Information Clustering

Agenda

- Why Clustering?
- Definition of Similarity Measures
- Types of Clustering
- Application to Content Detection & Analysis

Why Clustering?

- Develop clusters so that
  - We have a *classification* of our data (dimension reduction)
  - We can *investigate* useful conceptual schemes for grouping entities within our data
  - *Hypothesis generation* through data exploration
  - *Hypothesis testing* or attempting to determine if types defined through other procedures are in fact present in the dataset (Aldenderfer and Blashfield, 1984)

Clustering is Subjective

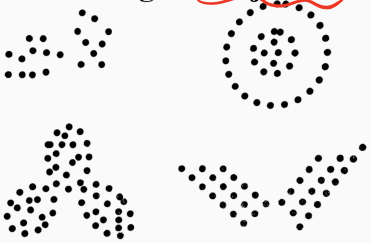
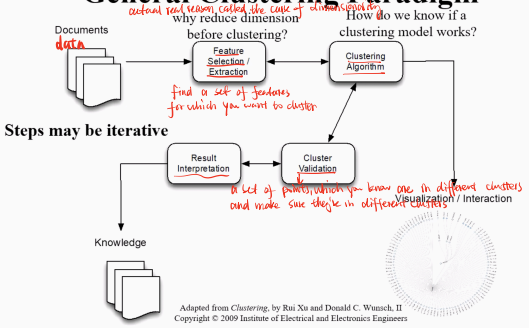


Fig. 1.1. Illustration of subjectivity of cluster analysis. Clustering at a coarse level produces four major clusters, while a finer clustering leads to nine clusters.

- *Clustering*, by Rui Xu and Donald C. Wunsch, II

General Clustering Paradigm



Role of Clustering in Content Analysis

- **Cleaning** - Removal of noise can be done at cluster level instead of document level.
  - Example: clustering can tell us languages in multi-language corpus
- **Analysis based on the category** - Clusters represents categories of documents. Each category can be treated separately for analysis
- **Error detection and generalization** - precisely identify the category of documents that are having trouble in analysis
- **Extraction of content** - extraction programs guarantees the yield on all documents in a cluster if it works for a single document in the cluster.

Question

How do you quantify that two documents are similar or dissimilar?

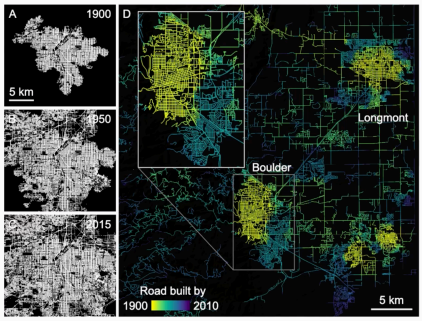
Clustering

- Task of grouping a set of objects in a way such that objects in the same group are more similar (in some sense or another) to each other than to those in other groups. - *Wikipedia*
- Grouping set of objects into groups, such that -
  - Objects within a cluster are highly similar (score is high)
  - Objects across the clusters are highly dissimilar (score is less)
- In terms of distance measure -
  - Objects within a cluster are near (distance is small)
  - Objects across the clusters are far (distance is high)
- There are many ways to achieve this task

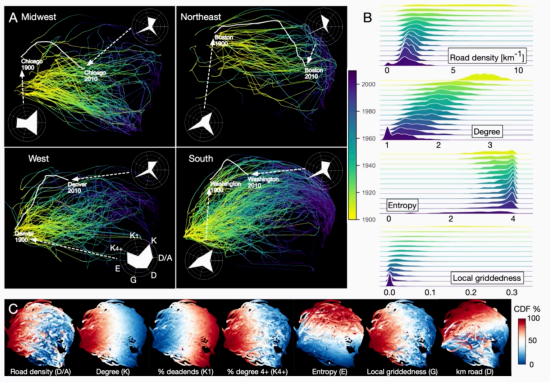
Classes of Similarity Measures

- For continuous variables
  - Metric
    - Euclidean distance, Minkowski distance, city block distance, sup distance
  - Not a Metric
    - Point symmetry distance, pearson correlation, cosine similarity (most common for document clustering)
- For discrete variables
  - Simple Matching Coefficient, Jaccard Coefficient (most common for document clustering)

Example: road development over time

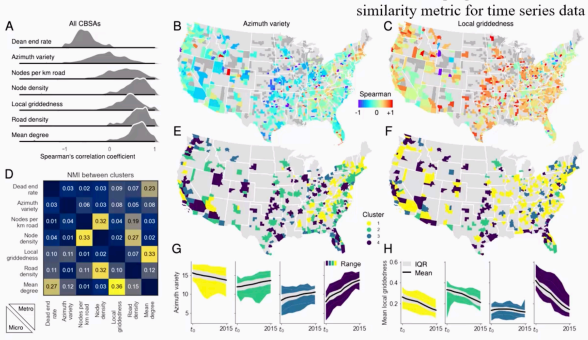


Embedding time series data



Clustering these data

"Time warping distance" as a similarity metric for time series data



School of Engineering

Another example:

Identifying Similar Documents

Similarity Measure: a scale from 0.0 to 1.0, in which 0.0 being extremely dissimilar and 1.0 being extremely similar. (This scale is usually a continuous scale)
Examples:
Cosine Similarity, Jaccard Similarity

Distance Measure: a non negative scale in which lowest value (usually 0) being extremely similar and a large value being extremely dissimilar. This scale is either discrete or continuous.
Examples :
Euclidean distance, Edit Distance, Normalized google Distance

⇒ It is possible to convert between similarity and distance measures via normalization, so used interchangeably

Minimum Edit Distance

- Number of editing operations required to transform one sequence into another.
- Three basic editing operations: INSERT, REMOVE and REPLACE.
- An useful measure to quantify how similar (or dissimilar) two strings are.
- Wide range of applications :
  - Spelling Correction (NLP)
  - Measure similarity of DNA sequences (Bioinformatics)
  - Information Extraction
  - Clustering
- Example :
  - CAR → CARS : INSERT 'S'
  - SLIDE → SIDE : REMOVE 'L'
  - ARE → ART : REPLACE 'E' with 'T'

Tree Edit Distance

- Edit distance on trees instead of sequences
- Standard measure for similarity in hierarchical data (tree)
- Same three fundamental editing operations: INSERT, REMOVE and REPLACE
- Has many applications :
  - NLP : parse trees
  - Information extraction : align and match
  - In this context : content analysis of web pages
- Algorithms :
  - Zhang and Shasha 's simple and fast algorithm using dynamic programming

Types of Clustering

- Hierarchical Clustering
  - Merge/Cluster
- Partitional Clustering
  - E.g., k-means, graph-theory, etc.
- NN-based Clustering
  - Neural Nets
- Kernel-based Clustering
  - SVM, Principal Component Analysis
- Sequential Data Clustering

Similarity Measure: Jaccard's Index

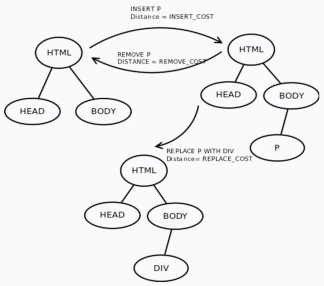
- Two items are similar to the extent of its overlapping features
- It is the ratio of common features to over all features, i.e. the ratio of intersection over union.
- similarity = $\frac{|A \cap B|}{|A| + |B| - |A \cap B|}$

Minimum Edit Distance

- Also called Levenshtein Distance
- The most efficient algorithm so far is based on dynamic programming
- Bottoms up string alignment to find minimum distance
- Backtrace to obtain string alignments
- More details:
  https://web.stanford.edu/class/cs124/lec/med.pdf

Example

- Tree Edit Distance



Popular Clustering Methods: K-means

- Pros
  - Fast
  - Easy
- Cons
  - Prior knowledge of number of clusters (parameter k) is required (but silhouette score can help us choose)
  - Limited to clusters with spherical shape
  - May not always be repeatable, due to random start state
- try it yourself:
  https://mahout.apache.org/users/clustering/k-means-clustering.html
  http://spark.apache.org/docs/latest/mllib-clustering.html

Connector Rank K-means clustering

- Exhaustive k-means clustering
- Mattmann 2007 (DISCO)
  - Related to software connector metadata, featurized, and turned into a probability/rank
  - Ranks are used as k-means centroid values

Algorithm 4: Exhaustive K-means Connector Clustering (k=2)

input : connector rank list R, distribution scenario S
output: answer key K, containing a set of appropriate connectors, and inappropriate connectors

```
Data: c_n ⟵ 2^|R|
Data: min_d ⟵ max ℚ
Data: connMap ⟵ ∅
for i=0 to |R| do
    add {1 → R[i]} to connMap
Data: low_G ⟵ ∅
for i=0 to c_n do
    Data: cSetSpec ⟵ toBinaryString(i)
    Data: cg_1, cg_2 ⟵ ∅
    for j=0 to |cSetSpec| do
        if cSetSpec{j} = 1 then
            add connMap{j} to cg_1
        else if cSetSpec{j} = 0 then
            add connMap{j} to cg_2
    Data: ad_1 ⟵ avg_d(cg_1)
    Data: ad_2 ⟵ avg_d(cg_2)
    if ad_1 + ad_2 < min_d then
        low_G ⟵ {cg_1, cg_2}
        min_d = ad_1 + ad_2
Data: K ⟵ (S, low_G)
return K
```

School of Engineering

Similarity Based Shared Nearest Neighbor Clustering

Algorithm:

- ➔ Compute initial list of near neighbors for each document using similarity matrix. (pick all documents that match your neighborhood criteria)
- ➔ Create a table where each row has a document and its near neighbors
- ➔ for all pairs of rows in table:
  - ➔ if at least *k* neighbors are common (i.e. shared):
    - ➔ merge the two rows into one (combine clusters)
    - ➔ update the reference of merged row to new row across all other remaining rows
- ➔ Repeat the previous step until no more clusters can be combined or a MaxIteration is reached.
- ➔ The remaining rows in the table are final clusters.

Paper :

Jarvis, R.A.; Patrick, Edward A., "Clustering Using a Similarity Measure Based on Shared Near Neighbors," in Computers, IEEE Transactions on , vol.C-22, no.11, pp.1025-1034, Nov. 1973

http://ieeexplore.ieee.org.libproxy1.usc.edu/stamp/stamp.jsp?arnumber=1672233

Example Implementation:

USC IRDS has an implementation based on bitwise operations : https://git.io/vgxq4

School of Engineering

Continuous features and the curse of dimensionality

- More dimensions
  - Less data along each dimension (sparsity)
  - Noise is not uniform (many dimensions could make correct clustering difficult)
  - Distances start to become more similar (harder to know what is "close" or "far"
- Motivation for dimension reduction techniques (takes advantage of the "blessing of nonuniformity")
- Distance metrics matter!
  - Some are more robust to the CoD
  - Some better capture "near" and "far" points

More tips for ML are here: https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf

Shared Nearest Neighbors Clustering

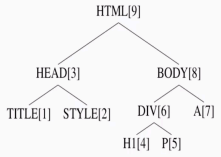
- Advantage over K Means
  - Prior knowledge of number of clusters is not required
  - Not limited to spherical cluster shape
  - Works with pairwise similarity measures
  - Flexibility of any similarity measure (not restricted to euclidean or vector space)
- Cons :
  - Computationally expensive - pairwise comparison
  - Requires more memory

Concept:

- If documents have high similarity ⇒ they belong to same cluster
- If two clusters share a threshold number of common documents (aka shared near neighbors) ⇒ they can be merged into one large cluster

Zang-Snasha's Algorithm

- The elements in tree are indexed in post order:
- The tree is incrementally built from smaller forests and the edit cost between two forests is computed by gradually aligning nodes with Insert, Remove and Replace operations.
- Dynamic programming is used to efficiently compute the edit distance between the root nodes of two DOM trees.
- Pros : Space complexity $O(n^2)$
- Cons : Time complexity $O(n^4)$
- Example Implementation: https://git.io/vgxuW



K. Zhang, & D. Shasha, (1989). Simple fast algorithms for the editing distance between trees and related problems. SIAM journal on computing, 18(6), 1245-1262.

Application: Clustering Documents

- USC Information Retrieval and Data Science Group (irds.usc.edu) is actively working
- Structural similarity using Tree Edit Distance
  - HTML DOM trees are compared
- Style similarity using Jaccards index
  - CSS class names are used as features of set
- Aggregated similarity = structure + style
- Cluster the pairwise similarity using shared near neighbor method
- https://github.com/uscdataScience/autoextractor/wiki

Summary

- Why Clustering?
- Definition of Similarity Measures
- Types of Clustering
- Application to Content Detection & Analysis