Clustering: Mixture Models and HMMs

Hong Chang

Institute of Computing Technology,
Chinese Academy of Sciences

Machine Learning (Spring 2015)
Outline I

1. Introduction
2. k-means
3. Gaussian Mixture Models and EM Algorithm
   - Gaussian Mixture
   - EM Algorithm
   - EM for Gaussian Mixture
4. Hidden Markov Models
   - The Model
   - Three Fundamental Problems
   - Applications
5. Hierarchical Clustering
Different Approaches to Density Estimation

- **Parametric:** $p(x|C_i)$ is represented by a single parametric model.
- **Semiparametric:** $p(x|C_i)$ is represented by a mixture of densities.
- **Nonparametric:** $p(x|C_i)$ cannot be represented by a single parametric model or a mixture model; the data speaks for itself.
- The model flexibility increases (and hence the model bias decreases) from parametric to semiparametric to nonparametric methods.
Mixture Densities

- Mixture density:
  \[ p(x) = \sum_{j=1}^{k} p(x|G_j)P(G_j) \]

  where
  - \( G_j \): mixture components (or clusters or groups)
  - \( p(x|G_j) \): component densities
  - \( P(G_j) \): mixing proportions (or priors)

- Gaussian mixture:
  \[ p(x|G_j) = \mathcal{N}(\mu_j, \Sigma_j) \]

  parameters: \( \Phi = \{ P(G_j), \mu_j, \Sigma_j \}_{j=1}^{k} \)
### Classes vs. Clusters

<table>
<thead>
<tr>
<th>Supervised</th>
<th>Unsupervised</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample $\mathcal{X} = {x^{(i)}, y^{(i)}}_{i=1}^N$</td>
<td>Sample $\mathcal{X} = {x^{(i)}}_{i=1}^N$</td>
</tr>
<tr>
<td><strong>Classes</strong> $C_j, j = 1, \ldots, K$</td>
<td><strong>Clusters</strong> $G_j, j = 1, \ldots, k$</td>
</tr>
<tr>
<td>$p(x) = \sum_{j=1}^K p(x</td>
<td>C_j)P(C_j)$</td>
</tr>
<tr>
<td>where $p(x</td>
<td>C_j) = \mathcal{N}(\mu_j, \Sigma_j)$</td>
</tr>
<tr>
<td><strong>Parameters</strong> $\Phi = {P(C_j), \mu_j, \Sigma_j}_{j=1}^K$</td>
<td><strong>Parameters</strong> $\Phi = {P(G_j), \mu_j, \Sigma_j}_{j=1}^k$</td>
</tr>
<tr>
<td>$\hat{P}(C_j) = \frac{\sum_i y^{(i)}_j}{N}$</td>
<td>$P(G_j)$</td>
</tr>
<tr>
<td>$m_j = \frac{\sum_i y^{(i)}_j x^{(i)}}{\sum_i y^{(i)}_j}$</td>
<td>$\mu_j, \Sigma_j$</td>
</tr>
<tr>
<td>$S_j = \frac{\sum_i y^{(i)}_j (x^{(i)} - m_j)(x^{(i)} - m_j)^T}{\sum_i y^{(i)}_j}$</td>
<td>(mixture density estimation)</td>
</tr>
</tbody>
</table>

(mixture density estimation)
A variable can be *unobserved (latent)*.

- It is an imaginary quantity meant to provide some simplified and abstractive view of the data generation process. E.g., speech recognition models, mixture models
- It is a real-world object and/or phenomena, but difficult or impossible to measure. E.g., the temperature of a star, causes of a disease, evolutionary ancestors
- It is a real-world object and/or phenomena, but sometimes was not measured, because of faulty sensors; or was measure with a noisy channel, etc. E.g., traffic radio, aircraft signal on a radar screen

*Discrete* latent variables can be used to *partition/cluster data* into sub-groups (mixture models, HMM in this lecture).

*Continuous* latent variables can be used for *dimensionality reduction* (later lecture).
Example of clustering problem: color quantization from high to lower resolution (special case of vector quantization).

**Problem formulation:**
- Given a sample set $\mathcal{X} = \{x^{(i)}\}_{i=1}^{N}$.
- Find $k$ reference vectors (or prototypes or codebook vectors or codewords) $\mathbf{m}_j$ ($j = 1, \ldots, k$) which best represent the data.

**Encoding/decoding view:**
- Encoding: from a data point $x^{(i)}$ to the index $l$ of a reference vector.
- Decoding: from an index $l$ to the corresponding reference vector $\mathbf{m}_l$. 
Encoding/Decoding

- Each data points $x^{(i)}$ is represented by the index $l$ of the nearest reference vector:
  \[ l = \arg \min_j \| x^{(i)} - m_j \| \]

- Encoding can lead to data compression: instead of storing (or transmitting) $x^{(i)}$, we only need to store (or transmit) $l$.

- Since $x^{(i)}$ is represented by $m_l$ after encoding and then decoding, reconstruction error is incurred.

- Total reconstruction error:
  \[
  E(\{m_i\}_{i=1}^k | \mathcal{X}) = \sum_i \sum_l b_l^{(i)} \| x^{(i)} - m_l \|^2
  \]

  where
  \[
  b_l^{(i)} = \begin{cases} 
  1 & \text{if } l = \arg \min_j \| x^{(i)} - m_j \| \\
  0 & \text{otherwise}
  \end{cases}
  \]
The best reference vectors are those that minimize the total reconstruction error, so this corresponds to an optimization problem. However, since $b_l^{(i)}$ also depends on $\mathbf{m}_l$, the optimization problem cannot be solved analytically, but iteratively. The \textit{k-means clustering algorithm} is an iterative algorithm for solving the optimization problem.
**$k$-Means Algorithm**

Initialize $m_l, l = 1, \ldots, k$ (e.g., $k$ randomly selected $x^{(i)}$)

Repeat

For all $x^{(i)} \in \mathcal{X}$

$$b^{(i)}_l = \begin{cases} 1 & \text{if } l = \arg \min_j \|x^{(i)} - m_j\| \\ 0 & \text{otherwise} \end{cases}$$

For all $m_l, l = 1, \ldots, k$

$$m_l = \frac{\sum_i b^{(i)}_l x^{(i)}}{\sum_i b^{(i)}_l}$$

Until $m_l$ converge.
Evolution of $k$-Means

(a) original dataset; (b) random initialization; (c-f) illustration of running two iterations of $k$-means. (Images from Michael Jordan)
Convergence of $k$-Means

- $k$-means is exactly *coordinate descent* on the reconstruction error $E$.
- $E$ monotonically decreases, and the value of $E$ converges, so do the clustering results.
- It is possible for $k$-means to oscillate between a few different clusterings, but this almost never happens in practice.
- $E$ is non-convex, so coordinate descent on $E$ cannot guaranteed to converge to global minimum. One common thing to do is running $k$-means many times and pick the best one.
Gaussian Mixture Models

Consider a mixture of $k$ Gaussian components:

- The latent class indicator vector $\mathbf{z}^{(i)} = (z_1^{(i)}, \ldots, z_k^{(i)})^T$:

$$z_l^{(i)} \begin{cases} 1 & \text{if } \mathbf{x}^{(i)} \text{ belongs to cluster } G_l \\ 0 & \text{otherwise} \end{cases}$$

- The likelihood of a sample $\mathbf{x}$:

$$p(\mathbf{x}|\Phi) = \sum_{l=1}^{k} p(z_l = 1|\Phi)p(\mathbf{x}|z_l = 1, \Phi) = \sum_l \pi_l \mathcal{N}(\mathbf{x}|\mu_l, \Sigma_l)$$

- Gaussian component densities: $p(\mathbf{x}|G_l) = \mathcal{N}(\mu_l, \Sigma_l)$, parameter $\Theta_l = \{\mu_l, \Sigma_l\}$
- The prior probability: $P(G_l) = \pi_l$
The Learning is Hard

- In fully observed IID settings, the log likelihood decomposes into a sum of local terms.

\[ \mathcal{L}(\Phi | \mathcal{X}, \mathcal{Z}) = \sum_i \log p(x^{(i)}, z^{(i)} | \Phi) = \sum_i (\log p(z^{(i)} | \Phi_z) + \log p(x^{(i)} | z^{(i)}, \Phi_x)) \]

- With latent variables, all the parameters become coupled together via marginalization.

\[ \mathcal{L}(\Phi | \mathcal{X}) = \sum_i \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)} | \Phi) \]
Expectation-Maximization Algorithm

- The **expectation-maximization (EM)** algorithm may be regarded as a probabilistic extension of k-means, although historically it was not derived as such.
- EM finds the component density parameters that maximize the likelihood with respect to a mixture model.
- **Log likelihood:**

\[
\mathcal{L}(\Phi|\mathcal{X}) = \log \prod_i p(x^{(i)}|\Phi) = \sum_i \log \sum_{l=1}^k p(x^{(i)}|G_l) P(G_l)
\]

where the parameters \(\Phi\) include the priors \(P(G_l)\) and the sufficient statistics of the component densities \(p(x^{(i)}|G_l)\).

- **Optimization** of \(\mathcal{L}(\Phi|\mathcal{X})\) w.r.t. \(\Phi\) cannot be solved analytically.

The expectation-maximization (EM) algorithm may be regarded as a probabilistic extension of k-means, although historically it was not derived as such. EM finds the component density parameters that maximize the likelihood with respect to a mixture model. Log likelihood:

\[
\log \prod_i p(x^{(i)}|\Phi) = \sum_i \log \sum_{l=1}^k p(x^{(i)}|G_l) P(G_l)
\]

where the parameters \(\Phi\) include the priors \(P(G_l)\) and the sufficient statistics of the component densities \(p(x^{(i)}|G_l)\). Optimization of \(\mathcal{L}(\Phi|\mathcal{X})\) w.r.t. \(\Phi\) cannot be solved analytically.
EM is an iterative algorithm for solving the maximum likelihood estimation (MLE) problem.
EM is suitable for problems that involves two sets of random variables: observation variables $X$ and hidden variables $Z$.
Goal: maximize the likelihood $\mathcal{L}(\Phi|\mathcal{X})$ given $\mathcal{X}$.
EM is suitable if it is difficult to maximizing the incomplete-data likelihood $\mathcal{L}(\Phi|\mathcal{X})$ directly but it is easier to work with the complete-data likelihood $\mathcal{L}_C(\Phi|\mathcal{X}, Z)$.
Because the $Z$ value are not observed, we cannot work directly with $\mathcal{L}_C(\Phi|\mathcal{X}, Z)$. Instead, we work with an auxiliary function which is the expectation of the complete-data likelihood given $\mathcal{X}$ and the current (iteration $t$) parameter values $\Phi^t$:

$$Q(\Phi|\Phi^t) = E[\mathcal{L}_C(\Phi|\mathcal{X}, Z)|\mathcal{X}, \Phi^t]$$
EM iterates between two steps:

- **E-step**: evaluation of expectation
  \[
  Q(\Phi|\Phi^t) = E[\mathcal{L}_C(\Phi|\mathcal{X}, \mathcal{Z})|\mathcal{X}, \Phi^t]
  \]

- **M-step**: maximization of expectation
  \[
  \Phi^{t+1} = \arg \max \ Q(\Phi|\Phi^t)
  \]

- \(k\)-means can be used to initialize EM.
- An increase in \(Q\) implies an increase in the incomplete-data likelihood:
  \[
  \mathcal{L}(\Phi^{t+1}|\mathcal{X}) \geq \mathcal{L}(\Phi^t|\mathcal{X})
  \]

- EM finds a local maximum of the likelihood.
Jensen’s Inequality

**Theorem**

Let $f$ be a convex function and $X$ a random variable. Then,

$$E[f(X)] \geq f(E[X]).$$

Moreover, if $f$ is strictly convex, then $E[f(X)] = f(E[X])$ holds true if and only if $X = E[X]$ with probability 1 (i.e., $X$ is a constant).

- Jensen’s inequality also holds for concave functions $f$, but with reversed inequalities, i.e., $E[f(X)] \leq f(E[X])$.
- $f$ is (strictly) concave if and only if $-f$ is (strictly) convex.
Jensen’s Inequality in Likelihood Function

Let \( Q \) be some distribution over \( z \)'s (\( \sum_z Q(z) = 1 \) and \( Q(z) \geq 0 \)), we have:

\[
\mathcal{L}(\Phi | \mathcal{X}) = \sum_i \log p(x^{(i)} | \Phi) = \sum_i \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)} | \Phi) \tag{1}
\]

\[
= \sum_i \log \sum_{z^{(i)}} Q(z^{(i)}) \frac{p(x^{(i)}, z^{(i)} | \Phi)}{Q(z^{(i)})} \tag{2}
\]

\[
\geq \sum_i \sum_{z^{(i)}} Q(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)} | \Phi)}{Q(z^{(i)})} \tag{3}
\]

- \( f(x) = \log x \) is a concave function
- \( E_z[ \frac{p(x^{(i)}, z^{(i)} | \Phi)}{Q(z^{(i)})} ] = \sum_{z^{(i)}} Q(z^{(i)}) \left( \frac{p(x^{(i)}, z^{(i)} | \Phi)}{Q(z^{(i)})} \right) \)

The above formula gives a lower-bound on the likelihood \( \mathcal{L}(\Phi | \mathcal{X}) \).
The Choice of Auxiliary Function

Choose $Q$ to make the lower-bound tight at some value of $\Phi$, i.e., make the inequality above hold with equality at current particular value of $\Phi$.

To make the Jensen’s inequality hold with equality, it is sufficient that the expectation is taken over a constant variable, i.e.,

$$\frac{p(x^{(i)}, z^{(i)} | \Phi)}{Q(z^{(i)})} = C$$

Since $\sum_{z(i)} Q(z^{(i)}) = 1$, we can choose:

$$Q(z^{(i)}) = \frac{p(x^{(i)}, z^{(i)} | \Phi)}{\sum_{z(i)} p(x^{(i)}, z^{(i)} | \Phi)}$$

$$= \frac{p(x^{(i)}, z^{(i)} | \Phi)}{p(x^{(i)} | \Phi)}$$

$$= p(z^{(i)} | x^{(i)}, \Phi)$$
The Convergence of EM (1)

- Let $\Phi^t$ and $\Phi^{t+1}$ be the parameters from two successive iterations of EM.
- We have chosen $Q^t(z^{(i)}) = p(z^{(i)}|x^{(i)}, \Phi^t)$ to ensure the Jensen’s equality:

$$\mathcal{L}(\Phi^t|\mathcal{X}) = \sum_i \sum_{z^{(i)}} Q^t(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}|\Phi^t)}{Q^t(z^{(i)})} \quad (4)$$
The Convergence of EM (2)

- $\Phi^{t+1}$ is then obtained by maximizing the right hand side of Eqn (4).

Thus,

\[
\mathcal{L}(\Phi^{t+1} | \mathcal{X}) \geq \sum_i \sum_{z(i)} Q^t(z(i)) \log \frac{p(x(i), z(i) | \Phi^{t+1})}{Q^t(z(i))} \quad (5)
\]

\[
\geq \sum_i \sum_{z(i)} Q^t(z(i)) \log \frac{p(x(i), z(i) | \Phi^t)}{Q^t(z(i))} \quad (6)
\]

\[
= \mathcal{L}(\Phi^t | \mathcal{X}) \quad (7)
\]

- Eqn (5): Jensen’s inequality (Eqn (3)) hold for all $Q$ and $\Phi$
- Eqn (6): $\Phi^{t+1}$ is chosen to maximize r.h.s. of Eqn (4)
- monotonically converge
EM for Gaussian Mixtures

- Indicator variables \( \mathbf{z}^{(i)} = (z_{1}^{(i)}, \ldots, z_{k}^{(i)})^T \):

\[
z_{l}^{(i)} = \begin{cases} 
1 & \text{if } x^{(i)} \text{ belones to cluster } G_{l} \\
0 & \text{otherwise}
\end{cases}
\]

- Gaussian component densities:

\[
p(x|G_{l}) = \mathcal{N}(\mu_{l}, \Sigma_{l})
\]

- Let the prior probabilities \( P(G_{l}) = \pi_{l} \), so

\[
P(z^{(i)}) = \prod_{l=1}^{k} \pi_{l}^{z_{l}^{(i)}}
\]
**EM for Gaussian Mixtures (2)**

- **Probability of observed variable given hidden variable:**

  \[ p(x^{(i)}|z^{(i)}) = \prod_{l=1}^{k} [p_l(x^{(i)})]^{z_{l}^{(i)}} \]

  where \( p_l(x^{(i)}) \) is the shorthand for \( p(x^{(i)}|G_l) \).

- **Joint density:**

  \[ p(x^{(i)}, z^{(i)}) = P(z^{(i)})p(x^{(i)}|z^{(i)}) \]

- **Complete-data log likelihood:**

  \[
  \mathcal{L}_C(\Phi|X, Z) = \log \prod_i p(x^{(i)}, z^{(i)}|\Phi) = \sum_i \log p(x^{(i)}, z^{(i)}|\Phi) \quad (8)
  \]

  \[
  = \sum_i [\log p(z^{(i)}|\Phi) + \log p(x^{(i)}|z^{(i)}, \Phi)] \quad (9)
  \]

  \[
  = \sum_i \sum_l z_{l}^{(i)} [\log \pi_l + \log p_l(x^{(i)}|\Phi)] \quad (10)
  \]
E-Step for Gaussian Mixtures

Evaluation of $Q(\Phi|\Phi^t)$:

$$Q(\Phi|\Phi^t) = E[\mathcal{L}_C(\Phi|\mathcal{X}, \mathcal{Z})|\mathcal{X}, \Phi^t]$$

$$= \sum_i \sum_l E[z_l(i)|\mathcal{X}, \Phi^t][\log \pi_l + \log p_l(x_l(i)|\Phi)]$$

where

$$E[z_l(i)|\mathcal{X}, \Phi^t] = E[z_l(i)|x_l(i), \Phi^t] = P(z_l(i) = 1|x_l(i), \Phi^t)$$

$$= \frac{p(x_l(i)|z_l(i) = 1, \Phi^t)P(z_l(i) = 1|\Phi^t)}{p(x_l(i)|\Phi^t)}$$

$$= \frac{p_l(x_l(i)|\Phi^t)\pi_l}{\sum_j p_j(x_l(i)|\Phi^t)\pi_j}$$

$$= \frac{p(x_l(i)|G_l, \Phi^t)P(G_l)}{\sum_j p(x_l(i)|G_j, \Phi^t)P(G_j)} = P(G_l|x_l(i), \Phi^t) \equiv h_l^{(i)}$$
The expected value of the hidden variable, $h_l^{(i)}$, is the posterior probability that $x^{(i)}$ is generated by component $G_l$, i.e., $P(G_l|x^{(i)}, \Phi^t)$.

Since $P(G_l|x^{(i)}, \Phi^t) \in [0, 1]$, it can be seen as a soft label, as opposed to the hard label ($\in \{0, 1\}$) for $k$-means.

Strictly speaking, the E-step only computes $h_l^{(i)}$ but not $Q(\Phi|\Phi^t)$.

For Gaussian components, $\hat{p}_l(x^{(i)}|\Theta_l) = \mathcal{N}(m_l, S_l)$, and so

$$h_l^{(i)} = \frac{\pi_l |S_l|^{-1/2} \exp[-\frac{1}{2}(x^{(i)} - m_l)^T S_l^{-1} (x^{(i)} - m_l)]}{\sum_j \pi_j |S_j|^{-1/2} \exp[-\frac{1}{2}(x^{(i)} - m_j)^T S_j^{-1} (x^{(i)} - m_j)]}$$
M-Step for Gaussian Mixtures

- **Maximization of** $Q(\Phi | \Phi^t)$:

  \[
  \Phi^{t+1} = \arg\max_{\Phi} Q(\Phi | \Phi^t) = \arg\max_{\Phi} \left[ \sum_i \sum_l h_i^{(i)} \log \pi_l + \sum_l \sum_i h_i^{(i)} \log p_l(x^{(i)} | \Phi) \right]
  \]

- The second term of $Q(\Phi | \Phi^t)$ does not depend on $\pi_l$. Using the constraint $\sum_l \pi_l = 1$ to define the Lagrangian, we solve for

  \[
  \nabla_{\pi_l} \left[ \sum_i \sum_l h_i^{(i)} \log \pi_l - \lambda \left( \sum_l \pi_l - 1 \right) \right] = 0
  \]

  to get

  \[
  \pi_l = \frac{\sum_i h_i^{(i)}}{N}
  \]
M-Step for Gaussian Mixtures (2)

- The first term of $Q(\phi | \phi^t)$ does not depend on $\Theta_i$. We solve for

$$\nabla_{\Theta_i} \sum_i \sum_l h_l(i) \log p_l(x(i)|\Phi) = 0$$

- For Gaussian components, $\hat{p}_l(x(i)|\Theta_i) = \mathcal{N}(m_l, S_l)$, and so we get

$$m_l^{t+1} = \frac{\sum_i h_l(i) x(i)}{\sum_i h_l(i)}$$

$$S_l^{t+1} = \frac{\sum_i h_l(i) (x(i) - m_l^{t+1})(x(i) - m_l^{t+1})^T}{\sum_i h_l(i)}$$
Introducing Model Bias

- When the sample is small, **overfitting** may occur.
- Possible solutions via introducing model bias:
  - Constraining the covariance matrices of the Gaussian components, e.g., use shared or diagonal covariance matrices.
  - Applying Principal Component Analysis (PCA) or Factor Analysis (FA) to perform **dimensionality reduction** in the components, e.g., mixtures of latent variable models:

\[
p(x^{(i)}|G_l) = \mathcal{N}(m_l, V_lV_l^T + \Psi_l)
\]
Summary: EM Algorithm

- A way of maximizing likelihood function for latent variable models. Finds MLE of parameters when the original (hard) problem can be broken up into two (easy) pieces:
  - Estimate some missing or unobserved data from observed data and current parameters.
  - Using this complete data, find the maximum likelihood parameter estimates.

- Alternate between filling in the latent variables using the best guess (posterior) and updating the parameters based on this guess.
  - E-step: $h_{i}^{t+1} = \mathbb{E}[z_{i} | \mathcal{X}, \Phi^{t}]$
  - M-step: $\Phi^{t+1} = \arg \max_{\Phi} \mathcal{Q}(\Phi | \Phi^{t})$

- In the M-step we optimize a lower bound on the likelihood. In the E-step we close the gap, making bound = likelihood.
Markov Chains

- Discrete-time state sequence:
  \[
  \omega(1), \ldots, \omega(t), \ldots
  \]
  where \( \omega(t) \) denotes the state at time \( t \)

- First-order Markov process:
  \[
  P(\omega(t + 1)|\omega(1), \ldots, \omega(t)) = P(\omega(t + 1)|\omega(t))
  \]

- Transition probability matrix:
  \[
  \begin{bmatrix}
  a_{11} & a_{12} & a_{13} \\
  a_{21} & a_{22} & a_{23} \\
  a_{31} & a_{32} & a_{33}
  \end{bmatrix}
  \]
  where \( a_{ij} = P(\omega(t + 1) = \omega_j|\omega(t) = \omega_i), \sum_j a_{ij} = 1 \)

- The state sequence may be observable or unobservable.
Hidden Markov Models

- The state sequence of a hidden Markov model (HMM) is unobservable.
- Each hidden or unobservable state is associated with a separate probability distribution of the observable output events.
- HMM diagram:
Model Formulation for Discrete HMM

- Set of all states: $\Omega = \{\omega_1, \ldots, \omega_c\}$
- Set of all observation symbols: $V = \{v_1, \ldots, v_m\}$
- State sequence: $\omega = \omega(1)\omega(2)\ldots\omega(T)$
- Observation sequence: $X = x(1)x(2)\ldots x(T)$
- State transition probability distribution: $A = \{a_{ij}|a_{ij} = P(\omega(t+1) = \omega_j|\omega(t) = \omega_i)\}$
- Observation symbol probability distribution: $B = \{b_{jk}|b_{jk} = P(x(t) = v_k|\omega(t) = \omega_j)\}$
- Initial state probability distribution: $\Pi = \{\pi_i|\pi_i = P(\omega(1) = \omega_i)\}$
- Model parameter vector: $\theta = (A, B, \Pi)$
Three Fundamental Problems

- **Likelihood evaluation problem:**
  - Given
    - Observation sequence \( X \)
    - Model parameter vector \( \theta \)
  - Find
    - Likelihood \( P(X|\theta) \)

- **State sequence decoding problem:**
  - Given
    - Observation sequence \( X \)
    - Model parameter vector \( \theta \)
  - Find
    - State sequence \( \omega \) that is optimal w.r.t. some optimality criterion

- **Parameter estimation (or learning) problem:**
  - Given
    - Observation sequence \( X \)
  - Find
    - ML estimate of \( \theta \) such that: \( \theta_{ML} = \arg \max_{\theta} P(X|\theta) \)
Likelihood Evaluation Problem

- **Brute-force computation:**
  \[
  P(X|\theta) = \sum_{\omega} P(X|\omega, \theta)P(\omega|\theta)
  \]
  \[
  P(X|\omega, \theta) = b_{\omega(1)}x(1)b_{\omega(2)}x(2) \cdots b_{\omega(T)}x(T)
  \]
  \[
  P(\omega|\theta) = \pi_{\omega(1)}a_{\omega(1)}\omega(2)a_{\omega(2)}\omega(3) \cdots a_{\omega(T-1)}\omega(T)
  \]

  Hence,
  \[
  P(X|\theta) = \sum_{\omega} \prod_{t=1}^{T} a_{\omega(t-1)}\omega(t)b_{\omega(t)}x(t)
  \]

  where \( a_{\omega(0)}\omega(1) \) denotes \( \pi_{\omega(1)} \)
  
  - Computationally expensive!
  
  - Computationally attractive alternatives:
    - Either the forward algorithm or the backward algorithm can be used to compute \( P(X|\theta) \).
Likelihood Evaluation Problem - Forward Algorithm

- **Forward variables**: \( \alpha_i(t) \equiv P(x(1) \ldots x(t), \omega(t) = \omega_i|\theta) \)
- **Basis step**: For every state \( i \):
  \[
  \alpha_i(1) = \pi_i b_{ix(1)}
  \]
- **General steps**: For every time step from \( t = 2 \) to \( T \), for every state \( j \):
  \[
  \alpha_j(t) = \left[ \sum_{i=1}^{c} \alpha_i(t - 1) a_{ij} \right] b_{jx(t)}
  \]
- **Final step**:
  \[
  P(X|\theta) = \sum_{i=1}^{c} \alpha_i(T)
  \]
- **Time complexity**: \( O(c^2 T) \)
Illustration of Forward Algorithm
Likelihood Evaluation Problem - Backward Algorithm

- **Backward variables**: $\beta_i(t) \equiv P(x(t+1)x(t+2) \ldots x(T)|\omega(t) = \omega_i, \theta)$
- **Basis step**: For every state $i$:
  \[
  \beta_i(T) = \frac{1}{c}
  \]
- **General steps**: For every time step from $t = T - 1$ to 1, for every state $i$:
  \[
  \beta_i(t) = \sum_{j=1}^{c} a_{ij} b_{ij(t+1)} \beta_j(t+1)
  \]
- **Final step**:
  \[
  P(X|\theta) = \sum_{i=1}^{c} \pi_i b_{ix(1)} \beta_i(1)
  \]
- **Time complexity**: $O(c^2 T)$
The Model

State Sequence Decoding Problem

Find the best state sequence $\omega^*$ such that

$$
\omega^* = \arg\max_{\omega} P(X, \omega|\theta) \\
= \arg\max_{\omega} P(\omega|X, \theta) P(X|\theta) \\
= \arg\max_{\omega} P(\omega|X, \theta)
$$

Thus, $\omega^*$ maximizes the posterior probability of $\omega$. 
State Sequence Decoding Problem (2)

- Equivalent form of optimality criterion:

\[ P(X, \omega | \theta) = P(\omega | \theta) P(X | \omega, \theta) \]

\[ = P(\omega(1), \omega(2), \ldots, \omega(T) | \theta) \cdot P(x(1), x(2), \ldots, x(T) | \omega(1), \omega(2), \ldots, \omega(T), \theta) \]

\[ = \pi_{\omega(1)} \prod_{t=1}^{T-1} P(\omega(t+1) | \omega(t), \theta) \prod_{t=1}^{T} P(x(t) | \omega(t), \theta) \]

\[ = \prod_{t=1}^{T} a_{\omega(t-1)\omega(t)} b_{\omega(t)} x(t) \]

where \( a_{\omega(0)\omega(1)} \) denotes \( \pi_{\omega(1)} \)

- Dynamic programming can be used to solve this optimization problem efficiently.
State Sequence Decoding Problem - Viterbi Algorithm

- **Best cumulative scores:**
  \[ \delta_i(t) = \max_{\omega(1), \ldots, \omega(t-1)} P(\omega(1), \ldots, \omega(t-1), \omega(t) = \omega_i, x(1), \ldots, x(t)|\theta) \]

- **Basis step:** For every state \(i\),
  \[ \delta_i(1) = \pi_i b_{i \cdot x(1)}, \quad \psi_i(1) = 0 \]

- **General steps:** For every time step from \(t = 2\) to \(T\), for every state \(j\),
  \[ \delta_j(t) = \left[ \max_{1 \leq i \leq c} \delta_i(t-1) a_{ij} \right] b_{j \cdot x(t)}, \quad \psi_j(t) = \arg \max_{1 \leq i \leq c} \delta_i(t-1) a_{ij} \]

- **Final step:**
  \[ P^* = \max_{1 \leq i \leq c} \delta_i(T), \quad \omega^*(T) = \arg \max_{1 \leq i \leq c} \delta_i(T) \]
State Sequence Decoding Problem - Viterbi Algorithm (2)

- **State sequence backtracking**: For every time step from $t = T - 1$ to 1,

  $$\omega^*(t) = \psi \omega^*(t+1)(t+1)$$

- **Time complexity**: $O(c^2 T)$

- A modified version of Viterbi algorithm works in the logarithm domain to replace the multiplication operations by additions. Another advantage of operating in the logarithm domain is that computational underflow can be avoided.
Parameter Estimation Problem

- **Baum-Welch re-estimation algorithm** (or called forward-backward algorithm):
  - An EM algorithm (similar to that for mixture density estimation)
  - Posterior probability of a state sequence being in state $i$ at time $t$ and in state $j$ at time $t + 1$, given $X$ and $\theta$:

$$\xi_{ij}(t) \equiv \frac{P(\omega(t) = \omega_i, \omega(t + 1) = \omega_j | X, \theta)}{P(X | \theta)} = \frac{\alpha_i(t) a_{ij} b_{jx(t+1)} \beta_j(t + 1)}{\sum_{i=1}^{c} \alpha_i(T)}$$
Parameter Estimation Problem (2)

- Posterior probability of state sequence being in state $i$ at time $t$, given $X$ and $\theta$:

$$
\gamma_i(t) \equiv P(\omega(t) = \omega_i|X, \theta) = \frac{P(\omega(t) = \omega_i, X|\theta)}{P(X|\theta)} = \frac{\alpha_i(t)\beta_i(t)}{\sum_{i=1}^{c}\alpha_i(T)}
$$
Baum-Welch Re-estimation Formula

Re-estimation of model parameters (changing $\theta$ to $\hat{\theta}$):

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

- expected number of transitions from state $i$ to $j$
- expected number of transitions from state $i$

$$\hat{b}_{jk} = \frac{\sum\{t|x(t)=v_k\} \gamma_j(t)}{\sum_{t=1}^{T} \gamma_j(t)}$$

- expected number of times in state $j$ when $v_k$ is observed
- expected number of times in state $j$

$$\hat{\pi}_i = \frac{\gamma_i(1)}{\sum_{i=1}^{c} \gamma_i(1)} = \gamma_i(1)$$

- expected frequency in state $i$ at $t = 1$
Using the EM ideas, it can be proved that

\[ P(X|\hat{\theta}) \geq P(X|\theta) \]

Algorithm:
- Initialize \( \theta \)
- Repeat
  - use Baum-Welch re-estimation formula to compute \( \hat{\theta} \) from \( \theta \) and \( X \)
  - \( \hat{\theta} \leftarrow \theta \)
- Until convergence
- Return \( \theta \)
Elements of an HMM $\theta = (A, B, \Pi)$
- state transition probability matrix $A = \{a_{ij}\}$:
  \[
a_{ij} = P(\omega(t+1) = \omega_j | \omega(t) = \omega_i)
\]
- observation probability distribution $B = \{b_{jk}\}$:
  \[
b_{jk} = P(x(t) = v_k | \omega(t) = \omega_j)
\]
- initial state distribution $\Pi = \{\pi_i\}$:
  \[
  \pi_i = P(\omega(1) = \omega_i)
\]

Basic problems
- evaluation problem (forward and backward algorithms)
- decoding problem (Viterbi algorithm)
- learning/estimation problem (Baum-Welch algorithm)

Variants: continuous-density HMMs, semi-continuous HMMs, etc.
Applications of HMMs

- HMM is probabilistic model for **sequential data**. It can be used to solve the learning problems such as:
  - $K$ HMMs for $K$ class **classification**
  - Mixture of HMMs for **clustering**

- Applications of HMMs
  - Speech recognition
  - Speech synthesis
  - Part-of-speech tagging
  - Machine translation
  - Gene prediction
  - Alignment of bio-sequences
  - Activity recognition
  - ...
Hierarchical clustering generally refers to methods that cluster instances into a hierarchical structure (called dendrogram) based on some similarity or distance measure between them.

- **Distance measures:**
  - **Minkowski distance:**
    \[
    d_m(x^{(r)}, x^{(s)}) = \left[ \sum_{j=1}^{D} |x_j^{(r)} - x_j^{(s)}|^p \right]^{1/p}
    \]
  - **Euclidean distance:** special case of Minkowski distance with \( p = 2 \)
  - **City-block distance:** special case of Minkowski distance with \( p = 1 \)
    \[
    d_{cb}(x^{(r)}, x^{(s)}) = \sum_{j=1}^{D} |x_j^{(r)} - x_j^{(s)}|
    \]
An **agglomerative clustering** algorithm starts with \( N \) groups each with one instance.

At each iteration, the two most similar groups are merged, reducing the number of groups by one.

The process stops when there is only one group left.

Dendrogram (can be intersected at any level to get the clusters):

![Dendrogram](image)

A **divisive clustering** algorithm starts with one group and goes in the opposite direction.
Distance between Groups $G_i$ and $G_j$

- **Single-link clustering:**

  \[
  d_{\text{min}}(G_i, G_j) = \min_{x^{(r)} \in G_i, x^{(s)} \in G_j} d(x^{(r)}, x^{(s)})
  \]

- **Complete-link clustering:**

  \[
  d_{\text{max}}(G_i, G_j) = \max_{x^{(r)} \in G_i, x^{(s)} \in G_j} d(x^{(r)}, x^{(s)})
  \]

- **Average-link clustering:**

  \[
  d_{\text{avg}}(G_i, G_j) = \frac{1}{|G_i| \cdot |G_j|} \sum_{x^{(r)} \in G_i, x^{(s)} \in G_j} d(x^{(r)}, x^{(s)})
  \]

- Other possibilities, e.g., distance between group means (centroids).
Choosing $k$

This is a **model selection** issue for clustering.

Some common possibilities:

- Defined by the **application**, e.g., image quantization.
- **Visualize** the data (e.g., in 2-D using PCA) and check for clusters.
- **Incremental approach**: add one at a time and monitor the reconstruction error, log likelihood, or intergroup distances.
- **Domain experts** check the clustering result.