The Films in 2008

The Critics

Film review scores by critics – data

What makes recommendations good?

Problem definition

A two-dimensional review space

Euclidean distance

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

Distance between \( D \)-dimensional vectors: \( u = (u_1, \ldots, u_D)^T \)

and \( v = (v_1, \ldots, v_D)^T \). 

\[
\Rightarrow \| u - v \|_2 = \sqrt{\sum_{i=1}^{D} (u_i - v_i)^2} 
\]

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

NB: \( r_2(\cdot) \) 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).
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>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></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

2D distance between User1 and critics

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

<table>
<thead>
<tr>
<th></th>
<th>Hancock</th>
<th>User1*</th>
<th>Travers</th>
<th>Turan</th>
<th>Denby*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hancock</td>
<td>0</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>User1</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>Travers</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>Turan</td>
<td>7</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>2</td>
</tr>
</tbody>
</table>

Simple strategy-1 for film recommendation

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

<table>
<thead>
<tr>
<th></th>
<th>Australia</th>
<th>Body of Lies</th>
<th>Burn</th>
<th>After</th>
<th>Hancock</th>
<th>Milk</th>
<th>Rev</th>
<th>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>
<td></td>
<td></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></td>
<td></td>
</tr>
<tr>
<td>M'stern</td>
<td>7</td>
<td>5</td>
<td>0</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td></td>
<td></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></td>
<td></td>
</tr>
<tr>
<td>Travers</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>9</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Turan</td>
<td>7</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

User1

- - - 2 - 7

User2

- 6 9 - - 6

Strategy-2 — based on distance between Movies

<table>
<thead>
<tr>
<th></th>
<th>Australia</th>
<th>Body of Lies</th>
<th>Burn</th>
<th>After</th>
<th>Hancock</th>
<th>Milk</th>
<th>Rev</th>
<th>Road</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denby</td>
<td>7.7</td>
<td>6.6</td>
<td>7.2</td>
<td>6.8</td>
<td>3.9</td>
<td>3.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>McCarthy</td>
<td>8.4</td>
<td>7.2</td>
<td>8.9</td>
<td>10.7</td>
<td>6.2</td>
<td>5.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M'stern</td>
<td>8.4</td>
<td>7.2</td>
<td>8.9</td>
<td>10.7</td>
<td>6.2</td>
<td>5.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Puig</td>
<td>6.2</td>
<td>7.2</td>
<td>8.9</td>
<td>10.7</td>
<td>6.2</td>
<td>5.6</td>
<td></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>3.2</td>
<td></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>5.6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Transposed problem

There's a choice. For example:

\[ \text{sim}(u, v) = \frac{1}{1 + r_2(u, v)} \]

Distance between entities

Similarity and recommendations

Normalisation, Pearson Correlation

Film recommendation for User2

<table>
<thead>
<tr>
<th></th>
<th>Australia</th>
<th>Body of Lies</th>
<th>Milk</th>
<th>Rev</th>
<th>Road</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denby</td>
<td>3</td>
<td>7</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>
</tr>
<tr>
<td>M'stern</td>
<td>7</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>
</tr>
<tr>
<td>Travers</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>Turan</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>Hancock</th>
<th>User2*</th>
<th>Morgenstern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hancock</td>
<td>0</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>User2</td>
<td>7</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Morgenstern</td>
<td>7</td>
<td>5</td>
<td>3</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 \(m\):

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

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

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

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

The normalisation outside each sum, means that if every critic has the same score, the (weighted) average will report that score.

Strategy-3

Consider a choice.

```
\[ \text{sim}(u, v) = \frac{1}{1 + r_2(u, v)} \]
```

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?
### Similarity and Recommendation systems

#### Distances between entities

- **Normalisation**, **Pearson Correlation**

#### Similarity and recommendations

- **Recommendations models**

#### Normalisation, Pearson Correlation

- **Estimate of ‘correlation’** between critics $c$ and $d$:
  
  $$ r_{cd} = \frac{1}{M-1} \sum_{m=1}^{M} \left( \frac{x_{cm} - \bar{x}_c}{s_c} \right) \left( \frac{x_{dm} - \bar{x}_d}{s_d} \right). $$

- **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 users/critics
- **$M$** = 17, 770 movies

- **C x M matrix of ratings** $\in \{1, 2, 3, 4, 5\}$

- **Full matrix ~ 10 billion cells**
- **~1%** cells filled (100,480,572 ratings available)

#### Rating prediction:

- Fill in entries of a C x M matrix
- A row is a feature vector of a critic
- Guess cells based on weighted average of similar rows

-used in the mix by the winning Netflix teams:


### 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

### NumPy programming example

```python
from numpy import *
c_scores = array([[ 3, 7, 4, 9, 9, 7],
                  [ 7, 5, 5, 3, 8, 8],
                  [ 7, 5, 5, 8, 6, 4],
                  [ 5, 6, 8, 5, 9, 8],
                  [ 5, 6, 8, 10, 9],
                  [ 7, 7, 8, 4, 7, 8]]) # C,M
u2_scores = array([3, 7, 4, 9, 9, 7]) # zero-based indices
u2_movies = array([2, 0]) # nonzero indices

r2 = dot(sum(c_scores[:, u2_movies] - u2_scores) ** 2, 1) / C,
      sin = 1 / (1 + r2) * C,
      pred_scores = dot(sin, c_scores) / sum(sin)
print(pred_scores)
```

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

### Matlab/Octave version

```matlab
c_scores = [3 7 4 9 9 7; 7 5 5 3 8 8; 7 5 5 8 6 4; 5 6 8 5 9 8; 5 6 8 10 9; 7 7 8 4 7 8];
u2_scores = [3 7 4 9 9 7]; u2_movies = [2 0]; % one-based indices
r2 = sqrt(sum((c_scores(:, u2_movies) - u2_scores).^2));
sin = 1 ./ (1 + r2) * C;
pred_scores = (sin * c_scores) / sum(sin); % C,M
```

- The next line is complicated. See also next slide.
  - $d_2 = \text{sum}((\text{c_scores}(:, \text{u2_movies}) - \text{u2_scores})^2, 2)$;
  - $\text{repmat}(\text{u2_scores}, \text{size}(\text{c_scores}, 1), 1)$;

### 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.
  - $d_2 = \text{sum}((\text{c_scores}(:, \text{u2_movies}) - \text{u2_scores})^2, 2)$;
  - $\text{repmat}(\text{u2_scores}, \text{size}(\text{c_scores}, 1), 1)$;

- % Old-school Matlab way to make sizes match:
  - $d_2 = \text{sum}((\text{c_scores}(:, \text{u2_movies}) - \text{u2_scores})^2, 2)$;
  - $\text{repmat}(\text{u2_scores}, \text{size}(\text{c_scores}, 1), 1)$;

- % Eq. distance is common, I have a general routine at:
  - % [homepage.inf.ed.ac.uk/imurray2/code/imurray-matlab/square_dist.m]
  - $\text{d}_2 = \text{square_dist}([\text{u2_scores}; \text{c_scores}(:,\text{u2_movies})])$;

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

### Summary

- **Rating prediction**: fill in entries of a C x 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

- **Q2**: How the missing data of critics scores should be treated?

- **Q3**: What if a user provides scores for a few films only?