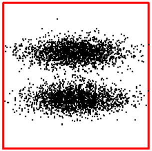
Clustering: k-means

Basics

- dataset D
- object $\mathbf{x} = (x_1, x_2, \ldots, x_d)$
  - every $\mathbf{x}$ is a point in a d-dimensional space

Clustering: groups the data into clusters

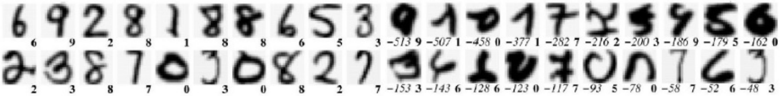
- objects in the same cluster have high similarity
- objects in different clusters are dissimilar to each other



- unlike classification, there is no class attribute in clustering
  - this is quite common in large databases, because assigning class labels to a large number of objects can be very costly
- unsupervised learning (on the other hand, classification is supervised learning)

Applications of Clustering

- business
  - discover distinct groups of customers based on their purchasing patterns
- biology
  - derive plant and animal taxonomies
- geographical data
  - group houses in a city according to house type and geographical location
- outlier detection
  - find credit card transactions that are not ordinary (fraud detection)

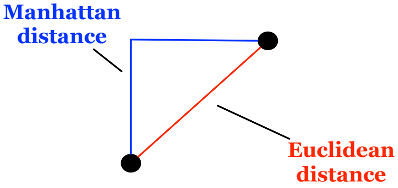


Dissimilarity

- measures how different two objects are (lower when objects are more alike)
- minimum dissimilarity is often 0

$\mathbf{x_1} = (x_{11}, x_{12}, \ldots, x_{1d})$ and $\mathbf{x_2} = (x_{21}, x_{22}, \ldots, x_{2d})$

- Euclidean distance: $dist(\mathbf{x_1}, \mathbf{x_2}) = \sqrt{\sum_{i=1}^{d} (x_{1i} - x_{2i})^2}$
- Manhattan distance: $dist(\mathbf{x_1}, \mathbf{x_2}) = \sum_{i=1}^{d} |x_{1i} - x_{2i}|$



- standardization is necessary if the scales of the attributes vary considerably

Similarity

- measures how alike two objects are
  - higher when objects are more alike
  - often falls in the range $[0, 1]$

Example

cosine similarity: $\cos(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1 \cdot \mathbf{x}_2}{\|\mathbf{x}_1\| \cdot \|\mathbf{x}_2\|}$

- $\mathbf{x}_1 = (3, 2, 0, 5, 0, 0, 0, 2, 0, 0)$
  $\mathbf{x}_2 = (1, 0, 0, 0, 0, 0, 0, 1, 0, 2)$
- cosine similarity is: 0.31

Similarity Measures for Binary Attributes

- $\mathbf{x_1}$ and $\mathbf{x_2}$: two objects containing d binary attributes
  - f_{00}: number of attributes with $\mathbf{x_1}$ is 0 and $\mathbf{x_2}$ is 0
  - f_{01}: number of attributes with $\mathbf{x_1}$ is 0 and $\mathbf{x_2}$ is 1
  - f_{10}: number of attributes with $\mathbf{x_1}$ is 1 and $\mathbf{x_2}$ is 0
  - f_{11}: number of attributes with $\mathbf{x_1}$ is 1 and $\mathbf{x_2}$ is 1
- simple matching coefficient: $SMC = \frac{f_{11}+f_{00}}{f_{01}+f_{10}+f_{00}+f_{11}} = \frac{f_{11}+f_{00}}{d}$

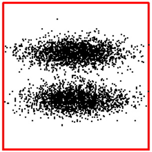
Example

- $\mathbf{x_1} = (1,0,0,0,0,0,0,0,0,0), \mathbf{x_2} = (0,0,0,0,0,0,1,0,0,1)$
- $f_{00} = 7, f_{01} = 2, f_{10} = 1, f_{11} = 0$; $SMC = \frac{0+7}{2+1+0+7} = 0.7$

Partitioning Methods

Basic idea

- organize the N objects into k partitions ($k < N$), where each partition represents a cluster
  - objects within a cluster are similar, whereas objects of different clusters are dissimilar



- the clusters are formed to minimize an objective criterion, such as a distance function

Will focus on the following partitioning method

- k-means

k-means

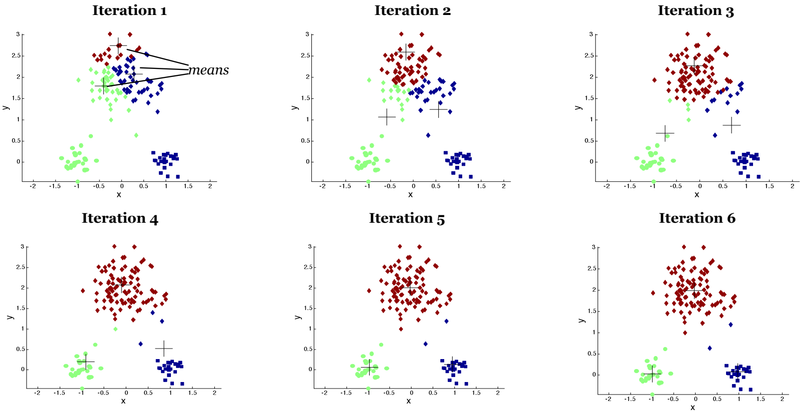
- C_i: a cluster; N_i: number of points in C_i
- mean of this cluster: $\mathbf{c}_i = \frac{1}{N_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}$
  - can be regarded as the centroid or center of gravity of the cluster

k-means algorithm

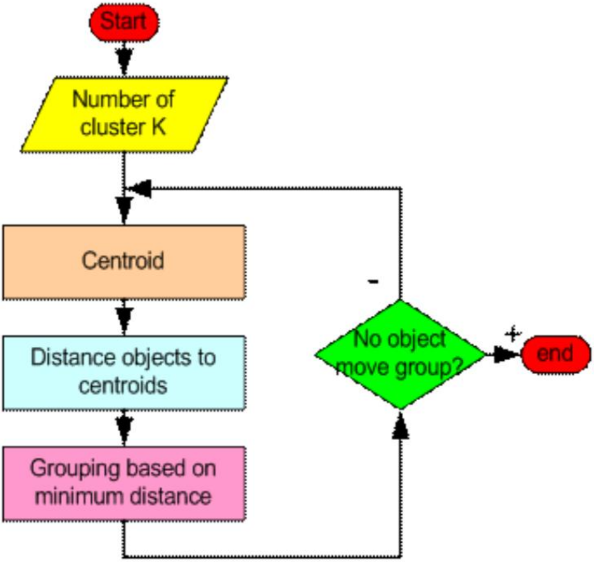
input: dataset D, number of clusters k
output: k clusters

1. randomly select k points from D as the initial means
2. **repeat**
3. form k clusters by assigning each point to its closest mean
4. recompute the mean for every cluster
5. **until** no changes in the mean
6. **return** the k clusters

Example



demo


Start
Number of cluster K
Centroid
Distance objects to centroids
Grouping based on minimum distance
No object move group?
-
+
end

Issues: Local Minimum

- objective criterion that k-means attempts to minimize

$$\text{(sum squared error) } SSE = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \mathbf{c}_i\|^2$$

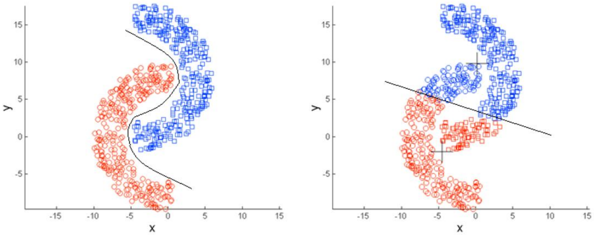
- converges to a local minimum $\rightarrow$ suboptimal clustering

  - different initial starts of the means gives different final answers $\rightarrow$ initial selection is very important

how to alleviate this problem?

1. multiple runs
2. pick the solution with minimum SSE

Issues: Implicit Assumption on Cluster Shape

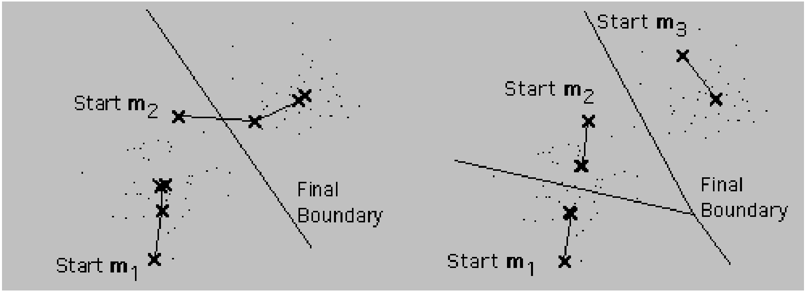


- can get wrong results when clusters have other shapes

Issues: Number of Clusters

You have to pick the number of clusters

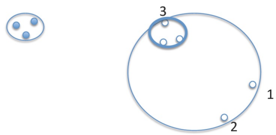
- in general, clustering result depends on k



Issues: Outliers

- the means may be fictitious (i.e., non-existent in the dataset)
- k-means is sensitive to outliers

Example



- by removing points 1 and 2, we obtain much tighter clusters