CS434a/541a: Pattern Recognition
Prof. Olga Veksler
Lecture 17
Today

- Parametric Unsupervised Learning
  - Expectation Maximization (EM)
    - one of the most useful statistical methods
    - oldest version in 1958 (Hartley)
    - seminal paper in 1977 (Dempster et al.)
    - can also be used when some samples are missing features
In unsupervised learning, where we are only given samples $x_1, \ldots, x_n$ without class labels.

Last 2 lectures: nonparametric approach (clustering)

Today, parametric approach
- assume parametric distribution of data
- estimate parameters of this distribution
- much “harder” than the supervised learning case
Parametric Unsupervised Learning

- Assume the data was generated by a model with known shape but unknown parameters

\[ P(x|\theta) \]

- Advantages of having a model
  - Gives a meaningful way to cluster data
    - adjust the parameters of the model to maximize the probability that the model produced the observed data
  - Can sensibly measure if a clustering is good
    - compute the likelihood of data induced by clustering
  - Can compare 2 clustering algorithms
    - which one gives the higher likelihood of the observed data?
Let us recall supervised parametric learning

- have $m$ classes
- have samples $x_1, \ldots, x_n$ each of class $1, 2, \ldots, m$
- suppose $D_i$ holds samples from class $i$
- probability distribution for class $i$ is $p_i(x|\theta_i)$
Parametric Supervised Learning

- Use the ML method to estimate parameters $\theta_i$
  - find $\theta_i$ which maximizes the likelihood function $F(\theta_i)$
    $$p(D_i \mid \theta_i) = \prod_{x \in D_i} p(x \mid \theta_i) = F(\theta_i)$$
  - or, equivalently, find $\theta_i$ which maximizes the log likelihood $l(\theta_i)$
    $$l(\theta_i) = \ln p(D_i \mid \theta_i) = \sum_{x \in D_i} \ln p(x \mid \theta_i)$$

$\hat{\theta}_1 = \arg\max_{\theta_1} [\ln p(D_1 \mid \theta_1)]$

$\hat{\theta}_2 = \arg\max_{\theta_2} [\ln p(D_2 \mid \theta_2)]$
Parametric Supervised Learning

- now the distributions are fully specified
- can classify unknown sample using MAP classifier
In unsupervised learning, no one tells us the true classes for samples. We still know:

- have $m$ classes
- have samples $x_1, \ldots, x_n$ each of unknown class
- probability distribution for class $i$ is $p_i(x|\theta_i)$

Can we determine the classes and parameters simultaneously?
Example: MRI Brain Segmentation

- In MRI brain image, different brain tissues have different intensities.
- Know that brain has 6 major types of tissues.
- Each type of tissue can be modeled by a Gaussian $\mathcal{N}(\mu_i, \sigma_i^2)$ reasonably well, parameters $\mu_i, \sigma_i^2$ are unknown.
- Segmenting (classifying) the brain image into different tissue classes is very useful.
  - don’t know which image pixel corresponds to which tissue (class).
  - don’t know parameters for each $\mathcal{N}(\mu_i, \sigma_i^2)$.

*Picture from M. Leventon*
Mixture Density Model

- Model data with *mixture density*

\[
p(x | \theta) = \sum_{j=1}^{m} p(x | c_j, \theta_j) P(c_j)
\]

- where \( \theta = \{\theta_1, \ldots, \theta_m\} \)
- \( P(c_1) + P(c_2) + \ldots + P(c_m) = 1 \)

- To generate a sample from distribution \( p(x | \theta) \)
  - first select class \( j \) with probability \( P(c_j) \)
  - then generate \( x \) according to probability law \( p(x | c_j, \theta_j) \)

![Diagram of Mixture Density Model](image)
Example: Gaussian Mixture Density

- Mixture of 3 Gaussians

\[
p_1(x) \equiv N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)
\]

\[
p_2(x) \equiv N\left(\begin{bmatrix} 6, 6 \end{bmatrix}, \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}\right)
\]

\[
p_3(x) \equiv N\left(\begin{bmatrix} 7, -7 \end{bmatrix}, \begin{bmatrix} 6 & 0 \\ 0 & 6 \end{bmatrix}\right)
\]

\[
p(x) = 0.2p_1(x) + 0.3p_2(x) + 0.5p_3(x)
\]
**Mixture Density**

\[ p(x \mid \theta) = \sum_{j=1}^{m} p(x \mid c_j, \theta_j) P(c_j) \]

- \( P(c_1), \ldots, P(c_m) \) can be known or unknown
- Suppose we know how to estimate \( \theta_1, \ldots, \theta_m \) and \( P(c_1), \ldots, P(c_m) \)
- Can “break apart” mixture \( p(x \mid \theta) \) for classification
- To classify sample \( x \), use MAP estimation, that is choose class \( i \) which maximizes

\[ P(c_i \mid x, \theta_i) \propto p(x \mid c_i, \theta_i) P(c_i) \]

- probability of component \( i \) to generate \( x \)
- probability of component \( i \)
**ML Estimation for Mixture Density**

\[ p(x \mid \theta, \rho) = \sum_{j=1}^{m} p(x \mid c_j, \theta_j) P(c_j) = \sum_{j=1}^{m} p(x \mid c_j, \theta_j) \rho_i \]

- Can use Maximum Likelihood estimation for a mixture density; need to estimate
  - \( \theta_1, \ldots, \theta_m \)
  - \( \rho_1 = P(c_1), \ldots, \rho_m = P(c_m) \), and \( \rho = \{\rho_1, \ldots, \rho_m\} \)
- As in the supervised case, form the logarithm likelihood function

\[ l(\theta, \rho) = \ln p(D \mid \theta, \rho) = \sum_{k=1}^{n} \ln p(x_k \mid \theta, \rho) = \sum_{k=1}^{n} \ln \left[ \sum_{j=1}^{m} p(x \mid c_j, \theta_j) \rho_i \right] \]
**ML Estimation for Mixture Density**

\[ I(\theta, \rho) = \sum_{k=1}^{n} \ln \left[ \sum_{j=1}^{m} p(x | c_j, \theta_j) \rho_i \right] \]

- need to maximize \( I(\theta, \rho) \) with respect to \( \theta \) and \( \rho \)
- As you may have guessed, \( I(\theta, \rho) \) is not the easiest function to maximize
  - If we take partial derivatives with respect to \( \theta, \rho \) and set them to \( \theta, \rho \), typically we have a “coupled” nonlinear system of equation
  - usually closed form solution cannot be found
- We could use the gradient ascent method
  - in general, it is not the greatest method to use, should only be used as last resort
- There is a better algorithm, called **EM**
Before EM, let’s look at the mixture density again

\[ p(x \mid \theta, \rho) = \sum_{j=1}^{m} p(x \mid c_j, \theta_j) \rho_j \]

Suppose we know how to estimate \( \theta_1, \ldots, \theta_m \) and \( \rho_1, \ldots, \rho_m \)

- Estimating the class of \( x \) is easy with MAP, maximize

\[ p(x \mid c_i, \theta_i) P(c_i) = p(x \mid c_i, \theta_i) \rho_i \]

Suppose we know the class of samples \( x_1, \ldots, x_n \)

- This is just the supervised learning case, so estimating \( \theta_1, \ldots, \theta_m \) and \( \rho_1, \ldots, \rho_m \) is easy

\[
\hat{\theta}_i = \arg\max_{\theta_i} \left[ \ln p(D_i \mid \theta_i) \right] \\
\hat{\rho}_i = \frac{|D_i|}{n}
\]

This is an example of chicken-and-egg problem

- ME algorithm approaches this problem by adding “hidden” variables
**Expectation Maximization Algorithm**

- **EM** is an algorithm for ML parameter estimation when the data has missing values. It is used when
  1. data is incomplete (has missing values)
     - some features are missing for some samples due to data corruption, partial survey responses, etc.
     - This scenario is very useful, covered in section 3.9
  2. Suppose data $X$ is complete, but $p(X|\theta)$ is hard to optimize. Suppose further that introducing certain hidden variables $U$ whose values are missing, and suppose it is easier to optimize the “complete” likelihood function $p(X,U|\theta)$. Then EM is useful.
     - This scenario is useful for the mixture density estimation, and is subject of our lecture today
- Notice that after we introduce artificial (hidden) variables $U$ with missing values, case 2 is completely equivalent to case 1
EM: Hidden Variables for Mixture Density

\[ p(x \mid \theta) = \sum_{j=1}^{m} p(x \mid c_j, \theta_j) \rho_j \]

- For simplicity, assume component densities are
  \[ p(x \mid c_j, \theta_j) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left( -\frac{(x - \mu_j)^2}{2\sigma^2} \right) \]
  - assume for now that the variance is known
  - need to estimate \( \theta = \{\mu_1, \ldots, \mu_m\} \)

- If we knew which sample came from which component (that is the class label), the ML parameter estimation is easy

- Thus to get an easier problem, introduce hidden variables which indicate which component each sample belongs to
EM: Hidden Variables for Mixture Density

- For $1 \leq i \leq n$, $1 \leq k \leq m$, define hidden variables $z_{i}^{(k)}$
  
  $z_{i}^{(k)} = \begin{cases} 
  1 & \text{if sample } i \text{ was generated by component } k \\
  0 & \text{otherwise} 
  \end{cases}$

  $x_i \rightarrow \{x_i, z_i^{(1)}, \ldots, z_i^{(m)}\}$

- $z_{i}^{(k)}$ are indicator random variables, they indicate which Gaussian component generated sample $x_i$

- Let $z_i = \{z_i^{(1)}, \ldots, z_i^{(m)}\}$, indicator r.v. corresponding to sample $x_i$

- Conditioned on $z_i$, distribution of $x_i$ is Gaussian
  
  $p(x_i | z_i, \theta) \sim N(\mu_k, \sigma^2)$

  - where $k$ is s.t. $z_i^{(k)} = 1$
EM: Joint Likelihood

- Let $z_i = \{z_i^{(1)}, \ldots, z_i^{(m)}\}$, and $Z = \{z_1, \ldots, z_n\}$
- The complete likelihood is
  \[
p(X, Z | \theta) = \prod_{i=1}^{n} p(x_i, z_i | \theta) = \prod_{i=1}^{n} p(x_i | z_i, \theta) p(z_i)
  \]
  gaussian part of $\rho_c$
- If we actually observed $Z$, the log likelihood $\ln[p(X, Z | \theta)]$ would be trivial to maximize with respect to $\theta$ and $\rho_i$
- The problem, is, of course, is that the values of $Z$ are missing, since we made it up (that is $Z$ is hidden)
Instead of maximizing $\ln[p(X,Z|\theta)]$ the idea behind EM is to maximize some function of $\ln[p(X,Z|\theta)]$, usually its expected value

$$E_Z[\ln p(X,Z | \theta)]$$

- If $\theta$ makes $\ln[p(X,Z|\theta)]$ large, then $\theta$ tends to make $E[\ln p(X,Z|\theta)]$ large
- The expectation is with respect to the missing data $Z$
- That is with respect to density $p(Z|X,\theta)$

However $\theta$ is our ultimate goal, we don’t know $\theta$!
EM Algorithm

- EM solution is to iterate
  1. start with initial parameters $\theta^{(0)}$

iterate the following 2 step until convergence

E. compute the expectation of log likelihood with respect to current estimate $\theta^{(t)}$ and $X$. Let’s call it $Q(\theta | \theta^{(t)})$

$$Q(\theta | \theta^{(t)}) = \mathbb{E}_Z[\ln p(X,Z | \theta) | X, \theta^{(t)}]$$

M. maximize $Q(\theta | \theta^{(t)})$

$$\theta^{(t+1)} = \arg\max_{\theta} Q(\theta | \theta^{(t)})$$
EM Algorithm: Picture

In $\log p(X | \theta)$ we’d like to find it but optimizing $p(X | \theta)$ is very difficult.
This curve corresponds to the correct $Z$, we should optimize for but $Z$ is not observed.

For mixture estimation, there are $m^n$ curves, each curve corresponds to a particular assignment of samples to classes.
EM Algorithm: Picture

\[ \ln p(X, Z | \theta) \]

\[ E_Z[\ln p(X, Z | \theta) | X, \theta^{(t)}] \]

\[ E_Z(\theta) \]
**EM Algorithm**

- It can be proven that EM algorithm converges to the local maximum of the log-likelihood
  \[ \ln p(X | \theta) \]

- Why is it better than gradient ascent?
  - Convergence of EM is usually **significantly** faster, in the beginning, very large steps are made (that is likelihood function increases rapidly), as opposed to gradient ascent which usually takes tiny steps
  - gradient descent is not guaranteed to converge
  - recall all the difficulties of choosing the appropriate learning rate
**EM for Mixture of Gaussians: E step**

- Let’s come back to our example $p(x | \theta) = \sum_{j=1}^{m} p(x | c_j, \theta_j) \rho_j$
  
  $$p(x | c_j, \theta_j) = \frac{1}{\sigma \sqrt{2\pi}} \exp \left( -\frac{(x - \mu_j)^2}{2\sigma^2} \right)$$

- need to estimate $\theta = \{\mu_1, \ldots, \mu_m\}$ and $\rho_1, \ldots, \rho_m$

- for $1 \leq i \leq n$, $1 \leq k \leq m$, define $z_i^{(k)}$
  
  $$z_i^{(k)} = \begin{cases} 1 & \text{if sample } i \text{ was generated by component } k \\ 0 & \text{otherwise} \end{cases}$$

- as before, $z_i = \{z_i^{(1)}, \ldots, z_i^{(m)}\}$, and $Z = \{z_1, \ldots, z_n\}$

- We need log-likelihood of observed $X$ and hidden $Z$
  
  $$\ln p(X, Z | \theta) = \ln \prod_{i=1}^{n} p(x_i, z_i | \theta) = \sum_{i=1}^{n} \ln p(x_i | z_i, \theta) \rho(z_i)$$
**EM for Mixture of Gaussians: E step**

- We need log-likelihood of observed $X$ and hidden $Z$

$$
\ln p(X, Z \mid \theta) = \ln \prod_{i=1}^{n} p(x_i, z_i \mid \theta) = \sum_{i=1}^{n} \ln p(x_i \mid z_i, \theta) P(z_i)
$$

- First let’s rewrite $p(x_i \mid z_i, \theta) P(z_i)$

$$
p(x_i \mid z_i, \theta) P(z_i) = \begin{cases} 
p(x_i \mid z_i^{(1)} = 1, \theta) P(z_i^{(1)} = 1) & \text{if } z_i^{(1)} = 1 \\
p(x_i \mid z_i^{(m)} = 1, \theta) P(z_i^{(m)} = 1) & \text{if } z_i^{(m)} = 1 
\end{cases}
$$

$$
= \prod_{k=1}^{m} \left[ p(x_i \mid z_i^{(k)} = 1, \theta) P(z_i^{(k)} = 1) \right]^{z_i^{(k)}}
$$

$$
= \prod_{k=1}^{m} \left[ \frac{1}{\sigma \sqrt{2\pi}} \exp \left( -\frac{(x_i - \mu_k)^2}{2\sigma^2} \right) P(z_i^{(k)} = 1) \right]^{z_i^{(k)}}
$$
EM for Mixture of Gaussians: E step

- log-likelihood of observed $X$ and hidden $Z$ is

$$\ln p(X, Z | \theta) = \sum_{i=1}^{n} \ln p(x_i | z_i, \theta)P(z_i)$$

$$= \sum_{i=1}^{n} \ln \prod_{k=1}^{m} \left[ \frac{1}{\sigma \sqrt{2\pi}} \exp \left( -\frac{(x_i - \mu_k)^2}{2\sigma^2} \right) \right] P(z_i^{(k)} = 1)$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{m} \ln \left[ \frac{1}{\sigma \sqrt{2\pi}} \exp \left( -\frac{(x_i - \mu_k)^2}{2\sigma^2} \right) \right] P(z_i^{(k)} = 1)$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{m} z_i^{(k)} \left[ \ln \frac{1}{\sigma \sqrt{2\pi}} - \frac{(x_i - \mu_k)^2}{2\sigma^2} + \ln P(z_i^{(k)} = 1) \right] + \ln P(c_k = \rho_k)$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{m} z_i^{(k)} \left[ \ln \frac{1}{\sigma \sqrt{2\pi}} - \frac{(x_i - \mu_k)^2}{2\sigma^2} + \ln \rho_k \right]$$
EM for Mixture of Gaussians: E step

- log-likelihood of observed $X$ and hidden $Z$ is

$$\ln p(X, Z | \theta) = \sum_{i=1}^{n} \sum_{k=1}^{m} z_{i}^{(k)} \left[ \ln \frac{1}{\sigma \sqrt{2\pi}} - \frac{(x_i - \mu_k)^2}{2\sigma^2} + \ln \rho_k \right]$$

- For the E step, we must compute

$$Q(\theta | \theta^{(t)}) = Q(\theta | \mu_1^{(t)}, ..., \mu_m^{(t)}, \rho_1^{(t)}, ..., \rho_m^{(t)}) = E_z[ \ln p(X, Z | \theta) | X, \theta^{(t)} ]$$

$$= E_z \left( \sum_{i=1}^{n} \sum_{k=1}^{m} z_{i}^{(k)} \left[ \ln \frac{1}{\sigma \sqrt{2\pi}} - \frac{(x_i - \mu_k)^2}{2\sigma^2} + \ln \rho_k^{(t)} \right] \right)$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{m} E_z[z_{i}^{(k)}] \left[ \ln \frac{1}{\sigma \sqrt{2\pi}} - \frac{(x_i - \mu_k)^2}{2\sigma^2} + \ln \rho_k \right]$$

$$= E_x \left[ \sum_i a_i x_i + b \right] = \sum_i a_i E_x [x_i] + b$$
EM for Mixture of Gaussians: E step

$$Q(\theta | \theta^{(t)}) = \sum_{i=1}^{n} \sum_{k=1}^{m} E_{Z}[z_{i}^{(k)}] \left( \ln \frac{1}{\sigma \sqrt{2\pi}} - \frac{(x_i - \mu_k)^2}{2\sigma^2} + \ln \rho_k \right)$$

- need to compute $E_{Z}[z_{i}^{(k)}]$ in the above expression

$$E_{Z}[z_{i}^{(k)}] = 0 \cdot P(z_{i}^{(k)} = 0 | \theta^{(t)}, x_i) + 1 \cdot P(z_{i}^{(k)} = 1 | \theta^{(t)}, x_i)$$

$$= P(z_{i}^{(k)} = 1 | \theta^{(t)}, x_i) = \frac{p(x_i | \theta^{(t)}, z_{i}^{(k)} = 1)P(z_{i}^{(k)} = 1 | \theta^{(t)})}{p(x_i | \theta^{(t)})}$$

$$= \frac{\rho_k^{(t)} \exp \left( -\frac{1}{2} (x_i - \mu_k^{(t)})^2 \right)}{\sum_{j=1}^{m} p(x_i | \theta^{(t)}, z_{i}^{(j)} = 1)P(z_{i}^{(j)} = 1 | \theta^{(t)})} = \frac{\rho_k^{(t)} \exp \left( -\frac{1}{2\sigma^2} (x_i - \mu_k^{(t)})^2 \right)}{\sum_{j=1}^{m} \rho_j^{(t)} \exp \left( -\frac{1}{2\sigma^2} (x_i - \mu_j^{(t)})^2 \right)}$$

- we are finally done with the E step
  - for implementation, just need to compute $E_{Z}[z_{i}^{(k)}]$’s don’t need to compute $Q$
EM for Mixture of Gaussians: $M$ step

\[ Q(\theta | \theta^{(t)}) = \sum_{i=1}^{n} \sum_{k=1}^{m} E_{z_i^{(k)}} \left( \ln \frac{1}{\sigma \sqrt{2\pi}} - \frac{(x_i - \mu_k)^2}{2\sigma^2} + \ln \rho_k \right) \]

- Need to maximize $Q$ with respect to all parameters
- First differentiate with respect to $\mu_k$

\[ \frac{\partial}{\partial \mu_k} Q(\theta | \theta^{(t)}) = \sum_{i=1}^{n} E_{z_i^{(k)}} \frac{(x_i - \mu_k)}{\sigma^2} = 0 \]

\[ \Rightarrow \text{new } \mu_k = \mu_k^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n} E_{z_i^{(k)}} x_i \]

the mean for class $k$ is weighted average of all samples, and this weight is proportional to the current estimate of probability that the sample belongs to class $k$
EM for Mixture of Gaussians: M step

\[ Q(\theta | \theta^{(t)}) = \sum_{i=1}^{n} \sum_{k=1}^{m} E_z[z_i^{(k)}] \left( \ln \frac{1}{\sigma \sqrt{2\pi}} - \frac{(x_i - \mu_k)^2}{2\sigma^2} + \ln \rho_k \right) \]

- For \( \rho_k \) we have to use Lagrange multipliers to preserve constraint \( \sum_{j=1}^{m} \rho_j = 1 \)
- Thus we need to differentiate \( F(\lambda, \rho) = Q(\theta | \theta^{(t)}) - \lambda \left( \sum_{j=1}^{m} \rho_j - 1 \right) \)

\[ \frac{\partial}{\partial \rho_k} F(\lambda, \rho) = \sum_{i=1}^{n} \frac{1}{\rho_k} E_z[z_i^{(k)}] - \lambda = 0 \Rightarrow \sum_{i=1}^{n} E_z[z_i^{(k)}] - \lambda \rho_k = 0 \]

- Summing up over all components: \( \sum_{k=1}^{m} \sum_{i=1}^{n} E_z[z_i^{(k)}] = \sum_{k=1}^{m} \lambda \rho_k \)
- Since \( \sum_{k=1}^{m} \sum_{i=1}^{n} E_z[z_i^{(k)}] = n \) and \( \sum_{k=1}^{m} \rho_k = 1 \) we get \( \lambda = n \)

\[ \rho_k^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n} E_z[z_i^{(k)}] \]
The algorithm on this slide applies ONLY to univariate gaussian case with known variances

1. Randomly initialize $\mu_1, \ldots, \mu_m$, $\rho_1, \ldots, \rho_m$ (with constraint $\sum \rho_i = 1$)

**iterate** until no change in $\mu_1, \ldots, \mu_m$, $\rho_1, \ldots, \rho_m$

E. for all $i, k$, compute

$$E_z[z_i^{(k)}] = \frac{\rho_k \exp\left(-\frac{1}{2\sigma^2} (x_i - \mu_k)^2 \right)}{\sum_{j=1}^m \rho_j \exp\left(-\frac{1}{2\sigma^2} (x_i - \mu_j)^2 \right)}$$

M. for all $k$, do parameter update

$$\mu_k = \frac{1}{n} \sum_{i=1}^n E_z[z_i^{(k)}] x_i \quad \rho_k = \frac{1}{n} \sum_{i=1}^n E_z[z_i^{(k)}]$$
**EM Algorithm**

- For the more general case of multivariate Gaussians with unknown means and variances

  - **E step:**  \[ E_z[z_i^{(k)}] = \frac{\rho_k p(x | \mu_k, \Sigma_k)}{\sum_{j=1}^{m} \rho_j p(x | \mu_j, \Sigma_j)} \]

    where  \[ p(x | \mu_k, \Sigma_k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left[-\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k)\right] \]

  - **M step:**

    \[ \rho_k = \frac{1}{n} \sum_{i=1}^{n} E_z[z_i^{(k)}] \]
    \[ \mu_k = \frac{\sum_{i=1}^{n} E_z[z_i^{(k)}] x_i}{\sum_{i=1}^{n} E_z[z_i^{(k)}]} \]

    \[ \Sigma_k = \frac{\sum_{i=1}^{n} E_z[z_i^{(k)}] (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^{n} E_z[z_i^{(k)}]} \]
**EM Algorithm and K-means**

- *k*-means can be derived from EM algorithm
- Setting mixing parameters equal for all classes,

$$E_z[z_i^{(k)}] = \frac{\rho_k \exp \left( -\frac{1}{2\sigma^2} (x_i - \mu_k)^2 \right)}{\sum_{j=1}^{m} \rho_j \exp \left( -\frac{1}{2\sigma^2} (x_i - \mu_j)^2 \right)} = \frac{\exp \left( -\frac{1}{2\sigma^2} (x_i - \mu_k)^2 \right)}{\sum_{j=1}^{m} \exp \left( -\frac{1}{2\sigma^2} (x_i - \mu_j)^2 \right)}$$

- If we let $\sigma \to \infty$, then

$$E_z[z_i^{(k)}] = \begin{cases} 1 & \text{if } \forall j, \| x_i - \mu_k \| > \| x_i - \mu_j \| \\ 0 & \text{otherwise} \end{cases}$$

- so at the *E* step, for each current mean, we find all points closest to it and form new clusters
- at the *M* step, we compute the new means inside current clusters

$$\mu_k = \frac{1}{n} \sum_{i=1}^{n} E_z[z_i^{(k)}] x_i$$
EM Gaussian Mixture Example
EM Gaussian Mixture Example

After first iteration
EM Gaussian Mixture Example

After second iteration
EM Gaussian Mixture Example

After third iteration
After 20th iteration
EM Example

- Example from R. Gutierrez-Osuna
- Training set of 900 examples forming an annulus
- Mixture model with $m = 30$ Gaussian components of unknown mean and variance is used

Training:
- Initialization:
  - means to 30 random examples
  - covariance matrices initialized to be diagonal, with large variances on the diagonal (compared to the training data variance)
- During EM training, components with small mixing coefficients were trimmed
  - This is a trick to get in a more compact model, with fewer than 30 Gaussian components
EM Example

from R. Gutierrez-Osuna
EM Texture Segmentation Example

Figure from “Color and Texture Based Image Segmentation Using EM and Its Application to Content Based Image Retrieval”, S.J. Belongie et al., ICCV 1998
EM Motion Segmentation Example

Three frames from the MPEG “flower garden” sequence

Figure from “Representing Images with layers,” by J. Wang and E.H. Adelson, IEEE Transactions on Image Processing, 1994, c 1994, IEEE
Summary

- Advantages
  - If the assumed data distribution is correct, the algorithm works well

- Disadvantages
  - If assumed data distribution is wrong, results can be quite bad.
    - In particular, bad results if use incorrect number of classes (i.e. the number of mixture components)