Knowledge Discovery and Data Mining I

Winter Semester 2018/19
Agenda

1. Introduction

2. Basics
   2.1 Data Representation
   2.2 Data Reduction
   2.3 Visualization
   2.4 Privacy

3. Unsupervised Methods

4. Supervised Methods

5. Advanced Topics
Objects and Attributes

Entity-Relationship Diagram (ER)

Student

- name
- skills
- major
- semester

UML Class Diagram

Student

- name
- semester
- major
- skills

Data Tables (Relational Model)

<table>
<thead>
<tr>
<th>name</th>
<th>sem</th>
<th>major</th>
<th>skills</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ann</td>
<td>3</td>
<td>CS</td>
<td>Java, C, R</td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>CS</td>
<td>Java, PHP</td>
</tr>
<tr>
<td>Charly</td>
<td>4</td>
<td>History</td>
<td>Piano</td>
</tr>
<tr>
<td>Debra</td>
<td>2</td>
<td>Arts</td>
<td>Painting</td>
</tr>
</tbody>
</table>
### Overview of (Attribute) Data Types

<table>
<thead>
<tr>
<th>Simple Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric/metric, Categorical/nominal, ordinal</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Composed Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets, sequences, vectors</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Complex Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Multimedia: Images, videos, audio, text, documents, web pages, etc.</td>
</tr>
<tr>
<td>▶ Spatial, geometric: Shapes, molecules, geography, etc.</td>
</tr>
<tr>
<td>▶ Structures: Graphs, networks, trees, etc.</td>
</tr>
</tbody>
</table>
Simple Data Types: Numeric Data

- Numbers: natural, integer, rational, real numbers
- Examples: age, income, shoe size, height, weight
- Comparison: difference
- Example: 3 is more similar to 30 than to 3,000
Simple Data Types: Categorical Data

- "Just identities"
- Examples:
  - occupation = { butcher, hairdresser, physicist, physician, . . . }
  - subjects = { physics, biology, math, music, literature, . . . }
- Comparison: How to compare values?
  - Trivial metric:
    \[
    d(p, q) = \begin{cases} 
    0 & \text{if } p = q \\
    1 & \text{else}
    \end{cases}
    \]
- Generalization hierarchy: Use path length
Metric Space

Metric space \((O, d)\) consists of object set \(O\) and *metric distance* function \(d : O \times O \rightarrow \mathbb{R}^{\geq 0}\) which fulfills:

- **Symmetry:** \(\forall p, q \in O : d(p, q) = d(q, p)\)
- **Identity of Indiscernibles:** \(\forall p, q \in O : d(p, q) = 0 \iff p = q\)
- **Triangle Inequality:** \(\forall p, q, o \in O : d(p, q) \leq d(p, o) + d(o, q)\)

**Example:** Points in 2D space with Euclidean distance
Simple Data Types: Ordinal

**Characteristic**

There is a (total) order $\leq$ on the set of possible data values $O$:

- **Transitivity:** $\forall p, q, o \in O : p \leq q \land q \leq o \implies p \leq o$
- **Antisymmetry:** $\forall p, q \in O : p \leq q \land q \leq p \implies p = q$
- **Totality:** $\forall p, q \in O : p \leq q \lor q \leq p$

**Examples**

- Words & lexicographic ordering: $high \leq highschool \leq highscore$
- (Vague) sizes: $tiny \leq small \leq medium \leq big \leq huge$
- Frequencies: $never \leq seldom \leq rarely \leq occasionally \leq sometimes \leq often \leq frequently \leq regularly \leq usually \leq always$
### Composed Data Types: Sets

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Unordered collection of individual values</th>
</tr>
</thead>
</table>

#### Example

- skills = \{ Java, C, Python \}

#### Comparison

- **Symmetric Set Difference:**
  
  \[ R \Delta S = (R - S) \cup (S - R) = (R \cup S) - (R \cap S) \]

- **Jaccard Distance:**
  
  \[ d(R, S) = \frac{|R \Delta S|}{|R \cup S|} \]
Composed Data Types: Sets

Bitvector Representation

Given a set $S$, an ordered base set $B = (b_1, \ldots, b_n)$, create binary vector $r \in \{0, 1\}^n$ with $r_i = 1 \iff b_i \in S$.

Hamming distance: Sum of different entries (equals cardinality of symmetric set difference)

Example

Base: $B = (\text{Math, Physics, Chemistry, Biology, Music, Arts, English})$

$S = \{ \text{Math, Music, English} \} = (1,0,0,0,1,0,1)$

$R = \{ \text{Math, Physics, Arts, English} \} = (1,1,0,0,0,1,1)$

$\text{Hamming}(R, S) = 3$
Composed Data Types: Sequences, Vectors

### Characteristic

- Put $n$ values of a domain $D$ together
- Order does matter: $I_n \rightarrow D$ for an index set $I_n = \{1, \ldots, n\}$

### Examples

- **(Simple) sum**
  
  $d_1(o, q) = \sum_{i=1}^{n} |o_i - q_i|$  
  (Manhattan)

- **Root of sum of squares**
  
  $d_2(o, q) = \sqrt{\sum_{i=1}^{n} (o_i - q_i)^2}$  
  (Euclidean)

- **Maximum**
  
  $d_3(o, q) = \max_{i=1}^{n} |o_i - q_i|$  
  (Maximum)

- **General formula**
  
  $d_4(o, q) = \sqrt[p]{\sum_{i=1}^{n} |o_i - q_i|^p}$  
  (Minkowski)

- **Weighting of dimensions**
  
  $d_5(o, q) = \sqrt[p]{\sum_{i=1}^{n} w_i \cdot |o_i - q_i|^p}$  
  (Weighted Minkowski)
Complex Data Types

Components

- Structure: graphs, networks, trees
- Geometry: shapes/contours, routes/trajectories
- Multimedia: images, audio, text, etc.

Similarity models: Approaches

- Direct measures – highly data type dependent
- Feature engineering – explicit vector space embedding with hand-crafted features
- Feature learning – explicit vector space embedding learned by machine learning model, e.g. neural network
- Kernel trick – implicit vector space embedding
## Complex Data Types

### Examples for similarity models

<table>
<thead>
<tr>
<th></th>
<th>Direct</th>
<th>Feature engineering</th>
<th>Feature learning</th>
<th>Kernel-based</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graphs</strong></td>
<td>Structural Alignment</td>
<td>Degree Histograms</td>
<td>Node embeddings</td>
<td>Label Sequence Kernel</td>
</tr>
<tr>
<td><strong>Geometry</strong></td>
<td>Hausdorff Distance</td>
<td>Shape Histograms</td>
<td>Spectral Neural Network</td>
<td>Spatial Pyramid Kernel</td>
</tr>
<tr>
<td><strong>Sequences</strong></td>
<td>Edit Distance</td>
<td>Symbol Histograms</td>
<td>Recurrent neural network (RNN)</td>
<td>Cosine Distance</td>
</tr>
</tbody>
</table>
Feature Extraction

- Objects from database DB are mapped to feature vectors

- Feature vector space
  - Points represent objects
  - Distance corresponds to (dis-)similarity
Similarity Queries

- Similarity queries are basic operations in (multimedia) databases
- Given: Universe $O$, database $DB$, distance function $d$ and query object $q$

### Range query

**Range query for range parameter** $\epsilon \in \mathbb{R}_0^+$:

$$\text{range}(DB, q, d, \epsilon) = \{ o \in DB \mid d(o, q) \leq \epsilon \}$$

### Nearest neighbor query

$$\text{NN}(DB, q, d) = \{ o \in DB \mid \forall o' \in DB : d(o, q) \leq d(o', q) \}$$
**Similarity Queries**

### k-nearest neighbor query

*k*-nearest neighbor query for parameter $k \in \mathbb{N}$:

$$\text{NN}(DB, q, d, k) \subset DB \text{ with } |\text{NN}(DB, q, d, k)| = k \text{ and }$$

$$\forall o \in \text{NN}(DB, q, d, k), o' \in DB - \text{NN}(DB, q, d, k) : d(o, q) \leq d(o', q)$$

### Ranking query

Ranking query (partial sorting query): ”get next” functionality for picking database objects in an increasing order w.r.t. their distance to $q$:

$$\forall i \leq j : d(q, rank_{DB,q,d}(i)) \leq d(q, rank_{DB,q,d}(j))$$
Similarity Search

- Example: Range query
  
  \[
  \text{range}(DB, q, d, \epsilon) = \{ o \in DB \mid d(o, q) \leq \epsilon \}
  \]

- Naive search by sequential scan
  - Fetch database objects from secondary storage (e.g. disk): \(O(n)\)
  - Check distances individually: \(O(n)\)

- Fast search by applying database techniques
  - Filter-refine architecture
    - Filter: Boil database \(DB\) down to (small) candidate set \(C \subseteq DB\)
    - Refine: Apply exact distance calculation to candidates from \(C\) only
  - Indexing structures
    - Avoid sequential scans by (hierarchical or other) indexing techniques
    - Data access in (fast) \(O(n)\), \(O(\log n)\) or even \(O(1)\)
Filter-Refine Architecture

- Principle of multi-step search:
  1. Fast filter step produces candidate set $C \subset DB$ (by approximate distance function $d'$)
  2. Exact distance function $d$ is calculated on candidate set $C$ only.

- Example: Dimensionality reduction$^a$

- ICES$^b$ criteria for filter quality
  - Indexable – Index enabled
  - Complete – No false dismissals
  - Efficient – Fast individual calculation
  - Selective – Small candidate set

$^a$GEMINI: Faloutsos 1996; KNOP: Seidl & Kriegel

$^b$Assent, Wenning, Seidl: ICDE 2006
Indexing

- Organize data in a way that allows for fast access to relevant objects, e.g. by heavy pruning.

- R-Tree as an example for spatial index structure:
  - Hierarchy of minimum bounding rectangles
  - Disregard subtrees which are not relevant for the current query region
Indexing

- Example: Phone book
- Indexed using alphabetical order of participants
- Instead of sequential search:
  - Estimate region of query object (interlocutor)
  - Check for correct branch
  - Use next identifier of query object
  - Repeat until query is finished