{\displaystyle\sum_{k=0}^{L} P(S_0 = s_i \,|\, \boldsymbol{o},\, \lambda^{old})} \;=\; \frac{\displaystyle\sum_{\substack{k=0;\\ O_k = o_j}}^{L} \gamma_k(i)}{\displaystyle\sum_{k=0}^{L} \gamma_k(i)} \tag{131}$$

Maximizing Transition Probabilities. Simplifying the third term (114) yields:

$$\sum_{\boldsymbol{s} \in \boldsymbol{S}} \sum_{k=1}^{L} \log\left[v_{s_{k-1}\, s_k}(d_k)\right] P(\boldsymbol{s} \,|\, \boldsymbol{o},\, \lambda^{old}) = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{L} \log\left[v_{ij}(d_k)\right] P(S_{k-1} = s_i,\, S_k = s_j \,|\, \boldsymbol{o},\, \lambda^{old}) \tag{132}$$

In contrast to π and $b_i(o_j)$, the maximization problem is more complex, since $v_{ij}(d_k)$ is a function of $\boldsymbol{P}$ and $\boldsymbol{D}(t)$:

$$v_{ij}(d_k) = f_{ij}\big(\boldsymbol{P},\, \boldsymbol{D}(d_k)\big) = \begin{cases} p_{ij}\; d_{ij}(d_k) & \text{if } i \neq j \\ 1 - \displaystyle\sum_{h=1}^{N} p_{ih}\; d_{ih}(d_k) & \text{if } i = j \end{cases} \tag{133}$$

where $p_{ii} = 0$ is assumed.

Maximizing p_{ij} is very similar to the estimation of a_{ij} in discrete-time HMMs. The constraint for optimization is:

$$\forall\, i: \sum_j p_{ij} \stackrel{!}{=} 1, \quad p_{ii} = 0 \tag{134}$$

Since $p_{ii} = 0$, we only have to consider the first line of Equation 133 where $i \neq j$.

In order to shorten notations, let

$$P(s_i,\, s_j,\, k \,|\, \boldsymbol{o},\, \lambda^{old}) := P(S_{k-1} = s_i,\, S_k = s_j \,|\, \boldsymbol{o},\, \lambda^{old}). \tag{135}$$

and hence the Lagrangian optimization equation is:

$$\frac{\partial}{\partial p_{ij}} \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{L} \log\left[v_{ij}(d_k)\right] P(s_i,\, s_j,\, k \,|\, \boldsymbol{o},\, \lambda^{old}) - \sum_{i=1}^{N} \varphi_i \Big(\sum_{\substack{j=1;\\ j\neq i}}^{N} p_{ij} - 1 \Big) \right) = 0 \tag{136}$$

$$\Leftrightarrow \quad \frac{\partial}{\partial f_{ij}} \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{L} \log\left[f_{ij}\big(\boldsymbol{P},\, \boldsymbol{D}(d_k)\big)\right] P\,(\,\cdot\,) \right) \frac{\partial f_{ij}\big(\boldsymbol{P},\, \boldsymbol{D}(d_k)\big)}{\partial p_{ij}} - \varphi_i = 0 \tag{137}$$

$$\Leftrightarrow \quad \sum_{k=1}^{L} \left(\frac{1}{p_{ij}\; d_{ij}(d_k)}\, P(s_i,\, s_j,\, k \,|\, \boldsymbol{o},\, \lambda^{old})\; d_{ij}(d_k) \right) - \varphi_i = 0 \tag{138}$$

$$\Leftrightarrow \quad p_{ij} = \frac{\displaystyle\sum_{k=1}^{L} P(s_i,\, s_j,\, k \,|\, \boldsymbol{o},\, \lambda^{old})}{\varphi_i} \tag{139}$$

It can be seen from Equation 139 that d_{ij} are cancelled out and hence p_{ij} can be determined independently from $d_{ij}(t)$. The solution is found by summing up p_{ij} and solving for φ_i similar to Equation 129:

$$p_{ij} = \frac{\displaystyle\sum_{k=1}^{L} P(S_{k-1} = s_i,\, S_k = s_j \,|\, \boldsymbol{o},\, \lambda^{old})}{\displaystyle\sum_{\substack{j=1;\\ j\neq i}}^{N} \sum_{k=1}^{L} P(S_{k-1} = s_i,\, S_k = s_j \,|\, \boldsymbol{o},\, \lambda^{old})} = \frac{\displaystyle\sum_{k=0}^{L-1} \xi_k(i,j)}{\displaystyle\sum_{\substack{j=1;\\ j\neq i}}^{N} \sum_{k=0}^{L-1} \xi_k(i,j)} \tag{140}$$

Since $\xi_k(i,j)$ builds on $g_{ij}(t)$ and $g_{ii}(t)$ is equal to zero by definition, the restriction $j \neq i$ can be omitted.

Maximizing $d_{ij}(t)$ is more complicated than maximizing p_{ij}. The reason for this is that in the case of p_{ij}, the "balance" between all outgoing transitions can be taken care of by the Lagrangian. Additionally, by defining p_{ii} to be equal to zero it was possible to focus only on the case $i \neq j$ and hence not to consider $v_{ii}(t)$ (second case in Equation 133). However, this approach is not admissible

for $d_{ij}(t)$ since

$$v_{ii}(t) = 1 - \sum_{h=1}^{N} p_{ih}\, d_{ih}(d_k) \not\equiv 0 \tag{141}$$

expressing the fact that $v_{ii}(t)$ is depending on $v_{ij}(t);\ i \neq j$.

The informal explanation for why an independent treatment of all durations $i \neq j$ is not admissible is that if all $d_{ij}(d_k);\ i \neq j$ were treated independently, the maximum of Equation 132 with respect to $d_{ij}(dk)$ would be that $d_{ij}(t) \equiv 1;\ t > 0$. Since $d_{ij}(t)$ are cumulative probability distributions specifying the probability that a transition takes place at time $T \leq t$, data likelihood (Equation 132) is maximal if a probability of 1 is assigned to all durations. In this case, v_{ii} would be equal to zero and the GHSMM would actually be turned into a standard HMM, building on a DTMC.

Therefore, data likelihood has to be maximized for duration densities $d_{ij}{}'(t)$ leading to the following optimization problem:

$$\bar{d_{ij}}'(t) = \underset{d_{ij}{}'(t)}{\arg\max} \sum_{k=1}^{L} \log\left[p_{ij}\, d_{ij}{}'(d_k)\right] P(S_{k-1} = s_i,\, S_k = s_j \,|\, \boldsymbol{o},\, \lambda^{old}) \tag{142}$$

The interpretation of this is intuitive: The probability that a transition from state s_i to s_j of duration t occurs has to be adjusted in such a way that the distribution fits very well for durations of transitions that are very likely. Likelihood of a transition $s_i \rightarrow s_j$ at time t_{k-1} having duration of $d_k = t_k - t_{k-1}$ is expressed by $P(S_{k-1} = s_i,\, S_k = s_j \,|\, \boldsymbol{o},\, \lambda^{old})$. The optimization problem given in Equation 142 is in fact a classical maximum likelihood optimization problem with weighted data points. A visualization is provided by Figure 16 on page 32).

Since in GHSMMs, duration distributions are modeled as a weighted mixture of probability distributions, an embedded EM algorithm is used to fit the parameters of the kernels to the data. From the definition of transition duration distributions (Equation 45) can be inferred that the densities have the following similar form:

$$d_{ij}{}'(t) = \sum_{r=0}^{R} w_{ij,r}\, \kappa'_{ij,\,r}(t|\theta_{ij,r}) \tag{143}$$

$$\text{s.t.} \sum_{r=0}^{R} w_{ij,r} = 1 \tag{144}$$

where $w_{ij,r}$ denotes the weight for kernel r and $\kappa'_{ij,\,r}(t|\theta_{ij,r})$ denotes a probability density with parameters $\theta_{ij,r}$.

Considering that transition probabilities can be expressed by

$$P(S_{k-1} = s_i,\, S_k = s_j \,|\, \boldsymbol{o},\, \lambda^{old}) = \xi_{k-1}(i,j) \tag{145}$$

and keeping in mind that constant factors p_{ij} do not affect the arg max operator, a more finegrained representation of the maximization problem given in Equation 142 is:

$$\bar{\theta}_{ij} = \underset{\theta_{ij}}{\arg\max} \sum_{k=1}^{L} \log\Big[\sum_{r=0}^{R} w_{ij,r}\, \kappa'_{ij,\,r}(d_k, \theta_{ij,r})\Big]\, \xi_{k-1}(i,j) \tag{146}$$

Maximization of Equation 146 follows very closely the optimization of a mixture of Gaussians, which can be found in many textbooks (e.g., [25]), except that data points are weighted here.

As stated before, EM algorithms assume the hidden data to be known (forming the complete dataset) and optimize the expected value with respect to the hidden data. In this case the unobservable data is the knowledge, to which kernel r each delay is assigned. Hence the Q-function for the parameters of duration distributions for one transition $s_i \rightarrow s_j$ is:

$$Q_{d_{ij}}(\theta_{ij}, \theta_{ij}^{old}) = \int_{\boldsymbol{r}} \log\left[P(\boldsymbol{d}, \boldsymbol{r}|\theta_{ij})\right]\, P(\boldsymbol{r} \,|\, \boldsymbol{d},\, \theta_{ij}^{old})\, d\boldsymbol{r} \tag{147}$$

$$= \sum_{k=1}^{L}\sum_{r=0}^{R} P(d_k)\, \log\left[P(r)\, P(d_k \,|\, r,\, \theta_{ij,r})\right]\, P\big(r \,|\, d_k,\, \theta_{ij}^{old}\big) \tag{148}$$

$$= \sum_{k=1}^{L}\sum_{r=0}^{R} \xi_{k-1}(i,j)\, \log\left[w_{ij,r}\, {\kappa_{ij,r}}'(d_k|\theta_{ij,r})\right]\, P\big(r \,|\, d_k,\, \theta_{ij}^{old}\big) \tag{149}$$

where $P(\boldsymbol{d}, \boldsymbol{r}|\theta_{ij})$ denotes weighted data likelihood for all delays $\boldsymbol{d} = [d_k]$ and kernel assignment vector $\boldsymbol{r} = [r_k]$, where each delay is weighted by $P(d_k)$.

From Bayes' theorem follows that

$$P\left(r \,|\, d_k,\, \theta_{ij}^{old}\right) = \frac{P(d_k \,|\, \theta_{ij,r}^{old})\, P(r)^{old}}{\sum_r P(d_k \,|\, \theta_{ij,r}^{old})\, P(r)^{old}} = \frac{\kappa'_{ij,r}(d_k \,|\, \theta_{ij,r}^{old})\, w_{ij,r}^{old}}{\sum_r \kappa'_{ij,r}(d_k \,|\, \theta_{ij,r}^{old})\, w_{ij,r}^{old}} \tag{150}$$

Kernel weights $\boldsymbol{w_{ij,r}}$ Derivation of Equation 149 with respect to kernel weights $w_{ij,r}$ and including a Lagrangian multiplier for the constraint yields:

$$\frac{\partial}{\partial w_{ij,r}} \left[\sum_{k=1}^{L}\sum_{r=0}^{R} \xi_{k-1}(i,j)\, \log\left[w_{ij,r}\, {\kappa_{ij,r}}'(d_k)\right]\, P\big(r \,|\, d_k,\, \theta_{ij}^{old}\big) \; - \varphi\Big(\sum_{r=0}^{R} w_{ij,r} - 1\Big)\right] \stackrel{!}{=} 0 \tag{151}$$

$$\Leftrightarrow \quad w_{ij,r} = \frac{\sum_{k=1}^{L} \xi_{k-1}(i,j)\, P\big(r \,|\, d_k,\, \theta ij^{old}\big)}{\varphi} \tag{152}$$

and solving for φ yields:

$$w_{ij,r} = \frac{\sum_{k=1}^{L} \xi_{k-1}(i,j)\, P\big(r \,|\, d_k,\, \theta_{ij}^{old}\big)}{\sum_{k=1}^{L} \xi_{k-1}(i,j)} \tag{153}$$

Kernel parameters $\boldsymbol{\theta_{ij,r}}$ In order to determine kernel parameters $\theta_{ij,r}$, Equation 149 must be derived with respect to $\theta_{ij,r}$. Since this derivation is depending on the particular form of the kernel

that is used, only a general optimization equation can be derived here:

$$\begin{aligned}\frac{\partial\, Q_{d_{ij}}}{\partial\, \theta_{ij,r}} &= \sum_{k=1}^{L}\sum_{r=0}^{R} \xi_{k-1}(i,j)\, \log\left[w_{ij,r}\, \kappa'_{ij,r}(d_k|\theta_{ij,r})\right]\, P\big(r \mid d_k,\, \theta_{ij}^{old}\big) \\ &= \sum_{k=1}^{L} \xi_{k-1}(i,j)\, \frac{1}{\kappa'_{ij,r}(d_k|\theta_{ij,r})}\, \frac{\partial\, \kappa'_{ij,r}(d_k|\theta_{ij,r}^{old})}{\partial\, \theta_{ij,r}}\, \frac{\kappa'_{ij,r}(d_k|\theta_{ij,r}^{old})\, w_{ij,r}^{old}}{\sum_r \kappa'_{ij,r}(d_k|\theta_{ij,r}^{old})\, w_{ij,r}^{old}} \;\stackrel{!}{=}\; 0\end{aligned} \tag{154}$$

Appendix A provides the solution to Equation 154 for some well-known distributions. In case that no formal solution exists, numerical solution techniques may be used.

10 Complexity

As it is the case for most machine learning techniques, two cases have to be distinguished for complexity assessment:

- Training of the model. Training is in most cases performed offline using previously recorded training data. Although feasibility restrictions exist, training complexity is not as critical as application complexity.
- Application of the model. After training, the obtained model is applied, which means that it is supplied with new, unknown data. The new data may either arrive continuously during runtime or a huge amount of data must be processed. Therefore, for both types of application complexity is much more critical. This is especially true for applications where processing deadlines need to be met.

In terms of hidden Markov models, application complexity refers to the forward, backward and Viterbi algorithm while training complexity refers to the Baum-Welch algorithm.

Application complexity. The forward algorithm of standard discrete-time HMMs belongs to the class $\mathcal{O}(L\,N^2)$ since for each of the $L+1$ symbols of the sequence, a sum over N terms has to be computed for each of the N states. However, this is only true if really all predecessors are taken into account. If the implementation uses adjacency lists, it is only true if the topology is a clique. This is really rare: many applications use a left-to-right structure where only a few predecessors have to be summed up, leading to a complexity of roughly $\mathcal{O}(LN)$.

Complexity of the Viterbi algorithm is the same since the sum of the forward algorithm is simply replaced by a maximum operator, which also has to investigate all N predecessor in order to select the maximum value.

Complexity of the Backward algorithm is also equal to the forward algorithm, although multiplication of $b_i(O_t)$ cannot be factored out – but since constant factors do not change the class of complexity in the $\mathcal{O}$-calculus, the same class results.

Turning to GHSMMs, the algorithms belong to the same class of complexity, since the only difference between the algorithms is that a_{ij} is replaced by $v_{ij}(d_k)$. More precisely: a single multiplication is substituted by computations that are a bit more complex. To reiterate,

$$v_{ij}(d_k) = \begin{cases} p_{ij}\, d_{ij}(d_k) & \text{if } i \neq j \\ 1 - \sum_{h=1}^{N} p_{ih}\, d_{ih}(d_k) & \text{if } i = j \end{cases} \tag{155}$$

where each $d_{ij}(d_k)$ is a mixture of kernels:

$$d_{ij}(t) = \sum_{r=0}^{R} w_{ij,r}\, \kappa_{ij,r}(t|\theta_{ij,r}) \tag{156}$$

Therefore, for cases $i \neq j$,

$$a_{ij} \Leftrightarrow p_{ij} \sum_{r=0}^{R} w_{ij,r}\, \kappa_{ij,r}(t|\theta_{ij,r}) \tag{157}$$

$\kappa_{ij,r}(t)$ are cumulative probability distributions that have to be evaluated. Depending on the type of distribution this might involve some computations since for, e.g., Gaussian distributions, there is no formula for the cumulative distribution. However, since R remains constant irrespective of N and L, it is a constant factor and complexity in terms of the $\mathcal{O}$-calculus is the same as for discrete-time HMMs. But even if constant factors are concerned, the complexity overhead is rather small since R is in many cases a very small number (less than ten). For $i = j$, computations are even less costly if the products $p_{ij}\, d_{ij}(t);\ i \neq j$ are summed up "on the fly".

Training complexity. Estimating overall complexity of the Baum-Welch algorithm is a hard task since the number of iterations is depending on convergence with respect to data likelihood – and this depends on many factors such as

- model initialization, which is in many cases random
- quality and quantity of the training data
- appropriateness of the HMM assumptions
- appropriateness of the HMM topology
- number of parameters of the HMM, determined by $N + N^2 + NM \in \mathcal{O}(N^2)$ in case of a fully connected HMM.[7]

Due to especially the last point, it is assumed that the number of iterations is $\mathcal{O}(N^2)$, which in reality is a quite loose upper bound. In fact convergence might even be better with an increasing amount of training data. M is assumed to be application dependent and hence constant.

[7] M denotes the size of the observation symbol alphabet.

Nevertheless, complexity of one reestimation step can be determined: The E-Step of the EM algorithm involves execution of the forward-backward algorithm. Then, to accomplish the M-step, reestimation of

$\boldsymbol{\pi}$ requires $\mathcal{O}(N)$ steps

$\boldsymbol{B}$ requires $\mathcal{O}(NL)$ steps[8]

$\boldsymbol{A}$ requires $\mathcal{O}(N^2L)$ steps

for each sequence. Similar to the case for the forward-backward algorithm, complexity of real models (e.g., left-to-right topology) is less. Putting this together with the number of iterations, overall training complexity is of class $\mathcal{O}(N^4L)$.

Turning to GHSMMs, reestimation of $\boldsymbol{\pi}$ and $\boldsymbol{B}$ remains the same and reestimation of $\boldsymbol{P}$ equals reestimation of $\boldsymbol{A}$. Hence the complexity overhead of GHSMMs derives from estimation of $d_{ij}(t)$. There may be $N(N-1)$ different $d_{ij}(t)$, each consisting of a small number of kernels (e.g., $0 \leq R < 10$), which are reestimated by an embedded EM algorithm. The number of iterations is not depending on N and will rather decrease with increasing L, hence the number of iterations is assumed to be of $\mathcal{O}(1)$. Each reestimation step involves $(R+1)\,\mathcal{O}(L) = \mathcal{O}(L)$ steps. Putting all this together, reestimation of transition durations is of complexity

$$N(N-1) * \mathcal{O}(1) * \mathcal{O}(L) = \mathcal{O}(N^2L) \tag{158}$$

Assuming the number of iterations of the outer EM algorithm to be again $\mathcal{O}(N^2)$, this yields an overall complexity of $\mathcal{O}(N^4L)$.

11 Conclusions

Temporal sequence processing for event-driven time series can be accomplished by three different approaches: (a) the time line can be split up into time slots, (b) delay events can be defined in order to obtain a periodic time series or (c) continuous time can be incorporated into the model that is being used. This report has focused on the third approach. More specifically, it has introduced a new approach how hidden Markov models (HMMs) can be extended to continuous time. The approach builds on turning the stochastic process of hidden state traversal into a semi-Markov process. Since a large variety of transition duration probability distributions can be used, the resulting model is called Generalized Hidden Semi-Markov Model (GHSMM).

It has been shown in the report, how GHSMMs can be used to address the principle problems of temporal sequence processing: sequence generation, sequence recognition and sequence prediction. Additionally, it has been shown how the parameters of GHSMMs can be determined from a set

[8] M is considered constant, here.

of training data. The training procedure uses an embedded expectation-maximization algorithm, which can under certain conditions be simplified to standard distribution moment estimation. A proof of convergence has been described and complexity of the entire approach has been assessed.

References

[1] Ron Sun. Introduction to sequence learning. In Ron Sun and C. Lee Giles, editors, *Sequence Learning: Paradigms, Algorithms, and Applications*, volume 1828 of *Lecture Notes in Computer Science*, pages 1–11. Springer, Berlin / Heidelberg, 2001.

[2] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice Hall, Englewood Cliffs, New Jersey, third edition, 1994.

[3] C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: alternative data models. In *IEEE Proceedings of the 1999 Symposium on Security and Privacy*, pages 133–145, 1999.

[4] A. Daidone, F. Di Giandomenico, A. Bondavalli, and S. Chiaradonna. Hidden Markov models as a support for diagnosis: Formalization of the problem and synthesis of the solution. In *IEEE Proceedings of the 25th Symposium on Reliable Distributed Systems (SRDS 2006)*, Leeds, UK, Oct. 2006.

[5] Wei Wei, Bing Wang, and Don Towsley. Continuous-time hidden Markov models for network performance evaluation. *Performance Evaluation*, 49(1-4):129–146, 2002.

[6] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall, Upper Saddle River, NJ, USA, 2001.

[7] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

[8] Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press, Cambridge, UK, 1998.

[9] M. Russell and A. Cook. Experimental evaluation of duration modelling techniques for automatic speech recognition. In *IEEE Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP '87)*, volume 12, pages 2376–2379, Apr. 1987.

[10] Shun-Zheng Yu, Zhen Liu, M. S. Squillante, Cathy Xia, and Li Zhang. A hidden semi-Markov model for web workload self-similarity. In *IEEE Proceedings of 21st International Performance, Computing, and Communications Conference*, pages 65–72, 2002.

[11] D. R. Cox and H. D. Miller. *The Theory of Stochastic Processes.* Chapman and Hall, London, UK, first edition, 1965.

[12] J. Ferguson. Variable duration models for speech. In *Proceedings of the Symposium on the Application of HMMs to Text and Speech*, pages 143–179, 1980.

[13] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb. 1989.

[14] Martin J. Russell and R. K. Moore. Explicit modelling of state occupancy in hidden Markov models for automatic speech recognition. In *IEEE Proceedings of Int. Conf. on Acoustics, Speech and Signal Processing*, pages 5–8, Mar. 1985.

[15] S. E. Levinson. Continuously variable duration hidden Markov models for automatic speech recognition. *Computer Speech and Language*, 1(1):29–45, 1986.

[16] C.D. Mitchell and L.H. Jamieson. Modeling duration in a hidden Markov model with the exponential family. In *IEEE Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP-93)*, volume 2, pages 331–334, Apr. 1993.

[17] Weon-Goo Kim, Jeung-Yoon Choi, and Dae Hee Youn. HMM with global path constraint in viterbi decoding for isolatedword recognition. In *IEEE Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP-94)*, volume 1, pages 605–608, Apr. 1994.

[18] A. E. Cook and M. J. Russell. Improved duration modeling in hidden Markov models using series-parallel configurations of states. *Proc. Inst. Acoust.*, 8:299–306, 1986.

[19] A. Noll and H. Ney. Training of phoneme models in a sentence recognition system. In *IEEE Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP '87)*, volume 12, pages 1277–1280, Apr. 1987.

[20] Janne Pylkkönen. Phone duration modeling techniques in continuous speech recognition. Master's thesis, Helsinki University of Technology, Department of Computer Science and Engineering, Laboratory of Computer and Information Science, 2004.

[21] Xue Wang. Durationally constrained training of hmm without explicit state durational pdf. In *Proceedings of the Institute of Phonetic Sciences, University of Amsterdam*, volume 18, pages 111–130, 1994.

[22] Antonio Bonafonte, Josep Vidal, and Albino Nogueiras. Duration modeling with expanded hmm applied to speech recognition. In *IEEE Proceedings of the Fourth International Conference on Spoken Language (ICSLP 96)*, volume 2, pages 1097–1100, Oct. 1996.

[23] Padma Ramesh and Jay G. Wilpon. Modeling state durations in hidden Markov models for automatic speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-92)*, volume 1, pages 381–384, 1992.

[24] Carl Mitchell, Mary Harper, and Leah Jamieson. On the complexity of explicit duration hmm's. *IEEE Transactions on Speech and Audio Processing*, 3(3):213–217, May 1995.

[25] Bert-Uwe Köhler. *Konzepte der statistischen Signalverarbeitung.* Springer, Berlin, Heidelberg, Germany, 2005.

[26] Vidyadhar G. Kulkarni. *Modeling and Analysis of Stochastic Systems.* Chapman and Hall, London, UK, first edition, 1995.

[27] B. H. Juang, S. E. Levinson, and M. M. Sondhi. Maximum likelihood estimation for multivariate mixture observations of Markov chains. *IEEE Transactions on Information Theory*, 32(2):307–309, 1986.

[28] L. A. Liporace. Maximum likelihood estimation for multivariate observations of Markov sources. *IEEE Transactions on Information Theory*, 28(5):729–734, Sep. 1982.

[29] John Aldrich. R.A. Fisher and the making of maximum likelihood 1912–1922. *Statistical Science*, 12(3):162–176, 1997.

[30] Rainer Schlittgen. *Einführung in die Statistik: Analyse und Modellierung von Daten.* Oldenbourg-Wissenschaftsverlag, München, Wien, 9 edition, 2000.

[31] Leonard E. Baum and George R. Sell. Growth transformations for functions on manifolds. *Pacific Journal of Mathematics*, 27(2):211–227, 1968.

[32] 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*, 39(1):1–38, 1977.

[33] T. Minka. Expectation-Maximization as lower bound maximization. Tutorial published on the web at `http://research.microsoft.com/users/minka/papers/minka-em-tut.ps.gz`, 1998.

[34] M. S. Bazaraa and C. M. Shetty. *Nonlinear Programming.* John Wiley and Sons, New York, 1979.

[35] Johan Ludwig William Valdemar Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30(1):175–193, Dec. 1906.

[36] Jeff A. Bilmes. A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian Mixture and Hidden Markov Models. Tech. report ICSI-TR-97-021, U.C. Berkeley, International Computer Science Institute, Berkeley, CA, Apr. 1998.

Appendix

A Maximum Likelihood Estimates for Selected Distributions

Formulas for maximum likelihood estimation of probability distribution parameters are provided by many textbooks. However, in the case of GHSMMs, estimation from *weighted* set of data is required. This appendix summarizes the formulas for some selected distributions. Maximum likelihood estimation is derived in detail for the exponential distribution while only resulting formulas are reported for the others.

A.1 Exponential Distribution

The exponential distribution is depending on one parameter λ. Its density has the form

$$f(x) = \lambda \, e^{-\lambda x} \tag{159}$$

Maximum likelihood estimation for a weighted set of data points is

$$\hat{\lambda} = \arg\max_{\lambda} \sum_{i=1}^{N} P(x_i) \; \log\big(f(x_i)\big) \tag{160}$$

where $P(x_i)$ is the weight for data point x_i. Maximization is performed by derivation with respect to λ:

$$\frac{\partial}{\partial\lambda} \sum_{i=1}^{N} P(x_i) \; \log\big(f(x_i)\big) \stackrel{!}{=} 0 \tag{161}$$

$$\Leftrightarrow \quad \sum_{i=1}^{N} P(x_i) \; \frac{1}{f(x_i)} \; \frac{\partial}{\partial\lambda} f(x_i) \stackrel{!}{=} 0 \tag{162}$$

$$\Leftrightarrow \quad \sum_{i=1}^{N} P(x_i) \; \frac{1}{f(x_i)} \; f(x_i) \left(\frac{1}{\lambda} - x_i\right) \stackrel{!}{=} 0 \tag{163}$$

$$\Leftrightarrow \quad \hat{\lambda} = \frac{\sum_{i=1}^{N} P(x_i)}{\sum_{i=1}^{N} P(x_i) \; x_i} \tag{164}$$

which is actually the inverse of a weighted mean corresponding to the fact that the expectation value for exponential distributions is $\frac{1}{\lambda}$.

A.2 Normal Distribution

The Normal distribution's density is

$$f(x) = \frac{1}{\sqrt{2\,\pi\,\sigma^2}} \; \exp\Big(-\frac{(x-\mu)^2}{2\,\sigma^2}\Big) \tag{165}$$

The maximum likelihood estimation for the mean value yields:

$$\hat{\mu} = \frac{\sum_{i=1}^{N} P(x_i)\ x_i}{\sum_{i=1}^{N} P(x_i)} \tag{166}$$

and

$$\hat{\sigma^2} = \frac{\sum_{i=1}^{N} P(x_i)\ (x_i - \hat{\mu})^2}{\sum_{i=1}^{N} P(x_i)} \tag{167}$$

A.3 Log-Normal Distribution

The Log-Normal distribution's density is

$$f(x) = \frac{1}{x\sqrt{2\,\pi\,\sigma^2}}\ \exp\Big(-\frac{(\log x - \mu)^2}{2\,\sigma^2}\Big) \tag{168}$$

The maximum likelihood estimation for the mean value yields:

$$\hat{\mu} = \frac{\sum_{i=1}^{N} P(x_i)\ \log x_i}{\sum_{i=1}^{N} P(x_i)} \tag{169}$$

and

$$\hat{\sigma^2} = \frac{\sum_{i=1}^{N} P(x_i)\ (\log x_i - \hat{\mu})^2}{\sum_{i=1}^{N} P(x_i)} \tag{170}$$

A.4 Pareto Distribution

Probability density of the Pareto distribution is:

$$f(x) = \frac{k\ x_{min}^k}{x^{k+1}} = \frac{k}{x_{min}}\Big(\frac{x_{min}}{x}\Big)^{k+1} \tag{171}$$

In order to estimate both parameters x_{min} and k, we have:

$$\hat{x}_{min} = \min_i x_i \tag{172}$$

and

$$\hat{k} = \frac{\sum_{i=1}^{N} P(x_i)}{\sum_{i=1}^{N} P(x_i)\ \big(\log(x_i) - \log(\hat{x}_{min})\big)} \tag{173}$$

A.5 Gamma Distribution

The Gamma distribution's density has the form:

$$f(x) = x^{k-1} \frac{\exp(-\frac{x}{\theta})}{\theta^k \, \Gamma(k)} \tag{174}$$

However, finding optimal estimates for θ and k is a bit more complicated, since no formal solution can be found. However, using the approximation

$$\log(k) - \Gamma(k) \approx \frac{1}{k} \left(\frac{1}{2} + \frac{1}{12\,k + 2} \right) \tag{175}$$

the approximately optimal estimate for k is

$$\hat{k} \approx \frac{3 - s + \sqrt{(s-3)^2 + 24s}}{12s} \tag{176}$$

where

$$s := \log \left(\frac{\sum_{i=1}^{N} P(x_i) \; x_i}{\sum_{i=1}^{N} P(x_i)} \right) - \frac{\sum_{i=1}^{N} P(x_i) \; \log x_i}{\sum_{i=1}^{N} P(x_i)} \tag{177}$$

It can be shown that this estimate is within a 1.5% bound of the true maximum. The approximate estimate could be used as starting point for a Newton-Raphson numerical optimization. However, since the estimation is part of an EM algorithm, an increase in data likelihood is already sufficient.[9] Therefore, the approximate value of Equation 176 is sufficient here.

Derivation of the likelihood function with respect to θ yields:

$$\hat{\theta} = \frac{\sum_{i=1}^{N} P(x_i) \; x_i}{\hat{k} \sum_{i=1}^{N} P(x_i)} \tag{178}$$

[9] The algorithm is then called Generalized Expectation Maximization (GEM)