Middle East Technical University   Department of Computer Engineering

CENG 462

Artificial Intelligence

Fall '2021-2022

Homework 6

Due date: 5 February 2022, Saturday, 23:55

1 Objectives

This assignment aims to familiarize you with the Viterbi algorithm at the implementation level and enable you to gain hands-on experience in solving problems with this method.

2 Problem Definition

Hidden Markov Models are probabilistic frameworks to model temporal processes (processes execute in sequential time steps) where the internal workings of the processes are unknown but particular information which is generated during the processes can be obtained. HMMs aim to learn the dynamics of these processes by solely depending on this observable information. The internal evolution of the processes is represented with hidden states. Learning corresponds to approximating (extracting) transition (which governs state to state transition dynamics) and observation emission functions (which governs the observation generation in each state). In this assignment, we focus on the problem of finding the most probable state sequence in accordance with the provided observations by a particular process. For this purpose, you are expected to implement the Viterbi algorithm.

In this assignment, we assume that the transition and observation emission functions of processes are known already (e.g a hidden Markov model might have been trained to learn them) and processes execute in discrete time intervals (states change and observations are emitted in discrete time steps). Figure 1 represents one of such processes. Each state is represented with a circle where the observations are drawn as ellipses. State transition functions are shown with directed arrows originating from all states and probability scores. Similarly, the observation emission functions are provided with different colors. Tables 1 and 2 represent this information in a tabular format.

State	state1	state2
state1	0.2	0.8
state2	0.3	0.7

Table 1: State transition probabilities of Figure 1.

Observation	state1	state2
obs1	0.75	0.4
obs2	0.25	0.6

Table 2: Observation emission probabilities of Figure 1.

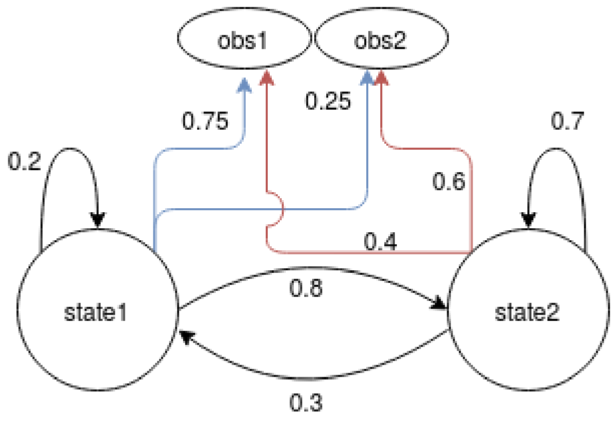

Figure 1: An example process with the state transition and observation emission probabilities.

3 Specifications

- You are going to implement the Viterbi algorithm in Python 3.
- Each problem is represented with a file and this file will be fed to your implementation.
- You are expected to implement the following function:

```python
def viterbi(problem_file_name):
```

  whose parameter specification is as follows:

  - `problem_file_name`: specifies the path of the problem file to be solved. It takes values such as "process1.txt", "process2.txt", etc.

  During the evaluation process, this function will be called and returned information will be inspected. The returned information should be as follows:

```python
state_sequency, sequence_probability, state_probabilities = viterbi(
    problem_file_name)
```

  - `state_sequence`: is the array containing the most probable sequence of states for the given observations.
  - `sequence_probability`: is the probability value of the most probable state sequence.
  - `state_probabilities`: is the dictionary that stores the intermediate probability scores of each state at each process time step and has the following form: {state1:[value1, value2, value3 ...], state2: [value4, value5, value6,...], ...}.

- Problems are described in ".txt" files and have the following structure:

```
[states]
state1|state2|state3|....
[start probabilities]
state1:value1|state2:value2|state3:value3...
[transition probabilities]
state1-state2:value1|state1-state3:value2|...
[observation probabilities]
state1-observation1:value1|state1-observation2:value|...
[observations]
observation1|observation2|observation3|...
```

Each section of the file provides information for a different part of the process. The first section lists all states that are in the process. Each state is separated with the '|' symbol. Since in order to perform probability calculations with the algorithm we need to know the initial probability distribution of the states (at time step 0), the second section provides it. The third section stores the transition probability function. Transition probabilities between each pair of states are separated with "|". Probability values are given after ":". Similarly, the fourth section provides the observation emission function. Finally, the last section keeps the observation sequence depending on which we need to find the most probable state sequence. The sequence is separated with "|".

State and observation names could be any string sequence (of alphanumeric characters) that does not contain the space character (`" "`) (e.g. state1, state3, observation1, observation2, sunny, rainy, snowy).

In a process, there could be an arbitrary number of states and observations.

- No `import` statement is allowed and all methods should be implemented in a single solution file. You are expected to implement all the necessary operations by yourself without utilizing any external library.
- Commenting is crucial for understanding your implementation and decisions made during the process. Your implementation can be manually inspected at random or when it does not satisfy the expected outputs.

4 Sample I/O

process1.txt:

```
[states]
state1|state2
[start probabilities]
state1:0.50|state2:0.50
[transition probabilities]
state1-state1:0.20|state1-state2:0.80|state2-state1:0.30|state2-state2:0.70
[observation probabilities]
state1-obs1:0.75|state1-obs2:0.25|state2-obs1:0.40|state2-obs2:0.60
[observations]
obs1|obs2|obs2|obs2|obs1
```

Expected outputs of the `viterbi` function with different methods on applied on process1.txt:

```
>>> import hw6solutioneXXXXXXX as hw
>>> hw.viterbi("process1.txt")
(['state1', 'state2', 'state2', 'state2', 'state2'], 0.008890559999999999, {'
    state1': [0.375, 0.018750000000000003, 0.013500000000000002, 0.00567,
    0.007144199999999999], 'state2': [0.2, 0.18000000000000002, 0.0756,
    0.031751999999999996, 0.008890559999999999]})
```

process2.txt:

```
[states]
state1|state2|state3
[start probabilities]
state1:0.30|state2:0.50|state3:0.20
[transition probabilities]
state1-state1:0.50|state1-state2:0.30|state1-state3:0.20|state2-state1:0.10|
    state2-state2:0.75|state2-state3:0.15|state3-state1:0.25|state3-state2:0.45|
    state3-state3:0.30
[observation probabilities]
state1-obs1:0.25|state1-obs2:0.25|state1-obs3:0.50|state2-obs1:0.10|state2-obs2
    :0.60|state2-obs3:0.30|state3-obs1:0.35|state3-obs2:0.15|state3-obs3:0.50
[observations]
obs1|obs2|obs3
```

Expected outputs of `viterbi` function with different methods applied on process2.txt:

```
>>> import hw6solutioneXXXXXXX as hw
>>> hw.viterbi("process2.txt")
(['state2', 'state2', 'state2'], 0.0050625, {'state1': [0.075, 0.009375,
    0.00234375], 'state2': [0.05, 0.022500000000000003, 0.0050625], 'state3':
    [0.06999999999999999, 0.0031499999999999996, 0.0016875000000000002]})
```

Note: The problem files and their outputs here are provided as additional files on ODTUClass.

5 Regulations

1. **Programming Language:** You must code your program in Python 3. Since there are multiple versions and each new version adds a new feature to the language, in order to concur on a specific version, please make sure that your implementation runs on İnek machines.

2. **Implementation:** You have to code your program by only using the functions in the standard module of python. Namely, you **cannot** import any module in your program.

3. **Late Submission:** No late submission is allowed. Since we have a strict policy on submissions of homework in order to be able to attend the final exam, please pay close attention to the deadlines.

4. **Cheating: We have zero-tolerance policy for cheating**. People involved in cheating (any kind of code sharing and codes taken from the internet included) will be punished according to the university regulations.

5. **Discussion:** You must follow ODTUClass for discussions and possible updates on a daily basis. If you think that your question concerns everyone, please ask them on ODTUClass.

6. **Evaluation:** Your program will be evaluated automatically using the "black-box" technique so make sure to obey the specifications. A reasonable timeout will be applied according to the complexity of test cases. This is not about the code efficiency, its only purpose is avoiding infinite loops due to an erroneous code. In the case your implementation does not conform to expected outputs for the solutions, it will be inspected manually. So having comments that explain the implementation in an elaborate manner may become crucial for grading in this case.

6 Submission

Submission will be done via ODTUClass system. You should upload a **single** python file named in the format **hw6_e<your-student-id>.py** (i.e. hw6_e1234567.py).

7 References

- For the `Viterbi` algorithm, you can refer to the lecture notes.
- Announcements Page
- Discussions Page