Recommendation Systems

Deepak Venugopal

4745

Some Slides Courtesy Vibhav Gogate, Padhraic Smyth

Recommendation Systems

- Some references
- Chapter by Melville and Sindhwani , encyclopedia of ML, 2010
- Chapter on recommendation systems, Mining massive datasets, Rajaraman, Leskovec and Ullman

Recommendation Systems

- Present a user with items he/she may like
- Useful in several practical cases
  - Movies
  - Products
  - Services
  - Etc.
- https://en.wikipedia.org/wiki/Collaborative_filtering#/media/File:Collaborative_filtering.gif

Ratings data

- Data with users N and M items
- Represent as a NXM matrix
- Entries of the matrix
  - Ratings
  - Binary value (did user like/dislike, watch/did-not-watch, listen/did-not-listen,…)

Examples

- Amazon
- Ebay
- Netflix
- Youtube
- Pandora
- Goodreads
- …

Netflix Challenge

- Netflix held a 1Million$ challenge to improve their recommendation system
- Privacy issues??
- Currently the Netflix dataset has been taken down

Users

Item-1	Item-2	Item-3	Item-4	Item-5
2	2	1	3	
1		1	1	
2		1	3	3
2		1	2	1
….				

Training data

Item-1	Item-2	Item-3	Item-4	Item-5
1	?	2	1	?

Testing data

Evaluation Metrics

- Mean squared error across all ratings for all users
  - r_{ui}: Rating of user u for item i
  - p_{ui}: Predicted rating of user u for item I
  - R set of ratings in the test set
- $MSE = \frac{1}{|R| \sum_{u,i \in R} (r_{ui} - p_{ui})^2}$
- Assumes that are ratings are equally important

Evaluation Metrics

- Precision based metrics
  - Rank the predictions
  - Measure precision in the top K places of the rankings
  - More emphasis on getting important predictions right
- Real ideal evaluation: Would the user be persuaded to buy an item X that they would otherwise not buy?
  - Harder to do

Content based recommendations

- Recommend an item to a user if the user previously liked similar item
- Ignore the preferences of other users
- How to measure similarity?
  - Movies with certain actors, genre, director, etc.
  - Music genre, year, etc.

Content based recommendation

- Represent each item as a feature vector
  - $x_1, \ldots, x_M$
- Map a user into the same feature space
  - x_u
- For each item i, compute $Sim(x_u, x_i)$

Example:Text Content

- Items=documents
- Features=words or phrases
  - Represent a document as “bag of words”
  - Count the frequency of words in the document
- Use TF-IDF for vectorizing a document from the frequency counts

Example:Text

- TF_{ij}: Frequency of term (feature) j in document (item) i
- n_i: Number of documents that mention i
- N: total number of documents
- $IDF_i = \log\left(\frac{N}{n_i}\right)$
- TF-IDF $w_{ij} = TF_{ij} IDF_i$

Example:Text

- Feature vector for items
  - x_i is the vector of words represented by the TF-IDF scores
- Feature vector for user
  - Average TF-IDF for items rated by the user weighted by the user rating
- Prediction
  - Cosine similarity $\cos(x_u, x\_i) = x_u . x_i / (||x_u||.||x_i||)$

Tf-idf

Document 1

TERM	TERM-COUNT
This	1
Is	1
a	2
Sample	1

Document 2

TERM	TERM-COUNT
This	1
Is	1
a	2
example	3

Tf-idf

Document 1

TERM	TF
This	1/5
Is	1/5
a	2/5
Sample	1/5

Document 2

TERM	TERM-COUNT
This	1/7
Is	1/7
a	2/7
example	3/7

Tf-idf

Document 1

TERM	idf
This	Log(2/2)=0
Is	0
a	0
Sample	Log(2/1)

Document 2

TERM	idf
This	0
Is	0
a	0
example	Log(2/1)

Content based recommendation

- Pros
  - Easy to implement
  - Only need data for a user
  - New items with no ratings can be included as long as we can compute its feature vector
- Cons
  - Items must have similar features which is not always the case
  - Finding features might be hard
  - How do we perform predictions for new users?

User-User Collaborative Filtering

- Make recommendations for a given user based on similar users
- Let K_a be the nearest-neighbors of a
- Generate predictions for user a based on a weighted combination of K_a's ratings

User-User Collaborative Filtering

- Define a similarity weight between user a and user u as correlation coefficient (Pearson's coefficient)

$$w_{a,u} = \frac{\sum_{i \in I} (r_{a,i} - \overline{r}_a)(r_{u,i} - \overline{r}_u)}{\sqrt{\sum_{i \in I} (r_{a,i} - \overline{r}_a)^2 \sum_{i \in I} (r_{u,i} - \overline{r}_u)^2}}$$

- I is the set of items rated by both a and u. r_{ui} is the rating given by user u to i. $\overline{r_u}$ is the mean rating of u across items in I.

User-User Collaborative Filtering

- Prediction of a's rating to item i
  - $p_{ai} = \overline{r_a} + \sum_{u \in \mathrm{K}} (r_{ui} - \overline{r_u}) w_{au} / \sum_{u \in \mathrm{K}} w_{au}$
  - $\overline{r_u}$ is the average rating for u
  - w_{au} is the similarity weight between a and u
  - K is the set of nearest neighbors for a based on w_{au}
- We are computing deltas for each user in a's neighborhood weighting each delta by similarity of the user with a

User-User Collaborative Filtering

P1	P2	P3	P4	P5	P6
1		3	?	1	
1		3	2	4	
1	2	3	3	5	4

0.18

0

$$p_{13} = 1.6 + (2 - 2.5) * \frac{0.18}{0.18} = 1.1$$

User-User Algorithm

- Steps
  - For a user a, find the K nearest-neighbors using w_{au}
  - Predict a's ratings for each item using K
- Problems
  - Naïve complexity = O(NM), for N items and M users
  - Amazon has 100 million customers and 10 million items
  - A user may have neighbors with very few co-rated items

Heuristics

- Remove customers with too few purchases
- Randomly sample customers
- Remove rarely purchased events
- Cluster customers/items etc.

Item based Collaborative Filtering

- Match a user's rated item to similar items
  - More scalable (Fewer items)

$$w_{i,j} = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}}$$

- U: users who have rated both items i and j
- $\overline{r_i}$: Average rating for item i across all users in U
- Can compute the above weights offline

Item based Collaborative Filtering

- Predict the rating for item i by user a
  - $p_{ai} = \sum_{j \in \mathrm{K}} r_{aj} w_{ij} / \sum_{j \in \mathrm{K}} |w_{ij}|$
  - K is the neighborhood set most similar to item i among all items rated by user a

Toy Example of Item-Item Collaborative Filtering

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3			5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

- unknown rating   - rating between 1 to 5

Figures and example courtesy of Jure Leskovec, Stanford

Toy Example of Item-Item Collaborative Filtering

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

- estimate rating of movie **1** by user **5**

Figures and example courtesy of Jure Leskovec, Stanford

Toy Example of Item-Item Collaborative Filtering

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12	$w_{1,j}$
1	1		3		?	5			5		4		**1.00**
2	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3	**-0.18**
3	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		**<u>0.41</u>**
4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		**-0.10**
5			4	3	4	2					2	5	**-0.31**
<u>6</u>	1		3		3			2			4		**<u>0.59</u>**

Using Pearson correlation as similarity:

1) Subtract mean rating r_i from each movie i

$r_1 = (1+3+5+5+4)/5 = \mathbf{3.6}$

row 1: *[-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]*

2) Compute correlation between rows

Neighbor selection:
Identify movies similar to movie **1, rated by user 5**

Figures and example courtesy of Jure Leskovec, Stanford

Toy Example of Item-Item Collaborative Filtering

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12	$w_{1,j}$
1	1		3		?	5			5		4		**1.00**
2			5	4			4			2	1	3	**-0.18**
3	2	4		1	2		3		4	3	5		**0.41**
4		2	4		5			4			2		**-0.10**
5			4	3	4	2					2	5	**-0.31**
6	1		3		3			2			4		**0.59**

Find the 2 nearest neighbors, with similarity weights $w_{13}=0.41$, $w_{16}=0.59$

Figures and example courtesy of Jure Leskovec, Stanford

Toy Example of Item-Item Collaborative Filtering

movies \ users	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		**2.6**	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Predict by taking weighted average:

$P_{1,5}$ = (0.41*2 + 0.59*3) / (0.41+0.59) = 2.6

$$p_{a,i} = \frac{\sum_{j \in K} r_{a,j} w_{i,j}}{\sum_{j \in K} |w_{i,j}|}$$

Figures and example courtesy of Jure Leskovec, Stanford

User-User Vs Item

- Item-Item appears to be more stable
- Item dimensionality is smaller. More data per item than per user
- Algorithmically Item-Item can be implemented faster than User-User (Linden,Smith and York, Amazon Recommender System)

Pros/cons of collaborative filtering

- Pros
  - Works for any kind of item
  - No feature engineering since we don’t need features (in content based we needed features)
- Cons
  - Cannot recommend items that have not been rated
  - Need to have users in the system (“cold start”: what to do when there are no users as yet)
  - Popularity bias

Hybrid methods

- Implement both content based recommendation and collaborative filtering
- Combine the recommender results