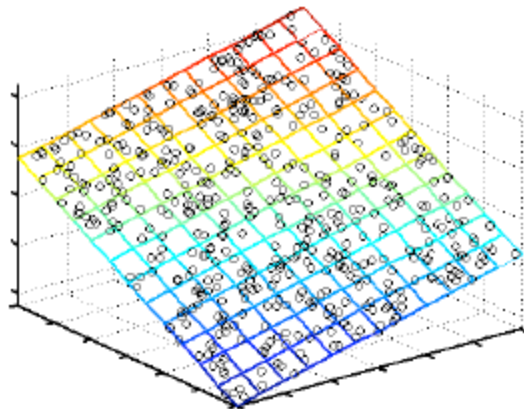
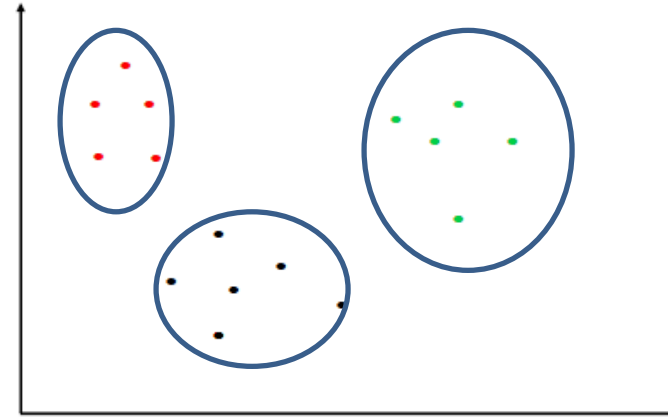


Unsupervised Learning

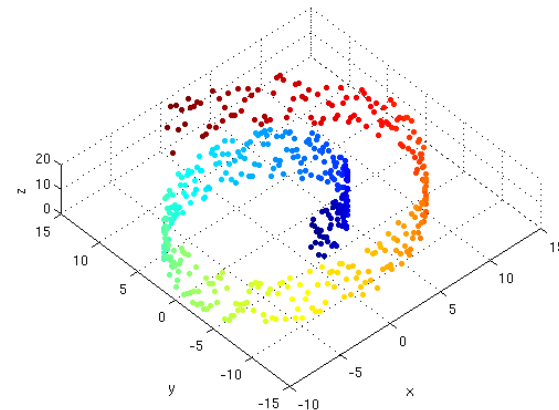
CS534

What can we learn from unlabeled data?

- Finding Group of clusters in the data
- Finding low dimensional representation – dimension reduction

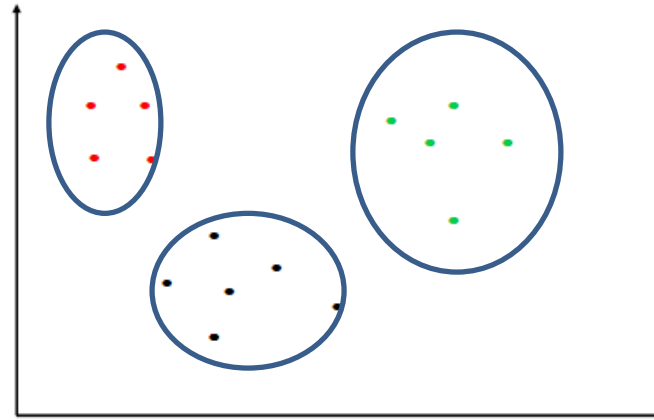


PCA



Nonlinear embedding

Clustering



- Clustering: the process of grouping a set of objects into classes of similar objects
 - high within-class similarity
 - low cross-class similarity
- It is the most common form of unsupervised learning

Example Applications

- Find genes that are similar in their functions
- Group documents based on topics
- Categorize customers based on their purchase history
- Group images based on their contents
-

Critical Issues in Clustering

- What makes objects similar?
 - Definition of "similarity/distance"
- How many clusters?
 - Fixed a priori?
 - Completely data driven?
 - Avoid "trivial" clusters - too large or small
- Clustering Algorithms
 - Flat vs hierarchical
 - Hard vs soft

What is similarity



Hard to define but
We know it when we see it

- The real meaning of similarity is a philosophical question. We will take a more pragmatic approach
 - Depends on representation and algorithm. For many rep./alg., easier to think in terms of a distance (rather than similarity) between vectors

What properties should a distance measure have?

- D must be Symmetric
 - $D(A, B) = D(B, A)$
 - Otherwise, we can say A looks like B but B does not look like A
- Positivity, and self-similarity
 - $D(A, B) \geq 0$, and $D(A, B) = 0$ iff $A = B$
 - Otherwise there will be different objects that we cannot tell apart
- Must satisfy triangle inequality
 - $D(A, B) + D(B, C) \geq D(A, C)$
 - Otherwise one can say “ A is like B , B is like C , but A is not like C at all”

Distance Measures: Minkowski Metric

- Suppose two object x and y both have d features
 - $x = (x_1, \dots, x_d), y = (y_1, \dots, y_d)$

- The Minkowski metric of order r is defined by

$$d(x, y) = \sqrt[r]{\sum_i |x_i - y_i|^r}$$

- Common Minkowski metrics:

- Euclidean($r=2$): $d(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$, also called L_2 distance
- Manhattan distance($r=1$) : $d(x, y) = \sum_i |x_i - y_i|$, also called L_1 distance
- “Sup” distance($r = +\infty$): $d(x, y) = \max_i |x_i - y_i|$, also called L_∞ distance

Other Distances

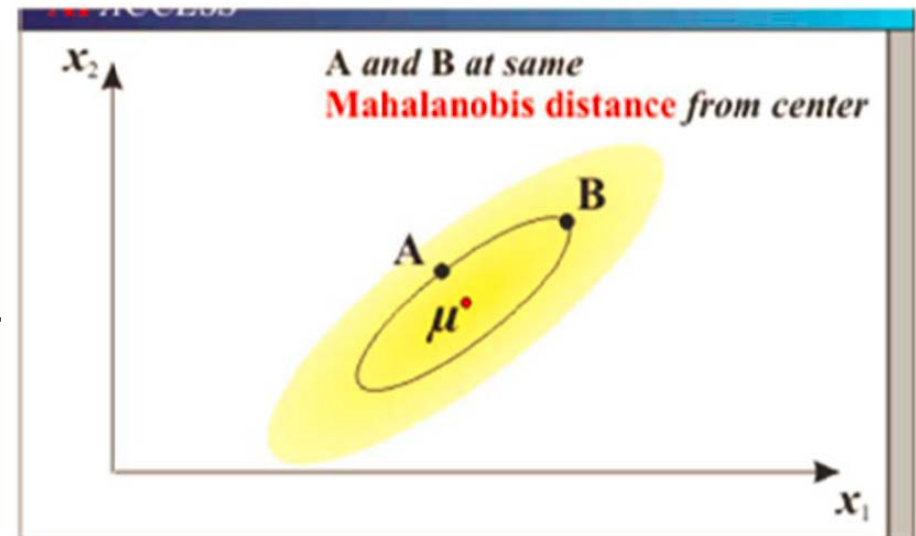
- Hamming distance (Manhattan distance on binary features)
 - # of features that differ
 - e.g.: distance of two sites based on their species composition

	sp1	sp2	sp3	sp4	sp5	sp6	sp7	sp8	sp9
Site A:	1	0	1	1	0	0	1	0	1
Site B:	0	0	1	0	1	1	1	0	1

$$D(A, B) = 4$$

- Mahalanobis distance (assuming \mathbf{x}, \mathbf{y} follows a Gaussian distribution with covariance matrix Σ)

$$D(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T \Sigma^{-1} (\mathbf{x} - \mathbf{y})}$$



Similarities

- Cosine similarities – commonly used to measure document similarity

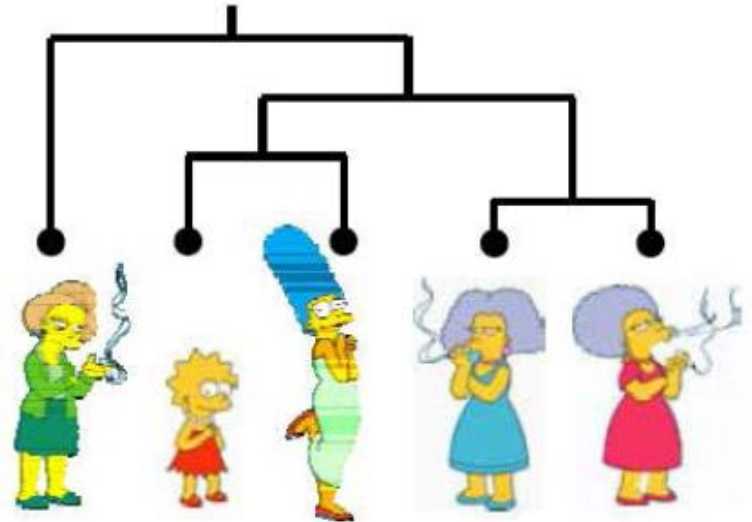
$$\cos(\mathbf{x}, \mathbf{x}') = \frac{\langle \mathbf{x} \cdot \mathbf{x}' \rangle}{|\mathbf{x}| \cdot |\mathbf{x}'|}$$

- Kernels – e.g., RBF (Gaussian) Kernel

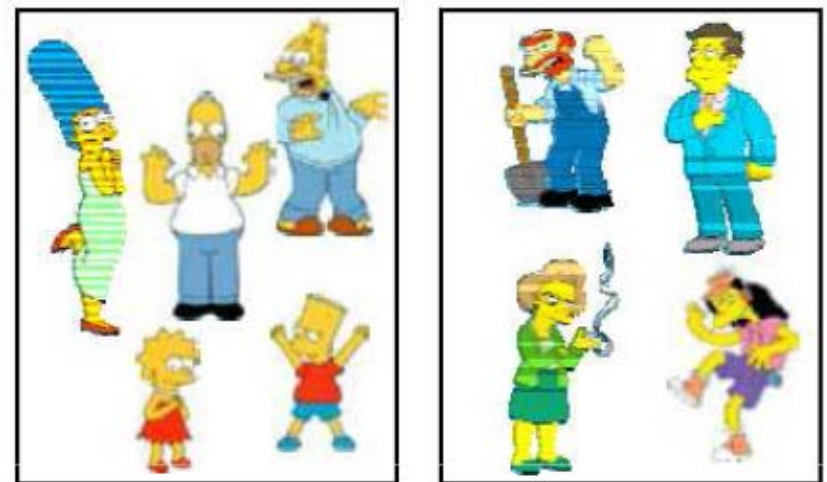
$$S(X, X') = \exp \frac{-|X - X'|^2}{2\sigma^2}$$

Clustering algorithms

- Hierarchical algorithms (not covered)
 - Bottom up – agglomerative
 - Top down – divisive

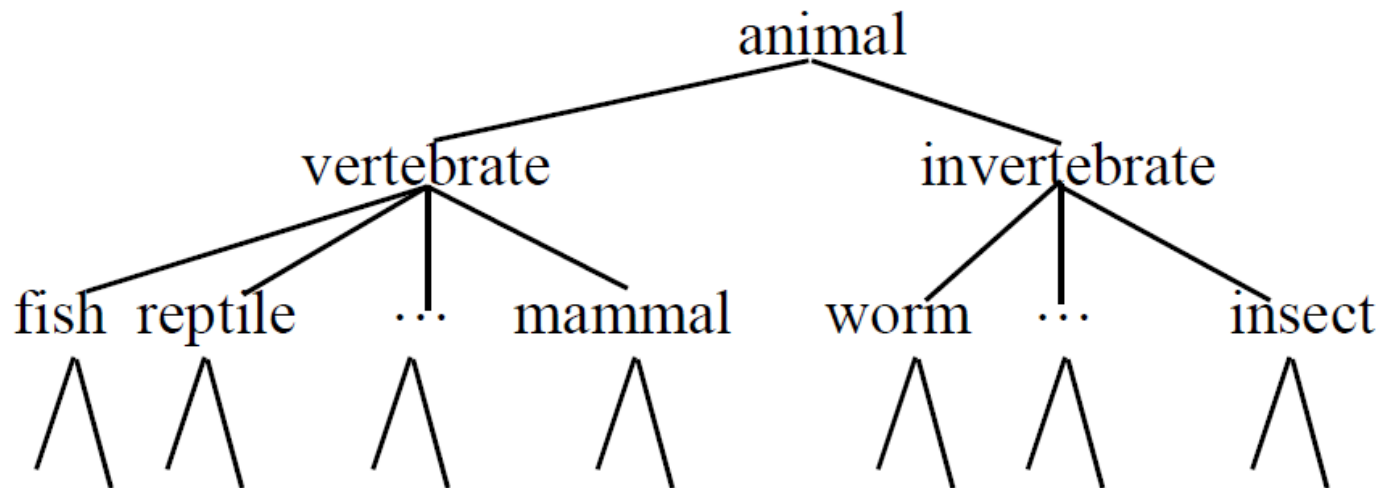


- Flat clustering algorithms
 - K-means
 - Mixture of Gaussian
 - Spectral Clustering (not covered)



Hierarchical Clustering

- Given a set of objects, build a tree-based taxonomy



- Hierarchies are convenient way for organizing information

Not covered

Hierarchical Agglomerative Clustering (HAC)

- Starts with each object in a separate cluster
- Repeatedly joins the closest pair of clusters
- until there is only one cluster

The history of merging forms a tree of hierarchy

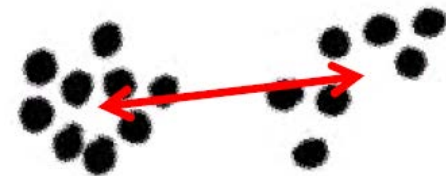
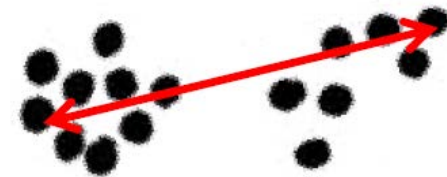
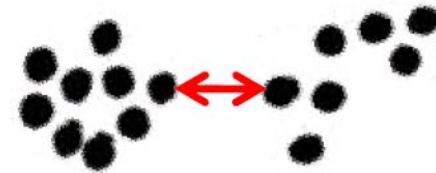
Question: how to measure the “closeness” of two clusters?

Not covered

Closest Pair of Clusters

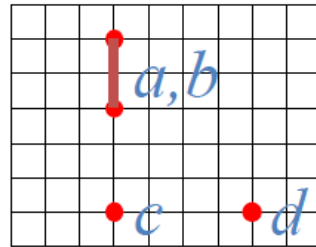
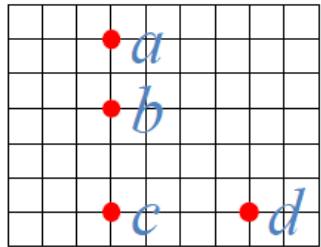
The distance between two clusters is defined as the distance between:

- Single-link
 - The nearest pair of points
- Complete-link
 - The furthest pair of points
- Centroid
 - The center of gravity
- Average-link
 - Average of all cross-cluster pairs

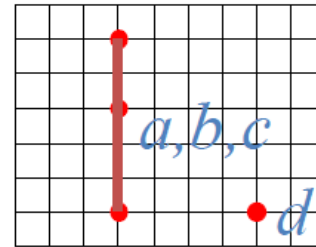


Not covered

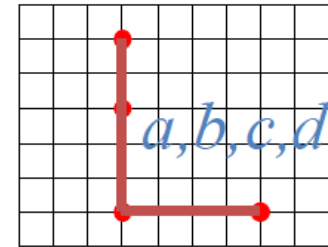
Single Link Method



(1)



(2)



(3)

	<i>b</i>	<i>c</i>	<i>d</i>
<i>a</i>	2	5	6
<i>b</i>		3	5
<i>c</i>			4

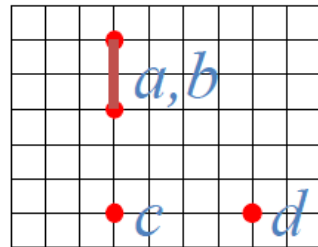
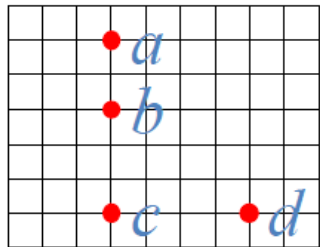
	<i>b</i>	<i>c</i>	<i>d</i>
<i>a</i>	2	5	6
<i>b</i>		3	5
<i>c</i>			4

	<i>c</i>	<i>d</i>
<i>a, b</i>	3	5
<i>c</i>		4

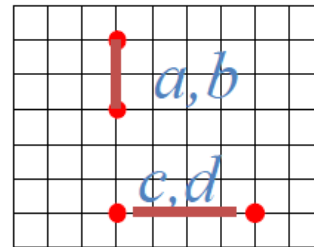
	<i>d</i>
<i>a, b, c</i>	4

Not covered

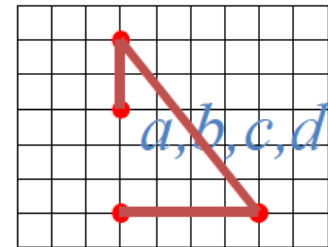
Complete Link Method



(1)



(2)



(3)

	<i>b</i>	<i>c</i>	<i>d</i>
<i>a</i>	2	5	6
<i>b</i>		3	5
<i>c</i>			4

	<i>b</i>	<i>c</i>	<i>d</i>
<i>a</i>	2	5	6
<i>b</i>		3	5
<i>c</i>			4

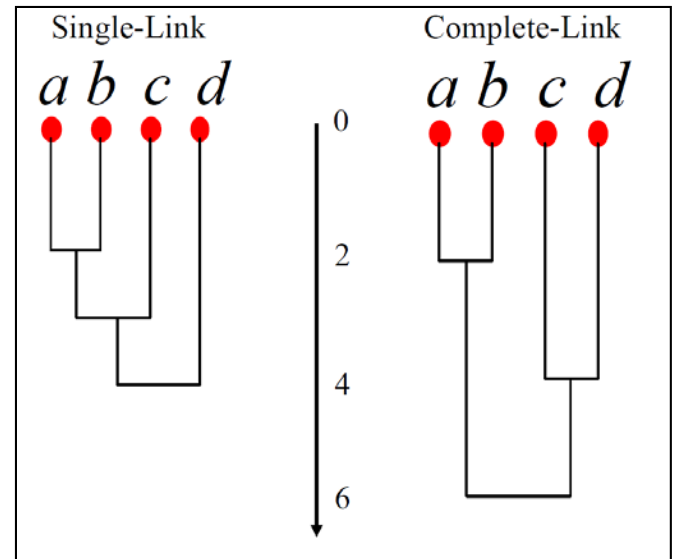
	<i>c</i>	<i>d</i>
<i>a, b</i>	5	6
<i>c</i>		4

	<i>c, d</i>
<i>a, b</i>	6

Not covered

Visualization: Dendrogram

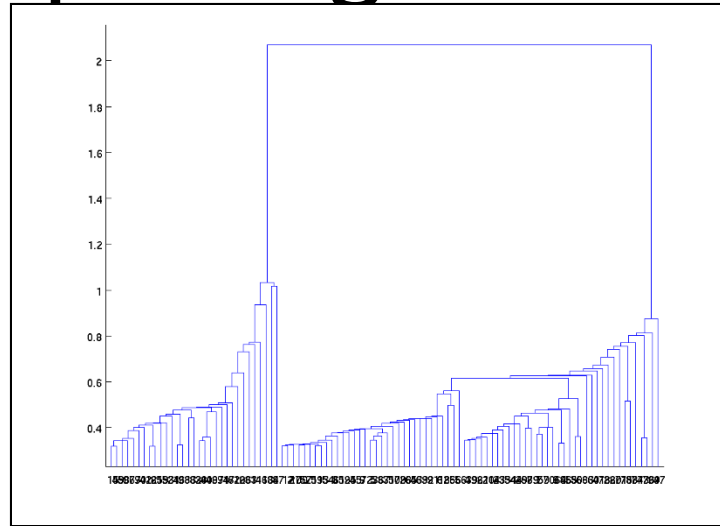
- Height of the joint = the distance between the two merge clusters
- The merge distance monotonically increases as we merge more and more for
 - Single, complete and average linkage methods
 - Not for the centroid method



This example is shown upside down.

Not covered

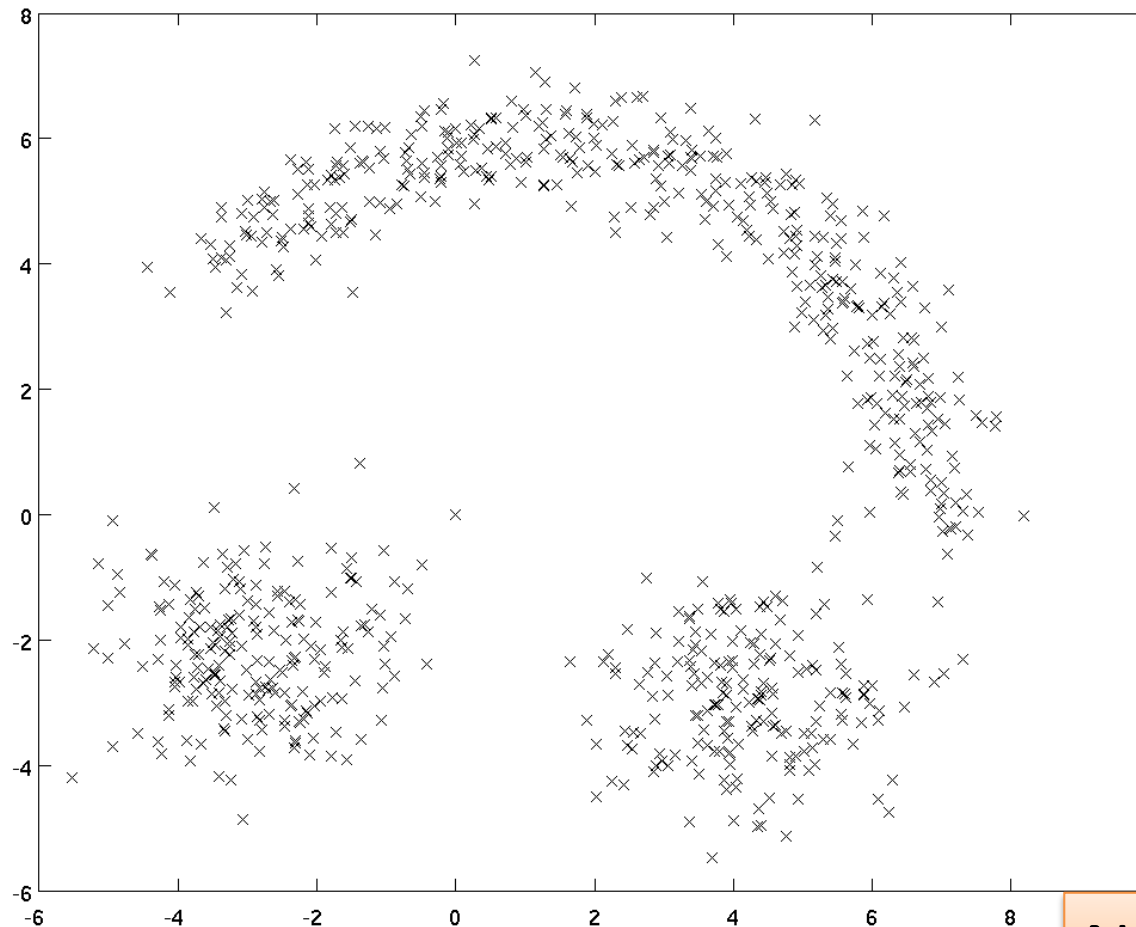
Interpreting Dendrogram



- Dendrogram can be used to visually identify
 - the number of clusters in data
 - A horizontal cut creates a partition
 - Moving the cut from root down creates more clusters
 - Large gaps indicate good cutting points
 - well-formed clusters
 - Some clusters are better formed than the other
 - This can be easy to see on a dendrogram

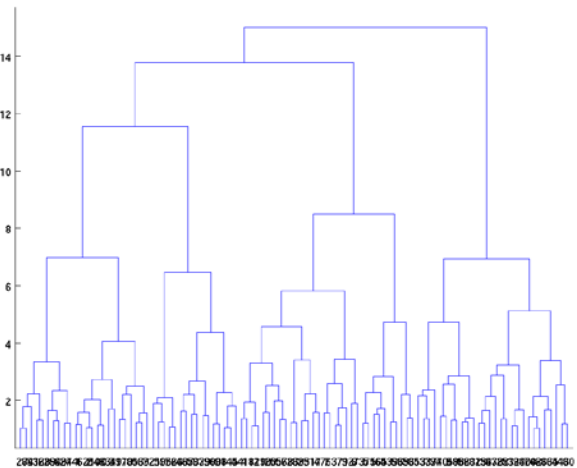
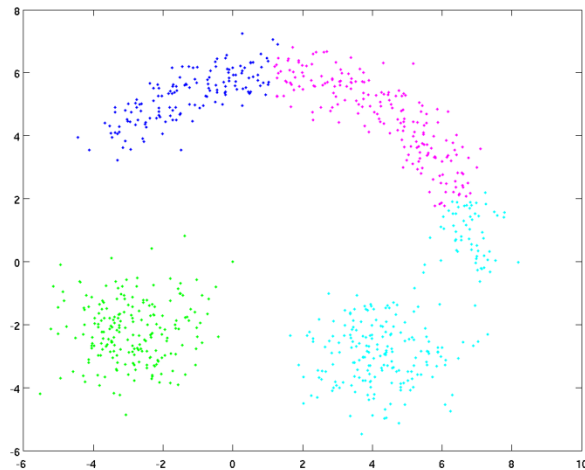
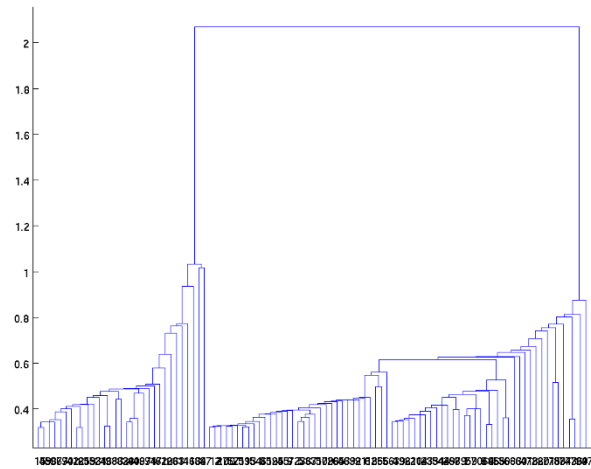
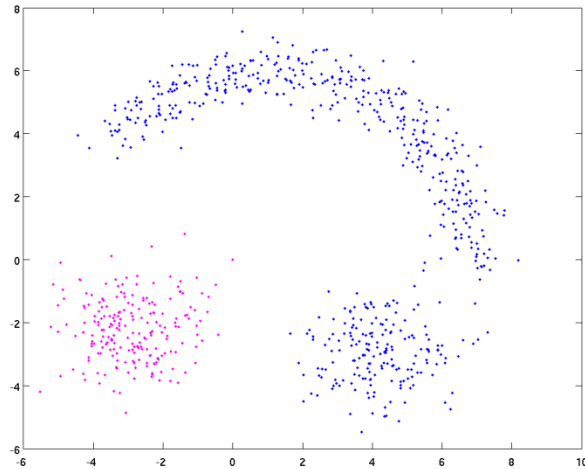
Not covered

Example



Not covered

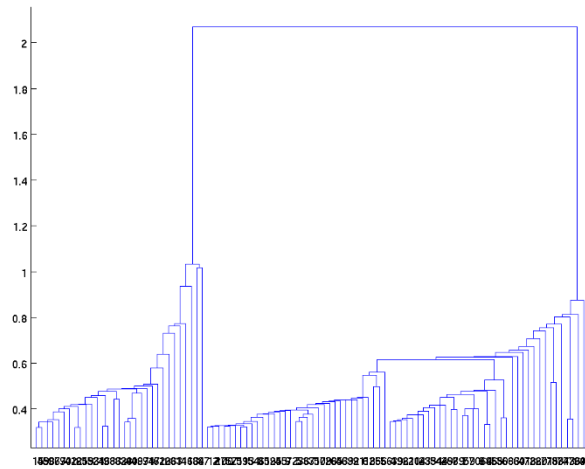
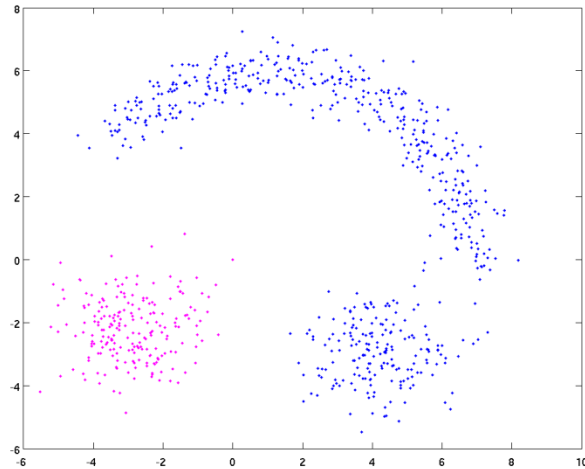
Single Link vs. Complete Link



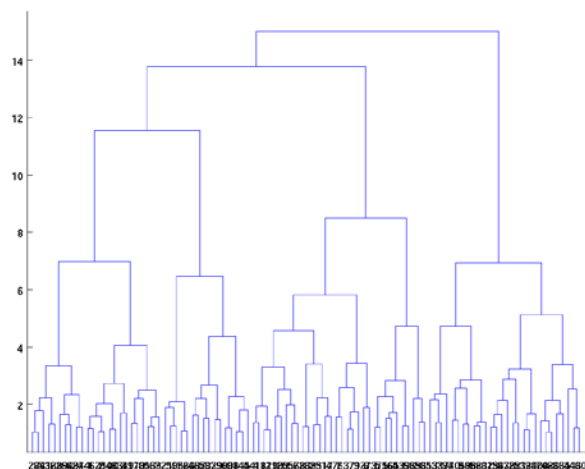
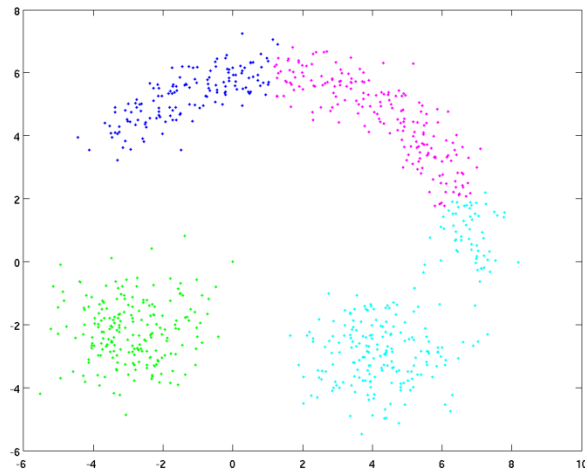
Not covered

Which is single link? Which is complete link?

Single Link vs. Complete Link



Single



Complete

Not covered

- Single-link creates straggly clusters due to chaining effect

Flat Clustering

- Given D , a data set of n points, we want to find k clusters
- For each cluster we would like to represent all points of that cluster with one representative, and the representation error should be minimized
- This leads to the following objective function

$$\min_{\mu, C} \sum_{i=1}^k \sum_{x \in C_i} |x - \mu_i|^2$$

Where $C = \{C_1, C_2, \dots, C_k\}$ is a partition of D such that

$$C_i \cap C_j = \emptyset, \forall i \neq j \text{ and } D = \cup_i C_i$$

and μ_i is the representative of C_i

- A combinatorial optimization problem
 - Discrete solution space
 - Exhaustive search for an optimal solution is not feasible

An Iterative Solution

- ***Initialization:*** Start with a random partition of the data
- ***Iterative step:*** the cluster assignments and cluster centers are updated to improve the objective
- ***Stopping criterion:*** if no improvement can be achieved.

K-Means

Algorithm

Input – Desired number of clusters, k

Initialize – the k cluster centers (randomly if necessary)

Iterate –

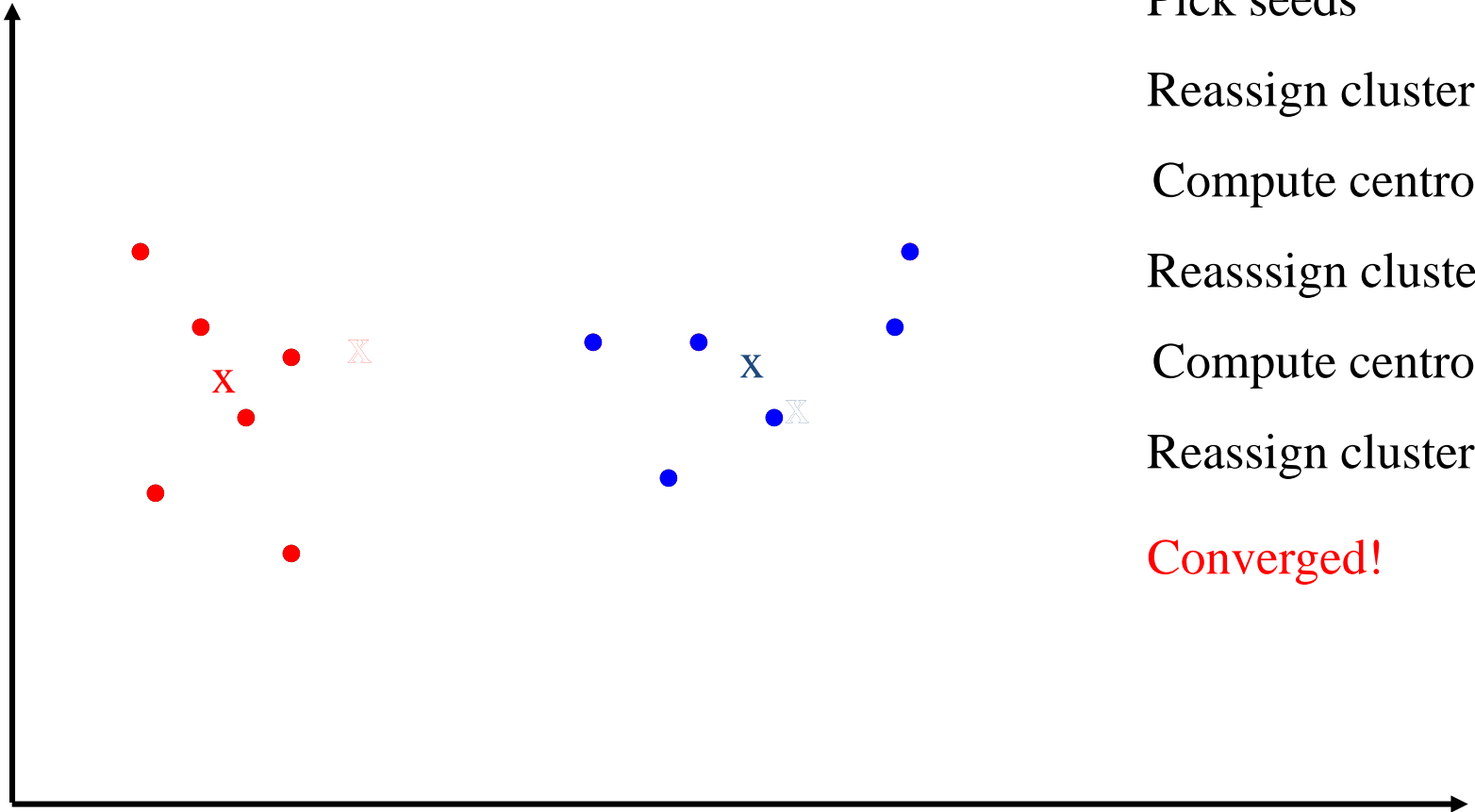
1. Assigning each of the N data points to its nearest cluster centers
2. Re-estimate the cluster center by assuming that the current assignment is correct

$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

Termination –

If none of the data points changed membership in the last iteration, exit.
Otherwise, go to 1

Demo: K-Means Example (K=2)



Pick seeds

Reassign clusters

Compute centroids

Reassign clusters

Compute centroids

Reassign clusters

Converged!

Does KMeans Converge?

Objective

$$\min_{\mu, C} \sum_{i=1}^k \sum_{x \in C_i} |x - \mu_i|^2$$

1. Fix μ , optimize C

- Assign each point to its closet center

Step 1 of kmeans

2. Fix C , optimize μ :

$$\min_{\mu} \sum_{i=1}^k \sum_{x \in C_i} |x - \mu_i|^2$$

- Take partial derivative of μ_i and set to zero, we have

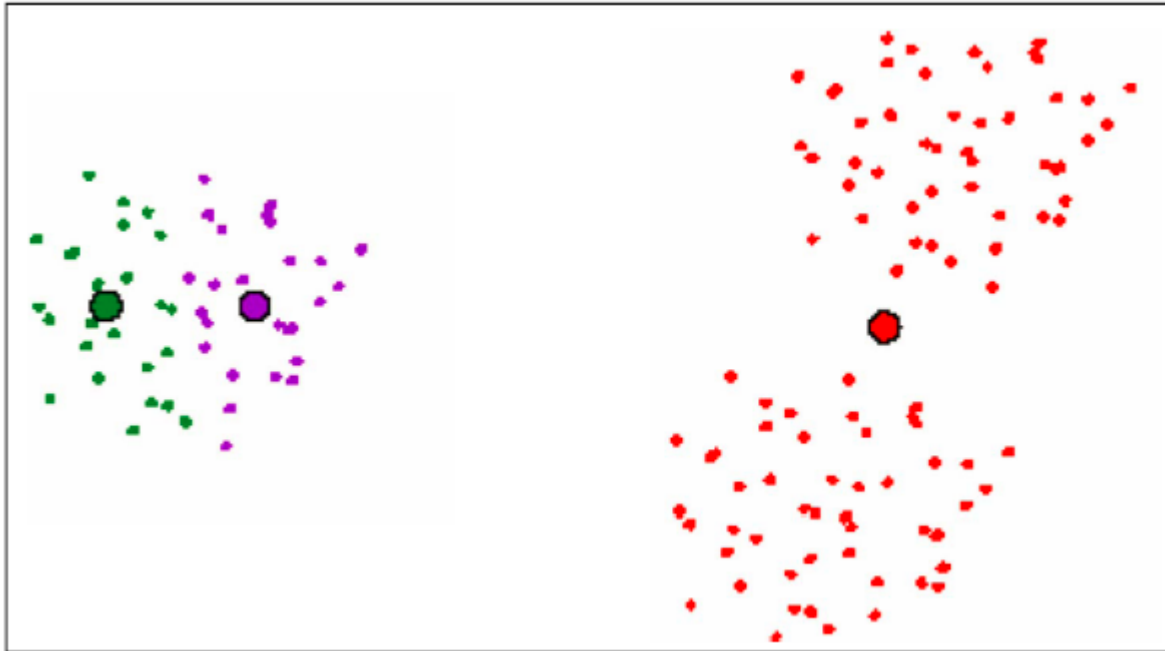
$$2 \sum_{x \in C_i} (x - \mu_i) = 0 \Rightarrow \mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

Step 2 of kmeans

Each iteration is guaranteed to decrease the objective and there are finite possible C 's – thus guaranteed to converge --- but only to local minimum

Impact of Initial Seeds

- Highly sensitive to the initial seeds



- Multiple random trials: choose the one with best sum of squared loss (important!)
- Heuristics for choosing better centers
 - choose initial centers to be far apart – furthest first traversal (kmeans++)

More Comments

- K-Means is exhaustive:
 - Cluster every data point, no notion of outlier
 - Outliers cause problems
 - Become singular clusters
 - Bias the centroid estimation
- K-medoids methods is more robust to outliers
 - Cluster medoid: must be one of the data points, one that has the minimum sum squared distance to all data points in the cluster
 - More expensive to compute
 - For each pt: sum squared dist with all other pts in cluster $O(|C|^2)$
- Need to specify k : difficult in practice

Hard vs. Soft Clustering

- Hard clustering:
 - Data point is deterministically assigned to one and only one cluster
 - But in reality clusters may overlap
- Soft-clustering:
 - Data points are assigned to clusters with certain probabilities
- Often referred to as model-based clustering for the use of probabilistic model for clusters

Side track: Gaussian Bayes Classifier

- We have k classes in our data
- Each class contains data generated from a particular Gaussian distribution $N(\mu_c, \Sigma_c), c = 1, \dots, k$
- The data is generated as follows:
 - first randomly select the class label according to prior $p(y)$
 - draw a random sample from the Gaussian distribution of that particular class
 - Repeat n times to generate n points
- In supervised learning, we observe the resulting (x, y) pairs and we simply need to estimate $p(y = c)$ and $p(x|y = c)$ for $c = 1, \dots, k$

MLE Estimates of Mean and Covariance for Gaussian

Given that $p(x|y = c) \sim N(\mu_c, \Sigma_c)$, and given a set of n_c training examples with $y = c$:

$$\mu_c = \frac{1}{\sum_i I(y_i = c)} \sum_{y_i=c} x_i$$

$$\Sigma_c = \frac{1}{\sum_i I(y_i = c)} \sum_{y_i=c} (x_i - \mu_c) (x_i - \mu_c)^T$$

Back to Unsupervised Learning

- Now assume we know our data is generated in the same way
- But for unsupervised learning, we don't have the y 's, and only observe x 's
 - We observe a mixture of Gaussians
- How can we learn the correct model from the incomplete data?
- We will still use Maximum likelihood estimation

Gaussian Mixture Model

$$\begin{aligned} P(\mathbf{x}) &= \sum_{i=1}^k P(\mathbf{x}, y = i) \\ &= \sum_{i=1}^k P(\mathbf{x} | y = i) P(y = i) \\ &= \sum_{i=1}^k \alpha_i P(\mathbf{x} | \theta_i) \end{aligned}$$

$\alpha_i = P(y = i)$: class priors
Mixing parameter

$\theta_i = \{\mu_i, \Sigma_i\}$

Goal of learning:

- Given a set of \mathbf{x} 's, find the values of $\{\alpha_1, \dots, \alpha_k, \theta_1, \dots, \theta_k\}$ that maximize the likelihood of observing the \mathbf{x} 's

Maximum Marginal Likelihood

$$\begin{aligned}\arg \max_{\theta} \prod_j P(\mathbf{x}^j) &= \arg \max_{\theta} \prod_j \sum_{i=1}^k P(\mathbf{x}^j, y^j = i) \\ &= \arg \max_{\theta} \sum_{j=1}^n \underbrace{\log \sum_{i=1}^k P(\mathbf{x}^j, y^j = i)}\end{aligned}$$

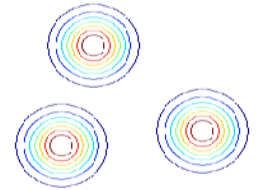
log sum is difficult to optimize !

Gradient ascent is doable but very inefficient

Expectation Maximization (EM)

- Commonly used approach for dealing with hidden (missing) data
 - Here the cluster labels are hidden
- Iterative algorithm that starts with an initial guess of the model parameters
- Iteratively performs two linked steps:
 - **Expectation (E-step)**: given current model parameters λ_t , compute the expectation for the hidden (missing) data
 - **Maximization (M-step)**: re-estimate the parameters λ_{t+1} assuming that the expected values computed in the E-step are the true values
- We will first show how it works for mixture of Gaussian

EM – simple case



- A simple case: spherical Gaussians
 - We have unlabeled data x^1, \dots, x^m
 - We know there are K classes
 - We know $\alpha_1 = P(y = 1), \dots, \alpha_K = P(y = K)$
 - We don't know $\mu_1 \dots \mu_K$, but know the common variance σ^2

Start with an initial guess for μ_1, \dots, μ_K ,

1. If we know μ_1, \dots, μ_K , we can easily compute probability that a point x^j belongs to class k :

$$P(y = k | x^j) \propto \exp\left(-\frac{1}{2\sigma^2} |x^j - \mu_k|^2\right) p(y = k)$$

Simply evaluate this, then normalize

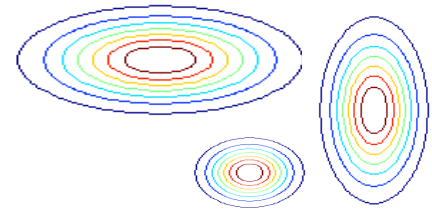
E-step

2. If we know *the* probability that each point belongs to each class, we can estimate the μ_1, \dots, μ_K

$$\mu_k = \frac{\sum_{j=1}^m P(y=k|x^j) x^j}{\sum_{j=1}^m p(y=k|x^j)}$$

M-step

EM – Axis-aligned Gaussian



- We have unlabeled data x^1, \dots, x^m
- We know there are K classes
- We know that the Gaussians are axis aligned

$$\Sigma_i = \begin{pmatrix} \sigma_{i,1}^2 & 0 & 0 & \dots & 0 & 0 \\ 0 & \sigma_{i,2}^2 & 0 & \dots & 0 & 0 \\ 0 & 0 & \sigma_{i,3}^2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_{i,m-1}^2 & 0 \\ 0 & 0 & 0 & \dots & 0 & \sigma_{i,m}^2 \end{pmatrix}$$

Start with an initial guess for $\mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_K, \alpha_1, \dots, \alpha_K$,

1. Given parameters, we can easily compute probability that a point x^j belongs to class i :

$$p(y = k | x^j) \propto p(x^j | \mu_k, \Sigma_k) p(y = k)$$

E-step

Simply evaluate this, then normalize

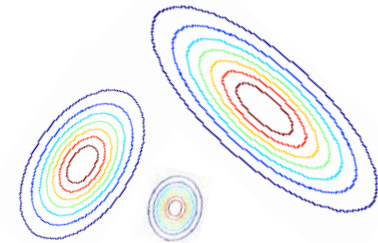
2. Given the probability of each point belonging to each class, re-estimate the $\mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_K, \alpha_1, \dots, \alpha_K$,

$$\mu_k = \frac{\sum_{j=1}^m P(y=k | x^j) x^j}{\sum_{j=1}^m P(y=k | x^j)} \quad \alpha_k = \frac{\sum_{j=1}^m P(y=k | x^j)}{m}$$

$$\sigma_{kl}^2 = \frac{\sum_{j=1}^m P(y=k | x^j) (x_l^j - \mu_{kl})^2}{\sum_{j=1}^m P(y=k | x^j)}$$

M-step

EM – General Gaussian



Start with an initial guess for $\mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_K, \alpha_1, \dots, \alpha_K$,

1. If we know the parameters, we can easily compute probability that a point x^j belongs to class k :

$$P(y = k|x^j) \propto p(x^j|\mu_k, \Sigma_k) p(y = k)$$

E-step

Simply evaluate this, then normalize

2. If we know *the* probability that each point belongs to each class, we can estimate the $\mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_K, \alpha_1, \dots, \alpha_K$,

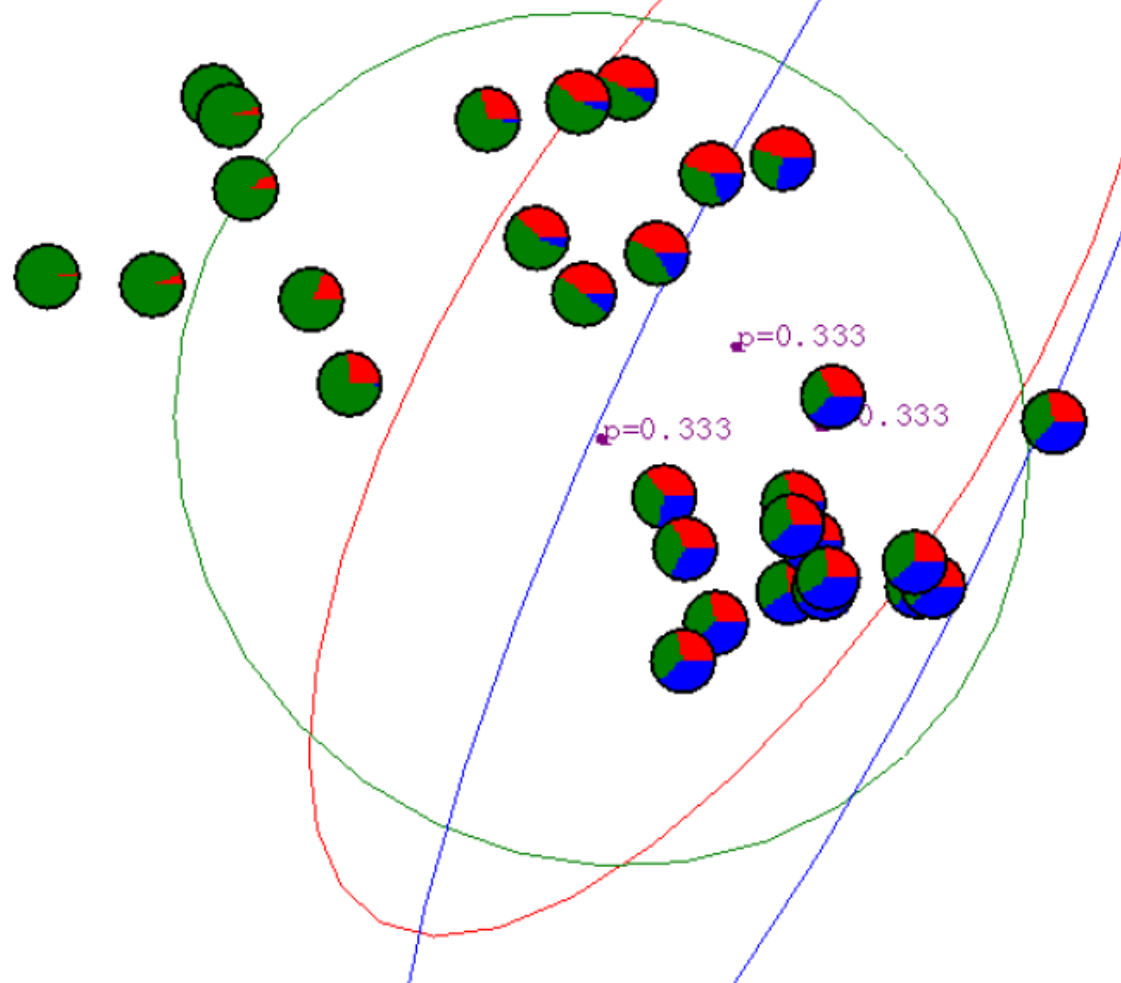
$$\mu_k = \frac{\sum_{j=1}^m P(y=k|x^j) x^j}{\sum_{j=1}^m P(y=k|x^j)}$$

$$\alpha_k = \frac{\sum_{j=1}^m P(y = k|x^j)}{m}$$

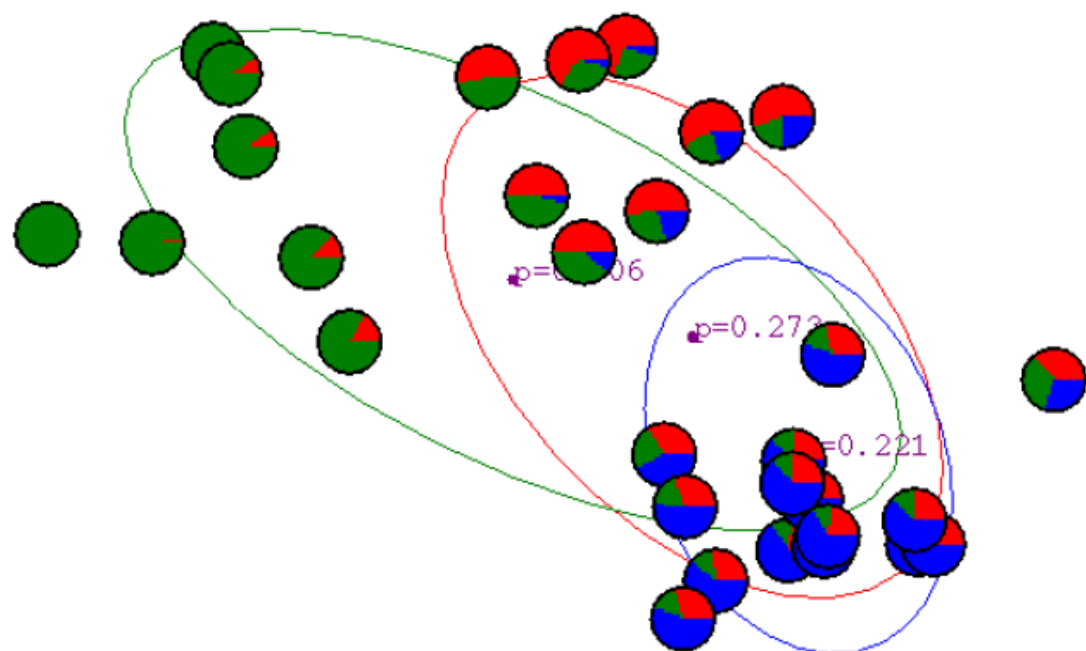
$$\Sigma_k = \frac{\sum_{j=1}^m P(y = k|x^j) (x^j - \mu_k)(x^j - \mu_k)^T}{\sum_{j=1}^m P(y = k|x^j)}$$

M-step

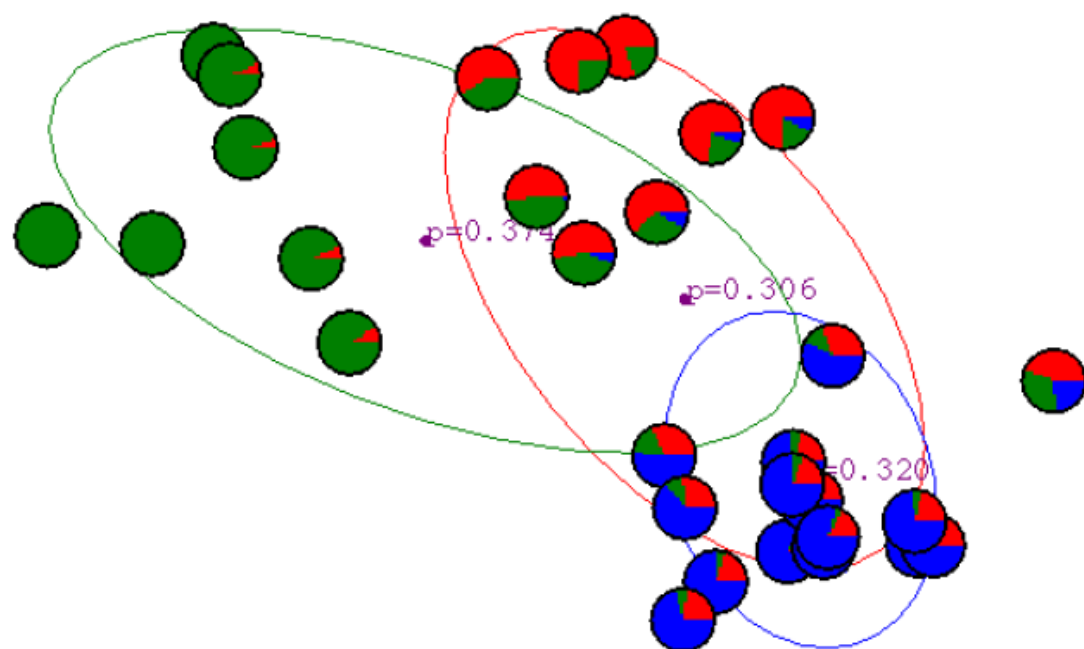
Gaussian Mixture Example: Start



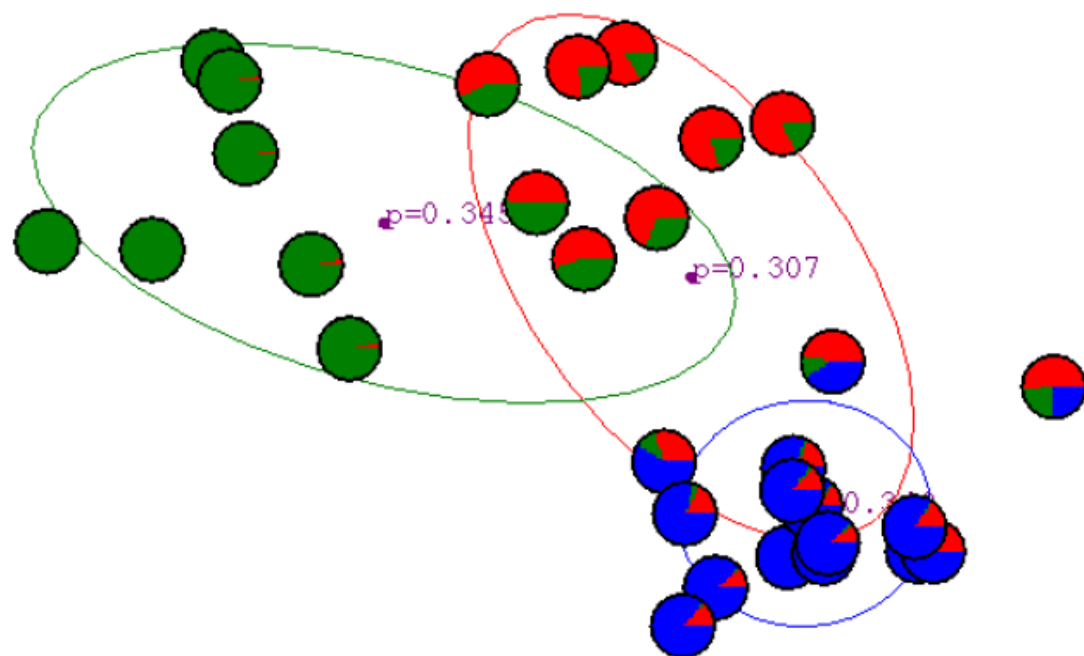
After first iteration



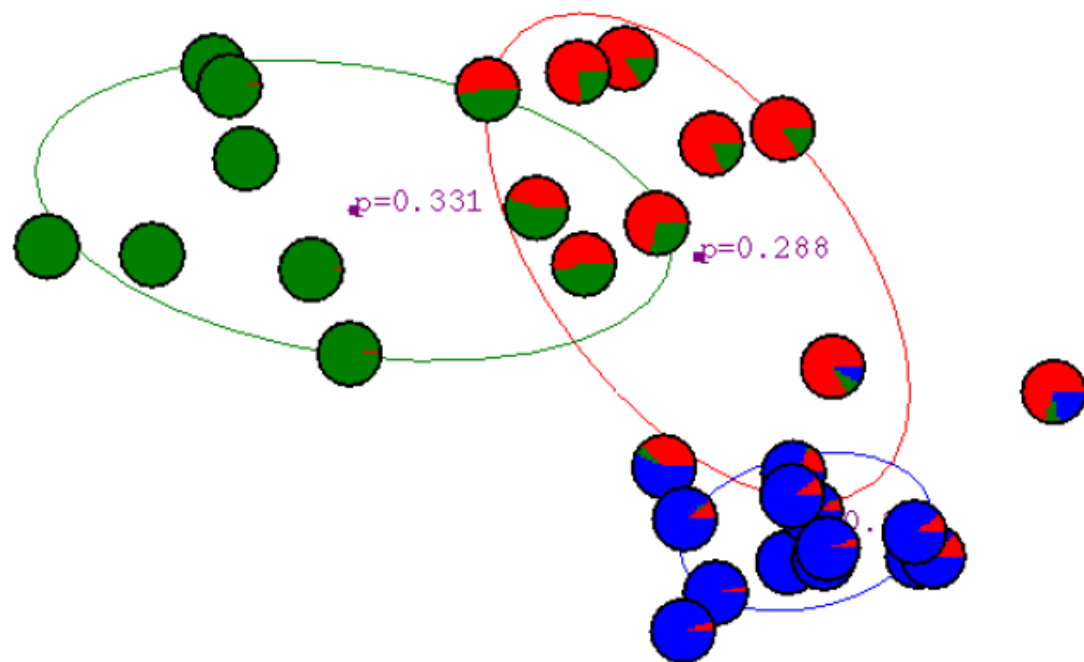
After 2nd iteration



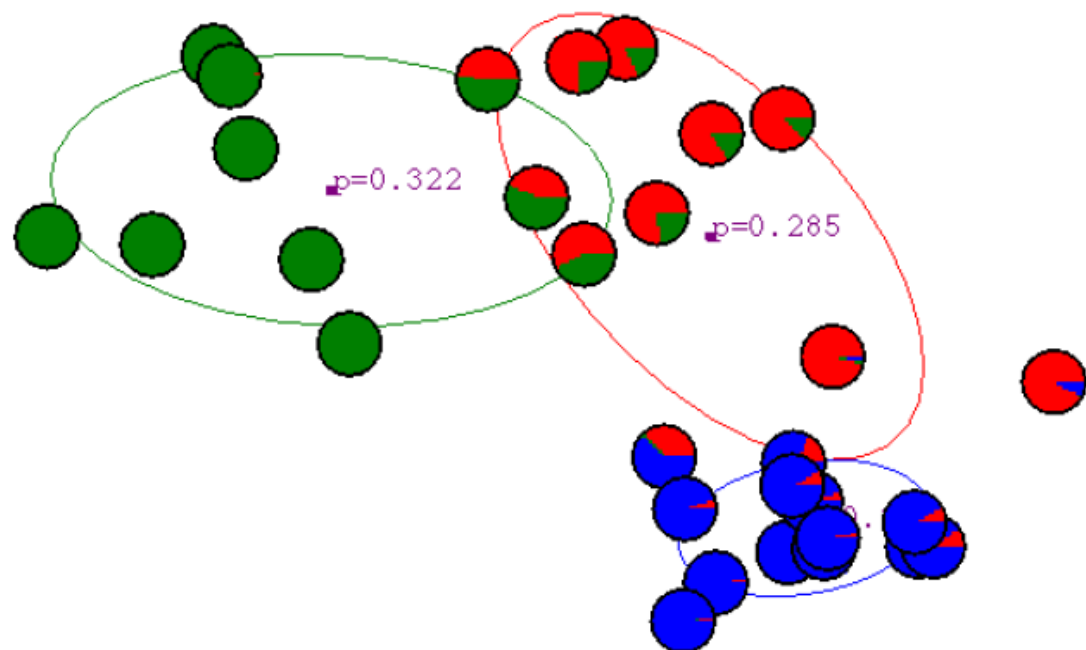
After 3rd iteration



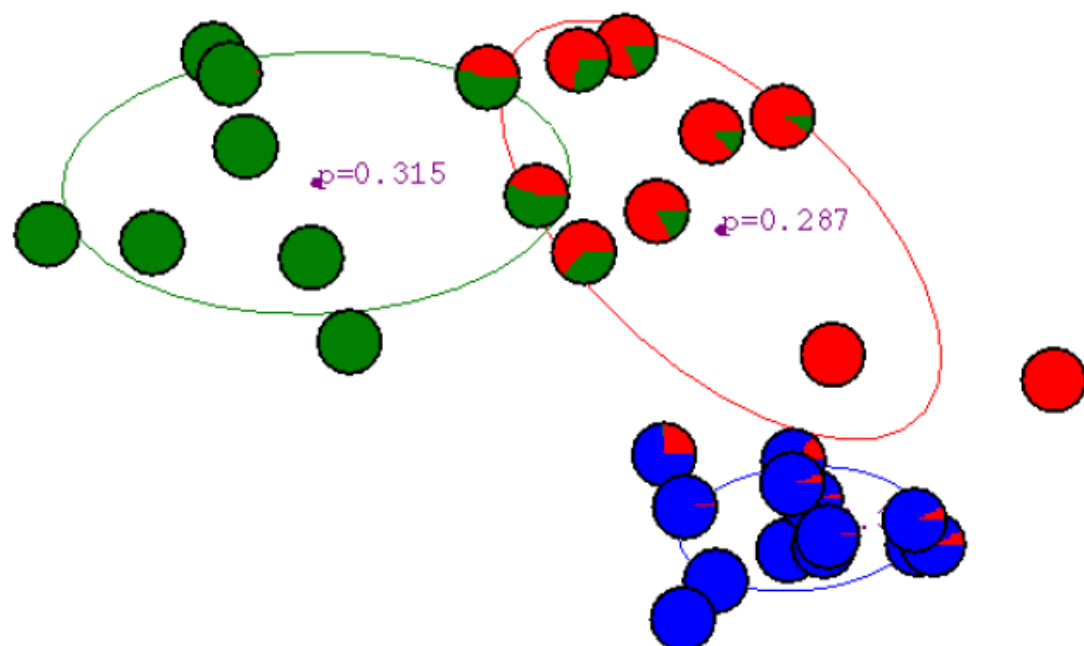
After 4th iteration



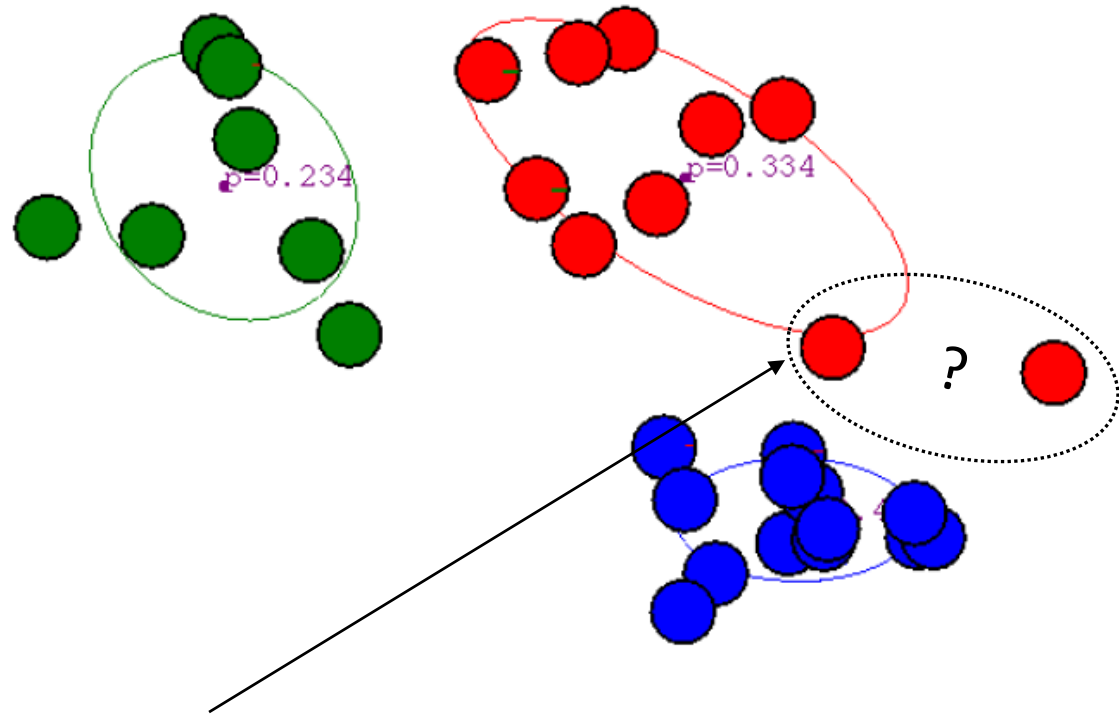
After 5th iteration



After 6th iteration



After 20th iteration



Q: Why are these two points red?

Behavior of EM

- It is guaranteed to converge
 - Convergence proof is based on the fact that the marginal likelihood $P(x; \Theta)$ must increase or remain same between iterations (not obvious)
 - In practice it may converge slowly, one can stop early if the change in log-likelihood is smaller than a threshold
- It converges to a local optimum
 - Multiple restart is recommended
 - Choosing the solution with the highest marginal likelihood