ECE7252
Statistical Learning for Signal Processing

Lecture 24: Unsupervised Learning

Chin-Hui Lee
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA
chl@ece.gatech.edu

Outline

- Unsupervised learning (Competitive Learning) and Clustering
  - automated methods to assign feature vectors to *K* clusters
- *K*-Means Clustering Algorithm

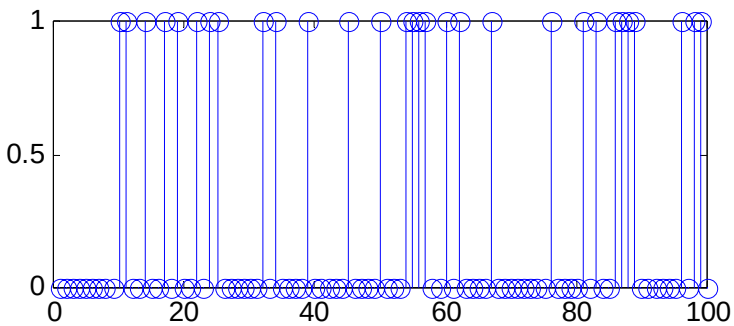
Unsupervised Learning or Clustering

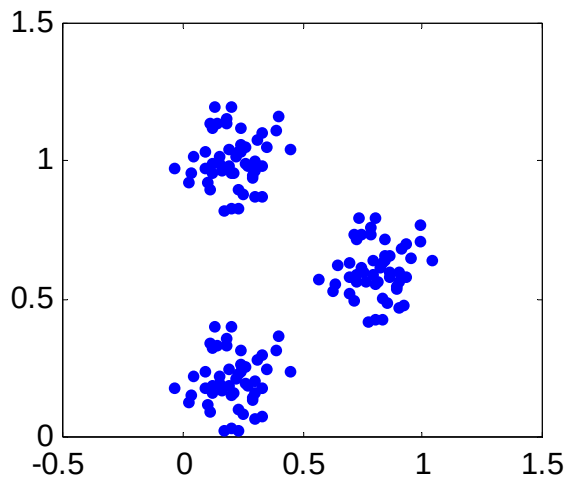
- Previously, each data point had a class label
  - This is “supervised learning”
- In many problems there are no class labels
  - This is “unsupervised learning”
  - Human learning: how do we form categories of objects?
    - Humans are good at creating groups/categories/clusters from data
  - In scientific data analysis finding groups in data is very useful
    - e.g., finding genes that have similar properties
    - e.g., finding galaxies that look the same

Unsupervised Learning

- Data Mining
  - Understand internal/hidden structure of data distribution
- Labeling (Target value, teaching input) Cost is High
  - Large amount of feature vectors
  - Sampling may involve costly experiments
  - Data label may not be available at all
- Pre-processing for classification
  - features within the same cluster are similar
  - often belong to the same class

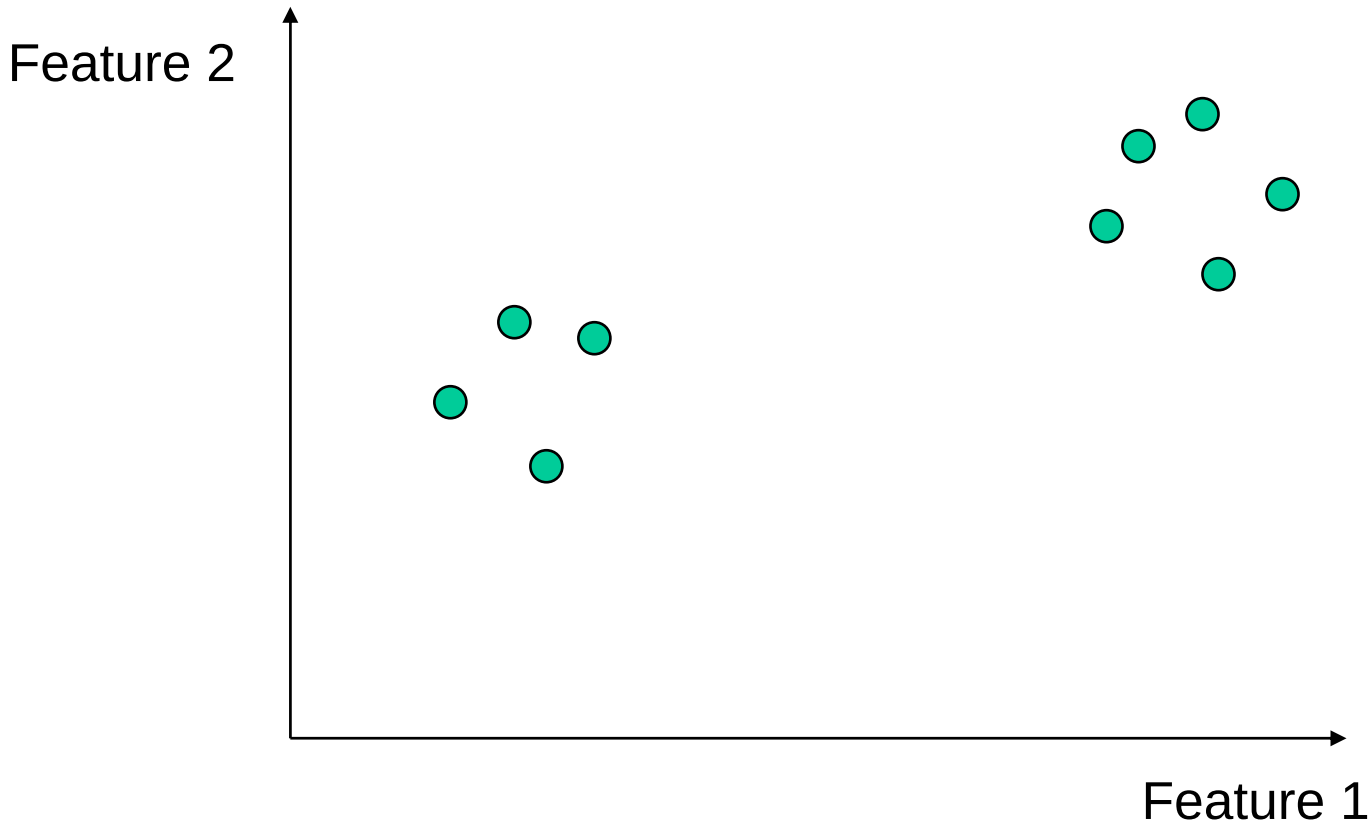
What is “Clustering”?





- What can we learn from these “unlabeled” data samples?
  - Structures: Some samples are closer to each other than other samples
  - The closeness between samples are determined using a “similarity measure”
  - The number of samples per unit volume is related to the concept of “density” or “distribution”

Example: Data in 2 Clusters



The Clustering Problem

- Let $\underline{x} = (x_1, x_2, \ldots, x_d)$ be a d-dim feature vector
- Let *D* be a set of $\underline{x}$ vectors,
  - $D = \{ \underline{x}(1), \underline{x}(2), \ldots \underline{x}(N) \}$
- Given data *D*, group the *N* vectors into *K* groups such that the grouping is "optimal"
- One definition of "optimal":
  - Let mean_k be the mean (centroid) of the *k*-th group
  - Let d_i be the distance from vector $\underline{x}(i)$ to the closest mean
    - so each data point $\underline{x}$(i) is assigned to one of the K means

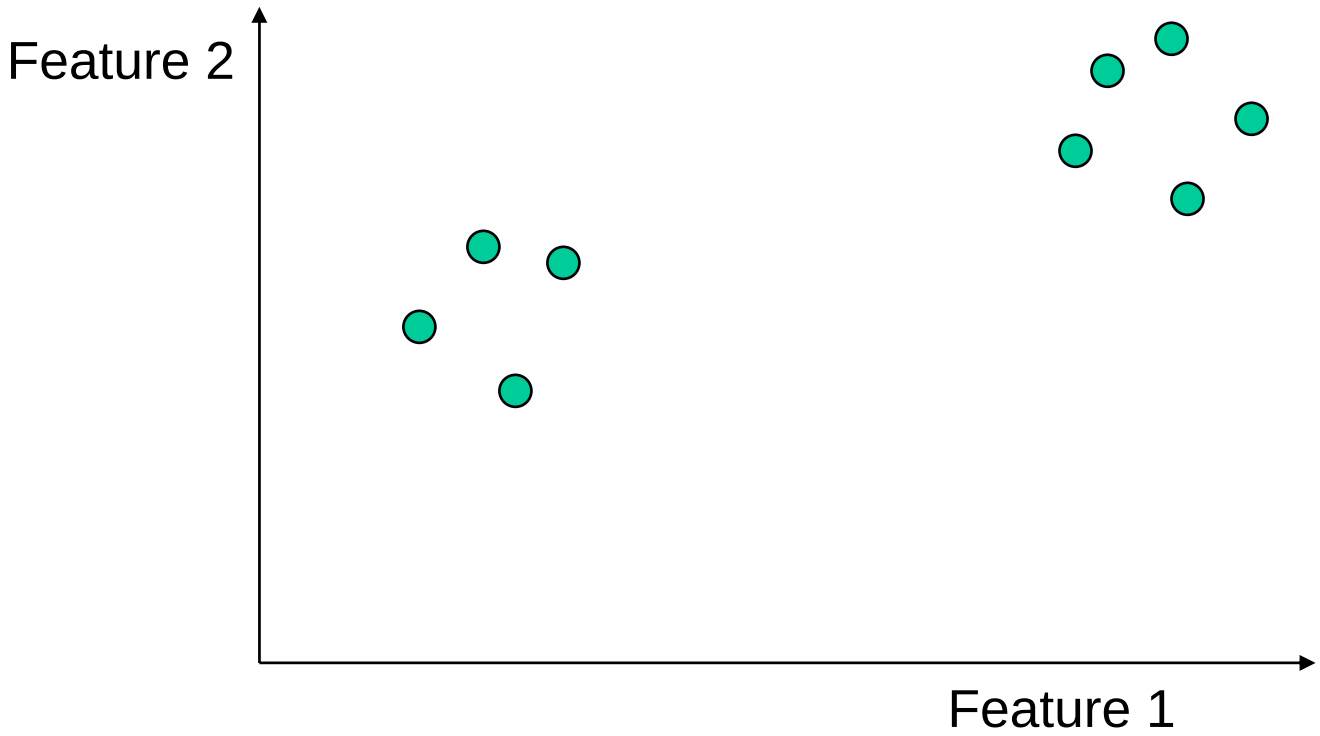
Optimal Clustering

- Let mean_k be the mean of the k-th cluster
  - mean_k is the average vector of all vectors $\underline{x}$ “assigned to” cluster k
- One definition of “optimal”:
  - Let d_i be the distance from vector $\underline{x}(i)$ to the closest mean
    - so each data point $\underline{x}(i)$ is assigned to one of the *K* means
    - Q_k = quality of cluster k = \Sum d_i ,
      - where the sum is over $\underline{x}$(i) assigned to cluster *k*
      - then Q_k’s measure how “compact” each cluster is
    - We want to minimize the total sum of the Q_k’s

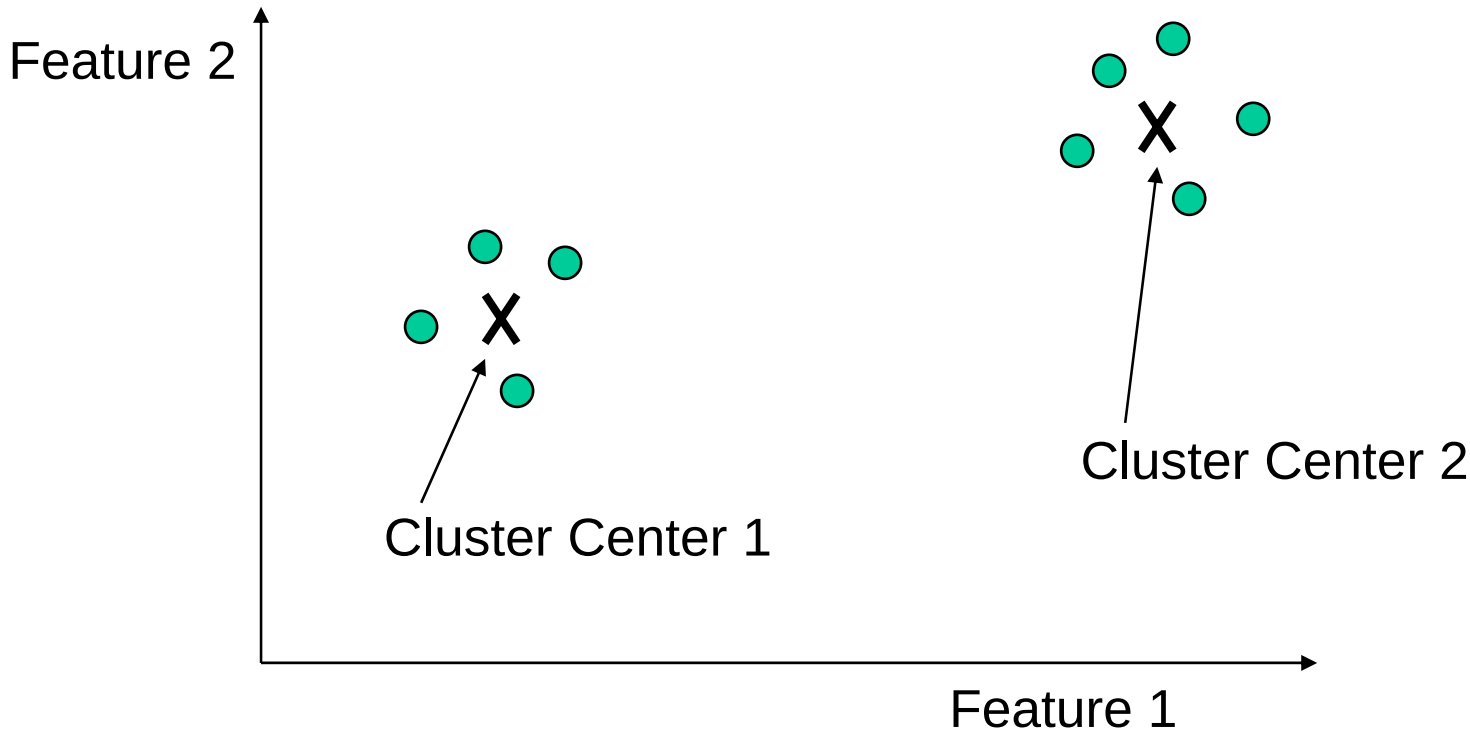
The Total Squared Error Objective

- Let d_i = distance from feature vector x(i) to the closest mean = squared Euclidean distance between x(i) and mean_k
- Now Q_k = sum of squared distances for points in a cluster
- Total Squared Error (TSE)
  - TSE = Total Squared_Error = \sum Q_k
    - where sum is over all K clusters (and each Q_k is itself a sum)
  - TSE measures how “compact” a clustering is

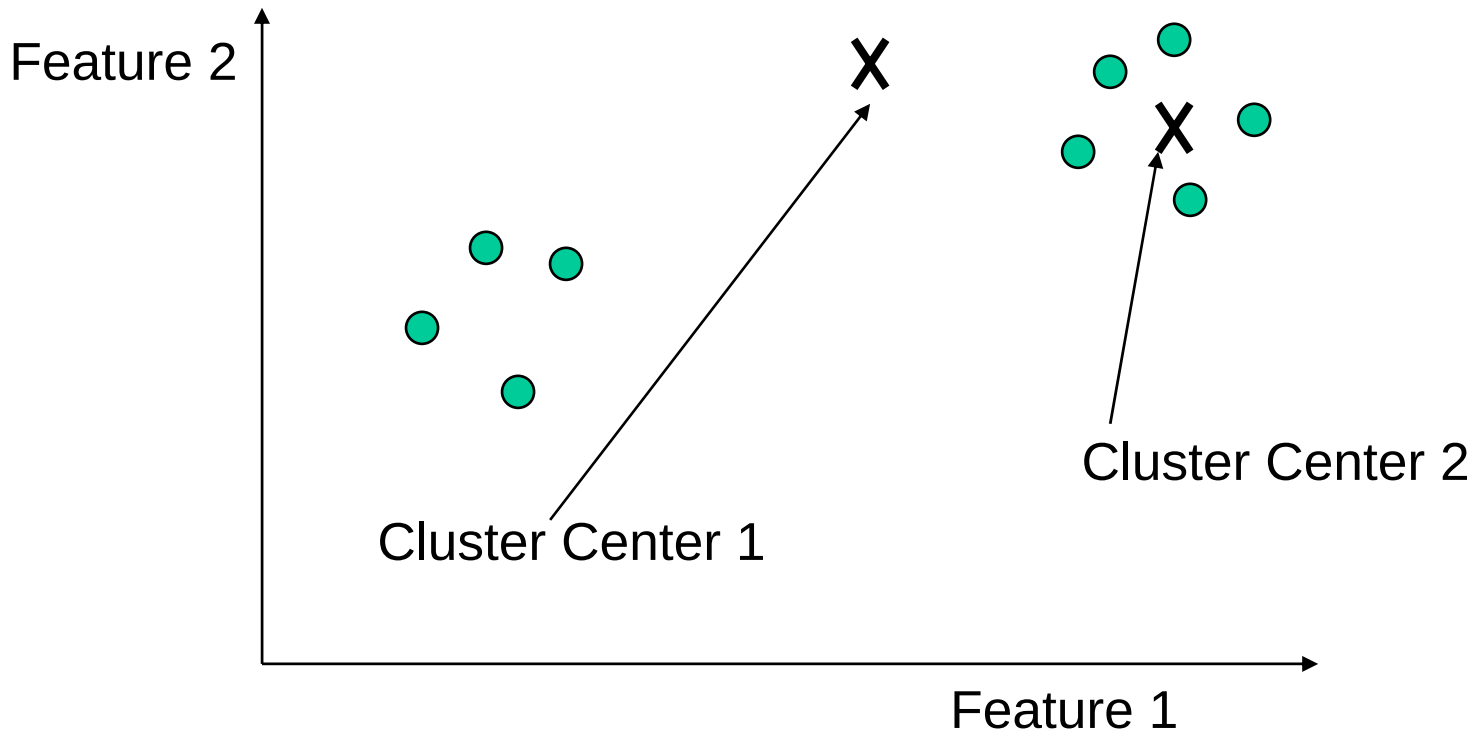
Example: Data in 2 Clusters



“Compact” Clustering: Low TSE



“Non-Compact” Clustering: High TSE



The Clustering Problem

- Let *D* be a set of $\underline{x}$ vectors,
  - $D = \{ \underline{x}(1), \underline{x}(2), \ldots\ldots \underline{x}(N) \}$
- Fix a value for *K*, e.g., $K = 2$
- Find the locations of the *K* means that minimize the TSE
  - No direct solution
  - Can use an iterative search algorithm to minimize TSE

The Algorithm for K-means Clustering

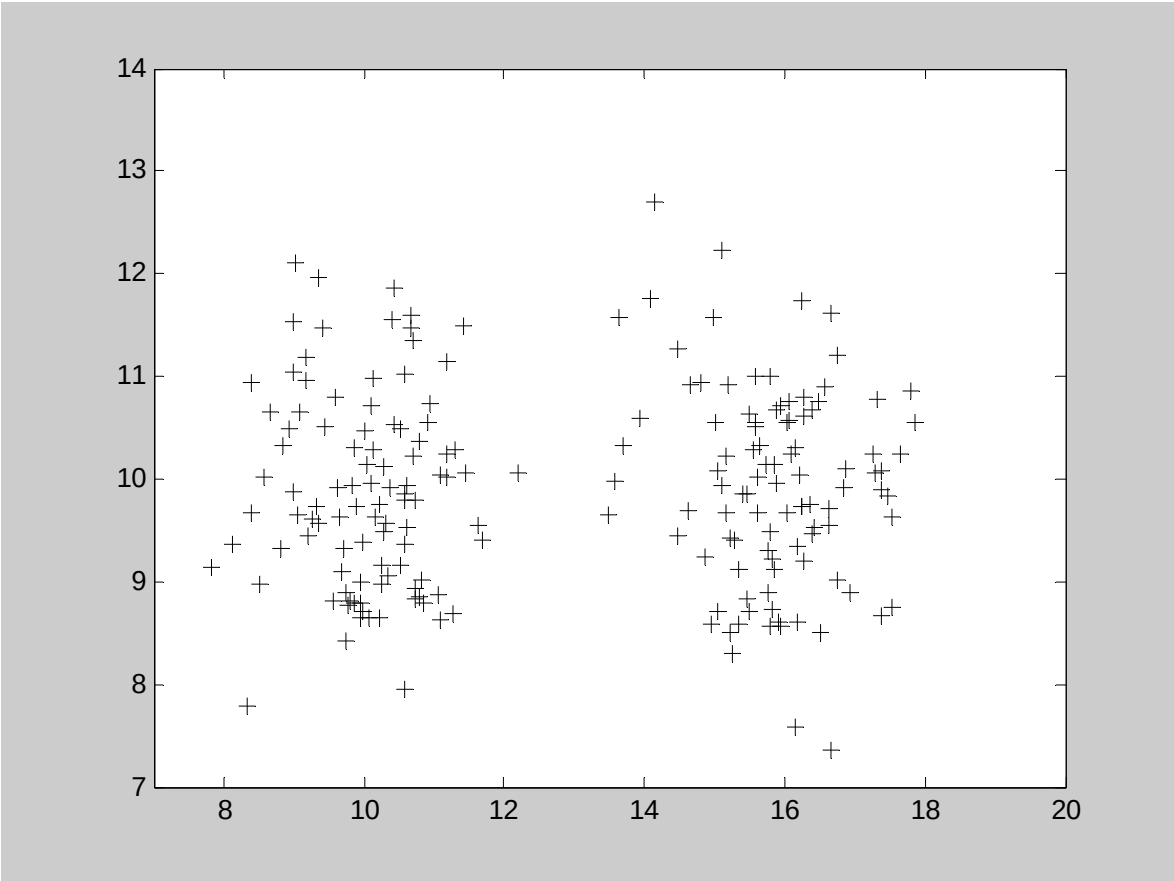
Inputs: data *D*, with *N* feature vectors
K = number of clusters

Outputs: *K* mean vectors (centers of *K* clusters)
memberships for each of the *N* feature vectors

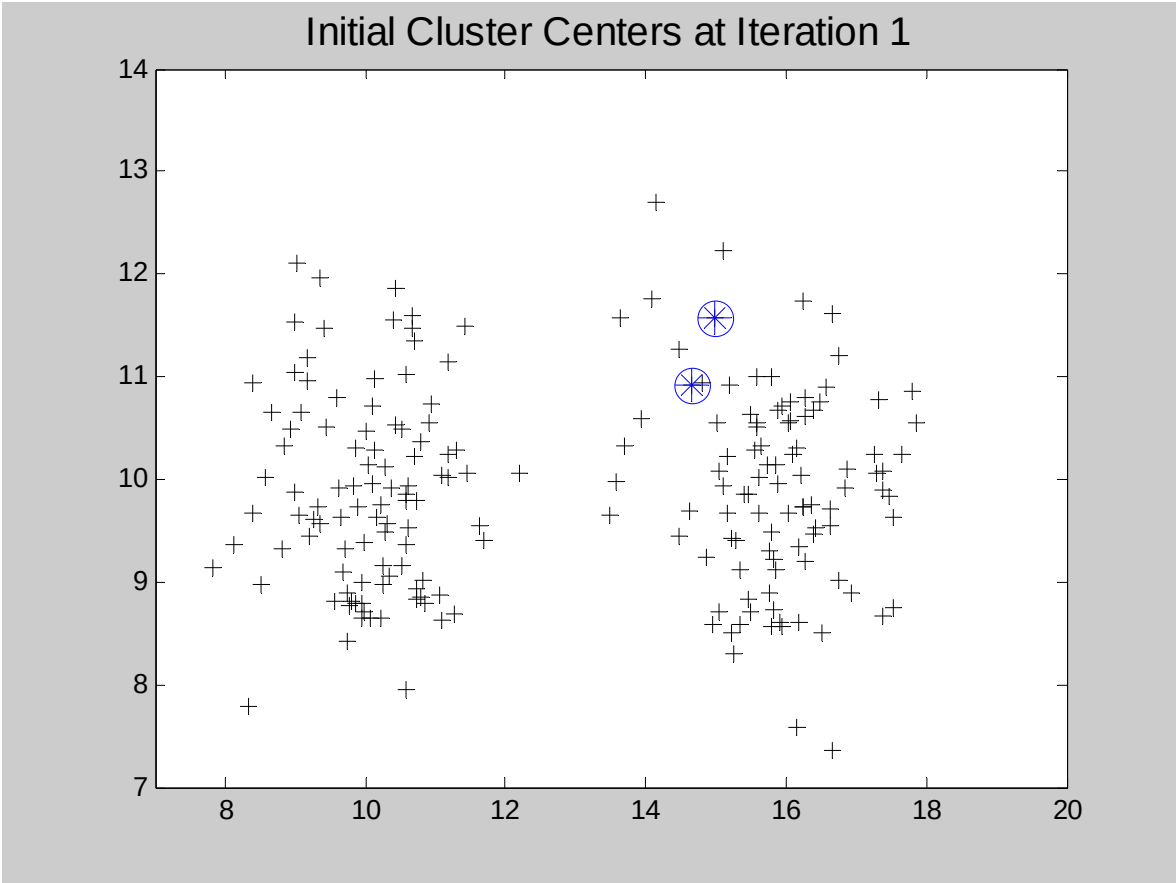
K-means Clustering (Pseudo Code)

```
kmeans(D, k)
        choose K initial means randomly (e.g., pick K points randomly from D)
        while means_are_changing
                %  assign each point to a cluster
                for i = 1: N
                      membership[x(i)] = cluster with mean closest to x(i)
                end

                % update the means
                for k = 1:K
                      mean_k = average of vectors x(i) assigned to cluster k
                end

                % check for convergence
                          if (new means are the same as old means) then halt
                else means_are_changing = 1
        end
```

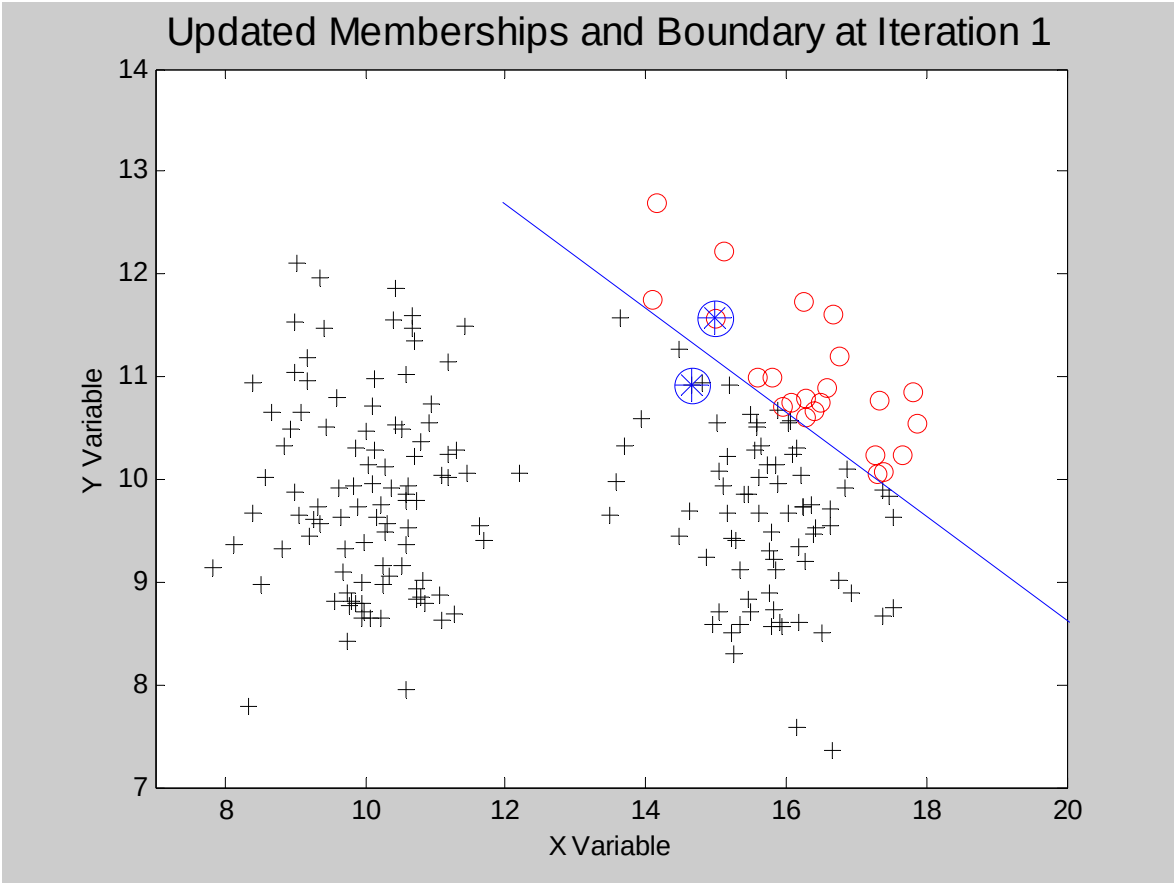
Original Data (2 Dimensions)



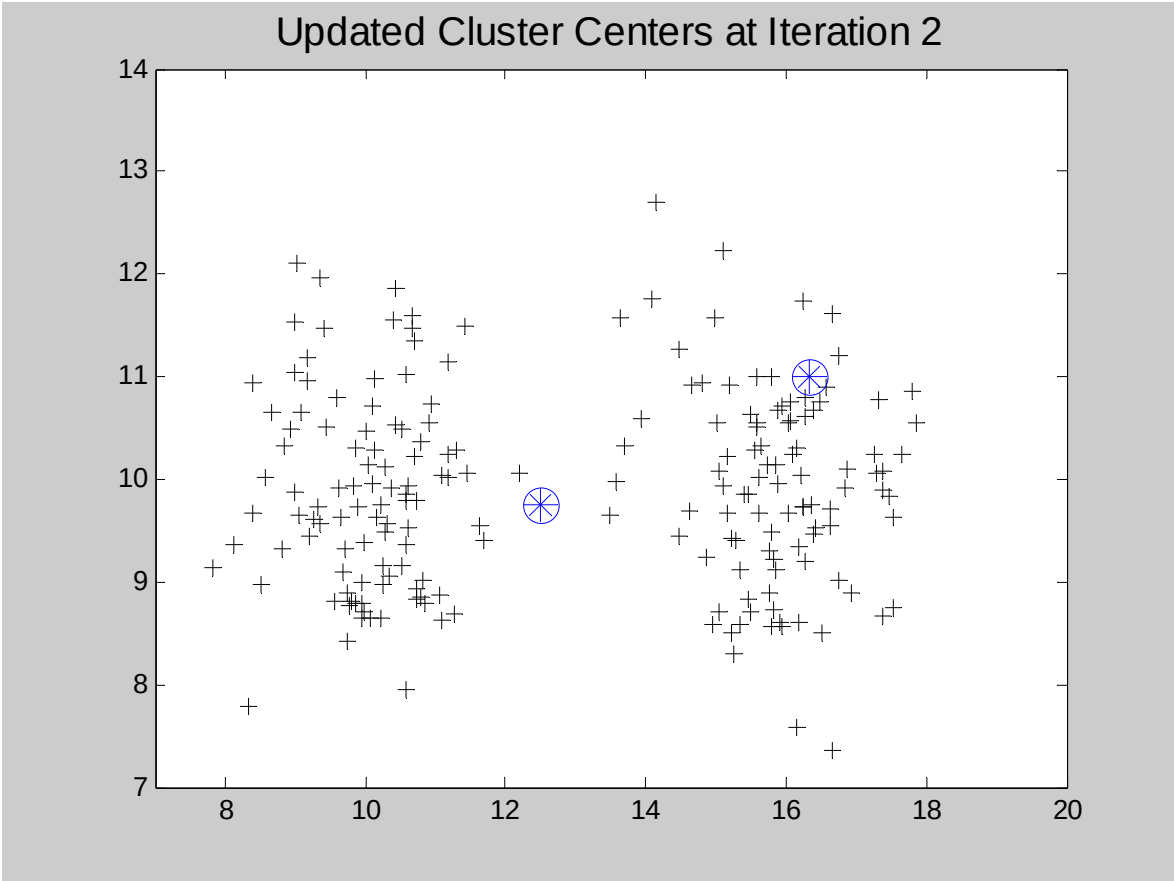
Initial Cluster Centers for *K*-means



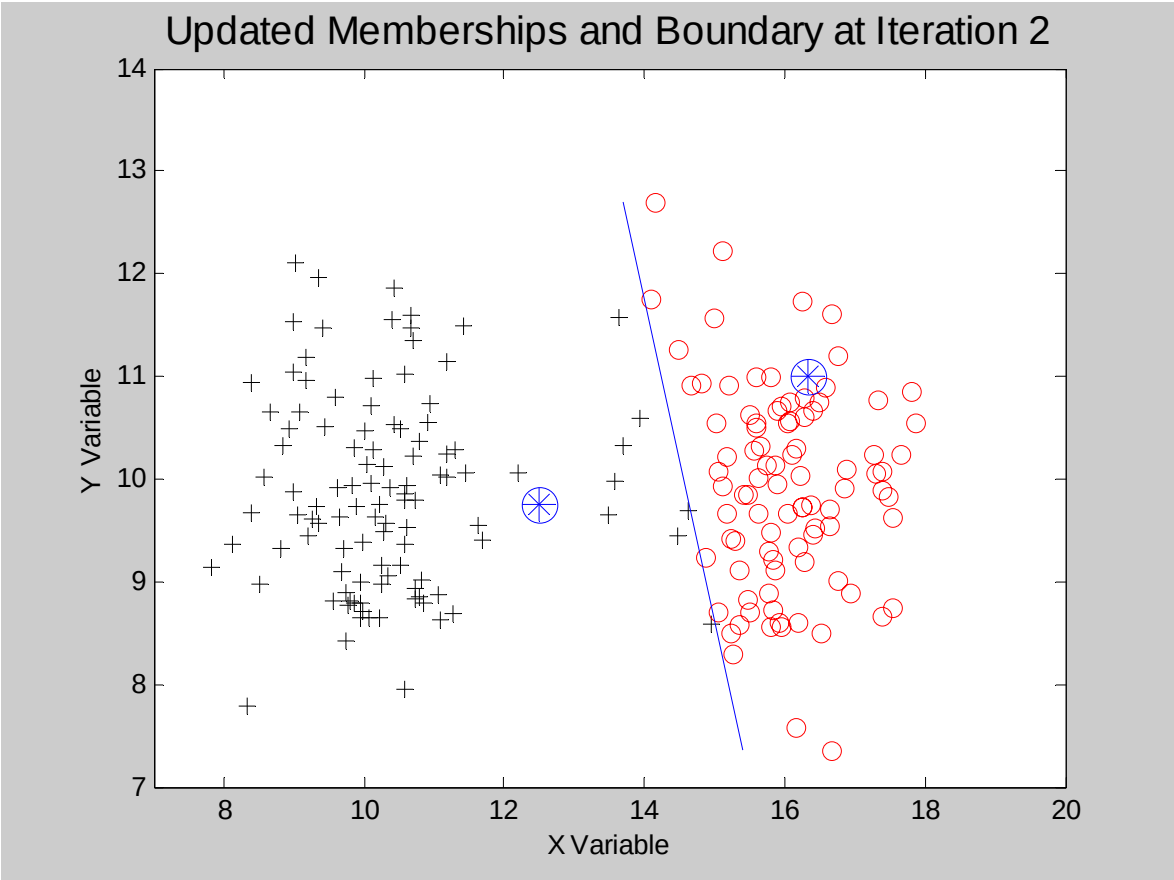
Update Memberships (Iteration 1)



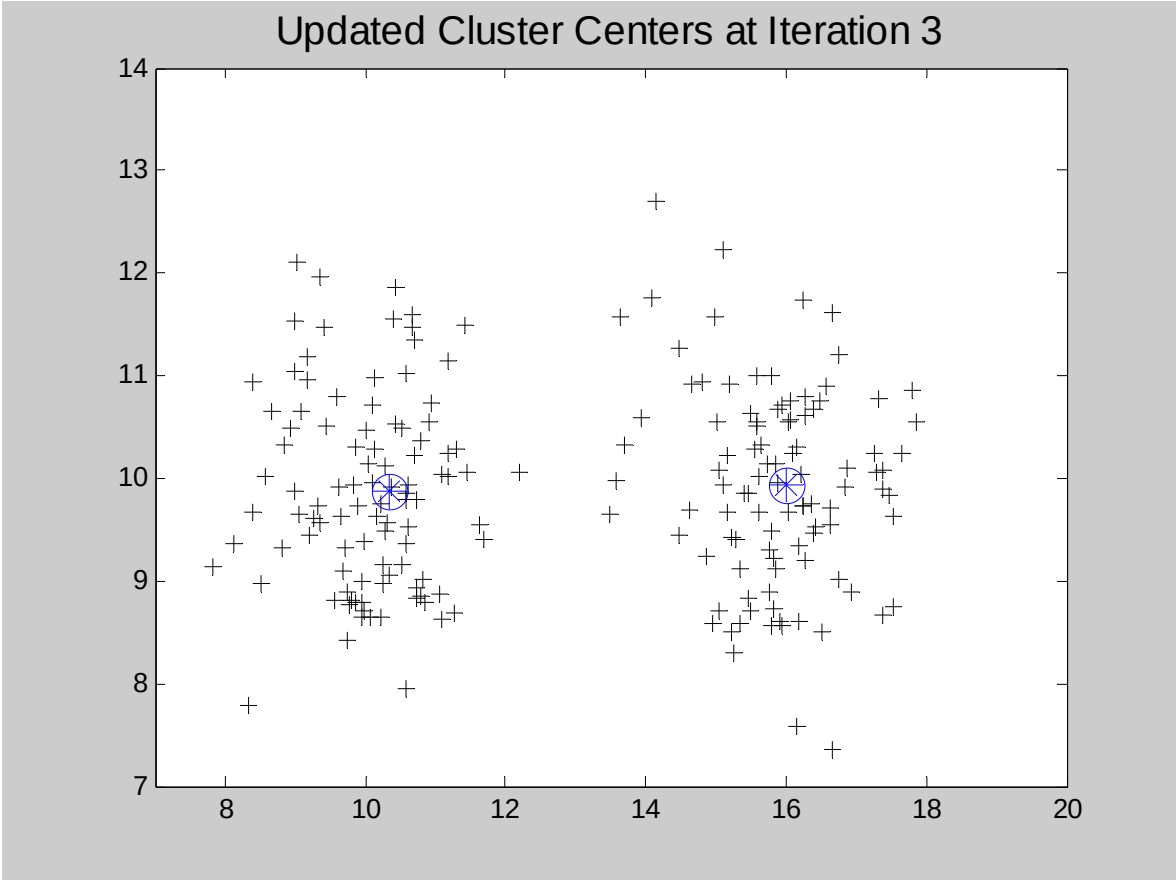
Update Cluster Centers at Iteration 2



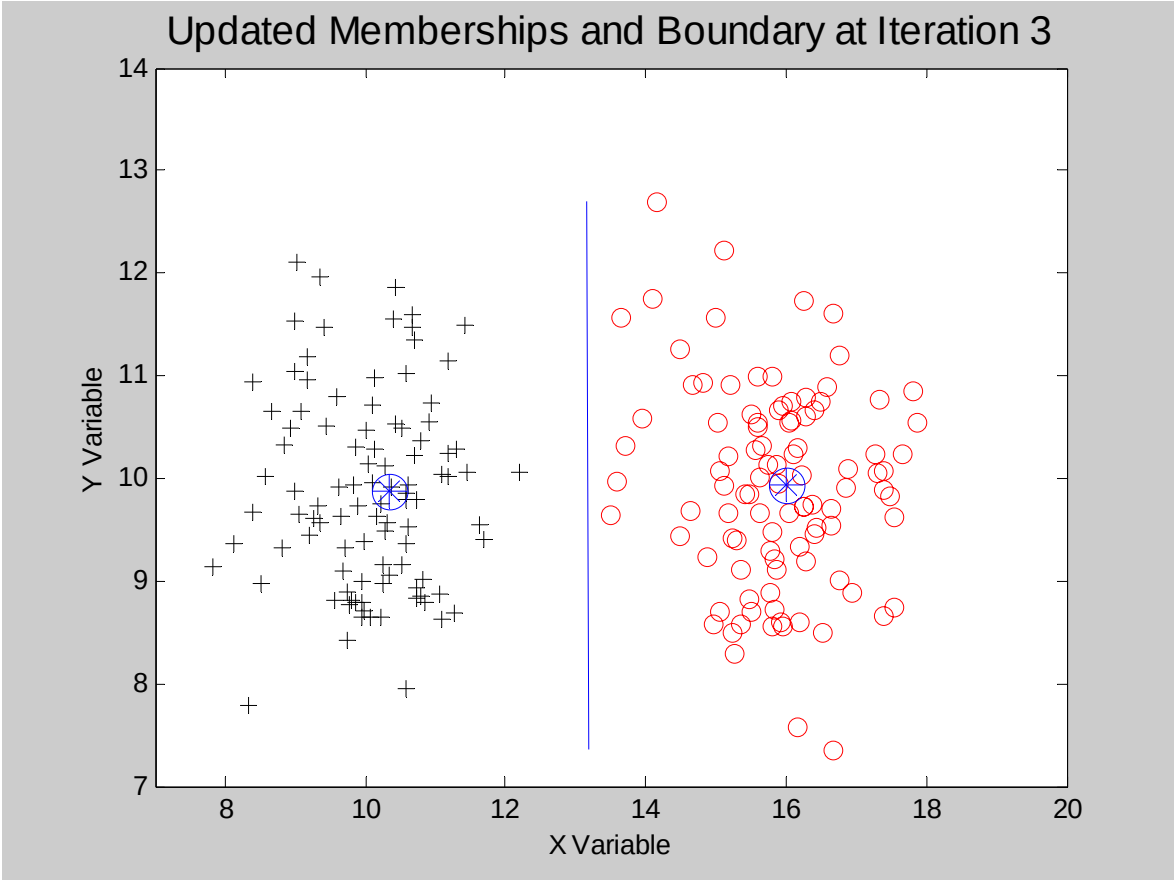
Update Memberships (Iteration 2)



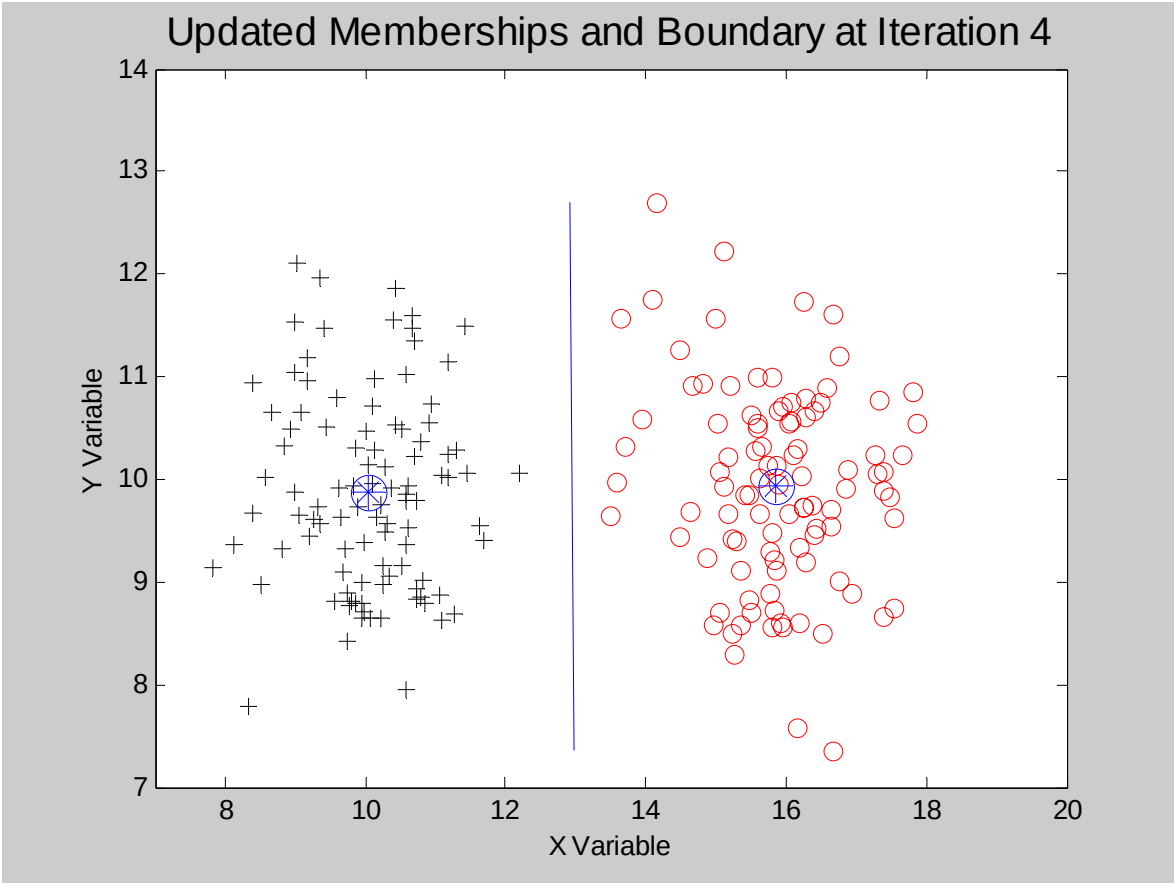
Update Cluster Centers at Iteration 3



Update Memberships (Iteration 3)



Updated Memberships (Iteration 4)



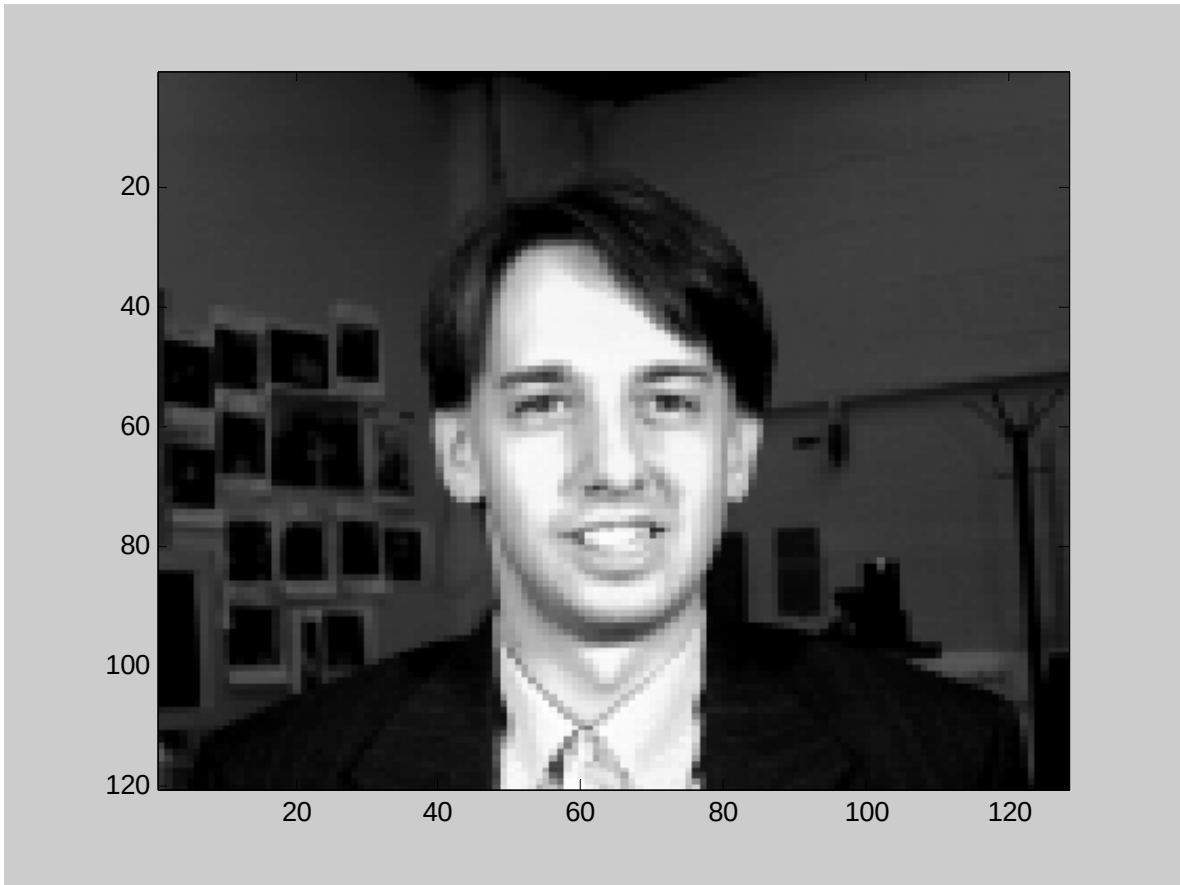
Comments on the K-means Algorithm

- Time Complexity
  - Per iteration = Nd + Kd = O(Nd)
- Can show that TSE decreases (or converges) at each iteration
- Does it find the global minimum of TSE?
  - No, not necessarily
  - In a sense it is doing “steepest descent” from a random initial starting point
  - Thus, results will be sensitive to the starting point
    - in practice, we can run it from multiple starting points and pick the solution with the lowest TSE (the most “compact” solution)

Example: Clustering Pixels in an Image

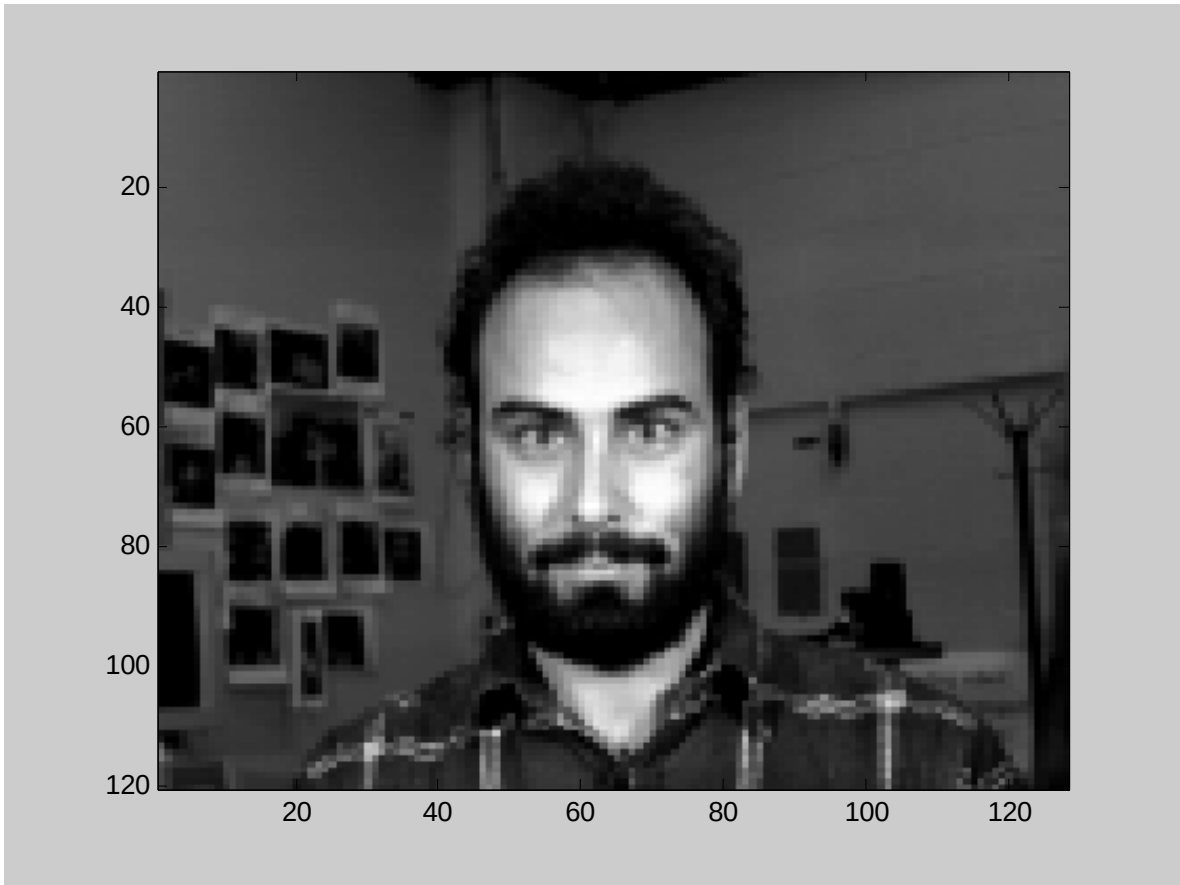
- We can use *K*-means to cluster pixel intensities in an image into *K* clusters
  - This provides a simple way to "segment" an image into *K* regions of similar "compact" image intensities
  - More automated than manual thresholding of an image
- How to do this?
  - Size(image pixel matrix) = *m x n*
  - convert to a vector with (*m x n*) rows and 1 column
    - this is a 1-dimensional feature vector of pixel intensities
  - run the *k*-means with input = vector of intensities
  - assign each pixel the "color or grayscale" of the cluster it is assigned to

Image Example: Original Image



Segmentation with *K*-means: *K* = 2

Another Image Example



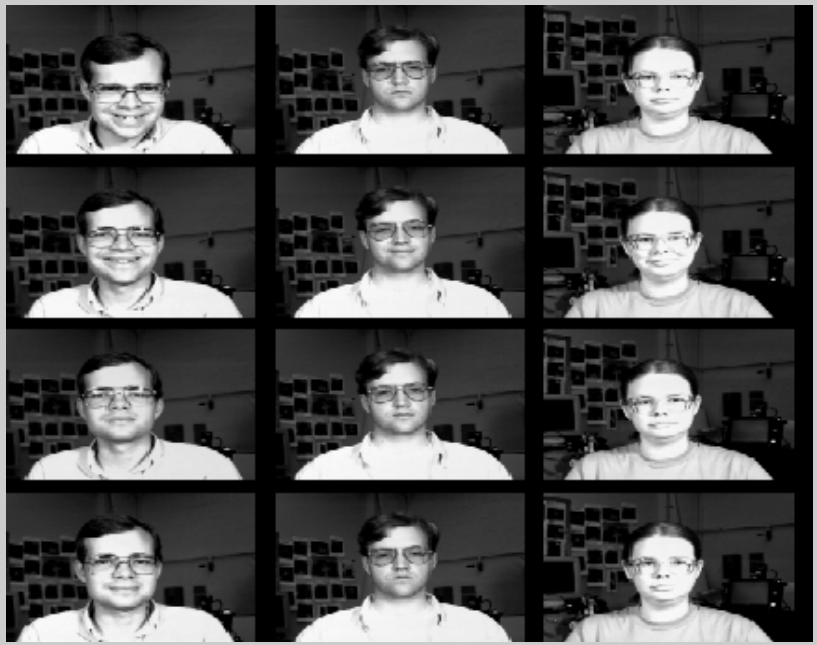
Segmentation with K=3

Clustering Full Images

- We can also cluster sets of images into groups
  - Now each vector = a full image (dimensions 1 x (*mxn*))
  - *N* images of size *m x n*
    - A matrix with *N* rows and (*m x n*) columns
  - Call *k*-means (MATLAB) with *D* = this matrix
    - Clustering in an (*m x n*) dimensional space
  - *K*-means will group the images into *K* groups

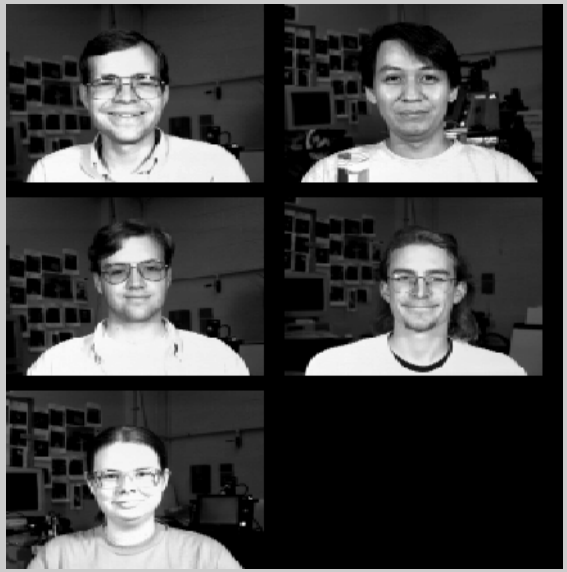
Example: First 5 Individuals, K = 2

Cluster 1



Cluster 2

All Individuals, Happy Faces, K=5



K-Means Word Clustering

Example: 9492 words into 100 clusters (from web)

oub bank Singapore cent uob db account share singtel trade Bangkok manage save entity annual ocbc tangible debt sti keppel custom transact currency deposit card sixth citibank integer subscribe handset creation loan auditor merger autom merge sharehold attract uncondi asx optu sembawang ibra restructur singland landlord uic yaw sgx

K-means Document Clustering

- 2000 documents into 100 clusters (one example)

- N Korea Proposes Resumed Talks with S Korea-Yonhap
- North Korea Proposes Resuming Talks with Seoul
- South Korea Set for Key Vote on Approach to North
- Korea to Replace Four to Eight Ministers on Friday
- S.Korea to Push North Policy Despite Kim Setback

……

Competitive Learning

- A form of unsupervised learning
- Neurons compete against each other with their activation values. The winner(s) reserve the privilege to update their weights. The losers may even be punished by updating their weights in opposite direction
- Competitive and Cooperative Learning:
  - Competitive: Only one neuron's activation can be reinforced
  - Cooperative: Several neurons' activation can be reinforced

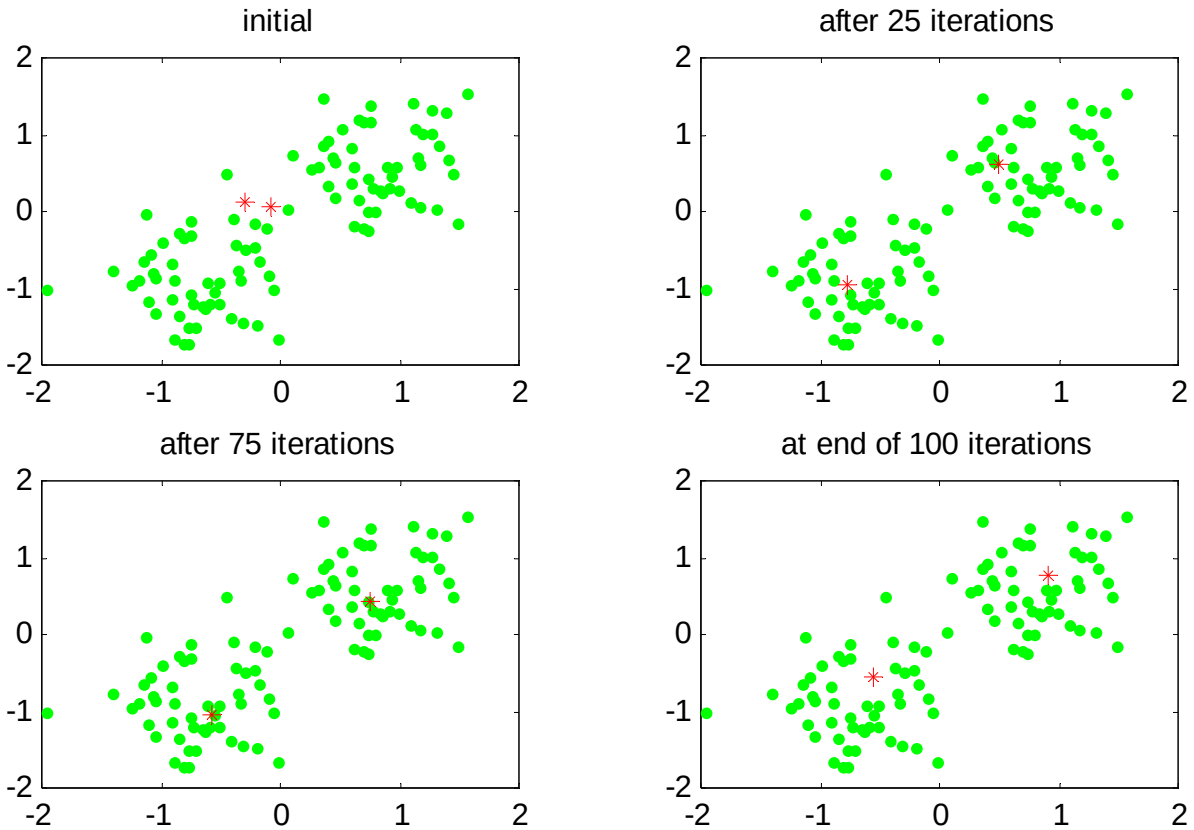
Competitive Learning Rule

- A neuron WINS the competition if its output is largest among all neurons for the same input *x(n)*
- The weights of the winning neuron (*k*-th) is adjusted:

$$D\, w_k(n) \propto [x(n) - w_k(n)]$$

- The positions of losing neurons remain unchanged
- If the weights of a neuron represents its POSITION. If the output of a neuron is inversely proportional to the distance between *x(n)* and $w_k(n)$, then
  - Competitive Learning = CLUSTERING!

Competitive Learning Example



Summary: Clustering Problem

- Given a set of vectors $\{x_n; 1 \le n \le N\}$, find a set of K clustering centers $\{\boldsymbol{w}(k); 1 \le k \le K\}$ such that each x_n is assigned to a cluster, say, $\mathbf{w}(k^*)$, according to a distance (distortion, similarity) measure $d(x_n, \mathbf{w}(k))$ to minimize average distortion

$$D = \frac{1}{N} \sum_{k=1}^{K} \sum_{n=1}^{N} I(x_n, k) d(x_n, w(k))$$

- $\boldsymbol{I}(x_n,k) = 1$ if x is assigned to cluster k with cluster center $w(k)$; and = 0 otherwise -- indicator function
- Key: reasonable distortion (or similarity) measure

Summary: *K*-means (EM) Algorithm

Initialization: Initial cluster center $\boldsymbol{w}(k)$; $1 \le k \le K$, $D(-1) = 0$, $\boldsymbol{I}(x_n,k) = 0$, $1 \le n \le N$, $1 \le k \le K$;

Repeat

(A) Assign cluster membership (Expectation step)

Evaluate $d(x_n, \boldsymbol{w}(k))$; $1 \le n \le N$, $1 \le k \le K$

$\boldsymbol{I}(x_n,k) = 1$ if $d(x_n, \boldsymbol{w}(k)) < d(x_n, \boldsymbol{w}(m))$, $m \ne k$; $1 \le n \le N$

$= 0$; otherwise

(B) Evaluate distortion D: $D(iter) = \sum_{n=1}^{N} I(x_n,k) d(x_n, w(k)) \quad 1 \le k \le K$

(C) Update code words with new assignment (Maximization)

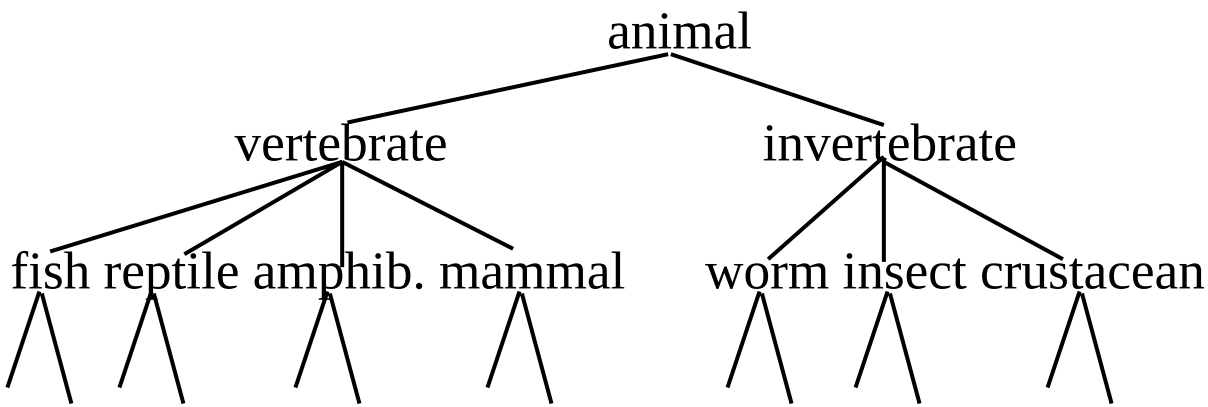
$$w(k) = \sum_{n=1}^{N} I(x_n,k) x_n, \; N_k = \sum_{n=1}^{N} I(x_n,k), 1 \le k \le K$$

(D) Check for convergence

if $1 - D(\text{Iter}-1)/D(\text{Iter}) < e$, then convergence = TRUE,

Hierarchical Clustering

- Build a tree-based hierarchical taxonomy (*dendrogram*) from a set of unlabeled examples



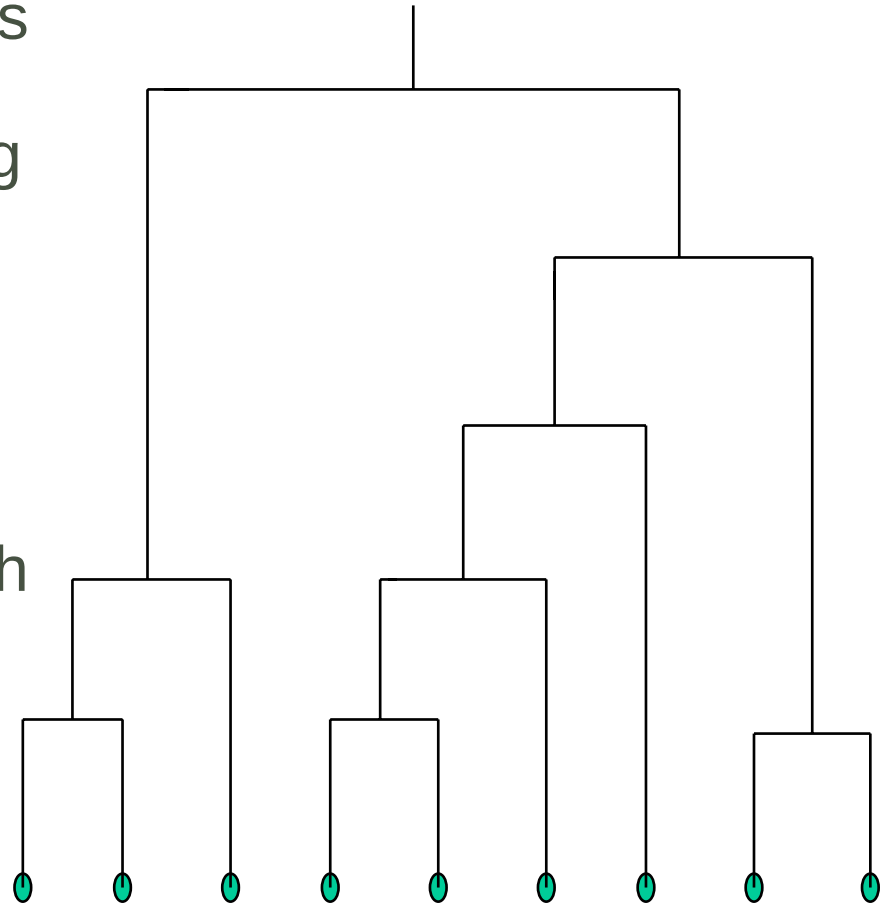
- One option to produce a hierarchical clustering is recursive application of a partition clustering algorithm to produce a hierarchical clustering

Hierarchical Agglomerative Clustering

- HAC: Assuming a goodness-of-fit function for determining the similarity of two instances
- Starting with all instances in a separate cluster and then repeatedly joins the two clusters that are most similar until there is only one cluster
  - Among the current clusters, determine the two clusters, c_i and c_j, that are most similar
  - Replace c_i and c_j with a single cluster $c_i \cup c_j$
- The history of merging forms a binary tree or hierarchy

A Dendrogram: Hierarchical Clustering

- Dendrogram: Decomposes data objects into a several levels of nested partitioning (tree of clusters).
- Clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster.



Hierarchical Clustering Algorithms

- **Agglomerative (bottom-up):**
  - Starting with each document being a single cluster
  - Eventually all documents belong to the same cluster
- **Divisive (top-down):**
  - Start with all documents belong to the same cluster
  - Eventually each node forms a cluster on its own
- Does not require the number of clusters ***k*** in advance
- Needs a termination/readout condition
  - The final mode in both agglomerative and divisive is of no use

“Closest Pair” of Clusters

- Many variants to defining closest pair of clusters
- “Center of gravity”
  - Clusters whose centroids (centers of gravity) are the most cosine-similar
- Average-link
  - Average cosine between pairs of elements
- Single-link
  - Similarity of the most cosine-similar (single-link)
- Complete-link
  - Similarity of the “furthest” points, the least cosine-similar

Key Concerns with HAC

- Key problem: as clusters are being formed, how to represent the location of each cluster, to tell which pair of clusters is closest?
- Euclidean case: each cluster has a *centroid* = average of its points
  - Measure intercluster distances by distances of centroids

Single Link Agglomerative Clustering

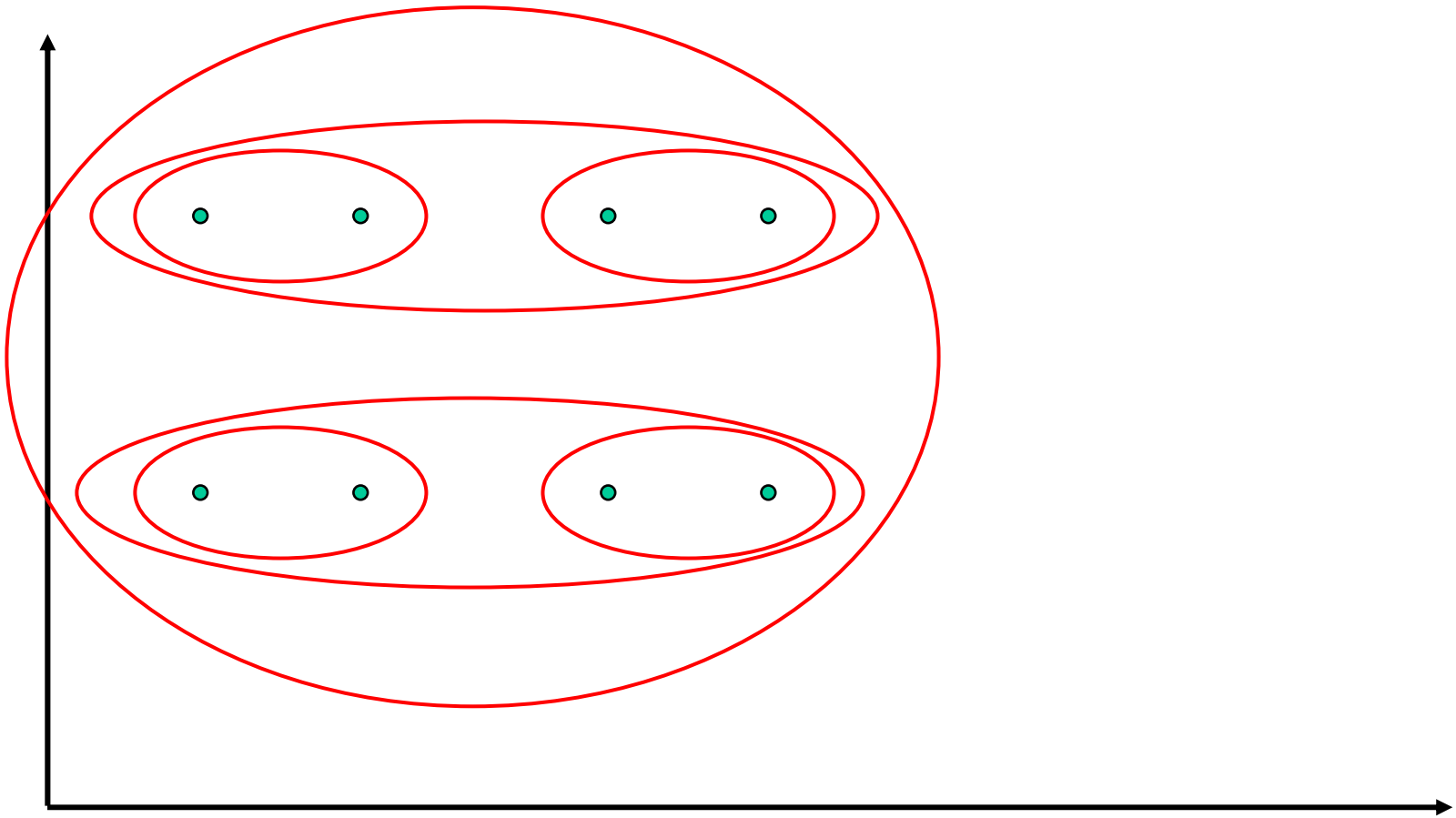
- Use maximum similarity of pairs:

$$sim(c_i, c_j) = \max_{x \in c_i, y \in c_j} sim(x, y)$$

- Can result in "straggly" (long and thin) clusters due to chaining effect.
- After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:

$$sim((c_i \cup c_j), c_k) = \max(sim(c_i, c_k), sim(c_j, c_k))$$

A Single Link Example



Complete Link Agglomerative Clustering

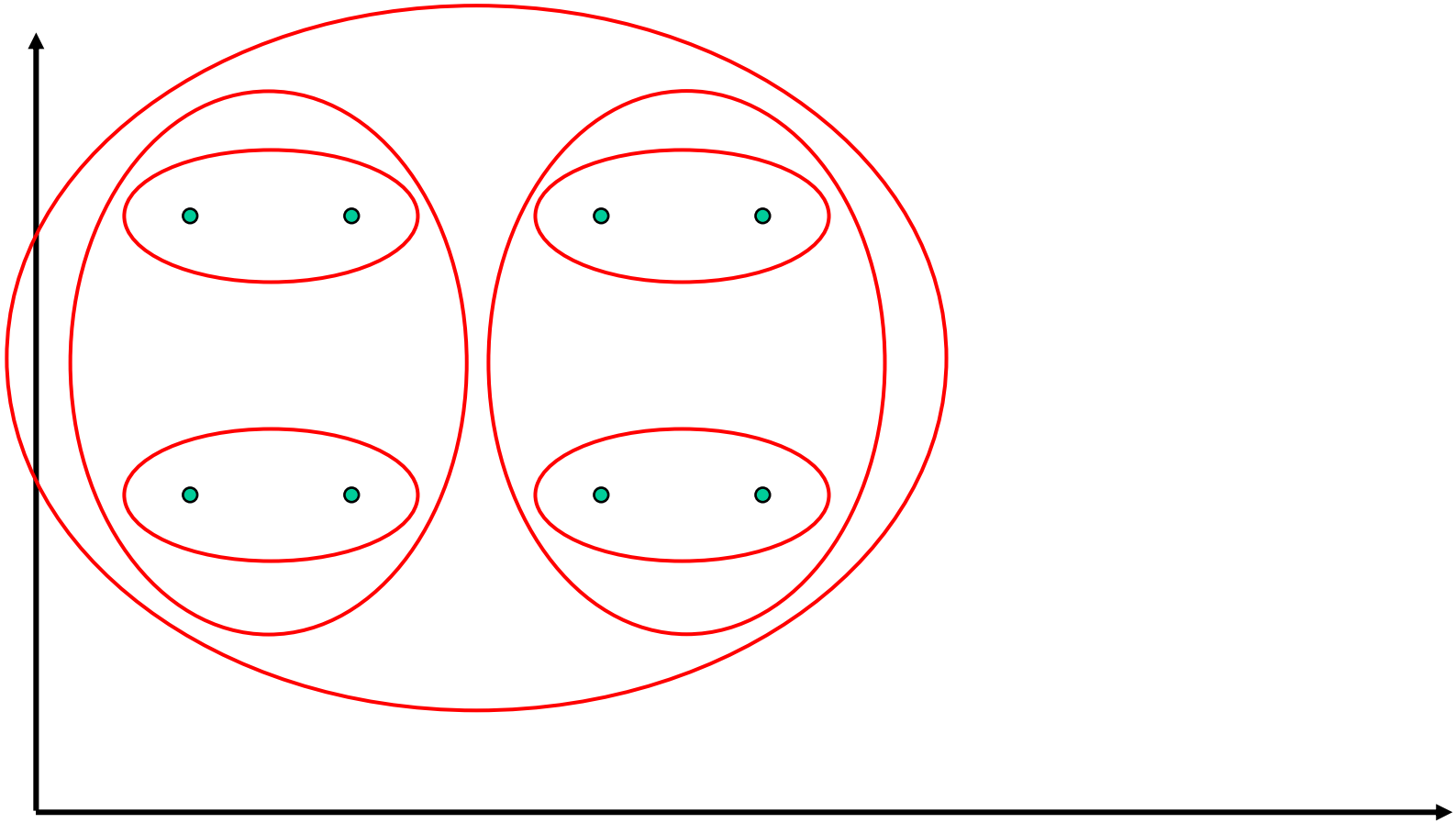
- Use minimum similarity of pairs:

$$sim(c_i, c_j) = \min_{x \in c_i, y \in c_j} sim(x, y)$$

- Makes "tighter," spherical clusters that are typically preferable.
- After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:

$$sim((c_i \cup c_j), c_k) = \min(sim(c_i, c_k), sim(c_j, c_k))$$

A Complete Link Example



Summary

- Today's Class
  - Unsupervised Learning (Chapter 14)
- Next Classes
  - Summary
- Project Presentation: April 21 and 23
- Reading Assignments
  - HTF, Chapter 14