Recommender Systems

Lecture 7

Previous lecture

- Weighted Matrix Factorization
- Optimization methods
  - Stochastic gradient descent (SGD)
  - Alternating minimization (ALS)

$$\mathcal{J}(P,Q) = \frac{1}{2}\sum_{i,j\in\mathcal{O}} \left(a_{ij} - \boldsymbol{p}_i^\top \boldsymbol{q}_j\right)^2 + \lambda\left(\|\boldsymbol{p}_i\|^2 + \left\|\boldsymbol{q}_j\right\|^2\right)$$

Today's lecture

- User feedback
  - Understanding its nature
  - Different preprocessing schemes
  - Confidence-based ALS model

- Special methods in recsys
  - MF meets neighborhood models
  - MF meets similarity

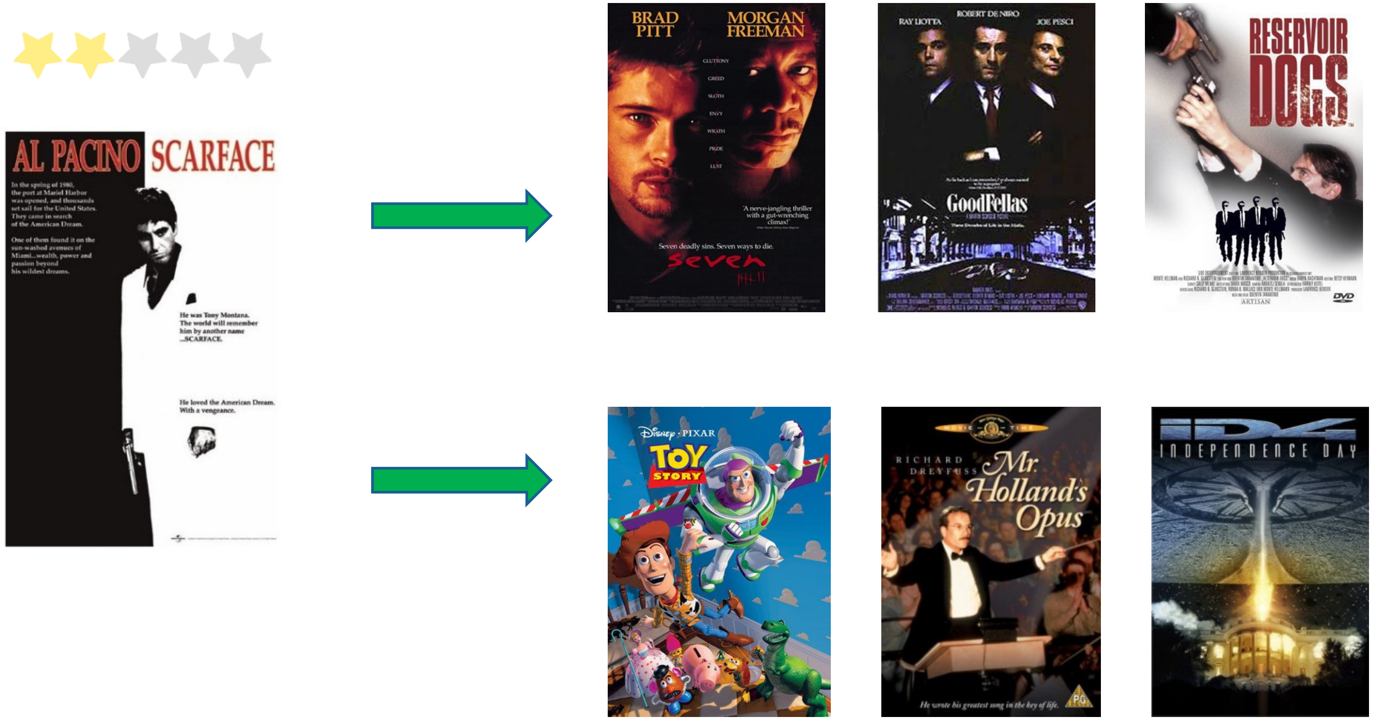
- Beyond pointwise optimization
  - Learning to rank approach

Many facets of user feedback

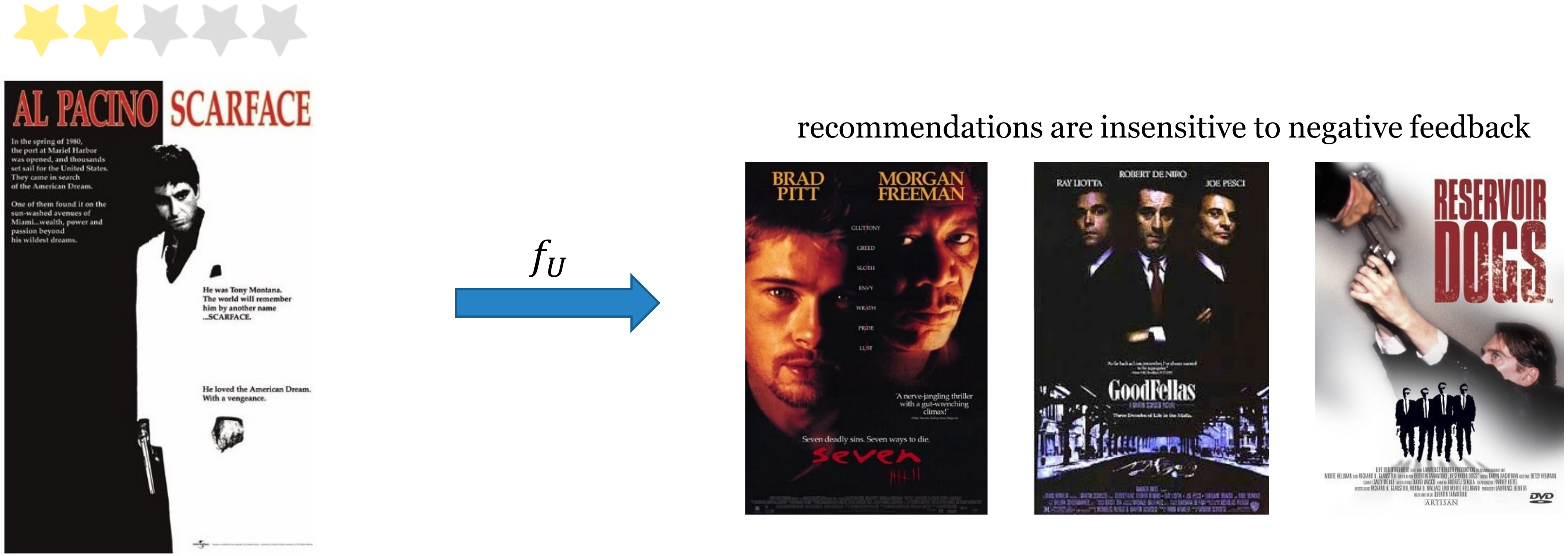
Guess standard algorithm behavior

What is likely to be recommended in this case?

User feedback is negative!
Probably he or she doesn’t like criminal movies.



Problematic warm start scenario

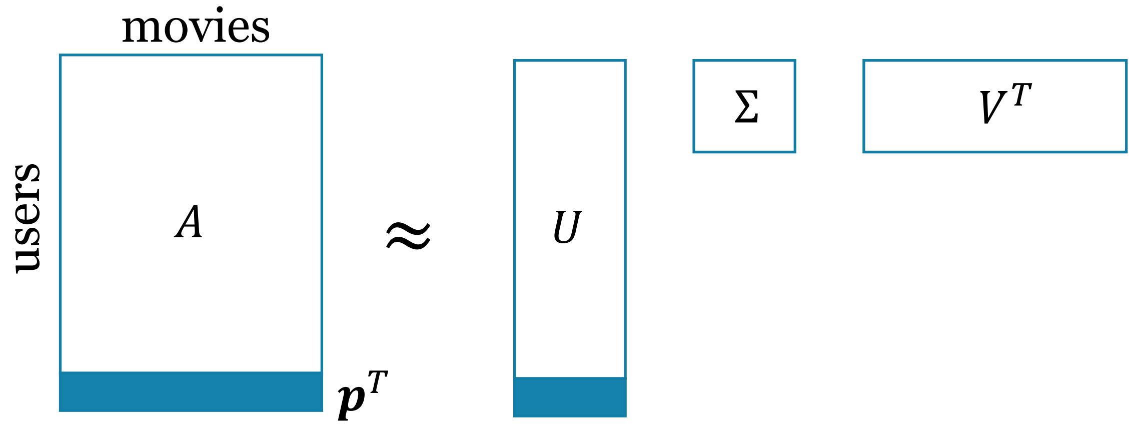


Traditional models unable to recommend relevant items for new user if given low rated examples only.

Explanation

Alice	2	5	3
Bob	4		5
Carol	2	5	
Tom	2	???	???

MF prediction



predicted scores (*folding-in*):

$$\boldsymbol{r} \approx VV^T\boldsymbol{p}$$

top-n recommendations task:

$$\operatorname{toprec}(\boldsymbol{p}, n) \coloneqq \arg\overset{n}{\max}\, \boldsymbol{r}$$

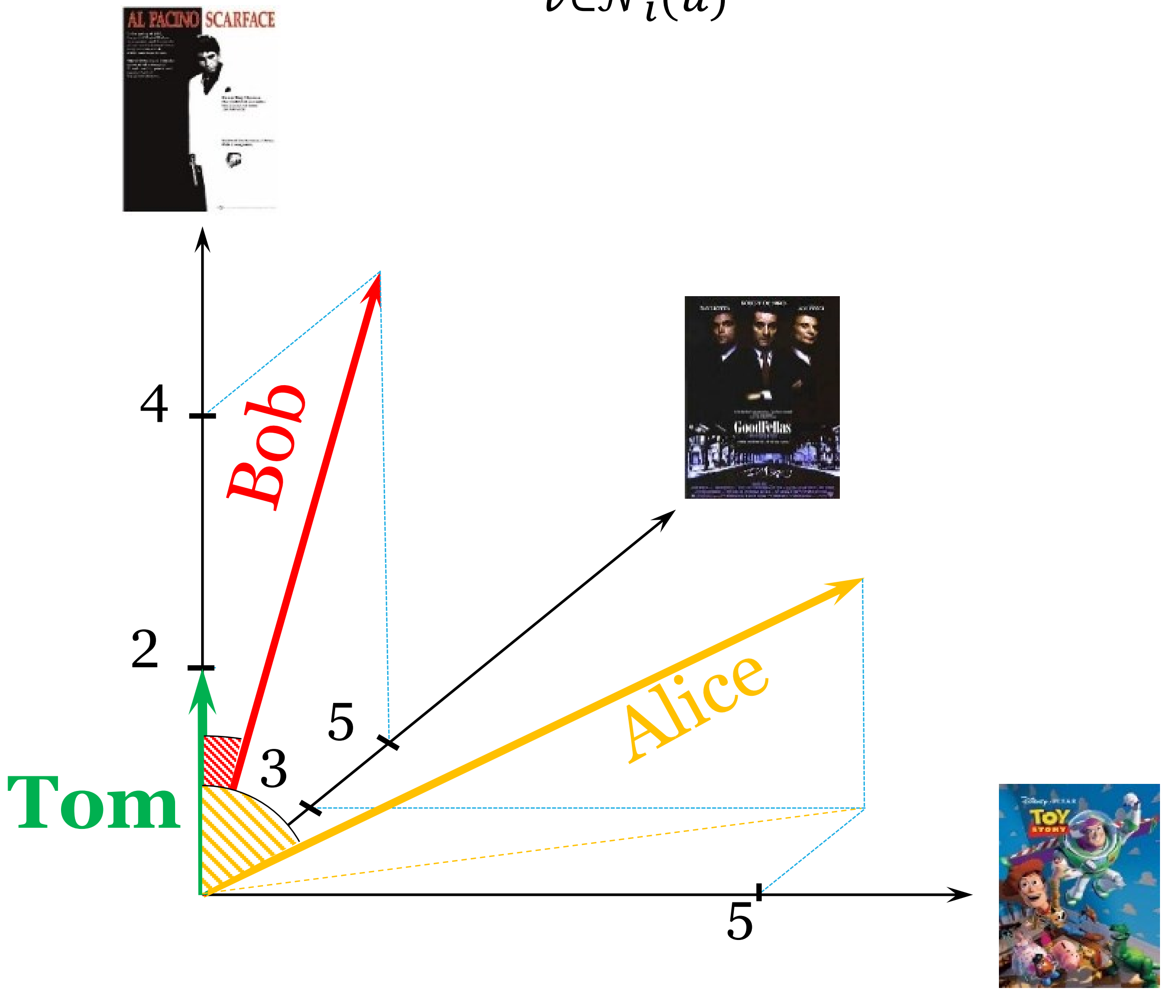
$$\arg\max VV^T(0, \ldots, 0, \underset{★★☆☆☆}{\mathbf{2}}, 0, \ldots, 0)^T \equiv \arg\max VV^T(0, \ldots, 0, \underset{★★★★★}{\mathbf{5}}, 0, \ldots, 0)^T$$

Rating value doesn't change ranking of the items!

Same problem for naïve similarity-based model

Alice	2	5	3
Bob	4		5
Carol	2	5	
Tom	2	???	???
		kNN prediction	
		2.6	**3.1**

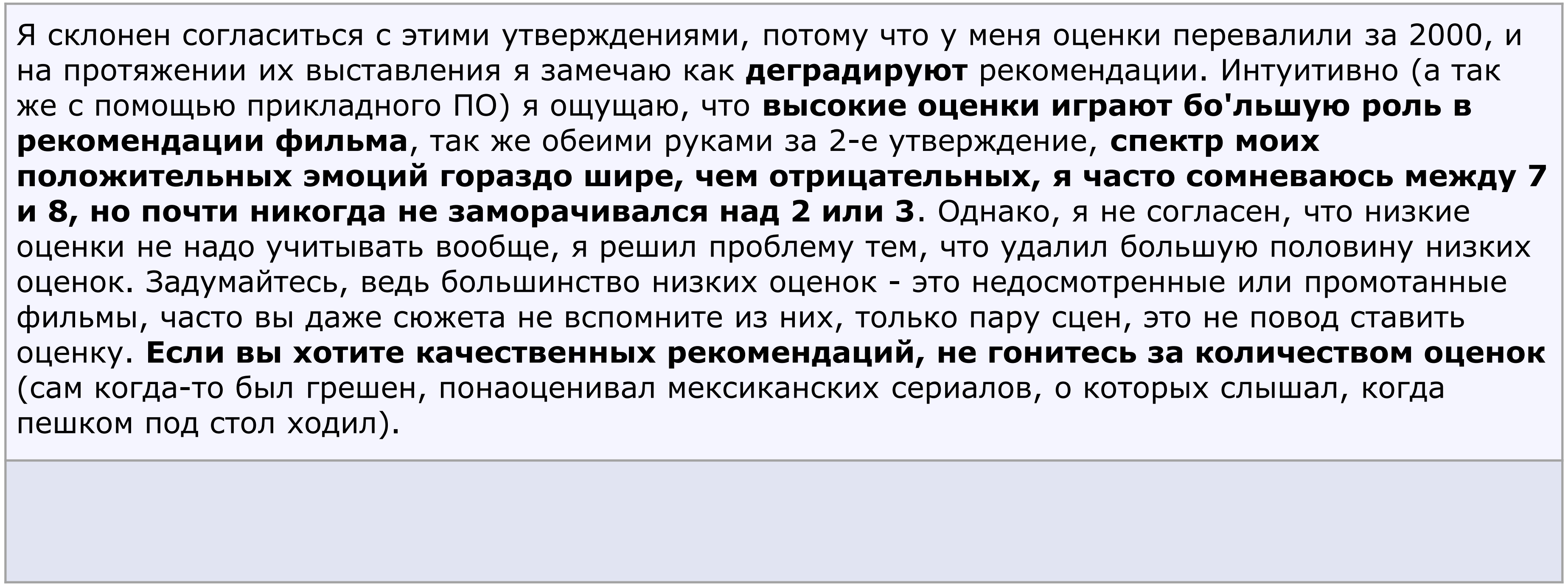
$$r_{ui} = \frac{1}{z} \cdot \sum_{v \in \mathcal{N}_i(u)} \cos(u, v) \cdot a_{vi}$$



Assumption:
users share not only what they like, but also what they dislike

Case study: (old) Kinopoisk

Some user’s “know-how” on improving the quality of recommendations.



Я склонен согласиться с этими утверждениями, потому что у меня оценки перевалили за 2000, и на протяжении их выставления я замечаю как **деградируют** рекомендации. Интуитивно (а так же с помощью прикладного ПО) я ощущаю, что **высокие оценки играют бо'льшую роль в рекомендации фильма**, так же обеими руками за 2-е утверждение, **спектр моих положительных эмоций гораздо шире, чем отрицательных, я часто сомневаюсь между 7 и 8, но почти никогда не заморачивался над 2 или 3**. Однако, я не согласен, что низкие оценки не надо учитывать вообще, я решил проблему тем, что удалил большую половину низких оценок. Задумайтесь, ведь большинство низких оценок - это недосмотренные или промотанные фильмы, часто вы даже сюжета не вспомните из них, только пару сцен, это не повод ставить оценку. **Если вы хотите качественных рекомендаций, не гонитесь за количеством оценок** (сам когда-то был грешен, понаоценивал мексиканских сериалов, о которых слышал, когда пешком под стол ходил).

https://forum.kinopoisk.ru/showthread.php?t=57341 (no longer exists)

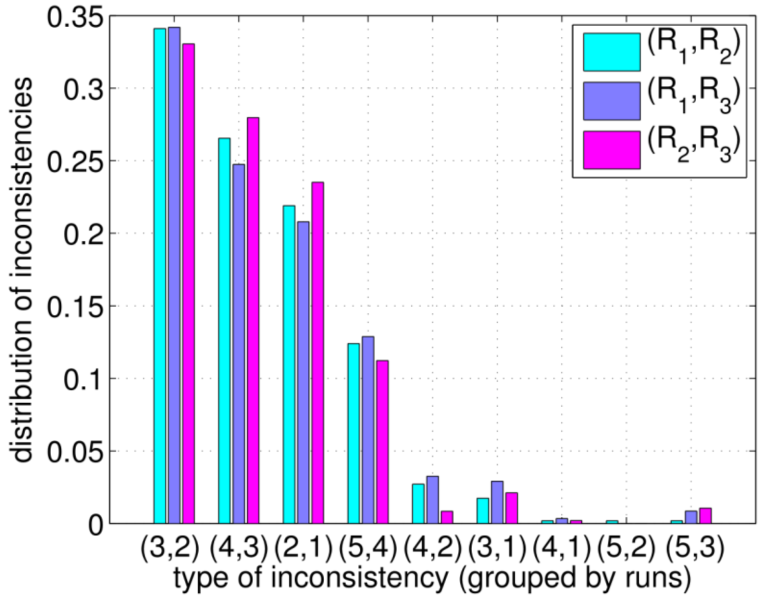
User feedback peculiarities

2.5x better?

Traditional recommender models treat ratings as **cardinal numbers**.

From neoclassical economics: utility is an **ordinal concept**.

Ratings scale consist of **unequal intervals***:

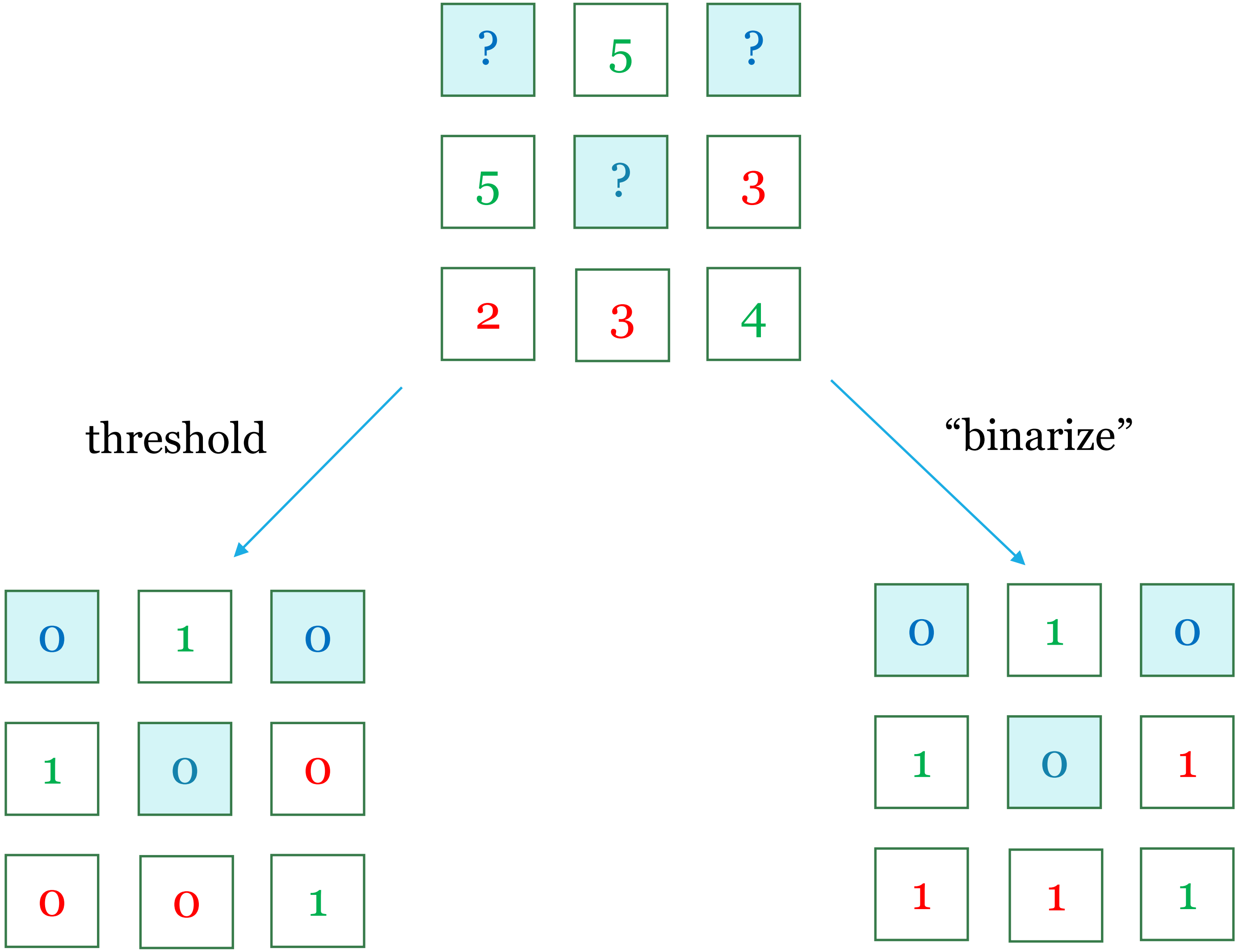


*"I like it... I like it not: Evaluating User Ratings Noise in Recommender Systems" by Xavier Amatriain, Josep M. Pujol, and Nuria Oliver

User feedback is a source for constructing a utility function

Implicit feedback	Explicit feedback
Easy to collect	Hard to collect
Lots of data	Less data
Intrinsic	Subjective
Hard to interpret	“Easy” to interpret

Explicit to implicit



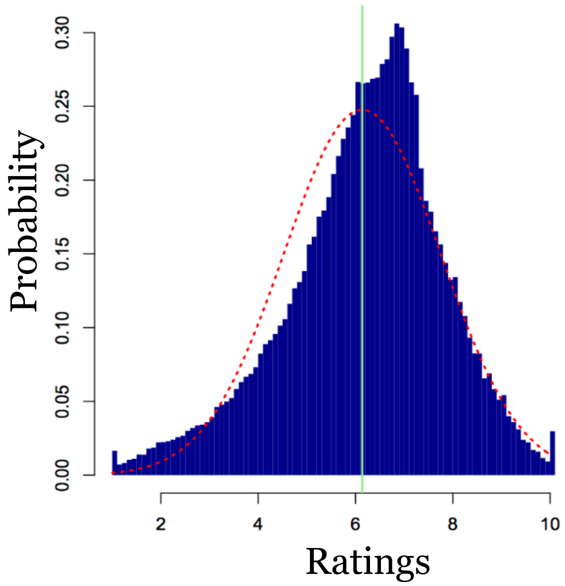
?	5	?
5	?	3
2	3	4

threshold

0	1	0
1	0	0
0	0	1

“binarize”

0	1	0
1	0	1
1	1	1



Which scheme to use depends on a kind of questions you want to answer.

Combining implicit and explicit feedback

Previously:

$$\mathcal{J}(P,Q) = \frac{1}{2}\sum_{i,j=1}^{M,N} w_{ij}\left(a_{ij} - \boldsymbol{p}_i^\top \boldsymbol{q}_j\right)^2 + \lambda\left(\|\boldsymbol{p}_i\|^2 + \|\boldsymbol{q}_j\|^2\right)$$

$$w_{ij} = \begin{cases} 1, & \text{if } a_{ij} \text{ is known,} \\ 0, & \text{otherwise.} \end{cases}$$

Recall,

- ratings are not what we want to predict, but
- ratings still indicate utility/relevance of an item;
- we generally want better predictions on higher ratings

What if $w_{ij} = f(a_{ij})$ would be a non-trivial function of feedback?

Combining implicit and explicit feedback

New formulation:

$$\mathcal{J}(P,Q) = \frac{1}{2}\sum_{i,j=1}^{M,N} w_{ij}\left(s_{ij} - \boldsymbol{p}_i^\top \boldsymbol{q}_j\right)^2 + \lambda\left(\|\boldsymbol{p}_i\|^2 + \left\|\boldsymbol{q}_j\right\|^2\right)$$

$$w_{ij} = f\left(a_{ij}\right), \qquad s_{ij} = \begin{cases} 1, & \text{if } a_{ij} \text{ is known,} \\ 0, & \text{otherwise.} \end{cases}$$

Predicting implicit feedback with explicit-feedback-based weighting

Confidence-based model (a.k.a iALS / WRMF)

$$\mathcal{J}(\Theta) = \mathcal{L}(\Theta) + \Omega(\Theta)$$

$$\mathcal{L}(\Theta) = \frac{1}{2}\left\| W \odot (S - PQ^\top) \right\|_\mathrm{F}^2, \qquad S\colon \begin{cases} s_{ij} = 1, & \text{if } a_{ij} \text{ is known,} \\ s_{ij} = 0, & \text{otherwise.} \end{cases}$$

Weights $W = \left[w_{ij}^{1/2}\right]$ are not binary anymore:

$$w_{ij} = \begin{cases} 1 + \alpha f(a_{ij}), & \text{if } a_{ij} \text{ is known,} \\ 1, & \text{otherwise.} \end{cases}$$

Examples of weighting function: $f(x) = x$ or $f(x) = \log\left(x + \frac{1}{\epsilon}\right)$

Complexity of iALS updates

$$\mathcal{J}(\Theta) = \frac{1}{2}\sum_i \|\boldsymbol{s}_i - Q\boldsymbol{p}_i\|^2_{W^{(i)}} + \frac{1}{2}\lambda\sum_i \|\boldsymbol{p}_i\|^2_2 + \frac{1}{2}\lambda\,\|Q\|^2_{\mathrm{F}}$$

$W^{(i)}$ is diagonal matrix of size $N \times N$ with elements $w_{i1} \dots w_{iN}$

Complexity of iALS updates

$$\mathcal{J}(\Theta) = \frac{1}{2}\sum_{i=1}^{M}\|\boldsymbol{s}_i - Q\boldsymbol{p}_i\|_{W^{(i)}}^2 + \frac{1}{2}\lambda\sum_{i=1}^{M}\|\boldsymbol{p}_i\|_2^2 + \frac{1}{2}\lambda\,\|Q\|_{\mathrm{F}}^2$$

$W^{(i)} = \mathrm{diag}\{w_{i1}, \dots, w_{iN}\}$ is $N \times N$ diagonal matrix

$$\boldsymbol{p}_i = (Q^\top W^{(i)} Q + \lambda I)^{-1} Q^\top W^{(i)} \boldsymbol{s}_i$$

Complexity of updates in naïve approach:

$$\mathrm{O}\left(MN \cdot d^2\right) + \mathrm{O}\left((M+N)d^3\right)$$

iALS performance

$$\boldsymbol{p}_i = (Q^\top W^{(i)} Q + \lambda I)^{-1} Q^\top W^{(i)} \boldsymbol{s}_i$$

$$w_{ij} = \begin{cases} 1 + c_{ij}(a_{ij}), & \text{if } a_{ij} \text{ is known,} \\ 1, & \text{otherwise.} \end{cases}, \qquad W^{(i)} = I + C^{(i)}$$

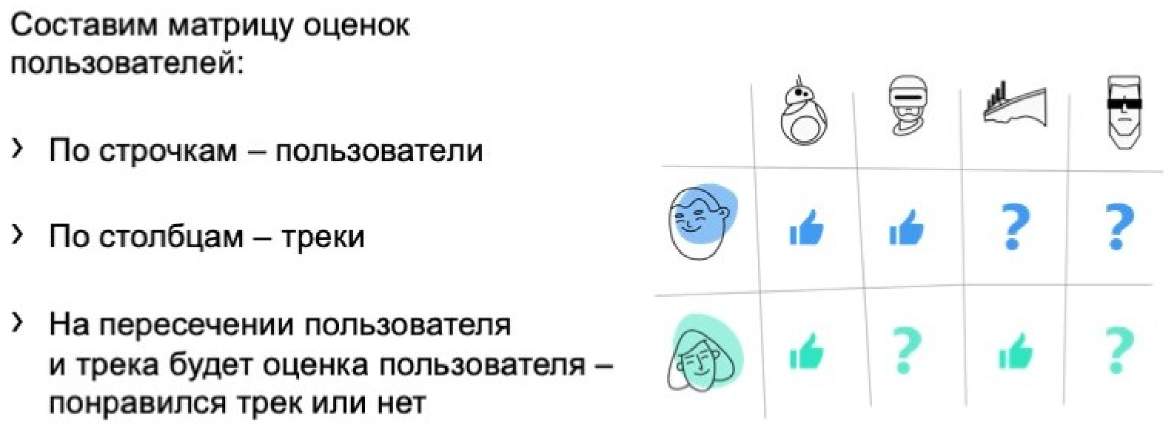
Trick: pre-calculate and store $d \times d$ matrix $Q^\top Q$

$$Q^\top W^{(i)} Q = Q^\top Q + Q^\top \left(C^{(i)} - I\right) Q$$

Overall complexity reduces to:

$$O\left(nnz_A \cdot d^2\right) + O\left((M + N) \cdot d^3\right)$$

Case study: Yandex Music



Составим матрицу оценок пользователей:

- По строчкам – пользователи
- По столбцам – треки
- На пересечении пользователя и трека будет оценка пользователя – понравился трек или нет

	👍	👍	?	?
	👍	?	👍	?

https://academy.yandex.ru/posts/kholodnye-polzovateli-i-mnogorukie-bandity

Need to take into account different signals! For example:

- number of seconds a user spent listening to a track
- number of skips / fast forwards
- adding to playlist / favorites
- …

Solved by iALS / WRMF model

$$\mathcal{L} = \sum_{ij} w(a_{ij}) \cdot \ l(s_{ij} - r_{ij})$$

Confidence for negative samples

- Standard confidence-based scenario:
  - higher weight on errors *over observed interactions*

- Inverted case:
  - higher weight on errors *over unseen interactions*
  - possible cases
    - missing interactions of active users are more likely to be negatives
    - not clicked popular item is a strong signal about user preferences
  - weighting functions[*]: $w_{ui} = \alpha \times |I_u|$ or $w_{ui} = \alpha(M - |U_i|)$

[*] Pan, Rong, Yunhong Zhou, Bin Cao, Nathan N. Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. "One-class collaborative filtering." In *2008 Eighth IEEE International Conference on Data Mining*, pp. 502-511. IEEE, 2008.

Note on negative sampling

- SGD:
  - negative sampling
- iALS:
  - confidence weights
- PureSVD
  - data normalization

All these methods aim to balance contribution of positive and negative items!

ALS vs SGD vs SVD

ALS

- More stable
- Fewer hyper-parameters to tune
- Higher complexity, however requires fewer iterations
- Embarrassingly parallel
- Higher communication cost in distributed environment

SGD

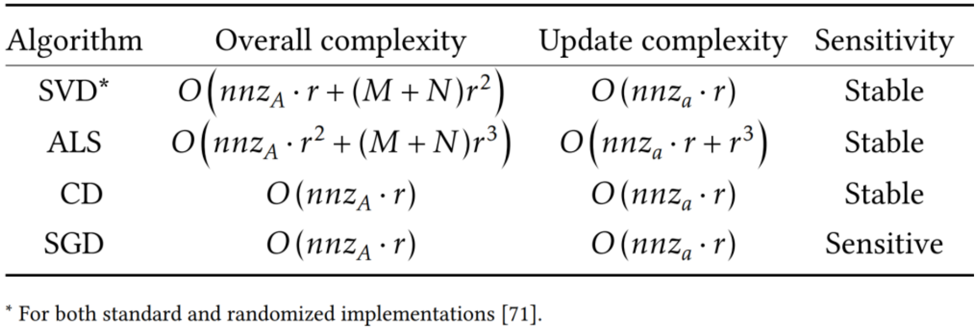
- Sensitive to hyper-parameters
- Requires special treatment of learning rate
- Lower complexity, but slower convergence
- Inherently sequential (parallelization is tricky for RecSys)

Unlike SVD:

More involved optimization (no rank truncation).
No global convergence.
Allow for custom optimization objectives. $\mathcal{L}(A, R) \rightarrow \mathcal{L}(f(A, R))$

Recap: scalability of MF algorithms

- PureSVD:
  - Randomized SVD for large-scale data
    - Criteo
    - Facebook’s RSVD
  - streaming and 0-communication versions of SVD
  - “exact” folding in ([Brand 2002])
- ALS:
  - embarrassingly parallel
  - but communication cost can be too high
  - Coordinate Descent can be a good alternative (e.g., eALS by [He et al. 2016])
- SGD:
  - inherently sequential
    - Hogwild! algorithm is not directly applicable in CF settings
  - there’re tricks to run in parallel within sub-blocks of rating matrix
  - slow convergence in general
    - using adaptive learning rate schedule (ADAM, Adagard, etc.) helps



Algorithm	Overall complexity	Update complexity	Sensitivity
SVD*	$O\left(nnz_A \cdot r + (M+N)r^2\right)$	$O(nnz_a \cdot r)$	Stable
ALS	$O\left(nnz_A \cdot r^2 + (M+N)r^3\right)$	$O\left(nnz_a \cdot r + r^3\right)$	Stable
CD	$O(nnz_A \cdot r)$	$O(nnz_a \cdot r)$	Stable
SGD	$O(nnz_A \cdot r)$	$O(nnz_a \cdot r)$	Sensitive

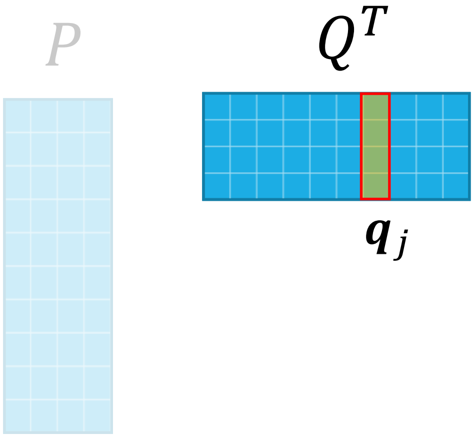
* For both standard and randomized implementations [71].

General rule of thumb: avoid distributed setups.

Special methods in recsys

Factorization meets neighborhood - NSVD

Key idea: user is represented as a combination of items



$$r_{ui} = \left(\sum_{j \in I_u} \boldsymbol{q}_j^\top \right) \boldsymbol{q}_i$$

omitting bias terms

items rated by user u
form a “neighborhood” I_u

- helpful in the case of extreme sparsity
- reduced storage requirements
- potentially less prone to overfitting

What is the corresponding matrix form?

Hint: you are given the matrix of implicit interactions S.

$$R =$$

A. Paterek, “*Improving regularized singular value decomposition for collaborative filtering*”, 2007.

SVD++

Key idea: user is described by implicit and explicit interactions

$$\boldsymbol{p}_u \leftarrow \boldsymbol{p}_u + \frac{1}{|I_u|}\sum_{j\in I_u}\bar{\boldsymbol{q}}_j$$

Explicit part

Implicit part

independent latent spaces!

$$r_{ui} = \left(\boldsymbol{p}_u + \frac{1}{|I_u|}\sum_{j\in I_u}\bar{\boldsymbol{q}}_j\right)^\top \boldsymbol{q}_i$$

omitting bias terms

$$\boxed{R = (P + S_\mathrm{n}\bar{Q})Q^\top}$$

binary

$$S_\mathrm{n} = D^{-1}S$$

$$D = \mathrm{diag}\{\|\boldsymbol{s}_1\|^2, \dots, \|\boldsymbol{s}_M\|^2\}$$

For several types of implicit feedback $P + S_\mathrm{n}^{(1)}\bar{Q}^{(1)} + S_\mathrm{n}^{(2)}\bar{Q}^{(2)} + \cdots$

Increases the number of parameters!

Do you see a common pattern is these models?

- I2I
- Item kNN
- NSVD
- PureSVD

A reoccurring pattern here – “user-free” approaches.
Let’s explore it further.

EIGENREC

- Our algorithms are correlational machines!

- For inter-item correlation matrix C we have:

$$C = A_0^\top A_0 = V\Sigma^2 V^\top, \qquad A_0 = U\Sigma V^\top$$

$$c_{ij} = \overline{\boldsymbol{a}}_i^\top \overline{\boldsymbol{a}}_j = \|\overline{\boldsymbol{a}}_i\| \|\overline{\boldsymbol{a}}_j\| \cdot \cos\phi_{ij}$$

- More generally, as a product of a scaling and similarity functions:

$$c_{ij} = \xi(i,j) \cdot \kappa(i,j)$$

Nikolakopoulos AN, Kalantzis V, Garofalakis JD. EIGENREC: An efficient and scalable latent factor family for top-N recommendation.

Generating new EIGENREC models

$$c_{ij} = \xi(i,j) \cdot \kappa(i,j)$$

- Options for scaling:

$$\xi(i,j) = \left(\|\bar{\boldsymbol{a}}_i\|\|\bar{\boldsymbol{a}}_j\|\right)^f$$

- Options for similarity $\kappa(i,j)$:
  - Cosine / Adjusted Cosine similarity
  - Pearson correlation
  - Jaccard index / Weighted Jaccard Index
  - …

EIGENREC recipe

- solve an eigendecomposition problem for the scaled item similarity/proximity matrix
- use leading eigenvectors for generating recommendations

$$\boldsymbol{r}_u^\top =$$

What about asymmetric / probabilistic similarity?

PureSVD normalization via EIGENREC

PureSVD is equivalent to an eigendecomposition problem for the scaled cosine similarity matrix

$$A_0^\top A_0 = DSD = V\Sigma^2 V^\top$$

$$s_{ij} = \frac{\bar{\boldsymbol{a}}_i^\top \bar{\boldsymbol{a}}_j}{d_i d_j}, \qquad D = \operatorname{diag}\{d_1, \dots, d_N\}, \qquad d_i = \|\bar{\boldsymbol{a}}_i\|$$

Normalization:

$$DSD \rightarrow \text{EIGENREC}\left(D^f\, S D^f\right) \text{ or } \text{PureSVD}(\qquad\qquad)$$

- What is the effect of f on the model behavior?
- Other options: user-based normalization or combined

Sparse LInear Methods: **SLIM**

We want our model's weights to be sparse and non-negative.

Why:

- computational / storage efficiency
- interpretability

The model:

$$R = AW$$

Task: suggest a trivial minimizer W to

$$\mathcal{L}(A, R) = ||A - R||_{\mathrm{F}}^2$$

Interpretability of SLIM

- User as a *non-trivial* weighted sum of items' representation:

$$\boldsymbol{r}_u^\top = \boldsymbol{a}_u^\top W$$

- How do we impose interpretability?

SLIM optimization task

$$\underset{W}{\text{minimize}} \quad \frac{1}{2}||A - AW||_{\mathrm{F}}^2 + \frac{\beta}{2}||W||_{\mathrm{F}}^2 + \lambda||W||_1$$

$$\text{s.t.} \quad \mathrm{W} \geq 0$$

$$\mathrm{diag(W)} = 0$$

Properties:

- positive item relations
- asymmetric

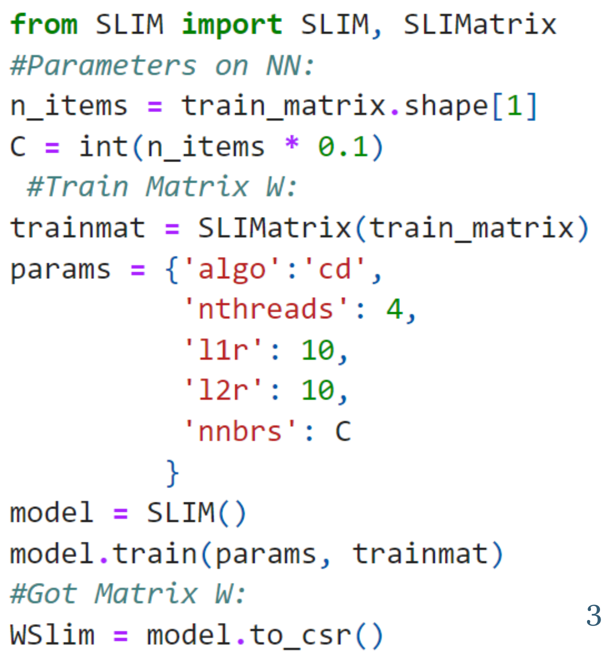
SLIM computation

$$\underset{\bar{\boldsymbol{w}}_i}{\text{minimize}} \quad \frac{1}{2}\|\bar{\boldsymbol{a}}_i - A\bar{\boldsymbol{w}}_i\|_2^2 + \frac{\beta}{2}\|\bar{\boldsymbol{w}}_i\|_2^2 + \lambda|\bar{\boldsymbol{w}}_i|_1$$

$$\text{subject to} \quad \bar{\boldsymbol{w}}_i > 0$$

$$w_{ii} = 0$$

- an elastic net problem[*]
  - can be solved with coordinate descent and soft thresholding
  - runs independently for each item (columns of A)



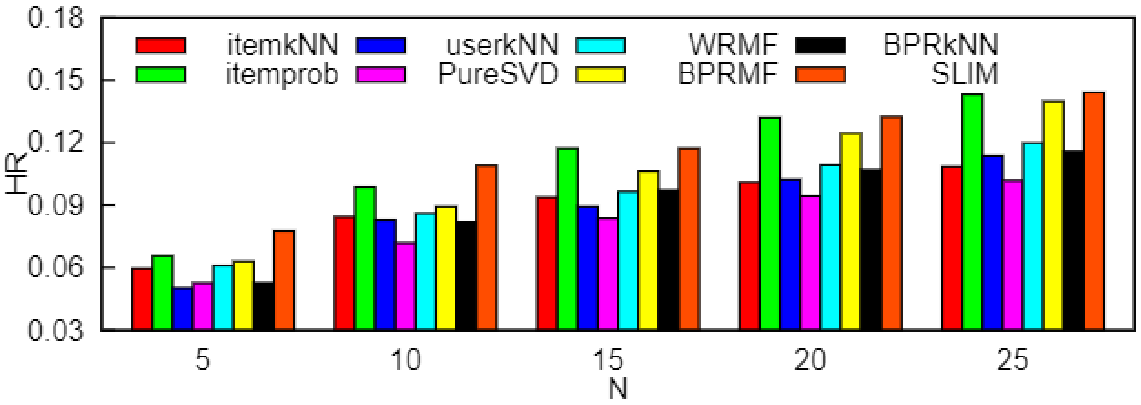
```
from SLIM import SLIM, SLIMatrix
#Parameters on NN:
n_items = train_matrix.shape[1]
C = int(n_items * 0.1)
 #Train Matrix W:
trainmat = SLIMatrix(train_matrix)
params = {'algo':'cd',
          'nthreads': 4,
          'l1r': 10,
          'l2r': 10,
          'nnbrs': C
         }
model = SLIM()
model.train(params, trainmat)
#Got Matrix W:
WSlim = model.to_csr()
```

[*] https://github.com/nikolakopoulos/SLIM
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html
https://github.com/MaurizioFD/RecSys2019_DeepLearning_Evaluation/tree/master/SLIM_ElasticNet

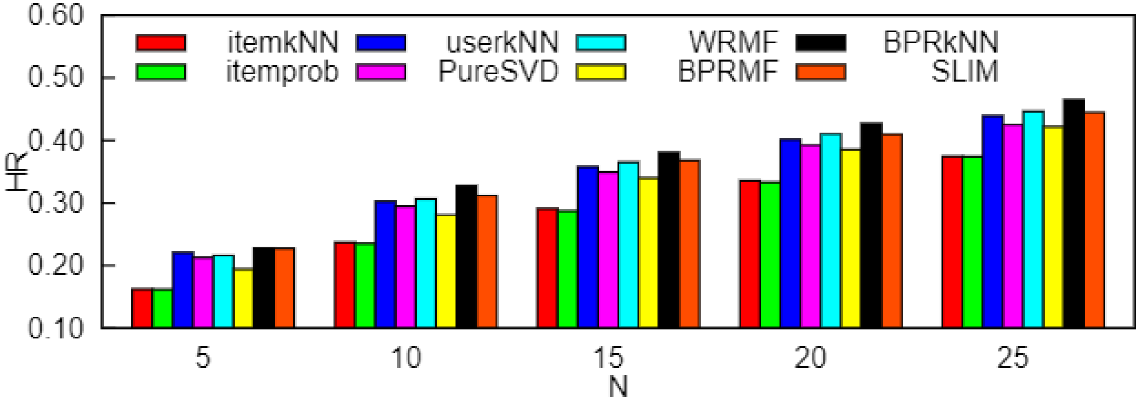
SLIM top-n recommendations

$$\boldsymbol{r}_u^\top = \boldsymbol{a}_u^\top W$$

$O(nnz_a \cdot nnz_w)$ complexity



(a) BX



(b) ML10M

Images from: Ning, Xia, and George Karypis. "Slim: Sparse linear methods for top-n recommender systems." In *2011 IEEE 11th international conference on data mining*, pp. 497-506. IEEE, 2011.

SLIM vs. other linear models

Unlike item-to-item, kNN:

- the notion of similarity/proximity is learned from data
- potentially more expressive item model

Unlike basic kNN, item-to-item, PureSVD, NSVD:

- asymmetric relations

Improving performance of SLIM

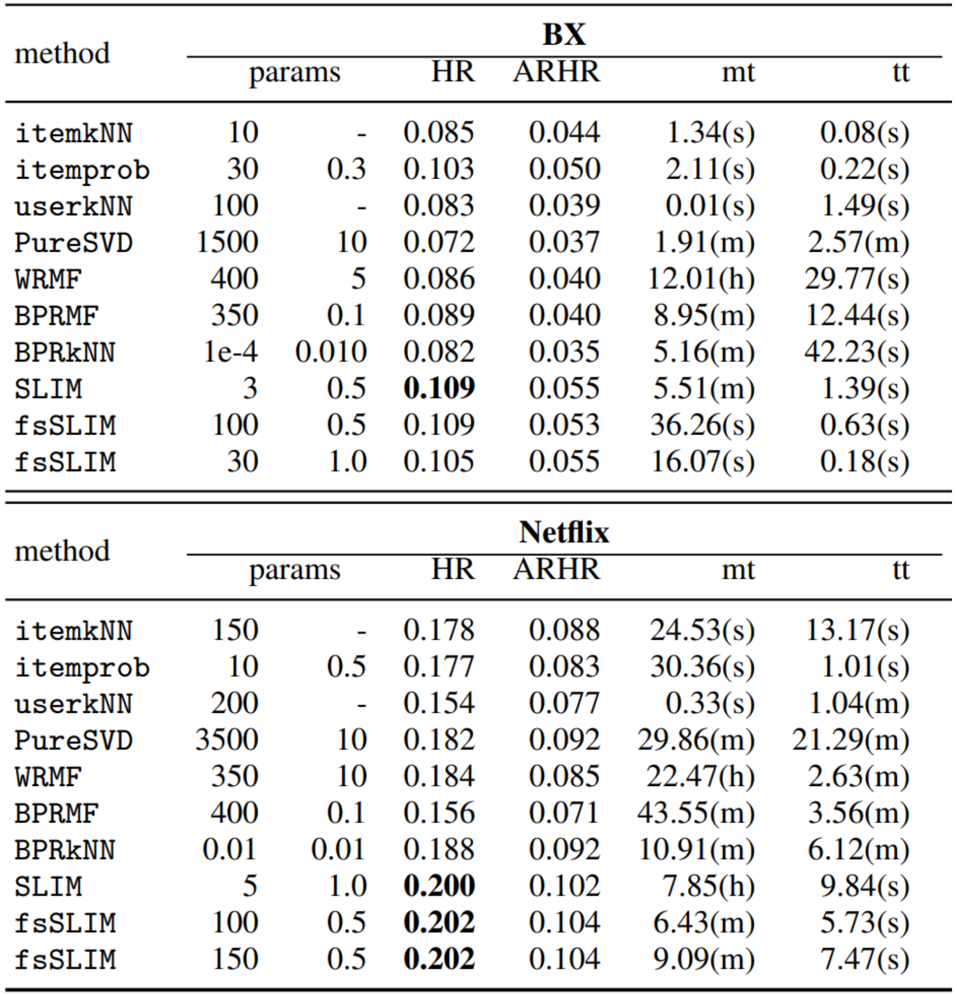
- each column depends on other columns (independent variables)
  - prone to spurious correlations
  - increased computation time

- Idea: reduce the space of variables prior to a regression step
- How can we achieve this?

Feature-selection SLIM

- fsSLIM
    - item-based kNN for selecting best matching independent variables in regression
    - simple cosine similarity works fine

- advantages:
    - significant reduction of training and recommendation generation time
    - almost no impact on quality of recommendations



method	BX					
	params		HR	ARHR	mt	tt
itemkNN	10	-	0.085	0.044	1.34(s)	0.08(s)
itemprob	30	0.3	0.103	0.050	2.11(s)	0.22(s)
userkNN	100	-	0.083	0.039	0.01(s)	1.49(s)
PureSVD	1500	10	0.072	0.037	1.91(m)	2.57(m)
WRMF	400	5	0.086	0.040	12.01(h)	29.77(s)
BPRMF	350	0.1	0.089	0.040	8.95(m)	12.44(s)
BPRkNN	1e-4	0.010	0.082	0.035	5.16(m)	42.23(s)
SLIM	3	0.5	**0.109**	0.055	5.51(m)	1.39(s)
fsSLIM	100	0.5	0.109	0.053	36.26(s)	0.63(s)
fsSLIM	30	1.0	0.105	0.055	16.07(s)	0.18(s)

method	Netflix					
	params		HR	ARHR	mt	tt
itemkNN	150	-	0.178	0.088	24.53(s)	13.17(s)
itemprob	10	0.5	0.177	0.083	30.36(s)	1.01(s)
userkNN	200	-	0.154	0.077	0.33(s)	1.04(m)
PureSVD	3500	10	0.182	0.092	29.86(m)	21.29(m)
WRMF	350	10	0.184	0.085	22.47(h)	2.63(m)
BPRMF	400	0.1	0.156	0.071	43.55(m)	3.56(m)
BPRkNN	0.01	0.01	0.188	0.092	10.91(m)	6.12(m)
SLIM	5	1.0	**0.200**	0.102	7.85(h)	9.84(s)
fsSLIM	100	0.5	**0.202**	0.104	6.43(m)	5.73(s)
fsSLIM	150	0.5	**0.202**	0.104	9.09(m)	7.47(s)

Table from: Ning, Xia, and George Karypis. "Slim: Sparse linear methods for top-n recommender systems." In *2011 IEEE 11th international conference on data mining*, pp. 497-506. IEEE, 2011.

EASE®

Idea: find a closed-form solution for SLIM.

How: omit l_1 regularization and non-negativity constraints.

$$\underset{W}{\text{minimize}} \quad \frac{1}{2}\|A - AW\|_F^2 + \frac{\lambda}{2}\|W\|_F^2, \qquad \text{s.t.} \quad \text{diag(W)} = 0$$

Solution:

$$W = I - P\,\text{diag}(\text{diag}(P)^{-1}), \qquad P = (A^\top A + \lambda I)^{-1}$$

$$w_{ij} = \begin{cases} 0, & \text{if } i = j, \\ -\dfrac{p_{ij}}{p_{jj}}, & \text{otherwise.} \end{cases}$$

Steck, Harald. "Embarrassingly shallow autoencoders for sparse data." In *The World Wide Web Conference*, pp. 3251-3257. 2019.