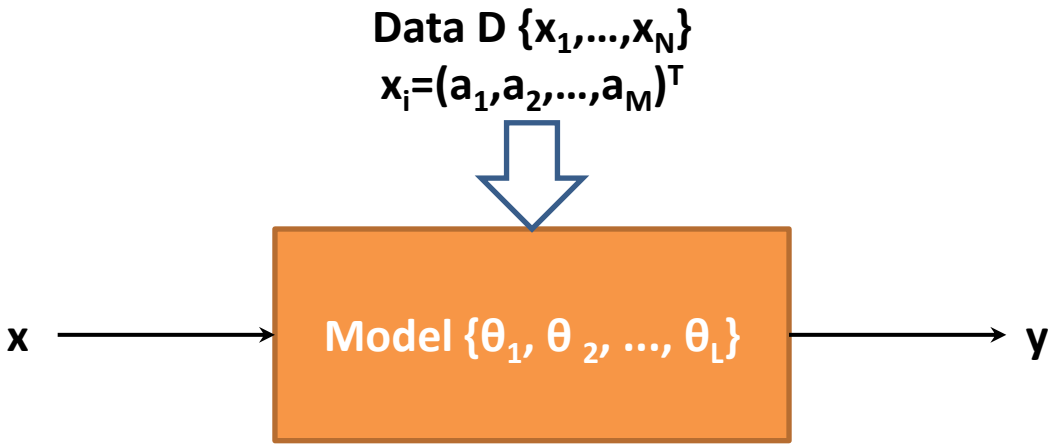
*k*Means (Unsupervised Learning)

Boris Velichkov

Exercises by Week

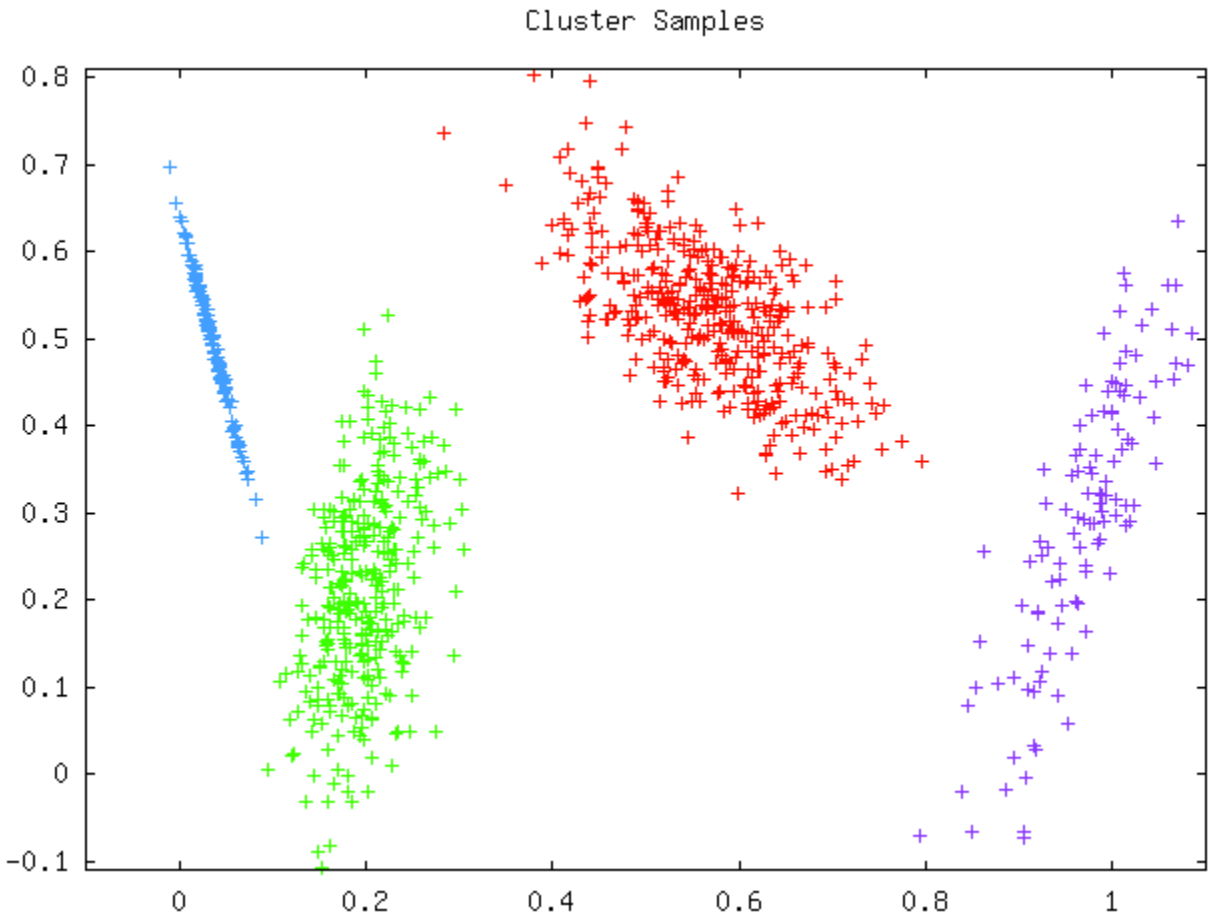
1. Introduction
2. **Uninformed *(Blind)* Search**
3. **Informed *(Heuristic)* Search**
4. **Constraint Satisfaction Problems**
5. **Genetic Algorithms**
6. **Games**
7. Introduction to Machine Learning
8. ***k*-Nearest Neighbors**
9. **Naïve Bayes Classifier**
10. **Decision Tree**
11. ***k*Means**
12. **Neural Networks**
13. *Additional Topics, Questions and Homeworks' Presentations*
14. *Additional Topics, Questions and Homeworks' Presentations*
15. *Additional Topics, Questions and Homeworks' Presentations*

Unsupervised Learning

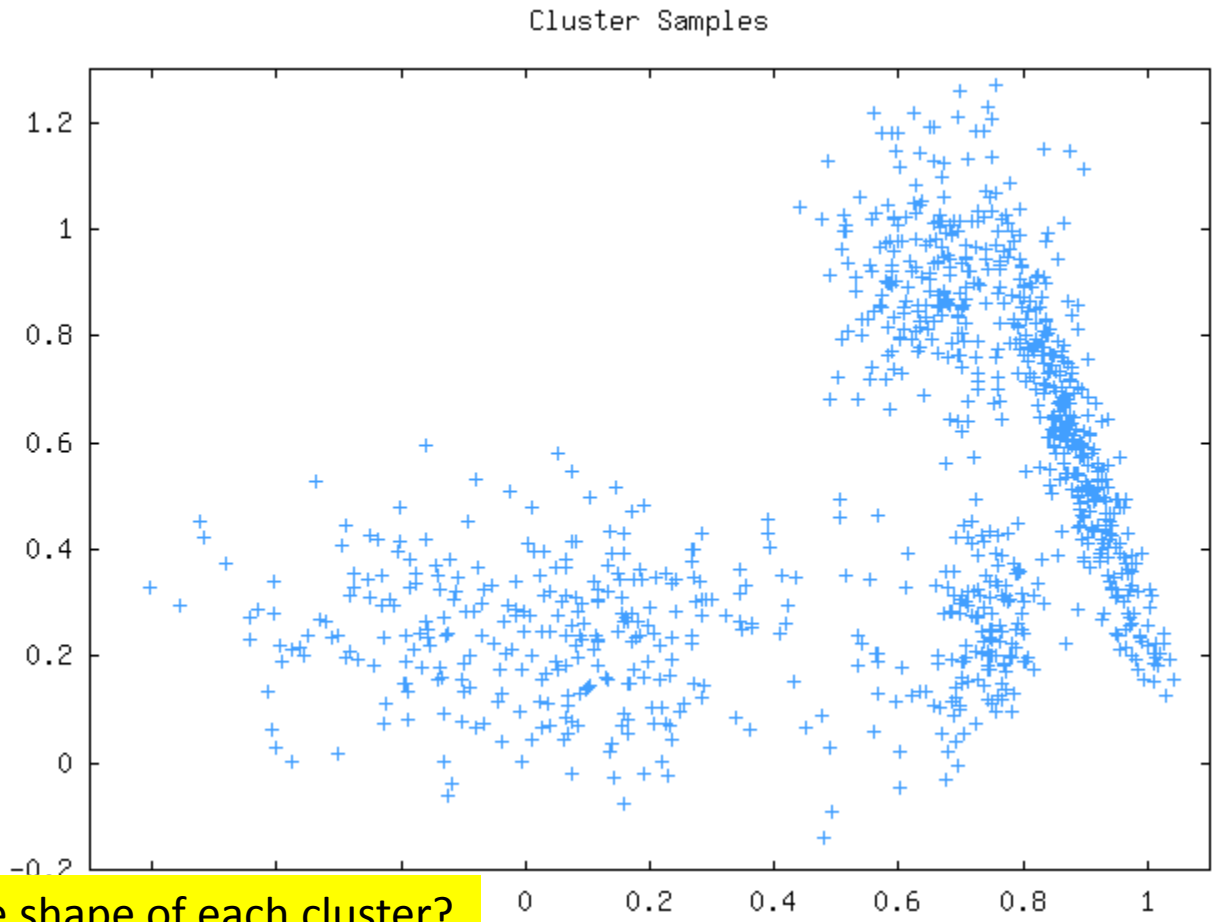


Clustering: y categorical
Dimensionality Reduction:
$x \in \Re^M \Rightarrow y \in \Re^K$, with K<M

Clustering

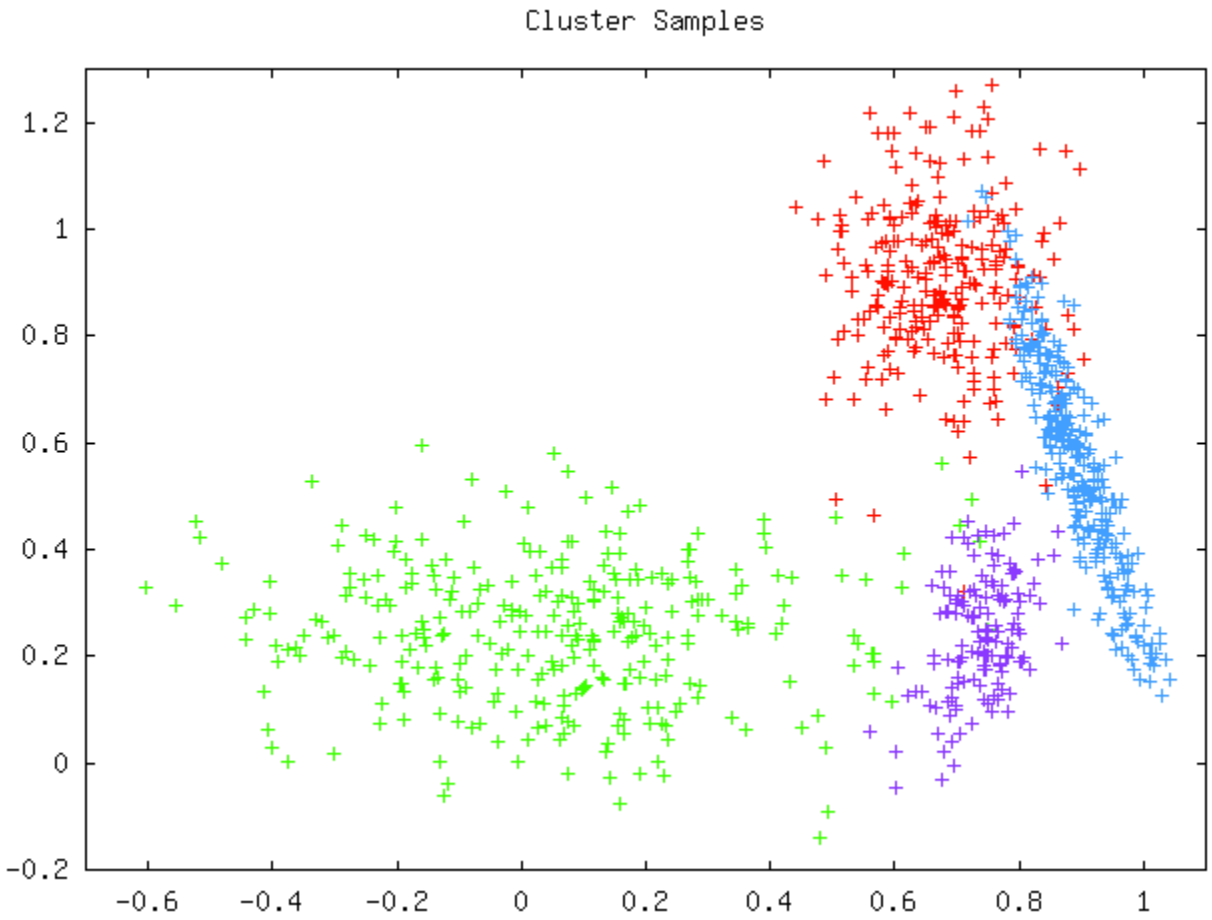


Where Are the Clusters?



- What is the shape of each cluster?
- How many clusters?

Clustering



The Clustering Problem

- Given a set of data samples $x_1,\ldots,x_N$,
- Assign the data to K clusters
  - Partitioning the dataset
  - Also called segmentation
- K may be given, or chosen automatically

- Techniques fall into:
  - Combinatorial techniques: work directly on data
  - Mixture modeling: Assume data is IID, and models underlying pdf
  - Mode seeking: aka bump hunting

Clustering Techniques

- We'll focus on the following
  - K-means
  - Gaussian Mixture modeling (Also called soft K-means)
  - Hierarchical clustering (Agglomerative/divisive) clustering

- These techniques are used regularly, often as part of a much larger system that might include supervised learning
  - e.g. discretize continuous input to make classification easier
  - e.g. Representing pdf for Bayes classifier

K-Means

- Wonderfully simple algorithm
- K-means:
  - Initialize cluster centers
  - Repeat until done
    - Assign each data point to nearest cluster center
    - Replace each cluster center with the mean of the data points associated to it

K-Means Concepts

- Let's assume the data is 2-D, and was generated from K clusters
- We'll model the problem with K *prototype* vectors
  - We'll call these *means*, and you'll see why
- We assign a data point *x* to a cluster based on *distance*
  - Data point *x* is assigned to the *closest* prototype

How do we define distance?

$$y = \arg\min_{k=1\ldots K} Distance(x, m_k)$$

Prototypes

Note, this is similar to nearest neighbor classification and regression methods, which we will come back to

K-Means Update

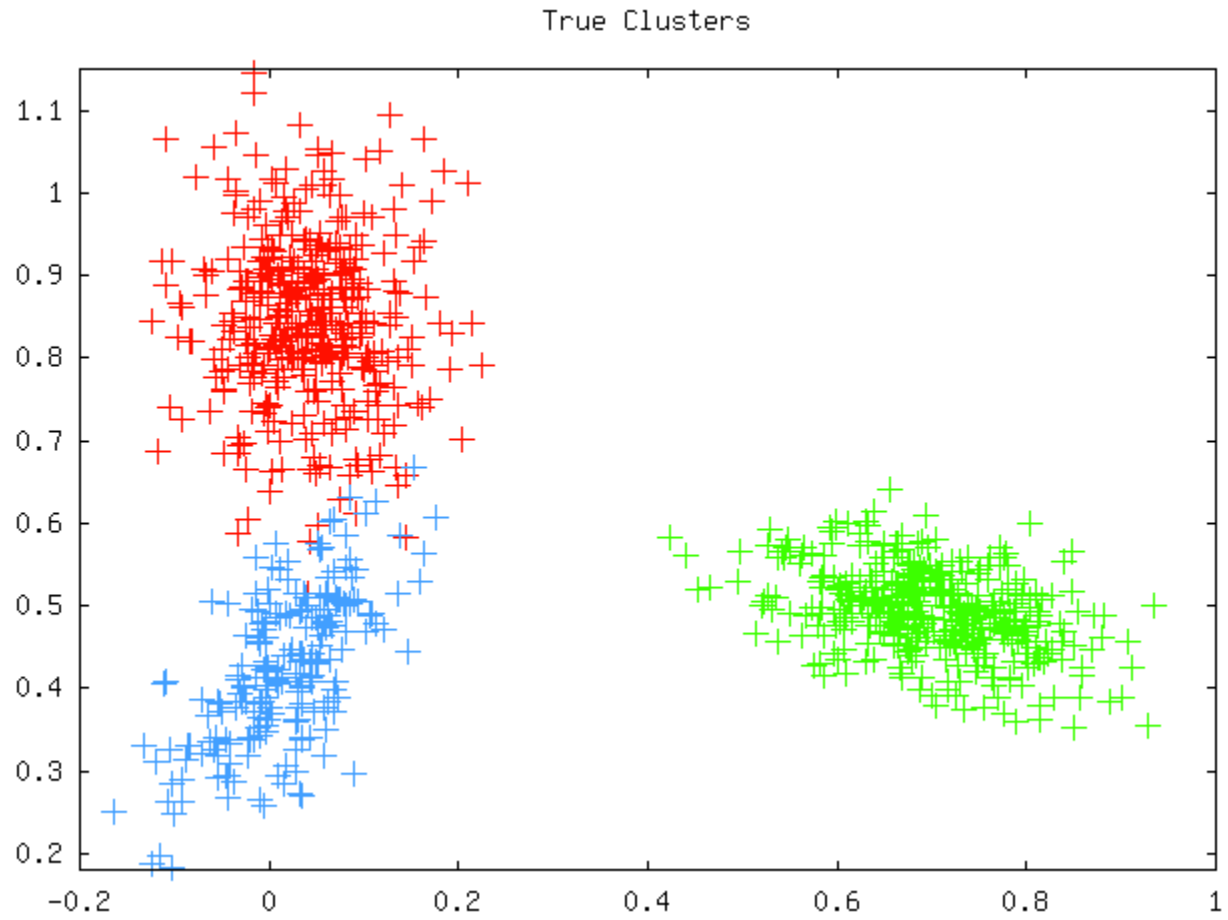
- We *update* our prototypes based on the points that were assigned to it, but taking the average/centroid/mean

$$m'_k = \frac{1}{N_k} \sum_{x_i \text{ assigned to } k} x_i \qquad k = 1 \ldots K$$

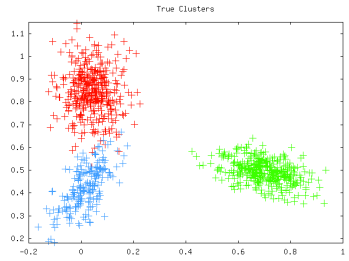
New prototype

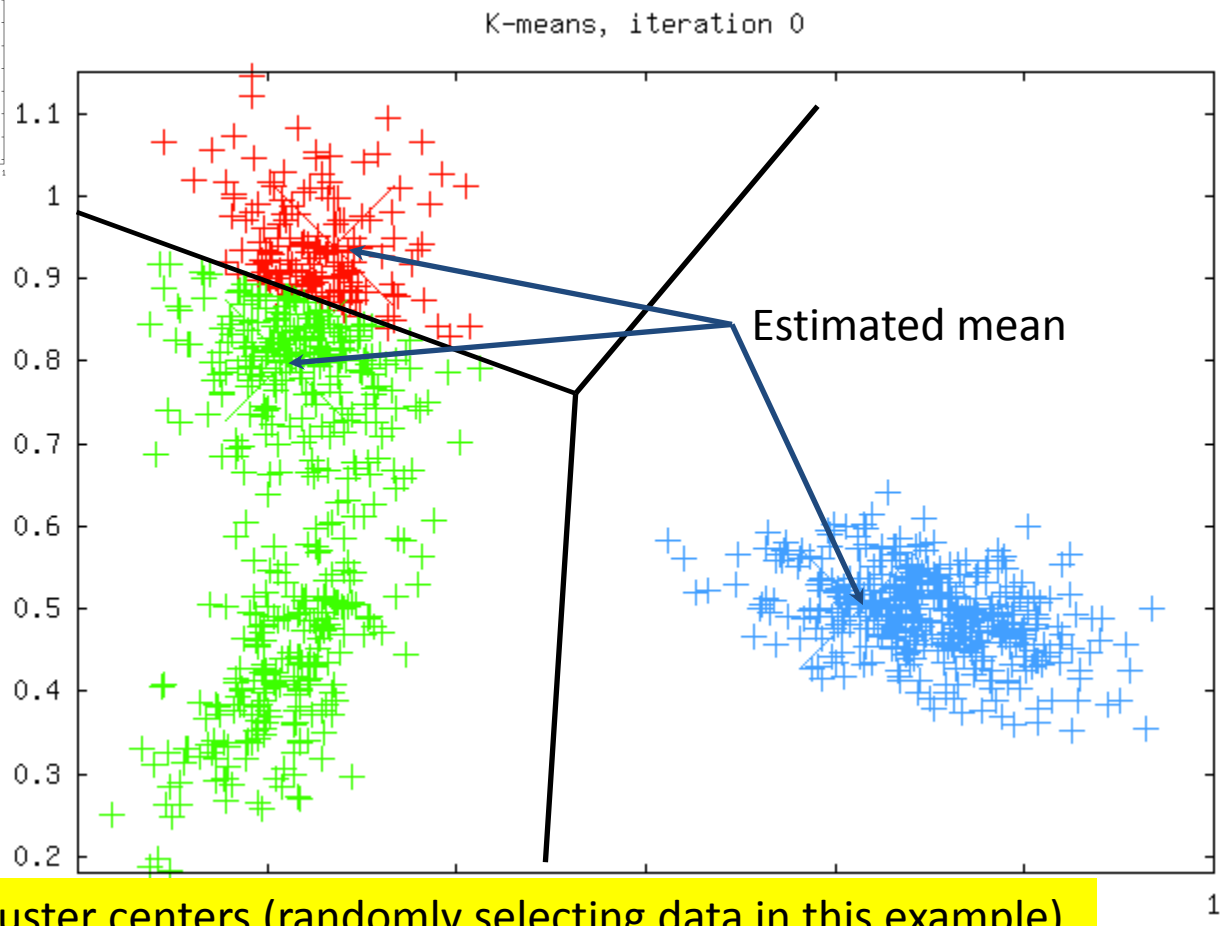
Mean of points assigned to *k*

Example



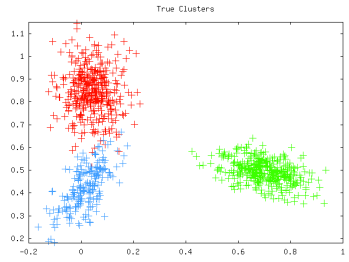
Example: Initialization

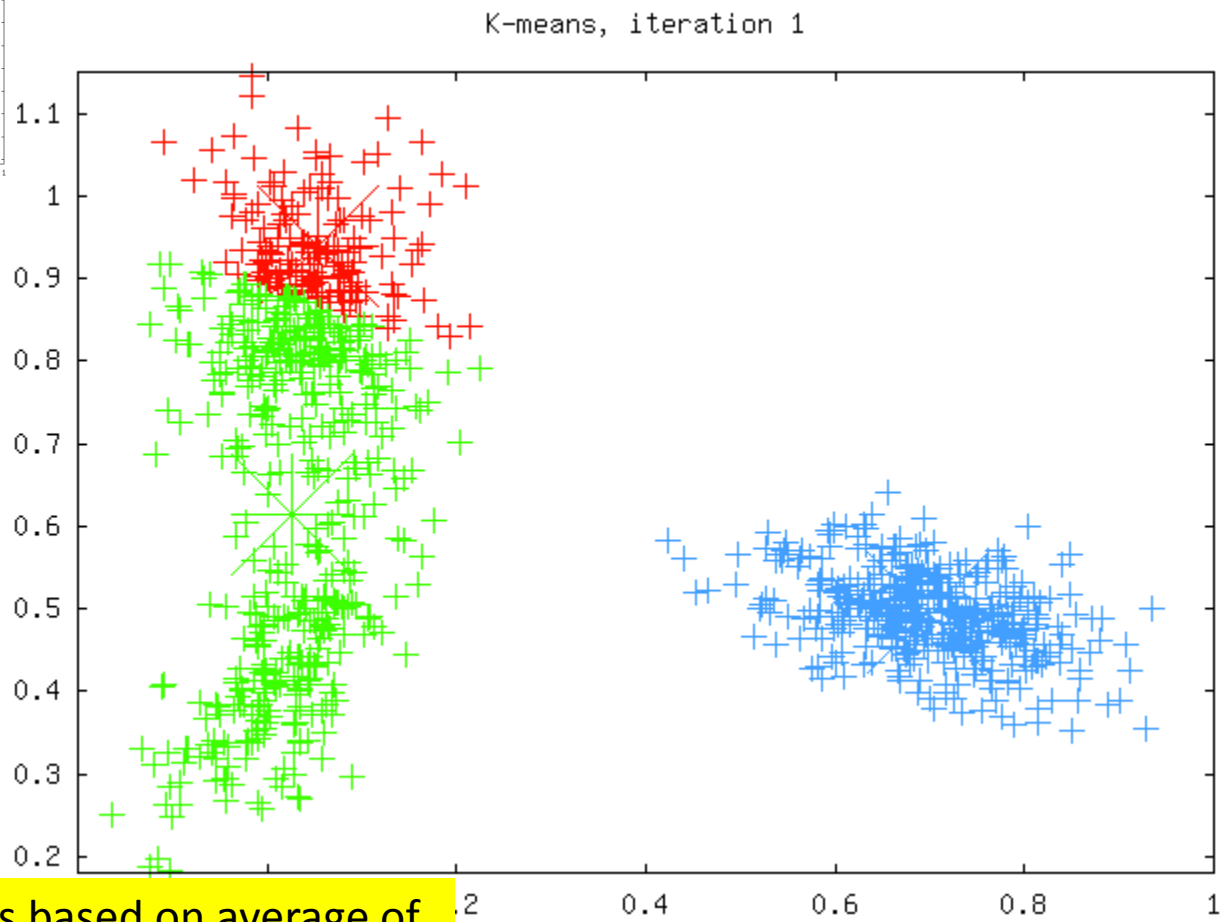




- Initialize cluster centers (randomly selecting data in this example)
- Assign to cluster centers based on nearest prototype mean

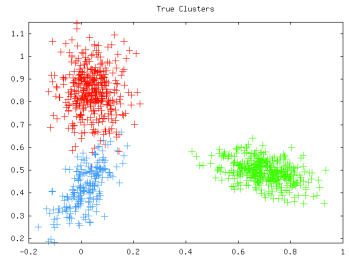
Example: Iteration 1

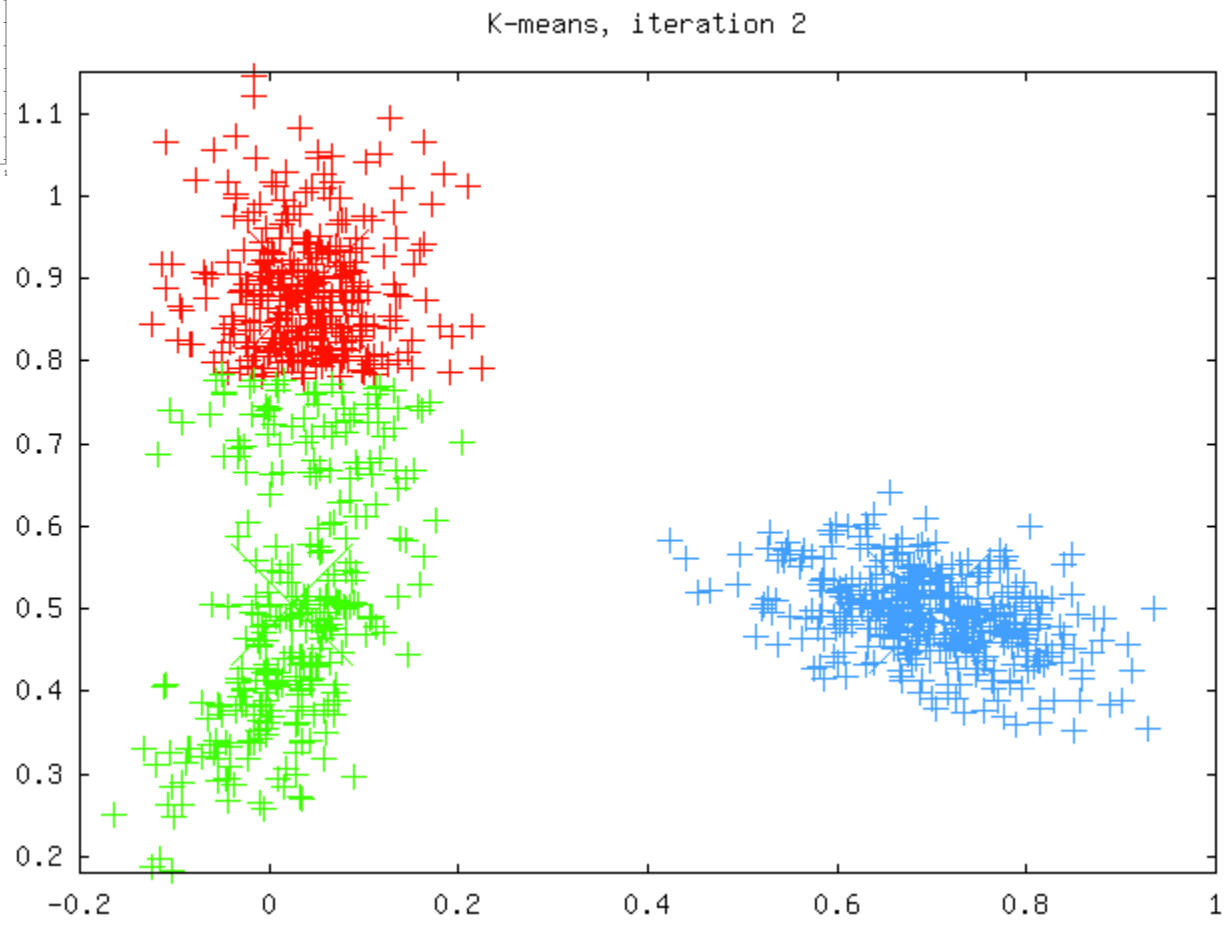




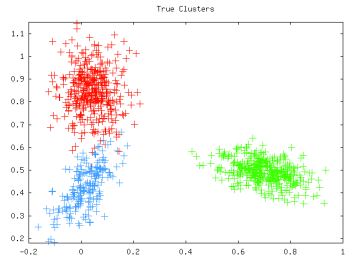
Update means based on average of points assigned to prototype

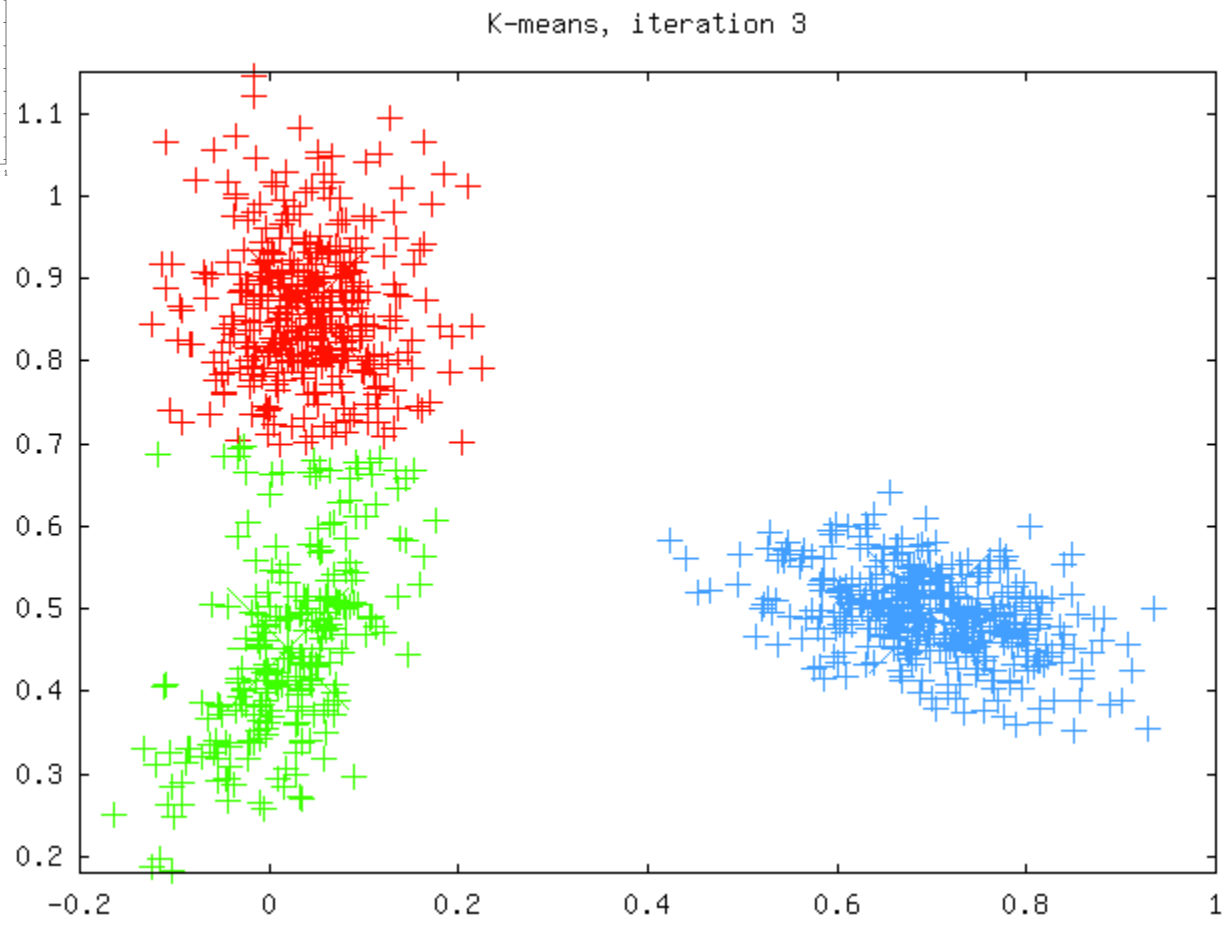
Example: Iteration 2



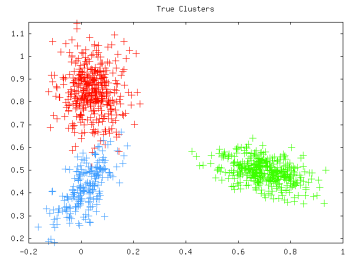


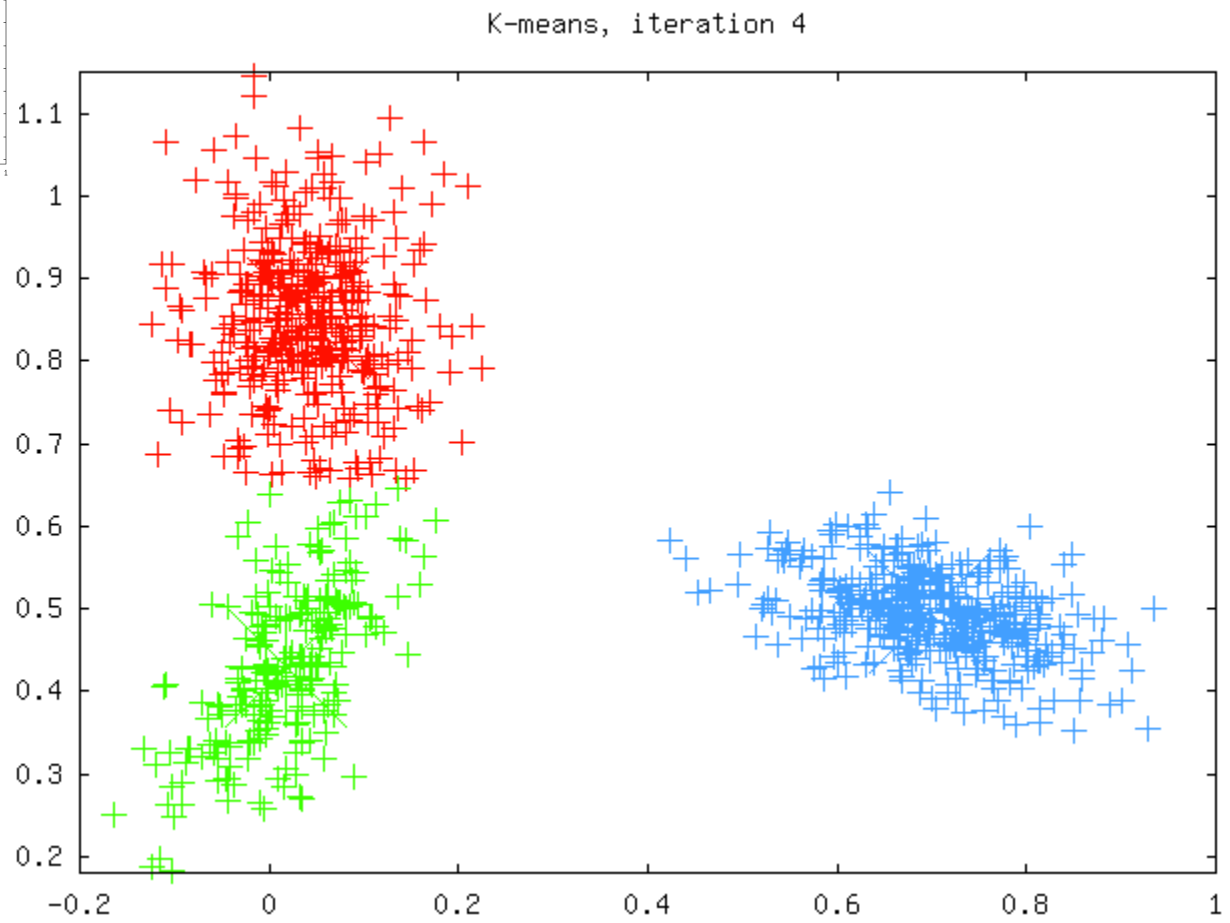
Example: Iteration 3



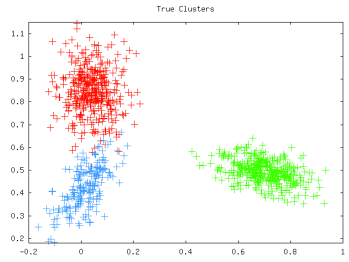


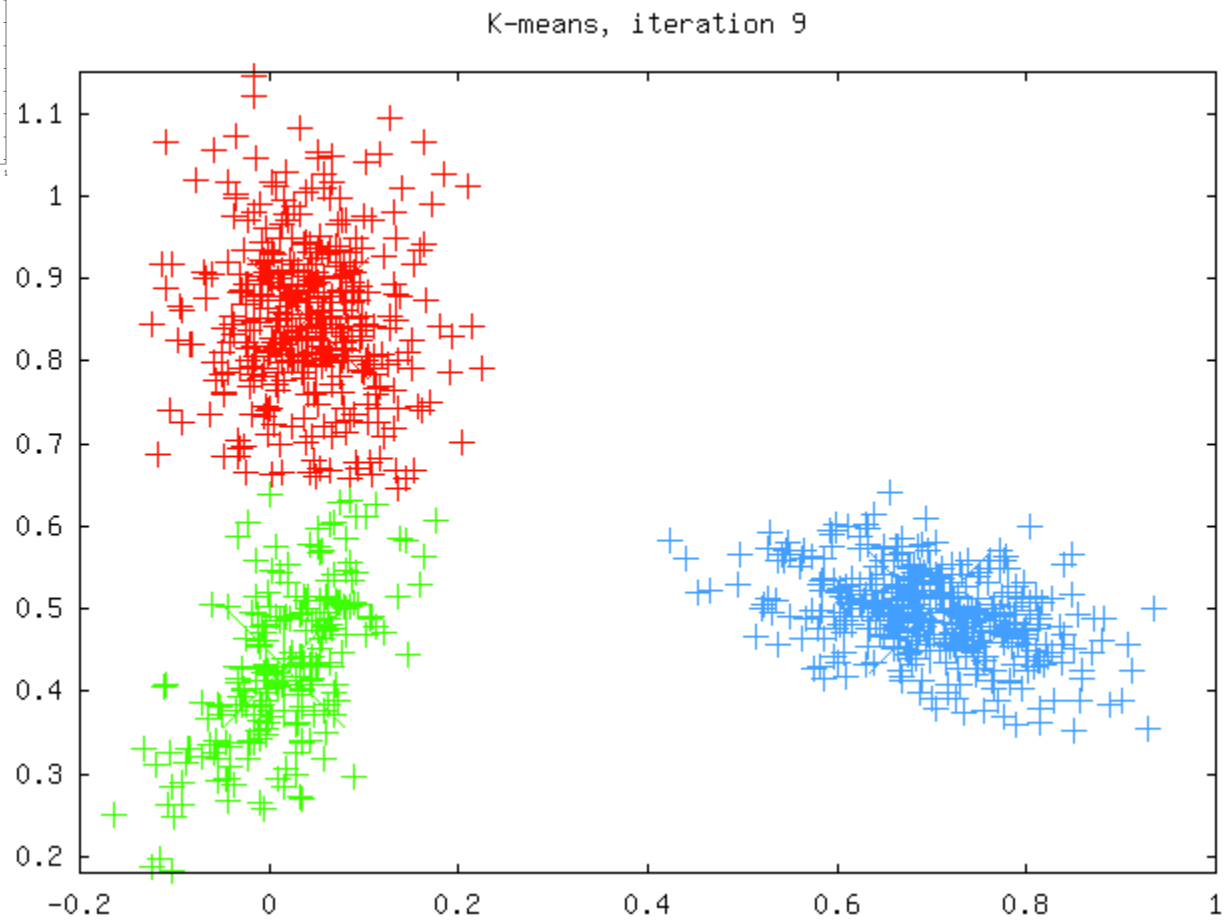
Example: Iteration 4





Example: Iteration 9





K-Means

K-means clustering: how it works

Question Time

- How well does it fit the data?
- When should we terminate? Will it always terminate?
- Does it always work?
- How do we tell how many clusters are there (ie. what is *K*)?

How Well Does It Fit The Data?

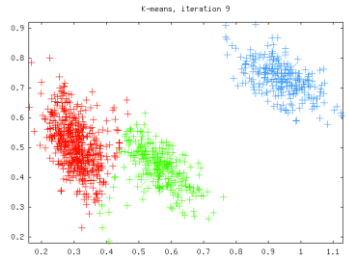
- K-means is a local search technique for optimizing the distortion of the data

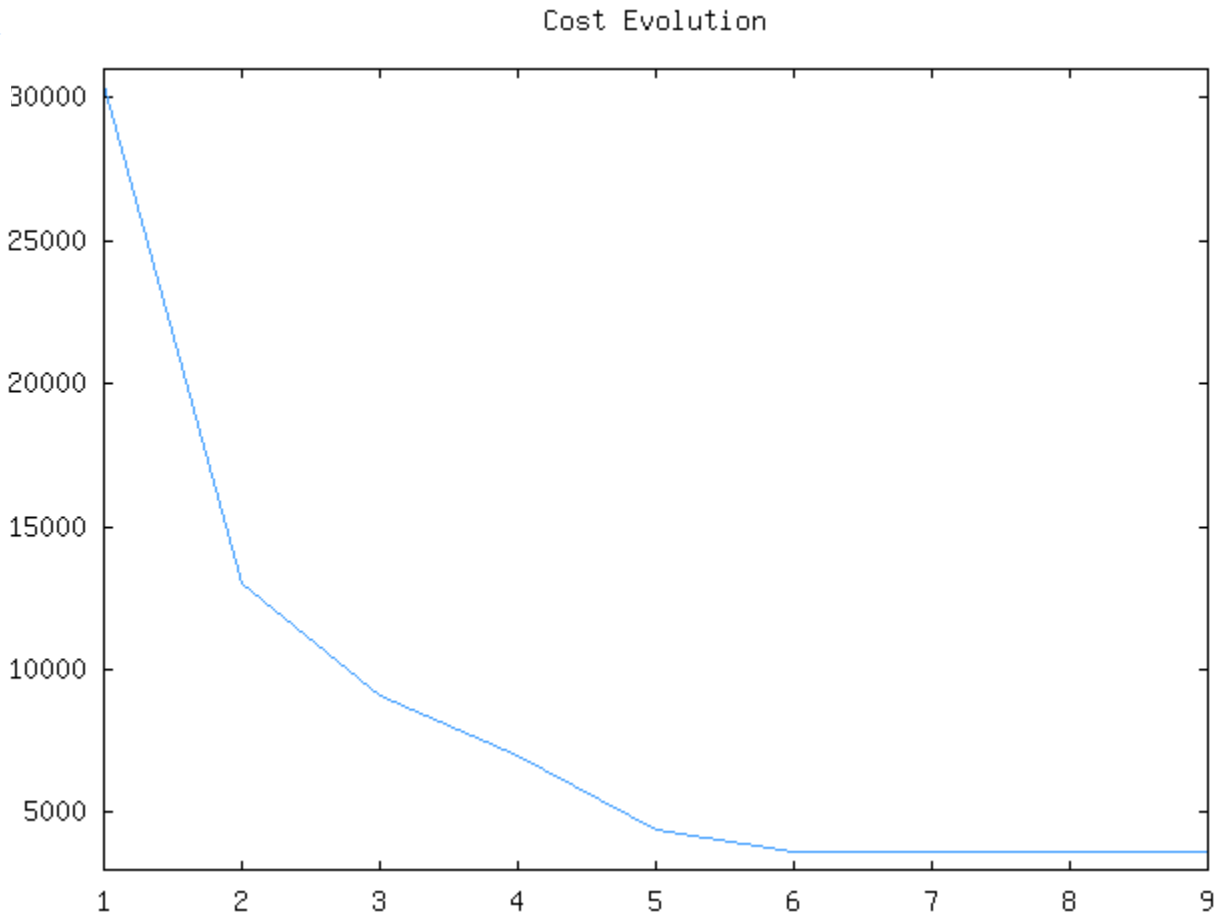
- Formally, K-means tries to optimize the within-point scatter

$$C = \sum_{k} N_k \sum_{y_i=k} \|x_i - m_k\|^2$$

How does this change with each iteration?

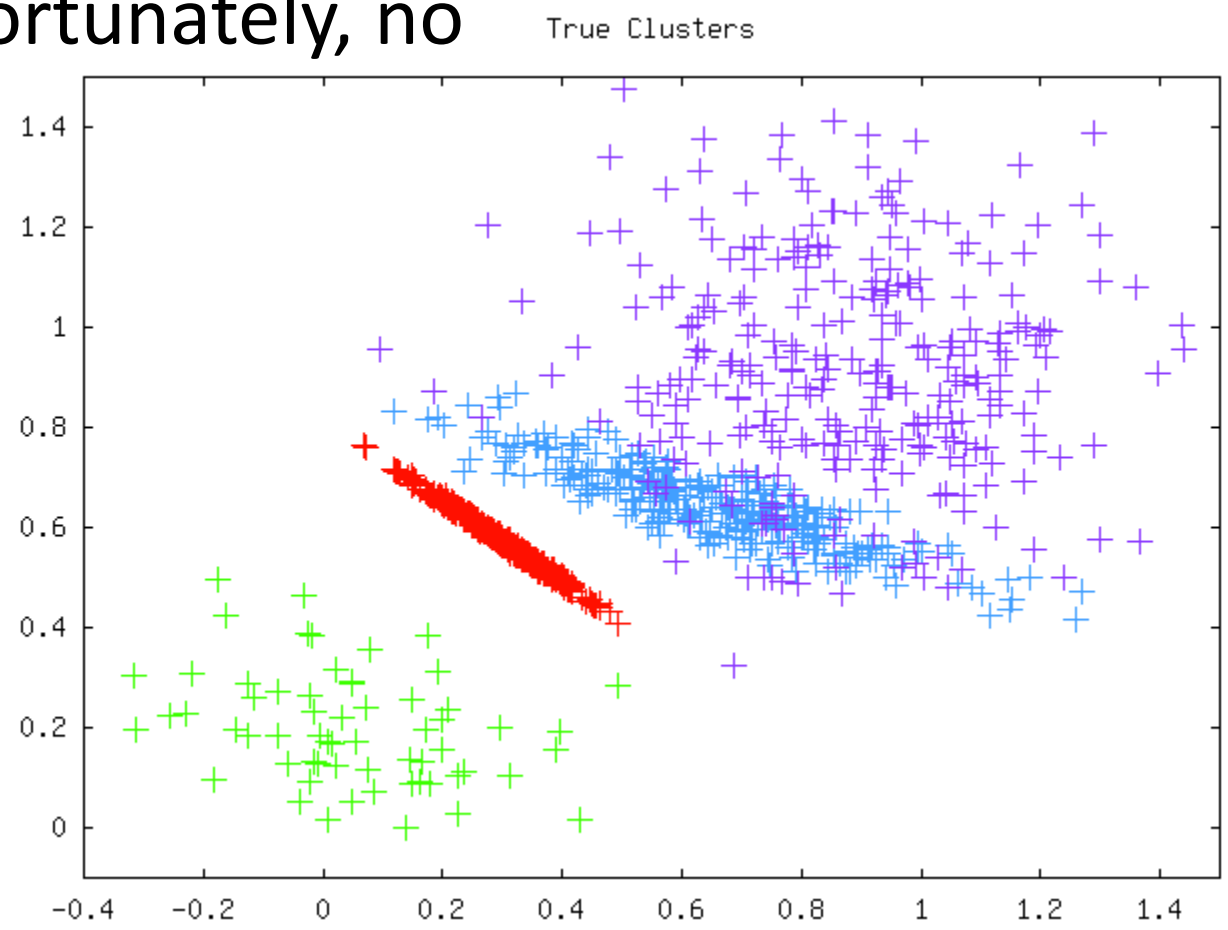
From Previous Example



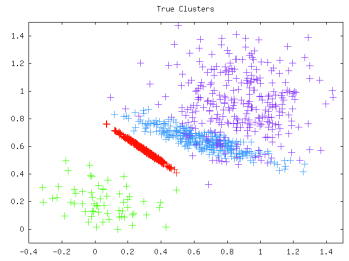


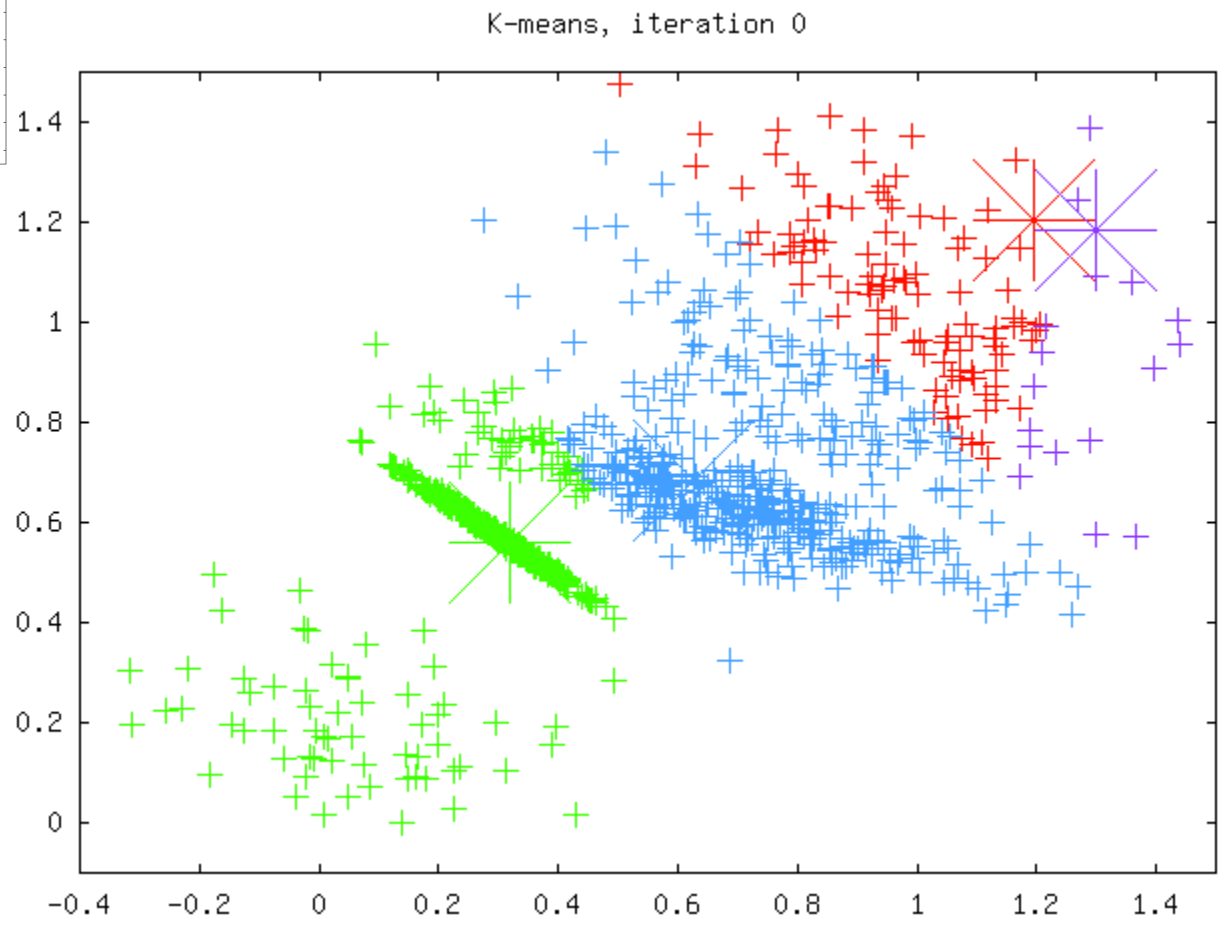
Does It Always Work?

- Unfortunately, no

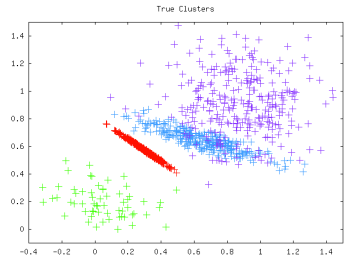


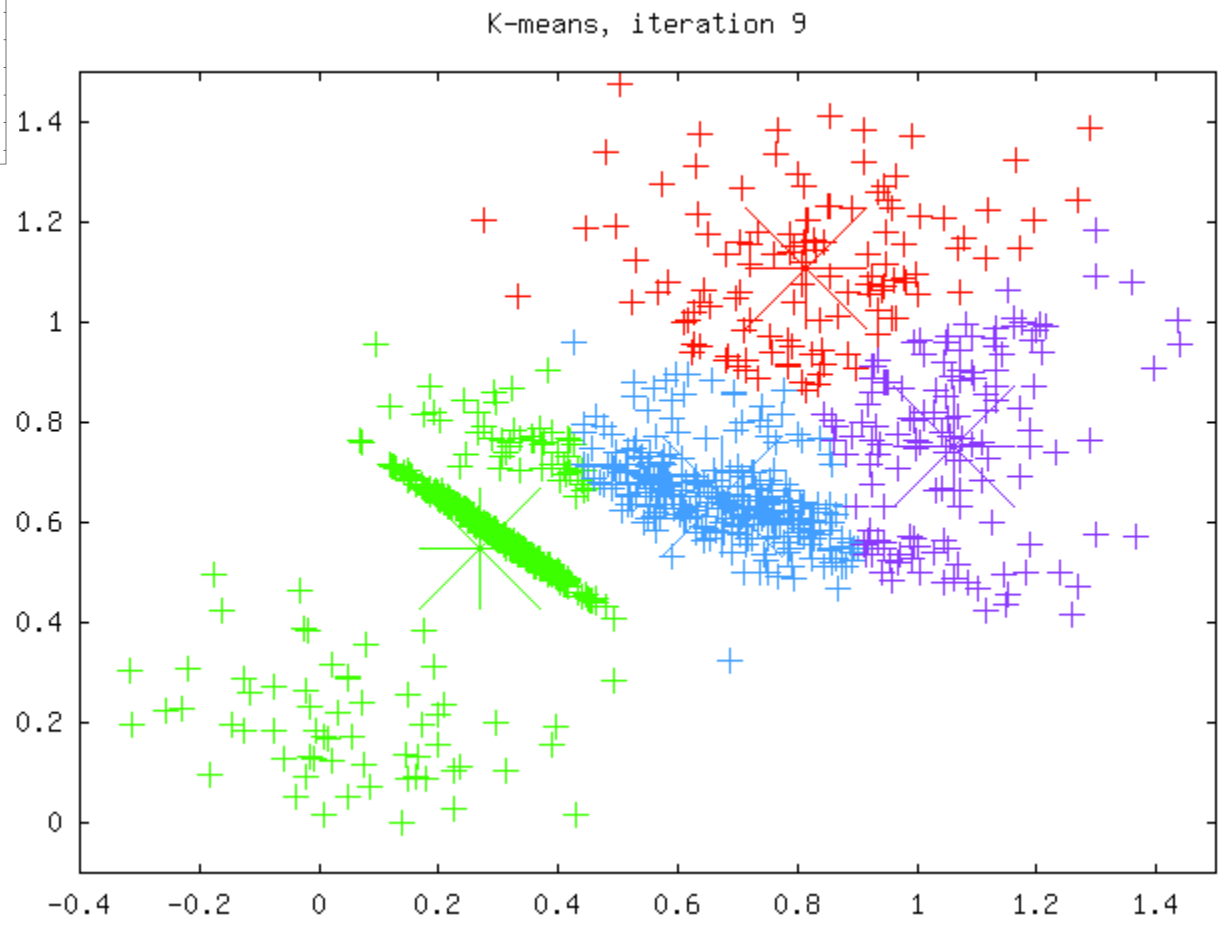
Does It Always Work?





After 9 Iterations

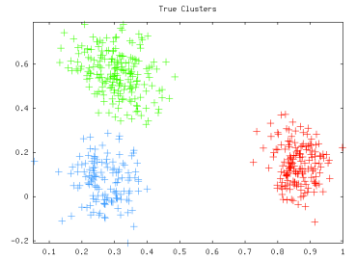


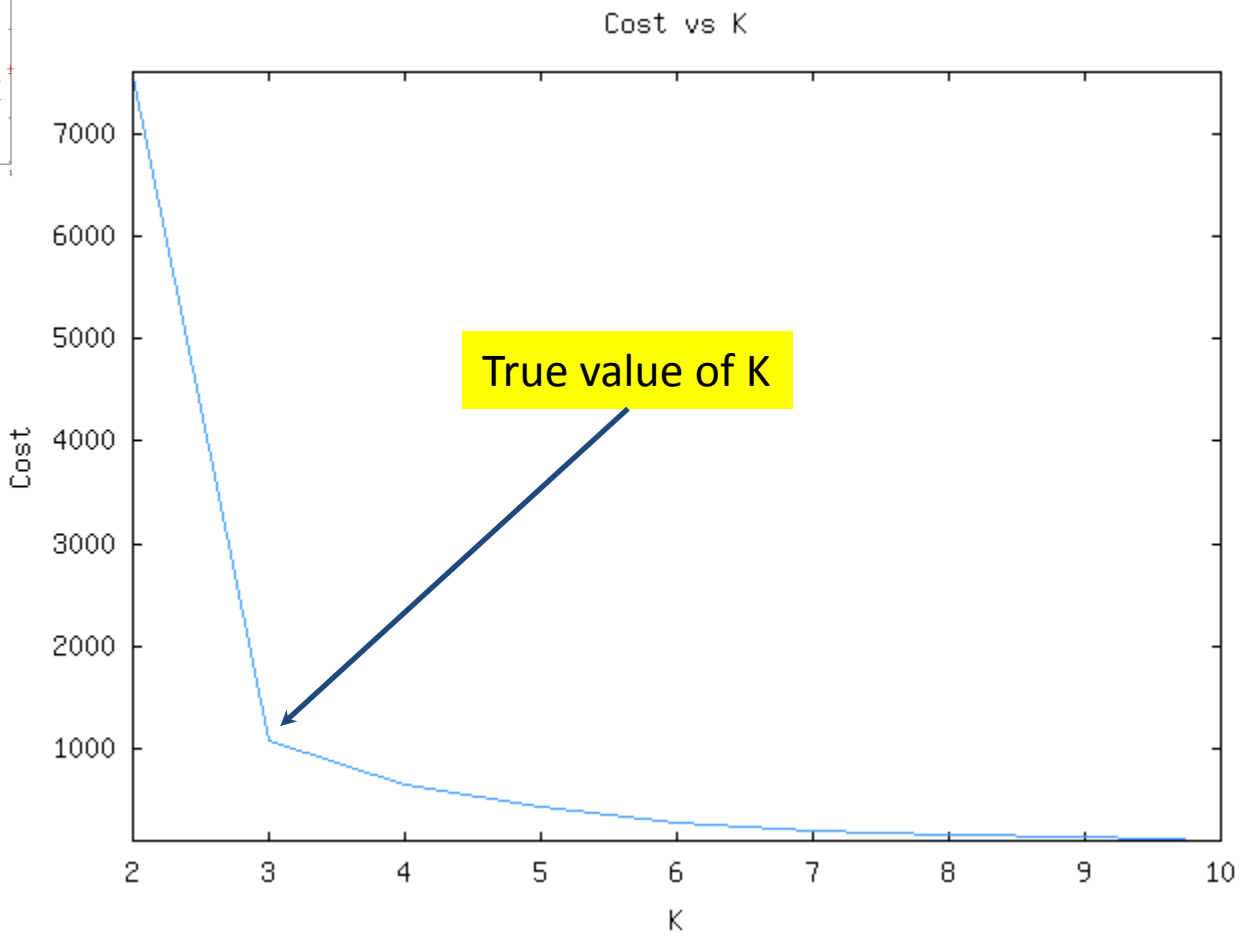


What Happened?

- K-means can get stuck in local optima
  - Effectively, it will depend on the starting condition
- How can we “fix” this?
  - Use random restarts (remember local search?)
  - Keep track of best solution so far
- K-means will converge
  - May want to limit iterations though

How Many Clusters?





K-Means Summary

- Practical algorithm, good to have in the tool box
- Implementation
  - Need to run with random restarts
  - Need to keep track of best solution found
  - Need to provide (or estimate) K
  - Can be slow on big datasets
- Can use different distance metrics
  - Part of algorithm design
- Speeding up K-means
  - Faster nearest neighbor algorithms/data structures

K-Means Variants

- K-Means
  - If your distance is **squared Euclidean distance**, use ***k*-means**
    - k-means minimizes within-cluster variance, which equals squared Euclidean distances.
    - In general, the arithmetic mean does this. It does not optimize distances, but squared deviations from the mean.
- K-Medians
  - If your distance is **Taxicab metric**, use ***k*-medians**
    - k-medians minimizes absolute deviations, which equals Manhattan distance.
- K-Medoids
  - If you have **any other distance**, use ***k*-medoids**

K-Means Variants

- K-Modes
  - KMeans uses mathematical measures (distance) to cluster continuous data. The lesser the distance, the more similar our data points are. Centroids are updated by Means.
  - KModes uses the dissimilarities (total mismatches) to cluster categorical data. The lesser the dissimilarities the more similar our data points are. It uses modes instead of means.
- K-Means++
  - For finding initial centroids K-Means clustering uses randomization. The initial k-centroids are picked randomly from the data points. This randomization affects the final formed clusters. There are two approaches to avoid this problem:
    - Repeat K-means: Repeating the algorithm and initialization of centroids several times and pick the clustering approach that has small intracluster distance and large intercluster distance (random restart).
    - K-Means++: K-Means++ is a smart centroid initialization technique.
- Soft K-Means
  - K means
    - Hard assign a data point to one particular cluster on convergence.
    - It makes use of the L2 norm when optimizing (Min {Theta} L2 norm point and its centroid coordinates).
  - EM
    - Soft assigns a point to clusters (so it give a probability of any point belonging to any centroid).
    - It doesn't depend on the L2 norm, but is based on the Expectation, i.e., the probability of the point belonging to a particular cluster. This makes K-means biased towards spherical clusters.

K-Means++

- K-Means++ is a smart centroid initialization technique and the rest of the algorithm is the same as that of K-Means. The steps to follow for centroid initialization are:
  - Pick the first centroid point ($\boldsymbol{C_1}$) randomly.
  - Compute distance of all points in the dataset from the selected centroid. The distance of $\boldsymbol{x_i}$ point from the farthest centroid can be computed by

$$d_i = \max_{(j:1 \to m)} \left\| x_i - C_j \right\|^2$$

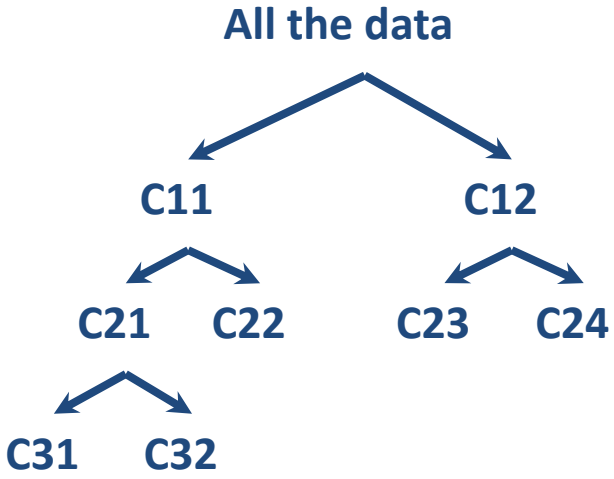
d_i: Distance of x_i point from the farthest centroid
m: number of centroids already picked

  - Make the point $\boldsymbol{x_i}$ as the new centroid that is having maximum probability proportional to $\boldsymbol{d_i}$.
  - Repeat the above two steps till you find k-centroids.

Soft k-Means Clustering

Hierarchical Methods

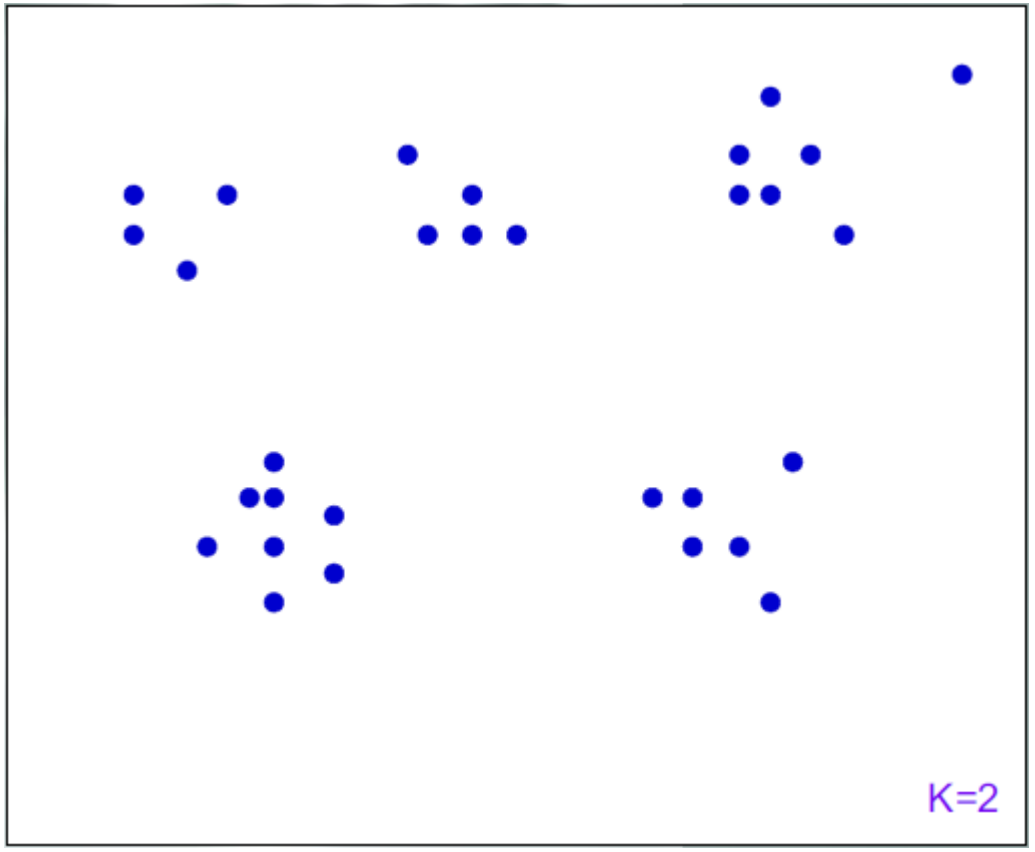
- Recall K-means
  - Input = K, measure of dissimilarity (distances)
  - Output = Cluster centers
- Hierarchical techniques avoid needing to specify K
  - Input = Measure of dissimilarity (e.g. distances)
  - Output = Hierarchical model of data similarity
- Output is a tree (dendogram)



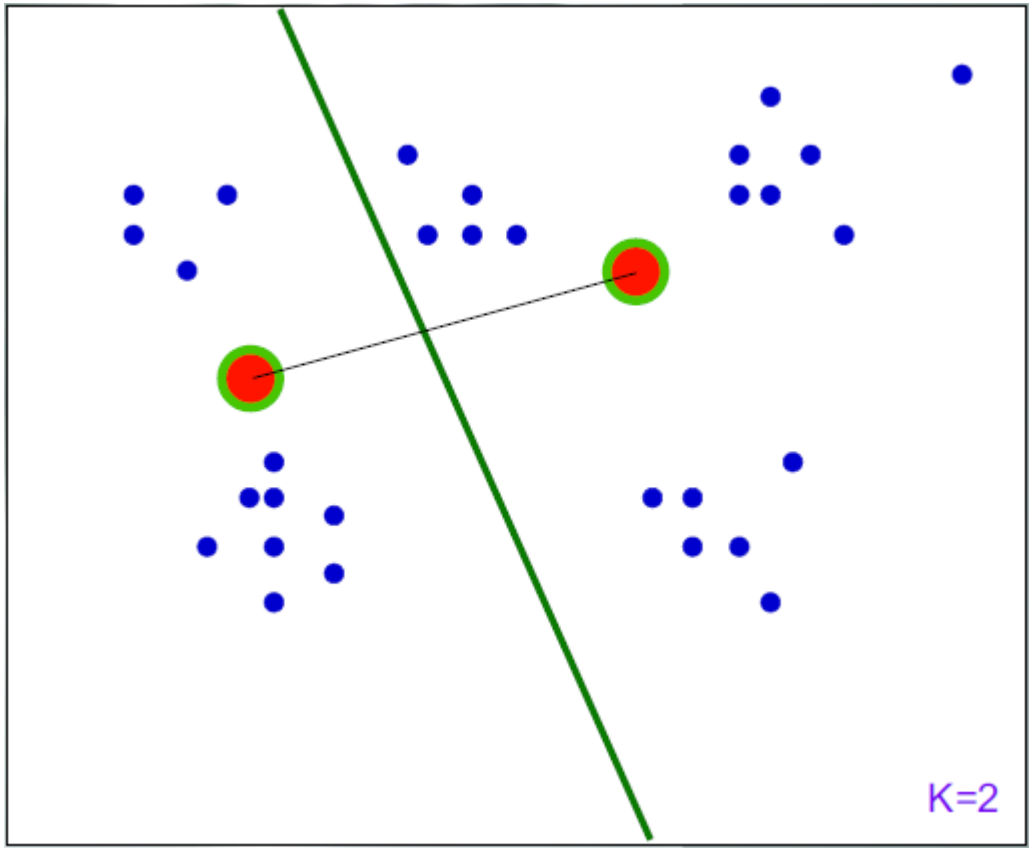
Divisive Methods

- Two types of hierarchical clustering:
  - Divisive (top down), and agglomerative (bottom up)
- Hierarchical K-means is a divisive method
  - Start with all the data in 1 cluster
  - Split using “flat” K-means
  - For each cluster, recursively split each cluster
- K is usually small
- Need to decide when to stop

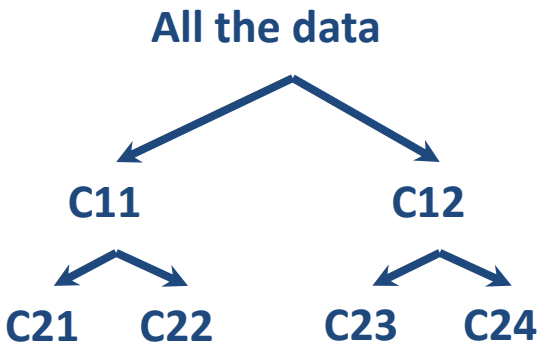
Hierarchical K-Means

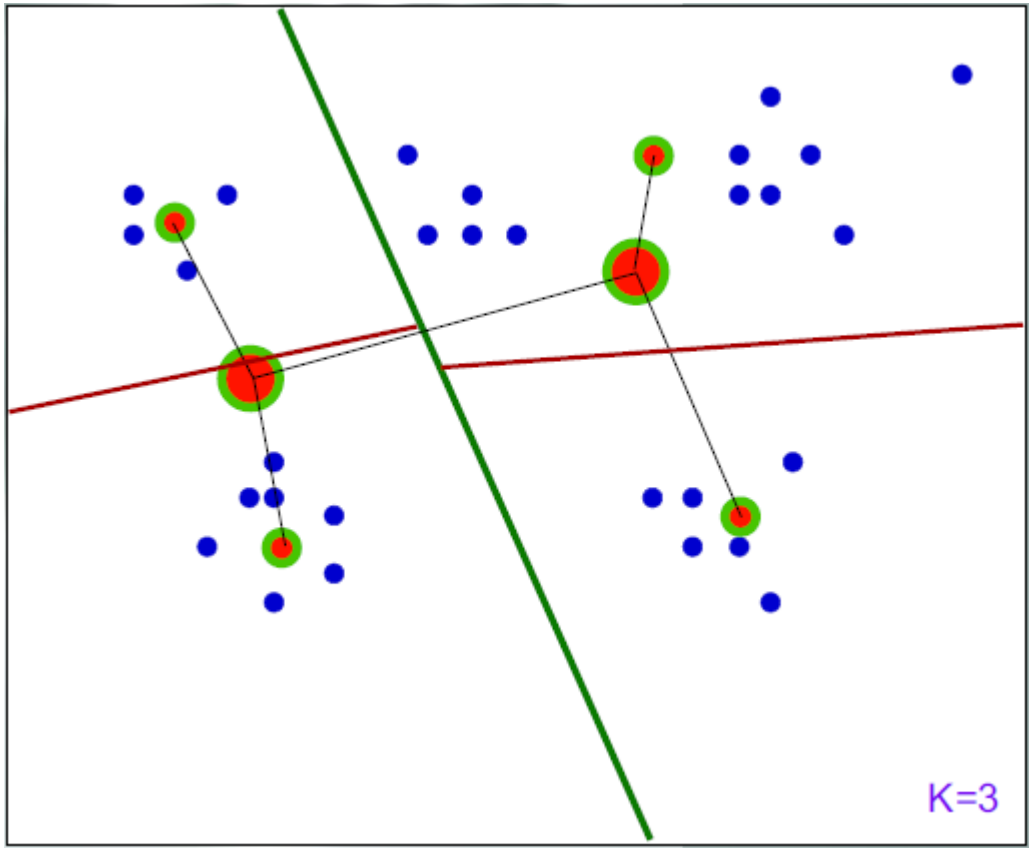


Hierarchical K-Means

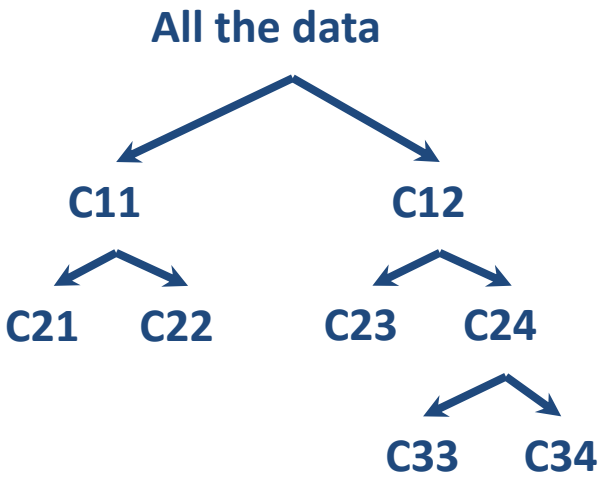


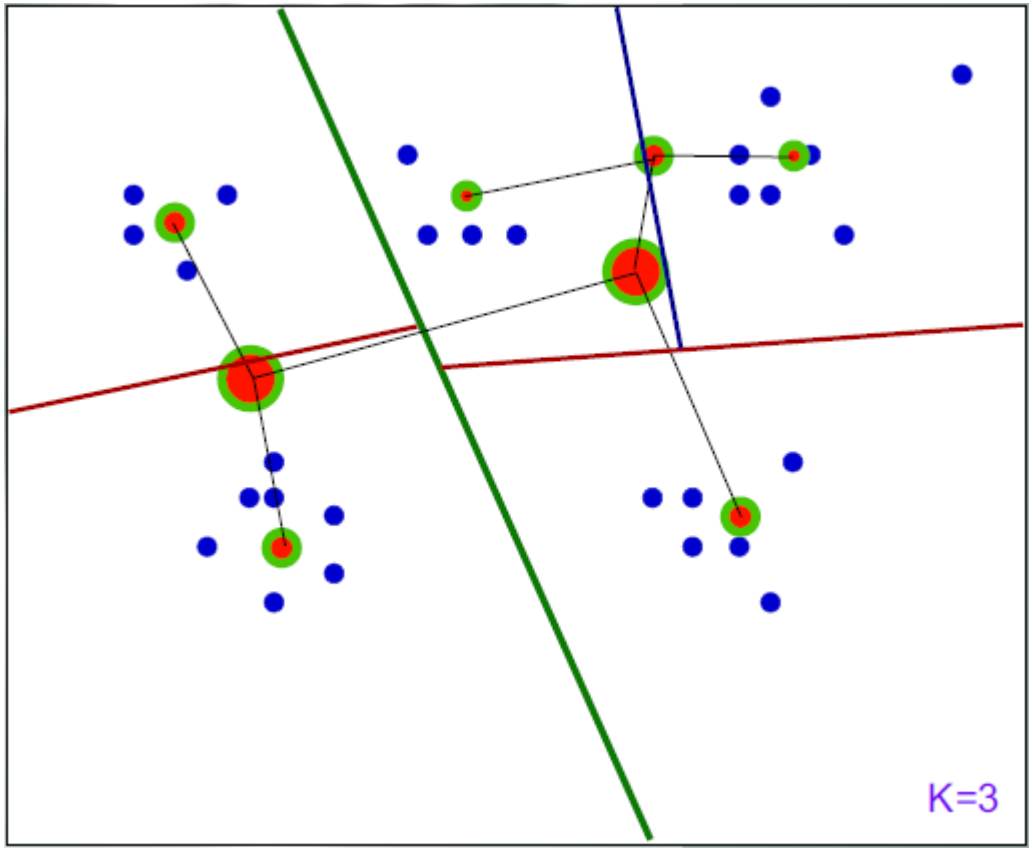
Hierarchical K-Means





Hierarchical K-Means





Agglomerative Techniques

- Work in reverse direction (bottom up)
- Given N data points and dissimilarity measure
  - Start with all the data in separate classes
  - Repeat N-1 times
    - Find *closest* two groups and merge them

- How do we measure dissimilarity between groups?

Agglomerative Clustering

- Define dissimilarity between two pairs of data *d*
- Distance between two groups G_1 and G_2
- Single linkage (SL)

$$d_{SL}(G_1, G_2) = \min_{i \in G_1, j \in G_2} d_{ij}$$

- Complete linkage (CL)

$$d_{CL}(G_1, G_2) = \max_{i \in G_1, j \in G_2} d_{ij}$$

- Group Average (GA)

$$d_{GA}(G_1, G_2) = \frac{1}{N_{G_1} N_{G_2}} \sum_{i \in G_1} \sum_{j \in G_2} d_{ij}$$

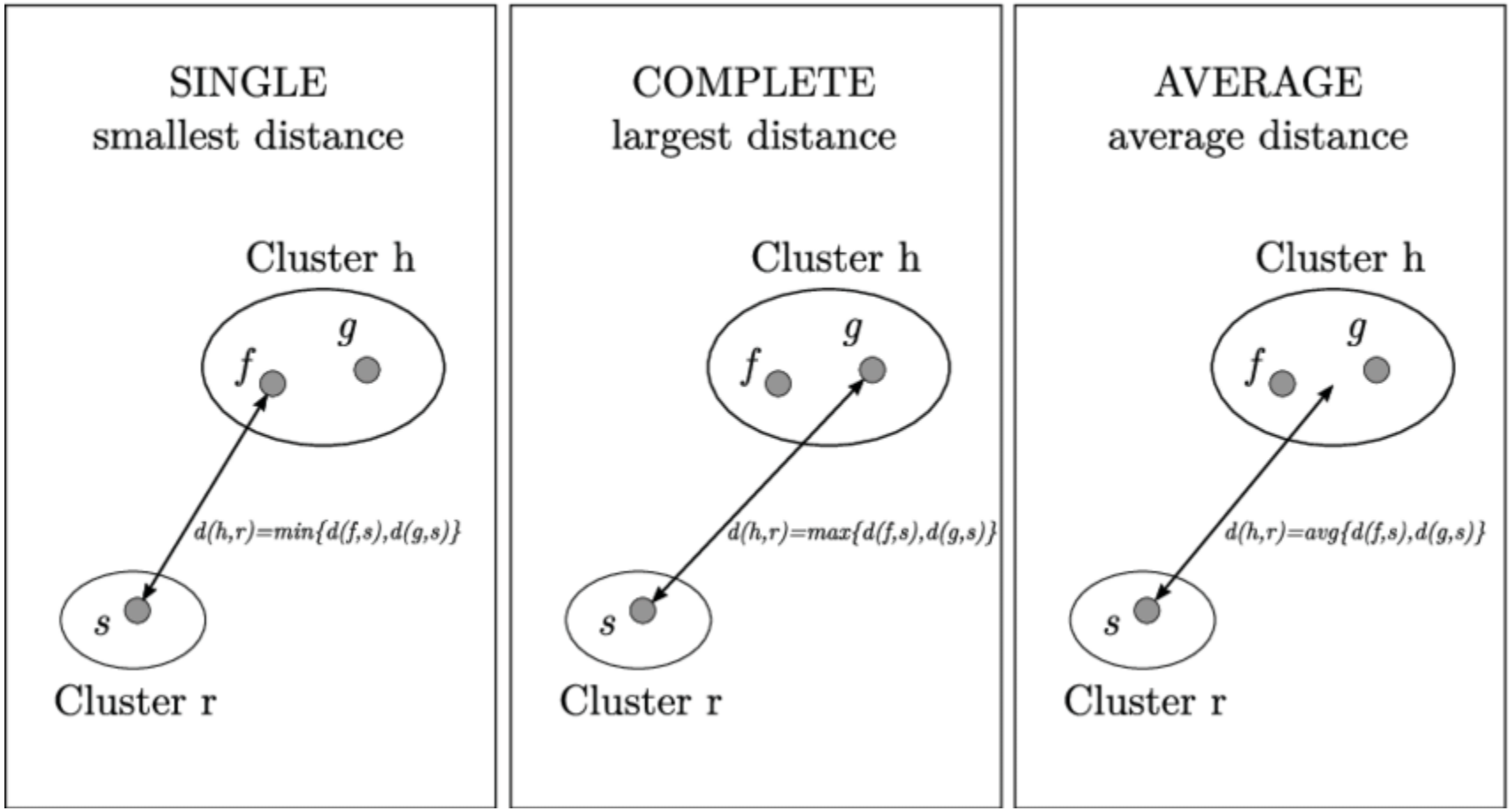
Dissimilarity Measures

- If data is nicely clustered, particular choice doesn't matter
- If data is *not* nicely clustered, you will get different clusters

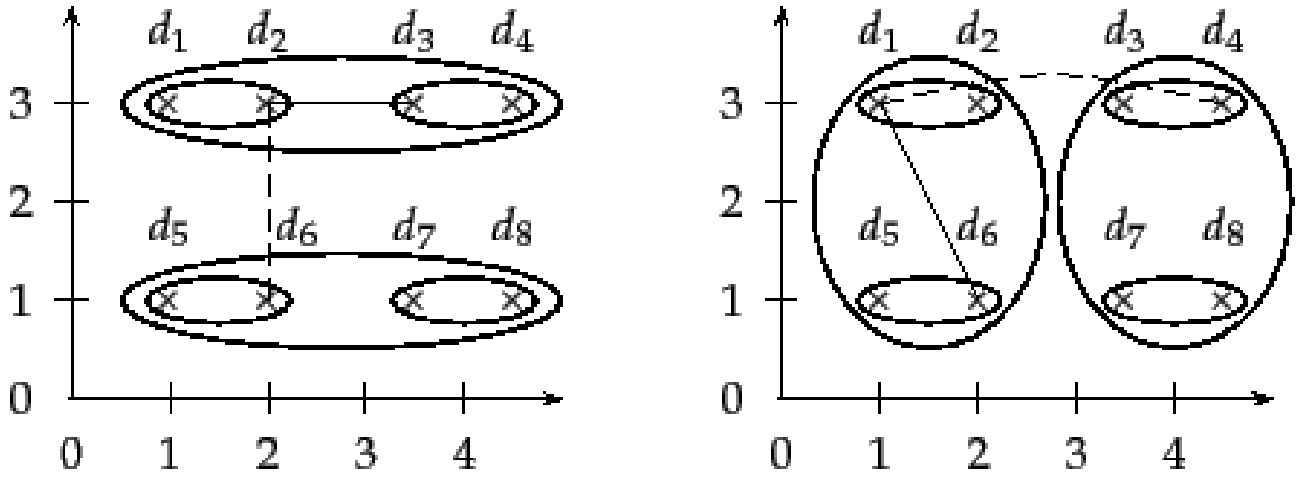
Less compact clusters (chaining)

More compact clusters

Different Linkage Methods

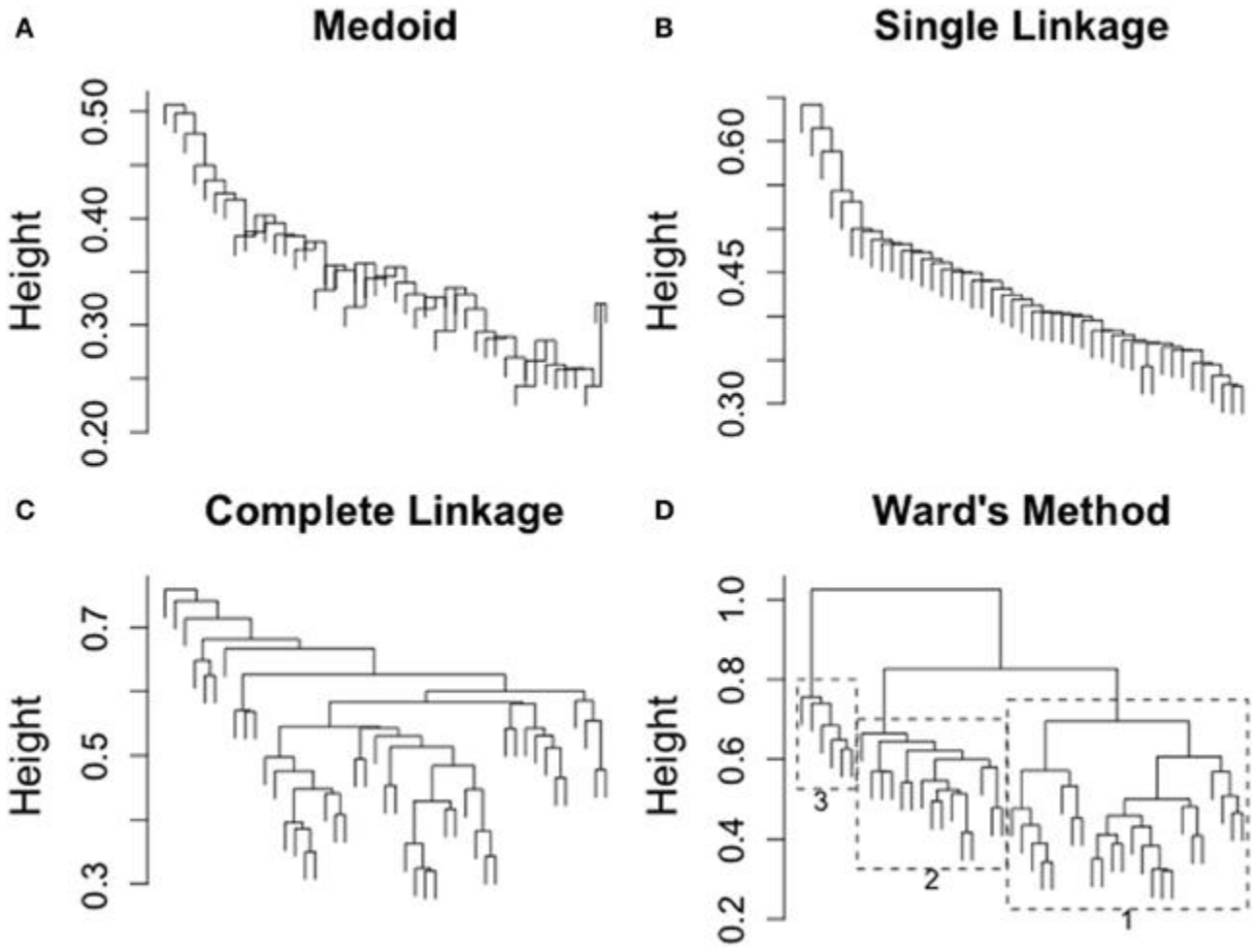


Single Link vs Complete Link



► **Figure 17.2** A single-link (left) and complete-link (right) clustering of eight documents. The ellipses correspond to successive clustering stages. Left: The single-link similarity of the two upper two-point clusters is the similarity of d_2 and d_3 (solid line), which is greater than the single-link similarity of the two left two-point clusters (dashed line). Right: The complete-link similarity of the two upper two-point clusters is the similarity of d_1 and d_4 (dashed line), which is smaller than the complete-link similarity of the two left two-point clusters (solid line).

Dendograms


A
Medoid
Height
0.20
0.30
0.40
0.50
B
Single Linkage
Height
0.30
0.45
0.60
C
Complete Linkage
Height
0.3
0.5
0.7
D
Ward's Method
Height
0.2
0.4
0.6
0.8
1.0
3
2
1