CSCE 670 - Information Storage and Retrieval
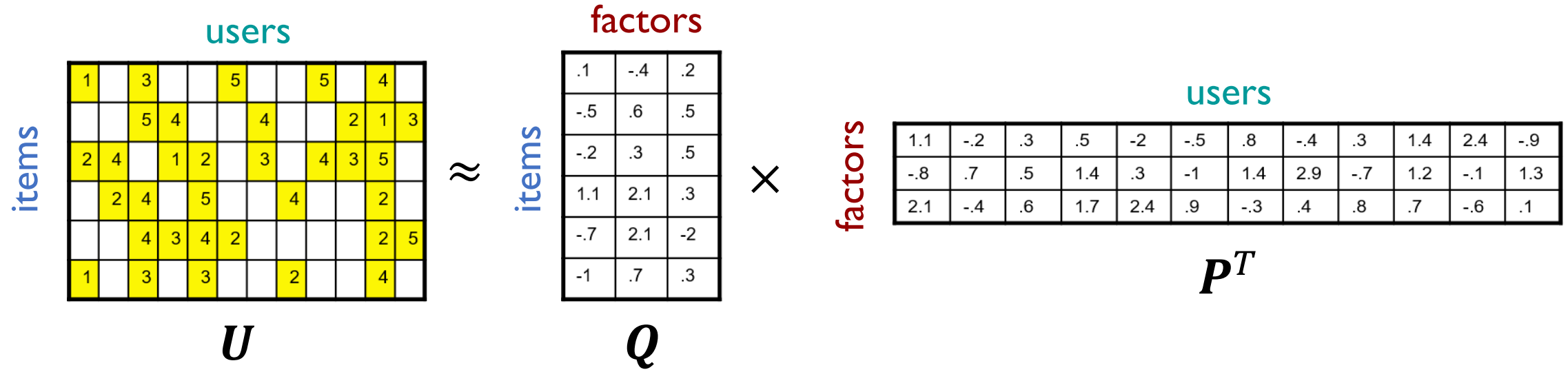
Week 7: Fundamentals of Recommender Systems (Cont'd)

Yu Zhang

yuzhang@tamu.edu

Course Website: https://yuzhang-teaching.github.io/CSCE670-S26.html

Recap: Latent Factor Model (without Bias)

$$U_{xi} = \sum_{\phi:\ \text{all factors}} Q_{i\phi} \cdot P_{x\phi}$$

Recap: Latent Factor Model (with Bias)

$$U_{xi} = \mu + b_x + b_i + \boldsymbol{q}_i \boldsymbol{p}_x^T$$

- μ: overall mean movie rating
  - E.g., $\mu = 2.7$
- b_x: rating deviation of user x (*to be learned*)
  - E.g., Bob is a critical reviewer. Based on the training data, his rating will be 0.7 star lower than the mean $\Rightarrow b_x = -0.7$.
- b_i: rating deviation of item i (*to be learned*)
  - E.g., Star Wars will get a mean rating of 0.5 higher than the average $\Rightarrow b_i = 0.5$
- $\boldsymbol{q}_i$ and $\boldsymbol{p}_x$: vector of user x and item i in the latent factor space (*to be learned*)
  - E.g., based on the genre, Bob likes Star Wars $\Rightarrow \boldsymbol{q}_i \boldsymbol{p}_x^T = 0.3$
- $U_{xi} = 2.7 - 0.7 + 0.5 + 0.3 = 2.8$
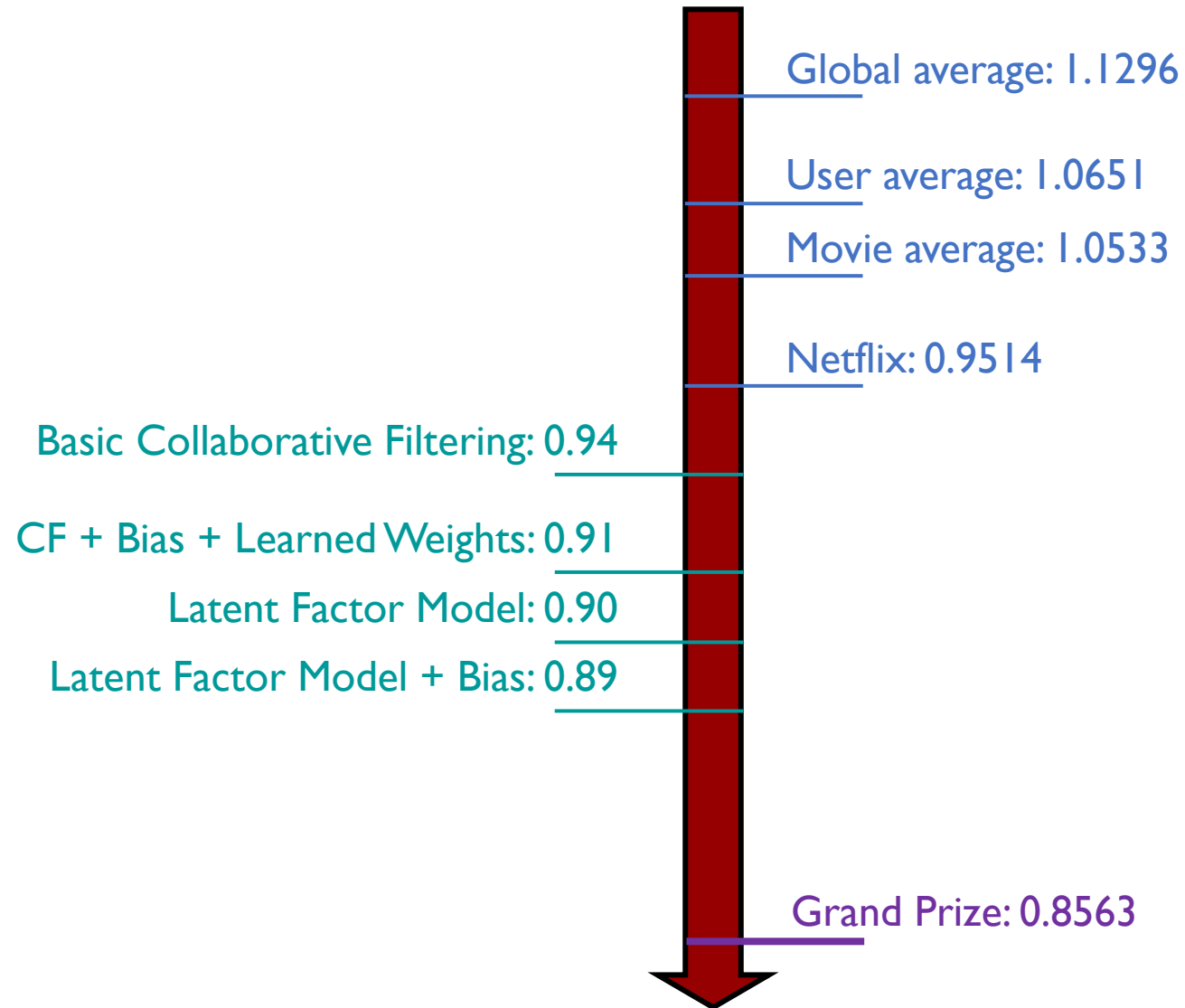
Recap: Latent Factor Model (with Bias)

$$\min_{\boldsymbol{Q},\boldsymbol{P},b_x,b_i} J = \sum_{(x,i) \text{ known}} (U_{xi} - (\mu + b_x + b_i + \boldsymbol{q}_i\boldsymbol{p}_x^T))^2$$

$$+ \left[c_1 \sum_x \|\boldsymbol{p}_x\|^2 + c_2 \sum_i \|\boldsymbol{q}_i\|^2 + c_3 \sum_x \|b_x\|^2 + c_4 \sum_i \|b_i\|^2 \right]$$

- Both biases b_x, b_i as well as interactions $\boldsymbol{q}_i, \boldsymbol{p}_x$ are treated as parameters to be learned via gradient descent

  - $P_{x\phi} = P_{x\phi} - \eta \frac{\partial J}{\partial P_{x\phi}}, \qquad Q_{i\phi} = Q_{i\phi} - \eta \frac{\partial J}{\partial Q_{i\phi}}$

 - $b_x = b_x - \eta \frac{\partial J}{\partial b_x}, \qquad b_i = b_i - \eta \frac{\partial J}{\partial b_i}$

Recap: Performance of Various Models

Global average: 1.1296

User average: 1.0651

Movie average: 1.0533

Netflix: 0.9514

Basic Collaborative Filtering: 0.94

CF + Bias + Learned Weights: 0.91
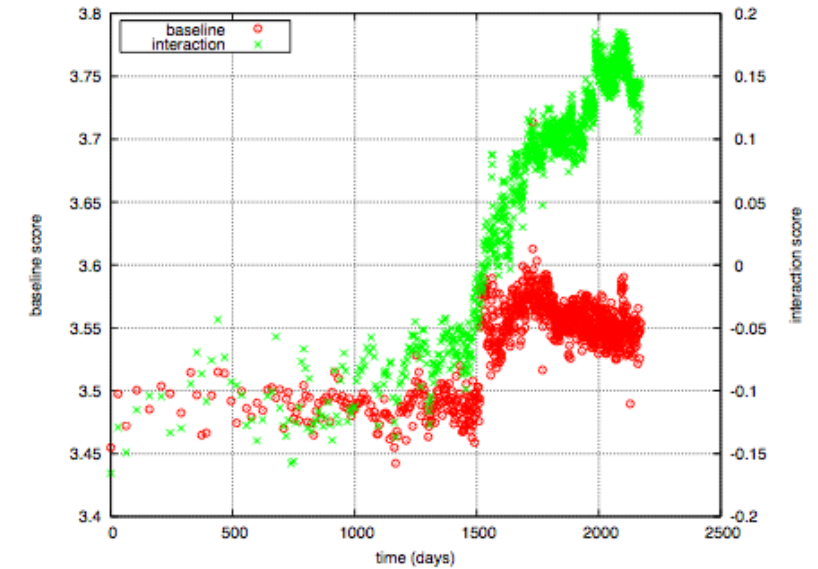
Latent Factor Model: 0.90

Latent Factor Model + Bias: 0.89

Grand Prize: 0.8563

Extended Content: The Netflix Challenge 2006-2009
(will not appear in quizzes or the exam)

Temporal Biases Of Users [Koren, KDD 2009]

- A sudden surge in the average movie rating observed in early 2004.
  - Possible reasons:
    - Improvements in Netflix
    - GUI improvements
    - Meaning of rating changed

- For the rating of a single movie, its age is an important factor.
  - Users prefer the newest movies
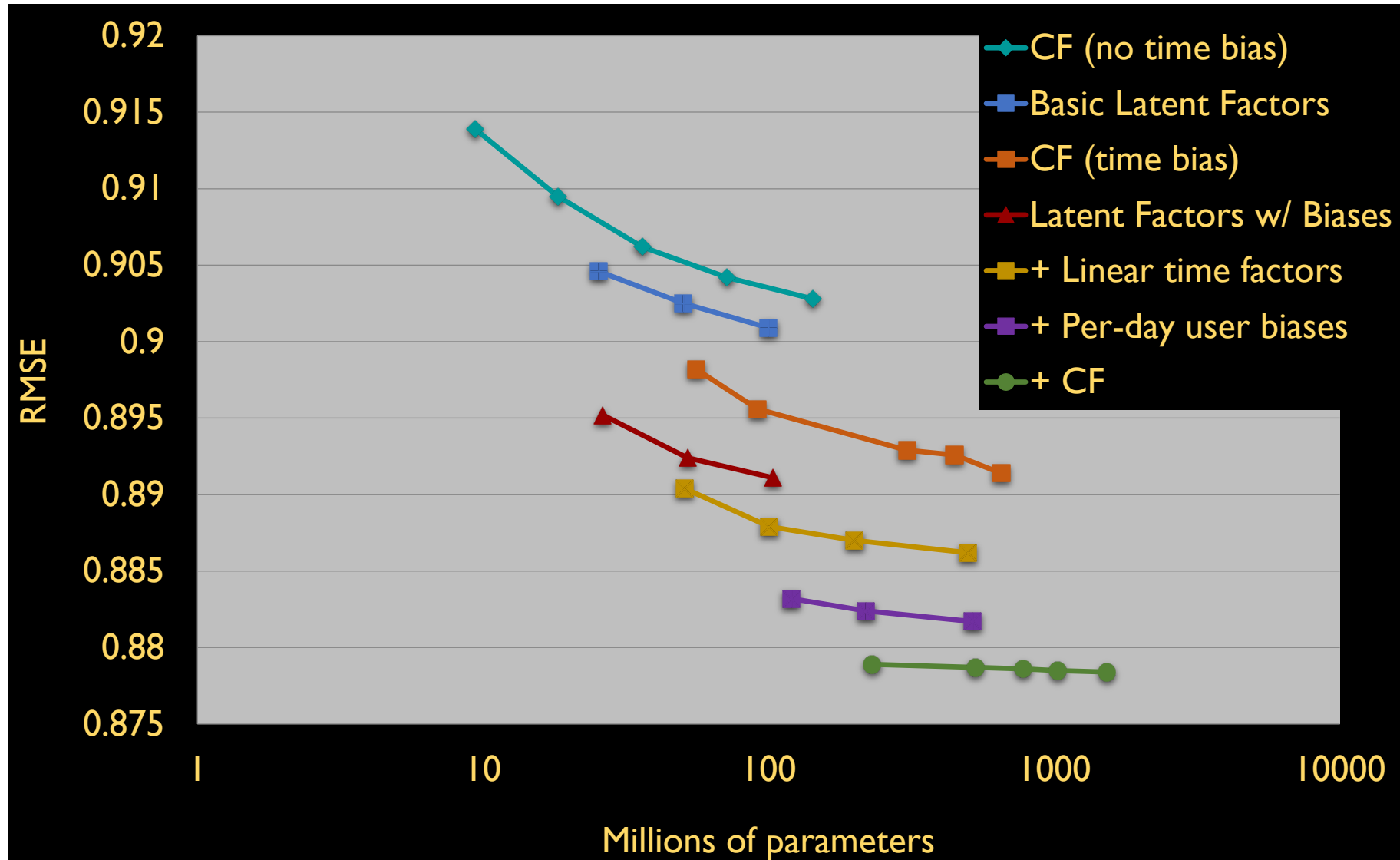 - For not that new movies, people believe even older movies are just inherently better
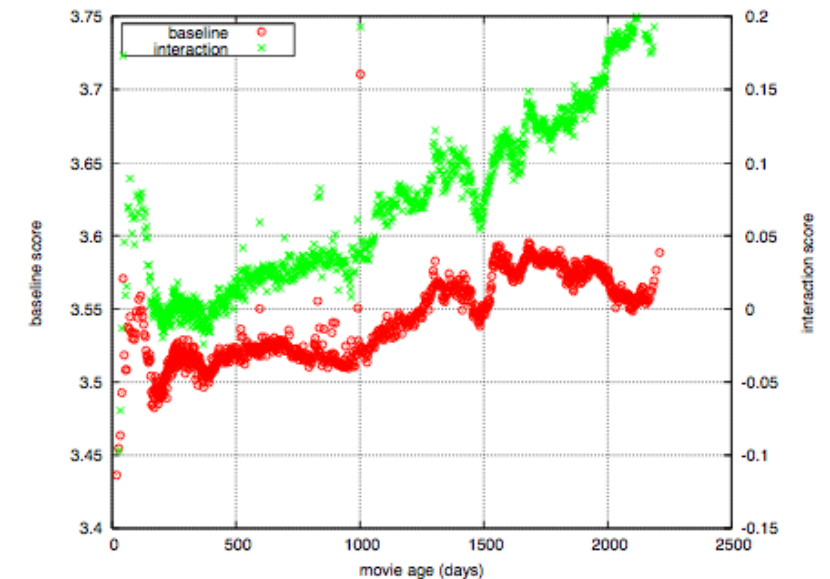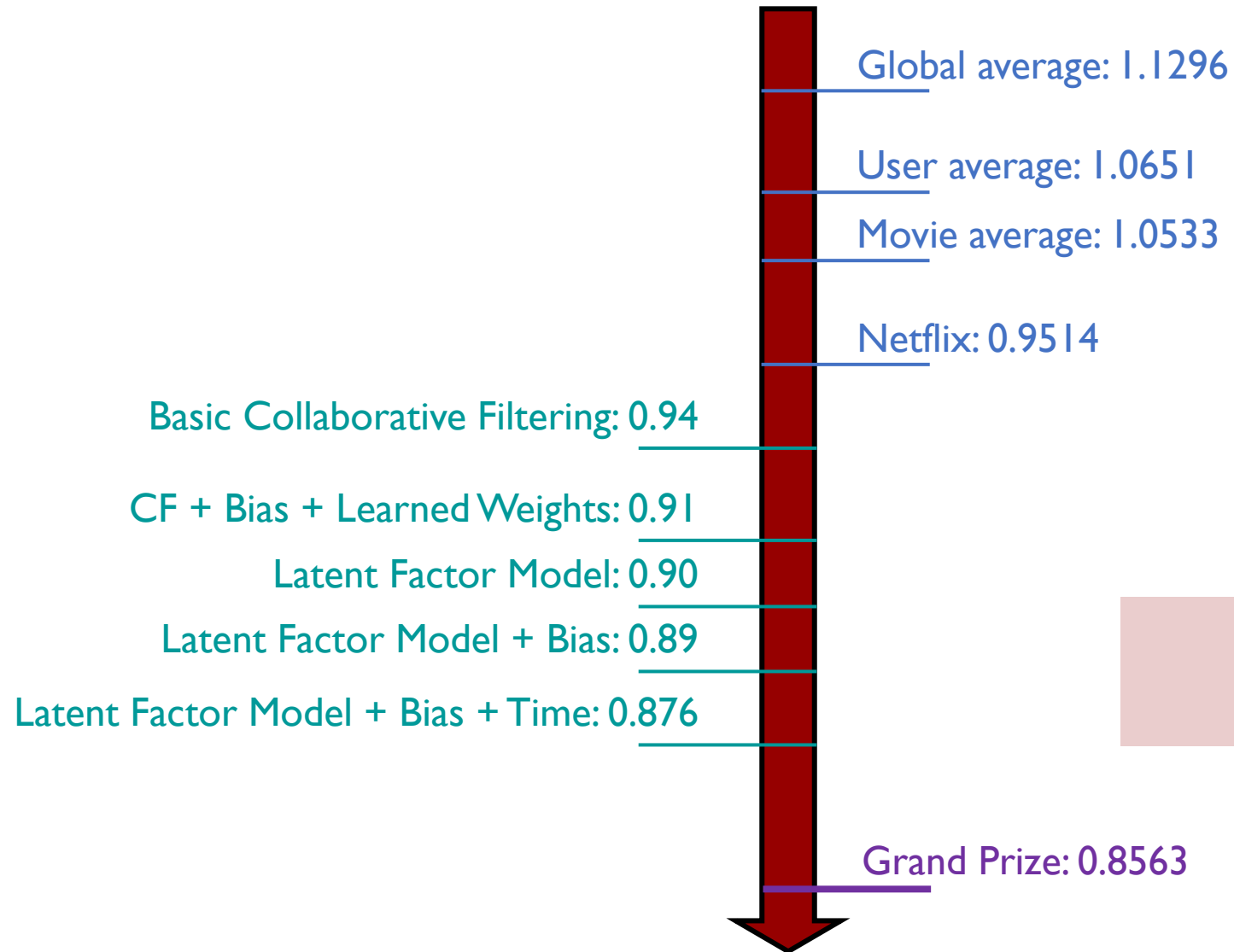
Temporal Biases and Factors

- Latent Factor Model with Constant Bias: $U_{xi} = \mu + b_x + b_i + \boldsymbol{q}_i \boldsymbol{p}_x^T$
- Latent Factor Model with Temporal Bias: $U_{xi} = \mu + b_x(t) + b_i(t) + \boldsymbol{q}_i \boldsymbol{p}_x^T$
  - Make parameters b_x and b_i to depend on time
    - Parameterize time-dependence by linear trends
    - Each bin corresponds to 10 consecutive weeks
      - $b_i(t) = b_i + b_{i,\mathrm{Bin}(t)}$

- One can further add temporal dependence to user/item vectors
  - $\boldsymbol{p}_x(t)$: user preference vector on day t

Adding Temporal Effects

Performance of Various Models

Global average: 1.1296

User average: 1.0651

Movie average: 1.0533

Netflix: 0.9514

Basic Collaborative Filtering: 0.94

CF + Bias + Learned Weights: 0.91

Latent Factor Model: 0.90

Latent Factor Model + Bias: 0.89

Latent Factor Model + Bias + Time: 0.876
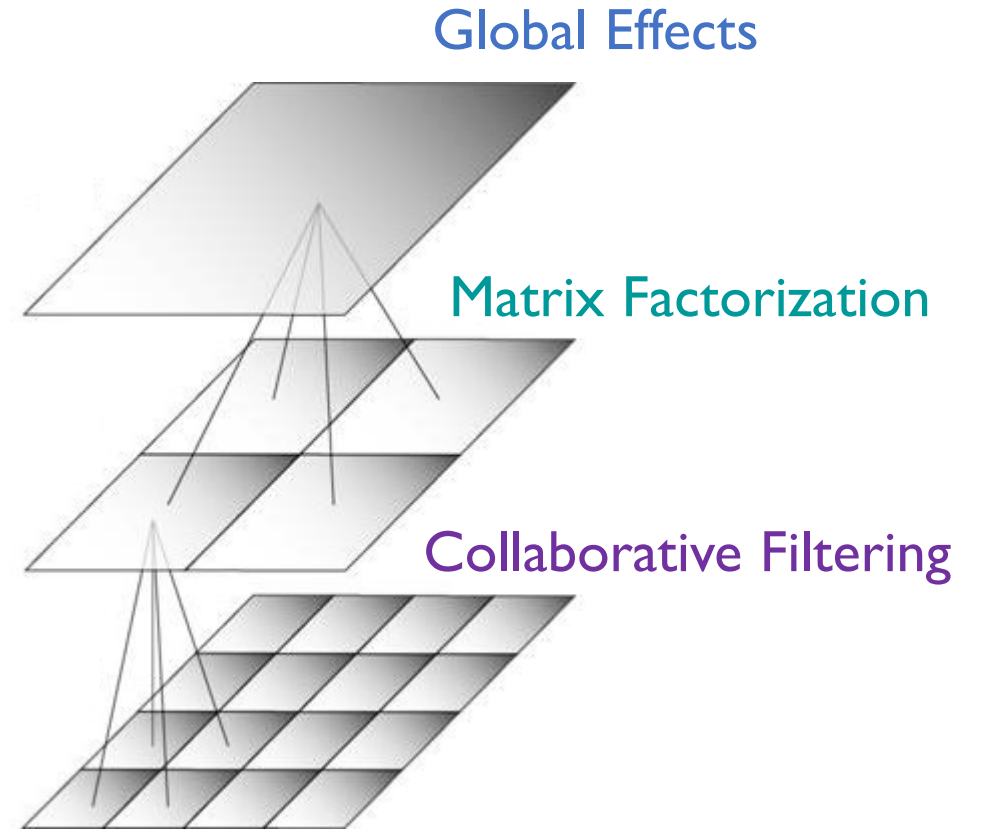
Still no prize!
Getting desperate.

Grand Prize: 0.8563

$ 1 Million Awarded on September 21st, 2009
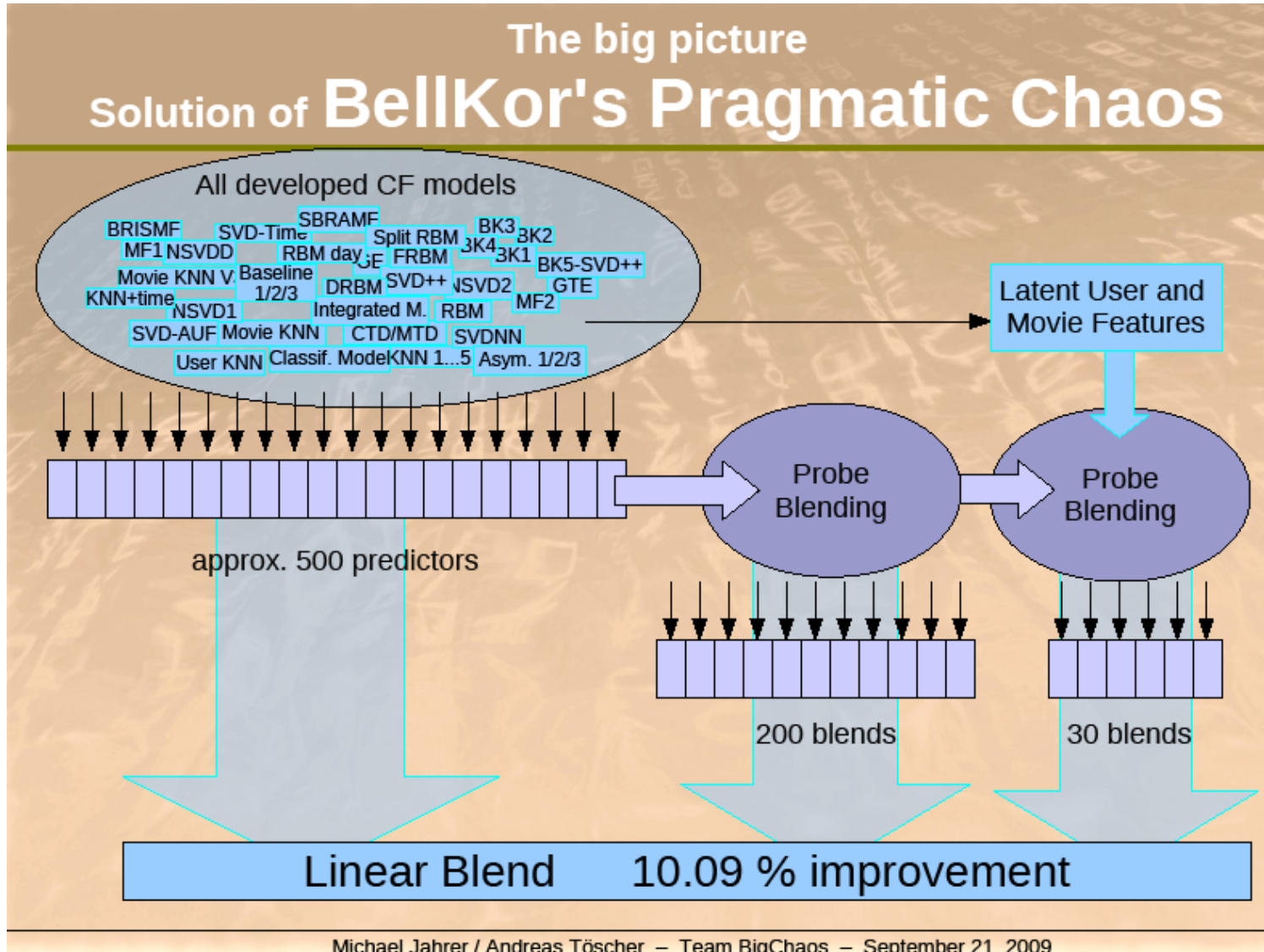
BellKor Recommender System: Rough Idea

- 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
- Matrix Factorization:
  - Addressing "regional" effects
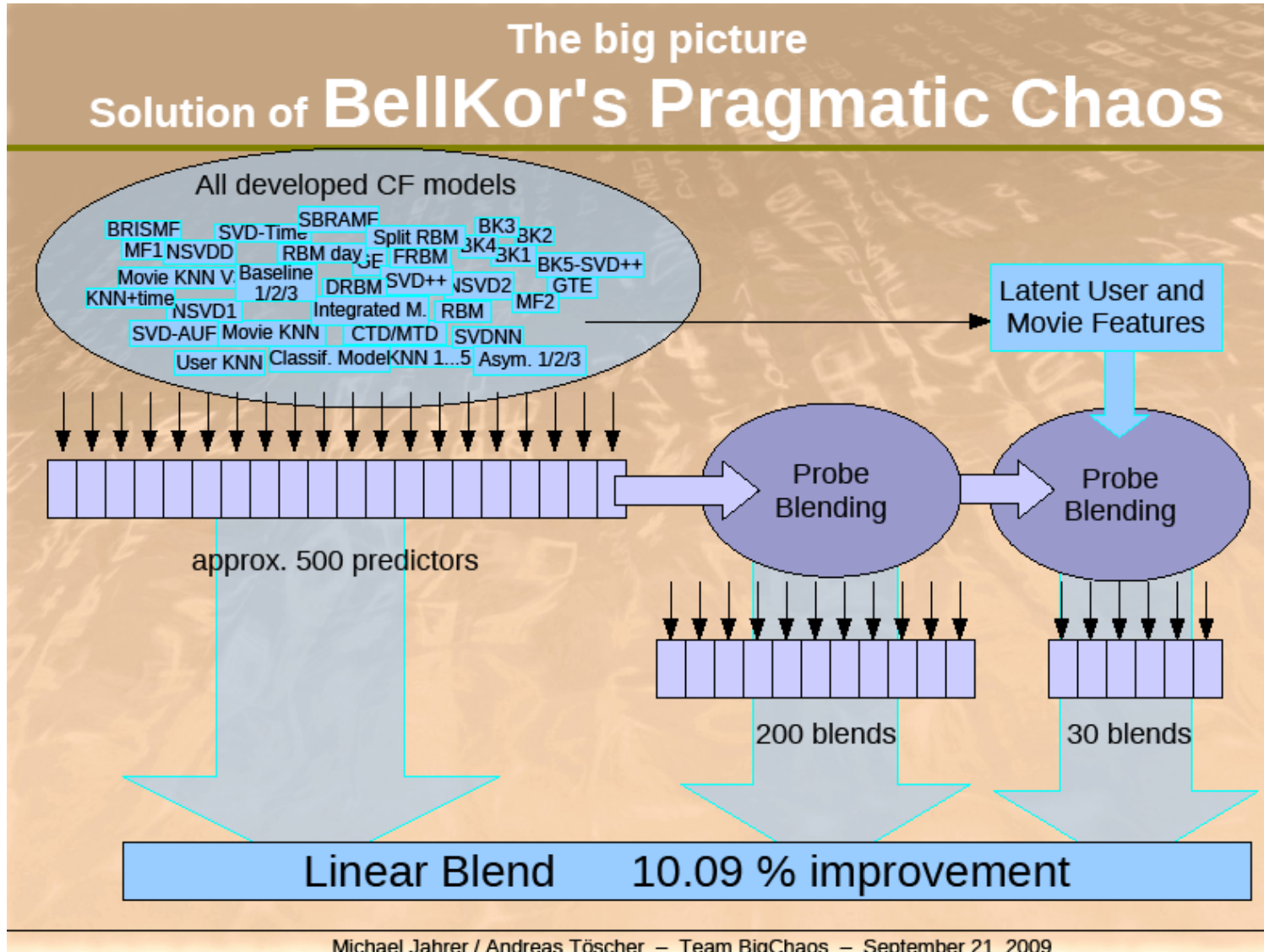- Collaborative Filtering:
  - Extract local patterns

Global Effects

Matrix Factorization

Collaborative Filtering

BellKor Recommender System: A "Kitchen Sink" Approach

The big picture

Solution of **BellKor's Pragmatic Chaos**

All developed CF models

BRISMF   SVD-Time   SBRAMF   Split RBM   BK3 BK2
MF1 NSVDD   RBM day GE FRBM 3K4 3K1 BK5-SVD++
Movie KNN v. Baseline DRBM SVD++ NSVD2   GTE
KNN+time 1/2/3   Integrated M. RBM MF2
NSVD1
SVD-AUF Movie KNN   CTD/MTD SVDNN
User KNN Classif. ModeKNN 1...5 Asym. 1/2/3

Latent User and Movie Features

approx. 500 predictors

Probe Blending → Probe Blending

200 blends   30 blends

Linear Blend   10.09 % improvement

Michael Jahrer / Andreas Töscher – Team BigChaos – September 21, 2009

- Linear combination of many models:
  - Different numbers of factors
  - Different ways to model time
  - Handling implicit feedback
  - Nearest-neighbor models
  - Restricted Boltzmann Machine
  - …

BellKor Recommender System: A "Kitchen Sink" Approach

The big picture
Solution of **BellKor's Pragmatic Chaos**

All developed CF models

BRISMF, SVD-Time, SBRAMF, Split RBM, BK3, BK2, MF1, NSVDD, RBM day, GE, FRBM, 3K4, 3K1, BK5-SVD++, Movie KNN V. Baseline 1/2/3, DRBM, SVD++, NSVD2, GTE, KNN+time, Integrated M., RBM, MF2, NSVD1, SVD-AUF, Movie KNN, CTD/MTD, SVDNN, User KNN, Classif. ModeKNN 1...5, Asym. 1/2/3

Latent User and Movie Features

Probe Blending

Probe Blending

approx. 500 predictors

200 blends

30 blends

Linear Blend   10.09 % improvement

Michael Jahrer / Andreas Töscher – Team BigChaos – September 21, 2009

- For a *research project*, this is a very bad idea (since you don't know which part works or why).

- To *achieve a certain level of model performance* (and win a prize), this might be an unavoidable path to take.

Leaderboard on June 26ᵗʰ, 2009

Netflix Prize

Home | Rules | Leaderboard | Register | Update | Submit | Download

Leaderboard

Display top 20 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
1	BellKor's Pragmatic Chaos	0.8558	10.05	2009-06-26 18:42:37
Grand Prize - RMSE <= 0.8563				
2	PragmaticTheory	0.8582	9.80	2009-06-25 22:15:51
3	BellKor in BigChaos	0.8590	9.71	2009-05-13 08:14:09
4	Grand Prize Team	0.8593	9.68	2009-06-12 08:20:24
5	Dace	0.8604	9.56	2009-04-22 05:57:03
6	BigChaos	0.8613	9.47	2009-06-23 23:06:52
Progress Prize 2008 - RMSE = 0.8616 - Winning Team: BellKor in BigChaos				
7	BellKor	0.8620	9.40	2009-06-24 07:16:02
8	Gravity	0.8634	9.25	2009-04-22 18:31:32
9	Opera Solutions	0.8638	9.21	2009-06-26 23:18:13
10	BruceDengDaoCiYiYou	0.8638	9.21	2009-06-27 00:55:55
11	pengpengzhou	0.8638	9.21	2009-06-27 01:06:43
12	xlvector	0.8639	9.20	2009-06-26 13:49:04
13	xiangliang	0.8639	9.20	2009-06-26 07:47:34
14	Feeds2	0.8641	9.18	2009-06-26 22:51:55
15	Ces	0.8642	9.17	2009-06-24 14:34:14

Performance of Various Models

Global average: 1.1296

User average: 1.0651

Movie average: 1.0533

Netflix: 0.9514

Basic Collaborative Filtering: 0.94

CF + Bias + Learned Weights: 0.91

Latent Factor Model: 0.90

Latent Factor Model + Bias: 0.89

Latent Factor Model + Bias + Time: 0.876

BelKor: 0.8558

Grand Prize: 0.8563
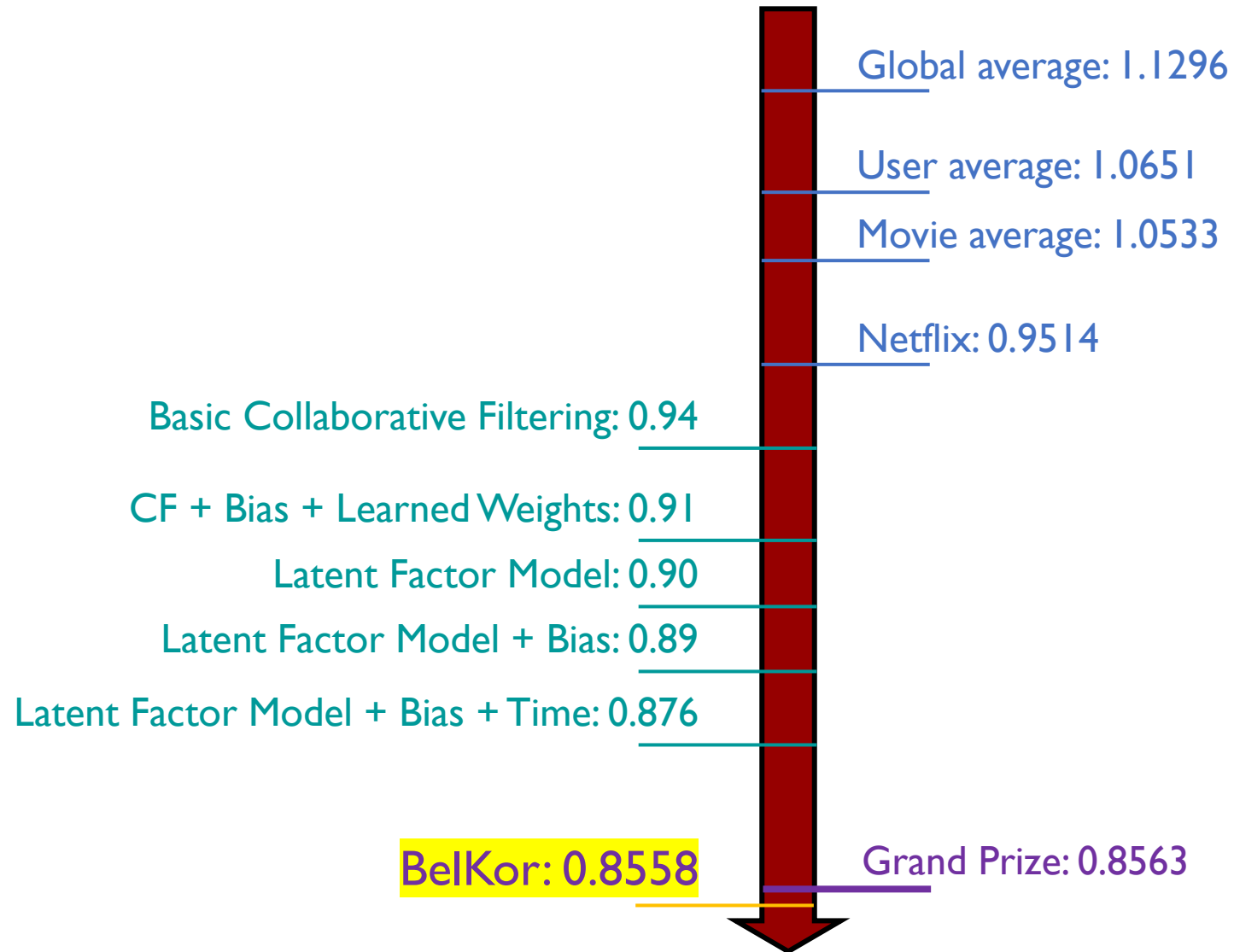
Recommendation with Implicit Feedback

Implicit Feedback

- So far, we have focused mostly on estimating explicit ratings (e.g., 1 star – 5 stars)

- What if we only have implicit feedback?
    - E.g., clicks, likes, views, …
    - Only a (small) fraction of customers who purchase a product actually leave a rating
    - Not to mention that the number of users who merely view or click on it is much larger

    - Challenge 1: No negative feedback!
        - If I have not viewed a YouTube video, does that mean I hate it? Or I just have not been exposed to it yet?
    - Challenge 2: Evaluation is tricky — no RMSE to measure!

Bayesian Personalized Ranking [Rendle et al., UAI 2009]

BPR: Bayesian Personalized Ranking from Implicit Feedback

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner and Lars Schmidt-Thieme
{srendle, freudenthaler, gantner, schmidt-thieme}@ismll.de
Machine Learning Lab, University of Hildesheim
Marienburger Platz 22, 31141 Hildesheim, Germany

Abstract

Item recommendation is the task of predicting a personalized ranking on a set of items (e.g. websites, movies, products). In this paper, we investigate the most common scenario with implicit feedback (e.g. clicks, purchases). There are many methods for item recommendation from implicit feedback like matrix factorization (MF) or adaptive k-nearest-neighbor (kNN). Even though these methods are designed for the item prediction task of personalized ranking, none of

sonalization is attractive both for content providers, who can increase sales or views, and for customers, who can find interesting content more easily. In this paper, we focus on item recommendation. The task of item recommendation is to create a user-specific ranking for a set of items. Preferences of users about items are learned from the user's past interaction with the system – e.g. his buying history, viewing history, etc.

Recommender systems are an active topic of research. Most recent work is on scenarios where users provide explicit feedback, e.g. in terms of ratings. Nevertheless, in real-world scenarios most feedback is not explicit but implicit. Implicit feedback is tracked au-

Big Idea: Let's go back to ranking!

- Our previous discussion has focused on ratings prediction (e.g., optimizing RMSE)
- But with implicit feedback, let's instead focus on how well we rank items

- Example of implicit feedback

Exile on Main St.   *Run the Jewels*   *Pet Sounds*   *Illmatic*   *The Blueprint*

User 1

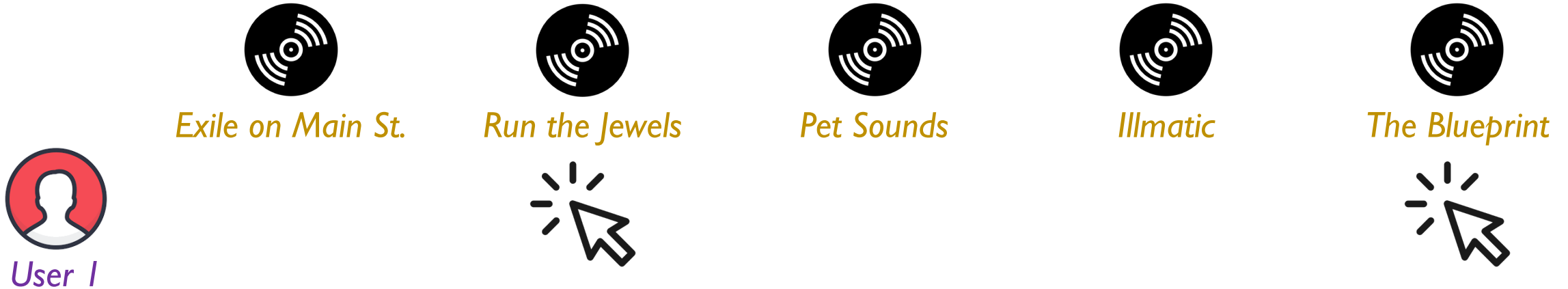# From Implicit Feedback to Item Ranking

	Exile on Main St.	Run the Jewels	Pet Sounds	Illmatic	The Blueprint
		🖱️			🖱️

User 1

- Can we derive the pairwise preferences for *User 1*?

	Exile on Main St.	Run the Jewels	Pet Sounds	Illmatic	The Blueprint
Exile on Main St.	■				
Run the Jewels		■			
Pet Sounds			■		
Illmatic				■	
The Blueprint					■

From Implicit Feedback to Item Ranking

Exile on Main St. Run the Jewels Pet Sounds Illmatic The Blueprint

User 1

- Principle 1: The user prefers each clicked item over all other non-observed items.

	Exile on Main St.	Run the Jewels	Pet Sounds	Illmatic	The Blueprint
Exile on Main St.		+			+
Run the Jewels	-		-	-	
Pet Sounds		+			+
Illmatic		+			+
The Blueprint	-		-	-	

From Implicit Feedback to Item Ranking

Exile on Main St.   **Run the Jewels**   **Pet Sounds**   **Illmatic**   **The Blueprint**

User 1

- **Principle 2:** We cannot infer any preference between two non-observed items.

	Exile on Main St.	Run the Jewels	Pet Sounds	Illmatic	The Blueprint
Exile on Main St.		+	?	?	+
Run the Jewels	-		-	-	
Pet Sounds	?	+		?	+
Illmatic	?	+	?		+
The Blueprint	-		-	-	

From Implicit Feedback to Item Ranking

+ = user prefers the column item
- = user prefers the row item
? = we don't know

Exile on Main St. Run the Jewels Pet Sounds Illmatic The Blueprint

User 1
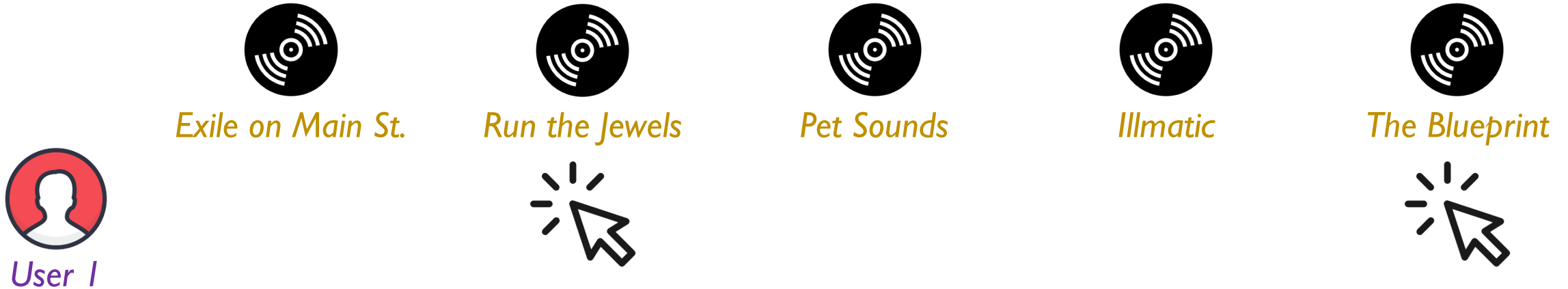
- Principle 3: We cannot infer any preference between two clicked items.

	Exile on Main St.	Run the Jewels	Pet Sounds	Illmatic	The Blueprint
Exile on Main St.		+	?	?	+
Run the Jewels	-		-	-	?
Pet Sounds	?	+		?	+
Illmatic	?	+	?		+
The Blueprint	-	?	-	-	

Consider All Users

- From one user-item matrix
  - + = clicked
  - ? = not observed

- To many item-item matrices
  - + = the user prefers the column item
  - - = the user prefers the row item
  - ? = we don't know

Our Task

- To predict the personalized preference between any two items
    - Given a user u and two items $i \neq j$, is it
    - $i >_u j$, or
    - $i <_u j$?

- Equivalently, to estimate a personalized ranking function $x(u, i, j)$:
    - $x(u, i, j) > 0$ if i is preferred by u
    - $x(u, i, j) < 0$ if j is preferred by u

- How can we know if we are doing a good job?
    - Challenge 2: Evaluation is tricky — no RMSE to measure!

Counting the Successes

- Let's count all the times the function correctly guesses.

- Define $\delta(\hat{x}(u,i,j) > 0) = \begin{cases} 1, & \text{if } x(u,i,j) > 0, \\ 0, & \text{otherwise.} \end{cases}$

  - If the ground truth is $i >_u j$, and you predict $i >_u j$, you get a score of 1.
  - If the ground truth is $i >_u j$, and you predict $i <_u j$, you get a score of 0.

- Enumerate all the cases where $i >_u j$:

$$\text{AUC}(u) = \frac{1}{|I_u^+||I \backslash I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \backslash I_u^+} \delta(\hat{x}(u,i,j) > 0)$$

I_u^+: the set of items clicked by u

"Area Under the Curve" (AUC)

AUC

$$\mathrm{AUC}(u) = \frac{1}{|I_u^+||I \backslash I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \backslash I_u^+} \delta(\hat{x}(u,i,j) > 0)$$

- AUC essentially counts how many times the model correctly identifies that u prefers the item they clicked (positive feedback) over the item they did not.

- What is the AUC of a perfect model?
    - 1

- What is the AUC of a random guesser?
    - 0.5

- Then we can find the average AUC for all users

$$\mathrm{AUC} = \frac{1}{|U|} \sum_{u \in U} \mathrm{AUC}(u)$$

AUC

$$\text{AUC}(u) = \frac{1}{|I_u^+||I \backslash I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \backslash I_u^+} \delta(\hat{x}(u, i, j) > 0)$$

- But how to develop a model to calculate $\hat{x}(u, i, j)$?

- Recall pairwise learning to rank (e.g., RankNet)

  - Given a query q and two documents d_1 and d_2

  - Step 1: We calculate $\text{score}(q, d_1)$ and $\text{score}(q, d_2)$

  - Step 2: We calculate their difference $\text{score}(q, d_1) - \text{score}(q, d_2)$

  - Step 3: Based on this difference and the ground truth, we train the model using gradient descent.

$\text{Prob}(d_1 > d_2)$

$\text{score}(q, d_1) - \text{score}(q, d_2)$
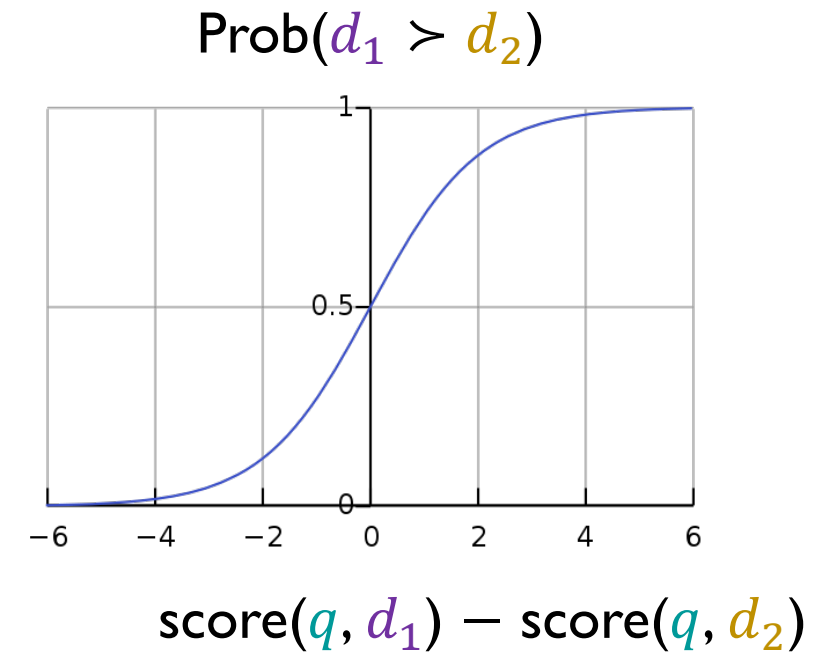
Extending the Idea

- Given a user u and two items i and j, we want to calculate $\hat{x}(u, i, j)$

$$\hat{x}(u, i, j) = \hat{x}(u, i) - \hat{x}(u, j)$$

- $\hat{x}(u, i)$ larger if u likes i to a greater extent
- $\hat{x}(u, i)$ smaller if u likes i to a lesser extent
- Independent of $\hat{x}(u, j)$

Example

- Suppose
  - $\hat{x}(u, \text{Exile on Main St.}) = 2$
  - $\hat{x}(u, \text{Run the Jewels}) = 8$
  - $\hat{x}(u, \text{Illmatic}) = 3$
  - $\hat{x}(u, \text{The Blueprint}) = 7$
- What is $\text{AUC}(u)$?

		2	8	3	7
		Exile on Main St.	Run the Jewels	Illmatic	The Blueprint
2	Exile on Main St.	■	+	?	+
8	Run the Jewels	-	■	-	?
3	Illmatic	?	+	■	+
7	The Blueprint	-	?	-	■

Example

		2	8	3	7
		Exile on Main St.	*Run the Jewels*	*Illmatic*	*The Blueprint*
2	*Exile on Main St.*	■	+	?	+
8	*Run the Jewels*	-	■	-	?
3	*Illmatic*	?	+	■	+
7	*The Blueprint*	-	?	-	■

- All 8 entries are correctly predicted, so $\mathrm{AUC}(u) = \frac{1}{8} \times 8 = 1$

		3	7	2	8
		Exile on Main St.	*Run the Jewels*	*Illmatic*	*The Blueprint*
3	*Exile on Main St.*	■	+	?	+
7	*Run the Jewels*	-	■	-	?
2	*Illmatic*	?	+	■	+
8	*The Blueprint*	-	?	-	■

Example

- What are the rankings in the previous two examples?
    - [*Run the Jewels*, *The Blueprint*, *Illmatic*, *Exile on Main St.*]
    - [*The Blueprint*, *Run the Jewels*, *Exile on Main St*, *Illmatic*,]
    - Both are [Clicked, Clicked, Not Clicked, Not Clicked]

- What is the AUC if we have [Clicked, Not Clicked, Clicked, Not Clicked]?
    - AUC = $\frac{1}{4}(1 + 1 + 0 + 1) = 0.75$

How to calculate $\hat{x}(u, i)$?

One Idea: Matrix Factorization!

- $\hat{x}(u,i) = \boldsymbol{q}_i \boldsymbol{p}_u^T$

- $\hat{x}(u,i,j) = \hat{x}(u,i) - \hat{x}(u,j) = \boldsymbol{q}_i \boldsymbol{p}_u^T - \boldsymbol{q}_j \boldsymbol{p}_u^T = (\boldsymbol{q}_i - \boldsymbol{q}_j)\boldsymbol{p}_u^T$

- **Task**: Learning all user vectors $\boldsymbol{p}_x^T$ and item vectors $\boldsymbol{q}_i$ from implicit feedback

- Did we see anything similar (but not quite the same) before?
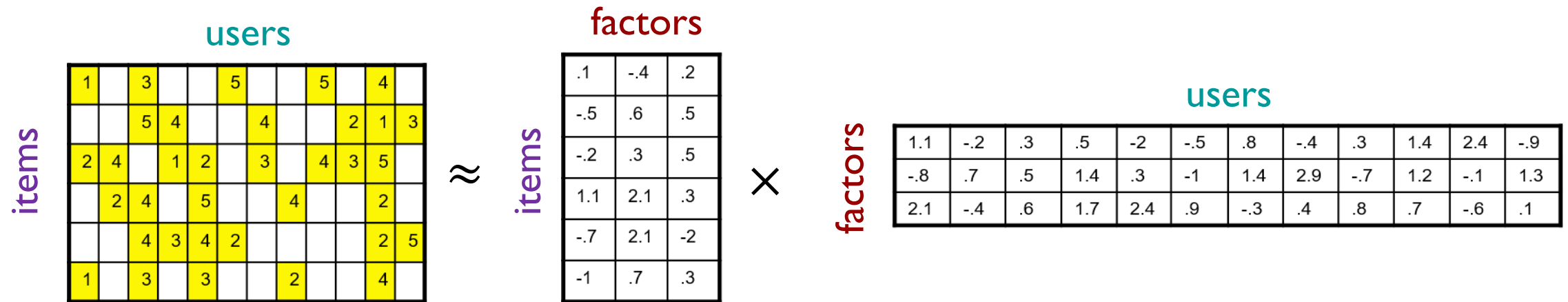
Slide 32 in Week 5: Training a Linear SVM for Ranking

- But we don't have pointwise training data!
  - Remember we only have lots of (c_i, c_k), where we already know $f(\psi_i) > f(\psi_k)$
  - We don't know the value of $f(\psi_i)$ or $f(\psi_k)$

- Idea: Create a new instance space from pairwise learning
  - We have $c_i > c_k \iff f(\psi_i) > f(\psi_k)$
  - We also have $f(\psi_i) = w^T \psi_i$ and $f(\psi_k) = w^T \psi_k$
  - So $c_i > c_k \iff w^T \psi_i > w^T \psi_k \iff w^T(\psi_i - \psi_k) > 0$
  - Let's create a new instance $\phi_u = \psi_i - \psi_k$
  - And $z_u = +1, 0, -1$ as $c_i >, =, < c_k$
  - From training data $\mathcal{S} = \{\phi_u\}$, we train an SVM

Learning Objective

- Task: Learning all user vectors $\boldsymbol{p}_x^T$ and item vectors $\boldsymbol{q}_i$ from implicit feedback
- What is our learning objective?
  - When we have explicit feedback:
    - Evaluation metric: RMSE
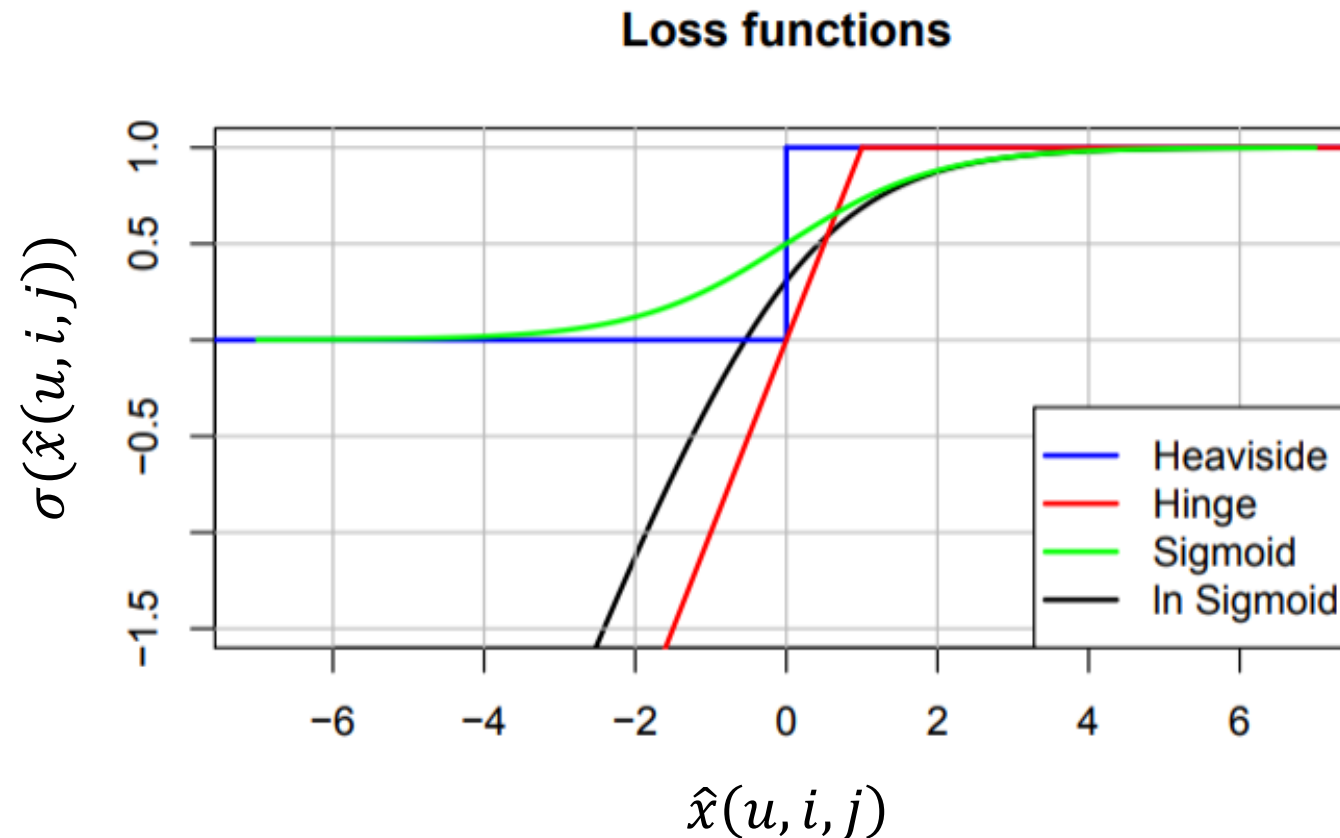   - Learning objective: RMSE
  - When we have implicit feedback:
    - Evaluation metric: AUC

$$\text{AUC}(u) = \frac{1}{|I_u^+||I \backslash I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \backslash I_u^+} \delta(\hat{x}(u, i, j) > 0)$$

  - Learning objective: Can we use AUC?
    - No! Because $\delta(\cdot)$ is not differentiable.

Can we replace $\delta(\hat{x}(u,i,j) > 0)$ with something similar?

- Instead of a counting function $\delta(\hat{x}(u,i,j) > 0)$ (1 or 0), let's pick a smooth function $\sigma(\hat{x}(u,i,j))$

Loss functions

BPR uses Log Sigmoid

$$\sigma(x) = \ln \frac{1}{1 + e^{-x}}$$

- Then the learning objective becomes:

$$\hat{x}(u, i, j)$$

$$J = \frac{1}{|I_u^+||I \backslash I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \backslash I_u^+} \sigma \left((\boldsymbol{q}_i - \boldsymbol{q}_j) \boldsymbol{p}_u^T \right)$$

- Gradient Descent

$$\frac{\partial J}{\partial \theta} = \frac{1}{|I_u^+||I \backslash I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \backslash I_u^+} \frac{\mathrm{d}\sigma}{\mathrm{d}\hat{x}} \cdot \frac{\partial \hat{x}}{\partial \theta}$$

- What is $\frac{\mathrm{d}\sigma}{\mathrm{d}\hat{x}}$?

  - $\frac{\mathrm{d}\sigma}{\mathrm{d}\hat{x}} = \frac{e^{-\hat{x}}}{1 + e^{-\hat{x}}}$

BPR uses Log Sigmoid

- Then the learning objective becomes:

$$\hat{x}(u, i, j)$$

$$J = \frac{1}{|I_u^+||I \backslash I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \backslash I_u^+} \sigma \left(\boxed{(\boldsymbol{q}_i - \boldsymbol{q}_j)\boldsymbol{p}_u^T} \right)$$

- Gradient Descent

$$\frac{\partial J}{\partial \theta} = \frac{1}{|I_u^+||I \backslash I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \backslash I_u^+} \frac{\mathrm{d}\sigma}{\mathrm{d}\hat{x}} \cdot \frac{\partial \hat{x}}{\partial \theta}$$

- What is $\frac{\partial \hat{x}}{\partial \theta}$?

  - If θ is $\boldsymbol{q}_i$, $\quad \frac{\partial \hat{x}}{\partial \theta} = \boldsymbol{p}_u^T$

  - If θ is $\boldsymbol{q}_j$, $\quad \frac{\partial \hat{x}}{\partial \theta} = -\boldsymbol{p}_u^T$

  - If θ is $\boldsymbol{p}_u^T$, $\quad \frac{\partial \hat{x}}{\partial \theta} = \boldsymbol{q}_i - \boldsymbol{q}_j$
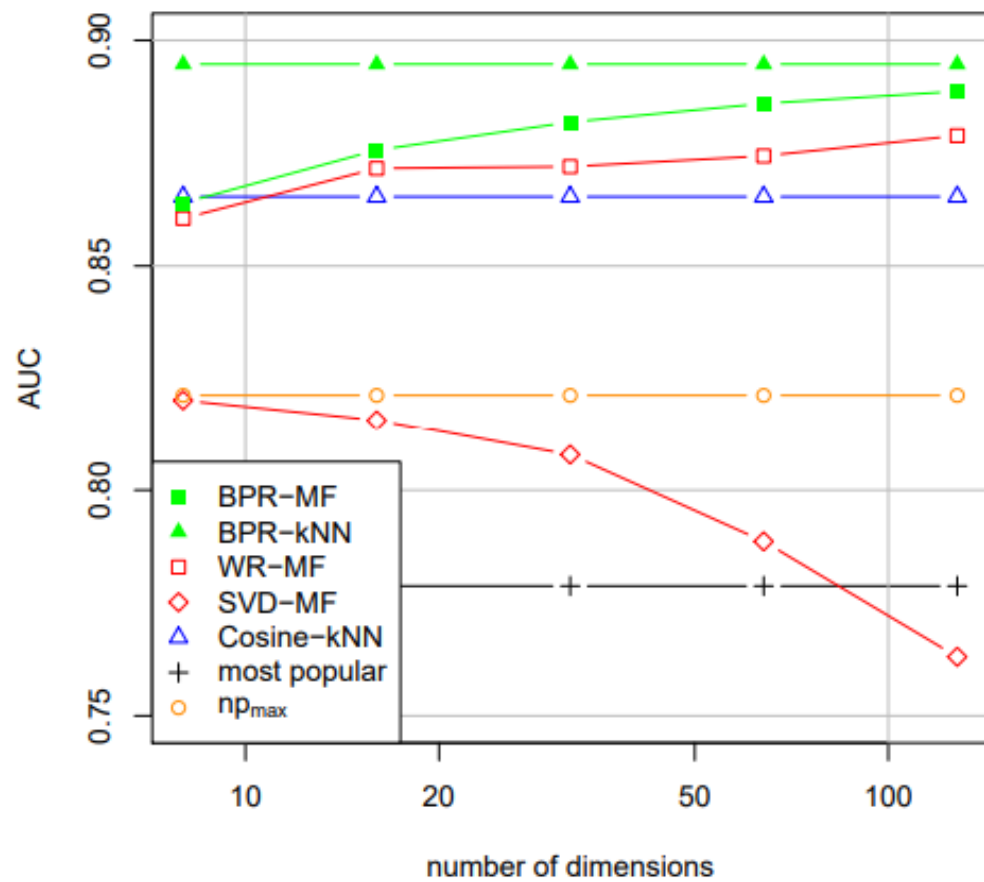
The Gradient Descent Process

1: **procedure** LEARNBPR(D_S, Θ)
2:     initialize Θ
3:     **repeat**
4:         draw (u, i, j) from D_S
5:         $\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda_\Theta \cdot \Theta \right)$
6:     **until** convergence
7:     **return** $\hat{\Theta}$
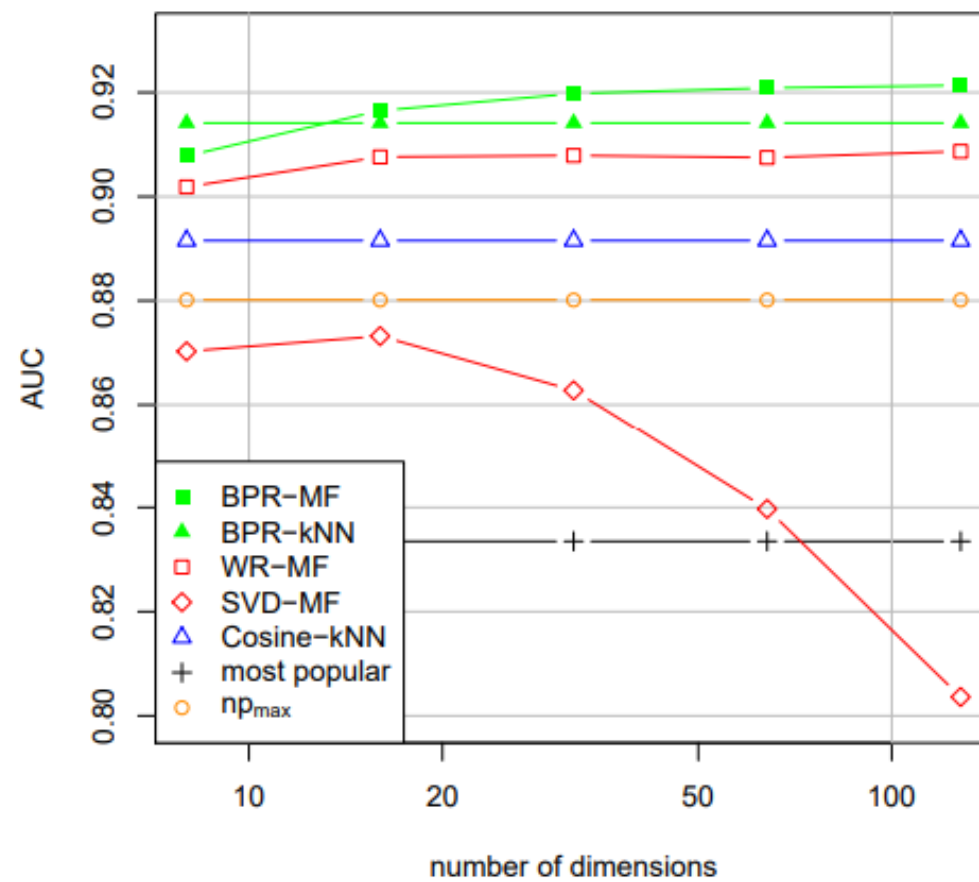8: **end procedure**

Regularization:
$\lambda_\Theta \|\Theta\|^2$ in the objective

Figure 4: Optimizing models for BPR with bootstrapping based stochastic gradient descent. With learning rate α and regularization λ_Θ.

Does BPR really work?

Takeaways

- BPR is a foundational method for recommendation with implicit feedback.

- BPR optimizes for ranking not for ratings.

- We can build BPR on top of other recommendation methods, but a classic approach uses Matrix Factorization.

Going Back to Search Engines

Can RecSys techniques inspire us to improve our search engine?

- What problems does our lexical-based search engine (TF-IDF and BM25) have now?

- Query: *"large language models"*

- TF-IDF and BM25 can only find documents containing *"large"* or *"language"* or *"models"*

- How about the following documents?

    - *"ChatGPT is a generative artificial intelligence chatbot developed by OpenAI and released in 2022."*

    - *"In-the-Flow Agentic System Optimization for Effective Planning and Tool Use"*

- Some words are semantically close, but not lexically the same (or similar).

    - E.g., *"car"* vs. *"automobile"*; *"agricultural"* vs. *"farming"*

- Some documents are semantically close, but do not share any common words.

Document-Word Occurrence Matrix

- Given a document d and a word w, let's define

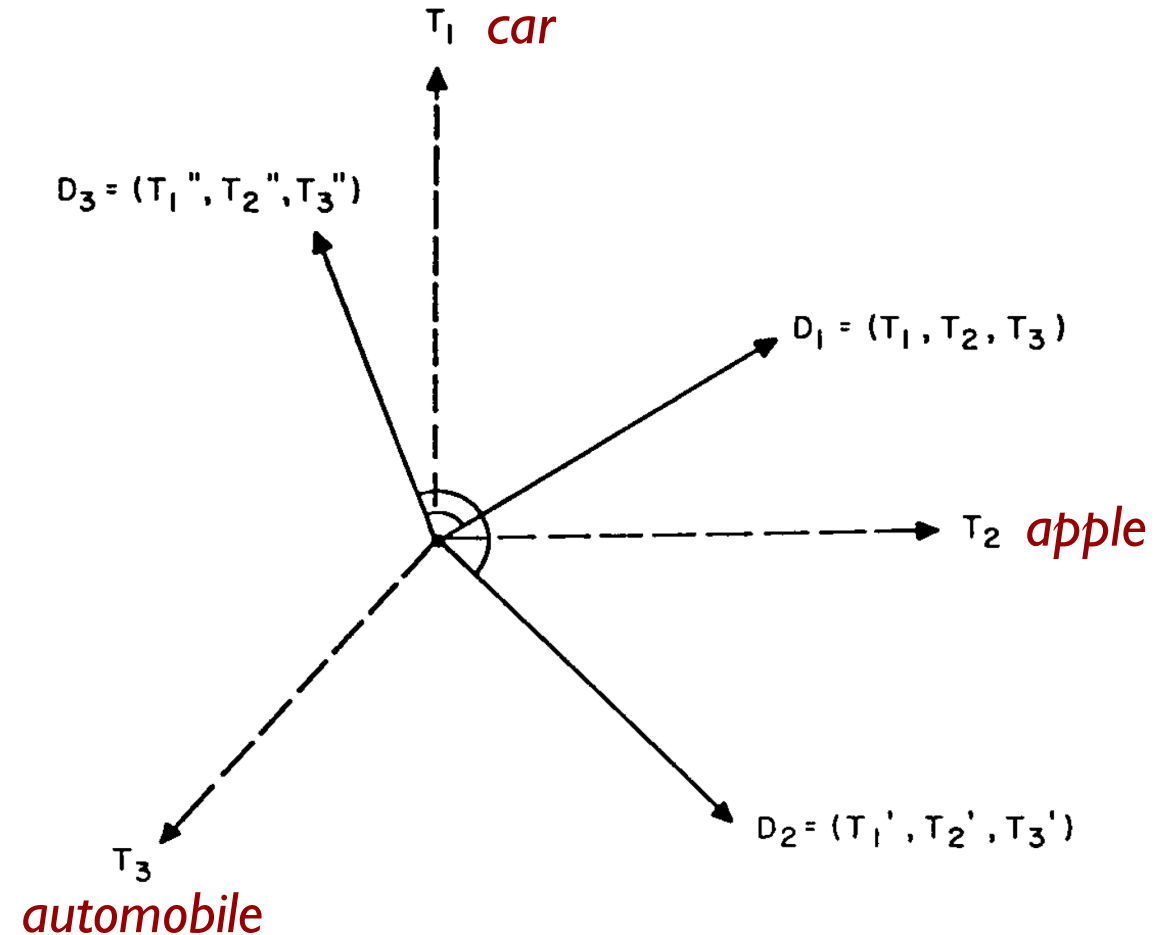$$U_{dw} = \begin{cases} 1, & \text{if } w \text{ appears in } d \\ 0, & \text{otherwise} \end{cases}$$

	large	language	models	chatgpt	generative	openai
Doc 1	1	1	1	0	0	0
Doc 2	0	0	0	1	1	1

- Doc 1 and Doc 2 are totally irrelevant according to TF-IDF and BM25. Why?
  - Because we directly use the corresponding row to represent each document.
  - Doc 1: [1, 1, 1, 0, 0, 0]
  - Doc 2: [0, 0, 0, 1, 1, 1]
  - The inner product or cosine similarity between them is 0.

Recap: Vector Space Model

- A document is more relevant to a query if they are more "similar" in the vector space.

- What are the axes?

- How about each dimension representing a word in the vocabulary?
  - In this case, we assume that any two distinct words are orthogonal to each other!
  - cos(*car*, *apple*) = 0
  - cos(*car*, *automobile*) = 0
  - cos(*car*, *car*) = 1

Fig. 1. Vector representation of document space.

T_1 *car*

$D_3 = (T_1'', T_2'', T_3'')$

$D_1 = (T_1, T_2, T_3)$

T_2 *apple*

$D_2 = (T_1', T_2', T_3')$

T_3
automobile

How about Recommender Systems?

- If we were to adopt the idea of lexical matching, then

$$U_{xi} = \begin{cases} 1, & \text{if user } x \text{ has purchased item } i \\ 0, & \text{otherwise} \end{cases}$$

- The task of a recommender system is to suggest items that the user has not purchased. If we were to predict all unpurchased items as 0, there would be nothing to recommend. Therefore, the methods we introduced earlier clearly do not operate in this manner.
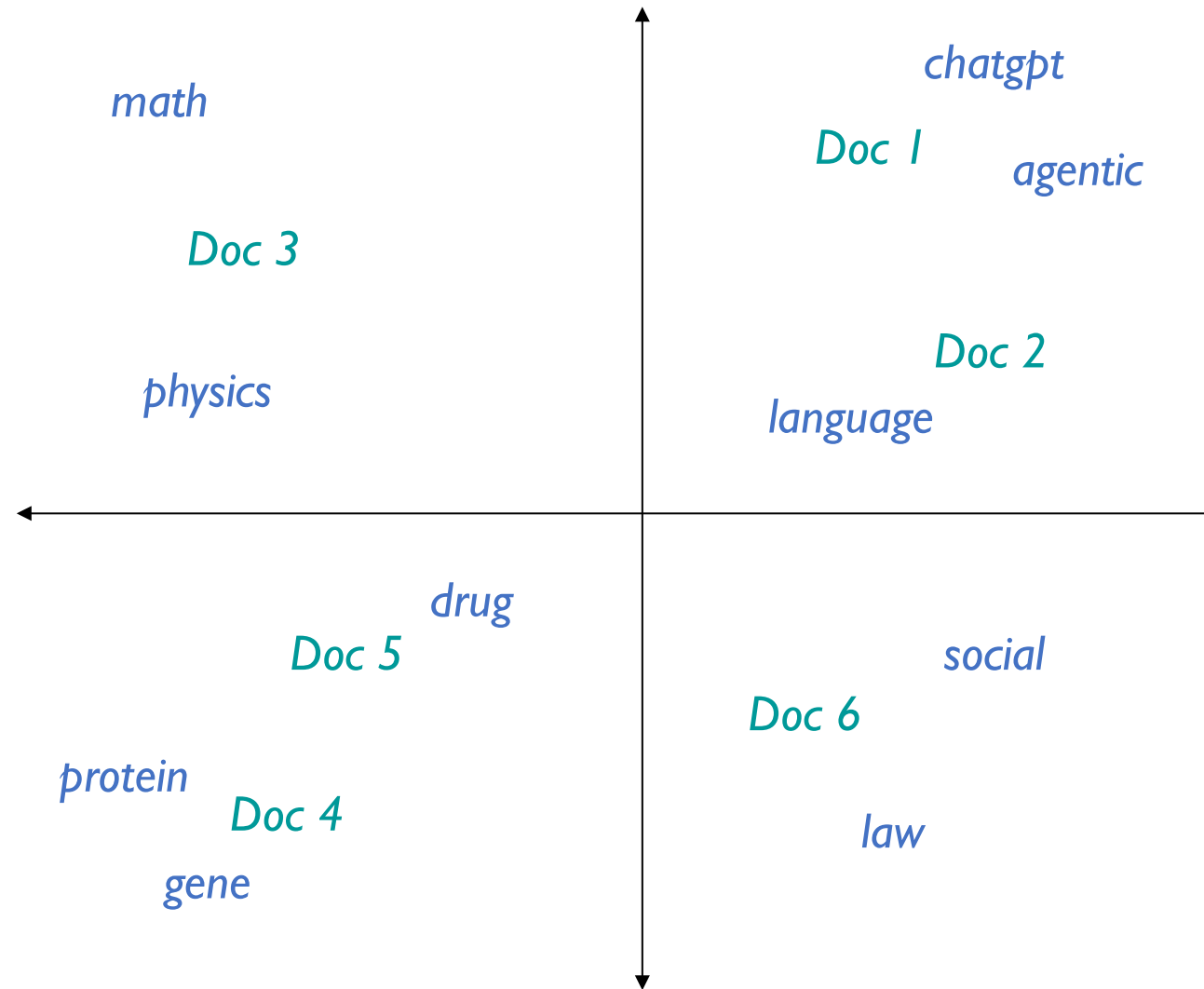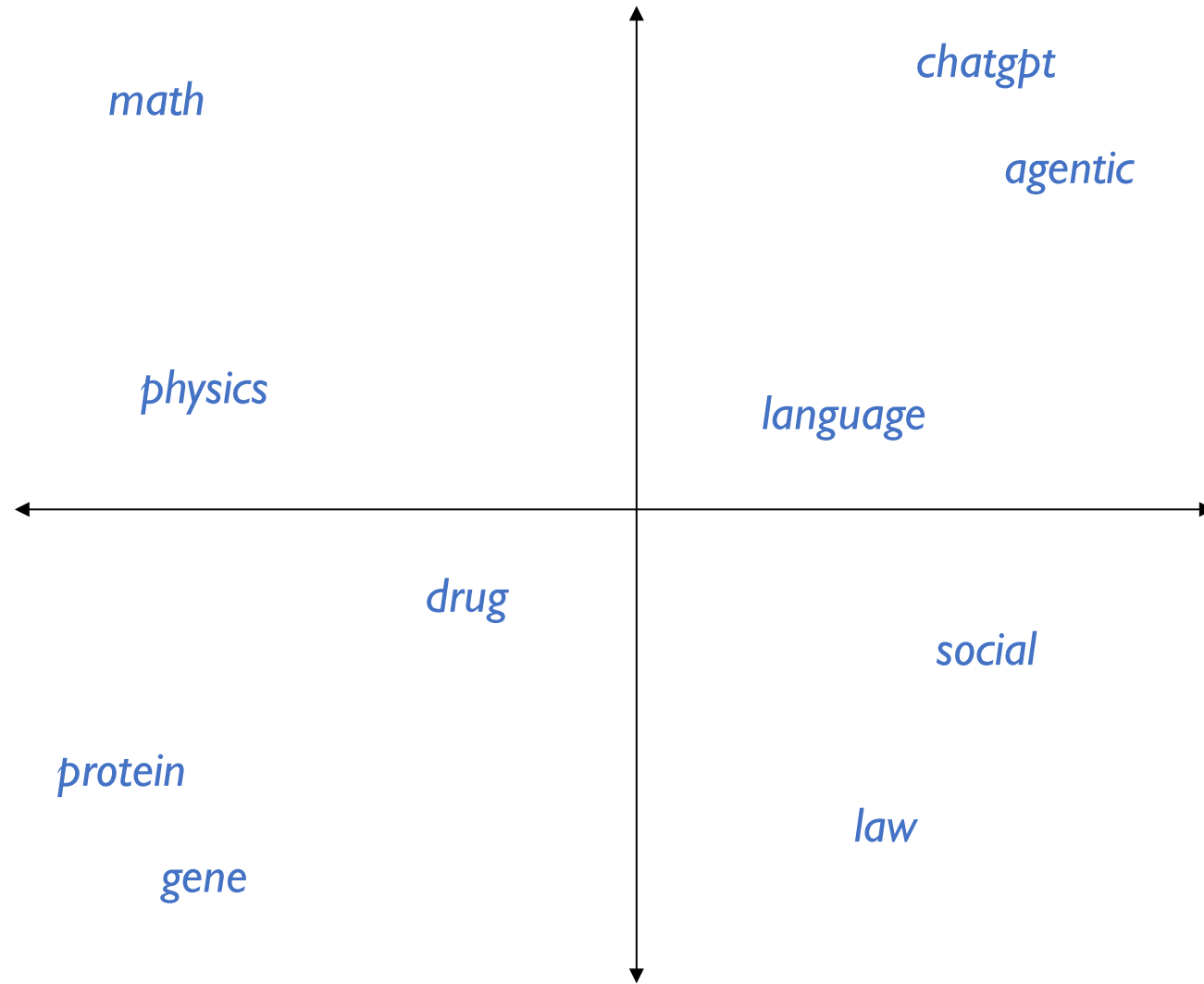
- What did we do?
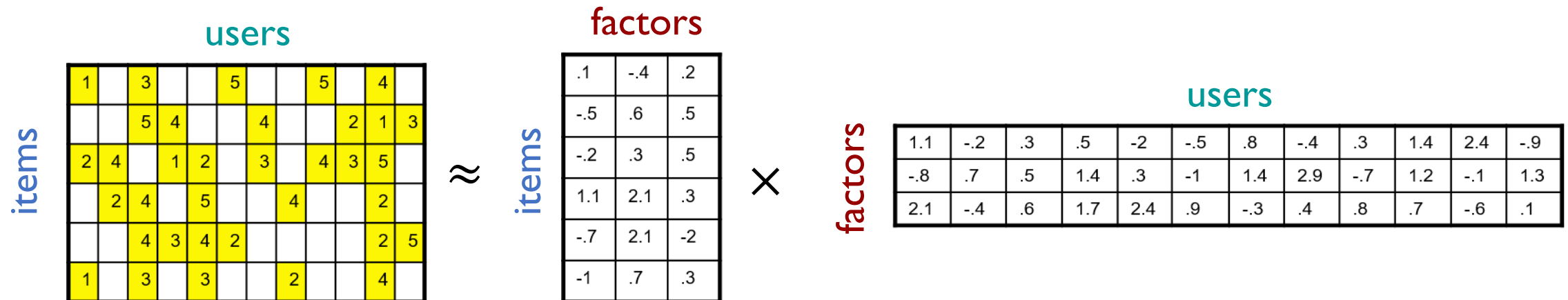
Latent Space for Users and Items
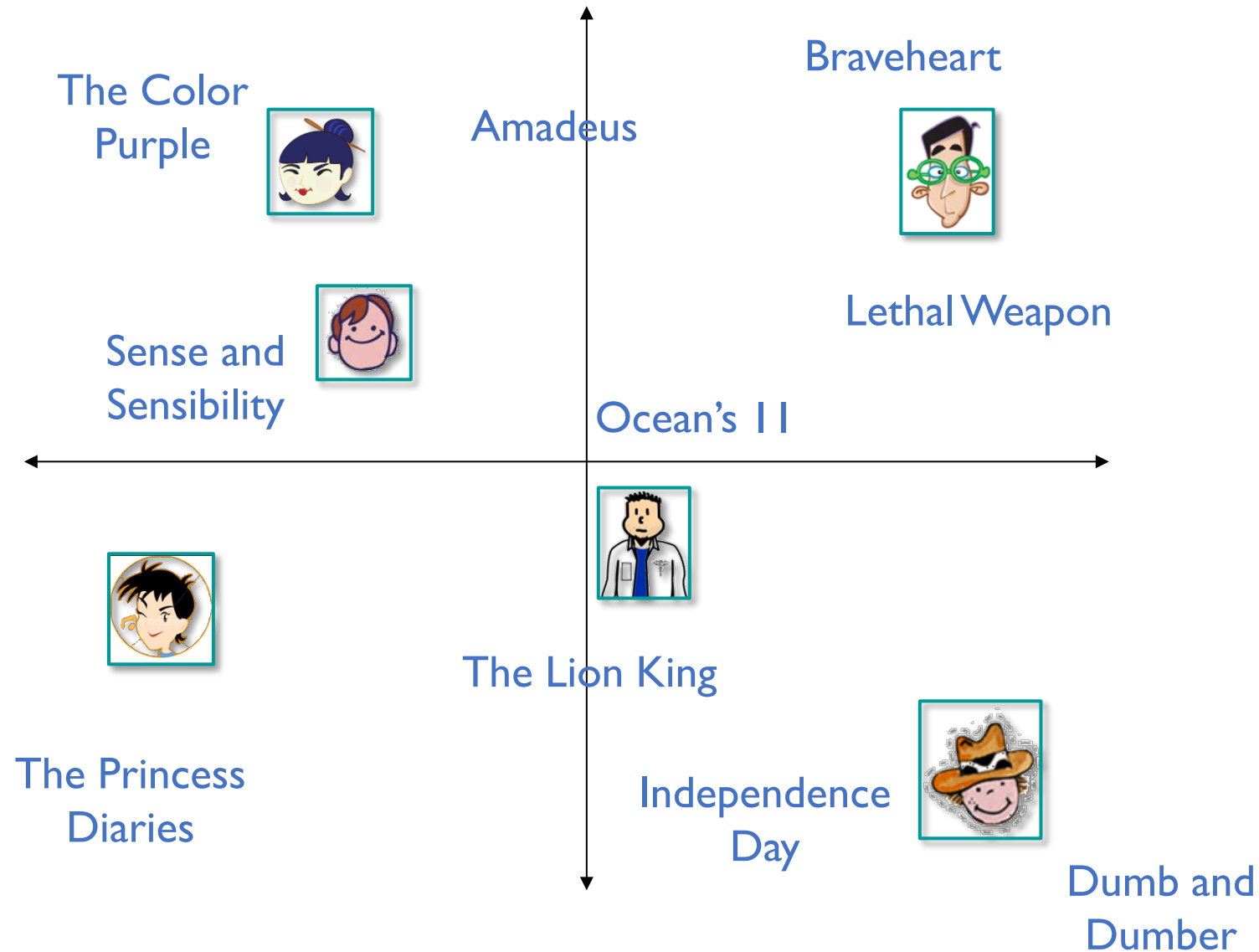
Latent Space for Words and Documents?

Latent Space for Words Only?

chatgpt

math

agentic

physics

language

drug

social

protein

law

gene

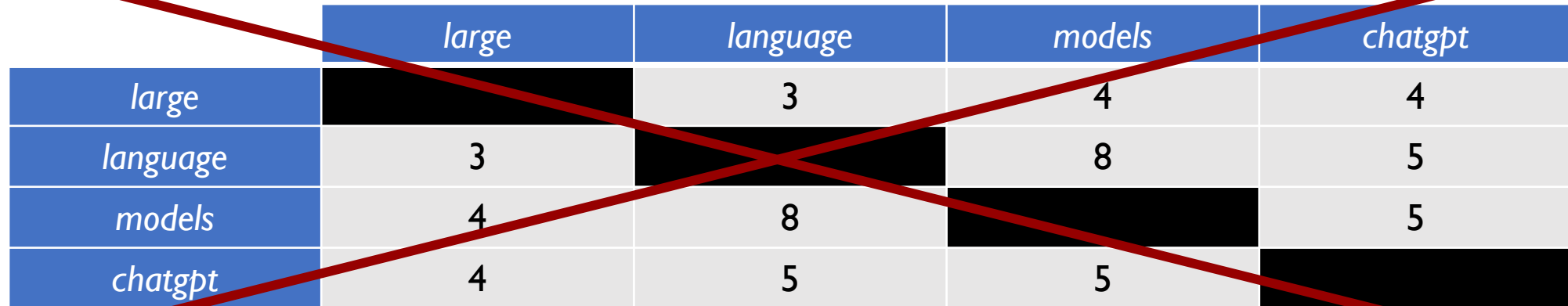# Latent Space for Words Only

- Question 1: Given a new document or a new query, how can we derive its vector?
  - Average of word vectors
  - Vector("*large language models*") = $\frac{1}{3}$ (Vector(*large*) + Vector(*language*) + Vector(*models*))

- Question 2: Which matrix shall we factorize in this case?
  - Word-word co-occurrence matrix
  - How do we define "co-occur"?
    - Two words "co-occur" if they are in the same document.
    - Two words "co-occur" if they are no more 5 words apart in a document.
      - word2vec [Mikolov et al., NIPS 2013]
      - GloVe [Pennington et al., EMNLP 2014]
    - …

Word-Word Co-occurrence Matrix

- In your imagination, this matrix might look as follows:

	large	language	models	chatgpt
large	⬛	3	4	4
language	3	⬛	8	5
models	4	8	⬛	5
chatgpt	4	5	5	⬛

- [Levy and Golberg, NIPS 2014] word2vec is mathematically equivalent to factorizing a word-word matrix U, where

$$U_{xy} = \log \frac{\#(x,y)|\mathcal{D}|}{\#(x)\#(y)} - \log b$$

Next Lecture: Word Embedding, particularly word2vec

Announcing the NeurIPS 2023 Paper Awards

COMMUNICATIONS CHAIRS 2023 / 2023 Conference / awards, neurips2023

Test of Time

This year, following the usual practice, we chose a NeurIPS paper from 10 years ago to receive the Test of Time Award, and **"Distributed Representations of Words and Phrases and their Compositionality"** by Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean, won.

Published at NeurIPS 2013 and cited over 40,000 times, the work introduced the seminal word embedding technique word2vec. Demonstrating the power of learning from large amounts of unstructured text, the work catalyzed progress that marked the beginning of a new era in natural language processing.

Next Lecture: Word Embedding, particularly word2vec

- "*… Demonstrating the power of learning from large amounts of unstructured text, the work catalyzed progress that marked the beginning of a new era in natural language processing.*"

- What should you do when you have no labels at all but still want to find a classification task to train your model?
  - Mask a word in your text and predict it using its context.

ChatGPT is a [MASK] artificial intelligence chatbot

generative

  - What is the label space in this classification task?
    - The entire vocabulary!

Quiz 2!

- All policies are the same as Quiz 1 (number of questions, time limit, grading, etc.)

- Scope:
  - Week 5 (Learning to Rank)
  - Week 6 (Content-Based Recommendation, Collaborative Filtering, Matrix Factorization)
  - Week 7 (Bayesian Personalized Ranking)
  - Homework 1

Thank You!

Course Website: https://yuzhang-teaching.github.io/CSCE670-S26.html