Middle East Technical University Department of Computer Engineering

CENG 498

Introduction to Machine Learning

Spring 2017-2018

Homework 3 - Hidden Markov Models

Due date: 04 06 2018, Monday, 23:59

1 Introduction

The objective of this assignment is learn the details of Hidden Markov Models (HMM) by implementing forward and viterbi algorithms. You will implement your HMM algorithms using python and you are only allowed to use numerical library numpy and other standard python libraries. You will be given a python template with empty function definitions to implement these algorithms. These templates contain some example inputs for you to check whether you correctly implemented the algorithms.

2 Hidden Markov Models

HMM is a probabilistic sequence model where it computes the most probable labels to a sequence of units. This problem is common for speech and text applications since they are based on language with specific units that contain certain hidden labels with specific meaning. HMM consists of the following 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 a transition from state i to state j and $\sum_{j=1}^{N} a_{ij} = 1 \; \forall i$.
$O = o_1 o_2 ... o_T$	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 Forward Algorithm

Forward algorithm is used to calculate the probability of given observations with a given HMM state transition and observation probabilities. Using brute force to calculate the probability would result in exponential algorithm. The reason for this is the fact that you need to calculate the probability of observation using every possible state combination at observation time step. This will result in a complexity

$O(N^T)$ where T is the observation step count and N is the number of states. To overcome this problem dynamic programming is used to calculate and store the intermediate values at each step t for every state so that same calculations will only be done once for each time step and state. This results in a polynomial $O(N^2T)$ complexity and becomes a valid solution. Details of this algorithm is given in the recitation document. Your job is to write a generic function that can be applied to any HMM to calculate the probability of observations. Template code of this function is given below:

```
def forward(trans, estimate, obs):
pass
```

Arguments are explained below:

- `trans` is the transition probability matrix with size $N \times N$ where transitions are given in the same way as the HMM definition2. N is the number of states.
- `estimate` is the estimate probability matrix with size $N \times V$ where V is the size of the observation vocabulary. Each row corresponds to observation estimates for a single state. Observation vocabulary is not given and each column is assumed to be the index of a observation unit in the observation vocabulary.
- `obs` is the observations you need to calculate the probability for with size T. T is **NOT** a constant number and can change for every call of the function. Although I will not most likely test it with large number, do not assume a maximum value.

Return value should be a floating point number between 0 and 1 which is the likelihood (probability) of the observation for the given HMM.

4 Viterbi Algorithm

Viterbi algorithm is used to calculate the most probable states the HMM would visit given observations. Similar to forward algorithm, viterbi uses dynamic programming to create a polynomial solution where the maximum probability of each state at each time-step 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 time-step 1. Details of this algorithm is al given in the recitation document. Using this algorithm, you will implement a generic function that can be applied to any HMM to calculate the states given observations. Template code of this function is given below:

```
def viterbi(trans, estimate, orbs):
pass
```

Arguments are explained below:

- `trans` is the transition probability matrix with size $N \times N$ where transitions are given in the same way as the HMM definition2. N is the number of states.
- `estimate` is the estimate probability matrix with size $N \times V$ where V is the size of the observation vocabulary. Each row corresponds to observation estimates for a single state. Observation vocabulary is not given and each column is assumed to be the index of a observation unit in the observation vocabulary.
- `obs` is the observations you need to calculate the most probable states to be visited for, with size T. T is **NOT** a constant number and can change for every call of the function. Although I will not most likely test it with large number, do not assume a maximum value.

Return value should any type of indexable data structure (list, array etc) where each value is the index of the state most likely to be visited. Start and end states should be excluded and the state at time t should be stored at index $t - 1$.

5 Specifications

- The codes must be in python. You are only allowed to use numpy as a non-standard library. Although you do not need it, numpy makes it easier to do vector and matrix operations and will make it easier to implement your solution. You are also allowed to use any standard python library. You are **NOT** allowed to use any other library.
- Your programs should be able to run on department machines running Ubuntu. Grading will be done black-box. However, good coding practices and commenting are highly encouraged. This will help if there is a problem with your solution and need to come to the objection.
- You have total of 3 late days for all your homeworks. For each day you have submitted late, you will lose 10 points. If your late days exceed 3, any homework you have submitted late will be 0. Your used late submission days will be displayed together with your homeworks on cow.
- Using any piece of code that is not your own is strictly forbidden and constitutes as cheating. This includes friends, previous homeworks, or the internet. This is especially critical for the first part of the homework since you are doing all the implementation. The violators will be punished according to the department regulations.
- Follow the course page on cow for any updates and clarifications. Please ask your questions on cow instead of e-mailing if the question does not contain code or solution.

6 Submission

Submission will be done via COW. You will submit a tar file called "hw3.tar.gz" that contain all your source code. Only the functions in hmm.py will be tested.