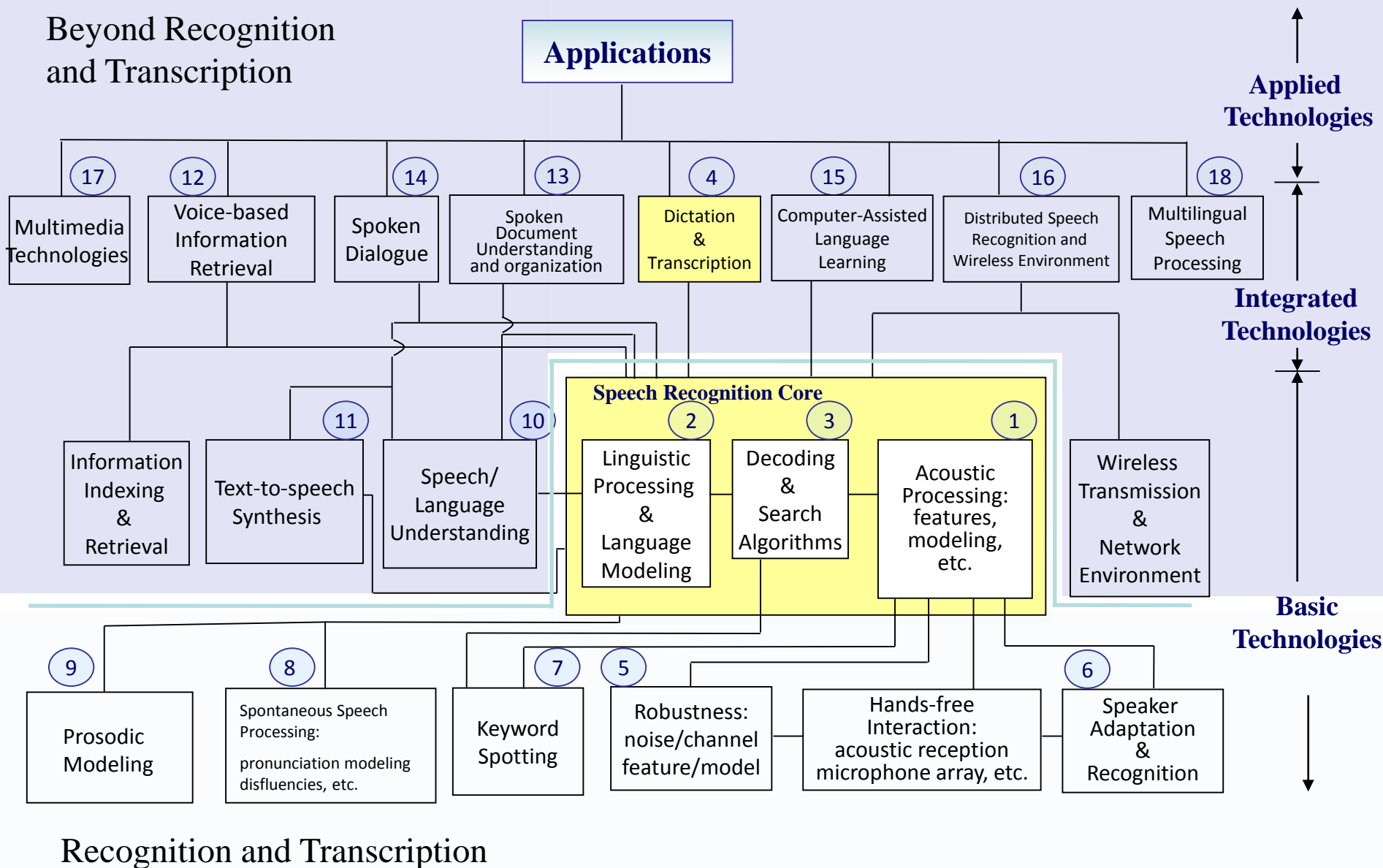


3.0 Map of Subject Areas

Hierarchy of Research Areas



4.0 More about Hidden Markov Models

Reference: 1. 6.1-6.6, Rabiner and Juang

2. 4.4.1 of Huang

Markov Model

- **Markov Model (Markov Chain)**

- First-order Markov chain of N states is a triplet $(S, \mathbf{A}, \boldsymbol{\pi})$

- S is a set of N states

- \mathbf{A} is the $N \times N$ matrix of state transition probabilities

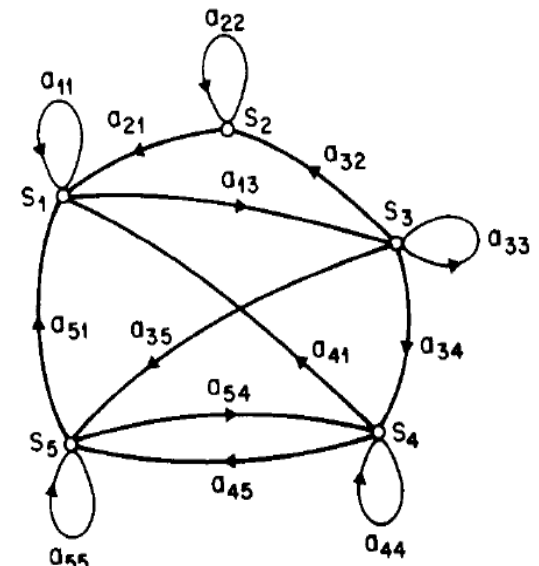
$$P(q_t=j|q_{t-1}=i, q_{t-2}=k, \dots) = P(q_t=j|q_{t-1}=i) \equiv \mathbf{a}_{ij}$$

- $\boldsymbol{\pi}$ is the vector of initial state probabilities

$$\pi_j = P(q_0=j)$$

- The output for any given state is an observable event (deterministic)

- The output of the process is a sequence of observable events



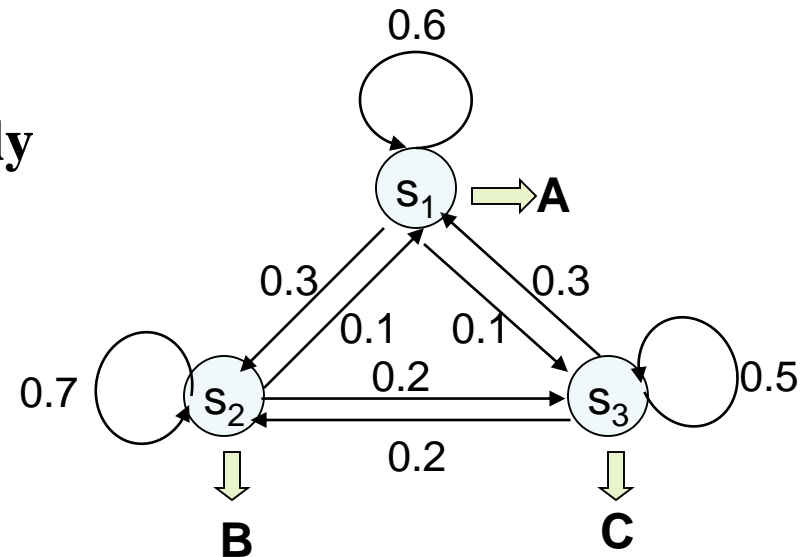
A Markov chain with 5 states (labeled S_1 to S_5) with state transitions.

Markov Model

- **An example : a 3-state Markov Chain λ**

- State 1 generates symbol **A only**,
State 2 generates symbol **B only**,
and State 3 generates symbol **C only**

$$\mathbf{A} = \begin{bmatrix} 0.6 & 0.3 & 0.1 \\ 0.1 & 0.7 & 0.2 \\ 0.3 & 0.2 & 0.5 \end{bmatrix}$$
$$\pi = [0.4 \quad 0.5 \quad 0.1]$$



- Given a sequence of observed symbols $\mathbf{O}=\{\text{CABBCABC}\}$, the **only one** corresponding state sequence is $\{S_3S_1S_2S_2S_3S_1S_2S_3\}$, and the corresponding probability is

$$P(\mathbf{O}|\lambda)=P(q_0=S_3)$$

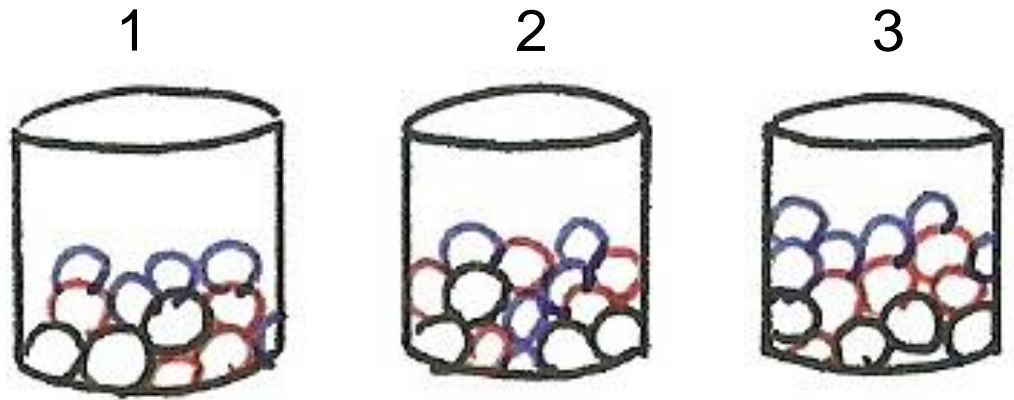
$$P(S_1/S_3)P(S_2/S_1)P(S_2/S_2)P(S_3/S_2)P(S_1/S_3)P(S_2/S_1)P(S_3/S_2)$$
$$=0.1 \times 0.3 \times 0.3 \times 0.7 \times 0.2 \times 0.3 \times 0.3 \times 0.2 = 0.00002268$$

Hidden Markov Model

- **HMM, an extended version of Markov Model**
 - The observation is **a probabilistic function (discrete or continuous) of a state** instead of an one-to-one correspondence of a state
 - The model is a **doubly embedded** stochastic process with an underlying stochastic process that is not directly observable (hidden)
 - What is hidden? *The State Sequence*
According to the observation sequence, we never know which state sequence generates it
- **Elements of an HMM $\{S, A, B, \pi\}$**
 - S is a set of N states
 - A is the $N \times N$ matrix of state transition probabilities
 - B is a set of N probability functions, each describing the observation probability with respect to a state
 - π is the vector of initial state probabilities

Simplified HMM

RGBGGBBGRRR.....



Hidden Markov Model

- **Two types of HMM's according to the observation functions**

Discrete and finite observations :

- The observations that **all** distinct states generate are finite in number
 $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \dots, \mathbf{v}_M\}, \mathbf{v}_k \in \mathbf{R}^D$
- the set of observation probability distributions $B = \{b_j(\mathbf{v}_k)\}$ is defined as
 $b_j(\mathbf{v}_k) = P(\mathbf{o}_t = \mathbf{v}_k | \mathbf{q}_t = j), 1 \leq k \leq M, 1 \leq j \leq N$
 \mathbf{o}_t : observation at time t , \mathbf{q}_t : state at time t
 \Rightarrow for state j , $b_j(\mathbf{v}_k)$ consists of **only M probability values**

Continuous and infinite observations :

- The observations that **all** distinct states generate are infinite and continuous,
 $\mathbf{V} = \{\mathbf{v} | \mathbf{v} \in \mathbf{R}^D\}$
- the set of observation probability distributions $B = \{b_j(\mathbf{v})\}$ is defined as
 $b_j(\mathbf{v}) = P(\mathbf{o}_t = \mathbf{v} | \mathbf{q}_t = j), 1 \leq j \leq N$
 $\Rightarrow b_j(\mathbf{v})$ is a **continuous probability density function** and is often assumed to be a mixture of Gaussian distributions

$$b_j(\mathbf{v}) = \sum_{k=1}^M c_{jk} \left(\frac{1}{(\sqrt{2\pi})^D |\Sigma_{jk}|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} ((\mathbf{v} - \boldsymbol{\mu}_{jk})^T \Sigma_{jk}^{-1} (\mathbf{v} - \boldsymbol{\mu}_{jk})) \right) \right) = \sum_{k=1}^M c_{jk} b_{jk}(\mathbf{v})$$

Hidden Markov Model

- An example : a 3-state discrete HMM λ

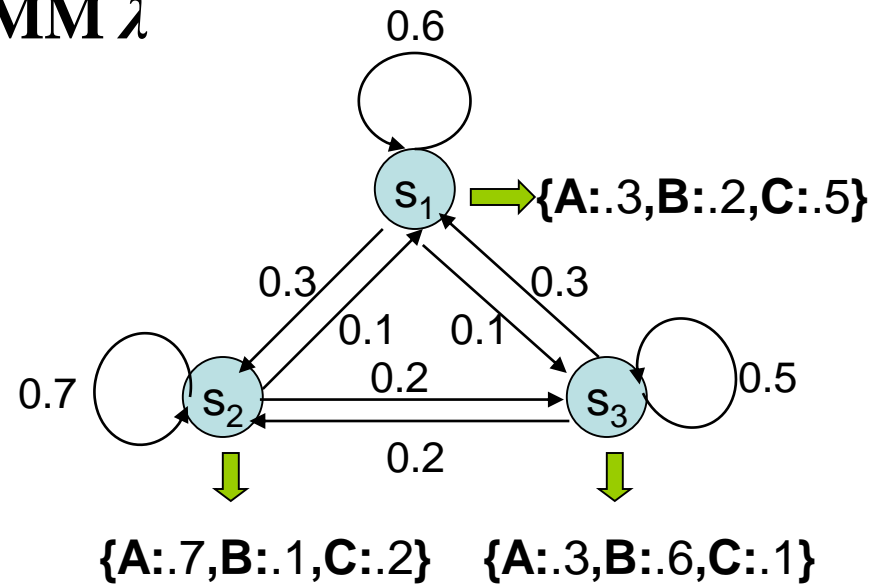
$$\mathbf{A} = \begin{bmatrix} 0.6 & 0.3 & 0.1 \\ 0.1 & 0.7 & 0.2 \\ 0.3 & 0.2 & 0.5 \end{bmatrix}$$

$$b_1(\mathbf{A}) = 0.3, b_1(\mathbf{B}) = 0.2, b_1(\mathbf{C}) = 0.5$$

$$b_2(\mathbf{A}) = 0.7, b_2(\mathbf{B}) = 0.1, b_2(\mathbf{C}) = 0.2$$

$$b_3(\mathbf{A}) = 0.3, b_3(\mathbf{B}) = 0.6, b_3(\mathbf{C}) = 0.1$$

$$\pi = [0.4 \quad 0.5 \quad 0.1]$$



- Given a sequence of observations $\bar{\mathbf{O}} = \{\text{ABC}\}$, there are **27 possible** corresponding state sequences, and therefore the corresponding probability is

$$P(\bar{\mathbf{O}}|\lambda) = \sum_{i=1}^{27} P(\bar{\mathbf{O}}, \mathbf{q}_i|\lambda) = \sum_{i=1}^{27} P(\bar{\mathbf{O}}|\mathbf{q}_i, \lambda) P(\mathbf{q}_i|\lambda), \quad \mathbf{q}_i : \text{statesequence}$$

$$e.g. \text{ when } \mathbf{q}_i = \{s_2 s_2 s_3\}, P(\bar{\mathbf{O}}|\mathbf{q}_i, \lambda) = P(\mathbf{A}|s_2) P(\mathbf{B}|s_2) P(\mathbf{C}|s_3) = 0.7 * 0.1 * 0.1 = 0.007$$

$$P(\mathbf{q}_i|\lambda) = P(q_0 = s_2) P(s_2|s_2) P(s_3|s_2) = 0.5 * 0.7 * 0.2 = 0.07$$

Hidden Markov Model

- **Three Basic Problems for HMMs**

Given an observation sequence $\bar{O}=(o_1,o_2,\dots,o_T)$, and an HMM

$\lambda=(A,B,\pi)$

- Problem 1 :

How to *efficiently* compute $P(\bar{O} | \lambda)$?

\Rightarrow *Evaluation problem*

- Problem 2 :

How to choose an optimal state sequence $\mathbf{q}=(q_1,q_2,\dots,q_T)$?

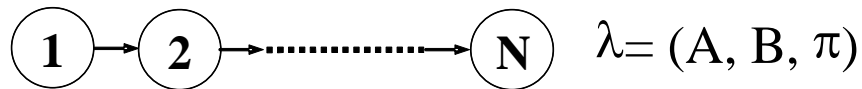
\Rightarrow *Decoding Problem*

- Problem 3 :

Given some observations \bar{O} for the HMM λ , how to adjust the model parameter $\lambda=(A,B,\pi)$ to maximize $P(\bar{O} | \lambda)$?

\Rightarrow *Learning /Training Problem*

Basic Problem 1 for HMM



$\bar{O} = o_1 o_2 o_3 \dots o_t \dots o_T$ observation sequence

$\bar{q} = q_1 q_2 q_3 \dots q_t \dots q_T$ state sequence

- **Problem 1:** Given λ and \bar{O} ,
find $P(\bar{O}|\lambda) = \text{Prob}[\text{observing } \bar{O} \text{ given } \lambda]$
- **Direct Evaluation:** considering all possible state sequence \bar{q}

$$P(\bar{O}|\lambda) = \sum_{\text{all } \bar{q}} P(\bar{O}, \bar{q}|\lambda) = \sum_{\text{all } \bar{q}} P(\bar{O}|\bar{q}, \lambda) P(\bar{q}|\lambda)$$

$$P(\bar{O}|\bar{q}, \lambda)$$

$$P(\bar{O}|\lambda) = \sum_{\text{all } \bar{q}} \left(\begin{array}{c} \uparrow \\ [b_{q_1}(o_1) \cdot b_{q_2}(o_2) \cdot \dots \cdot b_{q_T}(o_T)] \cdot \\ [\pi_{q_1} \cdot a_{q_1 q_2} \cdot a_{q_2 q_3} \cdot \dots \cdot a_{q_{T-1} q_T}] \end{array} \right)$$



$$P(\bar{q}|\lambda)$$

total number of different \bar{q} : N^T

huge computation requirements

Basic Problem 1 for HMM

- **Forward Algorithm: defining a forward variable $\alpha_t(i)$**

$$\alpha_t(i) = P(o_1 o_2 \dots o_t, q_t = i | \lambda)$$

$$= \text{Prob}[\text{observing } o_1 o_2 \dots o_t, \text{ state } i \text{ at time } t | \lambda]$$

- Initialization

$$\alpha_1(i) = \pi_i b_i(o_1), \quad 1 \leq i \leq N$$

- Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(o_{t+1})$$

$$1 \leq j \leq N$$

$$1 \leq t \leq T-1$$

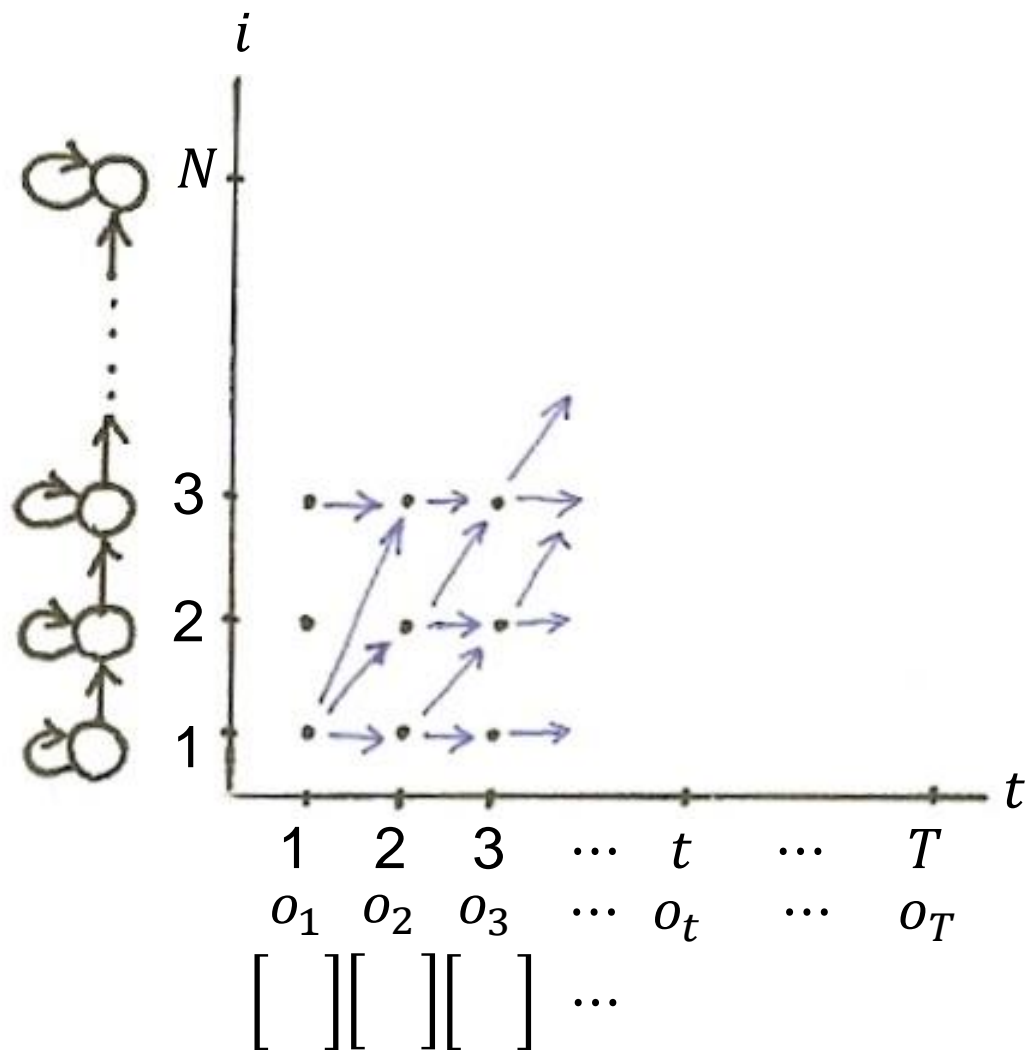
- Termination

$$P(\bar{O} | \lambda) = \sum_{i=1}^N \alpha_T(i)$$

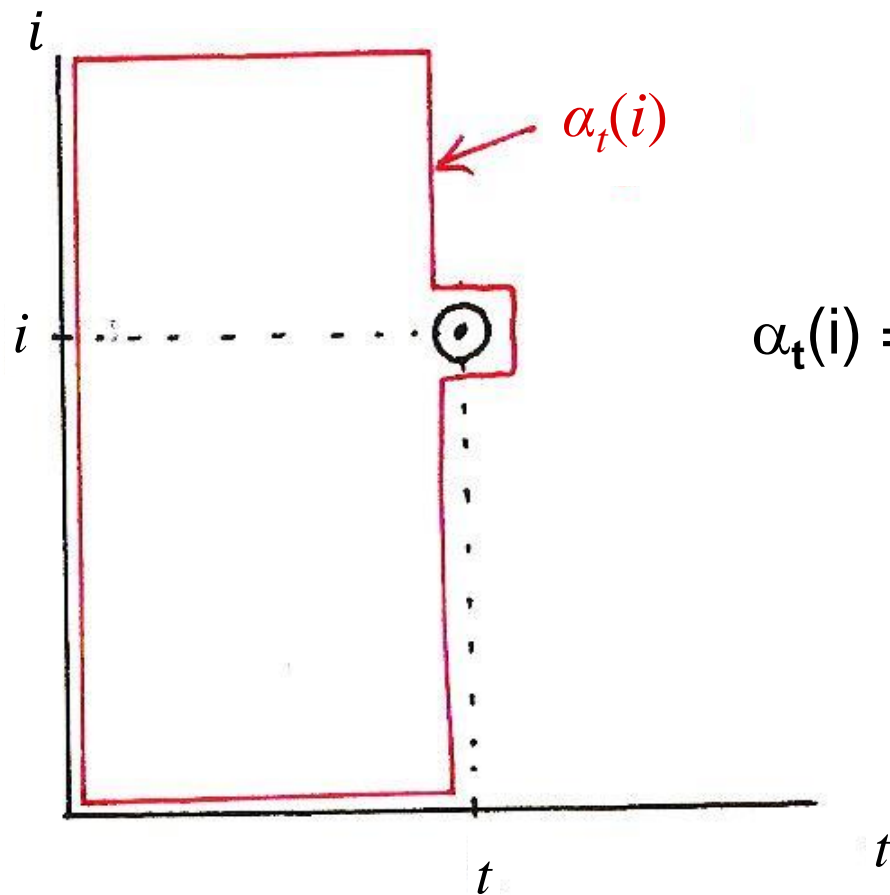
See Fig. 6.5 of Rabiner and Juang

- All state sequences, regardless of how long previously, merge to the N state at each time instant t

Basic Problem 1

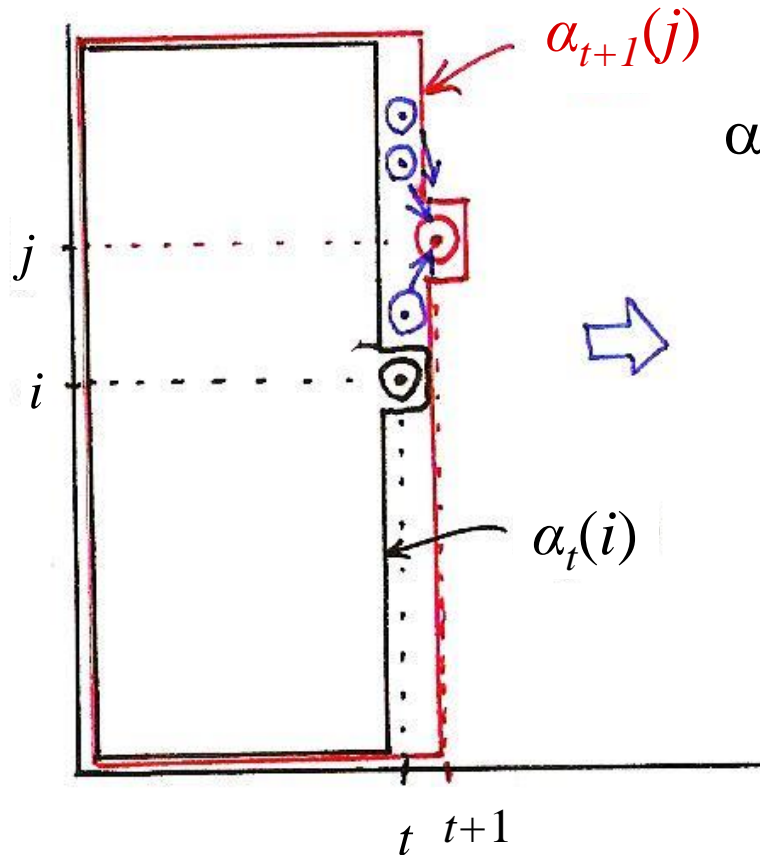


Basic Problem 1



$$\alpha_t(i) = P(o_1 o_2 \dots o_t, q_t = i | \lambda)$$

Basic Problem 1



$$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(o_{t+1})$$

$$1 \leq j \leq N$$

$$1 \leq t \leq T-1$$

Forward Algorithm

Basic Problem 2 for HMM

- **Problem 2:** Given λ and $\bar{O} = o_1 o_2 \dots o_T$, find a best state sequence $\bar{q} = q_1 q_2 \dots q_T$
- **Backward Algorithm :** defining a backward variable $\beta_t(i)$

$$\begin{aligned}\beta_t(i) &= P(o_{t+1}, o_{t+2}, \dots, o_T | q_t = i, \lambda) \\ &= \text{Prob}[\text{observing } o_{t+1}, o_{t+2}, \dots, o_T | \text{state } i \text{ at time } t, \lambda]\end{aligned}$$

- Initialization

$$\beta_T(i) = 1, \quad 1 \leq i \leq N \quad \left(\beta_{T-1}(i) = \sum_{j=1}^N a_{ij} b_j(o_T) \right)$$

- Induction

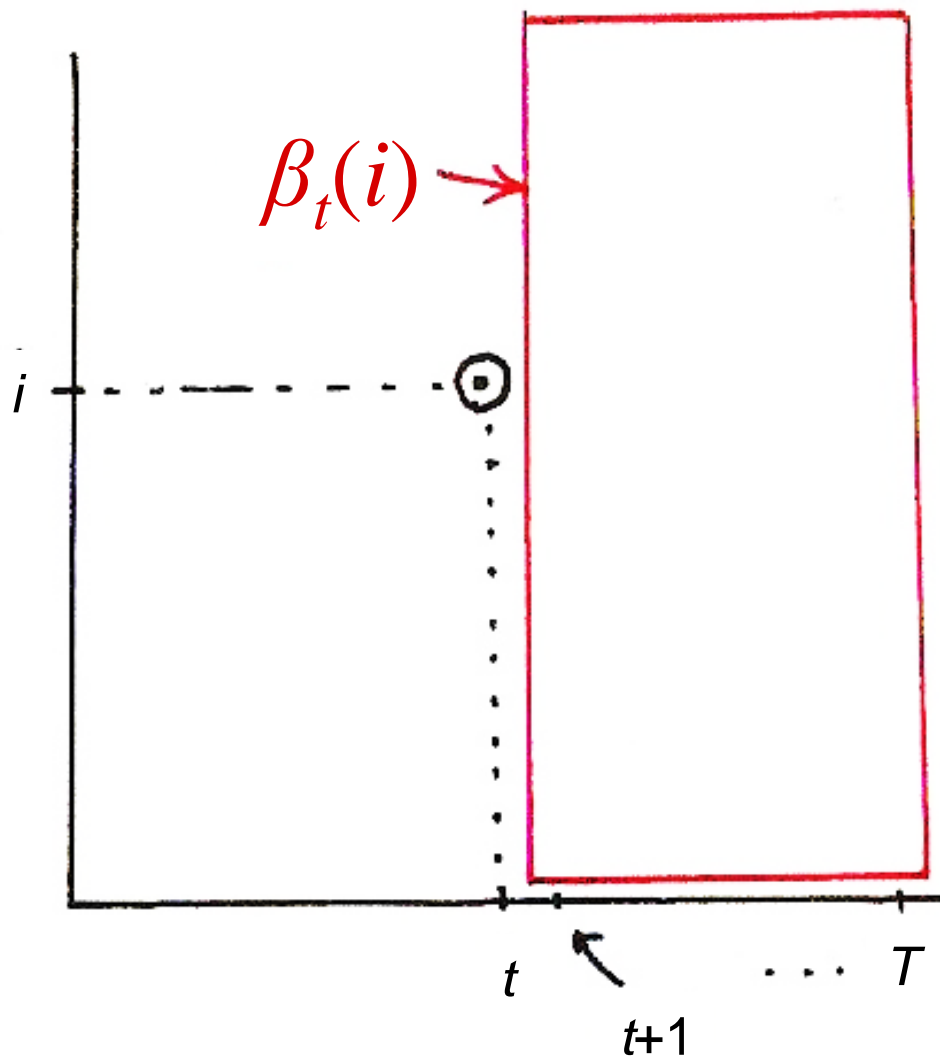
$$\begin{aligned}\beta_t(i) &= \sum_{j=1}^N a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \\ t &= T-1, T-2, \dots, 2, 1, \quad 1 \leq i \leq N\end{aligned}$$

See Fig. 6.6 of Rabiner and Juang

- **Combining Forward/Backward Variables**

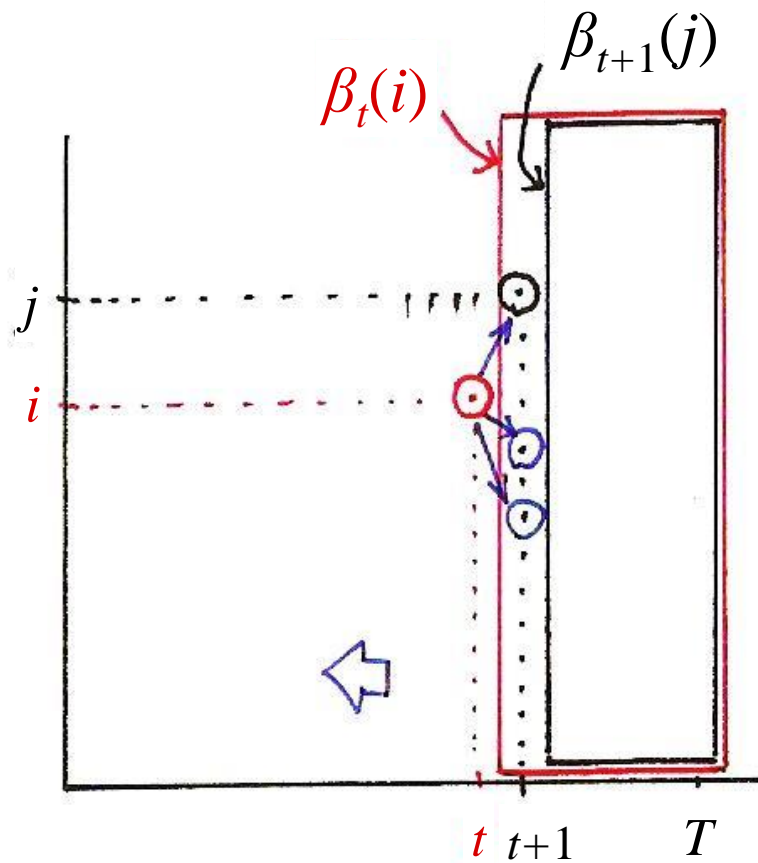
$$\begin{aligned}&P(\bar{O}, q_t = i | \lambda) \\ &= \text{Prob} [\text{observing } o_1, o_2, \dots, o_t, \dots, o_T, q_t = i | \lambda] \\ &= \alpha_t(i) \beta_t(i) \\ &P(\bar{O} | \lambda) = \sum_{i=1}^N P(\bar{O}, q_t = i | \lambda) = \sum_{i=1}^N [\alpha_t(i) \beta_t(i)]\end{aligned}$$

Basic Problem 2



$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \dots, o_T | q_t = i, \lambda)$$

Basic Problem 2

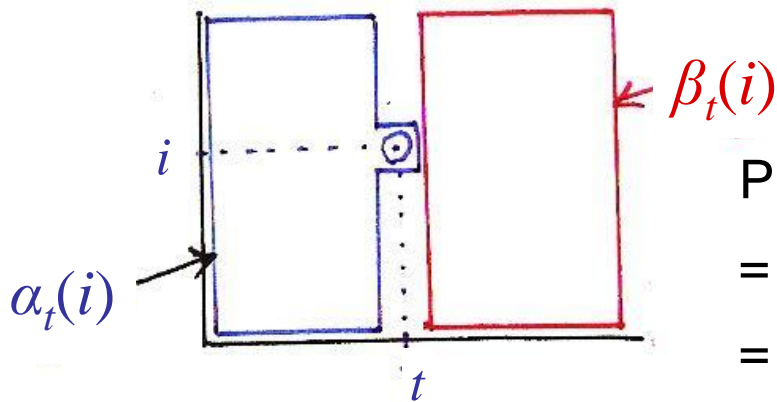


$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

$$t = T-1, T-2, \dots, 2, 1, \quad 1 \leq i \leq N$$

Backward Algorithm

Basic Problem 2



$$P(\bar{O}, q_t = i | \lambda)$$

$$= \text{Prob} [\text{observing } o_1, o_2, \dots, o_t, \dots, o_T, q_t = i | \lambda]$$

$$= \alpha_t(i) \beta_t(i)$$

$$A: (o_1 o_2 \cdots o_t | \lambda)$$

$$B: (o_{t+1}, o_{t+2}, \cdots o_T | \lambda)$$

$$C: (q_t = i | \lambda)$$

$$P(A, B, C) = P(A, C) P(B | A, C)$$

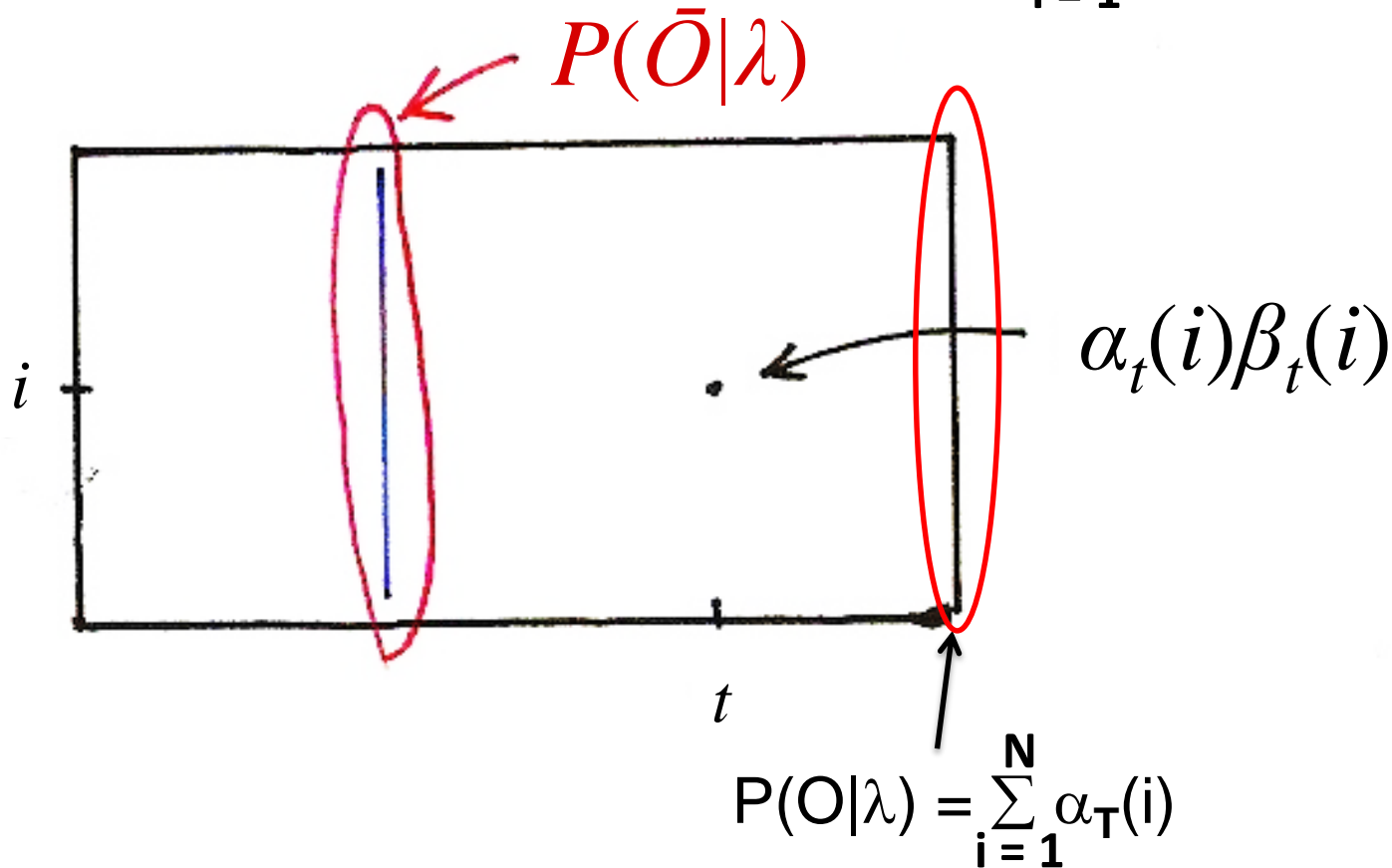
$$\begin{array}{ccc} // & // & // \end{array} \quad (B \perp A)$$

$$P(\bar{O}, q_t = i | \lambda) \quad \alpha_t(i) \quad P(B | C)$$

$$\begin{array}{c} // \\ \beta_t(i) \end{array}$$

Basic Problem 2

$$P(\bar{O}|\lambda) = \sum_{i=1}^N P(\bar{O}, q_t = i | \lambda) = \sum_{i=1}^N [\alpha_t(i)\beta_t(i)]$$



Basic Problem 2 for HMM

- **Approach 1 – Choosing state q_t^* individually as the most likely state at time t**

- Define a new variable $\gamma_t(i) = P(q_t = i \mid \bar{O}, \lambda)$

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^N \alpha_t(i)\beta_t(i)} = \frac{P(\bar{O}, q_t = i \mid \lambda)}{P(\bar{O} \mid \lambda)}$$

- Solution

$$q_t^* = \arg \max_{1 \leq i \leq N} [\gamma_t(i)], \quad 1 \leq t \leq T$$

in fact

$$\begin{aligned} q_t^* &= \arg \max_{1 \leq i \leq N} [P(\bar{O}, q_t = i \mid \lambda)] \\ &= \arg \max_{1 \leq i \leq N} [\alpha_t(i)\beta_t(i)] \end{aligned}$$

- Problem

maximizing the probability at each time t individually

$\bar{q}^* = q_1^* q_2^* \dots q_T^*$ may not be a valid sequence

(e.g. $a_{q_t^* q_{t+1}^*} = 0$)

Basic Problem 2 for HMM

- **Approach 2 —Viterbi Algorithm - finding the single best sequence**

$$\bar{q}^* = q_1^* q_2^* \dots q_T^*$$

- Define a new variable $\delta_t(i)$

$$\delta_t(i) = \max_{q_1, q_2, \dots, q_{t-1}} P[q_1, q_2, \dots, q_{t-1}, q_t = i, o_1, o_2, \dots, o_t | \lambda]$$

= the highest probability along a certain single path ending at state i at time t for the first t observations, given λ

- Induction

$$\delta_{t+1}(j) = \max_i [\delta_t(i) a_{ij}] \cdot b_j(o_{t+1})$$

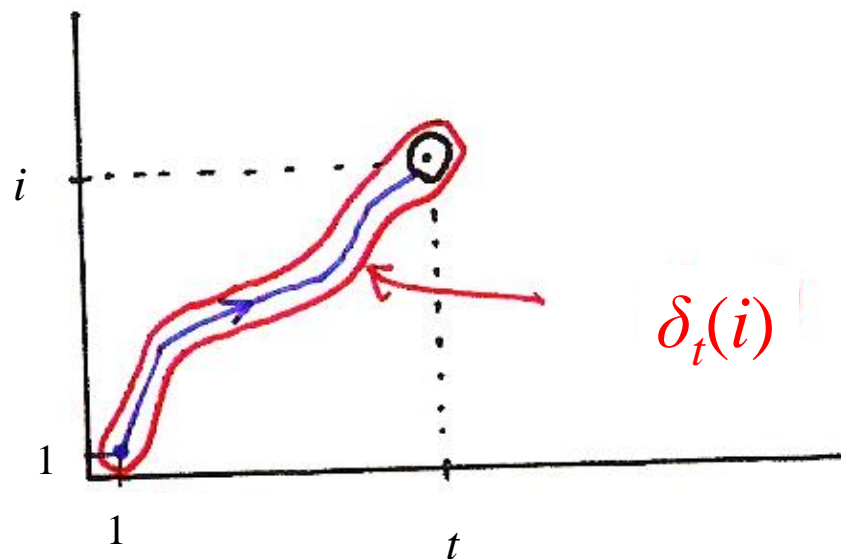
- Backtracking

$$\psi_{t+1}(j) = \arg \max_{1 \leq i \leq N} [\delta_t(i) a_{ij}]$$

the best previous state at $t-1$ given at state j at time t

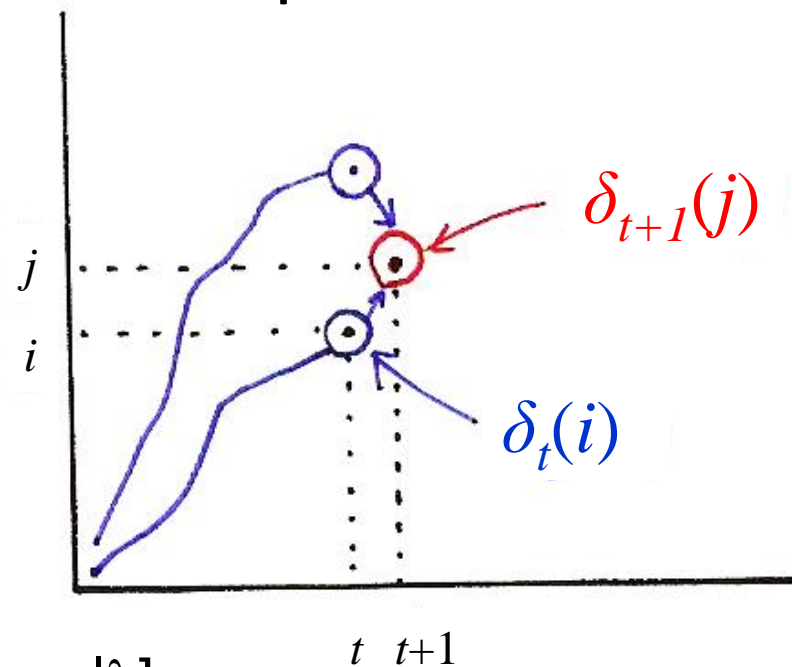
keeping track of the best previous state for each j and t

Viterbi Algorithm

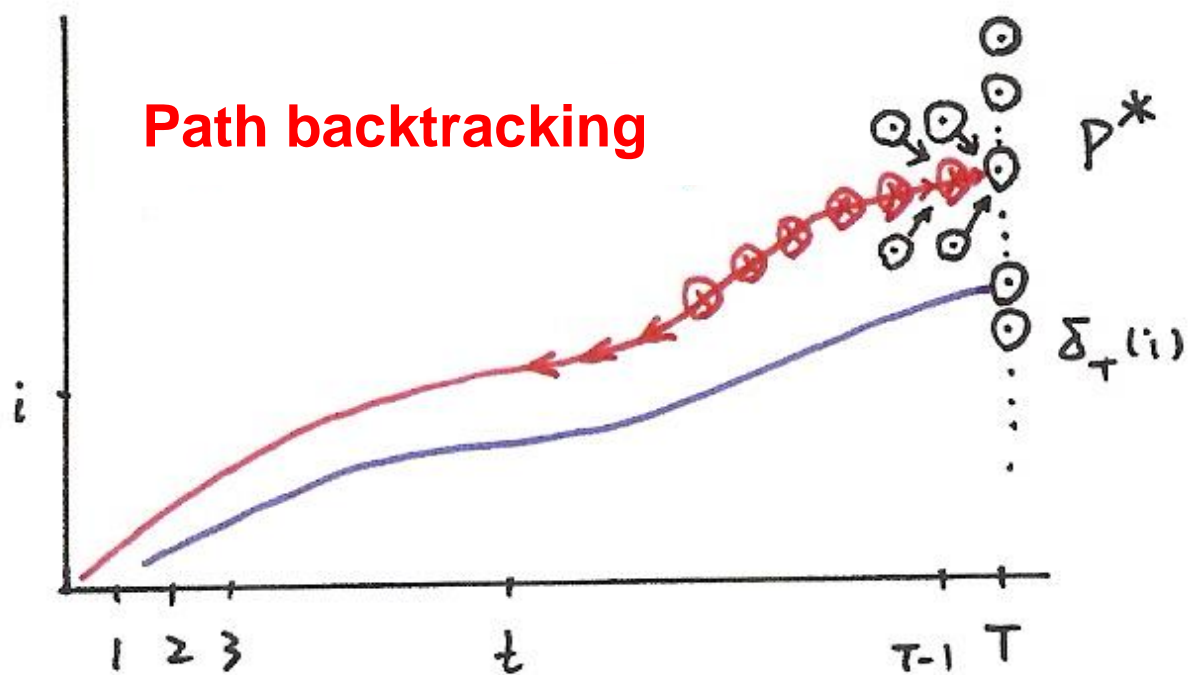


$$\delta_t(i) = \max_{q_1, q_2, \dots, q_{t-1}} P[q_1, q_2, \dots, q_{t-1}, q_t = i, o_1, o_2, \dots, o_t | \lambda]$$

$$\delta_{t+1}(j) = \max_i [\delta_t(i) a_{ij}] \cdot b_j(o_{t+1})$$



Viterbi Algorithm



Basic Problem 2 for HMM

- **Complete Procedure for Viterbi Algorithm**

- Initialization

$$\delta_1(i) = \pi_i b_i(o_1), \quad 1 \leq i \leq N$$

- Recursion

$$\delta_{t+1}(j) = \max_{1 \leq i \leq N} [\delta_t(i) a_{ij}] \cdot b_j(o_{t+1})$$

$$1 \leq t \leq T-1, \quad 1 \leq j \leq N$$

$$\psi_{t+1}(j) = \arg \max_{1 \leq i \leq N} [\delta_t(i) a_{ij}]$$

$$1 \leq t \leq T-1, \quad 1 \leq j \leq N$$

- Termination

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)]$$

$$q_T^* = \arg \max_{1 \leq i \leq N} [\delta_T(i)]$$

- Path backtracking

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \dots, 2, 1$$

Basic Problem 2 for HMM

- **Application Example of Viterbi Algorithm**

- Isolated word recognition

$$\lambda_0 = (A_0, B_0, \pi_0)$$

$$\lambda_1 = (A_1, B_1, \pi_1)$$

$$\vdots$$

$$\lambda_n = (A_n, B_n, \pi_n)$$

observation

$$\bar{O} = (o_1, o_2, \dots, o_T)$$

$$k^* = \arg \max_{1 \leq i \leq n} P[\bar{O} | \lambda_i] \approx \arg \max_{1 \leq i \leq n} [P^* | \lambda_i]$$



Basic Problem 1
Forward Algorithm
(for all paths)



Basic Problem 2
Viterbi Algorithm
(for a single best path)

-The model with the highest probability for the most probable path usually also has the highest probability for all possible paths.

Basic Problem 3 for HMM

- **Problem 3:** Give \bar{O} and an initial model $\lambda=(A,B,\pi)$, adjust λ to maximize $P(\bar{O}|\lambda)$
 - Baum-Welch Algorithm (Forward-backward Algorithm)
 - Define a new variable

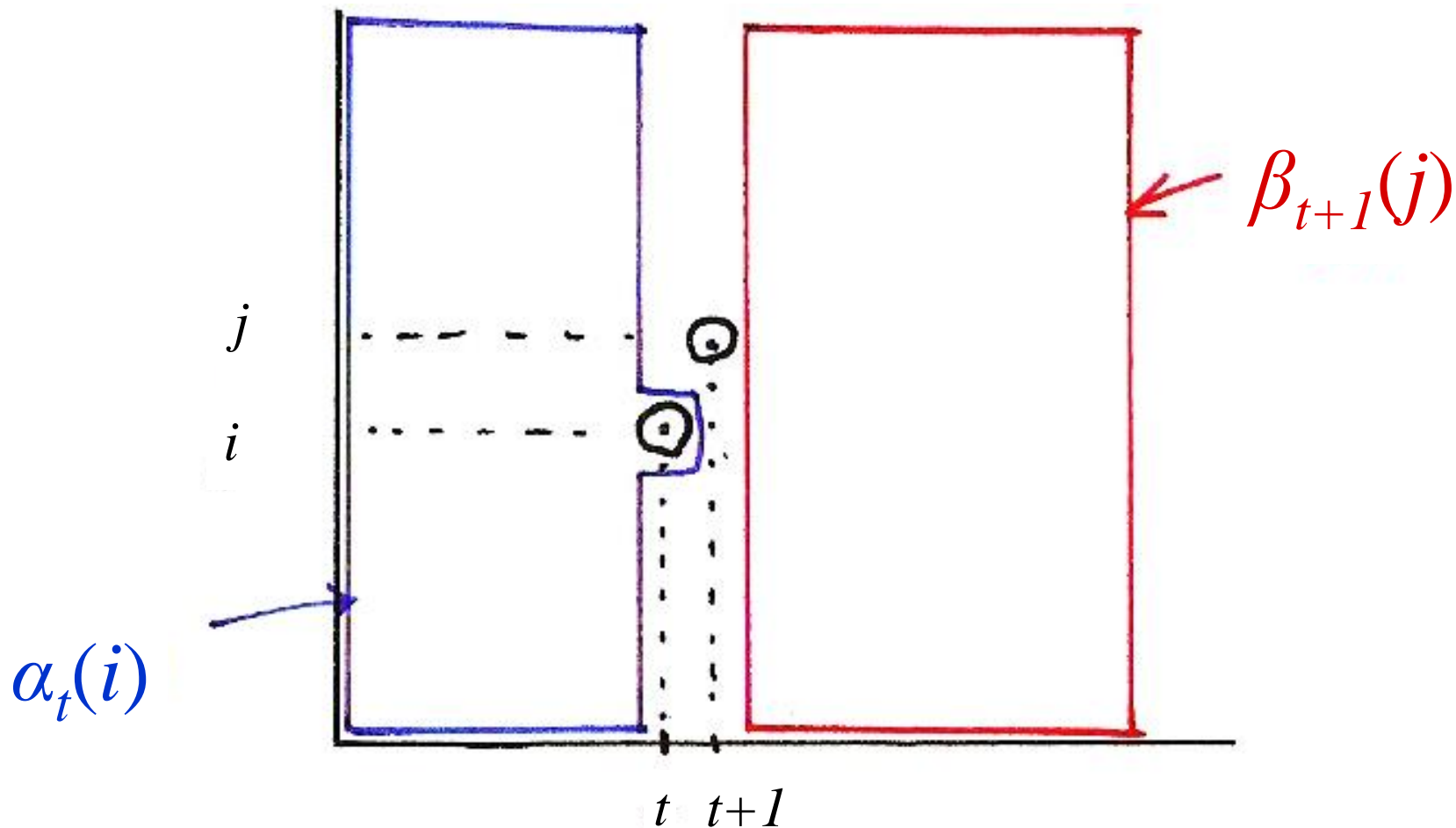
$$\begin{aligned}\epsilon_t(i, j) &= P(q_t = i, q_{t+1} = j \mid \bar{O}, \lambda) \\ &= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^N \sum_{j=1}^N [\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)]} \\ &= \frac{\text{Prob}[\bar{O}, q_t = i, q_{t+1} = j \mid \lambda]}{P(\bar{O} \mid \lambda)}\end{aligned}$$

See Fig. 6.7 of Rabiner and Juang

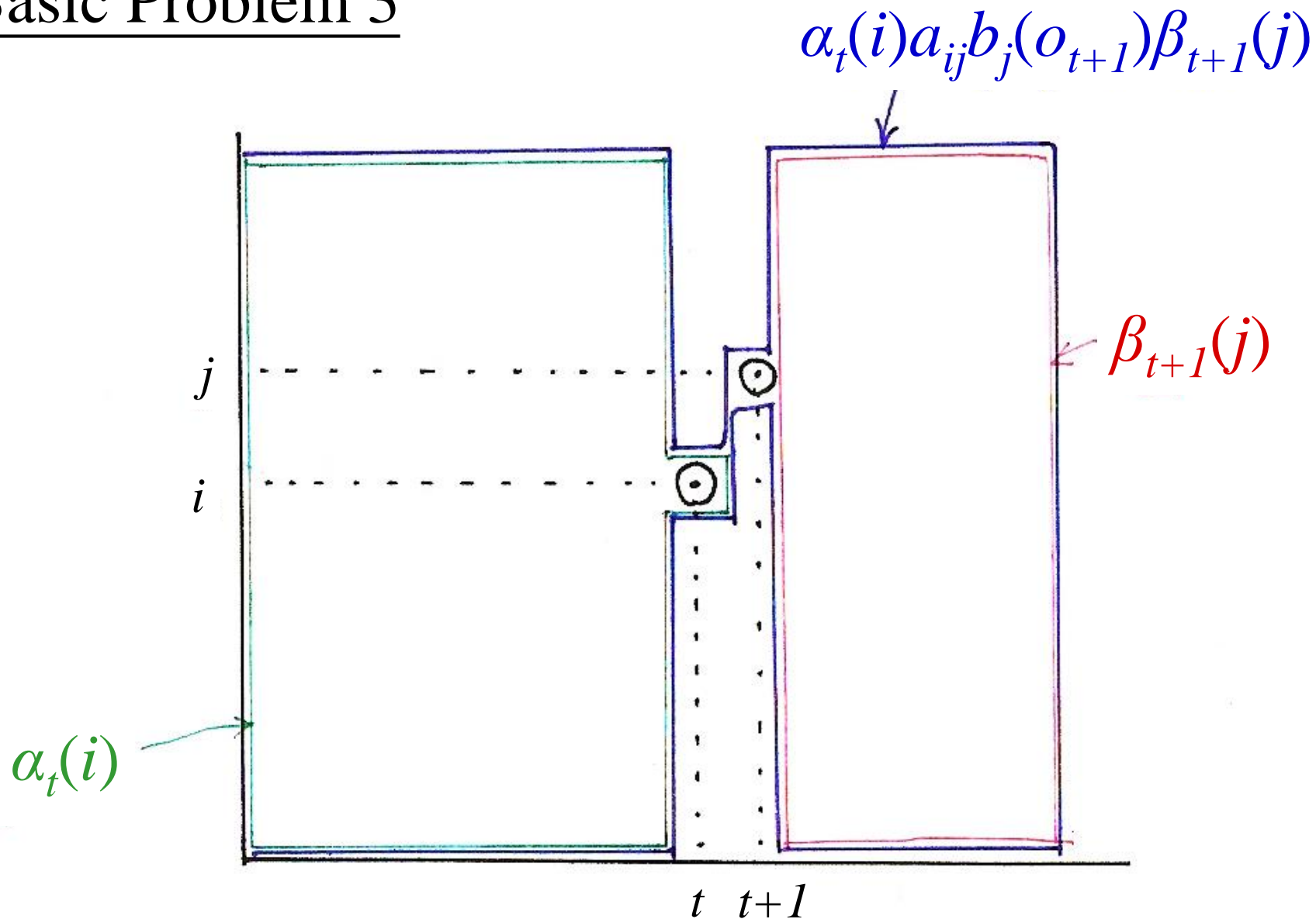
- Recall $\gamma_t(i) = P(q_t = i \mid \bar{O}, \lambda)$
 $\sum_{t=1}^{T-1} \gamma_t(i) =$ expected number of times that state i is visited in \bar{O} from $t = 1$ to $t = T-1$
 $=$ expected number of transitions from state i in \bar{O}
 $\sum_{t=1}^{T-1} \epsilon_t(i, j) =$ expected number of transitions from state i to state j in \bar{O}

Basic Problem 3

$$\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$



Basic Problem 3

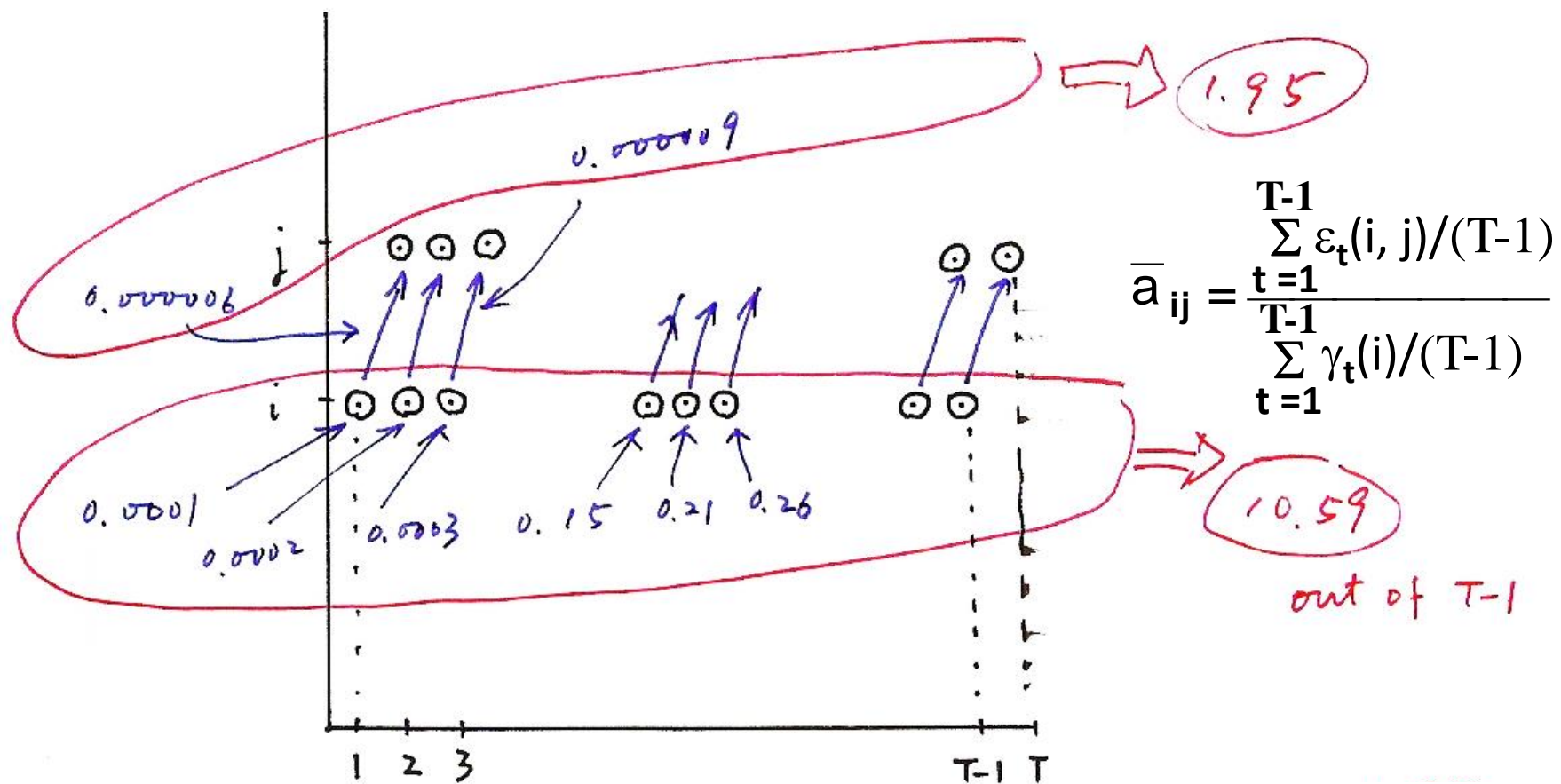


Basic Problem 3

$$\gamma_t(i) = \frac{\alpha_t(i) \beta_t(i)}{\sum_{i=1}^N [\alpha_t(i) \beta_t(i)]} = \frac{P(\bar{O}, q_t = i | \lambda)}{P(\bar{O} | \lambda)} = P(q_t = i | \bar{O}, \lambda)$$

$$\begin{aligned} \varepsilon_t(i, j) &= \frac{\alpha_t(i) \textcolor{red}{a}_{ij} \textcolor{red}{b}_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{j=1}^{\textcolor{red}{N}} \sum_{i=1}^N \alpha_t(i) \textcolor{red}{a}_{ij} \textcolor{red}{b}_j(o_{t+1}) \beta_{t+1}(j)} \\ &= \frac{P(\bar{O}, q_t = i, q_{t+1} = j | \lambda)}{P(\bar{O} | \lambda)} = P(q_t = i, q_{t+1} = j | \bar{O}, \lambda) \end{aligned}$$

Basic Problem 3



$$\bar{a}_{ij} = \frac{1.95/69}{10.59/69}$$

Basic Problem 3 for HMM

- Results

$$\bar{\pi}_i = \gamma_1(i)$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \varepsilon_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\bar{b}_j(k) = \text{Prob}[o_t = v_k \mid q_t = j] = \frac{\sum_{t=1}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)}$$

(for discrete HMM)

• Continuous Density HMM

$$b_j(o) = \sum_{k=1}^M c_{jk} N(o; \mu_{jk}, U_{jk})$$

$N(\cdot)$: Multi-variate Gaussian

μ_{jk} : mean vector for the k-th mixture component

U_{jk} : covariance matrix for the k-th mixture component

$$\sum_{k=1}^M c_{jk} = 1 \text{ for normalization}$$

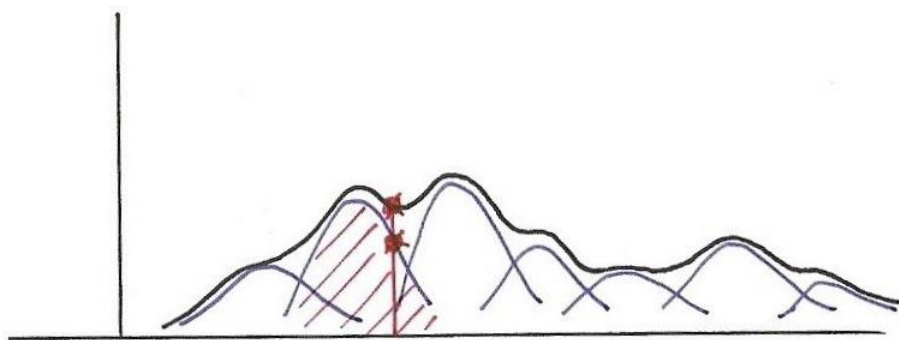
Basic Problem 3 for HMM

- **Continuous Density HMM**

- Define a new variable

$\gamma_t(j, k) = \gamma_t(j)$ but including the probability of o_t evaluated in the k -th mixture component out of all the mixture components

$$= \left(\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^N \alpha_t(j)\beta_t(j)} \right) \left(\frac{c_{jk} N(o_t; \mu_{jk}, U_{jk})}{\sum_{m=1}^M c_{jm} N(o_t; \mu_{jm}, U_{jm})} \right)$$



- Results

$$\bar{c}_{jk} = \frac{\sum_{t=1}^T \gamma_t(j, k)}{\sum_{t=1}^T \sum_{k=1}^M \gamma_t(j, k)}$$

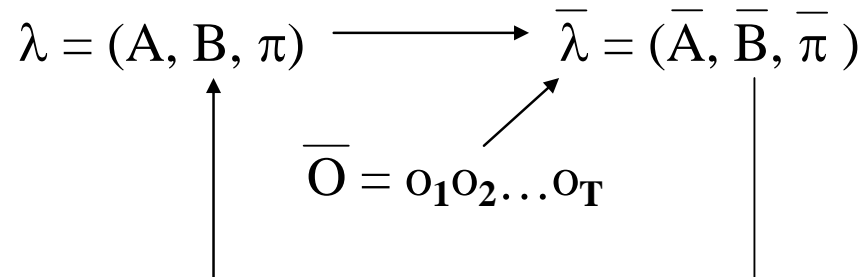
See Fig. 6.9 of Rabiner and Juang

Basic Problem 3 for HMM

- **Continuous Density HMM**

$$\bar{\mu}_{jk} = \frac{\sum_{t=1}^T [\gamma_t(j, k) \cdot o_t]}{\sum_{t=1}^T \gamma_t(j, k)}$$
$$\bar{U}_{jk} = \frac{\sum_{t=1}^T [\gamma_t(j, k)(o_t - \mu_{jk})(o_t - \mu_{jk})']}{\sum_{t=1}^T \gamma_t(j, k)}$$

- **Iterative Procedure**



- It can be shown (by EM Theory (or EM Algorithm))

$P(\bar{O}|\bar{\lambda}) \geq P(\bar{O}|\lambda)$ after each iteration

Basic Problem 3

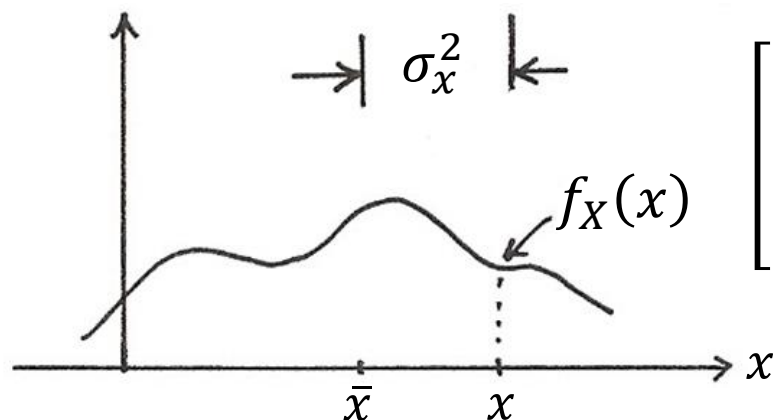
$$\bar{\mu}_{jk} = \frac{\sum_{t=1}^T [\gamma_t(j,k) \cdot o_t]}{\sum_{t=1}^T \gamma_t(j,k)}$$

$$\int_{-\infty}^{\infty} x f_X(x) dx = \bar{x}$$

$$\bar{U}_{jk} = \frac{\sum_{t=1}^T [\gamma_t(j,k) (o_t - \mu_{jk})(o_t - \mu_{jk})']]}{\sum_{t=1}^T \gamma_t(j,k)}$$

$$\int_{-\infty}^{\infty} (x - \bar{x})^2 f_X(x) dx = \sigma_x^2$$

$f_X(x)$: prob. density function



$$\begin{bmatrix} o_{t_1} - \mu_{jk_1} \\ o_{t_2} - \mu_{jk_2} \\ \vdots \\ o_{t_D} - \mu_{jk_D} \end{bmatrix} [o_{t_1} - \mu_{jk_1} \ o_{t_2} - \mu_{jk_2} \ \cdots \ o_{t_D} - \mu_{jk_D}]$$

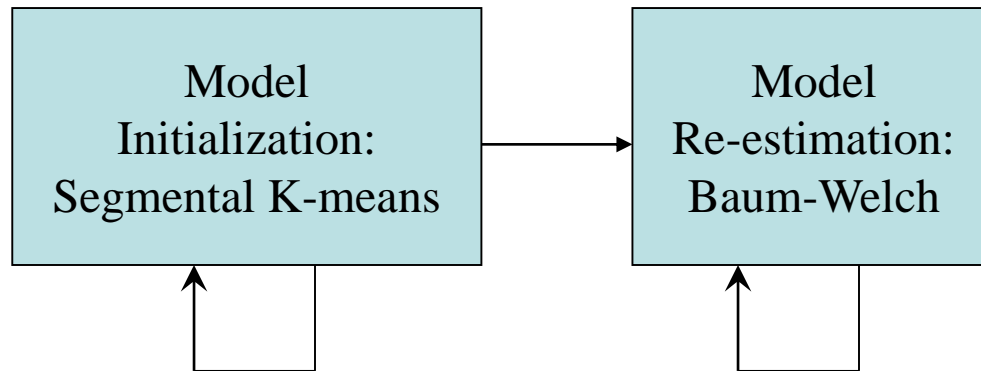
Basic Problem 3

$$\bar{U} = \begin{bmatrix} & m \\ & \vdots \\ & \\ l & \cdots \bar{u}_{lm} \cdots \\ & \vdots \\ & \end{bmatrix} = E \left(\begin{bmatrix} x_1 - \bar{x}_1 \\ x_2 - \bar{x}_2 \\ \vdots \\ \vdots \end{bmatrix} [x_1 - \bar{x}_1, x_2 - \bar{x}_2, \cdots] \right)$$

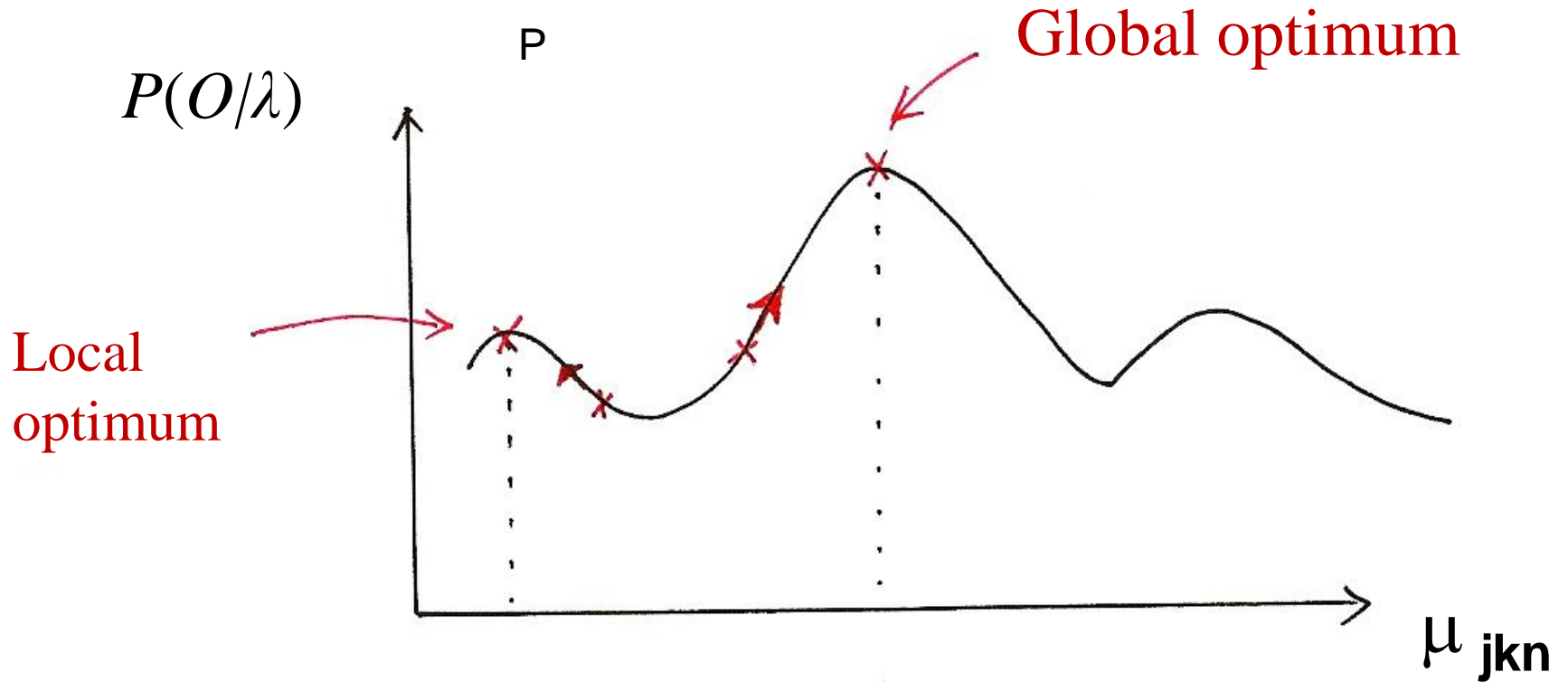
$$\bar{u}_{lm} = E[(x_l - \bar{x}_l)(x_m - \bar{x}_m)]$$

Basic Problem 3 for HMM

- No closed-form solution, but approximated iteratively
- An initial model is needed-model initialization
- May converge to local optimal points rather than global optimal point
 - heavily depending on the initialization
- Model training

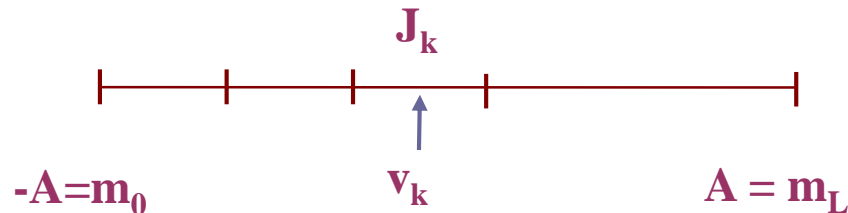


Basic Problem 3



Vector Quantization (VQ)

- **An Efficient Approach for Data Compression**
 - replacing a set of real numbers by a finite number of bits
- **An Efficient Approach for Clustering Large Number of Sample Vectors**
 - grouping sample vectors into clusters, each represented by a single vector (codeword)
- **Scalar Quantization**
 - replacing a single real number by an R-bit pattern
 - a mapping relation



$$S = \bigcup_{k=1}^L J_k, V = \{ v_1, v_2, \dots, v_L \}$$

$$Q : S \rightarrow V$$

$$Q(x[n]) = v_k \text{ if } x[n] \in J_k$$

$$L = 2^R$$

Each v_k represented by an R-bit pattern

– Quantization characteristics (codebook)

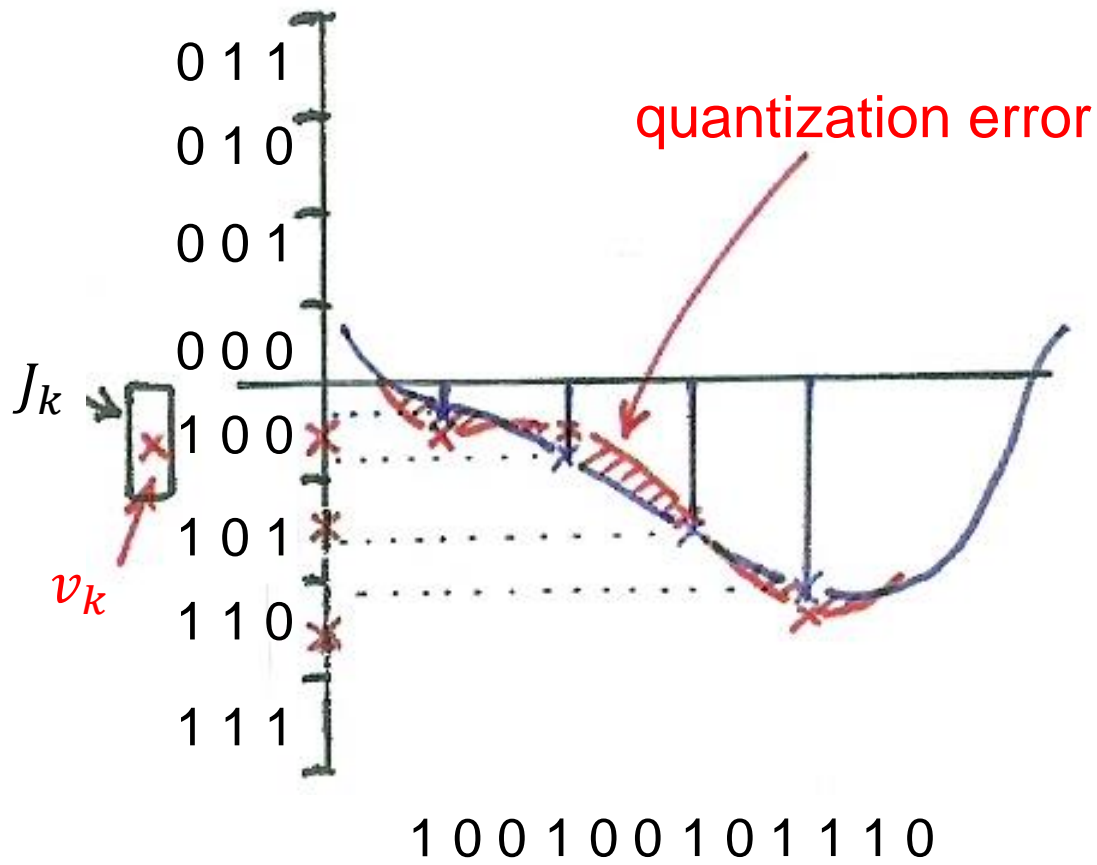
$\{ J_1, J_2, \dots, J_L \}$ and $\{ v_1, v_2, \dots, v_L \}$
designed considering at least

1. error sensitivity

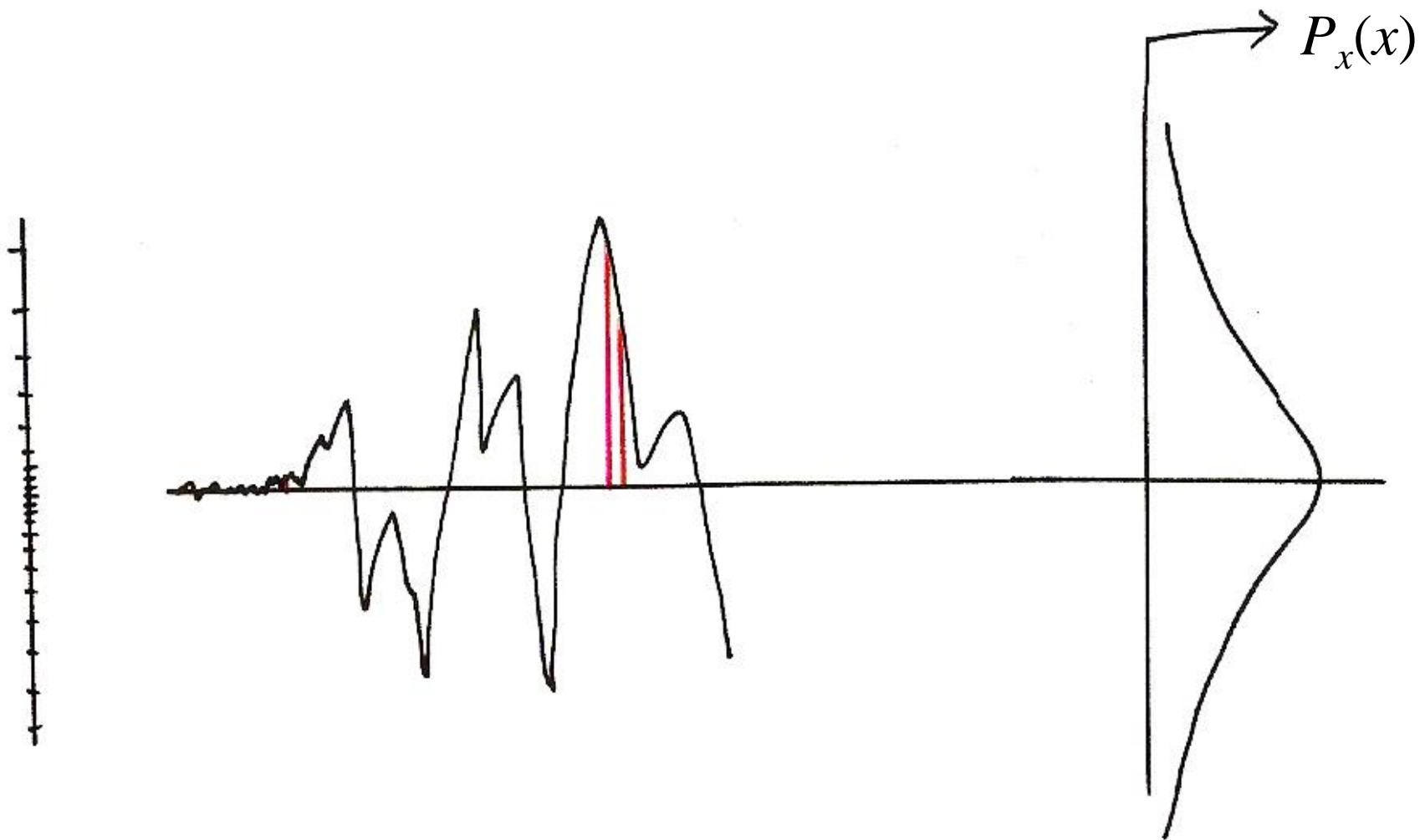
2. probability distribution of $x[n]$

Vector Quantization

Scalar Quantization : Pulse Coded Modulation (PCM)



Vector Quantization



Vector Quantization (VQ)

2-dim Vector Quantization (VQ)

Example:

$$\bar{x}_n = (x[n], x[n+1])$$

$$S = \{ \bar{x}_n = (x[n], x[n+1]) ; |x[n]| < A, |x[n+1]| < A \}$$

•VQ

– S divided into L 2-dim regions

$$\{ J_1, J_2, \dots, J_k, \dots, J_L \}$$

$$S = \bigcup_{k=1}^L J_k$$

each with a representative

$$\text{vector } \bar{v}_k \in J_k, V = \{ \bar{v}_1, \bar{v}_2, \dots, \bar{v}_L \}$$

– $Q : S \rightarrow V$

$$Q(\bar{x}_n) = \bar{v}_k \text{ if } \bar{x}_n \in J_k$$

$$L = 2^R$$

each \bar{v}_k represented by an R-bit pattern

– Considerations

1. error sensitivity may depend on $x[n]$, $x[n+1]$ jointly

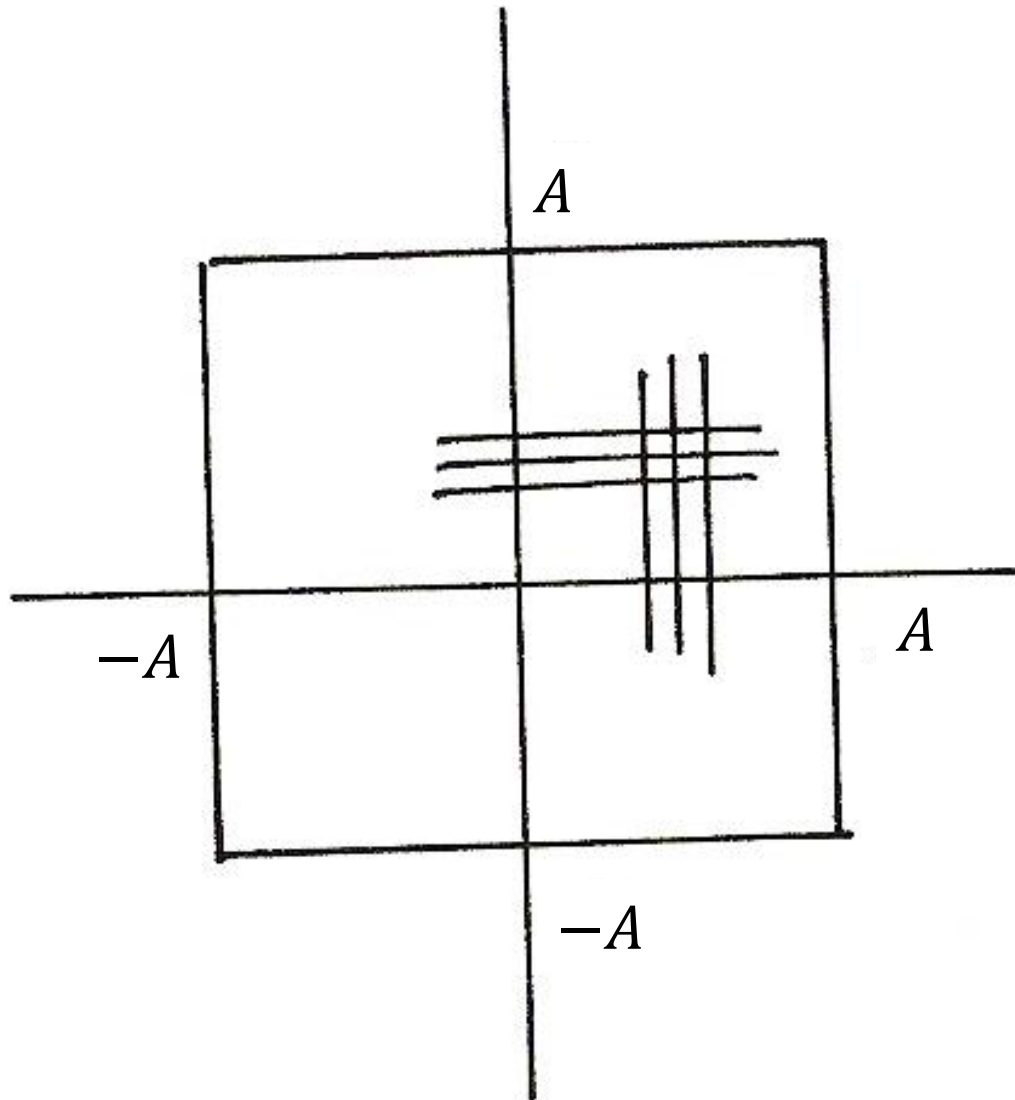
2. distribution of $x[n]$, $x[n+1]$ may be correlated statistically

3. more flexible choice of J_k

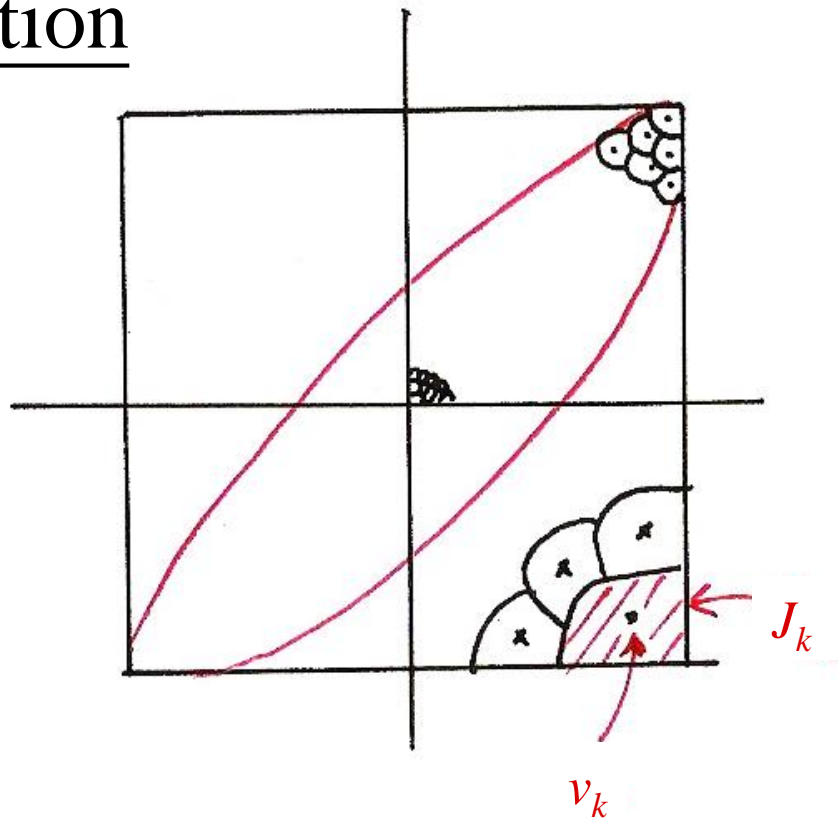
– Quantization Characteristics
(codebook)

$$\{ J_1, J_2, \dots, J_L \} \text{ and } \{ \bar{v}_1, \bar{v}_2, \dots, \bar{v}_L \}$$

Vector Quantization



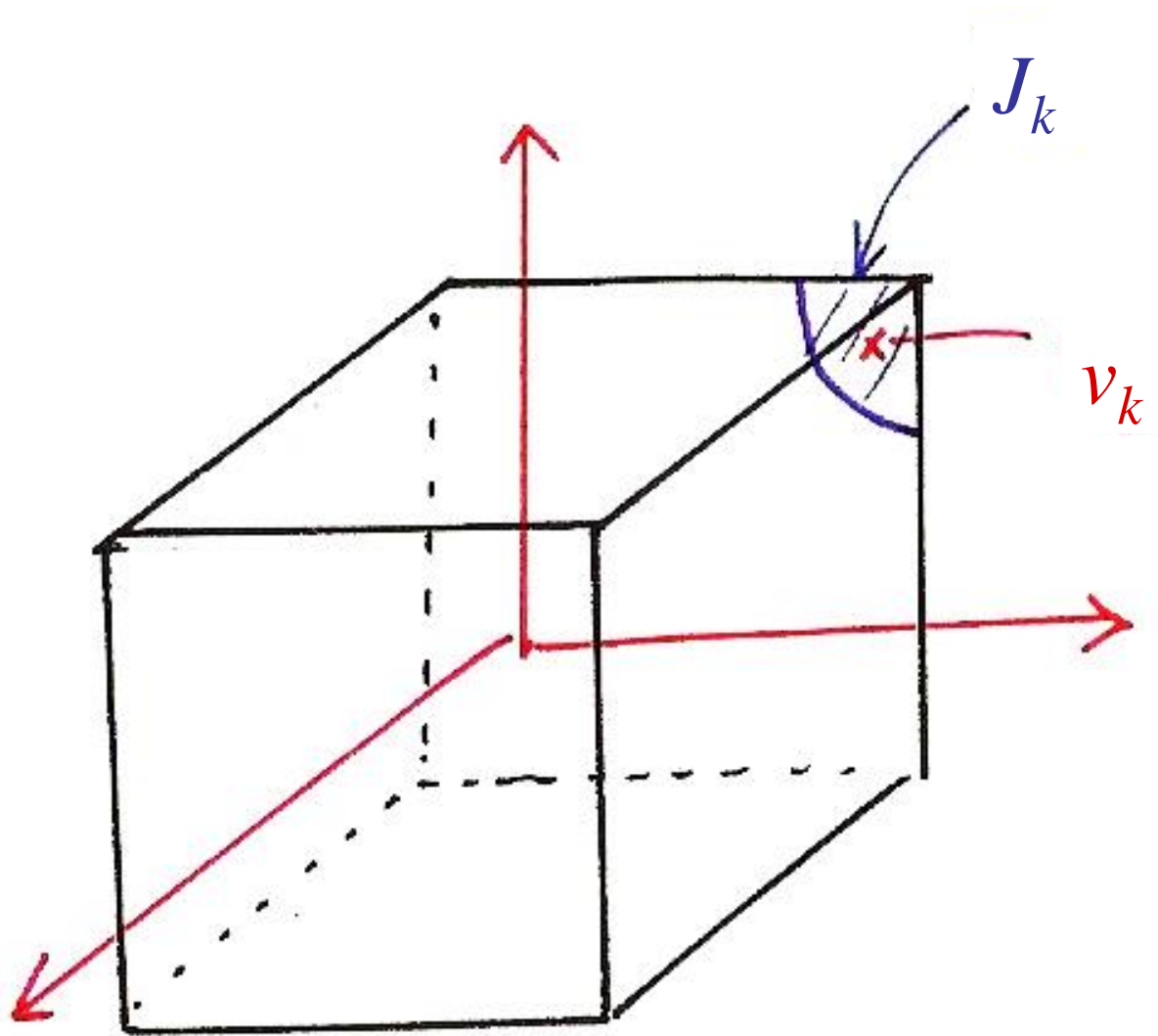
Vector Quantization



$$(256)^2 = (2^8)^2 = 2^{16}$$

$$1024 = 2^{10}$$

Vector Quantization



Vector Quantization (VQ)

N-dim Vector Quantization

$$\bar{x} = (x_1, x_2, \dots, x_N)$$

$$S = \{ \bar{x} = (x_1, x_2, \dots, x_N), \\ |x_k| < A, k = 1, 2, \dots, N \}$$

$$S = \bigcup_{k=1}^L J_k$$

$$V = \{ \bar{v}_1, \bar{v}_2, \dots, \bar{v}_L \}$$

$$Q : S \rightarrow V$$

$$Q(\bar{x}) = \bar{v}_k \text{ if } \bar{x} \in J_k$$

$$L = 2^R, \text{ each } \bar{v}_k \text{ represented} \\ \text{by an R-bit pattern}$$

Codebook Trained by a Large Training Set

Define distance measure between two vectors \bar{x}, \bar{y}

$$d(\bar{x}, \bar{y}) : S \times S \rightarrow \mathbb{R}^+ \text{ (non-negative real numbers)}$$

-desired properties

$$d(\bar{x}, \bar{y}) \geq 0$$

$$d(\bar{x}, \bar{x}) = 0$$

$$d(\bar{x}, \bar{y}) = d(\bar{y}, \bar{x})$$

$$d(\bar{x}, \bar{y}) + d(\bar{y}, \bar{z}) \geq d(\bar{x}, \bar{z})$$

examples :

$$d(\bar{x}, \bar{y}) = \sum_i (x_i - y_i)^2$$

$$d(\bar{x}, \bar{y}) = \sum_i |x_i - y_i|$$

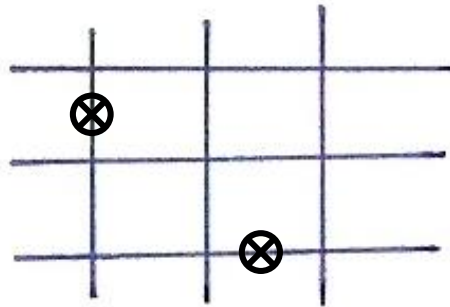
$$d(\bar{x}, \bar{y}) = (\bar{x} - \bar{y})^t \Sigma^{-1} (\bar{x} - \bar{y})$$

Mahalanobis Distance

Σ : Co-variance Matrix

Distance Measures

$$d(\bar{x}, \bar{y}) = \sum_i |x_i - y_i| \quad \text{city block distance}$$



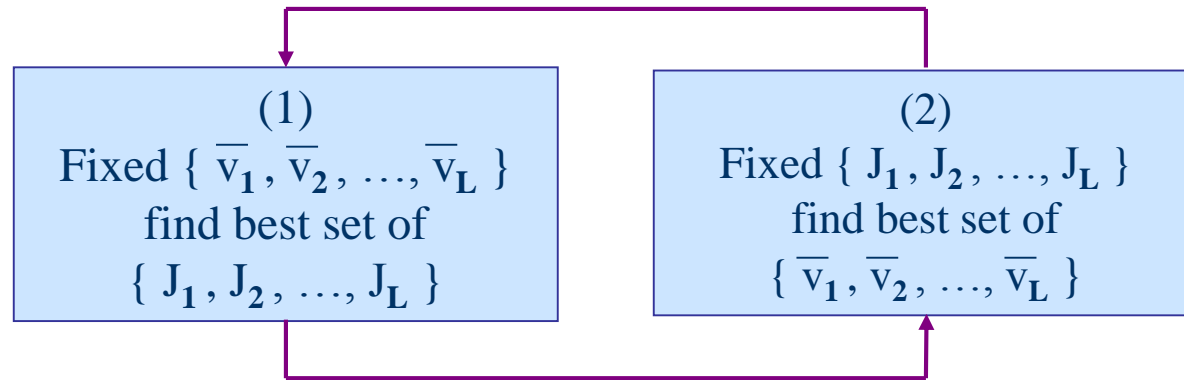
$$d(\bar{x}, \bar{y}) = (\bar{x} - \bar{y})^t \Sigma^{-1} (\bar{x} - \bar{y}) \quad \text{Mahalanobis distance}$$

$$\Sigma = \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix}, d(\bar{x}, \bar{y}) = \sum_i (x_i - y_i)^2$$

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \dots & 0 \\ \vdots & \sigma_2^2 & \vdots \\ 0 & \dots & \ddots & \sigma_n^2 \end{bmatrix}, d(\bar{x}, \bar{y}) = \sum_i \frac{(x_i - y_i)^2}{\sigma_i^2}$$

Vector Quantization (VQ)

- **K-Means Algorithm/Lloyd-Max Algorithm**



(1) $J_k = \{ \bar{x} \mid d(\bar{x}, \bar{v}_k) < d(\bar{x}, \bar{v}_j), j \neq k \}$
 $\rightarrow D = \sum_{\text{all } \bar{x}} d(\bar{x}, Q(\bar{x})) = \min$
nearest neighbor condition

(2) For each k
$$\bar{v}_k = \frac{1}{M} \sum_{\bar{x} \in J_k} \bar{x}$$
$$\rightarrow D_k = \sum_{\bar{x} \in J_k} d(\bar{x}, \bar{v}_k) = \min$$

centroid condition

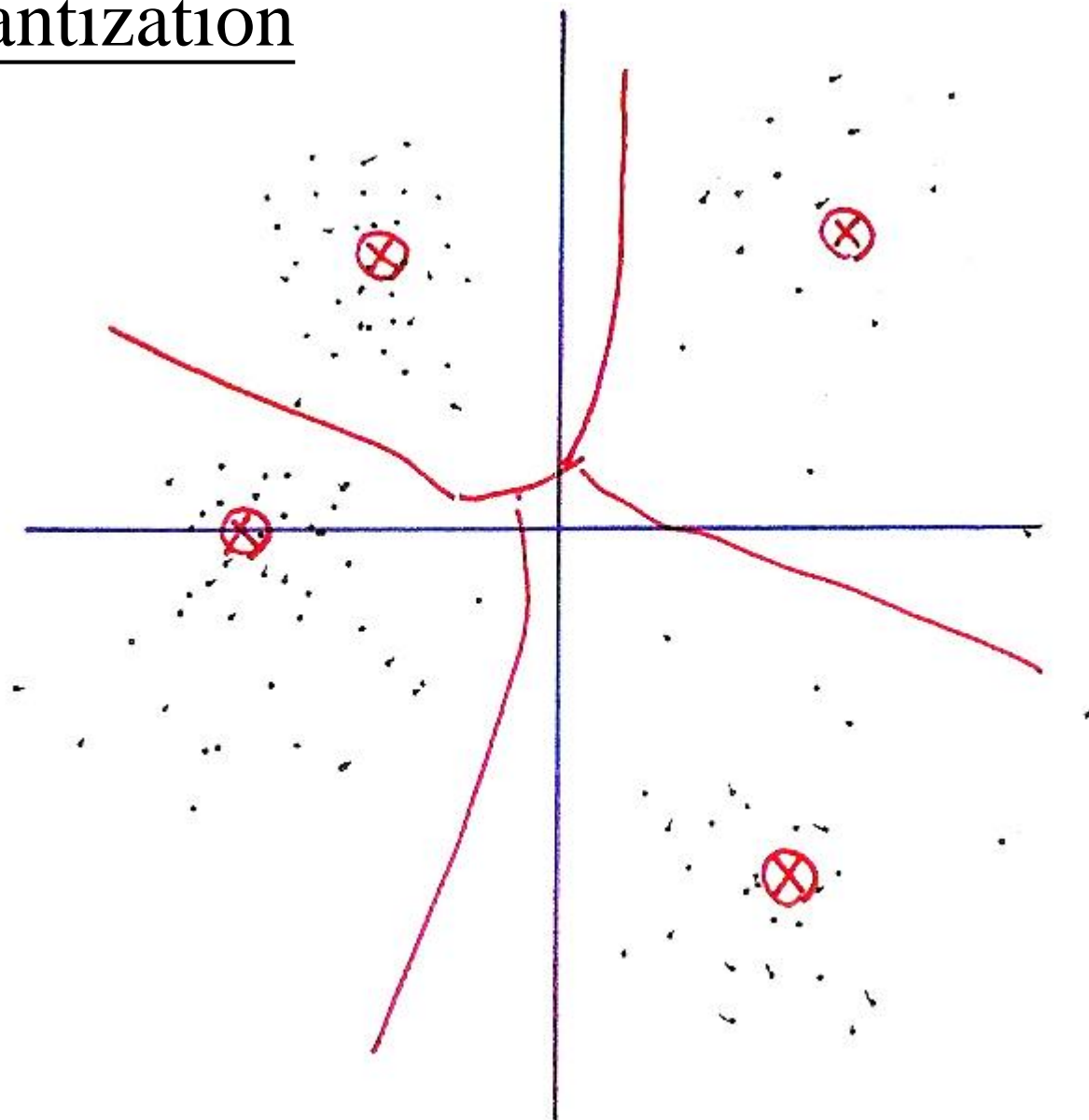
(3) Convergence condition

$$D = \sum_{k=1}^L D_k$$

after each iteration D is reduced, but $D \geq 0$
 $|D^{(m+1)} - D^{(m)}| < \epsilon, m : \text{iteration}$

- **Iterative Procedure to Obtain Codebook from a Large Training Set**

Vector Quantization



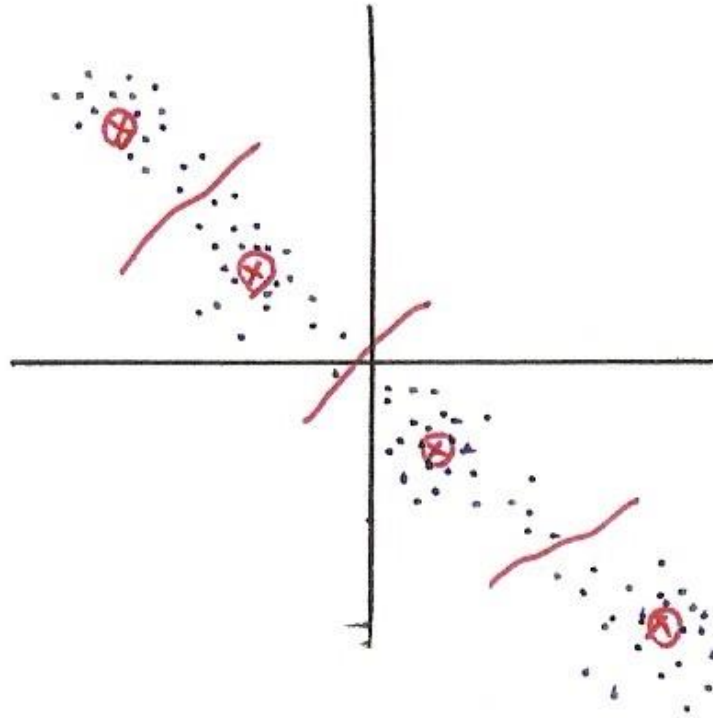
Vector Quantization (VQ)

- **K-means Algorithm may Converge to Local Optimal Solutions**
 - depending on initial conditions, not unique in general
- **Training VQ Codebook in Stages— LBG Algorithm**
 - step 1: Initialization. $L = 1$, train a 1-vector VQ codebook
$$\bar{\mathbf{v}} = \frac{1}{N} \sum_j \bar{\mathbf{x}}_j$$
 - step 2: Splitting.

Splitting the L codewords into $2L$ codewords, $L = 2L$

 - example 1
$$\bar{\mathbf{v}}_k^{(1)} = \bar{\mathbf{v}}_k(1 + \varepsilon)$$
$$\bar{\mathbf{v}}_k^{(2)} = \bar{\mathbf{v}}_k(1 - \varepsilon)$$
 - example 2
$$\bar{\mathbf{v}}_k^{(1)} = \bar{\mathbf{v}}_k$$
$$\bar{\mathbf{v}}_k^{(2)}: \text{the vector most far apart}$$
 - step 3: K-means Algorithm: to obtain L -vector codebook
 - step 4: Termination. Otherwise go to step 2
- **Usually Converges to Better Codebook**

LBG Algorithm



Initialization in HMM Training

- **An Often Used Approach— Segmental K-Means**

- Assume an initial estimate of all model parameters (e.g. estimated by segmentation of training utterances into states with equal length)

- For discrete density HMM

$$b_j(k) = \frac{\text{number of vectors in state } j \text{ associated with codeword } k}{\text{total number of vectors in state } j}$$

- For continuous density HMM (M Gaussian mixtures per state)

⇒ cluster the observation vectors within each state j into a set of M clusters
(e.g. with vector quantization)

c_{jm} = number of vectors classified in cluster m of state j
divided by number of vectors in state j

μ_{jm} = sample mean of the vectors classified in cluster m of state j

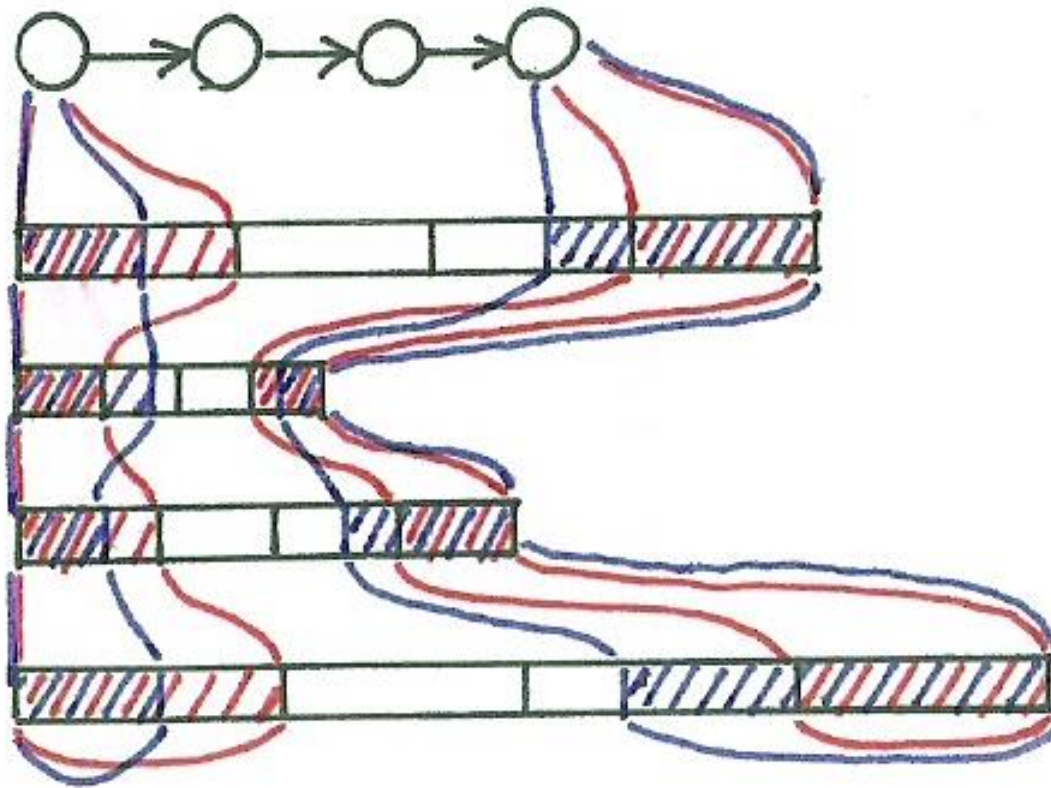
Σ_{jm} = sample covariance matrix of the vectors classified in cluster m of state j

- Step 1 : re-segment the training observation sequences into states based on the initial model by Viterbi Algorithm
- Step 2 : Reestimate the model parameters (same as initial estimation)

- Step 3: Evaluate the model score $P(\bar{O}|\lambda)$:

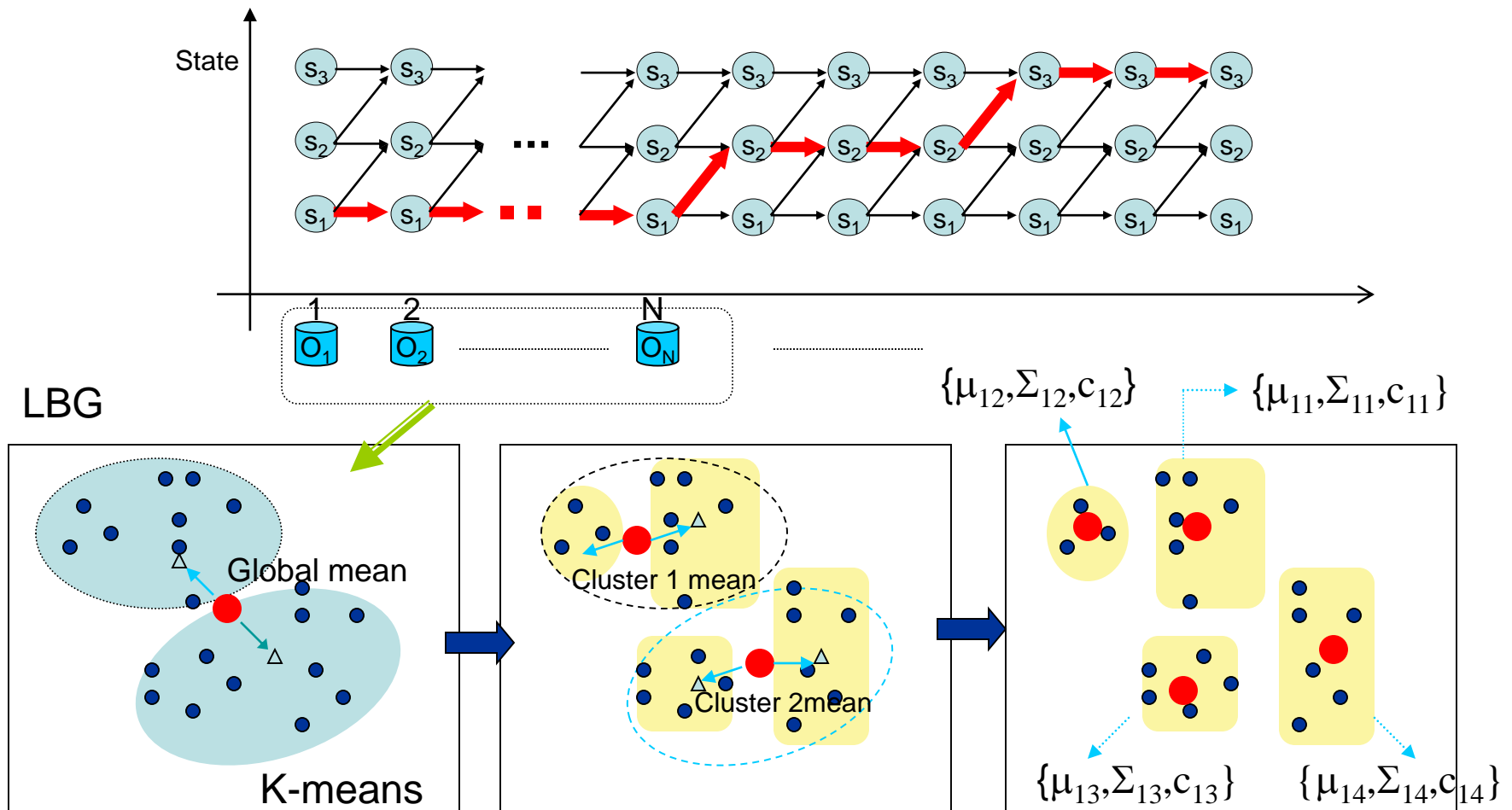
If the difference between the previous and current model scores exceeds a threshold, go back to Step 1, otherwise stop and the initial model is obtained

Segmental K-Means



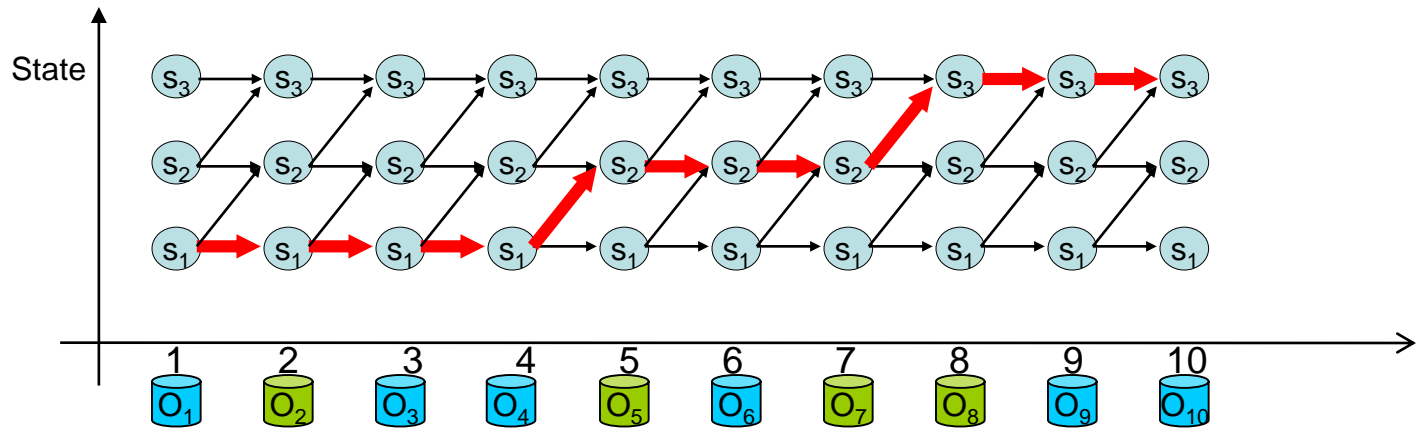
Initialization in HMM Training


- **An example for Continuous HMM**
 - 3 states and 4 Gaussian mixtures per state



Initialization in HMM Training

- An example for discrete HMM
 - 3 states and 2 codewords



\mathbf{v}_1 

\mathbf{v}_2 

$$b_1(\mathbf{v}_1)=3/4, b_1(\mathbf{v}_2)=1/4$$

$$b_2(\mathbf{v}_1)=1/3, b_2(\mathbf{v}_2)=2/3$$

$$b_3(\mathbf{v}_1)=2/3, b_3(\mathbf{v}_2)=1/3$$