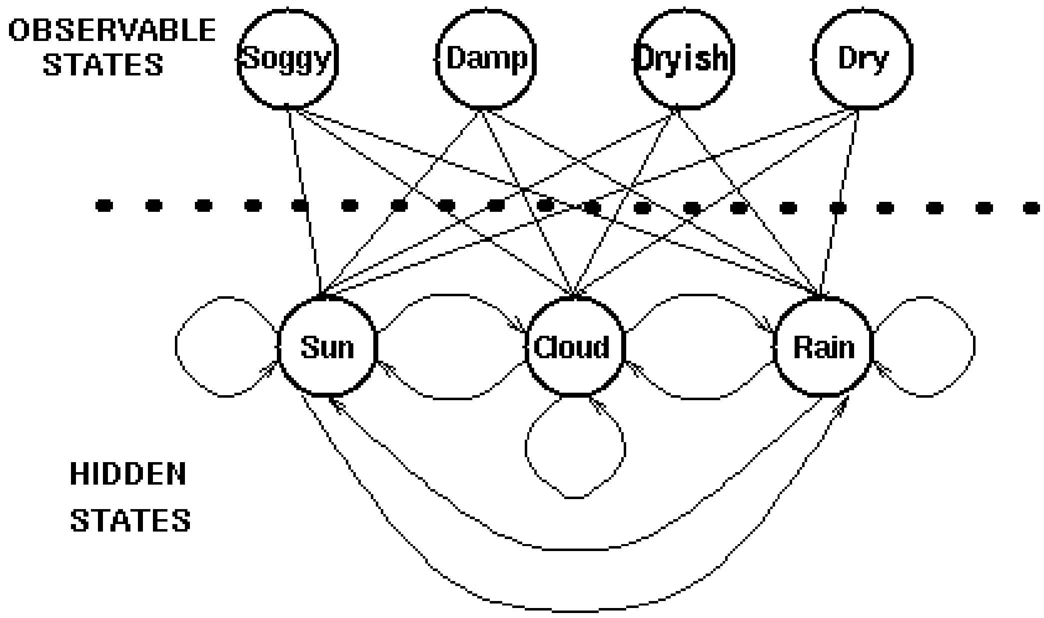
Machine Learning

CSE 6363 (Fall 2016)

Lecture 17 Hidden Markov Model

Heng Huang, Ph.D.
Department of Computer Science and Engineering

Hidden Markov Models



Hidden states : the (TRUE) states of a system that may be described by a Markov process (e.g., the weather).

Observable states : the states of the process that are 'visible' (e.g., seaweed dampness).

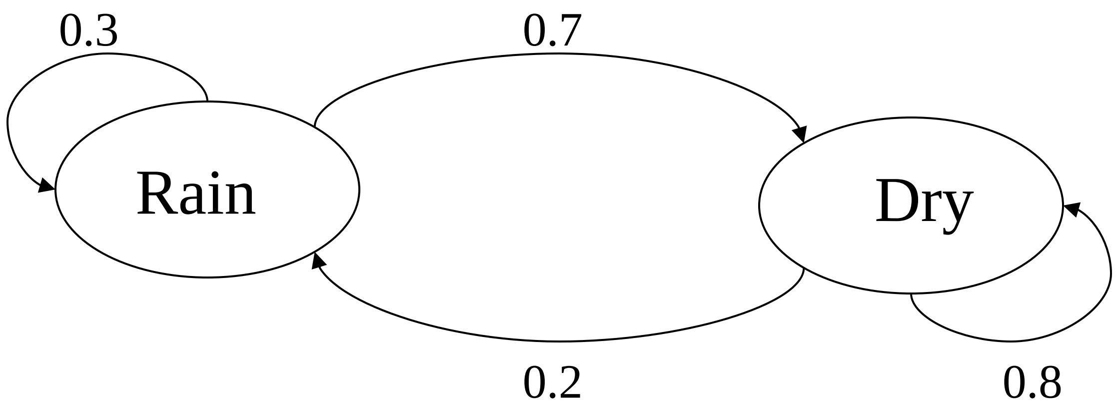
Markov Models

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

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

- To define Markov model, the following probabilities have to be specified: transition probabilities $a_{ij} = P(s_i \mid s_j)$ and initial probabilities $\pi_i = P(s_i)$
- A *Markov* model is a probabilistic model of symbol sequences in which the probability of the current event is conditioned only by the previous event.

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:

$$P(s_{i1}, s_{i2}, \ldots, s_{ik}) = P(s_{ik} \mid s_{i1}, s_{i2}, \ldots, s_{ik-1}) P(s_{i1}, s_{i2}, \ldots, s_{ik-1})$$
$$= P(s_{ik} \mid s_{ik-1}) P(s_{i1}, s_{i2}, \ldots, s_{ik-1}) = \ldots$$
$$= P(s_{ik} \mid s_{ik-1}) P(s_{ik-1} \mid s_{ik-2}) \ldots P(s_{i2} \mid s_{i1}) P(s_{i1})$$

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

P({'Dry','Dry','Rain',Rain'})
=P('Rain'|'Rain') P('Rain'|'Dry') P('Dry'|'Dry') P('Dry')
= 0.3*0.2*0.8*0.6

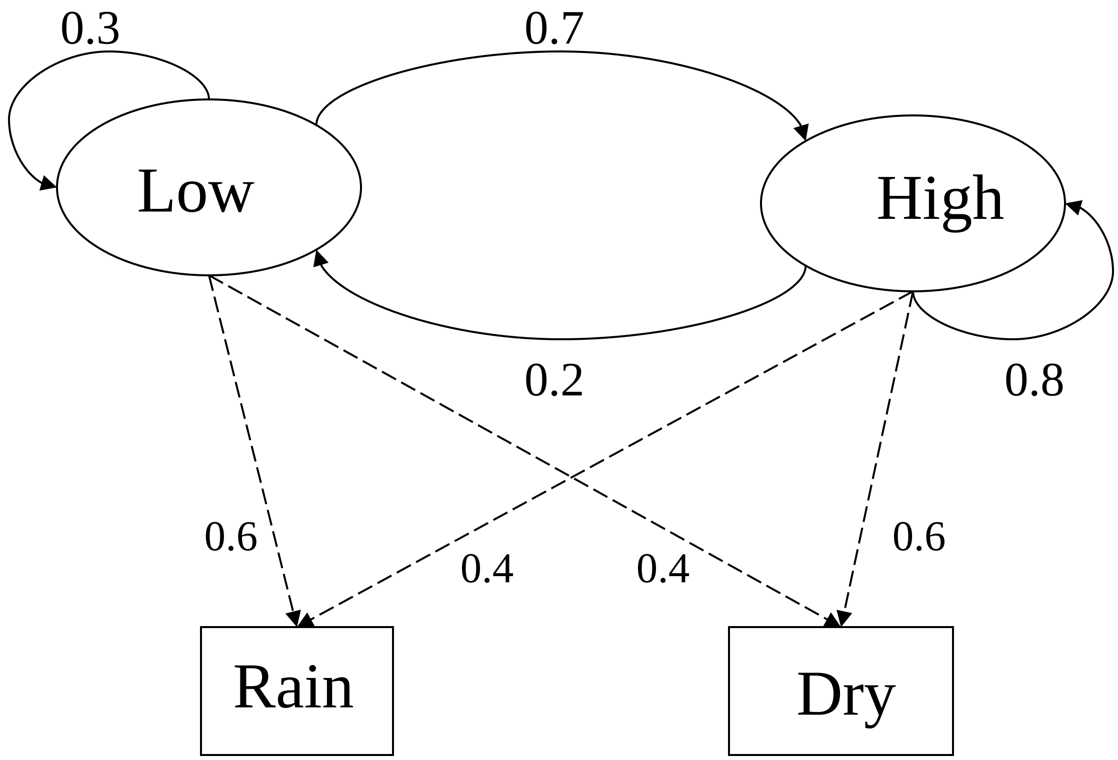
Hidden Markov models

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

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

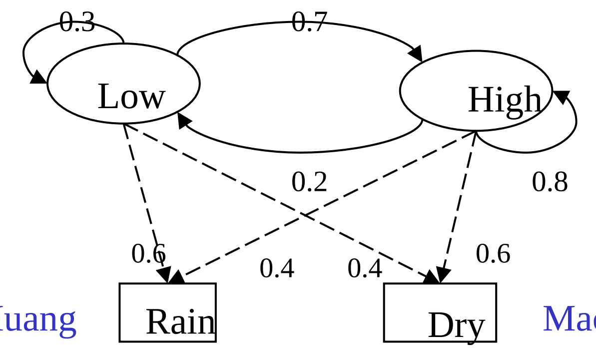
- States are not visible, but each state randomly generates one of M observations (or visible states) $\{v_1, v_2, \ldots, v_M\}$
- To define hidden Markov model, the following probabilities have to be specified: matrix of transition probabilities $A=(a_{ij})$, $a_{ij}= P(s_i \mid s_j)$, matrix of observation probabilities $B=(b_i(v_m))$, $b_i(v_m) = P(v_m \mid s_i)$ and a 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(‘Low’|‘Low’)=0.3 , P(‘High’|‘Low’)=0.7 , P(‘Low’|‘High’)=0.2, P(‘High’|‘High’)=0.8
- Observation probabilities : P(‘Rain’|‘Low’)=0.6 , P(‘Dry’|‘Low’)=0.4 , P(‘Rain’|‘High’)=0.4 , P(‘Dry’|‘High’)=0.3 .
- Initial probabilities: say P(‘Low’)=0.4 , P(‘High’)=0.6 .



Calculation of observation sequence probability

•Suppose we want to calculate a probability of a sequence of observations in our example, {'Dry','Rain'}.
•Consider all possible hidden state sequences:
P({'Dry','Rain'}) = P({'Dry','Rain'} , {'Low','Low'}) + P({'Dry','Rain'} , {'Low','High'}) + P({'Dry','Rain'} , {'High','Low'}) + P({'Dry','Rain'} , {'High','High'})

where first term is :
P({'Dry','Rain'} , {'Low','Low'})=
P({'Dry','Rain'} | {'Low','Low'}) P({'Low','Low'}) =
P('Dry'|'Low')P('Rain'|'Low') P('Low')P('Low'|'Low)
= 0.4*0.4*0.6*0.4*0.3

Main issues using HMMs :

- **Evaluation problem.** Given the HMM M=(A, B, π) and the observation sequence $O=o_1\ o_2\ ...\ o_K$, calculate the probability that model M has generated sequence O .
- **Decoding problem.** Given the HMM M=(A, B, π) and the observation sequence $O=o_1\ o_2\ ...\ o_K$, calculate the most likely sequence of hidden states s_i that produced this observation sequence O.
- **Learning problem.** Given some training observation sequences $O=o_1\ o_2\ ...\ o_K$ and general structure of HMM (numbers of hidden and visible states), determine HMM parameters M=(A, B, π) that best fit training data.

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

Evaluation Problem

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

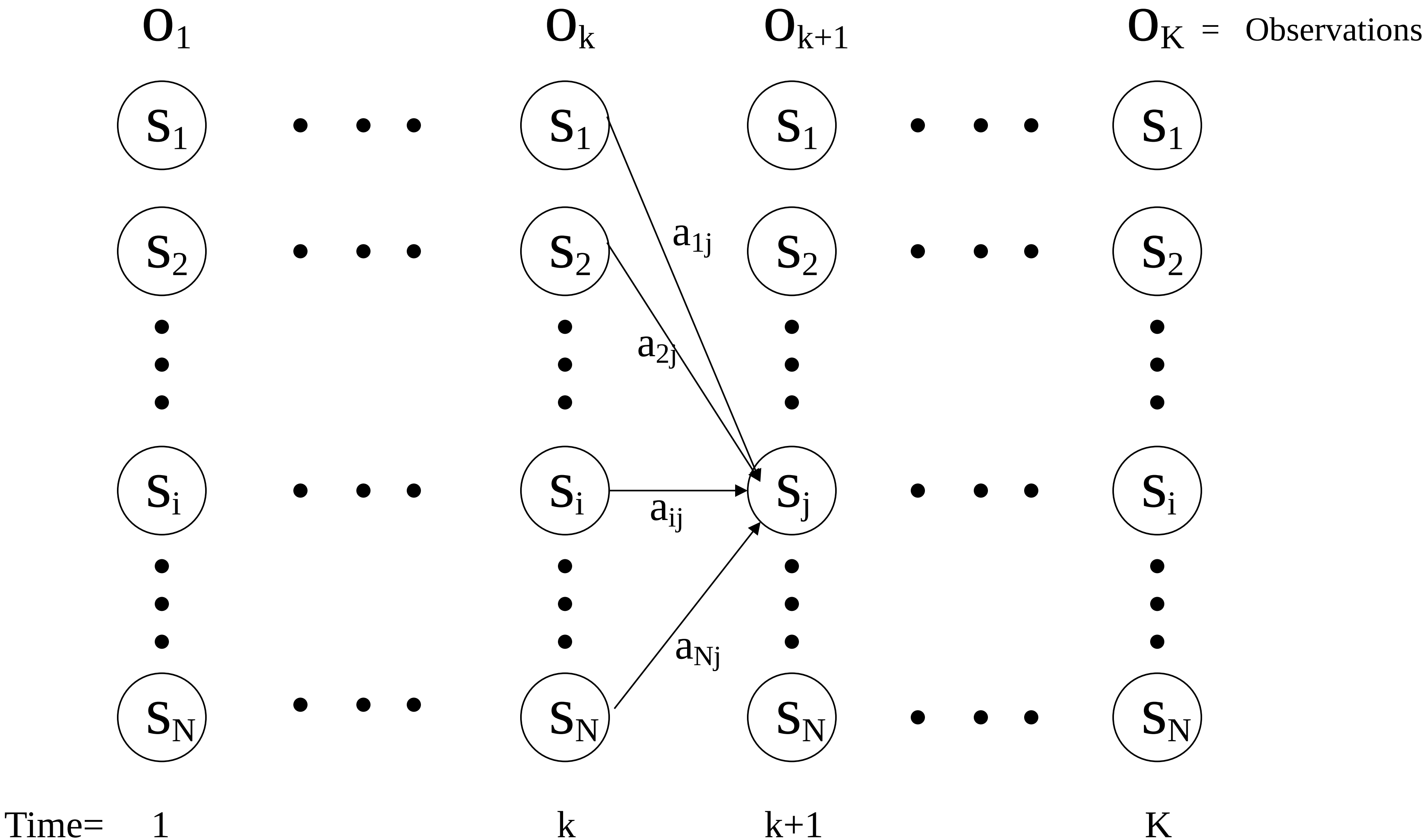
• Trying to find probability of observations $O=o_1 o_2 \ldots o_K$ by means of considering all hidden state sequences (as was done in example) is impractical:

N^K hidden state sequences - exponential complexity.

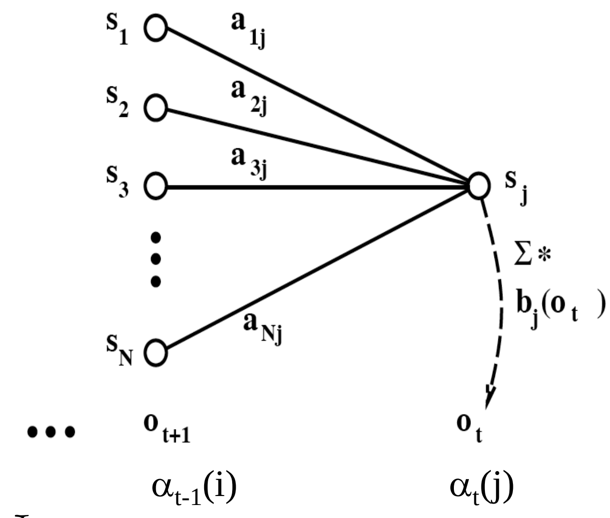
• Use **Forward-Backward HMM algorithms** for efficient calculations.

• Define the forward variable $\alpha_k(i)$ as the joint probability of the partial observation sequence $o_1 o_2 \ldots o_k$ and that the hidden state at time k is s_i : $\alpha_k(i)= P(o_1 o_2 \ldots o_k , q_k= s_i)$

Trellis representation of an HMM



Forward Recursion for HMM



- Initialization:

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

- Forward recursion:

  $\alpha_{k+1}(j) = P(o_1\, o_2 \ldots o_{k+1}, q_{k+1} = s_j) =$

  $\Sigma_i\, P(o_1\, o_2 \ldots o_{k+1}, q_k = s_i, q_{k+1} = s_j) =$

  $\Sigma_i\, P(o_1\, o_2 \ldots o_k, q_k = s_i)\, a_{ij}\, b_j(o_{k+1}) =$

  $[\Sigma_i\, \alpha_k(i)\, a_{ij}]\, b_j(o_{k+1})$, $1 <= j <= N, 1 <= k <= K-1$.

- Termination:

  $P(o_1\, o_2 \ldots o_K) = \Sigma_i\, P(o_1\, o_2 \ldots o_K, q_K = s_i) = \Sigma_i\, \alpha_K(i)$

- Complexity :

  N^2K operations.

Backward Recursion for HMM

- Define the forward variable $\beta_k(i)$ as the joint probability of the partial observation sequence $o_{k+1}\, o_{k+2} \ldots o_K$ given that the hidden state at time k is s_i : $\beta_k(i) = P(o_{k+1}\, o_{k+2} \ldots o_K \mid q_k = s_i)$
- <u>Initialization:</u>

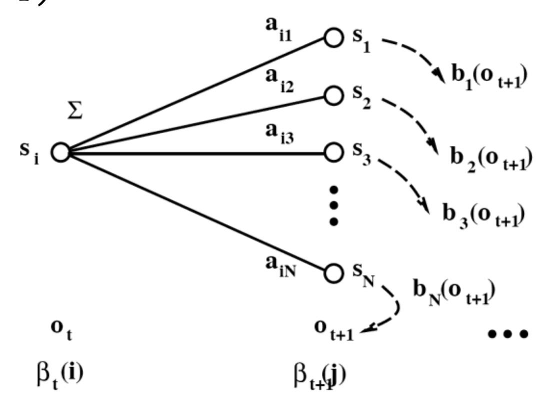
  $\beta_K(i) = 1 \quad , 1 <= i <= N.$
- <u>Backward recursion:</u>

  $\beta_k(j) = P(o_{k+1}\, o_{k+2} \ldots o_K \mid q_k = s_j) =$

  $\Sigma_i\, P(o_{k+1}\, o_{k+2} \ldots o_K, q_{k+1} = s_i \mid q_k = s_j) =$

  $\Sigma_i\, P(o_{k+2}\, o_{k+3} \ldots o_K \mid q_{k+1} = s_i)\, a_{ji}\, b_i(o_{k+1}) =$

  $\Sigma_i\, \beta_{k+1}(i)\, a_{ji}\, b_i(o_{k+1}) , \quad 1 <= j <= N, 1 <= k <= K-1.$



- <u>Termination:</u>

  $P(o_1\, o_2 \ldots o_K) = \Sigma_i\, P(o_1\, o_2 \ldots o_K, q_1 = s_i) =$

  $\Sigma_i\, P(o_1\, o_2 \ldots o_K \mid q_1 = s_i)\, P(q_1 = s_i) = \Sigma_i\, \beta_1(i)\, b_i(o_1)\, \pi_i$

Decoding problem

- **Decoding problem.** Given the HMM $M=(A, B, \pi)$ and the observation sequence $O=o_1\, o_2 \ldots o_K$, calculate the most likely sequence of hidden states s_i that produced this observation sequence.
- We want to find the state sequence $Q= q_1 \ldots q_K$ which maximizes $P(Q \mid o_1\, o_2 \ldots o_K)$, or equivalently $P(Q, o_1\, o_2 \ldots o_K)$.
- 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 $o_1\, o_2 \ldots o_k$ when moving along any hidden state sequence $q_1 \ldots q_{k-1}$ and getting into $q_k = s_i$.

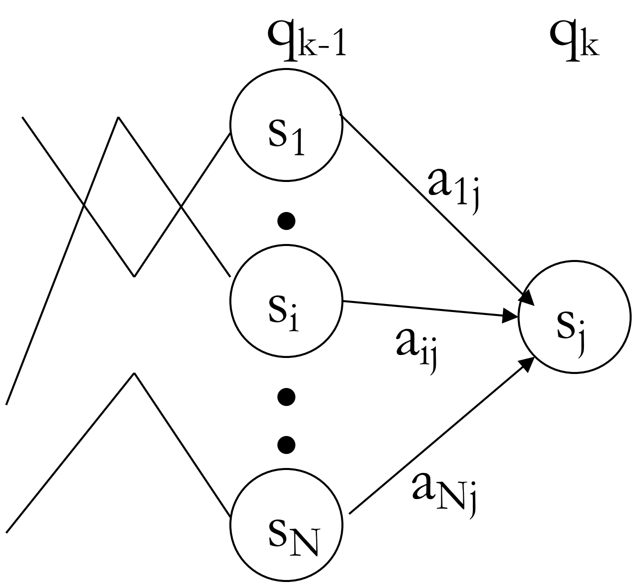
$$\delta_k(i) = \max P(q_1 \ldots q_{k-1}, q_k = s_i, o_1\, o_2 \ldots o_k)$$

where max is taken over all possible paths $q_1 \ldots q_{k-1}$.

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)

- <u>Initialization:</u>

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

- <u>Forward recursion:</u>

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

- <u>Termination:</u> 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 Σ 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, that is maximizes $P(O \mid M)$.

- There is no algorithm producing optimal parameter values.

- Use iterative expectation-maximization algorithm to find local maximum of $P(O \mid M)$ - **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:

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

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

Baum-Welch algorithm

General idea:

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

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

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

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 $o_1\, o_2 \dots o_K$.

$$\xi_k(i,j) = P(q_k = s_i\ , q_{k+1} = s_j\ |\ o_1\, o_2 \dots o_K)$$

$$\xi_k(i,j) = \frac{P(q_k = s_i\ , q_{k+1} = s_j\ , o_1\, o_2 \dots o_k)}{P(o_1\, o_2 \dots o_k)} =$$

$$\frac{P(q_k = s_i\ , o_1\, o_2 \dots o_k)\ a_{ij}\ b_j\,(o_{k+1})\ P(o_{k+2} \dots o_K\ |\ q_{k+1} = s_j)}{P(o_1\, o_2 \dots o_k)} =$$

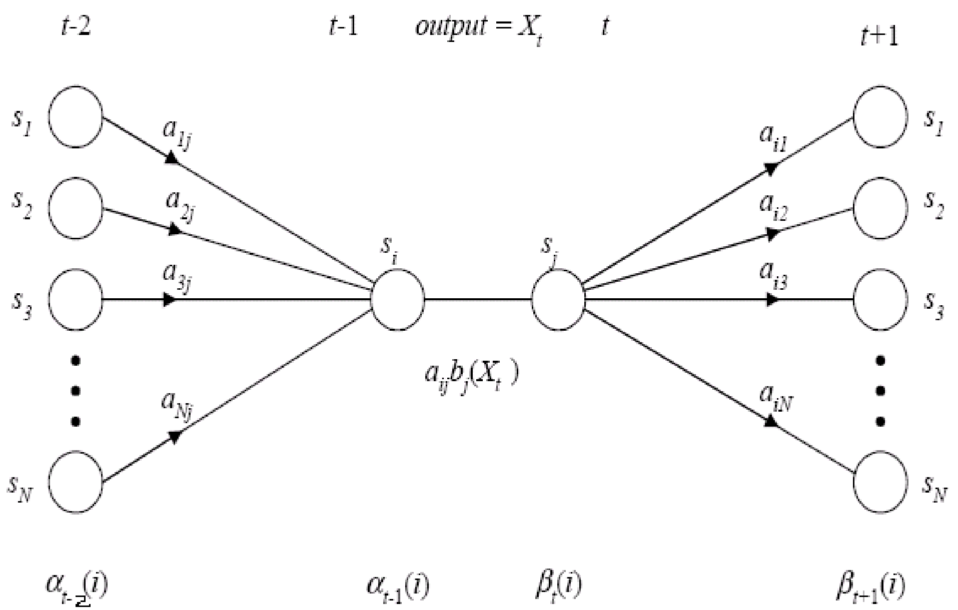
$$\frac{\alpha_k(i)\ a_{ij}\ b_j\,(o_{k+1})\ \beta_{k+1}(j)}{\Sigma_i\, \Sigma_j\ \alpha_k(i)\ a_{ij}\ b_j\,(o_{k+1})\ \beta_{k+1}(j)}$$

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 $o_1\, o_2 \ldots o_K$.

$$\gamma_k(i) = P(q_k = s_i \mid o_1\, o_2 \ldots o_K)$$

$$\gamma_k(i) = \frac{P(q_k = s_i, o_1\, o_2 \ldots o_k)}{P(o_1\, o_2 \ldots o_k)} = \frac{\alpha_k(i)\, \beta_k(i)}{\Sigma_i\, \alpha_k(i)\, \beta_k(i)}$$



Baum-Welch algorithm: expectation step(3)

- We calculated $\xi_k(i,j) = P(q_k = s_i, q_{k+1} = s_j \mid o_1 o_2 \ldots o_K)$
  and $\gamma_k(i) = P(q_k = s_i \mid o_1 o_2 \ldots o_K)$

- Expected number of transitions from state s_i to state s_j =
  $= \Sigma_k \xi_k(i,j)$
- Expected number of transitions out of state $s_i = \Sigma_k \gamma_k(i)$

- Expected number of times observation v_m occurs in state s_i =
  $= \Sigma_t \gamma_t(i)$, t is such that $o_t = v_m$
- Expected frequency in state s_i at time $k=1$: $\gamma_1(i)$.

Baum-Welch algorithm: maximization step

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

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

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