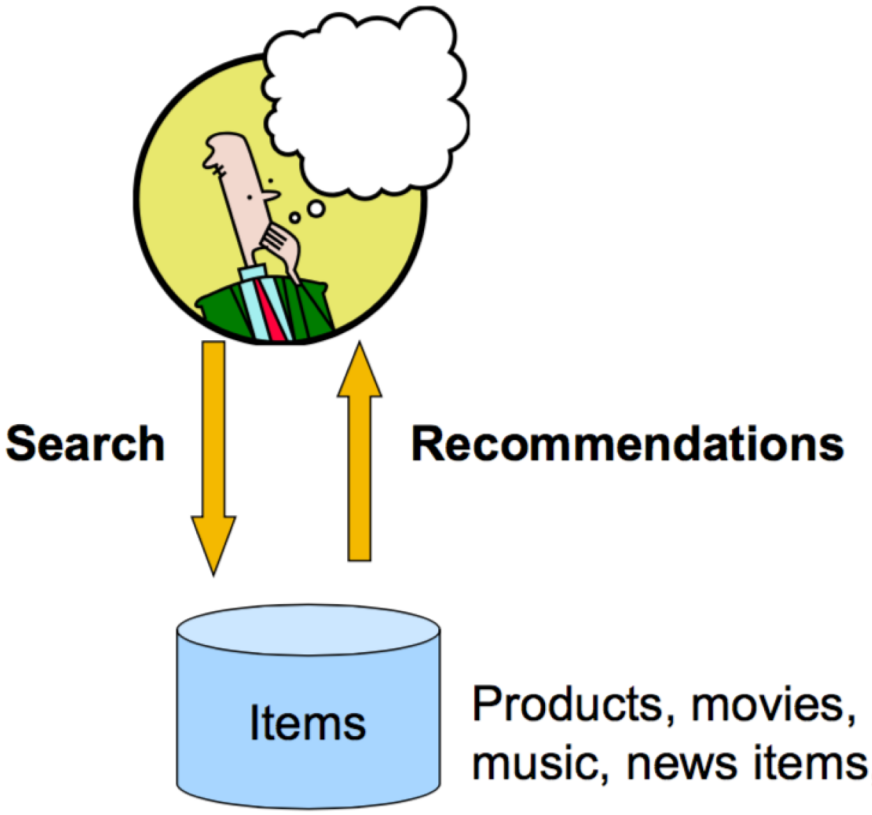
Making Recommendations: Content Filtering Collaborative Filtering Latent Factors

Jay Urbain, Ph.D.
Electrical Engineering and Computer Science Department
Milwaukee School of Engineering

Credits: See last page with references

Recommendations



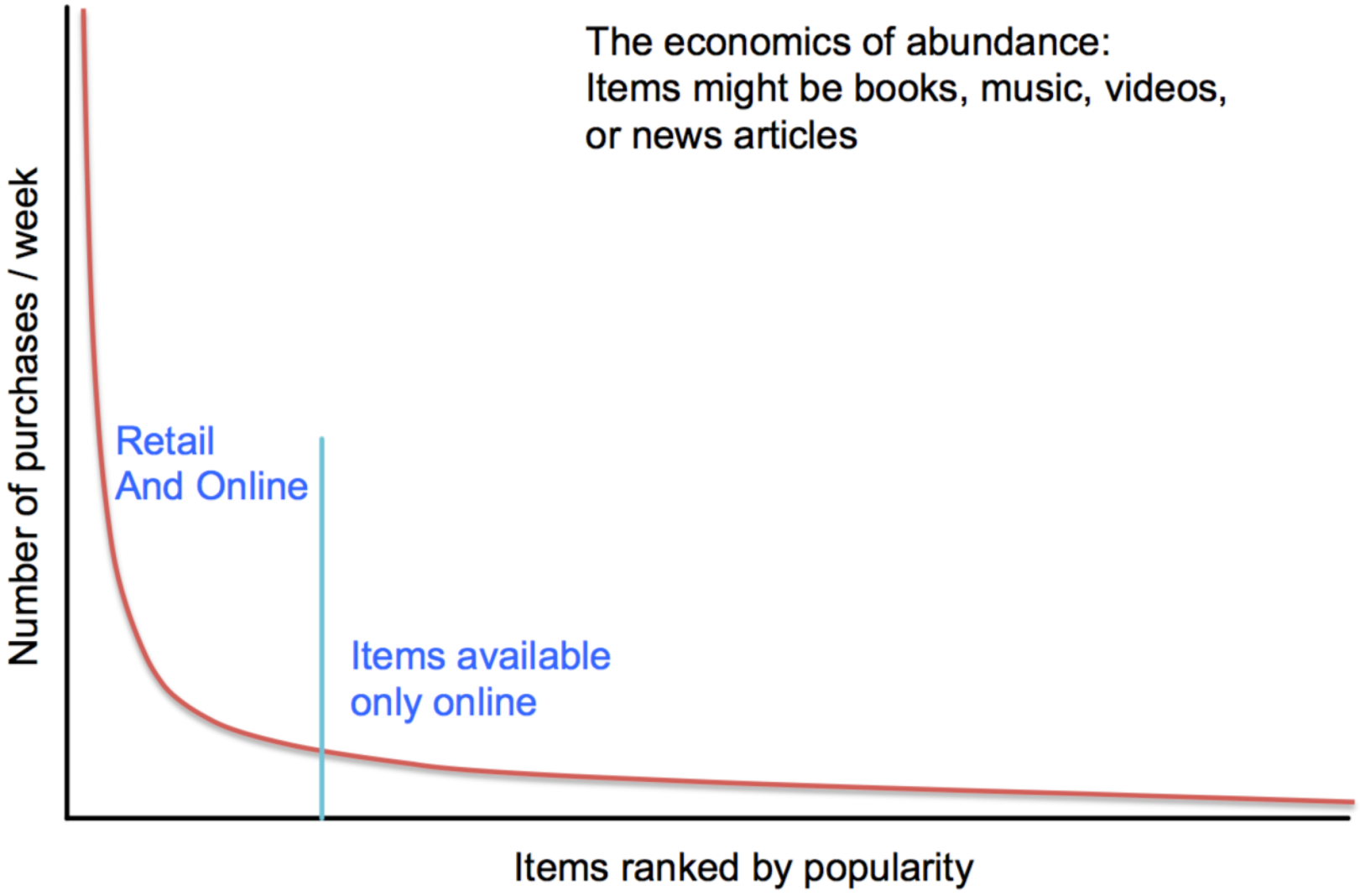
Products, movies, music, news items, …

amazon.com
PANDORA
StumbleUpon
del.icio.us
NETFLIX
movielens
helping you find the *right* movies
last·fm
the social music revolution
Google News
YouTube
XBOX LIVE

From scarcity to abundance

- Shelf space is a scarce commodity for traditional retailers
  - Also: TV networks, movie theaters,...
- The web enables near zero-cost dissemination of information about products
  - From scarcity to abundance
  - Gives rise to the “Long Tail” phenomenon (Wired)

Long Tail


The economics of abundance:
Items might be books, music, videos,
or news articles
Number of purchases / week
Retail
And Online
Items available
only online
Items ranked by popularity

Long Tail

- More choice necessitates better filters
- Recommendation engines
- How **Into Thin Air** made **Touching the Void** a bestseller (http://www.wired.com/wired/archive/12.10/tail.html)
- Examples
  - Books, movies, music, new car sales, medical treatments
  - People (friend recommendations on Facebook, LinkedIn, and Twitter)

Types of recommendations

- Editorial and hand curated
  - List of favorites
  - Lists of “essential” items
- Simple aggregates
  - Top 10, Most Popular, Recent Uploads
- **Tailored to individual users**
  - Amazon, Netflix, Pandora ...
  - Our focus here

Recommendation Model

- ***C*** = set of **Customers**
- ***S*** = set of **Items**
- **Utility function *u*: *C* × *S* → *R***
  - ***R*** = set of ratings
  - ***R*** is a totally ordered set, e.g., **0-5** stars, real number in **[0,1]**

Utility Matrix

	Avatar	LOTR	Matrix	Pirates
Alice	1		0.2	
Bob		0.5		0.3
Carol	0.2		1	
David				0.4

Problems

- Gathering “known” ratings for matrix
  - How to collect the data in the utility matrix
- Extrapolate unknown ratings from the known ones
  - Mainly interested in high unknown ratings
  - Not typically interested in knowing what you don’t like but what you like
- Evaluating extrapolation methods
  - How to measure success/performance of recommendation methods

Gathering Ratings

- Explicit
  - Ask people to rate items
  - Doesn’t scale: only a small fraction of users leave ratings and reviews
- Implicit
  - Learn ratings from user actions
  - E.g., purchase implies high rating
  - What about low ratings?

Extrapolating Utilities

- Key problem: matrix ***U*** is **sparse**
- Most people have not rated most items
- Cold start:
  - New items have no ratings
  - New users have no history
- Three approaches to recommender systems
  1. Content-based
  2. Collaborative
  3. Latent factor based

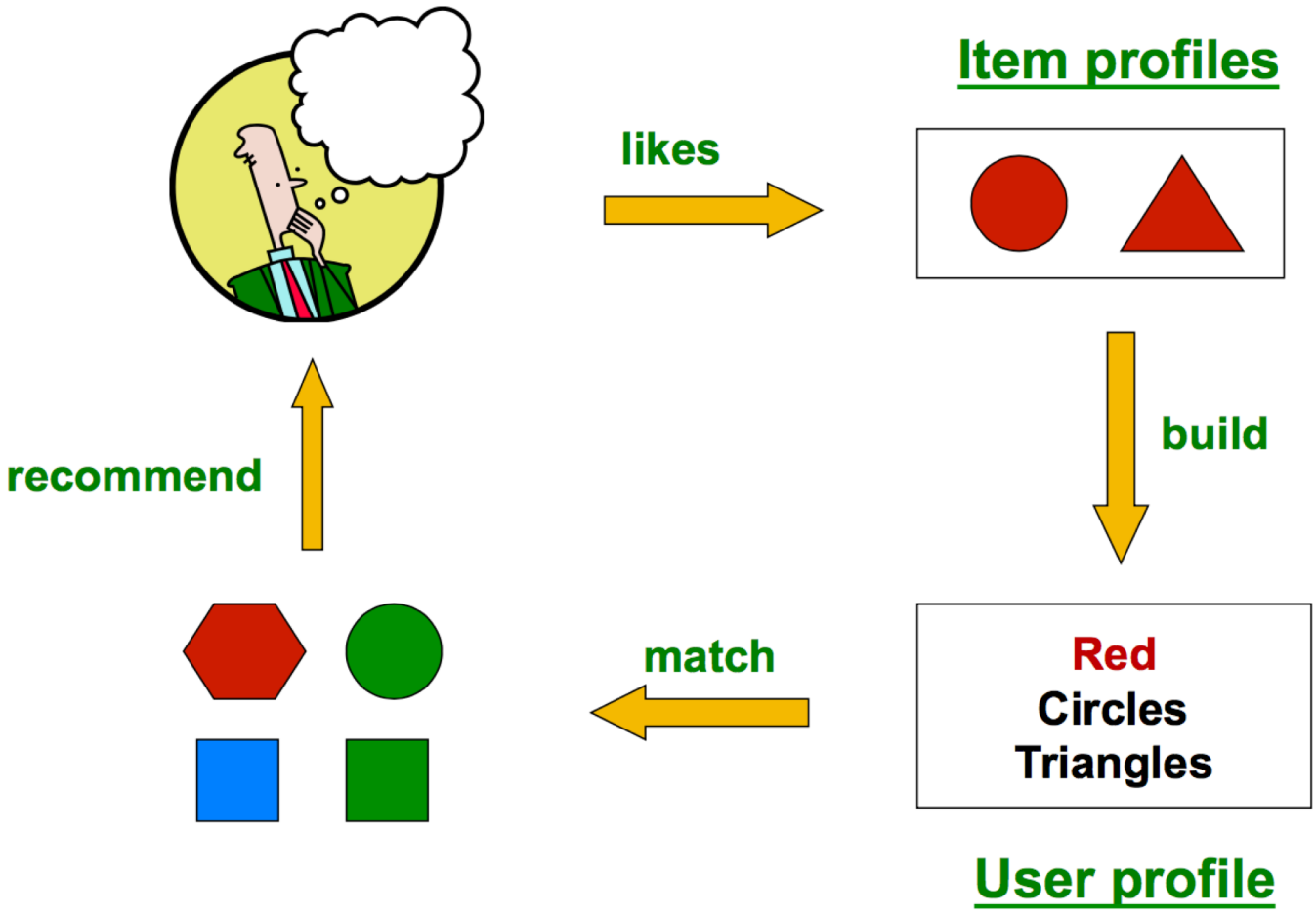
Content-based

Main idea: Recommend items to customer ***x*** similar to previous items rated highly by ***x***

Examples:

- Movies
  - Same actor(s), director, genre, ...
- Websites, blogs, news
  - Articles with “similar” content
- People
  - Recommend people with many common friends

Build item profiles, make recommendations from profile



Item profiles

- For each item, create an **item profile**
- Profile is a set of features
  - **Movies:** author, title, actor, director,...
  - **Images, videos:** metadata and tags
  - **People:** Set of friends
- Think of the item profile as a feature vector
  - One attribute per feature per item (e.g., each actor, director,...)
  - Vector might be boolean or real-valued or mix
- Predict how similar any item is to any other item

User profiles

- User has rated items with profiles $\boldsymbol{i_1,\ldots,i_n}$
- Simple: (weighted) average of rated **item profiles**
- Variant: Normalize weights using average rating of user
- More sophisticated aggregations possible – learn feature weights.

Example 1: Boolean Utility Matrix

- Items are movies, only feature is “Actor”
  - Item profile: vector with 0 or 1 for each Actor
- Suppose user x has watched 5 movies
  - 2 movies featuring actor A
  - 3 movies featuring actor B

- User profile = mean of item profiles
  - Feature A’s weight = 2/5 = 0.4
  - Feature B’s weight = 3/5 = 0.6

Example 2: Star Ratings

- Same example, 1-5 star ratings
  - Actor A's movies rated 3 and 5
  - Actor B's movies rated 1, 2 and 4
- Useful step: Normalize ratings by subtracting user's mean rating (mean=3)
  - Actor A's normalized ratings = 0, +2
  - Profile weight = (0 + 2)/2 = 1
  - Actor B's normalized ratings = -2, -1, +1
  - Profile weight = -2/3

Making Predictions

- User profile **x,** Item profile **I**
- Estimate U(**x**,**i**) = cos(θ) = (**x** . **i**)/(|**x**||**i**|)
- Or other similarity measurement

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Pros: Content-based approach

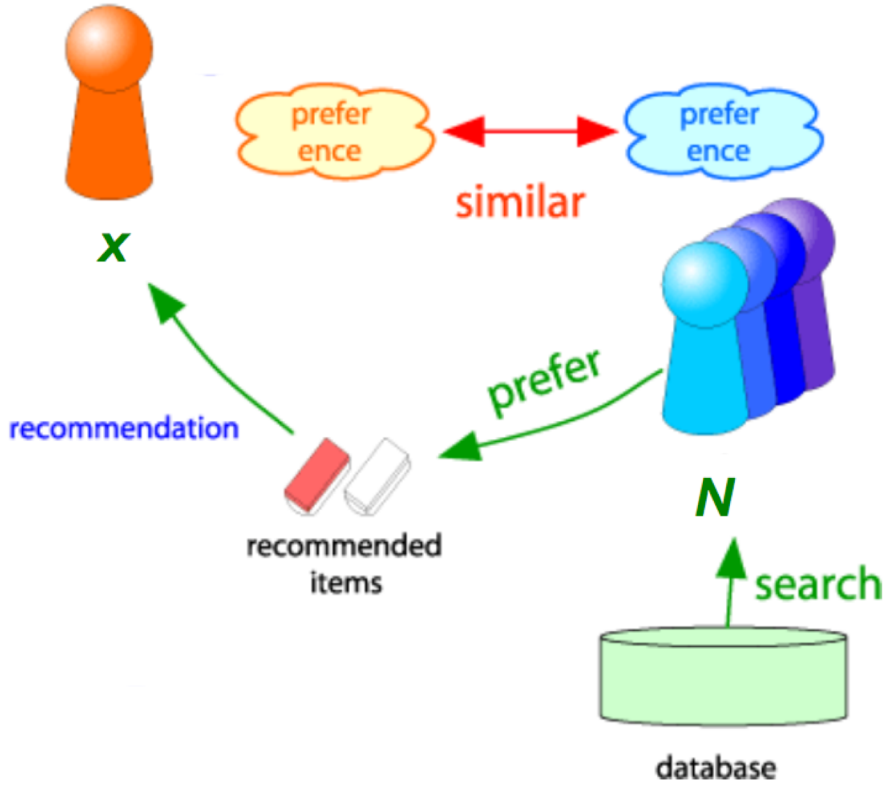
- No need for data on other users
- Able to recommend to users with unique tastes
- Able to recommend new & unpopular items
- No first-rater problem
- Explanations for recommended items
- Content features that caused an item to be recommended

Cons: Content-based approach

- Finding the appropriate features is hard
  - E.g., images, movies, music
- Overspecialization
  - Never recommends items outside user's content profile
  - People might have multiple interests
  - Unable to exploit quality judgments of other users
- Cold-start problem for new users
  - How to build a user profile?

Collaborative Filtering

- Consider user ***x***
- Find set ***N*** of other users whose ratings are "**similar**" to ***x***'s ratings
- Estimate ***x***'s ratings based on ratings of users in ***N***



Similar Users

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- Consider users ***x*** and ***y*** with rating vectors $\boldsymbol{r}_x$ and $\boldsymbol{r}_y$
- We need a similarity metric sim(***x***, ***y***)
- Capture intuition that sim(A,B) > sim(A,C)

Jaccard Similarity

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- $sim(A,B) = | r_A \cap r_B | / | r_A \cup r_B |$
- sim(A,B) = 1/5; sim(A,C) = 2/4
  - sim(A,B) < sim(A,C)
- Problem: Ignores rating values!

Cosine Similarity

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- $sim(A,B) = cos(r_A, r_B)$
- sim(A,B) = 0.38, sim(A,C) = 0.32
  - sim(A,B) < sim(A,C), but not by much
- Problem: treats missing ratings as negative

Centered Cosine – Pearson Correlation Coefficient

- Normalize ratings by subtracting row mean

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	$2/3$			$5/3$	$-7/3$		
B	$1/3$	$1/3$	$-2/3$				
C				$-5/3$	$1/3$	$4/3$	
D		0					0

Centered Cosine

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	$2/3$			$5/3$	$-7/3$		
B	$1/3$	$1/3$	$-2/3$				
C				$-5/3$	$1/3$	$4/3$	
D		0					0

- sim(A,B) = cos(r_A, r_B) = 0.09; sim(A,C) = -0.56
  - sim(A,B) > sim(A,C)
- Captures intuition better
  - Missing ratings treated as "average"
  - Handles "tough raters" and "easy raters"
- Also known as Pearson Correlation

Rating Predictions

- Let r_x be the vector of user x's ratings
- Let N be the set of k users most similar to x who have also rated item i
- Prediction for user x and item i

- Option 1: $r_{xi} = 1/k \sum_{y \in N} r_{yi}$
- Option 2: $r_{xi} = \sum_{y \in N} s_{xy} r_{yi} / \sum_{y \in N} s_{xy}$

where $s_{xy} = \text{sim}(x,y)$

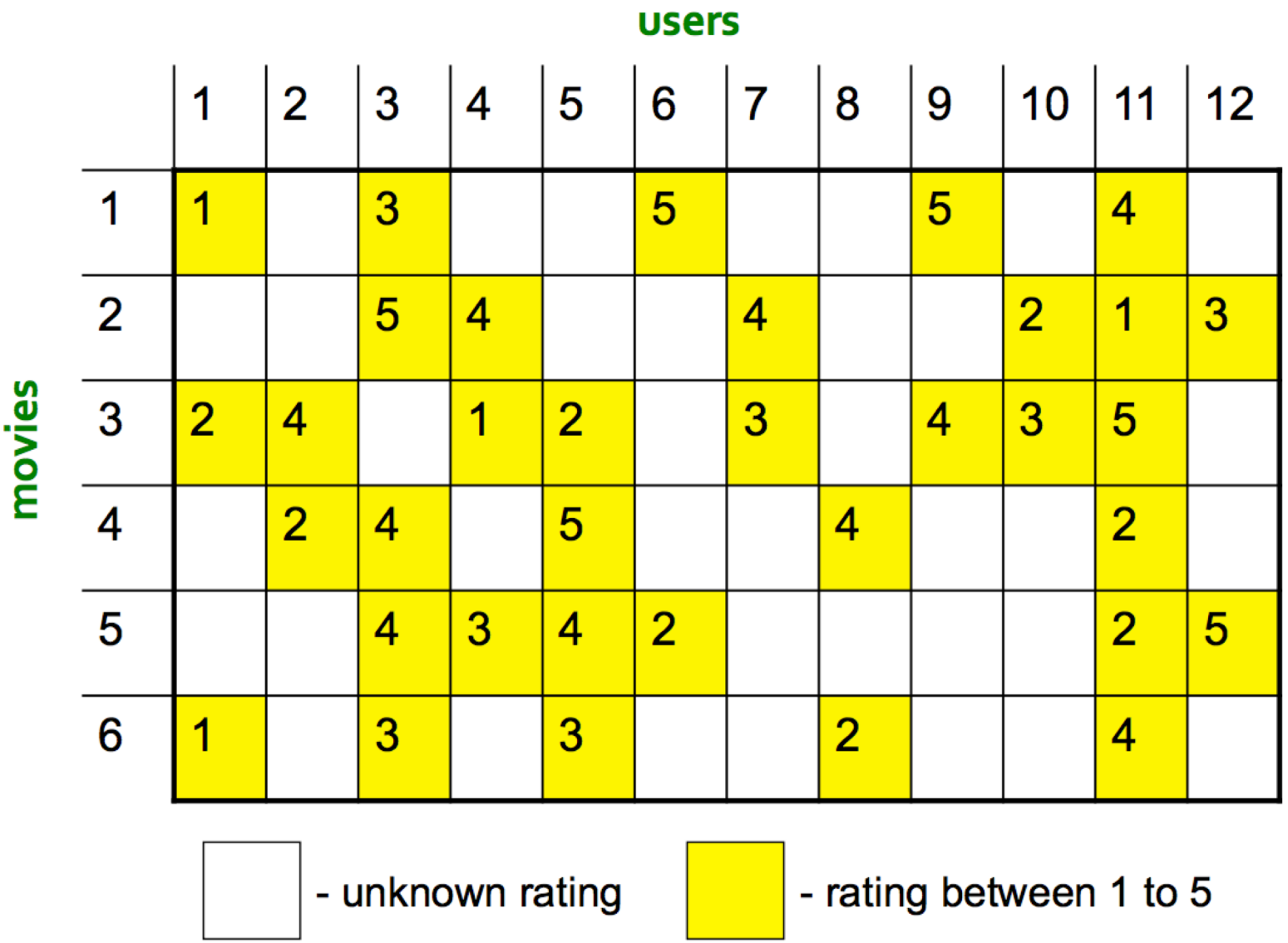
Item-Item Collaborative Filtering

- **So far: User-user collaborative filtering**
- **Another view: Item-item**
  - For item ***i***, find other similar items
  - Estimate rating for item ***i*** based on ratings for similar items
  - Can use same similarity metrics and prediction functions as in user-user model

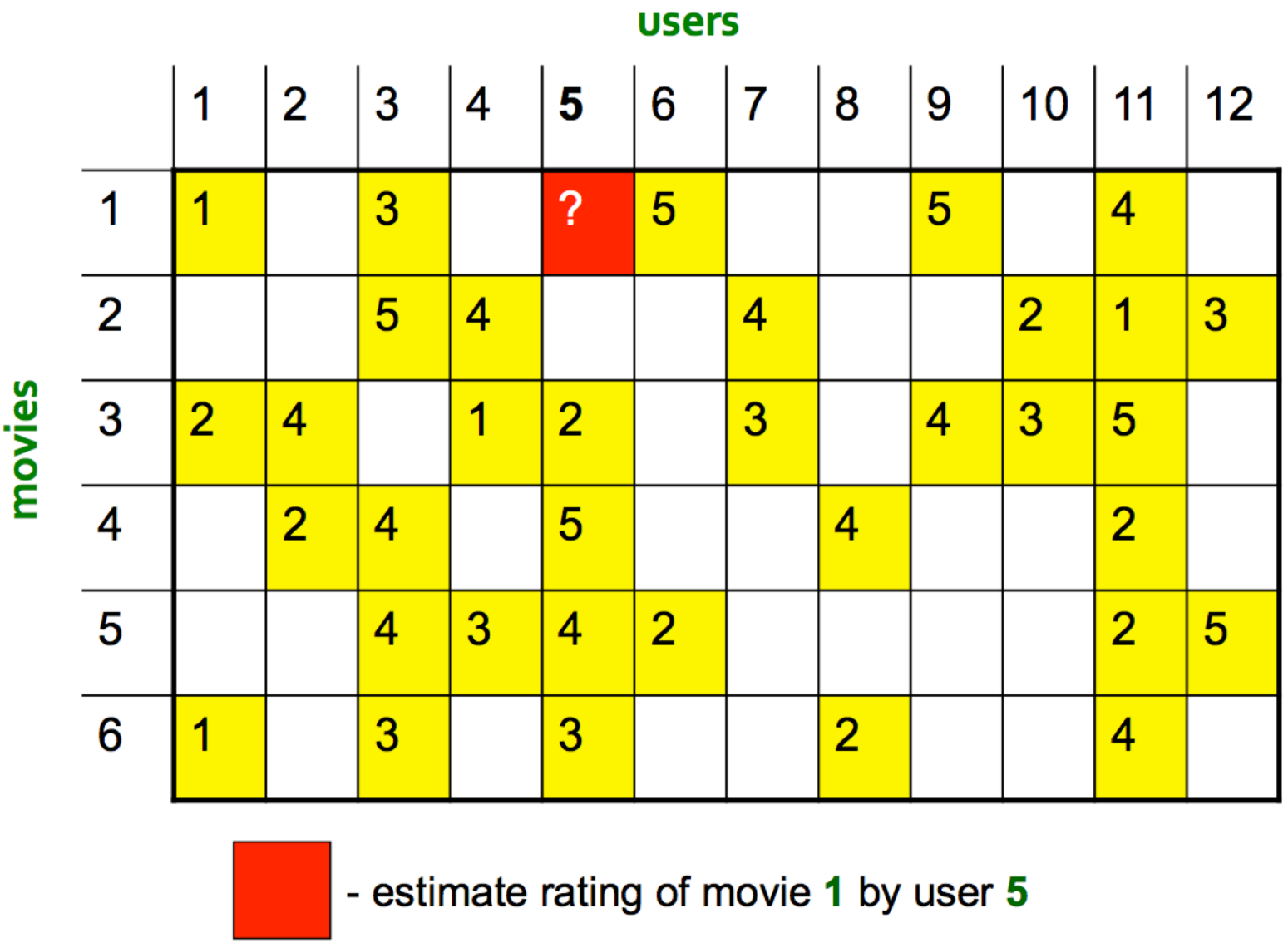
$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

s_{ij}… similarity of items ***i*** and ***j***
r_{xj}…rating of user ***x*** on item ***j***
$N(i;x)$… set items rated by ***x*** similar to ***i***

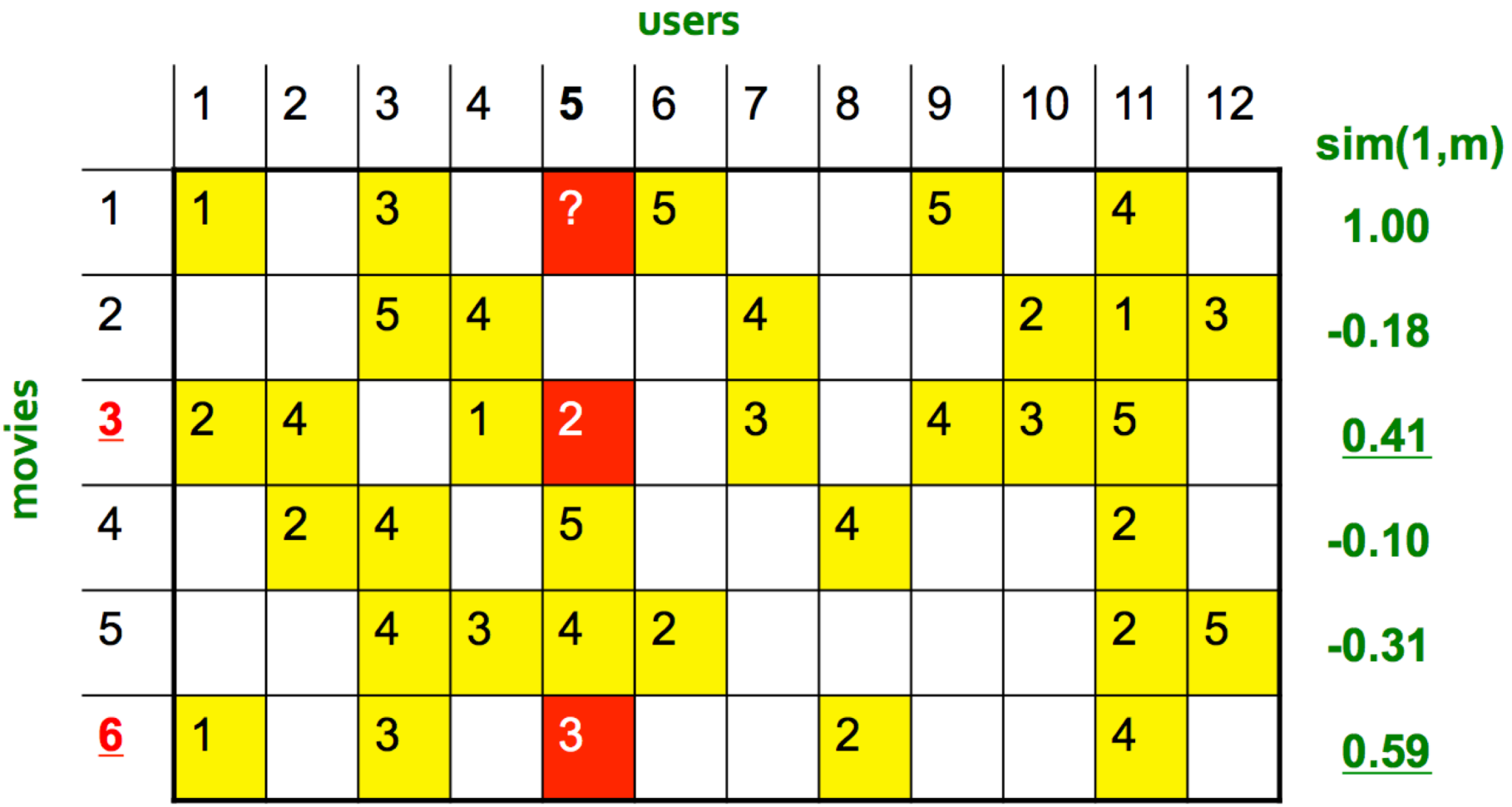
Item-Item CF (n=2)



Item-Item CF (n=2)



Item-Item CF (n=2)



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

Here we use Pearson correlation as similarity:

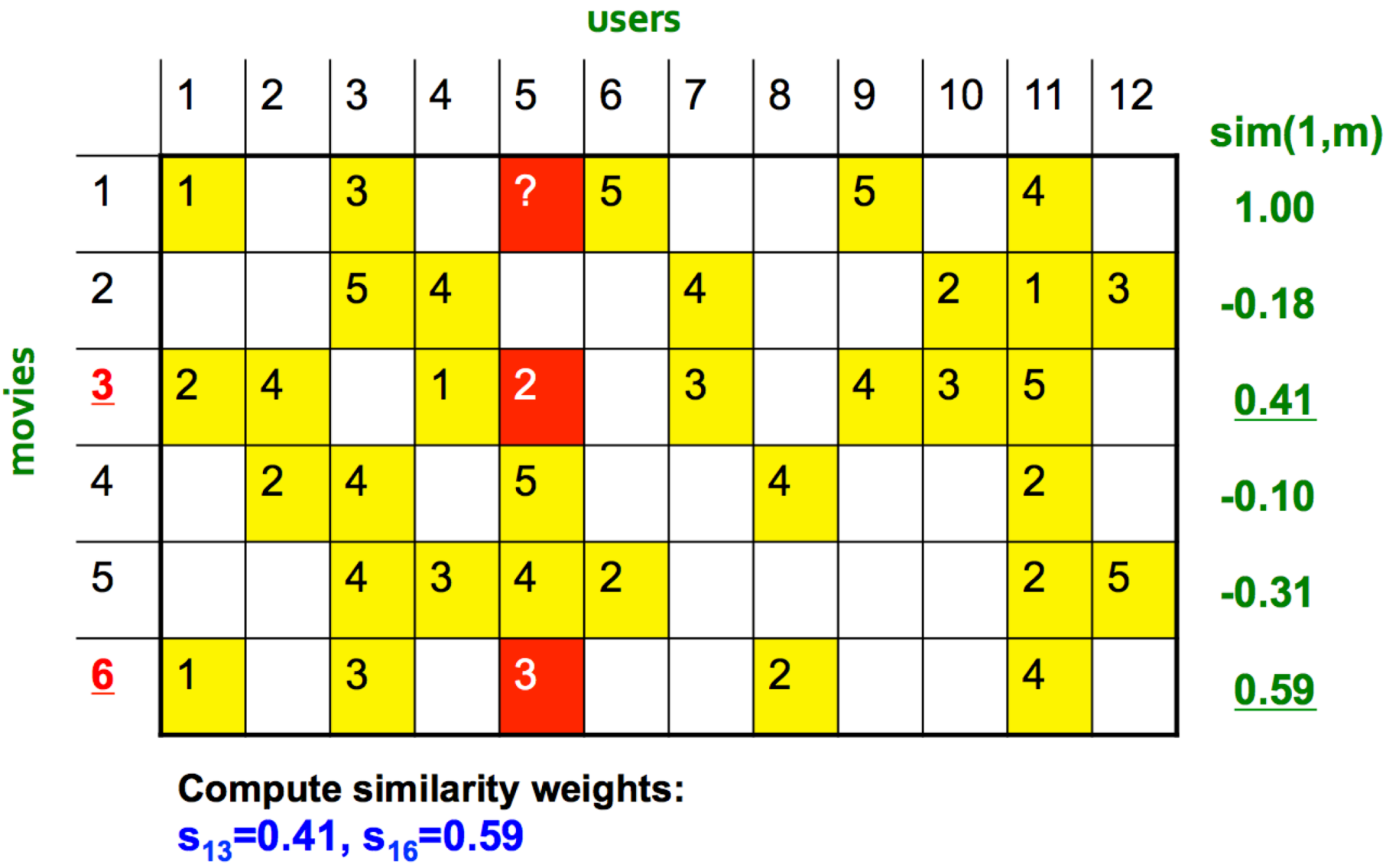
1) Subtract mean rating m_i from each movie ***i***

$m_1 = (1+3+5+5+4)/5 = 3.6$

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

2) Compute cosine similarities between rows

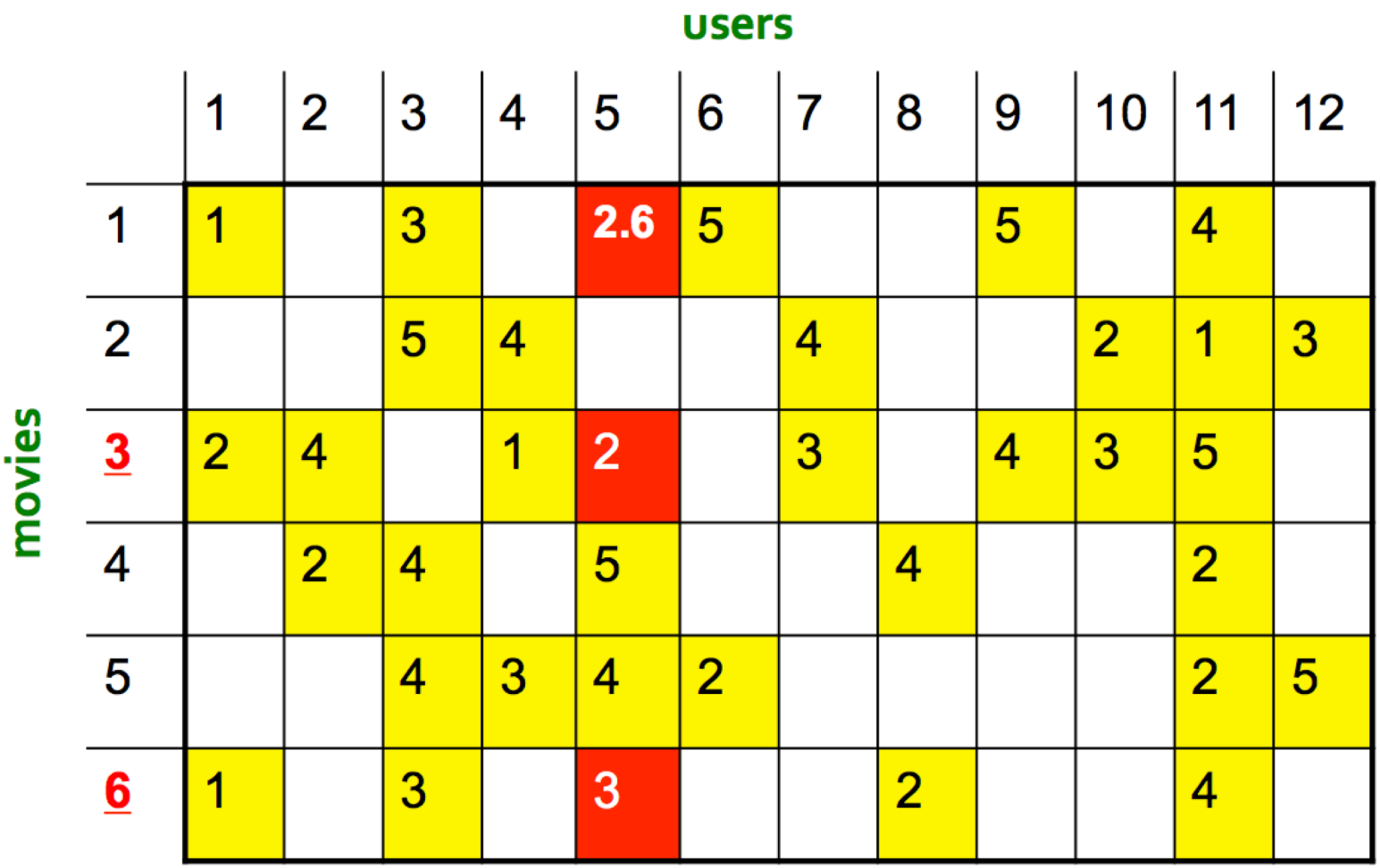
Item-Item CF (n=2)



Compute similarity weights:

s_{13}=0.41, s_{16}=0.59

Item-Item CF (n=2)



Predict by taking weighted average:

$r_{15} = (0.41*2 + 0.59*3) / (0.41+0.59) = 2.6$

Item-item vs. User-user

- In theory, user-user and item-item are dual approaches
- In practice, item-item outperforms user-user in many use cases
- Items are “simpler” than users
  - Items belong to a small set of “genres”, users have varied tastes
  - Item Similarity is more meaningful than User Similarity

Modern Recommendation System

- **Multi-scale modeling of the data:**
  Combine top level, "regional" modeling of the data, with a refined, local view:
  - **Global:**
    - Overall deviations of users/movies
  - **Factorization:**
    - Addressing "regional" effects
  - **Collaborative filtering:**
    - Extract local patterns

Modeling Local & Global Effects

- **Global:**
  - Mean movie rating: **3.7 stars**
  - *The Sixth Sense* is **0.5** stars above avg.
  - Joe rates **0.2** stars below avg.

  ⇒ Baseline estimation:

  ***Joe* will rate *The Sixth Sense* 4 stars**
- **Local neighborhood (CF/NN):**
  - *Joe* didn't like related movie *Signs*
  - **⇒ Final estimate:**

    ***Joe* will rate *The Sixth Sense* 3.8 stars**

Recap: CF

- **Earliest and most popular collaborative filtering method**
- Derive unknown ratings from those of "**similar**" movies (item-item variant)
- Define **similarity measure s_{ij}** of items ***i*** and ***j***
- Select ***k***-nearest neighbors, compute the rating
  - ***N(i; x):*** items most similar to ***i*** that were rated by ***x***

$$\hat{r}_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

s_{ij} similarity of items ***i*** and ***j***
r_{uj} rating of user ***x*** on item ***j***
N(i;x) set of items similar to item ***i*** that were rated by ***x***

Modeling Local & Global Effects

- **In practice we get better estimates if we model deviations:**

$$\hat{r}_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}}$$

baseline estimate for r_{xi}

$$\boldsymbol{b_{xi} = \mu + b_x + b_i}$$

$\boldsymbol{\mu}$ = overall mean rating
$\boldsymbol{b_x}$ = rating deviation of user $\boldsymbol{x}$
= (*avg. rating of* user $\boldsymbol{x}$) – $\boldsymbol{\mu}$
$\boldsymbol{b_i}$ = (*avg. rating of* movie $\boldsymbol{i}$) – $\boldsymbol{\mu}$

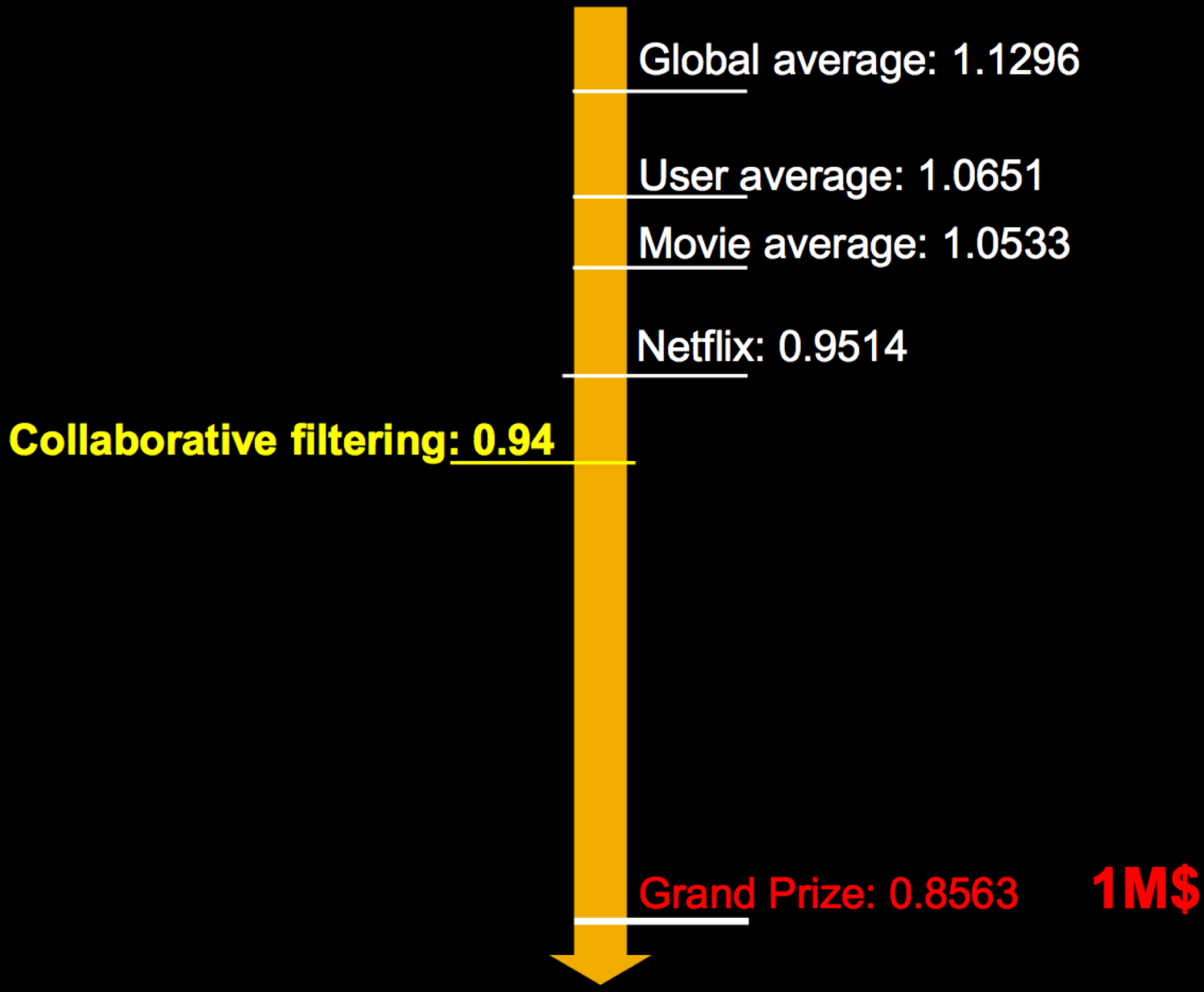
Problems/Issues:

1) Similarity measures are "arbitrary"

2) Pairwise similarities neglect interdependencies among users

3) Taking a weighted average can be restricting

RMSE of Various Methods



STOP

What does it mean to learn?

- Improve our performance by interacting with our environment.
  - Build model of the world we interact with to improve future performance – *prediction*.
- *Confabulation* - human brain is constantly deciding among multiple competing predications.

- *In what ways do we interact with our environment to learn?*
  - *Read*
  - *Watch*
  - *Hear*
  - *???*

Learning

- We often learn by *interacting with each other.*

- **Common insight**: personal tastes are *correlated*.
  - If Alice and Bob both like *X,* and Alice likes *Y* then Bob is more likely to like *Y*.
  - Especially (perhaps) if Bob knows Alice. Or Bob and Alice like other similar things.

Collective Intelligence

- Can we model these interactions? ***Sure***!
  - Google
  - Amazon
  - Netflix
  - Pandora
  - Futures markets
- All of these organizations combine the behavior, preferences, or ideas of a *group* of people to create novel insights.

Collective Intelligence

- Collecting answers from a *large* group of people lets you draw statistical conclusions about the group that no individual member would have known by themselves.

- Building new conclusions from independent contributors is what *collective intelligence* is all about.

- *So how do we go about doing that?*
  - *Collaborative Filtering, Content Filtering, Collaborative + Content Filtering*

Content filtering

Collaborative Filtering

- What is the low-tech way to get recommends for products, movies , interesting web sites, or entertaining things to do?

Collaborative Filtering

- What is the low-tech way to get recommendations for products, movies , interesting web sites, or entertaining things to do?

- Collective Intelligence - *ask your friends!*

- Do some of your friends have better taste than others?
  - These friends are more likely to *influence* our decisions.

Collaborative filtering - recommend items based on similarity measures between users and/or items.

Collaborative Filtering

Lets say we want to use the combined "*wisdom*" of an extended group of friends to pick movies.

1) Collect preferences
   - Use the Web
   - Build a database of individual ratings on movies

username	title	rating	rating_norm
bellmangreent	Raiders of the lost ark	3	0.5
bellmangreent	2012	3	0.5
bellmangreent	Aeon Flux	4	0.75
bellmangreent	Avatar	5	1
bellmangreent	Batman Begins	5	1
bellmangreent	Bourne Identity	3	0.5
bellmangreent	Children of Men	5	1
bellmangreent	Die Hard	4	0.75
bellmangreent	Die Hard with a Ven...	4	0.75
bellmangreent	Die Harder	4	0.75
bellmangreent	District 9	5	1
bellmangreent	Donnie Darko	3	0.5
bellmangreent	Harry Potter and th...	3	0.5

Collaborative Filtering

2) Finding similar users

Need method for correlating choices:

- Euclidian Distance
- Pearson Correlation
- Singular Value Decomposition
- Clustering
- Classification

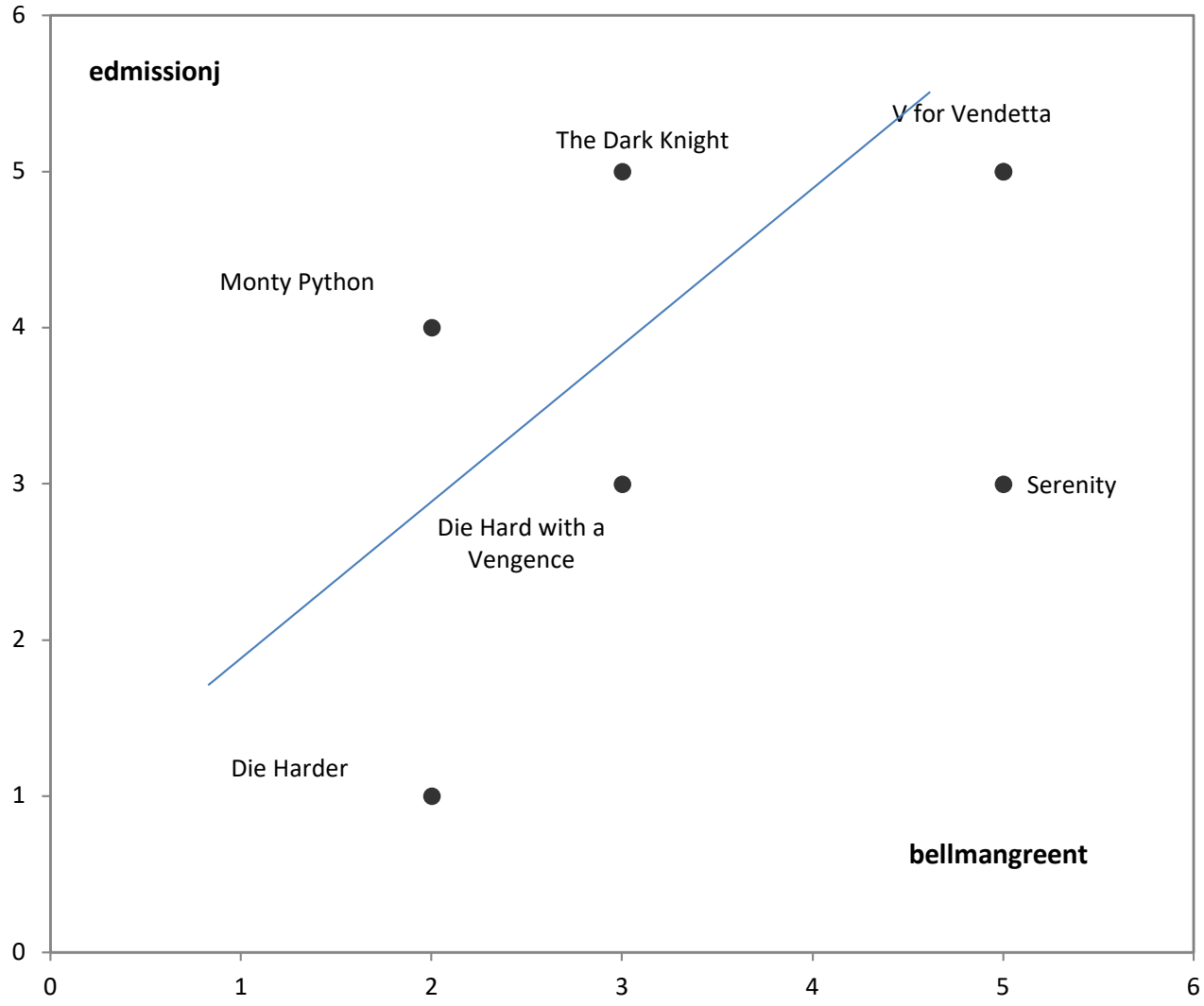
Compare Users

Select User 1: bellmangreent

Select User 2: edmissonj

User 1	User 2	Title	Rating 1	Rating 2
bellmangreent	edmissonj	Die Hard	4.0	4.0
bellmangreent	edmissonj	V for Vendetta	4.0	4.0
bellmangreent	edmissonj	Die Hard with a Vengence	4.0	4.0
bellmangreent	edmissonj	The Dark Knight	5.0	5.0
bellmangreent	edmissonj	Monty Python: The Holy Gr	4.0	4.0
bellmangreent	edmissonj	Die Harder	4.0	4.0
bellmangreent	edmissonj	Batman Begins	5.0	5.0
bellmangreent	edmissonj	Serenity	5.0	5.0
bellmangreent	edmissonj	Sherlock Holmes	4.0	4.0
bellmangreent	edmissonj	Iron Man	5.0	5.0
bellmangreent	edmissonj	Star Trek	5.0	4.0
bellmangreent	edmissonj	Matrix	5.0	4.0
bellmangreent	edmissonj	Avatar	5.0	4.0
bellmangreent	edmissonj	The Expendables	4.0	5.0
bellmangreent	edmissonj	District 9	5.0	4.0
bellmangreent	edmissonj	Bourne Identity	3.0	4.0
bellmangreent	edmissonj	The Hulk	3.0	4.0

Finding similar users – Euclidean



Collaborative Filtering

2) Finding similar users – Euclidian Distance
   1) Take the differences in each axis
   2) Square them & add them together
   3) Then take the square root of the sum

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \mathrm{d}(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

select sqrt(sum(pow(r1.rating_norm-r2.rating_norm,2))) as sc

Bellmangreent edmissonj 0.535898384862245

bellmangreent bellmangreent 1.00

Problems with Euclidean distance?

Collaborative Filtering

2) Finding similar users –

- Euclidian is just an *absolute* distance measure
- *Correlation* is a measure of how well two sets of data fit on a straight line, i.e., relative strength of association.
  - Corrects for "*grade inflation*"
  - Remove consistent differences

Collaborative Filtering

2) Finding similar users –

Pearson (product-moment) correlation coefficient

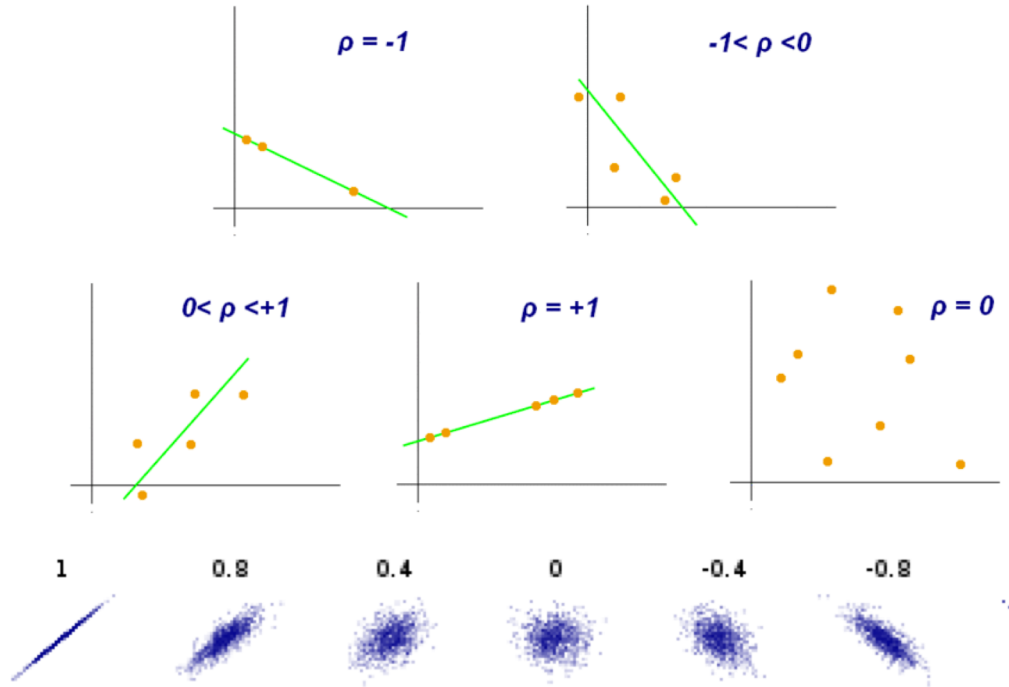
- Measure of the **correlation** (linear dependence) between two variables *X* and *Y*, given a value between +1 and −1 inclusive.
- Defined by the covariance of the two variables divided by the product of their standard deviations.
- Corresponds to the cosine of the angle θ between two regression lines.

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

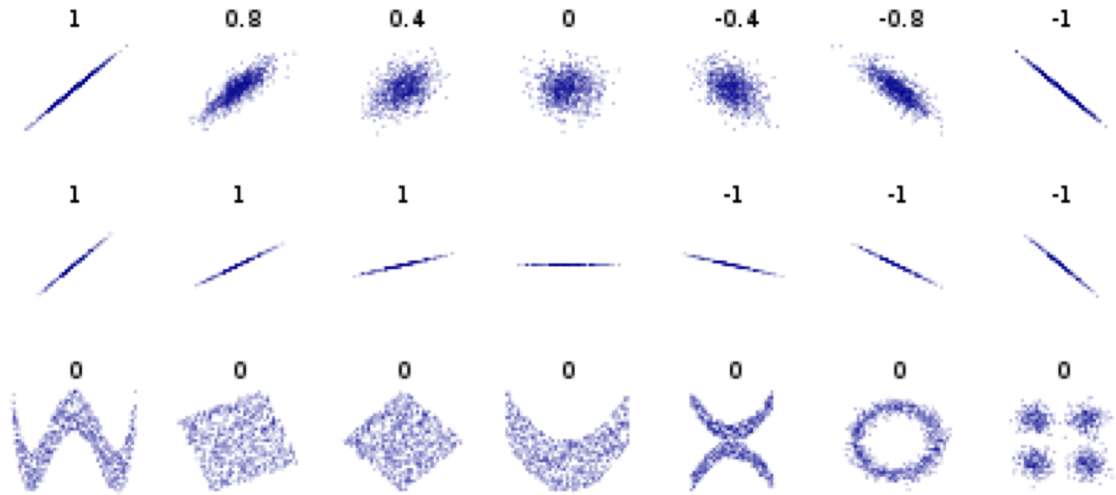
Collaborative Filtering

Pearson correlation coefficient

different values of correlation coefficient (ρ)



the correlation reflects the non-linearity and direction of a linear relationship (top row), but not the slope



Collaborative Filtering

2) Finding similar users – Pearson Correlation Coefficient

Mean of products formulation

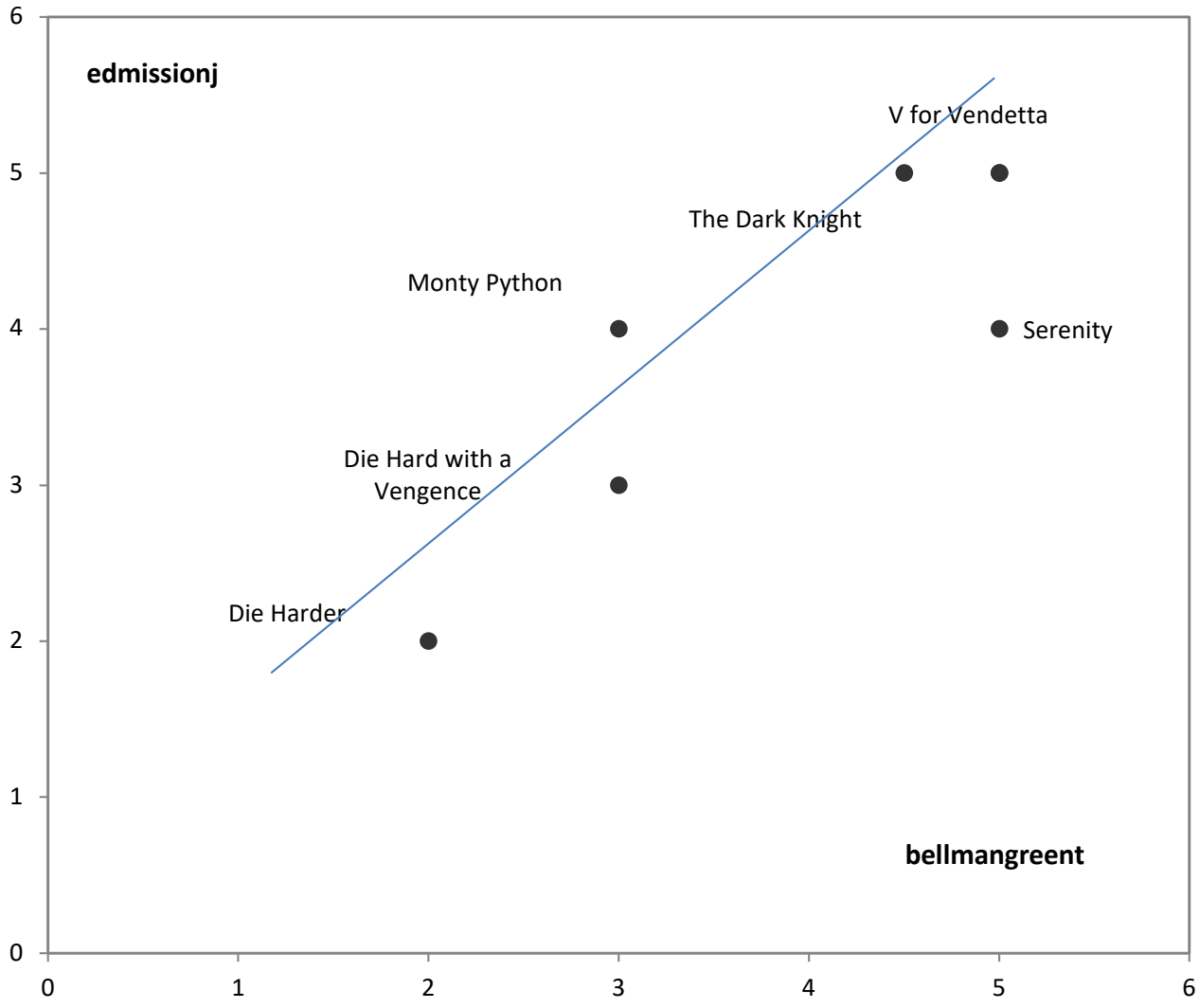
$$\frac{\sum_i (r_{1,i} * r_{2,i}) - \left(\frac{\sum_i (r_{1,i}) * \sum_i (r_{2,i})}{n}\right)}{\sqrt{\left(\sum_i (r_{1,i})^2 - \left(\frac{\sum_i (r_{1,i})^2}{n}\right)\right) * \left(\sum_i (r_{2,i})^2 - \left(\frac{\sum_i (r_{2,i})^2}{n}\right)\right)}}$$

STDEV r1

STDEV r2

```
( sum(r1.rating_norm*r2.rating_norm) -
      (sum(r1.rating_norm)*sum(r2.rating_norm))/count(*) ) /
sqrt( ( sum(pow(r1.rating_norm,2))-
      (pow(sum(r1.rating_norm),2)/count(*)) ) * (
      sum(pow(r2.rating_norm,2))-(pow(sum(r2.rating_norm),2)/count(*))
      ) ) as sc
```

Finding similar users – Normalized (Pearson Correlation Coefficient)



Collaborative Filtering

3. Ranking the reviewers

- Now that we can compare any two people.
- Score everyone against a given person and find the closest matches.
- Allows us to find the reviewers whose taste are most similar to any individual.

Collaborative Filtering

3. Ranking the reviewers

create a correlation coefficient similarity table

```
insert into sim
select u1.userid as userid1, u2.userid as userid2,
( sum(r1.rating_norm*r2.rating_norm) -
      (sum(r1.rating_norm)*sum(r2.rating_norm))/count(*) ) /
sqrt( ( sum(pow(r1.rating_norm,2))-(pow(sum(r1.rating_norm),2)/count(*)) ) * (
      sum(pow(r2.rating_norm,2))-(pow(sum(r2.rating_norm),2)/count(*)) ) )
as sc
from ratings r1, ratings r2, users u1, users u2, movies m
where r1.userid=u1.userid
and r2.userid=u2.userid
and r1.itemid=r2.itemid
and r1.itemid=m.movieid
and u1.username<>u2.username
group by u1.username, u2.username
order by u1.username, sc desc;
```

Collaborative Filtering

3. Ranking the reviewers

```
# rank the reviewers
select x.username1, max(x.sc) as maxsc
from (
select u1.username as username1, u2.username as username2, s.sc
from sim s, users u1, users u2
where s.userid1=u1.userid
and s.userid2=u2.userid) x
group by x.username1
order by maxsc;
```

Collaborative Filtering

4. Recommending Items

- Using the correlation coefficient that ranks users with respect to each other.
- Use this score to rank products.

```
# Finally! Make recommendations for a given user!
select u1.username, m.title, sum( r.rating_norm*s.sc ) / sum(s.sc) rating
from ratings r, users u1, users u2, movies m, sim s
where r.userid=u2.userid
and r.itemid=m.movieid
and s.userid1=u1.userid
and s.userid2=u2.userid
and u1.username='breuerr'
and r.itemid not in (
select r.itemid
from users u, ratings r
where r.userid=u.userid
and u.username='breuerr'
)
group by m.title
order by rating desc;
```

Algorithms for Collaborative Filtering 1: Memory-Based Algorithms (Breese et al, UAI98)

- K-nearest neighbor

$$w(a,i) = \begin{cases} 1 & \text{if } i \in \text{neighbors}(a) \\ 0 & \text{else} \end{cases}$$

- Pearson correlation coefficient (Resnick ’94, Grouplens):

$$w(a,i) = \frac{\sum_j (v_{a,j} - \overline{v}_a)(v_{i,j} - \overline{v}_i)}{\sqrt{\sum_j (v_{a,j} - \overline{v}_a)^2 \sum_j (v_{i,j} - \overline{v}_i)^2}}$$

- Cosine distance (from IR)

$$w(a,i) = \sum_j \frac{v_{a,j}}{\sqrt{\sum_{k \in I_a} v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{k \in I_i} v_{i,k}^2}}$$

Algorithms for Collaborative Filtering 1: Memory-Based Algorithms (Breese et al, UAI98)

- Cosine with "inverse user frequency" $f_i = log(n/n_j)$, where n is number of users, n_j is number of users voting for item j

$$w(a,i) = \frac{\sum_j f_j \sum_j f_j v_{a,j} v_{i,j} - (\sum_j f_j v_{a,j})(\sum_j f_j v_{i,j}))}{\sqrt{UV}}$$

where

$$U = \sum_j f_j (\sum_j f_j v_{a,j}^2 - (\sum_j f_j v_{a,j})^2)$$

$$V = \sum_j f_j (\sum_j f_j v_{i,j}^2 - (\sum_j f_j v_{i,j})^2)$$

LIBRA Book Recommender

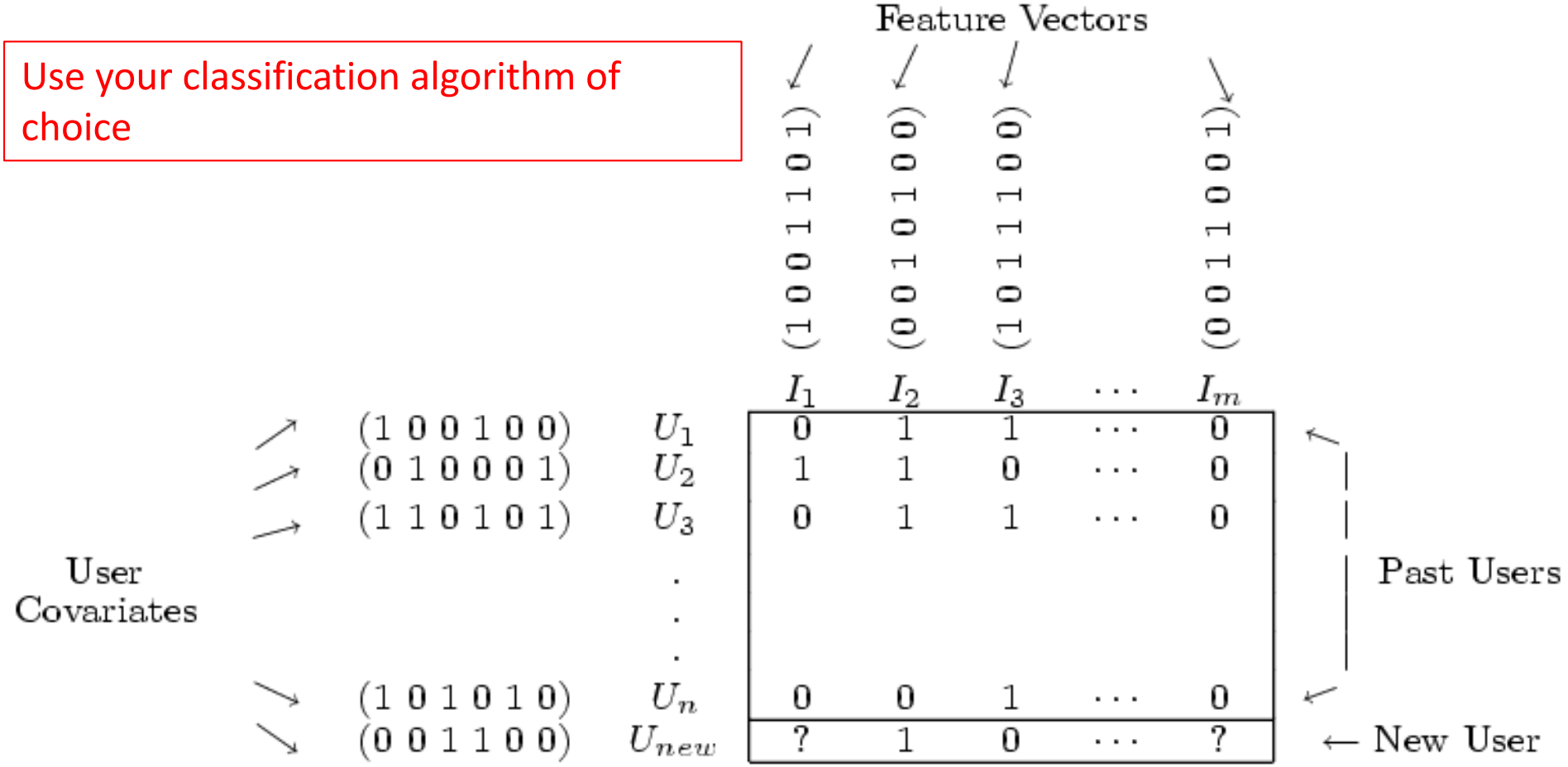
Content-Based Book Recommending Using Learning for Text Categorization. Raymond J. Mooney, Loriene Roy, Univ Texas/Austin; DL-2000

[CF] assumes that a given user's tastes are generally the same as another user ... Items that have not been rated by a sufficient number of users cannot be effectively recommended. Unfortunately, statistics on library use indicate that most books are utilized by very few patrons. ... [CF] approaches ... recommend popular titles, perpetuating homogeneity.... this approach raises concerns about privacy and access to proprietary customer data.

Collaborative + Content Filtering

(Basu et al, AAAI98; Condliff et al, AI-STATS99)

Use your classification algorithm of choice



Content Filtering

(Basu et al, AAAI98; Condliff et al, AI-STATS99)

Engineer features for content filtering, Integrate within ensemble method with Collaborative filtering

		Airplane	**Matrix**	**Room with a View**	**...**	**Hidalgo**
		comedy	**action**	**romance**	**...**	**action**
Joe	*27,M,70k*	9	7	2		7
Carol	*53,F,20k*	8		9		
...						
Kumar	*25,M,22k*	9	3			6
U_a	*48,M,81k*	4	7	?	?	?

Collaborative + Content Filtering
As Classification (Basu et al, AAAI98)

genre={romance}, age=48, sex=male, income=81k,
usersWhoLikedMovie={Carol}, moviesLikedByUser={Matrix,Airplane}, ...

genre={action}, age=48, sex=male, income=81k, usersWhoLikedMovie = {Joe,Kumar}, moviesLikedByUser={Matrix,Airplane},...

		Airplane	**Matrix**	**Room with a View**	**...**	**Hidalgo**
		comedy	**action**	**romance**	**...**	**action**
Joe	*27,M,70k*	1	1	0		1
Carol	*53,F,20k*	1		1		0
...						
Kumar	*25,M,22k*	1	0	0		1
U_a	*48,M,81k*	1	1	?	?	?

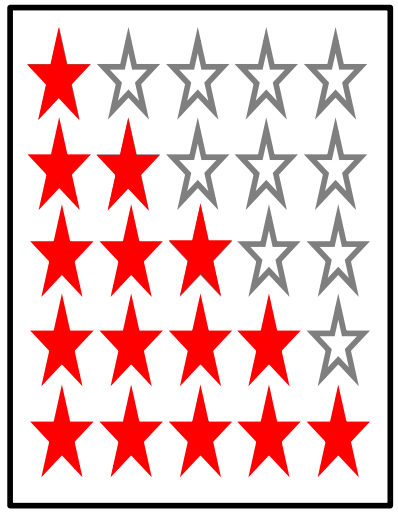
Content + Collaborative Filtering - Theory

Optional content – content + collaborative filtering

Example: Predicting movie ratings

User rates movies using one to five stars

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last				
Romance forever				
Cute puppies of love				
Nonstop car chases				
Swords vs. karate				



n_u = no. users

n_m = no. movies

$r(i,j)$ = 1 if user j has rated movie i

$y^{(i,j)}$ = rating given by user j to movie i (defined only if $r(i,j)=1$)

Content-based recommender systems

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)		
Love at last	5	5	0	0		
Romance forever	5	?	?	0		
Cute puppies of love	?	4	0	?		
Nonstop car chases	0	0	5	4	0.1	1.0
Swords vs. karate	0	0	5	?	0	0.9

For each user j, learn a parameter $\theta^{(j)} \in \mathbb{R}^3$. Predict user j as rating movie i with $(\theta^{(j)})^T x^{(i)}$ stars.

Problem formulation

$r(i,j) = 1$ if user j has rated movie i (0 otherwise)
$y^{(i,j)} =$ rating by user j on movie i (if defined)

$\theta^{(j)}$ = parameter vector for user j
$x^{(i)}$ = feature vector for movie i
For user j, movie i, predicted rating: $(\theta^{(j)})^T(x^{(i)})$

$m^{(j)}$ = no. of movies rated by user j
To learn $\theta^{(j)}$:

Optimization objective:

To learn $\theta^{(j)}$ (parameter for user j):

$$\min_{\theta^{(j)}} \frac{1}{2} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

To learn $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(n_u)}$:

$$\min_{\theta^{(1)},\ldots,\theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

Optimization algorithm:

$$\min_{\theta^{(1)},\ldots,\theta^{(n_u)}} \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(\theta_k^{(j)})^2$$

Gradient descent update:

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} \quad (\text{for } k = 0)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} + \lambda\theta_k^{(j)}\right) \quad (\text{for } k \neq 0)$$

Problem motivation – back to collaborative filtering

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	0.9	0
Romance forever	5	?	?	0	1.0	0.01
Cute puppies of love	?	4	0	?	0.99	0
Nonstop car chases	0	0	5	4	0.1	1.0
Swords vs. karate	0	0	5	?	0	0.9

Problem motivation

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	?	?
Romance forever	5	?	?	0	?	?
Cute puppies of love	?	4	0	?	?	?
Nonstop car chases	0	0	5	4	?	?
Swords vs. karate	0	0	5	?	?	?

$$\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}, \theta^{(2)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}, \theta^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}, \theta^{(4)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$$

Optimization algorithm

Given $\theta^{(1)}, \ldots, \theta^{(n_u)}$, to learn $x^{(i)}$:

$$\min_{x^{(i)}} \frac{1}{2} \sum_{j:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} (x_k^{(i)})^2$$

Given $\theta^{(1)}, \ldots, \theta^{(n_u)}$, to learn $x^{(1)}, \ldots, x^{(n_m)}$:

$$\min_{x^{(1)}, \ldots, x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2$$

Collaborative filtering

Given $x^{(1)}, \ldots, x^{(n_m)}$ (and movie ratings),
can estimate $\theta^{(1)}, \ldots, \theta^{(n_u)}$

Given $\theta^{(1)}, \ldots, \theta^{(n_u)}$,
can estimate $x^{(1)}, \ldots, x^{(n_m)}$

Collaborative filtering optimization objective

Given $x^{(1)}, \ldots, x^{(n_m)}$, estimate $\theta^{(1)}, \ldots, \theta^{(n_u)}$:

$$\min_{\theta^{(1)},\ldots,\theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

Given $\theta^{(1)}, \ldots, \theta^{(n_u)}$, estimate $x^{(1)}, \ldots, x^{(n_m)}$:

$$\min_{x^{(1)},\ldots,x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2$$

Minimizing $x^{(1)}, \ldots, x^{(n_m)}$ and $\theta^{(1)}, \ldots, \theta^{(n_u)}$ simultaneously:

$$J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)}) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

$$\min_{\substack{x^{(1)},\ldots,x^{(n_m)} \\ \theta^{(1)},\ldots,\theta^{(n_u)}}} J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)})$$

Collaborative filtering algorithm

1. Initialize $x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)}$ to small random values.
2. Minimize $J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)})$ using gradient descent (or an advanced optimization algorithm). E.g. for every $j = 1, \ldots, n_u, i = 1, \ldots, n_m$:

$$x_k^{(i)} := x_k^{(i)} - \alpha \left(\sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) \theta_k^{(j)} + \lambda x_k^{(i)} \right)$$

3. For a user with parameters θ and a movie with (learned) features x, predict a star rating of $\theta^T x$.

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} + \lambda \theta_k^{(j)} \right)$$

Collaborative filtering

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last	5	5	0	0
Romance forever	5	?	?	0
Cute puppies of love	?	4	0	?
Nonstop car chases	0	0	5	4
Swords vs. karate	0	0	5	?

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix}$$

Collaborative filtering - vectorization

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix}$$

Predicted ratings:

$$\begin{bmatrix} (\theta^{(1)})^T(x^{(1)}) & (\theta^{(2)})^T(x^{(1)}) & \dots & (\theta^{(n_u)})^T(x^{(1)}) \\ (\theta^{(1)})^T(x^{(2)}) & (\theta^{(2)})^T(x^{(2)}) & \dots & (\theta^{(n_u)})^T(x^{(2)}) \\ \vdots & \vdots & \vdots & \vdots \\ (\theta^{(1)})^T(x^{(n_m)}) & (\theta^{(2)})^T(x^{(n_m)}) & \dots & (\theta^{(n_u)})^T(x^{(n_m)}) \end{bmatrix}$$

Finding related movies

For each product i, we learn a feature vector $x^{(i)} \in \mathbb{R}^n$.

How to find movies j related to movie i?

5 most similar movies to movie i:
Find the 5 movies j with the smallest $\|x^{(i)} - x^{(j)}\|$.

Users who have not rated any movies – mean normalization

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	Eve (5)
Love at last	5	5	0	0	?
Romance forever	5	?	?	0	?
Cute puppies of love	?	4	0	?	?
Nonstop car chases	0	0	5	4	?
Swords vs. karate	0	0	5	?	?

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix}$$

$$\min_{\substack{\theta^{(1)},\ldots,\theta^{(n_u)} \\ x^{(1)},\ldots,x^{(n_m)}}} \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

Mean Normalization:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} \qquad \mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix} \rightarrow Y = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

For user j, on movie i predict:

User 5 (Eve):

References

- Trevor Hastie, Robert Tibshirani, Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Second Edition, 2009.
- Anand Rajaraman, Jeffrey D. Ullman. *Mining of Massive Datasets*. 2009.
- Toby Seagram*, Collective Intelligence*. 2007.
- *Recommending And Evaluating Choices In A Virtual Community Of Use.* Will Hill, Larry Stead, Mark Rosenstein and George Furnas, Bellcore; CHI 1995.
- Andrew Ng, Standford
- William W. Cohen, CMU