Ch9: CONTEXT DEPENDENT CLASSIFICATION

- Remember: Bayes rule

$$P(\omega_i | \underline{x}) > P(\omega_j | \underline{x}), \ \forall j \neq i$$

- Here: The class to which a feature vector belongs depends on:
  - Its own value
  - The values of the other features vectors
  - An existing relation among the various classes
- This interrelation demands the classification to be performed simultaneously for all available feature vectors
- Thus, we will assume that the training vectors $\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_N$ occur in sequence, one after the other and we will refer to them as **observations**

❖ The Context Dependent Bayesian Classifier

- Let $X : \{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_N\}$
- Let $\omega_i,\ i = 1,2,\ldots, M$
- Let Ω_i be a sequence of classes, that is

$$\Omega_i : \omega_{i1}\ \omega_{i2} \ldots \omega_{iN}$$

  There are M^N of those

- Thus, the Bayesian rule can equivalently be stated as

$$\boxed{X \to \Omega_i :\ P(\Omega_i | X) > P(\Omega_j | X)\ \forall i \neq j,\ \ i, j = 1,2,\ldots, M^N}$$

❖ Markov Chain Models (for class dependence)

$$P(\omega_{i_k} | \omega_{i_{k-1}}, \omega_{i_{k-2}}, \ldots, \omega_{i_1}) = P(\omega_{i_k} | \omega_{i_{k-1}})$$

- NOW remember:

$$P(\Omega_i) = P(\omega_{i_1}, \omega_{i_2}, \ldots, \omega_{i_N}) =$$

$$= P(\omega_{i_N} \left| \omega_{i_{N-1}}, \ldots, \omega_{i_1} \right.).P(\omega_{i_{N-1}} \left| \omega_{i_{N-2}}, \ldots, \omega_{i_1} \right.)\ldots P(\omega_{i_1})$$

or

$$\boxed{P(\Omega_i) = (\prod_{k=2}^{N} P(\omega_{i_k} \left| \omega_{i_{k-1}} \right.))P(\omega_{i_1})}$$

- Assume:
  - $\underline{x}_i$ statistically mutually independent
  - The pdf in one class independent of the others, then

$$\boxed{p\left(X \mid \Omega_i\right) = \prod_{k=1}^{N} p\left(\underline{x}_k \mid \omega_{i_k}\right)}$$

❖ From the above, the Bayes rule is readily seen to be equivalent to:

$$P(\Omega_i | X)(><)P(\Omega_j | X)$$

$$P(\Omega_i)\, p(X | \Omega_i)(><)P(\Omega_j)\, p(X | \Omega_j)$$

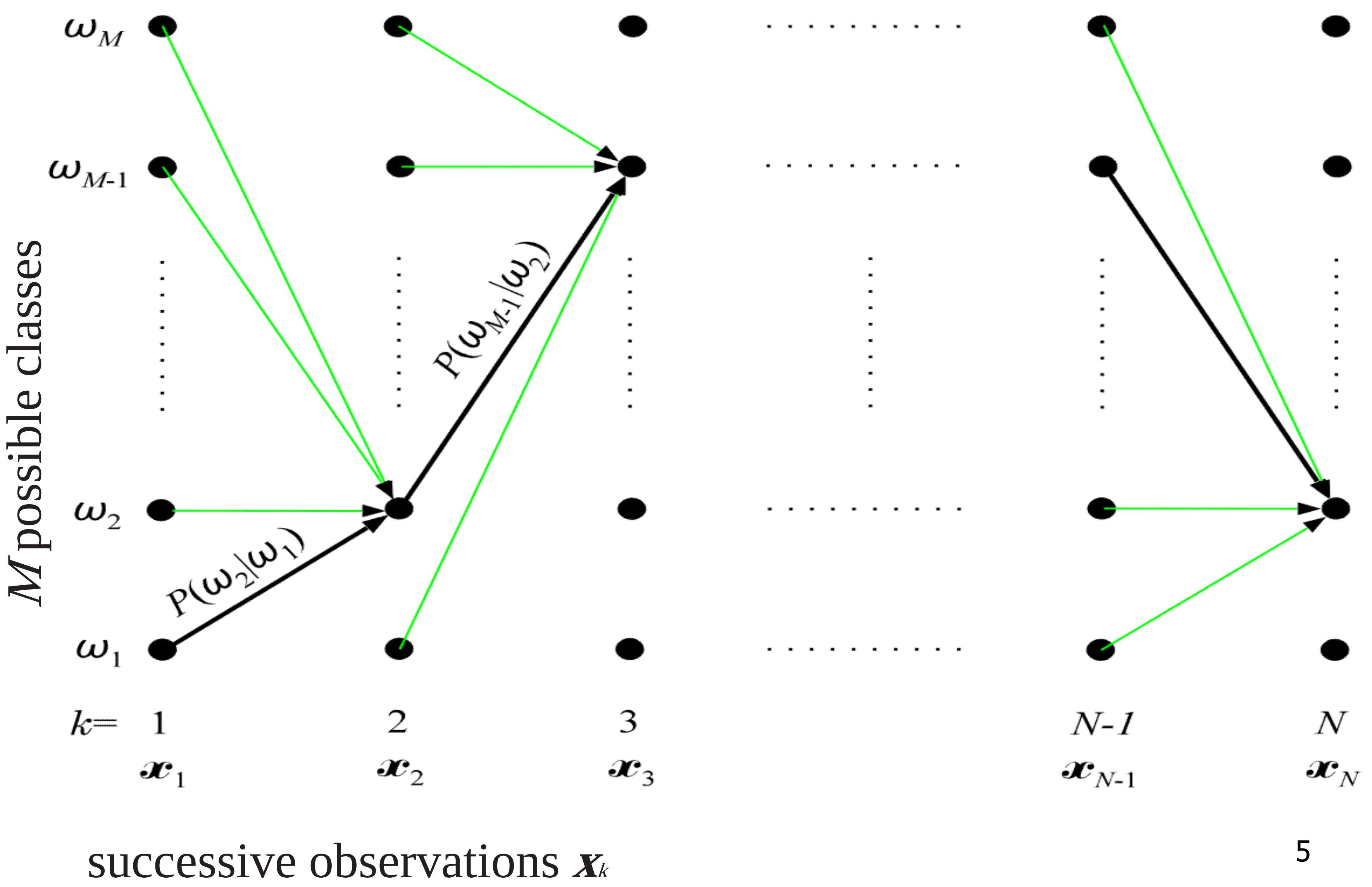
that is, it rests on

$$\boxed{p(X | \Omega_i)P(\Omega_i) = P(\omega_{i_1})\, p(\underline{x}_1 | \omega_{i_1}) \prod_{k=2}^{N} P(\omega_{i_k} | \omega_{i_{k-1}})\, p(\underline{x}_k | \omega_{i_k})}$$

❖ To find the above maximum in brute-force task we need $\mathrm{O}(NM^N)$ operations!!

❖ Given a sequence of observations $\{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_N\}$, find the path of successive (class) transitions that maximizes above equation.

The Viterbi Algorithm



- Each Ω_i corresponds to one path through the trellis diagram. One of them is the optimum (e.g., black). The classes along the optimal path determine the classes to which ω_i are assigned.

- To each transition corresponds a cost. For our case

  - $\hat{d}(\omega_{i_k}, \omega_{i_{k-1}}) = P(\omega_{i_k} | \omega_{i_{k-1}}).p(\underline{x}_k | \omega_{i_k})$

  - $\hat{d}(\omega_{i_1}, \omega_{i_0}) \equiv P(\omega_{i_1}) p(\underline{x}_1 | \omega_{i_1})$

  - $\hat{D} = \prod_{k=1}^{N} \hat{d}(\omega_{i_k}, \omega_{i_{k-1}}) = p(X | \Omega_i) P(\Omega_i)$

the overall cost to be optimized becomes

- Equivalently

$$\ln \hat{D} = \sum_{k=1}^{N} \ln \hat{d}(\omega_{i_k}, \omega_{i_{k-1}}) \equiv \sum_{k=1}^{N} d(\omega_{i_k}, \omega_{i_{k-1}}) \equiv D$$

where,

$$d(\omega_{i_k}, \omega_{i_{k-1}}) = \ln \hat{d}(\omega_{i_k}, \omega_{i_{k-1}})$$

- Define the cost for reaching class ω_{i_k} at stage k via a path i as

$$D(\omega_{i_k}) = \sum_{r=1}^{k} d(\omega_{i_r}, \omega_{i_{r-1}})$$

Bellman's Principle:

$$(i_0, j_0) \xrightarrow[(i,j)]{opt} (i_f, j_f) = (i_0, j_0) \xrightarrow{opt} (i,j) \oplus (i,j) \xrightarrow{opt} (i_f, j_f)$$

$\oplus$ denotes concatenation of paths

➢ Bellman's principle now states

$$\boxed{\begin{aligned} &D_{\max}(\omega_{i_k}) = \max_{i_{k-1}}\left[D_{\max}(\omega_{i_{k-1}}) + d(\omega_{i_k}, \omega_{i_{k-1}})\right] \\ &i_k, i_{k-1} = 1, 2, \ldots, M \end{aligned}}$$

with $D_{\max}(\omega_{i_0}) = 0$

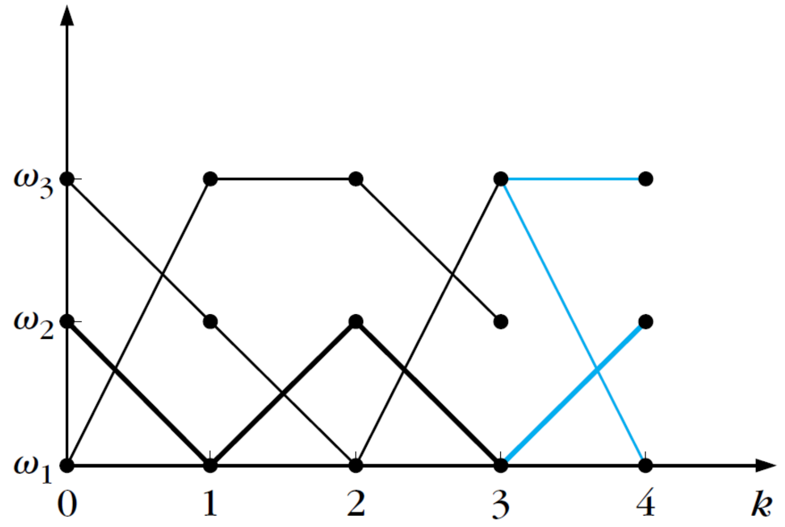
➢ The optimal path terminates at ω_{iN}^{*} :

$$\omega_{i_N}^{*} = \arg\max_{\omega_{i_N}} D_{\max}(\omega_{i_N})$$

- Complexity $O(NM^2)$

Example

❖ Apply the Viterbi algorithm to compute the optimal paths up to stage $k = 4$, Assume that $x_4 = 1.2$ and that the observations reside in the one-dimensional space. Let the task involve three classes, namely, ω_1 , ω_2, ω_3. We will further assume that the optimal paths up to stage $k=3$ have been computed and are shown in black lines in Figure. Let the optimal costs associated with each class at stage $k = 3$ be equal to $D(\omega_1) = -0.5$, $D(\omega_2) = -0.6$, $D(\omega_3) = -0.2$.



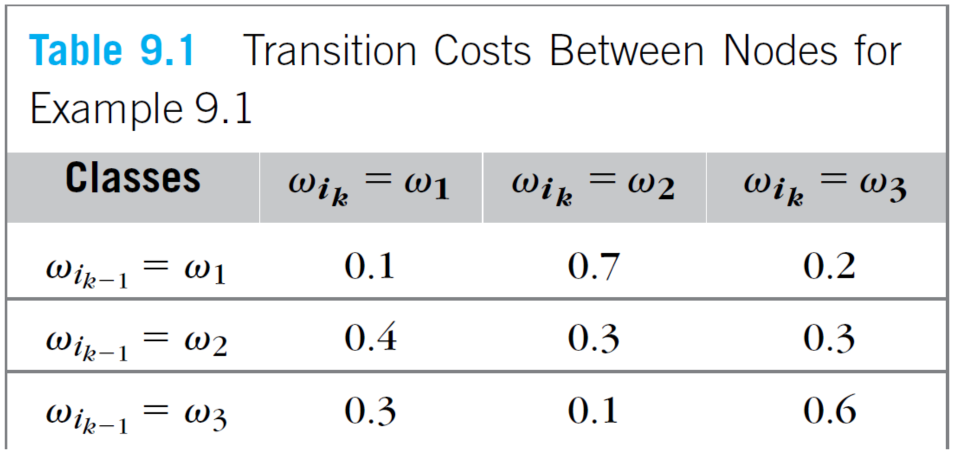


Table 9.1 Transition Costs Between Nodes for Example 9.1

Classes	$\omega_{i_k} = \omega_1$	$\omega_{i_k} = \omega_2$	$\omega_{i_k} = \omega_3$
$\omega_{i_{k-1}} = \omega_1$	0.1	0.7	0.2
$\omega_{i_{k-1}} = \omega_2$	0.4	0.3	0.3
$\omega_{i_{k-1}} = \omega_3$	0.3	0.1	0.6

$$p(x|\omega_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right)$$

$\mu_1 = 1.0$ and $\sigma_1^2 = 0.03$, $\mu_2 = 1.5$ and $\sigma_2^2 = 0.02$, $\mu_3 = 0.5$ and $\sigma_3^2 = 0.01$.

We will first compute the optimal path reaching class ω_1 at stage k= 4.

$$\ln p(x_4 = 1.2|\omega_{i_4} = \omega_1) = -0.1578$$

Total cost for the transition from $\omega_{i_3} = \omega_1$ to $\omega_{i_4} = \omega_1$ is equal to

$-0.5 + \ln(0.1) - 0.1578 = -2.9604$.

Total cost for the transition from $\omega_{i_3} = \omega_2$ to $\omega_{i_4} = \omega_1$ is equal to

$-0.6 + \ln(0.4) - 0.1578 = -1.6741$.

Total cost for the transition from $\omega_{i_3} = \omega_3$ to $\omega_{i_4} = \omega_1$ is equal to

$-0.2 + \ln(0.3) - 0.1578 = -1.5617$.

Hence, the optimal path reaching class ω_1 at stage k=4 is through ω_3 at stage k =3.

For the transitions to ω_2 at $k = 4$, we have

$$\ln p(x_4 = 1.2|\omega_{i_4} = \omega_2) = -0.2591$$

and the respective values for the paths reaching class ω_2 from classes ω_1, ω_2 and ω_3 at $k = 3$ are -1.1158, -2.0631, -2.7617. Thus the optimal path reaching ω_2 at $k = 4$ is through ω_1 at $k = 3$.
Finally, the respective values for the paths reaching ω_3 at $k = 4$ are

$$\ln p(x_4 = 1.2|\omega_{i_4} = \omega_3) = -2.2176$$

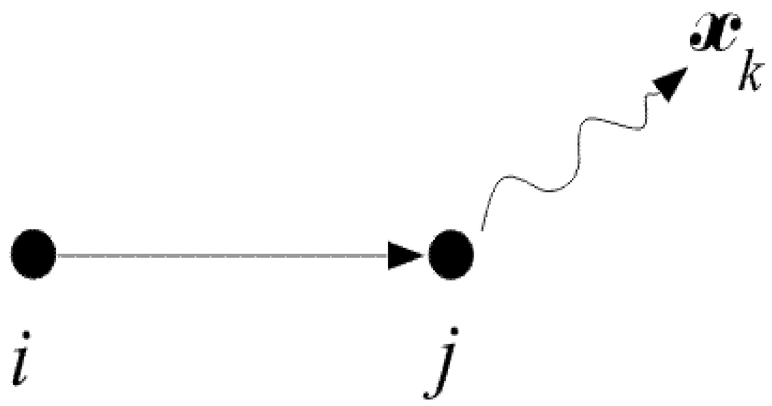
and -4.3271, -4.0216, -2.9285. As a result, the best path reaching node ω_3 at stage $k = 4$ goes through class ω_3 at stage $k = 3$ (self-transition).

If $k = 4$ is the final stage, that is, only four observations are available, then the optimal path, denoted by a bold line in Figure 9.2, is the one ending at stage ω_2 with an overall cost equal to -1.1158. Going backwards along the optimal path (backtracking), we assign: x_4 to ω_2, x_3 to ω_1, x_2 to ω_2, x_1 to ω_1 and x_0 to ω_2.

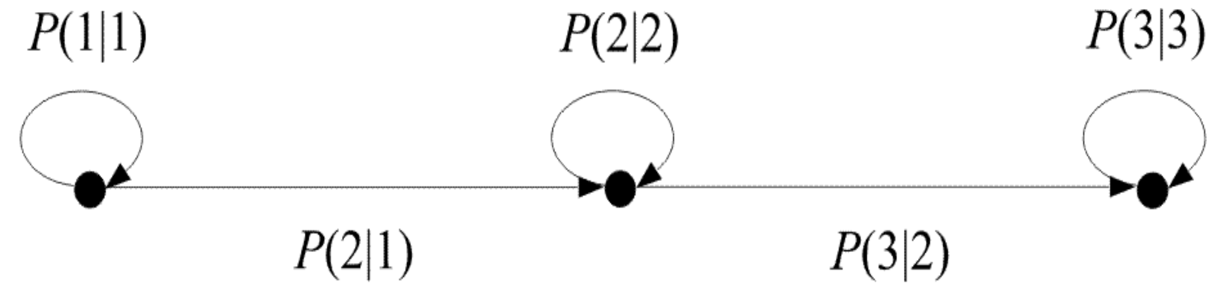
❖Hidden Markov Models (PR-Ch3-p6)(ML-chap15)

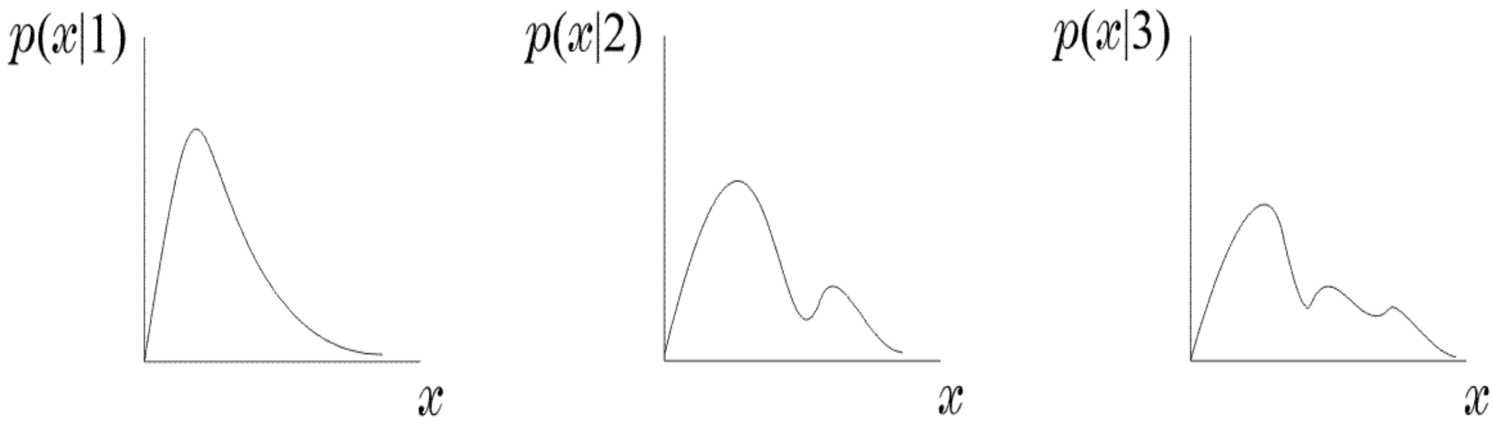
- In some problems like the channel equalization, the states are observable and can be "learned" during the training period
- Now we shall assume that states are not observable and can only be inferred from the training data
- Applications:
  - Speech and Music Recognition
  - OCR
  - Blind Equalization
  - Bioinformatics

- An HMM is a stochastic finite state automaton, that generates the observation sequence, $\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_N$
- We assume that: The observation sequence is produced as a result of **successive** transitions between states, upon arrival at a state:



- This type of modeling is used for nonstationary stochastic processes that undergo distinct transitions among a set of different stationary processes.



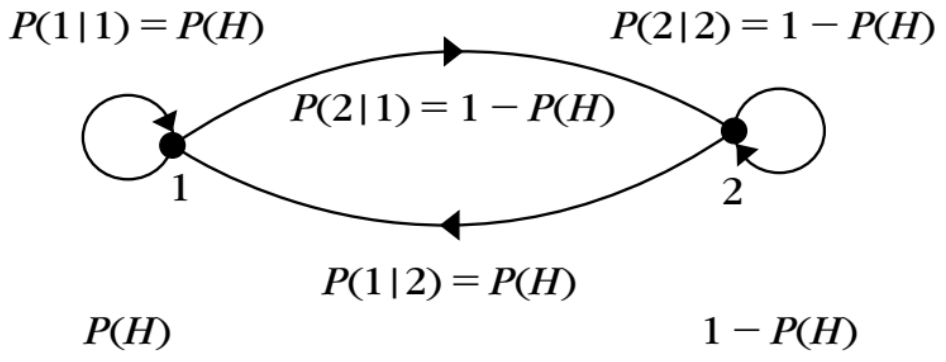


- Examples of HMM:
  - The single coin case: Assume a coin that is tossed behind a curtain. All it is available to us is the outcome, i.e., H or T. Assume the two states to be:

$$S = 1 \rightarrow H$$
$$S = 2 \rightarrow T$$

This is also an example of a random experiment with observable (not hidden) states. The model is characterized by a single parameter, e.g., $P(H)$. Note that

$$P(1|1) = P(H), \qquad P(2|1) = P(T) = 1 - P(H)$$

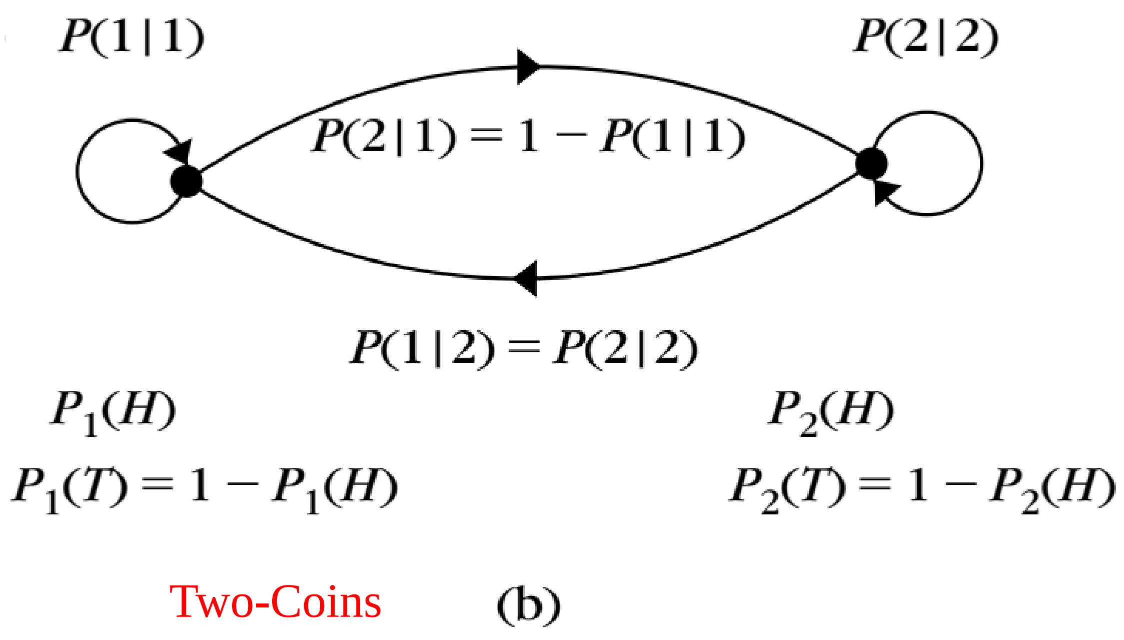


Single Coin

$P(i|j)$ denotes the transition probability from state s_j to state s_i once the coin has been tossed and an observation has been made available to us.

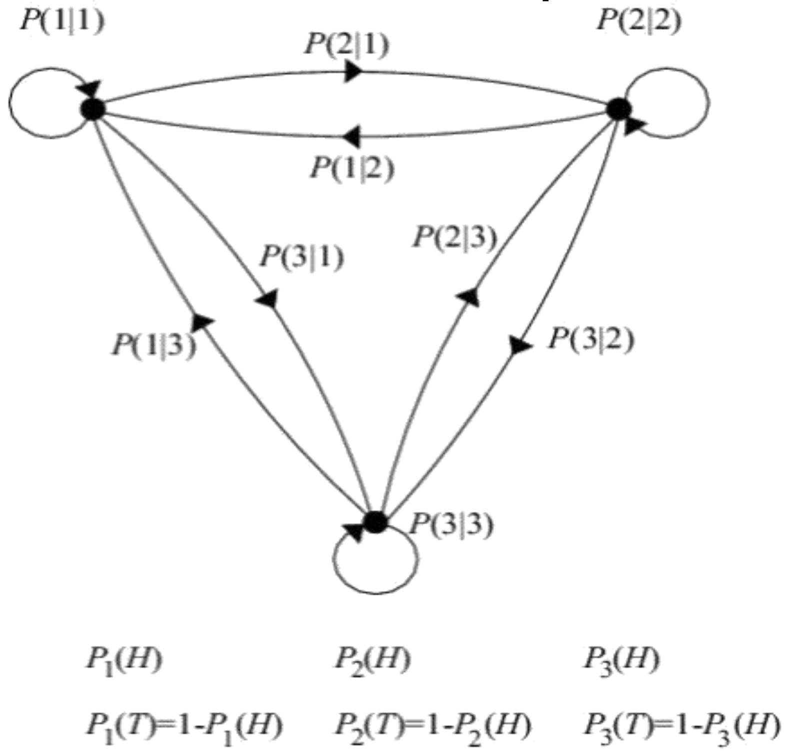
- The two-coins case: For this case, we observe a sequence of H or T. However, we have no access to know which coin was tossed. Identify one state for each coin. This is an example where states are not observable. H or T can be emitted from either state. The model depends on four parameters.

$$P_1(H),\ P_2(H),\quad P(1|1),\ P(2|2)$$



Note: $P(1|2)$ is the probability that the current observation (which can be either H or T) is the outcome of an experiment performed using coin 1 (state i=1) and that the previous observation was the result of tossing coin 2 (state j=2).

- The three-coins case example is shown below:



Nine parameters are now required

- Note that in all previous examples, specifying the model is equivalent to knowing:
  - The probability of each observation (H,T) to be emitted from each state.
  - The transition probabilities among states: $P(i|j)$.

- A general HMM model is characterized by the following set of parameters

1) K_s, number of states, s=1,2,…,M

2) $P(i \mid j),\ i, j = 1, 2, ..., K_s$

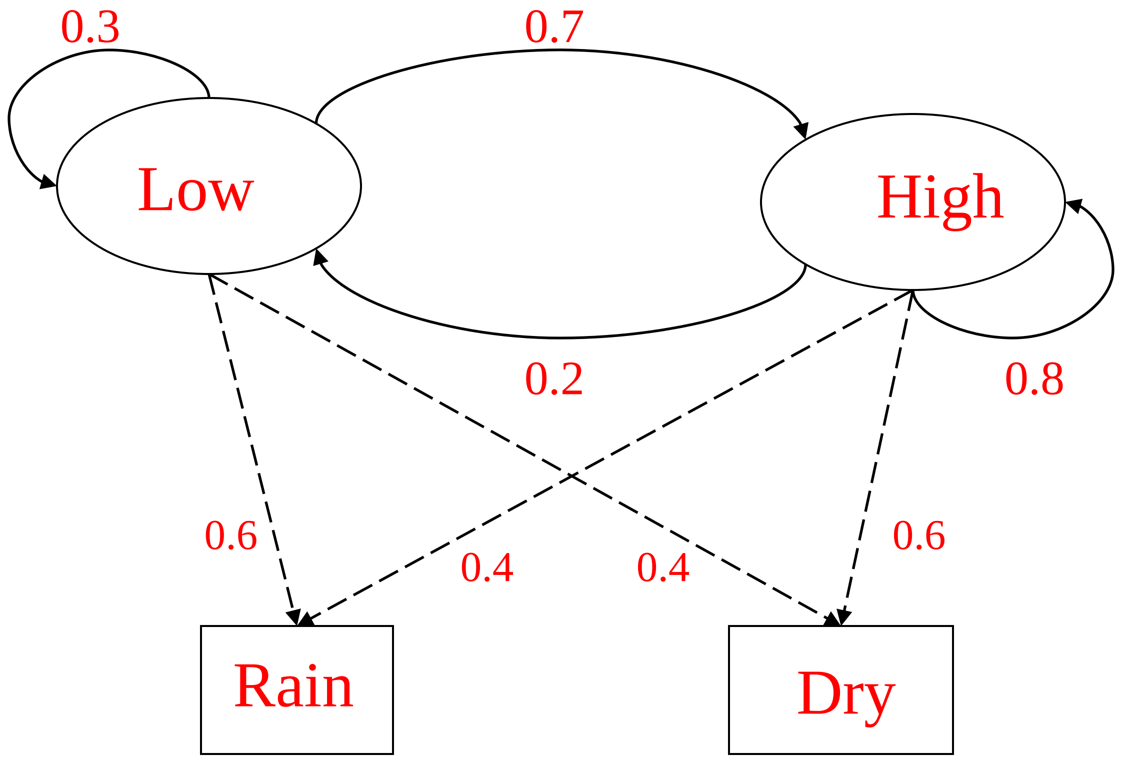
3) $p(\underline{x} \mid j),\ j = 1, 2, ..., K_s$

4) $P(i)$, i=1,2,…,K_S, initial state probabilities, $P(.)$

That is:

$$S = \left\{P(i \mid j), p(\underline{x} \mid i), P(i), K_S\right\}$$

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’)=0.6 .
- 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)

= 0.4×0.6×0.4×0.3=0.0288

- What is the problem in Pattern Recognition
  - Given M reference patterns, each described by an HMM, find the parameters, S, for each of them (training or learning)
  - Suppose we have an HMM as well as a set of observations X. Determine the most likely sequence of hidden states that led to those observations (decoding)
  - Given an unknown pattern, find to which one of the M, known patterns, matches best (recognition or evaluation)

- **Recognition:** Any path method
  - Assume the M models to be known (M classes).
  - A sequence of observations, X, is given.
  - Assume observations to be emissions upon the arrival on successive states
  - Decide in favor of the model S^* (from the M available) according to the Bayes rule

$$\boxed{S^* = \arg\max_S P(S|X)}$$

    - for equiprobable patterns

$$\boxed{S^* = \arg\max_S p(X|S)}$$

- For each model S there is more than one possible sets of successive state transitions Ω_i, each with probability $P(\Omega_i | S)$

Thus:

$$\boxed{\begin{aligned} P(X \mid S) &= \sum_i p(X, \Omega_i | S) \\ &= \sum_i p(X | \Omega_i, S) P(\Omega_i | S) \end{aligned}}$$

- For the efficient computation of the above DEFINE

  - forward variable $\alpha(i_k)$

$$\alpha(i_k) = p(\underline{x}_1, \ldots, \underline{x}_k, i_k \mid S)$$

$$= \sum_{i_k} \alpha(i_{k-1}) P(i_k \mid i_{k-1}) p(\underline{x}_k \mid i_k), \quad k = 2, \ldots, N$$

$$\alpha(i_1) = P(i_1) p(\underline{x}_1 \mid i_1)$$

- $\alpha(i_k)$ is the probability density of the joint event:

(a) a path is at state i_k ($i_k \in \{1, 2, \ldots, K_s\}$) at stage k and

(b) observations $\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_{k-1}$ have been emitted at the previous stages and

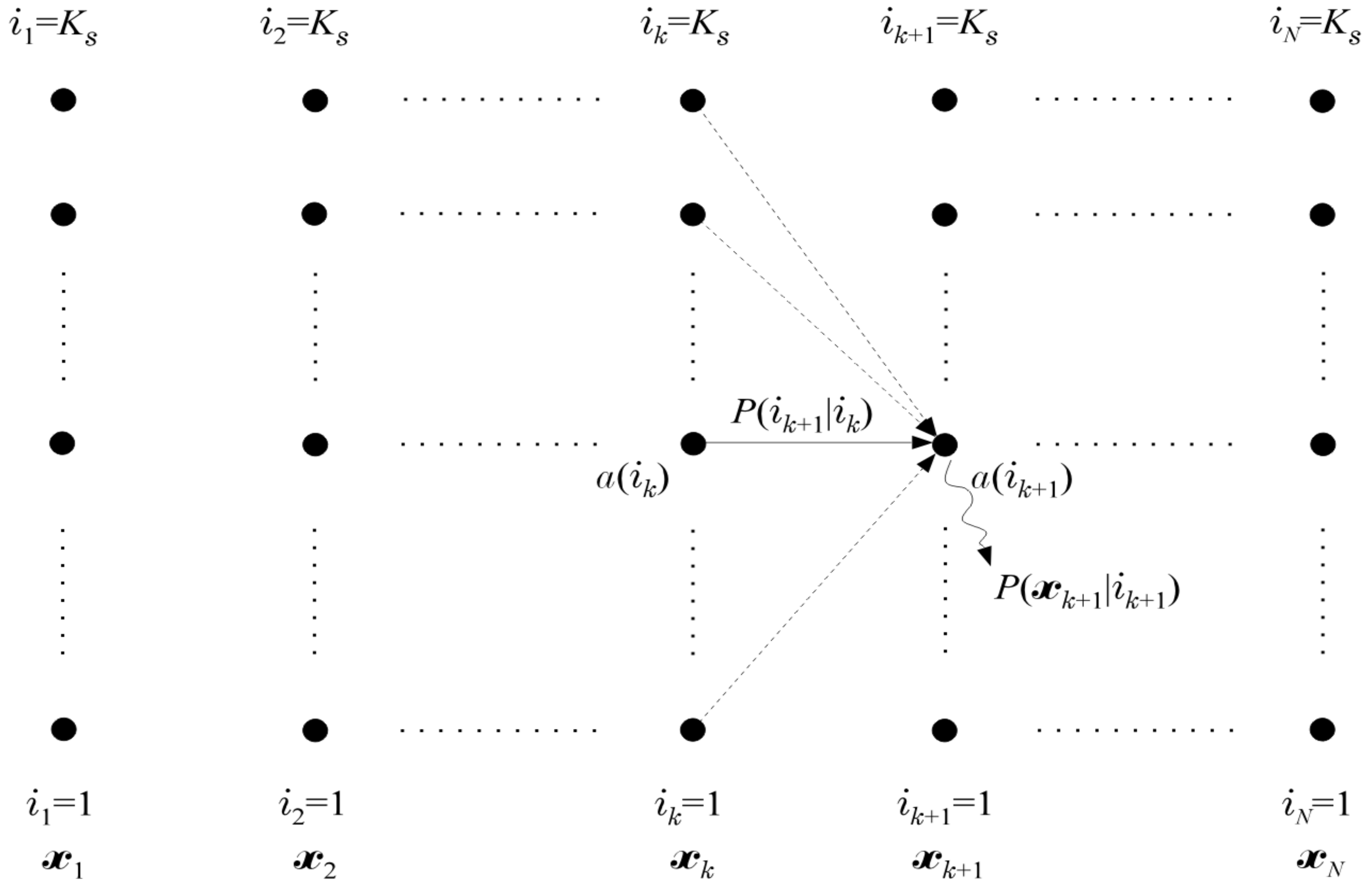
(c) observation x_k is emitted from the state i_k at stage k.

$$\alpha(i_{k+1}) = p(\underline{x}_1, \ldots, \underline{x}_{k+1}, i_{k+1} | S)$$

$$= \sum_{i_k} \underbrace{\alpha(i_k)}_{\text{History}} \underbrace{P(i_{k+1} | i_k)\, p(\underline{x}_{k+1} | i_{k+1})}_{\text{Local activity}} \qquad k = 1, 2, \ldots, N-1$$

$$\alpha(i_1) = P(i_1)\, p(\underline{x}_1 \mid i_1)$$

$$\alpha(i_1) = P(i_1)\,p(\underline{x}_1 \mid i_1)$$



$$\boxed{P(X|S) = \sum_{i_N=1}^{K_S} \alpha(i_N)}$$

Compute this for each S

- **Some more quantities**

- Backward variable $\beta(i_k)$: The probability density function of the event: observations $\mathbf{x}_{k+1},\ldots,\mathbf{x}_N$ occur at stages k+1, ..., N, given that at stage k the path is at state i_k.

$$\beta(i_k) = p(\underline{x}_{k+1}, \underline{x}_{k+2}, \ldots, \underline{x}_N \mid i_k, S)$$

$$= \sum_{i_{k+1}} \beta(i_{k+1}) P(i_{k+1} \mid i_k)\, p(\underline{x}_{k+1} \mid i_{k+1}), \qquad k = N-1, \ldots, 1$$

$$\beta(i_N) = 1, \quad i_N \in \{1, 2, \ldots, K_S\}$$

$\gamma(i_k)$: The probability density of the joint event: (a) a path is at state i_k at stage k and (b) $\mathbf{x}_1,\ldots,\mathbf{x}_N$ have been observed is:

$$\gamma(i_k) = p(\underline{x}_1, \ldots, \underline{x}_N, i_k \mid S)$$

$$= p(\underline{x}_1, \ldots, \underline{x}_k, i_k \mid S)\, p(\underline{x}_{k+1}, \ldots, \underline{x}_N \mid i_k, S) = \alpha(i_k)\beta(i_k)$$

➢Training

- The philosophy:

  Given a training set X, known to belong to the specific model, estimate the unknown parameters of S, so that the **output** of the model, e.g.

$$p(X|S) = \sum_{i_N=1}^{K_s} \alpha(i_N)$$

  to be maximized

➢ This is a ML estimation problem with missing data

➢ The number of computations is of the order of NK_s^2 (compare with NK_s^N).

Baum–Welch Reestimation

- ✓ Assumption: Data $\underline{x}$ discrete

$$\underline{x} \in \{1,2,\ldots,r\} \Rightarrow p(\underline{x}|i) \equiv P(\underline{x}|i)$$

- ✓ Definitions:

- $\xi_k(i, j, X \mid S)$ = the probability of the joint event:
- ➢ (a) a path passes through state *i* at stage *k* and
- ➢ (b) through state *j* at the next stage *k*+1 and
- ➢ (c) the model generates the available sequence of observations *X*, given the parameters of the model *S*.

- $\gamma_k(i \mid X, S)$ = the probability of the event: a path passes through state *i* at stage *k* given the model and the available observation sequence.

Can be shown

$$\xi_k(i,j) = \xi_k(i,j \mid X, S) = \frac{\xi_k(i,j,X \mid S)}{P(X \mid S)}$$

$$\xi_k(i,j) = \frac{\alpha(i_k = i) P(j|i) P(\underline{x}_{k+1}|j) \beta(i_{k+1} = j)}{P(X|S)}$$

$$\gamma_k(i) = \gamma_k(i|X,S) = \frac{\alpha(i_k = i)\beta(i_k = i)}{P(X|S)}$$

- $\sum_{k=1}^{N} \gamma_k(i)$ can be regarded as the expected (over the number of stages) number of times state i occurs, given the model S and the observation sequence X. When the upper index in the summation is $N-1$, this quantity is the expected number of transitions from state i.
- $\sum_{k=1}^{N-1} \xi_k(i,j)$ can be regarded as the expected number of transitions from state i to state j, given the model and the observation sequence.

- The Algorithm:
  - Initial conditions for all the unknown parameters.

    Compute $P(X \mid S)$

  - Step 1: From the current estimates of the model parameters reestimate the new model $\overline{S}$ from

$$\left\{\begin{array}{l}
- \quad \overline{P}(j|i) = \dfrac{\sum_{k=1}^{N-1} \xi_k(i,j)}{\sum_{k=1}^{N-1} \gamma_k(i)} \quad \left(= \dfrac{\#\text{ of transitions from } i \text{ to } j}{\#\text{ of transitions from } i} \right) \\[3em]
- \quad \overline{P}_{\underline{x}}(r|i) = \dfrac{\sum\limits_{(k=1 \text{ and } \underline{x} \to r)}^{N} \gamma_k(i)}{\sum\limits_{k=1}^{N} \gamma_k(i)} \quad \left(= \dfrac{\text{at state } i \text{ and } \underline{x} = r}{\neq \text{ of being at state } i} \right) \\[3em]
- \quad \overline{P}(i) = \gamma_1(i)
\end{array}\right.$$

- Step 2: Compute $P(X|\overline{S})$. If $P(X|\overline{S}) - P(X|S) > \varepsilon,\ S = \overline{S}$ go to step 1. Otherwise stop.

- Remarks:
  - Each iteration improves the model

    $$\overline{S} : P(X|\overline{S}) > P(X|S)$$

  - The algorithm converges to a maximum (local or global)
  - The algorithm is an implementation of the EM algorithm

Normalization is Important

- Normalization is required to avoid such recursive algorithms from accumulating large amounts of computational noise.

- We can apply a normalization factor at each step of the calculation:

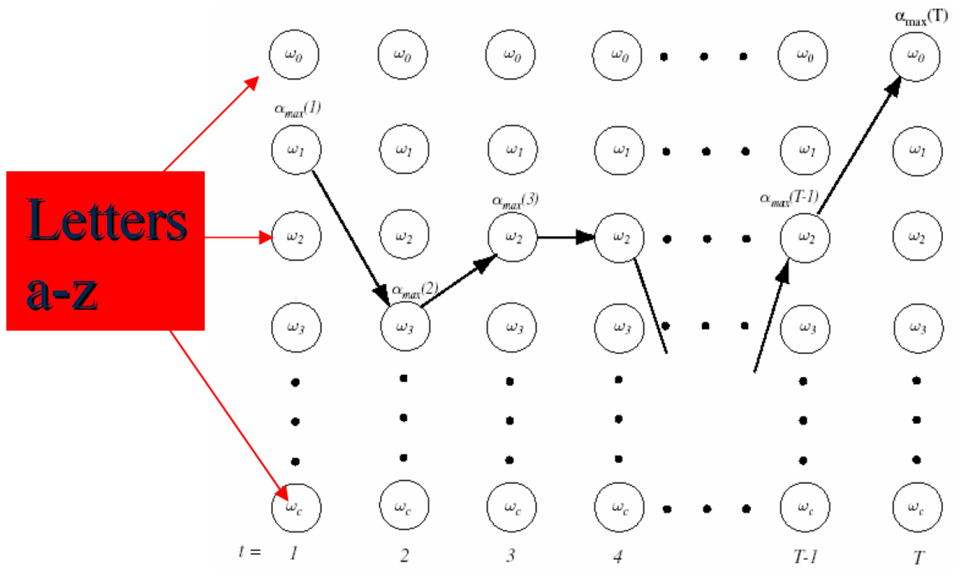
$$\alpha'_j(t) = \frac{\alpha_j(t)}{\prod_{i=0}^{t} Q_i}$$ where the scale factor, Q, is given by: $$Q_i = \sum_{i=1}^{N} \alpha'_i(t), \; Q_0 = 1$$

- This is applied once per state per unit time, and simply involves scaling the current α's by their sum at each epoch (e.g., a frame).

- Also, likelihoods tend to zero as time increases and can cause underflow. Therefore, it is more common to operate on log probabilities to maintain numerical precision . This converts products to sums but still involves essentially the same algorithm (though an approximation for the log of a sum is used to compute probabilities involving the summations). publications/courses/ece_8443/lectures/current/lecture_12.ppt

HMM Word Recognition

- HMM can model all possible words
- Each state corresponds to each letter of alphabet
- Letter transition probabilities are calculated for each pair of letters
- Letter confusion probabilities are symbol probabilities
- Separate HMMs are used to model each word



- Each word, e.g., cat, dog, etc, has an associated HMM
- For a test utterance determine which model has highest probability
- HMMs for speech are left-to-right models
- HMM produces a class conditional class-probability