Hidden Markov Model

Bor-shen Lin
bslin@cs.ntust.edu.tw
http://www.cs.ntust.edu.tw/~bslin

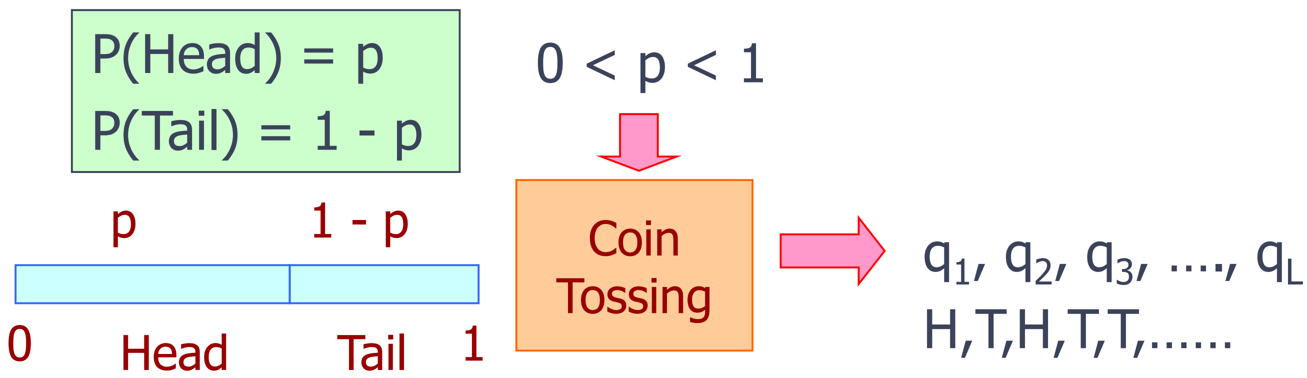
Observations from Random Process

- Observable output generated by real-world random process
- Discrete vs. Continuous
  - Character, gender, class, …
  - Speech, temperature,…

Hidden Markov Model

- A type of *stochastic model*
- First published in mathematic journals
  - Baum and his colleagues, late 1960s
- Applied to speech recognition
  - Baker at CMU/Jelinek at IBM, 1970s
- A general approach
  - Can be applied to many other areas

A Simple Stochastic Process



- The outcomes $q_1, q_2, \ldots, q_T$ can be obtained through random tests
  - q_i has Two states: Head (H) and Tail (T)
- Each tossing is *independent* of the results of previous tossings
  - Random variables $q_1, q_2, \ldots, q_L$ are i.i.d.
- States (H,T): possible outcomes of random variables q_t

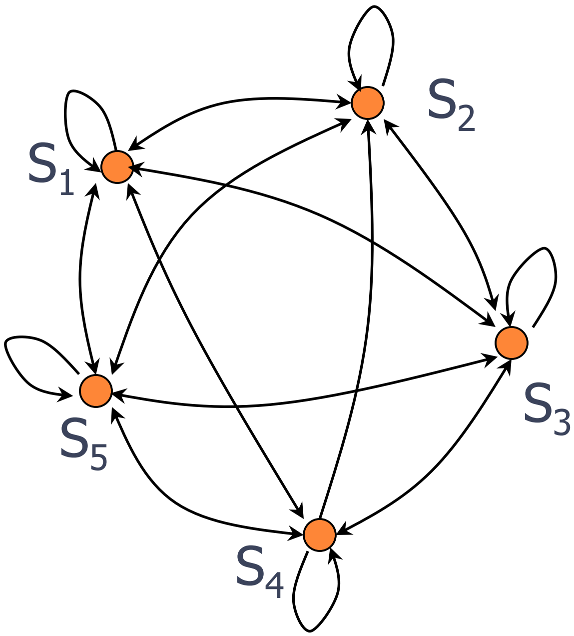
Dice Tossing

- Fair dice: (1/6, 1/6, 1/6, 1/6, 1/6, 1/6) for all faces
- Every tossing is independent of the others
  - Current tossing does not depend on previous results
- Number of states : 6
  - $q_t = S_i$, $i = 1, 2, \ldots, 6$
  - q_t random variables
  - S_i outcome symbols (number of points)

DICE TOSSING (CONT'D)

- 6 dices numbered from 1 to 6
  - With different distributions (probably unfair)
    - Dice 1: (1/3, 1/10, …)
    - Dice 2: (1/4., 1/6, …)
    - …
    - Dice 6: (1/5, 1/8, …)
  - The choice of dice depends on the outcome of previous tossing
    - E.g. if $q_t = S_4$ (4 points) in t_{th} tossing, then choose Dice 4 for $(t+1)_{th}$ tossing
  - q_t depends on q_{t-1}
    - $q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow \dots \rightarrow q_T$

Discrete Markov Process



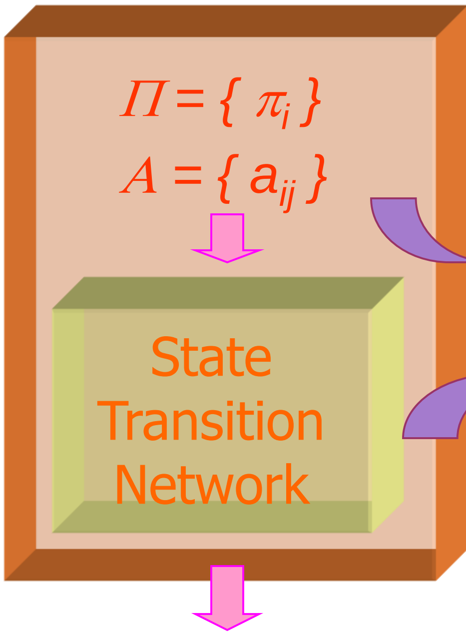
A system with 5 states $S_1 - S_5$.

- System is in one of N distinct states $(S_1, S_2, \ldots, S_N)$ at any time, and *state sequence* $q_1, q_2, \ldots, q_T$ is observed
- First-order Markov Chain
  - $P(q_t=S_j|q_{t-1}=S_i, q_{t-2}=S_k\ldots) = P(q_t=S_j|q_{t-1}=S_i)$
  - $a_{ij} \equiv P(q_t=S_j|q_{t-1}=S_i) \qquad 1 \leq i,j \leq N$

    $a_{ij} > 0, \ \Sigma_{j=1}^{N} a_{ij} = 1$
  - $\pi_i \equiv P(q_1=S_i) \qquad 1 \leq i \leq N$
- State transition depends on *only the previous state*
- State transits *stochastically*
- Imagine: N dices with N faces each

Markov Process Simulator

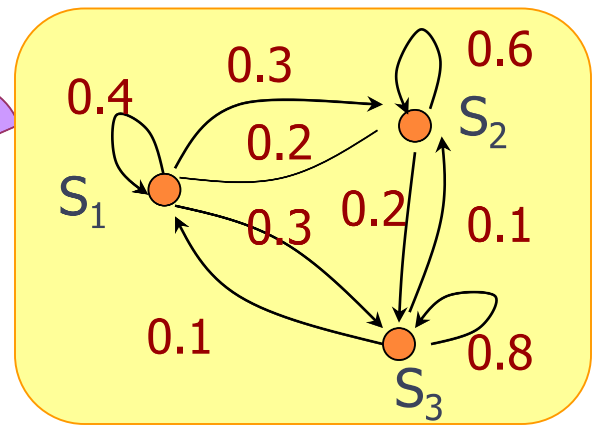
Discrete 1_{st} Order Markov Process



$$\Pi = [\ 0.2 \quad 0.3 \quad 0.5\ ]$$

$$A = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

S_1: rainy
S_2: cloudy
S_3: sunny

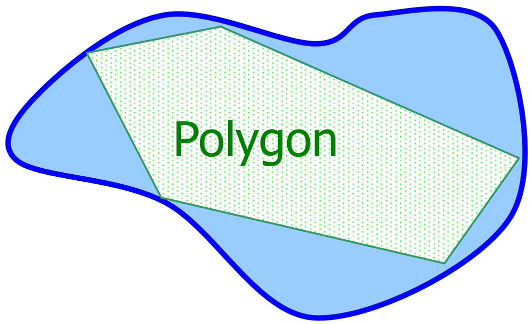


Markov chain $q_1, q_2, \ldots, q_T$

Example: $S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3$
sunny, sunny, sunny, rainy, rainy, sunny, cloudy, sunny,…

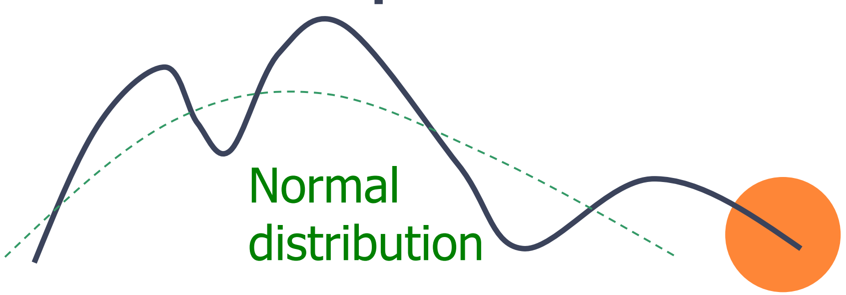
Model vs. Reality

- Model ≠ Reality
  - Does God use Markov Process to generate weather patterns?
  - Real-world is usually *unknown* and thus *approximated* by some model
  - Amount of parameters may influence the accuracy



Area of a lake

True p.d.f.



DISCUSSIONS

- Simulation
  - It is possible to simulate a (true) Markov process, and generate random observations accordingly
- Modeling the real world
  - In modeling the true random process in real world, we have no idea whether the true (unknown) random process is Markov process or not.

    We simply model it!
  - The observations can be used to estimate the model parameters such that the model can best account for the statistics of the observations

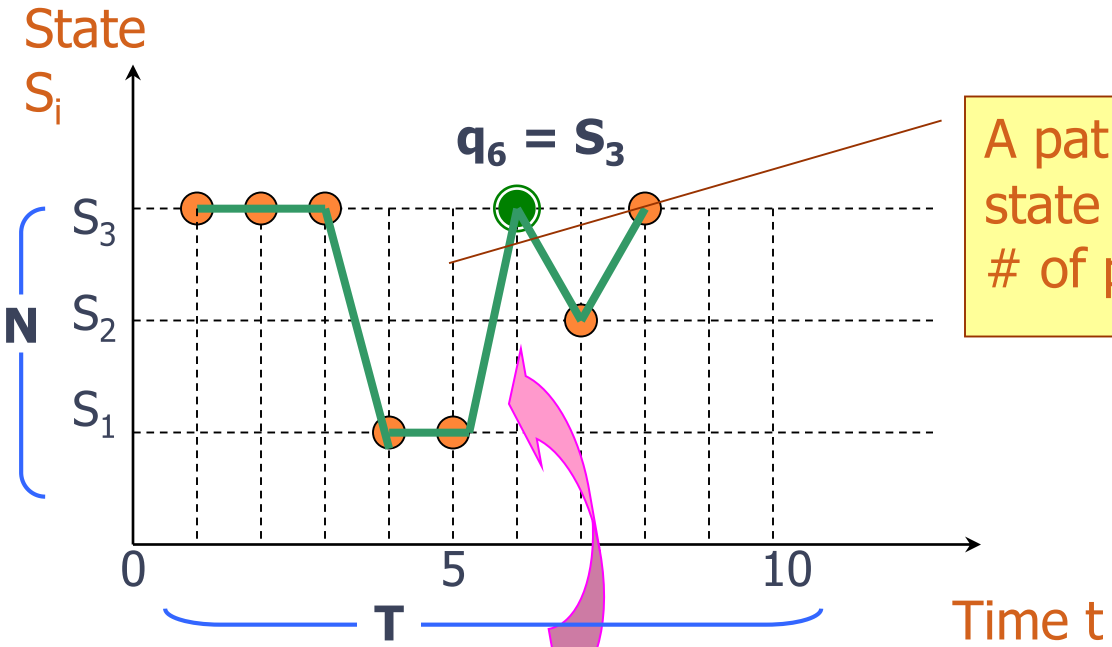
Estimation of Model Parameters

- $\grave{a}_{ij} = \frac{c_{ij}}{\sum_k c_{ik}}$

  c_{ik} is the occurrence count of transitions from state i to state k in all observation sequences
- $\grave{\pi}_j = \frac{c_j}{\sum_k c_k}$

  c_k is the occurrence count that the first state in a state sequence is state k for all observation sequences
- Example
  - $S_1, S_2, S_2, S_3, S_1, S_1, S_3, \ldots$
  - c_{12}++, c_{22}++, c_{23}++, c_{31}++, c_{11}++, c_{13}++, …

State Diagram



A path on a $N \times T$ state space
# of paths: N^T

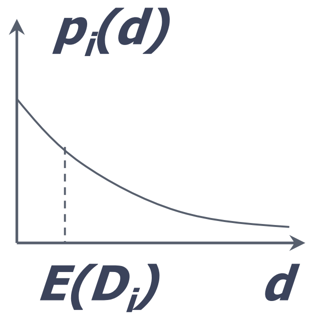
State sequence $Q = \{ q_t \}$

$q_1, q_2, q_3, q_4, q_5, \mathbf{q_6}, q_7, q_8, \ldots, q_T$

$S_3, S_3, S_3, S_1, S_1, \mathbf{S_3}, S_2, S_3, \ldots$

Probability of State Sequence

- The probability of Markov model (Π, A) generating the state sequence $Q = S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3$
  - $P(Q|\Pi, A) = P(S_3)\cdot P(S_3|S_3)\cdot P(S_3|S_3)\cdot P(S_1|S_3)\cdot$
    $P(S_1|S_1)\cdot P(S_3|S_1)\cdot P(S_2|S_3)\cdot P(S_3|S_2)$
    $= \pi_3\cdot a_{33}\cdot a_{33}\cdot a_{31}\cdot a_{11}\cdot a_{13}\cdot a_{32}\cdot a_{23}$

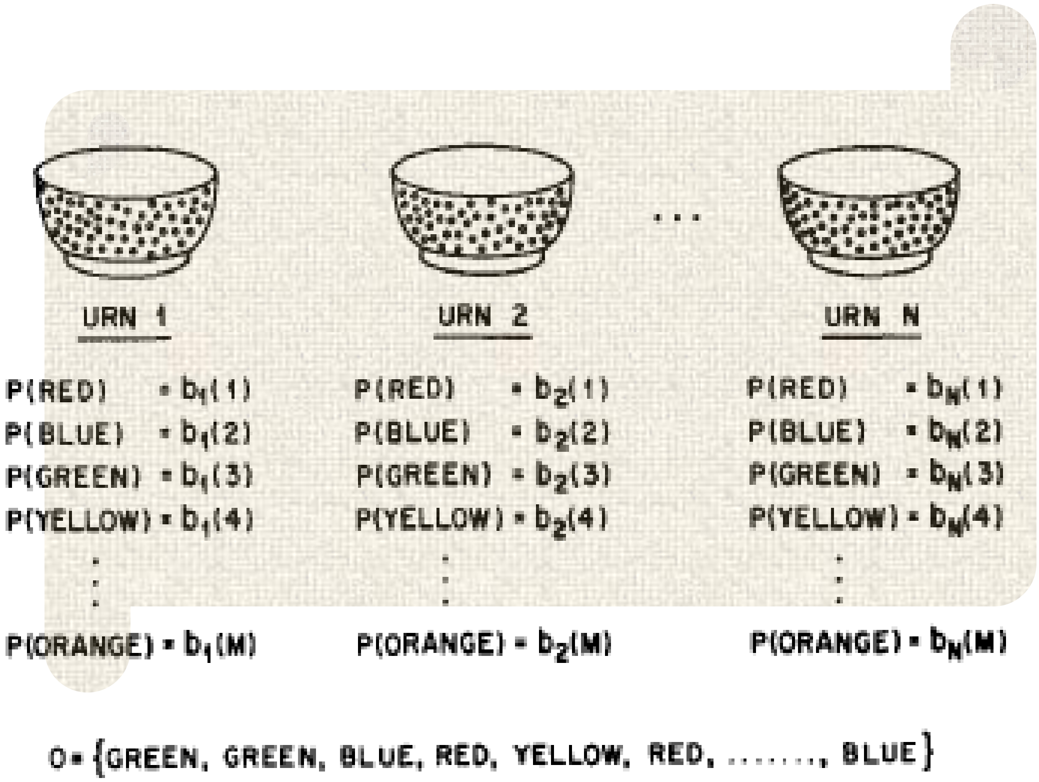


- System stays in S_i for time D_i (random variable)
  - $P(D_i = d|\Pi, A) = P(Q = \overbrace{S_i, S_i, S_i, \ldots, S_i}^{d}, S_{j\neq i}|\Pi, A, q_1 = S_i)$
    $= (a_{ii})^{d-1}(1-a_{ii}) \equiv p_i(d) \qquad d = 1, 2, \ldots$
  - Expectation $E(D_i) = \Sigma_d(d\cdot p_i(d)) = 1/(1-a_{ii}) \qquad a_{ii}\uparrow,\ E(D_i)\uparrow$
- Transition matrixes: modeling the duration

From Markov Model to HMM

- Markov Model
  - State transits stochastically and state sequence can be observed
- *Hidden* Markov Model
  - State transits stochastically, but state sequence *cannot be observed*
  - State sequence is unseen { q_t }
    - rainy, cloudy, sunny
  - Observation sequence { Ot }
    - very hot (VH), hot(H), warm(W), cool(CL), cold(CD)

Urn and Ball Model



- State transition
  - Roll the dice to select a urn
  - Multiple dices
- Observation
  - Pick up a ball from the selected urn
- The urns(j) has an associated distribution of balls, $b_j(O_t)$
- Double stochastic processes

Elements of HMM

- N : number of states
  - $S_1, S_2, \ldots, S_N$ (outcome symbols, not random)
- Initial state distribution $\Pi = \{\pi_i\}$
  - $\pi_i = P(q_1 = S_i)$
- State transition distribution $A = \{a_{ij}\}$
  - $a_{ij} = P(q_{t+1}=S_j|q_t = S_i)$
- State observation distribution $B = \{b_j(O_t)\}$
  - $b_j(O_t) = P(O_t|q_t = S_j)$
  - *The output* O_t depends on the current state q_t
  - Could be *discrete* or *continuous* functions
- $\lambda = (\Pi, A, B)$

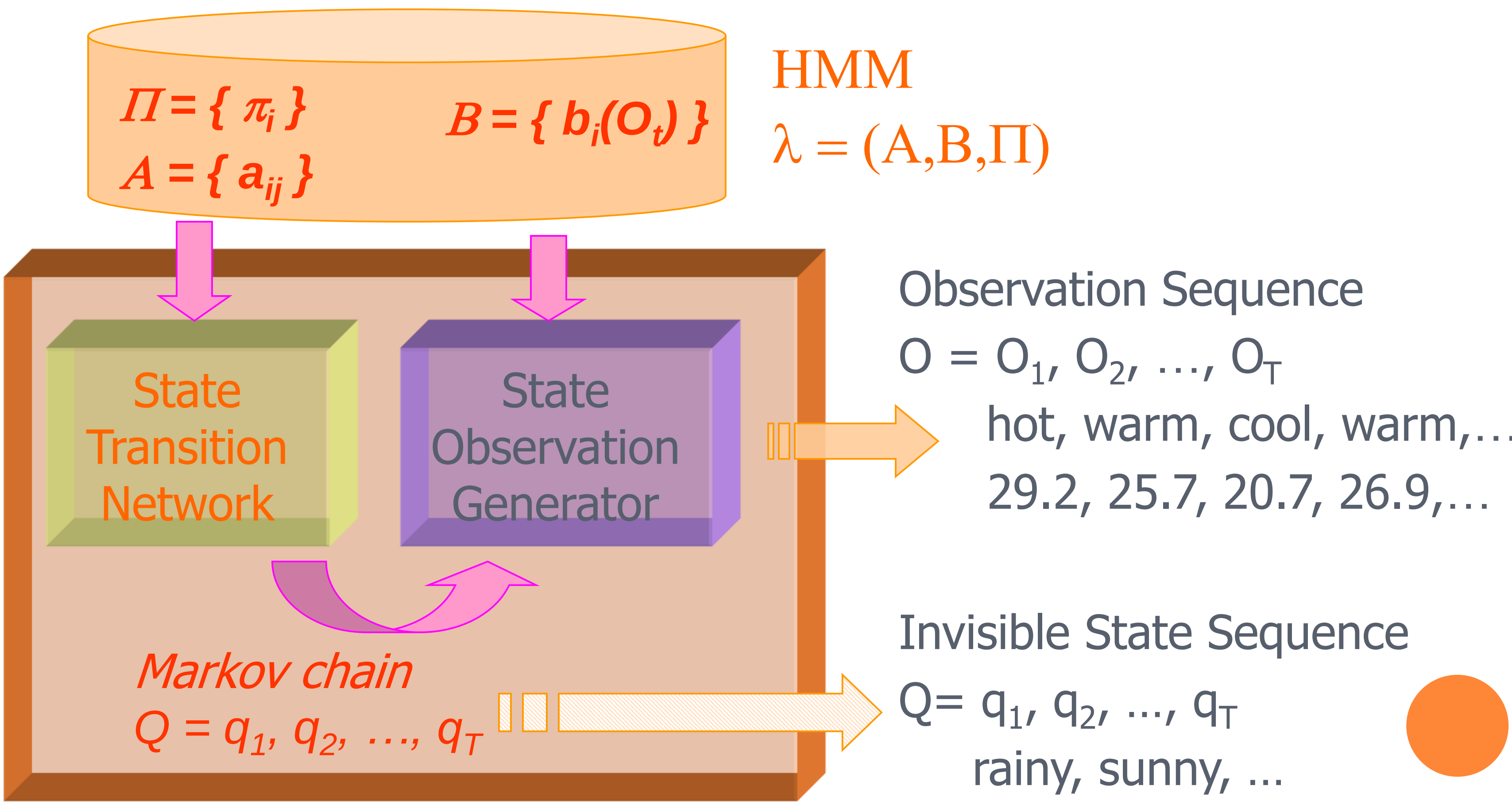
Observation Distribution

- Discrete : Probability weighting function
  - $b_j(O_t = v_k)$ k= 1, 2, …, K
  - v_k's may be obtained through encoding/quantization
  - Ex. $O_t=v_k$: very hot, hot, warm, cool, cold, very code
- Continuous : Probability density function
  - $b_j(O_t=x)$ could be a continuous function in any region which satisfied $\int b_j(O_t=x)dx = 1$
    - Ex. temperature could be 34, 31.5, 28, 29.3, ….
  - Example: weighted Gaussian mixtures
    - $b_j(O_t=x) = \Sigma_{m=1} c_{jm} \cdot \mathcal{N}(x; m_{jm}, \sigma_{jm}^2)$ $-\infty < x < \infty$

Semi-Continuous Observation Distr.

- Discrete : $b_j(O_t = v_k) = b_{jk}$
  - Quantization error, insufficient training (e.g. $b_{kj} = 0$)
- Continuous: $\mathcal{N}(m_{jm}, \sigma_{jm}^2)$ *for state j & mixture m*
  - Computation load is dependent of the numbers of models, states, or mixtures
  - 500 HMMs, 4 states, 10 mix → 20000 computations!
- Semi-Continuous : $b_j(O_t=x) = \Sigma_{k=1} b_{jk} \cdot \mathcal{N}(x; m_k, \sigma_k^2)$
  - Common Gaussian mixtures are shared among all states, but with all states have different mixture weights
  - Trade-off between discrete and continuous distribution
  - Can reduce the effect of quantization errors and insufficient training due to parameter sharing
  - m_k and σ_k are obtained from observations clustered to codeword v_k (codebook size is fixed)

HMM Simulator



Concepts of Hidden States

- Economic conditions
  - Hidden States: economic conditions
  - Observation: economic indexes
- Moods
  - Hidden States: mood states
  - Observation: color of dressing, face expression, interaction style
- Periods of life
  - Hidden States: stages of growth
  - Observation: weight, height, and other features
- Seasons
  - 4 seasons

To Evaluate The Learned

- Given HMM λ and unknown observation O
  - Calculation of $P(O \mid \lambda)$
  - $i^* = argmax_i \; P(O \mid \lambda_i)$: to find the HMM that most probably generates O
- Use for Recognition
  - With observed sequence O and the HMMs for some λ_i's, calculate $P(O \mid \lambda_i)$
  - Decide which model is most probable to generate the unknown observation O

To Learn From Observation

- We have the observation O, and *assume* it is generated by a HMM.
  - Is it possible to *learn/train* the *HMM parameters* λ that is optimal in some sense from the observation O?
  - HMM λ^* can *best represent* the observation O
  - The stochastic characteristics of observation O are caught and kept in model parameters of λ^*
  - $O \rightarrow \lambda^* = (\Pi, A, B)$
- Example
  - Temperature sequence $O \rightarrow$ HMM λ^*

To Guess the Unseen States Q

- Assume observation O is generated by an HMM λ.
  - Is it possible to reconstruct the *state sequence Q** that is optimal in some sense?
- Example
  - Temperature (O) & HMM (λ) → weather (Q*)
    - C, H, H, VH,… → rainy, sunny, sunny, sunny, …

Formulate Problems

- Evaluating: $P(O \mid \lambda)$
- Learning: $O \rightarrow \lambda^*$
- Labeling: $O, \lambda \rightarrow Q^*$
- $O = O_1, O_2, \ldots, O_T$ (random variables)
- $Q = q_1, q_2, \ldots, q_T$ (random variables)
- λ : model parameters
- Important Algorithms
  - Forward-backward procedure
  - Expectation-Maximization (EM) algorithm
  - Viterbi algorithm

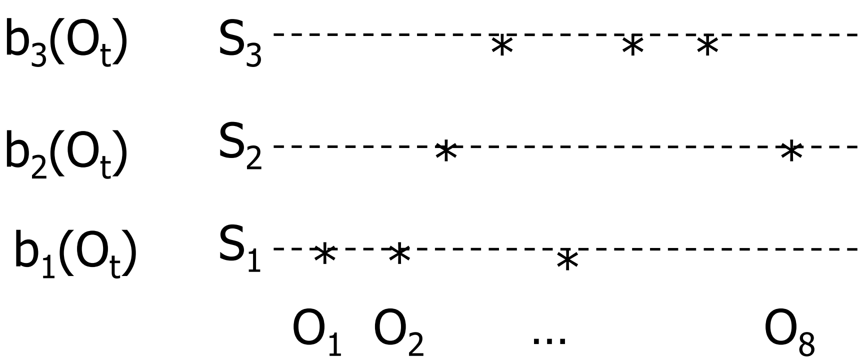
JOINT PROBABILITY P(O,Q)

- $P(Q|\lambda) = P(q_1)\cdot \Pi_{t=2\sim T}\, P\,(q_t|q_t,\ \lambda)$

  $= \pi_{q_1}\cdot a_{q_1 q_2}\cdot a_{q_2 q_3}\cdot\ \ldots\ \cdot a_{q_{T\text{-}1} q_T}\ (Q = q_1, q_2,\ \ldots, q_T)$
- $P(O|Q,\lambda) = \Pi_{t=1\sim T}\, P\,(O_t|q_t,\ \lambda)$

  $= b_{q_1}(O_1)\ \cdot b_{q_2}(O_2)\ \cdot \ldots\ \cdot b_{q_T}(O_T)$
- $P(O,Q|\lambda)\quad = P(Q|\lambda)\cdot P(O|\lambda,Q)$

  $= \pi_{q_1}\cdot\ b_{q_1}(O_1)\cdot a_{q_1 q_2}\cdot\ b_{q_2}(O_2)\ \cdot \ldots a_{q_{T\text{-}1} q_T}\cdot b_{q_T}(O_T)$



$P(Q)=\pi_1 a_{11} a_{12} a_{23} a_{31} a_{13} a_{33} a_{32}$

$P(O|Q)=b_1(O_1)\ b_1(O_2)\ b_2(O_3)\ b_3(O_4)\ \cdot b_1(O_5)\ b_3(O_6)\ b_3(O_7)\ b_2(O_8)$

Issue of Computation Complexity

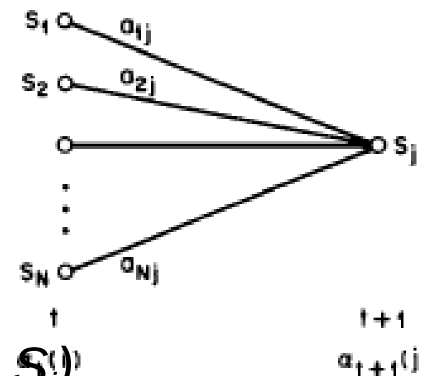
- $P(O|\lambda) = \Sigma_Q P(O,Q|\lambda)$
  $= \Sigma_Q P(Q|\lambda) \cdot P(O|\lambda,Q)$
  $= \Sigma_Q [\pi_{q_1} \cdot b_{q_1}(O_1) \cdot a_{q_1 q_2} \cdot b_{q_2}(O_2) \cdot \ldots a_{q_{T-1} q_T} \cdot b_{q_T}(O_T)]$
- $P(O|\lambda)$ is the *summation of $P(O,Q|\lambda)$* over *ALL paths* on the state space
  - Note: path Q is *hidden* in HMM (cannot be seen)
  - $P(X) = \Sigma_Y P(X, Y)$
- On $N \times T$ state space, there are totally *N^T paths*
  - Infeasible if every path is calculated independently
  - $(2T-1)*N^T$ multiplications and N^T-1 additions

Forward Procedure

- $\alpha_t(j) \equiv P(O_1, \ldots, O_t, q_t = S_j|\lambda)$
- $\alpha_1(j) \;= P(O_1, q_1 = S_j|\lambda)$

  $= P(q_1 = S_j|\lambda) \cdot P(O_1|\lambda, q_1 = S_j)$

  $= \pi_j \cdot b_j(O_1)$



- $\alpha_{t+1}(j) \equiv P(O_1, \ldots, O_{t+1}, q_{t+1} = S_j|\lambda)$

  $= P(O_1, \ldots, O_t, q_{t+1} = S_j|\lambda) \cdot P(O_{t+1}|\lambda, O_1, \ldots, O_t, q_{t+1} = S_j)$

  $= [\Sigma_i P(O_1, \ldots, O_t, q_t = S_i, q_{t+1} = S_j|\lambda)] \cdot b_j(O_{t+1})$

  $= \{\Sigma_i [P(O_1, \ldots, O_t, q_t = S_i|\lambda) \cdot P(q_{t+1} = S_j|\lambda, O_1, \ldots, O_t, q_t = S_i)]\} \cdot b_j(O_{t+1})$

  $= [\Sigma_i (\alpha_t(i) \cdot a_{ij})] \cdot b_j(O_{t+1})$

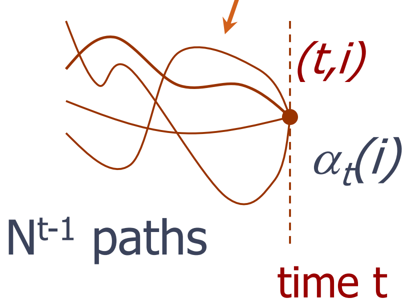
- $\mathbf{P(O|\lambda) = \Sigma_i \alpha_T(i)}$

Stochastic Dependency

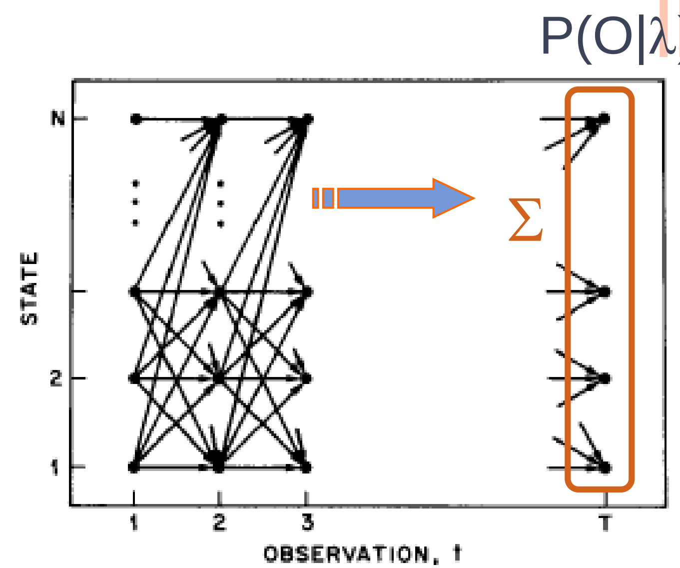
$$
\begin{array}{ccccccccccc}
q_1 & \rightarrow & q_2 & \rightarrow & q_3 & \rightarrow & \ldots & \rightarrow & q_t & \rightarrow \ldots \rightarrow & q_T \\
\downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \downarrow \\
O_1 & & O_2 & & O_3 & & \ldots & & O_t & & O_T
\end{array}
$$

Forward Procedure (cont'd)

- Calculate $\alpha_t(i)$ for every *grid location (t,i)* on state space in left-to-right direction
- $\alpha_t(i)$ is the *temporary summation* of $P(O,Q|\lambda)$ over *all partial paths terminating at (t, i)*
- Dynamic Programming
- Complexity ≒ N^2T



$\alpha_t(i) \equiv P(O_1, \ldots, O_t, q_t = S_i|\lambda)$



Backward Procedure

- $\beta_t(i) \equiv P(O_{t+1}, \ldots, O_T \mid q_t = S_i, \lambda)$

  $= \Sigma_j P(O_{t+1}, \ldots, O_T, q_{t+1}=S_j \mid q_t = S_i, \lambda)$

  $= \Sigma_j \{ P(q_{t+1}=S_j \mid q_t = S_i, \lambda) \cdot P(O_{t+1}, \ldots, O_T \mid q_t=S_j, q_{t+1} = S_j, \lambda) \}$

  $= \Sigma_j \{ a_{ij} \cdot [P(O_{t+1} \mid q_{t+1} = S_j, \lambda) \cdot P(O_{t+2}, \ldots, O_T \mid q_{t+1} = S_j, O_{t+1}, \lambda)] \}$

  $= \Sigma_j [a_{ij} \cdot b_j(O_{t+1}) \cdot \beta_{t+1}(j)]$

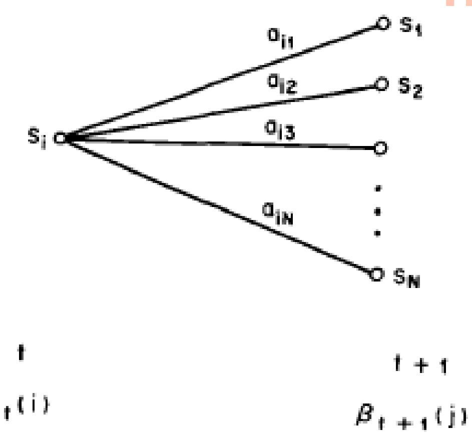
- $\beta_{T-1}(i) = P(O_T \mid q_{T-1} = S_i, \lambda) = \Sigma_j P(O_T, q_T = S_j, \mid q_{T-1} = S_i, \lambda)$

  $= \Sigma_j [P(q_T = S_j, \mid q_{T-1} = S_i, \lambda) \cdot P(O_T \mid q_T = S_j, q_{T-1} = S_i, \lambda)]$

  $= \Sigma_j [a_{ij} \cdot b_j(O_T)]$

- It is there appropriate to assign $\beta_T(i) \equiv 1$ such that the inductive formula holds for t = T-1

- $\beta_t(i)$ is calculated for every grid point (t,i) in right-to-left direction



Find Optimal States q_t^*

- $P(O, q_t = S_i|\lambda) = P(O_1, \ldots, O_t, O_{t+1}, \ldots, O_T, q_t = S_i|\lambda)$

  $= P(O_1, \ldots, O_t, q_t = S_i|\lambda) \cdot P(O_{t+1}, \ldots, O_T \mid q_t = S_i, O_1, \ldots, O_t, \lambda)$

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

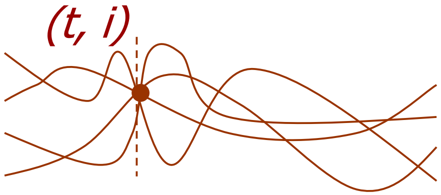
  - Summation of $P(O,Q|\lambda)$ over all paths passing (t, i)

$q_t^* \equiv \text{argmax}_i \{ P(q_t = S_i|O, \lambda) \}$

$= \text{argmax}_i \{ P(O, q_t = S_i|\lambda) / P(O|\lambda) \}$

$= \text{argmax}_i \{ \gamma_t(i) \}$

  - $\gamma_t(i) \equiv P(q_t = S_i|O,\lambda) = \alpha_t(i) \cdot \beta_t(i) / P(O|\lambda)$



N^{T-1} paths

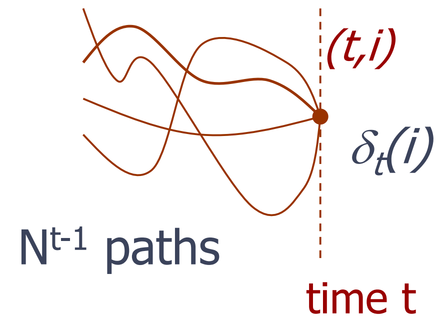
- q_t^*'s are the states optimized for each time individually
- $P(q_1^*, q_2^*, \ldots, q_T^*, O|\lambda)$ is *NOT* the path with highest probability $P(Q, O|\lambda)$ among all paths Q's

Find Optimal Path - Viterbi Algorithm

- $P^* = \max_Q P(O,Q|\lambda)$, $Q^* = \operatorname{argmax}_Q P(O,Q|\lambda)$

  Q^* is the path with the highest probability P^*

- $\delta_t(i) \equiv \max_{q_1,q_2,\ldots,q_t=S_i} P(O_1, \ldots, O_t, q_t = S_i \mid \lambda)$
- $\delta_t(i)$ is the *temporary maximum* of $P(O,Q|\lambda)$ for *all partial paths terminating at (t, i)*



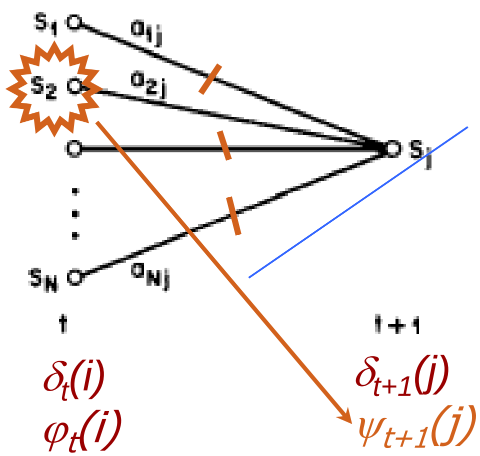
- $\delta_1(j) = \pi_j \cdot b_j(O_1)$, $\varphi_1(j) = 0$
- $\delta_{t+1}(j) = \max_i [\delta_t(i) \cdot a_{ij}] \cdot b_j(O_{t+1})$

  $\psi_{t+1}(j) = \operatorname{argmax}_i [\delta_t(i) \cdot a_{ij}]$
- $P^* = \max_i [\delta_T(i)]$

  $q_T' = \operatorname{argmax}_i [\delta_T(i)]$

  $q_t' = \varphi_{t+1}(q_{t+1}')$ backtracking

  $Q^* = q_1', q_2', \ldots, q_T'$

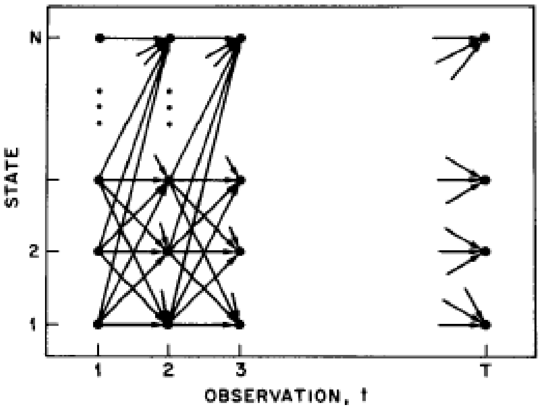


At any time, only the winner can survive and be memorized

Viterbi Algorithm (cont'd)

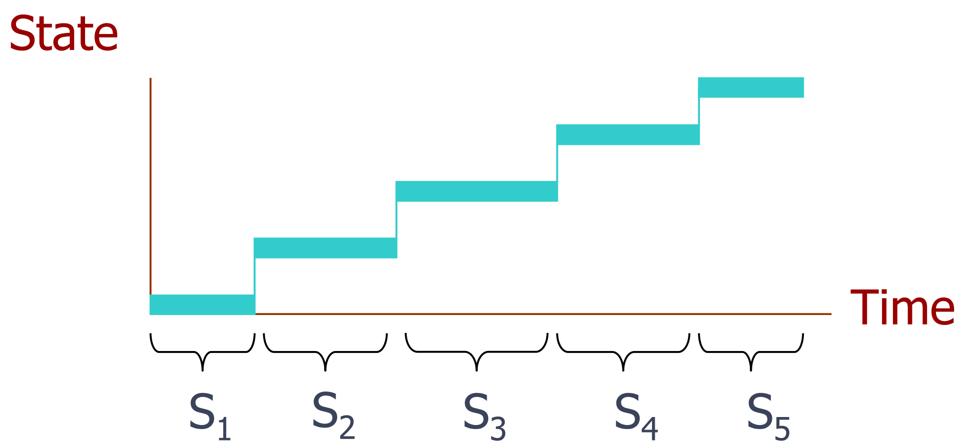
- Take *log*, the multiplications becomes additions
  - $\delta'_{t+1}(j) = \max_i [\delta'_t(i) + \log(a_{ij})] + \log[b_j(O_{t+1})]$

    $\psi_{t+1}(j) = \operatorname{argmax}_i [\delta'_t(i) + \log[a_{ij})]$
  - Avoid underflow for long observation sequence
- Concept analogy
  - To pick up the maximum points on the street
  - Can proceed in directions → and $\boxed{\nearrow}$
  - Get $\log(a_{ij})$ points by transit from i to j
  - Get $\log[b_j(O_t)]$ points on the cross (t, j)



Viterbi Algorithm (cont'd)

- For left-to-right HMM, the optimal path Q* obtained from Viterbi algorithm can segment the observation sequence into different states
- Can be used for automatic segmentation

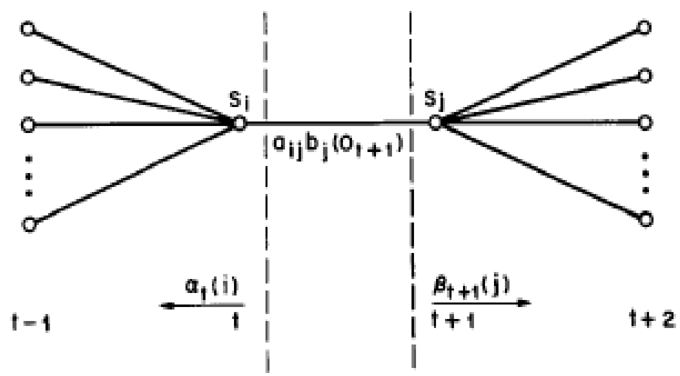


EM ALGORITHM

- Expectation-Maximization (8.30)
- Find λ^* by iterative procedure
  - $P(O\,|\,\lambda') > P(O\,|\,\lambda)$ and can find local maximum
  - Converge when the change of likelihood is small
- $\gamma_t(i) \equiv P(q_t = S_i|O,\lambda)$ *the contribution of* O_t *for state i*
- $\varepsilon_t(i, j) \equiv P(q_t = S_i,\ q_{t+1} = S_j|O,\lambda)$

$$\gamma_t(i) = \frac{\alpha_t(i)\ \beta_t(i)}{P(O|\lambda)}$$

$$\xi_t(i, j) = \frac{\alpha_t(i)\ a_{ij} b_j(O_{t+1})\ \beta_{t+1}(j)}{P(O|\lambda)}$$



EM Algorithm (cont'd)

- Baum-Welch Reestimation Procedure

$$\bar{\pi}_i = \text{expected frequency (number of times) in state } S_i \text{ at time } (t = 1) = \gamma_1(i)$$

$$\bar{a}_{ij} = \frac{\text{expected number of transitions from state } S_i \text{ to state } S_j}{\text{expected number of transitions from state } S_i}$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

Σ_t: summation the contributions of all observations $\{O_t\}$

$$\bar{b}_j(k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j}$$

$$= \frac{\sum_{\substack{t=1 \\ \text{s.t. } O_t = v_k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}.$$

For Discrete state observation distribution

$b_j(O_t = v_k)$

EM Algorithm (cont'd)

For Continuous state observation distribution

$b_j(O_t=x) = \Sigma_k\, c_{jk} \cdot N(x;\mu_{jk},U_{jk})$

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

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

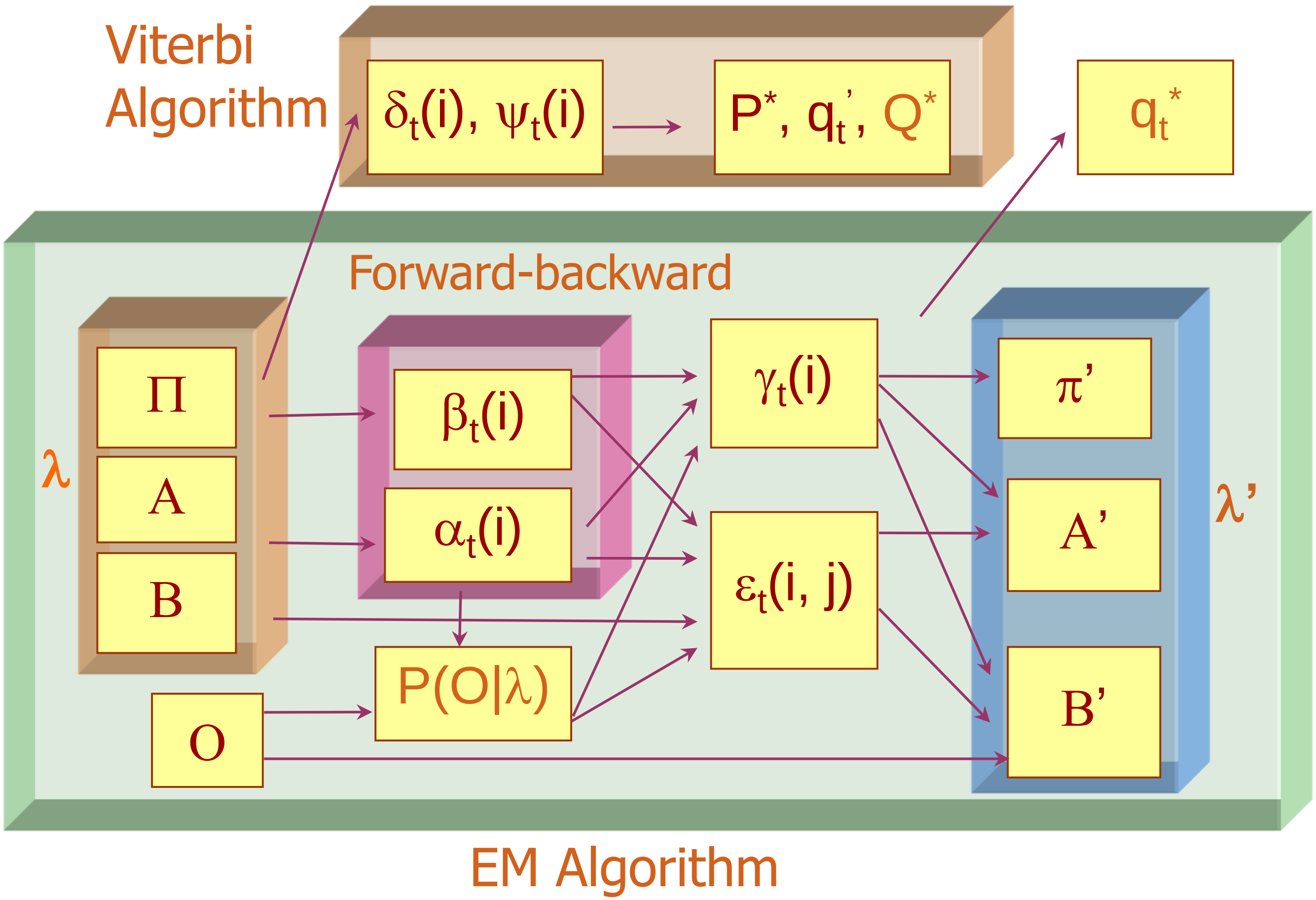
$$\bar{U}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot (O_t - \mu_{jk})(O_t - \mu_{jk})'}{\sum_{t=1}^{T} \gamma_t(j,k)}$$

$$\gamma_t(j,k) = \left[\frac{\alpha_t(j)\,\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\,\beta_t(j)}\right]\left[\frac{\boxed{c_{jk}\,\mathfrak{N}(O_t, \mu_{jk}, U_{jk})}}{\sum_{m=1}^{M} c_{jm}\,\mathfrak{N}(O_t, \mu_{jm}, U_{jm})}\right].$$

$\gamma_t(j)$: How likely O_t belongs to state j

How likely O_t belongs to mixture k if in state j

Dependency Diagram



HMM for Pattern Recognition

- Assume there are M patterns to be recognized, and we have observations for these patterns. Then, we can train the model λ_i for i-th pattern using the observations of this pattern
- $i^* = \text{argmax}_i\, P(O\,|\,\lambda_i)$ i=1, 2, …, M

Topology of HMM

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}.$$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}.$$

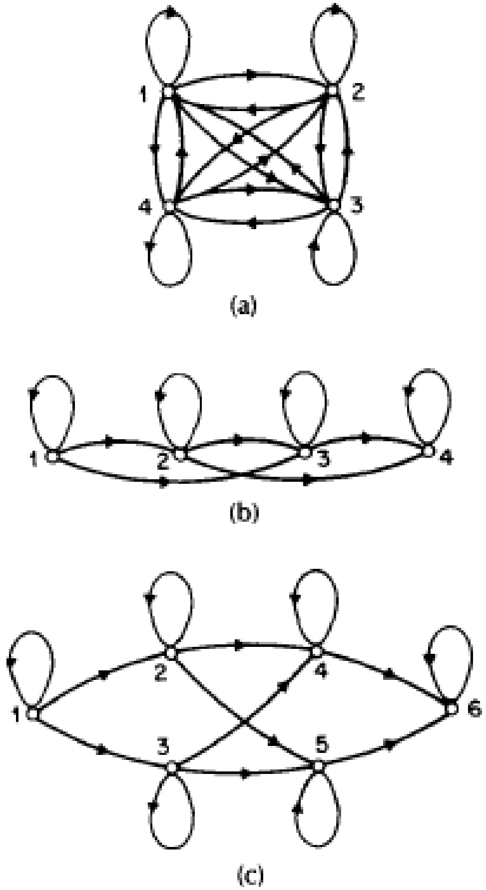


Fig. 7. Illustration of 3 distinct types of HMMs. (a) A 4-state ergodic model. (b) A 4-state left-right model. (c) A 6-state parallel path left-right model.

Scaling Issue

- If the length of observation sequence is long, there may arise the problem of underflow since all probabilities are less than 1.
  - $P(O \mid \lambda) = \Sigma_Q \pi_{q_1} \cdot b_{q_1}(O_1) \cdot a_{q_1 q_2} \cdot b_{q_2}(O_2) \cdot \ldots a_{q_{T-1} q_T} \cdot b_{q_T}(O_T)$
- $S_t = 1/\Sigma_i \, \alpha_t(i)$ is used for scaling $\alpha_t(i)$ and $\beta_t(i)$ at time t in Forward-backward procedure
  - It can be proved scaling will NOT influence the reestimation formula
- In Viterbi algorithm, probability is taken “log” and then accumulated

Multiple Observation Sequences

$$\overline{a_{ij}} = \frac{\displaystyle\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i)\, a_{ij} b_j(O_{t+1}^{(k)})\, \beta_{t+1}^k(j)}{\displaystyle\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i)\, \beta_t^k(i)}$$

$$\overline{b_j}(\ell) = \frac{\displaystyle\sum_{k=1}^{K} \frac{1}{P_k} \sum_{\substack{t=1 \\ \text{s.t. } O_t = v_\ell}}^{T_k-1} \alpha_t^k(i)\, \beta_t^k(i)}{\displaystyle\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i)\, \beta_t^k(i)}$$

- The Reestimation formula can be modified for multiple observation sequences
- $P_k = P(O^{(k)} \mid \lambda)$

Initial Model

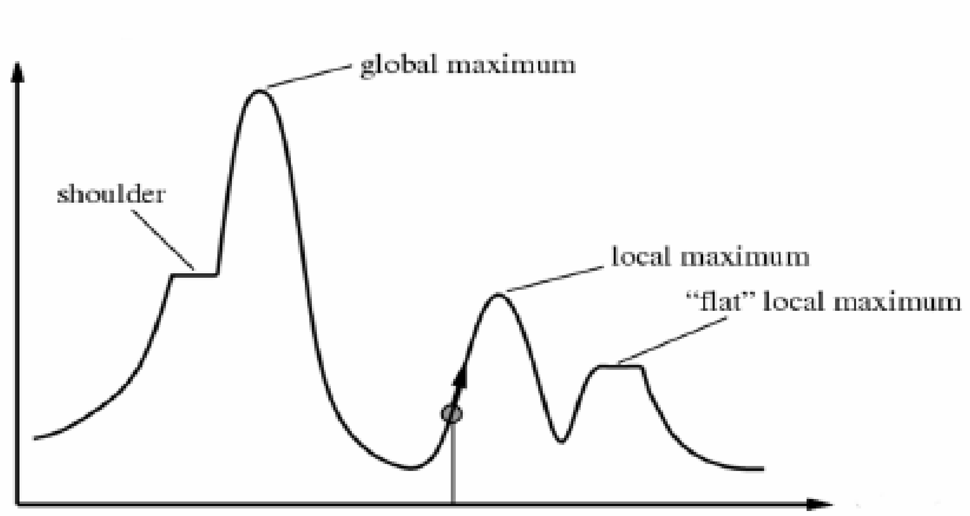
- Good initial estimate is important for avoiding local maximum
- Random or uniform distributed parameters are adequate for Π and A, but not enough for B.
- To obtain better initial estimate
  - Manual segmentation
  - Segmental K-means (Viterbi segmentation)

Limitation of HMM

- First-Order Assumption
  - Markov Chain
  - Leading to exponential duration model
- Conditional Independence Assumption
  - The Observation depends ONLY on the CURRENT state
  - Maximum entropy Markov model (MEMM)
  - Conditional random field (CRF)

Limitation of HMM (cont'd)

- Local optimum guaranteed by EM algorithm

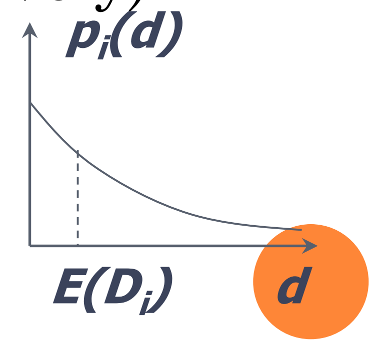


Generalization of MM/HMM

- Observation O can be vector instead of scalar
- Concept of time can be generalized as other physical dimension
  - Example: x-axis in Euclidean space
  - State sequence $Q = \{ q_t \}$
  - Observation sequence $O = \{ O_t \}$
  - Recognition of patterns which vary with some dimension (e.g. image, gene pattern, …)

Generalization of MM/HMM

- State can be flexibly defined
  - Number of state might be very large
- Better duration model
  - Use Gaussian or Gamma function as duration distribution instead of exponential function (a_{ij} is exponential duration distribution effectively)
  - Slight improvement achieved



Applications of HMM

- Statistical Model for *Trajectory*
- Speech recognition/synthesis
  - Spectrum trajectory
- Image recognition
  - row trajectory
- Natural language processing
  - Word segmentation / POS tagging
- Gesture recognition
  - 2D/3D trajectory
  - Action recognition (multi-point trajectory)
- Melody recognition/synthesis
  - Note trajecotry