# Homework 4 Part A: HMM, Viterbi

CS 585, UMass Amherst, Fall 2016

## Overview

Due **Friday Oct 21**.

Get starter code from the course website's schedule page. You should submit a zipped directory (please don't use other compression formats like .rar) named hw4a_YOUR-USERNAME that contains:

- your vit.py file
- writeup

Our course's collaboration policy is specified on the website.

## 1 HMM

*[15 total points]*

Answer the following questions using the transition matrix $T$ and emission probabilities $E$ below. Below, $\Delta$ and $\square$ are two output variables, $A$ and $B$ are two hidden states; $s_n$ refers to the $n^{th}$ hidden state in the sequence and $o_n$ refers to the $n^{th}$ observation.

$T =$

| | A | B | END |
|---|---|---|---|
| START | 0.5 | 0.5 | 0.0 |
| A | 0.2 | 0.3 | 0.5 |
| B | 0.4 | 0.4 | 0.2 |

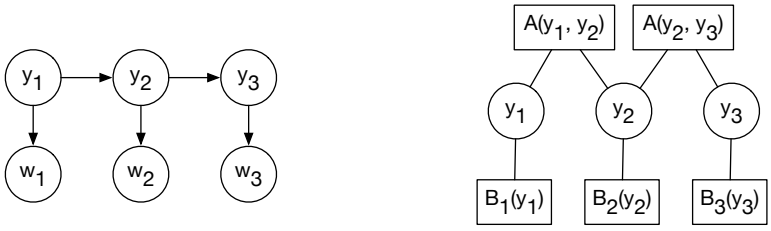
$E =$

| | $\Delta$ | $\square$ |
|---|---|---|
| A | 0.5 | 0.5 |
| B | 0.3 | 0.7 |

1. *[2 points]* Does $P(o_2 = \Delta | s_1 = B) = P(o_2 = \Delta | o_1 = \square)$?
2. *[2 points]* Does $P(s_2 = B | s_1 = A) = P(s_2 = B | s_1 = A, o_1 = \Delta)$?
3. *[3 points]* Does $P(o_2 = \Delta | s_1 = A) = P(o_2 = \square | s_1 = A, s_3 = A)$?
4. *[3 points]* Compute the probability of observing $\square$ as the first emission of a sequence generated by an HMM with transition matrix $T$ and emission probabilities $E$.
5. *[5 points]* Compute the probability of the first state being $A$ given that the last token in an observed sequence of length 2 was the token $\Delta$.

## 2 Viterbi (log-additive form)

*[20 total points]*




One HMM chain is shown on the left. The corresponding "factor graph" version is shown on the right. This simply shows the structure of the $A$ and $B_t$ log-prob tables and which variables they express preferences over. $A$ is the "transition" factor that has preferences for the two neighboring variables; for example, $A(y_1, y_2)$ shows how happy the model is with the transition from $y_1$ to $y_2$. The same transition preference function is used at all positions $(t-1, t)$ for each $t = 2..T$. $B_t$ is the "emission" factor that has preferences for the variable $y_t$. As a goodness function it is e.g. $B_1(y_1)$, $B_2(y_2)$, etc.

Let $\vec{y} = (y_1, y_2, ..., y_T)$, a proposed tag sequence for a $T$ length sentence. The total goodness function for a solution $\vec{y}$ is

$$G(\vec{y}) = \sum_{t=1}^{T} B_t(y_t) + \sum_{t=2}^{T} A(y_{t-1}, y_t)$$

**Question 2.1.** *[2 points]* Define $A$ and $B_t$ in terms of the HMM model, such that $G$ is the same thing as $\log p(\vec{y}, \vec{w})$ under the HMM.

**Question 2.2.** *[18 points]* Implement additive log-space Viterbi in "vit.py", by completing the *viterbi()* function. It takes in tables that represent the $A$ and $B$ functions as input. We give you an implementation of $G()$ you can check to make sure you understand the data structures, and also the exhaustive decoding algorithm too. Feel free to add debugging print statements as needed. The main code runs the exercise example by default.

When debugging, you should make new A and B examples that are very simple. This will test different code paths. Also you can try the *randomized_test()* from the starter code.

Look out for negative indexes as a bug. In python, if you use an index that's too high to be in the list, it throws an error. But it will silently accept a negative index ... it interprets that as indexing from the right.