# Inf2b - Learning

Lecture 2: Similarity and Reocommendation systems

*Hiroshi Shimodaira*
(*Credit: Iain Murray and Steve Renals*)

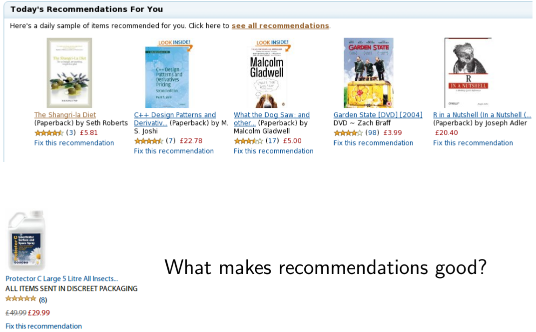
Centre for Speech Technology Research (CSTR)
School of Informatics
University of Edinburgh

http://www.inf.ed.ac.uk/teaching/courses/inf2b/
https://piazza.com/ed.ac.uk/spring2020/infr08028
Office hours: Wednesdays at 14:00-15:00 in IF-3.04

Jan-Mar 2020



## Recommender systems




What makes recommendations good?



## Today's schedule

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

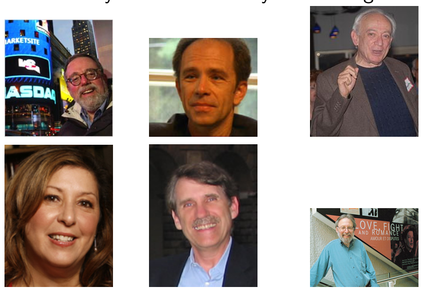


## The Films in 2008




## The Critics

David Denby Todd McCarthy Joe Morgenstern



Claudia Puig Peter Travers Kenneth Turan



## 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 | 0 | 8 | 4 |
| Puig | 5 | 6 | 8 | 5 | 9 | 8 |
| Travers | 5 | 8 | 8 | 8 | 10 | 9 |
| Turan | 7 | 7 | 8 | 4 | 7 | 8 |

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 $c$:
$x_{cm}, \quad \mathrm{sc}_c(m)$

Score vector by critic $c$:
$\mathbf{x}_c = (x_{c1}, \ldots, x_{cM})^T$

aka feature vector



## Problem definition

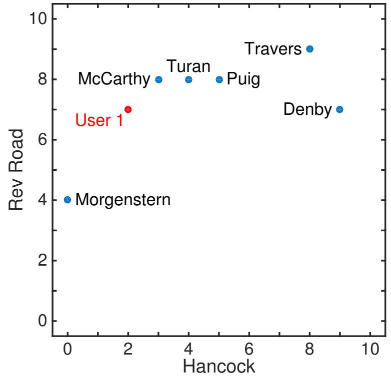
| | *Australia* | *Body of Lies* | *Burn After* | *Hancock* | *Milk* | *Rev Road* |
|---|---|---|---|---|---|---|
| Denby | 3 | 7 | 4 | 9 | 9 | 7 |
| McCarthy | 7 | 5 | 5 | 3 | 8 | 8 |
| M'stern | 7 | 5 | 5 | 0 | 8 | 4 |
| Puig | 5 | 6 | 8 | 5 | 9 | 8 |
| Travers | 5 | 8 | 8 | 8 | 10 | 9 |
| Turan | 7 | 7 | 8 | 4 | 7 | 8 |
| User1 | - | - | - | 2 | - | 7 |
| User2 | - | 6 | 9 | - | - | 6 |

Predict user's score $\hat{x}_{um}$ for unseen film $m$ based on the film review scores by the critics. $\Rightarrow$ Film recommendation

(Fill the missing elements based on others)



## A two-dimensional review space






## Euclidean distance

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

$$r_2(\boldsymbol{u}, \boldsymbol{v}) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2}$$

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

$$r_2(\boldsymbol{u}, \boldsymbol{v}) = \sqrt{\sum_{k=1}^{D} (u_k - v_k)^2}$$

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

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

$$r_2(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{m=1}^{M} (x_{im} - x_{jm})^2}$$

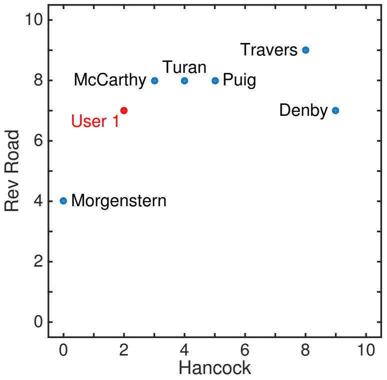
| | Denby | McCarthy | M'stern | Puig | Travers | Turan |
|---|---|---|---|---|---|---|
| Denby | | 7.7 | 10.6 | 6.2 | 5.2 | 7.9 |
| McCarthy | 7.7 | | 5.0 | 4.4 | 7.2 | 3.9 |
| M'stern | 10.6 | 5.0 | | 7.5 | 10.7 | 6.8 |
| Puig | 6.2 | 4.4 | 7.5 | | 3.9 | 3.2 |
| Travers | 5.2 | 7.2 | 10.7 | 3.9 | | 5.6 |
| Turan | 7.9 | 3.9 | 6.8 | **3.2** | 5.6 | |

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

The closest pair is Puig and Turan



## 2D distance between User1 and critics




$r_2(\text{User1, McCarthy})$

$= \sqrt{(2-3)^2 + (7-8)^2}$

$= \sqrt{2}$

$r_2(\text{User1, Turan})$

$= \sqrt{(2-4)^2 + (7-8)^2}$

$= \sqrt{5}$



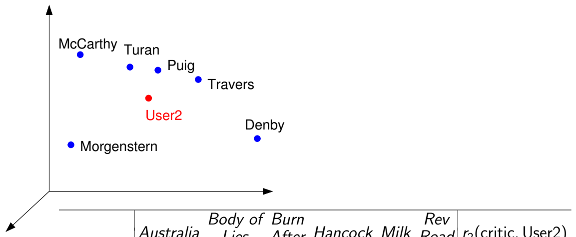
## Simple strategy 1 for film recommendation

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

| | *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 | 0 | 8 | 4 |
| Puig | 5 | 6 | 8 | 5 | 9 | 8 |
| Travers | 5 | 8 | 8 | 8 | 10 | 9 |
| Turan | 7 | 7 | 8 | 4 | 7 | 8 |
| **User1** | - | - | - | 2 | - | 7 |
| User2 | - | 6 | 9 | - | - | 6 |



## Film recommendation for User2




| | *Australia* | *Body of Lies* | *Burn After* | *Hancock* | *Milk* | *Rev Road* | $r_2$(critic, User2) |
|---|---|---|---|---|---|---|---|
| Denby | 3 | 7 | 4 | 9 | 9 | 7 | $\sqrt{27} \approx 5.2$ |
| McCarthy | 7 | 5 | 5 | 3 | 8 | 8 | $\sqrt{21} \approx 4.6$ |
| M'stern | 7 | 5 | 5 | 0 | 8 | 4 | $\sqrt{21} \approx 4.6$ |
| Puig | 5 | 6 | 8 | 5 | 9 | 8 | $\sqrt{5} \approx 2.2$ |
| Travers | 5 | 8 | 8 | 8 | 10 | 9 | $\sqrt{14} \approx 3.7$ |
| Turan | 7 | 7 | 8 | 4 | 7 | 8 | $\sqrt{6} \approx 2.4$ |
| User2 | - | 6 | 9 | - | - | 6 | |



## 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.

$$\hat{x}_{um} = \frac{1}{\sum_{c=1}^{C} \mathrm{sim}(\mathbf{x}_u, \mathbf{x}_c)} \sum_{c=1}^{C} \left(\mathrm{sim}(\mathbf{x}_u, \mathbf{x}_c) \cdot x_{cm}\right)$$

cf. Weighted arithmetic mean (weighted average) in maths:

$$\bar{x} = \frac{w_1 x_1 + w_2 x_2 + \cdots + w_n x_n}{w_1 + w_2 + \cdots w_n} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

NB: if every $x_i$ has the same value, so does $\bar{x}$.



## Similarity measures

There's a choice. For example:

$$\mathrm{sim}(\mathbf{u}, \mathbf{v}) = \frac{1}{1 + r_2(\mathbf{u}, \mathbf{v})}$$

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)



## Critic review score statistics

| | *Australia* | *Body of Lies* | *Burn After* | *Hancock* | *Milk* | *Rev Road* | mean | std. |
|---|---|---|---|---|---|---|---|---|
| Denby | 3 | 7 | 4 | 9 | 9 | 7 | **6.5** | **2.5** |
| McCarthy | 7 | 5 | 5 | 3 | 8 | 8 | **6.0** | **2.0** |
| M'stern | 7 | 5 | 5 | 0 | 8 | 4 | **4.8** | **2.8** |
| Puig | 5 | 6 | 8 | 5 | 9 | 8 | **6.8** | **1.7** |
| Travers | 5 | 8 | 8 | 8 | 10 | 9 | **8.0** | **1.7** |
| Turan | 7 | 7 | 8 | 4 | 7 | 8 | **6.8** | **1.5** |



## 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.

$\Rightarrow$ Create 'standardised score' with mean zero and st. dev. 1.

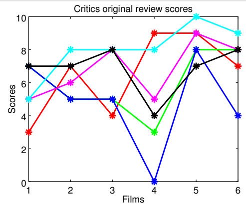
**Standard score:**

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

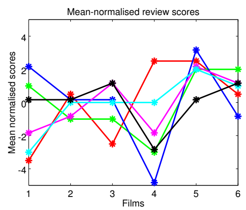
Many learning systems work better with standardised features / outputs

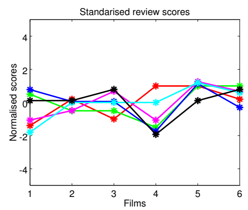


## Normalisation of critics review scores




| | *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 | 0 | 8 | 4 |
| Puig | 5 | 6 | 8 | 5 | 9 | 8 |
| Travers | 5 | 8 | 8 | 8 | 10 | 9 |
| Turan | 7 | 7 | 8 | 4 | 7 | 8 |



## Pearson correlation coefficient

Estimate of 'correlation' between critics $c$ and $d$:

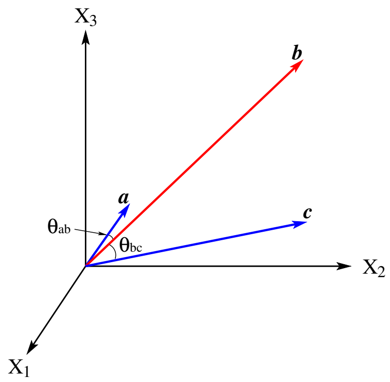
$$r_{cd} = \frac{1}{M-1}\sum_{m=1}^{M} z_{cm}\, z_{dm} = \frac{1}{M-1}\sum_{m=1}^{M}\left(\frac{x_{cm}-\bar{x}_c}{s_c}\right)\left(\frac{x_{dm}-\bar{x}_d}{s_d}\right).$$

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

Used in the mix by the winning netflix teams:
https://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pd



## Pearson correlation coefficient *(cont.)*



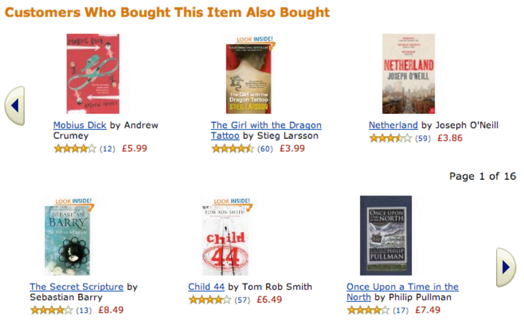



1. Distances between entities
2. Similarity and recommendations
3. Normalisation, Pearson Correlation
4. Transposed problem

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

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

$C \times M$ matrix of ratings $\in \{1,2,3,4,5\}$ (ordinal values)

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

References *(NE)*

- https://www.netflixprize.com
- https://doi.org/10.1109/MSPEC.2009.4907383
- https://doi.org/10.1109/MC.2009.263



## Further reading *(NE)*

- J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez, Recommender systems survey, Knowledge-Based Systems, Volume 46, 2013, pp.109-132. https://doi.org/10.1016/j.knosys.2013.03.012
- Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, Guangquan Zhang, Recommender system application developments: A survey, Decision Support Systems, Volume 74, 2015, pp.12-32. https://doi.org/10.1016/j.dss.2015.03.008
- Shuai Zhang, Lina Yao, Aixin Sun, Yi Tay Deep Learning based Recommender System: A Survey and New Perspectives, ACM Computing Surveys (CSUR), February 2019, Article No.: 5. https://doi.org/10.1145/3285029



## Quizzes

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



## 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

- Lab1 on 21th at 11:10-13:00, 22nd Jan. at 13:10-15:00 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 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
```



## 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, 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.
```