SUNG KYUN KWAN UNIVERSITY

Non-Hierarchical Clustering (Partitioning Method)

Jong-Seok Lee

Sungkyunkwan University

Partitioning Method

- Predefine the number of clusters (*k*).
- Divide data objects into non-overlapping subgroups (clusters) such that each object is in exactly one cluster.
- Initial bad clustering can be recovered later.
- Useful for a large dataset

- Often-used Methods
  - k-means clustering
  - k-medoids clustering
    - PAM (Partitioning Around Medoids)
  - Model-based clustering (Mixture of Gaussian)
  - Grid-based clustering
    - SOM (Self-Organizing Map)

k-means Clustering

- A partitional clustering approach
- Each cluster is associated with a centroid (center point; mean)
- Each object is assigned to the cluster with the closest centroid.
- The number of clusters is predefined.
- Its objective is to minimize the compactness (SSW) of clusters.

k-means Clustering

- Mathematical formulation

Minimize $\quad Z = \sum_{i=1}^{n} \sum_{j=1}^{k} d\left(\mathbf{x}_i, \mathbf{c}_j\right) \cdot a_{ij}$

subject to

$$\mathbf{c}_j = \frac{\sum_{i=1}^{n} \mathbf{x}_i a_{ij}}{\sum_{i=1}^{n} a_{ij}} \;, \quad j = 1, 2, \ldots, k$$

$$\sum_{j=1}^{k} a_{ij} = 1 \;, \quad i = 1, 2, \ldots, n$$

$$\sum_{i=1}^{n} a_{ij} \geq 1 \;, \quad j = 1, 2, \ldots, k$$

$$a_{ij} = 1 \text{ or } 0 \;, \quad i = 1, 2, \ldots, n; \; j = 1, 2, \ldots, k$$

$$a_{ij} = \begin{cases} 1, & \text{if } i\text{th object belongs to cluster } j \\ 0, & \text{otherwise} \end{cases}$$

- Find clusters (a_{ij}'s) that minimize the objective function.
- Enumerating all possible (feasible) solutions is impossible in practice. (NP-Hard problem)
- We can have a near-optimal solution by employing a heuristic algorithm.

k-means Clustering

- Algorithm
- Step 0
  - Select *k* objects as initial centroids.
- Step 1 (Assignment)
  - For each object, compute the distances to *k* centroids.
  - Assign each object to the cluster to which it is the closest.
- Step 2 (New Centroids)
  - Compute new centroids for each cluster.
- Step 3 (Convergence)
  - Stop if the selected stopping criterion is satisfied.
  - Otherwise, go to Step 1.

k-means Clustering

- Stopping criteria
  - No change in centroids
    - If the newly updated centroids are the same with the previous ones, then stop.
  - Objective function value
    - If the objective function value is less than a predefined value, then stop.
  - Number of iterations
    - Repeat Step 1 and Step 2 m times, where m is a predefined number of iterations.

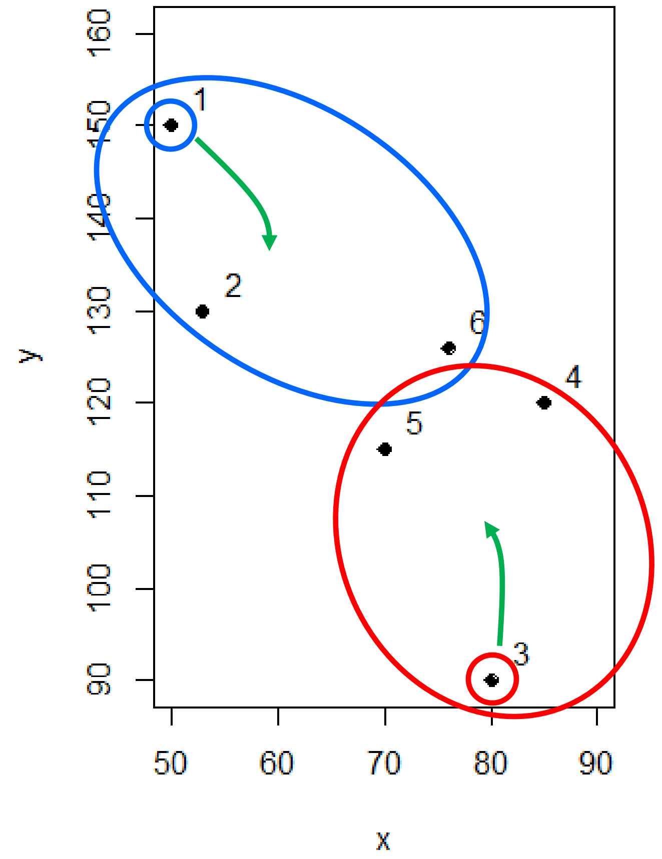
k-means Clustering

- Example
  - k=2

Initial seeds

object	X_1	X_2
1	50	150
2	53	130
3	80	90
4	85	120
5	70	115
6	76	126

object	*1*	*3*
1	0	67.1
2	20.2	48.3
3	67.1	0
4	46.1	30.4
5	40.3	26.9
6	35.4	36.2

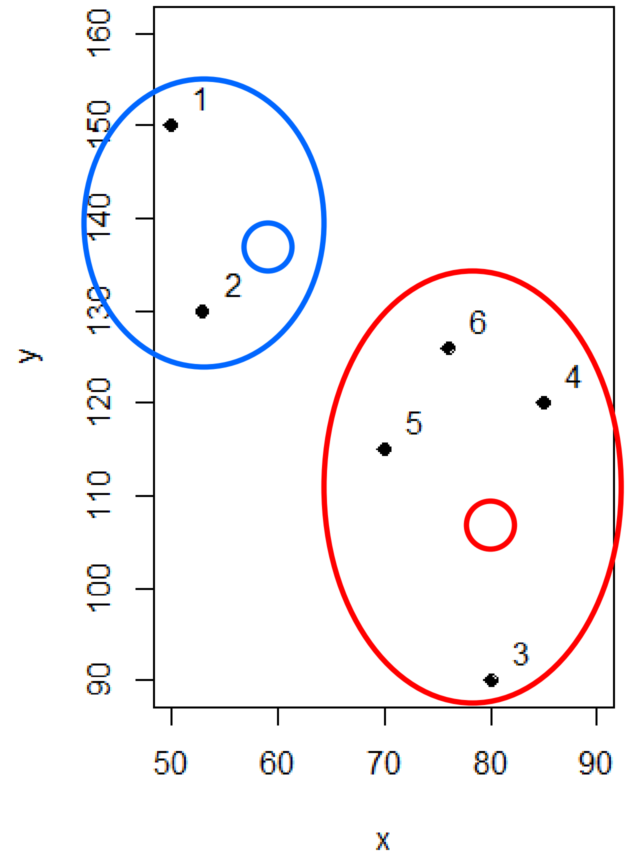


k-means Clustering

- Example (cont'd)

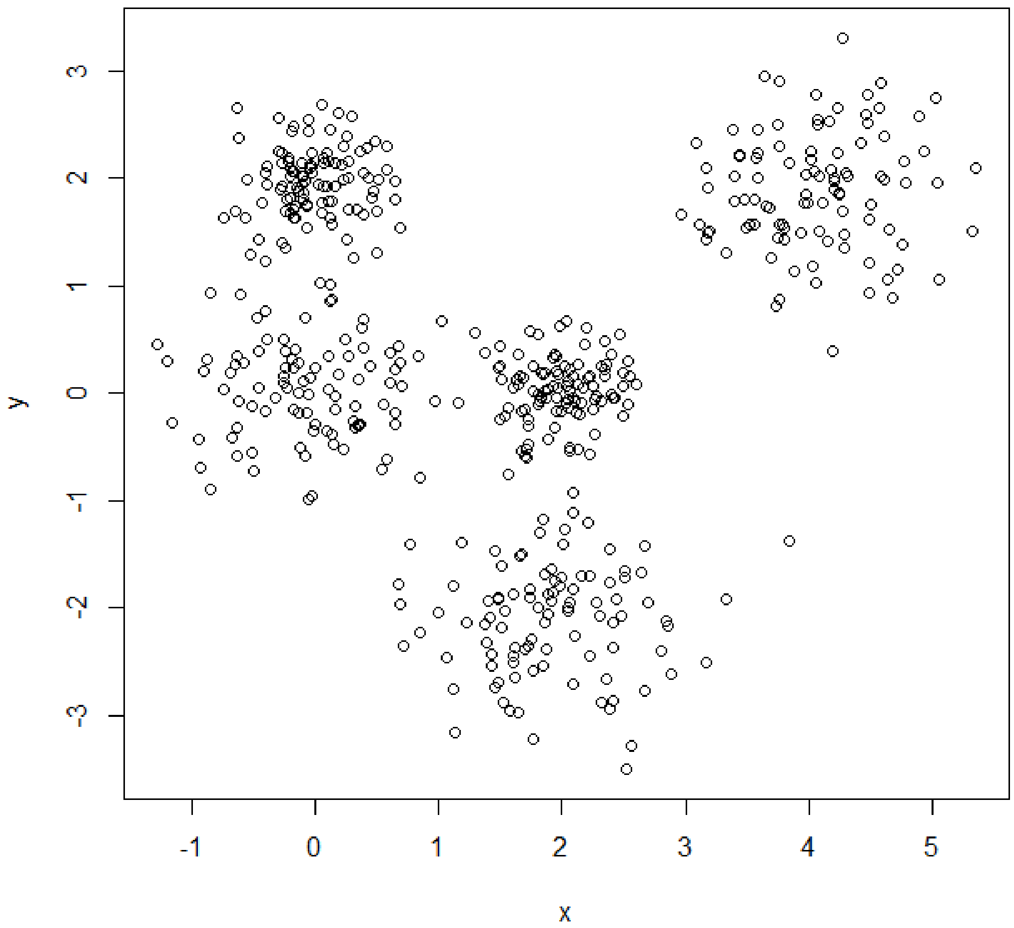
	Cluster 1	Cluster 2
Objects	O_1, O_2, O_6	O_3, O_4, O_5
Coordinates	(59.6, 135.3)	(78.3, 108.3)

object	c_1	c_2
1	17.56	50.40
2	8.46	33.33
3	49.63	18.38
4	29.65	13.48
5	22.81	10.67
6	18.85	17.85



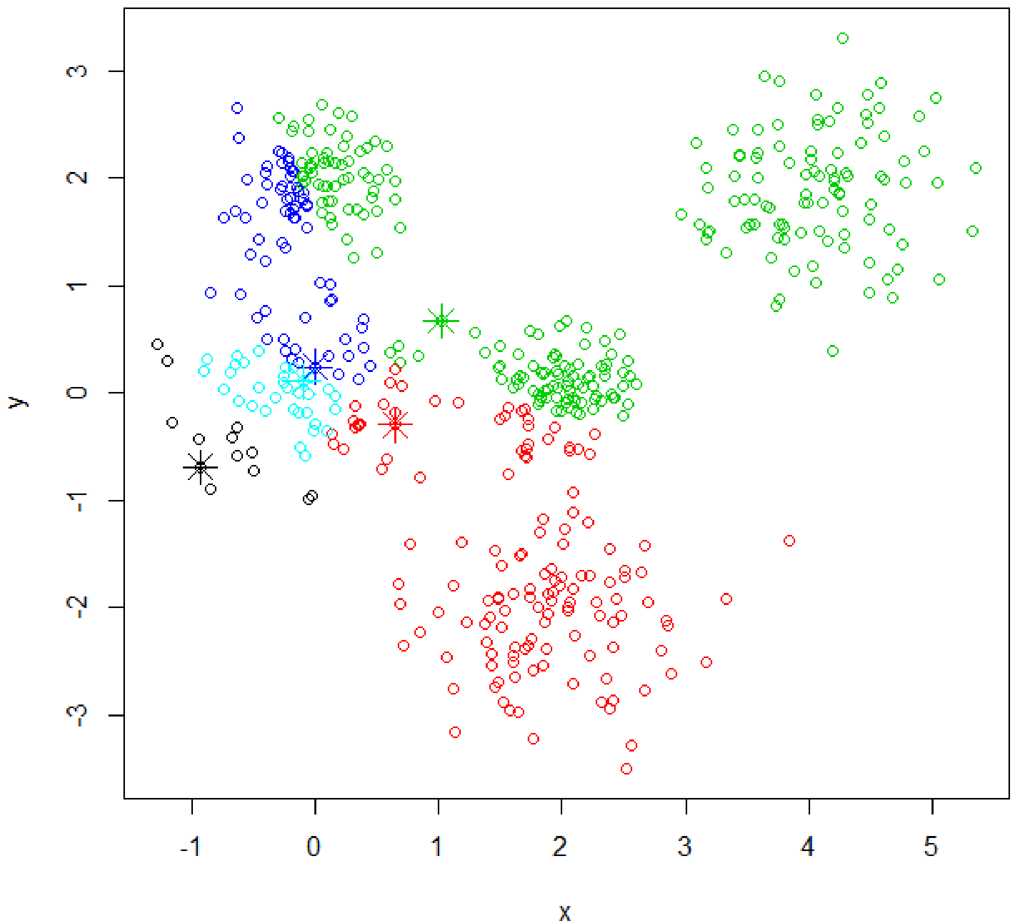
k-means Clustering

- Demo (k=5)



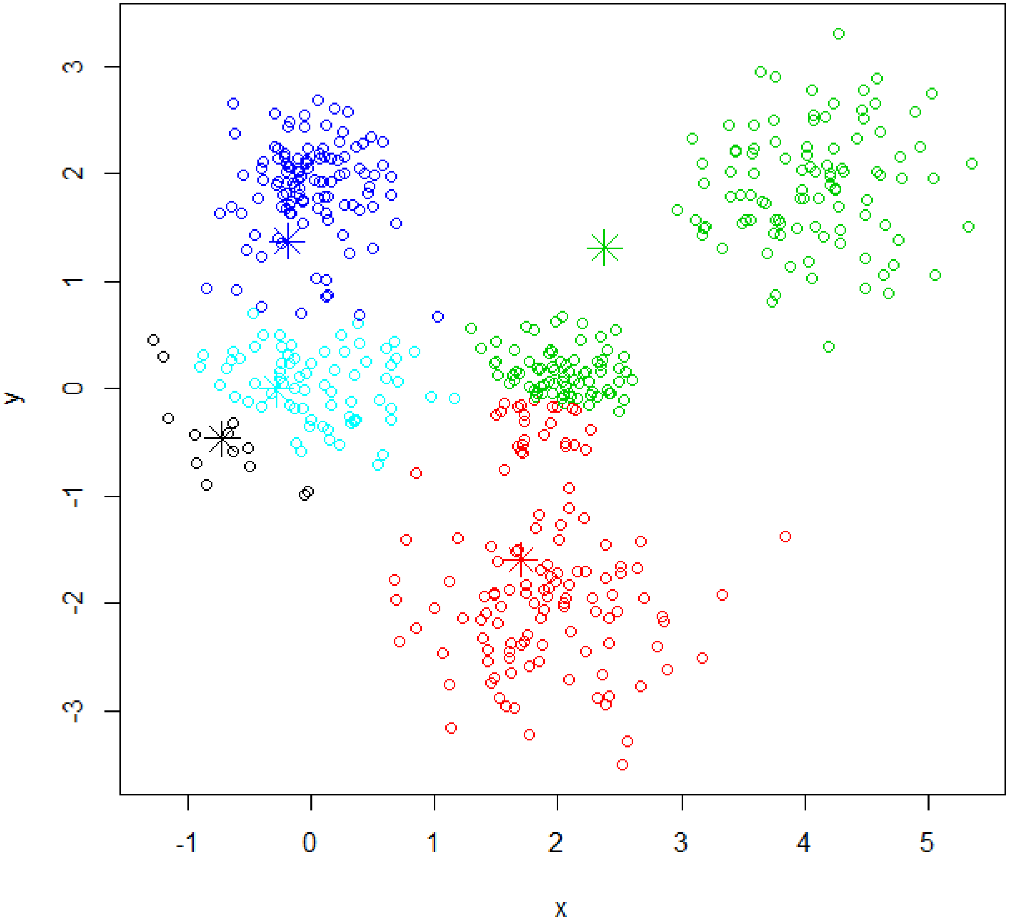
k-means Clustering

- Demo (Initial clusters)



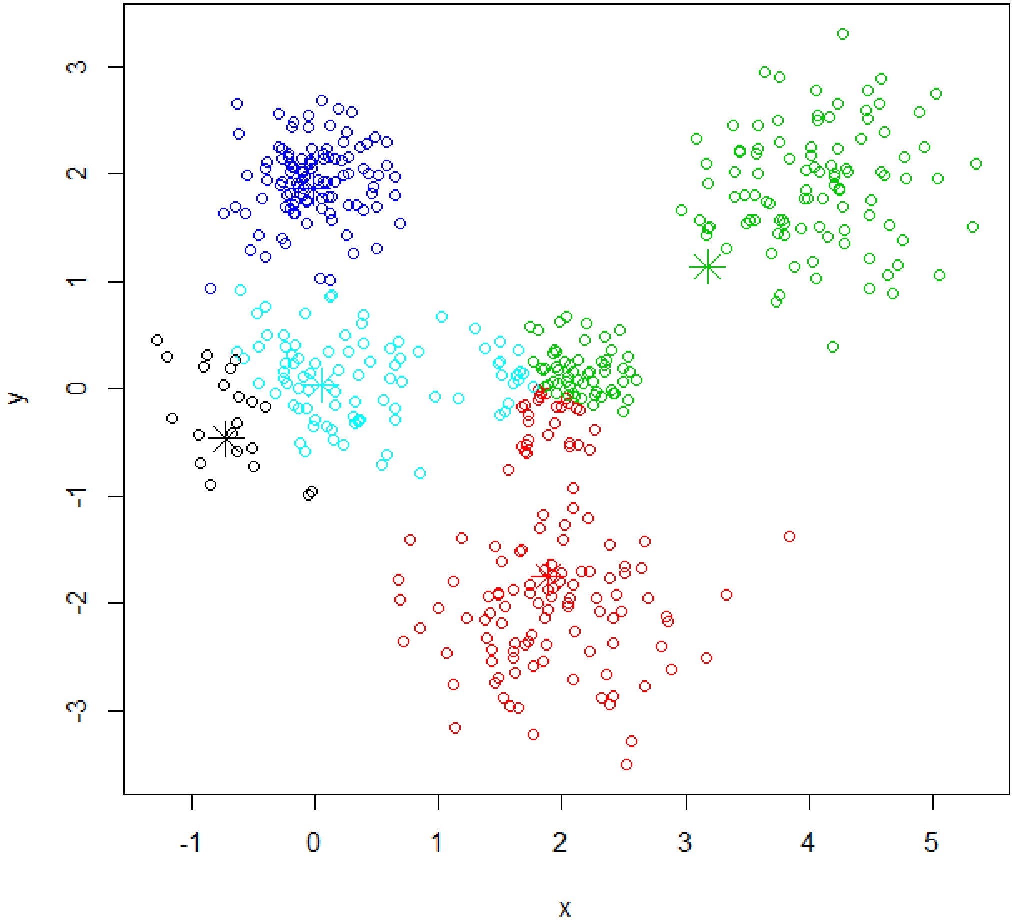
k-means Clustering

- Demo (Iteration 1)



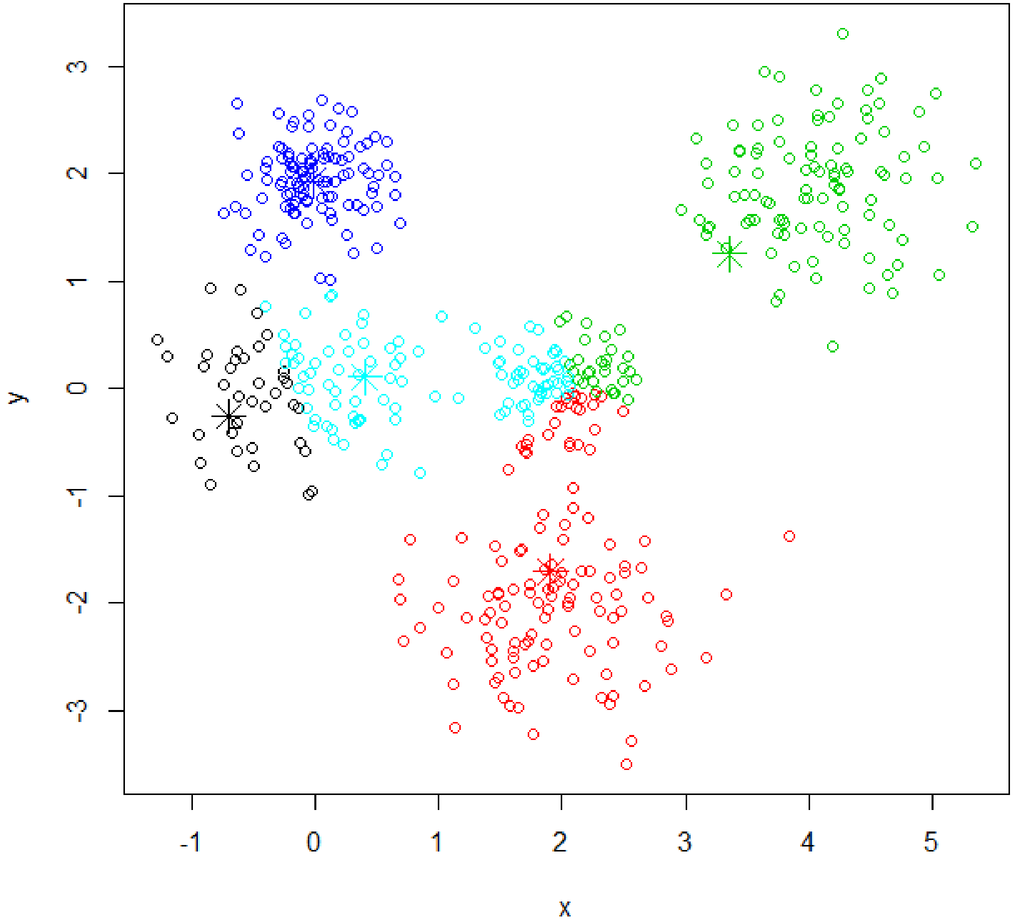
k-means Clustering

- Demo (Iteration 2)



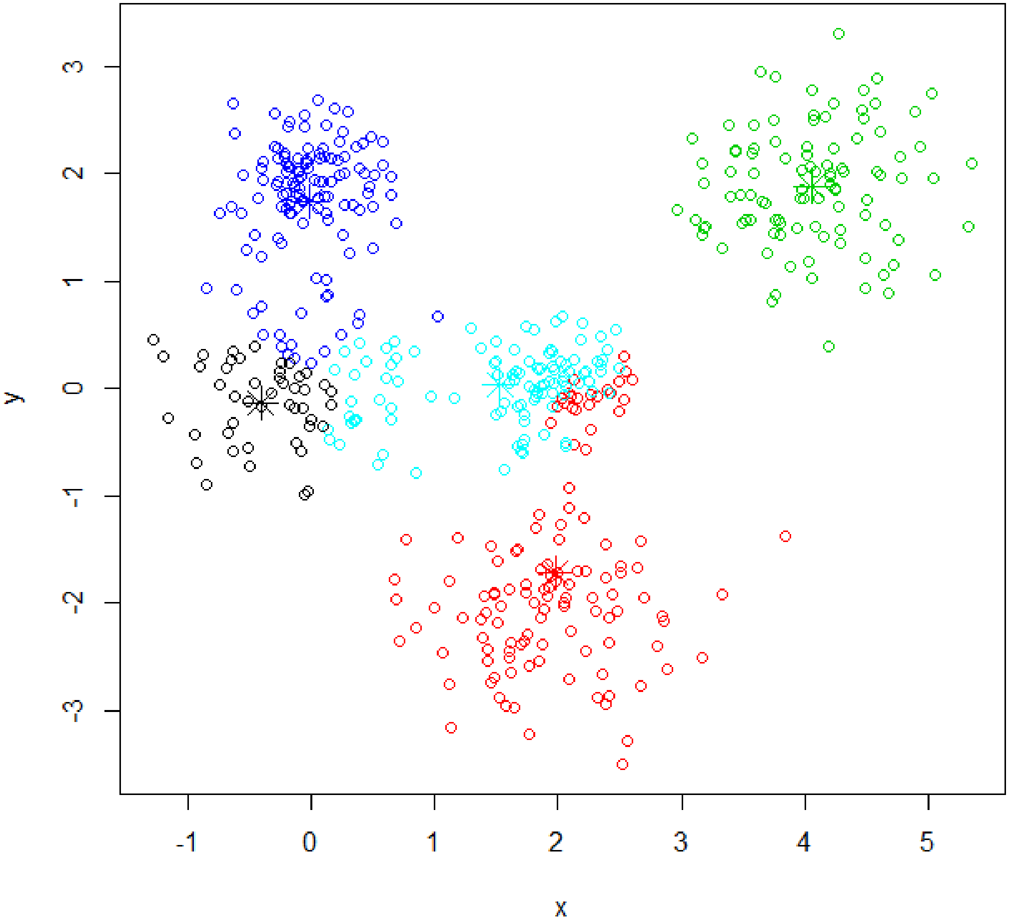
k-means Clustering

- Demo (Iteration 3)



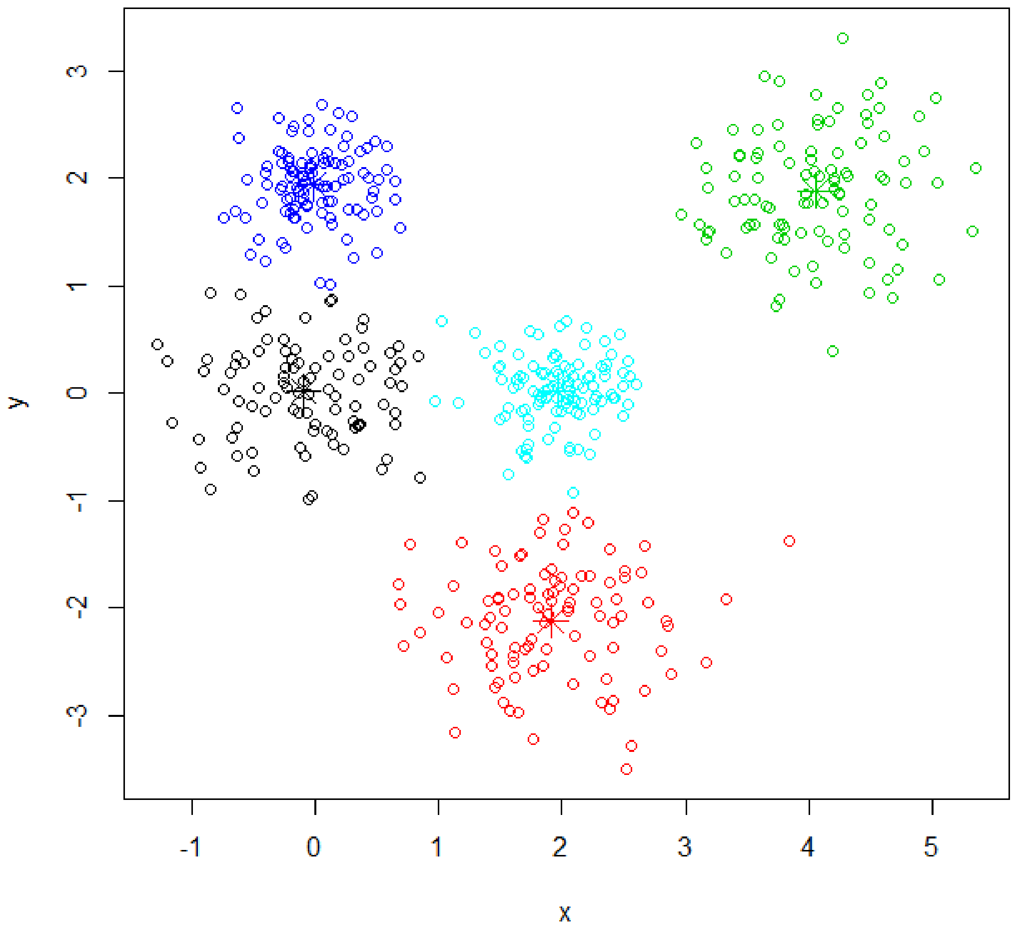
k-means Clustering

- Demo (Iteration 4)



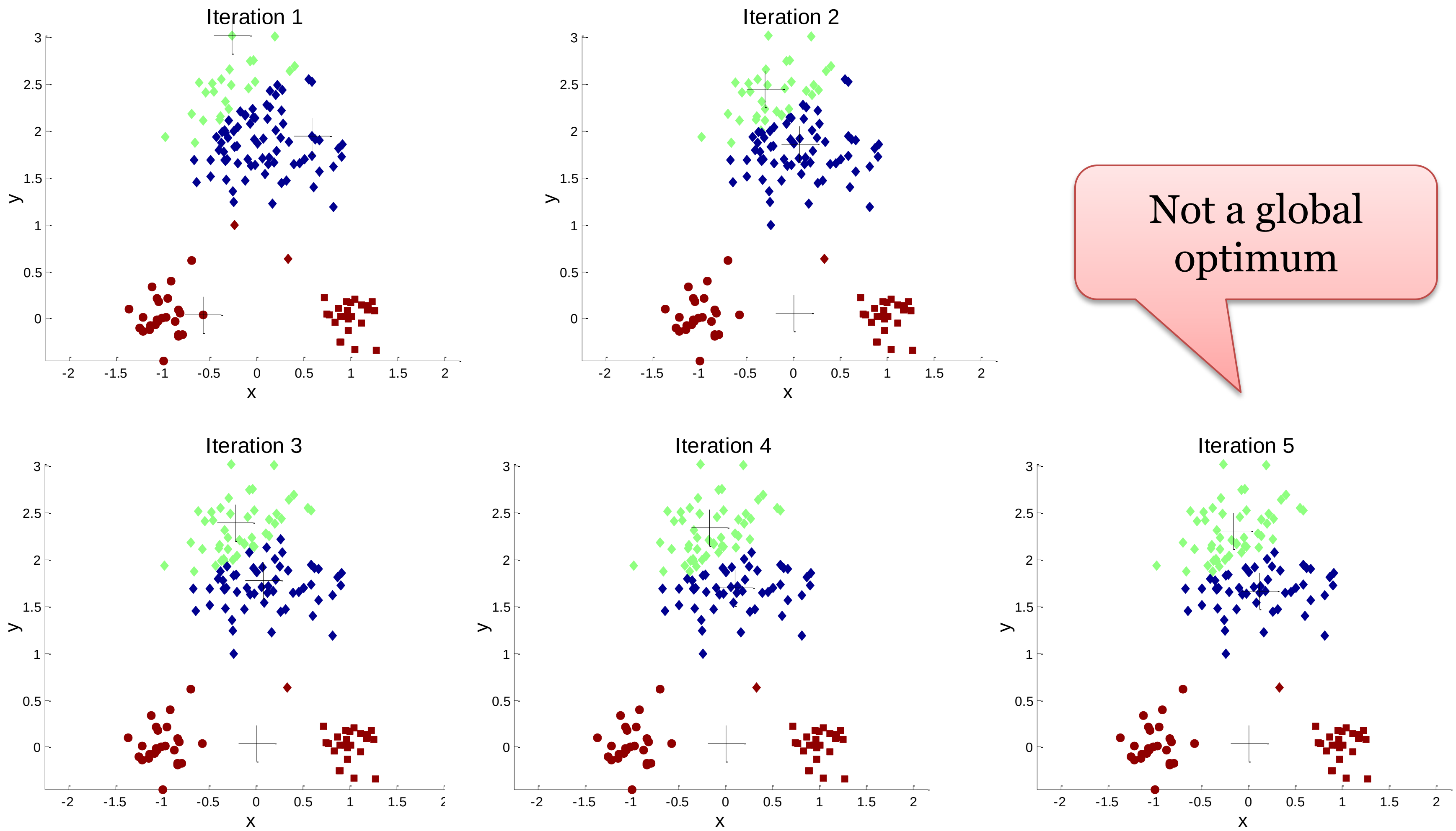
k-means Clustering

- Demo (Iteration 5)



k-means Clustering

- Importance of choosing initial centroids

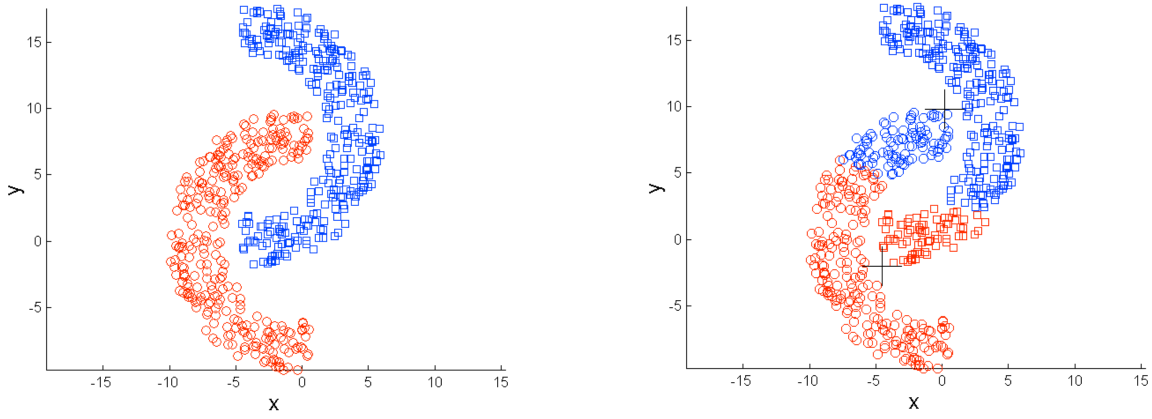


Not a global optimum

k-means Clustering

- Solutions to the initial centroids problem
  - Multiple runs with randomly chosen centroids
  - Sample objects and use hierarchical clustering to determine initial centroids.
  - Select most widely separated (farthest) objects.

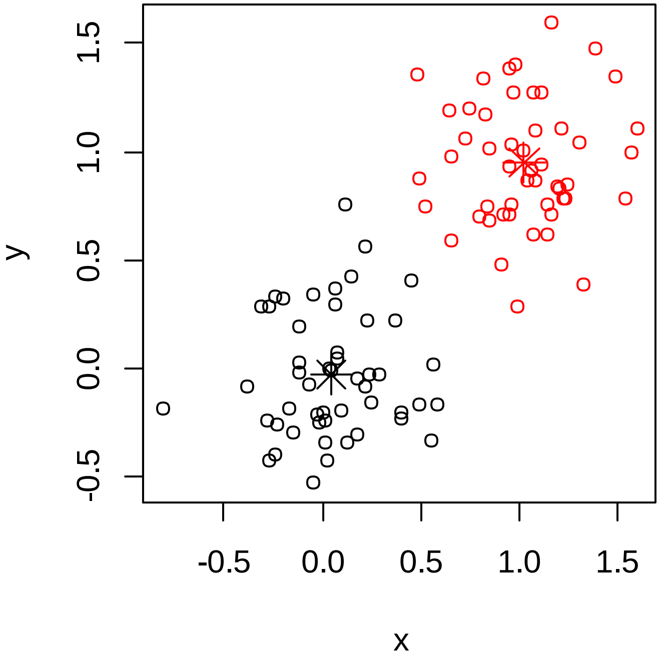
- Limitations of k-means clustering
  - Not working for irregular shaped clusters



R Example

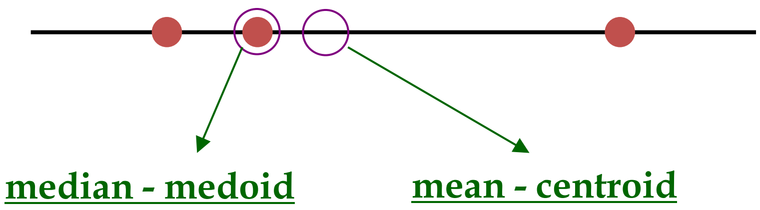
- k-means clustering

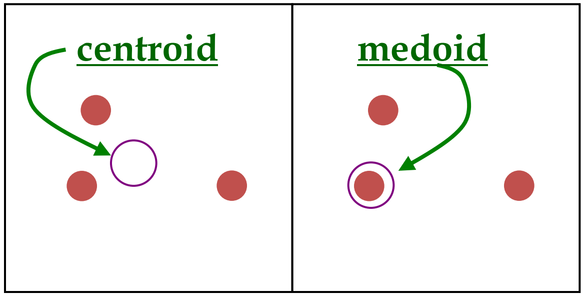
```
> # a 2-dimensional example
> x <- rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2))
> colnames(x) <- c("x", "y")
> cl <- kmeans(x, 2)
> plot(x, col = cl$cluster)
> points(cl$centers, col = 1:2, pch = 8, cex=2)

> kmeans(x,1)$withinss
[1] 62.9905

> kmeans(dt.cars, 2, iter.max=10, nstart=3)
> # see the results
```



k-medoids Clustering

- A medoid is less influenced by outliers or other extreme values than a centroid.
- Centroid vs. medoid





- A medoid can be defined as the object of a cluster, whose average distance to all objects in the cluster is minimal.
- After finding a set of medoids, each object in data is assigned to the closest medoid.
- The *k* medoids should minimize the objective function, which is the sum of the distances of all objects to their nearest medoid.

k-medoids Clustering

- Mathematical formulation

Minimize $$Z = \sum_{i=1}^{n} \sum_{j=1}^{n} d\left(\mathbf{x}_i, \mathbf{x}_j\right) \cdot a_{ij}$$

subject to

$$\sum_{j=1}^{n} a_{ij} = 1 , \quad i = 1, 2, \ldots, n$$

$$a_{ij} \le b_j , \quad i, j = 1, 2, \ldots, n$$

$$\sum_{j=1}^{n} b_j = k$$

$$a_{ij} = 1 \text{ or } 0 , \quad i = 1, 2, \ldots, n; \ j = 1, 2, \ldots, n$$

$$b_j = 1 \text{ or } 0 , \quad j = 1, 2, \ldots, n$$

$$a_{ij} = \begin{cases} 1, & \text{if } j\text{th object is the closest medoid of object } i \\ 0, & \text{otherwise} \end{cases}$$

$$b_j = \begin{cases} 1, & \text{if } j\text{th object is a medoid} \\ 0, & \text{otherwise} \end{cases}$$

PAM (Partitioning Around Medoids)

- It operates on a distance matrix of the given dataset, or the algorithm first computes a distance matrix when the data is presented with an $n \times p$ data matrix.

- The algorithm proceeds in two steps:
- BUILD step:
  - This step sequentially selects the k objects to be used as initial medoids.
- SWAP step:
  - If the objective function can be reduced (improved) by interchanging (swapping) a selected object with an unselected object, then the swap is carried out. This is continued until the objective function can no longer be decreased.

PAM Algorithm

- Notation
  - A set of objects in data: $S = \{O_1, O_2, \ldots, O_n\}$
  - A set of selected objects as medoids: M
  - A set of unselected objects: $U = S \setminus M$
  - Distance between object j and its closest object in M: D_j
  - Distance between object j and its second closest object in M: E_j
  - Amount (distance) of contribution of object j to the decision of selecting object i as a medoid: C_{ji}
  - Amount (distance) of contribution of object j to the decision of swapping object i and h, where $O_i \in M$ and $O_h \in U$: C_{jih}

  - Note that

$$D_j = 0 \text{ if and only if } O_j \in M$$

$$D_j \le E_j$$

PAM Algorithm

- BUILD
  - Step 0. Initialize M by adding to it an object for which the sum of the distances to all other objects is minimal.
  - Step 1. Consider an object i in U as a candidate for inclusion into M.
  - Step 2. For an object j in $U \setminus \{O_i\}$, compute D_j, which is the distance between object j and the closest object in M.
  - Step 3. If $D_j > d(j,i)$, object j will contribute to the decision to select object i as a medoid. Let $C_{ji} = \max\left(D_j - d(j,i),\ 0\right),\ \ i, j \in U$.
  - Step 4. Compute the total gain obtained by adding object i to M:
    $$g_i = \sum_{j \in U} C_{ji}$$

PAM Algorithm

- BUILD (cont'd)
  - Step 5. Choose the object i that maximizes g_i, and update M and U by $M \leftarrow M \cup \{O_i\}$ and $U \leftarrow U \setminus \{O_i\}$.
  - Step 6. If the number of objects in M is k, stop. Otherwise, go to Step 1.

PAM Algorithm

- SWAP
  - This step is done by considering all pairs of (O_i, O_h), $O_i \in M$, $O_h \in U$ consisting of computing the effect T_{ih} on the sum of distances between objects and the closest medoid (objective function), caused by swapping *i* and *h*, that is, transferring *i* from *M* to *U* and transferring *h* from *U* to *M*.
  - The computation of T_{ih} involves the computation C_{jih} of each object $O_j \in U \setminus \{O_h\}$ to the swap of object *i* and *h*. Note that we have either $d(j,i) > D_j$ or $d(j,i) = D_j$.
  - Step 1. Compute C_{jih} with consideration of the following 4 cases.

PAM Algorithm

- SWAP (cont'd)
  - Step 1. (cont'd)
    - (a) if $d(j,i) > D_j$, then two subcases occur:
      - (i) if $d(j,h) \geq D_j$, then $C_{jih} = 0$.
      - (ii) if $d(j,h) < D_j$, then $C_{jih} = d(j,h) - D_j$.
      - In both subcases, $C_{jih} = \min\{d(j,h) - D_j,\ 0\}$.
    - (b) if $d(j,i) = D_j$, then two subcases occur:
      - (i) if $d(j,h) < E_j$, then $C_{jih} = d(j,h) - D_j$.
        In this case, C_{jih} can be either positive or negative.
      - (ii) if $d(j,h) \geq E_j$, then $C_{jih} = E_j - D_j$.
        In this case, C_{jih} is always positive.
      - In both above subcases, $C_{jih} = \min\{d(j,h),\ E_j\} - D_j$.

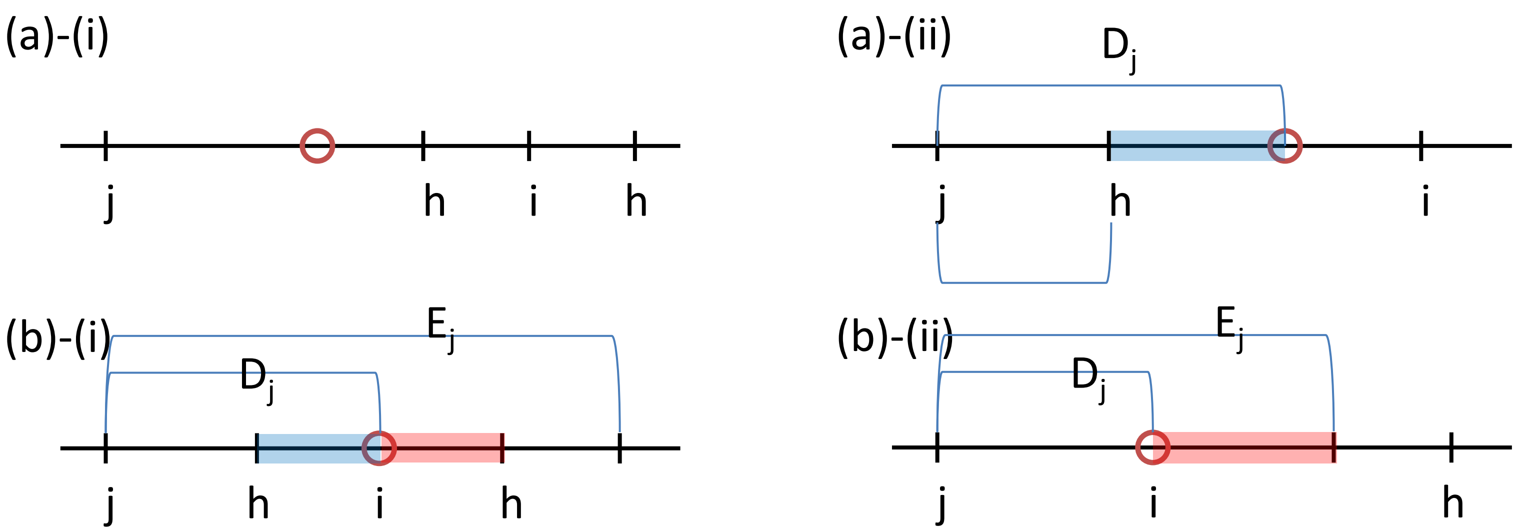
PAM Algorithm

- SWAP (cont'd)
  - Step 2. Compute $T_{ih} = \sum_{j \in U \setminus \{O_h\}} C_{jih}$.
  - Step 3. Select a pair of (O_i, O_h) that minimizes T_{ih} .
  - Step 4. If $T_{ih} < 0$, the swap is carried out. Also update the clustering result and D_j and E_j , and go to Step 1. Otherwise, stop.

- (a) if $d(j,i) > D_j$, then two subcases occur:
  - (i) if $d(j,h) \geq D_j$, then $C_{jih} = 0$.
  - (ii) if $d(j,h) < D_j$, then $C_{jih} = d(j,h) - D_j$.
  - In both subcases, $C_{jih} = \min\{d(j,h) - D_j,\ 0\}$.
- (b) if $d(j,i) = D_j$, then two subcases occur:
  - (i) if $d(j,h) < E_j$, then $C_{jih} = d(j,h) - D_j$.

    In this case, C_{jih} can be either positive or negative.
  - (ii) if $d(j,h) \geq E_j$, then $C_{jih} = E_j - D_j$.

    In this case, C_{jih} is always positive.
  - In both above subcases, $C_{jih} = \min\{d(j,h),\ E_j\} - D_j$.



PAM Algorithm

- Example
  - *k=2*

object	X_1	X_2
1	3	3
2	5	4
3	11	8
4	13	6
5	14	6
6	15	7

Distance matrix

object	1	2	3	4	5	6	Sum
1	0.00	2.24	9.43	10.44	11.40	12.65	46.16
2	2.24	0.00	7.21	8.25	9.22	10.44	37.35
3	9.43	7.21	0.00	2.83	3.61	4.12	27.20
4	10.44	8.25	2.83	0.00	1.00	2.24	24.75
5	11.40	9.22	3.61	1.00	0.00	1.41	26.64
6	12.65	10.44	4.12	2.24	1.41	0.00	30.86

Build

- Step 0. $M = \{O_4\}$

PAM Algorithm

- Example (cont'd)

i	j	D_j	$d(j,i)$	$D_j - d(j,i)$	C_{ji}
1	2	8.25	2.24	6.01	6.01
1	3	2.83	9.43	-6.61	0
1	5	1	11.4	-10.4	0
1	6	2.24	12.65	-10.41	0
				$g_i =$	6.01
2	1	10.44	2.24	8.2	8.2
2	3	2.83	7.21	-4.38	0
2	5	1	9.22	-8.22	0
2	6	2.24	10.44	-8.2	0
				$g_i =$	8.2
3	1	10.44	9.43	1.01	1.01
3	2	8.25	7.21	1.04	1.04
3	5	1	3.61	-2.61	0
3	6	2.24	4.12	-1.89	0
				$g_i =$	2.05
5	1	10.44	11.4	-0.96	0
5	2	8.25	9.22	-0.97	0
5	3	2.83	3.61	-0.78	0
5	6	2.24	1.41	0.82	0.82
				$g_i =$	0.82
6	1	10.44	12.65	-2.21	0
6	2	8.25	10.44	-2.19	0
6	3	2.83	4.12	-1.29	0
6	5	1	1.41	-0.41	0
				$g_i =$	0

Build

- Step 1. ~ 4.
- Step 5.

$$M = \{O_2, O_4\}$$

- Step 6. Stop, because k=2.

PAM Algorithm

- Example (cont'd)

Swap (Iteration 1.)

- Step 1. ~ 3.
- Step 4.

$$M = \{O_2, O_5\}$$

i	h	j	$d(j,i)$	$d(j,h)$	D_j	E_j	Case	C_{jih}
2	1	3	7.21	9.43	2.83	7.21	a(i)	0.00
		5	9.22	11.40	1.00	9.22	a(i)	0.00
		6	10.44	12.65	2.24	10.44	a(i)	0.00
							$T_{ih} =$	0.00
	3	1	2.24	9.43	2.24	10.44	b(i)	7.19
		5	9.22	3.61	1.00	9.22	a(i)	0.00
		6	10.44	4.12	2.24	10.44	a(i)	0.00
							$T_{ih} =$	7.19
	5	1	2.24	11.40	2.24	10.44	b(ii)	8.20
		3	7.21	3.61	2.83	7.21	a(i)	0.00
		6	10.44	1.41	2.24	10.44	a(ii)	-0.83
							$T_{ih} =$	7.37
	6	1	2.24	12.65	2.24	10.44	b(ii)	8.20
		3	7.21	4.12	2.83	7.21	a(i)	0.00
		5	9.22	1.41	1.00	9.22	a(i)	0.00
							$T_{ih} =$	8.20
4	1	3	2.83	9.43	2.83	7.21	b(ii)	4.38
		5	1.00	11.40	1.00	9.22	b(ii)	8.22
		6	2.24	12.65	2.24	10.44	b(ii)	8.20
							$T_{ih} =$	20.81
	3	1	10.44	9.43	2.24	10.44	a(i)	0.00
		5	1.00	3.61	1.00	11.40	b(i)	2.61
		6	2.24	4.12	2.24	12.65	b(ii)	10.41
							$T_{ih} =$	13.02
	5	1	10.44	11.40	2.24	10.44	a(i)	0.00
		3	2.83	3.61	2.83	9.43	b(i)	0.78
		6	2.24	1.41	2.24	12.65	b(i)	-0.83
							$T_{ih} =$	-0.05
	6	1	10.44	12.65	2.24	10.44	a(i)	0.00
		3	2.83	4.12	2.83	9.43	b(i)	1.29
		5	1.00	1.41	1.00	11.40	b(i)	0.41
							$T_{ih} =$	1.71

PAM Algorithm

- Example (cont'd)

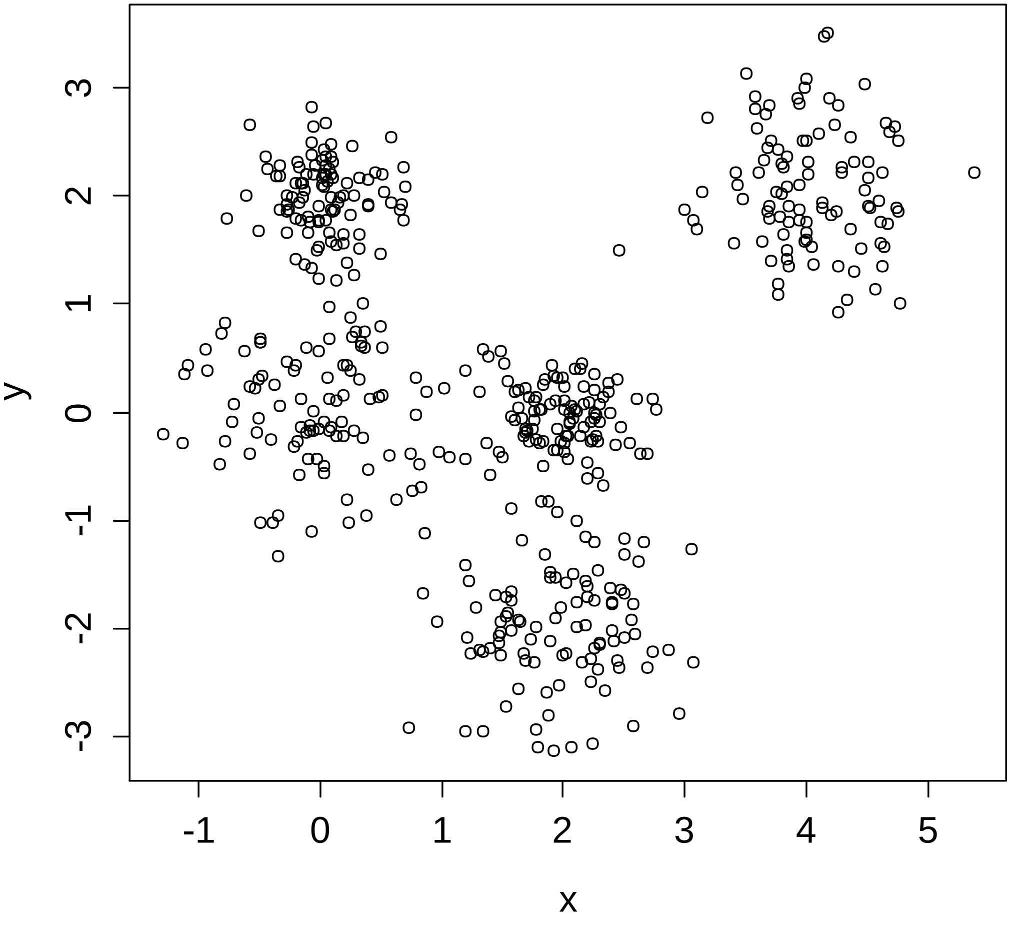
  Iteration 2.

  Fill in the next table and see which objects should be swapped.

i	h	j	$d(j,i)$	$d(j,h)$	D_j	E_j	Case	C_{jih}
2	1	3						
		4						
		6						
	3	1						
		4						
		6						
	4	1						
		3						
		6						
	6	1						
		3						
		4						
5	1	3						
		4						
		6						
	3	1						
		4						
		6						
	4	1						
		3						
		6						
	6	1						
		3						
		4						

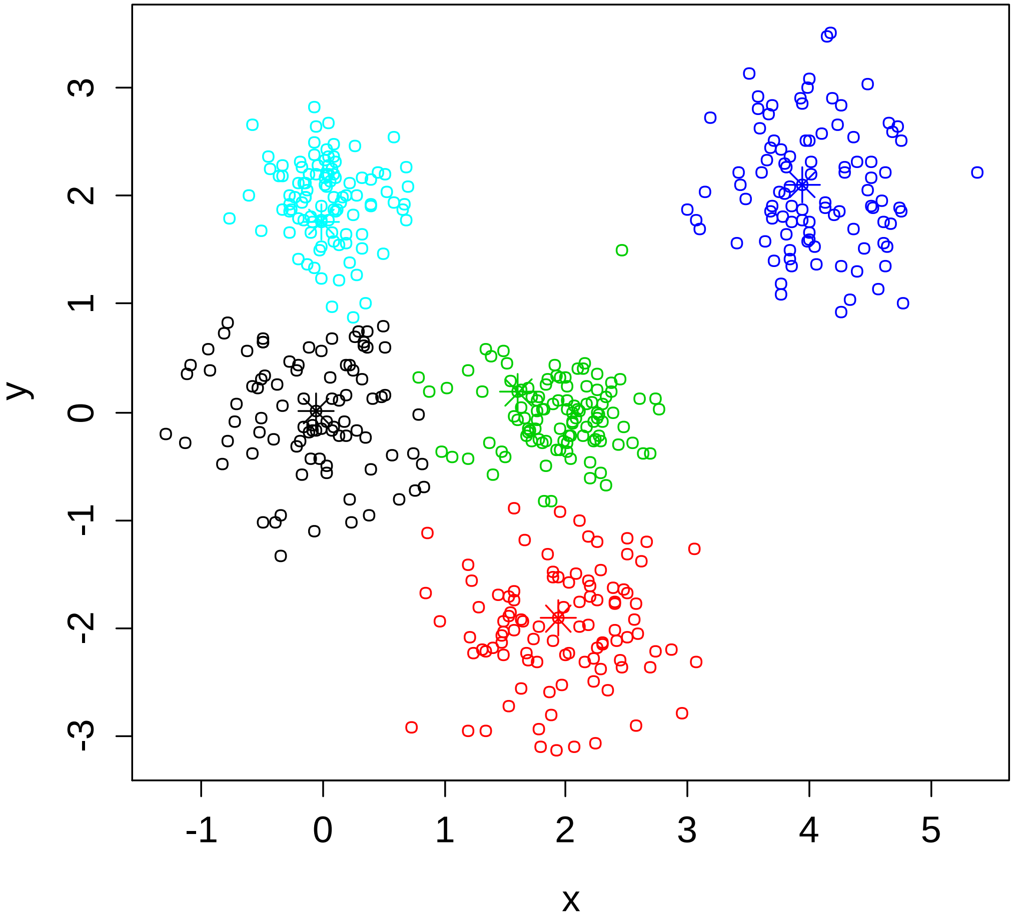
PAM Algorithm

- Demo (k=5)



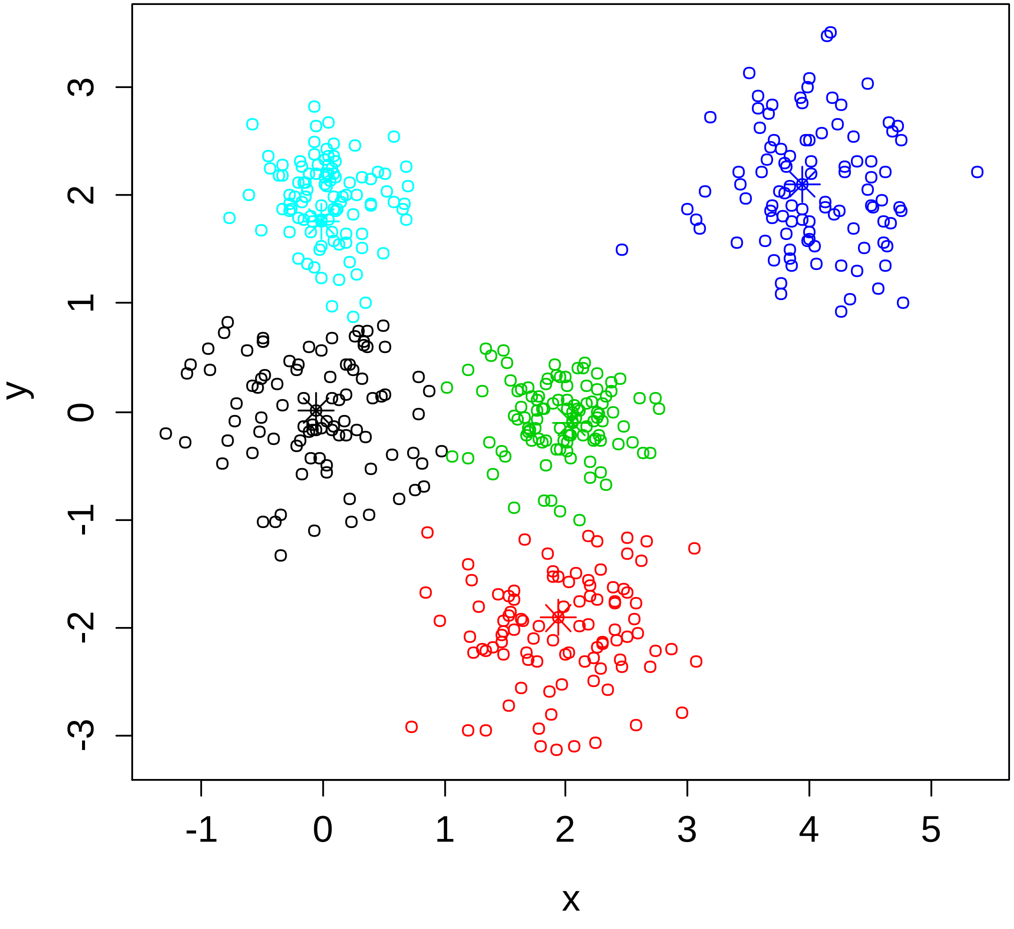
PAM Algorithm

- Demo (BUILD)



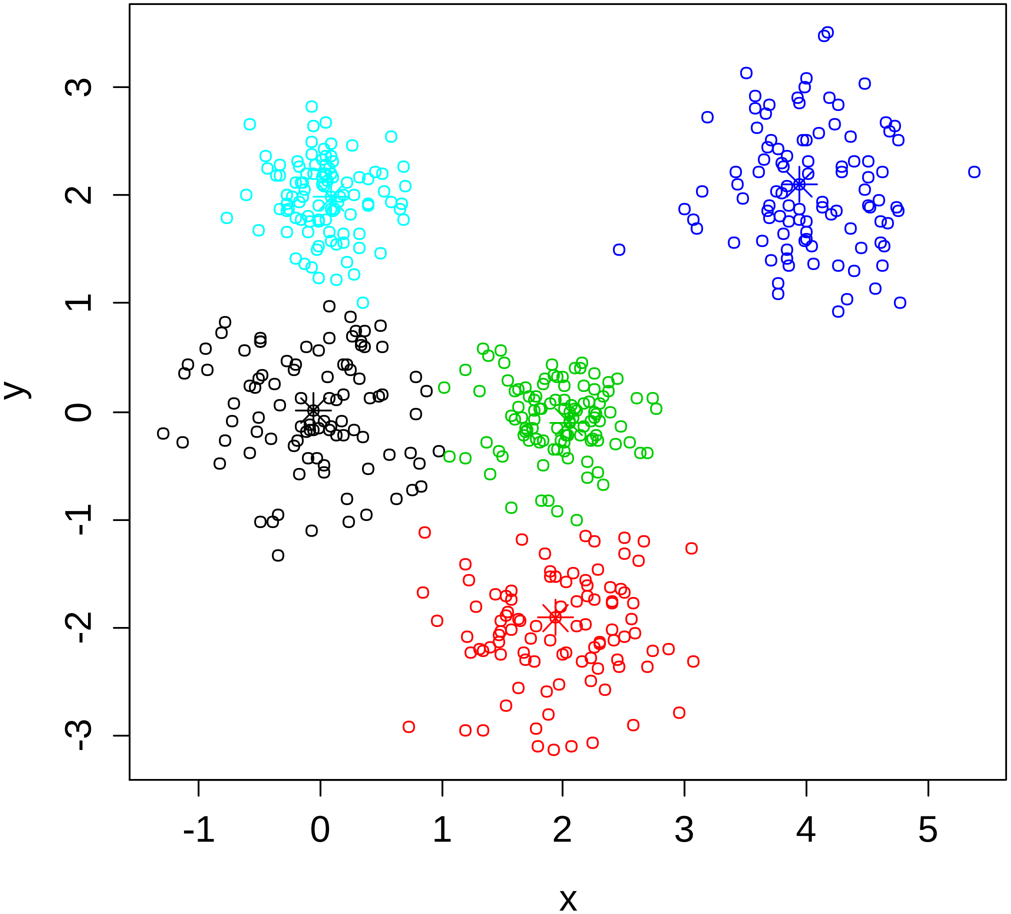
PAM Algorithm

- Demo (SWAP: Iteration 1)



PAM Algorithm

- Demo (SWAP: Iteration 2, Converged)



R Example

- Partitioning Around Medoids

```
> library(cluster)
> ## generate 25 objects, divided into 2 clusters.
> x <- rbind(cbind(rnorm(10,0,0.5), rnorm(10,0,0.5)),
+            cbind(rnorm(15,5,0.5), rnorm(15,5,0.5)))
> pamx <- pam(x, 2)
> pamx
Medoids:
     ID
[1,]  9 0.03925164 0.05020356
[2,] 18 5.07522377 5.18310846
Clustering vector:
 [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
Objective function:
    build      swap
0.8288646 0.6135317

> plot(pamx)
```