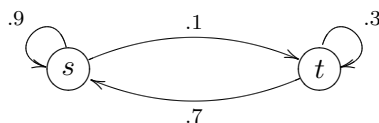


Q520: Homework on HMMs
 Due: Thursday, February 21

To make life simpler in this homework set, we are going to work with the example HMM from class:



Starting probability of s is $.4$, of t is $.6$.

In s , $\Pr(A) = .5$, $\Pr(B) = .3$, $\Pr(C) = .2$.

In t , $\Pr(A) = .1$, $\Pr(B) = .8$, $\Pr(C) = .1$.

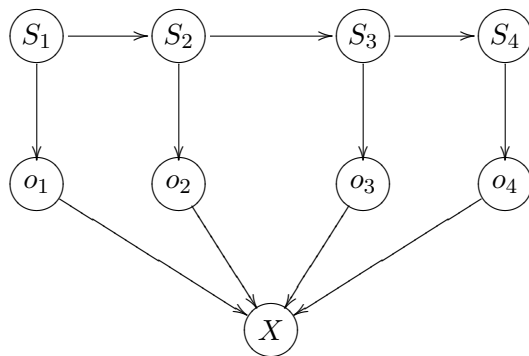
Problem 1: Practice With the Forward Algorithm

Using the example HMM from class, compute the probability of $ABCB$. Use the result from class on the probability of ABC and the next step of the Forward Algorithm.

[The idea here is to convince you that the Forward Algorithm is going to be more efficient than the naive algorithm of listing all the possible analyses. If we used the naive algorithm, then adding a letter would mean doubling the number of analyses and hence doubling the work. But using the Forward Algorithm, going from ABC to $ABCB$ requires only a constant number of arithmetic operations.]

Problem 2: HMMs and Bayesian Nets

Consider the following specification of a Bayesian net: The dag is



Note that what we have are random variables on the nodes. The upper-case S is supposed to remind us of *state*, and the o should call to mind *output*. (I would use an upper-case O , but it would easily be confused with the number zero.) The value spaces are $V(S_1) = \{s, t\}$, $V(S_2) = \{s, t\}$, $V(S_3) = \{s, t\}$, $V(S_4) = \{s, t\}$, $V(o_1) = \{A, B, C\}$, $V(o_2) = \{A, B, C\}$, $V(o_3) = \{A, B, C\}$, $V(o_4) = \{A, B, C\}$, and $V(X) = \{y, n\}$.

The conditional probability tables are given by

$$\Pr(S_1 = s) = .4, \text{ hence } \Pr(S_1 = t) = .6.$$

For $i = 1, 2, 3$, $\Pr(S_{i+1} = s | S_i = s) = .9$, $\Pr(S_{i+1} = t | S_i = s) = .7$.

For $i \leq 4$, $\Pr(o_i = A | S_i = s) = .5$, $\Pr(o_i = B | S_i = s) = .3$, and $\Pr(o_i = C | S_i = s) = .2$.

For $i \leq 4$, $\Pr(o_i = A | S_i = t) = .1$, $\Pr(o_i = B | S_i = t) = .8$, and $\Pr(o_i = C | S_i = t) = .1$.

$\Pr(X = y | o_1 = A, o_2 = B, o_3 = C, o_4 = B) = 1$;

in all other cases, $\Pr(X = y | o_1 = _, o_2 = _, o_3 = _, o_4 = _) = 0$.

Recall the difference between the specification of a Bayesian net and an actual net. The actual net always is based on a probability space \mathcal{S} and uses random variables over \mathcal{S} corresponding to the nodes of our dag; and the important thing is that the conditional independence assertions of the dag are satisfied by the random variables. A specification of a Bayesian net is the dag plus those tables, and we get an actual Bayesian net by the *tuple construction* that we saw in class and in the homework problems.

1. How big is \mathcal{S} ? [Hint: too big to work with by hand!]
2. Prove that the probability in \mathcal{S} of the event

$$(S_1 = s) \cap (o_1 = A) \cap (S_2 = t)$$

is the same as the probability that our HMM starts in s , outputs A , and then moves to t . The main point of this problem is to give a precise answer. So please quote carefully all the facts about Bayesian nets and HMMs that you use.

3. (Optional) State some other facts like the one in part (2). (The reason I say this is optional is I haven't stated it so precisely. The main part of this problem is the last one, just below. This part is a step that I used in doing the last problem, and I suspect that every correct solution to the last part will need to appeal to some fact of the kind that I'm looking for here.)
4. What is $\Pr(X = y)$ in \mathcal{S} ? [Hint: the number is worked out for you in problem 1. The point here is to say exactly *why* this is correct. You must use the conditional independence assertions in coming from the Bayesian net.]

[There are several reasons why I present this problem. One is to show something about the relation between HMMs and Bayesian nets. The problem suggests that the algorithm that Bayesian nets employ to go from a specification to a real net generalizes the Forward Algorithm of HMMs. (We didn't go over that Bayesian net algorithm in class, but our hint here is correct: it is another dynamic programming algorithm, and it's actually a bit more complicated notationally.) Another reason for this problem is to make the point that the relations between different mathematical models is a very worthwhile and interesting thing to pursue.]

Problem 3: The idea behind the Viterbi Algorithm

One of the things that we most want to do with HMMs is to take an (unanalyzed) output such as $AABCBCACBACB$ and find the state sequence s_1, \dots, s_{12} such that

$$Pr((s_1, A)(s_2, A)(s_3, B)(s_4, C) \cdots (s_{11}C)(s_{12}B)) \quad (1)$$

is as large as possible. The algorithm to do this efficiently is called the *Viterbi algorithm*, and this problem presents the central idea in a concrete way.

Suppose someone tells you two things:

Point 1: Of all the sequences of 12 states that end with s , the sequence that maximizes (1) is $sstttststss$, and the probability in (1) for this choice would be .0012.

Point 2: Of all the sequences of 12 states that end with t , the sequence that maximizes (1) is $tttststsstst$, and the probability in (1) for this choice would be .00049.

(I'm not saying that these two points are true, and in fact, I'm quite sure that they are not true (because the numbers are way too big). But please *pretend* that they are, because *some* facts like them *are* true.) The problem is to go one step further. Let's say we're interested in $AABCBCACBACBC$; this is the same as our string $AABCBCACBACB$ from before, except with one more C on the end. We naturally want to find the state sequence s_1, \dots, s_{13} such that

$$Pr((s_1, A)(s_2, A)(s_3, B)(s_4, C) \cdots (s_{11}C)(s_{12}B)(s_{13}C)) \quad (2)$$

is as large as possible. Actually, let's assume that Points 1 and 2 are correct, and use them to find two things: first, the sequence of 13 states ending in s that makes (2) as large as possible; second, the sequence of 13 states ending in t that makes (2) as large as possible.

How can we do this (pretending that Points 1 and 2 are correct)? Be sure to give a good reason.

[If you get stuck on this, you can look at the book, the web sites on HMMs linked to our class page, or lots of other sources. The one by R. D. Boyle will probably be most helpful. If you are comfortable with this problem is not terribly difficult.]

Problem 4: Deriving the Viterbi Algorithm

Give an algorithm which takes an unanalyzed string and gives the most likely analyzed string which has the given string as its sequence of outputs.

[Hint: the idea is to use the last problem and also the "dynamic programming" technique that we've seen in the Forward Algorithm. We want an algorithm that works along bigger and bigger pieces of the given string and also asks for more along the way than we are ultimately interested in. In our problem as in the Forward Algorithm, the extra information we need has something to do with the states in our HMM.]

Problem 5: Problems for our later analysis of the Baum-Welch Algorithm

Let h be an HMM, and let c be an unanalyzed corpus. Let c^* be the expected analyzed corpus of c on h .

The definition of the likelihood of c^* given h is

$$L(c^*, h) = \prod_{x \in X} \text{Pr}_h(x)^{c^*(x)}.$$

1. Show that

$$L(c^*, h) = \prod_{y \in Y} \prod_{x \in A(y)} \text{Pr}_h(x)^{c(y) \text{Pr}_h(x) / \text{Pr}_h(y)}.$$

2. Take a log to show

$$\log L(c^*, h) = \sum_{y \in Y} \sum_{x \in A(y)} \frac{c(y) \text{Pr}_h(x)}{\text{Pr}_h(y)} \log \text{Pr}_h(x).$$

3. Write a similar (and much shorter) expression for $\log L(c, h)$.

[Hint: you will want to recall the laws of logarithms, especially that the log of a product is the sum of the logs: $\log xy = \log x + \log y$.]