

# GRAPHICAL-DURATION HIDDEN MARKOV MODEL

MSC THESIS IN APPLIED MATHEMATICS

LÁSZLÓ KERESZTES

SUPERVISOR: BALÁZS CSANÁD CSÁJI, PHD  
DEPARTMENT OF PROBABILITY THEORY AND STATISTICS,  
EÖTVÖS LORÁND UNIVERSITY (ELTE); AND  
INSTITUTE FOR COMPUTER SCIENCE AND CONTROL (SZTAKI)



EÖTVÖS LORÁND UNIVERSITY  
FACULTY OF SCIENCE  
2023

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Hidden Markov Models</b>	<b>3</b>
2.1	Structure . . . . .	3
2.2	Algorithms . . . . .	5
<b>3</b>	<b>EM learning</b>	<b>12</b>
3.1	EM learning in general . . . . .	12
3.2	EM learning for HMMs - Baum-Welch algorithm . . . . .	14
3.3	Complexity of the Baum-Welch algorithm . . . . .	18
<b>4</b>	<b>Graph representation of distributions</b>	<b>19</b>
4.1	Representation graphs . . . . .	19
4.2	Representation of distribution families . . . . .	24
<b>5</b>	<b>Graphical-Duration Hidden Markov Model</b>	<b>31</b>
5.1	Hidden Semi-Markov Models . . . . .	31
5.2	The GD-HMM . . . . .	31
<b>6</b>	<b>Learning the parameters of the GD-HMM</b>	<b>36</b>
6.1	M-step for transition parameters in categorical case . . . . .	39
6.2	Time complexity of M-step . . . . .	42
<b>7</b>	<b>Efficiency of representation in GD-HMM</b>	<b>44</b>
<b>8</b>	<b>Numerical experiments</b>	<b>50</b>
8.1	Monotonically increasing log-likelihood of GD-HMM . . . . .	51
8.2	Log-likelihood and AIC evaluation of GD-HMM . . . . .	51
8.3	Simulated manufacturing use case for GD-HMM . . . . .	53



## **Acknowledgement**

I would like to thank Balázs Csáji for his help, great ideas, guidance, and encouragement.

# 1 Introduction

In my thesis, I examine a well-applicable statistical model called Hidden Markov Model (HMM) [1,3,4]. Most applications of HMMs involve time series or signals; i.e. data that have time dimension. In the most common case, HMMs are used to infer an underlying hidden process that generates the observable data. The model assumes that the (hidden) process is in a (hidden) state for some discrete time units, generates data (that is specific to the state), then the process moves to another state. Therefore the amount of time spent in a hidden state is crucial both from theoretical and practical viewpoints. General HMM handles the duration with a geometric distribution with a state-specific parameter. This makes the model handy; the underlying process is a plain Markov chain.

A more advanced HMM would use a different family for modeling duration. Different variants of HMMs emerged to solve this issue, these models fall under the category of Hidden Semi-Markov Models [2, 5, 6]. HSMMs mostly use categorical distribution on  $\{1, \dots, D\}$ , where  $D$  is the maximum duration hyperparameter. HSMMs are applied in the following areas: protein structure prediction, internet traffic modeling, speech synthesis, change-point/end-point detection for semiconductor manufacturing, etc.

Categorical distribution is way more flexible, and even infinite-support discrete distributions can be approximated well with categorical distributions using a large  $D$ .

However, it seems like there is a gap between the simple geometric and the very flexible categorical with hyperparameter  $D$ . What if one knows that the duration distribution is negative binomial with known, fixed order  $N$  and unknown parameter  $p$ ? The geometric distribution is simply not flexible enough, and the categorical distribution is not efficient at all: e.g. with  $N = 10$  and  $p = 0.1$ , choosing a  $D$  that is capable of representing the duration 95% of the time results in  $D = 144$ . So, we learn  $D - 1 = 143$  parameters instead of 1 and

the resulting distribution is neither negative binomial, nor (necessarily) truncated negative binomial. Also, there is still a possibility that the truncation results in significant errors either in learning or in inference.

In my thesis, I establish a new framework, Graphical-Duration Hidden Markov Model (GD-HMM) that can be used with many distribution families. In the case of infinite-support distributions, there is no need to represent them as fixed-support categorical distributions. The motivation is to make the transition model part of HMM as flexible as the observation model part while maintaining computational efficiency.

In this framework, I model duration distributions with the distribution of the first passage time (to the absorption state) on a Markov chain. Representing a duration distribution with a graph (a Markov chain) is broad enough to include geometric, negative binomial, categorical, and several other distributions.

I show how these graphs can be merged to form an HSMM with the desired duration distributions and provide the appropriate Expectation-Maximization (EM) learning variant.

Using the framework for the categorical distribution results in a model for which the new EM gives back the state-of-the-art forward-backward computation efficiency [6]. It can be also shown that this efficiency is optimal in some sense.

Finally, I present results from numerical experiments on the implemented Graphical-Duration Hidden Markov Model. The results prove that the implementation was successful, and observation accuracy can be increased by using a GD-HMM instead of an HMM.

## 2 Hidden Markov Models

A Hidden Markov Model (HMM) can be viewed as a noisy observation of a Markov chain. This model emerged in the 1960s, and now it has important applications in signal processing, control theory, speech recognition, and sequential bioinformatics. In the HMM framework, there is a hidden Markov process that influences the observations, but we cannot observe it directly. Usually, the inference for this hidden process is the task to solve, where the hidden process is our real process of interest, such as a sequence of words in speech recognition or specific DNA regions in the DNA sequence. An HMM has a transition model and an observation model. The transition model controls the hidden process, at each time step, we stochastically move to the next hidden state. The observation model tells us how the observations are generated from a hidden state. Each hidden state has a data-generating distribution, these distributions came from a parametric family, such as Gaussians or Categorical distributions. The parametric family should be selected in advance, based on prior knowledge or empirical data distribution.

The transition model is a Markov chain, which can be viewed as a directed graph. The structure of the graph can be chosen according to domain expert knowledge. Building these expert thoughts correctly into the model makes it more reasonable and robust.

The next step would be to choose a parametric family for duration distributions by the experts and build this information into the model.

### 2.1 Structure

In my thesis, I deal with discrete-time, finite-state, categorical Hidden Markov Models, but I refer to them as HMMs. The theorems and proofs are designed for the categorical observation model, but the ideas apply to any other observation model.

**Definition 1.** (*Discrete-time, finite-state, categorical Hidden Markov Model*)

Let  $(Z_t)$  and  $(X_t)$  be discrete-time stochastic processes with  $t \geq 1$ .

Let  $Z_t : \Omega \rightarrow \{1, \dots, M\}$ , where  $\{1, \dots, M\}$  is the state space.

Let  $X_t : \Omega \rightarrow \{1, \dots, L\}$ , where  $\{1, \dots, L\}$  is the observation space.

The pair  $(Z_t, X_t)$  is a Hidden Markov Model if:

1.  $(Z_t)$  is a Markov process (that cannot be observed directly),
2.  $\forall t \forall x_t \in \{1, \dots, L\} :$

$$P(X_t = x_t | Z_s = z_s \forall s \geq 1, X_s = x_s \forall s \neq t) = P(X_t = x_t | Z_t = z_t).$$

We assume that the HMM is time-homogeneous ( $p(z_t = j | z_{t-1} = i)$  and  $p(x_t | z_t = i)$  are independent of  $t$ ).

An HMM is a pair of a hidden process, a discrete  $z_t \in \{1, \dots, M\}$  Markov chain in discrete time ( $t \in \{1, \dots, T\}$ ), and an observation model  $p(x_t | z_t)$ . The joint distribution has the form

$$p(z_{1:T}, x_{1:T}) = p(z_1) \prod_{t=2}^T p(z_t | z_{t-1}) \prod_{t=1}^T p(x_t | z_t).$$

The initial distribution  $\pi_i = p(z_1 = i)$  is a probability distribution on  $\{1, \dots, M\}$ .

The transition model  $A_{ij} = p(z_t = j | z_{t-1} = i)$  is independent of the time  $t$  (time-homogeneous).  $A$  is an  $M \times M$  matrix, also called the transition matrix.

The observation model represents discrete distributions, it is a matrix  $B$ , where  $B_{il} = p(x_t = l | z_t = i)$  for the  $l = 1, \dots, L$  categories and for the  $i = 1, \dots, M$  hidden states.

In the next chapters, we will consider HMMs as in Definition 1 ( $z_1, \dots, z_T \in \{1, \dots, M\}$  and  $x_1, \dots, x_T \in \{1, \dots, L\}$ ). The HMM has parameters  $\theta = (\pi, A, B)$ .

## 2.2 Algorithms

Given a predefined HMM with parameters  $\theta = (\pi, A, B)$  and the sequence of observed data  $x_{1:T}$ , the most basic inference tasks are filtering (with the forward algorithm), smoothing (with the forward-backward algorithm), and Viterbi decoding.

In filtering, we want to compute (online) the  $\alpha_t(i) = p(z_t = i | x_{1:t})$  belief state. The next forward dynamic programming algorithm shows the computation.

---

### Algorithm 1: Forward algorithm

---

**Input** : Observation sequence  $x_{1:T}$ ,

HMM parameters  $\theta = (\pi, A, B)$

**Output**:  $\alpha$  values,  $\log p(x_{1:T} | \theta)$  log-likelihood

$a_1(i) = B_{i,x_1} \pi_i$  for  $i = 1, \dots, M$

$Z_1 = \sum_{i=1}^M a_i$

$\alpha_1(i) = a_1(i)/Z_1$  for  $i = 1, \dots, M$

**for**  $t = 2, \dots, T$  **do**

$a_t(i) = B_{i,x_t} \sum_{j=1}^M A_{ji} \alpha_{t-1}(j)$  for  $i = 1, \dots, M$

$Z_t = \sum_{i=1}^M a_t(i)$

$\alpha_t(i) = a_t(i)/Z_t$  for  $i = 1, \dots, M$

**end**

$p(x_{1:T} | \theta) = \sum_{t=1}^T \log Z_t$

---

### Statement 1. (Forward algorithm)

*The Forward algorithm returns the correct  $\alpha$  values and  $\log p(x_{1:T} | \theta)$  log-likelihood.*

*Proof.* We prove this by induction. For  $t = 1$ :

$$\begin{aligned} \alpha_1(i) &= p(z_1 = i | x_1) = \frac{p(x_1 | z_1 = i) p(z_1 = i)}{p(x_1)} = \frac{B_{i,x_1} \pi_i}{p(x_1)} = \frac{a_1(i)}{p(x_1)} \\ &= \frac{a_1(i)}{\sum_i p(x_1 | z_1 = i) p(z_1 = i)} = a_1(i) / Z_1. \end{aligned}$$

Now assume we know for  $t - 1$ . Then:

$$\begin{aligned}
\alpha_t(i) &= p(z_t = i | x_{1:t}) = p(z_t = i | x_t, x_{1:t-1}) \\
&= \frac{p(z_t = i, x_t, x_{1:t-1})}{p(x_t, x_{1:t-1})} = \frac{p(x_t, z_t = i, x_{1:t-1})}{p(z_t = i, x_{1:t-1})} \frac{p(z_t = i, x_{1:t-1})}{p(x_{1:t-1})} \frac{p(x_{1:t-1})}{p(x_t, x_{1:t-1})} \\
&= p(x_t | z_t = i, x_{1:t-1}) p(z_t = i | x_{1:t-1}) \frac{1}{p(x_t | x_{1:t-1})}.
\end{aligned}$$

For the term  $p(x_t | z_t = i, x_{1:t-1})$ :

$$p(x_t | z_t = i, x_{1:t-1}) = p(x_t | z_t = i) = B_{i, x_t}.$$

For the term  $p(z_t = i | x_{1:t-1})$ :

$$\begin{aligned}
p(z_t = i | x_{1:t-1}) &= \sum_{j=1}^M p(z_t = i | z_{t-1} = j, x_{1:t-1}) p(z_{t-1} = j | x_{1:t-1}) \\
&= \sum_{j=1}^M A_{ji} \alpha_{t-1}(j).
\end{aligned}$$

For the term  $p(x_t | x_{1:t-1})$ :

$$\begin{aligned}
p(x_t | x_{1:t-1}) &= \sum_{i=1}^M p(x_t | z_t = i, x_{1:t-1}) p(z_t = i | x_{1:t-1}) \\
&= \sum_{i=1}^M a_t(i) = Z_t.
\end{aligned}$$

Combining the terms:

$$\alpha_t(i) = B_{i, x_t} \sum_{j=1}^M A_{ji} \alpha_{t-1}(j) / Z_t = a_t(i) / Z_t.$$

For the log-likelihood:

$$\log p(x_{1:T} | \theta) = \log p(x_{1:T}) = \sum_{t=1}^T \log p(x_t | x_{1:t-1}) = \sum_{t=1}^T \log Z_t.$$

□

In smoothing, we want to compute (offline) the  $\gamma_t(i) = p(z_t = i | x_{1:T})$  given all the data and this can be done by the forward algorithm and the backward algorithm. In the backward algorithm, we compute  $\beta_t(j) = p(x_{t+1:T} | z_t = j)$ . The backward algorithm is a backward DP, and then  $\gamma_t(j) \propto \alpha_t(j)\beta_t(j)$  can be get.

In learning, besides filtering and smoothing, computing the two-slice marginals  $\xi_{t,t+1}(i, j) = p(z_t = i, z_{t+1} = j | x_{1:T})$  is also essential. This can be done as  $\xi_{t,t+1}(i, j) \propto \alpha_t(i)A_{ij}\beta_{t+1}(j)p(x_{t+1}|z_{t+1} = j)$  from the already computed  $\alpha, \beta$  values.

---

**Algorithm 2:** Backward algorithm

---

**Input :** Observation sequence  $x_{1:T}$ ,

HMM parameters  $\theta = (\pi, A, B)$

**Output:**  $\beta$  values

$\beta_T(i) = 1$  for  $i = 1, \dots, M$

**for**  $t = T - 1, \dots, 1$  **do**

$\beta_t(i) = \sum_{j=1}^M \beta_{t+1}(j)B_{j,x_{t+1}}A_{ij}$  for  $i = 1, \dots, M$

**end**

---

**Statement 2.** (*Backward algorithm*) *The backward algorithm returns the correct  $\beta$  values.*

*Proof.* We prove the recursive formula.  $\beta_T$  is defined only for convenience.

If  $t < T - 1$ :

$$\begin{aligned}
\beta_t(i) &= p(x_{t+1:T}|z_t = i) = \sum_{j=1}^M p(z_{t+1} = j, x_{t+1}, x_{t+2:T}|z_t = i) \\
&= \sum_{j=1}^M p(x_{t+2:T}|z_{t+1} = j, x_{t+1}, z_t = i) p(z_{t+1} = j, x_{t+1}|z_t = i) \\
&= \sum_{j=1}^M p(x_{t+2:T}|z_{t+1} = j) p(x_{t+1}|z_{t+1} = j) p(z_{t+1} = j|z_t = i) \\
&= \sum_{j=1}^M \beta_{t+1}(j) B_{j, x_{t+1}} A_{ij}.
\end{aligned}$$

If  $t = T - 1$ :

$$\begin{aligned}
\beta_{T-1}(i) &= p(x_{T:T}|z_{T-1} = i) = \sum_{j=1}^M p(z_T = j, x_T|z_{T-1} = i) \\
&= \sum_{j=1}^M 1 \cdot p(x_T|z_T = j) p(z_T = j|z_{T-1} = i) \\
&= \sum_{j=1}^M \beta_T(j) B_{j, x_T} A_{ij}.
\end{aligned}$$

□

---

**Algorithm 3:**  $\gamma$  and  $\xi$  computation

---

**Input :**  $\alpha$  values,  $\beta$  values

**Output:**  $\gamma$  values,  $\xi$  values

**for**  $t = 1, \dots, T$  **do**

$c_t = \sum_{i=1}^M \alpha_t(i) \beta_t(i)$   
     $\gamma_t(i) = \alpha_t(i) \beta_t(i) / c_t$  for  $i = 1, \dots, M$

**end**

**for**  $t = 2, \dots, T$  **do**

$e_{t-1,t} = \sum_{i=1}^M \sum_{j=1}^M \alpha_{t-1}(i) B_{j,x_t} \beta_t(j) A_{ij}$   
     $\xi_{t-1,t}(i, j) = \alpha_{t-1}(i) B_{j,x_t} \beta_t(j) A_{ij} / e_{t-1,t}$  for  $i, j = 1, \dots, M$

**end**

---

**Statement 3.** ( $\gamma$  and  $\xi$  computation)

*The previous algorithm returns the correct  $\gamma$  and  $\xi$  values.*

*Proof.* We start with  $\gamma$ .

For  $t = T$ :

$$\gamma_T(i) = p(z_T = i | x_{1:T}) = \alpha_T(i) = \alpha_T(i) \beta_T(i).$$

For  $t < T$ :

$$\begin{aligned} \gamma_t(i) &= p(z_t = i | x_{1:T}) = \frac{1}{p(x_{t+1:T} | x_{1:t})} p(z_t = i, x_{t+1:T} | x_{1:t}) \\ &= \frac{1}{p(x_{t+1:T} | x_{1:t})} p(x_{t+1:T} | z_t = i) p(z_t = i | x_{1:t}) \\ &= \frac{1}{p(x_{t+1:T} | x_{1:t})} \beta_t(i) \alpha_t(i). \end{aligned}$$

For the constant  $p(x_{t+1:T} | x_{1:t})$ :

$$p(x_{t+1:T} | x_{1:t}) = \sum_{i=1}^M p(z_t = i, x_{t+1:T} | x_{1:t}) = \sum_{i=1}^M \beta_t(i) \alpha_t(i).$$

We proceed with the  $\xi$  values. We use that  $c_{t-1} = p(x_{t:T}|x_{1:t-1})$ . Then we have

$$\begin{aligned}
\xi_{t-1,t}(i, j) &= p(z_{t-1} = i, z_t = j | x_{1:T}) \\
&= \frac{1}{p(x_{t:T}|x_{1:t-1})} p(z_{t-1} = i, z_t = j, x_{t:T} | x_{1:t-1}) \\
&= \frac{1}{c_{t-1}} p(z_{t-1} = i | x_{1:t-1}) p(z_t = j, x_{t:T} | z_{t-1} = i, x_{1:t-1}) \\
&= \frac{1}{c_{t-1}} \alpha_{t-1}(i) p(z_t = j, x_{t:T} | z_{t-1} = i) \\
&= \frac{1}{c_{t-1}} \alpha_{t-1}(i) p(z_t = j | z_{t-1} = i) p(x_{t:T} | z_t = j, z_{t-1} = i) \\
&= \frac{1}{c_{t-1}} \alpha_{t-1}(i) A_{ij} p(x_t, x_{t+1:T} | z_t = j) \\
&= \frac{1}{c_{t-1}} \alpha_{t-1}(i) A_{ij} p(x_t | z_t = j) p(x_{t+1:T} | x_t, z_t = j) \\
&= \frac{1}{c_{t-1}} \alpha_{t-1}(i) A_{ij} B_{j,x_t} \beta_t(j).
\end{aligned}$$

For the constant  $p(x_{t:T}|x_{1:t-1})$  we have:

$$\begin{aligned}
c_{t-1} &= p(x_{t:T}|x_{1:t-1}) = \sum_{i=1}^M \sum_{j=1}^M p(z_{t-1} = i, z_t = j, x_{t:T} | x_{1:t-1}) \\
&= \sum_{i=1}^M \sum_{j=1}^M \alpha_{t-1}(i) A_{ij} B_{j,x_t} \beta_t(j) = e_{t-1,t}.
\end{aligned}$$

□

The time complexities of the previous algorithms are the following: both the forward and backward algorithms take  $\mathcal{O}(TM^2)$  time, further computation of  $\gamma$  requires  $\mathcal{O}(TM)$  time and further computation of  $\xi$  requires  $\mathcal{O}(TM^2)$  time. Therefore the complete computation of  $\gamma$  and  $\xi$  values takes  $\mathcal{O}(TM^2)$  time.

The time complexity can be substantially reduced if the graph of the  $(Z_t)$  Markov chain is sparse.

**Definition 2.** Let  $\theta = (\pi, A, B)$  an HMM. The number of edges in the HMM:

$$E = |\{(i, j) : A_{ij} > 0\}|.$$

For the  $E$  edge number, we have:

$$E = \sum_{i=1}^M \deg_{out}(i) = \sum_{i=1}^M \deg_{in}(i).$$

Now assume that we have a sparse HMM with  $E \ll M^2$  and for each  $i \in \{1, \dots, M\}$  we have  $Out(i) = \{j : A_{ij} > 0\}$  and  $In(i) = \{j : A_{ji} > 0\}$ .

Then, in the forward algorithm we can iterate over  $j \in In(i)$  instead of  $j \in \{1, \dots, M\}$ . This leads to a time complexity of  $\mathcal{O}(T \sum_{i=1}^M |In(i)|) = \mathcal{O}(TE)$ . In the backward algorithm we can iterate over  $j \in Out(i)$  and this leads to  $\mathcal{O}(TE)$  time. The further computation of  $\gamma$  remains  $\mathcal{O}(TM)$ , but the further computation of  $\xi$  can be reduced, because for  $A_{ij} = 0$ ,  $\xi_{t-1,t}(i, j) = 0$ . Therefore we should iterate over the set  $\{(i, j) : A_{ij} > 0\}$ , and the computation takes  $\mathcal{O}(TE)$  time.

In summary, the complete computation of  $\gamma$  and  $\xi$  can be done in  $\mathcal{O}(TE)$  time, and this is significantly less than  $\mathcal{O}(TM^2)$ , if the HMM is sparse.

Another important computation is the search for the maximum probability hidden state sequence that explains the data:

$$\arg \max_{z_{1:T}} p(x_{1:T} | z_{1:T}).$$

This can be done with an offline dynamic programming algorithm also known as Viterbi decoding. The Viterbi decoding is not presented here, it can be found in [3, 4].

### 3 EM learning

Learning in HMM means we want to learn the initial distribution  $p(z_1)$ , the transition probabilities  $p(z_t|z_{t-1})$ , and the parameters of the observation model.

Because of the usually unobserved hidden process, we cannot maximize directly the likelihood function, therefore an iterative approach called Expectation-Maximization is applied.

#### 3.1 EM learning in general

The idea of EM is the following. We usually want to maximize the log-likelihood of the observed data:

$$l(\theta) = \log p(x_{1:T}|\theta) = \log \left[ \sum_{z_{1:T}} p(x_{1:T}, z_{1:T}|\theta) \right].$$

This is hard to optimize, therefore instead we maximize the complete data log-likelihood:

$$l_c(\theta) = \log p(x_{1:T}, z_{1:T}|\theta).$$

This cannot be computed, since  $z_t$  are unknown. Define the expected complete data log-likelihood as the following:

$$Q(\theta; \theta^{n-1}) = E[l_c(\theta)|x_{1:T}, \theta^{n-1}] = E_{z_{1:T}|x_{1:T}, \theta^{n-1}}[l_c(\theta)] = E_{z_{1:T} \sim p(z_{1:T}|x_{1:T}, \theta^{n-1})}[l_c(\theta)].$$

Here, the  $z_t$  are replaced with their expected value conditioned on the data and the previous parameter set.

The idea of the EM is that since we do not know the actual values of  $z_t$ , starting from an initial guess of parameters, we can iteratively estimate  $z_t$  with probabilities from the parameters (and data), then estimate the parameters using the  $z_t$  estimates.

The condition is usually on the amount of gain in the  $Q$  function or the number of iterations.

---

**Algorithm 4:** Expectation-Maximization (EM) algorithm

---

**Input** : Observation sequence  $x_{1:T}$ ,

initial parameters  $\theta^0$

**Output:** Parameters  $\theta^N$

Until condition:

- E step: Compute  $Q(\theta; \theta^{n-1})$  or the expected sufficient statistics (for parameter update)
- M step:

$$\theta^n = \arg \max_{\theta} Q(\theta; \theta^{n-1})$$

---

The EM algorithm in general finds a local optimum (with certain assumptions) by increasing the observed data log-likelihood at every EM step. [1, 3]

**Statement 4.** (*EM increases the observed data log-likelihood*)

*For the  $(\theta^n)$  parameter series from the EM algorithm:*

$$l(\theta^{n+1}) \geq l(\theta^n).$$

*Proof.* Denote  $X = x_{1:T}$ ,  $Z = z_{1:T}$ . Denote the distribution  $q^n(Z) = p(Z|X, \theta^n)$ .

Let  $D$  denote the information divergence, and  $H$  the entropy function.

$$\begin{aligned} l(\theta) &= \log p(X|\theta) = \log p(X, Z|\theta) - \log p(Z|X, \theta) \\ &= \sum_Z q^n(Z) \log p(X, Z|\theta) - \sum_Z q^n(Z) \log p(Z|X, \theta) \\ &= Q(\theta; \theta^n) - \sum_Z q^n(Z) \log \left[ \frac{p(Z|X, \theta)}{q^n(Z)} q^n(Z) \right] \\ &= Q(\theta; \theta^n) + D(q^n(Z) || p(Z|X, \theta)) + H(q^n(Z)). \end{aligned}$$

This is true for every  $\theta$ . Now setting  $\theta = \theta^n$ :

$$\begin{aligned} l(\theta^n) &= Q(\theta^n; \theta^n) + D(q^n(Z) || p(Z|X, \theta^n)) + H(q^n(Z)) \\ &= Q(\theta^n; \theta^n) + H(q^n(Z)). \end{aligned}$$

By differentiating the two equations, we have:

$$\begin{aligned} l(\theta) - l(\theta^n) &= Q(\theta; \theta^n) - Q(\theta^n; \theta^n) + D(q^n(Z) || p(Z|X, \theta)) \\ &\geq Q(\theta; \theta^n) - Q(\theta^n; \theta^n). \end{aligned}$$

Selecting

$$\theta^{n+1} = \arg \max_{\theta} Q(\theta, \theta^n)$$

shows that  $l(\theta^{n+1}) \geq l(\theta^n)$ . □

One of the best practices is to use multiple randomized initializations for the EM algorithm and select the best parameters.

In some cases (e.g. with HMM) both the E-step and M-step have an analytical solution. This can be also true with different parameter constraints: e.g. with parameter tying, and re-parameterization.

### 3.2 EM learning for HMMs - Baum-Welch algorithm

Applying the EM algorithm for learning HMM parameters, the complete data log-likelihood is simply the log of the joint:

$$l_c(\theta) = \log p(z_1 | \theta) + \sum_{t=2}^T \log p(z_t | z_{t-1}, \theta) + \sum_{t=1}^T \log p(x_t | z_t, \theta).$$

The auxiliary  $Q(\theta; \theta^n)$  function has the following form:

$$\begin{aligned}
Q(\theta; \theta^n) &= E_{\underline{z} \sim p(\underline{z}|\underline{x}, \theta^n)}[l_c(\theta)] \\
&= E_{z_1 \sim p(z_1|\underline{x}, \theta^n)}[\log p(z_1|\theta)] + \\
&\quad + \sum_{t=2}^T E_{(z_{t-1}, z_t) \sim p((z_{t-1}, z_t)|\underline{x}, \theta^n)}[\log p(z_t|z_{t-1}, \theta)] + \\
&\quad + \sum_{t=1}^T E_{z_t \sim p(z_t|\underline{x}, \theta^n)}[\log p(x_t|z_t, \theta)] \\
&= \sum_{i=1}^M \log \pi_i \cdot p(z_1 = i|\underline{x}, \theta^n) + \\
&\quad + \sum_{t=2}^T \sum_{i=1}^M \sum_{j=1}^M \log A_{ij} \cdot p(z_{t-1} = i, z_t = j|\underline{x}, \theta^n) + \\
&\quad + \sum_{t=1}^T \sum_{i=1}^M \sum_{l=1}^L \log B_{il} \mathbb{I}(x_t = l) \cdot p(z_t = i|\underline{x}, \theta^n) \\
&= \sum_{i=1}^M \log \pi_i \gamma_1^n(i) + \sum_{t=2}^T \sum_{i=1}^M \sum_{j=1}^M \log A_{ij} \xi_{t-1,t}^n(i, j) + \\
&\quad + \sum_{t=1}^T \sum_{i=1}^M \sum_{l=1}^L \log B_{il} \mathbb{I}(x_t = l) \gamma_t^n(i).
\end{aligned}$$

The E step involves the computation of the expected sufficient statistics:

- $\gamma_t^n(i) = p(z_t = i|x_{1:T}, \theta^n)$ ,
- $\xi_{t-1,t}^n(i, j) = p(z_{t-1} = i, z_t = j|x_{1:T}, \theta^n)$ .

Conditioning on  $\theta^n$  means computing the  $\gamma$  and  $\xi$  values on the HMM with parameters  $\theta^n$ . As we have already seen, the  $\gamma$  and  $\xi$  values can be computed with dynamic programming algorithms.

The M step involves constrained optimization: we want to optimize in  $\pi$ ,  $A$ ,  $B$ , but we must ensure that:

- $\pi$  is a probability distribution on  $\{1, \dots, M\}$ ,
- $\forall i A_{i,:}$  is a probability distribution on  $\{1, \dots, M\}$ ,

- $\forall i$   $B_{i,:}$  is a probability distribution on  $\{1, \dots, L\}$ .

We can optimize separately in  $\pi$ ,  $A_{i,:}$  for  $i = 1, \dots, M$ , and  $B_{i,:}$  for  $i = 1, \dots, M$ .

**Statement 5.** (*M step optimization as divergence minimization*)

Let  $a_i \geq 0$  for  $i = 1, \dots, M$ . The probability distribution  $p$  on  $\{1, \dots, M\}$  that maximizes

$$\sum_{i=1}^M \log p_i \cdot a_i$$

is  $p_i = a_i/a$ , if  $a = \sum_{i=1}^M a_i > 0$ .

*Proof.* If  $a_i = 0 \forall i$ , then any  $p$  maximizes the term. Note that the following proof returns correctly that  $a_i = 0 \implies p_i = 0$ .

Define the probability distribution  $\hat{a}$  with  $\hat{a}_i = \frac{a_i}{a}$ . Then

$$\begin{aligned} \arg \max_p \sum_{i=1}^M \log p_i \cdot a_i &= \arg \max_p \sum_{i=1}^M \log p_i \cdot \hat{a}_i \\ &= \arg \max_p \sum_{i=1}^M \hat{a}_i \log p_i - \sum_{i=1}^M \hat{a}_i \log \hat{a}_i \\ &= \arg \max_p -D(\hat{a}||p). \end{aligned}$$

We have  $-D(\hat{a}||p) \leq 0$  and equality holds if and only if  $p = \hat{a}$ .  $\square$

For the  $\theta^{n+1}$  updated parameters:

$$\begin{aligned}
\pi^{n+1} &= \arg \max_{\pi} \sum_{i=1}^M \log \pi_i \gamma_1^n(k) = \gamma_1^n \\
A_{i,:}^{n+1} &= \arg \max_{A_{i,:}} \sum_{t=2}^T \sum_{j=1}^M \log A_{ij} \xi_{t-1,t}^n(i, j) \\
&= \arg \max_{A_{i,:}} \sum_{j=1}^M \log A_{ij} \left( \sum_{t=2}^T \xi_{t-1,t}^n(i, j) \right) \\
&\propto \left( \sum_{t=2}^T \xi_{t-1,t}^n(i, j) \right)_{j=1, \dots, M} \\
B_{i,:}^{n+1} &= \arg \max_{B_{i,:}} \sum_{t=1}^T \sum_{l=1}^L \log B_{il} \mathbb{I}(x_t = l) \gamma_t^n(i) \\
&= \arg \max_{B_{i,:}} \sum_{l=1}^L \log B_{il} \left( \sum_{t=1}^T \mathbb{I}(x_t = l) \gamma_t^n(i) \right) \\
&\propto \left( \sum_{t=1}^T \mathbb{I}(x_t = l) \gamma_t^n(i) \right)_{l=1, \dots, L}
\end{aligned}$$

The results are quite intuitive:

- $\pi_i^{n+1} = \gamma_1^n(i)$ ,
- $A_{ij}^{n+1} \propto \sum_{t=2}^T \xi_{t-1,t}^n(i, j)$ ,
- $B_{il}^{n+1} \propto \sum_{t=1}^T \gamma_t^n(i) \mathbb{I}(x_t = l)$ .

These are all expected counts on the corresponding events. The EM learning in the HMM framework is called the Baum-Welch algorithm.

**Statement 6.** (Zero persistency in EM - transition probability)

If we initialize the EM algorithm with such  $\theta^0$  that has  $A_{ij}^0 = 0$ , then:

$$\forall n : A_{ij}^n = 0.$$

*Proof.* It is enough to show that  $A_{ij}^1 = 0$ .

From the computation of  $\xi$ , we know that if  $A_{ij}^0 = 0$ , then  $\xi_{t-1,t}^0(i, j) = 0$  for  $t = 2, \dots, T$ .

But  $A_{ij}^1 \propto \sum_{t=2}^T \xi_{t-1,t}^0(i, j) = 0$ , which shows that  $A_{ij}^1 = 0$ .  $\square$

**Statement 7.** (*Zero persistency in EM - initial distribution*)

*If we initialize the EM algorithm with such  $\theta^0$  that has  $\pi_i^0 = 0$ , then:*

$$\forall n : \pi_i^n = 0.$$

*Proof.* It is enough to show that  $\pi_i^1 = 0$ . From the computation of  $\gamma$ , we know that if  $\pi_i^0 = 0$ , then  $\alpha_1^0(i) = 0$  and  $\gamma_1^0(i) = 0$ . But  $\pi_i^1 = \gamma_1^0(i) = 0$ .  $\square$

### 3.3 Complexity of the Baum-Welch algorithm

One iteration of the Baum-Welch algorithm involves an E-step and an M-step computation for the HMM. Now assume that  $E$  is the edge number of the  $\theta^0$  initialized HMM ( $E \geq M - 1$ ).

As we already know, the E-step is the computation of  $\gamma$  and  $\xi$  values and that takes  $\mathcal{O}(TM^2)$  time or  $\mathcal{O}(TE)$  time in a sparse graph.

On the time complexity of the M-step: for the update of  $\pi$  we need  $\mathcal{O}(M)$  time. For the update of  $A$ , we need to update at  $M^2$  or  $E$  places, and each takes  $\mathcal{O}(T)$  time.

For the matrix  $B$ , we have  $M \times L$  parameters, for each of them, it takes  $\mathcal{O}(T)$  time to update. But for each  $l \in \{1, \dots, L\}$  we only need to sum over  $T_l = \{t : x_t = l\}$ :  $B_{il} \propto \sum_{t \in T_l} \gamma_t(i)$ . Thus for each  $i \in \{1, \dots, M\}$ , the complexity is  $\sum_{l=1}^L |T_l| = T$ , since  $T_l$  is a partition of the  $\{1, \dots, T\}$  indices.

Hence the full time complexity of an M-step is  $\mathcal{O}(M + TM^2 + TM) = \mathcal{O}(TM^2)$  or  $\mathcal{O}(M + TE + TM) = \mathcal{O}(TE)$ .

Therefore one iteration of the Baum-Welch takes  $\mathcal{O}(TM^2)$  time or  $\mathcal{O}(TE)$  time. As we have already seen,  $E$  does not increase with the Baum-Welch algorithm. We conclude that the initial number of edges  $E$  strongly affects the time complexity of the Baum-Welch.

## 4 Graph representation of distributions

Let the notation  $p(v|u)$  for  $u, v$  (hidden) states denote the short form of the time-independent  $p(z_t = v | z_{t-1} = u)$ .

In general, one main setback of HMMs is that each hidden state  $i$  has a duration  $T_i \sim \text{Geo}(p_i)$ . The geometric distribution corresponds to the most simple graph (Figure 1): vertices are  $\{r, v_1, s\}$ , edges are  $\{(r, v_1), (v_1, v_1), (v_1, s)\}$  with  $p(v_1|r) = 1$ ,  $p(v_1|v_1) = 1 - p$  and also  $p(s|v_1) = p$ . The first arrival to the vertex  $s$  (starting from  $r$  at index 0) signs the transition to another state. One can extend the graph with  $p(s|s) = 1$  to ensure a stochastic transition matrix and therefore a Markov chain (but this does not alter the computation). Given this graph, the probability that the first arrival to  $s$  is at step  $d + 1$  is

$$P(\inf\{k : x_k = s\} = d + 1) = (1 - p)^{d-1}p = \text{Geo}(p)(d)$$

for the  $(x)_k$  Markov chain starting from  $x_0 = r$ . The duration  $d \geq 1$  refers to the same logic as in graphical models, if we step into a state, we must spend at least 1 time-unit there (in discrete time).

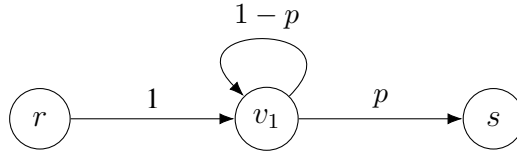


Figure 1: Representation graph of  $\text{Geo}(p)$  distribution.

The generalization of the previous idea (representing duration distributions with graphs) is possible.

### 4.1 Representation graphs

Formalizing the occurred concepts:

**Definition 3.** (*Duration distribution*)

Let  $X : \Omega \rightarrow \mathbb{N}_+$  be a random variable. Then  $T = p_X$ , the distribution of  $X$  is a duration distribution.

Duration distributions are for example geometric distribution, categorical distribution on  $\{1, \dots, D\}$ , and negative binomial distribution. A mixture of duration distributions is also a duration distribution. The Poisson distribution is not a duration distribution, but if we truncate it to  $[1, \infty)$  and normalize it (to integrate to 1), we get a duration distribution (we will call it Poisson duration distribution).

**Definition 4.** (*Parametric family of duration distributions*)

Let  $\Theta$  be a parameter space. If for every  $\theta \in \Theta$ :  $X(\theta) : \Omega \rightarrow \mathbb{N}_+$ , then  $\{T(\theta) : \theta \in \Theta\} = \{p_{X(\theta)} : \theta \in \Theta\}$  is a parametric family of duration distributions.

Parametric families of duration distributions are for example geometric distributions with parameter  $p$ , categorical distributions on  $\{1, \dots, D\}$  with parameters  $p_1, \dots, p_D$ , negative binomial distributions with parameters  $N, p$ , negative binomial distributions of fixed order  $N$  with parameter  $p$ , Poisson duration distributions with parameter  $\lambda$ .

One can think of learning a self-transition probability in the HMM framework as learning  $p$  from the family of geometric distributions. That is, similarly to the observation model, a family is given. So, if the duration comes from a geometric family, it is easy to solve. But what if we know that the duration comes from another family? Such as  $Cat(\{1, \dots, D\})$ ?

It will be shown that some duration distribution families can be represented as graphs, and in the next chapter, it would be introduced that one can "merge" these graphs to form a "two-layer" HMM with state durations from the desired family. There may be many possible representations for a given family, therefore we might measure the "efficiency" of those representations.

**Definition 5.** (*Representation graph*)

A  $G(\eta)$  Markov chain is a representation graph if we have  $r, s$  nodes that:

1.  $r$  is the starting node with probability 1,
2.  $G(\eta)$  stays in  $r$  only at index 0,
3.  $s$  is the ending node with probability 1,
4.  $p(s|r) = 0$ .

For a representation graph the following properties hold:

1.  $r, v_1, \dots, v_n, s$  are the nodes,
2.  $r$  is the starting node with probability 1,
3.  $G(\eta)$  stays in  $r$  only at index 0,
4.  $s$  is the ending node with probability 1 ( $P(\inf\{k : x_k = s\} < \infty) = 1$ ),
5.  $p(r|r) = 0, p(s|r) = 0, p(s|s) = 1$ ,
6.  $\forall i : p(r|v_i) = 0$ ,
7.  $\exists i : p(s|v_i) > 0$ ,
8.  $E(G) = E_{fix}(G) \dot{\cup} E_{prob}(G)$ , where the probabilities in  $E_{fix}$  are fixed 0s or 1s, and the probabilities in  $E_{prob}$  are fully controlled by  $\eta$ .

The indexing starts from 0 for a  $G(\eta)$  sample and the number of steps taken in  $G(\eta)$  (or the duration) for a sample is  $d$ , if the first arrival to  $s$  is at  $d + 1$ .

We denote the distribution of duration from  $G(\eta)$  generated samples with  $T[G(\eta)]$ .

If we denote two representation graphs with  $G(\eta_1)$  and  $G(\eta_2)$  it means that they have the same structure, only the probabilities on the non-fixed edges can differ.

Formally, if  $X_0, X_1, \dots$  is the Markov chain  $G(\eta)$  with  $X_0 = r$ , then:

$$\begin{aligned}
T[G(\eta)](d) &= P(\inf\{k : X_k = s\} = d + 1) \\
&= P(X_{d+1} = s, X_d \neq s) \\
&= P(X_{d+1} = s \text{ first time}) \\
&= P(X_{d+1} = s \text{ ft}) = P_{G(\eta)}(X_{d+1} = s \text{ ft}).
\end{aligned}$$

The first example of the geometric distribution is a  $G(p)$  representation graph with  $E_{fix} = \{(r, v_1)\}$  and  $E_{prob} = \{(v_1, v_1), (v_1, s)\}$ . As we already observed,  $T[G(p)] = Geo(p)$ .

**Definition 6.** (*Properties of a representation graph*)

Let  $G(\eta)$  be a representation graph. Then:

- $e_{in} \doteq |\{i : p(v_i|r) \neq 0\}|$  the number of incoming edges,
- $e_{out} \doteq |\{i : p(s|v_i) \neq 0\}|$  the number of outgoing edges,
- $e \doteq |\{i, j : p(v_j|v_i) \neq 0\}|$  the number of inner edges,
- $n \doteq |V(G)| - 2$  the number of nodes,
- $V_{inn} \doteq \{v_1, \dots, v_n\}$  the set of inner nodes.

An edge  $(u, v)$  is  $p(v|u) \neq 0$  in this definition, if  $(u, v) \in E_{fix}(G)$  with probability 1 or if  $(u, v) \in E_{prob}(G)$ .

The geometric distribution representation graph  $G(p)$  has the following edge numbers:  $e_{in} = 1$ ,  $e_{out} = 1$ ,  $e = 1$ . The number of nodes is  $n = 1$ .

**Definition 7.** (*Graph representation of duration distribution*)

Let  $T$  be a duration distribution. Let  $G(\eta)$  be a representation graph.  $G(\eta)$  represents  $T$  if  $T = T[G(\eta)]$ .

**Definition 8.** (*Graph representation of duration distribution families*)

Let  $T(\theta)$  be a parametric family of duration distributions. Let  $\{G(\eta) : \eta \in H\}$  be a family of representation graphs based on the same structure.

$G$  represents  $T(\theta)$  (the family) if there is a bijection between the parameters

$$\begin{aligned}\forall \theta \exists \eta T(\theta) &= T[G(\eta)], \\ \forall \eta \exists \theta T[G(\eta)] &= T(\theta).\end{aligned}$$

In our cases, the same  $\theta$  can be used for parameterization, leading to  $\forall \theta T(\theta) = T[G(\theta)]$ .

For example, the family of geometric distributions with parameter  $p$  can be represented with the same graph structure as at the beginning of the chapter with the  $\eta = p$  parameter.

The main question is how other distribution families can be represented with graphs.

For example consider the representation graph  $G(p)$  (Figure 2) with nodes  $r, v_1, v_2, v_3, s$  and with the following non-zero probabilities:

- $p(v_1|r) = 1$ ,
- $p(v_1|v_1) = p(v_2|v_2) = p(v_3|v_3) = 1 - p$ ,
- $p(v_2|v_1) = p(v_3|v_2) = p(s|v_3) = p$ .

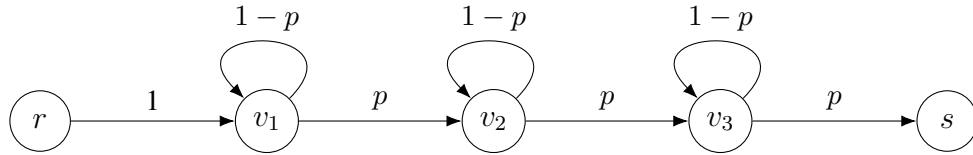


Figure 2: Representation graph of  $NB_3(p)$  negative binomial distribution.

It is not hard to see that  $G$  represents the family of negative binomial distributions of fixed order 3. [3]

**Statement 8.** (Walk-based description)

Let  $X_0, X_1, \dots$  be the Markov chain of the  $G(\eta)$  representation graph. Let  $W_{d+1} = \{x_0, x_1, \dots, x_{d+1} : x_0 = r, x_{d+1} = s, x_i \neq s \forall i \leq d\}$  denote the set of  $r \rightarrow s$  walks with length  $d + 1$  (and without  $s$  as an inner point). Then:

$$P(X_{d+1} = s \text{ ft}) = \sum_{w \in W_{d+1}} \prod_{e \in w} p(e).$$

*Proof.* The form of the Markov chain indicates that  $\{X_d \neq s\} = \{X_i \neq s \forall i \leq d\}$ .

$$\begin{aligned} P(X_{d+1} = s \text{ ft}) &= \sum_{\substack{x_0, \dots, x_{d+1} \\ x_0=r, x_{d+1}=s \\ x_d \neq s}} P(X_0 = x_0, \dots, X_{d+1} = x_{d+1}) \\ &= \sum_{\substack{x_0, \dots, x_{d+1} \\ x_0=r, x_{d+1}=s \\ x_d \neq s}} \prod_{j=1}^{d+1} p(x_j | x_{j-1}) \\ &= \sum_{w \in W_{d+1}} \prod_{e \in w} p(e) \end{aligned}$$

□

## 4.2 Representation of distribution families

The following duration distribution families have a graph representation: geometric family with parameter  $p$ , negative binomial distributions of fixed order  $N$  with parameter  $p$ , categorical distributions on  $\{1, \dots, D\}$  with parameters  $p_1, \dots, p_D$ .

**Statement 9.** (*Representation of geometric family*)

*The  $\text{Geo}(p)$  geometric family can be represented by a  $G(p)$  graph (Figure 1) with nodes  $r, v_1, s$  and with the following non-zero probabilities:*

- $p(v_1 | r) = 1$ ,
- $p(v_1 | v_1) = 1 - p$ ,
- $p(s | v_1) = p$ .

*Proof.* We know that  $\text{Geo}(p)(d) = (1 - p)^{d-1}p$  for  $d \geq 1$ . Using the definition of  $T[G(p)]$ :

$$\begin{aligned}
T[G(p)](d) &= P(\inf\{k : x_k = s\} = d + 1) \\
&= P(X_0 = r, X_1 = v_1, \dots, X_d = v_1, X_{d+1} = s) \\
&= P(X_0 = r) \cdot P(X_1 = v_1 | X_0 = r) \cdot \prod_{i=2}^d P(X_i = v_1 | X_{i-1} = v_1) \cdot \\
&\quad \cdot P(X_{d+1} = s | X_d = v_1) \\
&= 1 \cdot p(v_1 | r) \cdot \prod_{i=2}^d p(v_1 | v_1) \cdot p(s | v_1) \\
&= 1 \cdot 1 \cdot (1 - p)^{d-1} \cdot p = (1 - p)^{d-1} p.
\end{aligned}$$

There is a clear bijection between  $Geo(p)$  instances and  $G(p)$  instances; using the same  $p$ .  $\square$

**Statement 10.** (*Representation of negative binomial family of fixed order  $N$* )

*The  $NB_N(p)$  negative binomial family can be represented by a  $G(p)$  graph with nodes  $r, v_1, \dots, v_N, s$  and with the following non-zero probabilities:*

- $p(v_1 | r) = 1$ ,
- $p(v_i | v_i) = 1 - p$  for  $i = 1, \dots, N$ ,
- $p(v_i | v_{i-1}) = p$  for  $i = 2, \dots, N$ ,
- $p(s | v_N) = p$ .

*Proof.* (The representation graph of  $NB_N(p)$  is the same graph as in Figure 2 with  $N$  inner nodes.)

We know that  $NB_N(p)(d) = \binom{d-1}{N-1} (1-p)^{d-N} p^N$  for  $d \geq N$ .

We prove the statement by induction.

For  $N = 1$ , this is the geometric distribution and the previous statement.

For  $N > 1$ , assume we know the statement for  $N - 1$ . By separating on the first arrival to  $v_N$ , and using the induction step:

$$\begin{aligned}
T[G(p)](d) &= P(X_{d+1} = s \text{ ft}) \\
&= \sum_{i=N}^d P(X_{d+1} = s \text{ ft}, X_i = v_N \text{ ft}) \\
&= \sum_{i=N}^d P(X_i = v_N \text{ ft}) P(X_{d+1} = s \text{ ft} \mid X_i = v_N \text{ ft}) \\
&= \sum_{i=N}^d NB_{N-1}(p)(i-1)Geo(p)(d-i+1) \\
&= \sum_{i=N}^d \binom{i-2}{N-2} (1-p)^{i-N} p^{N-1} (1-p)^{d-i} p \\
&= (1-p)^{d-N} p^N \sum_{i=N}^d \binom{i-2}{N-2} \\
&= (1-p)^{d-N} p^N \left[ \binom{N-2}{N-2} + \sum_{i=N+1}^d \left( \binom{i-1}{N-1} - \binom{i-2}{N-1} \right) \right] \\
&= (1-p)^{d-N} p^N \binom{d-1}{N-1}.
\end{aligned}$$

There is a clear bijection between  $NB_N(p)$  instances and  $G(p)$  instances; using the same  $p$ .

□

**Statement 11.** (*Representation of categorical distributions on  $\{1, \dots, D\}$* )  
*The  $Cat(\{1, \dots, D\})$  categorical family with parameters  $p_1, \dots, p_D$  can be represented by a  $G(p_1, \dots, p_D)$  graph (Figure 3) with nodes  $r, v_1, \dots, v_D, s$  and with the following non-zero probabilities:*

- $p(v_1|r) = 1$ ,
- $p(v_d|v_1) = p_{D+2-d}$  for  $d = 2, \dots, D$ ,
- $p(v_d|v_{d-1}) = 1$  for  $d = 3, \dots, D$ ,
- $p(s|v_D) = 1$ ,

- $p(s|v_1) = p_1$ .

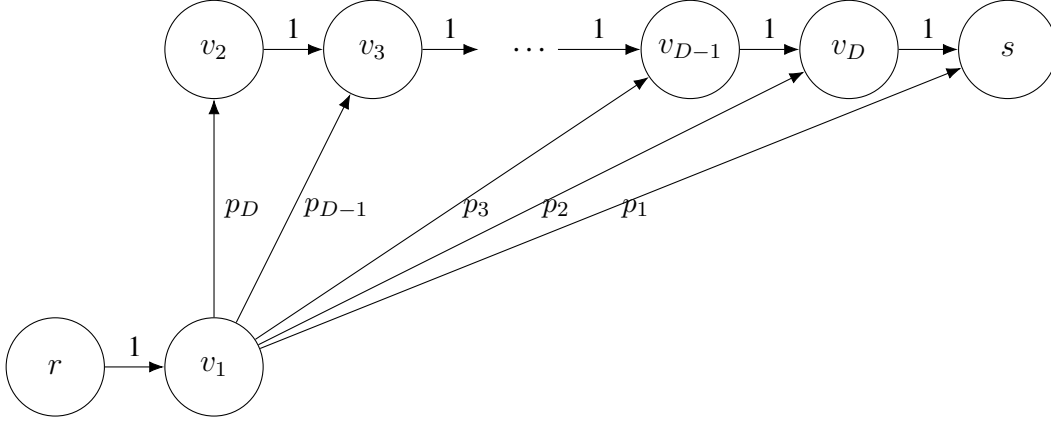


Figure 3: Representation graph of  $Cat(\{1, \dots, D\})$  distribution.

*Proof.* We know that the  $Cat(\{1, \dots, D\})$  distribution has  $p_1, \dots, p_D$  parameters with  $\sum_{d=1}^D p_d = 1$ ,  $p_d \geq 0$  and  $Cat(\{1, \dots, D\})(d) = p_d$ .

We will use the walk-based description:

$$P(X_{d+1} = s \text{ ft}) = \sum_{w \in W_{d+1}} \prod_{e \in w} p(e).$$

For  $d = 1$ :  $W_2 = \{(r, v_1, s)\}$ , therefore  $P(X_2 = s \text{ ft}) = p(v_1|r)p(s|v_1) = p_1$ .

For  $2 \leq d \leq D$ :  $W_{d+1} = \{(r, v_1, v_{D+2-d}, \dots, v_D, s)\}$ , therefore

$$\begin{aligned} P(X_{d+1} = s \text{ ft}) &= p(v_1|r) \cdot p(v_{D+2-d}|v_1) \cdot \prod_{j=D+2-d}^{D-1} p(v_{j+1}|v_j) \cdot p(s|v_D) \\ &= p_{D+2-(D+2-d)} = p_d. \end{aligned}$$

For  $d > D$ :  $W_{d+1} = \emptyset$ , therefore  $P(X_{d+1} = s \text{ ft}) = 0$ . □

In the next chapter, we will see two more graph representations for  $Cat\{1, \dots, D\}$ .

It is not hard to see that the mixture distributions can be represented if all the individuals can be represented.

**Statement 12.** (*Representation of mixture distributions*)

Let the family  $\{T_i(\theta_i) : \theta_i \in \Theta_i\}$  be represented by a graph  $G_i(\eta_i)$  for  $i = 1, 2$ . Then the family  $\{\rho T_1(\theta_1) + (1 - \rho)T_2(\theta_2) : \rho \in [0, 1], \theta_1 \in \Theta_1, \theta_2 \in \Theta_2\}$  can be represented by a graph  $G(\rho, \eta_1, \eta_2)$  with nodes  $r, V_{inn}(G_1), V_{inn}(G_2), s$  and with the following non-zero probabilities:

- $p(v_i^1|r) = \rho \cdot p_{G_1(\theta_1)}(v_i^1|r)$  for  $v_i^1 \in V_{inn}(G_1)$ ,
- $p(v_i^2|r) = (1 - \rho) \cdot p_{G_2(\theta_2)}(v_i^2|r)$  for  $v_i^2 \in V_{inn}(G_2)$ ,
- $p(v_j^1|v_i^1), p(s|v_i^1)$  as in  $G_1(\theta_1)$ ,
- $p(v_j^2|v_i^2), p(s|v_i^2)$  as in  $G_2(\theta_2)$ .

*Proof.* It is enough to prove that

$$T[G(\rho, \eta_1, \eta_2)] = \rho T[G_1(\eta_1)] + (1 - \rho)T[G_2(\eta_2)].$$

We denote  $P_{G(\rho, \eta_1, \eta_2)}$  with  $P$  for brevity.

$$\begin{aligned} T[G(\rho, \eta_1, \eta_2)] &= P(X_{d+1} = s \text{ ft}) \\ &= P(X_{d+1} = s \text{ ft} | X_1 \in V_{inn}(G_1))P(X_1 \in V_{inn}(G_1)) + \\ &\quad + P(X_{d+1} = s \text{ ft} | X_1 \in V_{inn}(G_2))P(X_1 \in V_{inn}(G_2)) \\ &= \rho P_{G_1(\eta_1)}(X_{d+1} = s \text{ ft}) + (1 - \rho)P_{G_2(\eta_1)}(X_{d+1} = s \text{ ft}) \\ &= \rho T[G_1(\eta_1)] + (1 - \rho)T[G_2(\eta_2)] \end{aligned}$$

□

However, not every distribution family and not every distribution can be represented.

**Statement 13.** (*Non-representation of light-tailed distributions*)

Let  $T$  be a duration distribution with infinite support and with the following property:

$$\limsup_{d \rightarrow \infty} \frac{T(d)}{\alpha^d} = 0 \quad \forall \alpha > 0.$$

Then there is no finite graph that can represent the distribution  $T$ .

*Proof.* Assume that  $G(\eta)$  represents  $T$ .

If  $G(\eta)$  has no positive circle, then it can only represent a finite-support distribution. (Because in this case, the nodes form a DAG, therefore a topological order exists, and the maximum length of an  $rs$  walk is  $n(G(\eta)) + 1$ .)

Let  $d_0$  be large enough ( $d_0 > n(G(\eta)) + 1$ ), and consider the walk-based description:

$$T(d_0) = P(X_{d_0+1} = s \text{ ft}) = \sum_{w \in W_{d_0+1}} \prod_{e \in w} p(e).$$

Select a  $w \in W_{d_0+1}$  positive walk; there must be at least one circle in this walk (otherwise it would not have length  $d_0$ ). Select a circle  $C$  from the walk. We denote the walk before  $C$  with  $w_0$  and the walk after  $C$  with  $w_1$ .

So  $w = w_0 C w_1$ , and let  $c = |C|$  be the length of  $C$  (i.e. the number of edges). Use the notation  $p_w = \prod_{e \in w} p(e)$  for any walk  $w$ , then we have:

$$\begin{aligned} T(d_0) &\geq \prod_{e \in w} p(e) \\ &= \prod_{e \in w_0} p(e) \prod_{e \in C} p(e) \prod_{e \in w_1} p(e) \\ &= p_{w_0} p_C p_{w_1} > 0. \end{aligned}$$

Define the following series:

$$d_j = d_0 + cj, \quad j = 0, 1, \dots$$

Then for  $j \geq 0$ ; the walk  $w^j = w_0 C^{j+1} w_1$  is a positive,  $d_j$ -length walk, so:

$$T(d_j) \geq p_{w_0} p_C^{j+1} p_{w_1} > 0.$$

Let  $\alpha < p_C^{1/c}$ , then:

$$\begin{aligned}
\limsup_{d \rightarrow \infty} \frac{T(d)}{\alpha^d} &\geq \limsup_{j \rightarrow \infty} \frac{T(d_j)}{\alpha^{d_j}} \\
&= \limsup_{j \rightarrow \infty} \frac{T(d_j)}{\alpha^{d_0 + cj}} \\
&\geq \lim_{j \rightarrow \infty} \frac{p_{w_0} p_C^{j+1} p_{w_1}}{\alpha^{d_0} \alpha^{cj}} \\
&= \frac{p_{w_0} p_C p_{w_1}}{\alpha^{d_0}} \lim_{j \rightarrow \infty} \left( \frac{p_C}{\alpha^c} \right)^j \\
&= \infty.
\end{aligned}$$

So the light-tailed property is violated, therefore no such  $G(\eta)$  representation graph exists.  $\square$

**Statement 14.** (*Non-representation of Poisson duration distributions*)

*Let  $T$  be one member of the Poisson duration distribution family.*

*Then there is no finite graph that can represent the distribution  $T$ .*

*Proof.* We have  $T(d) = C \frac{\lambda^d}{d!}$ , so  $T$  has infinite-support and  $T$  is light-tailed, therefore the previous statement applies.  $\square$

## 5 Graphical-Duration Hidden Markov Model

### 5.1 Hidden Semi-Markov Models

One can construct HMM-like models that are aware of time, different options can be found in the review of Yu [5]. The variants are usually called Hidden Semi-Markov Model, Variable Duration Hidden Markov Model, or Explicit Duration Hidden Markov Model.

Each solution in the review of Yu introduces new graphical models with "counter states", and does not try to capture duration times inside the HMM framework.

One solution from the review of Yu is the residential time HMM which assumes that a state transition is either  $(i, 1) \rightarrow (j, \tau)$  for  $j \neq i$  or  $(i, \tau) \rightarrow (i, \tau - 1)$  where  $\tau$  is the residential time of state  $i$ . [5, 6]

Yu and Kobayashi provided the forward-backward algorithm for the model, a modification of the HMM forward-backward algorithm. The algorithm takes  $\mathcal{O}((M^2 + MD)T)$  steps, where  $T$  is the length of the observation sequence,  $M$  is the number of hidden states,  $D$  is the maximum residential time (or maximum duration, the maximum steps allowed to be in one state without transition).

Using the idea of representation graphs, a new aspect of the previous result can be given, with a similar model and with a similar learning algorithm which has the same computational complexity as in Yu & Kobayashi [6].

Firstly a new, general definition of the Graphical-Duration Hidden Markov Model (GD-HMM) should be established with the usage of representation graphs.

### 5.2 The GD-HMM

The GD-HMM is a simple HMM with parameter tyings and reparameterization. The model builds up from a simple HMM structure and replaces the initial states with the desired graphs, which represent the duration families.

Consider the  $(\pi, A, \theta_o)$  HMM model with  $M$  hidden states, where  $\pi$  is the initial distribution,  $A$  is the transition matrix and  $\theta_o$  is the observation parameter matrix. For simplicity, we assume that  $A_{ii} = 0$  for all  $i$ .

Let  $\{T_i(\theta_i)\}$  be a parametric family of duration distributions, represented with the family  $\{G_i(\eta_i)\}$ .

For the graph  $G_i$ , we use the following notations:

- $D_i = n(G_i)$  the number of (inner) nodes,
- $r_i = r(G_i)$  starting node,
- $s_i = s(G_i)$  ending node,
- $\{i_1, \dots, i_{D_i}\} = V_{inn}(G_i)$ ,
- $e_{in}^i = e_{in}(G_i)$  the number of incoming edges,
- $e_{out}^i = e_{out}(G_i)$  the number of outgoing edges,
- $e^i = e(G_i)$  the number of inner edges.

The  $G_i(\eta_i)$  graph is still a Markov chain on nodes  $r_i, i_1, \dots, i_{D_i}, s_i$  with transition probabilities  $p_{G_i(\eta_i)}(v|u)$  for  $u, v$  (hidden) states.

**Definition 9. (GD-HMM)**

Let  $(\pi, A, \theta_o)$  be an HMM model with  $M$  hidden states, and let  $T_i$  be a duration distribution, represented with  $G_i(\eta_i) \forall i = 1, \dots, M$ .

The GD-HMM is a  $(\tilde{\pi}, \tilde{A}, \tilde{\theta}_o)$  HMM model.

For  $i = 1, \dots, M$ :

- *hidden states*:  $i_d \in V_{inn}(G_i)$  for  $d = 1, \dots, D_i$ ,
- *transition probabilities*
  - $\tilde{A}(i_k, i_l) \doteq p_{G_i(\eta_i)}(i_l|i_k)$  for  $k, l = 1, \dots, D_i$ ,
  - $\tilde{A}(i_k, j_l) \doteq p_{G_i(\eta_i)}(s_i|i_k)A_{ij}p_{G_j(\eta_j)}(j_l|r_j)$  for  $k = 1, \dots, D_i$  for  $l = 1, \dots, D_j$  for  $j \neq i$ ,
- *initial distribution*  $\tilde{\pi}(i_1) = \pi(i)$ ,  $\tilde{\pi}(i_k) = 0$  for  $k = 2, \dots, D_i$ ,
- *observation model parameters*  $\tilde{\theta}_o(i_k) = \theta_o(i)$  for  $k = 1, \dots, D_i$ .

The parameters of the GD-HMM are  $(\pi, A, \theta_o, (\eta_1, \dots, \eta_M))$ .

Example (Figure 4): hidden states and transition probabilities of a GD-HMM with 3 initial states ( $i$ ,  $j$  and  $k$ ) and categorical distributions with maximum durations of  $D_i = 3$ ,  $D_j = 3$  and  $D_k = 2$ . We use the representation graph of the categorical distribution from Statement 11. The probabilities from the  $i$  states are marked on the solid edges, while the other probabilities on the dashed edges are not shown.

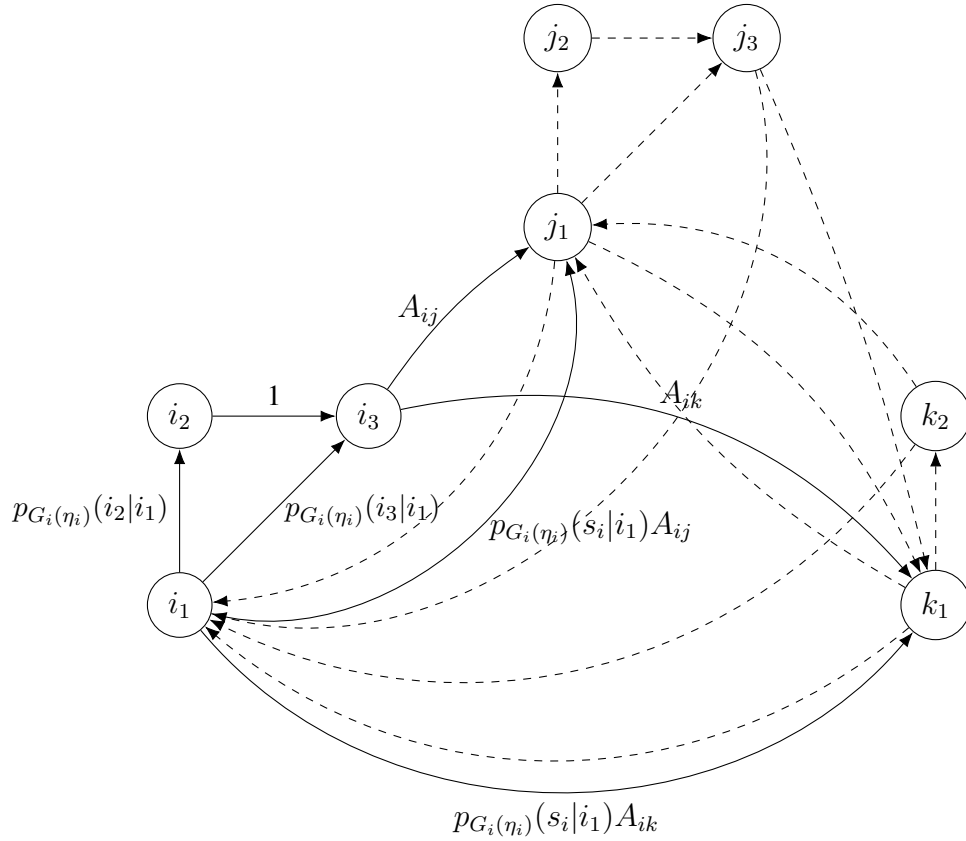


Figure 4: Hidden states and transition probabilities of a GD-HMM.

If we want to build a GD-HMM from an HMM with  $A_{ii} > 0$ , then in the computation of  $\tilde{A}(i_k, j_l)$ , we should work with  $\frac{A_{ij}}{1-A_{ii}}$  instead of  $A_{ij}$ .

**Statement 15.** ( $\tilde{A}$  is a transition matrix)

$$\sum_v \tilde{A}(i_k, v) = 1$$

*Proof.*

$$\begin{aligned} \sum_v \tilde{A}(i_k, v) &= \sum_{l=1}^{D_i} \tilde{A}(i_k, i_l) + \sum_{\substack{j=1 \\ j \neq i}}^M \sum_{l=1}^{D_j} \tilde{A}(i_k, j_l) \\ &= \sum_{l=1}^{D_i} p_{G_i(\eta_i)}(i_l | i_k) + \sum_{\substack{j=1 \\ j \neq i}}^M \sum_{l=1}^{D_j} p_{G_i(\eta_i)}(s_i | i_k) \frac{A_{ij}}{1 - A_{ii}} p_{G_j(\eta_j)}(j_l | r_j) \\ &= 1 - p_{G_i(\eta_i)}(s_i | i_k) + p_{G_i(\eta_i)}(s_i | i_k) \sum_{\substack{j=1 \\ j \neq i}}^M \frac{A_{ij}}{1 - A_{ii}} \sum_{l=1}^{D_j} p_{G_j(\eta_j)}(j_l | r_j) \\ &= 1 \end{aligned}$$

□

The GD-HMM has two layers of representation: a lower-level representation with  $i_d$  states, which form a Markov chain, and a higher-level representation with  $i \leftrightarrow \{i_1, \dots, i_{D_i}\}$  super states, which correspond to the original hidden states.

The number of (non-zero) edges in a GD-HMM is:

$$E = \sum_{i=1}^M e^i + \sum_{i=1}^M \sum_{\substack{j=1 \\ j \neq i}}^M e_{out}^i e_{in}^j \mathbb{I}(A_{ij} > 0).$$

The number of (non-zero) edges in a dense GD-HMM (when the original HMM is complete) is:

$$E = \sum_{i=1}^M e^i + \sum_{i=1}^M \sum_{\substack{j=1 \\ j \neq i}}^M e_{out}^i e_{in}^j.$$

The number of nodes is  $V = \sum_{i=1}^M D_i$ . The number of parameters in a GD-HMM can be upper-bounded by  $V$  (initial distribution) +  $E$  (real transitions) +  $VL$  (observation parameters).

If we assume that all  $D_i = D$  are equal, and  $e^i = \mathcal{O}(D)$ ,  $e_{in}^i = \mathcal{O}(1)$  and  $e_{out}^i = \mathcal{O}(1)$ , then the number of nodes is  $MD$  and the number of edges is  $\mathcal{O}(MD + M^2)$ , which results in a sparse graph if  $D \gg M$ .

**Example 1.** (*HMM is a subclass of the GD-HMM*)

Let  $(\pi, A, \theta_o)$  be an HMM with  $M$  hidden states. Let  $T_i = \text{Geo}(1 - p_i)$ , and  $\forall i = 1, \dots, M$  consider the representation graph  $G_i(p_i)$  with nodes  $r_i, i_1, s_i$  and the following non-zero probabilities:

- $p(i_1|r_i) = 1$ ,
- $p(i_1|i_1) = p_i$ ,
- $p(s_i|i_1) = 1 - p_i$ .

The resulting GD-HMM is a  $(\pi, \tilde{A}, \theta_o)$  HMM model on the  $\{1, \dots, M\}$  nodes with:

$$\tilde{A}_{ij} = \begin{cases} (1 - p_i)A_{ij}/(1 - A_{ii}) & \text{if } j \neq i \\ p_i & \text{if } j = i \end{cases}$$

This gives back the original HMM if  $p_i = A_{ii} \forall i$ .

## 6 Learning the parameters of the GD-HMM

In the previous section, a new HMM variant was presented, but because of its special properties, we must go through the Baum-Welch algorithm to see what steps need to be modified. As the general GD-HMM model has  $\eta_1, \dots, \eta_M$  parameters, we have to specify the graph structures and the trainable parameters before we examine a learning algorithm for the parameters.

We consider the representation for categorical distributions as in 11. Each graph can have a different  $D_i$  maximum duration. The following reasoning is designed for the categorical emission distributions, but it can be easily extended to other observation distributions.

Assume that the initialization is correct, i.e. we construct the  $\theta^0$  GD-HMM from a  $(\pi^0, A^0, B^0)$  HMM with  $A_{ii}^0 = 0$  and from the  $G_i(\eta_i^0)$  categorical representation graphs as in the definition. For the categorical family with maximum duration  $D$  we have  $(p_1, \dots, p_D)$  parameters between the appropriate states (as in the statement), so let  $\eta_i^0 = (p_i^0(i_2|i_1), \dots, p_i^0(i_{D_i}|i_1), p_i^0(s_i|i_1))$  with the usage of the simple  $p_i(i_l|i_k)$  notation, which refers to the appropriate  $p_{i,d}$  parameter.

The initialized GD-HMM has  $E = \sum_{i=1}^M e^i + \sum_{i=1}^M \sum_{\substack{j=1 \\ j \neq i}}^M e_{out}^i e_{in}^j \mathbb{I}(A_{ij}^0 > 0)$  edges. We have already seen that  $E$  does not increase during the EM.

As the model is still an HMM, the E-step (forward-backward algorithm) including every related computation can be done as before:  $\alpha, \beta, \gamma, \xi$ . The time complexity is  $\mathcal{O}(TE)$  as we have already seen. The Viterbi decoding also can be done as before.

However, the M-step must be changed, because of the parameter tyings for  $\tilde{B}$  and parameter relation for  $\tilde{A}$ . Here, the reformulation of the M-step is presented, firstly to general  $(\eta_1, \dots, \eta_M)$  parameters, then to the categorical case.

The auxiliary function  $Q(\theta; \theta^n)$  for a simple HMM on  $\{1, \dots, M\}$  nodes has

the following form:

$$\begin{aligned}
Q(\theta; \theta^n) &= \sum_{i=1}^M \log \pi_i \gamma_1^n(i) + \sum_{t=2}^T \sum_{i=1}^M \sum_{j=1}^M \log A_{ij} \xi_{t-1,t}^n(i, j) + \\
&+ \sum_{t=1}^T \sum_{i=1}^M \sum_{l=1}^L \log B_{il} \mathbb{I}(x_t = l) \gamma_t^n(i).
\end{aligned}$$

The GD-HMM has nodes  $\{i_k : k \in \{1, \dots, D_i\}, i \in \{1, \dots, M\}\}$ :

$$\begin{aligned}
Q(\theta; \theta^n) &= \sum_{i=1}^M \sum_{k=1}^{D_i} \log \tilde{\pi}_{i_k} \gamma_1^n(i_k) + \sum_{t=2}^T \sum_{i=1}^M \sum_{k=1}^{D_i} \sum_{j=1}^M \sum_{l=1}^{D_j} \log \tilde{A}_{i_k, j_l} \xi_{t-1,t}^n(i_k, j_l) + \\
&+ \sum_{t=1}^T \sum_{i=1}^M \sum_{k=1}^{D_i} \sum_{l=1}^L \log \tilde{B}_{i_k, l} \mathbb{I}(x_t = l) \gamma_t^n(i_k).
\end{aligned}$$

We need to rewrite the auxiliary function to a function of  $(\pi, A, B, (\eta_1, \dots, \eta_M))$ . Using the short notations  $p_i = p_{G_i(\eta_i)}$ ,  $\xi(i_k, j_l) = \sum_{t=2}^T \xi_{t-1,t}^n(i_k, j_l)$ ,  $T_l = \{t : x_t = l\}$ , we rewrite the function term by term.

Initial distribution:

$$\sum_{i=1}^M \sum_{k=1}^{D_i} \log \tilde{\pi}_{i_k} \gamma_1^n(i_k) = \sum_{i=1}^M \log \pi_i \gamma_1^n(i_1)$$

Transition probabilities:

$$\begin{aligned}
& \sum_{t=2}^T \sum_{i=1}^M \sum_{k=1}^{D_i} \sum_{j=1}^M \sum_{l=1}^{D_j} \log \tilde{A}_{i_k, j_l} \xi_{t-1, t}^n(i_k, j_l) = \\
& \sum_{i=1}^M \sum_{k=1}^{D_i} \sum_{l=1}^{D_i} \log p_i(i_l | i_k) \xi(i_k, i_l) + \\
& \sum_{i=1}^M \sum_{k=1}^{D_i} \sum_{\substack{j=1 \\ j \neq i}}^M \sum_{l=1}^{D_j} \log(p_i(s_i | i_k) A_{ij} p_j(j_l | r_j)) \xi(i_k, j_l) = \\
& \sum_{i=1}^M \sum_{k=1}^{D_i} \left[ \sum_{l=1}^{D_i} \log p_i(i_l | i_k) \xi(i_k, i_l) + \log p_i(s_i | i_k) \left( \sum_{\substack{j=1 \\ j \neq i}}^M \sum_{l=1}^{D_j} \xi(i_k, j_l) \right) \right] + \quad (1)
\end{aligned}$$

$$\sum_{i=1}^M \left[ \sum_{\substack{j=1 \\ j \neq i}}^M \log A_{ij} \left( \sum_{k=1}^{D_i} \sum_{l=1}^{D_j} \xi(i_k, j_l) \right) \right] + \quad (2)$$

$$\sum_{j=1}^M \left[ \sum_{l=1}^{D_j} \log p_j(j_l | r_j) \left( \sum_{\substack{i=1 \\ i \neq j}}^M \sum_{k=1}^{D_i} \xi(i_k, j_l) \right) \right] \quad (3)$$

Emission probabilities:

$$\begin{aligned}
& \sum_{t=1}^T \sum_{i=1}^M \sum_{k=1}^{D_i} \sum_{l=1}^L \log \tilde{B}_{i_k, l} \mathbb{I}(x_t = l) \gamma_t^n(i_k) = \\
& \sum_{i=1}^M \left[ \sum_{l=1}^L \log B_{il} \left( \sum_{t=1}^T \sum_{k=1}^{D_i} \mathbb{I}(x_t = l) \gamma_t^n(i_k) \right) \right] = \\
& \sum_{i=1}^M \left[ \sum_{l=1}^L \log B_{il} \left( \sum_{t \in T_l} \sum_{k=1}^{D_i} \gamma_t^n(i_k) \right) \right] = \\
& \sum_{i=1}^M \left[ \sum_{l=1}^L \log B_{il} \left( \sum_{k=1}^{D_i} \sum_{t=1}^T \mathbb{I}(x_t = l) \gamma_t^n(i_k) \right) \right]
\end{aligned}$$

We see that an analytical update is possible in the M-step because the  $Q$

function can be written as a sum of separately parameterized  $\sum_{i \in I} a_i \log p_i$  terms, where  $(p_i : i \in I)$  is a probability distribution and  $a_i \geq 0 \forall i \in I$  are known and fixed.

The update of  $\pi$  initial distribution:  $\pi_i^{n+1} = \gamma_1^n(i_1)$ .

The update of  $B$  observation parameters is also simple. Use the following notation for the statistics:

$$\underline{b(i)} = \left( \sum_{t=1}^T \mathbb{I}(x_t = l) \gamma_t^n(i) \right)_{l=1, \dots, L}$$

Then:

$$\begin{aligned} B_{i,:}^{n+1} &= \arg \max_{B_{i,:}} \sum_{l=1}^L \log B_{il} \left( \sum_{k=1}^{D_i} \sum_{t=1}^T \mathbb{I}(x_t = l) \gamma_t^n(i_k) \right) \\ &\propto \left( \sum_{k=1}^{D_i} \sum_{t=1}^T \mathbb{I}(x_t = l) \gamma_t^n(i_k) \right)_{l=1, \dots, L} \\ &\propto \sum_{k=1}^{D_i} \underline{b(i_k)}, \end{aligned}$$

which is simply the sum of the statistics from the corresponding lower-level states.

## 6.1 M-step for transition parameters in categorical case

Finally, the update of  $A$  and  $(\eta_1, \dots, \eta_M)$  is not always simple, it depends on the representation graphs. For the categorical case, we need to examine each of the (1), (2), and (3) terms from the transition probability part of the  $Q(\theta; \theta^n)$  function.

The possible transitions in the categorical GD-HMM:

$$\begin{aligned} \forall i \neq j : i_1 &\rightarrow j_1, i_{D_i} \rightarrow j_1 \\ \forall i : i_1 &\rightarrow i_2, \dots, i_1 \rightarrow i_{D_i} \\ \forall i : i_2 &\rightarrow i_3, \dots, i_{D_i-1} \rightarrow i_{D_i} \quad (\text{with fix probability of 1}) \end{aligned}$$

So every other type of  $i_k \rightarrow j_l$  is not possible because of the initialization and therefore  $\xi(i_k, j_l) = 0$  for them. Thus we can rewrite the terms eliminating the  $\xi(i_k, j_l) = 0$  terms wherever possible.

Term (1):

$$\begin{aligned}
& \sum_{i=1}^M \sum_{k=1}^{D_i} \left[ \sum_{l=1}^{D_i} \log p_i(i_l|i_k) \xi(i_k, i_l) + \log p_i(s_i|i_k) \left( \sum_{\substack{j=1 \\ j \neq i}}^M \sum_{l=1}^{D_j} \xi(i_k, j_l) \right) \right] = \\
& \sum_{i=1}^M \sum_{k=1}^{D_i} \left[ \sum_{l=1}^{D_i} \log p_i(i_l|i_k) \xi(i_k, i_l) + \log p_i(s_i|i_k) \left( \sum_{\substack{j=1 \\ j \neq i}}^M \xi(i_k, j_1) \right) \right] = \\
& \sum_{i=1}^M \left[ \sum_{l=2}^{D_i} \log p_i(i_l|i_1) \xi(i_1, i_l) + \log p_i(s_i|i_1) \left( \sum_{\substack{j=1 \\ j \neq i}}^M \xi(i_1, j_1) \right) \right] + \\
& \sum_{i=1}^M \sum_{k=2}^{D_i-1} \left[ \log p_i(i_{k+1}|i_k) \xi(i_k, i_{k+1}) + 0 \right] + \\
& \sum_{i=1}^M \left[ 0 + \log p_i(s_i|i_{D_i}) \left( \sum_{\substack{j=1 \\ j \neq i}}^M \xi(i_{D_i}, j_1) \right) \right]
\end{aligned}$$

Term (2):

$$\begin{aligned}
& \sum_{i=1}^M \left[ \sum_{\substack{j=1 \\ j \neq i}}^M \log A_{ij} \left( \sum_{k=1}^{D_i} \sum_{l=1}^{D_j} \xi(i_k, j_l) \right) \right] = \\
& \sum_{i=1}^M \left[ \sum_{\substack{j=1 \\ j \neq i}}^M \log A_{ij} \left( \xi(i_1, j_1) + \xi(i_{D_i}, j_1) \right) \right]
\end{aligned}$$

Term (3):

$$\begin{aligned}
& \sum_{j=1}^M \left[ \sum_{l=1}^{D_j} \log p_j(j_l | r_j) \left( \sum_{\substack{i=1 \\ i \neq j}}^M \sum_{k=1}^{D_i} \xi(i_k, j_l) \right) \right] = \\
& \sum_{j=1}^M \left[ \sum_{l=1}^{D_j} \log p_j(j_l | r_j) \left( \sum_{\substack{i=1 \\ i \neq j}}^M \xi(i_1, j_l) + \xi(i_{D_i}, j_l) \right) \right] = \\
& \sum_{j=1}^M \left[ \log p_j(j_1 | r_j) \left( \sum_{\substack{i=1 \\ i \neq j}}^M \xi(i_1, j_1) + \xi(i_{D_i}, j_1) \right) \right]
\end{aligned}$$

Lastly, the M-step is the argmax computation for the parameter vectors.

For the  $p_i^{n+1}(\cdot | i_k)$  ( $2 \leq k \leq D_i - 1$ ), the  $p_i^{n+1}(\cdot | i_{D_i})$  and the  $p_j^{n+1}(\cdot | r_j)$  parameter vectors the optimization is trivial, since only one element has positive statistics in each case, which leads to  $p_i^{n+1}(i_{k+1} | i_k) = 1$ ,  $p_i^{n+1}(s_i | i_{D_i}) = 1$  and  $p_j^{n+1}(j_1 | r_j) = 1$ . These are the initial fixed 1s in the representation graph.

From Term (1),  $p_i(\cdot | i_1)$ :

$$\begin{aligned}
p_i^{n+1}(\cdot | i_1) &= \arg \max_{p_i(\cdot | i_1)} \sum_{l=2}^{D_i} \log p_i(i_l | i_1) \xi(i_1, i_l) + \log p_i(s_i | i_1) \sum_{\substack{j=1 \\ j \neq i}}^M \xi(i_1, j_1) \\
&\propto \left( \xi(i_1, i_2), \dots, \xi(i_1, i_{D_i}), \sum_{\substack{j=1 \\ j \neq i}}^M \xi(i_1, j_1) \right)
\end{aligned}$$

From Term (2),  $A_i$ :

$$A_i^{n+1} \propto \left( \xi(i_1, j_1) + \xi(i_{D_i}, j_1) \right)_{j=1, \dots, M}$$

Denote the normalizing constants with  $N_i$  and  $M_i$  respectively:

$$\begin{aligned}
N_i &= \xi(i_1, i_2) + \dots + \xi(i_1, i_{D_i}) + \sum_{\substack{j=1 \\ j \neq i}}^M \xi(i_1, j_1), \\
M_i &= \sum_{\substack{j=1 \\ j \neq i}}^M \xi(i_1, j_1) + \xi(i_{D_i}, j_1).
\end{aligned}$$

To finalize the M-step we can assign the new  $\tilde{A}_{i_k, j_l}^{n+1}$  values to every non-fixed  $(i_k, j_l)$  edge:

$$\begin{aligned}\tilde{A}_{i_1, i_l}^{n+1} &= p_i^{n+1}(i_l|i_1) = \frac{\xi(i_1, i_l)}{N_i}, \\ \tilde{A}_{i_1, j_1}^{n+1} &= p_i^{n+1}(s_i|i_1)A_{ij}^{n+1} = \frac{\sum_{j \neq i}^M \xi(i_1, j_1)}{N_i} \cdot \frac{\xi(i_1, j_1) + \xi(i_{D_i}, j_1)}{M_i}, \\ \tilde{A}_{i_{D_i}, j_1}^{n+1} &= A_{ij}^{n+1} = \frac{\xi(i_1, j_1) + \xi(i_{D_i}, j_1)}{M_i}.\end{aligned}$$

## 6.2 Time complexity of M-step

It is true in general (not just for categorical) that the time-complexity of the M-step is  $\mathcal{O}(TE)$ :

- $\mathcal{O}(M + \sum_{i=1}^M D_i)$  for the initial distribution
- $\mathcal{O}(TE)$  for the transition probabilities  $(A_{ij}, (\eta_1, \dots, \eta_M))$ :
  1.  $\mathcal{O}(TE)$  for computing  $\xi(i_k, j_l) = \sum_{t=2}^T \xi_{t-1, t}^n(i_k, j_l)$  values for  $\{(i_k, j_l) : \tilde{A}_{i_k, j_l}^0 > 0\}$ , the others are zeroes
  2.  $\mathcal{O}(E)$  for computing  $\sum_{j=1}^M \sum_{l=1}^{D_j} \xi(i_k, j_l)$  coefficients for all  $(i_k, s_i)$  exit edges:  $\sum_{i=1}^M e_{out}^i \sum_{j=1}^M e_{in}^j \mathbb{I}(A_{ij}^0 > 0) \leq E$ , because  $\xi(i_k, j_l) > 0$  implies that  $(i_k, s_i)$  exit edge,  $A_{ij}^0 > 0$  and  $(r_j, j_l)$  entry edge
  3.  $\mathcal{O}(E)$  for computing  $\sum_{k=1}^{D_i} \sum_{l=1}^{D_j} \xi(i_k, j_l)$  coefficients for all  $\{(i, j) : A_{ij} > 0\}$ : similarly as previous
  4.  $\mathcal{O}(E)$  for computing  $\sum_{i=1}^M \sum_{k=1}^{D_i} \xi(i_k, j_l)$  coefficients for all  $(r_j, j_l)$  entry edges: similarly as previous
  5.  $\mathcal{O}(E)$  for updating  $p_i(i_l|i_k)$  and  $p_i(s_i|i_k)$  parameters for all  $i_k$ : for each  $i$ , we have  $e^i + e_{out}^i$  non-zero edges in  $G_i$ ,  $\sum_{i=1}^M e^i + e_{out}^i \leq E$
  6.  $\mathcal{O}(E)$  for updating  $A_{ij}$  parameters for all  $i, j$ :  $\sum_{i=1}^M \sum_{j=1}^M \mathbb{I}(A_{ij} > 0) \leq E$
  7.  $\mathcal{O}(E)$  for updating  $p_j(j_l|r_j)$  parameters for all  $j_l$ :  $\sum_{j=1}^M e_{in}^j \leq E$
  8.  $\mathcal{O}(E)$  for assigning every non-zero  $(i_k, j_l)$  edge their new  $\tilde{A}_{i_k, j_l} =$

$p_i(s_i|i_k)A_{ij}p_j(j_l|r_j)$  probability and every  $(i_k, i_l)$  edge their new  $\tilde{A}_{i_k, i_l} = p_i(i_l|i_k)$  probability

- $\mathcal{O}(T(M + \sum_{i=1}^M D_i))$  for the emission probabilities ( $B_{il}$ ):
  1.  $\mathcal{O}(T \sum_{i=1}^M D_i)$  for summing up  $\gamma$  values:  $\gamma_t(i) \doteq \sum_{k=1}^{D_i} \gamma_t^n(i_k)$
  2.  $\mathcal{O}(TM)$  for updating  $B_{il}$  parameters:  $\sum_{i=1}^M \sum_{l=1}^L \log B_{il} \sum_{t \in T_l} \gamma_t(i)$  as in simple HMM (for all  $i$ :  $B_{il}$  needs  $|T_l|$  additions)
  3.  $\mathcal{O}(T \sum_{i=1}^M D_i)$  for assigning the corresponding emission probabilities:  $\tilde{B}_{i_k, l} = B_{il}$

In summary, we have that the M-step can be done in  $\mathcal{O}(TE)$  time, such as in the simple Baum-Welch algorithm, and therefore one EM iteration for GD-HMM takes  $\mathcal{O}(TE)$  time.

## 7 Efficiency of representation in GD-HMM

As we have already seen, the number of (non-zero) edges is the key measure of the time complexity of the forward-backward algorithm (and EM algorithm) in any HMM.

We advance the usefulness of the number of edges and define the efficiency of representation.

**Definition 10.** (*Representation efficiency of GD-HMM*)

Let  $\theta = (\pi, A, \theta_o)$  be an HMM and let  $T_i$  be duration distributions represented with graphs  $G_i$ . The full efficiency of the representation is the number of edges in the resulting GD-HMM:

$$E(\{G_i\}, \{T_i\}, \theta) = \sum_{i=1}^M e^i + \sum_{i=1}^M \sum_{\substack{j=1 \\ j \neq i}}^M e_{out}^i e_{in}^j \mathbb{I}(A_{ij} > 0).$$

We would like to measure how efficient the representation of  $T_i$  is with  $G_i$ , therefore we should create a simpler definition of efficiency that does not depend on the  $\theta$  HMM. We can examine only the GD-HMMs from complete HMMs ( $\forall i \neq j : A_{ij} > 0$ ).

**Definition 11.** (*Representation efficiency function*)

Let  $\theta$  be a complete HMM. Let  $T_i$  be duration distributions represented with graphs  $G_i$ . The efficiency-function of representation is  $E : \mathbb{N}_+ \rightarrow \mathbb{N}_+$  defined by the following:

$$E(\{G_i\}, \{T_i\})(M) = \sum_{i=1}^M e^i + \sum_{i=1}^M \sum_{\substack{j=1 \\ j \neq i}}^M e_{out}^i e_{in}^j.$$

Now we can measure the goodness of representations together. Next, we want to measure the efficiency of individual representations. The motivation is that

each  $T_i$  may come from the same family. To succeed, we assume that every  $T_i$  is represented with  $G(\eta_i)$  (the representation graphs have the same structure).

**Definition 12.** (*Representation efficiency function of graphs*)

Let  $\{T(\theta) : \theta \in \Theta\}$  be a parametric family of duration distributions. Let  $G$  be the representation graph of  $\{T(\theta)\}$ . The efficiency function of the representation is the following:

$$E(G, \{T(\theta)\})(M) = Me(G) + M(M - 1)e_{out}(G)e_{in}(G).$$

which is simply the previous definition for the case of  $G$  representing all  $T_i$ .

Remember that the geometric distribution representation graph  $G(p)$  has the following edge numbers:  $e_{in} = 1$ ,  $e_{out} = 1$ ,  $e = 1$ . Therefore the efficiency function is  $E(G(p), Geo(p))(M) = M + M(M - 1) = M^2$  which is the number of edges in a complete HMM.

From the previous definition, it is clear that we want more efficient representations for duration distribution families; i.e. representations with fewer edges.

For example consider the family of categorical distributions on  $\{1, \dots, D\}$  with parameters  $p_1, \dots, p_D$ . Here I present the construction of three different graphs  $G_1, G_2, G_3$ , each of them represents the family, but with different efficiency.

Graph  $G_1$  has  $D + 2$  nodes and has the following non-zero probability transitions:

- $p(v_d|r) = p_{D+1-d}$  for  $d = 1, \dots, D$ ,
- $p(v_d|v_{d-1}) = 1$  for  $d = 2, \dots, D$ ,
- $p(s|v_D) = 1$ .

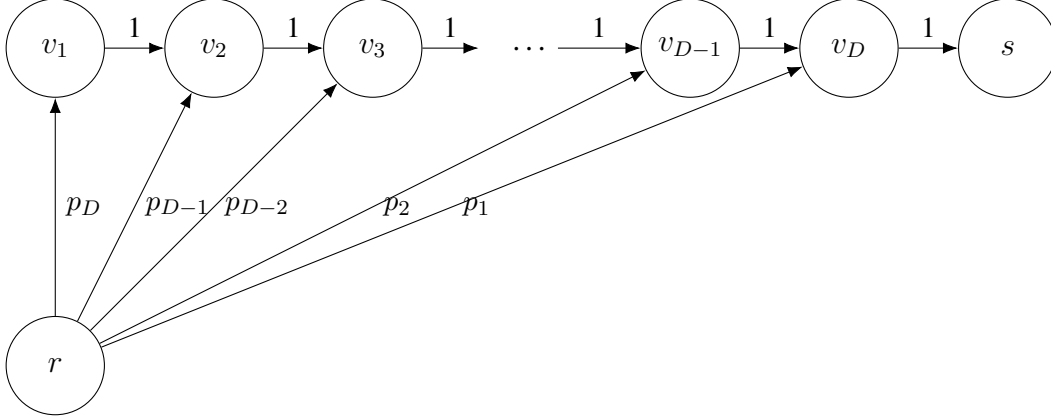


Figure 5:  $G_1$  representation graph for  $Cat(\{1, \dots, D\})$  distribution.

The efficiency is  $M(D - 1) + M(M - 1)D$ . This representation comes from Yu & Kobayashi [6].

Graph  $G_2$  has  $D + 2$  nodes and has the following non-zero probability transitions:

- $p(v_1|r) = 1$ ,
- $p(v_d|v_1) = p_{D+2-d}$  for  $d = 2, \dots, D$ ,
- $p(v_d|v_{d-1}) = 1$  for  $d = 3, \dots, D$ ,
- $p(s|v_D) = 1$ ,
- $p(s|v_1) = p_1$ .



The efficiency is  $M(D-1)(D-2)/2 + M(M-1)D^2$ . This is the worst of the three representations.

The following statements tell us that the second representation is optimal.

**Statement 16.** (*Optimal representation of categorical distributions*)

Let  $\{T(\theta) : \theta \in \Theta\}$  be the family of categorical distributions on  $\{1, \dots, D\}$ , with  $\theta = (p_1, \dots, p_D)$ . The  $G$  graph represents this family. Then

$$E(G, \{T(\theta)\})(M) \geq M(D-1) + M(M-1).$$

*Proof.* Remember the walk-based description:

$$P(X_{d+1} = s \text{ ft}) = \sum_{w \in W_{d+1}} \prod_{e \in w} p(e).$$

Let  $p_D > 0$  and let  $G(\eta)$  represent  $T(p_1, \dots, p_D)$ . Then:

$$0 < p_D = P_{G(\eta)}(X_{D+1} = s \text{ ft}) = \sum_{w \in W_{D+1}} \prod_{e \in w} p_\eta(e).$$

Notice that  $p(e)$  only depends on  $\eta$ , because  $G$  is fixed. So we have at least one  $D+1$ -length  $w = (r, x_1, \dots, x_D, s)$   $r \rightarrow s$  walk with positive probability.

The  $x_1, \dots, x_D$  nodes are all inner nodes.

I claim they are all different. Assume that  $\exists i < j : x_i = x_j$ . In this case,  $C = x_i \dots x_j$  is a positive circle. Denote the walk before  $C$  with  $w_0$ , and the walk after  $C$  with  $w_1$ . Let  $c = |C| \geq 1$  be the length of the circle. Then  $\forall j \geq 1$ :  $w^j = w_0 C^{j+1} w_1$  is a positive walk with length  $D+1+jc$ . Thus  $T[G(\eta)](D+jc) > 0 \forall j$ , but  $T(p_1, \dots, p_D)(D+jc) = 0$ . Because  $G(\eta)$  represents  $T$ , all nodes have to be different.

We have  $D$  different inner nodes:  $x_1, \dots, x_D$ . Using the positive walk  $w = (r, x_1, \dots, x_D, s)$ :  $e_{in} \geq 1$ ,  $e_{out} \geq 1$  and  $e \geq D-1$ . We have:

$$E(G, \{T(\theta)\})(M) = Me(G) + M(M-1)e_{out}(G)e_{in}(G) \geq M(D-1) + M(M-1).$$

□

Thus, the second representation has efficiency  $\mathcal{O}(MD + M^2)$  and the optimal efficiency is also  $\mathcal{O}(MD + M^2)$ . The time complexity of the forward-backward algorithm (and one iteration of EM) is  $\mathcal{O}(TE)$  in a sparse graph. Now  $E = MD + M^2 \ll (MD)^2$ . This leads to an  $\mathcal{O}(T(MD + M^2))$  complexity using the efficient representation for the categorical family. No better time complexity can be achieved with a different representation and this complexity is the same as in [6].

## 8 Numerical experiments

In this section, I use the deduction of the M-step for the categorical GD-HMM (Section 6) to implement the model. In the Python ecosystem, the *hmmlearn* (<https://github.com/hmmlearn/hmmlearn>) package gives a fine, modular implementation for different observation models; the version I used (0.2.6) supports Gaussian, Gaussian Mixture Model and, Multinomial (more precisely Categorical) distributions. It also has a BaseHMM class, enabling the developers to implement their own HMM models for other observation distribution families.

For all the next occurrences of Multinomial distribution, I refer to the Categorical distribution, i.e. we are given a set of symbols:  $\{1, \dots, L\}$ , and at each step, we are emitting only one symbol. Actually, the MultinomialHMM class (from *hmmlearn*) is designed for the categorical observation model, but in some fields (e.g. natural language processing) the name multinomial distribution is used for the categorical distribution. (In version 0.2.8. a CategoricalHMM class is used for categorical observations and the MultinomialHMM has a parameter  $n\_trials$  to support multinomial distributions.)

As it turned out, the GD-HMM model is an HMM with special properties. The calculations and algorithms for HMMs can be used in the building of the GD-HMM model. More precisely, only the initialization and M-step procedures should be changed (see Section 6).

For the experiments, I implemented a MultinomialCatGDHMM class by subclassing the MultinomialHMM class and overwriting the `_init` and `_do_mstep` methods. The class handles the Categorical Graphical-Duration Hidden Markov Model for Multinomial (Categorical) observations (discussed throughout the thesis) with the graph architecture discussed in Statement 11 and Section 5. For simplicity, the maximum duration parameter is the same for all super states  $\forall i : D = D_i$ .

## 8.1 Monotonically increasing log-likelihood of GD-HMM

As it was discussed in Section 3, the EM algorithm increases the observed data log-likelihood  $l(\theta^n) = \log p(x_{1:T}|\theta^n)$ . In Section 2, we have seen that with the Forward algorithm we can compute the log-likelihood for a given parameter vector (besides the  $\alpha$  values). In the *hmmlearn* implementation, the *fit* method follows the Baum-Welch algorithm, and the log-likelihood is computed as part of the Forward algorithm.

I tested the implemented MultinomialCatGDHMM class with different model initializations and training data. In each case, the log-likelihood of the model increased monotonically, which proves that the M-step is developed correctly and the model actually (statistically) fits the data.

## 8.2 Log-likelihood and AIC evaluation of GD-HMM

**Definition 13.** (*Akaike Information Criterion - AIC*)

*For a statistical model with  $k$  parameters and  $\hat{l}$  maximized log-likelihood, the AIC score is the following:*

$$AIC = 2k - 2\hat{l}.$$

The AIC score measures the goodness of fit for a statistical model (it contains the log-likelihood) but penalizes the number of parameters. If the model has too many parameters, it means that the model overfits, just as in other fields of machine learning. A lower score of AIC means a better model.

In this experiment series, the log-likelihood (LL) and AIC scores of the GD-HMM and a plain HMM were compared when both models were trained on data generated from a GD-HMM.

The Markov models had only  $n\_super\_states = 2$  (super states are 0 and 1) and  $n\_symbols = 2$  (symbols/observations are 0 and 1). In each experiment, the following hyperparameters were chosen: *max\_duration* (between 2 and 11),

*sample\_size* (between 1000 and 16000), and *n\_iter* (between 100 and 800) for the number of EM learning iterations.

An experiment had the following structure:

1. Initialize a data generator GD-HMM with *n\_super\_states*, *max\_duration*, and random weights,
2. Generate  $X$  observation sequence with length of *sample\_size* using the data generator GD-HMM,
3. Initialize 10 different GD-HMMs with *n\_super\_states*, *max\_duration*, and random weights. Train all 10 models for *n\_iter* iterations, select the best GD-HMM based on log-likelihood,
4. Initialize 10 different HMMs with *n\_super\_states* number of states and random weights. Train all 10 models for *n\_iter* iterations, select the best HMM based on log-likelihood,
5. Calculate the number of parameters, log-likelihood, and AIC score of both the GD-HMM and HMM.

We would like to see that the GD-HMM fits the (GD-HMM generated) data better than the HMM. This means that the log-likelihood is higher, and the AIC score is smaller.

In the following table, one can see that GD-HMM is similar to HMM in log-likelihood when random initialization for emission probabilities was used in the data generator model. Because of the random initialization, the generated data was possibly close to noise, therefore these results do not give insights into the performance of GD-HMM.

n_super_states	max_duration	n_symbols	sample_size	n_iter	gdhmm_n_params	hmm_n_params	gdhmm_ll	hmm_ll	gdhmm_ll > hmm_ll	gdhmm_aic	hmm_aic	gdhmm_aic < hmm_aic
2	2	2	1000	100	5	5	-537.4	-537.6	True	1084.8	1085.3	True
2	2	2	1000	200	5	5	-651.9	-652.3	True	1313.9	1314.5	True
2	2	2	2000	200	5	5	-1344.5	-1349.6	True	2699.0	2709.2	True
2	2	2	4000	200	5	5	-2533.8	-2533.6	False	5077.6	5077.2	False
2	2	2	4000	400	5	5	-2569.3	-2569.9	True	5148.7	5149.7	True
2	2	2	2000	800	5	5	-1381.3	-1385.6	True	2772.6	2781.3	True
2	3	2	1000	100	7	5	-691.7	-692.6	True	1397.4	1395.1	False
2	3	2	1000	200	7	5	-676.1	-677.4	True	1366.2	1364.8	False
2	3	2	2000	200	7	5	-1217.1	-1221.2	True	2448.2	2452.3	True
2	3	2	2000	400	7	5	-1368.5	-1370.2	True	2751.0	2750.4	False
2	7	2	1000	100	15	5	-688.4	-688.2	False	1406.8	1386.5	False
2	7	2	2000	100	15	5	-1381.0	-1382.2	True	2791.9	2774.5	False
2	7	2	4000	100	15	5	-2753.5	-2753.8	True	5537.0	5517.6	False
2	7	2	8000	100	15	5	-5492.5	-5496.6	True	11014.9	11003.2	False
2	7	2	16000	100	15	5	-11061.3	-11061.3	False	22152.6	22132.6	False

Table 1: Experiment with randomly initialized emission probabilities

However, when the emission probabilities were initialized as the symbols 0 and 1 are highly correlated with the hidden states 0 and 1 ( $p(x_t = 0|z_t = 0) = p(x_t = 1|z_t = 1) = 0.9$ ), then a significant advantage for GD-HMM emerged.

n_super_states	max_duration	n_symbols	sample_size	n_iter	gdhmm_n_params	hmm_n_params	gdhmm_ll	hmm_ll	gdhmm_ll > hmm_ll	gdhmm_aic	hmm_aic	gdhmm_aic < hmm_aic
2	7	2	1000	100	15	5	-577.8	-633.0	True	1185.5	1275.9	True
2	2	2	1000	100	5	5	-606.4	-618.1	True	1222.8	1246.2	True
2	2	2	1000	100	5	5	-552.6	-685.3	True	1115.1	1380.7	True
2	2	2	1000	100	5	5	-595.1	-655.4	True	1200.1	1320.7	True
2	5	2	1000	100	11	5	-621.1	-665.6	True	1264.2	1341.2	True
2	5	2	1000	100	11	5	-620.4	-666.1	True	1262.8	1342.2	True
2	5	2	1000	100	11	5	-646.6	-679.6	True	1315.2	1369.2	True
2	11	2	1000	100	23	5	-546.8	-599.2	True	1139.6	1208.4	True
2	11	2	1000	100	23	5	-577.2	-608.9	True	1200.4	1227.7	True
2	11	2	1000	100	23	5	-546.7	-594.6	True	1139.5	1199.2	True
2	11	2	1000	200	23	5	-539.4	-581.4	True	1124.9	1172.7	True
2	11	2	1000	200	23	5	-561.6	-602.3	True	1169.1	1214.7	True
2	11	2	1000	200	23	5	-545.8	-581.6	True	1137.5	1173.1	True
2	11	2	2000	200	23	5	-1108.3	-1190.4	True	2262.5	2390.7	True
2	11	2	2000	200	23	5	-1126.7	-1203.1	True	2299.3	2416.1	True
2	11	2	2000	200	23	5	-1100.2	-1178.6	True	2246.5	2367.2	True

Table 2: Experiment with correlated emission probabilities

### 8.3 Simulated manufacturing use case for GD-HMM

Consider the following simplified scenario of a manufacturing machine. The machine can be in two (hidden) states DOWN and UP in each second, which are the states of not producing and producing respectively. The movement of the machine is measured in each second resulting in either a MOVING or NOT MOVING observation. The hidden states are highly correlated with the movement

measurements: in the DOWN state, with the probability of 95% the machine is NOT MOVING (so the probability of MOVING is 5% in DOWN), and in the UP state, with the probability of 80% the machine is MOVING (so the probability of NOT MOVING is 20% in UP).

Although the measurements do not give us precisely the hidden information of the states, we know a prior that the machine works in a certain way: the DOWN state lasts between 9 and 11 seconds and the UP state lasts between 4 and 6 seconds.

With this information in hand, one would prefer using the categorical distribution for the duration. During the learning, the model can learn that in the DOWN state the probabilities of staying in 9-11 seconds are positive, while the others are close to zero (and similarly to the UP state).

The following figure shows the hidden machine states in blue (0-DOWN and 1-UP) and the movement observations in red (0-NOT MOVING and 1-MOVING) in the first 400 seconds.

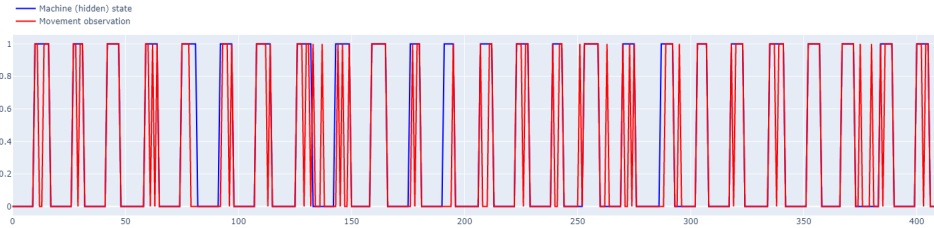


Figure 8: Simulated manufacturing states and observations I.

With movement observations of length 1000, a GD-HMM and an HMM were trained. The GD-HMM has a  $max\_duration = 15$ , and both models were trained for 100 iterations. The resulting GD-HMM has 31 parameters, a log-likelihood of -386, and an AIC score of 834. The HMM has 5 parameters, a log-likelihood of -531, and an AIC score of 1071. The hidden states were inferred using the Viterbi

decoding for both models. The GD-HMM fits the data better, and the following figure shows this perfectly. The GD-HMM prediction of hidden states is in green, the HMM prediction is in yellow.

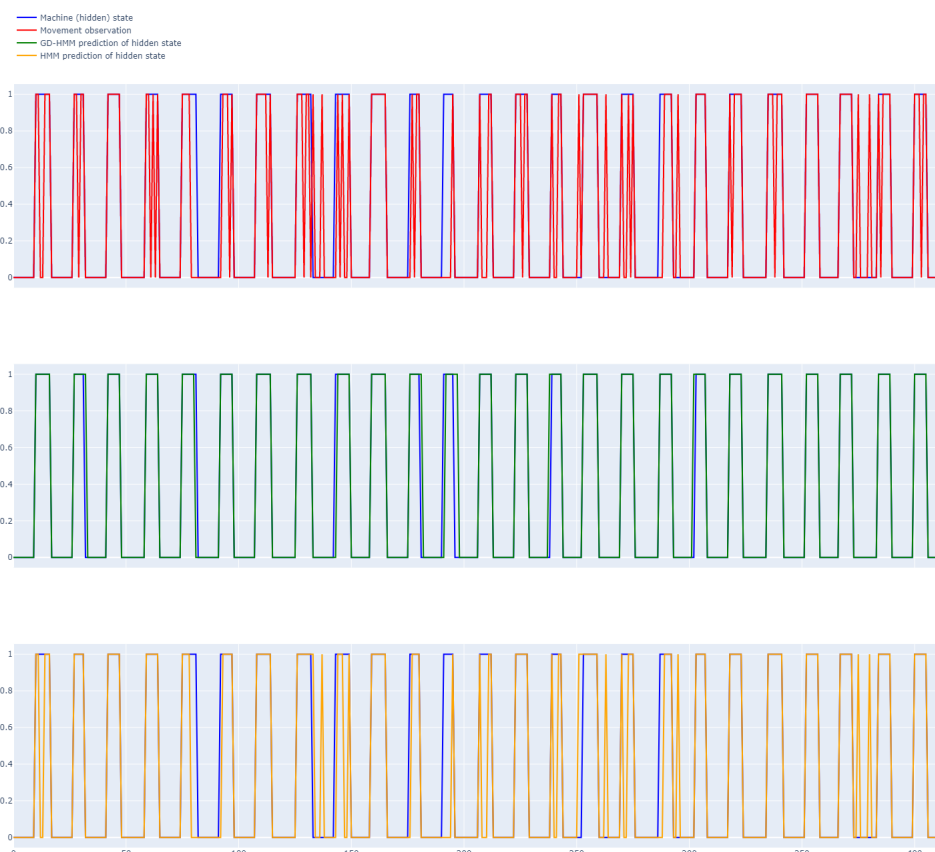


Figure 9: GD-HMM and HMM prediction of hidden states I.

The GD-HMM gives back all the hidden UP/DOWN states only with 1-2 second differences in the starts or ends. At the same time, HMM does some filtering but fails to recover the hidden states.

A stronger version of the previous experiment is presented next. The entropy in observation generation was increased (e.g setting MOVING and NOT MOVING

probabilities to 50% in the UP state) and the *max\_duration* parameter was also increased (to 30) for the trained GD-HMM (which is the equivalent of less prior information or more parameters in model). In exchange, also the length of the observation sequence has been increased, from 1000 to 4000.

The machine states and observations in this setting:

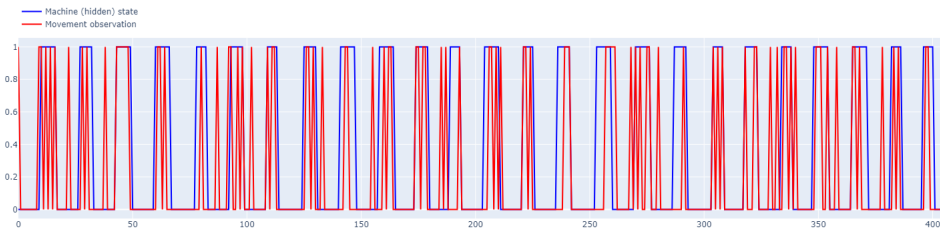


Figure 10: Simulated manufacturing states and observations II.

In this case, the resulting GD-HMM has 61 parameters, a log-likelihood of -1710, and an AIC score of 3543. The HMM has 5 parameters, a log-likelihood of -1995, and an AIC score of 4000. With the increased observation length, both the log-likelihood and AIC score shows that GD-HMM fits the data better. The figure clearly shows that GD-HMM reveals the true location and width of UP states, and HMM fails because it cannot learn the duration time.

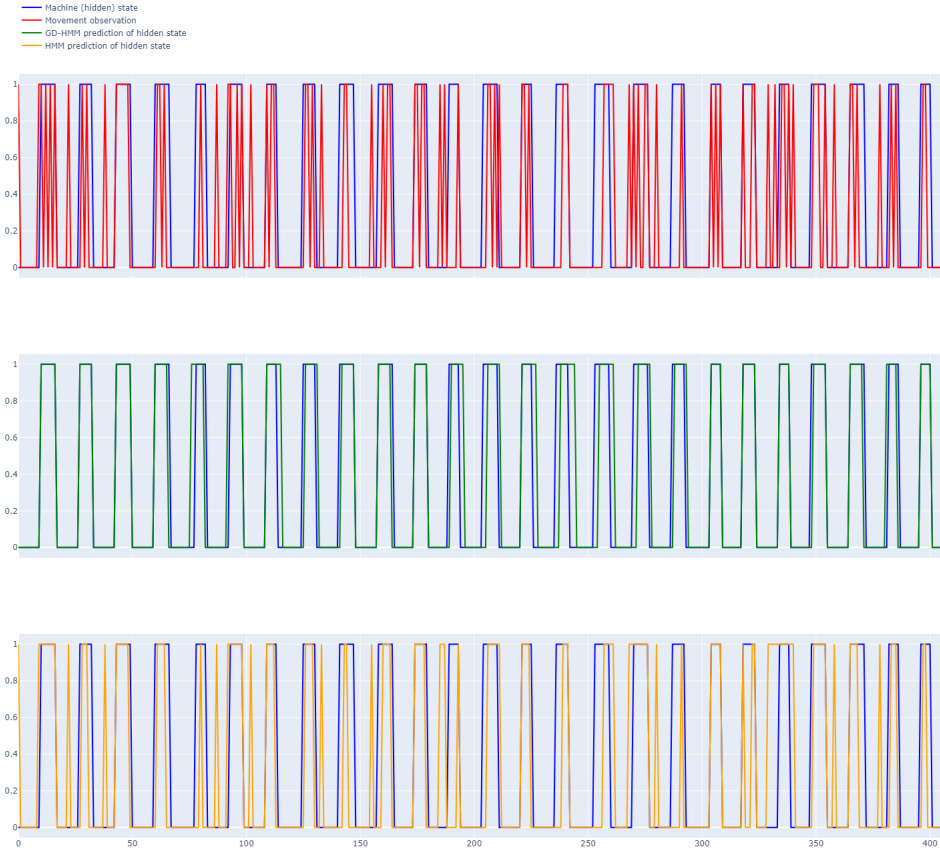


Figure 11: GD-HMM and HMM prediction of hidden states II.

The GD-HMM reveals all the UP states along the 4000 seconds, even the ones where no MOVING observations were generated, which is a very powerful result showing that the usage of duration information can compensate for the lack of useful observation signals in Hidden Markov Models.

## 9 Conclusion

Throughout the thesis, I established a new HMM framework, the Graphical-Duration Hidden Markov Model, which is able to represent not only geometric duration distributions, but many others. I identified a few duration families that have a graph representation, and also provided a distribution property that excludes all graph representations. I examined the EM learning of the GD-HMM and deduced the analytical form of updates for the Categorical (Duration Distribution) GD-HMM with categorical emissions. The resulting EM variant has the same time complexity as the Baum-Welch's:  $\mathcal{O}(TE)$ . I showed that a distribution can be represented with multiple different graphs, and provided an optimal representation for the categorical distribution family. Finally, I gave small proofs that the implemented GD-HMM outperforms the plain HMM if the data has specific distribution.

The main motivation was to stretch the HMM framework in order to build duration information into the model. Although other HSMM variants were implemented in the past, I came up with a partially new, general framework which is equivalent to the HMM framework, but can use duration information.

As the visual proofs in Section 8.3 show, the GD-HMM has great potential over HMM, if one has the expert knowledge or information about the times in the modelled process. The continuation of this work should be the usage of GD-HMM for modelling a practical process, where the duration distributions are not geometric.

## References

- [1] O. Cappé, E. Moulines, and T. Ryden. *Inference in Hidden Markov Models (Springer Series in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2005.

- [2] C. Mitchell, M. Harper, and L. Jamieson. On the complexity of explicit duration HMM's. *IEEE Transactions on Speech and Audio Processing*, 3(3):213–217, 1995.
- [3] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.
- [4] L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. pages 257–286, 1989.
- [5] S.-Z. Yu. Hidden Semi-Markov Models. *Artificial Intelligence*, 174(2):215–243, 2010. Special Review Issue.
- [6] S.-Z. Yu and H. Kobayashi. An efficient forward-backward algorithm for an Explicit-Duration Hidden Markov Model. *IEEE Signal Processing Letters*, 10(1):11–14, 2003.

## NYILATKOZAT

Név: Keresztes László

ELTE Természettudományi Kar, szak: Alkalmazott matematikus MSc

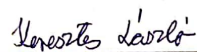
NEPTUN azonosító: YKO46E

Szakedolgozat címe:

Graphical-Duration Hidden Markov Model

A szakdolgozat szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló szellemi alkotásom, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2023.01.01.



a hallgató aláírása