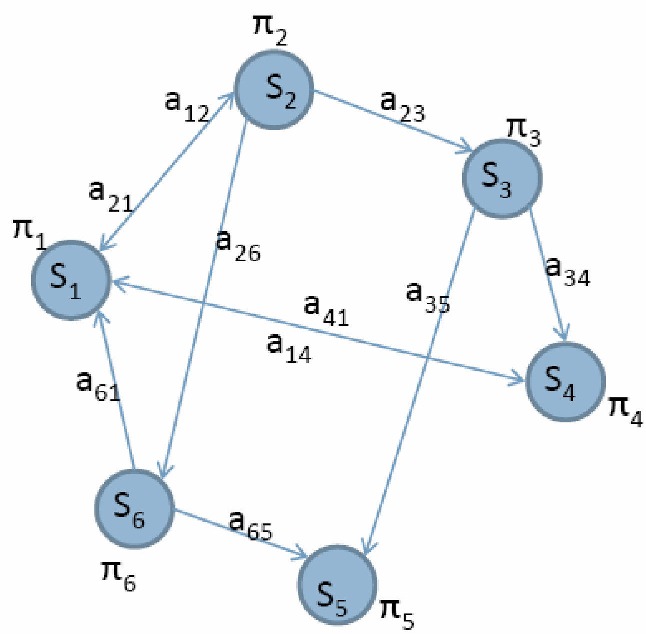
The Problem of Modeling Sequential Data

- Time series generated by a dynamic system
- A sequence generated by a spatial process

The Solutions

- Classic Approaches
  - Linear Models : Regression
  - NonLinear Models: Neural Networks, Decision Trees
- Problems with Classic Approaches
  - Data dependency is not incorporated in the prediction of the future
- State Space Model
  - A **state space** is a description of a configuration of discrete states used as a simple model of machines. Formally, it can be defined as a tuple [N, A, S, G] where:
    - N is a set of states
    - A is a set of arcs connecting the states
    - S is a nonempty subset of N that contains start states
    - G is a nonempty subset of N that contains the goal states.

Markov Process



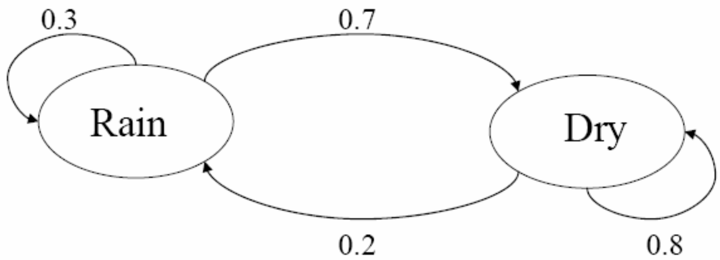
Markov Models

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

$$P(s_k \mid s_1, s_2, \ldots, s_{k-1}) = P(s_k \mid s_{k-1})$$

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

Example of Markov Model



- Two states : 'Rain' and 'Dry'.
- Transition probabilities: P('Rain'|'Rain')=0.3 ,

P('Dry'|'Rain')=0.7 , P('Rain'|'Dry')=0.2, P('Dry'|'Dry')=0.8

- Initial probabilities: say P('Rain')=0.4 , P('Dry')=0.6 .

Calculation of Sequence

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

$$P(s_1, s_2, \ldots, s_k) = P(s_k \mid s_1, s_2, \ldots, s_{k-1}) P(s_1, s_2, \ldots, s_{k-1})$$
$$= P(s_k \mid s_{k-1}) P(s_1, s_2, \ldots, s_{k-1}) = \ldots$$
$$= P(s_k \mid s_{k-1}) P(s_{k-1} \mid s_{k-2}) \ldots P(s_2 \mid s_1) P(s_1)$$

- 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: $\{s_1, s_2, \ldots, s_N\}$

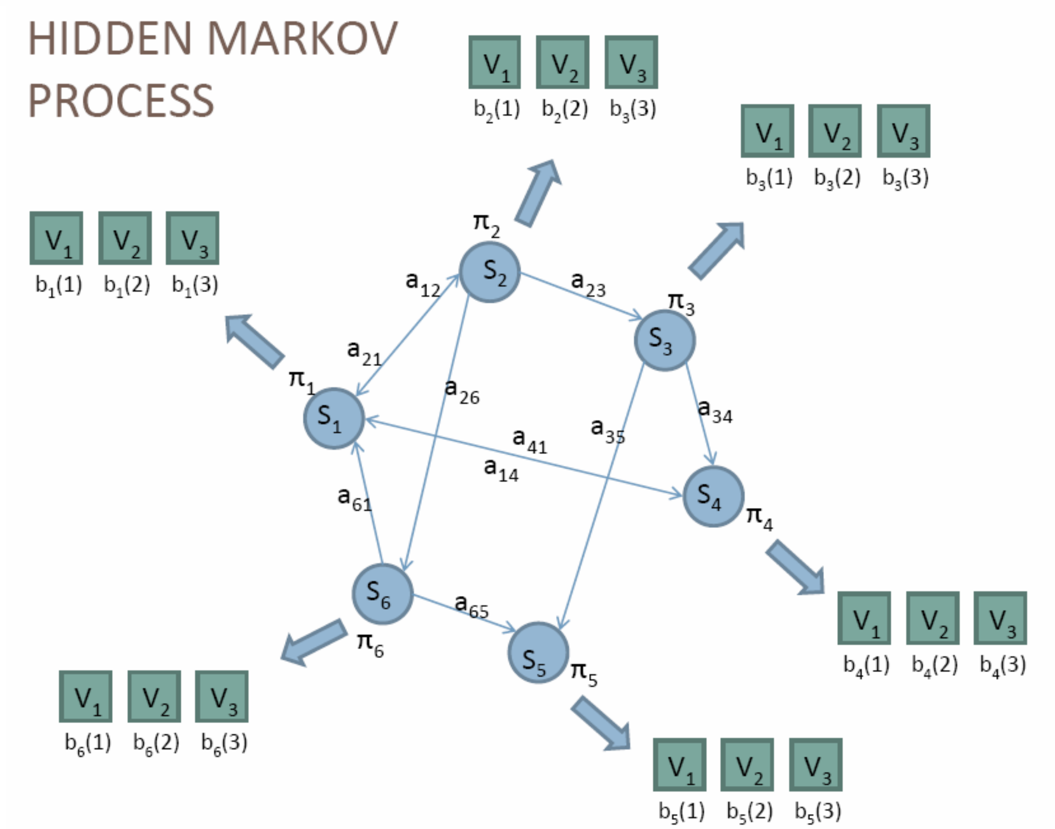
- Process moves from one state to another generating a sequence of states : $s_1, s_2, \ldots, s_k, \ldots$

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

$$P(s_k \mid s_1, s_2, \ldots, s_{k-1}) = P(s_k \mid s_{k-1})$$

- States are not visible, but each state randomly generates one of M observations (or visible states) $\{v_1, v_2, \ldots, v_M\}$

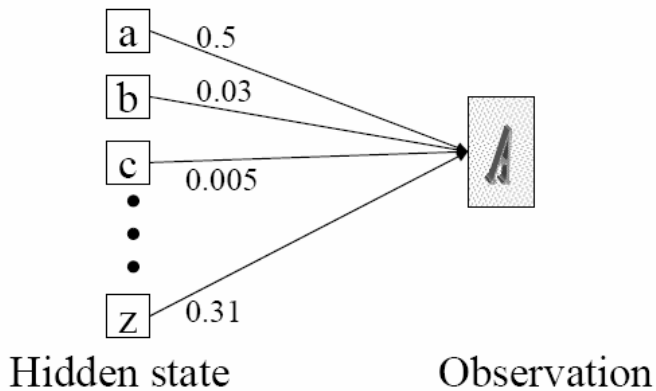
- To define hidden Markov model, the following probabilities have to be specified: matrix of transition probabilities $A=(a_{ij})$, $a_{ij}= P(s_i| s_j)$, matrix of observation probabilities $B=(b_i(v_m))$, $b_i(v_m)= P(v_m| s_i)$ and a vector of initial probabilities $\pi=(\pi_i)$, $\pi_i = P(s_i)$. Model is represented by $M=(A, B, \pi)$.



Word Recognition

• Typed word recognition, assume all characters are separated.

• Character recognizer outputs probability of the image being particular character, P(image|character).



• Hidden states of HMM = characters.

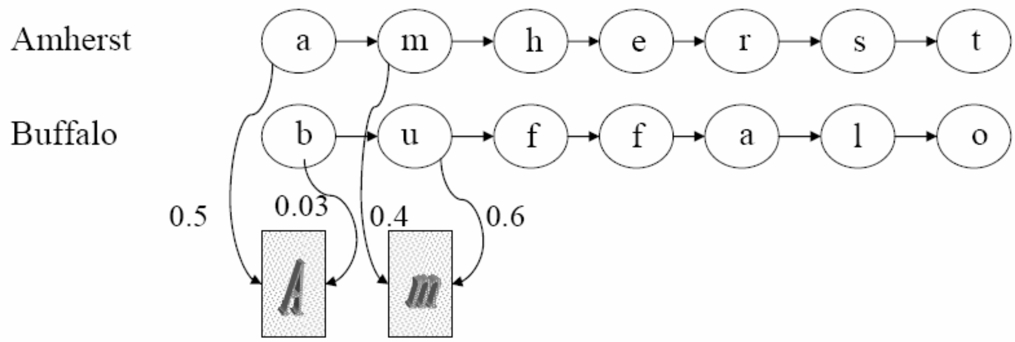
• Observations = typed images of characters segmented from the image v_α . Note that there is an infinite number of observations

• Observation probabilities = character recognizer scores.

$$B = \left(b_i(v_\alpha)\right) = \left(P(v_\alpha \mid s_i)\right)$$

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

• 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.

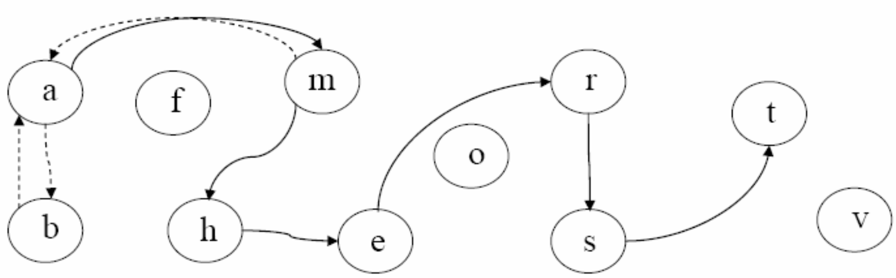
•This is an application of **Evaluation problem.**

• 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.

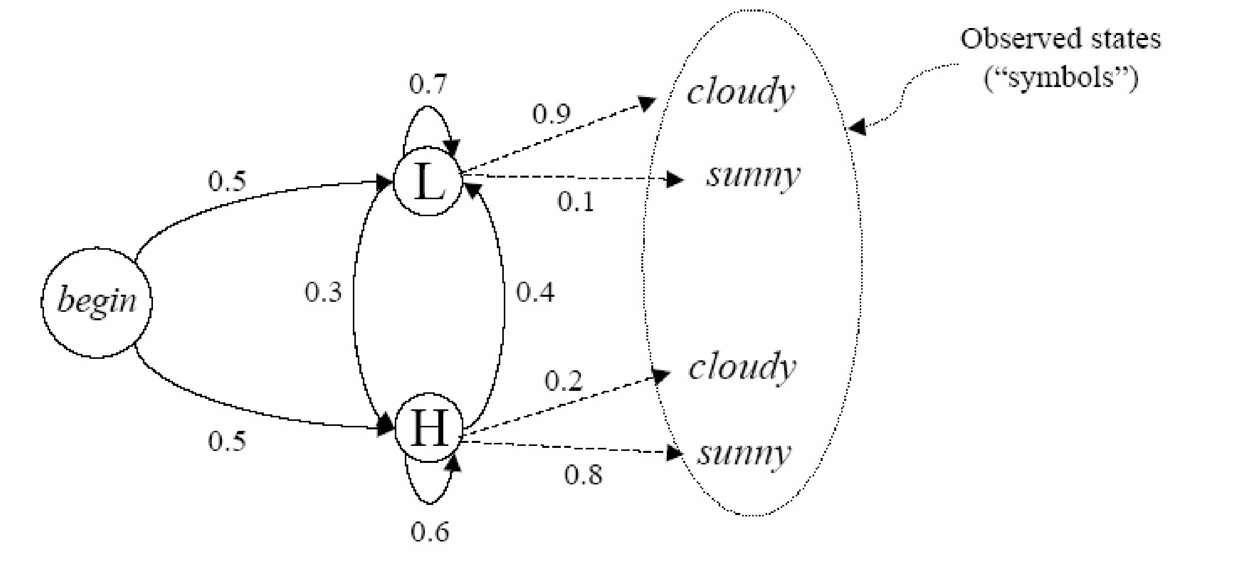
• Observations and observation probabilities are as before.



• 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.**

Example of HMM



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:

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

where first term is :

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

$$P(\{\text{'Dry','Rain'}\} \mid \{\text{'Low','Low'}\}) \; P(\{\text{'Low','Low'}\}) =$$

$$P(\text{'Dry'}|\text{'Low'})P(\text{'Rain'}|\text{'Low'}) \; P(\text{'Low'})P(\text{'Low'}|\text{'Low})$$

Main Issues with HMM

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* $o_k \in \{v_1, \dots, v_M\}$.

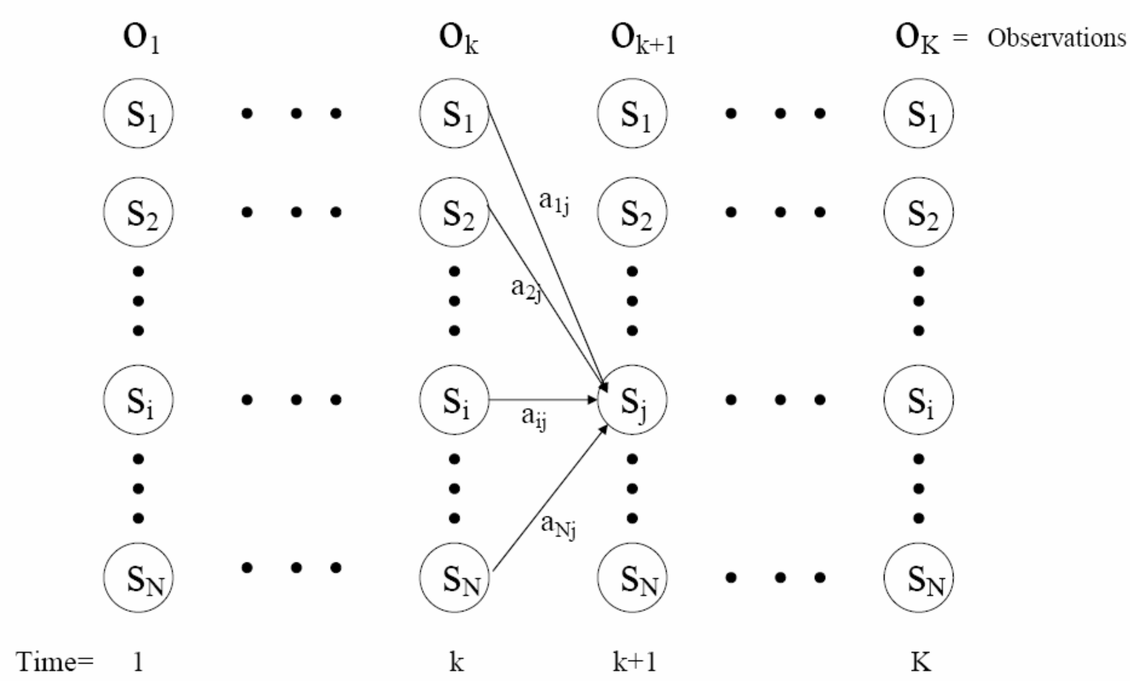
Evaluation Problem

- **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 : $\alpha_k(i)= P(o_1 o_2 \dots o_k, q_k=s_i)$



Forward Recursion for HMM

- Initialization:

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

- Forward recursion:

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

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

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

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

- 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.

Example

f		Sunny	Cloudy	Sunny
(begin)	1	0	0	0
Low	0	$(0.1)(0.5) = 0.05$	$0.9\begin{bmatrix}(0.05)(0.7)\\+\\(0.4)(0.4)\end{bmatrix} = 0.9(0.035 + 0.16) = 0.1755$	$0.1\begin{bmatrix}(0.1755)(0.7)\\+\\(0.051)(0.4)\end{bmatrix} = 0.1(0.12285 + 0.0204) = 0.014325$
High	0	$(0.8)(0.5) = 0.4$	$0.2\begin{bmatrix}(0.05)(0.3)\\+\\(0.4)(0.6)\end{bmatrix} = 0.2(0.015 + 0.24) = 0.051$	$0.8\begin{bmatrix}(0.1755)(0.3)\\+\\(0.051)(0.6)\end{bmatrix} = 0.8(0.05265 + 0.0306) = 0.0666$

$P(x)$=P(sunny, cloudy, sunny) = 0.014325(1) + 0.0666(1) = 0.080925

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

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

- Backward recursion:

$$\beta_k(j) = P(o_{k+1}\, o_{k+2} \ldots o_K \,|\, q_k = s_j) =$$
$$\Sigma_i\, P(o_{k+1}\, o_{k+2} \ldots o_{K\,,}\, q_{k+1} = s_i \,|\, q_k = s_j) =$$
$$\Sigma_i\, P(o_{k+2}\, o_{k+3} \ldots o_K \,|\, q_{k+1} = s_i)\; a_{ji}\; b_i(o_{k+1}) =$$
$$\Sigma_i\, \beta_{k+1}(i)\; a_{ji}\; b_i(o_{k+1}) \;, \qquad 1 <= j <= N,\ 1 <= k <= K-1.$$

- Termination:

$$P(o_1\, o_2 \ldots o_K) = \Sigma_i\, P(o_1\, o_2 \ldots o_{K\,,}\, q_1 = s_i) =$$
$$\Sigma_i\, P(o_1\, o_2 \ldots o_K \,|q_1 = s_i)\; P(q_1 = s_i) = \Sigma_i\, \beta_1(i)\; b_i(o_1)\; \pi_i$$

Example

b		Sunny	Cloudy	Sunny
(begin)				0
Low	0.5(0.1)(0.2265) =0.011325	0.7(0.9)(0.31) + 0.3(0.2)(0.52) = 0.2265	0.7(0.1)(1) + 0.3(0.8)(1) =0.31	1
High	0.5(0.8)(0.174) = 0.0696	0.4(0.9)(0.31) + 0.6(0.2)(0.52) = 0.174	0.4(0.1)(1) + 0.6(0.8)(1) =0.52	1

$\sum = 0.080925$ ✔

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 $q_k= s_i$.

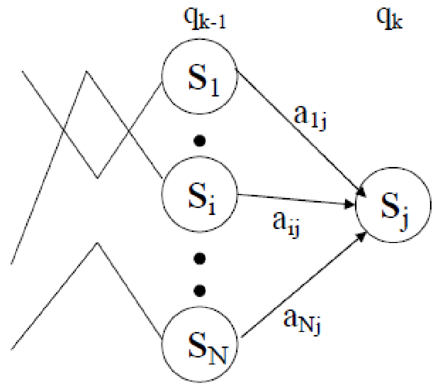
$$\delta_k(i) = \max P(q_1 \ldots\, q_{k-1}\, ,\, q_k= s_i\, ,\, o_1\, o_2 \ldots\, o_k)$$

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

Viterbi Algorithm

• 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 $q_{k-1} = s_i$.



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

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

• <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>

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

•<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.*

Example

v		Sunny	Cloudy	Sunny
(begin)	1	0	0	0
Low	0	$(0.1)(0.5) = 0.05$	$0.9\max\begin{cases}(0.05)(0.7)\\(0.4)(0.4)\end{cases} = 0.9\max\begin{cases}0.035\\0.16\end{cases} = (0.9)(0.16) = 0.144$	$0.1\max\begin{cases}(0.144)(0.7)\\(0.048)(0.4)\end{cases} = 0.1\max\begin{cases}0.1008\\0.0192\end{cases} = (0.1)(0.1008) = 0.01008$
High	0	$(0.8)(0.5) = 0.4$	$0.2\max\begin{cases}(0.05)(0.3)\\(0.4)(0.6)\end{cases} = 0.2\max\begin{cases}0.015\\0.24\end{cases} = (0.2)(0.24) = 0.048$	$0.8\max\begin{cases}(0.144)(0.3)\\(0.048)(0.6)\end{cases} = 0.8\max\begin{cases}0.0432\\0.0288\end{cases} = (0.8)(0.0432) = 0.03456$

Learning Problem

•**Learning problem.** Given some training observation sequences $O = o_1\, o_2 \ldots o_K$ 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.**

- Training HMM to encode obs seq such that HMM should identify a similar obs seq in future
- Find $\lambda=(A,B,\pi)$, maximising $P(O|\lambda)$
- General algorithm:
  - Initialise: λ_0
  - Compute new model λ, using λ_0 and observed sequence O
  - Then $\lambda_o \leftarrow \lambda$
  - Repeat steps 2 and 3 until:

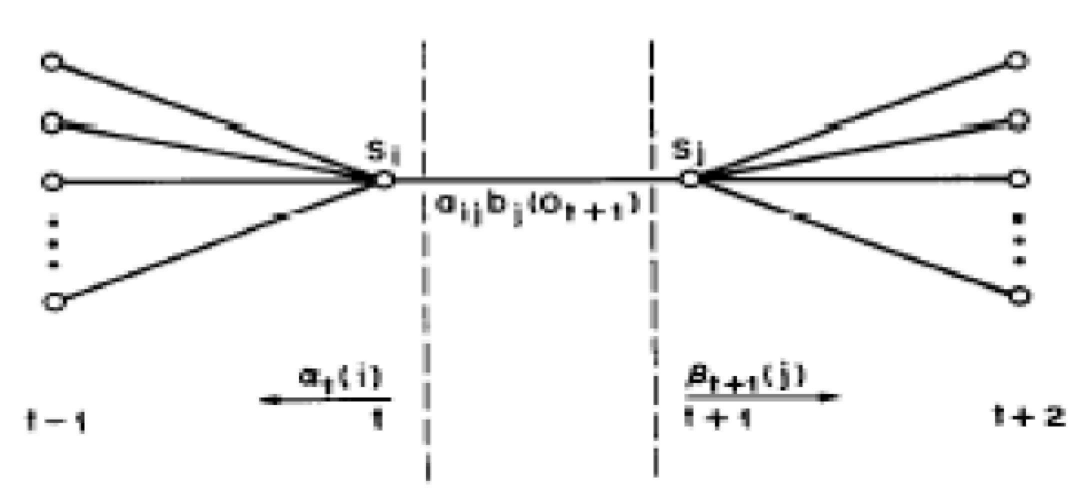
$$\log P(O \mid \lambda) - \log P(O \mid \lambda_0) < d$$

Step 1 of Baum-Welch algorithm:

- Let *ξ(i,j)* be a probability of being in state *i* at time *t* and at state *j* at time *t+1,* given λ and O seq

$$\xi(i,j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O \mid \lambda)}$$

$$= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}$$



Operations required for the computation of the joint event that the system is in state Si and time t and State Sj at time t+1

- Let $\gamma_t(i)$ be a probability of being in state *i* at time *t*, given *O*

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j)$$

- $\sum_{t=1}^{T-1} \gamma_t(i)$ - expected no. of transitions from state *i*
- $\sum_{t=1}^{T-1} \xi_t(i)$ - expected no. of transitions $i \to j$

Step 2 of Baum-Welch algorithm:

- $\hat{\pi} = \gamma_1(i)$ the expected frequency of state *i* at time *t=1*
- $\hat{a}_{ij} = \frac{\sum \xi_t(i, j)}{\sum \gamma_t(i)}$ ratio of expected no. of transitions from state *i* to *j* over expected no. of transitions from state *i*
- $\hat{b}_j(k) = \frac{\sum_{t, o_t = k} \gamma_t(j)}{\sum \gamma_t(j)}$ ratio of expected no. of times in state *j* observing symbol *k* over expected no. of times in state *j*