Algorithmic Methods of Data Mining

Cluster Analysis

Ioannis Chatzigiannakis

Sapienza University of Rome

Laboratory 8

Internet news data with user engagement

How can we determine the popularity of an article before it is published online ?

- Internet news data was collected between 03.09.2019 and 04.11.2019.
- Articles listed as the top in popularity at the publisher website.
- Multiple well-known publishers.
- Using Facebook GraphAPI the data was enriched with engagement features such as shares, reactions, and comments count

`https://www.kaggle.com/szymonjanowski/internet-articles-data-with-users-engagement`

Internet news data with user engagement (1)

1. Row counter.
2. `Sourceid` – publisher unique identifier.
3. `Source_name` – publisher name.
4. `Author` – article author. Some publishers do not share information about authors of their news, in this case usually `source_name` replaces that information.
5. `Title` – headline of an article.
6. `Description` – short article description usually visible in popups or recommendation boxes on the publisher's website.
7. `Url` – URL of the publisher website.
8. `Urltoimage` – main image associated with the article.
9. `Published_at` – exact date and time of publishing the article in UTC (+000) time format.
10. `Content` – unformatted content of the article (max 260 char).

Internet news data with user engagement (2)

11. `Top_article` – 1 if article was listed as a top article on publisher website, otherwise 0.
12. `engagement_reaction_count` – counts user reactions on posts on Facebook involving article URL.
13. `engagement_comment_count` – number of comments posted on Facebook involving article URL.
14. `engagement_share_count` – number of time original post was shared by a use on Facebook involving article URL.
15. `engagement_comment_plugin_count` – number of comments made by users from an external site using their Facebook account.

```python
import modin.pandas as pd

data = pd.read_csv('articles_data.csv')

data.shape
(10437, 15)

data.describe()
```

```python
data.isnull().sum()

Unnamed: 0                          0
source_id                           0
source_name                         0
author                           1020
title                               2
description                        24
url                                 1
url_to_image                      656
published_at                        1
content                          1292
top_article                         2
engagement_reaction_count         118
engagement_comment_count          118
engagement_share_count            118
engagement_comment_plugin_count   118
dtype: int64
```

```python
data['author'].fillna('', inplace=True)
data['title'].fillna('', inplace=True)
data['description'].fillna('', inplace=True)
data['url'].fillna('', inplace=True)
data['url_to_image'].fillna('', inplace=True)
data['published_at'].fillna('2019-09-03T16:22:20Z',
                            inplace=True)
data['content'].fillna('', inplace=True)
data['top_article'].fillna(0, inplace=True)
data['engagement_reaction_count'].fillna(0,
                            inplace=True)
data['engagement_comment_count'].fillna(0,
                            inplace=True)
data['engagement_share_count'].fillna(0,
                            inplace=True)
data['engagement_comment_plugin_count'].fillna(0,
                            inplace=True)
```

```python
data['author'].value_counts().head(20)

data.loc[data.author == 'https://www.facebook.com/bbcnews']
        = 'BBC News'
```

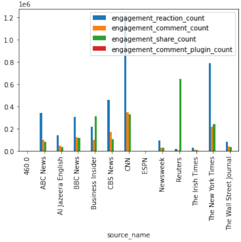
- Other data that needs to be cleaned?

```python
data['engagement_reaction_count'] =
  data['engagement_reaction_count']
  .apply(pd.to_numeric, errors='coerce')
```

- Other data that needs to be cleaned?

Engagement per publisher

```
data.groupby('source_name').sum().plot.bar()
```



Analysis of Data

1. Engagement per author
2. Articles collected every day
3. Distribution Of Engagement Reaction Counts

Text Mining

```
data['title'].sample(20)
```

- Remove stop words.
- Remove punctuations.
- Remove numbers.
- → Produce histogram of title lengths.
- Split title in vector of words.
- Convert to lower-case
- → Plot most used words.
- Analyze Description.

Analysis of Data

- Viewing and analyzing vast amounts of data in its unstructured entirety can be perplexing.
- It is easier to interpret data if it is organized into clusters that combine similar (i.e., related) data points.

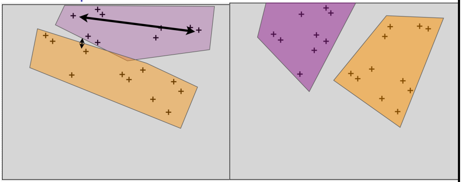
The Clustering Problem

- Motivation: Find patterns in a sea of data
- Input
  - A (large) number of datapoints: N
  - A measure of distance between any two data points d_{ij}
- Output
  - Groupings (clustering) of the elements into K (the number can be user-specified or automatically determined) 'similarity' classes
  - Sometimes there is also an objective measure that the obtained clustering seeks to minimize.

Clustering Principles

- Homogeneity – elements of the same cluster are maximally close to each other.
- Separation – elements in separate clusters are maximally far apart from each other.
- One is actually implied by the other (in many cases).
- Generally it is a hard problem.
  - Clustering in 2 dimensions looks easy
  - Clustering small amounts of data looks easy
  - High-dimensional spaces look different – Almost all pairs of points are at about the same distance

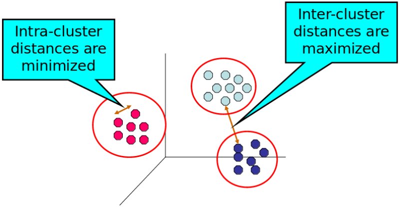
Some Examples



- Both principles are violated
- Points in the same cluster are far apart
- Points in different cluster are close

- More reasonable assignment.
- We need to use an objective function to optimize cluster assignment.

Intra/Inter Cluster Distances



- Suitably select distance metric.
- Maximize Inter-cluster distances.
- Minimize Intra-cluster distances.

Distance Measures

- Each clustering problem is based on some kind of "distance" between points.
- Two major classes of distance measure:
  1. Euclidean
  2. Non-Euclidean
- A Euclideanspace has some number of real-valued dimensions.
  - There is a notion of "average" of two points.
  - A Euclidean distance is based on the locations of points in such a space.
- A Non-Euclidean distance is based on properties of points, but not their "location" in a space.

Axioms of a Distance Measure

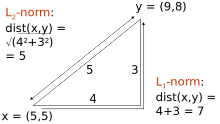
d is a distance measure if it is a function from pairs of points to real numbers such that:

1. $d(x, y) > 0$
2. $d(x, y) = 0$ *iff* $x = y$
3. $d(x, y) = d(y, x)$
4. $d(x, y) < d(x, z) + d(z, y)$ (triangle inequality)

Some Euclidean Distances

L_2 norm: $d(x, y)$ = square root of the sum of the squares of the differences between x and y in each dimension.
The most common notion of "distance".

L_1 norm: sum of the differences in each dimension.
Manhattan distance = distance if you had to travel along coordinates only.



Some Non-Euclidean Distances

Jaccard distance for sets = 1 minus ratio of sizes of intersection and union.

Cosine distance = angle between vectors from the origin to the points in question.

Edit distance = number of inserts and deletes to change one string into another.

Jaccard Distance for Sets

Example: $p_1 = 10111$; $p_2 = 10011$.
Size of intersection = 3; size of union = 4, Jaccard similarity (not distance) = $\frac{3}{4}$.
$d(x, y) = 1-$(Jaccard similarity) $= \frac{1}{4}$.

Why JD is a distance measure?

1. $d(x, x) = 0$ because $x \cap x = x \cup x$
2. $d(x, y) = d(y, x)$ because union and intersection are symmetric
3. $d(x, y) \geq 0$ because $|x \cap y| \leq |x \cup y|$
4. $d(x, y) < d(x, z) + d(z, y)$ more difficult...
   $\left(1 - \frac{|x \cap z|}{|x \cup z|}\right) + \left(1 - \frac{|y \cap z|}{|y \cup z|}\right) \geq 1 - \frac{|x \cap y|}{|x \cup y|}$

Edit Distance

The edit distance of two strings is the number of inserts and deletes of characters needed to turn one into the other. Equivalently:

$$d(x, y) = |x| + |y| - 2|LCS(x, y)|$$

LCS = longest common subsequence = any longest string obtained both by deleting from x and deleting from y.

Example

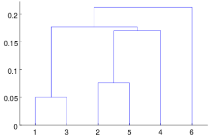
- x = abcde ; y = bcduve.
- Turn x into y by deleting a, then inserting u and v after d. Edit distance = 3.
- Or, LCS(x,y) = bcde.
- Note: $|x| + |y| - 2|LCS(x, y)| = 5 + 6 - 2 \times 4 = 3 =$ edit dist

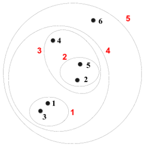
Why Edit Distance is a Distance Measure?

1. $d(x, x) = 0$ because 0 edits suffice.
2. $d(x, y) = d(y, x)$ because insert/delete are inverses of each other
3. $d(x, y) \geq 0$ no notion of negative edits
4. $d(x, y) < d(x, z) + d(z, y)$ Triangle inequality: changing x to z and then to y is one way to change x to y.

Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram – A tree like diagram that records the sequences of merges or splits





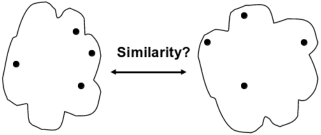
Agglomerative Hierarchical Clustering

- Initially, each point is a cluster
- Repeatedly combine the two "nearest" clusters into one

```
Compute the proximity matrix
Let each data point be a cluster
Repeat
        Merge the two closest clusters
        Update the proximity matrix
Until only a single cluster remains
```

- Key operation is the computation of the proximity of two clusters
- Different approaches to defining the distance between clusters distinguish the different algorithms

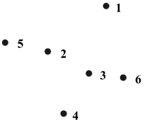
How to define Inter-cluster similarity?

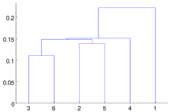


- Minimum – based on the two most similar (closest) points in the different clusters
- Maximum – based on the two least similar (most distant) points in the different clusters
- Group Average

Minimum – Example

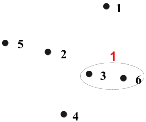
Minimum – based on the two most similar (closest) points in the different clusters

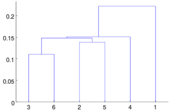




Minimum – Example

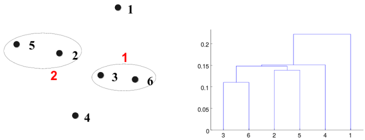
Minimum – based on the two most similar (closest) points in the different clusters





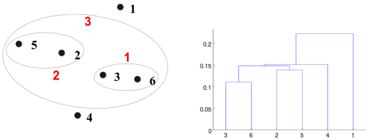
Minimum – Example

Minimum – based on the two most similar (closest) points in the different clusters



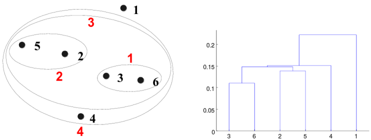
Minimum – Example

Minimum – based on the two most similar (closest) points in the different clusters



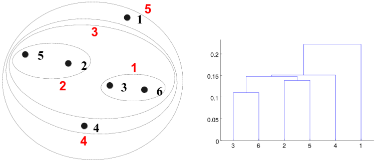
Minimum – Example

Minimum – based on the two most similar (closest) points in the different clusters

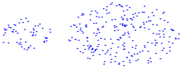


Minimum – Example

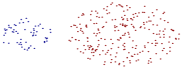
Minimum – based on the two most similar (closest) points in the different clusters



Minimum – Strength

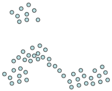


Original Points

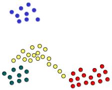


Two Clusters

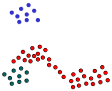
Minimum – Limitations



Original Points



Four clusters



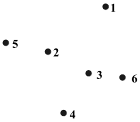
Three clusters:

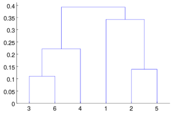
The yellow points got wrongly merged with the red ones, as opposed to the green one.

Sensitive to noise and outliers

Maximum – Example

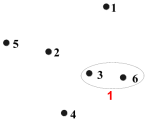
Maximum – based on the two least similar (most distant) points in the different clusters

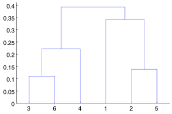




Maximum – Example

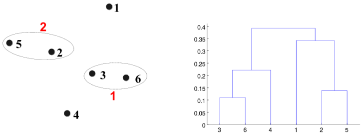
Maximum – based on the two least similar (most distant) points in the different clusters





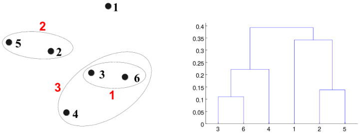
Maximum – Example

Maximum – based on the two least similar (most distant) points in the different clusters



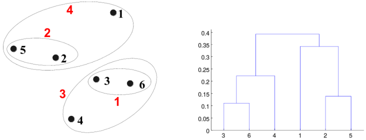
Maximum – Example

Maximum – based on the two least similar (most distant) points in the different clusters



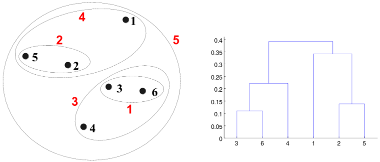
Maximum – Example

Maximum – based on the two least similar (most distant) points in the different clusters

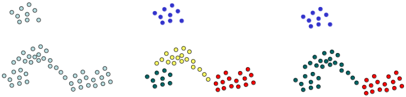


Maximum – Example

Maximum – based on the two least similar (most distant) points in the different clusters



Maximum – Strength

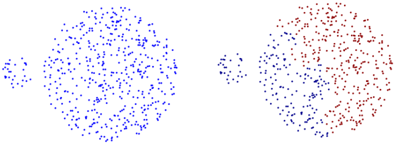


Original Points **Four clusters** **Three clusters:**

The yellow points get now merged with the green one.

Less susceptible respect to noise and outliers

Maximum – Limitations



Original Points **Two Clusters**

K-means Algorithm

- Developed and published in Applied Statistics by Hartigan and Wong, 1979.
- Many variations have been proposed since then.
- Standard/core function of R, Python, Matlab, ...
- Assumes Euclidean space/distance

The aim of the K-means algorithm is to divide M points in N dimensions into k clusters so that the within-cluster sum of squares is minimized.

$$\min{}_{C_1,\ldots,C_K} \sum_{k=1}^{k} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

Cluster Initialization

- Start by picking k, the number of clusters
- Initialize clusters by picking one point per cluster

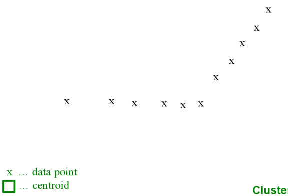
Example: Pick one point at random, then $k-1$ other points, each as far away as possible from the previous points

Populating Clusters

1. For each point, place it in the cluster whose current centroid it is nearest
2. After all points are assigned, update the locations of centroids of the k clusters
3. Reassign all points to their closest centroid
   - Sometimes moves points between clusters
4. Repeat 2 and 3 until convergence

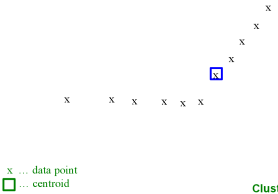
Convergence: Points do not move between clusters and centroids stabilize

A Simple Example



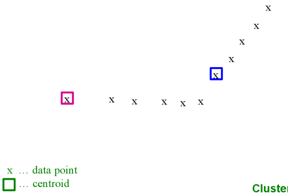
Clusters after round 1

A Simple Example



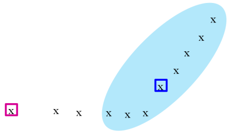
Clusters after round 1

A Simple Example



Clusters after round 1

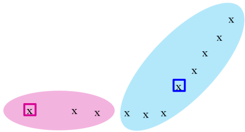
A Simple Example



x … data point
□ … centroid

Clusters after round 1

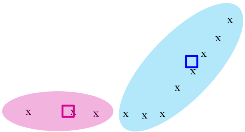
A Simple Example



x … data point
□ … centroid

Clusters after round 1

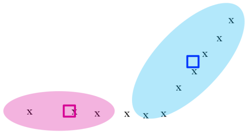
A Simple Example



x … data point
□ … centroid

Clusters after round 2

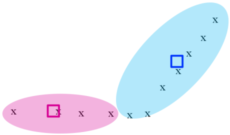
A Simple Example



x … data point
□ … centroid

Clusters after round 2

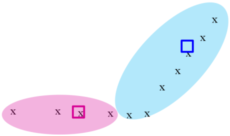
A Simple Example



x … data point
□ … centroid

Clusters after round 2

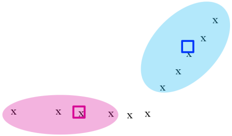
A Simple Example



x … data point
□ … centroid

Clusters at the end

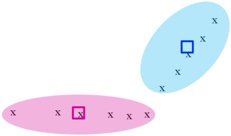
A Simple Example



x … data point
□ … centroid

Clusters at the end

A Simple Example

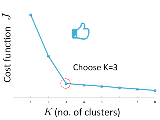


x … data point
□ … centroid

Clusters at the end

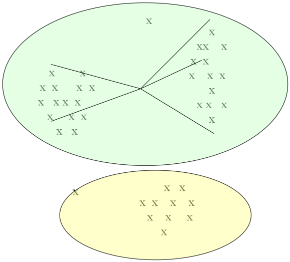
How to select k?

- We use the elbow method to determine the optimum number of clusters.
- Try different k, looking at the change in the average distance to centroid as k increases.
- Average falls rapidly until right k, then changes little.



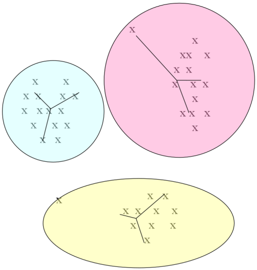
Selection of k – an example

Too few;
many long
distances
to centroid.



Selection of k – an example

Just right;
distances
rather short.



Selection of k – an example

Too many;
little improvement
in average
distance.