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

Jan-Mar 2015
Recommender systems

What makes recommendations good?
Today’s schedule

1. Data and distances between entities
2. Similarity and recommendations
3. Normalisation, Pearson Correlation
The Films in 2008

- Australia
- Body of Lies
- Burn After Reading
- Milk
- Hancock
- Revolutionary Road
The Critics

David Denby   Todd McCarthy   Joe Morgenstern

Claudia Puig   Peter Travers   Kenneth Turan
Film review scores by critics – data

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

Representation of data (notation):

\[ X = \begin{pmatrix} 3 & 7 & 4 & 9 & 9 & 7 \\ 7 & 5 & 5 & 3 & 8 & 8 \\ 7 & 5 & 5 & 0 & 8 & 4 \\ 5 & 6 & 8 & 5 & 9 & 8 \\ 5 & 8 & 8 & 8 & 10 & 9 \\ 7 & 7 & 8 & 4 & 7 & 8 \end{pmatrix} \]

Score of movie \( m \) by critic \( c \):

\[ x_{cm}, \quad s_{c}(m) \]

Score vector by critic \( c \):

\[ x_{c} = (x_{c1}, \ldots, x_{cM})^{T} \]
## Problem definition

Predict user’s score $\hat{x}_{um}$ for unseen film $m$ based on the film review scores by the critics. ⇒ Film recommendation
A two-dimensional review space

![Graph showing a two-dimensional review space with points for authors and a user.]
Euclidean distance

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

\[
r_2(\mathbf{u}, \mathbf{v}) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2}
\]

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

\[
r_2(\mathbf{u}, \mathbf{v}) = \sqrt{\sum_{k=1}^{d} (u_k - v_k)^2}
\]

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

NB: \( r_2(\ ) \) 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).
Distances between critics

\[ r_2(x_i, x_j) = \sqrt{\sum_{m=1}^{M} (x_{im} - x_{jm})^2} \]

<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7.9</td>
</tr>
<tr>
<td>McCarthy</td>
<td>7.7</td>
<td></td>
<td>5.0</td>
<td>4.4</td>
<td>7.2</td>
<td>3.9</td>
</tr>
<tr>
<td>M’stern</td>
<td>10.6</td>
<td>5.0</td>
<td></td>
<td>7.5</td>
<td>10.7</td>
<td>6.8</td>
</tr>
<tr>
<td>Puig</td>
<td>6.2</td>
<td>4.4</td>
<td>7.5</td>
<td></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></td>
<td></td>
<td>5.6</td>
</tr>
<tr>
<td>Turan</td>
<td>7.9</td>
<td></td>
<td>6.8</td>
<td></td>
<td>3.2</td>
<td></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

\[ r_2(\text{User1}, \text{McCarthy}) = \sqrt{(2 - 3)^2 + (7 - 8)^2} = \sqrt{2} \]
Simple strategy-1 for film recommendation

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

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

<table>
<thead>
<tr>
<th></th>
<th>User1</th>
<th>User2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
**Strategy-2 — based on distance between Movies**

<table>
<thead>
<tr>
<th></th>
<th>Australia</th>
<th>Body of Lies</th>
<th>Burn After</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>5.8</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'`
1. Distances between entities
2. Similarity and recommendations
3. Normalisation, Pearson Correlation
## Film recommendation for User2

![Graph showing the recommendation system for User2 with a user similarity matrix.](image)

The graph represents the similarity between users and movies, with User2 at the center. Each dot represents a user, and the distances between them indicate similarity. User2 is closest to McCarthy, Puig, and Turan.

### Similarity Matrix

<table>
<thead>
<tr>
<th></th>
<th>Australia</th>
<th>Body of Lies</th>
<th>Burn After</th>
<th>Hancock</th>
<th>Milk</th>
<th>Rev Road</th>
<th>$r_2($critic, User2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denby</td>
<td>3</td>
<td>7</td>
<td>4</td>
<td>9</td>
<td>9</td>
<td>7</td>
<td>$\sqrt{27} = 5.2$</td>
</tr>
<tr>
<td>McCarthy</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>8</td>
<td>$\sqrt{21} = 4.6$</td>
</tr>
<tr>
<td>M’stern</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>8</td>
<td>4</td>
<td>$\sqrt{21} = 4.6$</td>
</tr>
<tr>
<td>Puig</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>5</td>
<td>9</td>
<td>8</td>
<td>$\sqrt{5} = 2.2$</td>
</tr>
<tr>
<td>Travers</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>9</td>
<td>$\sqrt{14} = 3.7$</td>
</tr>
<tr>
<td>Turan</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>$\sqrt{6} = 2.4$</td>
</tr>
<tr>
<td>User2</td>
<td>-</td>
<td>6</td>
<td>9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
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.

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

The *normalisation* outside each sum, means that if every critic has the same score, the (weighted) average will report that score.
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 not all critics have seen the same movies? (missing data)
- What if some critics rate more highly than others?
- What if some critics have a wider spread than others?
<table>
<thead>
<tr>
<th></th>
<th>Australia</th>
<th>Body of Lies</th>
<th>Burn After</th>
<th>Hancock</th>
<th>Milk</th>
<th>Rev Road</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denby</td>
<td>3</td>
<td>7</td>
<td>4</td>
<td>9</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>McCarthy</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>M’stern</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>Puig</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>5</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>Travers</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>Turan</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

**Mean** 6.5 2.5

**Std.** 2.5
1. Distances between entities

2. Similarity and recommendations

3. Normalisation, Pearson Correlation
Normalisation

**Sample mean** and **sample standard deviation** of critic $c$’s scores:

$$\bar{x}_c = \frac{1}{M} \sum_{m=1}^{M} x_{cm}$$

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

Different means and spreads make reviewers look different.

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

$$z_{cm} = \frac{x_{cm} - \bar{x}_c}{s_c}$$

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

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

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

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

- Tends to one value as $M \to \infty$
- 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:

The Netflix million dollar prize

\[ C = 480,189 \text{ users/critics} \]
\[ M = 17,770 \text{ movies} \]
\[ C \times M \text{ matrix of ratings } \in \{1, 2, 3, 4, 5\} \]

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

Q1: Give examples for $r_{cd} \approx -1, 0, \text{ and } 1$.
Q2: How the missing data of critics scores should be treated?
Q3: What if a user provides scores for a few films only?
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, 8, 10, 9],
    [7, 7, 8, 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.
Matlab/Octave version

c_scores = [
    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 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(bsxfun(@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 * CxM
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.