CS 307 Introduction to Machine Learning
Fall 2022
Assignment 6: Hidden Markov Models
Part I (Written Problems): *Due by Wednesday, December 21, 11:59 p.m.*
Part II (Extra Credit): *Due by Friday, December 30, 11:59 p.m.*

Instructions:

1. Your solution to this assignment must be submitted via gradescope.
2. For the written problems, submit your solution as a **single PDF** file to the *Assignment 6 - Part 1 (CS 307)* folder.
   - We prefer that you type your solution. If you choose to hand-write your solution, scan your PDF using a **scanner** and upload it. Make sure your final PDF is **legible**. **Regrades due to non-compliance will receive a 30% score penalty**.
   - Verify that both your answers and procedure are **correct, ordered, clean, and self-explanatory** before writing. Please ask yourself the following questions before submitting:
     - Are my answers and procedure legible?
     - Are my answers and procedure in the same order as they were presented in the assignment? Do they follow the specified notation?
     - Are there any corrections or scratched out parts that reflect negatively on my work?
     - Can my work be easily understood by someone else? Did I properly define variables or functions that I am using? Can the different steps of my development of a problem be easily identified, followed, and understood by someone else? Are there any gaps in my development of the problem that need any sort of justification (be it calculations or a written explanation)? Is it clear how I arrived to each and every result in my procedure and final answers? Could someone describe my submission as messy?
3. **IMPORTANT:** As long as you follow these guidelines, your submission should be in good shape; if not, we reserve the right to penalize answers and/or submissions as we see fit.

Part I: Written Problems (30 points)

1. **Hidden Markov Models (30 points)**

   Consider an HMM defined by three states (y_1, y_2, y_3), three output symbols (x_1, x_2, x_3, x_4), and the following transition probabilities (see the left table) and emission probabilities (see the right table). The HMM starts with y_1, y_2, and y_3 with probabilities 0.5, 0.4, and 0.1 respectively.

q_{t-1} \ q_t	y_1	y_2	y_3
y_1	0.5	0.2	0.3
y_2	0.1	0.6	0.3
y_3	0.2	0.4	0.4

y_i \ x_i	x_1	x_2	x_3
y_1	0.2	0.7	0.1
y_2	0.4	0.4	0.2
y_3	0.7	0.1	0.2

   (a) **(5 pts)** What is $P(x_1x_2)$? Show your work.

   (b) **(4 pts)** What is $P(y_1y_2y_3)$? Show your work.

   (c) **(7 pts)** What is $P(q_2 = y_2 \mid o_2 = x_2)$? Show your work.

   (d) **(7 pts)** What is $P(q_2 = y_2 \mid o_2 = x_2, o_1 = x_1)$? Show your work.

   (e) **(7 pts)** What is $P(q_2 = y_2 \mid o_1 = x_1)$? Show your work.

Part II: Extra Credit Programming (80 points)

Part II is entirely **optional** and is intended for those of you who are interested in getting bonus points. In this problem you will write a program that uses the Baum-Welch algorithm to perform unsupervised training of an HMM.

Implementation Details:

- To implement the Baum-Welch algorithm, you should use Python (3.6.9), C++ (g++ 7.5.0), or Java (openjdk 11.0.13 2021-10-19). Do **not** use any non-standard libraries (except numpy (1.19.5) and pandas (1.1.5)) in your code.
- The number of states, the number of output symbols, and the number of EM iterations should be determined at runtime. Given an HMM with n states, we assume without loss of generality that the n-th state is the final/stop state (i.e., no symbol will be emitted from the n-th state and the sequence will terminate once this state is reached). In addition, the states are denoted as s_1, s_2 ... s_n. The m output symbols will be denoted as o_1, o_2, ..., o_m.
- Your program should allow exactly seven arguments to be specified in the **command line** invocation of your program:

  1. n (the number of states in the HMM);
  2. m (the number of output symbols);
  3. the number of EM iterations;

4. the path to a `.start` file that specifies how the start probabilities should be initialized. Specifically, this file contains the start probabilities $\pi(o_i)$, for $i = 1, \ldots, n$. Line i of this file stores $\pi(o_i)$.
5. the path to a `.trans` file that specifies how the transitiion probabilities should be initialized. Specifically, this file contains an n by n matrix, where the (i, j)-th entry is the probability of transitioning from state s_i to state s_j.
6. the path to a `.emit` file that specifies how the emission probabilities should be initialized. Specifically, this file contains an n by m matrix, where the (i, j)-th entry is the probability of emitting symbol o_j from state s_i; and
7. the path to a `.data` file that consists of the unlabeled observation sequences used for training.

Your program should allow exactly these seven arguments to be specified in the command line invocation of your program. There should be no graphical user interface (GUI). Any program that does not conform to the above specification will receive no credit. Sample input files can be found in the assignment page of the course website.

- For an HMM with n states and m symbols, the `.start` file should contain n rows corresponding to the n states, the `.trans` file should contain n-1 rows and n columns, and the `.emit` file should contain n-1 rows and m columns. Note that (1) the probability of starting at the final/stop state can be non-zero and (2) the `.trans` and `.emit` file do not contain any probabilities corresponding to the final/stop state.
- Following the convention we used in lecture, the final/stop state is not explicitly shown in the (unlabeled) training sequences in the data file. Every observation sequence in the training file is assumed to terminate at the final/stop state.
- Your algorithm should process the training sequnces in the same order as they appear in the data set.
- The model parameters should be initialized using the values read from the `.start`, `.trans`, and `.emit` files.
- You may submit as many source files as needed, but you must make sure you provide a main code entry that follows the following naming convention. Specifically, if you are using:
  - **Python**
    - Make sure that your primary source file is `main.py` and that your code runs successfully after executing `python main.py <num_states> <num_symbols> <num_iter> <path_to_start_file> <path_to_trans_file> <path_to_emit_file> <path_to_data_file>`.
  - **C++**
    - Make sure that your primary source file is `main.cpp` and that your code runs successfully after executing `g++ main.cpp -o a.out -std=c++17` and `./a.out <num_states> <num_symbols> <num_iter> <path_to_start_file> <path_to_trans_file> <path_to_emit_file> <path_to_data_file>`.

- **Java**
  - Make sure that your primary source file is `Main.java` and that your code runs successfully after executing `javac Main.java` and `java Main <num_states> <num_symbols> <num_iter> <path_to_start_file> <path_to_trans_file> <path_to_emit_file> <path_to_data_file>`.

What to Do

Use Baum-Welch to learn an HMM from the given unlabeled observation sequences. After each EM iteration, print to `stdout` (1) the parameters re-estimated in the M-step, including the start probabilities, the transition probabilities, and the emission probabilities. Assuming that there are 3 states with 4 output symbols, the output for the first two iterations may look like this (do not worry about the actual values shown below; they are all made up):

```
After iteration 1:
pi_1:  0.6569 pi_2:  0.3431 pi_3:  0.0000
a_11:  0.2124 a_12:  0.4010 a_13:  0.3866
a_21:  0.2008 a_22:  0.1685 a_23:  0.6307
b_1(o1):  0.3313 b_1(o2):  0.2821 b_1(o3):  0.2198 b_1(o4):
0.1668
b_2(o1):  0.1563 b_2(o2):  0.2130 b_2(o3):  0.2848 b_2(o4):
0.3459

After iteration 2:
pi_1:  0.8609 pi_2:  0.1391 pi_3:  0.0000
a_11:  0.1377 a_12:  0.6833 a_13:  0.1790
a_21:  0.0456 a_22:  0.1006 a_23:  0.8538
b_1(o1):  0.4237 b_1(o2):  0.3972 b_1(o3):  0.1064 b_1(o4):
0.0726
b_2(o1):  0.0585 b_2(o2):  0.0877 b_2(o3):  0.4082 b_2(o4):
0.4455
```

To facilitate the grading process, please follow the format of the sample output provided in the assignment page of the course website when printing to `stdout`. In particular, when printing the means, the variances, and the priors, print exactly the four digits after the decimal place.

Submission

Once you are done, sign in to gradescope. You will be able to see *Assignment 6 - Part 2 (CS 307)* under the Assignments section. Directly submit all your source files to this submission folder. Do not create any folder and do not rename the files or upload the files in a zip file or folder (your homework will not be graded otherwise). Each group should hand in a single copy of the code. The names of all the members of the group should appear in the README file.

Grading

We will be using an output-based auto-grader for this submission, so make sure you follow the formatting from the example test files: be careful not to insert extra lines, tabs instead of spaces, etc ... When you submit, your code will be **graded using hidden test cases**, so we encourage you to test your code thoroughly. More information about the autograder will be available on Piazza shortly.