Outline

Goal: ‘Find similar / interesting things’
• Intro to DB
• Indexing - similarity search
• Data Mining

Indexing - Detailed outline
• primary key indexing
• secondary key / multi-key indexing
• spatial access methods
• fractals
• text
• Singular Value Decomposition (SVD)
• multimedia
• ...

SVD - Detailed outline
• Motivation
• Definition - properties
• Interpretation
• Complexity
• Case studies
• SVD properties
• Conclusions

SVD - Case studies
• multi-lingual IR; LSI queries
• compression
• PCA - ‘ratio rules’
• Karhunen-Lowe transform
• query feedbacks
• google/Kleinberg algorithms

Case study - LSI
Q1: How to do queries with LSI?
Q2: multi-lingual IR (english query, on spanish text?)
Case study - LSI

Q1: How to do queries with LSI?
Problem: Eg., find documents with ‘data’

\[
\begin{bmatrix}
1 & 1 & 0 & 0 \\
2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 \\
5 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 \\
0 & 0 & 0 & 3 \\
0 & 0 & 1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
0.18 \\
0.36 \\
0.18 \\
0.90 \\
0.53 \\
0.80 \\
0.27 \\
\end{bmatrix}
= x
\begin{bmatrix}
0.64 \\
0 \\
0 \\
5.29 \\
0.58 \\
0.58 \\
0.58 \\
0.71 \\
0.71 \\
\end{bmatrix}
\]

Q1: How to do queries with LSI?
A: map query vectors into ‘concept space’ – how?

\[
\begin{bmatrix}
1 & 1 & 0 & 0 \\
2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 \\
5 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 \\
0 & 0 & 0 & 3 \\
0 & 0 & 1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
0.18 \\
0.36 \\
0.18 \\
0.90 \\
0.53 \\
0.80 \\
0.27 \\
\end{bmatrix}
= x
\begin{bmatrix}
0.64 \\
0 \\
0 \\
5.29 \\
0.58 \\
0.58 \\
0.58 \\
0.71 \\
0.71 \\
\end{bmatrix}
\]

Case study - LSI

Q1: How to do queries with LSI?
A: map query vectors into ‘concept space’ – how?

\[
q = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}
\]

A: inner product (cosine similarity) with each ‘concept’ vector \( v_i \)

Case study - LSI

Q1: How to do queries with LSI?
A: map query vectors into ‘concept space’ – how?

\[
q = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}
\]

compactly, we have:
\[
q_{\text{concept}} = q \cdot V
\]

Eg:

\[
q = \begin{bmatrix} 0.58 & 0 & 0.58 & 0 \end{bmatrix}
\]

\[
q_{\text{concept}} = \begin{bmatrix} 0.58 \end{bmatrix}
\]

Case study - LSI

Q1: How to do queries with LSI?
A: map query vectors into ‘concept space’ – how?

\[
q = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}
\]

A: inner product (cosine similarity) with each ‘concept’ vector \( v_i \)

Case study - LSI

Q1: How to do queries with LSI?
A: map query vectors into ‘concept space’ – how?

\[
q = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}
\]

term-to-concept similarities

\[
q_{\text{concept}} = \begin{bmatrix} 0.58 \end{bmatrix}
\]
**Case study - LSI**

Drill: how would the document (‘information’, ‘retrieval’) handled by LSI?

```
  term-to-concept
  similarities
```

15-826

**Case study - LSI**

Drill: how would the document (‘information’, ‘retrieval’) handled by LSI? A: SAME:

\[
d_{\text{concept}} = d \sqrt{v} \\
\text{Eg:} \\
\begin{bmatrix}
  \text{inf} & \text{data} & \text{brain} & \text{lung} \\
  0 & 1 & 1 & 0 \\
\end{bmatrix} \rightarrow \\
\begin{bmatrix}
  0.58 & 0.58 & 0 & 0.71 \\
  0.58 & 0 & 0 & 0.71 \\
\end{bmatrix} \rightarrow \\
\begin{bmatrix}
  1.16 & 0 \\
\end{bmatrix} \\
\text{CS-concept} \\
\text{term-to-concept} \\
\text{similarities}
```

15-826

**Case study - LSI**

Observation: document (‘information’, ‘retrieval’) will be retrieved by query (‘data’), although it does not contain ‘data’!!

```
  CS-concept
  d = d = [1.16 0] \\
  q = q = [0.58 0]
```

15-826

**Case study - LSI**

Q1: How to do queries with LSI?

Q2: multi-lingual IR (english query, on spanish text?)

15-826

**Case study - LSI**

- Problem:
  - given many documents, translated to both languages (eg., English and Spanish)
  - answer queries across languages

15-826

**Case study - LSI**

- Solution: ~ LSI

```
  CS
  MD
  \begin{bmatrix}
  1 & 1 & 1 & 0 & 0 \\
  2 & 2 & 0 & 0 & 0 \\
  1 & 1 & 1 & 0 & 0 \\
  0 & 0 & 0 & 3 & 3 \\
  0 & 0 & 0 & 0 & 1 \\
\end{bmatrix} \\
  \begin{bmatrix}
  1 & 1 & 1 & 0 & 0 \\
  1 & 1 & 1 & 0 & 0 \\
  1 & 1 & 1 & 0 & 0 \\
  0 & 0 & 0 & 2 & 2 \\
  0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
```

15-826
Case study - LSI

- Solution: ~ LSI

<table>
<thead>
<tr>
<th></th>
<th>data</th>
<th>retrieval</th>
<th>lung</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>1 1 0 0</td>
<td>1 1 0 0</td>
<td>1 1 0 0</td>
</tr>
<tr>
<td>MD</td>
<td>2 2 2 0</td>
<td>1 2 2 0</td>
<td>1 2 2 0</td>
</tr>
<tr>
<td></td>
<td>5 5 5 0</td>
<td>1 1 1 0</td>
<td>5 5 4 0</td>
</tr>
<tr>
<td></td>
<td>0 0 0 2 2</td>
<td>0 0 0 0 2</td>
<td>0 0 0 0 2</td>
</tr>
<tr>
<td></td>
<td>0 0 0 3 3</td>
<td>0 0 0 0 3</td>
<td>0 0 0 0 3</td>
</tr>
<tr>
<td></td>
<td>0 0 0 1 1</td>
<td>0 0 0 0 1</td>
<td>0 0 0 0 1</td>
</tr>
</tbody>
</table>

SVD - Case studies

- multi-lingual IR; LSI queries
- compression
- PCA - ‘ratio rules’
- Karhunen-Lowe transform
- query feedbacks
- google/Kleinberg algorithms

Problem - specs

- ~10^6 rows; ~10^3 columns; no updates;

Problem: 

- given a matrix
- compress it, but maintain ‘random access’
  (surprisingly, its solution leads to data mining and visualization...)

Case study: compression

[Korn+97]

Idea

SVD - reminder

- space savings: 2:1
- minimum RMS error
**Case study: compression**

- Outliers? 
  - A: treat separately
  - (SVD with ‘Deltas’)

- First singular vector

**Compression - Performance**

- 3 pass algo (→ scalability) (HOW?)
- Random cell(s) reconstruction
- 10:1 compression with < 2% error

**Performance - scaleup**

- Error
  - Space (SVD)

**Compression - Visualization**

- No Gaussian clusters; Zipf-like distribution

**SVD - Case studies**

- Multi-lingual IR; LSI queries
- Compression
- PCA - ‘ratio rules’
- Karhunen-Lowe transform
- Query feedbacks
- Google/Kleinberg algorithms

**PCA - ‘Ratio Rules’**

[Korn+00]

Typically: ‘Association Rules’ (eg.,

\{bread, milk\} → \{butter\})

But:

- Which set of rules is ‘better’?
- How to reconstruct missing/corrupted values?
- Need binary/bucketized values
PCA - ‘Ratio Rules’

Idea: try to find ‘concepts’:
- singular vectors dictate rules about ratios:
  bread:milk:butter = 2:4:3

Q2: how to reconstruct missing/corrupted values?
Eg:
- rule: bread:milk = 3:4
- a customer spent $6 on bread - how about milk?

harder cases: overspecified/underspecified

over-specified:
- milk:bread:butter = 1:2:3
- a customer got
  - $2 bread and $4 milk
  - how much milk?

Answer: minimize distance between ‘feasible’ and ‘expected’ values (using SVD...)
PCA - ‘Ratio Rules’

bottom line: we can reconstruct any count of missing values
This is very useful:
• can spot outliers (how?)
• can measure the 'goodness' of a set of rules (how?)

PCA - ‘Ratio Rules’

Identical to PCA = Principal Components Analysis
✓ – Q1: which set of rules is ‘better’?
✓ – Q2: how to reconstruct missing/corrupted values?
✓ – Q3: is there need for binary/bucketized values?
✓ – Q4: how to interpret the rules (= 'principal components')?

PCA - ‘Ratio Rules’

• Q1: which set of rules is ‘better’?
• A: the ones that needs the fewest outliers:
  – pretend we don’t know a value (eg., $ of ‘Smith’ on ‘bread’)
  – reconstruct it
  – and sum up the squared errors, for all our entries
• (other Answers are also reasonable)

PCA - ‘Ratio Rules’

Identical to PCA = Principal Components Analysis
✓ – Q1: which set of rules is ‘better’?
✓ – Q2: how to reconstruct missing/corrupted values?
✓ – Q3: is there need for binary/bucketized values?
✓ – Q4: how to interpret the rules (= ‘principal components’)?

PCA - Ratio Rules

NBA dataset
~500 players:
~30 attributes
PCA - Ratio Rules

- PCA: get singular vectors $v_1$, $v_2$, ...
- ignore entries with small abs. value
- try to interpret the rest

Ratio Rules - example

- RR1: minutes:points = 2:1
- corresponding concept?

Ratio Rules - example

- RR2: points:rebounds negatively
correlated(!) - concept?
Ratio Rules - example

- RR2: points: rebounds negatively correlated(!) - concept?
- A: position: offensive/defensive

SVD - Case studies

- multi-lingual IR; LSI queries
- compression
- PCA - ‘ratio rules’
- Karhunen-Lowe transform
- query feedbacks
- google/Kleinberg algorithms

K-L transform

A subtle point:
SVD will give vectors that go through the origin
Q: how to find $v_1$?

A subtle point:
SVD will give vectors that go through the origin
Q: how to find $v_1$?

A: ‘centered’ PCA, i.e., move the origin to center of gravity

A: ‘centered’ PCA, i.e., move the origin to center of gravity

and THEN do SVD

[1] Duda & Hart; [Fukunaga]
K-L transform

• How to ‘center’ a set of vectors (= data matrix)?
• What is the covariance matrix?
• A: see textbook
• (‘whitening transformation’)

Conclusions

• SVD: popular for dimensionality reduction / compression
• SVD is the ‘engine under the hood’ for PCA (principal component analysis)
• … as well as the Karhunen-Lowe transform
• (and there is more to come …)

References


References