

Recommender Systems

| | |
|-------------------|---|
| Class | Algorithmic Methods of Data Mining |
| Program | M. Sc. Data Science |
| University | Sapienza University of Rome |
| Semester | Fall 2016 |
| Lecturer | Ioannis Chatzigiannakis |
| Slides | Carlos Castillo http://chato.cl/ |

Sources:

- Ricci, Rokach and Shapira: Introduction to Recommender Systems Handbook [[link](#)]
- Bobadilla et al. Survey 2013 [[link](#)]
- Xavier Amatriain 2014 tutorial on rec systems [[link](#)]
- Ido Guy 2011 tutorial on social rec systems [[link](#)]
- Alex Smola's tutorial on recommender systems [[link](#)]

Why recommender systems?

Definition

Recommender systems are software tools and techniques providing suggestions for items to be of use to a user.

User-based recommendations

These recommendations are based on [items you own](#) and more.

view: [All](#) | [New Releases](#) | [Coming Soon](#)

1.



[Short and Sweet](#)

by Dan Lepard (29 Sep 2011)

Average Customer Review: (92)

In stock

RRP: £25.00

Price: £14.00

[39 used & new from £7.91](#)

Add to Basket

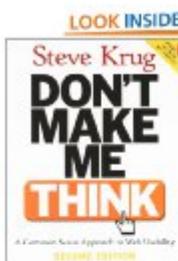
Add to Wish List



Rate this item

Recommended because you purchased **Mary Berry's Baking Bible** and more (Fix this)

2.



[Don't Make Me Think!: A Common Sense Approach to Web Usability](#)

by Steve Krug (18 Aug 2005)

Average Customer Review: (99)

In stock

RRP: £24.99

Price: £17.49

[73 used & new from £8.50](#)

Add to Basket

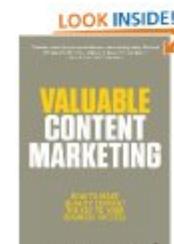
Add to Wish List



Rate this item

Recommended because you purchased **Content Strategy for the Web (Voices That Matter)** (Fix this)

3.



[Valuable Content Marketing: How to Make Quality Content the Key to Your Business Success](#)

by Sonja Jefferson (3 Jan 2013)

Average Customer Review: (24)

In stock

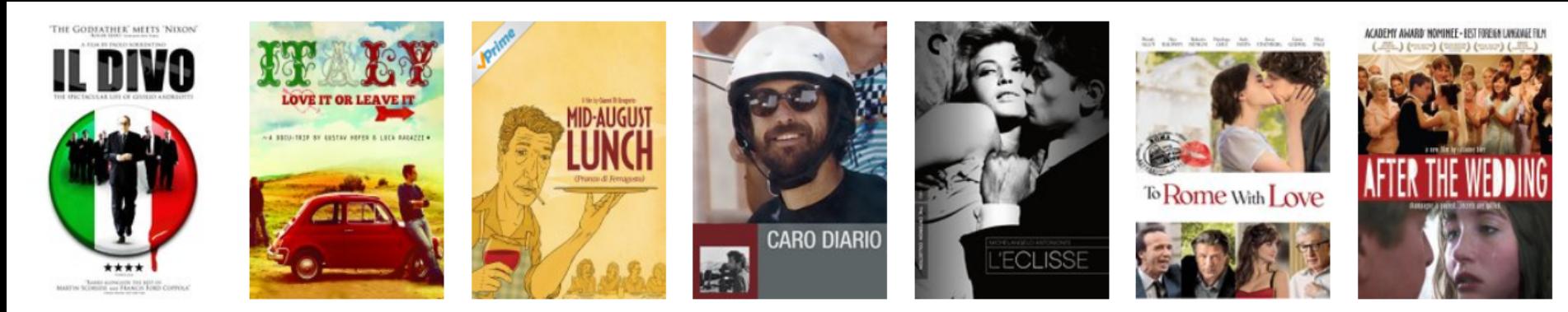
RRP: £19.99

Price: £17.59

[55 used & new from £12.40](#)

Add to Basket

Add to Wish List



Assumptions

- Users rely on recommendations
- Users lack sufficient personal expertise
- Number of items is very large
 - e.g. around 10^{10} books in Amazon
- Recommendations need to be personalized

Amazon as of December 2015

Paperback (30,020,393)

Hardcover (11,107,514)

Kindle Edition (2,861,600)

Audible Audio Edition (87,578)

Printed Access Code (30,338)

Digital Access Code (4)

Loose Leaf (94,891)

Audio CD (433,467)

Board Book (168,736)

Who uses recommender systems?

- Retailers and e-commerce in general
 - Amazon, Netflix, etc.
- Service sites, e.g. travel sites
- Media organizations
- Dating apps
- ...

Why?

- Increase number of items sold
 - 2/3 of Netflix watched are recommendations
 - 1/3 of Amazon sales are from recommendations
 - ...

Why? (cont.)

- Sell more diverse items
- Increase user satisfaction
 - Users enjoy the recommendations
- Increase user fidelity
 - Users feel recognized (but not creeped out)
- Understand users (see next slides)

By-products

- Recommendations generate by-products
- Recommending requires understanding users and items, which is valuable by itself
- Some recommender systems are very good at this (e.g. factorization methods)
- Automatically identify marketing profiles
- Describe users to better understand them

The recommender system problem

Estimate the utility for a user of an item for which the user has not expressed utility

What information can be used?

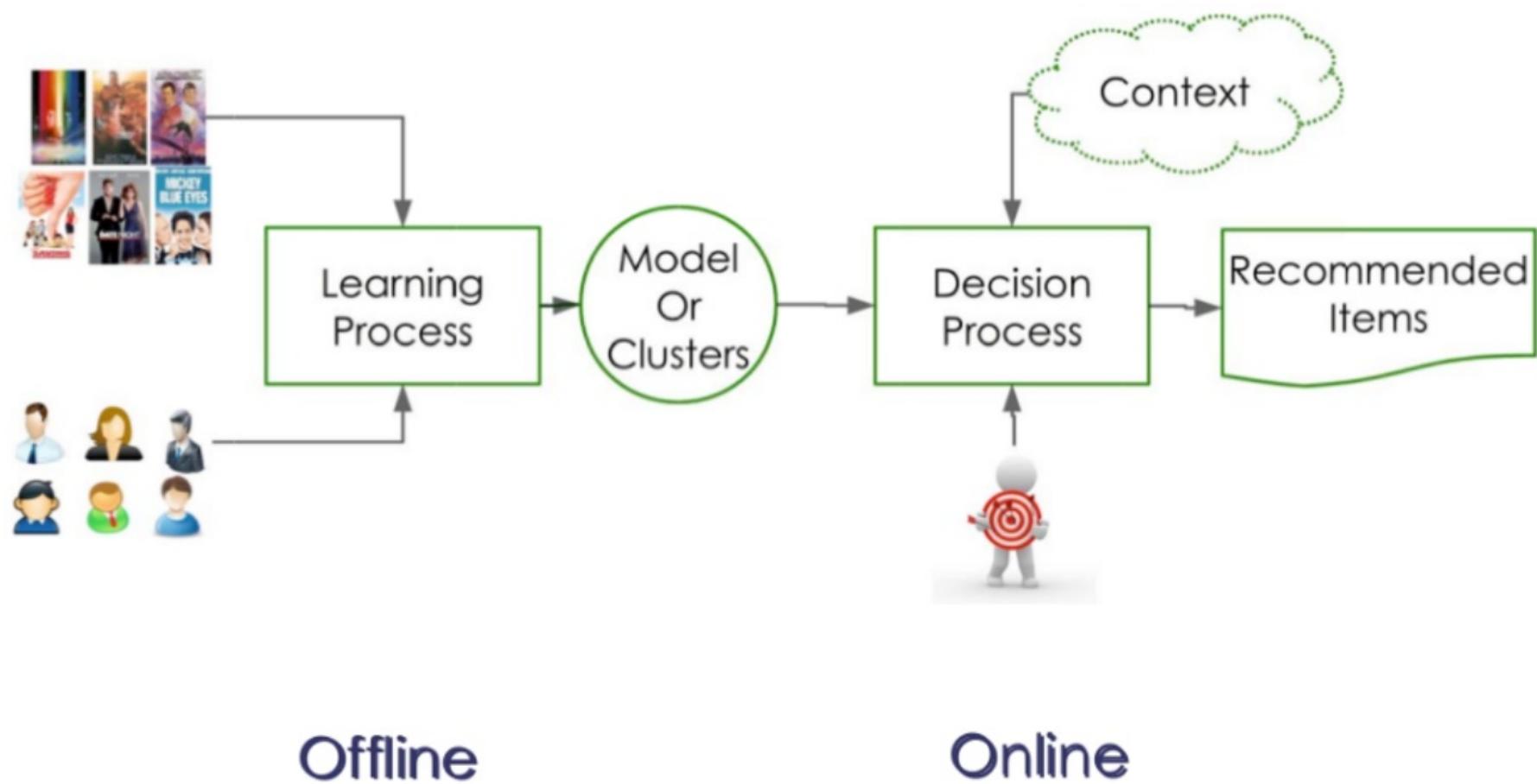
Types of problem

- Find some good items (most common)
- Find all good items
- Annotate in context (why I would like this)
- Recommend a sequence (e.g. tour of a city)
- Recommend a bundle (camera+lens+bag)
- Support browsing (seek longer session)
- ...

Data sources

- Items, Users
 - Structured attributes, semi-structured or unstructured descriptions
- Transactions
 - Appraisals
 - Numerical ratings (e.g. 1-5) 
 - Binary ratings (like/dislike)
 - Unary ratings (like/don't know)
 - Sales
 - Tags/descriptions/reviews

Recommender system process



Why is part of the processing done offline?

Aspects of this process

- Data preparation
 - Normalization, removal of outliers, feature selection, dimensionality reduction, ...
- Data mining
 - Clustering, classification, rule generation, ...
- Post-processing
 - Visualization, interpretation, meta-mining, ...

Desiderata for recommender system

- Must inspire trust
- Must convince users to try the items
- Must offer a good combination of novelty, coverage, and precision
- Must have a somewhat transparent logic
- Must be user-tunable

We're recommending **Foreign Films in Video** because



3 Idiots
Aamir Khan

Don't use for recommendations



Kaminey
Shahid Kapoor

Don't use for recommendations



My Name Is Khan
Shah Rukh Khan

Don't use for recommendations

X

Human factors

- Advanced systems are **conversational**
- Transparency and scrutability
 - Explain users how the system works
 - Allow users to tell the system it is wrong
- Help users make a good decision
- Convince users in a persuasive manner
- Increase enjoyment to users
- Provide serendipity

Serendipity

- “An aptitude for making desirable discoveries by accident”
- Don't recommend items the user already knows
- Delight users by expanding their taste
 - But still recommend them something somewhat familiar
- It can be controlled by specific parameters

High-level approaches

- Memory-based
 - Use data from the past in a somewhat “raw” form
- Model-based
 - Use models built from data from the past

Approaches

- Collaborative filtering
- Content-based (item features)
- Knowledge-based (expert system)
- Personalized learning to rank
 - Estimate ranking function
- Demographic
- Social/community based
 - Based on connections
- Hybrid (combination of some of the above)

Collaborative filtering

Collaborative Filtering approach

- User has seen/liked certain items
- Community has seen/liked certain items
- Recommend to users items similar to the ones they have seen/liked
 - Based on finding similar users
 - Based on finding similar items

Algorithmic elements

- M users and N items
- Transaction matrix $R_{M \times N}$
- Active user
- Method to compute similarity of users
- Method to sample high-similarity users
- Method to aggregate their ratings on an item

k nearest users algorithm

- Compute common elements with other users
- Compute distance between rating vectors
- Pick top 3 most similar users
- For every unrated item
 - Average rating of 3 most similar users
- Recommend highest score unrated items

Ratings data



ITEMS 

USERS



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | | 5 | | | | 2 | | 0 | 1 | 2 |
| 2 | 5 | | 1 | | 5 | | 4 | | | 5 | 3 | 2 |
| 3 | | 5 | 5 | | | 1 | | 3 | | | 2 | 4 |
| 4 | | 3 | | | 5 | 4 | | | | 5 | | 1 |
| 5 | | | | 2 | 5 | 4 | | | 4 | 5 | | |
| 6 | 4 | | 3 | | | 1 | | | 4 | | 2 | |
| 7 | | 4 | 1 | | 4 | | 5 | | | | 4 | 1 |

Try it! Generate recommendations

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



$$\text{dist}(i, j) = \frac{\sum_{\alpha \in U_i \cap U_j} |r_i(\alpha) - r_j(\alpha)|}{|U_i \cap U_j|}$$

Given the **red** user

Determine 3 nearest users

Average their ratings on unrated items

Pick top 3 unrated elements

Compute user intersection size

| | ITEMS | | | | | | | | | | | | | |
|---|-------|---|---|---|---|---|---|---|---|----|----|----|-----------------------------|--|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | | |
| 1 | 1 | | | 5 | | | | 2 | | 3 | | | $ U_i \cap U_j \sim(i, j)$ | |
| 2 | 5 | | 1 | | 5 | | 4 | | 5 | 3 | 2 | | 0 | |
| 3 | | 5 | 5 | | 1 | | 3 | | | 2 | 4 | | 3 | |
| 4 | 3 | | | 5 | 4 | | | 5 | | 1 | | | 3 | |
| 5 | | | | 2 | 5 | 4 | | 4 | 5 | | | | 3 | |
| 6 | 4 | | 3 | | | 1 | | 4 | | 2 | | | 1 | |
| 7 | | 4 | 1 | | 4 | 5 | | | | 4 | 1 | | 3 | |

Compute user similarity

| | ITEMS | | | | | | | | | | | | | |
|---|-------|---|---|---|---|---|---|---|---|----|----|----|--------------------------------|------|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | | |
| 1 | 1 | | | 5 | | | | 2 | | 3 | | | $ U_i \cap U_j $ sim(i, j) | |
| 2 | 5 | | 1 | | 5 | | 4 | | | 5 | 3 | 2 | 0 | |
| 3 | | 5 | 5 | | 1 | | 3 | | | 2 | 4 | | 3 | 0.33 |
| 4 | 3 | | | 5 | 4 | | | 5 | | 1 | | | 3 | 2.67 |
| 5 | | | 2 | 5 | 4 | | | 4 | 5 | | | | 3 | 0.00 |
| 6 | 4 | | 3 | | 1 | | | 4 | | 2 | | | 1 | 3.00 |
| 7 | | 4 | 1 | | 4 | 5 | | | | 4 | 1 | | 3 | 0.67 |

Pick top-3 most similar

| | | ITEMS | | | | | | | | | | | | | | |
|--|--|-------|---|---|---|---|---|---|---|---|----|----|----|------------------|--------------------|--|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | | | |
| | | 2 | 5 | 1 | 5 | 4 | | | | 5 | 3 | 2 | | $ U_i \cap U_j $ | $\text{sim}(i, j)$ | |
| | | 4 | 3 | 5 | 4 | | | | 5 | 1 | | | | | | |
| | | 5 | | 2 | 5 | 4 | | | 4 | 5 | | | | | | |
| | | 7 | | 4 | 1 | | 4 | 5 | | | 4 | 1 | | | | |

Estimate unrated items

| | | ITEMS | | | | | | | | | | | | | | |
|--|--|-------|-----|---|-----|-----|---|---|-----|---|-----|----|-----|---|------------------|--------------------|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | | | |
| | | 2 | 5 | 1 | 5 | 4 | | | | 5 | 3 | 2 | | | $ U_i \cap U_j $ | $\text{sim}(i, j)$ |
| | | 4 | 5.0 | 3 | 1.0 | 2.0 | 5 | 4 | 4.5 | - | 4.0 | 5 | 3.5 | 1 | | |
| | | 5 | | | 2 | 5 | 4 | | | 4 | 5 | | | | 3 | 0.00 |
| | | 7 | | 4 | 1 | | 4 | 5 | | | | 4 | 1 | | 3 | 0.67 |

Recommend top-3 estimated

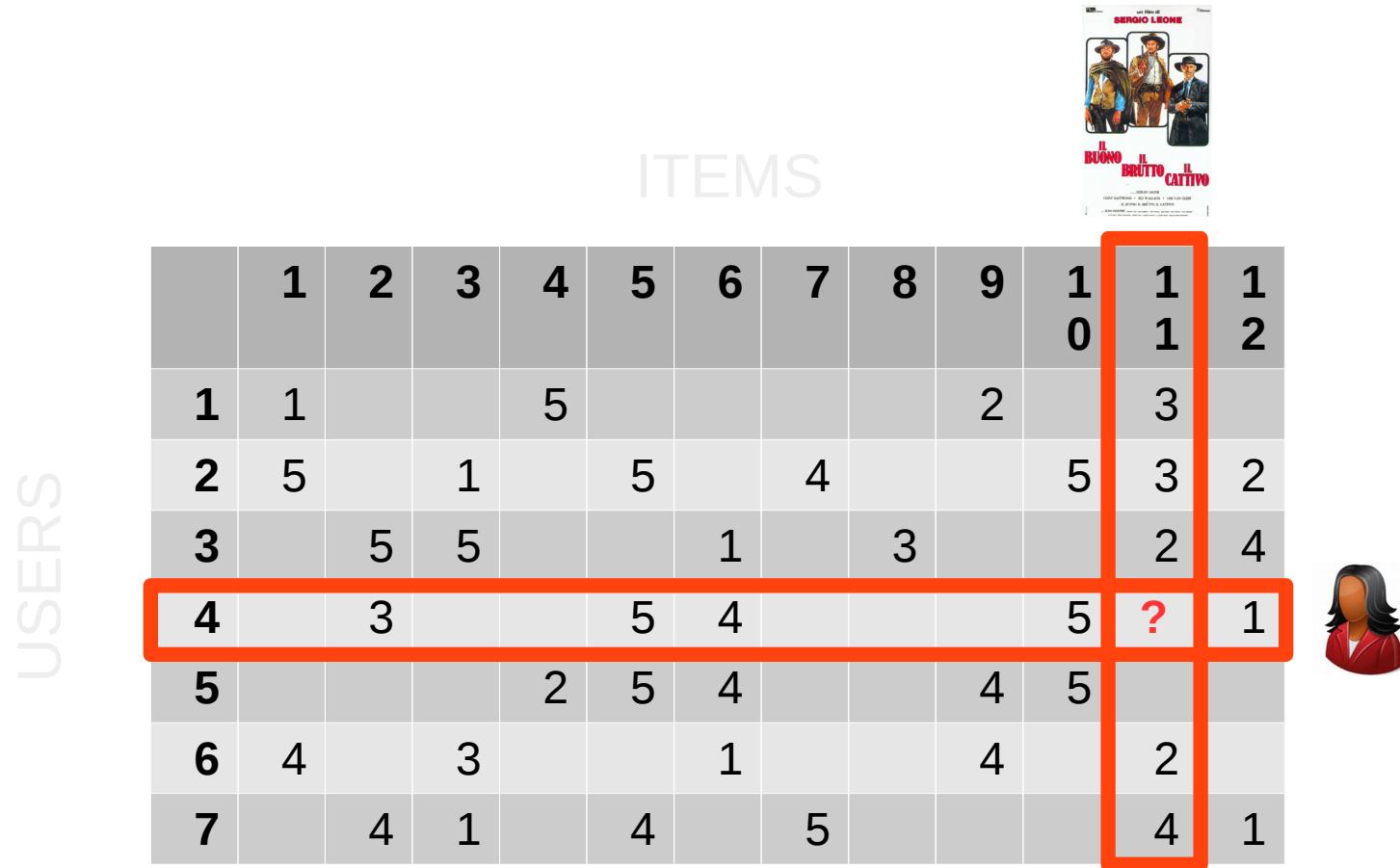
ITEMS

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | |
|--|---|-----|---|-----|-----|---|---|-----|---|-----|----|-----|-----------------------------|
| | 2 | 5 | | 1 | | 5 | | 4 | | 5 | 3 | 2 | $ U_i \cap U_j \sim(i, j)$ |
| | 4 | 5.0 | 3 | 1.0 | 2.0 | 5 | 4 | 4.5 | - | 4.0 | 5 | 3.5 | 1 |
| | 5 | | | 2 | 5 | 4 | | | 4 | 5 | | | |
| | 7 | | 4 | 1 | | 4 | | 5 | | | 4 | 1 | |

Improvements?

- *How would you improve the algorithm?*
- *How would you provide explanations?*

Item-based collaborative filtering



- Would user 4 like item 11?

Item-based collaborative filtering



- Compute pair-wise similarities to target item

Item-based collaborative filtering



- Pick k most similar items

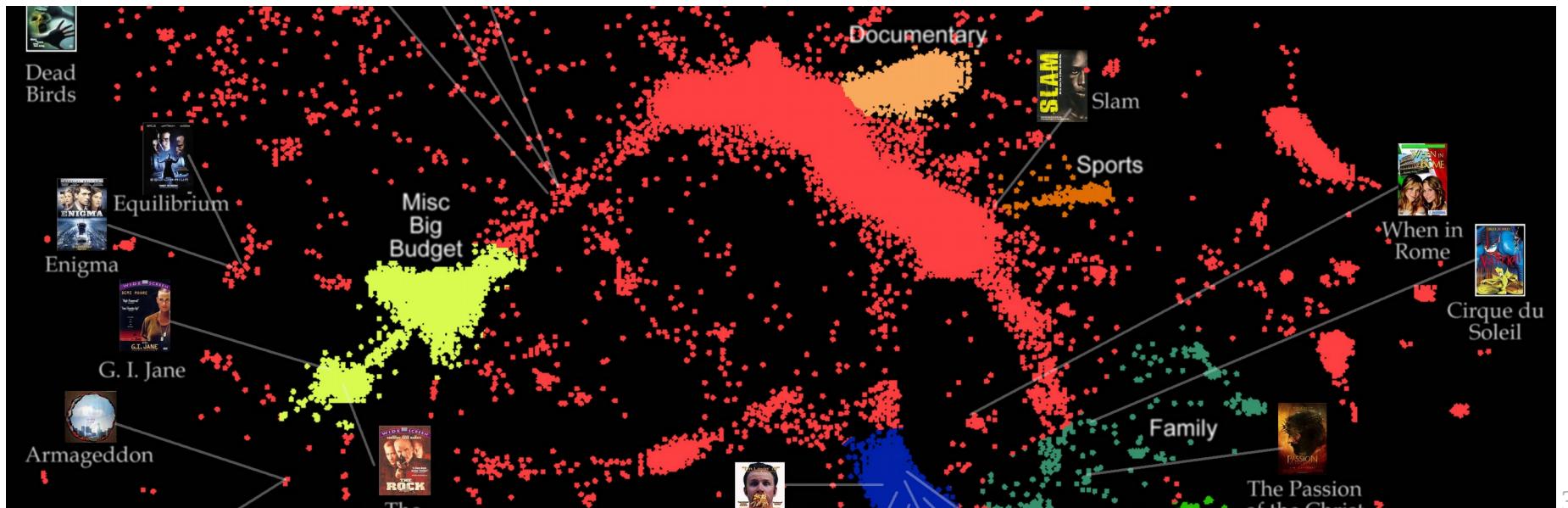
Item-based collaborative filtering



- Average ratings of target user on item

Performance implications

- Similarity between users is uncovered slowly
- Similarity between items is supposedly static
 - Can be precomputed!
- Item-based clusters can also be precomputed



37

[source]

Weaknesses

- Assumes standardized products
 - E.g. a touristic destination at any time of the year and under any circumstance is the same item
- Does not take into account context
- Requires a relatively large number of transactions to yield reasonable results

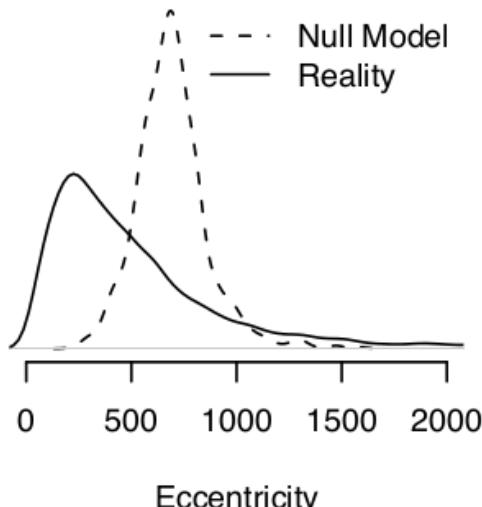
Cold-start problem

- What to do with a new item?
- What to do with a new user?

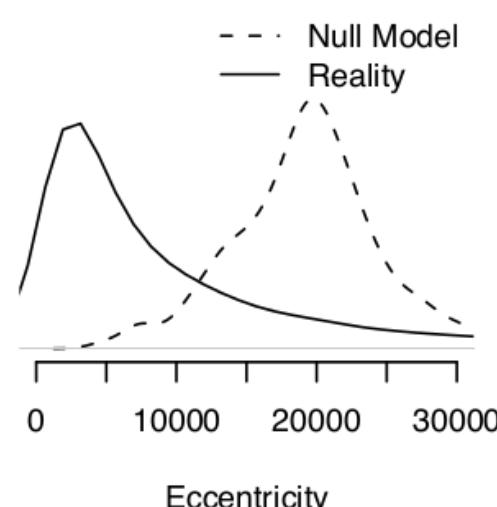
Assumptions

- Collaborative filtering assumes the following:
 - We take recommendations from friends
 - Friends have similar tastes
 - A person who have similar tastes to you, could be your friend
 - Discover people with similar tastes, use them as friends
- BUT, people's tastes are complex!

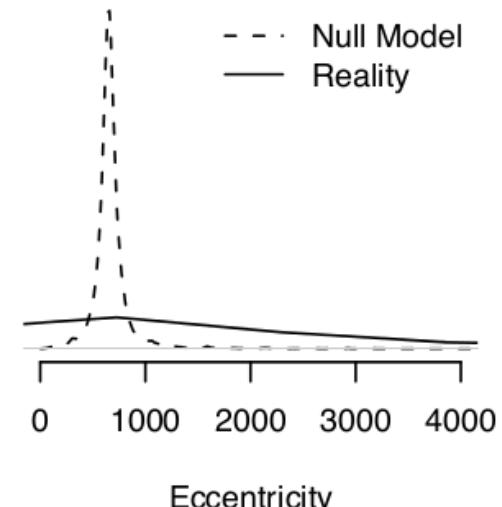
Ordinary people and extraordinary tastes



(a) Netflix



(b) Yahoo! Music



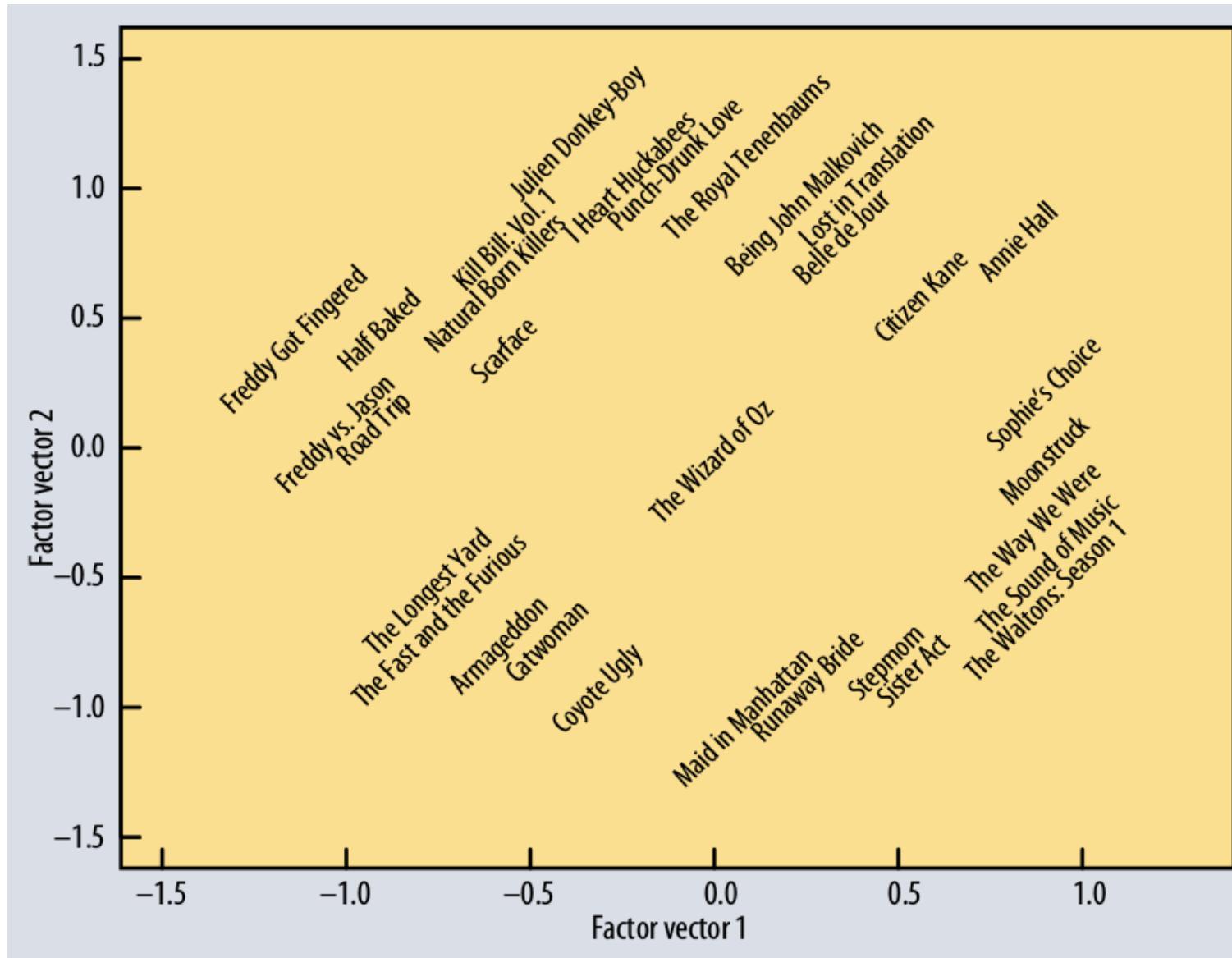
(c) Web Browsing (Nielsen)

Distribution of user eccentricity: the median rank of consumed items.

In the null model, users select items proportional to item popularity

Matrix factorization approaches

2D projection of interests



SVD approach

$$R_{n \times m} = U_{n \times f} S_{f \times f} V_{f \times m}$$

- R is the matrix of ratings
 - n users, m items
- U is a user-factor matrix
- S is a diagonal matrix, strength of each factor
- V is a factor-item matrix
- Matrices USV can be computed used an approximate SVD method

General factorization approach

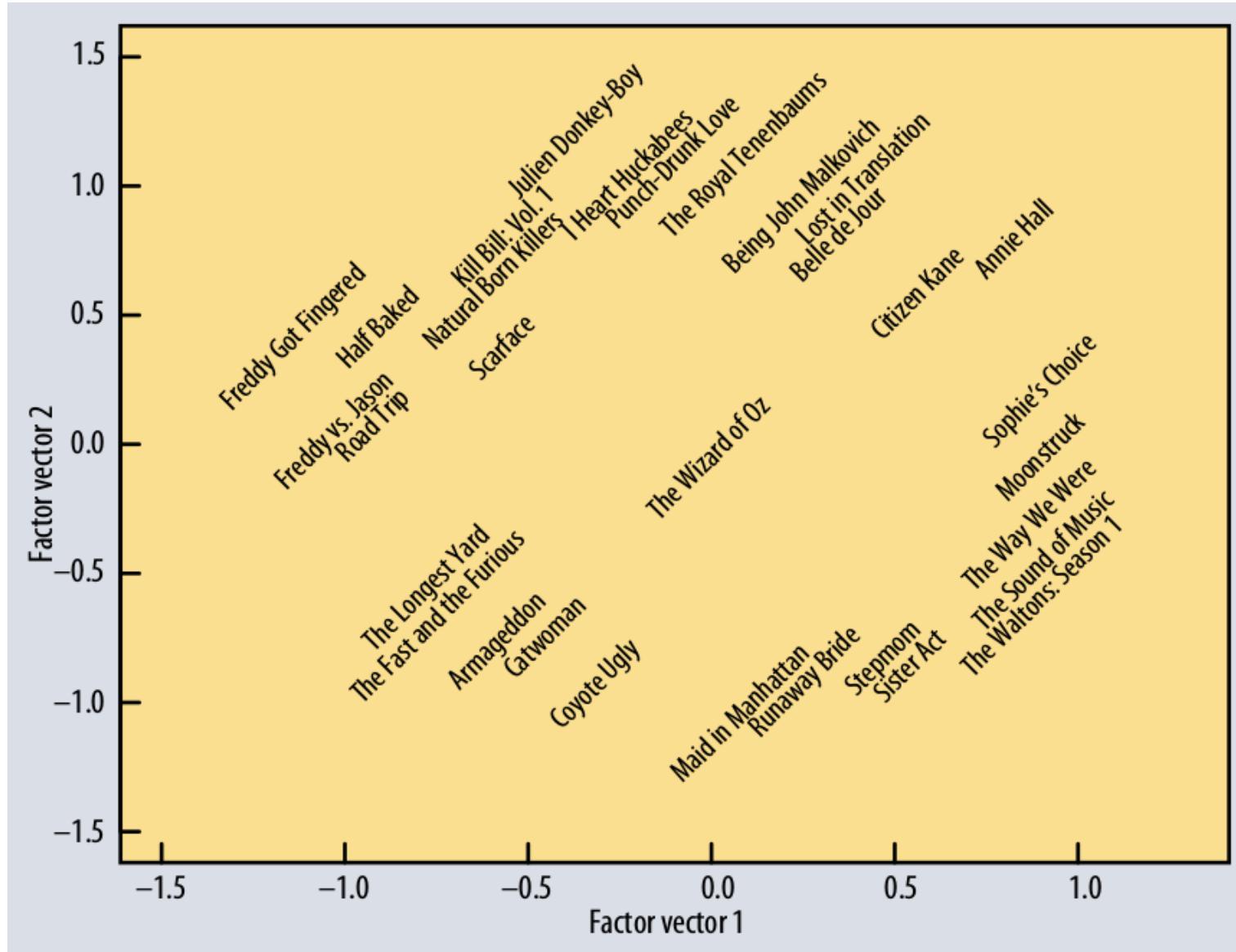
$$R_{n \times m} = P_{n \times f} Q_{f \times m}$$

- R is the matrix of ratings
 - n users, m items
- P is a user-factor matrix
- Q is a factor-item matrix

(Sometimes we force P, Q to be non-negative:
factors are easier to interpret!)

$$R_{n \times m} = P_{n \times f} Q_{f \times m}$$

What is this plot?



Computing expected ratings

- Given:
 - user vector $p_u \in \mathbb{R}^f$
 - item vector $q_i \in \mathbb{R}^f$
- Expected rating is $\hat{r}_{ui} = \langle p_u, q_i \rangle$

Model directly observed ratings

$$\min_{q \cdot, p \cdot} \sum_{(u,i) \in R_o} (r_{ui} - \langle p_u, q_i \rangle)^2 + \lambda(||q_i||^2 + ||p_u||^2)$$

- R_o are the observed ratings
- We want to minimize a reconstruction error
- Second term avoids over-fitting
 - Parameter λ found by cross-validation
- Two basic optimization methods

1. Stochastic gradient descend

$$\min_{q \cdot, p \cdot} \sum_{(u,i) \in R_o} (r_{ui} - \langle p_u, q_i \rangle)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

- Compute reconstruction error

$$e_{ui} = r_{ui} - \langle p_u, q_i \rangle$$

- Update in opposite direction to gradient

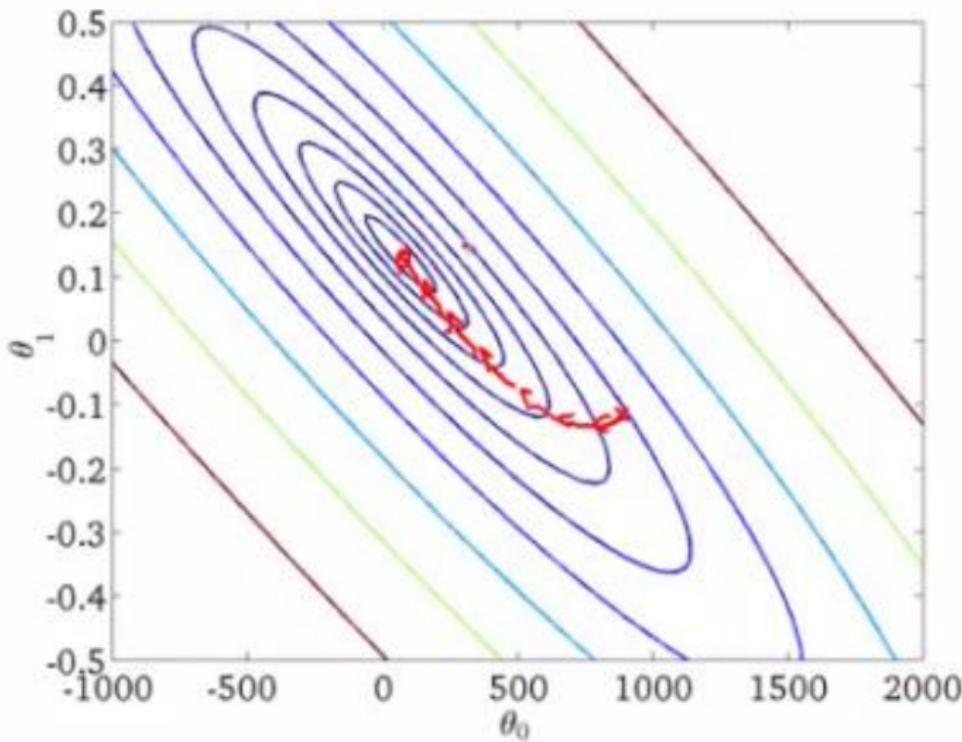
$$p_u \leftarrow p_u - \eta_t q_i e_{ui} - \eta_t \lambda p_u$$

$$q_i \leftarrow q_i - \eta_t p_u e_{ui} - \eta_t \lambda q_i$$



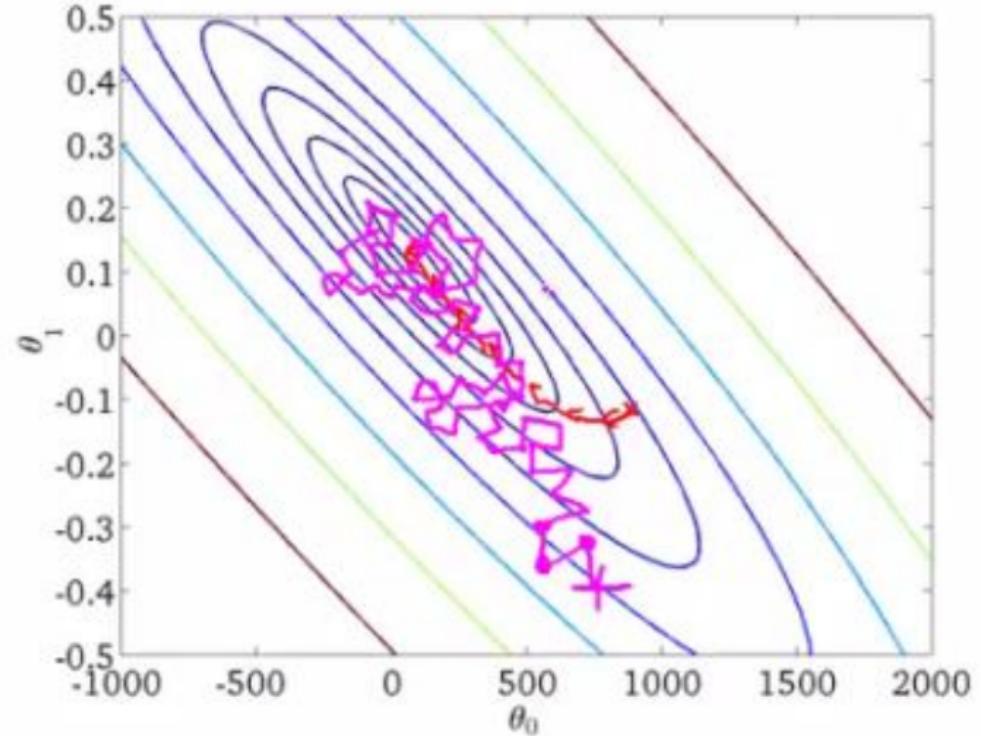
learning speed

Illustration: batch gradient descent vs stochastic gradient descent



Batch: gradient

$$x \leftarrow x - \eta \nabla F(x)$$



Stochastic: single-example gradient

$$x \leftarrow x - \eta \nabla F_i(x)$$

[source]

A simpler example of gradient descent

Fit a set of n two-dimensional data points (x_i, y_i) with a line $L(x) = w_1 + w_2 x$, means minimizing:

$$F(w) = \sum_{i=1}^n (L(x_i) - y_i)^2 = \sum_{i=1}^n (w_1 + w_2 x_i - y_i)^2$$

The update rule is to take a random point and do:

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \leftarrow \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} - \eta \begin{bmatrix} 2(w_1 + w_2 x_i - y_i) \\ 2x_i(w_1 + w_2 x_i - y_i) \end{bmatrix}$$

2. Alternating least squares

$$\min_{q \cdot, p \cdot} \sum_{(u,i) \in R_o} (r_{ui} - \langle p_u, q_i \rangle)^2 + \lambda(||q_i||^2 + ||p_u||^2)$$

- With vectors p fixed:
 - Find vectors q that minimize above function
- With vectors q fixed
 - Find vectors p that minimize above function
- Iterate until convergence
- Slower in general, but parallelizes better

UNDERSTANDING ONLINE STAR RATINGS:

★★★★★ [HAS ONLY ONE REVIEW]

★★★★★☆ EXCELLENT

★★★★☆☆ OK

★★★★☆☆☆

★★★★☆☆☆☆

★★★★☆☆☆☆☆

★★☆☆☆☆☆

★★☆☆☆☆☆☆

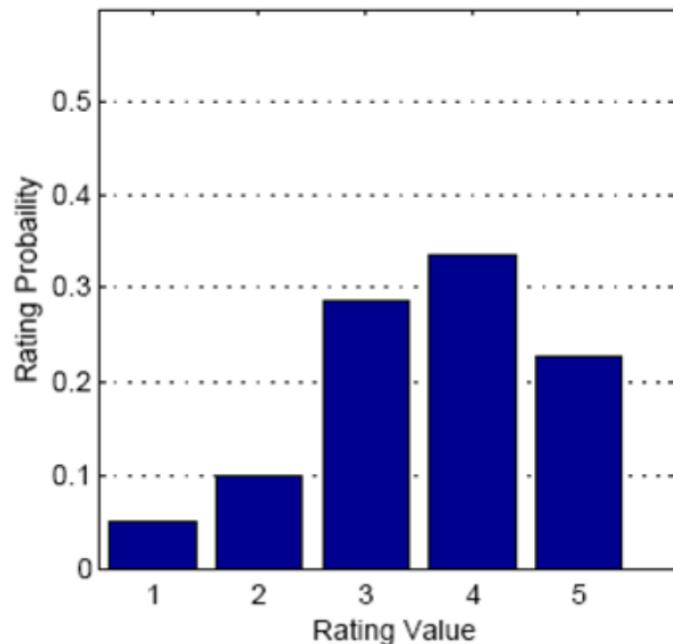
★☆☆☆☆☆☆

★☆☆☆☆☆☆☆

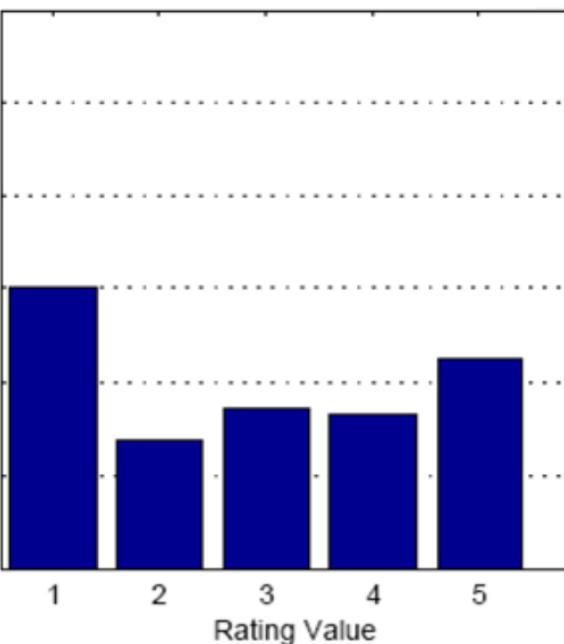
CRAP

Ratings are not normally-distributed

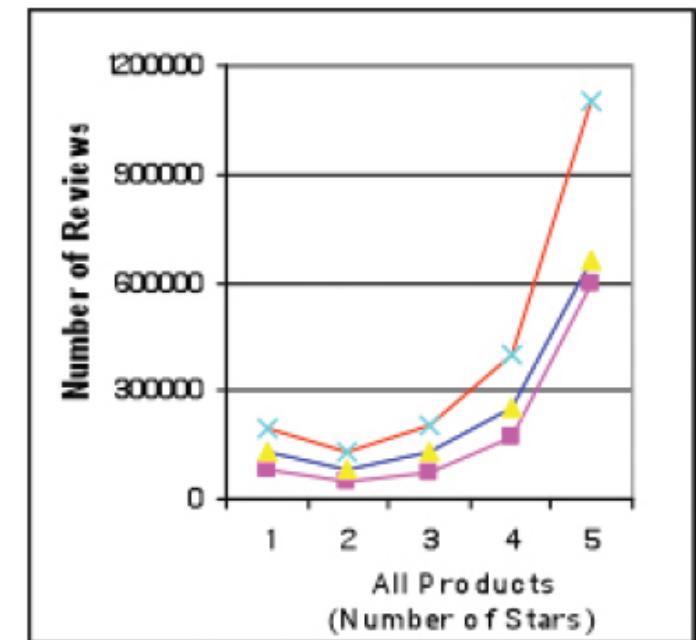
Netflix ratings



Yahoo! music ratings

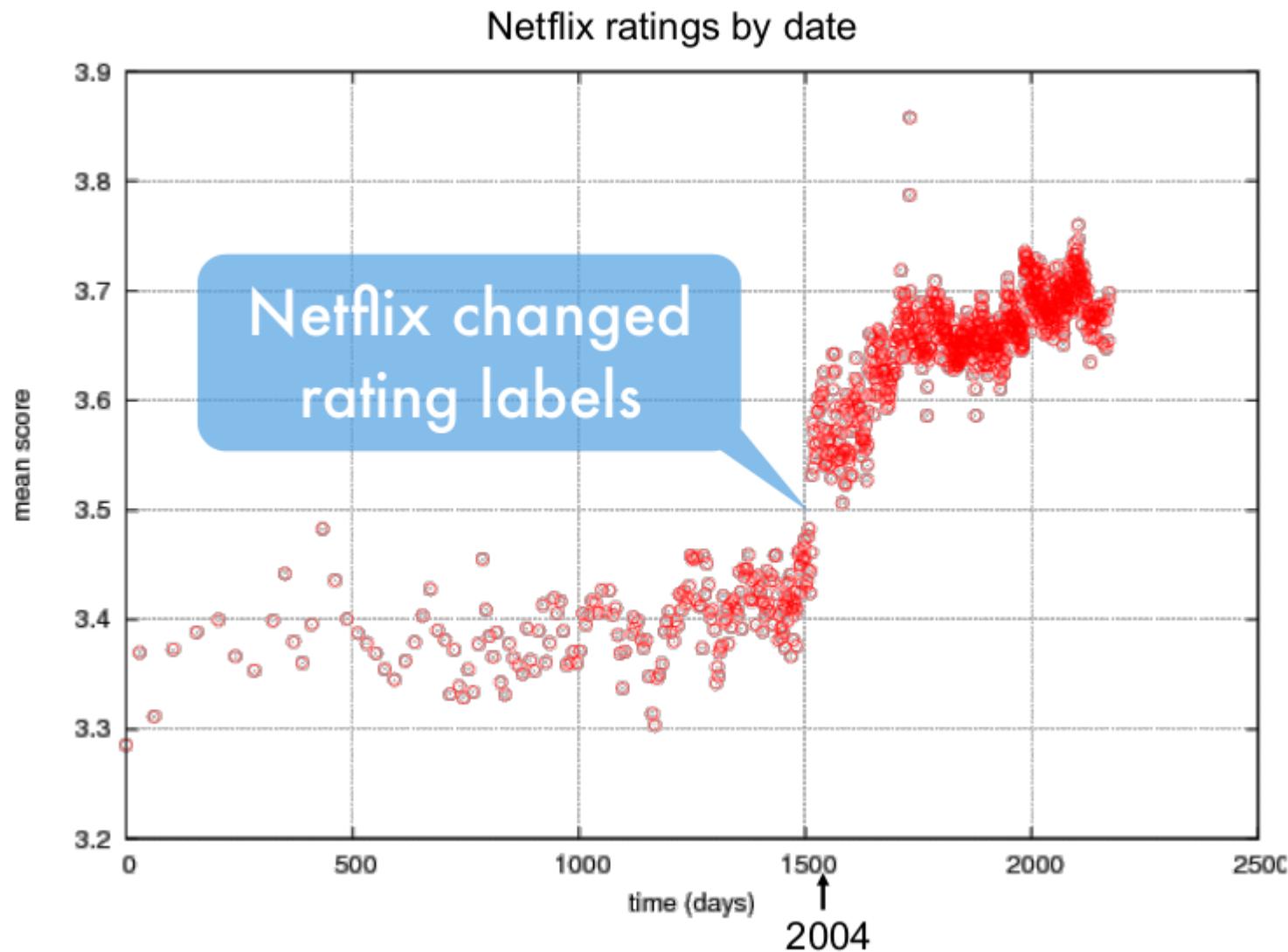


Amazon
(DVDs, Videos, Books)



Sometimes referred to as the “J” distribution of ratings

How you label ratings matters



In general, there are many biases

- Some movies always get better (or worse) ratings than others
- Some people always give better (or worse) ratings than others
- Some systems make people give better (or worse) ratings than others, e.g. labels

$$\hat{r}_{ui} = \mu + b_u + b_i + \langle p_u, q_i \rangle$$

- Time-sensitive user preferences

$$\hat{r}_{ui}(t) = \mu + b_u(t) + b_i + \langle p_u(t), q_i \rangle$$

Other approaches

Other approaches

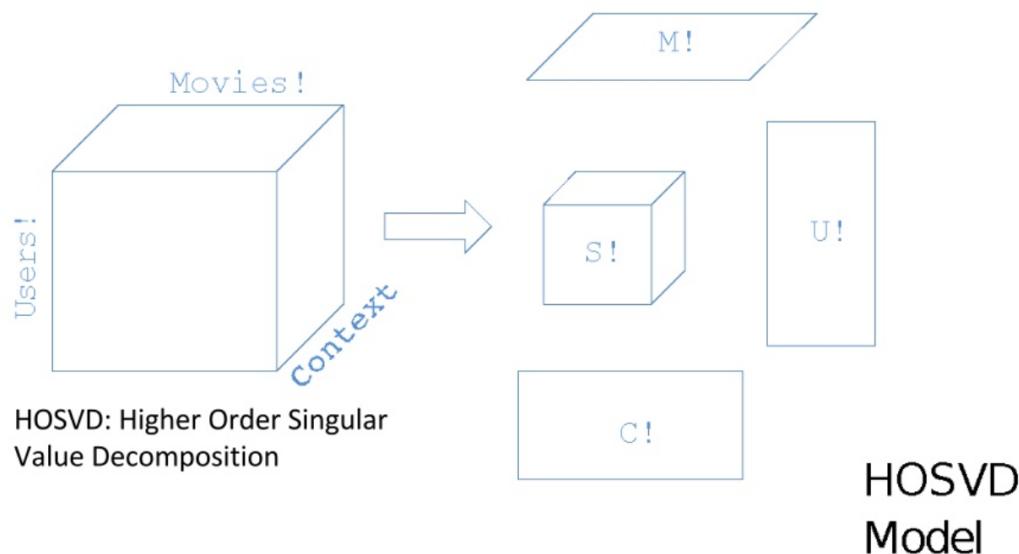
- Association rules (sequence-mining based)
- Regression (e.g. using a neural network)
 - e.g. based on user characteristics, number of ratings in different tags/categories
- Clustering
- Learning-to-rank

Hybrid methods (some types)

- Weighted
 - E.g. average recommended scores of two methods
- Switching
 - E.g. use one method when little info is available, a different method when more info is available
- Cascade
 - E.g. use a clustering-based approach, then refine using a collaborative filtering approach

Context-sensitive methods

- Context: where, when, how, ...
- Pre-filter
- Post-filter
- Context-aware methods
 - E.g. tensor factorization



Evaluation

Evaluation methodologies

- User experiments
- Precision @ Cut-off
- Ranking-based metrics
 - E.g. Kendall's Tau
- Score-based metrics
 - E.g. RMSE

Example user testing

- [Liu et al. IUI 2010] News recommender

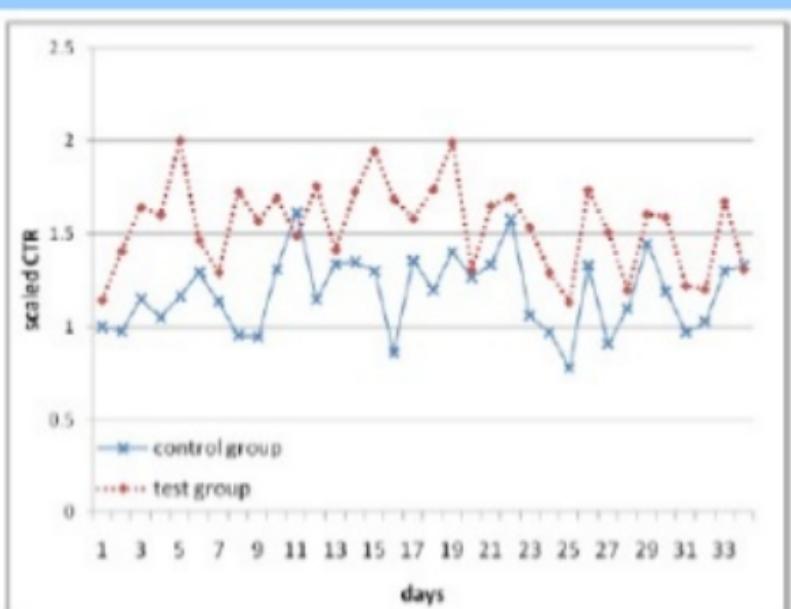


Figure 4. CTR of the recommended news section

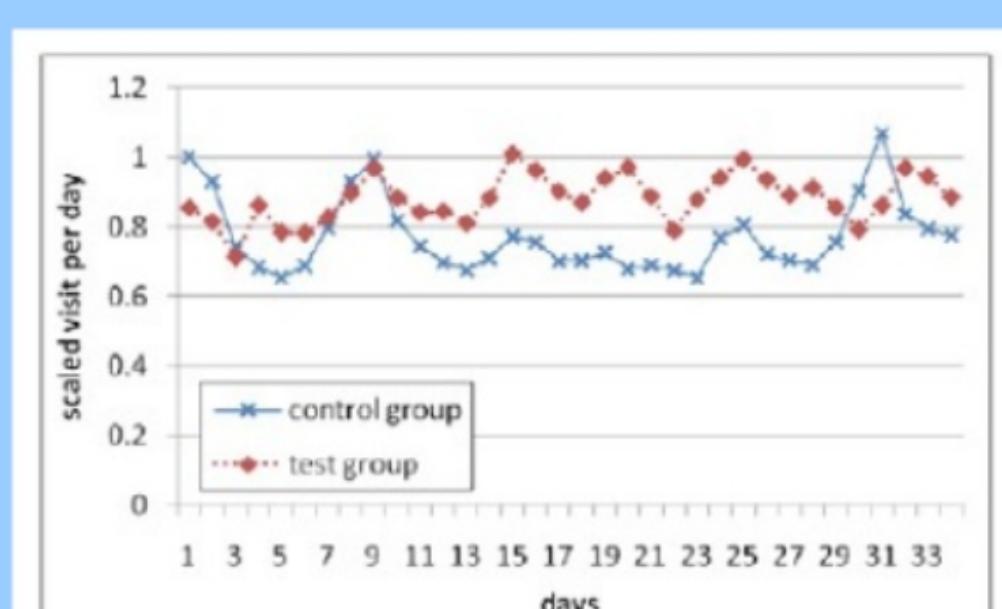


Figure 6. Frequency of website visit per day

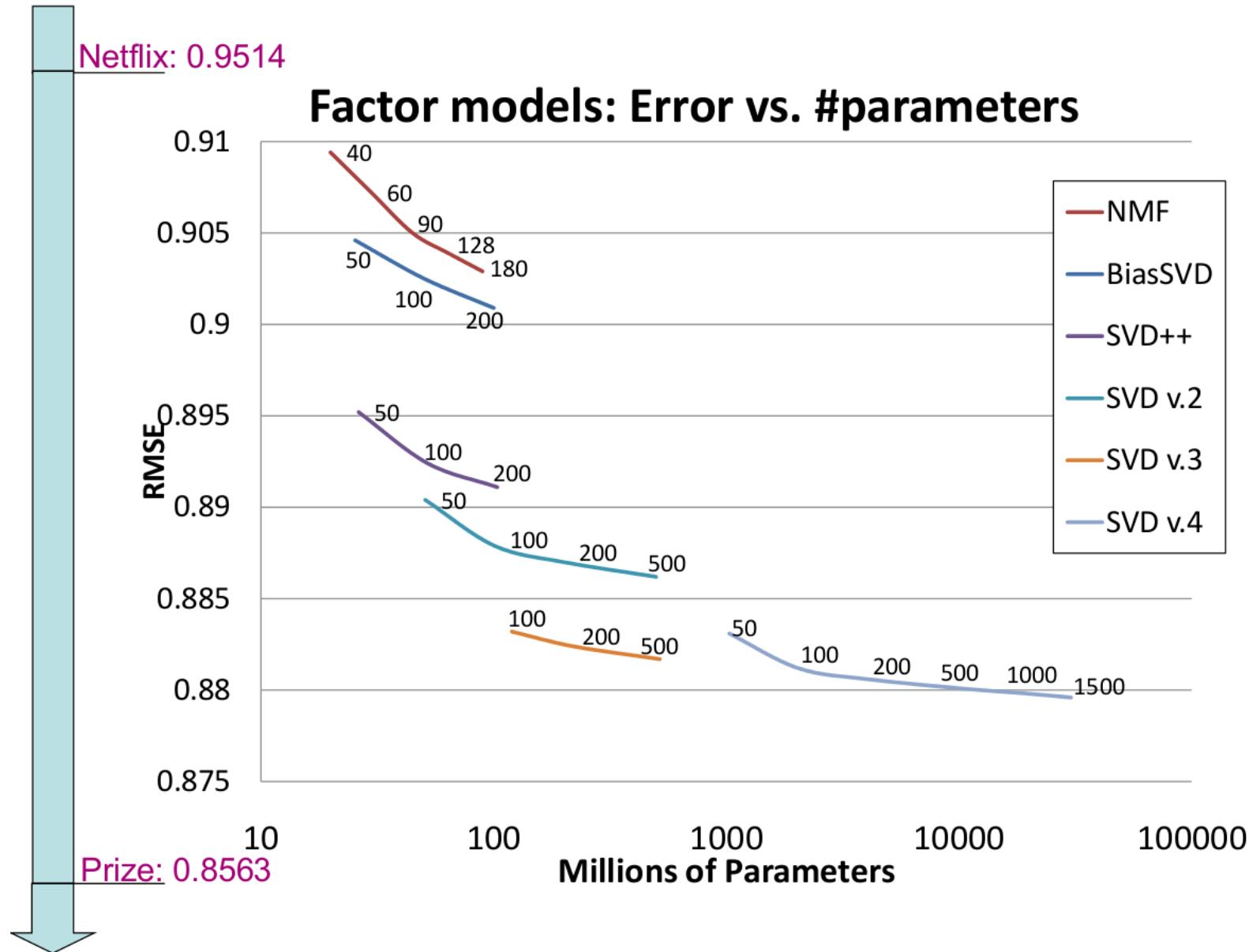
Score-based metric: RMSE

- “Root of mean square error”

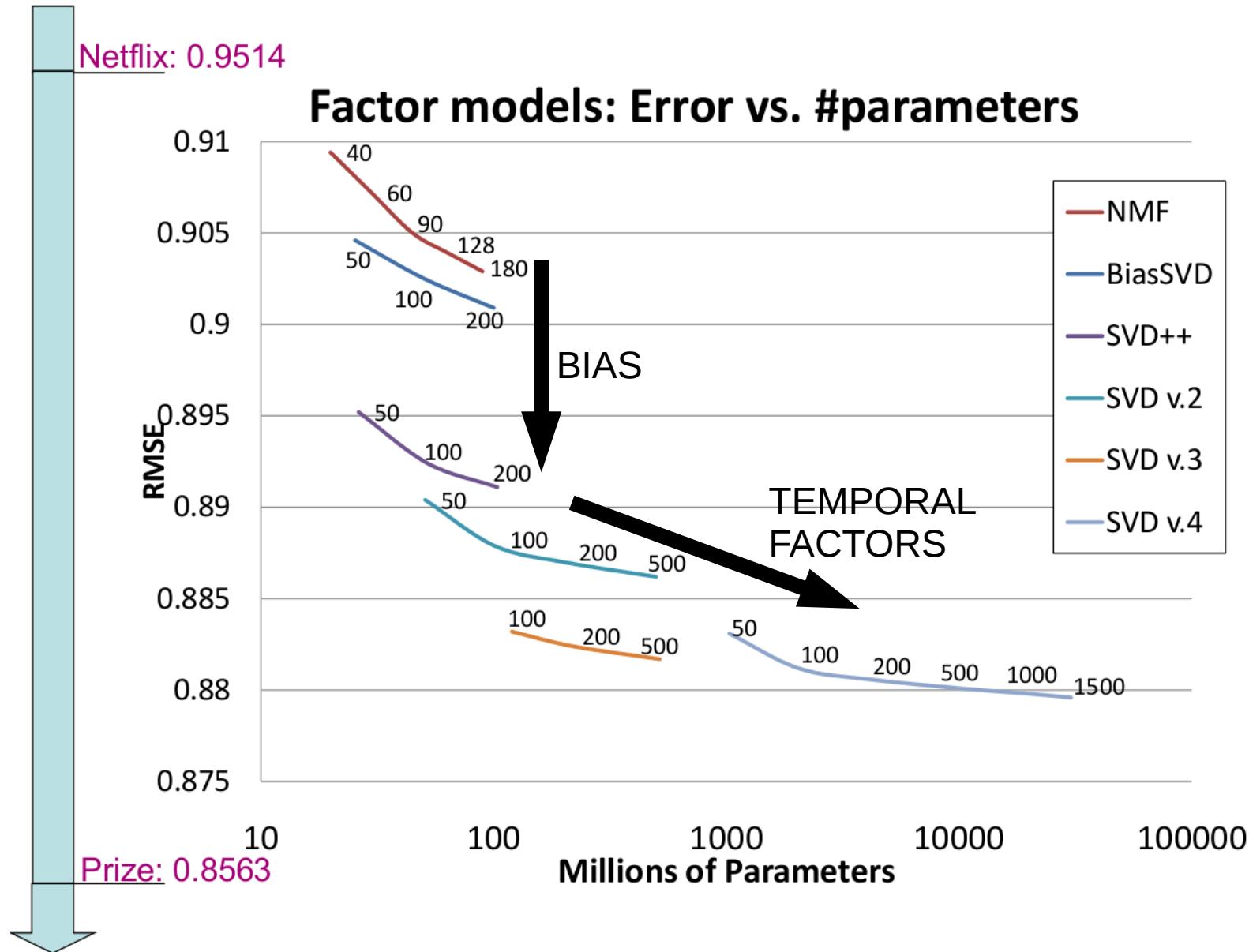
$$\text{RMSE} = \sqrt{\frac{1}{|R_o|} \sum_{(u,i) \in R_o} (r_{ui} - \hat{r}_{ui})^2}$$

- Problem with all score-based metrics: niche items with good ratings (by those who consumed them)

Evaluation by RMSE



Evaluation by RMSE



Netflix challenge results

- It is easy to provide reasonable results
- It is hard to improve them

