Inf2b Learning and Data

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

Jan-Mar 2014

Recommender systems

What makes recommendations good?

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)

Also available: dates of ratings; possibly movie information
We'll start with a smaller, simpler setup.

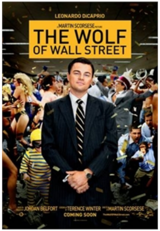
Today's schedule

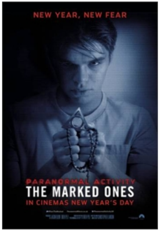
1. Distances between entities
2. Similarity and recommendations
3. Normalization, Pearson Correlation

And a trick: transpose your data matrix and run your code again.
The result is sometimes interesting.

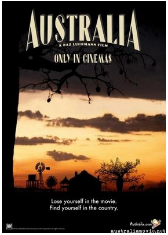
Films

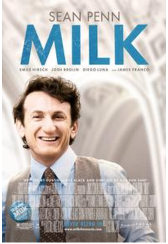
Which films do you want to see?

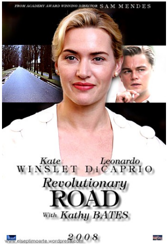




The Films in 2008







The Critics

David Denby | Todd McCarthy | Joe Morgenstern





Claudia Puig | Peter Travers | Kenneth Turan

The Data

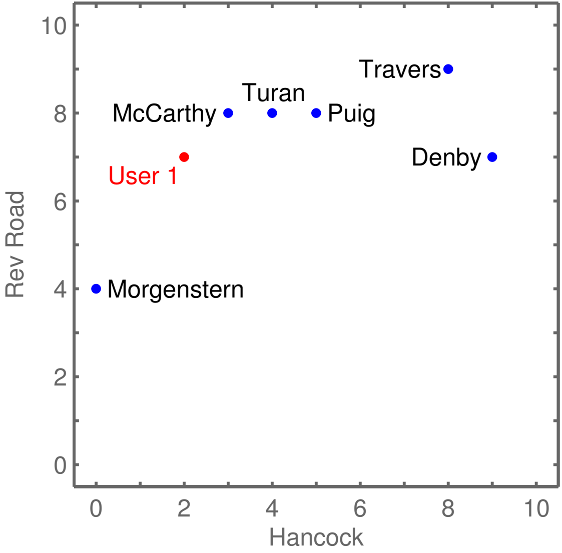
	Australia	*Body of Lies*	*Burn After*	*Hancock*	*Milk*	*Rev Road*
Denby	3	7	4	9	9	7
McCarthy	7	5	5	3	8	8
M'stern	7	5	5	0	8	4
Puig	5	6	8	5	9	8
Travers	5	8	8	8	10	9
Turan	7	7	8	4	7	8

Notations:

source code	slides
$x(c, m)$	$x_m^{(c)}, \quad \mathrm{sc}_c(m)$ $\mathbf{x}^{(c)} = (x_1^{(c)}, \ldots, x_M^{(c)})$

c: critic, m: movie

A two-dimensional review space



Euclidean distance

Distance between $2D$ vectors: $\mathbf{a} = (x, y)$ and $\mathbf{b} = (x', y')$

$$r_2(\mathbf{a}, \mathbf{b}) = \sqrt{(x - x')^2 + (y - y')^2}$$

Distance between D-dimensional vectors: $\mathbf{x}$ and $\mathbf{x}'$

$$r_2(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{d=1}^{D} (x_d - x'_d)^2}$$

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.

Distances between critics

	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

The closest pair is Puig and Turan

Transposed problem

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'`

And a trick: transpose your data matrix and run your code again.
The result is sometimes interesting.

User 2

Austra	*Body of Lies*	*Burn After Reading*	*Rev. Road*	*Australia*	*Hancock*	*Milk*
User2	6	9	6	?	?	?

Now measuring distances in 3D:

Critic	r_2(critic, user2)
Denby	$\sqrt{27} = 5.2$
McCarthy	$\sqrt{21} = 4.6$
Morgenstern	$\sqrt{21} = 4.6$
Puig	$\sqrt{5} = 2.2$
Travers	$\sqrt{14} = 3.7$
Turan	$\sqrt{6} = 2.4$

$\Rightarrow$ User 2 seems most similar to Claudia Puig

Recommendation strategies

How to predict $\mathrm{sc}_u(z)$?
— the recommendation score of film z to User u

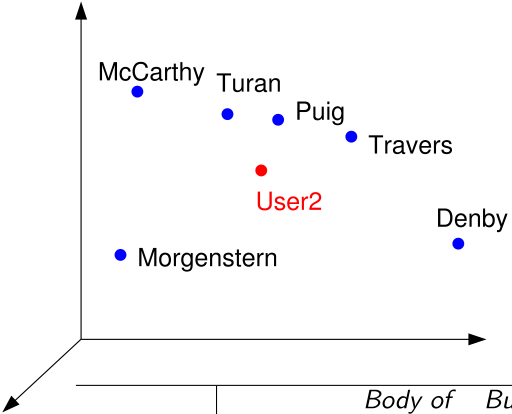
Option 1
- Find the closest critic, c^*, to User2.
- Use $\mathrm{sc}_{c^*}(z)$.

Option 2
- Consider not only the closest critic but also all the critics,
- weighting the critic's film scores according to the similarity between the critic and user.

$$\mathrm{sim}(\mathbf{x}^{(u)}, \mathbf{x}^{(c)}) \cdot \mathrm{sc}_c(z), \quad c = 1, \ldots, C$$

$\Rightarrow$ "Weighted average"

Option n?



	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

Weighted averages

Weighted average for user u, based on *similarity to critics*:

$$\mathrm{sc}_u(z) = \frac{1}{\sum_{c=1}^{C} \mathrm{sim}(\mathbf{x}^{(u)}, \mathbf{x}^{(c)})} \sum_{c=1}^{C} \mathrm{sim}(\mathbf{x}^{(u)}, \mathbf{x}^{(c)}) \cdot \mathrm{sc}_c(z)$$

The normalization outside each sum, means that if every critic has the same score, the (weighted) average will report the mean or average of critic scores for movie z:

$$\frac{1}{C} \sum_{c=1}^{C} \mathrm{sc}_c(z)$$

Simple recommender system

Predicted score: average critic score weighted by similarity

Similarity measures: There's a choice. For example:

$$\text{sim}(\mathbf{x}, \mathbf{y}) = \frac{1}{1 + r_2(\mathbf{x}, \mathbf{y})}$$

Can now predict scores for User 2 (see notes)

Good measure?

- Consider distances 0, ∞, and in between.
- What if not all critics have seen the same movies?
- What if some critics rate more highly than others?
- What if some critics have a wider spread than others?

And a trick: transpose your data matrix and run your code again.
The result is sometimes interesting.

Normalization

Mean and standard deviation of critic c's scores:

$$\mu^{(c)} = \frac{1}{M}\sum_{m=1}^{M} x_m^{(c)} \; ; \qquad \sigma^{(c)} = \sqrt{\frac{1}{M-1}\sum_{m=1}^{M}\left(x_m^{(c)} - \mu^{(c)}\right)^2}$$

Different means and spreads make reviewers look different
$\Rightarrow$ Create 'standard score' with mean zero and st. dev. 1

Standardized score:

$$z_m^{(c)} = \frac{x_m^{(c)} - \mu^{(c)}}{\sigma^{(c)}}$$

Many learning systems work better with standardized features/outputs

Pearson correlation

Estimate of 'correlation' between critics c and d:

$$\rho(c,d) = \frac{1}{M-1}\sum_{m=1}^{M} z_m^{(c)}\, z_m^{(d)}$$
$$= \frac{1}{M-1}\sum_{m=1}^{M} \frac{\left(x_m^{(c)} - \mu^{(c)}\right)}{\sigma^{(c)}} \frac{\left(x_m^{(d)} - \mu^{(d)}\right)}{\sigma^{(d)}}.$$

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

Based on standard scores

(a shift and stretch of a reviewer's scale makes no difference)

Used in the mix by the winning netflix teams:

http://www2.research.att.com/~volinsky/netflix/Bellkor2008.pdf

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
- could transpose matrix and run same 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.
```

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.