Today’s schedule
- Data and distances between entities
- Similarity and recommendations
- Normalisation, Pearson Correlation
- Transposed problem

Problem definition
- Predict user’s score $\hat{x}_{um}$ for unseen film $m$ based on the film review scores by the critics. ⇒ Film recommendation
  (Fill the missing elements based on others)

Euclidean distance
- Distance between 2D vectors: $u = (u_1, u_2)^T$ and $v = (v_1, v_2)^T$
  \[ r_U(u, v) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2} \]
- Distance between $D$-dimensional vectors: $u = (u_1, \ldots, u_D)^T$
  and $v = (v_1, \ldots, v_D)^T$
  \[ r_U(u, v) = \sqrt{\sum_{i=1}^{D} (u_i - v_i)^2} \]
- Measures similarities between feature vectors
  i.e., similarities between digits, critics, movies, genes, …
  NB: $r_U(\cdot)$ denotes “2-norm”, c.f. p-norm or Lp-norm. [Note 2]

A two-dimensional review space

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

Today’s schedule
1. Data and distances between entities
2. Similarity and recommendations
3. Normalisation, Pearson Correlation
4. Transposed problem

Today’s schedule
1. Data and distances between entities
2. Similarity and recommendations
3. Normalisation, Pearson Correlation
4. Transposed problem
Distances between critics

2D distance between User1 and critics

Simple strategy 1 for film recommendation

Similarity measures

Strategies

Critic review score statistics

Normalisation of critics review scores

Similarity measures

Strategies

Critic review score statistics

Normalisation of critics review scores
Similarity and Recommendation systems

5.8 3.7 6.6 5.9 4.0
10.9 6.6 8.9 10.9 8.4
27
8.9 5.9 7.0 10.9 4.8
5.8 5.3 10.9 8.9 7.2
22
25
21
26
5.3 3.7 8.9 7.0 4.5
19

Normalisation, Pearson Correlation

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.

Another strategy — based on distance between Movies

Australia Body of Lies Burn After Hancock Milk Road Rev. Road
Australia 5.8 5.3 10.9 8.9 7.2
Body of Lies 5.3 3.7 6.6 5.9 4.0
Burn After 10.9 6.5 8.9 7.0 4.5
Hancock 8.9 5.9 7.0 10.9 4.8
Milk Road 7.2 4.0 4.5 8.4 4.8

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 × M matrix of ratings ∈ {1, 2, 3, 4, 5} (ordinal values)
Full matrix ~ 10 billion cells
~ 1% cells filled (100,480,507 ratings available)

Q1: Give examples for rcd ≈ −1, 0, and 1.
Q2: Show the Pearson correlation coefficient can be rewritten as
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 × 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 coeff.
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:

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 5 9 8; 5 8 8 8 10 9;
7 7 8 4 7 8]; % C × M
u2_scores = [6 9 6];
u2_movies = [2 3 6]; % one-based indices

% The next line is complicated. See also next slide:
d2 = sum(abs(u2_movies - c_scores(:, u2_movies)) * 2); r2 = sqrt(d2);
sin = 1./(1 + r2); % 1xC
pred_scores = (sin * c_scores) / sum(sin); % 1xC = C × M

http://piazza.com/ed.ac.uk/spring2018/infr08091learning
### 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.
- \[ d2 = \text{sum}( (c\text{\_scores}(:,u2\text{\_movies}) - u2\text{\_scores})^2, 2)'; \]
- Old-school Matlab way to make sizes match:
  \[ d2 = \text{sum}((c\text{\_scores}(:,u2\text{\_movies}) - \text{repmat}(u2\text{\_scores}, \text{size}(c\text{\_scores},1), 1)).^2, 2); \]

- 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

```python
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.