CS 683: Advanced Design and Analysis of Algorithms

Lecture 23, March 12, 2008
Lecturer: John Hopcroft
Scribe: Ethan Feldman

March 24, 2008

1 Note from last class

When building the table to get the most likely output sequence, we don't need to keep track of the start state, so we only need the probability of being in each state at each time t.

Example

	A	B
$t = 3$		
$t = 2$		
$t = 1$		
$t = 0$	$1 \cdot 1/2$	0

So runtime $= n^2T$ where
n - number of states
T - length of sequence

2 Hidden Markov Models

Given
a sequence of output symbols $O_0, O_1, ..., O_T$
an upper bound on number of states,

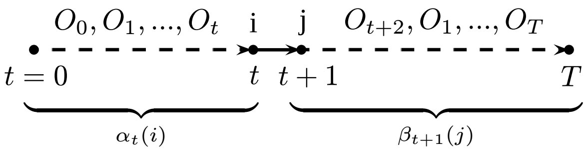
what is the most likely Hidden Markov Model to have produced this output?

2.1 Algorithm

Let A_{ij} be the probability of going to state j from state i
$b_i(O_k)$ be the probability of outputting O_k in state i

Assume some initial values for A_{ij} and b_i
Upgrade the quality of the approximation until some value is reached.

Let $\delta_t(i, j)$ be the probability of going from state i to state j at time t.



Let $\alpha_t(i)$ be the probability of seeing $O_1, O_2, ..., O_t$ ending in state i.
Let $\beta_{t+1}(j)$ be the probability of seeing output symbols $O_{t+2}, O_{t+3}, ..., O_T$ starting in state j.

$$\delta_t(i,j) = \frac{\overbrace{\alpha_t(i)}^{\text{in state } i} \cdot \overbrace{A_{ij}}^{\text{go from } i \to j} \cdot \overbrace{b_j(O_{t+1})}^{\text{output } O_{t+1}} \cdot \overbrace{\beta_{t+1}(j)}^{\text{end in state } j}}{\underbrace{P(O)}_{\text{probability of given output}}}$$

$$P(O) = \sum_i \sum_j \alpha_t(t) \cdot A_{ij} \cdot b_j(O_{t+1}) \cdot \beta_{t+1}(j)$$

Let $s_t(i)$ be the probability of being in state i at time t.

$$s_t(i) = \sum_j \delta_t(i,j)$$

Estimating A_{ij}:

$$\begin{aligned} A_{ij} &= \frac{\text{number of times we go from } i \to j}{\text{number of times in state } i} \\ &= \frac{\sum_{t=1}^{T-1} \delta_t(i,j)}{\sum_{t=1}^{T-1} s_t(i)} \end{aligned}$$

$$\begin{aligned} b_j(O_{(k)}) &= \frac{\text{expected number of times in state } j \text{ and output } O_{(k)}}{\text{expected number of times in state } j} \\ &= \frac{\sum_{t \text{ when } O_t = O_{(k)}} s_t(j)}{\sum_{t=1}^{T-1} s_t(j)} \end{aligned}$$

Note: $O_{(k)}$ is the k^{th} symbol of the alphabet.

2.2 Questions

1. Why should new values be better?
2. Does the process always converge?

2.3 Example used to code the algorithm

A_{ij}:

	1	2	3
1	1/2	1/4	1/4
2	1/4	1/4	1/2
3	1/3	1/3	1/3

$b_j(O_{(k)})$:

	A	B
1	3/4	1/4
2	1/4	3/4
3	1/3	2/3

Notes:

1. Store logarithms of the probabilities to avoid precision errors.
2. This type of algorithm is known in the literature as a message passing algorithm.

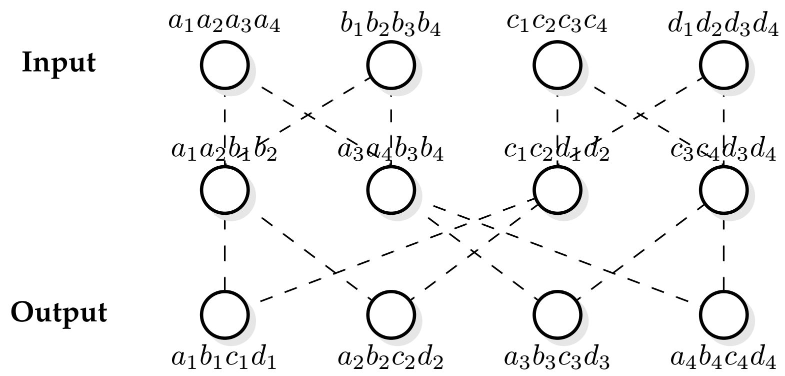
☆ Project: Try programming this algorithm to see how well it works.

3 Random Walks

How far do we have to walk to say we are at a random location? (forgotten everything about our start node)

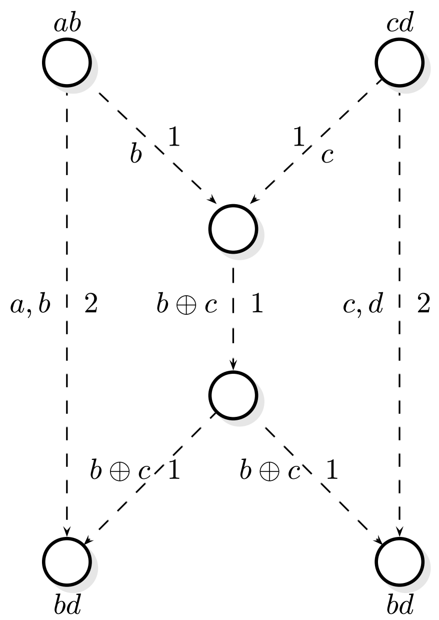
3.1 Matrix Transpose

Matrix stored by rows, want stored by columns

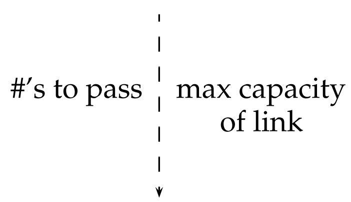


$O(n \log n)$ operations.
From subset of input to subset of output, need a min-cut in the inner structure to allow the information to flow.

3.2 Another Example



It looks like you need to pass both b & c through the middle of the link, but it's possible by using $\oplus$.



3.3 Expanders

A graph $G = (V, E)$ is an expander if for every subset $S \subseteq V$, $|S| \leq \frac{n}{2}$, there are at least $c|S|$ vertices outside of S connected to S.

1. When will a random graph will be an expander with probability 1?

   Ex. Random d-regular graph is an expander.

   There will be $d|S|$ edges, $d|S|\frac{n-|S|}{n}$ go to vertices outside of S.
   need to show: $d\frac{n-|S|}{n} \geq c$.

$$\begin{aligned} d\frac{n-|S|}{n} &\geq d\frac{n-\frac{n}{2}}{n} \quad \left[|S| \leq \frac{n}{2}\right] \\ &\geq \frac{d}{2} \\ &\geq c \end{aligned}$$

2. Is there a deterministic way to create an expander?