Flat Text Clustering

K-means, DBSCAN

CHIEN CHIN CHEN

What is Text Clustering (1/4)

Text Clustering **groups** a set of documents into subsets or ***clusters***.

- Clusters are coherent internally, but different from each other.

In other words …

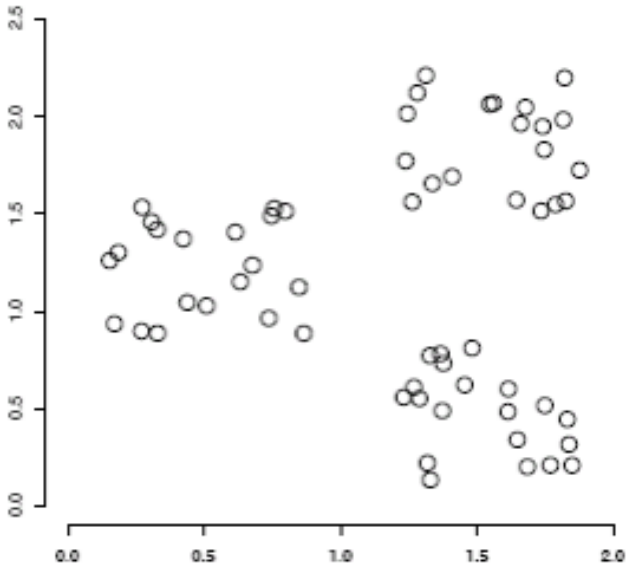
- Documents within a cluster should be as similar as possible.
- Documents in one cluster should be as dissimilar as possible from documents in other clusters.

Clustering is of ***unsupervised learning***.

- There is **no** human expert who has assigned documents to classes.
  - Or teach how to put documents in classes.
  - Totally different to text classification!!
- It is the **distribution and makeup** of the data that will determine cluster membership.

What is Text Clustering (2/4)

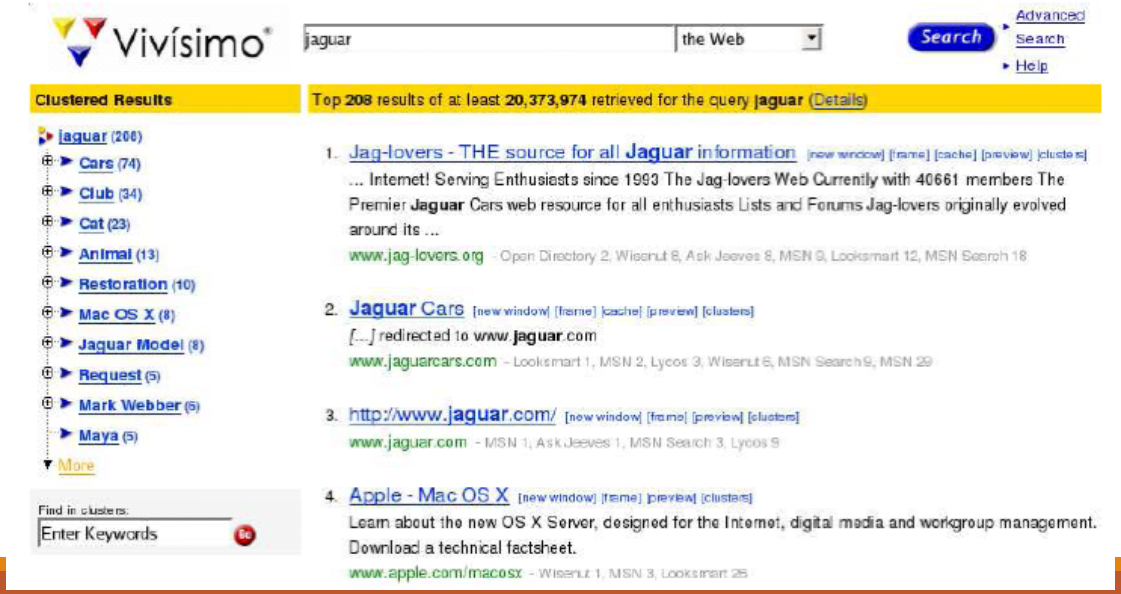
For example, there are three distinct clusters of points in the following figure.



Applications of Text Clustering (1/3)

Search result clustering:

- The default presentation of search results in information retrieval is simple list.
- It is often easier to scan a few coherent groups than many individual documents.
  - Particularly useful if a search term has **different word senses**.



Applications of Text Clustering (2/3)

News document clustering:

- News reading is not really search, but rather a process of selecting a subset of stories about recent events.
- Need to frequently **re-compute** the clustering to make sure that users can access the latest breaking stories.

Google News: https://news.google.com/

Applications of Text Clustering (3/3)

Latent topic discovery:

- To learn topics in a set of documents.
  - A group of documents talking about a certain subject.
  - A group of words referring to a certain subject.

For instance, hotel reviews could contain aspects about: ***location***, ***service***, and ***cleanliness***.

- Topic ***location*** is constituted of "*MRT*", "*mall*", "*museum*" …

We talk about latent topic mining when learning ***LDA***.

Types of Clustering

Structure:

- ***Flat clustering***: create a **flat set of clusters** without any explicit structure that would relate clusters to each other.
- ***Hierarchical clustering***: create a **hierarchy of clusters**.

Hard/soft clustering:

- ***Hard clustering***: each document is a member of **exactly one cluster**, also called ***partitional clustering***.
- ***Soft clustering***: a document's assignment is a distribution over all clusters.

Exhaustive clustering:

- Each document **must** be assigned to a cluster.
- Non-exhaustive clustering – some document will be assigned to no cluster.

Let's start with hard-flat clustering

Hard Flat Clustering Definition

Given …

- A set of documents $D = \{d_1, \ldots, d_N\}$.
- A desired number of clusters K. (optional for some clustering algorithms)
- An ***objective function*** that evaluates the quality of a clustering.
  - For instance, the **average distance between documents and their centroid**.

We want …

- Compute an **assignment** $\gamma : D \rightarrow \{1, \ldots, K\}$ that **minimizes** (or **maximizes**) the objective function.
- We usually represent document as term vectors; less distance means more content similar.

Usually, we require that none of the K clusters is empty.

Cardinality – The Number of Clusters

A difficult issue in clustering is determining the cardinality of a clustering (i.e., K).

The brute force solution would be to enumerate all possible clustering and pick the best.

- However, there are **exponentially** many partitions, so is **not feasible**.
- $1 + 2^N + 3^N + \ldots$

Often K is nothing more than a **good guess** based on experience or domain knowledge.

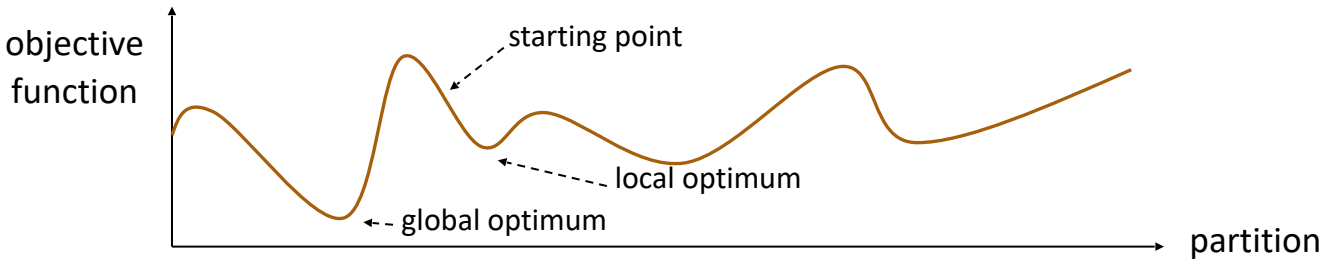
Starting Point

Clustering is essentially a search problem.

- To enumerate all possible clustering and pick the best (brute force solution) is not feasible.

Most flat clustering algorithms (e.g., *K*-means) begin with an **initial partition** (clustering result) and refine the partition **iteratively**.

If the search starts at an unfavorable initial point, we may miss the global optimum.



Finding a good starting point is another important problem in flat clustering.

Evaluation of Clustering (1/8)

In practice, It is very difficult to evaluate a clustering result.

- We cluster data because we have no ideal of what data look like.
- Some would use *(intra-cluster similarity) – (inter-cluster similarity)* to evaluate a clustering result. `
  - Note that a good score does not necessarily translate into good effectiveness in an application.
  - Some topics are very scattered.

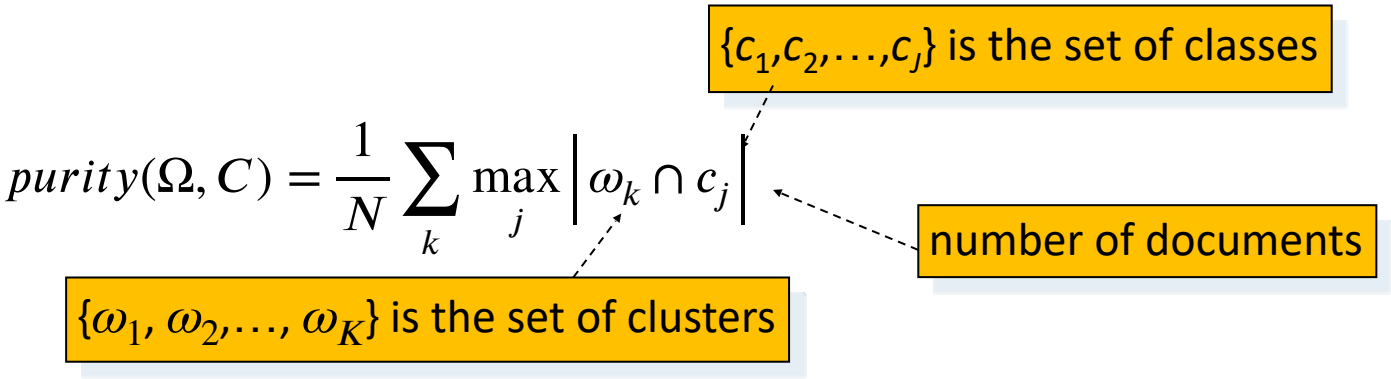
Regarding clustering research

- Benchmark data with human judges are used to evaluate how well the clustering matches the **gold standard classes**.
- Criteria of clustering quality: ***purity***, ***rand index***, ***precision***, ***recall***, and ***F1***.

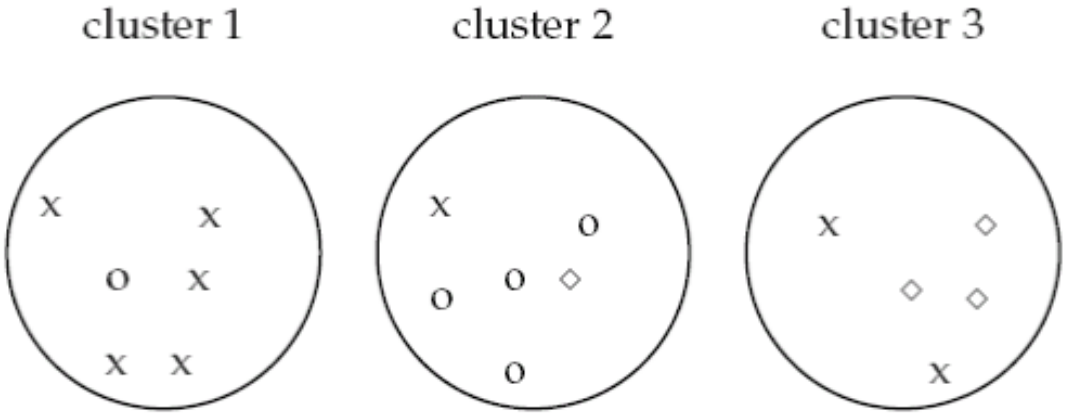
Evaluation of Clustering (2/8)

Purity:

- Each cluster is assigned to the class which is most frequent in the cluster.
- Then measure the **accuracy** of the assignment.



Evaluation of Clustering (3/8)



► **Figure 16.4** Purity as an external evaluation criterion for cluster quality. Majority class and number of members of the majority class for the three clusters are: x, 5 (cluster 1); o, 4 (cluster 2); and ◇, 3 (cluster 3). Purity is $(1/17) \times (5 + 4 + 3) \approx 0.71$.

Bias - high purity is easy to achieve when the number of cluster is large.

- 1 if each document gets its own cluster.

Evaluation of Clustering (4/8)

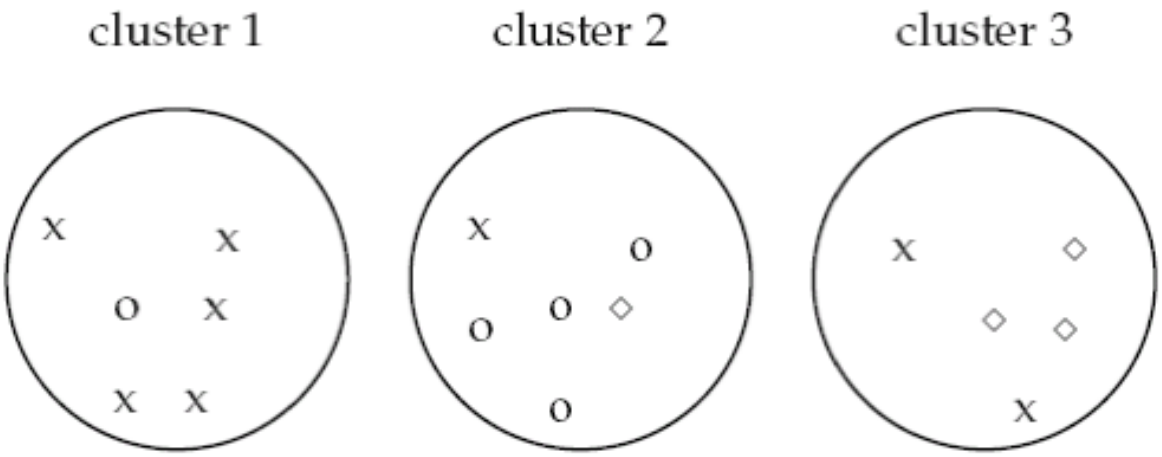
We can view clustering as a series of decision, one for each of the $N(N\text{-}1)/2$ pairs of documents in the collection.

- ***True positive*** (TP): assign two similar (class) documents to the same cluster.
- ***True negative*** (TN): assign two dissimilar (different-class) documents to different clusters.
- ***False positive*** (FP): assign two dissimilar documents to the same cluster.
- ***False negative*** (FN): assign two similar documents to different clusters.

Evaluation of Clustering (5/8)

The ***rand index*** (RI) measures the percentage of decisions that are correct.

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$



TP+FP = $C^6_2+C^6_2+C^5_2$ =40

TP = $C^5_2+C^4_2+C^3_2+C^2_2$ =20

Evaluation of Clustering (6/8)

	Same cluster	Different clusters
Same class	TP = 20	FN = 24
Different classes	FP = 20	TN = 72

RI is then (20 + 72) / (20 + 20 + 24 + 72) ≈ 0.68.

F measure:

$$Precison = \frac{TP}{TP + NP} \qquad Recall = \frac{TP}{TP + FN} \qquad F_1 = \frac{2PR}{P + R}$$

Evaluation of Clustering (7/8)

A concern of rand index is that the score of a random clustering may not be close to 0.

Adjusted rand index considers the expected agreements to make the score of random clustering close to 0.

- Range within [1, -1]: 1 for perfect match; -1 (rare) means the clustering is worse than random.

Evaluation of Clustering (8/8)

Let N be the number of documents to be clustered;

$C = \{c_1, c_2, \ldots, c_J\}$ is the set of classes (ground truth);

$\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$ is the set of clusters (clustering result).

			...		
c_1	n_{11}	n_{12}	...	n_{1K}	a_1
c_2	n_{21}	n_{22}	...	n_{2K}	a_2
...	...	...	...	...	...
c_J	n_{J1}	n_{J2}	...	n_{JK}	a_J
	b_1	b_2	...	b_K	

$$ARI = \frac{\sum_{jk} C_2^{n_{jk}} - (\sum_{j=1}^{J} C_2^{a_j} \sum_{k=1}^{K} C_2^{b_k}) / C_2^N}{\frac{1}{2}(\sum_{j=1}^{J} C_2^{a_j} + \sum_{k=1}^{K} C_2^{b_k}) - (\sum_{j=1}^{J} C_2^{a_j} \sum_{k=1}^{K} C_2^{b_k}) / C_2^N}$$

K-means (1/5)

The most popular flat clustering algorithm.

Its objective is to minimize the average squared Euclidean distance of documents from their ***cluster center***.

- A cluster center is defined as the **mean** or **centroid** $\vec{\mu}$ of the documents in a cluster ω:

$$\vec{\mu}(\omega) = \frac{1}{|\omega|} \sum_{\vec{x} \in \omega} \vec{x}$$

document vector

centroid vector of cluster ω

size of ω

K-means (2/5)

How well the centroid**s** represent the members of their clusters:

- ***Residual sum of squares*** (**RSS**) – the **squared distance** of each vector from its centroid, summed over all vectors:

$$RSS_k = \sum_{\vec{x} \in \omega_k} \left| \vec{x} - \vec{\mu}(\omega_k) \right|^2$$

for a certain cluster *k*

$$RSS = \sum_{k=1}^{K} RSS_k$$

RSS is the **objective function** in *K*-means and our goal is to minimize it.

- Since *N* is fixed, minimizing RSS is equivalent to minimizing the average squared distance.

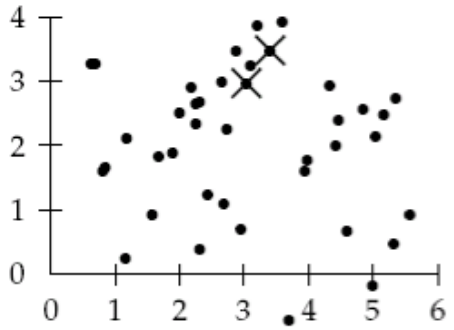
K-means (3/5)

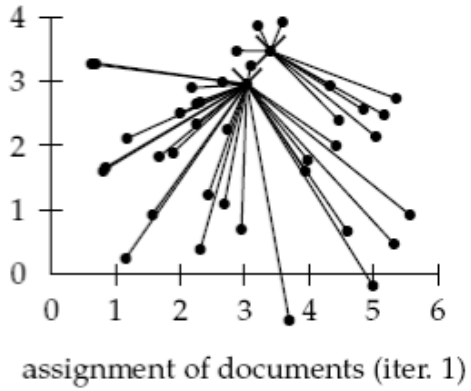
The first step of *K*-means is to randomly select *K* documents, as the **seeds** (or initial cluster centroids).

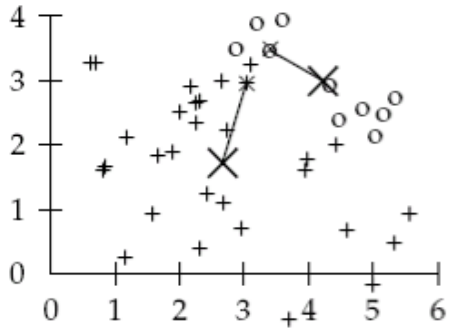
It then **iteratively** repeats **two steps** until **a stopping criterion** is met.

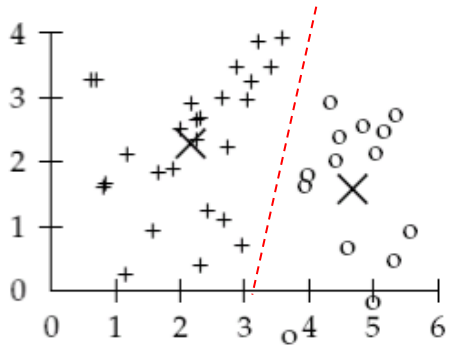
1. **Re-assigning** documents to the cluster with the closest centroid.
2. **Re-computing** each centroid based on the current members of its cluster.

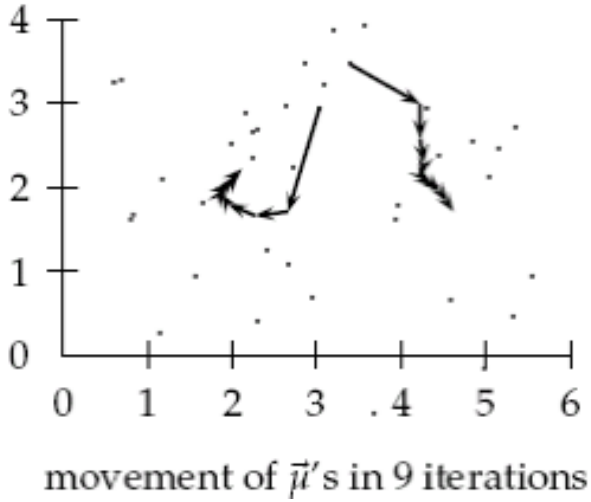
K-means (4/5)


selection of seeds


assignment of documents (iter. 1)


recomputation/movement of $\vec{\mu}$'s (iter. 1)


$\vec{\mu}$'s after convergence (iter. 9)


movement of $\vec{\mu}$'s in 9 iterations

K-means (5/5)

Termination conditions:

- **A fixed number of iterations** has been completed.
  - Limit the runtime of the clustering algorithm.
  - In some cases, the quality of the clustering will be poor because of an insufficient number of iterations.
- Assignment of documents to clusters does **not change between iterations**.
  - Equivalent to "centroids do not change between iterations".
  - Generally produce a good clustering.
  - But runtime may be unacceptably long.
- Terminate when RSS (or decrease in RSS) falls below a threshold.

Issues with *K*-means (1/4)

Seed selection: *K*-means is fast in clustering documents; but its outcome heavily depends on the seed initialization.

If an ***outlier*** is chosen as an initial seed, then no other vector is assigned to it during subsequent iterations.

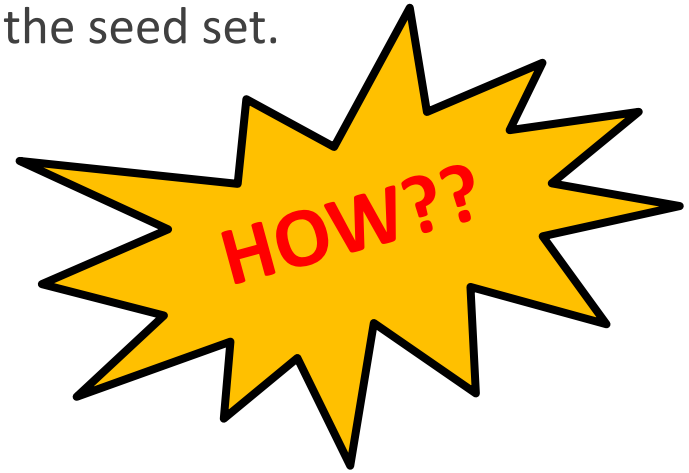
- ***Outlier***: A document that is far from any other documents.
- Thus, we end up with a ***singleton cluster*** (a cluster with only one document).
- There is probably a clustering with lower RSS (a better local minimum).

Issues with *K*-means (2/4)

Conversely, if seeds are too close, more iterations are needed to separate data.

Effective heuristics for seed selection:

- Trying out multiple starting points and choosing the clustering with lowest cost.
- Excluding outliers from the seed set.



Issues with *K*-means (3/4)

- Selecting **dis-similar seeds**:
  - Randomly select the **first seed** from the data.
  - For each data point, compute its distance from the nearest, previously chosen seed.
  - Select the **next seed** from the remaining data points according to a distribution that the point having maximum distance is most likely to be chosen.
  - Repeat the above two steps until *K* seeds have been selected.

This is what we called **K-means++**

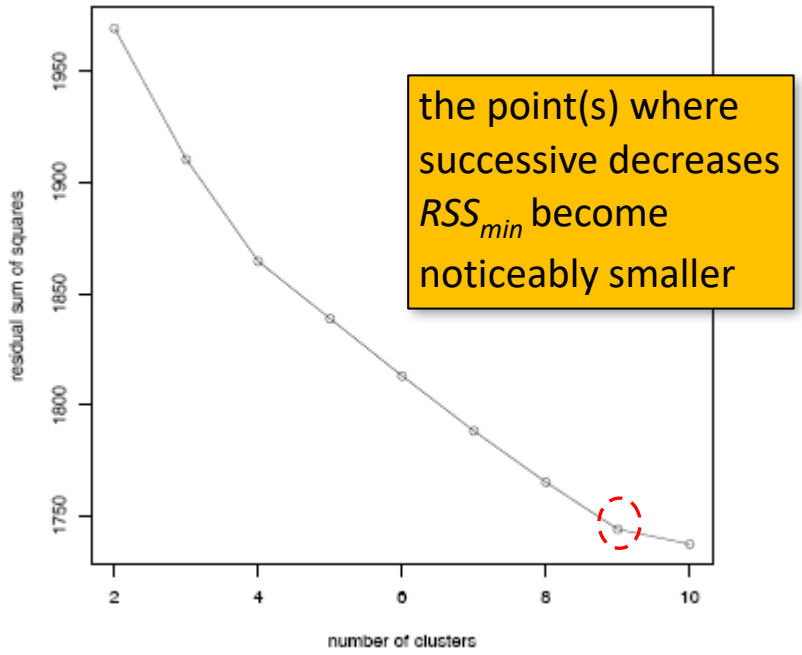
Issues with *K*-means (4/4)

Cluster Cardinality – the number of clusters *K*:

- What do we do if we cannot come up with a plausible guess for *K*?

The elbow method:

- Examine the RSS values within a range of acceptable of *K*s.
- Perform *i* (e.g., *i* = 10) clusterings with *K* clusters (each with a different initialization).
  - Denote $RSS_{min}(K)$ the minimum of the *i* RSS values.
- We inspect the values $RSS_{min}(K)$ as *K* increases and find the ***knee(s)*** in the curve.

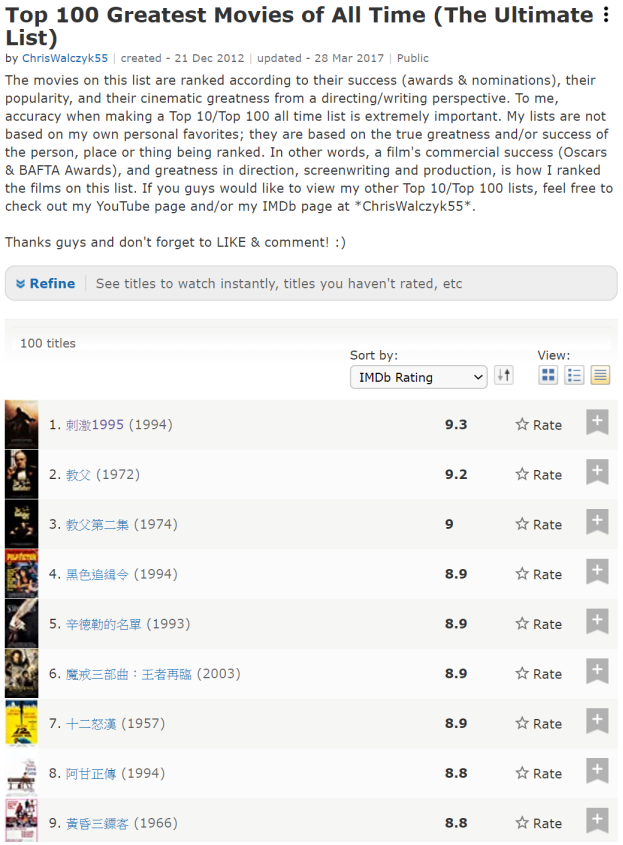


Let's practice (1/7)

In this practice, we cluster movie synopses.

- We save the top 70 in the **imdb_70.xlsx** file.

	A	B	C	D	E
1	title	synopsis	genre	genre_id	
2	The Shawsh	In 1947, Andy	Drama	0	
3	The Godfath	In late summ	Crime	1	
4	The Godfath	The Godfath	Crime	1	
5	Pulp Fiction	Late one mor	Crime	1	
6	Schindler's L	The relocatic	Biography	2	
7	The Lord of t	In the openin	Action	3	
8	12 Angry Me	In a New Yor	Crime	1	
9	Forrest Gum	The film begi	Drama	0	
10	The Good, th	The film tells	Western	4	
11	One Flew Ov	In 1963 Oreg	Drama	0	
12	Goodfellas	The film oper	Biography	2	
13	The Silence	Promising FE	Crime	1	
14	The Green M	The movie of	Crime	1	
15	Saving Priva	An American	Drama	0	
16	Star Wars: E	An opening ti	Action	3	
17	It's a Wonder	This movie is	Drama	0	
18	Gladiator	Shouting "Rc	Action	3	
19	Psycho	In a Phoenix	Horror	5	
20	Casablanca	In the early y	Drama	0	
21	The Pianist	"The Pianist"	Biography	2	
22	City Lights	The officials	Comedy	6	
23	Apocalypse N	The story op	Drama	0	
24	Raiders of th	In the spring	Action	3	
25	Rear Window	L.B. "Jeff" Je	Mystery	7	



https://www.imdb.com/list/ls055592025/

Let’s practice (2/7)

Load data from xlsx file

In [1]:
```
import pandas as pd
```

In [25]:
```
dataframe = pd.read_excel('./imdb_70.xlsx')
dataframe.head(5)
```

Out[25]:

	title	synopsis	genre	genre_id
0	The Shawshank Redemption	In 1947, Andy Dufresne (Tim Robbins), a banker...	Drama	0
1	The Godfather	In late summer 1945, guests are gathered for t...	Crime	1
2	The Godfather: Part II	The Godfather Part II presents two parallel st...	Crime	1
3	Pulp Fiction	Late one morning in the Hawthorne Grill, a res...	Crime	1
4	Schindler's List	The relocation of Polish Jews from surrounding...	Biography	2

In [26]:
```
labels = dataframe['genre_id']
texts = dataframe['synopsis']
#print(texts[0])
```

Let's practice (3/7)

Convert text into vectors

- Since we only have 70 files, some unimportant terms that rarely appear in the corpus would have a large tfidf weight.
  - We filter rare terms by means of `min_df = 2`.
  - And remove stop words from text.

```
In [27]: from sklearn.feature_extraction.text import TfidfVectorizer

In [28]: TFIDF_vectorizer = TfidfVectorizer(min_df=2, stop_words='english')

In [29]: TFIDF_vectors = TFIDF_vectorizer.fit_transform(texts)

In [30]: TFIDF_vectors.shape
Out[30]: (70, 5458)
```

Let's practice (4/7)

Search for a good *K* using the **elbow method**.

- Examining K from 2 to 15.
- **`inertia_`** : Sum of squared distances of samples to their closest cluster center.

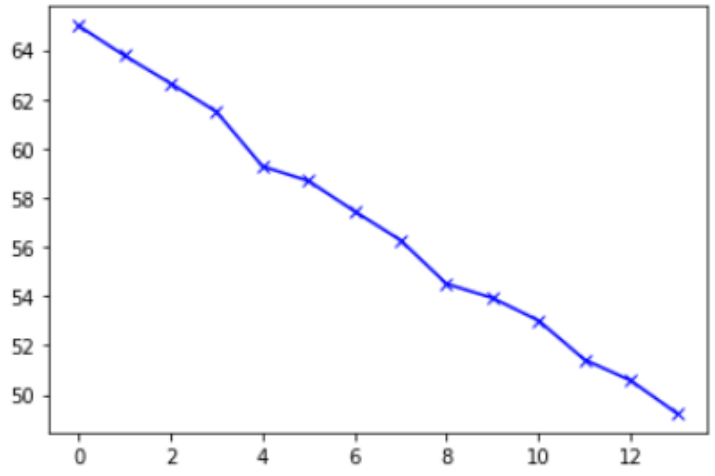
In this case, we set *K* at 6 and conduct the final clustering

```
In [8]: from sklearn.cluster import KMeans

In [31]: n_clusters = 16
         cost = []
         for i in range(2,n_clusters):
             kmeans = KMeans(n_clusters=i)
             kmeans.fit(TFIDF_vectors)
             cost.append(kmeans.inertia_)

In [32]: import matplotlib.pyplot as plt
         plt.plot(cost, 'bx-')

Out[32]: [<matplotlib.lines.Line2D at 0x7fd67fe6e7d0>]
```



Let's practice (5/7)

Conduct the final clustering

In [33]:
```
final_n_clusters = 6
final_kmeans = KMeans(final_n_clusters)
final_kmeans.fit(TFIDF_vectors)
```

Out[33]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=6, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

In [34]:
```
final_kmeans.labels_
```

Out[34]:
```
array([2, 3, 3, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 5,
       0, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 5, 2, 2, 2, 2, 1, 2, 2, 2, 1,
       2, 2, 2, 4, 2, 2, 2, 2, 2, 5, 1, 2, 4, 2, 4, 2, 2, 1, 2, 2, 2, 2,
       2, 2, 2, 2], dtype=int32)
```

Let's practice (6/7)

Measure the performance

```
In [35]: from sklearn import metrics

In [36]: predicted_labels = final_kmeans.labels_

In [37]: metrics.adjusted_rand_score(labels, predicted_labels)
Out[37]: 0.00943694538822599
```

It is so poor …

But you would understand if you take a look of the top-70 synopses (the topics are so diverse).

Let's practice (7/7)

Examine **the most representative keywords** of each cluster.

山提諾·桑尼·柯里昂

虛構人物

譯自英文 - 桑蒂諾·桑尼·科里昂（Santino" Sonny" Corleone）是馬里奧·普佐（Mario Puzo）1969年的小說《教父》及其1972年改編的電影中的虛構人物。他是黑手黨Don Vito Corleone和Carmela Corleone的長子。 維基百科（英文）

查看原文說明

弗雷多·柯里昂

虛構人物

譯自英文 - 弗雷多·柯里昂（Frederico Corleone）是馬里奧·普佐（Mario Puzo）1969年的小說《教父》中的虛構人物。

維基百科（英文）

海門・羅斯

虛構人物

譯自英文 - 海曼·羅斯（Hyman Roth）是1974年的電影《教父II》中的虛構人物和主要反對者。他還是2004年小說《教父歸來》中的次要人物。羅斯（Roth）是猶太人和投資人，也是維托·柯里昂（Vito Corleone）和後來的兒子邁克爾·柯里昂（Michael Corleone）的商業夥伴。 維基百科（英文）

```
In [42]: order_centroids = final_kmeans.cluster_centers_.argsort()[:,::-1]

In [43]: print("----  Top Terms of Each Cluster ----")
         for i in range(final_n_clusters):
             print("\n\nCluster %d keywords: " % i)
             for ind in order_centroids[i, :6]:
                 print(TFIDF_vectorizer.get_feature_names()[ind])
```

```
---- Top Terms of Each Cluster ----


Cluster 0 keywords:
marion
ark
jews
nazi
sam
factory


Cluster 1 keywords:
joe
sheldrake
music
sound
jerry
max


Cluster 2 keywords:
jake
tells
butch
man
father
time


Cluster 3 keywords:
michael
corleone
vito
fredo
sonny
roth
```

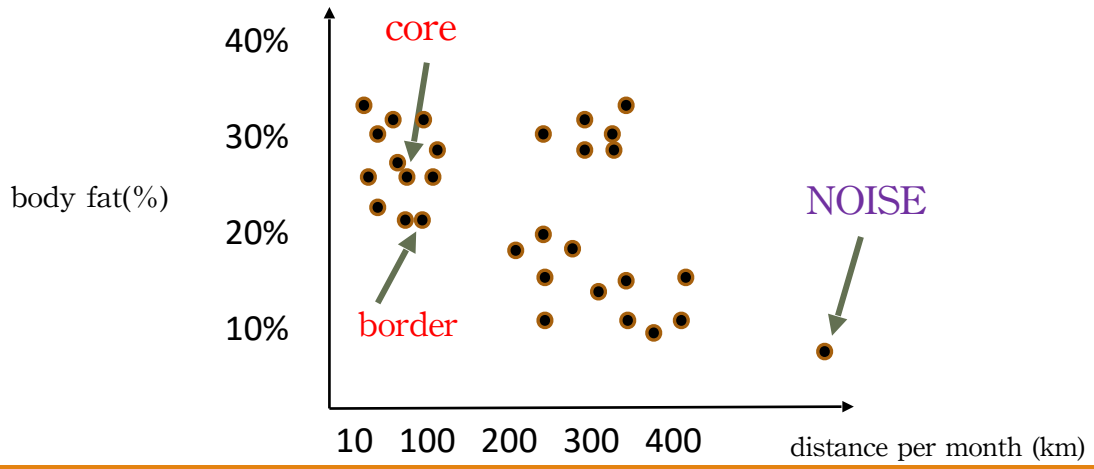
DBSCAN (1/3)

Density-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise.

- A well-know clustering algorithm of data mining

Intuitively, a cluster consists of data points (vectors) close to each other.

- Form a high **density area** in a high-dimensional vector space.
- There are two types of points in a cluster: **core** and **border**.



DBSCAN (2/3)

How does DBSCAN find clusters?

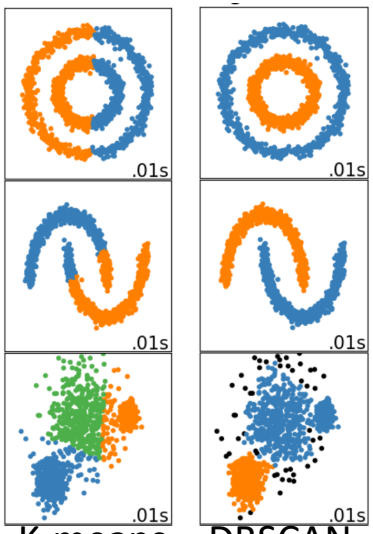
1. Decide the value of `eps` and `minPts`.
2. For each point `x`:
   - Calculate its distance from all other points. If the distance is less than or equal to `eps` then mark that point as a neighbor of `x`.
   - If the point `x` gets a neighboring count greater than or equal to `minPts`, then mark it as a **core point**.
3. For each core point, if it is not already assigned to a cluster than create a new cluster. Recursively find all its neighboring points and assign them the same cluster as the core point.
4. Continue these steps until all the unvisited points are covered.

Every data point not contained in any cluster is considered to be **noise (outlier)**.

DBSCAN (3/3)

Why DBSCAN?

- **No need to specify the number of clusters!!**
- Is able to detect outliers (noise data points).
- Can form **un-spherical** clusters.
  - The K-means algorithm assumes clusters are spherical!!



K-means DBSCAN

Disadvantage:

- Not good at handling data with varying density.
- difficult to determine `eps` and `minPts`.

Let's Practice

```
In [8]: from sklearn.cluster import DBSCAN
        dbscan = DBSCAN(metric='euclidean', eps=1.2, min_samples=2)
        dbscan.fit(TFIDF_vectors)

Out[8]: DBSCAN(algorithm='auto', eps=1.2, leaf_size=30, metric='euclidean',
               metric_params=None, min_samples=2, n_jobs=None, p=None)

In [9]: dbscan.labels_

Out[9]: array([-1,  0,  0,  1,  2, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
                3, -1,  2, -1, -1,  3, -1, -1,  4, -1, -1, -1, -1, -1, -1, -1, -1,
               -1, -1, -1, -1, -1, -1, -1,  5,  5,  4, -1, -1, -1, -1, -1,  4, -1,
               -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  1, -1,
               -1, -1])

In [10]: from collections import Counter
         print(Counter(dbscan.labels_))

         Counter({-1: 57, 4: 3, 0: 2, 1: 2, 2: 2, 3: 2, 5: 2})
```

noise data label

The two movies are clustered together

As mentioned before, it is difficult to set `eps` and `min_samples`

Many data are labeled as outliers

- Not the algorithm problem, but the synopses are too diverse…