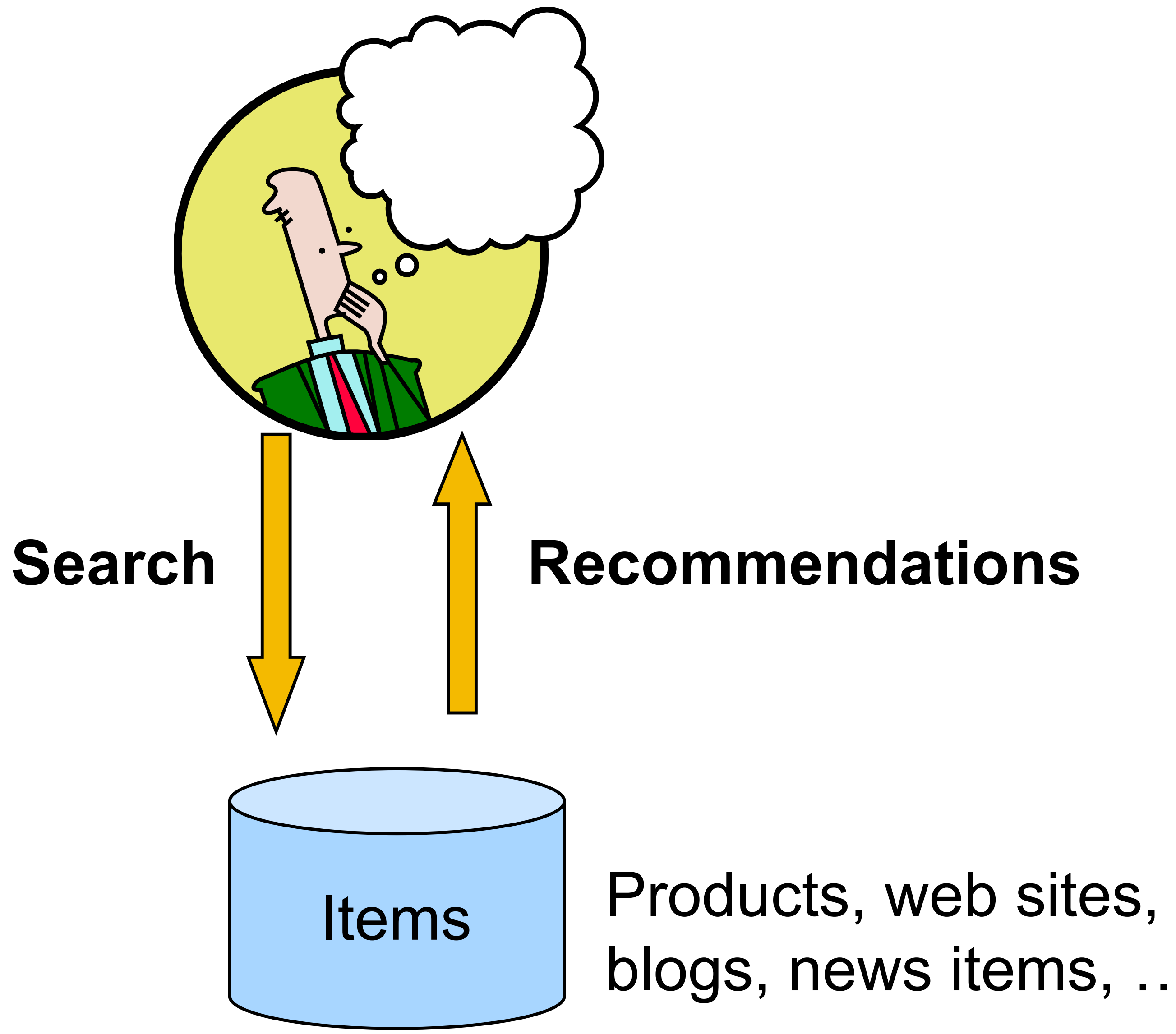
Recommendation Systems

CS 534: Machine Learning

Slides adapted from Alex Smola, Jure Leskovec, Anand Rajaraman, Jeff Ullman, Lester Mackey, Dietmar Jannach, and Gerhard Friedrich

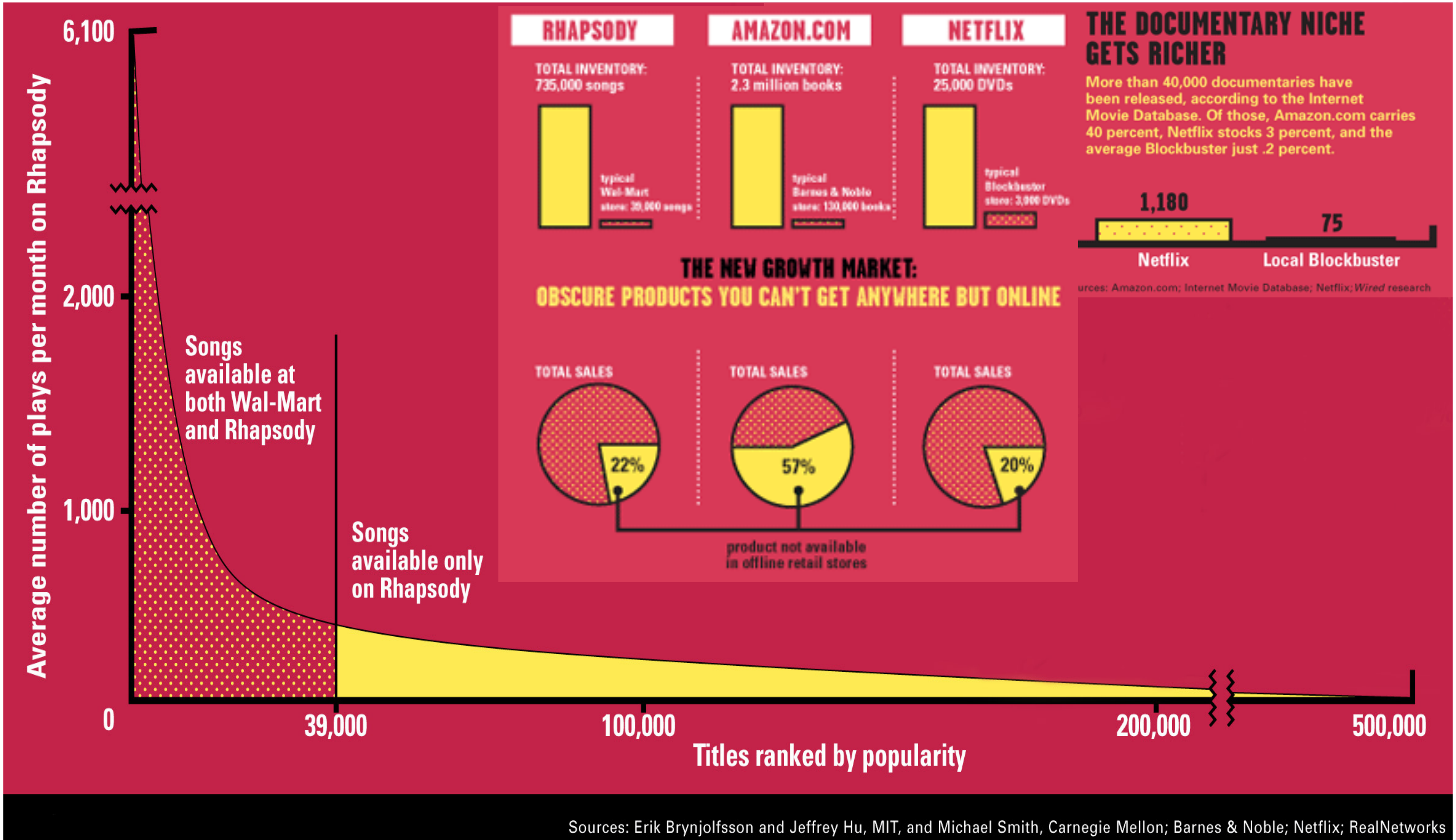
Recommender Systems (RecSys)



RecSys is Everywhere

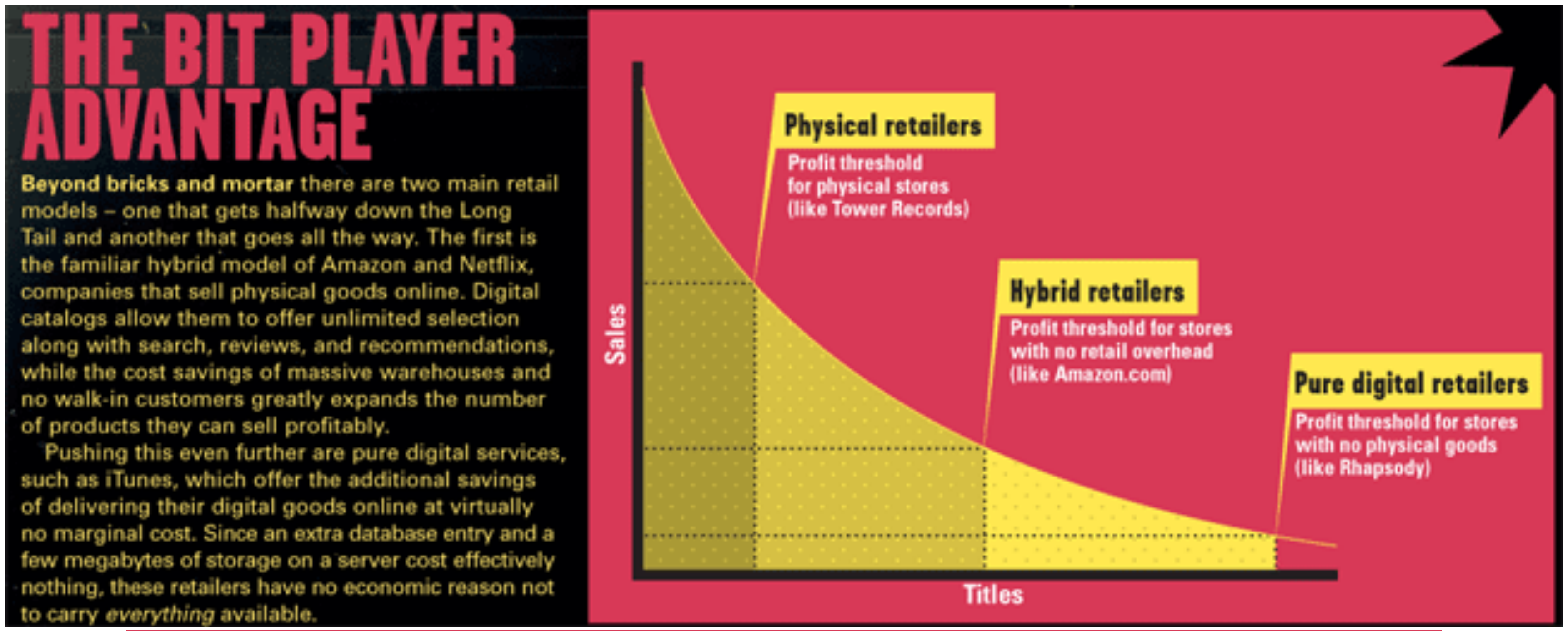
System that provides or suggests items to the end users

Long Tail Phenomenon

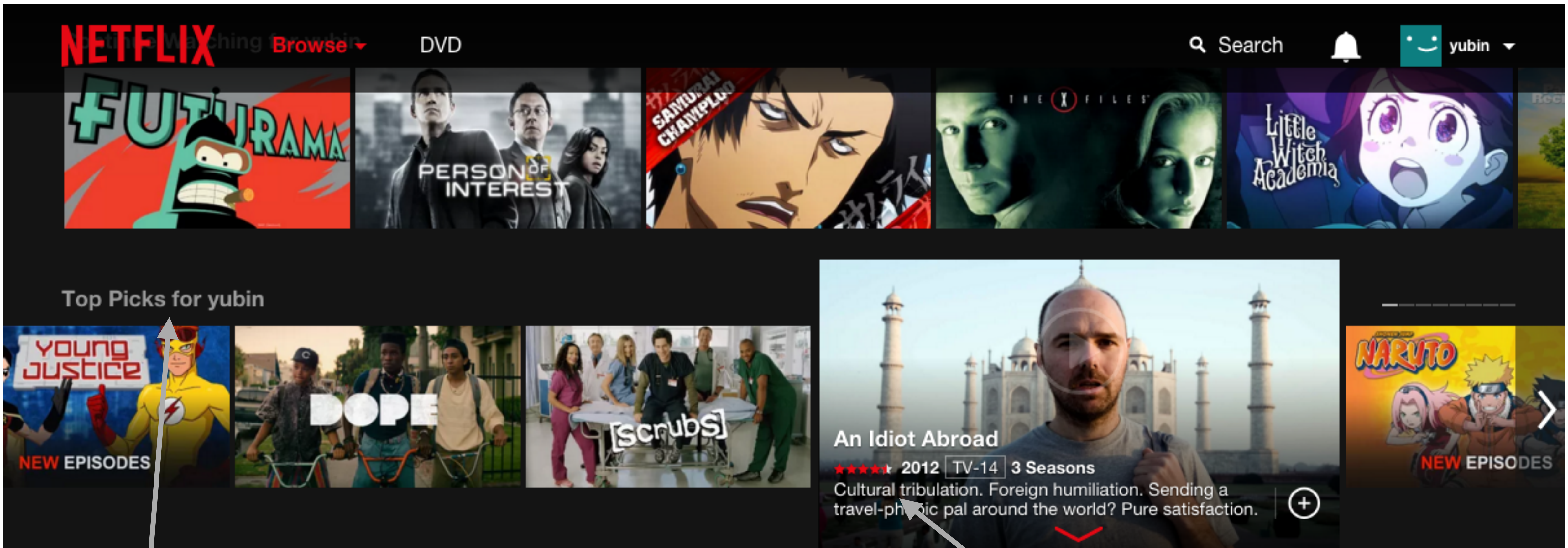


Source: Chris Anderson (2004)

Physical vs Online Presence



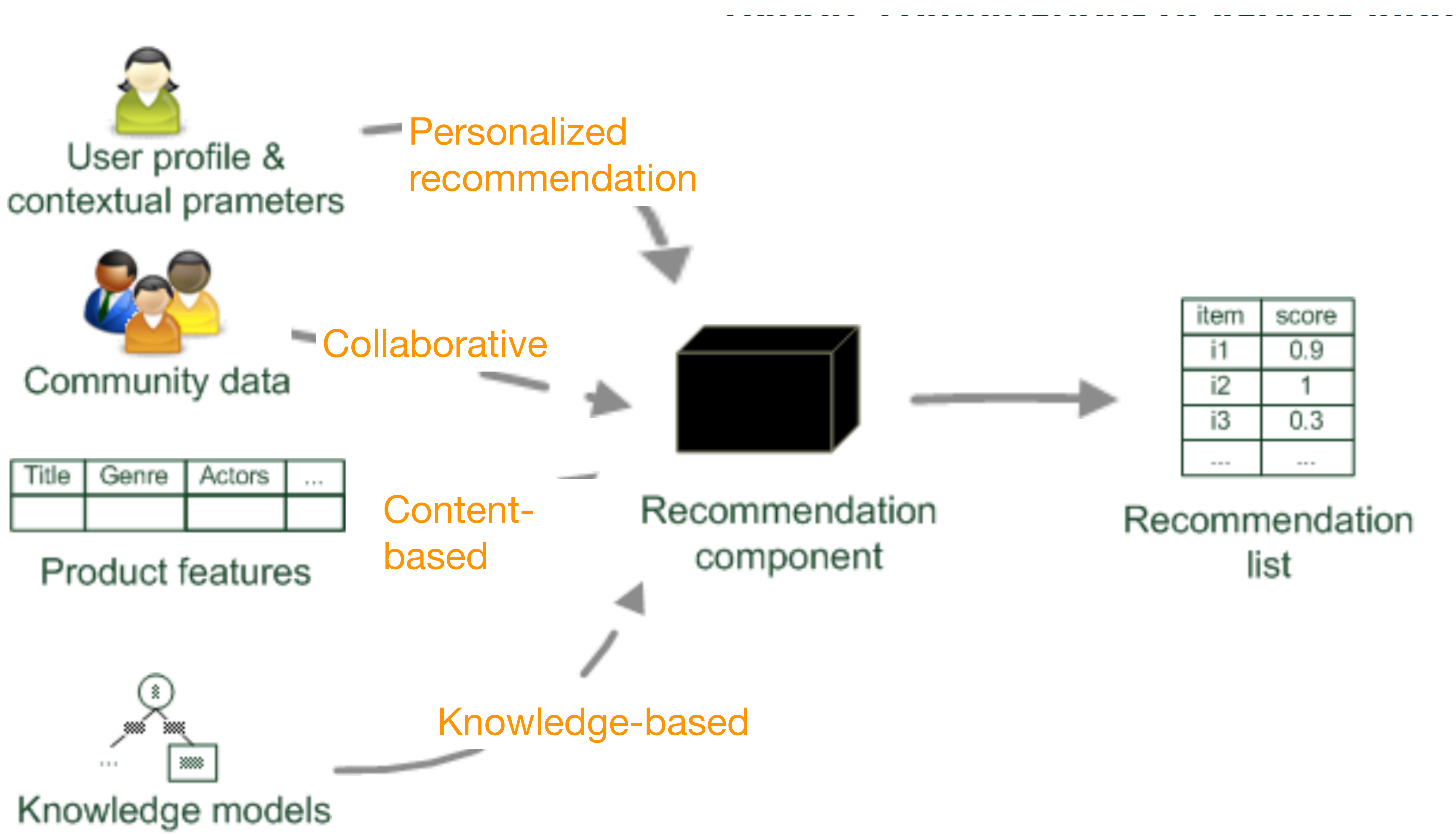
RecSys: Tasks



Task 1: Predict user rating

Task 2: Top N recommendation

RecSys: Paradigms

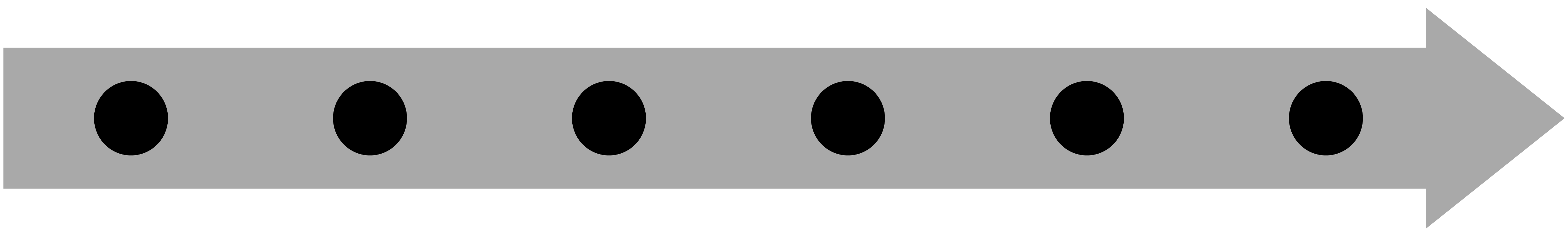


RecSys: Evolution

Item hierarchy: You bought a printer, will also need ink

Collaborative filtering & user-user similarity: People like you who bought beer also bought diapers

Social + interest graph based: Your friends like Lady Gaga so you will like Lady Gaga



Attribute based: You like action movies starring Clint Eastwood, you will also like Good, Bad, and Ugly

Collaborative filtering & item-item similarity: You like Godfather so you will like Scarface

Model based: Training SVM, LDA, SVD for implicit features

RecSys: Basic Techniques

	Pros	Cons
Collaborative	No knowledge- engineering effort, serendipity of results, learns market segments	Requires some form of rating feedback, cold start for new users and new items
Content-based	No community required, comparison between items possible	Content descriptions necessary, cold start for new users, no surprises
Knowledge-based	Deterministic recommendations, assured quality, no cold- start, can resemble sales dialogue	Knowledge engineering effort to bootstrap, basically static, does not react to short-term trends

RecSys: Challenges

- Scalability - millions of objects and users
- Cold start
    - Changing user base
    - Changing inventory (movies, stories, goods)
    - Attributes
- Imbalanced dataset - user activity / item reviews are power law distributed

Netflix Prize: $1 M (2006-2009)

Netflix Movie Recommendation

- Training Data:
  - 480,000 users
  - 17,700 movies
  - 6 years of data: 2000-2005
- Test data: most recent ratings of each user

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	?

Evaluation Metrics

Error on unseen test set Q, not on training error

- Root Mean Square Error

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

- Mean Absolute Error

$$\text{MAE} = \frac{1}{|S|} \sum_{(i,u) \in S} |\hat{r}_{ui} - r_{ui}|$$

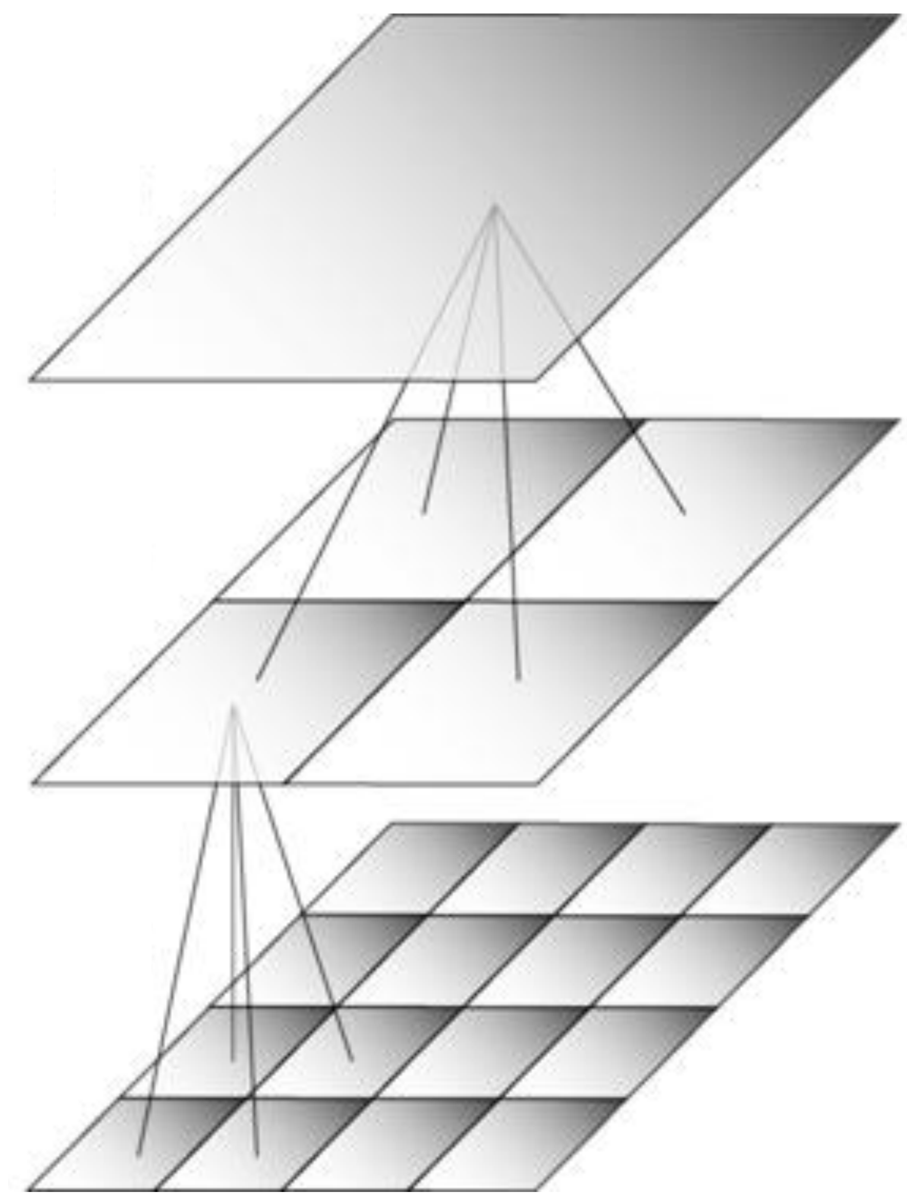
- Rank-based objectives (e.g., What fraction of true top-10 preferences are in predicted top 10?)

Netflix Prize

- Evaluation criterion: RMSE
- Cinematch (Netflix) system RMSE: 0.9514
- Competition
  - 2700+ teams
  - $1 M prize for 10% improvement on Netflix

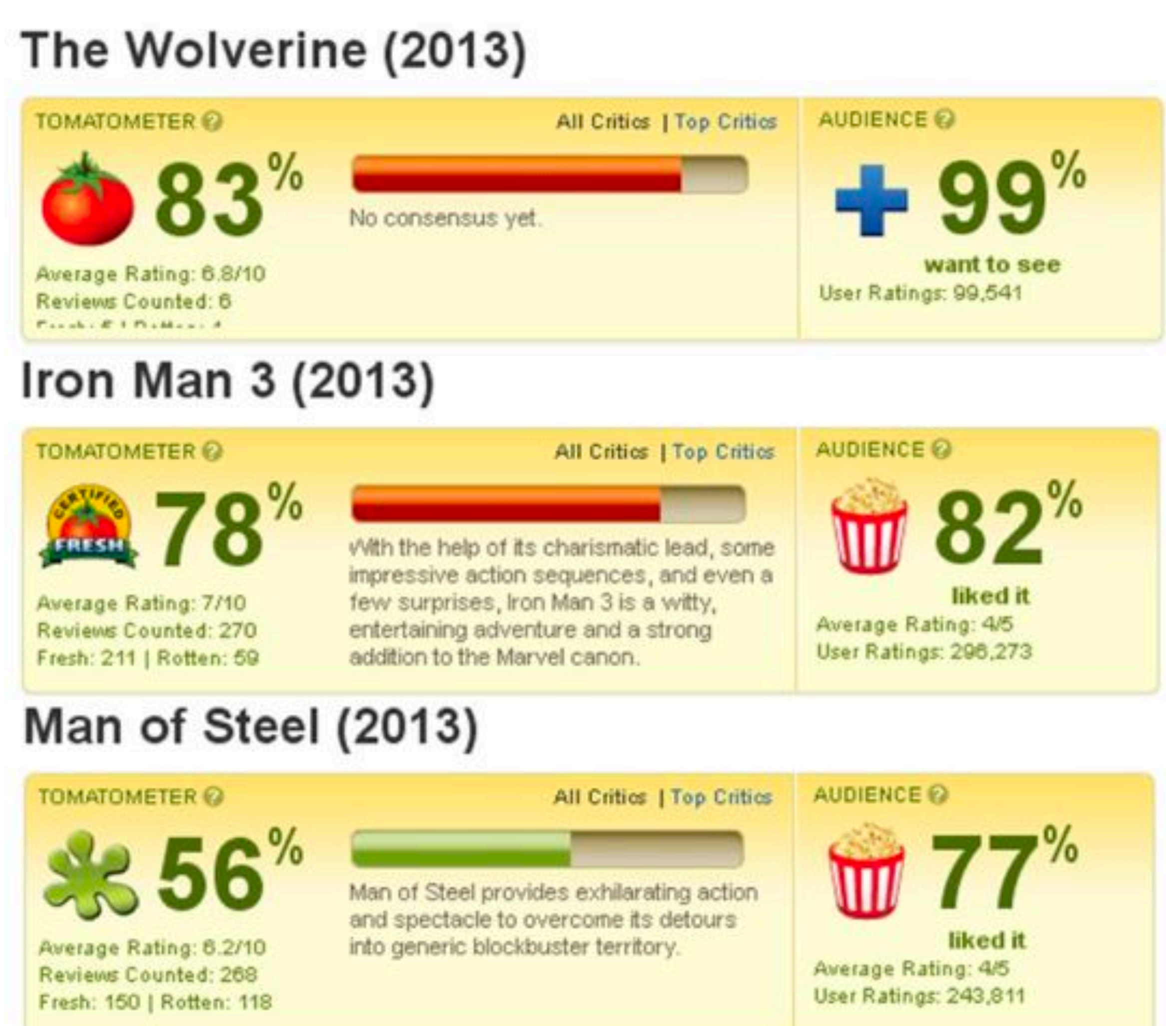
Netflix Winner: BellKor

- Multi-scale modeling of the data:
  - Global: Overall deviations of users & movies
  - Factorization: “Regional” effects
  - Collaborative filtering: Extract local patterns



Normalization / Global Bias

- Mean movie rating across all movies
- Some users tend to give higher ratings than others
- Some movies tend to receive higher rating than others



Example: Global & Local Effects

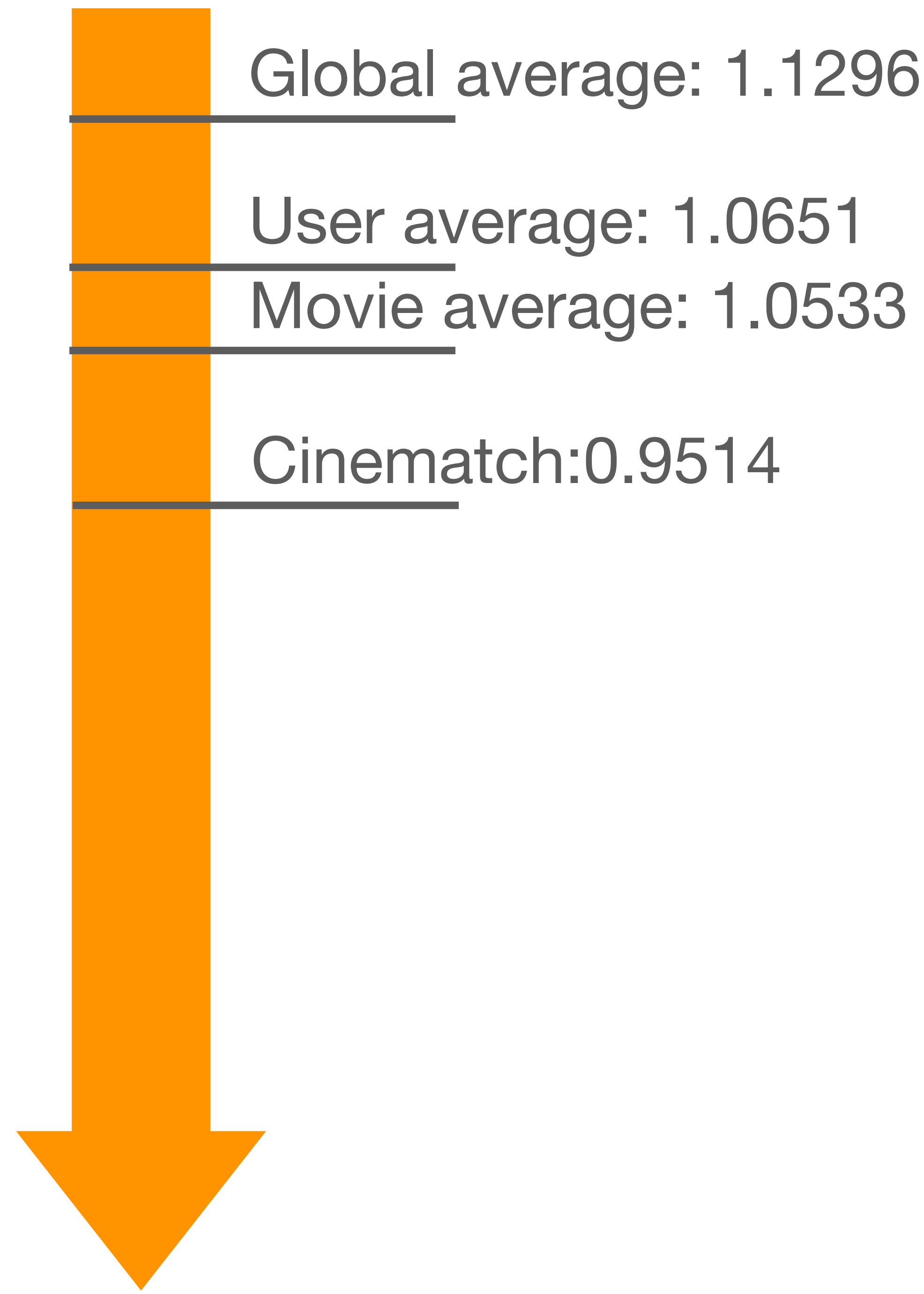
- Global effect
  - Mean movie rating: 3.7 stars
  - The Sixth Sense is 0.5 stars above average
  - Joe rates 0.2 stars below average

Baseline estimate: 4 stars

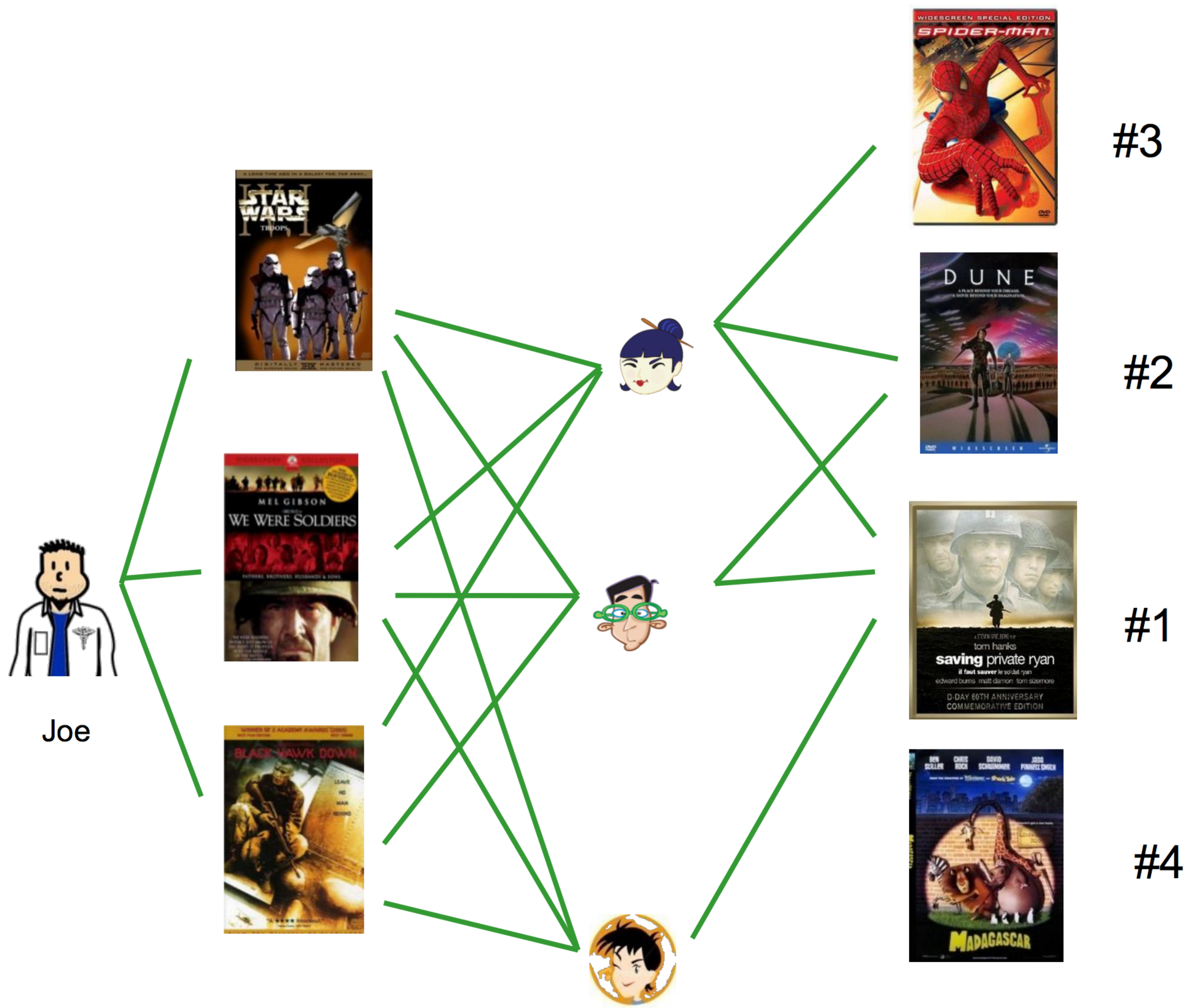
- Local effect
  - Joe doesn't like related movie Signs

Final estimate: 3.8 stars

Netflix Performance

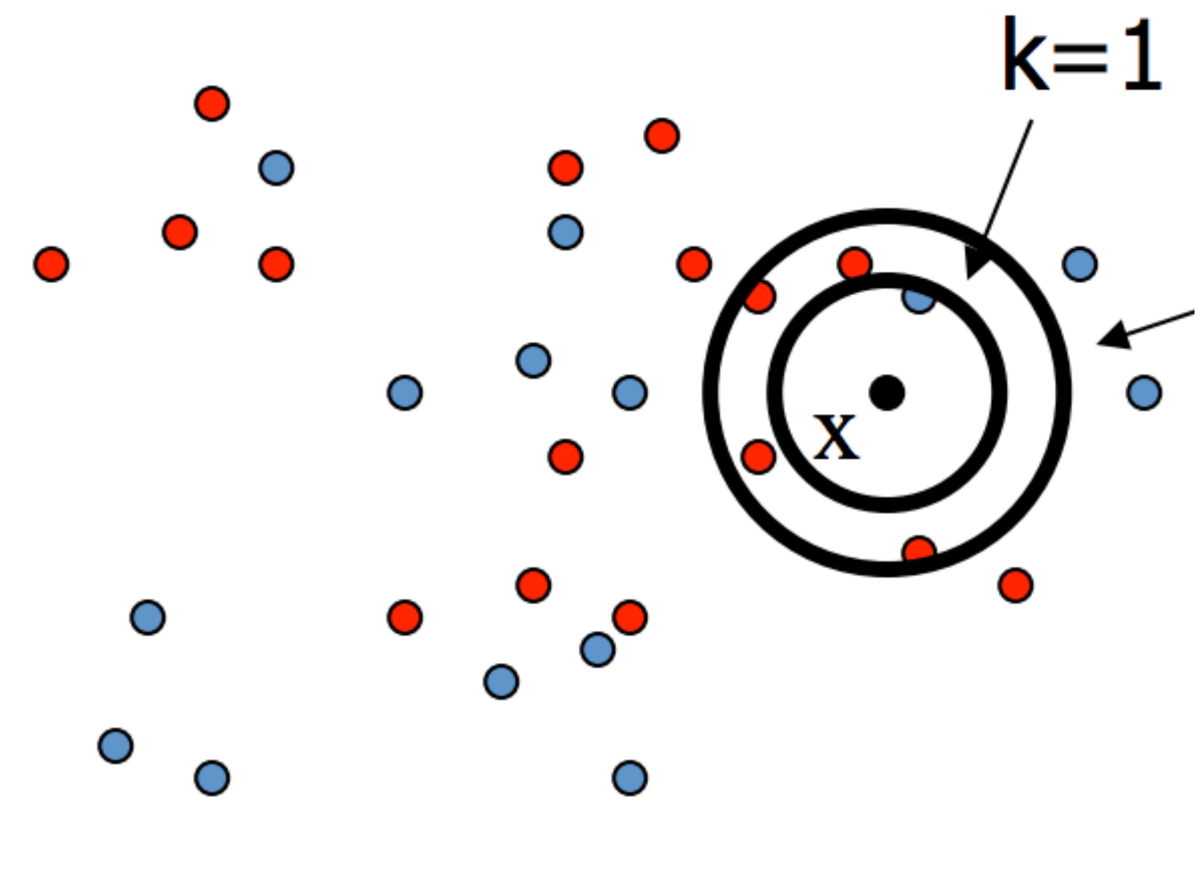


Neighborhood Methods: Basic Idea



Review: k-NN

- Examine the k-“closest” training data points to new point x
  - Closest depends on distance metric used
- Assign the object the most frequently occurring class (majority vote) or the average value (regression)



http://cs.nyu.edu/~dsontag/courses/ml13/slides/lecture11.pdf

k-NN: User-based

- Intuition: Similar users will rate the item similarly
- Represent each user as incomplete vector of item ratings
- Find set of N users who are ‘similar’ to Joe’s ratings
- Estimate Joe’s ratings based on ratings of users in set N

k-NN: User-based

- What is the right distance metric then?
  - Jaccard similarity: $D(\mathbf{x}, \mathbf{y}) = \frac{|\mathbf{x} \cap \mathbf{y}|}{|\mathbf{x} \cup \mathbf{y}|}$ ignores value of the rating
  - Cosine similarity: $D(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}||_2 ||\mathbf{y}||_2}$ missing ratings are "negative"
  - Pearson correlation coefficient

$$D(\mathbf{x}, \mathbf{y}) = \frac{\sum_{s \in S_{xy}} (\mathbf{x}_s - \bar{\mathbf{x}})(\mathbf{y}_s - \bar{\mathbf{y}})^\top}{\sqrt{\sum_{s \in S_{xy}} (\mathbf{x}_s - \bar{\mathbf{x}})^2} \sqrt{\sum_{s \in S_{xy}} (\mathbf{y}_s - \bar{\mathbf{y}})^2}}$$

k-NN: Item-based

- Intuition: Users rate similar items similarly
- Represent each item as incomplete vector of user ratings
- Find other similar items
- Estimate rating for item based on ratings for similar items

k-NN: Item-based

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	

k-NN: Item-based

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		**<u>0.41</u>**
4		2	4		5			4			2		**-0.10**
5			4	3	4	2					2	5	**-0.31**
6	1		3		3			2			4		**<u>0.59</u>**

use weighted average to predict

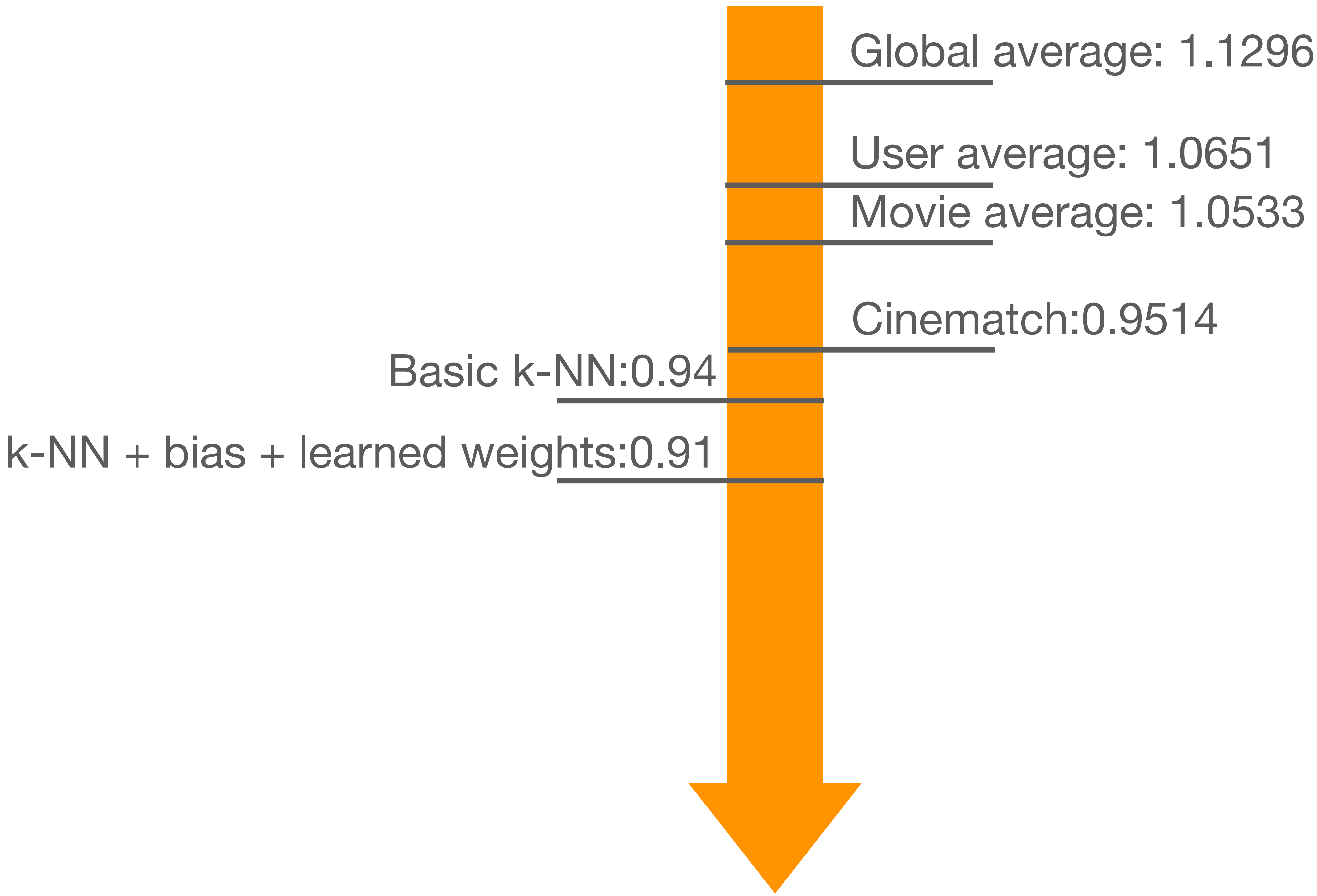
k-NN: Advantages

- Intuitive interpretation: you will like what your neighbors like
- Easy to implement and zero training time
- No feature selection needed — works for any kind of item

k-NN: Disadvantages

- Cold start
  - Need enough users in the system to find a match
  - New items and esoteric items may not have any ratings
- Sparse, high-dimensional similarity search is not easy
- Tends to recommend popular items
- Need to store all items or user vectors in memory

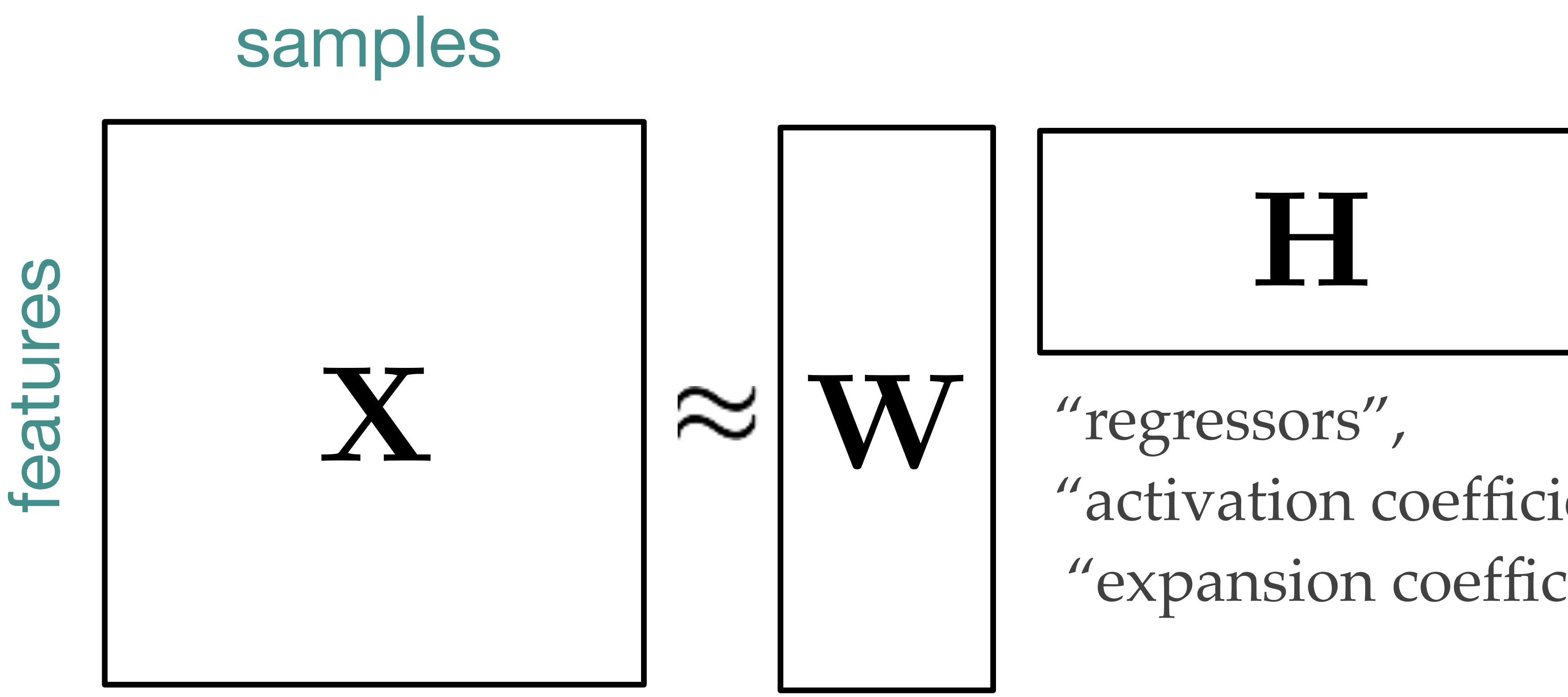
Netflix Performance



Review: Dimensionality Reduction

- Generate a low-dimensional encoding of a high-dimensional space
- Purposes:
  - Data compression / visualization
  - Robustness to noise and uncertainty
  - Potentially easier to interpret

Review: Matrix Factorization

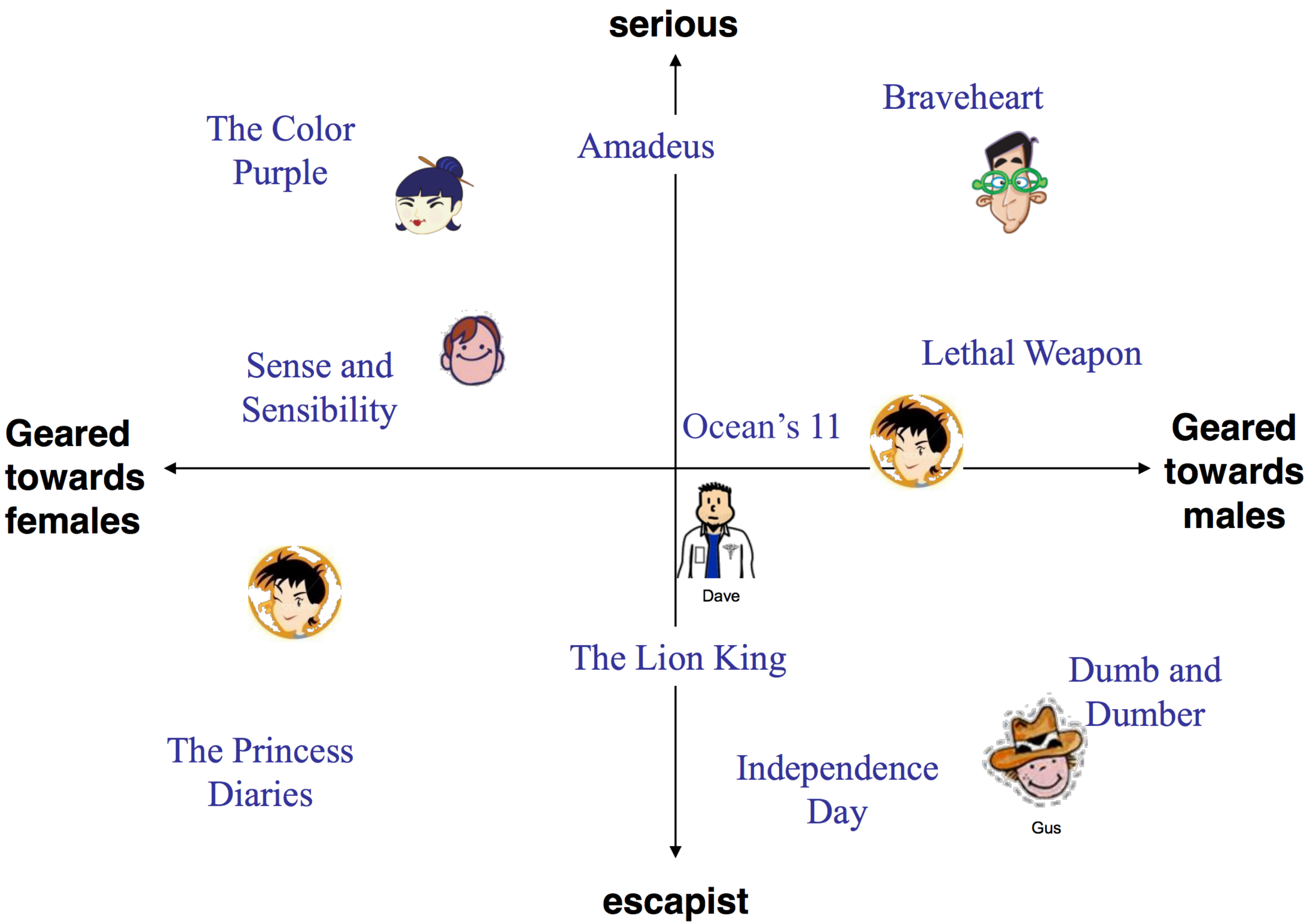


“regressors”,
“activation coefficients”,
“expansion coefficients”

Data matrix

“dictionary”, “patterns”,
“topics”, “basis”,
“explanatory variables”

Dimensionality Reduction



Review: SVD

- Each matrix can be decomposed using singular value decomposition (SVD):

$$\underbrace{\mathbf{X}}_{n \times p} = \boxed{\underbrace{\mathbf{U}}_{n \times p}}\ \boxed{\underbrace{\mathbf{D}}_{p \times p}}\ \boxed{\underbrace{\mathbf{V}}_{p \times p}}^{\top}$$

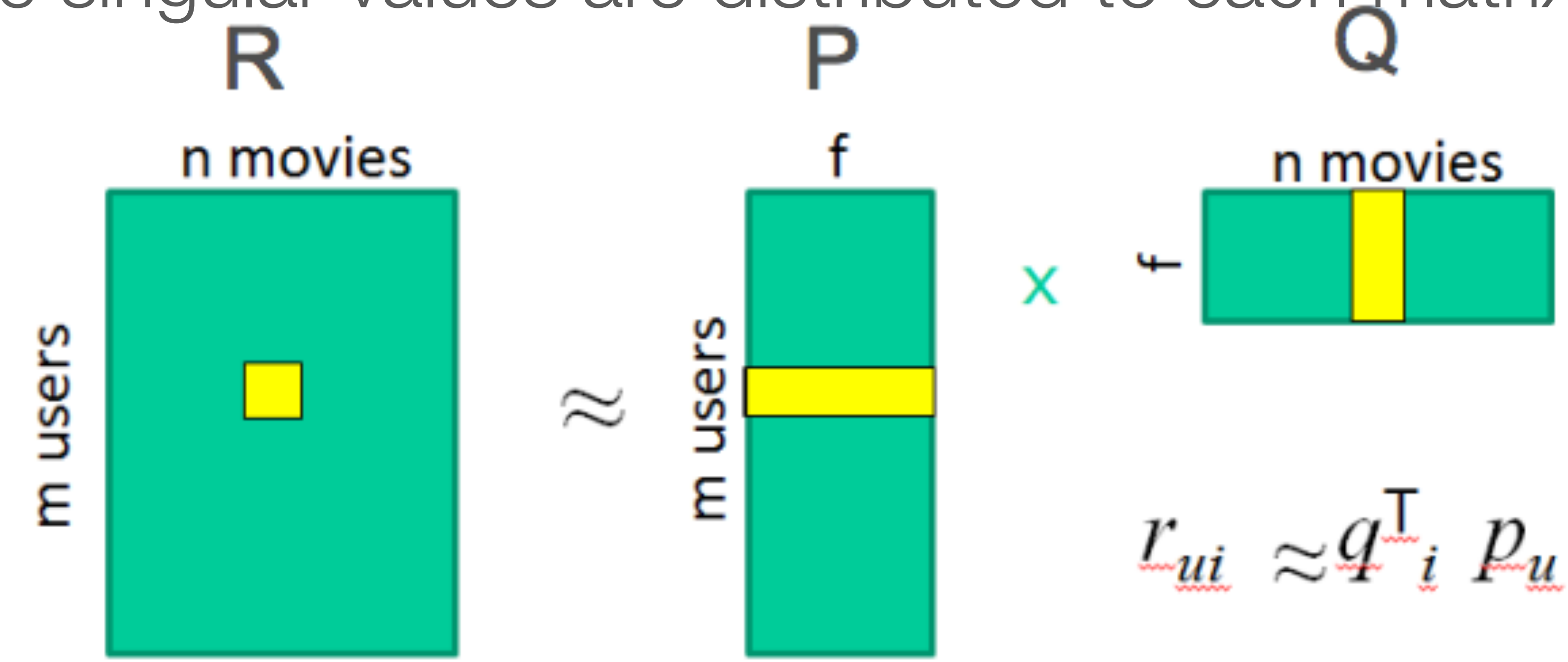
orthonormal columns which are principal components

orthonormal columns which are normalized PC scores

diagonal matrix which if each diagonal element is squared and divided by n gives variance explained

SVD to MF

- Create two new matrices (user and item matrices) where the square root of the singular values are distributed to each matrix U and V



- Interpretation:
  - p_u indicates how much user likes each latent factor f
  - q_i means the contribution of item to each of the latent factors f

RecSys: SVD

- SVD is great as it minimizes SSE which is monotonically related to RMSE
- Conventional SVD is undefined for missing entries
  - No rating can be interpreted as zero rating — is that right?

RecSys: SVD

- One idea: Expectation maximization as form of imputation
  - Fill in unknown entries with best guess
  - Apply SVD
  - Repeat
- Can be expensive and inaccurate imputation can distort data

SVD w/ Missing Values

- New idea: Model only the observed entries and avoid overfitting via regularization

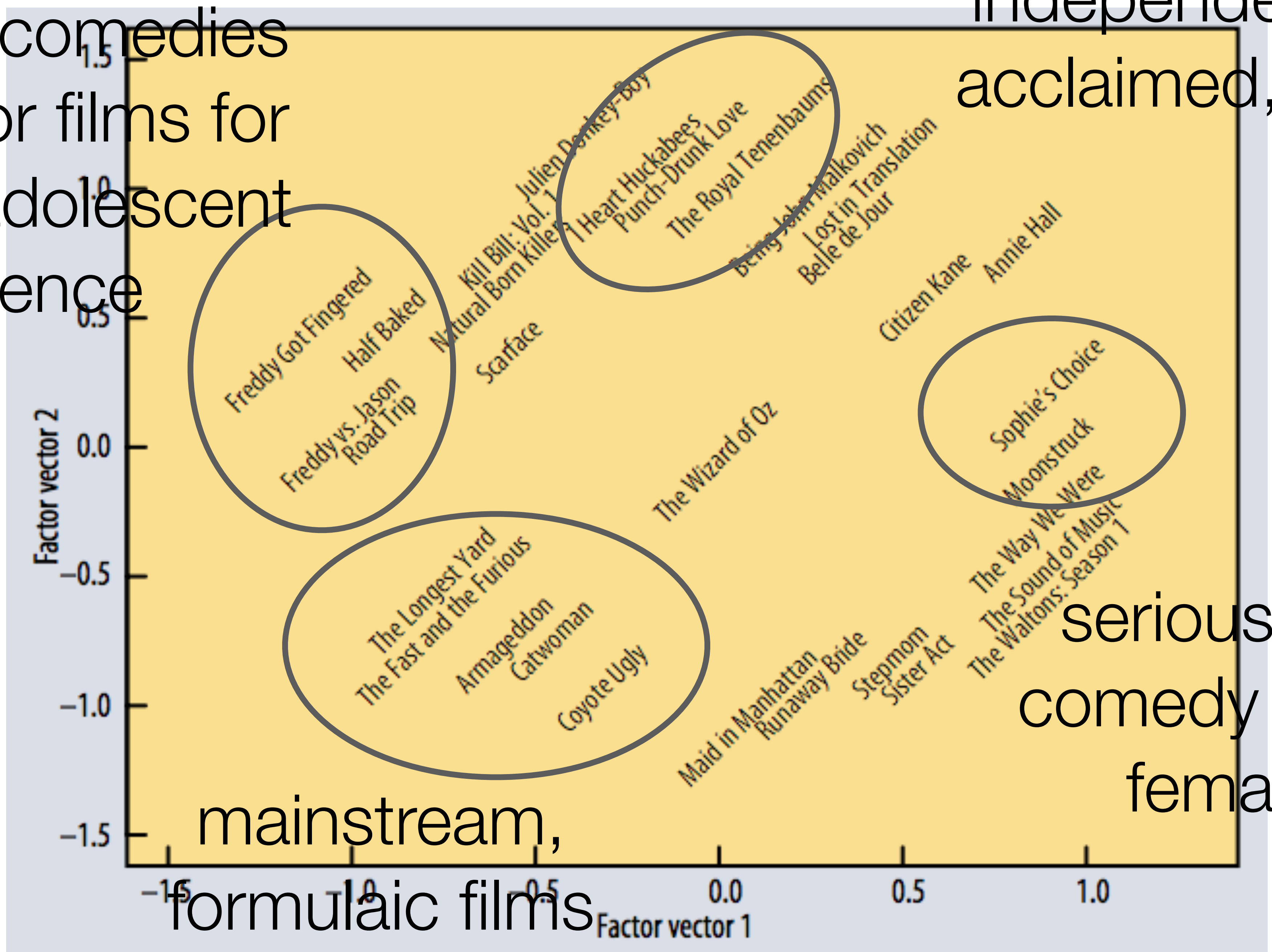
$$\min_{q\cdot,p\cdot} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2)$$

- Two methods for solving the new model
  - Stochastic gradient descent
  - Alternating least squares - easier to parallelize as each q_i is independent and more memory efficient

Netflix Results: Latent Factors

lowbrow comedies and horror films for male or adolescent audience

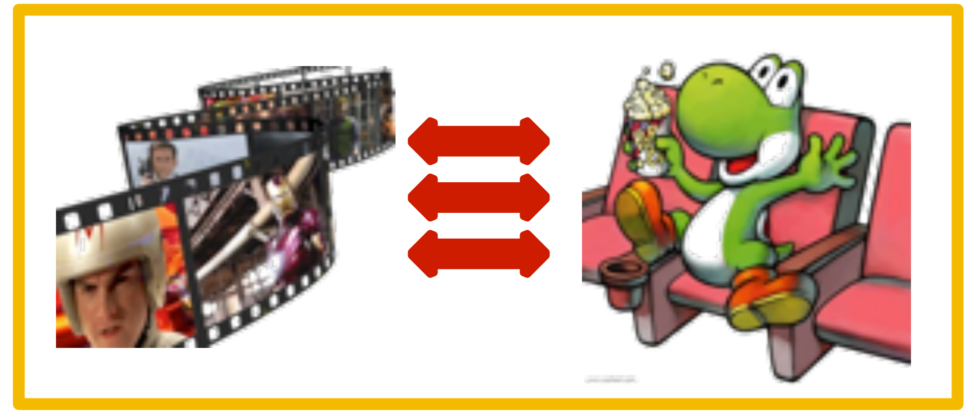
independent, critically acclaimed, quirky films



serious drama or comedy with strong female leads

mainstream, formulaic films

SVD with Bias



Baseline predictor

- Separates users and movies
- Benefits from insights into user's behavior
- Among the main practical contributions of the competition

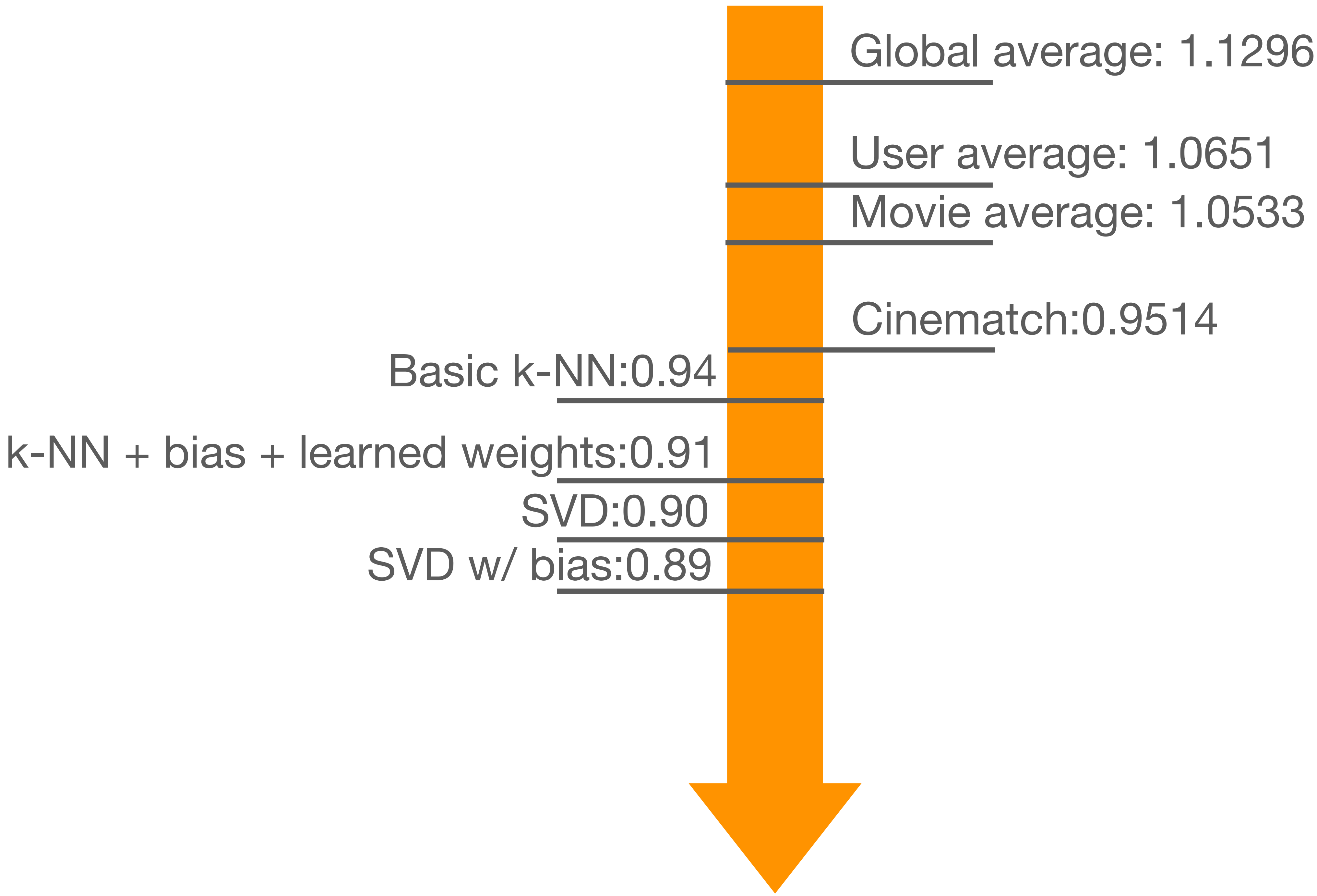
User-Movie interaction

- Characterizes the matching between users and movies
- Attracts most research in the field
- Benefits from algorithmic and mathematical innovations

$$\underset{p,q}{\text{minimize}} \sum_{(u,i)\in S} \left(r_{ui} - (\mu + b_u + b_i + \langle p_u, q_i \rangle)\right)^2 +$$

$$\lambda \left[\|p\|_{\text{Frob}}^2 + \|q\|_{\text{Frob}}^2 + \|b_{\text{users}}\|^2 + \|b_{\text{items}}\|^2 \right]$$

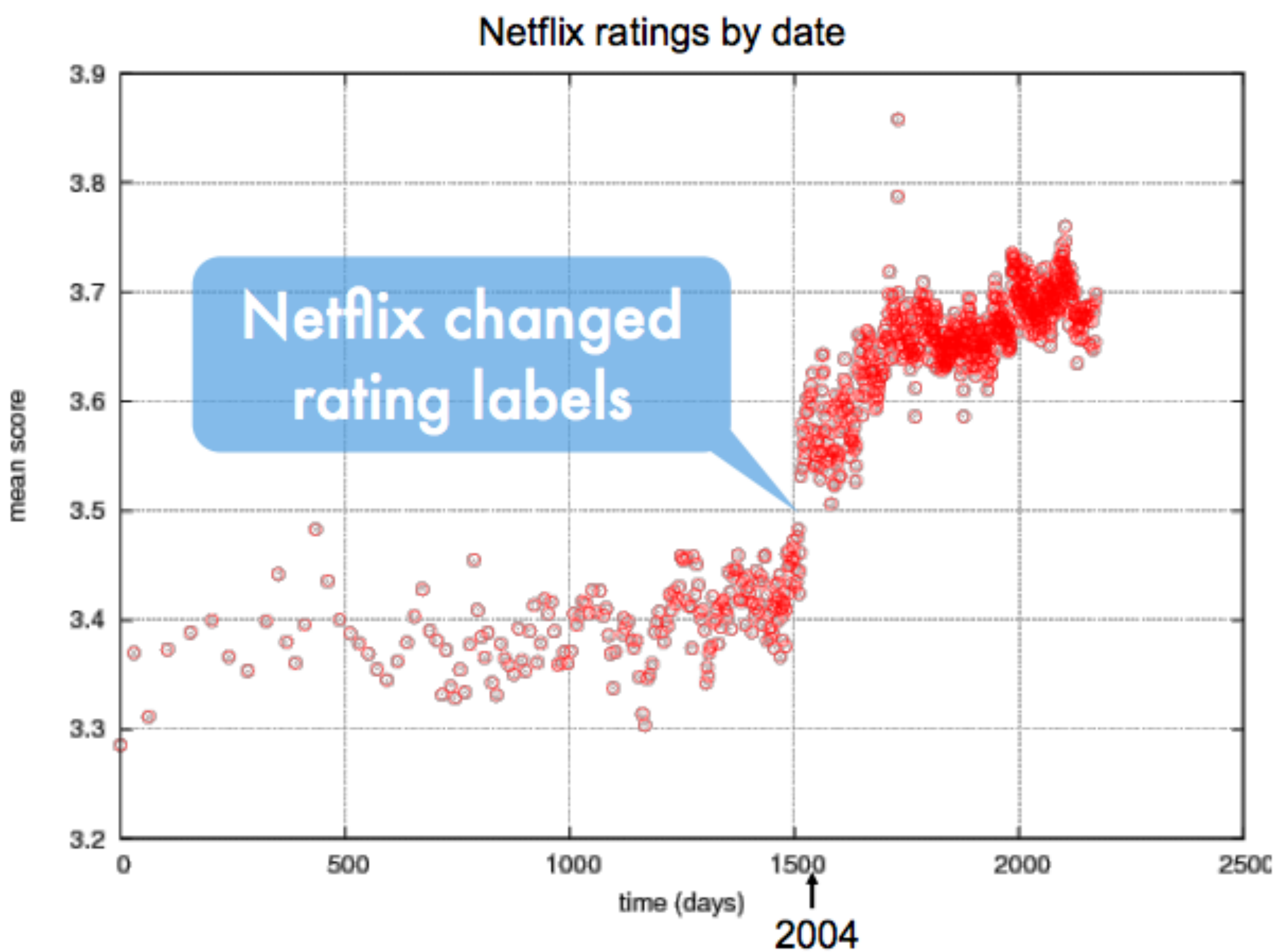
Netflix Performance



Implicit Feedback

- May have access to binary information reflecting implicit user preferences
  - Is a movie in a user's queue?
- Test source can be source — we know that user u rated item i, just don't know the rating
  - Data is not "missing at random"
  - Fact that user rated item provides information

Netflix: Temporal Bias



Netflix: Temporal Bias

- Items
  - Seasonal effects
  - Public perception (Oscars, SAG, etc.)
  - Grow and fade in popularity
  - …
- Users
  - Changed review labels
  - Anchoring (relative to previous movie)
  - Selection bias for time of viewing
  - …

Temporal SVD

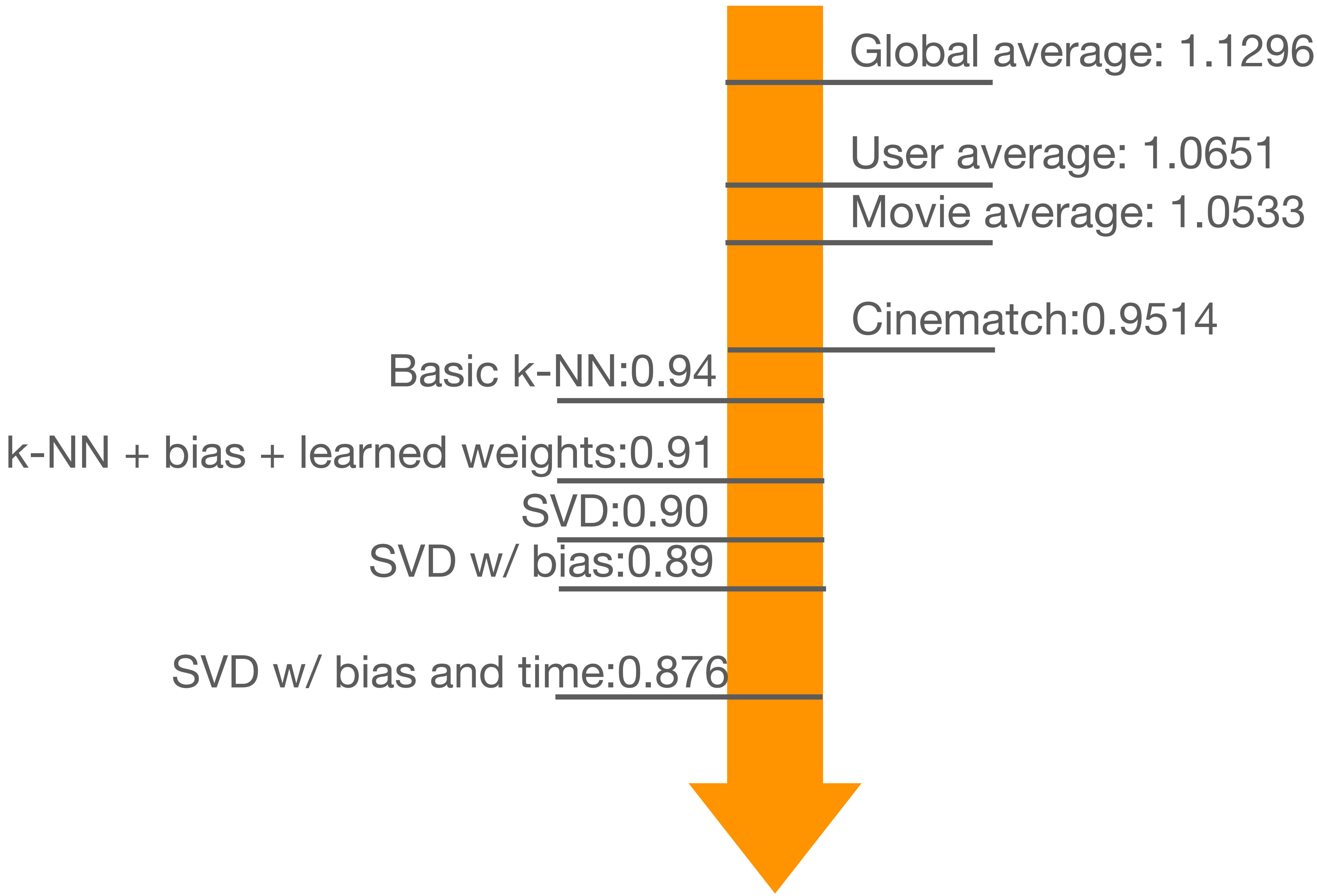
linear modeling of user biases

movie-related temporal effects

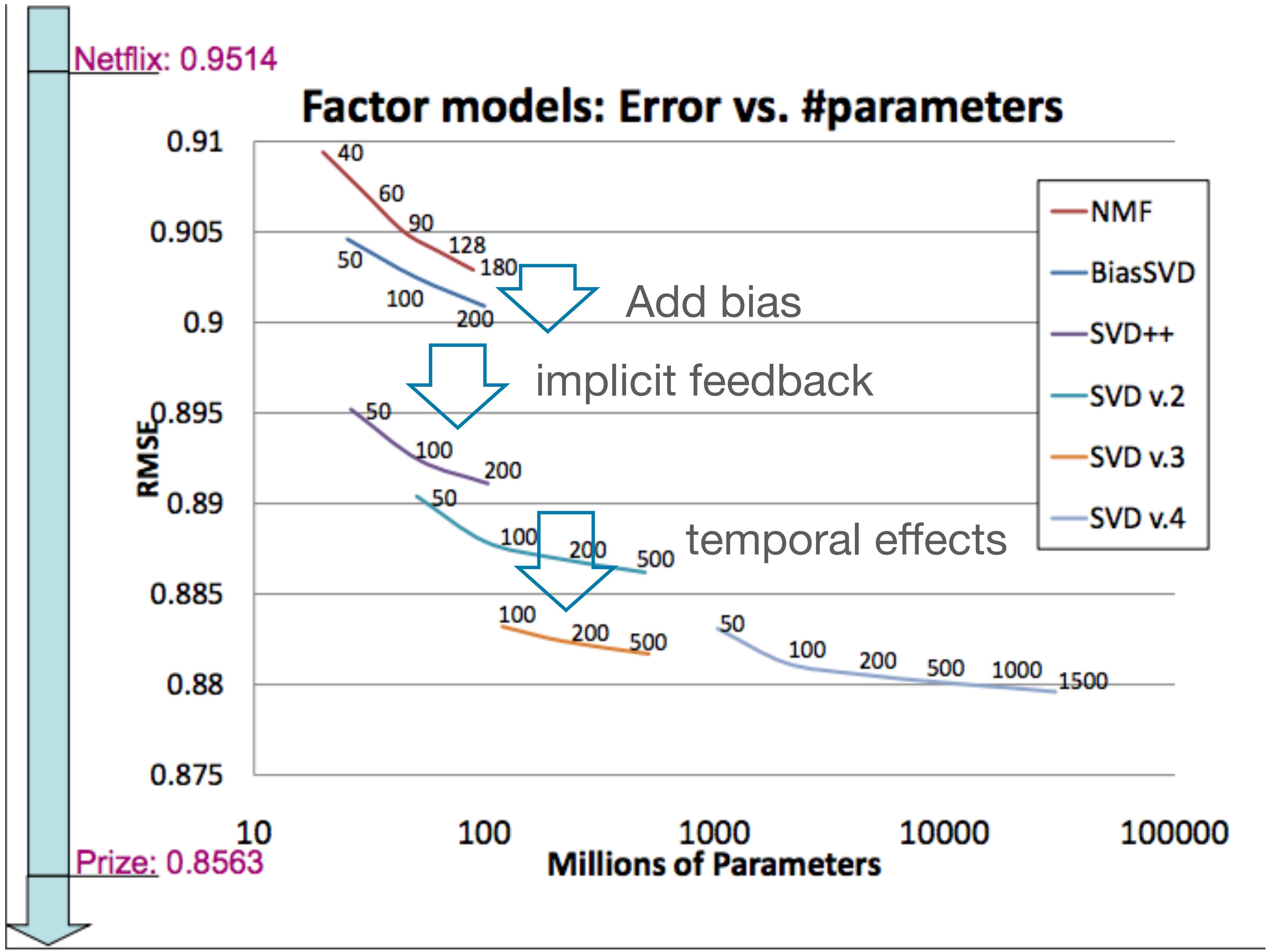
$$\min \sum_{(u,i,t)\in\mathcal{K}} (r_{ui}(t) - \mu - b_u - \boxed{\alpha_u \mathrm{dev}_u(t)} - \boxed{b_{u,t}} - b_i - \boxed{b_{i,\mathrm{Bin}(t)}})^2$$
$$+ \lambda(b_u^2 + \alpha_u^2 + b_{u,t}^2 + b_i^2 + b_{i,\mathrm{Bin}(t)}^2)$$

single day effect — sudden “spikes”

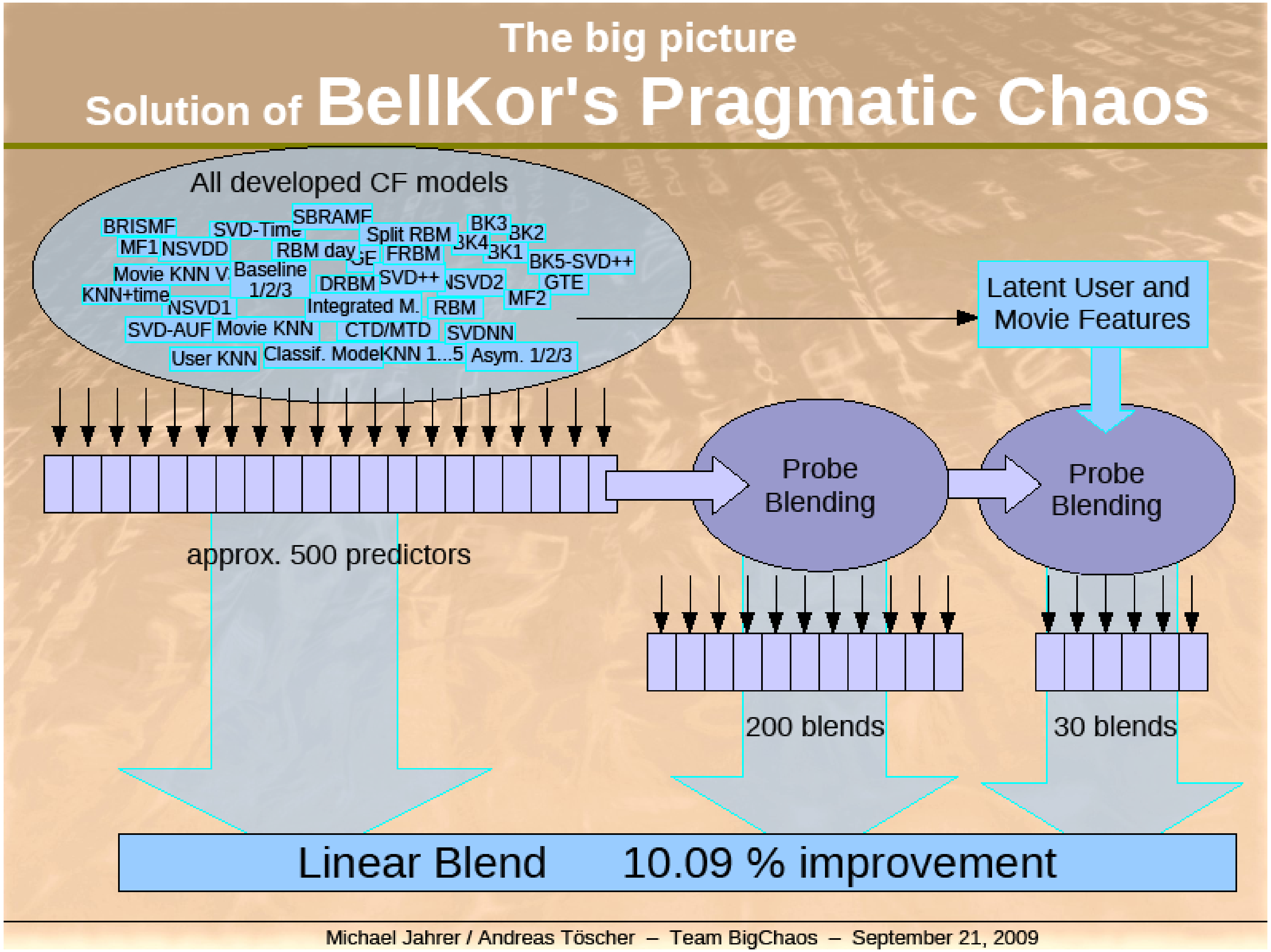
Netflix Performance



Netflix Results: RMSE



Final Solution: Kitchen Sink Approach



Winning Solution

- Beat Netflix by 10.06% in RMSE
- Tied with another team but won because submitted 20 minutes earlier
- Computationally intensive and impractical

Many More Ideas

- Cold start (new users)
- Different regularization for different parameter groups and differs users
- Sharing of statistical strength between users
- Hierarchical matrix co-clustering / factorization
- Incorporate social network, user profiles, item profiles
- …

RecSys: Challenges

- Relevant objectives
  - Predicting actual rating may be useless!
  - May care more about ranking of items
- Missing at random assumption
  - How can our models capture information in choices of our ratings?
- Handling users and items with few ratings

RecSys: Challenges

- Multiple individuals using the same account — individual preference
- Preference versus intention
  - Distinguish between liking and interested in seeing / purchasing
  - Worthless to recommend an item a user already has
- Scalability
  - Simple and parallelizable algorithms are preferred