CS5340 Assignment 3: Hidden Markov Models

1. Overview

In this assignment, you will implement the Baum-Welch algorithm to find the unknown parameters of 1-dimensional Gaussian hidden Markov models (HMMs).

References: Lecture 8

Honour Code. This coding assignment constitutes **15%** of your final grade in CS5340. Note that plagiarism will not be condoned! You may discuss with your classmates and check the internet for references, but you MUST NOT submit code/report that is copied directly from other sources!

2. Submission Instructions

Items to be submitted:

- **Source code (lab3.py).** This is where you fill in all your code.
- **Report (report.pdf).** This should describe your implementation and be no more than one page.

Please indicate clearly your name and student number (the one that looks like A1234567X) in the report as well as the top of your source code. Zip the two files together and name it in the following format: **A1234567X_lab3.zip** (replace with your student number).

Submit your assignment by **29 October2023, 2359HRS** to Canvas. 25% of the total score will be deducted for each day of late submission.

3. Test Cases

As with previous assignments, we have provided a few test cases and their expected outputs, which can be found in the **data/** folder. You can find the code that loads the sample data and checks your program output in **test_lab3.py**. During grading, we will also check your code with additional held out cases.

Note that for the overall Baum-Welch implementation, we are unable to provide the checking code since your algorithm output may differ slightly from ours. Instead, we provide the groundtruth parameters, and you should use them to check the outputs. In addition, we also list the parameters obtained using our reference implementation in the Appendix.

4. NOTATIONS

We will use the same notations as the lectures.

- K: Number of states
- N: Number of time steps
- $\boldsymbol{\theta}$: Parameters $\{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\phi}\}$ of the HMM which we want to estimate.
- $\boldsymbol{\pi}$: Initial state distribution, represented as a numpy array of size K.
- $\mathbf{A}$: Time-independent stochastic transition matrix. This is represented as a $K \times K$ numpy array.
- $\boldsymbol{\phi}$: Parameters $\{\phi_1, \ldots, \phi_K\}$ of the conditional distribution of the emission probabilities. For the Gaussian HMM we are considering, it consists of the means $\{\mu_1, \ldots, \mu_K\}$ and standard deviations $\{\sigma_1, \ldots, \sigma_K\}$. Each is implemented as a numpy array of size K.
- $\mathbf{X}$: Observed data $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
- $\mathbf{Z}$*:* Latent variables $\{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$
- $\gamma(\mathbf{z}_n)$: Marginal posterior distribution, i.e. $p(\mathbf{z}_n|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$
- $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$: Joint posterior distribution of two successive latent variables, $p(\mathbf{z}_{n-1}, \mathbf{z}_n|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$

5. BAUM-WELCH ALGORITHM

As we have learned in the lectures, the Baum-Welch algorithm, also known as the forward-backward algorithm is used to find the unknown maximum likelihood parameters of a hidden Markov model, i.e.

$$\text{argmax}_{\boldsymbol{\theta}}\, p(\mathbf{X}|\boldsymbol{\theta})$$

The above quantity is intractable to maximise directly, so the Baum-Welch algorithm uses an Expectation-Maximization (EM) scheme to perform the maximization. The EM algorithm starts with an initialization of the model parameters, which we denote by $\boldsymbol{\theta}^{\text{old}}$. It then alternates between the E-step and M-step.

The E-step finds the posterior distribution $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{old})$ of the latent variables. This allows us to evaluate the expression for the expectation of the log complete-data likelihood:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \mathbb{E}_{\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}}[\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})] = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$$

The M-step then maximises $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ w.r.t. to the parameters $\boldsymbol{\theta}$**.**

5.1 E-STEP

For an HMM, the expected log complete-data likelihood can be represented as:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{k=1}^{K} \gamma(\mathbf{z}_{1k}) \ln \boldsymbol{\pi}_k + \sum_{n=2}^{N} \sum_{j=1}^{K} \sum_{k=1}^{K} \xi(\mathbf{z}_{n-1}, \mathbf{z}_n) \ln \mathbf{A}_{jk} + \sum_{n=1}^{N} \sum_{j=1}^{K} \gamma(\mathbf{z}_{nk}) \ln p(\mathbf{x}_n | \boldsymbol{\phi}_k)$$

The goal of the E-step is thus to compute the quantities of the marginal and posterior distributions $\gamma(\mathbf{z}_n)$ and $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$.

Now, implement the function **`e_step()`** using the alpha-beta variant of the algorithm taught in the lectures. This function takes as input $\mathbf{X}$ and the estimated parameters from the M-step, $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\phi}\}$. It outputs $\gamma(\mathbf{z}_n)$ and $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$.

Since the training data $\mathbf{X}$ is a list of sequences, the outputs from your function $\gamma(\mathbf{z}_n)$ and $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$ should also be a list of the same cardinality. You should compute the outputs for each sequence separately, and concatenate them into a list.

As mentioned in the lectures, the values of α can go to zero quickly for long sequences. <u>You will need to implement the scaling factors to receive full credit for this portion.</u>

Additional Hints:

- We have provided two sequences to check the implementation of your E-step. Your algorithm should work on the short sequence even without the scaling factors; you can use it to check your implementation before you include the scaling factors.
- The indexing of dimensions may be tricky. Use the time indices (i.e. whether it's $\mathbf{z}_{n-1}$ or $\mathbf{z}_n$) to keep track of which dimensions should add/multiply with each other.

5.2 M-STEP

Now, write the code for **`m_step()`**, which implements the M-step of the Baum-Welch algorithm. This function should take in $\mathbf{X}, \gamma(\mathbf{z}_n), \xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$ and estimate the HMM parameters $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\phi}\}$.

For this assignment, we will only consider 1D Gaussian HMMs, so the emission probabilities parameters $\boldsymbol{\phi}$ consist of the means $\{\mu_1, \dots, \mu_K\}$ and $\{\sigma_1, \dots, \sigma_K\}$.

5.3 PUTTING THEM TOGETHER

You are now ready to put the e-step and m-step together. Implement **`fit_hmm()`** which calls your implemented e- and m-step functions to estimate the HMM parameters.

To facilitate grading, we have implemented an initialization routine for $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\phi}\}$ which uses a *fixed random seed*. The randomly initialised $\boldsymbol{\pi}, \mathbf{A}$ will satisfy the required summation and non-negativity constraints. To obtain the initial gaussian parameters $\boldsymbol{\phi} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$, we run k-means clustering on the observed measurements which will provide a good starting point for your EM algorithm. **You should make use of the initialized θ in your first iteration of the e-step.**

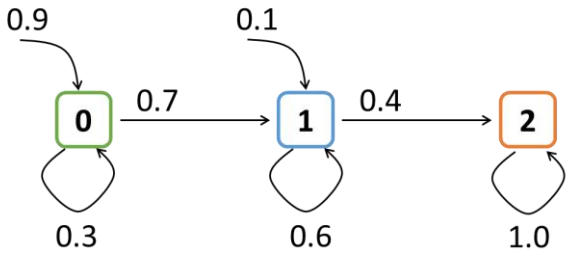
The last thing to handle is the convergence condition. For this, you can use any reasonable scheme. Our suggestion is to monitor the parameters $\boldsymbol{\theta}$, and terminate it when their values do not change much in the iteration (e.g. the parameters change by a maximum of <1e-4).

As mentioned earlier, we are unable to provide the checking code for this part since your algorithm output may differ slightly from ours. Instead, do refer to the appendix for the groundtruth parameters used to generate the observations, as well as the values estimated by our reference implementation.

During grading, we will evaluate your solution using the likelihood $p(\mathbf{X}|\boldsymbol{\theta})$.

APPENDIX: TRAINING DATASETS

In this section, we show the data we provided for checking your code, as well as our estimated parameters.



	State 0	State 1	State 2
μ	-1.5	0.5	-0.2
σ	0.51	0.21	0.29

(a) Model used to generate data

π	0.91	0.09	0.0

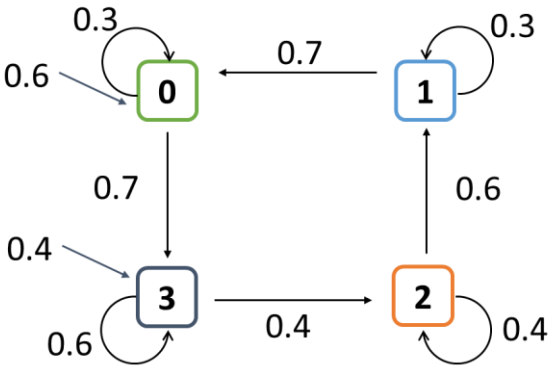
A	0.28	0.72	0.0
	0.0	0.6	0.4
	0.0	0.0	1.0

ϕ:

μ	-1.5	0.52	-0.21
σ	0.51	0.21	0.29

(b) Our Estimated parameters

Figure 1: "seq_short " dataset, which consists of 200 sequences each having 8 timesteps: (a) The groundtruth model used to generate the observations, and (b) Estimated parameters using the reference implementation.



	State 0	State 1	State 2	State 3
μ	0.0	1.1	2.0	1.0
σ	0.5	0.3	0.4	0.2

(a) Model used to generate data

π	0.64	0.0	0.0	0.36

A	0.3	0.0	0.0	0.7
	0.7	0.3	0.0	0.0
	0.0	0.58	0.42	0.0
	0.0	0.0	0.43	0.57

ϕ:

μ	0.0	1.07	2.0	1.0
σ	0.52	0.31	0.41	0.19

(b) Our Estimated parameters

Figure 2: "seq_long" dataset, which consists of 5 sequences each having 1000 timesteps. (a) The groundtruth model used to generate the observations, and (b) Estimated parameters using the reference implementation.