CpE 646 Pattern Recognition and Classification

Prof. Hong Man

Department of Electrical and Computer Engineering
Stevens Institute of Technology

STEVENS Institute of Technology

Visual Information Environment Laboratory

Classification and Parameter Estimation

Chapter 2 (Section 2.10):

- Missing and Noisy Features

Chapter 3 (Section 3.9 – 3.10):

- Expectation-Maximization (EM)
- Hidden Markov Model

Classification with Missing Features

- We assume we have a Bayes classifier developed using perfect training data.
- However in certain particular test sample, some feature may be missing.
- Example: Four categories with 2-D feature space $[x_1, x_2]$. In one test sample, x_1 is missing, and x_2 is measured as $\hat{x}_2$
  - Option 1: we use the average of x_1 of four mean vectors as the missing feature for this test sample, so its feature vector becomes $[\bar{x}_1, \hat{x}_2]$, and the classification decision should be ω_3
  - Option 2: we find the maximum likelihood using x_2 only, i.e. $\max_i p(\hat{x}_2 \mid \omega_i)$ and the decision should be ω_2. This looks like a better decision if all priors are equal.

Classification with Missing Features

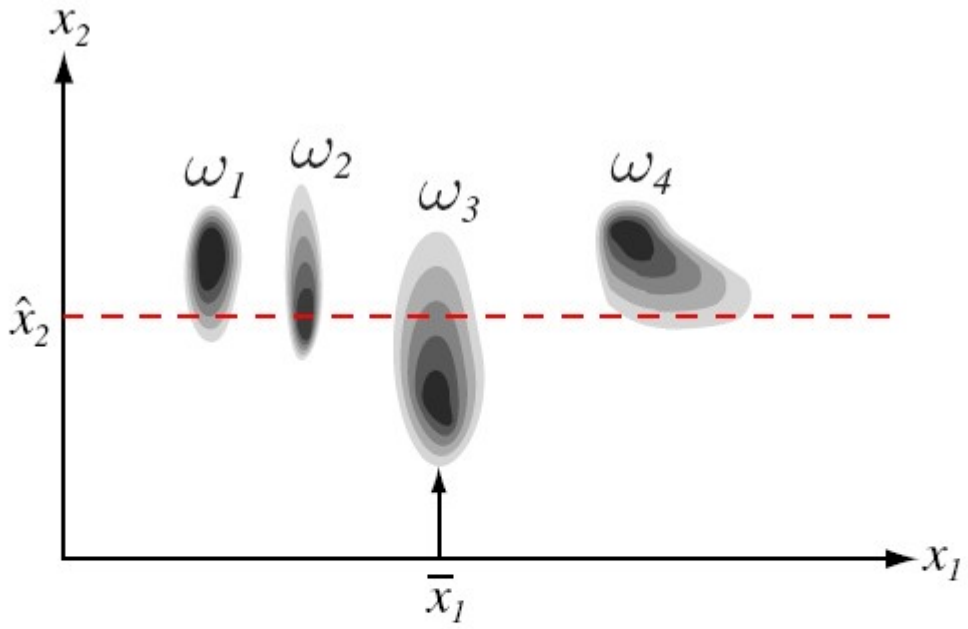


FIGURE 2.22. Four categories have equal priors and the class-conditional distributions shown. If a test point is presented in which one feature is missing (here, x_1) and the other is measured to have value $\hat{x}_2$ (red dashed line), we want our classifier to classify the pattern as category ω_2, because $p(\hat{x}_2|\omega_2)$ is the largest of the four likelihoods. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Classification with Missing Features

- In general, let $x=[x_g, x_b]$ where x_g represents good features, and x_b represents missing (or bad) features.
- We seek the Bayes rule based on only the good features

$$P(\omega_i \mid x_g) = \frac{p(\omega_i, x_g)}{p(x_g)} = \frac{\int p(\omega_i, x_g, x_b)dx_b}{p(x_g)}$$

$$= \frac{\int P(\omega_i \mid x_g, x_b)\, p(x_g, x_b)dx_b}{p(x_g)}$$

$$= \frac{\int g_i(x)\, p(x)dx_b}{\int p(x)dx_b}$$

where $g_i(x)=g_i(x_g,x_b)=P(\omega_i|x_g,x_b)$ is one of the minimum error rate discriminant function (with all features).

Classification with Missing Features

- This new posterior probability expression suggests that we should integrate the old posterior probabilities (with all feature variables) over the missing or bad features.
- The term $\int p(\omega_i, x_g, x_b) dx_b$ is called the marginal distribution, i.e. the full joint distribution is marginalized over the feature variable x_b.
- The Bayes decision is then based on

$$\max_i P(\omega_i \mid x_g)$$

Classification with Noisy Features

- When some particular feature has been corrupted by statistically independent noise, we may generalize the missing feature problem.
- Let the feature vector of a particular sample be $\boldsymbol{x}=[x_g, x_b]$, and also let x_t be the true value of the observed x_b.
- We assume we know the noise model for x_b, i.e. $p(x_b|x_t)$.
- We try to express $P(\omega_i|x_g,x_b)$ and $p(x_g, x_b)$ in terms of $g_i(x)=P(\omega_i|x_g,x_t)$, $p(x)=p(x_g, x_t)$ and $p(x_b|x_t)$.
- Consider that if x_t is known, x_b is independent to x_g and ω_i, so

$$P(\omega_i \mid x_g, x_b, x_t) = P(\omega_i \mid x_g, x_t)$$

$$p(x_g, x_b, x_t) = p(x_g, x_t)\,p(x_b \mid x_t)$$

Classification with Noisy Features

- We have

$$P(\omega_i \mid x_g, x_b) = \frac{\int p(\omega_i, x_g, x_b, x_t)dx_t}{p(x_g, x_b)}$$

$$= \frac{\int P(\omega_i \mid x_g, x_b, x_t)\, p(x_g, x_b, x_t)dx_t}{\int p(x_g, x_t)\, p(x_b \mid x_t)dx_t}$$

$$= \frac{\int P(\omega_i \mid x_g, x_t)\, p(x_g, x_t)\, p(x_b \mid x_t)dx_t}{\int p(x_g, x_t)\, p(x_b \mid x_t)dx_t}$$

$$= \frac{\int g_i(x)\, p(x)\, p(x_b \mid x_t)dx_t}{\int p(x) p(x_b \mid x_t)dx_t}$$

Estimation with Missing Features

- The expectation-maximization (EM) algorithm is able to learn the parameters of a distribution from training data sample, some of which may have missing features.
- The EM algorithm is to iteratively estimate the likelihood given the training data.
- Given training sample $\boldsymbol{D}=\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ from a single distribution, and any sample point may be written as $\boldsymbol{x}_k=[\boldsymbol{x}_{kg}, \boldsymbol{x}_{kb}]$, where $\boldsymbol{x}_{kg}$ are good features within the sample vector, and $\boldsymbol{x}_{kb}$ are bad (or missing) features within the sample vector. For convenience, we denote individual features into two sets, $\boldsymbol{D}_g$ and $\boldsymbol{D}_b$, with $\boldsymbol{D}=\boldsymbol{D}_g \cup \boldsymbol{D}_b$.
- $\boldsymbol{D}_g$ is called observed variables, and $\boldsymbol{D}_b$ is called latent variables

(http://www.answers.com/topic/expectation-maximization-algorithm)

Expectation-Maximization

- Define a function

$$Q(\theta;\theta^i) = \mathcal{E}_{D_b}[\ln p(D_g, D_b \mid \theta) \mid D_g, \theta^i]$$

$$= \int [\ln p(D_g, D_b \mid \theta)] p(D_b \mid D_g, \theta^i) dD_b$$

  - On the LHS, $Q(\theta; \theta^i)$ is a function of θ (one candidate parameter set for improved estimation) with fixed θ^i (the estimated θ at i-th iteration).
  - On the RHS, it calculates the expected value of the log-likelihood of the data over the missing feature D_b assuming θ^i are the true parameters. In discrete case, the integration becomes summation.
  - The algorithm will select

$$\theta^{i+1} = \theta = \arg\max_{\theta} Q(\theta;\theta^i)$$

Expectation-Maximization

- The EM algorithm

1. Begin initialize θ^0, ε, $i \leftarrow 0$
2. do $i \leftarrow i+1$
3. E step: evaluate $Q(\theta;\theta^i)$ for all θ
4. M step: $\theta^{i+1} \leftarrow \arg\max_{\theta} Q(\theta;\theta^i)$
5. until $Q(\theta^{i+1};\theta^i) - Q(\theta^i;\theta^{i-1}) \le \varepsilon$
6. Return $\hat{\theta} \leftarrow \theta^{i+1}$

Expectation-Maximization

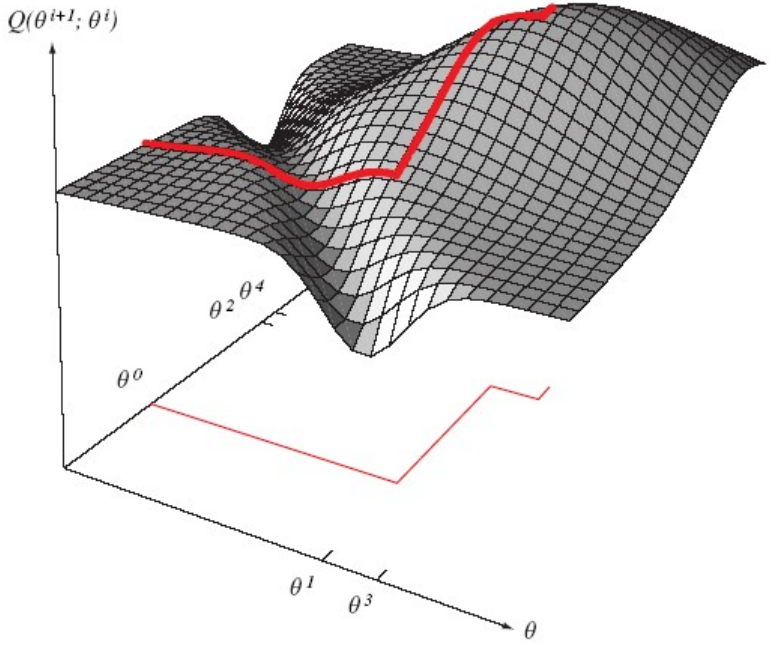


FIGURE 3.7. The search for the best model via the EM algorithm starts with some initial value of the model parameters, θ^0. Then, via the **M step**, the optimal θ^1 is found. Next, θ^1 is held constant and the value θ^2 is found that optimizes $Q(\cdot;\ \cdot)$. This process iterates until no value of θ can be found that will increase $Q(\cdot;\ \cdot)$. Note in particular that this is different from a gradient search. For example here θ^1 is the global optimum (given fixed θ^0), and would not necessarily have been found via gradient search. (In this illustration, $Q(\cdot;\ \cdot)$ is shown symmetric in its arguments; this need not be the case in general, however.) From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Example: EM for 2-D Normal Model

- Given four points in two dimension $\boldsymbol{D} = \{x_1, x_2, x_3, x_4\} = \{ [0,2]^t, [1,0]^t, [2,2]^t, [*,4]^t \}$, where * represents a missing feature x_{41} in point x_4. $D_b=[x_{41}]$ and D_g includes all other feature values.
- Assume a Gaussian with diagonal covariance and arbitrary mean, the parameters to be estimated are

$$\theta = [\mu_1, \mu_2, \sigma_1^2, \sigma_2^2]^t$$

- The initial guess of parameters are

$$\theta^0 = [0, 0, 1, 1]^t$$

Example: EM for 2-D Normal Model

- The E-step

$$Q(\theta;\theta^0) = \mathcal{E}_{x_{41}}[\ln p(D_g, D_b \mid \theta) \mid D_g, \theta^0]$$

$$= \int_{-\infty}^{+\infty} \left[\ln p(x_1 \mid \theta) + \ln p(x_2 \mid \theta) + \ln p(x_3 \mid \theta) + \ln p(x_4 \mid \theta)\right] p(x_{41} \mid x_{42}, \theta^0) dx_{41}$$

$$= \sum_{k=1}^{3} \ln p(x_k \mid \theta) + \int_{-\infty}^{+\infty} \ln p(x_4 \mid \theta) \frac{p(x_{41}, x_{42} \mid \theta^0)}{p(x_{42} \mid \theta^0) \equiv K} dx_{41}$$

$$= \sum_{k=1}^{3} \ln p(x_k \mid \theta) + \frac{1}{K} \int_{-\infty}^{+\infty} \ln p\left(\begin{pmatrix} x_{41} \\ 4 \end{pmatrix} \middle| \theta\right) \frac{1}{2\pi \left|\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right|} \exp\left[-\frac{1}{2}(x_{41}^2 + 4^2)\right] dx_{41}$$

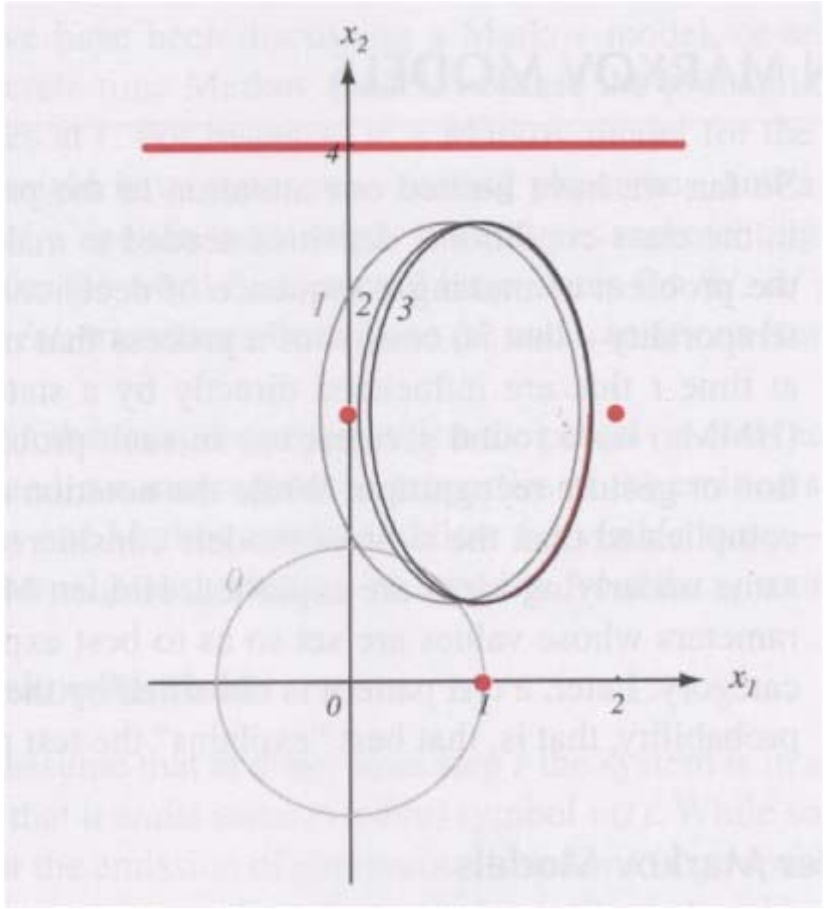
$$= \sum_{k=1}^{3} \ln p(x_k \mid \theta) - \frac{1+\mu_1^2}{2\sigma_1^2} - \frac{(4-\mu_2)^2}{2\sigma_2^2} - \ln(2\pi\sigma_1\sigma_2)$$

Example: EM for 2-D Normal Model

- The M-step:
  - Find the value $\theta=[\mu_1, \mu_2, \sigma_1^2, \sigma_2^2]^t$ that maximizes $Q(\theta;\theta^i)$, we have $\theta^1 =[0.75, 2.0, 0.938, 2.0]^t$.
- Repeat the E-step and M-step for several iterations we will have

$$\mu = \begin{pmatrix} 1.0 \\ 2.0 \end{pmatrix}, \text{ and } \Sigma = \begin{pmatrix} 0.667 & 0 \\ 0 & 2.0 \end{pmatrix}$$

Example: EM for 2-D Normal Model



Generalized EM Algorithm

- Generalized expectation-maximization (GEM) algorithms require merely that an improved θ^{i+1} rather than optimal θ^{i+1} is found in the M-step.
- GEM algorithms have slower convergence but are simpler in computation.

Markov Models

- We have discussed how to estimate the parameters in the class-conditional densities and then make a single decision (classification).
- In some temporal processes, we need to make a sequence of decisions
- Applications: speech recognition, gesture recognition, face recognition, and DNA sequencing etc.
- First order Markov models (or Markov Chains)
  - Consider a sequence of states at successive times, the state at time t is denoted as $\omega(t)$
  - Processes that unfold in time, states at time t are influenced by a state at time t-1

First Order Markov Models

- A temporal process without memory may produce a sequence of states $\omega^T = \{\omega(1), \omega(2), \omega(3), \ldots, \omega(T)\}$
- The system can revisit a state at different steps and not every state need to be visited. Example: $\omega^6 = \{\omega_1, \omega_4, \omega_2, \omega_2, \omega_1, \omega_4\}$
- The productions of any sequence is described by the transition probabilities $P(\omega_j(t+1) \mid \omega_i(t)) = a_{ij}$, meaning the time-independent probability of having state ω_j at time $t+1$ given that the state at t was ω_i
- The joint probability of a produced sequence in the example ω^6 is $P(\omega^T|\theta) = P(\omega_1)a_{14}a_{42}a_{22}a_{21}a_{14}$.

First Order Markov Models

- This model can be described as $\theta = (a_{ij})$ (may also include the priors if available)
- Example: speech recognition "production of spoken words"
  - Production of the word: "pattern" represented by phonemes /p/ /a/ /tt/ /er/ /n/ // (// = silent state)
  - Transitions from /p/ to /a/, /a/ to /tt/, /tt/ to er/, /er/ to /n/ and /n/ to a silent state

First Order Markov Models

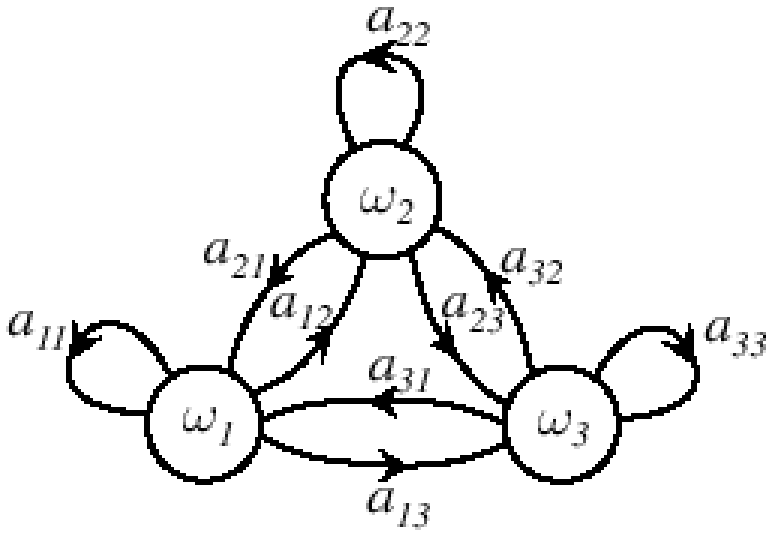


FIGURE 3.8. The discrete states, ω_i, in a basic Markov model are represented by nodes, and the transition probabilities, a_{ij}, are represented by links. In a first-order discrete-time Markov model, at any step t the full system is in a particular state $\omega(t)$. The state at step $t+1$ is a random function that depends solely on the state at step t and the transition probabilities. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Hidden Markov Models

- We now assume at time t the system is at state $\omega(t)$, but we can only observe a symbol $v(t)$ that is produced by the system. We consider the symbol is from a discrete set v_k for $k=1$ to K.
- The system has the property that at state $\omega_j(t)$, the probability of emitting a symbol $v_k(t)$ is $P(v_k(t)|\ \omega_j(t))=b_{jk}$.
- This give us a chance to estimate the system state based on the visible symbols.
- Because the actual states are hidden, this model is called hidden Markov model (HMM).

Hidden Markov Models

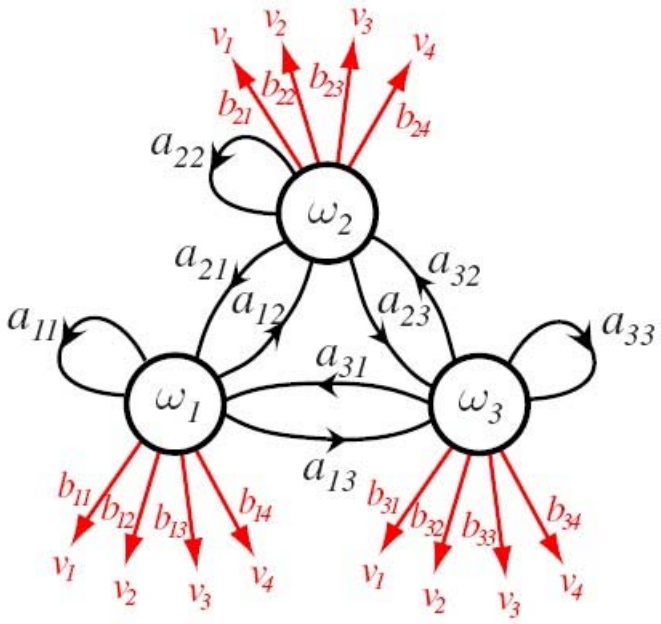


FIGURE 3.9. Three hidden units in an HMM and the transitions between them are shown in black while the visible states and the emission probabilities of visible states are shown in red. This model shows all transitions as being possible; in other HMMs, some such candidate transitions are not allowed. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Hidden Markov Models

- A Markov model is strictly causal, because the probabilities depend only upon previous state.
- A Markov model is called ergodic if every one of the states has a non-zero probability of occurring given some starting state.
- An absorbing state is the one that once entered, the system will never get out.
- HMM formulation
  - $a_{ij} = P(\omega_j(t+1) \mid \omega_i(t))$, $\Sigma_j a_{ij}=1$ for all i
  - $b_{jk} = P(v_k(t) \mid \omega_j(t))$, $\Sigma_k b_{jk}=1$ for all j

Hidden Markov Models

- HMM computation
  - The evaluation problem: given a_{ij} and b_{jk}, calculate the likelihood of an observed symbol sequence V^T
  - The decoding problem: given a_{ij} and b_{jk} and a set of observation sequences V^T, determine the most likely hidden state sequence ω^T
  - The learning problem: given a structure of an HMM (the number of hidden states, the visible symbol set), using a set of training observation sequence to determine a_{ij} and b_{jk}.

HMM Evaluation Problem

- For a particular sequence $\boldsymbol{V^T}$ with number of steps as $\boldsymbol{T}$, its probability is

$$P(V^T) = \sum_{r=1}^{r_{\max}} P(V^T \mid \omega_r^T) P(\omega_r^T)$$

  where each $\boldsymbol{\omega_r^T}=\{\boldsymbol{\omega(1)}, \boldsymbol{\omega(2)}, \boldsymbol{\omega(3)}, \ldots, \boldsymbol{\omega(T)}\}$ is a possible state sequence that produces this $\boldsymbol{V^T}$. If the total number of states is $\boldsymbol{c}$, then $\boldsymbol{r_{max}}=\boldsymbol{c^T}$ (i.e. c to the power of T).
- Consider only the first-order Markov process, we have

$$P(\omega_{\mathrm{r}}^{\mathrm{T}}) = \prod_{t=1}^{T} P(\omega(t) \mid \omega(t-1))$$

- The observed symbol only depends on the state, we have

$$P(V^T \mid \omega_r^T) = \prod_{t=1}^{T} P(v(t) \mid \omega(t))$$

HMM Evaluation Problem

- Therefore we have

$$P(V^T) = \sum_{r=1}^{r_{\max}} \prod_{t=1}^{T} P(v(t) \mid \omega(t))\, P(\omega(t) \mid \omega(t-1))$$

- This is an $O(c^T T)$ calculation.

HMM Evaluation Problem

- We can calculate this recursively. Define $\alpha_j(t)$ be the probability that the HMM is in hidden state ω_j at step t AND having generated the observation $v(t)$ from step 0 to t

$$\alpha_j(t) = \begin{cases} 0 & t = 0 \text{ and } j \neq \text{initial state} \\ 1 & t=0 \text{ and } j=\text{initial state} \\ \left[\sum_i \alpha_i(t-1)a_{ij}\right] b_{jk}(v(t)=v_k) & \text{otherwise} \end{cases}$$

- Here the notation $b_{jk}(v(t)=v_k)$ just means b_{jk} where k is selected according to the observation at step t, i.e. if $v(t)=v_k$

HMM Evaluation Problem

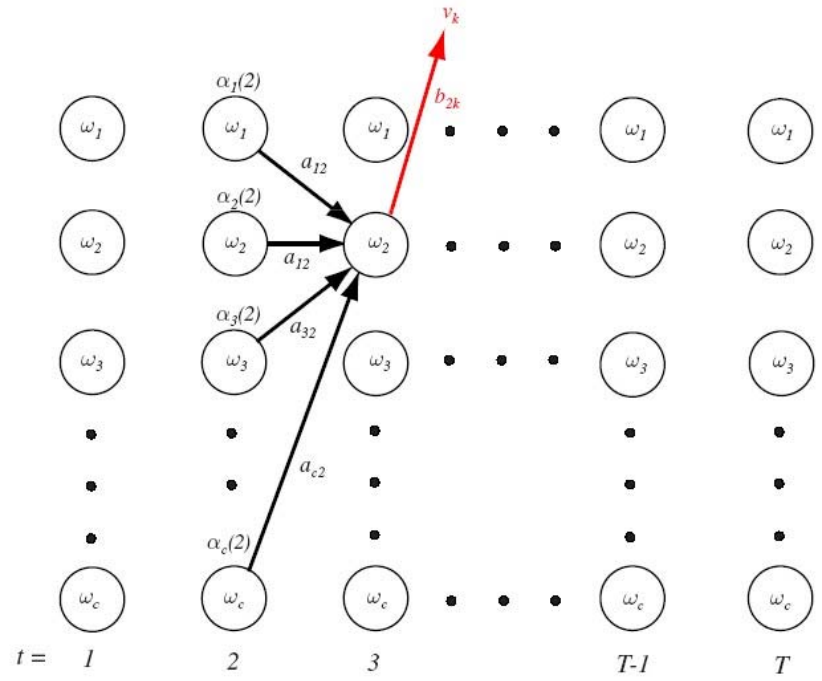


FIGURE 3.10. The computation of probabilities by the Forward algorithm can be visualized by means of a trellis—a sort of "unfolding" of the HMM through time. Suppose we seek the probability that the HMM was in state ω_2 at $t = 3$ and generated the observed visible symbol up through that step (including the observed visible symbol v_k). The probability the HMM was in state $\omega_j(t = 2)$ and generated the observed sequence through $t = 2$ is $\alpha_j(2)$ for $j = 1, 2, \ldots, c$. To find $\alpha_2(3)$ we must sum these and multiply the probability that state ω_2 emitted the observed symbol v_k. Formally, for this particular illustration we have $\alpha_2(3) = b_{2k} \sum_{j=1}^{c} \alpha_j(2) a_{j2}$. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

HMM Evaluation Problem

- The HMM forward algorithm

1. initialize $\boldsymbol{t \leftarrow 0}$, $\boldsymbol{a_{ij}}$, $\boldsymbol{b_{jk}}$, $\boldsymbol{V^T}$, $\boldsymbol{\alpha_j(0)}$
2. for $\boldsymbol{t \leftarrow t+1}$
3. $$\alpha_j(t) = \left[\sum_{i=1}^{c} \alpha_i(t-1) a_{ij}\right] b_{jk}(v(t) = v_k)$$
4. until $t = \boldsymbol{T}$
5. return $\boldsymbol{P(V^T) \leftarrow \alpha_0(T)}$, where $\boldsymbol{\alpha_0(T)}$ is the probability of the sequence $\boldsymbol{V^T}$ ending to the known final state $\boldsymbol{\omega_0}$ (e.g. the quite state in speech.) The calculation is $O(c^2T)$

HMM Evaluation Example

- Consider a 4-state ergodic HMM with 3 regular states (ω_1, ω_2, ω_3) emitting 4 visible symbols (v_1, v_2, v_3, v_4), and a known absorber state ω_0 emitting a unique null symbol v_0, and the following transition and emission probabilities:

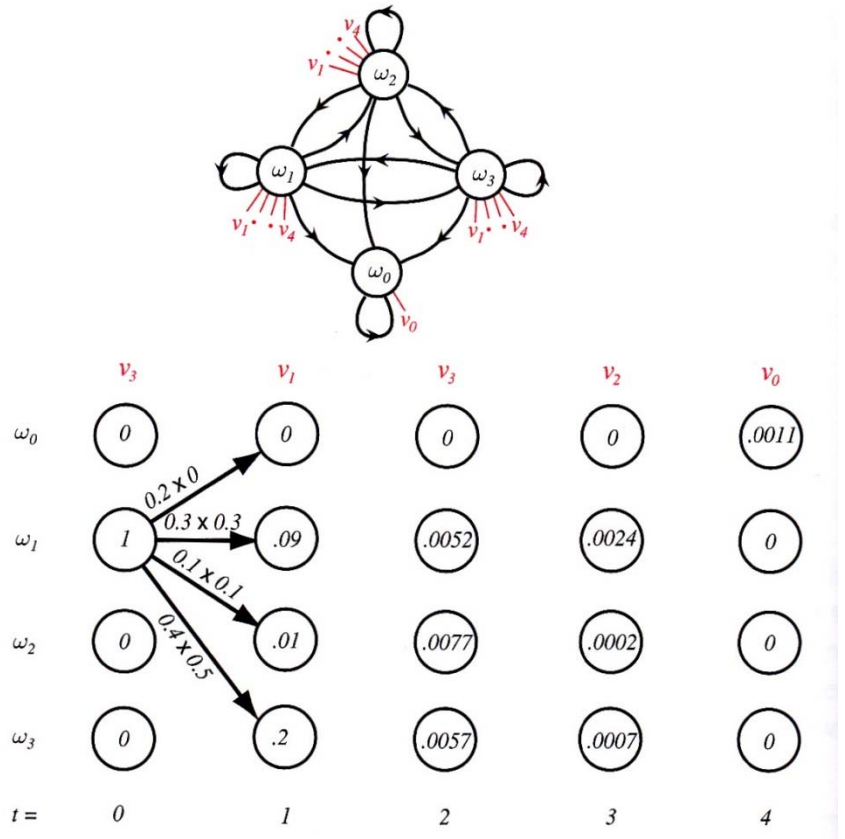
$$a_{ij} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.2 & 0.3 & 0.1 & 0.4 \\ 0.2 & 0.5 & 0.2 & 0.1 \\ 0.8 & 0.1 & 0.0 & 0.1 \end{pmatrix}, \text{ and } b_{jk} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0.4 & 0.1 & 0.2 \\ 0 & 0.1 & 0.1 & 0.7 & 0.1 \\ 0 & 0.5 & 0.2 & 0.1 & 0.2 \end{pmatrix}$$

$$i, j \in [0, 3], k \in [0,4]$$

1^{st} index$\rightarrow$ row, 2^{nd} index$\rightarrow$column

HMM Evaluation Example

- Assume the observed symbol sequence is $\boldsymbol{V^T}=\{\boldsymbol{v}_3, \boldsymbol{v}_1, \boldsymbol{v}_3, \boldsymbol{v}_2, \boldsymbol{v}_0\}$
- Assume the initial state is $\boldsymbol{\omega}_1$
- $t=0$, $\boldsymbol{v}(t=0)=\boldsymbol{v}_3$, $\alpha_1(0)=1$
- $t=1$, $\boldsymbol{v}(t=1)=\boldsymbol{v}_1$
  - $\alpha_0(1)=\alpha_1(0)a_{10}b_{01}=0$
  - $\alpha_1(1)=\alpha_1(0)a_{11}b_{11}=0.09$
  - $\alpha_2(1)=\alpha_1(0)a_{12}b_{21}=0.01$
  - $\alpha_3(1)=\alpha_1(0)a_{13}b_{31}=0.2$
- $t=2$, $\boldsymbol{v}(t=2)=\boldsymbol{v}_3$
- The final likelihood is $\boldsymbol{P(V^T|\theta)=0.0011}$



HMM Evaluation Problem

- The Bayes formular $$P(\theta \mid V^T) = \frac{P(V^T \mid \theta)P(\theta)}{P(V^T)}$$
- In HMM pattern recognition, we will construct several HMMs, one for each category.
- Classification of a test sequence should be based on the highest posterior probability.
- The prior probabilities are usually from domain knowledge, and frequently assumed to be uniform. Then the classification is based on the highest likelihood $\boldsymbol{P(V^T|\theta)}$

HMM Decoding Problem

- The HMM decoding algorithm (Viterbi algorithm)

1. initialize $t \leftarrow 0$, $path \leftarrow \{\}$
2. for $t \leftarrow t+1$
3. $j \leftarrow 1$
4. for $j \leftarrow j+1$
5. $$\alpha_j(t) \leftarrow \left[\sum_{i=1}^{c} \alpha_i(t-1)a_{ij}\right] b_{jk}(v(t)=v_k)$$
6. until $j=c$
7. $j' \leftarrow \arg\max_j \alpha_j(t)$
8. append $\omega_{j'}$ to $path$
9. until $t=T$
10. Return $path$

HMM Decoding Problem

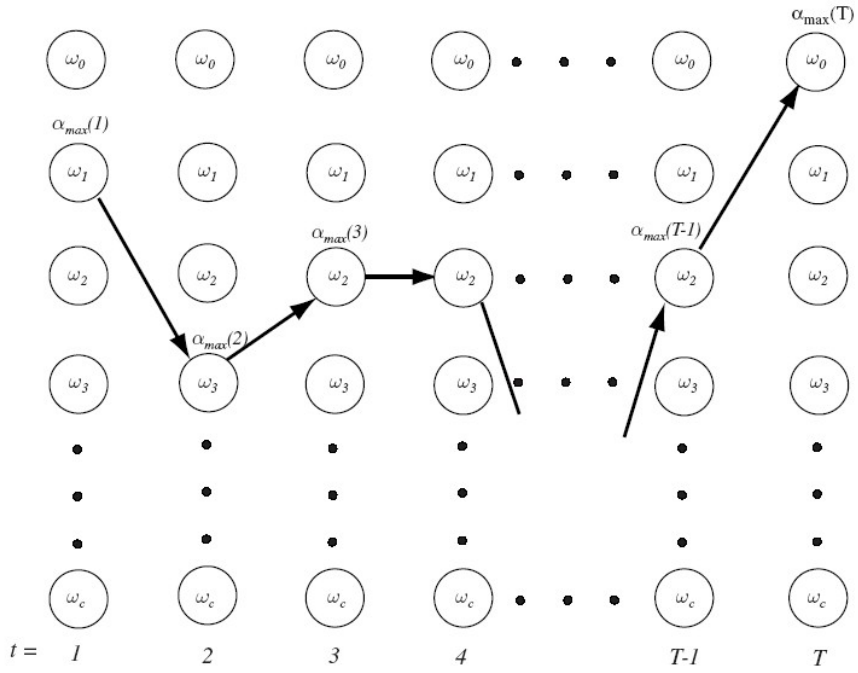
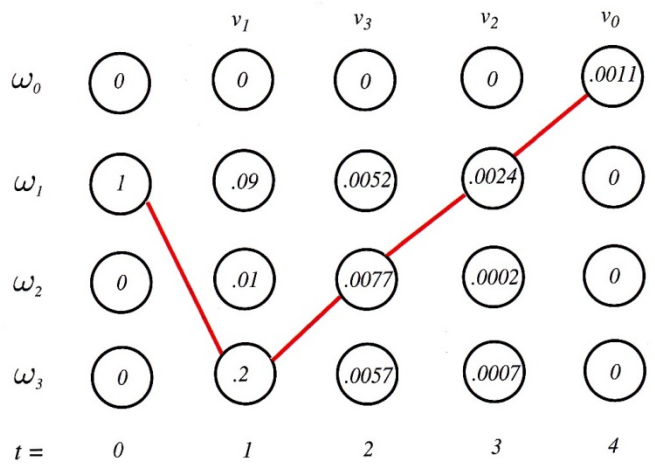


FIGURE 3.12. The decoding algorithm finds at each time step t the state that has the highest probability of having come from the previous step and generated the observed visible state v_k. The full path is the sequence of such states. Because this is a local optimization (dependent only upon the single previous time step, not the full sequence), the algorithm does not guarantee that the path is indeed allowable. For instance, it might be possible that the maximum at $t = 5$ is ω_1 and at $t = 6$ is ω_2, and thus these would appear in the path. This can even occur if $a_{12} = P(\omega_2(t+1)|\omega_1(t)) = 0$, precluding that transition. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

HMM Decoding Problem Example



The locally optimal path through the HMM trellis of Example 3.

- This decoding algorithm is a simplification of an exhaustive search of all possible paths, and the result is therefore sub-optimal (not necessary the true path).
- In this example, note that ω_3 to ω_2 transition is actually not allowed in matrix a_{ij}.

HMM Learning Problem

- There is no known method for obtaining the optimal or most likely set of parameters from the data.
- The forward-backward algorithm (Baum-Welch algorithm) is a generalized expectation maximization algorithm
- We define $\beta_i(t)$ to be the probability that the model is in state $\omega_i(t)$ and will generate the remainder of the given target sequence from step t to T.

$$\beta_i(t) = \begin{cases} 0 & t = T \text{ and } i \neq \text{final state} \\ 1 & t = T \text{ and } i = \text{final state} \\ \left[\sum_j \beta_j(t+1) a_{ij} b_{jk} (v(t+1) = v_k) \right] & \text{otherwise} \end{cases}$$

HMM Learning Problem

- The HMM backward algorithm

  1. initialize $t \leftarrow T$, a_{ij}, b_{jk}, V^T, $\beta_j(T)$
  2. for $t \leftarrow t - 1$
  3. $$\beta_j(t) = \left[\sum_{j=1}^{c} \beta_j(t+1) a_{ij} b_{jk} (v(t+1) = v_k) \right]$$
  4. until $t = 1$
  5. return $P(V^T) \leftarrow \beta_i(0)$ for the known initial state

- The backward algorithm is the time-reversed version of the forward algorithm. It can be used in the evaluation phase.

HMM Learning Problem

- We define γ_{ij} to be the probability of transition between $\omega_i(t\text{-}1)$ and $\omega_j(t)$ given the model generated the entire training sequence V^T by any path.

$$\gamma_{ij}(t) = \frac{\alpha_i(t-1)a_{ij}b_{jk}\beta_j(t)}{P(V^T \mid \theta)}$$

- The expected number of transitions between state $\omega_i(t\text{-}1)$ and $\omega_j(t)$ at any time in the sequence is $\sum_{t=1}^{T} \gamma_{ij}(t)$ and the total expected number of transitions from ω_i is $\sum_{t=1}^{T}\sum_{l=0}^{c} \gamma_{il}(t)$

HMM Learning Problem

- The improved estimate of the probability of a transition from $\omega_i(t\text{-}1)$ to $\omega_j(t)$ is

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} \gamma_{ij}(t)}{\sum_{t=1}^{T} \sum_{l} \gamma_{il}(t)}$$

- The improved estimate of b_{jk} is

$$\hat{b}_{jk} = \frac{\sum_{t=1, v(t)=v_k}^{T} \sum_{l} \gamma_{jl}(t)}{\sum_{t=1}^{T} \sum_{l} \gamma_{jl}(t)}$$

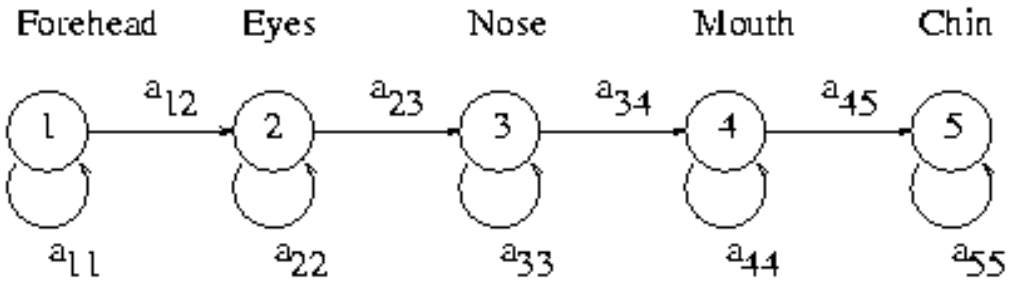
HMM Learning Problem

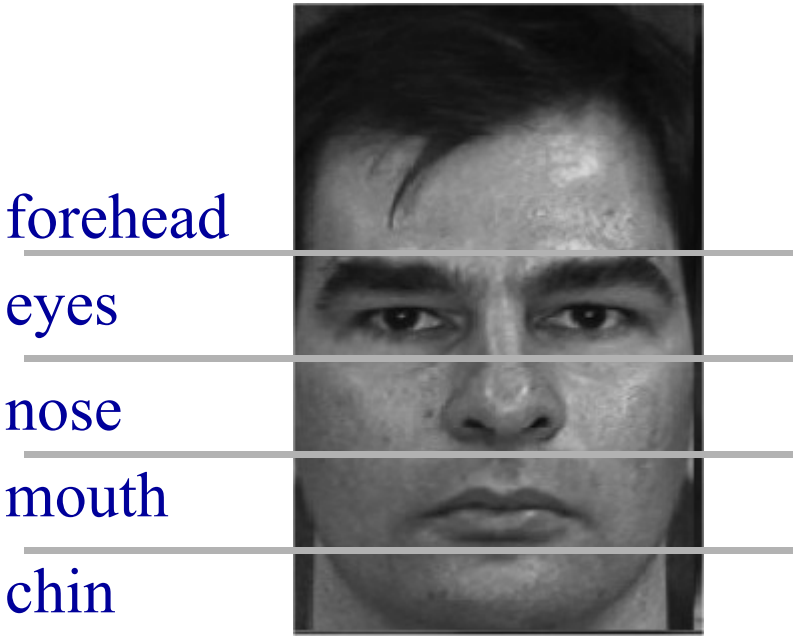
- The HMM forward-backward algorithm
  1. initialize a_{ij}, b_{jk}, V^T, ε, $z \leftarrow 0$
  2. do $z \leftarrow z + 1$,
  3. compute $\hat{a}(z)$ from $\boldsymbol{a}(\boldsymbol{z}\text{-}\boldsymbol{1})$ and $\boldsymbol{b}(\boldsymbol{z}\text{-}\boldsymbol{1})$
  4. compute $\hat{b}(z)$ from $\boldsymbol{a}(\boldsymbol{z}\text{-}\boldsymbol{1})$ and $\boldsymbol{b}(\boldsymbol{z}\text{-}\boldsymbol{1})$
  5. $a_{ij}(z) \leftarrow \hat{a}_{ij}(z)$
  6. $b_{jk}(z) \leftarrow \hat{b}_{jk}(z)$
  7. until $\mathbf{max}_{i,j,k}\ [\boldsymbol{a}_{ij}(\boldsymbol{z})\text{-}\boldsymbol{a}_{ij}(\boldsymbol{z}\text{-}\boldsymbol{1}), \boldsymbol{b}_{jk}(\boldsymbol{z})\text{-}\boldsymbol{b}_{jk}(\boldsymbol{z}\text{-}\boldsymbol{1})] < \boldsymbol{\varepsilon}$
  8. Return $a_{ij} \leftarrow a_{ij}(z)$ and $b_{jk} \leftarrow b_{jk}(z)$

Three Basic Problems of HMM

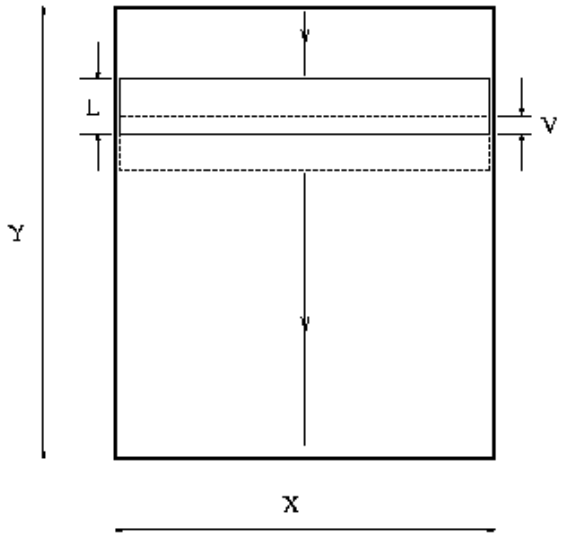
- Find $p(O|\Lambda)$: the probability of the observations (O) given the model (Λ): **forward algorithm**
- Find $p(Q|\Lambda,O)$: the most likely state trajectory (Q) given the model and observations: **Viterbi algorithm**
- Adjust $\Lambda=\{A,B,\Pi\}$ to maximize $p(O|\Lambda)$: **Baum-Welch algorithm**

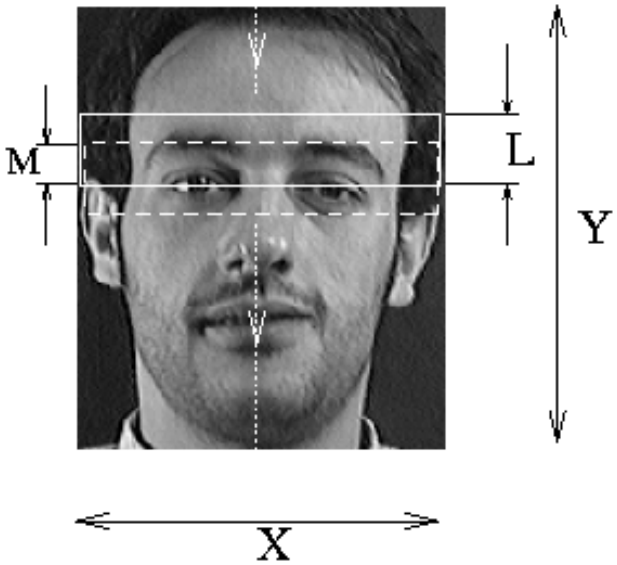
State Assignment

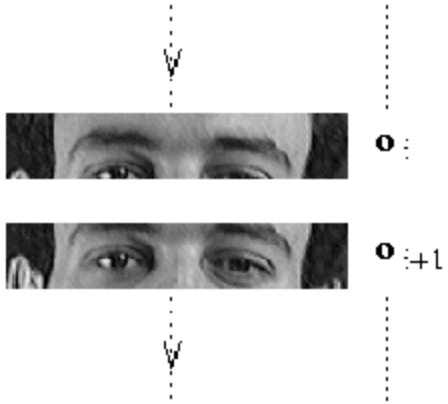




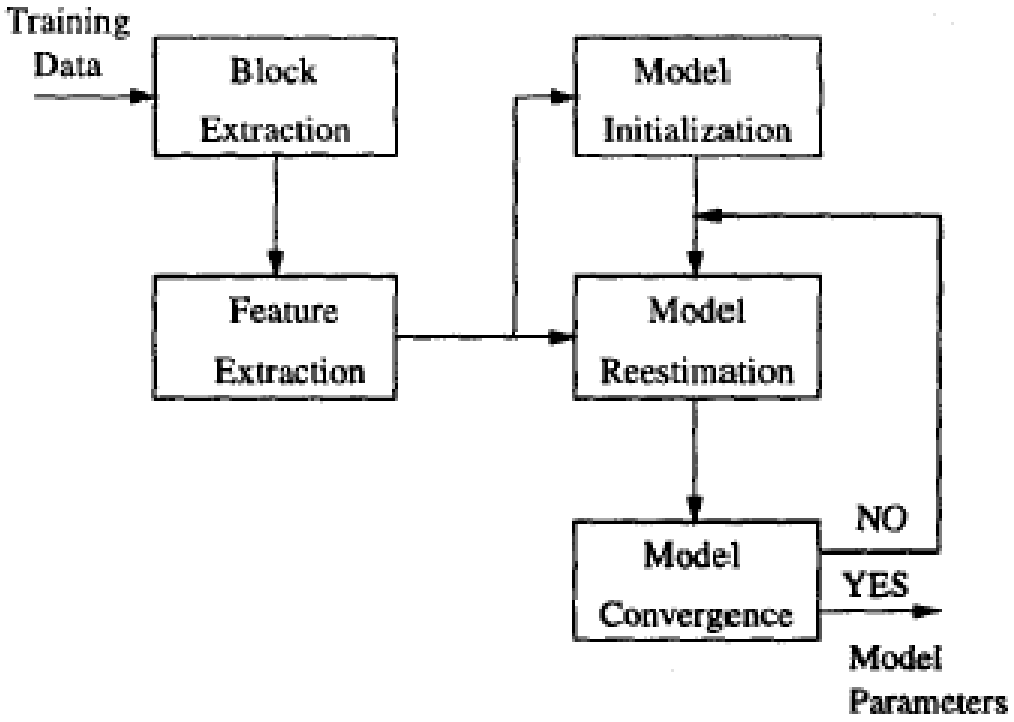
Sampling Scheme



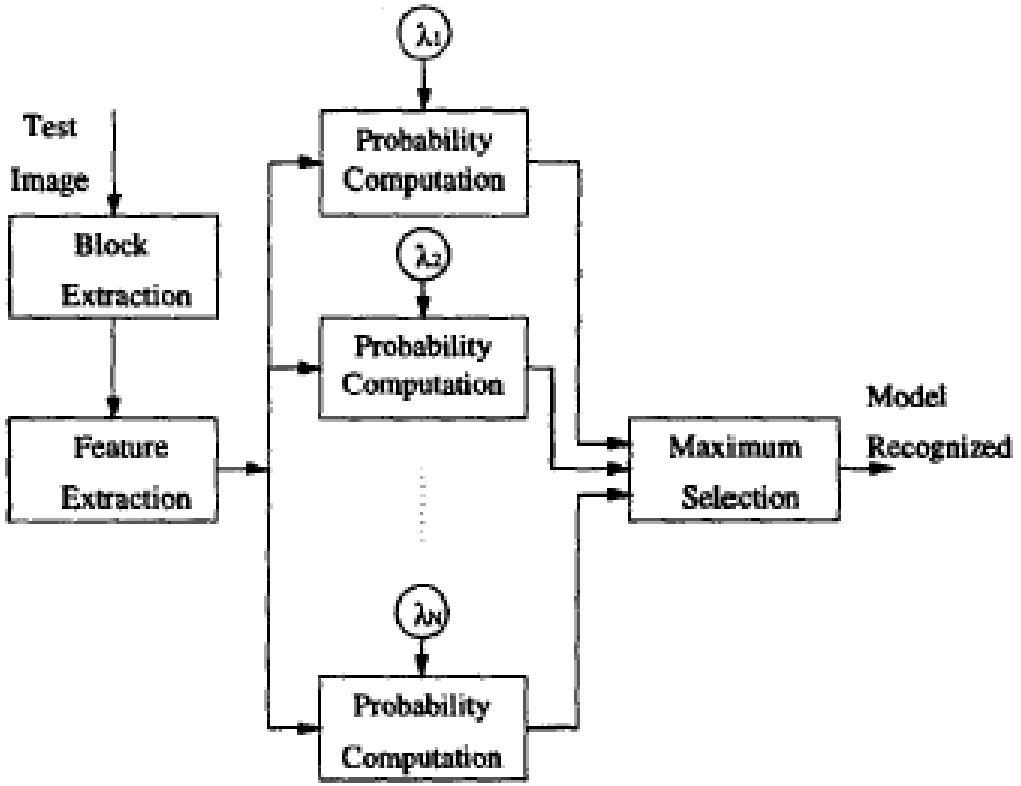




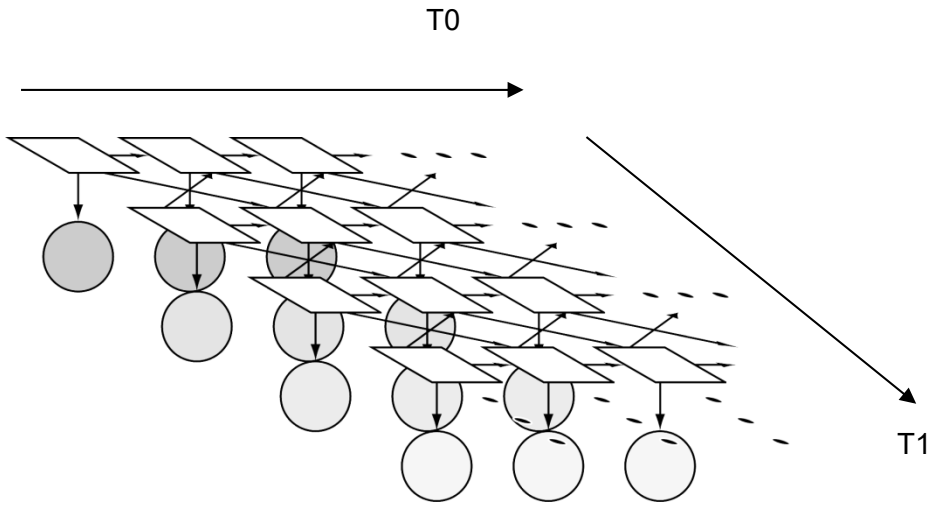
HMM Training Scheme



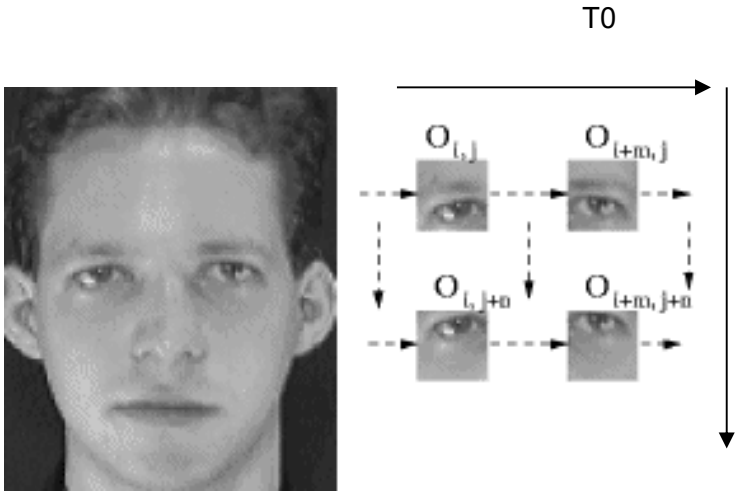
HMM Recognition Scheme



2-D HMM



A coupled HMM for face recognition



Face parameterization and feature extraction