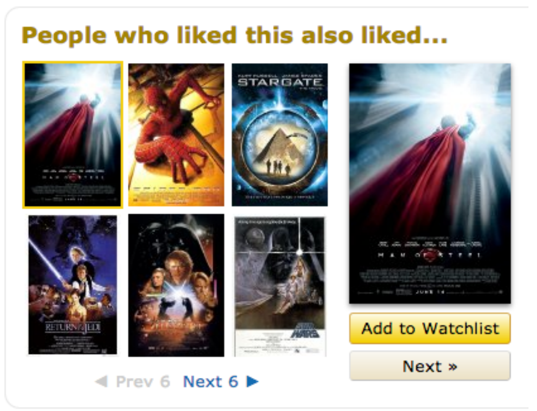
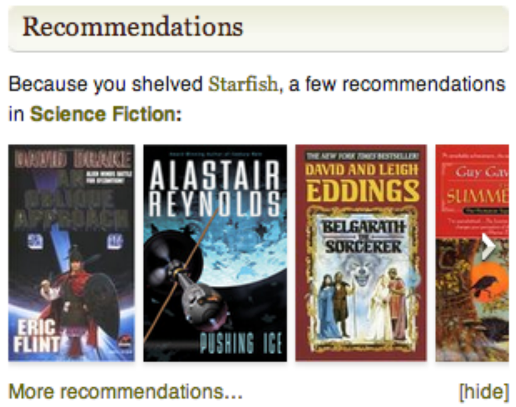
Recommender Systems

Techniques of AI

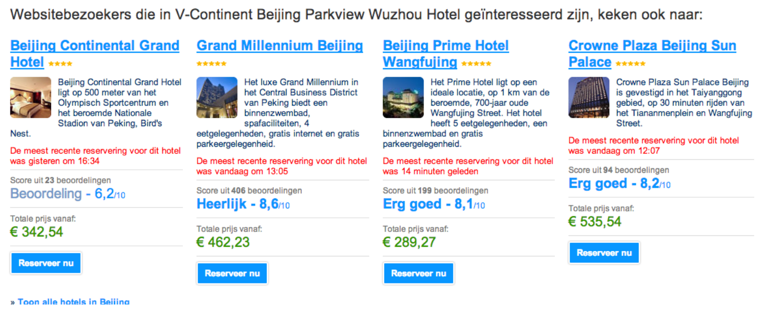
Recommender Systems


People who liked this also liked...


Recommendations

Because you shelved Starfish, a few recommendations in Science Fiction:

More recommendations... [hide]


Websitebezoekers die in V-Continent Beijing Parkview Wuzhou Hotel geïnteresseerd zijn, keken ook naar:

Beijing Continental Grand Hotel ★★★★

Beijing Continental Grand Hotel ligt op 500 meter van het Olympisch Sportcentrum en het beroemde Nationale Stadion van Peking, Bird's Nest.

De meest recente reservering voor dit hotel was gisteren om 16:34

Score uit 23 beoordelingen
Beoordeling - 6,2/10

Totale prijs vanaf:
€ 342,54

Reserveer nu

Grand Millennium Beijing ★★★★★

Het luxe Grand Millennium in het Central Business District van Peking biedt een binnenzwembad, spafaciliteiten, 4 eetgelegenheden, gratis internet en gratis parkeergelegenheid.

De meest recente reservering voor dit hotel was vandaag om 13:05

Score uit 406 beoordelingen
Heerlijk - 8,6/10

Totale prijs vanaf:
€ 462,23

Reserveer nu

Beijing Prime Hotel Wangfujing ★★★★★

Het Prime Hotel ligt op een ideale locatie, op 1 km van de beroemde, 700-jaar oude Wangfujing Street. Het hotel heeft 5 eetgelegenheden, een binnenzwembad en gratis parkeergelegenheid.

De meest recente reservering voor dit hotel was 14 minuten geleden

Score uit 199 beoordelingen
Erg goed - 8,1/10

Totale prijs vanaf:
€ 289,27

Reserveer nu

Crowne Plaza Beijing Sun Palace ★★★★★

Crowne Plaza Sun Palace Beijing is gevestigd in het Taiyanggong gebied, op 30 minuten rijden van het Tiananmenplein en Wangfujing Street.

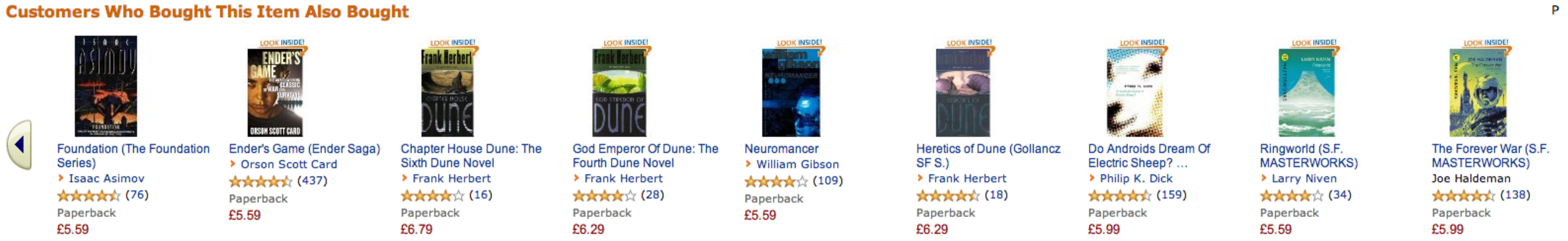
De meest recente reservering voor dit hotel was vandaag om 12:07

Score uit 94 beoordelingen
Erg goed - 8,2/10

Totale prijs vanaf:
€ 535,54

Reserveer nu

» Toon alle hotels in Beijing


Customers Who Bought This Item Also Bought

Foundation (The Foundation Series)	Isaac Asimov	★★★★½	(76)	Paperback	£5.59
Ender's Game (Ender Saga)	Orson Scott Card	★★★★½	(437)	Paperback	£5.59
Chapter House Dune: The Sixth Dune Novel	Frank Herbert	★★★★☆	(16)	Paperback	£6.79
God Emperor Of Dune: The Fourth Dune Novel	Frank Herbert	★★★★☆	(28)	Paperback	£6.29
Neuromancer	William Gibson	★★★★☆	(109)	Paperback	£5.59
Heretics of Dune (Gollancz SF S.)	Frank Herbert	★★★★½	(18)	Paperback	£6.29
Do Androids Dream Of Electric Sheep? ...	Philip K. Dick	★★★★½	(159)	Paperback	£5.99
Ringworld (S.F. MASTERWORKS)	Larry Niven	★★★★☆	(34)	Paperback	£5.59
The Forever War (S.F. MASTERWORKS)	Joe Haldeman	★★★★½	(138)	Paperback	£5.99

User ratings

- Collect user preferences (scores, likes, purchases, views ...)
- Find similarities between items and/or users
- Predict user scores for new items
- Recommend items with high predicted scores

The Problem

Users

Ratings

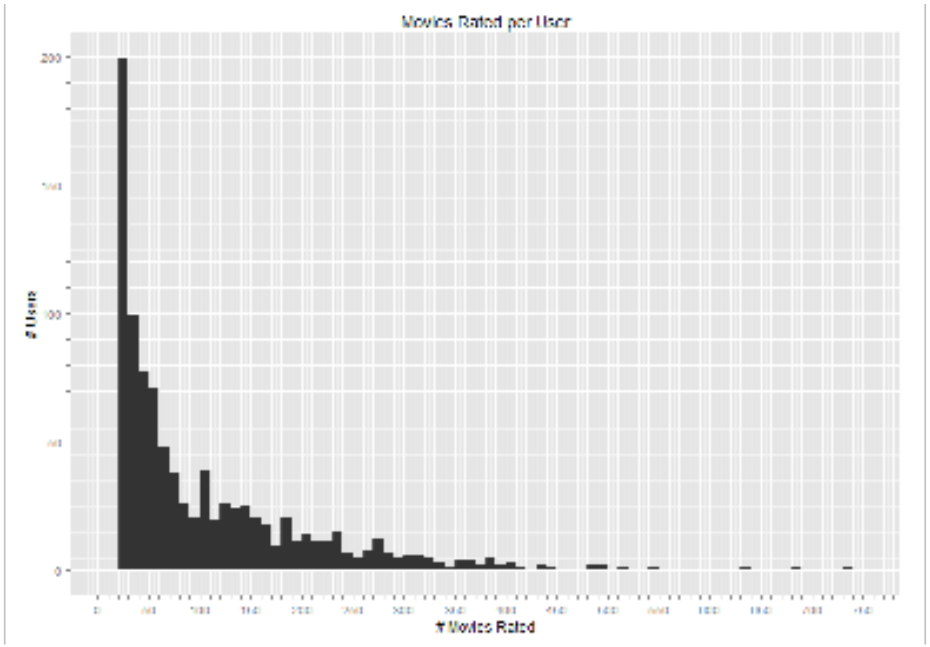
Items

movie / user	Alice	Bob	Carol	Dave
The Dark Knight	?	0	1	4
Die Hard	5	0	?	?
The Hunt for Red October	5	?	0	5
Love Actually	?	5	4	0
Bridget Jones Diary	1	5	?	?
Four Weddings and a Funeral	0	?	5	1

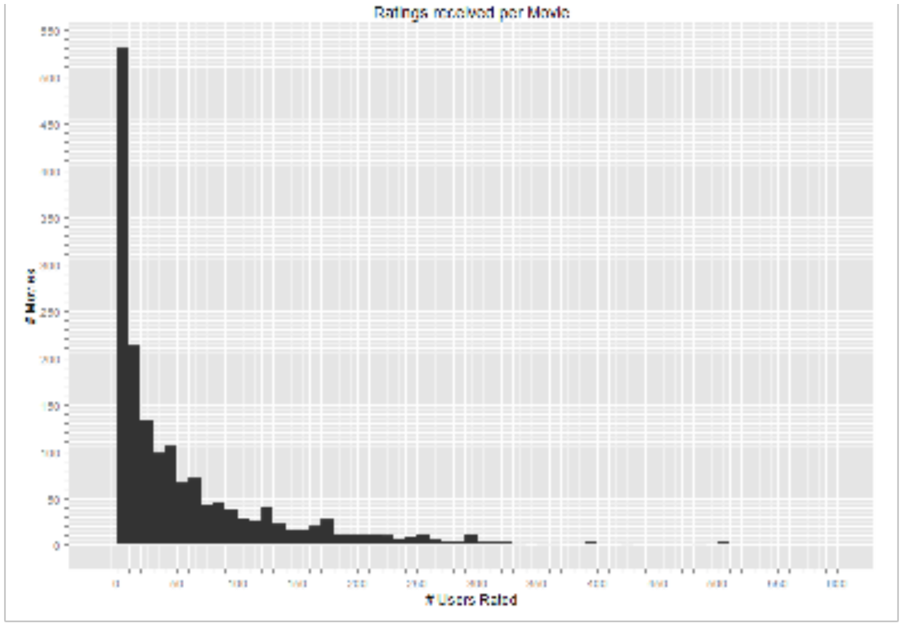
Fill in ?s

Rating Distributions

# Movies rated per user



rating received per movie



Recommender Approaches

- Content-Based Recommenders:

  Use item descriptions (metadata) to map items to scores

- Collaborative Filtering:

  Predict scores by finding similarities between item and/or user scores

- Hybrid Approaches:

  Combine above approaches

Content Based Approaches

- Rely on descriptions of items (features)
- **Case-Based**: find items similar to items user likes
- **Profile-Based**: build user profile of likes/dislikes & match items to profile
- Do not directly rely on similarities between users

Content Based Systems

movie	Director	Lead actor	Genre	Period
The Dark Knight	Christopher Nolan	Christian Bale	action	00’s
Die Hard	John McTiernan	Bruce Willis	action	80’s
The Hunt for Red October	John McTiernan	Sean Connery	action	90’s
Love Actually	Richard Curtis	Hugh Grant	romance	00s
Bridget Jones Diary	Sharon Maguire	Renée Zellweger	romance	00s
Four Weddings and a Funeral	Mike Newell	Hugh Grant	romance	90s

Case Based Reasoning

See Mitchell book section 8.5 for details on CBR

Lazy learning - based on querying stored data, rather than explicit model:

1. Store samples of the form <item features, rating>
2. Define a similarity between items. (e.g. simple similarity for the movie example: how many features match)
3. To make a prediction, simply retrieve unrated items similar to items that have high ratings

Case-Based

- Bob likes:

Love Actually	Richard Curtis	Hugh Grant	romance	00s

➡ recommend *Four Weddings and a Funeral*
(same genre & lead actor =similarity 2)

- Dave Likes:

The Hunt for Red October	John McTiernan	Sean Connery	action	90’s

➡ recommend *Die Hard* (same director & genre = similarity 2)

Profile Based Recommendations

Main idea:

1. Aggregate rated items to build user taste profile
2. Match unrated items to user’s profile
3. recommend best matches

Profile Based

Alice Likes:

movie / user	Director	Lead actor	Genre	release
Die Hard	John McTiernan	Bruce Willis	action	80’s
The Hunt for Red October	John McTiernan	Sean Connery	action	90’s

Profile for Alice: < John McTiernan,?, action,? >

-> Recommend action movies by John McTiernan

Vector Space Model

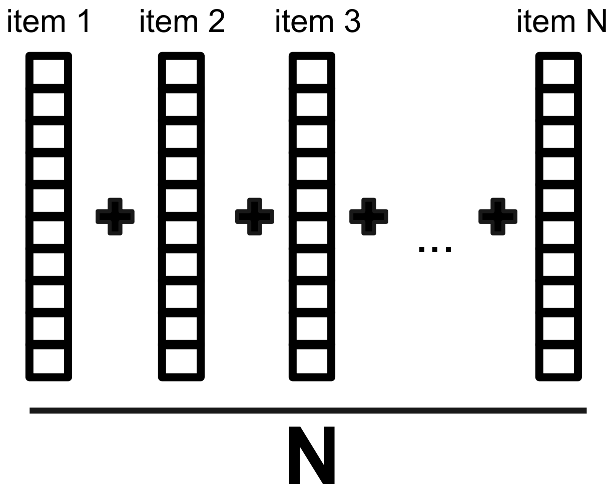
- Each item is represented as high-dimensional boolean or real valued vector

  Die Hard: [0,0,0,1, ….,1,0,0]

- Boolean vectors indicate presence absence of feature (e.g. is Bruce Willis in the movie?, is it action?, etc.)
- Real values can be used to indicate strength of feature, e.g. frequency of a tag, frequency of a word in a text

-> More info see Text Mining slides

Building User Profiles



- User profiles can be created by averaging vectors of items they have rated
- Ratings can be used to get a weighted average
- Each user profile is now also a vector with the same features

Example

movie	Director	Lead actor	Genre	Period	Alice Score
Die Hard	John McTiernan	Bruce Willis	action	80’s	**5**
The Hunt for Red October	John McTiernan	Sean Connery	action	90’s	**5**
Bridget Jones Diary	Sharon Maguire	Renée Zellweger	romance	00s	**1**
Four Weddings and a Funeral	Mike Newell	Hugh Grant	romance	90s	**0**

Alice’s profile:

(weighted) feature “action”:

(5*1+5*1+1*0+0*0)/11 = 0.9

(weighted) feature “romance”:

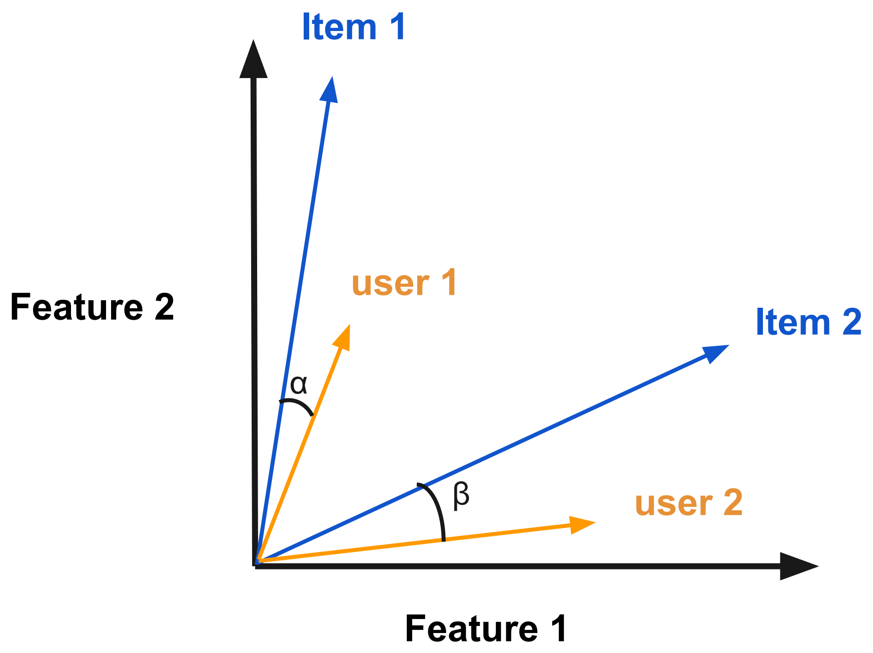
(5*0+5*0+1*1+0*1)/11 = 0.09

(weighted) feature “Bruce Willis”:

(5*1+5*0+1*0+0*0)/11 = 0.45

…

Similarity



- Items and users can now be represented as vectors in the same space
- Angle between vectors indicates how well they match
- cosine similarity :

sim(item1,user1) = cos(α)

Advantages

- Simple approach, easy to implement
- Explainable to users:
  - recommended because you liked *Die Hard*
  - recommended because you enjoyed *action movies*
- Case Based approaches are suitable for short term interactions (e.g. looking for a hotel, buying a camera, …)
- No need for input from other users to start making recommendations

Disadvantages

- Need description of every item (often labor intensive for large item sets)
- Items need good set of attributes to distinguish between items (difficult for some classes)
- Mainly finds similar items, can’t discover new things

Collaborative Filtering

- Find similarities between users or items based on the score matrix:
  - user/user: find users who give similar scores
  - item/item: find items that get similar scores
  - dimensionality reduction: learn features to find both similar users and similar items
- Does not need external information sources, only ratings

User - User CF: Idea

To get a prediction for rating of user *u* on item *i* P(u,i):

1. select a set of *n* users who have rated *i*
2. average their ratings on item i:

$$P(u,i) = \frac{1}{n}\sum_{k=1}^{n} r(k,i)$$

3. recommend items with highest predictions

User - User CF: Normalization

- One issue with this approach is that users might use different rating scales (i.e. one user might consider 6/10 good while another thinks 9/10 is good)
- This can be solved by normalizing the scores, i.e. do not look at the raw ratings, but rather at the difference between the item’s rating and the average rating given by that user:

$$P(u,i) = \bar{r}(u) + \frac{1}{n}\sum_{k=1}^{n}(r(k,i) - \bar{r}(k))$$

where $\bar{r}(k)$ is the average rating given by user k

User - User CF: weights

- To get better personalized predictions, ratings can be weighted according to how similar user *k* is to user *u:*

$$P(u,i) = \frac{\sum_{k=1}^{n} r(k,i) * w(k,u)}{\sum_{k=1}^{n} w(k,u)}$$

where *w(k,u)* is a similarity between users *k* and *u*

- Multiple possibilities to select neighborhood of users:
  - n users with highest *w(k,u)*
  - all users with *w(k,u)* over a given threshold

Calculating user similarity

- User similarity is calculated based on items both users have rated
- Multiple approaches are possible. Simplest is to use the Pearson Correlation Coefficient:

$$w(k,u) = \frac{\sum_{j=1}^{m}(r(k,j) - \bar{r}(k))(r(u,j) - \bar{r}(u))}{\sigma(k)\sigma(u)}$$

User-User CF

Combining weighting and normalization we get:

$$P(u,i) = \bar{r}(u) + \frac{\sum_{k=1}^{n} (r(k,i) - \bar{r}(k)) * w(k,u)}{\sum_{k=1}^{n} |w(k,u)|}$$

where the sum is taken over *n* neighboring users

Example

movie / user	Alice	Bob	Carol	Dave
The Dark Knight	?	0	1	4
Die Hard	5	0	?	?
The Hunt for Red October	5	?	0	5
Love Actually	?	5	4	0
Bridget Jones Diary	1	5	?	?
Four Weddings and a Funeral	0	?	5	1

$\bar{r}$:

Alice	Bob	Carol	Dave
2.75	2.5	2.5	2.5

w(k,Carol):

Alice	Bob	Dave
-1	1	-0.9

P(Carol,Die Hard):
2.5+1/2(-1(5-2.75)+1(0-2.5))= 0.1

P(Carol,Bridget Jones):
2.5+1/2(-1(1-2.75)+1(5-2.5))= 4.6

Weaknesses

- Not scalable, calculating correlations is costly
- Issues with sparsity: with large item sets or few customer ratings, no predictions might be possible.
- technical issues: out of bound predictions, Pearson coefficient not always suitable

Item-Item CF

- Calculate rating based similarities between *items*
- Often more suitable for cases where

  #users >> #items

  example: 1K items, 100K users, 50 ratings/user

  -> ~5000 ratings / item

  -> easier to find similar items than similar users

Item-Item CF (2)

We can calculate predictions as before:

$$P(u,i) = \bar{r}(i) + \frac{\sum_{j=1}^{n} (r(u,j) - \bar{r}(j)) * w(i,j)}{\sum_{j=1}^{n} |w(i,j)|}$$

but now averages are taken over item ratings, *w(i,j)* calculates similarity between items *i* and *j*

and the sum is taken over neighboring *items*

Advantages

- Item similarities tend to be more stable than user similarities (a single user adding ratings, can change user similarities a lot, but typically won’t change item similarities)
- Stable similarities allow offline computation
- Number of similarities to compute can often be limited to fixed value (e.g only consider 1000 most similar items)
- Precomputed similarities can be used to generate immediate recommendations for items user is looking at

Dimensionality Reduction CF

Instead of directly comparing users or items, we can first calculate features from the ratings that describe users and items.

These features can then be used to make predictions and compare items or users.

Learning user preferences

If we had features describing the movie content, we could learn user weights describing their preferences:

movie / user	**x1 (action)**	**x2 (romance)**
The Dark Knight	0.8	0.2
Die Hard	0.9	0.1
The Hunt for Red October	1.0	0
Love Actually	0	1.0
Bridget Jones Diary	0.1	0.9
Four Weddings and a Funeral	0.2	0.8

Make predictions for a user by learning user weights for each feature:

$P(u,i) = \boldsymbol{\theta_u 1 * x_i 1 + \theta_u 2 * x_i 2}$

How much user likes action (x1)

How much user likes romance (x2)

Preferences optimization

We can now learn the preferences of users by minimizing following error:

$$\min_{\theta_u} \sum_{i=1}^{m} (\theta_u^T x_i - r(u, i))^2$$

This problem can be solved by gradient descent

Example

The calculated features for Bob are:

user	θ1 (action)	θ2 (romance)
Bob	-0.8846	5.2692

This gives following predictions:

P(**The Hunt for Red October**,Bob)= 1*(-0.8846)+0*(5.2692) = -0.8846

P(**Four Weddings and a funeral**,Bob)= 0.2*(-0.8846)+0.8*(5.2692) = 4.03844

Learning movie features

If we have the features θ describing user preferences, we can move in the other direction and calculate features x_i for a movie *i* by minimizing following objective:

$$\min_{x_i} \sum_{u=1}^{n} (\theta_u^T x_i - r(u,i))^2$$

Example

user	**θ1 (action)**	**θ2 (romance)**
Alice	0.9	0.1
Bob	0	1.0
Carol	0.2	0.8
Dave	0.9	0.2

Results in following features:

Movie	**x1 (action)**	**x2 (romance)**
The Hunt for Red October	5.7939	-1.4368

P(**The Hunt for Red October**,Bob)= 5.7939*(0)-1.4368*(1.0) = -1.4368

Learning all features

It is possible to learn user preferences and movie features at the same time. Idea:

- randomly initialize user preferences θ
- use θ to compute movie features X
- use X to recompute θ
- repeat

Perform cross-validation to find a good number of features

Rating Decomposition

Learning user & movie features decomposes the ratings matrix into the product of a user feature matrix θ and a movie feature matrix X:

$$\underset{(n\_movies \times n\_users)}{R} = \underset{(n\_users \times k)}{\theta} * \underset{(n\_movies \times k)^T}{X^T}$$

Example

movie / user	Alice	Bob	Carol	Dave
The Dark Knight	?	-1.6	-0.6	2.3
Die Hard	2.5	-2.5	?	?
The Hunt for Red October	1.6	?	-3.3	1.6
Love Actually	?	2.0	1.0	-3.0
Bridget Jones Diary	-2.0	2.0	?	?
Four Weddings and a Funeral	-2.0	?	3.0	-1.0

First normalize ratings matrix, so each movie has mean rating 0

Example

movie	x1	x2	x3
The Dark Knight	0.7	-0.2	-0.9
Die Hard	1.2	0.4	-0.7
The Hunt for Red October	0.5	1.2	-0.8
Love Actually	-0.8	0.2	1.2
Bridget Jones Diary	-1.0	-0.3	0.6
Four Weddings and a Funeral	-0.8	-1.2	0.4

user	θ1	θ2	θ3
Alice	1.3	0.5	-0.6
Bob	-1.2	-0.2	0.9
Carol	-0.7	-1.7	0.7
Dave	0.9	-0.2	-1.5

We now have features describing each movie and each user preference

Making Predictions

We can now predict ratings by multiplying movie feature vectors with user preference vectors (and adding the mean we used to normalize):

- Bob’s score for *The Hunt for Red October:*

  $\bar{r}$ *(Hunt)* + $X_{hunt} * \theta_{Bob}$ ~= 1.7

- Bob’s score for *Four Weddings and a Funeral:*

  $\bar{r}$ *(4wedding)*+ $X_{4weddings} * \theta_{Bob}$ ~= 3.6

- Bob’s score for *Love Actually:*

  $\bar{r}$*(LoveActually)*+ $X_{love\_actually} * \theta_{Bob}$ ~= 5

Finding Similar movies

We can now also find similar movies by directly comparing their features:

$$\textit{difference}(\textit{movie1},\textit{movie2}) = ||X_{movie1} - X_{movie2}||$$

for *Die Hard*:

(lower is more similar)

The Dark Knight	0.8
The Hunt for Red October	1.1
Love Actually	2.8
Bridget Jones Diary	2.7
Four Weddings and a Funeral	2.8

Summary

- Dimensionality reduction based CF no longer needs to directly compare user/item ratings
- Results in features for describing both users and items
- Allows us to compare user and items based on features (even if they don’t have ratings in common)
- Downside: the features are not easy to interpret or explain to users

Some Other issues

- Most approaches suffer from cold start (need initial ratings to get started)
- Binary Ratings (like/dislike)
- Indirect ratings (purchases, page views)
- How to add context information (mobile recommendations)