Baum-Welch Algorithm for Hidden Markov Models Using Algebraic Decision Diagrams

Sebastian Aaholm*, Lars Emanuel Hansen†, Daniel Runge Petersen‡,

Abstract—This is a placeholder abstract. The whole template is used in semester projects at Aalborg University (AAU).

1 Introduction

In this section we present some introductory ways to use the tools within LaTeX in general, and this template in particular. For example, this is a citation [1], while this is a multi-citation[1, 2].

The column width of the IEEE template is 3.5 inches, so if you generate your plots with this width or less, the output will be the best. For example, Listing 1 contains the code to generate the image in Figure 1 using Python with matplotlib, and exported as pgf (TEX).

```
import matplotlib.pyplot as plt

plt.rcParams.update({
    "pgf.texsystem": "pdflatex",
    "font.family": "serif",  # use serif/main font for text elements
    "pgf.preamble": "\n".join([
         r"\usepackage[utf8x]{inputenc}",
         r"\usepackage[T1]{fontenc}",
    ]),
})

fig, ax = plt.subplots(figsize=(3.5, 3.5))

ax.plot(range(5))
ax.text(0.5, 3., "serif")
ax.text(0.5, 2., "monospace")
ax.text(2.5, 2., "sans-serif")
ax.set_xlabel(r"µ is not $\mu$")

fig.tight_layout(pad=.5)
fig.savefig("graph.pgf")
```

Listing 1. Code to generate the graph.pgf

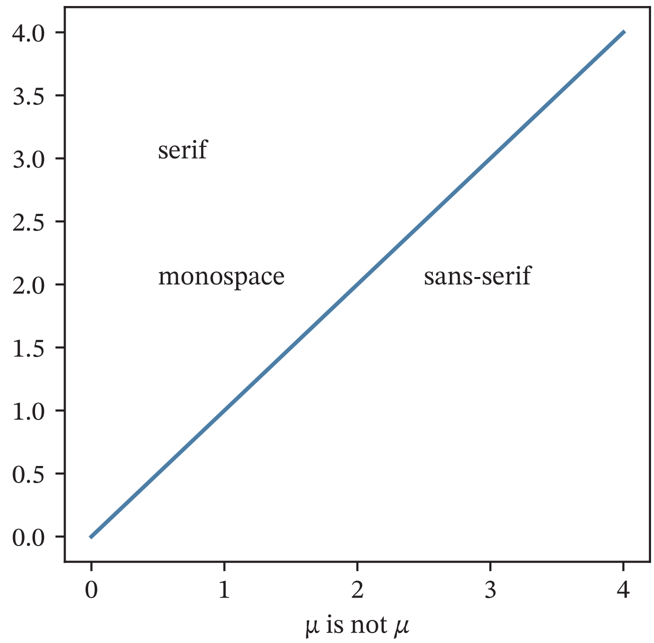

Fig. 1. An example graph drawn using Python's matplotlib library.

1.1 Tables and Figures

1.2 Algorithms, Theorems, and Proofs

There are a few different things outside the normal figure and table floats that are very relevant when writing a scientific paper or article. For example, you may wish to typeset theorems as in Theorem 1.

Theorem 1 (Pythagorean theorem)**.** *This is a theorem about right triangles and can be summarized in the next equation*

$$x^2 + y^2 = z^2$$

Or ref like Theorem 1 Similarly, for proofs:

Proof. To prove it by contradiction try and assume that the statement is false, proceed from there and at some point you will arrive to a contradiction. □

Note that proofs are not a numbered environment, and as such can't be referenced by default.

2 Preliminaries

This section provides an overview of the theoretical background necessary to understand the rest of the report. We begin by defining the key concepts of a Hidden Markov Model (HMM) and a Markov Decision Process (MDP), which are the two main models used in this report.

- *All authors are with the Dept. of Computer Science, Aalborg University, Aalborg, Denmark*
- *E-mails: *saahol20, †leha20, ‡dpet20 @student.aau.dk*

TABLE 1
Example of a pretty, twocolumn table.

	Klasser				
Hændelser	Reservation	Gæst	Borgerforening	Kalender	Betaling
Anmodet	✓	✓	✓		
Godkendt	✓		✓		
Afvist	✓		✓		
Redigeret	✓	✓	✓		
Annulleret	✓	✓	✓		✓
Betalt					✓
Refunderet					✓
Kvitteret		✓	✓		
Registreret	✓			✓	
Påmindet		✓	✓		

```
INSERTION-SORT(A, n)
for i ← 2 to n
    key ← A[i]
    // Insert A[i] into the sorted subarray A[1 : i − 1].
    j ← i − 1
    while j > 0 and A[j] > key
        A[j + 1] ← A[j]
        j ← j − 1
    A[j + 1] ← key
```

Algorithm 1. Test

2.1 Hidden Markov Model

HMMs were introduced by Baum and Petrie in 1966 [**NOTFOUND**] and have since been widely used in various fields, such as speech recognition [**NOTFOUND**], bioinformatics [**NOTFOUND**], and finance [**NOTFOUND**].

Definition 1 (Hidden Markov Model)**.** *A Hidden Markov Model (HMM) is a tuple $\mathcal{M} = (S, \mathcal{L}, \ell, \tau, \pi)$, where:*

- *S is a finite set of states.*
- *$\mathcal{L}$ is a finite set of labels.*
- *$\ell : S \to D(\mathcal{L})$ is the emission function.*
- *$\tau : S \to D(S)$ is the transition function.*
- *$\pi \in D(S)$ is the initial distribution.*

$D(X)$ denotes the set of probability distributions over a finite set X. The emission function ℓ describes the probability of emitting a label given a state. The transition function τ describes the probability of transitioning from one state to another. The initial distribution π describes the probability of starting in a given state. An HMM is a statistical model that describes a system that evolves over time. The system is assumed to hold the Markov property, meaning that the future state of the system only depends on the current state and not on the past states. The system is also assumed to be unobservable, meaning that the states are hidden and cannot be directly observed. Instead, the system emits observations, which are used to infer the hidden states.

An example of an HMM is a weather model where the hidden state represents the actual weather (sunny, rainy, or cloudy), but we only observe indirect signals, such as whether someone is carrying an umbrella.

2.2 Continuous Time Hidden Markov Model

In the above definition, we have defined a discrete-time HMM, meaning that the system evolves in discrete time steps. In this report, we are interested in continuous-time systems, where the system evolves in continuous time. To model continuous-time systems, we use a Continuous Time Hidden Markov Model (CTHMM), which is an extension of the HMM to continuous time.

Definition 2 (Continuous Time Hidden Markov Model)**.** *A Continuous Time Hidden Markov Model (CTHMM) is a tuple $\mathcal{M} = (S, \mathcal{L}, \ell, \tau, \pi, \lambda)$, where S, $\mathcal{L}$, ℓ, τ, and π are as defined in the HMM definition.*

- *$\lambda : S \times S \to \mathbb{R}_{\geq 0}$ is the rate function.*

The rate function λ determines the transition rate between states, meaning that the time spent in a state before transitioning follows an exponential distribution.

2.3 Markov Decision Process

MDPs were introduced by Bellman in 1957 [**NOTFOUND**] and have since been widely used in various fields, such as robotics [**NOTFOUND**], finance [**NOTFOUND**], and healthcare [**NOTFOUND**].

Definition 3 (Markov Decision Process)**.** *A Markov Decision Process (MDP) is a tuple $\mathcal{M} = (S, A, \mathcal{L}, \ell, \tau, R, \gamma)$, where:*

- *S is a finite set of states.*
- *A is a finite set of actions.*
- *$\mathcal{L}$ is a finite set of labels.*
- *$\tau : S \times A \to D(S)$ is the transition function.*
- *$R : S \times A \to \mathbb{R}$ is the reward function.*
- *$\pi \in D(S)$ is the initial distribution.*

A MDP works by an agent interacting with an environment. Unlike an HMM, the agent takes actions in the environment and receives rewards. An agent is the decision-maker that interacts with the environment by taking actions. The environment is modeled as a MDP, which consists of a set of states S, a set of actions A, a set of labels $\mathcal{L}$. Each state is directly observable, meaning the agent always knows which state it is in. When the agent takes an action in a state, it transitions to a new state according to the transition function τ.

The reward function R describes the reward received when taking an action in a state. The goal of an MDP is to find a policy $\theta : S \rightarrow D(A)$ that maximizes the expected cumulative reward. The policy θ describes the probability of taking an action given a state.

Definition 4 (Policy)**.** *A policy θ is a function that maps states to actions, i.e., $\theta : S \rightarrow D(A)$.*

An example of an MDP is a robot navigating a grid, where it can choose actions (move up, down, left, or right) and receives rewards based on reaching certain goal locations

3 Experiments

In this section, we describe the experiments conducted to evaluate the performance of the symbolic implementation of the Baum-Welch algorithm in the CuPAAL library by comparing it to the recursive implementation in Jajapy. The evaluation is based on two key aspects: execution time and accuracy.

We conduct two experiments:

- **Performance Comparison** - Measuring runtime and accuracy across different models.
- **Scalability Analysis** - Evaluating performance as the number of states increases.

These experiments aim to answer the following research questions:

- **Question 1**: How does the symbolic implementation of the Baum-Welch algorithm in CuPAAL compare to the recursive implementation in Jajapy in terms of runtime and accuracy?
- **Question 2**: How does the performance of the CuPAAL implementation scale with the number of states in the model?

3.1 Experimental Setup

All experiments are conducted using a set of Discrete Time Markov Chains (DTMCs) and Continuous Time Markov Chains (CTMCs) obtained from publicly available benchmarks[1].

Each experiment is run ten times, and results are reported as the average runtime and number of iterations. We stop the experiments when we reach a convergence threshold of 0.05, which was the default value in the Jajapy implementation. The training data is randomly generated based on these models, consisting of a specified number of observation sequences of fixed length.

3.1.1 Experiment 1: Performance Comparison of Forward-Backward Implementations

The first experiment is based on the ideas from the experiment conducted in [3]. The models used in this experiment are shown in Table 2 and Table 3. The experiment evaluates the efficiency and accuracy of the symbolic approach (CuPAAL) versus the recursive approach (Jajapy). We measure:

- **Runtime Efficiency** - The average time per run.
- **Convergence Speed** - The average number of iterations required.
- **Accuracy** - Measured using log-likelihood and average relative parameter error.

The **relative parameter error** is calculated as follows $\frac{|r-e|}{e}$, where r is the real value and e is the expected value.

Log-likelihood: Measures how well the learned model explains the observed data. Given an observation sequence O and a model M, the log-likelihood is calculated as:

$\log P(O \mid M) = \sum_{t=1}^{T} \log P(O_t \mid M)$ where $P(O_t|M)$ is the probability of observing O_t given the model.

Following the approach in [4], initial parameters are randomly sampled from the range [0.00025, 0.0025]. The implementations used are:

1) The original Jajapy implementation.
2) The symbolic CuPAAL implementation.

We expect the CuPAAL implementation to have the same accuracy as the Jajapy implementation and have the same number of iterations. However, we expect the CuPAAL implementation to be faster due to the symbolic approach, which can avoid redundant calculations.

The results are presented for DTMCs in Table 4, Table 5, Table 6, Table 7, and Table 8.

For CTMCs, the results are shown in Table 9, Table 10, Table 11, Table 12, and Table 13. All tables show the average time per run for each implementation, average number of iterations and log-likelihood and relative parameter error.

TABLE 2
DTMC models

Name	Number of States	Number of Parameters
Leader_sync	274	2
Herman	512	1
Oscillators	453	6
Brp	886	2
Crowds	1145	2

TABLE 3
CTMC models

Name	Number of States	Number of Parameters
Toggle-switch	99	1
Mapk	118	2
Cluster	820	3
Embedded	3480	2
Kanban	58000	1

TABLE 4
Leader_sync results

Implementation	Iter	Time(s)	Avg δ	Log-likelihood
Jajapy	0	0	0	0
CuPAAL	0	0	0	0

1. The models are available at https://qcomp.org/benchmarks/. The source files describe the parameters and what is observable.

TABLE 5
Herman results

implementation	Iter	Time(s)	avg δ	log-likelihood
Jajapy	0	0	0	0
CuPAAL	0	0	0	0

TABLE 6
Oscillators results

implementation	Iter	Time(s)	avg δ	log-likelihood
Jajapy	0	0	0	0
CuPAAL	0	0	0	0

TABLE 7
Brp results

implementation	Iter	Time(s)	avg δ	log-likelihood
Jajapy	0	0	0	0
CuPAAL	0	0	0	0

TABLE 8
Crowds results

implementation	Iter	Time(s)	avg δ	log-likelihood
Jajapy	0	0	0	0
CuPAAL	0	0	0	0

TABLE 9
Toggle-switch results

implementation	Iter	Time(s)	avg δ	log-likelihood
Jajapy	0	0	0	0
CuPAAL	0	0	0	0

TABLE 10
Mapk results

implementation	Iter	Time(s)	avg δ	log-likelihood
Jajapy	0	0	0	0
CuPAAL	0	0	0	0

TABLE 11
Cluster results

implementation	Iter	Time(s)	avg δ	log-likelihood
Jajapy	0	0	0	0
CuPAAL	0	0	0	0

TABLE 12
Embedded results

implementation	Iter	Time(s)	avg δ	log-likelihood
Jajapy	0	0	0	0
CuPAAL	0	0	0	0

TABLE 13
Kanban results

implementation	Iter	Time(s)	avg δ	log-likelihood
Jajapy	0	0	0	0
CuPAAL	0	0	0	0

missing picture

Fig. 2. Scalability results for the tandem model

3.2 Scalability Experiment

The primary objective of this experiment is to evaluate the scalability of the proposed symbolic implementation of the Baum-Welch algorithm in comparison to the recursive implementation in Jajapy. Specifically, we aim to measure the time required to learn DTMCs and CTMCs over the number of states. We measure:

- **Runtime efficiency** - The average time per run.

The experiment is run on the polling model for CTMCs, where the number of states is increased from 36 to 1334, and the initial parameters are resampled in the range [0.00025, 0.0025]. For DTMCs, we use the leader_sync model, where the number of states is increased from 26 to 1050, and the initial parameters are resampled in the range [0.00025, 0.0025].

The two models were chosen because they are representative of the models used in the performance comparison experiment, and both models can be scaled to a large number of states.

This experiment is conducted to assess scalability, a key factor in Baum-Welch algorithm performance. It provides insights into the efficiency of the symbolic approach for both DTMCs and CTMCs

The scalability is determined by the time taken to learn the model as the number of states increases.

While we expect CuPAAL to have lower runtime and improved scalability, we also expect the accuracy to be consistent with the Jajapy implementation.

The results are shown in Figure 2.

Acronyms

AAU Aalborg University. 1

CTHMM Continuous Time Hidden Markov Model. 2
CTMC Continuous Time Markov Chain. 3, 4

DTMC Discrete Time Markov Chain. 3, 4

HMM Hidden Markov Model. 1, 2

MDP Markov Decision Process. 1, 2

References

[1] M. Goossens, F. Mittelbach, and A. Samarin, *The LaTeX Companion*. Reading, Massachusetts: Addison-Wesley, 1993.

[2] G. D. Greenwade, "The Comprehensive Tex Archive Network (CTAN)," *TUGBoat*, vol. 14, no. 3, pp. 342–351, 1993.

[3] R. Reynouard et al., “On learning stochastic models: From theory to practice,” 2024.

[4] G. Bacci, A. Ingólfsdóttir, K. G. Larsen, and R. Reynouard, “Mm algorithms to estimate parameters in continuous-time markov chains,” 2023. arXiv: 2302.08588 `[cs.LG]`.

Appendix A
Cheatsheet

If something is represented with a greek letter, it is something we calculate.

Symbol	**Meaning**
$\mathbb{R}$	Real numbers
$\mathbb{Q}$	Rational numbers
$\mathbb{N}$	Natural numbers
$s \in S$	States
$l \in L$	Labels
$a \in A$	Actions
$\mathcal{M}$	Markov Model
$\mathcal{H}$	Hypothesis
$o \in O \in \mathcal{O}$	Observations
π	Initial distribution
τ	Transition function
ι or ω	Emission function
α	Forward probabilities
β	Backward probabilities
γ	State probabilities
ξ	Transition probabilities
$\lambda = (\pi, \tau, \omega)$	Model Parameters
ϕ or ψ	Scheduler
μ	Mean
σ	Standard deviation
$\theta = (\mu, \sigma^2)$	Parameters of a distribution
$P(\mathcal{O}; \lambda)$	Probability of $\mathcal{O}$ given λ
$\ell(\lambda; \mathcal{O})$	Log likelihood of λ under $\mathcal{O}$