Hidden Markov Model

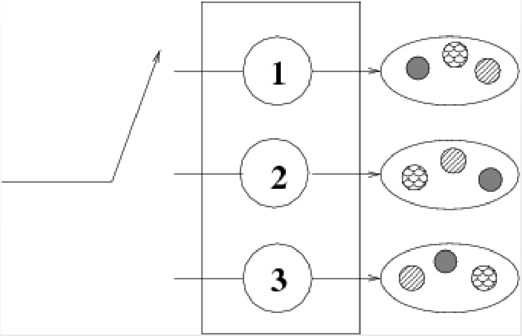
Jia Li

Department of Statistics
The Pennsylvania State University

Email: jiali@stat.psu.edu
http://www.stat.psu.edu/∼jiali

Hidden Markov Model

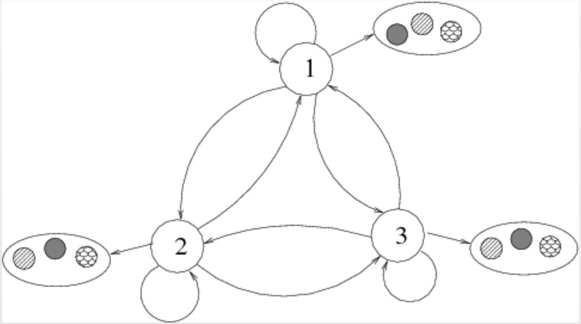
- Hidden Markov models have close connection with mixture models.
- A mixture model generates data as follows.



- For sequence or spatial data, the assumption of independent samples is too constrained.
- The statistical dependence among samples may bear critical information.
- Examples:
  - Speech signal
  - Genomic sequences

Model Setup

- Suppose we have a sequential data $\mathbf{u} = \{u_1, u_2, ..., u_t, ..., u_T\}$, $u_t \in \mathcal{R}^d$.
- As in the mixture model, every u_t, $t = 1, ..., T$, is generated by a hidden state, s_t.



- The underlying states follow a Markov chain.
  - Given present, the future is independent of the past:

$$P(s_{t+1} \mid s_t, s_{t-1}, \ldots, s_0) = P(s_{t+1} \mid s_t) \; .$$

  - Transition probabilities:

$$a_{k,l} = P(s_{t+1} = l \mid s_t = k) \; ,$$

$k, l = 1, 2, \ldots, M$, where M is the total number of states. Initial probabilities of states: π_k.

$$\sum_{l=1}^{M} a_{k,l} = 1 \quad \text{for any } k \; , \sum_{k=1}^{M} \pi_k = 1 \; .$$

- $P(s_1, s_2, ..., s_T) = P(s_1)P(s_2|s_1)P(s_3|s_2) \cdots P(s_T|s_{T-1})$
  $= \pi_{s_1} a_{s_1,s_2} a_{s_2,s_3} \cdots a_{s_{T-1},s_T}$.
- Given the state s_t, the observation u_t is independent of other observations and states.
- For a fixed state, the observation u_t is generated according to a fixed probability law.

- Given state k, the probability law of U is specified by $b_k(u)$.
  - Discrete: suppose U takes finitely many possible values, $b_k(u)$ is specified by the pmf (probability mass function).
  - Continuous: most often the Gaussian distribution is assumed.

$$b_k(u) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_k|}} \exp(-\frac{1}{2}(u - \mu_k)^t \Sigma_k^{-1} (u - \mu_k))$$

- In summary:

$$
\begin{aligned}
P(\mathbf{u}, \mathbf{s}) &= P(\mathbf{s})P(\mathbf{u} \mid \mathbf{s}) \\
&= \pi_{s_1} b_{s_1}(u_1) a_{s_1,s_2} b_{s_2}(u_2) \cdots a_{s_{T-1},s_T} b_{s_T}(u_T) \ .
\end{aligned}
$$

$$
\begin{aligned}
P(\mathbf{u}) &= \sum_{\mathbf{s}} P(\mathbf{s})P(\mathbf{u} \mid \mathbf{s}) \qquad \textit{total prob. formula} \\
&= \sum_{\mathbf{s}} \pi_{s_1} b_{s_1}(u_1) a_{s_1,s_2} b_{s_2}(u_2) \cdots a_{s_{T-1},s_T} b_{s_T}(u_T)
\end{aligned}
$$

Example

- Suppose we have a video sequence and would like to automatically decide whether a speaker is in a frame.
- Two underlying states: with a speaker (state 1) vs. without a speaker (state 2).
- From frame 1 to T, let s_t, $t = 1, ..., T$ denotes whether there is a speaker in the frame.
- It does not seem appropriate to assume that s_t's are independent. We may assume the state sequence follows a Markov chain.
  - If one frame contains a speaker, it is highly likely that the next frame also contains a speaker because of the strong frame-to-frame dependence. On the other hand, a frame without a speaker is much more likely to be followed by another frame without a speaker.

- For a computer program, the states are unknown. Only features can be extracted for each frame. The features are the observation, which can be organized into a vector.
- The goal is to figure out the state sequence given the observed sequence of feature vectors.
- We expect the probability distribution of the feature vector to differ according to the state. However, these distributions may overlap, causing classification errors.
- By using the dependence among states, we may make better guesses of the states than guessing each state separately using only the feature vector of that frame.

Model Estimation

- Parameters involved:
  - Transition probabilities: $a_{k,l}$, $k, l = 1, ..., M$.
  - Initial probabilities: π_k, $k = 1, ..., M$.
  - For each state k, μ_k, Σ_k.

Definitions

- Under a given set of parameters, let $L_k(t)$ be the conditional probability of being in state k at position t given the entire observed sequence $\mathbf{u} = \{u_1, u_2, ..., u_T\}$.

$$L_k(t) = P(s_t = k|\mathbf{u}) = \sum_{\mathbf{s}} P(\mathbf{s} \mid \mathbf{u}) I(s_t = k) \ .$$

- Under a given set of parameters, let $H_{k,l}(t)$ be the conditional probability of being in state k at position t and being in state l at position $t+1$, i.e., seeing a transition from k to l at t, given the entire observed sequence $\mathbf{u}$.

$$\begin{aligned} H_{k,l}(t) &= P(s_t = k, s_{t+1} = l|\mathbf{u}) \\ &= \sum_{\mathbf{s}} P(\mathbf{s} \mid \mathbf{u}) I(s_t = k) I(s_{t+1} = l) \end{aligned}$$

- Note that $L_k(t) = \sum_{l=1}^{M} H_{k,l}(t)$, $\sum_{k=1}^{M} L_k(t) = 1$.

- Maximum likelihood estimation by EM:
  - E step: Under the current set of parameters, compute $L_k(t)$ and $H_{k,l}(t)$, for $k, l = 1, ..., M$, $t = 1, ..., T$.
  - M step: Update parameters.

$$\mu_k = \frac{\sum_{t=1}^{T} L_k(t) u_t}{\sum_{t=1}^{T} L_k(t)}$$

$$\Sigma_k = \frac{\sum_{t=1}^{T} L_k(t)(u_t - \mu_k)(u_t - \mu_k)^t}{\sum_{t=1}^{T} L_k(t)}$$

$$a_{k,l} = \frac{\sum_{t=1}^{T-1} H_{k,l}(t)}{\sum_{t=1}^{T-1} L_k(t)} \; .$$

- Note: the initial probabilities of states π_k are often manually determined. We can also estimate them by

$$\begin{aligned} \pi_k &\propto \sum_{t=1}^{T} L_k(t)\ , \quad \sum_{k=1}^{M} \pi_k = 1 \\ \text{or} \quad \pi_k &\propto L_k(1) \end{aligned}$$

Comparison with the Mixture Model

- $L_k(t)$ is playing the same role as the posterior probability of a component (state) given the observation, i.e., $p_{t,k}$.

$$\begin{aligned} L_k(t) &= P(s_t = k \mid u_1, u_2, ..., u_t, ..., u_T) \\ p_{t,k} &= P(s_t = k \mid u_t) \end{aligned}$$

If we view a mixture model as a special hidden Markov model with the underlying state process being i.i.d (a reduced Markov chain), $p_{t,k}$ is exactly $L_k(t)$.

- The posterior probabilities $p_{t,k}$ in the mixture model can be determined using only sample u_t because of the independent sample assumption.
- $L_k(t)$ depends on the entire sequence because of the underlying Markov process.
- For a mixture model, we have

$$\mu_k = \frac{\sum_{t=1}^{T} p_{t,k} u_t}{\sum_{t=1}^{T} p_{t,k}}$$

$$\Sigma_k = \frac{\sum_{t=1}^{T} p_{t,k} (u_t - \mu_k)(u_t - \mu_k)^t}{\sum_{t=1}^{T} p_{t,k}}$$

Derivation from EM

- The incomplete data are $\mathbf{u} = \{u_t : t = 1, ..., T\}$. The complete data are $\mathbf{x} = \{s_t, u_t : t = 1, ..., T\}$.
- Note $Q(\theta'|\theta) = E(\log(f(\mathbf{x}|\theta'))|\mathbf{u}, \theta)$.
- Let $\mathcal{M} = \{1, 2, ..., M\}$.

- The function $f(\mathbf{x} \mid \theta')$ is

$$\begin{aligned} f(\mathbf{x} \mid \theta') &= P(\mathbf{s} \mid \theta')P(\mathbf{u} \mid \mathbf{s}, \theta') \\ &= P(\mathbf{s} \mid a'_{k,l} : k, l \in \mathcal{M})P(\mathbf{u} \mid \mathbf{s}, \mu'_k, \mathbf{\Sigma}'_k : k \in \mathcal{M}) \\ &= \pi'_{s_1} \prod_{t=2}^{T} a'_{s_{t-1},s_t} \times \prod_{t=1}^{T} P(u_t \mid \mu'_{s_t}, \mathbf{\Sigma}'_{s_t}) \, . \end{aligned}$$

We then have

$$\begin{aligned} \log f(\mathbf{x} \mid \theta') &= \log(\pi'_{s_1}) + \sum_{t=2}^{T} \log a'_{s_{t-1},s_t} + \\ & \sum_{t=1}^{T} \log P(u_t \mid \mu'_{s_t}, \Sigma'_{s_t}) \end{aligned} \qquad (1)$$

$$
\begin{aligned}
& E(\log f(\mathbf{x} \mid \theta') | \mathbf{u}, \theta) \\
= & \sum_{\mathbf{s}} P(\mathbf{s}|\mathbf{u}, \theta) \left[\log(\pi'_{s_1}) + \sum_{t=2}^{T} \log a'_{s_{t-1}, s_t} + \right. \\
& \left. \sum_{t=1}^{T} \log P(u_t \mid \mu'_{s_t}, \boldsymbol{\Sigma}'_{s_t}) \right] \\
= & \sum_{k=1}^{M} L_k(1) \log(\pi'_k) + \sum_{t=2}^{T} \sum_{k=1}^{M} \sum_{l=1}^{M} H_{k,l}(t) \log a'_{k,l} \\
& + \sum_{t=1}^{T} \sum_{k=1}^{M} L_k(t) \log P(u_t \mid \mu'_k, \Sigma'_k)
\end{aligned}
$$

- Prove the equality of the second term

$$\sum_{\mathbf{s}} P(\mathbf{s}|\mathbf{u}, \theta) \sum_{t=2}^{T} \log a'_{s_{t-1}, s_t}$$

$$= \sum_{t=2}^{T} \sum_{k=1}^{M} \sum_{l=1}^{M} H_{k,l}(t) \log a'_{k,l}$$

Similar proof applies to the equality corresponding to other terms.

$$
\begin{aligned}
& \sum_{\mathbf{s}} P(\mathbf{s}|\mathbf{u},\theta) \sum_{t=2}^{T} \log a'_{s_{t-1},s_t} \\
= & \sum_{\mathbf{s}} P(\mathbf{s}|\mathbf{u},\theta) \sum_{t=2}^{T} \sum_{k=1}^{M} \sum_{l=1}^{M} I(s_{t-1}=k) I(s_t=l) \log a'_{k,l} \\
= & \sum_{\mathbf{s}} \sum_{t=2}^{T} \sum_{k=1}^{M} \sum_{l=1}^{M} P(\mathbf{s}|\mathbf{u},\theta) I(s_{t-1}=k) I(s_t=l) \log a'_{k,l} \\
= & \sum_{t=2}^{T} \sum_{k=1}^{M} \sum_{l=1}^{M} \left[\sum_{\mathbf{s}} P(\mathbf{s}|\mathbf{u},\theta) I(s_{t-1}=k) I(s_t=l) \right] \log a'_{k,l} \\
= & \sum_{t=2}^{T} \sum_{k=1}^{M} \sum_{l=1}^{M} H_{k,l}(t) \log a'_{k,l}
\end{aligned}
$$

- The maximization of the above expectation gives the update formulas in the M-step.
- Note that the optimization of μ'_k, Σ'_k can be separated from that of $a'_{k,l}$ and π_k. The optimization of $a'_{k,l}$ can be separated for different k.
- The optimization of μ'_k and Σ'_k is the same as for the mixture model with $p_{t,k}$ replaced by $L_k(t)$.

Forward-Backward Algorithm

- The forward-backward algorithm is used to compute $L_k(t)$ and $H_{k,l}(t)$ efficiently.
- The amount of computation needed is at the order of M^2T. Memory required is at the order of MT.
- Define the forward probability $\alpha_k(t)$ as the joint probability of observing the first t vectors u_τ, $\tau = 1, ..., t$, and being in state k at time t.

$$\alpha_k(t) = P(u_1, u_2, ..., u_t, s_t = k)$$

- This probability can be evaluated by the following recursive formula:

$$\alpha_k(1) = \pi_k b_k(u_1) \quad 1 \le k \le M$$

$$\alpha_k(t) = b_k(u_t) \sum_{l=1}^{M} \alpha_l(t-1) a_{l,k} \,, \quad 1 < t \le T, \ 1 \le k \le M \,.$$

- Proof:

$$
\begin{aligned}
& \alpha_k(t) = P(u_1, u_2, \ldots, u_t, s_t = k) \\
= & \sum_{l=1}^{M} P(u_1, u_2, \ldots, u_t, s_t = k, s_{t-1} = l) \\
= & \sum_{l=1}^{M} P(u_1, \ldots, u_{t-1}, s_{t-1} = l) \cdot P(u_t, s_t = k \mid s_{t-1} = l, u_1, \ldots, u_{t-1}) \\
= & \sum_{l=1}^{M} \alpha_l(t-1) P(u_t, s_t = k \mid s_{t-1} = l) \\
= & \sum_{l=1}^{M} \alpha_l(t-1) P(u_t \mid s_t = k, s_{t-1} = l) \cdot P(s_t = k \mid s_{t-1} = l) \\
= & \sum_{l=1}^{M} \alpha_l(t-1) P(u_t \mid s_t = k) P(s_t = k \mid s_{t-1} = l) \\
= & \sum_{l=1}^{M} \alpha_l(t-1) b_k(u_t) a_{l,k}
\end{aligned}
$$

The fourth equality comes from the fact given s_{t-1}, s_t is independent of all s_τ, $\tau = 1, 2, \ldots, t-2$ and hence u_τ, $\tau = 1, \ldots, t-2$. Also s_t is independent of u_{t-1} since s_{t-1} is given.

- Define the backward probability $\beta_k(t)$ as the conditional probability of observing the vectors after time t, u_τ, $\tau = t+1, ..., T$, given the state at time t is k.

$$\begin{aligned} \beta_k(t) &= P(u_{t+1}, ..., u_T \mid s_t = k)\ , 1 \leq t \leq T-1 \\ & \text{Set } \ \beta_k(T) = 1\ , \quad \text{for all } k\ . \end{aligned}$$

- As with the forward probability, the backward probability can be evaluated using the following recursion

$$\begin{aligned} \beta_k(T) &= 1 \\ \beta_k(t) &= \sum_{l=1}^{M} a_{k,l} b_l(u_{t+1}) \beta_l(t+1) \quad 1 \leq t < T\ . \end{aligned}$$

- Proof:

$$\begin{aligned}
\beta_k(t) &= P(u_{t+1}, \ldots, u_T \mid s_t = k) \\
&= \sum_{l=1}^{M} P(u_{t+1}, \ldots, u_T, s_{t+1} = l \mid s_t = k) \\
&= \sum_{l=1}^{M} P(s_{t+1} = l \mid s_t = k) P(u_{t+1}, \ldots, u_T \mid s_{t+1} = l, s_t = k) \\
&= \sum_{l=1}^{M} a_{k,l} P(u_{t+1}, \ldots, u_T \mid s_{t+1} = l) \\
&= \sum_{l=1}^{M} a_{k,l} P(u_{t+1} \mid s_{t+1} = l) P(u_{t+2}, \ldots, u_T \mid s_{t+1} = l, u_{t+1}) \\
&= \sum_{l=1}^{M} a_{k,l} P(u_{t+1} \mid s_{t+1} = l) P(u_{t+2}, \ldots, u_T \mid s_{t+1} = l) \\
&= \sum_{l=1}^{M} a_{k,l} b_l(u_{t+1}) \beta_l(t+1)
\end{aligned}$$

- The probabilities $L_k(t)$ and $H_{k,l}(t)$ are solved by

$$
\begin{aligned}
L_k(t) &= P(s_t = k \mid \mathbf{u}) = \frac{P(\mathbf{u}, s_t = k)}{P(\mathbf{u})} \\
&= \frac{1}{P(\mathbf{u})} \alpha_k(t)\beta_k(t)
\end{aligned}
$$

$$
\begin{aligned}
H_{k,l}(t) &= P(s_t = k, s_{t+1} = l \mid \mathbf{u}) \\
&= \frac{P(\mathbf{u}, s_t = k, s_{t+1} = l)}{P(\mathbf{u})} \\
&= \frac{1}{P(\mathbf{u})} \alpha_k(t) a_{k,l} b_l(u_{t+1}) \beta_l(t+1) .
\end{aligned}
$$

- Proof for $L_k(t)$:

$$\begin{aligned} & P(\mathbf{u}, s_t = k) = P(u_1, ..., u_t, ..., u_T, s_t = k) \\ = \; & P(u_1, ..., u_t, s_t = k) P(u_{t+1}, ..., u_T \mid s_t = k, u_1, ..., u_t) \\ = \; & \alpha_k(t) P(u_{t+1}, ..., u_T \mid s_t = k) \\ = \; & \alpha_k(t) \beta_k(t) \end{aligned}$$

- Proof for $H_{k,l}(t)$:

$$
\begin{aligned}
& P(\mathbf{u}, s_t = k, s_{t+1} = l) \\
= \; & P(u_1, ..., u_t, ..., u_T, s_t = k, s_{t+1} = l) \\
= \; & P(u_1, ..., u_t, s_t = k) \cdot \\
& P(u_{t+1}, s_{t+1} = l \mid s_t = k, u_1, ..., u_t) \cdot \\
& P(u_{t+2}, ..., u_T \mid s_{t+1} = l, s_t = k, u_1, ..., u_{t+1}) \\
= \; & \alpha_k(t) P(u_{t+1}, s_{t+1} = l \mid s_t = k) \cdot \\
& P(u_{t+2}, ..., u_T \mid s_{t+1} = l) \\
= \; & \alpha_k(t) P(s_{t+1} = l \mid s_t = k) \cdot \\
& P(u_{t+1} \mid s_{t+1} = l, s_t = k) \beta_l(t+1) \\
= \; & \alpha_k(t) a_{k,l} P(u_{t+1} \mid s_{t+1} = l) \beta_l(t+1) \\
= \; & \alpha_k(t) a_{k,l} b_l(u_{t+1}) \beta_l(t+1)
\end{aligned}
$$

- Note that the amount of computation for $L_k(t)$ and $H_{k,l}(t)$, $k, l = 1, ..., M$, $t = 1, ..., T$ is at the order of $M^2 T$.
- Note:

$$P(\mathbf{u}) = \sum_{k=1}^{M} \alpha_k(t)\beta_k(t)\,, \quad \text{for any } t$$

- In particular, if we let $t = T$,

$$P(\mathbf{u}) = \sum_{k=1}^{M} \alpha_k(T)\beta_k(T) = \sum_{k=1}^{M} \alpha_k(T)\,.$$

Proof:

$$
\begin{aligned}
P(\mathbf{u}) &= P(u_1, \ldots, u_t, \ldots, u_T) \\
&= \sum_{k=1}^{M} P(u_1, \ldots, u_t, \ldots, u_T, s_t = k) \\
&= \sum_{k=1}^{M} P(u_1, \ldots, u_t, s_t = k) P(u_{t+1}, \ldots, u_T \mid s_t, u_1, \ldots, u_t) \\
&= \sum_{k=1}^{M} \alpha_k(t) P(u_{t+1}, \ldots, u_T \mid s_t) \\
&= \sum_{k=1}^{M} \alpha_k(t) \beta_k(t)
\end{aligned}
$$

The Estimation Algorithm

The estimation algorithm iterates the following steps:

- Compute the forward and backward probabilities $\alpha_k(t)$, $\beta_k(t)$, $k = 1, ..., M$, $t = 1, ..., T$ under the current set of parameters.

$$\alpha_k(1) = \pi_k b_k(u_1) \quad 1 \leq k \leq M$$

$$\alpha_k(t) = b_k(u_t) \sum_{l=1}^{M} \alpha_l(t-1) a_{l,k} \ , \quad 1 < t \leq T, \ 1 \leq k \leq M \ .$$

$$\beta_k(T) = 1$$

$$\beta_k(t) = \sum_{l=1}^{M} a_{k,l} b_l(u_{t+1}) \beta_l(t+1) \quad 1 \leq t < T \ .$$

- Compute $L_k(t)$, $H_{k,l}(t)$ using $\alpha_k(t)$, $\beta_k(t)$. Let $P(\mathbf{u}) = \sum_{k=1}^{M} \alpha_k(1)\beta_k(1)$.

$$L_k(t) = \frac{1}{P(\mathbf{u})}\alpha_k(t)\beta_k(t)$$

$$H_{k,l}(t) = \frac{1}{P(\mathbf{u})}\alpha_k(t)a_{k,l}b_l(u_{t+1})\beta_l(t+1) .$$

- Update the parameters using $L_k(t)$, $H_{k,l}(t)$.

$$\mu_k = \frac{\sum_{t=1}^{T} L_k(t) u_t}{\sum_{t=1}^{T} L_k(t)}$$

$$\Sigma_k = \frac{\sum_{t=1}^{T} L_k(t)(u_t - \mu_k)(u_t - \mu_k)^t}{\sum_{t=1}^{T} L_k(t)}$$

$$a_{k,l} = \frac{\sum_{t=1}^{T-1} H_{k,l}(t)}{\sum_{t=1}^{T-1} L_k(t)} .$$

Multiple Sequences

- If we estimate an HMM using multiple sequences, the previous estimation algorithm can be extended naturally.
- For brevity, let's assume all the sequences are of length T. Denote the ith sequence by $\mathbf{u}_i = \{u_{i,1}, u_{i,2}, ..., u_{i,T}\}$, $i = 1, ..., N$.
- In each iteration, we compute the forward and backward probabilities for each sequence separately in the same way as previously described.
- Compute $L_k(t)$ and $H_{k,l}(t)$ separately for each sequence, also in the same way as previously described.
- Update parameters similarly.

- Compute the forward and backward probabilities $\alpha_k^{(i)}(t)$, $\beta_k^{(i)}(t)$, $k = 1, ..., M$, $t = 1, ..., T$, $i = 1, ..., N$, under the current set of parameters.

$$\alpha_k^{(i)}(1) = \pi_k b_k(u_{i,1}), 1 \le k \le M, \ 1 \le i \le N .$$

$$\alpha_k^{(i)}(t) = b_k(u_{i,t}) \sum_{l=1}^{M} \alpha_l^{(i)}(t-1) a_{l,k} , \quad 1 < t \le T, \ 1 \le k \le M, \ 1 \le i \le N .$$

$$\beta_k^{(i)}(T) = 1 , 1 \le k \le M, \ 1 \le i \le N$$

$$\beta_k^{(i)}(t) = \sum_{l=1}^{M} a_{k,l} b_l(u_{i,t+1}) \beta_l^{(i)}(t+1) \quad 1 \le t < T, \ 1 \le k \le M, \ 1 \le i \le N .$$

- Compute $L_k^{(i)}(t)$, $H_{k,l}^{(i)}(t)$ using $\alpha_k^{(i)}(t)$, $\beta_k^{(i)}(t)$. Let $P(\mathbf{u}_i) = \sum_{k=1}^{M} \alpha_k^{(i)}(1)\beta_k^{(i)}(1)$.

$$
\begin{aligned}
L_k^{(i)}(t) &= \frac{1}{P(\mathbf{u}_i)} \alpha_k^{(i)}(t)\beta_k^{(i)}(t) \\
H_{k,l}^{(i)}(t) &= \frac{1}{P(\mathbf{u}_i)} \alpha_k^{(i)}(t) a_{k,l} b_l(u_{i,t+1})\beta_l^{(i)}(t+1) \,.
\end{aligned}
$$

- Update the parameters using $L_k(t)$, $H_{k,l}(t)$.

$$\mu_k = \frac{\sum_{i=1}^{N} \sum_{t=1}^{T} L_k^{(i)}(t) u_{i,t}}{\sum_{i=1}^{N} \sum_{t=1}^{T} L_k^{(i)}(t)}$$

$$\Sigma_k = \frac{\sum_{i=1}^{N} \sum_{t=1}^{T} L_k^{(i)}(t)(u_{i,t} - \mu_k)(u_{i,t} - \mu_k)^t}{\sum_{i=1}^{N} \sum_{t=1}^{T} L_k^{(i)}(t)}$$

$$a_{k,l} = \frac{\sum_{i=1}^{N} \sum_{t=1}^{T-1} H_{k,l}^{(i)}(t)}{\sum_{i=1}^{N} \sum_{t=1}^{T-1} L_k^{(i)}(t)} .$$

HMM with Discrete Data

- Given a state k, the distribution of the data U is discrete, specified by a pmf.
- Assume $U \in \mathcal{U} = \{1, 2, ...J\}$. Denote $b_k(j) = q_{k,j}$, $j = 1, ..., J$.
- Parameters in the HMM: $a_{k,l}$ and $q_{k,j}$, $k, l = 1, ..., M$, $j = 1, ..., J$.

- Model estimation by the following iteration:
  - Compute the forward and backward probabilities $\alpha_k(t)$, $\beta_k(t)$. Note that $b_k(u_t) = q_{k,u_t}$.
  - Compute $L_k(t)$, $H_{k,l}(t)$ using $\alpha_k(t)$, $\beta_k(t)$.
  - Update the parameters as follows:

$$a_{k,l} = \frac{\sum_{t=1}^{T-1} H_{k,l}(t)}{\sum_{t=1}^{T-1} L_k(t)}, \quad k, l = 1, ..., M$$

$$q_{k,j} = \frac{\sum_{t=1}^{T} L_k(t) I(u_t = j)}{\sum_{t=1}^{T} L_k(t)}, \quad k = 1, ..., M; \quad j = 1, ..., J$$

Viterbi Algorithm

- In many applications using HMM, we need to predict the state sequence $\mathbf{s} = \{s_1, ..., s_T\}$ based on the observed data $\mathbf{u} = \{u_1, ..., u_T\}$.
- Optimization criterion: find $\mathbf{s}$ that maximizes $P(\mathbf{s} \mid \mathbf{u})$:

$$\mathbf{s}^* = \arg\max_{\mathbf{s}} P(\mathbf{s} \mid \mathbf{u}) = \arg\max_{\mathbf{s}} \frac{P(\mathbf{s}, \mathbf{u})}{P(\mathbf{u})} = \arg\max_{\mathbf{s}} P(\mathbf{s}, \mathbf{u})$$

- This criterion is called the rule of *Maximum A Posteriori* (MAP).
- The optimal sequence $\{s_1, s_2, ..., s_T\}$ can be found by the Viterbi algorithm.
- The amount of computation in the Viterbi algorithm is at the order of M^2T. Memory required is at the order of MT.

- The Viterbi algorithm maximizes an objective function $G(\mathbf{s})$, where $\mathbf{s} = \{s_1, ..., s_T\}$, $s_t \in \{1, ..., M\}$, is a state sequence and $G(\mathbf{s})$ has a special property.
- Brute-force optimization of $G(\mathbf{s})$ involves an exhaustive search of all the M^T possible sequences.
- Property of $G(\mathbf{s})$:

$$G(\mathbf{s}) = g_1(s_1) + g_2(s_2, s_1) + g_3(s_3, s_2) + \cdots + g_T(s_T, s_{T-1})$$

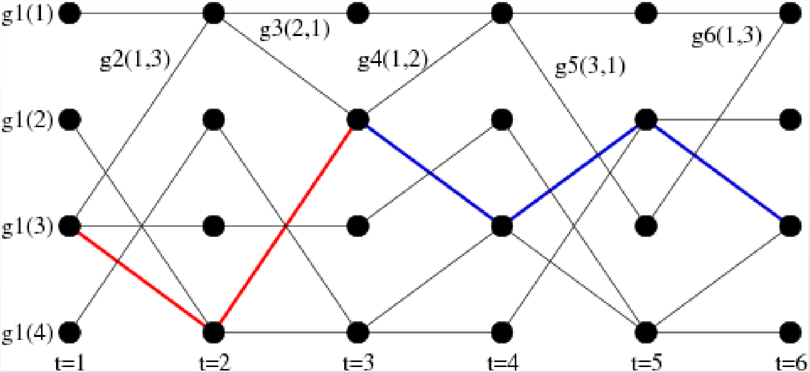
- The key is the objective function can be written as a sum of “merit” functions depending on one state and its preceding one.

- A Markovian kind of property:
  - Suppose in the optimal state sequence $\mathbf{s}^*$, the tth position $s_t^* = k$. To maximize $G(s_1, s_2, ..., s_T)$, we can maximize the following two functions separately:

$$G_{t,k}(s_1, ..., s_{t-1}) = g_1(s_1) + g_2(s_2, s_1) + \cdots + g_t(k, s_{t-1})$$
$$\bar{G}_{t,k}(s_{t+1}, ..., s_T) = g_{t+1}(s_{t+1}, k) + \cdots + g_T(s_T, s_{T-1})$$

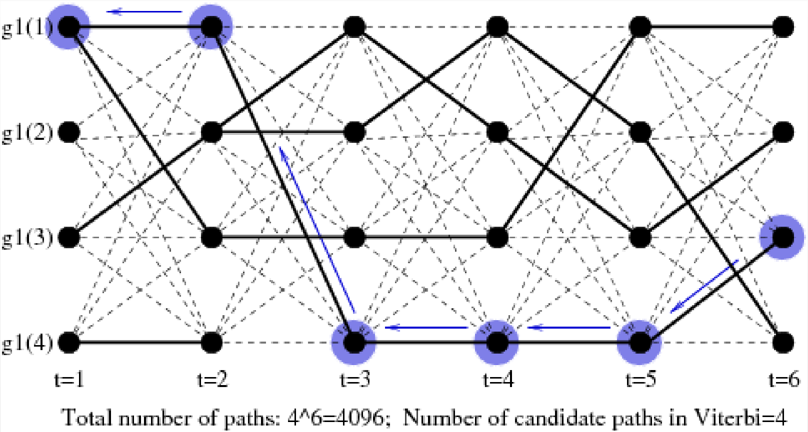
    The first function involves only states before t; and the second only states after t.
  - Also note the recursion of $G_{t,k}(s_1, ..., s_{t-1})$:

$$G_{t,l}(s_1, ..., s_{t-2}, k) = G_{t-1,k}(s_1, ..., s_{t-2}) + g_t(l, k) \ .$$


g1(1)
g1(2)
g1(3)
g1(4)
g2(1,3)
g3(2,1)
g4(1,2)
g5(3,1)
g6(1,3)
t=1
t=2
t=3
t=4
t=5
t=6

- Every state sequence $\mathbf{s}$ corresponds to a path from $t = 1$ to $t = T$.
- We put weight $g_t(k, l)$ on the link from state l at $t - 1$ to state k at t.
- At the starting node, we put weight $g_1(k)$ for state k.
- $G(\mathbf{s})$ is the sum of the weights on the links in path $\mathbf{s}$.
- In the figure, suppose the colored path is the optimal one. At $t = 3$, this path passes through state 2. Then the sub-path before $t = 3$ should be the best among all paths from $t = 1$ to $t = 3$ that end at state 2. The sub-path after $t = 3$ should be the best among all paths from $t = 3$ to $t = 6$ that start at state 2.

How the Viterbi Algorithm Works (Pseudocode)



Pseudocode

- At $t = 1$, for each node (state) $k = 1, ..., M$, record $G^*_{1,k} = g_1(k)$.
- At $t = 2$, for each node $k = 1, ..., M$, only need to record which node is the best preceding one. Suppose node k is linked to node l^* at $t = 1$, record l^* and $G^*_{2,k} = \max_{l=1,2,...,M}[G^*_{1,l} + g_2(k, l)] = G^*_{1,l^*} + g_2(k, l^*)$.
- The same procedure is applied successively for $t = 2, 3, ..., T$. At every node, link it to its best preceding one. Set $G^*_{t,k} = \max_{l=1,2,...,M}[G^*_{t-1,l} + g_t(k, l)] = G^*_{t-1,l^*} + g_t(k, l^*)$. $G^*_{t,k}$ is the sum of weights of the best path up to t and with the end tied at state k and l^* is the best preceding state. Record l^* and $G^*_{t,k}$.
- At the end, only M paths are formed, each ending with a different state at $t = T$. The objective function for a path ending at node k is $G^*_{T,k}$. Pick k^* that maximizes $G^*_{T,k}$. Trace the path backwards from the last state k^*.

Proof for the Viterbi Algorithm

Notation:

- Let $\mathbf{s}^*(t,k)$ be the sequence $\{s_1,...,s_{t-1}\}$ that maximizes $G_{t,k}(s_1,...,s_{t-1})$:

$$\mathbf{s}^*(t,k) = \arg\max_{s_1,...,s_{t-1}} G_{t,k}(s_1,...,s_{t-1})$$

  Let $G^*_{t,k} = \max_{s_1,...,s_{t-1}} G_{t,k}(s_1,...,s_{t-1})$.
- Let $\bar{\mathbf{s}}^*(t,k)$ be the sequence $\{s_{t+1},...,s_T\}$ that maximizes $\bar{G}_{t,k}(s_{t+1},...,s_T)$:

$$\bar{\mathbf{s}}^*(t,k) = \arg\max_{s_{t+1},...,s_T} \bar{G}_{t,k}(s_{t+1},...,s_T)$$

Key facts for proving the Viterbi algorithm:

- If the optimal state sequence $\mathbf{s}^*$ has the last state $s^*_T = k$, then the subsequence of $\mathbf{s}^*$ from 1 to $T-1$ should be $\mathbf{s}^*(T,k)$ and

$$\max_{\mathbf{s}} G(\mathbf{s}) = G_{T,k}(\mathbf{s}^*(T,k)) \; .$$

- Since we don't know what should be s^*_T, we should compare all the possible states $k = 1, ..., M$:

$$\max_{\mathbf{s}} G(\mathbf{s}) = \max_{k} G_{T,k}(\mathbf{s}^*(T,k)) \; .$$

- $G_{t,k}(\mathbf{s}^*(t,k))$ and $\mathbf{s}^*(t,k)$ can be obtained recursively for $t = 1, ..., T$.

Proof for the recursion:

- Suppose $G_{t-1,k}(\mathbf{s}^*(t-1,k))$ and $\mathbf{s}^*(t-1,k)$ for $k = 1, ..., M$ have been obtained. For any $l = 1, ..., M$:

$$
\begin{aligned}
G_{t,l}(\mathbf{s}^*(t,l)) &= \max_{s_1,\ldots,s_{t-1}} G_{t,l}(s_1, ..., s_{t-1}) \\
&= \max_k \max_{s_1,\ldots,s_{t-2}} G_{t,l}(s_1, ..., s_{t-2}, k) \\
&= \max_k \max_{s_1,\ldots,s_{t-2}} (G_{t-1,k}(s_1, ..., s_{t-2}) + g_t(l,k)) \\
&= \max_k (g_t(l,k) + \max_{s_1,\ldots,s_{t-2}} G_{t-1,k}(s_1, ..., s_{t-2})) \\
&= \max_k (g_t(l,k) + G_{t-1,k}(\mathbf{s}^*(t-1,k)))
\end{aligned}
$$

- Suppose k^* achieves the maximum, that is, $k^* = \arg\max_k (g_t(l,k) + G_{t-1,k}(\mathbf{s}^*(t-1,k)))$. Then $\mathbf{s}^*(t,l) = \{s^*(t-1,k^*), k^*\}$, that is, for $\mathbf{s}^*(t,l)$, the last state $s^*_{t-1} = k^*$ and the subsequence from position 1 to $t-2$ is $\mathbf{s}^*(t-1,k^*)$.
- The amount of computation involved in deciding $G_{t,l}(\mathbf{s}^*(t,l))$ and $\mathbf{s}^*(t,l)$ for all $l = 1, ..., M$ is at the order of M^2. For each l, we have to exhaust M possible k's to find k^*.
- To start the recursion, we have

$$G_{1,k}(\cdot) = g_1(k)\,,\ \mathbf{s}^*(1,k) = \{\}\ .$$

Note: at t=1, there is no preceding state.

Optimal State Sequence for HMM

- We want to find the optimal state sequence $\mathbf{s}^*$:

$$\mathbf{s}^* = \arg\max_{\mathbf{s}} P(\mathbf{s}, \mathbf{u}) = \arg\max_{\mathbf{s}} \log P(\mathbf{s}, \mathbf{u})$$

- The objective function:

$$\begin{aligned} G(\mathbf{s}) &= \log P(\mathbf{s}, \mathbf{u}) = \log[\pi_{s_1} b_{s_1}(u_1) a_{s_1,s_2} b_{s_2}(u_2) \cdots a_{s_{T-1},s_T} b_{s_T}(u_T)] \\ &= [\log \pi_{s_1} + \log b_{s_1}(u_1)] + [\log a_{s_1,s_2} + \log b_{s_2}(u_2)] + \\ &\quad \cdots + [\log a_{s_{T-1},s_T} + \log b_{s_T}(u_T)] \end{aligned}$$

If we define

$$\begin{aligned} g_1(s_1) &= \log \pi_{s_1} + \log b_{s_1}(u_1) \\ g_t(s_t, s_{t-1}) &= \log a_{s_t,s_{t-1}} + \log b_{s_t}(u_t) , \end{aligned}$$

then $G(\mathbf{s}) = g_1(s_1) + \sum_{t=2}^{T} g_t(s_t, s_{t-1})$. Hence, the Viterbi algorithm can be applied.

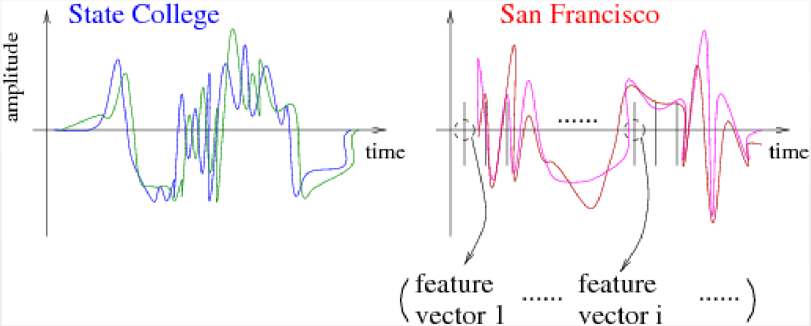
Viterbi Training

- Viterbi training to HMM resembles the classification EM estimation to a mixture model.
- Replace "soft" classification reflected by $L_k(t)$ and $H_{k,l}(t)$ by "hard" classification.
- In particular:
  - Replace the step of computing forward and backward probabilities by selecting the optimal state sequence $\mathbf{s}^*$ under the current parameters using the Viterbi algorithm.
  - Let $L_k(t) = I(s_t^* = k)$, i.e., $L_k(t)$ equals 1 when the optimal state sequence is in state k at t; and zero otherwise. Similarly, let $H_{k,l}(t) = I(s_{t-1} = k)I(s_t = l)$.
  - Update parameters using $L_k(t)$ and $H_{k,l}(t)$ and the same formulas.

Applications

Speech recognition:

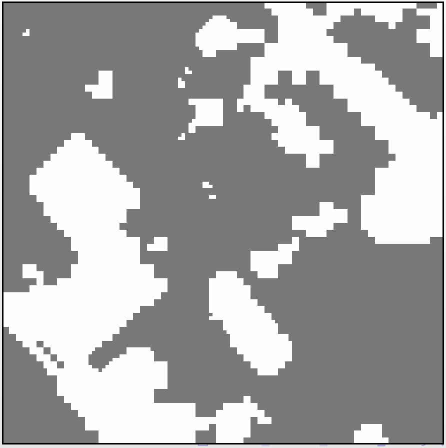
- Goal: identify words spoken according to speech signals
  - Automatic voice recognition systems used by airline companies
  - Automatic stock price reporting
- Raw data: voice amplitude sampled at discrete time spots (a time sequence).
- Input data: speech feature vectors computed at the sampling time.


State College
amplitude
time
San Francisco
......
time
feature vector 1
......
feature vector i
......

- Methodology:
  - Estimate an Hidden Markov Model (HMM) for each word, e.g., `State College`, `San Francisco`, `Pittsburgh`. The training provides a dictionary of models $\{\mathcal{W}_1, \mathcal{W}_2, ...\}$.
  - For a new word, find the HMM that yields the maximum likelihood. Denote the sequence of feature vectors extracted for this voice signal by $\mathbf{u} = \{u_1, ..., u_T\}$. Classify to word i^* if $\mathcal{W}_{i^*}$ maximizes $P(\mathbf{u} \mid \mathcal{W}_i)$.
  - Recall that $P(\mathbf{u}) = \sum_{k=1}^{M} \alpha_k(T)$, where $\alpha_k(T)$ are the forward probabilities at $t = T$, computed using parameters specified by $\mathcal{W}_{i^*}$.
- In the above example, HMM is used for "profiling". Similar ideas have been applied to genomics sequence analysis, e.g., profiling families of protein sequences by HMMs.

Supervised learning:

- Use image classification as an example.
- The image is segmented into man-made and natural regions.



- Training data: the original images and their manually labeled segmentation.
- Associate each block in the image with a class label. A block is an element for the interest of learning.
- At each block, compute a feature vector that is anticipated to reflect the difference between the two classes (man-made vs. natural).
- For the purpose of classification, each image is an array of feature vectors, whose true classes are known in training.

- If we ignore the spatial dependence among the blocks, an image becomes a collection of independent samples $\{u_1, u_2, ..., u_T\}$. For training data, we know the true classes $\{z_1, ..., z_T\}$. Any classification algorithm can be applied.
- Mixture discriminant analysis: model each class by a mixture model.
- What if we want to take spatial dependence into consideration?
  - Use a hidden Markov model! A 2-D HMM would be even better.
  - Assume each class contains several states. The underlying states follow a Markov chain. We need to scan the image in a certain way, say row by row or zig-zag.
  - This HMM is an extension of mixture discriminant analysis with spatial dependence taken into consideration.

- Details:
  - Suppose we have M states, each belonging to a certain class. Use $C(k)$ to denote the class state k belongs to. If a block is in a certain class, it can only exist in one of the states that belong to its class.
  - Train the HMM using the feature vectors $\{u_1, u_2, ..., u_T\}$ and their classes $\{z_1, z_2, ..., z_T\}$. There are some minor modifications from the training algorithm described before since no class labels are involved there.
  - For a test image, find the optimal sequence of states $\{s_1, s_2, ..., s_T\}$ with maximum a posteriori probability (MAP) using the Viterbi algorithm.
  - Map the state sequence into classes: $\hat{z}_t = C(s_t^*)$.

Unsupervised learning:

- Since a mixture model can be used for clustering, HMM can be used for the same purpose. The difference lies in the fact HMM takes spatial dependence into consideration.
- For a given number of states, fit an HMM to a sequential data.
- Find the optimal state sequence $\mathbf{s}^*$ by the Viterbi algorithm.
- Each state represents a cluster.
- Examples: image segmentation, etc.