Data Mining Tasks

- 1st important task: classification
- 2nd important task: clustering
- 3rd important task: feature selection

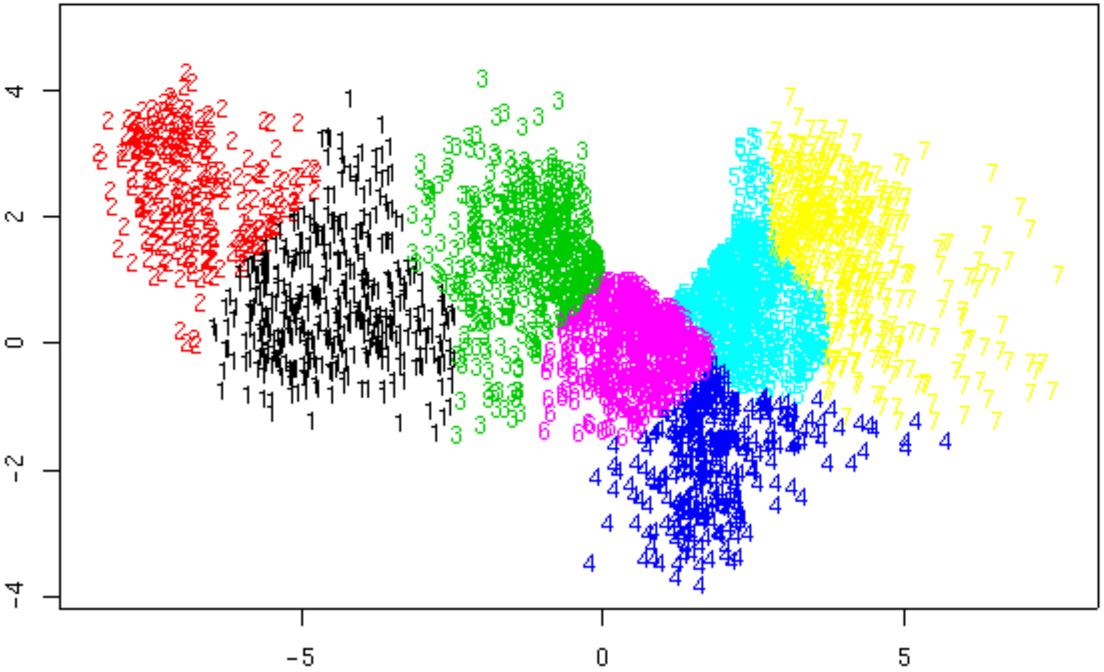
Once you know the basics of data mining, most people will say:

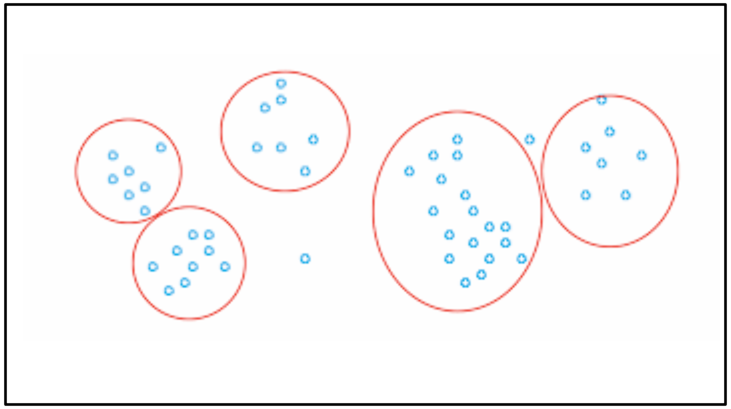
The most important thing is feature generation: this include

- dimension reduction
- data projection (linear and non-linear)
- graph embedding

Data Clustering

Data clustering : divide data into groups





Data Clustering Knowledge Discovery

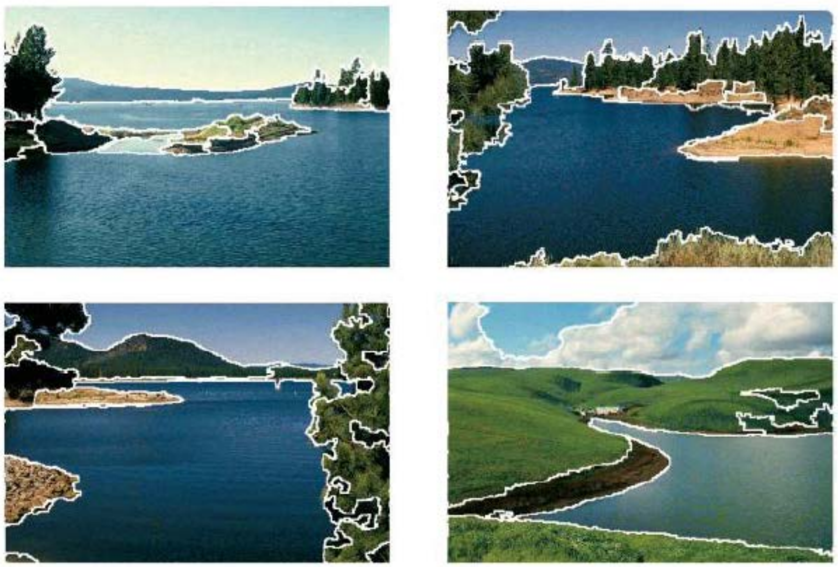
Credit card usage/record, use clustering to discover different purchase behaviors and fraud

- Buy airplane ticket, hotel, restaurants, etc → customer is traveling
- Buy restaurant, movie, withdraw $200 → normal evening out
- Withdraw a lot of money, buy several very expressive items with short time → fraud

Social Networks

clustering to discover different communities

Image Segmentation



Many many other applications

Data Clustering

- Also called cluster analysis
- It is unsupervised learning (no class labels are given)
- The common process of discovering new patterns (behaviors,knowledge)

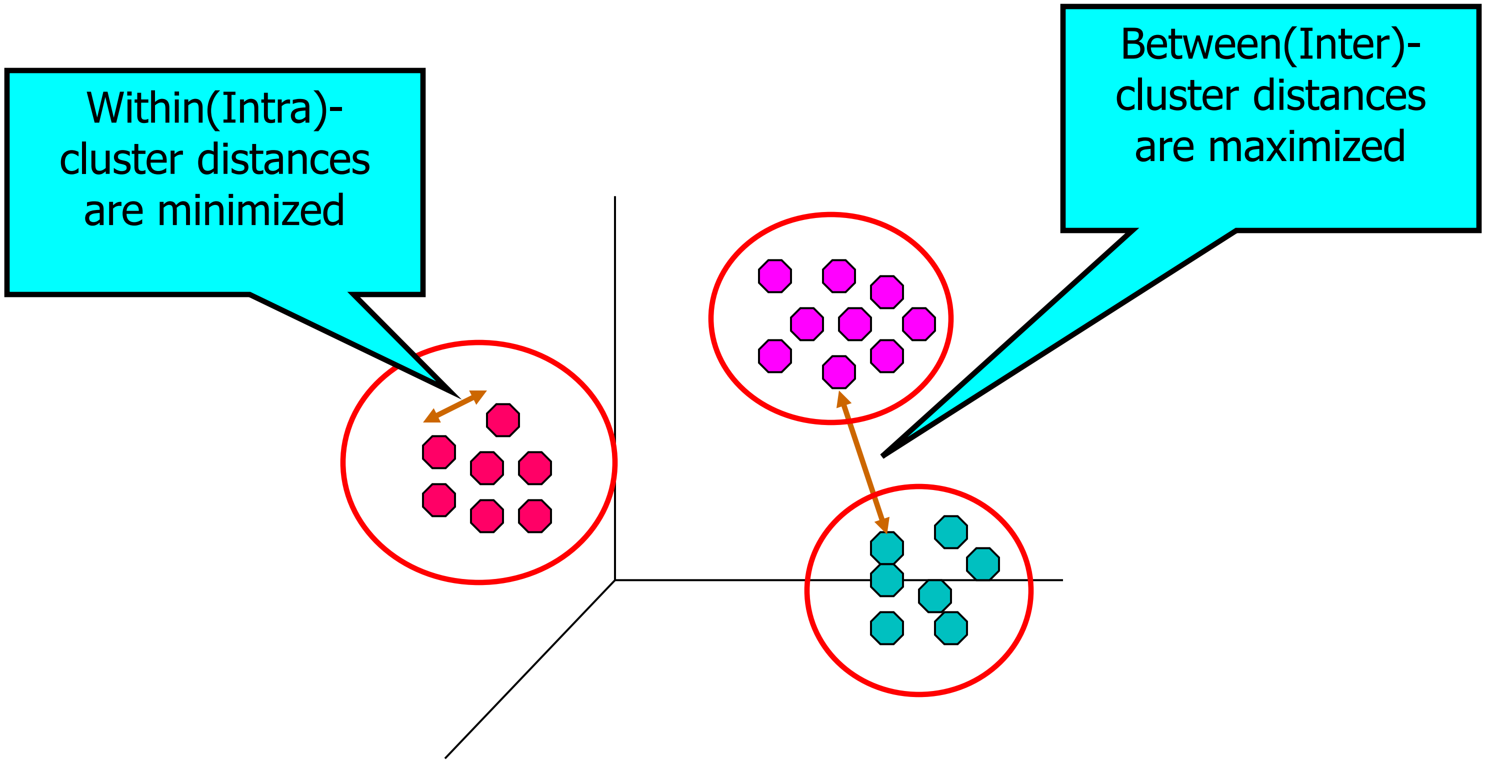
A clustering algorithm groups objects into groups, such that

- Objects within the same group are similar (coherent, related, like each other)
- Objects between different groups are different (distinct, un-related)

If you divide 10 people into 2 groups, it is desirable that
people within a group know each other, or better, they are good friends.
This increase the cohesion of each group.

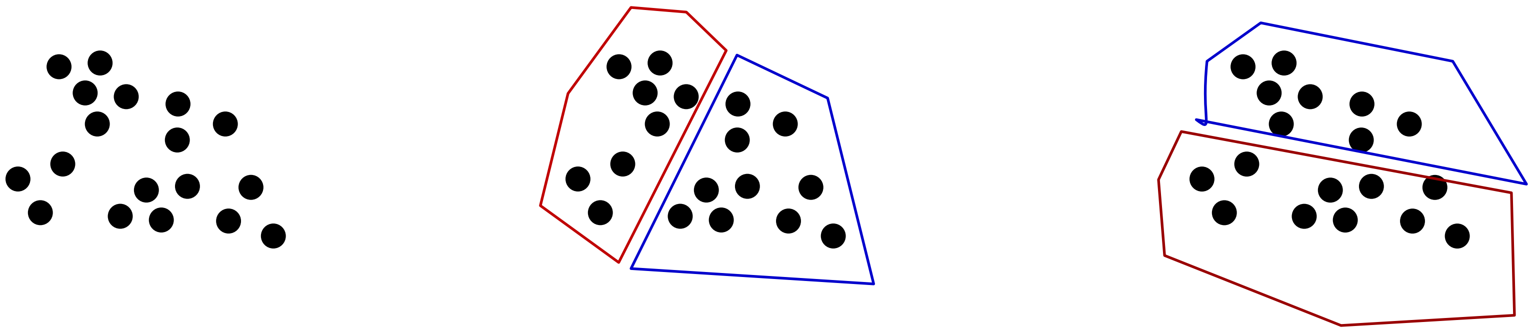
What is Cluster Analysis?

- 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



Data Clustering

Input data One clustering Another clustering



This clustering is better :
Data within each class are more close to each other
Data between different classes are more distant

A clustering algorithm finds different ways to group data, and pick the optimal clustering solution according to a fixed objective function

K-means Clustering

- The most popular clustering algorithm/approach
- First invented in 1965 （S. Lloyd)
- After 53 years, K-means clustering is still the best
- Many newer clustering methods are based on K-means

K-means Clustering

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

Select K points as the initial centroids.
repeat
    Form K clusters by assigning all points to the closest centroid. ← **assignment step**
    Recompute the centroid of each cluster. ← **Recompute centroid step**
until The centroids don't change

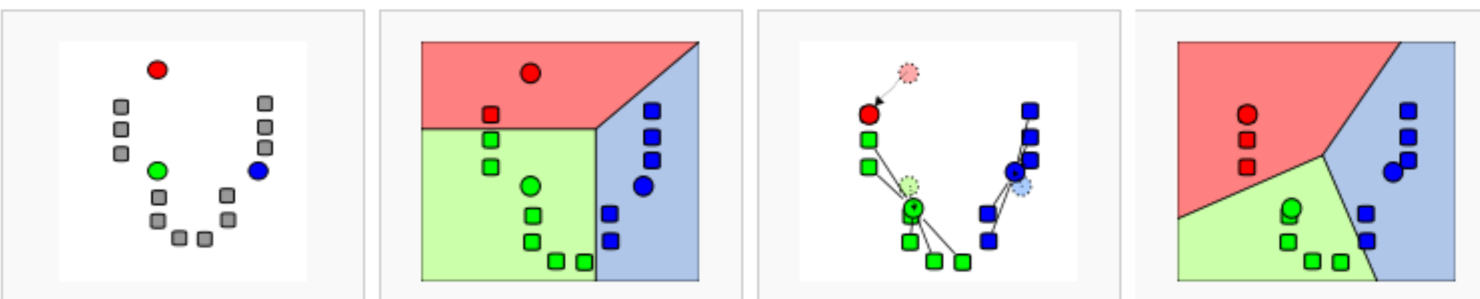
Description [edit]

Given a set of observations ($\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$), where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k ($\leq n$) sets $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares (WCSS) (sum of distance functions of each point in the cluster to the K center). In other words, its objective is to find:

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|^2$$

where $\boldsymbol{\mu}_i$ is the mean of points in S_i.

Demonstration of the standard algorithm



1. k initial "means" (in this case k=3) are randomly generated within the data domain (shown in color).

2. k clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

3. The centroid of each of the k clusters becomes the new mean.

4. Steps 2 and 3 are repeated until convergence has been reached.

K-Means Clustering (contd.)

- Example

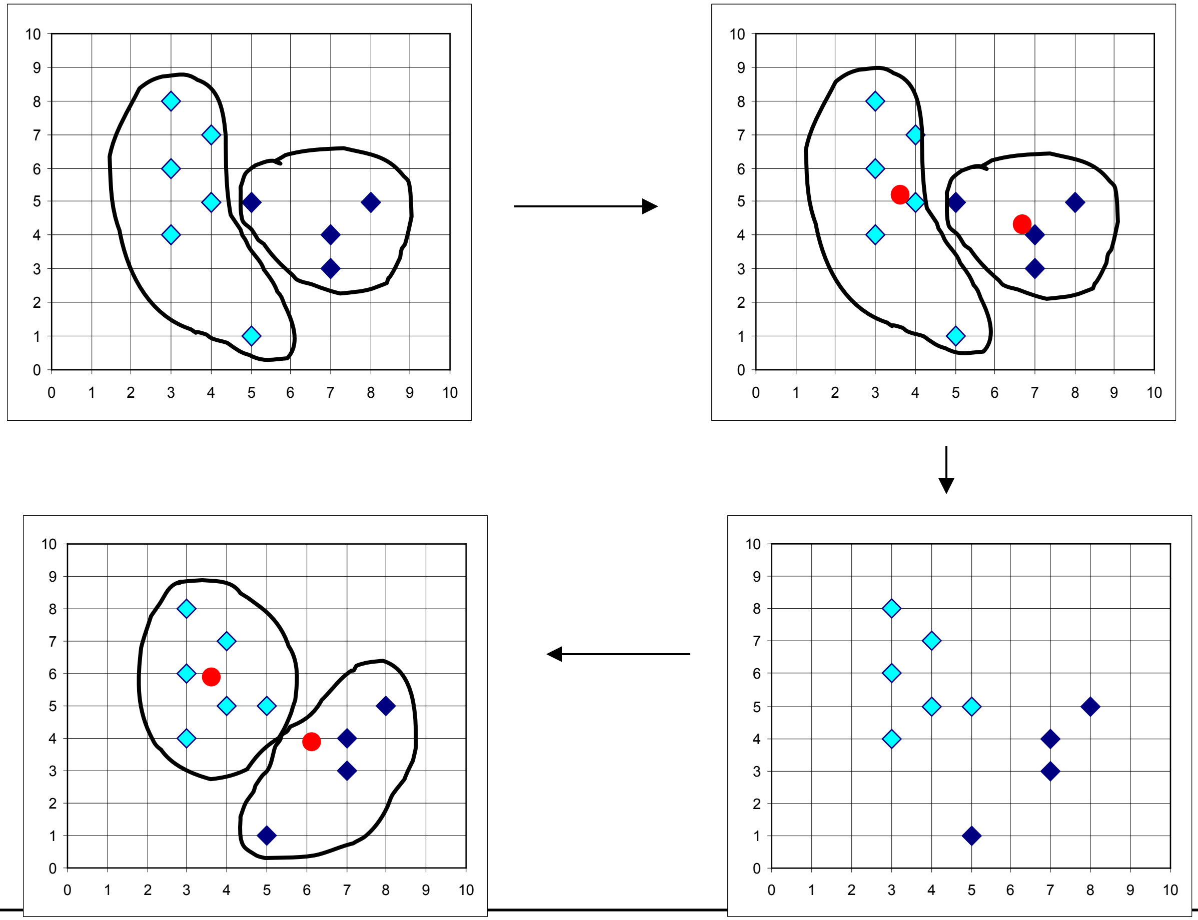
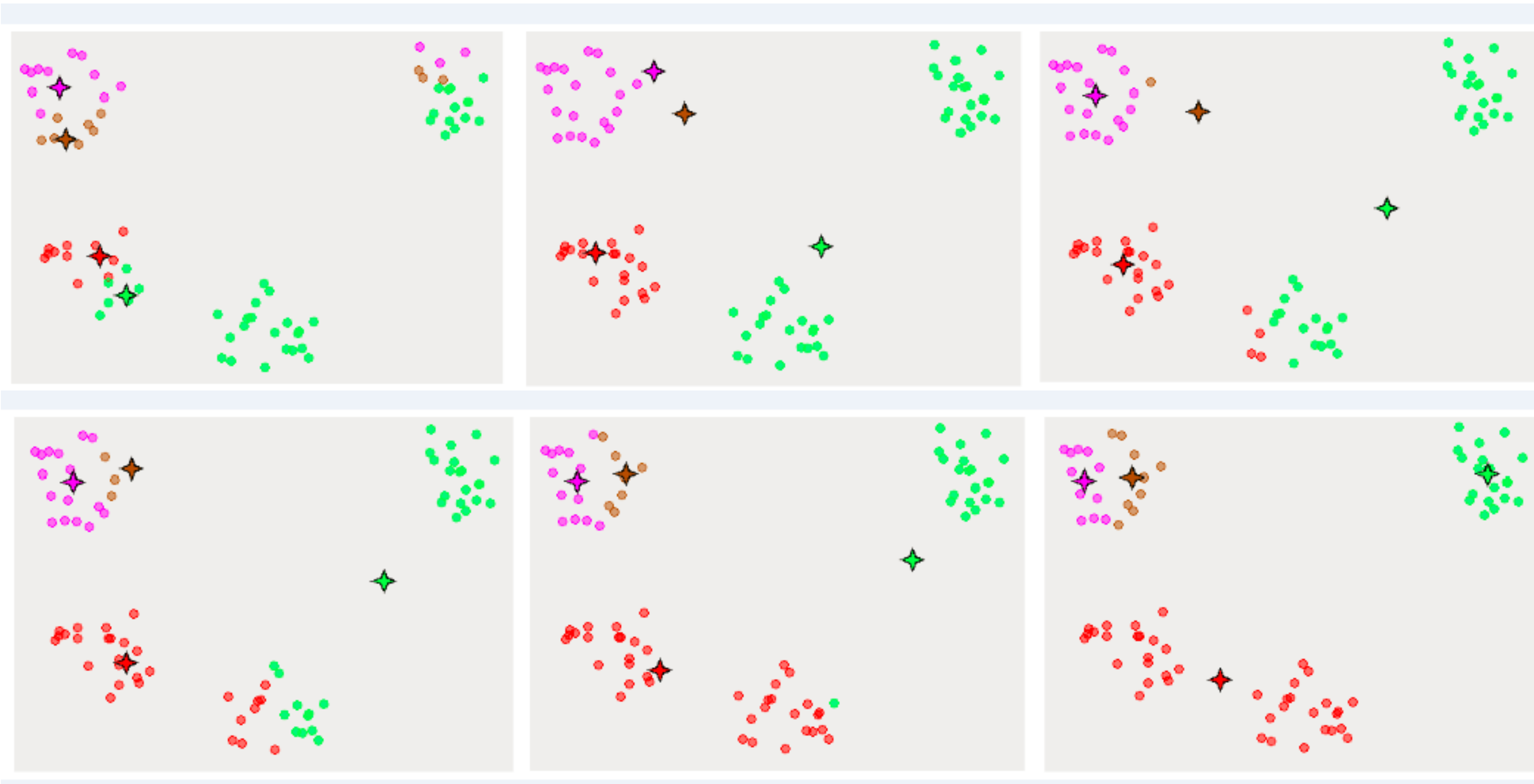
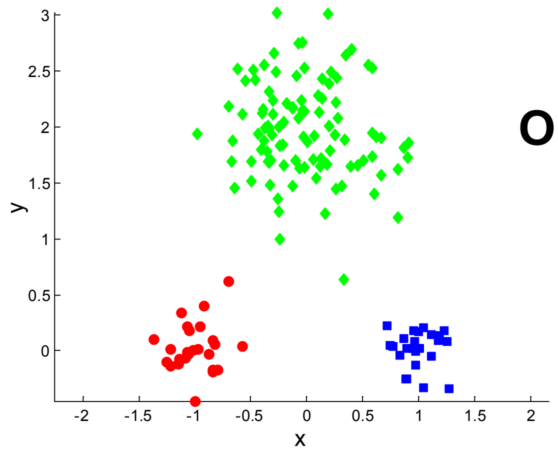


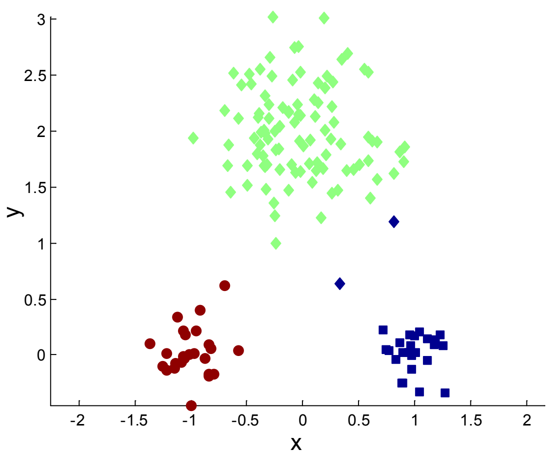
Illustration of K-means Algorithm



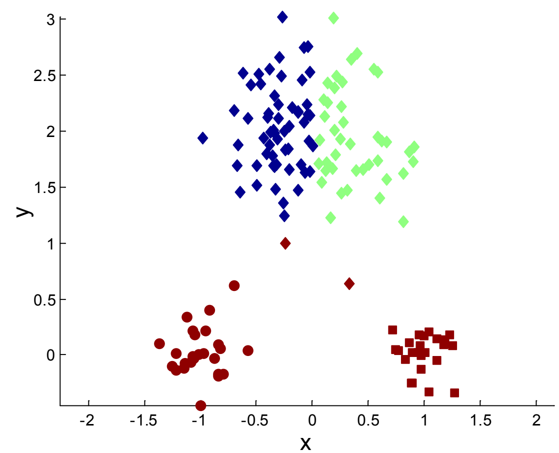
Two different K-means Clusterings



Original Points

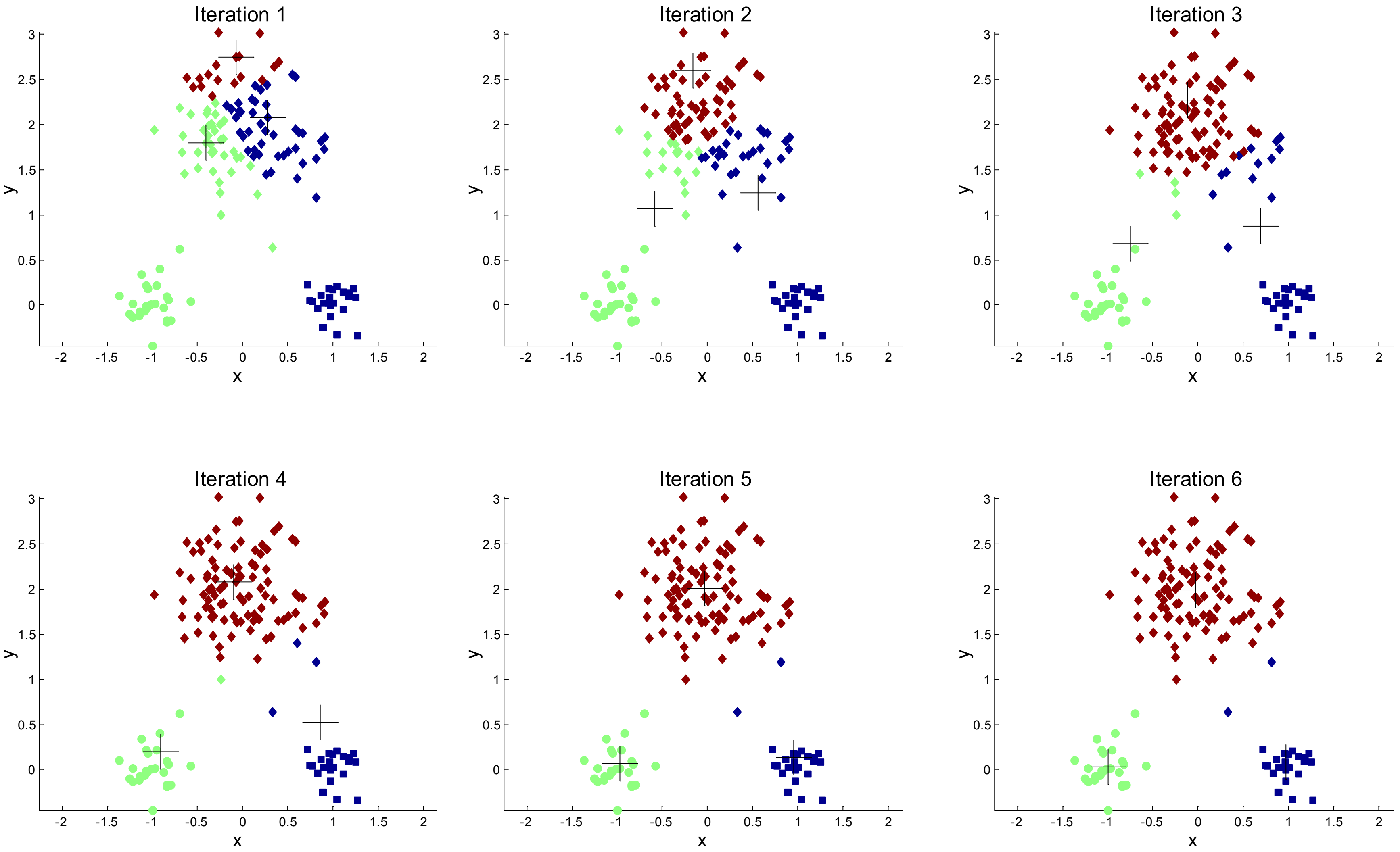


Optimal Clustering

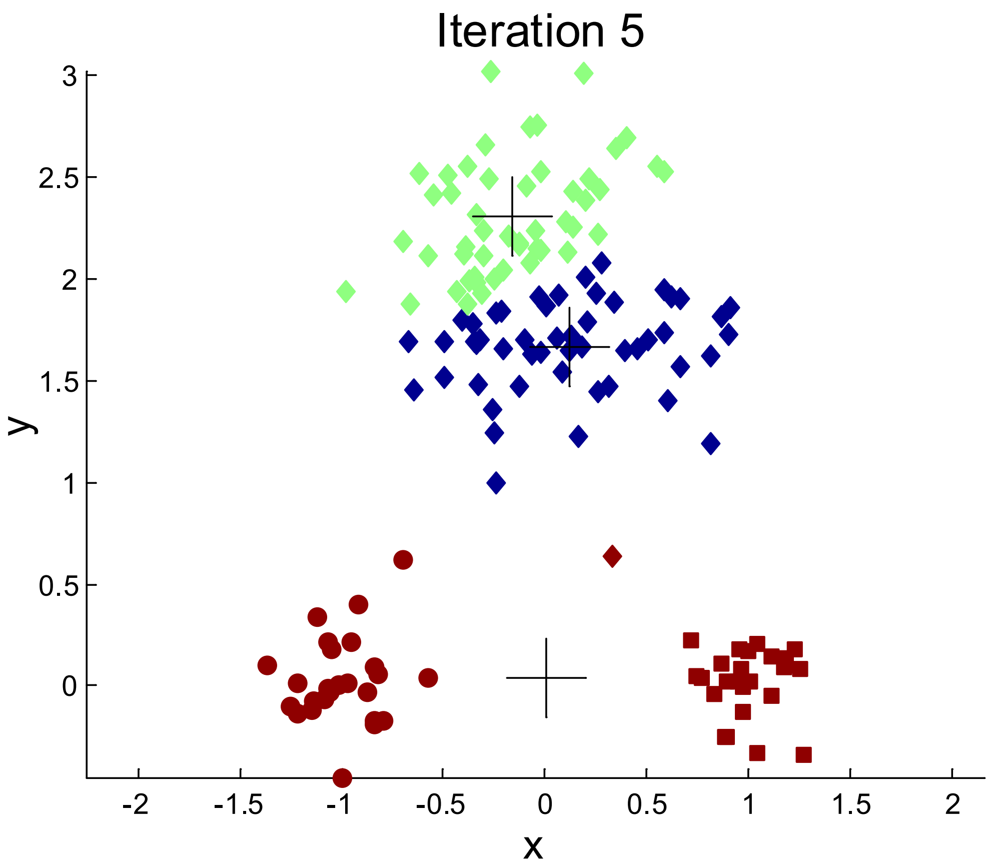


Sub-optimal Clustering

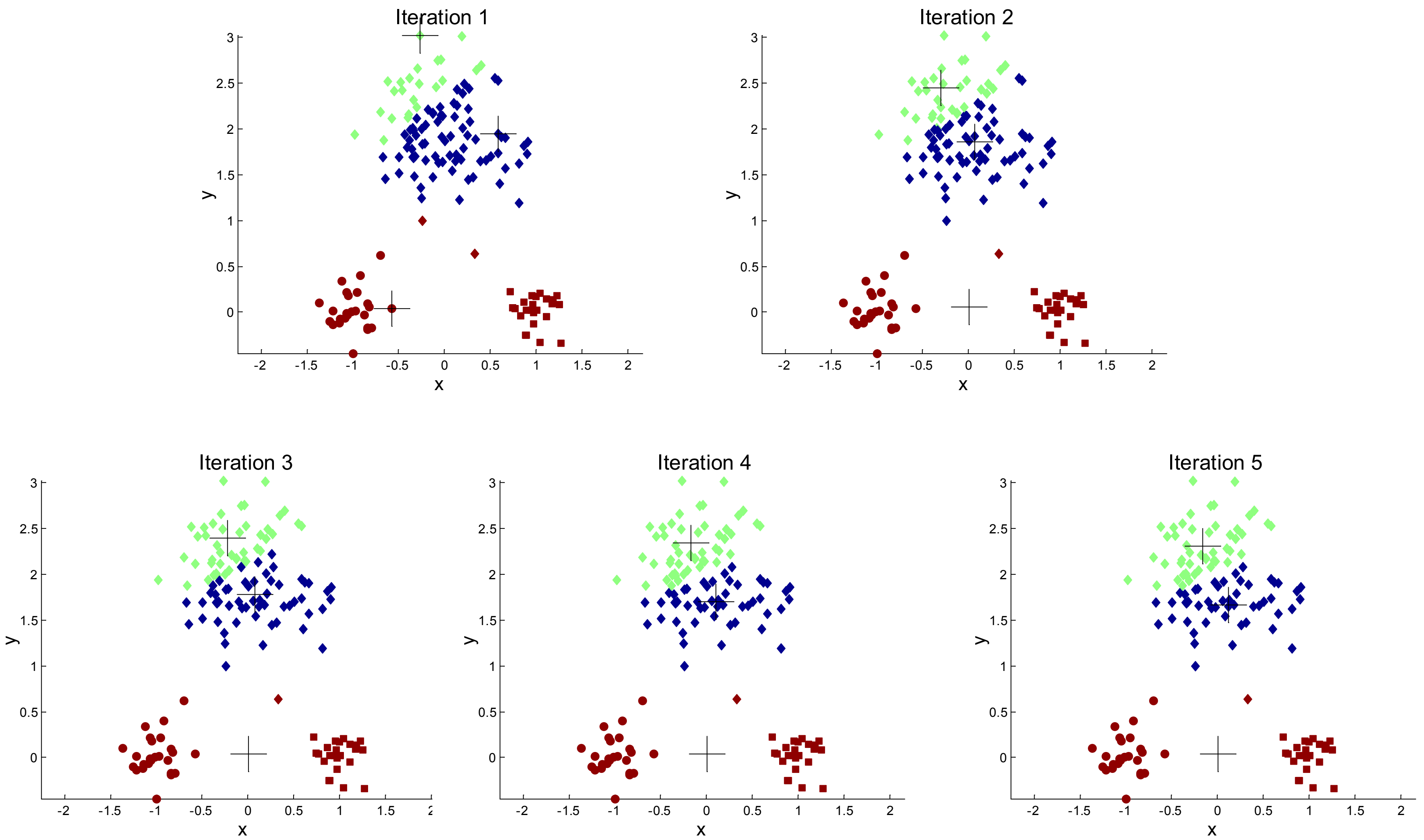
Importance of Choosing Initial Centroids: good choice



A bad choice of initial centroids



Importance of Choosing Initial Centroids: bad choice



K-means Clustering – Details

- Initial centroids are often chosen randomly.
  - Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- 'Closeness' is measured by Euclidean distance, cosine similarity, correlation, etc.
- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to 'Until relatively few points change clusters'
- Complexity is O(n * K * I * d)
  - n = number of points, K = number of clusters,
I = number of iterations, d = number of attributes

Evaluating K-means Clusters

- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest cluster
  - To get SSE, we square these errors and sum them.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

  - x is a data point in cluster C_i and m_i is the representative point for cluster C_i
    - can show that m_i corresponds to the center (mean) of the cluster
  - Given two clusters, we can choose the one with the smallest error
  - One easy way to reduce SSE is to increase K, the number of clusters
    - A good clustering with smaller K can have a lower SSE than a poor clustering with higher K

Distance measure

- **Euclidean distance**

$$d(g_1, g_2) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- **Manhattan distance**

$$d(g_1, g_2) = \sum_{i=1}^{n} \left|(x_i - y_i)\right|$$

- **Minkowski distance**

$$d(g_1, g_2) = \sqrt[m]{\sum_{i=1}^{n} (x_i - y_i)^m}$$

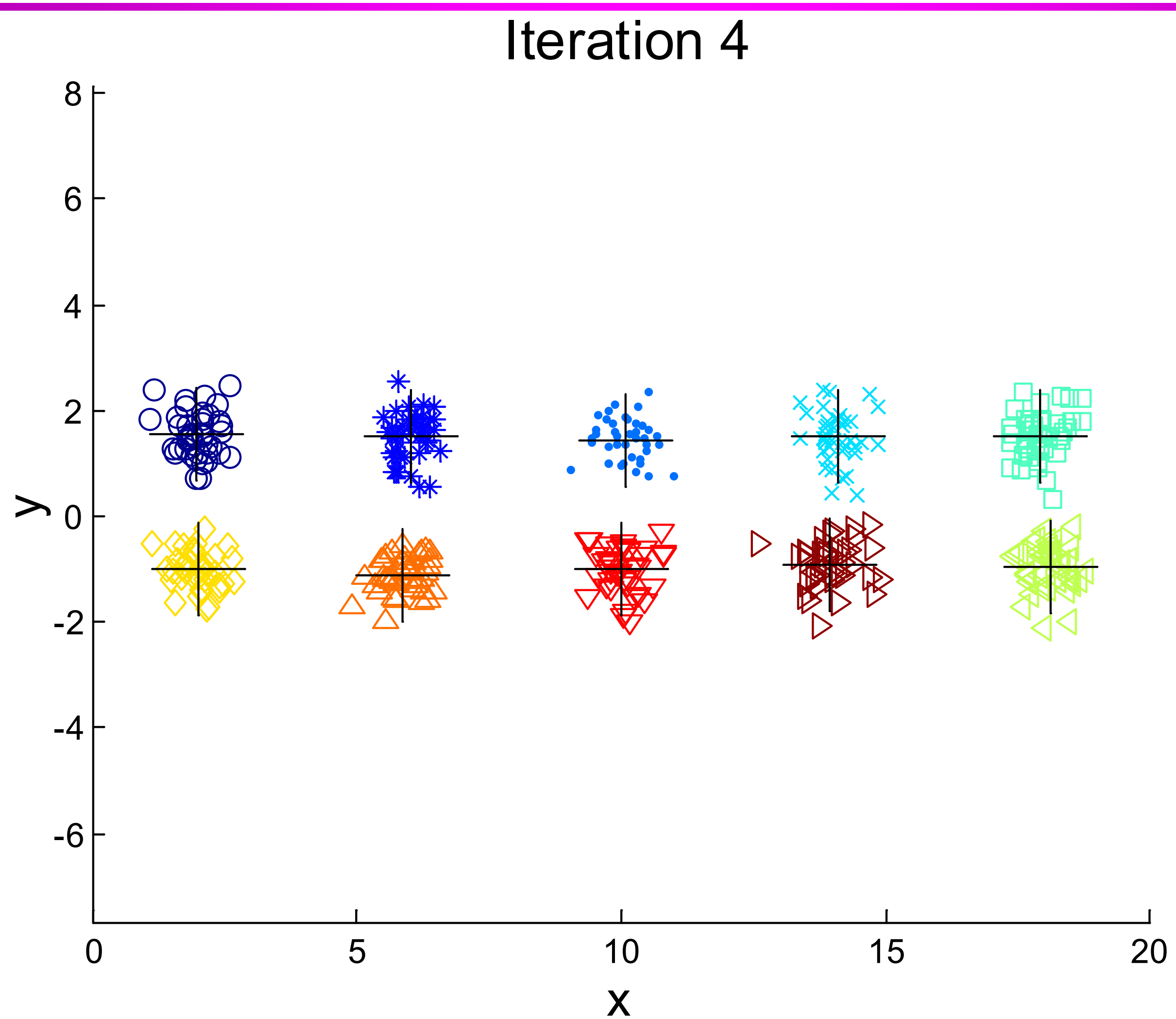
Problems with Selecting Initial Points

- 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

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

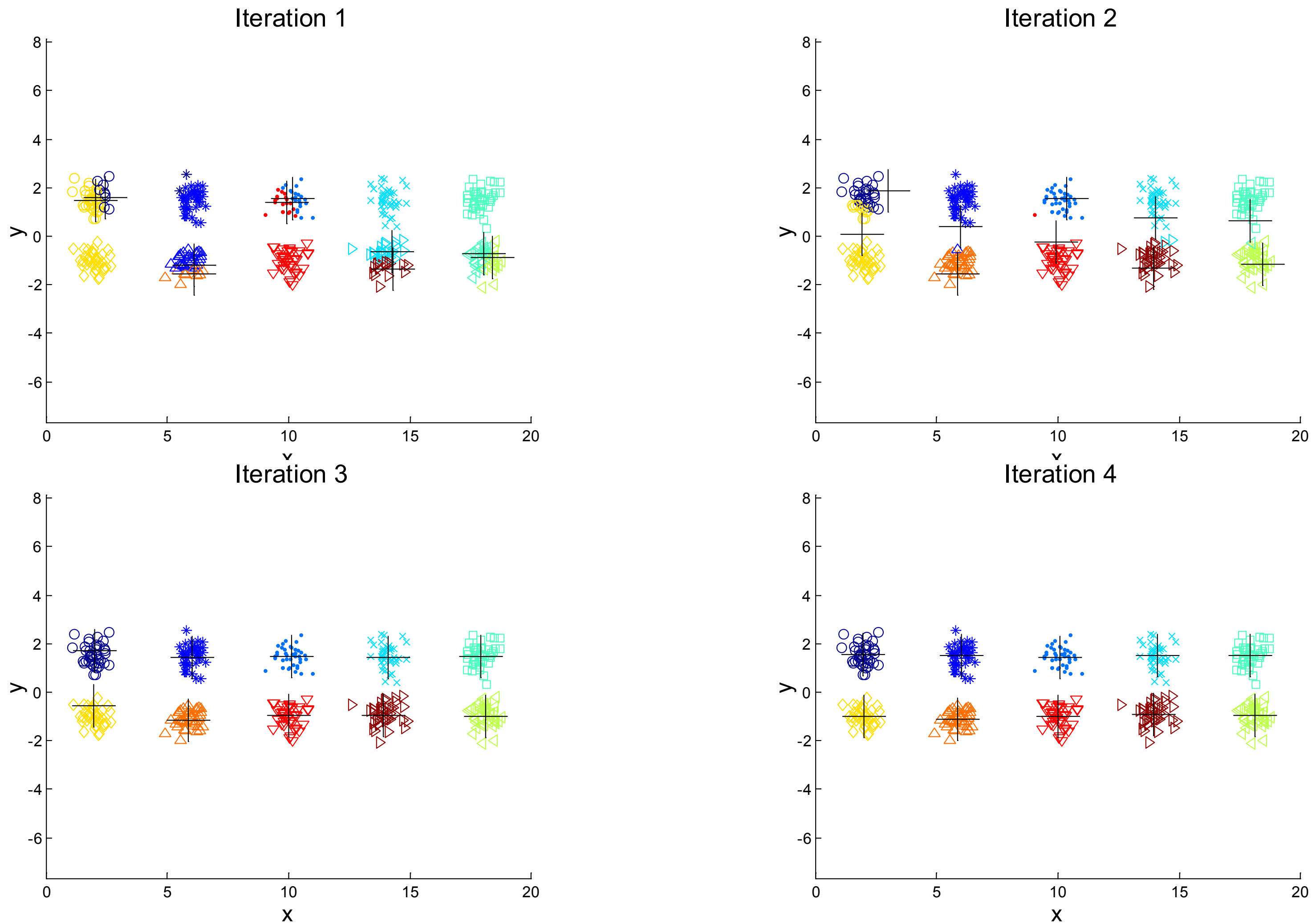
  - For example, if K = 10, then probability = $10!/10^{10}$ = 0.00036
  - Sometimes the initial centroids will readjust themselves in 'right' way, and sometimes they don't
  - Consider an example of five pairs of clusters

10 Clusters Example



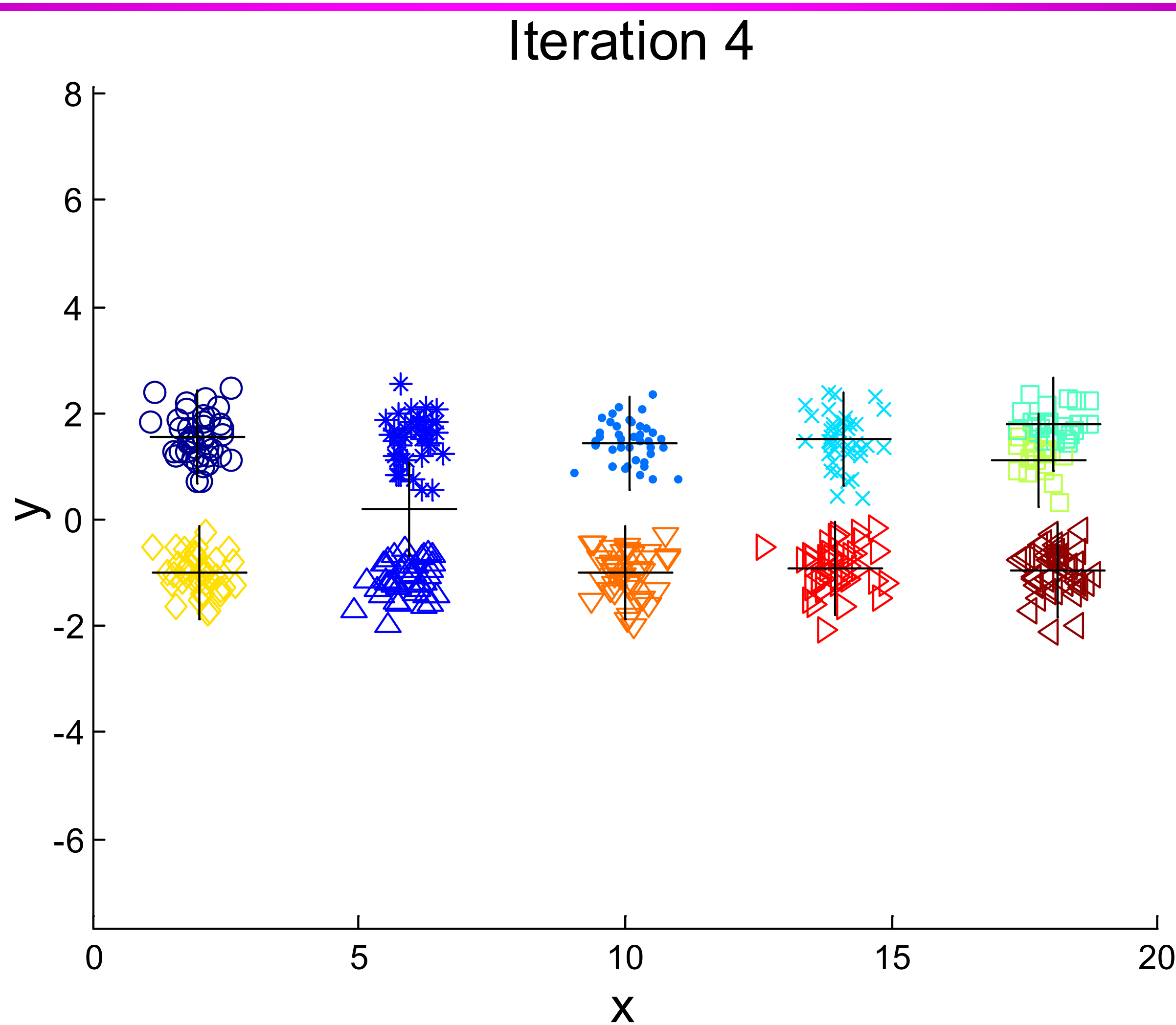
Starting with two initial centroids in one cluster of each pair of clusters

10 Clusters Example



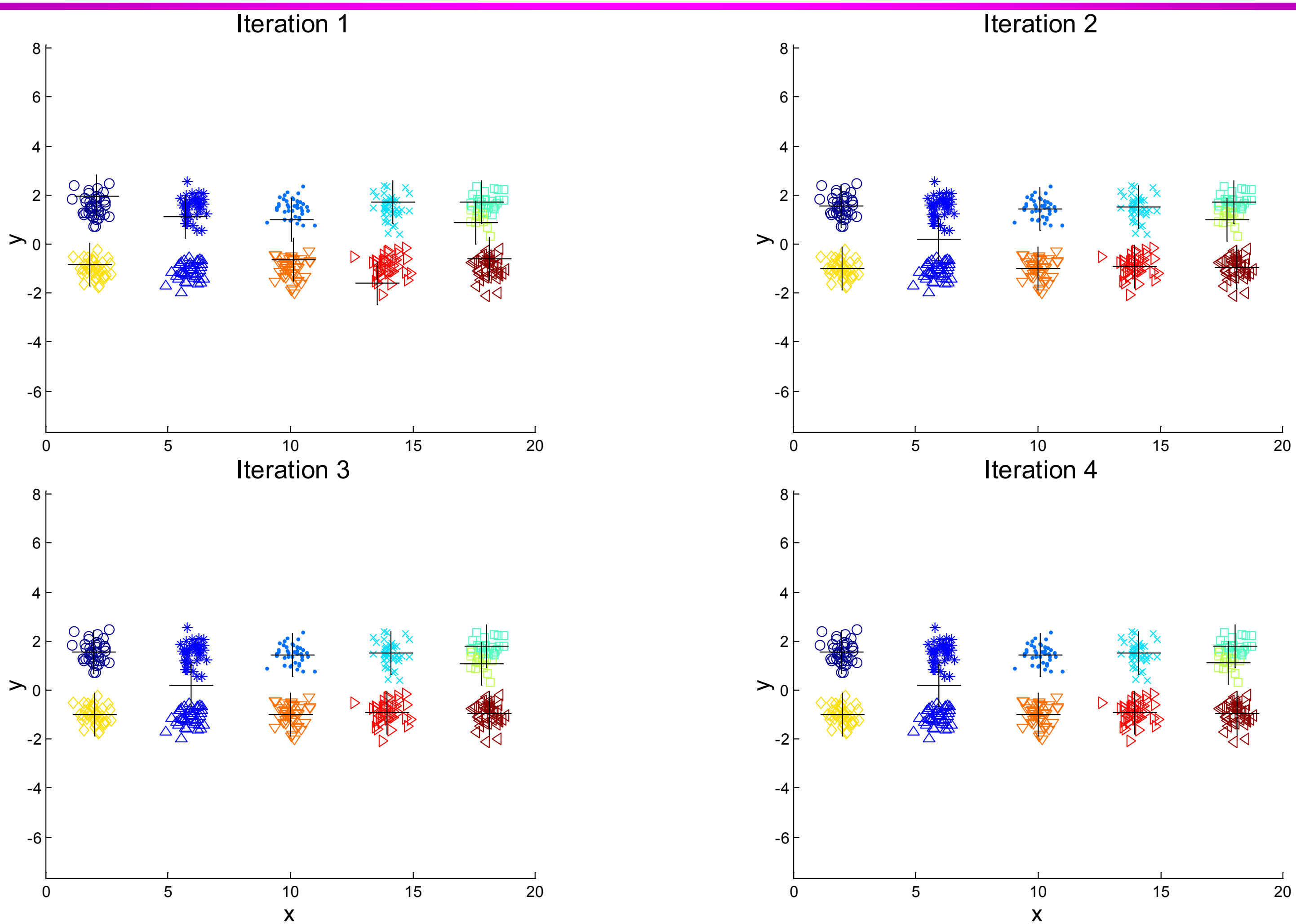
Starting with two initial centroids in one cluster of each pair of clusters

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.

Solutions to Initial Centroids Problem

- Multiple runs
  - Helps, but probability is not on your side
- Sample and use hierarchical clustering to determine initial centroids
- Select more than k initial centroids and then select among these initial centroids
  - Select most widely separated
- Postprocessing
- Bisecting K-means
  - Not as susceptible to initialization issues

Handling Empty Clusters

- Basic K-means algorithm can yield empty clusters

- Several strategies
  - Choose the point that contributes most to SSE
  - Choose a point from the cluster with the highest SSE
  - If there are several empty clusters, the above can be repeated several times.

Updating Centers Incrementally

- Batch algorithm (standard algorithm)
  - Centroids are updated after all data points are assigned to their corresponding centroids

- Online/incremental algorithm
  - Update the centroids after each assignment (one cluster gets one more data point, another cluster loose one data point)
  - More expensive
  - Introduces an order dependency
  - Never get an empty cluster

Pre-processing and Post-processing

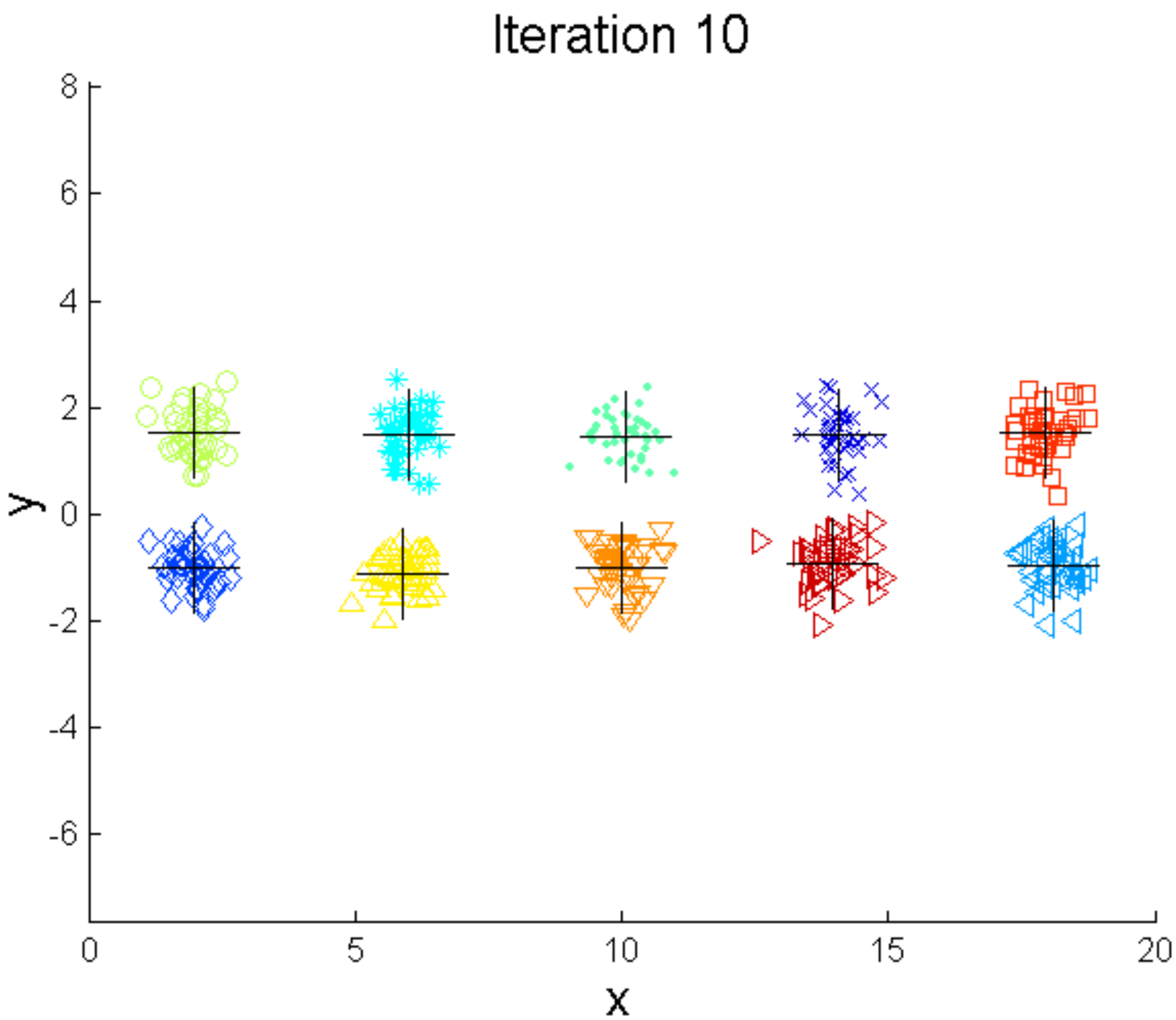
- Pre-processing
  - Normalize the data
  - Eliminate outliers
- Post-processing
  - Eliminate small clusters that may represent outliers
  - Split 'loose' clusters, i.e., clusters with relatively high SSE
  - Merge clusters that are 'close' and that have relatively low SSE
  - Can use these steps during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

Bisecting K-means

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

```
Initialize the list of clusters to contain the cluster containing all points.
repeat
   Select a cluster from the list of clusters
   for i = 1 to number_of_iterations do
      Bisect the selected cluster using basic K-means
   end for
   Add the two clusters from the bisection with the lowest SSE to the list of clusters.
until Until the list of clusters contains K clusters
```

Bisecting K-means Example

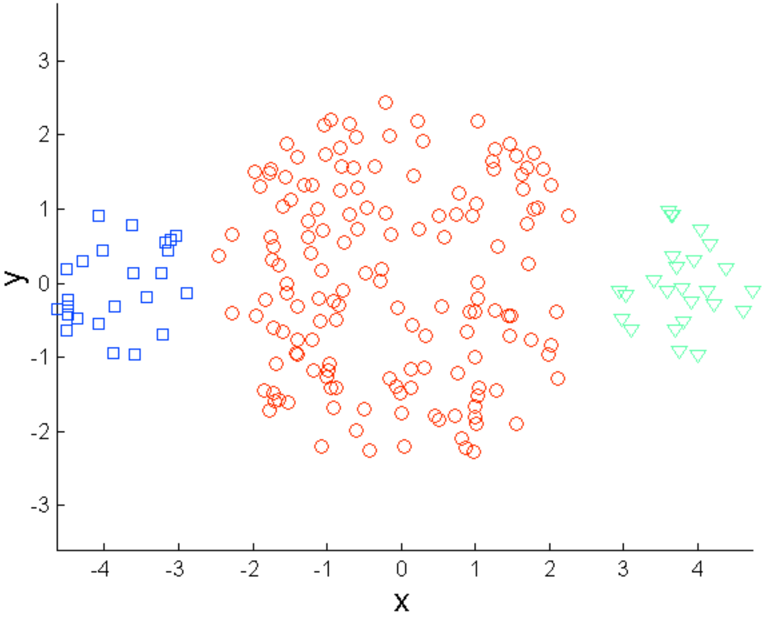


Limitations of K-means

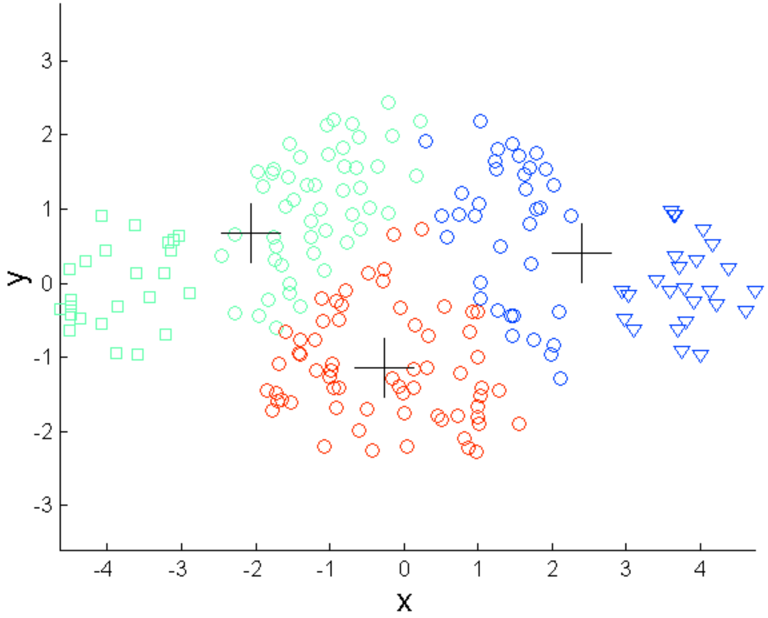
- K-means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes

- K-means has problems when the data contains outliers.

Limitations of K-means: Differing Sizes

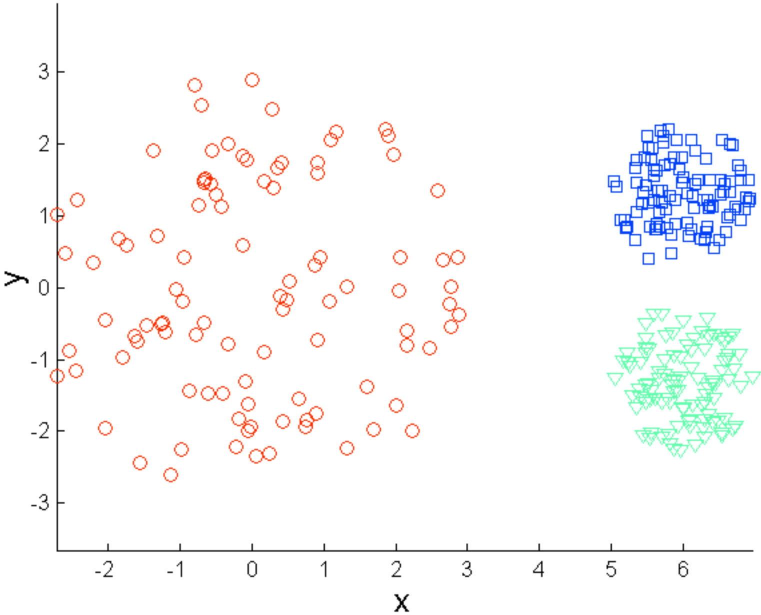


Original Points

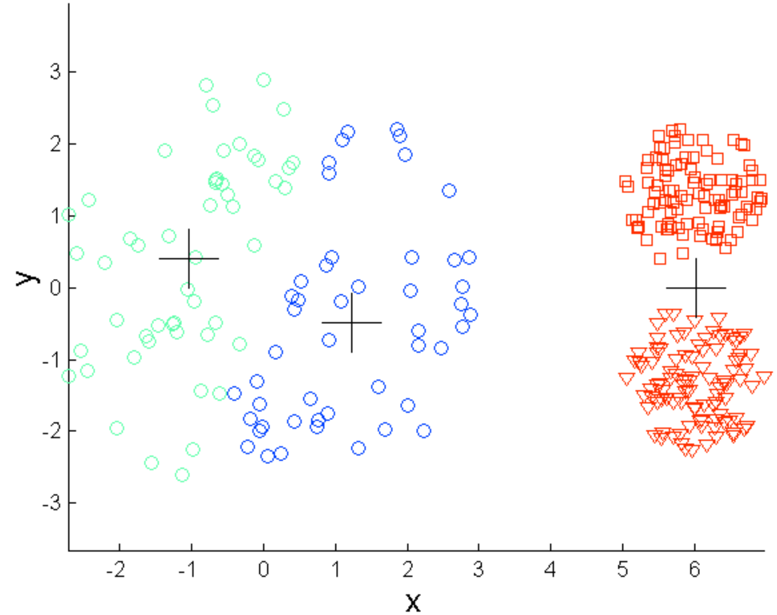


K-means (3 Clusters)

Limitations of K-means: Differing Density

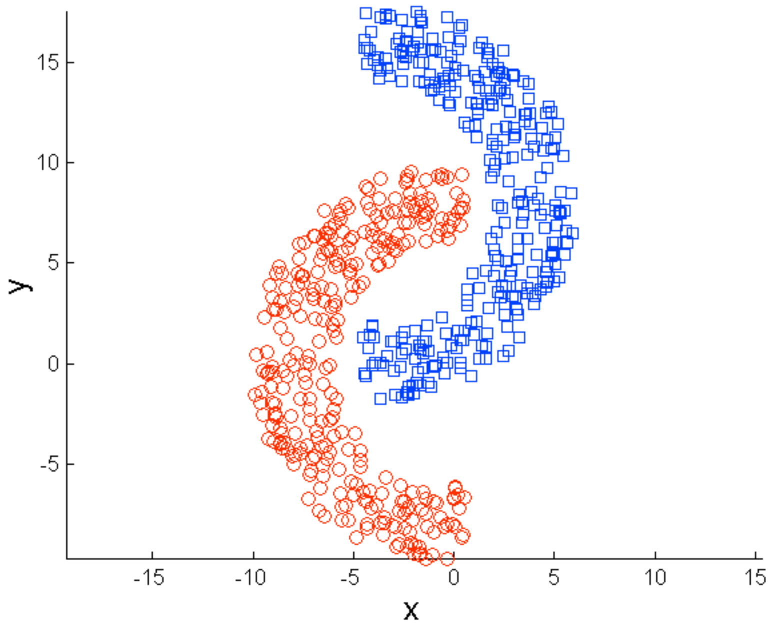


Original Points

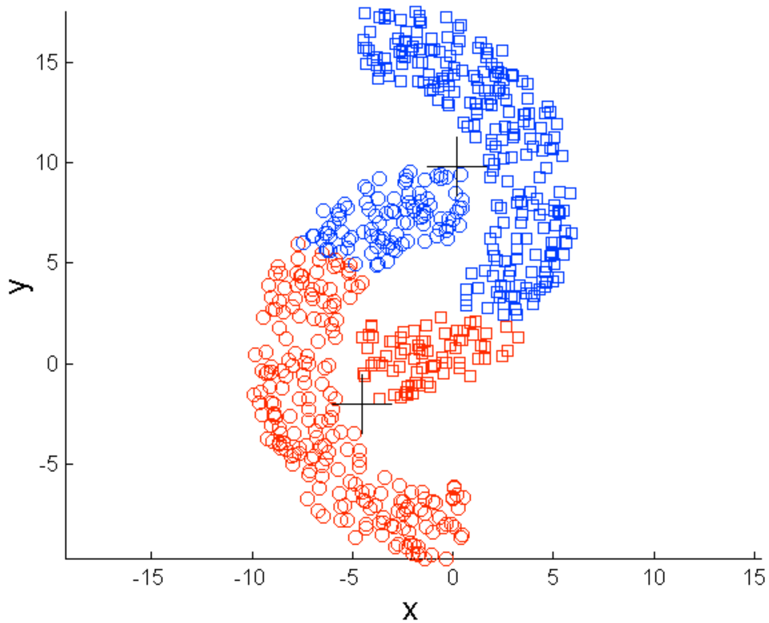


K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes

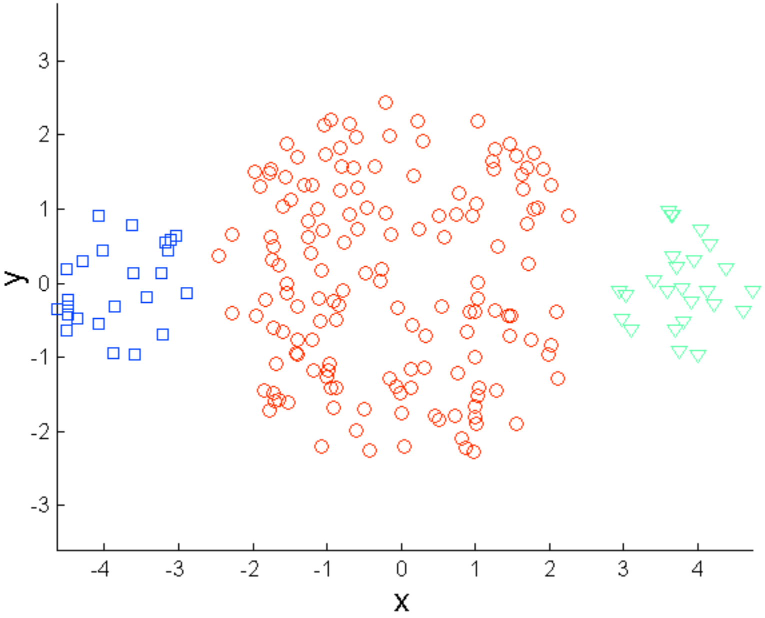


Original Points

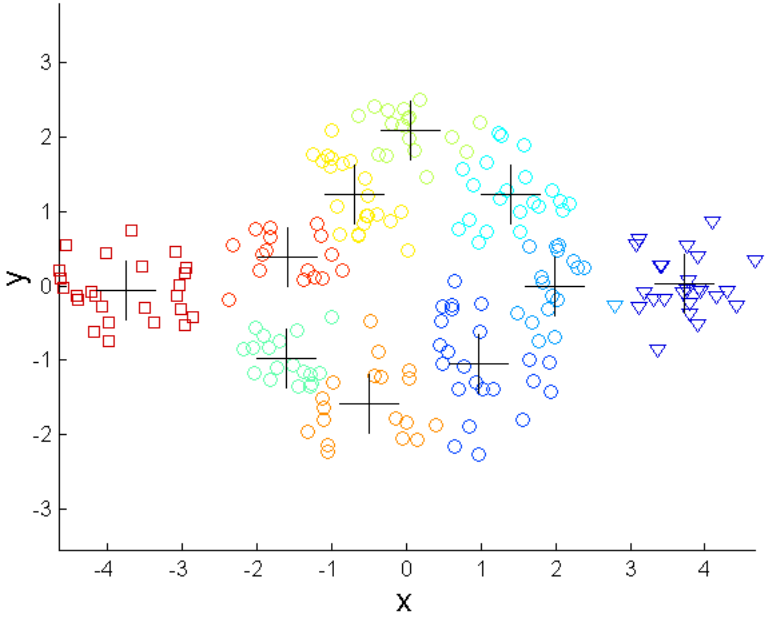


K-means (2 Clusters)

Overcoming K-means Limitations



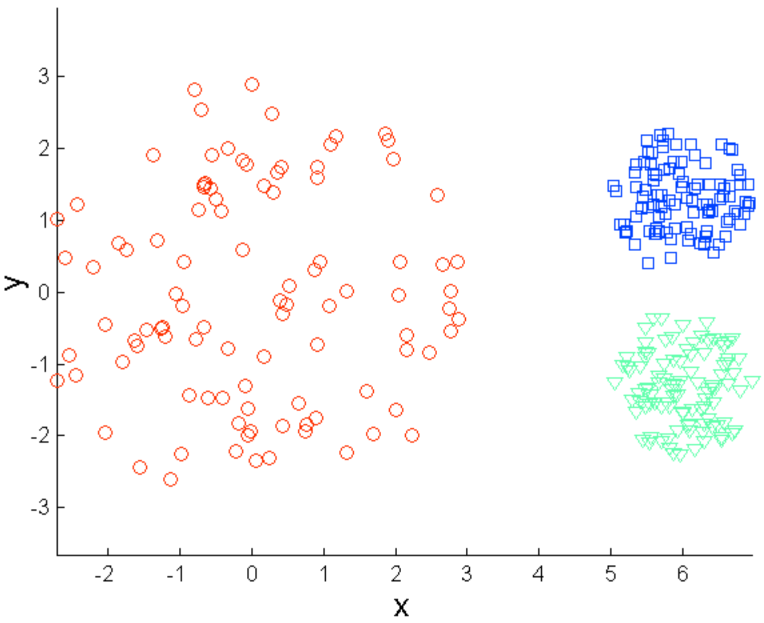
Original Points



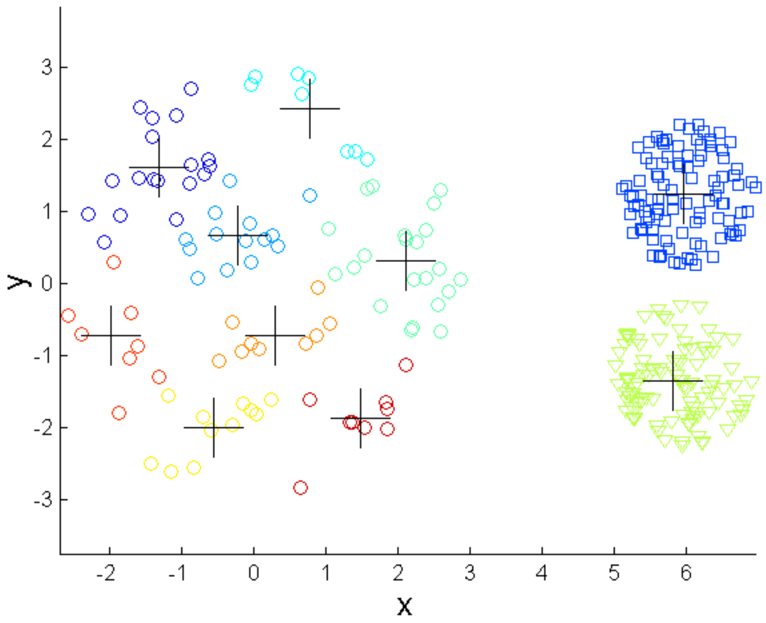
K-means Clusters

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

Overcoming K-means Limitations

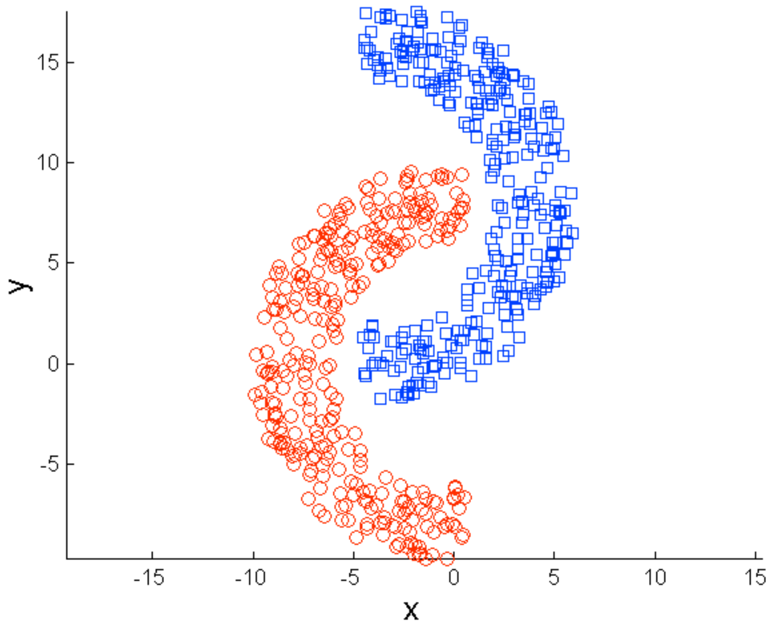


Original Points

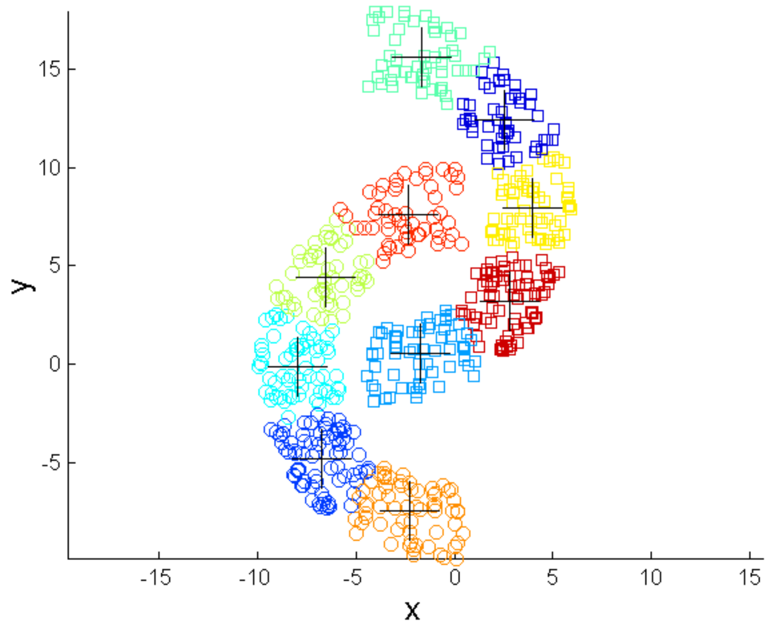


K-means Clusters

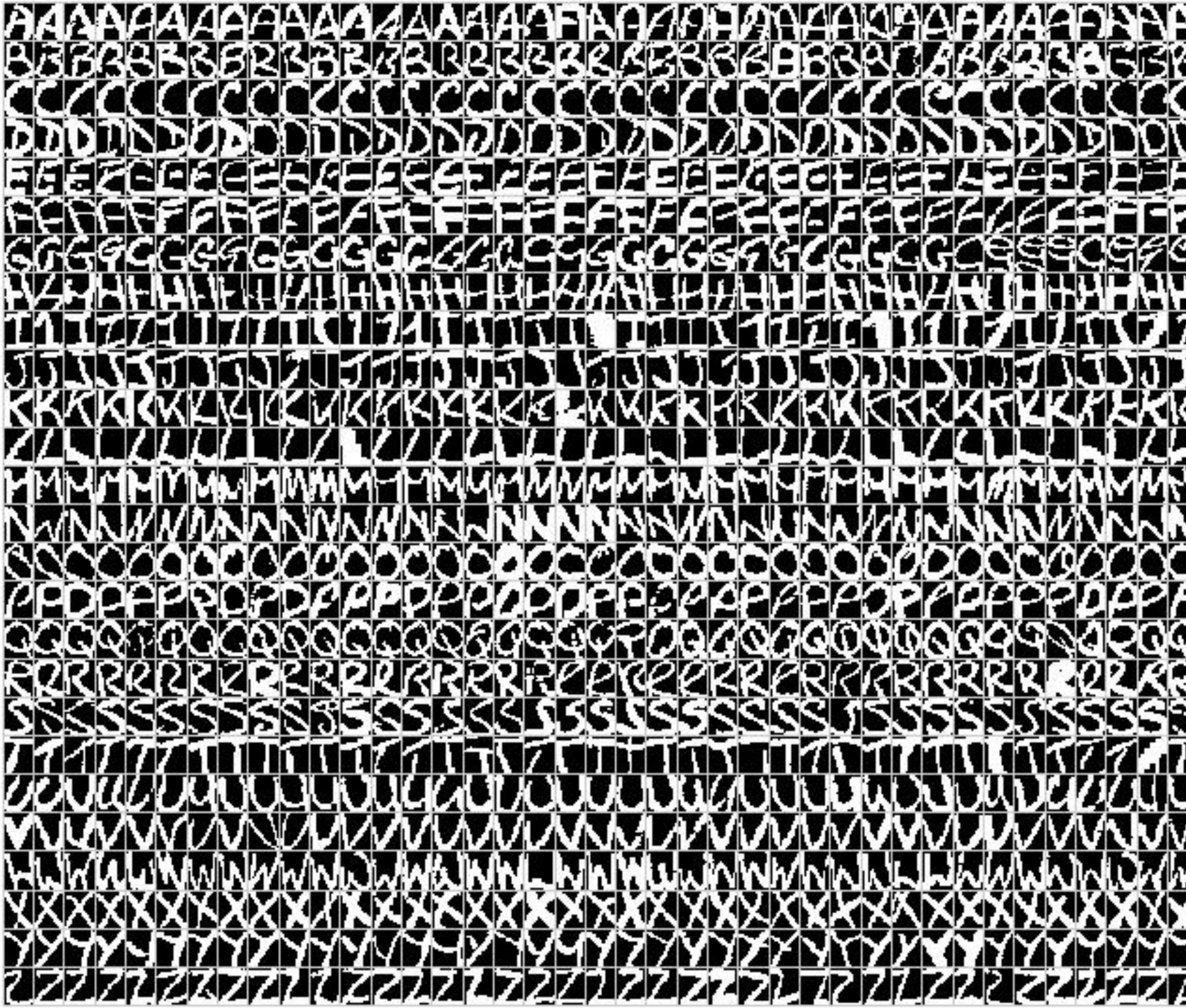
Overcoming K-means Limitations



Original Points



K-means Clusters



Vector Quantization: Store large number of information using codebooks