

# Data Clustering: K-means and Hierarchical Clustering

Piyush Rai

CS5350/6350: Machine Learning

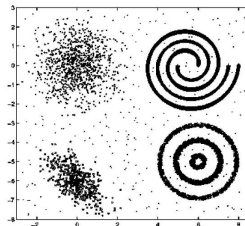
October 4, 2011

# What is Data Clustering?

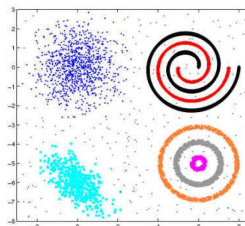
- Data Clustering is an **unsupervised learning** problem

# What is Data Clustering?

- Data Clustering is an **unsupervised learning** problem
- Given:  $N$  **unlabeled** examples  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ ; the number of partitions  $K$
- Goal: Group the examples into  $K$  partitions



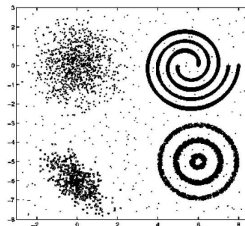
(a) Input data



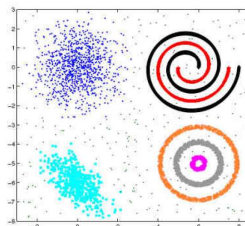
(b) Desired clustering

# What is Data Clustering?

- Data Clustering is an **unsupervised learning** problem
- Given:  $N$  **unlabeled** examples  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ ; the number of partitions  $K$
- Goal: Group the examples into  $K$  partitions



(a) Input data

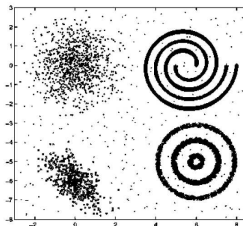


(b) Desired clustering

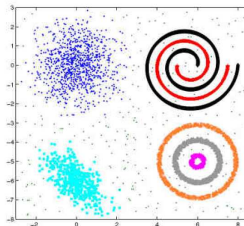
- The only information clustering uses is the **similarity between examples**
- Clustering groups examples based of their mutual similarities

# What is Data Clustering?

- Data Clustering is an **unsupervised learning** problem
- Given:  $N$  **unlabeled** examples  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ ; 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**
- Clustering groups examples based of their mutual similarities
- A good clustering is one that achieves:
  - **High within-cluster similarity**
  - **Low inter-cluster similarity**

Picture courtesy: "Data Clustering: 50 Years Beyond K-Means", A.K. Jain (2008)

# Notions of Similarity

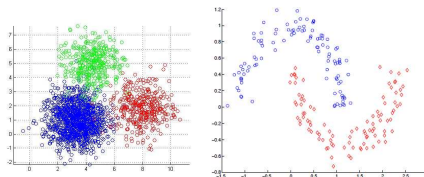
- Choice of the **similarity measure** is **very important** for clustering
- Similarity is inversely related to distance

# Notions of Similarity

- Choice of the **similarity measure** is **very important** for clustering
- Similarity is inversely related to distance
- Different ways exist to measure distances. Some examples:
  - Euclidean distance:  $d(\mathbf{x}, \mathbf{z}) = \|\mathbf{x} - \mathbf{z}\| = \sqrt{\sum_{d=1}^D (x_d - z_d)^2}$
  - Manhattan distance:  $d(\mathbf{x}, \mathbf{z}) = \sum_{d=1}^D |x_d - z_d|$
  - Kernelized (non-linear) distance:  $d(\mathbf{x}, \mathbf{z}) = \|\phi(\mathbf{x}) - \phi(\mathbf{z})\|$

# Notions of Similarity

- Choice of the **similarity measure** is **very important** for clustering
- Similarity is inversely related to distance
- Different ways exist to measure distances. Some examples:
  - Euclidean distance:  $d(\mathbf{x}, \mathbf{z}) = \|\mathbf{x} - \mathbf{z}\| = \sqrt{\sum_{d=1}^D (x_d - z_d)^2}$
  - Manhattan distance:  $d(\mathbf{x}, \mathbf{z}) = \sum_{d=1}^D |x_d - z_d|$
  - Kernelized (non-linear) distance:  $d(\mathbf{x}, \mathbf{z}) = \|\phi(\mathbf{x}) - \phi(\mathbf{z})\|$



- For the left figure above, Euclidean distance may be reasonable
- For the right figure above, kernelized distance seems more reasonable



# Similarity is Subjective

- Similarity is often hard to define



- Different similarity criteria can lead to different clusterings

# Data Clustering: Some Real-World Examples

- Clustering images based on their perceptual similarities
- Image segmentation (clustering pixels)



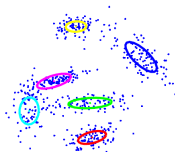
- Clustering webpages based on their content
- Clustering web-search results
- Clustering people in social networks based on user properties/preferences
- .. and many more..

Picture courtesy: <http://people.cs.uchicago.edu/~pff/segment/>

# Types of Clustering

## • **Flat or Partitional clustering** ( $K$ -means, Gaussian mixture models, etc.)

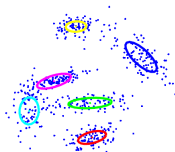
- Partitions are **independent of each other**



# Types of Clustering

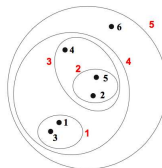
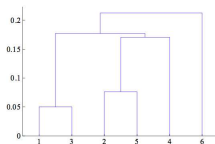
## 1 Flat or Partitional clustering ( $K$ -means, Gaussian mixture models, etc.)

- 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** (i.e., can refine/coarsen clusters) using different  $K$



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

- **Input:**  $N$  examples  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  ( $\mathbf{x}_n \in \mathbb{R}^D$ ); the number of partitions  $K$

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

- **Input:**  $N$  examples  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  ( $\mathbf{x}_n \in \mathbb{R}^D$ ); the number of partitions  $K$
- **Initialize:**  $K$  cluster centers  $\mu_1, \dots, \mu_K$ . Several initialization options:
  - Randomly initialized anywhere in  $\mathbb{R}^D$
  - Choose any  $K$  examples as the cluster centers

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

- **Input:**  $N$  examples  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  ( $\mathbf{x}_n \in \mathbb{R}^D$ ); the number of partitions  $K$
- **Initialize:**  $K$  cluster centers  $\mu_1, \dots, \mu_K$ . Several initialization options:
  - Randomly initialized anywhere in  $\mathbb{R}^D$
  - Choose any  $K$  examples as the cluster centers
- **Iterate:**
  - Assign each of example  $\mathbf{x}_n$  to its closest cluster center

$$\mathcal{C}_k = \{n : k = \arg \min_k \|\mathbf{x}_n - \mu_k\|^2\}$$

( $\mathcal{C}_k$  is the set of examples closest to  $\mu_k$ )

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

- **Input:**  $N$  examples  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  ( $\mathbf{x}_n \in \mathbb{R}^D$ ); the number of partitions  $K$
- **Initialize:**  $K$  cluster centers  $\mu_1, \dots, \mu_K$ . Several initialization options:
  - Randomly initialized anywhere in  $\mathbb{R}^D$
  - Choose any  $K$  examples as the cluster centers
- **Iterate:**
  - Assign each of example  $\mathbf{x}_n$  to its closest cluster center

$$\mathcal{C}_k = \{n : k = \arg \min_k \|\mathbf{x}_n - \mu_k\|^2\}$$

( $\mathcal{C}_k$  is the set of examples closest to  $\mu_k$ )

- Recompute the new cluster centers  $\mu_k$  (mean/centroid of the set  $\mathcal{C}_k$ )

$$\mu_k = \frac{1}{|\mathcal{C}_k|} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n$$



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

- **Input:**  $N$  examples  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  ( $\mathbf{x}_n \in \mathbb{R}^D$ ); the number of partitions  $K$
- **Initialize:**  $K$  cluster centers  $\mu_1, \dots, \mu_K$ . Several initialization options:
  - Randomly initialized anywhere in  $\mathbb{R}^D$
  - Choose any  $K$  examples as the cluster centers
- **Iterate:**
  - Assign each of example  $\mathbf{x}_n$  to its closest cluster center

$$\mathcal{C}_k = \{n : k = \arg \min_k \|\mathbf{x}_n - \mu_k\|^2\}$$

( $\mathcal{C}_k$  is the set of examples closest to  $\mu_k$ )

- **Recompute** the new cluster centers  $\mu_k$  (mean/centroid of the set  $\mathcal{C}_k$ )

$$\mu_k = \frac{1}{|\mathcal{C}_k|} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n$$

- **Repeat** while not converged

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

- **Input:**  $N$  examples  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  ( $\mathbf{x}_n \in \mathbb{R}^D$ ); the number of partitions  $K$
- **Initialize:**  $K$  cluster centers  $\mu_1, \dots, \mu_K$ . Several initialization options:
  - Randomly initialized anywhere in  $\mathbb{R}^D$
  - Choose any  $K$  examples as the cluster centers
- **Iterate:**
  - Assign each of example  $\mathbf{x}_n$  to its closest cluster center

$$\mathcal{C}_k = \{n : k = \arg \min_k \|\mathbf{x}_n - \mu_k\|^2\}$$

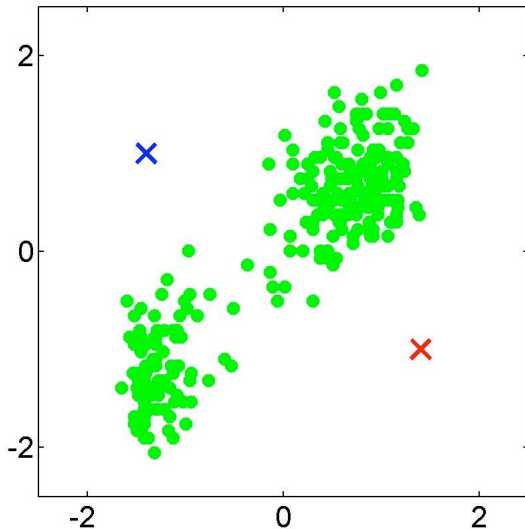
( $\mathcal{C}_k$  is the set of examples closest to  $\mu_k$ )

- Recompute the new cluster centers  $\mu_k$  (mean/centroid of the set  $\mathcal{C}_k$ )

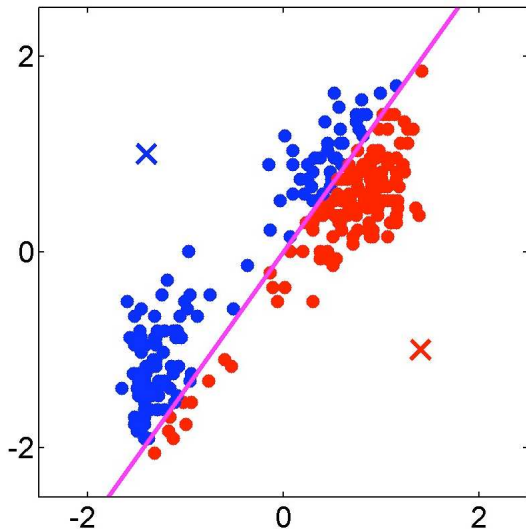
$$\mu_k = \frac{1}{|\mathcal{C}_k|} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n$$

- Repeat while not converged
- A possible convergence criteria: cluster centers do not change anymore

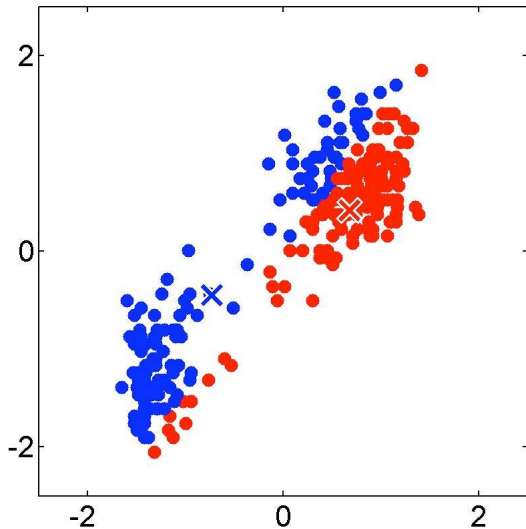
## $K$ -means: Initialization (assume $K = 2$ )



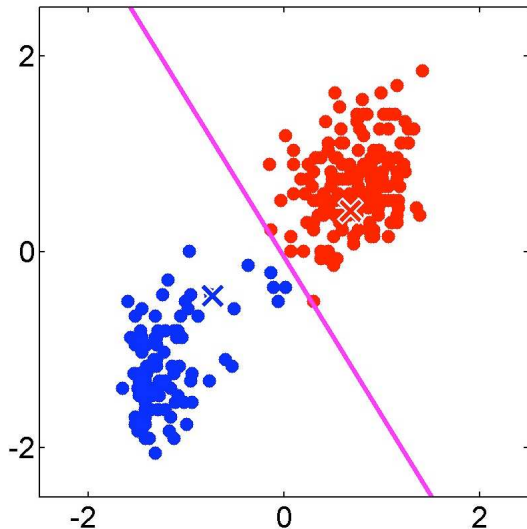
## $K$ -means iteration 1: Assigning points



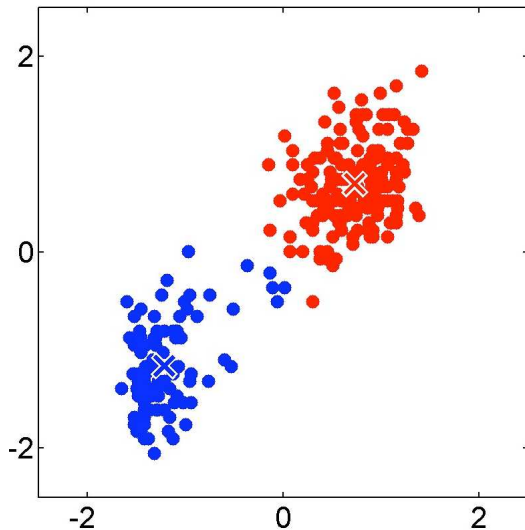
## $K$ -means iteration 1: Recomputing the cluster centers



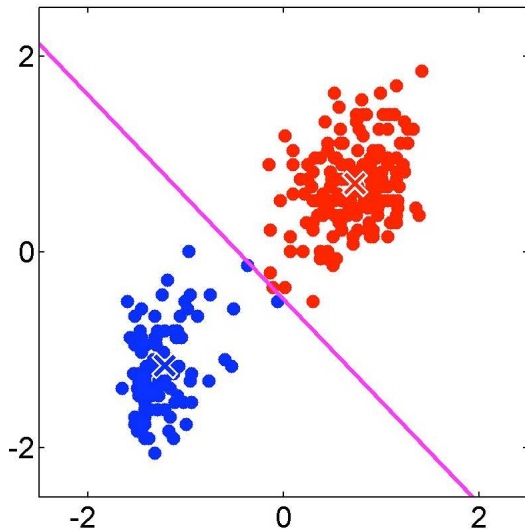
## $K$ -means iteration 2: Assigning points



## $K$ -means iteration 2: Recomputing the cluster centers

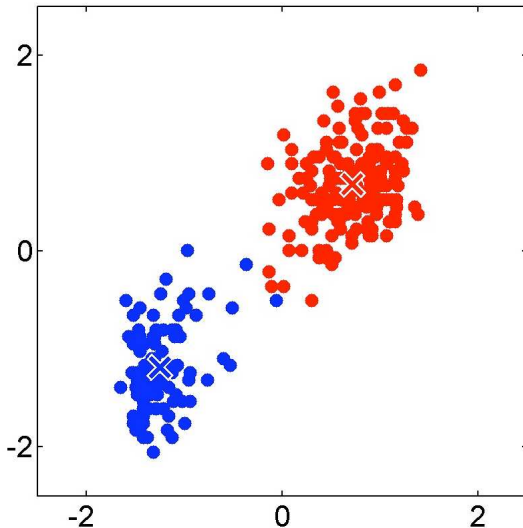


## $K$ -means iteration 3: Assigning points

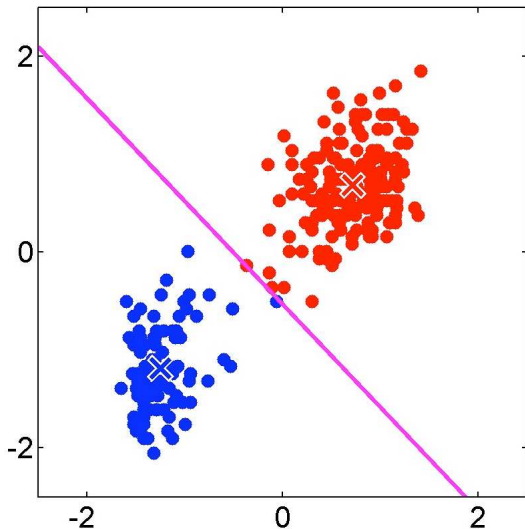




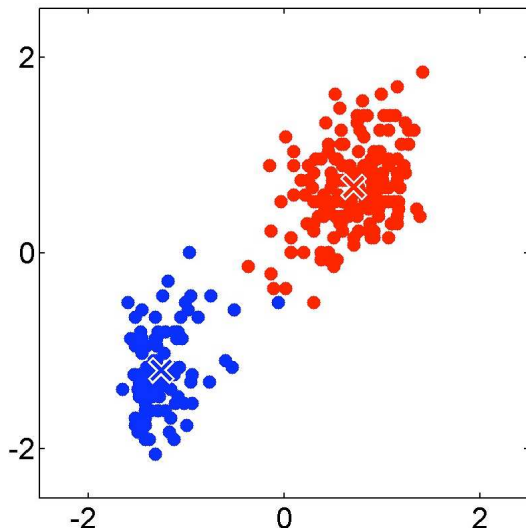
## $K$ -means iteration 3: Recomputing the cluster centers



## $K$ -means iteration 4: Assigning points



## $K$ -means iteration 4: Recomputing the cluster centers



# K-means: The Objective Function

The  $K$ -means objective function

- Let  $\mu_1, \dots, \mu_K$  be the  $K$  cluster centroids (means)
- Let  $r_{nk} \in \{0, 1\}$  be **indicator** denoting whether point  $\mathbf{x}_n$  belongs to cluster  $k$
- $K$ -means objective minimizes the total **distortion** (sum of distances of points from their cluster centers)

$$J(\mu, r) = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \mu_k\|^2$$

# K-means: The Objective Function

The  $K$ -means objective function

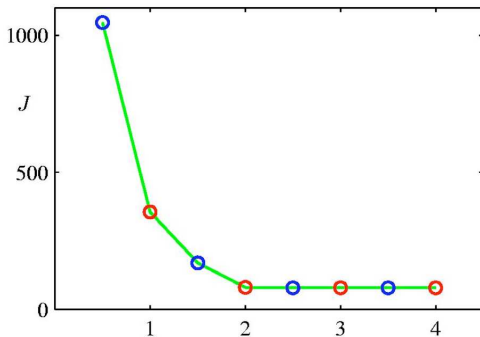
- Let  $\mu_1, \dots, \mu_K$  be the  $K$  cluster centroids (means)
- Let  $r_{nk} \in \{0, 1\}$  be **indicator** denoting whether point  $\mathbf{x}_n$  belongs to cluster  $k$
- $K$ -means objective minimizes the total **distortion** (sum of distances of points from their cluster centers)

$$J(\mu, r) = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \mu_k\|^2$$

- Note: **Exact optimization** of the  $K$ -means objective is **NP-hard**
- The  $K$ -means algorithm is a **heuristic** that converges to a local optimum

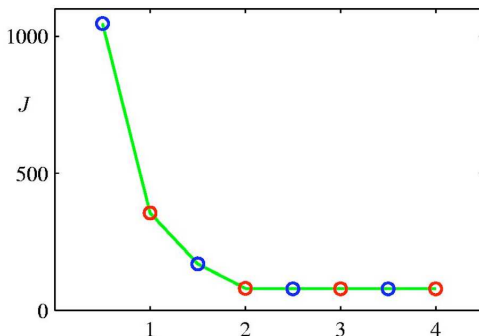
# K-means: Choosing the number of clusters $K$

- One way to select  $K$  for the  $K$ -means algorithm is to try different values of  $K$ , plot the  $K$ -means objective versus  $K$ , and look at the “elbow-point” in the plot



# K-means: Choosing the number of clusters $K$

- One way to select  $K$  for the  $K$ -means algorithm is to try different values of  $K$ , plot the  $K$ -means objective versus  $K$ , and look at the “elbow-point” in the plot



- For the above plot,  $K = 2$  is the elbow point

Picture courtesy: "Pattern Recognition and Machine Learning, Chris Bishop (2006)

# $K$ -means: Initialization issues

- $K$ -means is **extremely sensitive to cluster center initialization**
- Bad initialization can lead to
  - Poor convergence speed
  - Bad overall clustering



# 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.

# 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**

# 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

# K-means: Limitations

- Makes **hard assignments** of points to clusters
  - A point either completely belongs to a cluster or not belongs at all

# K-means: Limitations

- Makes **hard assignments** of points to clusters
  - A point either completely belongs to a cluster or not belongs at all
  - No notion of a **soft assignment** (i.e., **probability** of being assigned to each cluster: say  $K = 3$  and for some point  $\mathbf{x}_n$ ,  $p_1 = 0.7, p_2 = 0.2, p_3 = 0.1$ )

# K-means: Limitations

- Makes **hard assignments** of points to clusters
  - A point either completely belongs to a cluster or not belongs at all
  - No notion of a **soft assignment** (i.e., **probability** of being assigned to each cluster: say  $K = 3$  and for some point  $\mathbf{x}_n$ ,  $p_1 = 0.7, p_2 = 0.2, p_3 = 0.1$ )
  - **Gaussian mixture models** and **Fuzzy K-means** allow soft assignments

# K-means: Limitations

- Makes **hard assignments** of points to clusters
  - A point either completely belongs to a cluster or not belongs at all
  - No notion of a **soft assignment** (i.e., **probability** of being assigned to each cluster: say  $K = 3$  and for some point  $\mathbf{x}_n$ ,  $p_1 = 0.7, p_2 = 0.2, p_3 = 0.1$ )
  - **Gaussian mixture models** and **Fuzzy K-means** allow soft assignments
- Sensitive to **outlier examples** (such examples can affect the mean by a lot)



# K-means: Limitations

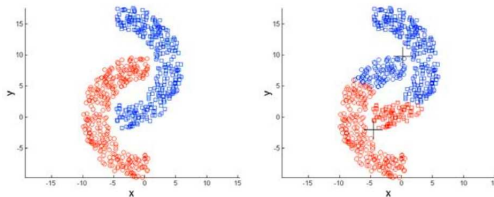
- Makes **hard assignments** of points to clusters
  - A point either completely belongs to a cluster or not belongs at all
  - No notion of a **soft assignment** (i.e., **probability** of being assigned to each cluster: say  $K = 3$  and for some point  $\mathbf{x}_n$ ,  $p_1 = 0.7$ ,  $p_2 = 0.2$ ,  $p_3 = 0.1$ )
  - **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

# K-means: Limitations

- Makes **hard assignments** of points to clusters
  - A point either completely belongs to a cluster or not belongs at all
  - No notion of a **soft assignment** (i.e., **probability** of being assigned to each cluster: say  $K = 3$  and for some point  $\mathbf{x}_n$ ,  $p_1 = 0.7$ ,  $p_2 = 0.2$ ,  $p_3 = 0.1$ )
  - **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

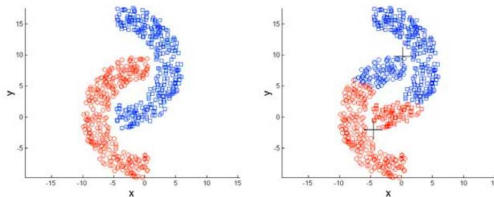
# K-means Limitations Illustrated

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

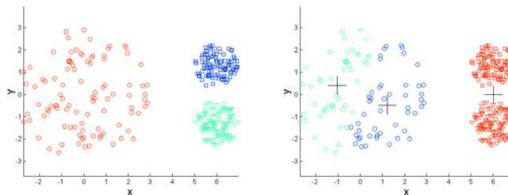


# K-means Limitations Illustrated

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

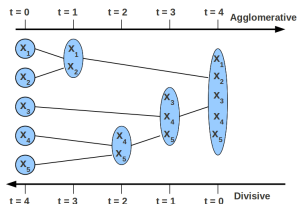


Clusters with different densities

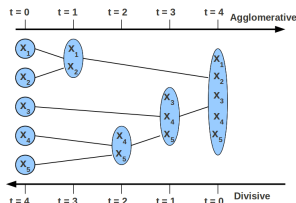


Picture courtesy: Christof Monz (Queen Mary, Univ. of London)

# Hierarchical Clustering



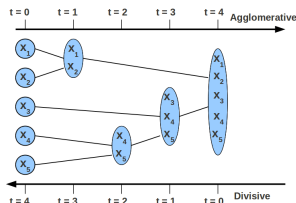
# Hierarchical Clustering



- **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

# Hierarchical Clustering



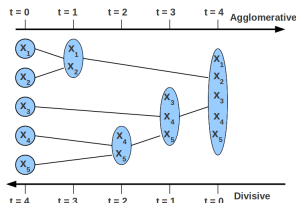
- **Agglomerative (bottom-up) Clustering**

- 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

# 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 accurate)



# Hierarchical Clustering: (Dis)similarity between clusters

- We know how to compute the dissimilarity  $d(\mathbf{x}_i, \mathbf{x}_j)$  between two examples
- How to compute the dissimilarity between two clusters  $R$  and  $S$ ?

# Hierarchical Clustering: (Dis)similarity between clusters

- We know how to compute the dissimilarity  $d(\mathbf{x}_i, \mathbf{x}_j)$  between two examples
- How to compute the dissimilarity between two clusters  $R$  and  $S$ ?
- **Min-link or single-link:** results in chaining (clusters can get very large)

$$d(R, S) = \min_{\mathbf{x}_R \in R, \mathbf{x}_S \in S} d(\mathbf{x}_R, \mathbf{x}_S)$$

# Hierarchical Clustering: (Dis)similarity between clusters

- We know how to compute the dissimilarity  $d(\mathbf{x}_i, \mathbf{x}_j)$  between two examples
- How to compute the dissimilarity between two clusters  $R$  and  $S$ ?
- **Min-link or single-link:** results in chaining (clusters can get very large)

$$d(R, S) = \min_{\mathbf{x}_R \in R, \mathbf{x}_S \in S} d(\mathbf{x}_R, \mathbf{x}_S)$$

- **Max-link or complete-link:** results in small, round shaped clusters

$$d(R, S) = \max_{\mathbf{x}_R \in R, \mathbf{x}_S \in S} d(\mathbf{x}_R, \mathbf{x}_S)$$

# Hierarchical Clustering: (Dis)similarity between clusters

- We know how to compute the dissimilarity  $d(\mathbf{x}_i, \mathbf{x}_j)$  between two examples
- How to compute the dissimilarity between two clusters  $R$  and  $S$ ?
- **Min-link or single-link:** results in chaining (clusters can get very large)

$$d(R, S) = \min_{\mathbf{x}_R \in R, \mathbf{x}_S \in S} d(\mathbf{x}_R, \mathbf{x}_S)$$

- **Max-link or complete-link:** results in small, round shaped clusters

$$d(R, S) = \max_{\mathbf{x}_R \in R, \mathbf{x}_S \in S} d(\mathbf{x}_R, \mathbf{x}_S)$$

- **Average-link:** compromise between single and complete linkage

$$d(R, S) = \frac{1}{|R||S|} \sum_{\mathbf{x}_R \in R, \mathbf{x}_S \in S} d(\mathbf{x}_R, \mathbf{x}_S)$$

# Hierarchical Clustering: (Dis)similarity between clusters

- We know how to compute the dissimilarity  $d(\mathbf{x}_i, \mathbf{x}_j)$  between two examples
- How to compute the dissimilarity between two clusters  $R$  and  $S$ ?
- **Min-link or single-link:** results in chaining (clusters can get very large)

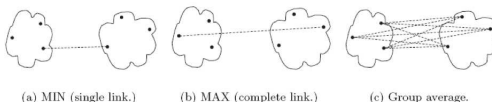
$$d(R, S) = \min_{\mathbf{x}_R \in R, \mathbf{x}_S \in S} d(\mathbf{x}_R, \mathbf{x}_S)$$

- **Max-link or complete-link:** results in small, round shaped clusters

$$d(R, S) = \max_{\mathbf{x}_R \in R, \mathbf{x}_S \in S} d(\mathbf{x}_R, \mathbf{x}_S)$$

- **Average-link:** compromise between single and complete linkage

$$d(R, S) = \frac{1}{|R||S|} \sum_{\mathbf{x}_R \in R, \mathbf{x}_S \in S} d(\mathbf{x}_R, \mathbf{x}_S)$$



# Flat vs Hierarchical Clustering

- Flat clustering produces a single partitioning
- Hierarchical Clustering can give different partitionings depending on the level-of-resolution we are looking at
- Flat clustering needs the number of clusters to be specified
- Hierarchical clustering doesn't need the number of clusters to be specified
- Flat clustering is usually more efficient run-time wise
- Hierarchical clustering can be slow (has to make several merge/split decisions)
- No clear consensus on which of the two produces better clustering