Mixture Models & EM

Nicholas Ruozzi
University of Texas at Dallas

based on the slides of Vibhav Gogate
Previously...

- We looked at $k$-means and hierarchical clustering as mechanisms for unsupervised learning
  - $k$-means was simply a block coordinate descent scheme for a specific objective function
- Today: how to learn probabilistic models for unsupervised learning problems
EM: Soft Clustering

• Clustering (e.g., k-means) typically assumes that each instance is given a “hard” assignment to exactly one cluster

• Does not allow uncertainty in class membership or for an instance to belong to more than one cluster
  • Problematic because data points that lie roughly midway between cluster centers are assigned to one cluster

• Soft clustering gives probabilities that an instance belongs to each of a set of clusters
Probabilistic Clustering

- Try a probabilistic model!
  - Allows overlaps, clusters of different size, etc.
- Can tell a generative story for data
  - \( p(x|y) p(y) \)
- Challenge: we need to estimate model parameters without labeled \( y \)'s (i.e., in the unsupervised setting)

<table>
<thead>
<tr>
<th>Z</th>
<th>( X_1 )</th>
<th>( X_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>??</td>
<td>0.1</td>
<td>2.1</td>
</tr>
<tr>
<td>??</td>
<td>0.5</td>
<td>-1.1</td>
</tr>
<tr>
<td>??</td>
<td>0.0</td>
<td>3.0</td>
</tr>
<tr>
<td>??</td>
<td>-0.1</td>
<td>-2.0</td>
</tr>
<tr>
<td>??</td>
<td>0.2</td>
<td>1.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Clusters of different shapes and sizes

Clusters can overlap! \((k\text{-means doesn’t allow this})\)
Finite Mixture Models

• Given a dataset: $x^{(1)}, \ldots, x^{(N)}$

• **Mixture model**: $\Theta = \{\lambda_1, \ldots, \lambda_k, \theta_1, \ldots, \theta_k\}$

\[
p(x|\Theta) = \sum_{y=1}^{k} \lambda_y p_y(x|\theta_y)
\]

• $p_y(x|\theta_y)$ is a mixture component from some family of probability distributions parameterized by $\theta_y$ and $\lambda \geq 0$ such that $\sum_y \lambda_y = 1$ are the mixture weights

• We can think of $\lambda_y = p(Y = y|\Theta)$ for some random variable $Y$ that takes values in $\{1, \ldots, k\}$
Finite Mixture Models

Uniform mixture of 3 Gaussians
Multivariate Gaussian

• A $d$-dimensional multivariate Gaussian distribution is defined by a $d \times d$ covariance matrix $\Sigma$ and a mean vector $\mu$

$$p(x|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d \text{det}(\Sigma)}} \exp\left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu)\right)$$

• The covariance matrix describes the degree to which pairs of variables vary together

• The diagonal elements correspond to variances of the individual variables
A $d$-dimensional multivariate Gaussian distribution is defined by a $d \times d$ covariance matrix $\Sigma$ and a mean vector $\mu$

$$p(x|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

The covariance matrix must be a symmetric positive definite matrix in order for the above to make sense

- Positive definite: all eigenvalues are positive & matrix is invertible
- Ensures that the quadratic form is concave
Gaussian Mixture Models (GMMs)

- We can define a GMM by choosing the $k^{th}$ component of the mixture to be a Gaussian density with parameters

$$\theta_k = \{\mu_k, \Sigma_k\}$$

$$p(x|\mu_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^d\det(\Sigma_k)}} \exp \left( -\frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) \right)$$

We could cluster by fitting a mixture of $k$ Gaussians to our data

How do we learn these kinds of models?
Learning Gaussian Parameters

• MLE for supervised univariate Gaussian

$$
\mu_{MLE} = \frac{1}{N} \sum_{i=1}^{N} x^{(i)}
$$

$$
\sigma_{MLE}^2 = \frac{1}{N} \sum_{i=1}^{N} (x^{(i)} - \mu_{MLE})^2
$$

• MLE for supervised multivariate Gaussian

$$
\mu_{MLE} = \frac{1}{N} \sum_{i=1}^{N} x^{(i)}
$$

$$
\Sigma_{MLE} = \frac{1}{N} \sum_{i=1}^{N} (x^{(i)} - \mu_{MLE})(x^{(i)} - \mu_{MLE})^T
$$
Learning Gaussian Parameters

• MLE for supervised multivariate mixture of $k$ Gaussian distributions

$$\mu_{MLE}^k = \frac{1}{|M_k|} \sum_{i \in M_k} x^{(i)}$$

$$\Sigma_{MLE}^k = \frac{1}{|M_k|} \sum_{i \in M_k} (x^{(i)} - \mu_{MLE}^k)(x^{(i)} - \mu_{MLE}^k)^T$$

Sums are over the observations that were generated by the $k^{th}$ mixture component (this requires that we know which points were generated by which distribution!)
The Unsupervised Case

- What if our observations do not include information about which of the \( k \) mixture components generated them?

- Consider a joint probability distribution over data points, \( x^{(i)} \), and mixture assignments, \( y \in \{1, \ldots, k\} \)

\[
\arg\max_\Theta \prod_{i=1}^N p(x^{(i)}|\Theta) = \arg\max_\Theta \prod_{i=1}^N \sum_{y=1}^k p(x^{(i)}, Y = y|\Theta)
\]

\[
= \arg\max_\Theta \prod_{i=1}^N \sum_{y=1}^k p(x^{(i)}|Y = y, \Theta)p(Y = y|\Theta)
\]
The Unsupervised Case

• What if our observations do not include information about which of the $k$ mixture components generated them?

• Consider a joint probability distribution over data points, $x^{(i)}$, and mixture assignments, $y \in \{1, \ldots, k\}$

$$\arg \max_{\Theta} \prod_{i=1}^{N} p(x^{(i)}|\Theta) = \arg \max_{\Theta} \prod_{i=1}^{N} \sum_{y=1}^{k} p(x^{(i)}, Y = y|\Theta)$$

$$= \arg \max_{\Theta} \prod_{i=1}^{N} \sum_{y=1}^{k} p(x^{(i)}|Y = y, \Theta)p(Y = y|\Theta)$$

We only know how to compute the probabilities for each mixture component.
The Unsupervised Case

- In the case of a Gaussian mixture model
  \[ p(x^{(i)}|Y = y, \Theta) = N(x^{(i)}|\mu_y, \Sigma_y) \]
  \[ p(Y = y|\Theta) = \lambda_y \]

- Differentiating the MLE objective yields a system of equations that is difficult to solve in general
  - The solution: modify the objective to make the optimization easier
Expectation Maximization
Jensen’s Inequality

For a convex function $f : \mathbb{R}^n \to \mathbb{R}$, any $a_1, \ldots, a_k \in [0,1]$ such that $a_1 + \cdots + a_k = 1$, and any $x^{(1)}, \ldots, x^{(k)} \in \mathbb{R}^n$,

$$a_1 f(x^{(1)}) + \cdots + a_k f(x^{(k)}) \geq f(a_1 x^{(1)} + \cdots + a_k x^{(k)})$$

Inequality is reversed for concave functions
EM Algorithm

\[
\log \ell(\Theta) = \sum_{i=1}^{N} \log \sum_{y=1}^{k} p(x^{(i)}, Y = y | \Theta)
\]

\[
= \sum_{i=1}^{N} \log \sum_{y=1}^{k} \frac{q_i(y)}{q_i(y)} p(x^{(i)}, Y = y | \Theta)
\]

\[
\geq \sum_{i=1}^{N} \sum_{y=1}^{k} q_i(y) \log \frac{p(x^{(i)}, Y = y | \Theta)}{q_i(y)}
\]

\[
\equiv F(\Theta, q)
\]
EM Algorithm

\[ \log \ell(\Theta) = \sum_{i=1}^{N} \log \sum_{y=1}^{k} p(x^{(i)}, Y = y | \Theta) \]

\[ = \sum_{i=1}^{N} \log \sum_{y=1}^{k} \frac{q_i(y)}{q_i(y)} p(x^{(i)}, Y = y | \Theta) \]

\[ \geq \sum_{i=1}^{N} \sum_{y=1}^{k} q_i(y) \log \frac{p(x^{(i)}, Y = y | \Theta)}{q_i(y)} \]

\[ \equiv F(\Theta, q) \]

$q_i(y)$ is an arbitrary positive probability distribution.
EM Algorithm

\[
\log \ell(\Theta) = \sum_{i=1}^{N} \log \sum_{y=1}^{k} p(x^{(i)}, Y = y | \Theta)
\]

\[
= \sum_{i=1}^{N} \log \sum_{y=1}^{k} \frac{q_i(y)}{q_i(y)} p(x^{(i)}, Y = y | \Theta)
\]

\[
\geq \sum_{i=1}^{N} \sum_{y=1}^{k} q_i(y) \log \frac{p(x^{(i)}, Y = y | \Theta)}{q_i(y)}
\]

\[
\equiv F(\Theta, q)
\]
EM Algorithm

\[
\arg\max_{\Theta, q_1, \ldots, q_N} \sum_{i=1}^{N} \sum_{y=1}^{k} q_i(y) \log \frac{p(x^{(i)}, Y = y | \Theta)}{q_i(y)}
\]

- This objective is not jointly concave in \(\Theta\) and \(q_1, \ldots, q_N\)
  - Best we can hope for is a local maxima (and there could be a LOT of them)

- The EM algorithm is a block coordinate ascent scheme that finds a local optimum of this objective
  - Start from an initialization \(\Theta^0\) and \(q_1^0, \ldots, q_N^0\)
EM Algorithm

- E step: with the \( \theta \)'s fixed, maximize the objective over \( q \)

\[
q^{t+1} \in \arg \max_{q_1, \ldots, q_N} \sum_{i=1}^{N} \sum_{y=1}^{k} q_i(y) \log \left( \frac{p(x^{(i)}, Y = y | \Theta^t)}{q_i(y)} \right)
\]

- Using the method of Lagrange multipliers for the constraint that \( \sum_y q_i(y) = 1 \) gives

\[
q_i^{t+1}(y) = p(Y = y | X = x^{(i)}, \Theta^t)
\]
EM Algorithm

- M step: with the $q$'s fixed, maximize the objective over $\Theta$

\[
\theta^{t+1} \in \arg \max_{\Theta} \sum_{i=1}^{N} \sum_{y=1}^{k} q_{i}^{t+1}(y) \log \frac{p(x^{(i)}, Y = y | \theta)}{q_{i}^{t+1}(y)}
\]

- For the case of GMM, we can compute this update in closed form
  - This is not necessarily the case for every model
  - May require gradient ascent
EM Algorithm

• Start with random parameters

• E-step maximizes a lower bound on the log-sum for fixed parameters

• M-step solves a MLE estimation problem for fixed probabilities

• Iterate between the E-step and M-step until convergence
EM for Gaussian Mixtures

- **E-step:**

\[
q^t_i(y) = \frac{\lambda^t_y \cdot p(x^{(i)} | \mu^t_y, \Sigma^t_y)}{\sum_{y'} \lambda^t_{y'} \cdot p(x^{(i)} | \mu^t_{y'}, \Sigma^t_{y'})}
\]

- **M-step:**

\[
\mu_{y}^{t+1} = \frac{\sum_{i=1}^{N} q^t_i(y) x^{(i)}}{\sum_{i=1}^{N} q^t_i(y)}
\]

\[
\Sigma_{y}^{t+1} = \frac{\sum_{i=1}^{N} q^t_i(y) (x^{(i)} - \mu_{y}^{t+1})(x^{(i)} - \mu_{y}^{t+1})^T}{\sum_{i=1}^{N} q^t_i(y)}
\]

\[
\lambda_{y}^{t+1} = \frac{1}{N} \sum_{i=1}^{N} q^t_i(y)
\]

Probability of \(x^{(i)}\) under the appropriate multivariate normal distribution
EM for Gaussian Mixtures

- **E-step:**

\[ q_i^t(y) = \frac{\lambda^t_y \cdot p(x^{(i)} | \mu^t_y, \Sigma^t_y)}{\sum_{y'} \lambda^t_{y'} \cdot p(x^{(i)} | \mu^t_{y'}, \Sigma^t_{y'})} \]

- **M-step:**

\[ \mu^{t+1}_y = \frac{\sum_{i=1}^N q_i^t(y)x^{(i)}}{\sum_{i=1}^N q_i^t(y)} \]

\[ \Sigma^{t+1}_y = \frac{\sum_{i=1}^N q_i^t(y)(x^{(i)} - \mu^{t+1}_y)(x^{(i)} - \mu^{t+1}_y)^T}{\sum_{i=1}^N q_i^t(y)} \]

\[ \lambda^{t+1}_y = \frac{1}{N} \sum_{i=1}^N q_i^t(y) \]

Probability of \( x^{(i)} \) under the mixture model
Gaussian Mixture Example: Start
After first iteration
After 2nd iteration
After 3rd iteration
After 4th iteration
After 5th iteration
After 6th iteration

\[ p = 0.315 \]

\[ p = 0.287 \]
After 20th iteration
Properties of EM

- EM converges to a local optimum
  - This is because each iteration improves the log-likelihood
  - Proof same as $k$-means (just block coordinate ascent)
    - E-step can never decrease likelihood
    - M-step can never decrease likelihood
  - If we make hard assignments instead of soft ones, algorithm is equivalent to $k$-means!