Statistical Methods for NLP

Information Extraction, Hidden Markov Models

Sameer Maskey

Week 5, Oct 3, 2012

*many slides provided by Bhuvana Ramabhadran, Stanley Chen, Michael Picheny

Announcement

- Homework 1 Due Thursday Oct 4 (11:59pm)
- Project Intermediate Report I Due Oct 17 Wednesday (11:59pm)
- Homework 2 will be out Friday Oct 5.
- Homework 2 Due Oct 25 Thursday (11:59pm)

Topics for Today

- Information Extraction
- Hidden Markov Models

Information Extraction

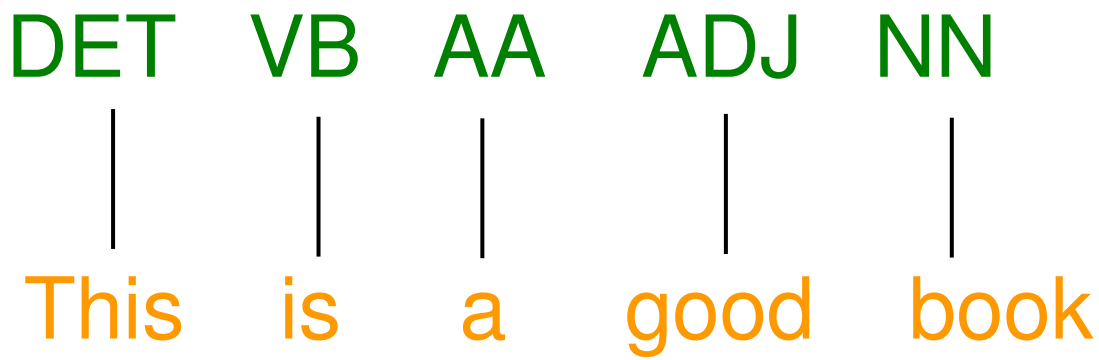
- Extract relevant information from large amount of unstructured text
- Extracted information can be in structured form
  - Can be used to populate databases for example
- We can define the kind of information we want to extract

Examples of Information Extraction Tasks

- Named Entity Identification
- Relation Extraction
- Coreference resolution
- Term Extraction
- Lexical Disambiguation
- Event Detection and Classification

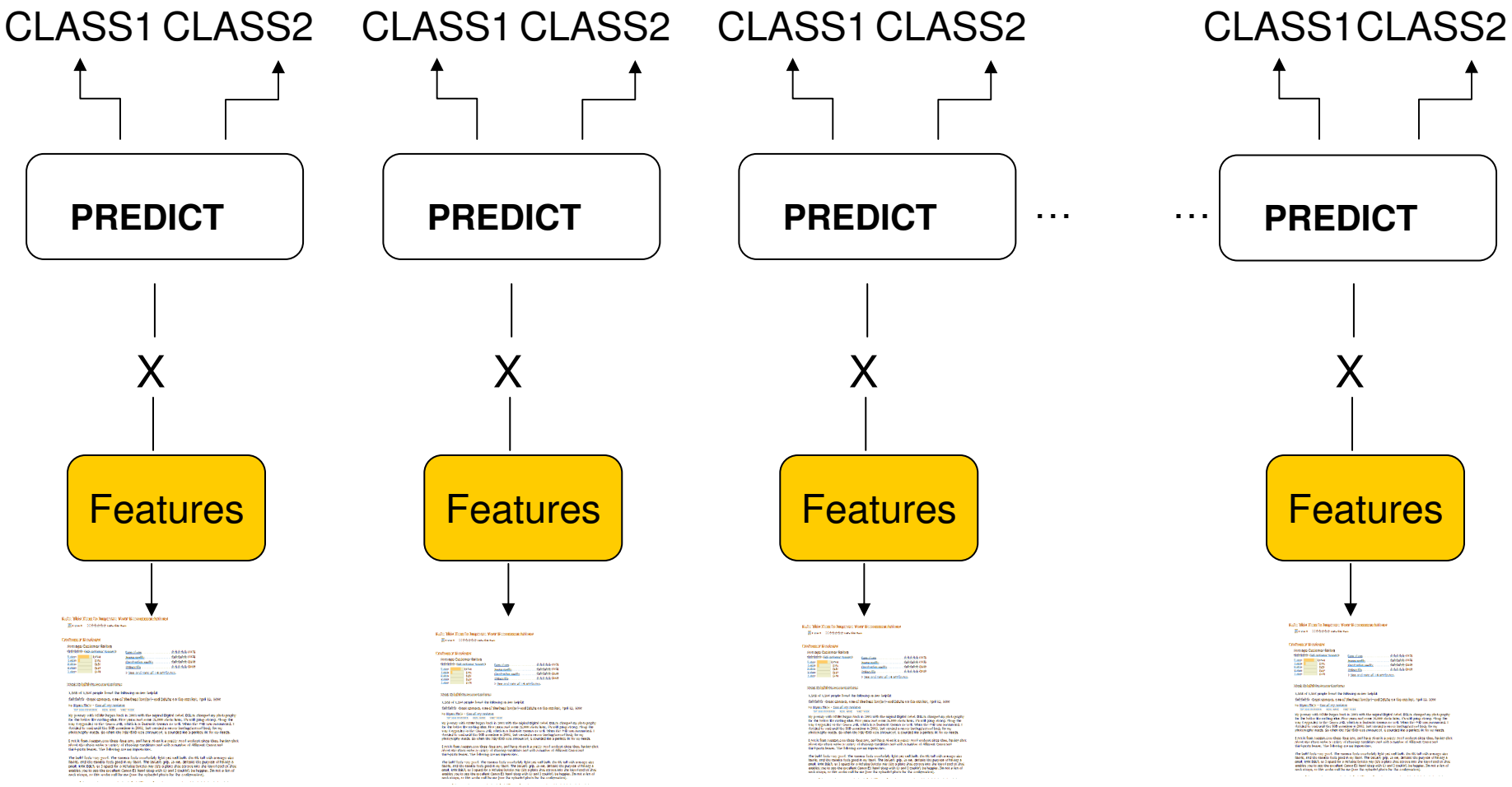
Classifiers for Information Extraction

- Sometimes extracted information can be a sequence
- Extract Parts of Speech for the given sentence

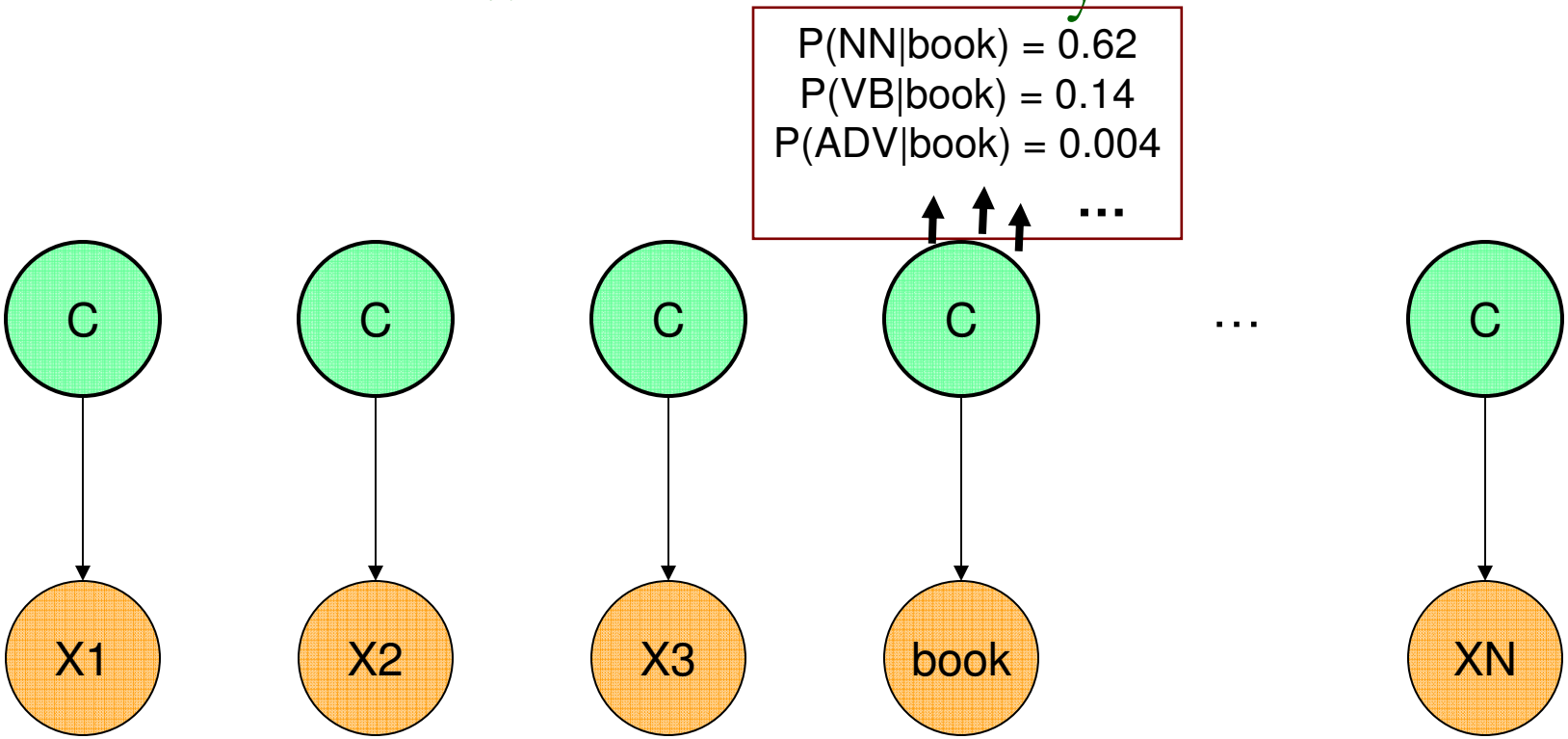


- What kind of classifier may work well for this kind of sequence classification?

Classification without Memory



Classification without Memory



- C(t) (class) is dependent on current observations X(t)
- C(t) can be POS tags, document class, word class, X(t) can be text based features
- Perceptron is an example of classifier without memory

Probability Sequence Computation for Models without Memory

- A coin has probability of "heads" = p , probability of "tails" = 1-p
- Flip the coin 10 times. Assume I.I.D. random sequence. There are 2^{10} possible sequences.
- Sequence: 1 0 1 0 0 0 1 0 0 1
  Probability: $p(1-p)p(1-p)(1-p)(1-p)\ p(1-p)(1-p)p = p^4(1-p)^6$
  - **Models without memory:** Observations are Independent. 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}

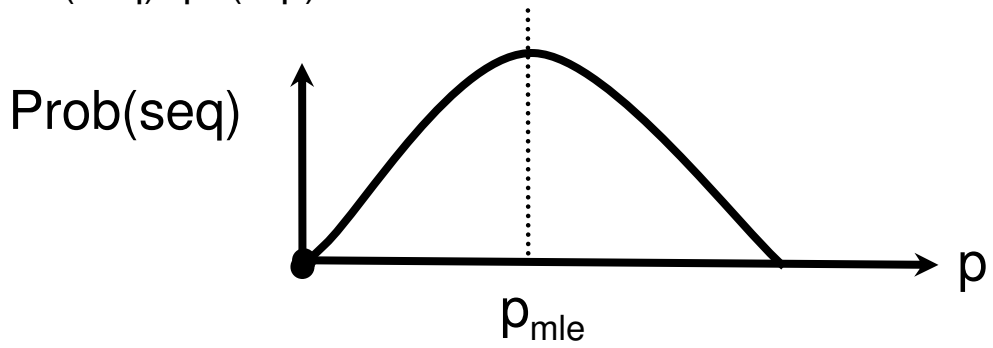
Models without Memory: Learning Model Parameters

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 actually the "best" choice for p?

What is "best"? Consider the probability of the observed sequence.
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})

Models without Memory: Example, cont'd

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

Proof: $\text{Prob(seq)} = p^H(1-p)^T$

Maximizing Prob is equivalent to maximizing log(Prob)

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

$$\frac{\partial L}{\partial p} = H/p + T/(1-p)$$

L maximized when $\frac{\partial L}{\partial p} = 0$

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

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

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

$p_{mle} = H/N$

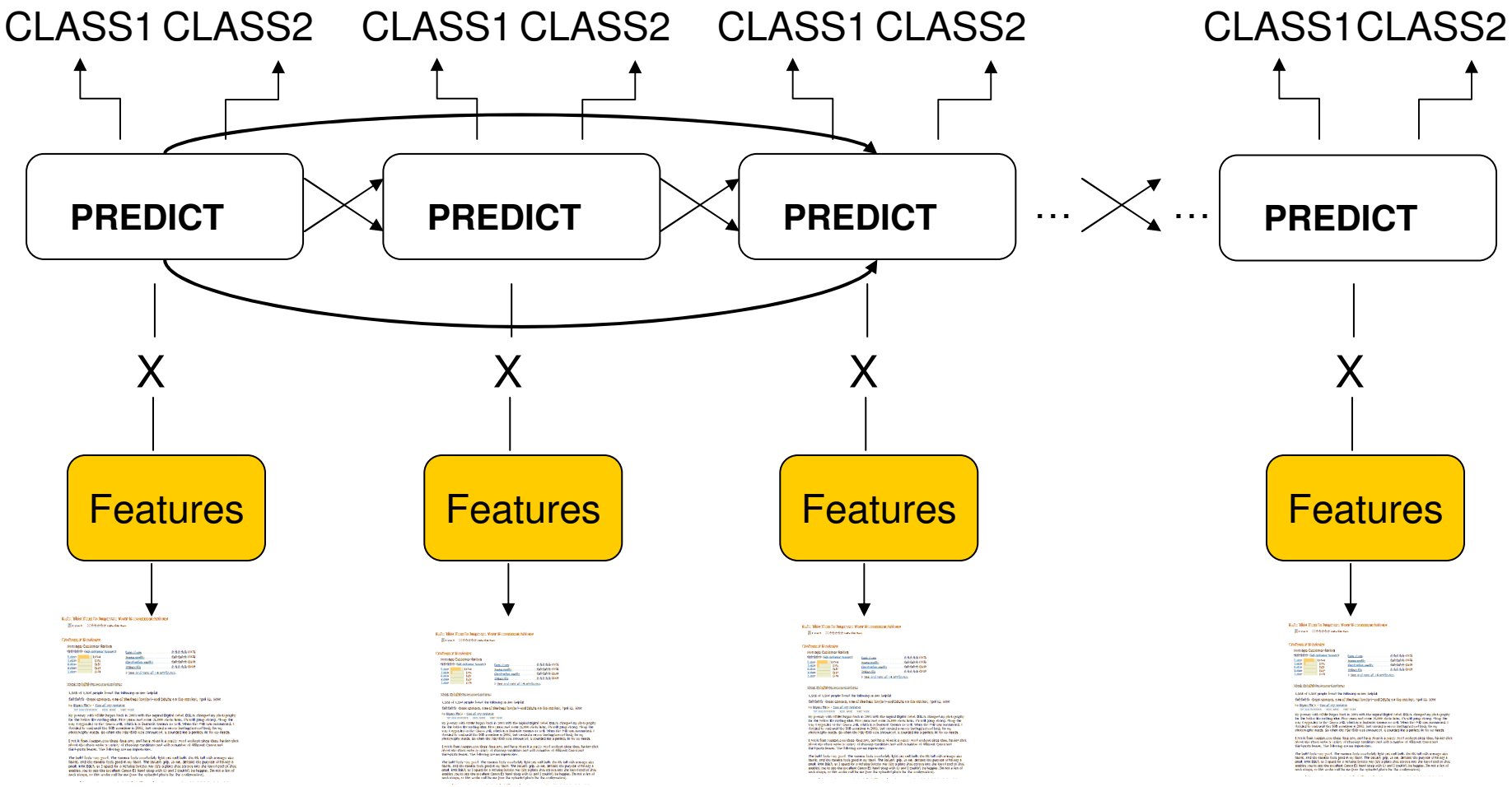
Models without Memory Example, cont'd

- 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.

Models with Memory: Markov Models

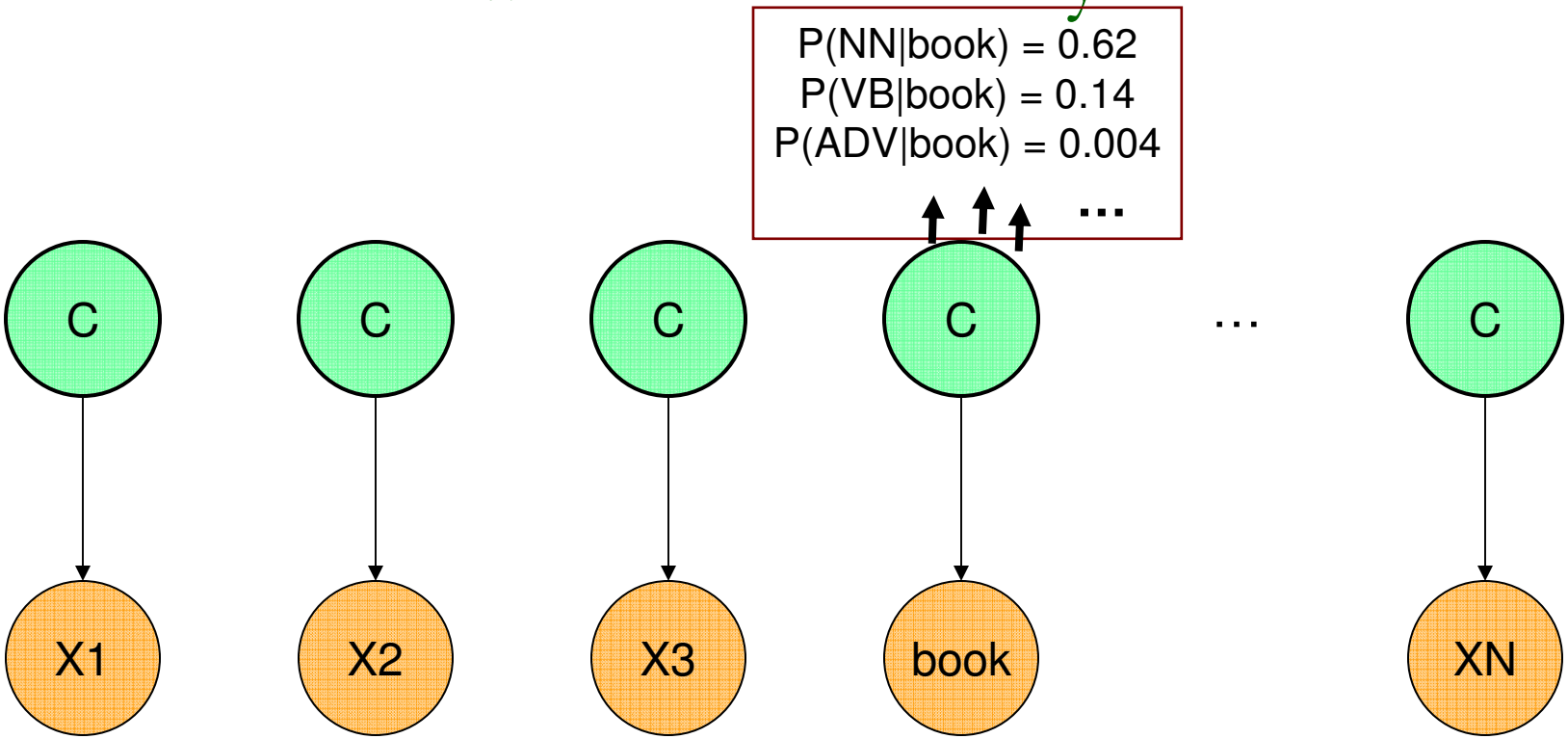
- Flipping a coin was memory-less. The outcome of each flip did not depend on the outcome of the other flips.
- Adding memory to a memory-less model gives us a Markov Model. Useful for modeling sequences of events.
- For POS tagging adding memory to classifier could be useful

Classification with Memory



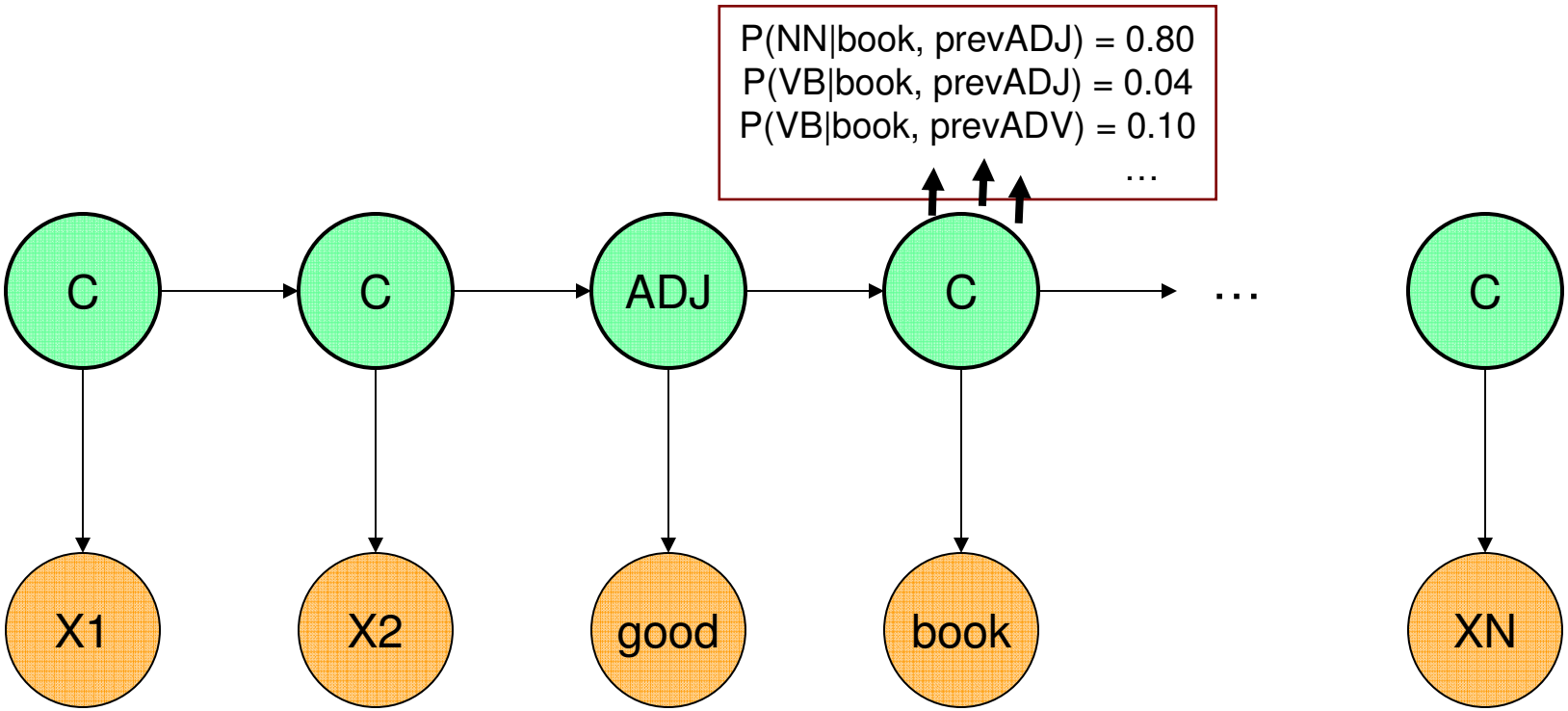
Current Prediction depends only previous predictions and current observation

Classification without Memory



- C(t) (class) is dependent on current observations X(t)
- C(t) can be POS tags, document class, word class, X(t) can be text based features
- Perceptron is an example of classifier without memory

Classification with Memory



- C(t) (class) is dependent on current observations X(t) and previous state of classification (C(t-1))
- C(t) can be POS tags, document class, word class, X(t) can be text based features

Sequential Stochastic Models

- We can add memory to learning model by adding dependencies across classification labels over time
- Probabilistic models that can model such dependencies across time is useful for many tasks
  - Information Extraction
  - Speech Recognition
  - Computational Biology
- We can build Markov model for underlying sequence of labels and associate the observations with each state

Adding Memory to Coin Example

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

ADDING MEMORY

- 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 = .0648
  H T H T prob =
  0.9 x 0.1 x 0.2 x 0.1 = .0018

what would be probability if no memory was added?

- Sequences with consecutive heads or tails are more likely.
- The sequence has memory - order matters.
- Order matters for language.
  - Adjective noun probably more common than adjective adjective

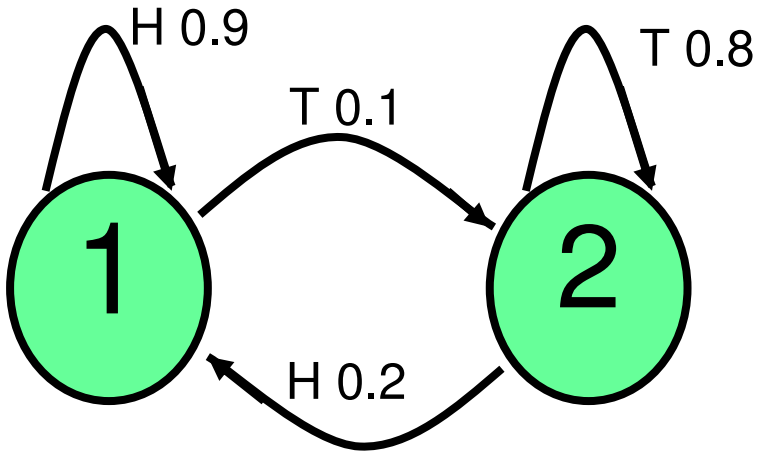
Markov Models – State Space Representation

- Consider 2 coins.

  Coin 1: $p_H = 0.9$, $p_T = 0.1$

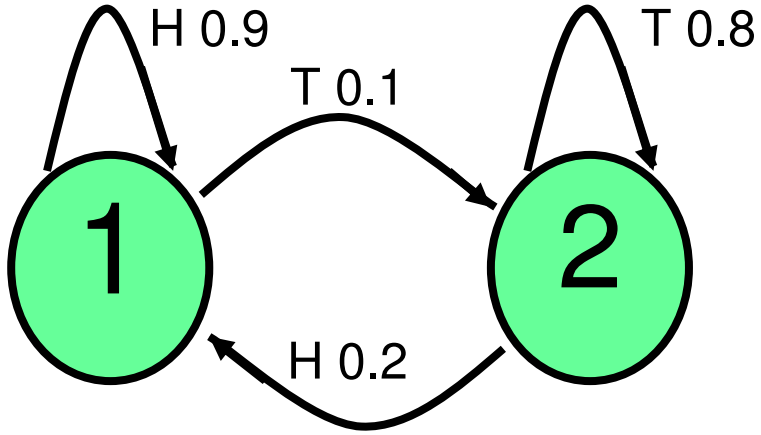
  Coin 2: $p_H = 0.2$, $p_T = 0.8$

State-space representation of previous example



Markov Models – State Space Representation (Con't)

- 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

Back to Memory-Less Models: Hidden Information

Let's return to the memory-less coin flip model

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

ADDING HIDDEN VARIABLE

Experiment:

For J=1..4

Flip coin 0. If outcome == "H"

Flip coin 1 and record.

else

Flip coin 2 and record.

Hiding Information (cont.)

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 x 0.7 + 0.8 x 0.3 = 0.31

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

Hidden Information + Markov Model

We added Memory

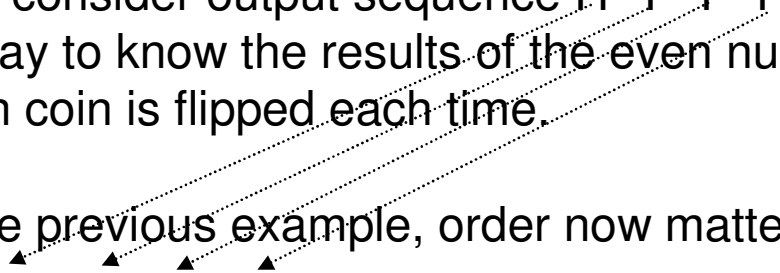
We introduced Hidden Variable

Hidden Markov Model

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

- Experiment:
  Flip 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.

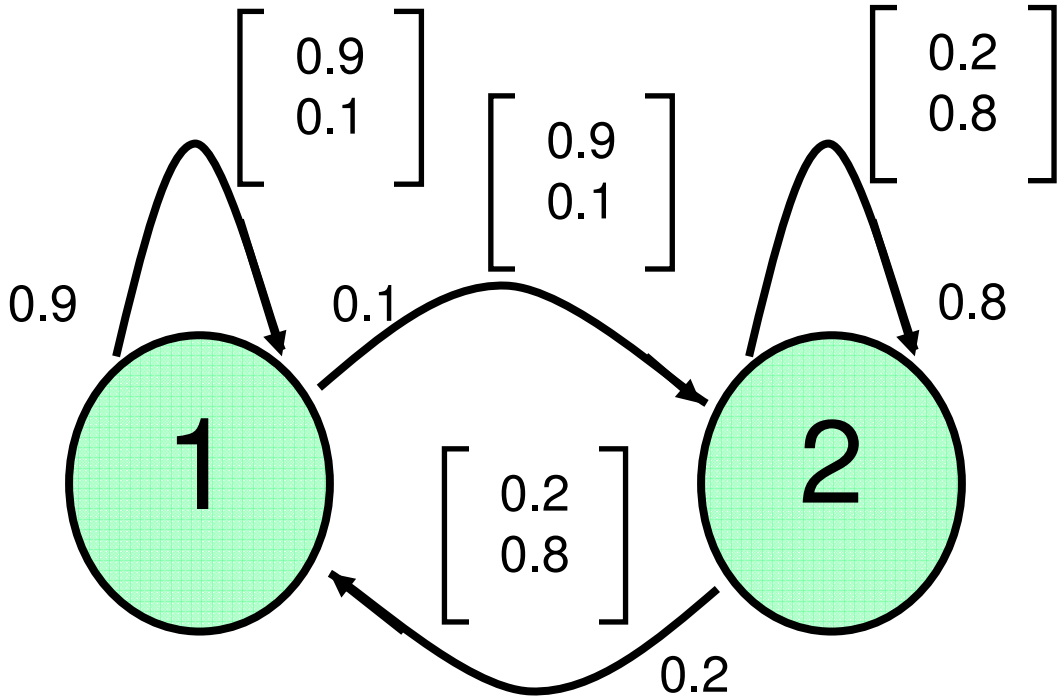
- Now, consider output sequence H T T T.
- No way to know the results of the even numbered flips, so no way to know which coin is flipped each time.



- Unlike previous example, order now matters (start with coin 1, $p_H = 0.9$)
  - H H T T T H T = .9 x .9 x .1 x .1 x .8 x .2 x .1 = .0001296
  - T T T H T T H = .1 x .1 x .8 x .2 x .1 x .1 x .2 = .0000032
- Even worse, same output sequence corresponds to multiple probabilities!
  - H T T H T T T = .9 x .1 x .8 x .2 x .1 x .1 x .8 = .0001152

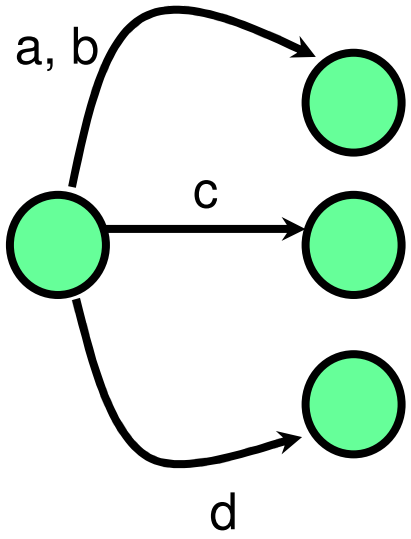
Hidden Markov Model

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

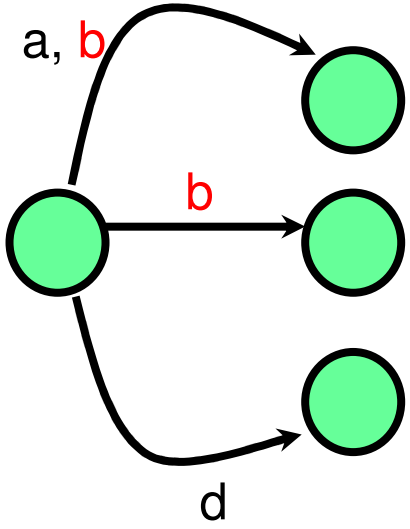


Is a Markov Model Hidden or Not?

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



State-observable



Hidden

Three problems of general interest for an HMM

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

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

- 2. Given X, find the most likely state sequence (Viterbi algorithm)

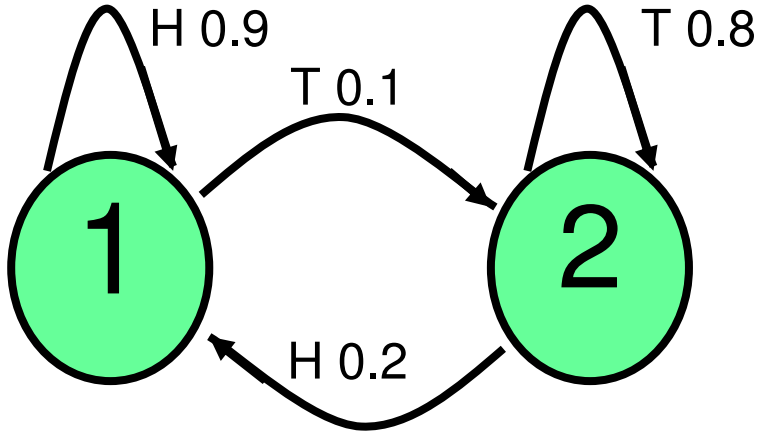
3. Estimate the parameters of the model (training)

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

1. Given an observed output sequence $X=x_1x_2..x_T$, compute $P_\theta(X)$ 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

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

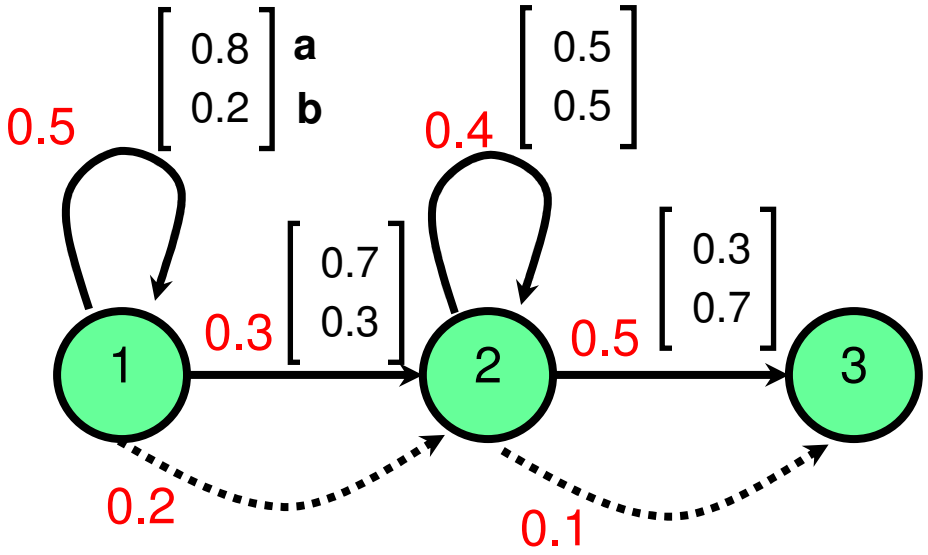
Sum over all possible state sequences:

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

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

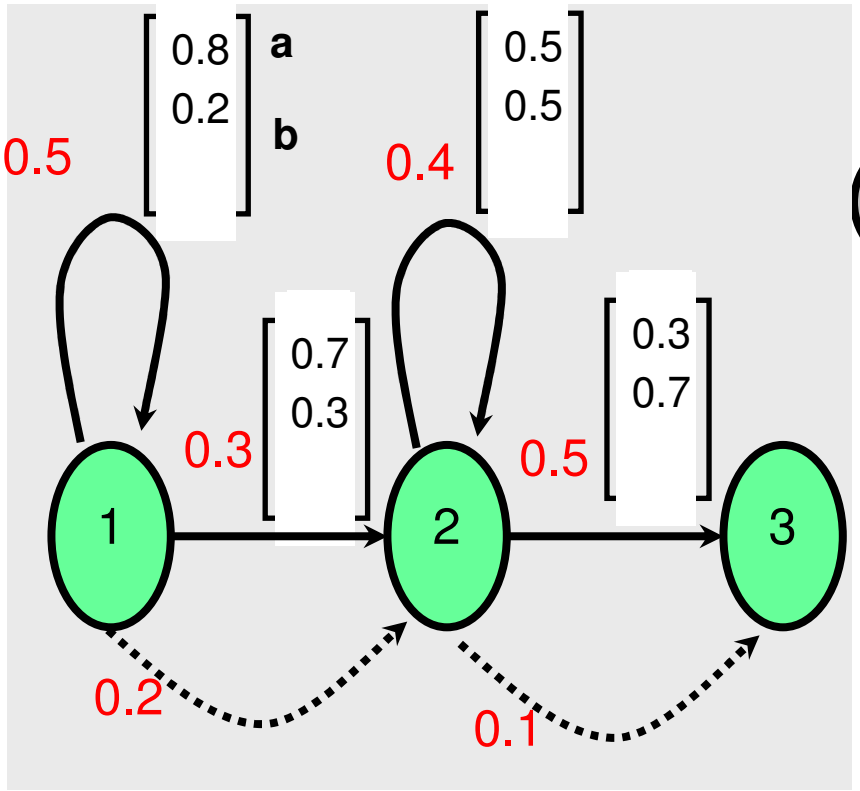
Unfortunately, this calculation is exponential in the length of the sequence

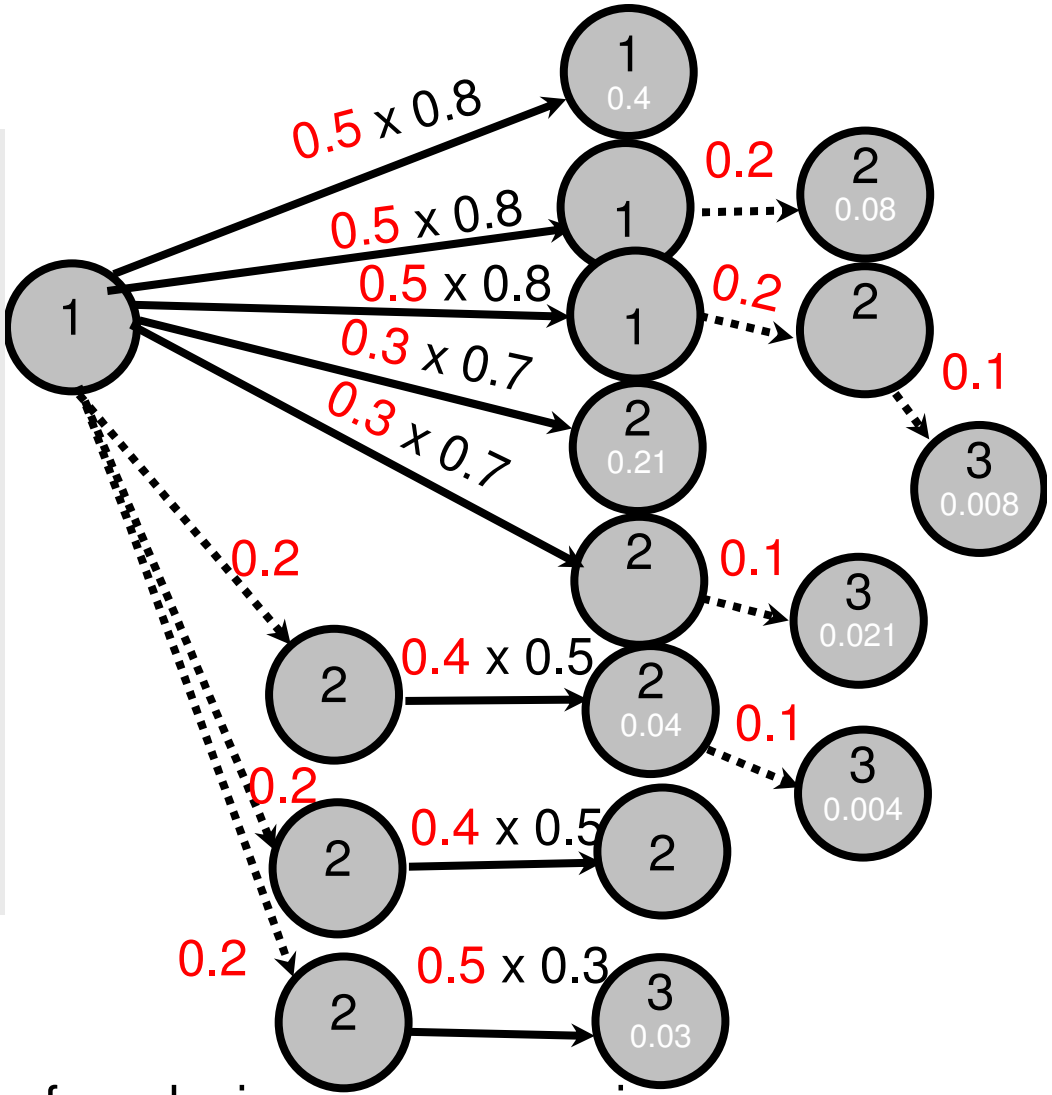
Example for Problem 1



Compute $P_\theta(X)$ for X=aabb, assuming we start in state 1

Example for Problem 1,cont'd

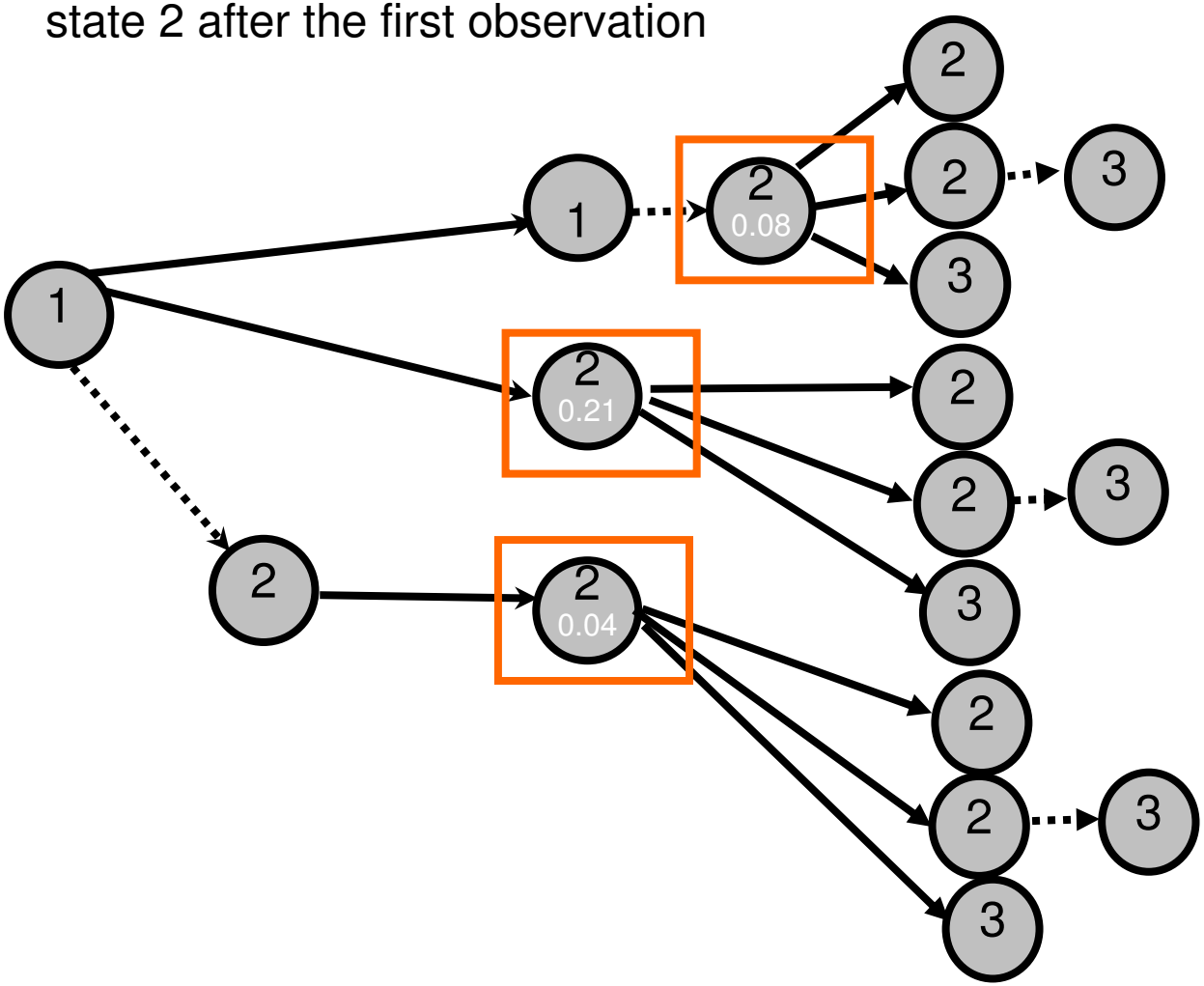




Let's enumerate all possible ways of producing x_1=a, assuming we start in state 1.

Example for Problem 1, cont'd

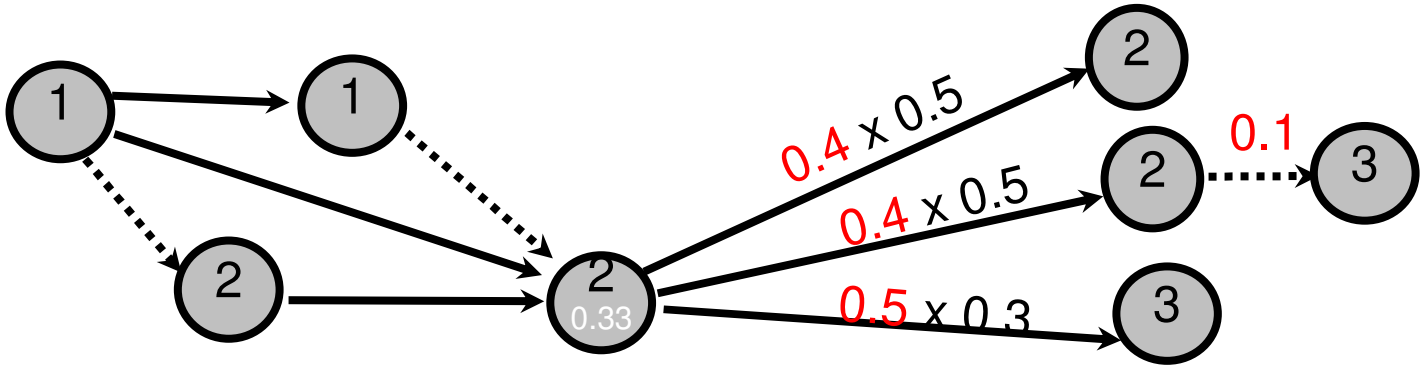
- Now let's think about ways of generating x1x2=aa, for all paths from state 2 after the first observation



Example for Problem 1,cont'd

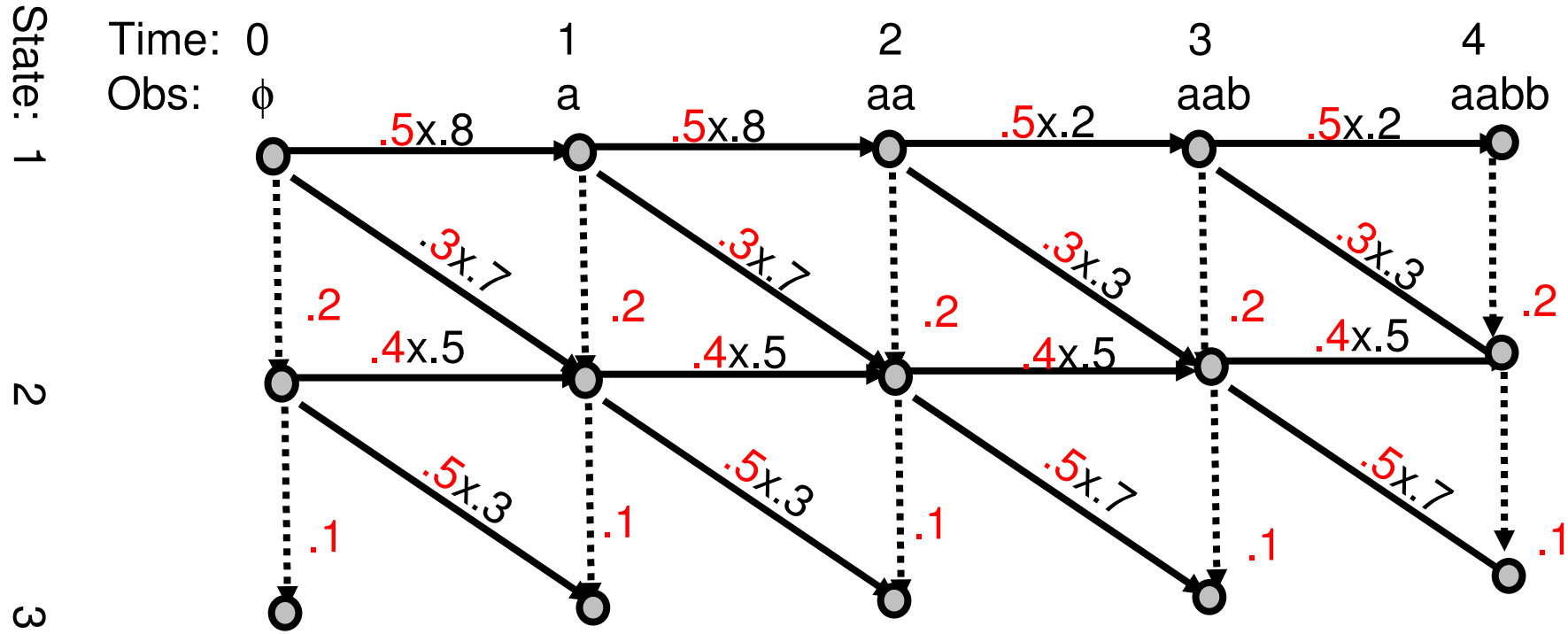
We can save computations by combining paths.

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



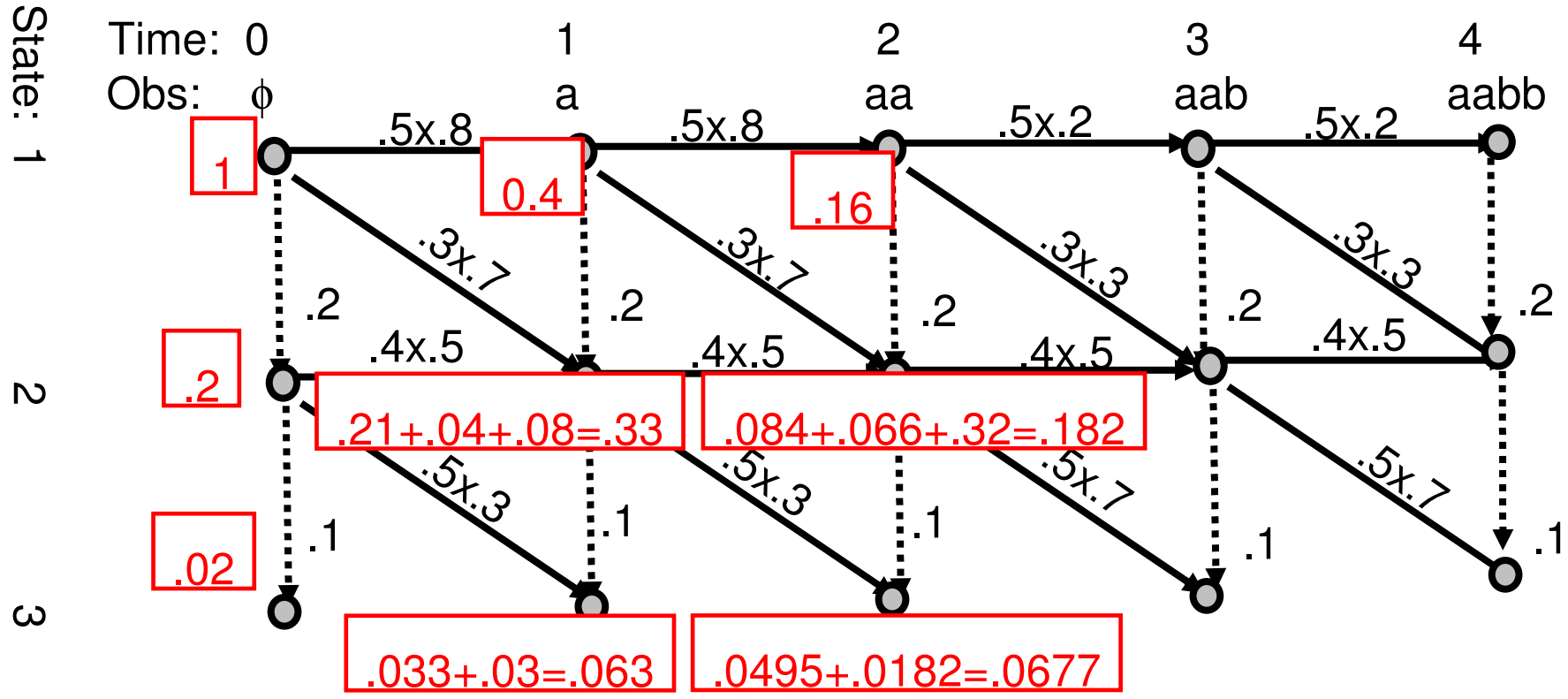
Problem 1: Trellis Diagram

- 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.

Problem 1: Forward Pass Algorithm

Let $\alpha_t(s)$ for t ε {1..T} be the probability of being in state s at time t and having produced output $x_1^t = x_1..x_t$

$$\alpha_t(s) = \sum_{s'} \alpha_{t-1}(s') P_\theta(s|s') P_\theta(x_t|s'->s) + \sum_{s'} \alpha_t(s') P_\theta(s|s')$$

1st term: sum over all output producing arcs 2nd term: all null arcs

This is called the Forward Pass algorithm.

This calculation allows us to solve Problem 1 efficiently:

$$P(x_1, x_2, ..., x_T; \theta) = \sum_s \alpha_T(s)$$

N^2 * T

$$P(x_1, x_2, ..., x_T; \theta) = \sum_{s_1,...S_T} P(x_1, x_2, ..., x_T, s_1, s_2, ..., s_T)$$

N^T

Problem 2

Given the observations X, find the most likely state sequence

This is solved using the Viterbi algorithm

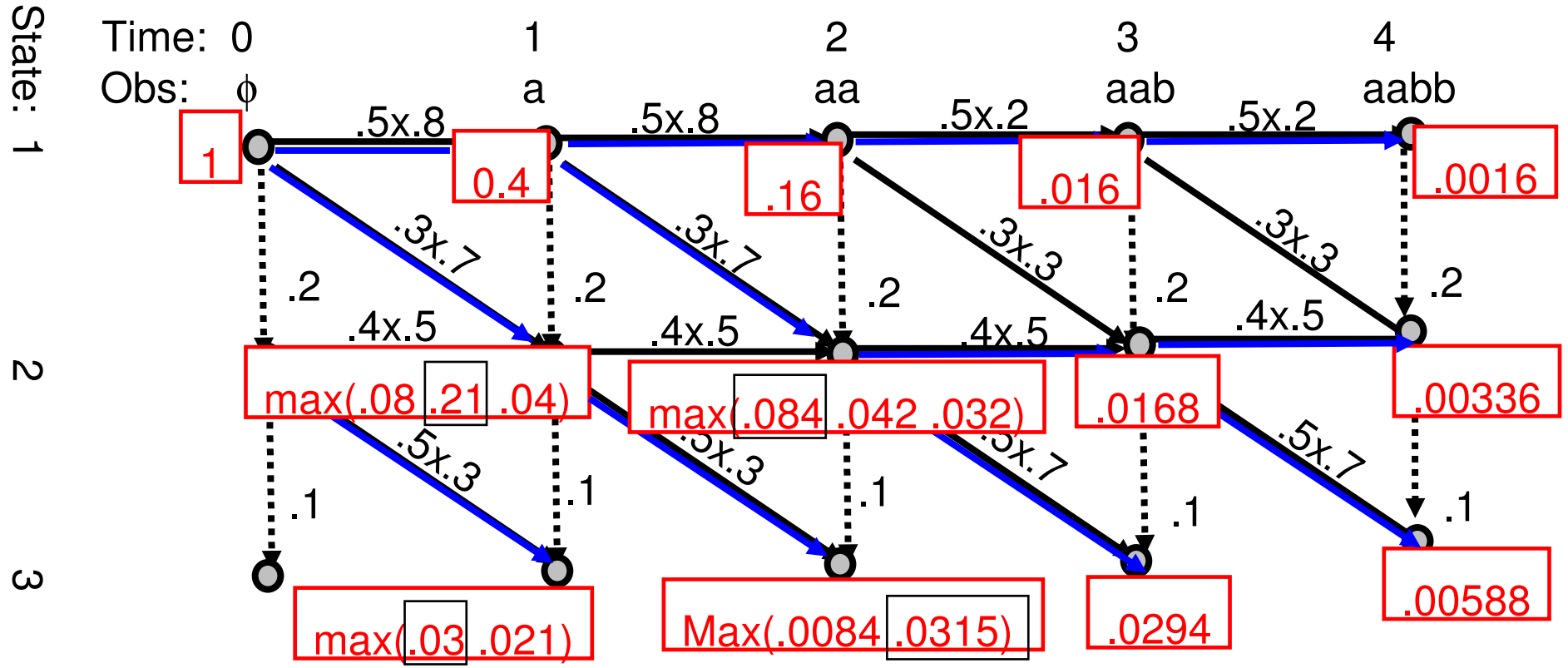
Preview:

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

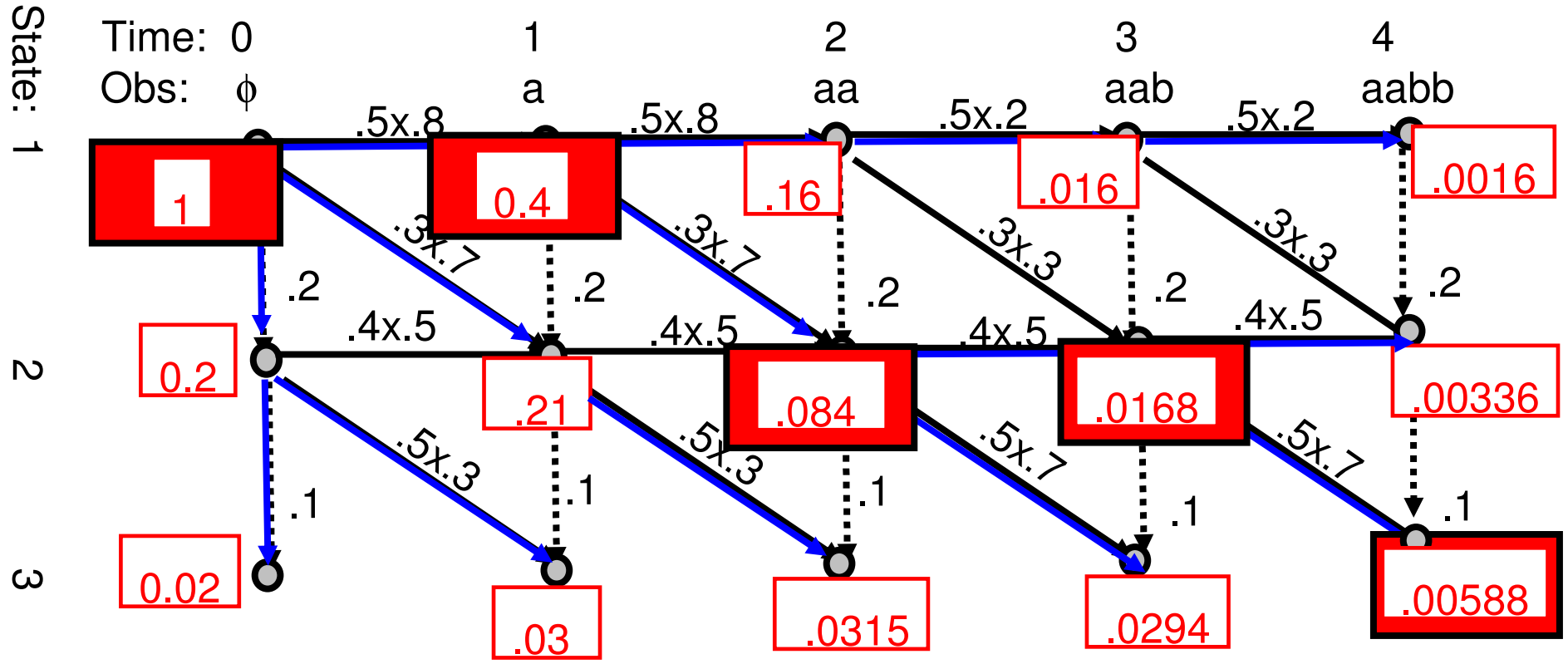
Problem 2: Viterbi algorithm

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.



Hidden Markov Models

- State : $Q = q_1 q_2 q_N$
- Transition Probabilities $T = a_{11} a_{12} a_{nn}$
- Emission Probabilities $B = b_i(o_t)$
- Observation Sequence $O = o_1 o_2 o_T$
- Start and Final State q_0, q_F

Markov Model with 5 states with 10 possible observation in each state will have T and B of what sizes?

Three problems of general interest for an HMM

3 problems need to be solved for HMM's:

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

$$P(x_1, x_2, , x_T; \theta)$$

- Given X, find the most likely state sequence (Viterbi algorithm)

find best $\hat{S}_1, \ldots, \hat{S}_T$ using $\hat{x}_1, \ldots, \hat{x}_T$

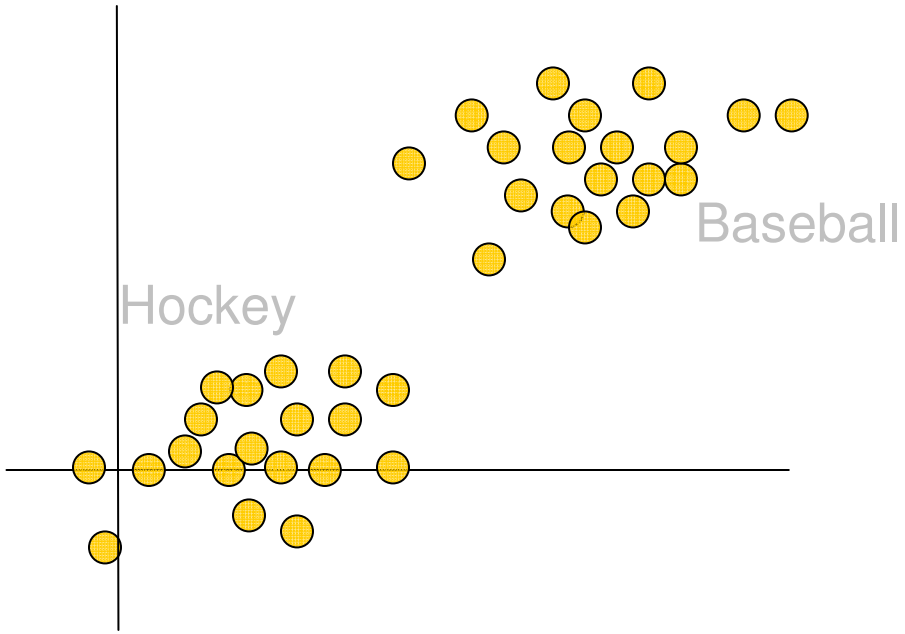
- Estimate the parameters of the model (training) using n observed sequences of varying length

$$q(S_t|S_{t-1}), \; b_i(o_t|S_t)$$

Detour: Unsupervised Learning

- Given the training data with class labels we saw we can compute Maximum Likelihood estimate for Naïve Bayes by getting relative frequencies of the word in the class
- When we do not have labels we can run E-M algorithm
  - E step – compute fractional counts
  - M step – maximize based on the fractional counts

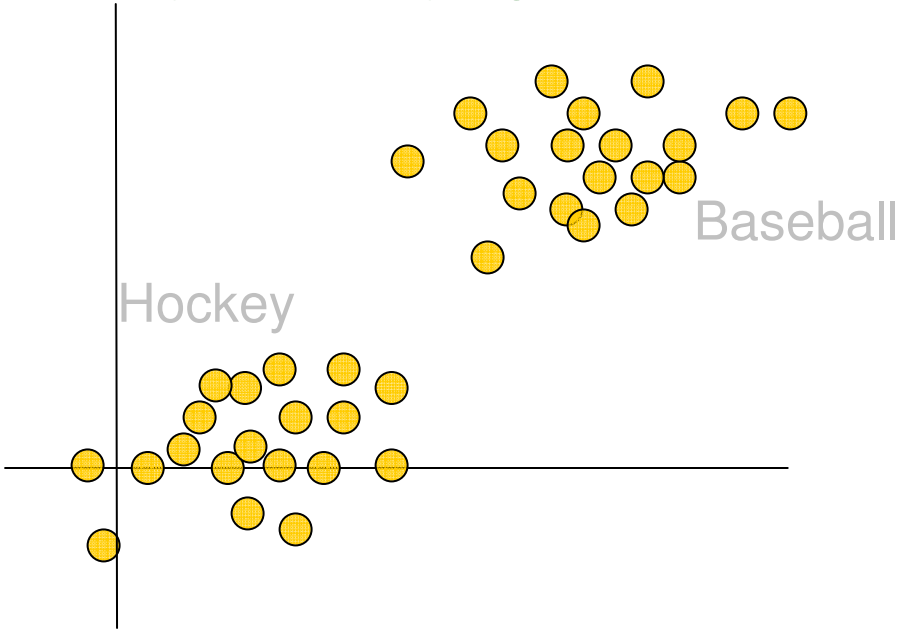
Classification with Hidden Variables



~ Do not know the class labels
~ Treat class labels as hidden variables
~ Maximize log-likelihood of unlabeled training data
~ Fractional counts can be thought of as assignment to a given class

Explaining Expectation Maximization

- EM is like fuzzy K-means
- Parameters to estimate for K classes
- Let us assume we can model this data with mixture of two Gaussians



- Start with 2 Gaussians (initialize mu and sigma values)
- Compute distance of each point to the mu of 2 Gaussians and assign it a soft class label (Ck)
- Use the assigned points to recompute mu and sigma for 2 Gaussians; but weight the updates with soft labels

Expectation

Maximization

The Baum-Welch algorithm

The Baum-Welch algorithm is a generalized expectation-maximization algorithm for computing maximum likelihood estimates for the parameters of a Hidden Markov Model when given only observations as training data.

It is a special case of the EM algorithm for HMMs.

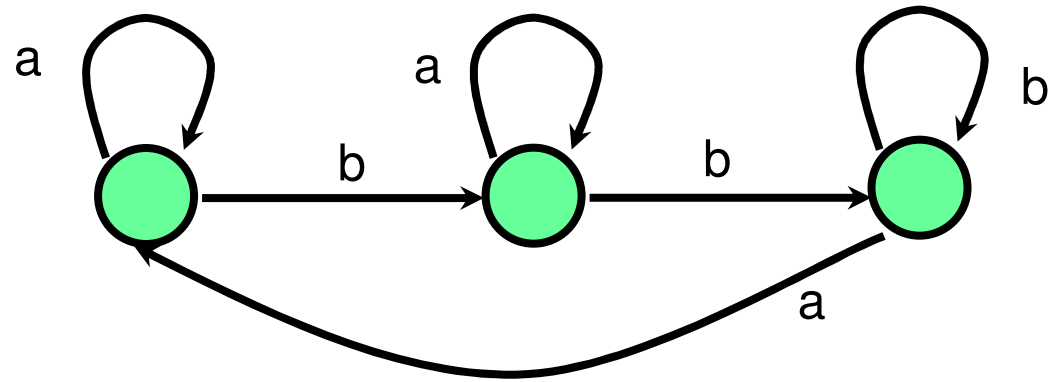
Problem 3

Estimate the parameters of the model. (training)

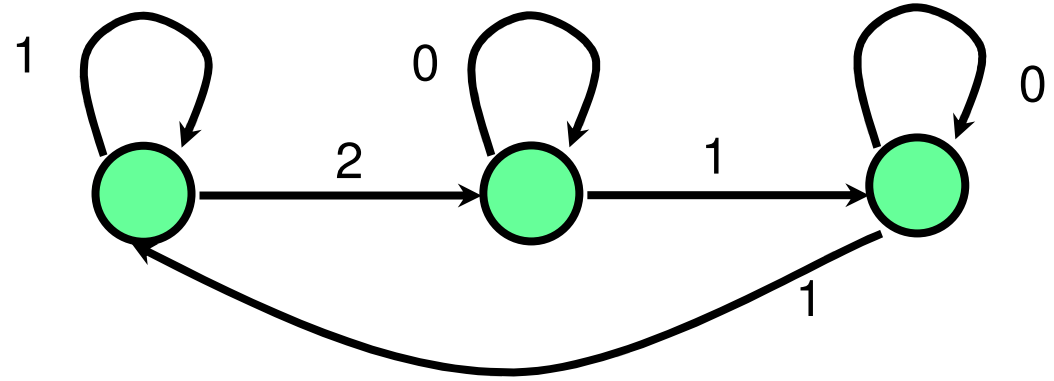
- Given a model topology and an output sequence, find the transition and output probabilities such that the probability of the output sequence is maximized.

Problem 3 – State Observable Example

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

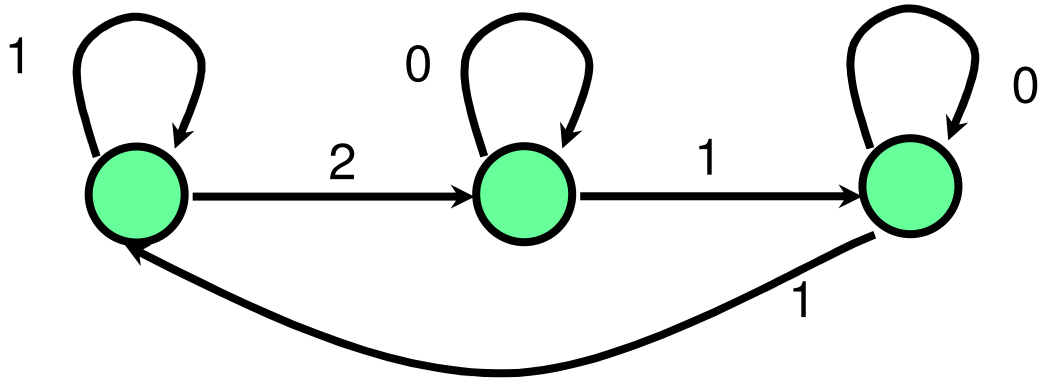


- Observed counts along transitions:

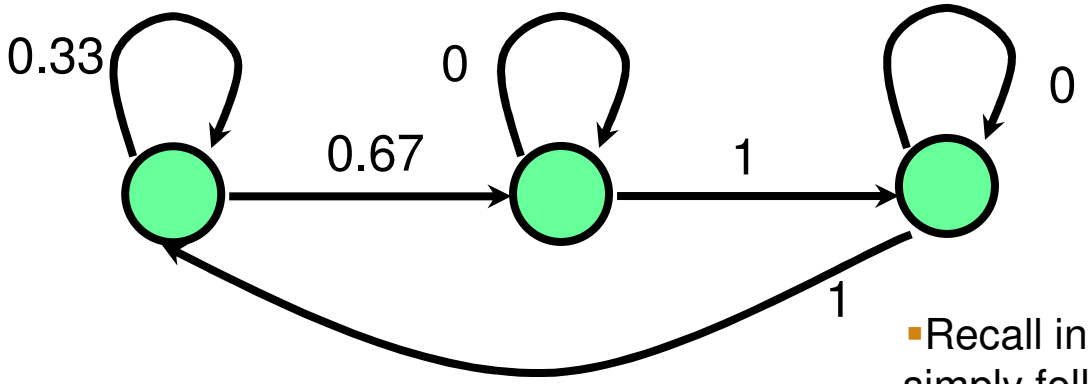


Problem 3 – State Observable Example

Observed counts along transitions:



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



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

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

Estimating Transition and Emission Probabilities

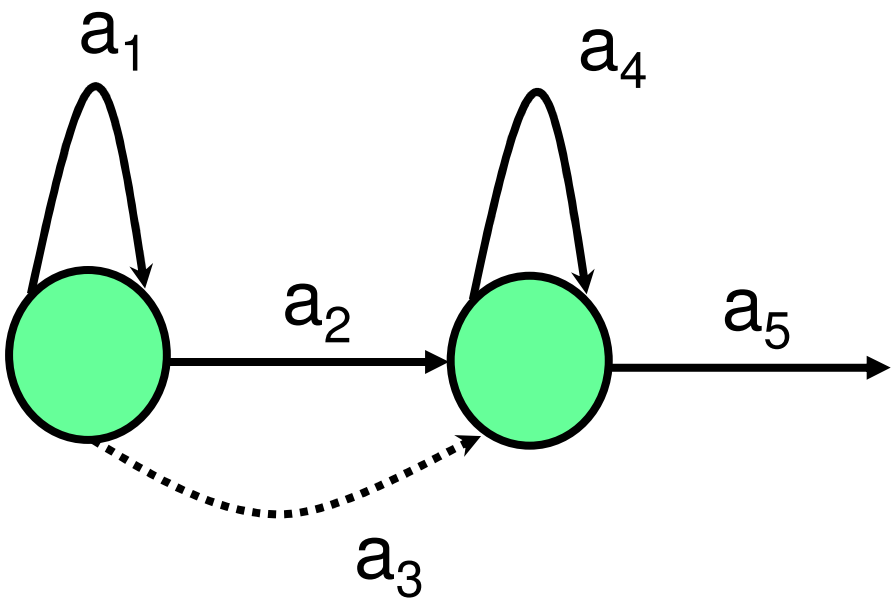
$$a_{ij} = \frac{count(i \rightarrow j)}{\sum_{q \in Q} count(i \rightarrow q)}$$

$$\hat{a}_{ij} = \frac{\text{Expected number of transitions from state i to j}}{\text{Expected number of transitions from state i}}$$

$$\hat{b}_j(x_t) = \frac{\text{Expected number of times in state j and observing symbol xt}}{\text{Expected number of time in state j}}$$

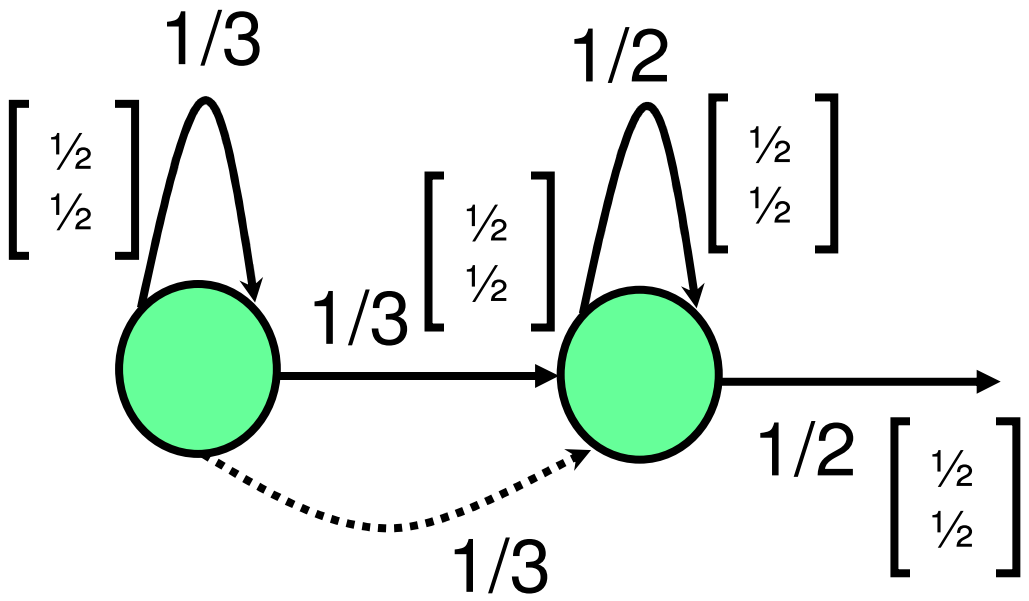
Problem 3: Enumerative Example – Assigning fractional counts

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

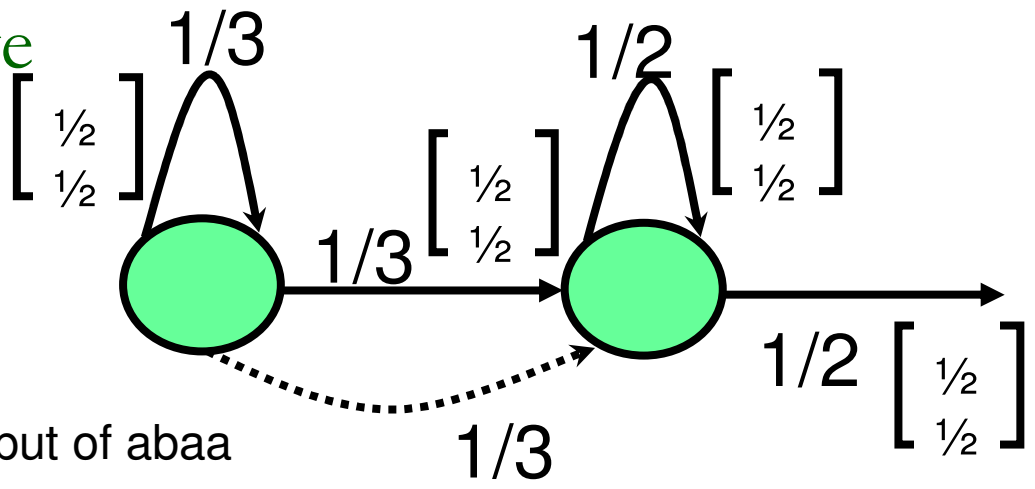


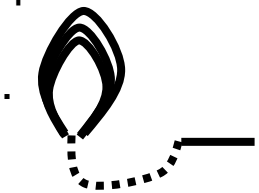
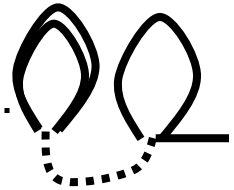
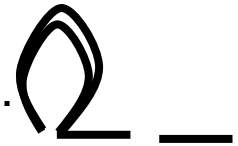
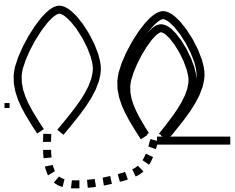
Problem 3: Enumerative Example - Assigning fractional counts

- Initial guess: equiprobable

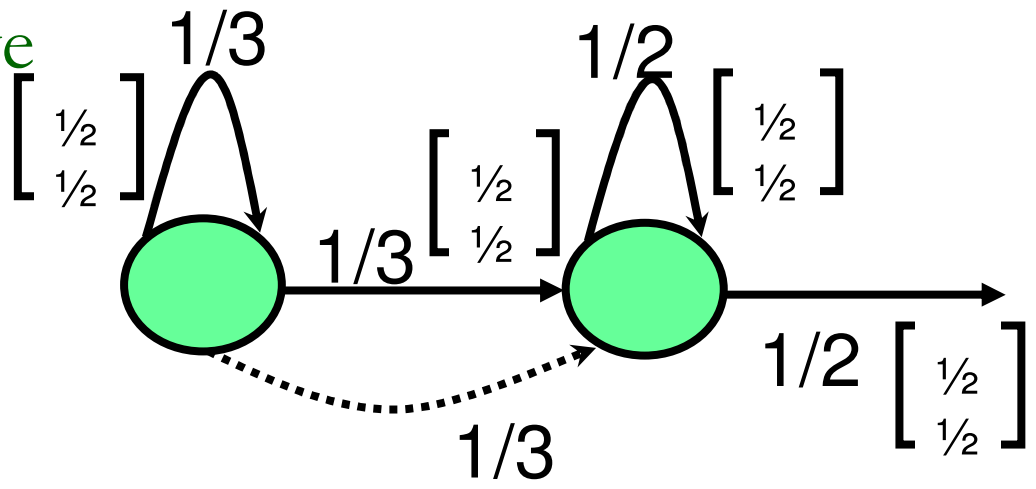


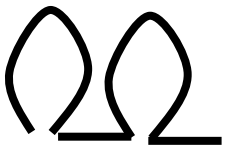
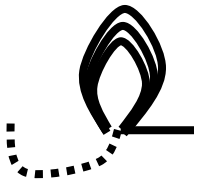
Problem 3: Enumerative Example cont'd



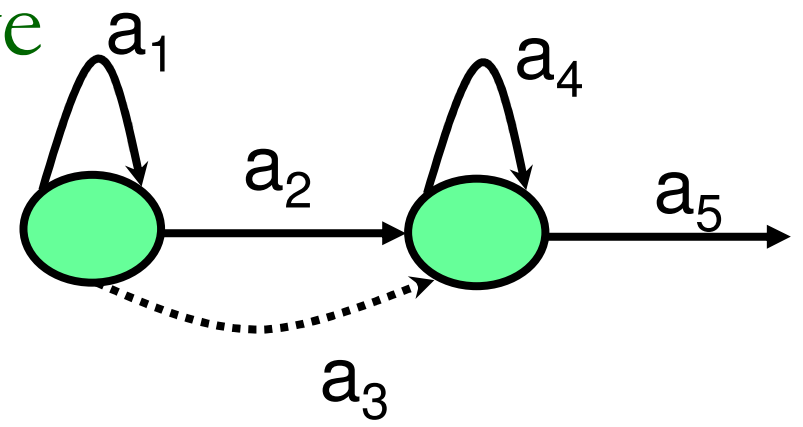
- 7 paths corresponding to an output of abaa
- 1.  pr(X,path1)=1/3x1/2x1/3x1/2x1/3x1/2x1/3x1/2x1/2=.000385
- 2.  pr(X,path2)=1/3x1/2x1/3x1/2x1/3x1/2x1/2x1/2x1/2=.000578
- 3.  pr(X,path3)=1/3x1/2x1/3x1/2x1/3x1/2x1/2x1/2=.001157
- 4.  pr(X,path4)=1/3x1/2x1/3x1/2x1/2x1/2x1/2x1/2x1/2=.000868

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)=1/3x1/2x1/2x1/2x1/2x1/2x1/2x1/2x1/2=.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
- $C_i = pr(X,path_i)/pr(X)$
-
- $C_1 = .045$ $C_2 = .067$ $C_3 = .134$ $C_4 = .100$ $C_5 = .201$ $C_6 = .150$ $C_7 = .301$

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

a1= C(a1)/{C(a1) + C(a2) + C(a3)}

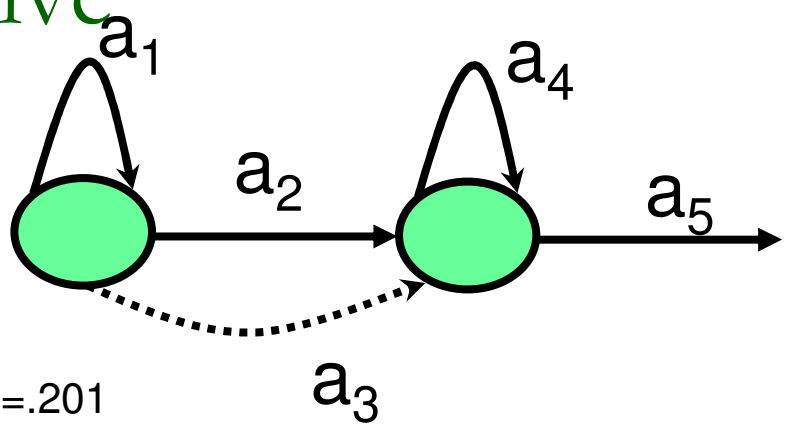
- New estimates:
- $a_1 = .46$ $a_2 = .34$ $a_3 = .20$

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

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

1st term 2C1 because in abaa, last 'a' by a5 so 2'a's in aba

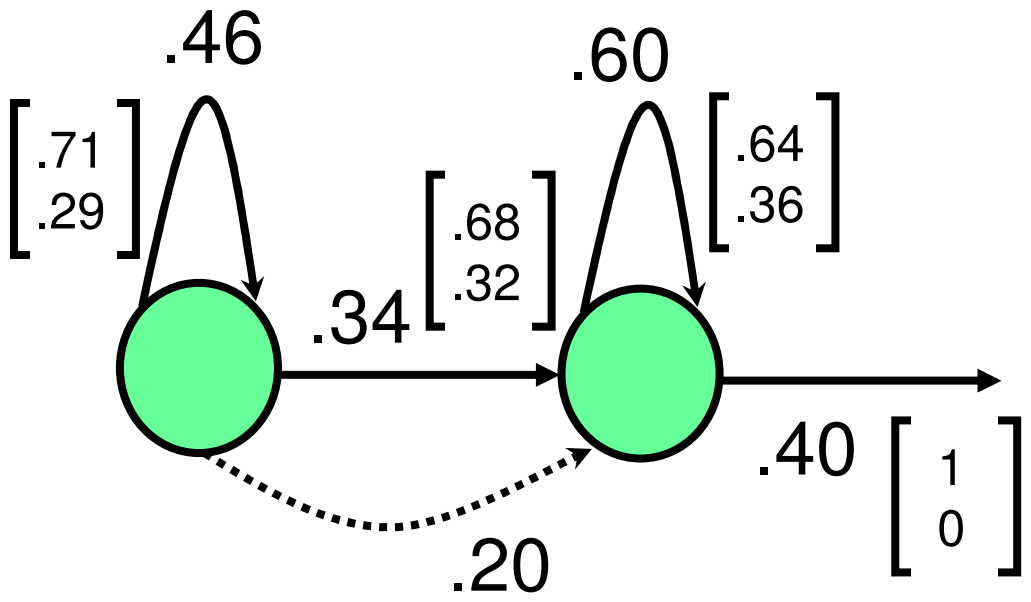
Problem 3: Enumerative Example cont'd



- 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

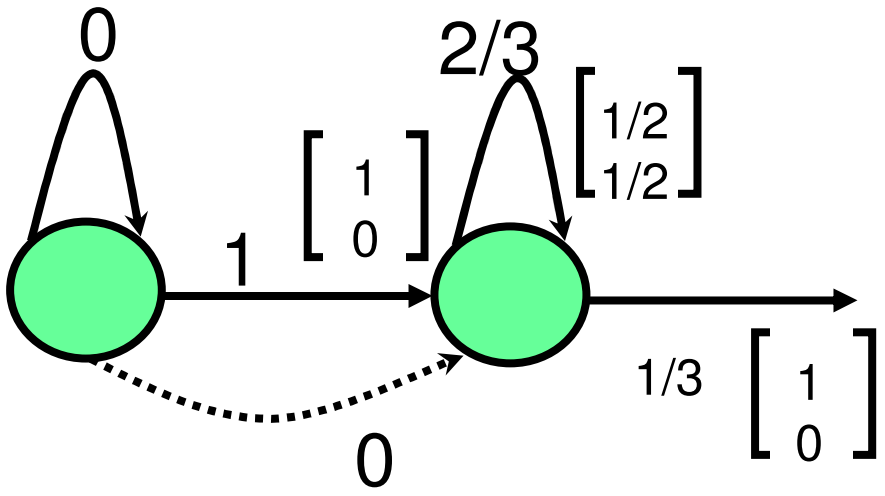
- New parameters



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

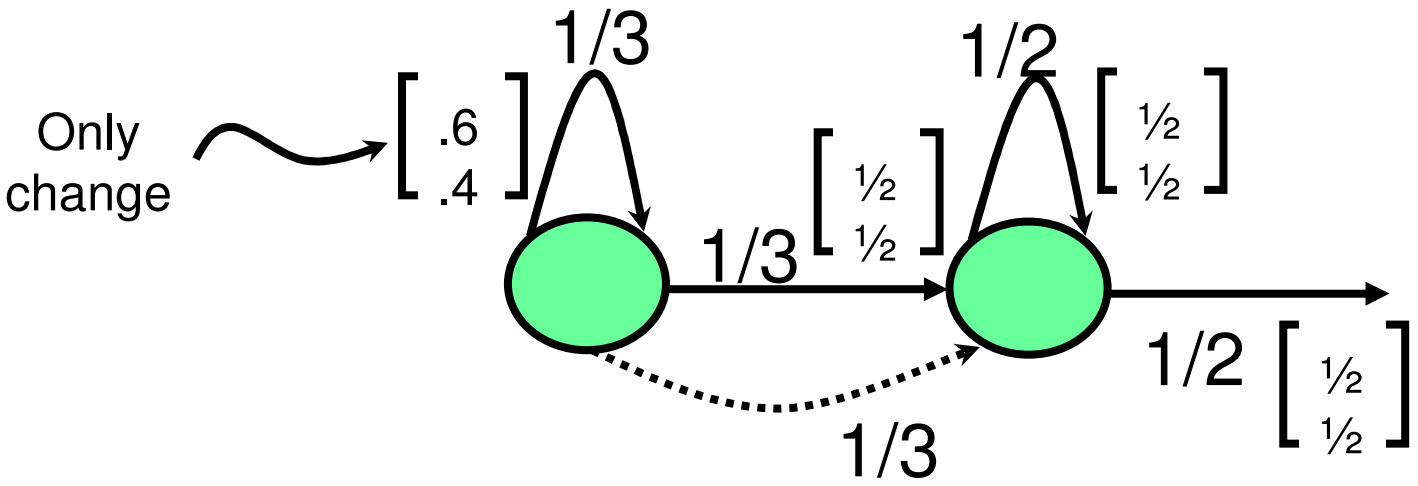
Problem 3: Enumerative Example cont'd

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



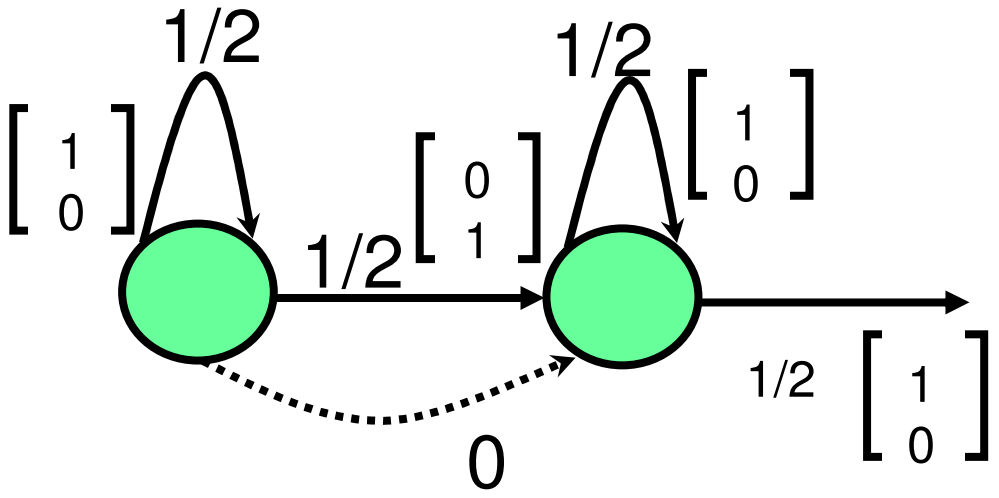
Problem 3: Enumerative Example cont'd

- 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



Problem 3: Parameter Estimation 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.

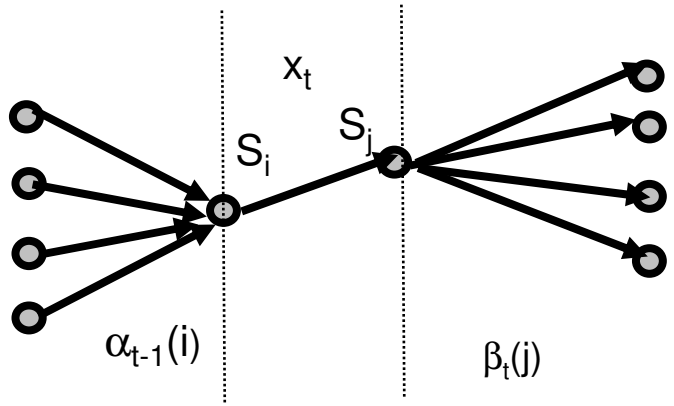
Problem 3: Forward-Backward Algorithm

- 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 tr_{ij} is taken at time t, 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

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

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

Problem 3: F-B algorithm cont'd

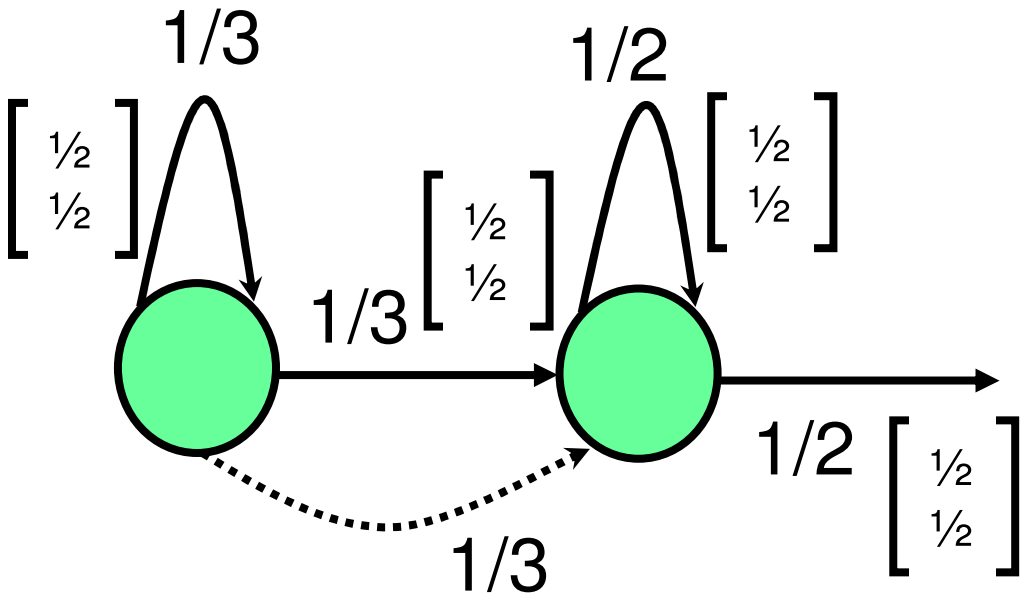
- Transition count $c_t(tr_{ij}|X) = p_t(tr_{ij},X) / Pr(X)$
- The β's are computed recursively in a backward pass (analogous to the forward pass for the α's)

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

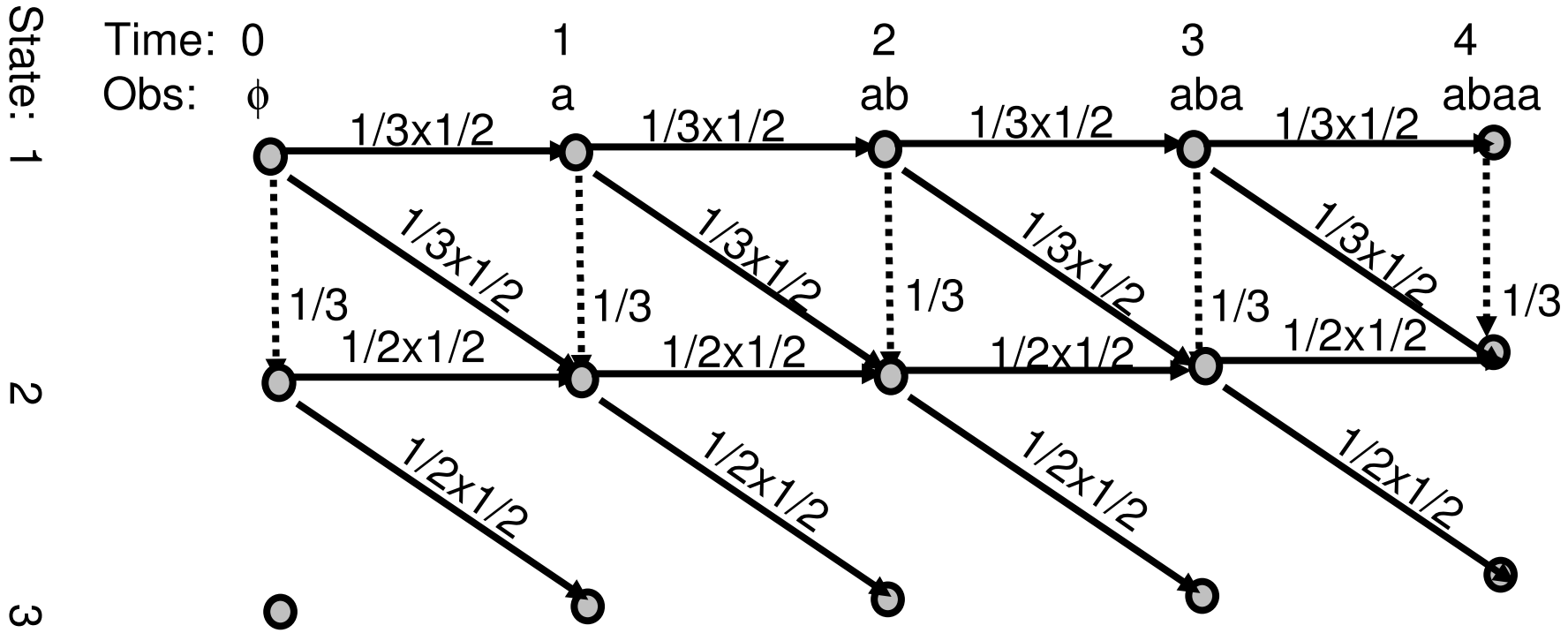
$+ \Sigma_k \beta_t(k) \; a_{jk}$ (for all null arcs)

Problem 3: F-B algorithm cont'd

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

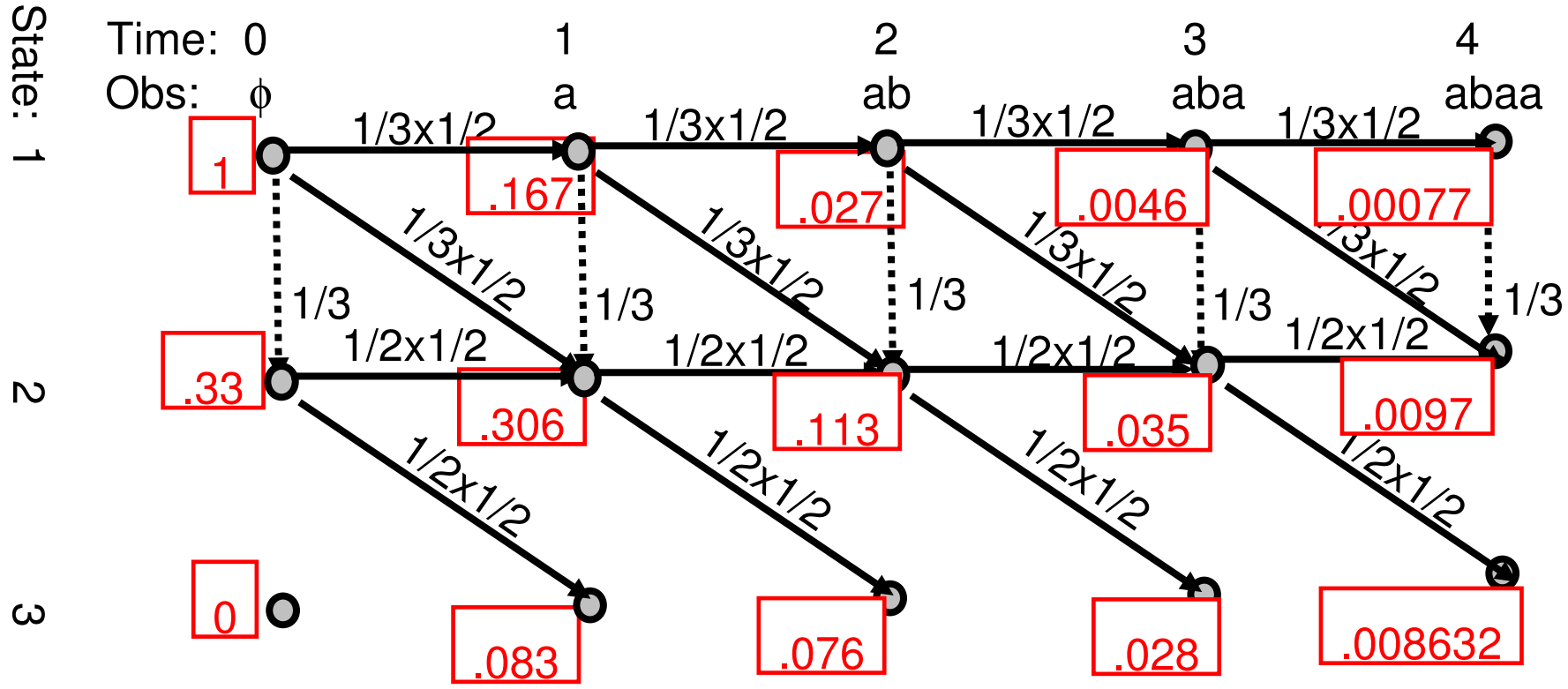


Problem 3: F-B algorithm, cont'd



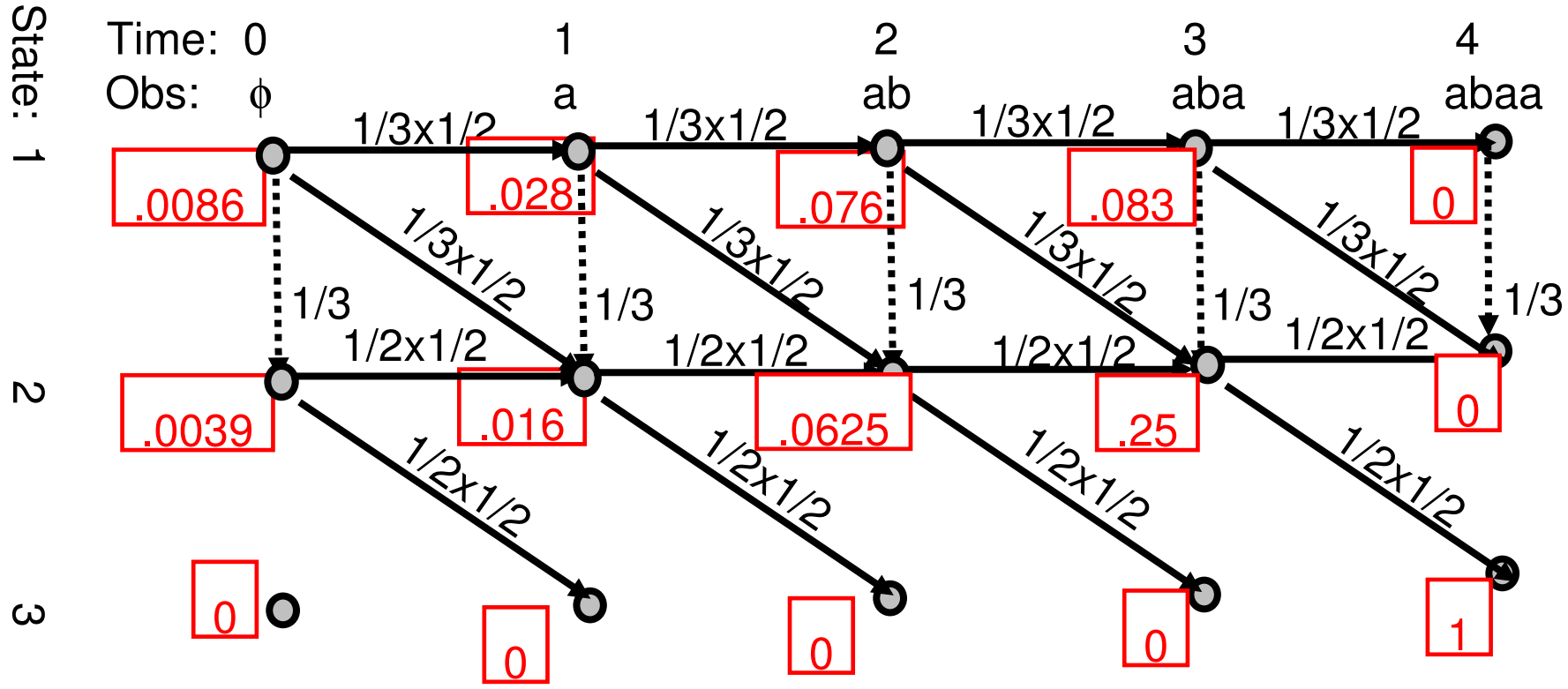
Problem 3: F-B algorithm, cont'd

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



Problem 3: F-B algorithm, cont'd

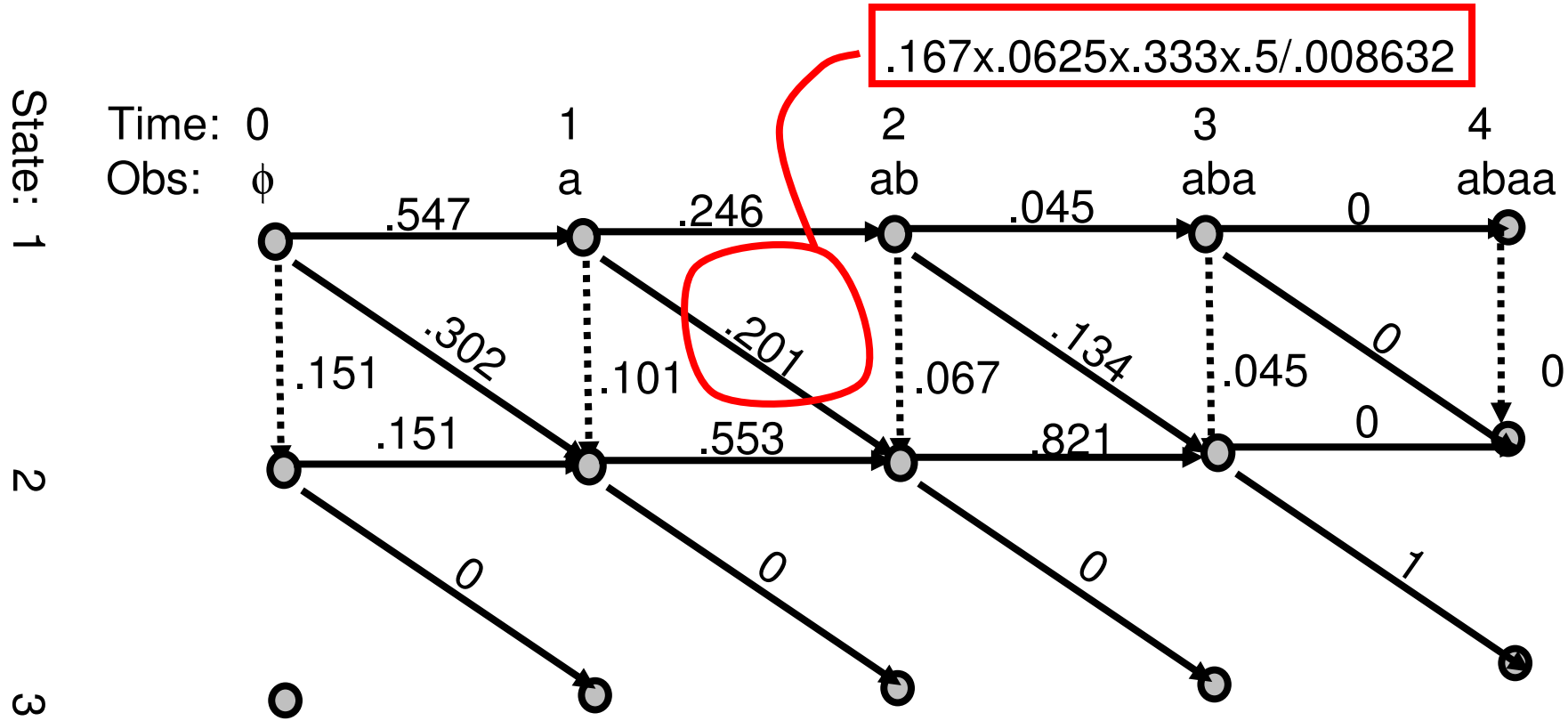
Compute β's.



Problem 3: F-B algorithm, cont'd

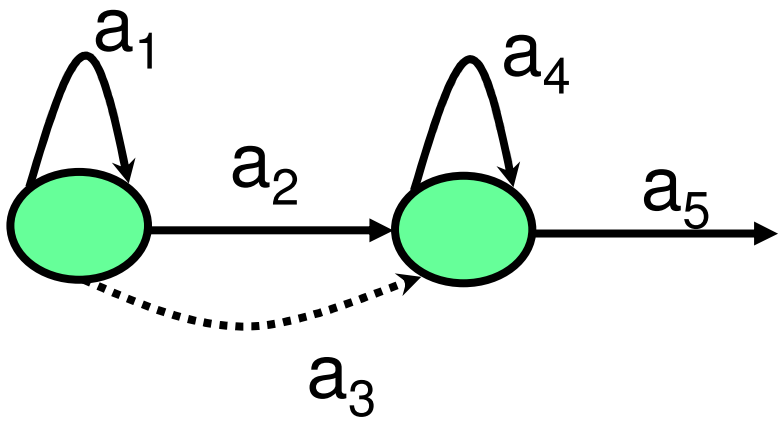
Compute counts. (a posteriori probability of each transition)

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



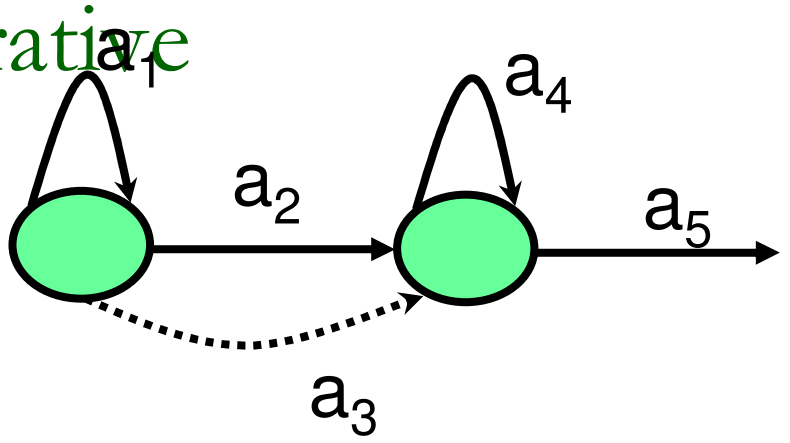
Problem 3: F-B algorithm cont'd

- $C(a_1)$=.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

Remember our Enumerative Example?



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

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

$a1 = C(a1)/\{C(a1) + C(a2) + C(a3)\}$

- New estimates:
- $a_1 = .46$ $a_2 = .34$ $a_3 = .20$

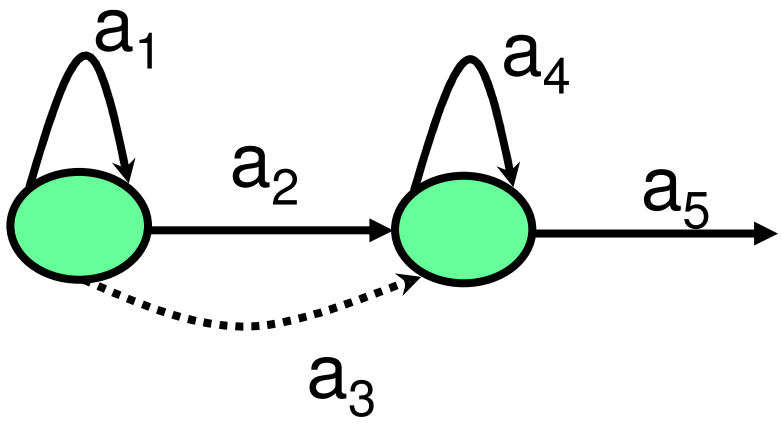
- $Count(a_1,'a') = 2C_1+C_2+C_3+C_4+C_5 = .592$ $Count(a_1,'b') = C_1+C_2+C_3 = .246$

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

1st term 2C1 because in abaa, last 'a' by a5 so 2'a's in aba

Problem 3: F-B algorithm cont'd

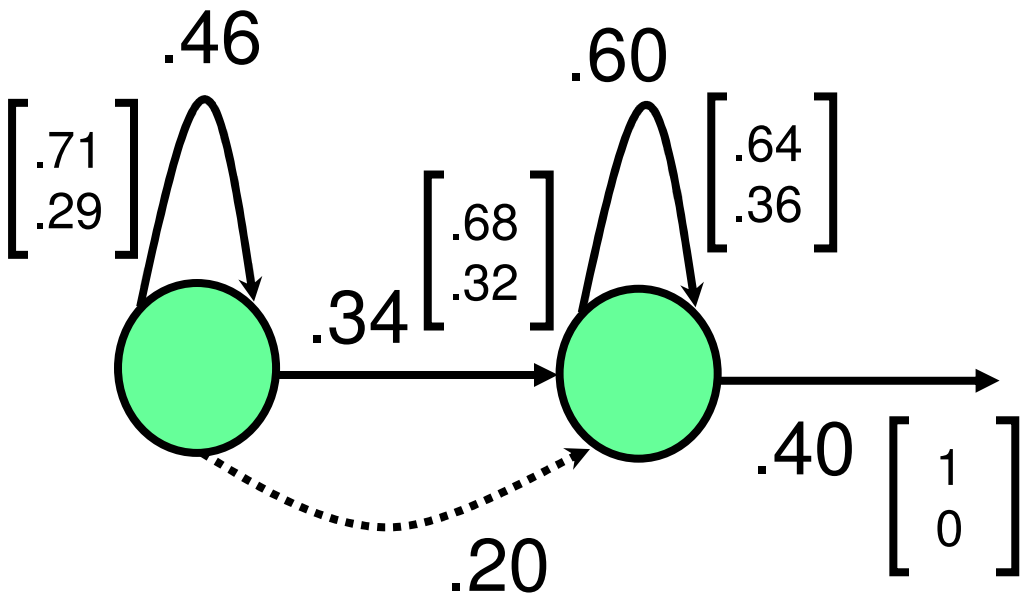
- $C(a_1)$=.547+.246+.045 = 0.838
- $C(a_2)$=.302+.201+.134
- $C(a_3)$=.151+.101+.067+.045
- $C(a_4)$=.151+.553+.821
- $C(a_5)$=1



- C(a1,'a')=.547+.045, =0.592 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!!

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 (time evolution). 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

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

  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.

Reference

- http://www.cs.jhu.edu/~jason/papers/#tnlp02
- http://www.ee.columbia.edu/~stanchen/fall09/e6870/