Euclidean distance

Distance between 2D vectors: $u = (u_1, u_2)^T$ and $v = (v_1, v_2)^T$

$$d_2(u, v) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2}$$

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

$$d_2(u, v) = \sqrt{\sum_{i=1}^{D} (u_i - v_i)^2}$$

Measures similarities between feature vectors

i.e., similarities between digits, critics, movies, genes, …

NB: $d_2(\cdot)$ denotes “2-norm”, c.f. $p$-norm or $L^p$-norm. [Note 2]

cf. other distance measures, e.g. Hamming distance, city-block distance ($L^1$ norm).

Problem definition

A two-dimensional review space

Denby 3 7 4 9 9 7
McCarthy 7 5 5 3 8 8
M'stern 7 5 5 3 8 8
Puig 5 6 8 5 9 8
Travers 5 8 8 8 10 9
Turan 7 7 8 4 7 8

Today’s schedule

- Data and distances between entities
- Similarity and recommendations
- Normalisation, Pearson Correlation
- Transposed problem

Recommender systems

Data and distances between entities

2 Similarity and recommendations

4 Transposed problem

Similarity and recommendations

Transposed problem

Today’s schedule

1 Data and distances between entities
2 Similarity and recommendations
3 Normalisation, Pearson Correlation

The Critics

Denby 3 7 4 9 9 7
McCarthy 7 5 5 3 8 8
M’stern 7 5 5 3 8 8
Puig 5 6 8 5 9 8
Travers 5 8 8 8 10 9
Turan 7 7 8 4 7 8

Film review scores by critics – data

Australia

Body of Lies

Burn After Hancock

Milk

Rev Road

Denby
3
7
4
9
9
7

McCarthy
7
5
5
3
8
8

M’stern
7
5
5
3
8
8

Puig
5
6
8
5
9
8

Travers
5
8
8
8
10
9

Turan
7
7
8
4
7
8

What makes recommendations good?

Denby
McCarthy
Morgenstern
Puig
Travers
Turan

Problem definition

A two-dimensional review space

Distance between 2D vectors: $u = (u_1, u_2)^T$ and $v = (v_1, v_2)^T$

$$d_2(u, v) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2}$$

Today’s schedule

1 Data and distances between entities
2 Similarity and recommendations
3 Normalisation, Pearson Correlation
4 Transposed problem

Inf2b Learning and Data: Lecture 2

Similarity and Recommendation systems

Hiroshi Shimodaira

(Credit: Iain Murray and Steve Renals)

Centre for Speech Technology Research (CSTR)
School of Informatics
University of Edinburgh
http://www.inf.ed.ac.uk/teaching/courses/inf2b/
https://piazza.com/ed.ac.uk/spring2019/inf2blearning

Office hours: Wednesdays at 14:00-15:00 in IF-3.04
Jan-Mar 2019
Similarity and Recommendation systems

**Distances between critics**

\[ r_2(x_1, x_2) = \sqrt{\sum_{m=1}^{6} (x_{1m} - x_{2m})^2} \]

<table>
<thead>
<tr>
<th>Critic</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>7.1</td>
<td>7.0</td>
<td>6.2</td>
<td>5.2</td>
<td>7.9</td>
</tr>
<tr>
<td>McCarthy</td>
<td>7.7</td>
<td>7.1</td>
<td>7.0</td>
<td>6.2</td>
<td>5.2</td>
<td>7.9</td>
</tr>
<tr>
<td>M'stern</td>
<td>10.6</td>
<td>5.0</td>
<td>7.5</td>
<td>7.5</td>
<td>3.9</td>
<td>3.2</td>
</tr>
<tr>
<td>Puig</td>
<td>6.2</td>
<td>4.4</td>
<td>7.5</td>
<td>7.5</td>
<td>3.9</td>
<td>3.2</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>5.6</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>5.6</td>
</tr>
</tbody>
</table>

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

The closest pair is Puig and Turan

**2D distance between User1 and critics**

\[ r_2(\text{User1}, x_m) = \sqrt{(2 - 3)^2 + (7 - 8)^2} = 2 \]

**Simple strategy 1 for film recommendation**

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

**Film recommendation for User2**

**Strategy 2**

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

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

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

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

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

\[ \hat{x}_{um} = \frac{1}{c} \sum_{i=1}^{c} \frac{\sum_{j=1}^{C} \text{sim}(x_{im}, x_{jm}) \cdot x_{jm}}{\sum_{j=1}^{C} \text{sim}(x_{im}, x_{jm})} \]

**Similarity measures**

- There's a choice. For example:
  \[ \text{sim}(u, v) = \frac{1}{1 + r_2(u, v)} \]

- 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)

**Normalisation of critics review scores**

**Critics original review scores**

<table>
<thead>
<tr>
<th>Film</th>
<th>Australia Lies</th>
<th>Body of Lies</th>
<th>After Hancock</th>
<th>Milkroad</th>
<th>mean</th>
<th>std.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denby</td>
<td>3</td>
<td>7</td>
<td>9</td>
<td>9</td>
<td>7</td>
<td>2.7</td>
</tr>
<tr>
<td>McCarthy</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>3.4</td>
</tr>
<tr>
<td>M'stern</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>3.4</td>
</tr>
<tr>
<td>Puig</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>5</td>
<td>9</td>
<td>2.2</td>
</tr>
<tr>
<td>Travers</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>9</td>
<td>3.7</td>
</tr>
<tr>
<td>Turan</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>2.4</td>
</tr>
</tbody>
</table>

**Standarised review scores**

\[ x_{c'}c = \frac{x_{cm} - \bar{x}_c}{s_c} \]

Many learning systems work better with standardised features / outputs

**Normalisation**
**Pearson correlation coefficient**

Estimate of ‘correlation’ between critics \( c \) and \( d \):

\[
rcd = \frac{1}{M-1} \sum_{m=1}^{M} \left( \frac{x_{cm} - \bar{x}_c}{s_c} \right) \left( \frac{x_{dm} - \bar{x}_d}{s_d} \right).
\]

- Based on standard scores
  - (a shift and stretch of a reviewer’s scale makes no difference – shift/scale 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:


---

**Transposed problem**

Another strategy — based on distance between

**Movies**

<table>
<thead>
<tr>
<th></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>5.3</td>
<td>5.3</td>
<td>10.9</td>
<td>8.9</td>
<td>7.2</td>
</tr>
<tr>
<td>Body of Lies</td>
<td>5.8</td>
<td>5.3</td>
<td>10.9</td>
<td>8.9</td>
<td>7.2</td>
</tr>
<tr>
<td>Burn After</td>
<td>4.3</td>
<td>3.7</td>
<td>6.6</td>
<td>5.9</td>
<td>4.0</td>
</tr>
<tr>
<td>Hancock</td>
<td>10.9</td>
<td>6.6</td>
<td>8.9</td>
<td>7.0</td>
<td>4.5</td>
</tr>
<tr>
<td>Milk</td>
<td>8.9</td>
<td>5.9</td>
<td>7.0</td>
<td>10.9</td>
<td>8.4</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>
</tr>
</tbody>
</table>

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

Transpose of data in numpy is \( \text{data.T} \), in Matlab/Octave it’s \( \text{data}^\prime \)

---

**The Netflix million dollar prize**

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

(ordinal values)

Full matrix \( \sim 10 \) billion cells

\( \sim 1\% \) cells filled (100,480,507 ratings available)

---

**Quizzes**

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

Q2: Show the Pearson correlation coefficient can be rewritten as

\[
rcd = \frac{\sum_{m=1}^{M} (x_{cm} - \bar{x}_c)(x_{dm} - \bar{x}_d)}{\sqrt{\sum_{m=1}^{M} (x_{cm} - \bar{x}_c)^2} \sqrt{\sum_{m=1}^{M} (x_{dm} - \bar{x}_d)^2}}.
\]

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

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

---

**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**

- Lab2 on 28th/29th Jan. at 11:10-13:00 in AT-6.06.
  - “Similarity and recommender systems”
- Lab worksheet available from the course web page.
- Questions outside the lab hours:

http://piazza.com/ed.ac.uk/spring2019/infr08009inf2blearning
Matlab/Octave version

c_scores = [
  3 7 4 9 9 7;
  7 5 5 0 8 4;
  5 6 8 5 9 8;
  7 8 8 8 10 9;
  7 7 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(bsxfun(@minus, 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

d2 = square_dist(u2_scores', c_scores(:,u2_movies)');

% Other ways to get square distances:
% Works in recent builds of Octave.
d2 = sum((c_scores(:,u2_movies) - u2_scores).^2, 2)';

% 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)';

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

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

d2 = square_dist(u2_scores', c_scores(:,u2_movies)');

% 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, 0, 8, 4],
   [5, 6, 8, 5, 9, 8],
   [7, 8, 8, 8, 10, 9],
   [7, 7, 4, 7, 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.