GENERAL ASSEMBLY

INTRO TO DATA SCIENCE

SESSION 12: RECOMMENDER SYSTEMS

Rob Hall

DAT13 SF // April 15, 2015 (Have you done your taxes?)

LAST TIME:

- HANDS-ON SQL TUTORIAL
- USING SQLITE AND PYTHON
- DATABASE CONCEPTS, NOSQL

QUESTIONS?

AGENDA

I. CONTENT-BASED FILTERING
II. COLLABORATIVE FILTERING

LABS:
III-A. BEER RECOMMENDER
III-B. PYTHON-RECSYS WITH MOVIELENS DATA

IV. CONCLUSION: THE NETFLIX PRIZE
V. APPENDIX (FOR SELF-STUDY): MATRIX FACTORIZATION

A **recommendation system** *aims to match users to products/items/brand/etc that they likely haven't experienced yet.*

A **recommendation system** *aims to match users to products/items/brand/etc that they likely haven't experienced yet.*

This rating is produced by analyzing other user/item ratings (and sometimes item characteristics) to provide personalized recommendations to users.

There are two general approaches to the design:

There are two general approaches to the design:

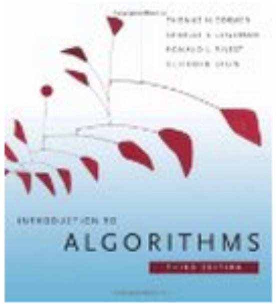
In **content-based filtering**, *items are mapped into a feature space, and recommendations depend on item characteristics.*

In contrast, the only data under consideration in **collaborative filtering** *are user-item ratings, and recommendations depend on user preferences.*

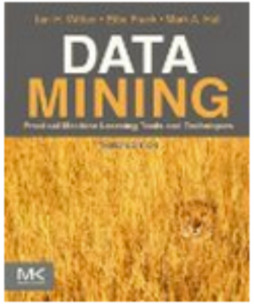
EXAMPLES – AMAZON CONTENT-BASED

Recommendations for You in Books

Cracking the Coding Interview: 150...
› Gayle Laakmann McDowell
Paperback
(166)
~~$39.95~~ $23.22
Why recommended?

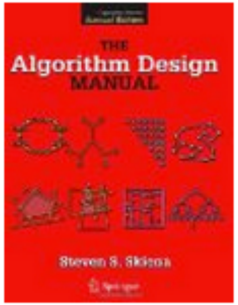


Introduction to Algorithms
Thomas H. Cormen, Charles E...
Hardcover
(85)
~~$92.00~~ $80.00
Why recommended?



Data Mining: Practical Machine...
› Ian H. Witten, Eibe Frank, Mark A. Hall
Paperback
(27)
~~$69.95~~ $42.09
Why recommended?

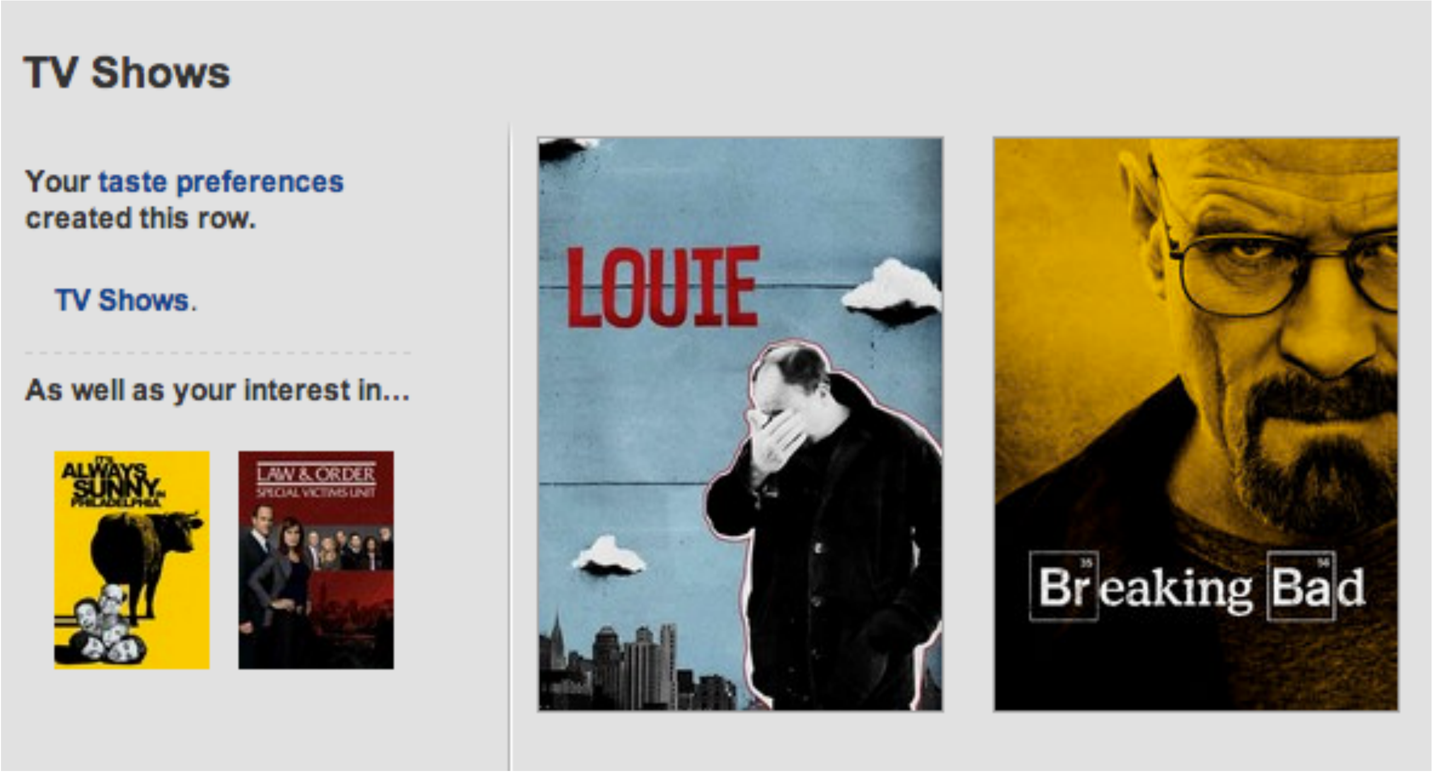
Elements of Programming Interviews...
› Amit Prakash, Adnan Aziz, Tsung-Hsien Lee
Paperback
(25)
~~$29.99~~ $26.18
Why recommended?



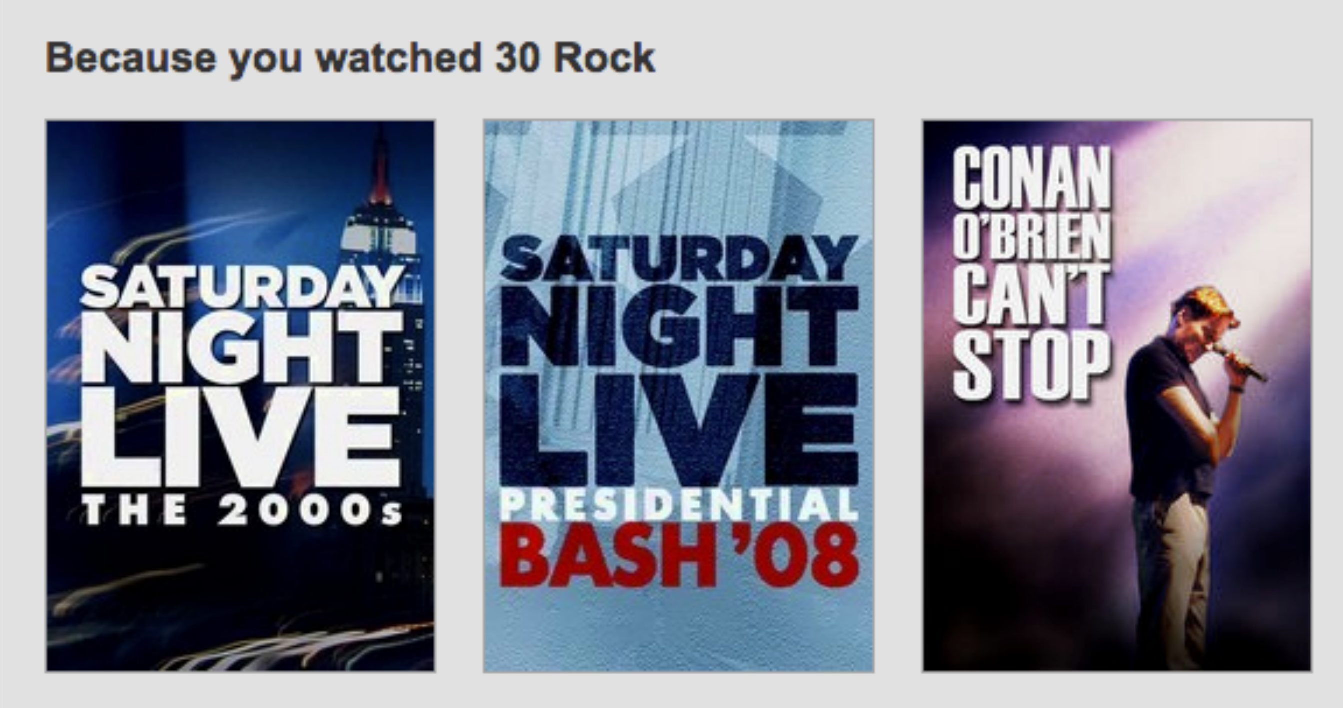
The Algorithm Design Manual
› Steve Skiena
Paperback
(47)
~~$89.95~~ $71.84
Why recommended?

EXAMPLES – AMAZON COLLABORATIVE FILTERING

EXAMPLES – NETFLIX



EXAMPLES – NETFLIX



EXAMPLES – YOUTUBE

Recommended for you because you watched
Sugar Minott - Oh Mr Dc (Studio One)

Mikey Dread - Roots and Culture
by klaxonklaxon · 1,164,133 views

Lyrics:
Now here comes a special request
To each and everyone

Recommended for you because you watched
Thelonious Monk Quartet - Monk In Denmark

Bill Evans Portrait in Jazz (Full Album)
by hansgy1 · 854,086 views

Bill Evans Portrait in Jazz 1960
1. Come Rain or Come Shine - 3.19 (0:00)
2. Autumn Leaves - 5.23 (3:24)

Recommended for you because you watched
Bob Marley One Drop

Bob Marley - She's gone
by Dionysios29 · 1,058,704 views

This is one of the eleven songs of album Kaya that Bob Marley and The Wailers creative in 1978.
Lyrics:

How can we find good recommendations?

- Manual Curation
- Manually Tag Attributes
- Audio Content, Metadata, Text Analysis
- Collaborative Filtering

content-based filtering

EXAMPLES – NYTIMES.COM

MOST E-MAILED | RECOMMENDED FOR YOU

1. How Big Data Is Playing Recruiter for Specialized Workers
2. SLIPSTREAM
   When Your Data Wanders to Places You've Never Been
3. MOTHERLODE
   The Play Date Gun Debate
4. For Indonesian Atheists, a Community of Support Amid Constant Fear
5. Justice Breyer Has Shoulder Surgery
6. BILL KELLER
   Erasing History

EXAMPLES – NYTIMES.COM

8. How do you determine my Most Read Topics? Back to top ▲

Each NYTimes.com article is assigned topic tags that reflect the content of the article. As you read articles, we use these tags to determine your most-read topics.

To search for additional articles on one of your most-read topics, click that topic on your personalized Recommendations page. To learn more about topic tags, visit Times Topics.

NOTE

Collaborative or Content based?

EXAMPLES – NYTIMES.COM

8. How do you determine my Most Read Topics?

Back to top ▲

Each NYTimes.com article is assigned topic tags that reflect the content of the article. As you read articles, we use these tags to determine your most-read topics.

To search for additional articles on one of your most-read topics, click that topic on your personalized Recommendations page. To learn more about topic tags, visit Times Topics.

NOTE

Collaborative or Content based?

CONTENT BASED ☺

I. CONTENT-BASED FILTERING

Content-based filtering *begins by mapping each item into a feature space. Both users and items are represented by vectors in this space.*

Content-based filtering *begins by mapping each item into a feature space. Both users and items are represented by vectors in this space.*

Item vectors *measure the degree to which the item is described by each feature, and* ***user vectors*** *measure a user's preferences for each feature.*

Content-based filtering *begins by mapping each item into a feature space. Both users and items are represented by vectors in this space.*

Item vectors *measure the degree to which the item is described by each feature, and* ***user vectors*** *measure a user's preferences for each feature.*

Ratings are generated by taking ***dot products*** *of user & item vectors.*

features = (big box office, aimed at kids, famous actors)

Items (movies):

Finding Nemo = (5, 5, 2)
Mission Impossible = (3, -5, 5)
Jiro Dreams of Sushi = (-4, -5, -5)

features = (big box office, aimed at kids, famous actors)

Items (movies):

Finding Nemo = (5, 5, 2)

Mission Impossible = (3, -5, 5)

Jiro Dreams of Sushi = (-4, -5, -5)

Users:

Alice = (-3, 2, -2)

Bob = (4, -3, 5)

EXAMPLE – CONTENT-BASED FILTERING

features = (big box office, aimed at kids, famous actors)

Items (movies):

Finding Nemo = (5, 5, 2)

Mission Impossible = (3, -5, 5)

Jiro Dreams of Sushi = (-4, -5, -5)

User:

Alice = (-3, 2, -2)

Prediction (for Alice)

5-3 + 5*2 + 2*-2 = -9*

EXAMPLE – CONTENT-BASED FILTERING

features = (big box office, aimed at kids, famous actors)

Items (movies):	**Prediction (for Alice)**
Finding Nemo = (5, 5, 2)	*5*-3 + 5*2 + 2*-2 = -9*
Mission Impossible = (3, -5, 5)	*3*-3 + -5*2 + 5*-2 = -29*
Jiro Dreams of Sushi = (-4, -5, -5)	

User:

Alice = (-3, 2, -2)

features = (big box office, aimed at kids, famous actors)

Items (movies):	**Prediction (for Alice)**	
Finding Nemo = (5, 5, 2)	*5*-3 + 5*2 + 2*-2*	*= -9*
Mission Impossible = (3, -5, 5)	*3*-3 + -5*2 + 5*-2*	*= -29*
Jiro Dreams of Sushi = (-4, -5, -5)	*-4*-3 + -5*2 + -5*-2*	*= +12*

User:

Alice = (-3, 2, -2)

EXAMPLE – CONTENT-BASED FILTERING

features = (big box office, aimed at kids, famous actors)

Items (movies):	***Prediction (for Alice)***	
Finding Nemo = (5, 5, 2)	*5*-3 + 5*2 + 2*-2*	*= -9*
Mission Impossible = (3, -5, 5)	*3*-3 + -5*2 + 5*-2*	*= -29*
Jiro Dreams of Sushi = (-4, -5, -5)	*-4*-3 + -5*2 + -5*-2*	*= +12*

User:

Alice = (-3, 2, -2)

features = (big box office, aimed at kids, famous actors)

Items (movies):

Finding Nemo = (5, 5, 2)

Mission Impossible = (3, -5, 5)

Jiro Dreams of Sushi = (-4, -5, -5)

Prediction (for Bob)

User:

Bob = (4, -3, 5)

features = (big box office, aimed at kids, famous actors)

Items (movies):	***Prediction (for Bob)***	
Finding Nemo = (5, 5, 2)	*5*4 + 5*-3 + 2*5*	*= +15*
Mission Impossible = (3, -5, 5)	*3*4 + -5*-3 + 5*5*	*= +52*
Jiro Dreams of Sushi = (-4, -5, -5)	*-4*4 + -5*-3 + -5*5*	*= -26*

User:

Bob = (4, -3, 5)

EXAMPLE – CONTENT-BASED FILTERING

features = (big box office, aimed at kids, famous actors)

Items (movies):	**Prediction (for Bob)**	
Finding Nemo = (5, 5, 2)	*5*4 + 5*-3 + 2*5*	*= +15*
Mission Impossible = (3, -5, 5)	*3*4 + -5*-3 + 5*5*	*= +52*
Jiro Dreams of Sushi = (-4, -5, -5)	*-4*4 + -5*-3 + -5*5*	*= -26*

User:

Bob = (4, -3, 5)

One notable example of content-based filtering is Pandora, which maps songs into a feature space using features (or “genes”) designed by the Music Genome Project.

Using song vectors that depend on these features, Pandora can create a station with music having similar properties to a song the user selects.

VISUALIZATION OF SIMILAR ARTISTS

http://www.music-map.com/

Content-based filtering has some difficulties:

Content-based filtering has some difficulties:

- *Must map items into a feature space (usually by hand!)*
- *Recommendations are limited in scope (items must be similar to each other)*
- *Hard to create cross-content recommendations (eg books/music films…this would require comparing elements from different feature spaces!)*

II. COLLABORATIVE FILTERING

Collaborative filtering *refers to a family of methods for predicting ratings where instead of thinking about users and items in terms of a feature space, we are only interested in the existing user-item ratings themselves.*

Collaborative filtering *refers to a family of methods for predicting ratings where instead of thinking about users and items in terms of a feature space, we are only interested in the existing user-item ratings themselves.*

In this case, our dataset is a ratings matrix whose columns correspond to items, and whose rows correspond to users.

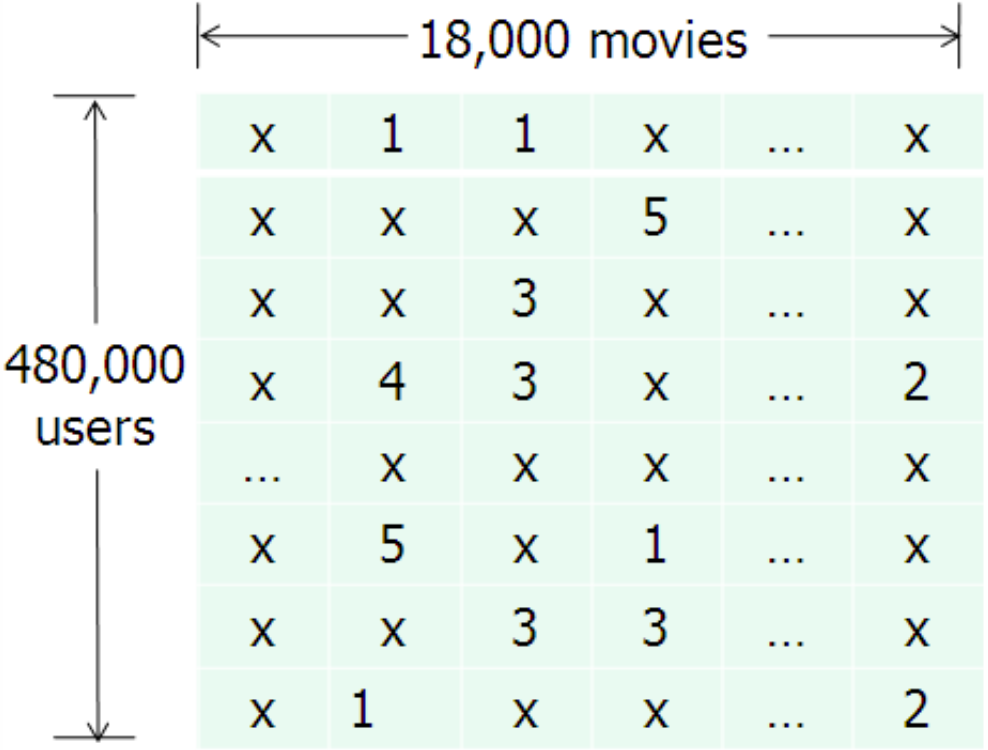
Collaborative filtering *refers to a family of methods for predicting ratings where instead of thinking about users and items in terms of a feature space, we are only interested in the existing user-item ratings themselves.*

In this case, our dataset is a ratings matrix whose columns cor[...] to items, and whose rows correspond to users.

NOTE

The idea here is that users get value from recommendations based on other users with similar *tastes*.

RATINGS MATRIX



NOTE

This matrix will always be *sparse*!

source: http://www.eecs.berkeley.edu/~zhanghao/main/publications/subfolder/netflix.png

Main difference between content and collaborative filtering:

Content Based:

maps items and users into a feature space

Collaborative:

relies on previous user-item ratings

We will look at collaborative filtering in a user-user sense.

We will look at collaborative filtering in a user-user sense.

We will take a given user, and find the K most similar users, and then recommend brands from the similar users!

We will look at collaborative filtering in a user-user sense.

We will take a given user, and find the K most similar users, and then recommend brands from the similar users!

NOTE

Sound familiar? It's similar to KNN!

ITEM-BASED COLLABORATIVE FILTERING

The system cannot draw inferences because it hasn’t gathered enough information yet.

The cold start problem arises because we've been relying only on ratings data, or on **explicit feedback** *from users.*

The cold start problem arises because we've been relying only on ratings data, or on **explicit feedback** *from users.*

Until users rate several items, we don't know anything about their preferences!

COLD START PROBLEM

The cold start problem arises because we've been relying only on ratings data, or on **explicit feedback** *from users.*

Until users rate several items, we don't know anything about their preferences!

We can get around this by enhancing our recommendations using **implicit feedback***, which may include things like item browsing behavior, search patterns, purchase history, etc.*

While explicit feedback (ratings, likes, purchases) leads to high quality ratings, the data is sparse and cold starts are problematic.

While explicit feedback (ratings, likes, purchases) leads to high quality ratings, the data is sparse and cold starts are problematic.

Meanwhile implicit feedback (browsing behavior, etc.) leads to less accurate ratings, but the data is much more dense (and less invasive to collect).

III. LAB: PYTHON EXAMPLE

IV. THE NETFLIX PRIZE

The Netflix prize was a competition to see if anyone could make a 10% improvement to Netflix's recommendation system (accuracy measured by RMSE).

THE NETFLIX PRIZE

The Netflix prize was a competition to see if anyone could make a 10% improvement to Netflix's recommendation system (accuracy measured by RMSE).

The grand prize was $1m dollars

The Netflix prize was a competition to see if anyone could make a 10% improvement to Netflix's recommendation system (accuracy measured by RMSE).

The grand prize was $1m dollars

The ratings matrix contained >100mm numerical entries (1-5 stars) from ~500k users across ~17k movies. The data was split into train/quiz/test sets to prevent overfitting on the test data by answer submission (this was a clever idea!)

The competition began in 2006, and the grand prize was eventually awarded in 2009. The winning entry was a stacked ensemble of 100's of models (including neighborhood & matrix factorization models) that were blended using boosted decision trees.

Ultimately, the competition ended in a photo finish. The winning strategy came down to last-minute team mergers & creative blending schemes to shave 3rd & 4th decimals off RMSE (concerns that would not be important in practice).

The competition began in 2006, and the grand prize was eventually awarded in 2009. The winning entry was a stacked ensemble of 100's of models (including neighborhood & matrix factorization models) that were blended using boosted decision trees.

Ultimately, the competition ended in a photo finish. The winning strategy came down to last-minute team mergers & creative blending schemes to shave 3rd & 4th decimals off RMSE (concerns that would not be important in practice).

The competition did much to spur interest and research advances in recsys technology, and the prize money was donated to charity.

Though they adopted some of the modeling techniques that emerged from the competition, Netflix never actually implemented the prizewinning solution.

Why do you think that's true?

V. APPENDIX: A SIMPLE MATRIX FACTORIZATION MODEL

Matrix factorization decomposes the ratings matrix and maps users and items into a low-dimensional vector space spanned by a basis of latent factors.

SIMPLE MATRIX FACTORIZATION MODEL

Matrix factorization decomposes the ratings matrix and maps users and items into a low-dimensional vector space spanned by a basis of latent factors.

Predicted ratings are given by inner products in this space, so for user u and item i we can write:

$$\hat{r}_{ui} = q_i^T r_u$$

Factoring the ratings matrix via SVD leads to difficulty, since the matrix is typically sparse and therefore our information about the data is incomplete.

Factoring the ratings matrix via SVD leads to difficulty, since the matrix is typically sparse and therefore our information about the data is incomplete.

Interpolating missing values is an expensive process and can lead to inaccurate predictions, so we need another way to perform this factorization.

One possibility is to learn the feature vectors using the observed ratings only. Since this dramatically reduces the size of the ratings matrix, we have to be careful to avoid overfitting.

One possibility is to learn the feature vectors using the observed ratings only. Since this dramatically reduces the size of the ratings matrix, we have to be careful to avoid overfitting.

We can learn these feature vectors by minimizing the loss function:

$$\min_{q,p} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2)$$

where κ denotes the set of known ratings, and λ is a hyperparameter.

One possibility is to learn the feature vectors using the observed ratings only. Since this dramatically reduces the size of the ratings … have to be careful to avoid overfitting.

We can learn these feature vectors by minimizing the loss f…

NOTE

The loss function has two unknowns (q, p) and so is not convex!

This can be minimized using a method called *alternating least squares*.

$$\min_{q,p} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2)$$

where κ *denotes the set of known ratings, and* λ *is a hyperparameter.*

It turns out that much of the variation in observed ratings is due to user or item biases (eg, some users are very critical, or some items are universally popular).

SIMPLE MATRIX FACTORIZATION MODEL

It turns out that much of the variation in observed ratings is due to user or item biases (eg, some users are very critical, or some items are universally popular).

We can capture these biases in our model by generalizing $\hat{r}_{ui}$:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T r_u$$

Here μ *is a global average rating,* b_i *is the item bias,* b_u *is the user bias, and* $q_i^T r_u$ *is the user-item interaction.*

With this generalization, our minimization problem becomes:

$$\min_{q,p,b} \sum_{(u,i)\in\kappa} (r_{ui} - \mu - b_u - b_i - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2 + b_u^2 + b_i^2)$$

With this generalization, our minimization problem becomes:

$$\min_{q,p,b} \sum_{(u,i)\in\kappa} (r_{ui} - \mu - b_u - b_i - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2 + b_u^2 + b_i^2)$$

Further modifications can be made to this model (incorporating implicit feedback, capturing temporal effects, attaching confidence scores to predictions), and you can look up the details in the references.