COMP 182 HW 6 Problem 3

Yanjun Chen

Apr. 21^{th} 2017

1

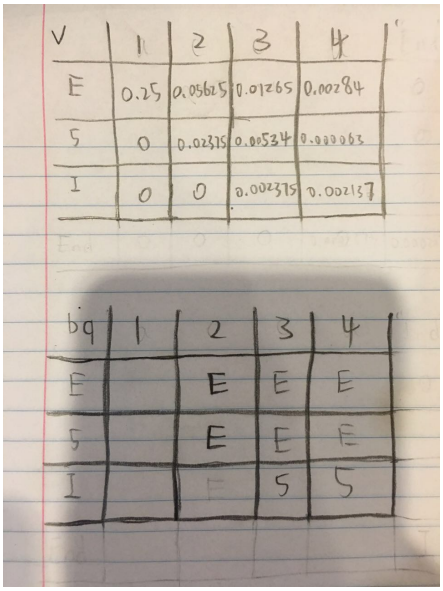
1.1

$P(acca|EE5I) = P(a|E) \times P(c|E) \times P(c|5) \times P(a|I) = 0.25 \times 0.25 \times 0.95 \times 0.4 = 0.02375$

1.2

$P(acca) = P(acca, EE5I) + P(acca, E5II) = P(acca|EE5I) \times P(EE5I) + P(acca|E5II) \times P(E5II) = P(acca|EE5I) \times P(start \to E) \times P(E \to E) \times P(E \to 5) \times P(5 \to I) \times P(I \to end) + P(acca|E5II) \times P(start \to E) \times P(E \to 5) \times P(5 \to I) \times P(I \to I) \times P(I \to end) = 0.02375 \times (1 \times 0.9 \times 0.1 \times 1 \times 0.1) + 0.0095 \times (1 \times 0.1 \times 1 \times 0.9 \times 0.1) = 0.00029925$

1.3



2

2.1

States: the set of nodes, $S = V = \{1, 2, 3, ..., n\}$

Initial state distribution: $\pi_l = \frac{1}{n}, (l \in S)$

Transition matrix: Suppose we have the edges E in the graph represented by an adjacency matrix M, in which $M[i, j] = 1$ if nodes i and j are neighbors and $M[i, j] = 0$ if they are not adjacent. Then, our transition matrix A will be:

$$A[i, j] = \frac{M[i, j]}{\sum_{l \in S} M[i, l]}, (i, j \in S)$$

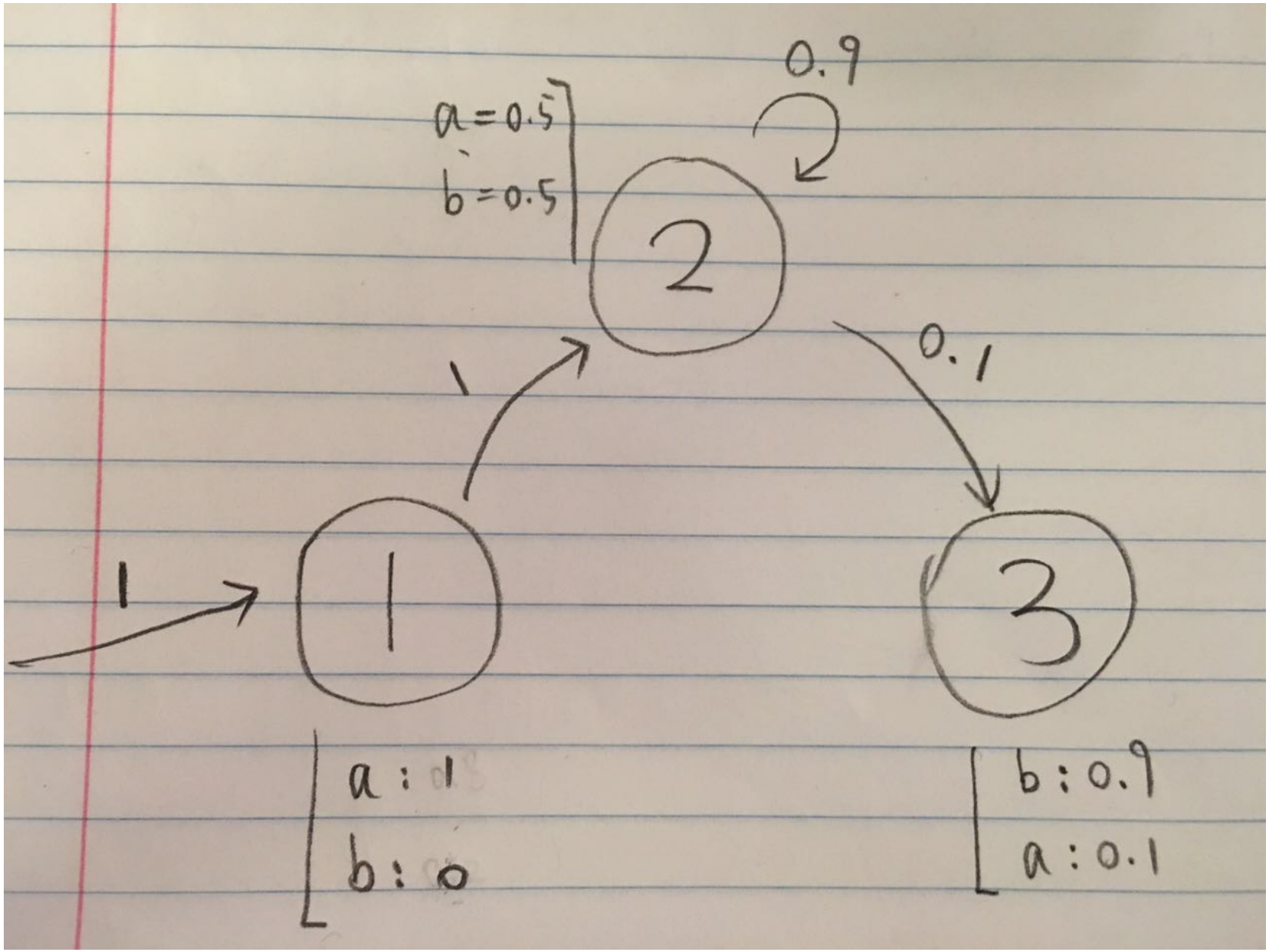
In order to solve the problem, we could use matrix multiplication to get the probability. Based on the definition, the probability that a particle whose location at time t_0 is node u would be at node v at time t_n is its entry in the matrix after we multiply it by itself n times: $A^n[u, v]$.

2.2

If we want to know the probability that a particle whose location at time t_0 is any of the nodes in the graph ends up at specific node v at time t_n, we have to take into account the initial probability distribution. We consider every node in the graph and calculate their probability of reaching node u at time t_n. We sum up those probability together to get the result: $\sum_{i \in S} \frac{1}{n} \times A^n[i, v] = \frac{1}{n} \sum_{i \in S} A^n[i, v]$

3

Because $P(X, Z) = P(X|Z) \times P(Z)$, maximizing $P(X|Z)$ does not take into account $P(Z)$, which may be very small (or even zero) given the largest $P(X|Z)$.
Consider the HMM in the following page:
We want to know the path that has the highest probability of emitting the sequence "abb". If we maximize $P(X, Z)$ as Viterbi algorithm did, we will get the path 122, whose probability is 0.225. However, if we only focus on maximizing $P(X|Z)$, we will scan over the emission probability and find out that state 1 has the highest probability of emitting a and state 3 has the highest probability of emitting b. Thus, we will get the path 133. However, state 1 has a zero probability to transit to state 3, $P(Z = 133) = 0$, so the total probability will be 0, which is evidently not the path with the highest probability.


0.9
a=0.5
b=0.5
2
1
0.1
1
1
3
a : 1
b : 0
b : 0.9
a : 0.1

4

Algorithm 1: SecondOrderViterbi

Input: A second-order HMM M with states $T = S \times S = \{(1,1),(1,2),...,(K,K)\}$ given by its transition matrix A, emission probability matrix E (alphabet $\sum$) and probability distribution π on the (initial) states; a sequence X of length L (indexed from 0 to $L-1$).

Output: A sequence Z, with $|Z| = |X|$, that maximizes $P(Z, X)$.

1 $v[(m,n),1] \leftarrow (\pi_{m,n} \cdot E_m(X_0) \cdot E_n(X_1))$ for every $(m,n) \in T$;
2 **for** $i \leftarrow 2$ ***to*** $L-1$ **do**
3 **foreach** $(m,n) \in T$ **do**
4 $v[(m,n),i] \leftarrow E_n(X_i) \cdot max_{(p,m)\in T}(v[(p,m),i-1] \cdot A[(p,m),(m,n)])$;
5 $bq[(m,n),i] \leftarrow argmax_{(p,m)\in T}(v[(p,m),i-1] \cdot A[(p,m),(m,n)])$;
6 $Z_{L-2}, Z_{L-1} \leftarrow argmax_{(m,n)\in T}\, v[(m,n),L-1]$;
7 **for** $i \leftarrow L-3$ ***downto*** *0* **do**
8 $Z_i \leftarrow$ the first element in tuple: $bq[(Z_{i+1}, Z_{i+2}), i+1]$;
9 **return** Z;

5

The running time for the second-order Viterbi algorithm is $O(mK^3)$

First of all, in the first line we create a matrix and initialize its values. Since the number of states is $|S| = K$, the number of second-order states is $|T| = |S \times S| = K^2$. We are creating a matrix with row representing the second-order states and column representing the position of the sequence, so the matrix is $(K^2 \times m)$, which takes us mK^2 operations.
Then, in the nested loop at line 2 and 3, the algorithm iterates over every entry in the matrix, so the loop itself takes mK^2 operations. Inside the loop, the algorithm tends to find the maximum value among all states that end with m (there are K states of them in total): $(p,m) \in T, p \in S$. Finding the maximum in K states take K operations, and so does finding the argmax of it. As a result, the algorithm from line 2 to line 5 takes $O(mk^2 \cdot K) = O(mk^3)$ operations.
In line 6, the algorithm finds the argmax among all states, which takes K^2 operations. In line 7, the for loop iterates m times. Line 8 takes $O(1)$ operation, so line 7 and line 8 takes in total $O(m)$ operations.

As a result, the running time for the second-order Viterbi algorithm is $O(mK^3)$