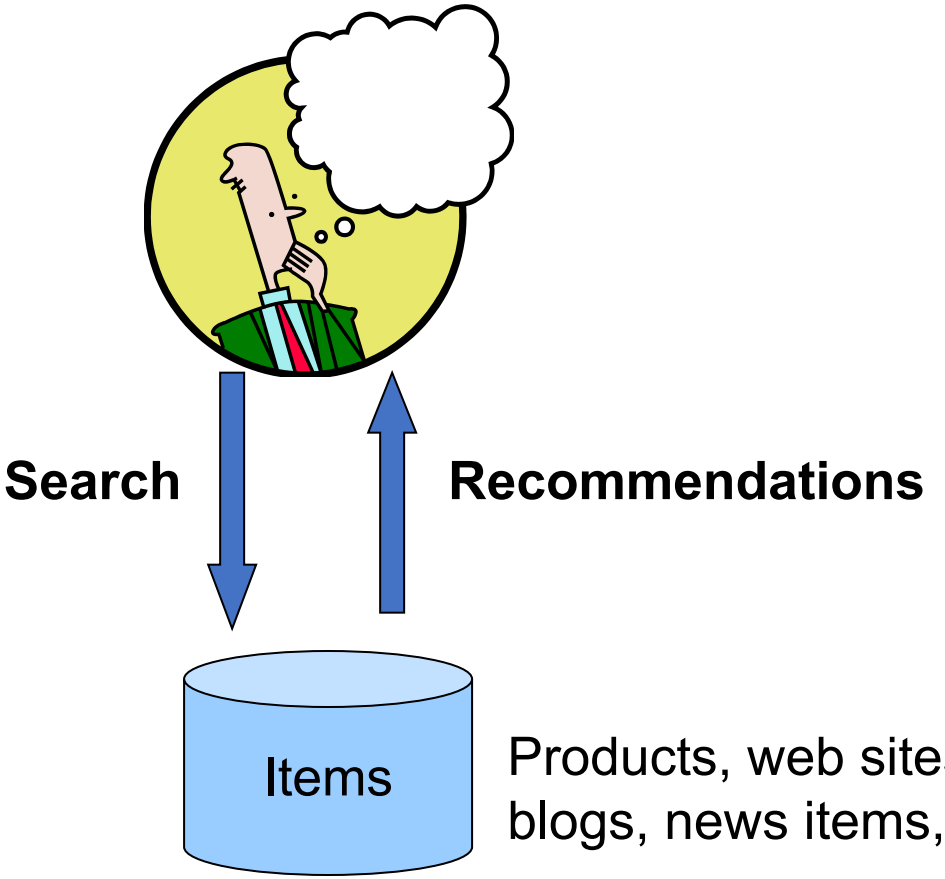
COMP9313: Big Data Management

Recommender System

Source from Dr. Xin Cao

Recommendations



Products, web sites, blogs, news items, …

Examples:

amazon.com

PANDORA

StumbleUpon

del.icio.us

NETFLIX

movielens
helping you find the *right* movies

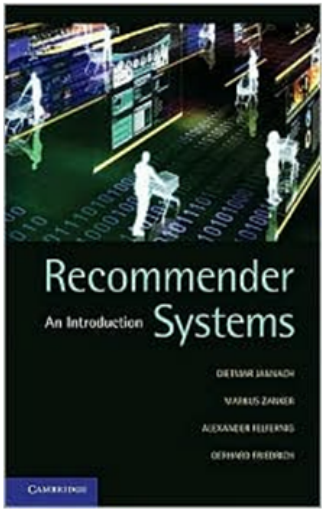
last·fm
the social music revolution

Google News

YouTube

XBOX LIVE

Recommender Systems



Recommender Systems: An Introduction
by Dietmar Jannach, Markus Zanker, Alexander Felfernig, Gerhard Friedrich

AVERAGE CUSTOMER RATING:

(Be the first to review)

Gefällt mir Registrieren, um sehen zu können, was deinen Freunden gefällt.

FORMAT:
Hardcover

NOOKbook (eBook) - not available
Tell the publisher you want this in NOOKbook format

NEW FROM BN.COM

~~$65.00~~ List Price
$52.00 Online Price
(You Save 20%)

Add to Cart

NEW & USED FROM OUR

New starting at **$56.46** (You S
Used starting at **$51.98** (You S

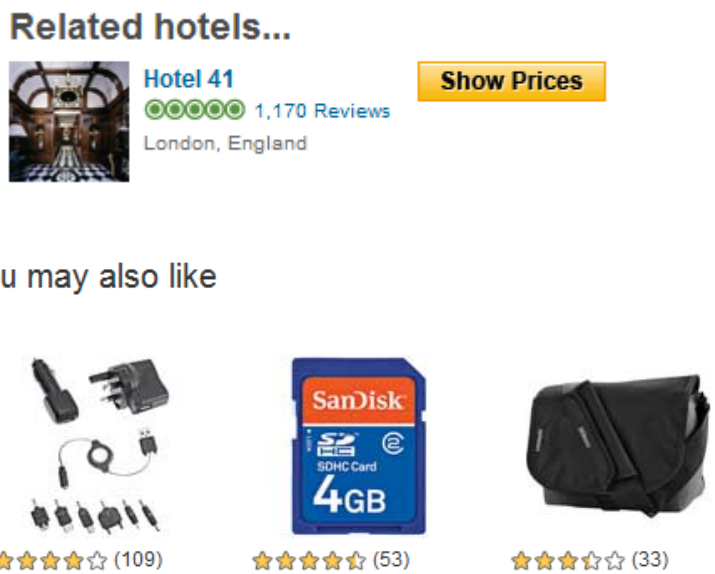
See All Prices

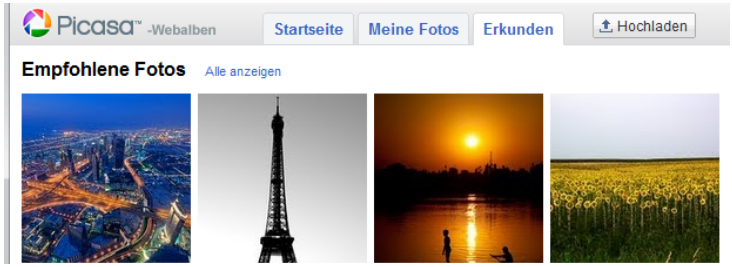
Table of Contents

Customers who bought this also bought

Recommender Systems

- Application areas
  - Movie recommendation (Netflix)
  - Related product recommendation (Amazon)
  - Web page ranking (Google)
  - Social recommendation (Facebook)
  - … …





Netflix Movie Recommendation

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	?

Why using Recommender Systems?

- Value for the customer
  - Find things that are interesting
  - Narrow down the set of choices
  - Help me explore the space of options
  - Discover new things
  - Entertainment
  - …

- Value for the provider
  - Additional and probably unique personalized service for the customer
  - Increase trust and customer loyalty
  - Increase sales, click trough rates, conversion etc.
  - Opportunities for promotion, persuasion
  - Obtain more knowledge about customers
  - …

Recommender systems

- RS seen as a **function**
- Given:
  - User model (e.g. ratings, preferences, demographics, situational context)
  - Items (with or without description of item characteristics)
- Find:
  - Relevance score. Used for ranking.
- Finally:
  - Recommend items that are assumed to be relevant
- But:
  - Remember that relevance might be context-dependent
  - Characteristics of the list itself might be important (diversity)

Formal Model

- $\boldsymbol{X}$ = set of Customers
- $\boldsymbol{S}$ = set of Items
- Utility function u: $\boldsymbol{X} \times \boldsymbol{S} \rightarrow \boldsymbol{R}$
  - $\boldsymbol{R}$ = set of ratings
  - $\boldsymbol{R}$ 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

Key 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
    - We are not interested in knowing what you don't like but what you like
- Evaluating extrapolation methods
  - How to measure success/performance of recommendation methods

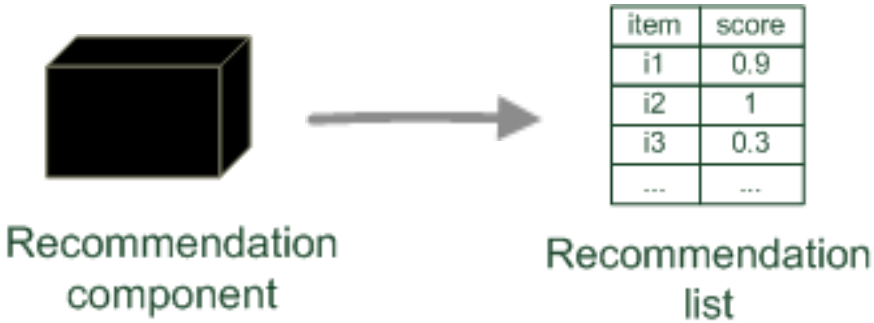
Gathering Ratings

- Explicit
  - Ask people to rate items
  - Doesn't work well in practice – people can't be bothered

- Implicit
  - Learn ratings from user actions
    - E.g., purchase implies high rating

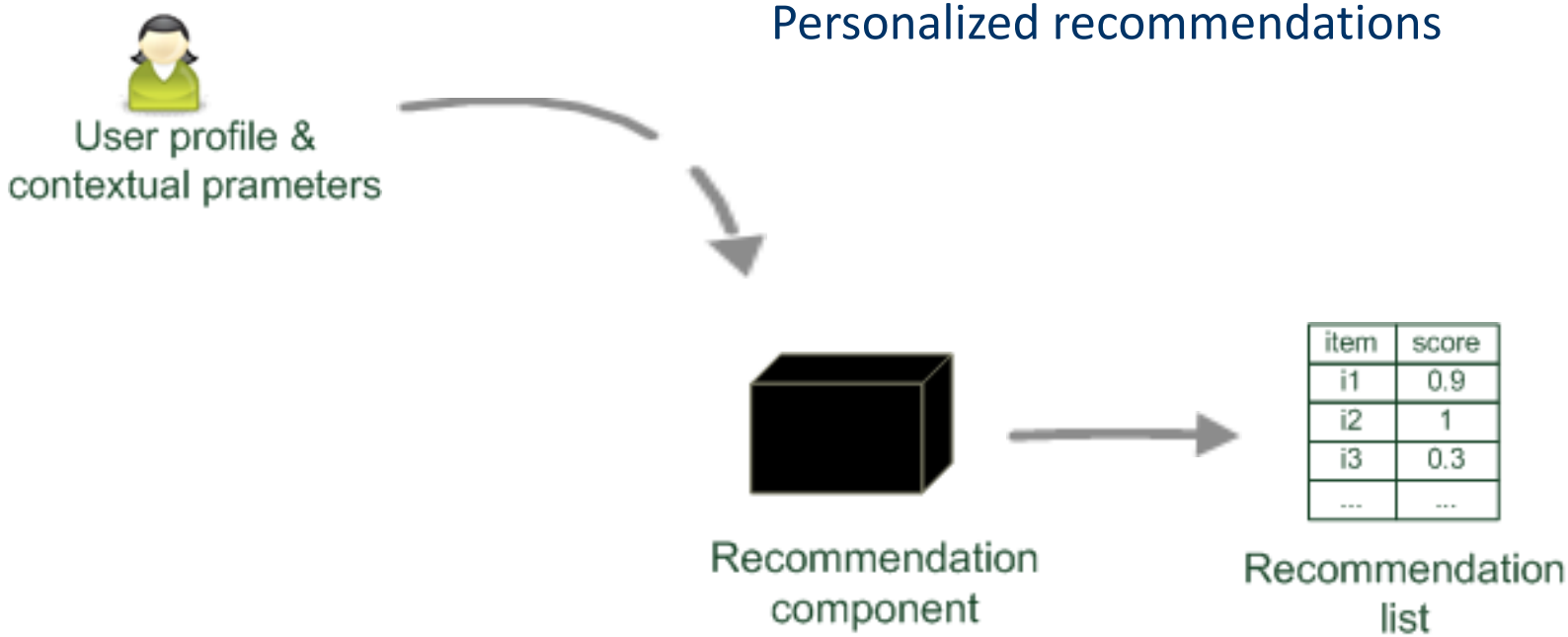
Paradigms of recommender systems

Recommender systems reduce information overload by estimating relevance



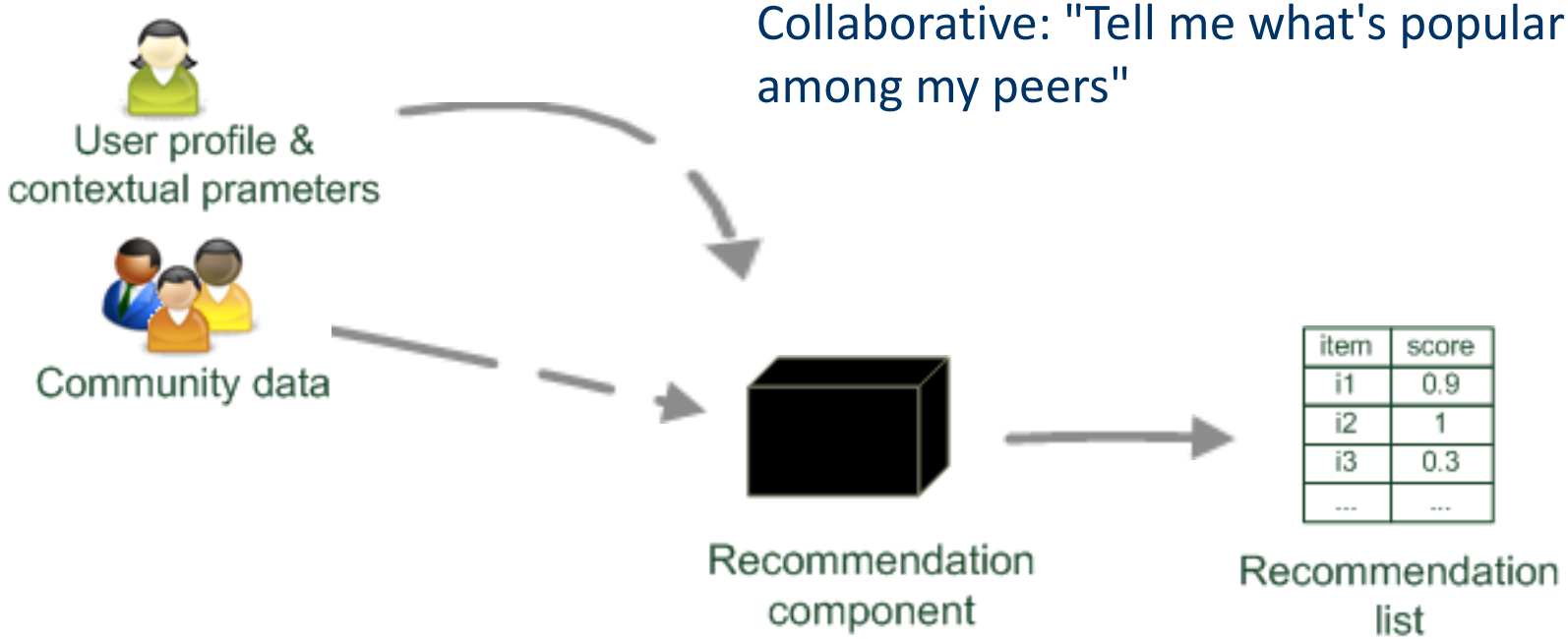
Paradigms of recommender systems

Personalized recommendations



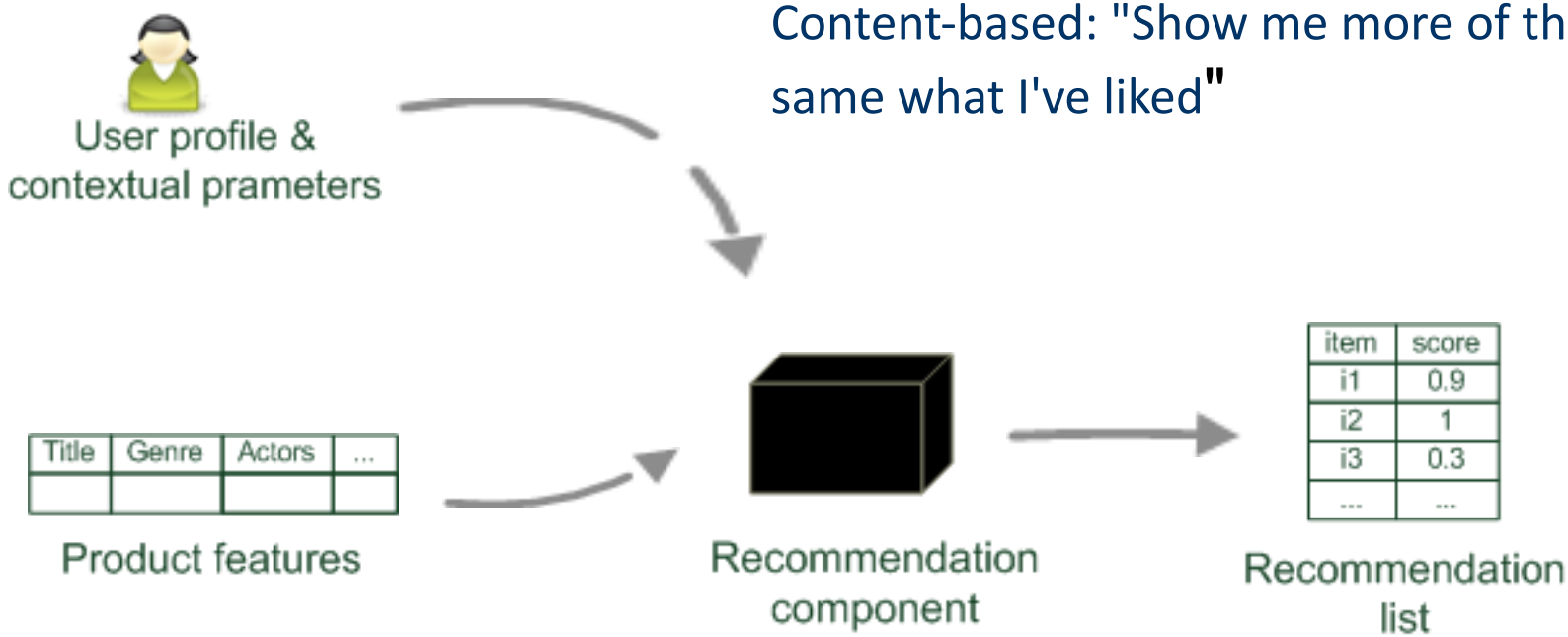
Paradigms of recommender systems

Collaborative: "Tell me what's popular among my peers"



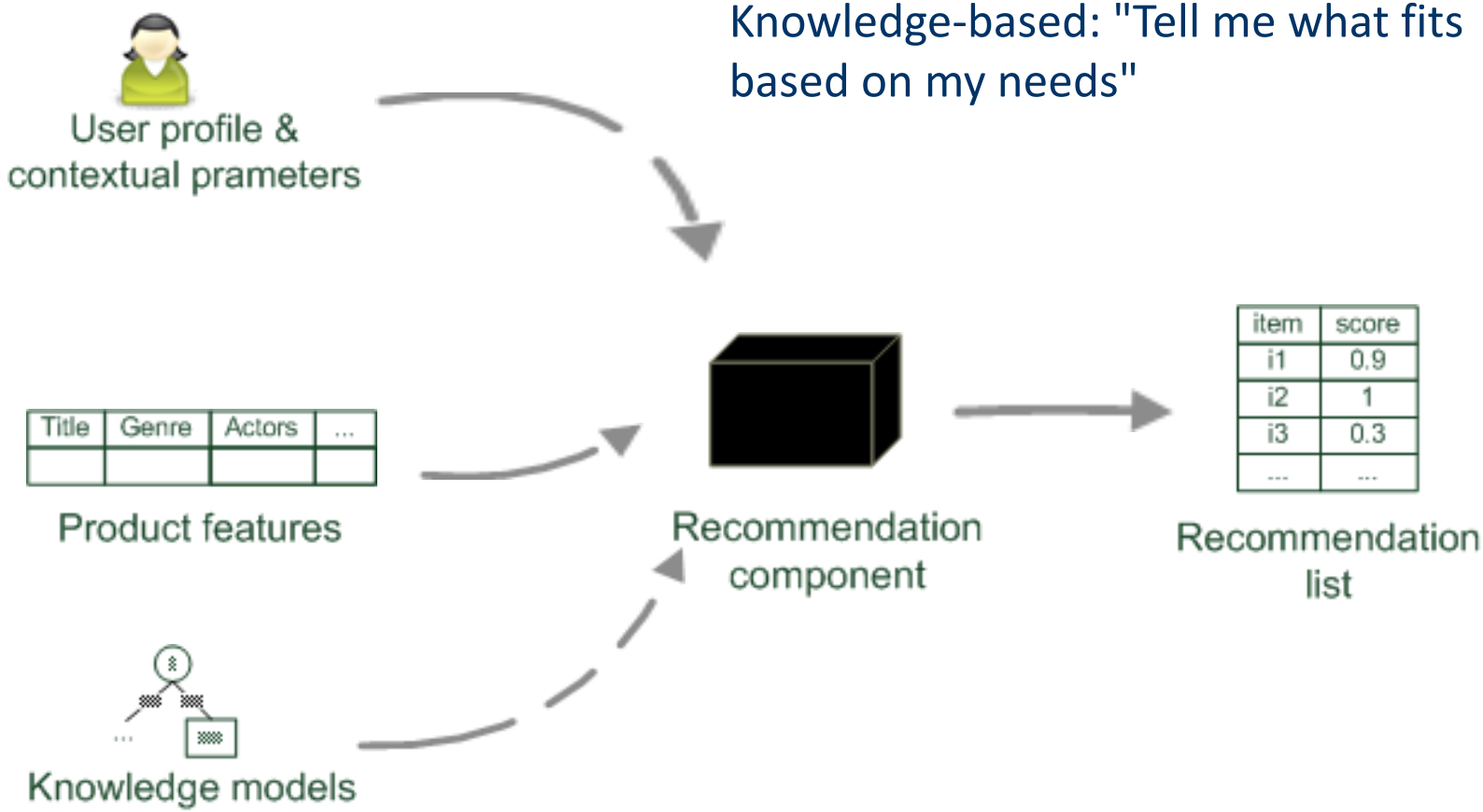
Paradigms of recommender systems

Content-based: "Show me more of the same what I've liked"



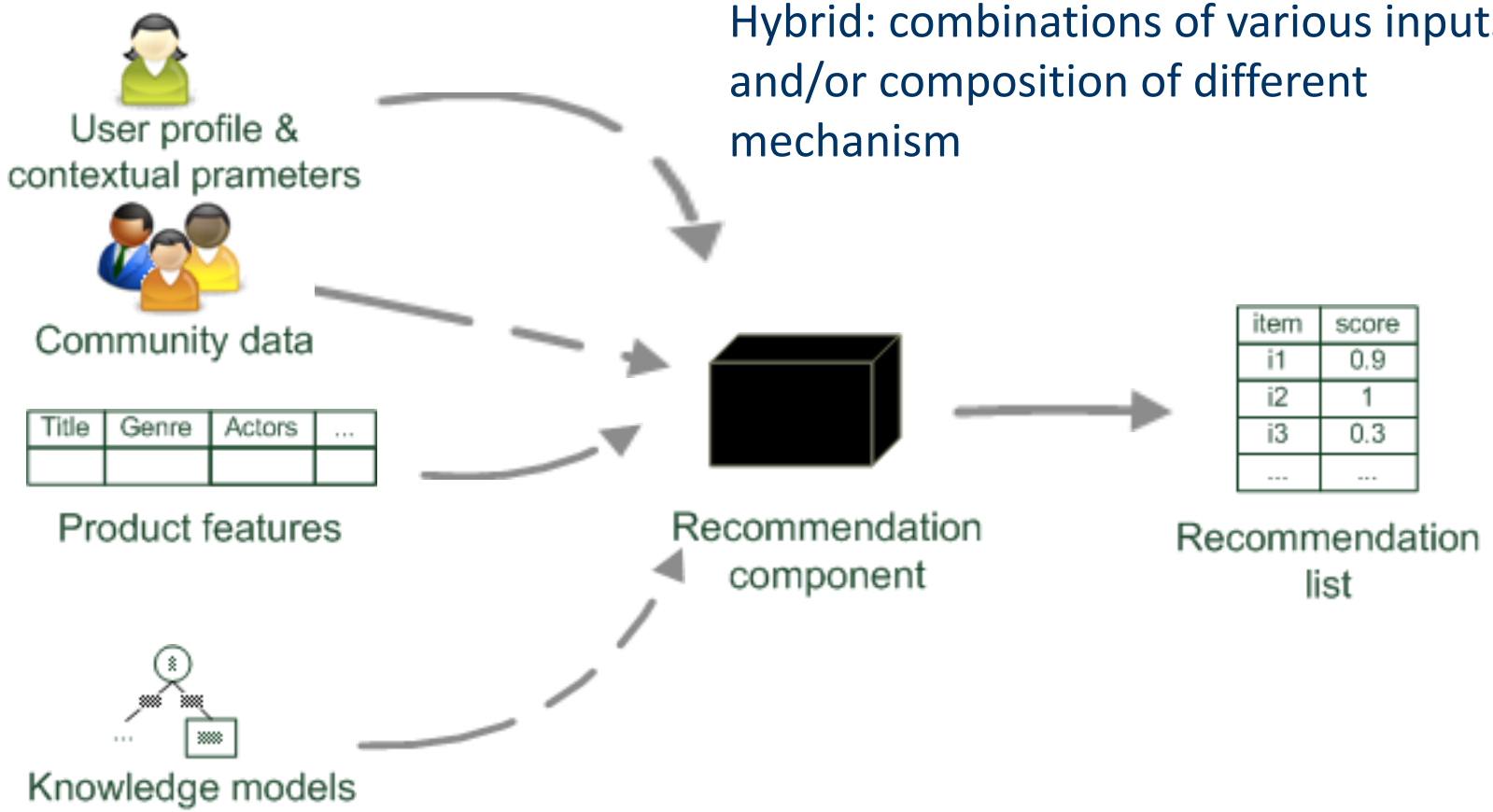
Paradigms of recommender systems

Knowledge-based: "Tell me what fits based on my needs"

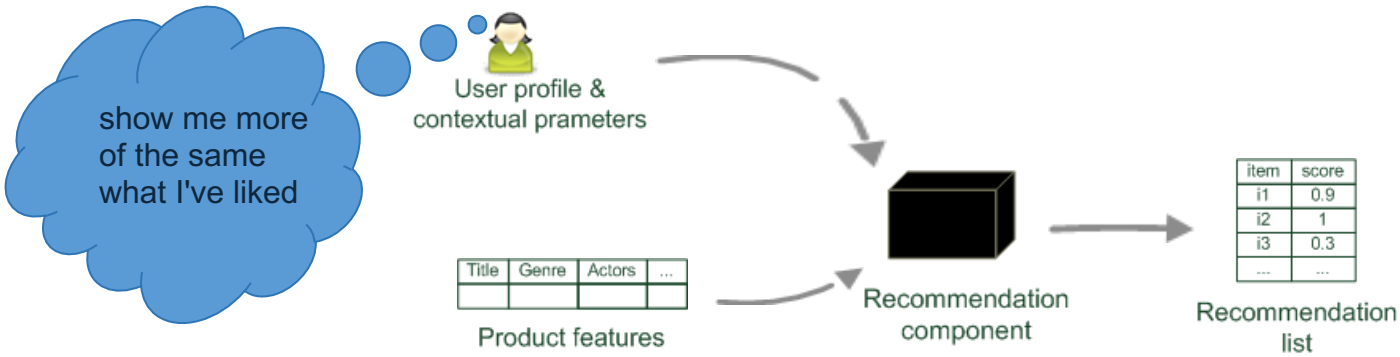


Paradigms of recommender systems

Hybrid: combinations of various inputs and/or composition of different mechanism



Content-based Recommendation



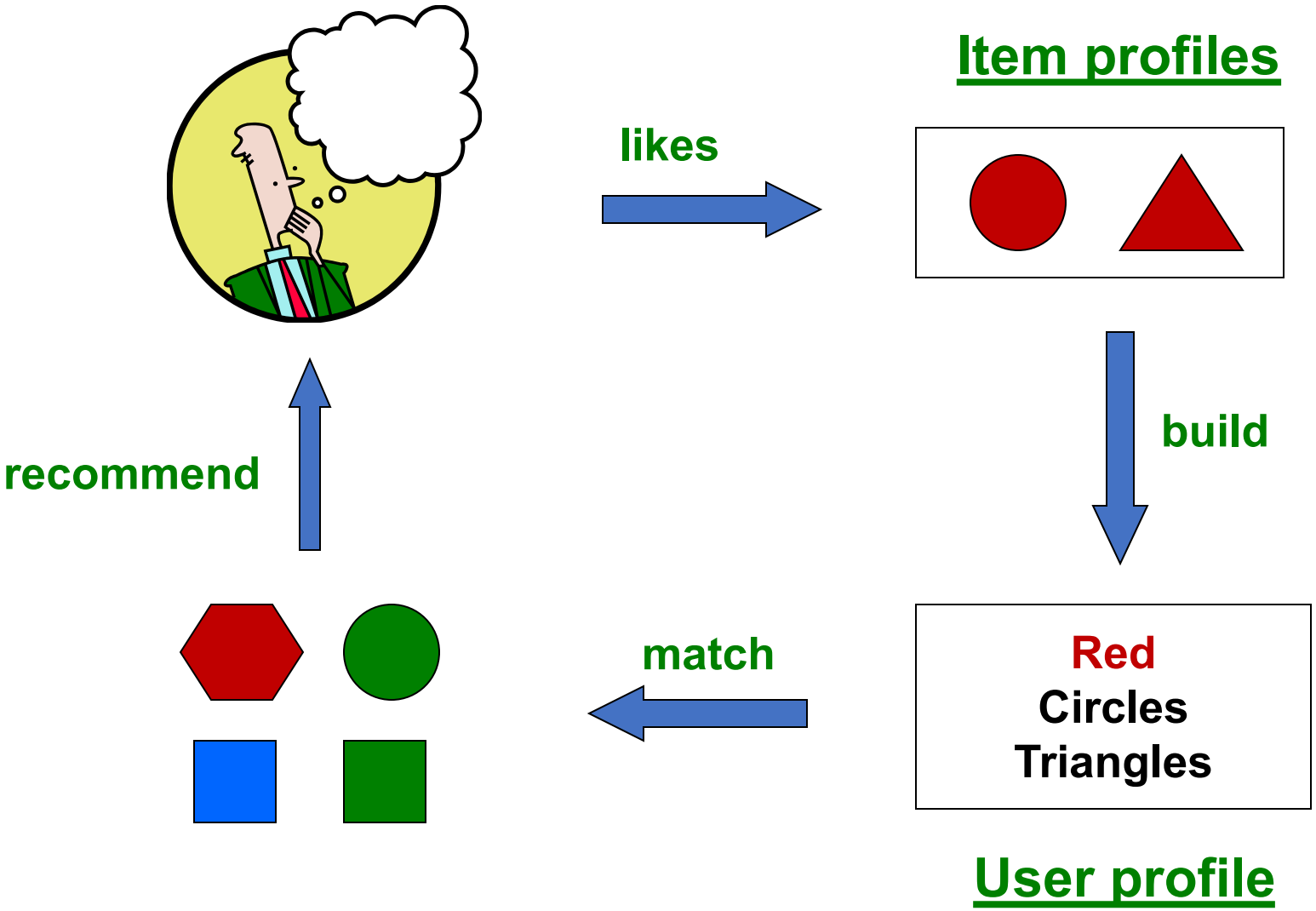
Content-based Recommendations

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

- What do we need:
  - Some information about the available items such as the genre ("content")
  - Some sort of *user profile* describing what the user likes (the preferences)

- ***Example:***
  - Movie recommendations:
    - Recommend movies with same actor(s), director, genre, …
  - Websites, blogs, news:
    - Recommend other sites with “similar” content

Plan of Action



What is the “Content"?

- Most CB-recommendation techniques were applied to recommending text documents.
  - Like web pages or newsgroup messages for example.
- Content of items can also be represented as text documents.
  - With textual descriptions of their basic characteristics.
  - Structured: Each item is described by the same set of attributes

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	Press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, murder, neo-Nazism

  - Unstructured: free-text description.

Item Profiles

- For each item, create an **item profile**
- Profile is a set (vector) of features
  - **Movies:** author, title, actor, director,…
  - **Text:** Set of "important" words in document
- How to pick important features?
  - Usual heuristic from text mining is **TF-IDF** (Term frequency * Inverse Doc Frequency)
    - **Term … Feature**
    - **Document … Item**

User Profiles and Prediction

- User profile possibilities:
  - Weighted average of rated item profiles
  - **Variation:** weight by difference from average rating for item
  - …

- **Prediction heuristic:**
  - Given user profile $\boldsymbol{x}$ and item profile $\boldsymbol{i}$, estimate

$$u(\boldsymbol{x}, \boldsymbol{i}) = \cos(\boldsymbol{x}, \boldsymbol{i}) = \frac{\boldsymbol{x} \cdot \boldsymbol{i}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{i}||}$$

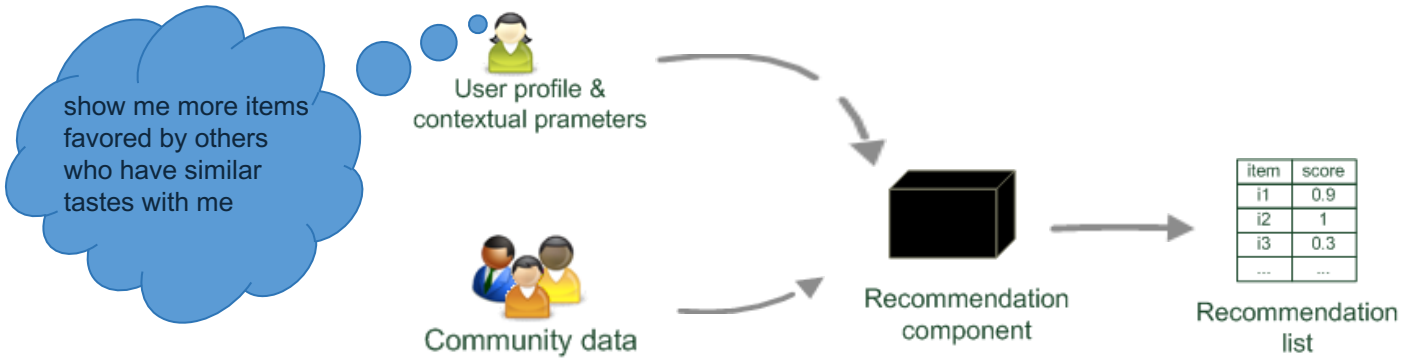
Pros: Content-based Approach

- **+: 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
- **+: Able to provide explanations**
  - Can provide explanations of recommended items by listing content-features that caused an item to be recommended

Cons: Content-based Approach

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

Collaborative Filtering



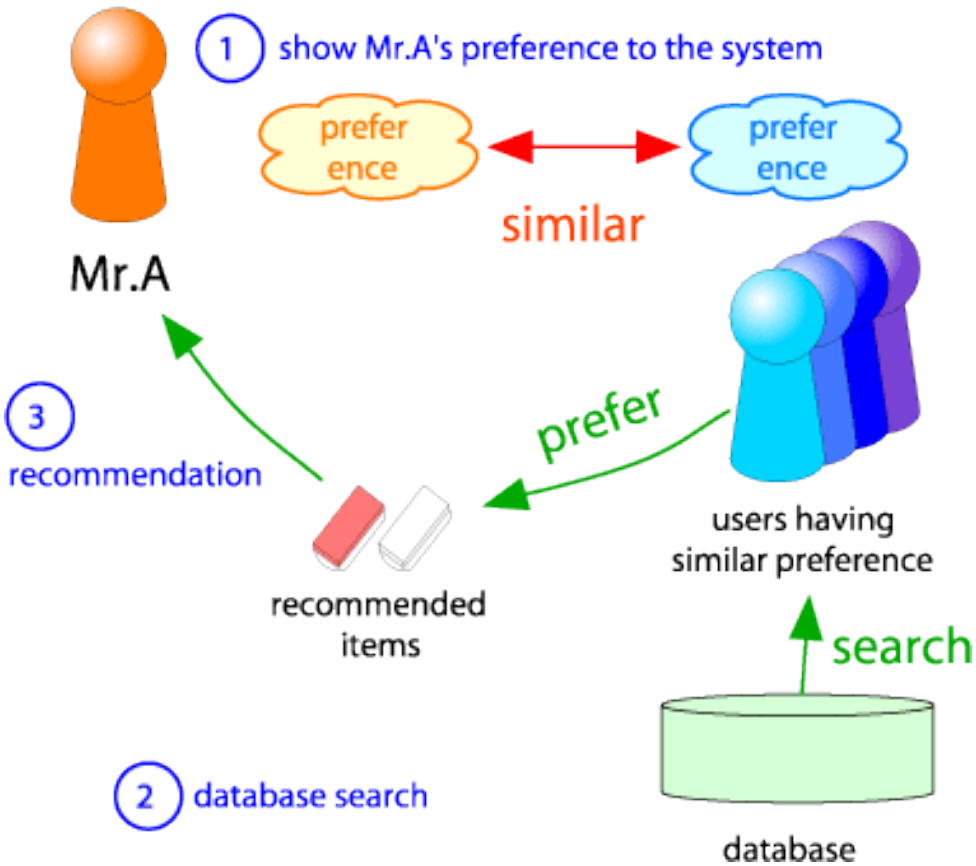
Collaborative Filtering (CF)

- The most prominent approach to generate recommendations
  - used by large, commercial e-commerce sites
  - well-understood, various algorithms and variations exist
  - applicable in many domains (book, movies, DVDs, ..)
- Approach
  - use the "wisdom of the crowd" to recommend items

- Basic assumption and idea
  - Users give ratings to catalog items (implicitly or explicitly)
  - Customers who had similar tastes in the past, will have similar tastes in the future

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***



User-based Nearest-Neighbor Collaborative Filtering

- The basic technique
  - Given an "active user" (Alice) and an item *i* not yet seen by Alice
    - find a set of users (peers/nearest neighbors) who liked the same items as Alice in the past **and** who have rated item *i*
    - use, e.g. the average of their ratings to predict, if Alice will like item *i*
    - do this for all items Alice has not seen and recommend the best-rated

- Basic assumption and idea
  - If users had similar tastes in the past they will have similar tastes in the future
  - User preferences remain stable and consistent over time

User-based Nearest-Neighbor Collaborative Filtering

- Example
  - A database of ratings of the current user, Alice, and some other users is given:

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

  - Determine whether Alice will like or dislike *Item5*, which Alice has not yet rated or seen

User-based Nearest-Neighbor Collaborative Filtering

- Some first questions
  - How do we measure similarity?
  - How many neighbors should we consider?
  - How do we generate a prediction from the neighbors' ratings?

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Finding "Similar" Users

r_x = [*, _, _, *, ***]
r_y = [*, _, **, **, _]

- Let $\boldsymbol{r_x}$ be the vector of user $\boldsymbol{x}$'s ratings
- **Jaccard similarity measure** $\frac{||r_x \cap r_y||}{||r_x \cup r_y||}$
  - **Problem:** Ignores the value of the rating

r_x, r_y as sets:
r_x = {1, 4, 5}
r_y = {1, 3, 4}

- **Cosine similarity measure**
  - $\text{sim}(\boldsymbol{x}, \boldsymbol{y}) = \cos(\boldsymbol{r_x}, \boldsymbol{r_y}) = \frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$
  - **Problem:** Treats missing ratings as "negative"

r_x, r_y as points:
r_x = {1, 0, 0, 1, 3}
r_y = {1, 0, 2, 2, 0}

- **Pearson correlation coefficient**
  - $\boldsymbol{S_{xy}}$ = items rated by both users $\boldsymbol{x}$ and $\boldsymbol{y}$

$$\boldsymbol{sim(x,y)} = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$$

$\overline{r_x}$, $\overline{r_y}$ … avg. rating of **x**, **y**

Similarity Metric

Cosine similarity:

$$sim(x,y) = \frac{\sum_i r_{xi} \cdot r_{yi}}{\sqrt{\sum_i r_{xi}^2} \cdot \sqrt{\sum_i r_{yi}^2}}$$

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

- **Intuitively we want: sim(A, B) > sim(A, C)**
- **Jaccard similarity:** 1/5 < $\frac{4 \times 5}{\sqrt{4^2+5^2+1^2}\sqrt{5^2+5^2+4^2}} = 0.380$
- **Cosine similarity:** 0.380 > 0.322
  - Considers missing ratings as "negative"

	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 vs. A,C:
0.092 > -0.559

Notice cosine sim. is correlation when data is centered at 0

Similarity Metric (Cont')

- A popular similarity measure in user-based CF: **Pearson correlation**

$$\boldsymbol{sim(x,y)} = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$$

  - Possible similarity values between -1 and 1;

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

sim = 0,85

sim = 0,00

sim = 0,70

sim = -0,79

Rating Predictions

From similarity metric to recommendations:

- Let $\boldsymbol{r_x}$ be the vector of user $\boldsymbol{x}$'s ratings
- Let $\boldsymbol{N}$ be the set of $\boldsymbol{k}$ users most similar to $\boldsymbol{x}$ who have rated item $\boldsymbol{i}$
- **Prediction for item *s* of *user x*:**
  - $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$
  - $r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$
  - Other options?
- **Many other tricks possible…**

Shorthand:
$s_{xy} = sim(x, y)$

Memory-based and Model-based Approaches

- User-based CF is said to be "memory-based"
  - the rating matrix is directly used to find neighbors / make predictions
  - does not scale for most real-world scenarios
  - large e-commerce sites have tens of millions of customers and millions of items

- Model-based approaches
  - based on an offline pre-processing or "model-learning" phase
  - at run-time, only the learned model is used to make predictions
  - models are updated / re-trained periodically
  - large variety of techniques used
  - model-building and updating can be computationally expensive

Item-Item Collaborative Filtering

- **So far: User-user collaborative filtering**
- **Another view: Item-item**
  - Basic idea:
    - Use the similarity between items (and not users) to make predictions
  - 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 ***u*** on item ***j***
$N(i;x)$… set items rated by ***x*** similar to ***i***

Item-Item Collaborative Filtering

- Example:
  - Look for items that are similar to Item5
  - Take Alice's ratings for these items to predict the rating for Item5

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Item-Item CF (|N|=2)

users

movies

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

- unknown rating

- rating between 1 to 5

Item-Item CF (|N|=2)

users

movies

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

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

Item-Item CF (|N|=2)

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		**1.00**
2			5	4			4			2	1	3	**-0.18**
3	2	4		1	2		3		4	3	5		**0.41**
4		2	4		5			4			2		**-0.10**
5			4	3	4	2					2	5	**-0.31**
6	1		3		3			2			4		**0.59**

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

Here we use adjust cosine 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)

users

movies

	1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		0.41
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		0.59

Compute similarity weights:

$s_{1,3}=0.41$, $s_{1,6}=0.59$

Item-Item CF (|N|=2)

users

movies

	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		2.6	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Predict by taking weighted average:

$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{jx}}{\sum s_{ij}}$$

$r_{1.5}$ = (0.41*2 + 0.59*3) / (0.41+0.59) = 2.6

Item-Item vs. User-User

	Avatar	LOTR	Matrix	Pirates
Alice	1		0.8	
Bob		0.5		0.3
Carol	0.9		1	0.8
David			1	0.4

- In practice, it has been observed that item-item often works better than user-user
  - Why? Items are simpler, users have multiple tastes

More on Ratings – Explicit Ratings

- Probably the most precise ratings
- Most commonly used (1 to 5, 1 to 7 Likert response scales)
- Main problems
  - Users not always willing to rate many items
    - number of available ratings could be too small → sparse rating matrices → poor recommendation quality
  - How to stimulate users to rate more items?

More on Ratings – Implicit Ratings

- Typically collected by the web shop or application in which the recommender system is embedded
- When a customer buys an item, for instance, many recommender systems interpret this behavior as a positive rating
- Clicks, page views, time spent on some page, demo downloads …
- Implicit ratings can be collected constantly and do not require additional efforts from the side of the user
- Main problem
  - One cannot be sure whether the user behavior is correctly interpreted
  - For example, a user might not like all the books he or she has bought; the user also might have bought a book for someone else
- Implicit ratings can be used in addition to explicit ones; question of correctness of interpretation

Collaborative Filtering: Complexity

- Expensive step is finding ***k*** most similar customers: **O(|X|)**
- **Too expensive to do at runtime**
  - Could pre-compute
- Naïve pre-computation takes time **O(|X|^2)**
  - X … set of customers
- Ways of doing this:
  - Near-neighbor search in high dimensions (**LSH**)
  - Clustering
  - Dimensionality reduction
  - … …

- Supported by Hadoop: Apache Mahout
  https://mahout.apache.org/users/basics/algorithms.html

What is a Good Recommendation in Practice?

- Total sales numbers
- Promotion of certain items
- …
- Click-through-rates
- Interactivity on platform
- …
- Customer return rates
- Customer satisfaction and loyalty

Evaluation

movies

users

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

Evaluation

movies

users

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

Test Data Set

Evaluating Predictions

- **Compare predictions with known ratings**
  - **Root-mean-square error** (RMSE)
    - $\sqrt{\sum_{xi}(r_{xi} - r^*_{xi})^2}$ where $\boldsymbol{r_{xi}}$ is predicted, $\boldsymbol{r^*_{xi}}$ is the true rating of $\boldsymbol{x}$ on $\boldsymbol{i}$
  - **Precision at top 10**:
    - % of those in top 10
  - **Rank Correlation**:
    - Spearman's *correlation* between system's and user's complete rankings

- **Another approach: 0/1 model**
  - **Coverage:**
    - Number of items/users for which system can make predictions
  - **Precision:**
    - Accuracy of predictions
  - **Receiver operating characteristic** (ROC)
    - Tradeoff curve between false positives and false negatives

The Netflix Prize

- **Training data**
  - 100 million ratings, 480,000 users, 17,770 movies
  - 6 years of data: 2000-2005
- **Test data**
  - Last few ratings of each user (2.8 million)
  - **Evaluation criterion:** Root Mean Square Error (RMSE) $= \frac{1}{|R|}\sqrt{\sum_{(i,x)\in R}(\hat{r}_{xi} - r_{xi})^2}$
  - **Netflix's system RMSE: 0.9514**
- **Competition**
  - 2,700+ teams
  - **$1 million** prize for 10% improvement on Netflix

The Netflix Utility Matrix R

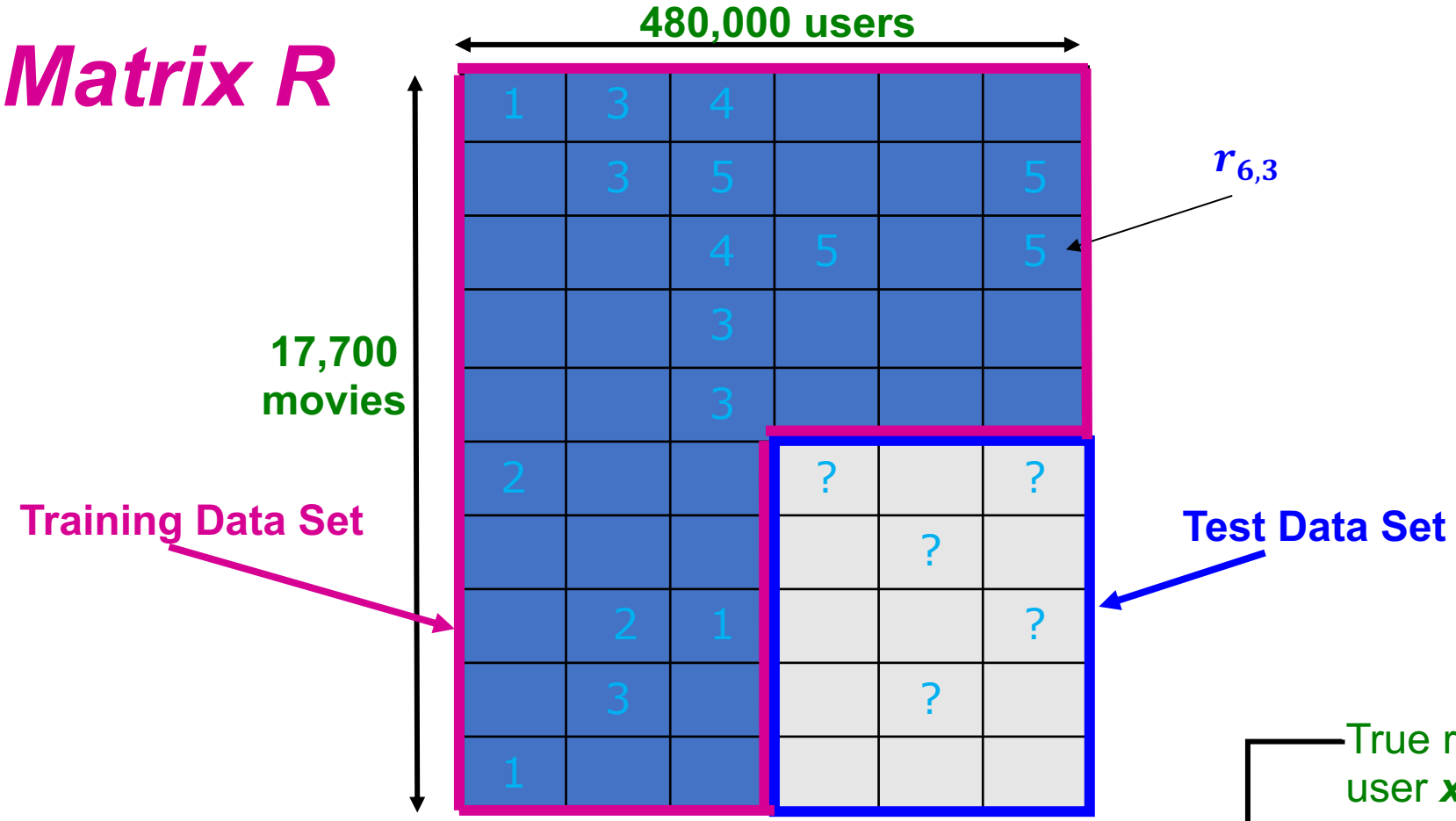
Matrix R

480,000 users

17,700 movies

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

Utility Matrix R: Evaluation

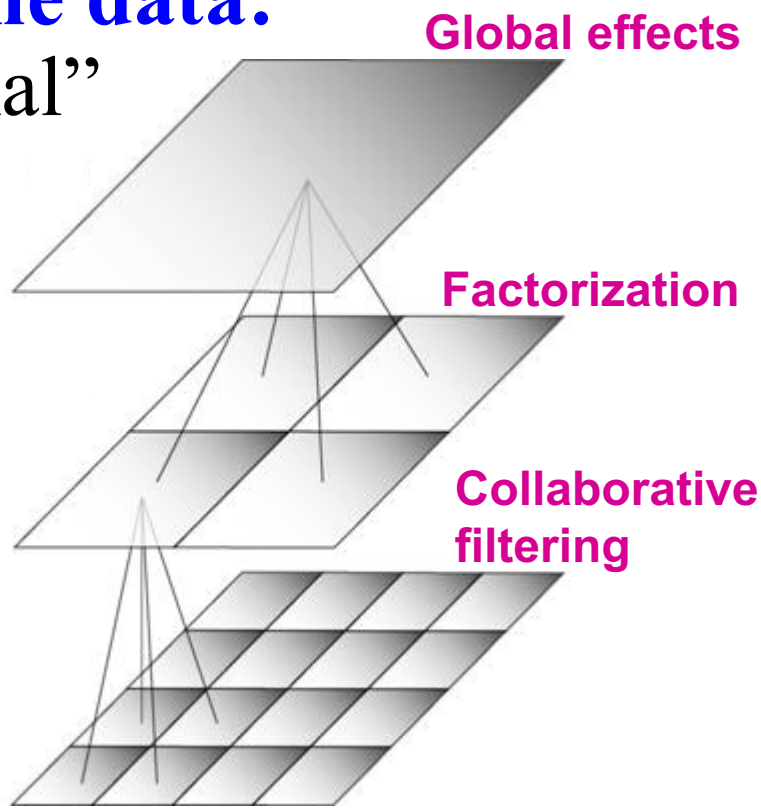


True rating of user ***x*** on item ***i***

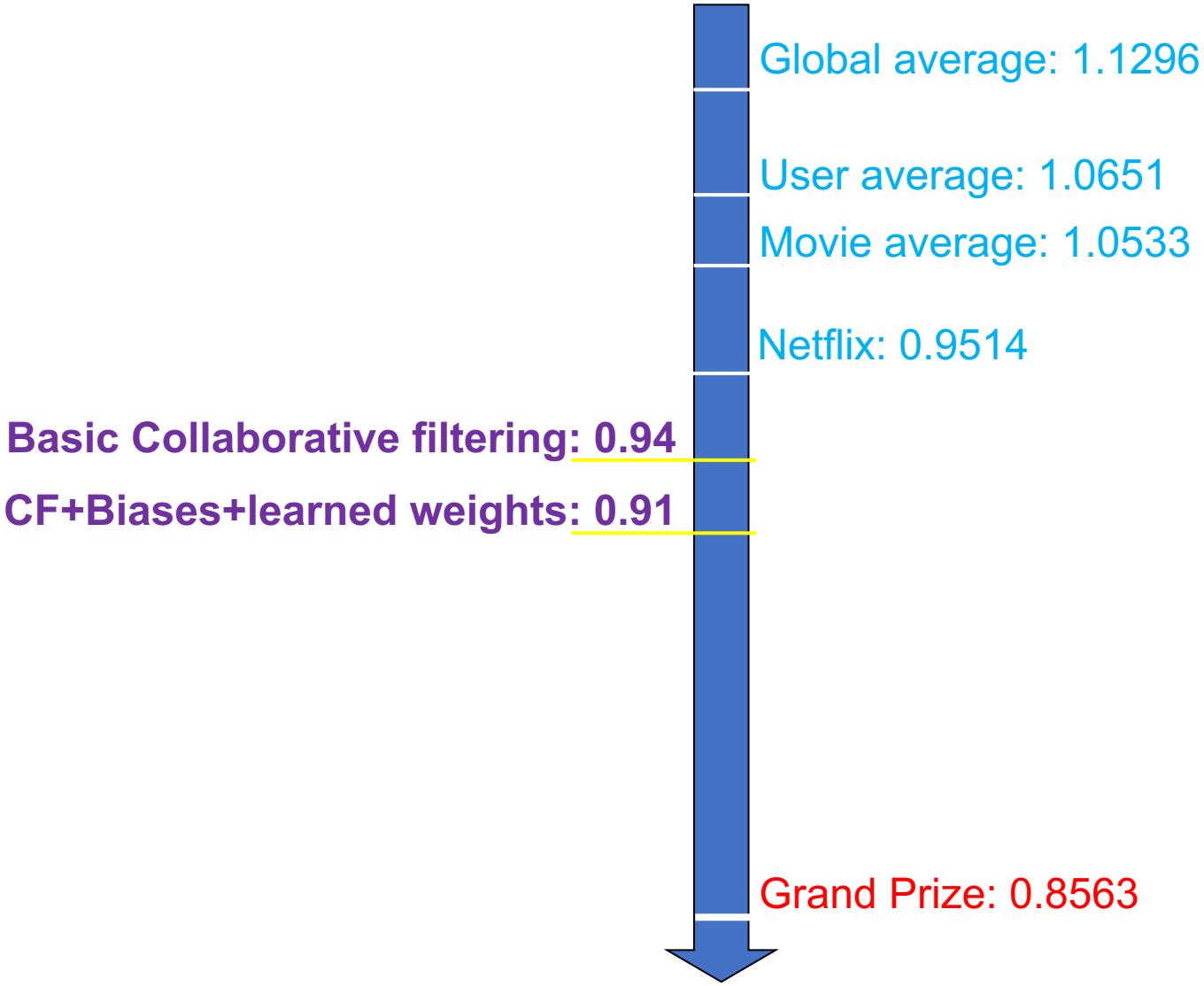
$$\textbf{RMSE} = \frac{1}{|\mathrm{R}|}\sqrt{\sum_{(i,x)\in R}(\hat{r}_{xi} - r_{xi})^2}$$

BellKor Recommender System

- **The winner of the Netflix Challenge!**
- **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



Performance of Various Methods



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**

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}

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

μ = overall mean rating
b_x = rating deviation of user x
= (*avg. rating of* user x) – μ
b_i = (*avg. rating of* movie i) – μ

Problems/Issues:

1) Similarity measures are "arbitrary"

2) Pairwise similarities neglect interdependencies among users

3) Taking a weighted average can be restricting

Solution: Instead of s_{ij} use w_{ij} that we estimate directly from data

Idea: Interpolation Weights w_{ij}

- Use a **weighted sum** rather than **weighted avg.**:

$$\widehat{r_{xi}} = b_{xi} + \sum_{j \in N(i;x)} w_{ij}\left(r_{xj} - b_{xj}\right)$$

- **A few notes:**
  - $\boldsymbol{N(i;x)}$ … set of movies rated by user $\boldsymbol{x}$ that are similar to movie $\boldsymbol{i}$
  - $\boldsymbol{w_{ij}}$ is the interpolation weight (some real number)
    - We allow: $\sum_{j \in N(i,x)} w_{ij} \neq 1$
  - $\boldsymbol{w_{ij}}$ models interaction between pairs of movies (it does not depend on user $\boldsymbol{x}$)

Idea: Interpolation Weights w_{ij}

- $\widehat{r_{xi}} = b_{xi} + \sum_{j \in N(i,x)} w_{ij}\left(r_{xj} - b_{xj}\right)$
- **How to set w_{ij}?**
  - Remember, error metric is:
    $\frac{1}{|R|}\sqrt{\sum_{(i,x)\in R}(\hat{r}_{xi} - r_{xi})^2}$ or equivalently **SSE:**
    $\boldsymbol{\sum_{(i,x)\in R}(\hat{r}_{xi} - r_{xi})^2}$
  - Find $\boldsymbol{w_{ij}}$ that minimize **SSE** on **training data!**
    - Models relationships between item ***i*** and its neighbors ***j***
  - $\boldsymbol{w_{ij}}$ can be **learned/estimated** based on ***x*** and all other users that rated ***i***

Recommendations via Optimization

- **Goal:** Make good recommendations
  - Quantify goodness using **RMSE:**
    Lower RMSE ⇒ better recommendations
  - Want to make good recommendations on items that user has not yet seen. Can't really do this!

  - **Let's build a system such that it works well on known (user, item) ratings**
    And **hope** the system will also predict well the **unknown ratings**

Recommendations via Optimization

- **Idea: Let's set values *w* such that they work well on known (user, item) ratings**

- **How to find such values *w*?**
- **Idea:** Define an objective function and solve the optimization problem

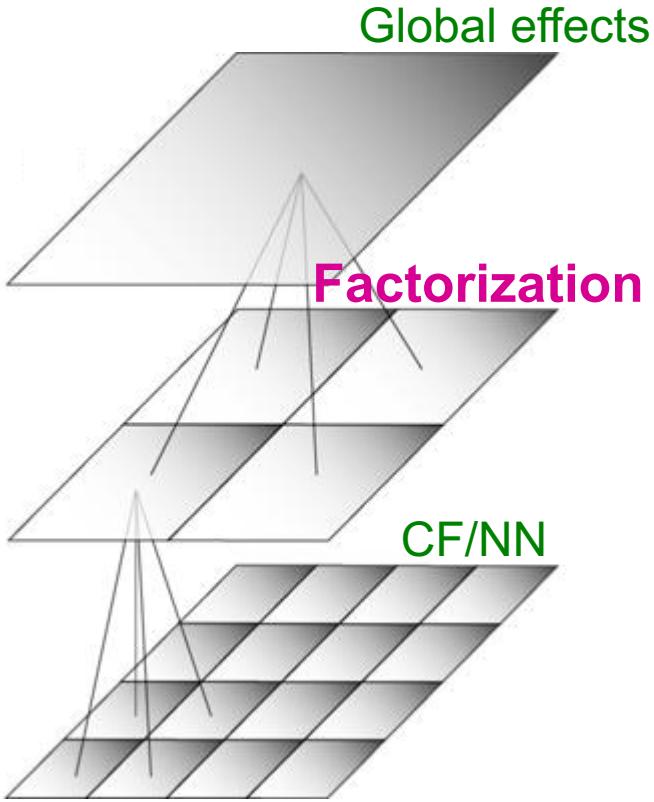
- Find $\boldsymbol{w_{ij}}$ that minimize **SSE** on **training data**!

$$J(w) = \sum_{x,i} \left(\underbrace{\left[b_{xi} + \sum_{j \in N(i;x)} w_{ij}\left(r_{xj} - b_{xj}\right) \right]}_{\text{Predicted rating}} - \underbrace{r_{xi}}_{\text{True rating}} \right)^2$$

- Think of ***w*** as a vector of numbers

Interpolation Weights

- **So far:** $\widehat{r_{xi}} = b_{xi} + \sum_{j \in N(i;x)} w_{ij}\left(r_{xj} - b_{xj}\right)$
  - Weights $\boldsymbol{w_{ij}}$ derived based on their role; **no use of an arbitrary similarity measure** ($\boldsymbol{w_{ij}} \neq \boldsymbol{s_{ij}}$)
  - Explicitly account for interrelationships among the neighboring movies



- **Next: Latent factor model**
  - Extract "regional" correlations

More model-based approaches

- Plethora of different techniques proposed in the last years, e.g.,
  - Matrix factorization techniques, statistics
    - singular value decomposition, principal component analysis
  - Association rule mining
    - compare: shopping basket analysis
  - Probabilistic models
    - clustering models, Bayesian networks, probabilistic Latent Semantic Analysis
  - Various other machine learning approaches
- Costs of pre-processing
  - Usually not discussed
  - Incremental updates possible?

Matrix Factorization

- Informally, the SVD theorem (Golub and Kahan 1965) states that a given matrix M can be decomposed into a product of three matrices as follows
  - where U and V are called *left* and *right singular vectors* and the values of the diagonal of Σ are called the *singular values*
- We can approximate the full matrix by observing only the most important features – those with the largest singular values
- In the example, we calculate U, V, and Σ (with the help of some linear algebra software) but retain only the three most important features by taking only the first three columns of U and V^T

Matrix Factorization

- "SVD" on Netflix data: $\mathbf{R} \approx \boldsymbol{Q} \cdot \boldsymbol{P}^{T}$

users

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	

R

≈

factors

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

Q

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

factors

$\boldsymbol{P}^{T}$

- For now let's assume we can approximate the rating matrix $\boldsymbol{R}$ as a product of "thin" $\boldsymbol{Q} \cdot \boldsymbol{P}^{T}$
  - $\boldsymbol{R}$ has missing entries but let's ignore that for now!
    - Basically, we will want the reconstruction error to be small on known ratings and we don't care about the values on the missing ones

Ratings as Products of Factors

- **How to estimate the missing rating of user *x* for item *i*?**

$$\hat{r}_{xi} = q_i \cdot p_x^T = \sum_f q_{if} \cdot p_{xf}$$

q_i = row i of Q
p_x = column x of P^T

users

items

1		3			5			5		4	
		5	4	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

factors

Q

•

users

factors

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

P^T

Matrix Factorization

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

Q

users

factors

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

P^T

- **SVD isn't defined when entries are missing!**
- **Use specialized methods to find *P*, *Q***
  - $\min_{P,Q} \sum_{(i,x)\in R} (r_{xi} - q_i \cdot p_x^T)^2$
  - **Note:**
    - We don't require cols of ***P, Q*** to be orthogonal/unit length
    - ***P, Q*** map users/movies to a latent space

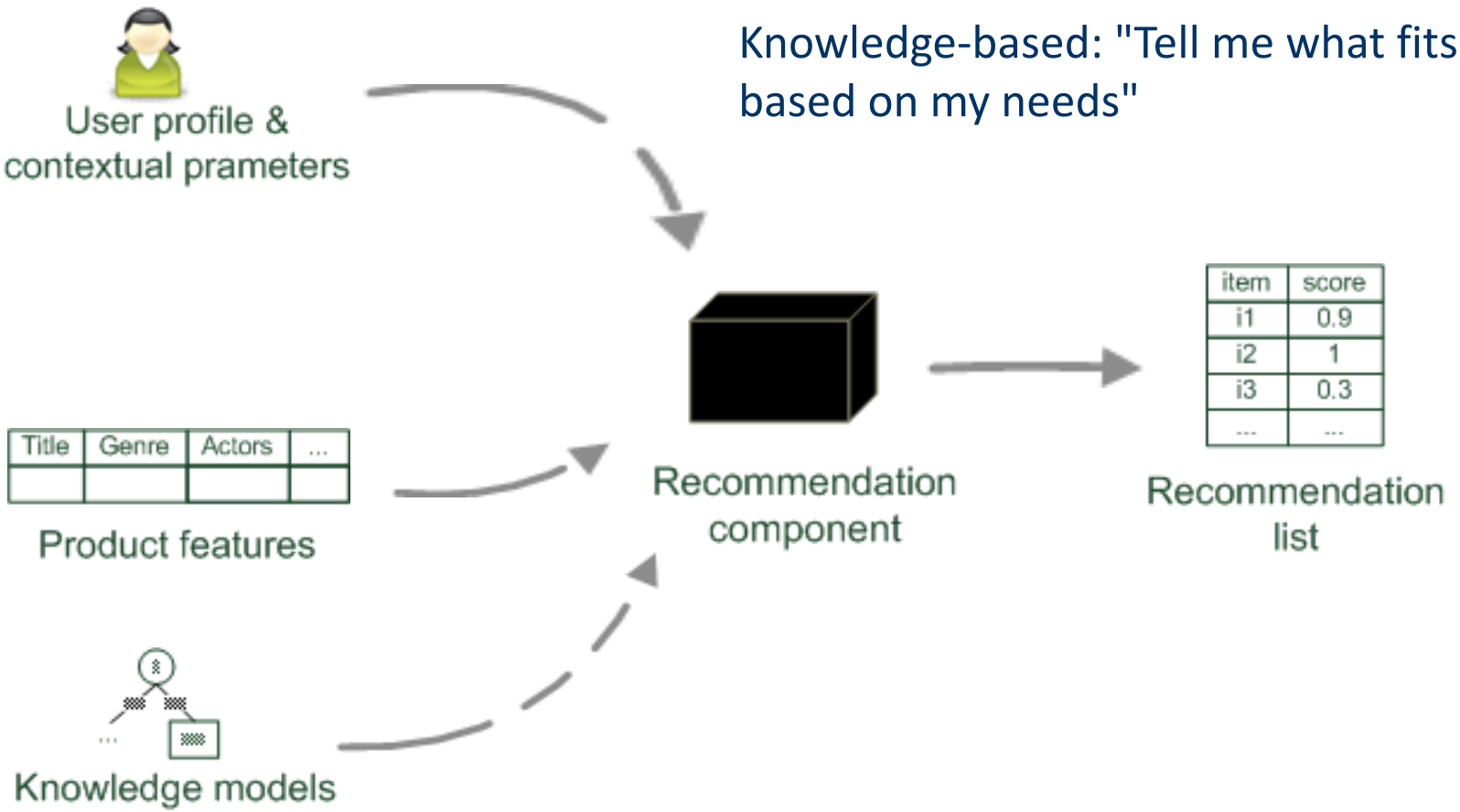
$$\hat{r}_{xi} = q_i \cdot p_x^T$$

Pros/Cons of Collaborative Filtering

- **+ Works for any kind of item**
  - No feature selection needed
- **- Cold Start:**
  - Need enough users in the system to find a match
- **- Sparsity:**
  - The user/ratings matrix is sparse
  - Hard to find users that have rated the same items
- **- First rater:**
  - Cannot recommend an item that has not been previously rated
  - New items, Esoteric items
- **- Popularity bias:**
  - Cannot recommend items to someone with unique taste
  - Tends to recommend popular items

Knowledge-Based Recommendation

Knowledge-based: "Tell me what fits based on my needs"



Why do we need knowledge-based recommendation?

- Products with low number of available ratings

- Time span plays an important role
  - Five-year-old ratings for computers
  - User lifestyle or family situation changes
- Customers want to define their requirements explicitly
  - “The color of the car should be black"

Knowledge-based Recommendation

- Constraint-based
  - based on explicitly defined set of recommendation rules
  - fulfill recommendation rules

- Case-based
  - based on different types of similarity measures
  - retrieve items that are similar to specified requirements

- Both approaches are similar in their conversational recommendation process
  - users specify the requirements
  - systems try to identify solutions
  - if no solution can be found, users change requirements

Hybrid Methods

- Implement two or more different recommenders and combine predictions
  - Perhaps using a linear model
- Add content-based methods to collaborative filtering
  - Item profiles for new item problem
  - Demographics to deal with new user problem