

# Patten Recognition

## Lecture 10

Lecturer: Xinchao Wang

xinchao@nus.edu.sg



# Last week...

- Bagging
  - Random Forest
- Boosting

# Overview

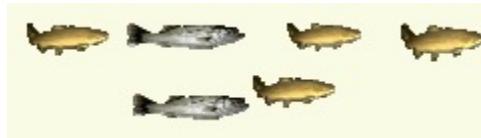
- Unsupervised Learning
  - Supervised vs. unsupervised learning
  - Unsupervised learning
  - Flat clustering (k-means)
  - Hierarchical clustering
- Expectation Maximization
  - Gaussian Mixture Model
  - Optimization questions (How to derive the final parameters) won't be in exams!

# Supervised vs. Unsupervised Learning

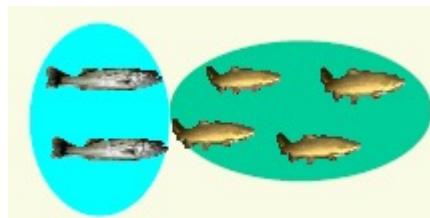
- Up to now we considered **supervised learning** scenarios, where we are given:
  1. samples  $x_1, \dots, x_n$
  2. class labels for all samples
    - This is also called learning with teacher, since the correct answer (the true class) is provided
- Here, we consider **unsupervised learning** scenarios, where we are only given:
  1. Only samples  $x_1, \dots, x_n$ 
    - This is also called learning without teacher, since the correct answer is not provided
    - Do not split data into training and test sets

# Unsupervised Learning

- Data are *not labeled*



- Parametric Approach
  - Assume parametric **distribution** of data
  - Estimate parameters of this distribution
    - Expectation Maximization
- Non-Parametric Approach
  - Group the data into **clusters**, each cluster (hopefully) says something about classes present in the data

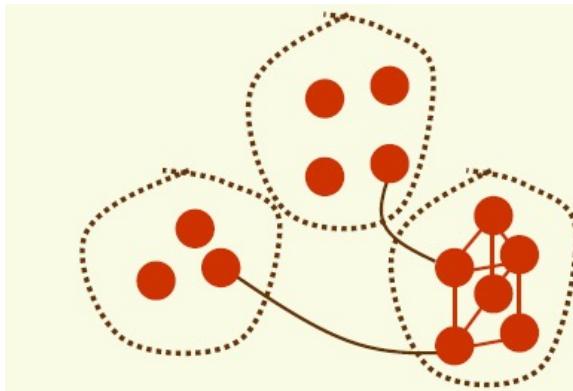


# Why Unsupervised Learning?

- Unsupervised learning is harder
  - How do we know if results are meaningful? No answer (labels) is available
    - Let the experts 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, object detection in images)
    - 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

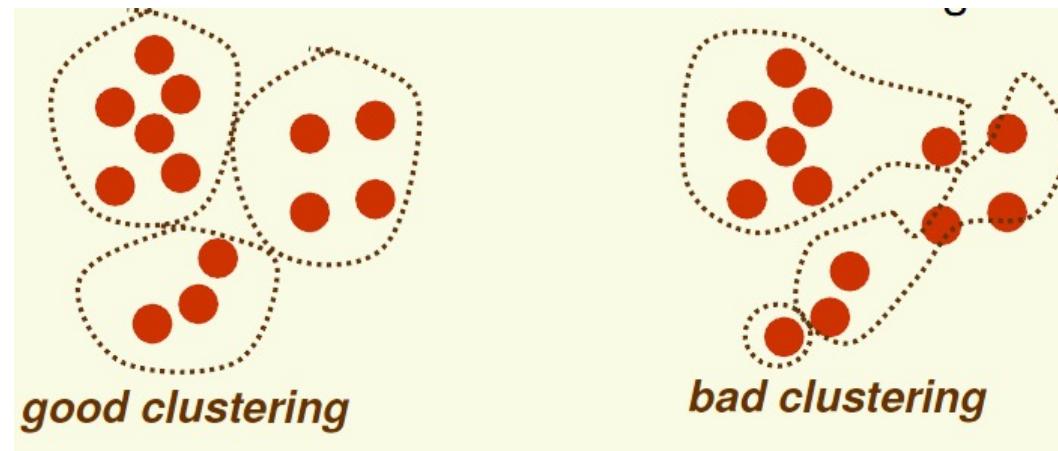
- Seek “natural” clusters in the data



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

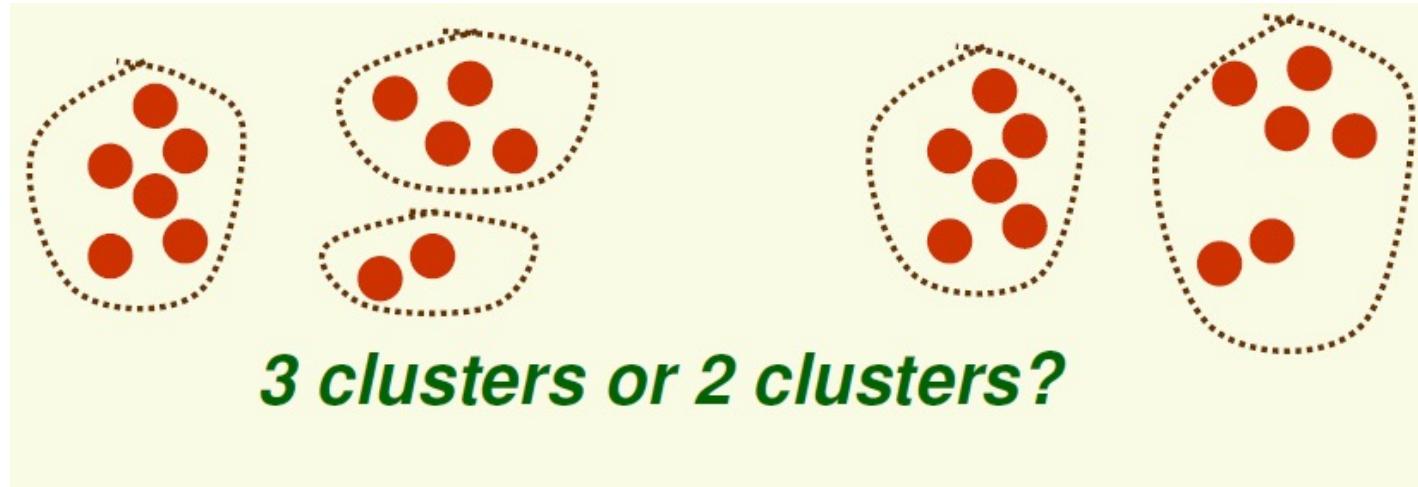
# What we need for Clustering

1. Proximity measure, either
  - similarity measure  $s(x_i, x_k)$ : large if  $x_i, x_k$  are similar
  - dissimilarity(or distance) measure  $d(x_i, x_k)$ : small if  $x_i, x_k$  are similar
2. Criterion function to evaluate a clustering



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

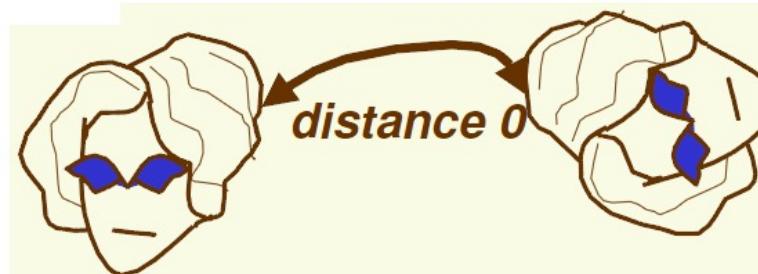
# Clustering Issue: How Many Clusters?



- 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)

# Clustering Issue: Proximity Measures

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



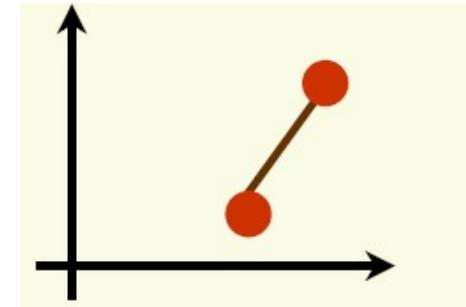
- For character recognition, invariance to rotation is bad



# Clustering Issue: Distance Measures

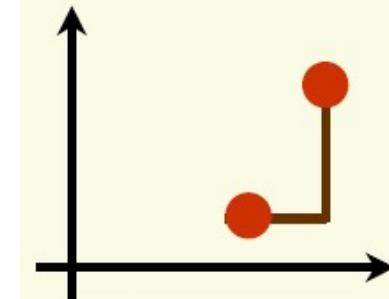
- Euclidean distance

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^d (\mathbf{x}_i^{(k)} - \mathbf{x}_j^{(k)})^2}$$



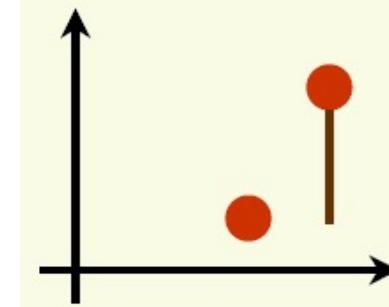
- Manhattan (city block) distance

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^d |\mathbf{x}_i^{(k)} - \mathbf{x}_j^{(k)}|$$



- Chebyshev distance

$$d(\mathbf{x}_i, \mathbf{x}_j) = \max_{1 \leq k \leq d} |\mathbf{x}_i^{(k)} - \mathbf{x}_j^{(k)}|$$



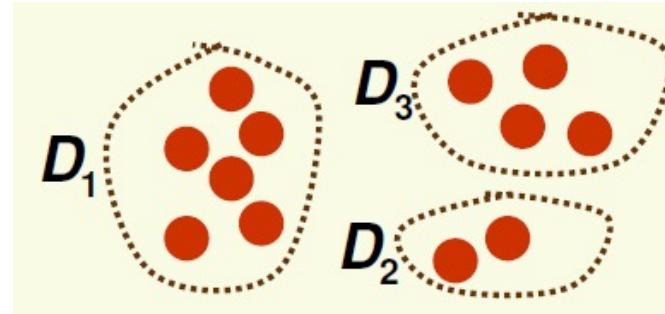
# Clustering Issue: Feature Scaling

- Old problem: how to choose appropriate relative scale for features?
  - [length (in meters or cms?), weight (in grams or kgs?)]
- In supervised learning, we 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***



# Criterion Functions for Clustering

- Given samples  $\mathbf{x}_1, \dots, \mathbf{x}_n$
- Partition them into  $c$  subsets  $D_1, \dots, D_c$



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

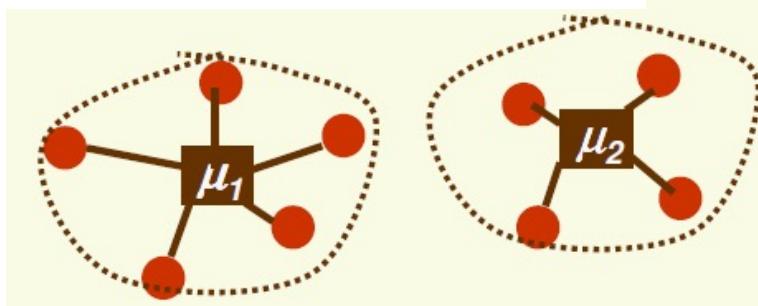
# Sum of Squared Error (SSE) Criterion Function

- Let  $n_i$  be the number of samples in  $D_i$ , and define the mean of samples in  $D_i$

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

- Then the sum-of-squared errors criterion function (to minimize) is:

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

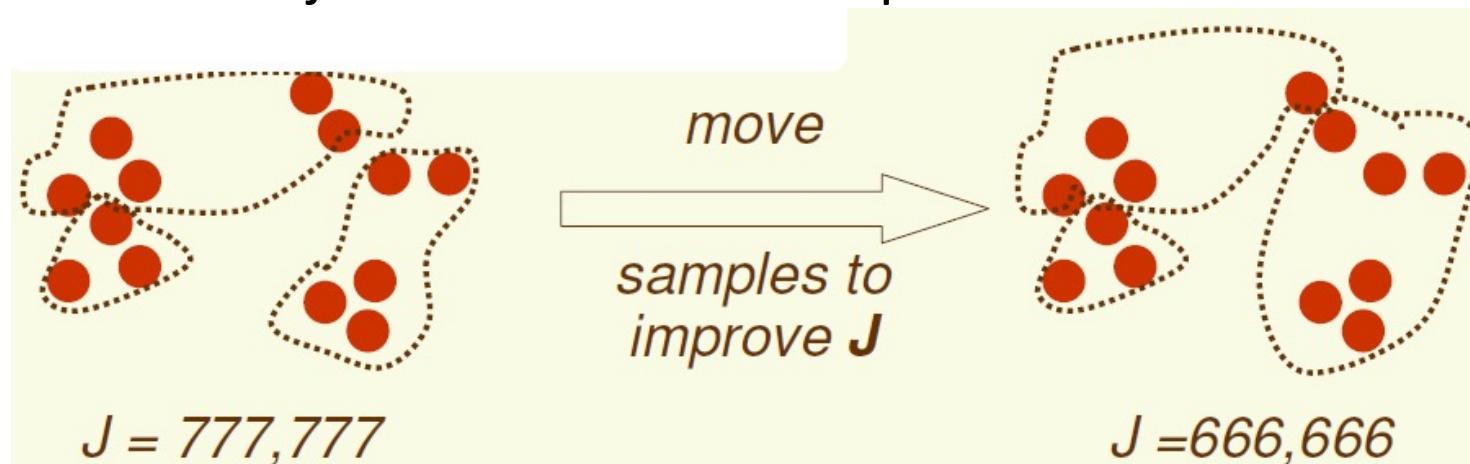


- Note that the number of clusters,  $c$ , is fixed

# K-means Clustering

# Iterative Optimization Algorithms

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



# Iterative Optimization Algorithms

- Iterative optimization algorithms are similar to gradient descent
  - Move in a direction of descent, but not in the steepest descent direction since they have no derivative of the objective function
  - Solution depends on the initial point
  - Cannot find global minimum
- Main Issue
  - How to move from current partitioning to the one which improves the objective function

# K-means Clustering

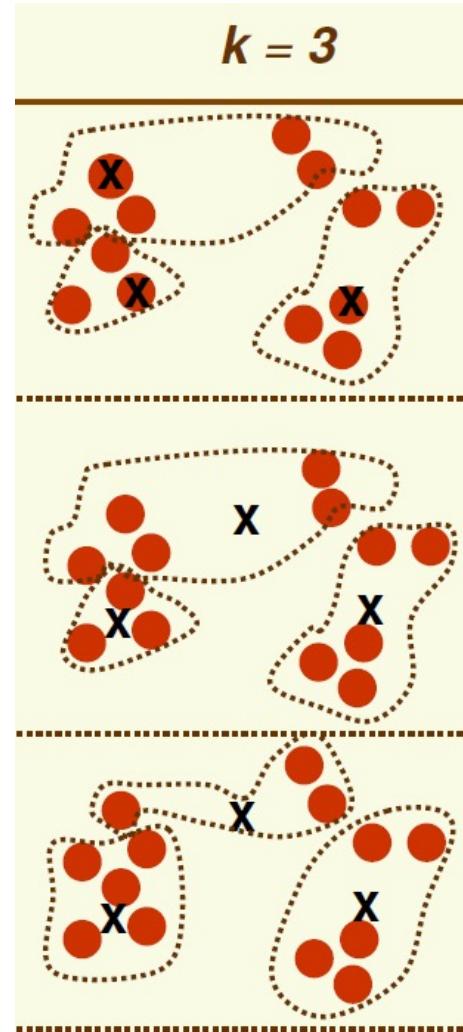
- We now consider an example of iterative optimization algorithm for the special case of  $J_{SSE}$  objective function

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

- K-means is probably the most famous clustering algorithm
  - It has a smart way of moving from current partitioning to the next one
- Fix number of clusters to  $k$  ( $c = k$ )

# K-means Clustering

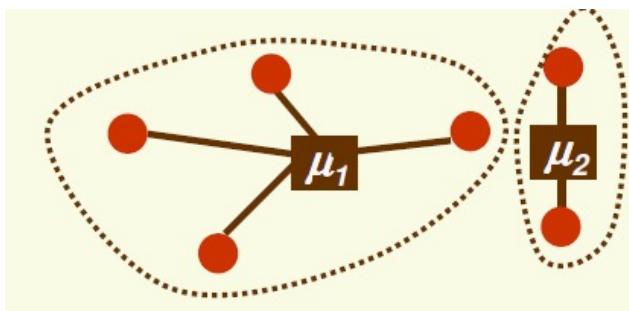
1. Initialize
  - Pick  $k$  cluster centers arbitrarily
  - Assign each example to closest center
2. Compute sample means for each cluster
3. Reassign all samples to the closest mean
4. If clusters changed at step 3, go to step 2



# K-means Clustering

Consider steps 2 and 3 of the algorithm

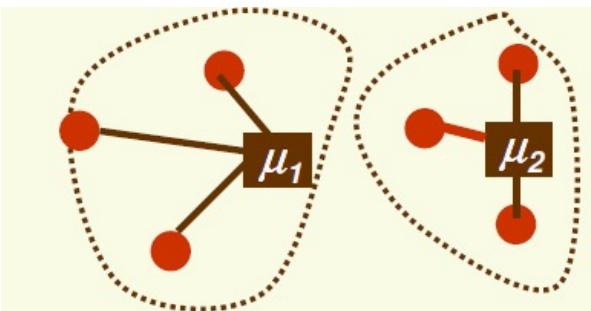
2. compute sample means for each cluster



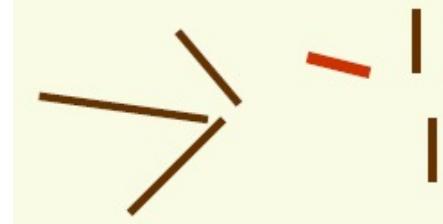
$$J_{SSE} = \sum_{i=1}^k \sum_{x \in D_i} \|x - \mu_i\|^2$$

= sum of

3. reassign all samples to the closest mean

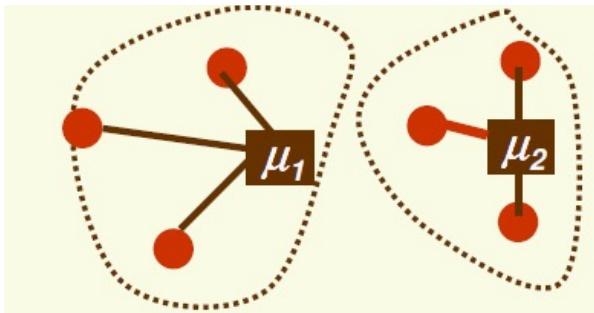


If we represent clusters by their old means, the error has decreased

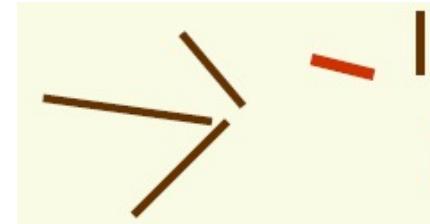


# K-means Clustering

## 3. reassign all samples to the closest mean



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



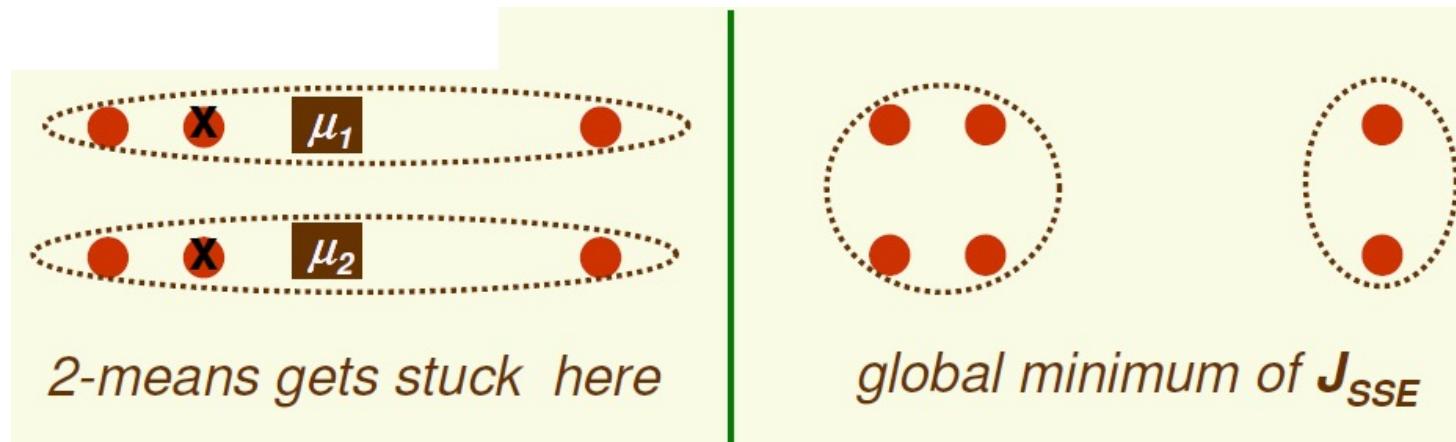
- However we represent clusters by their new means, and the mean always results in the smallest sum of squared distances

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

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

# K-means Clustering

- Proved that by repeating steps 2 and 3, the objective function is reduced
  - Found a “smart” move which decreases the objective function
- Thus k-means converges after a finite number of iterations of steps 2 and 3
- However k-means is not guaranteed to find a global minimum



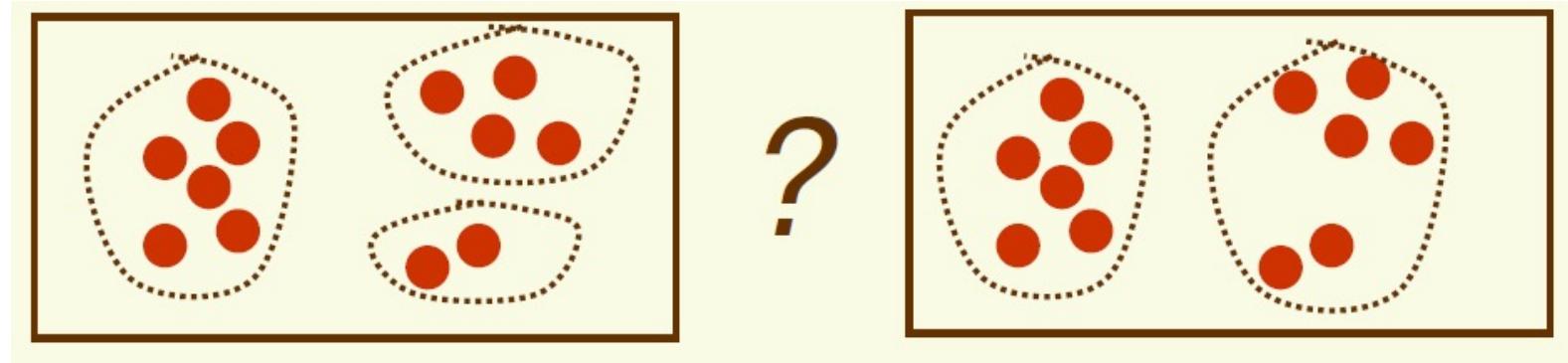
# K-means Clustering

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

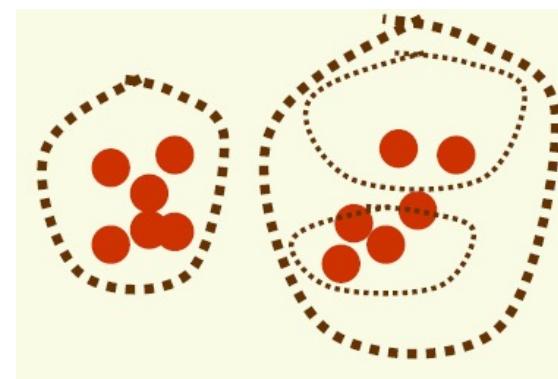
# Hierarchical Clustering

# Hierarchical Clustering

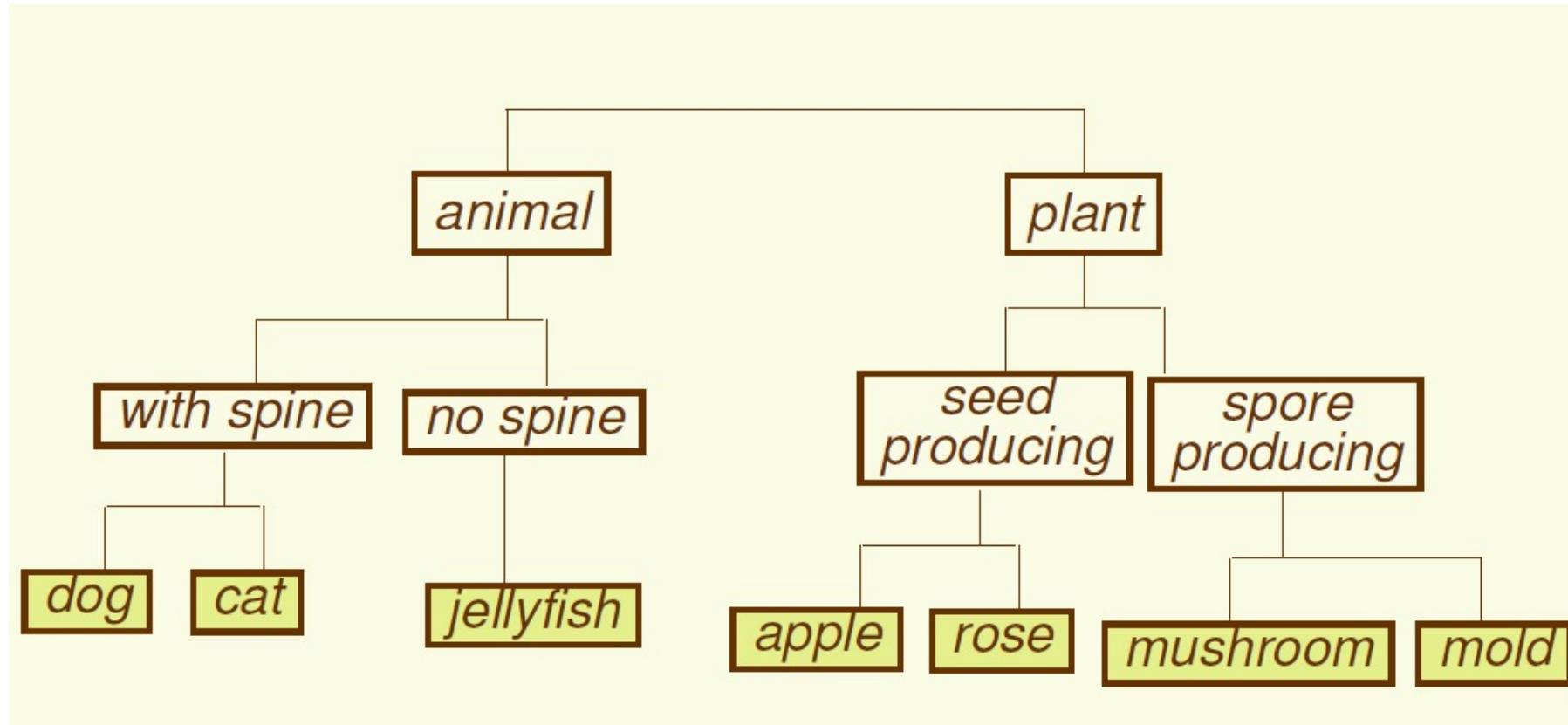
- Up to now considered flat clustering



- For some data, hierarchical clustering is more appropriate than “flat” clustering
- Hierarchical clustering

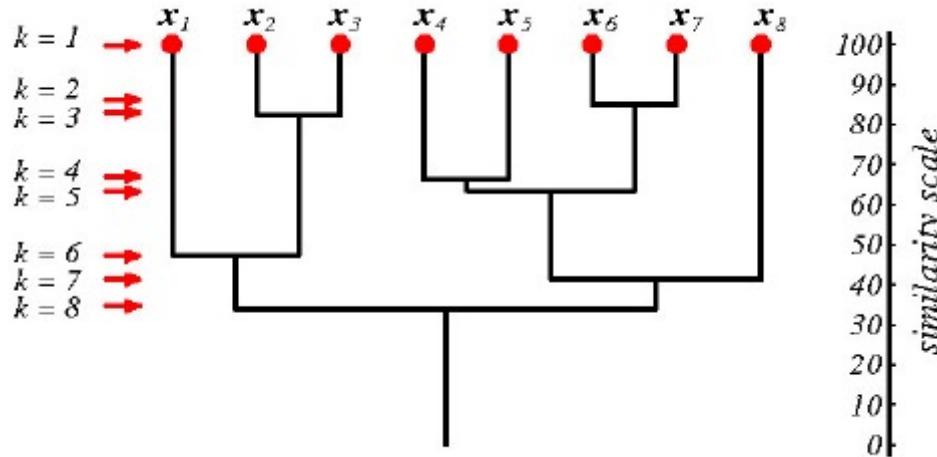


# Example of Hierarchical Clustering



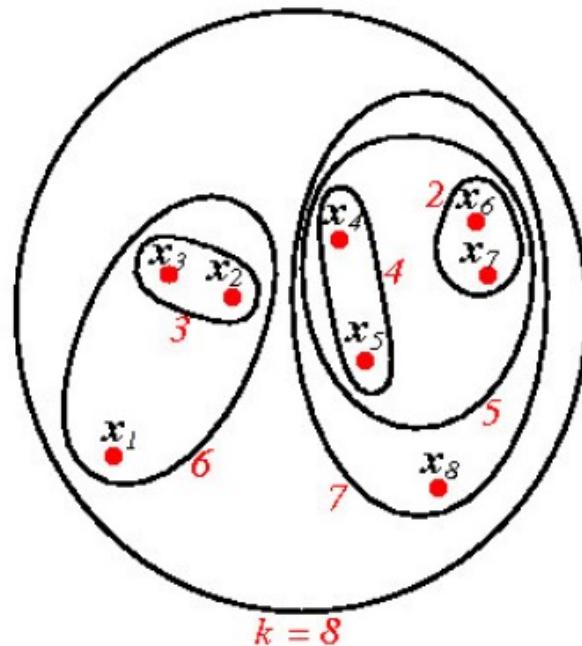
# Hierarchical Clustering: Dendrogram

- The preferred way to represent a hierarchical clustering is a dendrogram
  - Binary tree
  - Level  $k$  corresponds to partitioning with  $n-k+1$  clusters
  - If  $k$  clusters required, use clustering from level  $n-k+1$
  - If samples are in the same cluster at level  $k$ , they stay in the same cluster at higher levels
  - The dendrogram typically shows the similarity of grouped clusters



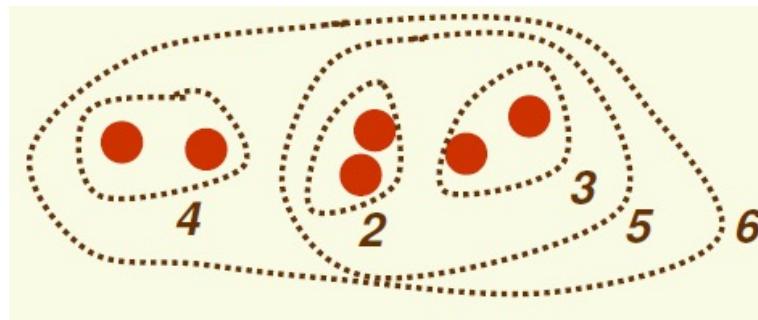
# Hierarchical Clustering: Venn Diagram

- Can also use Venn diagram to show hierarchical clustering, but similarity is not represented quantitatively



# Hierarchical Clustering

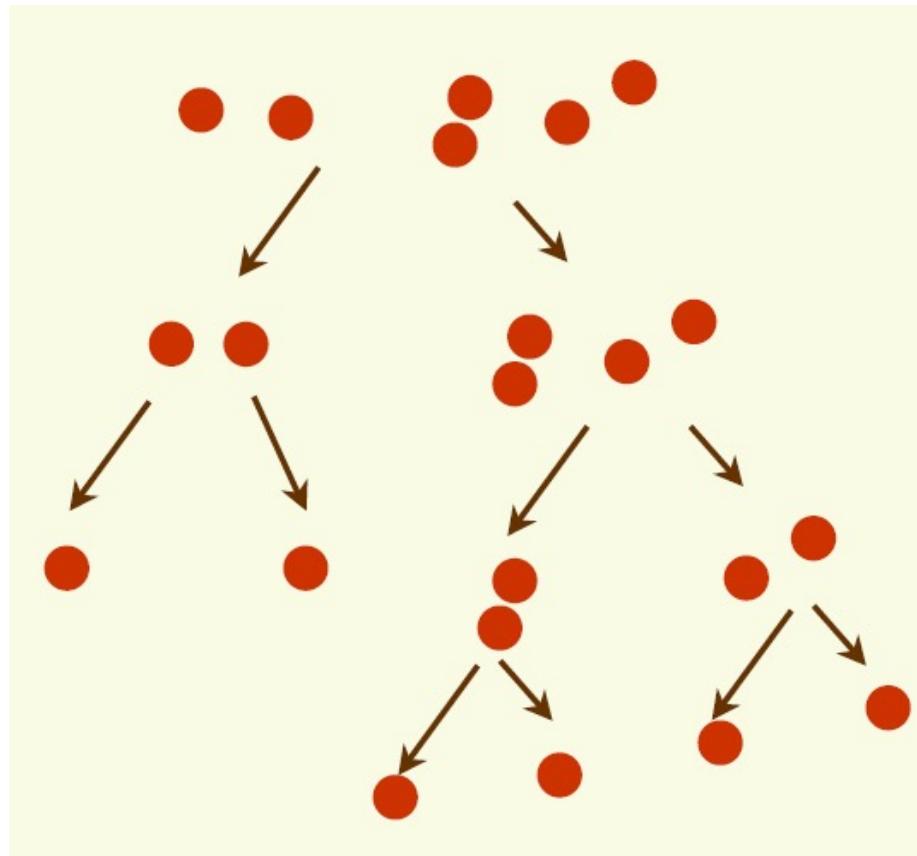
- Algorithms for hierarchical clustering can be divided into two types:
  1. Agglomerative (bottom up) procedures
    - Start with  $n$  singleton clusters
    - Form hierarchy by merging most similar clusters



2. Divisive (top down) procedures
  - Start with all samples in one cluster
  - Form hierarchy by splitting the “worst” clusters

# Divisive Hierarchical Clustering

- Any “flat” algorithm which produces a fixed number of clusters can be used
  - Set  $c = 2$



# Agglomerative Hierarchical Clustering

initialize with each example in singleton cluster

**while** there is more than 1 cluster

1. find 2 nearest clusters
2. merge them

- Four common ways to measure cluster distance

1. minimum distance

$$d_{\min}(D_i, D_j) = \min_{x \in D_i, y \in D_j} \|x - y\|$$

2. maximum distance

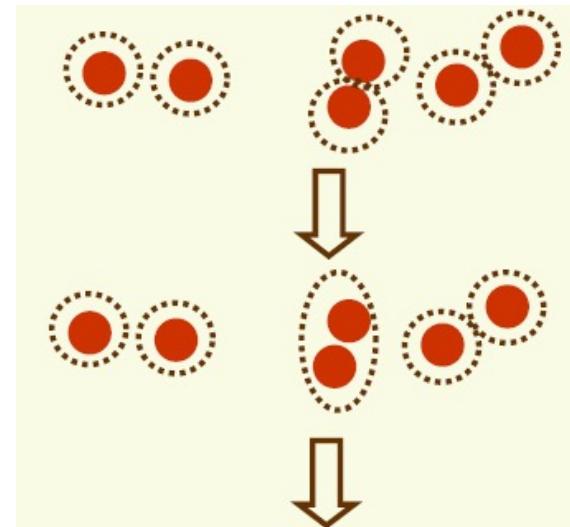
$$d_{\max}(D_i, D_j) = \max_{x \in D_i, y \in D_j} \|x - y\|$$

3. average distance

$$d_{avg}(D_i, D_j) = \frac{1}{n_i n_j} \sum_{x \in D_i} \sum_{y \in D_j} \|x - y\|$$

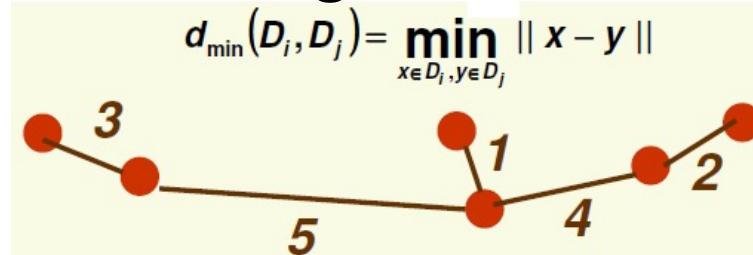
4. mean distance

$$d_{mean}(D_i, D_j) = \|\mu_i - \mu_j\|$$

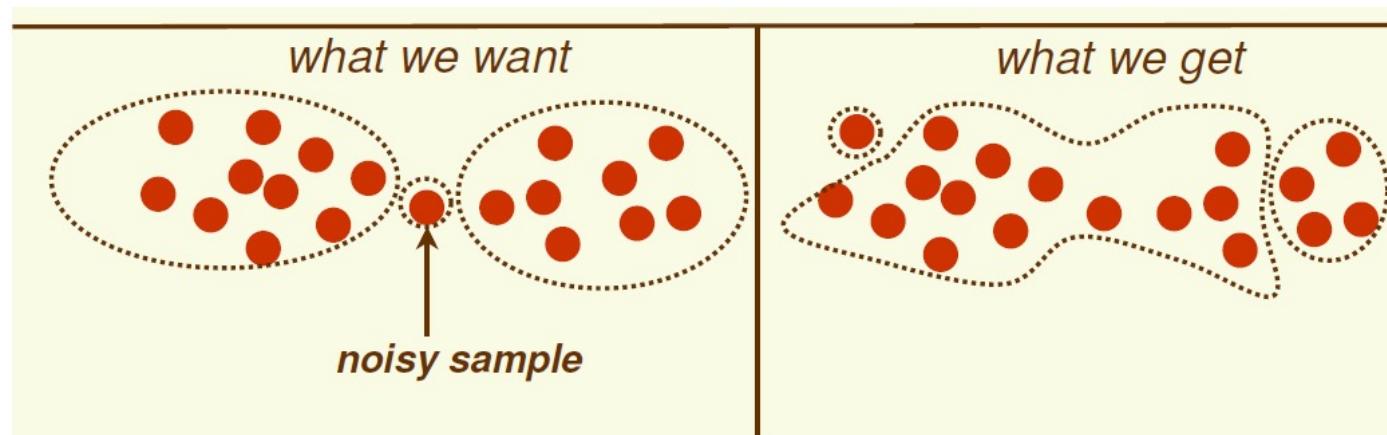


# Single Linkage or Nearest Neighbor

- Agglomerative clustering with minimum distance



- Generates minimum spanning tree
- Encourages growth of elongated clusters
- Disadvantage: very sensitive to noise

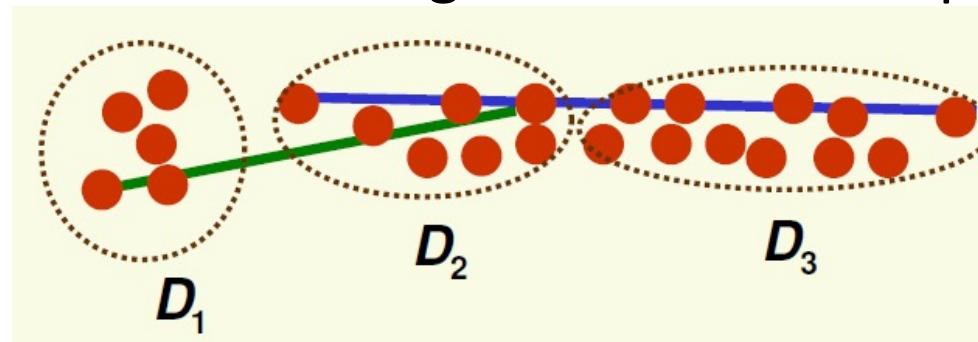


# Complete Linkage or Farthest Neighbor

- Agglomerative clustering with maximum distance

$$d_{\max}(D_i, D_j) = \max_{x \in D_i, y \in D_j} \|x - y\|$$

- Encourages compact clusters
- Does not work well if elongated clusters are present



- $d_{\max}(D_1, D_2) < d_{\max}(D_2, D_3)$
- thus  $D_1$  and  $D_2$  are merged instead of  $D_2$  and  $D_3$

# Average and Mean Agglomerative Clustering

- Agglomerative clustering is more robust under the average or the mean cluster distance

$$d_{avg}(D_i, D_j) = \frac{1}{n_i n_j} \sum_{x \in D_i} \sum_{y \in D_j} \|x - y\|$$

$$d_{mean}(D_i, D_j) = \|\mu_i - \mu_j\|$$

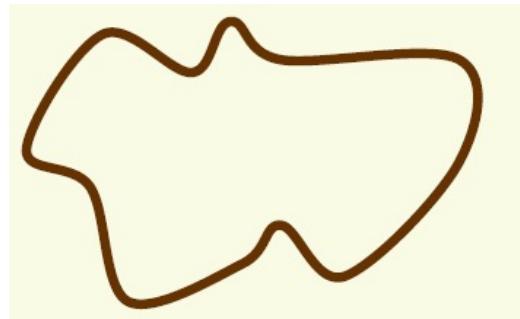
- Mean distance is cheaper to compute than the average distance
- Unfortunately, there is not much to say about agglomerative clustering theoretically, but it does work reasonably well in practice

# Agglomerative vs. Divisive

- Agglomerative is faster to compute, in general
- Divisive may be less “blind” to the global structure of the data

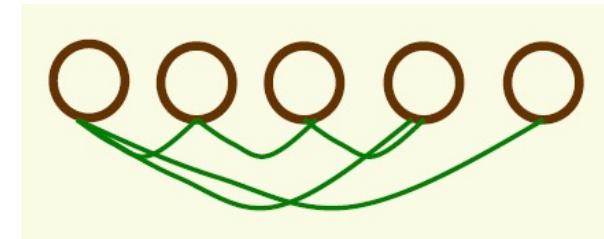
## ***Divisive***

- when taking the first step (split), it has access to all the data; can find the best possible split in 2 parts



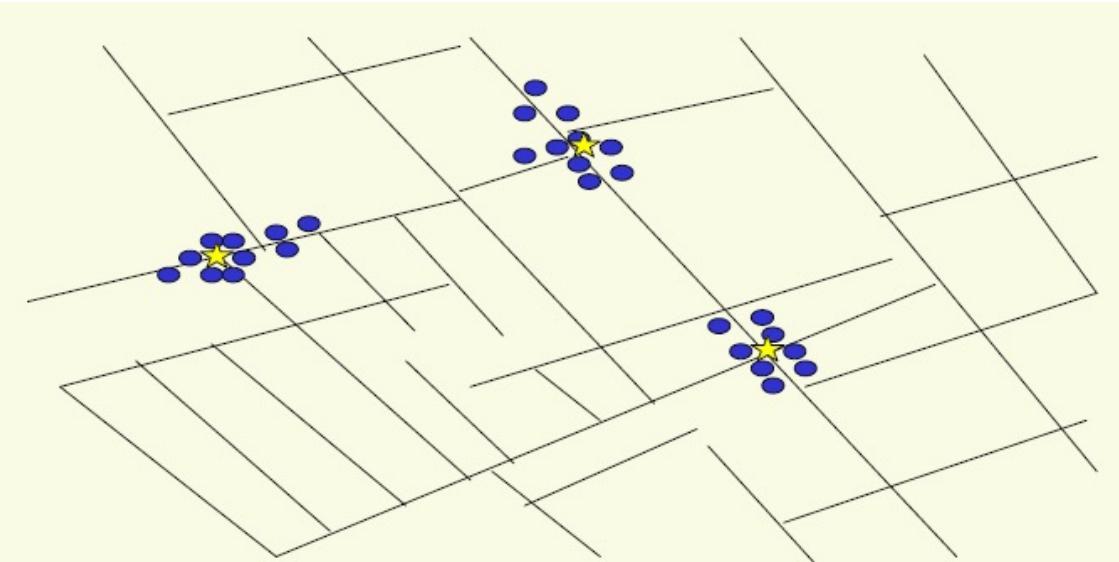
## ***Agglomerative***

- when taking the first step (merge), it does not consider the global structure of the data, only looks at pairwise structure



# First (?) Application of Clustering

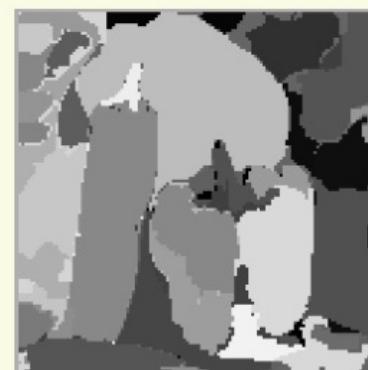
- John Snow, a London physician plotted the location of cholera deaths on a map during an outbreak in the 1850s.
- The locations indicated that cases were clustered around certain intersections where there were polluted wells -- thus exposing both the problem and the solution.



From: Nina Mishra HP Labs

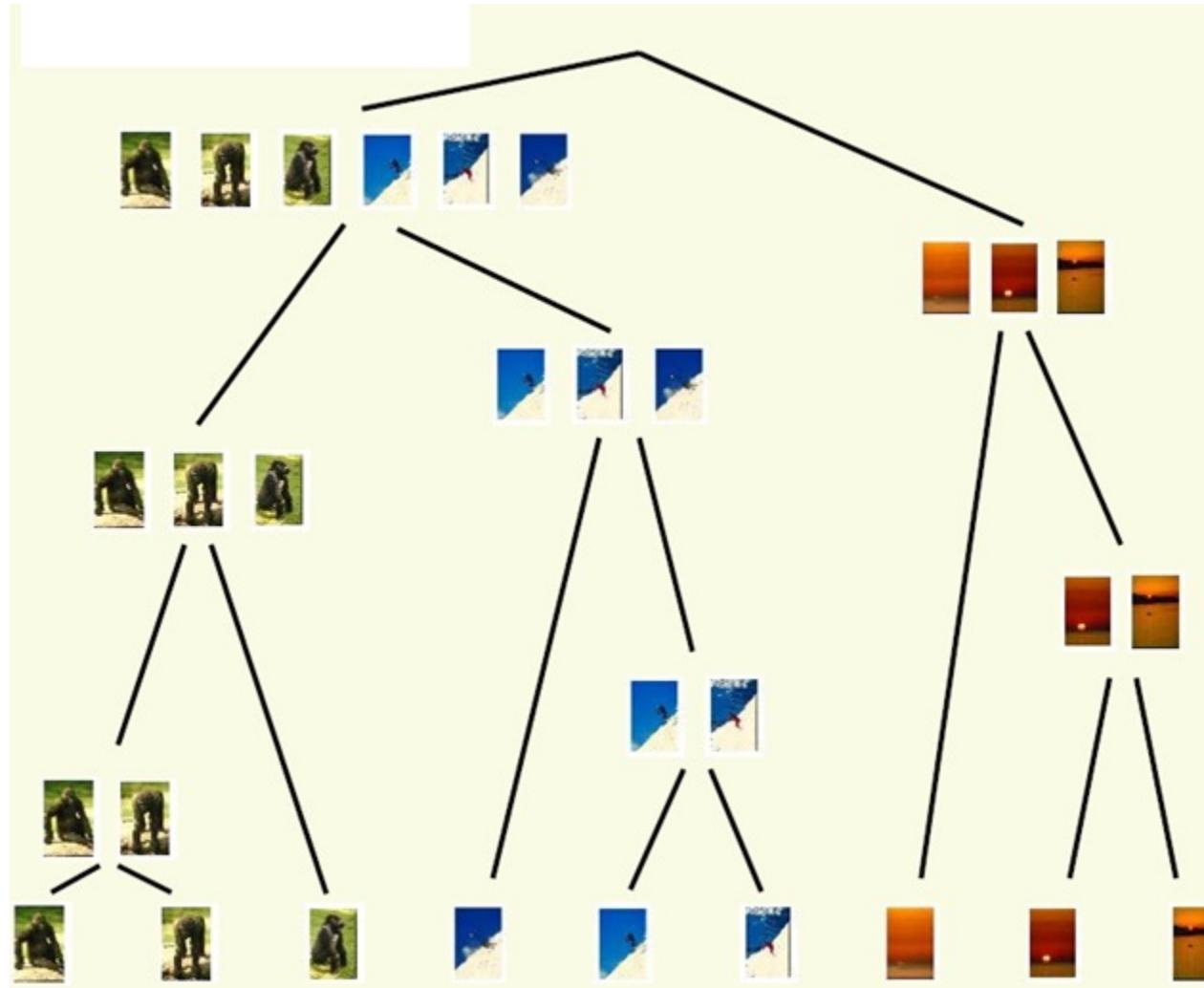
# Applications of Clustering

- Image segmentation
  - Find coherent “objects” in images



From: Image Segmentation by Nested Cuts, O. Veksler, CVPR2000

# Image Database Organization



# Clustering Summary

- Clustering (nonparametric learning) is useful for discovering inherent structure in data
- Clustering is immensely useful in different fields
- Clustering comes naturally to humans (in up to 3 dimensions), but not so to computers
- It is very easy to design a clustering algorithm, but it is very hard to make theoretical claims on performance
- General purpose clustering is unlikely to exist; for best results, clustering should be tuned to application at hand

# EM: Expectation Maximization (GMM: Gaussian Mixture Model)

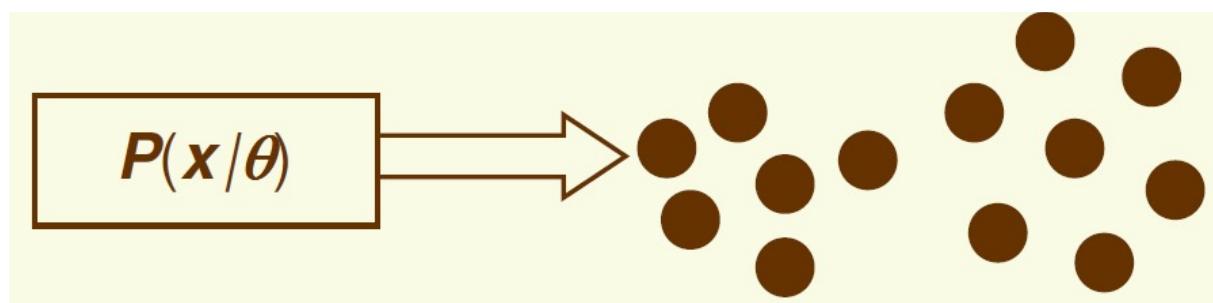
Slides Credits: Olga Veksler

# Unsupervised Learning

- In unsupervised learning, where we are only given samples  $x_1, \dots, x_n$  without class labels
- Nonparametric approach: clustering
- Parametric approach:
  - assume parametric distribution of data
  - estimate parameters of this distribution
  - much “harder” than the supervised learning case

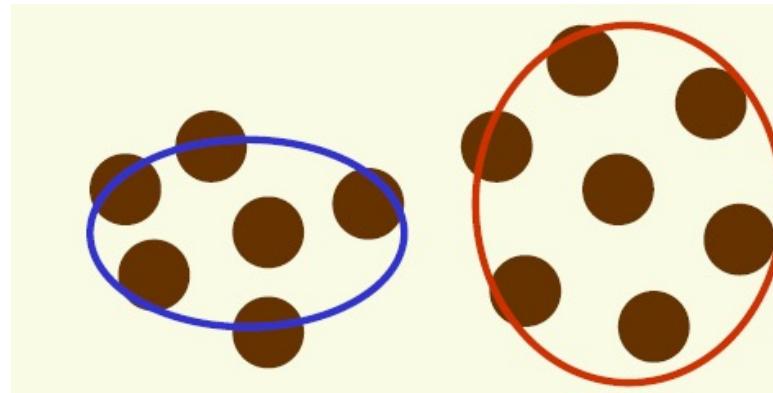
# Parametric Unsupervised Learning

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

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



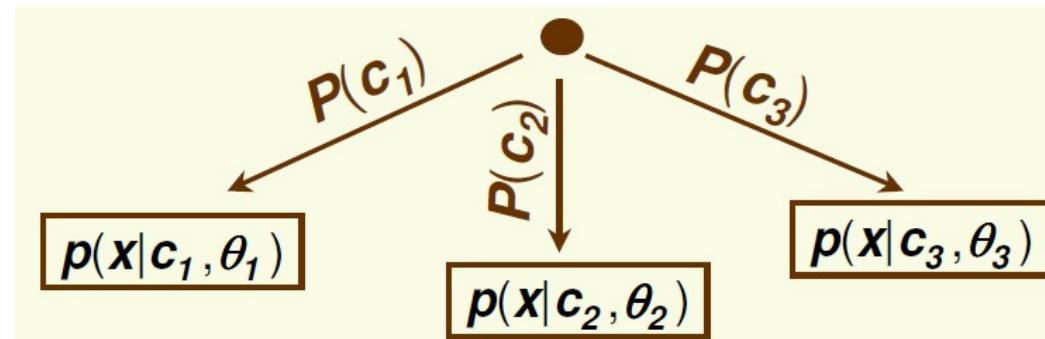
# Mixture Density Model

- Model data with mixture density

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

*component densities*  
*mixing parameters*

- where  $\theta = \{\theta_1, \dots, \theta_m\}$
- $P(c_1) + P(c_2) + \dots + P(c_m) = 1$
- 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  $p(x | c_j, \theta_j)$



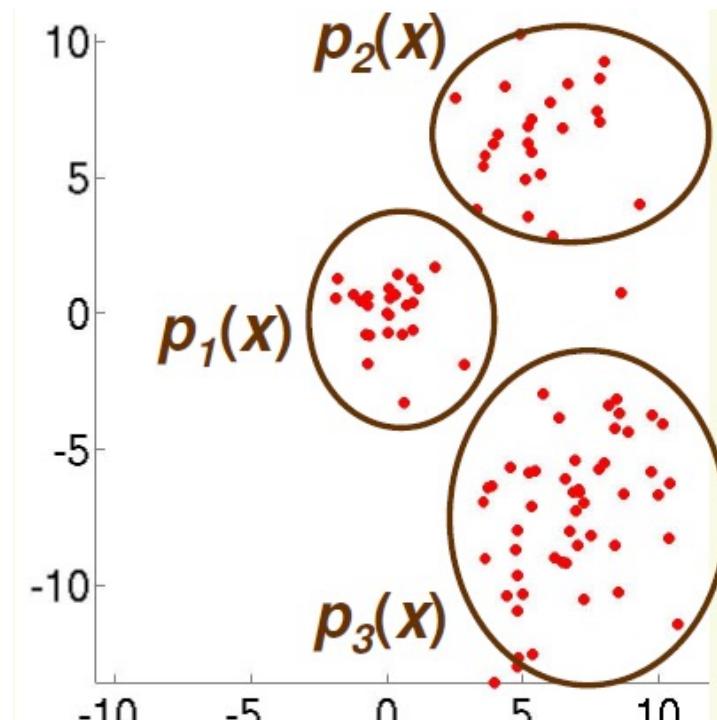
# Example: Gaussian Mixture Density

- Mixture of 3 Gaussians

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

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

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



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

# ML Estimation for Mixture Density

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

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

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

# ML Estimation for Mixture Density

- Need to maximize  $I(\theta, \rho)$  with respect to  $\theta$  and  $\rho$
- $I(\theta, \rho)$  is not the easiest function to maximize
  - If we take partial derivatives with respect to  $\theta$ ,  $\rho$  and set them to 0, typically we have a “coupled” nonlinear system of equations
  - usually closed form solution cannot be found
- There is a better algorithm, called **EM**

# Mixture Density

- Before EM, let's look at the mixture density again

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

- Suppose we know how to estimate  $\theta_1, \dots, \theta_m$  and  $\rho_1, \dots, \rho_m$ 
  - Estimating the class of  $x$  is easy with MAP, maximize:

$$p(x | c_i, \theta_i) P(c_i) = p(x | c_i, \theta_i) \rho_i$$

- Suppose we know the class of samples  $x_1, \dots, x_n$ 
  - This is just the supervised learning case, so estimating  $\theta_1, \dots, \theta_m$  and  $\rho_1, \dots, \rho_m$  is easy

$$\hat{\theta}_i = \underset{\theta_i}{\operatorname{argmax}} [\ln p(D_i | \theta_i)] \quad \hat{\rho}_i = \frac{|D_i|}{n}$$

- This is an example of chicken-and-egg problem
  - The EM algorithm approaches this problem by adding “hidden” variables

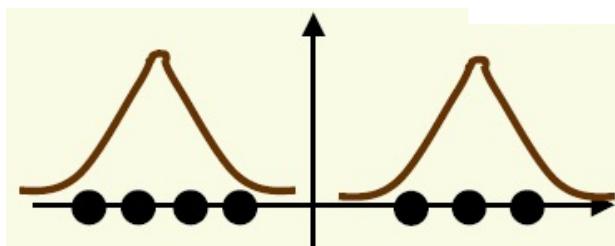
# EM: Hidden Variables for Mixture Density

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

- For simplicity, assume component densities are

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

- assume for now that the variance is known
  - need to estimate  $\theta = \{\mu_1, \dots, \mu_m\}$



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

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

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

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

# EM: Hidden Variables for Mixture Density

- Let  $z_i = \{z_i^{(1)}, \dots, z_i^{(m)}\}$ , be indicator r.v. corresponding to sample  $x_i$
- Conditioned on  $z_i$ , the distribution of  $x_i$  is Gaussian  
where  $k$  is s.t.  $z_i^{(k)} = 1$

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

# EM: Joint Likelihood

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

$$\begin{aligned} p(X, Z | \theta) &= p(x_1, \dots, x_n, z_1, \dots, z_n | \theta) = \prod_{i=1}^n p(x_i, z_i | \theta) \\ &= \prod_{i=1}^n \underbrace{p(x_i | z_i, \theta)}_{gaussian} \underbrace{p(z_i)}_{part\ of\ \rho_c} \end{aligned}$$

- 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, that the values of  $Z$  are missing, since we made it up (that is,  $Z$  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

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

- 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

# The EM Algorithm

- The EM solution is:

- Start with initial parameters  $\theta^{(0)}$
- Iterate the following 2 steps until convergence
  - E. compute the expectation  $Q(\theta \mid \theta^{(t)})$  of the log likelihood with respect to current estimate  $\theta^{(t)}$  and  $X$

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

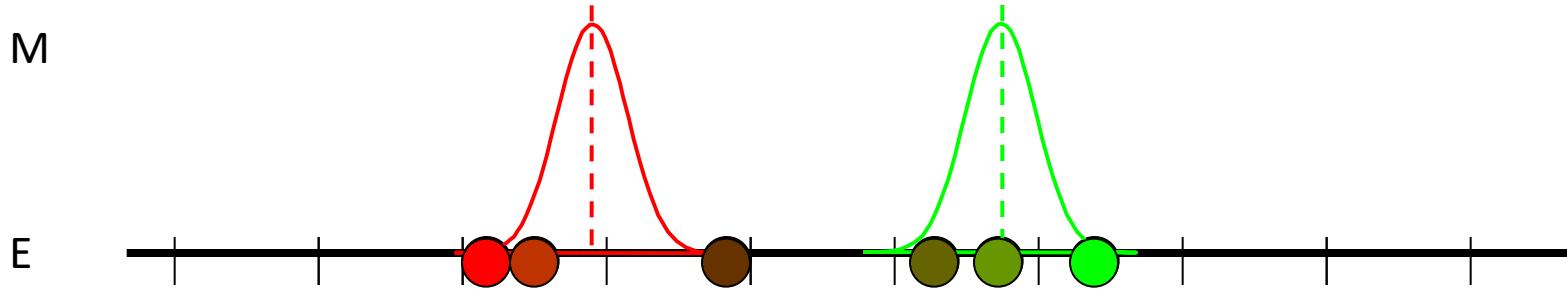
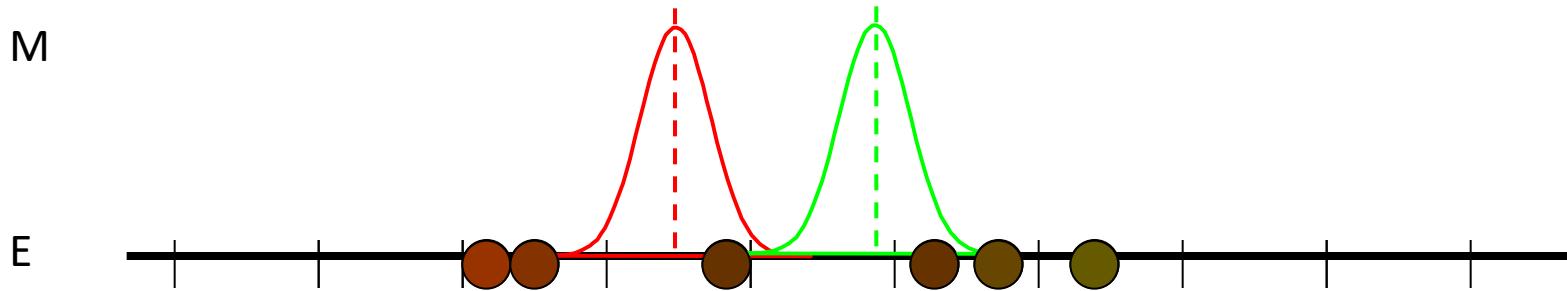
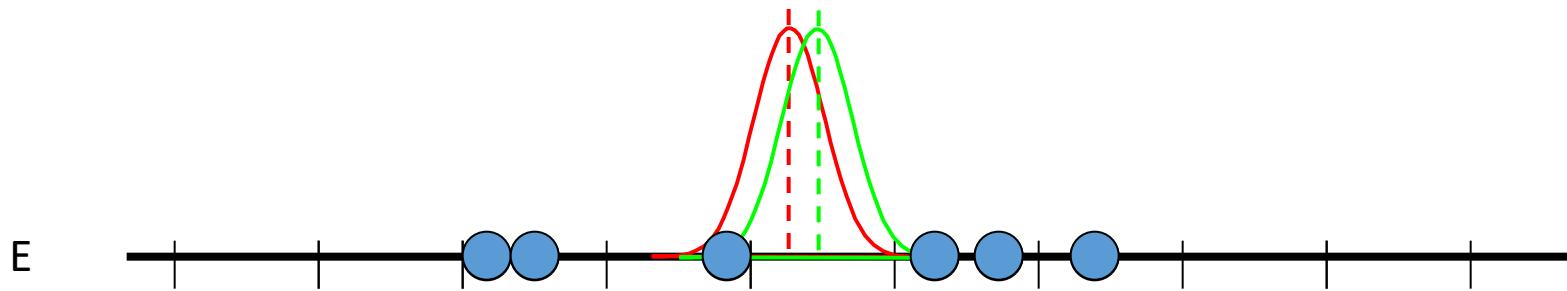
- M. maximize  $Q(\theta \mid \theta^{(t)})$

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

# EM in Pictures

**E-step:** Compute a *distribution* on the labels of the points, using current parameters

**M-step:** Update parameters using current guess of label distribution.



## EM for Mixture of Gaussians: E step

- Let's revisit the example:  $p(x | \theta, \rho) = \sum_{j=1}^m p(x | c_j, \theta_j) \rho_j$ 
  - with:  $p(x | c_j, \theta_j) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x - \mu_j)^2}{2\sigma^2}\right)$
  - Need to estimate  $\theta_1, \dots, \theta_m$  and  $\rho_1, \dots, \rho_m$
- Define:  $z_i^{(k)} = \begin{cases} 1 & \text{if sample } i \text{ was generated by component } k \\ 0 & \text{otherwise} \end{cases}$ 
  - and  $Z_i = \{z_i^{(1)}, \dots, z_i^{(m)}\}$  and  $Z = \{Z_1, \dots, Z_n\}$
  - We need the log-likelihood of observed X and hidden Z

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

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

# EM for Mixture of Gaussians: E step

- Omitting several steps
- ...
- We need to compute  $E_z[z_i^{(k)}]$  (the expected value of the latent variables)

$$\begin{aligned}E_z[z_i^{(k)}] &= \mathbf{0} * P(z_i^{(k)} = \mathbf{0} | \theta^{(t)}, \mathbf{x}_i) + \mathbf{1} * P(z_i^{(k)} = \mathbf{1} | \theta^{(t)}, \mathbf{x}_i) \\&= P(z_i^{(k)} = \mathbf{1} | \theta^{(t)}, \mathbf{x}_i) = \frac{p(\mathbf{x}_i | \theta^{(t)}, z_i^{(k)} = \mathbf{1})P(z_i^{(k)} = \mathbf{1} | \theta^{(t)})}{p(\mathbf{x}_i | \theta^{(t)})} \\&= \frac{\rho_k^{(t)} \exp\left(-\frac{1}{2\sigma^2}(\mathbf{x}_i - \mu_k^{(t)})^2\right)}{\sum_{j=1}^m P(\mathbf{x}_i | \theta^{(t)}, z_i^{(j)} = \mathbf{1})P(z_i^{(j)} = \mathbf{1} | \theta^{(t)})} = \frac{\rho_k^{(t)} \exp\left(-\frac{1}{2\sigma^2}(\mathbf{x}_i - \mu_k^{(t)})^2\right)}{\sum_{j=1}^m \rho_j^{(t)} \exp\left(-\frac{1}{2\sigma^2}(\mathbf{x}_i - \mu_j^{(t)})^2\right)}\end{aligned}$$

## EM for Mixture of Gaussians: M step

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

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

$$\begin{aligned} \frac{\partial}{\partial \mu_k} Q(\theta | \theta^{(t)}) &= \sum_{i=1}^n E_z[z_i^{(k)}] \frac{(x_i - \mu_k)}{\sigma^2} = 0 \\ \Rightarrow \text{new } \mu_k &= \mu_k^{(t+1)} = \frac{\sum_{i=1}^n E_z[z_i^{(k)}] x_i}{\sum_{i=1}^n E_z[z_i^{(k)}]} \end{aligned}$$

the mean for class k is the 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 the 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 EM Algorithm: Univariate Gaussian Case

*The algorithm on this slide applies ONLY to the univariate Gaussian case with known variances*

- randomly initialize  $\mu_1, \dots, \mu_m$  and  $\rho_1, \dots, \rho_m$  (subject to  $\sum \rho_i = 1$ )
- iterate the following 2 steps until there is no change in  $\mu_1, \dots, \mu_m$  and  $\rho_1, \dots, \rho_m$

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

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

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

# The EM Algorithm

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

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

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

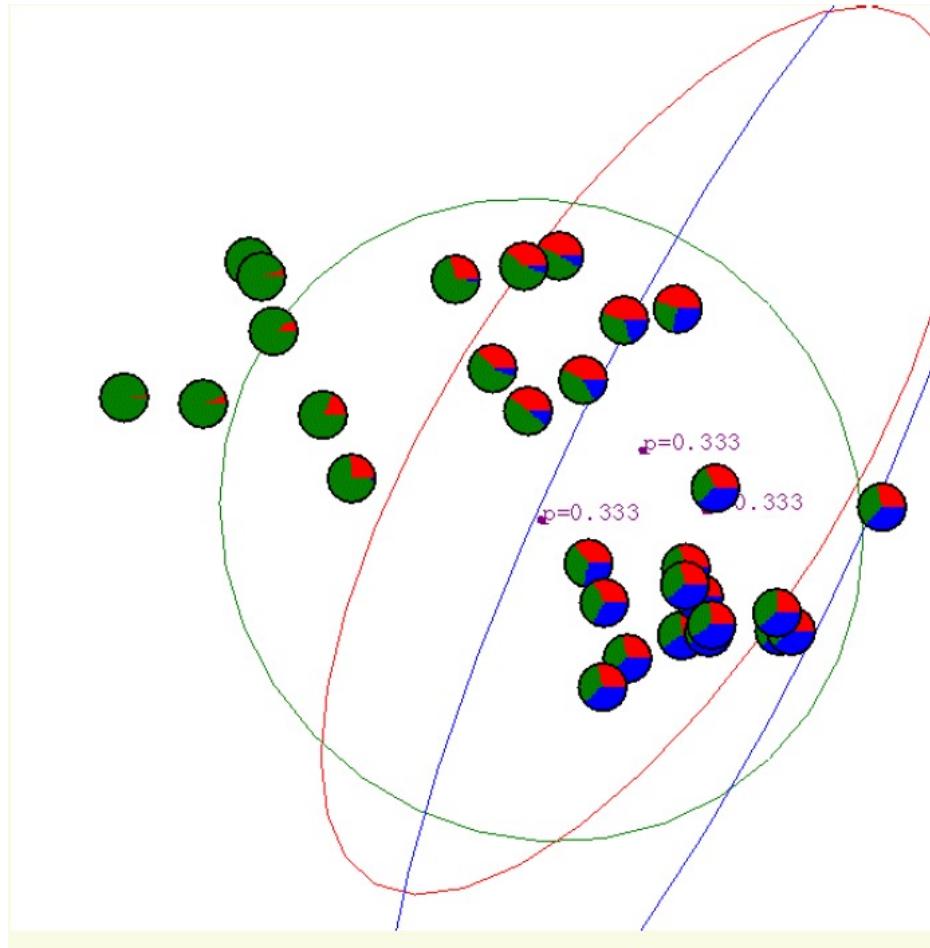
- M step

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

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

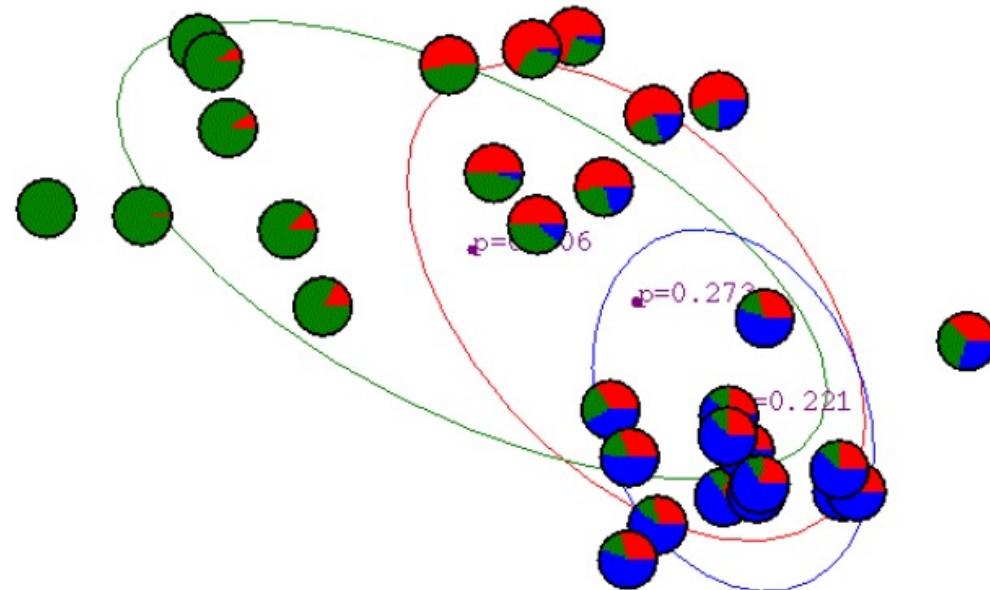
$$\Sigma_k = \frac{\sum_{i=1}^n E_z[z_i^{(k)}] (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^n E_z[z_i^{(k)}]}$$

# EM Gaussian Mixture Example



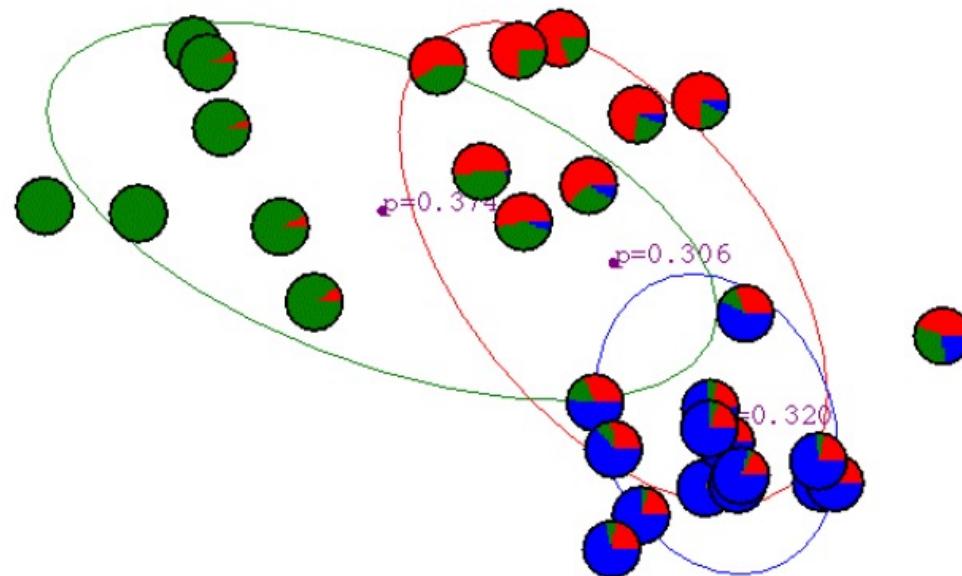
# EM Gaussian Mixture Example

*After first iteration*

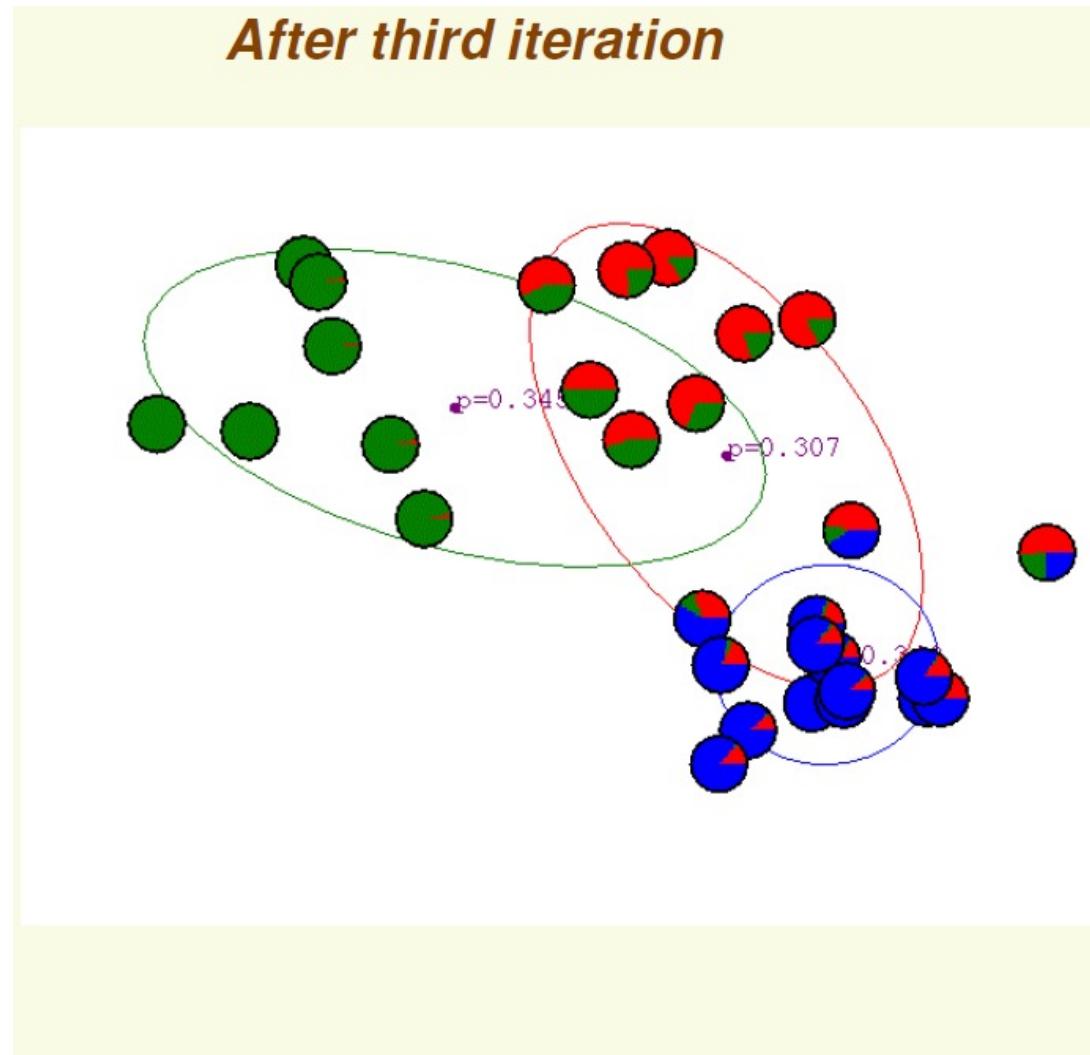


# EM Gaussian Mixture Example

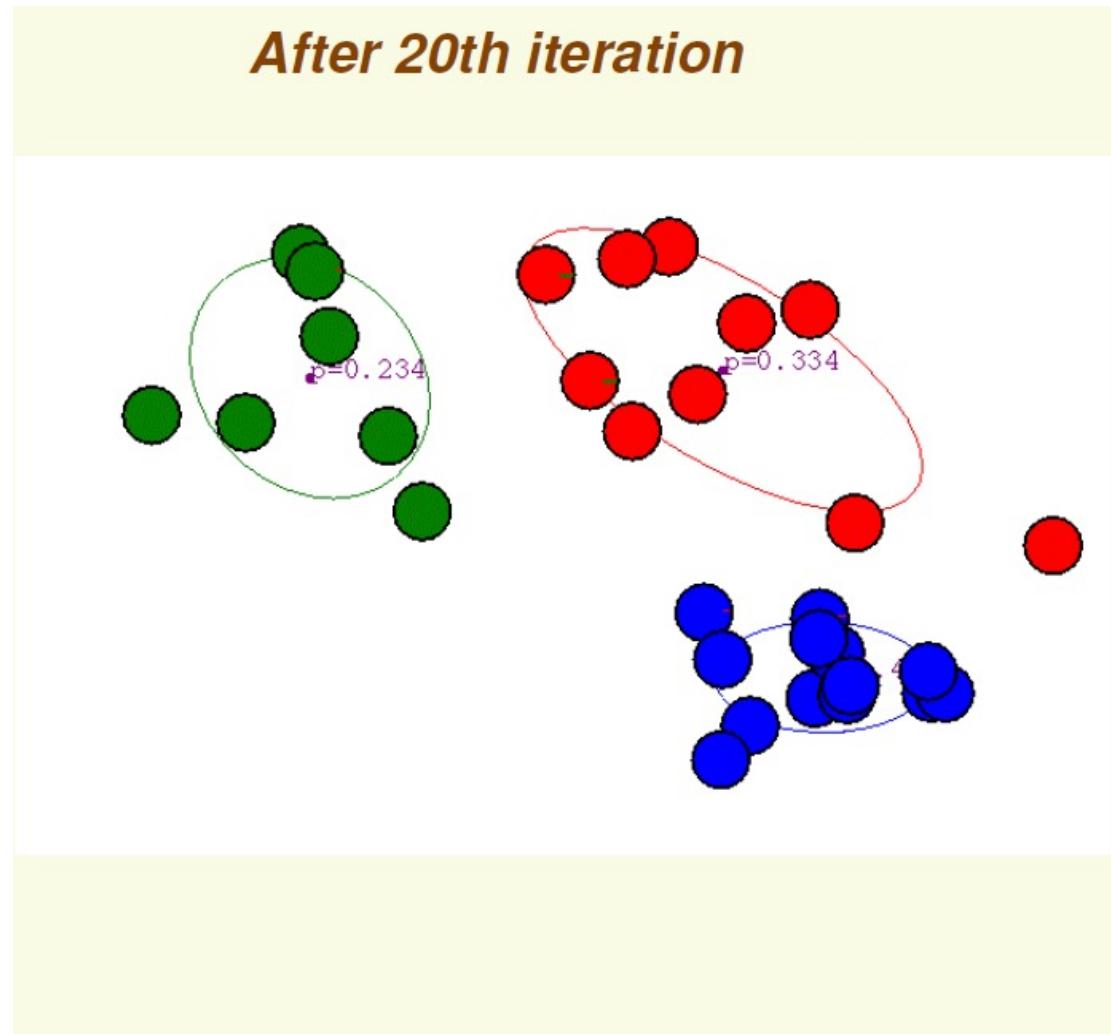
*After second iteration*



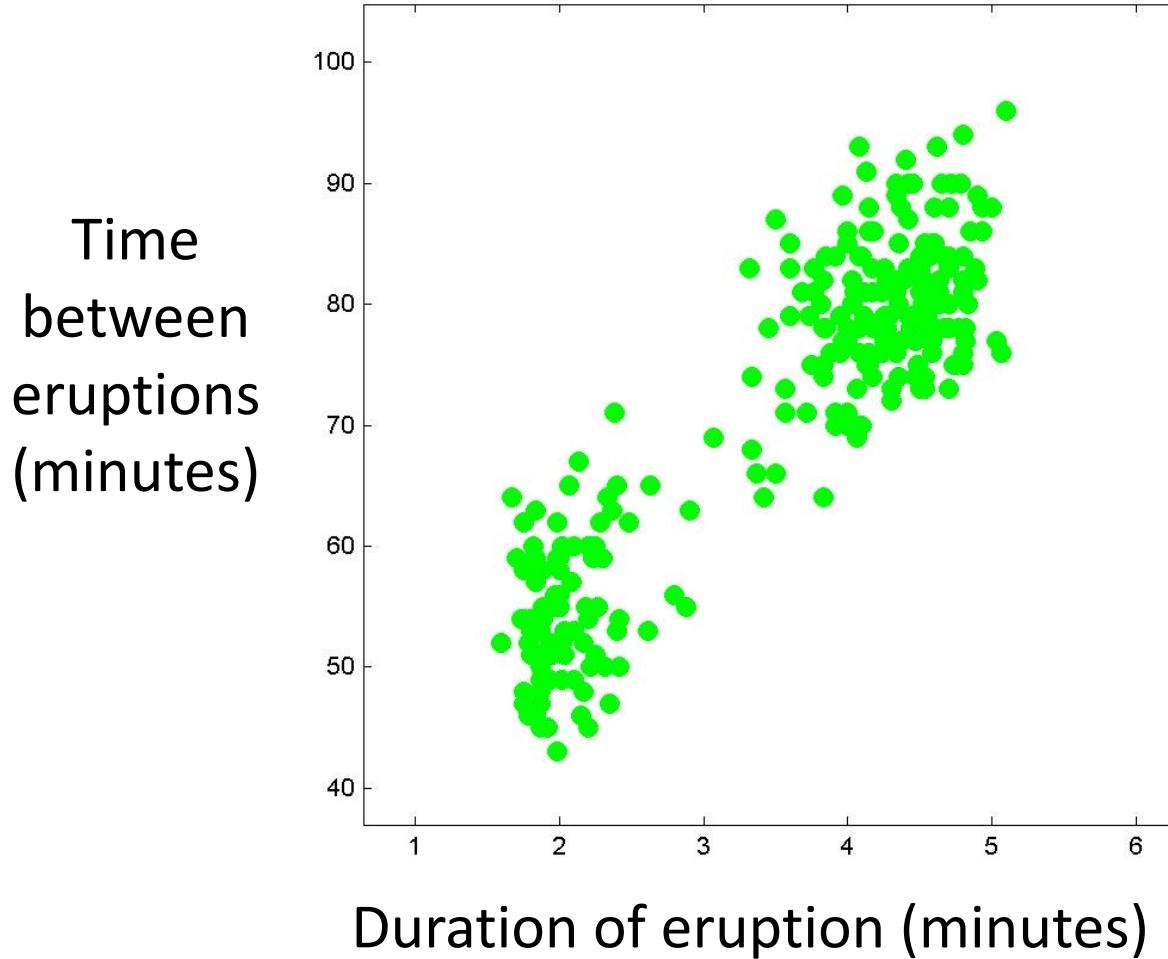
# EM Gaussian Mixture Example

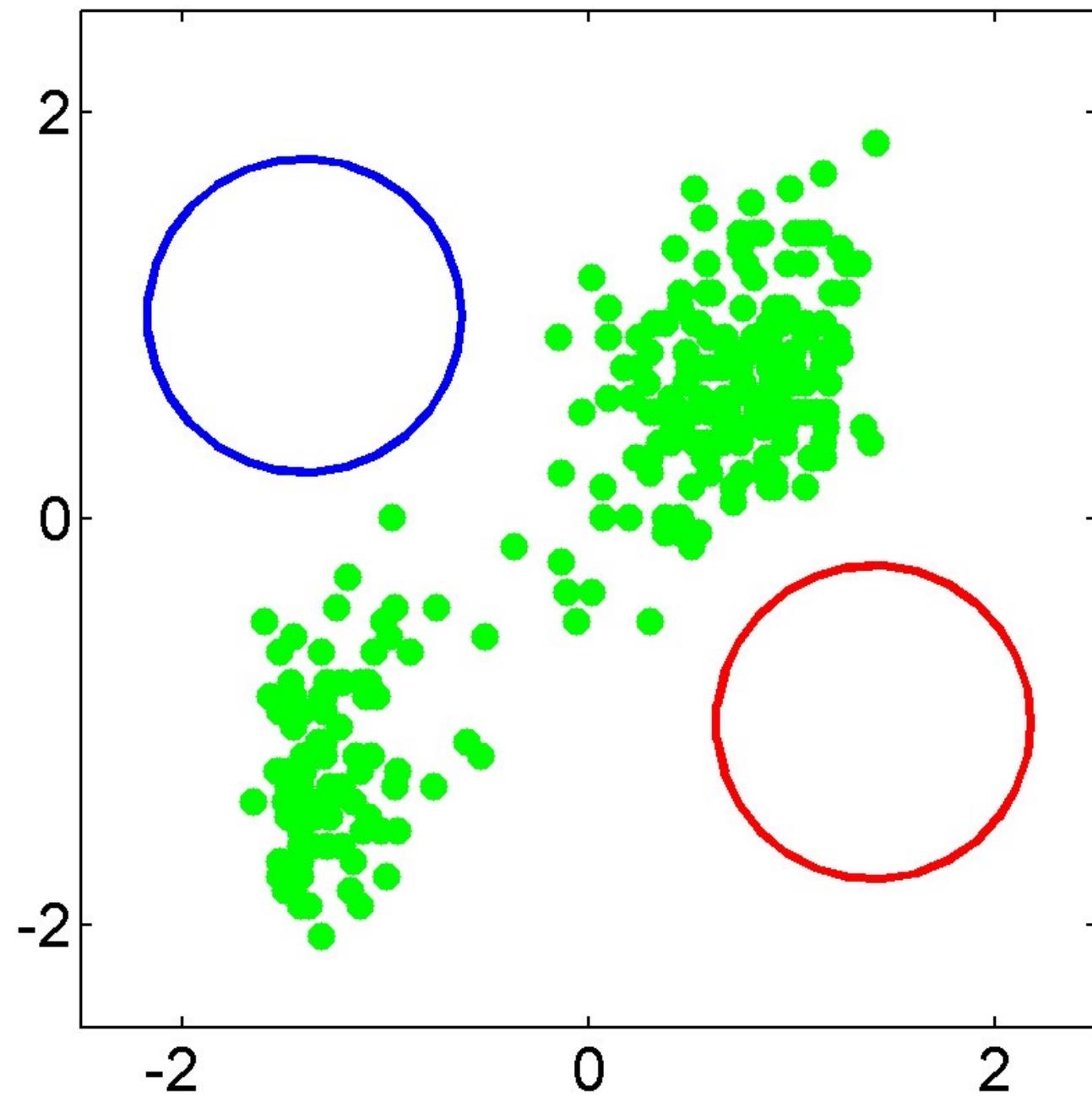


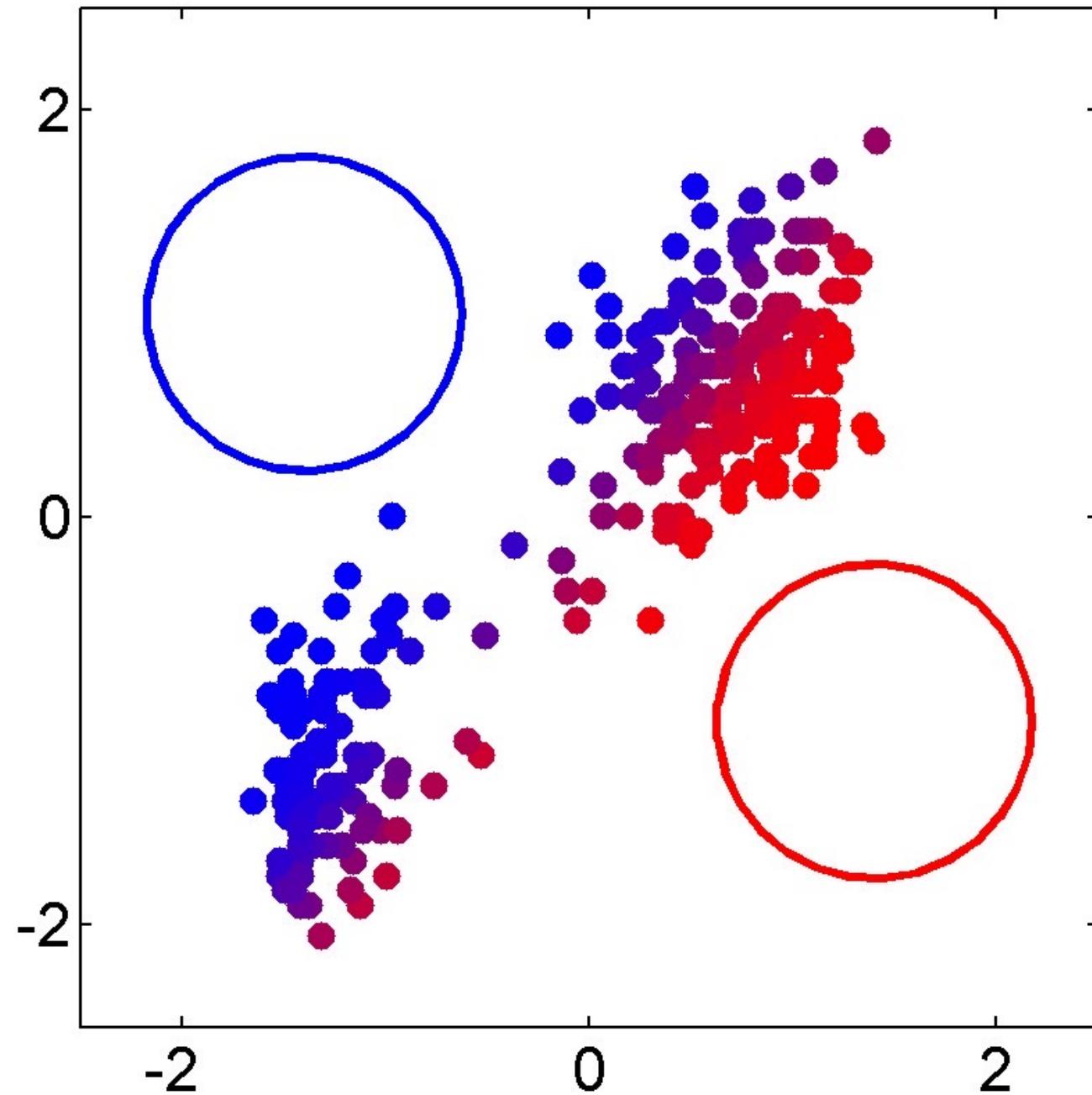
# EM Gaussian Mixture Example

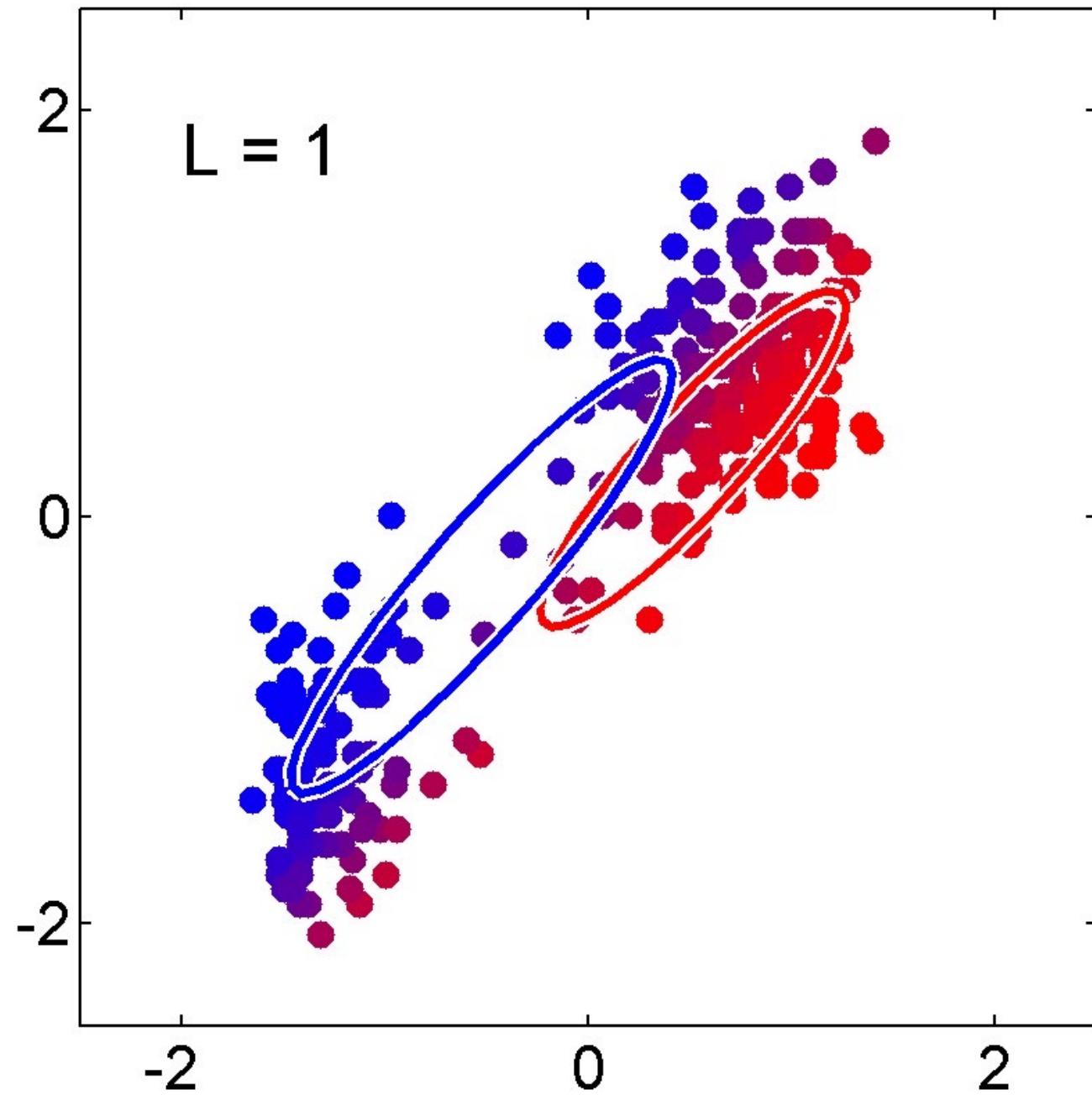


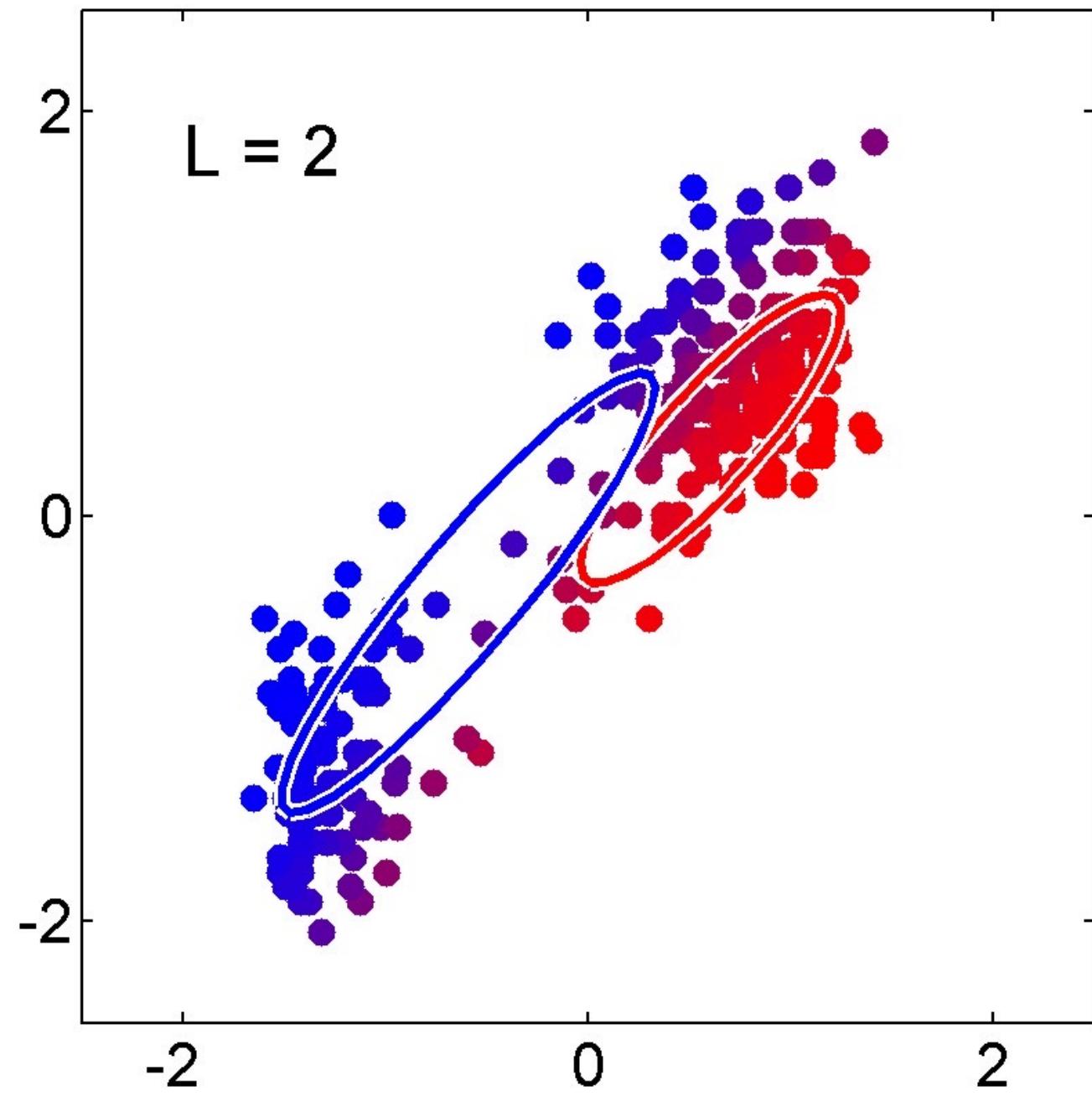
# Volcano Eruption Data Set

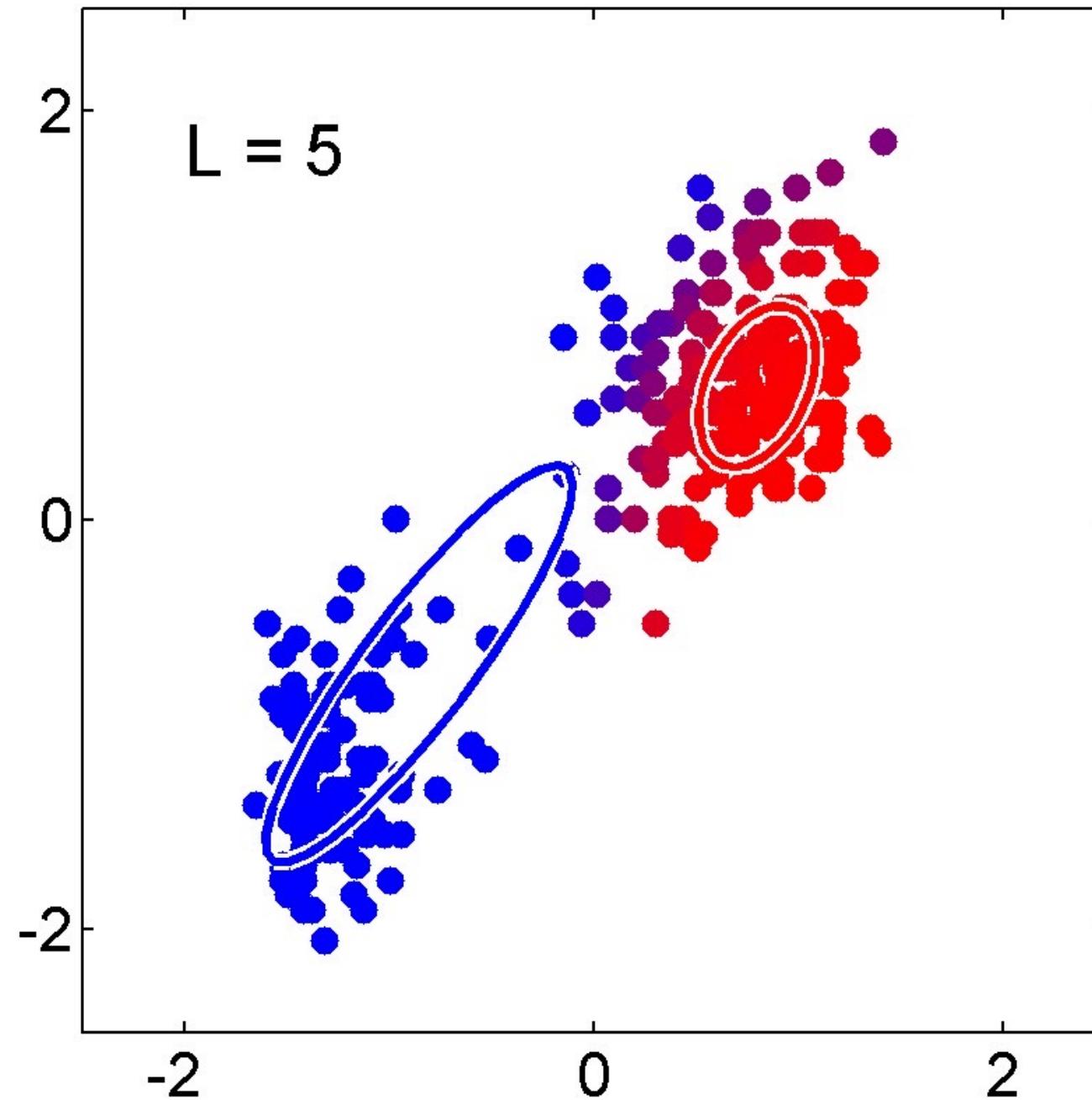


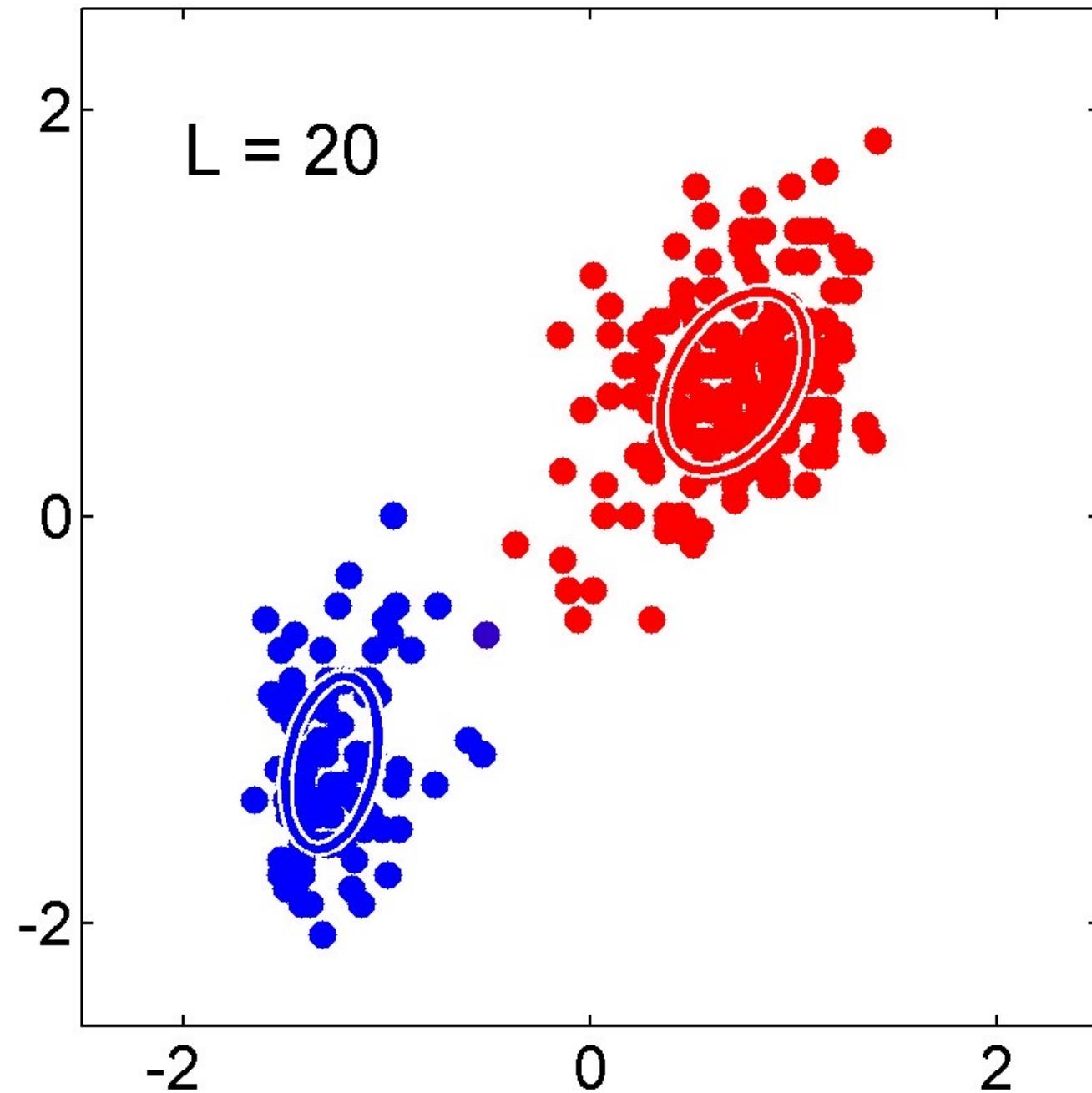












# EM Summary

- Advantages
  - If the assumed data distribution is correct, the algorithm works well
- Disadvantages
  - If the assumed data distribution is wrong, results can be quite bad
  - In particular, bad results if incorrect number of mixture components is used

# Q & A

- You can always approach me:
  - Xinchao Wang, ECE, [xinchao@nus.edu.sg](mailto:xinchao@nus.edu.sg)
- Thanks, and stay safe and healthy!
- Best of luck in your future endeavor!