# Knowledge Discovery and Data Mining 1 (VO) (707.003)

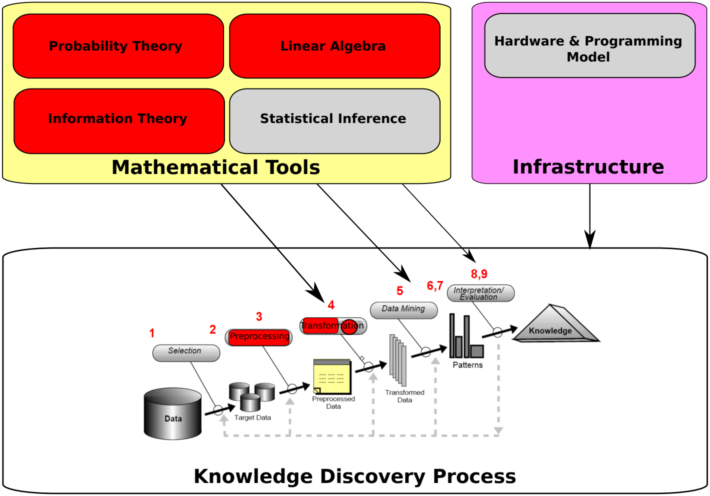
## Data Matrices and Vector Space Model

Denis Helic

KTI, TU Graz

Nov 6, 2014

# Big picture: KDDM



# Outline

# Recap

## Recap

Review of the preprocessing, feature extraction and selection phase

# Recap – Preprocessing

- Initial phase of the Knowledge Discovery process
- ... acquire the data to be analyzed
- e.g. by **crawling** the data from the Web
- ... prepare the data
- e.g. by **cleaning** and **removing outliers**

# Recap – Feature Extraction

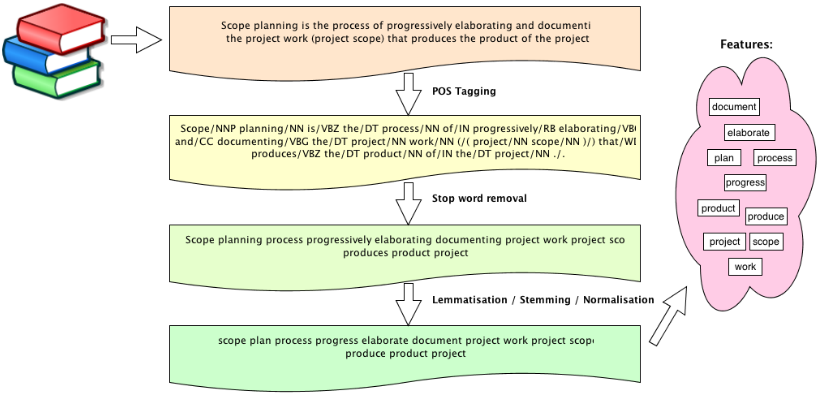
- Example of features:
- Images → colors, textures, contours, ...
- Signals → frequency, phase, samples, spectrum, ...
- Time series → ticks, trends, self-similarities, ...
- Biomed → dna sequence, genes, ...
- Text → words, POS tags, grammatical dependencies, ...

Features encode these properties in a way suitable for a chosen algorithm

# Recap – Feature Selection

- Approach: select the sub-set of all features without redundant or irrelevant features
  - Unsupervised, e.g. heuristics (black & white lists, score for ranking, . . . )
  - Supervised, e.g. using a training data set (calculate measure to assess how discriminative is a feature, e.g. information gain)
  - Penalty for more features, e.g. regularization

# Recap – Text mining



# Representing data

- Once when we have the features that we want: how can we represent them?
- Two representations:
  1. Mathematical (model, analyze, and reason about data and algorithms in a formal way)
  2. Representation in computers (data structures, implementation, performance)
- In this course we concentrate on mathematical models
- KDDM2 is about second representation

# Representing data

- **Given:** Preprocessed data objects as a set of features
- E.g. for text documents set of words, bigrams, n-grams, ...
- **Given:** Feature statistics for each data object
- E.g. number of occurrences, magnitudes, ticks, ...
- **Find:** Mathematical model for calculations
- E.g. similarity, distance, add, subtract, transform, ...

# Representing data

- Let us think (geometrically) about features as dimensions (coordinates) in an $m$-dimensional space
- For two features we have 2D-space, for three features we have 3D-space, and so on
- For numeric features the feature statistics will be numbers coming from e.g. $\mathbb{R}$
- Then each data object is a point in the $m$-dimensional space
- The value of a given feature is then the value for its coordinate in this $m$-dimensional space

# Representing data: Example

## Text documents

Suppose we have the following documents:

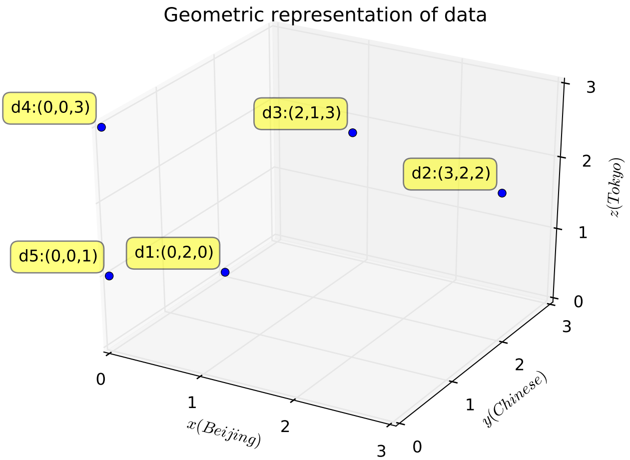
| DocID | Document |
|---:|---:|
| d1 | Chinese Chinese |
| d2 | Chinese Chinese Tokyo Tokyo Beijing Beijing Beijing |
| d3 | Chinese Tokyo Tokyo Tokyo Beijing Beijing |
| d4 | Tokyo Tokyo Tokyo |
| d5 | Tokyo |

# Representing data: Example

- Now, we take words as features
- We take word occurrences as the feature values
- Three features: *Beijing*, *Chinese*, *Tokyo*

| Feature / Doc | *Beijing* | *Chinese* | *Tokyo* |
|---:|---:|---:|---:|
| d1 | 0 | 2 | 0 |
| d2 | 3 | 2 | 2 |
| d3 | 2 | 1 | 3 |
| d4 | 0 | 0 | 3 |
| d5 | 0 | 0 | 1 |

# Representing data: Example



# Representing data

- Now, an intuitive representation of the data is a matrix
- In a general case
- Columns correspond to features, i.e. dimensions or coordinates in an $m$-dimensional space
- Rows correspond to data objects, i.e. data points
- An element $d_{ij}$ in the $i$-th row and the $j$-th column is the $j$-th coordinate of the $i$-th data point
- The data is represented by a matrix $\mathbf{D} \in \mathbb{R}^{n \times m}$, where $n$ is the number of data points and $m$ the number of features

# Representing data

- For text
- Columns correspond to terms, words, and so on
- I.e. each word is a dimension or a coordinate in an $m$-dimensional space
- Rows correspond to documents
- An element $d_{ij}$ in the $i$-th row and the $j$-th column is the e.g. number of occurrences of the $j$-th word in the $i$-th document
- This matrix is called **document-term** matrix

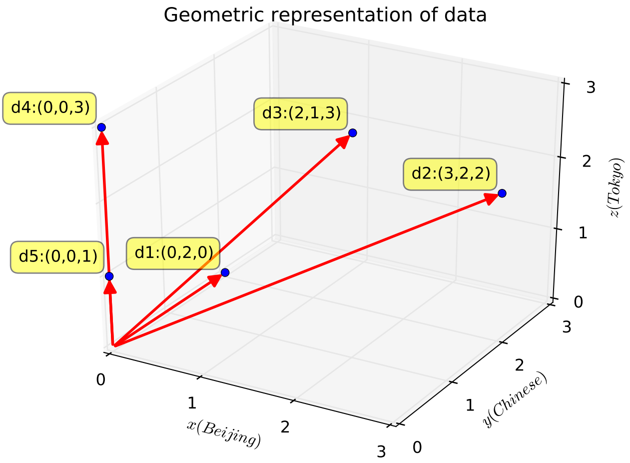
# Representing data: example

$$\mathbf{D} = \begin{pmatrix} 0 & 2 & 0 \\ 3 & 2 & 2 \\ 2 & 1 & 3 \\ 0 & 0 & 3 \\ 0 & 0 & 1 \end{pmatrix}$$

# Vector interpretation

- Alternative and very useful interpretation of data matrices is in terms of vectors
- Each data point, i.e. each row in the data matrix can be interpreted as a vector in an $m$-dimensional space
- This allows us to work with:
  1. the vector direction, i.e. the line connecting the origin and a given data point
  2. the vector magnitude (length), i.e. the distance from the origin and a given data point $\rightarrow$ feature weighting
  3. similarity, distance, correlations between vectors
  4. manipulate vectors, and so on.
- **Vector space model**

# Vector interpretation: Example



# Vector space model: feature weighting

- What we did so far is that we counted word occurrences and used these counts as the coordinates in a given dimension
- This weighting scheme is referred to as *term frequency* (TF)
- Three features: *Beijing*, *Chinese*, *Tokyo*

| Doc \ Feature | *Beijing* | *Chinese* | *Tokyo* |
|---:|---:|---:|---:|
| d1 | 0 | 2 | 0 |
| d2 | 3 | 2 | 2 |
| d3 | 2 | 1 | 3 |
| d4 | 0 | 0 | 3 |
| d5 | 0 | 0 | 1 |

# Vector space model: TF

- Please note the difference between the vector space model with TF weighting and e.g. Standard boolean model in information retrieval
- Standard boolean model uses binary feature values
- If the feature $j$ occurs in the document $i$ (regardless how many times) then $d_{ij} = 1$
- Otherwise $d_{ij} = 0$
- Allows very simple manipulation of the model, but does not capture the relative importance of the feature $j$ for the document $i$
- If a feature occurs more frequently in a document then it is more important for that document and TF captures this intuition

# Vector space model: feature weighting

- However, raw TF has a critical problem: all terms are considered equally important for assessing relevance
- In fact, certain terms have no discriminative power at all considering relevance
- For example, a collection of documents on the auto industry will have most probably term "auto" in every document
- We need to penalize terms which occur too often in the complete collection
- We start with the notion of *document frequency* (DF) of a term
- This is the number of documents that contain a given term

# Vector space model: IDF

- If a term is discriminative then it will only appear in a small number of documents
- I.e. its DF will be low, or its *inverse document frequency* (IDF) will be high
- In practice we typically calculate IDF as:

$$IDF_j = log(\frac{N}{DF_j}),$$

- where $N$ is the number of documents and $DF_j$ is the document frequency of the term $j$.

# Vector space model: TFxIDF

- Finally, we combine TF and IDF to produce a composite weight for a given feature

$$TFxIDF_{ij} = TF_{ij} log(\frac{N}{DF_j})$$

- Now, this quantity will be high if the feature $j$ occurs only a couple of times in the whole collection, and most of these occurrences are in the document $i$

# TFxIDF: Example

## Text documents

Suppose we have the following documents:

| DocID | Document |
|---|---|
| d1 | Chinese Chinese |
| d2 | Chinese Chinese Tokyo Tokyo Beijing Beijing Beijing |
| d3 | Chinese Tokyo Tokyo Tokyo Beijing Beijing |
| d4 | Tokyo Tokyo Tokyo |
| d5 | Tokyo |

# TFxIDF: Example

- Three features: *Beijing*, *Chinese*, *Tokyo*
- TF:

| Doc \ Feature | *Beijing* | *Chinese* | *Tokyo* |
|---|---|---|---|
| d1 | 0 | 2 | 0 |
| d2 | 3 | 2 | 2 |
| d3 | 2 | 1 | 3 |
| d4 | 0 | 0 | 3 |
| d5 | 0 | 0 | 1 |

# TFxIDF: Example

- Three features: *Beijing*, *Chinese*, *Tokyo*
- DF:

$$
\begin{aligned}
DF_{Beijing} &= 2 \\
DF_{Chinese} &= 3 \\
DF_{Tokyo} &= 4
\end{aligned}
$$

# TFxIDF: Example

- Three features: *Beijing*, *Chinese*, *Tokyo*
- IDF:

$$IDF_j = log(\frac{N}{DF_j})$$

$$\begin{aligned} IDF_{Beijing} &= 0.3979 \\ IDF_{Chinese} &= 0.2218 \\ IDF_{Tokyo} &= 0.0969 \end{aligned}$$

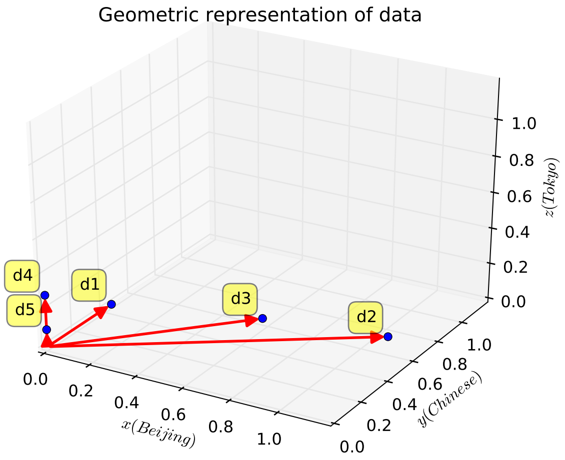
# TFxIDF: Example

- Three features: *Beijing*, *Chinese*, *Tokyo*
- TFxIDF:

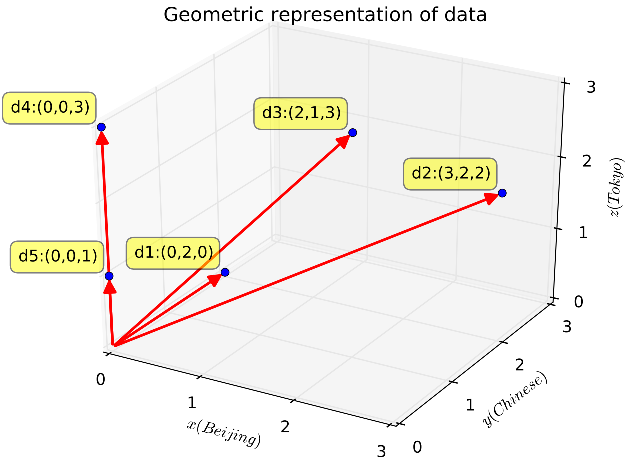
$$TFxIDF_{ij} = TF_{ij} log(\frac{N}{DF_j})$$

| Doc \ Feature | *Beijing* | *Chinese* | *Tokyo* |
|---|---|---|---|
| d1 | 0 | 0.4436 | 0 |
| d2 | 1.1937 | 0.4436 | 0.1938 |
| d3 | 0.7958 | 0.2218 | 0.2907 |
| d4 | 0 | 0 | 0.2907 |
| d5 | 0 | 0 | 0.0969 |

# TFxIDF: Example



# TFxIDF: Example



# Vector space model: TFxIDF

- Further TF variants are possible
- Sub-linear TF scaling, e.g. $\sqrt{TF}$ or $1 + log(TF)$
- Unlikely that e.g. 20 occurrences of a single term carry 20 times the weight of a single occurrence
- Also very often TF-normalization by e.g. dividing by the maximal TF in the collection

# Vector space model: vector manipulation

- Vector addition: concatenate two documents
- Vector subtraction: diff two documents
- Transform the vector space in a new vector space: projection and feature selection
- Similarity calculations
- Distance calculations
- Scaling for visualization

# Vector space model: similarity and distance

- Similarity and distance allow us to order documents
- This is useful in many applications:
  1. Relevance ranking (information retrieval)
  2. Classification
  3. Clustering
  4. Transformation and projection

# Properties of similarity and distance

## Definition

Let $D$ be the set of all data objects, e.g. all documents. Similarity and distance are functions $sim : DxD \rightarrow \mathbb{R}$, $dist : DxD \rightarrow \mathbb{R}$ with the following properties:

- Symmetry

$$\begin{aligned} sim(d_i, d_j) &= sim(d_j, d_i) \\ dist(d_i, d_j) &= dist(d_j, d_i) \end{aligned}$$

- Self-similarity (self-distance)

$$sim(d_i, d_j) \leq sim(d_i, d_i) = sim(d_j, d_j)$$
$$dist(d_i, d_j) \geq dist(d_i, d_i) = dist(d_j, d_j) = 0$$

# Properties of similarity and distance

## Additional properties

Some additional properties of the *sim* function:

- Normalization of similarity to the interval $[0, 1]$

$$\begin{aligned} sim(d_i, d_j) &\geq 0 \\ sim(d_i, d_i) &= 1 \end{aligned}$$

- Normalization of similarity to the interval $[-1, 1]$

$$\begin{aligned} sim(d_i, d_j) &\geq -1 \\ sim(d_i, d_i) &= 1 \end{aligned}$$

# Properties of similarity and distance

## Additional properties

Some additional properties of the *dist* functions:

- Triangle inequality

$$dist(d_i, d_j) \leq dist(d_i, d_k) + dist(d_k, d_j)$$

- If triangle inequality is satisfied then *dist* function defines a norm on the vector space

# Euclidean distance

## Norm

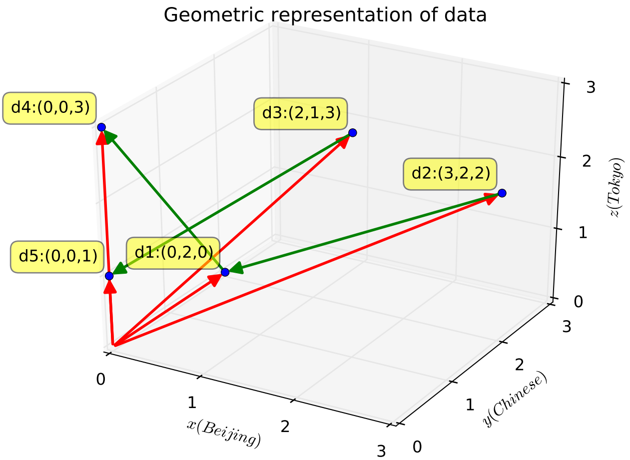
We can apply the Euclidean or $\ell_2$ norm as a distance metric in the vector space model.

$$||\mathbf{x}||_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

$$||\mathbf{x}||_2 = \sqrt{\mathbf{x}^T \mathbf{x}}$$

- Euclidean distance of two vectors $\mathbf{d}_i$ and $\mathbf{d}_j$ is the length of their displacement vector $\mathbf{x} = \mathbf{d}_i - \mathbf{d}_j$

# Euclidean distance: Example



# Euclidean distance

$$\mathbf{D} = \begin{pmatrix} \mathbf{d}_1^T \\ \mathbf{d}_2^T \\ \vdots \\ \mathbf{d}_n^T \end{pmatrix}$$

# Euclidean distance

$$\mathbf{D} = \begin{pmatrix} 0 & 2 & 0 \\ 3 & 2 & 2 \\ 2 & 1 & 3 \\ 0 & 0 & 3 \\ 0 & 0 & 1 \end{pmatrix}$$

# Euclidean distance

$$\mathbf{dist} = \begin{pmatrix} 0. & 3.60555128 & 3.74165739 & 3.60555128 & 2.23606798 \\ 3.60555128 & 0. & 1.73205081 & 3.74165739 & 3.74165739 \\ 3.74165739 & 1.73205081 & 0. & 2.23606798 & 3. \\ 3.60555128 & 3.74165739 & 2.23606798 & 0. & 2. \\ 2.23606798 & 3.74165739 & 3. & 2. & 0. \end{pmatrix}$$

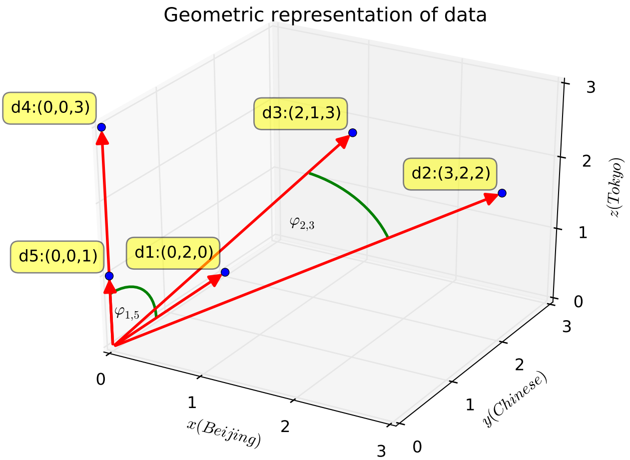
# Cosine similarity

**Angle between vectors**

We can think of the angle $\varphi$ between two vectors $\mathbf{d}_i$ and $\mathbf{d}_j$ as a measure of their similarity.

- Smaller angles mean that the vectors point in two directions that are close to each other
- Thus, smaller angles mean that the vectors are more similar
- Larger angles imply smaller similarity
- To obtain a numerical value from the interval $[0, 1]$ we can calculate $cos\varphi$
- Cosine of an angle can be also negative! Why do we always get the values from $[0, 1]$

# Cosine similarity: Example



# Cosine similarity

- We can calculate the cosine similarity we make use of the dot product and Euclidean norm:

$$sim(d_i, d_j) = \frac{\mathbf{d}_i^T \mathbf{d}_j}{||\mathbf{d}_i||_2 ||\mathbf{d}_j||_2}$$

# Cosine similarity

$$\mathbf{D} = \begin{pmatrix} 0 & 2 & 0 \\ 3 & 2 & 2 \\ 2 & 1 & 3 \\ 0 & 0 & 3 \\ 0 & 0 & 1 \end{pmatrix}$$

# Cosine similarity

$$
\mathbf{sim} = \begin{pmatrix}
1. & 0.48507125 & 0.26726124 & 0. & 0. \\
0.48507125 & 1. & 0.90748521 & 0.48507125 & 0.48507125 \\
0.26726124 & 0.90748521 & 1. & 0.80178373 & 0.80178373 \\
0. & 0.48507125 & 0.80178373 & 1. & 1. \\
0. & 0.48507125 & 0.80178373 & 1. & 1.
\end{pmatrix}
$$

# Representing data as matrices

- There are many other sources of the data that can be represented as large matrices
- We use also matrices to represent the social networks, etc.
- Measurement data, and much more
- In recommender systems, we represent user ratings of items as a utility matrix

# Recommender systems

- Recommender systems predict user responses to options
- E.g. recommend news articles based on prediction of user interests
- E.g. recommend products based on the predictions of what user might like
- Recommender systems are necessary whenever users interact with huge catalogs of items
- E.g. thousands, even millions of products, movies, music, and so on.

# Recommender systems

- These huge numbers of items arise because online you can interact also with the "long tail" items
- These are not available in retail because of e.g. limited shelf space
- Recommender systems try to support this interaction by suggesting certain items to the user
- The suggestions or recommendations are based on what the systems know about the user and about the items

# Formal model: the utility matrix

- In a recommender system there are two classes of entities: users and items

## Utility Function

Let us denote the set of all users with $U$ and the set of all items with $I$.
We define the utility function $u : UxI \rightarrow R$, where $R$ is a set of ratings and is a totally ordered set.
For example, $R = \{1, 2, 3, 4, 5\}$ set of star ratings, or $R = [0, 1]$ set of real numbers from that interval.

# The Utility Matrix

- The utility function maps pairs of users and items to numbers
- These numbers can be represented by a utility matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$, where $n$ is the number of users and $m$ the number of items
- The matrix gives a value for each user-item pair where we know about the preference of that user for that item
- E.g. the values can come from an ordered set (1 to 5) and represent a rating that a user gave for an item

# The Utility Matrix

- We assume that the matrix is sparse
- This means that most entries are unknown
- The majority of the user preferences for specific items is unknown
- An unknown rating means that we do not have explicit information
- It does not mean that the rating is low
- Formally: the goal of a recommender system is to predict the blank in the utility matrix

# The Utility Matrix: example

| User \ Movie | HP1 | HP2 | HP3 | Hobbit | SW1 | SW2 | SW3 |
|---|---|---|---|---|---|---|---|
| A | 4 | | | 5 | 1 | | |
| B | 5 | 5 | 4 | | | | |
| C | | | | 2 | 4 | 5 | |
| D | | 3 | | | | | 3 |

# Recommender systems: short summary

- Three key problems in the recommender systems:
- Collecting data for the utility matrix
  1. Explicit by e.g. collecting ratings
  2. Implicit by e.g. interactions (users buy a product implies a high rating)
- Predicting missing values in the utility matrix
  1. Problems are sparsity, cold start, and so on.
  2. Content-based, collaborative filtering, matrix factorization
- Evaluating predictions
  1. Training dataset, test dataset, error