Recommender Systems

Based on: Toby Segaran: Programming Collective Intelligence

Recommendation

- We are trying to build your personal concierge service
  - I see you like X, have you tried Y?
  - You're in the mood for A? I suggest B.
- Let's say we are recommending movies. What sort of information would you take into account?
  - about the movie?
  - about the user?

Example Recommender Systems

- Netflix
- Pandora
- Amazon
- Personalized results
  - Google search
  - Facebook
  - iphone search

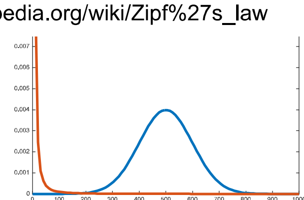
Some interesting Netflix analysis
https://www.igvita.com/2006/10/29/dissecting-the-netflix-dataset/

Challenges

- Scalability
  - millions of items
  - many millions of users
- Sparsity
  - Active users may have purchased well under 1% of the items
    - E.g. 1% of 2 million books is 20,000 books. Who buys 20k books in their lifetime? (That's 5 books a week for ~80 years)

Challenges

- Cold start
  - new users/items – what to do?
- Imbalanced data
  - "everyone" watches some movies (e.g. Oscar winners, classics)
  - Some movies are watched by (essentially) no one
  - Power law in data (red line below)
    - https://en.wikipedia.org/wiki/Power_law
    - https://en.wikipedia.org/wiki/Zipf%27s_law



Other things to consider

- Diversity in ratings
  - I may have seen many star wars movies. That doesn't mean I only want to watch star wars movies.
- Getting the high ratings correct is most important
  - 1 vs 2 stars not as important as 3 vs 4 vs 5

Collaborative Filtering: the Data

Items

Users

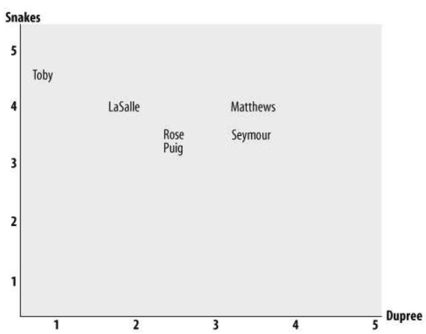
	Lady in Water	Snakes on a plane	Just my Luck	Superman Returns	You me and Dupree	The Night Listener
Lisa Rose	2.5	3.5	3.0	3.5	2.5	3.0
Gene Seymour	3.0	3.5	1.5	5.0	3.5	3
Michael Phillips	2.5	3.0		3.5		4.0
Claudia Puig		3.5	3.0	4.0	2.5	4.5
Mick LaSalle	3.0	4.0	2.0	3.0	2.0	3.0
Jack Matthews	3.0	4.0		5.0	3.5	3.0
Toby		4.5		4.0	1.0	

Collaborative Filtering: the Predictions

- We wish to predict
  – a rating $r_{u,i}$ for a particular user u and item I
  – A list of top un-rated items for a user u based on $r_{u,i}$

Finding Similar Users

- Simple way to calculate a similarity score is to use Euclidean distance, which considers the items that people have ranked in common.



People in preference space (assuming two movies)

Euclidean Distance

- Suppose we have two vectors of ratings:

$$\mathbf{x} = [x_1, \ldots, x_m]$$
$$\mathbf{y} = [y_1, \ldots, y_m]$$

- Then we can measure their similarity by the Euclidean distance:

$$sim_{\mathbf{x},\mathbf{y}} = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

Only the i's for which both **x** and **y** have known ratings participate

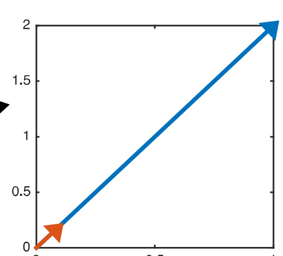
Problem with Euclidean Distance

E.g.,

- suppose a critic rated five movies by 1, 2, 3, 5, 8.
- and another critic rated those movies by .01, .02, .03, .05, .08.

- Obviously they are quite similar in the relative tastes, yet their Euclidean distance is big.

Plot of the first 2 dims

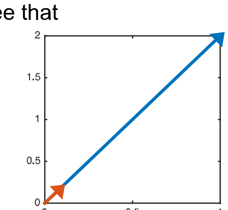


Fix

E.g.,

- suppose a user rated five movies by 1, 2, 3, 5, 8.
- and another user rated those movies by .01, .02, .03, .05, .08.

- We can consider vectors **x**=[1, 2, 3, 5, 8] and **y**=[.01, .02, .03, .05, .08] and see that the **angle** (theta) between them is 0 degrees, i.e. they point in the same direction.



- So, we can employ the **cosine** of theta:
  – the greater the cosine, the closer to 0 degrees theta is,
  – i.e. the more similar the two rating vectors are,
  – i.e. the more similar the users are.

$$sim_{x,y} = \cos\theta = \frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} = \frac{1.03}{\sqrt{103}\sqrt{0.0103}} = 1$$

The greatest value **cos** can have, which happens for theta=0.

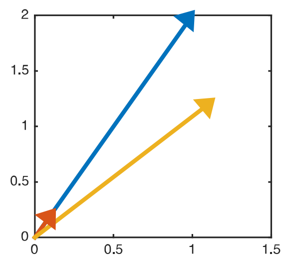
However…

E.g.,

- suppose a user rated five movies by 1, 2, 3, 5, 8.
- and another user rated those movies by .01, .02, .03, .05, .08.

- Now suppose the second user, seeing he has been too harsh, increases by 0.1 all his ratings (now the yellow line). So, the vectors are now

  x=[1, 2, 3, 5, 8] and **y**=[.11, .12, .13, .15, .18]



However…

- They are still very similar, but cosine similarity will get confused:

$$sim_{\mathbf{x},\mathbf{y}} = \cos\theta = \frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} = \frac{2.93}{\sqrt{103}\sqrt{0.0983}} = 0.92$$

- Their similarity is reduced, while it should have been (intuitively) invariant.

Better Fix: Pearson Correlation

- Recall the vectors are:

  x=[1, 2, 3, 5, 8] and **y**=[.11, .12, .13, .15, .18]

- First **centralize** by subtracting their mean, then compute cosine similarity.
- m_x = (1 + 2 + 3 + 5 + 8)/5 = 3.8
- m_y = (.11 + .12 + .13 + .15 + .18)/5 = .138

x'=[1-3.8, 2-3.8, 3-3.8, 5-3.8, 8-3.8] =
[−2.8, −1.8, −0.8, 1.2, 4.2]

y'=[.11-.138, .12-.138, .13-.138, .15-.138, .18-.138] =
[−0.028, −0.018, −0.008, 0.012, 0.042]

$$sim_{\mathbf{x},\mathbf{y}} = \cos\theta = \frac{\mathbf{x}'\cdot\mathbf{y}'}{\|\mathbf{x}'\|\|\mathbf{y}'\|} = \frac{0.308}{\sqrt{30.8}\sqrt{0.00308}} = 1$$

as we intuitively expect.

This is called **Pearson Correlation Coefficient**.

Administrivia

- I am at a conference next week
  - **Class cancelled Nov 1**
    - Please use this time to work on your **mid-term project reports due Nov 8**
  - Guest lectures Nov 2 & 4
    - Dr George Tzanatakis: Data mining for music
    - David Johnson: Piano tutor based on computer vision
  - There will be questions on your final from these lectures.
  - **No office hours** on Friday Nov 4
    - feel free to email/post in forums if you have questions

Mid Term Project Report

- Due Nov 8 on Connex
- Should be self-contained (i.e. don't assume I remember anything from your proposal)
  - What problem are you working on?
  - Why is it interesting/important?
  - What is the data (input) and what are you predicting (output)

Mid Term Project Report

- Include an update on your work so far
  - If you are creating a dataset, give an example of what the data looks like, how much data you have, some stats about the data (i.e. 60% spam 40% not spam, over 10,000 unique words)
  - If you are not creating a dataset, you should have some results by now
  - No code, please
- If you haven't gotten to the point where you have any results, consider making **an infographic** to tell me about your data

Pearson Correlation Formula

$$\mathbf{x} = [x_1, \ldots, x_m]$$
$$\mathbf{y} = [y_1, \ldots, y_m]$$

- Formula:

$$sim_{\mathbf{x},\mathbf{y}} = \frac{\sum_{i=1}^{m} (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{m} (x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{m} (y_i - \bar{y})^2}}$$

Only the *i*'s for which both **x** and **y** have known ratings participate

Pearson Correlation Numbers

- The correlation coefficient is always between -1 and +1.
- The closer the correlation is to +/-1, the closer to a perfect linear relationship. **E.g.**

-1.0 to -0.7 strong negative association.

-0.7 to -0.3 weak negative association.

-0.3 to +0.3 little or no association.

+0.3 to +0.7 weak positive association.

+0.7 to +1.0 strong positive association.

Now we can measure similarity

- We know if user X and Y are similar

$$sim_{\mathbf{x},\mathbf{y}} = \frac{\sum_{i=1}^{m} (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{m} (x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{m} (y_i - \bar{y})^2}}$$

- How can we use this to make *item* suggestions for a user?

Recall the Data

	Lady in Water	Snakes on a plane	Just my Luck	Superman Returns	You me and Dupree	The Night Listener
Lisa Rose	2.5	3.5	3.0	3.5	2.5	3.0
Gene Seymour	3.0	3.5	1.5	5.0	3.5	3
Michael Phillips	2.5	3.0		3.5		4.0
Claudia Puig		3.5	3.0	4.0	2.5	4.5
Mick LaSalle	3.0	4.0	2.0	3.0	2.0	3.0
Jack Matthews	3.0	4.0		5.0	3.5	3.0
Toby		4.5		4.0	1.0	

User-User similarity

```
rat_mat = np.array([[2.5, 3.5,3.0,3.5,2.5,3.0],
        [3.0,3.5,1.5,5,3.5,3],
        [2.5,3.0,0,3.5,0,4],
        [0,3.5,3,4,2.5,4.5],
        [3,4,2,3,2,3],
        [3,4,0,5,3.5,3],
        [0,4.5,0,4,1,0]])

# … some work here to calculate user similarity in
# variable named d_user_user …

print d_user_user
[[ 1.     0.396  0.405  0.567  0.594  0.747  0.991]
 [ 0.396  1.     0.205  0.315  0.412  0.964  0.381]
 [ 0.405  0.205  1.     1.    -0.258  0.135 -1.   ]
 [ 0.567  0.315  1.     1.     0.567  0.029  0.893]
 [ 0.594  0.412 -0.258  0.567  1.     0.211  0.924]
 [ 0.747  0.964  0.135  0.029  0.211  1.     0.663]
 [ 0.991  0.381 -1.     0.893  0.924  0.663  1.   ]]
```


How to use similarity?

- To recommend item *i* to user *u*, predict how *u* would have rated *i* by computing a **weighted average** of the ratings that other users have given to *i*.
- The weights are the similarities.
  - The more similar a user *v* is to *u*, the bigger the weight for *v*'s rating of *i*.

$$\hat{r}_{u,i} = \frac{\sum_{v \in U_i} r_{v,i} \cdot sim_{v,u}}{\sum_{v \in U_i} sim_{v,u}}$$

U_i is the set of users who have rated *i*.

This is using "user-user" similarities

Recommendation Example

```
>>> rat_mat[-1]
array([ 0. ,  4.5,  0. ,  4. ,  1. ,  0. ])
```
(Toby's ratings)

```
>>> d_user_user[-1]
array([ 0.991,  0.381, -1.   ,  0.893,  0.924,  0.663,  1.   ])
```
(Toby's similarity to other users)

```
>>> rat_mat[:,0]
array([ 2.5,  3. ,  2.5,  0. ,  3. ,  3. ,  0. ])
```
(All ratings for movie 0)

```
>>> rat_mat[:,2]
array([ 3. ,  1.5,  0. ,  3. ,  2. ,  0. ,  0. ])

>>> rat_mat[:,5]
array([ 3. ,  3. ,  4. ,  4.5,  3. ,  3. ,  0. ])

e.g for movie 0
(2.5*0.991 +  3.0*0.381 + 2.5*-1 + 3*0.924  + 3*0.663)/
(0.991 + 0.381 -1 + 0.924  + 0.663)
```

$$\hat{r}_{u,i} = \frac{\sum_{v \in U_i} r_{v,i} \cdot sim_{v,u}}{\sum_{v \in U_i} sim_{v,u}}$$

Recommending for Toby.

'Toby': {'Snakes on a Plane': 4.5, 'Superman Returns': 4.0, 'You, Me and Dupree': 1.0}

predicted ratings:
item 0 3.00
item 2 2.53
item 5 3.11

Matching Products

- Recall Amazon…

Customers who bought this item also bought

Learning Python, Second Edition by Mark Lutz
Python Cookbook by Alex Martelli
Python in a Nutshell by Alex Martelli
Python Essential Reference (2nd Edition) by David Beazley
Foundations of Python Network Programming (Foundations) by John Goerzen

Explore similar items : Books (42)

Here we have extreme sparsity and imbalance of ratings

$$\hat{r}_{u,i} = \frac{\sum_{v \in U_i} r_{v,i} \cdot sim_{v,u}}{\sum_{v \in U_i} sim_{v,u}}$$

user-user similarity may be suboptimal

Transform the data

```
>>> rat_mat
array([[ 2.5,  3.5,  3. ,  3.5,  2.5,  3. ],
       [ 3. ,  3.5,  1.5,  5. ,  3.5,  3. ],
       [ 2.5,  3. ,  0. ,  3.5,  0. ,  4. ],
       [ 0. ,  3.5,  3. ,  4. ,  2.5,  4.5],
       [ 3. ,  4. ,  2. ,  3. ,  2. ,  3. ],
       [ 3. ,  4. ,  0. ,  5. ,  3.5,  3. ],
       [ 0. ,  4.5,  0. ,  4. ,  1. ,  0. ]])
>>> rat_mat.T
array([[ 2.5,  3. ,  2.5,  0. ,  3. ,  3. ,  0. ],
       [ 3.5,  3.5,  3. ,  3.5,  4. ,  4. ,  4.5],
       [ 3. ,  1.5,  0. ,  3. ,  2. ,  0. ,  0. ],
       [ 3.5,  5. ,  3.5,  4. ,  3. ,  5. ,  4. ],
       [ 2.5,  3.5,  0. ,  2.5,  2. ,  3.5,  1. ],
       [ 3. ,  3. ,  4. ,  4.5,  3. ,  3. ,  0. ]])
```

Then, compute similarities between items, rather than users.

Recommendations using item-item similarities

$$\hat{r}_{u,i} = \frac{\sum_{j \in I_u} r_{u,j} \cdot sim_{i,j}}{\sum_{j \in I_u} sim_{i,j}}$$

I_u is the set of items rated by user u.

This is using "item-item" similarities

$$\hat{r}_{u,i} = \frac{\sum_{v \in U_i} r_{v,i} \cdot sim_{v,u}}{\sum_{v \in U_i} sim_{v,u}}$$

U_i is the set of users who have rated i.

compare to user-user


```
>>> print d_item_item
[[ 1.     0.764 -0.945  0.488  0.333 -0.612]
 [ 0.764  1.    -0.333  0.112 -0.645 -0.566]
 [-0.945 -0.333  1.    -0.423 -0.486  0.556]
 [ 0.488  0.112 -0.423  1.     0.658 -0.18 ]
 [ 0.333 -0.645 -0.486  0.658  1.    -0.25 ]
 [-0.612 -0.566  0.556 -0.18  -0.25   1.   ]]

>>> rat_mat[-1]
array([ 0. ,  4.5,  0. ,  4. ,  1. ,  0. ])

e.g. for movie 0:
(4.5*0.764+4*0.488+1*0.333)/(0.764 + 0.488 + 0.333)
```

$$\hat{r}_{u,i} = \frac{\sum_{j \in I_u} r_{u,j} \cdot sim_{i,j}}{\sum_{j \in I_u} sim_{i,j}}$$

predicted ratings:
item 0 3.61
item 2 2.96
item 5 3.53

Whom to invite to a premiere?

- For another example, reversing the products with the people, as done here, would allow an online retailer to search for people who might buy certain products.

Pros and Cons

- **User-user**:
  – Diversified recommendations, even for cold start users.
    - However, recommendations might not be good for cold start users.
  – Eccentric (black-sheep) users will not get good recommendations.

- **Item-item**:
  – Similarities more reliable between items (unless item is niche)
  – Eccentric (black-sheep) users will get better recommendations.
  – There might be just too few items a cold start user has rated so far, and will only get non-diversified recommendations.

Evaluating Rec. Systems

- For each existing rating, hide it and try to predict it.
- Compute the average squared error.

$$RMSE = \sqrt{\frac{\sum_{u=1}^{U} \sum_{all\ r_{u,i}>0} \left(r_{u,i} - \hat{r}_{u,i}\right)^2}{\sum_{u=1}^{U} \sum_{all\ r_{u,i}>0} 1}}$$

Based on

- Toby Segaran. Programming Collective Intelligence. O'Reilly 2007 .