4.0 More about Hidden Markov Models

Reference: 1. 6.1-6.6, Rabiner and Juang

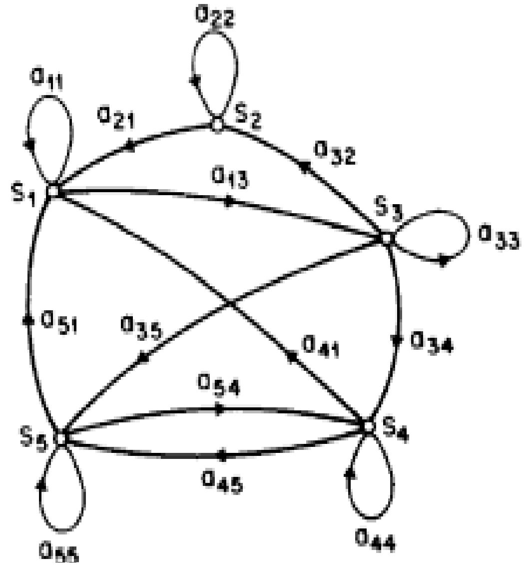
2. 4.4.1 of Huang

Markov Model

- **Markov Model (Markov Chain)**
  - First-order Markov chain of N states is a triplet (S,**A**,π)
    - S is a set of *N* states
    - **A** is the $N \times N$ matrix of state transition probabilities

      $P(q_t=j|q_{t-1}=i, q_{t-2}=k, \ldots\ldots)=P(q_t=j|q_{t-1}=i) \equiv \mathbf{a}_{ij}$
    - π is the vector of initial state probabilities

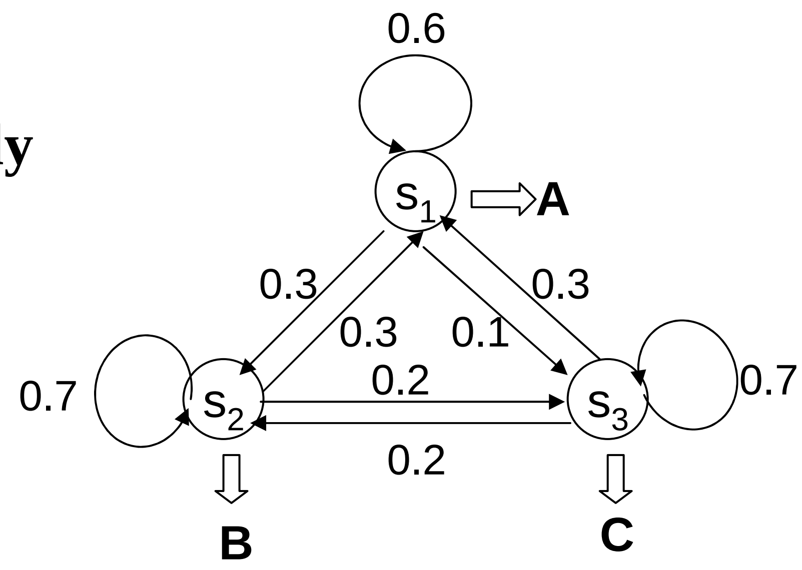
      $\pi_j = P(q_0=j)$
  - The output for any given state is an observable event (deterministic)
  - The output of the process is a sequence of observable events



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**



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

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

  - 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,π}**
  - 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

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***

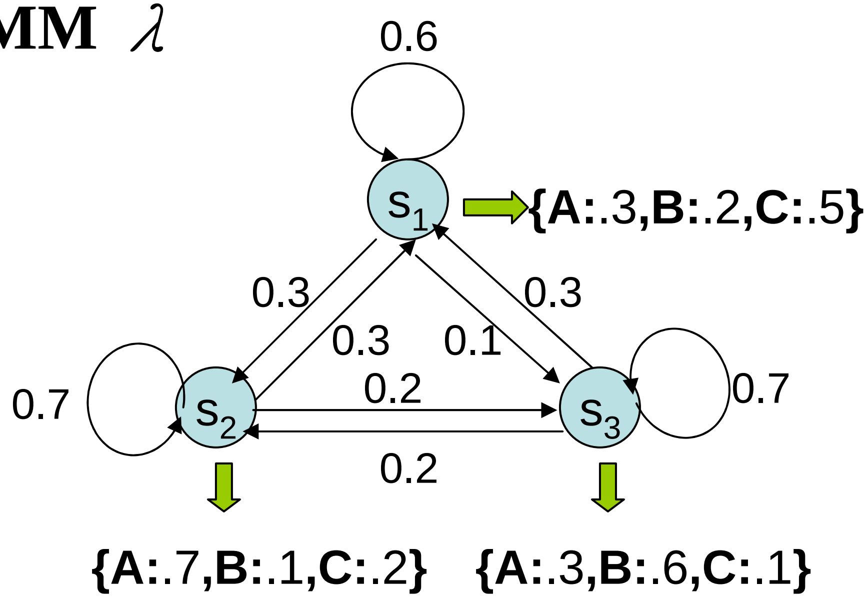
$$b_j(\mathbf{v})=\sum_{k=1}^{M} c_{jk}\left(\frac{1}{\left(\sqrt{2\pi}\right)^D\left|\Sigma_{jk}\right|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(\left(\mathbf{v}-\boldsymbol{\mu}_{jk}\right)^t\Sigma_{jk}^{-1}\left(\mathbf{v}-\boldsymbol{\mu}_{jk}\right)\right)\right)\right)=\sum_{k=1}^{M} c_{jk}b_{jk}(V)$$

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

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

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

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

Hidden Markov Model

- **Three Basic Problems for HMMs**

  Given an observation sequence $O=(o_1,o_2,\ldots,o_T)$**, and an HMM** $\lambda=(A,B,\pi)$

  – Problem *1* :

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

    ⇨ *Evaluation problem*

  – Problem *2* :

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

    ⇨ *Decoding Problem*

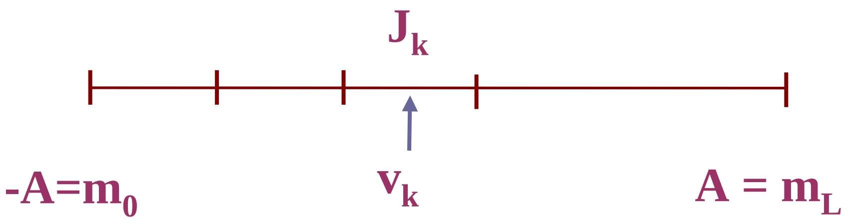
  – Problem *3* :

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

    ⇨ *Learning /Training Problem*

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 (VQ)

2-dim Vector Quantization (VQ)

Example:

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

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

- **VQ**
  - S divided into L 2-dim regions

    $\{ J_{\mathbf{1}} , J_{\mathbf{2}} , \ldots, J_{\mathbf{k}} , \ldots J_{\mathbf{L}} \}$

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

    each with a representative

    vector $\overline{v}_{\mathbf{k}} \in J_{\mathbf{k}}$, $V = \{ \overline{v}_{\mathbf{1}} , \overline{v}_{\mathbf{2}} , \ldots, \overline{v}_{\mathbf{L}} \}$
  - $Q : S \rightarrow V$

    $Q(\overline{x}_{\mathbf{n}}) = \overline{v}_{\mathbf{k}}$ if $\overline{x}_{\mathbf{n}} \in J_{\mathbf{k}}$

    $L = 2^{\mathbf{R}}$

    each $\overline{v}_{\mathbf{k}}$ 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 $J_{\mathbf{k}}$
  - Quantization Characteristics (codebook)

    $\{ J_{\mathbf{1}} , J_{\mathbf{2}} , \ldots, J_{\mathbf{L}} \}$ and $\{ \overline{v}_{\mathbf{1}} , \overline{v}_{\mathbf{2}} , \ldots, \overline{v}_{\mathbf{L}} \}$

Vector Quantization (VQ)

N-dim Vector Quantization

$\bar{x} = (x_{\mathbf{1}}, x_{\mathbf{2}}, \ldots, x_{\mathbf{N}})$

$S = \{\bar{x} = (x_{\mathbf{1}}, x_{\mathbf{2}}, \ldots, x_{\mathbf{N}}),$

$\quad |x_{\mathbf{k}}| < A, k = 1,2,\ldots N\}$

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

$V = \{\bar{v}_{\mathbf{1}}, \bar{v}_{\mathbf{2}}, \ldots, \bar{v}_{\mathbf{L}}\}$

$Q : S \rightarrow V$

$Q(\bar{x}) = \bar{v}_{\mathbf{k}}$ if $\bar{x} \in J_{\mathbf{k}}$

$L = 2^{\mathbf{R}}$, each $\bar{v}_{\mathbf{k}}$ represented by an R-bit pattern

Codebook Trained by a Large Training Set

- **Define distance measure between two vectors $\bar{\mathbf{x}}, \bar{\mathbf{y}}$**

$d(\bar{x}, \bar{y}) : S \times S \rightarrow R^{+}$ (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 :

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

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

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

Mahalanobis Distance

Σ: Co-variance Matrix

Vector Quantization (VQ)

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

(1)
Fixed $\{ \bar{v}_1, \bar{v}_2, \ldots, \bar{v}_L \}$
find best set of
$\{ J_1, J_2, \ldots, J_L \}$

(2)
Fixed $\{ J_1, J_2, \ldots, J_L \}$
find best set of
$\{ \bar{v}_1, \bar{v}_2, \ldots, \bar{v}_L \}$

(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 $D \geq 0$

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

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

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**

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)
  - Step 1 : re-segment the training observation sequences into states based on the initial model by Viterbi Algorithm
  - Step 2 :
    - 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)

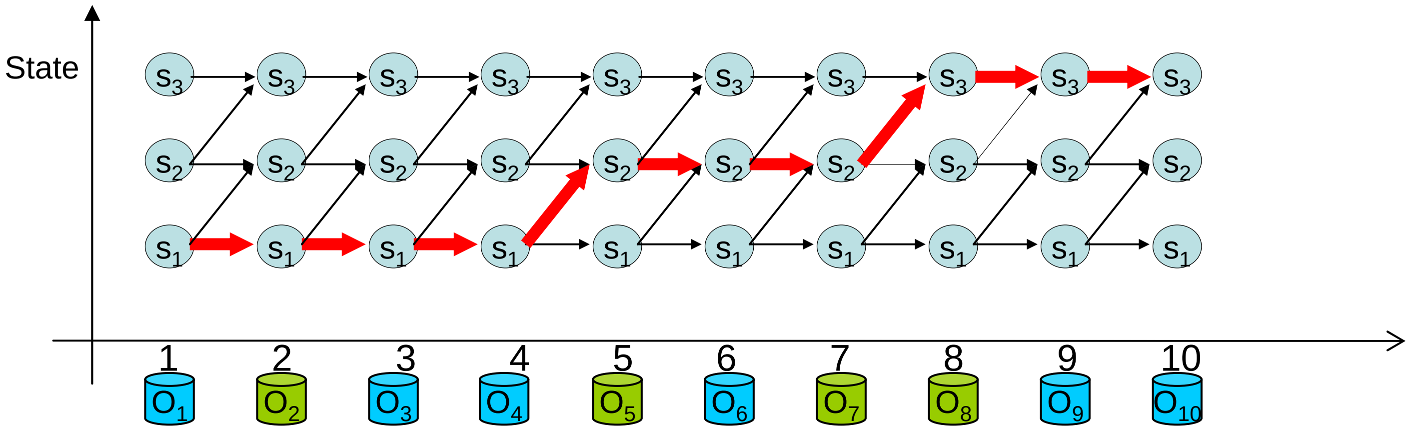
$$\Rightarrow \text{cluster the observation vectors within each state } j \text{ into a set of } M \text{ clusters}$$
$$\text{(e.g. with vector quantiziation)}$$
$$c_{jm} = \text{number of vectors classified in cluster } m \text{ of state } j$$
$$\text{divided by number of vectors in state } j$$
$$\mu_{jm} = \text{sample mean of the vectors classified in cluster } m \text{ of state } j$$
$$\Sigma_{jm} = \text{sample covariance matrix of the vectors classified in cluster } m \text{ of state } j$$

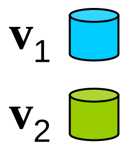
  - Step 3: Evaluate the model score $P(\overline{O}|\lambda)$:

    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

Initialization in HMM Training

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





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

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

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

Initialization in HMM Training

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