Mixture Models & EM algorithm
Lecture 20

Narges Razavian
New York University

Slides adapted from Carlos Guestrin, Dan Klein, Luke Zettlemoyer,
Dan Weld, Vibhav Gogate, and Andrew Moore
The Evils of “Hard Assignments”? 

• Clusters may overlap
• Some clusters may be “wider” than others
• Distances can be deceiving!
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 Ys

<table>
<thead>
<tr>
<th>Y</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>
The General GMM assumption

- $P(Y)$: There are $k$ components
- $P(X|Y)$: Each component generates data from a **multivariate Gaussian** with mean $\mu_i$ and covariance matrix $\Sigma_i$

Each data point is sampled from a **generative process**:

1. Choose component $i$ with probability $P(y=i)$
2. Generate datapoint $\sim N(\mu_i, \Sigma_i)$

Gaussian mixture model (GMM)
What Model Should We Use?

• Depends on X!
• Here, maybe Gaussian Naïve Bayes?
  – Multinomial over clusters Y
  – (Independent) Gaussian for each $X_i$ given Y

$$p(Y_i = y_k) = \theta_k$$

$$P(X_i = x \mid Y = y_k) = \frac{1}{\sigma_{ik} \sqrt{2\pi}} e^{\frac{-(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$
Could we make fewer assumptions?

- What if the $X_i$ co-vary?
- What if there are multiple peaks?
- Gaussian Mixture Models!
  - $P(Y)$ still multinomial
  - $P(X | Y)$ is a multivariate Gaussian distribution:

\[
P(X = x_j | Y = i) = \frac{1}{(2\pi)^{m/2} \| \Sigma_i \|^{1/2}} \exp \left[ -\frac{1}{2} (x_j - \mu_i)^T \Sigma^{-1}_i (x_j - \mu_i) \right]
\]
Multivariate Gaussians

\[ P(X=x_j) = \frac{1}{(2\pi)^{m/2} \| \Sigma \|^{1/2}} \exp\left[ -\frac{1}{2} (x_j - \mu_i)^T \Sigma^{-1} (x_j - \mu_i) \right] \]

\[ \Sigma \propto \text{identity matrix} \]
Multivariate Gaussians

\[ P(X = x_j) = \frac{1}{(2\pi)^{m/2} \| \Sigma \|^{1/2}} \exp \left[ -\frac{1}{2} (x_j - \mu_i)^T \Sigma^{-1} (x_j - \mu_i) \right] \]

\( \Sigma = \) diagonal matrix
\( X_i \) are independent \( a/l a \) Gaussian NB
Multivariate Gaussians

\[ P(X=x_j) = \frac{1}{(2\pi)^{m/2} \| \Sigma \|^{1/2}} \exp \left[ -\frac{1}{2} (x_j - \mu_i)^T \Sigma^{-1} (x_j - \mu_i) \right] \]

\[ \Sigma = \text{arbitrary (semidefinite) matrix:} \]
- specifies rotation (change of basis)
- eigenvalues specify relative elongation
Multivariate Gaussians

Eigenvalue, $\lambda$, of $\Sigma$

Covariance matrix, $\Sigma$, = degree to which $x_i$ vary together

$$P(X=x_j) = \frac{1}{(2\pi)^{m/2} \| \Sigma \|^{1/2}} \exp\left[-\frac{1}{2} \left(\mathbf{x}_j - \mu\right)^T \Sigma^{-1} \left(\mathbf{x}_j - \mu\right)\right]$$
Mixtures of Gaussians (1)

Old Faithful Data Set

![scatter plot of Old Faithful data set with time to eruption on the y-axis and duration of last eruption on the x-axis]
Mixtures of Gaussians (1)

Old Faithful Data Set

Single Gaussian

Mixture of two Gaussians
Mixtures of Gaussians (2)

Combine simple models into a complex model:

\[ p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x | \mu_k, \Sigma_k) \]

Component Mixing coefficient

\[ \forall k : \pi_k \geq 0 \quad \sum_{k=1}^{K} \pi_k = 1 \]

\( K = 3 \)
Mixtures of Gaussians (3)
Eliminating Hard Assignments to Clusters

Model data as mixture of multivariate Gaussians
Eliminating Hard Assignments to Clusters

Model data as mixture of multivariate Gaussians
Eliminating Hard Assignments to Clusters

Model data as mixture of multivariate Gaussians

Shown is the posterior probability that a point was generated from $i^{th}$ Gaussian: $\Pr(Y = i \mid x)$
ML estimation in **supervised** setting

- **Univariate Gaussian**

  \[
  \mu_{MLE} = \frac{1}{N} \sum_{i=1}^{N} x_i \\
  \sigma^2_{MLE} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{\mu})^2
  \]

- **Mixture of Multivariate Gaussians**

  ML estimate for each of the Multivariate Gaussians is given by:

  \[
  \mu_{ML}^k = \frac{1}{n} \sum_{j=1}^{n} x_n \\
  \Sigma_{ML}^k = \frac{1}{n} \sum_{j=1}^{n} (x_j - \mu_{ML}^k)(x_j - \mu_{ML}^k)^T
  \]

  Just sums over \(x\) generated from the \(k\)’th Gaussian
That was easy!
But what if *unobserved data*?

- **MLE:**
  - \( \arg\max_\theta \prod_j P(y_j, x_j) \)
  - \( \theta \): all model parameters
    - eg, class probs, means, and variances
- **But we don’t know \( y_j \)'s!!!**
- **Maximize *marginal likelihood*:**
  - \( \arg\max_\theta \prod_j P(x_j) = \arg\max \prod_j \sum_{k=1}^{K} P(Y_j=k, x_j) \)
How do we optimize? Closed Form?

• Maximize *marginal likelihood*:
  \[ \text{argmax}_\theta \prod_j P(x_j) = \text{argmax} \prod_j \sum_{k=1}^K P(Y_j=k, x_j) \]

• Almost always a hard problem!
  – Usually no closed form solution
  – Even when \( \log P(X,Y) \) is convex, \( \log P(X) \) generally isn’t…
  – For all but the simplest \( P(X) \), we will have to do gradient ascent, in a big messy space with lots of local optimum…
Learning general mixtures of Gaussian

\[ P(y = k \mid x_j) \propto \frac{1}{(2\pi)^{m/2} \| \Sigma_k \|^{1/2}} \exp \left[ -\frac{1}{2} (x_j - \mu_k)^T \Sigma_k^{-1} (x_j - \mu_k) \right] P(y = k) \]

- Marginal likelihood:

\[ \prod_{j=1}^{m} P(x_j) = \prod_{j=1}^{m} \sum_{k=1}^{K} P(x_j, y = k) \]

\[ = \prod_{j=1}^{m} \sum_{k=1}^{K} \frac{1}{(2\pi)^{m/2} \| \Sigma_k \|^{1/2}} \exp \left[ -\frac{1}{2} (x_j - \mu_k)^T \Sigma_k^{-1} (x_j - \mu_k) \right] P(y = k) \]

- Need to differentiate and solve for \( \mu_k, \Sigma_k, \) and \( P(Y=k) \) for \( k=1..K \)
- There will be no closed form solution, gradient is complex, lots of local optimum
- *Wouldn’t it be nice if there was a better way!*??
Expectation Maximization
The EM Algorithm

• A clever method for maximizing marginal likelihood:
  – \( \text{argmax}_\theta \prod_j P(x_j) = \text{argmax}_\theta \prod_j \sum_{k=1}^K P(Y_j=k, x_j) \)
  – A type of gradient ascent that can be easy to implement (eg, no line search, learning rates, etc.)

• Alternate between two steps:
  – Compute an expectation
  – Compute a maximization

• Not magic: *still optimizing a non-convex function with lots of local optima*
  – The computations are just easier (often, significantly so!)
EM: Two Easy Steps

Objective: \( \arg\max_{\theta} \log \prod_j \sum_{k=1}^K P(Y_j=k, x_j | \theta) = \sum_j \log \sum_{k=1}^K P(Y_j=k, x_j | \theta) \)

Data: \( \{x_j \mid j=1 .. n\} \)

- **E-step**: Compute expectations to “fill in” missing y values according to current parameters, \( \theta \)
  - For all examples j and values k for \( Y_j \), compute: \( P(Y_j=k \mid x_j, \theta) \)

- **M-step**: Re-estimate the parameters with “weighted” MLE estimates
  - Set \( \theta = \arg\max_{\theta} \sum_j \sum_k P(Y_j=k \mid x_j, \theta) \log P(Y_j=k, x_j | \theta) \)

Especially useful when the E and M steps have closed form solutions!!!
Given a set of Parameters and training data

- Supervised learning problem
- Relearn the parameters based on the new training data
- Estimate the class of each training example using the parameters yielding new (weighted) training data

Class assignment is probabilistic or weighted (soft EM)
Class assignment is hard (hard EM)
Simple example: learn means only!

Consider:

- 1D data
- Mixture of \( k=2 \) Gaussians
- Variances fixed to \( \sigma=1 \)
- Distribution over classes is uniform
- Just need to estimate \( \mu_1 \) and \( \mu_2 \)

\[
\prod_{j=1}^{m} \sum_{k=1}^{K} P(x, Y_j = k) \propto \prod_{j=1}^{m} \sum_{k=1}^{K=2} \exp \left[ -\frac{1}{2\sigma^2} \|x - \mu_k\|^2 \right] P(Y_j = k)
\]
EM for GMMs: only learning means

**Iterate:** On the $t'$th iteration let our estimates be

$$\lambda_t = \{ \mu_1^{(t)}, \mu_2^{(t)} \ldots \mu_K^{(t)} \}$$

**E-step**

Compute “expected” classes of all datapoints

$$P(Y_j = k | x, \mu_1 \ldots \mu_K) \propto \exp\left( \frac{-1}{2\sigma^2} \|x_j - \mu_k\|^2 \right) P(Y_j = k)$$

**M-step**

Compute most likely new $\mu$s given class expectations

$$\mu_k = \frac{\sum_{j=1}^{m} P(Y_j = k | x_j) x_j}{\sum_{j=1}^{m} P(Y_j = k | x_j)}$$
E.M. for General GMMs

Iterate: On the $t'$th iteration let our estimates be

$$\lambda_t = \{ \mu_1^{(t)}, \mu_2^{(t)}, \ldots, \mu_K^{(t)}, \Sigma_1^{(t)}, \Sigma_2^{(t)}, \ldots, \Sigma_K^{(t)}, p_1^{(t)}, p_2^{(t)}, \ldots, p_K^{(t)} \}$$

E-step

Compute “expected” classes of all datapoints for each class

$$P(Y_j = k | x_j, \lambda_t) \propto p_k^{(t)} p \left( x_j | \mu_k^{(t)}, \Sigma_k^{(t)} \right)$$

M-step

Compute weighted MLE for $\mu$ given expected classes above

$$\mu_k^{(t+1)} = \frac{\sum_j P(Y_j = k | x_j, \lambda_t) x_j}{\sum_j P(Y_j = k | x_j, \lambda_t)}$$

$$\Sigma_k^{(t+1)} = \frac{\sum_j P(Y_j = k | x_j, \lambda_t) \left[ x_j - \mu_k^{(t+1)} \right] \left[ x_j - \mu_k^{(t+1)} \right]^T}{\sum_j P(Y_j = k | x_j, \lambda_t)}$$

$$p_k^{(t+1)} = \frac{\sum_j P(Y_j = k | x_j, \lambda_t)}{m}$$

$p_k^{(t)}$ is shorthand for estimate of $P(y=k)$ on $t'$th iteration

$m = \#training\ examples$
Gaussian Mixture Example: Start
After first iteration
After 2nd iteration
After 3rd iteration
After 4th iteration
After 5th iteration
After 6th iteration
After 20th iteration