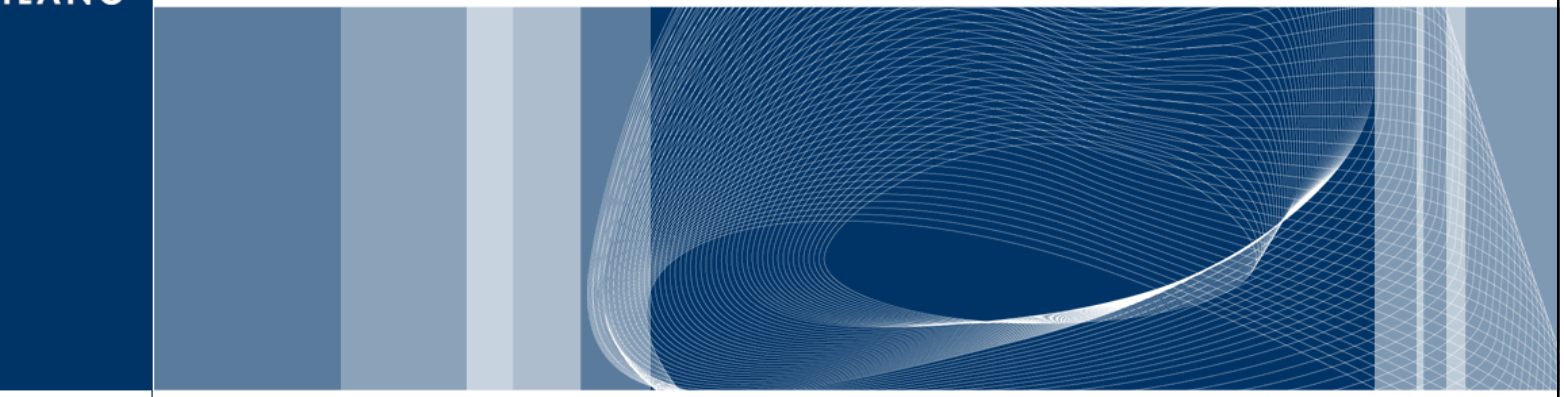
POLITECNICO DI MILANO



Representative-Based Clustering

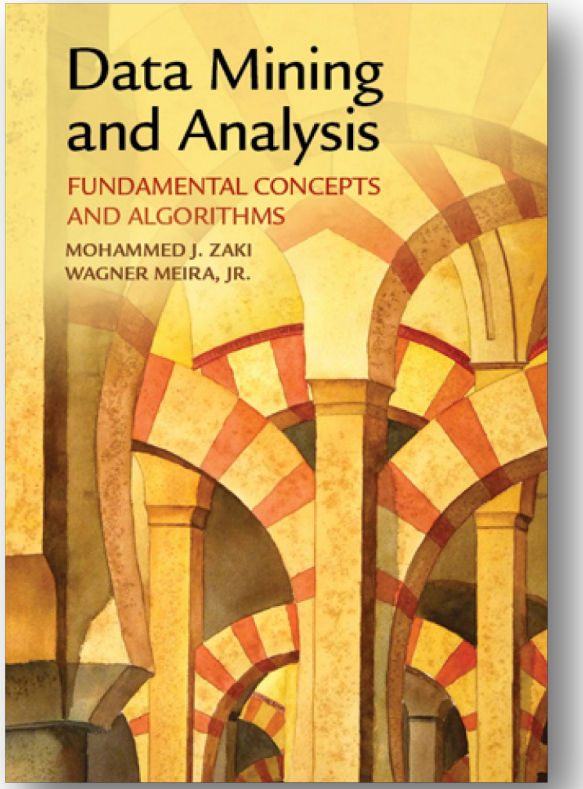
Data Science for Mobility

Prof. Pier Luca Lanzi

Readings

- "Data Mining and Analysis" by Zaki & Meira
  - Chapter 13

- http://www.dataminingbook.info



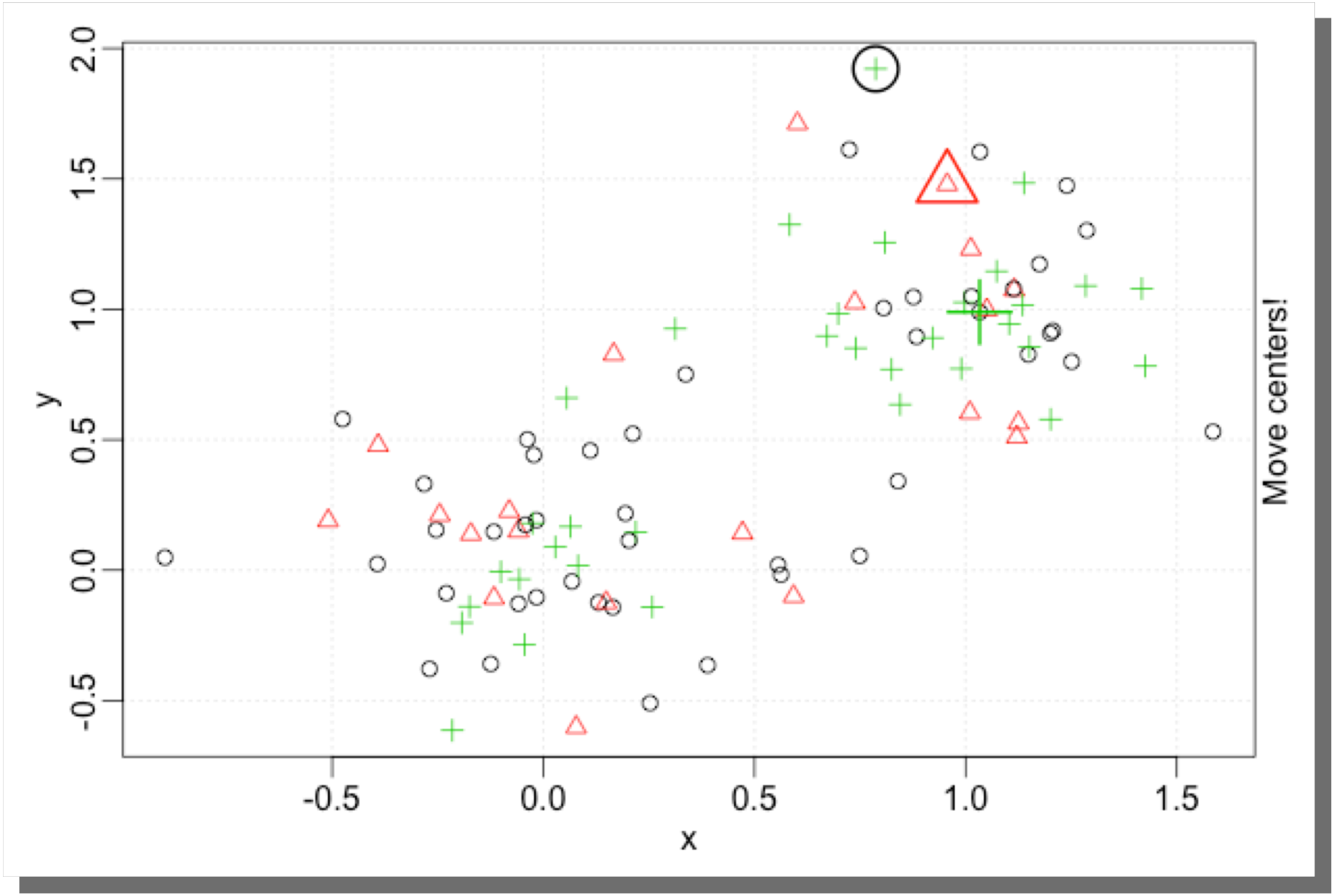
How can we represent clusters?

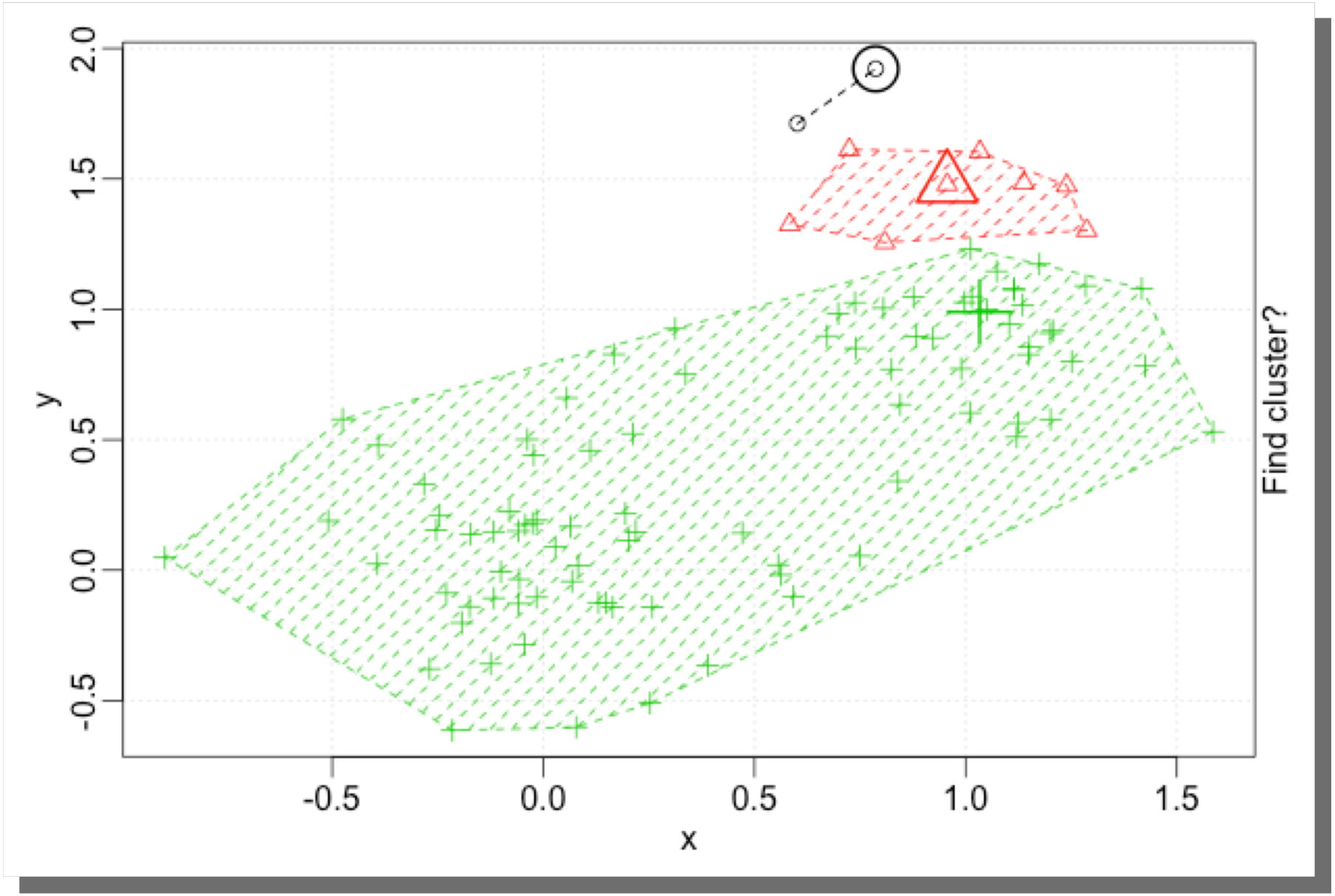
Representation-Based Algorithms

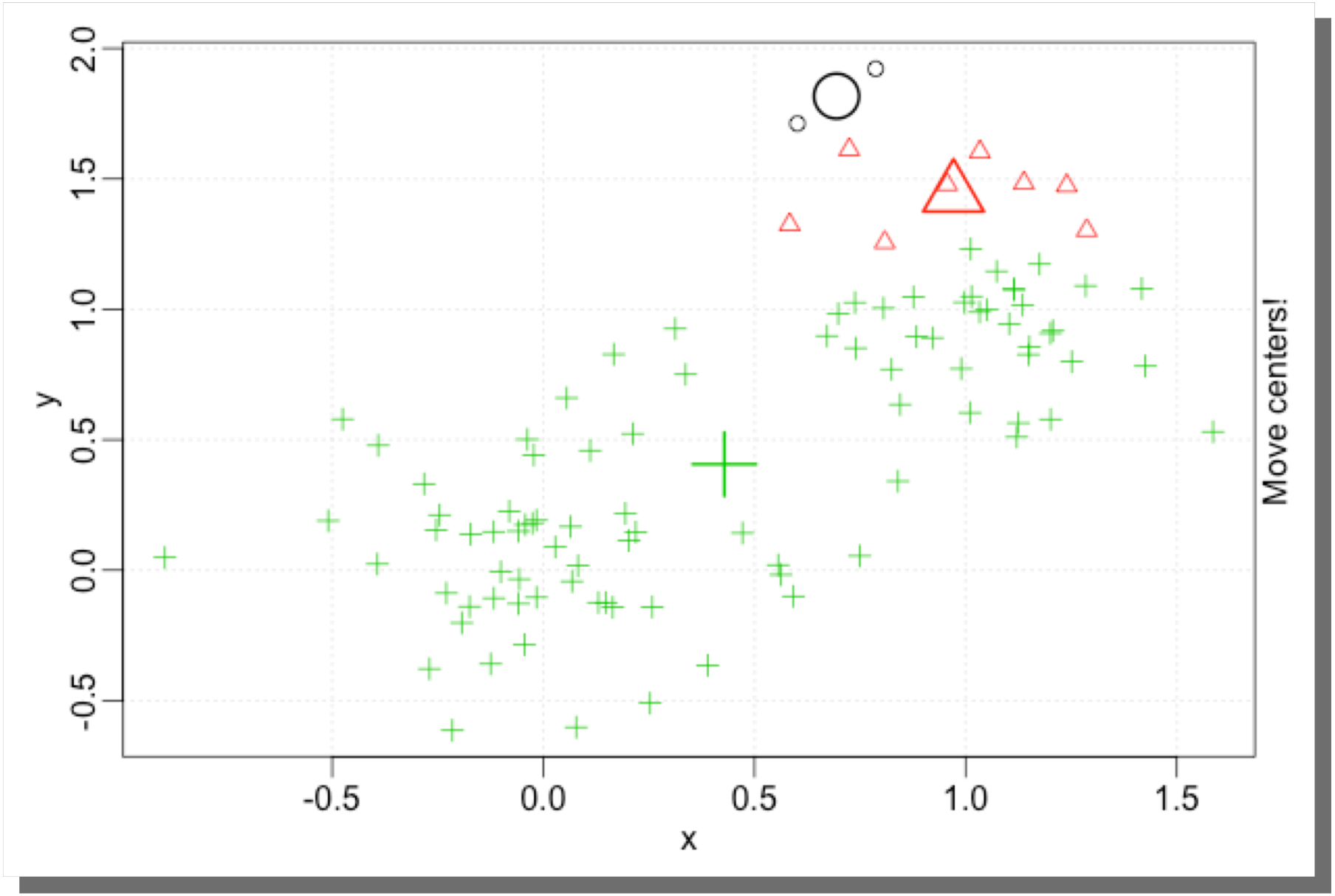
- Given a dataset of N instances, and a desired number of clusters k, this class of algorithms generates a partition C of N in k clusters $\{C_1, C_2, \ldots, C_k\}$
- For each cluster there is a point that summarizes the cluster
- The common choice being the mean of the points in the cluster

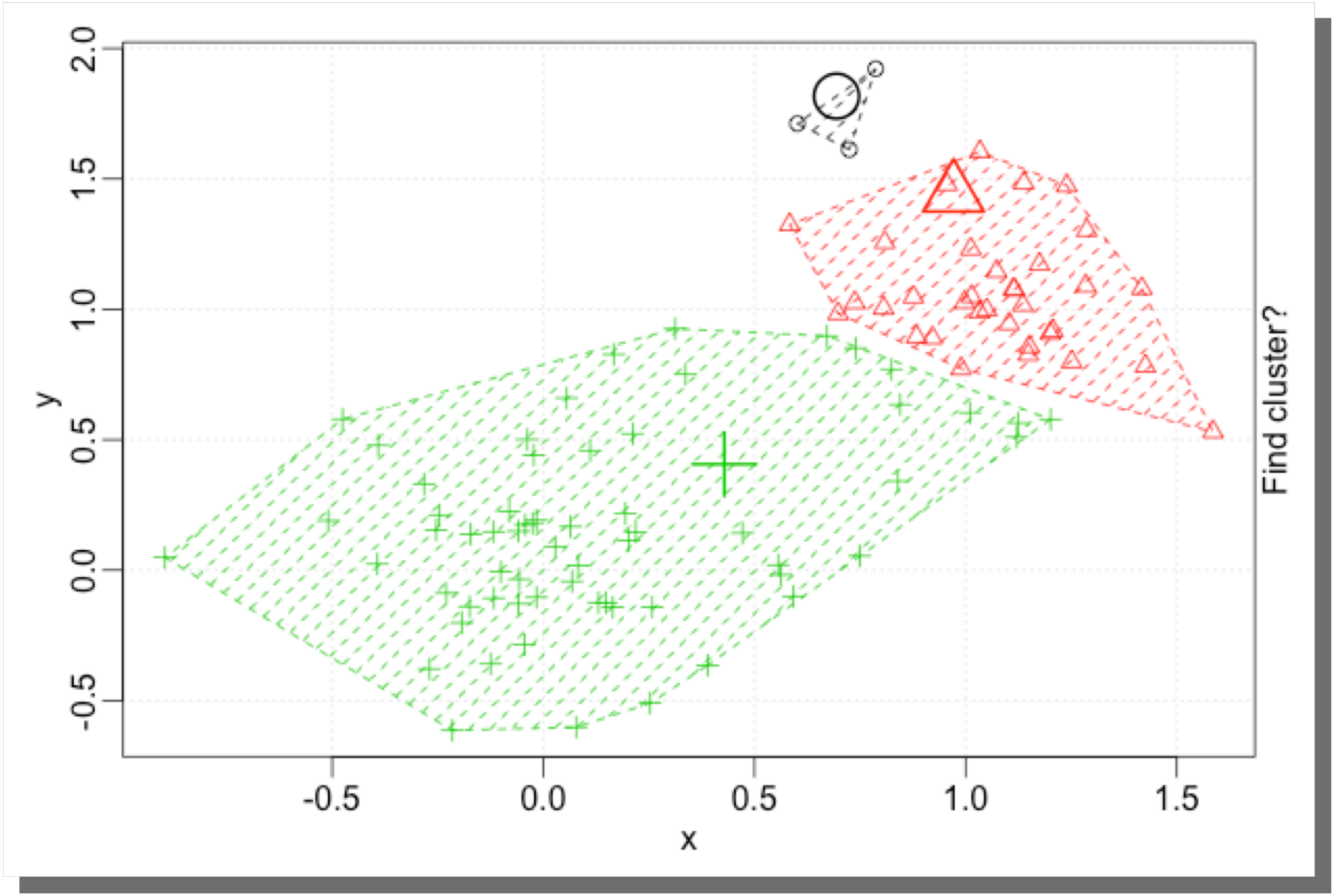
$$\mu_i = \frac{1}{n_i} \sum_{x_j \in C_i} x_j$$

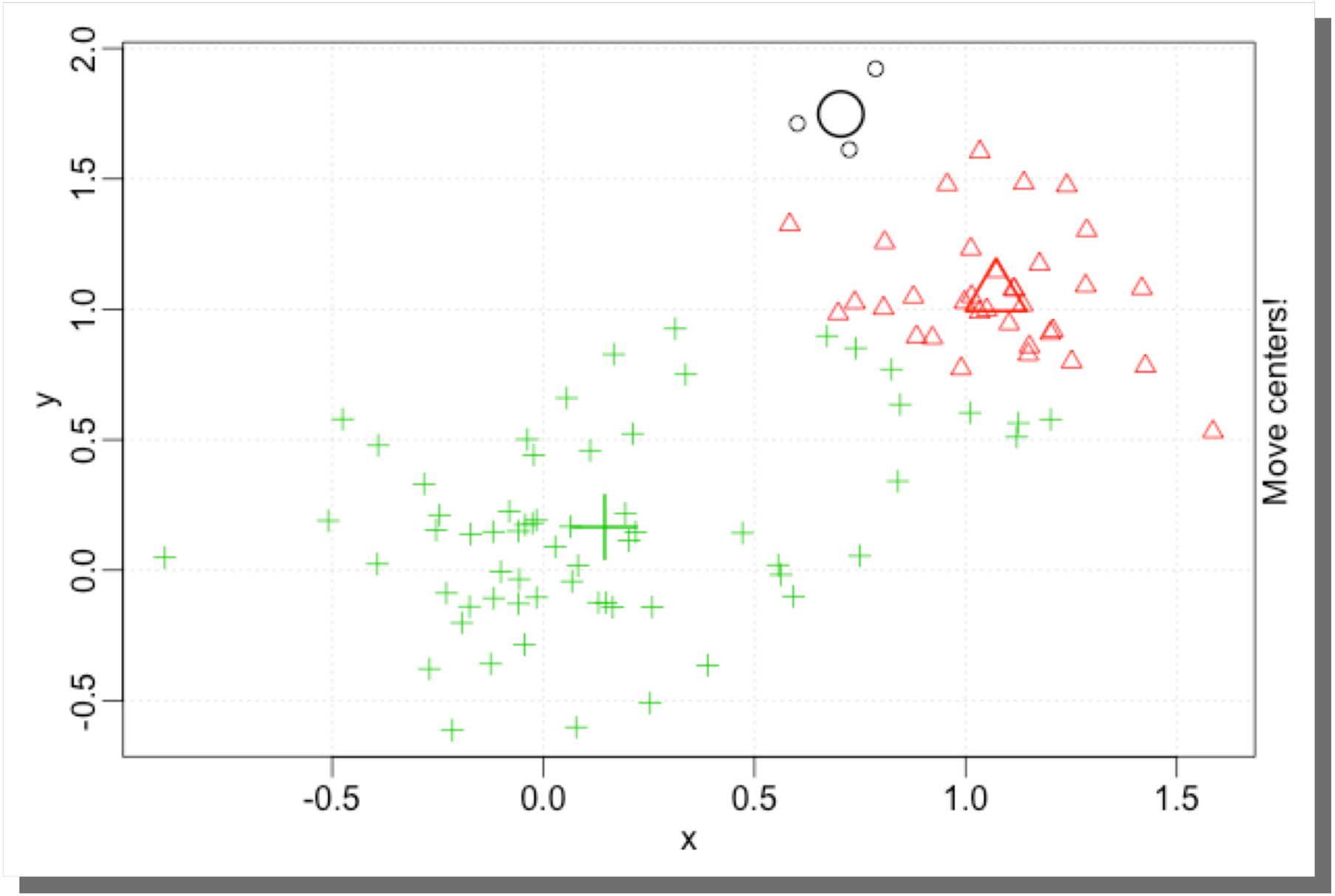
where $n_i = |C_i|$ and $\boldsymbol{\mu}_i$ is the centroid

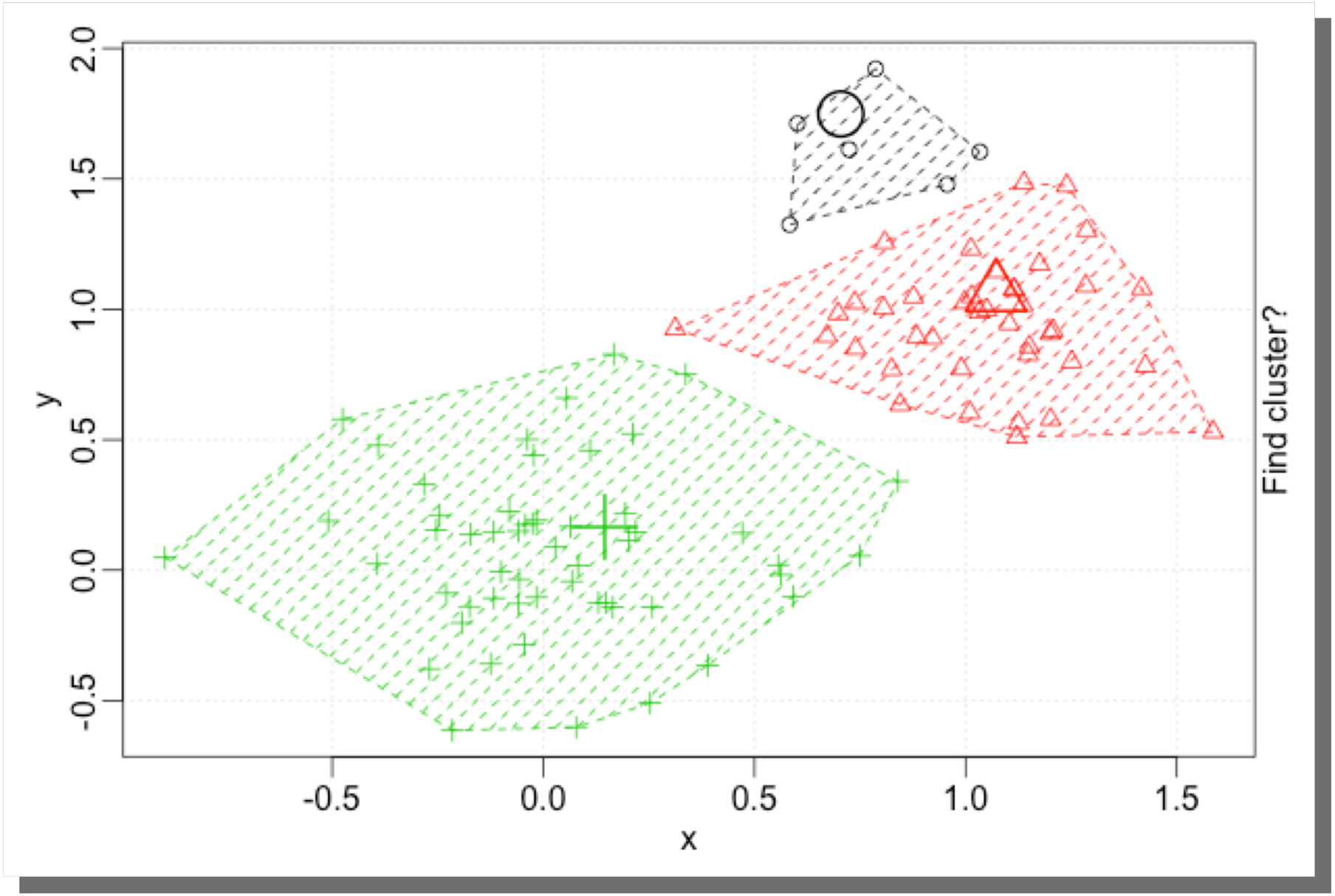

2.0
1.5
1.0
0.5
0.0
-0.5
y
-0.5
0.0
0.5
1.0
1.5
x
Move centers!

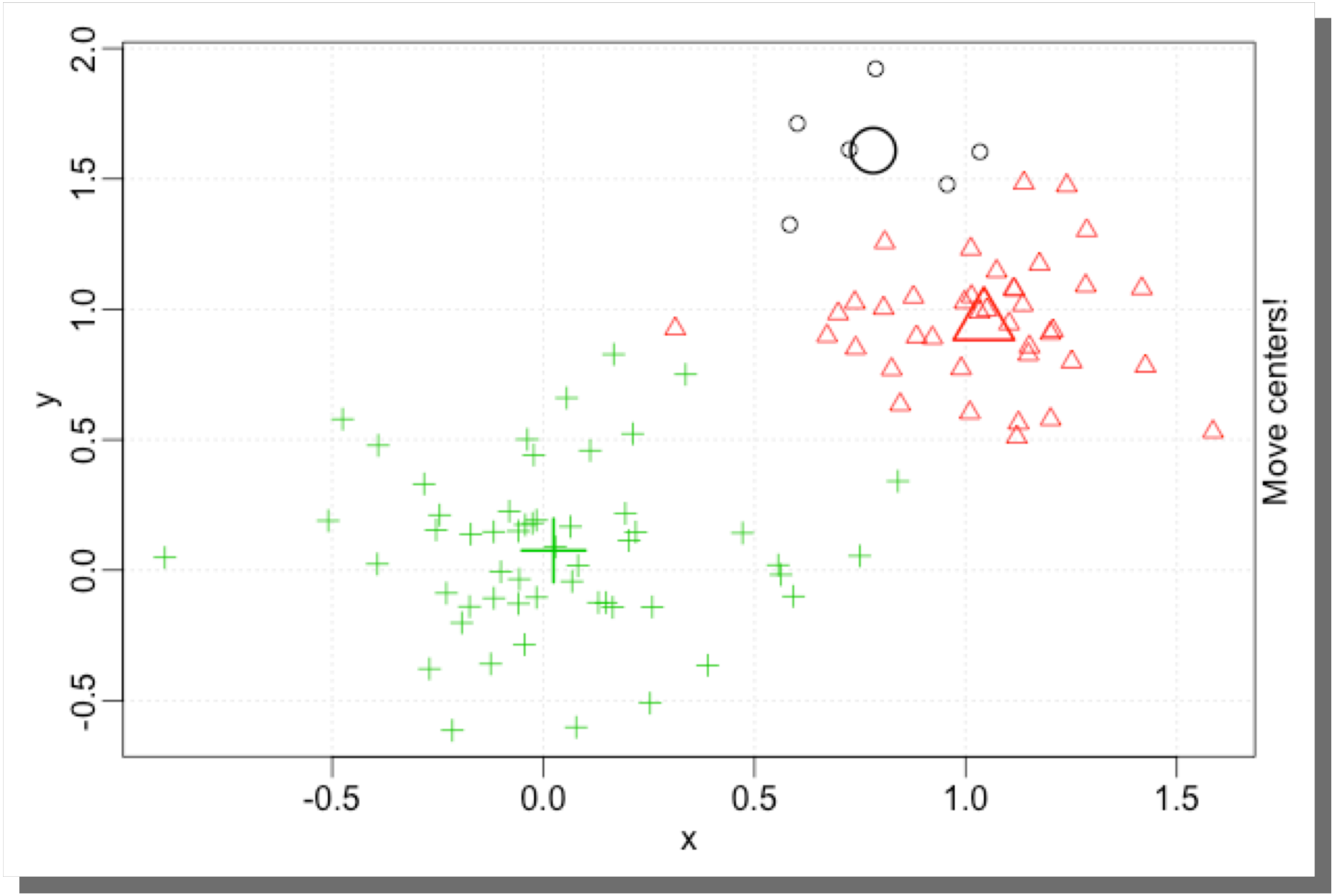

2.0
1.5
1.0
0.5
0.0
-0.5
y
-0.5
0.0
0.5
1.0
1.5
x
Find cluster?

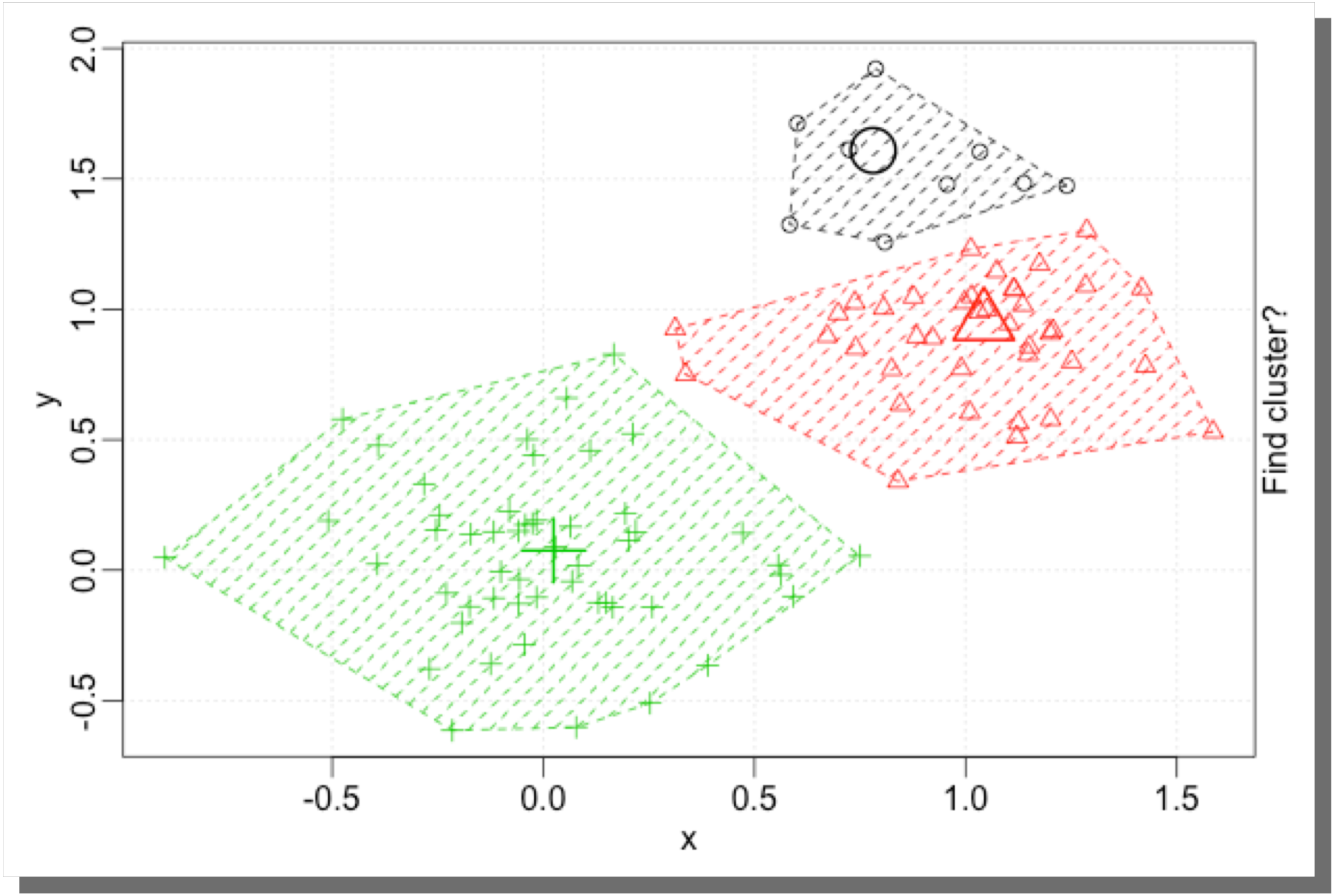

2.0
1.5
1.0
0.5
0.0
-0.5
y
-0.5
0.0
0.5
1.0
1.5
x
Move centers!

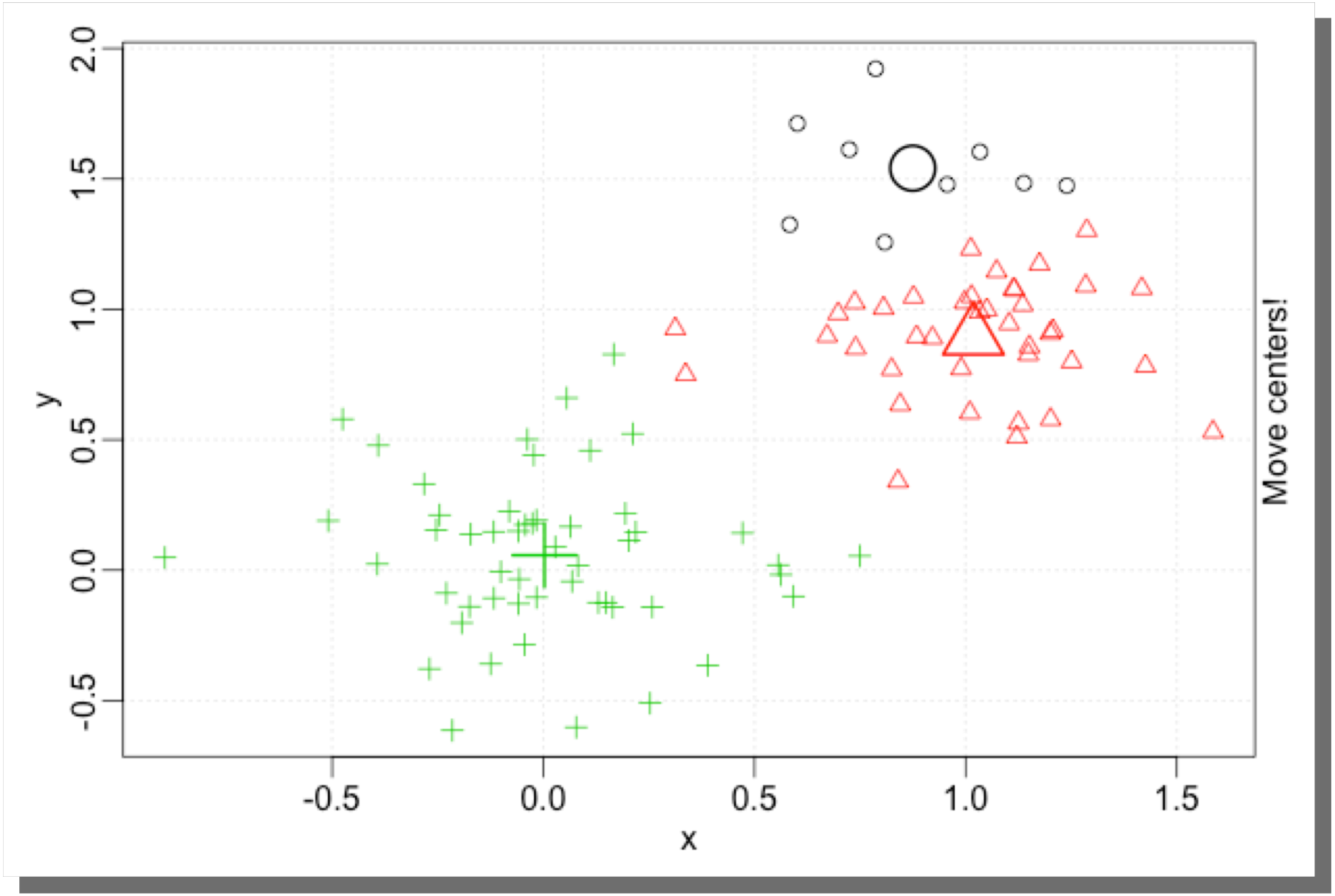

2.0
1.5
1.0
0.5
0.0
-0.5
y
-0.5
0.0
0.5
1.0
1.5
x
Find cluster?

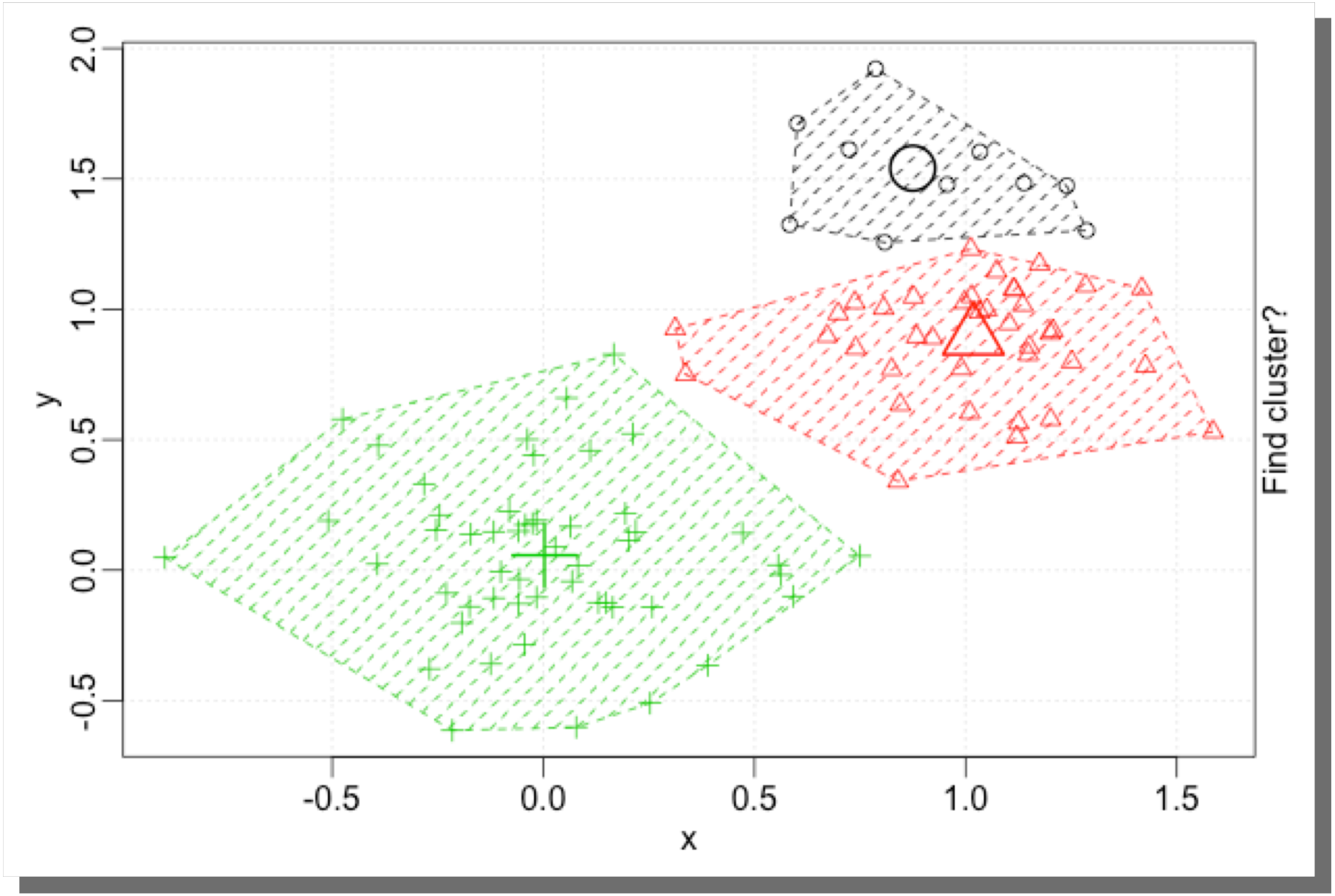

2.0
1.5
1.0
0.5
0.0
-0.5
y
-0.5
0.0
0.5
1.0
1.5
x
Move centers!

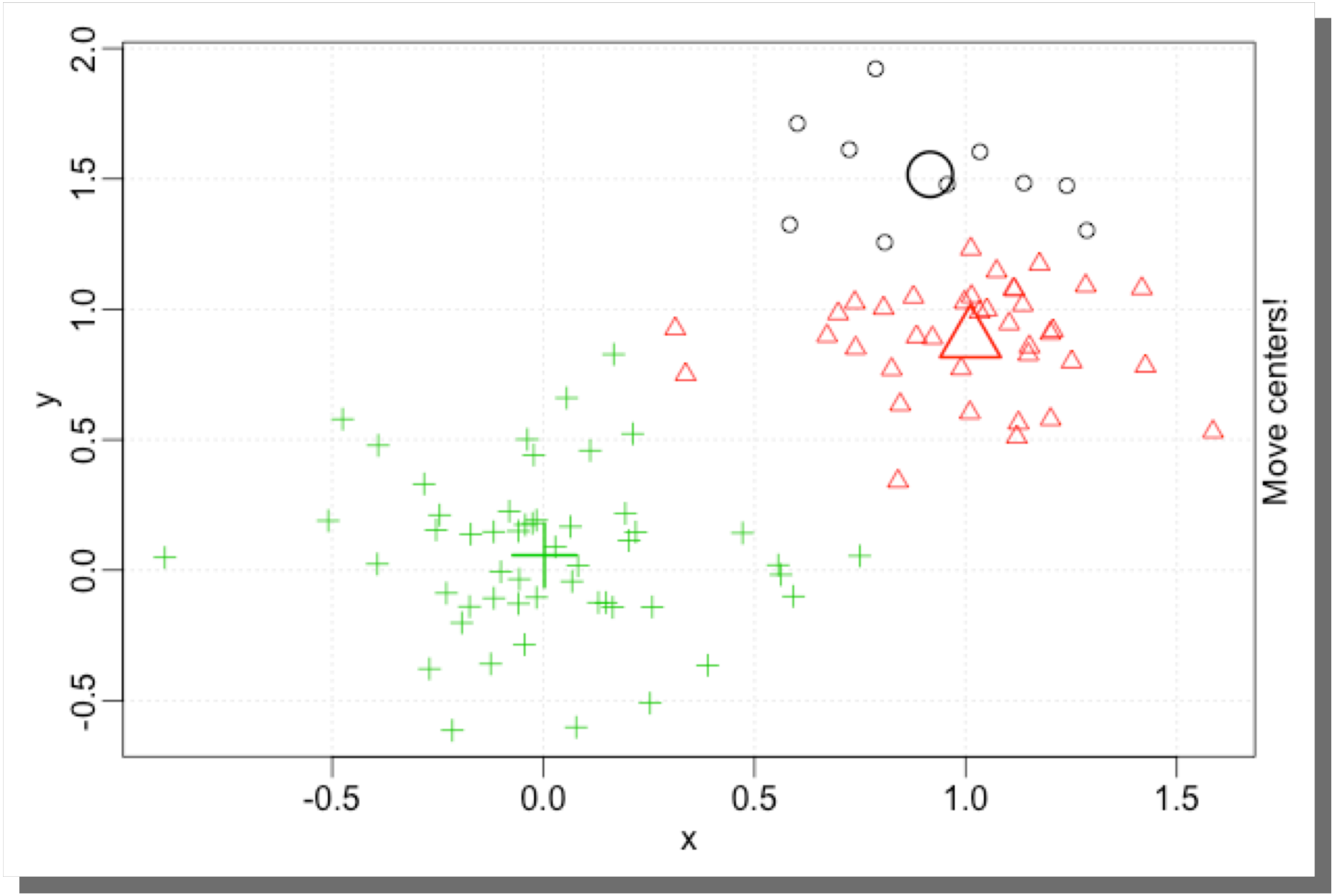

2.0
1.5
1.0
0.5
0.0
-0.5
y
-0.5
0.0
0.5
1.0
1.5
x
Find cluster?

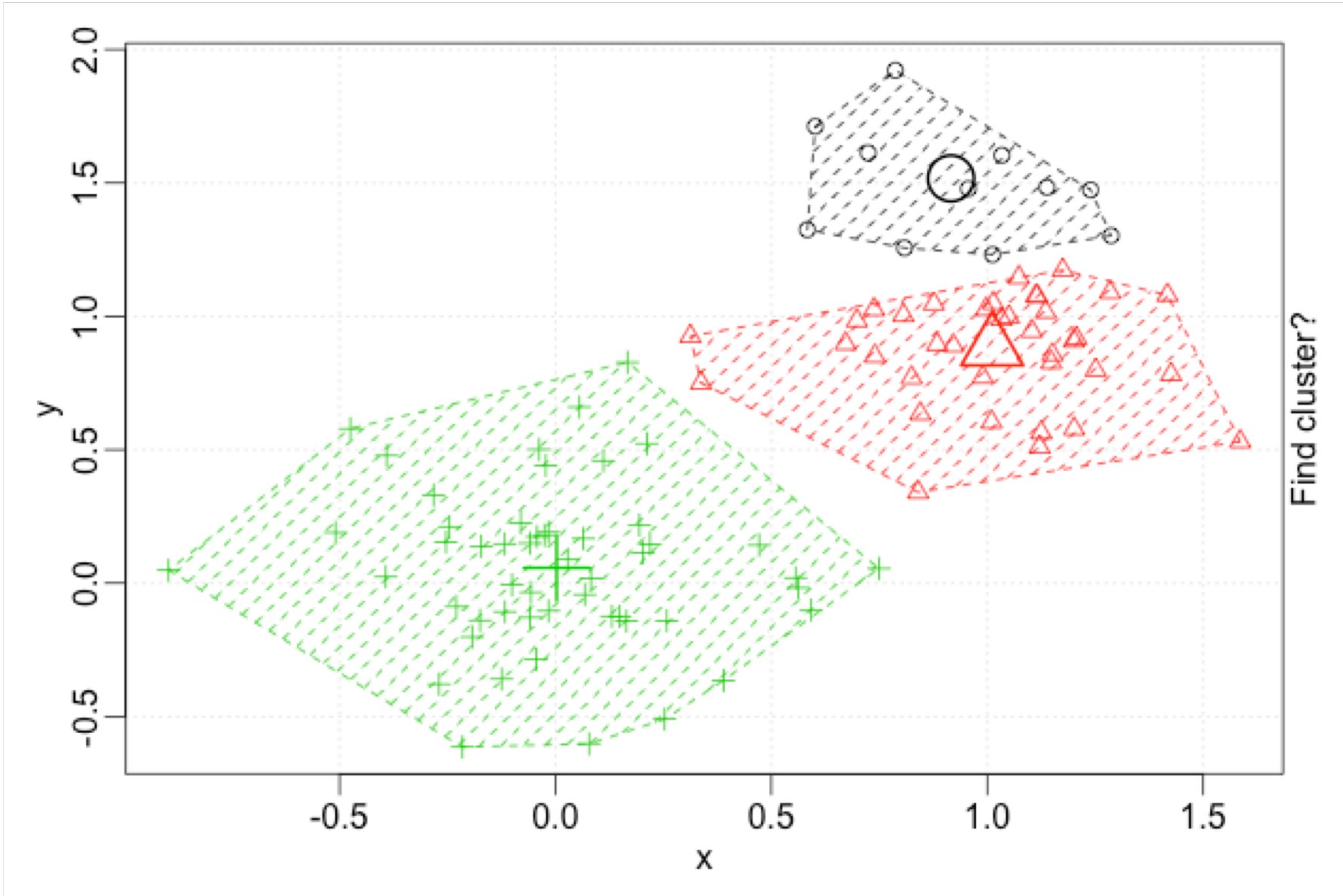

2.0
1.5
1.0
0.5
0.0
-0.5
y
-0.5
0.0
0.5
1.0
1.5
x
Move centers!

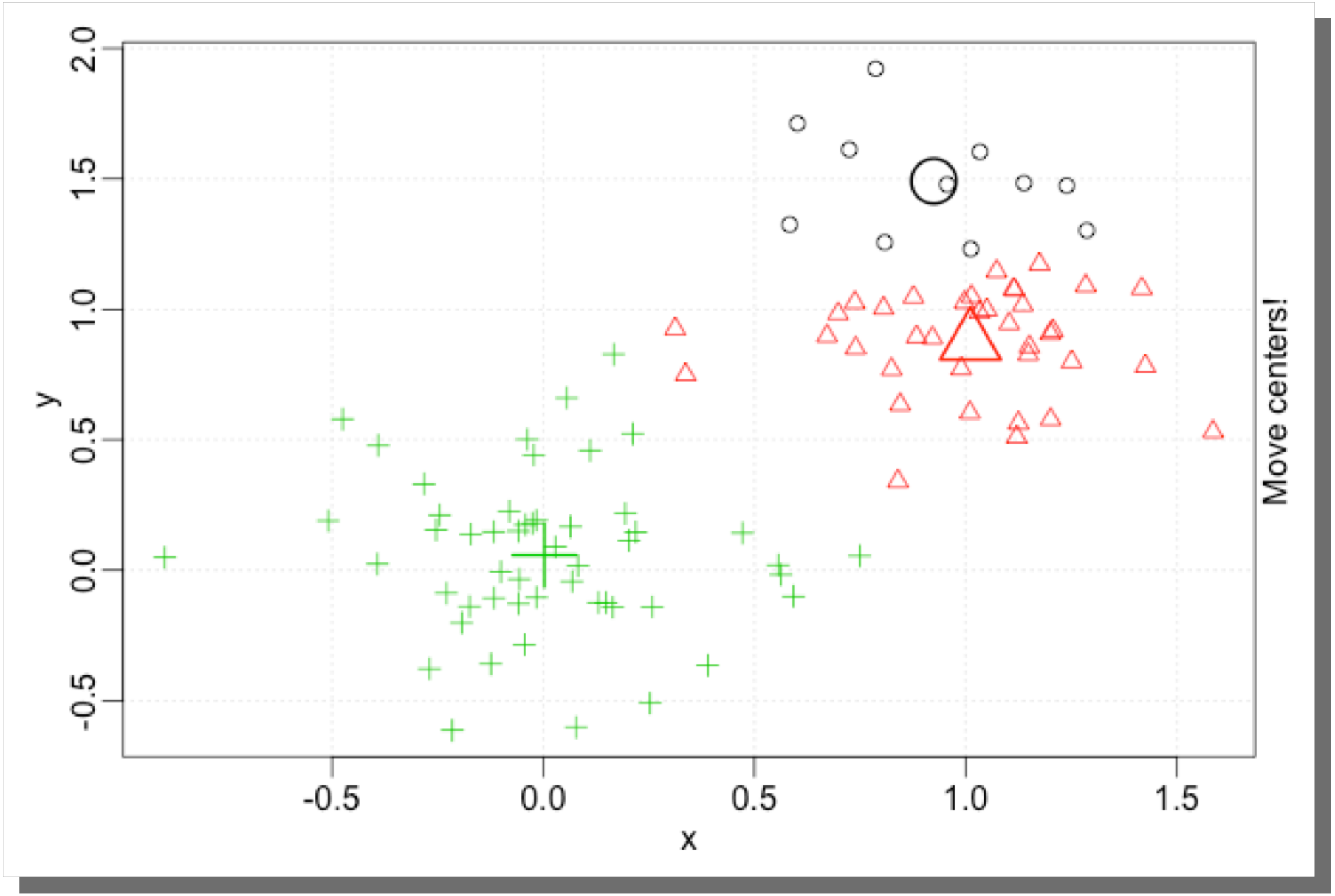

2.0
1.5
1.0
0.5
0.0
-0.5
y
-0.5
0.0
0.5
1.0
1.5
x
Find cluster?

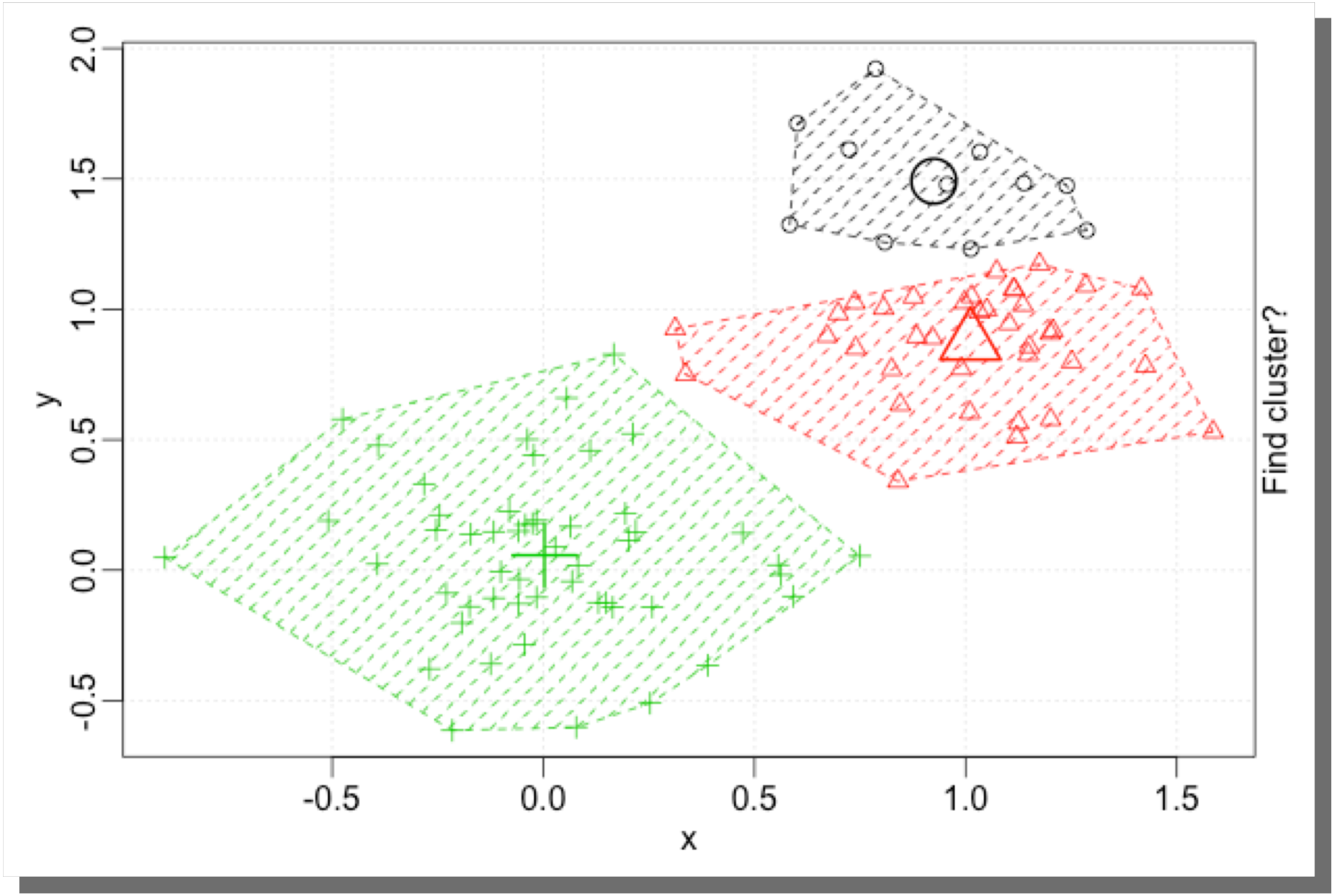

2.0
1.5
1.0
0.5
0.0
-0.5
y
-0.5
0.0
0.5
1.0
1.5
x
Move centers!


2.0
1.5
1.0
0.5
0.0
-0.5
y
-0.5
0.0
0.5
1.0
1.5
x
Find cluster?


2.0
1.5
1.0
0.5
0.0
-0.5
y
-0.5
0.0
0.5
1.0
1.5
x
Move centers!


2.0
1.5
1.0
0.5
0.0
-0.5
y
-0.5
0.0
0.5
1.0
1.5
x
Find cluster?


2.0
1.5
1.0
0.5
0.0
-0.5
y
-0.5
0.0
0.5
1.0
1.5
x
Move centers!


2.0
1.5
1.0
0.5
0.0
-0.5
y
-0.5
0.0
0.5
1.0
1.5
x
Find cluster?

Representation-Based Algorithms

- The goal of the clustering process is to select the best partition according to some scoring function
- Sum of squared errors is the most common scoring function

$$\mathrm{SSE}(C) = \sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_j - \mu_i||^2$$

- The goal of the clustering process is thus to find

$$C^\star = \arg\min_{C} \mathrm{SSE}(C)$$

- Brute-force Approach
  - Generate all the possible clustering $C = \{C_1, C_2, \ldots, C_k\}$ and select the best one. Unfortunately, there are $O(k^N/k!)$ possible partitions

k-Means Algorithm

- Most widely known representative-based algorithm
- Assumes an Euclidean space but can be easily extended to the non-Euclidean case
- Employs a greedy iterative approaches that minimizes the SSE objective. Accordingly it can converge to a local optimal instead of a globally optimal clustering.

```
1. Initially choose k points that are
   likely to be in different clusters;

2. Make these points the centroids of
   their clusters;

3. FOR each remaining point p DO
     Find the centroid to which p is closest;
     Add p to the cluster of that centroid;
     Adjust the centroid of that
     cluster to account for p;
   END;
```

K-MEANS ($\mathbf{D}, k, \epsilon$):

$t = 0$

Randomly initialize k centroids: $\boldsymbol{\mu}_1^t, \boldsymbol{\mu}_2^t, \ldots, \boldsymbol{\mu}_k^t \in \mathbb{R}^d$

repeat

  $t \leftarrow t + 1$

  $C_j \leftarrow \emptyset$ for all $j = 1, \cdots, k$

  `// Cluster Assignment Step`

  foreach $\mathbf{x}_j \in \mathbf{D}$ **do**

    $j^* \leftarrow \arg\min_i \left\{ \left\| \mathbf{x}_j - \boldsymbol{\mu}_i^t \right\|^2 \right\}$ `// Assign` $\mathbf{x}_j$ `to closest centroid`

    $C_{j^*} \leftarrow C_{j^*} \cup \{\mathbf{x}_j\}$

  `// Centroid Update Step`

  foreach $i = 1$ *to* k **do**

    $\boldsymbol{\mu}_i^t \leftarrow \frac{1}{|C_i|} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j$

until $\sum_{i=1}^{k} \left\| \boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1} \right\|^2 \leq \epsilon$

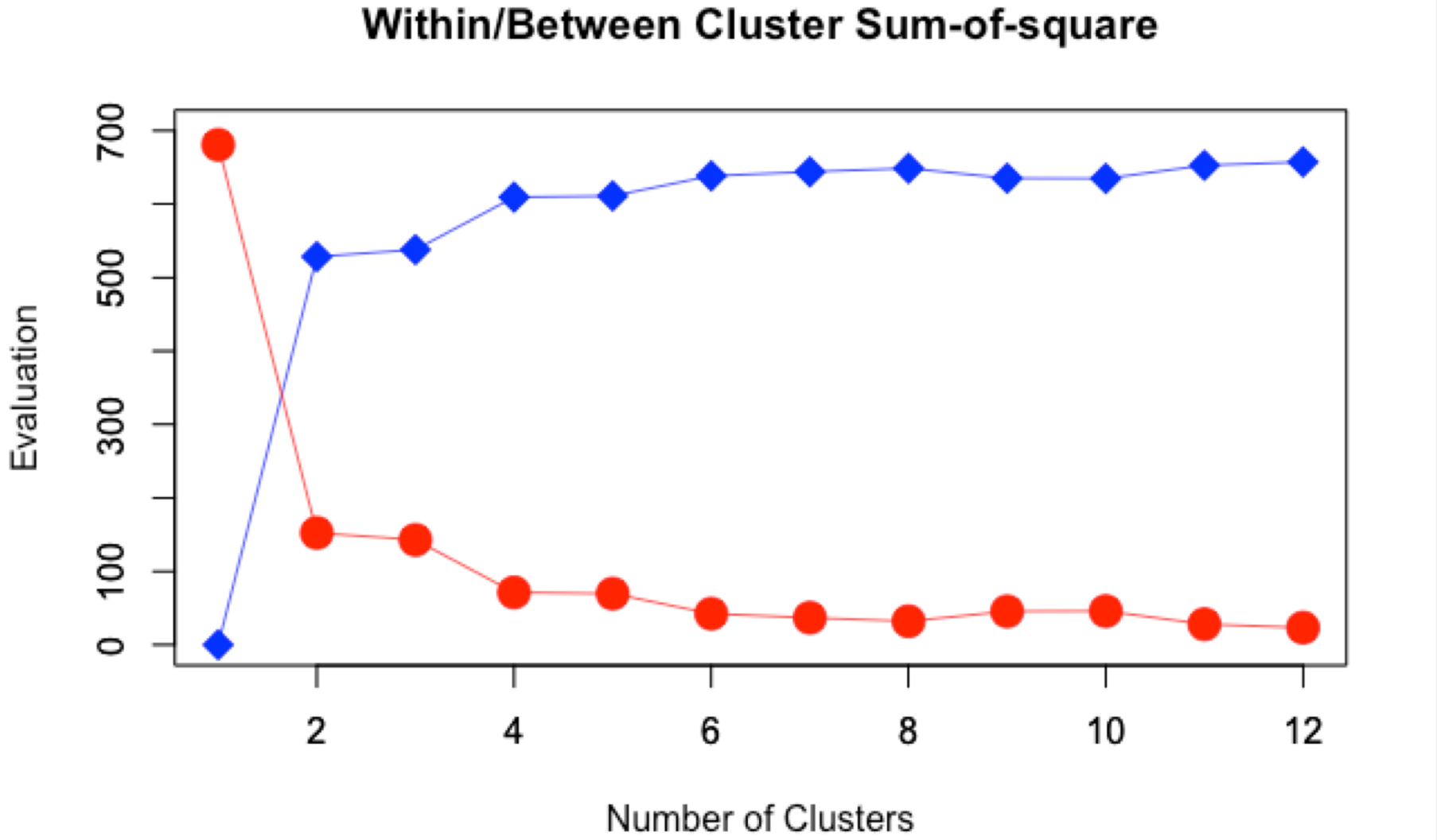
Initializing Clusters

- Solution 1
  - Pick points that are as far away from one another as possible.
- Variation of solution 1

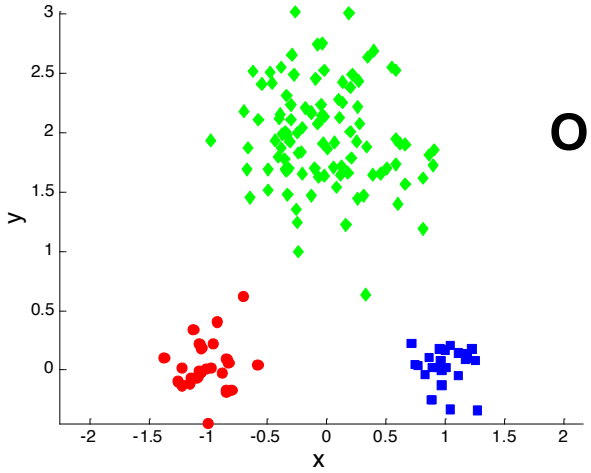
```
Pick the first point at random;
WHILE there are fewer than k points DO
    Add the point whose minimum distance
    from the selected points is as large as
    possible;
END;
```

- Solution 2
  - Cluster a sample of the data, perhaps hierarchically, so there are k clusters. Pick a point from each cluster, perhaps that point closest to the centroid of the cluster.

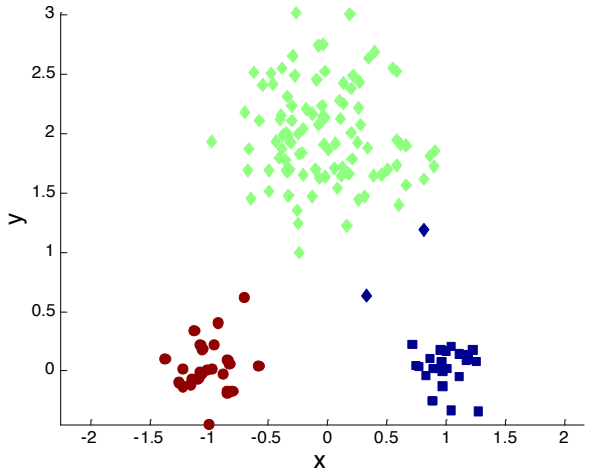
Elbow & Knee Analysis



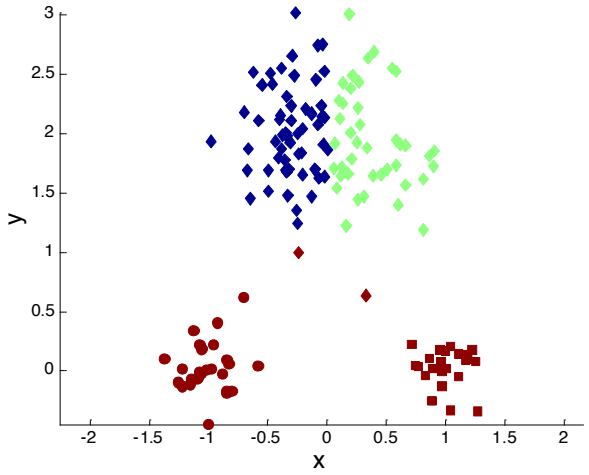
Two different K-means Clusterings



Original Points

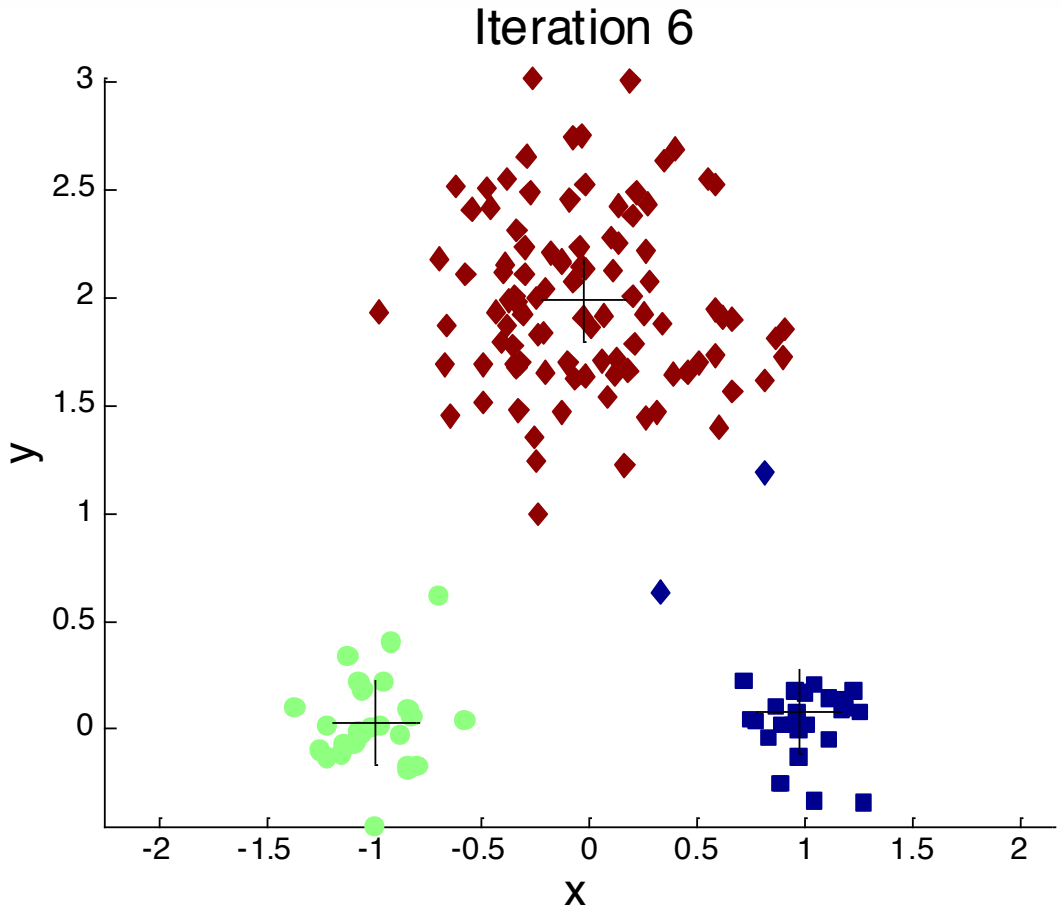


Optimal Clustering

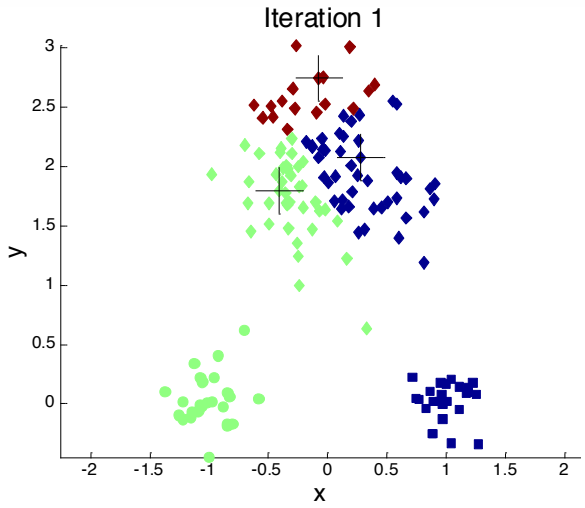


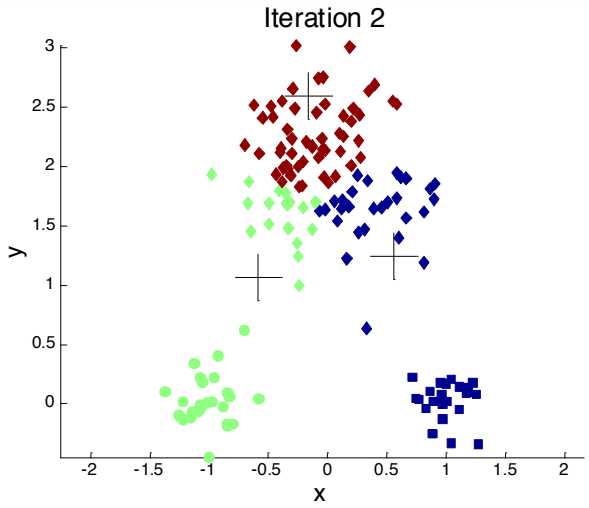
Sub-optimal Clustering

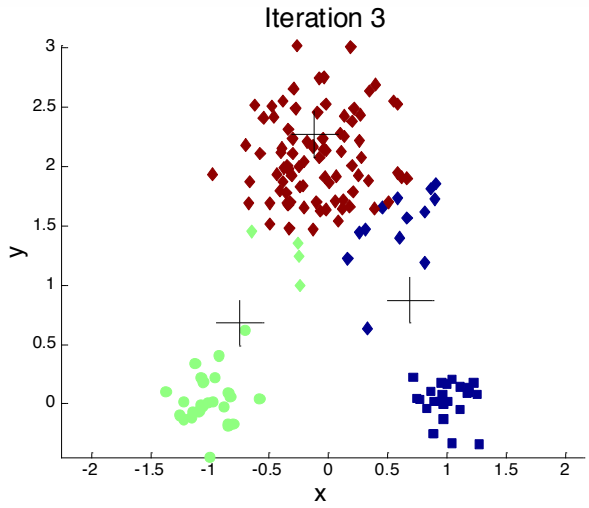
Importance of Choosing the Initial Centroids

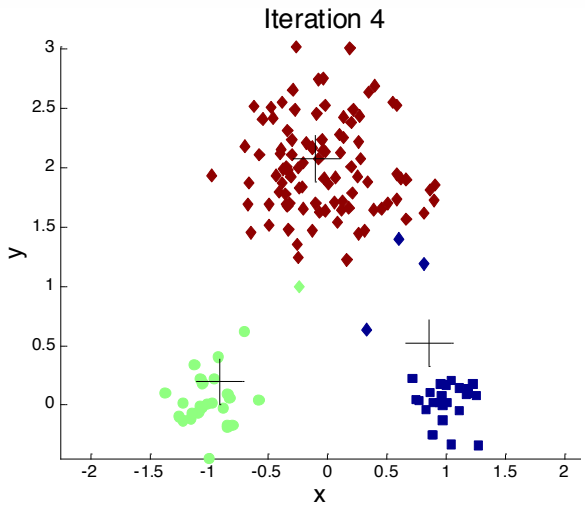


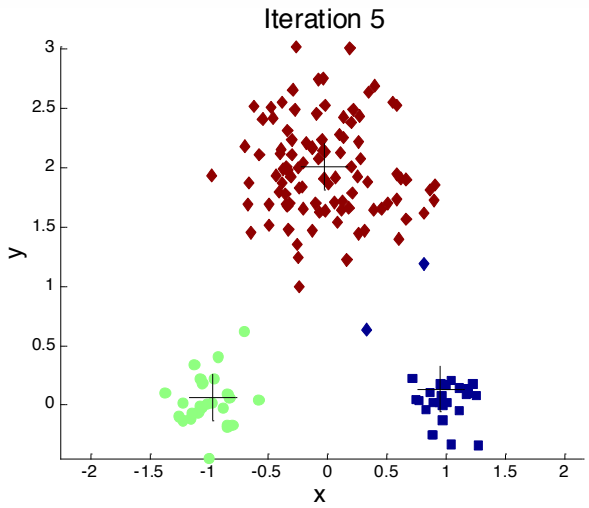
Importance of Choosing the Initial Centroids

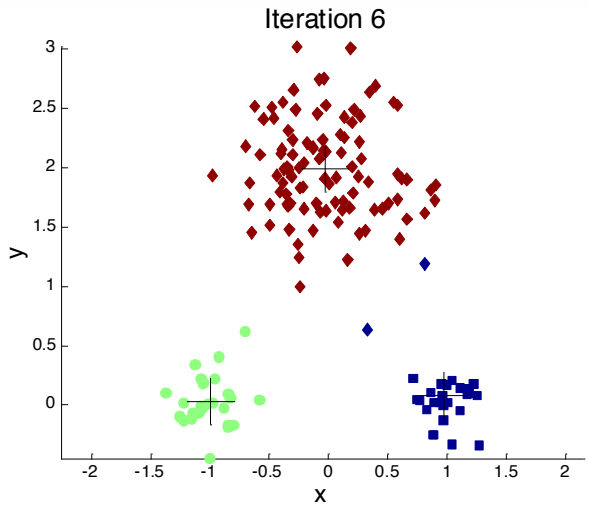




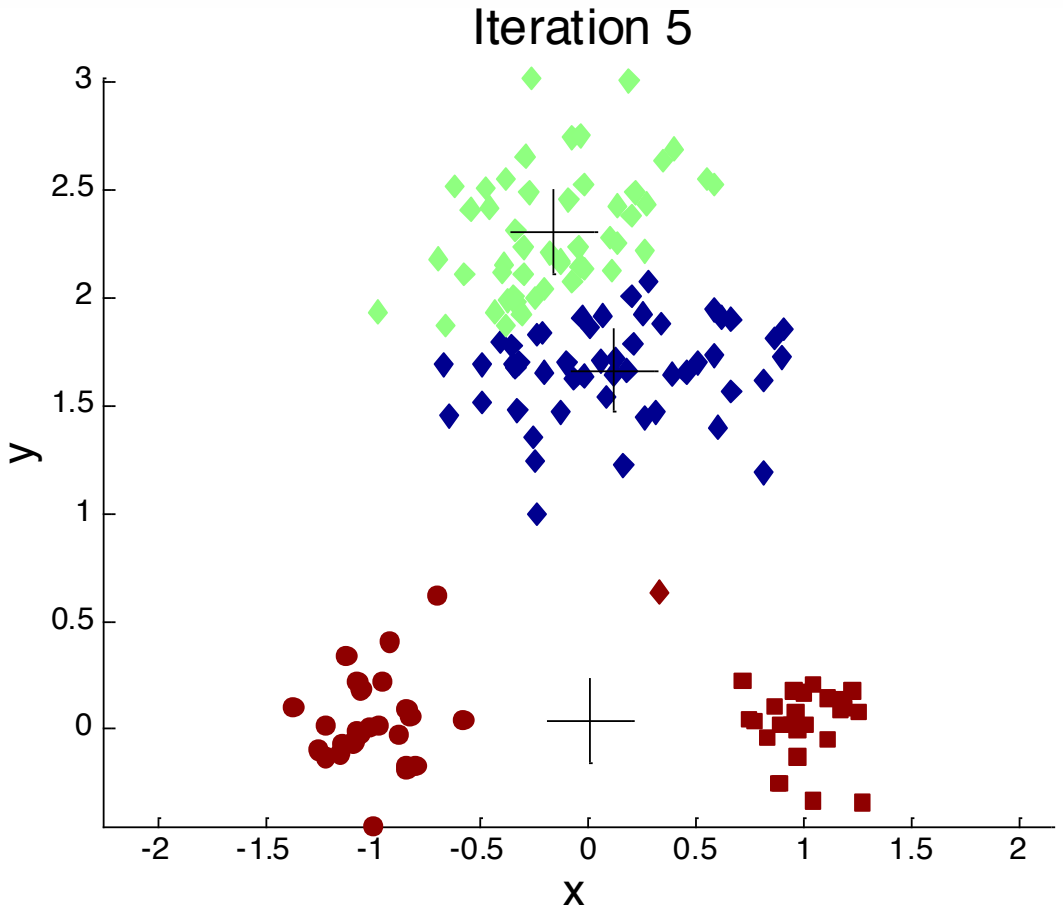




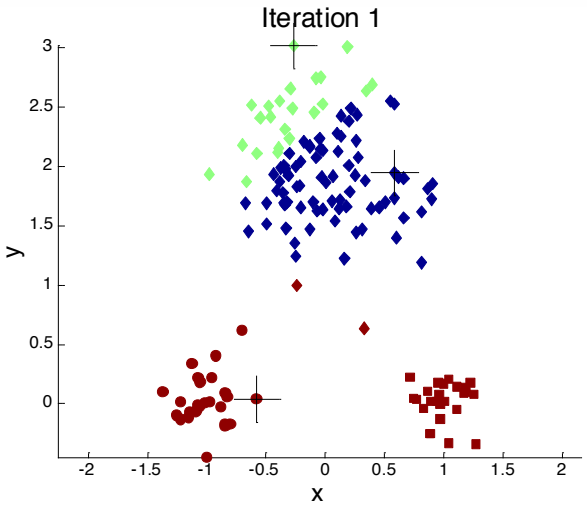


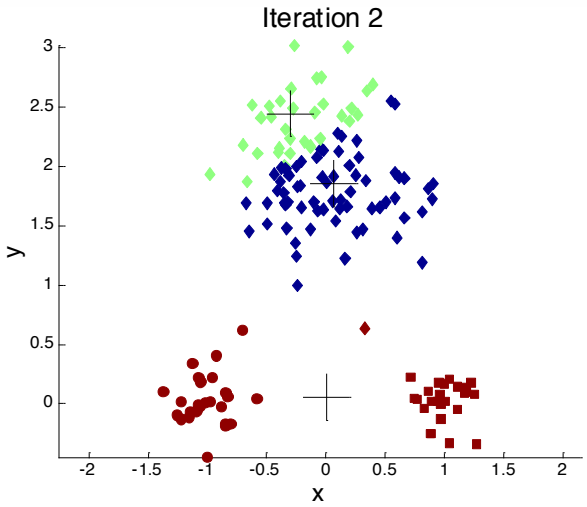


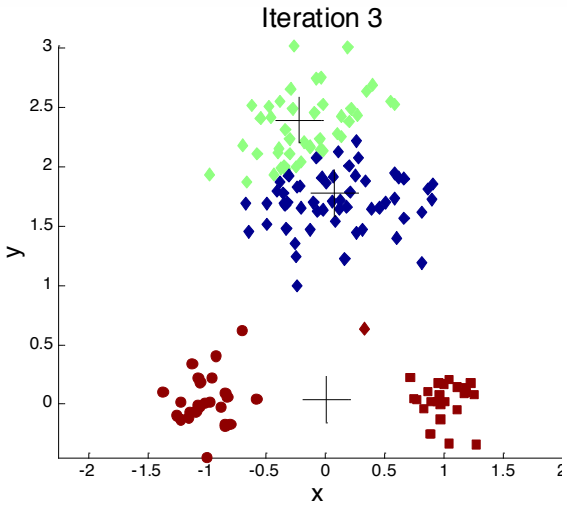
Importance of Choosing the Initial Centroids

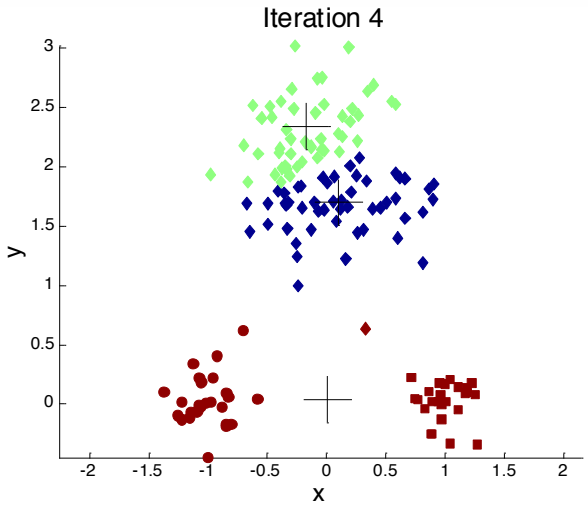


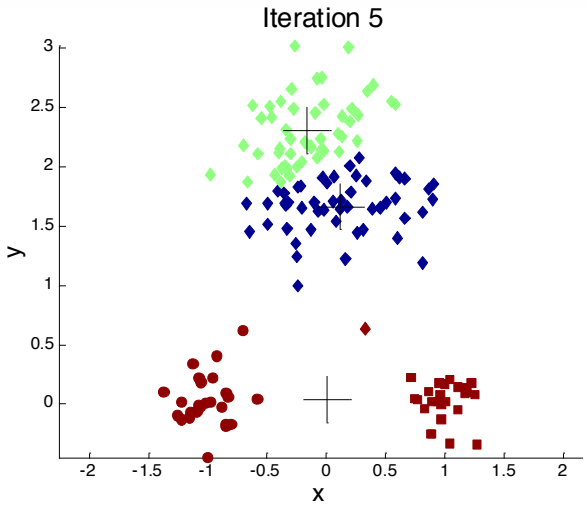
Importance of Choosing the Initial Centroids











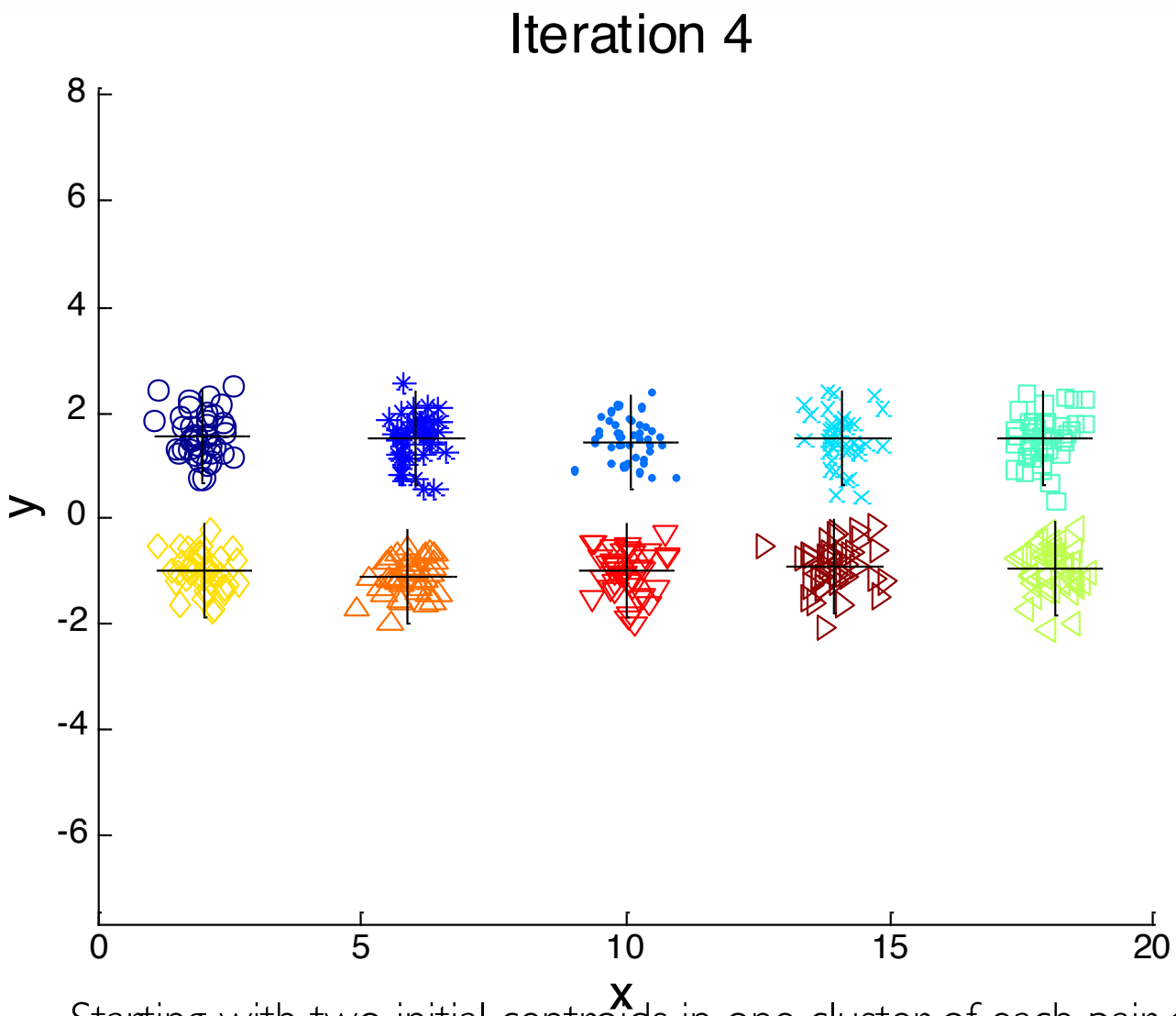
Why Selecting the Best Initial Centroids is Difficult?

- If there are K 'real' clusters then the chance of selecting one centroid from each cluster is small.
- Chance is relatively small when K is large
- If clusters are the same size, n, then

$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K!n^K}{(Kn)^K} = \frac{K!}{K^K}$$

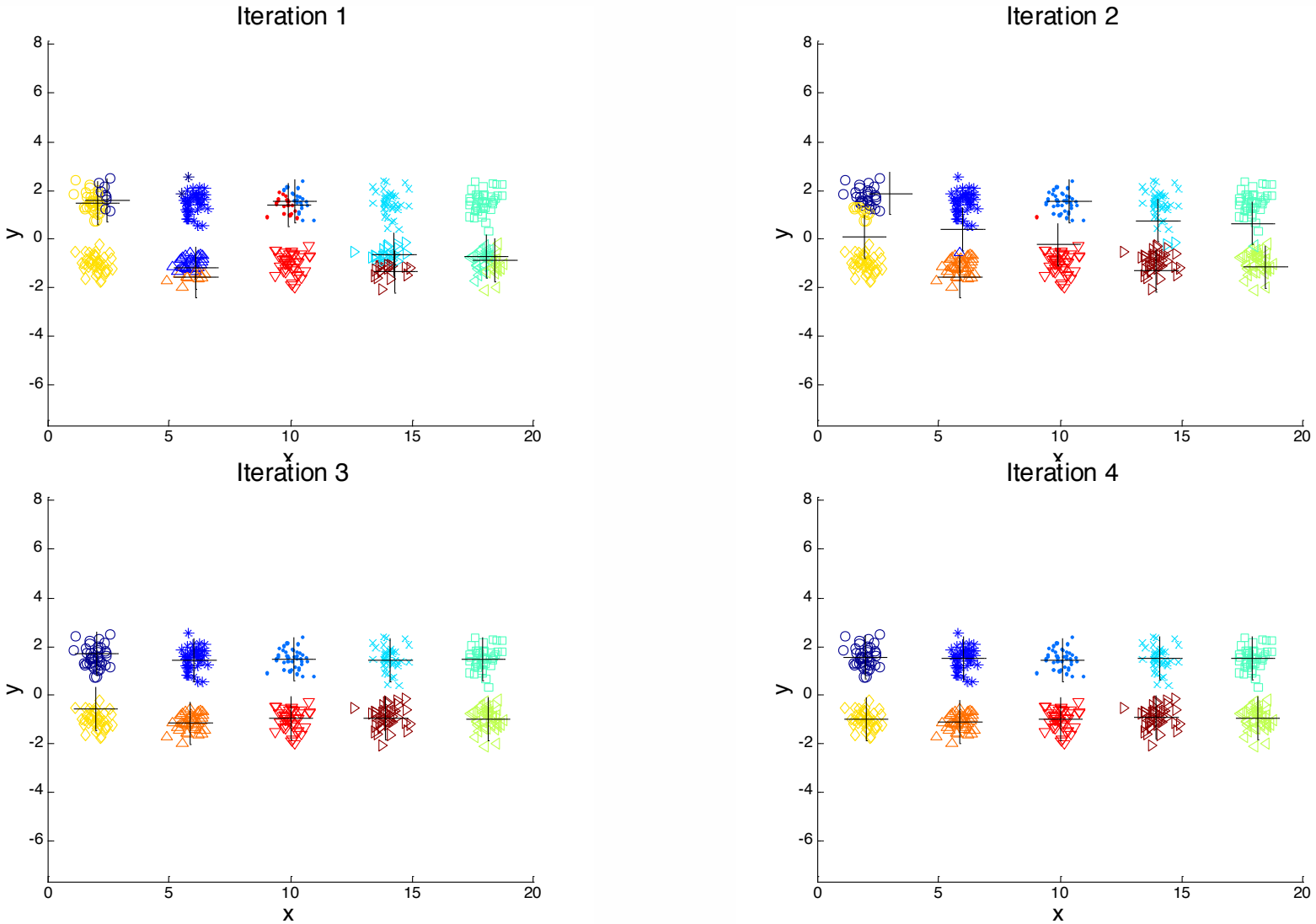
- For example, if K = 10, then probability = 10!/1010 = 0.00036
- Sometimes the initial centroids will readjust themselves in 'right' way, and sometimes they don't
- Consider an example of five pairs of clusters

10 Clusters Example



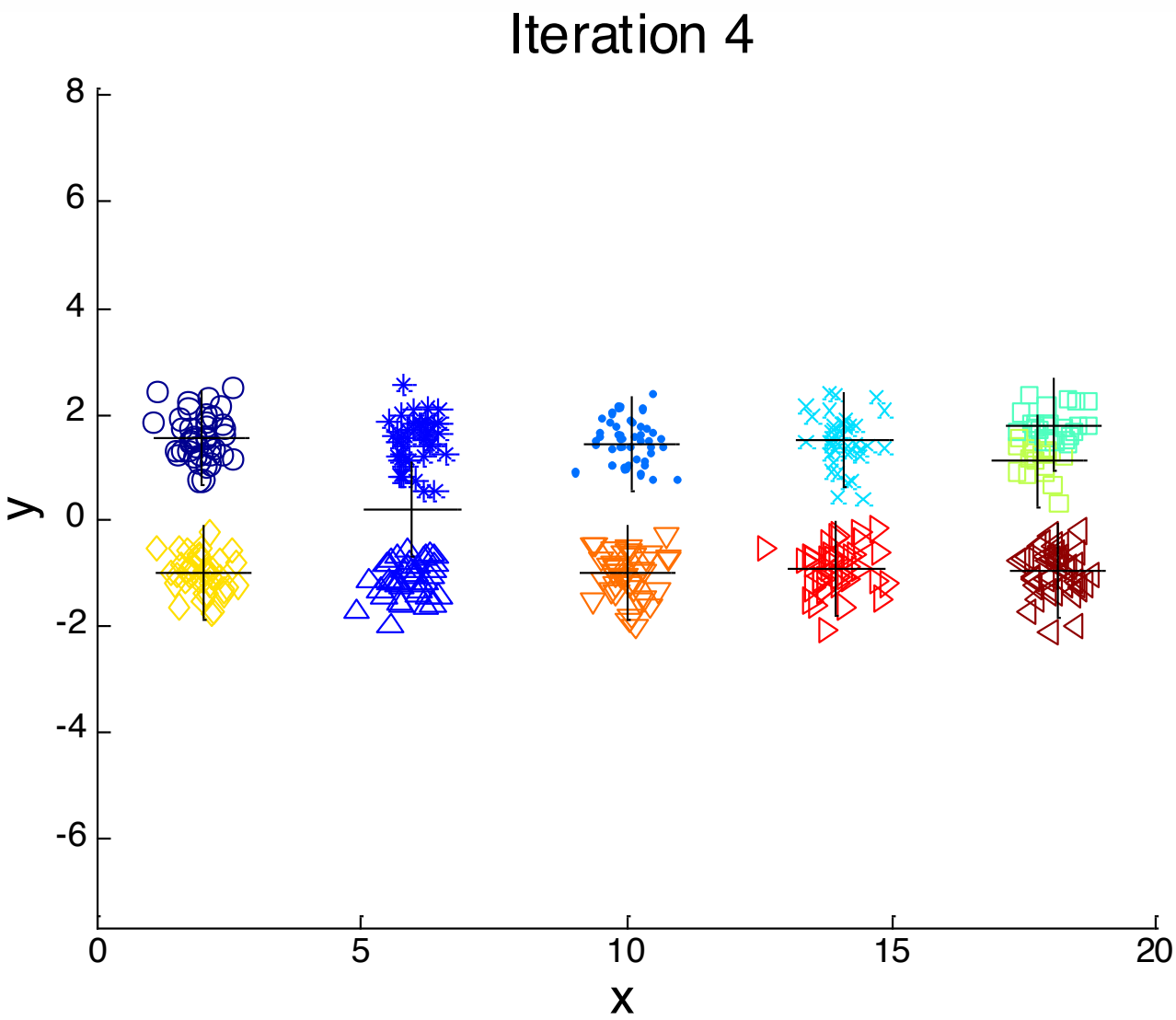
Starting with two initial centroids in one cluster of each pair of clusters

10 Clusters Example



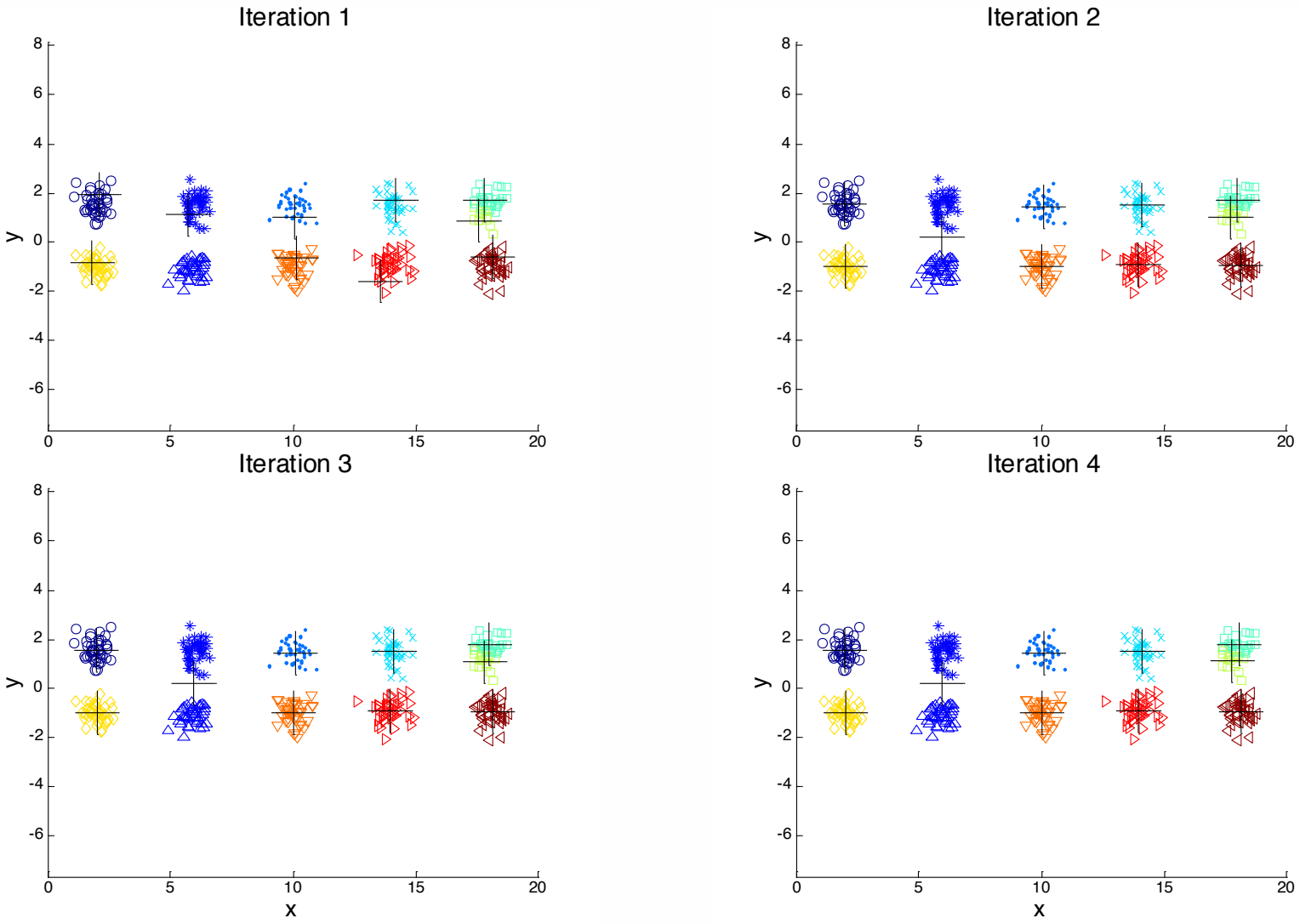
Starting with two initial centroids in one cluster of each pair of clusters

10 Clusters Example



Starting with some pairs of clusters having three initial centroids, while other have only one.

10 Clusters Example



Starting with some pairs of clusters having three initial centroids, while other have only one.

Dealing with the Initial Centroids Issue

- Multiple runs, helps, but probability is not on your side
- Sample and use another clustering method (hierarchical?) to determine initial centroids
- Select more than k initial centroids and then select among these initial centroids
- Postprocessing
- Bisecting K-means, not as susceptible to initialization issues

Updating Centers Incrementally

- In the basic K-means algorithm, centroids are updated after all points are assigned to a centroid
- An alternative is to update the centroids after each assignment (incremental approach)
  - Each assignment updates zero or two centroids
  - More expensive
  - Introduces an order dependency
  - Never get an empty cluster
  - Can use "weights" to change the impact

Pre-processing and Post-processing

- Pre-processing
  - Normalize the data
  - Eliminate outliers
- Post-processing
  - Eliminate small clusters that may represent outliers
  - Split 'loose' clusters, i.e., clusters with relatively high SSE
  - Merge clusters that are 'close' and that have relatively low SSE
  - These steps can be used during the clustering process

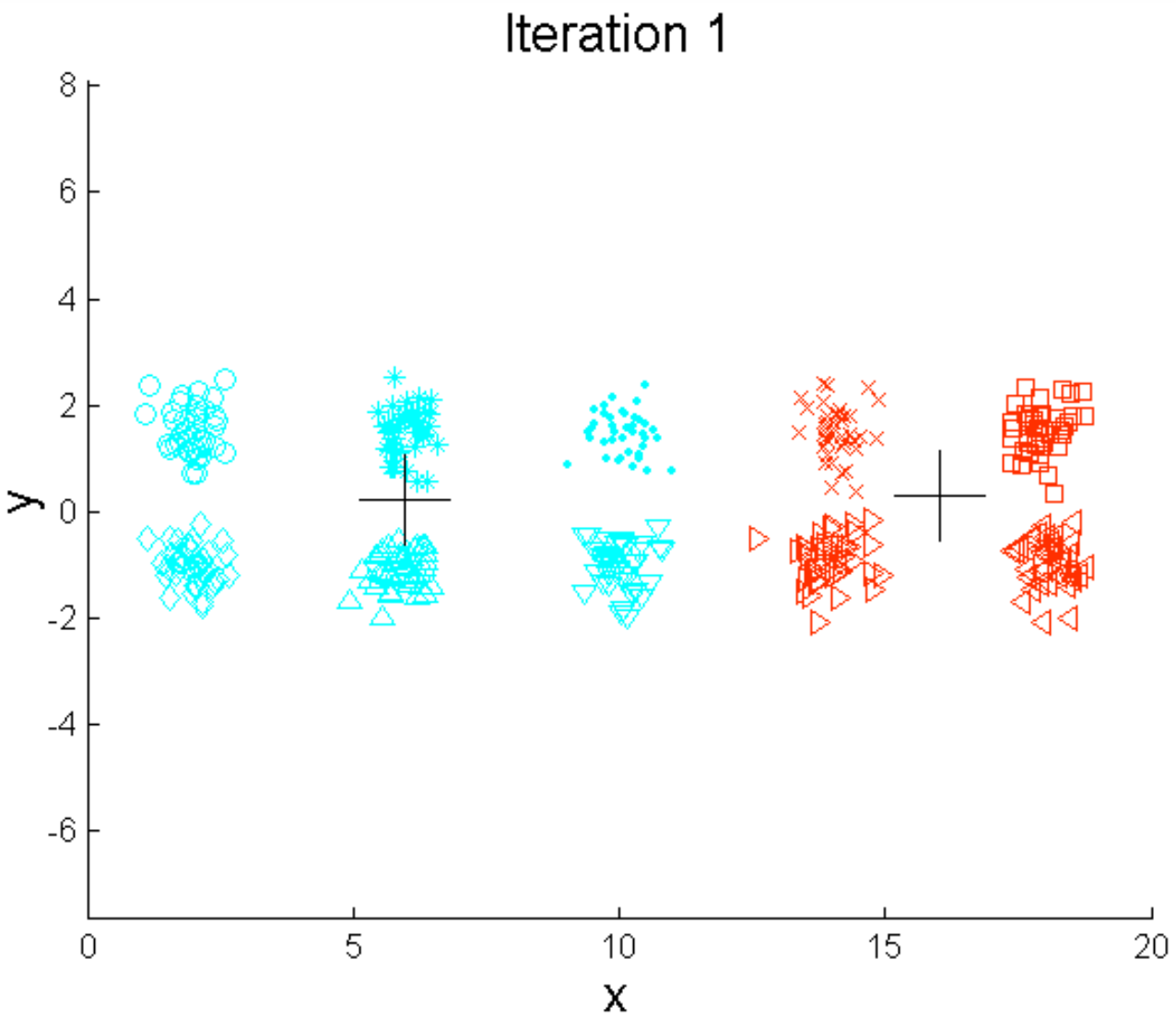
Bisecting K-means

- Variant of K-means that can produce a partitional or a hierarchical clustering

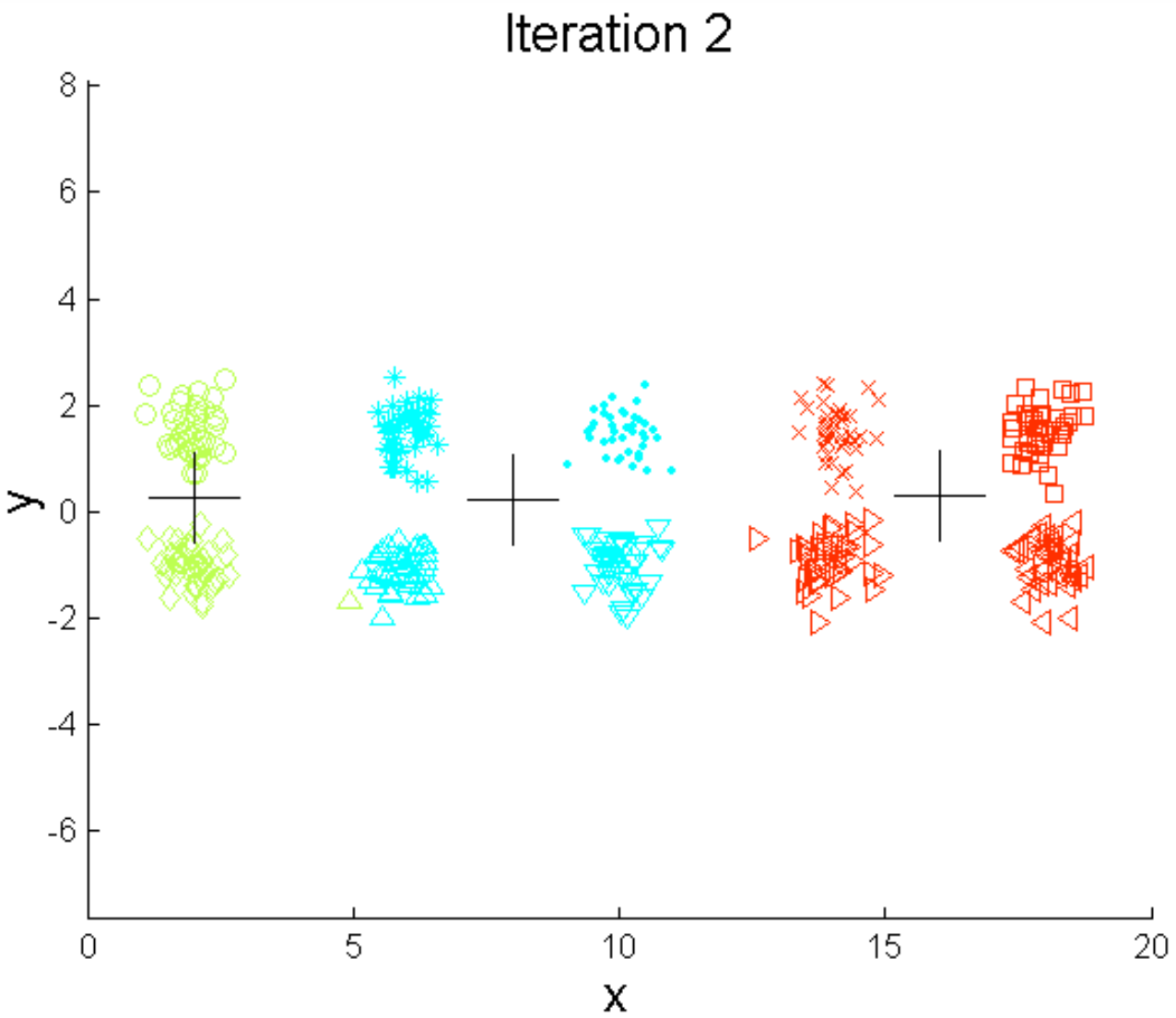
Algorithm 3 Bisecting K-means Algorithm.

1: Initialize the list of clusters to contain the cluster containing all points.
2: **repeat**
3: Select a cluster from the list of clusters
4: **for** $i = 1$ to $number\_of\_iterations$ **do**
5: Bisect the selected cluster using basic K-means
6: **end for**
7: Add the two clusters from the bisection with the lowest SSE to the list of clusters.
8: **until** Until the list of clusters contains K clusters

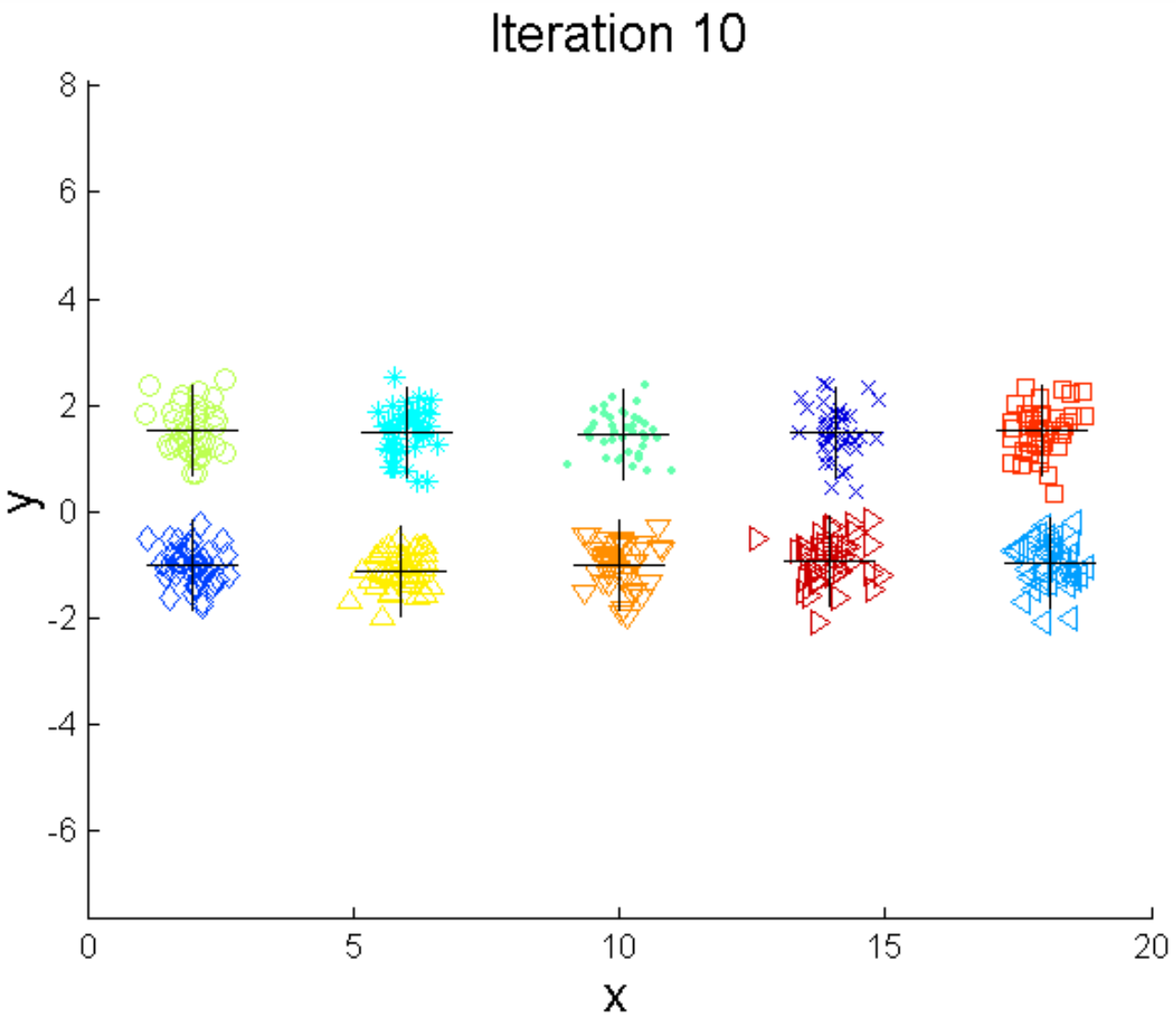
Bisecting K-means Example



Bisecting K-means Example



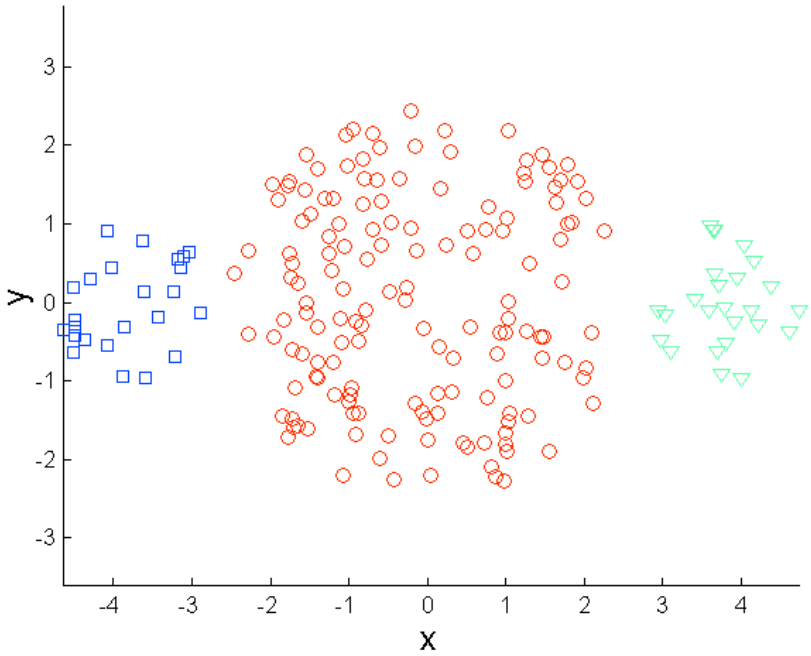
Bisecting K-means Example



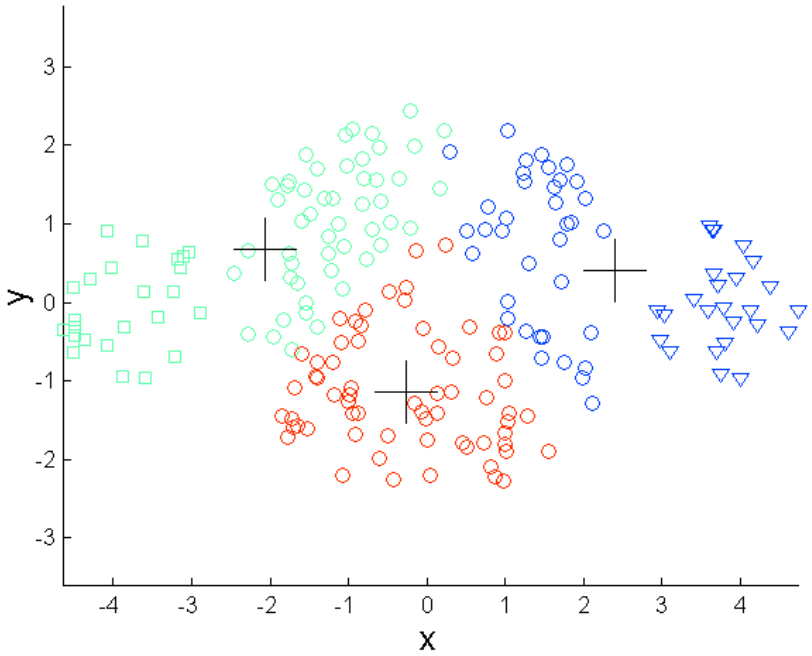
Limitations of K-means

- K-means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes
- K-means has also problems when the data contains outliers.

Limitations of K-means: Differing Sizes

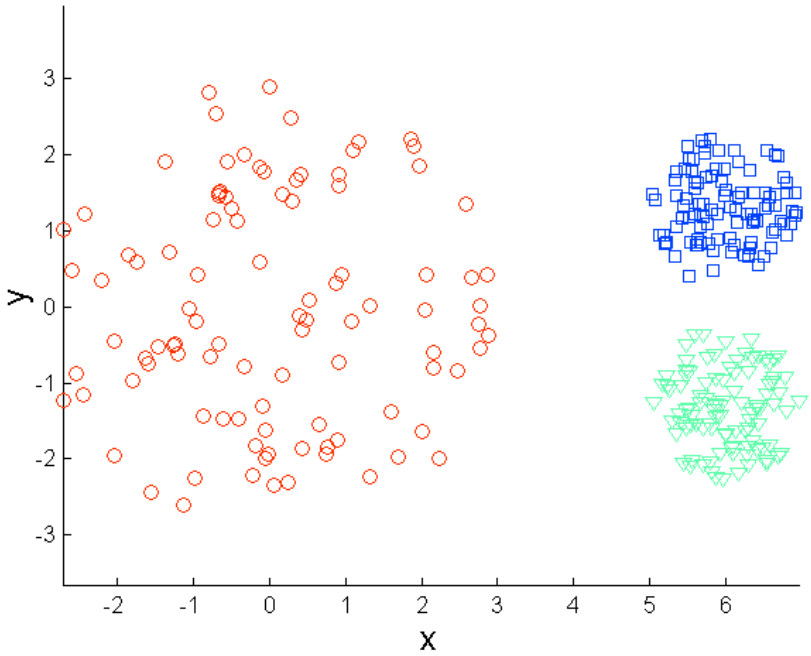


Original Points

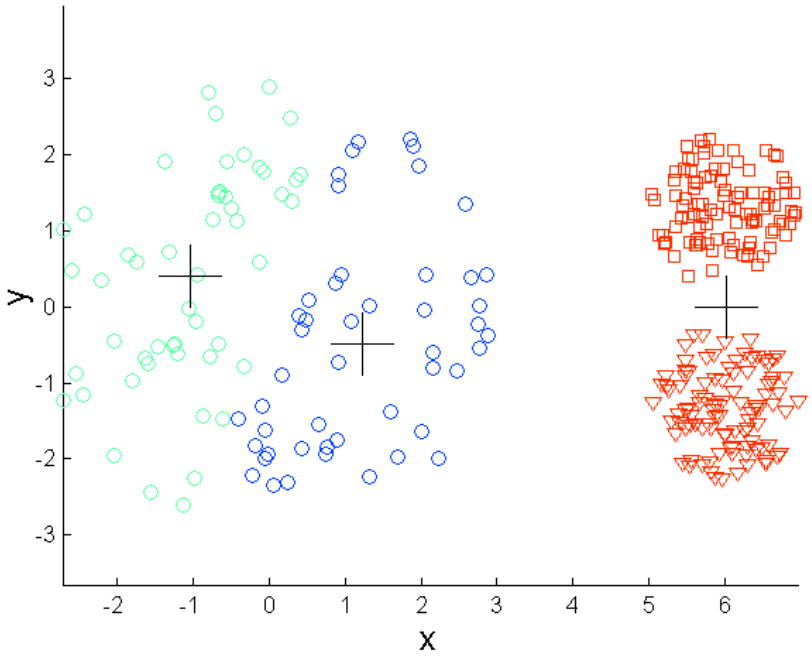


K-means (3 Clusters)

Limitations of K-means: Differing Density

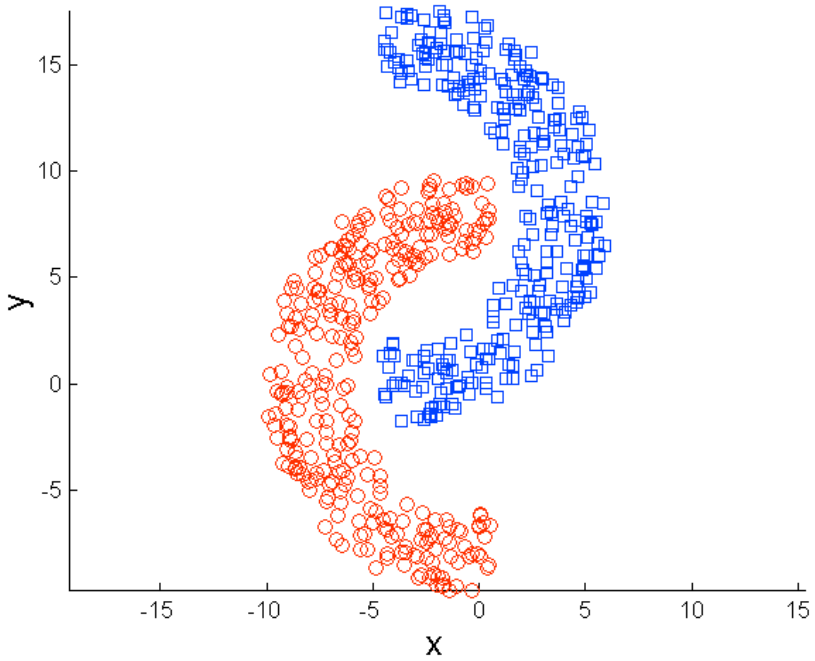


Original Points

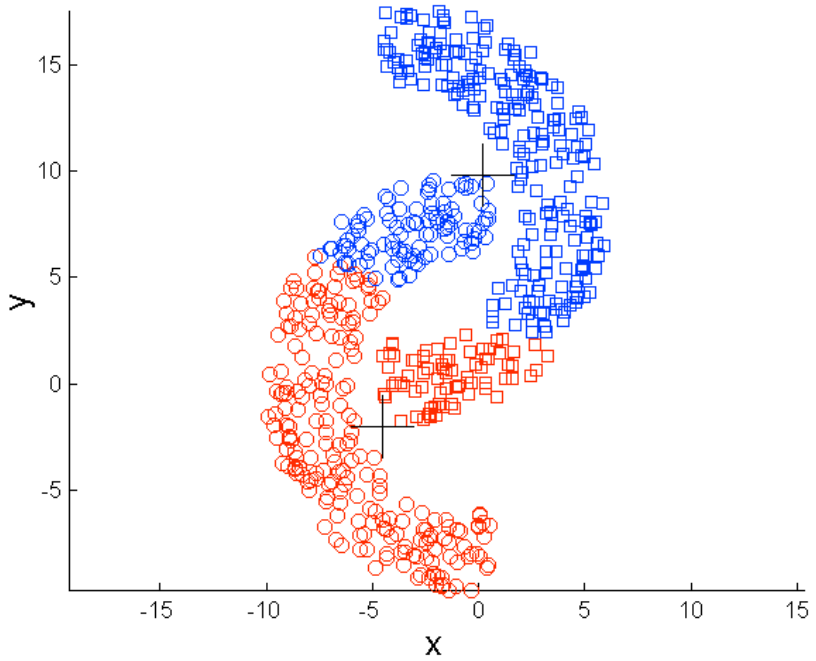


K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes

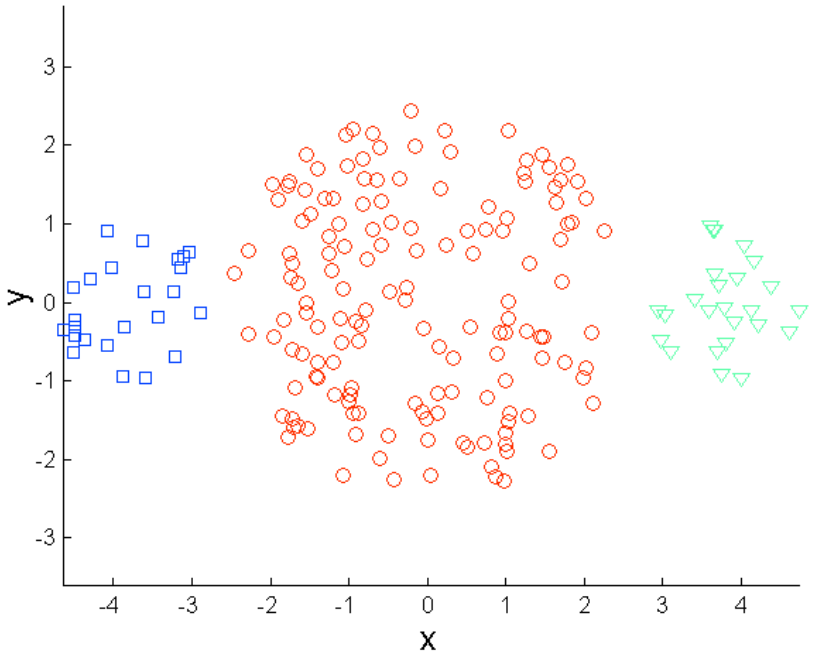


Original Points

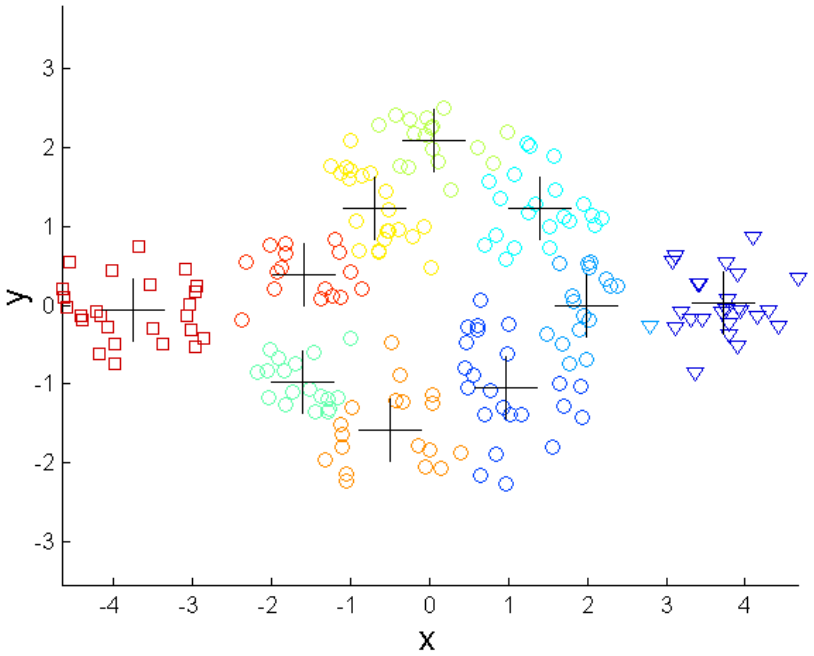


K-means (2 Clusters)

Overcoming K-means Limitations



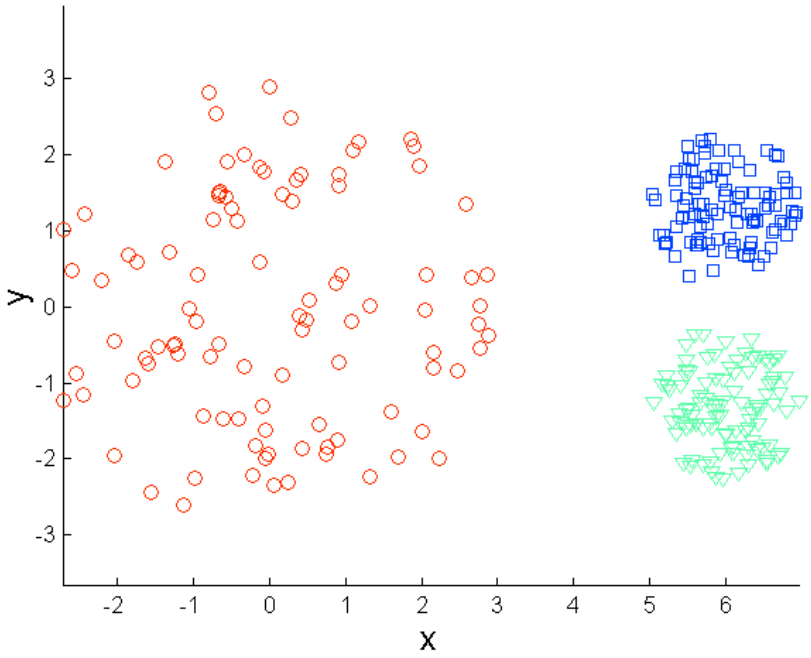
Original Points



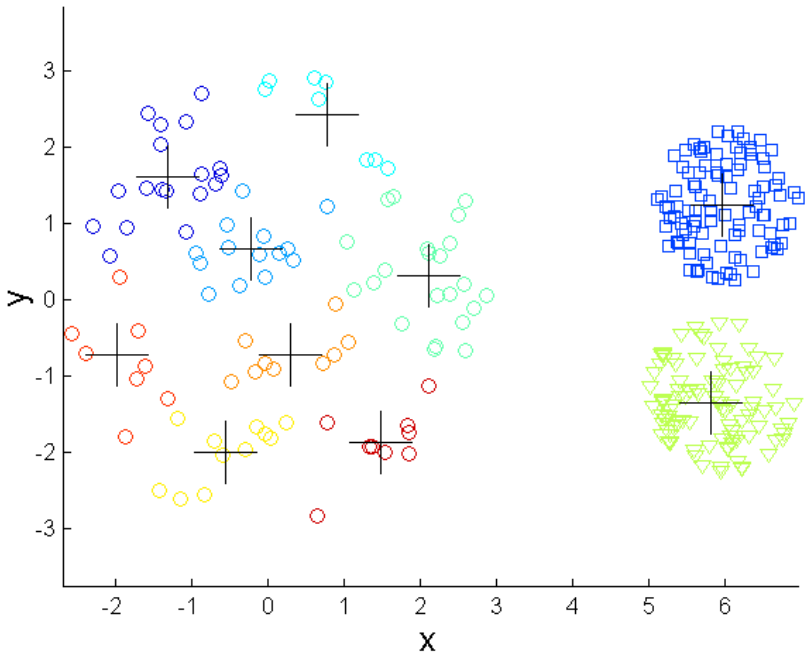
K-means Clusters

One solution is to use many clusters.
Find parts of clusters, but need to put together.

Overcoming K-means Limitations

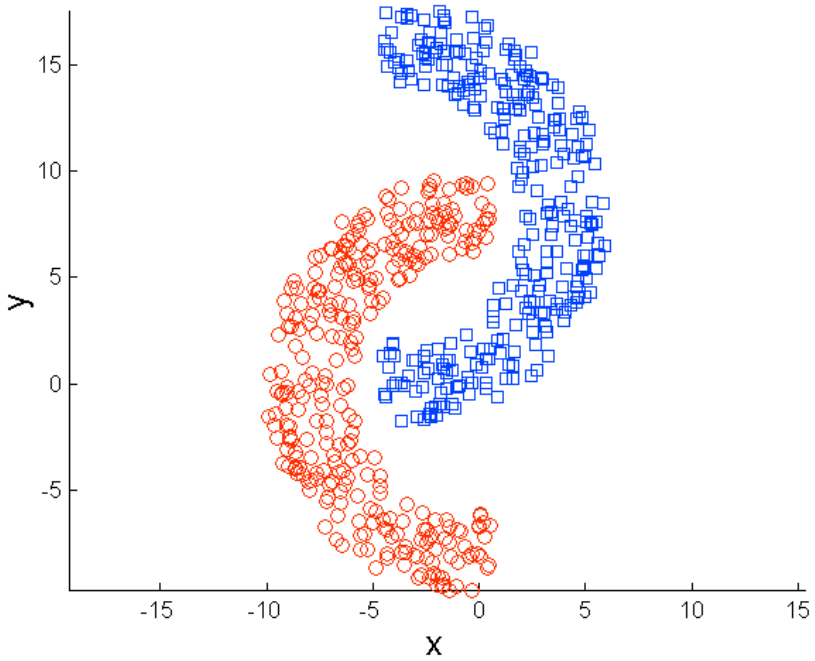


Original Points

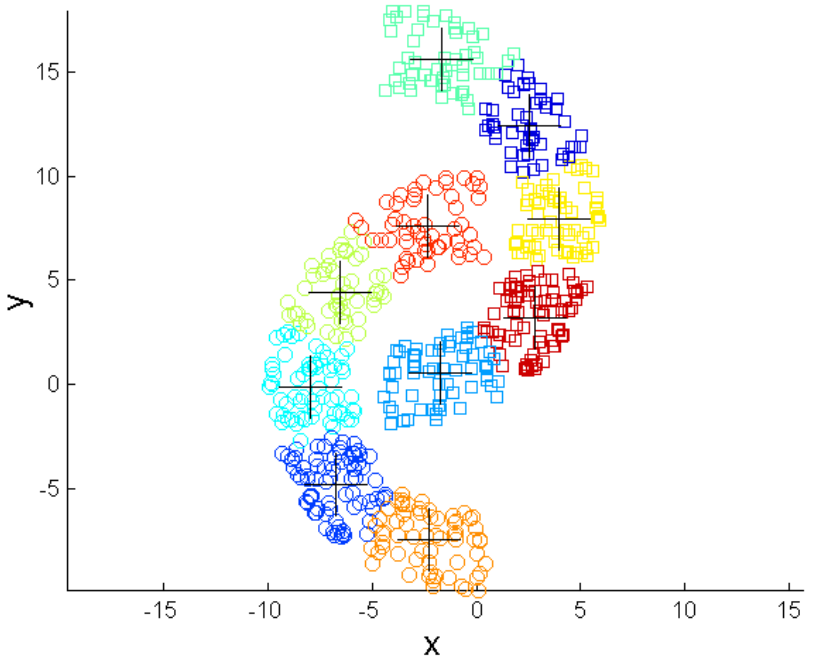


K-means Clusters

Overcoming K-means Limitations



Original Points



K-means Clusters

K-Means Clustering Summary

- Strength
  - Relatively efficient
  - Often terminates at a local optimum
  - The global optimum may be found using techniques such as: deterministic annealing and genetic algorithms
- Weakness
  - Applicable only when mean is defined, then what about categorical data?
  - Need to specify k, the number of clusters, in advance
  - Unable to handle noisy data and outliers
  - Not suitable to discover clusters with non-convex shapes

K-Means Clustering Summary

- Advantages
  - Simple, understandable
  - Items automatically assigned to clusters
- Disadvantages
  - Must pick number of clusters before hand
  - All items forced into a cluster
  - Too sensitive to outliers

Variations of the K-Means Method

- A few variants of the k-means which differ in
  - Selection of the initial k means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: k-modes
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - Using a frequency-based method to update modes of clusters
  - A mixture of categorical and numerical data: k-prototype method

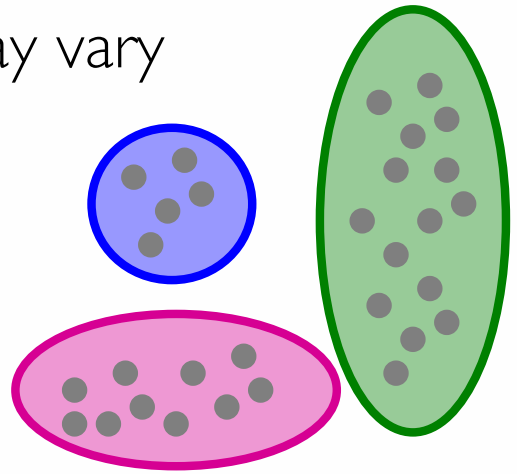
Variations of the K-Means Method

- A few variants of the k-means which differ in
  - Selection of the initial k means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: k-modes
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - Using a frequency-based method to update modes of clusters
  - A mixture of categorical and numerical data: k-prototype method

The BFR Algorithm

The BFR Algorithm

- BFR [Bradley-Fayyad-Reina] is a variant of k-means designed to handle very large (disk-resident) data sets
- Assumes that clusters are normally distributed around a centroid in a Euclidean space
- Standard deviations in different dimensions may vary
- Clusters are axis-aligned ellipses
- Efficient way to summarize clusters (want memory required O(clusters) and not O(data))



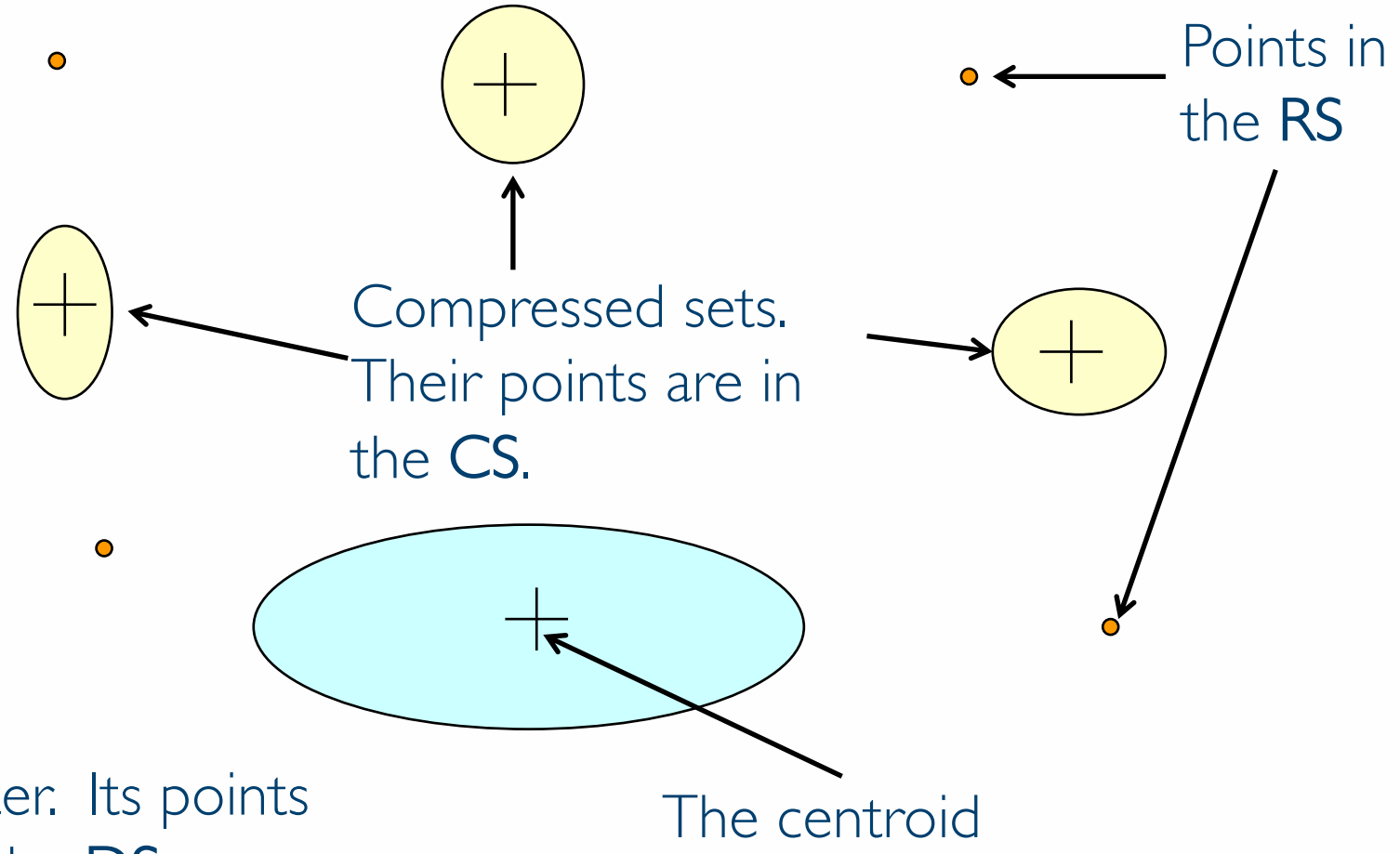
The BFR Algorithm

- Points are read from disk one chunk at the time (so to fit into main memory)
- Most points from previous memory loads are summarized by simple statistics
- To begin, from the initial load we select the initial k centroids by some sensible approach
  - Take k random points
  - Take a small random sample and cluster optimally
  - Take a sample; pick a random point, and then k–1 more points, each as far from the previously selected points as possible

Three Classes of Points

- Discard set (DS)
  - Points close enough to a centroid to be summarized
- Compression set (CS)
  - Groups of points that are close together but not close to any existing centroid
  - These points are summarized, but not assigned to a cluster
- Retained set (RS)
  - Isolated points waiting to be assigned to a compression set

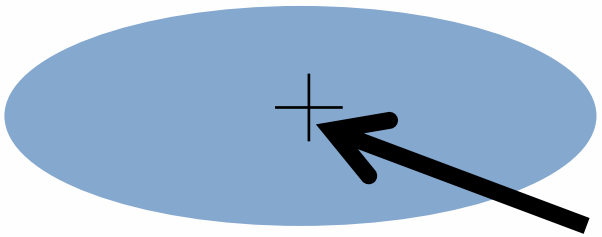
The Status of BFR Algorithm



Discard set (DS): Close enough to a centroid to be summarized
Compression set (CS): Summarized, but not assigned to a cluster
Retained set (RS): Isolated points

Summarizing Sets of Points

- For each cluster, the discard set (DS) is summarized by:
- The number of points, N
- The vector SUM, whose component SUM(i) is the sum of the coordinates of the points in the ith dimension
- The vector SUMSQ whose component SUMSQ(i) is the sum of squares of coordinates in ith dimension



A cluster.
All its points are in the DS.

The centroid

Summarizing Points: Comments

- 2d + 1 values represent any size cluster
  (d is the number of dimensions)
- Average in each dimension (the centroid) can be calculated as SUM(i)/N
- Variance of a cluster's discard set in dimension i is computed as (SUMSQ(i)/N) – $(SUM(i)/N)^2$
- And standard deviation is the square root of that variance

Processing Data in the BFR Algorithm

1. First, all points that are "sufficiently close" to the centroid of a cluster are added to that cluster (by updating its parameters) then the point is discharged
2. The points that are not "sufficiently close" to any centroid are clustered along with the points in the retained set. Any algorithm can be used even the hierarchical one in this step.
3. The miniclusters derived for new points and the old retained set are merged (e.g., by using the same criteria used for hierarchical clustering)
4. Any point outside a cluster or a minicluster are dropped.

When the last chunk of data is processed, the remaining miniclusters and the points in the retained set which might be labeled as outliers or alternatively can be assigned to one of the centroids (as k-means would do).

Note that for miniclusters we only have N, SUM and SUMSQ so it is easier to used criteria based on variance and similar statistics. So we might combine two clusters if their combined variance is below some threshold.

"Sufficiently Close"

- Two approaches have been proposed to determine whether a point is sufficiently close to a cluster
- Add p to a cluster if
  - It has the centroid closest to p
  - It is also very unlikely that, after all the points have been processed, some other cluster centroid will be found to be nearer to p
- We can measure the probability that, if p belongs to a cluster, it would be found as far as it is from the centroid of that cluster
  - This is where the assumption about the clusters containing normally distributed points aligned with the axes of the space is used

Mahalanobis Distance

- It is used to decide whether a point is closed enough to a cluster
- It is computed as the distance between a point and the centroid of a cluster, normalized by the standard deviation of the cluster in each dimension.
- Given p = (p_1, … p_d) and c = (c_1, … c_d), the Mahalanobis distance between p and c is computed as

$$\sqrt{\sum_{i=1}^{d}\left(\frac{p_i - c_i}{\sigma_i}\right)^2}$$

- We assign p to the cluster with the least Mahalanobis from p provided that the distance is below a certain threshold. A threshold of 4 means that we have only a chance in a million not to include something that belongs to the cluster

Expectation Maximization

Expectation-Maximization (EM) Clustering

- k-means assigns each point to only one cluster (hard assignment)
- The approach can be extended to consider soft assignment of points to clusters, so that each point has a probability of belonging to each cluster
- We assume that each cluster C_i is characterized by a multivariate normal distribution and thus identified by
  - The mean vector $\boldsymbol{\mu}_i$
  - The covariance matrix $\boldsymbol{\Sigma}_i$
- A clustering is identified by a vector of parameter $\boldsymbol{\theta}$ defined as

$$\boldsymbol{\theta} = \{\boldsymbol{\mu}_i\ \boldsymbol{\Sigma}_i\ P(C_i)\}$$

where $P(C_i)$ are the prior probability of all the clusters C_i which sum up to one

Expectation-Maximization (EM) Clustering

- The goal of maximum likelihood estimation (MLE) is to choose the parameters θ that maximize the likelihood, that is

$$\theta^\star = \arg\max_\theta P(D|\theta)$$

- General idea
  - Starts with an initial estimate of the parameter vector
  - Iteratively rescores the patterns against the mixture density produced by the parameter vector
  - The rescored patterns are used to update the parameter updates
  - Patterns belonging to the same cluster, if they are placed by their scores in a particular component

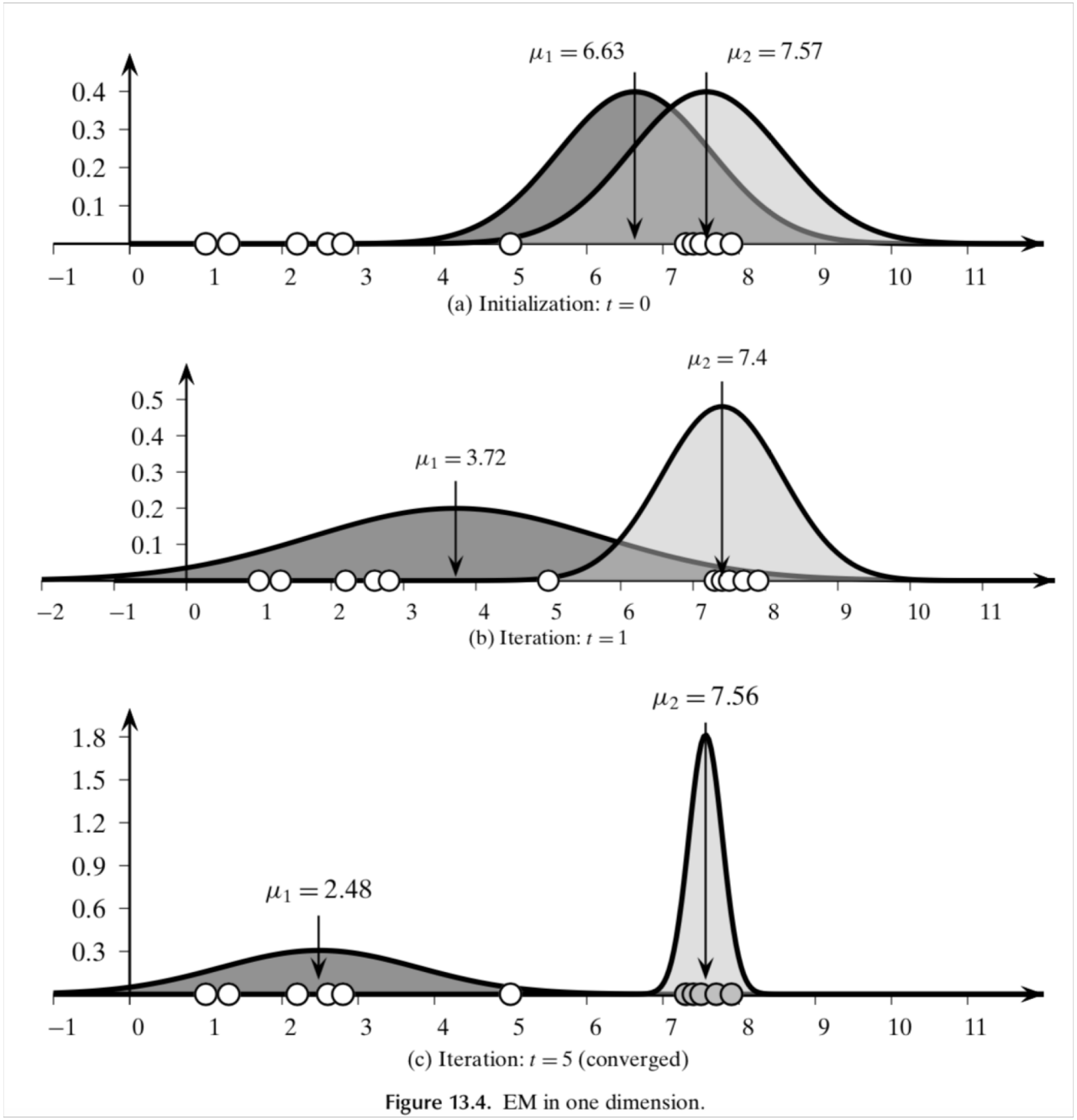


Figure 13.4. EM in one dimension.

Example from Chapter 13 of the textbook

Run the python notebooks and the KNIME workflows for this lecture