CS840a
Learning and Computer Vision
Prof. Olga Veksler

Lecture 5

Unsupervised Learning

Today

- New Topic: *Unsupervised Learning*
  - Supervised vs. unsupervised learning
  - Unsupervised learning
    - nonparametric unsupervised learning = clustering
      - Proximity Measures
      - Criterion Functions
      - k-means
    - parametric unsupervised learning
      - Expectation Maximization (EM)

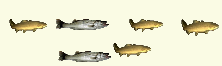
Supervised vs. Unsupervised Learning

- Up to now we considered ***supervised learning*** scenario, where we are given
  1. samples $\boldsymbol{x_1},\ldots,\boldsymbol{x_n}$
  2. class label $\boldsymbol{y_i}$ for all samples $\boldsymbol{x_i}$
  - This is also called learning with teacher, since correct answer (the true class) is provided

- In the next few lectures we consider ***unsupervised learning*** scenario, where we are only given
  1. samples $\boldsymbol{x_1},\ldots,\boldsymbol{x_n}$
  - This is also called learning without teacher, since correct answer is not provided
  - do not split data into training and test sets

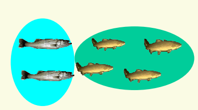
Unsupervised Learning

- Data is *not labeled*



a lot is known
"easier"

1. Parametric Approach
   - assume parametric distribution of data
   - estimate parameters of this distribution
   - much "harder" than supervised case
- NonParametric Approach
  - group the data into ***clusters***, each cluster (hopefully) says something about categories (classes) present in the data



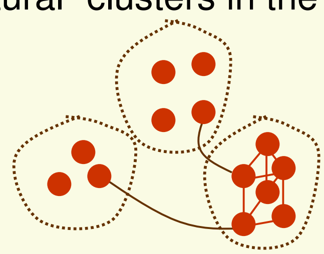
little is known
"harder"

Why Unsupervised Learning?

- Unsupervised learning is harder
  - How do we know if results are meaningful? No answer labels are available.
    - Let the expert look at the results (external evaluation)
    - Define an objective function on clustering (internal evaluation)
- We nevertheless need it because
  1. Labeling large datasets is very costly (speech recognition)
     - sometimes can label only a few examples by hand
  2. May have no idea what/how many classes there are (data mining)
  3. May want to use clustering to gain some insight into the structure of the data before designing a classifier
     - Clustering as data description

Clustering

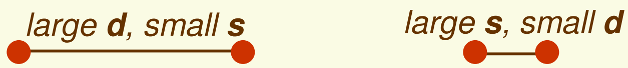
- Seek "natural" clusters in the data



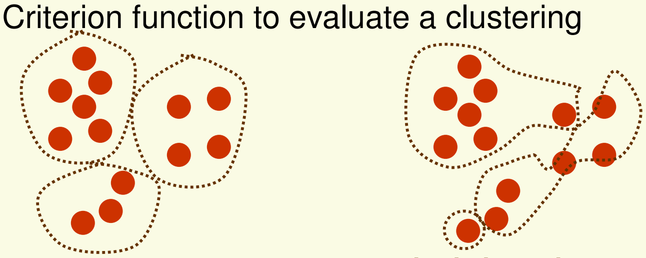
- What is a good clustering?
  - internal (within the cluster) distances should be small
  - external (intra-cluster) should be large
- Clustering is a way to discover new categories (classes)

What we Need for Clustering

1. Proximity measure*, either*
    - similarity measure $\mathbf{s}(\mathbf{x}_i,\mathbf{x}_k)$: large if $\mathbf{x}_i,\mathbf{x}_k$ are similar
    - dissimilarity(or distance) measure $\mathbf{d}(\mathbf{x}_i,\mathbf{x}_k)$: small if $\mathbf{x}_i,\mathbf{x}_k$ are similar



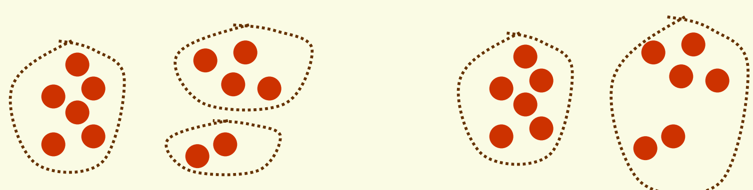
2. Criterion function to evaluate a clustering



good clustering *bad clustering*

3. Algorithm to compute clustering
    - For example, by optimizing the criterion function

How Many Clusters?

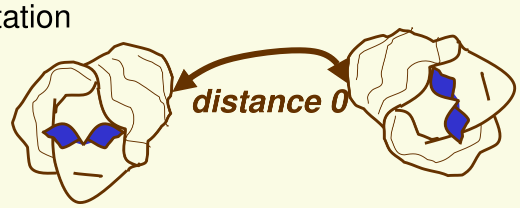


3 clusters or 2 clusters?

- Possible approaches
    1. fix the number of clusters to ***k***
    2. find the best clustering according to the criterion function (number of clusters may vary)

Proximity Measures

- good proximity measure is VERY application dependent
  - Clusters should be invariant under the transformations "natural" to the problem
  - For example for object recognition, should have invariance to rotation



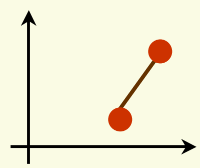
  - For character recognition, no invariance to rotation

Distance (dissimilarity) Measures

- Euclidean distance

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{d} \left(x_i^{(k)} - x_j^{(k)}\right)^2}$$

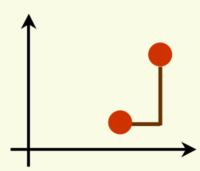
  - translation invariant



- Manhattan (city block) distance

$$d(x_i, x_j) = \sum_{k=1}^{d} \left|x_i^{(k)} - x_j^{(k)}\right|$$

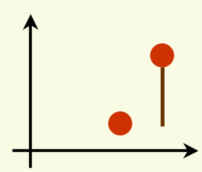
  - approximation to Euclidean distance, cheaper to compute



- Chebyshev distance

$$d(x_i, x_j) = \max_{1 \le k \le d} |\, x_i^{(k)} - x_j^{(k)} \,|$$

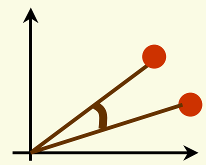
  - approximation to Euclidean distance, cheapest to compute



Similarity Measures

- Cosine similarity:

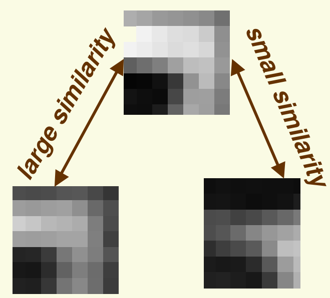
$$s(x_i, x_j) = \frac{x_i^T x_j}{\| x_i \| \, \| x_j \|}$$



  - the smaller the angle, the larger the similarity
  - scale invariant measure
  - popular in text retrieval

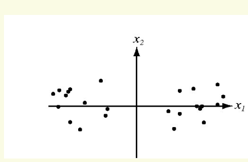
- Correlation coefficient
  - popular in image processing

$$s(x_i, x_j) = \frac{\sum_{k=1}^{d} \left(x_i^{(k)} - \bar{x}_i\right)\left(x_i^{(k)} - \bar{x}_i\right)}{\left[\sum_{k=1}^{d} \left(x_i^{(k)} - \bar{x}_i\right)^2 \sum_{k=1}^{d} \left(x_j^{(k)} - \bar{x}_j\right)^2\right]^{1/2}}$$

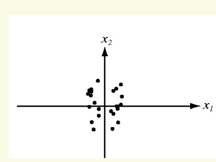


Feature Scale

- old problem: how to choose appropriate relative scale for features?
  - [length (in meters or cms?), weight(in in grams or kgs?)]
  - In supervised learning, can normalize to zero mean unit variance with no problems
  - in clustering this is more problematic, ***if variance in data is due to cluster presence, then normalizing features is not a good thing***



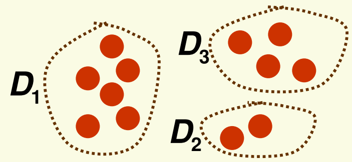
before normalization



after normalization

Criterion Functions for Clustering

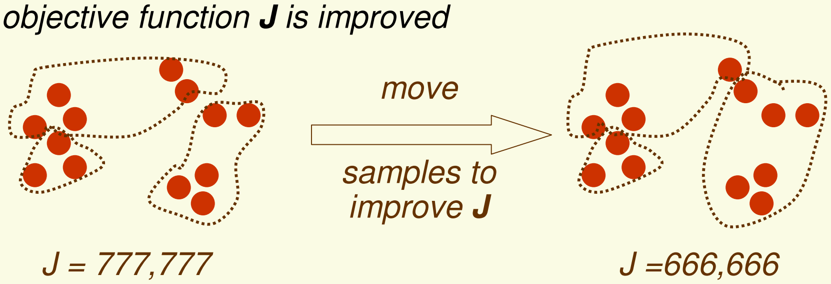
- Have samples $\mathbf{x}_1,\ldots,\mathbf{x}_n$
- Suppose partitioned samples into c subsets $D_1,\ldots,D_c$



- There are approximately $c^n/c!$ distinct partitions
- Can define a criterion function $J(D_1,\ldots,D_c)$ which measures the quality of a partitioning $D_1,\ldots,D_c$
- Then the clustering problem is a well defined problem
  - the optimal clustering is the partition which optimizes the criterion function

Iterative Optimization Algorithms

- Now have both proximity measure and criterion function, need algorithm to find the optimal clustering
- Exhaustive search is impossible, since there are approximately $c^n/c!$ possible partitions
- Usually some iterative algorithm is used
  1. Find a reasonable initial partition
  2. Repeat: *move samples from one group to another s.t. the objective function **J** is improved*



K-means Clustering

- Iterative clustering algorithm
- Want to optimize the $\boldsymbol{J_{SSE}}$ objective function

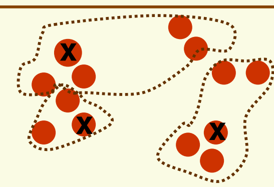
$$J_{SSE} = \sum_{i=1}^{k} \sum_{x \in D_i} \| \boldsymbol{x} - \mu_i \|^2$$

  - for a different objective function, we need a different optimization algorithm, of course
- Fix number of clusters to $\boldsymbol{k}$ ($\boldsymbol{c} = \boldsymbol{k}$)
- ***k***-means is probably the most famous clustering algorithm
  - it has a smart way of moving from current partitioning to the next one

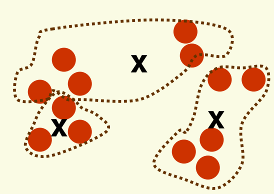
K-means Clustering

***k* = 3**

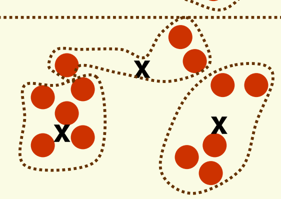
1. Initialize
   - pick ***k*** cluster centers arbitrary
   - assign each example to closest center



2. compute sample means for each cluster



3. reassign all samples to the closest mean

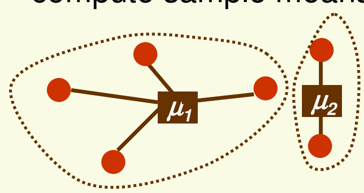


4. if clusters changed at step 3, go to step 2

K-means Clustering

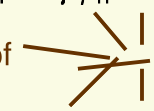
- Consider steps *2* and *3* of the algorithm

2. compute sample means for each cluster

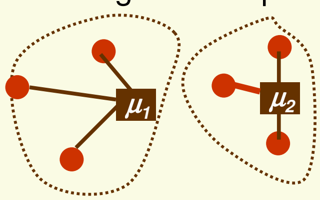


$$J_{SSE} = \sum_{i=1}^{k} \sum_{x \in D_i} || x - \mu_i ||^2$$

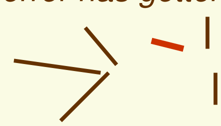
= sum of



3. reassign all samples to the closest mean

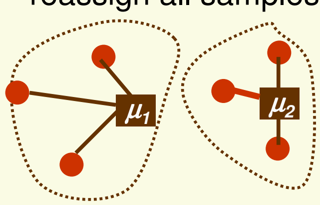


If we represent clusters by their old means, the error has gotten smaller

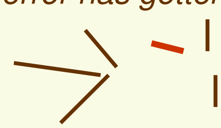


K-means Clustering

3. reassign all samples to the closest mean



If we represent clusters by their old means, the error has gotten smaller



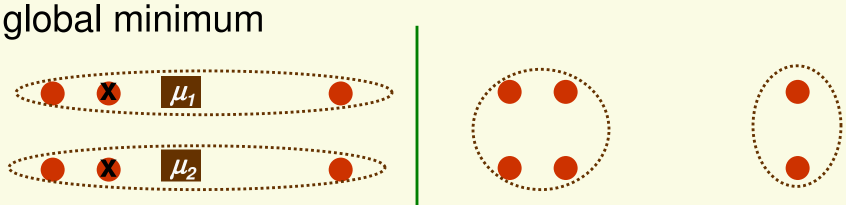
- However we represent clusters by their new means, and mean is always the smallest representation of a cluster

$$\frac{\partial}{\partial z} \sum_{x \in D_i} \frac{1}{2} || x - z ||^2 = \frac{\partial}{\partial z} \sum_{x \in D_i} \frac{1}{2} \left(|| x ||^2 - 2x^t z + || z ||^2 \right) = \sum_{x \in D_i} \left(-x + z \right) = 0$$

$$\Rightarrow z = \frac{1}{n_i} \sum_{x \in D_i} x$$

K-means Clustering

- We just proved that by doing steps *2* and *3*, the objective function goes down
  - in two step, we found a "smart " move which decreases the objective function
- Thus the algorithm converges after a finite number of iterations of steps *2* and *3*
- However the algorithm is not guaranteed to find a global minimum



2-means gets stuck here

global minimum of J_{SSE}

K-means Clustering

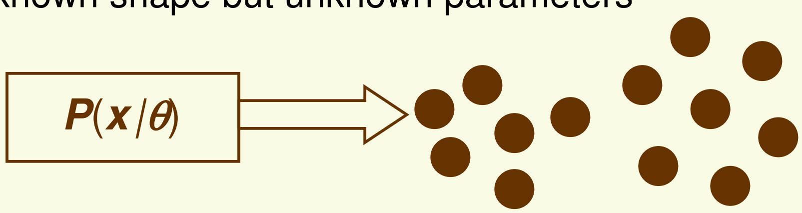
- Finding the optimum of J_{SSE} is NP-hard
- In practice, **k**-means clustering performs usually well
- It is very efficient
- Its solution can be used as a starting point for other clustering algorithms
- Still 100's of papers on variants and improvements of **k**-means clustering every year

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

Parametric Unsupervised Learning

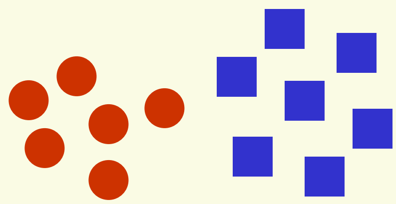
- Assume the data was generated by a model with known shape but unknown parameters



- 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?

Parametric Supervised Learning

- Let us recall supervised parametric learning
  - have $\boldsymbol{m}$ classes
  - have samples $\boldsymbol{x_1},\ldots,\boldsymbol{x_n}$ each of class $\boldsymbol{1}, \boldsymbol{2},\ldots,\boldsymbol{m}$
  - suppose $\boldsymbol{D_i}$ holds samples from class $\boldsymbol{i}$
  - probability distribution for class $\boldsymbol{i}$ is $\boldsymbol{p_i}(\boldsymbol{x}|\boldsymbol{\theta_i})$



$p_1(x|\theta_1)$ $p_2(x|\theta_2)$

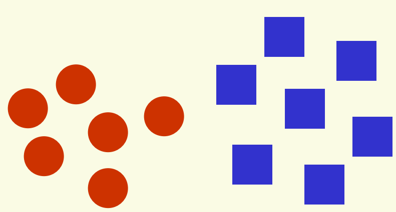
Parametric Supervised Learning

- Use the ML method to estimate parameters $\boldsymbol{\theta_i}$
  - find $\boldsymbol{\theta_i}$ which maximizes the likelihood function $\boldsymbol{F}(\boldsymbol{\theta_i})$

$$p(D_i \,/\, \theta_i) = \prod_{x \in D_i} p(x \,/\, \theta_i) = F(\theta_i)$$

  - or, equivalently, find $\boldsymbol{\theta_i}$ which maximizes the log likelihood $\boldsymbol{l}(\boldsymbol{\theta_i})$

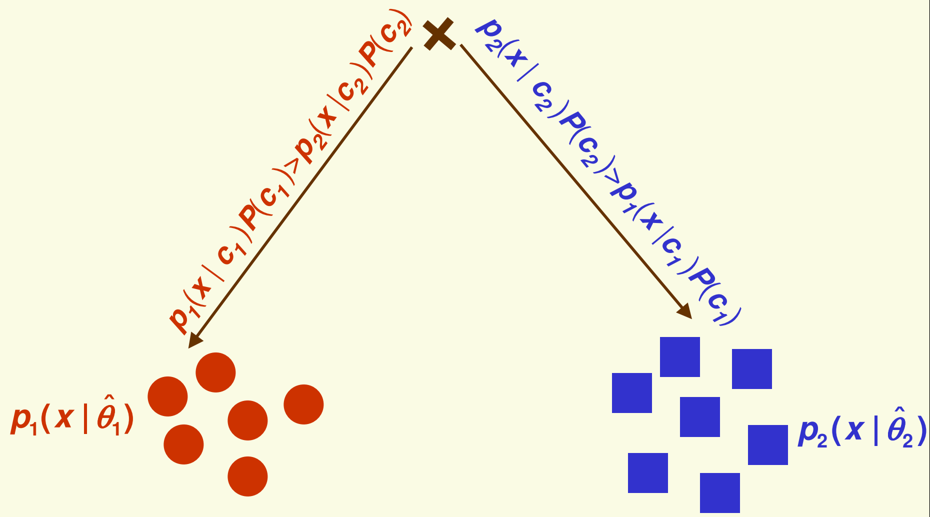
$$l(\theta_i) = \ln p(D_i \mid \theta_i) = \sum_{x \in D_i} \ln p(x \mid \theta_i)$$



$$\hat{\theta}_1 = \underset{\theta_1}{\arg\max}\left[\ln p(D_1 \mid \theta_1)\right] \qquad \hat{\theta}_2 = \underset{\theta_2}{\arg\max}\left[\ln p(D_2 \mid \theta_2)\right]$$

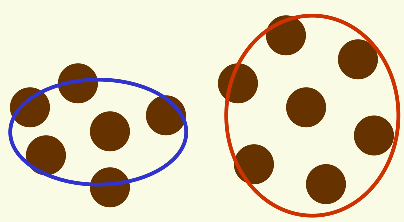
Parametric Supervised Learning

- now the distributions are fully specified
- can classify unknown sample using MAP classifier

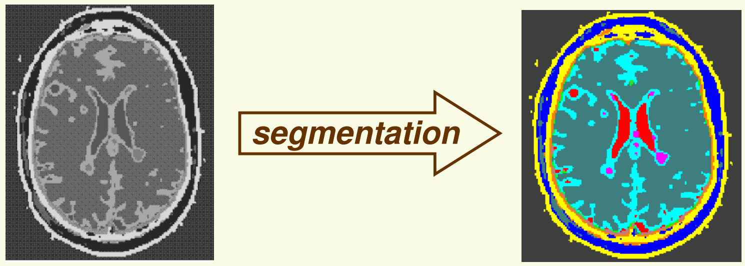


Parametric Unsupervised Learning

- In unsupervised learning, no one tells us the true classes for samples. We still know
  - have $\boldsymbol{m}$ classes
  - have samples $\boldsymbol{x_1},\dots,\boldsymbol{x_n}$ each of ***unknown*** class
  - probability distribution for class $\boldsymbol{i}$ is $\boldsymbol{p_i}(\boldsymbol{x}|\boldsymbol{\theta_i})$
- Can we determine the classes and parameters simultaneously?



Example: MRI Brain Segmentation



Picture from M. Leventon

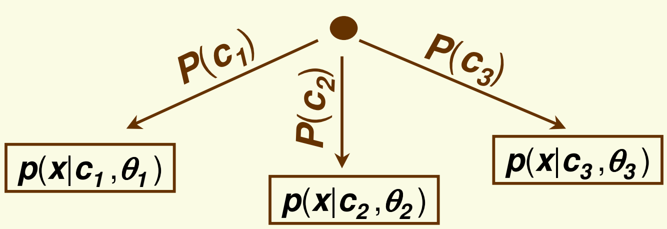
- In MRI brain image, different brain tissues have different intensities
- Know that brain has 6 major types of tissues
- Each type of tissue can be modeled by a Gaussian $\boldsymbol{N}(\mu_i, \sigma_i^2)$ reasonably well, parameters μ_i, σ_i^2 are unknown
- Segmenting (classifying) the brain image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

Mixture Density Model

- Model data with ***mixture density***

$$p(x \mid \theta) = \sum_{j=1}^{m} \overbrace{p(x \mid c_j, \theta_j)}^{\text{component densities}} \underbrace{P(c_j)}_{\text{mixing parameters}}$$

  - 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 $p(x|c_j, \theta_j)$



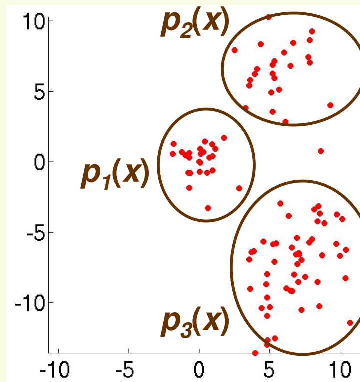
Example: Gaussian Mixture Density

- Mixture of 3 Gaussians

$$p_1(x) \cong N\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)$$

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

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



$$p(x) = 0.2p_1(x) + 0.3p_2(x) + 0.5p_3(x)$$

Mixture Density

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

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

$$P(c_i \mid x, \theta_i) \propto \underbrace{p(x \mid c_i, \theta_i)}_{\text{probability of component } i \text{ to generate } x} \underbrace{P(c_i)}_{\text{probability of component } i}$$

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

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

- need to maximize $l(\theta, \rho)$ with respect to θ and ρ
- As you may have guessed, $l(\theta, \rho)$ is not the easiest function to maximize
  - If we take partial derivatives with respect to θ, ρ and set them to 0, 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***

Mixture Density

- 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

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

- 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 responces, 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***

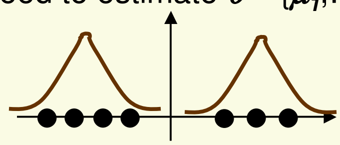
EM: Hidden Variables for Mixture Density

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

- 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 $1 \le i \le n$, $1 \le k \le m$, define hidden variables $z_i^{(k)}$

$$z_i^{(k)} = \begin{cases} 1 & \text{if sample } i \text{ was generated by component } k \\ 0 & \text{otherwise} \end{cases}$$

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

- $z_i^{(k)}$ 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

$$p(x_i \mid z_i, \theta) \sim N(\mu_k, \sigma^2)$$

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

EM: Joint Likelihood

- Let $\boldsymbol{z_i} = \{\boldsymbol{z_i}^{(1)},\dots, \boldsymbol{z_i}^{(m)}\}$, and $\boldsymbol{Z} = \{\boldsymbol{z_1},\dots, \boldsymbol{z_n}\}$
- The complete likelihood is

$$p(X,Z\mid\theta) = p(x_1,\dots,x_n,z_1,\dots,z_n\mid\theta) = \prod_{i=1}^{n} p(x_i,z_i\mid\theta)$$

$$= \prod_{i=1}^{n} \underbrace{p(x_i\mid z_i,\theta)}_{\text{gaussian}}\ \underbrace{p(z_i)}_{\text{part of } \rho_c}$$

- If we actually observed $\boldsymbol{Z}$, the log likelihood $\ln[p(X,Z|\theta)]$ would be trivial to maximize with respect to θ and ρ_i
- The problem, is, of course, is that the values of $\boldsymbol{Z}$ are missing, since we made it up (that is $\boldsymbol{Z}$ is hidden)

EM Derivation

- 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

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

  - If θ makes $\ln[p(X,Z|\theta)]$ large, then θ tends to make $E[\ln p(X,Z|\theta)]$ large
  - the expectation is with respect to the missing data $\boldsymbol{Z}$
  - that is with respect to density $p(Z|X,\theta)$
- however θ is our ultimate goal, we don't know θ!

EM Algorithm

- *EM* solution is to iterate

  1. start with initial parameters $\boldsymbol{\theta}^{(0)}$

  iterate the following 2 step until convergence

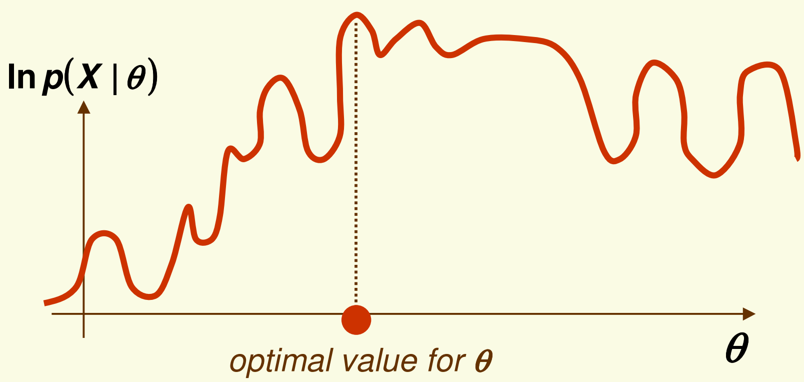
  E. compute the expectation of log likelihood with respect to current estimate $\boldsymbol{\theta}^{(t)}$ and $\boldsymbol{X}$. Let's call it $\boldsymbol{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$

$$Q\left(\theta \mid \theta^{(t)}\right)=E_Z\left[\ln p(X,Z \mid \theta) \mid X,\theta^{(t)}\right]$$

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

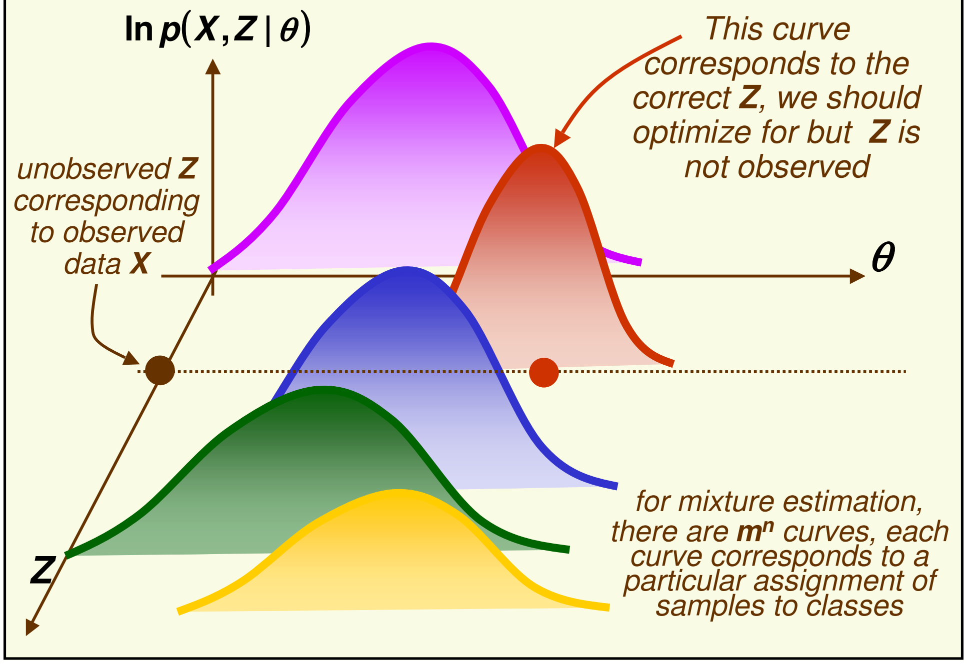
$$\theta^{(t+1)}=\underset{\theta}{\operatorname{argmax}}\, Q\left(\theta \mid \theta^{(t)}\right)$$

EM Algorithm: Picture

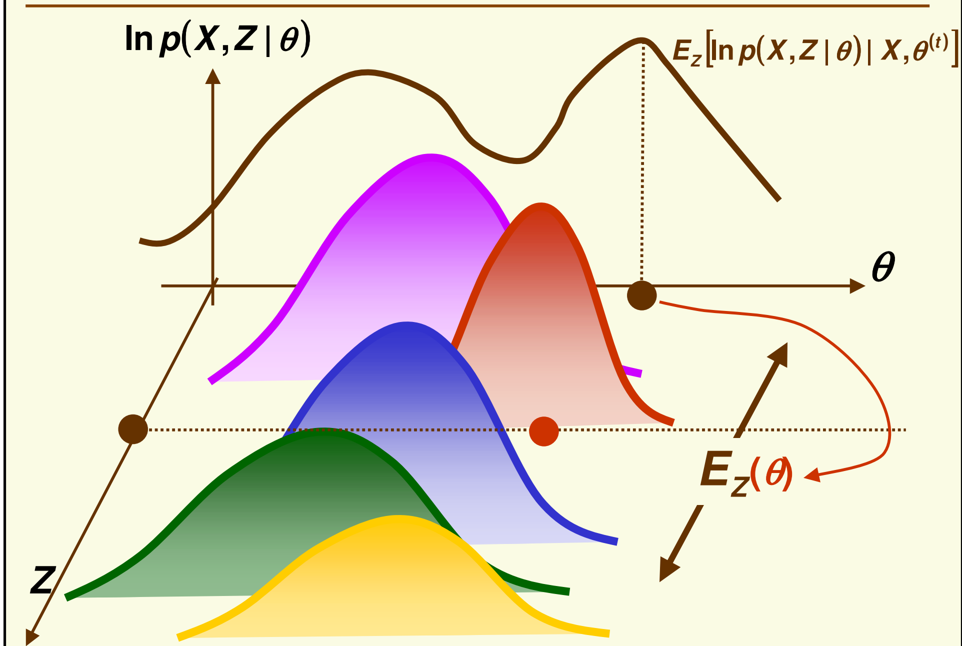


optimal value for $\boldsymbol{\theta}$
we'd like to find it but optimizing $\boldsymbol{p}(\boldsymbol{X}\,|\boldsymbol{\theta})$ *is very difficult*

EM Algorithm: Picture



EM Algorithm: Picture



EM Algorithm

- It can be proven that EM algorithm converges to the local maximum of the log-likelihood

$$\ln p(X \mid \theta)$$

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

$$p(x \mid c_j, \theta_j) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu_j)^2}{2\sigma^2}\right)$$

  - need to estimate $\theta = \{\mu_1, \ldots, \mu_m\}$ and $\rho_1, \ldots, \rho_m$
- for $1 \le i \le n$, $1 \le k \le m$, 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

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

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 $p(x_i\mid z_i,\theta)P(z_i)$

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

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

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

EM for Mixture of Gaussians: E step

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

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

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

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

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

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

EM for Mixture of Gaussians: E step

- log-likelihood of observed X and hidden Z is

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

- For the E step, we must compute

$$Q(\theta\mid\theta^{(t)})=Q(\theta\mid\mu_1^{(t)},\dots,\mu_m^{(t)},\rho_1^{(t)},\dots,\rho_m^{(t)})=E_Z\left[\ln p(X,Z\mid\theta)\mid X,\theta^{(t)}\right]$$

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

$$\Big\Downarrow \quad E_X\left[\sum_i a_i x_i+b\right]=\sum_i a_i E_X[x_i]+b$$

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

EM for Mixture of Gaussians: E step

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

- need to compute $E_Z[z_i^{(k)}]$ in the above expression

$$E_Z\left[z_i^{(k)}\right]=0*P\left(z_i^{(k)}=0\mid\theta^{(t)},x_i\right)+1*P\left(z_i^{(k)}=1\mid\theta^{(t)},x_i\right)$$

$$=P\left(z_i^{(k)}=1\mid\theta^{(t)},x_i\right)=\frac{p\left(x_i\mid\theta^{(t)},z_i^{(k)}=1\right)P\left(z_i^{(k)}=1\mid\theta^{(t)}\right)}{p\left(x_i\mid\theta^{(t)}\right)}$$

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

- 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

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

- Need to maximize Q with respect to all parameters
- First differentiate with respect to μ_k

$$\frac{\partial}{\partial\mu_k} Q(\theta \mid \theta^{(t)}) = \sum_{i=1}^{n} E_Z\left[z_i^{(k)}\right]\frac{(x_i-\mu_k)}{\sigma^2} = 0$$

$$\Rightarrow \textbf{new } \mu_k = \mu_k^{(t+1)} = \boxed{\frac{1}{n}\sum_{i=1}^{n} E_Z\left[z_i^{(k)}\right] x_i}$$

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

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

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

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

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

$$\boxed{\rho_k^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n} E_Z\left[z_i^{(k)}\right]}$$

EM Algorithm

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 $\Sigma\rho_i = 1$)

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

E. for all ***i***, ***k***, compute

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

M. for all ***k***, do parameter update

$$\mu_k = \frac{1}{n}\sum_{i=1}^{n} E_Z\left[z_i^{(k)}\right] x_i \qquad \rho_k = \frac{1}{n}\sum_{i=1}^{n} E_Z\left[z_i^{(k)}\right]$$

EM Algorithm

- For the more general case of multivariate Gaussians with unknown means and variances
  - ***E*** step: $E_Z\left[z_i^{(k)}\right] = \dfrac{\rho_k\, p(x \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{m}\rho_j\, p(x \mid \mu_j, \Sigma_j)}$

    where $p(x \mid \mu_k, \Sigma_k) = \dfrac{1}{(2\pi)^{d/2}\left|\Sigma_k^{-1}\right|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu_k)^t \Sigma_k^{-1}(x-\mu_k)\right]$
  - ***M*** step:

$$\rho_k = \frac{1}{n}\sum_{i=1}^{n} E_Z\left[z_i^{(k)}\right]$$

$$\Sigma_k = \frac{\sum_{i=1}^{n} E_Z\left[z_i^{(k)}\right](x_i-\mu_k)(x_i-\mu_k)^T}{\sum_{i=1}^{n} E_Z\left[z_i^{(k)}\right]}$$

$$\mu_k = \frac{\sum_{i=1}^{n} E_Z\left[z_i^{(k)}\right] x_i}{\sum_{i=1}^{n} E_Z\left[z_i^{(k)}\right]}$$

EM Algorithm and K-means

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

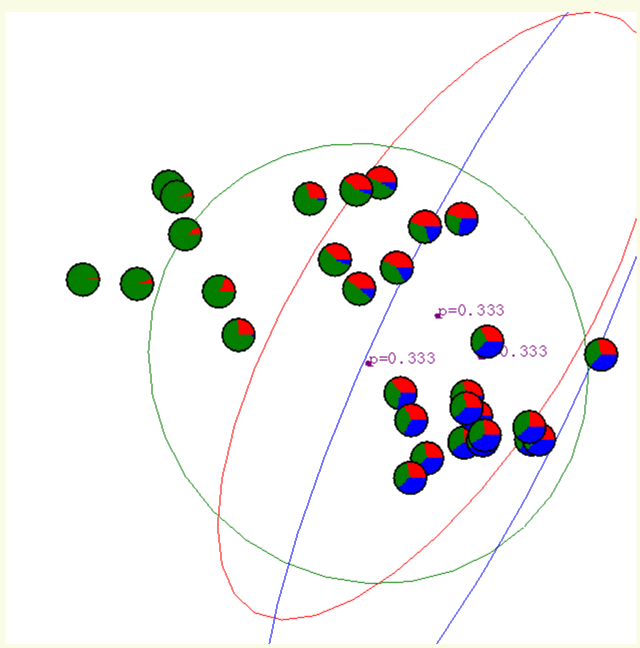
$$E_Z\left[z_i^{(k)}\right] = \frac{\rho_k \exp\left(-\frac{1}{2\sigma^2}(x_i-\mu_k)^2\right)}{\sum_{j=1}^{m}\rho_j \exp\left(-\frac{1}{2\sigma^2}(x_i-\mu_j)^2\right)} = \frac{\exp\left(-\frac{1}{2\sigma^2}(x_i-\mu_k)^2\right)}{\sum_{j=1}^{m}\exp\left(-\frac{1}{2\sigma^2}(x_i-\mu_j)^2\right)}$$

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

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

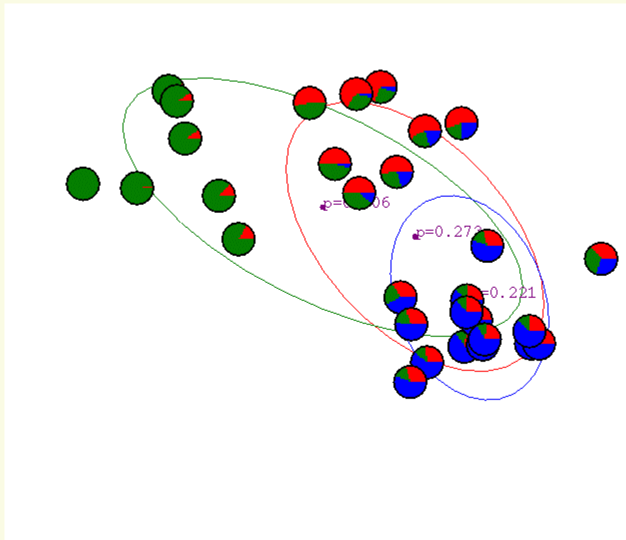
  - so at the *E* step, for each current mean, we find all points closest to it and form new clusters
  - at the *M* step, we compute the new means inside current clusters $\mu_k = \frac{1}{n}\sum_{i=1}^{n} E_Z\left[z_i^{(k)}\right] x_i$

EM Gaussian Mixture Example



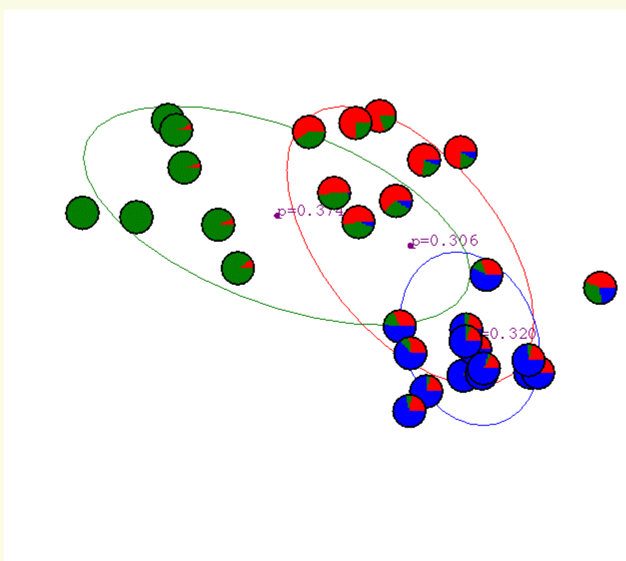
EM Gaussian Mixture Example

After first iteration



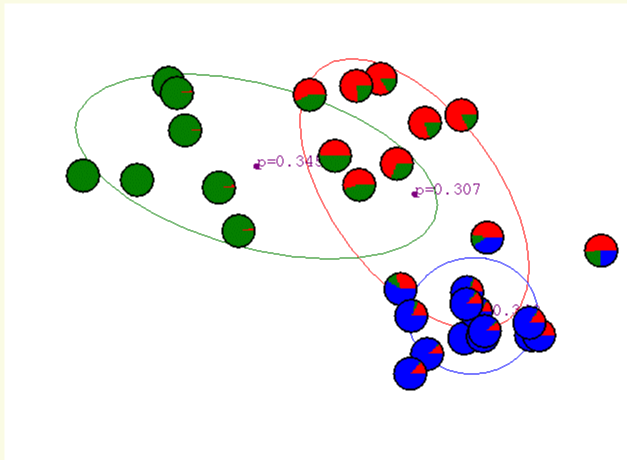
EM Gaussian Mixture Example

After second iteration



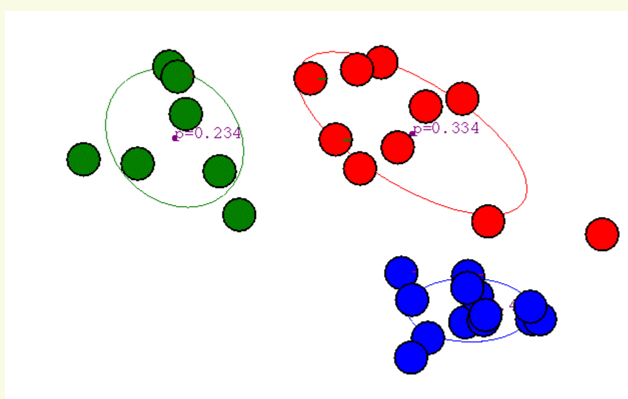
EM Gaussian Mixture Example

After third iteration



EM Gaussian Mixture Example

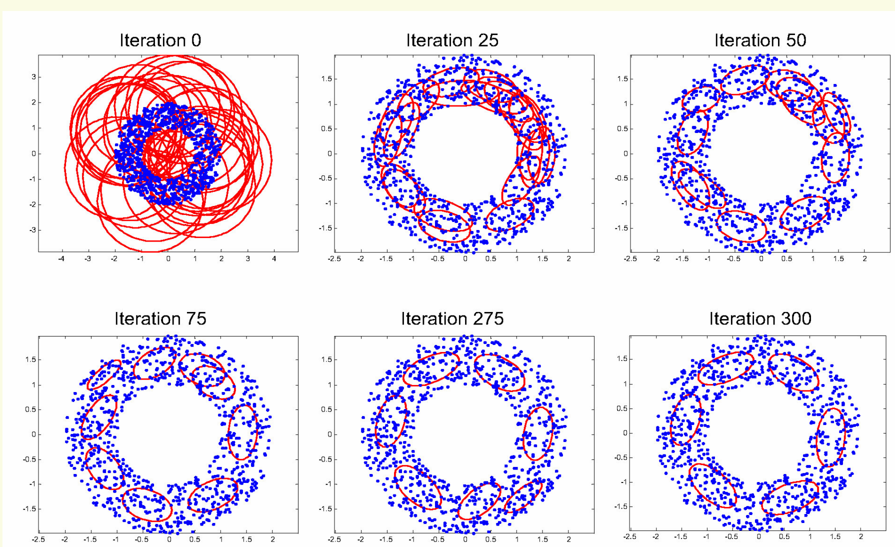
After 20th iteration



EM Example

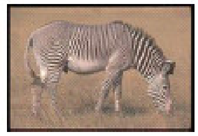
- Example from R. Gutierrez-Osuna
- Training set of 900 examples forming an annulus
- Mixture model with ***m*** = 30 Gaussian components of unknown mean and variance is used
- Training:
  - Initialization:
    - means to 30 random examples
    - covaraince matrices initialized to be diagonal, with large 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



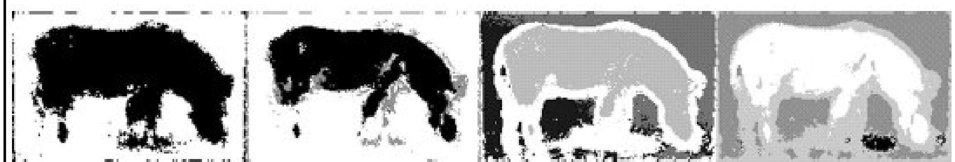


Figure from "Color and Texture Based Image Segmentation Using EM and Its Application to Content Based Image Retrieval",S.J. Belongie et al., ICCV 1998

EM Motion Segmentation Example

Three frames from the MPEG "flower garden" sequence

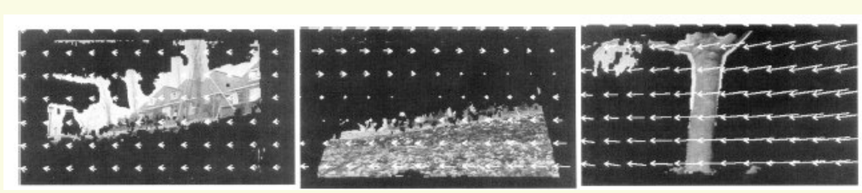


Figure from "Representing Images with layers,", by J. Wang and E.H. Adelson, IEEE Transactions on Image Processing, 1994, c 1994, IEEE

EM Algorithm 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)