Recommender Systems

Carlos Carvalho, Mladen Kolar and Robert McCulloch

11/05/2015

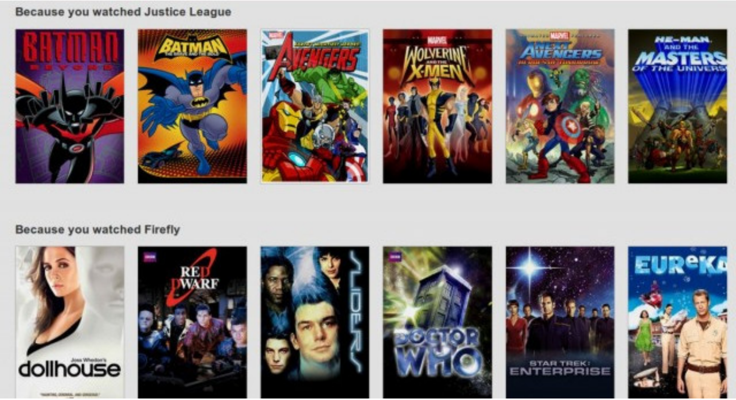
Tools so far

We already know many things that we can use to recommend new content.

- predict rating of a restaurant
- predict if a user is going to respond to a marketing campaign
- predict if user is going to like a music

Why do we need recommendations?

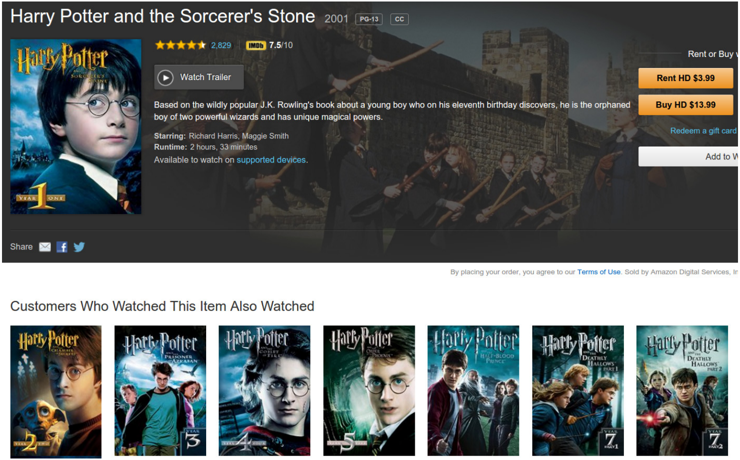
Help users discover new content



Why do we need recommendations?

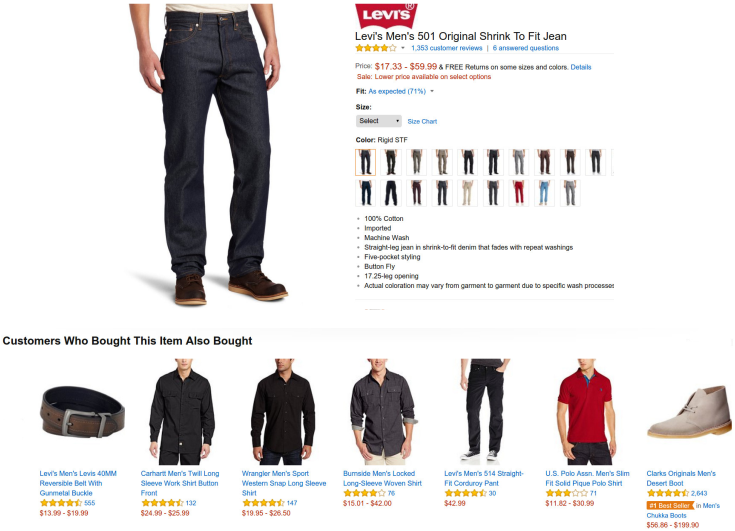
Help users find what they have been already looking for.

Prominently place content as a part of the user inerface.



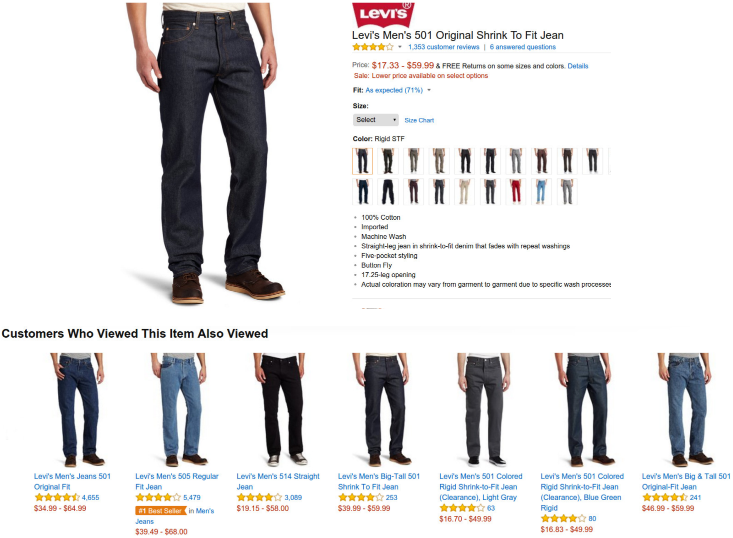
Why do we need recommendations?

Help users find complementary products



Why do we need recommendations?

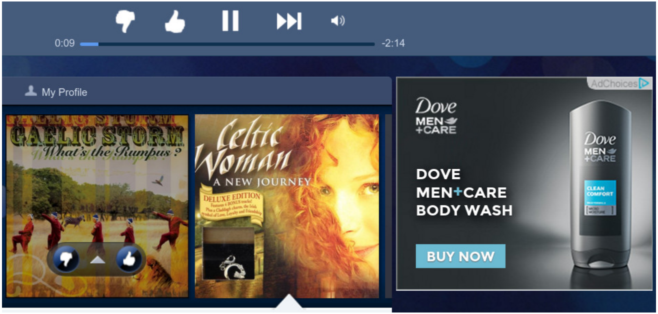
Help users find substitute products



Why do we need recommendations?

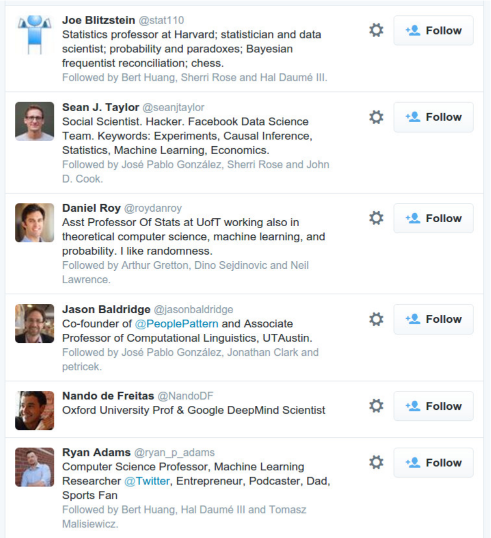
Personalize user experience based on the feedback

Recommend products based on our interests



Why do we need recommendations?

Who to follow?



Why do we need recommendations?

Help us find things we like

Why do we need recommendations?

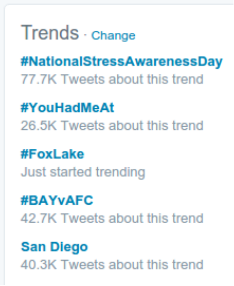
- Discover new content
- Help us find what we are looking for
- Personalize content based on our feedback/interests
- Help us find things we like
- ...

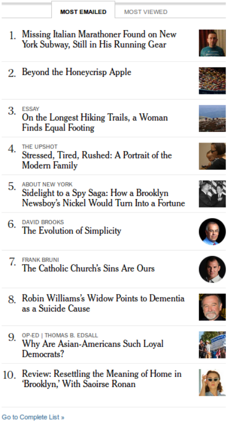
All of the above problems are slightly different from each other.

However, at the base of all of them is the need to model preferences, opinions and behaviour of users.

Popularity

What are people viewing now



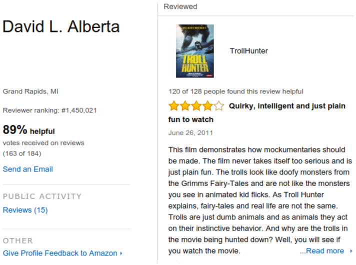


Limitations

- no context (what is my intention now)
- no personalization

Predicting rating using existing models

Build a model using features of a user and an item



Product Details

Genres	Fantasy, Drama, International, Comedy, Horror
Director	André Øvredal
Starring	Otto Jespersen, Glenn Erland Tosterud
Supporting actors	Johanna Mørck, Tomas Alf Larsen, Urmila Berg-Domaas, Hans Morten Hansen, Robert Stoltenberg, Knut Nærum, Eirik Bech, Inge Erik Henjesand, Tom Jørgensen, Benedicte Aubert Ringnes, Magne Skjævesland, Torunn Lødemel Stokkeland, Finn Norvald Øvredal, Kaja Halden Aarrestad, Robin De Lano, Jens Stoltenberg
Studio	Magnolia
MPAA rating	PG-13 (Parental Guidance Suggested)
Captions and subtitles	English Details
Rental rights	48 hour viewing period. Details
Purchase rights	Stream instantly and download to 2 locations Details
Format	Amazon Video (streaming online video and digital download)

Predicting rating using existing models

The Music Genome Project
`https://www.pandora.com/about/mgp`

... Each song in the Music Genome Project is analyzed using up to 450 distinct musical characteristics by a trained music analyst. These attributes capture not only the musical identity of a song, but also the many significant qualities that are relevant to understanding the musical preferences of listeners. The typical music analyst working on the Music Genome Project has a four-year degree in music theory, composition or performance, has passed through a selective screening process and has completed intensive training in the Music Genome's rigorous and precise methodology. ...

Predicting rating using existing models

Obtaining features may be expensive

Predicting rating using existing models

Build a model using features of a user and an item

- linear model
- decision tree
- boosting model
- ...

$$\text{rating} = f(\text{user features}, \text{item features})$$

Predicting rating using existing models

We could predict how many stars a user would give.

We could predict whether a user would give a high rating if they purchase an item.

We could predict whether a user will like an item.

Based on the predicted rating, the system would recommend

- diverse set of items
- new/unseen content
- something that the user likes or searches for

Predicting rating using existing models

We train a model based on user's past feedback and various features that characterize users and items.

Recommender system provides a rating and we provide recommendations based on this.

Similar to the example in marketing where customers are targeted based on $\hat{p}$.

Predicting rating using existing models

Approach does not suffer from a cold-start problem

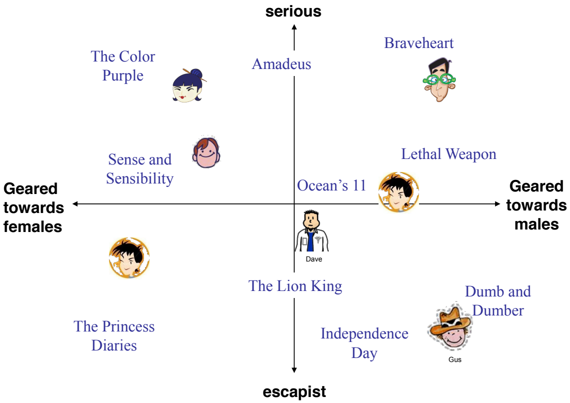
- Rate new movie from features of other movies user liked

Limitations

- sometimes, we do not have access to features
- often does not perform as well as collaborative filtering methods

Learning relationships

Recommender systems can uncover/model relationships between users and items they are evaluating, using historical ratings.



Collaborative filtering

Data we may have (example from Netflix challenge)

Training data

user	movie	date	score
1	21	5/7/02	1
1	213	8/2/04	5
2	345	3/6/01	4
2	123	5/1/05	4
2	768	7/15/02	3
3	76	1/22/01	5
4	45	8/3/00	4
5	568	9/10/05	1
5	342	3/5/03	2
5	234	12/28/00	2
6	76	8/11/02	5
6	56	6/15/03	4

Test data

user	movie	date	score
1	62	1/6/05	?
1	96	9/13/04	?
2	7	8/18/05	?
2	3	11/22/05	?
3	47	6/13/02	?
3	15	8/12/01	?
4	41	9/1/00	?
4	28	8/27/05	?
5	93	4/4/05	?
5	74	7/16/03	?
6	69	2/14/04	?
6	83	10/3/03	?

Collaborative filtering

Data commonly represented as a rating matrix.

users

items	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

Collaborative filtering

- a user viewed/bought an item (1)
- a user did not view/buy an item (0)

users

movies

1		3			5			5		4	
		5	4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

users

movies

1	0	1	0	0	1	0	0	1	0	1	0
0	0	1	1	0	0	1	0	0	1	1	1
1	1	0	1	1	0	1	0	1	1	1	0
0	1	1	0	1	0	0	1	0	0	1	0
0	0	1	1	1	1	0	0	0	0	1	1
1	0	1	0	1	0	0	1	0	0	1	0

Collaborative filtering

- 1 user viewed/bought an item and liked it
- -1 user viewed/bought an item and did not like it
- 0 user did not view/buy an item

users

movies

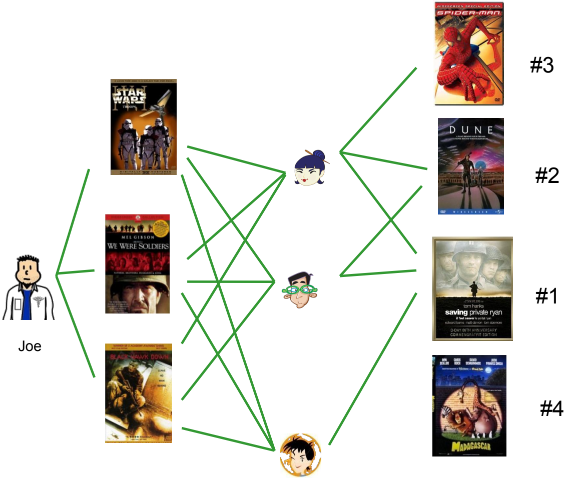
1		3			5			5		4	
		5	4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

users

movies

-1	0	-1	0	0	1	0	0	1	0	1	0
0	0	1	1	0	0	1	0	0	-1	-1	-1
-1	1	0	-1	-1	0	-1	0	1	-1	1	0
0	-1	1	0	1	0	0	1	0	0	-1	0
0	0	1	-1	1	-1	0	0	0	0	-1	1
-1	0	-1	0	-1	0	0	-1	0	0	1	0

Basic idea



Basic idea

users

items	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

Basic idea

users

items	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

similarity

$$s13 = 0.2$$

$$s16 = 0.3$$

Predict rating using weighted average

$$\frac{0.2 \cdot 2 + 0.3 \cdot 3}{0.2 + 0.3} = 2.6$$

Basic idea

(user, user) similarity to recommend items

(item, item) similarity to recommend new items that were also liked by the same users

The oldest known collaborative filtering method.

See Amazon recommendation system at scale:
`http://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf`

Similarities

How do we measure (user, user) similarity or (item, item) similarity?

- Euclidean distance
- Jaccard similarity
- Cosine similarity
- Pearson correlation

Notation:

$$\text{rating(user,item)} = \text{r(u,i)} = r_{ui}$$

$$I_u = \text{ set of items purchased by user } u$$

$$U_i = \text{ set of users who purchased by item } i$$

Euclidean distance

Distance between item i_1 and item i_2 is

$$\mathrm{dist}^2(i_1, i_2) = \sum_u (r_{u,i_1} - r_{u,i_2})^2$$

If each rating is 0 or 1 (user bought item or did not), then the distance becomes

$$\begin{aligned}\mathrm{dist}^2(i_1, i_2) =& \#\{\text{users that bought } i_1\text{, but not } i_2\} \\ &+ \#\{\text{users that bought } i_2\text{, but not } i_1\}\end{aligned}$$

Euclidean distance

Example:

$$U_1 = \{1, 4, 8, 9, 11, 23, 25, 34\}$$
$$U_2 = \{1, 4, 6, 8, 9, 11, 23, 25, 34, 35, 38\}$$
$$U_3 = \{4\}$$
$$U_4 = \{5\}$$

What is the distance between items 1 and 2?

What is the distance between items 3 and 4?

Euclidean distance

What is the distance between items 1 and 2? −> 3

What is the distance between items 3 and 4? −> 2

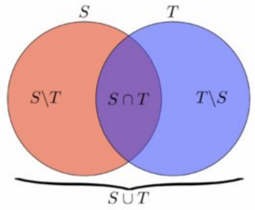
Problem:

- Favors small sets, even if they have few elements in common.

Jaccard similarity

Measures similarity between sets

$$\text{Jaccard}(U_i, U_j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|} = \frac{\text{bought i and j}}{\text{bought i or j}}$$



Key idea: normalize by popularity

Jaccard similarity

- Maximum of 1 if two items were purchased by the same set of users or if the two users purchased exactly the same set of items.

- Minimum of 0 if the two items were purchased by completely disjoint sets of users or if the two users purchased completely disjoint sets of items.

Jaccard similarity in action

How does amazon generate their recommendations?
http://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf

A user is looking at

U_i is the set of users who viewed this product.

Rank products according to $\frac{|U_i \cap U_j|}{|U_i \cup U_j|}$

.86 .84 .82 .79 ...

Cosine similarity

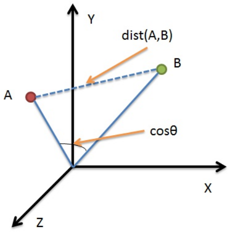
Jaccard similarity works only on 0/1 data

Cosine similarity works on arbitrary data.

	users											
movies	-1	0	-1	0	0	1	0	0	1	0	1	0
	0	0	1	1	0	0	1	0	0	-1	-1	-1
	-1	1	0	-1	-1	0	-1	0	1	-1	1	0
	0	-1	1	0	1	0	0	1	0	0	-1	0
	0	0	1	-1	1	-1	0	0	0	0	-1	1
	-1	0	-1	0	-1	0	0	-1	0	0	1	0

- 1 user viewed/bought an item and liked it
- -1 user viewed/bought an item and did not like it
- 0 user did not view/buy an item

Cosine similarity

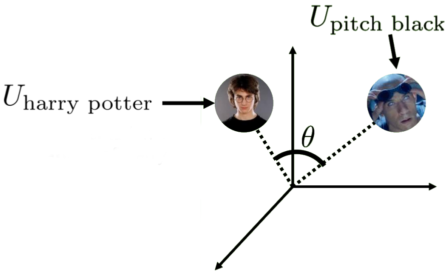


$$\text{similarity}(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

- $\cos(\theta) = 1$ $(\theta = 0)$ A and B point in the same direction
- $\cos(\theta) = -1$ $(\theta = 180)$ A and B point in the opposite direction
- $\cos(\theta) = 0$ $(\theta = 90)$ A and B are orthogonal

Cosine similarity

Each item is represented by a vector of users' ratings



$$U_{\text{harry potter}} = (0, 1, 1) \qquad U_{\text{pitch black}} = (1, 1, 0)$$

$$\text{similarity} = \frac{(0,1,1) \cdot (1,1,0)}{\sqrt{2} \cdot \sqrt{2}} = \frac{1}{2}$$

Pearson correlation

Very similar to cosine similarity

Cosine similarity would fail if naively applied to ratings

Example:

$$\begin{matrix} R_1 = (1,1,1) \\ R_2 = (5,5,5) \end{matrix} \longrightarrow \text{similarity} = 1$$

Pearson correlation will solve this by removing average from the rating vector

Pearson correlation

User ratings for item i:

1	?	?	5	5	3	?	?	?	4	2	?	?	?	?	4	?	5	4	1	?

User ratings for item j:

?	?	4	2	5	?	?	1	2	5	?	?	2	?	?	3	?	?	?	5	4

Pearson correlation computed over shared support

$$s_{ij} = \frac{\sum_{u \in U_i \cap U_j} (r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_{u \in U_i \cap U_j} (r_{ui} - \bar{r}_i)^2 \cdot \sum_{u \in U_i \cap U_j} (r_{uj} - \bar{r}_j)^2}}$$

Pearson correlation vs cosine similarity

Pearson correlation

$$s_{ij} = \frac{\sum_{u \in U_i \cap U_j} (r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_{u \in U_i \cap U_j} (r_{ui} - \bar{r}_i)^2 \cdot \sum_{u \in U_i \cap U_j} (r_{uj} - \bar{r}_j)^2}}$$

Cosine similarity

$$s_{ij} = \frac{\sum_{u \in U_i \cap U_j} r_{ui} \cdot r_{uj}}{\sqrt{\sum_{u \in U_i \cap U_j} r_{ui}^2 \cdot \sum_{u \in U_i \cap U_j} r_{uj}^2}}$$

How to use similarity to recommend?

Given similarity measure between items $s_i j$

Find set $s_k(i, u)$ of k-nearest neighbors to movie i that were rated by user u

Estimate rating using weighted average over the set of neighbors

$$\hat{r}_{ui} = \frac{\sum_{j \in s_k(i,u)} s_{ij} r_{uj}}{\sum_{j \in s_k(i,u)} s_{ij}}$$

Normalization and bias problem

Problems:

- Some items are significantly higher rated
- Some users rate substantially lower
- Ratings change over time

Bias correction is crucial for collaborative filtering approaches

- global bias μ
- offset per user b_u
- offset per movie b_i
- time effects (ignore for now)

Baseline rating for (user, movie) is $b(u, i) = \mu + b_u + b_i$

Normalization and bias problem

Mean rating of all movies $\mu = 3.7$

Troll hunter is 0.7 above mean ($b_i = 0.7$)

User rates 0.2 below mean ($b_u = -0.2$)

For this (user, movie) baseline rating is 4.2 stars

Estimating biases

$$\min_b \sum_{(u,i):r(u,i)\neq ?} (r(u,i) - \mu - b_u - b_i)^2 + \lambda(\sum_u b_u^2 + \sum_i b_i^2)$$

This is a linear model. Why?

How to recommend with biases?

Similar to the approch earlier

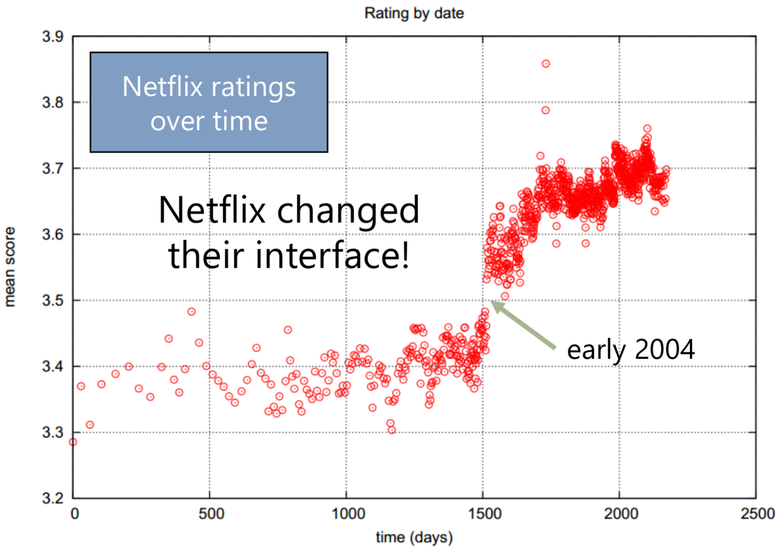
Given similarity measure between items $s_i j$

Find set $s_k(i, u)$ of k-nearest neighbors to movie i that were rated by user u

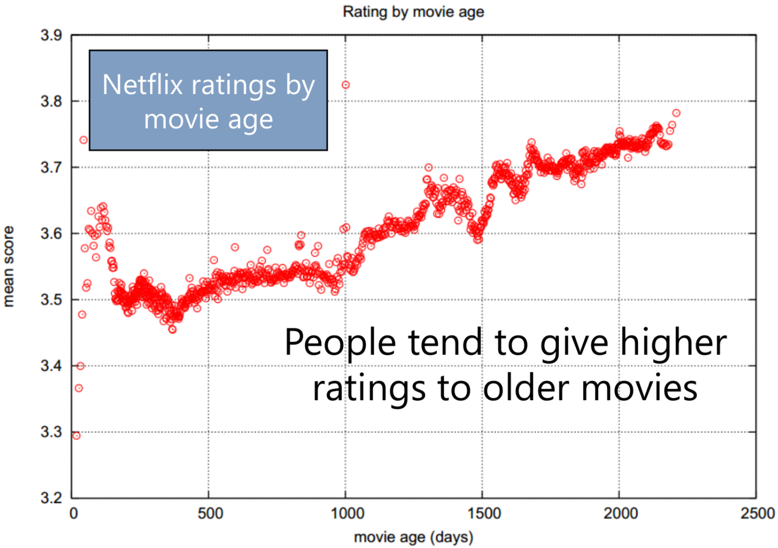
Estimate rating using weighted average over the set of neighbors

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in s_k(i,u)} s_{ij}(r_{uj} - b_{uj})}{\sum_{j \in s_k(i,u)} s_{ij}}$$

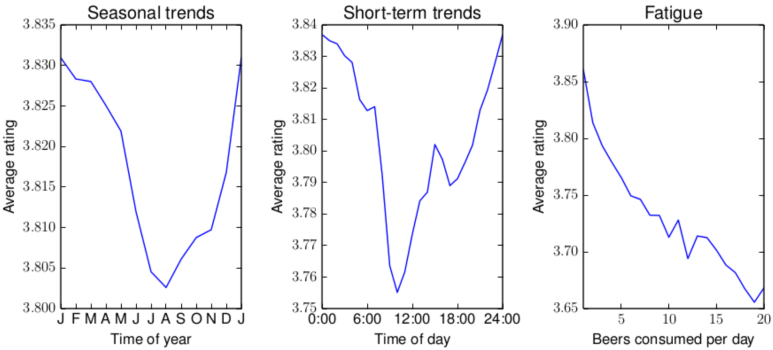
Temporal effects



Temporal effects



Temporal effects



A few temporal effects from beer reviews

Recommendations based on similarities

- intuitive
- there is no training
- easy to explain to a user
- widely used in practice
- surprisingly, collaborative filtering is extremely useful even though we have not looked at any features
- accurcy and scalability questionable
- cold start problem
- will not necessarily encourage diverse results

Cold start problem

What happens with new users where we have no ratings yet?

- Recommend popular items.
- Have some start-up questions (for example, "tell me 10 movies you love").

What do we do with new items?

- Content-based filtering techniques (that is, use features).
- Pay a focus group to rate them.

Modelling approach to recommendations

So far we've looked at approaches that try to define some definition of user/user and item/item similarity.

Recommendation consists of

1. Finding an item i that a user likes (gives a high rating)
2. Recommending items that are similar to it (that is, items j with a similar rating profile to i)

Hinges on finding a good measure of similarity

Modelling approach to recommendations

What we want to do next is to model the rating.

$$r_{ui} = f(\text{user}, \text{item}) + \text{noise}$$

Recommendation consists of identifying items with largest rating

$$\text{recommendation}(u) = \arg\max_{i \in \text{unseen items}} f(u, i)$$

Netflix yardstick

Netflix prize

- In 2006, Netflix created a dataset of 100,000,000 movie ratings
- Data looked like: (userID, itemID, time, rating)
- Whoever first manages to reduce the (R)MSE by 10% versus Netflix's solution wins $1,000,000
- Data were de-anonymized — lawsuit againt Netflix

Root mean squared error for predicting ratings

$$\mathrm{RMSE}(f) = \sqrt{\frac{1}{N}\sum_{u,i}(f(u,i) - r_{ui})^2}$$

Predicted rating $\hat{r}_{ui} = f(u,i)$

Netflix yardstick

A lot of research on minimizing the mean squared error.

Not clear that improving this metric will lead to better user experience.

When building a model we focus on minimizing root mean squared error.

Latent factor model

Suppose we had K features of movies and users

Describe movie i with features q_i

- How much is it action, romance, drama, ...

$$q_i = (0.9, 0.2, 0.5, \ldots)$$

Describe user u with features p_u

- How much she likes action, romance, drama, ...

$$p_u = (0.01, 0, 0.9, \ldots)$$

$f(u, i)$ is the product of the two vectors

$$f(u, i) = 0.9 \cdot 0.01 + 0.2 \cdot 0 + 0.5 \cdot 0.9 + \ldots$$

Discovering features via matrix factorization

users

items

1		3			5			5		4	
		5	4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

~

~

items

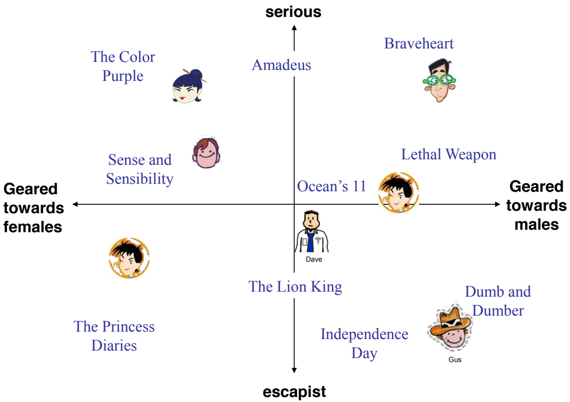
.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

•

users

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

Discovering features via matrix factorization



Discovering features via matrix factorization

users

items

1		3			5			5		4	
		5		?		4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

~

~

items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

•

users

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

Discovering features via matrix factorization

users

1		3			5			5		4	
		5		?		4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

items ~

~ items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

●

users

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

Discovering features via matrix factorization

users

items

1		3			5			5		4	
		5	2.4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

~

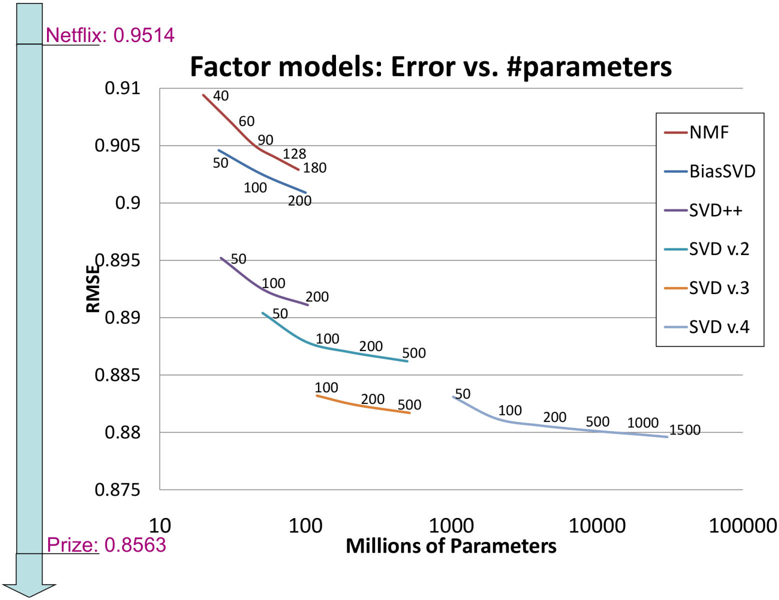
~ items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

•

users

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1


Netflix: 0.9514
Factor models: Error vs. #parameters
0.91
0.905
0.9
0.895
0.89
0.885
0.88
0.875
RMSE
10
100
1000
10000
100000
Millions of Parameters
NMF
BiasSVD
SVD++
SVD v.2
SVD v.3
SVD v.4
40
60
90
128
180
50
100
200
50
100
200
50
100
200
500
100
200
500
50
100
200
500
1000
1500
Prize: 0.8563

Finding parameters of the latent factor model

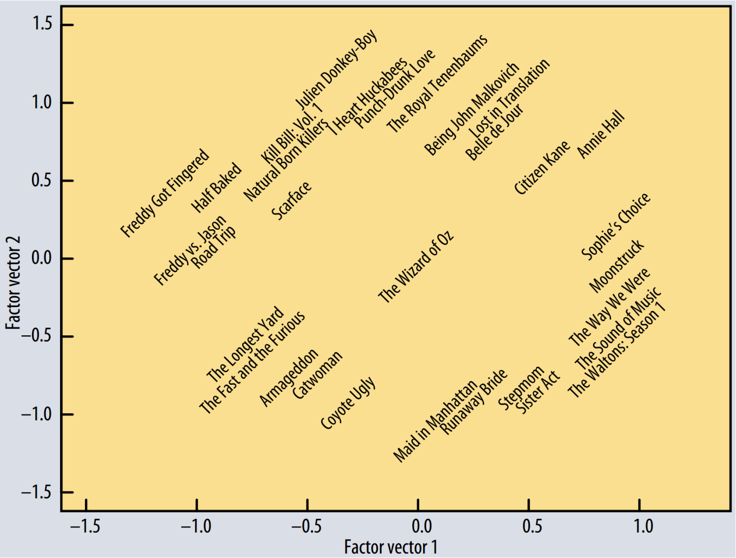
Model $f(u, i) = p_u \cdot q_i$

We will find p and q by minimizing the following objective

$$\min_{p,q} \sum_{(u,i):r_{ui} \neq ?} (r_{ui} - p_u \cdot q_i)^2 + \lambda(\sum_{u,k} p_{u,k}^2 + \sum_{i,k} q_{i,k}^2)$$

Stochastic gradient descent

Visualizing the first two factors



Visualizing the first two factors

First factor:

- left: lowbrow comedies and horror movies, aimed at a male or adolescent audience
- right: drama or comedy with serious undertones and strong female leads

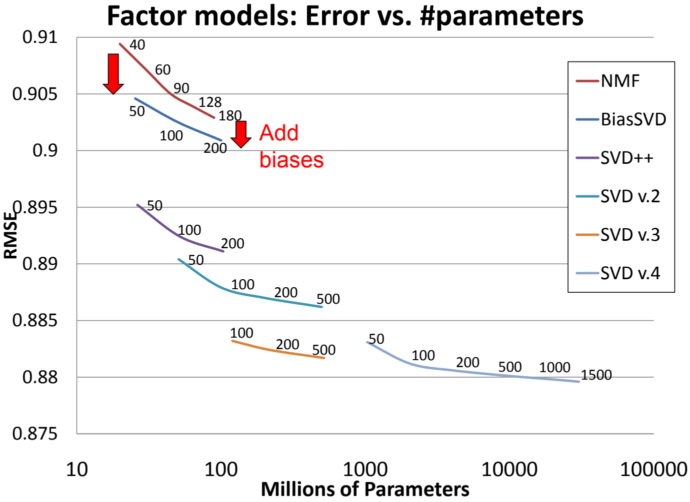
Second factor:

- top: independent, critically acclaimed, quirky films
- bottom: mainstream formulaic films

Improvements

- adding bias terms
- adding features and implicit feedback
- modelling temporal effects

Adding biases



$$f(u,i) = \mu + b_u + b_i + p_u \cdot q_i$$

Combining real and discovered features

Real features capture context

- Time of the day, what I just saw, user info, what I bought in the past

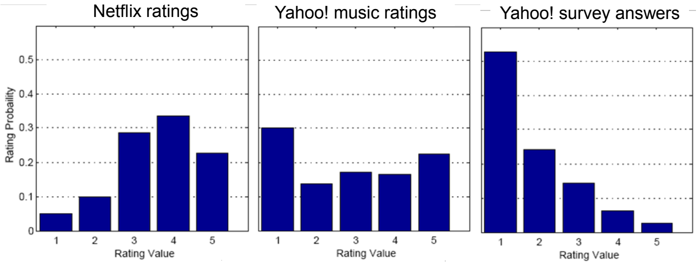
Discovered features from matrix factorization capture groups of users who behave similarly

- Users who like action movies and comedies

Mitigates cold-start problem

- Ratings for a new user from real features only
- As more information about user is discovered, matrix factorization "features" become more relevant

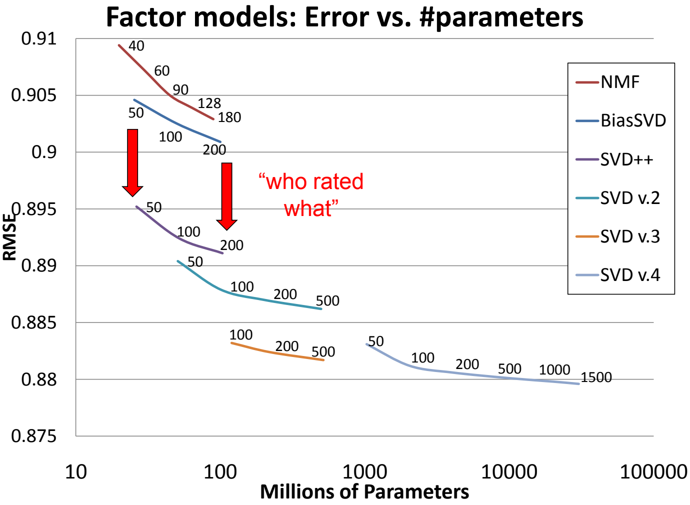
Implicit information



Our decision about whether to purchase a movie is a function of how we expect to rate it.

For items we have purchased, our decision to enter a rating or write a review is a function of our rating.

http://www.cs.toronto.edu/~marlin/research/papers/cfmar-uai2007.pdf


Factor models: Error vs. #parameters
RMSE
0.91
0.905
0.9
0.895
0.89
0.885
0.88
0.875
Millions of Parameters
10
100
1000
10000
100000
NMF
BiasSVD
SVD++
SVD v.2
SVD v.3
SVD v.4
40
60
90
128
180
50
100
200
50
100
200
50
100
200
500
100
200
500
50
100
200
500
1000
1500
“who rated what”

Modeling temporal change

Time-dependent bias

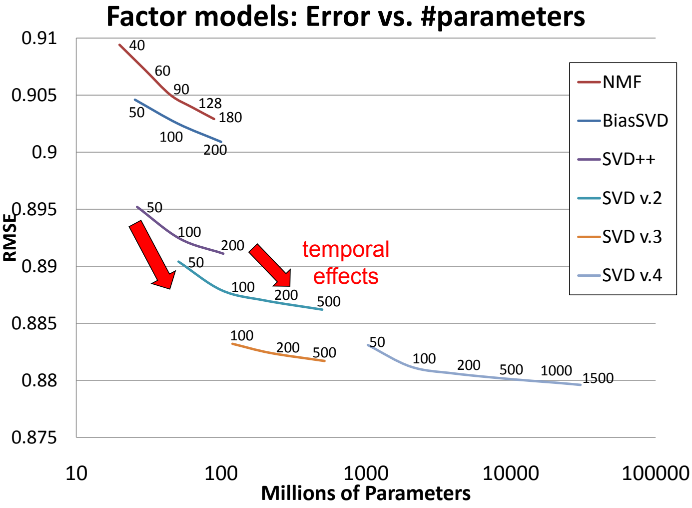
Time-dependent user preferences

Parameterize functions b and p

$$f(u, i, t) = \mu + b_u(t) + b_i(t) + q_i \cdot p_u(t)$$

Good parametrization is the key.

http://www.cc.gatech.edu/~zha/CSE8801/CF/kdd-fp074-koren.pdf



Factor models: Error vs. #parameters
RMSE
0.91
0.905
0.9
0.895
0.89
0.885
0.88
0.875
10
100
1000
10000
100000
Millions of Parameters
NMF
BiasSVD
SVD++
SVD v.2
SVD v.3
SVD v.4
40
60
90
128
180
50
100
200
50
100
200
50
100
200
500
100
200
500
50
100
200
500
1000
1500
temporal effects

Moral of the story

Increasing the number of parameters does not help much, but increasing the model complexity does.

\$1,000,000 seems to be incredibly cheap to get the amount of research that was devoted to the task.

The winning solution never made it into production at Netflix.

It is not clear that a solution which changes RMSE slightly will result in hugely improved user experience.