Statistical Sequence Recognition and Training: An Introduction to HMMs

EECS 225D

Nikki Mirghafori
nikki@icsi.berkeley.edu

March 7, 2005

Credit: many of the HMM slides have been borrowed and adapted, with permission, from Ellen Eide and Lalit Bahl at IBM, developed for the Speech Recognition Graduate Course at Columbia.

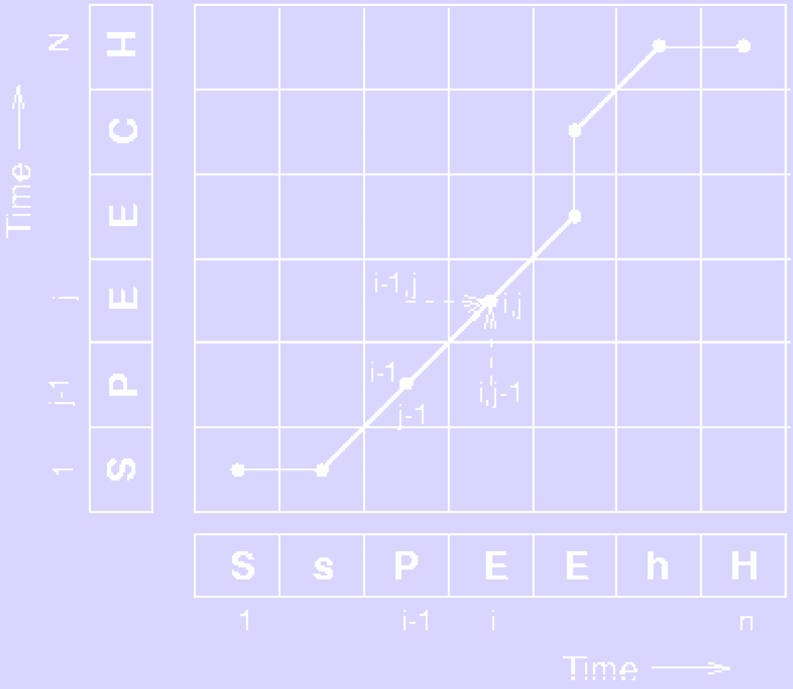
Overview

- Limitations of DTW (Dynamic Time Warping)
- The speech recognition problem
- Introduction to Hidden Markov Models (HMMs)
- Forward algorithm (a.k.a. alpha recursion) for Estimation of HMM probabilities
- Viterbi algorithm for Decoding (if time)

Recall DTW (Dynamic Time Warping) from Last Time

- Main idea of DTW:

  Find minimum distance between a given word and template, allowing for stretch and compression in the alignment



Beyond DTW

- Some limitations of DTW:
  - Requires end-point detection, which is error-prone
  - Is difficult to show the effect on global error
  - Requires templates (examples); using canonicals is better
- We need a way to represent
  - Dependencies of each sound/word on neighboring context
    - Continuous speech is more than concatenation of elements
  - Variability in the speech sample
- Statistical framework allows for the above, and
  - Provides powerful tools for density estimation, training data alignment, silence detection -- in general, for training and recognition

Markov Models

- Brief history:

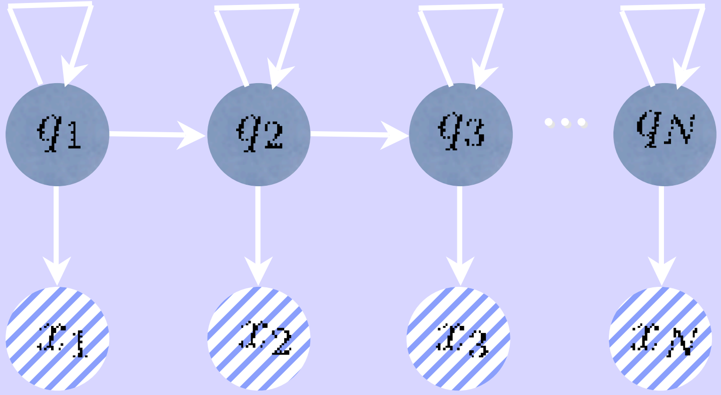
  Introduced by Baum et al. in 60’s, 70’s

  Applied to speech by Baker in the original CMU Dragon System (1974)

  Developed by IBM (Baker, Jelinek, Bahl, Mercer,….) (1970-1993)

  Took over ASR (automatic speech recog.) in 80’s
- Finite state automoton with stochastic transitions

A generative model: the states have outputs (a.k.a. observation feature vectors). Q’s are states and X’s are the observations.



The statistical approach to speech recognition

- W is a sequence of words, w1, w2, …, wN
- W* is the best sequence.
- X is a sequence of acoustic features: x1, x2, …., xT
- Θ is a set of model parameters.

$$W^* = \underset{W}{\arg\max}\, P(W \mid X, \Theta)$$

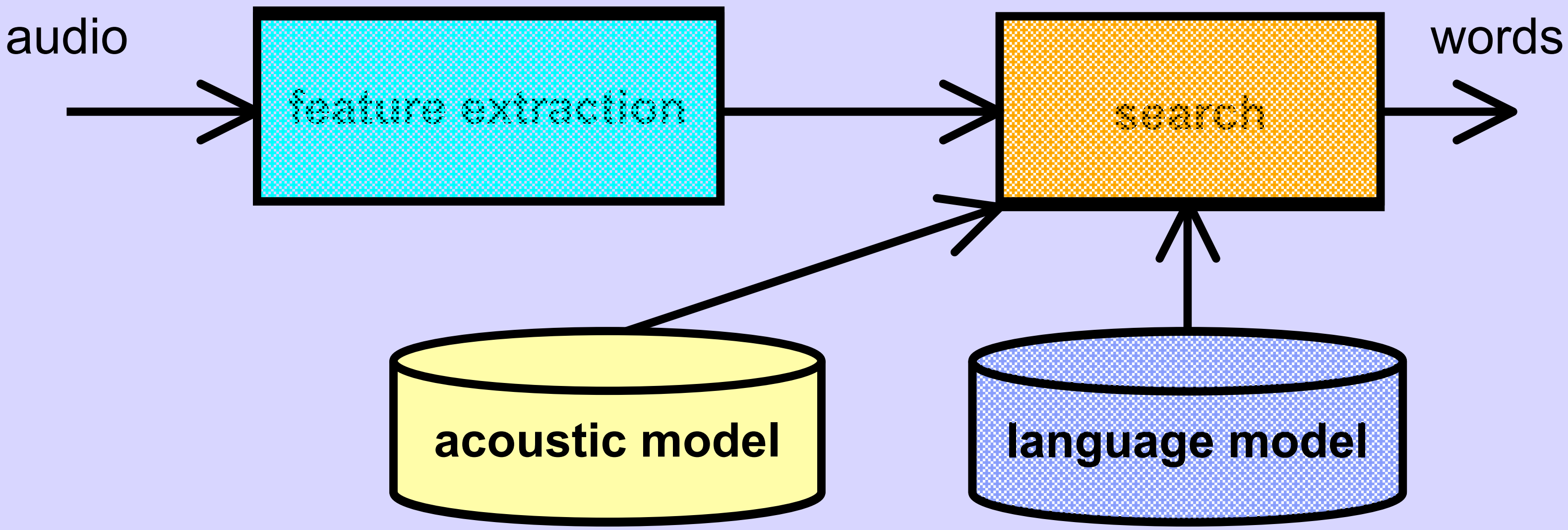
$$= \underset{W}{\arg\max}\, \frac{P(X \mid W, \Theta) P(W \mid \Theta)}{P(X)} \qquad \text{Bayes' rule}$$

$$= \underset{W}{\arg\max}\, P(X \mid W, \Theta) P(W \mid \Theta) \qquad \text{P(X) doesn't depend on W}$$

Bayes’ rule reminder:

$$P(A|B) = \frac{P(A,B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

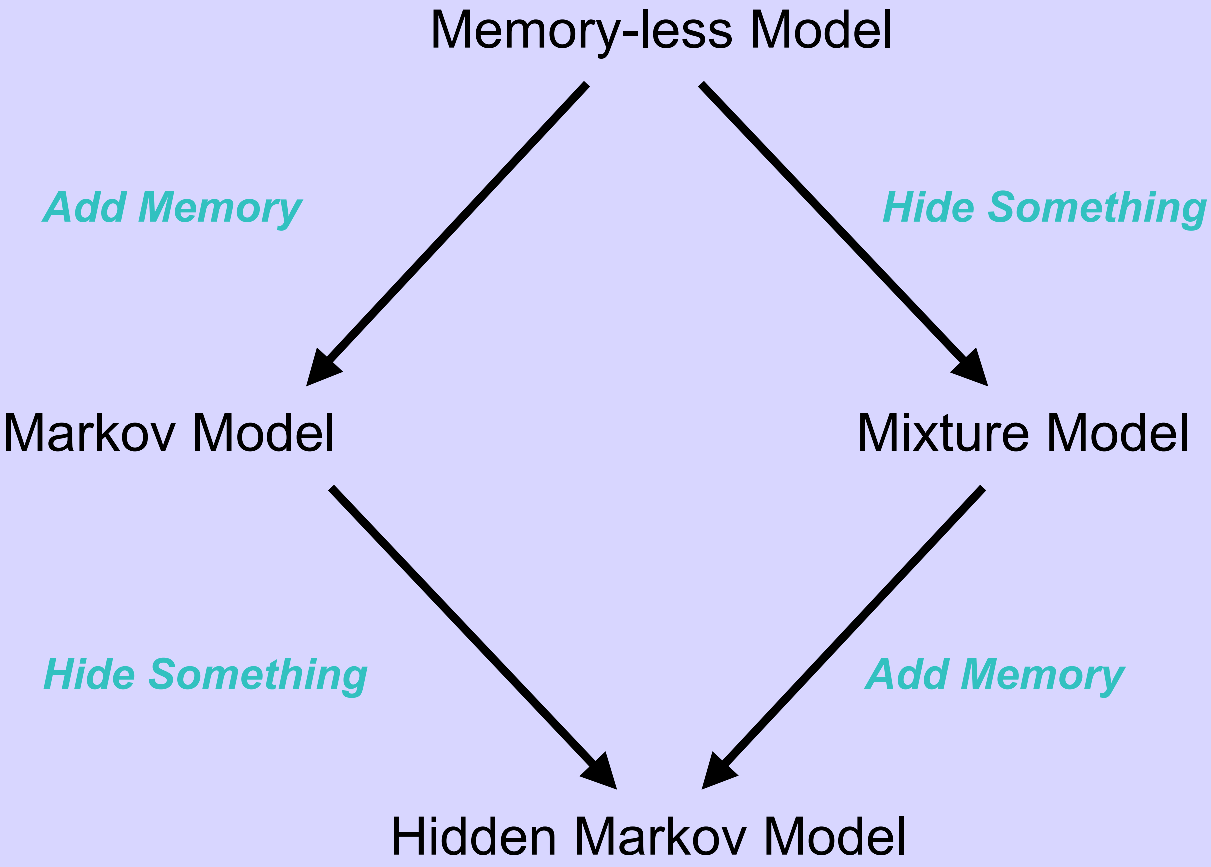
Automatic speech recognition – Architecture



$$W^* = \underset{W}{\arg\max}\; P(X \mid W, \Theta)\, P(W \mid \Theta)$$

Probability of “I no” vs “eye know” vs “I know”

For the rest of lecture, focus on acoustic modeling component



Memory-less Model

Add Memory

Hide Something

Markov Model

Mixture Model

Hide Something

Add Memory

Hidden Markov Model

Memory-less Model Example

- A coin has probability of “heads” = p , probability of “tails” = 1-p
- Flip the coin 10 times. (Bernoulli trials, I.I.D. random sequence.) There are 2^{10} possible sequences.
- Sequence: 1 0 1 0 0 0 1 0 0 1
  Probability: $p(1-p)p(1-p)(1-p)(1-p)\ p(1-p)(1-p)p \quad = \quad p^4(1-p)^6$
- Probability is the same for all sequences with 4 heads & 6 tails. Order of heads & tails does not matter in assigning a probability to the sequence, only the number of heads & number of tails
- Probability of 0 heads $(1-p)^{10}$
  1 head $p(1-p)^9$
  …
  10 heads p^{10}

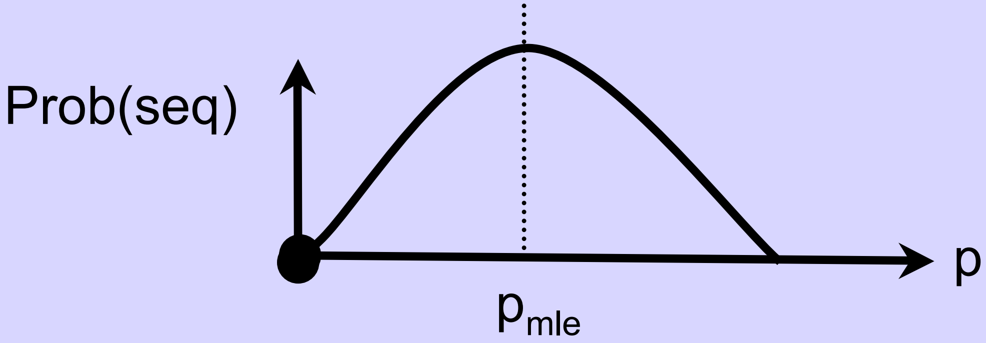
Memory-less Model Example, cont'd

If p is known, then it is easy to compute the probability of the sequence. Now suppose p is unknown.

We toss the coin N times, obtaining H heads and T tails, where H+T=N
We want to estimate p

A "reasonable" estimate is p=H/N. Is this the "best" choice for p?

First, define "best." Consider the probability of the observed sequence. $\text{Prob(seq)}=p^H(1-p)^T$



The value of p for which Prob(seq) is maximized is the Maximum Likelihood Estimate (MLE) of p. (Denote p_{mle})

Memory-less Model Example, cont’d

Theorem: $p_{mle} = H/N$

Proof: $Prob(seq)=p^H(1-p)^T$

Maximizing Prob is equivalent to maximizing log(Prob)

$L=\log(Prob(seq)) = H \log p + T \log (1-p)$

$\delta L/\delta p = H/p - T/(1-p)$

L maximized when $\delta L/\delta p = 0$

$H/p_{mle} - T/(1-p_{mle}) = 0$

$H - H p_{mle} = T p_{mle}$

$H = T p_{mle} + H p_{mle} = p_{mle} (T + H) = p_{mle} N$

$p_{mle} = H/N$

Memory-less Model Example, cont'd

- We showed that in this case
  MLE = Relative Frequency = H/N
- We will use this idea many times.
- Often, parameter estimation reduces to
  counting and normalizing.

Markov Models

- Flipping a coin was memory-less. The outcome of each flip did not depend on the outcome of the other flips.

- Adding memory to a memory-less model gives us a Markov Model. Useful for modeling sequences of events.

Markov Model Example

- Consider 2 coins.
  Coin 1: $p_H = 0.9$, $p_T = 0.1$
  Coin 2: $p_H = 0.2$, $p_T = 0.8$

- Experiment:

```
Flip Coin 1.
for J = 2 ; J<=4; J++
        if (previous flip == “H”)   flip Coin 1;
        else  flip Coin 2;
```

- Consider the following 2 sequences:
  H H T T prob = 0.9 x 0.9 x 0.1 x 0.8
  H T H T prob = 0.9 x 0.1 x 0.2 x 0.1

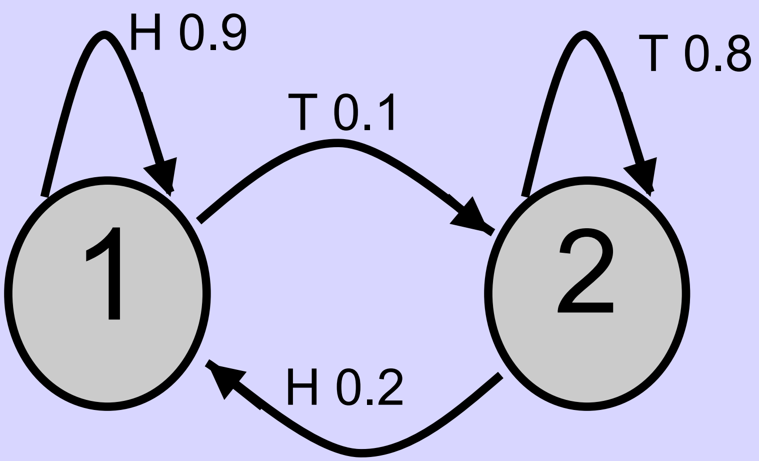
- Sequences with consecutive heads or tails are more likely.
- The sequence has memory.
- Order matters.
- Speech has memory. (The sequence of feature vectors for “rat” are different from the sequence of vectors for “tar.”)

Markov Model Example, cont'd

- Consider 2 coins.

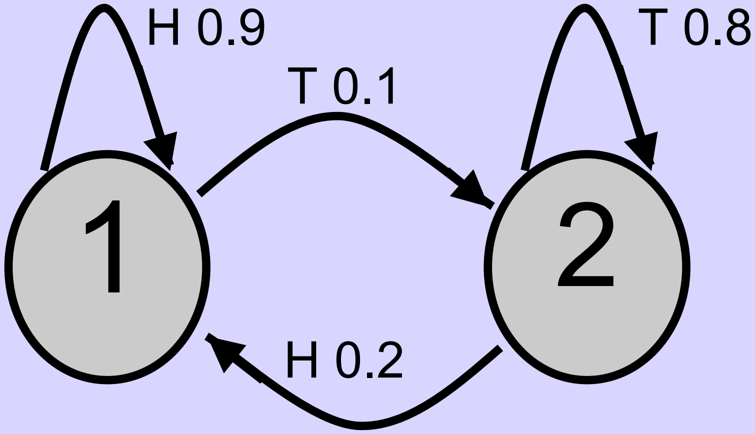
  Coin 1: $p_H = 0.9$, $p_T = 0.1$
  Coin 2: $p_H = 0.2$, $p_T = 0.8$

State-space representation:



Markov Model Example, cont’d

- State sequence can be uniquely determined from the outcome sequence, given the initial state.
- Output probability is easy to compute. It is the product of the transition probs for state sequence.



- Example: O: H T T T
  S: 1(given) 1 2 2
  Prob: 0.9 x 0.1 x 0.8 x 0.8

Mixture Model Example

- Recall the memory-less model. Flip 1 coin.
- Now, let's build on that model, hiding something.

Consider 3 coins. Coin 0: $p_H = 0.7$
Coin 1: $p_H = 0.9$
Coin 2 $p_H = 0.2$

Experiment:

```
For J=1..4
        Flip coin 0. If outcome == "H"
                Flip coin 1 and record.
            else
                Flip coin 2 and record.
```

Note: the outcome of coin 0 is not recorded -- it is "hidden."

Mixture Model Example, cont’d

Coin 0: p_H = 0.7 Coin 1: p_H = 0.9 Coin 2: p_H = 0.2

We cannot uniquely determine the output of the Coin 0 flips. This is hidden.

Consider the sequence H T T T.
What is the probability of the sequence?

Order doesn’t matter (memory-less)

$p(head)=p(head|coin0=H)p(coin0=H)+$
$p(head|coin0=T)p(coin0=T)= 0.9x0.7 + 0.2x0.3 = 0.69$

$p(tail) = 0.1 \times 0.7 + 0.8 \times 0.3 = 0.31$

$P(HTTT) = .69 \times .31^{3}$

Hidden Markov Model

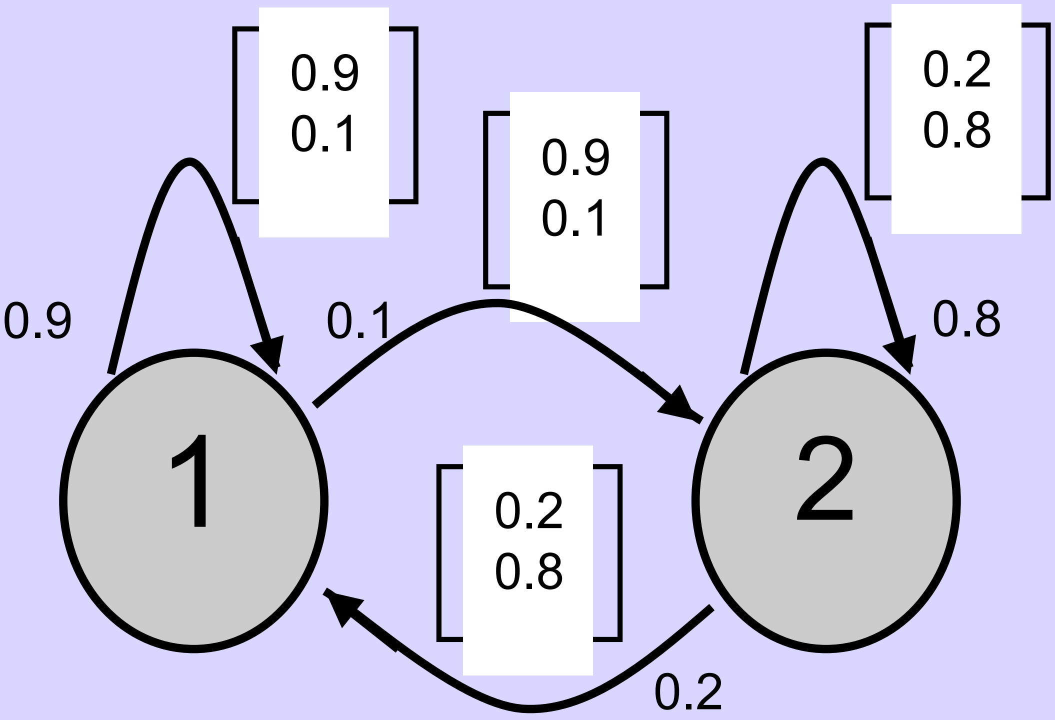
- The state sequence is hidden.
- Unlike Markov Models, the state sequence cannot be uniquely deduced from the output sequence.

- Experiment:
  Flipping the same two coins. This time, flip each coin twice. The first flip gets recorded as the output sequence. The second flip determines which coin gets flipped next.

- Now, consider output sequence H T T T.
- No way to know the results of the even numbered flips, so no way to know which coin is flipped each time.

Hidden Markov Model

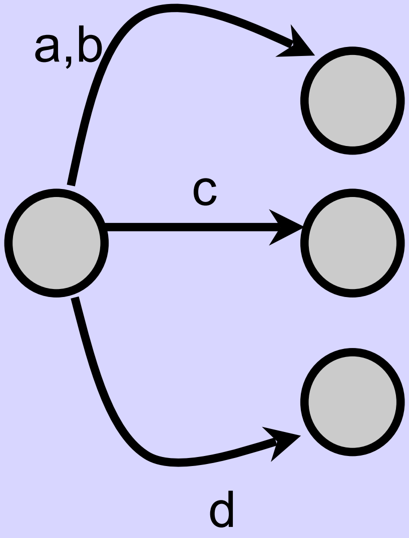
- The state sequence is hidden. Unlike Markov Models, the state sequence cannot be uniquely deduced from the output sequence.



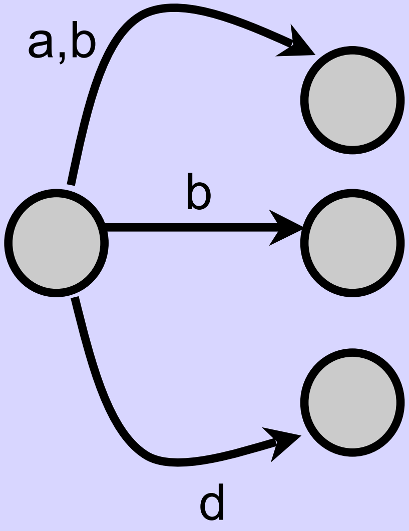
- In speech, the underlying states can be, say the positions of the articulators. These are hidden – they are not uniquely deduced from the output features. We already mentioned that speech has memory. A process which has memory and hidden states implies HMM.

Is a Markov Model Hidden or Not?

A necessary and sufficient condition for being state-observable is that all transitions from each state produce different outputs



State-observable



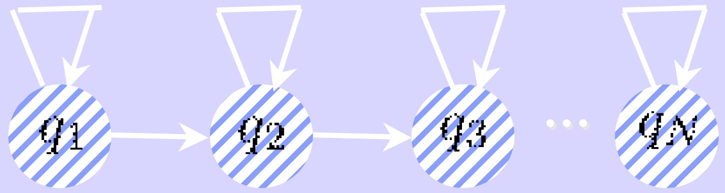
Hidden

Markov Models -- quick recap

- Markov model:

  States correspond to an observable (physical) event

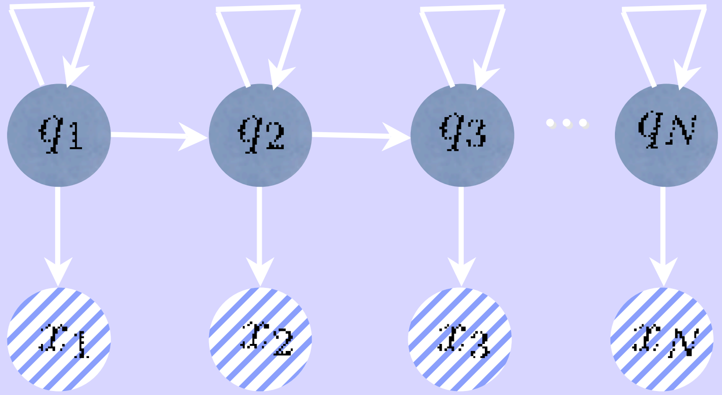
  In graph to right, each x can take one value -- x's are collapsed into q's



- **Hidden** Markov model:

  The observation is a probabilistic function of the state q

  Doubly stochastic process: both the transition between states, and the observation generation are probabilistic



Three problems of general interest for an HMM

3 problems need to be solved before we can use HMM's:

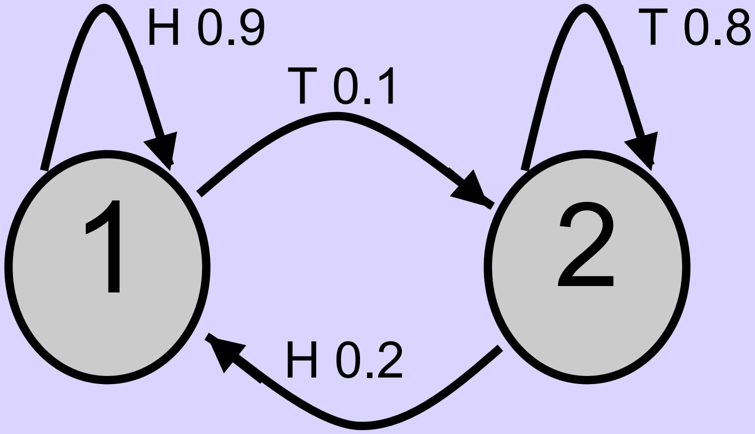
- 1. Evaluation: Given an observed output sequence $X=x_1x_2..x_T$, compute $P_\theta(X)$ for a given model θ. (solution: **Forward algorithm**)
- 2. Decoding: Given X, find the most likely state sequence (solution: **Viterbi algorithm**)
- 3. Training: Estimate the parameters of the model. (solution: **Baum-Welch algorithm**, a.k.a. Forward-Backward algorithm)

These problems are easy to solve for a state-observable Markov model. More complicated for an HMM because we need to consider all possible state sequences. Must develop a generalization….

Problem 1-- the state observable case (easy)

1. Given an observed output sequence $X=x_1x_2..x_T$, compute Pθ(X) for a given model θ

- Recall the state-observable case



- Example:

O:	H		T		T		T
S:	1(given)		1		2		2
Prob:	0.9	x	0.1	x	0.8	x	0.8

Problem 1 -- for a hidden Markov model (not easy)

1. Given an observed output sequence $X=x_1x_2..x_T$, compute $P_\theta(X)$ for a given model θ

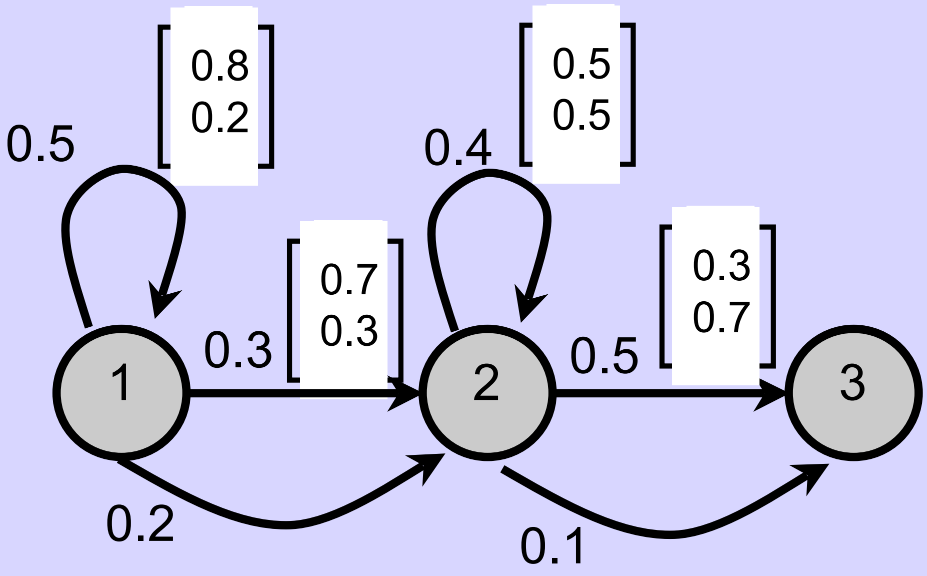
Sum over all possible state sequences:

$$P_\theta(X)=\Sigma_S P_\theta(X,S)$$

The obvious way of calculating $P_\theta(X)$ is to enumerate all state sequences that produce X

Unfortunately, this calculation is exponential in the length of the sequence

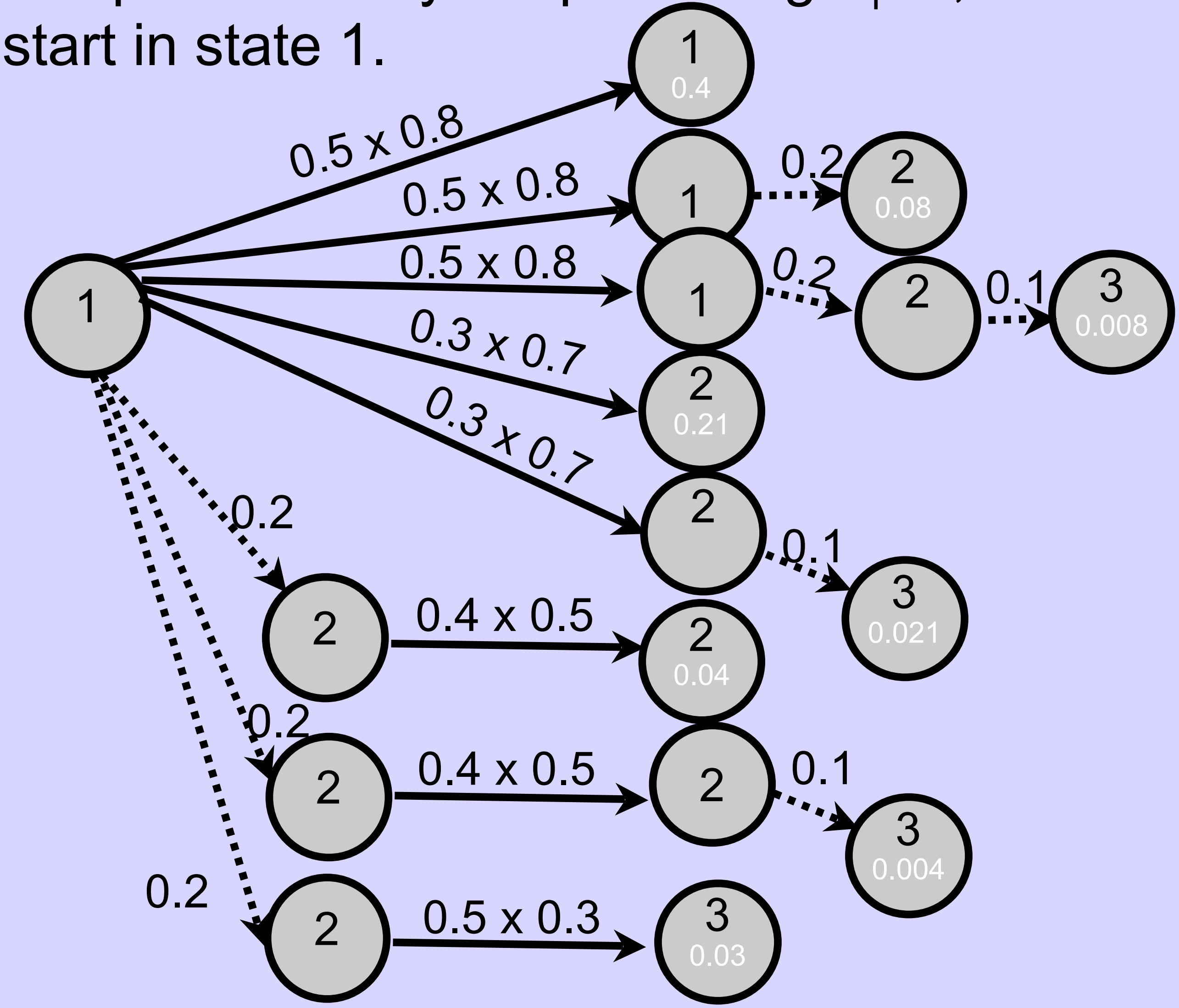
Example for Problem 1 -- for HMM



Compute Pθ(X) for X=aabb, assuming we start in state 1

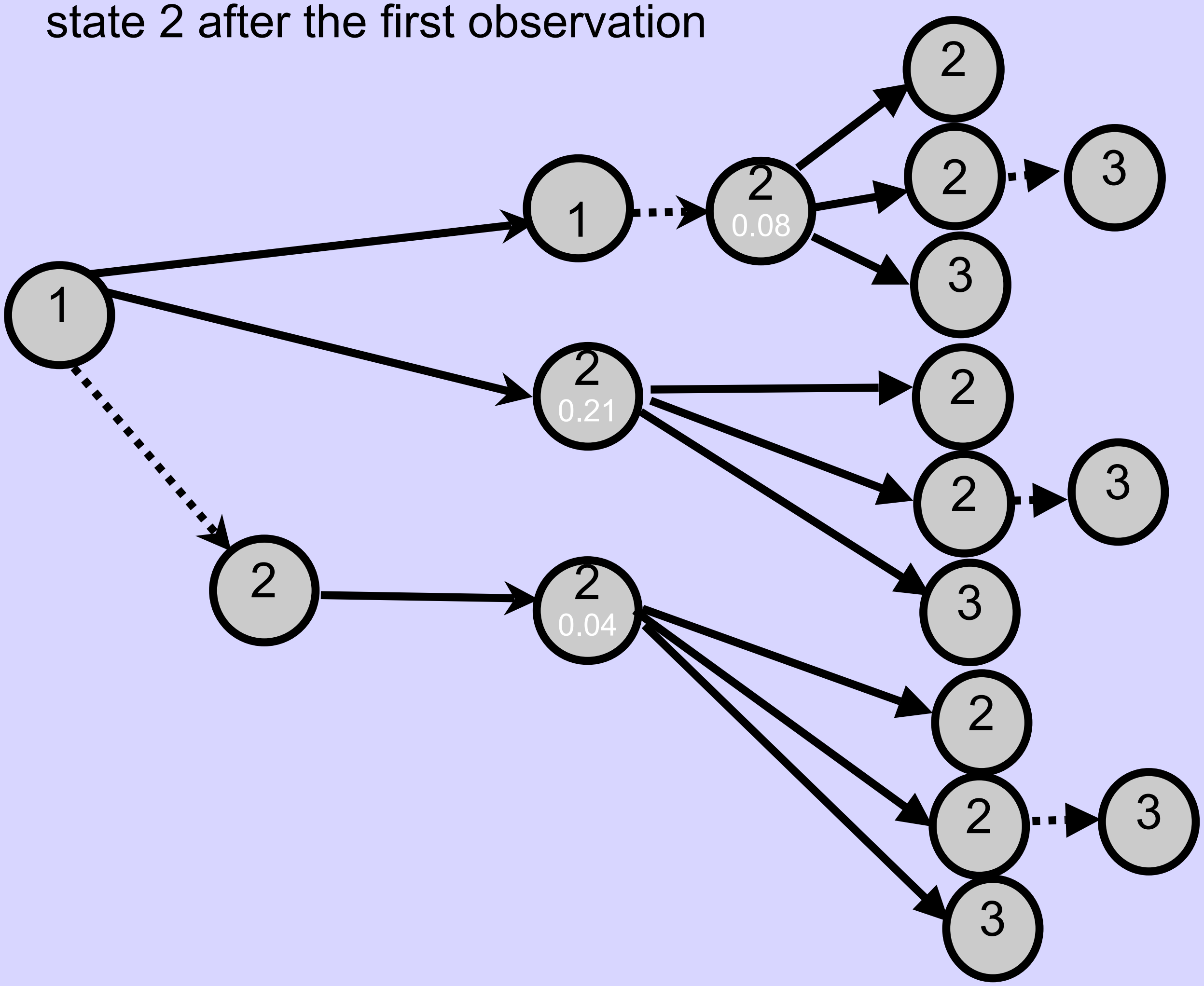
Example for Problem 1,cont'd

Let's enumerate all possible ways of producing x_1=a, assuming we start in state 1.



Example for Problem 1, cont'd

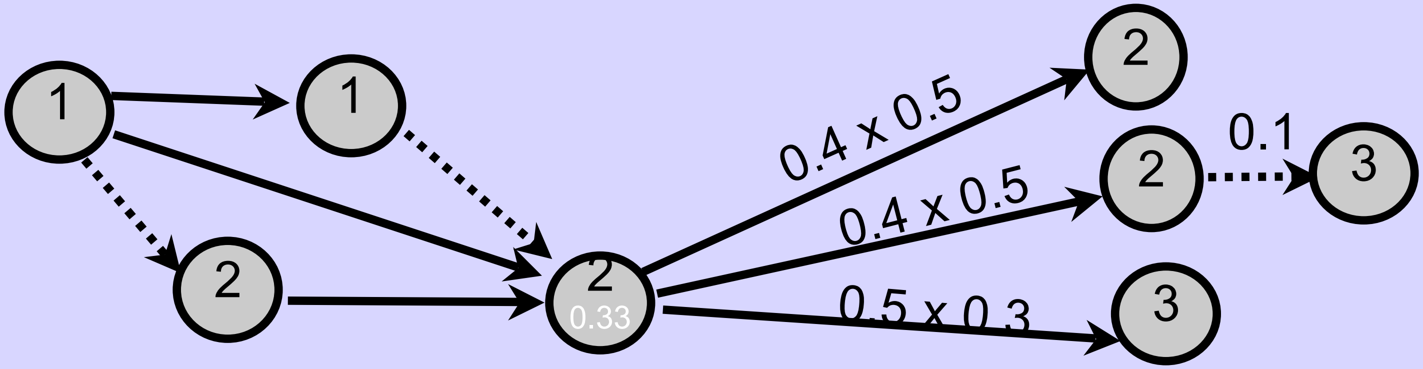
- Now let's think about ways of generating x1x2=aa, for all paths from state 2 after the first observation



Example for Problem 1,cont'd

We can save computations by combining paths.

This is a result of the Markov property, that the future doesn't depend on the past if we know the current state



Side note: Markov Property

- n-th order Markov chain:

  Sequence of discrete random variables that depend only on preceding n variables

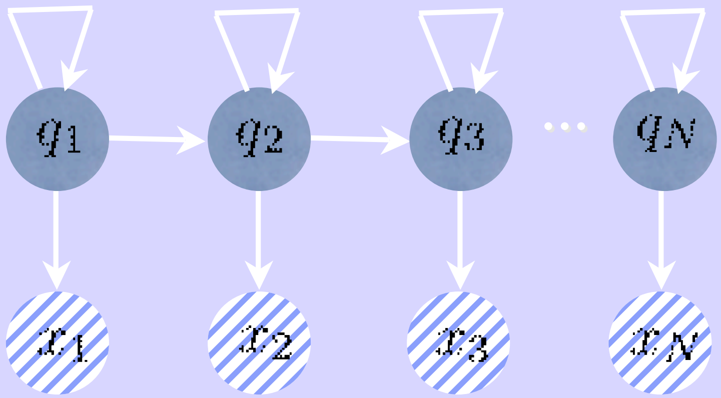
  We focus on “first order” -- depend only on preceding state

- By definition of joint and conditional probability:

$$P(Q = q_1 q_2 ... q_N) = P(q_1)P(q_2|q_1)P(q_3|q_2 q_1)P(q_4|q_3 q_2 q_1)...$$

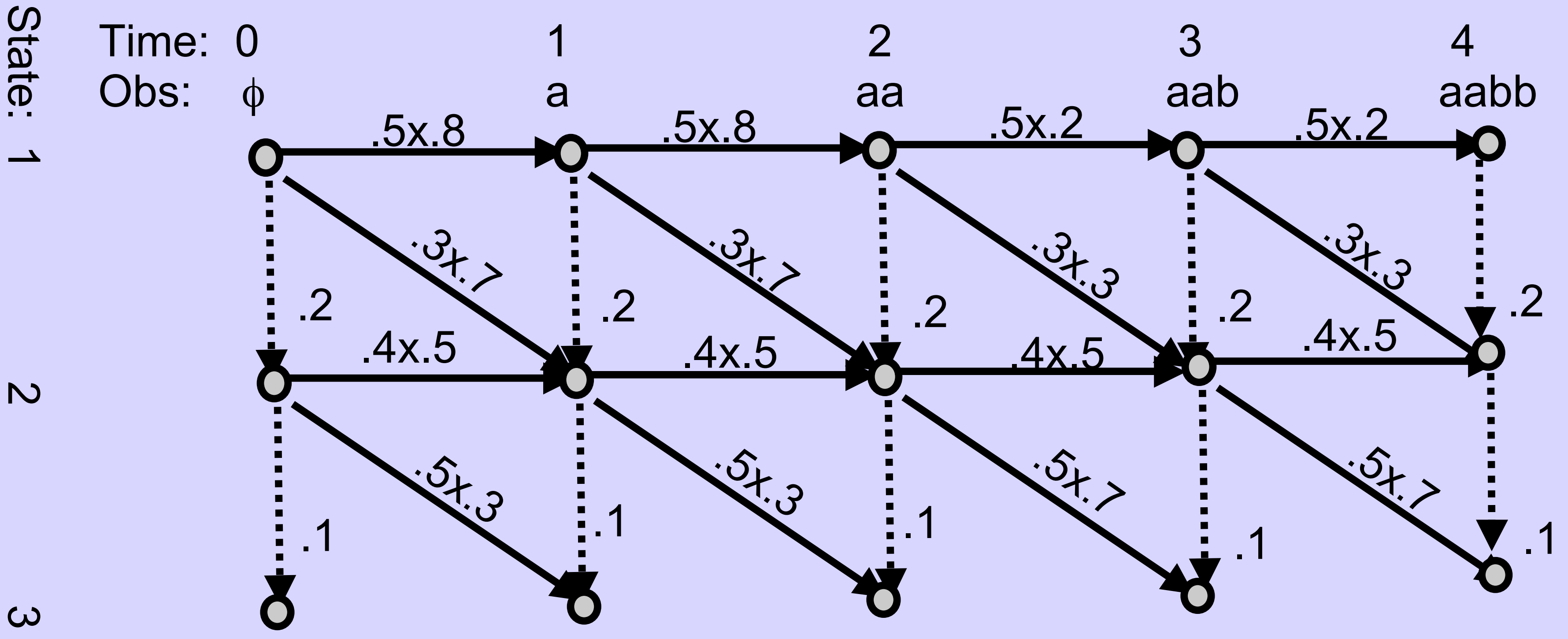
$$= P(q_1) \prod_{i=2}^{N} P(q_i | q_{i-1} q_{i-2} \cdots q_i)$$

cut here for 1st order Markov chain



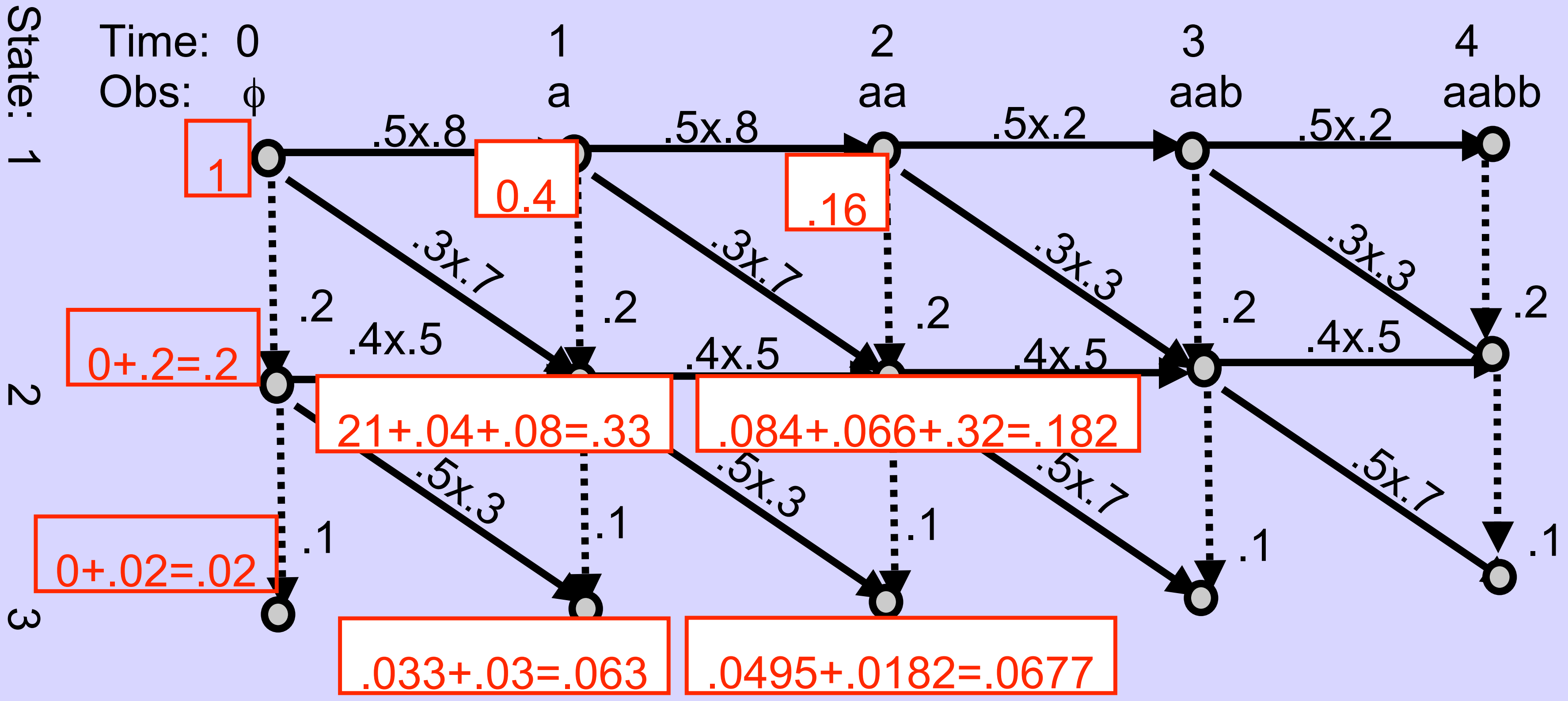
Problem 1: Trellis Diagram

- Expand the state-transition diagram in time.
- Create a 2-D lattice indexed by state and time.
- Each state transition sequence is represented exactly once.



Problem 1: Trellis Diagram, cont’d

- Now let’s accumulate the scores. Note that the inputs to a node are from the left and top, so if we work to the right and down all necessary input scores will be available.



Problem 1: Trellis Diagram, cont'd

Boundary condition:
Score of (state 1, ϕ) = 1.

Basic recursion:
Score of node i = 0

For the set of predecessor nodes j:

```
    Score of node i += score of predecessor node j  x
                       the transition probability from j to i  x
                       observation probability along
                         that transition if the transition is not null.
```

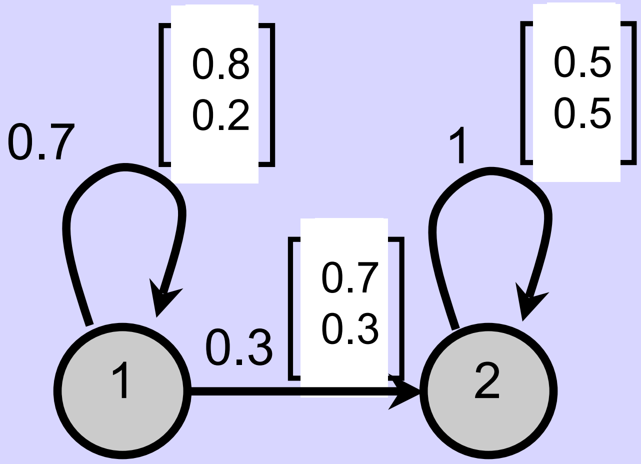
Mealy vs. Moore HMMs

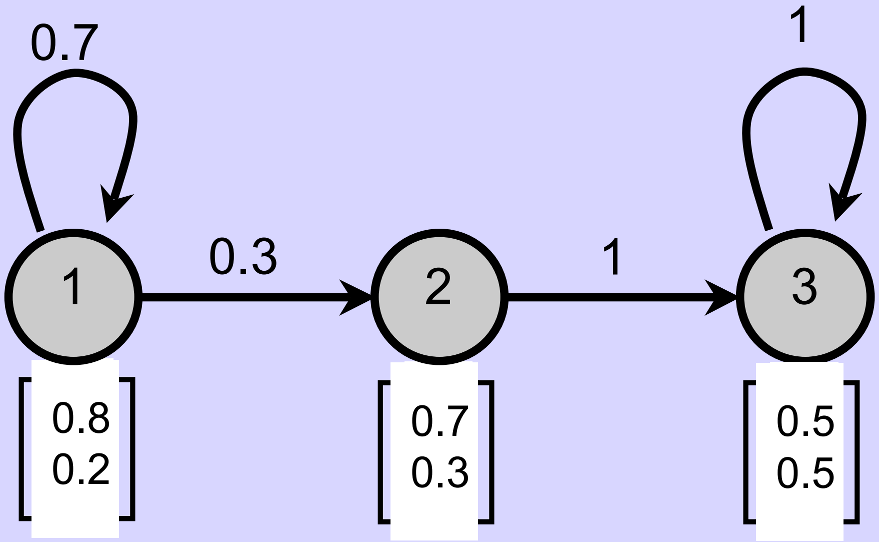
Mealy: “transition emitter” (the slides in this talk)

Moore: “state emitter” (the textbook, and most formulations in ASR)

Mealy and Moore formulations are equivalent!

Moore models require more (pun intended) states to represent the same model.


0.7
$\begin{bmatrix}0.8\\0.2\end{bmatrix}$
1
$\begin{bmatrix}0.5\\0.5\end{bmatrix}$
0.3
$\begin{bmatrix}0.7\\0.3\end{bmatrix}$
1
2


0.7
1
0.3
1
1
2
3
$\begin{bmatrix}0.8\\0.2\end{bmatrix}$
$\begin{bmatrix}0.7\\0.3\end{bmatrix}$
$\begin{bmatrix}0.5\\0.5\end{bmatrix}$

Forward Algorithm -- Mealy (emission on transition) vs. Moore (emission in state)

α_t(i): probability of being in state i at time t and having produced output $x_1^t=x_1..x_t$
a_{ij}: transition probability from state i to state j
$b_{ij}(x_t)$: emission probability of x at time t from state i to j (reduces to $b_j(x_t)$ for Moore)

Step 1 -- Initialization ("general" form)

$$\alpha_1(i) = \pi_i b_{ij}(X_1)$$

Mealy (there is no emission by starting in the initial state -- only on transition):

$$\alpha_1(i) = p(q_i^1)$$

Moore:

$$\alpha_1(i) = p(q_i^1)p(x_1 \mid q_i)$$

Forward Algorithm -- Mealy (emission on transition) vs. Moore (emission in state)

$\alpha_t(i)$: probability of being in state i at time t and having produced output $x_1^t = x_1..x_t$
a_{ij}: transition probability from state i to state j
$b_{ij}(x_t)$: emission probability of x at time t from state i to j (reduces to $b_j(x_t)$ for Moore)

Step 2 -- Induction ("general" form:)

Mealy:

$$\alpha_{t+1}(j) = \sum_{i=1}^{S} \alpha_t(i) a_{ij} b_{ij}(x_{t+1})$$

$$\alpha_{t+1}(j) = \sum_{i=1}^{S} \alpha_t(i)\, p(q_j \mid q_i) p(x_{t+1} \mid q_i \rightarrow q_j)$$

Moore (there is no i in b_{ij} for Moore, as emissions not on transition from i to j, but in state j)

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{S} \alpha_t(i)\, p(q_j \mid q_i) \right] p(x_{t+1} \mid q_j)$$

Forward Algorithm -- Mealy (emission on transition) vs. Moore (emission in state)

$\alpha_t(i)$: probability of being in state i at time t and having produced output $x_1^t = x_1..x_t$
a_{ij}: transition probability from state i to state j
$b_{ij}(x_t)$: emission probability of x at time t from state i to j (reduces to $b_j(x_t)$ for Moore)

- **Step 3 -- Termination:**

$$P(X \mid M) = \sum_{i=1}^{S} \alpha_N(i)$$

Important: The computational complexity of the forward algorithm is linear in time (or in the length of the observation sequence)

Problem 2

Given the observations X, find the most likely state sequence

This is solved using the Viterbi algorithm

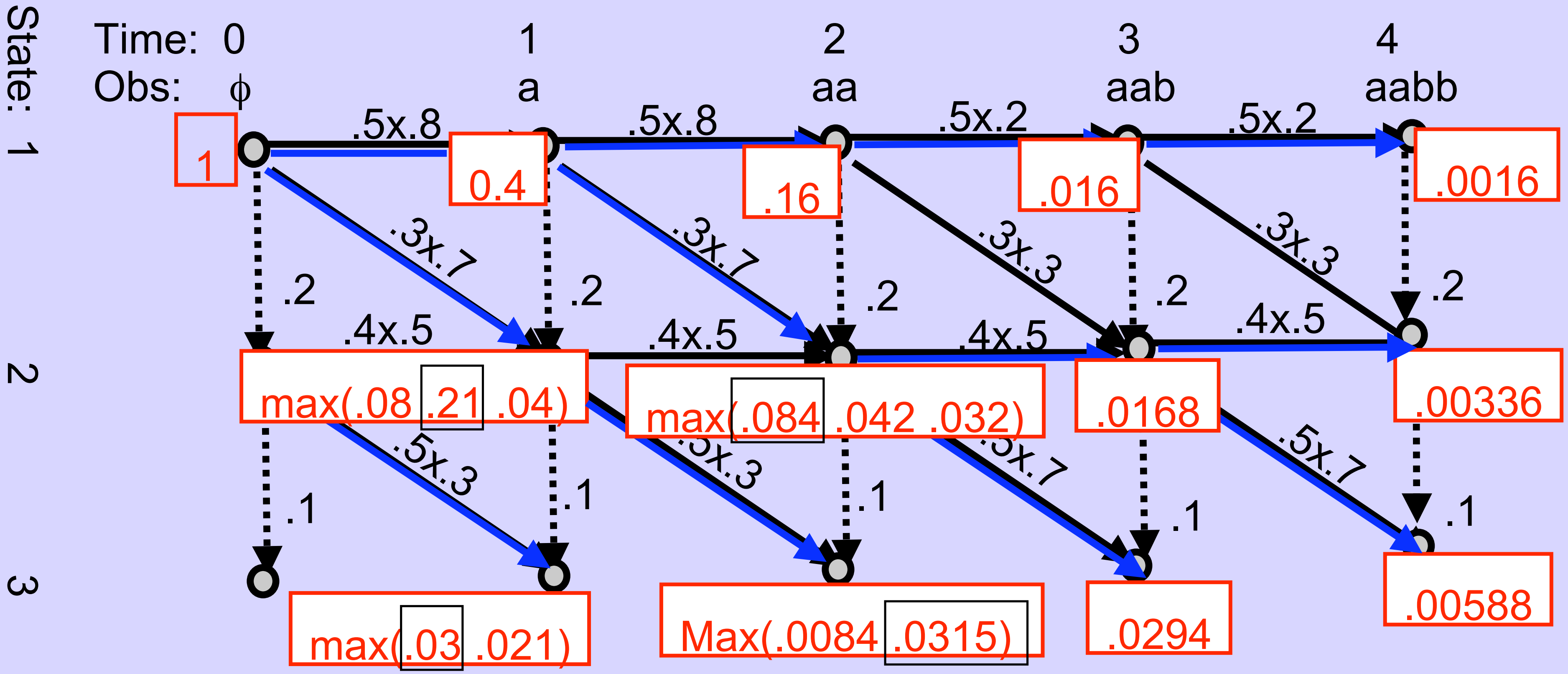
Preview:

The computation is similar to the forward algorithm, except we use max( ) instead of +

Also, we need to remember which partial path led to the max

Problem 2: Viterbi algorithm

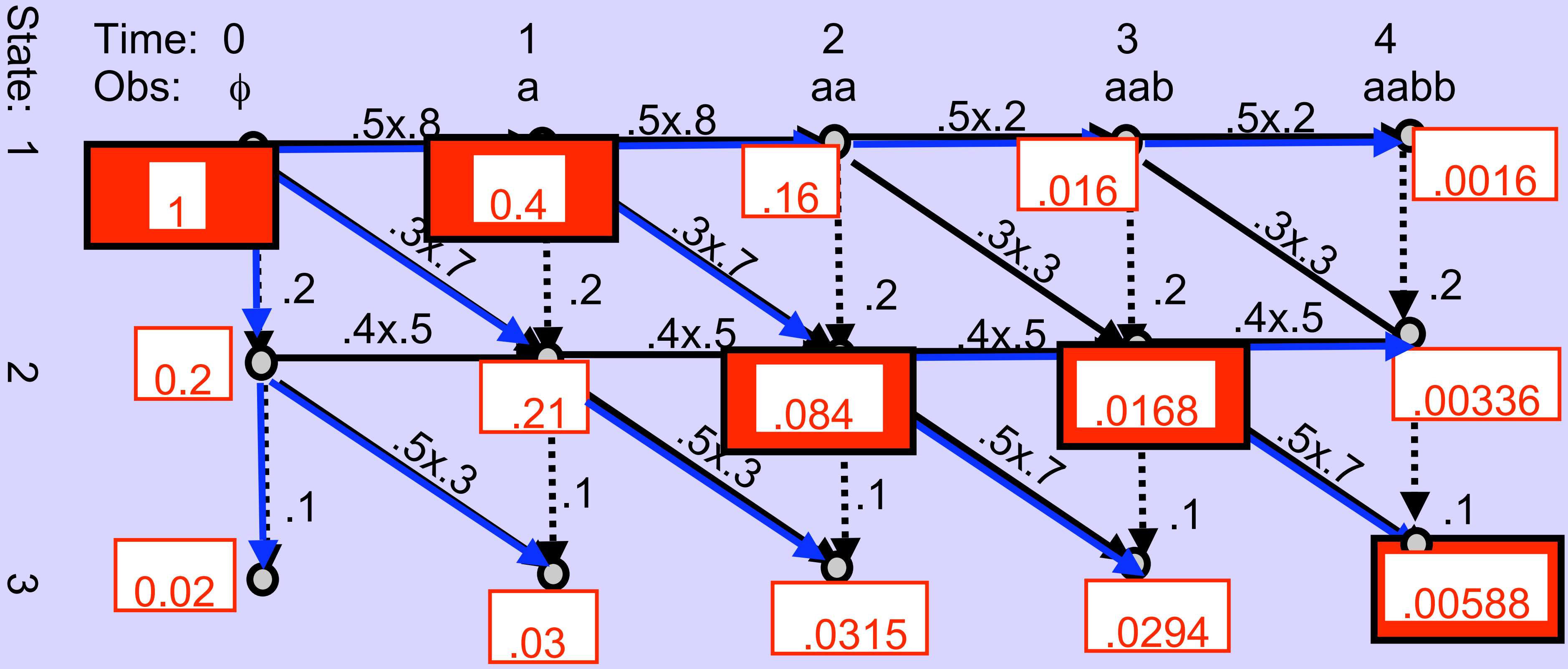
Returning to our example, let's find the most likely path for producing aabb. At each node, remember the max of predecessor score x transition probability. Also store the best predecessor for each node.



Problem 2: Viterbi algorithm, cont'd

Starting at the end, find the node with the highest score.

Trace back the path to the beginning, following best arc leading into each node along the best path.



Problem 3

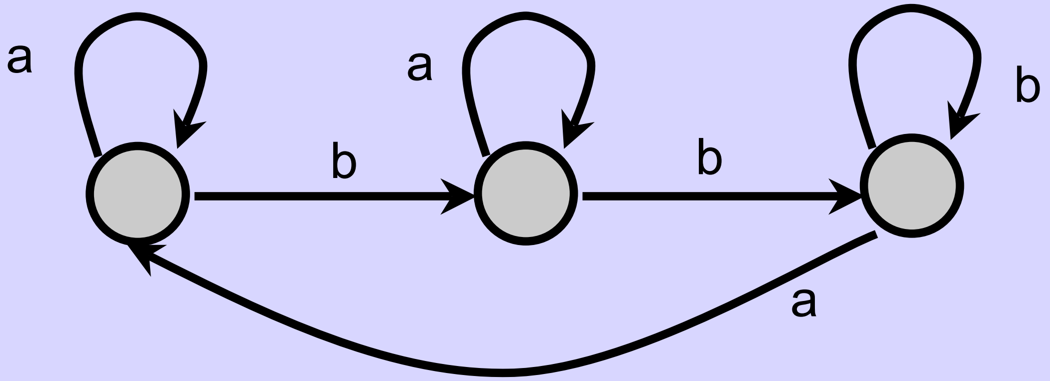
Estimate the parameters of the model. (training)

- Given a model topology and an output sequence, find the transition and output probabilities such that the probability of the output sequence is maximized.

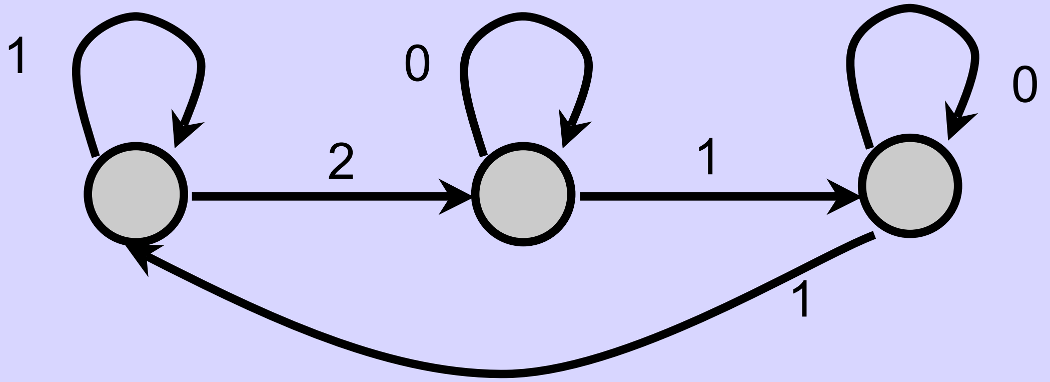
- Recall in the state-observable case, we simply followed the unique path, giving a count to each transition.

Problem 3 – State Observable Example

- Assume the output sequence X=abbab, and we start in state 1.

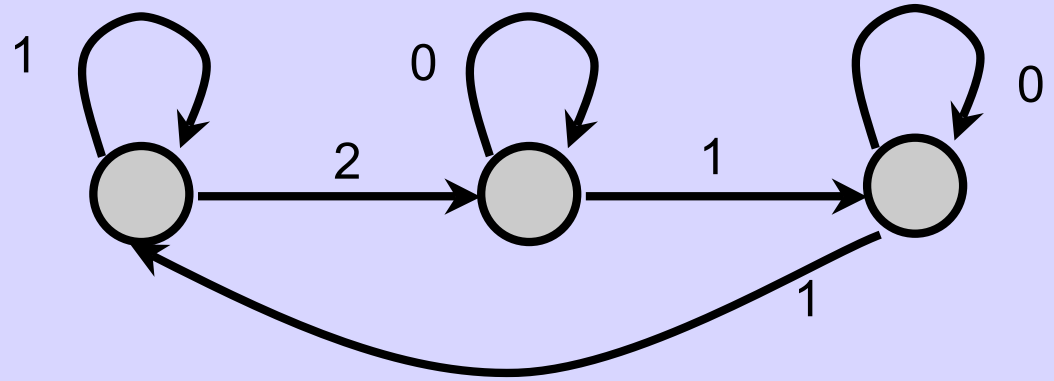


- Observed counts along transitions:

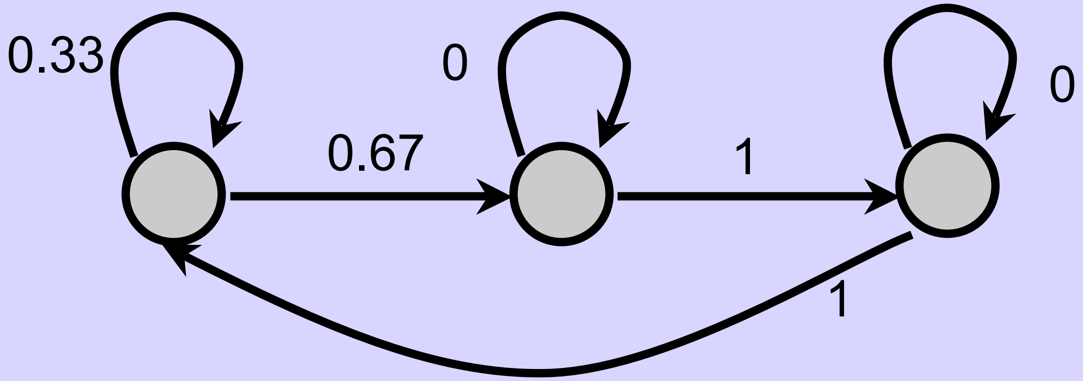


Problem 3 – State Observable Example

Observed counts along transitions:



Estimated transition probabilities. (this is of course too little data to estimate these well.)



Generalization to Hidden MM case

State-observable

- Unique path
- Give a count of 1 to each transition along the path

Hidden states

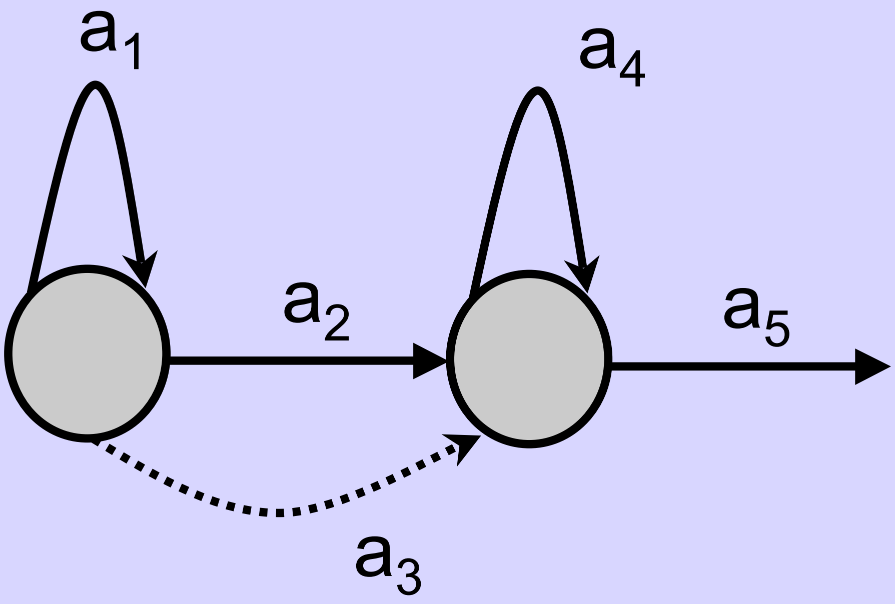
- Many paths
- Assign a fractional count to each path
- For each transition on a given path, give the fractional count for that path
- Sum of the fractional counts =1
- How to assign the fractional counts??

How to assign the fractional counts to the paths

- Guess some values for the parameters
- Compute the probability for each path using these parameter values
- Assign path counts in proportion to these probabilities
- Re-estimate parameter values
- Iterate until parameters converge

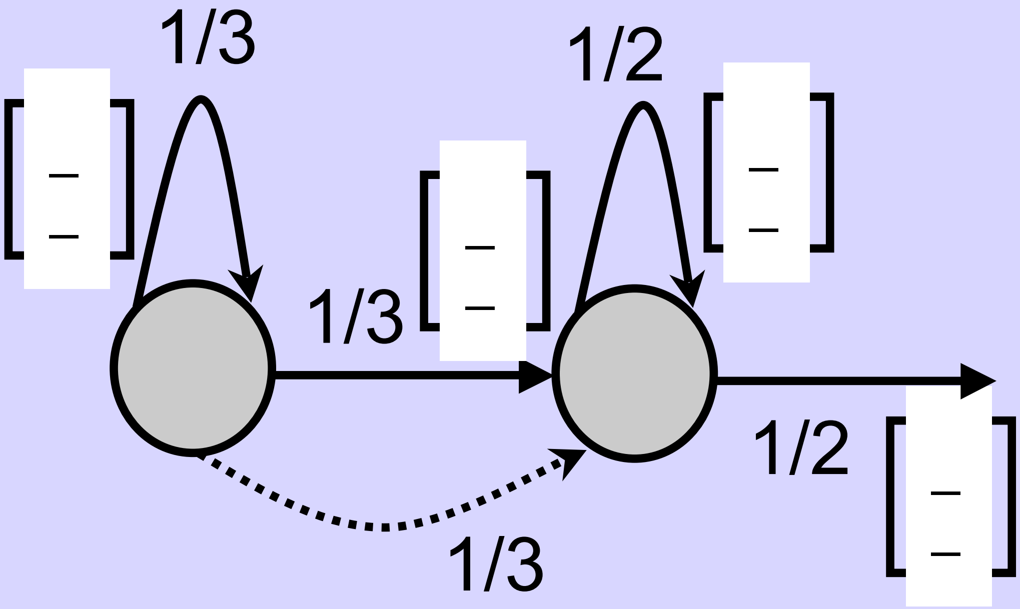
Problem 3: Enumerative Example

- For the following model, estimate the transition probabilities and the output probabilities for the sequence X=abaa

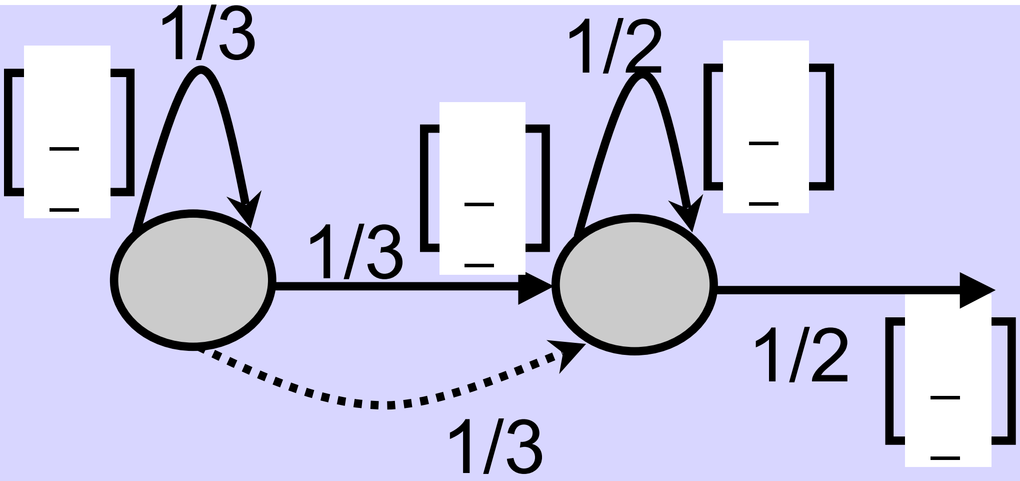


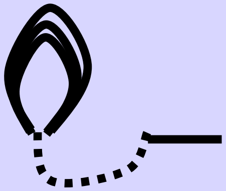
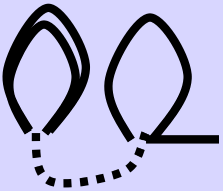
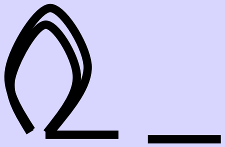
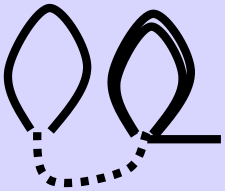
Problem 3: Enumerative Example

- Initial guess: equiprobable

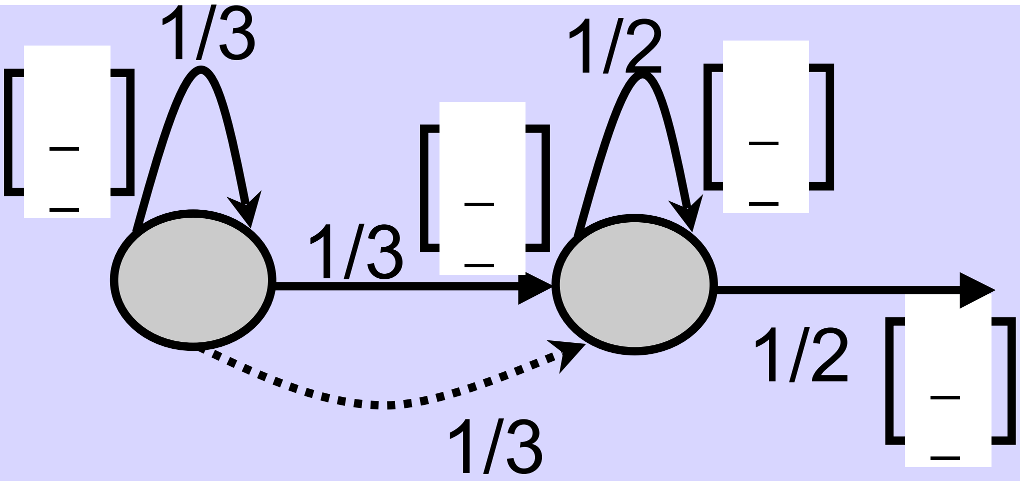


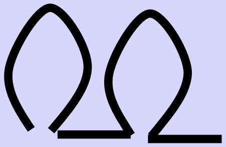
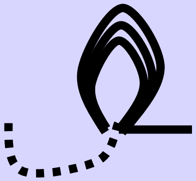
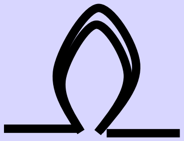
Problem 3: Enumerative Example cont’d



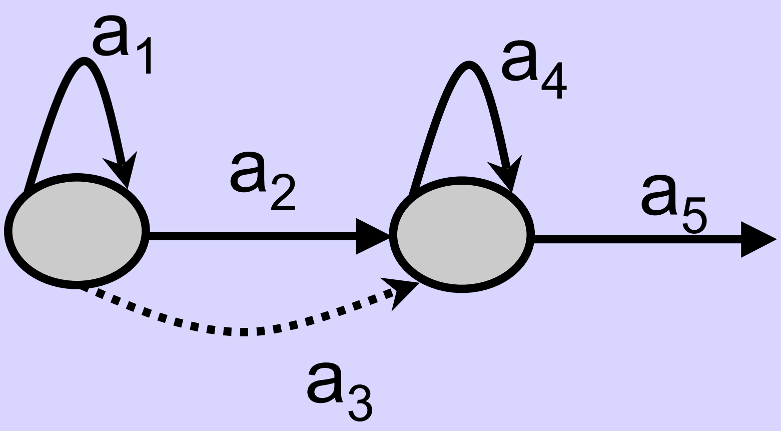
- 7 paths:
- 1.  pr(X,path1)=1/3x1/2x1/3x1/2x1/3x1/2x1/3x1/2x1/2=.000385
- 2.  pr(X,path2)=1/3x1/2x1/3x1/2x1/3x1/2x1/2x1/2x1/2=.000578
- 3.  pr(X,path3)=1/3x1/2x1/3x1/2x1/3x1/2x1/2x1/2=.001157
- 4.  pr(X,path4)=1/3x1/2x1/3x1/2x1/2x1/2x1/2x1/2x1/2=.000868

Problem 3: Enumerative Example cont'd



- 7 paths:
- 5.  pr(X,path5)=1/3x1/2x1/3x1/2x1/2x1/2x1/2x1/2=.001736
- 6.  pr(X,path6)=1/3x1/2x1/2x1/2x1/2x1/2x1/2x1/2x1/2=.001302
- 7.  pr(X,path7)=1/3x1/2x1/2x1/2x1/2x1/2x1/2x1/2=.002604
- $Pr(X) = \Sigma_i\, pr(X,path_i) = .008632$

Problem 3: Enumerative Example cont'd



- Let C_i be the a posteriori probability of path i
- $C_i = pr(X,path_i)/pr(X)$
-
- $C_1 = .045$ $C_2 = .067$ $C_3 = .134$ $C_4 = .100$ $C_5 = .201$ $C_6 = .150$ $C_7 = .301$

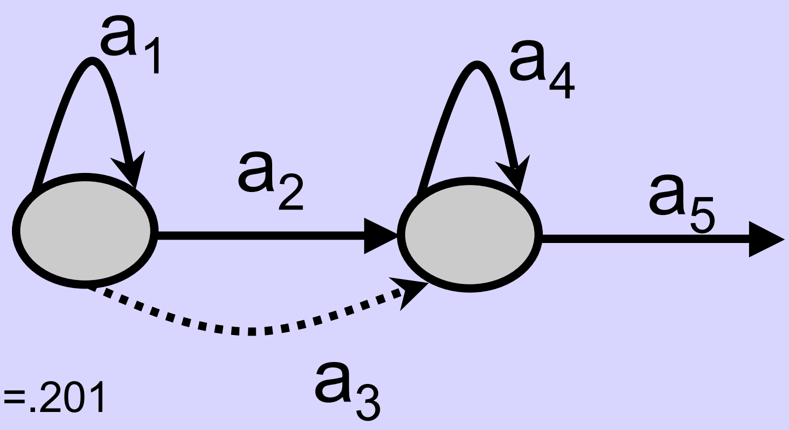
- $Count(a_1) = 3C_1+2C_2+2C_3+C_4+C_5 = .838$
- $Count(a_2) = C_3+C_5+C_7 = .637$
- $Count(a_3) = C_1+C_2+C_4+C_6 = .363$

- New estimates (after normalization to add up to 1):
- $a_1 = .46$ $a_2 = .34$ $a_3 = .20$

- $Count(a_1, 'a') = 2C_1+C_2+C_3+C_4+C_5 = .592$ $Count(a_1, 'b') = C_1+C_2+C_3 = .246$

- New estimates:
- $p(a_1, 'a') = .71$ $p(a_1, 'b') = .29$

Problem 3: Enumerative Example cont'd



- Count(a_2,'a') = C_3+C_7 = .436   Count(a_2,'b')=C_5 =.201

- New estimates:
- p(a_2,'a')= .68   p(a_2,'b')= .32

- Count(a_4)=C_2+2C_4+C_5+3C_6+2C_7 = 1.52
- Count(a_5)=C_1+C_2+C_3+C_4+C_5+C_6+C_7 = 1.00

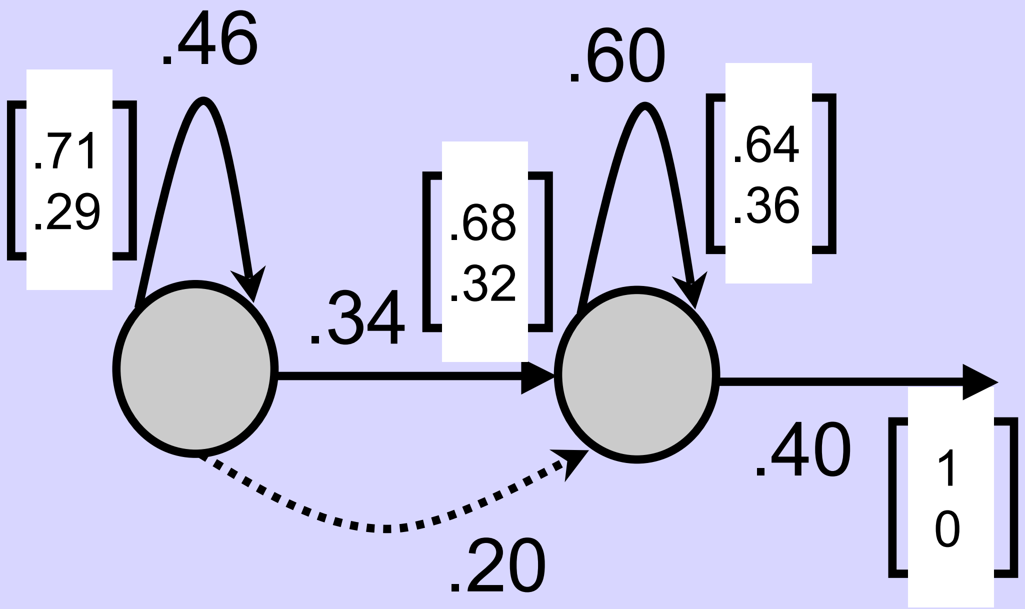
- New estimates: a_4=.60 a_5=.40
- Count(a_4,'a') = C_2+C_4+C_5+2C_6+C_7 = .972   Count(a_4,'b')=C_4+C_6+C_7=.553

- New estimates:
- p(a_4,'a')= .64   p(a_4,'b')= .36

- Count(a_5,'a') = C_1+C_2+C_3+C_4+C_5+2C_6+C_7 = 1.0 Count(a_5,'b')=0
- New estimates:
- p(a_5,'a')= 1.0   p(a_5,'b')= 0
-

Problem 3: Enumerative Example cont'd

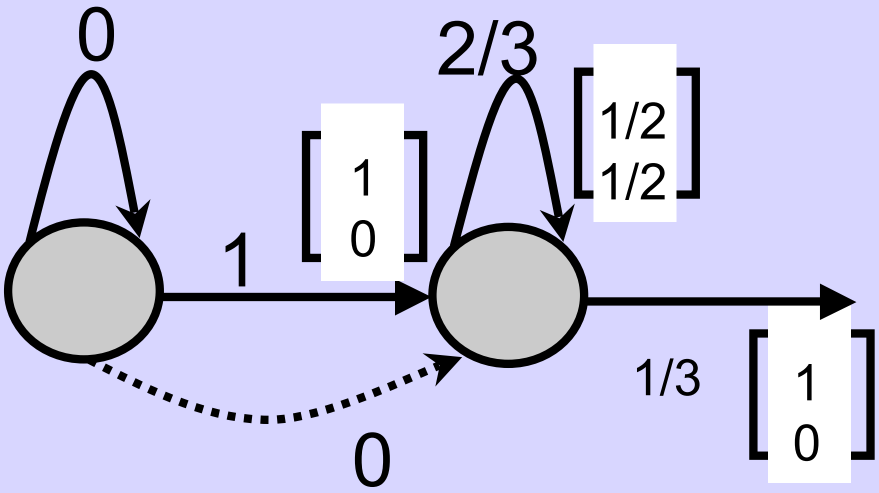
- New parameters



- Recompute Pr(X) = .02438 > .008632
- Keep on repeating…..

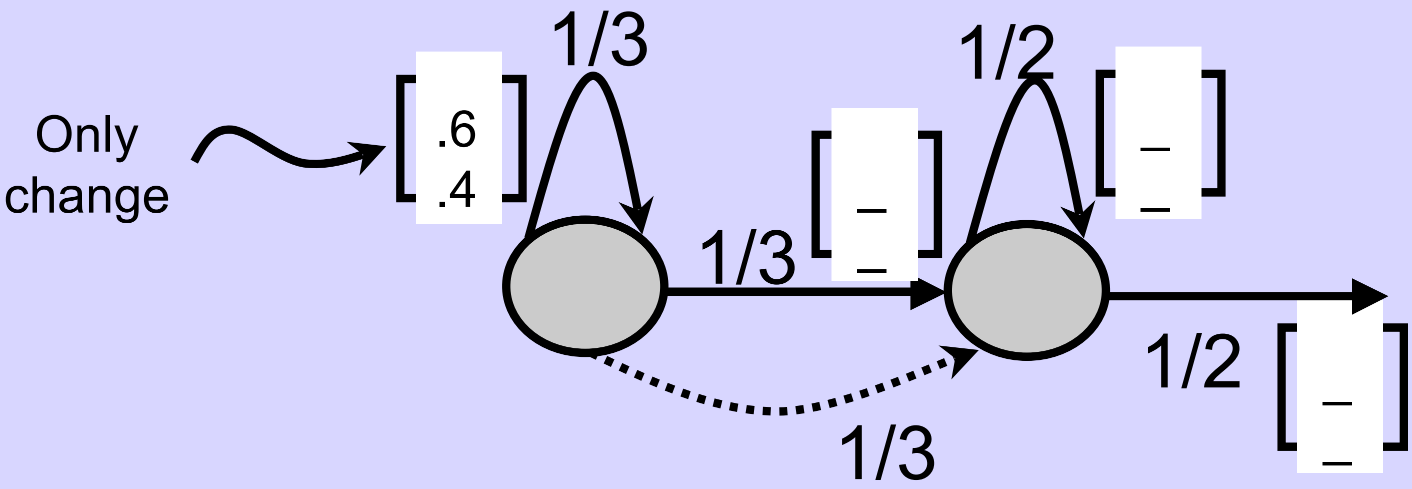
Problem 3: Enumerative Example cont'd

Step	Pr(X)	
1	0.008632	
2	0.02438	
3	0.02508	
100	0.03125004	
600	0.037037037	converged



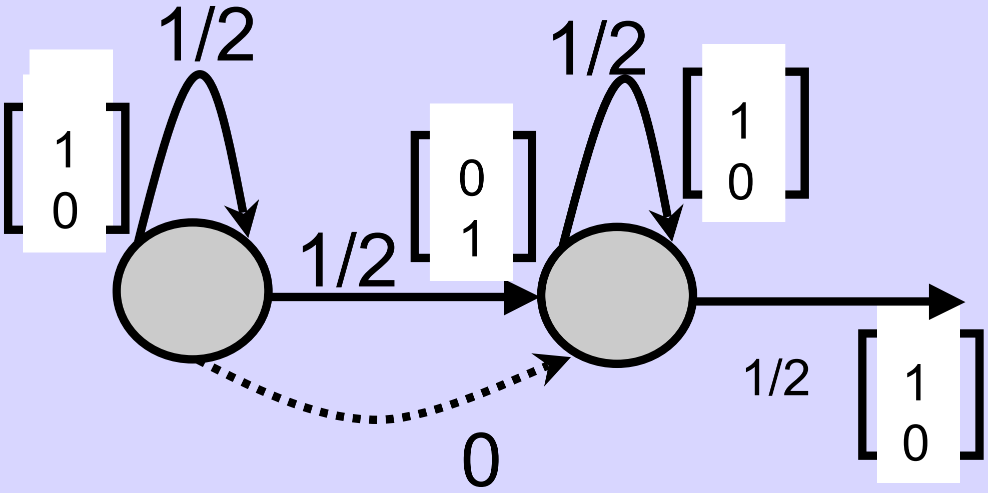
Problem 3: Enumerative Example cont’d

- Let’s try a different initial parameter set



Problem 3: Enumerative Example cont’d

Step	Pr(X)
1	0.00914
2	0.02437
3	0.02507
10	0.04341
16	0.0625 converged



Performance

- The above re-estimation algorithm converges to a local maximum.
- The final solution depends on the starting point.
- The speed of convergence depends on the starting point.

Problem 3: Forward-Backward Algorithm, a.k.a. Baum Welch

- The forward-backward algorithm improves on the enumerative algorithm by using the trellis

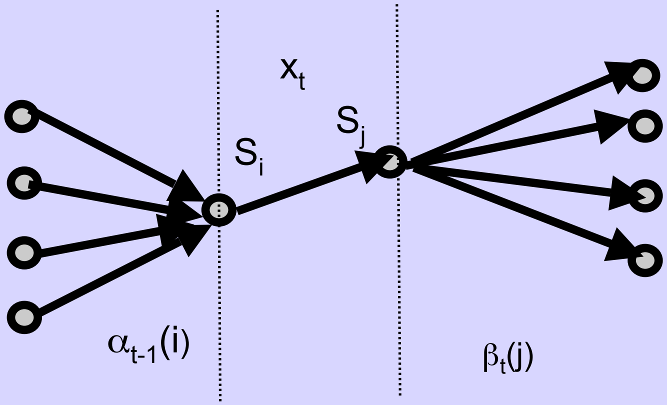
- Instead of computing counts for each path, we compute counts for each transition at each time in the trellis.

- This results in the reduction from exponential computation to linear computation.

Problem 3: Forward-Backward Algorithm

Consider transition from state i to j, tr_{ij}

Let $p_t(tr_{ij},X)$ be the probability that x_t is produced by tr_{ij}, and the complete output is X.



$$p_t(tr_{ij},X) = \alpha_{t-1}(i)\, a_{ij}\, b_{ij}(x_t)\ \ \beta_t(j)$$

Problem 3: F-B algorithm cont’d

$p_t(tr_{ij},X) = \alpha_{t-1}(i)\ a_{ij}\ b_{ij}(x_t)\ \ \beta_t(j)$

where:

$\alpha_{t-1}(i) = Pr(state=i, x_1 \ldots x_{t-1})$ = probability of being in state i and having produced $x_1 \ldots x_{t-1}$

a_{ij} = transition probability from state i to j

$\beta_t(j) = Pr(x_{t+1} \ldots x_T | state= j)$ = probability of producing $x_{t+1} \ldots x_T$ given you are in state j

$b_{ij}(x_t)$ = probability of output symbol x_t along transition ij

Problem 3: F-B algorithm cont'd

- Transition count $c_t(tr_{ij}|X) = p_t(tr_{ij},X) / Pr(X)$
- The β's are computed recursively in a backward pass (analogous to the forward pass for the α's)

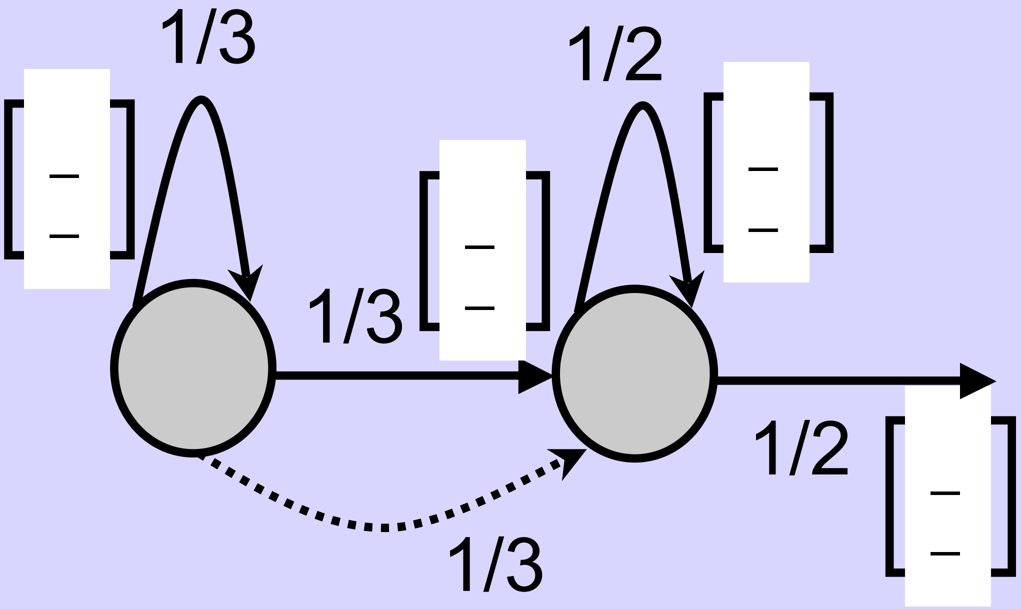
$\beta_t(j) = \Sigma_k\ \beta_{t+1}(k)\ a_{jk}\ b_{jk}(x_{t+1})$ (for all output producing arcs)
$+ \Sigma_k\ \beta_t(k)\ a_{jk}$ (for all null arcs)

$\alpha_t(i) = \Sigma_m\ \alpha_{t-1}(m)\ a_{mi}\ b_{mi}\ (X_t)$
$+ \Sigma_m\ \alpha_t(m)\ a_{mi}$

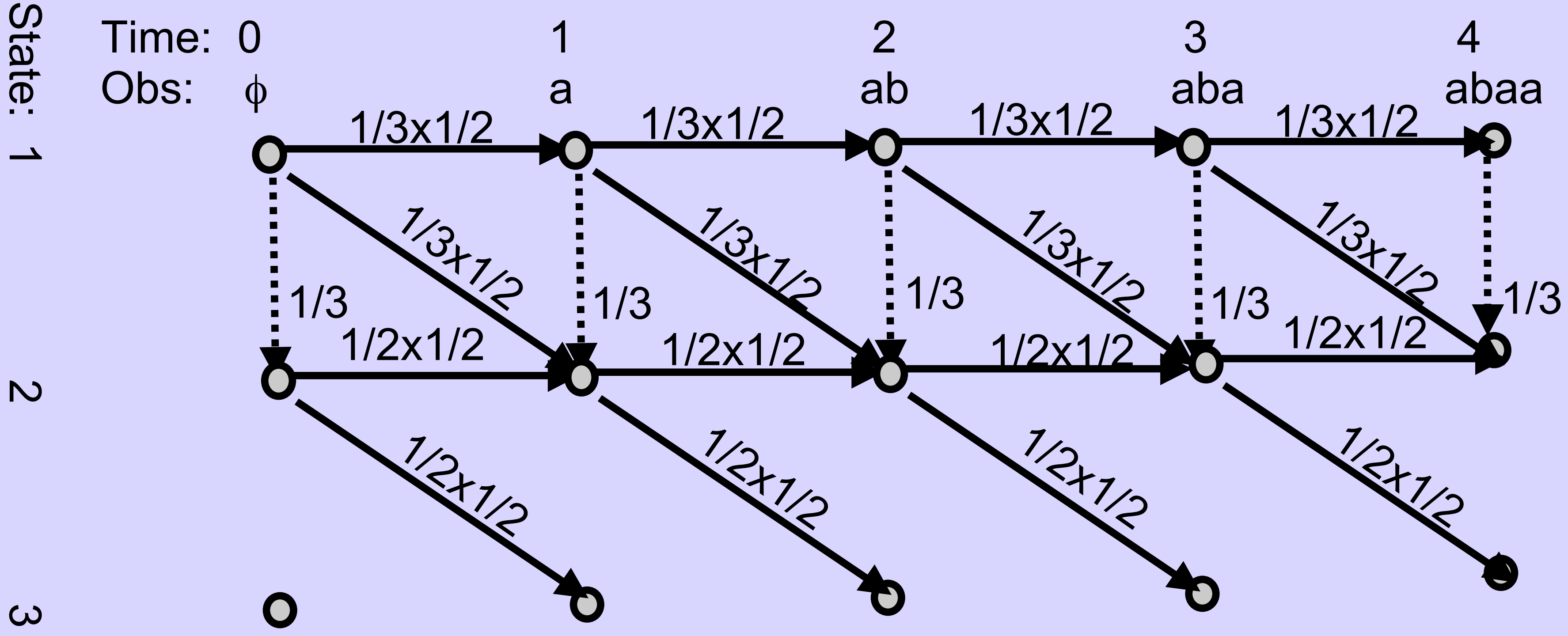
Note: the F-B algorithm is the same for Moore and Mealy forms, because of the way B's are defined -- they include the emission probabilty of t+1st transition (Mealy) / t+1st state (Moore).

Problem 3: F-B algorithm cont’d

- Let’s return to our previous example, and work out the trellis calculations

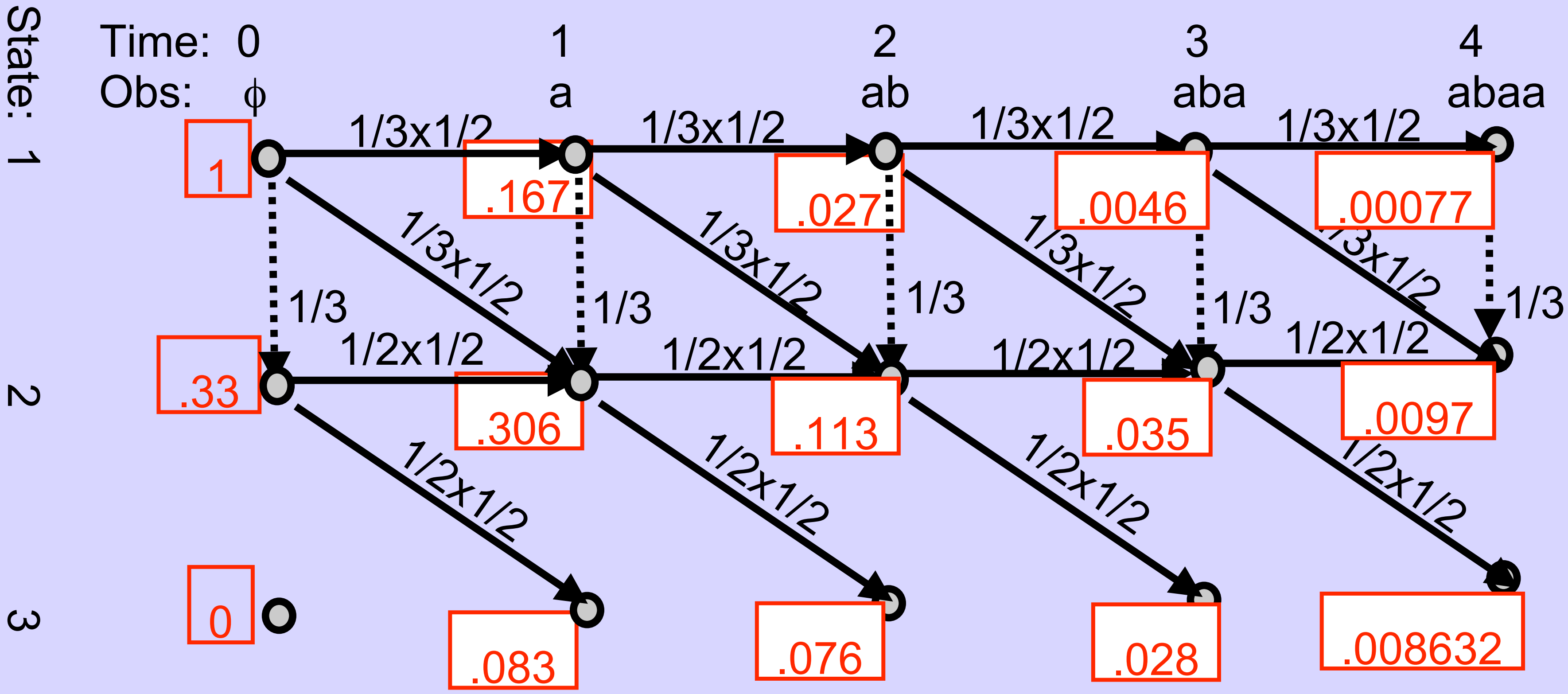


Problem 3: F-B algorithm, cont’d



Problem 3: F-B algorithm, cont'd

Compute α's. since forced to end at state 3, α_T=.008632=Pr(X)

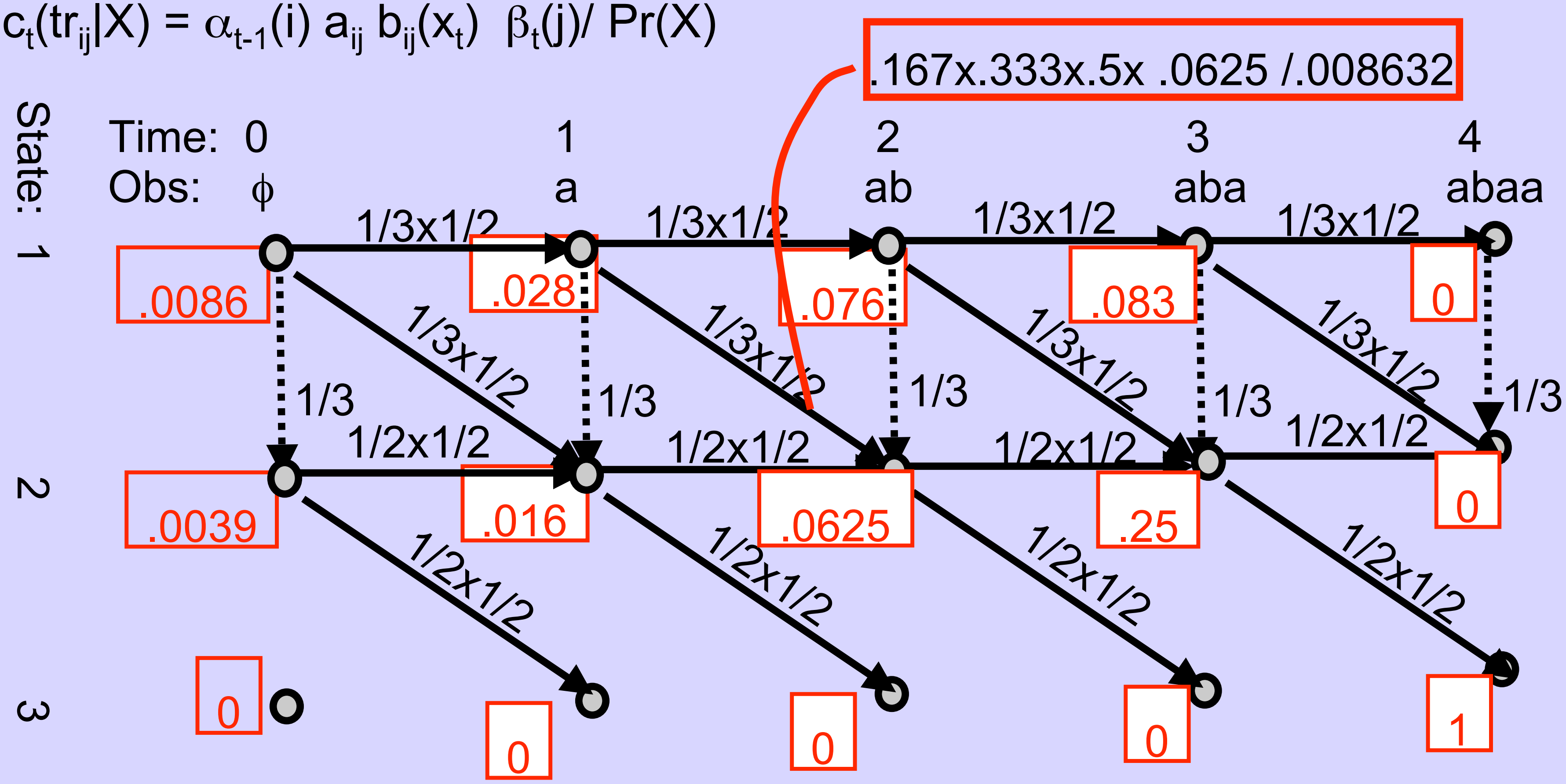


Problem 3: F-B algorithm, cont’d

Compute β’s.

Compute counts. (a posteriori probability of each transition)

$c_t(tr_{ij}|X) = \alpha_{t-1}(i)\ a_{ij}\ b_{ij}(x_t)\ \ \beta_t(j)/ \Pr(X)$



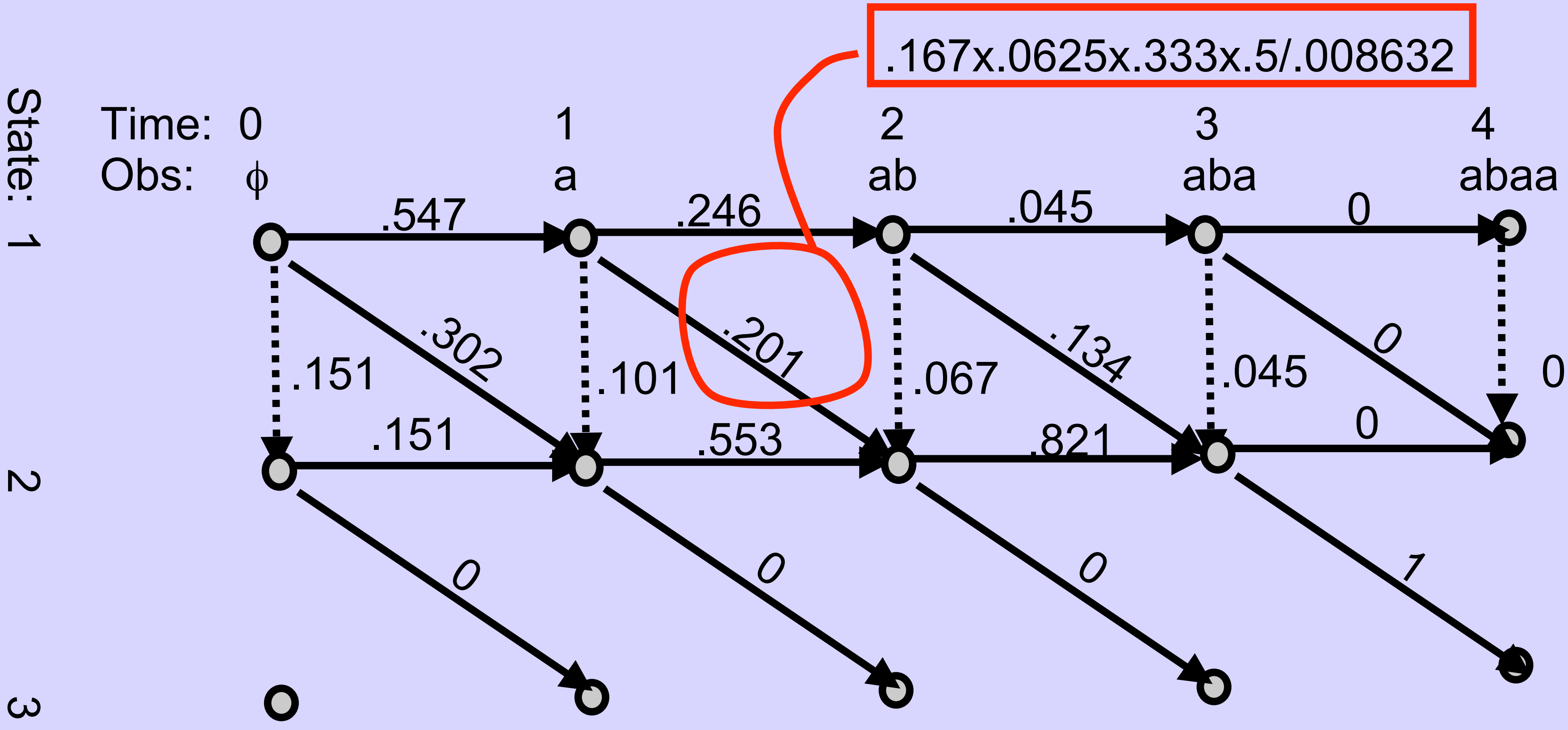
.167x.333x.5x .0625 /.008632

	Time: 0	1	2	3	4
	Obs: ϕ	a	ab	aba	abaa
State: 1	.0086	.028	.076	.083	0
2	.0039	.016	.0625	.25	0
3	0	0	0	0	1

Problem 3: F-B algorithm, cont’d

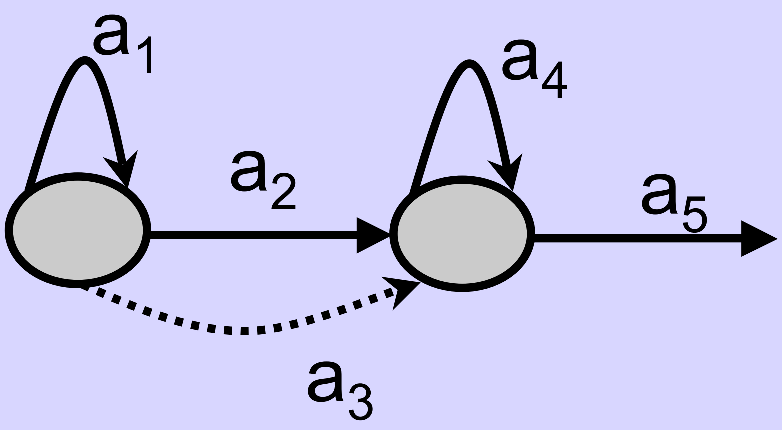
Compute counts. (a posteriori probability of each transition)

$$c_t(tr_{ij}|X) = \alpha_{t-1}(i)\ a_{ij}\ b_{ij}(x_t)\ \ \beta_t(j)/ \Pr(X)$$



.167x.0625x.333x.5/.008632

Problem 3: F-B algorithm cont'd

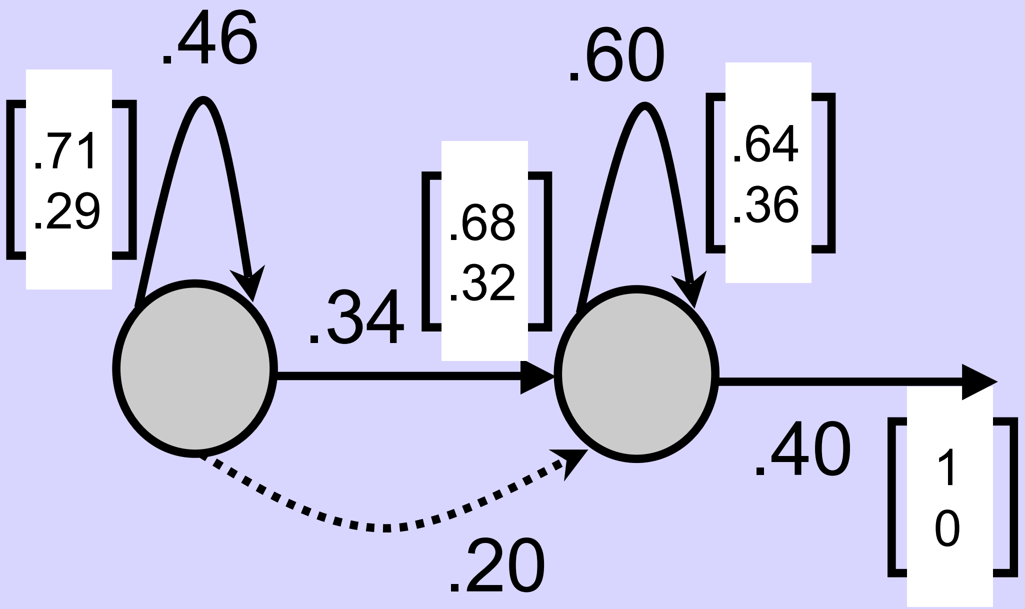


- $C(a_1)$=.547+.246+.045
- $C(a_2)$=.302+.201+.134
- $C(a_3)$=.151+.101+.067+.045
- $C(a_4)$=.151+.553+.821
- $C(a_5)$=1

- C(a1,'a')=.547+.045, C(a1,'b')=.246
- C(a2,'a')=.302+.134, C(a2,'b')=.201
- C(a4,'a')=.151+.821, C(a4,'b')=.553
- C(a5,'a')=1, C(a5,'b')=0

Problem 3: F-B algorithm cont'd

Normalize counts to get new parameter values.



Result is the same as from the enumerative algorithm!!

Continuous Hidden Markov models – Parameterization

Continuous Hidden Markov models (HMMs) have 3 sets of parameters.

1. A prior distribution over the states $\pi = P(s_0 = j); j = 1 \ldots N$

2. Transition probabilities between the states,
   $a_{ij} = P(s_t = j \mid s_{t-1} = i); i, j = 1 \ldots N$

3. A set of state-conditioned observation probabilities, $P(x_t \mid s_t = j); j = 1 \ldots N$
   The mixture of n-dimensional Gaussians is common:

$$P(x \mid \Theta) = \sum_{m=1}^{M_j} \frac{c_{jm}}{(2\pi)^{n/2} \mid \Sigma_{jm} \mid^{1/2}} e^{-\frac{1}{2}(x-\mu_{jm})^T \Sigma_{jm}{}^{-1}(x-\mu_{jm})}$$

Summary of Markov Modeling Basics

- Key idea 1: States for modeling sequences

Markov introduced the idea of state to capture the dependence on the past. A state embodies all the relevant information about the past. Each state represents an equivalence class of pasts that influence the future in the same manner.

- Key idea 2: Marginal probabilities

To compute Pr(X), sum up over all of the state sequences than can produce X

$$\Pr(X) = \Sigma_s \Pr(X,S)$$

For a given S, it is easy to compute Pr(X,S)

- Key idea 3: Trellis

The trellis representation is a clever way to enumerate all sequences. It uses the Markov property to reduce exponential-time enumeration algorithms to linear-time trellis algorithms.