

# CLUSTERING

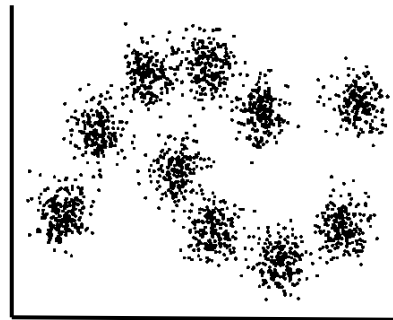
## Today

- Reading
  - Introduction to Information Retrieval (IR) Ch. 16, 17
  
- Goals
  - Flat clustering algorithms
    - K-means
  - Hierarchical clustering algorithms
    - Agglomerative clustering
    - Divisive clustering (it's divisive!)
  - Evaluating clusters

## Unsupervised Learning

- Learning without labels
  - Often comes down to clustering
  - Can be used as a surrogate for supervised learning

$$D = \begin{matrix} \mathbf{x}_1 & x_{11} & x_{12} & x_{13} & \cdots & x_{1M} \\ \mathbf{x}_2 & x_{21} & x_{22} & x_{23} & \cdots & x_{2M} \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ \mathbf{x}_N & x_{N1} & x_{N2} & x_{N3} & \cdots & x_{NM} \end{matrix}$$



## Clustering

- Grouping data into (hopefully useful) sets
- Why do clustering?
  - Labeling is costly
  - Data pre-processing
    - Text Classification (e.g., search engines, Google Sets)
  - Hypothesis Generation/Data Understanding
    - Clusters might suggest natural groups.
  - Visualization

## Terminology

- An **m-clustering** of  $D$  is a partition of  $D$  into sets (clusters)  $C_1, C_2, \dots, C_m$  such that
  - The clusters are non-empty
  - The union of the clusters is  $D$
  - The intersection of the clusters is empty
  
- The **centroid** of a cluster is the mean of all the elements in the cluster

## How many possible clusterings?

- Stirling number of the second kind
  - $n$  – size of dataset
  - $m$  – number of clusters

$$S(n, m) = \frac{1}{m!} \sum_{i=0}^m (-1)^{m-i} \binom{m}{i} i^n$$

$$S(15, 3) = 2,375,101$$

$$S(20, 4) = 45,232,115,901$$

$$S(100, 5) \approx 10^{68}$$

## How many possible clusterings?

- We can't try all possible clusterings.
- Clustering algorithms look at a small fraction of all partitions of the data.
- The exact partitions tried depend on the kind of clustering used.

$$S(n, m) = \frac{1}{m!} \sum_{i=0}^m (-1)^{m-i} \binom{m}{i} i^n$$

## Types of clustering algorithms

- Flat versus Hierarchical
  - ▣ Flat algorithms return an unstructured set of clusters
  - ▣ Hierarchical algorithms return a hierarchy of clusters
- Sequential (online) versus Batch
  - ▣ Sequential algorithms are typically fast
- Hard versus soft
  - ▣ Hard algorithms make a hard assignment of elements to clusters
  - ▣ Soft algorithms compute a distribution over clusters for each element

## K-means Clustering

- The most well-known (widely-used) clustering algorithm

“If someone is found slain, lying in a field in the land the LORD your God is giving you to possess, and it is not known who the killer was, your elders and judges shall go out and measure the distance from the body to the neighboring towns. Then the elders of the town nearest the body shall...”(Deuteronomy 21)

- User must set the number of clusters (K)
- Assumes instances  $x$  represented as normalized vectors in a real-valued space

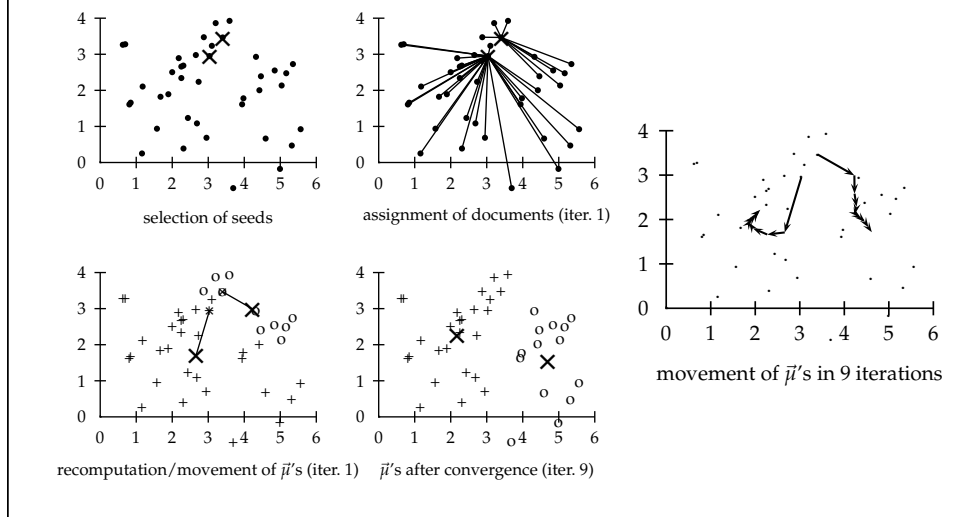
## K-means Clustering

```

K-MEANS( $\{\vec{x}_1, \dots, \vec{x}_N\}, K$ )
1  ( $\vec{s}_1, \vec{s}_2, \dots, \vec{s}_K$ )  $\leftarrow$  SELECTRANDOMSEEDS( $\{\vec{x}_1, \dots, \vec{x}_N\}, K$ )
2  for  $k \leftarrow 1$  to  $K$ 
3  do  $\vec{\mu}_k \leftarrow \vec{s}_k$ 
4  while stopping criterion has not been met
5  do for  $k \leftarrow 1$  to  $K$ 
6    do  $\omega_k \leftarrow \{\}$ 
7    for  $n \leftarrow 1$  to  $N$ 
8    do  $j \leftarrow \arg \min_j |\vec{\mu}_j - \vec{x}_n|$ 
9     $\omega_j \leftarrow \omega_j \cup \{\vec{x}_n\}$  (reassignment of vectors)
10   for  $k \leftarrow 1$  to  $K$ 
11   do  $\vec{\mu}_k \leftarrow \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}$  (recomputation of centroids)
12  return  $\{\vec{\mu}_1, \dots, \vec{\mu}_K\}$ 

```

## K-means Clustering



## K-means Clustering

- **Starting seeds (centroids)**
  - ▣ Randomly select K initial data points to be the centroids
  - ▣ Run multiple K-means each with different random seeds
  - ▣ Use results from a different clustering algorithm (run on random subset of data)
- **Stopping criteria**
  - ▣ See IR for a number of possible stopping criteria. For example:
  - ▣ Fixed number of iterations
  - ▣ Residual sum of squares (RSS) falls below threshold
- **Choosing K**
  - ▣ Rules-of-thumb or experience
  - ▣ Try multiple values for K and plot RSS. Look for the elbow.
  - ▣ Add regularization term

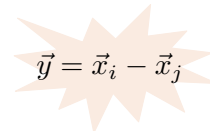
## Flat versus Hierarchical

- K-means
  - ▣ Returns unstructured set of clusters
  - ▣ Requires user to determine K
  - ▣ Non-deterministic
  - ▣ Linear run time  $O(KNM)$
  
- Hierarchical (e.g. Agglomerative clustering)
  - ▣ Returns a hierarchy of clusters
  - ▣ No need to (initially) determine K
  - ▣ Deterministic
  - ▣ Quadratic run time

## Hierarchical Clustering

- Cluster data points based on **distance metric** or **similarity measure**
- A **norm** is a function that assigns a positive real-valued number (representing length or size) with every vector in a vector space
- The **p-norm** ( $L_p$ ) of a vector  $y$  in  $R^M$  is given by

$$\|\vec{y}\|_p = \left( \sum_{m=1}^M |y_m|^p \right)^{1/p}$$



$$\vec{y} = \vec{x}_i - \vec{x}_j$$

- A **metric** is a function that assigns a positive real-valued number (representing distance) to every pair of vectors in some vector space
- **Minkowski distance** is given by

$$d_p(\vec{x}_i, \vec{x}_j) = \left( \sum_{m=1}^M |x_{im} - x_{jm}|^p \right)^{1/p}$$

## Hierarchical Clustering

- **Minkowski distance** is given by  $d_p(\vec{x}_i, \vec{x}_j) = \left( \sum_{m=1}^M |x_{im} - x_{jm}|^p \right)^{1/p}$
- For  $p = 1$ , Manhattan distance  $d_1(\vec{x}_i, \vec{x}_j) = \sum_{m=1}^M |x_{im} - x_{jm}|$
- For  $p = 2$ , Euclidean distance  $d_2(\vec{x}_i, \vec{x}_j) = \left( \sum_{m=1}^M |x_{im} - x_{jm}|^2 \right)^{1/2}$
- **Cosine similarity** also common measure (Note inverse of distance)

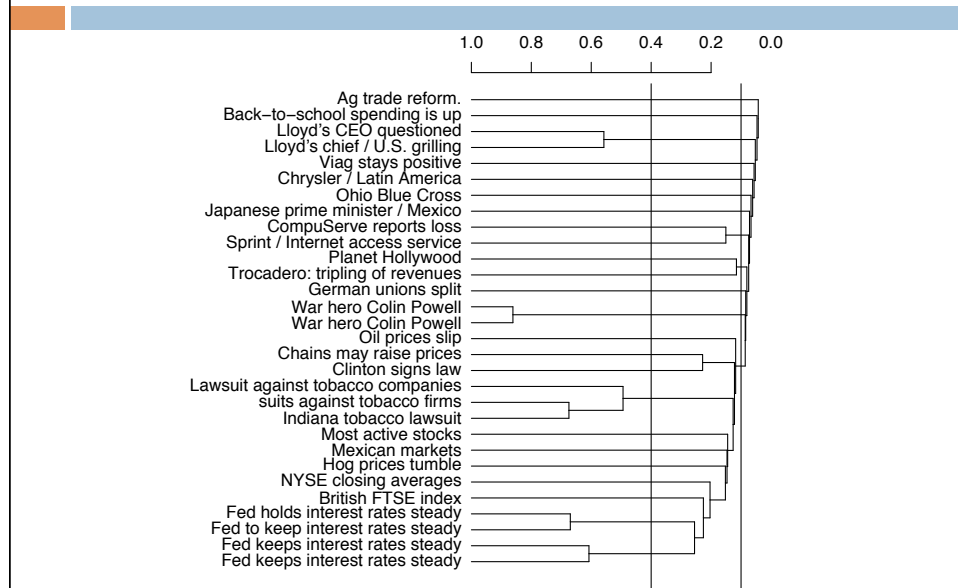
$$\cos(\vec{x}_i, \vec{x}_j) = \frac{\vec{x}_i^T \vec{x}_j}{\|\vec{x}_i\|_2 \|\vec{x}_j\|_2} = \frac{\sum_{m=1}^M x_{im} \cdot x_{jm}}{\|\vec{x}_i\|_2 \|\vec{x}_j\|_2}$$

## Hierarchical Clustering

- **Agglomerative clustering**
  - ▣ Start with  $N$  clusters each with one data point
  - ▣ Merge similar clusters to form larger clusters until there is only a single cluster left
- **Divisive Clustering**
  - ▣ Start with a single cluster containing all data points
  - ▣ Divide large clusters into smaller clusters until each cluster contains a single data point



## Agglomerative Clustering



## Agglomerative Clustering

```

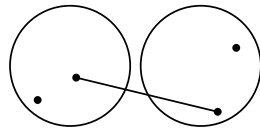
SIMPLEHAC( $d_1, \dots, d_N$ )
1  for  $n \leftarrow 1$  to  $N$ 
2  do for  $i \leftarrow 1$  to  $N$ 
3    do  $C[n][i] \leftarrow \text{SIM}(d_n, d_i)$ 
4     $I[n] \leftarrow 1$  (keeps track of active clusters)
5   $A \leftarrow []$  (assembles clustering as a sequence of merges)
6  for  $k \leftarrow 1$  to  $N - 1$ 
7  do  $\langle i, m \rangle \leftarrow \arg \max_{\{i, m\}: i \neq m \wedge I[i]=1 \wedge I[m]=1} C[i][m]$ 
8     $A.\text{APPEND}(\langle i, m \rangle)$  (store merge)
9  for  $j \leftarrow 1$  to  $N$ 
10 do  $C[i][j] \leftarrow \text{SIM}(i, m, j)$ 
11     $C[j][i] \leftarrow \text{SIM}(i, m, j)$ 
12     $I[m] \leftarrow 0$  (deactivate cluster)
13 return  $A$ 

```

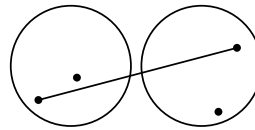
Compute similarity matrix (lines 3-4)  
 Find two clusters to merge (line 7)  
 The  $i^{\text{th}}$  row and column of  $C$  are now the distances for the new cluster (lines 10-11)  
 Similarity between cluster  $c_j$  and cluster  $c_i \cup c_m$  (lines 10-11)

► Figure 17.2 A simple, but inefficient HAC algorithm.

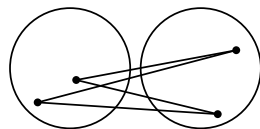
## Agglomerative Clustering



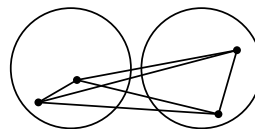
(a) single-link: maximum similarity



(b) complete-link: minimum similarity



(c) centroid: average inter-similarity



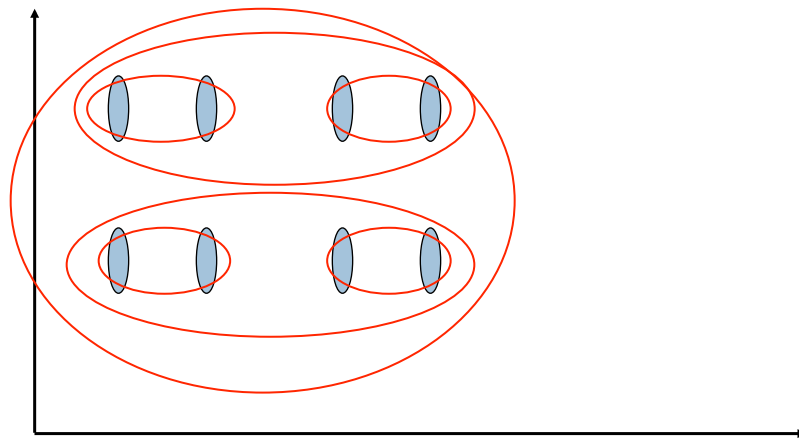
(d) group-average: average of all similarities

## Cluster similarity: Single-link

- Single link
  - ▣ Similarity of  $c_i$  and  $c_i \cup c_m$  is the similarity of their *most similar* members
  - ▣ Can result in unwanted “long” clusters due to chaining

$$\text{sim}((c_i \cup c_m), c_j) = \max(\text{sim}(c_i, c_j), \text{sim}(c_m, c_j))$$

## Cluster similarity: Single-link

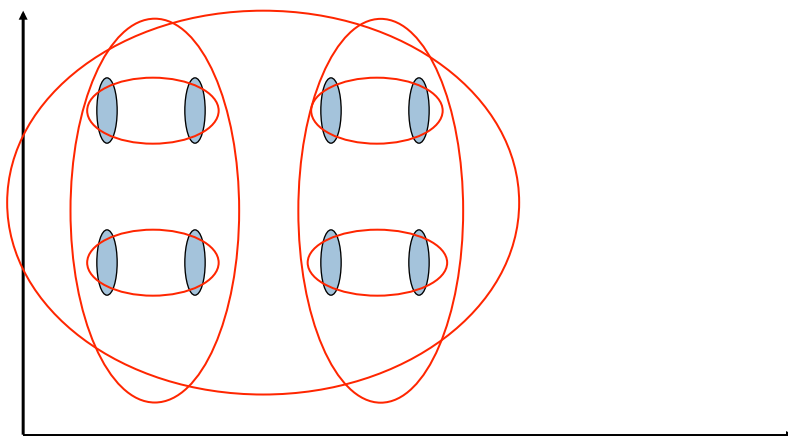


## Cluster similarity: Complete-link

- Complete link
  - ▣ Similarity of  $c_i$  and  $c_i \cup c_m$  is the similarity of their least similar members
  - ▣ Makes “tighter” spherical clusters that are typically preferable.
  - ▣ Sensitive to outliers

$$\text{sim}((c_i \cup c_m), c_j) = \min(\text{sim}(c_i, c_j), \text{sim}(c_m, c_j))$$

## Cluster similarity: Complete-link



## Cluster similarity: Group-average

- **Group-average (average-link)**
  - ▣ Uses all vectors in clusters  $c_j$  and  $c_i \cup c_m$  to compute similarity
  - ▣ Average similarity between all pairs of vectors from  $c_j$  and  $c_i \cup c_m$  (including pairs from same cluster)
  - ▣ Efficient computing of the group-average can be done if using cosine similarity

$$\text{sim}(c_i, c_j) = \frac{1}{\underbrace{(|c_i| + |c_j|)}_{\text{Total number of pairs}} \underbrace{(|c_i| + |c_j| - 1)}_{\text{All pairs of distinct vectors from } c_i \cup c_j}} \sum_{\substack{\vec{x}_n, \vec{x}_m \in c_i \cup c_j \\ \vec{x}_n \neq \vec{x}_m}} d(\vec{x}_n, \vec{x}_m)$$

## Cluster similarity: Centroid

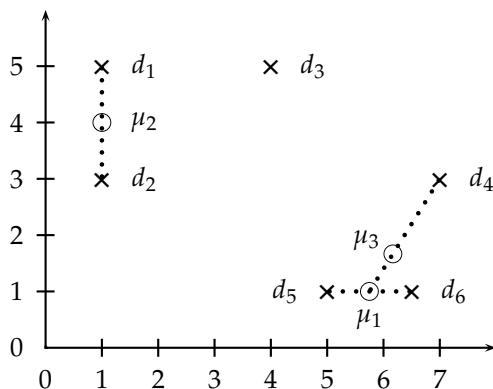
### □ Centroid clustering

- Similarity of cluster  $c_j$  and cluster  $c_i \cup c_m$  is the similarity of their centroids

$$\begin{aligned} \text{SIM-CENT}(\omega_i, \omega_j) &= \bar{\mu}(\omega_i) \cdot \bar{\mu}(\omega_j) \\ &= \left( \frac{1}{N_i} \sum_{d_m \in \omega_i} \vec{d}_m \right) \cdot \left( \frac{1}{N_j} \sum_{d_n \in \omega_j} \vec{d}_n \right) \\ &= \frac{1}{N_i N_j} \sum_{d_m \in \omega_i} \sum_{d_n \in \omega_j} \vec{d}_m \cdot \vec{d}_n \end{aligned}$$

- Equivalent to the average similarity of all pairs of documents from different clusters
- Similarity between clusters can increase as we merge clusters (known as inversions)
  - Horizontal merge lines can be lower than the previous merge line

## Cluster similarity: Centroid



## Computational Complexity

- In the first iteration, all HAC methods need to compute similarity of all pairs of  $N$  initial instances, which is  $O(N^2)$ .
- In each of the subsequent  $N-2$  merging iterations, compute the distance between the most recently created cluster and all other existing clusters.
- In order to maintain an overall  $O(N^2)$  performance, computing similarity to each other cluster must be done in constant time.
  - ▣ Often  $O(N^3)$  if done in a naïve way
  - ▣ or  $O(N^2 \log N)$  if done in a more clever way