

# Stochastic Processes



**Week 09 (Version 1.0)**

**Markov Chains & HMMs**

Hamid R. Rabiee

Fall 2023

# Overview

Markov Property

Markov Chains Definition

Markov Chains Stationary Property

Markov Chains Paths

Markov Chains Classification of States

Markov Chains Steady States

Hidden Markov Models

# Markov Property

- A discrete process has the **Markov** property if given its value at time  $t$ , the value at time  $t+1$  is independent of values at times before  $t$ .

That is:

$$\begin{aligned} Pr(X_{t+1} = x_{t+1} | X_t = x_t, X_{t-1} = x_{t-1}, \dots, X_1 = x_1) \\ = Pr(X_{t+1} = x_{t+1} | X_t = x_t) \end{aligned}$$

For all  $t, x_{t+1}, x_t, x_{t-1}, x_{t-2}, \dots, x_1$ .

# Stationary Property

- A Markov Process is called **stationary** if:

$$\Pr(X_{t+1} = u | X_t = v) = \Pr(X_1 = u | X_0 = v) \text{ for all } t.$$

- The evolution of stationary processes don't change over time.
- For defining the complete joint distribution of a stationary Markov Process it is sufficient to define  $\Pr(X_1 = u | X_0 = v)$  and  $\Pr(X_0 = v)$  for all  $u$  and  $v$ .
- We will mainly consider **stationary Markov processes** here.

# Markov Process Types

- There exist two types of Markov processes based on domain of  $X_t$  values:
  - Discrete
  - Continuous
- **Discrete Markov** processes are called “**Markov Chains**” (MC).

# Markov Process Types

State Space	Type of Parameter	
	Discrete	Continuous
Discrete	Discrete-Time Markov Chain	Continuous-Time Markov Chain
Continuous	Discrete-Time Markov Process	Continuous-Time Markov Process

- In this course we will focus on stationary MCs.

## Example (Coin Tossing Game)

- Consider a single player game in which at every step a biased coin is tossed and according to the result, the score will be increased or decreased by one point.
- The game ends if either the score reaches 100 (winning) or -100 (losing).
- Score of the player at each step  $t \geq 0$  is a random variable and the sequence of scores as the game progresses forms a random process  $X_0, X_1, \dots, X_t$ .

# Example (Coin Tossing Game)

- It is easy to verify that  $X$  is a stationary Markov chain: At the end of each step the score solely depends on the current score  $s_c$  and the result of tossing the coin (which is independent of time and previous tosses).

- Stating this mathematically (for  $s_c \notin \{-100, 100\}$ ):

$$Pr(X_{t+1} = s | X_t = s_c, X_{t-1} = s_{t-1}, \dots, X_0 = 0)$$

$$= \begin{cases} p & ; s = s_c + 1 \\ 1 - p & ; s = s_c - 1 \\ 0 & ; otherwise \end{cases} \quad \begin{array}{l} \text{Independent of } t \\ \text{and } s_0, \dots, s_{t-1} \end{array}$$

$$= Pr(X_{t+1} = s | X_t = s_c) = Pr(X_1 = s | X_0 = s_c)$$

- If value of  $p$  was subject to change in time, the process would not be stationary (in the formulation we would have  $p_t$  instead of  $p$ ).



# Transition matrix

- According to the Markov property and stationary property, at each time step the process moves according to a fixed transition matrix:

$$P(X_{t+1} = j | X_t = i) = p_{ij}$$

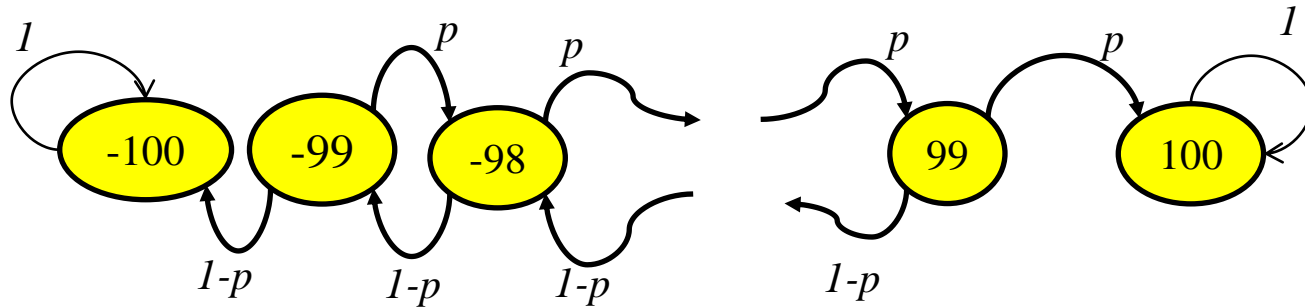
- **Stochastic matrix:** Rows sum up to 1.  
**Double stochastic matrix:** Rows and columns sum up to 1.

# State Graph

- It is convenient to **visualize a stationary Markov Chain** with a transition diagram:
  - A node represents a possible value of  $X_t$  (state). At each time  $t$ , we say the process is in state  $s$  if  $X_t=s$ .
  - Each edge represents the probability of going from one state to another (we omit edges with zero weight).
  - We should also specify the vector of initial probabilities  $\pi = (\pi_1, \dots, \pi_n)$  where  $\pi_i = \Pr(X_0 = i)$ .
- A **stationary discrete process** could be described as a person walking randomly on a graph (considering each step to depend only on the state he/she is currently in). The resulted path is called a “**Random Walk**”.

# Example

- The transition diagram of the coin tossing game is:

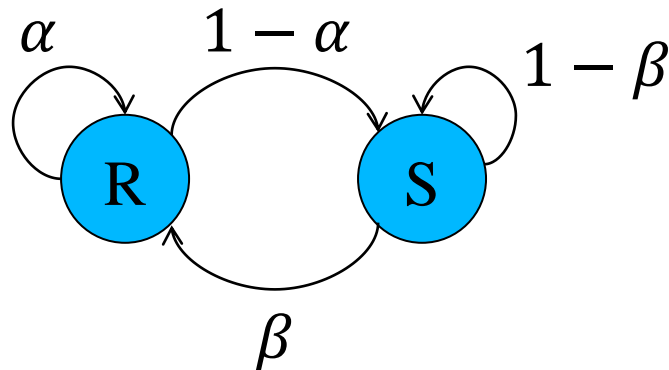


- We modeled winning and losing by states which when we get into, we never get out.
- Note that if the process was not stationary we were not able to visualize it in this way: For example consider the case that  $p$  is changing in time.

# Example 1 (Modeling Weather)

- Example: Assume each day is sunny or rainy. If a day is rainy, the next day is rainy with probability  $\alpha$  (and sunny with probability  $1 - \alpha$ ). If the day is sunny, the next day is rainy with probability  $\beta$  (and sunny with probability  $1 - \beta$ ).

$$S = \{\text{rainy, sunny}\}, \quad P = \begin{bmatrix} \alpha & 1 - \alpha \\ \beta & 1 - \beta \end{bmatrix}$$



## Exampler 2 (Modeling Weather)

Suppose that whether or not it rains today depends on previous weather conditions through the last two days.

Suppose that:

- if it has rained for the past two days, then it will rain tomorrow with probability 0.7
- if it rained today but not yesterday, then it will rain tomorrow with probability 0.5
- if it rained yesterday but not today, then it will rain tomorrow with probability 0.4
- if it has not rained in the past two days, then it will rain tomorrow with probability 0.2.

## Examplem 2 (Modeling Weather)

If we let the state at time  $n$  depend only on whether or not it is raining at time  $n$ , then the preceding model is not a Markov chain.

We can transform this model into a Markov chain by saying that the state at any time is determined by the weather conditions during both that day and the previous day:

**state 0** if it rained both today and yesterday,

**state 1** if it rained today but not yesterday,

**state 2** if it rained yesterday but not today,

**state 3** if it did not rain either yesterday or today.

## Example 2 (Modeling Weather)

	t-1	t	t+1 (p(R))
$S_0$	R	R	0.7
$S_1$	S	R	0.5
$S_2$	R	S	0.4
$S_3$	S	S	0.2

$$\begin{array}{c} S_0 \\ S_1 \\ S_2 \\ S_3 \end{array} \begin{bmatrix} S_0 & S_1 & S_2 & S_3 \\ 0.7 & 0 & 0.3 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.4 & 0 & 0.6 \\ 0 & 0.2 & 0 & 0.8 \end{bmatrix}$$

# The Chapman-Kolmogorov Equation

- Define the  $n$ -step transition  $p_{ij}^{(n)}$  as the probability that starting from state  $i$ , the process stops at state  $j$  after  $n$  time steps:

$$p_{ij}^{(n)} = P\{X_{n+m}=j \mid X_m=i\}$$

- Then the Chapman-Kolmogorov equation is given by:

$$p_{ij}^{(n+m)} = \sum_{k=0}^{\infty} p_{ik}^{(n)} p_{kj}^{(m)}$$

- Corollary 1:  $P^{(n)}$  can be calculated by:  $P^{(n)} = P^n$
- Corollary 2: If the process starts at time 0 with distribution  $\pi$  on the states then after  $n$  steps the distribution is  $\pi P^n$ .



# The Chapman-Kolmogorov Equation

$$\begin{aligned} P_{ij}^{n+m} &= P(X_{n+m} = j | X_0 = i) \\ &= \sum_{k=1}^v P(X_{n+m} = j, X_n = k | X_0 = i) \\ &= \sum_{k=1}^v P(X_{n+m} = j | X_n = k) P(X_n = k | X_0 = i) \\ &= \sum_{k=1}^v P_{ik}^n P_{kj}^m = (P^m P^n)(ij) \end{aligned}$$

# The Chapman-Kolmogorov Equation

- Corollary 1:  $P^{(n)}$  can be calculated by:  $P^{(n)} = P^n$

$$P^{(2)} = P^{(1+1)} = P^{(1)}P^{(1)} = P \cdot P = P^2$$

$$P^{(n)} = P^{(n-1+1)} = P^{(n-1)}P^{(1)} = P^{n-1}P = P^n$$

$$P^{(1)} = \begin{pmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{pmatrix}$$

$$P^{(2)} = P^2 = \begin{pmatrix} 0.61 & 0.39 \\ 0.52 & 0.48 \end{pmatrix}$$

$$P^{(4)} = P^2P^2 = \begin{pmatrix} 0.57 & 0.43 \\ 0.57 & 0.43 \end{pmatrix}$$

## Example 2 (Modeling Weather)

- If **Monday** and **Tuesday** are raining, what is the probability of raining on **Thursday**?

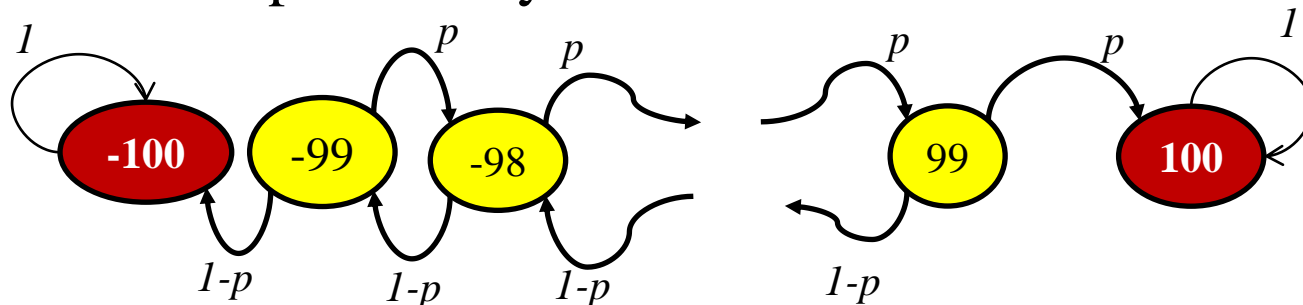
$$\begin{array}{c}
 S_0 \\
 S_1 \\
 S_2 \\
 S_3
 \end{array}
 \begin{array}{cccc}
 S_0 & S_1 & S_2 & S_3 \\
 \left[ \begin{array}{cccc}
 0.7 & 0 & 0.3 & 0 \\
 0.5 & 0 & 0.5 & 0 \\
 0 & 0.4 & 0 & 0.6 \\
 0 & 0.2 & 0 & 0.8
 \end{array} \right]
 \end{array}$$

$$P^2 = P^2 = \begin{bmatrix} 0.7 & 0 & 0.3 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.4 & 0 & 0.6 \\ 0 & 0.2 & 0 & 0.8 \end{bmatrix}^2 = \begin{bmatrix} 0.49 & 0.12 & 0.21 & 0.18 \\ 0.35 & 0.2 & 0.15 & 0.3 \\ 0.2 & 0.12 & 0.2 & 0.48 \\ 0.1 & 0.16 & 0.1 & 0.64 \end{bmatrix}$$

Since rain on Thursday is equivalent to the process being in either state 0 or state 1 on Thursday, the desired probability is given by  $P^2_{00} + P^2_{01} = 0.49 + 0.12 = 0.61$ .

# Absorbing Markov Chain

- An **absorbing state** is one in which the probability that the process remains in that state once it enters the state is 1 (i.e.,  $p_{ii} = 1$ ).
- A Markov chain is **absorbing** if it has at least one absorbing state, and if from every state it is possible to go to an absorbing state (not necessarily in one step).
- An absorbing Markov chain will eventually enter one of the absorbing states and never leave it.
- Example: The 100 and -100 states in coin tossing game (Note: After playing long enough, the player will either win or lose with probability 1).



# Absorption Theorem

- In an absorbing Markov chain the probability that the process will be absorbed is 1.
- Proof: From each non-absorbing state  $s_j$  it is possible to reach an absorbing state starting from  $s_j$ . Therefore there exists  $p$  and  $m$ , such that the probability of not absorbing after  $m$  steps is at most  $p$ , in  $2m$  steps at most  $p^2$ , etc.
- Since the probability of not being absorbed is monotonically decreasing, we have:

$$\lim_{n \rightarrow \infty} P(\text{not absorbed after } n \text{ steps}) = 0$$

# Classification of States

- **Accessibility:** State  $j$  is said to be accessible from state  $i$  if starting in  $i$  it is possible that the process will ever enter state  $j$ :  $(P^n)_{ij} > 0$ .
- **Communication:** Two states  $i$  and  $j$  that are accessible to each other are said to communicate.
  - Every node communicates with itself:
$$p_{ii}^{(0)} = P(X_0 = i | X_0 = i) = 1$$
  - Communication is an equivalence relation: It divides the state space up into a number of **separate classes** in which every pair of states communicate.
- The Markov chain is said to be **irreducible** if it has only one class.

# Transient and Recurrent states

- For any state  $i$  we let  $f_i$  denote the probability that, starting in state  $i$ , the process will ever reenter state  $i$ :

$$f_i = \Pr(\exists n: X_n = i \mid X_0 = i)$$

- State  $i$  is said to be **recurrent** if  $f_i = 1$  and **transient** if  $f_i < 1$ .
- **Theorem 1:** State  $i$  is recurrent if and only if, starting in state  $i$ , the expected number of time periods that the process is in state  $i$  is infinite:
- **Corollary 1:** A transient state will only be visited a finite number of times.

Proof:  $E[\text{size}(\{n: X_n = i\}) \mid X_0 = i]$

$$= \sum_{k=1}^{\infty} k \times \Pr(\text{size}(\{n: X_n = i\}) = k \mid X_0 = i)$$

$$= \dots + \infty \times \Pr(\text{size}(\{n: X_n = i\}) = \infty \mid X_0 = i) < \infty$$

$$\Rightarrow \Pr(\text{size}(\{n: X_n = i\}) = \infty \mid X_0 = i) = 0$$

# Transient and Recurrent states

- **Theorem 2:** State  $i$  is recurrent iff

$$\sum_{n=1}^{\infty} (P^n)_{ii} = \infty.$$

(Look at the reference book for proof).

- **Corollary 2:** A finite state Markov chain has at least one recurrent state.

If all states are transient there will be a finite number of steps that after that the process should not be in any state (which is a contradiction).



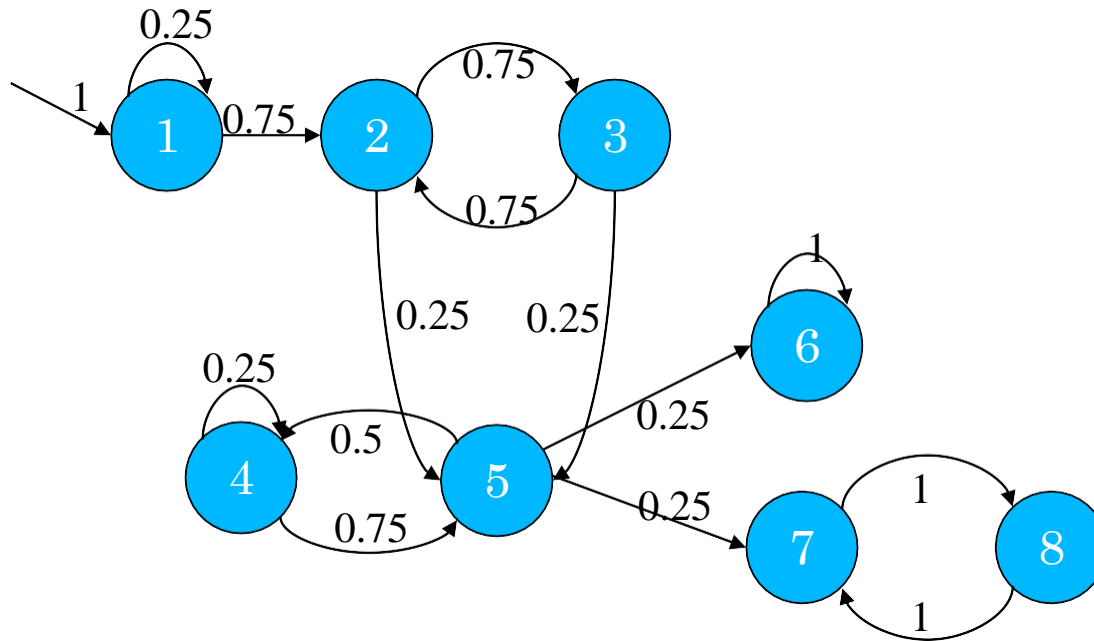
# Ergodic States

- If state  $i$  is recurrent, then it is said to be **positive recurrent** if, starting in  $i$ , the expected time until the process returns to state  $i$  is finite.
- In a finite-state MC, all recurrent states are positive recurrent.
- State  $i$  is said to have **period**  $d(i)$  if  $(p^n)_{ii} = 0$  whenever  $n$  is not divisible by  $d$ , and  $d$  is the largest integer with this property.
- Equivalently:  $d = \gcd\{n: \Pr(X_n = i \mid X_0 = i) > 0\}$
- A state with period 1 is said to be **aperiodic**.
- We call an MC aperiodic if all its states are aperiodic.

# Ergodic States

- A state  $i$  is said to be **ergodic** if it is aperiodic and positive recurrent.
- Period, recurrence and positive recurrence are all **class properties**. They are shared between states of a class.

# Example



*Classes:*  $\{1\}, \{2,3\}, \{4,5\}, \{6\}, \{7,8\}$

*Recurrent states:* 6,7,8

*Absorbing states:* 6

*Ergodic states:* 6

*Periodic states:* 2, 3, 7, 8:     *Period* 2

# Example

As time goes to infinity, what is the probability of being in each class?

**Answer:**

- The process will be in transient classes  $\{1\}, \{2,3\}, \{4,5\}$  with probability 0.
- Problem is symmetric for entering classes  $\{6\}$  and  $\{7,8\}$  as their only input edge is one from 5 with equal probabilities 0.25, and once it enters them, there is no way out.
- Therefore, at infinity probability of being in each of these two classes is 0.5.

# Example

If the process is absorbed in  $\{7,8\}$  (which could be considered as an absorbing super state) what will happen after that?

**Answer:**

- It will alternate between 7 and 8 to the end.  
Therefore, at time  $t \rightarrow \infty$  probability of being in 7 (or 8) will depend on the parity of  $t$ . In general finding the exact behavior of non-ergodic states as  $t \rightarrow \infty$  is not easy.

# Steady State

**Theorem:** For an irreducible ergodic Markov chain  $\lim_{n \rightarrow \infty} (P^n)_{ij}$  exists and is independent of  $i$ . Furthermore, letting:

$$\pi_j^* = \lim_{n \rightarrow \infty} (P^n)_{ij}$$

Then  $\pi^* = (\pi_1^*, \dots, \pi_d^*)^t$  is unique nonnegative solution of:

$$\begin{cases} \pi^* = \pi^* P \\ \sum_{j=1}^d \pi_j = 1 \end{cases}$$

- If the ergodicity condition is removed,  $\lim_{n \rightarrow \infty} (P^n)_{ij}$  does not exist in general, but the given equations yet have a unique solution  $\pi^* = (\pi_1^*, \dots, \pi_d^*)^t$  in which  $\pi_j^*$  will be equal to the long run proportion of time that the Markov chain is in state  $j$ .

# Example

- Consider the weather model example discussed before. We want to see how will the weather be when time goes to infinity:

$$P = \begin{bmatrix} \alpha & 1 - \alpha \\ \beta & 1 - \beta \end{bmatrix}$$

$$\begin{cases} \pi_0^* = \alpha\pi_0^* + \beta\pi_1^* \\ \pi_1^* = (1 - \alpha)\pi_0^* + (1 - \beta)\pi_1^* \\ \pi_0^* + \pi_1^* = 1 \end{cases} \quad \begin{array}{l} \text{One of these equations} \\ \text{is redundant. (why?)} \end{array}$$

- Which yields that  $\pi_0^* = \frac{\beta}{1+\beta-\alpha}$  and  $\pi_1^* = \frac{1-\alpha}{1+\beta-\alpha}$ .
- Exercise: In each of the following cases investigate the existence of solution and its meaning:
  - 1)  $\alpha=0$  and  $\beta = 1$
  - 2)  $\alpha=1$  and  $\beta = 0$

# **Introduction to Hidden Markov Models**



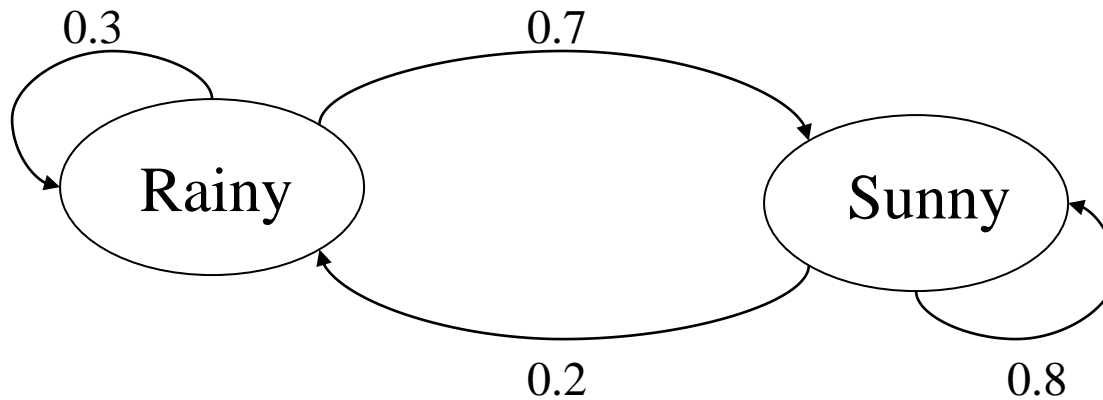
# Markov Models

- 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 a Markov model, the following probabilities have to be specified: transition probabilities  $a_{ij} = P(s_i \mid s_j)$  and initial probabilities  $\pi_i = P(s_i)$

# Example of Markov Model

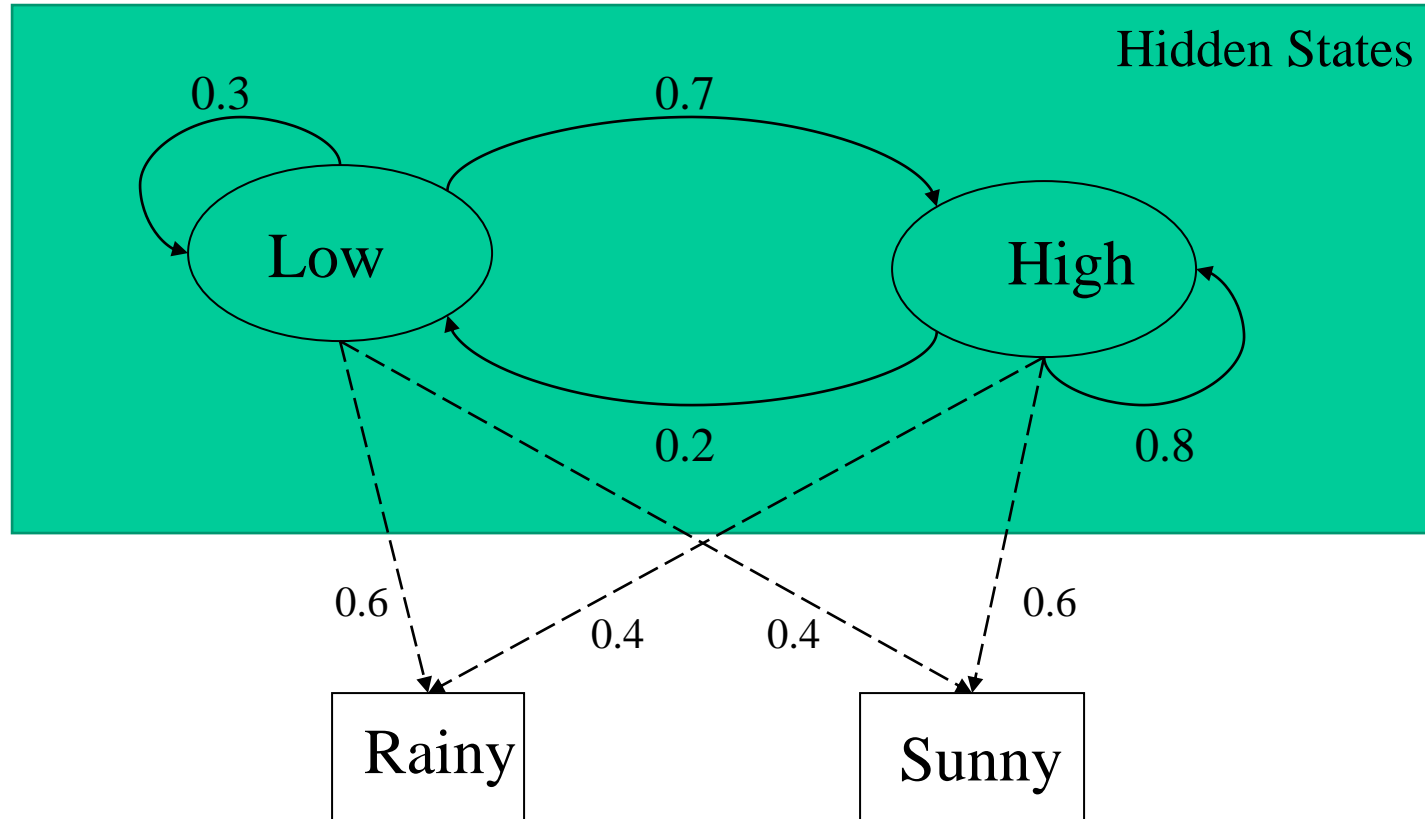


- Two states : ‘Rainy’ and ‘Sunny’.
- Transition probabilities:  $P(\text{‘Rainy’}|\text{‘Rainy’})=0.3$  ,  
 $P(\text{‘Sunny’}|\text{‘Rainy’})=0.7$  ,  $P(\text{‘Rainy’}|\text{‘Sunny’})=0.2$  ,  
 $P(\text{‘Sunny’}|\text{‘Sunny’})=0.8$
- Initial probabilities: say  $P(\text{‘Rainy’})=0.4$  ,  $P(\text{‘Sunny’})=0.6$  .

# 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:  
$$P(s_{ik} | s_{i1}, s_{i2}, \dots, s_{ik-1}) = P(s_{ik} | s_{ik-1})$$
- States are not visible, but each state randomly generates one of M observations (or visible states)  $\{v_1, v_2, \dots, v_M\}$
- To define hidden Markov model, the following probabilities have to be specified: matrix of transition probabilities  $\mathbf{A}=(a_{ij})$ ,  $a_{ij}= P(s_i | s_j)$  , matrix of observation probabilities  $\mathbf{B}=(b_i(v_m))$ ,  $b_i(v_m) = P(v_m | s_i)$  and a vector of initial probabilities  $\boldsymbol{\pi}=(\pi_i)$ ,  $\pi_i = P(s_i)$  . Model is represented by  $\mathbf{M}=(\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$ .

# Example of Hidden Markov Model



# Example of Hidden Markov Model

- Two states : ‘Low’ and ‘High’ atmospheric pressure.
- Two observations : ‘Rainy’ and ‘Sunny’.
- 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 :  $P(\text{‘Rainy’}|\text{‘Low’})=0.6$  ,  
 $P(\text{‘Sunny’}|\text{‘Low’})=0.4$  ,  $P(\text{‘Rainy’}|\text{‘High’})=0.4$  ,  
 $P(\text{‘Sunny’}|\text{‘High’})=0.3$  .
- 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, {'Sunny', 'Rainy'}.

- Consider all possible hidden state sequences:

$$P(\{\text{'Sunny'}, \text{'Rainy'}\}) = P(\{\text{'Sunny'}, \text{'Rainy'}\}, \{\text{'Low'}, \text{'Low'}\}) + P(\{\text{'Sunny'}, \text{'Rainy'}\}, \{\text{'Low'}, \text{'High'}\}) + P(\{\text{'Sunny'}, \text{'Rainy'}\}, \{\text{'High'}, \text{'Low'}\}) + P(\{\text{'Sunny'}, \text{'Rainy'}\}, \{\text{'High'}, \text{'High'}\})$$

where first term is :

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

# 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.

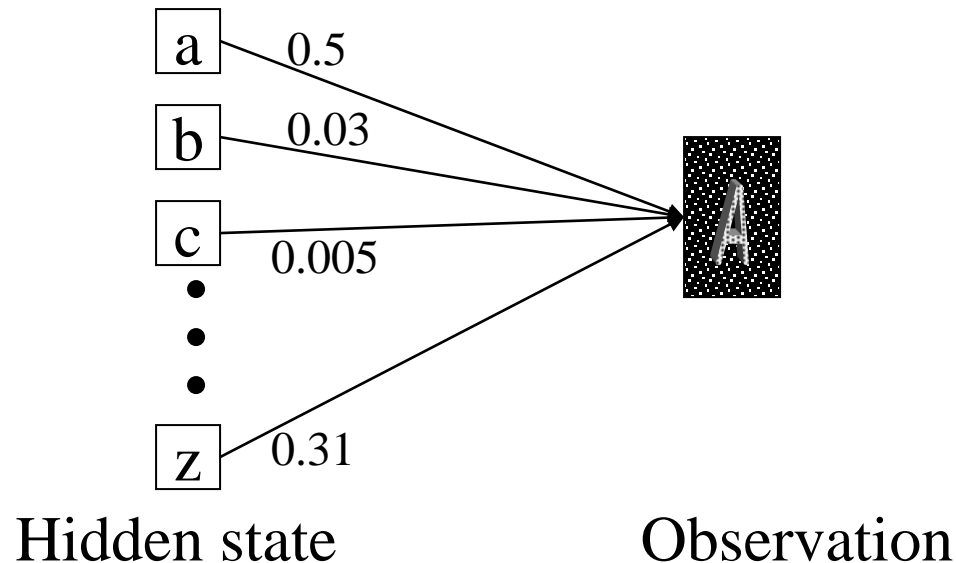
*$O=o_1 \dots o_K$  denotes a sequence of observations  $o_k \in \{v_1, \dots, v_M\}$ .*

# 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).

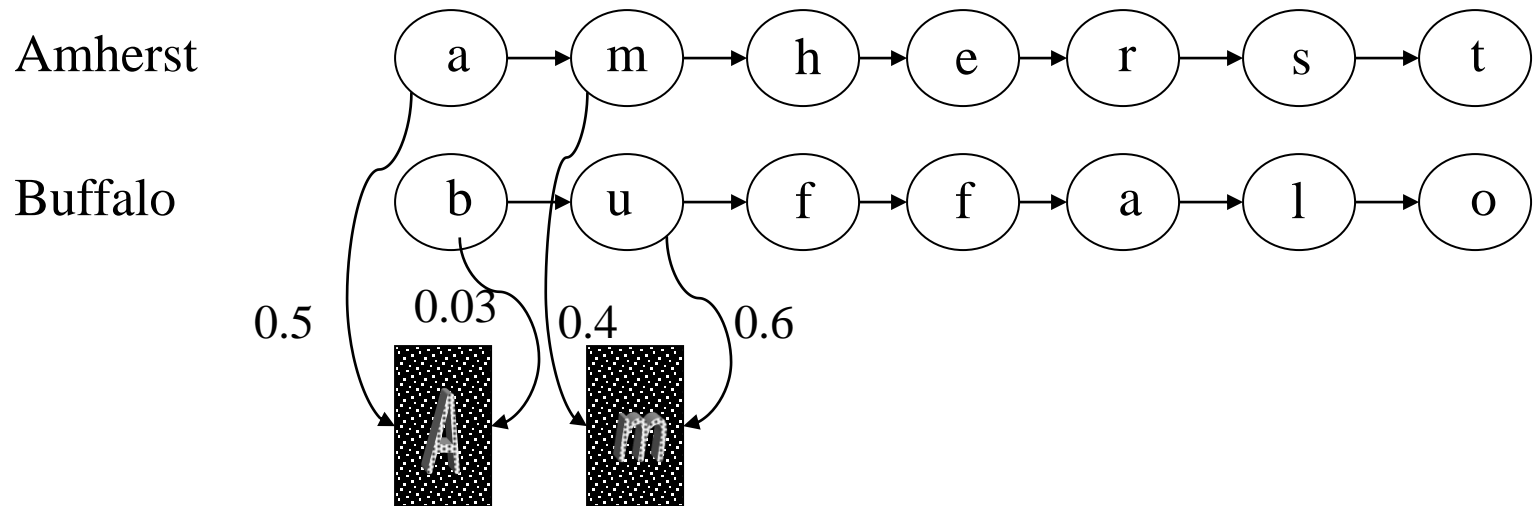
- Hidden states of HMM = characters.
- Observations = typed images of characters segmented from the image  $v_\alpha$ . Note that there is an infinite number of observations
- Observation probabilities = character recognizer scores.

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

- Transition probabilities will be defined differently in two subsequent models.

# Word recognition example(3).

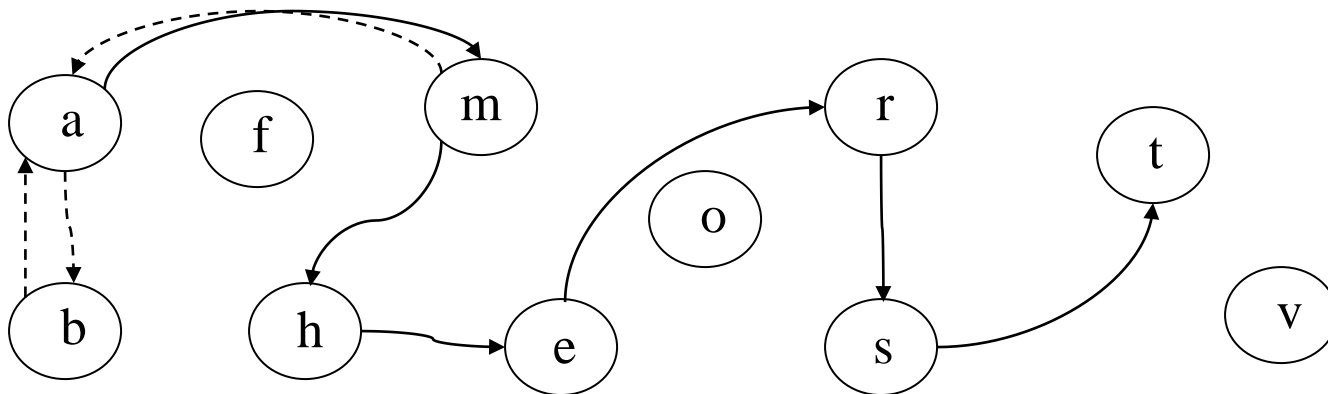
- If lexicon is given, we can construct separate HMM models for each lexicon word.



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

# Word recognition example(4).

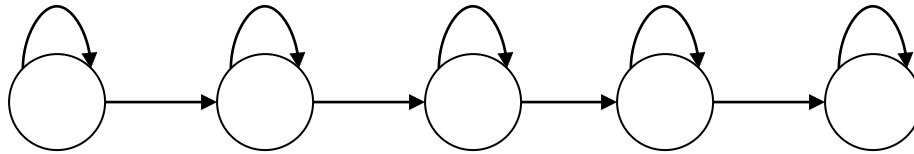
- We can construct a single HMM for all words.
- Hidden states = all characters in the alphabet.
- Transition probabilities and initial probabilities are calculated from language model.
- Observations and observation probabilities are as before.



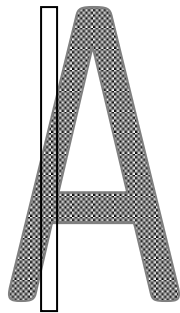
- Here we have to determine the best sequence of hidden states, the one that most likely produced word image.
- This is an application of **Decoding problem**.

# Character recognition with HMM example.

- The structure of hidden states is chosen.



- Observations are feature vectors extracted from vertical slices.



- Probabilistic mapping from hidden state to feature vectors:
  1. use mixture of Gaussian models
  2. Quantize feature vector space.

# 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  $O$ .

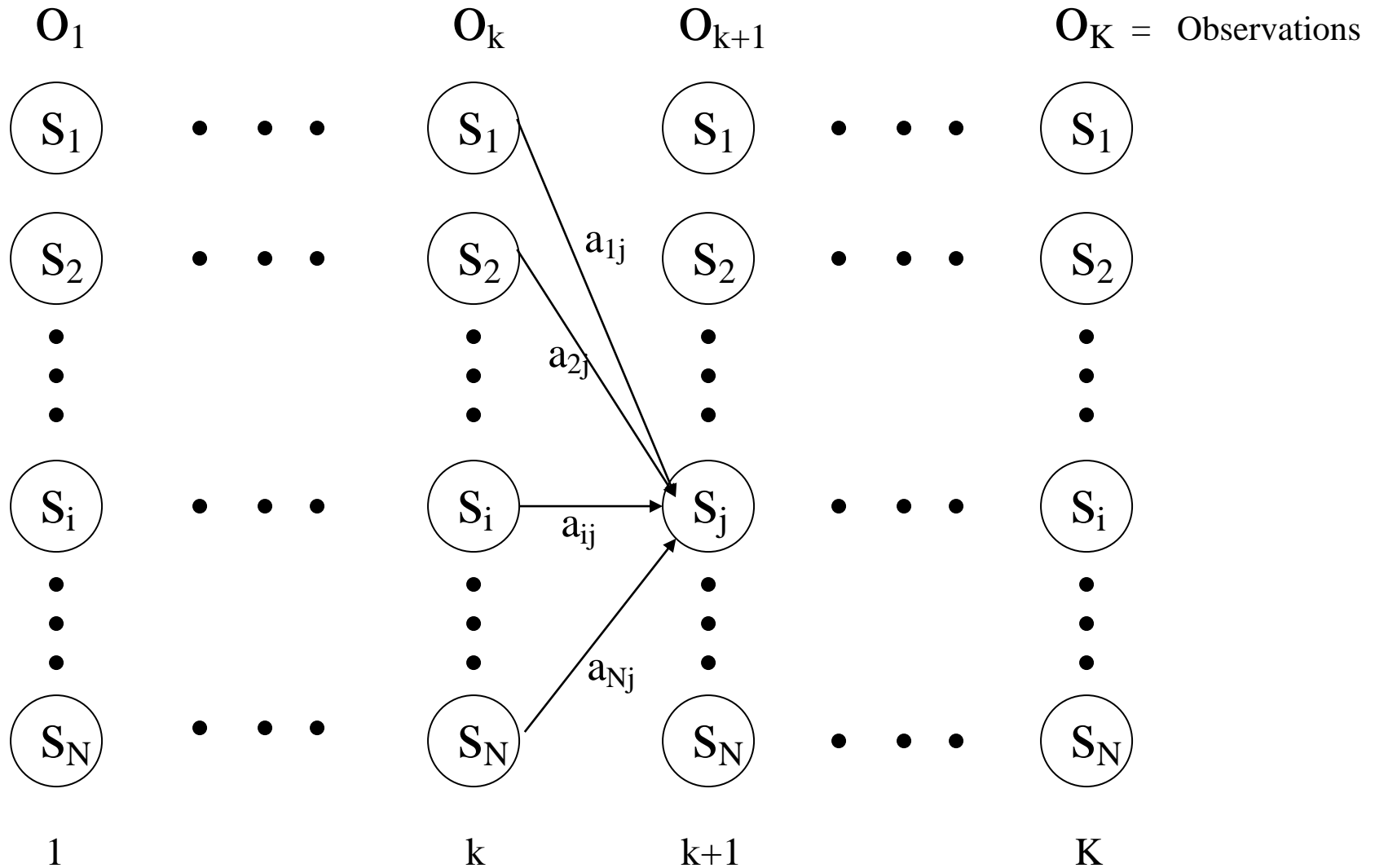
• Trying to find probability of observations  $O=O_1 O_2 \dots O_K$  by means of considering all hidden state sequences (as was done in example) is impractical:

$N^K$  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 that the hidden state at time  $k$  is  $S_i$  :  $\alpha_k(i) = P(O_1 O_2 \dots O_k, q_k = S_i)$

# Trellis representation of an HMM



# Forward recursion for HMM

- Initialization:

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

- Forward recursion:

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

- 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  $s_i$  :  $\beta_k(i) = P(o_{k+1} o_{k+2} \dots o_K | q_k = s_i)$

- Initialization:

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

- Backward recursion:

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

- Termination:

$$\begin{aligned} P(o_1 o_2 \dots o_K) &= \sum_i P(o_1 o_2 \dots o_K, q_1 = s_i) = \\ &= \sum_i P(o_1 o_2 \dots o_K | q_1 = s_i) P(q_1 = s_i) = \sum_i \beta_1(i) b_i(o_1) \pi_{i48} \end{aligned}$$



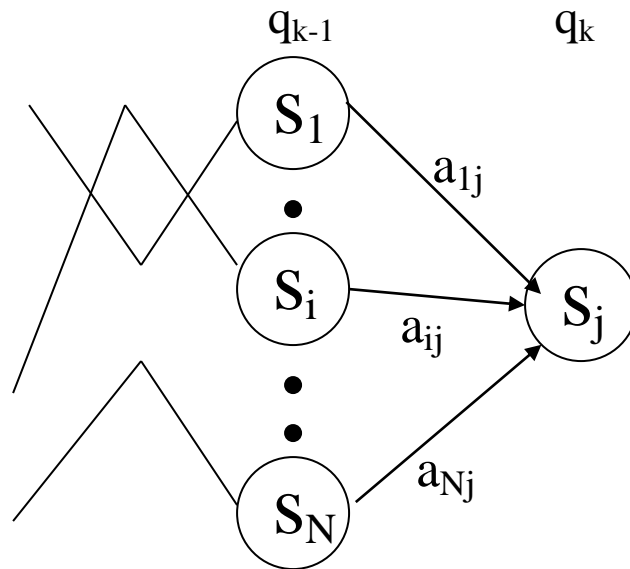
# Decoding problem

- **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  $P(Q | O_1 O_2 \dots O_K)$ , 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:

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

- 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  $\mathbf{O} = O_1 O_2 \dots O_K$  and general structure of HMM (numbers of hidden and visible states), determine HMM parameters  $\mathbf{M} = (\mathbf{A}, \mathbf{B}, \pi)$  that best fit training data, that is maximizes  $P(\mathbf{O} | \mathbf{M})$ .
- There is no algorithm producing optimal parameter values.
- Use iterative expectation-maximization algorithm to find local maximum of  $P(\mathbf{O} | \mathbf{M})$  (**Baum-Welch algorithm**).

# Expectation Maximization (EM)

Iteratively finding maximum likelihood using partial observation.

$X$ : observed data

$Z$ : unobserved data: (latent)

$\theta$ : Model parameters

$$P(X|\theta) = \int P(X, Z|\theta) dz = \int P(X|Z, \theta)P(Z|\theta) dz$$

# Expectation Maximization (EM)

E-Step (Expectation)

$Q(\theta|\theta^{(t)})$  = Expected latent log likelihood of  $\theta$

$$Q(\theta|\theta^{(t)}) = E[L(\theta; X; Z)]$$

M-Step (Maximization)

$$\theta^{(t+1)} = \operatorname{argmax}_{\theta} Q(\theta|\theta^{(t)})$$

# 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:

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

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

# Baum-Welch algorithm

General idea:

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

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

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



# Baum-Welch algorithm: expectation step(1)

- Define variable  $\xi_k(i,j)$  as the probability of being in state  $S_i$  at time  $k$  and in state  $S_j$  at time  $k+1$ , given the observation sequence  $O_1 O_2 \dots O_K$ .

$$\xi_k(i,j) = P(q_k = S_i, q_{k+1} = S_j \mid O_1 O_2 \dots O_K)$$

$$\xi_k(i,j) = \frac{P(q_k = S_i, q_{k+1} = S_j, O_1 O_2 \dots O_K)}{P(O_1 O_2 \dots O_K)} =$$

$$\frac{P(q_k = S_i, O_1 O_2 \dots O_k) a_{ij} b_j(O_{k+1}) P(O_{k+2} \dots O_K \mid q_{k+1} = S_j)}{P(O_1 O_2 \dots O_k)} =$$

$$\frac{\alpha_k(i) a_{ij} b_j(O_{k+1}) \beta_{k+1}(j)}{\sum_i \sum_j \alpha_k(i) a_{ij} b_j(O_{k+1}) \beta_{k+1}(j)}$$

# Baum-Welch algorithm: expectation step(2)

- Define variable  $\gamma_k(i)$  as the probability of being in state  $S_i$  at time  $k$ , given the observation sequence  $O_1 O_2 \dots O_K$ .

$$\gamma_k(i) = P(q_k = s_i \mid O_1 O_2 \dots O_K)$$

$$\gamma_k(i) = \frac{P(q_k = s_i, O_1 O_2 \dots O_k)}{P(O_1 O_2 \dots O_k)} = \frac{\alpha_k(i) \beta_k(i)}{\sum_i \alpha_k(i) \beta_k(i)}$$

# Baum-Welch algorithm: expectation step(3)

- We calculated  $\xi_k(i,j) = P(q_k = s_i, q_{k+1} = s_j \mid o_1 o_2 \dots o_K)$   
and  $\gamma_k(i) = P(q_k = s_i \mid o_1 o_2 \dots o_K)$
- Expected number of transitions from state  $S_i$  to state  $S_j =$   
 $= \sum_k \xi_k(i,j)$
- Expected number of transitions out of state  $S_i = \sum_k \gamma_k(i)$
- Expected number of times observation  $V_m$  occurs in state  $S_i =$   
 $= \sum_k \gamma_k(i)$ ,  $k$  is such that  $O_k = V_m$
- Expected frequency in state  $S_i$  at time  $k=1$  :  $\gamma_1(i)$  .

# Baum-Welch algorithm: maximization step

$$a_{ij} = \frac{\text{Expected number of transitions from state } S_j \text{ to state } S_i}{\text{Expected number of transitions out of state } S_j} = \frac{\sum_k \xi_k(i,j)}{\sum_k \gamma_k(i)}$$

$$b_i(v_m) = \frac{\text{Expected number of times observation } v_m \text{ occurs in state } S_i}{\text{Expected number of times in state } S_i} = \frac{\sum_k \xi_k(i,j)}{\sum_{k, o_k = v_m} \gamma_k(i)}$$

$$\pi_i = (\text{Expected frequency in state } S_i \text{ at time } k=1) = \gamma_1(i).$$

**Next Week:**

**Sampling**

**Have a good day!**