

# THIRD APPLICATION UNIT OF Q520: HIDDEN MARKOV MODELS

Larry Moss

Indiana University

Lecture notes for Q520

# HIDDEN MARKOV MODELS

These are some of the main tools used in various modeling areas. The topic comes from [speech recognition](#), and this is the source of the tutorial notes.

Incidentally, those notes are still the most readable general source, and are worthwhile even if you're not interested in speech recognition per se.

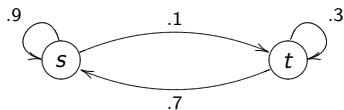
Similar to what we did for Bayesian Nets, the goals will be to

understand where all the numbers come from

understand what the basic formulas mean

understand what the major algorithms in the subject are doing

## AN EXAMPLE HIDDEN MARKOV MODEL



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

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

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

When we “run” this HMM, we produce two kinds of things:

★ **Unanalyzed output:** strings such as  $ABCBBBC$  or  $AABAC$ .

★ **Analyzed output:** strings such as

$(s, A)(t, B)(t, C)(t, B)(t, B)(s, B)(s, C)$ .

We also sometimes can write this one like this:

$A$	$B$	$C$	$B$	$B$	$B$	$C$
$s$	$t$	$t$	$t$	$t$	$s$	$s$

# THE DEFINITION

A **Hidden Markov Model (HMM)** consists of

- ★ A set  $S = \{s_1, \dots, s_K\}$  of *states*.
- ★ Starting probabilities  $\text{start}(s)$ . Again, the usual constraints that  $\sum_s \text{start}(s) = 1$  and each  $\text{start}(s) \in [0, 1]$ .
- ★ Probabilities  $\text{go}(s, t)$  of going from  $s$  to  $t$  in one step. We require that  $0 \leq \text{go}(s, t) \leq 1$ , and also

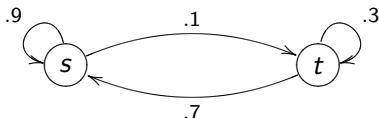
$$\sum_t \text{go}(s, t) = 1$$

for all states  $s$ .

An *output alphabet*  $\text{Alph} = \{A_1, \dots, A_L\}$  For each state  $s$  and each alphabet symbol  $A$ ,  $\text{out}(s, A)$ , the probability that in  $s$  we read out the symbol  $A$ . Again, the usual constraints.

We sometimes write the HMM as  $h = (S, \text{start}, \text{go}, \text{out})$ .

## AN EXAMPLE HIDDEN MARKOV MODEL



The formal details:

The set of states is  $S = \{s, t\}$ .

$\text{start}(s) = .4$  and  $\text{start}(t) = .6$ .

$\text{go}(s, s) = .9$  and  $\text{go}(s, t) = .1$ .

$\text{go}(t, s) = .7$  and  $\text{go}(t, t) = .3$ .

$\text{out}(s, A) = .5$ ,  $\text{out}(s, B) = .3$ ,  $\text{out}(s, C) = .2$ .

$\text{out}(t, A) = .1$ ,  $\text{out}(t, B) = .8$ ,  $\text{out}(t, C) = .1$ .

The model as a whole is  $h = (S, \text{start}, \text{go}, \text{out})$ .

## HOW DO WE STOP?

I've defined HMMs without explicit stop states.  
This will make life a little easier for what we do.  
It's possible to add stop states and other nifty features to HMMs,  
but the underlying mathematics is usually quite similar.  
We'll soon give examples of what it would mean to say that the  
probability of an output *AABA* is .00023.

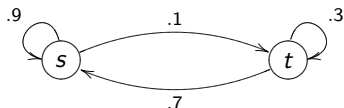
## ANALYZED AND UNANALYZED STRINGS

There usually are **lots of analyses** of an **unanalyzed output**.  
 In our example, the string *ABCBBBC* has  $2^7 = 128$  analyses.  
 Analyzed strings are easier to work with, as we'll see shortly.  
 But the data from the real world is unanalyzed. So there is work to do.

The typical data is a **corpus**: a set of possible outputs, allowing repetitions. For example:

<i>AABA</i>	<i>BBAB</i>
<i>AABAC</i>	<i>BBAB</i>
<i>ABCBBBC</i>	<i>BBAB</i>
<i>BABBA</i>	<i>ABCBBBC</i>
<i>CCAC</i>	<i>A</i>
<i>AABA</i>	<i>ABCBBBC</i>

# PROBABILITIES OF ANALYZED STRINGS



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

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

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

What is the probability of the output below?

$(s, A)(t, B)(t, C)(t, B)(t, B)(s, B)(s, C)$

It's easier to write it like this:

	$s$	$A$	$t$	$B$	$t$	$C$	$t$	$B$	$t$	$B$	$s$	$B$	$s$	$C$
	$.4$	$.5$	$.1$	$.8$	$.3$	$.1$	$.3$	$.8$	$.3$	$.8$	$.7$	$.3$	$.9$	$.2$

and so we get  $(.4)(.5)(.1)(.8)(.3)(.1)(.3)(.8)(.3)(.8)(.7)(.3)(.9)(.2)$ .

## SOME FACTS ON THESE PROBABILITIES

Here are two important facts to keep in mind:

For all  $n$ ,

$$\sum_{\substack{w \text{ is an unanalyzed string} \\ \text{of } n \text{ alphabet symbols}}} \Pr(w) = 1.$$

For example, if our alphabet is  $\{A, B\}$  without  $C$ ,  
 $\Pr(AA) + \Pr(AB) + \Pr(BA) + \Pr(BB) = 1.$

Second, for all  $n$ ,

$$\sum_{\substack{w \text{ is an analyzed string} \\ \text{of } n \text{ state-symbol pairs}}} \Pr(w) = 1.$$

## EVEN MORE

So the sum below is 1:

$$\begin{aligned}
 & \Pr((s, A)(s, A)) + \Pr((s, A)(t, A)) \\
 & + \Pr((t, A)(s, A)) + \Pr((t, A)(t, A)) \\
 & + \Pr((s, A)(s, B)) + \Pr((s, A)(t, B)) \\
 & + \Pr((t, A)(s, B)) + \Pr((t, A)(t, B)) \\
 & + \Pr((s, B)(s, A)) + \Pr((s, B)(t, A)) \\
 & + \Pr((t, B)(s, A)) + \Pr((t, B)(t, A)) \\
 & + \Pr((s, B)(s, B)) + \Pr((s, B)(t, B)) \\
 & + \Pr((t, B)(s, B)) + \Pr((t, B)(t, B))
 \end{aligned}$$

Again, we assume above that our alphabet is  $\{A, B\}$  without  $C$ .

For each unanalyzed word  $w$ , the probability of  $w$  is the sum of the analyses of  $w$ .

For example,  $\Pr(A) = \Pr((s, A)) + \Pr((t, A))$ .

## PROBABILITIES OF ANALYZED STRINGS

To get the probability of the *unanalyzed* string *ABCBBBC*, we *could*

- ★ Write all  $2^7 = 128$  analyses.
- ★ Then calculate the probabilities.
- ★ Then add these up.

The problem is that the length of *our input string* occurs as the *exponent* in the number of analyses.

So we have a problem requiring *exponential time*.

There are better ways to do get the probability we want.

By the way, how would we find the probability of an entire unanalyzed corpus?

# THE FORWARD ALGORITHM

To get the probability of the *unanalyzed* string  $ABC$ , we work in stages.

Stage 1 The probability of  $(s, A)$  is  $(.4)(.5) = .2$ .

For  $(t, A)$ , it's  $(.6)(.1) = .06$ .

Stage 2 The probability getting  $AB$  by some path or other **which happens to end up in state  $s$**  is

$$(.2)(.9)(.3) + (.06)(.7)(.3) = 0.0666.$$

For **which happens to end up in state  $t$** ,  
it's  $(.2)(.1)(.8) + (.06)(.3)(.8) = 0.0304$ .

# THE FORWARD ALGORITHM

To get the probability of the *unanalyzed* string  $ABC$ , we work in stages.

Stage 2 The probability getting  $AB$  by some path or other **which happens to end up in state  $s$**  is

$$(.2)(.9)(.3) + (.06)(.7)(.3) = 0.0666.$$

For **which happens to end up in state  $t$** ,  
it's  $(.2)(.1)(.8) + (.06)(.3)(.8) = 0.0304$ .

Stage 3 Getting  $ABC$  via a path ending in state  $s$ :

$$(.0666)(.9)(.2) + (.0304)(.7)(.2) = 0.016244.$$

Getting  $ABC$  via a path ending in state  $t$ :

$$(.0666)(.1)(.1) + (.0304)(.3)(.1) = 0.001578.$$

Grand total Probability of getting  $ABC$  is  $0.016244 + 0.001578 = 0.017822$ .

# DYNAMIC PROGRAMMING

The Forward Algorithm is an example of **dynamic programming**.

This refers to a style of algorithm with the following features:

★ The overall procedure consists of many small steps, each which uses the output of the last.

!!! The small steps give us **more than we need**.

The algorithm will run in polynomial time, and the naive algorithm (list everything and do calculations on each item) would be exponential.

# DYNAMIC PROGRAMMING

The Forward Algorithm is an example of **dynamic programming**.

This refers to a style of algorithm with the following features:

★ The overall procedure calculates consists of many small steps, each which uses the output of the last.

!!! The small steps give us **more than we need**.

In the Forward Algorithm, we end up knowing things like

$\Pr(A)$  ending at  $s$ .

$\Pr(AB)$  ending at  $t$ .

$\Pr(ABC)$  ending at  $t$ .

One would not guess at first that having all this extra information would make things faster.

# THE THREE MAIN PROBLEM FOR HMMs

First, given a HMM and an **unanalyzed string**  $s$ , what's the **probability that  $s$  was produced**?

For this, we use the Forward Algorithm that we studied last time.

Second, given a HMM and an **unanalyzed string**  $s$ , what's the **most likely sequence of states** that could have produced  $s$ ?

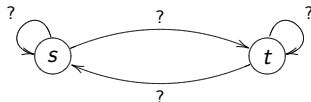
This one is similar, and you have the opportunity to work out the algorithm for yourself in the homework.

# THE THIRD (AND HARDEST) PROBLEM

Given a corpus, like

<i>AABA</i>	<i>BBAB</i>	<i>AABAC</i>	<i>BBAB</i>
<i>ABCBBBC</i>	<i>BBAB</i>	<i>BABBA</i>	<i>ABCBBBC</i>
<i>CCAC</i>	<i>A</i>	<i>AABA</i>	<i>ABCBBBC</i>

what numbers on the HMM maximize the corpus probability?



We need stuff like: starting probability of  $s$  is  $?$ , of  $t$  is  $?$ .

In  $s$ ,  $\Pr(A) = ?$ ,  $\Pr(B) = ?$ ,  $\Pr(C) = ?$ .

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

# LET'S DO IT FOR AN ANALYZED CORPUS FIRST

We need to fill in the blanks below, and we do it by gathering statistics.

Starting probability of  $s$  is \_\_, of  $t$  is \_\_.

In  $s$ ,  $\Pr(A) = \_$ ,  $\Pr(B) = \_$ ,  $\Pr(C) = \_$ .

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

## OUR FIRST WORD

<i>s</i>	<i>A</i>	<i>t</i>	<i>A</i>	<i>t</i>	<i>B</i>	<i>t</i>	<i>A</i>						
<i>s</i>	<i>B</i>	<i>s</i>	<i>B</i>	<i>t</i>	<i>A</i>	<i>t</i>	<i>B</i>						
<i>t</i>	<i>B</i>	<i>s</i>	<i>B</i>										
<i>s</i>	<i>A</i>	<i>t</i>	<i>B</i>	<i>t</i>	<i>C</i>	<i>s</i>	<i>B</i>	<i>t</i>	<i>B</i>	<i>s</i>	<i>B</i>	<i>t</i>	<i>C</i>

For the first analyzed string in our corpus of three:



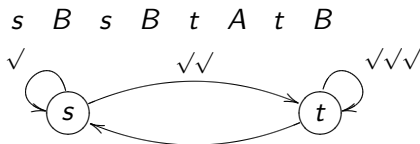
Starting probability of  $s$  is  $\frac{1}{2}$ , of  $t$  is  $\frac{1}{2}$ .

In  $s$ ,  $\Pr(A) = \frac{1}{2}$ ,  $\Pr(B) = \frac{1}{2}$ ,  $\Pr(C) = 0$ .

In  $t$ ,  $\Pr(A) = \frac{1}{4}$ ,  $\Pr(B) = \frac{1}{2}$ ,  $\Pr(C) = \frac{1}{4}$ .

# NOW THE SECOND WORD

Now we add the second string in our corpus:



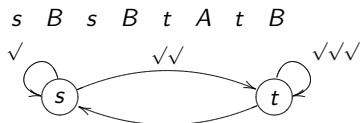
Starting probability of  $s$  is  $\sqrt{\sqrt{}}$ , of  $t$  is  $\_\_\_$ .

In  $s$ ,  $\Pr(A) = \sqrt{}$ ,  $\Pr(B) = \sqrt{\sqrt{}}$ ,  $\Pr(C) = \_\_\_$ .

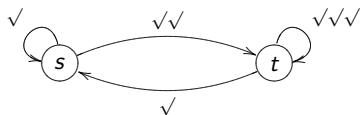
In  $t$ ,  $\Pr(A) = \sqrt{\sqrt{\sqrt{}}}$ ,  $\Pr(B) = \sqrt{\sqrt{}}$ ,  $\Pr(C) = \_\_\_$ .

# THE THIRD WORD IN THE CORPUS

We have



plus other stuff not shown. Now let's add the third line  $t B s B$ .



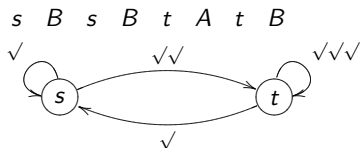
Starting probability of  $s$  is  $\sqrt\sqrt$ , of  $t$  is  $\sqrt$ .

In  $s$ ,  $\Pr(A) = \sqrt$ ,  $\Pr(B) = \sqrt\sqrt\sqrt$ ,  $\Pr(C) = \_$ .

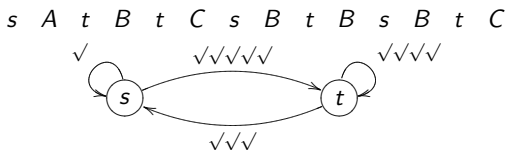
In  $t$ ,  $\Pr(A) = \sqrt\sqrt\sqrt$ ,  $\Pr(B) = \sqrt\sqrt\sqrt$ ,  $\Pr(C) = \_$ .

## ESTIMATING THE ANALYZED CORPUS, CONTINUED

We have



plus other probabilities not shown. We finally add

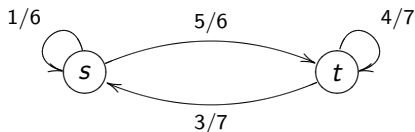


Starting probability of  $s$  is  $\sqrt{\sqrt{\sqrt{\phantom{x}}}}$ , of  $t$  is  $\sqrt{\phantom{x}}$ .

In  $s$ ,  $\Pr(A) = \sqrt{\sqrt{\phantom{x}}}$ ,  $\Pr(B) = \sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\phantom{x}}}}}}$ ,  $\Pr(C) = \underline{\quad}$ .

In  $t$ ,  $\Pr(A) = \sqrt{\sqrt{\sqrt{\phantom{x}}}}$ ,  $\Pr(B) = \sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\phantom{x}}}}}}$ ,  $\Pr(C) = \sqrt{\sqrt{\phantom{x}}}$ .

# CONVERT TO NUMBERS



Starting probability of  $s$  is  $3/4$ , of  $t$  is  $1/4$ .

In  $s$ ,  $\Pr(A) = 2/7$ ,  $\Pr(B) = 5/7$ ,  $\Pr(C) = 0$ .

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

This is the HMM which maximizes the probability of the original analyzed corpus.

Of course this screams out to be justified.

But once again, doing all this for **unanalyzed** corpora is our main goal!

# THE REASONING

We just presented a procedure to **estimate the parameters** of an analyzed corpus.

At this point in the course, we don't quite have the tools to say **why** the procedure maximizes the probability of the corpus.  
But we're getting there!

## THE REASONING, CONTINUED

Recall that our HMM  $h$  is determined by 12 numbers

$$\begin{aligned} & \text{start}(s), \text{start}(t), \text{go}(s, s), \text{go}(s, t), \text{go}(t, s), \text{go}(t, t), \\ & \text{out}(s, A), \text{out}(s, B), \text{out}(s, C), \text{out}(t, A), \text{out}(t, B), \text{out}(t, C). \end{aligned}$$

We don't know them, but we know certain sums are 1.

(For example,  $\text{go}(s, t) + \text{go}(s, s) = 1$ .)

And we want to maximize the likelihood of an analyzed corpus  $c$ .

What all the  $\sqrt{\phantom{x}}$ s show is that the likelihood of  $c$  is the big product

$$\begin{aligned} & \text{start}(s)^3 \text{start}(t)^1 \text{go}(s, s)^1 \text{go}(s, t)^5 \text{go}(t, s)^4 \text{go}(t, t)^3 \\ & \text{out}(s, A)^2 \text{out}(s, B)^5 \text{out}(s, C)^0 \text{out}(t, A)^3 \text{out}(t, B)^5 \text{out}(t, C)^2 \end{aligned}$$

We'll see later in the course that the values which maximize our likelihood and still sum to 1 in the appropriate ways are exactly the

**relative frequencies:**  $\text{start}(s) = 3/4$ ,  $\text{start}(t) = 1/4$ , etc.

# A SLOGAN

The relative frequency gives a maximum likelihood, provided the relative frequency is a probability distribution that your overall model is happy with.

A **model** in these discussions is a set of probability functions. In the HMM case, we have fixed sets  $S$  of states and  $Alph$  of output symbols.

Our model is then the set of all triples  $h = (\text{start}, \text{go}, \text{out})$  as in our definition of HMMs.

# PARAMETER ESTIMATION FOR UNANALYZED CORPORA

This is the most important problem for applications of HMMs.

In a real-world problem, how do we get all the numbers?

In general, we want to select a model which *maximizes* the likelihood that we see our data.

In the HMM case, this is one of those problems which is believed to be intractable (it is NP complete).

So the best one hopes for is either a good approximate answer, or to find a *local max* which might not be the absolute max.

The Baum-Welch Algorithm does the second thing. Beginning with any HMM  $h = (\text{start}, \text{go}, \text{out})$ , it *re-estimates* this to some other HMM  $h'$ .

We re-apply the algorithm and we get  $h_2$ , etc.

The process will head to some local maximum, but it might not be the global max. We might try beginning somewhere else and see if things improve.

# IDEAS BEHIND THE BAUM-WELCH ALGORITHM

A *corpus* was defined before as a set of unanalyzed words, allowing repetition.

It will be convenient to re-package this. Henceforth, a corpus will be a function

$$c : Y \rightarrow R^+$$

where  $Y$  is a finite set of unanalyzed words, and  $R^+$  is the positive reals. Usually we think of having  $c(y)$  to be a whole number.

But allowing any real number will make some things easier.

Then the **likelihood** of  $c$  on the HMM  $h = (\text{start}, \text{go}, \text{out})$  is given by

$$L(c, h) = \prod_{y \in Y} Pr_h(y)^{c(y)}.$$

Here  $Pr_h(y)$  is the probability of  $y$ , as we calculated in last time.

**Important:** Note that  $h$  is involved in all of this work!

# Y AND X

Y is our set of **unanalyzed** strings, and we have  $c : Y \rightarrow R^+$ .  
 Let X be the set of **analyzed** strings:

$$\begin{array}{ccc} Y & \xrightarrow{c} & R^+ \\ \uparrow \text{yield} & & \\ X & & \end{array}$$

Later on I will return to think of these as **random variables**  
 on the space of **infinite runs produced by the HMM**

## THE EXPECTED ANALYZED CORPUS

$Y$  is our set of **unanalyzed** strings, and we have  $c : Y \rightarrow R^+$ .

Let  $X$  be the set of **analyzed** strings  $x$  whose “yield”  $yield(x)$  belongs to  $Y$ .

The main thing is to take our unanalyzed corpus  $c$  and generate an **analyzed corpus**  $c^* : X \rightarrow R^+$ .

For each  $x \in X$ , we already know how to calculate  $Pr_h(x)$ .

In fact, for all  $y \in Y$ ,

$$Pr_h(y) = \sum_{\substack{x \in X \\ yield(x) = y}} Pr_h(x).$$

(Note that I’m being a little sloppy, since we are really dealing with *two* different probability spaces: the unanalyzed strings and the analyzed ones.)

## THE EXPECTED ANALYZED CORPUS, CONTINUED

For each  $y \in Y$ , let  $A(y) = \{x \in X : \text{yield}(x) = y\}$ .

$A(y)$  is the set of analyzed words whose yield is  $y$ .

We want to make each  $A(y)$  into a probability space, and so we define

$$Pr_{A(y)}(x) = Pr_h(x) / Pr_h(y).$$

We define  $c^*$ , the **expected analyzed corpus of  $c$  on  $h$** , by:

$$c^*(x) = c(\text{yield}(x)) Pr_{A(\text{yield}(x))}(x).$$

So if  $y = \text{yield}(x)$ , then  $c^*(x) = c(y) Pr_h(x) / Pr_h(y)$ .

Key point:  $c^*$  is an analyzed corpus, so we can find the maximally likely model  $h' = (\text{start}', \text{go}', \text{out}')$  for it.

*We know how to do this!*

(And we cannot assume that  $c^*$  will always give integer values.)

# AN EXAMPLE, TO SET THE IDEAS



Let's say that  $x_1$  is an analyzed string corresponding to the reading

$$(\text{Navy seals})_{NP} \text{report}_V$$

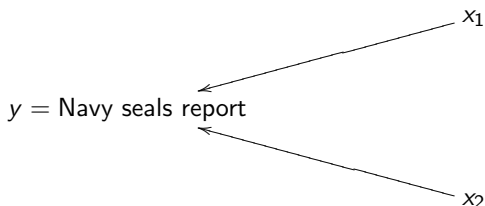
and  $x_2$  is an analyzed string corresponding to

$$\text{Navy}_{NP} (\text{seals}_V \text{report}_{NP})_{VP}$$

None of the grammatical facts will matter here.

Note also that the yield of both  $x_1$  and  $x_2$  is  $y$ .

# AN EXAMPLE, TO SET THE IDEAS



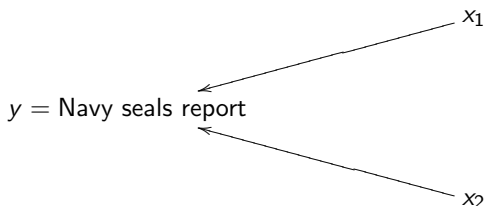
Let's say that  $c(y) = 100$ ,  $\Pr_h(x_1) = .0001$ , and  $\Pr_h(x_2) = .00001$ .

We could have gotten  $c(y) = 100$  from some actual corpus.

And we could have gotten the probabilities of the analyzed words  $x_1$  and  $x_2$  from *any* hmm  $h$ .

What we're doing here is to show how to *improve*  $h$ .

## AN EXAMPLE, TO SET THE IDEAS



Let's say that  $c(y) = 100$ ,  $\Pr_h(x_1) = .0001$ , and  $\Pr_h(x_2) = .00001$ .

We also would have  $\Pr_h(y) = .00011$ .

The arrows are the yield function, and  $A(y) = \{x_1, x_2\}$ .

We turn  $A(y)$  into a probability space by

$$\Pr_{A(y)}(x_1) = .0001/.00011 = .909.$$

$$\Pr_{A(y)}(x_2) = .00001/.00011 = .091.$$

Then the expected corpus  $c^*$  is given by

$$c^*(x_1) = 100 \times .909 = 90.9, \text{ and } c^*(x_2) = 100 \times .091 = 9.01.$$

# THE BAUM-WELCH ALGORITHM IN ABSTRACT FORM

We have fixed the set of states and output alphabet. We also have an unanalyzed corpus  $c$ . We want to find  $h$  to maximize  $L(c, h)$ . Given an unanalyzed corpus  $c$ , we compute iteratively as follows:

- 1 Let  $h$  be any HMM on our state and alphabet.
- 2 Let  $c_1$  be the expected analyzed corpus obtained by using  $h$  on the original corpus  $c$ .
- 3 Let  $h_1$  be the maximally likely HMM for  $c_1$ .
- 4 Let  $c_2$  be the expected analyzed corpus obtained by using  $h_1$  on the original corpus  $c$ .
- 5 Let  $h_2$  be the maximally likely HMM for  $c_2$ .
- 6 Let  $c_3$  be the expected analyzed corpus obtained by using  $h_2$  on the original corpus  $c$ .
- 7 ...

## MORE ON THE ABSTRACT FORM

There is no stopping: the algorithm is simply run until it looks like  $h_{n+1} = h_n$ .

What one can show is that

$$L(c, h) \leq L(c, h_1) \leq L(c, h_2) \leq \dots$$

and that the sequence of  $h$ 's will converge.

(Indeed, we will do this later this semester *unlike most sources*.)

But one cannot be sure that the value that you get is a *global maximum*; it could be some other local max.

In practice, one often has a good idea where to start, and this helps.

Or, you can start at many different points independently, and then take the largest value that you get.

# THE ABSTRACT FORM IS ESSENTIALLY THE EM ALGORITHM

The EM algorithm is a general algorithm for computing a maximum likelihood estimate of some model class.

It is much more general than our HMM work; that is, the abstract steps that we did had nothing to do with HMMs.

The only thing we needed to know was that for **analyzed** corpora, **the relative frequency estimate gives a maximum likelihood**.

The special case of this EM algorithm for HMMs is called the *Baum-Welch Algorithm* or the *Forward-Backward Algorithm*. It was discovered in the 1960's, and the more general EM algorithm was discovered in different settings a short time later.

A good discussion of it in the discrete setting may be found in the notes of Detlef Prescher.

# A PROBLEM WITH THE ABSTRACT ALGORITHM

The main problem with the abstract algorithm is that even if  $Y$  has just one word,  $X$  might be too large to write down (if there are many states).

But we don't really need to write  $X$  at all.

We just need to be able to go from our unanalyzed corpus  $Y$  to the  $\sqrt{\quad}$  statistics for  $X$ .

The idea is to do this in one fell swoop, without writing  $X$ , using the **dynamic programming** technique that we have already seen in the Forward Algorithm.

## BAUM-WELCH ALGORITHM, CONTINUED

Let  $y = A_1A_2 \cdots A_n$ .

For  $1 \leq j \leq n$  and  $s$  one of our states, we define  $\alpha(y, j, s)$  to be the probability in the space of analyzed words of reading out  $A_1A_2 \cdots A_j$  via some path of states of (length  $j$ ) which happens to end in state  $s$ .

This is related to the computations in the Forward Algorithm.

$$\alpha(y, 1, s) = \text{start}(s)\text{out}(s, A_1)$$

$$\alpha(y, j + 1, s) = \sum_{t \in S} \alpha(y, j, t)\text{go}(t, s)\text{out}(s, A_{j+1}).$$

The lines above give a *recursion* which can be implemented efficiently.

# BAUM-WELCH ALGORITHM, CONTINUED

Again, we write out  $y$  as  $A_1A_2 \cdots A_n$ .

We also need *Backward probabilities*.

Define  $\beta(y, j, s)$  to be the following conditional probability:

Given that the  $j$ th state is  $s$ , the  $(j + 1)$ st symbol will be  $A_{j+1}$ , the  $(j + 2)$ nd will be  $A_{j+2}$ ,  $\dots$ , the  $n$ th will be  $A_n$ .

For this, our equations go backward:

$$\beta(y, n, s) = 1$$

$$\beta(y, j, s) = \sum_{t \in S} \text{go}(s, t) \text{out}(t, A_{j+1}) \beta(y, j + 1, t).$$

It is a good exercise to convince yourself that these equations define the function that we described!

# BAUM-WELCH ALGORITHM

Recall that  $A(y)$  is the set of analyzed words with  $y$  as their yield.  
 We want the probability on  $A(y)$  that an analyzed sequence has  $s$  as its  $j$ th state, ( $A_{j+1}$  as its  $(j + 1)$ st symbol), and  $t$  as its  $(j + 1)$ st state.  
 This is called  $\gamma(y, j, s, t)$ .

$$\gamma(y, j, s, t) = \frac{\alpha(y, j, s) \text{go}(s, t) \text{out}(t, A_{j+1}) \beta(y, j + 1, t)}{\text{Pr}_h(y)}.$$

One more function: the probability on  $A(y)$  that an analyzed sequence has  $s$  as its  $j$ th state:

$$\delta(y, j, s) = \sum_t \gamma(y, j, s, t).$$

This only works for  $j < n = \text{length}(y)$ .

For  $j = \text{length}(y)$ , we need a slightly different function. What is it?

## READING OFF THE STATISTICS ABOUT $c^*$

Recall that we start with a corpus  $c : Y \rightarrow R^+$  of unanalyzed words.

We want to read off statistics about the expected analyzed corpus  $c^*$ , and then we would continue with the relative frequency calculations for  $c^*$ .

We have  $\gamma(y, j, s, t)$  and  $\delta(y, j, s)$  functions for each  $y \in Y$ .

With  $\gamma(y, j, s, t)$  and  $\delta(y, j, s)$ , we can read off the statistics we need about  $c^*$  **without ever writing down  $X$** .

For example, the starting state  $\surd$  number for state  $s$  will be

$$\sum_{y \in Y} c(y) \sum_{s \in S} \delta(y, 1, s).$$

## ABOUT $c^*$ , CONTINUED

Another example:

the  $\checkmark$  number for going from state  $s$  to state  $t$  will be

$$\sum_{y \in Y} c(y) \sum_{j \leq n} \gamma(y, j, s, t).$$

Finally, the  $\checkmark$  number for outputs of  $B$  from  $s$  will be

$$\sum_{y \in Y} c(y) \sum_{\substack{j \leq n \\ y_j = B}} \delta(y, j, s).$$

This takes some thinking, and the example I pass out should help.

# COMMENTS ON THE HOMEWORK PROBLEMS

## 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.

This is a straightforward problem.

You need to show that you know how the Forward Algorithm works by executing one more step of it, on top of the three steps we did in class.

For an animation of the algorithm, see the link from our course web page.

# COMMENTS ON THE HOMEWORK PROBLEMS

Problem 3 will be harder for most people, so let me continue with an easier one.

Problem 3: The idea behind the Viterbi Algorithm

Given the unanalyzed output *AABCBCACBACB*, which sequence of states  $s_1, \dots, s_{12}$  maximizes

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

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 *ssttttststss*, 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 *tttstststst*, and the probability in (1) for this choice would be .00049.

(I'm not saying that these two points are true, but please pretend that they are.)

## PROBLEM 3, CONTINUED

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.

How can we do this (pretending that Points 1 and 2 are correct)?

Be sure to give a good reason.

The reason can be a “common sense” mathematical point, but it could also involve inequalities.

## PROBLEM 4: MAKE PROB. 3 INTO AN ALGORITHM

The idea here is for you to come up with an algorithm which, when presented with an unanalyzed string like *CCBAABCAB* returns the most likely string.

Your algorithm must not ask us to list all the possible analyses and then take the most probable one.

That would be impractical for long strings!

Instead, you want to adapt the idea of the Forward Algorithm with the observation of Problem 3.

## PROBLEM 5

This is a math point related to the Baum-Welch Algorithm.  
We saw the formula

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

The  $x$ 's are analyzed words,

$A(y)$  is the set of all possible analyses of the word  $y$ .

We want to show in part (a) that this product in fact equals

$$\prod_{y \in Y} \prod_{x \in A(y)} Pr_h(x)^{c(y) Pr_h(x) / Pr_h(y)}.$$

Hint The elements of  $X$  fall into groups together, with the groups corresponding to the analyses of the same  $y \in Y$ .

## PROBLEM 5

The rest of the problem is mainly an exercise in algebraic manipulation with logarithms and products.

The reason for doing it is that the results of this problem are used in understanding the ideas behind the Baum-Welch algorithm, especially in showing that as we run it, the likelihoods go up.

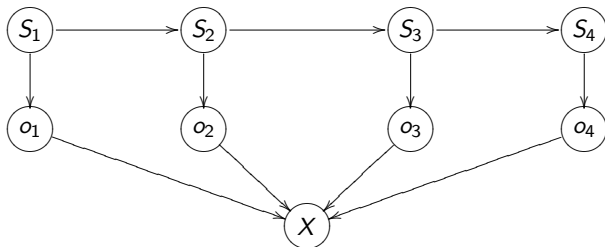
## PROBLEM 2: BAYESIAN NETS AND HMMs

The idea here is to use Bayesian Nets to evaluate HMM probabilities.

We are concerned with our same two-state HMM, and with the word *ABCB*.

You found the probability of this word in Problem 1.

## PROBLEM 2: BAYESIAN NETS AND HMMs



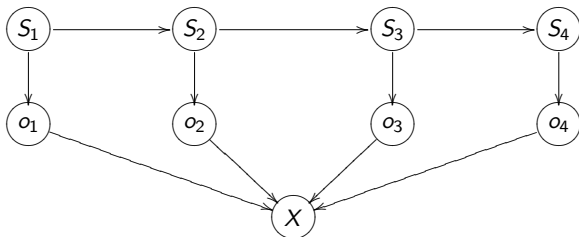
$$V(S_1) = \{s, t\},$$

$$V(S_2) = \{s, t\}, \quad V(S_3) = \{s, t\}, \quad V(S_4) = \{s, t\},$$

$$V(o_1) = \{A, B, C\}, \quad V(o_2) = \{A, B, C\},$$

$$V(o_3) = \{A, B, C\}, \quad V(o_4) = \{A, B, C\}, \quad \text{and} \quad V(X) = \{y, n\}.$$

## PROBLEM 2: BAYESIAN NETS AND HMMs



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

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

$\Pr(S_{i+1} = s | S_i = t) = .7$ .

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,

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

## PROBLEM 2: BAYESIAN NETS AND HMMs

We can think of the underlying space as either

(1) the space of all real-life runs of the HMM.

(2) the artificially-constructed space of tuples; in this case, they would be 9-tuples.

Your overall job: evaluate  $Pr(X = y)$  in the tuple setting.

This is the last part of the problem, and the previous parts lead you to this.

Be sure to use those parts.

You also should use facts about Bayesian nets, especially the one that says that

in a Bayesian Net specified by tables and constructed out of tuples, the probability assertions of the original tables automatically are true in the tuple space.