Expectation Maximization Algorithm

Vibhav Gogate
The University of Texas at Dallas

Slides adapted from Carlos Guestrin, Dan Klein, Luke Zettlemoyer and Dan Weld
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 $\sim N(m_i, \Sigma_i)$

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

- Depends on X!
- Here, maybe Gaussian Naïve Bayes?
  - Multinomial over clusters Y
  - Gaussian over 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 dist’n

$$P(X = x_j \mid Y = i) = \frac{1}{(2\pi)^{m/2} \| S_i \|^{1/2}} \exp \left( \frac{1}{2} (x_j - m_i)^T S_i^{-1} (x_j - m_i) \right)$$
The General GMM assumption

1. What’s a *Multivariate* Gaussian?

2. What’s a *Mixture Model*?
Review: Gaussians

\[ P(x \mid \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \]
Learning Gaussian Parameters (given fully-observable data)

\[
\hat{\mu}_{MLE} = \frac{1}{N} \sum_{i=1}^{N} x_i
\]

\[
\hat{\sigma}^2_{MLE} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{\mu})^2
\]
Geometry of the Multivariate Gaussian

\[ \Delta^2 = (x - \mu)^T \Sigma^{-1} (x - \mu) \]

\[ \Sigma^{-1} = \sum_{i=1}^{D} \frac{1}{\lambda_i} u_i u_i^T \]

\[ \Delta^2 = \sum_{i=1}^{D} \frac{y_i^2}{\lambda_i} \]

\[ y_i = u_i^T (x - \mu) \]

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

\[ \mathcal{N}(x | \mu, \Sigma) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right\} \]
Multivariate Gaussians

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

\[ \Sigma = \text{diagonal matrix} \]

\[ X_i \text{ are independent } \text{ala Gaussian NB} \]
Multivariate Gaussians

\[ \Sigma = \text{arbitrary (semidefinite) matrix} \]
\[ \text{specifies rotation (change of basis)} \]
\[ \text{eigenvalues specify relative elongation} \]
The General GMM assumption

1. What’s a Multivariate Gaussian?
2. What’s a *Mixture Model*?
Mixtures of Gaussians (1)

Old Faithful Data Set

Time to Eruption vs. Duration of Last Eruption
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:

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

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

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

\[ \pi_i = \text{probability point was generated from } i^{\text{th}} \text{ Gaussian} \]
Detour/Review: Supervised MLE for GMM

- How do we estimate parameters for Gaussian Mixtures with fully supervised data?
- Have to define objective and solve optimization problem.

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

- For example, MLE estimate has closed form solution:

\[
M_{\text{ML}} = \frac{1}{n} \sum_{j=1}^{n} x_j
\]

\[
M_{\text{ML}} = \frac{1}{n} \left( x_j - M_{\text{ML}} \right)^T \left( x_j - M_{\text{ML}} \right)
\]
Compare

- **Univariate Gaussian**

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

- **Mixture of Multivariate Gaussians**

  \[
  M_L = \frac{1}{n} \sum_{j=1}^{n} x_j \quad M_L = \frac{1}{n} \left( x_j \quad M_L \right) \left( x_j \quad M_L \right)^T
  \]
That was easy!
But what if *unobserved data*?

- **MLE:**
  - $\arg\max_\theta \prod_j P(y_j,x_j)$
  - $\theta$: all model parameters
    - e.g., class probs, means, and variance for naïve Bayes

- But we don’t know $y_j$’s!!!

- Maximize *marginal likelihood*:
  - $\arg\max_\theta \prod_j P(x_j) = \arg\max \prod_j \sum_{i=1}^k P(y_j=i,x_j)$
How do we optimize? Closed Form?

- Maximize **marginal likelihood**: 
  \[
  \argmax_{\theta} \prod_j P(x_j) = \argmax \prod_j \sum_{i=1}^k P(y_j=i|x_j)
  \]

- Almost always a hard problem!
  - Usually no closed form solution
  - Even when \( P(X,Y) \) is convex, \( P(X) \) 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

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

- Marginal likelihood:

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

- Need to differentiate and solve for \( \mu_i, \Sigma_i, \) and \( P(Y=i) \) for \( i=1..k \)
- 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:
  – \( \arg\max_\theta \prod_j P(x_j) = \arg\max_\theta \prod_j \sum_{i=1}^k P(y_j=i|x_j) \)
  – 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: \textit{still optimizing a non-convex function with lots of local optima}
  – The computations are just easier (often, significantly so!)
EM: Two Easy Steps

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

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

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

- **M-step**: Re-estimate the parameters with “weighted” MLE estimates
  - Set \( \theta = \arg\max_\theta \sum_j \sum_i P(y_j=i \mid x_j, \theta) \log P(y_j=i, x_j | \theta) \)

Especially useful when the E and M steps have closed form solutions!!!
EM algorithm: Pictorial View

Given a set of Parameters and training data

Supervised learning problem

Relearn the parameters based on the new training data

Class assignment is probabilistic or weighted (soft EM)

Class assignment is hard (hard EM)

Estimate the class of each training example using the parameters yielding new (weighted) training data
Simple example: learn means only!

Consider:

• 1D data
• Mixture of k=2 Gaussians
• Variances fixed to $\sigma=1$
• Dist’n over classes is uniform
• Just need to estimate $\mu_1$ and $\mu_2$

$$P(x, y=i) \mu_{i=1}^m \exp \frac{1}{2} \|x - \mu_i\|^2 P(y=i)$$
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

$$p(y = i | x_j, \mu_1 \ldots \mu_k) \propto \exp\left( -\frac{1}{2\sigma^2} \| x_j - \mu_i \|^2 \right) P(y = i)$$

**M-step**

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

$$\mu_i = \frac{\sum_{j=1}^{m} P(y = i | x_j) x_j}{\sum_{j=1}^{m} P(y = i | x_j)}$$
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

$$P(y = i | x_j, \lambda_t) \propto p_i^{(t)} p(x_j | \mu_i^{(t)}, \Sigma_i^{(t)})$$

**M-step**

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

$$\mu_i^{(t+1)} = \frac{\sum_j P(y = i | x_j, \lambda_t) x_j}{\sum_j P(y = i | x_j, \lambda_t)}$$

$$\Sigma_i^{(t+1)} = \frac{\sum_j P(y = i | x_j, \lambda_t) [x_j - \mu_i^{(t+1)}] [x_j - \mu_i^{(t+1)}]^{T}}{\sum_j P(y = i | x_j, \lambda_t)}$$

$$p_i^{(t+1)} = \frac{\sum_j P(y = i | x_j, \lambda_t)}{\sum_j P(y = i | x_j, \lambda_t)}$$

$p_i^{(t)}$ is shorthand for estimate of $P(y=i)$ on $t$’th iteration

$\mu_i^{(t)}$ is shorthand for the $i$'th component of the mean vector at the $t$'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
What if we do hard assignments?

**Iterate:** On the \( t \)'th iteration let our estimates be
\[
\theta_t = \{ \mu_1^{(t)}, \mu_2^{(t)} \ldots \mu_k^{(t)} \}
\]

**E-step**
Compute “expected” classes of all datapoints
\[
p(y = i | x_j, \mu_1 \ldots \mu_k) \propto \exp \left( -\frac{1}{2\sigma^2} \| x_j - \mu_i \| ^2 \right) P(y = i)
\]

**M-step**
Compute most likely new \( \mu \)s given class expectations
\[
\mu_i = \frac{\sum_{j=1}^{m} P(y = i | x_j) x_j}{\sum_{j=1}^{m} P(y = i | x_j)}
\]

\( \delta \) represents hard assignment to “most likely” or nearest cluster

Equivalent to k-means clustering algorithm!!!
We will argue that EM:

• Optimizes a bound on the likelihood
• Is a type of coordinate ascent
• Is guaranteed to converge to a (often local) optima
The general learning problem with missing data

- **Marginal likelihood:** $\mathbf{x}$ is observed, $\mathbf{z}$ (e.g., class labels, $\mathbf{y}$) is missing:

$$
\ell(\theta : \mathcal{D}) = \log \prod_{j=1}^{m} P(x_j | \theta)
$$

$$
= \sum_{j=1}^{m} \log P(x_j | \theta)
$$

$$
= \sum_{j=1}^{m} \log \sum_{\mathbf{z}} P(x_j, \mathbf{z} | \theta)
$$

- **Objective:** Find $\arg\max_{\theta} l(\theta : \text{Data})$
A Key Computation: E-step

• $\mathbf{x}$ is observed, $\mathbf{z}$ is missing

• Compute probability of missing data given current choice of $\theta$
  
  – $Q(\mathbf{z} | \mathbf{x}_j)$ for each $\mathbf{x}_j$
    • e.g., probability computed during classification step
    • corresponds to “classification step” in K-means

\[
Q^{(t+1)}(\mathbf{z} | \mathbf{x}_j) = P(\mathbf{z} | \mathbf{x}_j, \theta^{(t)})
\]
Properties of EM

• We will prove that
  – EM converges to a local minima
  – Each iteration improves the log-likelihood

• How? (Same as k-means)
  – E-step can never decrease likelihood
  – M-step can never decrease likelihood
Jensen’s inequality

- **Theorem:** \( \log \sum_z P(z) f(z) \geq \sum_z P(z) \log f(z) \)
- *e.g.*, Binary case for convex function \( f \):

\[
f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2).
\]
Applying Jensen’s inequality

- Use: \( \log \sum_z P(z) f(z) \geq \sum_z P(z) \log f(z) \)

\[
\ell(\theta^{(t)} : \mathcal{D}) = \sum_{j=1}^{m} \log \sum_{z} Q^{(t+1)}(z | x_j) \frac{P(z, x_j | \theta^{(t)})}{Q^{(t+1)}(z | x_j)}
\]

\[
\geq \sum_{j=1}^{m} \sum_{z} Q^{(t+1)}(z | x_j) \log \left( \frac{p(z, x_j | \theta^{(t)})}{Q^{(t+1)}(z | x_j)} \right)
\]

\[
= \sum_{j=1}^{m} \sum_{z} Q^{(t+1)}(z | x_j) \log \left( p(z, x_j | \theta^{(t)}) \right) - \sum_{j=1}^{m} \sum_{z} Q^{(t+1)}(z | x_j) \log \left( Q^{(t+1)}(z | x_j) \right)
\]

\[
\ell(\theta^{(t)} : \mathcal{D}) \geq \sum_{j=1}^{m} \sum_{z} Q^{(t+1)}(z | x_j) \log P(z, x_j | \theta^{(t)}) + m \cdot H(Q^{(t+1)})
\]
The M-step

Lower bound:
\[
    \ell(\theta^{(t)} : \mathcal{D}) \geq \sum_{j=1}^{m} \sum_{z} Q^{(t+1)}(z | x_j) \log P(z, x_j | \theta^{(t)}) + m . H(Q^{(t+1)})
\]

• Maximization step:
\[
    \theta^{(t+1)} \leftarrow \arg \max_{\theta} \sum_{j=1}^{m} \sum_{z} Q^{(t+1)}(z | x_j) \log P(z, x_j | \theta)
\]

• We are optimizing a lower bound!
• Use expected counts to do weighted learning:
  – If learning requires \( \text{Count}(x,z) \)
  – Use \( E_{Q(t+1)}[\text{Count}(x,z)] \)
  – *Looks a bit like boosting***!!!
Convergence of EM

• Define: potential function $F(\theta, Q)$:

$$
\ell(\theta : \mathcal{D}) \geq F(\theta, Q) = \sum_{j=1}^{m} \sum_{z} Q(z \mid x_j) \log \frac{P(z, x_j \mid \theta)}{Q(z \mid x_j)}
$$

– lower bound from Jensen’s inequality

• EM is coordinate ascent on $F$!

– Thus, maximizes lower bound on marginal log likelihood
M-step can’t decrease $F(\theta, Q)$: by definition!

$$\theta^{(t+1)} \leftarrow \arg \max_{\theta} \sum_{j=1}^{m} \sum_{z} Q^{(t+1)}(z \mid x_j) \log P(z, x_j \mid \theta)$$

- We are maximizing $F$ directly, by ignoring a constant!

$$F(\theta, Q^{(t+1)}) = \sum_{j=1}^{m} \sum_{z} Q^{(t+1)}(z \mid x_j) \log P(z, x_j \mid \theta) + m \cdot H(Q^{(t+1)})$$
E-step: more work to show that $F(\theta,Q)$ doesn’t decrease

- **KL-divergence**: measures distance between distributions

\[
KL(Q\|P) = \sum_z Q(z) \log \frac{Q(z)}{P(z)}
\]

- KL=zero if and only if Q=P
E-step also doesn’t decrease $F$: Step 1

- Fix $\theta$ to $\theta^{(t)}$, take a max over $Q$:

$$\ell(\theta^{(t)} : D) \geq F(\theta^{(t)}, Q) = \sum_{j=1}^{m} \sum_{z} Q(z | x_j) \log \frac{P(z, x_j | \theta^{(t)})}{Q(z | x_j)}$$

$$= \sum_{j=1}^{m} \sum_{z} Q(z | x_j) \log \left( \frac{P(z | x_j, \theta^{(t)}) P(x_j | \theta^{(t)})}{Q(z | x_j)} \right)$$

$$= \sum_{j=1}^{m} \sum_{z} Q(z | x_j) \log \left( P(x_j | \theta^{(t)}) \right) - \sum_{j=1}^{m} \sum_{z} Q(z | x_j) \log \left( \frac{Q(z | x_j)}{P(z | x_j, \theta^{(t)})} \right)$$

$$= \ell(\theta^{(t)} : D) - \sum_{j=1}^{m} KL \left( Q(z | x_j) \| P(z | x_j, \theta^{(t)}) \right)$$
E-step also doesn’t decrease F: Step 2

• Fixing $\theta$ to $\theta^{(t)}$:

$$
\ell(\theta^{(t)} : \mathcal{D}) \geq F(\theta^{(t)}, Q) = \ell(\theta^{(t)} : \mathcal{D}) + \sum_{j=1}^{m} \sum_{z} Q(z | x_j) \log \frac{P(z | x_j, \theta^{(t)})}{Q(z | x_j)}
$$

$$
= \ell(\theta^{(t)} : \mathcal{D}) - \sum_{j=1}^{m} KL \left( Q(z | x_j) \parallel P(z | x_j, \theta^{(t)}) \right)
$$

• Now, the max over Q yields:

  – $Q(z | x_j) \leftarrow P(z | x_j, \theta^{(t)})$
  
  – Why? The likelihood term is a constant; the KL term is zero iff the arguments are the same distribution!!

  – So, the E-step is actually a maximization / tightening of the bound. It ensures that:

$$
F(\theta^{(t)}, Q^{(t+1)}) = \ell(\theta^{(t)} : \mathcal{D})
$$
EM is coordinate ascent

\[ \ell(\theta : \mathcal{D}) \geq F(\theta, Q) = \sum_{j=1}^{m} \sum_{z} Q(z \mid x_j) \log \frac{P(z, x_j \mid \theta)}{Q(z \mid x_j)} \]

- **M-step:** Fix \( Q \), maximize \( F \) over \( \theta \) (a lower bound on \( \ell(\theta : \mathcal{D}) \)):

\[ \ell(\theta : \mathcal{D}) \geq F(\theta, Q^{(t)}) = \sum_{j=1}^{m} \sum_{z} Q^{(t)}(z \mid x_j) \log P(z, x_j \mid \theta) + m \cdot H(Q^{(t)}) \]

- **E-step:** Fix \( \theta \), maximize \( F \) over \( Q \):

\[ \ell(\theta^{(t)} : \mathcal{D}) \geq F(\theta^{(t)}, Q) = \ell(\theta^{(t)} : \mathcal{D}) - m \sum_{j=1}^{m} KL \left( Q(z \mid x_j) \parallel P(z \mid x_j, \theta^{(t)}) \right) \]

- “Realigns” \( F \) with likelihood:

\[ F(\theta^{(t)}, Q^{(t+1)}) = \ell(\theta^{(t)} : \mathcal{D}) \]
What you should know

• K-means for clustering:
  – algorithm
  – converges because it’s coordinate ascent

• Know what agglomerative clustering is

• EM for mixture of Gaussians:
  – Also coordinate ascent
  – How to “learn” maximum likelihood parameters (locally max. like.) in the case of unlabeled data
  – Relation to K-means
    • Hard / soft clustering
    • Probabilistic model

• Remember, E.M. can get stuck in local minima,
  – And empirically it **DOES**
Acknowledgements

• K-means & Gaussian mixture models presentation contains material from excellent tutorial by Andrew Moore:
  – http://www.autonlab.org/tutorials/

• K-means Applet:
  – http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html/AppletKM.html

• Gaussian mixture models Applet:
  – http://www.neurosci.aist.go.jp/%7Eakahoe/MixtureEM.html