Unsupervised Machine Learning

Implementing and Evaluating Recommendation Systems

Prof. Yannis Velegrakis

https://velgias.github.io

Collaborative Filtering: Complexity

- Expensive step is finding ***k*** most similar users (or items): **O(|U|)**
  - |U| = size of utility matrix
- Too expensive to do at runtime
  - Could pre-compute
  - Naïve pre-computation takes time **O(n ·|U|)**
    - Where n = number of users (items)
- **We already know how to do this!**
  - Near-neighbor search in high dimensions (**LSH**)
  - Clustering
  - Dimensionality reduction (coming soon!)

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 to deal with new user problem

Global Baseline Estimate

- Estimate Joe's rating for the movie *The Sixth Sense*
  - Problem: Joe has not rated any movie "similar" to *The Sixth Sense*

- Global Baseline approach
  - Mean movie rating: **3.7 stars**
  - *The Sixth Sense* is **0.5** stars above avg.
  - Joe rates **0.2** stars below avg.
  - **Baseline estimate: 3.7 + 0.5 – 0.2 = 4 stars**

Combining Global Baseline Estimate with CF

- Global Baseline estimate
  - Joe will give *The Sixth Sense* 4 stars
- Local neighborhood (CF/NN)
  - Joe didn't like related movie *Signs*
  - Rated it 1 star below his average rating
- Final estimate
  - Joe will rate *The Sixth Sense* 4 -1 = 3.5 stars

WRONG

CORRECT !!!

Evaluation

Matrix R

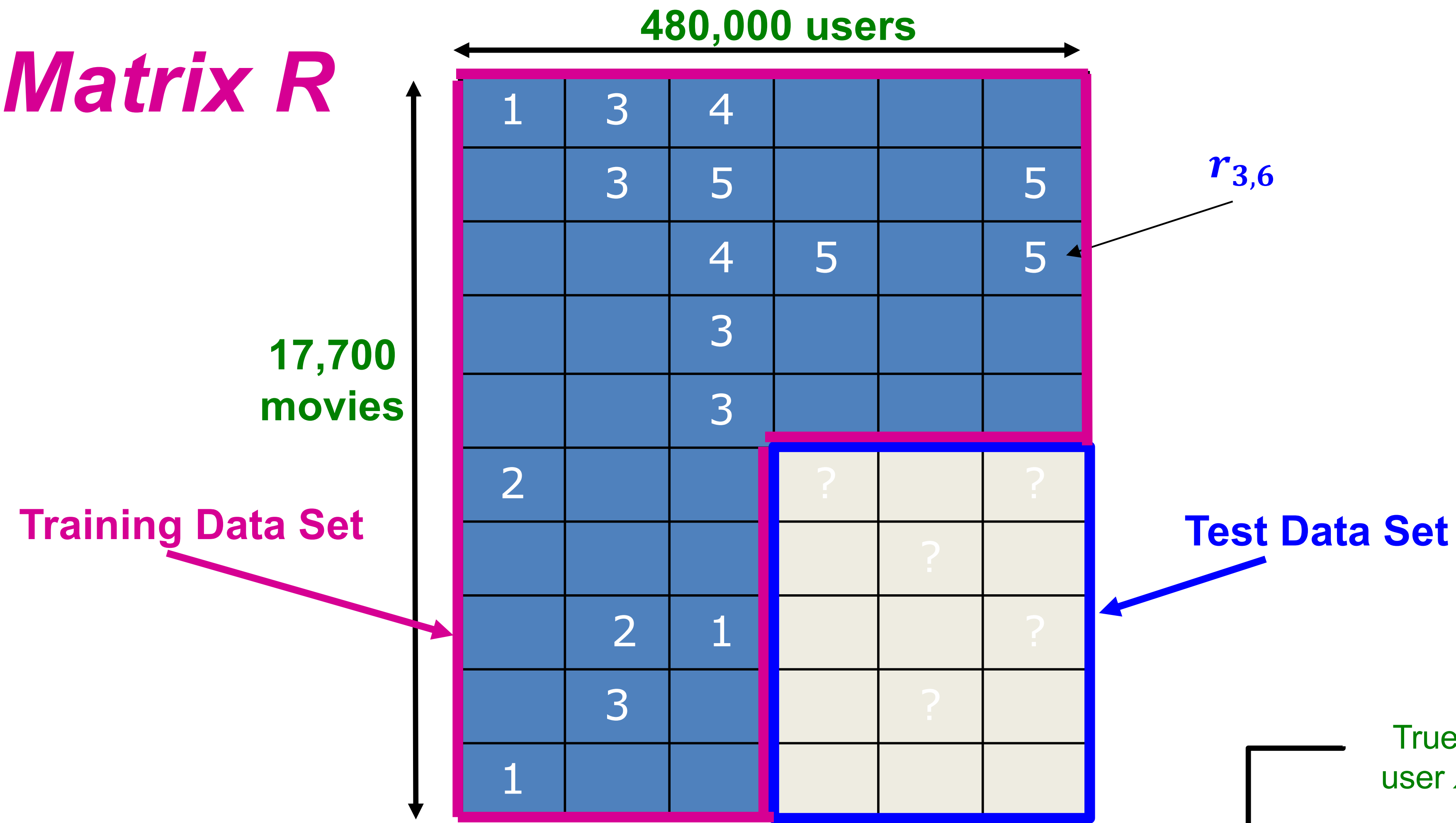
480,000 users

17,700 movies

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

Evaluation

Matrix R



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

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

Predicted rating

Problems with Error Measures

- **Narrow focus on accuracy sometimes misses the point**
  - Prediction Diversity
  - Prediction Context
  - Order of predictions
- **In practice, we care only to predict high ratings:**
  - RMSE might penalize a method that does well for high ratings and badly for others

Collaborative Filtering: Complexity

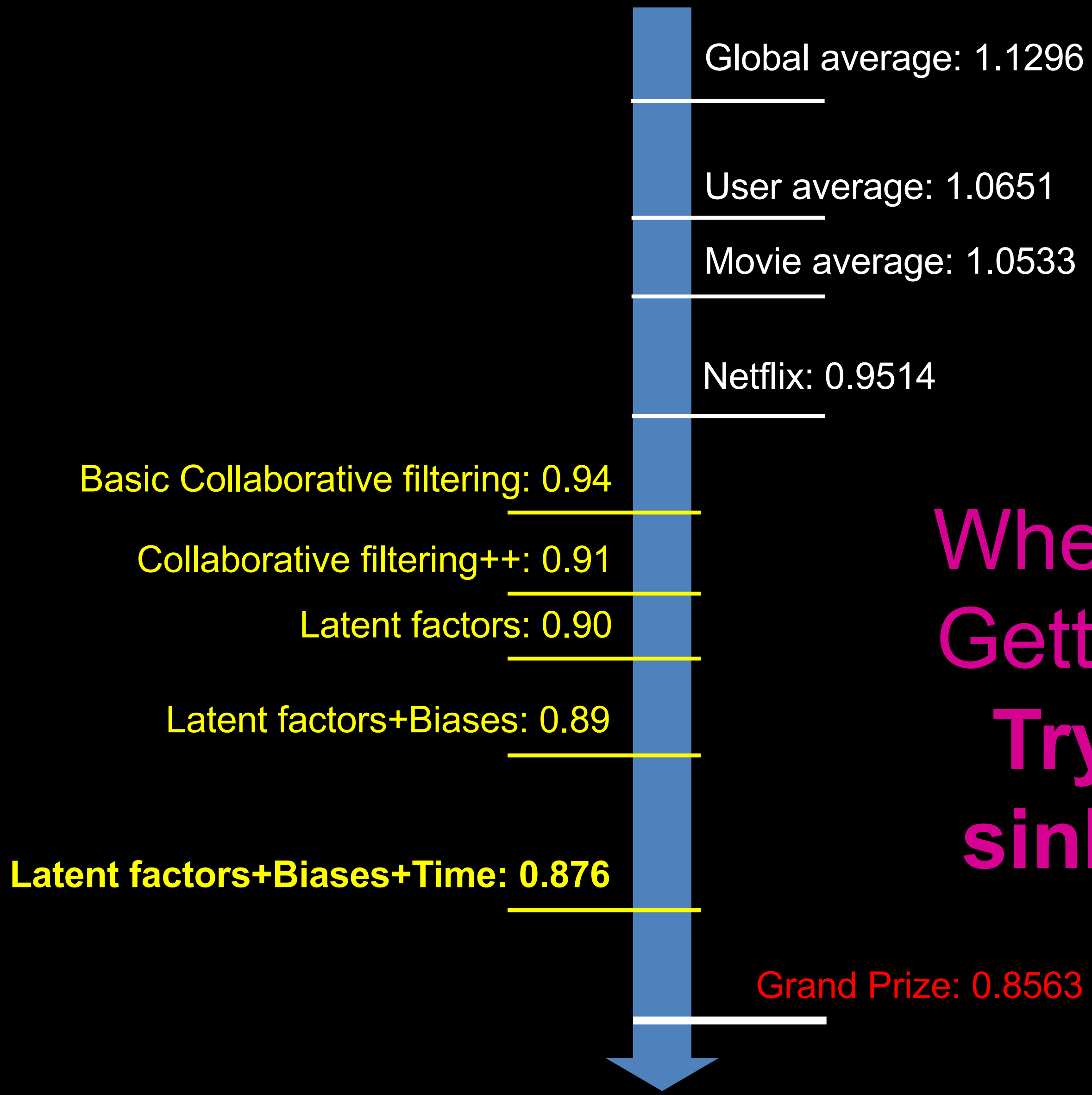
- Expensive step is finding ***k*** most similar customers: **$O(|X|)$**
- **Too expensive to do at runtime**
  - Could pre-compute
- Naïve pre-computation takes time **$O(k \cdot |X|)$**
  - X … set of customers
- **We already know how to do this!**
  - Near-neighbor search in high dimensions (**LSH**)
  - Clustering
  - Dimensionality reduction

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

Performance of Various Methods



When no prize…☹
Getting desperate.
Try a "kitchen sink" approach!

Rank aggregation

[Fagin, PODS 1996]

- Rank aggregation is the problem of combining several ranked lists of objects in a robust way to produce a single consensus ranking of the objects

Candidate	Candidate	Candidate	Candidate	Candidate
1	2	4	5	3
2	4	2	1	5
3	5	5	3	1
4	1	3	4	2
5	3	1	2	4
Judge 1	**Judge 2**	**Judge 3**	**Judge 4**	**Judge 5**

- What is the overall ranking?
- Who is the best candidate?