Item-Item Collaborative Filtering

- **So far: User-user collaborative filtering**
- **Another view: Item-item collaborative filtering**
  - 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

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

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

Item-Item CF (|N|=2)

華中科技大學
计算机科学与技术学院
School of Computer Science &Technology, HUST

users

movies

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

☐ - unknown rating ■ - rating between 1 to 5

Item-Item CF (|N|=2)

users

movies

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

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

Item-Item CF (|N|=2)

華中科技大學 计算机科学与技术学院 School of Computer Science &Technology, HUST

users

movies	12	11	10	9	8	7	6	**5**	4	3	2	1		sim(1,m)
		4		5			5	?		3		1	1	1.00
	3	1	2			4			4	5			2	-0.18
		5	3	4		3		2	1		4	2	<u>3</u>	<u>0.41</u>
		2			4			5		4	2		4	-0.1
	5	2					2	4	3	4			5	-0.31
		4			2			3		3		1	<u>6</u>	<u>0.59</u>

Neighbor selection:
Identify movies (N=2, so 2 movies)
similar to movie **1**, rated by user 5

Here we use Pearson correlation as similarity

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

Item-Item CF (|N|=2)

users

movies

12	11	10	9	8	7	6	5	4	3	2	1		sim(1,m)
	4		5			5	?		3		1	1	1.00
3	1	2			4			4	5			2	-0.18
	5	3	4		3		2	1		4	2	3	0.41
	2			4			5		4	2		4	-0.1
5	2					2	4	3	4			5	-0.31
	4			2			3		3		1	6	0.59

Compute similarity weights:

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

Item-Item CF (|N|=2)

users

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

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 (阿凡达)	LOTR (指环王)	Matrix (黑客帝国)	Pirates (加勒比海盗)
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 <u>item-item</u> often works better than user-user**
- **Why?** Items are simpler, users have multiple tastes

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

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 (人口统计学特征) to deal with new user problem

Key Problems

- **(1) Gathering "known" ratings for matrix**
  - How to collect the data in the utility matrix
- **(2) 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
- **(3) Evaluating extrapolation methods**
  - How to measure success/performance of recommendation methods

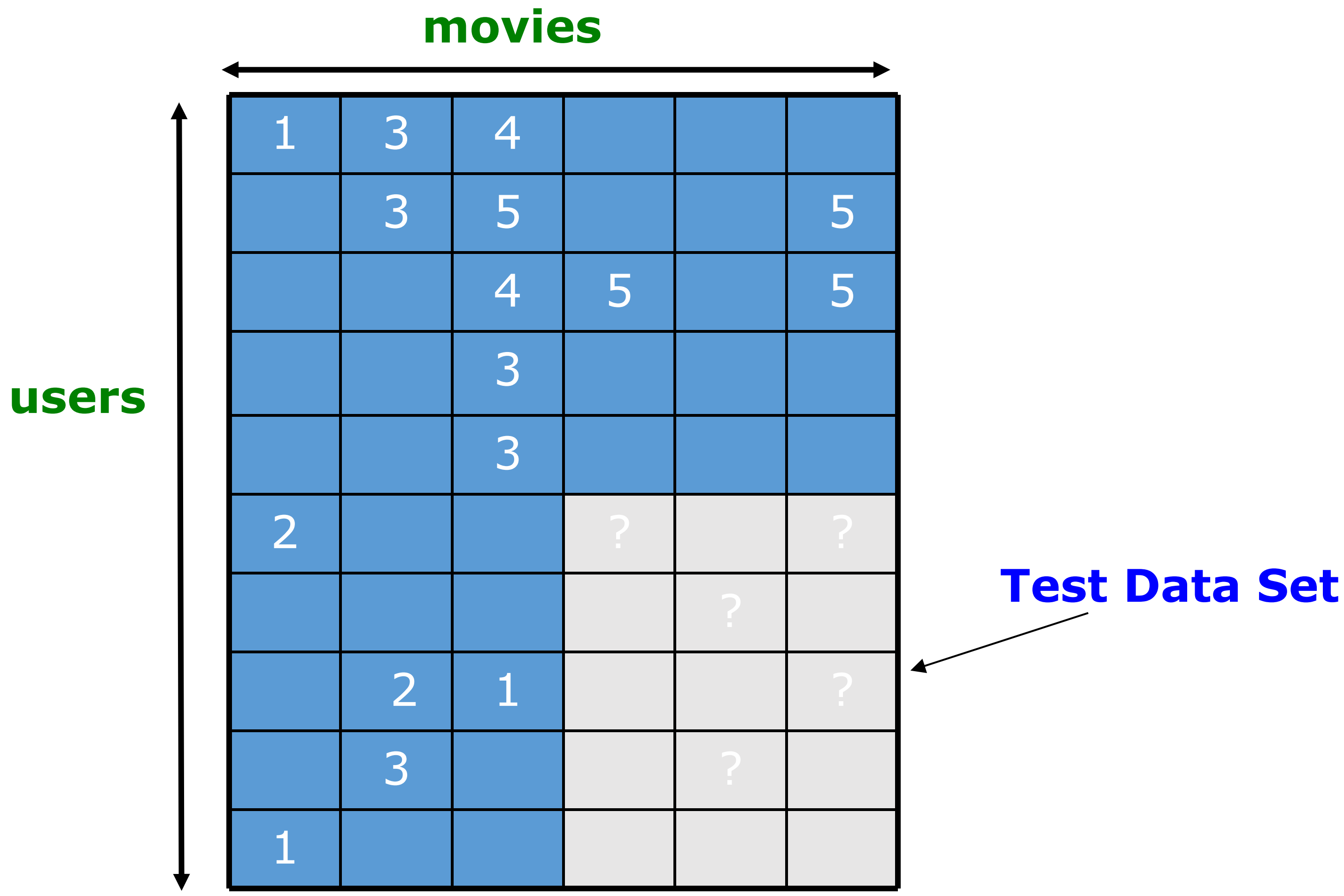
Evaluation

movies

users

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

Evaluation



Evaluating Predictions

Compare predictions with known ratings

- **Root-mean-square error** (RMSE, 均方根误差)
  - $\sqrt{\sum_{xi}(r_{xi}-r_{xi}^{*})^{2}}$ where $\boldsymbol{r_{xi}}$ is predicted, $\boldsymbol{r_{xi}^{*}}$ is the true rating of user $\boldsymbol{x}$ on item $\boldsymbol{i}$
- **Sum of square error** (SSE, 误差平方和)
  - $\sum_{xi}(r_{xi}-r_{xi}^{*})^{2}$ where $\boldsymbol{r_{xi}}$ is predicted, $\boldsymbol{r_{xi}^{*}}$ is the true rating of user $\boldsymbol{x}$ on item $\boldsymbol{i}$
- **Precision at top 10**:
  - % of those in top 10
- **Rank Correlation**:
  - Spearman’s *correlation* (斯皮尔曼等级相关系数) between system’s and user’s complete rankings

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$$

Evaluating Predictions

❑Another approach: 0/1 model

➢Coverage (覆盖率):

- Number of items/users for which system can make predictions
- 推荐系统能够推荐的物品占总物品的比例. 覆盖率高, 那么模型能够针对更多项产生推荐, 促进长尾效应的挖掘

➢Precision (精准度):

- Accuracy of predictions
- 推荐中的准确性，越高那么推荐系统越好

➢**Receiver operating characteristic** (ROC,受试者工作特征曲线)

- Tradeoff curve between false positives and false negatives
- false positives预测值为1，真实值为0
- false negatives预测值为0，真实值为1

Problems with Error Measures

- **Narrow focus on accuracy sometimes misses the point**
  - Prediction Diversity. e.g., HP1(哈利波特), then HP2, HP3
  - Prediction Context. e.g., car, but after buying car, no need to recommend
  - Order of predictions. e.g., MCU(漫威电影), Iron Man(钢铁侠) before Avengers(复仇者联盟)

- **In practice, we care only to predict high ratings:**
  - RMSE(均方根误差) might penalize a method that does well for high ratings and badly for others

Tip: Add Data

❑ Leverage all the data

- ➢ Don’ t try to reduce data size in an effort to make fancy algorithms work
- ➢ Simple methods on large data do best

❑ Add more data

- ➢ e.g., add IMDB data on genres

❑ More data beats better algorithms

http://anand.typepad.com/datawocky/2008/03/more-data-usual.html

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(k ·|X|)**
  - X … set of customers

- **How to do this?**
  - Clustering
  - Dimensionality reduction
  - Near-neighbor search in high dimensions (e.g., locality-sensitive hashing, LSH, 局部性敏感哈希)