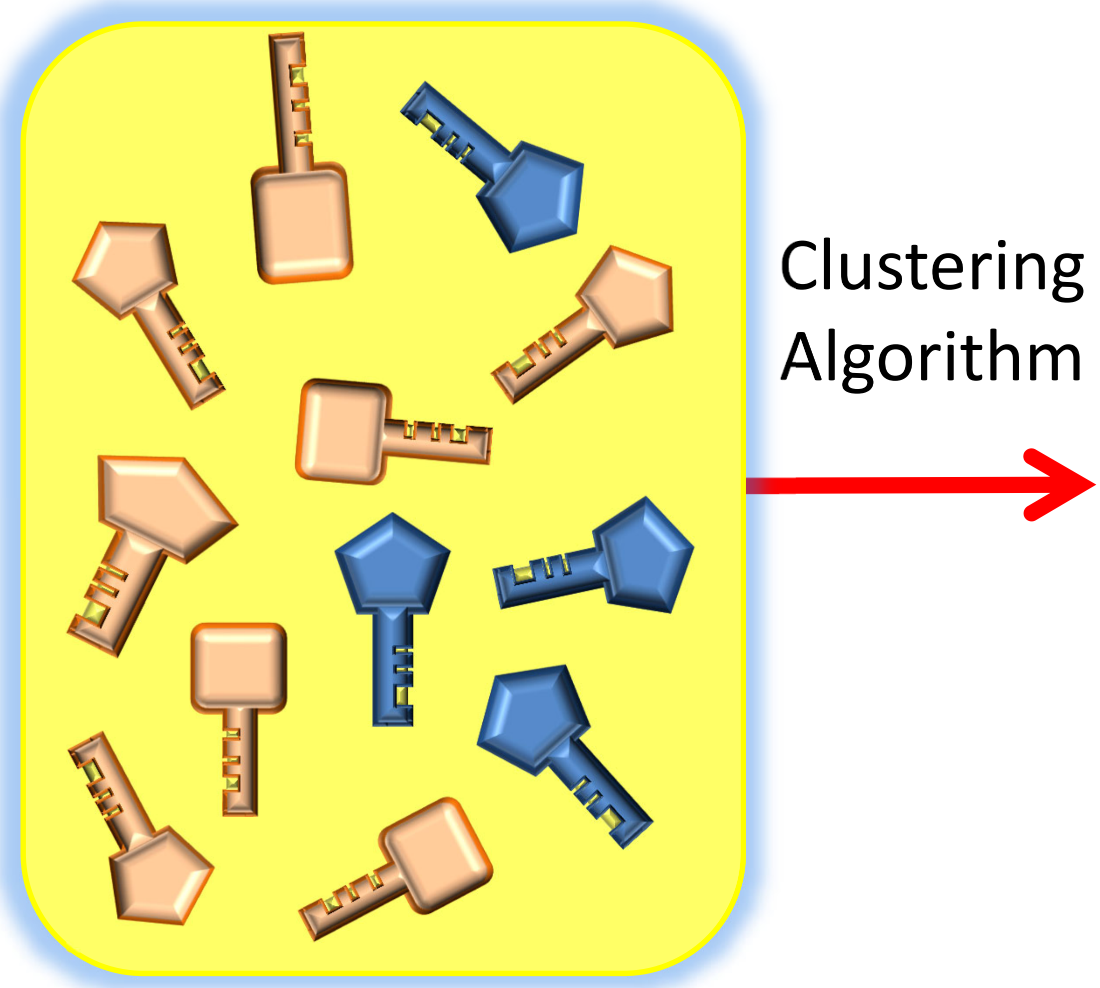
ECLT5810/SEEM5750
Clustering

What is Cluster Analysis?

- Cluster: a collection of data objects
  - Similar to one another within the same cluster
  - Dissimilar to the objects in other clusters
- Cluster analysis
  - Grouping a set of data objects into clusters
- Clustering is unsupervised classification: no predefined classes
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

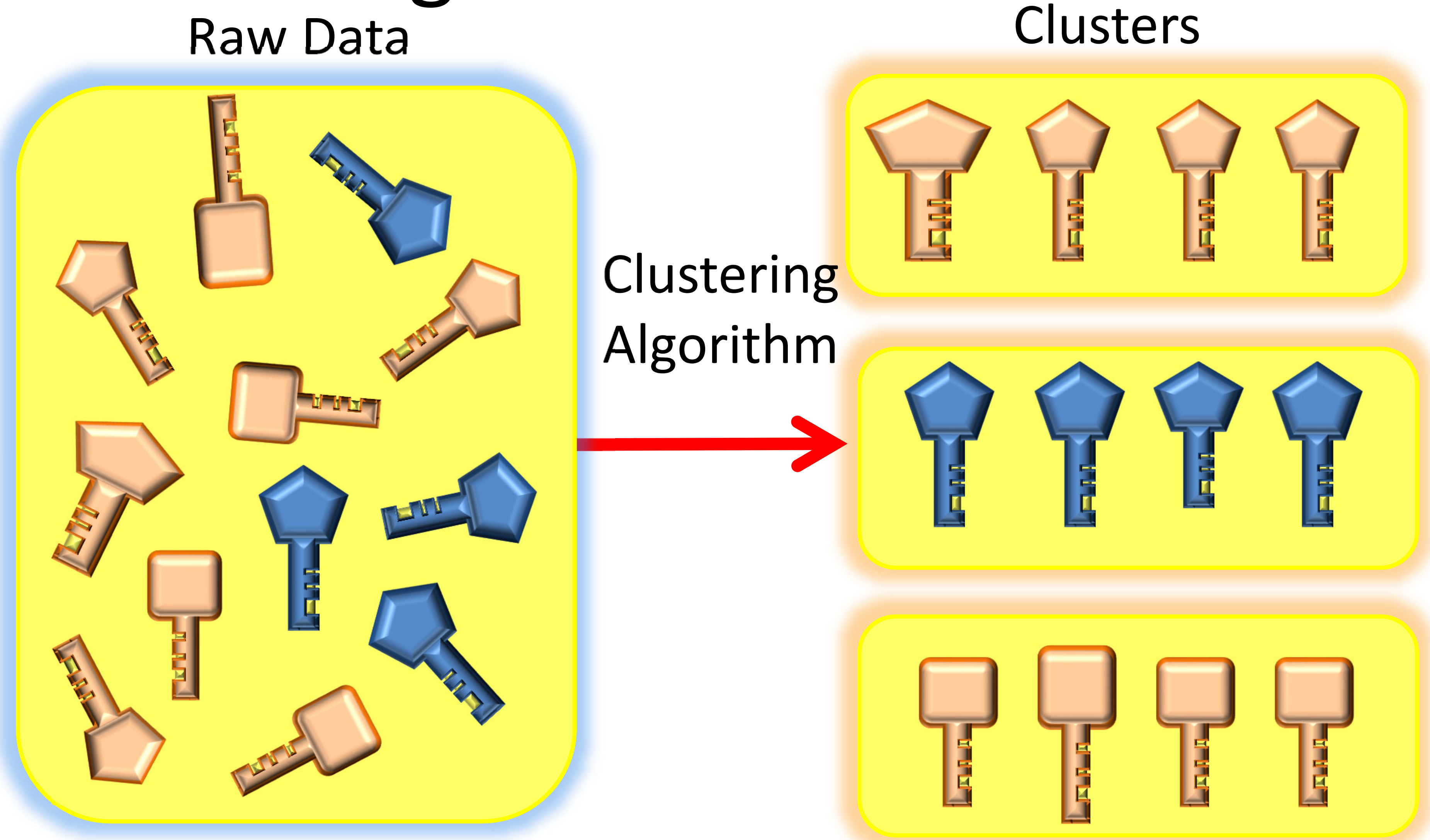
Clustering Method

Raw Data



Clustering Algorithm

Clustering Method



A clustering method attempts to find natural groups of data (objects) based on similarity

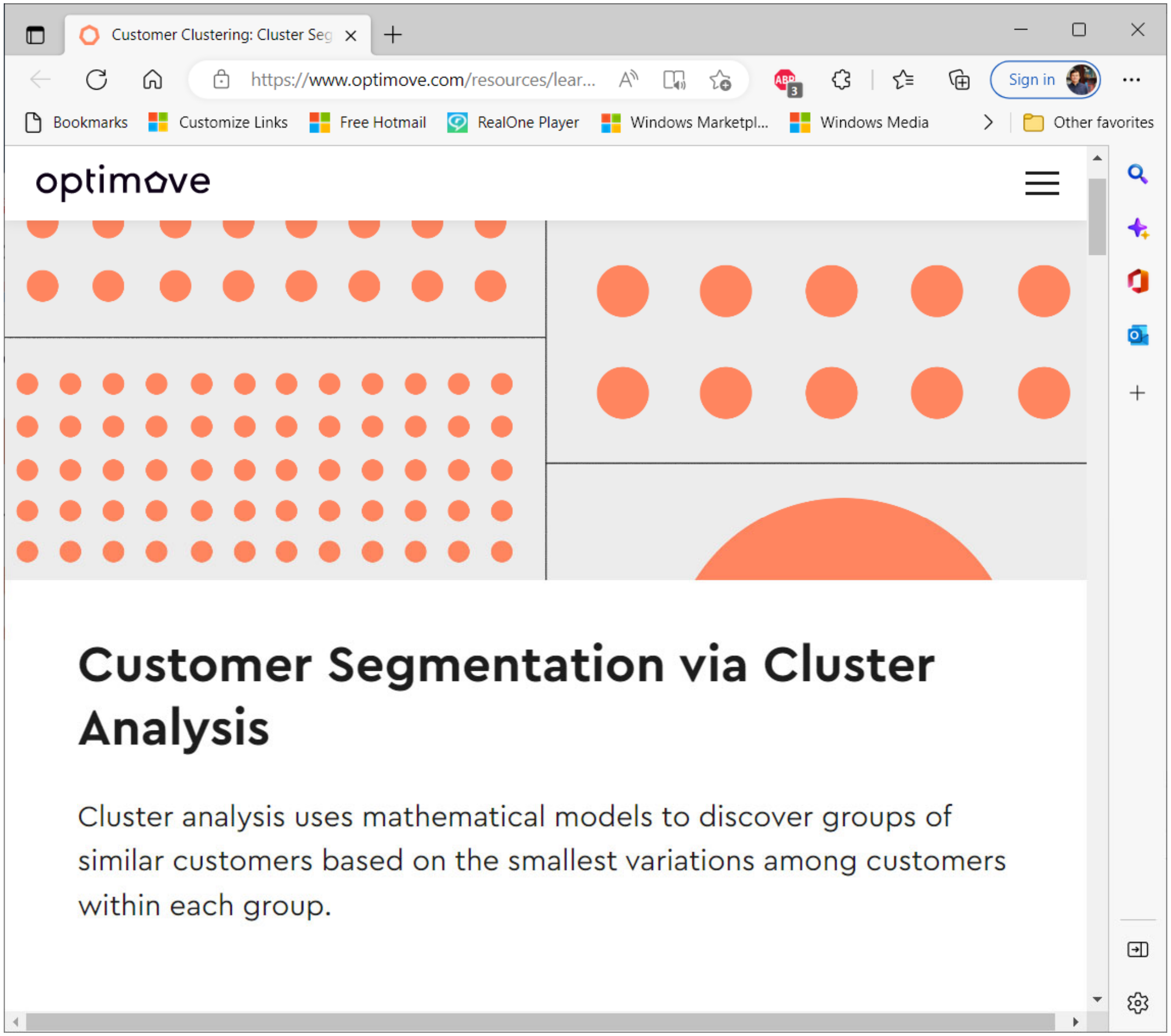
General Applications of Clustering

- Pattern Recognition
- Spatial Data Analysis
  - create thematic maps in GIS by clustering feature spaces
  - detect spatial clusters and explain them in spatial data mining
- Economic Science (especially market segmentation)
  - Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- WWW
  - Document classification
  - Cluster Weblog data to discover groups of similar access patterns

Marketing

- Create *market segmentation* of customers
  - Break down a wide market into identifiable and homogeneous groups of customers
- Understand customers
- Advertise to each segment with targeted strategy, content, deals, etc.

Marketing



https://www.optimove.com/resources/learning-center/customer-segmentation-via-cluster-analysis

What Is Good Clustering?

- A good clustering method will produce high quality clusters with
  - high intra-class similarity
  - low inter-class similarity
- The quality of a clustering result depends on both the similarity measure used by the method and its implementation.
- The quality of a clustering method is also measured by its ability to discover some or all of the hidden patterns.

Requirements of Clustering in Data Mining

- Scalability
- Ability to deal with different types of *attributes*
- Discovery of clusters with *arbitrary shape*
- Minimal requirements for domain knowledge to determine input parameters
- Able to deal with noise and outliers
- Insensitive to order of input records
- High dimensionality
- Incorporation of user-specified constraints
- Interpretability and usability

Data Structures

- Data matrix
  - *n* objects x *p* attributes (variables)

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

- Dissimilarity matrix
  - store difference between *n* objects x *n* objects

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ : & : & : & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

Distance/Similarity

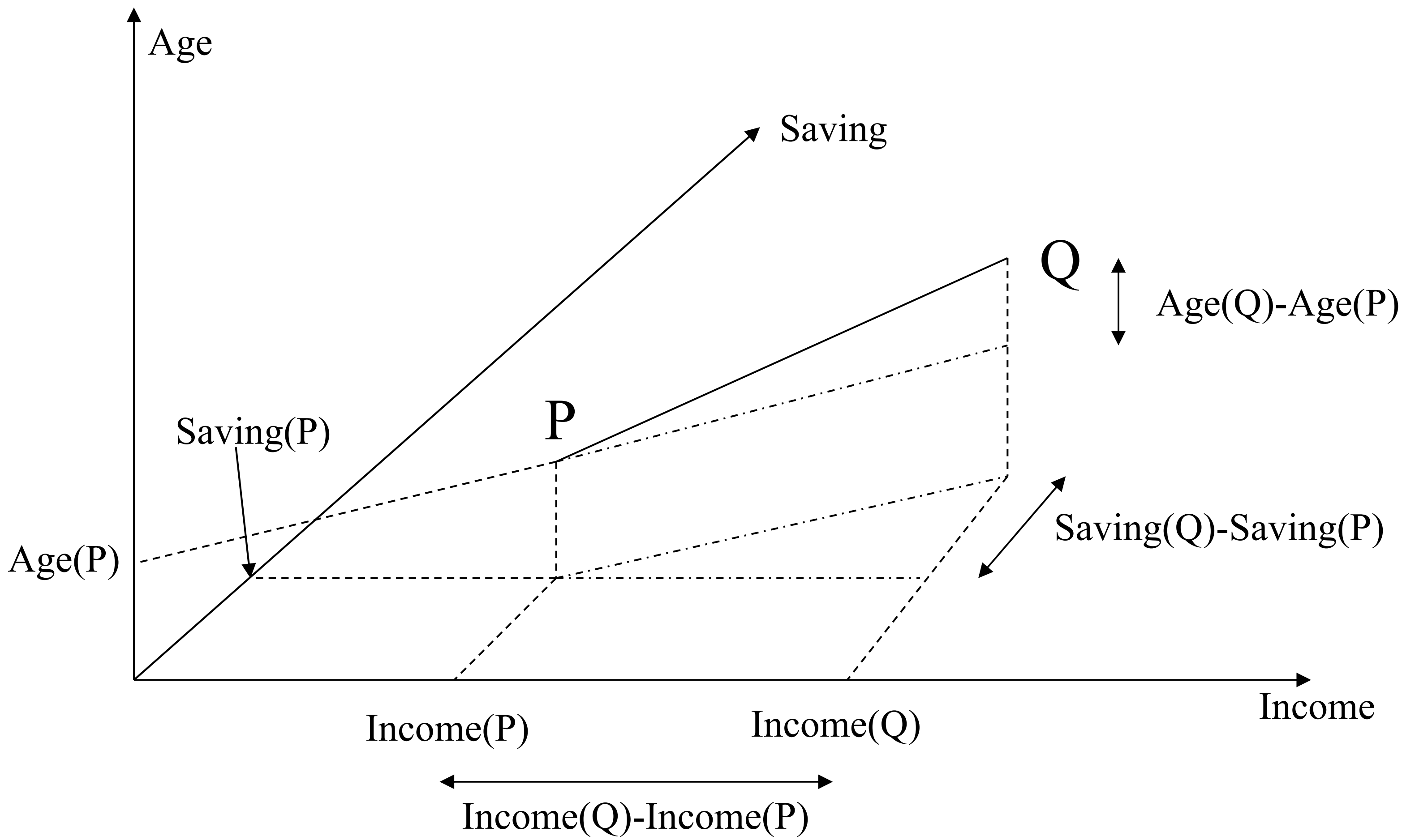
- Distances are normally used to measure the similarity or dissimilarity between two data objects
- Dissimilarity/Similarity metric: Similarity is expressed in terms of a distance function, which is typically metric:

$$d(i, j)$$

- There is a separate “quality” function that measures the “goodness” of a cluster.
- The definitions of distance functions are usually very different for *interval-scaled, Boolean, categorical,* and *ordinal* variables.
- It is hard to define “similar enough” or “good enough”
    - the answer is typically highly subjective.

Euclidean Distance

- Assume attribute values are real numbers



Euclidean Distance

- $\vec{x} = (x_1, x_2, \ldots, x_m)$
- $\vec{y} = (y_1, y_2, \ldots, y_m)$
- Formulation:

$$d(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

- Example
  - Let object A be represented as $\vec{x} = (5, 2, 3)$
  - Let object B be represented as $\vec{y} = (22, 17, 50)$
  - The Euclidean distance between A and B:

$$d(\vec{x}, \vec{y}) = \sqrt{(22-5)^2 + (17-2)^2 + (50-3)^2} \approx 52.18$$

Distance/Similarity Between Objects

- Assume that there are *p* numeric (interval-valued) attributes. Object *i* is represented by $i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ and object *j* is represented by $j = (x_{j1}, x_{j2}, \ldots, x_{jp})$.
- One example is *Minkowski distance*:

$$d(i,j) = \sqrt[q]{(|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \ldots + |x_{ip} - x_{jp}|^q)}$$

where *q* is a positive integer

- If *q = 1, d* is *Manhattan distance*

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{ip} - x_{jp}|$$

Distance/Similarity Between Objects

- If $q = 2$, d is *Euclidean distance*:

$$d(i,j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \ldots + |x_{ip} - x_{jp}|^2)}$$

  - Properties
    - $d(i,j) \geq 0$
    - $d(i,i) = 0$
    - $d(i,j) = d(j,i)$
    - $d(i,j) \leq d(i,k) + d(k,j)$
- Also, one can use *weighted distance*, *parametric Pearson product moment correlation*, or other *dissimilarity measures*

Data Transformation
Normalization

- Helps prevent attributes with large ranges outweigh ones with small ranges
  - Example:
    - income has range 2000-20000
    - age has range 10-100

Data Transformation

The Range Problem

	Income	Age
Peter	3,000	30
Mary	4,500	35
John	4,400	80

$$d(Peter, Mary) = \sqrt{(3000 - 4500)^2 + (30 - 35)^2} \approx 1500$$

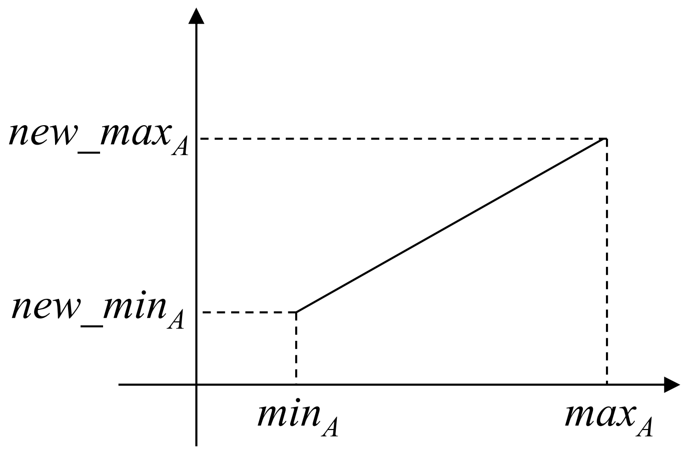
$$d(Peter, John) = \sqrt{(3000 - 4400)^2 + (30 - 80)^2} \approx 1400$$

- Before normalization, Peter is closer to John than Mary

Data Transformation
Normalization

- Min-Max normalization

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$



e.g. convert age=30 to range 0-1, when min=10,max=100.

new_age=(30-10)/(100-10)+0=2/9

Data Transformation
Normalization

$$new\_max = 1 \qquad new\_min = 0$$

$$min_{income} = 2{,}000 \qquad min_{age} = 0$$

$$max_{income} = 10{,}000 \qquad max_{age} = 100$$

	Income	Age
Peter	3,000	30
Mary	4,500	35
John	4,400	80

Min-Max normalization →

	Income	Age
Peter	0.125	0.30
Mary	0.3125	0.35
John	0.30	0.80

Data Transformation

After Normalization

	Income	Age
Peter	0.125	0.30
Mary	0.3125	0.35
John	0.30	0.80

$$d(Peter, Mary) = \sqrt{(0.125 - 0.3125)^2 + (0.30 - 0.35)^2} \approx 0.19$$

$$d(Peter, John) = \sqrt{(0.125 - 0.30)^2 + (0.30 - 0.80)^2} \approx 0.53$$

- After normalization, Peter is closer to Mary than John.

Binary Attributes

- A contingency table for binary data

		Object j		
		1	0	sum
Object i	1	a	b	$a+b$
	0	c	d	$c+d$
	sum	$a+c$	$b+d$	p

- Each cell represents the **number of binary attributes** that Object *i* is 0 or 1 and Object *j* is 0 or 1.

Binary Attributes

- A contingency table for binary data

		Object *j*		
		1	0	*sum*
	1	a	b	$a+b$
Object *i*	0	c	d	$c+d$
	sum	$a+c$	$b+d$	p

- A binary attribute is symmetric if both of its states are equally valuable and carry the same weight
  - That is, there is no preference on which outcome should be coded as 0 or 1.
  - One such example could be the attribute gender having the states of male and female.
- A common distance function for symmetric variables is: Simple Matching Coefficient:

$$d(i,j) = \frac{b+c}{a+b+c+d}$$

Dissimilarity between Binary Attributes

- Example

	Gender	Glasses	Have-Car	Student	Have-House
Jack	M	N	N	Y	Y
Mary	F	N	Y	Y	Y

  - Suppose all the attributes are symmetric
  - let the gender value M and F be set to 1 and 0 respectively
  - The contingency table:

		Mary	
		1	0
Jack	1	2	1
	0	1	1

  - The simple matching coefficient:

$$d(Jack, Mary) = \frac{1+1}{2+1+1+1} = \frac{2}{5} = 0.4$$

Asymmetric Binary Attributes

- A contingency table for binary data

		Object *j*		
		1	0	*sum*
	1	a	b	$a+b$
Object *i*	0	c	d	$c+d$
	sum	$a+c$	$b+d$	p

- A binary attribute is asymmetric if the outcomes of the states are not equally important
- For example, the positive and negative outcomes of a disease test. We may code the important outcome, such as positive as 1 and negative as 0.
- The agreement of two 1s (positive match) is considered more significant than that of two 0s (negative match).
- A common distance function for asymmetric attributes is: Jaccard Coefficient:

$$d(i, j) = \frac{b + c}{a + b + c}$$

Asymmetric Binary Attributes

- Example

	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	P	N	N	N	N

  - Assume that *gender* is a *symmetric* attribute
  - The remaining attributes are *asymmetric* binary attributes
  - Let the values Y and P be set to 1, and the value N be set to 0 (distance computed only based on asymmetric attributes)
  - Jaccard distance for *only* asymmetric attributes:

$$d(jack, mary) = \frac{0+1}{2+0+1} = 0.33$$

$$d(jack, jim) = \frac{1+1}{1+1+1} = 0.67$$

$$d(jim, mary) = \frac{1+2}{1+1+2} = 0.75$$

Nominal Attributes

- A generalization of the binary attribute in that it can take more than 2 states, e.g., red, yellow, blue, green
- Method 1: Simple matching
  - *m*: # of matches
  - *p*: total # of attributes

$$d(i, j) = \frac{p - m}{p}$$

- Method 2: use a large number of binary attributes
  - creating a new (asymmetric) binary attribute for each of the *M* nominal states

Nominal Attributes - Example

	Attribute1	Attribute2	Attribute3
obj1	red	red	green
obj2	red	blue	green

- Method 1: Simple matching

$$d(\text{obj1}, \text{obj2}) = \frac{3-2}{3}$$

- Method 2: use a large number of binary variables

create binary variables Attr1-red, Attr1-blue, …… ,

Attr2-red, Attr2-blue, …… ,

Ordinal Attributes

- An ordinal attribute can be discrete or continuous
- Order is important, e.g., rank
- Can be treated like interval-scaled
  - replace x_{if} by their rank $r_{if} \in \{1, \ldots, M_f\}$
  - map the range of each attribute onto *[0, 1]* by replacing *i*-th object in the *f*-th attribute by

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

  - e.g., age: young, middle, senior

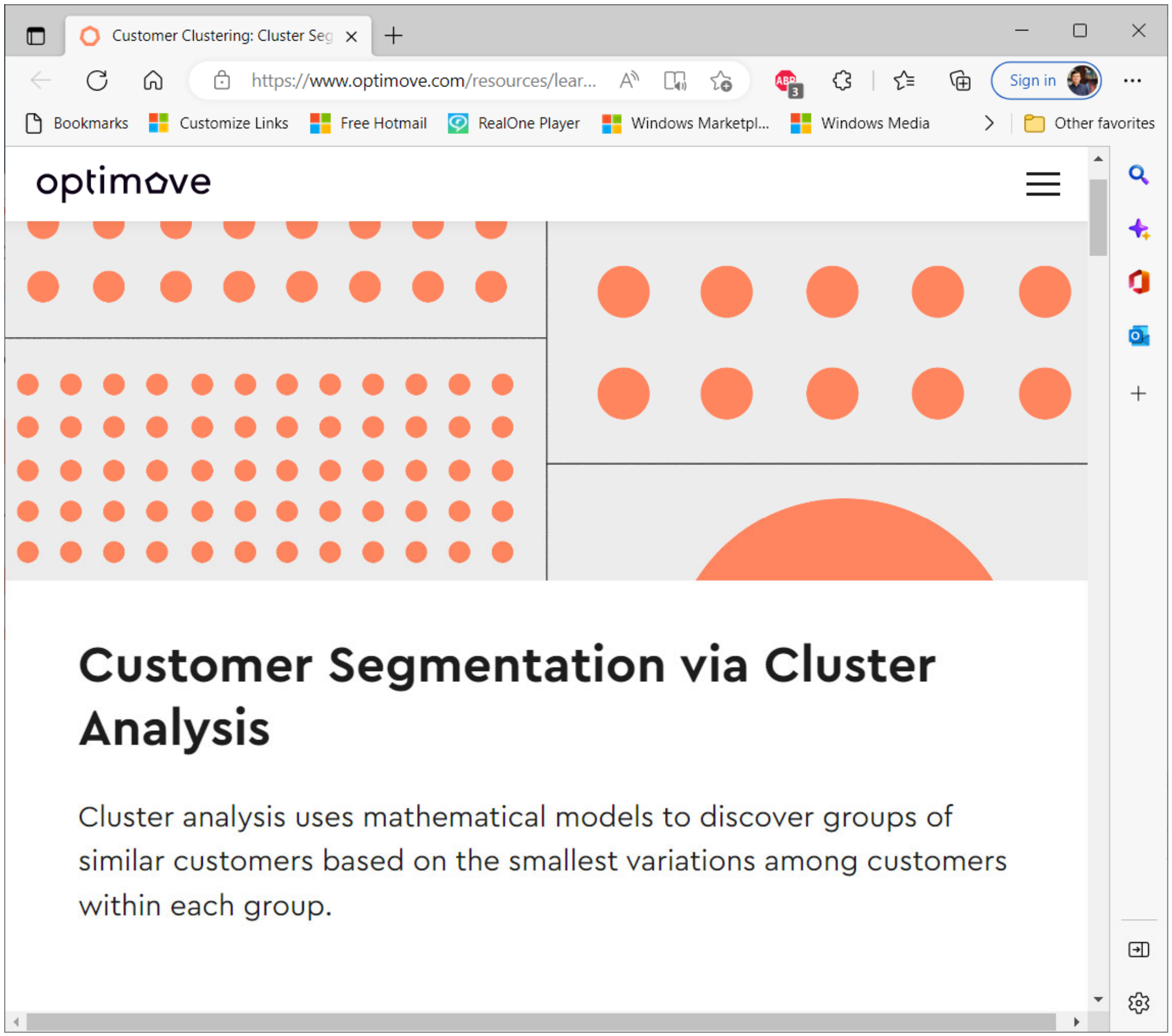
    young maps to 0

    middle maps to (2-1)/(3-1)=1/2
  - compute the dissimilarity using methods for interval-scaled attributes

Clustering Approach: Partitioning Algorithms

- *Partitioning method*: Construct a partition of a database D of n objects into a set of k clusters.
- Given a *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
    - infeasible
  - Popular method: k-means algorithms
  - *k-means*:
    - Each cluster is represented by the center of the cluster called *centroid*.
    - The centroid is computed by taking the average value of each attribute.

Marketing



https://www.optimove.com/resources/learning-center/customer-segmentation-via-cluster-analysis

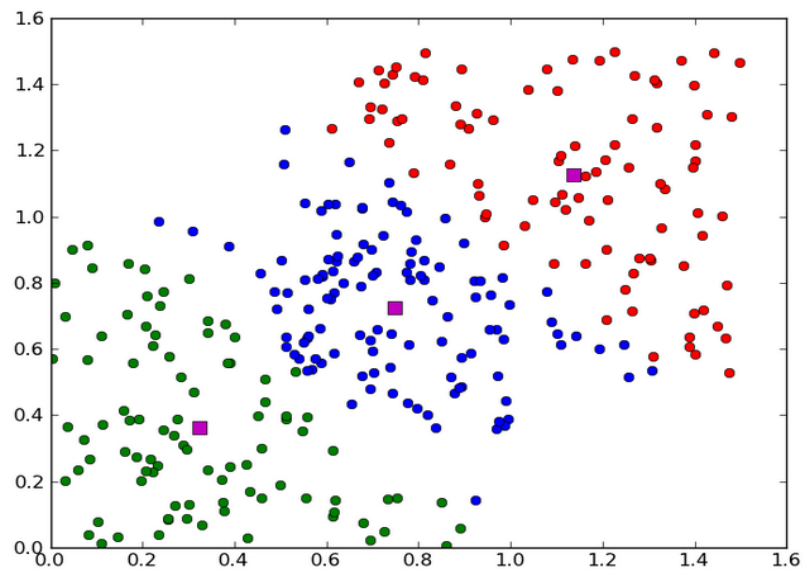
K-means Method

- Given a *K*, find a partition of *K clusters* to optimize the chosen partitioning criterion (objective function)
- Each cluster is represented by the centre of the cluster and the algorithm converges to stable centroids of clusters.

Cluster Centroid

The centroid of a cluster is a point whose coordinates are the mean of the coordinates of all the points in the clusters.

$$\overrightarrow{x_c} = \frac{1}{|c|}\left(\sum_{i=1}^{|c|} \overrightarrow{x_i}\right)$$



Cluster Centroid

Example

	Income	Age
Peter	0.125	0.30
Mary	0.3125	0.35
John	0.30	0.80

Suppose a cluster consists of Peter and Mary.

The centroid of this cluster is:

	Income	Age
centroid	(0.125+0.3125)/2=0.21875	(0.30+0.35)/2=0.325

K-means Method

Given a set of points $(\overrightarrow{x_1}, \overrightarrow{x_2}, \ldots, \overrightarrow{x_n})$

- each point is a d-dimensional real vector
- partition the n observations into k (≤ n) sets S = {S_1, S_2, …, S_k}
- $\overrightarrow{\mu_i}$ is the mean of points in S_i
- The objective function:

$$\min_S \sum_{i=1}^{k} \sum_{x \in S_i} \|\vec{x} - \overrightarrow{\mu_i}\|^2$$

- Minimize the within-cluster sum of squares
- K-means method is a procedure that can find a good solution (but may not be optimal)

K-means Method

Given the cluster number *K*, the *K-means* algorithm is carried out in three steps after initialization:

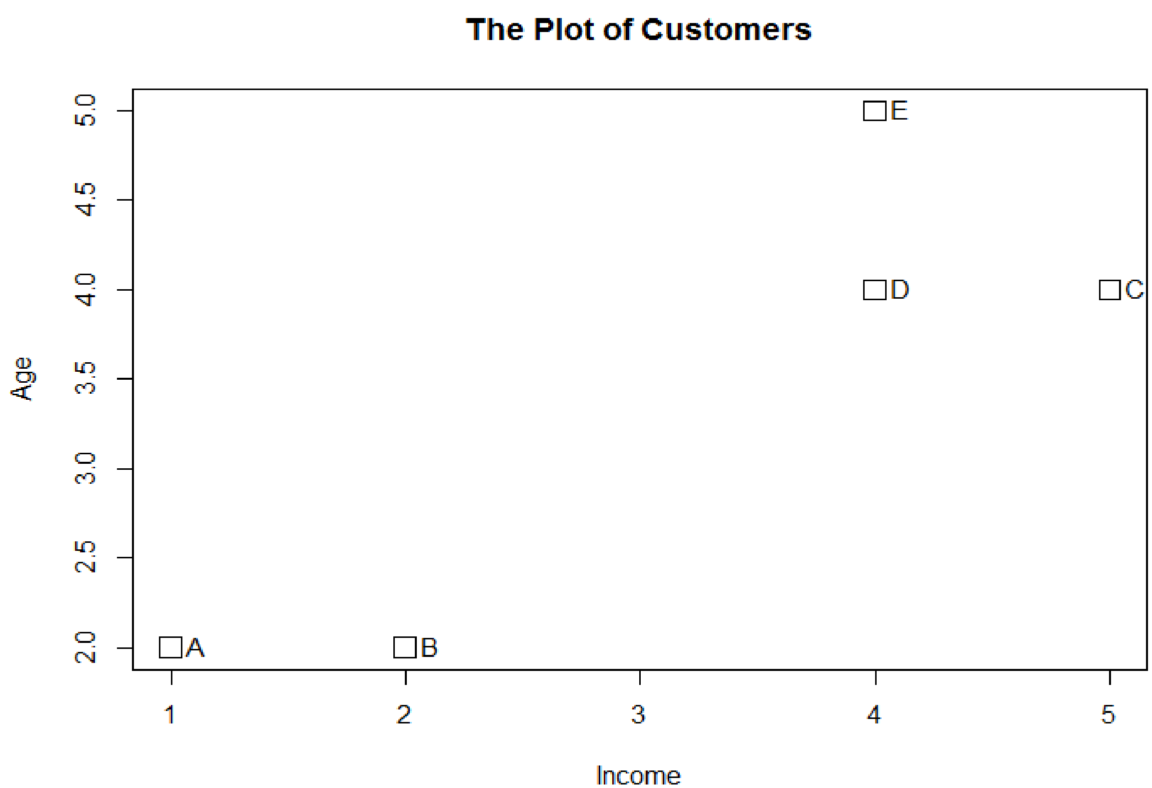
Initialisation: set seed points (randomly)

1) Assign each object to the cluster with the nearest seed point measured with a specific distance metric
2) Compute seed points as the centroids of the clusters of the current partition (the centroid is the centre of the cluster)
3) Go back to Step (1), stop when no more new assignment or membership in each cluster no longer change

Example

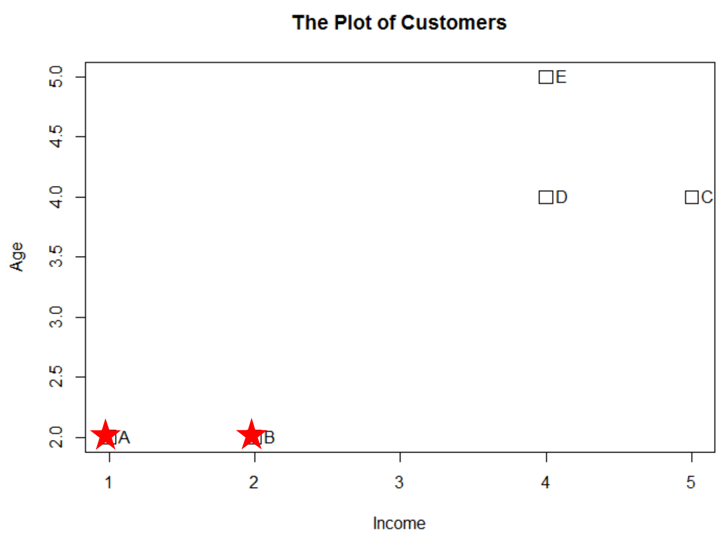
- Suppose we have 5 customers and each has two attributes (standardized income and standardized age). Our goal is to group these customers into *K=2* groups.

Customer	Income	Age
A	1	2
B	2	2
C	5	4
D	4	4
E	4	5



Example

- Step 1: Use existing objects as seed points for partitioning



$$c_1 = A, c_2 = B$$

$$D^0 = \begin{bmatrix} 0 & 1 & 4.5 & 3.6 & 4.2 \\ 1 & 0 & 3.6 & 2.8 & 3.6 \end{bmatrix} \quad \begin{matrix} C_1=(1,2) \\ C_2=(2,2) \end{matrix}$$

$$\begin{matrix} A & B & C & D & E \end{matrix}$$

$$\begin{bmatrix} 1 & 2 & 5 & 4 & 4 \\ 2 & 2 & 4 & 4 & 5 \end{bmatrix} \quad \begin{matrix} \text{Income} \\ \text{Age} \end{matrix}$$

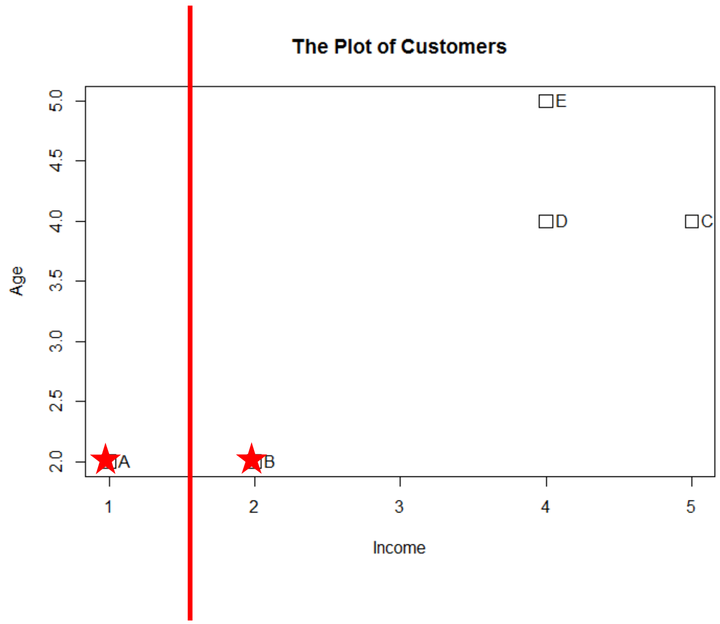
$$d(E, c1) = \sqrt{(4-1)^2 + (5-2)^2} \approx 4.2$$

$$d(E, c2) = \sqrt{(4-2)^2 + (5-2)^2} \approx 3.6$$

Assign each object to the cluster with the nearest seed point

Example

- Step 1: Use existing objects as seed points for partitioning



$$c_1 = A, c_2 = B$$

$$D^0 = \begin{bmatrix} 0 & 1 & 4.5 & 3.6 & 4.2 \\ 1 & 0 & 3.6 & 2.8 & 3.6 \end{bmatrix} \quad \begin{matrix} C_1=(1,2) \\ C_2=(2,2) \end{matrix}$$

A B C D E

$$\begin{bmatrix} 1 & 2 & 5 & 4 & 4 \\ 2 & 2 & 4 & 4 & 5 \end{bmatrix} \quad \begin{matrix} \text{Income} \\ \text{Age} \end{matrix}$$

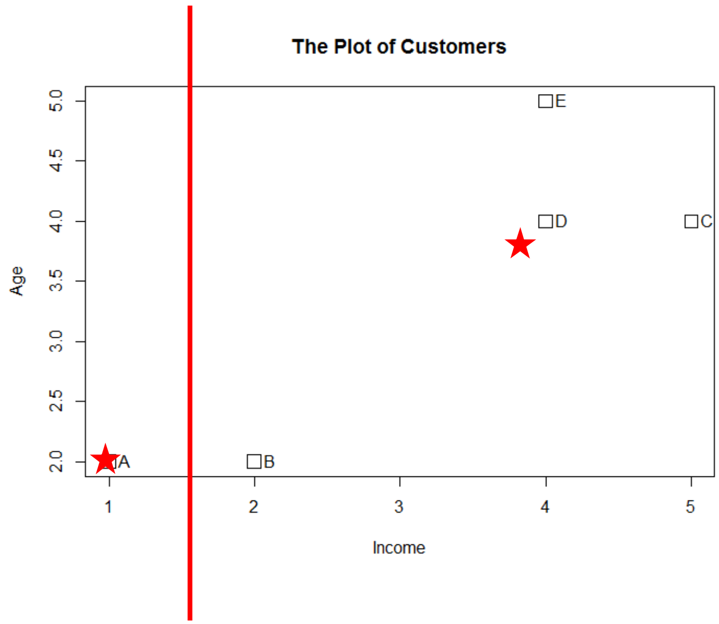
$$d(E, c1) = \sqrt{(4-1)^2 + (5-2)^2} \approx 4.2$$

$$d(E, c2) = \sqrt{(4-2)^2 + (5-2)^2} \approx 3.6$$

Assign each object to the cluster with the nearest seed point

Example

- Step 2: Compute new centroids of the current partition



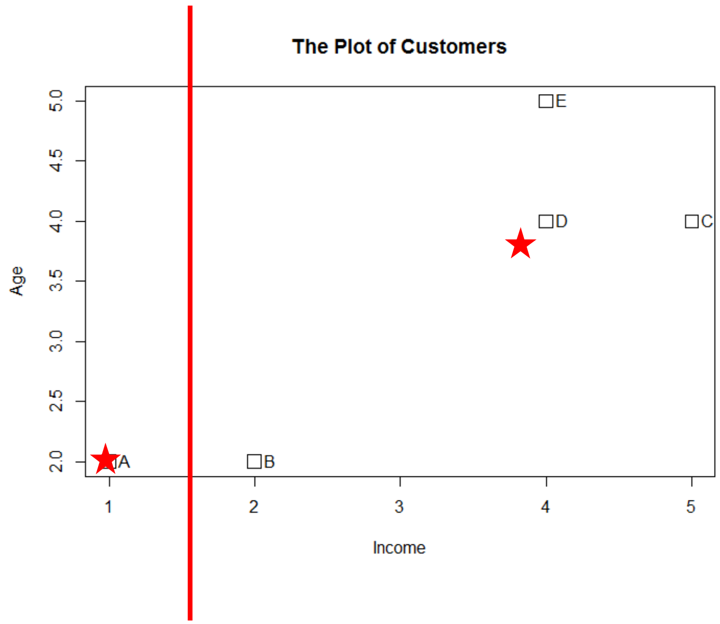
Knowing the members of each cluster, now we compute the new centroid of each group based on these new memberships.

$$c_1 = (1,2)$$

$$c_2 = \left(\frac{2+5+4+4}{4}, \frac{2+4+4+5}{4}\right) = (3.75, 3.75)$$

Example

- Step 2: Renew membership based on new centroids



Compute the distance of all objects to the new centroids

$$D^1 = \begin{bmatrix} 0 & 1 & 4.5 & 3.6 & 4.2 \\ 3.3 & 2.5 & 1.3 & 0.4 & 1.3 \end{bmatrix} \begin{matrix} C_1=(1,2) \\ C_2=(3.75,3.75) \end{matrix}$$

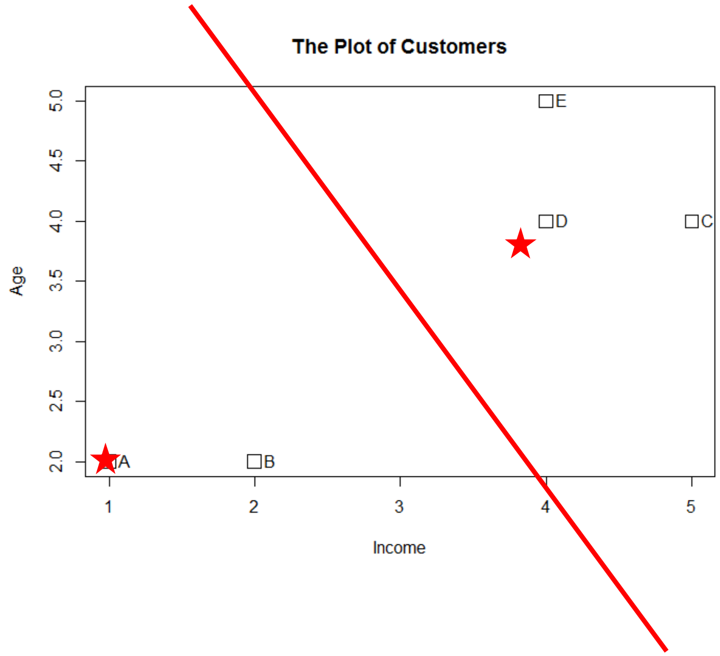
$$\begin{matrix} A & B & C & D & E \\ \end{matrix}$$

$$\begin{bmatrix} 1 & 2 & 5 & 4 & 4 \\ 2 & 2 & 4 & 4 & 5 \end{bmatrix} \begin{matrix} \text{Age} \\ \text{Income} \end{matrix}$$

Assign the membership to objects

Example

- Step 2: Renew membership based on new centroids



Compute the distance of all objects to the new centroids

$$D^1 = \begin{bmatrix} 0 & 1 & 4.5 & 3.6 & 4.2 \\ 3.3 & 2.5 & 1.3 & 0.4 & 1.3 \end{bmatrix} \begin{matrix} C_1=(1,2) \\ C_2=(3.75,3.75) \end{matrix}$$

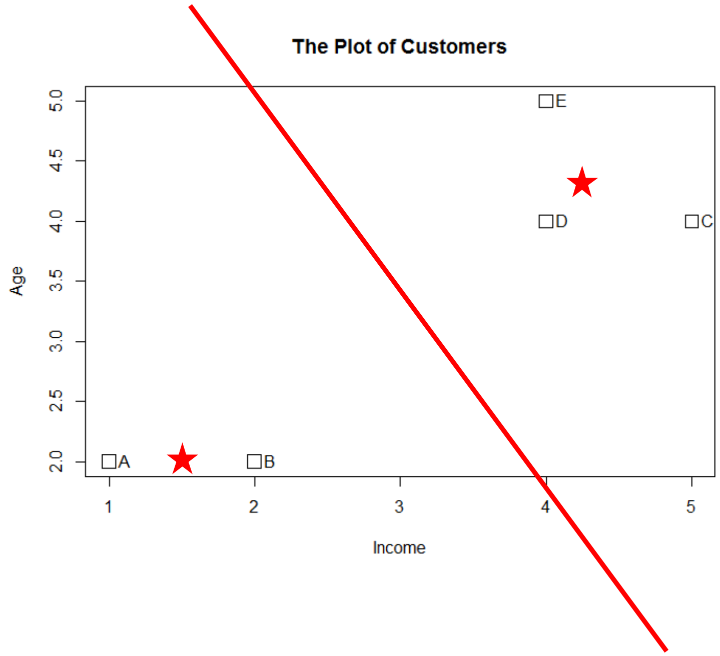
$$\begin{matrix} A & B & C & D & E \\ \end{matrix}$$

$$\begin{bmatrix} 1 & 2 & 5 & 4 & 4 \\ 2 & 2 & 4 & 4 & 5 \end{bmatrix} \begin{matrix} \text{Age} \\ \text{Income} \end{matrix}$$

Assign the membership to objects

Example

- Step 3: Repeat the first two steps until its convergence



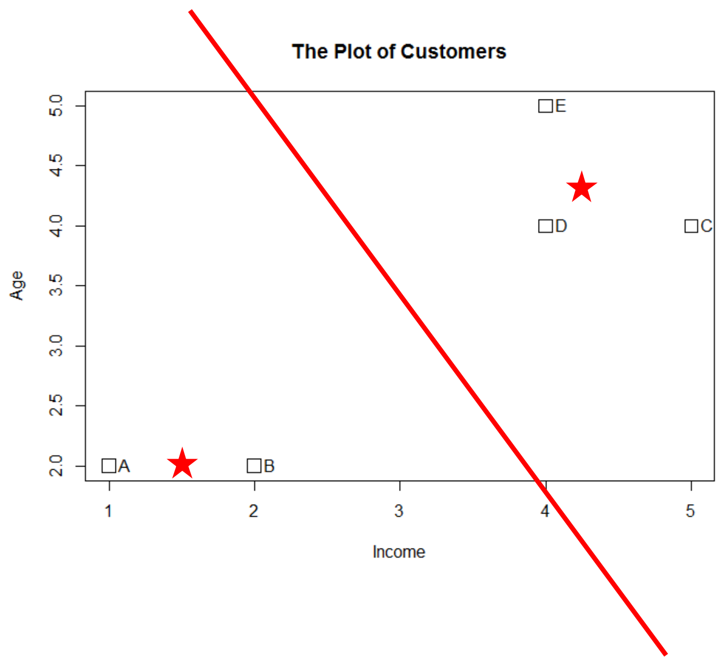
Knowing the members of each cluster, now we compute the new centroid of each group based on these new memberships.

$$c_1 = \left(\frac{1+2}{2}, \frac{2+2}{2}\right) = (1.5,2)$$

$$c_2 = \left(\frac{5+4+4}{3}, \frac{4+4+5}{3}\right) = (4.3,4.3)$$

Example

- Step 3: Repeat the first two steps until its convergence



Compute the distance of all objects to the new centroids

$$D^2 = \begin{bmatrix} 0.5 & 0.5 & 4.0 & 3.2 & 3.9 \\ 4.1 & 3.3 & 0.75 & 0.47 & 0.75 \end{bmatrix} \begin{matrix} C_1=(1.5,2) \\ C_2=(4.3,4.3) \end{matrix}$$

$$\begin{matrix} A & B & C & D & E \\ \end{matrix}$$

$$\begin{bmatrix} 1 & 2 & 5 & 4 & 4 \\ 2 & 2 & 4 & 4 & 5 \end{bmatrix} \begin{matrix} \text{Age} \\ \text{Income} \end{matrix}$$

Stop due to no new assignment
Membership in each cluster no longer change

Convergence and Termination

- Each iterative step necessarily lowers the sum of the distance
- Always converge
- None of the objects changed membership in the last iteration

The K-Means Clustering Method

Algorithm: ***k*-means.** The *k*-means algorithm for partitioning based on the mean value of the objects in the cluster.

Input: The number of clusters *k* and a database containing *n* objects.

Output: A set of *k* clusters that minimizes the squared-error criterion.

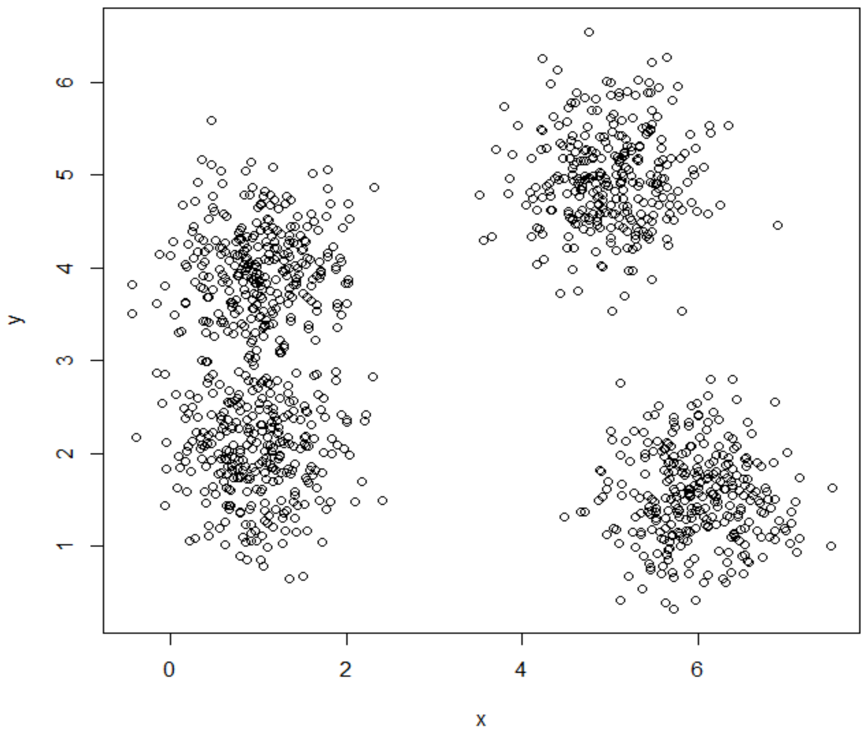
Method:

(1) arbitrarily choose *k* objects as the initial cluster centers;
(2) repeat
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) update the cluster means, i.e., calculate the mean value of the objects for each cluster;
(5) until no change;

How K-means partitions?

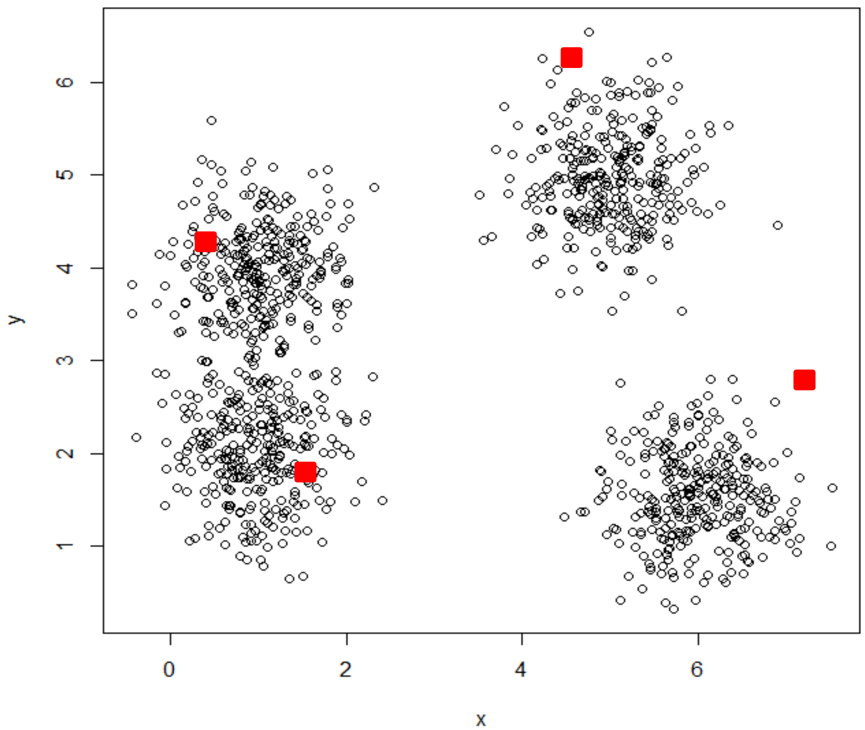
- *K* centroids are set/fixed
- The centroids partition the whole data space into *K* mutually exclusive subspaces to form a partition.
- A partition amounts to a Voronoi Diagram.
- Changing positions of centroids leads to a new partitioning.

K-means Method Visualization



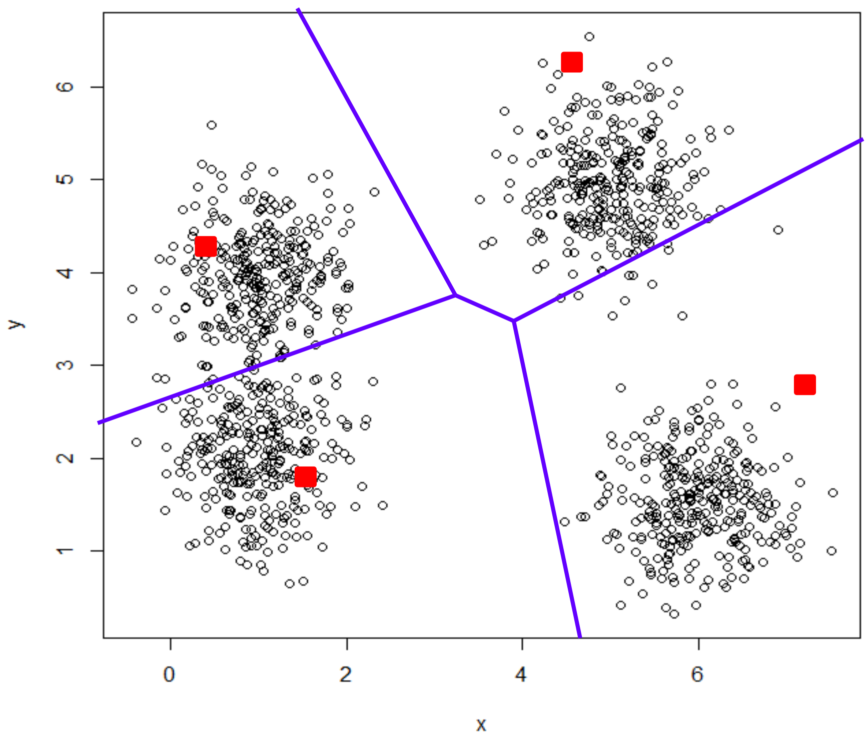
1. User set up the number of clusters they'd like. *(e.g. k=4)*

K-means Method Visualization



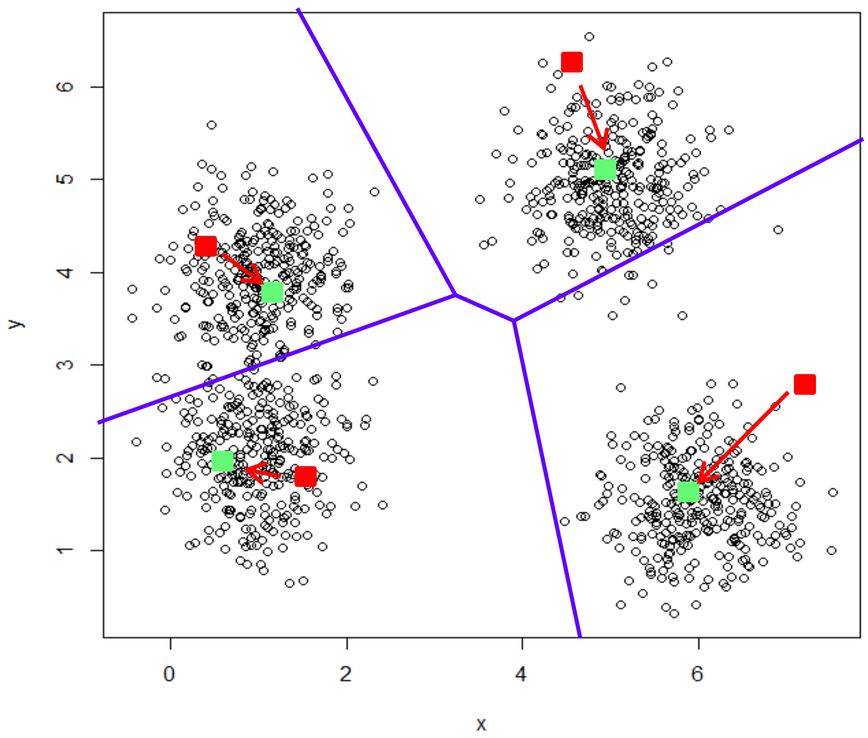
1. User set up the number of clusters they'd like. *(e.g. K=4)*
2. Randomly guess K cluster Center locations

K-means Method Visualization



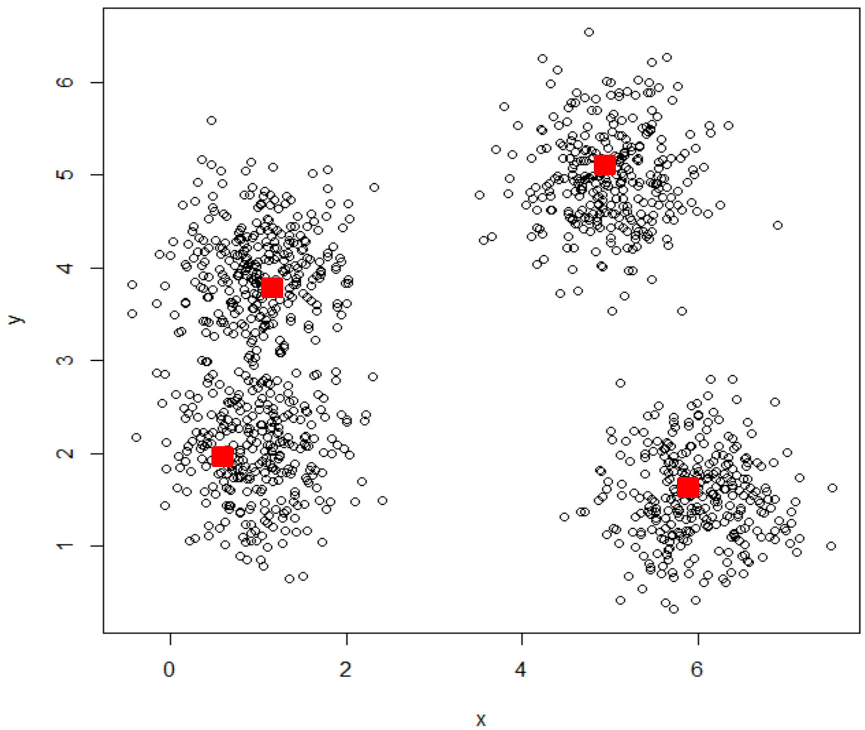
1. User set up the number of clusters they'd like. *(e.g. K=4)*
2. Randomly guess *K* cluster Center locations
3. Each data point finds out which Center it's closest to. (Thus each Center "owns" a set of data points)

K-means Method Visualization



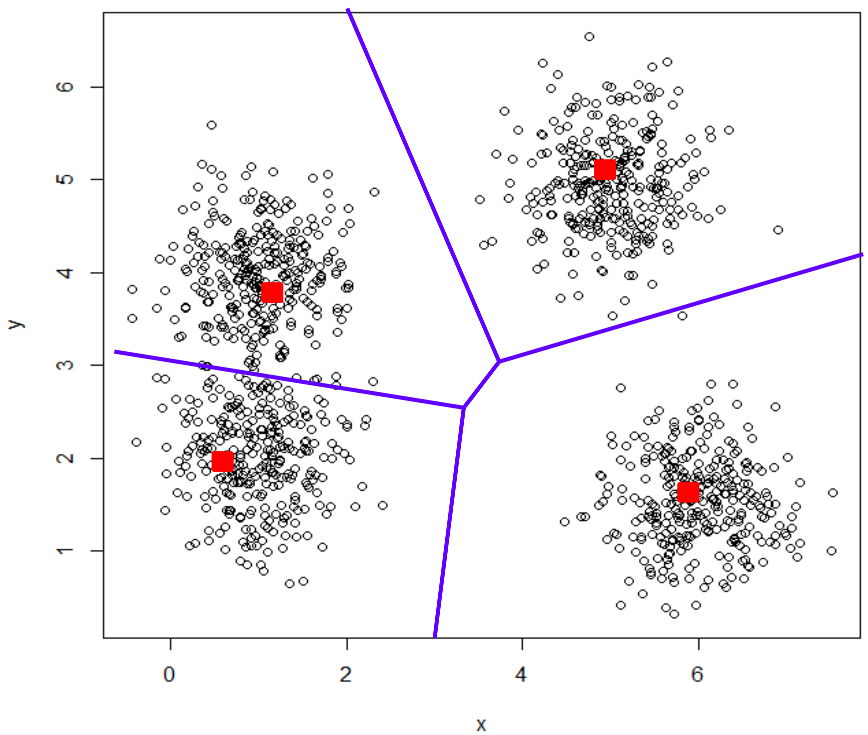
1. User set up the number of clusters they'd like. *(e.g. K=4)*
2. Randomly guess *K* cluster centre locations
3. Each data point finds out which centre it's closest to. (Thus each Center "owns" a set of data points)
4. Each centre finds the centroid of the points it owns

K-means Method Visualization



1. User set up the number of clusters they'd like. (e.g. *K=4*)
2. Randomly guess *K* cluster centre locations
3. Each data point finds out which centre it's closest to. (Thus each centre "owns" a set of data points)
4. Each centre finds the centroid of the points it owns
5. …and jumps there

K-means Method Visualization



1. User set up the number of clusters they'd like. (e.g. *K=5*)
2. Randomly guess *K* cluster centre locations
3. Each data point finds out which centre it's closest to. (Thus each centre "owns" a set of data points)
4. Each centre finds the centroid of the points it owns
5. …and jumps there
6. …Repeat until terminated!

K-means Method – Some Issues

- Efficient in computation
- Local optimum
  - sensitive to initial seed points
  - converge to a local optimum
- Other problems
  - Need to specify *K,* the *number* of clusters, in advance
  - Unable to handle noisy data and outliers (*K-Medoids* algorithm)
  - Not suitable for discovering clusters with non-convex shapes

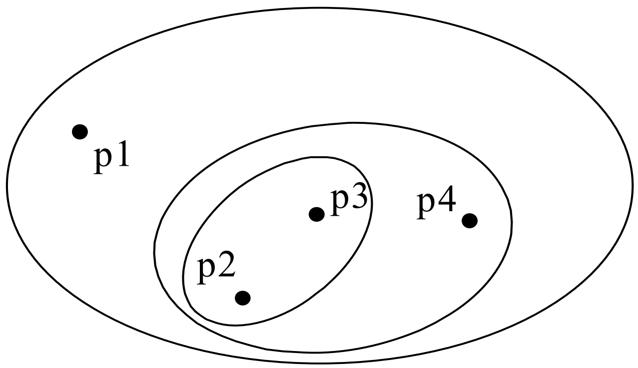
K-means Method – Some Issues

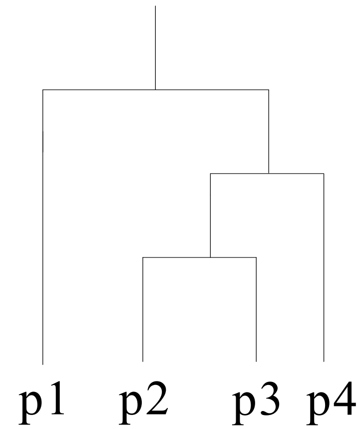
- *K*-means algorithm is a simple yet popular method for clustering analysis
- Its performance is determined by initialization and appropriate distance measure
- There are several variants of *K*-means to overcome its weaknesses
  - *K*-Medoids: resistance to noise and/or outliers
  - CLARA: extension to deal with large data sets
  - Mixture models: handling uncertainty of clusters

<u>Types of Clusterings</u>

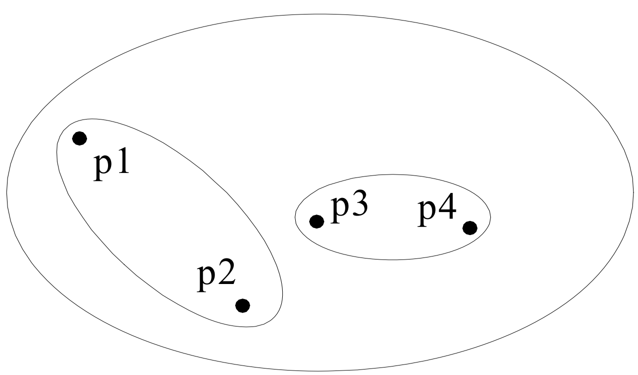
- Distinction between hierarchical and partitional sets of clusters
- Partitional Clustering
  - A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- Hierarchical clustering
  - A set of nested clusters organized as a hierarchical tree

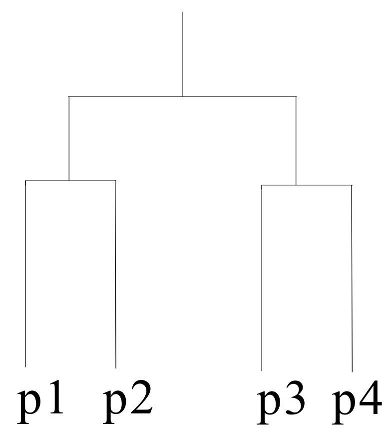
Hierarchical Clustering





Dendrogram 1





Dendrogram 2

Hierarchical Clustering

Outline of an Approach

- Bottom-up strategy
- Placing each object in its own cluster
- Merges these atomic clusters into larger and larger clusters
- Until all of the objects are in a single cluster.

Hierarchical Clustering

- Example: A data-set has five objects {a,b,c,d,e}