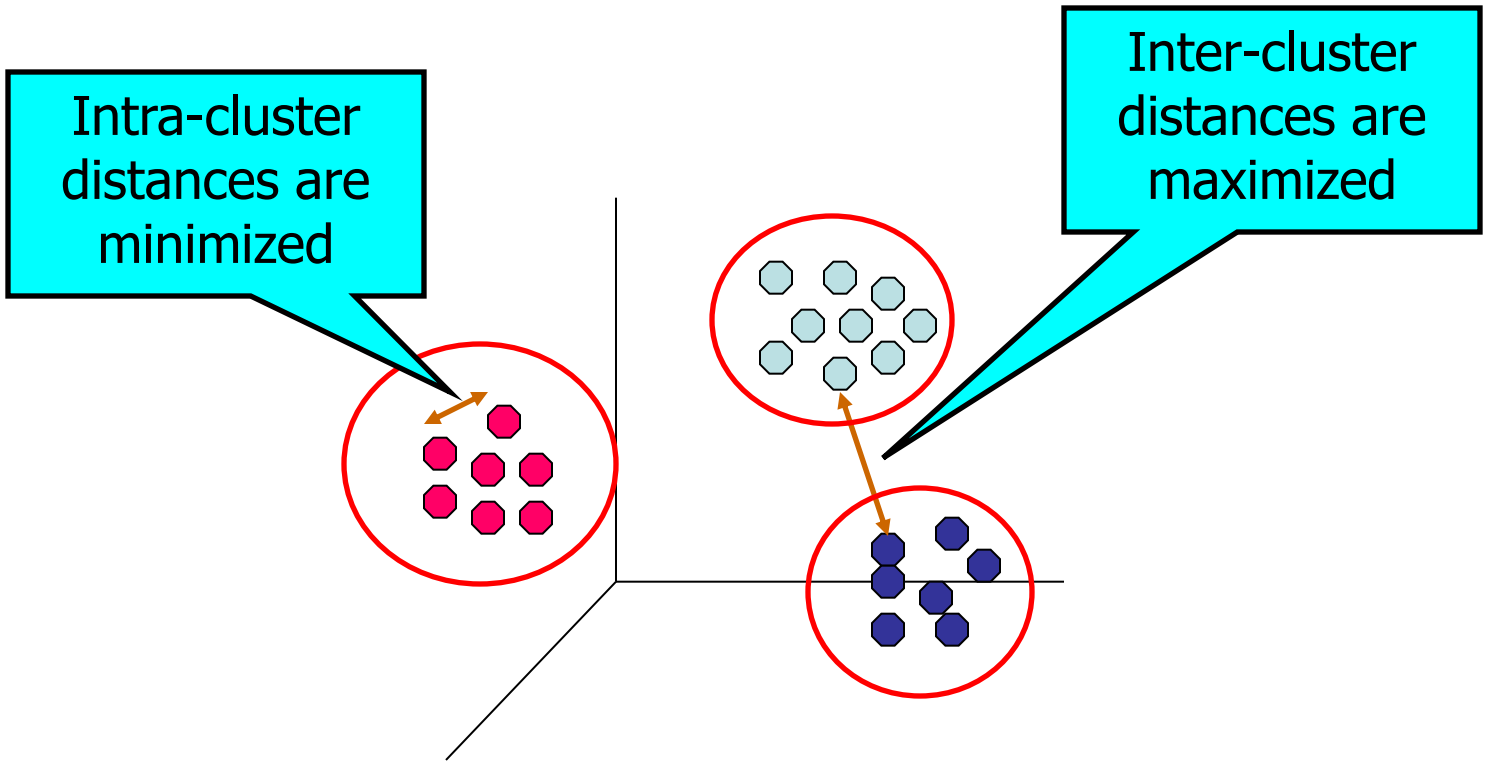
Cluster Analysis I

What is Cluster Analysis?

- Finding groups of objects
  - such that the objects in a group will be **similar** to one another and
  - **dissimilar** from the objects in other groups

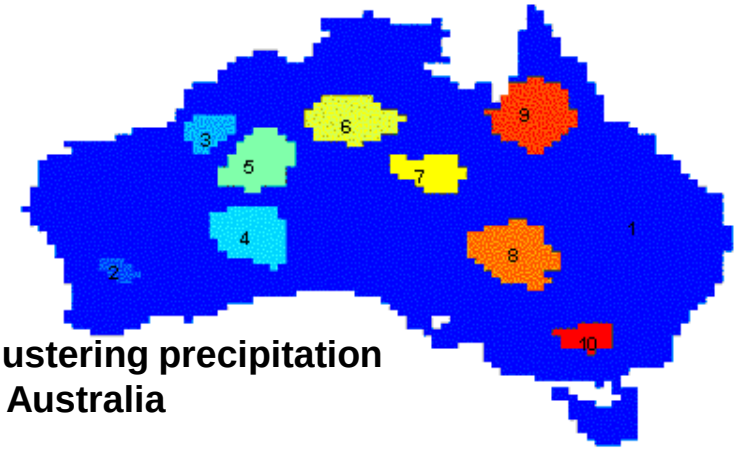


Applications of Cluster Analysis

- **Clustering for Understanding**
  - Group related documents for browsing
  - Group genes and proteins that have similar functionality
  - Group stocks with similar price fluctuations
  - Segment customers into a small number of groups for additional analysis and marketing activities.

- **Clustering for Summarization**
  - Reduce the size of large data sets

	Discovered Clusters	*Industry Group*
1	Applied-Matl-DOWN,Bay-Network-Down,3-COM-DOWN, Cabletron-Sys-DOWN,CISCO-DOWN,HP-DOWN, DSC-Comm-DOWN,INTEL-DOWN,LSI-Logic-DOWN, Micron-Tech-DOWN,Texas-Inst-Down,Tellabs-Inc-Down, Natl-Semiconduct-DOWN,Oracl-DOWN,SGI-DOWN, Sun-DOWN	Technology1-DOWN
2	Apple-Comp-DOWN,Autodesk-DOWN,DEC-DOWN, ADV-Micro-Device-DOWN,Andrew-Corp-DOWN, Computer-Assoc-DOWN,Circuit-City-DOWN, Compaq-DOWN, EMC-Corp-DOWN, Gen-Inst-DOWN, Motorola-DOWN,Microsoft-DOWN,Scientific-Atl-DOWN	Technology2-DOWN
3	Fannie-Mae-DOWN,Fed-Home-Loan-DOWN, MBNA-Corp-DOWN,Morgan-Stanley-DOWN	Financial-DOWN
4	Baker-Hughes-UP,Dresser-Inds-UP,Halliburton-HLD-UP, Louisiana-Land-UP,Phillips-Petro-UP,Unocal-UP, Schlumberger-UP	Oil-UP



Clustering precipitation in Australia

SIMILARITY AND DISSIMILARITY

Similarity and Dissimilarity

- **Similarity**
  - Numerical measure of how alike two data objects are.
  - Higher when objects are more alike.

- **Dissimilarity (Distance)**
  - Numerical measure of how different are two data objects
  - Lower when objects are more alike

Euclidean Distance

- When all the attributes are continuous we can use the Euclidean Distance

$$dist = \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2}$$

Where n is the number of dimensions (attributes) and p_k and q_k are, respectively, the k^{th} attributes (components) or data objects p and q.

- Attribute scaling is necessary, if scales differ
  - E.g. weight, salary have different scales

Minkowski Distance

- Minkowski Distance is a generalization of Euclidean Distance

$$dist = \sqrt[r]{\sum_{k=1}^{n} | p_k - q_k |^r}$$

Where r is a parameter, n is the number of dimensions (attributes) and p_k and q_k are, respectively, the kth attributes (components) or data objects **p** and **q**.

Examples

- $r = 1$. City block (Manhattan, taxicab, L_1 norm) distance.
- $r = 2$. Euclidean distance
- $r \rightarrow \infty$. "supremum" (L_{max} norm, L_∞ norm) distance.
  - This is the maximum difference between any component of the vectors

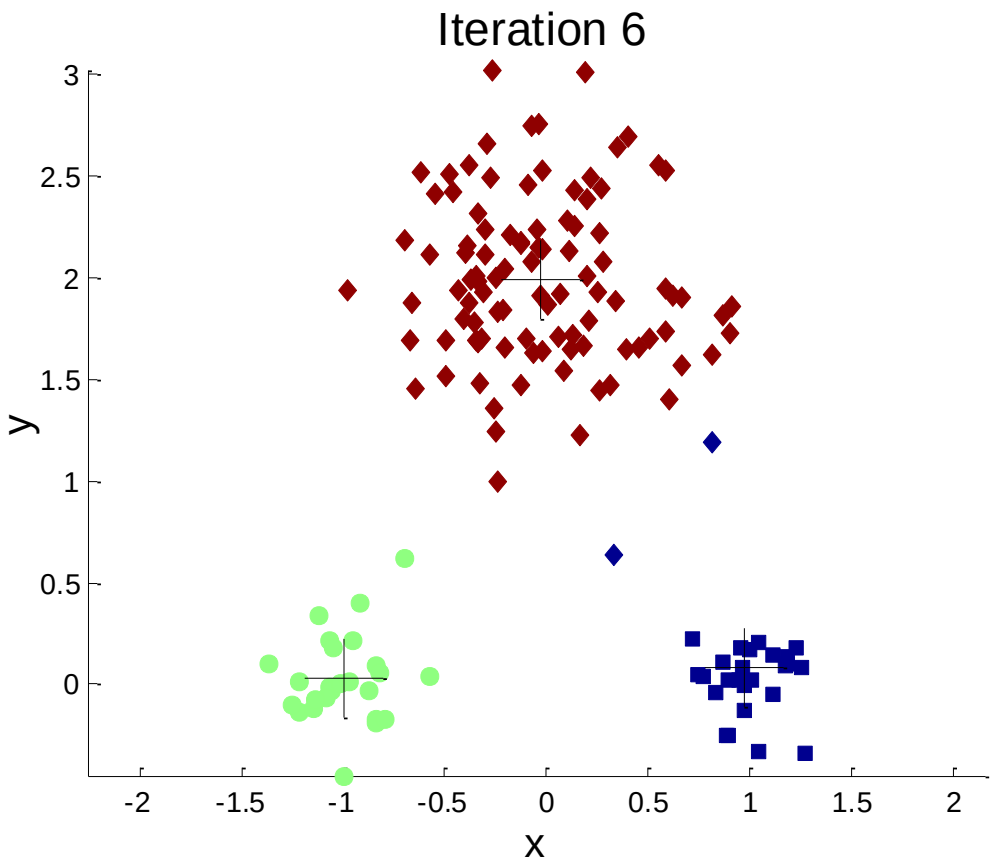
ALGORITHMS

K-means Clustering

- 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
- Basic algorithm is very simple

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

Example



K-means Clustering – Details

- Initial centroids may be chosen randomly.
  - Clusters produced vary from one run to another.
  - Rerun several times and pick the clustering with the smallest SSE (see next slide).
- The centroid is (typically) the mean of the points in the cluster.
- 'Closeness' is measured by Euclidean distance, cosine similarity, etc.

- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to 'Until relatively few points change clusters'

Evaluating K-means Clusters

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

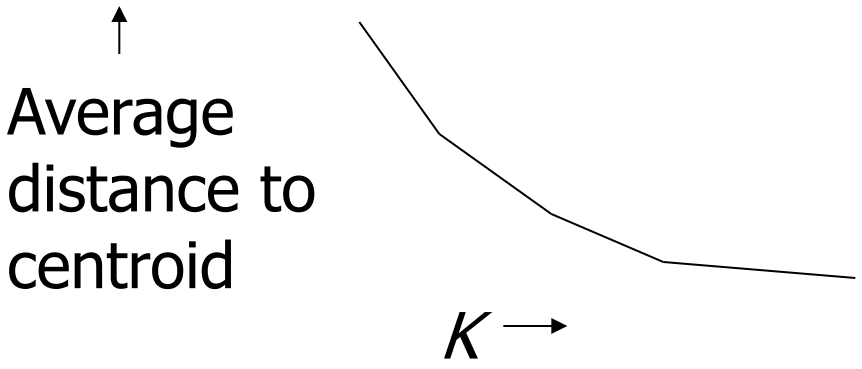
$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} [dist(m_i, x)]^2$$

x is a data point in cluster C_i and

m_i is the centroid for cluster C_i

Reducing SSE

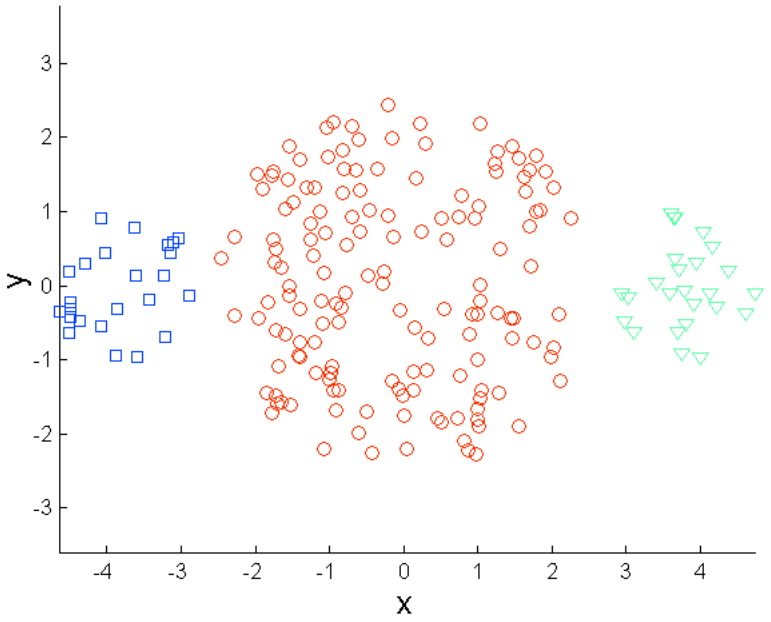
- Obvious way to reduce the SSE is to find more clusters, i.e., to use a larger *K*.
- Try different *K*, looking at the change in the average distance to centroid, as *K* increases.
- Average falls rapidly until right *K*, then changes little.



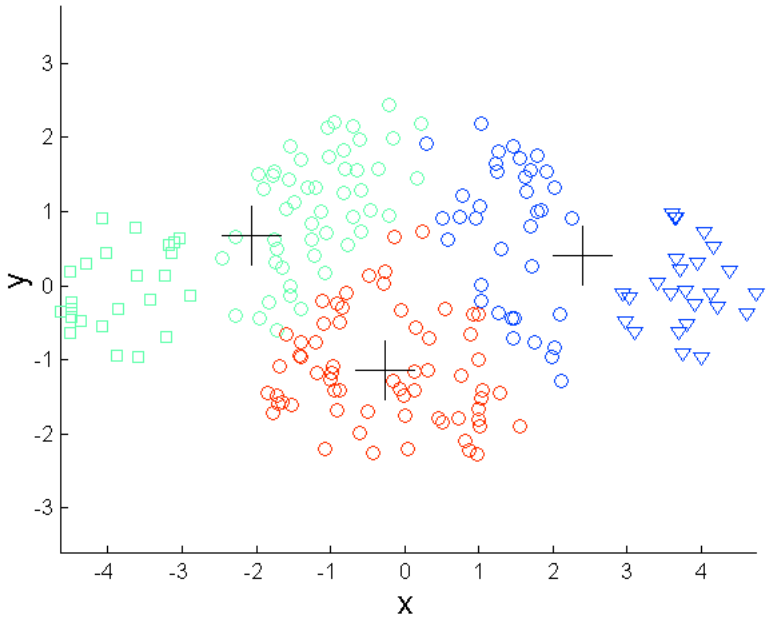
Limitations of K-means

- K-means has problems when (the real) clusters are of
  - Differing Sizes
  - Differing Densities
  - Non-globular shapes

Limitations of K-means: Differing Sizes

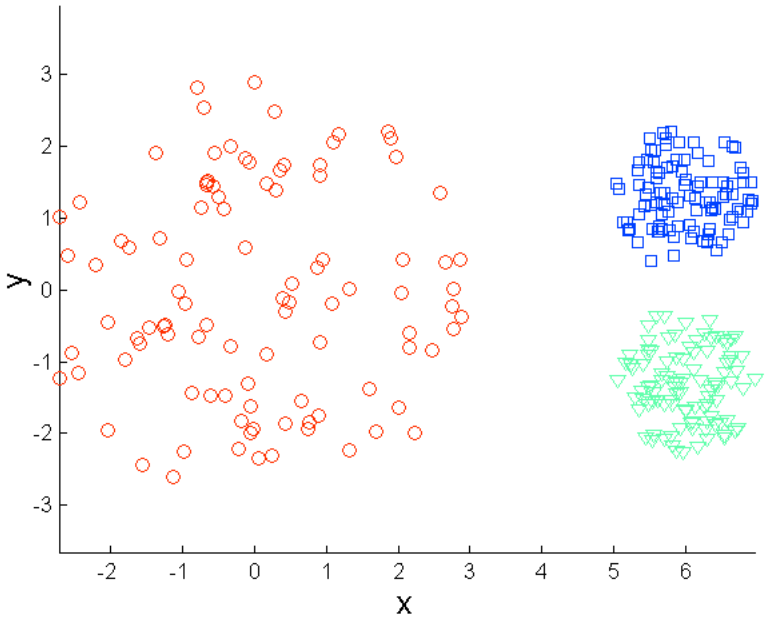


Original Points

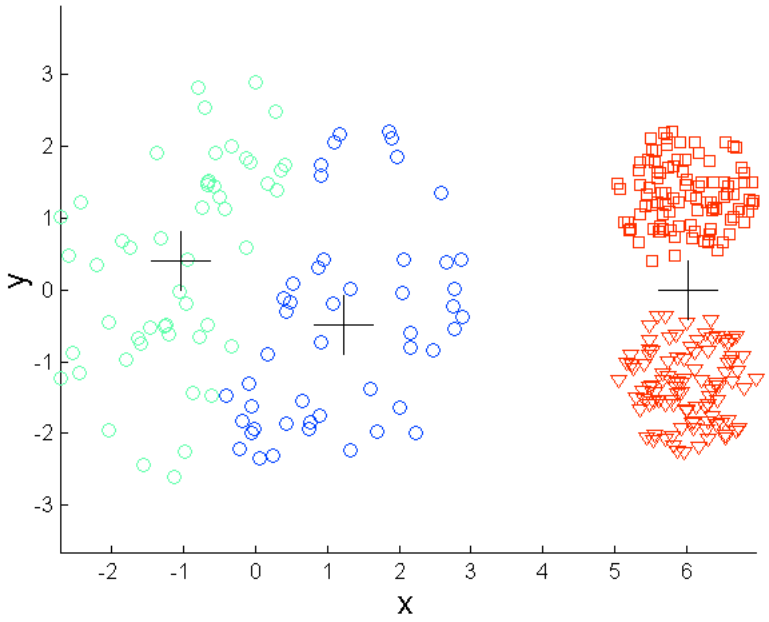


K-means (3 Clusters)

Limitations of K-means: Differing Density

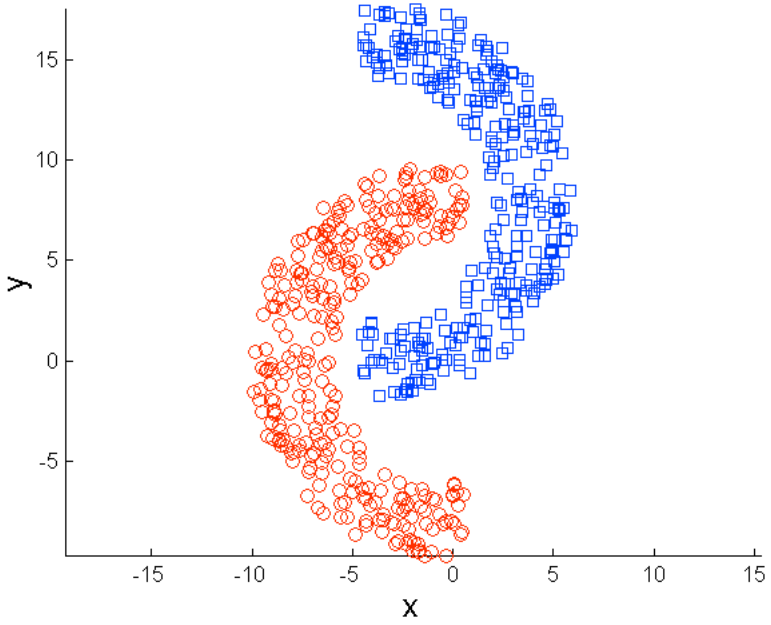


Original Points

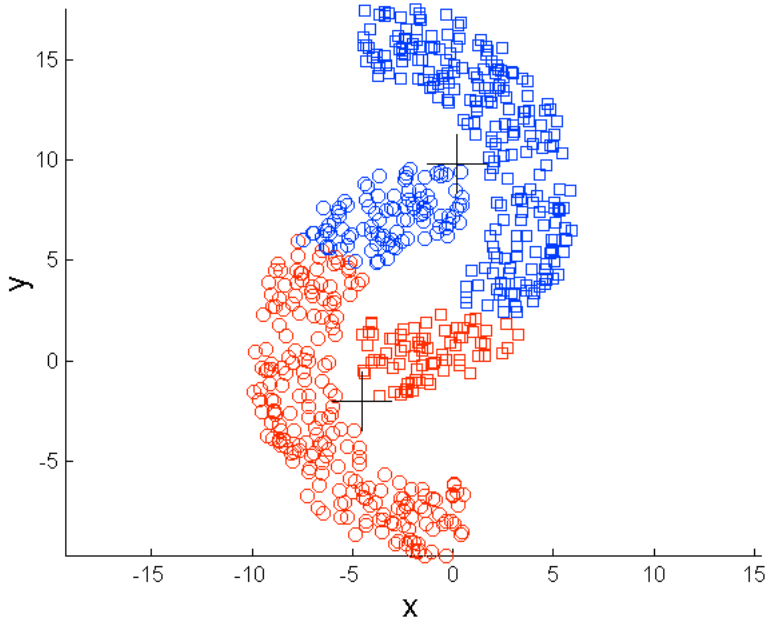


K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes

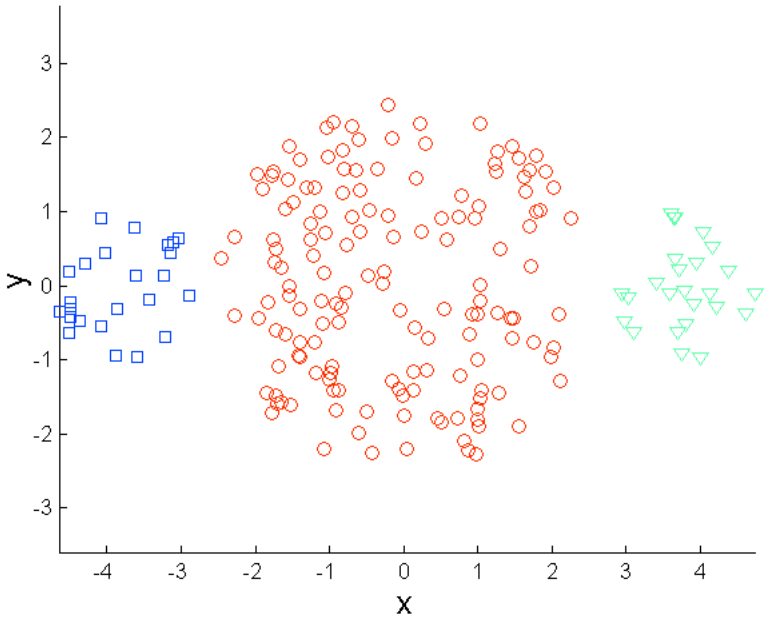


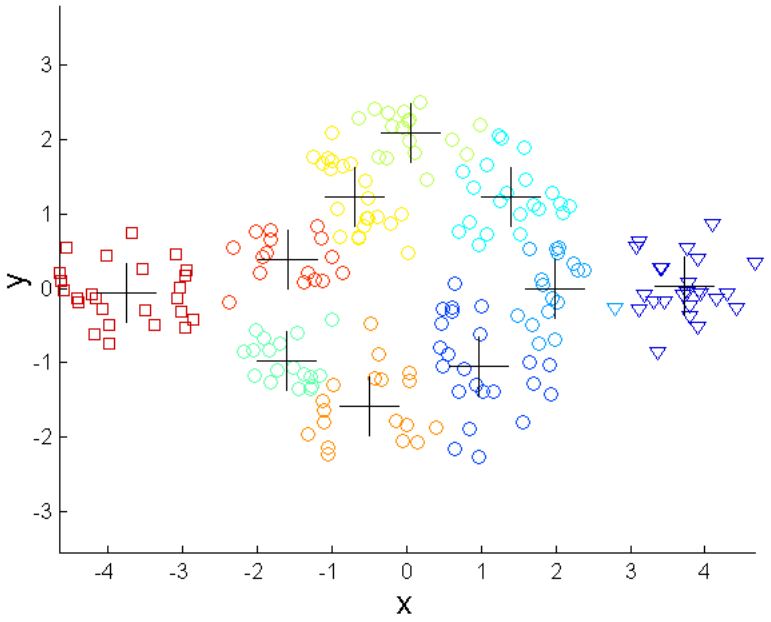
Original Points



K-means (2 Clusters)

Overcoming K-means Limitations





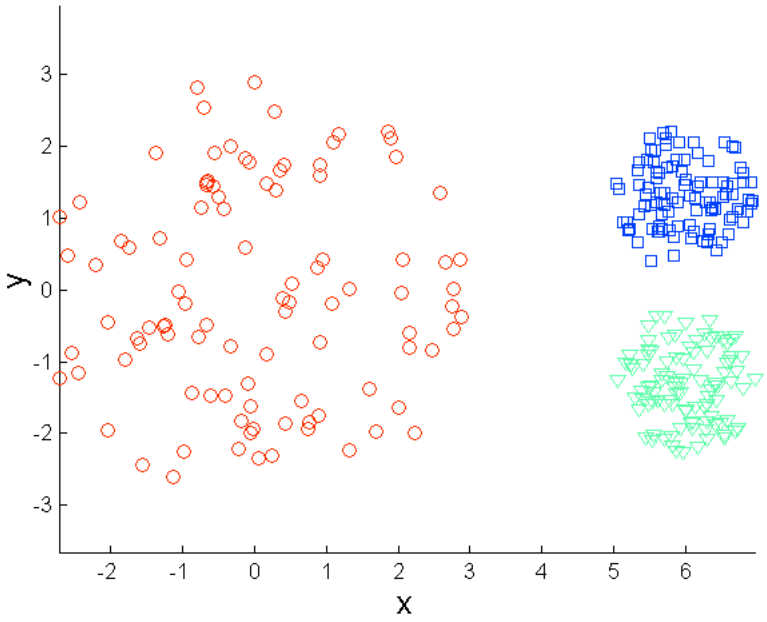
Original Points **K-means Clusters**

One solution is to use many clusters.

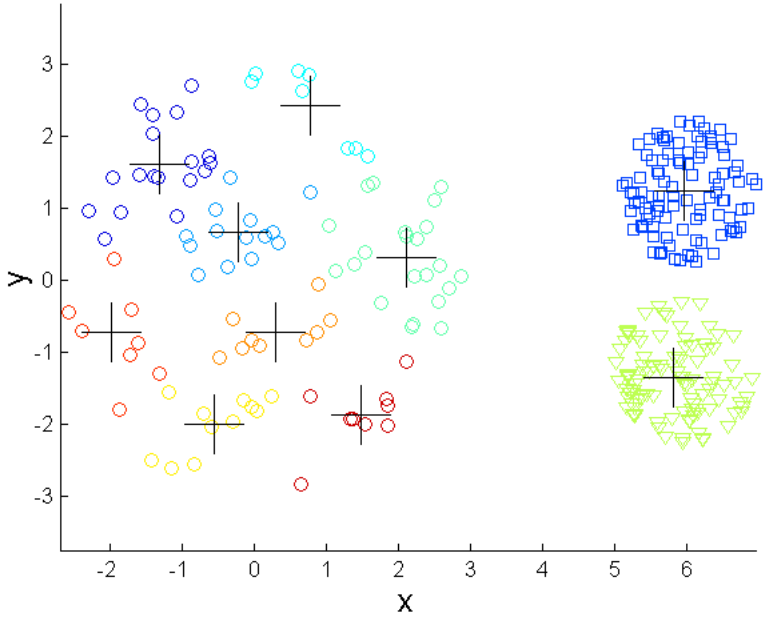
Find parts of clusters.

Apply **merge** strategy (merge clusters that would cause the least increase in SSE)

Overcoming K-means Limitations

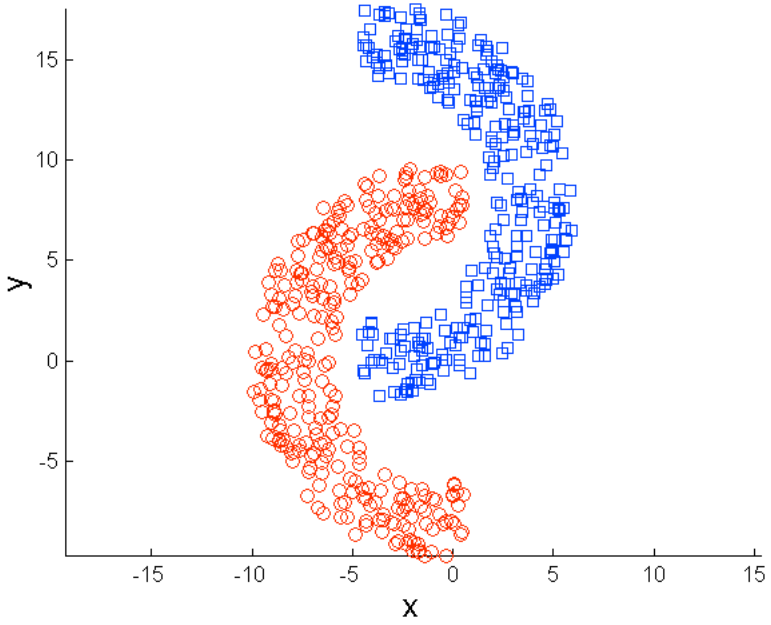


Original Points

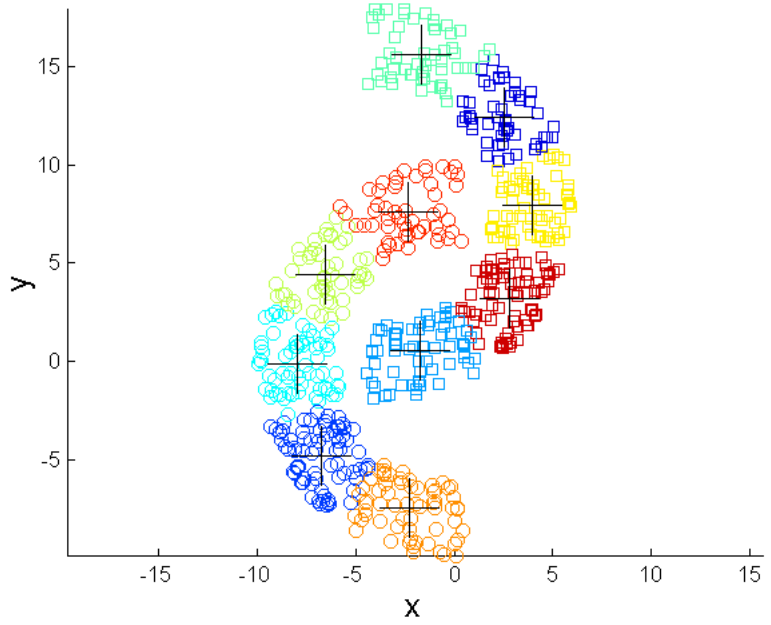


K-means Clusters

Overcoming K-means Limitations

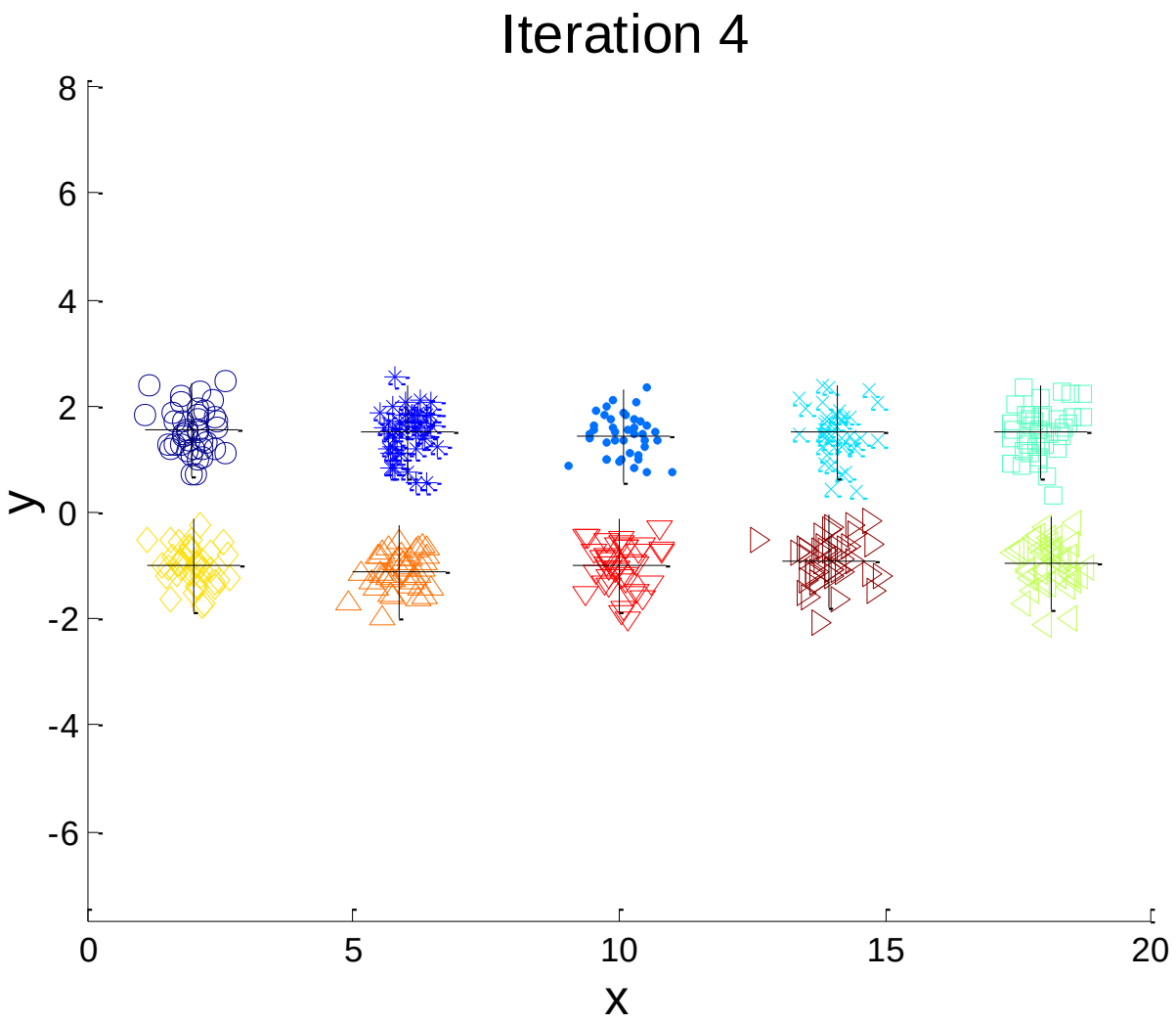


Original Points



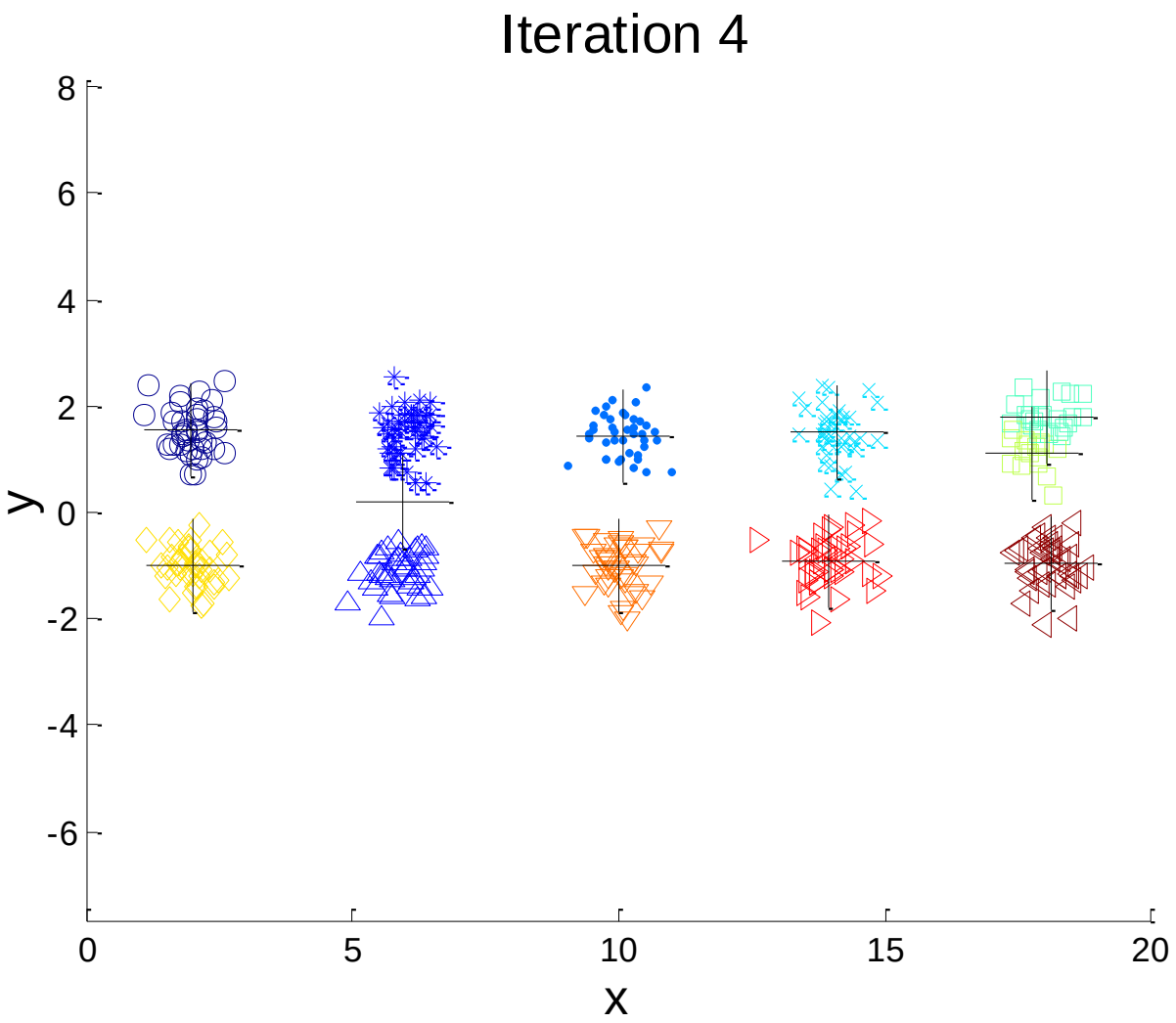
K-means Clusters

Importance of Choosing Initial Centroids



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

Importance of Choosing Initial Centroids



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

Problems with Selecting Initial Points

- The ideal would be to choose initial centroids, one from each true cluster. However, this is very 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
    - E.g. 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* = $10!/10^{10} = 0.00036$

Solutions to Initial Centroids Problem

- Multiple runs
  - Helps, but probability is not on your side

- Bisecting K-means
  - Not as susceptible to initialization issues

Bisecting Kmeans

- Straightforward extension of the basic Kmeans algorithm. Simple idea:

  To obtain K clusters, split the set of points into two clusters, select one of these clusters to split, and so on, until K clusters have been produced.

Algorithm

Initialize the list of clusters to contain the cluster consisting of all points.

repeat

  Choose and remove a cluster from the list of clusters.

    (biggest cluster or the cluster with the worst quality)

   //Perform several “trial” bisections of the chosen cluster.

  for *i* = 1 **to** number of trials **do**

    Bisect the selected cluster using basic *K*means (i.e. 2-means).

  end for

  Select the two clusters from the bisection with the lowest total SSE.

  Add these two clusters to the list of clusters.

until the list of clusters contains *K* clusters.

Bisecting K-means Example