Transposed problem

Data and distances between entities

Similarity and recommendations

Normalisation, Pearson Correlation

Transposed problem

What makes recommendations good?

A two-dimensional review space

Euclidean distance

Distance between 2D vectors: \( \mathbf{u} = (u_1, u_2)^T \) and \( \mathbf{v} = (v_1, v_2)^T \)

\[
rd(\mathbf{u}, \mathbf{v}) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2}
\]

Distance between D-dimensional vectors: \( \mathbf{u} = (u_1, \ldots, u_D)^T \) and \( \mathbf{v} = (v_1, \ldots, v_D)^T \)

\[
rd(\mathbf{u}, \mathbf{v}) = \sqrt{\sum_{i=1}^{D} (u_i - v_i)^2}
\]

Measures similarities between feature vectors
i.e., similarities between digits, critics, movies, genes, 

NB: \( rd(\cdot) \) denotes “2-norm”, e.g. \( p \)-norm or \( L^p \)-norm. [Note 2]

cf. other distance measures, e.g. Hamming distance, city-block distance (\( L^1 \) norm).
Similarity and Recommendation systems

1. **Distances between critics**

\[ r_c(x_c, x_u) = \sqrt{\sum_{c=1}^{C} (x_{cm} - x_{um})^2} \]

<table>
<thead>
<tr>
<th></th>
<th>Denby</th>
<th>McCarthy</th>
<th>M'stern</th>
<th>Puig</th>
<th>Travers</th>
<th>Turan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denby</td>
<td>7.7</td>
<td>10.6</td>
<td>6.2</td>
<td>5.2</td>
<td>7.9</td>
<td></td>
</tr>
<tr>
<td>McCarthy</td>
<td>7.7</td>
<td>5.0</td>
<td>4.4</td>
<td>7.2</td>
<td>3.9</td>
<td>3.9</td>
</tr>
<tr>
<td>M'stern</td>
<td>10.6</td>
<td>5.0</td>
<td>7.5</td>
<td>10.7</td>
<td>6.8</td>
<td></td>
</tr>
<tr>
<td>Puig</td>
<td>6.2</td>
<td>4.4</td>
<td>7.5</td>
<td>3.9</td>
<td>3.2</td>
<td></td>
</tr>
<tr>
<td>Travers</td>
<td>5.2</td>
<td>7.2</td>
<td>10.7</td>
<td>3.9</td>
<td>5.6</td>
<td></td>
</tr>
<tr>
<td>Turan</td>
<td>7.9</td>
<td>3.9</td>
<td>6.8</td>
<td>3.2</td>
<td>5.6</td>
<td></td>
</tr>
</tbody>
</table>

NB: Distances measured in a 6-dimensional space \((M = 6)\)

The closest pair is Puig and Turan

2. **2D distance between User1 and critics**

\[ r_s(x_u, c) = \sqrt{(2-3)^2 + (7-8)^2} = \sqrt{2} \]

<table>
<thead>
<tr>
<th></th>
<th>Hancock</th>
<th>Body of Lies</th>
<th>Milk</th>
<th>Rev Road</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denby</td>
<td>3</td>
<td>7</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>McCarthy</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>M'stern</td>
<td>5</td>
<td>7</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Puig</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Travers</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Turan</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>

3. **Simple strategy 1 for film recommendation**

- Find the closest critic, \(c^*\), to User \(u\).
- Use \(x_{c^*m}\) for \(\hat{x}_{um}\).

**Strategy 2**

Consider not only the closest critic but also all the critics.

**Option 1:** The mean or average of critic scores for film \(m\):

\[ \hat{x}_{um} = \frac{1}{C} \sum_{c=1}^{C} x_{cm} \]

**Option 2:** Weighted average over critics:

Weight critic scores according to the **similarity** between the critic and user.

\[ \hat{x}_{um} = \frac{1}{\sum_{c=1}^{C} \text{sim}(x_c, x_u)} \sum_{c=1}^{C} \text{sim}(x_c, x_u) \cdot x_{cm} \]

if every \(x_i\) has the same value, so does \(\bar{x}\).

4. **Strategy for film recommendation for User2**

- **Film recommendation for User2**

5. **Similarity measures**

There’s a choice. For example:

\[ \text{sim}(u, v) = \frac{1}{1 + r_s(u, v)} \]

Can now predict scores for User 2 (see notes)

**Good measure?**

- Consider distances \(0, \infty\), and in between.
- What if some critics rate more highly than others?
- What if some critics have a wider spread than others?
- What if not all critics have seen the same movies?

(missing data problem)

6. **Normalisation**

**Critic review score statistics**

<table>
<thead>
<tr>
<th></th>
<th>Australia</th>
<th>Body of Lies</th>
<th>Milk</th>
<th>Rev Road</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denby</td>
<td>3</td>
<td>7</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>McCarthy</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>M'stern</td>
<td>7</td>
<td>5</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Puig</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Travers</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Turan</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>

**Normalisation of critics review scores**

Sample mean and sample standard deviation of critic \(c\)’s scores:

\[ \hat{x}_c = \frac{1}{M} \sum_{m=1}^{M} x_{cm} \]

\[ s_c = \sqrt{\frac{1}{M-1} \sum_{m=1}^{M} (x_{cm} - \hat{x}_c)^2} \]

Different means and spreads make reviewers look different.

⇒ Create ‘standardised score’ with mean zero and st. dev. 1.

**Standard score:**

\[ z_{cm} = \frac{x_{cm} - \hat{x}_c}{s_c} \]

Many learning systems work better with standardised features / outputs.
**Pearson correlation coefficient**

Estimate of ‘correlation’ between critics $c$ and $d$:

$$r_{cd} = \frac{1}{M-1} \sum_{m=1}^{M} \frac{x_{cm} x_{dm}}{s_c s_d}$$

- Based on standard scores (a shift and stretch of a reviewer’s scale makes no difference — shift/stretch invariant)
- $-1 \leq r_{cd} \leq 1$
- How $r_{cd}$ can be used as a similarity measure?

Used in the mix by the winning netflix teams:


**Distances between entities**

**Similarity and recommendations**

**Normalisation, Pearson Correlation**

**Transposed problem**

And a trick: transpose your data matrix and run your code again. The result is sometimes interesting.

**Transposed problem**

**Another strategy — based on distance between Movies**

<table>
<thead>
<tr>
<th></th>
<th>Australia</th>
<th>Body of Lies</th>
<th>Burn</th>
<th>Hancock</th>
<th>Milk</th>
<th>Rev. Road</th>
</tr>
</thead>
<tbody>
<tr>
<td>Australia</td>
<td>5.8</td>
<td>5.3</td>
<td>10.9</td>
<td>8.9</td>
<td>7.2</td>
<td></td>
</tr>
<tr>
<td>Body of Lies</td>
<td>5.3</td>
<td>3.7</td>
<td>6.6</td>
<td>5.9</td>
<td>4.0</td>
<td></td>
</tr>
<tr>
<td>Burn After</td>
<td>5.3</td>
<td>3.7</td>
<td>8.9</td>
<td>7.0</td>
<td>4.5</td>
<td></td>
</tr>
<tr>
<td>Hancock</td>
<td>10.9</td>
<td>6.6</td>
<td>8.9</td>
<td>10.9</td>
<td>8.4</td>
<td></td>
</tr>
<tr>
<td>Milk</td>
<td>8.9</td>
<td>5.9</td>
<td>7.0</td>
<td>10.9</td>
<td>4.8</td>
<td></td>
</tr>
<tr>
<td>Rev. Road</td>
<td>7.2</td>
<td>4.0</td>
<td>4.5</td>
<td>8.4</td>
<td>4.8</td>
<td></td>
</tr>
</tbody>
</table>

Run the same code for distance between critics, simply transpose the data matrix first

Transpose of data in numpy is data.T, in Matlab/Octave it’s data’

**The Netflix million dollar prize**

- C = 480,189 users/critics
- $M = 17,770$ movies
- $C \times M$ matrix of ratings $\in \{1, 2, 3, 4, 5\}$ (ordinal values)

```
M = 17700;
M = (1:20000)';
C = 480189;
C = (1:5)';
5 = M * C;
5 = (1:5)';
```

- Full matrix $\sim 10$ billion cells
- $\sim 1\%$ cells filled ($100,480,507$ ratings available)

**Summary**

- Rating prediction: fill in entries of a $C \times M$ matrix
- A row is a feature vector of a critic
- Guess cells based on weighted average of similar rows
- Similarity based on distance and Pearson correlation coef.
- Could transpose matrix and run same code!
- NB: we considered a very simple case only.
- Try the exercises in Note 2, and do programming in Lab 2.

**Drop-in labs for Learning**

- Friday, 26th January, 14:10-15:00, in AT-5.05 (West Lab)
- “Similarity and recommender systems”
- Lab worksheet available from the course web page.
- Questions outside the lab hours:

http://piazza.com/ed.ac.uk/spring2018/infr08009learning

**Quizzes**

Q1: Give examples for $r_{cd} \approx -1, 0, \text{ and } 1$.

Q2: Show the Pearson correlation coefficient can be rewritten as

$$r_{cd} = \frac{\sum_{m=1}^{M} (x_{cm} - \bar{x}_c) (x_{dm} - \bar{x}_d)}{\sqrt{\sum_{m=1}^{M} (x_{cm} - \bar{x}_c)^2} \sqrt{\sum_{m=1}^{M} (x_{dm} - \bar{x}_d)^2}}$$

Q3: How the missing data of critics scores should be treated?

Q4: What if a user provides scores for a few films only?

**Matlab/Octave version**

```
c_scores = [3 7 4 9 9 7; 7 5 5 3 8 8; 5 6 8 5 9 8; 5 8 8 8 10 9; 7 7 8 4 7 8]; \% CxM
u2_scores = [6 9 6];
u2_movies = [2 3 6]; \% one-based indices

% The next line is complicated. See also next slide:
d2 = sum(absfu(m minus c_scores, u2_movies, u2_scores)."2, 2");
r2 = sqrt(d2);
sim = 1/(1 + r2); \% 1xC
pred_scores = (sim * c_scores) / sum(sim); \% 1xM = 1xC \times CxM
```
Matlab/Octave square distances

Other ways to get square distances:
- The next line is like the Python, but not valid Matlab.
- Works in recent builds of Octave.

```
d2 = sum((c_scores(:,u2_movies) - u2_scores).^2, 2)';
```

- Old-school Matlab way to make sizes match:

```
d2 = sum((c_scores(:,u2_movies) - repmat(u2_scores, size(c_scores,1), 1)).^2, 2)';
```

- Sq. distance is common; I have a general routine at:

```
% homepages.inf.ed.ac.uk/imurray2/code/imurray-matlab/square_dist.m
```

Or you could write a for loop and do it as you might in Java.
Worth doing to check your code.

---

NumPy programming example

```
from numpy import *

c_scores = array([3, 7, 4, 9, 9, 7],
[7, 5, 5, 3, 8, 8],
[7, 5, 5, 0, 8, 4],
[5, 6, 8, 5, 9, 8],
[5, 8, 8, 10, 9],
[7, 7, 8, 10, 8]]) # C,M
u2_scores = array([6, 9, 6])
u2_movies = array([1, 2, 5]) # zero-based indices

r2 = sqrt(sum((c_scores[:,u2_movies] - u2_scores)**2, 1).T) # C,
sim = 1/(1 + r2) # C,
pred_scores = dot(sim, c_scores) / sum(sim)
print(pred_scores)
```

# The predicted scores has predictions for all movies,
# including ones where we know the true rating from u2.