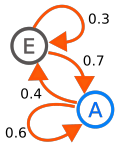
Middle East Technical University | Department of Computer Engineering

CENG 499

Introduction to Machine Learning

Fall '2018-2019
Assignment 3



Yusuf Mücahit Çetinkaya
{yusufc@ceng.metu.edu.tr}
Due date: January 11, 2019, Friday, 23:59

1 Overview

In this assignment, you are going to work on a probabilistic model, namely Hidden Markov Models(HMM) by implementing two algorithms: forward and viterbi. Hidden Markov models (HMMs) are a ubiquitous tool for modeling time series data. They are used in almost all current speech recognition systems, in numerous applications in computational molecular biology, in data compression, and in other areas of artificial intelligence and pattern recognition. Recently HMMs have also been used in computer vision applications such as image sequence modeling and object tracking.[1]

Your implementations will be done with Python programming language. You are allowed to use only numpy library for numerical operations. You will **not** be given any templates.

Keywords: *probabilistic model, HMM, forward, viterbi*

2 Hidden Markov Models

A hidden Markov model is a tool for representing probability distributions over sequences of observations. The hidden Markov model gets its name from two defining properties. First, it assumes that the observation at time t was generated by some process whose state S_t is *hidden* from the observer. Second, it assumes that the state of this hidden process satisfies the *Markov property*: that is, given the value of S_{t-1}, the current state S_t is independent of all the states prior to $(t-1)^a$. In other words, the state at some time encapsulates all we need to know about the history of the process in order to predict the future of the process. The outputs also satisfy a Markov property with respect to the states: given S_t, Y_t is independent of the states and observations at all other

time indices.[1]
HMM consists of five parts;

$Q = q_1 q_2 .. q_N$	a set of N states
$A = a_{11} a_{12} a_{13} ... a_{1N} ... a_{NN}$	a transition probability matrix A, where $a_{ij} represents$ the transition from state i to state j and $\sum_{j=1}^{N} a_{ij} = 1 i.$
$O = o_1 o_2 ... o_T$	a sequence of T observations each drawn from a vocabulary $V = v_1, v_2, v_3, ..., v_V$
$B = b_i(o_t)$	a sequence of emission likelihoods, also called emission probabilities where each express the probability of an observation o t being generated from state i.
q_0, q_f	special start and final state that are not associated with observations, together with transition probabilities $a_{01} a_{02} ... a_{0n}$ out of the start state and $a_{1F} a_{2F} ... a_{nF}$ into the end state

3 Tasks

3.1 Forward Algorithm

The forward algorithm is used to calculate the probability of given observations with the probability of a state at a certain time, given the history of evidence. The process is also known as filtering. The forward algorithm is closely related to, but distinct from, the Viterbi algorithm.

Calculation of this probability is easy by using brute force. However, it results in exponenial complexity. Since it needs to calculate the probability for each state N and observation step T, complexity is $O(N^T)$. Instead of exponential time algorithm, you are expected to implement forward algorithm with dynamic programming. You will store the intermediate values at each observation step T for every state N, so that these calculations will be done once for each time step and state. This results in a polynomial time solution $O(N^2 T)$. Details of the algorithm is explained in the recitation.

Executing your python script as follows should print only a floating point number between 0 and 1 which is the probability of the observation for the given HMM;

```
e123456@inek35: python hw3.py forward transition_matrix.txt estimate_matrix.txt
observations.txt
# 3.105820218004176e-07
```

3.2 Viterbi Algorithm

Viterbi algorithm is similar to Forward algorithm, whereas it is used to find the most probable states the HMM would visit given observations. Viterbi also uses dynamic programming to create a polynomial solution where the maximum probability of each state at each timestep is calculated and store. During these calculation it also stores the state with highest probability. After every calculation these stored states are returned starting from timestep 1. Details of the algorithm is explained in the recitation.

Executing your python script as follows should print a list with the indices of the states with the highest probability to be visited;

```
e123456@inek35: python hw3.py viterbi transition_matrix.txt estimate_matrix.txt
observations.txt
# [2, 2, 2, 4, 2, 2, 2, 2]
```

Given arguments are;

1. transition_matrix.txt: It is the file that contains a numpy array which is transition probability matrix with shape of NXN where transitions are given in the same way as the HMM definition. N is the number of hidden states.

2. estimate_matrix.txt: It is the file that contains a numpy array which is estimate probability matrix with shape of NXV where V is the size of observation vocabulary. Each row corresponds to observation estimates for a single state. Observation vocabulary is not given and each column is the index of an onservation unit in the observation vocabulary.

3. observations.txt: It is the file that contains a numpy array which is the observations you need to calculate the probability for with the shape of T. Be careful T is a varying parameter.

4 Submission & Evaluation

1. You will have **Source** and **Docs** directories. You will have only hw3.py file and it will be in Source directory. Any other file will be in the Docs directory. If your directory structure is messy, you will get **-5 pts. penalty**. Unfortunately, you will not get any extra points from this homework.

2. Your source codes will be tried with different configurations and various size data sets.

3. There will **not** be any partial grading!

4. We have zero tolerance policy for **cheating**. It does not matter to take code from internet or any other people to count it as cheating. People involved in cheating will be punished according to the university regulations and will get 0. You can discuss algorithmic choices, but sharing code between each other or using third party code is **strictly forbidden**. Even if you take a "part" of the code from somewhere/somebody else - this is also cheating. Please be aware that there are "very advanced tools" that detect if two codes are similar. So please do not think you can get away with by changing a code obtained from another source.

5. Your codes should be tested on inek machines before submit.

6. Do not put your .pyc file, IDE related files etc. Only .py files will be submitted.

7. Zip all directories and name it as "e<ID> _ <FullNameSurname>.zip" and submit it through COW. If given name is wrong you will get **-5 pts. penalty.** For example: e1234567_YusufMucahitCetinkaya.zip

References

[1] Ghahramani, Z. (2001). An introduction to hidden Markov models and Bayesian networks. International journal of pattern recognition and artificial intelligence, 15(01), 9-42.