Matrix Factorization

Jie Tang

Department of Computer Science and Technology
Tsinghua University

Some slides are from Alex Smola's Recommender Systems

Netflix Prize

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	?

Rating Prediction

- Least mean squares prediction error
  - Easy to define

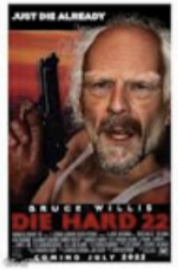
$$\mathrm{rmse}(S) = \sqrt{|S|^{-1} \sum_{(i,u)\in S} (\hat{r}_{ui} - r_{ui})^2}$$

  - Wrong measure for composing sessions!











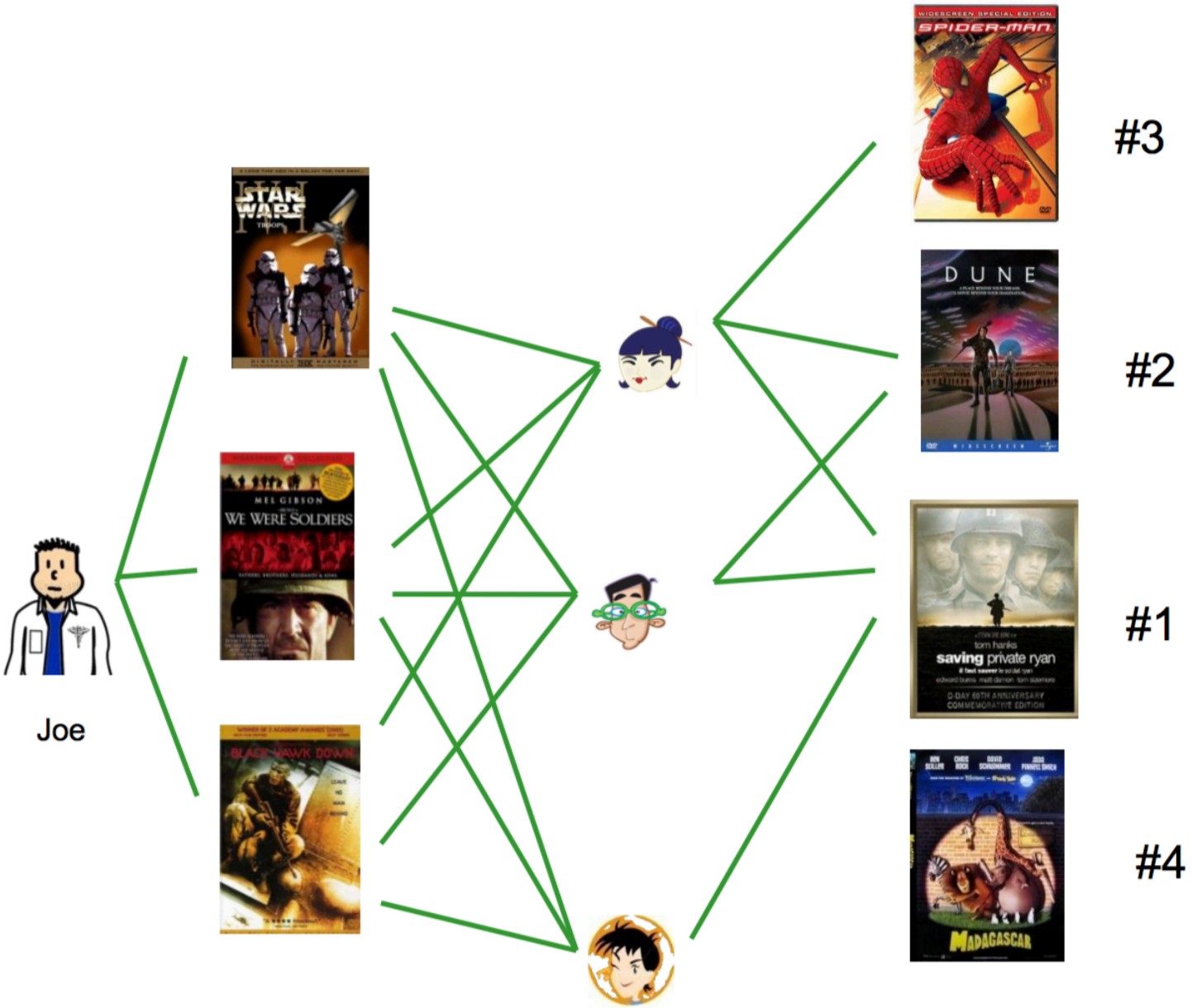






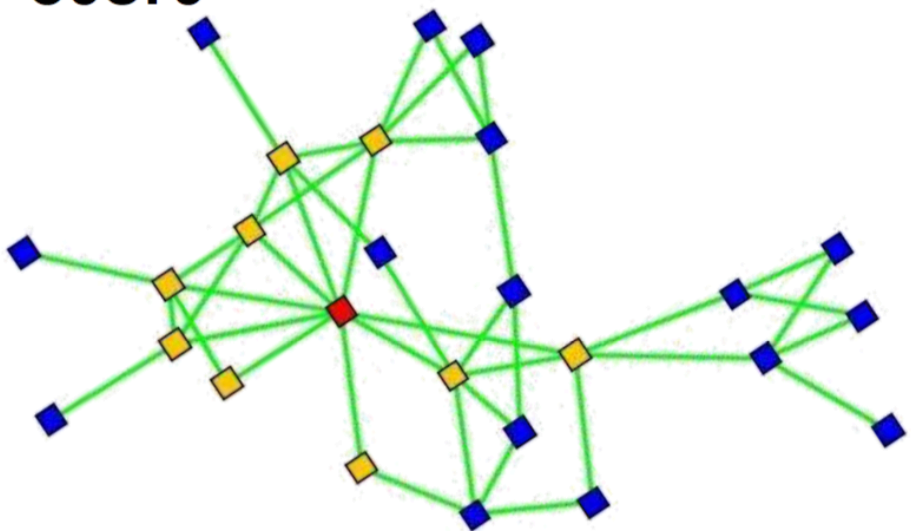

  - Consistent (in large sample size limit this will converge to minimizer)

Basic Idea

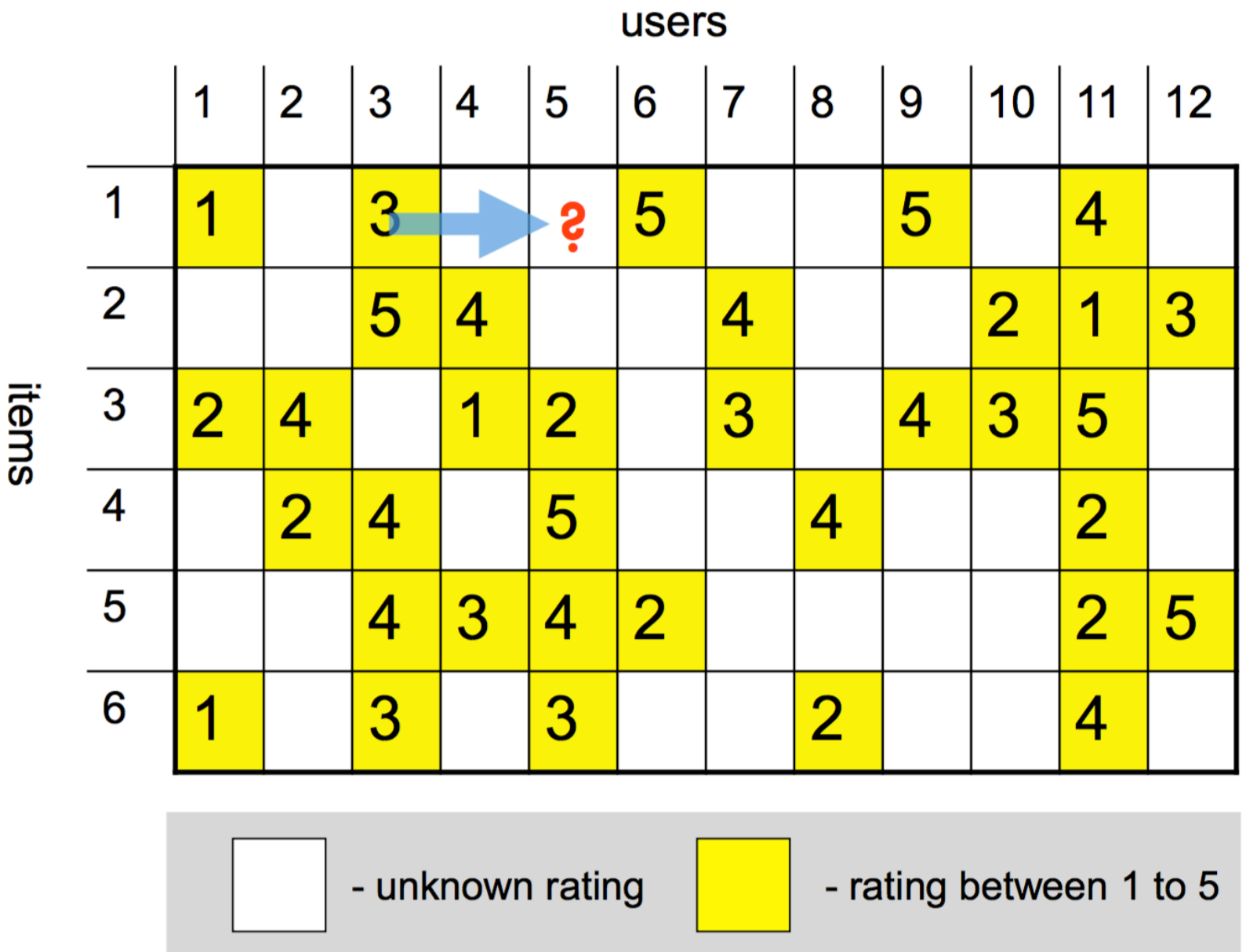


Basic Idea

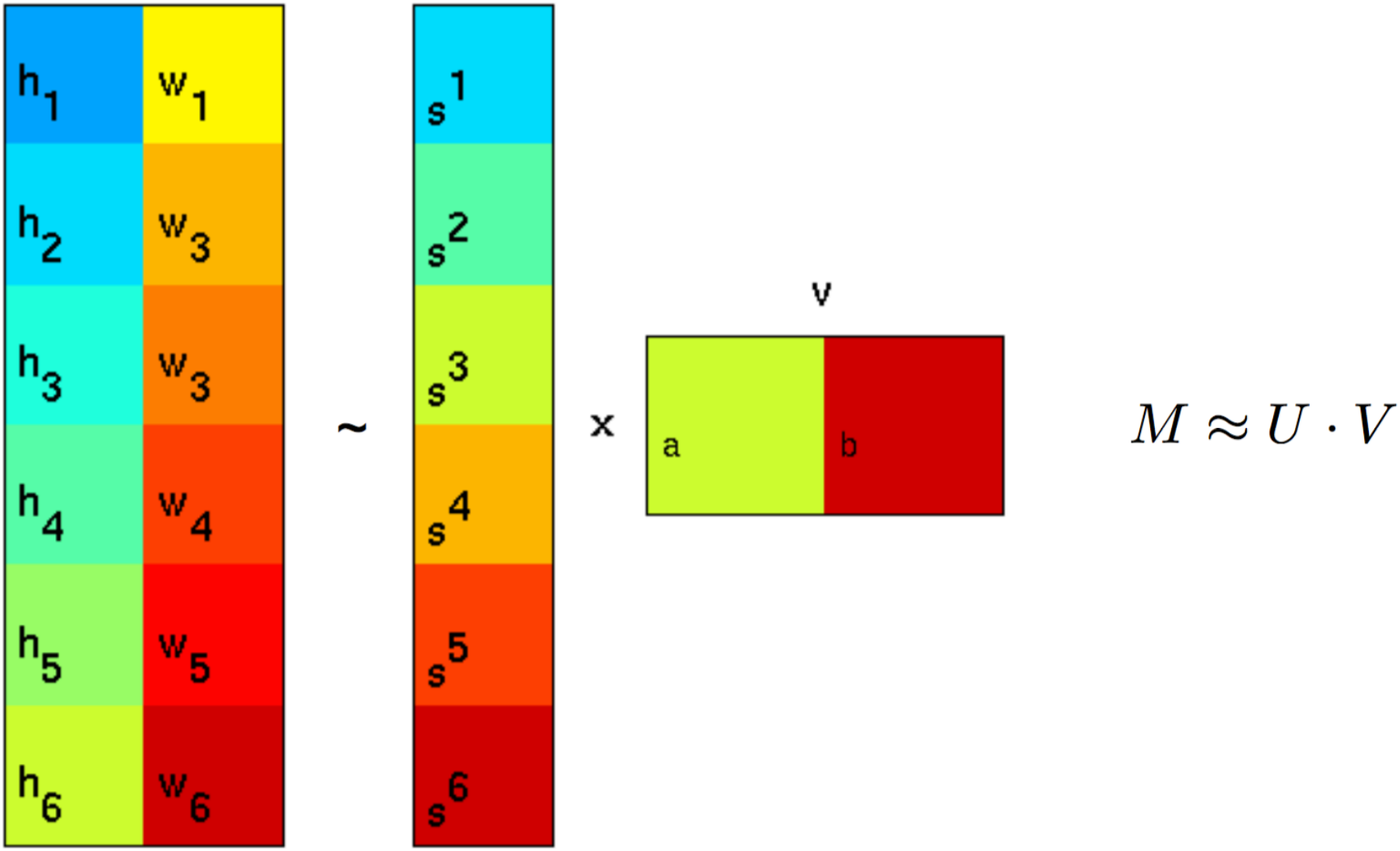
- (user,user) similarity to recommend items
  - good if item base is smaller than user base
  - good if item base changes rapidly
  - traverse bipartite similarity graph
- (item,item) similarity to recommend new items that were also liked by the same users
  - good if the user base is small is small
- Oldest known CF method



Intuitions

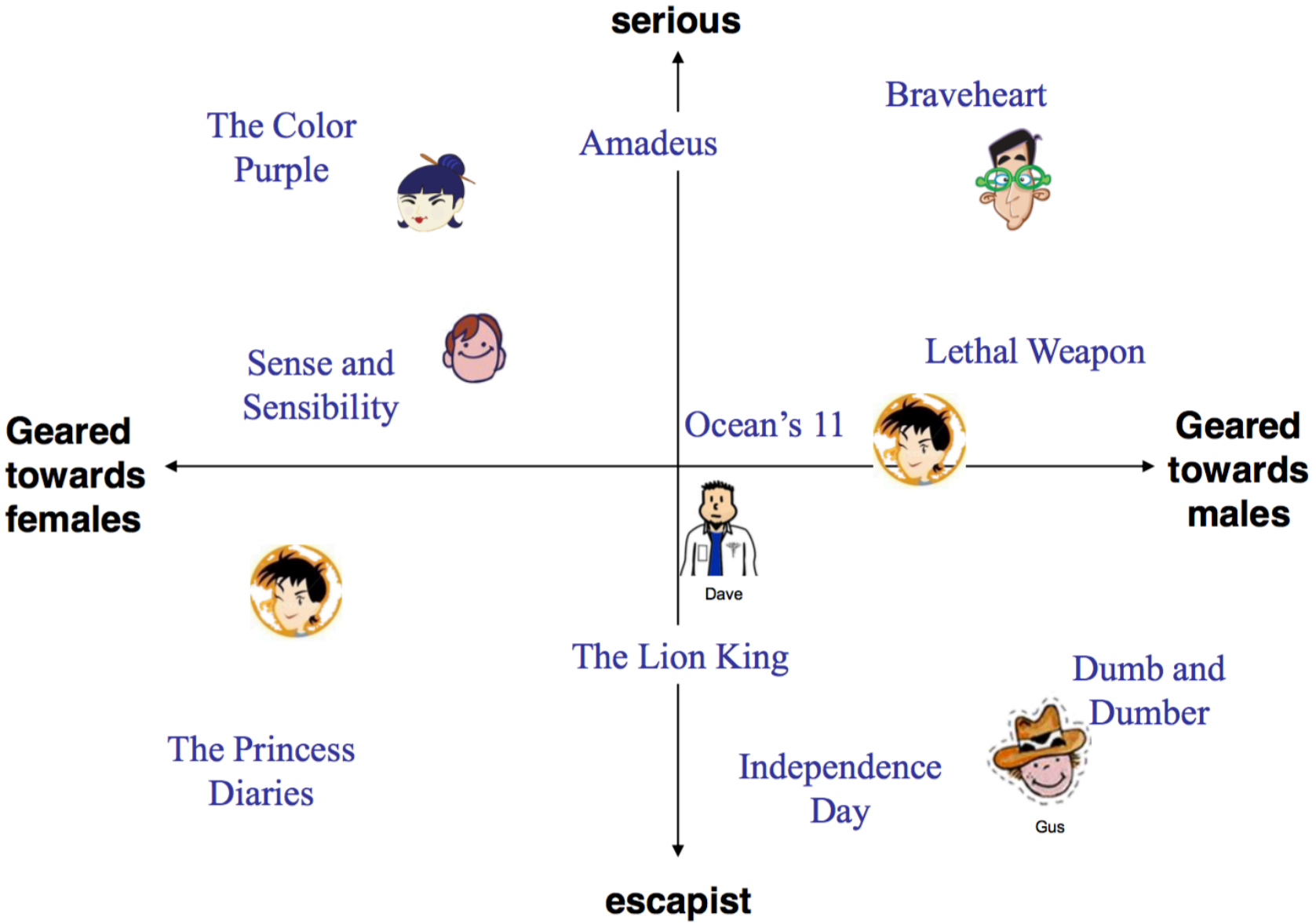

users
items
1 2 3 4 5 6 7 8 9 10 11 12
1 1 3 ? 5 5 4
2 5 4 4 2 1 3
3 2 4 1 2 3 4 3 5
4 2 4 5 4 2
5 4 3 4 2 2 5
6 1 3 3 2 4
- unknown rating
- rating between 1 to 5

Matrix Factorization



$$M \approx U \cdot V$$

Latent Dimension View



Basic Matrix Factorization

Unknown variable can be estimated by the inner product of the corresponding item representation and user representation

users

items

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

~

~

items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

●

users

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

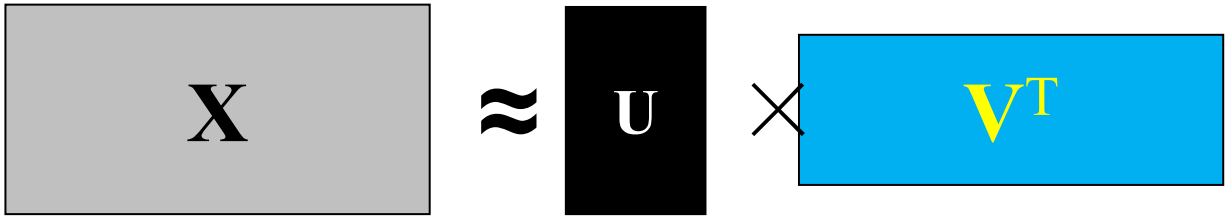
A rank-3 SVD approximation

Formal Definition

Formal Definition

Matrix Factorization is a group of algorithm in linear algebra where a matrix **X** is factorized into (usually) two matrices, **U** and **V**:

$$mf(\mathbf{X}) \rightarrow \mathbf{U}\mathbf{V}^{\mathrm{T}}$$



Measuring the Approximation

$$\mathbf{X}_{M\times N} \approx \mathbf{U}_{M\times K}\mathbf{V}^{\mathrm{T}}_{K\times N}$$

- $\mathcal{L}\left(\mathbf{X}, \mathbf{UV}^{\mathrm{T}}\right)$
  - square loss: $\|\mathbf{X} - \mathbf{UV}^{\mathrm{T}}\|^2$
  - divergence: $\mathrm{D}\left(\mathbf{X}, \mathbf{UV}^{\mathrm{T}}\right) = \sum_{i,j}\left(x_{ij}\log\frac{x_{ij}}{(\mathbf{UV}^{\mathrm{T}})_{ij}} - x_{ij} + \left(\mathbf{UV}^{\mathrm{T}}\right)_{ij}\right)$

Constraints/Regularizations

- Constraints on **U** and **V**
  - Nonnegative: all entries is larger or equals to zero
  - Orthonormal/orthogonal
- Regularizing **U** and **V**
  - ℓ_1-norm/ℓ_0-norm: sparse
  - ℓ_2-norm: robust
  - Trace norm: low rank
- In some cases, regularization is equivalent to constraints
  - Constraint $\|\vec{x}\|_2^2 \leq c$ is equivalent to regularizer $\lambda\|\vec{x}\|_2^2$

Matrix Factorization as Loss Function Minimization

$$\min_{\mathbf{U},\mathbf{V}} \mathcal{L}\left(\mathbf{X}, \mathbf{U}\mathbf{V}^{\mathrm{T}}\right) + \mathcal{R}(\mathbf{U}, \mathbf{V})$$
$$s.t. \mathbf{U} \in \mathcal{C}_{\mathrm{U}}; \mathbf{V} \in \mathcal{C}_{\mathrm{V}}$$

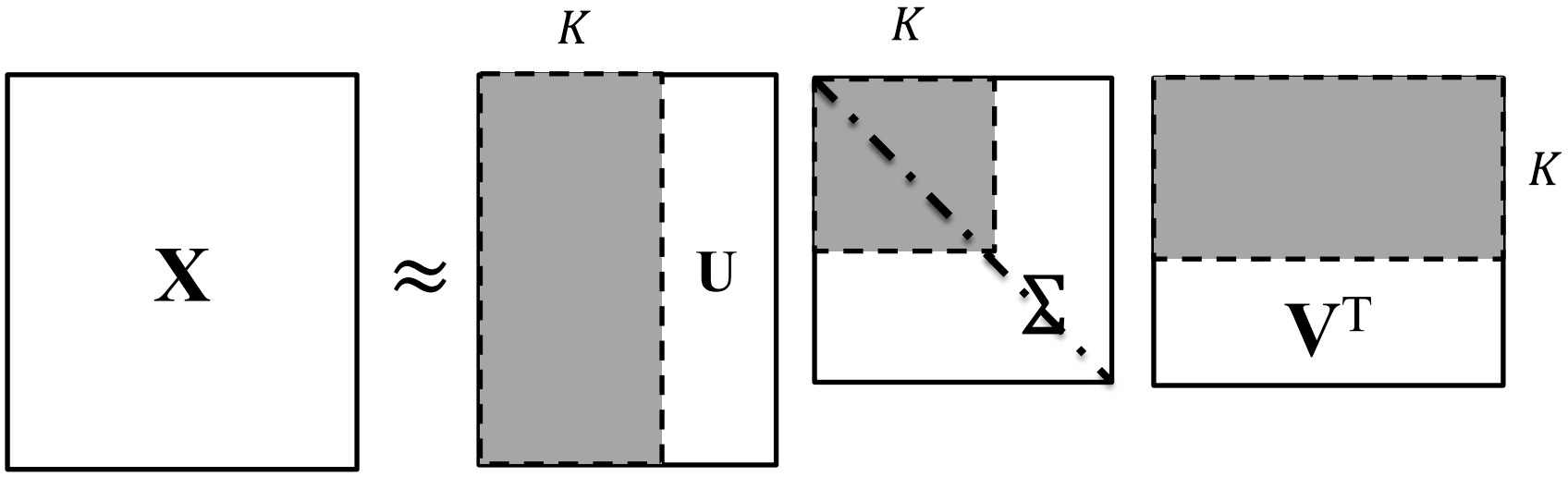
- $\mathcal{L}\left(\mathbf{X}, \mathbf{U}\mathbf{V}^{\mathrm{T}}\right)$: approximation error
  - Square loss $\left\|\mathbf{X} - \mathbf{U}\mathbf{V}^{\mathrm{T}}\right\|_2^2$
  - Divergence $\mathrm{D}\left(\mathbf{X} \parallel \mathbf{U}\mathbf{V}^{\mathrm{T}}\right)$
- $\mathcal{R}(\mathbf{U}, \mathbf{V})$: regularization
  - ℓ_1-norm/ℓ_0-norm: sparse
  - ℓ_2-norm/ℓ_p-norm: robust
  - trace norm: low rank
- $\mathcal{C}_{\mathrm{U}}$ and $\mathcal{C}_{\mathrm{V}}$: the constraints
  - Orthonormal/orthogonal
  - Nonnegative

Models

Models

- SVD probably can be considered as the first matrix factorization model
- Nonnegative Matrix Factorization—Nonnegative version of SVD.

Latent Semantic Analysis



$$\min_{\mathbf{U},\mathbf{\Sigma_K},\mathbf{V}} \mathcal{L}\left(\mathbf{X}, \mathbf{U}\mathbf{\Sigma}_\mathrm{K}\mathbf{V}^\mathrm{T}\right) = \left\|\mathbf{X} - \mathbf{U}\mathbf{\Sigma}_\mathrm{K}\mathbf{V}^\mathrm{T}\right\|_2^2$$

$$s.t.\ \mathbf{U}^\mathrm{T}\mathbf{U} = \mathbf{I}$$

$$\mathbf{V}\mathbf{V}^\mathrm{T} = \mathbf{I}$$

$\mathbf{\Sigma}_\mathrm{K}$ is diagonal

LSA Steps

- Rank-reduced Singular Value Decomposition (SVD)
  - perform SVD on the input matrix
  - all but the k highest singular values are set to 0
  - produces k-dimensional approximation of the original matrix (in least-squares sense)
  - this is the “semantic space”
- Compute similarities between entities in semantic space (usually with cosine)

Application Example

Titles:

c1: *Human* 1
c2: A *survey*
c3: The *EPS*
c4: *System* a
c5: Relation

m1: The gen
m2: The inte
m3: *Graph*
m4: *Graph*

Terms

human
interface
computer
user
system
response
time
EPS
survey
trees
graph
minors

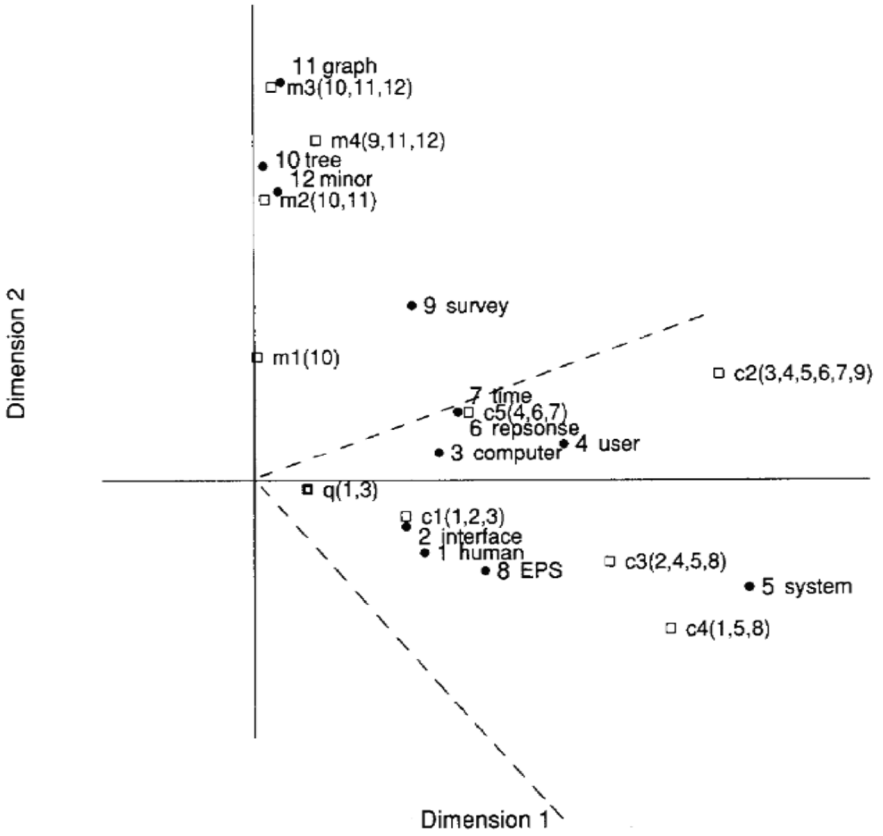


FIG. 1. A two-dimensional plot of 12 Terms and 9 Documents from the sampe TM set. Terms are represented by filled circles. Documents are shown as open squares, and component terms are indicated parenthetically. The query ("human computer interaction") is represented as a pseudo-document at point q. Axes are scaled for Document-Document or Term-Term comparisons. The dotted cone represents the region whose points are within a cosine of .9 from the query q. All documents about human-computer (c1–c5) are "near" the query (i.e., within this cone), but none of the graph theory documents [illegible] In this reduced space, even documents c3 and c5 which share no terms with the query are near it.

A sample dataset consisting of th
italicized. There are two classes
graphs (m1-m4). This dataset ca
indicates the frequency with whic

Nonnegative Matrix Factorization

- Given a nonnegative matrix **X**, find nonnegative matrix factors **U** and **V** such that:

$$\mathbf{X}_{M\times N} \approx \mathbf{U}_{M\times K}\mathbf{V}^{T}{}_{K\times N}$$

- NMF is distinguished from other methods by its nonnegative constraint

Loss Functions for NMF

- Measurements of the approximation
  - Square loss: $\|\mathbf{X} - \mathbf{U}\mathbf{V}^{\mathrm{T}}\|_2^2$
  - Divergence: $\mathrm{D}(\mathbf{X} \parallel \mathbf{Y}) = \sum_{i,j}\left(x_{ij}\log\frac{x_{ij}}{y_{ij}} - x_{ij} + y_{ij}\right)$

    Note $\mathrm{D}(\mathbf{X} \parallel \mathbf{Y})$ is not a distance

- Bad news: not convex!!!
- Good news: convex in **U** only or **V** only
- Alternative optimization
  - Fixing **V** updating **U**; then fixing **U** updating **V**; and then iterate
  - Finding local minima

Algorithm 1

$$\min_{\mathbf{U},\mathbf{V}} \left\|\mathbf{X} - \mathbf{U}\mathbf{V}^{\mathrm{T}}\right\|_2^2$$
$$s.t.\ u_{ij} \geq 0, v_{ij} \geq 0$$

Multiplicative update

1) Initialization: $\mathbf{V} \leftarrow$ random matrix
2) Update $\mathbf{U}$: $u_{ia} \leftarrow u_{ia} \frac{(\mathbf{X}\mathbf{V})_{ia}}{(\mathbf{U}\mathbf{V}^{\mathrm{T}}\mathbf{V})_{ia}}$
3) Update $\mathbf{V}$: $v_{a\mu} \leftarrow v_{a\mu} \frac{(\mathbf{U}^{\mathrm{T}}\mathbf{X})_{a\mu}}{(\mathbf{U}^{\mathrm{T}}\mathbf{U}\mathbf{V}^{\mathrm{T}})_{a\mu}}$
4) Go to 2) until converge

Algorithm 2

$$\min_{\mathbf{U},\mathbf{V}} \mathrm{D}\left(\mathbf{X} \parallel \mathbf{U}\mathbf{V}^{\mathrm{T}}\right)$$

$$s.t.\ u_{ij} \geq 0, v_{ij} \geq 0$$

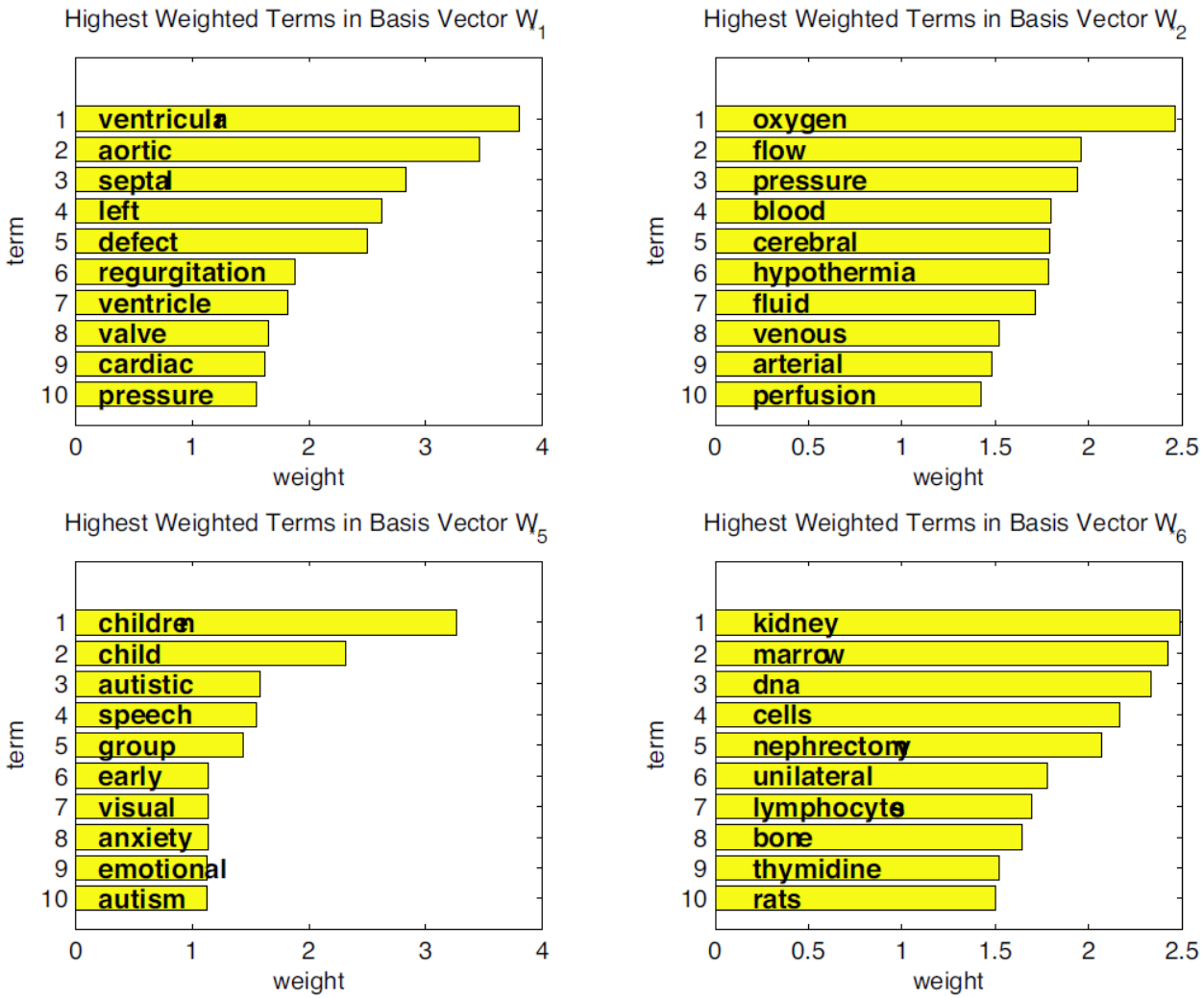
Multiplicative update

1) Initialization: $\mathbf{V} \leftarrow$ random matrix
2) Update $\mathbf{U}$: $u_{ia} \leftarrow u_{ia} \dfrac{\sum_{\mu} \dfrac{x_{i\mu}}{(\mathbf{U}\mathbf{V}^{\mathrm{T}})_{i\mu}} v_{\mu a}}{\sum_{\xi} v_{\xi a}}$
3) Update $\mathbf{V}$: $v_{a\mu} \leftarrow v_{a\mu} \dfrac{\sum_{i} u_{ia} \dfrac{x_{i\mu}}{(\mathbf{U}\mathbf{V}^{\mathrm{T}})_{i\mu}}}{\sum_{j} u_{ja}}$
4) Go to 2) until converge

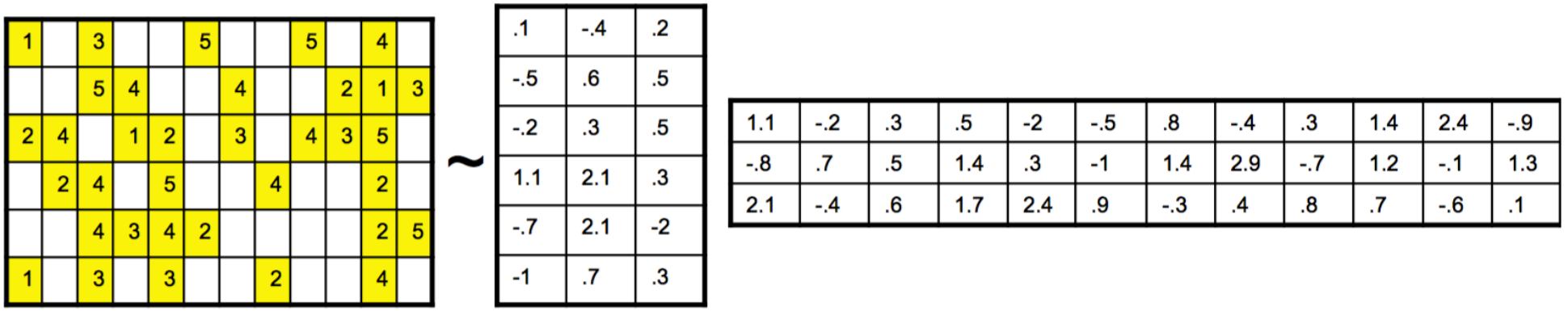
Why Nonnegative?

- Data: many data features are nonnegative
  - Text: tf-idf of words
  - Signal: energy measurements
  - Image: color histogram
- From the viewpoint of human
  - Easy to understand: how to interpret -0.33?
  - Natural
- But, from the viewpoint of algorithm
  - Rude constraints!
  - Not optimal solution

Application: NMF on MED Text Data



Properties of SVD



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

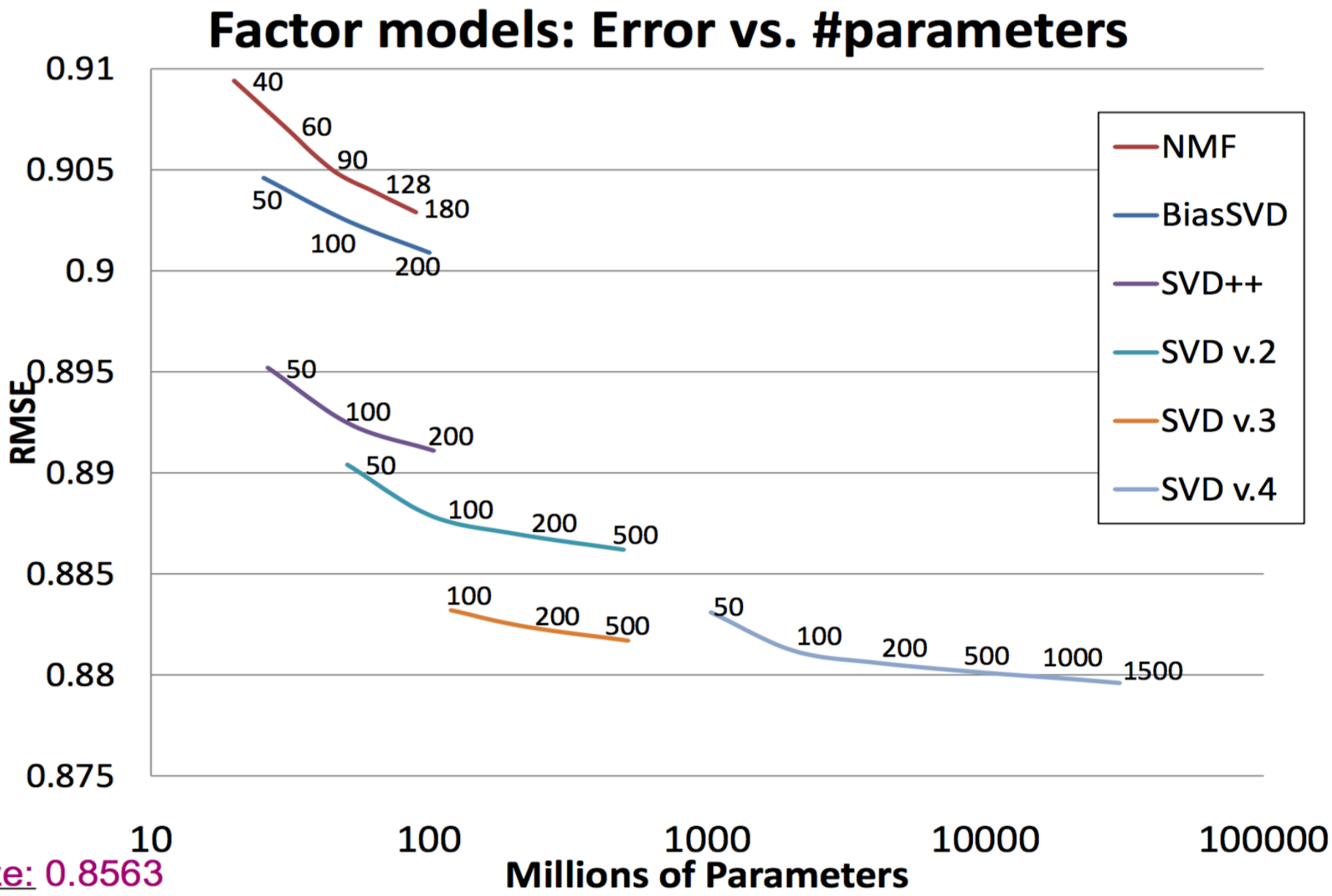
~

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

- SVD is undefined for missing entries
  - stochastic gradient descent (faster)
  - alternating optimization
- Overfitting without regularization
  particularly if fewer reviews than dimensions
- Very popular on Netflix

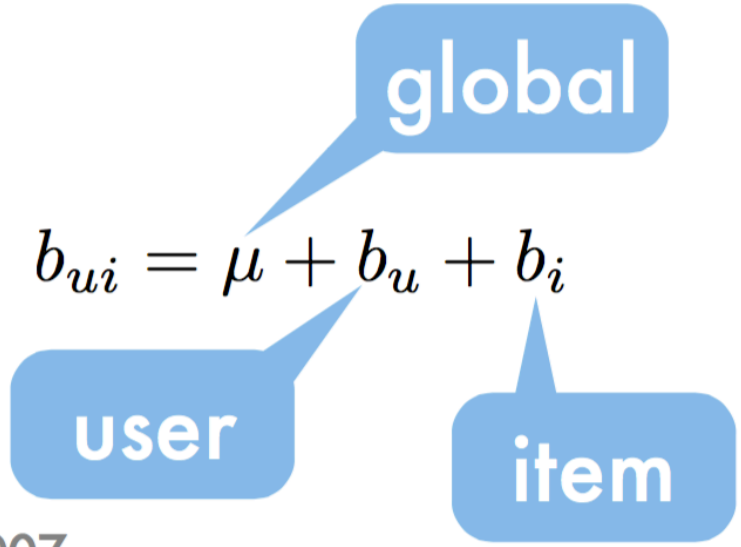
Netflix: 0.9514



Prize: 0.8563

Improving by Adding Normalization/Bias

- Problem
  - Some items are significantly higher rated
  - Some users rate substantially lower
  - Ratings change over time
- Bias correction is crucial for nearest neighborhood recommender algorithm
  - Offset per user
  - Offset per movie
  - Time effects
  - Global bias



Bell & Koren ICDM 2007

http://public.research.att.com/~volinsky/netflix/BellKorICDM07.pdf

Adding Normalization/Bias

- Mean rating is 3.7
- Troll Hunter is 0.7 above mean
- User rates 0.2 below mean
- Baseline is 4.2 stars
- Least mean squares problem



$$\underset{b}{\text{minimize}} \sum_{(u,i)} (r_{ui} - \mu - b_u - b_i)^2 + \lambda \left[\sum_u b_u^2 + \sum_i b_i^2 \right]$$

- Jointly convex. Alternatively remove mean & iterate

$$b_i = \frac{\sum_{u \in R(i)} (r_{ui} - \mu - b_u)}{\lambda + |R(i)|} \text{ and } b_u = \frac{\sum_{i \in R(u)} (r_{ui} - \mu - b_i)}{\lambda + |R(u)|}$$

Modeling with Bias

- **Objective Function**

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

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

- **Stochastic gradient descent**

$$p_u \leftarrow (1 - \lambda\eta_t)p_u - \eta_t q_i \rho_{ui}$$

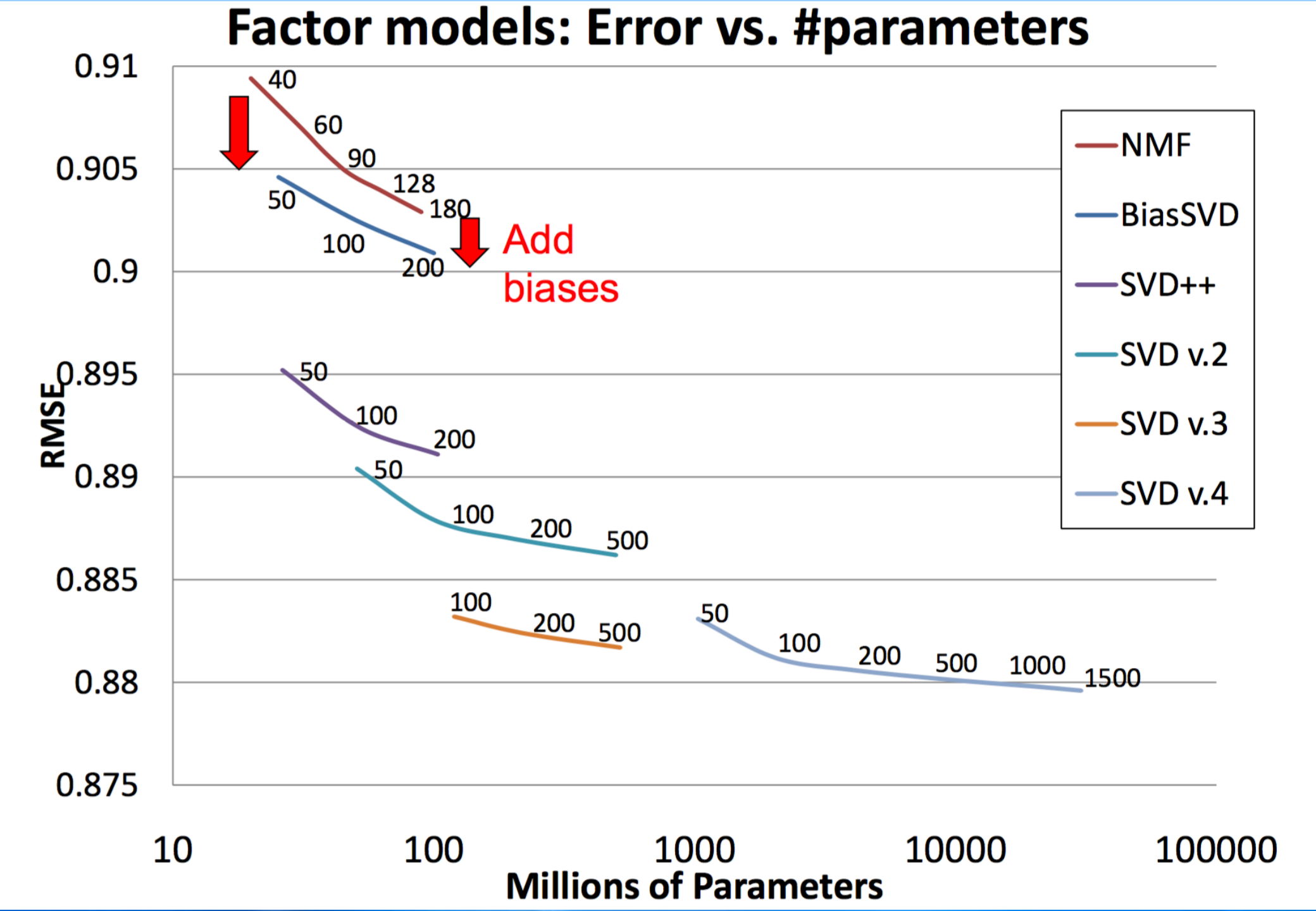
$$q_i \leftarrow (1 - \lambda\eta_t)q_i - \eta_t p_u \rho_{ui}$$

$$b_u \leftarrow (1 - \lambda\eta_t)b_u - \eta_t \rho_{ui}$$

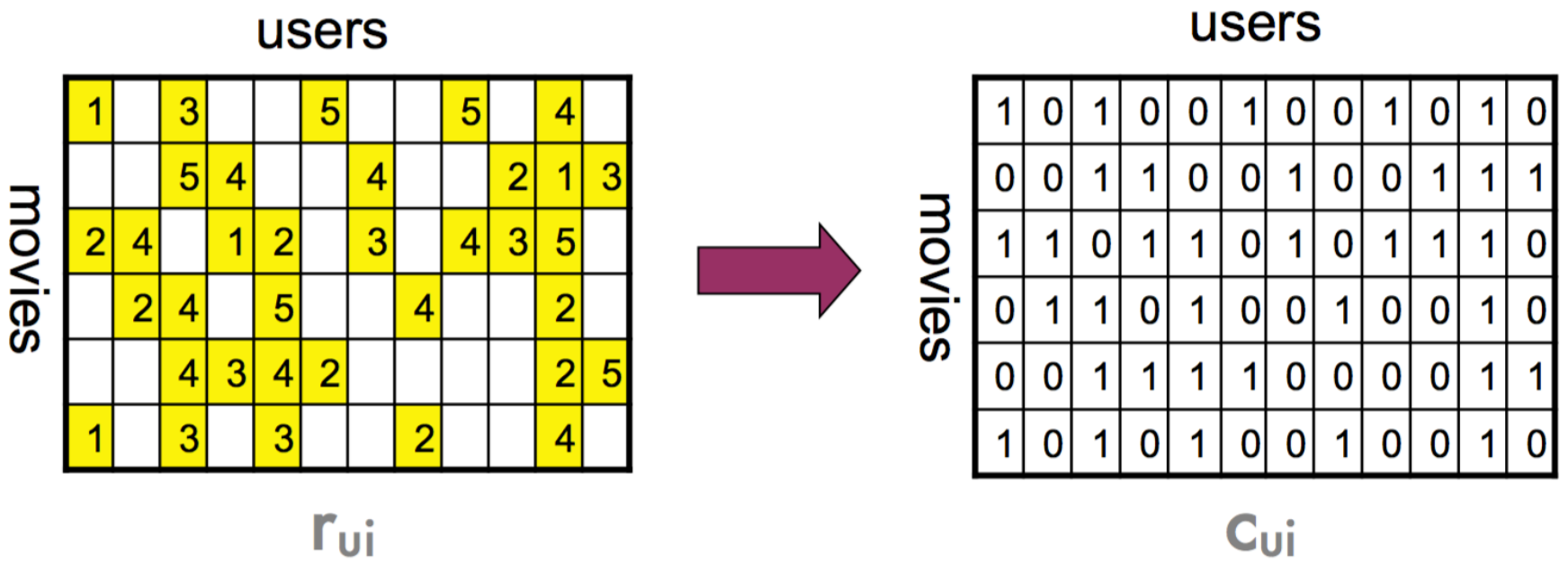
$$b_i \leftarrow (1 - \lambda\eta_t)b_i - \eta_t \rho_{ui}$$

$$\mu \leftarrow (1 - \lambda\eta_t)\mu - \eta_t \rho_{ui}$$

$$\text{where } \rho_{ui} = (r_{ui} - (\mu + b_i + b_u + \langle p_u, q_i \rangle))$$


Factor models: Error vs. #parameters
0.91
0.905
0.9
0.895
0.89
0.885
0.88
0.875
RMSE
40
60
90
128
180
50
100
200
Add
biases
50
100
200
50
100
200
500
100
200
500
50
100
200
500
1000
1500
NMF
BiasSVD
SVD++
SVD v.2
SVD v.3
SVD v.4
10
100
1000
10000
100000
Millions of Parameters

Adding Binary Matrix



- Characterize users by **which** movies they rated
  Edge attributes (observed, rating)
- Adding features to recommender system

$$r_{ui} = \mu + b_u + b_i + \langle p_u, q_i \rangle + \langle c_u, x_i \rangle$$

regression

Alternative Integration

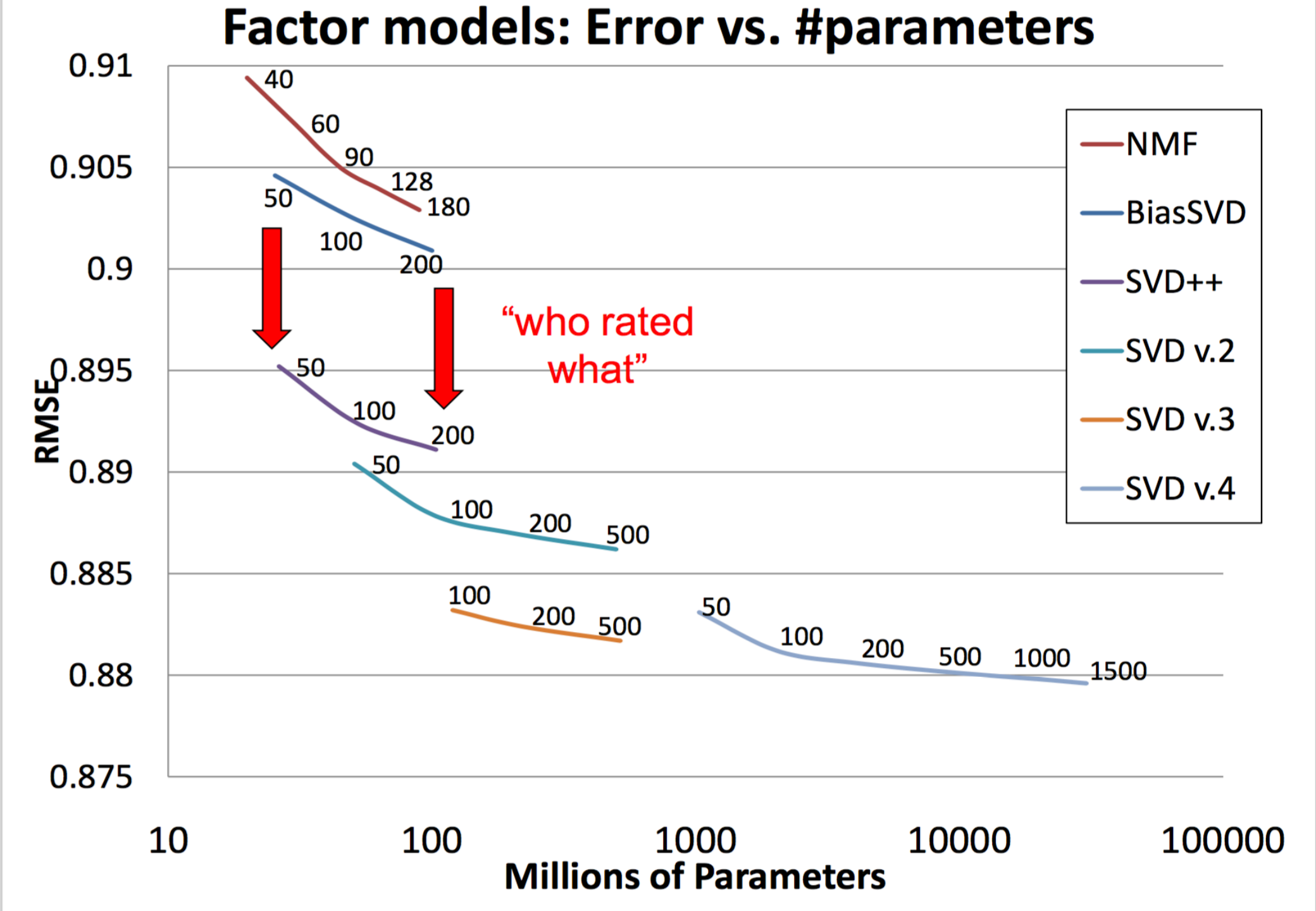
- Key idea - use related ratings to average
- Salakhudtinov & Mnih, 2007

$$q_i \leftarrow q_i + \sum_u c_{ui} p_u$$

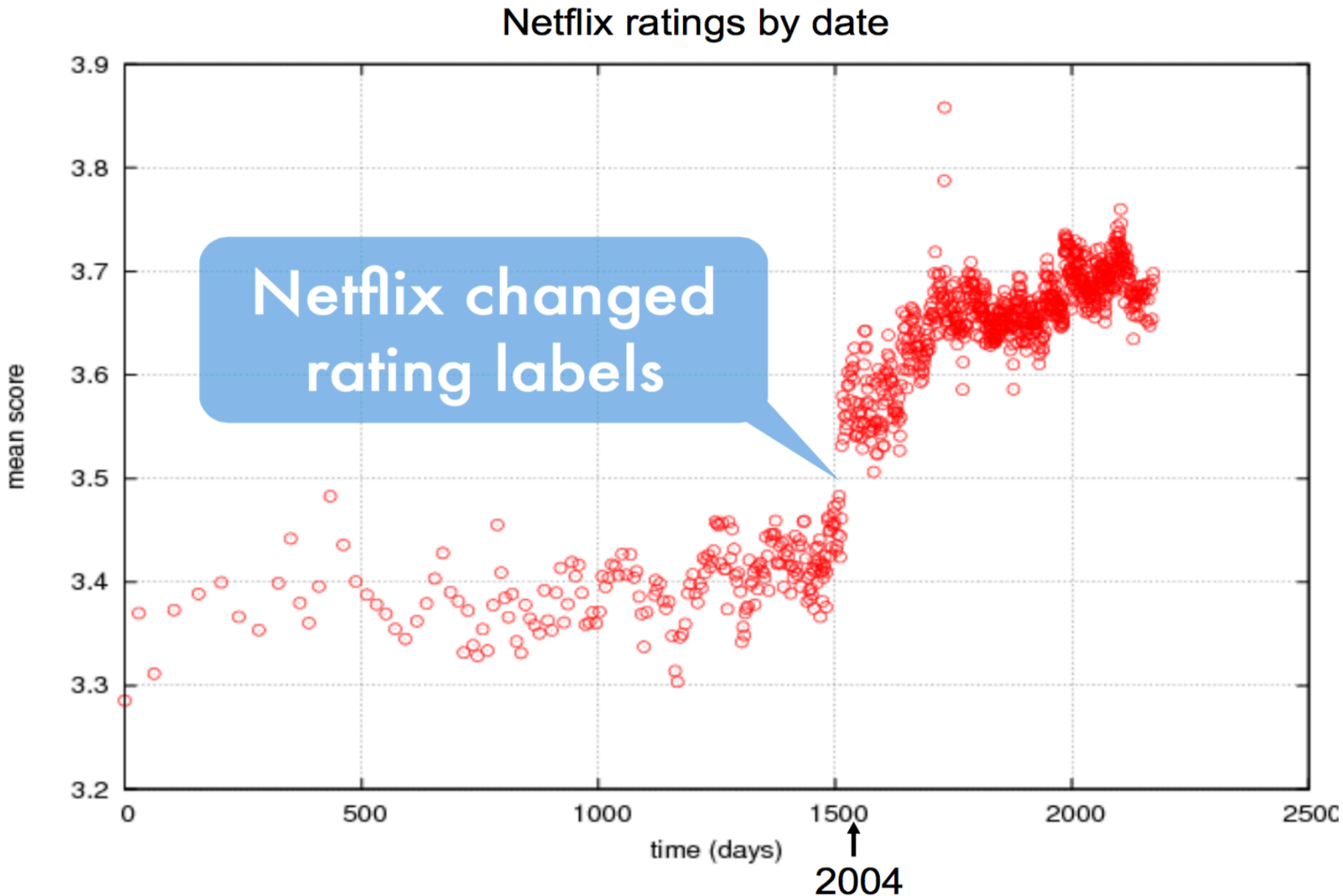
- Koren et al., 2008

$$q_i \leftarrow q_i + \sum_u c_{ui} x_j$$

Overparametrize items by q and x


Factor models: Error vs. #parameters
0.91
0.905
0.9
0.895
0.89
0.885
0.88
0.875
RMSE
40
60
90
128
180
50
100
200
50
100
200
50
100
200
500
100
200
500
50
100
200
500
1000
1500
"who rated what"
NMF
BiasSVD
SVD++
SVD v.2
SVD v.3
SVD v.4
10
100
1000
10000
100000
Millions of Parameters

Temporal Dynamics



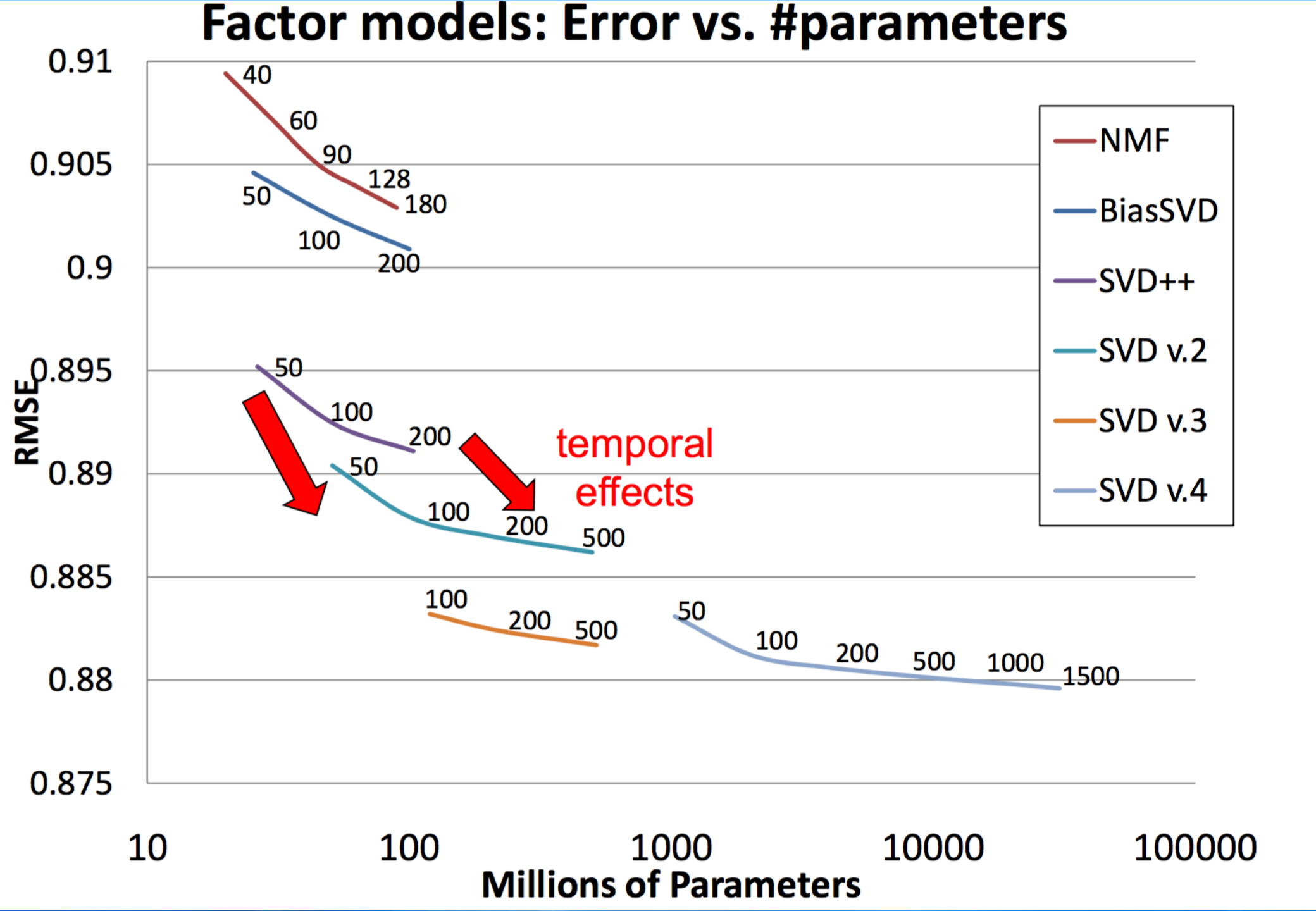
Modeling Temporal Dynamics

- Time-dependent bias
- Time-dependent user preferences

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

- Parameterize functions b and p
  - Slow changes for items
  - Fast sudden changes for users
  - Good parametrization is key

Koren et al., KDD 2009 (CF with temporal dynamics)


Factor models: Error vs. #parameters
0.91
0.905
0.9
0.895
0.89
0.885
0.88
0.875
RMSE
40
60
90
128
180
50
100
200
50
100
200
50
100
200
500
100
200
500
50
100
200
500
1000
1500
temporal
effects
NMF
BiasSVD
SVD++
SVD v.2
SVD v.3
SVD v.4
10
100
1000
10000
100000
Millions of Parameters

Question

- Most values of the user-item rating matrix are unknown, but traditional SVD-solving algorithms need a complete matrix. Three of the following answers are reasonable ways to fill the matrix. Which is not a good way to satisfy this requirement in a situation where the user ratings are on a 1 to 5 scale?

A. Use the user's mean rating.

B. Set all the missing values to 5, the maximum value.

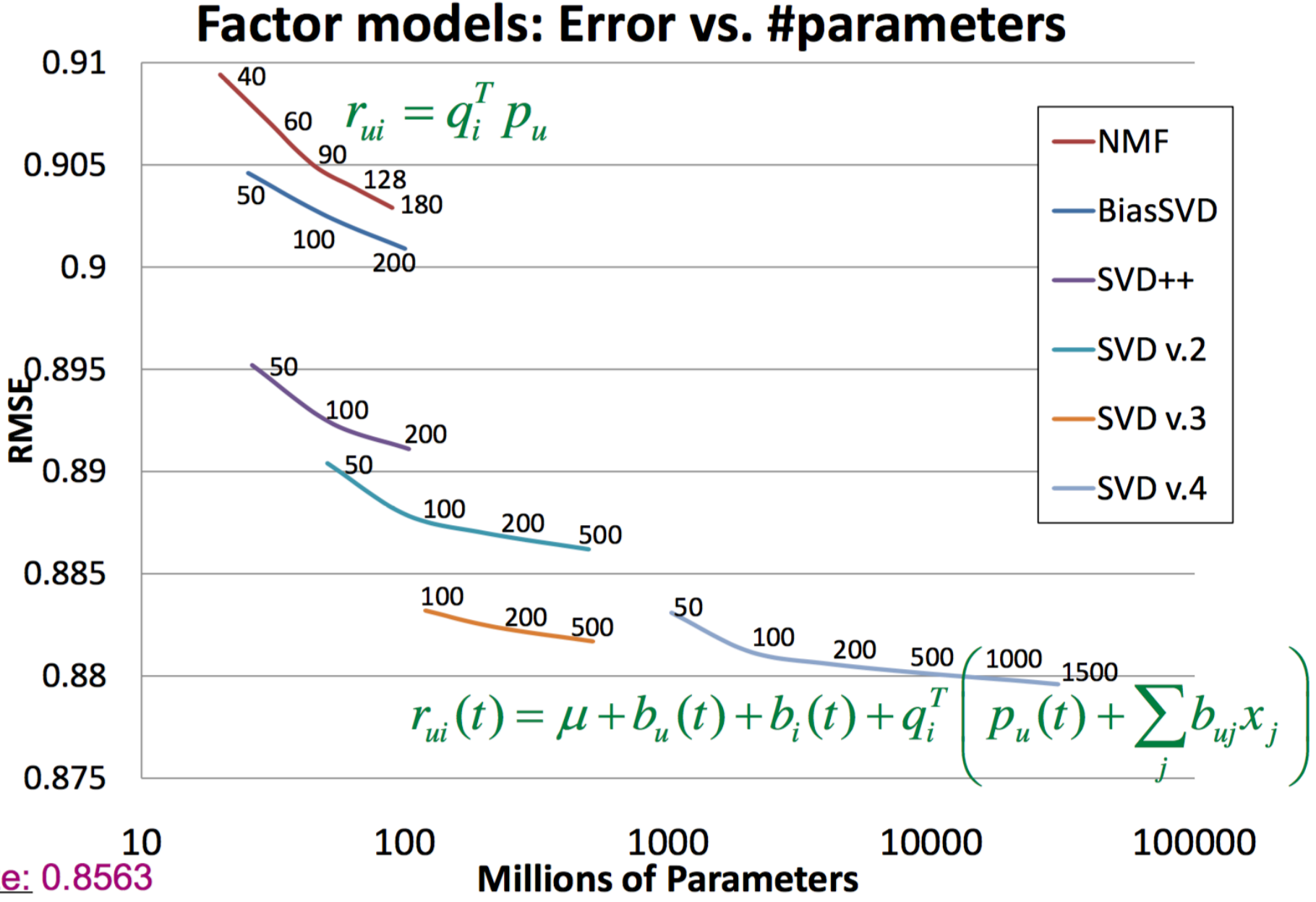
C. Use the item's mean rating.

D. Normalize the ratings and set missing values to 0.

More ideas

- Explain factorizations
- Cold start (new users)
- Different regularization for different parameter groups / different users
- Sharing of statistical strength between users
- Hierarchical matrix co-clustering / factorization (write a paper on that)

Netflix: 0.9514



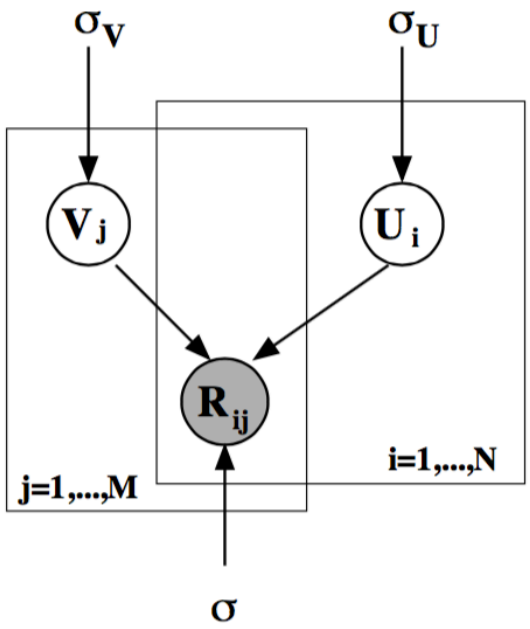
Prize: 0.8563

Probabilistic Interpretation

—Bayesian Probabilistic Matrix Factorization

Bayesian Probabilistic Matrix Factorization

- Aldous-Hoover factorization
  - normal distribution for user and item attributes
  - rating given by inner product
- Ratings



$$p(R_{ij}|U_i, V_j, \sigma^2) = \mathcal{N}(R_{ij}|U_i^T V_j, \sigma^2)$$

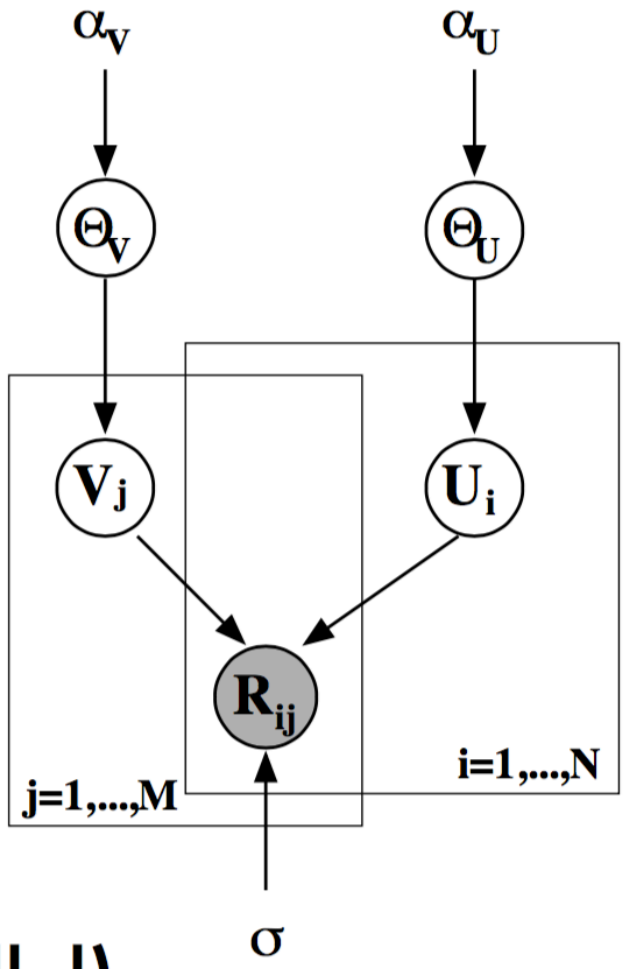
- Latent factors

$$p(U|\sigma_U^2) = \prod_{i=1}^{N} \mathcal{N}(U_i|0, \sigma_U^2 \mathrm{I}), \quad p(V|\sigma_V^2) = \prod_{j=1}^{M} \mathcal{N}(V_j|0, \sigma_V^2 \mathrm{I})$$

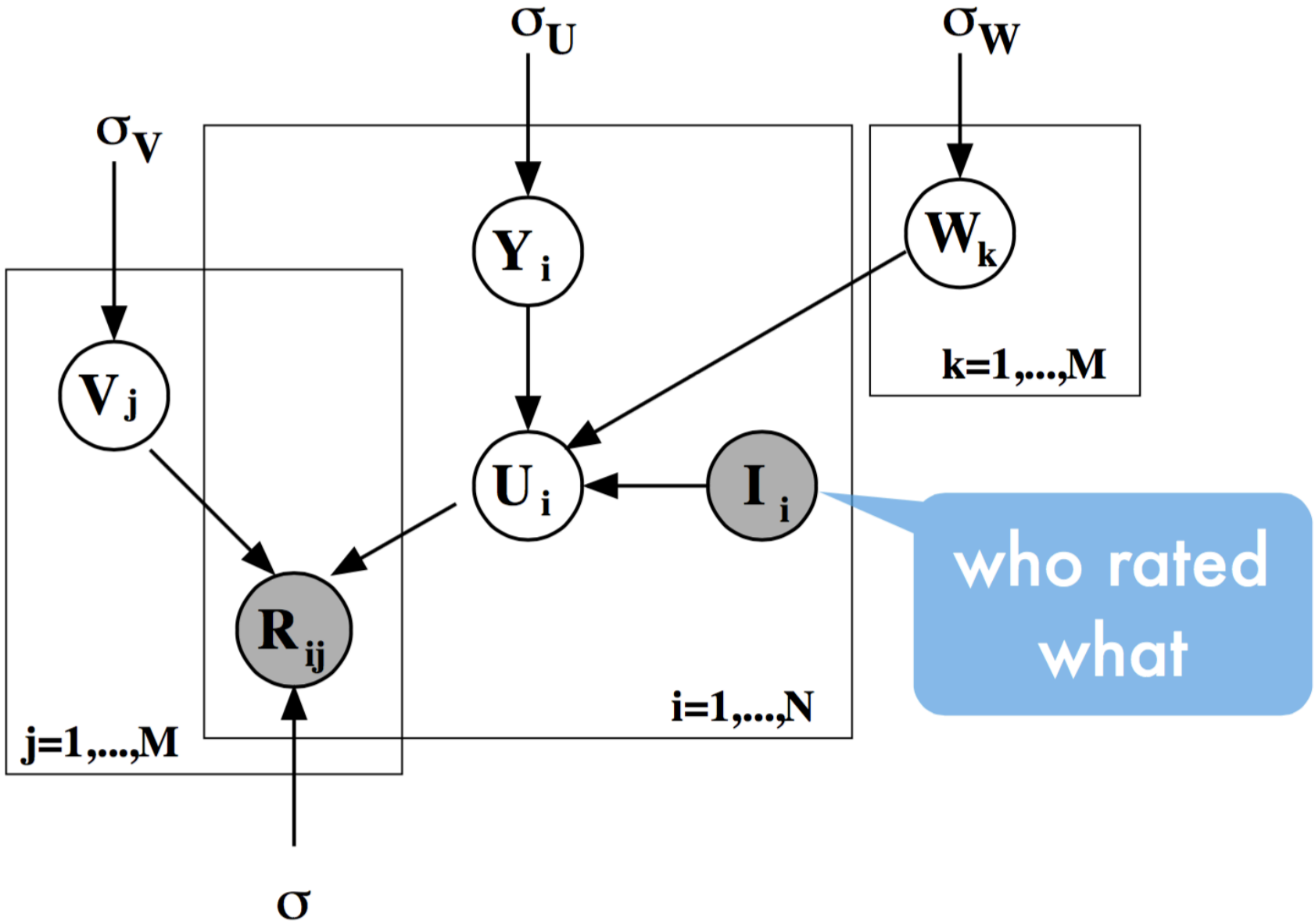
Salakhudtinov & Mnih, ICML 2008 BPMF

More…

- Priors on all factors
- Wishart prior is conjugate to Gaussian, hence use it
- Allows us to adapt the variance automatically
- Inference (Gibbs sampler)
  - Sample user factors (parallel)
  - Sample movie factors (parallel)
  - Sample hyperparameters (parallel)



More…



Sparse Coding

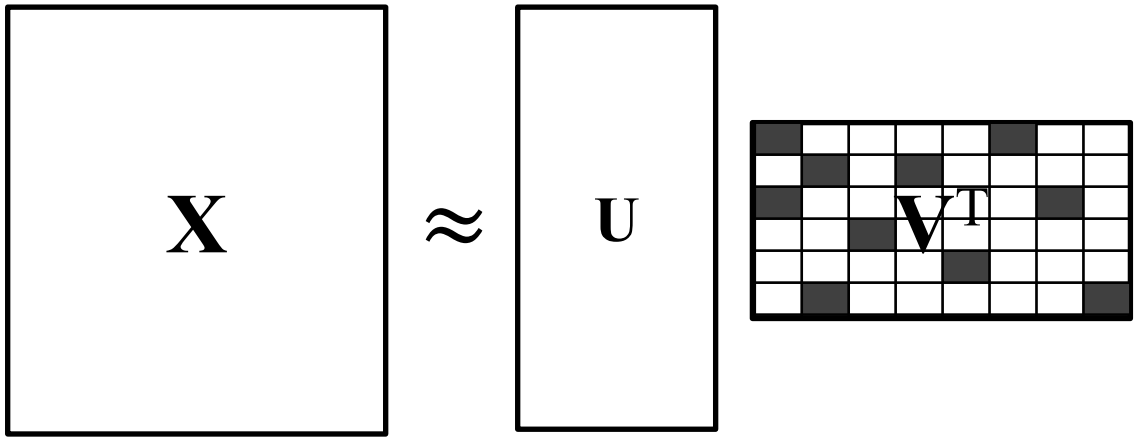
Matrix Factorization as Loss Function Minimization

$$\min_{\mathbf{U},\mathbf{V}} \mathcal{L}\left(\mathbf{X}, \mathbf{U}\mathbf{V}^{\mathrm{T}}\right) + \mathcal{R}(\mathbf{U}, \mathbf{V})$$
$$s.t. \mathbf{U} \in \mathcal{C}_{\mathrm{U}}; \mathbf{V} \in \mathcal{C}_{\mathrm{V}}$$

- $\mathcal{L}\left(\mathbf{X}, \mathbf{U}\mathbf{V}^{\mathrm{T}}\right)$: approximation error
  - Square loss $\left\|\mathbf{X} - \mathbf{U}\mathbf{V}^{\mathrm{T}}\right\|_2^2$
  - Divergence $\mathrm{D}\left(\mathbf{X} \parallel \mathbf{U}\mathbf{V}^{\mathrm{T}}\right)$
- $\mathcal{R}(\mathbf{U}, \mathbf{V})$: regularization
  - ℓ_1-norm/ℓ_0-norm: sparse
  - ℓ_2-norm/ℓ_p-norm: robust
  - trace norm: low rank
- $\mathcal{C}_{\mathrm{U}}$ and $\mathcal{C}_{\mathrm{V}}$: the constraints
  - Orthonormal/orthogonal
  - Nonnegative

Sparse Coding

- Also called dictionary learning



$\mathbf{X}_{M\times N}$ $\mathbf{U}_{M\times K}$ $\mathbf{V}^{\mathrm{T}}{}_{K\times N}$

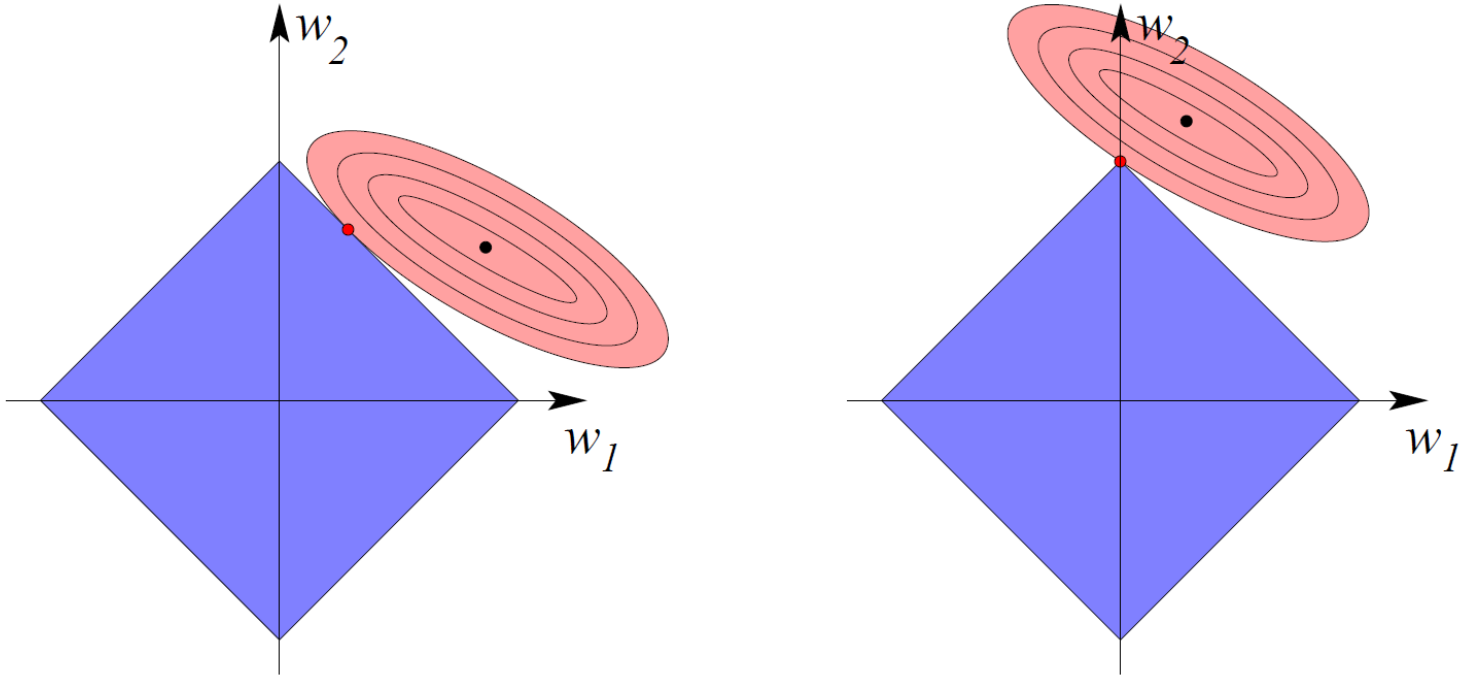
$$\min_{\mathbf{U},\mathbf{V}} \left\|\mathbf{X}-\mathbf{U}\mathbf{V}^{\mathrm{T}}\right\|_2^2 + \lambda_1 \sum_{i,j} u_{ij}^2 + \lambda_2 \sum_{i,j} |v_{ij}|$$

ℓ_1-norm and Sparsity

$$\min_{\mathbf{V}} \left\|\mathbf{X} - \mathbf{U}\mathbf{V}^{\mathrm{T}}\right\|_2^2 + \lambda_2 \sum_{i,j} |v_{ij}|$$

- ℓ_1 relaxes ℓ_0, which counts the number of nonzero items
- Geometric interpretation: penalizing is "equivalent" to constraining

$$\min_{\mathbf{V}} \left\|\mathbf{X} - \mathbf{U}\mathbf{V}^{\mathrm{T}}\right\|_2^2 \quad \text{s.t. } \textstyle\sum_{i,j} |v_{ij}| \leq \mathrm{C}$$



Optimization

- Bad/Good news: not convex/convex in **U** only or **V** only
- Alternative optimization
  - Fixing **V** updating **U**; then fixing **U** updating **V**;
- Fixing **V** updating **U** is not difficult: least square regression
  - $\min_{\mathbf{U}} \left\|\mathbf{X} - \mathbf{U}\mathbf{V}^{\mathrm{T}}\right\|_2^2 + \lambda_1 \sum_{i,j} u_{ij}^2$
- Fixing **U** updating **V** is nontrivial
  - $\min_{\mathbf{V}} \left\|\mathbf{X} - \mathbf{U}\mathbf{V}^{\mathrm{T}}\right\|_2^2 + \lambda_2 \sum_{i,j} \left|v_{ij}\right|$

Optimization with ℓ_1-norm

- Problem decomposition

$$\min_{\mathbf{V}} \left\|\mathbf{X} - \mathbf{U}\mathbf{V}^{\mathrm{T}}\right\|_2^2 + \lambda_2 \sum_{i,j} |v_{ij}| \quad \Leftrightarrow$$

$$\min_{\boldsymbol{v}_i} \|\mathbf{x}_i - \mathbf{U}\boldsymbol{v}_i\|_2^2 + \lambda_2 \textstyle\sum_j |v_{ij}|, \text{ for } i = 1, \cdots, N$$

- We will focus on

$$L(\boldsymbol{v}) = \|\boldsymbol{x} - \mathbf{U}\boldsymbol{v}\|_2^2 + \lambda_2 \sum_j |v_j|$$

- Stochastic coordinate descent

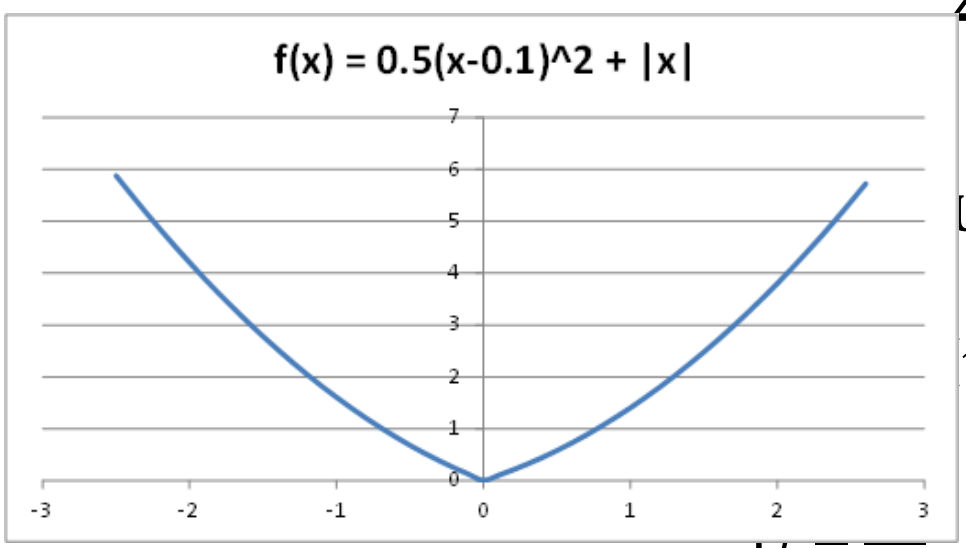


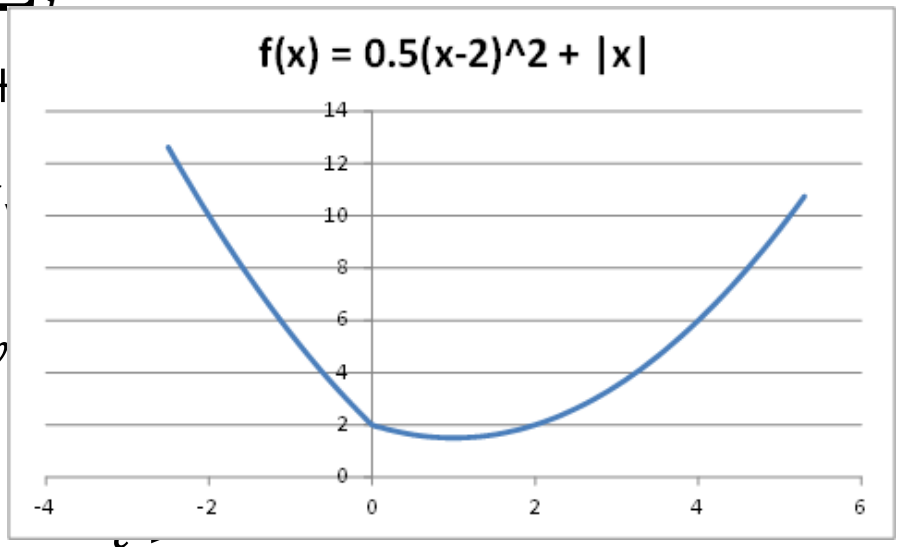
Optimization with ℓ_1-norm

$$L(\boldsymbol{v}) = \|\boldsymbol{x} - \mathbf{U}\boldsymbol{v}\|_2^2 + \lambda_2 \sum_j |v_j|$$

Let's fix all but the k^{th} entries in $\boldsymbol{v}$:

$$l(v_k) = \|\boldsymbol{x} - \mathbf{U}\boldsymbol{v}\|_2^2 + \lambda_2 \sum_j |v_j|$$





$$v = \begin{cases} t - \dfrac{\lambda}{2a} & t \geq -\dfrac{\lambda}{2a} \\ 0 & -\dfrac{\lambda}{2a} \leq t \leq -\dfrac{\lambda}{2a} \\ t + \dfrac{\lambda}{2a} & t \leq -\dfrac{\lambda}{2a} \end{cases}$$

Old Question: Why We Need Sparsity?

- Obvious benefits: saving space and computation time
- The world is sparse?
  - The data actually have sparse structure (what kinds of the data?)
  - The (human) brain evolved to extract sparse structure in data effectively and efficiently. How does the brain do this?
    We want to mimic what goes on in the brain for many real-world applications.

Factorization Machines

[1] Steffen Rendle,“Factorization Machines,” in ICDM’10, Sydney, Australia.

Kaggle dominance

#	Δrank	Team Name ‡ model uploaded * in the money	Score	Entries	Last Submission UTC (Best – Last Submission)
1	—	3 Idiots ‡ * • guestwalk • mandora • yolicat (FM)	0.44464	13	Tue, 23 Sep 2014 23:31:16
2	—	Michael Jahrer and Jeong-Yoon Lee *	0.44527	61	Tue, 23 Sep 2014 23:37:47 (-9.7d)
3	—	beile ‡ *	0.44610	67	Tue, 23 Sep 2014 23:07:36 (-0.8h)

#	Δrank	Team Name * in the money	Score	Entries	Last Submission UTC (Best – Last Submission)
1	—	4 Idiots * • guestwalk • Michael Jahrer • yolicat • mandora (FM)	0.3791384	273	Mon, 09 Feb 2015 19:37:27 (-43.6h)
2	—	Owen *	0.3803652	94	Mon, 09 Feb 2015 02:34:23 (-0.5h)
3	—	Random Walker *	0.3806351	242	Mon, 09 Feb 2015 10:59:10

Factorization Machines (FMs)

- FMs is inspired from previous factorization models
- Each data point has a feature vector x, and a target value (e.g., rating score) .
- FMs model second-order feature interaction to overcome the sparsity.

Factorization Machines (FMs)

- Let $x \in R^p$ be an input vector with p predictor variables.
- Model equation (degree k=2):

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{p} w_i x_i + \sum_{i=1}^{p} \sum_{j>i}^{p} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$$

- Model parameters:

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^p, \quad \mathbf{V} \in \mathbb{R}^{p \times k}$$

Factorization Machines: Discussion

- FMs work with real valued input.
- FMs include variable interactions like polynomial regression.
- Model parameters for interactions are factorized.
- Number of model parameters is O(kp) (instead of O(p^2) for polynomial regression).

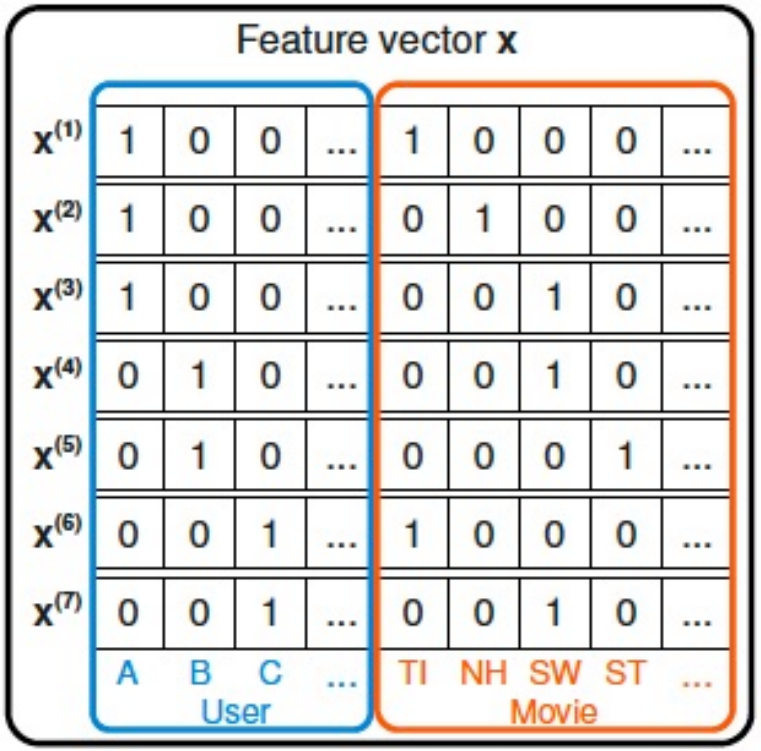
Question:
How are FMs related to matrix factorization?

FMs and MF

- Two categorical variables encoded with real valued predictor variables:

- with this data, the FM is identical to MF with bias:

$$\hat{y}(\mathbf{x}) = w_0 + w_u + w_i + \underbrace{\langle \mathbf{v}_u, \mathbf{v}_i \rangle}_{\text{MF}}$$



Computation Complexity

- Factorization Machine model equation:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{p} w_i x_i + \sum_{i=1}^{p} \sum_{j>i}^{p} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$$

- Trivial computation: $O(p^2 k)$
- Efficient computation can be done in: O(pk)
- Making use of many zeros in x even in: $O(N_z(x)k)$, where $N_z(x)$ is the number of non-zero elements in vector x.

Efficient Computation

- The model equation of an FM can be computed in O(pk).
- Proof:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{p} w_i x_i + \sum_{i=1}^{p} \sum_{j>i}^{p} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$$

$$= w_0 + \sum_{i=1}^{p} w_i x_i + \frac{1}{2} \sum_{f=1}^{k} \left[\left(\sum_{i=1}^{p} x_i v_{i,f} \right)^2 - \sum_{i=1}^{p} (x_i v_{i,f})^2 \right]$$

- In the sums over i , only non-zero x_i elements have to be summed up $\rightarrow$ O($N_z(x)k$)

Learning

- Stochastic gradient descent
- Alternating least squares/ Coordinate Descent

- All the learning algorithms have a runtime of $O(kN_z(x)i)$, where i is the number of iterations and $N_z(x)$ the number of non-zero elements in the design matrix X.

Applications

- FMs are especially suited for ML problems:
  - Categorical variables of large domain
  - Number of predictor variables is large.
  - Interactions between predictor variables are of interest.

- (Context-aware) recommender systems
- Clickthrough prediction
- Link prediction in social networks

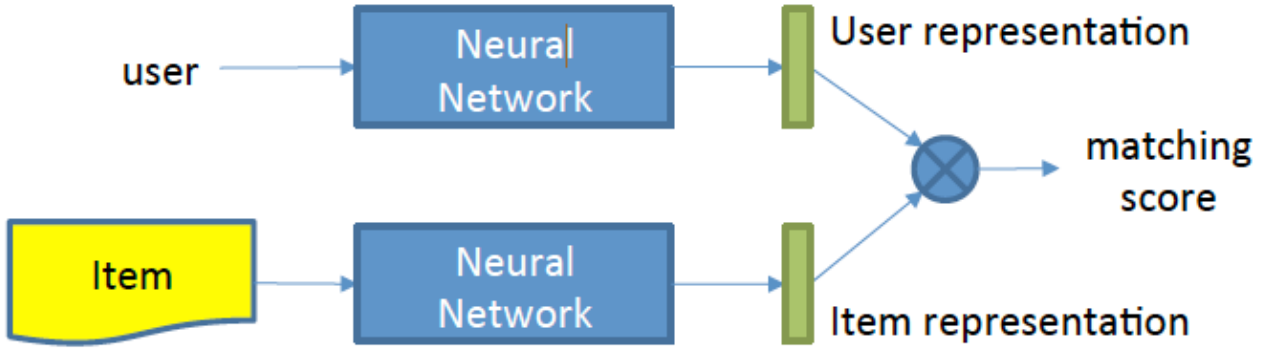
Summary

- Factorization Machines (FM) are a polynomial regression model with factorized interaction parameters.
- FMs bring together the generality of standard machine learning methods with the prediction quality of factorization models.
- FMs can be computed efficiently.

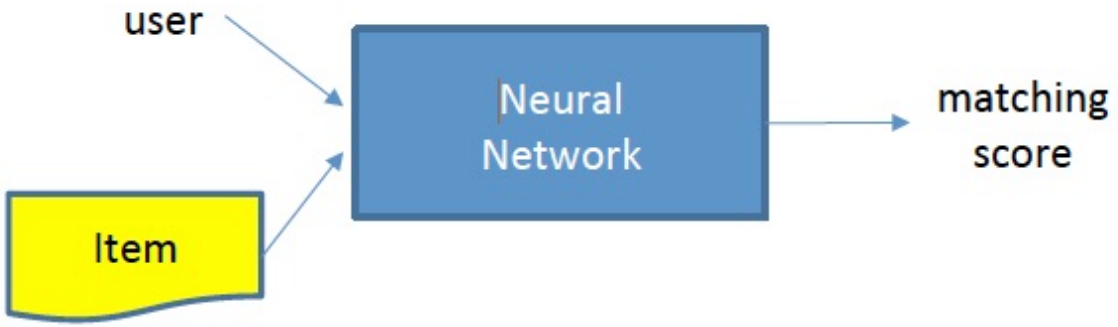
Neural models related to Matrix Factorization in the context of Recommender Systems

Deep Learning Models for CF

- Methods of representation learning

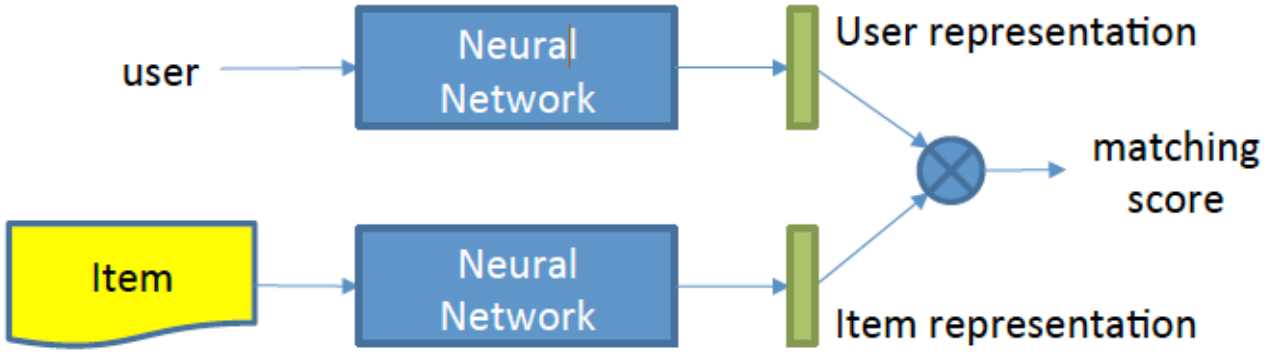


- Methods of matching function learning



Deep Matrix Factorization Models

- Methods of representation learning

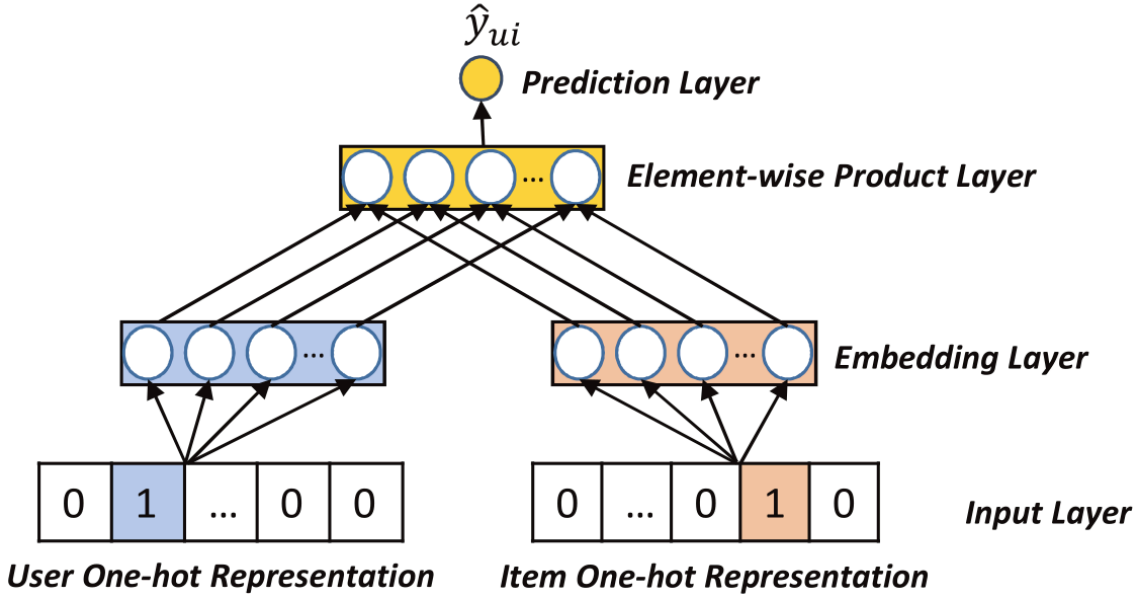


- DeepMF: Deep Matrix Factorization (Xue et al, IJCAI'17)
- MF as a Neural Network (Wang et al, SIGIR'17)

MF as a Neural Network (Wang et al, SIGIR'17)

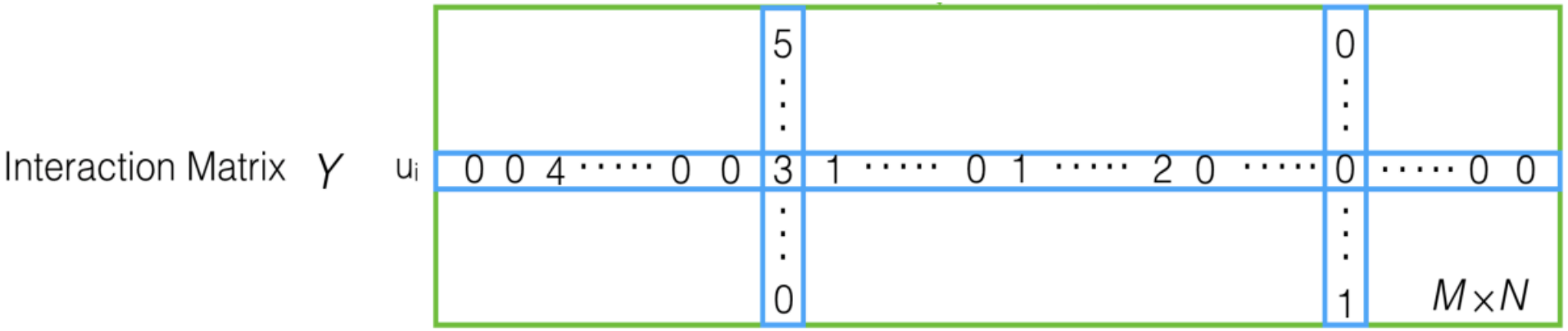
- **Input**: user -> ID (one-hot), item -> ID (one-hot).
- **Representation Function**: linear embedding layer.
- **Matching Function**: inner product.

$$f_{MF}(u, i|\mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u^\top \mathbf{q}_i = \sum_{k=1}^{K} p_{uk} q_{ik}$$



Deep Matrix Factorization (Xue et al, IJCAI'17)

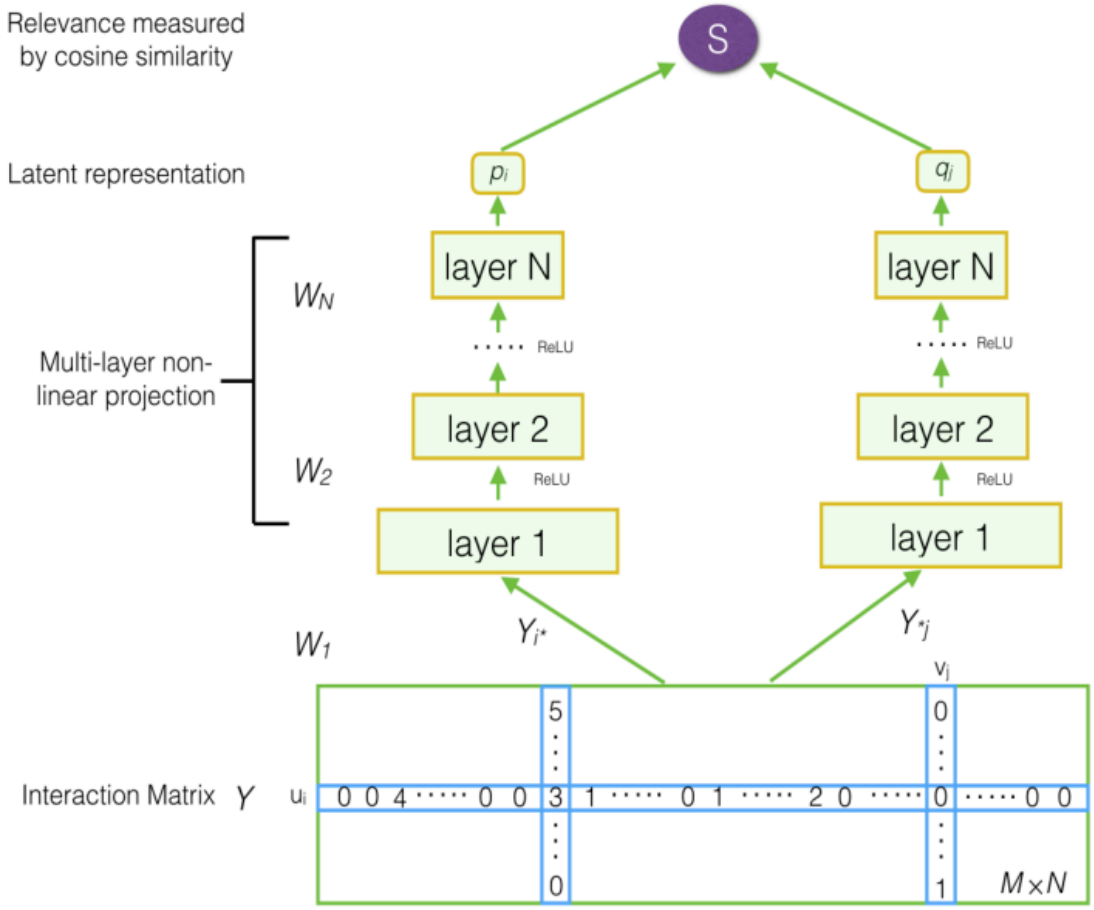
- Input:
  - user → items that she has rated (multi-hot), i.e., row vector of Y indicates the user's global preference
  - item → users who have rated it (multi-hot), i.e., column vector of Y indicates the item's rating profile.



Deep Matrix Factorization (Xue et al, IJCAI'17)

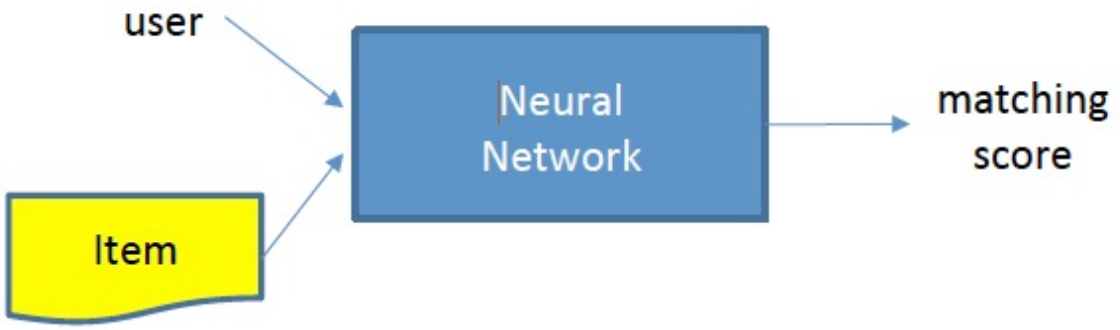
- Input:
  - Multi-Layer Perception
  - Matching Function: cosine similarity

$$cosine(p_i, q_j) = \frac{p_i^T q_j}{\|p_i\| \, \|q_j\|}$$



Next…

- Methods of matching function learning



- NeuMF: Neural Matrix Factorization (He et al, WWW'17)
- NNCF: Neighbor-based NCF (Bai et al, CIKM'17)

Why Using Neural Networks to Learn the Matching Function?

- The simple choice of inner product can limit the expressiveness of an embedding-based matching model.

$$\hat{y}_{ui} = \mathbf{U}_i^T \mathbf{V}_j \simeq cos(\mathbf{U}_i, \mathbf{V}_j)$$

(E.g., assuming a unit length)

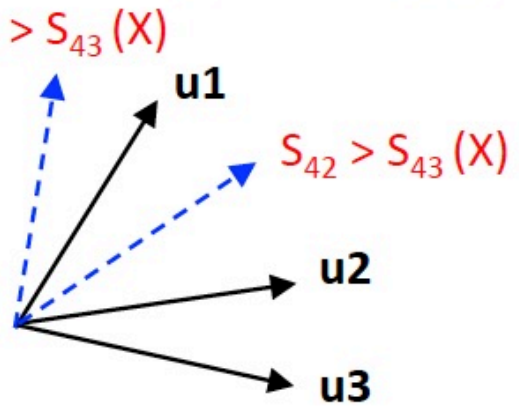
Example:

	i_1	i_2	i_3	i_4	i_5
u_1	1	1	1	0	1
u_2	0	1	1	0	0
u_3	0	1	1	1	0
u_4	1	0	1	1	1

sim(u1, u2) = 0.5

sim(u3, u1) = 0.4
sim(u3, u2) = 0.66

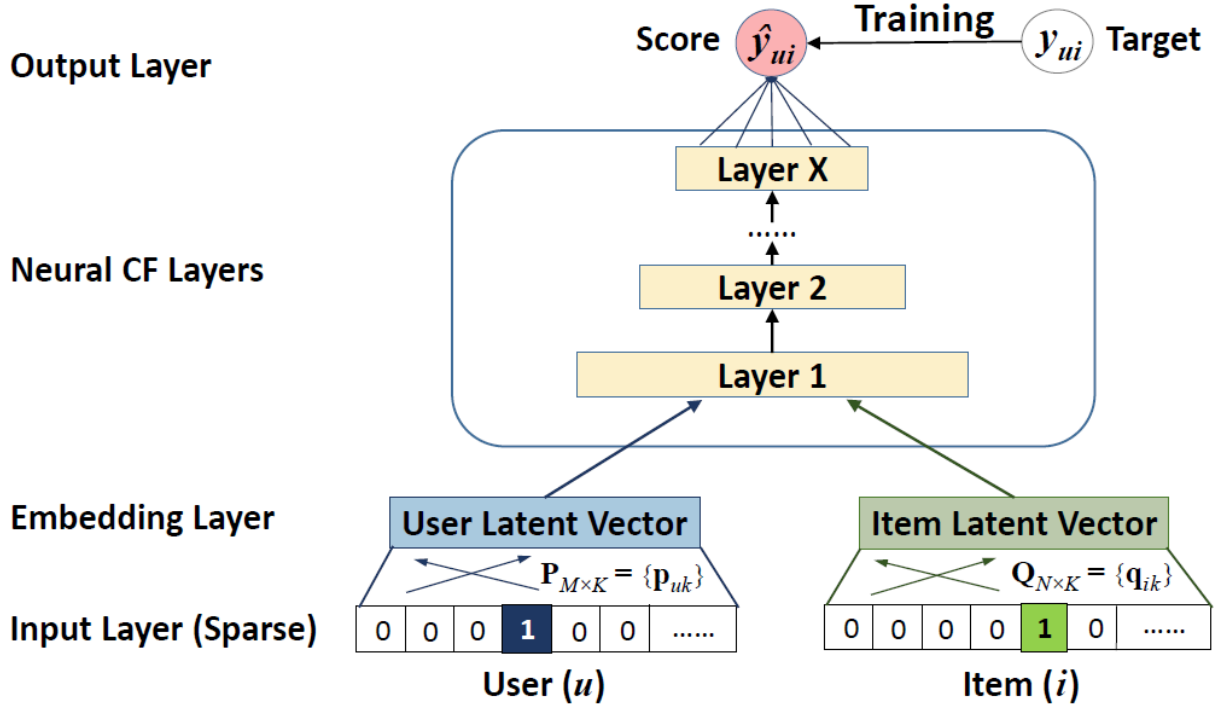
sim(u4, u1) = 0.6 *****
sim(u4, u2) = 0.2 *
sim(u4, u3) = 0.4 ***



Neural Collaborative Filtering Framework (He et al, WWW'17)

- NCF is a general framework that replaces the inner product with a neural network to learn the matching function.

$$\hat{y}_{ui} = f(\mathbf{p}_u, \mathbf{q}_i)$$

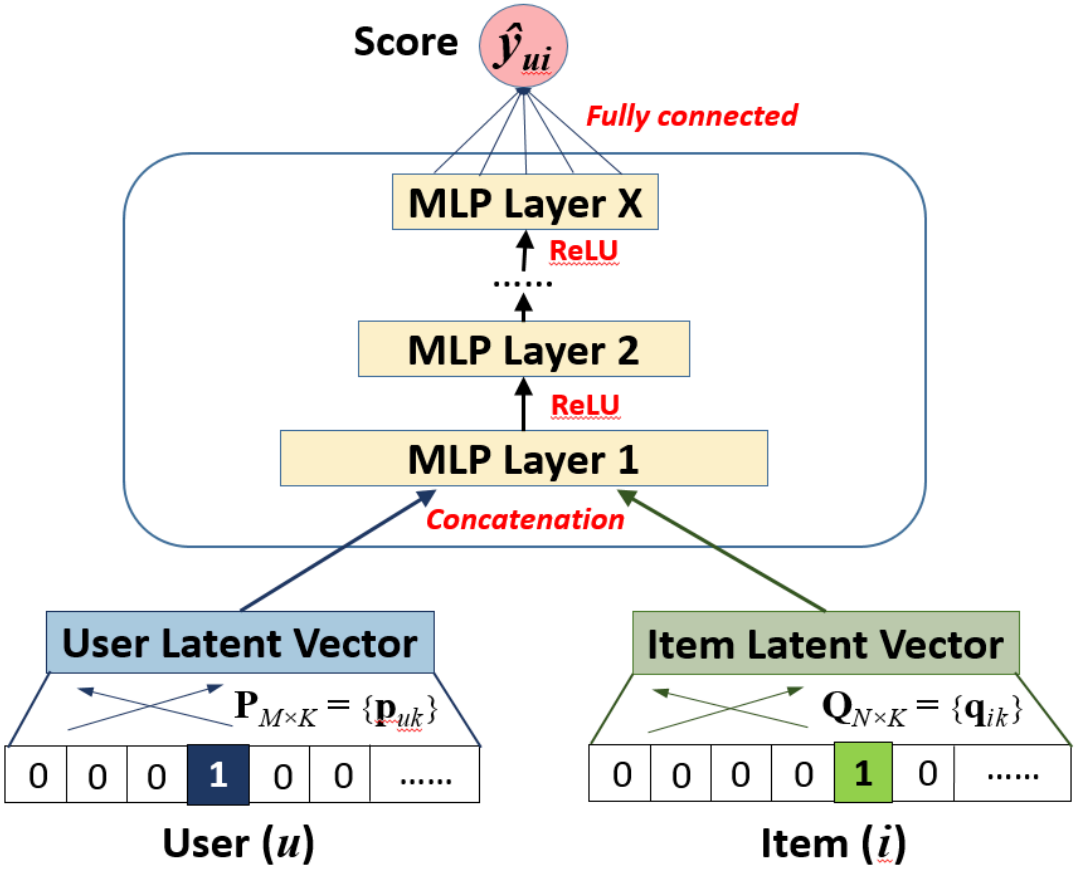


NN-based Matching function

Figure 2: Neural collaborative filtering framework

Multi-Layer Perceptron for CF

- The most intuitive idea is to use a Multi-Layer Perceptron as the matching function.

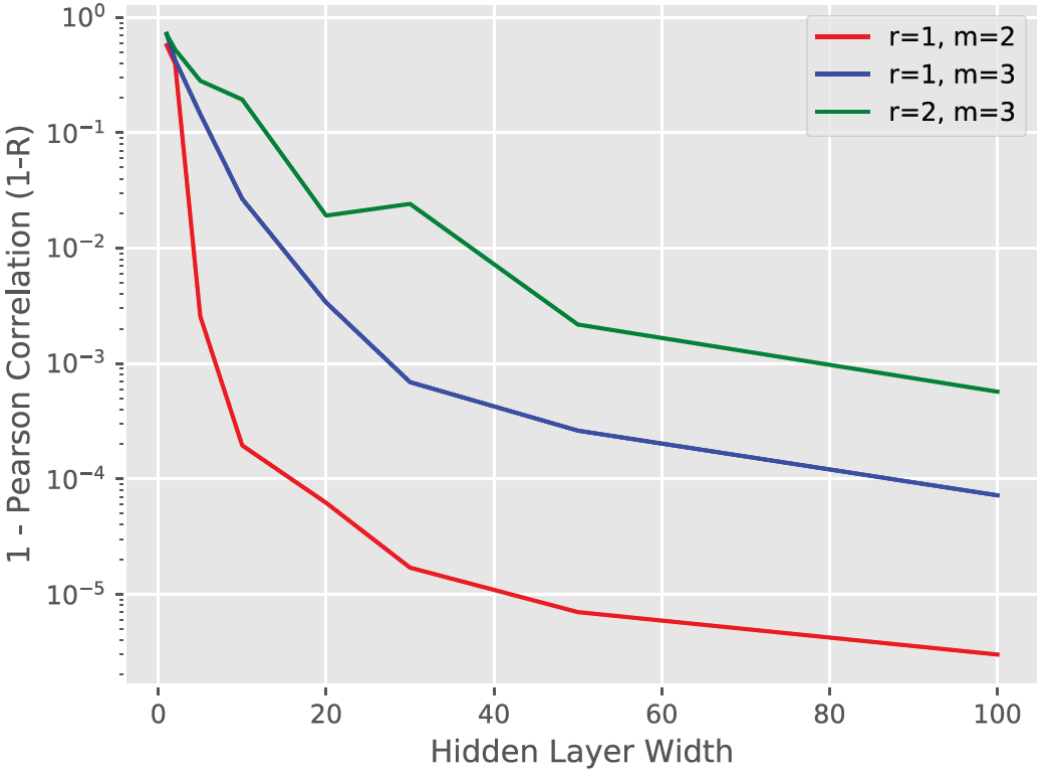


Unfortunately, MLP doesn't perform well and underperforms MF.

Why?

MLP is Weak in Capturing Low-Rank Relation (Beutel et al, WSDM'18)

Setting: Generating low-rank data, and using one-layer MLP to fit it.
r: rank size; m: data dimension (2 →matrix; 3 →3D tensor).

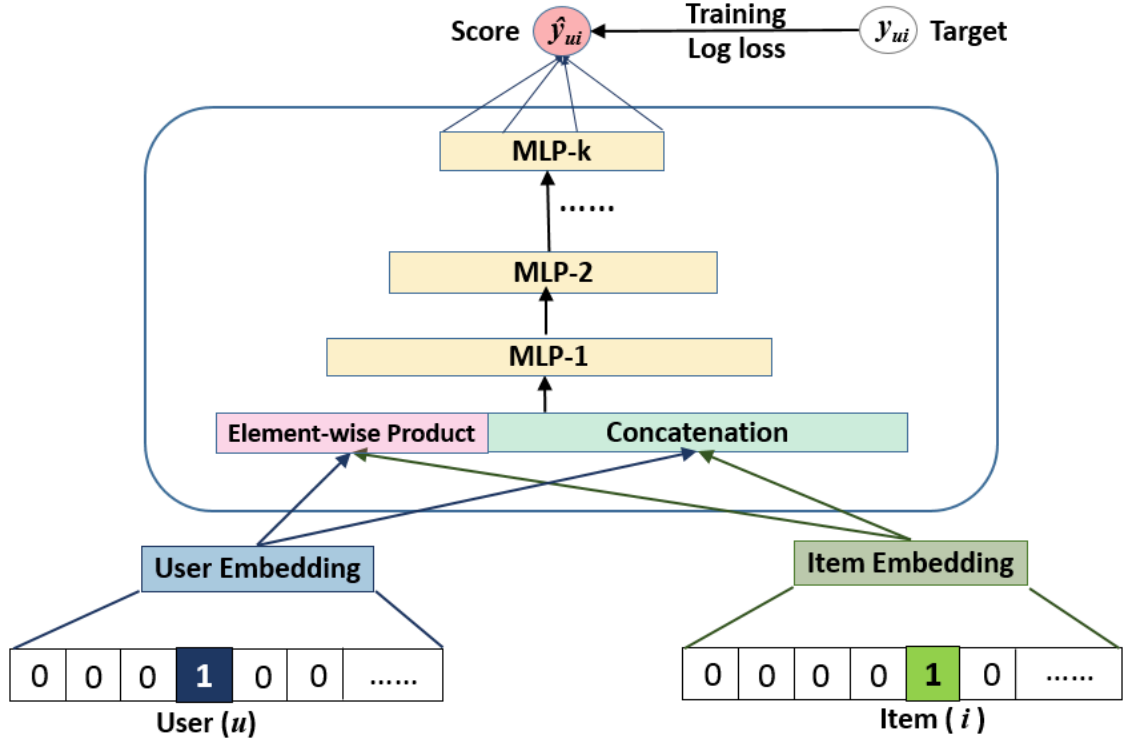


MLP can learn to approximate the low-rank relation, but is inefficient in doing so.

We have to design more effective models to make DNN work for CF!

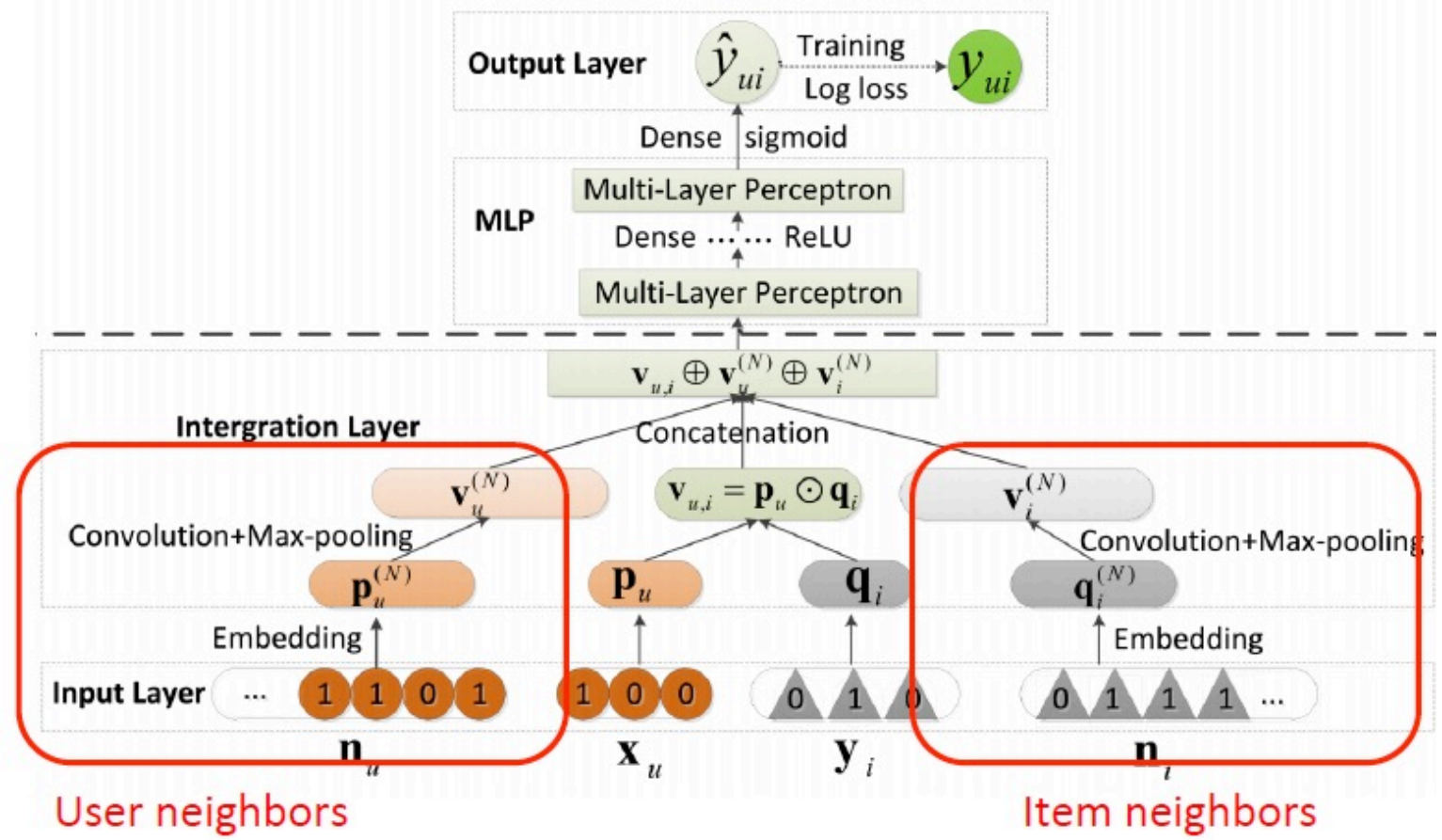
NeuMF: Neural Matrix Factorization (He et al, WWW'17)

- NeuMF unifies the strengths of MF and MLP in learning the matching function:
  - MF uses inner product to capture the low-rank relation
  - MLP is more flexible in using DNN to learn the matching function.



NNCF: Neighbor-based NCF (Bai et al, CIKM'17)

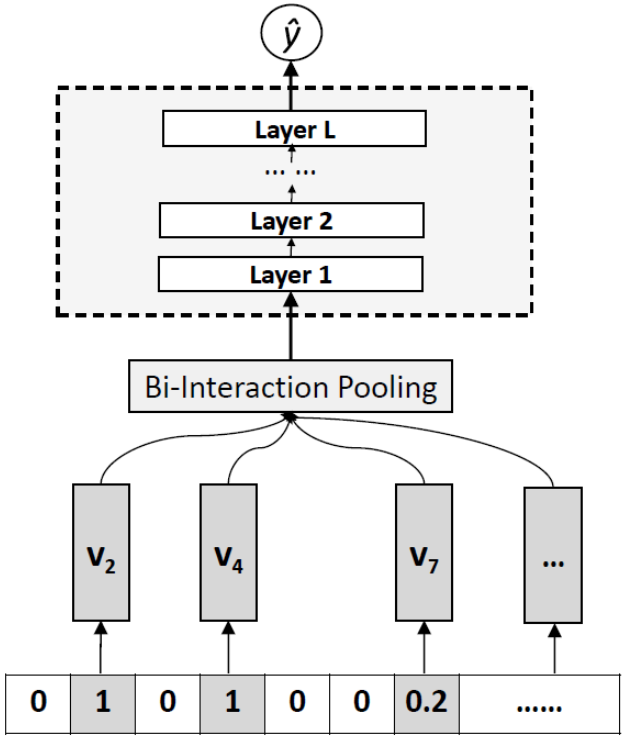
- Feeding user and item neighbors into the NCF framework



Can FMs be Extended to Neural Models?

NFM: Neural Factorization Machine (He and Chua, SIGIR'17)

- Inspired by FM, NFM models pairwise interactions between feature embeddings with multiplication.



Prediction Score

Hidden Layers → "Deep layers" learn higher-order feature interactions only, being much easier to train.

B-Interaction Layer

Embedding Layer

Input Feature Vector (sparse)

Bilinear Interaction Pooling:

$$f_{BI}(\mathcal{V}_x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i \mathbf{v}_i \odot x_j \mathbf{v}_j,$$

Figure 2: Neural Factorization Machines model (the first-order linear regression part is not shown for clarity).

Summary

- Deep methods for collaborative filtering
  - User/Item representation learning
  - Matching function learning
- Deep methods for feature-based recommendation
  - Cross features are important to encode

Addressing Cold Start in Recommender Systems: A Semi-supervised Co-training Algorithm

Mi Zhang, Jie Tang, Xuchen Zhang, Xiangyang Xue
Fudan University, Tsinghua University

[1] M. Zhang, J. Tang, X. Zhang, X. Xue. Addressing Cold Start in Recommender Systems: A Semi-supervised Co-training Algorithm. SIGIR'14, pp. 73-82.

Context-aware Factorization

- Our context-aware model:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u + \frac{\sum_{g \in genres(i)} b_{ug}}{|genres(i)|} + b_{ia} + b_{io} + b_{is} \quad (4)$$

user's preferences on a genre

item's preferences on user age interval

item's preferences on user gender

item's preferences on user occupation

- Model learning:

$$\min_{p_*, q_*, b_*} \sum_{(u,i) \in K} (r_{ui} - \hat{r}_{ui})^2 + \lambda(\|p_u\|^2 + \|q_i\|^2 + \sum b_*^2) \quad (5)$$

Semi-supervised Co-training

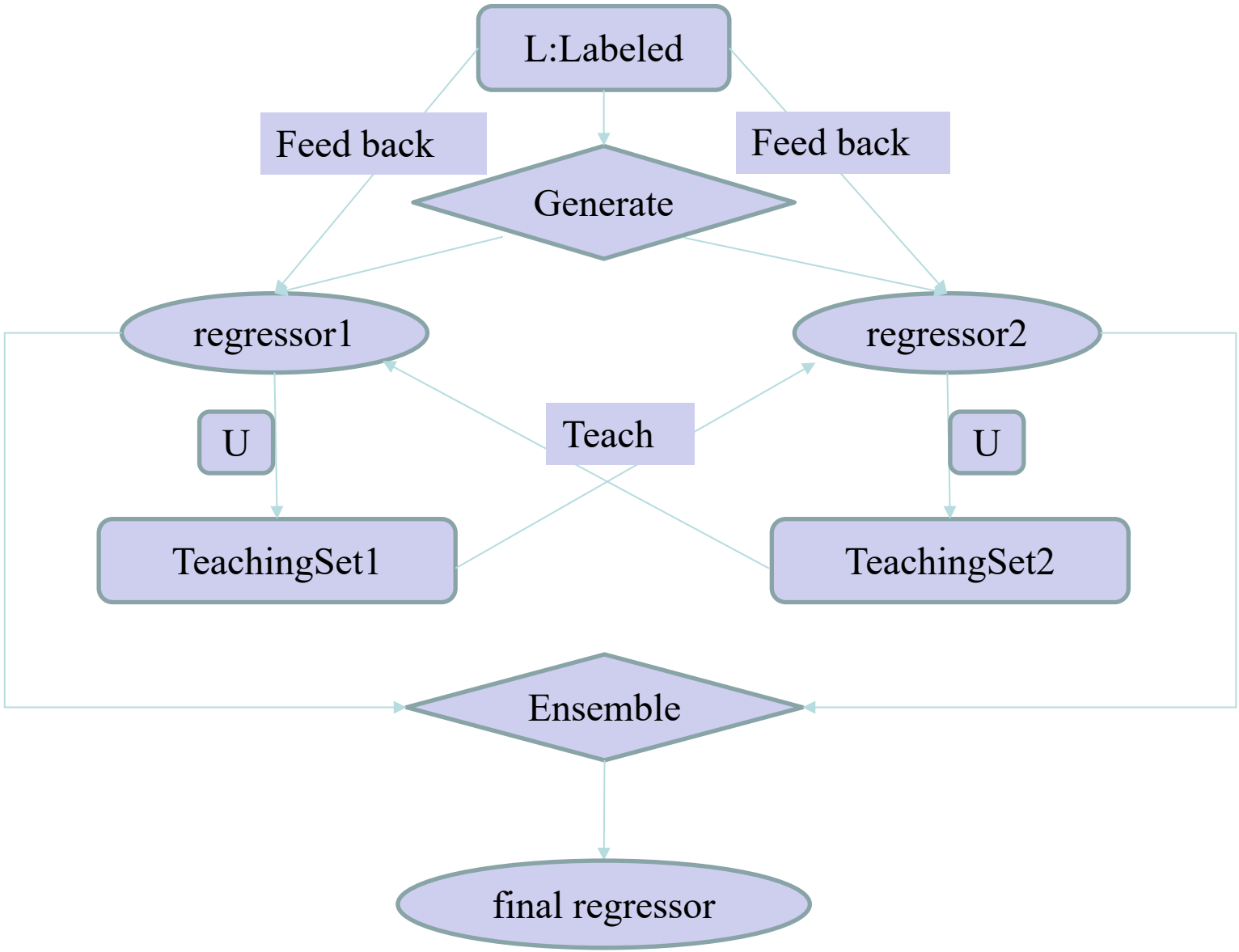
- Major three steps:

1.Constructing multiple regressors.

2.Co-Training.

3.Assembling the results.

Semi-supervised Co-training



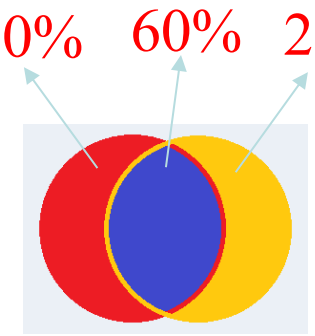
Semi-supervised Co-training

Major step 1: generate multiple regressor

1 Training Set
- 80%
- 80%

2 Attributes
- User context
- Item context

3 Hybrid
- User context 80%
- Item context 80%



User context

$$h_2(u,i) = \mu + b_u + b_i + q_i^T p_u + b_{ia} + b_{io} + b_{is}$$

$$h_1(u,i) = \mu + b_u + b_i + q_i^T p_u + \frac{1}{|genres(i)|} \sum_{g \in genres(i)} b_{ug}$$

Item context

Semi-supervised Co-training

Major step 2:

Select teachingset

Confidence Roulette for regressor2



$$C_j(x_{ui}) = \frac{d_u^{(j)} \times d_i^{(j)}}{\mathcal{N}}$$

U:unlabeled

U'

filter:
threshold

regressor2

TeachingSet2

Teach

regressor1

ensemble

regressor2

regressor2

Modelling Paying Behavior in Game Social Networks

Zhanpeng Fang[+], Xinyu Zhou[+], Jie Tang[+], Wei Shao[#], A.C.M. Fong[*], Longjun Sun[#], Ying Ding[-], Ling Zhou[+], and Jarder Luo[+]

[+]Tsinghua University

[#]Tencent Corporation

[*]Auckland University of Technology

[-]Indiana University

[1] Z. Fang, X. Zhou, J. Tang, W. Shao, A.C.M. Fong, L. Sun, Y. Ding, L. Zhou, and J. Luo. Modeling Paying Behavior in Game Social Networks. CIKM'14, pp. 411-420.

Billion Dollar Industry

- Facebook[1]
  - 250 million monthly players
  - 200 games with >1 million active players
  - 12% revenue

- Tencent[2] (Market Cap: ~150B $)
  - >400 million players
  - 50% revenue



[1] Facebook 2013 First Quarter Report
[2] Tencent 2013 Anual Report

Billion Dollar Industry

- Facebook[1]
  - 250 million monthly players
  - 200 games with >1 million active players

Not only keep players playing, but also make them pay.

- Tencent[2] (Market Cap: ~190B $)
  - >400 million players
  - 50% revenue



[1] Facebook 2013 First Quarter Report
[2] Tencent 2013 Anual Report

What we do

- Given users' data in online games, predict:

Free users -> Paying users

What we do

- Given users' data in online games, predict:

Free users -> Paying users

- Our goal:
  - Fundamental factors
  - Social effect
  - Predictive model

Two games: DNF

- Dungeon & Fighter Online (DNF)
  - Fight enemies by individuals or groups
  - 400+ million users
  - 2nd largest online game in China.





Two games: QQ Speed

- QQ Speed
  - Car racing against other users
  - 200+ million users



Datasets

- Statistics of the datasets

Category	Type	QQSpeed	DNF
User	all users	7.60M	347K
	free users	$\sim 10^6$	$\sim 10^5$
	paying users	$\sim 10^6$	$\sim 10^5$
	new payers	$\sim 10^5$	$\sim 10^4$
Relationship	co-playing	134M	7.30M
Guild	guilds	600K	49.6K
	co-guild	66.7M	51.7M
Activity	activity types	58	64
	activity logs	44.7B	5.71B
Date span	from	2013.6.20	2013.4.1
	to	2013.8.20	2013.6.30

Datasets

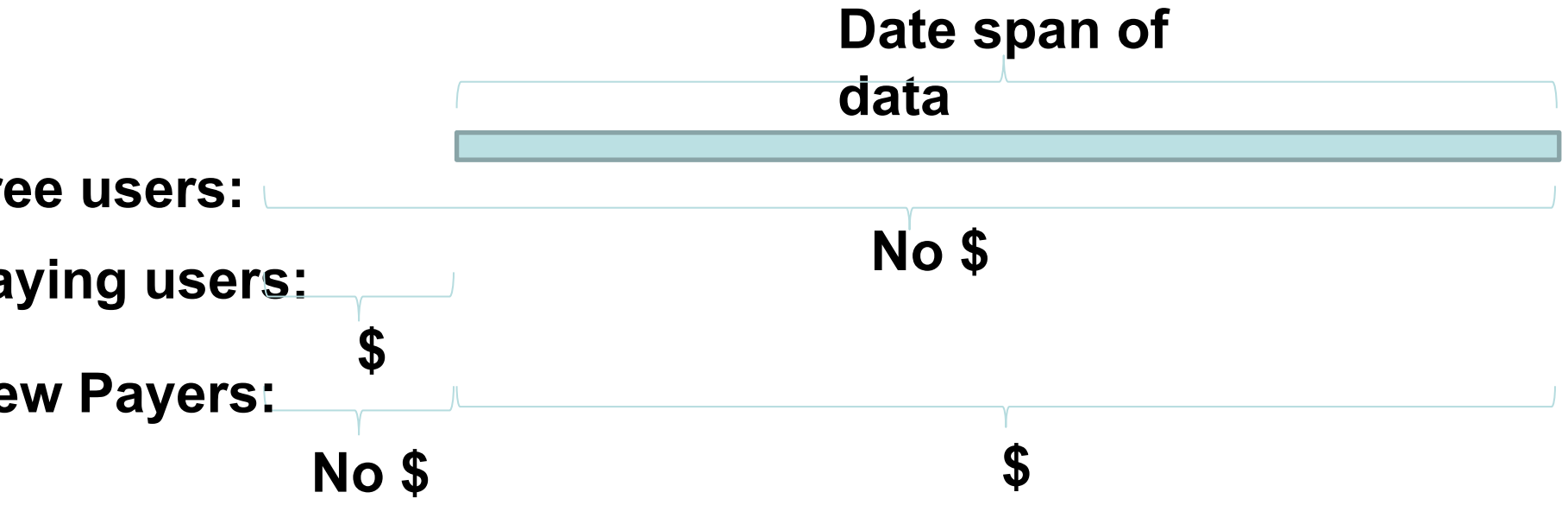
- Statistics of the datasets

Category	Type	QQSpeed	DNF
User	all users	7.60M	347K
	free users	$\sim 10^6$	$\sim 10^5$
	paying users	$\sim 10^6$	$\sim 10^5$
	new payers	$\sim 10^5$	$\sim 10^4$
Relationship	co-playing	134M	7.30M
Guild	guilds	600K	49.6K
	co-guild	66.7M	51.7M
Activity	activity types	58	64
	activity logs	44.7B	5.71B
Date span	from	2013.6.20	2013.4.1
	to	2013.8.20	2013.6.30

Datasets

- Statistics of the datasets

Category	Type	QQSpeed	DNF
User	all users	7.60M	347K
	free users	$\sim 10^6$	$\sim 10^5$
	paying users	$\sim 10^6$	$\sim 10^5$
	new payers	$\sim 10^5$	$\sim 10^4$



Datasets

- Statistics of the datasets

Category	Type	QQSpeed	DNF
User	all users	7.60M	347K
	free users	$\sim 10^6$	$\sim 10^5$
	paying users	$\sim 10^6$	$\sim 10^5$
	new payers	$\sim 10^5$	$\sim 10^4$
Relationship	co-playing	134M	7.30M
Guild	guilds	600K	49.6K
	co-guild	66.7M	51.7M
Activity	activity types	58	64
	activity logs	44.7B	5.71B
Date span	from	2013.6.20	2013.4.1
	to	2013.8.20	2013.6.30

Observation – Two Questions

- How do demographic attributes affect users' paying behavior?

- How do social factors influence users' paying behavior?

Observation - Demographics

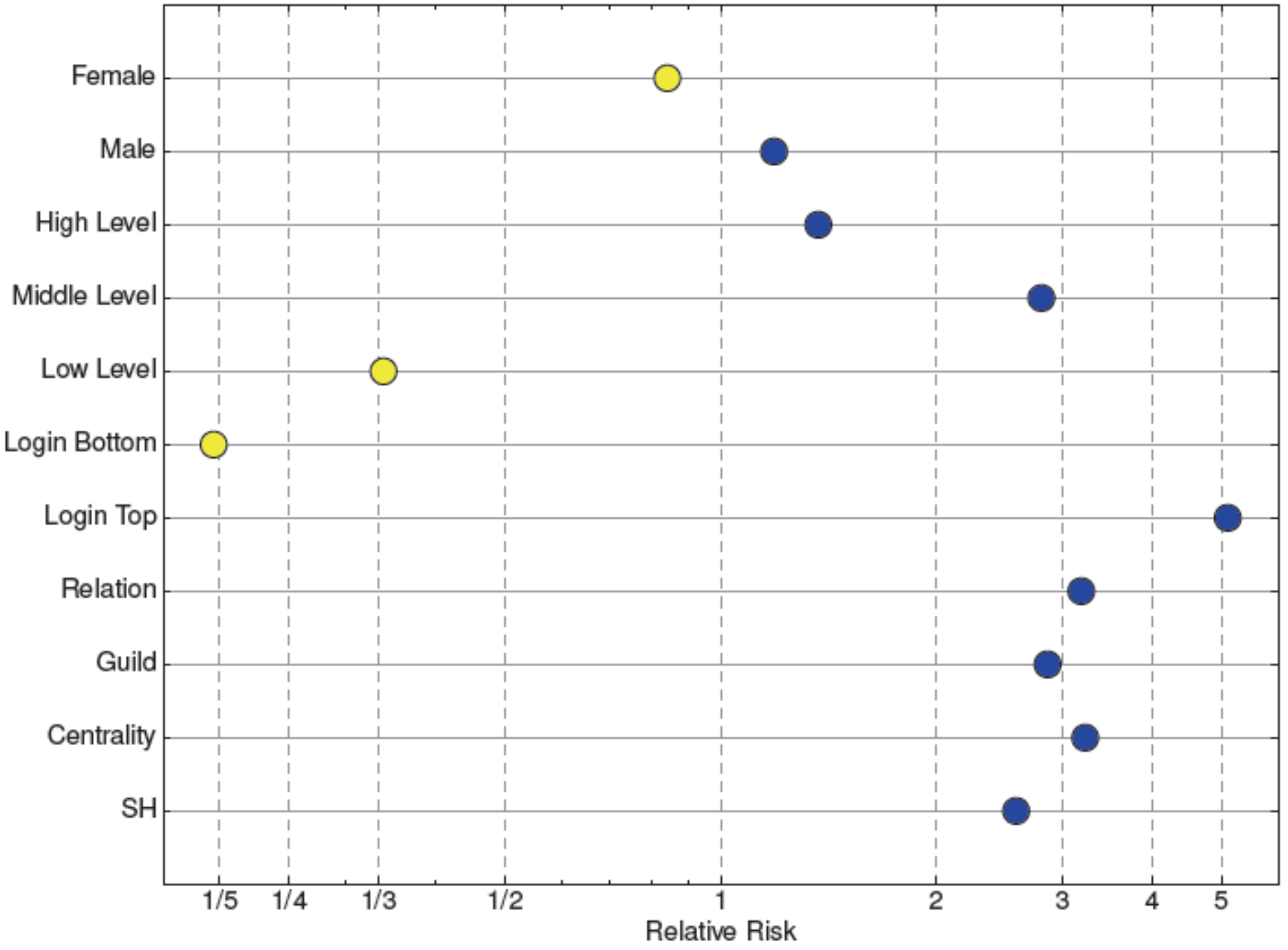
- Relative risk for attribute i:

$$RR(i) = \frac{P(\text{new payer}|\text{has attribute } i)}{P(\text{new payer}|\text{does not have attribute } i)}$$

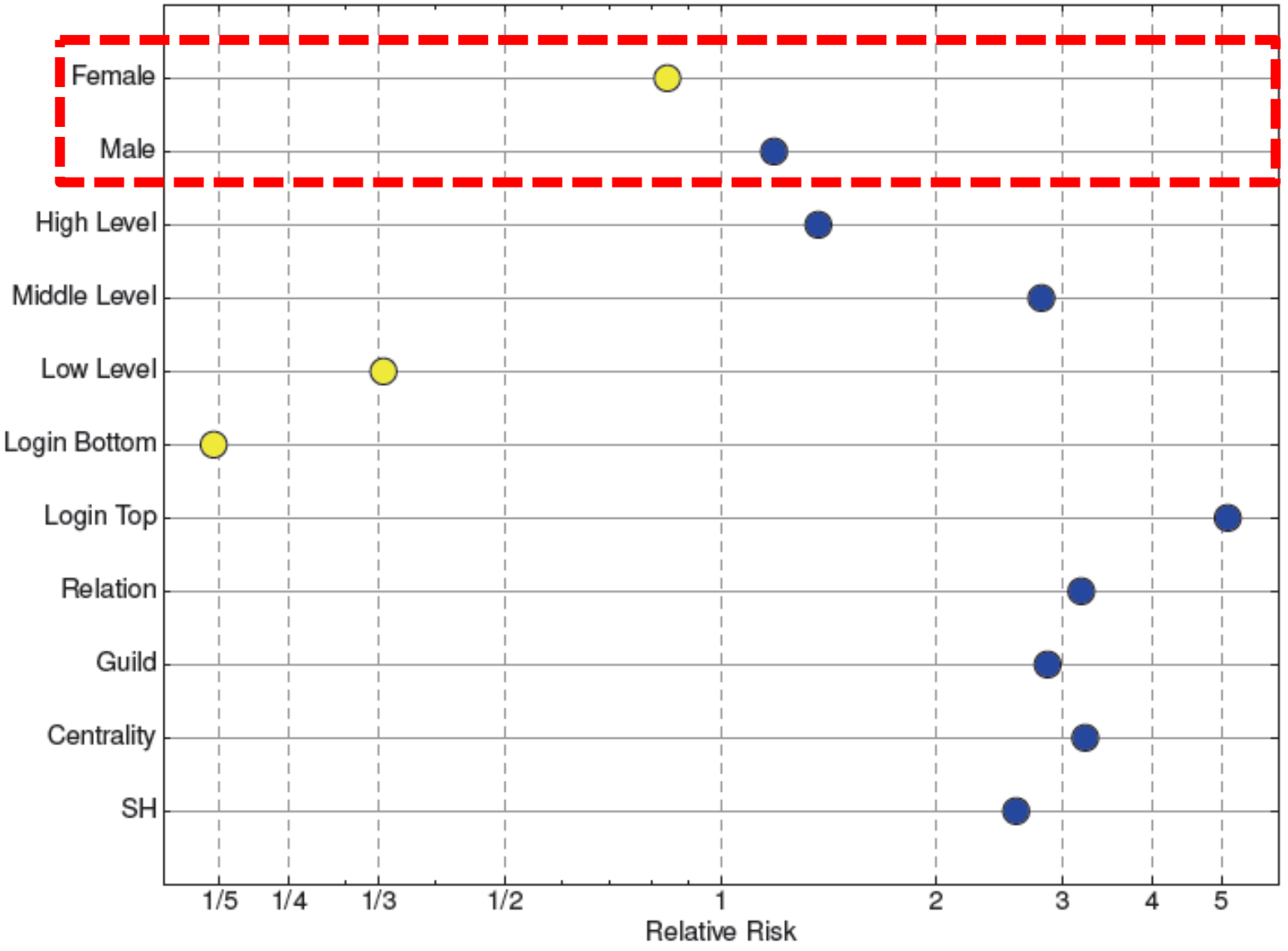
- RR(i) > 1: **more** likely to become paying users

- RR(i) < 1: **less** likely to become paying users

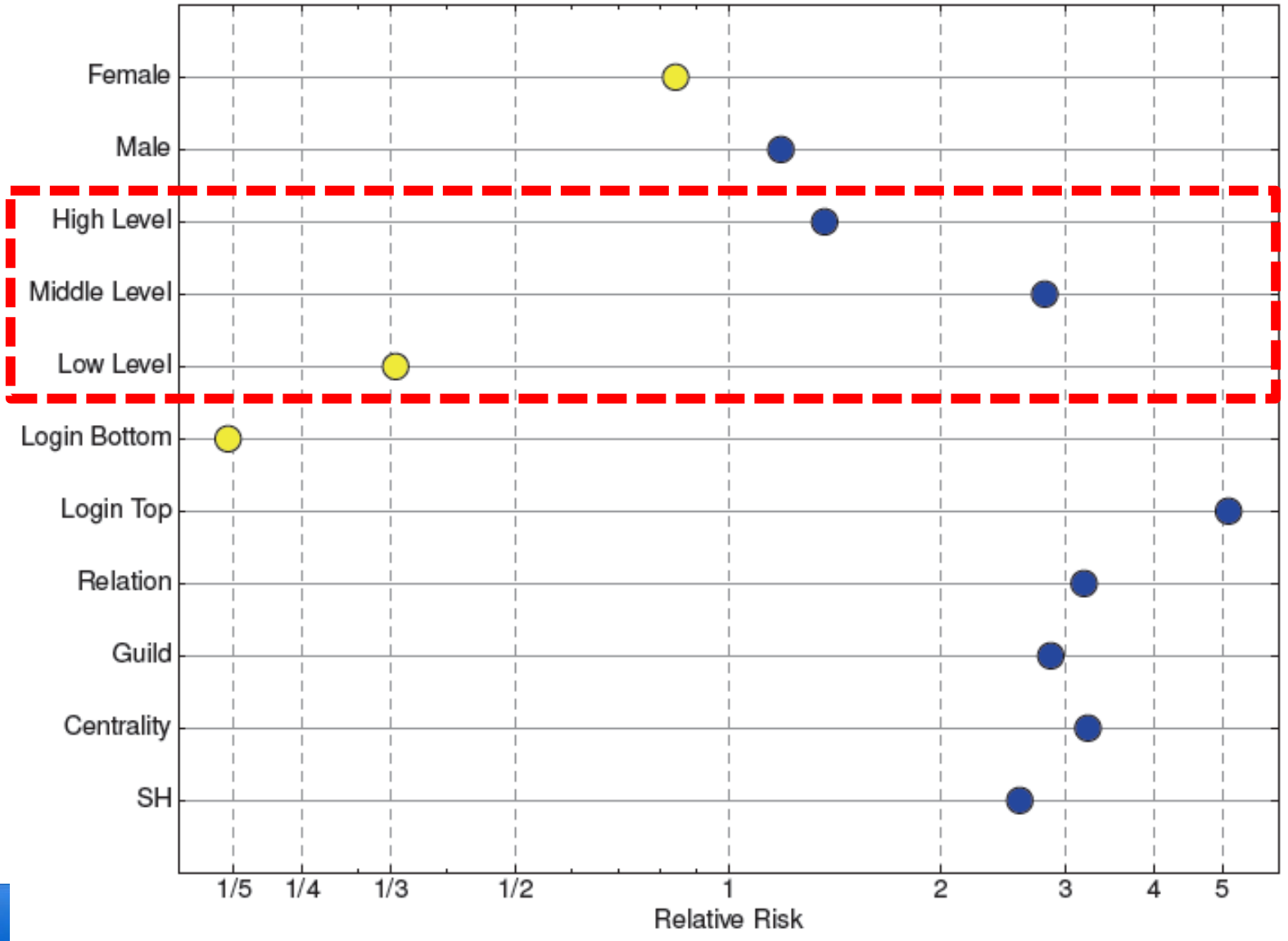
Observation - Demographics



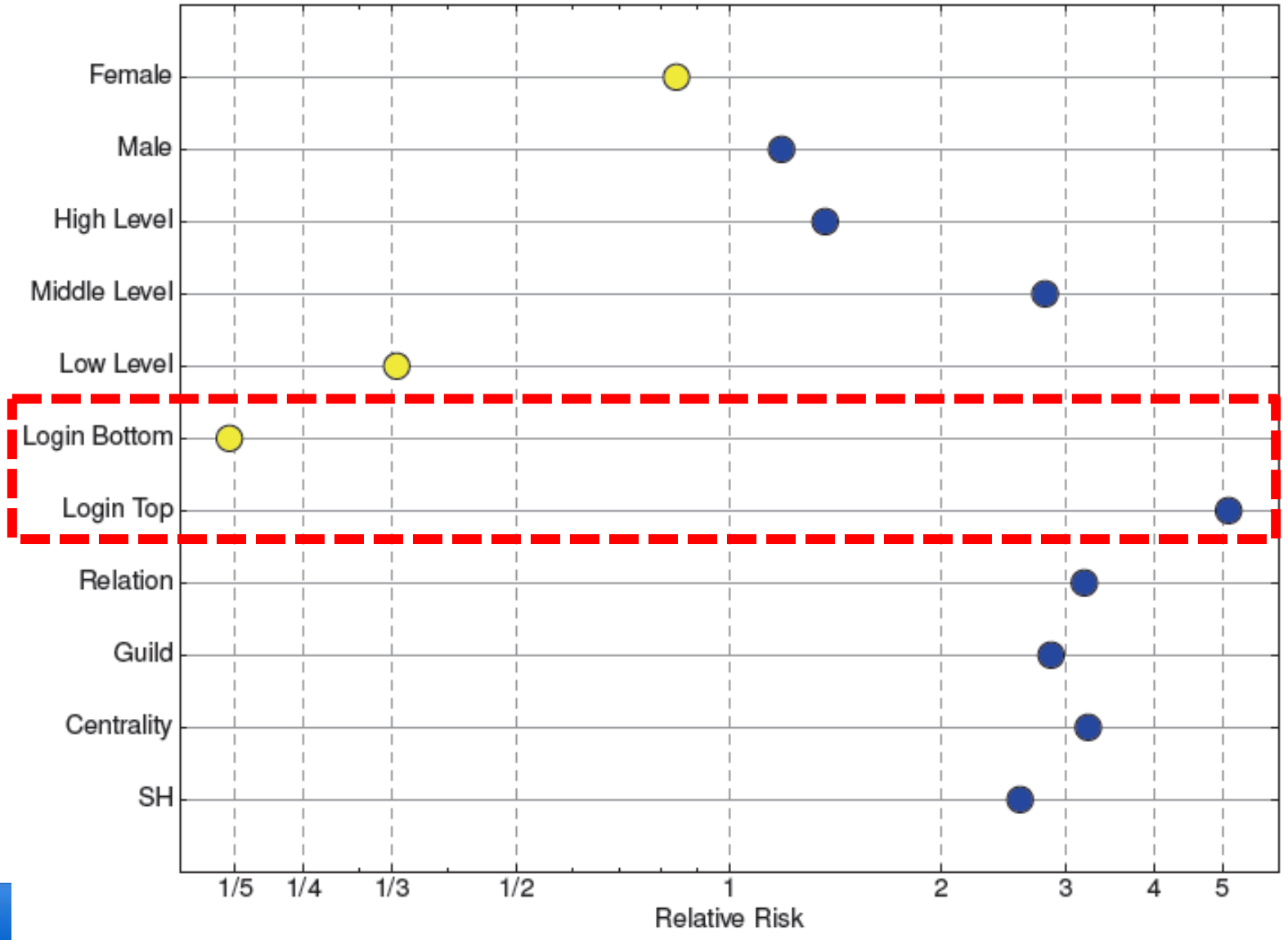
Observation - Demographics



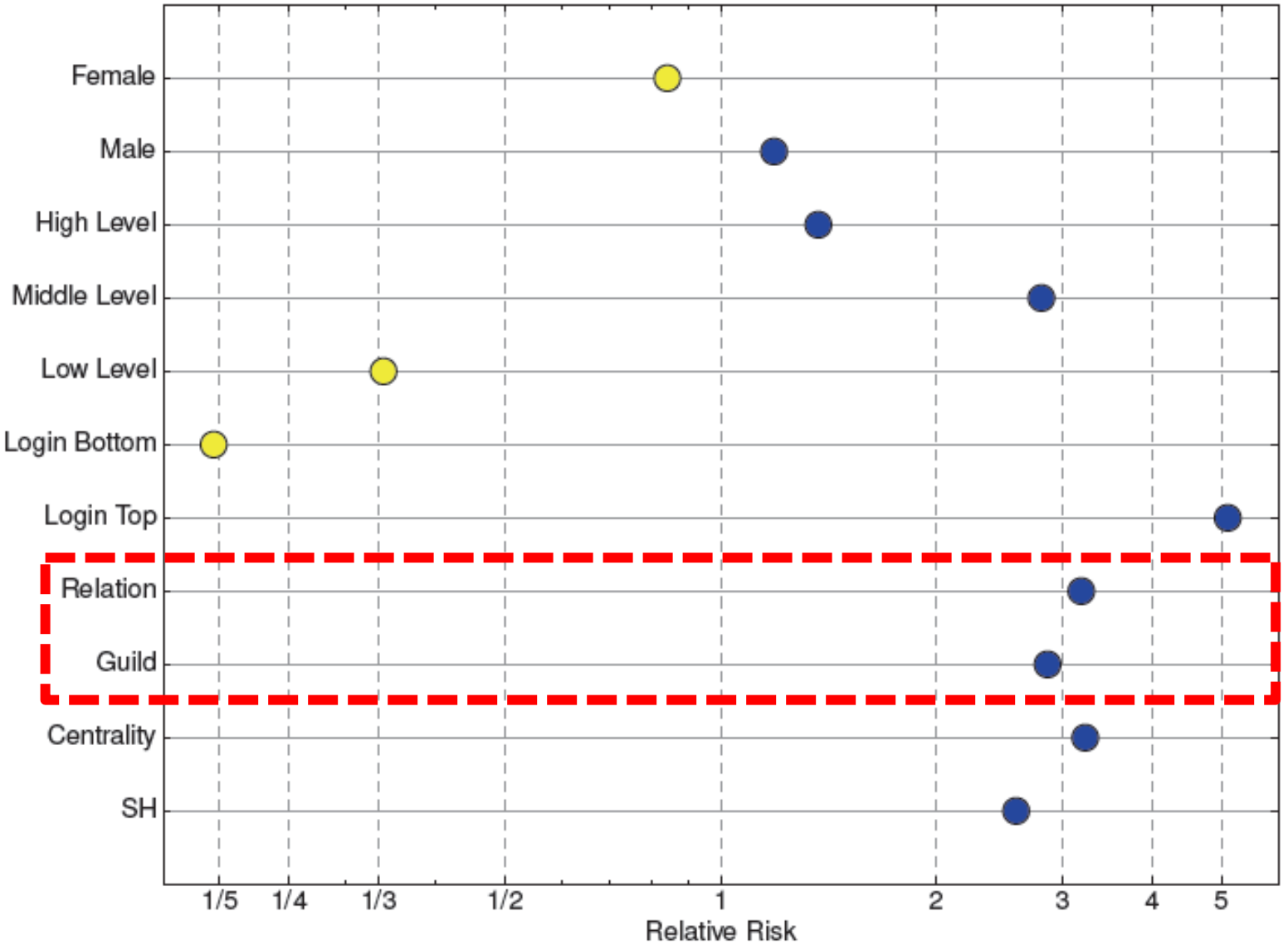
Observation - Demographics



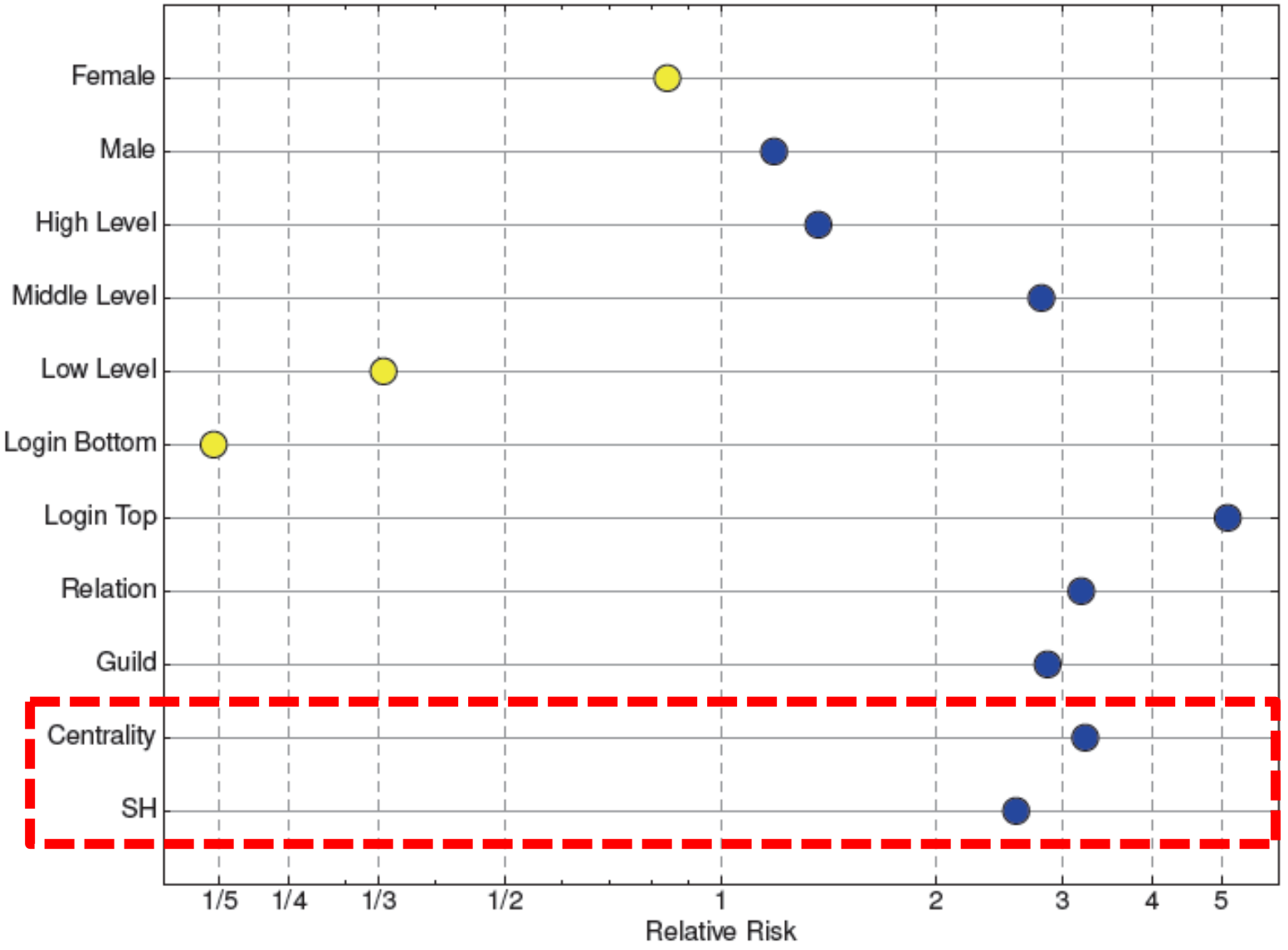
Observation - Demographics



Observation - Demographics



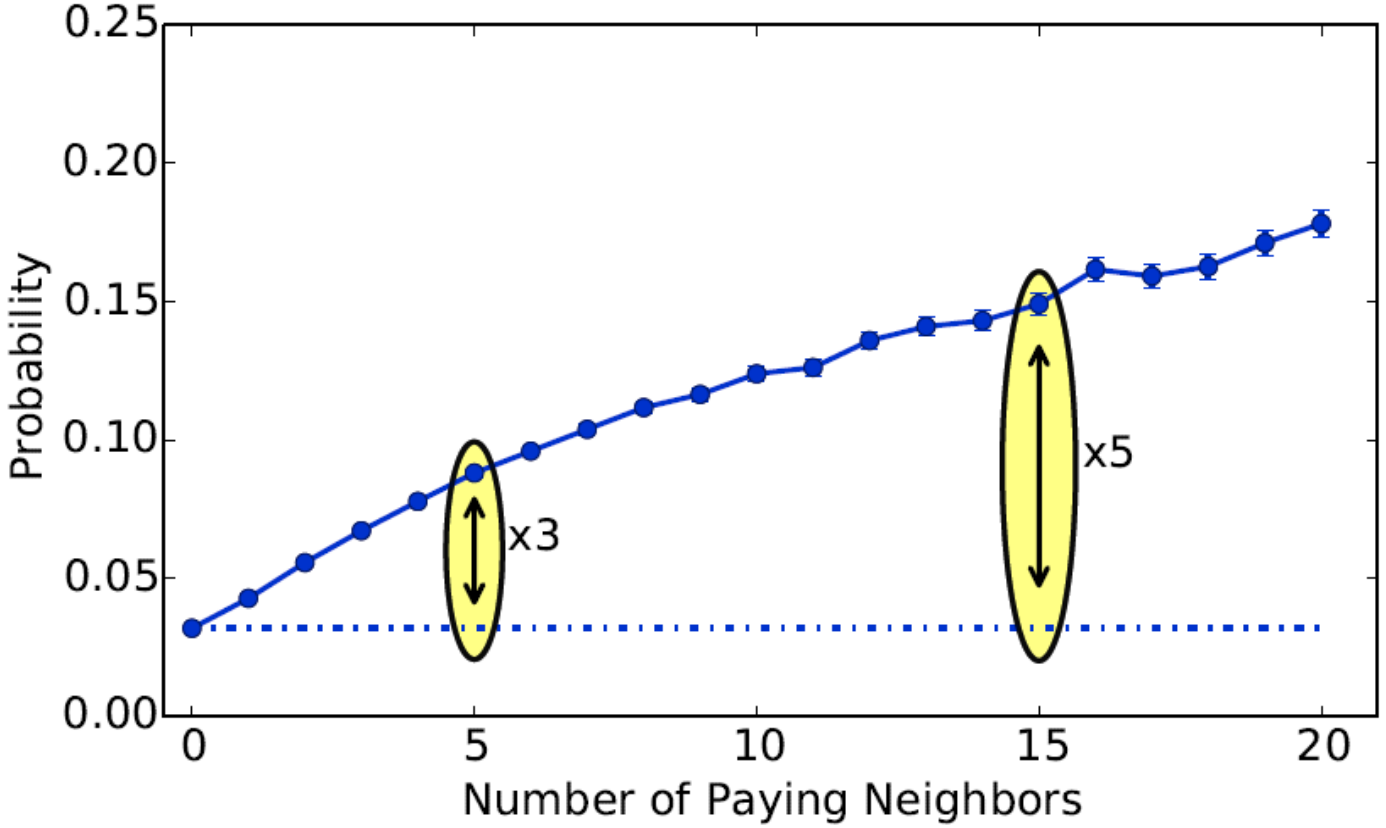
Observation - Demographics



Observation – Social Effects

- Social network construction
  - Co-playing network
- Social relationship
  - Social influence
  - Strong/Weak tie
  - Status
- Structural diversity

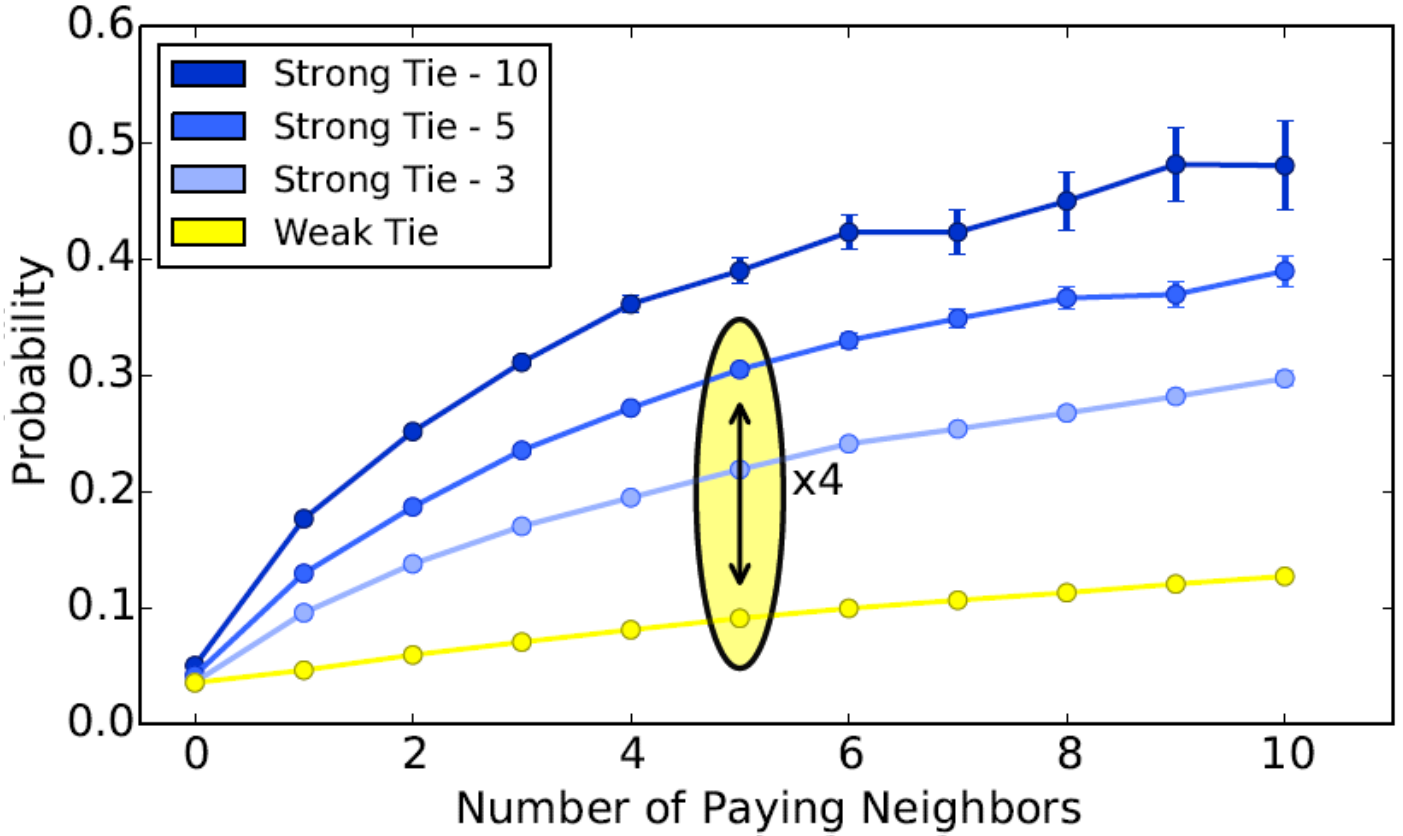
Social Relationship – Social Influence



Y: probability that a free user converts to a new payer

More paying neighbors ➡ Higher conversion probability

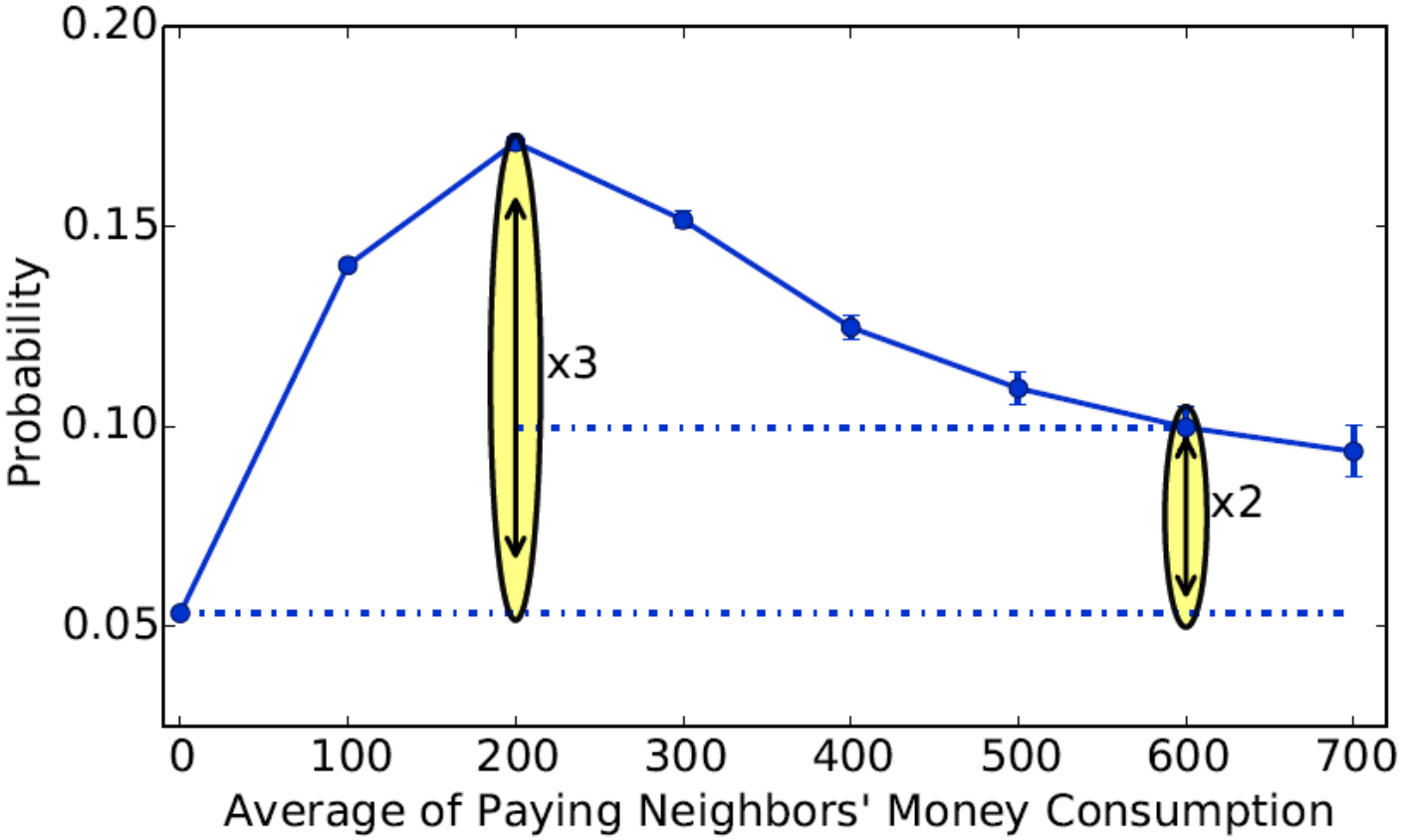
Social Relationship – Strong/Weak Tie



Y: probability that a free user converts to a new payer

Strong ties ➡ Strong influence

Social Relationship – User Status



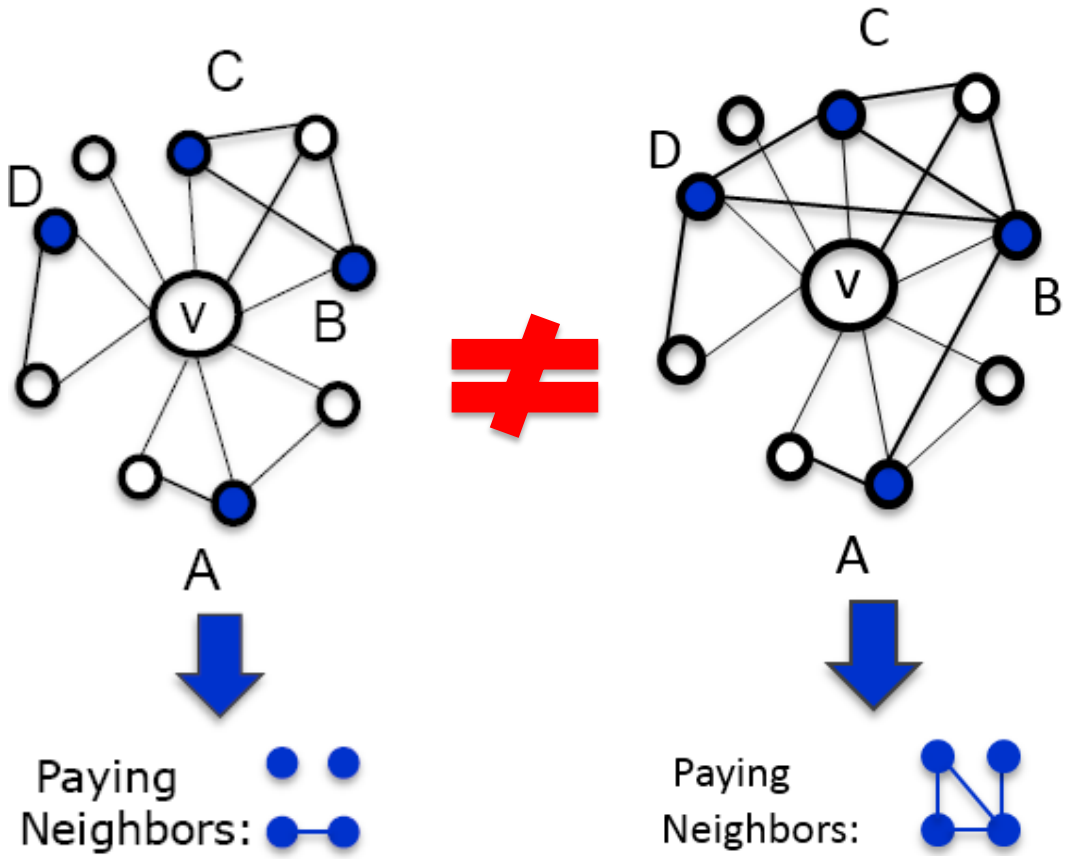
Y: probability that a free user converts to a new payer

Neighbors' money consumption increases



Conversion probability follows a unimodal shape

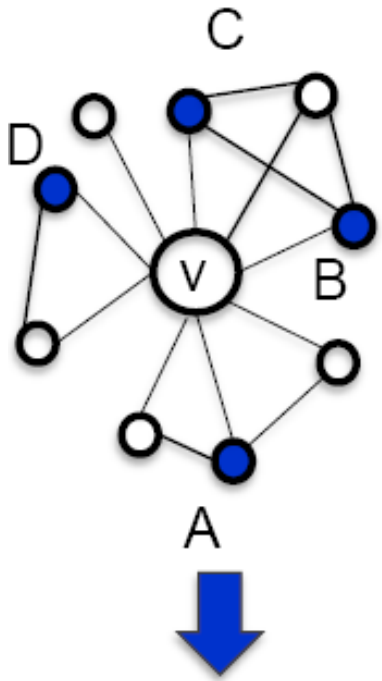
Structure Diversity

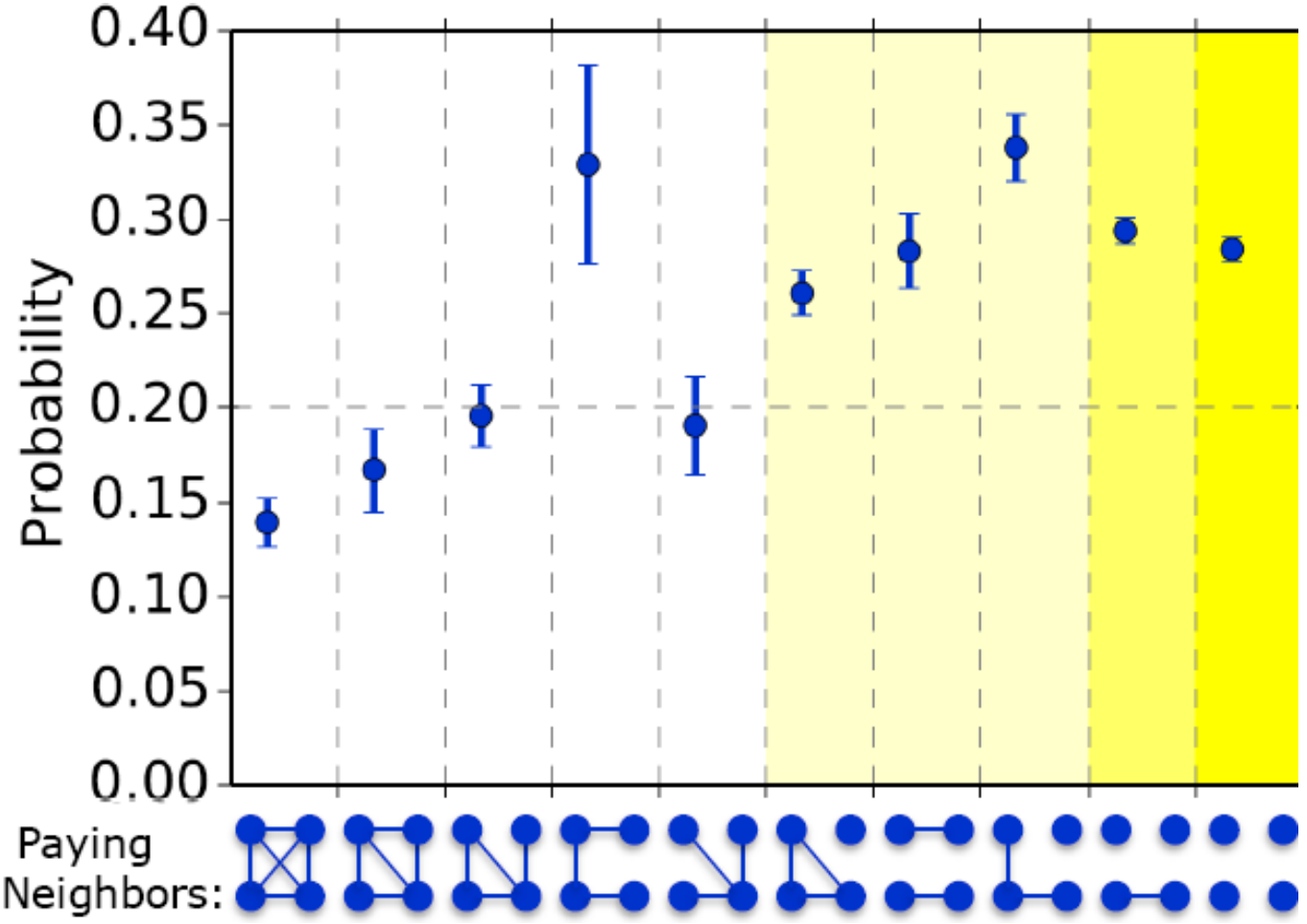


Different structures of a user's neighbors have different effects on the user's behavior[1]

[1] Ugander, J., Backstrom, L., Marlow, C., & Kleinberg, J. Structural diversity in social contagion. In PNSA'12.

Structure Diversity



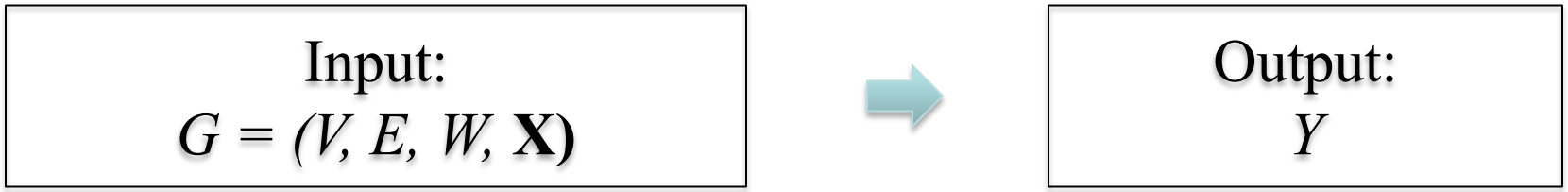


Extracted Features

- User attributes features
- Social effect features
- In-game behavior features
  - #purchased items
  - sum of virtual money consumption
  - etc.

Model Framework - Notations

- $G=(V, E, W, X)$ be a social network
- $W_{ij} \in W$: strength of the relationship
- $x_i \in X$: feature vector for user v_i
- $y_i \in Y$: paying potential for user v_i



Factorization Machines

- The prediction for feature vector x_i:

$$\hat{y}(\mathbf{x}_i) = w_0 + \sum_{j=1}^{d} w_j x_{i,j} + \sum_{j=1}^{d-1} \sum_{j'=j+1}^{d} x_{i,j} x_{i,j'} \langle \mathbf{p}_j, \mathbf{p}_{j'} \rangle$$

- Model parameters:

$$\Theta = \{w_0, w_1, ..., w_d, p_{1,1}, ..., p_{d,k}\}$$

Factorization Machines

- The prediction for feature vector x_i:

$$\hat{y}(\mathbf{x}_i) = w_0 + \sum_{j=1}^{d} w_j x_{i,j} + \sum_{j=1}^{d-1} \sum_{j'=j+1}^{d} x_{i,j} x_{i,j'} \langle \mathbf{p}_j, \mathbf{p}_{j'} \rangle$$

- Model parameters:

$$\Theta = \{w_0, w_1, ..., w_d, p_{1,1}, ..., p_{d,k}\}$$

- It can be rewritten as:

$$\hat{y}(\mathbf{x}_i) = w_0 + \sum_{j=1}^{d} w_j x_{i,j} + \frac{1}{2} \sum_{l=1}^{k} \left[\left(\sum_{j=1}^{d} p_{j,l} x_{i,j} \right)^2 - \sum_{j=1}^{d} p_{j,l}^2 x_{i,j}^2 \right]$$

Factorization Machine (cont')

- Objective function:

$$\mathcal{O}(\Theta) = \sum_{v_i \in V} \left(\hat{y}(\mathbf{x}_i) - y_i\right)^2 + \lambda \sum_{i=1}^{d} \|\mathbf{p}_i\|^2$$

- Solve by Stochastic Gradient Descent (SGD)

$$\frac{\partial \hat{y}(\mathbf{x})}{\partial \theta} = \begin{cases} 1, & \text{if } \theta \text{ is } w_0 \\ x_i, & \text{if } \theta \text{ is } w_i \\ x_i \sum_{j=1}^{d} p_{j,l} x_j - p_{j,l} x_i^2, & \text{if } \theta \text{ is } p_{i,l} \end{cases}$$

Local Consistent FM Model

- Consistency degree between two nodes:

$$c_{i,j} = \frac{\log(1 + W_{i,j})}{\sum_{v_{j'} \in NB(v_i)} \log(1 + W_{i,j'})}$$

- Incorporate the local consistency factor by a regularization term:

$$\mathcal{O}(\Theta) = \sum_{v_i \in V} \left(\hat{y}(\mathbf{x}_i) - y_i\right)^2 + \lambda \sum_{i=1}^{d} \|\mathbf{p}_i\|^2 + \mu \sum_{v_i \in V} \sum_{v_j \in NB(v_i)} c_{i,j} \left(\hat{y}(\mathbf{x}_i) - \hat{y}(\mathbf{x}_j)\right)^2$$

Model Learning – Two-step approach

- First step
  - Optimize the FM terms in training data by SGD.

.

$$\mathcal{O}(\Theta) = \sum_{v_i \in V} \left(\hat{y}(\mathbf{x}_i) - y_i\right)^2 + \lambda \sum_{i=1}^{d} \|\mathbf{p}_i\|^2 + \mu \sum_{v_i \in V} \sum_{v_j \in NB(v_i)} c_{i,j} \left(\hat{y}(\mathbf{x}_i) - \hat{y}(\mathbf{x}_j)\right)^2$$

Model Learning – Two-step approach

$$\mathcal{O}(\Theta) = \sum_{v_i \in V} \left(\hat{y}(\mathbf{x}_i) - y_i\right)^2 + \lambda \sum_{i=1}^{d} \|\mathbf{p}_i\|^2 + \mu \sum_{v_i \in V} \sum_{v_j \in NB(v_i)} c_{i,j} \left(\hat{y}(\mathbf{x}_i) - \hat{y}(\mathbf{x}_j)\right)^2$$

- Second step
  - Optimize the local consistency terms by local propagation.

$$\hat{y}_i = (1 - \gamma\mu)\hat{y}_i + \gamma \cdot \mu \sum_{v_j \in NB(v_i)} c_{i,j}\hat{y}_j$$

Where $\gamma \in [0, 1]$ is a parameter to control the propagation rate.

Time Complexity

- Our approach: $O(|V|T_1kd + |E|T_2)$
- Directly apply SGD: $O(|E|T_1kd)$

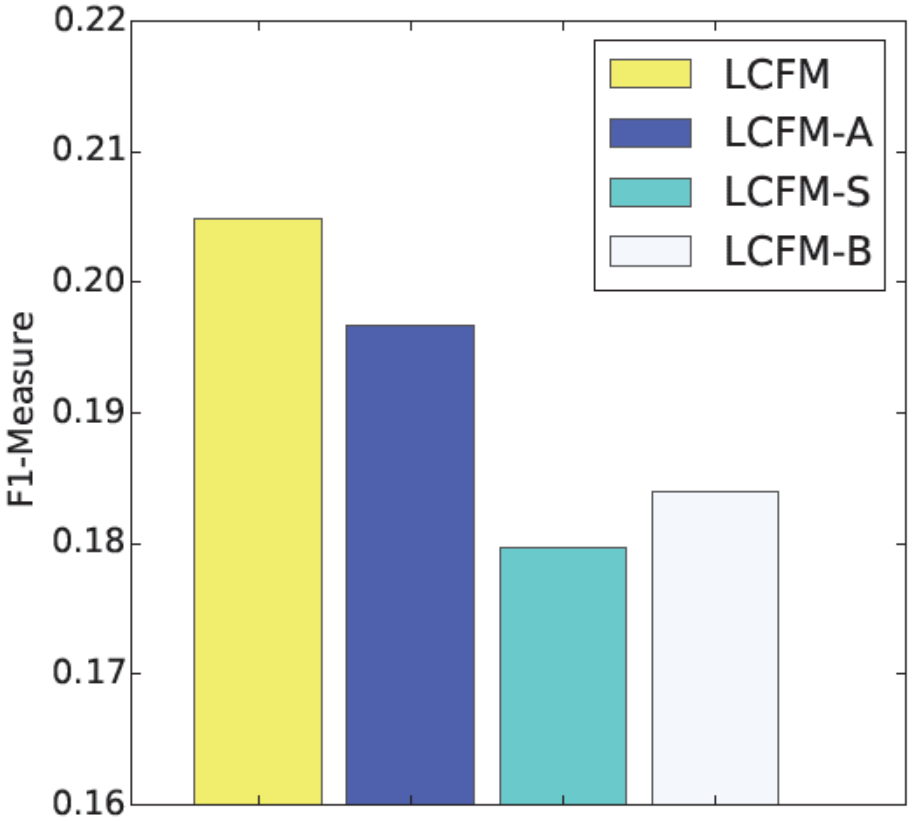
Experimental Setup

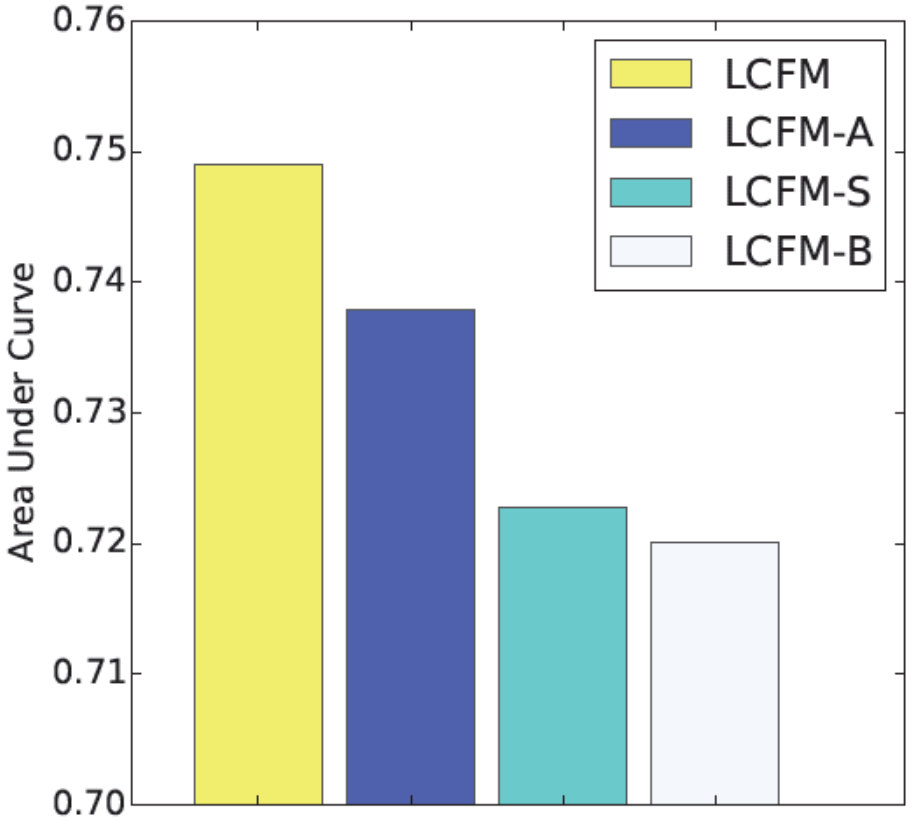
- Prediction setting
  - Predict whether a free user will become a new payer
  - Split the datasets into training and test sets by time
- Evaluation measures
  - Precision (Prec.)
  - Recall (Rec.)
  - F1-Measure (F1)
  - Area under Curve (AUC)

Results of Different Methods

Data	Method	AUC	Rec.	Prec.	F1
QQSpeed	FM	73.61	33.16	13.62	19.31
	LRC	73.17	30.75	14.00	19.24
	SVM	72.78	32.72	14.13	19.74
	RF	73.57	33.36	13.52	19.25
	GBDT	73.64	28.88	14.44	19.25
	LCFM	**74.90**	**33.67**	**14.72**	**20.49**
DNF	FM	77.56	34.76	24.51	28.75
	LRC	77.03	34.78	24.25	28.57
	SVM	76.48	32.53	25.31	28.47
	RF	77.11	30.91	24.00	27.02
	GBDT	77.73	34.10	25.05	28.88
	LCFM	**78.32**	**35.66**	**25.71**	**29.88**

Feature Contribution





LCFM-A: stands for removing attribute features

LCFM-S: stands for removing social effect features

LCFM-B: stands for removing in-game behavior features

Social Effect Contribution

	Features used	AUC	Rec.	Prec.	F1
A	Attribute&Behavior	72.28	32.30	12.45	17.97
	+Social influence	74.65	33.29	14.46	20.17(+2.20%)
	+Strong/Weak tie	74.75	33.31	14.67	20.37(+2.40%)
	+Status	74.08	32.39	13.88	19.43(+1.46%)
	+Structural diversity	74.75	32.39	14.86	20.38(+2.41%)
B	All Features	74.90	33.67	14.72	20.49
	-Social influence	74.88	33.73	14.67	20.45(-0.04%)
	-Strong/Weak tie	74.88	33.19	14.80	20.48(-0.01%)
	-Status	74.77	33.15	14.66	20.33(-0.16%)
	-Structural diversity	74.89	32.90	14.84	20.45(-0.04%)

Social Effect Contribution

	Features used	AUC	Rec.	Prec.	F1
A	Attribute&Behavior	72.28	32.30	12.45	17.97
	+Social influence	74.65	33.29	14.46	20.17(+2.20%)
	+Strong/Weak tie	74.75	33.31	14.67	20.37(+2.40%)
	+Status	74.08	32.39	13.88	19.43(+1.46%)
	+Structural diversity	74.75	32.39	14.86	20.38(+2.41%)
B	All Features	74.90	33.67	14.72	20.49
	-Social influence	74.88	33.73	14.67	20.45(-0.04%)
	-Strong/Weak tie	74.88	33.19	14.80	20.48(-0.01%)
	-Status	74.77	33.15	14.66	20.33(-0.16%)
	-Structural diversity	74.89	32.90	14.84	20.45(-0.04%)

Online Test

- Test setting
  - Two groups: *test group* and *control group*.
  - Send messages to invite the user to attend a promotion activity.



Online Test

- Test setting
  - Two groups: *test group* and *control group*.
  - Send messages to invite the user to attend a promotion activity.
- Evaluation metric:

$$Lift_Ratio = \frac{CR - CR_{prior}}{CR_{prior}}$$

where CR means the new payer **c**onverting **r**ate.

Prior strategy: suggests users mainly by their activities.

Online Test

- Test setting
  - Two groups: *test group* and *control group*.
  - Send messages to invite the user to attend a promotion activity.
- Evaluation metric:

$$Lift_Ratio = \frac{CR - CR_{prior}}{CR_{prior}}$$

where CR means the new payer **c**onverting **r**ate.

Prior strategy: suggests users mainly by their activities.

Online Test Results

- Online test 1
  - Test the effectiveness of our approach in online scenario.
  - Test group: LCFM
  - Control group: Prior strategy

	Online Test 1 2013.12.27 - 2014.1.3		Online Test 2 2014.1.24 - 2014.1.27		
Group name	test group	control group	test group	control group	prior group
Group size	600K	200K	400K	400K	200K
#Message read	345K	106K	229K	215K	106K
Message read rate	57.50%	53.00%	57.25%	53.75%	53.00%
#Message clicked	47584	7466	23325	20922	6299
Message clicked rate	7.93%	3.73%	5.83%	5.23%	3.15%
Lift_Ratio	196.87%	0%	126.81%	73.40%	0%

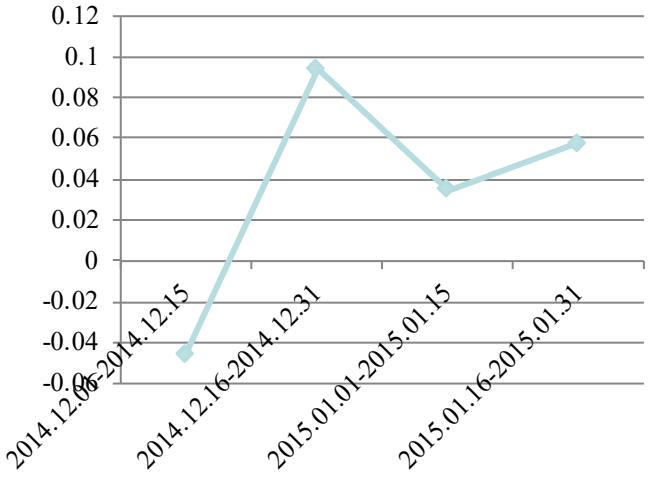
Online Test Results

- Online test 2
  - Test the contribution of social factors in online scenario.
  - Test group: LCFM
  - Control group: LCFM – Social effect features

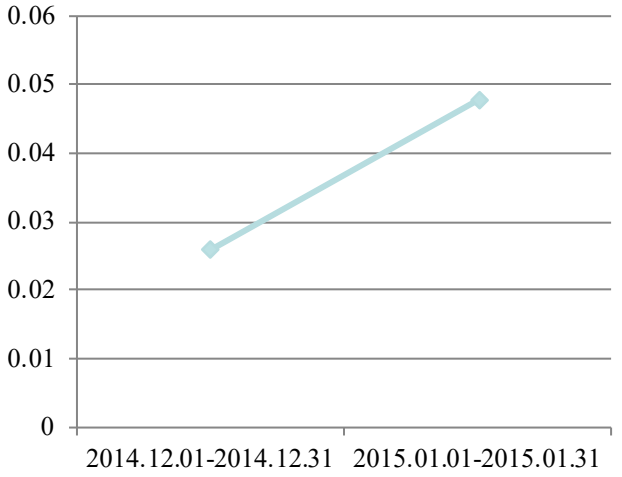
	Online Test 1 2013.12.27 - 2014.1.3		**Online Test 2 2014.1.24 - 2014.1.27**		
Group name	test group	control group	test group	control group	prior group
Group size	600K	200K	400K	400K	200K
#Message read	345K	106K	229K	215K	106K
Message read rate	57.50%	53.00%	57.25%	53.75%	53.00%
#Message clicked	47584	7466	23325	20922	6299
Message clicked rate	7.93%	3.73%	5.83%	5.23%	3.15%
Lift_Ratio	196.87%	0%	126.81%	73.40%	0%

Deploy

- The algorithm has been deployed in the online system (QQ Speed)
- Improve **154%**: convert free users into paying users
- Improve up to **9.4%**: total income



In one month



By month

Conclusion

- Discovered strong social influence on users' paying behavior in the game network.
- Proposed a LCFM model that incorporates network information into FM model.
- Confirmed the effectiveness of our approach by online test results.

[1] Z. Fang, X. Zhou, J. Tang, W. Shao, A.C.M. Fong, L. Sun, Y. Ding, L. Zhou, and J. Luo. Modeling Paying Behavior in Game Social Networks. CIKM'14, pp. 411-420.

References

- https://www.ismll.uni-hildesheim.de/aktuelles/tutorial-factorization-models-part2.pdf
- http://staff.ustc.edu.cn/~hexn/icmr18-recsys.pdf
- Steffen Rendle,“Factorization Machines,” in ICDM’10, Sydney, Australia.
- Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In WWW 2017.
- Xiangnan He, and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In SIGIR 2017.
- Ting Bai, Ji-Rong Wen, Jun Zhang, and Wayne Xin Zhao. A Neural Collaborative Filtering Model with Interaction-based Neighborhood. In CIKM 2017.

References

- Hong-Jian Xue, Xin-Yu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. Deep matrix factorization models for recommender systems. IJCAI 2017.
- Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince GaMo, and Ed H. Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In WSDM 2018.
- Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. Item silk road: Recommending items from information domains to social users. In SIGIR 2017.

Thanks!

Q&A

HP: http://keg.cs.tsinghua.edu.cn/jietang/

http://aminer.org