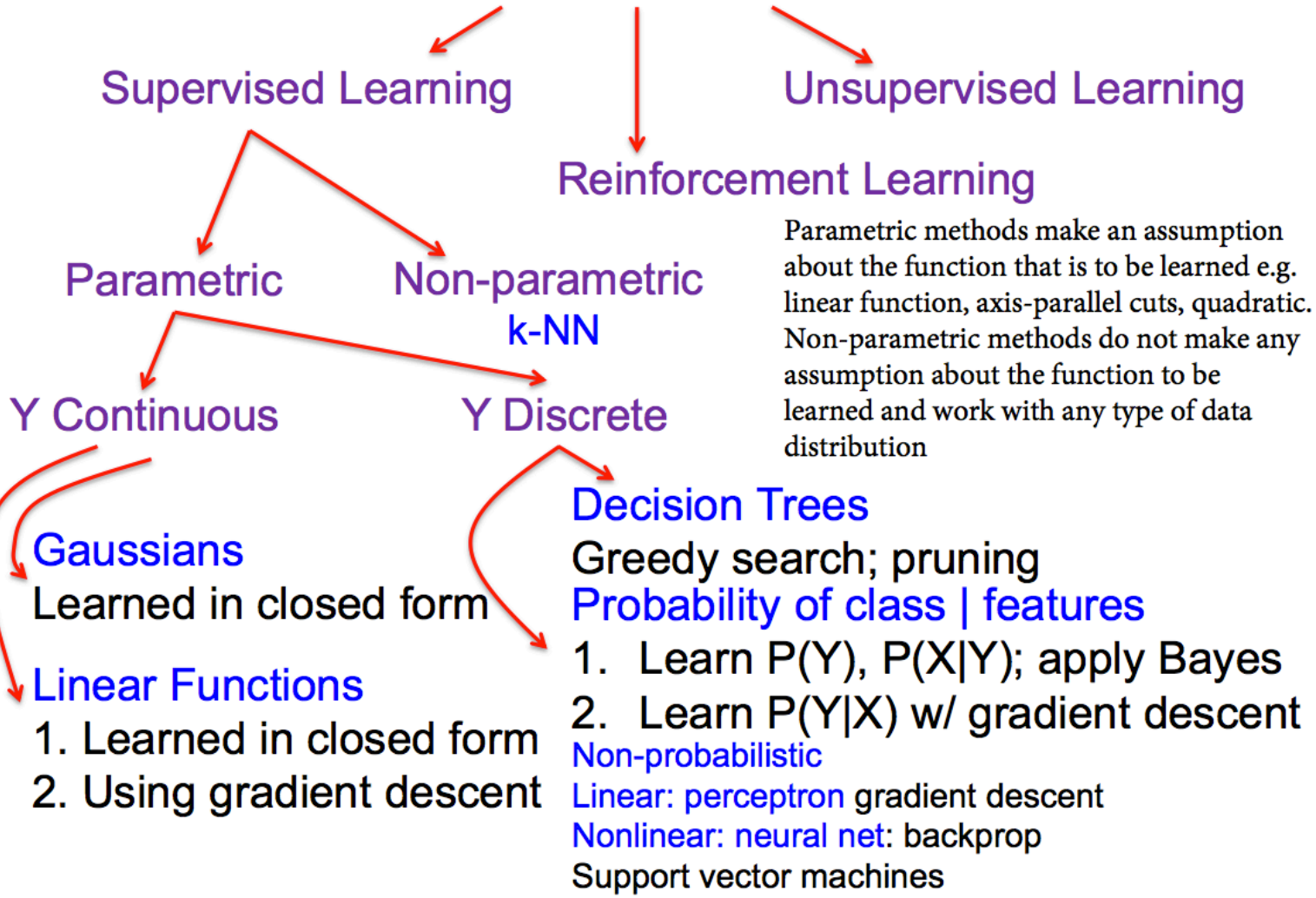
Unsupervised Learning

Types of Learning

- Supervised Learning:
  Training data is in the form: { $\mathbf{X}^{(i)}$, $Y^{(i)}$ }
  where $\mathbf{X}^{(i)}$ is the attributes vector and $Y^{(i)}$ is the class label.
  Aim: Learn the class separating function f: **X** -> Y, by proposing suitable hypothesis h(**X**)
- Unsupervised Learning
  Training data is in the form: { $\mathbf{X}^{(i)}$ } i.e. class labels are not provided.
  Aim: - Find natural grouping or clusters of data
  - Find natural patterns in data
  - Find out if there are any outliers or some data with unusual patterns.

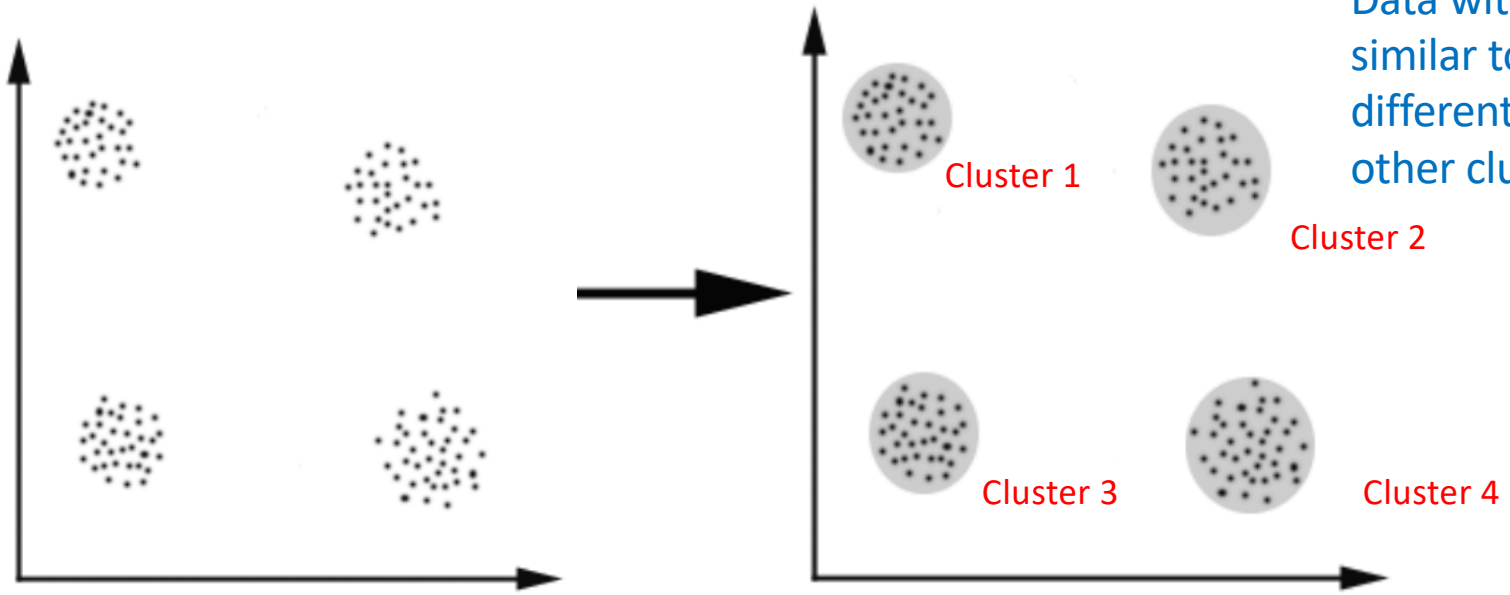
Machine Learning



Parametric methods make an assumption about the function that is to be learned e.g. linear function, axis-parallel cuts, quadratic. Non-parametric methods do not make any assumption about the function to be learned and work with any type of data distribution

Clustering

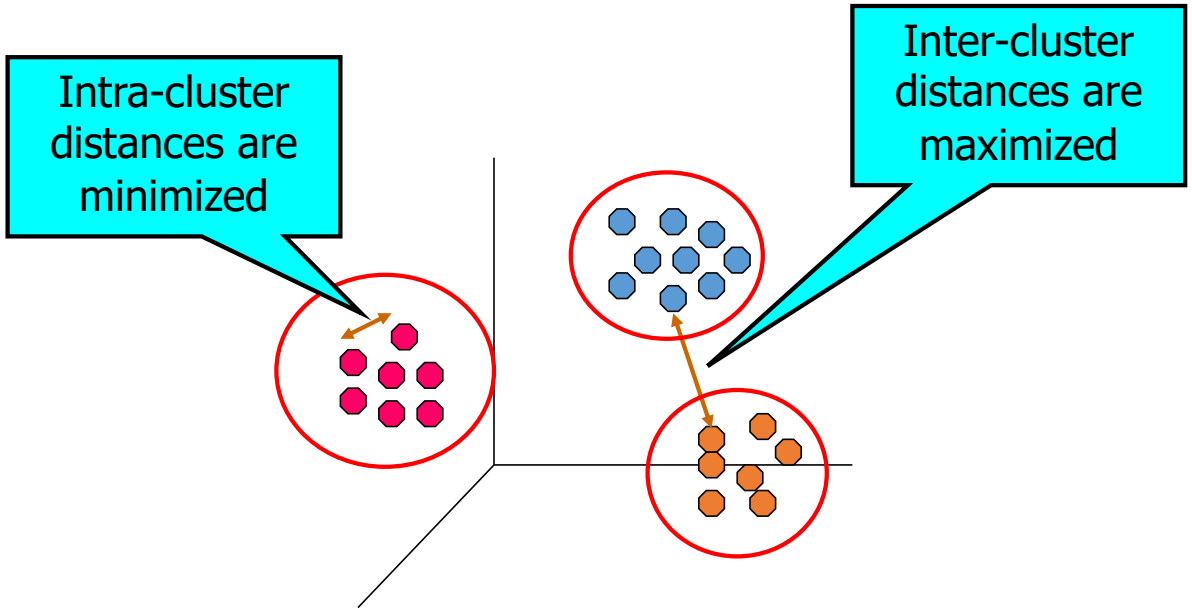
- Clustering is a technique for finding similarity groups in data, called **clusters**.
- Clustering groups data instances that are similar to (near) each other in one cluster and data instances that are very different (far away) from each other into different clusters.



Data within each cluster are similar to each other and different from instances in other clusters.

Cluster Analysis

- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



Notion of a Cluster can be Ambiguous

How many clusters?

Six Clusters

Two Clusters

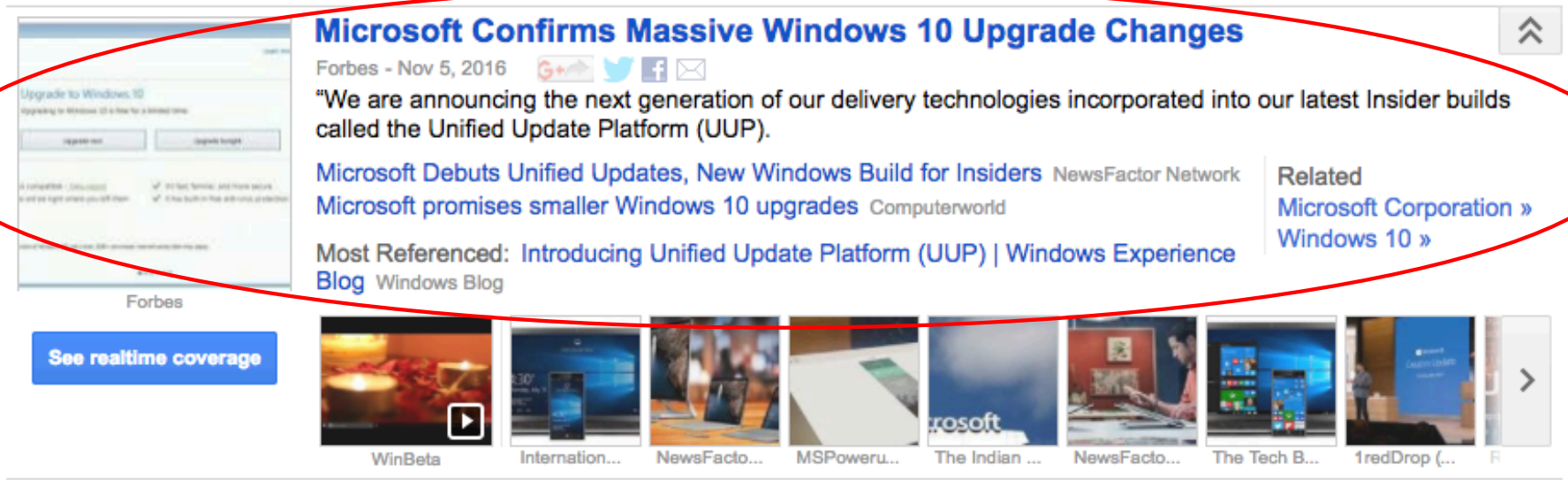
Four Clusters

Clustering

- Clustering statement:
  Given a dataset $D=\{X_1, X_2, , X_n\}$ of instances and an integer value representing the number of clusters k, the clustering problem refers to define a mapping f: D -> k, where each X_i is assigned to one cluster k_i such that $1<=j<=k$
- How is each cluster represented? By its mean or central value.
- How do you compare two instances? By a similarity (distance) metric that you define. E.g.
  - To compare instances in 2-D, you can use Euclidean distance
  - To compare people on a social network, you can propose a similarity metric.

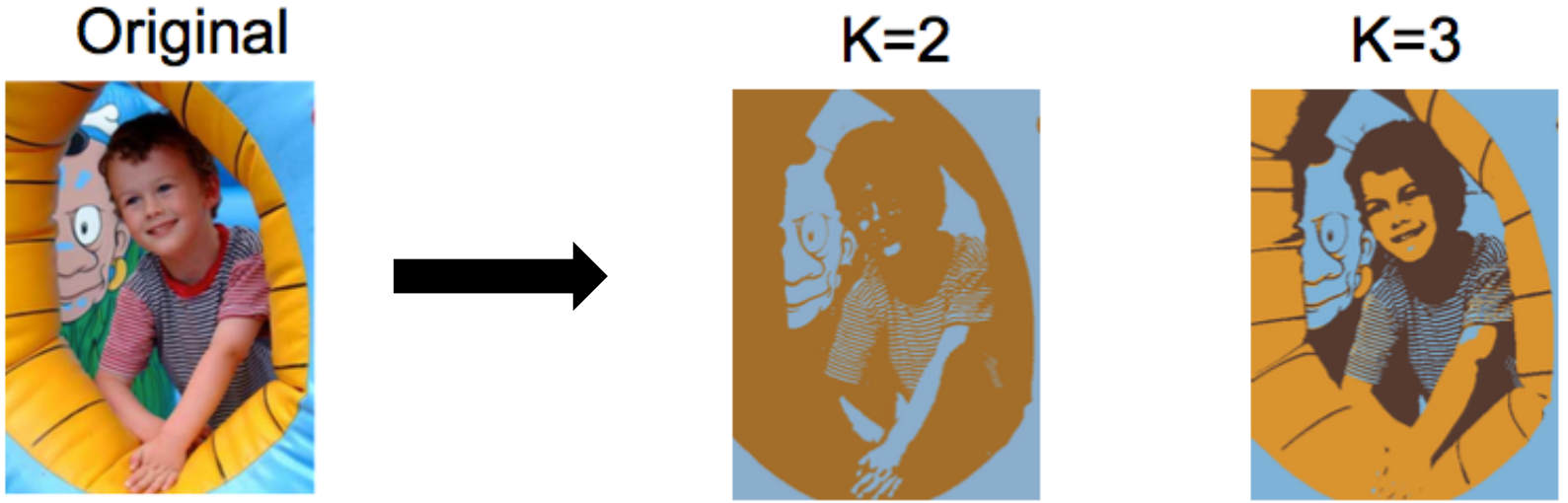
What is clustering for?

- Let us see some real-life examples
- Example 1: In marketing, segment customers according to their similarities to do targeted marketing.
- Example 2: Given a collection of text documents, we want to organize them according to their content similarities.
  e.g. Google News – similar news items are clustered together.



What is clustering for?

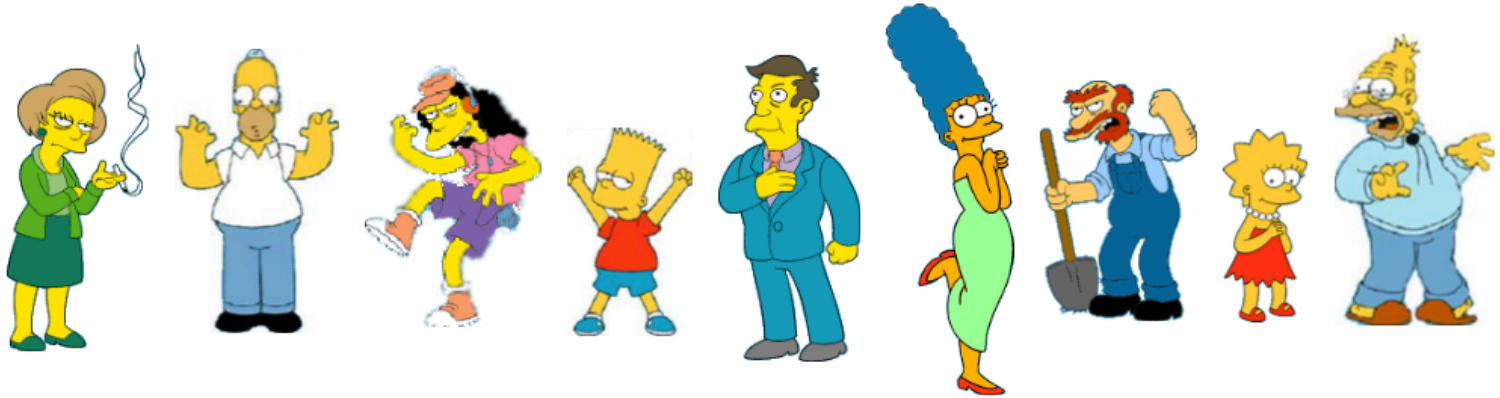
- Let us see some real-life examples
- Example 3: Find groups of related emails.
  This is different from spam/non-spam which is classification.
- Example 4: Finding clusters ("segments") in an image.



Clustering

- Clustering finds natural groupings of similar objects.
- To implement clustering, we need a distance measure between objects.
  e.g.
  Euclidean measure for 2-D
  Number of common friends in a social network
  RGB values of 2 pixels
  …
- Clustering is subjective i.e. it depends on your defined distance metric.

What is a natural grouping among these objects?



Clustering is subjective

Simpson's Family

School Employees

Females

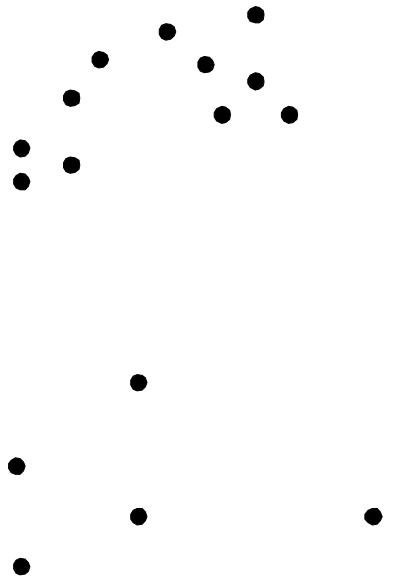
Males

Types of Clusterings

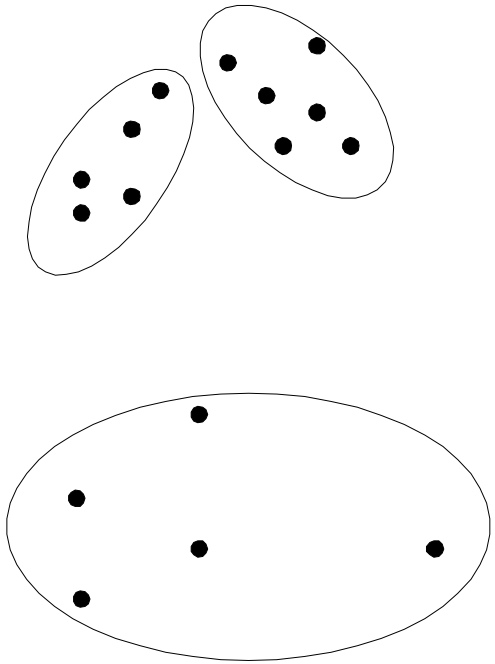
- A clustering is a set of clusters
- Important distinction between hierarchical and partitional sets of clusters
- Partitional Clustering
  - A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- Hierarchical clustering
  - A set of nested clusters organized as a hierarchical tree

Partitional Clustering

Note that each data point belongs to only one cluster.

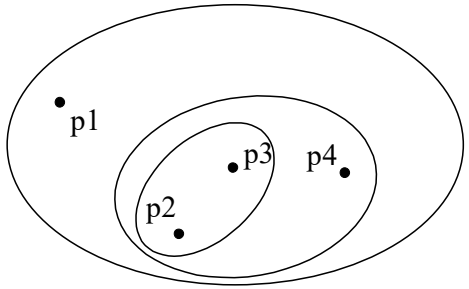


Original Points

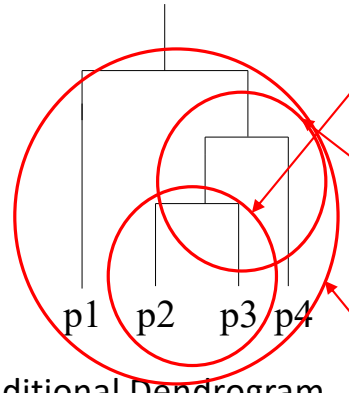


A Partitional Clustering

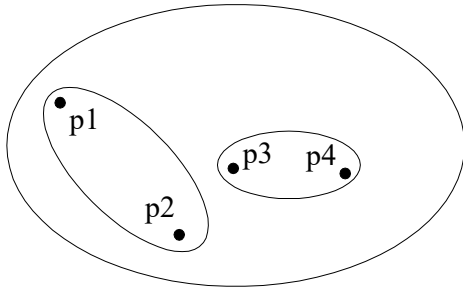
Hierarchical Clustering



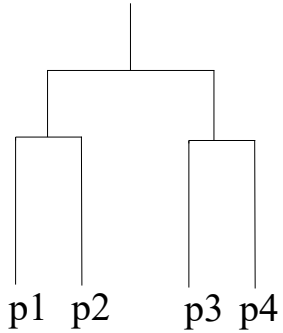
Traditional Hierarchical Clustering



Traditional Dendrogram



Non-traditional Hierarchical Clustering



Non-traditional Dendrogram

First put the 2 most similar items together in one cluster

Then combine the next two most similar items to get another cluster. Note: a cluster can be represented as a point

Each step results in a nested cluster leading to a final big overall cluster.

Partitional Clustering

- Let's start with partitional clustering.
- The most famous example of partitional clustering is the k-means algorithm.
- k-means seeks to partition the data into k clusters.
- Each cluster is identified by its mean (centroid).
- Let's look at the algorithm statement on the next slide.

K-means Clustering

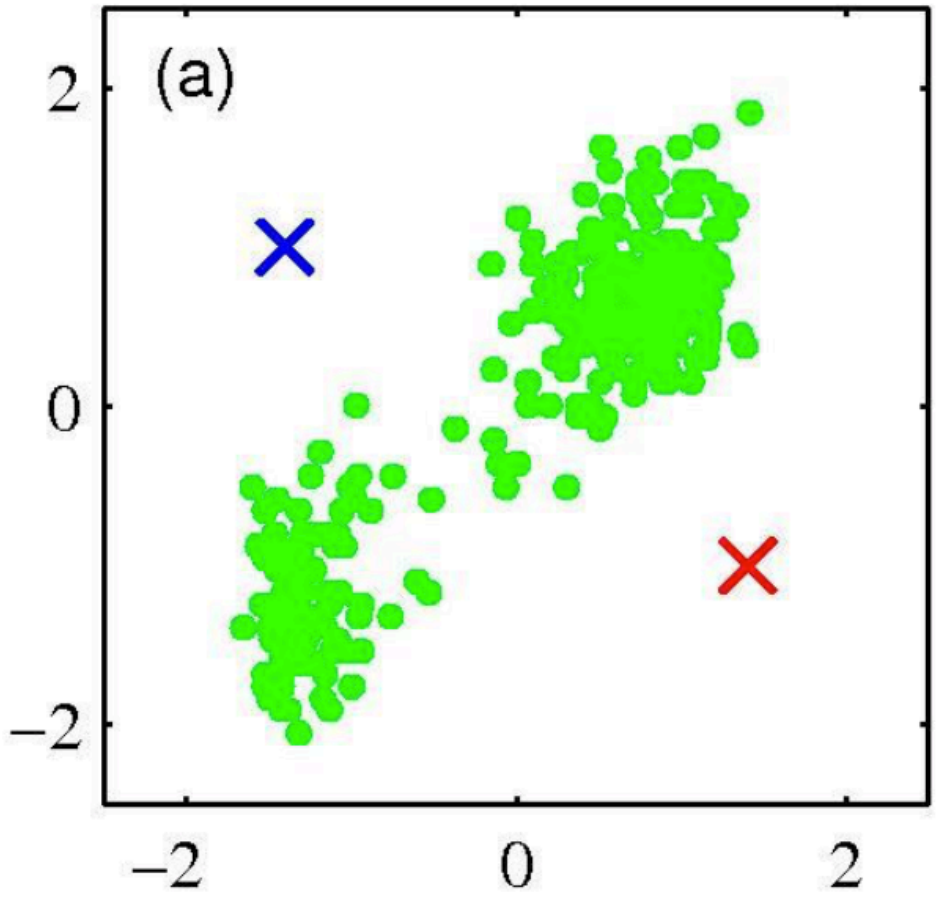
- Each cluster is associated with a centroid (center point)
- Each data item is assigned to the cluster with the closest centroid
- Number of clusters, K, must be specified
- The basic algorithm is very simple

Algorithm Statement

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

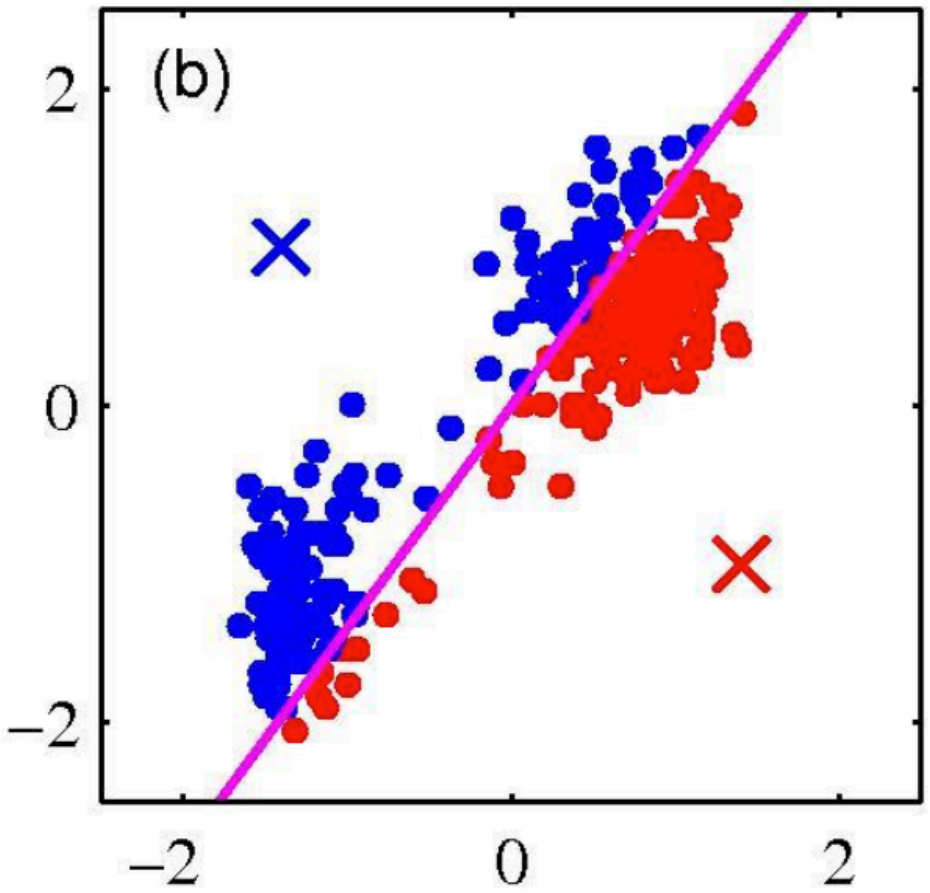
These K points don't have to be actual data points.

K-means clustering: Example



- Pick K random points as cluster centers (means)

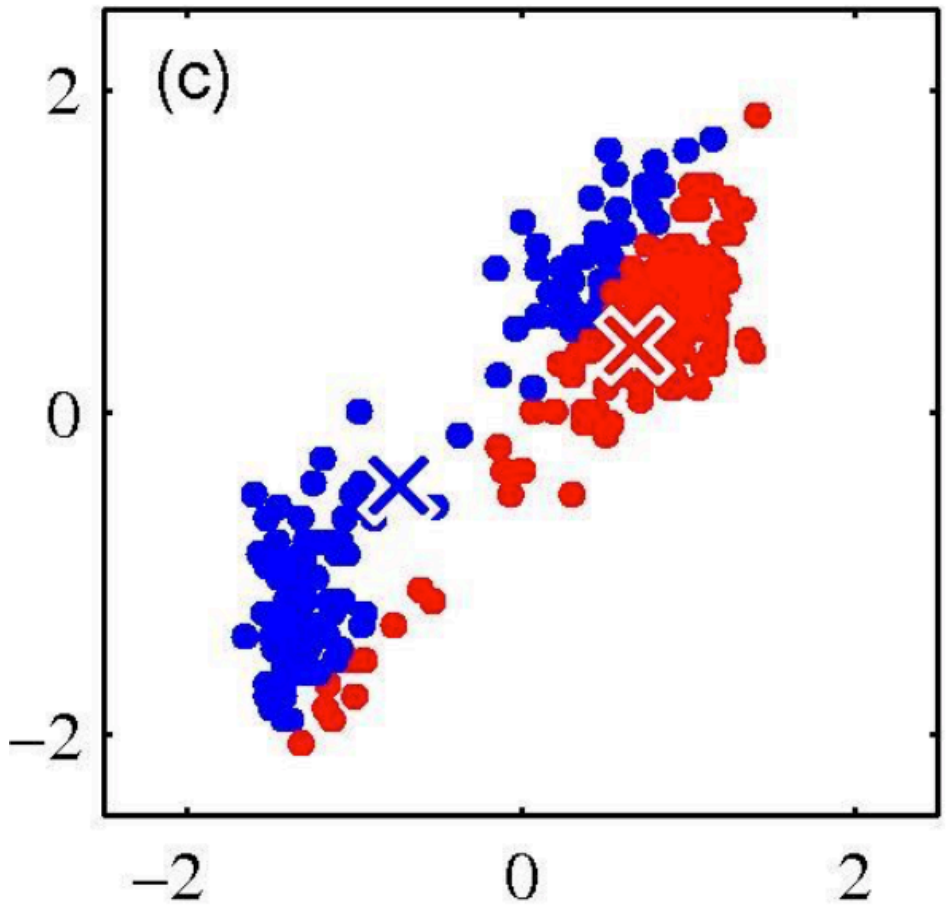
K-means clustering: Example



Iterative Step 1

- Assign data instances to closest cluster center

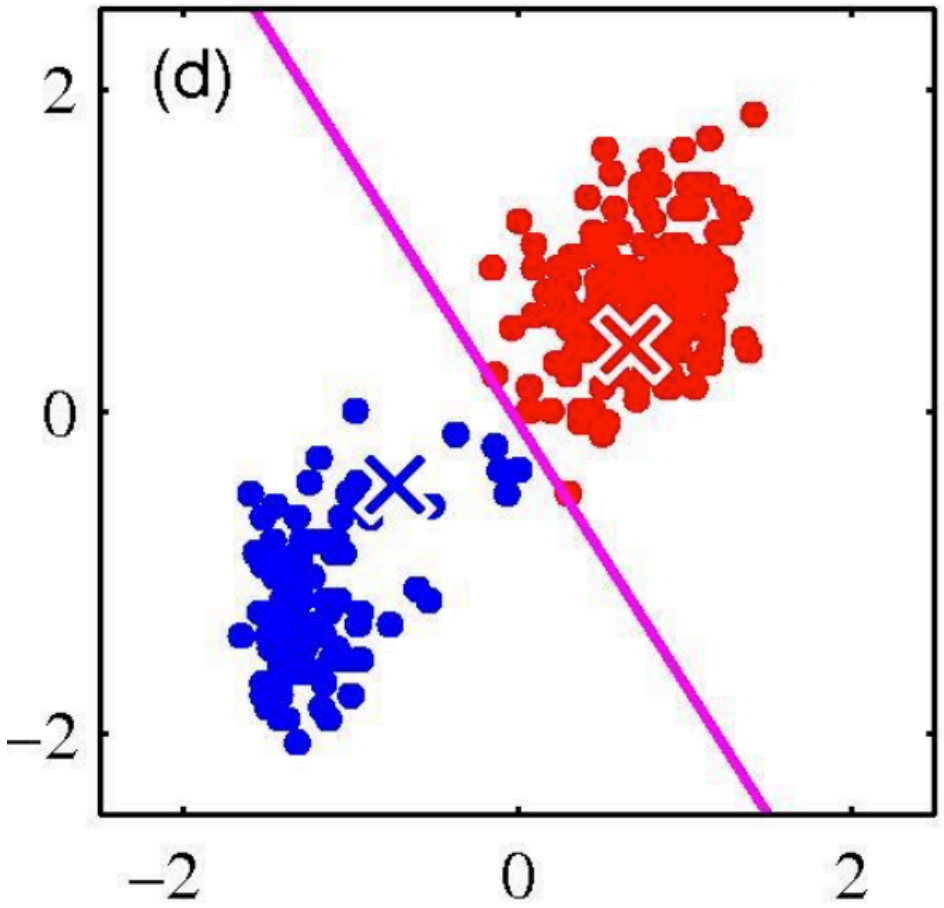
K-means clustering: Example



Iterative Step 2

- Change the cluster center to the average of the assigned points

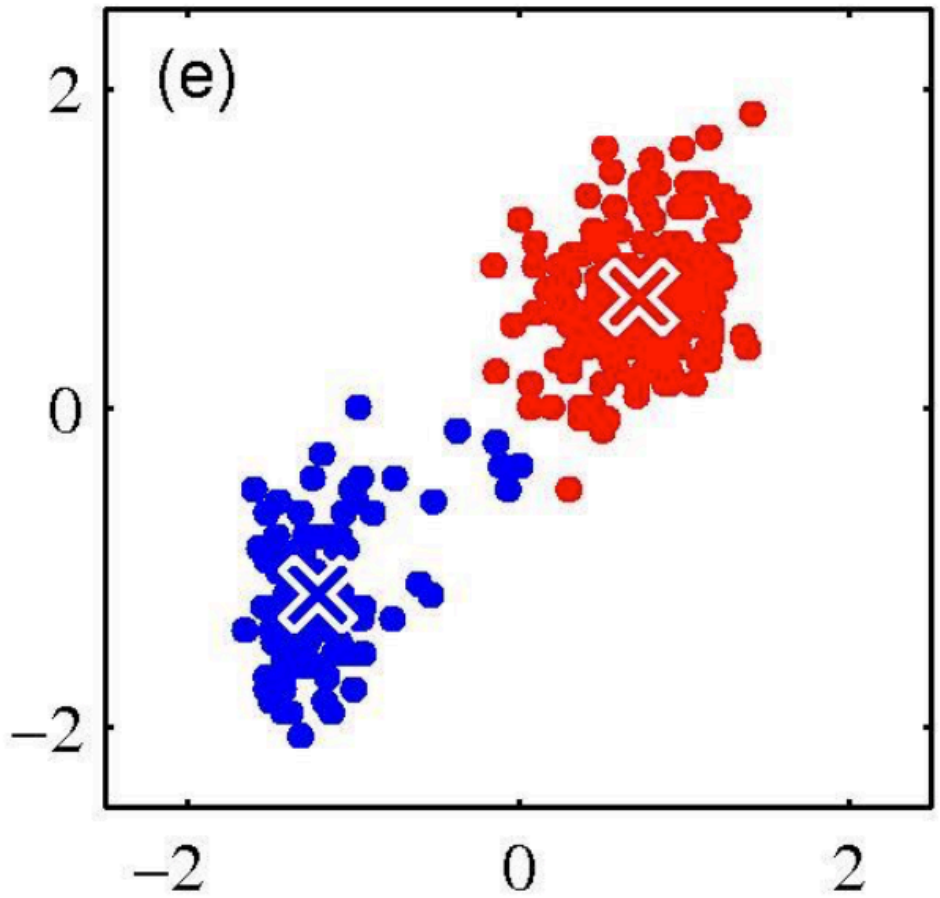
K-means clustering: Example



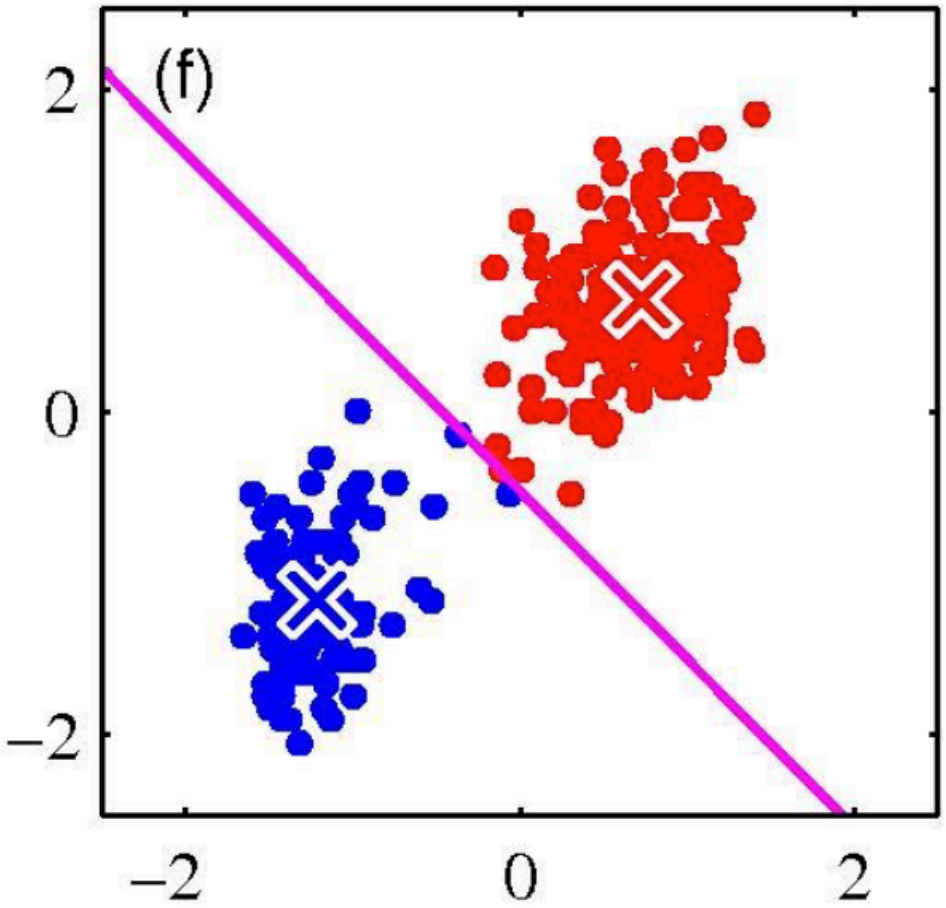
Re-assign points to closest clusters

K-means clustering: Example

Re-compute means

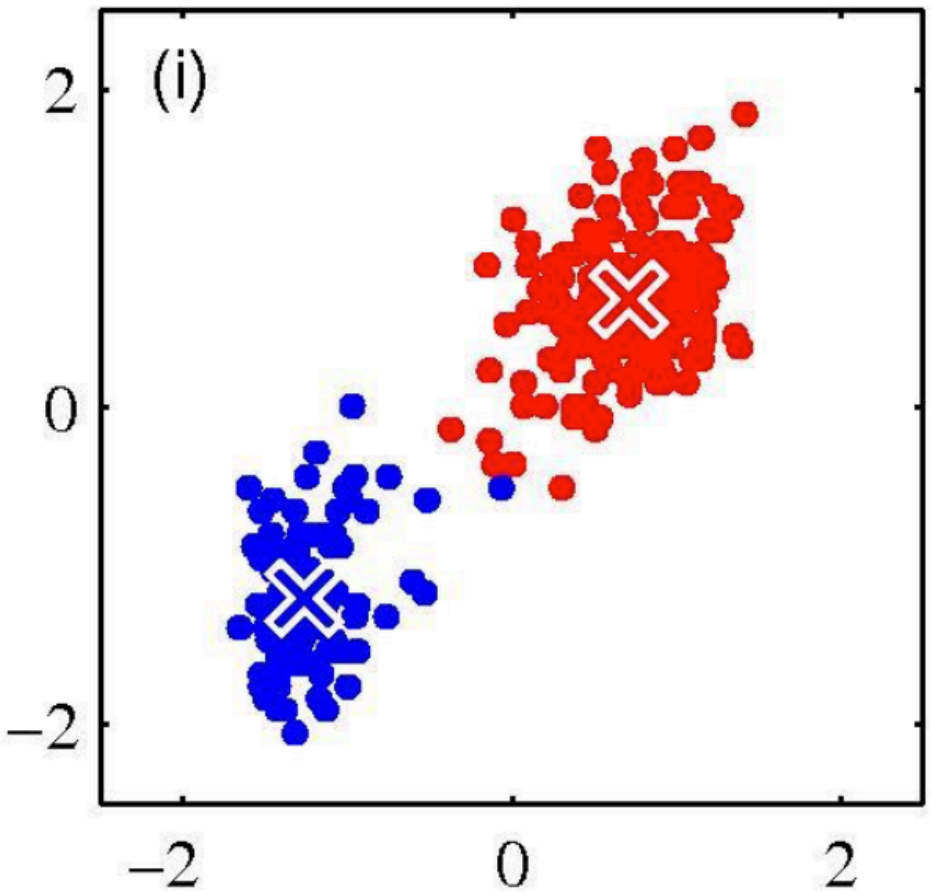


K-means clustering: Example



Re-assign points to closest clusters

K-means clustering: Example

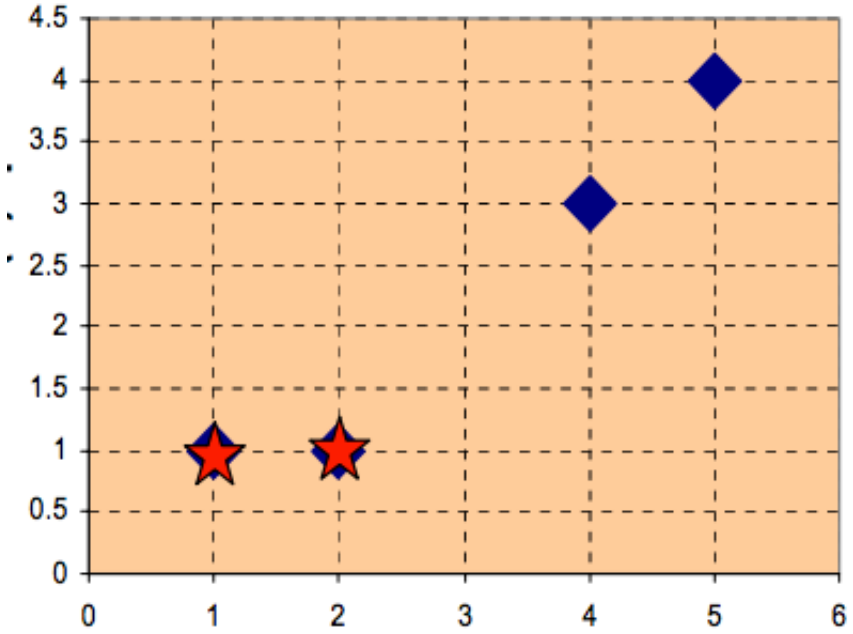


Final cluster assignment

K-means Example

Consider the following data points:
X1 = (1,1), X2 = (2,1), X3 = (4,3), X4 = (5,4)

Starting with the centers $\mu_1^{(0)} = (1, 1)$ and $\mu_2^{(0)} = (2, 1)$, run iterations of k-means on these data points till convergence. Use Manhattan distance.



1st iteration:

Assignment Step:

Point	Distance to μ1 (1,1)	Distance to μ2 (2,1)	Assignment
(1, 1)	0	1	μ1
(2, 1)	1	0	μ2
(4, 3)	5	4	μ2
(5, 4)	7	6	μ2

Update Step:

$$\mu_1^{(1)} = \left(\frac{1}{1}, \frac{1}{1}\right) = (1, 1)$$

$$\mu_2^{(1)} = \left(\frac{2+4+5}{3}, \frac{1+3+4}{3}\right) = (3.67, 2.67)$$

2nd iteration:

Assignment Step:

Point	Distance to μ1 (1,1)	Distance to μ2 (3.67,2.67)	Assignment
(1, 1)	0	4.34	μ1
(2, 1)	1	3.34	μ1
(4, 3)	5	0.66	μ2
(5, 4)	7	2.66	μ2

Update Step:

$$\mu_1^{(2)} = \left(\frac{1+2}{2}, \frac{1+1}{2}\right) = (1.5, 1.0)$$

$$\mu_2^{(2)} = \left(\frac{4+5}{2}, \frac{3+4}{2}\right) = (4.5, 3.5)$$

3rd iteration:

Assignment Step:

Point	Distance to μ1 (1.5,1.0)	Distance to μ2 (4.5,3.5)	Assignment
(1, 1)	0.5	6.0	μ1
(2, 1)	0.5	4.0	μ1
(4, 3)	4.5	1.0	μ2
(5, 4)	6.5	1.0	μ2

Update Step:

$$\mu_1^{(3)} = \left(\frac{1+2}{2}, \frac{1+1}{2}\right) = (1.5, 1.0)$$

$$\mu_2^{(3)} = \left(\frac{4+5}{2}, \frac{3+4}{2}\right) = (4.5, 3.5)$$

No change in means, so the algorithm terminates

K-means Algorithm

- The objective criterion that the algorithm seeks to optimize is:

  Given n data points, find the assignment of each point x_i to one of K clusters, such that the objective function below is minimized.

$$J(\boldsymbol{x}, \boldsymbol{\mu}) = \sum_{j=1}^{K} \sum_{x_i \in C_j} \left\| x_i - \mu_j \right\|^2$$

  where μ_j represents the centroid of cluster C_j

- So, there are 2 things that you need to optimize – assignment of data to clusters and centroid of clusters.

K-means Algorithm

- In iterative step 1, we keep the means fixed and re-compute assignment.
- In iterative step 2, we keep assignment fixed and re-compute means

K-Means Algorithm:

Initialize $\mu_1 \cdots \mu_K$ randomly

Repeat until convergence:

1. Assign each point x_i to the cluster with the closest mean μ_j
2. Calculate the new mean for each cluster

$$\mu_j \leftarrow \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

K-means Algorithm

- The optimization is a function of 3 parameters – points, assignments, and means.
  $\text{Optimization function } (\phi) = \phi(\{x_i\}, \{a_{ik}\}, \{c_k\}) = \min J(\boldsymbol{x}, \boldsymbol{\mu})$

  $\{x_i\}$ is the set of points
  $\{a_{ik}\}$ is the point-cluster assignment
  $\{c_k\}$ is the means of the clusters

- We want to find the $\boldsymbol{\phi}$ that minimizes the total distance J.

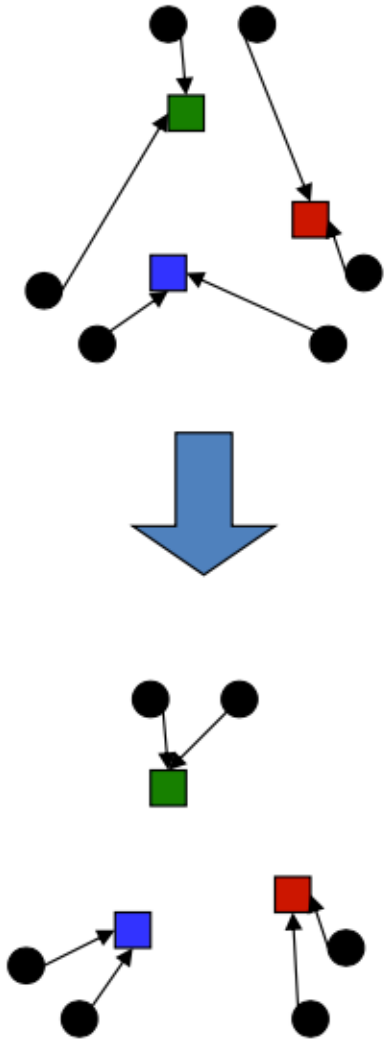
Why k-means works

- Step 1: Re-assignment Step
  Each point gets re-assigned to closest center.

- What happens to the total distance?

- It has to go down as points move closer to center.

- So, we have reduced *J* in this step.



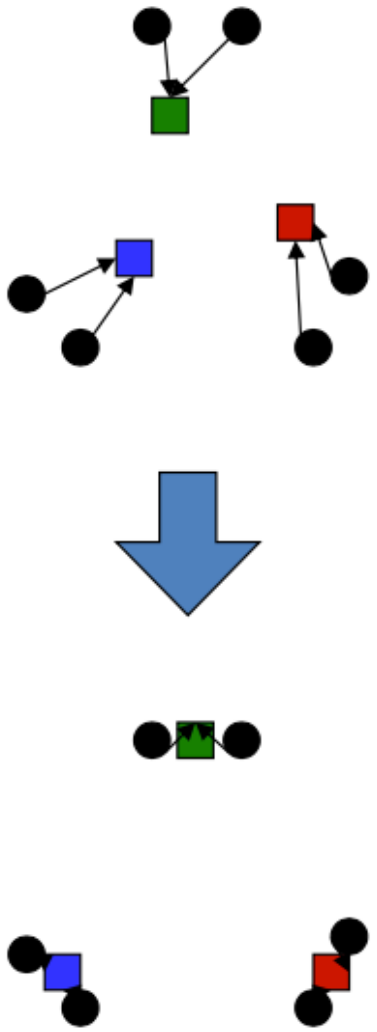
Why k-means works

- Step 2: Re-compute means step
  Means of each cluster gets re-computed.

- What happens to the total distance?

- It has to go down as center moves closer to assigned points

- So, we have reduced *J* in this step.



K-means

- Is the algorithm deterministic i.e. do different runs take same number of iterations and exactly the same clustering?
- No, it depends on initialization.

- How do you find best value of k?
- It's a parameter. Like other ML algorithms, you need to experiment to find best value.

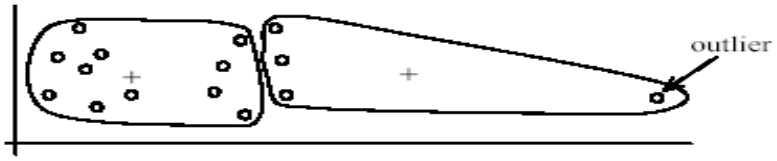
Strengths of k-means

- Strengths:
  - Simple: easy to understand and to implement
  - Efficient: Time complexity: $O(tkn)$,
    where n is the number of data points,
    k is the number of clusters, and
    t is the number of iterations.
  - Since both k and t are small. k-means is considered a linear algorithm in terms of n .
- K-means is the most popular clustering algorithm.
- Note that: it terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity.

Weaknesses of k-means

- The algorithm is only applicable if the mean is defined.
  - For categorical data, *k*-mode - the centroid is represented by most frequent values.
- The user needs to specify *k*.
- The algorithm is sensitive to **outliers**
  - Outliers are data points that are very far away from other data points.
  - Outliers could be errors in the data recording or some special data points with very different values.

Weaknesses of k-means: Problems with outliers

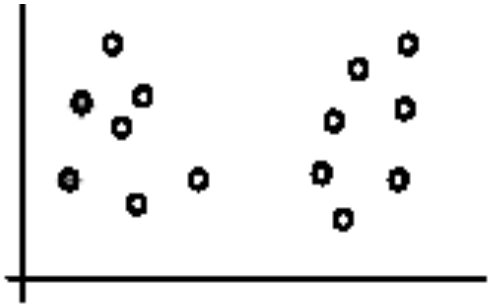


(A): Undesirable clusters

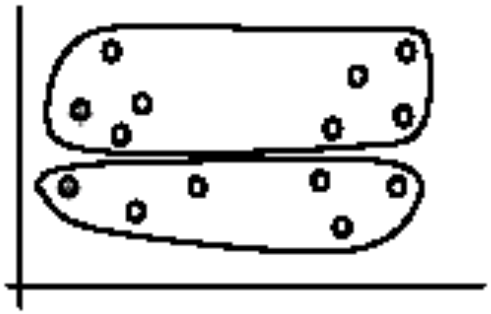
(B): Ideal clusters

Weaknesses of k-means (cont …)

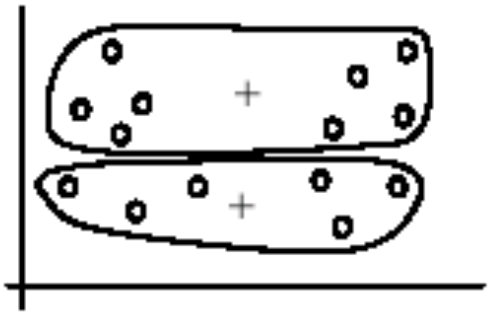
- The algorithm is sensitive to initial seeds.



(A). Random selection of seeds (centroids)



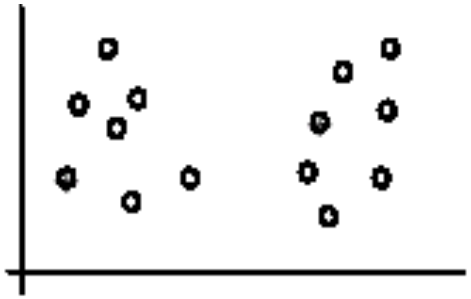
(B). Iteration 1



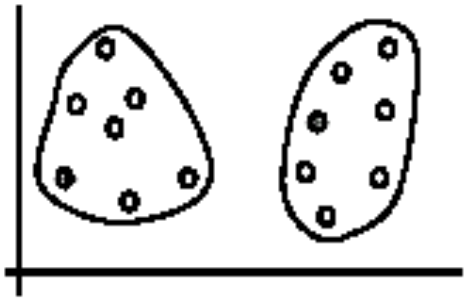
(C). Iteration 2

Weaknesses of k-means (cont …)

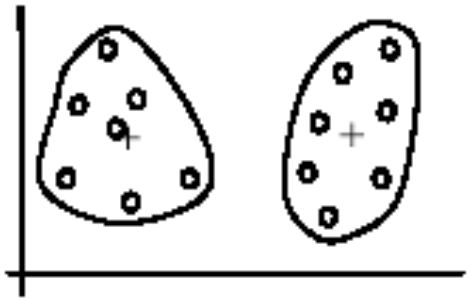
- If we use different seeds: good results



(A). Random selection of k seeds (centroids)



(B). Iteration 1

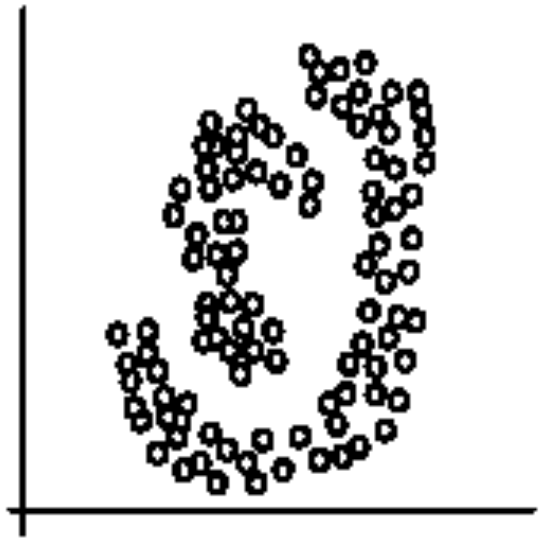


(C). Iteration 2

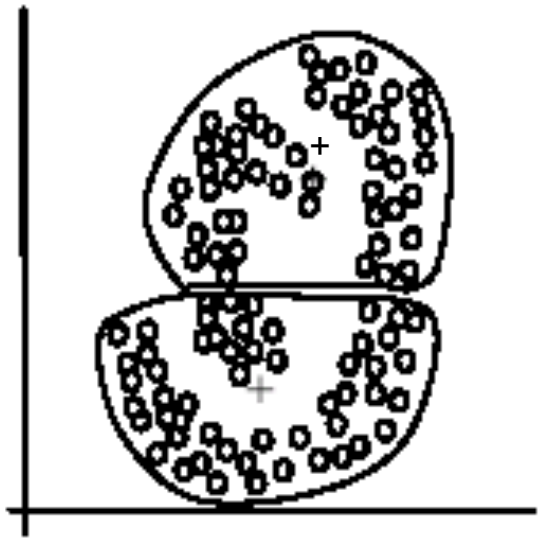
Weaknesses of k-means (cont …)

- The *k*-means algorithm is not suitable for discovering clusters that are not spherical shapes.

Source: http://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf



(A): Two natural clusters



(B): *k*-means clusters

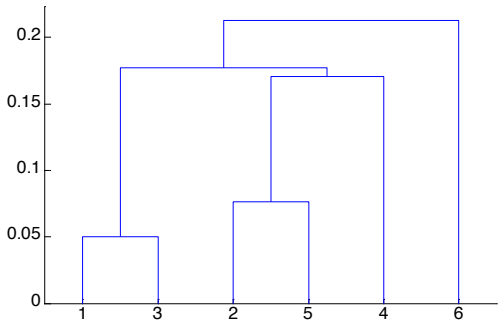
K-means summary

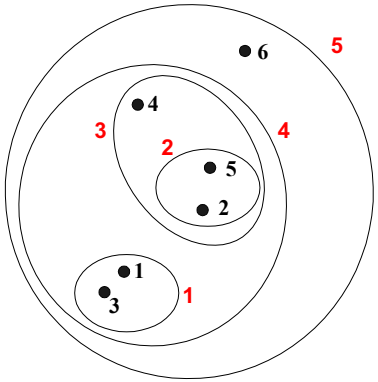
- Despite weaknesses, *k*-means is still the most popular algorithm due to its simplicity, efficiency and
  - other clustering algorithms have their own lists of weaknesses.
- No clear evidence that any other clustering algorithm performs better in general
  - although they may be more suitable for some specific types of data or applications.
- Comparing different clustering algorithms is a difficult task. No one knows the correct clusters

Hierarchical Clustering

Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges or splits





Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by 'cutting' the dendogram at the proper level

- They may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, …)

Hierarchical Clustering

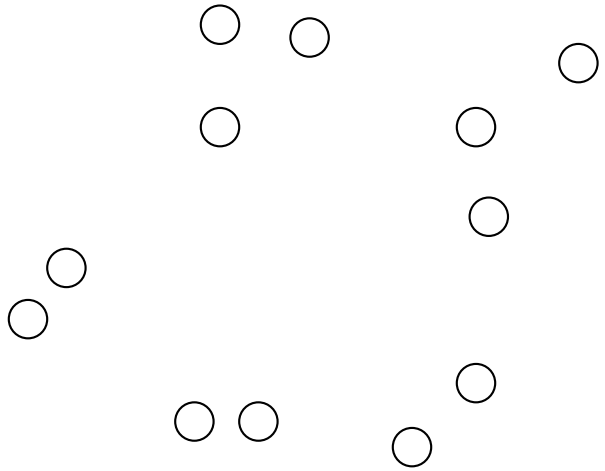
- Two main types of hierarchical clustering
  - Agglomerative:
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
  - Divisive:
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are k clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time

Agglomerative Clustering Algorithm

- More popular hierarchical clustering technique
- Basic algorithm is straightforward
  1. Compute the proximity matrix
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the proximity matrix
  6. **Until** only a single cluster remains
- Key operation is the computation of the proximity of two clusters
  - Different approaches to defining the distance between clusters distinguish the different algorithms

Starting Situation

- Start with clusters of individual points and a proximity matrix

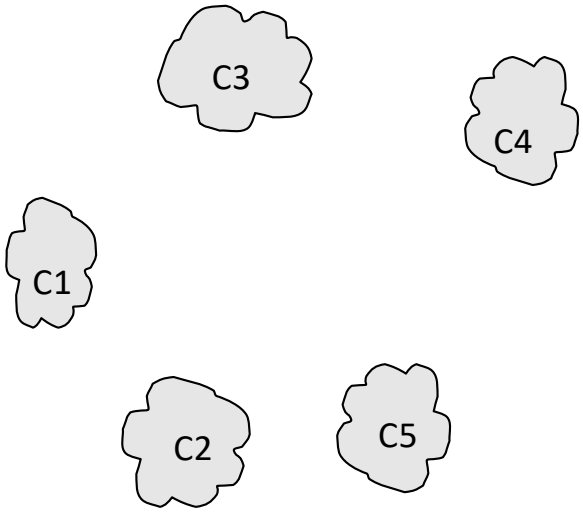


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

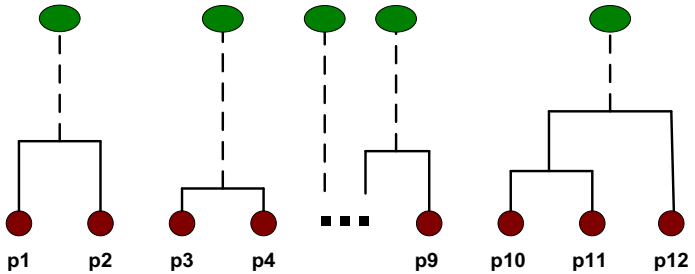
Intermediate Situation

- After some merging steps, we have some clusters



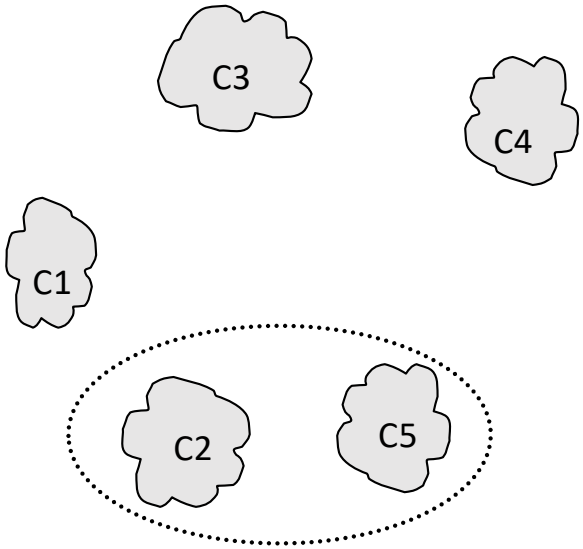
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



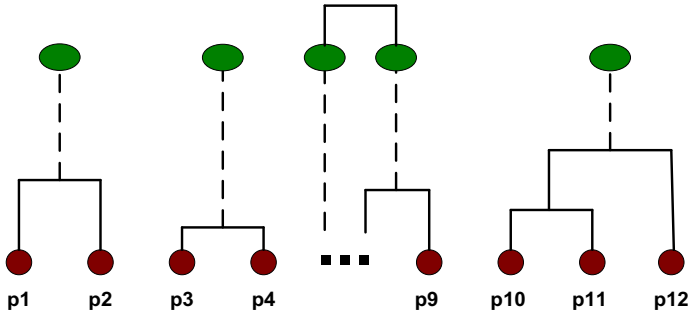
Intermediate Situation

- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.



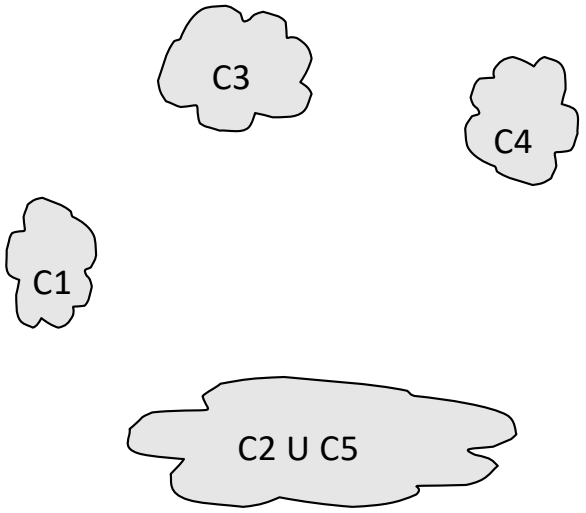
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



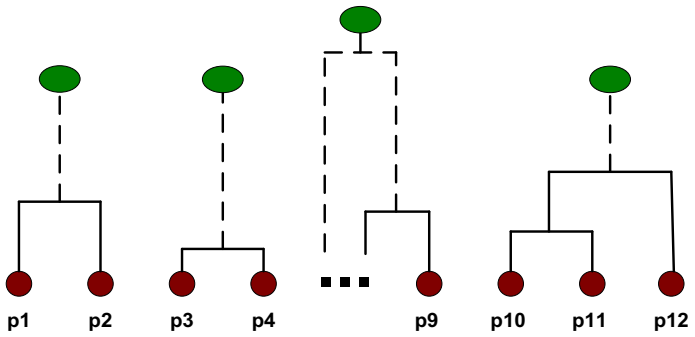
After Merging

- The question is “How do we update the proximity matrix?”

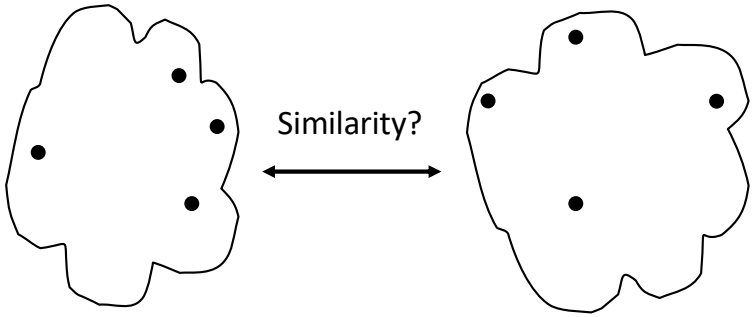


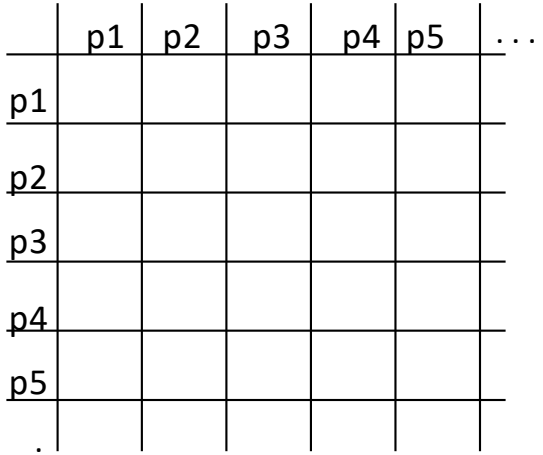
	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



How to Define Inter-Cluster Similarity

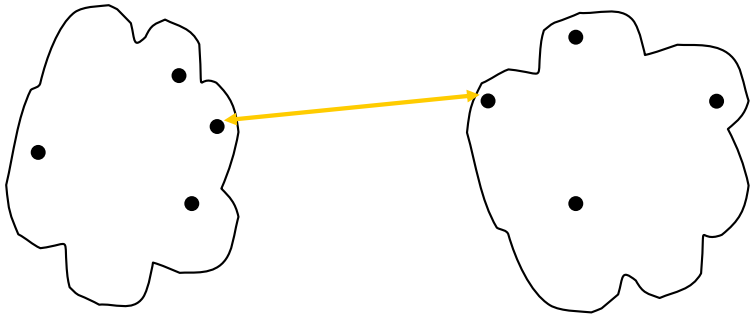


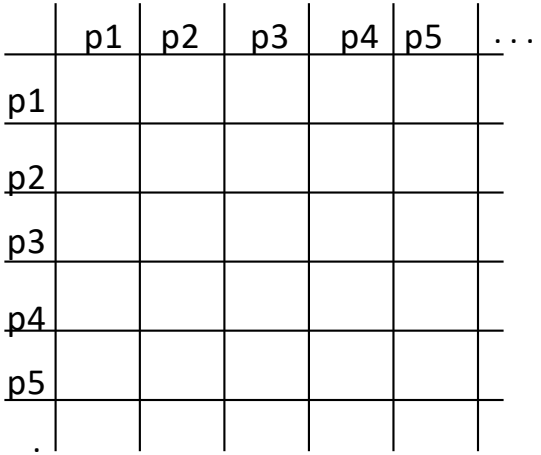


Proximity Matrix

- MIN (Single Link)
- MAX (Complete Link)
- Group Average (Average Link)
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

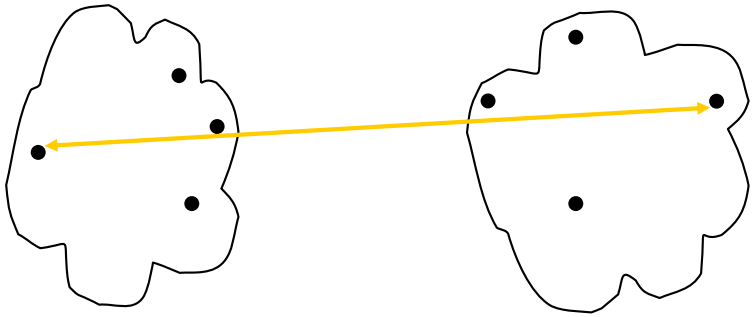


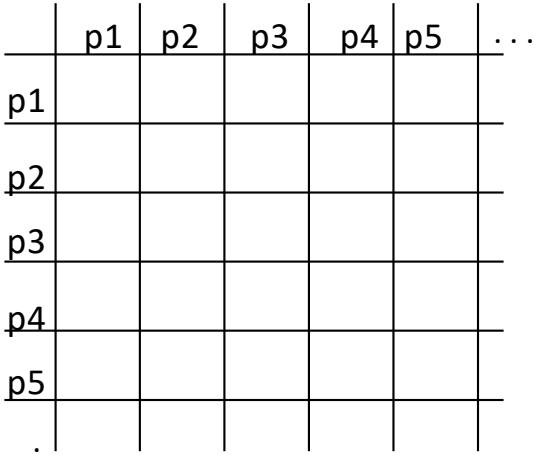


Proximity Matrix

- MIN (Single Link)
- MAX (Complete Link)
- Group Average (Average Link)
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

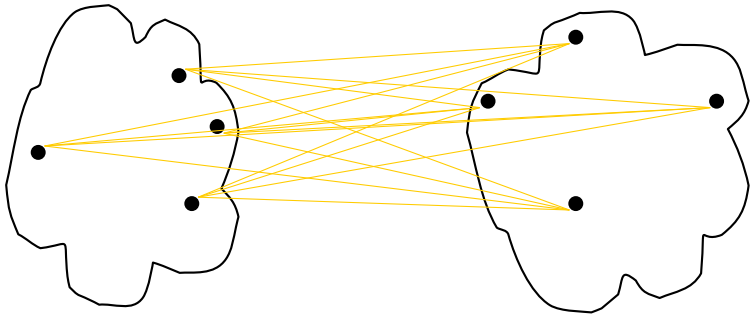


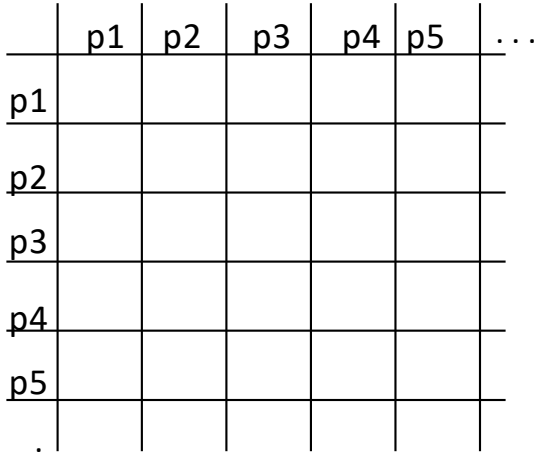


Proximity Matrix

- MIN (Single Link)
- MAX (Complete Link)
- Group Average (Average Link)
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

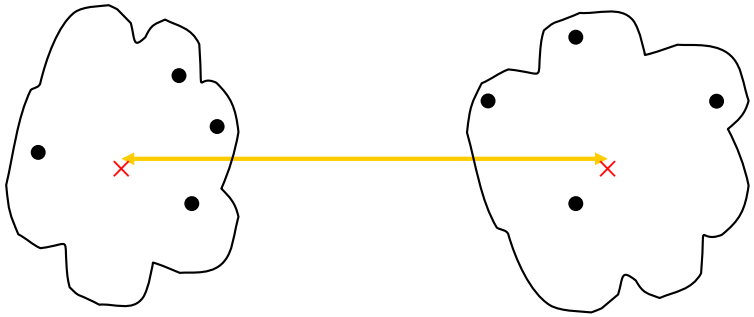


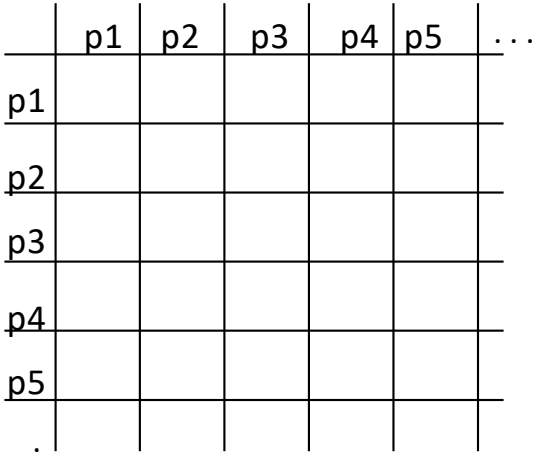


Proximity Matrix

- MIN (Single Link)
- MAX (Complete Link)
- Group Average (Average Link)
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity



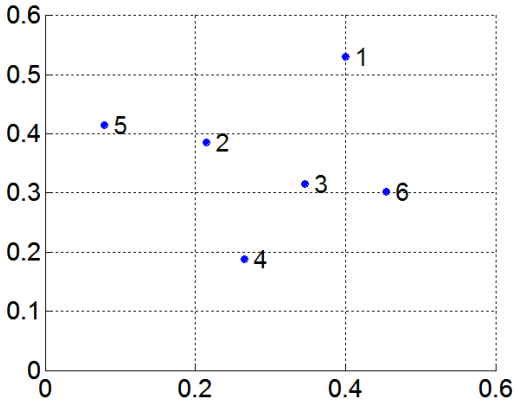


Proximity Matrix

- MIN (Single Link)
- MAX (Complete Link)
- Group Average (Average Link)
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

MIN or Single Link

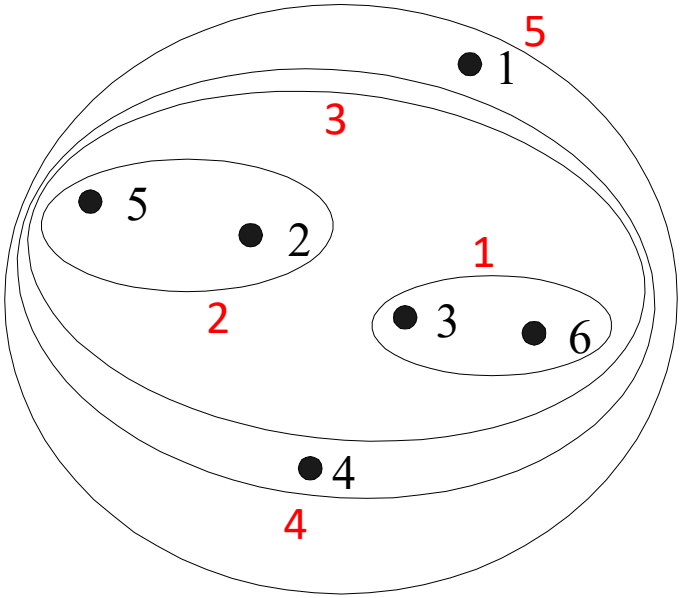
- Proximity of two clusters is based on the two closest points in the different clusters
  - Determined by one pair of points, i.e., by one link in the proximity graph
- Example:



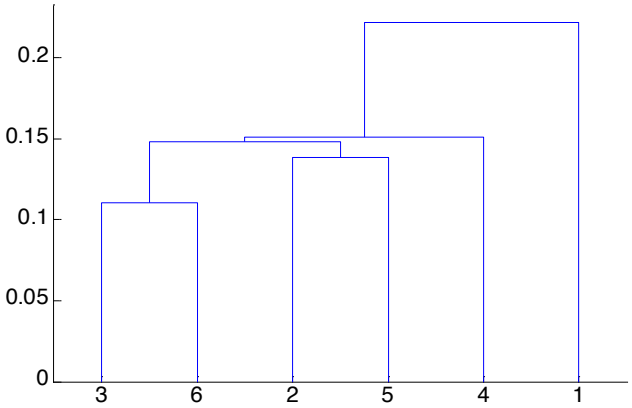
Distance Matrix:

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Hierarchical Clustering: MIN



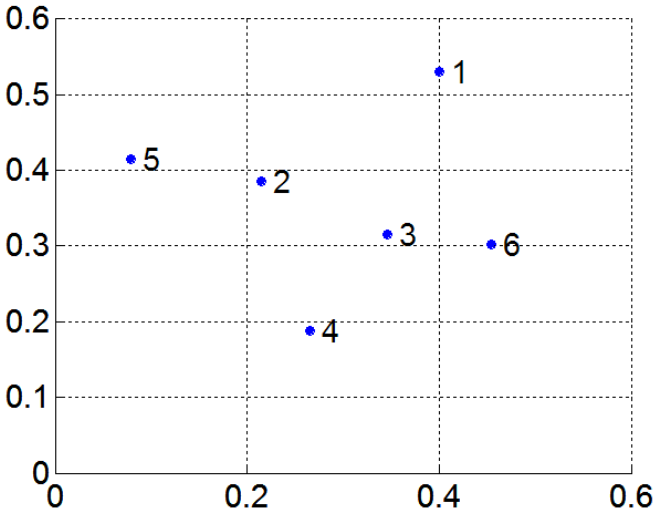
Nested Clusters



Dendrogram

MAX or Complete Linkage

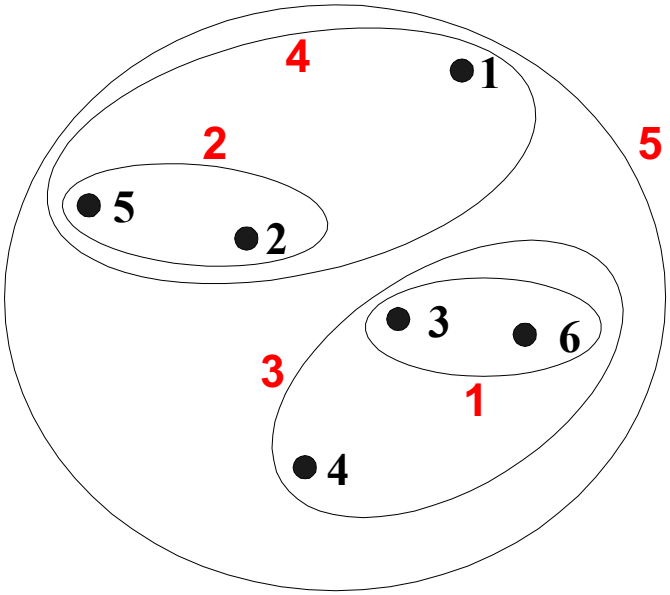
- Proximity of two clusters is based on the two most distant points in the different clusters
  - Determined by all pairs of points in the two clusters



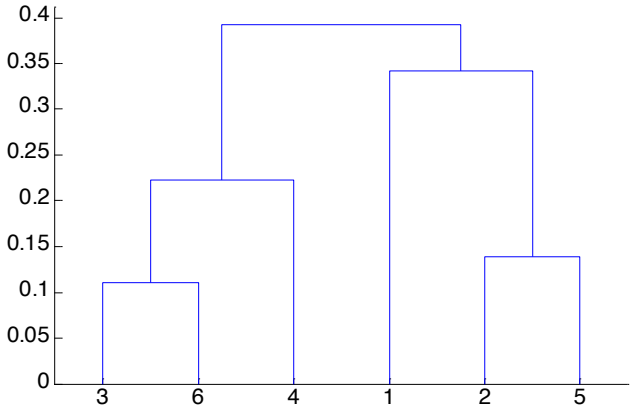
Distance Matrix:

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Hierarchical Clustering: MAX



Nested Clusters



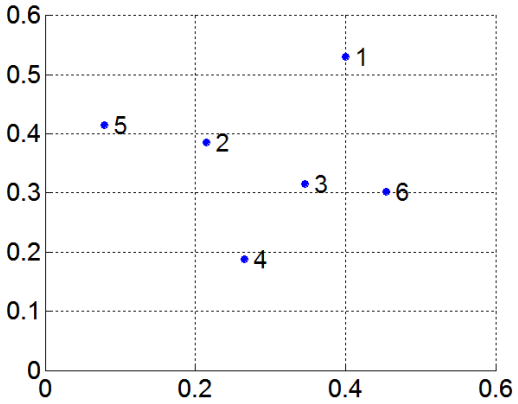
Dendrogram

Group Average

- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum_{\substack{p_i \in \text{Cluster}_i \\ p_j \in \text{Cluster}_j}} \text{proximity}(p_i, p_j)}{|\text{Cluster}_i| \times |\text{Cluster}_j|}$$

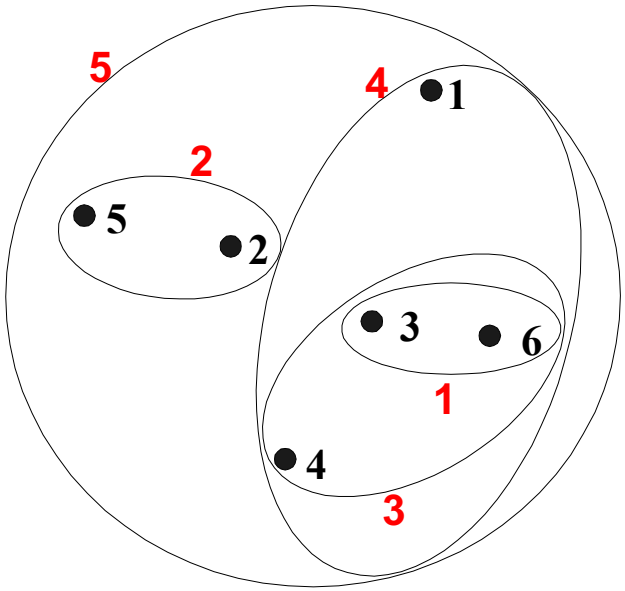
- Need to use average connectivity for scalability since total proximity favors large clusters



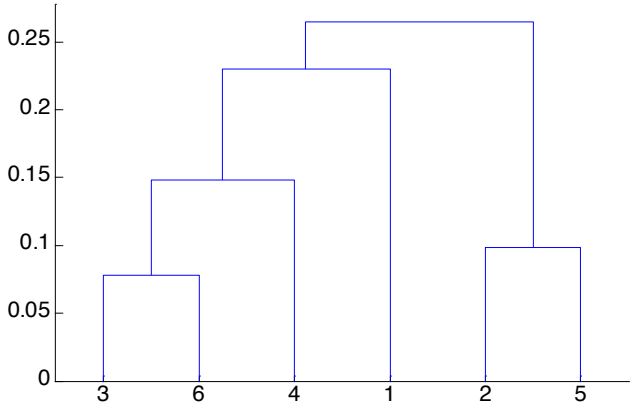
Distance Matrix:

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Hierarchical Clustering: Group Average



Nested Clusters



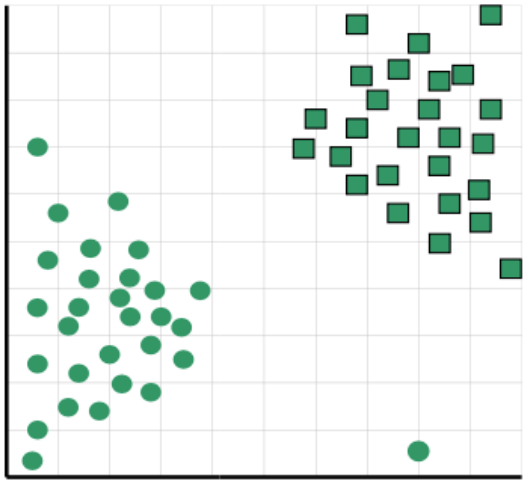
Dendrogram

Hierarchical Clustering: Group Average

- Compromise between Single and Complete Link
- Strengths
  - Less susceptible to noise and outliers
- Limitations
  - Biased towards globular clusters

One potential use of a dendrogram is to detect outliers

The single isolated branch is suggestive of a data point that is very different to all others



Outlier

Hierarchical Clustering: Problems and Limitations

- Once a decision is made to combine two clusters, it cannot be undone
- No objective function is directly minimized
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers
  - Difficulty handling different sized clusters and convex shapes
  - Breaking large clusters

Soft Clustering – Expectation Maximization

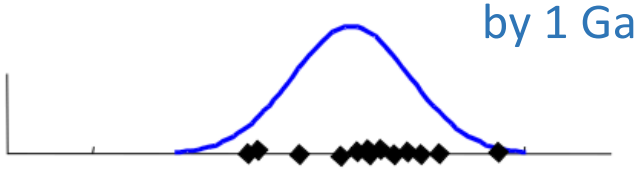
Soft Clustering – Expectation Maximization (EM)

- Clustering typically assumes that each instance is given a "hard" assignment to exactly one cluster.
- Does not allow uncertainty in class membership or for an instance to belong to more than one cluster.
- Problematic because data points that lie roughly midway between cluster centers are assigned to one cluster
- Soft clustering gives probabilities that an instance belongs to each of a set of clusters.
- Let's see how data is generated.

Gaussian Mixture Models

- Gaussian $P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\eta)^2}{2\sigma^2}}$
  - ex. height of one population

Data generated by 1 Gaussian

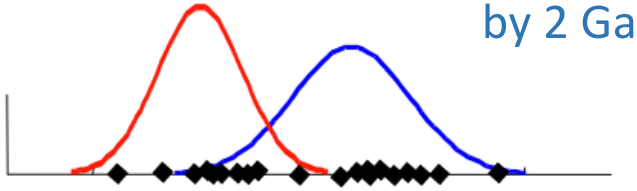


- Gaussian Mixture: Generative modeling framework

Two step process for data generation – choose a Gaussian, and then generate data from that

$$P(C=i) = w_i \quad , \quad P(x \mid C=i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(x-\eta_i)^2}{2\sigma_i^2}}$$

Data generated by 2 Gaussians



$$P(x) = \sum_i P(C=i, x) = \sum_i P(x \mid C=i) P(C=i) = \sum_i w_i \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(x-\eta_i)^2}{2\sigma_i^2}}$$

  - ex. height in two populations

Likelihood of a data point given the model

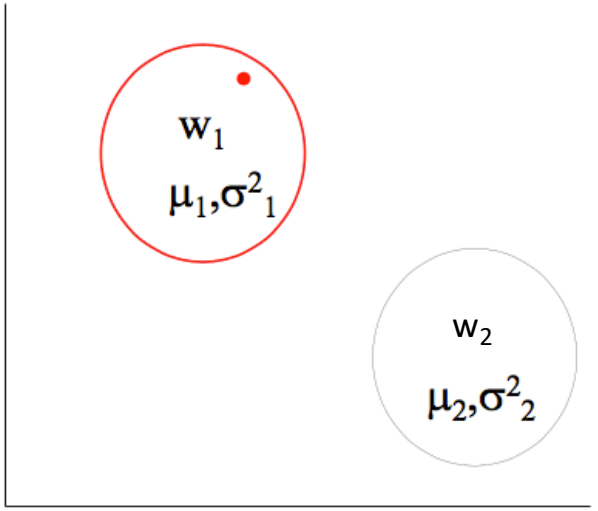
GMM: A generative model

$$\sum_i w_i = 1$$

- Assuming we know the number of components (k), their weights (w_i) and parameters (μ_i, $\sum_i$) we can generate new instances from a GMM in the following way:

  - Pick one component at random with probability w_i for each component

  - Sample a point x from $N(\mu_i, \sum_i)$



Problem is that we don't know the probability of each component (w_i) or parameters for each Gaussian. So how do we proceed?
Random initialization.

Estimating model parameters

- We have a weight, mean and covariance parameters for each class
- As usual we can write the likelihood function for our model

$$p(x_1 \cdots x_n \mid \theta) = \prod_{j=1}^{n} \left(\sum_{i=1}^{k} p(x_j \mid C = i) w_i \right)$$

How do we get the joint probability of x1, x2, …, xn?

For each point, compute its likelihood from each class

Then take the product for each term – assuming data independence

EM algorithm: Key Idea

- Start with random parameters
- Find a class for each example (E-step)
  - Since we are using probabilistic classification, each example will be given a vector of probabilities
- Now we have a supervised learning problem. Estimate the parameters of the model using the maximum likelihood method (M-step)
- Iterate between the E-step and M-step until convergence

GMM+EM = "Soft K-means"

- Decide the number of clusters, K
- Initialize parameters (randomly)
- E-step: assign *probabilistic* membership to all input samples *j*

One for each cluster

$$p_{i,j} = p(C = i \mid x_j) = \frac{p(x_j \mid C = i)p(C = i)}{\sum_k p(x_j \mid C = k)p(C = k)}$$

$$p_i = \sum_j p_{i,j}$$

- M-step: re-estimate parameters based on *probabilistic* membership

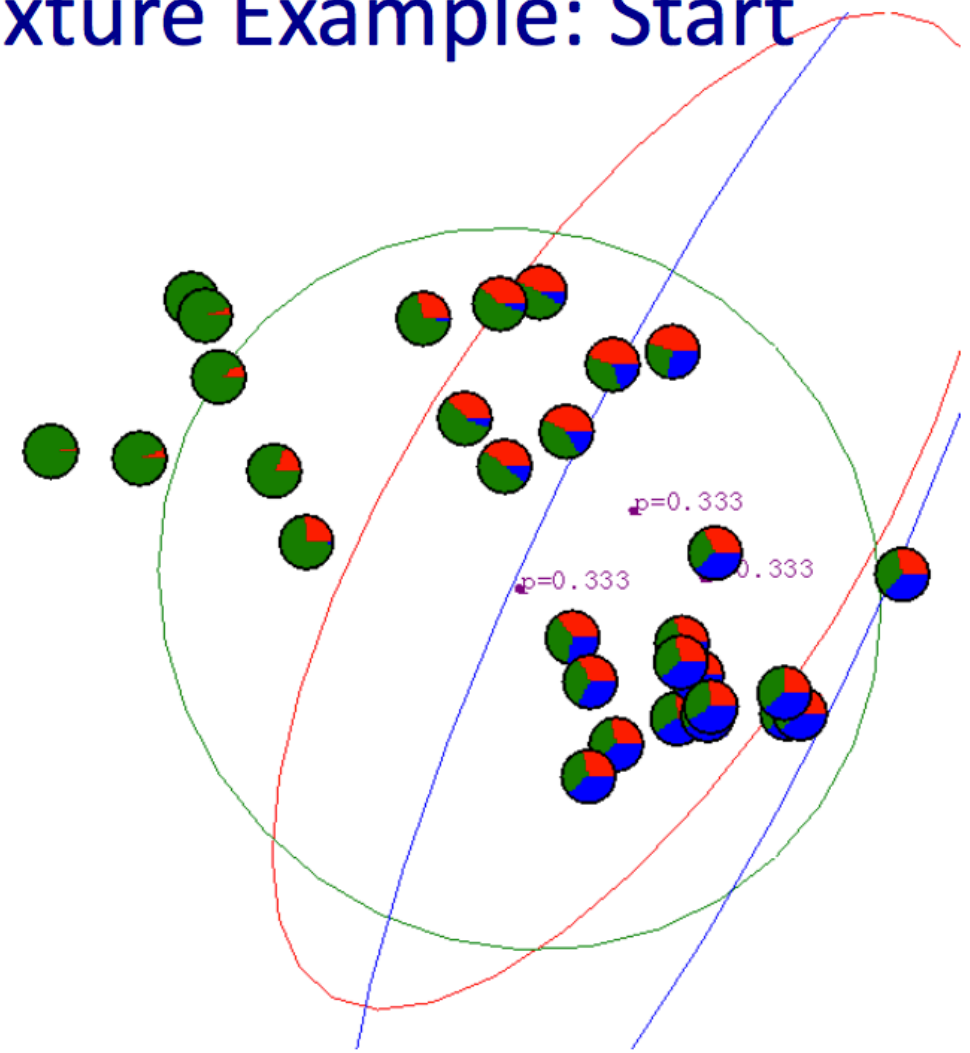
$$\mu_i \leftarrow \sum_j \frac{p_{i,j}\mathbf{x}_j}{p_i}$$

$$\Sigma_i \leftarrow \sum_j \frac{p_{i,j}\mathbf{x}_j\mathbf{x}_j^T}{p_i}$$

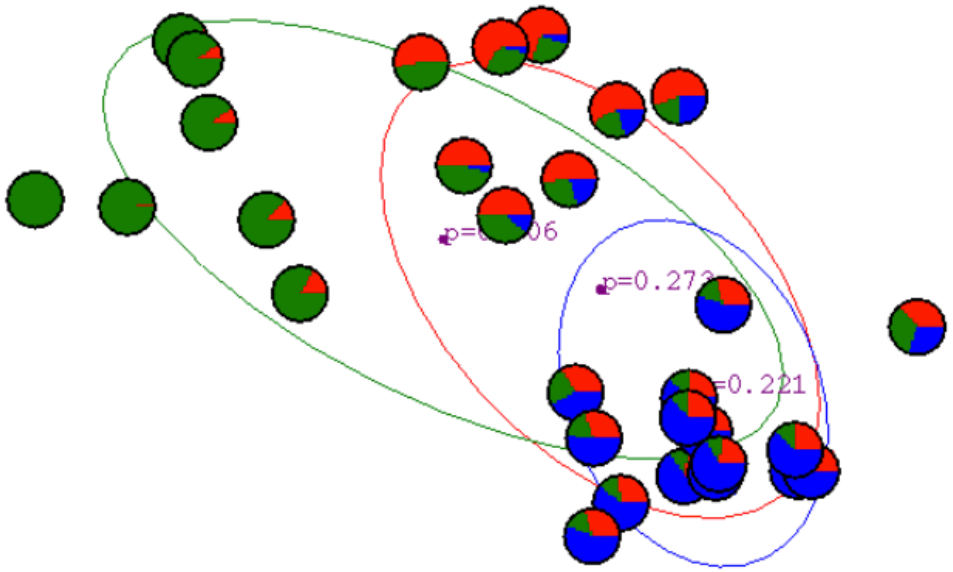
$$w_i = \frac{p_i}{\sum_j p_j}$$

- Repeat until change in parameters are smaller than a threshold

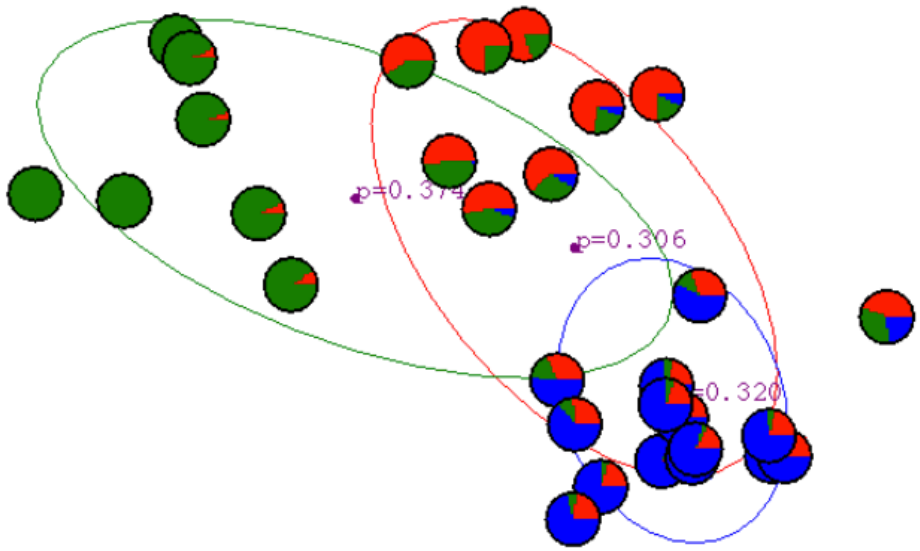
Gaussian Mixture Example: Start



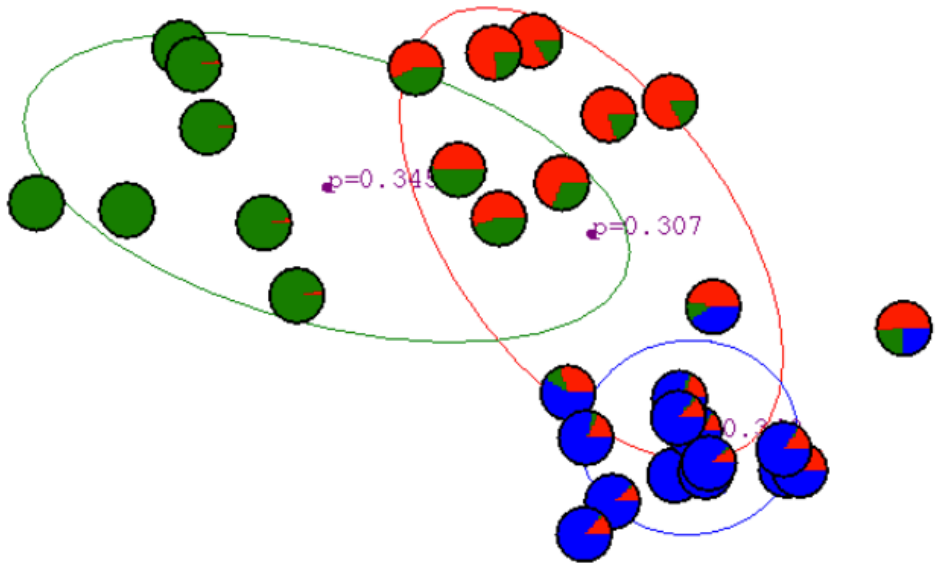
After first iteration



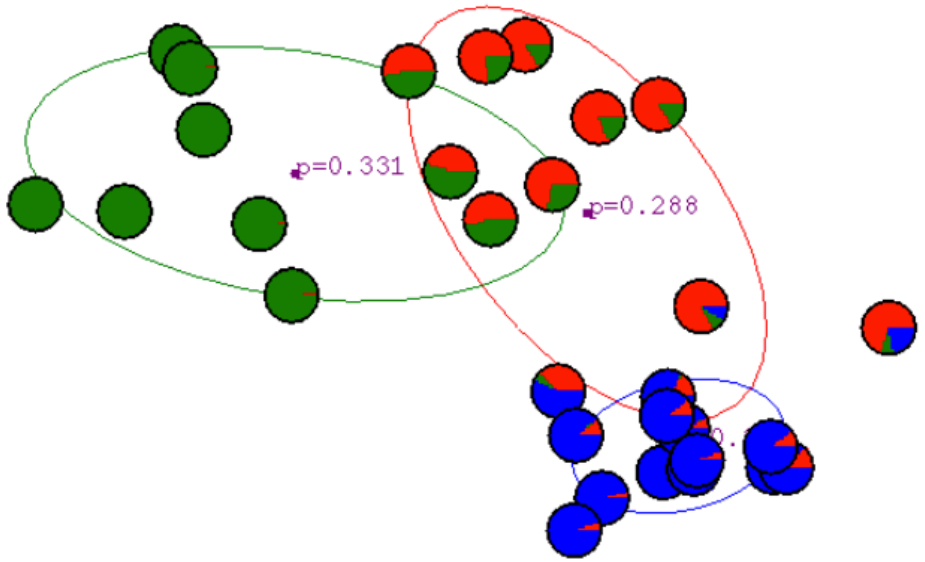
After 2nd iteration



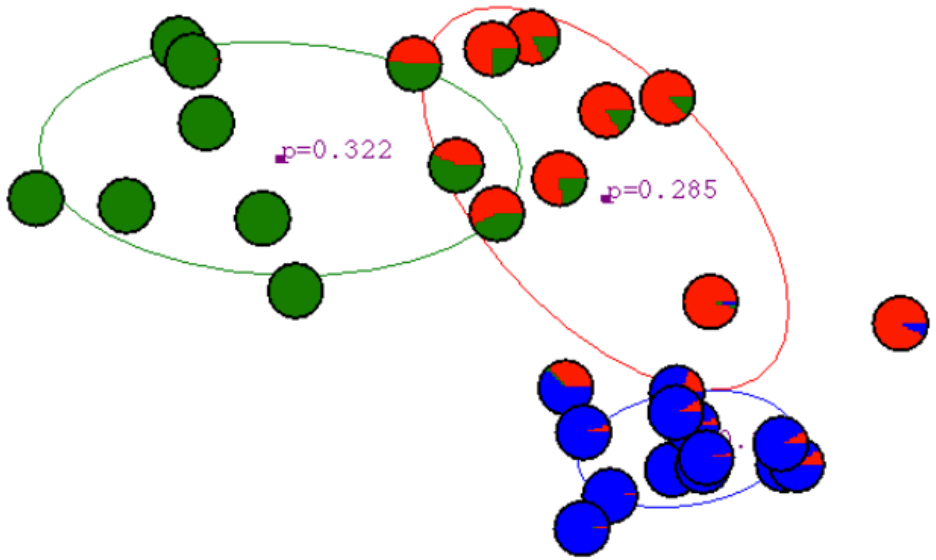
After 3rd iteration



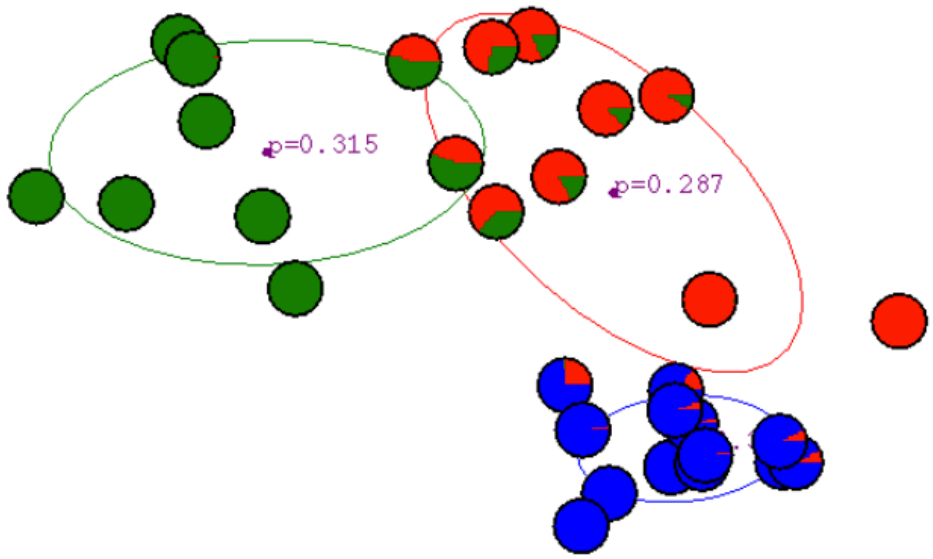
After 4th iteration



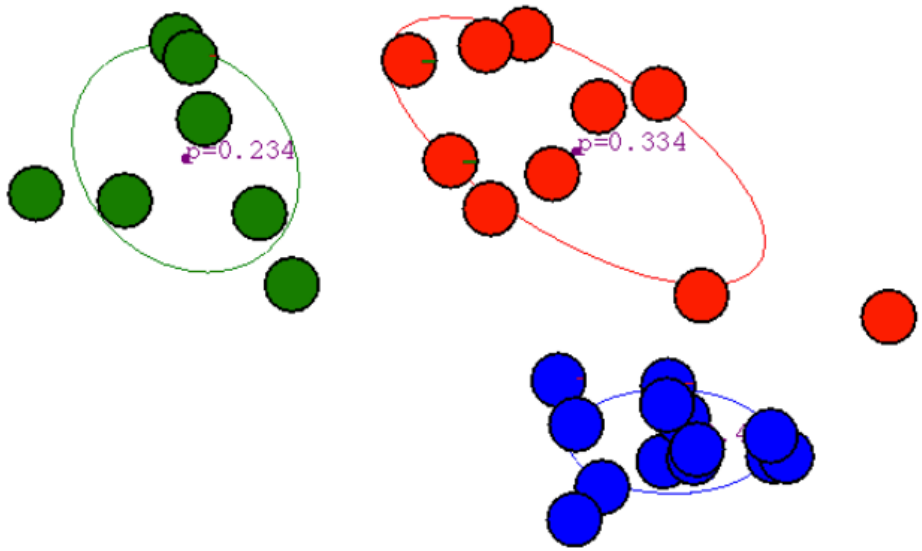
After 5th iteration



After 6th iteration



After 20th iteration



Algorithm: K-means and GMM

1. Decide on a value for *K*, the number of clusters.

2. Initialize the *K* cluster centers / parameters (randomly).

K-means

3. Decide the class memberships of the *N* objects by assigning them to the nearest cluster center.

4. Re-estimate the *K* cluster centers, by assuming the memberships found above are correct.

GMM

3. E-step: assign *probabilistic* membership

4. M-step: re-estimate parameters based on *probabilistic* membership

5. Repeat 3 and 4 until parameters do not change.

Properties of EM

- EM converges to a local minima
  - This is because each iteration improves the log-likelihood
  - Proof same as K-means
    - E-step can never decrease likelihood
    - M-step can never decrease likelihood
- If we make hard assignments instead of soft ones. Algorithm is equivalent to K-means!

Clustering methods: Comparison

	Hierarchical	**K-means**	**GMM**
Running time	naively, $O(N^3)$	fastest (each iteration is linear)	fast (each iteration is linear)
Assumptions	requires a similarity / distance measure	strong assumptions	strongest assumptions
Input parameters	none	K (number of clusters)	K (number of clusters)
Clusters	subjective (only a tree is returned)	exactly K clusters	exactly K clusters

What you should know

- K-means for clustering:
  - algorithm
  - converges because it's coordinate ascent
- Know what agglomerative clustering is
- EM for mixture of Gaussians:
- Remember, E.M. can get stuck in local minima,
  - And empirically it ***DOES!***