



# CS 564: Sequence Learning: Hidden Markov Model

# Observations leading to why probability is needed

---

- Many intelligence tasks are sequence labeling tasks
- Tasks carried out in layers
- Within a layer, there are limited windows of information
- This naturally calls for strategies for dealing with uncertainty
- Probability and Markov process give a way

*“I went with my friend to the bank to withdraw some money, but was disappointed to find it closed”*

---

---

POS	Bank (N/V)	closed (V/ adj)
-----	------------	-----------------

---

Sense	Bank (financial institution)	withdraw(take away)
-------	------------------------------	---------------------

---

Pronoun drop	But I/friend/money/bank	was disappointed
--------------	-------------------------	------------------

---

SCOPE	With	my friend
-------	------	-----------

---

Co-referencing	It -> bank
----------------	------------

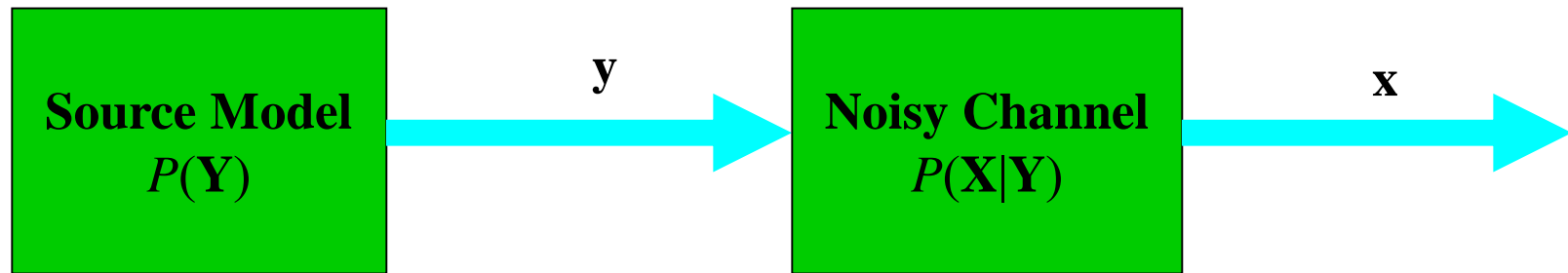
---

# Different Models for Sequence Learning

---

- HMM
- Maximum Entropy Markov Models
- Conditional Random Fields

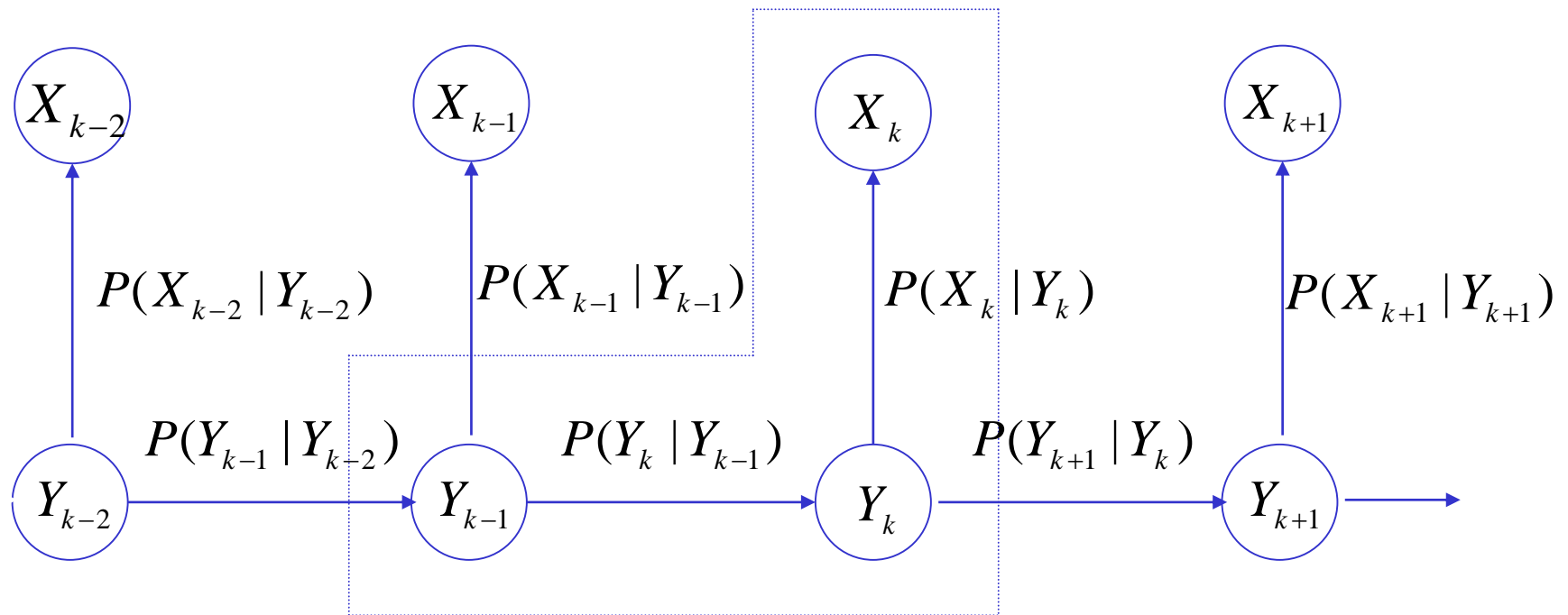
# Hidden Markov Model (HMM) : Generative Modeling



$$P(\mathbf{y}) = \prod_i P(y_i | y_{i-1})$$

$$P(\mathbf{x} | \mathbf{y}) = \prod_i P(x_i | y_i)$$

# Dependency (1st order)



# Markov Models

- **Set of states:**  $\{s_1, s_2, \dots, s_N\}$
- Process moves from one state to another generating a sequence of states:  $s_{i1}, s_{i2}, \dots, s_{ik}, \dots$
- **Markov chain property:** probability of each subsequent state depends only on what was the previous state:

$$P(s_{ik} \mid s_{i1}, s_{i2}, \dots, s_{ik-1}) = P(s_{ik} \mid s_{ik-1})$$

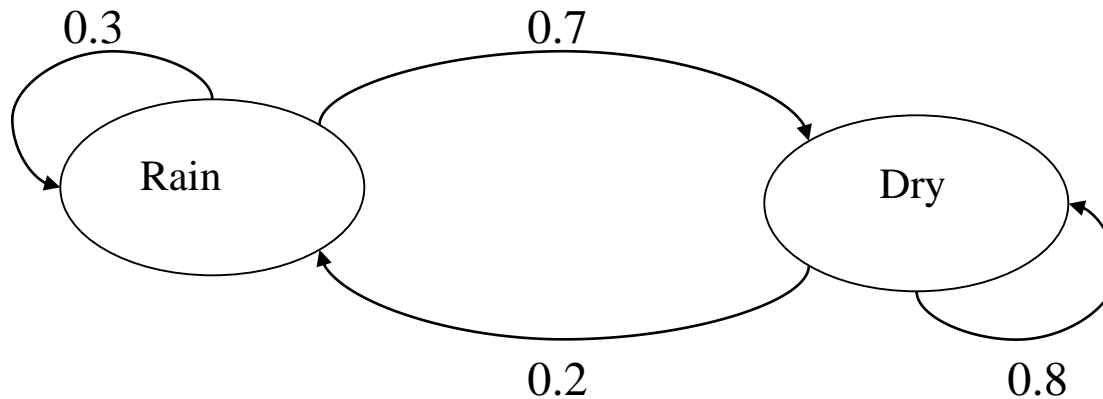
- To define Markov model, the following probabilities have to be specified: *transition probabilities* and *initial probabilities*

$$a_{ij} = P(s_i \mid s_j)$$

$$\pi_i = P(s_i)$$

# Example of Markov Model

---



- Two states : 'Rain' and 'Dry'.
- Transition probabilities:  $P(\text{'Rain'} \mid \text{'Rain'})=0.3$  ,  $P(\text{'Dry'} \mid \text{'Rain'})=0.7$  ,  $P(\text{'Rain'} \mid \text{'Dry'})=0.2$ ,  $P(\text{'Dry'} \mid \text{'Dry'})=0.8$
- Initial probabilities: say  $P(\text{'Rain'})=0.4$  ,  $P(\text{'Dry'})=0.6$  .



# Calculation of sequence probability

- By Markov chain property, probability of state sequence can be found by the formula:

$$\begin{aligned}P(s_{i1}, s_{i2}, \dots, s_{ik}) &= P(s_{ik} \mid s_{i1}, s_{i2}, \dots, s_{ik-1}) P(s_{i1}, s_{i2}, \dots, s_{ik-1}) \\&= P(s_{ik} \mid s_{ik-1}) P(s_{i1}, s_{i2}, \dots, s_{ik-1}) = \dots \\&= P(s_{ik} \mid s_{ik-1}) P(s_{ik-1} \mid s_{ik-2}) \dots P(s_{i2} \mid s_{i1}) P(s_{i1})\end{aligned}$$

- Suppose we want to calculate a probability of a sequence of states in our example, {‘Dry’, ‘Dry’, ‘Rain’, ‘Rain’}.

$$\begin{aligned}P(\{\text{'Dry'}, 'Dry', 'Rain', 'Rain'}\}) &= \\P(\text{'Rain'} \mid \text{'Rain'}) P(\text{'Rain'} \mid \text{'Dry'}) P(\text{'Dry'} \mid \text{'Dry'}) P(\text{'Dry'}) &= \\= 0.3 * 0.2 * 0.8 * 0.6\end{aligned}$$

# Hidden Markov models

- **Set of states:**  $\{s_1, s_2, \dots, s_N\}$

- Process moves from one state to another generating a sequence of states :

$$s_{i1}, s_{i2}, \dots, s_{ik}, \dots$$

- **Markov chain property:** probability of each subsequent state depends only on what was the previous state:

$$P(s_{ik} \mid s_{i1}, s_{i2}, \dots, s_{ik-1}) = P(s_{ik} \mid s_{ik-1})$$

- States are not visible, but each state randomly generates one of M observations (or visible states)

$$\{v_1, v_2, \dots, v_M\}$$

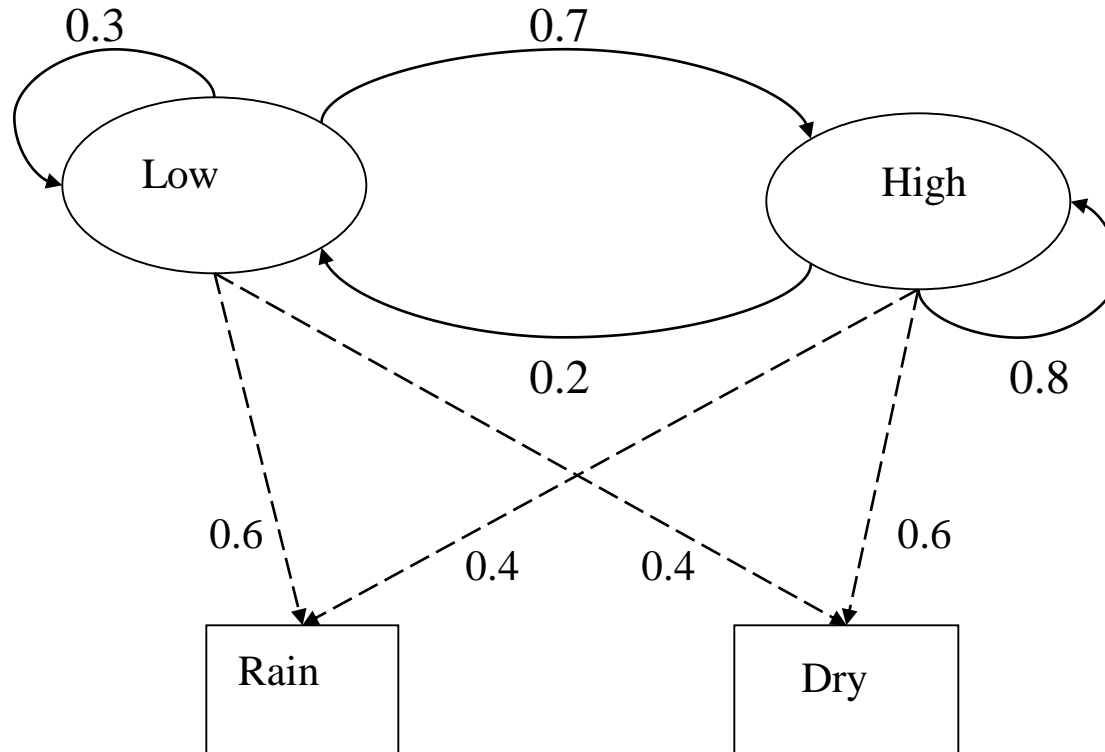
# Hidden Markov Models

---

To define hidden Markov model, the following probabilities have to be specified:

- *matrix of transition probabilities*  $A=(a_{ij})$ ,  $a_{ij}= P(s_i | s_j)$
- *matrix of observation probabilities*  $B=(b_i(v_m))$ ,  $b_i(v_m) = P(v_m | s_i)$
- *vector of initial probabilities*  $\pi=(\pi_i)$ ,  $\pi_i = P(s_i)$
- Model is represented by  $M=(A, B, \pi)$

# Example of Hidden Markov Model



# Example of Hidden Markov Model

---

- *Two states* : ‘Low’ and ‘High’ atmospheric pressure
- *Two observations* : ‘Rain’ and ‘Dry’
- *Transition probabilities*:  
 $P(\text{‘Low’}|\text{‘Low’})=0.3$ ,  $P(\text{‘High’}|\text{‘Low’})=0.7$ ,  
 $P(\text{‘Low’}|\text{‘High’})=0.2$ ,  $P(\text{‘High’}|\text{‘High’})=0.8$
- *Observation probabilities*:  
 $P(\text{‘Rain’}|\text{‘Low’})=0.6$        $P(\text{‘Dry’}|\text{‘Low’})=0.4$   
 $P(\text{‘Rain’}|\text{‘High’})=0.4$        $P(\text{‘Dry’}|\text{‘High’})=0.36$
- *Initial probabilities*: say  $P(\text{‘Low’})=0.4$  ,  $P(\text{‘High’})=0.6$

# Calculation of observation sequence probability

• Suppose we want to calculate a probability of a sequence of observations in our example,  $\{\text{'Dry'}, \text{'Rain'}\}$

• Consider all possible hidden state sequences:

$$P(\{\text{'Dry'}, \text{'Rain'}\}) = P(\{\text{'Dry'}, \text{'Rain'}\}, \{\text{'Low'}, \text{'Low'}\}) + P(\{\text{'Dry'}, \text{'Rain'}\}, \{\text{'Low'}, \text{'High'}\}) + P(\{\text{'Dry'}, \text{'Rain'}\}, \{\text{'High'}, \text{'Low'}\}) + P(\{\text{'Dry'}, \text{'Rain'}\}, \{\text{'High'}, \text{'High'}\})$$

where first term is :

$$\begin{aligned} P(\{\text{'Dry'}, \text{'Rain'}\}, \{\text{'Low'}, \text{'Low'}\}) &= \\ P(\{\text{'Dry'}, \text{'Rain'}\} \mid \{\text{'Low'}, \text{'Low'}\}) P(\{\text{'Low'}, \text{'Low'}\}) &= \\ P(\text{'Dry'} \mid \text{'Low'}) P(\text{'Rain'} \mid \text{'Low'}) P(\text{'Low'}) P(\text{'Low'} \mid \text{'Low'}) &= \\ = 0.4 * 0.4 * 0.6 * 0.4 * 0.3 \end{aligned}$$

# Main issues using HMMs

• **Evaluation problem.** Given the HMM  $M=(A, B, \pi)$  and the observation sequence  $O=o_1 o_2 \dots o_K$ , calculate the probability that model  $M$  has generated sequence  $O$

• **Decoding problem.** Given the HMM  $M=(A, B, \pi)$  and the observation sequence  $O=o_1 o_2 \dots o_K$ , calculate the most likely sequence of hidden states  $s_i$  that produced this observation sequence  $O$

• **Learning problem.** Given some training observation sequences  $O=o_1 o_2 \dots o_K$  and general structure of HMM (numbers of hidden and visible states), determine HMM parameters  $M=(A, B, \pi)$  that best fit training data

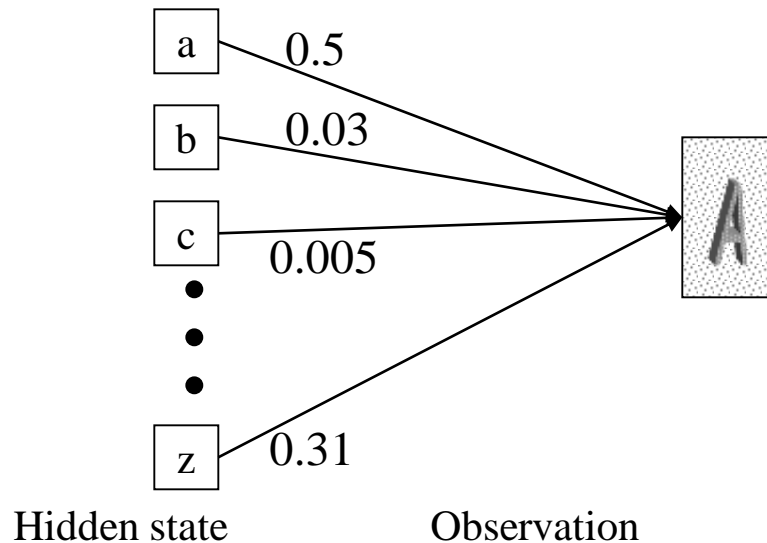
$O=o_1 \dots o_K$  denotes a sequence of observations  $o_k \in \{v_1, \dots, v_M\}$ .

# Word recognition example(1)

- Typed word recognition, assume all characters are separated



- Character recognizer outputs probability of the image being particular character,  $P(\text{image}|\text{character})$





# Word recognition example(2)

- Hidden states of HMM = characters
- Observations = typed images of characters segmented from the image
  - $v_\alpha$
- Note that there is an infinite number of observations

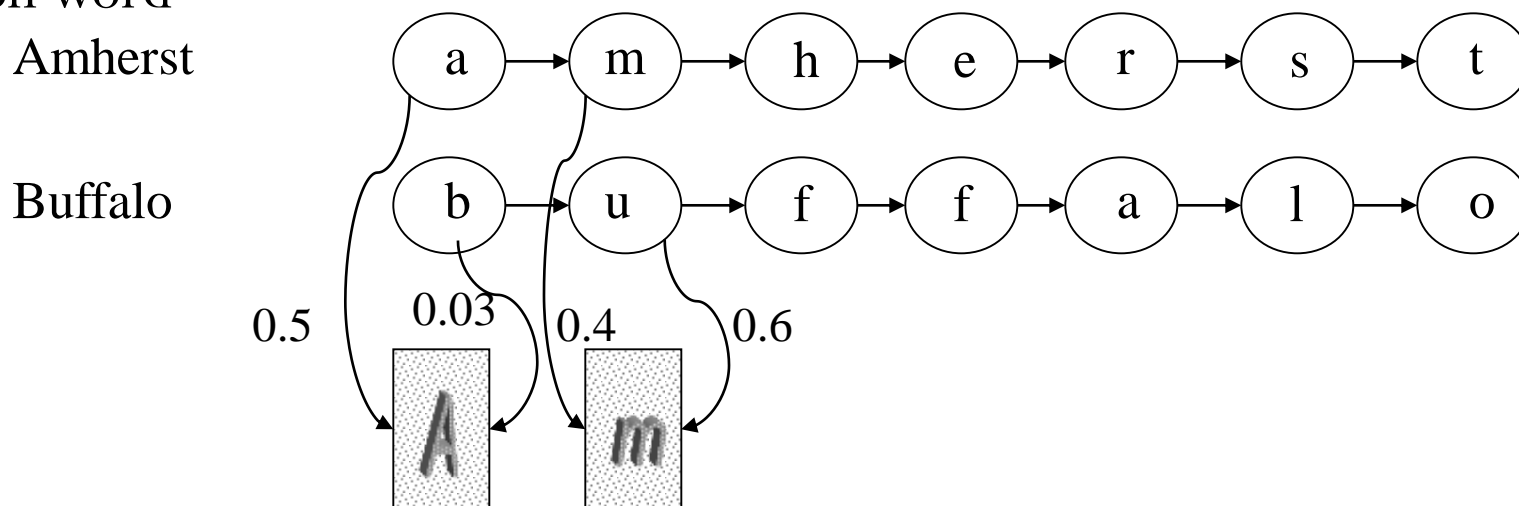
- Observation probabilities = character recognizer scores

$$B = (b_i(v_\alpha)) = (P(v_\alpha | s_i))$$

- Transition probabilities will be defined differently in two subsequent models

# Word recognition example(3)

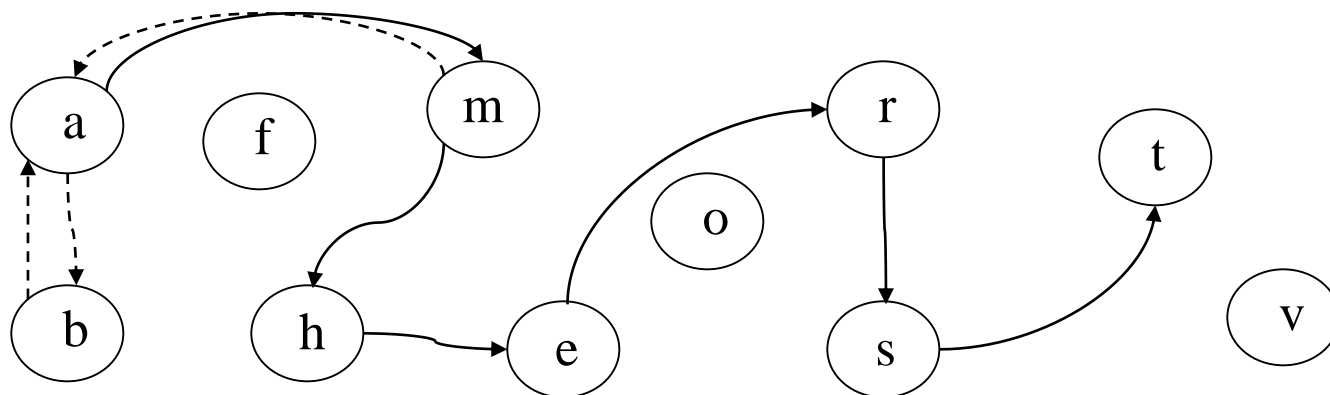
- If lexicon is given, we can construct separate HMM models for each lexicon word



- Here recognition of word image is equivalent to the problem of evaluating a few HMM models
- This is an application of **Evaluation problem**

# Word recognition example(4)

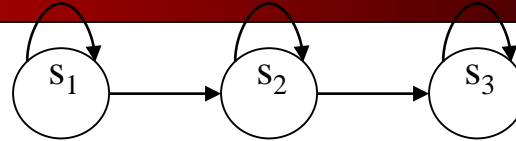
- We can construct a single HMM for all words
- Hidden states = all characters in the alphabet
- Transition probabilities and initial probabilities: calculated from language model
- Observations and observation probabilities (as before)



- *Here we have to determine the best sequence of hidden states, the one that most likely produced word image*
- This is an application of *Decoding problem*

# Exercise: Character recognition with HMM(1)

The structure of hidden states:

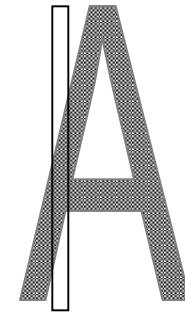


- Observation = number of islands in the vertical slice

- HMM for character 'A' :

Transition probabilities:  $\{a_{ij}\} = \begin{pmatrix} .8 & .2 & 0 \\ 0 & .8 & .2 \\ 0 & 0 & 1 \end{pmatrix}$

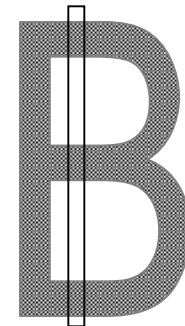
Observation probabilities:  $\{b_{jk}\} = \begin{pmatrix} .9 & .1 & 0 \\ .1 & .8 & .1 \\ .9 & .1 & 0 \end{pmatrix}$



- HMM for character 'B' :

Transition probabilities:  $\{a_{ij}\} = \begin{pmatrix} .8 & .2 & 0 \\ 0 & .8 & .2 \\ 0 & 0 & 1 \end{pmatrix}$

Observation probabilities:  $\{b_{jk}\} = \begin{pmatrix} .9 & .1 & 0 \\ 0 & .2 & .8 \\ .6 & .4 & 0 \end{pmatrix}$



# Exercise: character recognition with HMM(2)

---

- Suppose that after character image segmentation the following sequence of island numbers in 4 slices were observed:  
 $\{ 1, 3, 2, 1 \}$
- Which HMM is more likely to generate this observation sequence, HMM for 'A' or HMM for 'B' ?

# Exercise: character recognition with HMM(3)

Consider likelihood of generating given observation for each possible sequence of hidden states:

- HMM for character 'A':

Hidden state sequence	Transition probabilities	Observation probabilities
$s_1 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3$	$.8 * .2 * .2$	$* .9 * 0 * .8 * .9 = 0$
$s_1 \rightarrow s_2 \rightarrow s_2 \rightarrow s_3$	$.2 * .8 * .2$	$* .9 * .1 * .8 * .9 = 0.0020736$
$s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_3$	$.2 * .2 * 1$	$* .9 * .1 * .1 * .9 = 0.000324$
Total = 0.0023976		

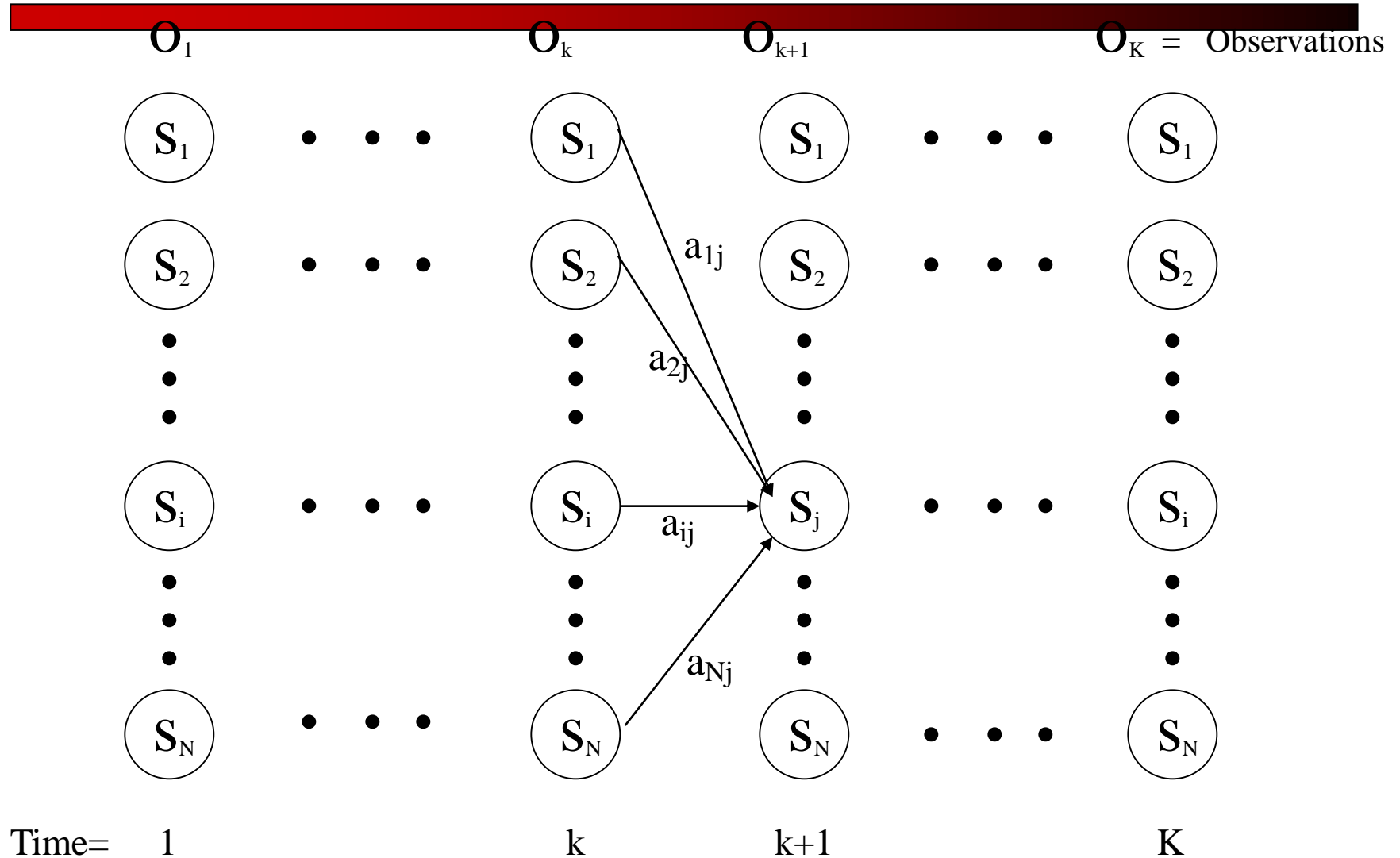
- HMM for character 'B':

Hidden state sequence	Transition probabilities	Observation probabilities
$s_1 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3$	$.8 * .2 * .2$	$* .9 * 0 * .2 * .6 = 0$
$s_1 \rightarrow s_2 \rightarrow s_2 \rightarrow s_3$	$.2 * .8 * .2$	$* .9 * .8 * .2 * .6 = 0.0027648$
$s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_3$	$.2 * .2 * 1$	$* .9 * .8 * .4 * .6 = 0.006912$
Total = 0.0096768		

# Evaluation Problem

- **Evaluation problem.** Given the HMM  $M=(A, B, \pi)$  and the observation sequence  $O=o_1 o_2 \dots o_K$ , calculate the probability that model  $M$  has generated sequence  $O$
- Trying to find probability of observations  $O=o_1 o_2 \dots o_K$  by means of considering all hidden state sequences (*as was done in example*) is impractical:  
 $N^K$  hidden state sequences-exponential complexity
- Use **Forward-Backward HMM algorithms** for efficient calculations
- Define the forward variable  $\alpha_k(i)$  as the joint probability of the partial observation sequence  $o_1 o_2 \dots o_k$  and the hidden state  $s_i$  at time  $k$ :  
$$\alpha_k(i) = P(o_1 o_2 \dots o_k, q_k = s_i)$$

# Trellis representation of an HMM





# Forward recursion for HMM

## Initialization:

$$\alpha_1(i) = P(o_1, q_1 = s_i) = \pi_i b_i(o_1), 1 \leq i \leq N$$

## • Forward recursion:

$$\begin{aligned} \alpha_{k+1}(j) &= P(o_1 o_2 \dots o_{k+1}, q_{k+1} = s_j) = \\ &\sum_i P(o_1 o_2 \dots o_{k+1}, q_k = s_i, q_{k+1} = s_j) = \\ &\sum_i P(o_1 o_2 \dots o_k, q_k = s_i) a_{ij} b_j(o_{k+1}) = \\ &[\sum_i \alpha_k(i) a_{ij}] b_j(o_{k+1}), \quad 1 \leq j \leq N, 1 \leq k \leq K-1 \end{aligned}$$

## • Termination:

$$P(o_1 o_2 \dots o_K) = \sum_i P(o_1 o_2 \dots o_K, q_K = s_i) = \sum_i \alpha_K(i)$$

## • Complexity :

$N^2K$  operations

# Backward recursion for HMM

• Define the backward variable  $\beta_k(i)$  as the joint probability of the partial observation sequence  $o_{k+1} o_{k+2} \dots o_K$  given that the hidden state at time  $k$  is  $s_i$  :  $\beta_k(i) = P(o_{k+1} o_{k+2} \dots o_K \mid q_k = s_i)$

• Initialization:

$$\beta_K(i) = 1, \quad 1 \leq i \leq N.$$

• Backward recursion:

$$\begin{aligned} \beta_k(j) &= P(o_{k+1} o_{k+2} \dots o_K \mid q_k = s_j) = \\ &\sum_i P(o_{k+1} o_{k+2} \dots o_K, q_{k+1} = s_i \mid q_k = s_j) = \\ &\sum_i P(o_{k+2} o_{k+3} \dots o_K \mid q_{k+1} = s_i) a_{ji} b_i(o_{k+1}) = \\ &\sum_i \beta_{k+1}(i) a_{ji} b_i(o_{k+1}), \quad 1 \leq j \leq N, \quad 1 \leq k \leq K-1. \end{aligned}$$

• Termination:

$$\begin{aligned} P(o_1 o_2 \dots o_K) &= \sum_i P(o_1 o_2 \dots o_K, q_1 = s_i) = \\ &\sum_i P(o_1 o_2 \dots o_K \mid q_1 = s_i) P(q_1 = s_i) = \sum_i \beta_1(i) b_i(o_1) \pi_i \end{aligned}$$

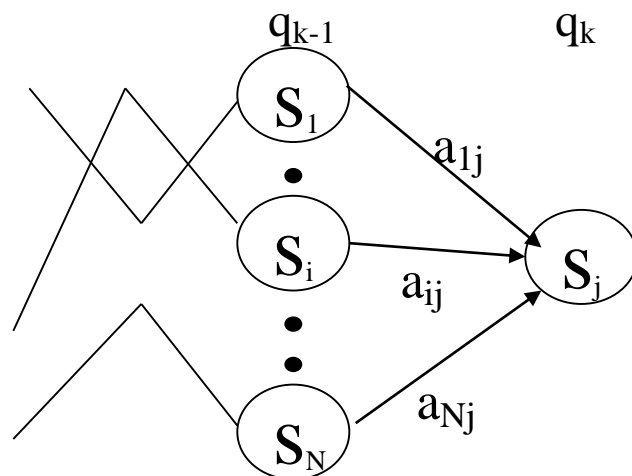
# Decoding problem

- **Decoding problem.** Given the HMM  $M=(A, B, \pi)$  and the observation sequence  $O=o_1 o_2 \dots o_K$ , calculate the most likely sequence of hidden states  $s_i$  that produced this observation sequence.
- We want to find the state sequence  $Q= q_1 \dots q_K$  which maximizes  $P(Q \mid o_1 o_2 \dots o_K)$ , or equivalently  $P(Q, o_1 o_2 \dots o_K)$ .
- Brute force consideration of all paths takes exponential time. Use efficient **Viterbi algorithm** instead.
- Define variable  $\delta_k(i)$  as the maximum probability of producing observation sequence  $o_1 o_2 \dots o_k$  when moving along any hidden state sequence  $q_1 \dots q_{k-1}$  and getting into  $q_k = s_i$ .
$$\delta_k(i) = \max P(q_1 \dots q_{k-1}, q_k = s_i, o_1 o_2 \dots o_k)$$
where max is taken over all possible paths  $q_1 \dots q_{k-1}$ .

# Viterbi algorithm (1)

- General idea:

if best path ending in  $q_k = s_j$  goes through  $q_{k-1} = s_i$  then it should coincide with best path ending in  $q_{k-1} = s_i$



- $$\delta_k(i) = \max P(q_1 \dots q_{k-1}, q_k = s_j, o_1 o_2 \dots o_k) = \max_i [ a_{ij} b_j(o_k) \max P(q_1 \dots q_{k-1} = s_i, o_1 o_2 \dots o_{k-1}) ]$$

- To backtrack best path keep info that predecessor of  $s_j$  was  $s_i$ .

# Viterbi algorithm (2)

- Initialization:

$$\delta_1(i) = \max P(q_1 = s_i, o_1) = \pi_i b_i(o_1), 1 \leq i \leq N$$

- Forward recursion:

$$\begin{aligned} \delta_k(j) &= \max P(q_1 \dots q_{k-1}, q_k = s_j, o_1 o_2 \dots o_k) = \\ &= \max_i [ a_{ij} b_j(o_k) \max P(q_1 \dots q_{k-1} = s_i, o_1 o_2 \dots o_{k-1}) ] = \\ &= \max_i [ a_{ij} b_j(o_k) \delta_{k-1}(i) ], \quad 1 \leq j \leq N, 2 \leq k \leq K. \end{aligned}$$

- Termination: choose best path ending at time K

$$\max_i [ \delta_K(i) ]$$

- Backtrack best path

*This algorithm is similar to the forward recursion of evaluation problem, with  $\Sigma$  replaced by max and additional backtracking*

# Learning problem (1)

- **Learning problem.** Given some training observation sequences  $O = o_1 o_2 \dots o_K$  and general structure of HMM (numbers of hidden and visible states), determine HMM parameters  $M = (A, B, \pi)$  that best fit training data, i.e. maximizes  $P(O | M)$
- Finding optimal parameter values- *very difficult* (no existing algorithm)
- Use *iterative expectation-maximization algorithm* to find local maximum of  $P(O | M)$ -**Baum-Welch algorithm**

# Learning problem (2)

- If training data has information about sequence of hidden states (as in word recognition example), then use maximum likelihood estimation of parameters:

$$a_{ij} = P(s_i | s_j) = \frac{\text{Number of transitions from state } s_j \text{ to state } s_i}{\text{Number of transitions out of state } s_j}$$

$$b_i(v_m) = P(v_m | s_i) = \frac{\text{Number of times observation } v_m \text{ occurs in state } s_i}{\text{Number of observation occurrences in state } s_i}$$

# Baum-Welch algorithm

---

*General idea:*

$$a_{ij} = P(s_i | s_j) = \frac{\text{Expected number of transitions from state } s_j \text{ to state } s_i}{\text{Expected number of transitions out of state } s_j}$$

$$b_i(v_m) = P(v_m | s_i) = \frac{\text{Expected number of times observation } v_m \text{ occurs in state } s_i}{\text{Expected number of total observations in state } s_i}$$

$$\pi_i = P(s_i) = \text{Expected frequency in state } s_i \text{ at time } k=1.$$



# Disadvantage of HMMs (1)

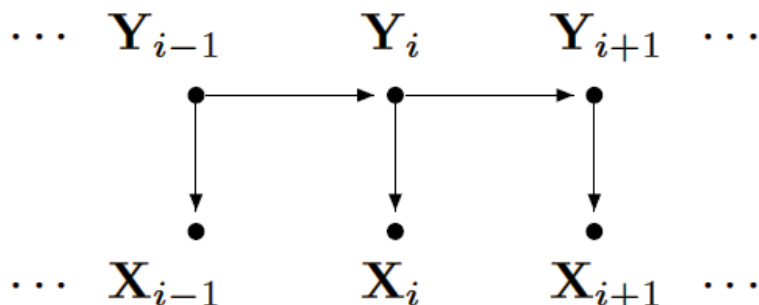
- No Rich Feature Information
  - Rich information are required
    - When  $x_k$  is complex
    - When data of  $x_k$  is sparse
- Example: Part-of-Speech (PoS) Tagging
  - How to evaluate  $P(w_k | t_k)$  for unknown words  $w_k$ ?
  - Useful features
    - Suffix, e.g., -ed, -tion, -ing, etc.
    - Capitalization
- Generative Model
  - Parameter estimation: maximize the joint likelihood of training examples

$$\sum_{(\mathbf{x}, \mathbf{y}) \in T} \log_2 P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y})$$

# Generative Models

- Hidden Markov models (HMMs) and stochastic grammars
  - Assign a joint probability to paired observation and label sequences
  - The parameters typically trained to maximize the joint likelihood of train examples

Standard tool is the hidden Markov Model (HMM).



$$P(\mathbf{X}, \mathbf{Y}) = \prod_i P(\mathbf{X}_i | \mathbf{Y}_i) P(\mathbf{Y}_i | \mathbf{Y}_{i-1})$$

# Generative Models (cont'd)

---

- Difficulties and disadvantages
  - Need to enumerate all possible observation sequences
  - Not practical to represent multiple interacting features or long-range dependencies of the observations
  - Very strict independence assumptions on the observations

# Making use of rich domain features

- A learning algorithm is as good as its features.
  - There are *many* useful features to include in a model
  - Most of them aren't independent of each other
- |                              |                             |
|------------------------------|-----------------------------|
| ■ Identity of word           | ■ Word to left is verb      |
| ■ Ends in "-shire"           | ■ Word to left is lowercase |
| ■ Is capitalized             | ■ Is in bold font           |
| ■ Is head of noun phrase     | ■ Is in hyperlink anchor    |
| ■ Is in a list of city names | ■ Other occurrences in doc  |
| ■ Is under node X in WordNet | ■ ...                       |