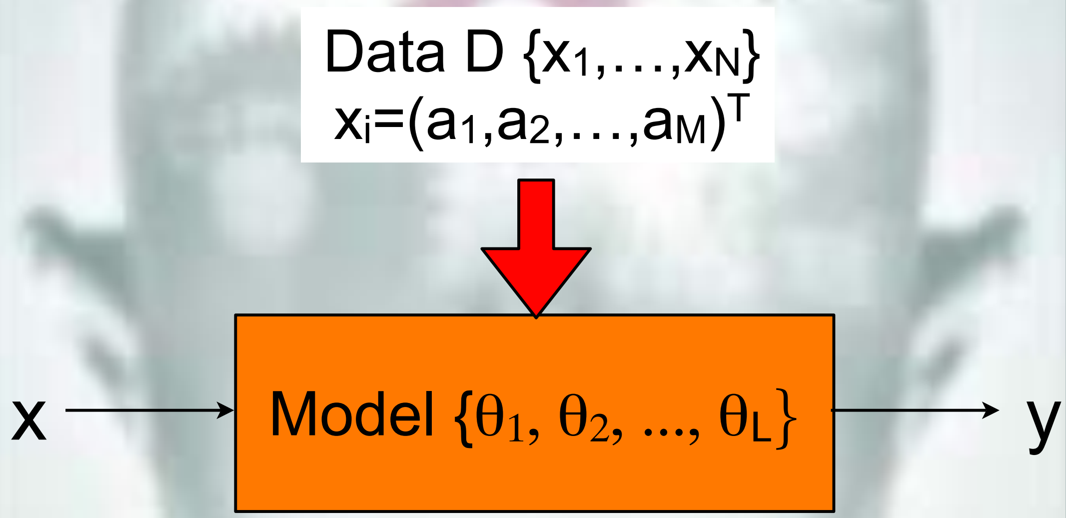
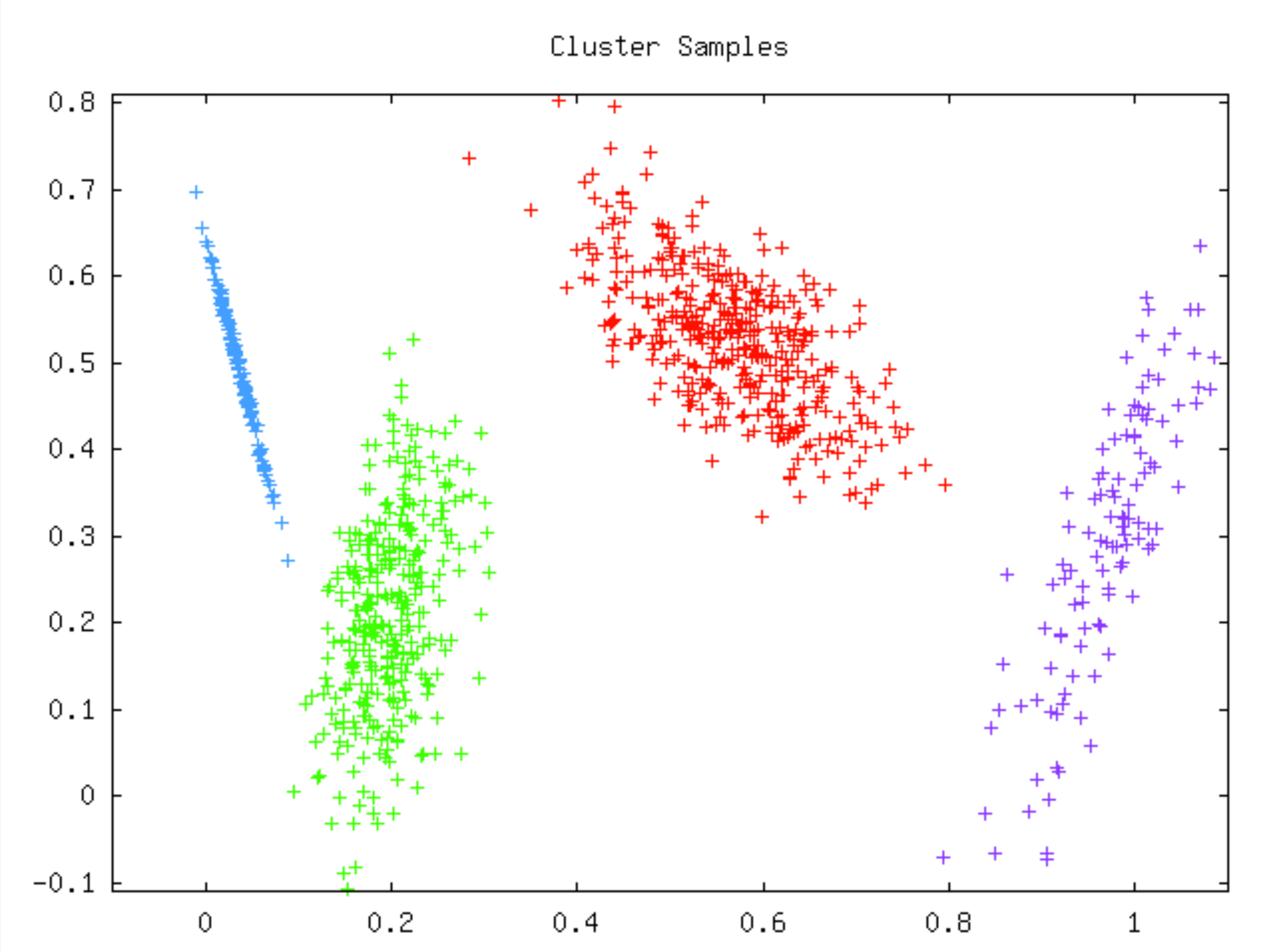
Recap: Unsupervised Learning



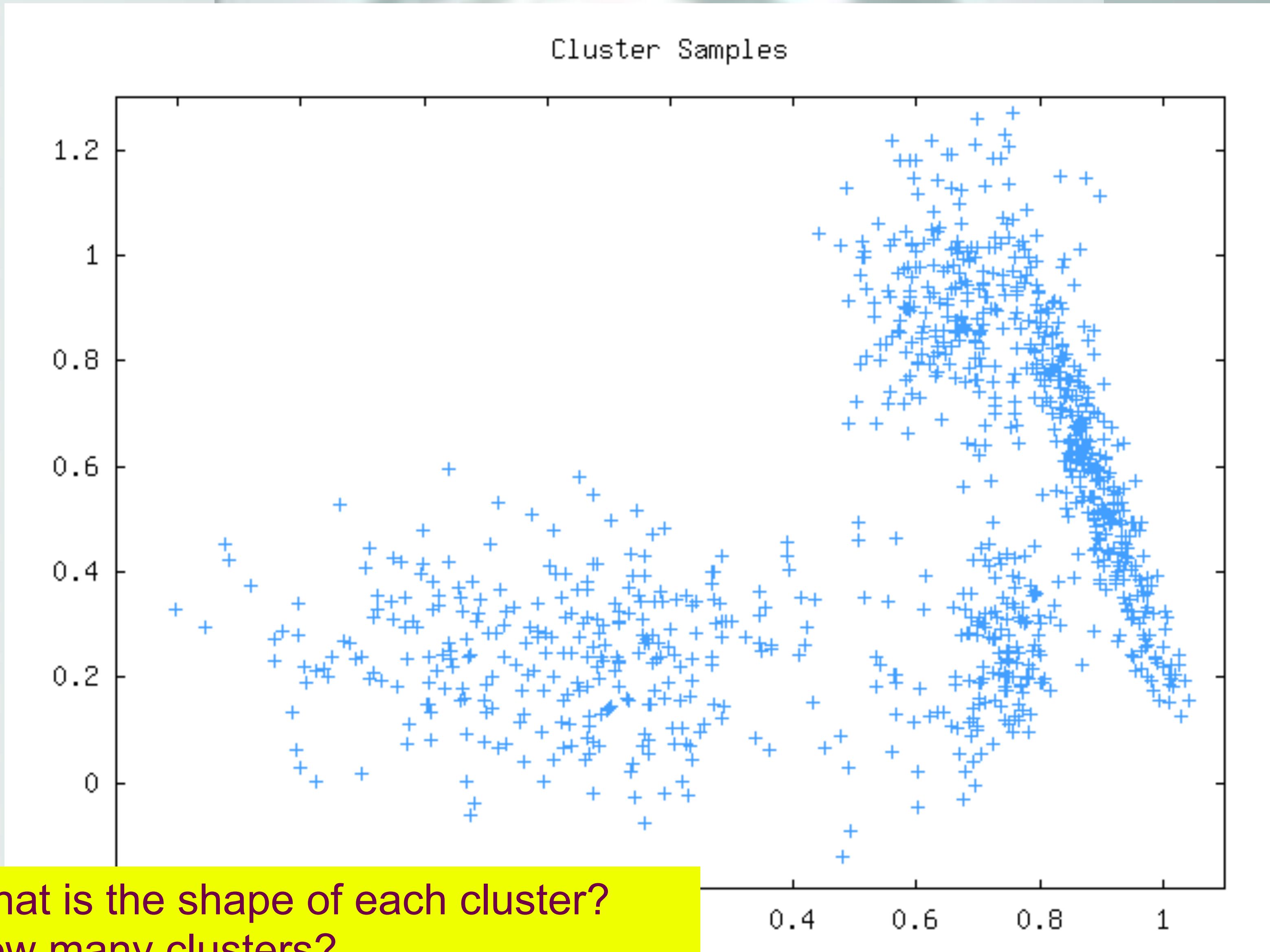
Clustering: y categorical

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

Clustering

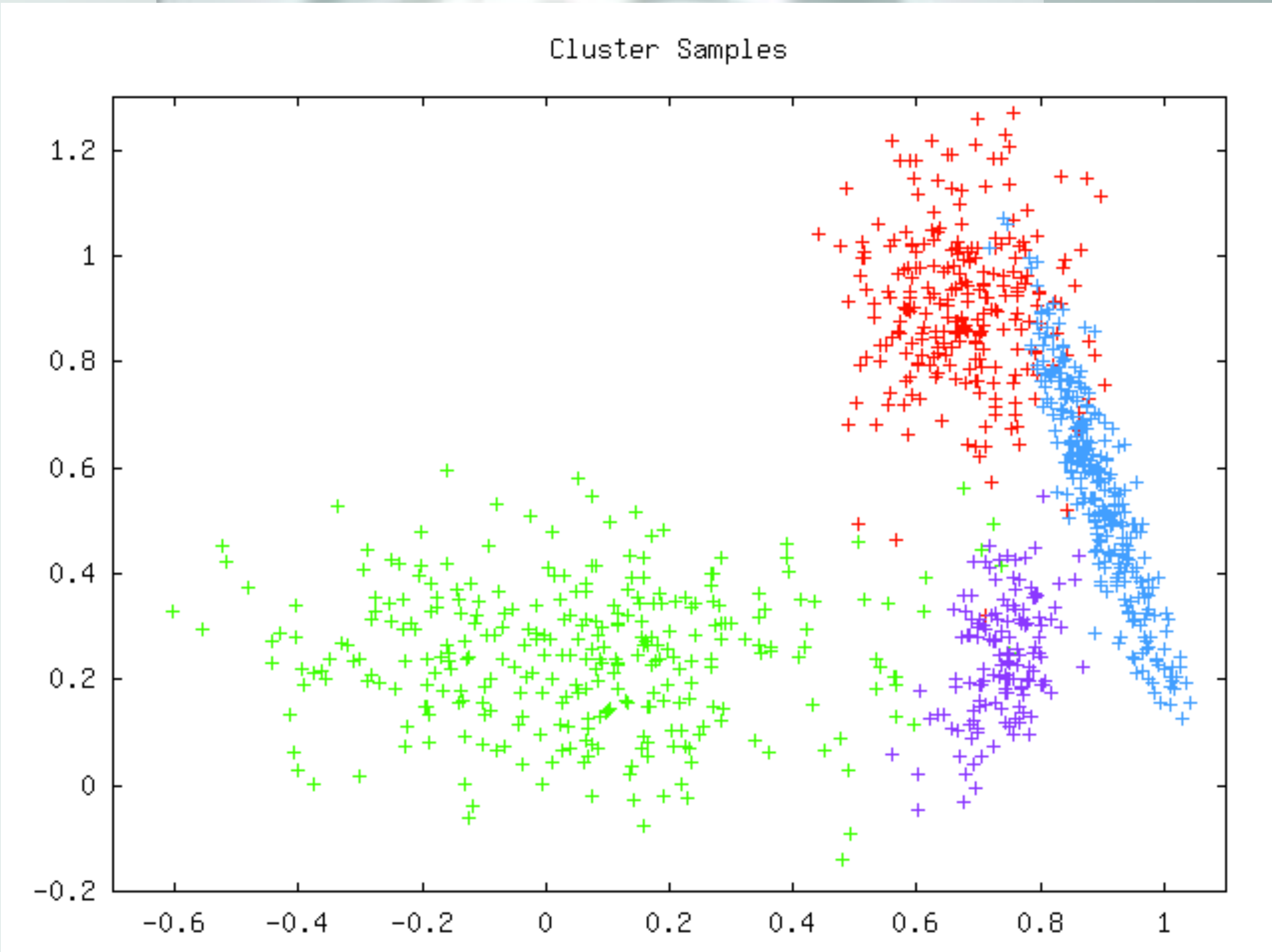


Where Are the Clusters?



- 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

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

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

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[illegible]
- 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

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

K-Means Update

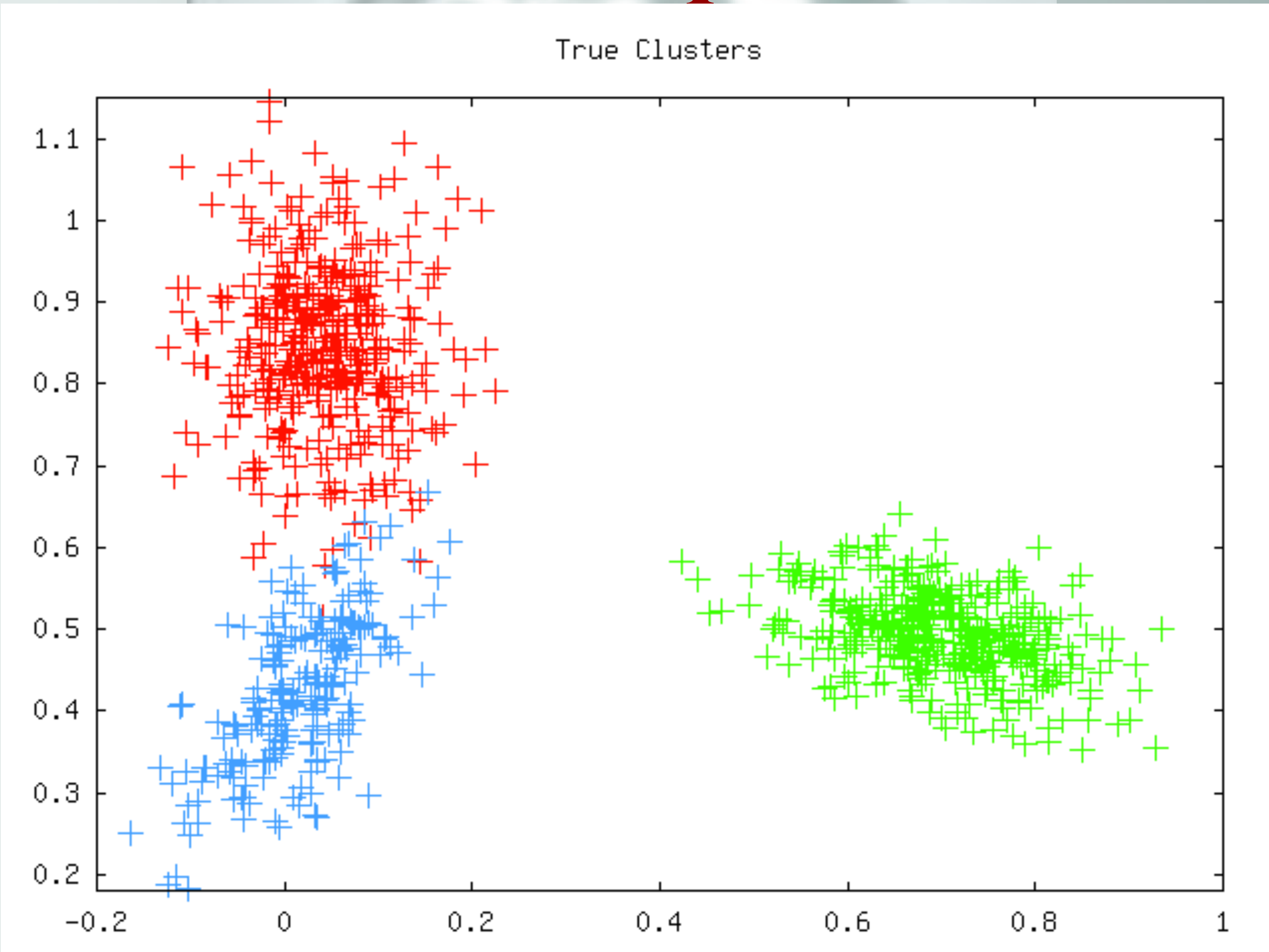
- We *update* our prototypes based on the points that were assigned to it, but taking the average/centroid/mean

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

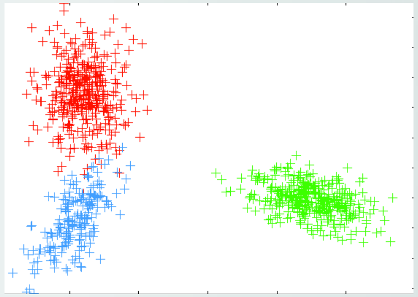
New prototype

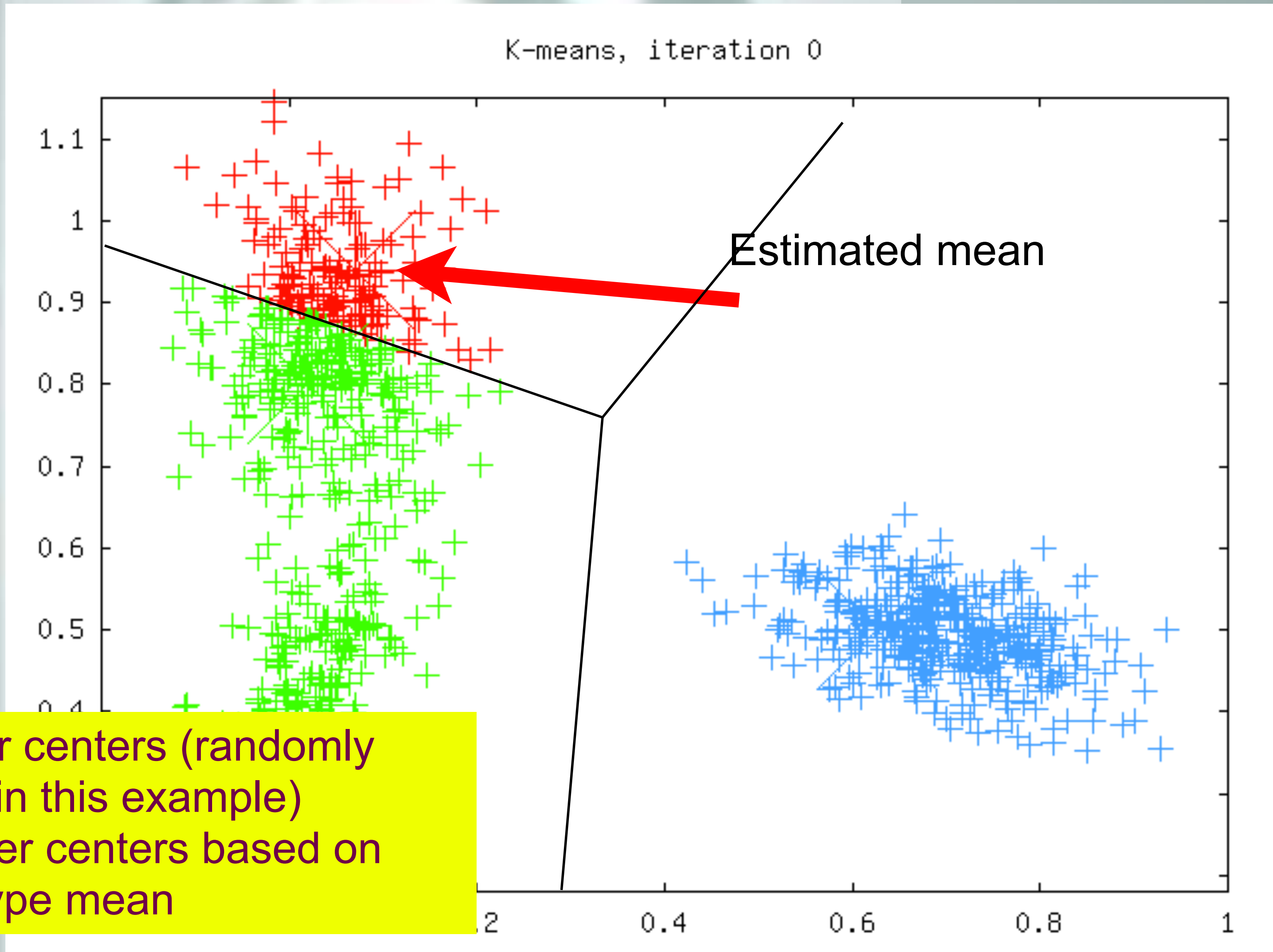
Mean of points assigned to *k*

Example



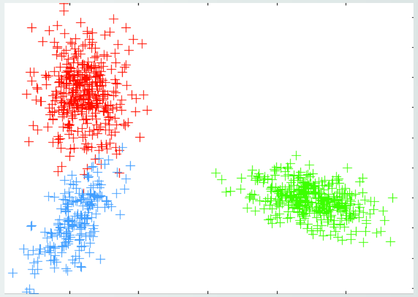
Example: Initialization

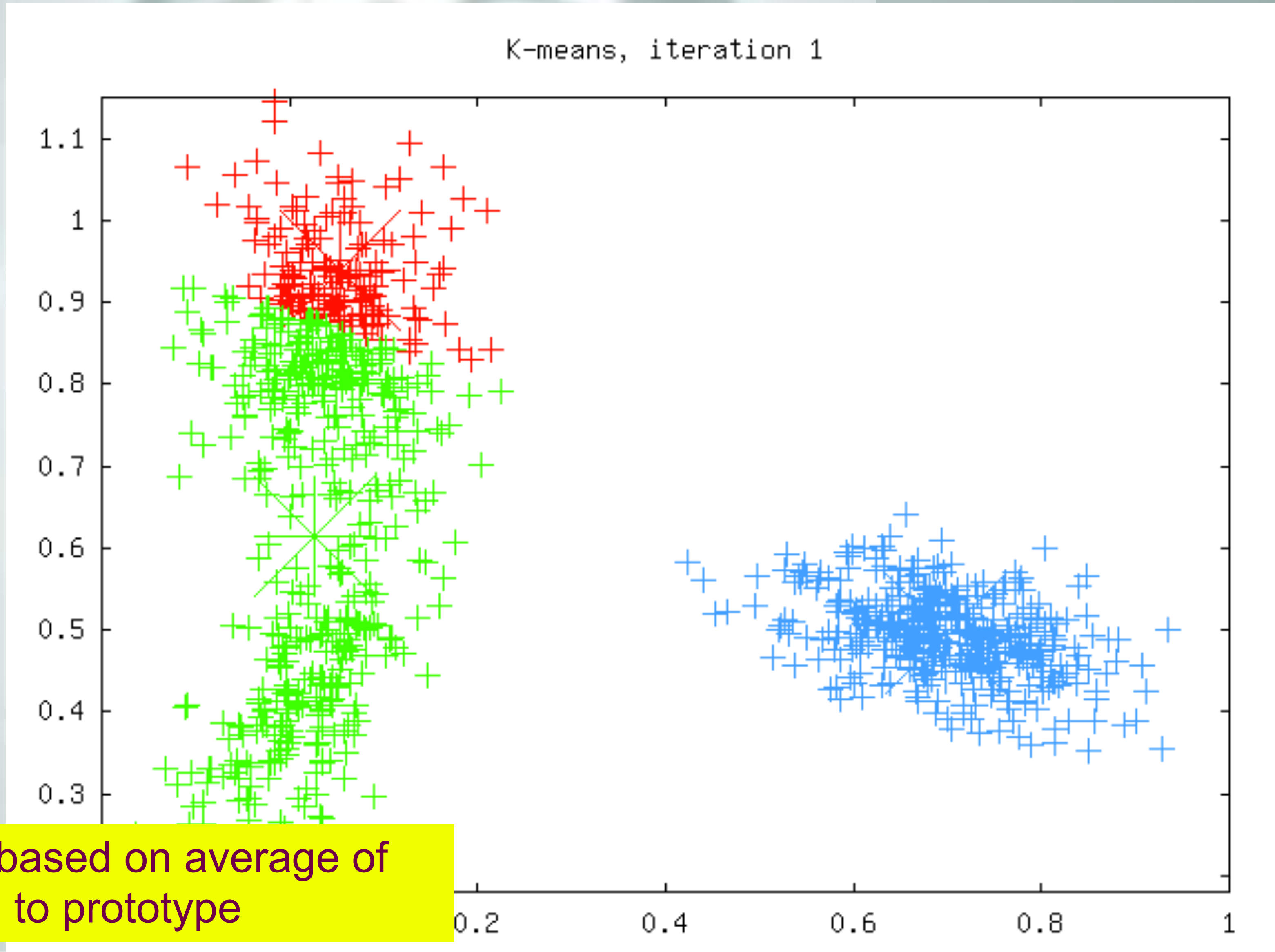




- Initialize cluster centers (randomly selecting data in this example)
- Assign to cluster centers based on nearest prototype mean

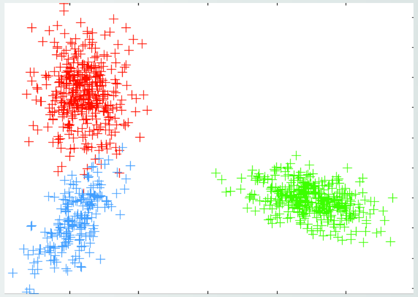
Example: 1 Iteration

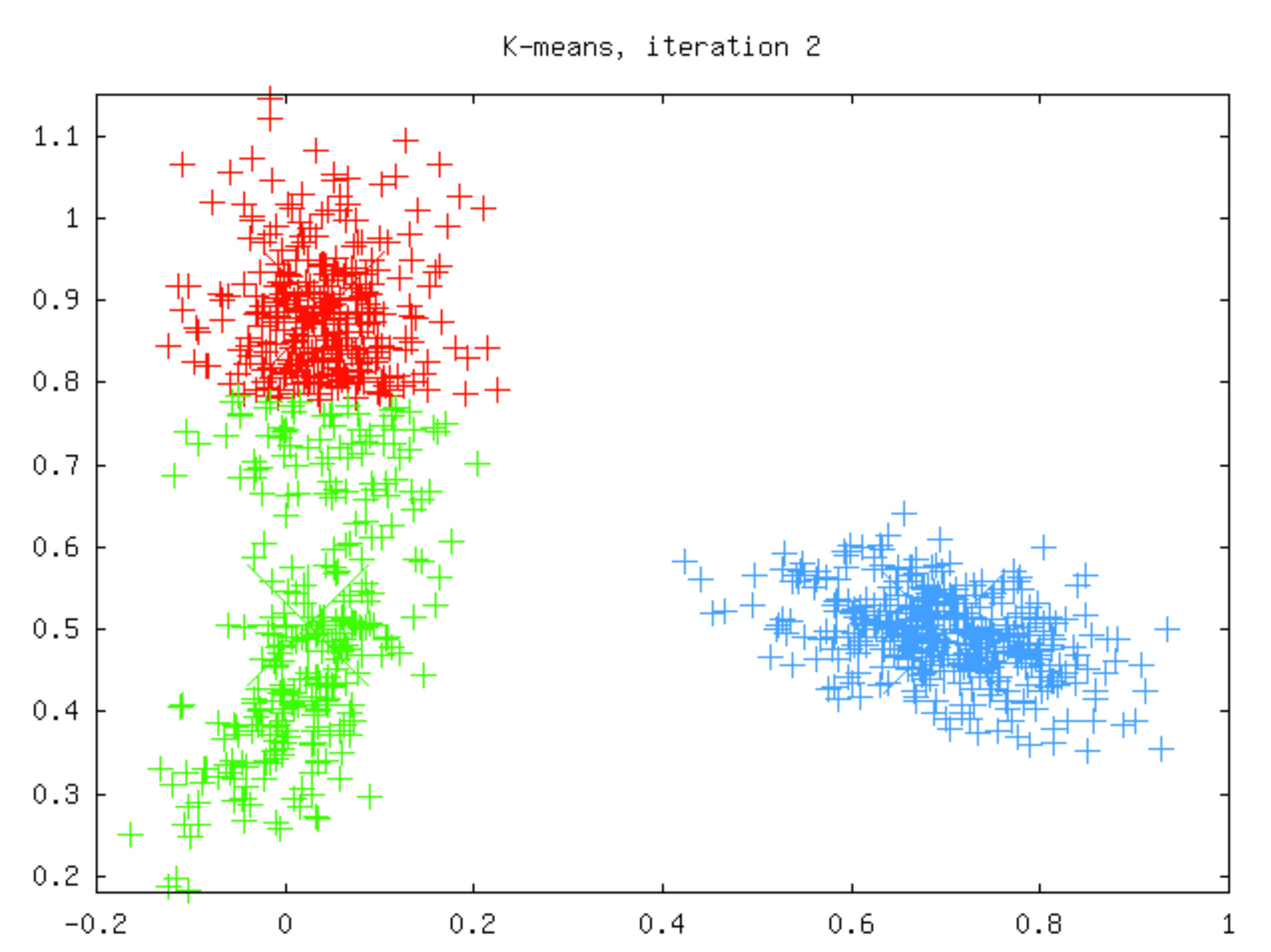




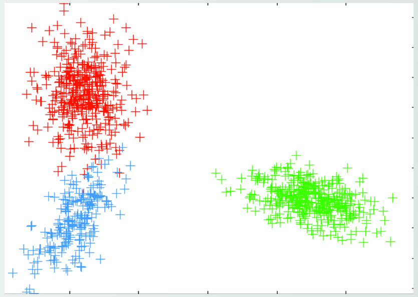
- Update means based on average of points assigned to prototype

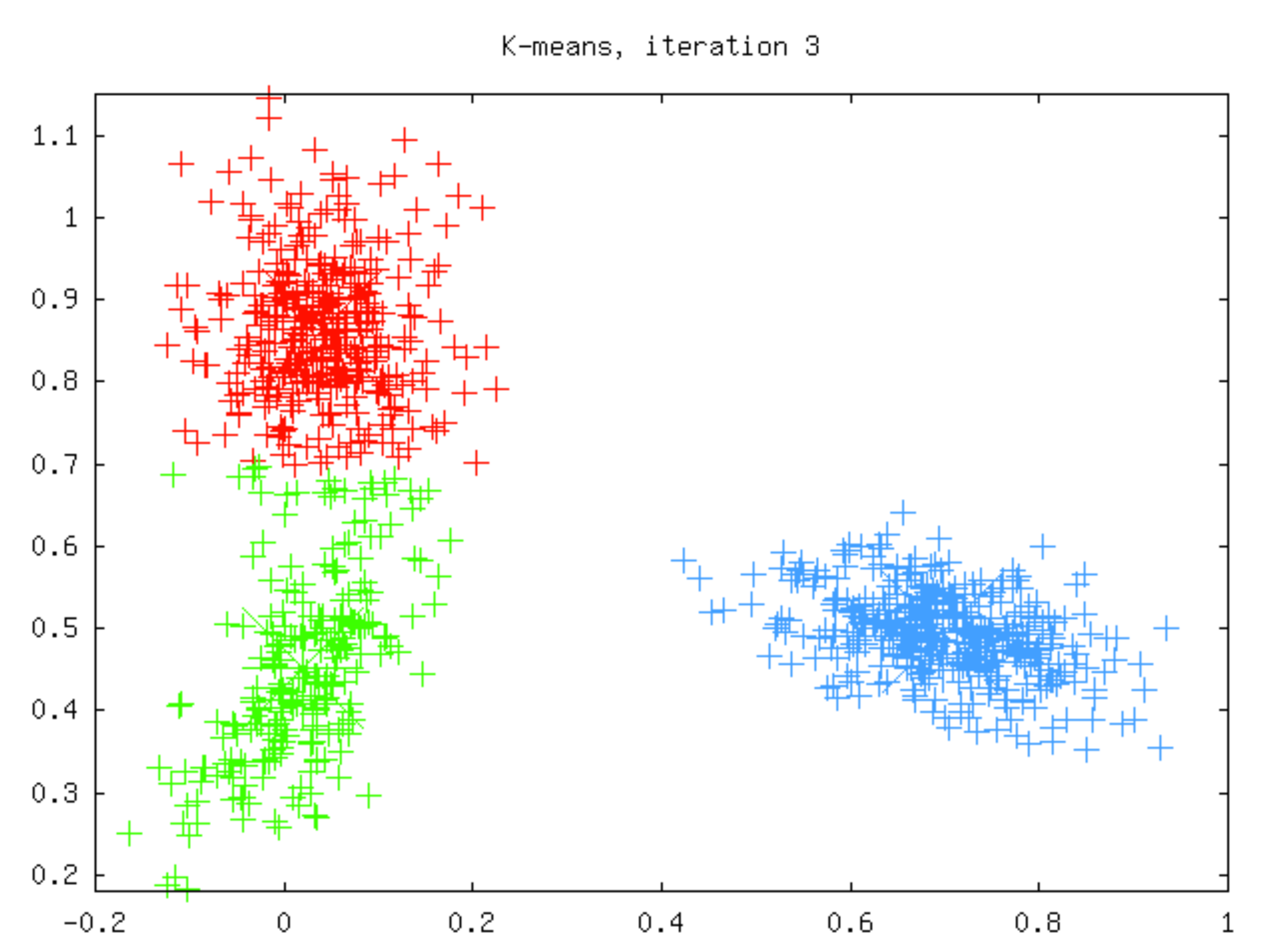
Iteration 2



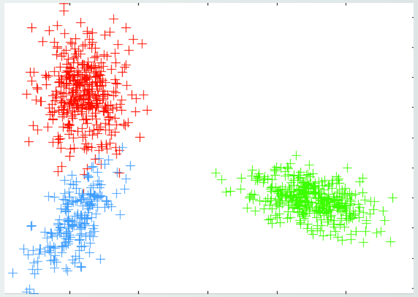


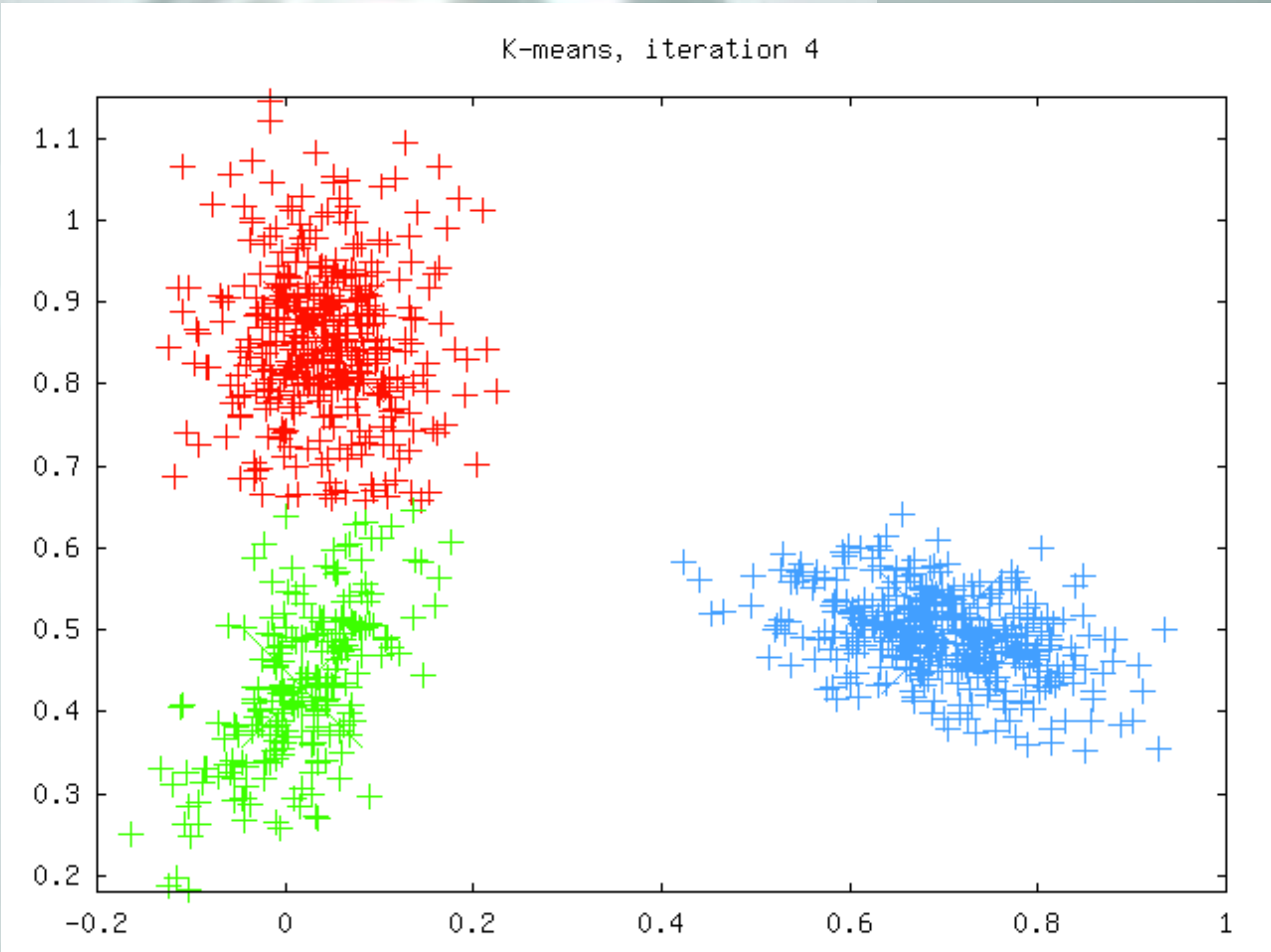
Iteration 3



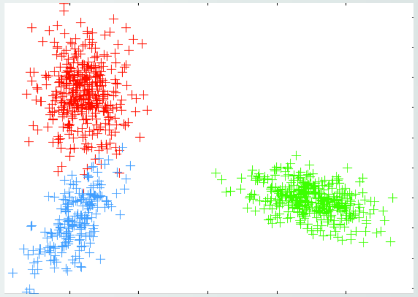


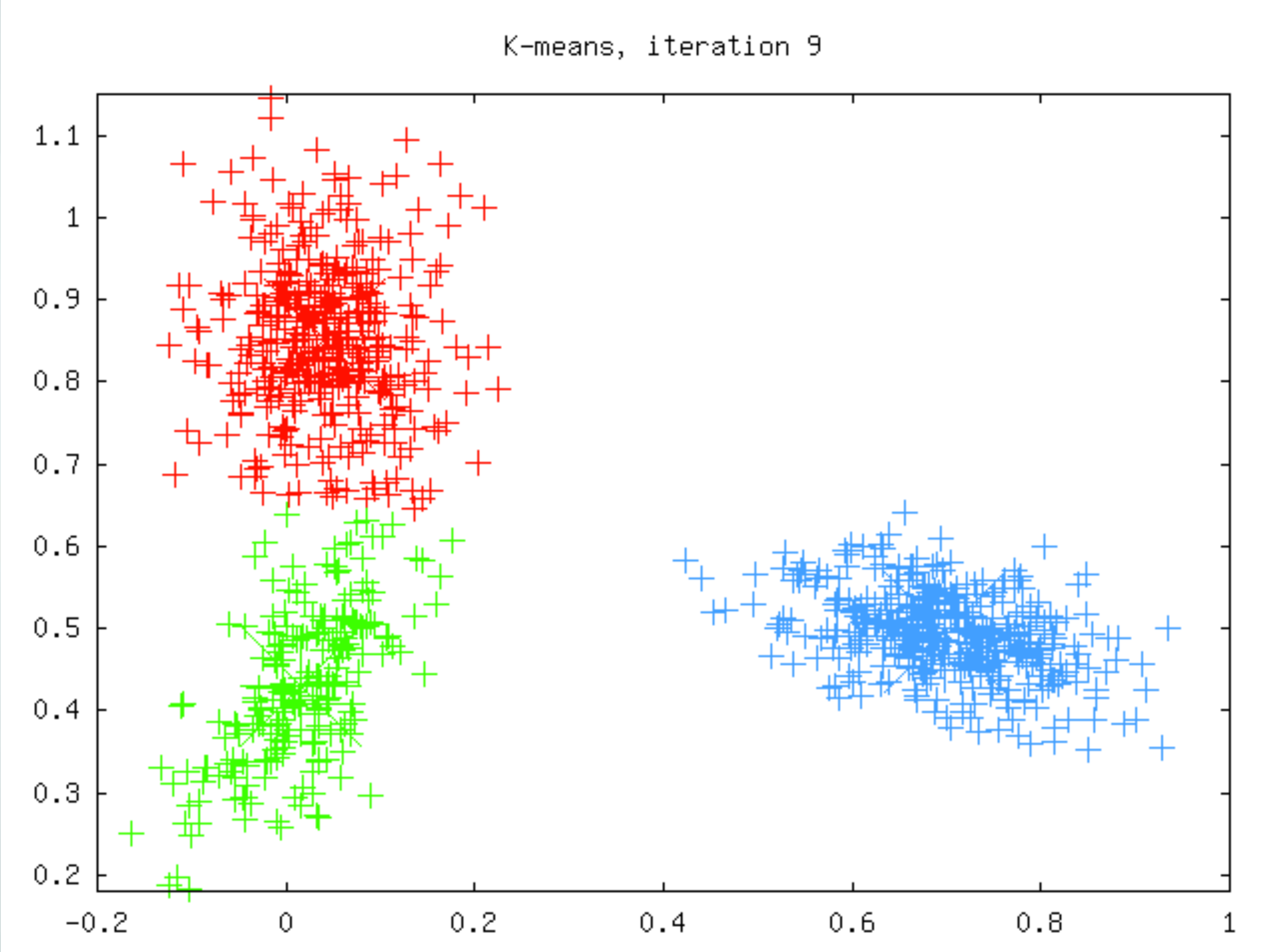
Iteration 4





Iteration 9





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?

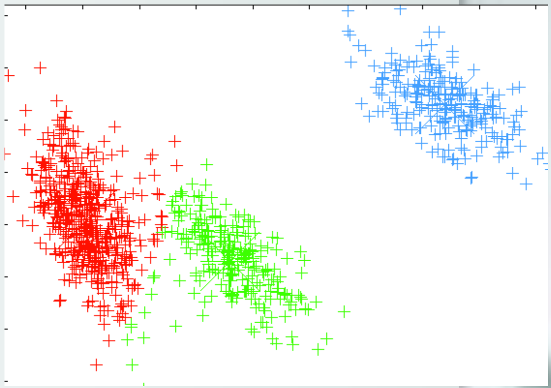
- K-means is a local search technique for optimizing the distortion of the data

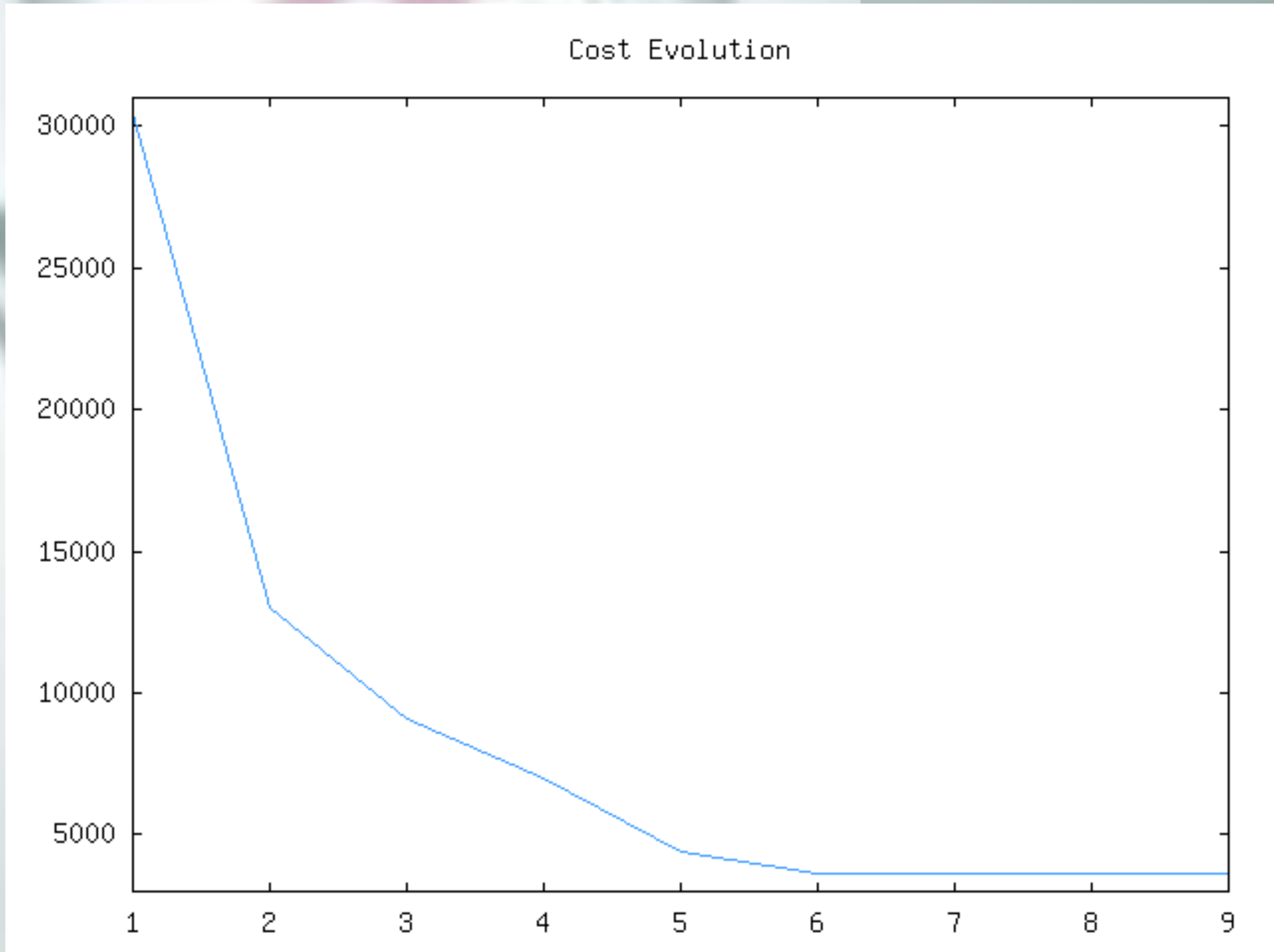
- Formally, K-means tries to optimize the within-point scatter

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

How does this change with each iteration?

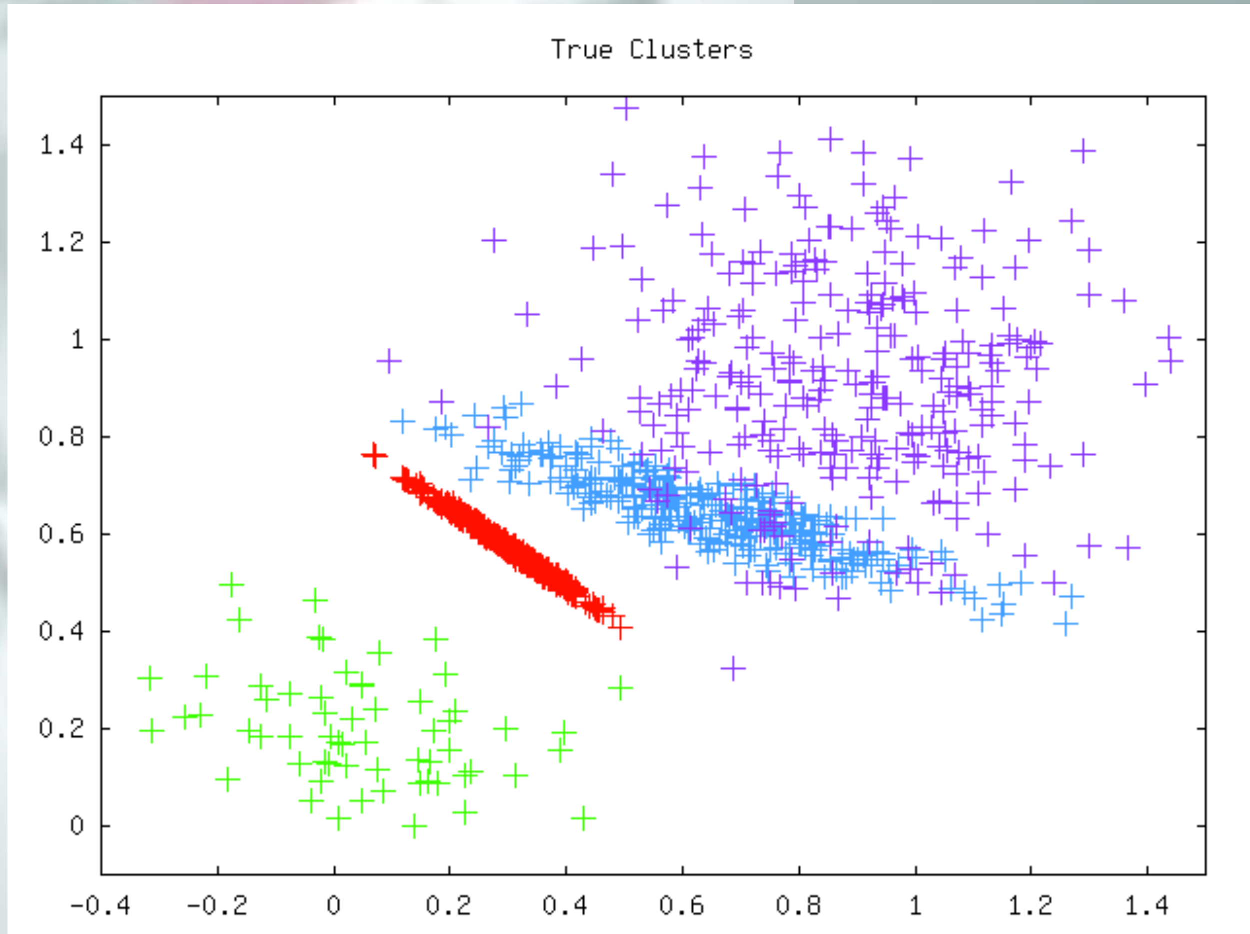
From Previous Example



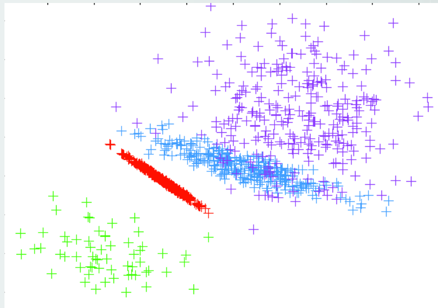


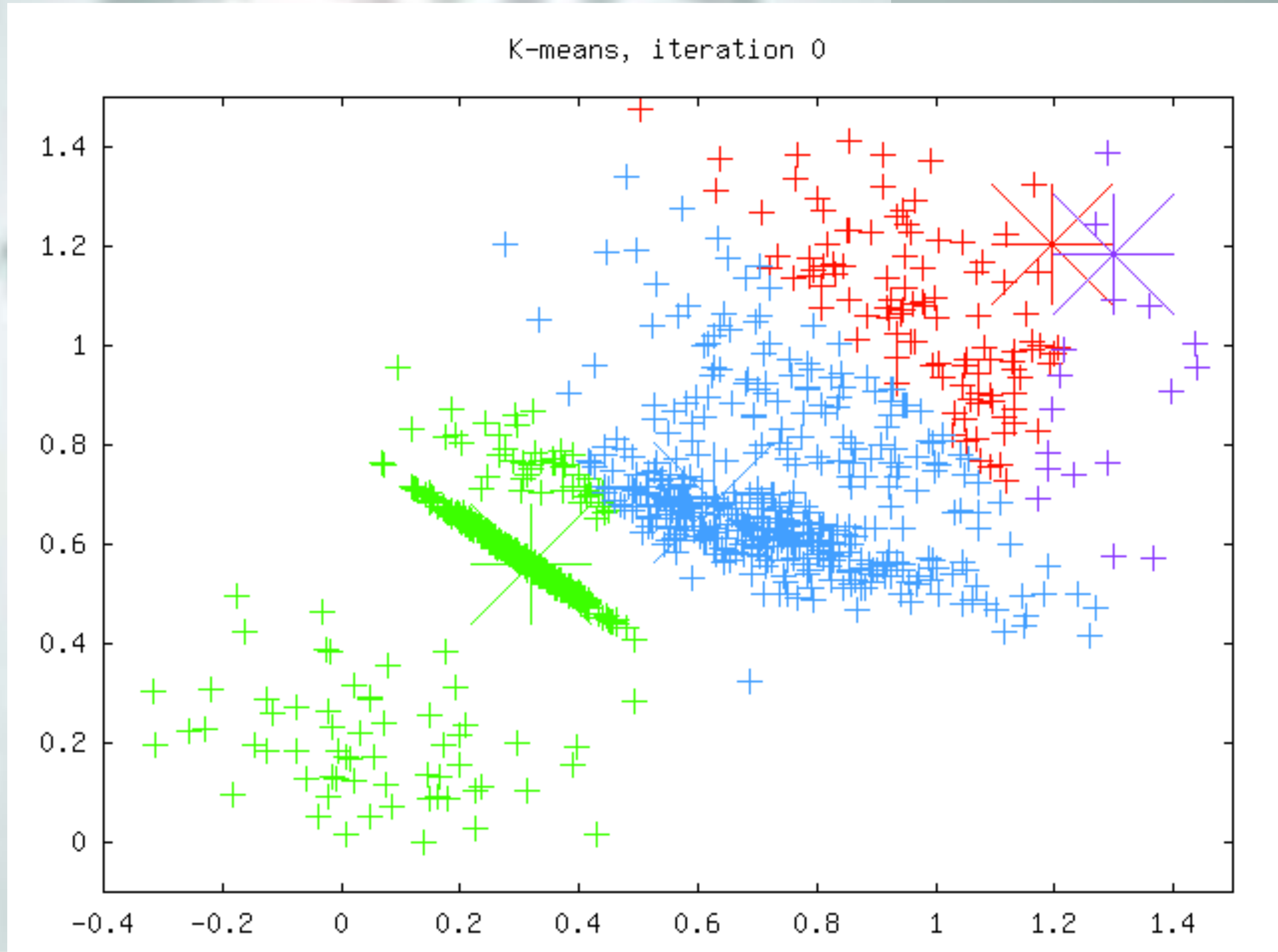
Does It Always Work?

- Unfortunately, no

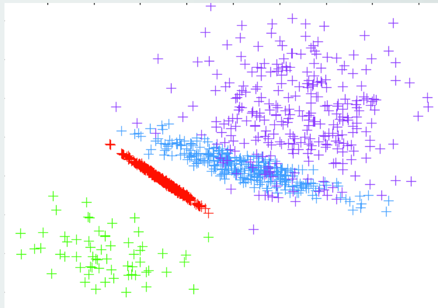


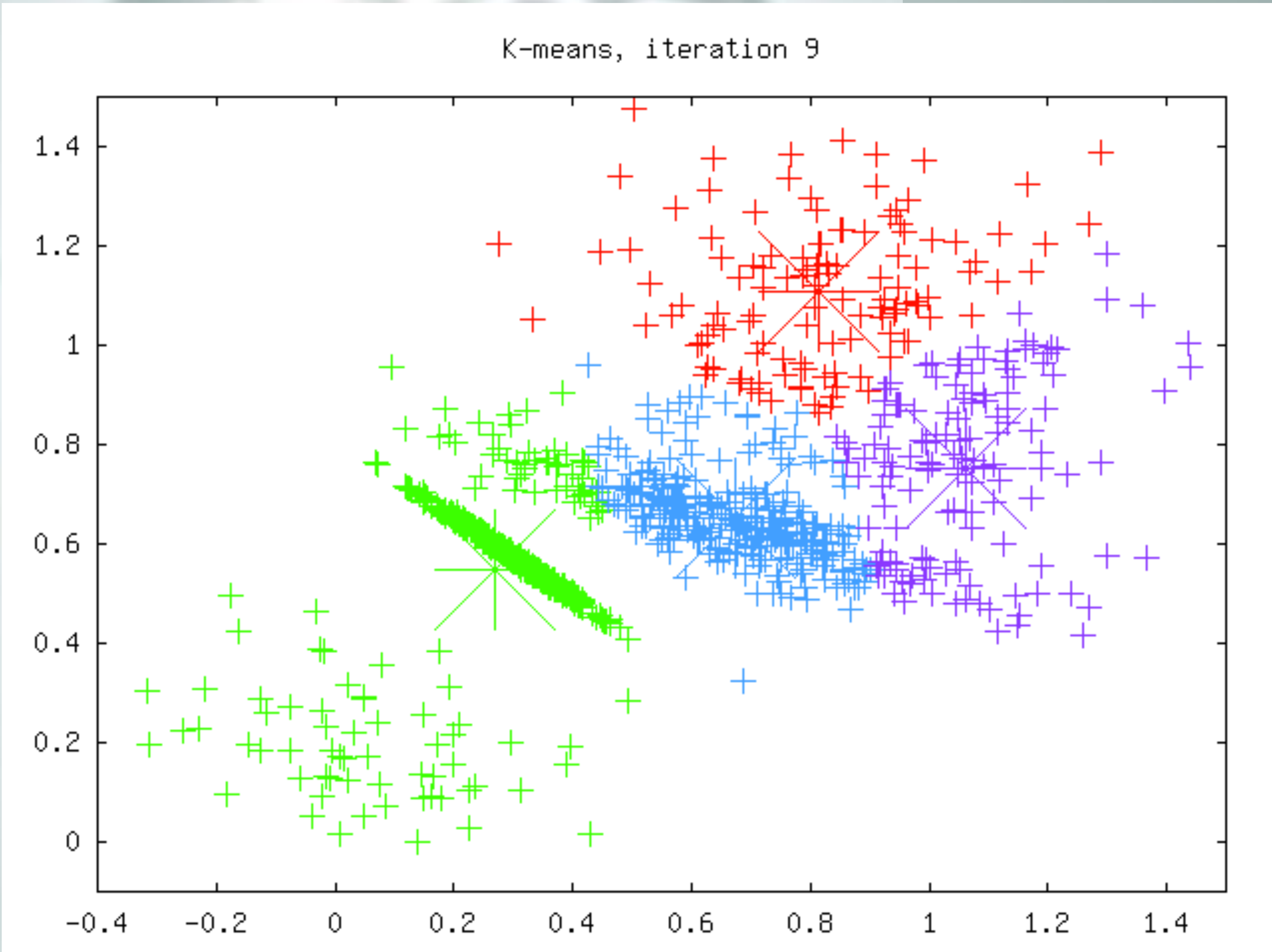
Does It Always Work?





After 9 Iterations





What Happened?

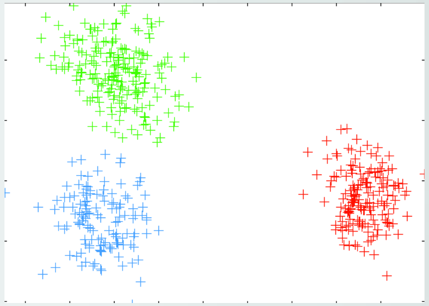
- K-means can get stuck in local optima
  - Effectively, it will depend on the starting condition

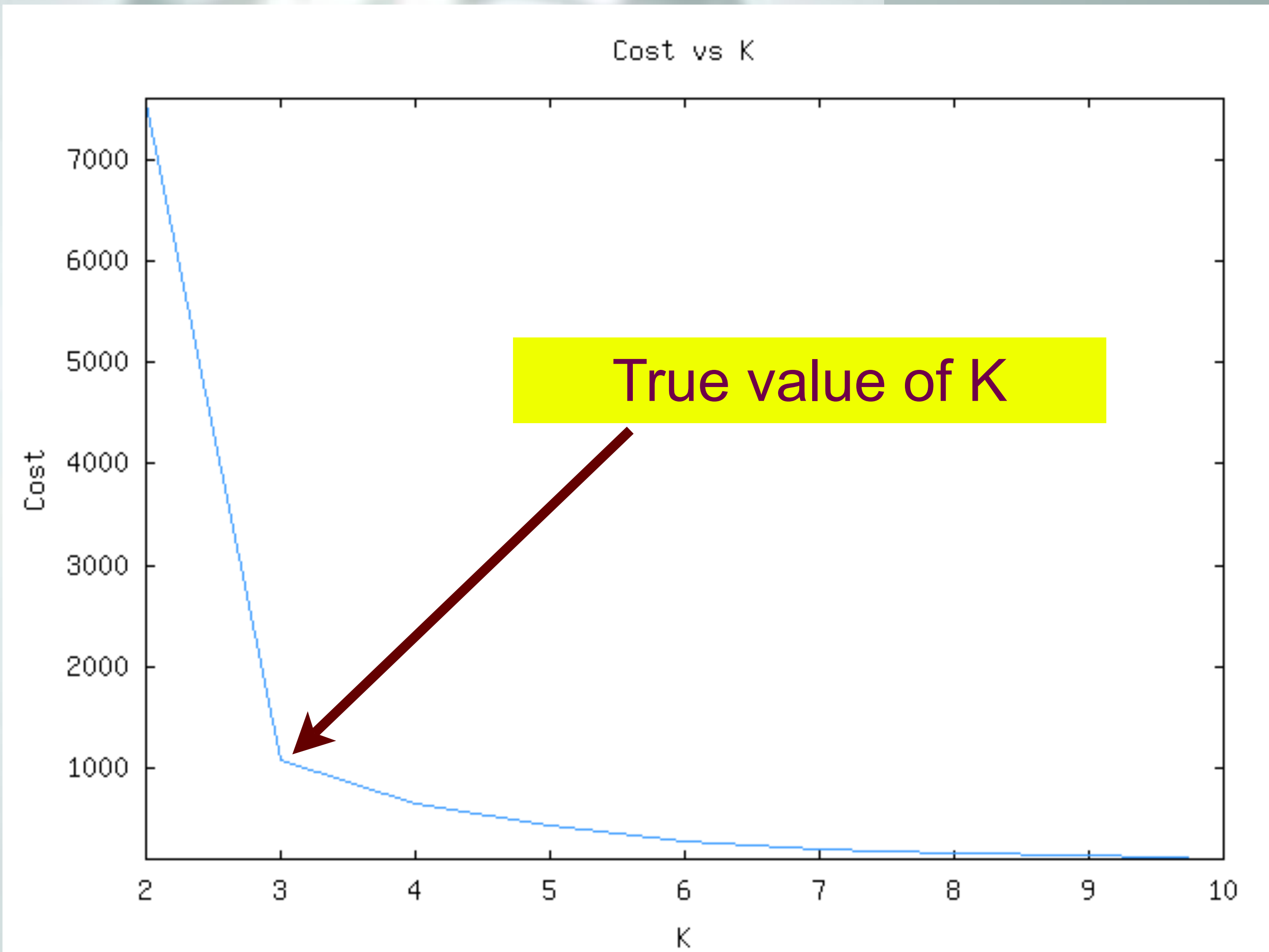
- How can we “fix” this?

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?



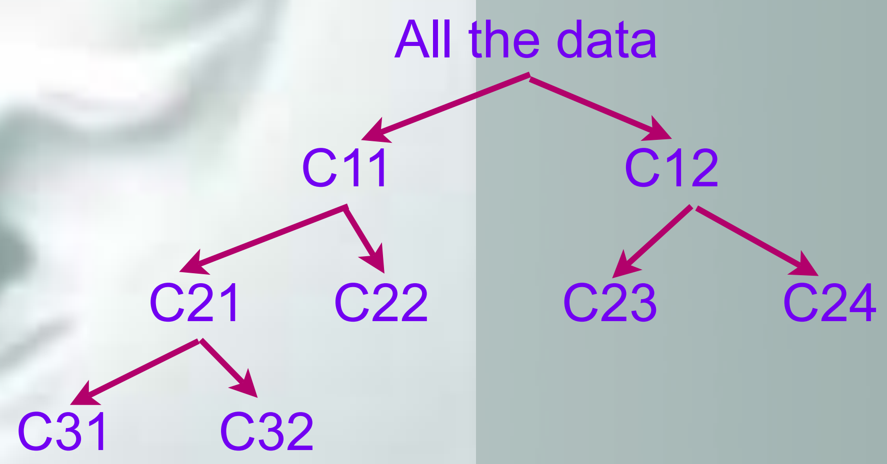


K-Means Summary

- 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

Hierarchical Methods

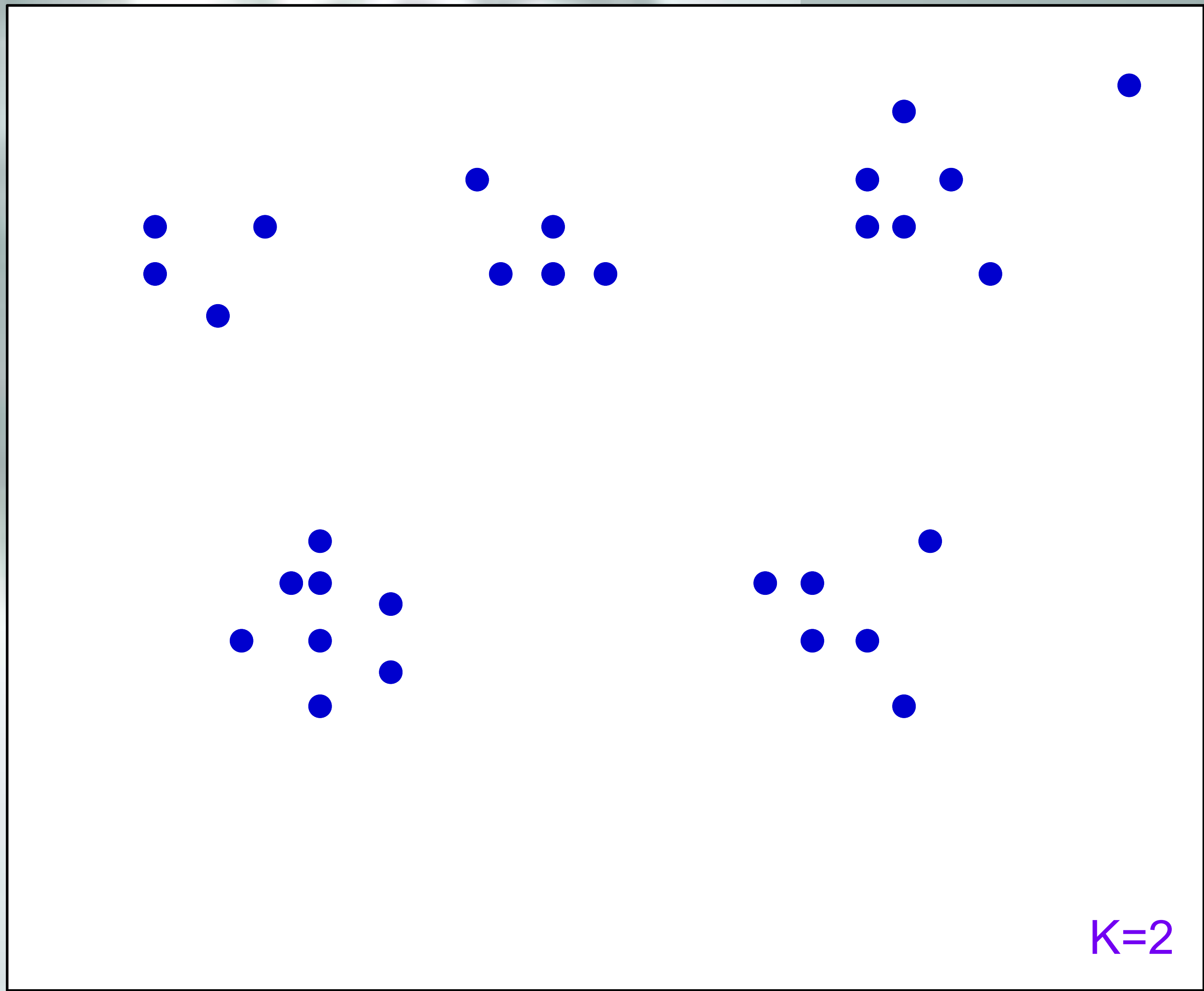
- Recall K-means
  - Input = K, measure of dissimilarity (distances)
  - Output = Cluster centers
- Hierarchical techniques avoid needing to specify K
  - Input = Measure of dissimilarity (e.g. distances)
  - Output = Hierarchical model of data similarity
- Output is a tree (dendogram)



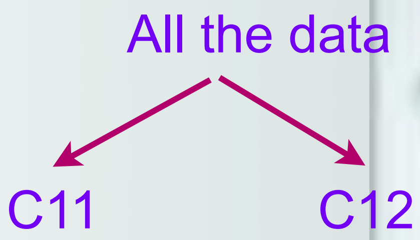
Divisive Methods

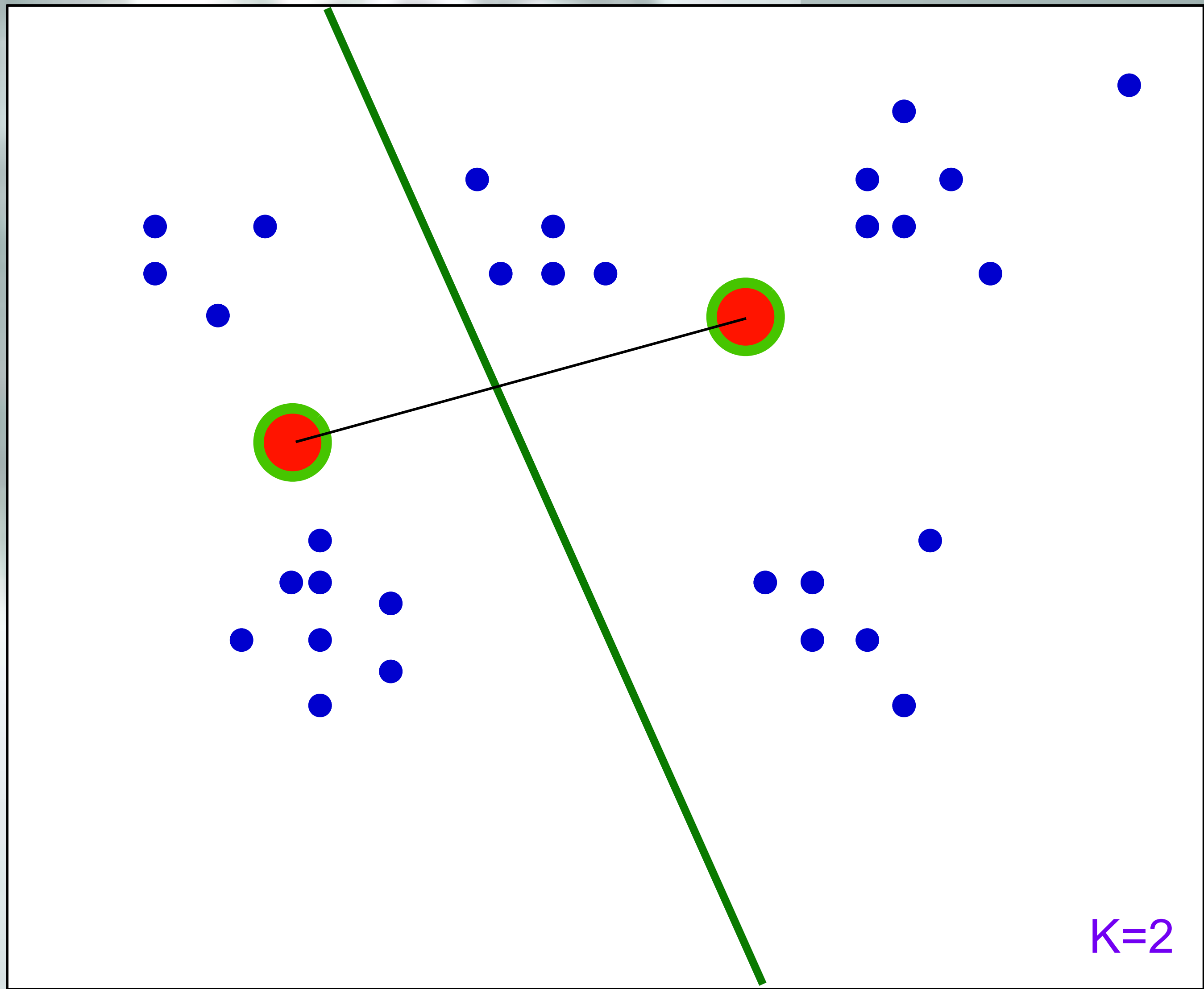
- Two types of hierarchical clustering:
  - Divisive (top down), and agglomerative (bottom up)
- Hierarchical K-means is a divisive method
  - Start with all the data in 1 cluster
  - Split using "flat" K-means
  - For each cluster, recursively split each cluster
- K is usually small
- Need to decide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

Hierarchical K-Means

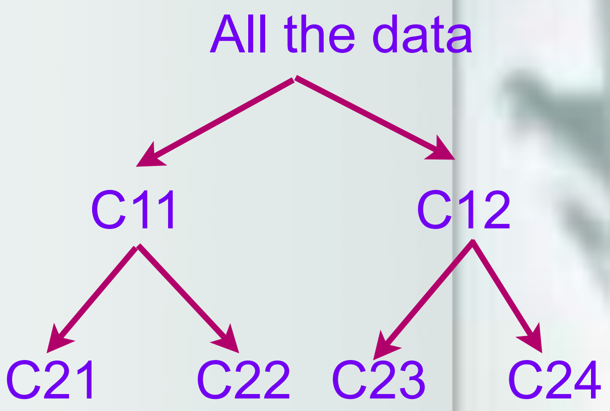


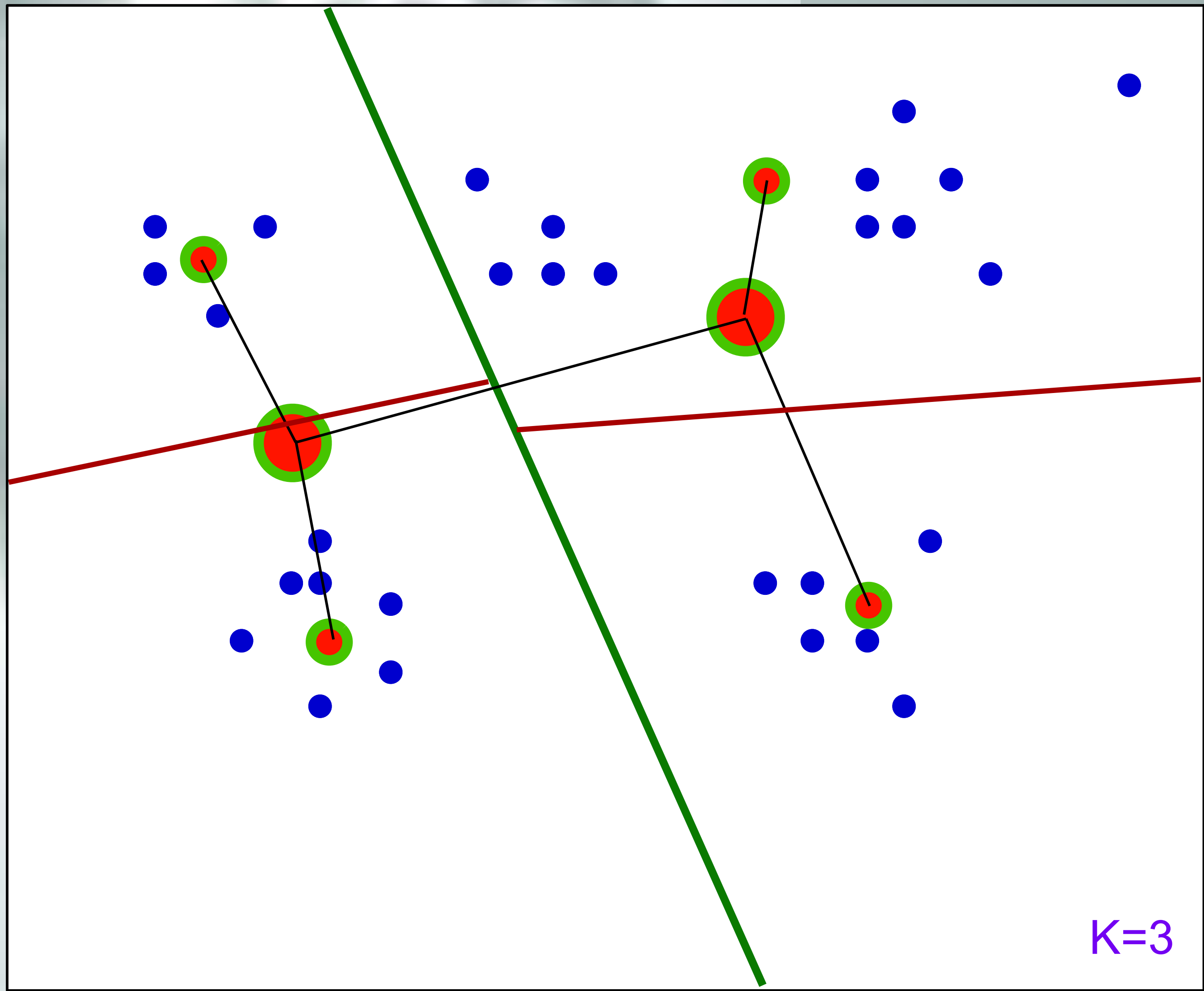
Hierarchical K-Means



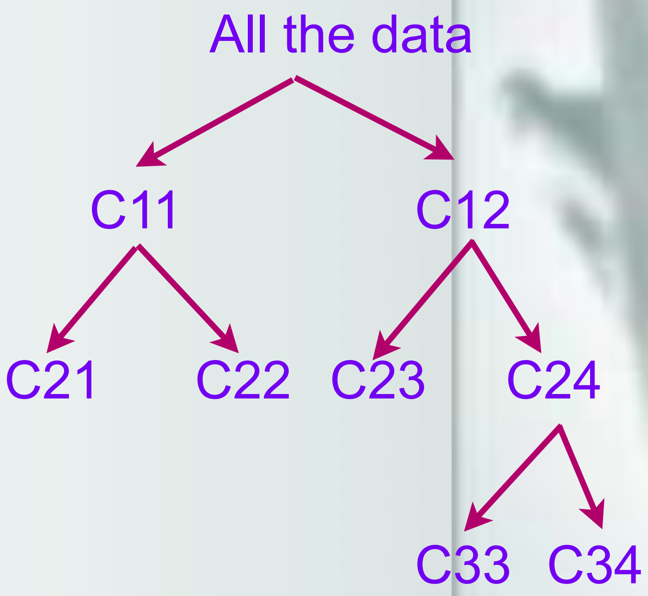


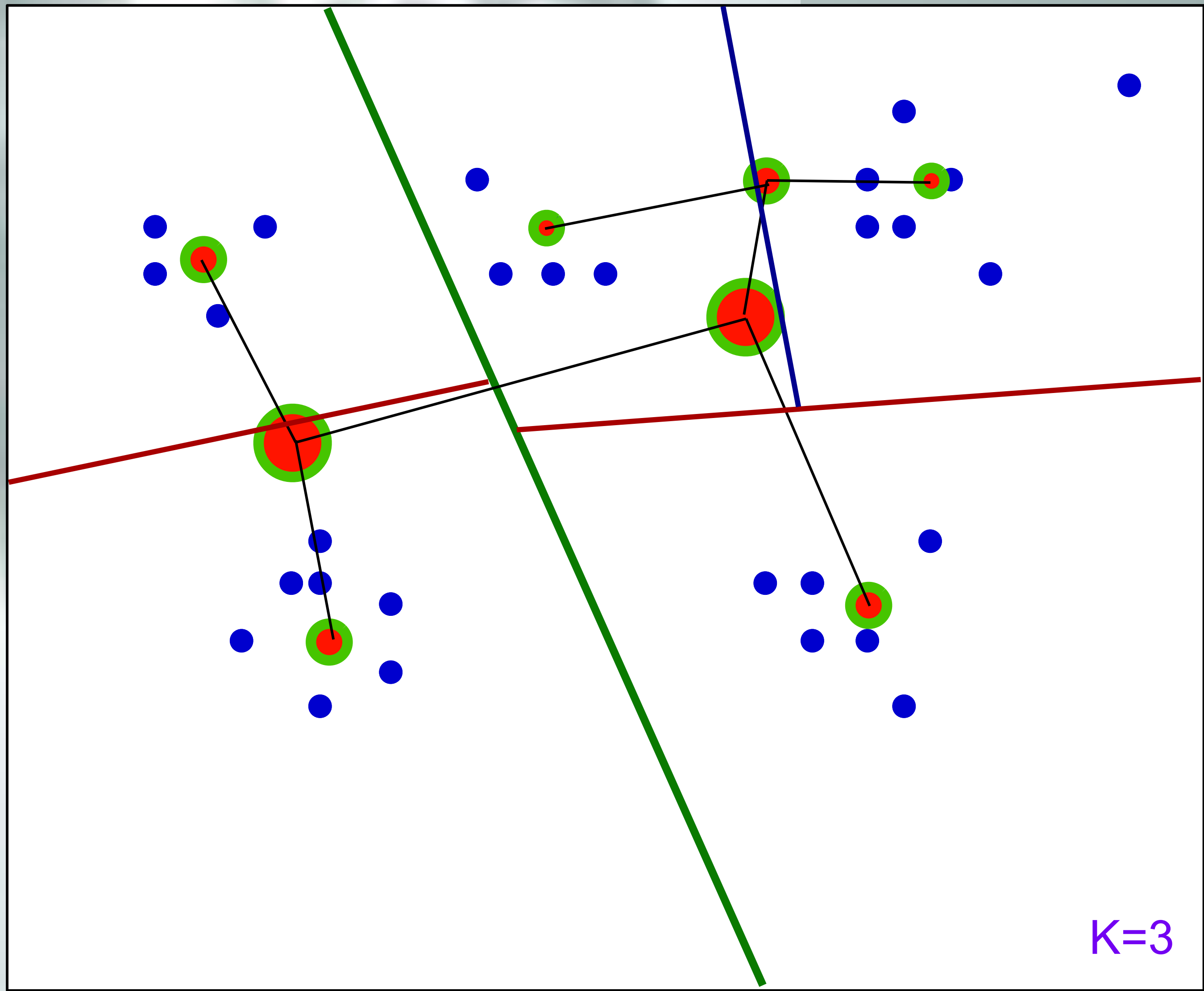
Hierarchical K-Means





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

Single Link — Group Average — Complete Link

Less compact clusters (chaining) ⟷ More compact clusters