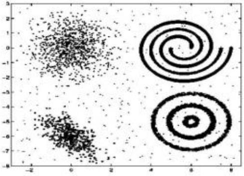
Clustering: K-means and Hierarchical Clustering

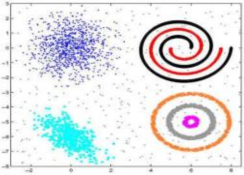
Machine Learning

Clustering

- Unsupervised learning problem
- Given: N unlabeled examples $\{x_1, \ldots, x_N\}$; the number of partitions K
- Goal: Group the examples into K partitions



(a) Input data



(b) Desired clustering

- The only information clustering uses is the similarity between examples
- A good clustering is one that achieves:
- High within-cluster similarity
  - Low inter-cluster similarity

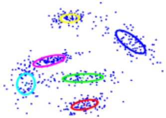
Similarity is Subjective

- Similarity is often hard to define

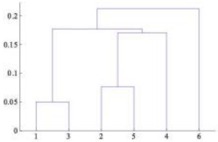
- Different similarity criteria can lead to different clusterings

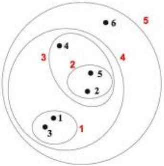
Types of Clustering

1. Flat or Partitional clustering (K -means, Gaussian mixture models, etc.)
   - Partitions are independent of each other

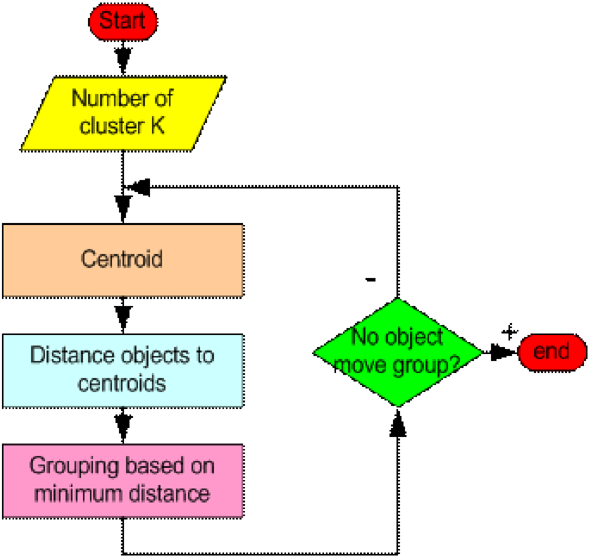


2. Hierarchical clustering (e.g., agglomerative clustering, divisive clustering)
   - Partitions can be visualized using a tree structure (a dendrogram)
   - Does not need the number of clusters as input
   - Possible to view partitions at different levels of granularities using different K

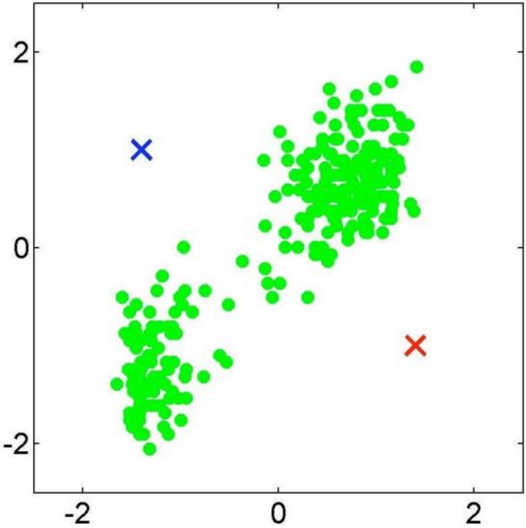




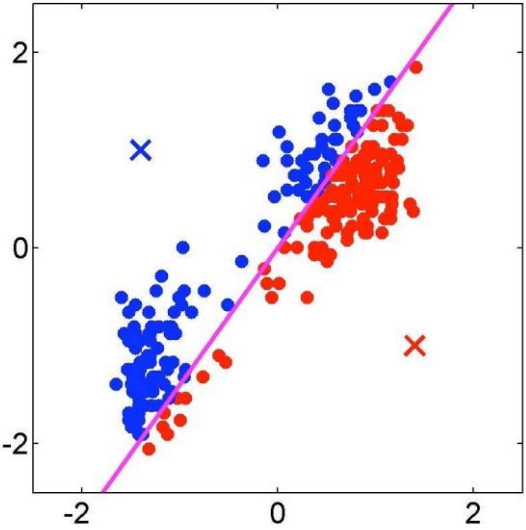
Flat Clustering: K -means algorithm (Lloyd, 1957)



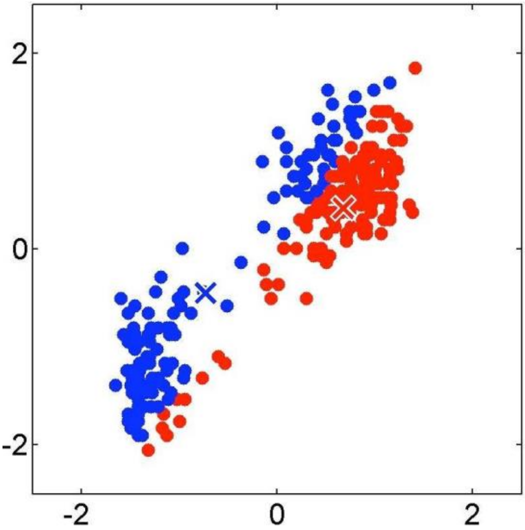
K -means: Initialization (assume $K = 2$)



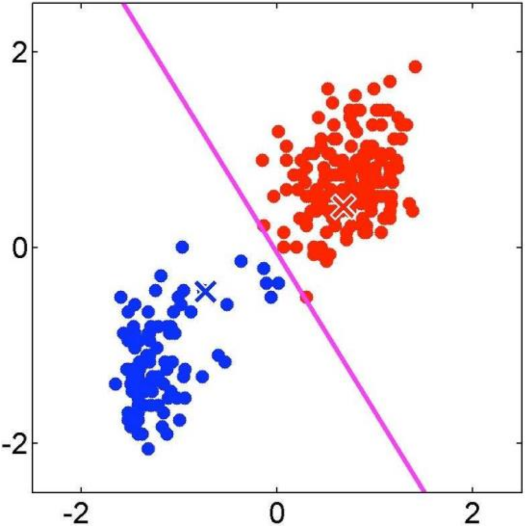
K-means iteration 1: Assigning points



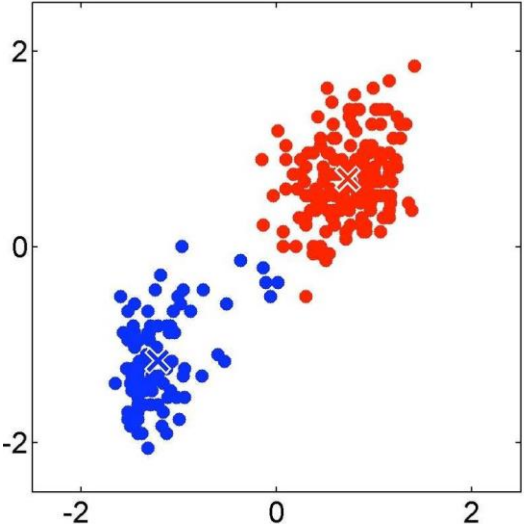
K -means iteration 1: Recomputing the cluster centers



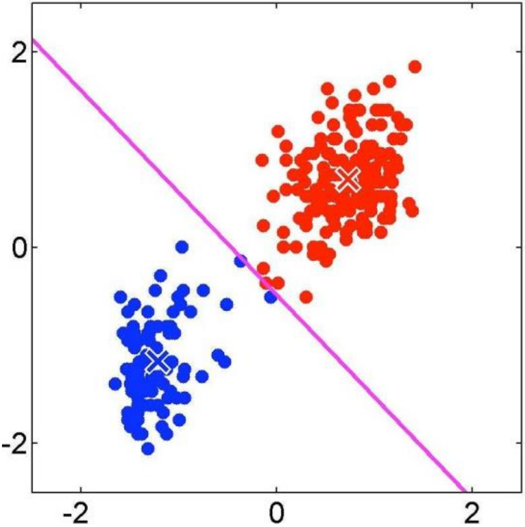
K -means iteration 2: Assigning points



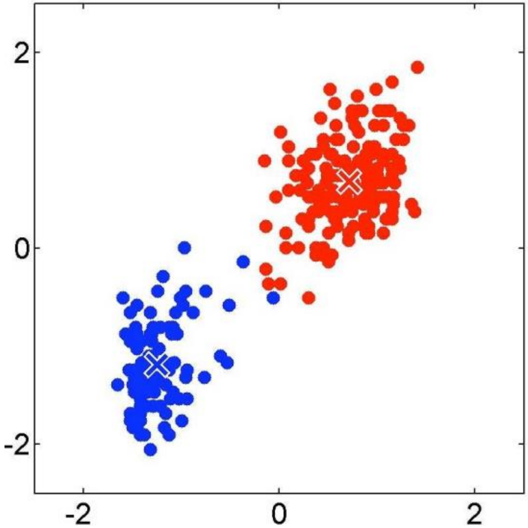
K-means iteration 2: Recomputing the cluster centers



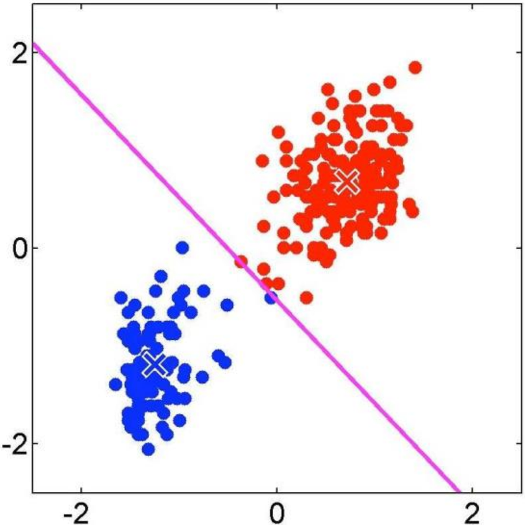
K -means iteration 3: Assigning points



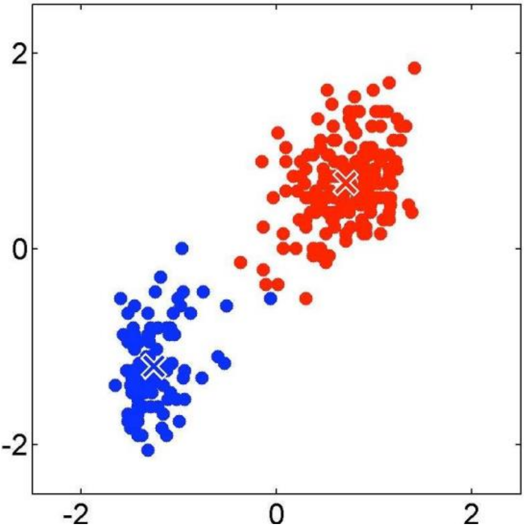
K-means iteration 3: Recomputing the cluster centers



K -means iteration 4: Assigning points



K-means iteration 4: Recomputing the cluster centers



Example

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

Step 1:

Initialization: Randomly we choose following two centroids (k=2) for two clusters.

In this case the 2 centroid are: m1=(1.0,1.0) and m2=(5.0,7.0).

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

Step 2:

Thus, we obtain two clusters containing:

{1,2,3} and {4,5,6,7}.

Their new centroids are:

$$m_1 = \left(\frac{1}{3}(1.0+1.5+3.0), \frac{1}{3}(1.0+2.0+4.0)\right) = (1.83, 2.33)$$

$$m_2 = \left(\frac{1}{4}(5.0+3.5+4.5+3.5), \frac{1}{4}(7.0+5.0+5.0+4.5)\right)$$
$$= (4.12, 5.38)$$

Individual	Centroid 1	Centroid 2
1	0	7.21
2 (1.5, 2.0)	1.12	6.10
3	3.61	3.61
4	7.21	0
5	4.72	2.5
6	5.31	2.06
7	4.30	2.92

$$d(m_1, 2) = \sqrt{|1.0-1.5|^2 + |1.0-2.0|^2} = 1.12$$
$$d(m_2, 2) = \sqrt{|5.0-1.5|^2 + |7.0-2.0|^2} = 6.10$$

Step 3:

Now using these centroids we compute the Euclidean distance of each object, as shown in table.

Therefore, the new clusters are:
{1,2} and {**3**,4,5,6,7}

Next centroids are:
m1=(1.25,1.5) and m2 = (3.9,5.1)

Individual	Centroid 1	Centroid 2
1	1.57	5.38
2	0.47	4.28
3	2.04	1.78
4	5.64	1.84
5	3.15	0.73
6	3.78	0.54
7	2.74	1.08

Step 4 :

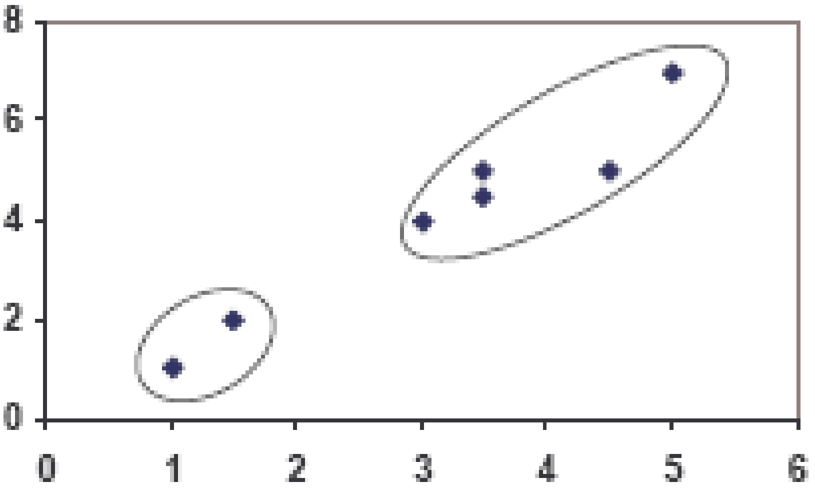
The clusters obtained are:

{1,2} and {3,4,5,6,7}

Individual	Centroid 1	Centroid 2
1	0.56	5.02
2	0. 56	3.92
3	3.05	1.42
4	6.66	2.20
5	4.16	0.41
6	4.78	0.61
7	3.75	0.72

Therefore, there is no change in the cluster.

Thus, the algorithm comes to a halt here and final result consist of 2 clusters {1,2} and {3,4,5,6,7}.


8
6
4
2
0
0
1
2
3
4
5
6

K -means iteration: Example

Documents (Data Points)	W1 (x-axis)	W2 (y-axis)
D1	2	0
D2	1	3
D3	3	5
D4	2	2
D5	4	6

D2 and D4 are the centroids
Euclidean distance – sample below:

Distance between D1 and D2	Distance between D1 and D4
$\sqrt{(2-1)^2+(0-3)^2}$	$\sqrt{(2-2)^2+(0-2)^2}$
$= \sqrt{(1)^2+(3)^2}$	$= \sqrt{(0)^2+(-2)^2}$
$= \sqrt{1+9}$	$= \sqrt{0+4}$
$= \sqrt{10} = 3.17$	$= \sqrt{4} = 2$

K -means iteration: Example

Documents (Data Points)	Distance between D2 and other data points	Distance between D4 and other data points
D1	3.17	2.0
D3	2.83	3.17
D5	4.25	4.48

Cluster 1 – D2, D3, D5
Cluster 2 – D4, D1
Next step – calculate new centroid for each cluster

Clusters	Mean value of data points along x -axis	Mean value of data points along y-axis
D1, D4	2.0	1.0
D2, D3, D5	2.67	4.67

(2, 1) and (2.67, 4.67)

K -means iteration: Example

Documents (Data Points)	Distance between centroid of cluster 1 and data points	Distance between centroid of cluster 2 and data points
D1	1.0	4.72
D2	2.24	2.37
D3	4.13	0.47
D4	1	2.76
D5	5.39	1.89

Previous:

Cluster 1 – D2, D3, D5

Cluster 2 – D4, D1

New:

Cluster 1 – D1, D2, D4

Cluster 2 – D3, D5

Calculate new centroids and repeat

K-means: Initialization issues

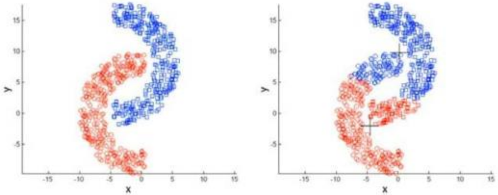
- K-means is extremely sensitive to cluster center initialization
- Bad initialization can lead to
  - Poor convergence speed
  - Bad overall clustering
- Safeguarding measures:
  - Choose first center as one of the examples, second which is the farthest from the first, third which is the farthest from both, and so on.
  - Try multiple initializations and choose the best result
  - Other smarter initialization schemes (e.g., look at the K-means++ algorithm by Arthur and Vassilvitskii)

K -means: Limitations

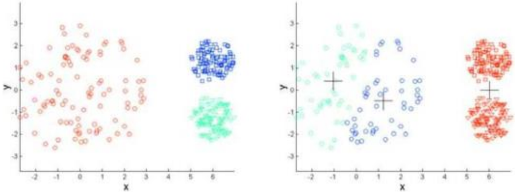
- Makes hard assignments of points to clusters
  - A point either completely belongs to a cluster or not at all
  - No notion of a soft assignment (i.e., probability of being assigned to each cluster: say $K = 3$ and for some point x_n , $p_1 = 0.7$, $p_2 = 0.2$, $p_3 = 0.1$)
  - Gaussian mixture models and Fuzzy K -means allow soft assignments
- Sensitive to outlier examples (such examples can affect the mean by a lot) K
  - -medians algorithm is a more robust alternative for data with outliers Reason:
  - Median is more robust than mean in presence of outliers
- Works well only for round shaped, and of roughly equal sizes/density clusters
- Does badly if the clusters have non-convex shapes
  - Spectral clustering or kernelized K -means can be an alternative

K -means Limitations Illustrated

Non-convex/non-round-shaped clusters: Standard K -means fails!

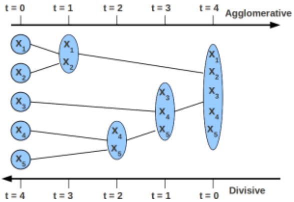


Clusters with different densities



Picture courtesy: Christof Monz (Queen Mary, Univ. of London)

Hierarchical Clustering



- Agglomerative (bottom-up) Clustering
  1. Start with each example in its own singleton cluster
  2. At each time-step, greedily merge 2 most similar clusters
  3. Stop when there is a single cluster of all examples, else go to 2
- Divisive (top-down) Clustering
  1. Start with all examples in the same cluster
  2. At each time-step, remove the “outsiders” from the least cohesive cluster
  3. Stop when each example is in its own singleton cluster, else go to 2
- Agglomerative is more popular and simpler than divisive (but less accurarate)