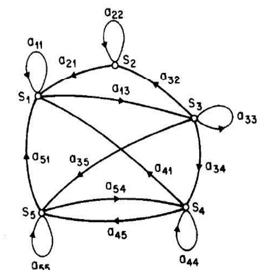
4.0 More about Hidden Markov Models

Reference: 1. 6.1-6.6, Rabiner and Juang

2. 4.4.1 of Huang

Markov Model

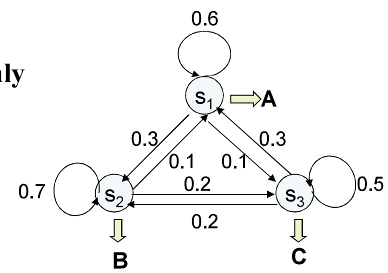
- **Markov Model (Markov Chain)**
  - First-order Markov chain of N states is a triplet (S,**A**,$\boldsymbol{\pi}$)
    - S is a set of N states
    - **A** is the $N \times N$ matrix of state transition probabilities
      $P(q_t=j|q_{t-1}=i, q_{t-2}=k, \ldots\ldots)=P(q_t=j|q_{t-1}=i) \equiv \mathbf{a}_{ij}$
    - $\boldsymbol{\pi}$ is the vector of initial state probabilities
      $\boldsymbol{\pi}_j = P(q_0=j)$
  - 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**



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

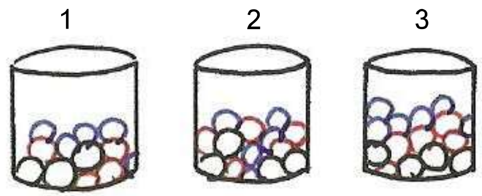
$$\pi = \begin{bmatrix} 0.4 & 0.5 & 0.1 \end{bmatrix}$$

  - Given a sequence of observed symbols **O**={CABBCABC}, the **only one** corresponding state sequence is $\{S_3S_1S_2S_2S_3S_1S_2S_3\}$, and the corresponding probability is
    $P(\mathbf{O}|\lambda)=P(q_0=S_3)$
    $P(S_1|S_3)P(S_2|S_1)P(S_2|S_2)P(S_3|S_2)P(S_1|S_3)P(S_2|S_1)P(S_3|S_2)$
    $=0.1\times0.3\times0.3\times0.7\times0.2\times0.3\times0.3\times0.2=0.00002268$

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,$\boldsymbol{\pi}$}**
  - 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
  - $\boldsymbol{\pi}$ 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
    $\mathbf{V}=\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \ldots\ldots, \mathbf{v}_M\}$, $\mathbf{v}_k \in \boldsymbol{R}^D$
  - the set of observation probability distributions $B=\{b_j(\mathbf{v}_k)\}$ is defined as
    $b_j(\mathbf{v}_k)=P(\mathbf{o}_t=\mathbf{v}_k|\boldsymbol{q}_t=j)$, $1\le k\le M$, $1\le j\le N$
    $\mathbf{o}_t$: *observation at time t,* $\boldsymbol{q}_t$: *state at time t*
    ⇨ *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, $\mathbf{V}=\{\mathbf{v}|\ \mathbf{v}\in \boldsymbol{R}^D\}$
  - the set of observation probability distributions $B=\{b_j(\mathbf{v})\}$ is defined as
    $b_j(\mathbf{v})=P(\mathbf{o}_t=\mathbf{v}|\boldsymbol{q}_t=j)$, $1\le j\le N$
    ⇨ $b_j(\mathbf{v})$ is a ***continuous probability density function and is often assumed to be a mixture of Gaussian distributions***

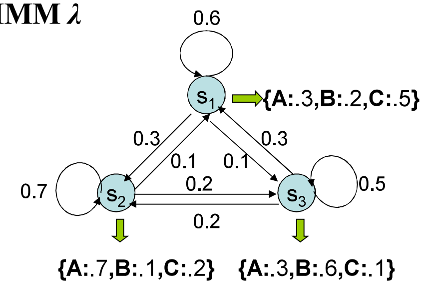
$$b_j(\mathbf{v})=\sum_{k=1}^{M} c_{jk}\left(\frac{1}{\left(\sqrt{2\pi}\right)^D \left|\Sigma_{jk}\right|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(\left(\mathbf{v}-\boldsymbol{\mu}_{jk}\right)^t \Sigma_{jk}^{-1}\left(\mathbf{v}-\boldsymbol{\mu}_{jk}\right)\right)\right)\right)=\sum_{k=1}^{M} c_{jk}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

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 $\overline{\mathrm{O}}=\{\mathrm{ABC}\}$, there are **27 possible** corresponding state sequences, and therefore the corresponding probability is

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

$$e.g.\ \text{when } \mathbf{q}_i=\{S_2 S_2 S_3\}, P(\overline{\mathbf{O}}|\mathbf{q}_i,\lambda)=P(\mathbf{A}|S_2)P(\mathbf{B}|S_2)P(\mathbf{C}|S_3)=0.7*0.1*0.1=0.007$$

$$P(\mathbf{q}_i|\lambda)=P(q_0=S_2)P(S_2|S_2)P(S_3|S_2)=0.5*0.7*0.2=0.07$$

Hidden Markov Model

- **Three Basic Problems for HMMs**

  Given an observation sequence $\overline{\mathrm{O}}=(o_1,o_2,\ldots..,o_T)$, and an HMM λ =(A,B,π)

  - Problem *1* :
    How to *efficiently* compute $P(\overline{\mathbf{O}}|\ \lambda)$?
    ⇨ *Evaluation problem*
  - Problem *2* :
    How to choose an optimal state sequence $\mathbf{q}=(q_1,q_2,\ldots\ldots, q_T)$?
    ⇨ *Decoding Problem*
  - Problem *3* :
    Given some observations $\overline{\mathrm{O}}$ for the HMM λ , how to adjust the model parameter $\boldsymbol{\lambda}$ **=(A,B,$\boldsymbol{\pi}$)** to maximize $P(\overline{\mathbf{O}}|\ \lambda)$?
    ⇨ *Learning /Training Problem*

Basic Problem 1 for HMM

$1 \rightarrow 2 \rightarrow \cdots\cdots\cdots \rightarrow N$ $\quad \lambda = (A, B, \pi)$

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

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

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

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

$$P(\bar{O}|\lambda)=\sum_{\text{all }\bar{q}}\Big(\overbrace{[b_{q_1}(o_1)\bullet b_{q_2}(o_2)\bullet\ldots\ldots b_{q_T}(o_T)]}^{P(\bar{O}|\bar{q},\lambda)}\bullet \underbrace{[\pi_{q_1}\bullet a_{q_1q_2}\bullet a_{q_2q_3}\bullet\ldots\ldots a_{q_{T-1}q_T}]}_{P(\bar{q}|\lambda)}\Big)$$

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

huge computation requirements

Basic Problem 1 for HMM

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

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

=Prob[observing $o_1 o_2 \ldots o_t$, state i at time t|λ]

- Initialization

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

- Induction

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

  $1 \le j \le N$

  $1 \le t \le T-1$

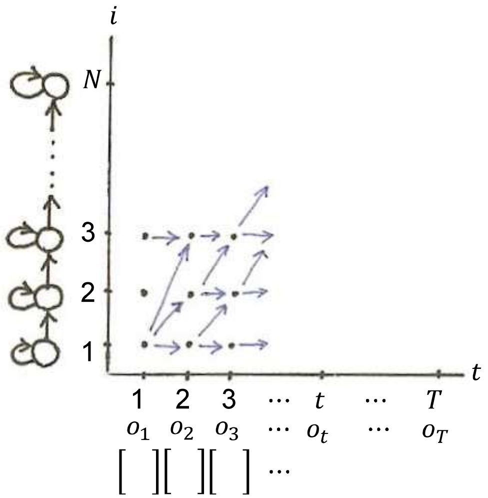
- Termination

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

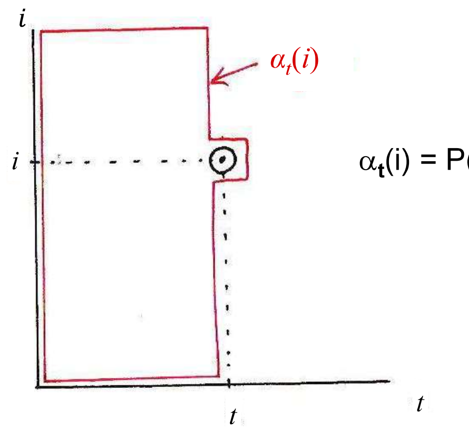
  See Fig. 6.5 of Rabiner and Juang

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

Basic Problem 1

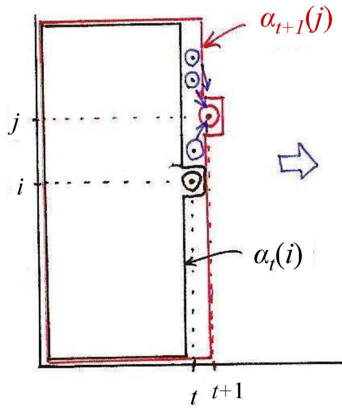


Basic Problem 1



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

Basic Problem 1



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

$$1 \le j \le N$$

$$1 \le t \le T-1$$

Forward Algorithm

Basic Problem 2 for HMM

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

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

$= \text{Prob}[\text{observing } o_{t+1}, o_{t+2}, \ldots, o_T | \text{state } i \text{ at time } t, \lambda]$

- Initialization

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

- Induction

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

  $t = T-1, T-2, \ldots, 2, 1, \qquad 1 \le i \le N$

  See Fig. 6.6 of Rabiner and Juang

- **Combining Forward/Backward Variables**

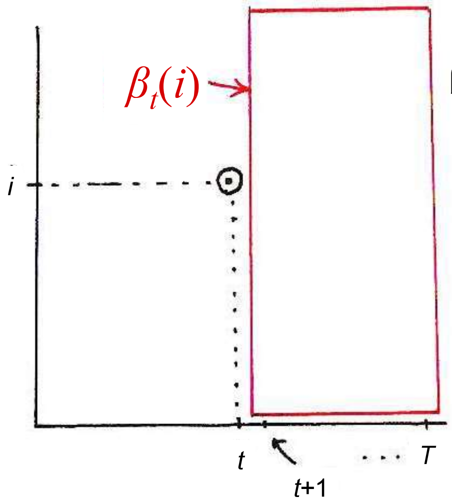
$P(\overline{O}, q_t = i | \lambda)$

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

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

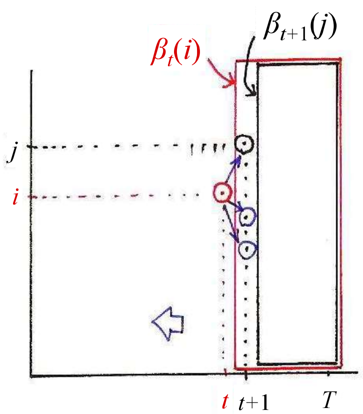
$P(\overline{O}|\lambda) = \sum_{i=1}^{N} P(\overline{O}, q_t = i | \lambda) = \sum_{i=1}^{N} [\alpha_t(i)\beta_t(i)]$

Basic Problem 2



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

Basic Problem 2

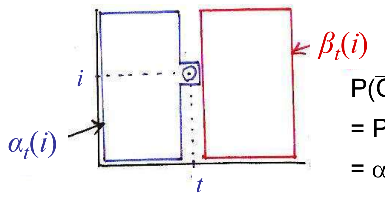


$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(o_{t+1})\beta_{t+1}(j)$$

$t = T-1, T-2, \ldots, 2, 1, \qquad 1 \le i \le N$

Backward Algorithm

Basic Problem 2



$P(\bar{O}, q_t = i \mid \lambda)$
$= \text{Prob [observing } o_1, o_2, \ldots, o_t, \ldots, o_T, q_t = i \mid \lambda]$
$= \alpha_t(i)\beta_t(i)$

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

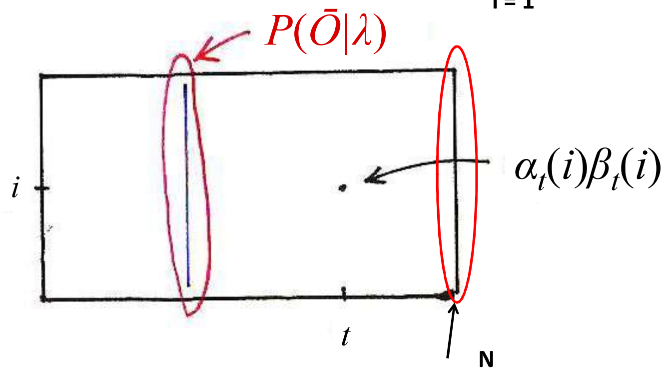
$B: (o_{t+1}, o_{t+2}, \cdots o_T \mid \lambda)$

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

$P(A,B,C) = P(A,C)\ P(B \mid A,C)$
$/\!/ \qquad /\!/ \qquad /\!/ \qquad (B \perp A)$
$P(\bar{O}, q_t = i \mid \lambda) \quad \alpha_t(i) \quad P(B \mid C)$
$/\!/$
$\beta_t(i)$

Basic Problem 2

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



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

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

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

    in fact

    $q_t^* = \arg\max_{1 \le i \le N} [P(\bar{O}, q_t = i \mid \lambda)]$

    $= \arg\max_{1 \le i \le N} [\alpha_t(i)\beta_t(i)]$

  - Problem

    maximizing the probability at each time t individually

    $\bar{q}^* = q_1^* q_2^* \ldots q_T^*$ 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 $\bar{q}^* = q_1^* q_2^* \ldots q_T^*$**
  - Define a new variable $\delta_t(i)$

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

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

  - Induction

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

  - Backtracking

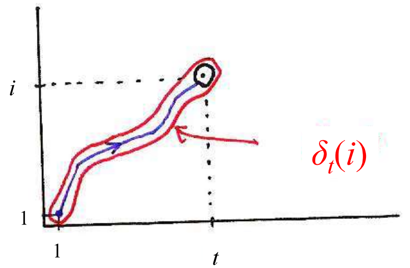
    $\psi_{t+1}(j) = \arg\max_{1 \le i \le N} [\delta_t(i)a_{ij}]$

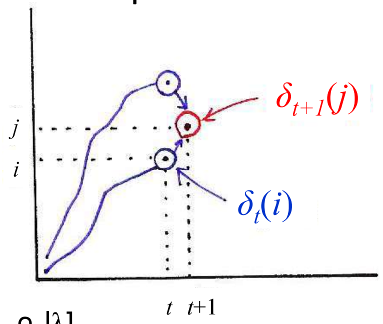
    the best previous state at t−1 given at state j at time t

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

Viterbi Algorithm

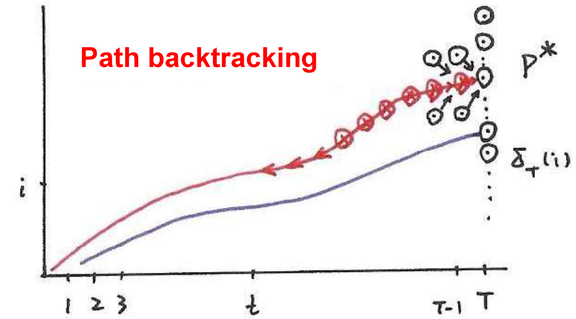
$$\delta_{t+1}(j) = \max_{i} [\delta_t(i)a_{ij}] \bullet b_j(o_{t+1})$$





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

Viterbi Algorithm



Basic Problem 2 for HMM

- **Complete Procedure for Viterbi Algorithm**
  - Initialization

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

  - Recursion

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

    $1 \le t \le T\text{-}1, \quad 1 \le j \le N$

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

    $1 \le t \le T\text{-}1, \quad 1 \le j \le N$

  - Termination

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

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

  - Path backtracking

    $q_t^* = \psi_{t+1}(q^*_{t+1})\,, \quad t = T-1, t-2, \ldots..2, 1$

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

$$\overline{O} = (o_1, o_2, \ldots o_T)$$

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

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

$$\varepsilon_t(i,j) = P(q_t = i, q_{t+1} = j \mid \overline{O}, \lambda)$$

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

$$= \frac{\text{Prob}[\overline{O}, q_t = i, q_{t+1} = j|\lambda]}{P(\overline{O}|\lambda)}$$

See Fig. 6.7 of Rabiner and Juang

  - Recall $\gamma_t(i) = P(q_t = i \mid \overline{O}, \lambda)$

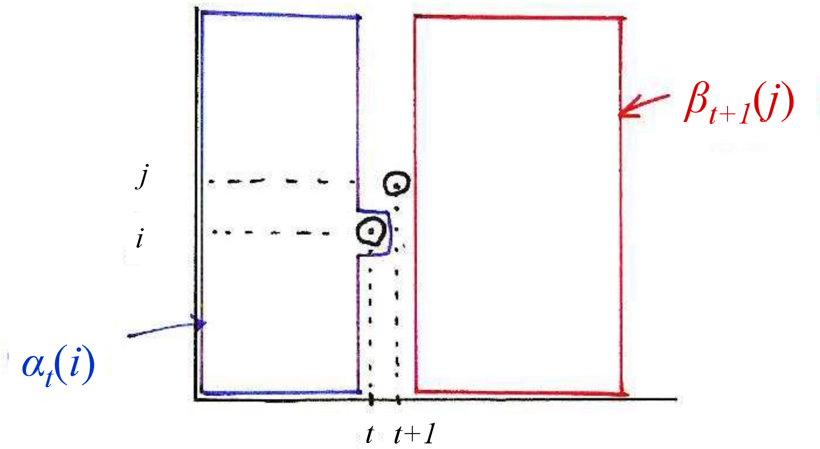
$\sum_{t=1}^{T-1} \gamma_t(i)$ = expected number of times that state i is visited in $\overline{O}$ from t = 1 to t = T−1

= expected number of transitions from state i in $\overline{O}$

$\sum_{t=1}^{T-1} \varepsilon_t(i,j)$ = expected number of transitions from state i to state j in $\overline{O}$

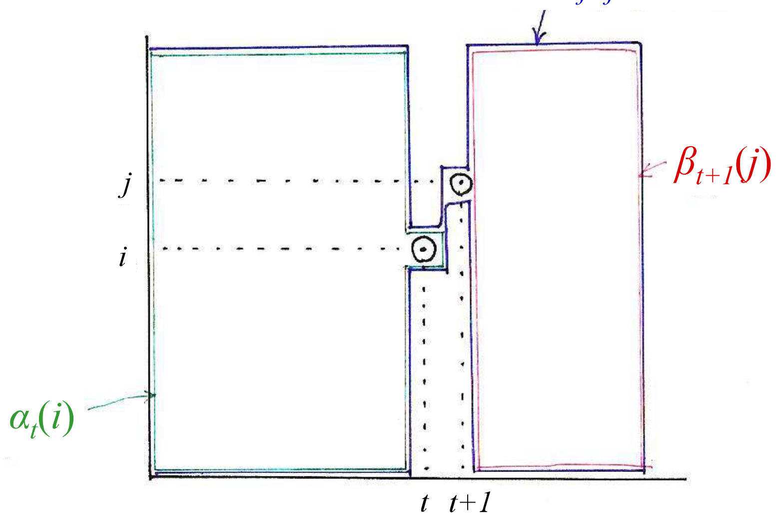
Basic Problem 3

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



Basic Problem 3

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



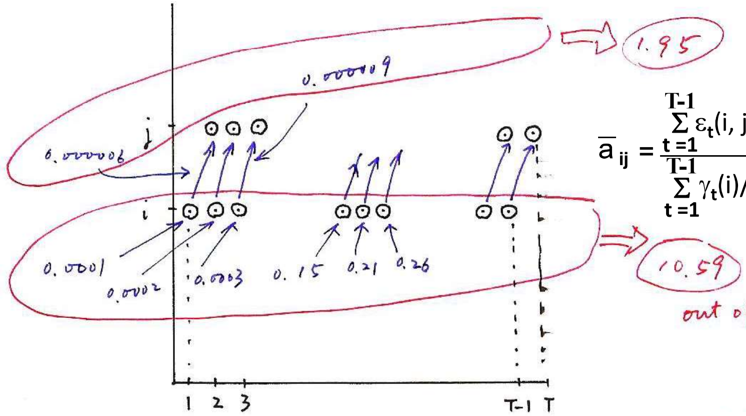
Basic Problem 3

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

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

$$= \frac{P(\bar{O}, q_t = i, q_{t+1} = j|\lambda)}{P(\bar{O}|\lambda)} = P(q_t = i, q_{t+1} = j|\bar{O},\lambda)$$

Basic Problem 3



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

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

Basic Problem 3 for HMM

- Results

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

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

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

(for discrete HMM)

- **Continuous Density HMM**

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

N( ): Multi-variate Gaussian

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

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

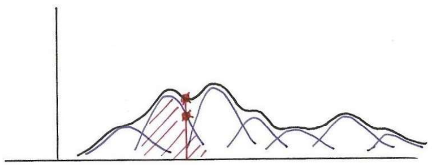
$\sum_{k=1}^{M} c_{jk} = 1$ for normalization

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**

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

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

- **Iterative Procedure**

$$\lambda = (A, B, \pi) \longrightarrow \bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$$

$$\bar{O} = o_1 o_2 \ldots o_T$$

- 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

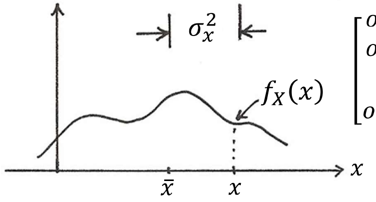
$$\bar{\mu}_{jk} = \frac{\sum_{t=1}^{T}[\boxed{\gamma_t(j,k)} \cdot o_t]}{\boxed{\sum_{t=1}^{T}\gamma_t(j,k)}}$$

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

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

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

$f_X(x)$: prob. density function



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

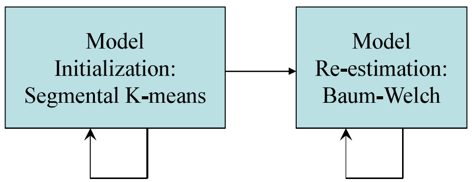
Basic Problem 3

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

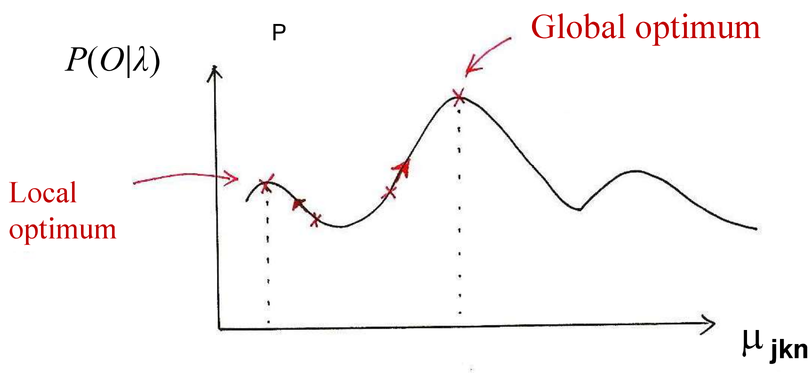
$$\bar{u}_{lm} = E[(x_l-\bar{x}_l)(x_m-\bar{x}_m)]$$

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

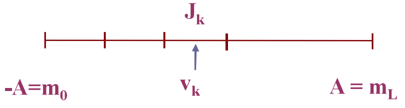


Basic Problem 3



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



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

$Q : S \rightarrow V$

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

$L = 2^R$

Each v_k represented by an R-bit pattern

– Quantization characteristics (codebook)

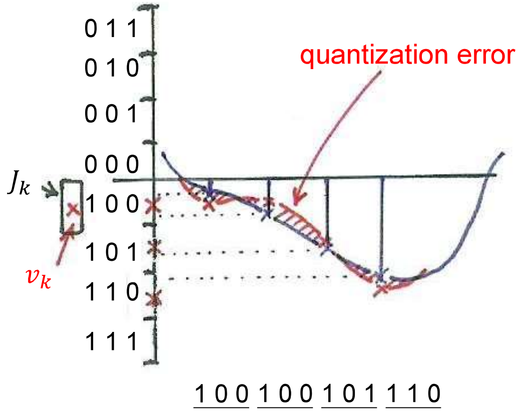
$\{ J_1 , J_2 , \ldots, J_L \}$ and $\{ v_1 , v_2 , \ldots, v_L \}$

designed considering at least

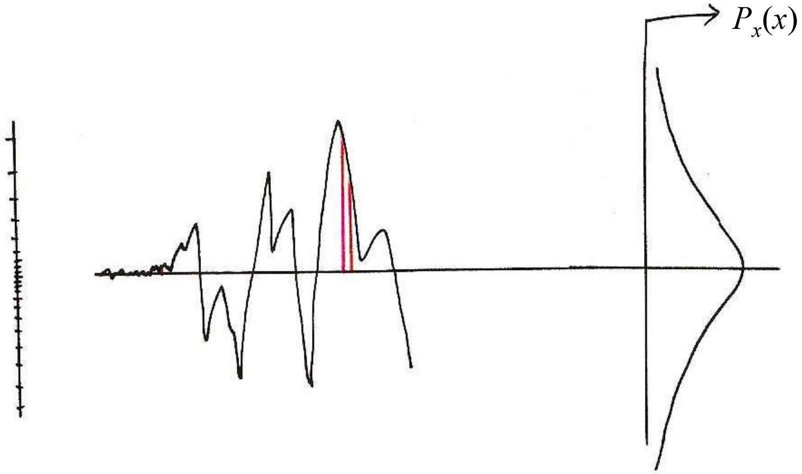
1. error sensitivity
2. probability distribution of x[n]

Vector Quantization

Scalar Quantization ：Pulse Coded Modulation (PCM)



Vector Quantization



Vector Quantization (VQ)

2-dim Vector Quantization (VQ)

Example:

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

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

•VQ

– S divided into L 2-dim regions

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

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

each with a representative vector $\overline{v}_k \in J_k$, $V = \{ \overline{v}_1 , \overline{v}_2 , \ldots, \overline{v}_L \}$

– $Q : S \rightarrow V$

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

$L = 2^R$

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

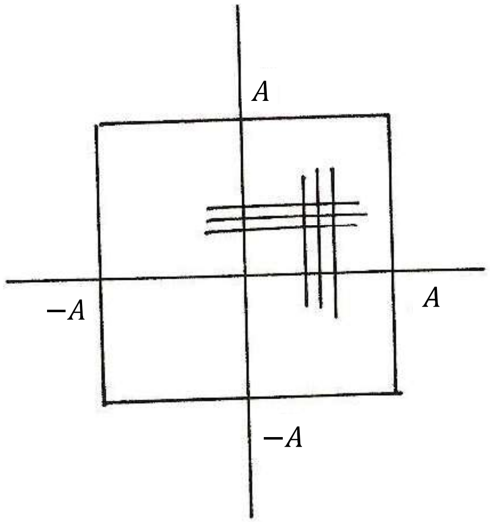
– Considerations

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

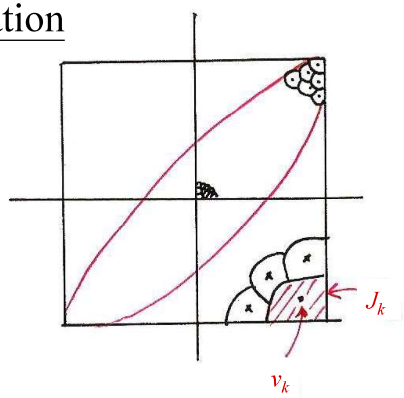
– Quantization Characteristics (codebook)

$\{ J_1 , J_2 , \ldots, J_L \}$ and $\{ \overline{v}_1 , \overline{v}_2 , \ldots, \overline{v}_L \}$

Vector Quantization



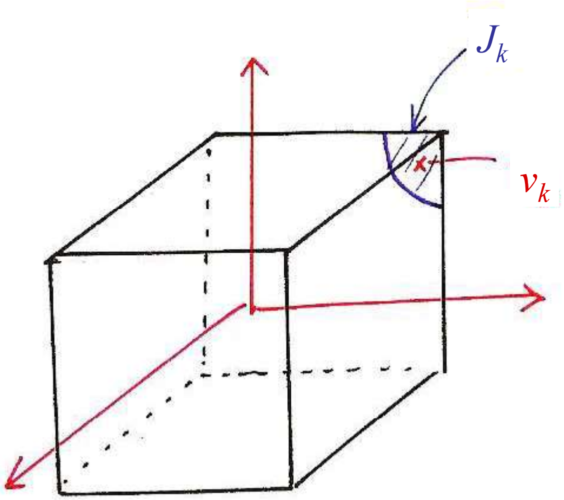
Vector Quantization



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

$$1024=2^{10}$$

Vector Quantization



Vector Quantization (VQ)

N-dim Vector Quantization

$\overline{x} = (x_1, x_2, \ldots, x_N)$

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

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

$V = \{\overline{v}_1, \overline{v}_2, \ldots, \overline{v}_L\}$

$Q : S \rightarrow V$

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

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

Codebook Trained by a Large Training Set

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

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

-desired properties

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

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

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

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

examples :

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

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

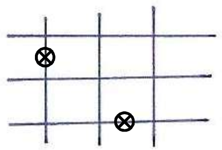
$d(\overline{x}, \overline{y}) = (\overline{x}-\overline{y})^t \Sigma^{-1} (\overline{x}-\overline{y})$

Mahalanobis Distance

Σ: Co-variance Matrix

Distance Measures

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



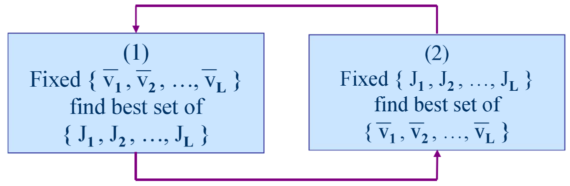
$$d(\bar{x},\bar{y}) = (\bar{x}-\bar{y})^t\, \Sigma^{-1}\, (\bar{x}-\bar{y})$$ Mahalanobis distance

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

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

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

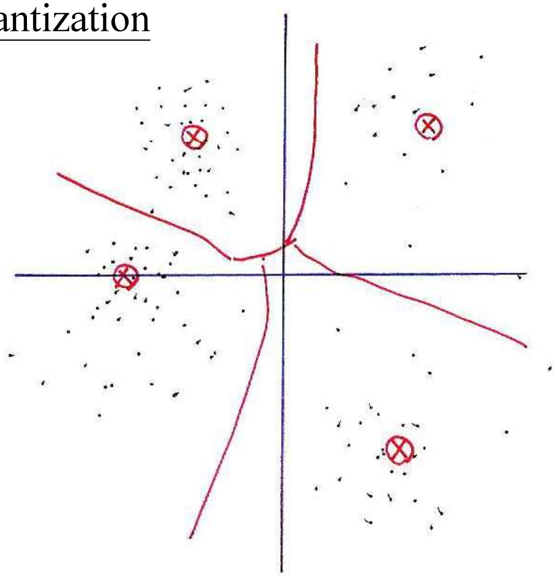
$$D = \sum_{k=1}^{L} D_k$$

after each iteration D is reduced, but $D \geq 0$

$| D^{(m+1)} - D^{(m)} | < \epsilon$, m : 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

$$\bar{v} = \frac{1}{N} \sum_j \bar{x}_j$$

  - step 2: Splitting.

    Splitting the L codewords into 2L codewords, L = 2L
    - example 1

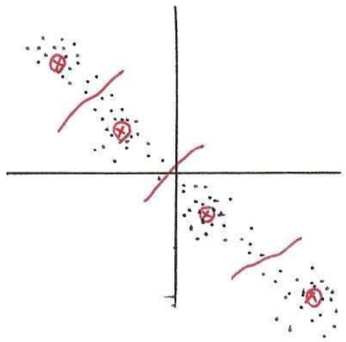
      $\bar{v}_k^{(1)} = \bar{v}_k(1+\varepsilon)$

      $\bar{v}_k^{(2)} = \bar{v}_k(1-\varepsilon)$
    - example 2

      $\bar{v}_k^{(1)} = \bar{v}_k$

      $\bar{v}_k^{(2)}$: 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)

$\Rightarrow$ cluster the observation vectors within each state j into a set of M clusters (e.g. with vector quantiziation)

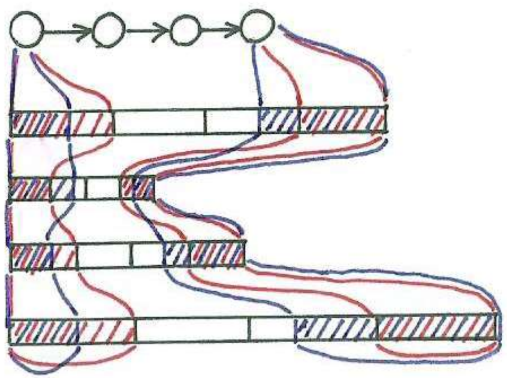
c_{jm} = number of vectors classified in cluster m of state j divided by number of vectors in state j

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

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

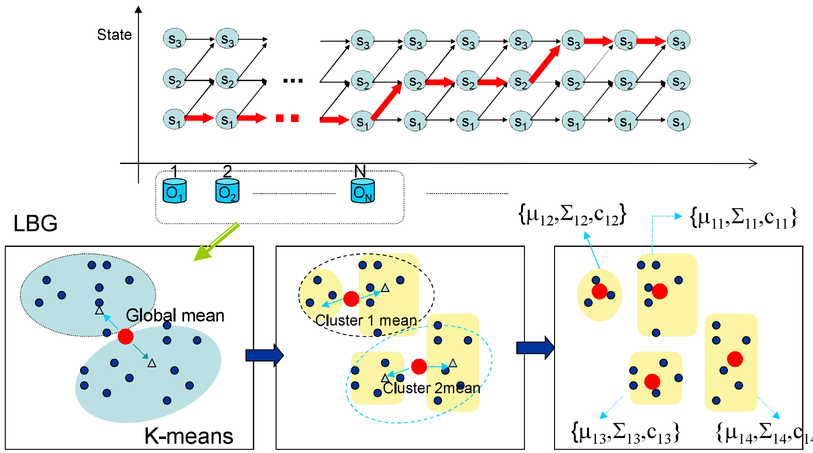
  - Step 1 : re-segment the training observation sequences into states based on the initial model by Viterbi Algorithm
  - Step 2 : Reestimate the model parameters (same as initial estimation)
  - Step 3: Evaluate the model score $P(\overline{O}|\lambda)$:
    If the difference between the previous and current model scores exceeds a threshold, go back to Step 1, otherwise stop and the initial model is obtained

Segmental K-Means



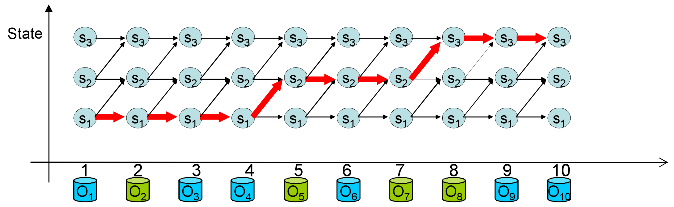
Initialization in HMM Training

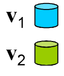
- **An example for Continuous HMM**
  - 3 states and 4 Gaussian mixtures per state



Initialization in HMM Training

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





$b_1(\mathbf{v}_1)$=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