Classic Data Analytics I

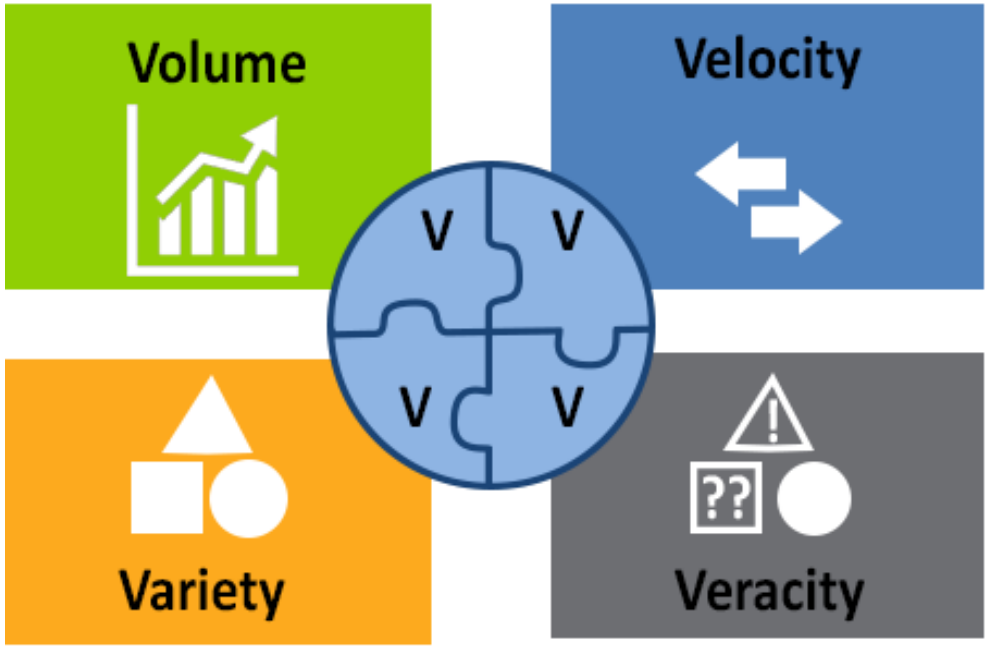
Zhang Chen
Department of Computing (COMP)

Announcements

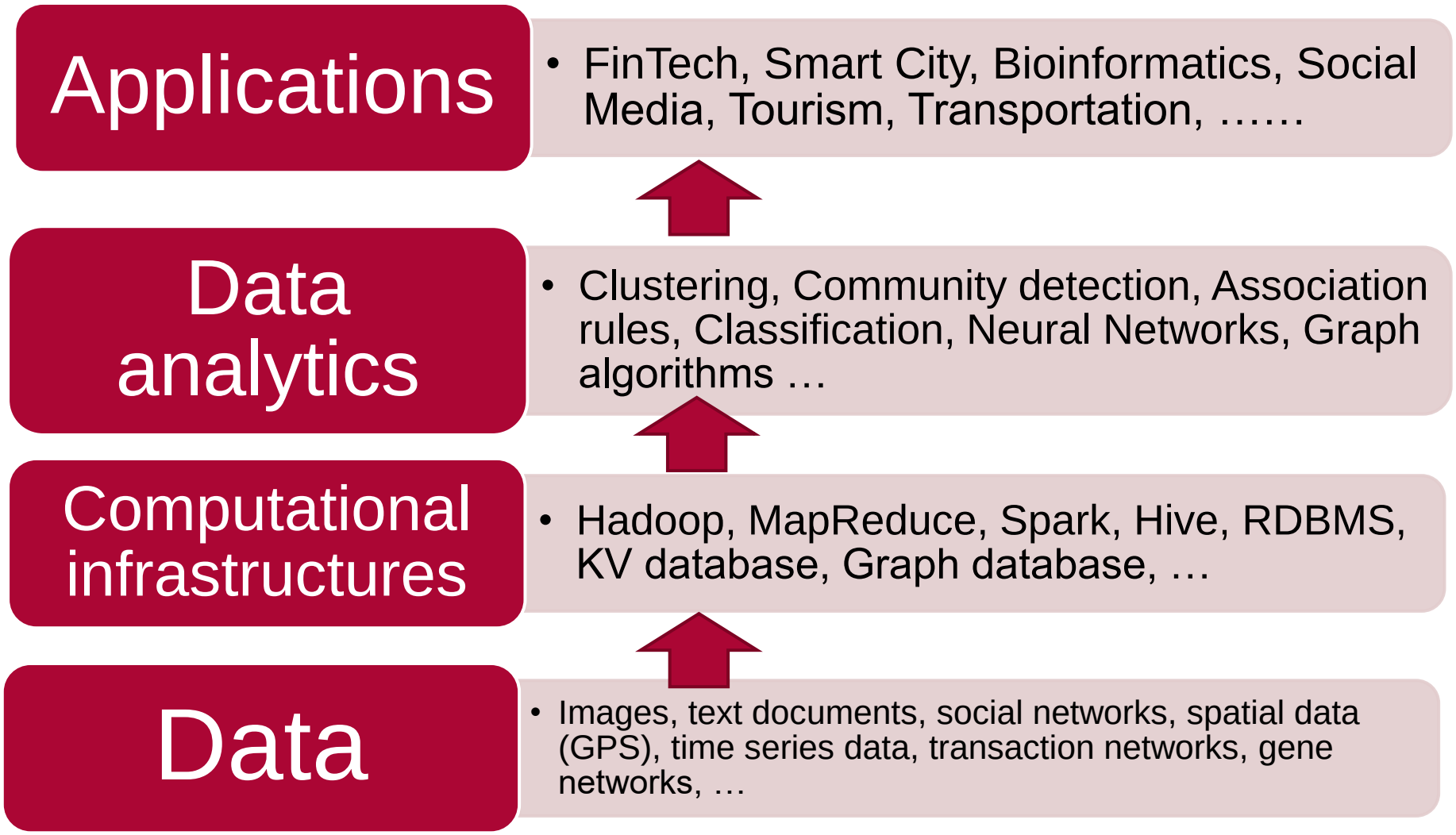
- Final Exam
  - **Open-book exam**: Students are allowed to bring one piece of A4-size paper to the exam venues. Standard Calculator is allowed.
- Group Presentation. **A forum** is setup for group matching on blackboard.
- Learning materials
  - **Class slides** and **reading lists** are updated to blackboard regularly

Recap

- Big data applications
- Big data characteristics



Recap: Overview

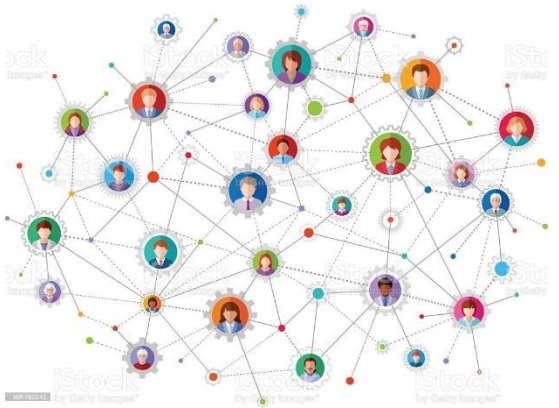


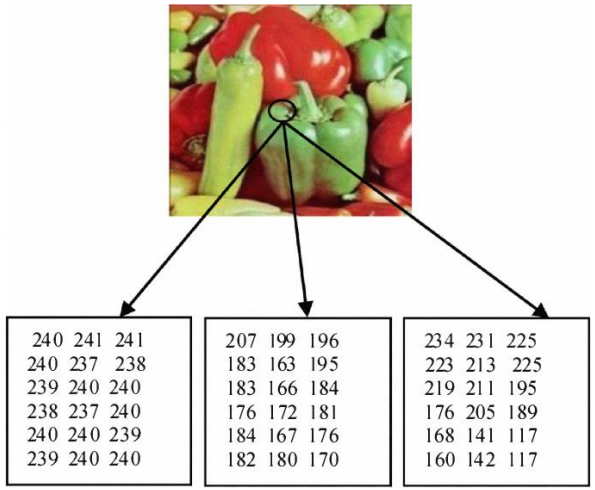
Recap: Data Models

Name	FName	City	Age	Salary
Smith	John	3	35	$280
Doe	Jane	1	28	$325
Brown	Scott	3	41	$265
Howard	Shemp	4	48	$359
Taylor	Tom	2	22	$250

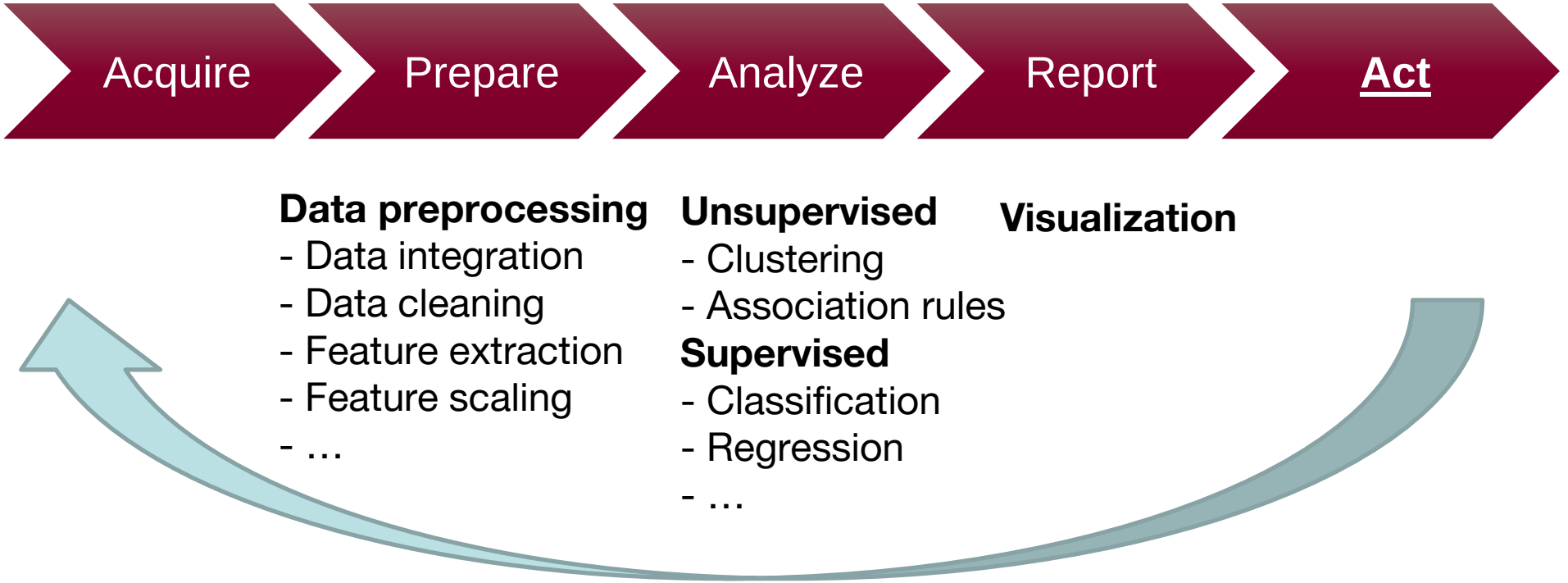
Tuple

1	John also likes to watch football games.
2	John likes to watch movies. Mary likes movies too.

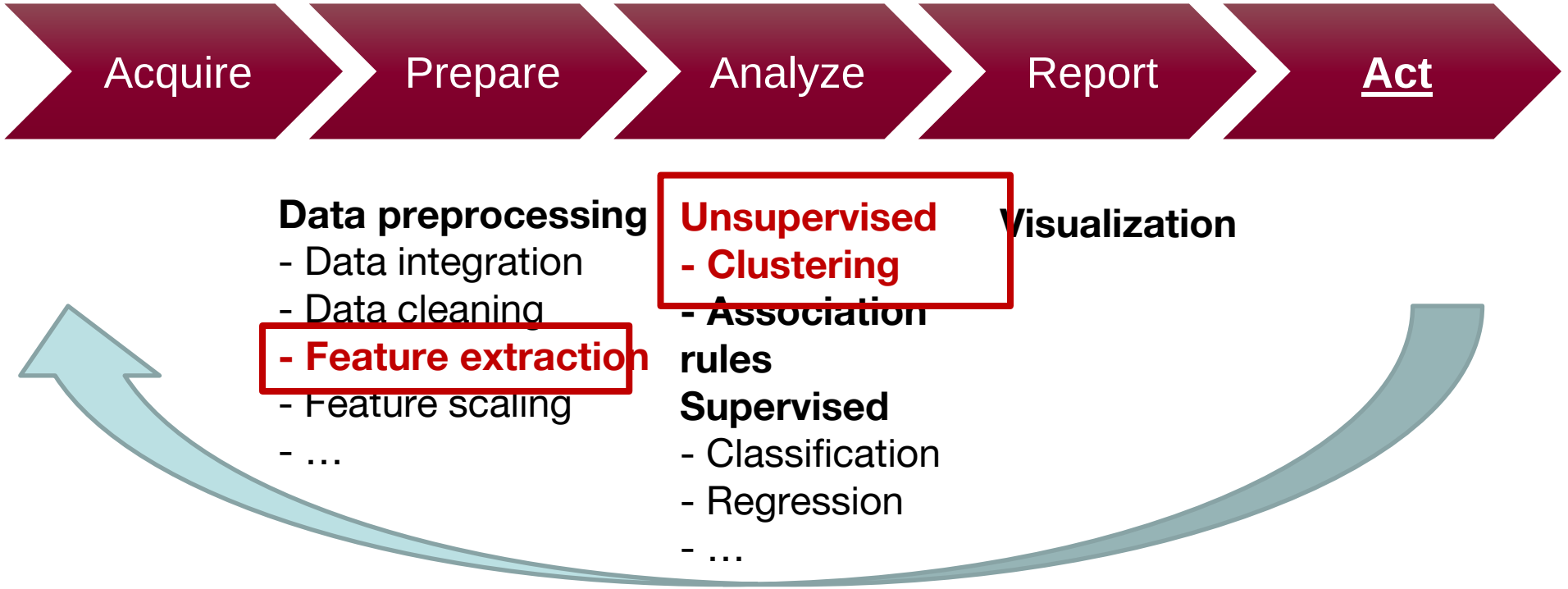




Recap: Big Data Process



Recap: Big Data Process



5-Step Data Analytical Process

- Prepare data (ii): **data preprocessing**
  - Data integration
  - Data cleaning
  - Feature extraction:
    - **the process of transforming raw data into numerical features that can be processed** while preserving the information in the original data set.
  - Feature scaling
  - …

Outline

- Start with Text Analytics
  - Text Data and Applications
  - Feature Extraction
  - Similarity Search
- Unsupervised Algorithms
  - Clustering
    - K-Means
    - DBSCAN
    - Hierarchical Clustering

Text/Document

- The term “document” is used to denote a text
  - It can be a text message, a blog post, or a review
- Example of a document (a hotel review):

Document 1:
It is a hotel; quiet hotel, cheap hotel: $200 at hotel. Nice hotel

Examples of Text Data

- Reviews

- Blogs / Microblogs

- Forums
- Instant messaging

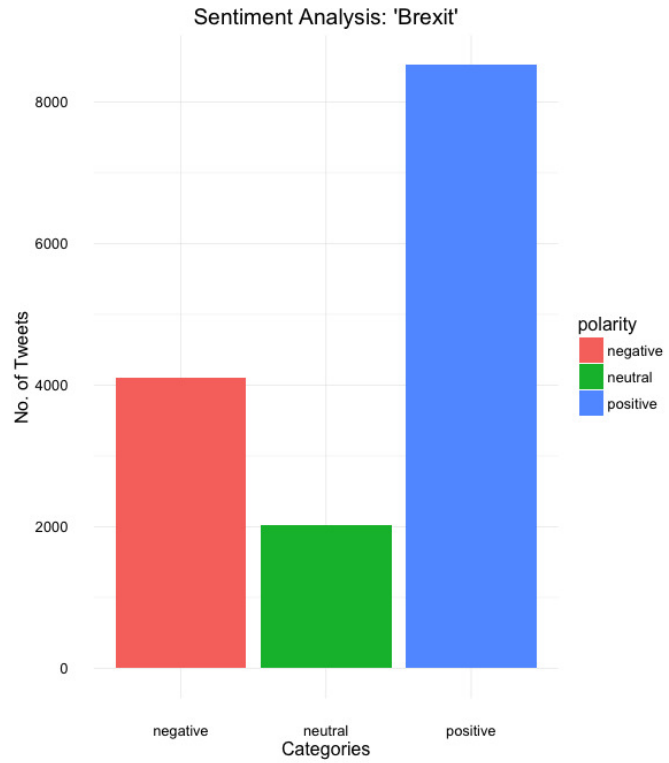
Text Analytics

- Word cloud
  - Visualization of keywords based on their frequencies in documents

- Searching
  - E.g., search top-10 documents that are most similar to a given document

Text Analytics

- Clustering
  - Grouping documents based on their hidden topics
- Classification
  - Spam email detection based on email content
- Sentiment analysis
  - It extracts social sentiment from a document (e.g., positive, negative, neutral)
    - Classification or regression problem?



Demo of Sentiment Analysis

- https://text2data.com/Demo
- https://www.hootsuite.com/social-media-tools/sentiment-analysis-tool?srsltid=AfmBOoq5zBmaVlqFIojwgFNljojtvjpCqiIp_EpKQTG4xffTnc7m3uTI

  - “Great restaurants and bars on site, all the staff are really friendly, everywhere is kept clean and tidy”
  - “If you want a quiet complex at night do not go here. The complex staff will not take action against noisy guests”
  - “I’m not sure if I like the new design”

Characteristics of Text Data

- Different types: characters, numbers, punctuations
- High-frequency words: “a”, “the”, “in”, “to”, “is” …
- Different forms of words:
  - “hotel”, “hotels”, “quiet”, “Quiet”, “cheap”, ”cheapest”
  - “computer”, “computational”, “computation”

Document 1:
It is a hotel; quiet hotel, cheap hotel: $200 at hotel. Nice hotel

Document 2:
It is a quiet hotel. Nice!

Document 3:
It is ok with the noise, since the hotel is cheapest among hotels.

Outline

- Start with Text Analytics
  - Text Data and Applications
  - Feature Extraction
    - Tokenization
    - Bag-of-Words and Vector Space Models
  - Similarity Search
- Unsupervised Algorithms
  - Clustering

Which is the Top-1 Similar Document?

- With respect to query keywords: **{cheap, quiet, nice, hotel}**
- **Application:** recommend a cheap, quiet, and nice hotel by searching the review comments of the 3 hotels

Document 1:
It is a hotel; quiet hotel, cheap hotel: $200 at hotel. Nice hotel

Document 2:
It is a quiet hotel. Nice!

Document 3:
It is ok with the noise, since the hotel is cheapest among hotels.

Unstructured Text Data

- Hard to be processed directly by computers
- Require data preprocessing, specifically, **feature extraction**

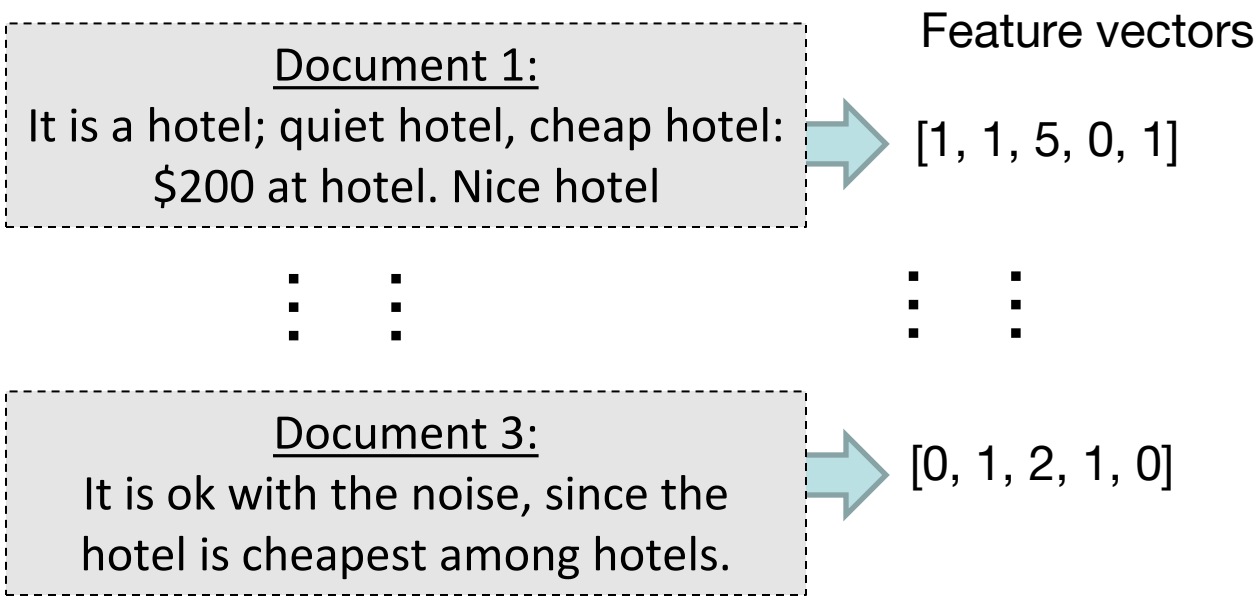
Document 1:
It is a hotel; quiet hotel, cheap hotel: $200 at hotel. Nice hotel

Document 2:
It is a quiet hotel. Nice!

Document 3:
It is ok with the noise, since the hotel is cheapest among hotels.

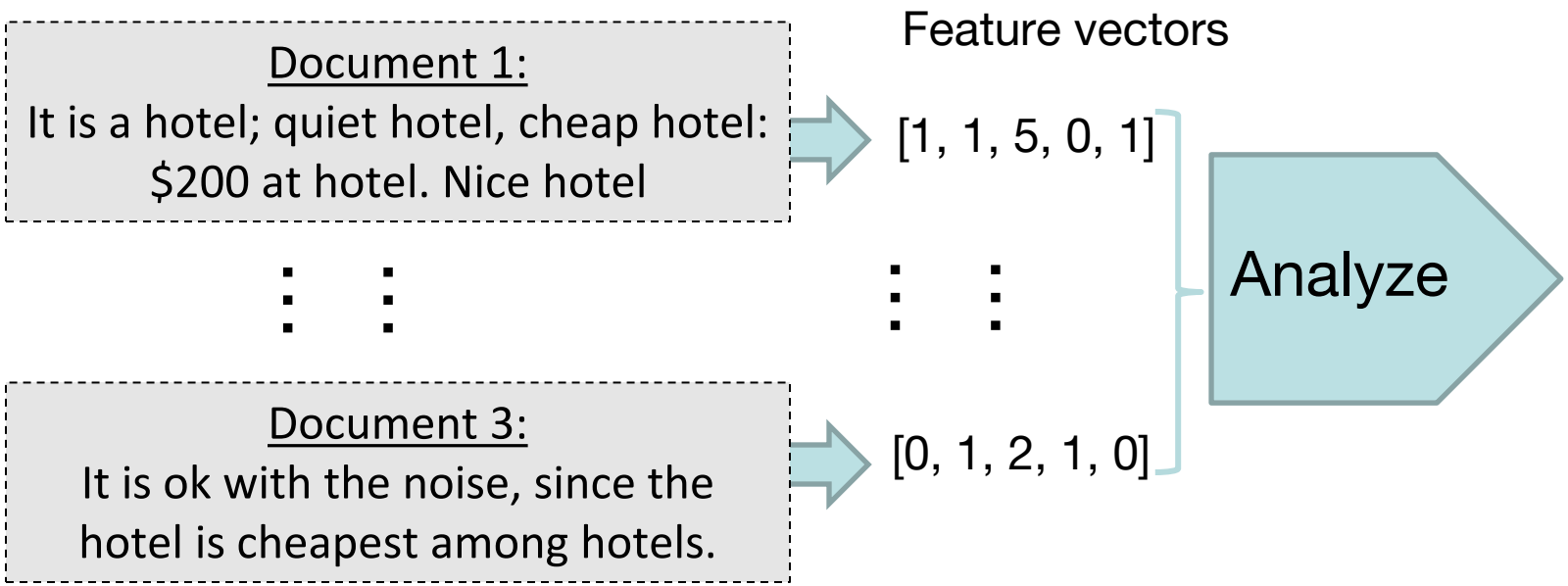
Feature Extraction on Text Data

- The process of transforming raw data into numerical features that can be processed while preserving the information in the original data set.



After Feature Extraction

- Then feed the extracted features into data analytical methods (e.g., clustering, classification, searching queries)



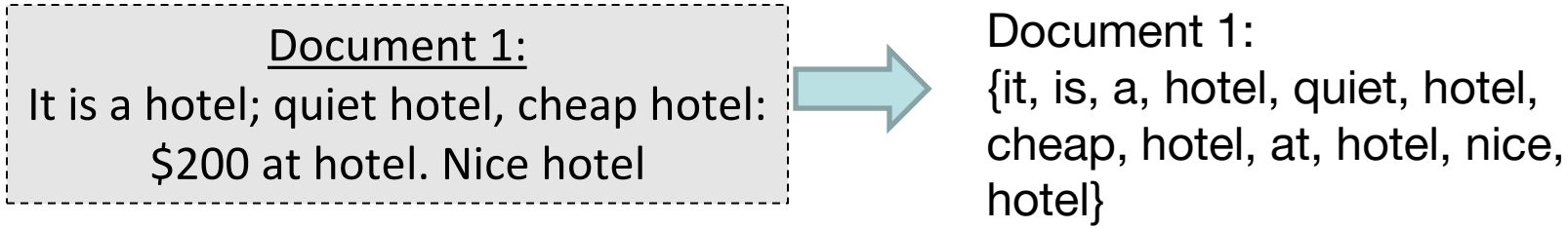
Tokenization

- Convert the text into a sequence of tokens (words/terms)
- Observations
  - Usually, punctuations, numbers, and case are meaningless
  - But sometimes punctuation (e-mail “@”), numbers (1999), and case (Republican vs. republican) can be meaningful

Document 1:
It is a hotel; quiet hotel, cheap hotel: $200 at hotel. Nice hotel

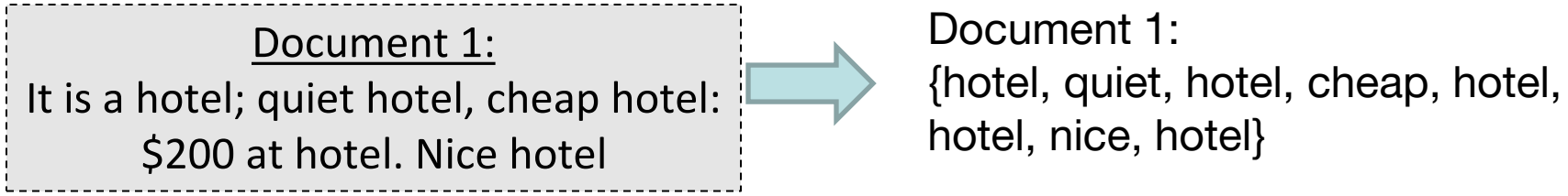
Tokenization

- A simple approach
  - Split by spaces
  - Ignore all numbers and punctuation
  - Use case-insensitive strings as tokens



Stopwords

- Exclude high-frequency words called **stopwords**
  - function words: “a”, “the”, “in”, “to”
  - pronouns: “I”, “he”, “she”, “it”
- Stopwords are language dependent
  - In English, there is a standard set of about 500 stopwords



Stemming

- For matching purpose, we convert keywords in the documents to their stems (base word forms)
  - Example:
    - Replace “hotels” by “hotel”
    - Replace “cheapest” by “cheap”

Document 1:
It is a hotel; quiet hotel, cheap hotel: $200 at hotel. Nice hotel

Document 2:
It is a quiet hotel. Nice!

Document 3:
It is ok with the noise, since the hotel is **cheapest** among **hotels**.

Porter Stemmer

- Porter Stemmer is a simple procedure for removing known prefixes/suffixes in English without using a dictionary
  - Developed by Martin Porter in 1980
  - It can convert similar tokens to the same token
  - “computer”, “computational”, “computation”

    → the same token “comput”
- Side effects
  - produce stems that are not (English) words: “comput”
  - produce a stem that has different meaning from the original word: “organization” → “organ”

Libraries for Tokenization

- English:
  - NLTK
    - https://www.nltk.org/
  - SpaCy
    - https://spacy.io/
  - CoreNLP
    - https://stanfordnlp.github.io/CoreNLP/
  - …

Libraries for Tokenization

- What about Chinese language?
  - Compared with English, what are the differences?
  - Example: “今天天气不错，出去玩。”
- THULAC
  - https://github.com/thunlp/THULAC-Python
- FoolNLTK
  - https://github.com/rockyzhengwu/FoolNLTK
- HanLP
  - https://github.com/hankcs/HanLP/

Text Corpus and Vocabulary

- **Text corpus** refers to the set of texts used for the task.

Document 1:
It is a hotel; quiet hotel, cheap hotel: $200 at hotel. Nice hotel

Document 2:
It is a quiet hotel. Nice!

Document 3:
It is ok with the noise, since the hotel is cheapest among hotels.

Text Corpus and Vocabulary

- The set of unique words used in the text corpus is referred to as the **vocabulary**.
  - e.g., the vocabulary is {quiet, cheap, hotel, noise, nice}
- Why such definitions of corpus and vocabulary?
  - We do not want to consider an unlimited scope of words
  - When processing raw text, everything is done around the vocabulary

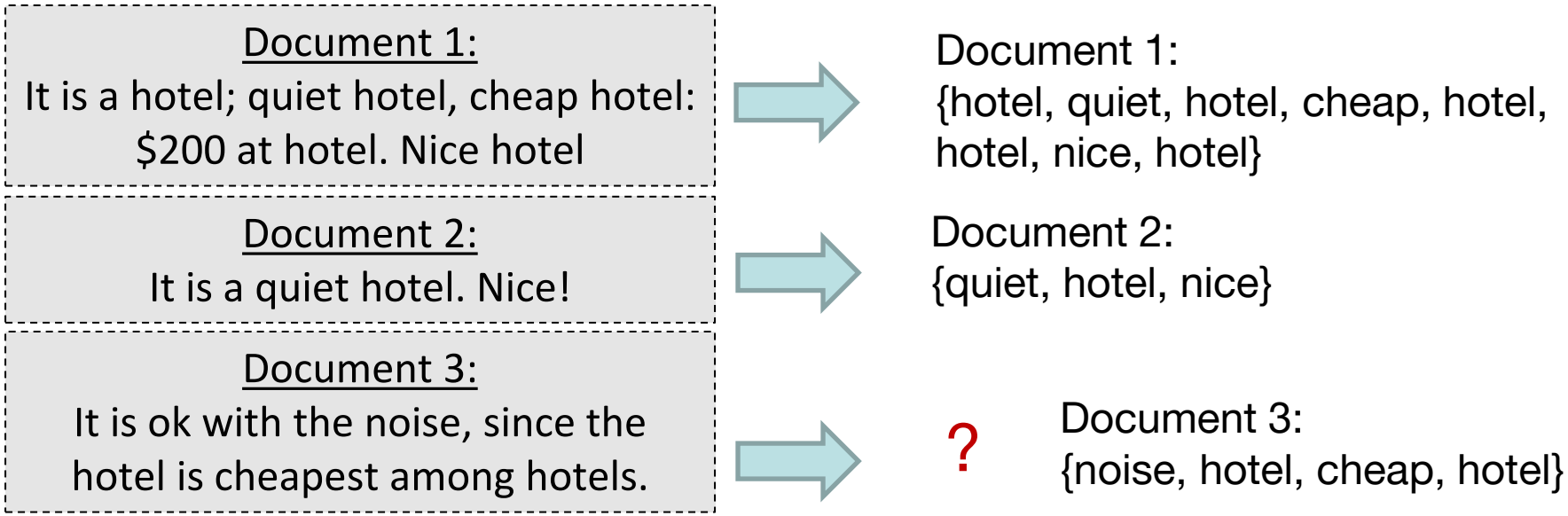
Document 1:
It is a hotel; quiet hotel, cheap hotel: $200 at hotel. Nice hotel

Document 2:
It is a quiet hotel. Nice!

Document 3:
It is ok with the noise, since the hotel is cheapest among hotels.

Text Corpus and Vocabulary

- Suppose that the vocabulary of the task is
  - {quiet, cheap, hotel, noise, nice}
- After the steps ahead, we have



Bag-of-Words Model

- Bags (Multiset) of words with **Term Frequency (TF)**
  - Document 1: {hotel, quiet, hotel, cheap, hotel, hotel, nice, hotel}

    ⇒ {hotel:5, quiet:1, cheap:1, nice:1}
  - Document 2: {quiet, hotel, nice}

    ⇒ {quiet:1, hotel:1, nice: 1}
  - Document 3: {noise, hotel, cheap, hotel}

    ⇒ {noise:1, hotel:2, cheap:1}

Vocabulary	quiet	cheap	hotel	noise	nice
Document 1	1	1	5	0	1
Document 2	1	0	1	0	1
Document 3	0	1	2	1	0

Bag-of-Words Model is one Example of Vector Space Model

- After **feature extraction**,
  - We convert each document to a **vector**
  - We convert the text corpus to a **matrix**

Vocabulary	quiet	cheap	hotel	noise	nice
Document 1	1	1	5	0	1
Document 2	1	0	1	0	1
Document 3	0	1	2	1	0

5-dim vector

3x5 matrix

Vector Space Model

- The vector space model is defined by basis vectors
  - Each term in **vocabulary** defines a basis vector T_i
  - Each basis vector is orthogonal to each other
  - For example, given five terms: quiet, cheap, hotel, noise, nice
    - T_1 for *quiet*: $[1,0,0,0,0]$
    - T_2 for *cheap*: $[0,1,0,0,0]$
    - T_3 for *hotel*: $[0,0,1,0,0]$
    - T_4 for *noise*: $[0,0,0,1,0]$
    - T_5 for *nice*: $[0,0,0,0,1]$
- Document D_j as t-dimensional vector
  - How to get t? Or what is t?
  - t is the size of vocabulary (in other words, t is the number of distinct terms that

Vector Space Model

- Document D_j as t-dimensional vector
  - $D_j = (w_{1j}, w_{2j}, \ldots, w_{tj})$
  - w_{ij} denotes the weight of term T_i in a document D_j

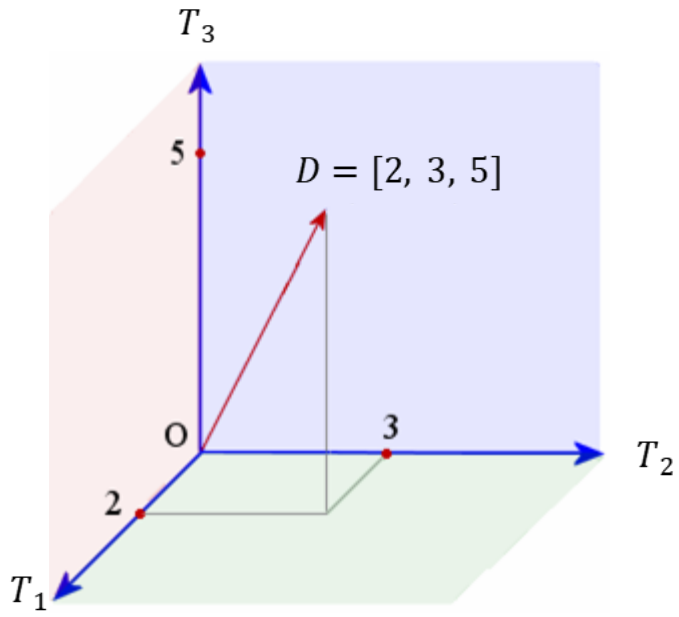
Document Collection

- Document matrix: model of a collection of n documents
  - An entry w_{ij} in the matrix denotes the "weight" of a term T_i in a document D_j
  - $w_{ij} = 0$ means the term doesn't exist in the document

$$\begin{array}{c} \\ D_1 \\ D_2 \\ \vdots \\ \vdots \\ D_n \end{array} \begin{array}{c} \begin{array}{cccc} T_1 & T_2 & \cdots & T_t \end{array} \\ \begin{pmatrix} w_{11} & w_{21} & \cdots & w_{t1} \\ w_{12} & w_{22} & \cdots & w_{t2} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ w_{1n} & w_{2n} & \cdots & w_{tn} \end{pmatrix} \end{array}$$

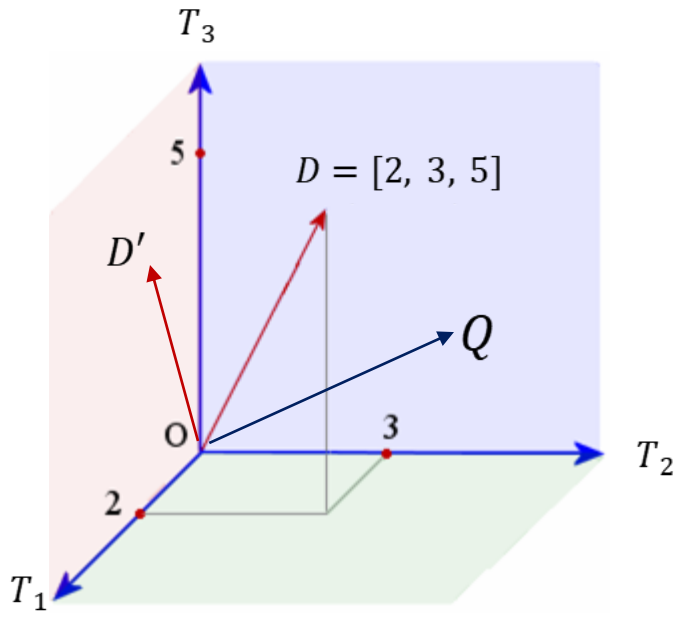
Geometric Interpretation

- Example of VSM: suppose that all possible terms are:
  - T_1
  - T_2
  - T_3
- Document $D = [2,\ 3,\ 5]$



Geometric Interpretation

- Example of VSM: suppose that all possible terms are:
  - T_1
  - T_2
  - T_3
- Document $D = [2,\ 3,\ 5]$
- Document $D' = [1,\ 0,\ 3]$
- Given a query Q, which is more like Q?
- How to measure the similarity?
  - Angle? Distance? Projection?



Outline

- Start with Text Analytics
  - Text Data and Applications
  - Feature Extraction
  - Similarity Search
    - Cosine Similarity
    - TF-IDF
- Unsupervised Algorithms
  - Clustering

Which is the Top-1 Similar Document?

- With respect to query text: **{cheap, quiet, nice, hotel}**
- Step 1:
  - **What do we need to do?**

Vocabulary	quiet	cheap	hotel	noise	nice
Document 1	1	1	5	0	1
Document 2	1	0	1	0	0
Document 3	0	1	2	1	0

Which is the Top-1 Similar Document?

- With respect to query text: **{cheap, quiet, nice, hotel}**
- Step 1:
  - **Convert query text to query vector**

[1, 1, 1, 0, 1]

Vocabulary	quiet	cheap	hotel	noise	nice
Document 1	1	1	5	0	1
Document 2	1	0	1	0	0
Document 3	0	1	2	1	0

Which is the Top-1 Similar Document?

- With respect to query text: **{cheap, quiet, nice, hotel}**
- Step 2:
  - We need a **similarity measure** between query and documents

[1, 1, 1, 0, 1]

Document 1: [1, 1, 5, 0, 1]
Document 2: [1, 0, 1, 0, 0]
Document 3: [0, 1, 2, 1, 0]

Inner Product

- Given two real-value vectors $\mathbf{q} = [q_1, ..., q_n]$ and $\mathbf{x} = [x_1, x_2, ..., x_n]$, their inner product is the sum of elementwise multiplication

$$\mathbf{q} \cdot \mathbf{x} = x_1 q_1 + ... + x_n q_n$$

- Inner product of [1, 1, 1, 0, 1] and [1, 1, 5, 0, 1]
  - 1*1 + 1*1 + 1*5 + 0*0 + 1*1 = 8

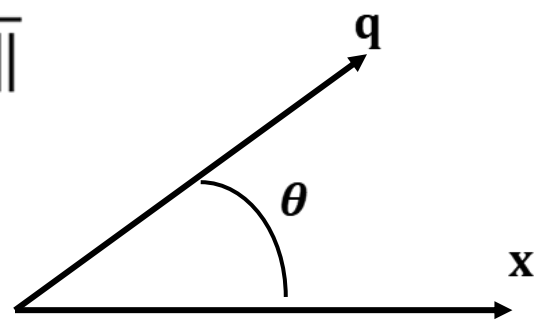
Cosine Similarity

- The **magnitude/length** of a n-dimensional vector $\mathbf{x} = [x_1, x_2, ..., x_n]$

$$\|\mathbf{x}\| = \sqrt{x_1^2 + ... + x_n^2}$$

- Given vectors $\mathbf{q} = [q_1, ..., q_n]$ and $\mathbf{x} = [x_1, x_2, ..., x_n]$, Cosine similarity measures the **cosine** of the angle between vectors

$$\text{CosSim}(\mathbf{q}, \mathbf{x}) = \frac{\mathbf{q} \cdot \mathbf{x}}{\|\mathbf{q}\|\|\mathbf{x}\|}$$



Cosine similarity

Inner Product and Cosine Similarity

- Both are defined in the inner product space
- Cosine similarity only cares about angle difference
- Inner product cares about angle and magnitude

Cosine Similarity: Example

- Given vectors $\mathbf{q} = [q_1, \ldots, q_n]$ and $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, Cosine similarity measures the cosine of the angle between vectors

$$\text{CosSim}(\mathbf{q}, \mathbf{x}) = \frac{\mathbf{q} \cdot \mathbf{x}}{\|\mathbf{q}\| \|\mathbf{x}\|}$$

Query:	[1, 1, 1, 0, 1]
Document 1:	[1, 1, 5, 0, 1]
Document 2:	[1, 0, 1, 0, 1]
Document 3:	[0, 1, 2, 1, 0]

$$\text{CosSim(query, doc1)} = \frac{1*1+1*1+1*5+1*1}{\sqrt{1+1+1+1}\sqrt{1+1+5^2+1}} = \frac{8}{\sqrt{4}\sqrt{28}} \approx 0.756$$

Cosine Similarity: Exercise

- Given vectors $\mathbf{q} = [q_1, \ldots, q_n]$ and $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, Cosine similarity measures the cosine of the angle between vectors

$$\text{CosSim}(\mathbf{q}, \mathbf{x}) = \frac{\mathbf{q} \cdot \mathbf{x}}{\|\mathbf{q}\| \|\mathbf{x}\|}$$

Query: [1, 1, 1, 0, 1]
Document 1: [1, 1, 5, 0, 1]
Document 2: [1, 0, 1, 0, 1]
Document 3: [0, 1, 2, 1, 0]

EXERCISE: calculate cosine similarity between query and document 2, between query and document 3.

Which is the Top-1 Similar Document?

- With respect to query keywords: **{cheap, quiet, nice, hotel}**
- **Application:** recommend a cheap, quiet, and nice hotel by searching the review comments of the 3 hotels
- **Are you happy about this?**

Document 1:
It is a hotel; quiet hotel, cheap hotel: $200 at hotel. Nice hotel

Document 2:
It is a quiet hotel. Nice!

Document 3:
It is ok with the noise, since the hotel is cheapest among hotels.

$\text{CosSim(query, doc1)} \approx 0.756$

$\textbf{CosSim(query, doc2)} = \frac{3}{2\sqrt{3}} \approx \mathbf{0.866}$

$\text{CosSim(query, doc3)} = \frac{3}{2\sqrt{6}} \approx 0.612$

Outline

- Start with Text Analytics
  - Text Data and Applications
  - Feature Extraction
  - Similarity Search
    - Cosine Similarity
    - TF-IDF
- Unsupervised Algorithms
  - Clustering

Determine the Importance of a Word: TF-IDF

- (i) How to determine important words in **a document**?
  - Term Frequency (TF)
- (ii) How to determine the degree of importance of a term within the **entire document collection**?
  - Which word is more/less important in the collection?

	quiet	cheap	hotel	noise	nice
Document 1	1	1	5	0	1
Document 2	1	0	1	0	1
Document 3	0	1	2	1	0

(i) Term Frequency TF

- More frequent terms in a document are more important, i.e., more indicative of the topic

  tf_{ij} = frequency of term i in document j

(ii) Inverse Document Frequency IDF

- Terms that appear in many different documents are less indicative of overall topic in a document

  df_i = number of documents containing term i

  idf_i = inverse document frequency of term i

  $= \log_2 \left(\frac{N}{df_i}\right)$ (N : total number of documents)

- IDF indicates a term's discrimination power

word	df	idf
hotel	3	$\log_2(3/3)=0$
quiet	2	$\log_2(3/2)=0.585$
cheap	2	0.585
nice	2	0.585
noise	1	$\log_2(3/1)=1.585$

	quiet	cheap	hotel	noise	nice
Document 1	1	1	5	0	1
Document 2	1	0	1	0	1
Document 3	0	1	2	1	0

- IDF of 'quiet': $\log_2 \frac{3}{2} = 0.585$; IDF of 'hotel': $\log_2 \frac{3}{3} = 0$

TF-IDF Weighting

- The combined term importance indicator is called $tf\text{-}idf$ weighting:

$$w_{ij} = tf_{ij} \cdot idf_i$$

- A term has high weight when
  - *it occurs frequently in the document*
  - *but rarely in the rest of the collection*
- Experiments show that $tf\text{-}idf$ is effective

TF-IDF Calculation

- Term Frequency of every word within a document:

	quiet	cheap	hotel	noise	nice
Document 1	1	1	5	0	1
Document 2	1	0	1	0	1
Document 3	0	1	2	1	0

- Compute the IDF of every word in the document collection
- Compute the TF-IDF of each word in each document

TF-IDF Calculation

- Term Frequency of every word within a document:

	quiet	cheap	hotel	noise	nice
Document 1	1	1	5	0	1
Document 2	1	0	1	0	1
Document 3	0	1	2	1	0

- IDF of 'quiet': $\log_2(\frac{3}{2}) = 0.585$; IDF of 'hotel': $\log_2(\frac{3}{3}) = 0$
- IDF of 'cheap': $\log_2(\frac{3}{2}) = 0.585$; IDF of 'noise': $\log_2 3 = 1.585$;

  IDF of 'nice' : $\log_2(\frac{3}{2})$ =0.585
- TF-IDF of each word within each document

	quiet	cheap	hotel	noise	nice
Document 1	0.585	0.585	0	0	0.585
Document 2	0.585	0	0	0	0.585
Document 3	0	0.585	0	1.585	0

In this extreme example, 'hotel' has 0 importance since it is too common

TF-IDF:
Which is the Top-1 Similar Document?

- With respect to query text: {cheap, quiet, nice, hotel}
- Query vector: [1, 1, 1, 0, 1]
- Calculate the **cosine similarity** between query and every document, using TF-IDF feature vectors of the documents

	quiet	cheap	hotel	noise	nice
Document 1	0.585	0.585	0	0	0.585
Document 2	0.585	0	0	0	0.585
Document 3	0	0.585	0	1.585	0

TF-IDF:
Which is the Top-1 Similar Document?

- With respect to query text: {cheap, quiet, nice, hotel}
- Query vector: [1, 1, 1, 0, 1]
- Calculate the **cosine similarity** between query and every document, using TF-IDF feature vectors of the documents
  - **Doc1**: $\frac{0.585+0.585+0.585}{\sqrt{4}*\sqrt{0.585^2+0.585^2+0.585^2}} = \frac{3*0.585}{2*\sqrt{3}*0.585} = 0.866$
  - Doc2: $\frac{2*0.585}{2*\sqrt{2}*0.585} = 0.707$
  - Doc3: $\frac{0.585}{2*\sqrt{0.585^2+1.585^2}} = 0.173$

Document 1:
It is a hotel; quiet hotel, cheap hotel: $200 at hotel. Nice hotel

	quiet	cheap	hotel	noise	nice
Document 1	0.585	0.585	0	0	0.585
Document 2	0.585	0	0	0	0.585
Document 3	0	0.585	0	1.585	0

Many Choices of TF

- TF variants: There are various other ways to define TF.
- Notation: Raw occurrence of term t in document d: $f_{t,d}$

Variants of TF

Variant	Definition
Binary	
Raw Count	
Term Frequency	
Log Normalization	
Double Normalization 0.5	
Double Normalization K	

quiet	cheap	hotel	noise	nice
1	1	5	0	1
1	0	1	0	1
0	1	2	1	0

Many Choices of IDF

- IDF variants: There are various other ways to define IDF.
- Notation: Term t exists in n_t of N documents ($n_t = |\{d \in D: t \in d\}|$)

Variants of IDF

Variant	Definition
Unary	
Inverse Document Frequency	
Inverse Document Frequency Smooth	
Probabilistic Inverse Document Frequency	

quiet	cheap	hotel	noise	nice
1	1	5	0	1
1	0	1	0	1
0	1	2	1	0

Top-k Similarity Search on Documents

- Given a collection of documents D, find k documents that are most similar to another document Q, based on TF-IDF.

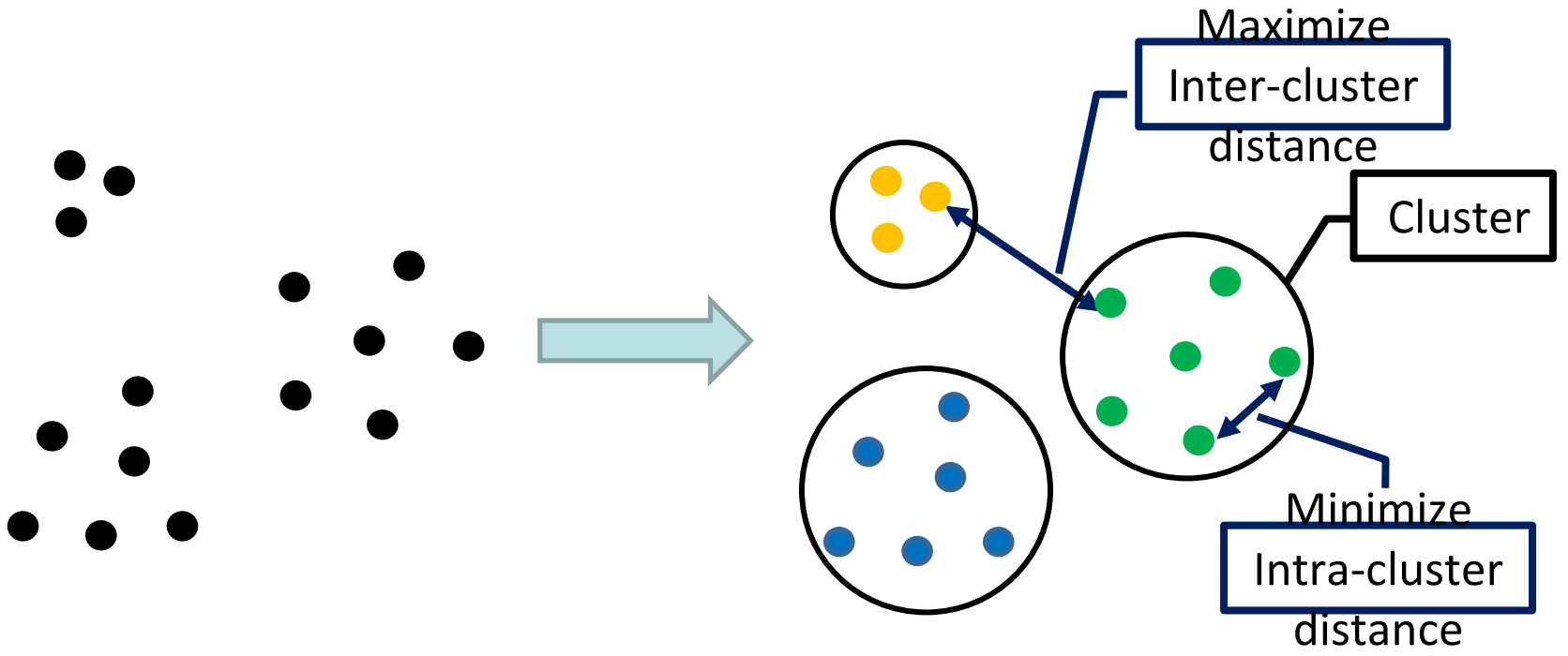
- Algorithm
  - Convert all documents in collection D to TF-IDF weighted vectors $\{\boldsymbol{d_j}\}$
  - Convert document Q to a vector $\boldsymbol{q}$
    - If Q is a document in D: use the TF-IDF-weighted vector
    - If not, two ways to convert:
      - Use the binary vector (e.g., the example we discussed before)
      - Use the TF-IDF-weighted vector
        - Do **NOT** re-calculate IDFs for the query. Just use IDFs from D.
  - For each $\boldsymbol{D_j}$ in D do
    - Compute score $CosSim(\boldsymbol{d_j}, \boldsymbol{q})$
  - Find the top-k scores;
  - Return top-k documents to the user

Outline

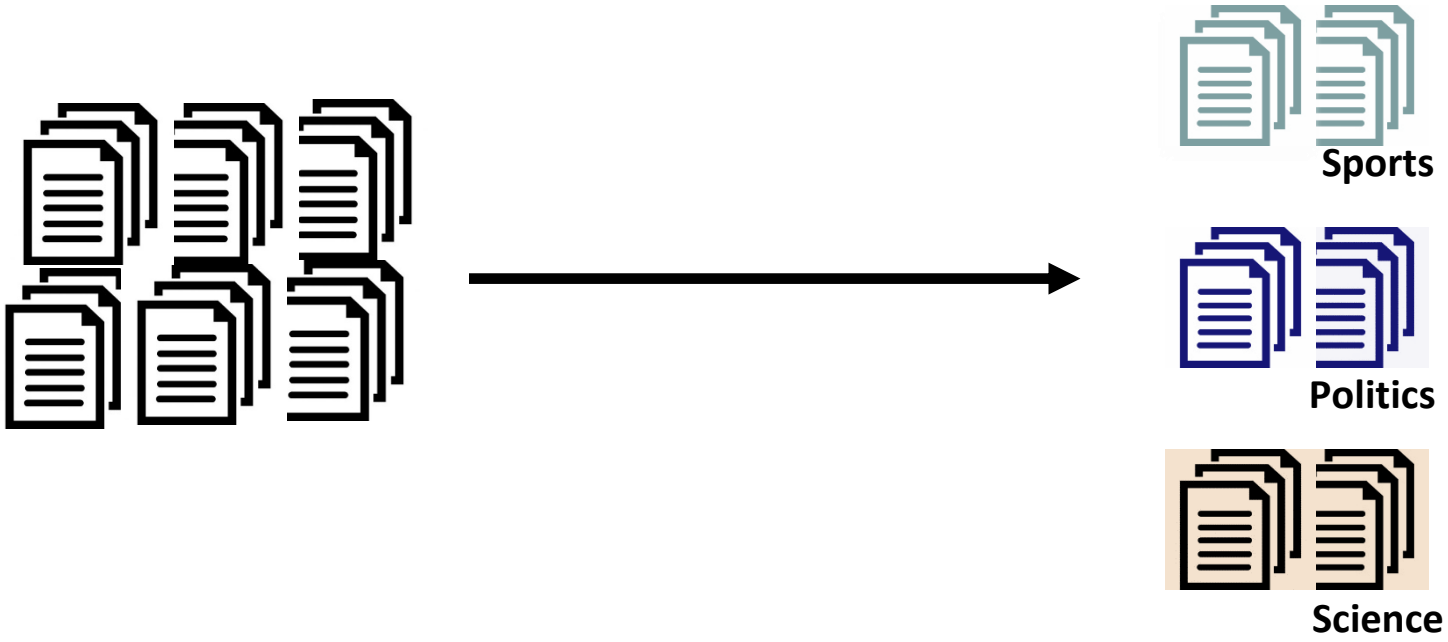
- Start with Text Analytics
  - Text Data and Applications
  - Feature Extraction
  - Similarity Search
- Unsupervised Algorithms
  - Clustering
    - Applications and Concepts
    - K-Means
    - DBSCAN
    - Hierarchical Clustering

What is Clustering

- **Clustering** is the task of grouping a set of objects in such a way that objects in the same group are more similar **(similarity measures)** to each other than to those in other groups (clusters)



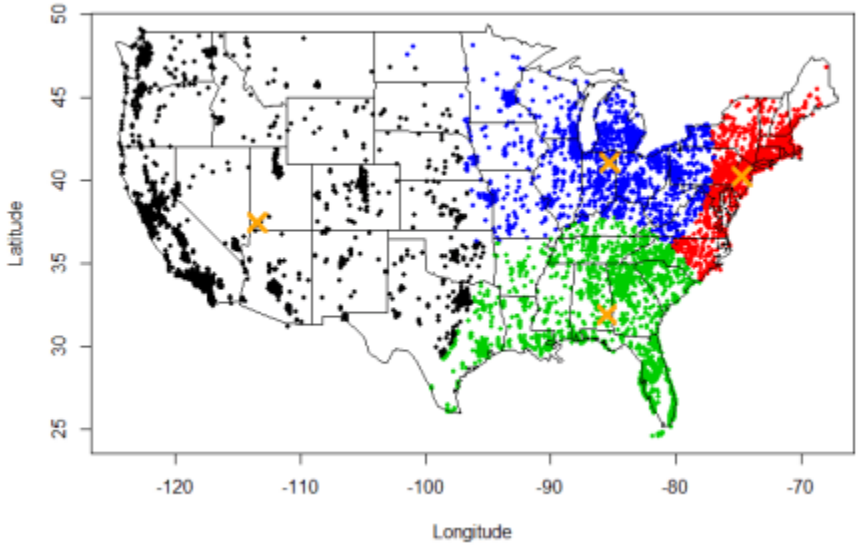
Applications: Document Clustering



Applications: Image Clustering

Identify similar visual patterns (pattern recognition)

Applications: Spatial Clustering



More Applications

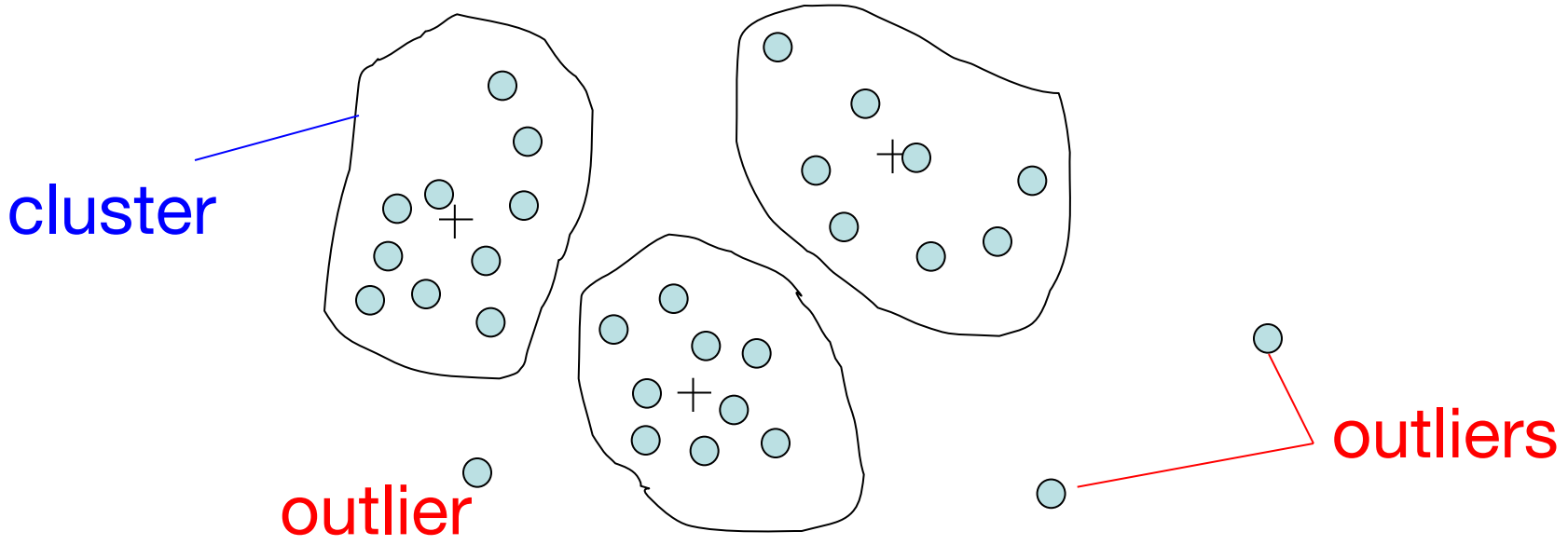
- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- Land use: Identification of areas of similar land use in an earth observation database
- Insurance: Identifying groups of motor insurance policy holders with a high average claim cost
- City-planning: Identifying groups of houses according to their house type, value, and geographical location
- Earth-quake studies: Observed earthquake epicenters should be clustered along continent faults

Clustering

- Clustering is used:
  - As a stand-alone tool to get insight into data distribution
    - Visualization of clusters may unveil important information
    - Applications mentioned ahead
  - As a preprocessing step for other algorithms
    - Efficient indexing or compression often relies on clustering
    - Data cleaning

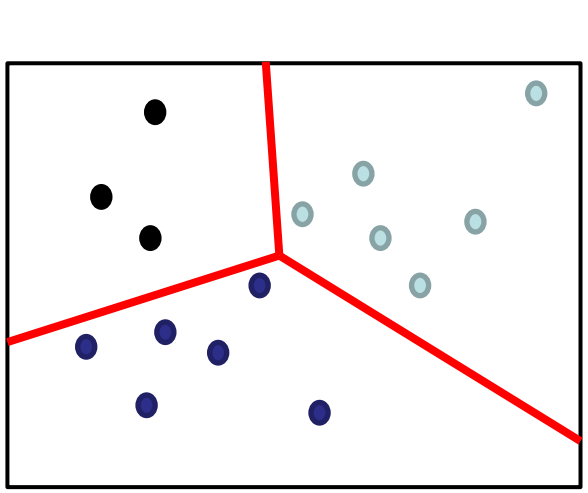
Outlier Analysis by Clustering

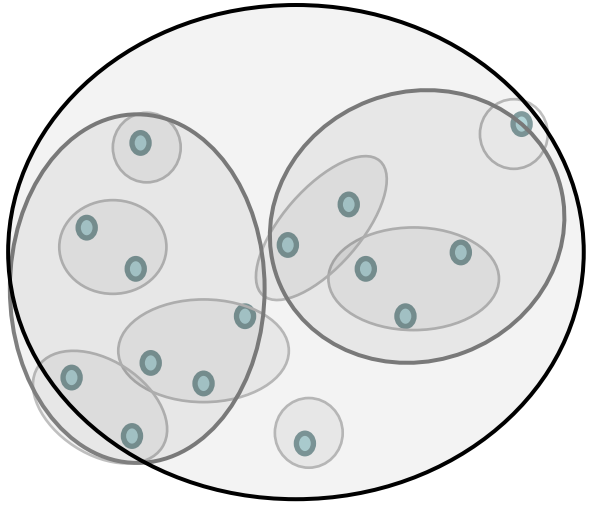
- Outliers are objects that do not belong to any cluster or form clusters of very small cardinality

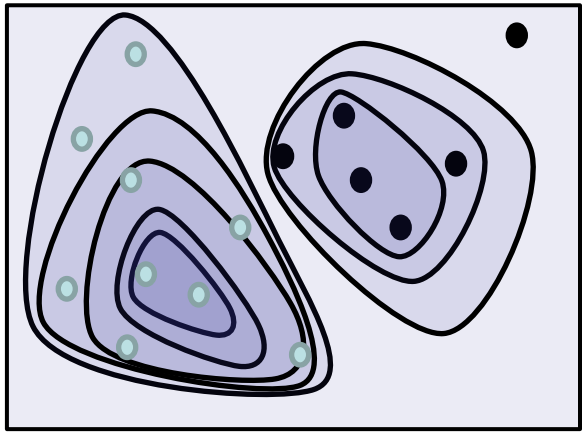


- In some applications we are interested in discovering outliers, not clusters **(outlier analysis)**

Clustering Algorithms Overview



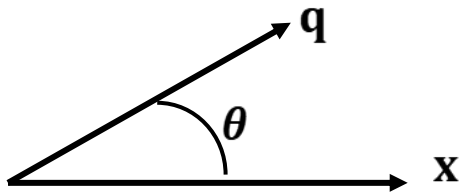




Partitioning Method	Hierarchical Clustering	Density-Based Method
	• Creates a hierarchical decomposition of the given set of data objects.	• Continue **growing** a given cluster as long as the **density** (number of objects or data points) in the "neighborhood" exceeds some threshold.

Distances on Numerical Values

- Distance (dissimilarity) is inversely proportional to similarity
- Cosine Similarity:
  - CosSim($\mathbf{q}, \mathbf{x}$)

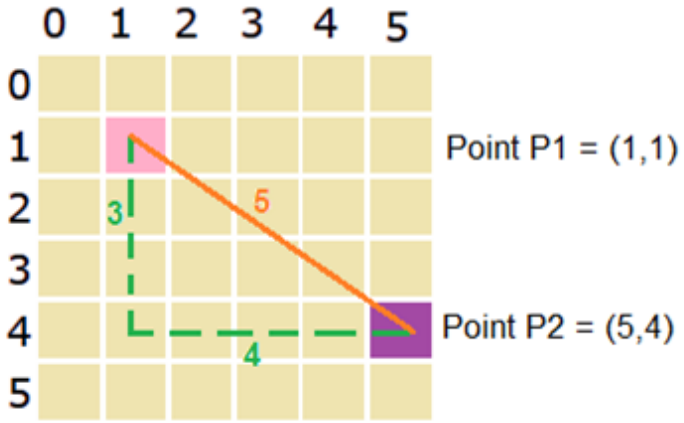


Cosine similarity

- Distance measures between p-dimensional vectors
- A p-dim vector is an object (point) in the p-dim space

Name	Definition
Minkowski	$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt[N]{\lvert x_{i1} - x_{j1}\rvert^N + \lvert x_{i2} - x_{j2}\rvert^N + \cdots + \lvert x_{ip} - x_{jp}\rvert^N}$
Manhattan (N=1)	$d(\mathbf{x}_i, \mathbf{x}_j) = \lvert x_{i1} - x_{j1}\rvert + \lvert x_{i2} - x_{j2}\rvert + \ \ldots \ + \lvert x_{ip} - x_{jp}\rvert$
Euclidean (N=2)	$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt[2]{\lvert x_{i1} - x_{j1}\rvert^2 + \lvert x_{i2} - x_{j2}\rvert^2 + \cdots + \lvert x_{ip} - x_{jp}\rvert^2}$

Distances



Euclidean distance = $\sqrt{(5-1)^2 + (4-1)^2} = 5$

Manhattan distance = $|5-1| + |4-1| = 7$

Name	Definition
Minkowski	$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt[N]{\lvert x_{i1} - x_{j1}\rvert^N + \lvert x_{i2} - x_{j2}\rvert^N + \cdots + \lvert x_{ip} - x_{jp}\rvert^N}$
Manhattan (N=1)	$d(\mathbf{x}_i, \mathbf{x}_j) = \lvert x_{i1} - x_{j1}\rvert + \lvert x_{i2} - x_{j2}\rvert + \ \ldots \ + \lvert x_{ip} - x_{jp}\rvert$
Euclidean (N=2)	$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt[2]{\lvert x_{i1} - x_{j1}\rvert^2 + \lvert x_{i2} - x_{j2}\rvert^2 + \cdots + \lvert x_{ip} - x_{jp}\rvert^2}$

Euclidean Distance

- Euclidean distance:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt[2]{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \ldots + |x_{ip} - x_{jp}|^2}$$

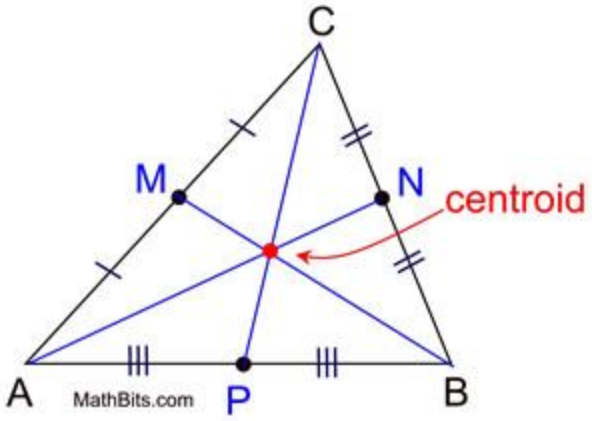
  - Properties
    - $d(\mathbf{x}_i, \mathbf{x}_j) \geq 0$ (**non-negativity**)
    - $d(\mathbf{x}_i, \mathbf{x}_i) = 0$ (**coincidence**)
    - $d(\mathbf{x}_i, \mathbf{x}_j) = d(\mathbf{x}_j, \mathbf{x}_i)$ (**symmetry**)
    - $d(\mathbf{x}_i, \mathbf{x}_j) \leq d(\mathbf{x}_i, \mathbf{x}_k) + d(\mathbf{x}_k, \mathbf{x}_j)$ (**triangular inequality**)
- Also one can use **weighted** distance:

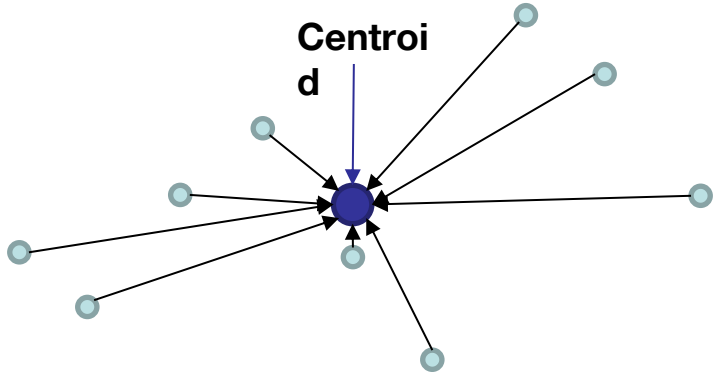
$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt[2]{w_1|x_{i1} - x_{j1}|^2 + w_2|x_{i2} - x_{j2}|^2 + \ldots + w_p|x_{ip} - x_{jp}|^2}$$

Basic Concepts of Clustering

- The **centroid** or geometric center of a plane figure is the arithmetic mean position of all the points in the shape.
- The centroid of n points $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}$ is calculated as

$$\mathbf{c} = (\mathbf{x_1} + \mathbf{x_2} + ... + \mathbf{x_n})/\boldsymbol{n}$$





Exercise: Centroid Computation

- $\mathbf{x_1} = [2,6,1]$
- $\mathbf{x_2} = [3,9,4]$
- $\mathbf{x_3} = [8,3,7]$
- Compute the centroid of the three vectors

Centroid **may not** correspond to an object in the database

Outline

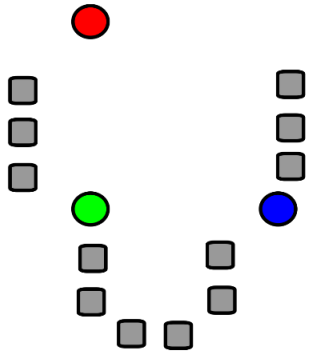
- Start with Text Analytics
- Unsupervised Algorithms
  - Clustering
    - Applications and Concepts
    - K-Means
    - DBSCAN
    - Hierarchical Clustering

K-Means

- Partitioning method: Construct a partition of a database ***D*** of ***n*** objects (tuples/points) into a set of ***k*** clusters
  - *k-means* (MacQueen'67): Each cluster is represented by the center of the cluster (i.e., centroid of the cluster)
- There are many k-means variants; we focus on k-means using centroid here

K-Means (n=12, k=3 example)

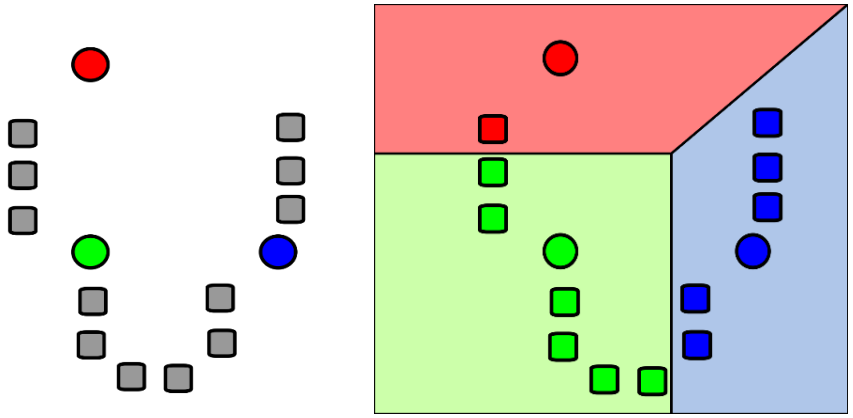
- Step 1: *k* initial random centroids in the data domain (in color)
  - Objects are in gray squares; colored circles are centroids



By I, Weston.pace, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=2463076

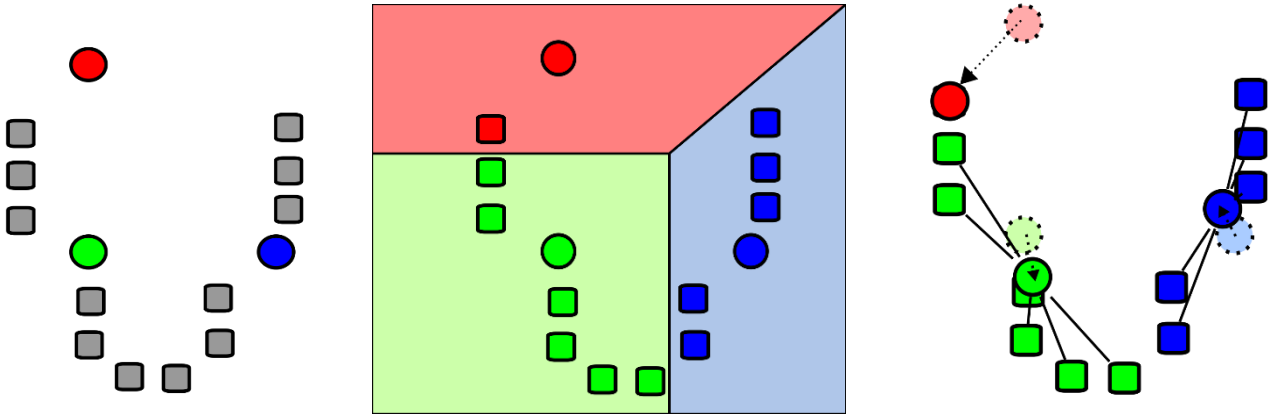
K-Means (n=12, k=3 example)

- Step 2: assign objects to nearest centroid to form clusters.



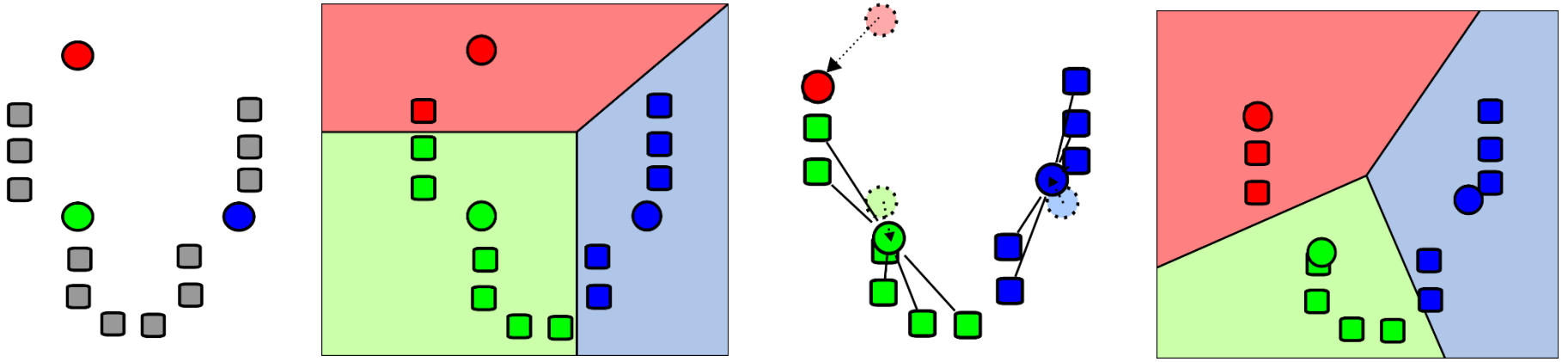
K-Means (n=12, k=3 example)

- Step 3: update centroids by computing the mean of each cluster



K-Means (n=12, k=3 example)

- Step 4: go to step 2 (assign objects to nearest centroid to form clusters)



- Repeat steps 2 to 4 until convergence

K-Means

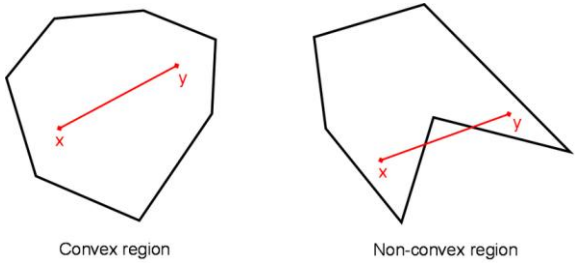
- Pseudo-code

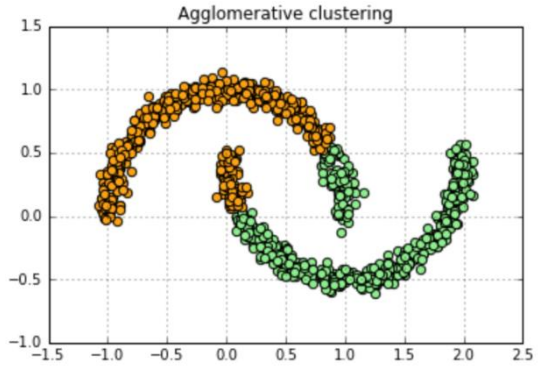
Algorithm 1 k-means algorithm

1: Specify the number k of clusters to assign.
2: Randomly initialize k centroids.
3: **repeat**
4: **expectation:** Assign each point to its closest centroid.
5: **maximization:** Compute the new centroid (mean) of each cluster.
6: **until** The centroid positions do not change.

Comments on the k-means Method

- Strength
  - *Relatively efficient*: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, k, $t << n$.
- Weakness
  - Applicable only when *mean* is defined (what about categorical data)?
  - Need to specify *k,* the *number* of clusters, in advance
  - Unable to handle noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*





Outline

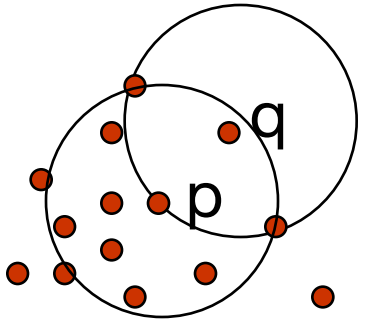
- Start with Text Analytics
- Unsupervised Algorithms
  - Clustering
    - Applications and Concepts
    - K-Means
    - DBSCAN
    - Hierarchical Clustering

Density-Based Clustering Methods

- Clustering based on **density** (local cluster criterion), such as density-connected points
- Major features:
  - Discover clusters of arbitrary shape (while k-means can only discover convex-shaped clusters)
  - Handle noise (while k-means cannot)
  - One scan (while k-means works iteratively)
  - Need density parameters as termination condition
- Several interesting studies:
  - **DBSCAN**: Ester, et al. (KDD'96)
  - OPTICS: Ankerst, et al (SIGMOD'99).
  - DENCLUE: Hinneburg & D. Keim (KDD'98)
  - CLIQUE: Agrawal, et al. (SIGMOD'98)

Density-Based Clustering: Concepts

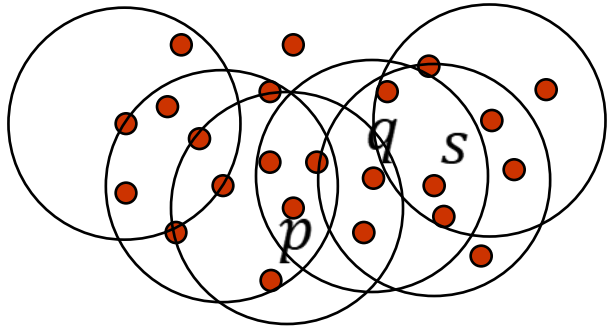
- **Eps-neighborhood** of point p = {all points within distance Eps from p}
  - $N_{Eps}(p) = \{q \mid dist(p,q) <= Eps\}$
  - Eps: Maximum **radius** of the neighborhood
- If the number of points in the Eps-neighborhood of p is at least $MinPts$ (i.e., $|N_{Eps}(p)| \geq MinPts$), then p is called a **core** object.
  - $MinPts$: Min number of points to make a Eps-neighborhood dense
  - $MinPts$ and Eps specify the density requirement
    - Intuitively There should be sufficient number of points in an area
- If an object q is not a core point, but it belongs to the Eps-neighborhood of a core point, then q is a **border** object



MinPts = 5
Eps = 1 cm

Density-Based Clustering

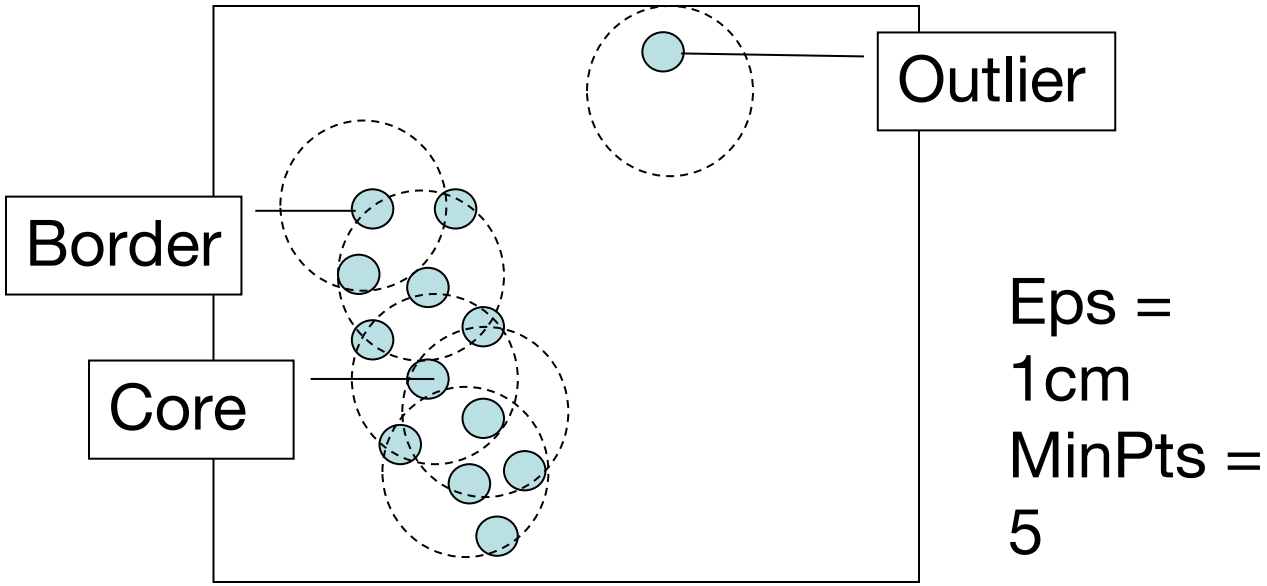
- A **core** point and its ***Eps*-neighborhood** define a **cluster**.
- If two **core points** p and q **belong to** the Eps-neighborhoods of **each other**, **merge** the corresponding clusters
- Continue the merge until no clusters can be merged



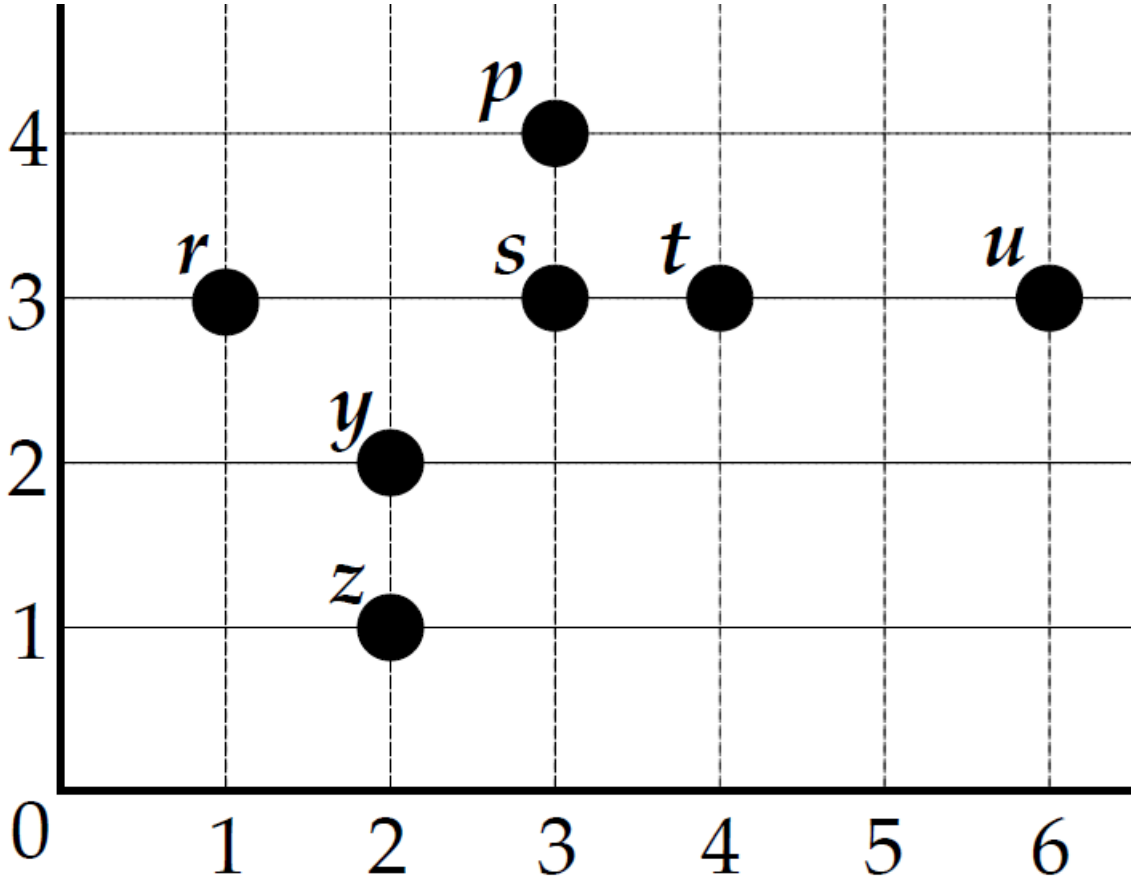
MinPts = 5
Eps = 1 cm

DBSCAN: Density Based Spatial Clustering of Applications with Noise

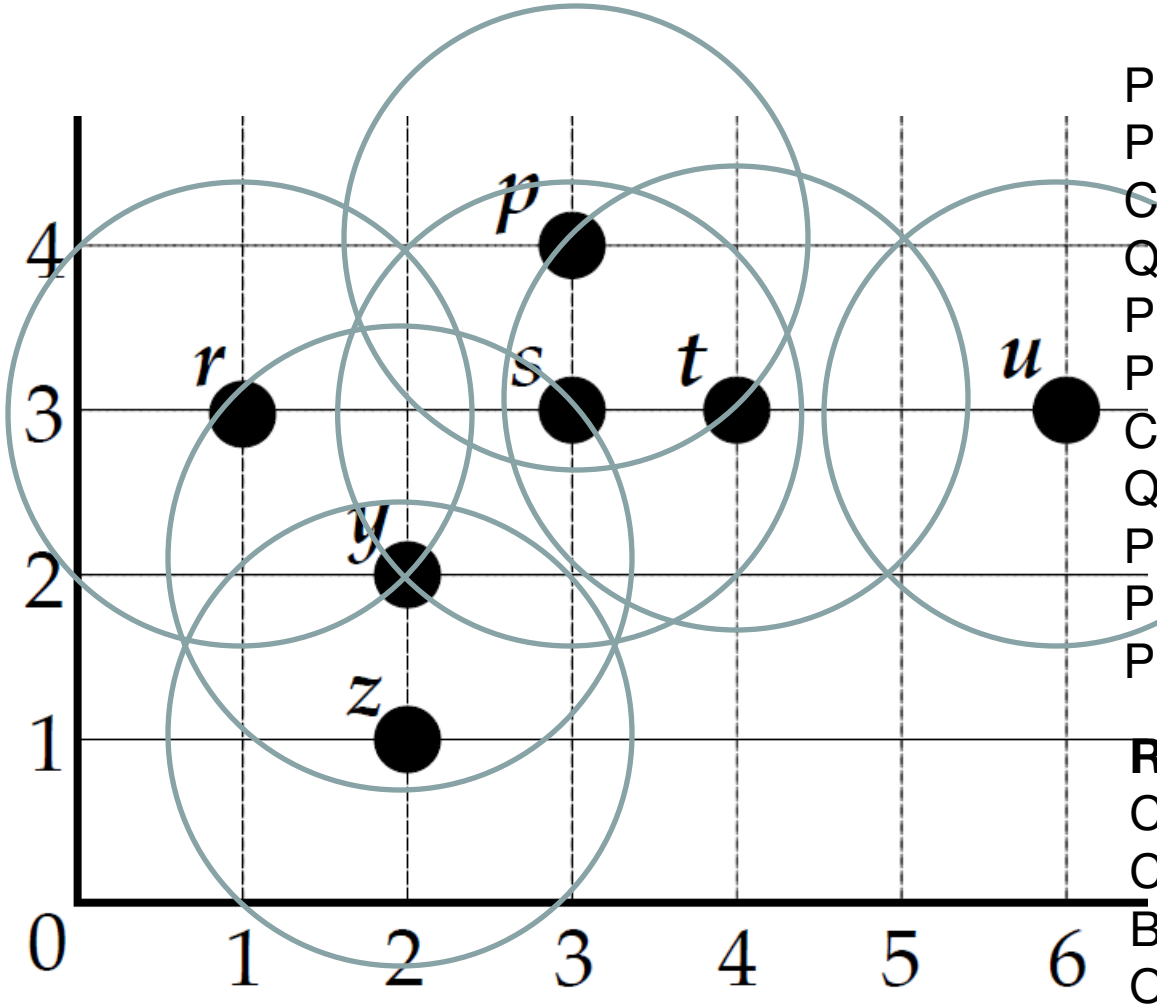
- Discovers clusters of arbitrary shape with noise
- In the end, if an object does not belong to any cluster, then it is an **outlier/noise**



Exercise: DBSCAN with MinPts=4 and EPS=$\sqrt{2}$



Exercise: DBSCAN with MinPts=4 and EPS=$\sqrt{2}$



Process z
Process y
Cluster 1: {y,z,r,s}
Queue: r,s
Process r
Process s
Cluster 1: {z,y,r,s,p,t}
Queue: p,t
Process p
Process t
Process u

Result:
Cluster: {z,y,r,s,p,t}
Core: y, s
Border: z,r,p,t
Outlier: u

DBSCAN: The Algorithm

1. Set all points as UNPROCESSED.
2. Select an UNPROCESSED point p.
3. Find Eps-Neighborhood of p using parameter Eps.
4. If p is a **core** point (based on $MinPts$), a cluster is formed. p is set as CORE.
   A. Put all UNPROCESSED points in Eps-Neighborhood of p in a queue S.
   B. For each point $q \in S$, check whether it is a **core** point.
      a. if yes, set q as CORE, merge two clusters.
         Note: Points in q's neighborhood will be added to S
      b. if no, set q as BORDER.
      Repeat until S is empty.
5. If p is NOT a core point, set p as **UNASSIGNED** to a cluster
   A. p may be included to a cluster later if it is found to be in the Eps-Neighborhood of a core point (**BORDER**)
      (If p is not in any clusters when terminates, it is set as **OUTLIER**)
6. If there are other UNPROCESSED points, goto 2.

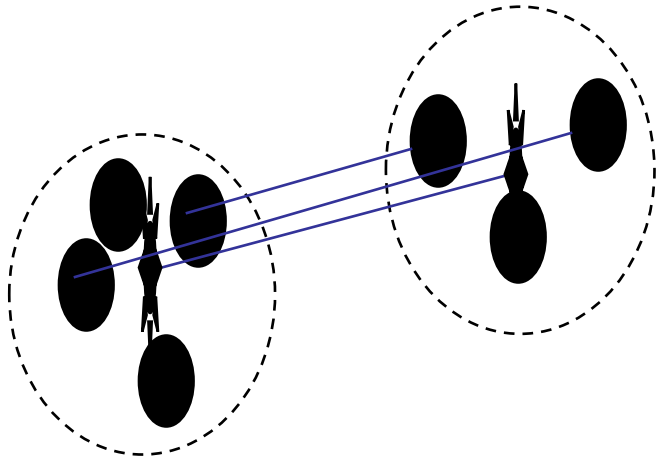
The status of a point: UNPROCESSED, CORE, BORDER, OUTLIER, UNASSIGNED (intermediate status)

Outline

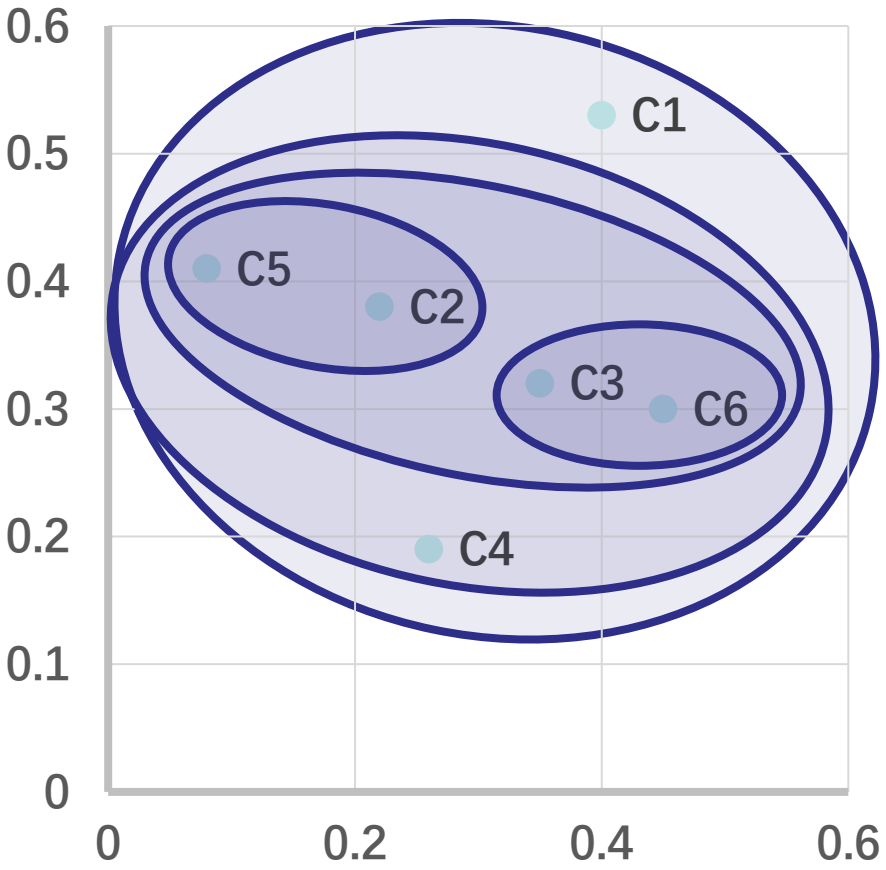
- Start with Text Analytics
- Unsupervised Algorithms
  - Clustering
    - Applications and Concepts
    - K-Means
    - DBSCAN
    - Hierarchical Clustering

Distance Between Clusters

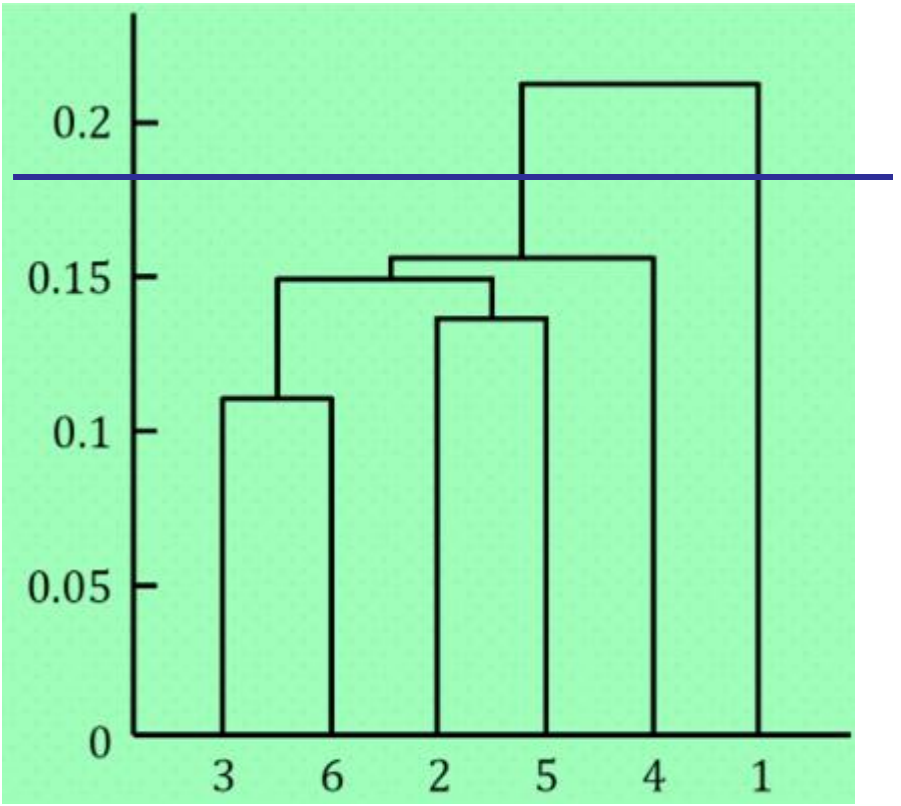
- Single Link: smallest distance between any points in two clusters
- Complete Link: largest distance between any points in two clusters
- Centroid: distance between the centroids of two clusters
- Average Link: average distance of all pairwise points in clusters
  - Average of the distances of the 4*3 pairs of points in the example



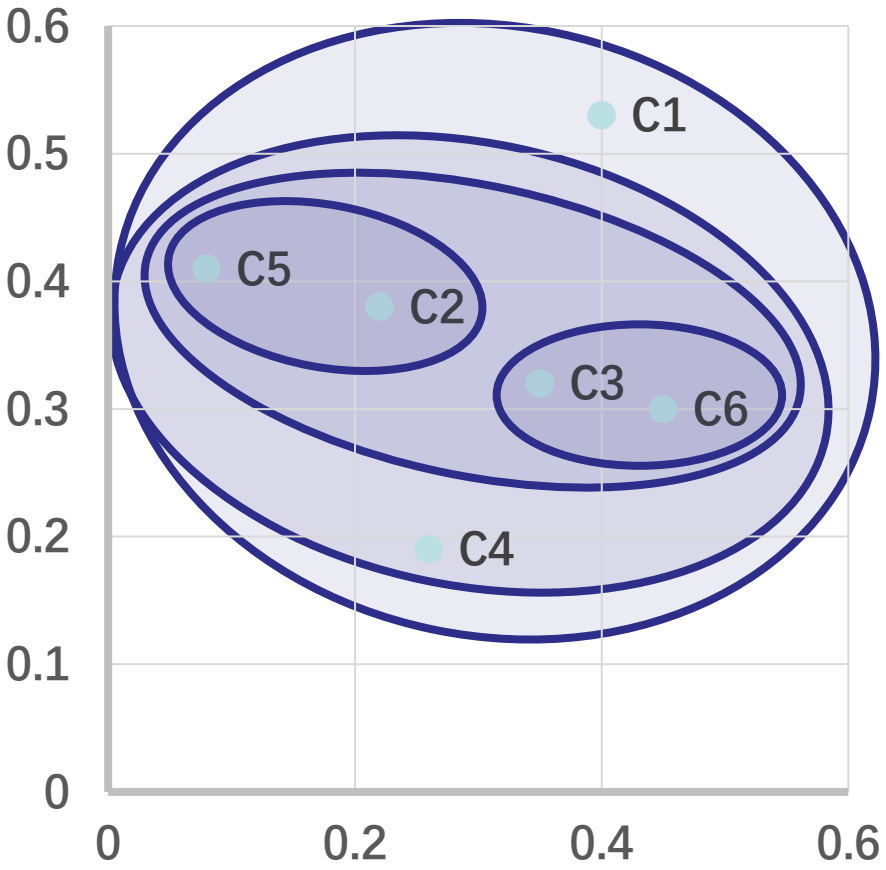
Example of Hierarchical Clustering

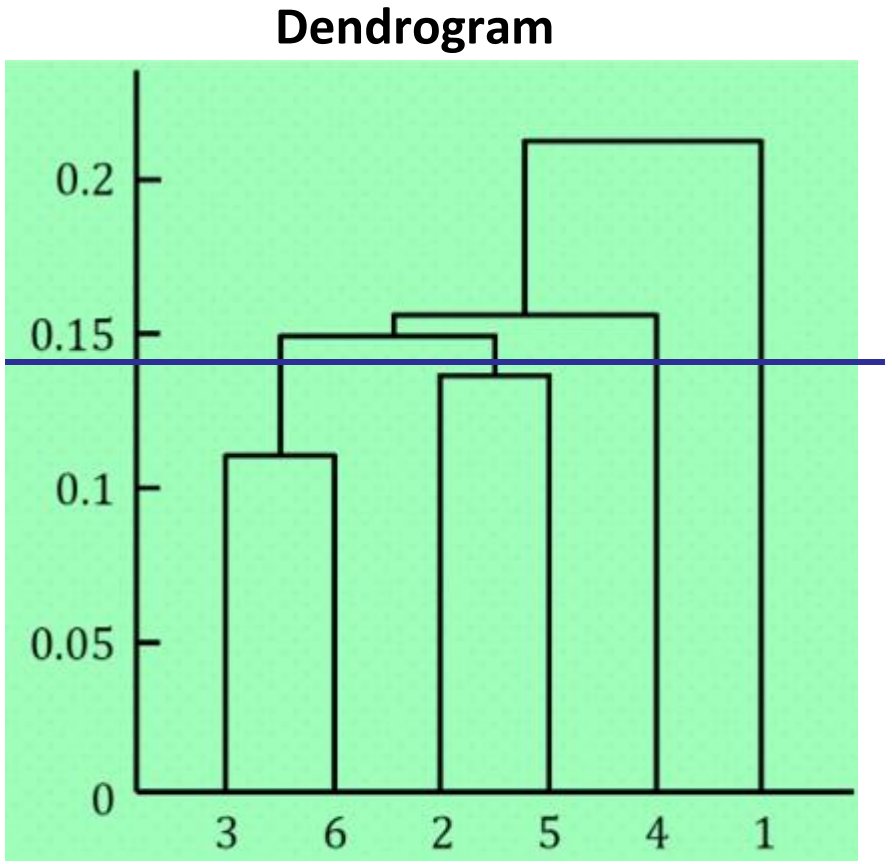


Dendrogram



Example of Hierarchical Clustering

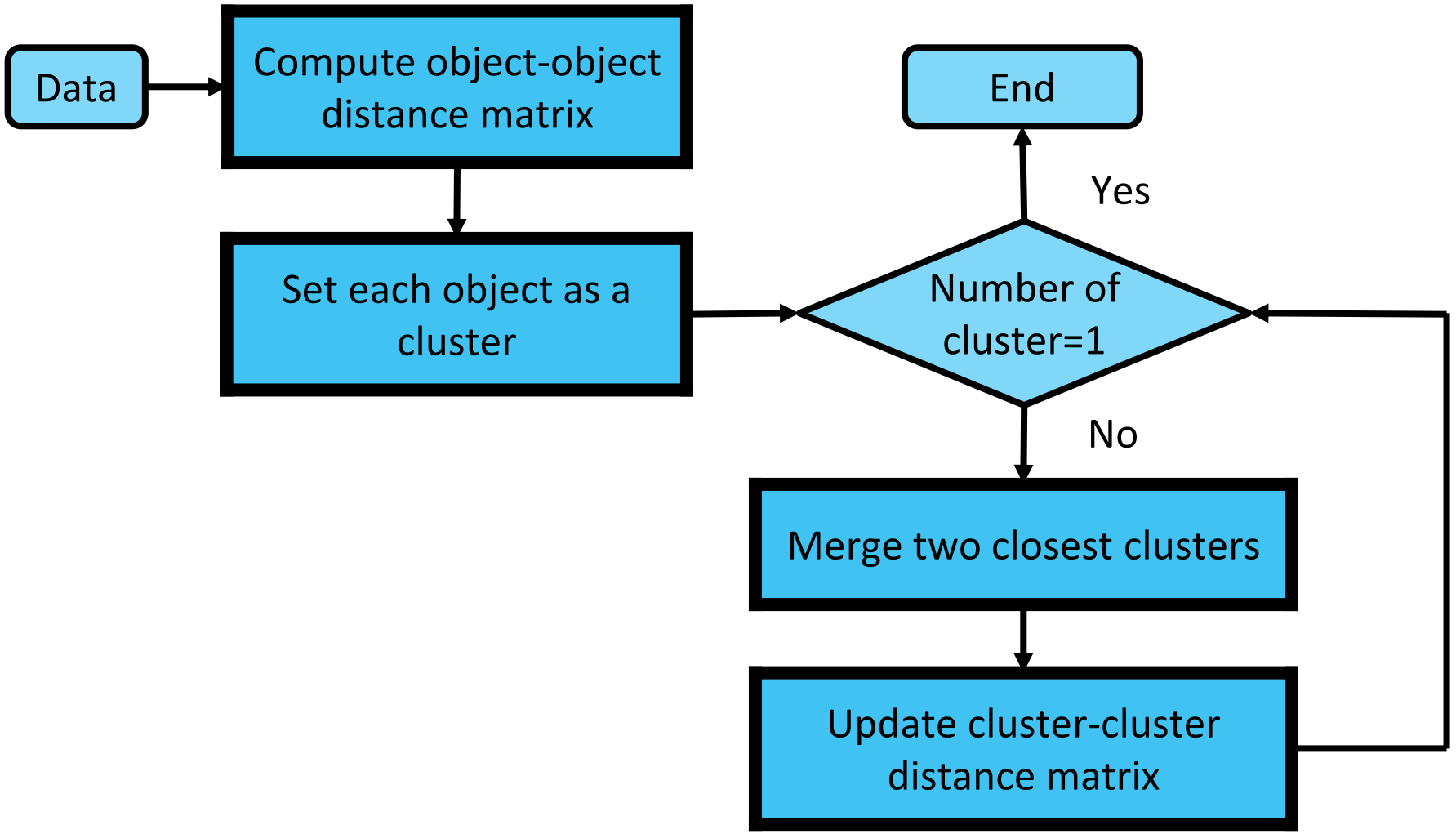




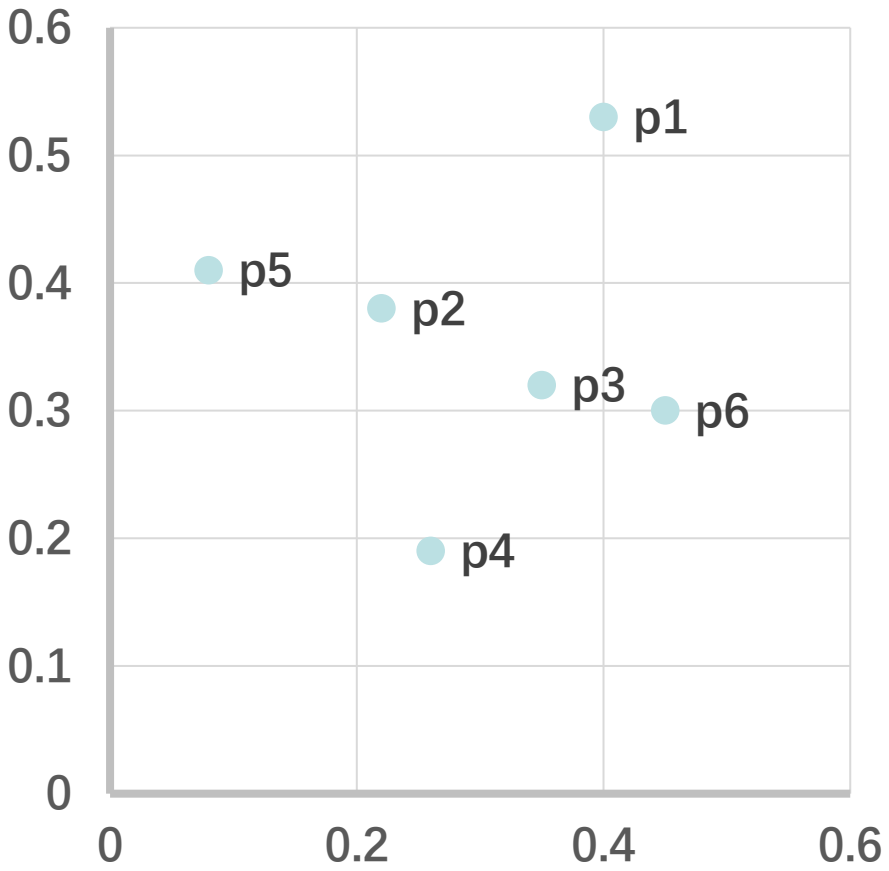
Hierarchical Clustering

- Clusters are created in levels
  - (actually, creating sets of clusters at each level).
- ***Agglomerative***
  - Initially each item in its own cluster
  - Iteratively clusters are merged together
  - Bottom Up
- ***Divisive***
  - Initially all items in one cluster
  - Large clusters are successively divided
  - Top Down

Agglomerative Hierarchical Clustering



Agglomerative Hierarchical Clustering with Single Link Cluster Distance



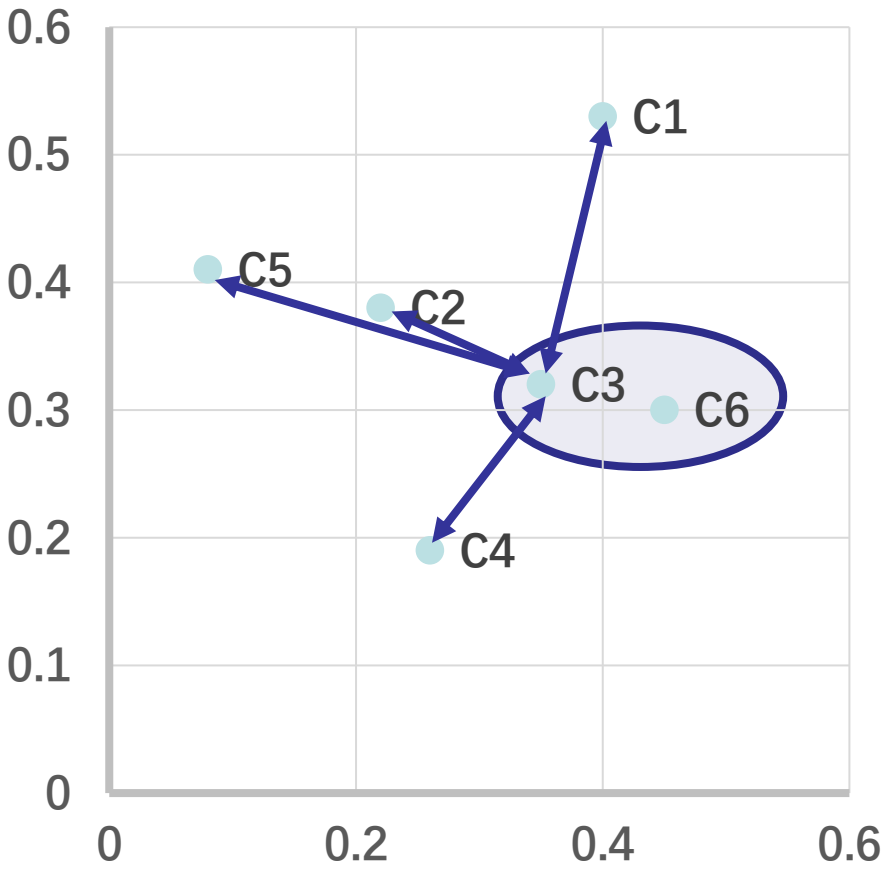
Data

Point	X	Y
p1	0.4	0.53
p2	0.22	0.38
p3	0.35	0.32
p4	0.26	0.19
p5	0.08	0.41
p6	0.45	0.3

Distance Matrix

Cluster	C1	C2	C3	C4	C5	C6
C1	0.00	0.24	0.22	0.37	0.34	0.23
C2		0.00	0.15	0.20	0.14	0.25
C3			0.00	0.15	0.28	0.11
C4				0.00	0.29	0.22
C5					0.00	0.39
C6						0.00

Agglomerative Hierarchical Clustering with Single Link Cluster Distance

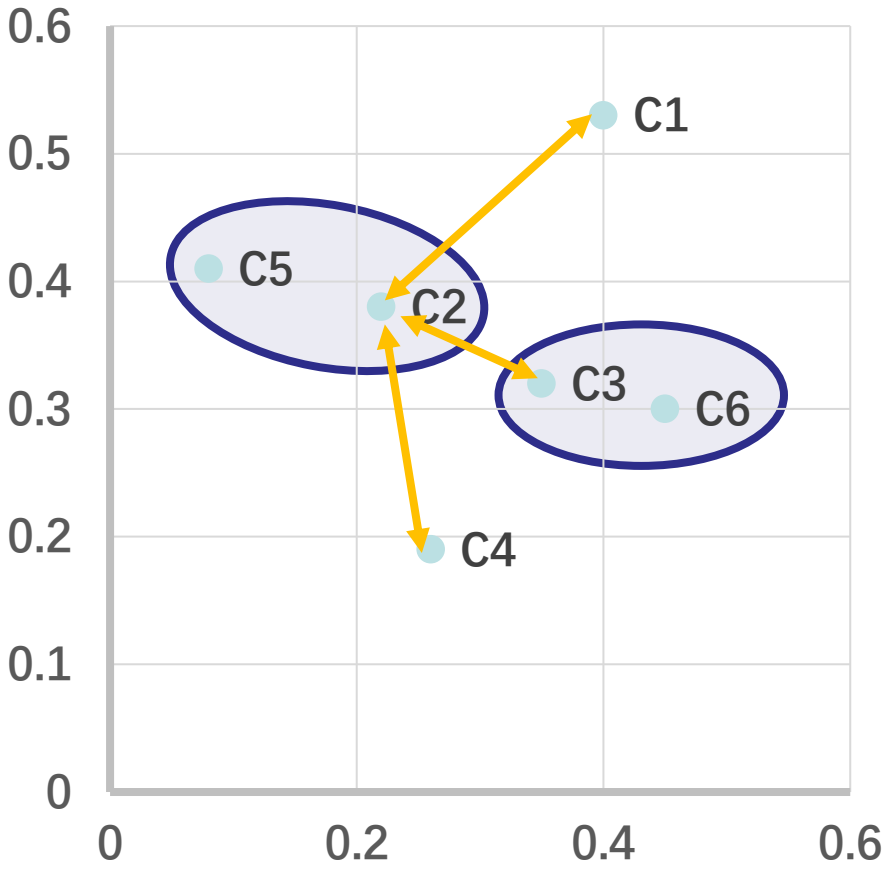


Distance Matrix

Cluster	C1	C2	C3	C4	C5	C6
C1	0.00	0.24	0.22	0.37	0.34	0.23
C2		0.00	0.15	0.20	0.14	0.25
C3			0.00	0.15	0.28	0.11
C4				0.00	0.29	0.22
C5					0.00	0.39
C6						0.00

Cluster	C1	C2	C4	C5	C3,6
C1	0.00	0.24	0.37	0.34	0.22
C2		0.00	0.20	0.14	0.15
C4			0.00	0.29	0.15
C5				0.00	0.28
C3,6					0.00

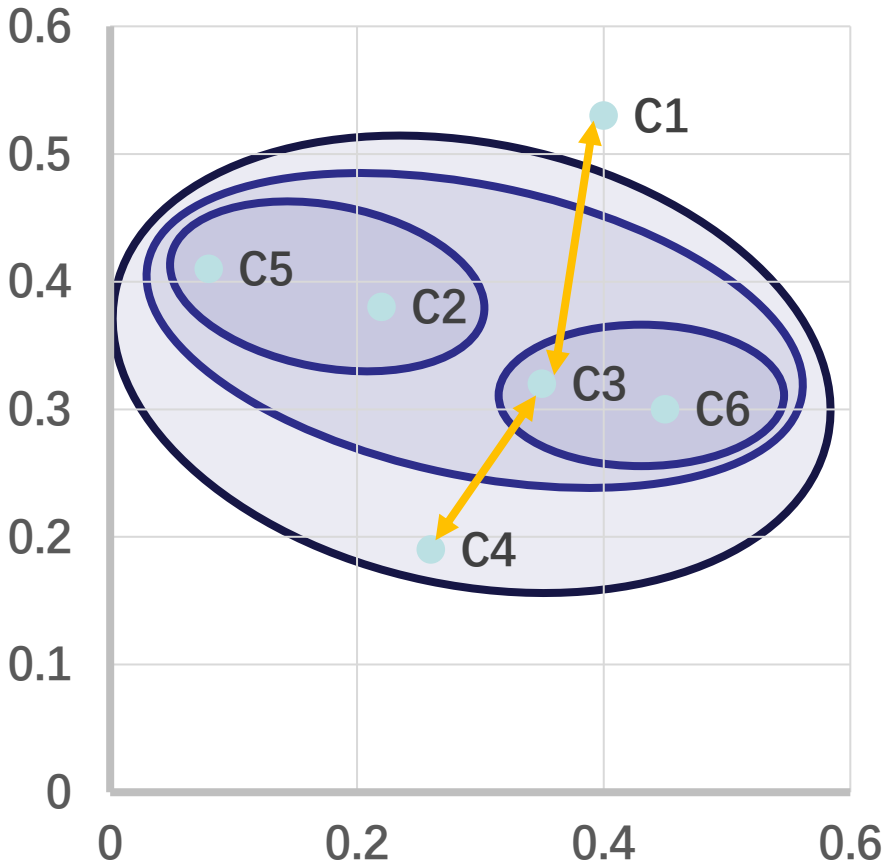
Agglomerative Hierarchical Clustering with Single Link Cluster Distance



Cluster	C1	C2	C4	C5	C3,6
C1	0.00	0.24	0.37	0.34	0.22
C2		0.00	0.20	0.14	0.15
C4			0.00	0.29	0.15
C5				0.00	0.28
C3,6					0.00

Cluster	C1	C4	C3,6	C2,5
C1	0.00	0.37	0.22	0.24
C4		0.00	0.15	0.20
C3,6			0.00	0.15
C2,5				0.00

Agglomerative Hierarchical Clustering with Single Link Cluster Distance

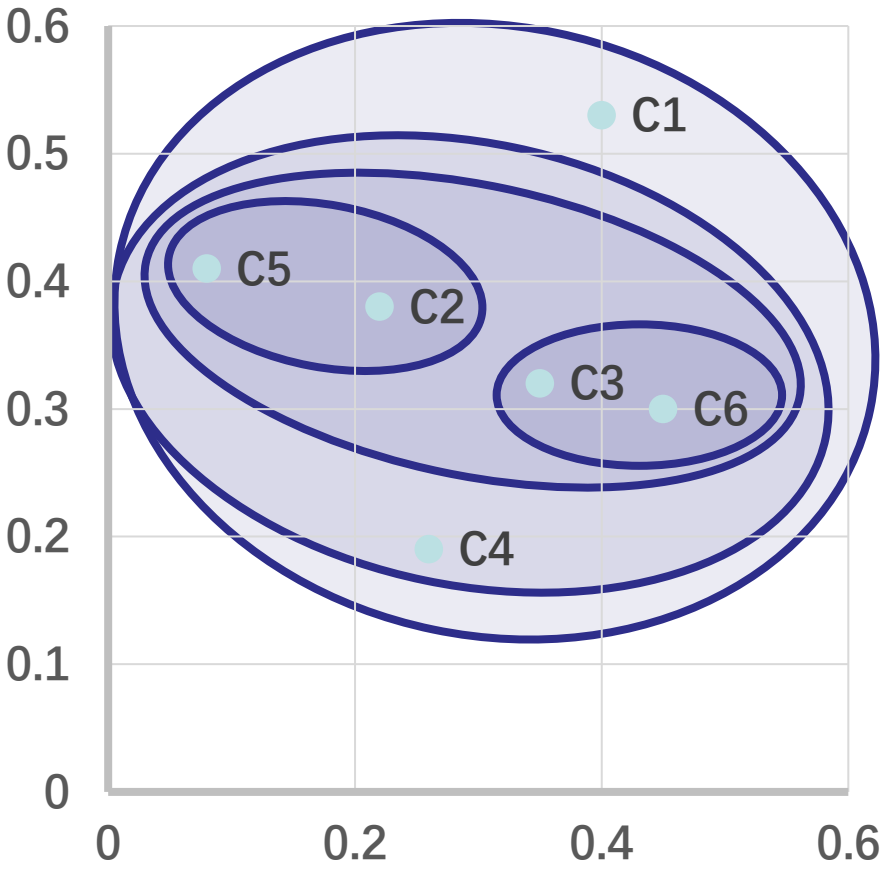


Cluster	C1	C4	C3,6	C2,5
C1	0.00	0.37	0.22	0.24
C4		0.00	0.15	0.20
C3,6			0.00	0.15
C2,5				0.00

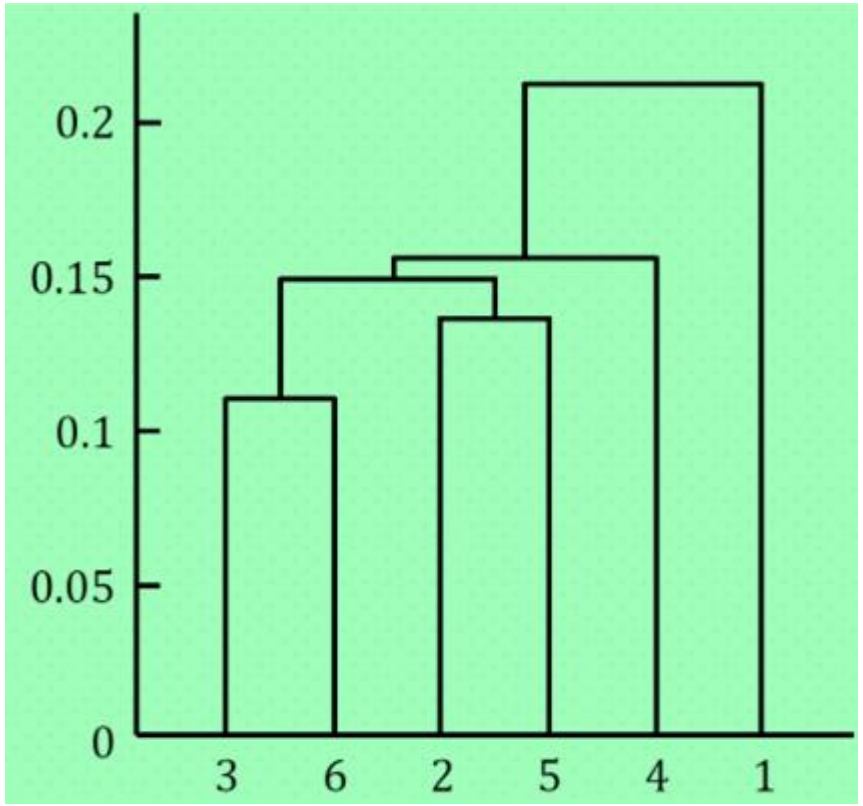
Cluster	C1	C4	C3,6,2,5
C1	0.00	0.37	0.22
C4		0.00	0.15
C3,6,2,5			0.00

Cluster	C1	C3,6,2,5,4
C1	0.00	0.22
C3,6,2,5,4		0.00

Agglomerative Hierarchical Clustering with Single Link Cluster Distance



Dendrogram



Thank you!