Statistical Methods for NLP

Information Extraction, Hidden Markov Models

Sameer Maskey

* Most of the slides provided by
Bhuvana Ramabhadran, Stanley Chen, Michael Picheny

Project Proposals

- Proposals due in 2 weeks (Feb 23)
- 1 Page

Topics for Today

- Information Extraction
- Hidden Markov Models

Information Extraction

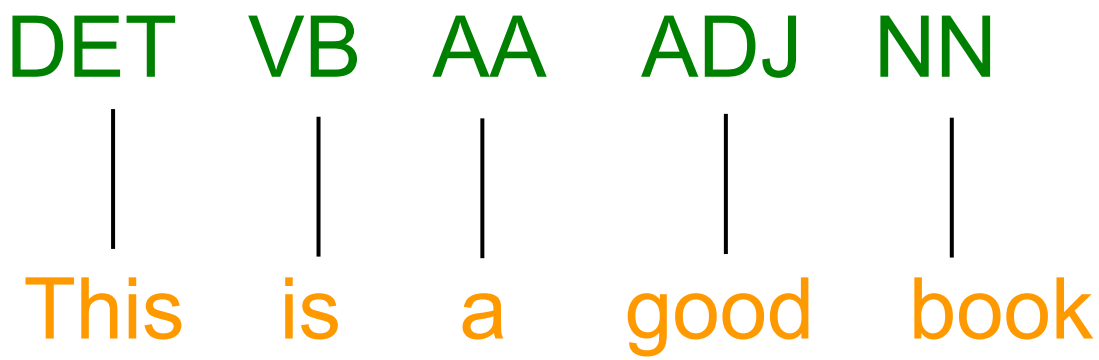
- Extract relevant information from large amount of unstructured text
- Extracted information can be in structured form
  - Can be used to populate databases for example
- We can define the kind of information we want to extract

Examples of Information Extraction Tasks

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

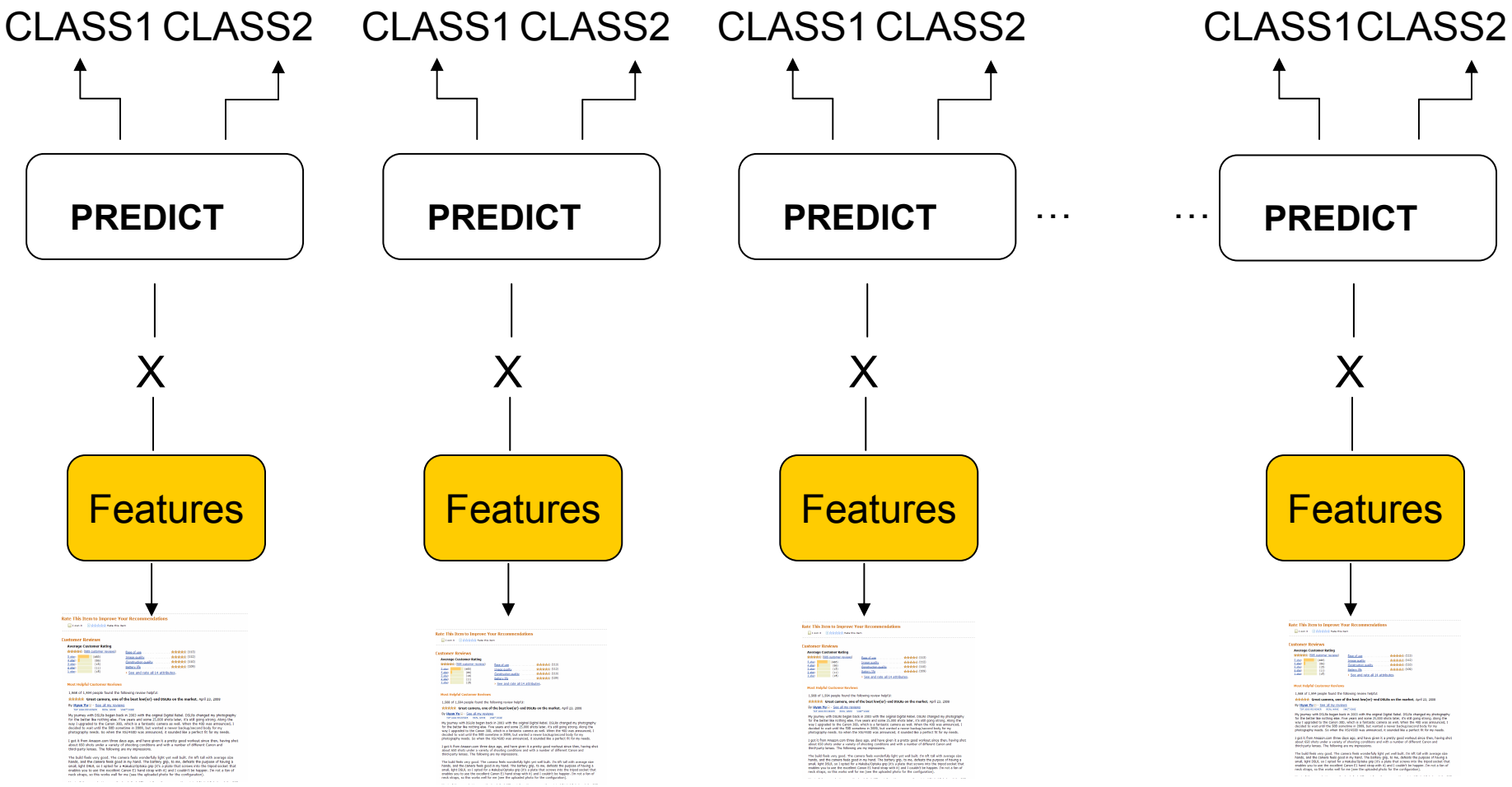
Classifiers for Information Extraction

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

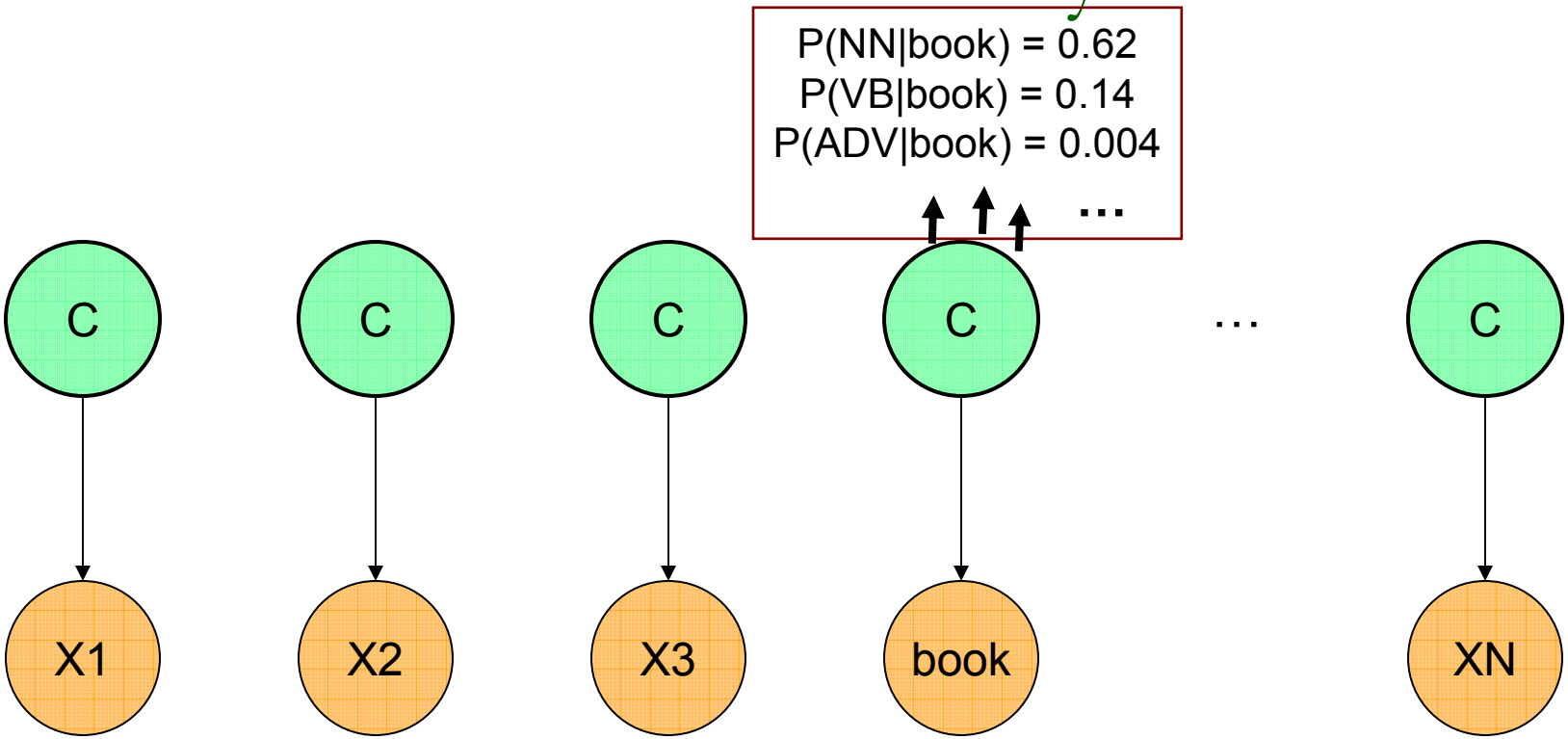


- What kind of classifier may work well for this kind of sequence classification?

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}

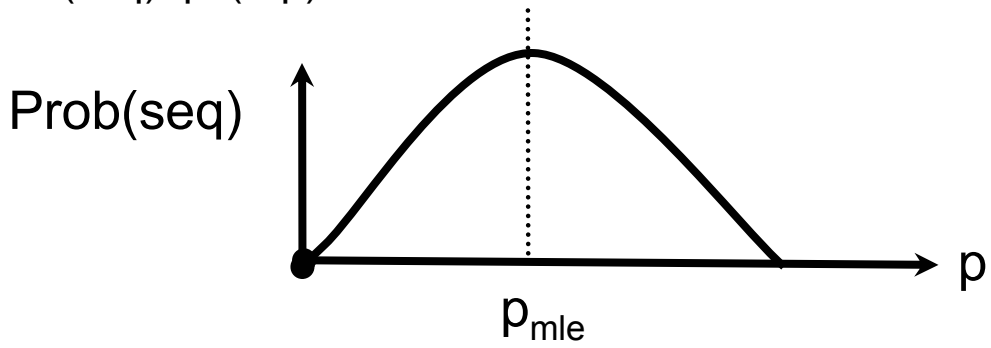
Models without Memory: Learning Model Parameters

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

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

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

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

$p_{mle} = H/N$

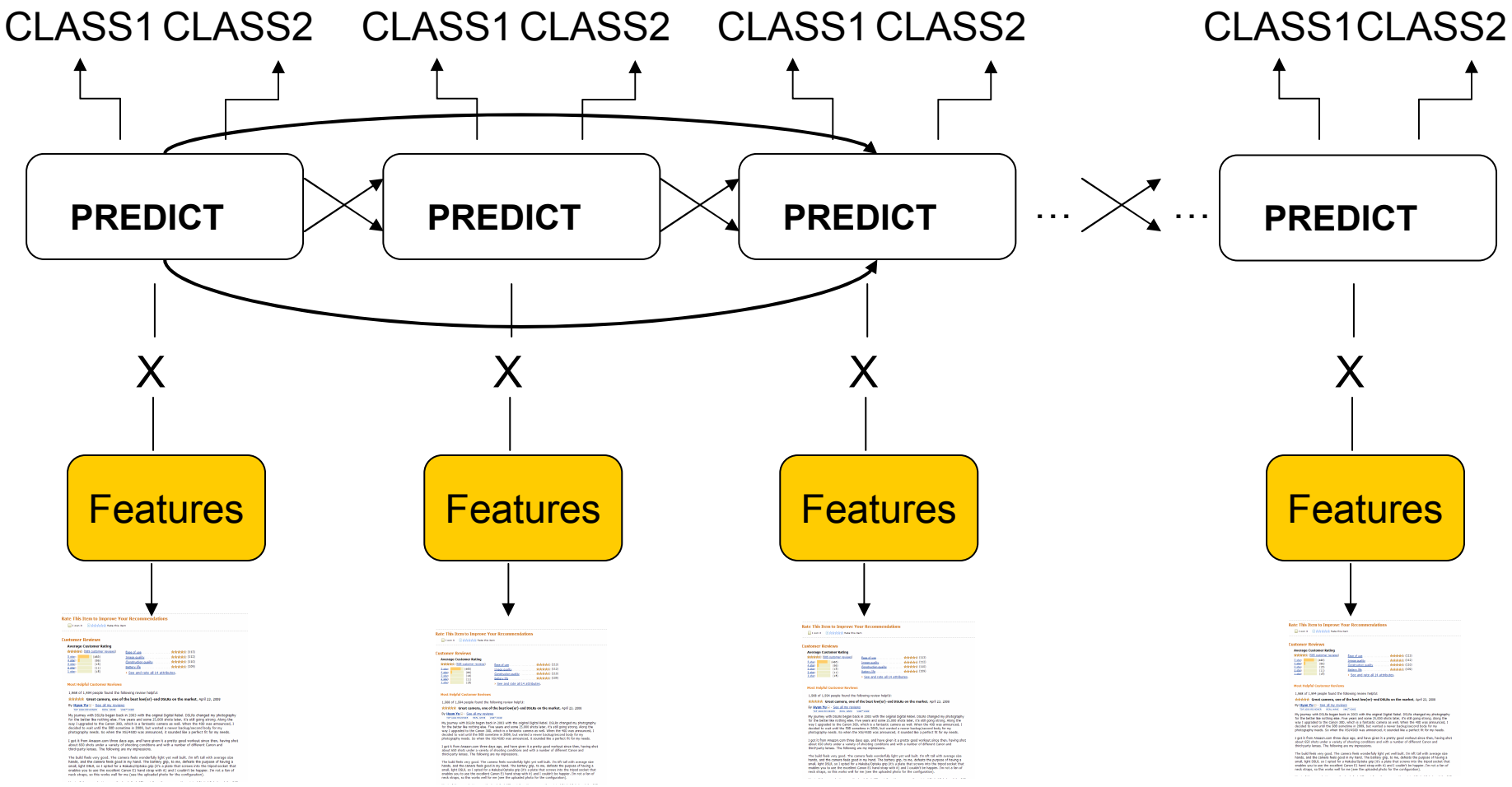
Models without Memory Example, cont'd

- 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

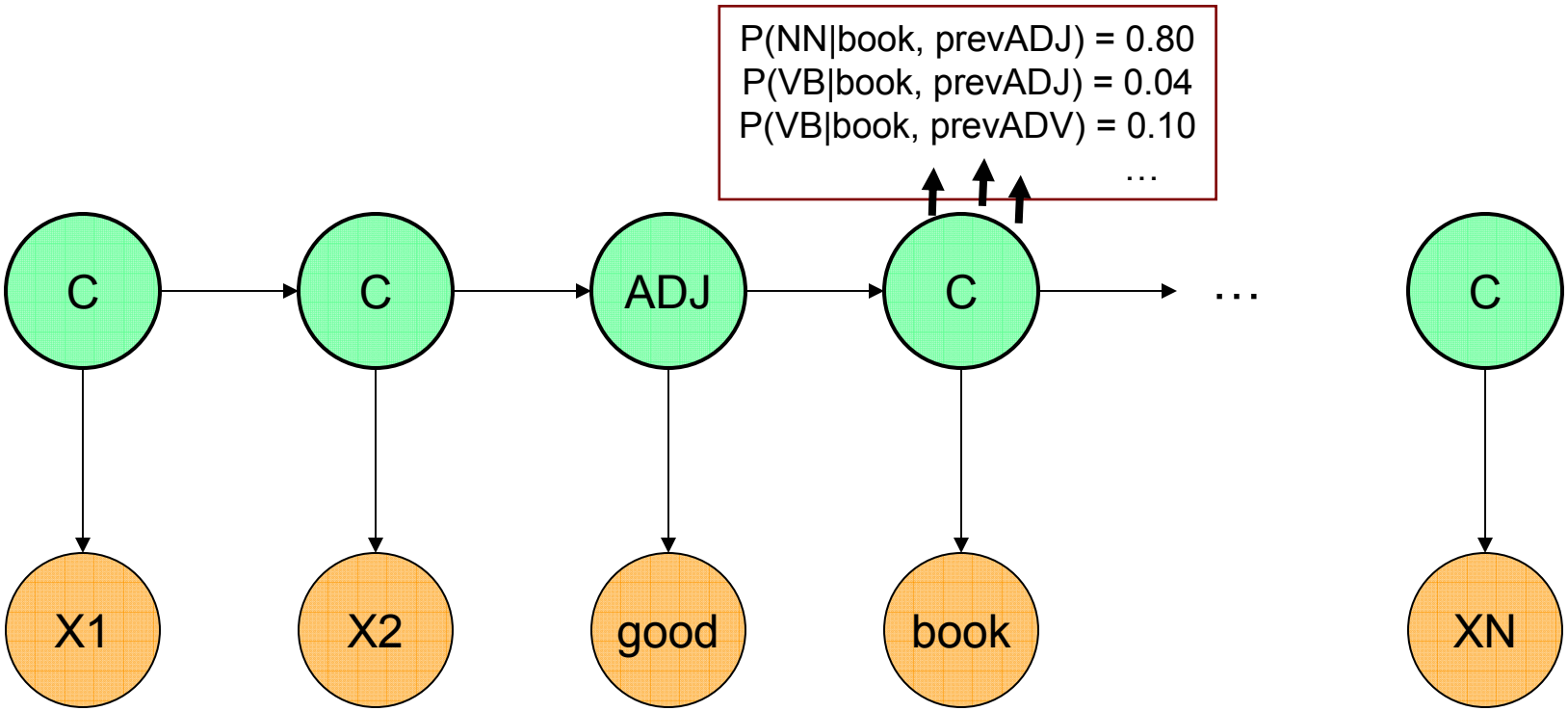
- Flipping a coin was memory-less. The outcome of each flip did not depend on the outcome of the other flips.
- Adding memory to a memory-less model gives us a Markov Model. Useful for modeling sequences of events.
- For POS tagging adding memory to classifier could be useful

Classification with Memory



Current Prediction depends only on previous predictions and current observation

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

Adding Memory to Coin Example

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

- 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

- 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

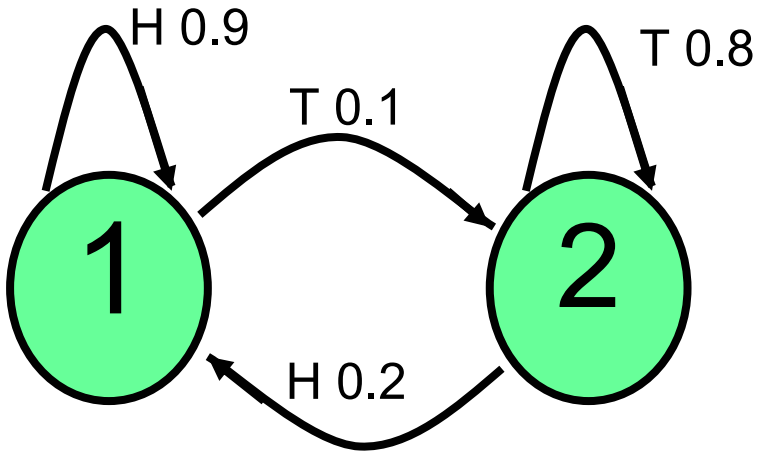
Markov Models – State Space Representation

- Consider 2 coins.

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

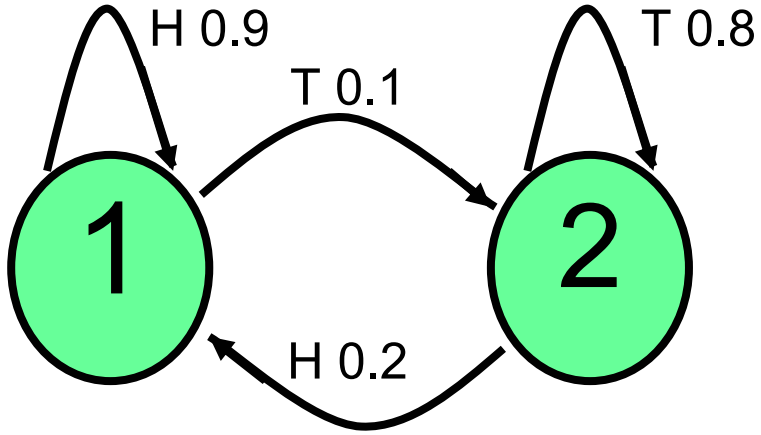
  Coin 2: $p_H = 0.2$, $p_T = 0.8$

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 $p_H = 0.2$

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 Markov Model

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

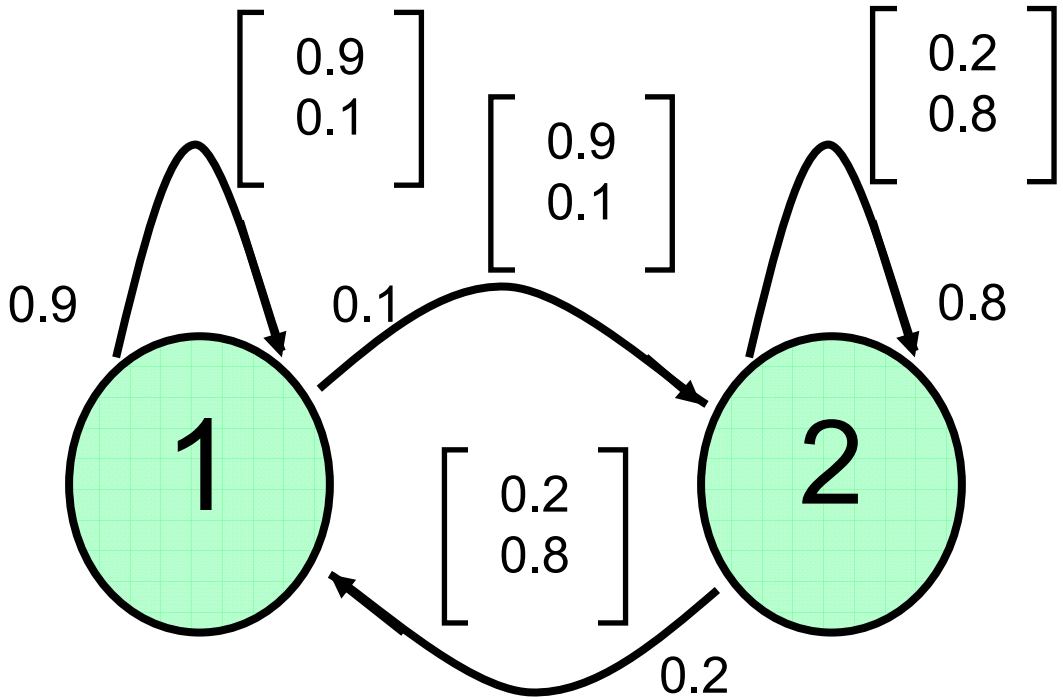
- Experiment:
  Flip the same two coins. This time, flip each coin twice. The first flip gets recorded as the output sequence. The second flip determines which coin gets flipped next.

- 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

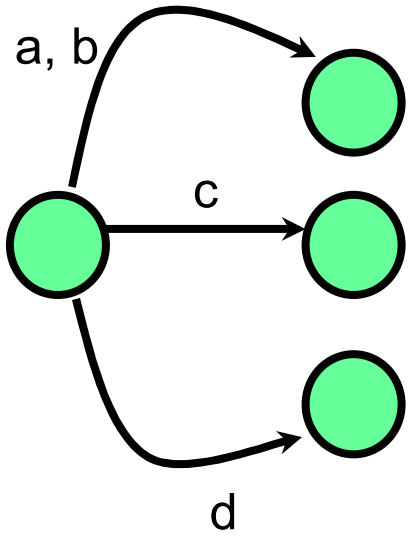
Hidden Markov Model

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

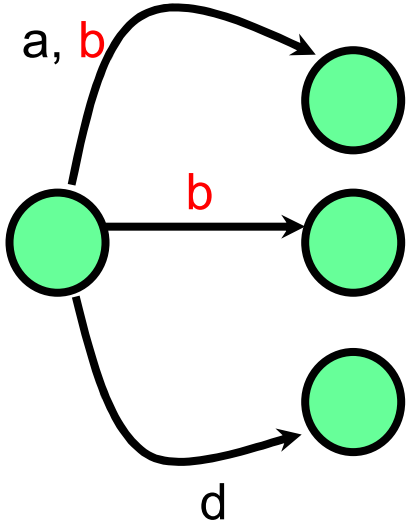


Is a Markov Model Hidden or Not?

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)

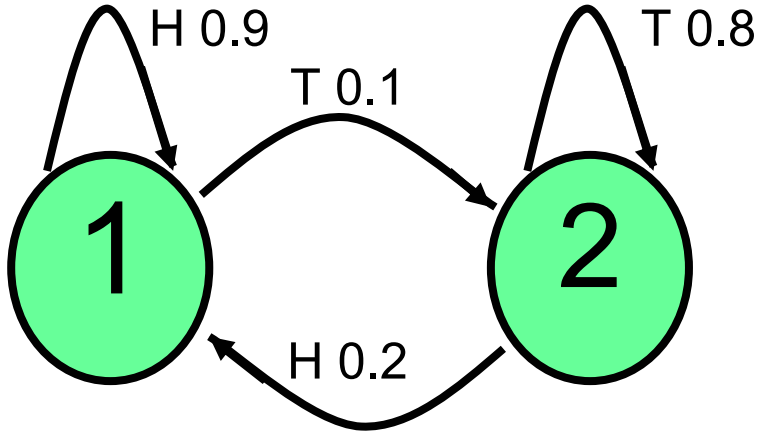
3. Estimate the parameters of the model (training)

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

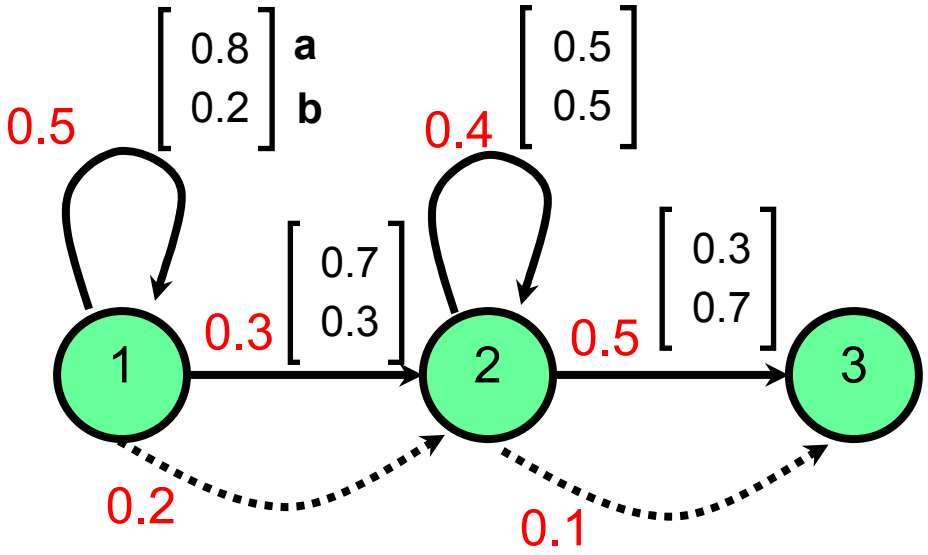
Sum over all possible state sequences:

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

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

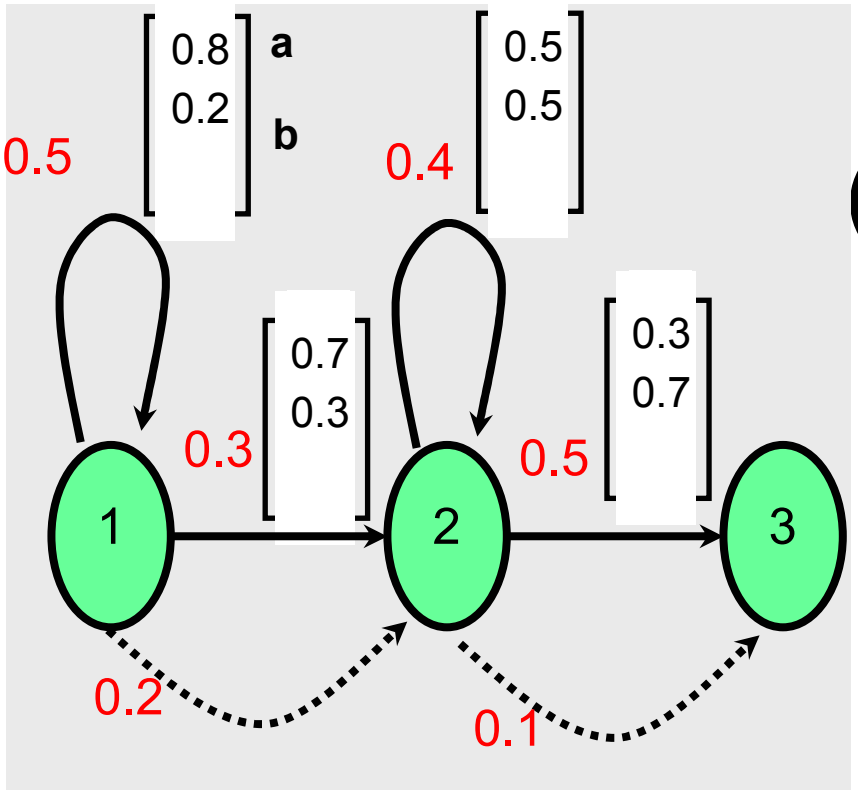
Unfortunately, this calculation is exponential in the length of the sequence

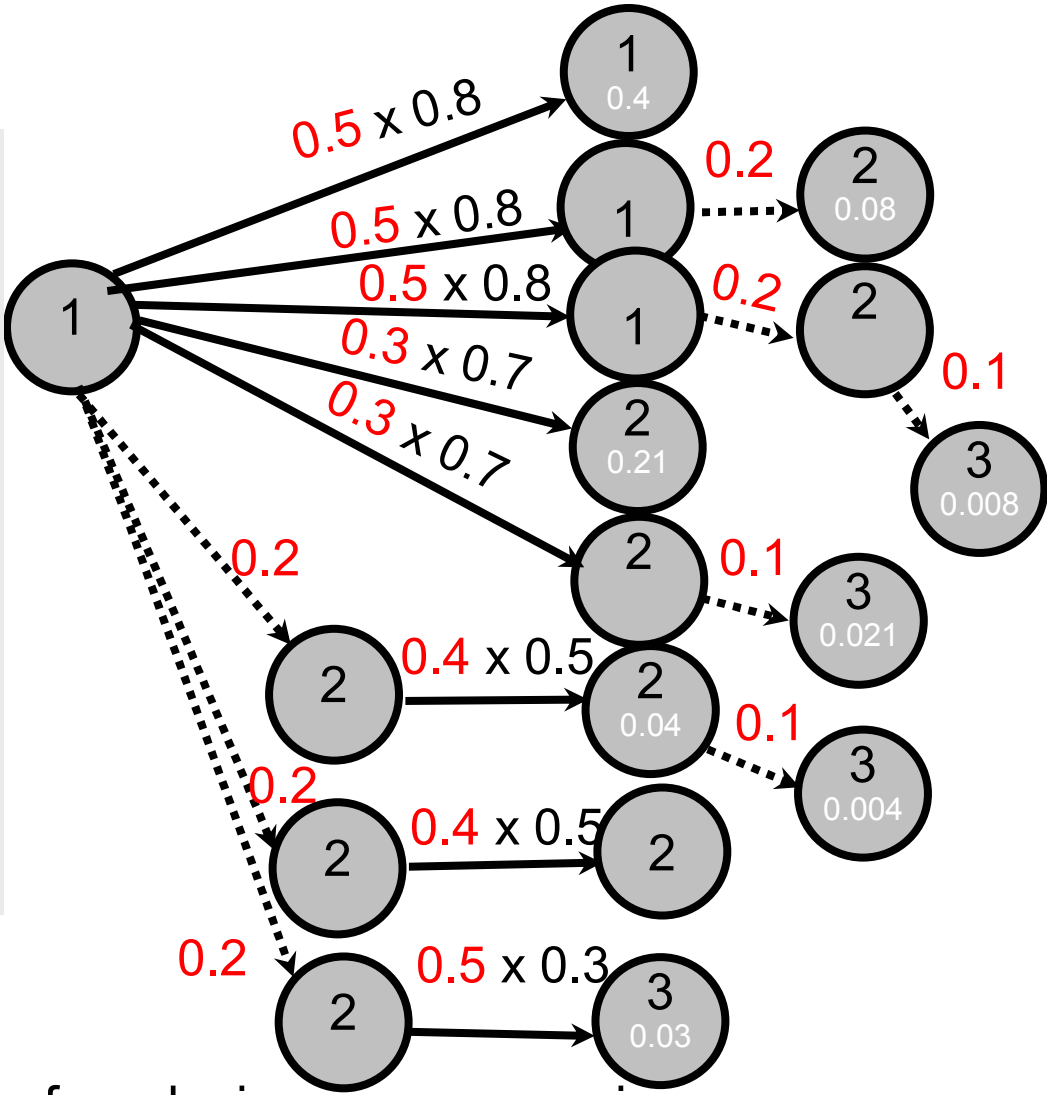
Example for Problem 1



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

Example for Problem 1,cont'd

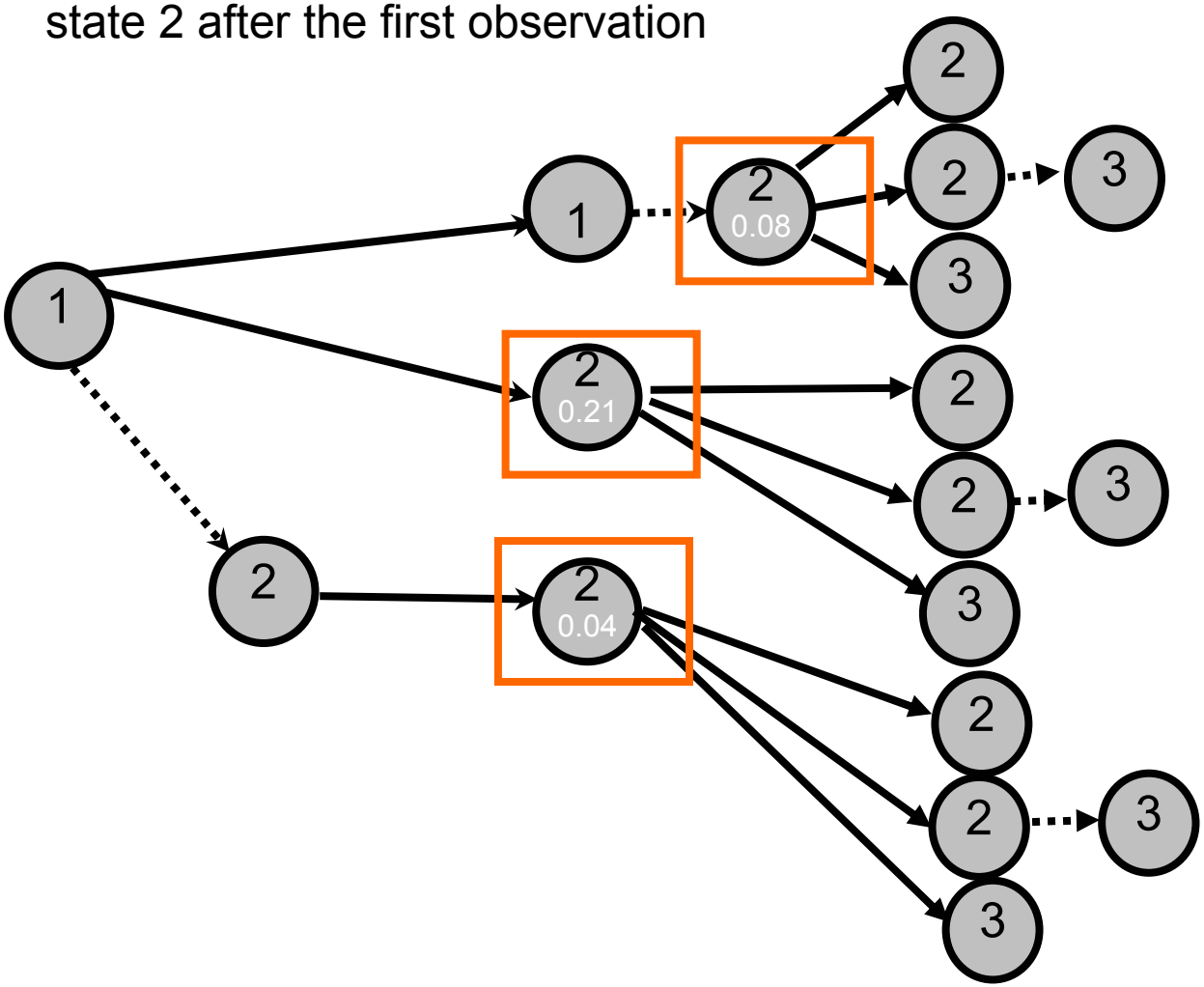




Let's enumerate all possible ways of producing x_1=a, assuming we start in state 1.

Example for Problem 1, cont'd

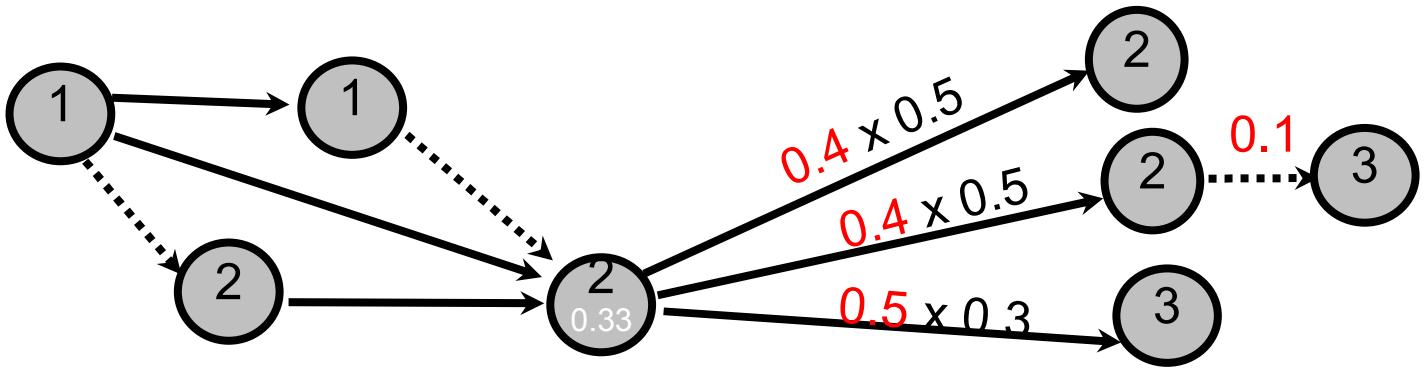
- Now let's think about ways of generating x1x2=aa, for all paths from state 2 after the first observation



Example for Problem 1,cont'd

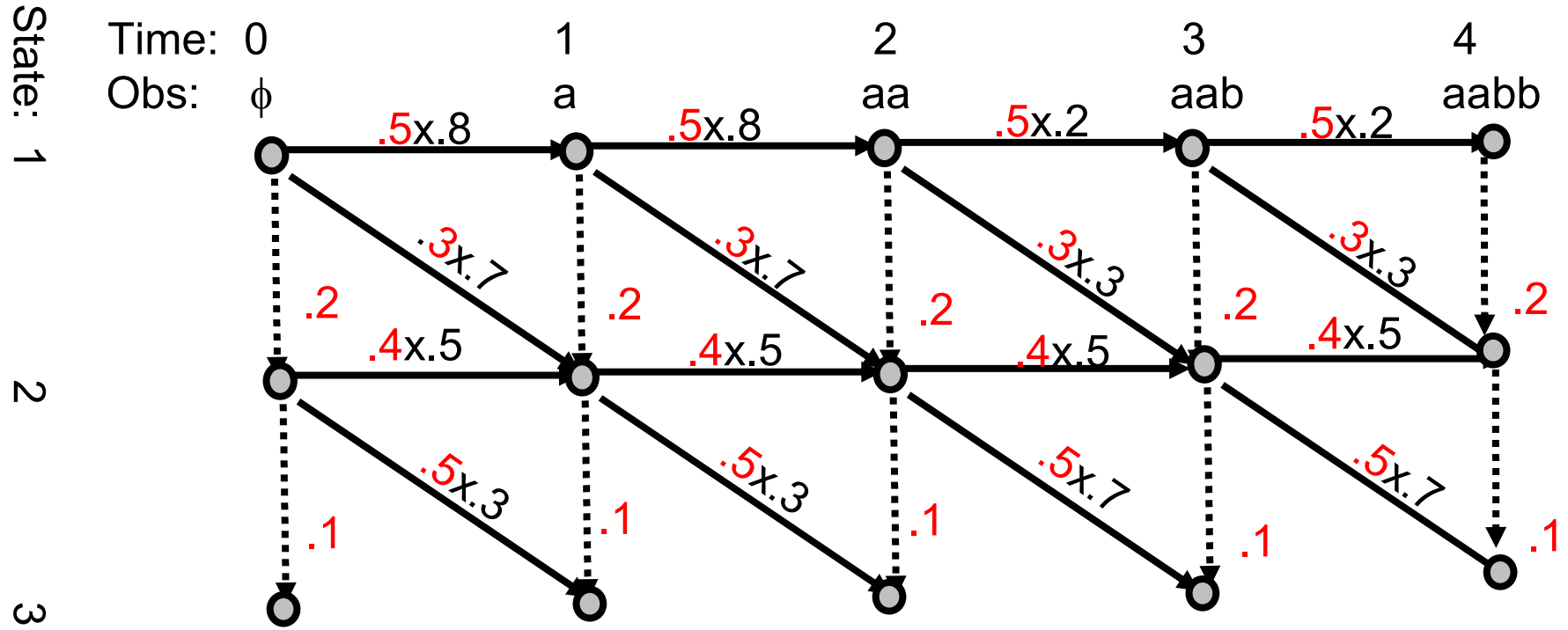
We can save computations by combining paths.

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



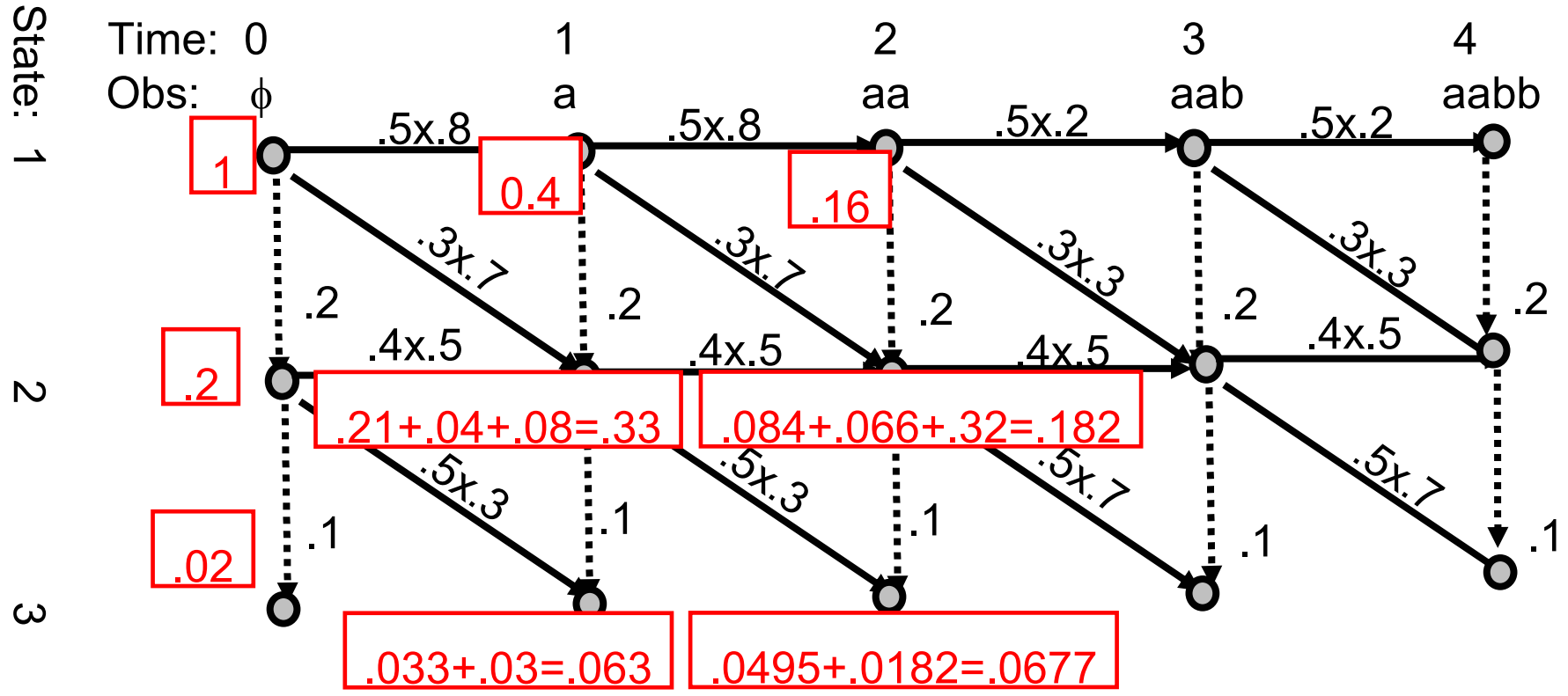
Problem 1: Trellis Diagram

- 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 $t \in \{1..T\}$ 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.

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

This calculation allows us to solve Problem 1:

$$\mathbf{P_\theta(X) = \sum_s \alpha_T(s)}$$

Problem 2

Given the observations X, find the most likely state sequence

This is solved using the Viterbi algorithm

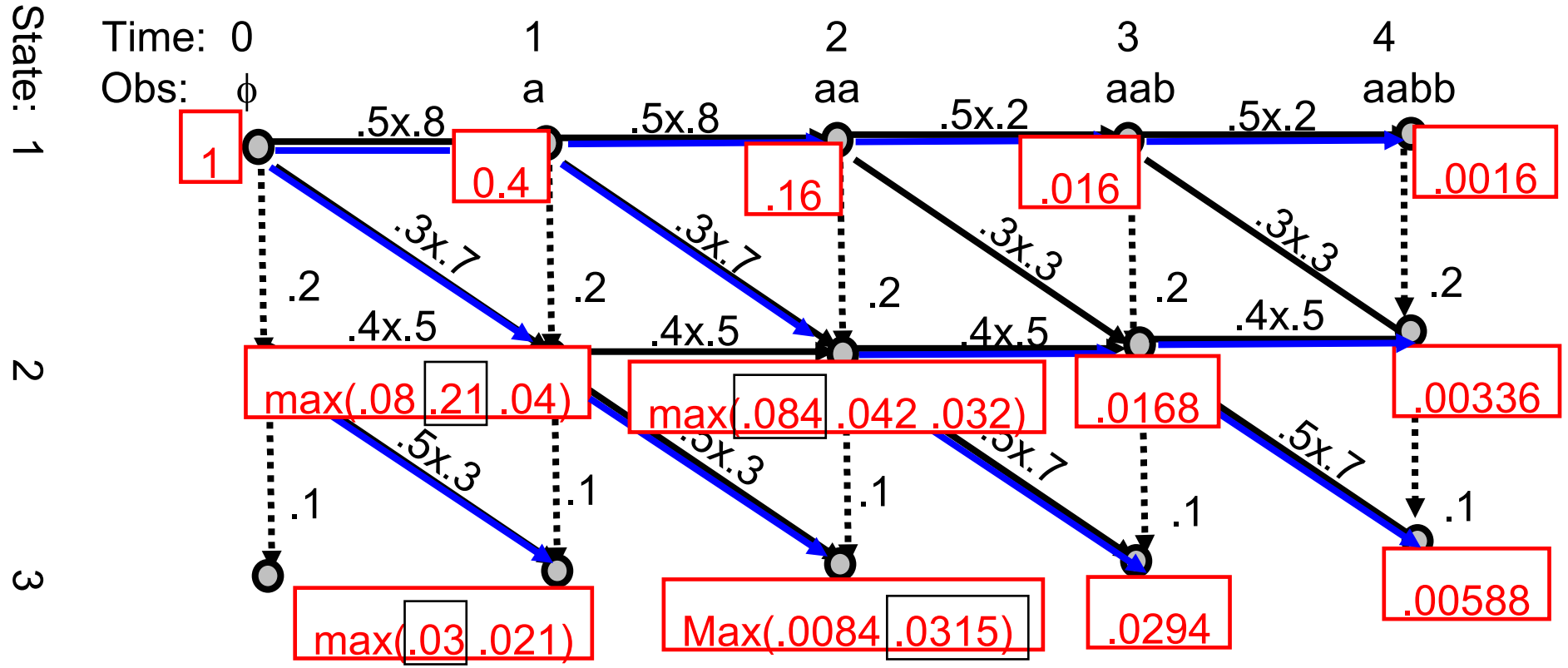
Preview:

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

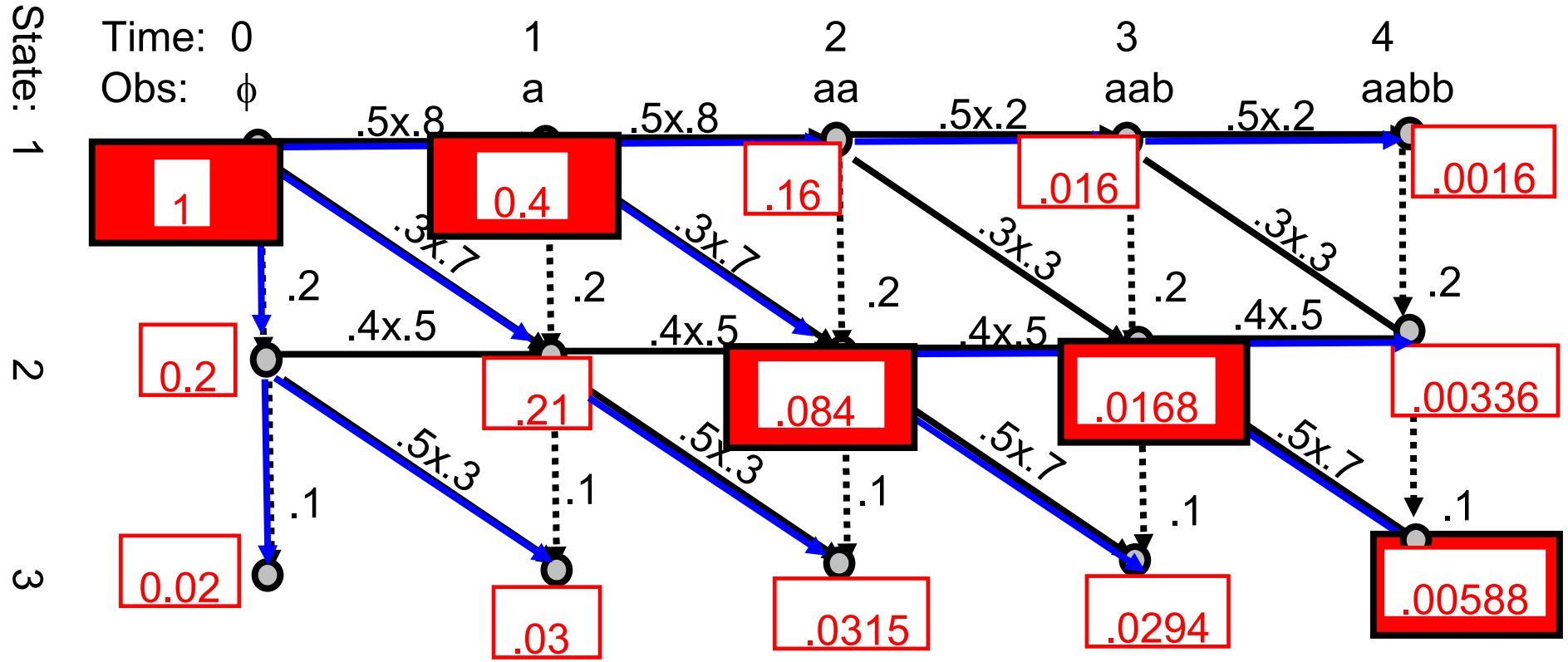
Problem 2: Viterbi algorithm

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.



Reference

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