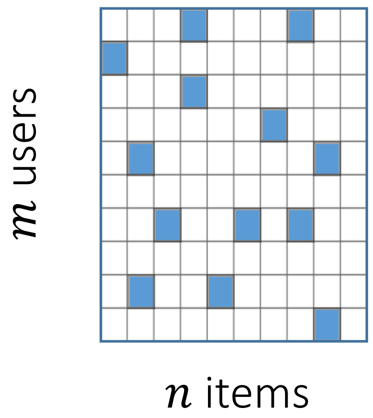
ACA ML 2018
Workshop on Recommender Systems

Part 3

Recap of the 2st part

A general view on matrix factorization

utility matrix A



Incomplete data:

- known entries
- unknown entries

Task: find utility (or relevance) function f_U such that:

$$f_R : \text{Users} \times \text{Items} \rightarrow \text{Relevance score}$$

As optimization problem with some *loss function* $\mathcal{L}$:

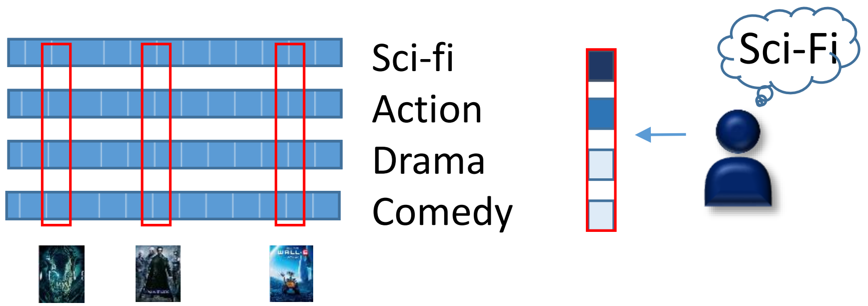
$$\boxed{\mathcal{L}(A, R) \rightarrow \min}$$

Any factorization model consists of:

- Utility function to generate R
- Optimization objective defined by $\mathcal{L}$
- Optimization method (algorithm)

$$R = PQ^T$$

Simplistic view: latent features ↔ genres



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)

For binary feedback complexity changes: $nnz \rightarrow MN$

Unlike SVD:

More involved model selection (no rank truncation).

No global convergence guarantees!

Asynchronous SGD is non-deterministic.

Allow for custom optimization objectives. $\mathcal{L}(A, R) \rightarrow \mathcal{L}(f(A, R))$

For explicit feedback:

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

* For both standard and randomized implementations [71].

Task

- You have a binary utility matrix (with “true” zeros) resulted from some implicit feedback information.
  - What will be the complexity of SGD?
  - What will be the SGD-based solution if you omit zero values?
  - Is it reasonable to use bias terms?

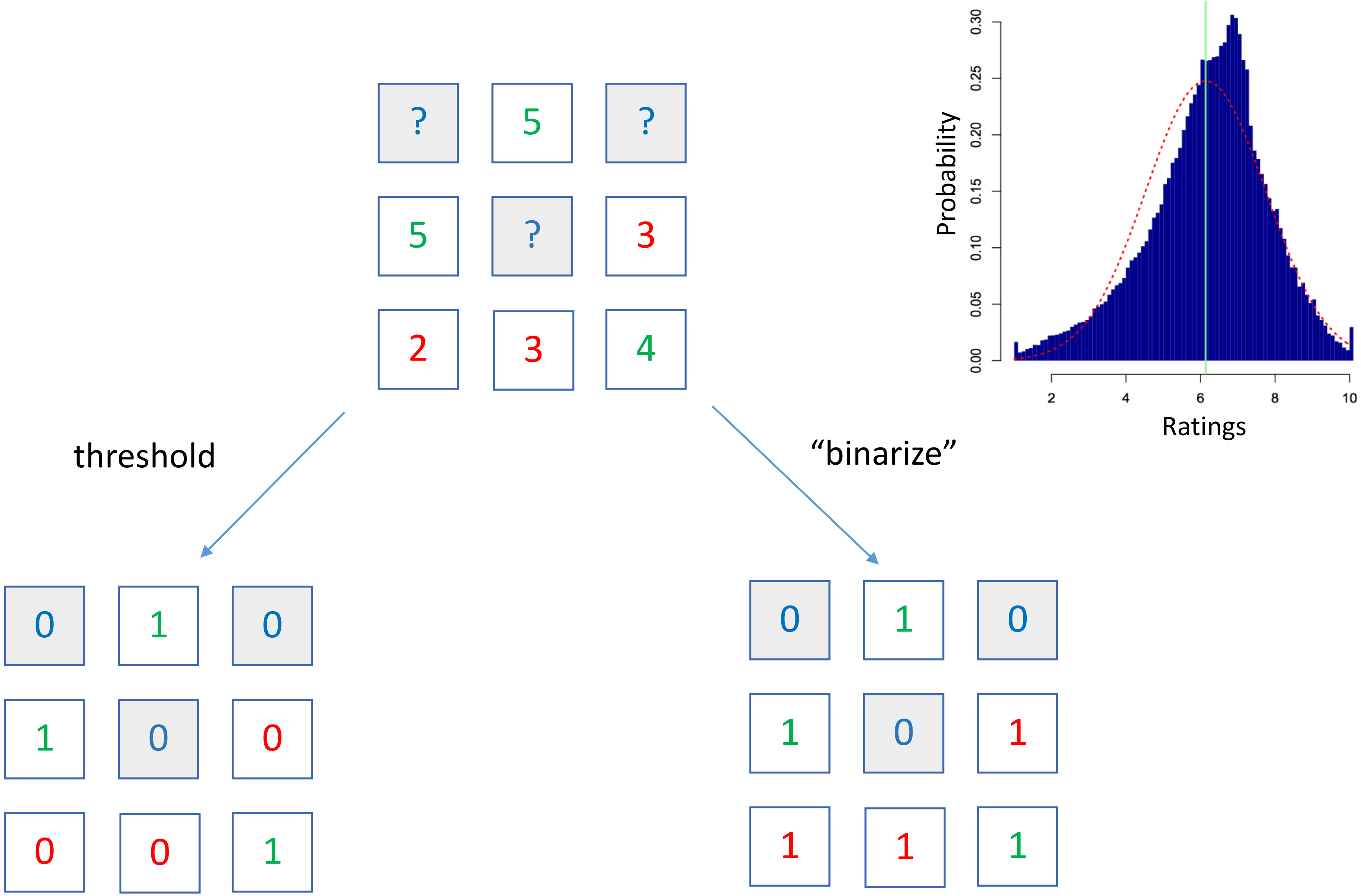
What are your solutions to 2nd home assignment?

Part 3

Today’s lecture:

- Advanced matrix factorization techniques
  - Mixed implicit and explicit feedback models (NSVD, SVD++)
  - Bilinear models: SVDFeature and Factorization Machines. Hybrid models.
- What can be optimized
- Tensor factorization

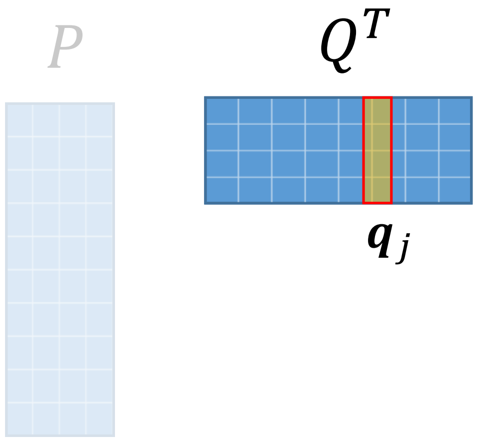
Explicit to implicit



NSVD

Key idea: user is represented as a combination of items

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



$$r_{ij} = \left(\sum_{k \in S^{(i)}} \boldsymbol{q}_k^T \right) \boldsymbol{q}_j$$

omitting bias terms

$$S^{(i)} = \{\, j{:}\ \ w_{ij} \neq 0\}$$

indices of items, rated by user i;
forms a "neighborhood" of items

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

What is the corresponding matrix form?

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

$$\boxed{R = SQQ^T}$$

Does it look familiar to you?

$$\boxed{\text{SVD: } R = A_0 V V^T}$$

SVD++

Key idea: user is described by implicit and explicit interactions

Y. Koren, "*Factorization meets the neighborhood: a multifaceted collaborative filtering model*", 2008.

independent latent spaces!

$$\boldsymbol{p}_{\mathrm{i}} \leftarrow \boldsymbol{p}_{\mathrm{i}} + \frac{1}{|S^{(i)}|} \sum_{k \in S^{(i)}} \bar{\boldsymbol{q}}_k \qquad\qquad r_{ij} = \left(\boldsymbol{p}_{\mathrm{i}} + \frac{1}{|S^{(i)}|} \sum_{k \in S^{(i)}} \bar{\boldsymbol{q}}_k \right)^T \boldsymbol{q}_j$$

Explicit part

Implicit part

omitting bias terms

binary

$$\boxed{R = (P + S_{\mathrm{n}} \bar{Q}) Q^T} \qquad S_{\mathrm{n}} = D^{-1} S$$

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

For several types of implicit feedback $\quad P + S_{\mathrm{n}}^{(1)} \bar{Q}^{(1)} + S_{\mathrm{n}}^{(2)} \bar{Q}^{(2)} + \cdots \quad$ Increases the number of parameters!

More general representation

$$R = (P + S_{\mathrm{n}}\bar{Q})Q^T = \overbrace{[I\ S_n]}^{X}\underbrace{\begin{bmatrix} P \\ \bar{Q} \end{bmatrix}}_{\bar{P}}$$

$$\boxed{R = (X\bar{P})Q^T}$$

X - is a “design” matrix

Matrix X encodes implicit information.
It can be extended to incorporate any side information.

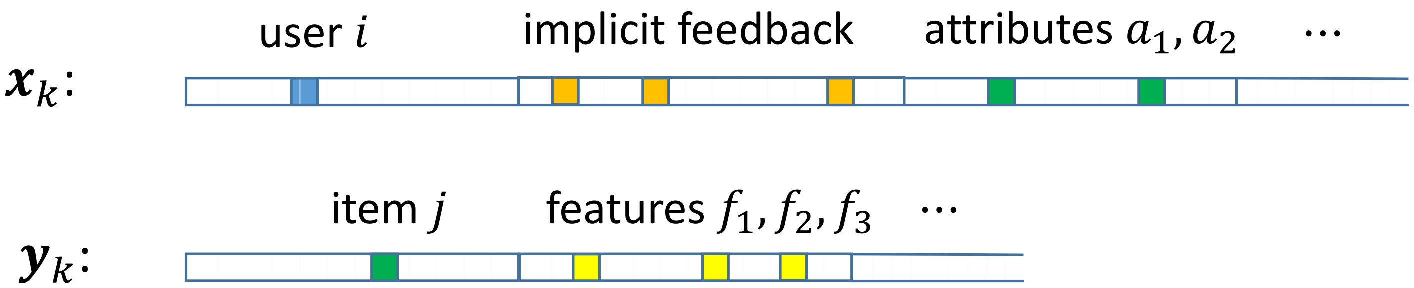
Row k of matrix X, $\boldsymbol{x}_k$:

SVDFeature

T. Chen, et al. “*Feature-based matrix factorization*”, 2011

$$\boxed{R = (XP)(YQ)^T}$$

$$X = [X_1\ X_2 \ldots X_m] \qquad Y = [Y_1\ Y_2 \ldots Y_n]$$



Including bias terms:

$$\boxed{r_{ij} = b_0 + \boldsymbol{t}^{\boldsymbol{T}}\boldsymbol{x}_i + \boldsymbol{f}^{\boldsymbol{T}}\boldsymbol{y}_j + \boldsymbol{x}_i^T PQ\boldsymbol{y}_j}$$

Optimized with ALS, SGD.

$b_0 = \sum_{g\in G}\gamma_g\mu_g$ is precomputed!

Model parameters: $\Theta = \{\boldsymbol{t}, \boldsymbol{f}, P, Q\}$

Factorization Machines

Idea: polynomial expansion

S. Rendle, "*Factorization machines*", 2010.

$$f(\boldsymbol{x}) = b_0 + \boldsymbol{b}^T\boldsymbol{x} + \boldsymbol{x}^T H\boldsymbol{x} + \cdots$$

Feature vector **x** / Target y

	User				Movie					Other Movies rated					Time	Last Movie rated					Target y	
	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...	Time	TI	NH	SW	ST	...		
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	13	0	0	0	0	...	5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...	1	$y^{(2)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...	4	$y^{(3)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...	5	$y^{(4)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...	1	$y^{(5)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...	5	$y^{(6)}$

Factorization Machines

$$r_{ij} = b_0 + \boldsymbol{t}^T \boldsymbol{x}_i + \boldsymbol{f}^T \boldsymbol{y}_j + \boldsymbol{x}_i^T PQ \boldsymbol{y}_j$$

$$\boldsymbol{b} = \begin{bmatrix} \boldsymbol{t} \\ \boldsymbol{f} \end{bmatrix} \quad \boldsymbol{z} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}$$ for some fixed i and j

$$r(\boldsymbol{z}) = b_0 + \boldsymbol{b}^T \boldsymbol{z} + \boldsymbol{z}^T H \boldsymbol{z} + \cdots$$

characterizes intra- and inter-relations between any encoded entities

Data is sparse → impose low-rank structure on H

H is symmetric positive semi-definite

$$H = VV^T$$

V embeds all users, items and their side information

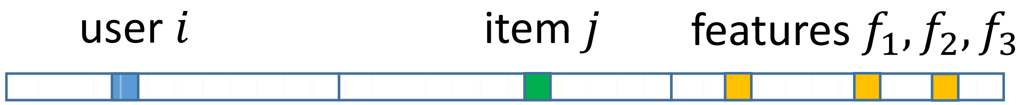
2nd order FM:

$$\boxed{r(\boldsymbol{z}) = b_0 + \sum_k b_k z_k + \sum_k \sum_{k'=k+1} \langle \boldsymbol{v}_k, \boldsymbol{v}_{k'} \rangle z_k z_{k'}}$$

$\langle \cdot, \cdot \rangle$ is a scalar product

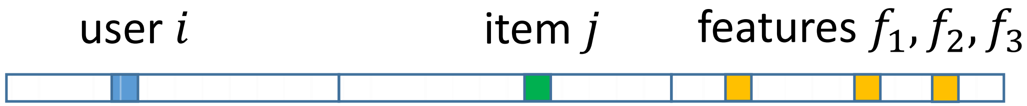
Model parameters: $\Theta = \{b_0, \boldsymbol{z}, V\}$

$\boldsymbol{z}_k$:



Example of Factorization Machines computation

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{f} \end{bmatrix} \qquad V = \begin{bmatrix} V_x \\ V_y \\ V_f \end{bmatrix} \qquad V \in \mathbb{R}^{n \times r} \qquad \boldsymbol{z} \in \mathbb{R}^n:$$



$$\boldsymbol{z}^T V V^T \boldsymbol{z} = \left(\begin{bmatrix} \boldsymbol{x}^T & \boldsymbol{y}^T & \boldsymbol{f}^T \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_f \end{bmatrix} \right) \left(\begin{bmatrix} V_x^T & V_y^T & V_f^T \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{f} \end{bmatrix} \right) = \left(\boldsymbol{x}^T V_x + \boldsymbol{y}^T V_y + \boldsymbol{f}^T V_f\right)\left(\boldsymbol{x}^T V_x + \boldsymbol{y}^T V_y + \boldsymbol{f}^T V_f\right)^T =$$

$$= \overbrace{\boldsymbol{x}^T V_x V_x^T \boldsymbol{x} + \boldsymbol{y}^T V_y V_y^T \boldsymbol{y} + \boldsymbol{f}^T V_f V_f^T \boldsymbol{f}}^{\text{“self-interaction” terms}} + \overbrace{2\big(\underbrace{\boldsymbol{x}^T V_x V_y^T \boldsymbol{y}}_{\text{user-item}} + \underbrace{\boldsymbol{x}^T V_x V_f^T \boldsymbol{f}}_{\text{user-feature}} + \underbrace{\boldsymbol{y}^T V_y V_f^T \boldsymbol{f}}_{\text{item-feature}}\big)}^{\text{actual 2}^{\text{nd}}\text{ order FM model } r(\boldsymbol{z}) \text{ (w/o biases)}}$$

interactions

$$\boldsymbol{v}_x = V_x^T \boldsymbol{x},\ \boldsymbol{v}_y = \dots \qquad \boxed{r(\boldsymbol{z}) = \frac{1}{2}\left[\left(\boldsymbol{v}_x + \boldsymbol{v}_y + \boldsymbol{v}_f\right)^T\left(\boldsymbol{v}_x + \boldsymbol{v}_y + \boldsymbol{v}_f\right) - \left(\|\boldsymbol{v}_x\|^2 + \|\boldsymbol{v}_y\|^2 + \|\boldsymbol{v}_f\|^2\right)\right] =}$$

$$= \frac{1}{2}\sum_{l=1}^{r}\left(\left(\sum_{k=1}^{n} v_{kl} z_k\right)^2 - \sum_{k=1}^{n}\left(v_{kl} z_k\right)^2\right)$$

reduces the number of operations

Connection of FM to other models

$$r(\boldsymbol{z}) = b_0 + \sum_k b_k z_k + \sum_k \sum_{k'=k+1} \delta_{kk'} \langle \boldsymbol{v}_k, \boldsymbol{v}_{k'} \rangle z_k z_{k'}$$

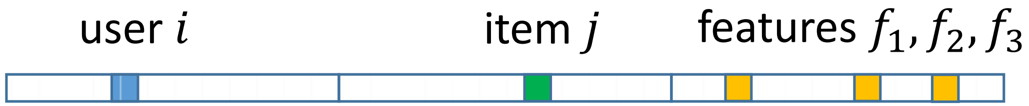
binary indicator variable, denotes "allowed" interactions

Example:
- restrict interactions to (user, item) pairs only
- forbid intra-relations between entities of the same type
- remove side information

What is the resulting model?

Matrix form of FM

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{f} \end{bmatrix} \qquad V = \begin{bmatrix} V_x \\ V_y \\ V_f \end{bmatrix} \qquad V \in \mathbb{R}^{n\times r} \qquad \boldsymbol{z} \in \mathbb{R}^{n}:$$



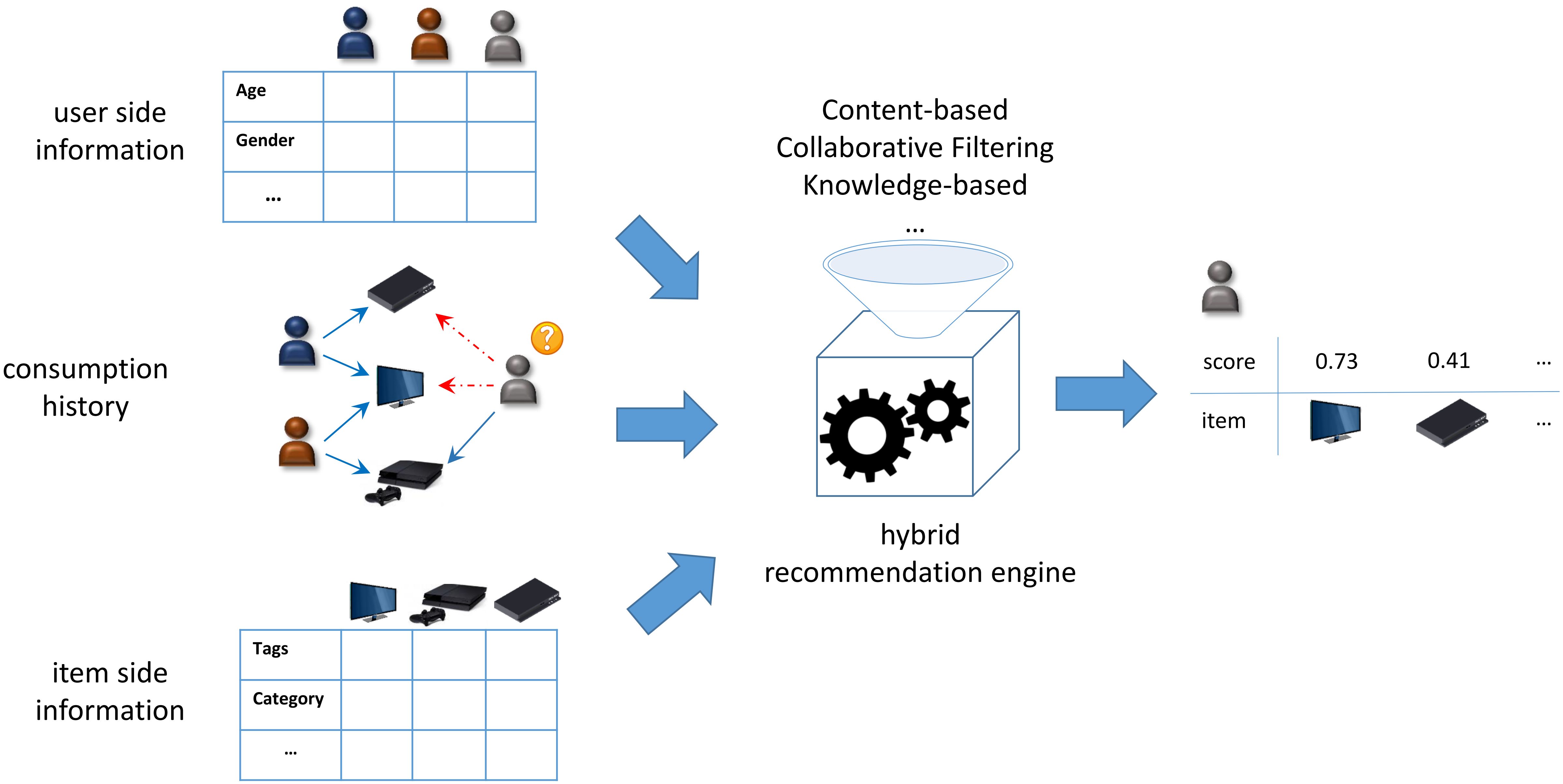
$$r(\boldsymbol{z}) = \frac{1}{2}\boldsymbol{z}^T \left(\begin{bmatrix} V_x \\ V_y \\ V_f \end{bmatrix} \begin{bmatrix} V_x^T & V_y^T & V_f^T \end{bmatrix} - \begin{bmatrix} V_x & & 0 \\ & V_y & \\ 0 & & V_f \end{bmatrix} \begin{bmatrix} V_x & & 0 \\ & V_y & \\ 0 & & V_f \end{bmatrix}^T \right) \boldsymbol{z} =$$

$$= \frac{1}{2}\boldsymbol{z}^T \begin{bmatrix} 0 & V_x V_y^T & V_x V_f^T \\ V_y V_x^T & 0 & V_y V_f^T \\ V_f V_x^T & V_f V_y^T & 0 \end{bmatrix} \boldsymbol{z} = \boldsymbol{z}^T H \boldsymbol{z}$$

$V_x \rightarrow P, V_y \rightarrow Q$ – conventional matrix factorization

$$H = \frac{1}{2} \begin{bmatrix} 0 & V_x V_y^T & V_x V_f^T \\ V_y V_x^T & 0 & V_y V_f^T \\ V_f V_x^T & V_f V_y^T & 0 \end{bmatrix}$$

Hybrid approach



Taxonomy

- **Ensemble design**

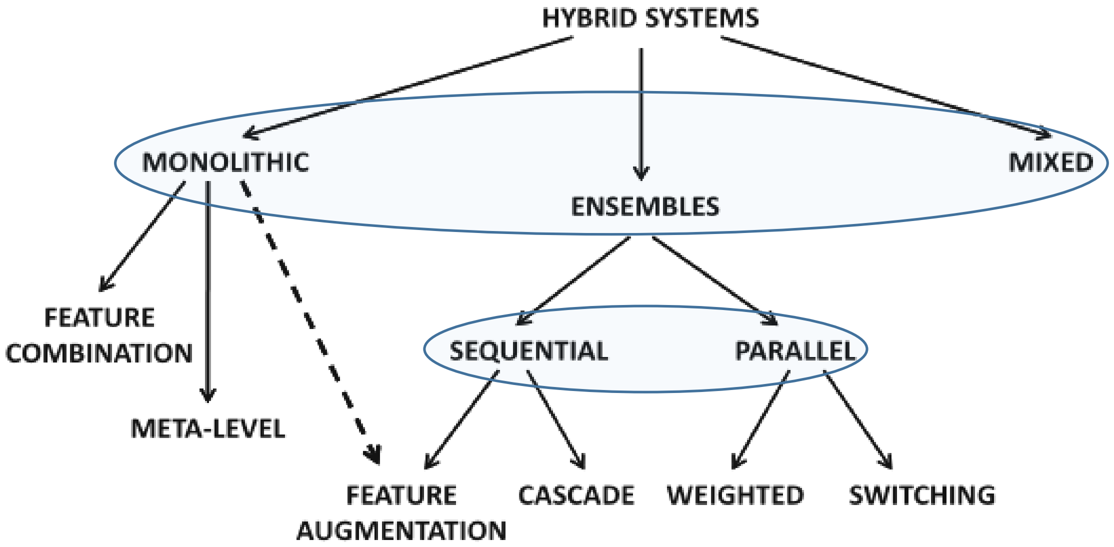
  combines several “black-box” algorithms to produce a single more robust output

- **Monolithic design**

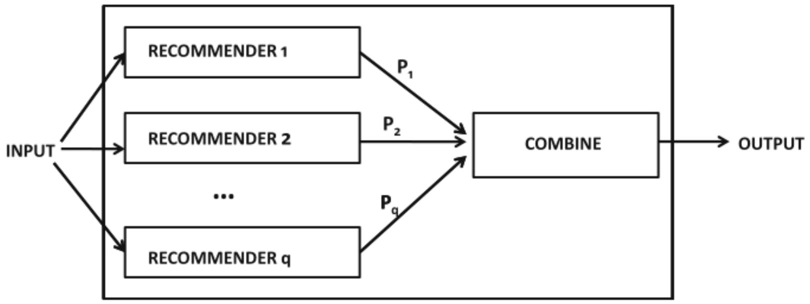
  obtained by integration of heterogeneous data sources and customization of models

- **Mixed systems**

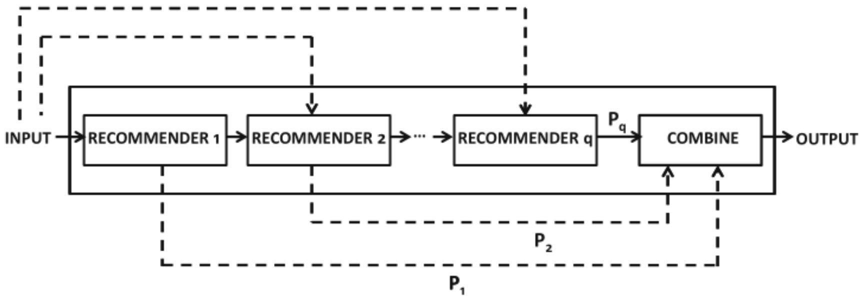
  present outputs of several recommender models simultaneously



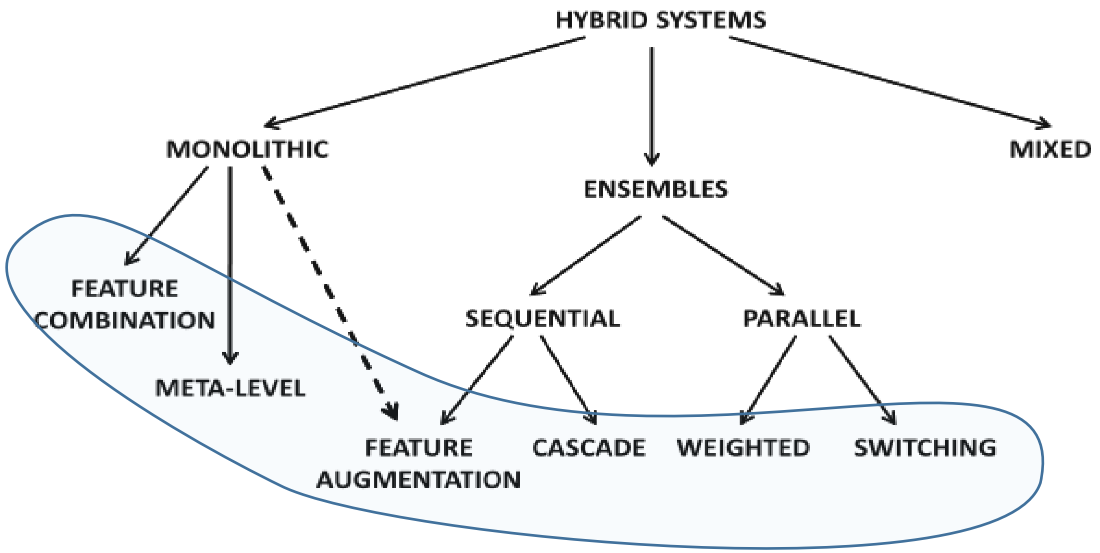
Parallel



Sequential



Taxonomy



Ensembles

- **Weighted**
  combines outputs of several models with varying degree of importance
- **Switching**
  situational choice of the most suitable recommendation model
- **Cascade**
  every consequent model tries to improve upon the results of the previous models
- **Feature Augmentation**
  one model generates new features or scores and feeds them into another model (stacking)

can be used "off-the-shelf"

Monolithic

- **Feature Combination**
  several data sources are fed into a single model
- **Meta-level**
  several models are glued together

no "off-the-shelf" solution

Task

Give an example of a feature combination approach

Meta-level system example

Rating matrix

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}
U_1	5	3	5	4	-	1	-	3	-	5	-	-
U_2	3	-	-	-	4	5	1	-	5	-	-	1
U_3	1	-	5	4	5	-	5	-	-	3	5	-

(a)

Test user

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}
U_4	5	-	1	-	-	4	-	-	3	-	-	5

(b)

Item features

	f_1	f_2	f_3	f_4
I_1	1	1	0	0
I_2	1	0	0	0
I_3	1	0	1	1
I_4	1	0	0	1
I_5	0	1	1	0
I_6	0	1	0	0
I_7	0	0	1	1
I_8	0	0	0	1
I_9	0	1	1	0
I_{10}	0	0	0	1
I_{11}	0	0	1	1
I_{12}	0	1	0	0

(c)

User profiles

	f_1	f_2	f_3	f_4
U_1	4	1	1	4
U_2	1	4	2	0
U_3	2	1	4	5

	f_1	f_2	f_3	f_4
U_4	1	4	1	0

Model: SVD over profiles

Prediction: folding-in + neighborhood

Symeonidis, P. "Content-based dimensionality reduction for recommender systems." *Data Analysis, Machine Learning and Applications* (2008): 619-626.

Some weighted hybrids

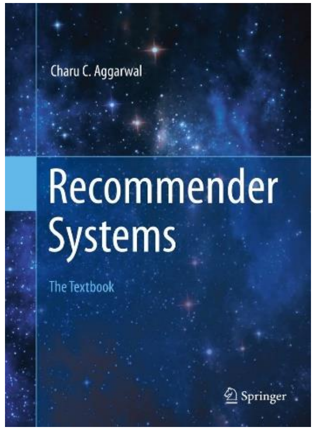
- Randomness Injection

  Many matrix factorization methods are inherently randomized due to initialization

Single data , several models →
single model, several data samples

- Bagging
  - Row-wise bootstrapping
  - Entry-wise bagging
- Subsampling
  - Row-wise subsampling
  - Entry-wise subsampling

Not storage-efficient



Read more:
Charu C. Aggarwal. “Recommender Systems. The Textbook”, 2016.
R. Burke. Hybrid recommender systems: Survey and experiments. User Modeling and User-adapted Interaction, 12(4), pp. 331–370, 2002.

Other factorization methods

- NMF
- MMF
- PMF
- CAMF
- SLIM
- FISM
- CliMF
- OrdRec
- CobaFi
- …

Remark on optimization objectives

basic MF

Pointwise loss: $L(A, \hat{A}) = \sum_{ij} l(a_{ij} - \hat{a}_{ij})$

$l(a_{ij} - \hat{a}_{ij}) = (a_{ij} - \boldsymbol{p}_i^T \boldsymbol{q}_j)^2$

minimizes RMSE

not the same functions!

"Learning to rank" approach

Pairwise loss: $L(A, \hat{A}) = \sum_{i} \sum_{\underbrace{j,j':a_{ij}>a_{ij'}}} l(\hat{a}_{ij} - \hat{a}_{ij'})$

better suited for ranking but harder to optimize

Listwise loss: $L(A, \hat{A})$: $l(\{a_{ij}\}, \{\hat{a}_{ij}\})$

Read more in: "Gradient descent optimization of smoothed information retrieval metrics", O. Chapelle, M. Wu, 2010

Bayesian personalized ranking (BPR)

Key idea – 2 step: 1. get predicted ratings (from any model)
2. try to improve ranking with a posteriori modification (max likelihood)

$$p(\Theta| >_u) \propto p(>_u |\Theta)\, p(\Theta)$$

$$p(i >_u j|\Theta) := \sigma(\hat{x}_{uij}(\Theta))$$

$$\hat{x}_{uij} := \hat{x}_{ui} - \hat{x}_{uj}$$

x_{ui}, x_{uj} are from some Collaborative Filtering model

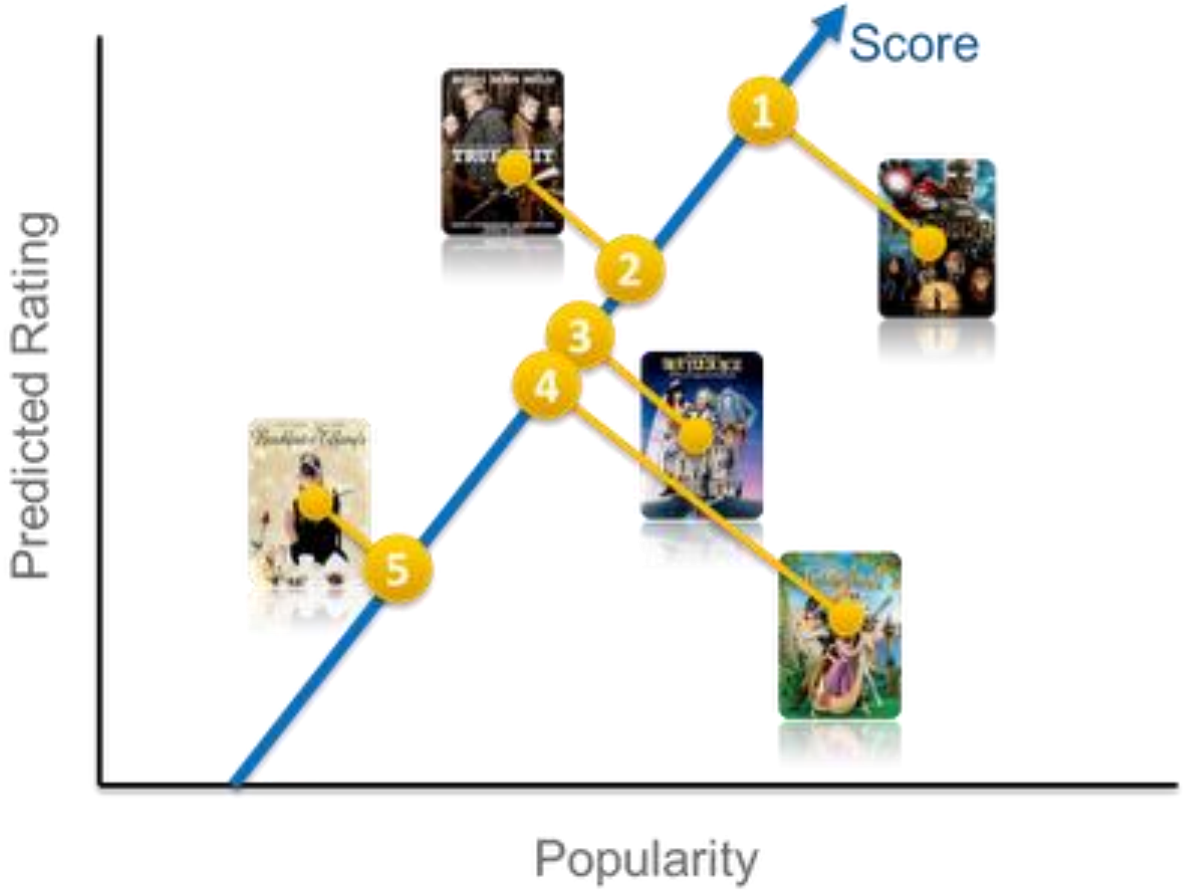
<u>pair-wise objective:</u>

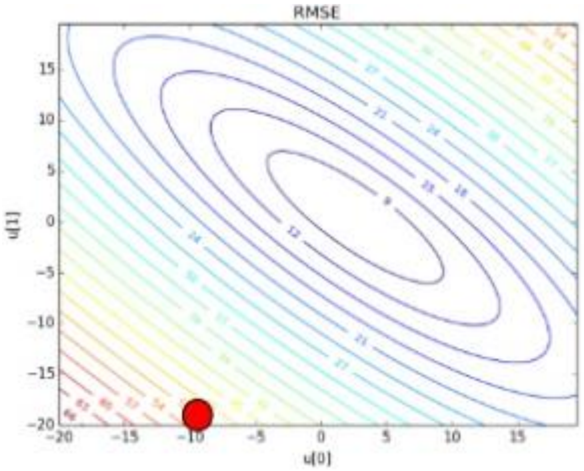
$$\sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2$$

Optimization with stochastic gradient descent.

“BPR: Bayesian Personalized Ranking from Implicit Feedback”, S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme

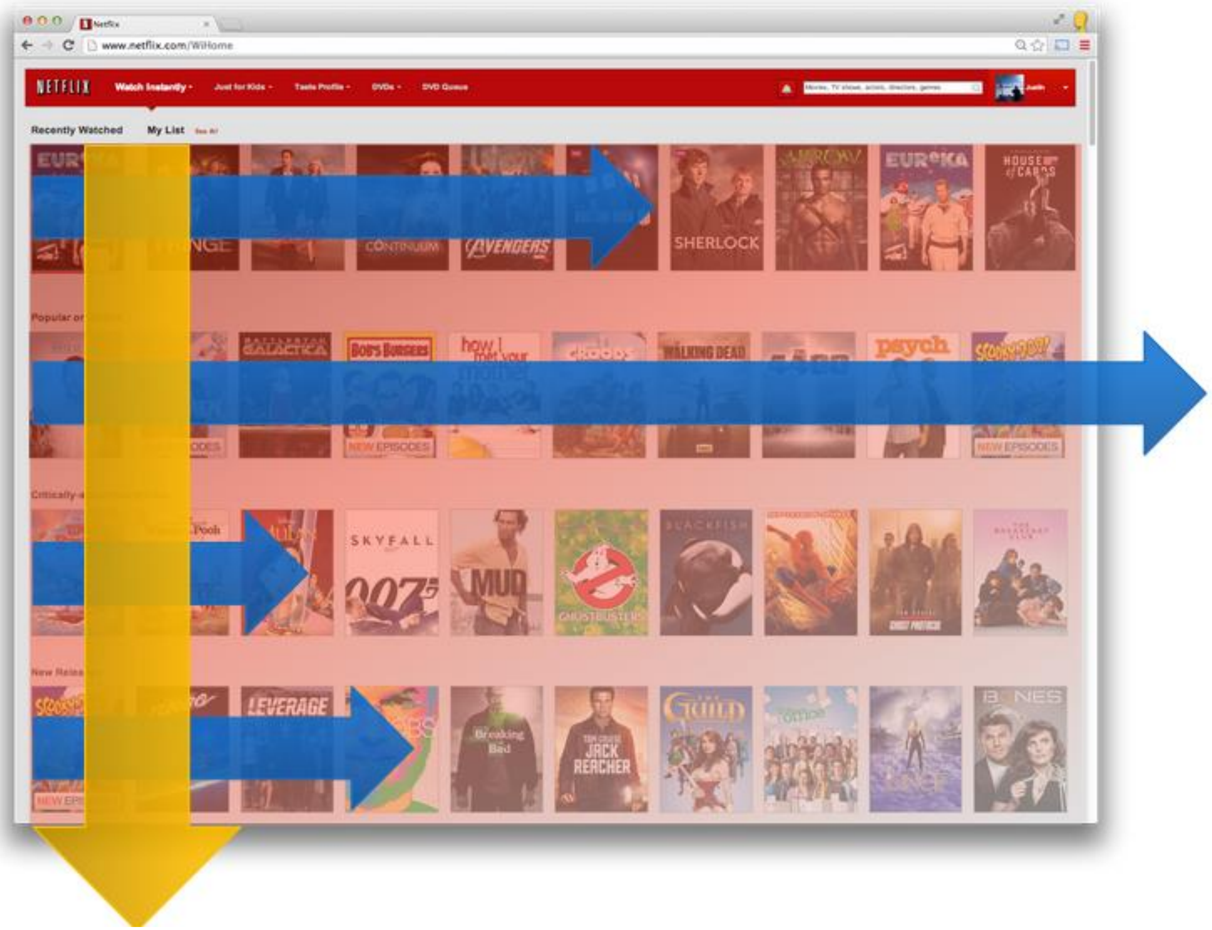
Learning to rank – simple approach





Source: http://technocalifornia.blogspot.ru/2013/07/recommendations-as-personalized.html#!/2013/07/recommendations-as-personalized.html

User Experience



Reading: How important is the user interface for a recommender system?
https://www.quora.com/How-important-is-the-user-interface-for-a-recommender-system/answer/Abhinav-Sharma?srid=cgo&share=e3430560

Image source: http://technocalifornia.blogspot.ru/2013/07/recommendations-as-personalized.html#!/2014/12/ten-lessons-learned-from-building-real.html

Lessons learned from building real-life recommender systems

http://www.slideshare.net/xamat/recsys-2016-tutorial-lessons-learned-from-building-reallife-recommender-systems

Tensor factorization

Guess standard algorithm behavior

What is likely to be recommended in this case?

Explanation

predicted scores (*folding-in*):

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

top-$\boldsymbol{n}$ recommendations task:

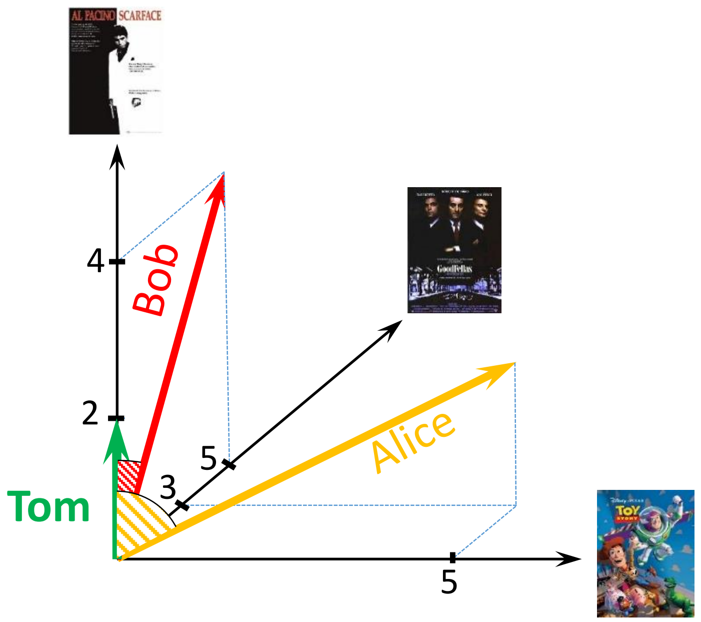
$$\text{toprec}(\boldsymbol{p}, n) \coloneqq \arg\overset{n}{\max}\,\boldsymbol{r}$$

$$\arg\max VV^T(0, \dots, 0, \color{red}{\mathbf{2}}, 0, \dots, 0)^T \equiv \arg\max VV^T(0, \dots, 0, \color{green}{\mathbf{5}}, 0, \dots, 0)^T$$

★★☆☆☆ ★★★★★

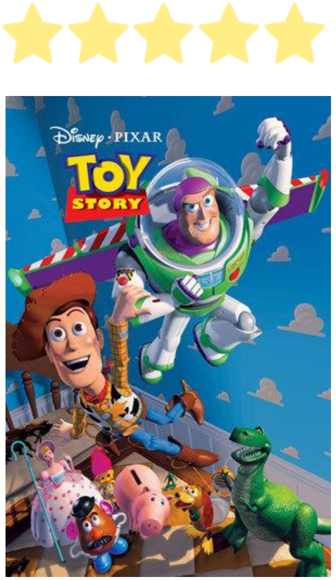
Rating value doesn't change ranking of the items!

Same for naïve similarity-based model:

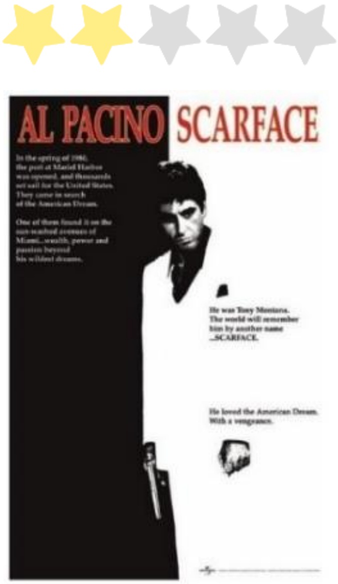


			
Alice	2	5	3
Bob	4		5
Carol	2	5	
Tom	2	???	???
		Prediction	
		2.6	**3.1**

Explicit feedback



2.5x better?

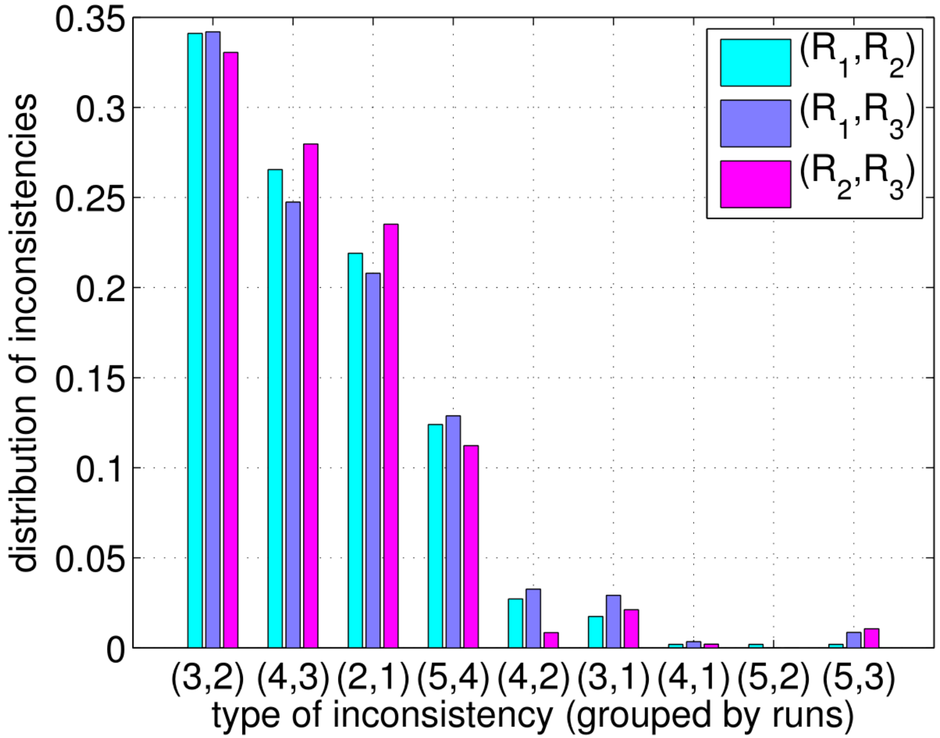


From neoclassical economics:
utility is an **ordinal concept**.

Explicit feedback

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

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

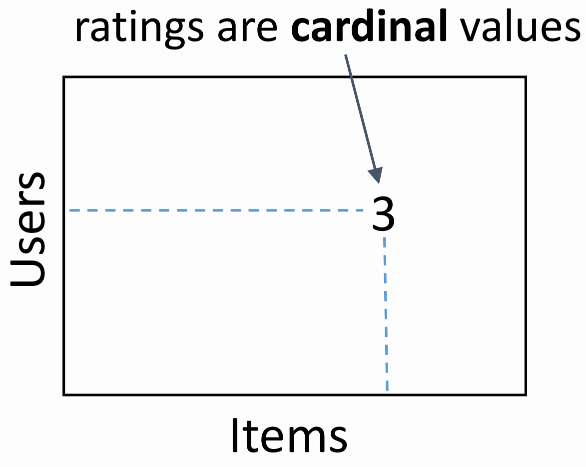


*X. Amatriain, J. M. Pujol, and N. Oliver "*I like it... I like it not: Evaluating User Ratings Noise in Recommender Systems*“, UMAP '09 Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization

Restating the problem

Standard model

$$User \times Item \rightarrow Rating$$

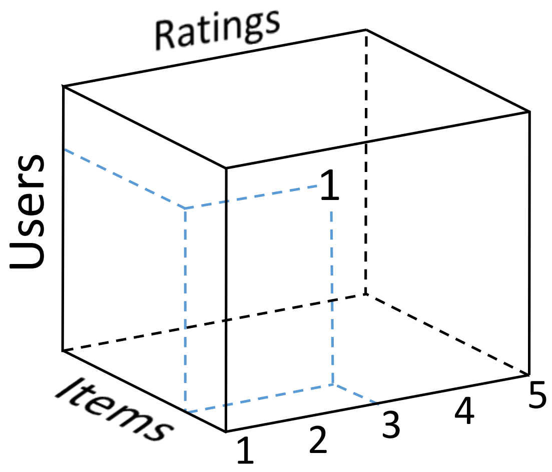


Technique: **Matrix factorization**

Collaborative Full Feedback model

CoFFee (proposed approach)

$$User \times Item \times Rating \rightarrow Relevance\ Score$$



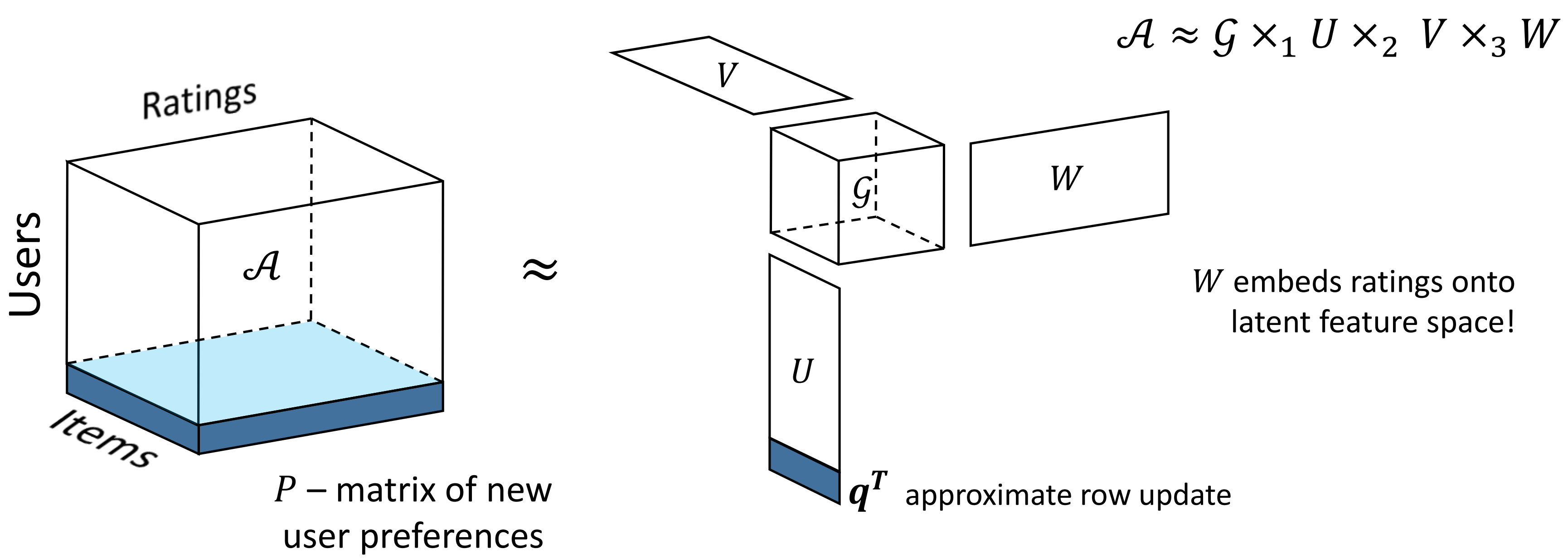
Technique: **Tensor Factorization**

based on Tucker Decomposition*

$$\mathcal{A} \approx \mathcal{G} \times_1 U \times_2 V \times_3 W$$

* T. G. Kolda and B. W. Bader, "Tensor Decompositions and Applications", 2009

Recommendations in real-time



"Shades of ratings"

Higher order *folding-in:*

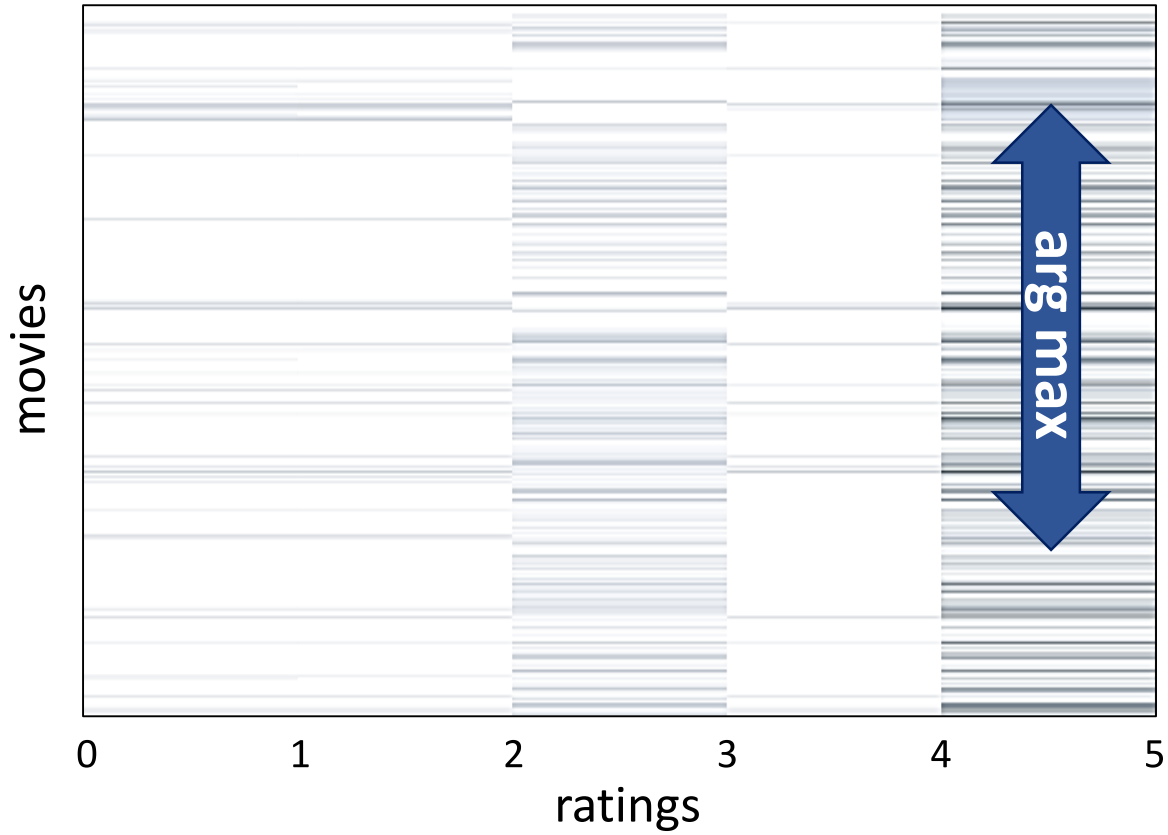
$R = VV^T P W W^T$ items relevance matrix

Compare to SVD: $\boldsymbol{r} = VV^T\boldsymbol{p}$

“Shades” of ratings

$$R = VV^T P W W^T$$

More dense colors correspond to higher relevance score.

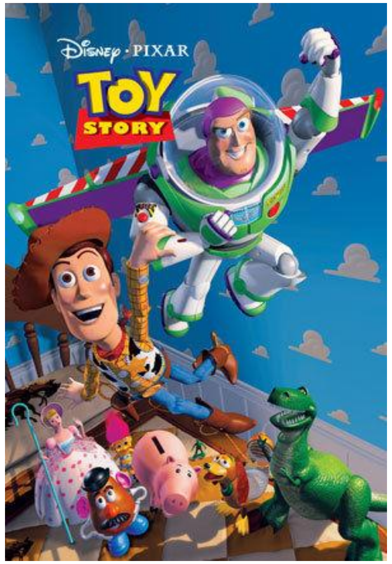


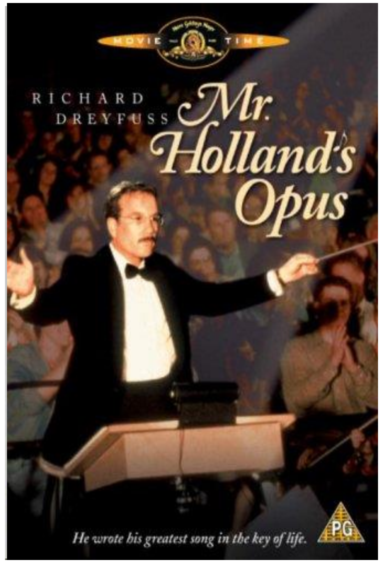
ranking task

Granular view of user preferences, concerning **all possible ratings**.

Model is **equally sensitive** to any kind of feedback.

Warm-start with CoFFee





	Scarface ★★☆☆☆	LOTR: The Two Towers ★★☆☆☆	Star Wars: Episode VII - The Force Awakens ★★★★★
CoFFee	Toy Story Mr. Holland's Opus Independence Day	Net, The Cliffhanger Batman Forever	Dark Knight, The Batman Begins Star Wars: Episode IV - A New Hope
SVD	Reservoir Dogs Goodfellas Godfather: Part II, The	LOTR: The Fellowship of the Ring Shrek LOTR: The Return of the King	Dark Knight, The Inception Iron Man

nDCG calculation

$$DCG = \sum_i \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

rel_i - true rating of a recommended item at position i

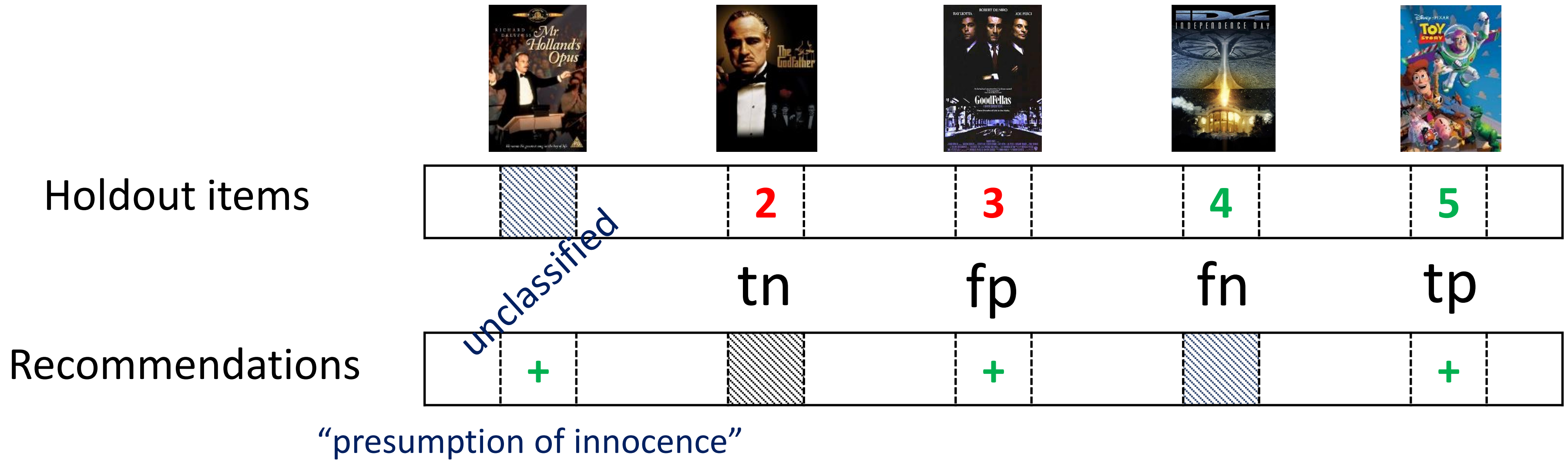
$$\boxed{nDCG = \frac{DCG}{iDCG}}$$

$iDCG = DCG$ with ideal ranking of items

Doesn't take into account the type of feedback: positive or negative.

Need to redefine metrics.

Using classical definition



Relevance based

$$Precision = \frac{tp}{tp + fp} \qquad Recall = \frac{tp}{tp + fn}$$

Ranking based

$$DCG = \sum_{p} \frac{2^{r_p} - 1}{\log_2(p+1)}$$

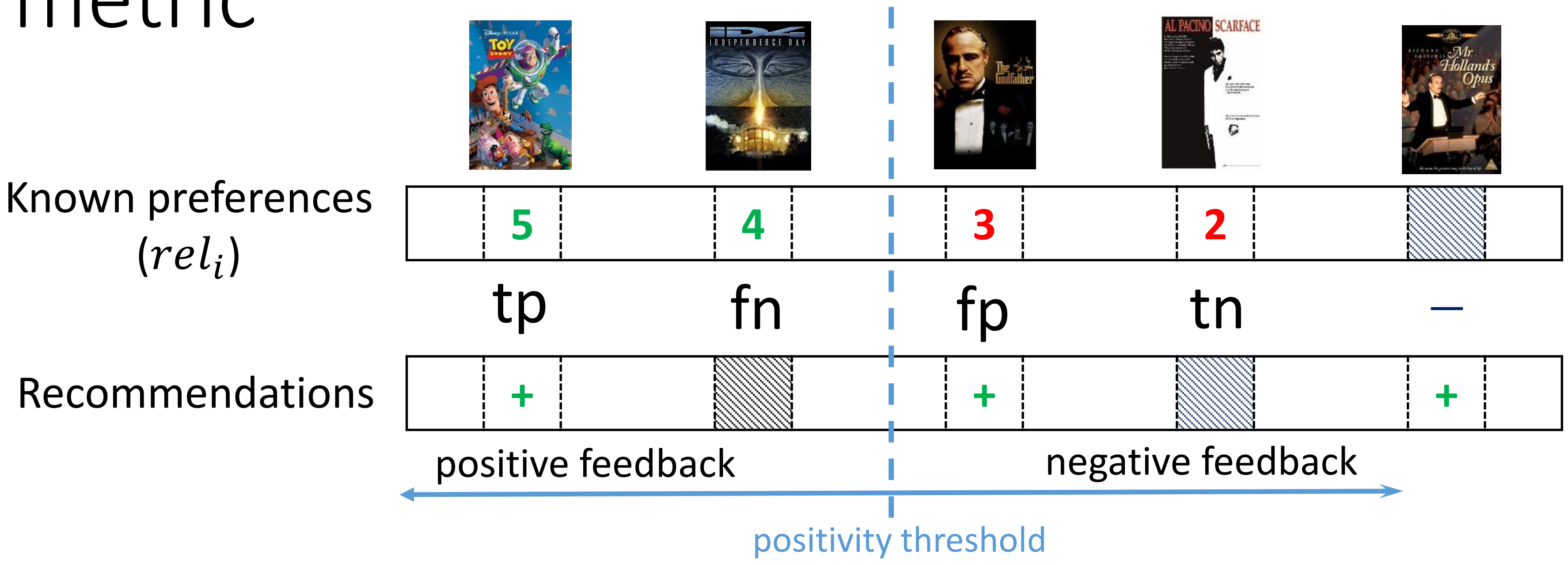
$$p : \{r_p \geq \text{positivity threshold}\}$$

r_p - value of positive feedback

New ranking metric

Methodology:

1. split feedback into positive and negative
2. treat negative part differently



Standard metric

$$DCG = \sum_{i \in I^+} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

Discounted Cumulative Gain

$I^+ = \{i: rel_i \geq r_p\}$ r_p - positivity threshold

measures potential user's enjoyment

New metric

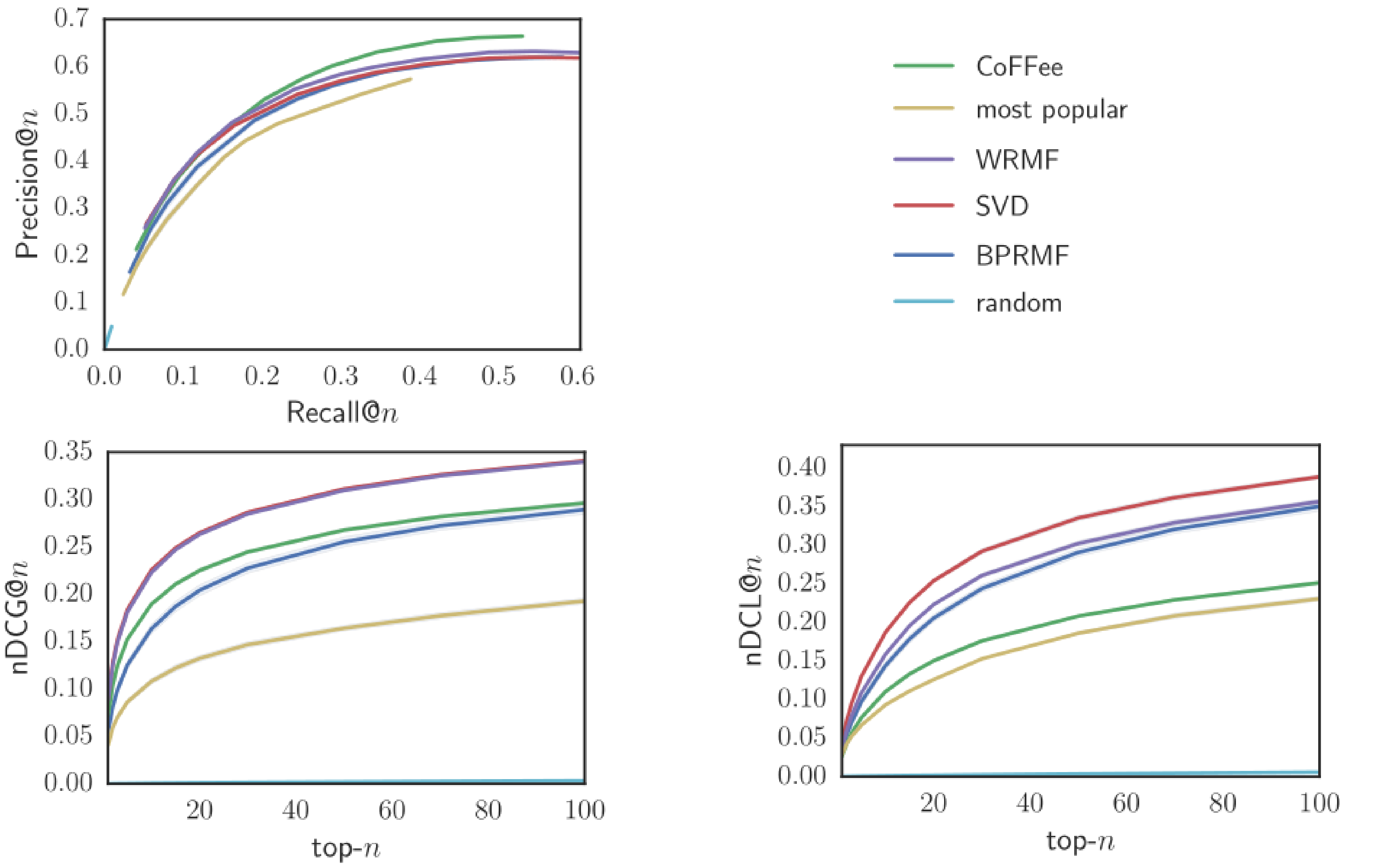
$$DCL = \sum_{i \in I^-} \frac{2^{r_p - rel_i} - 1}{-\log_2(i+1)}$$

Discounted Cumulative Loss

$I^- = \{i: rel_i < r_p\}$

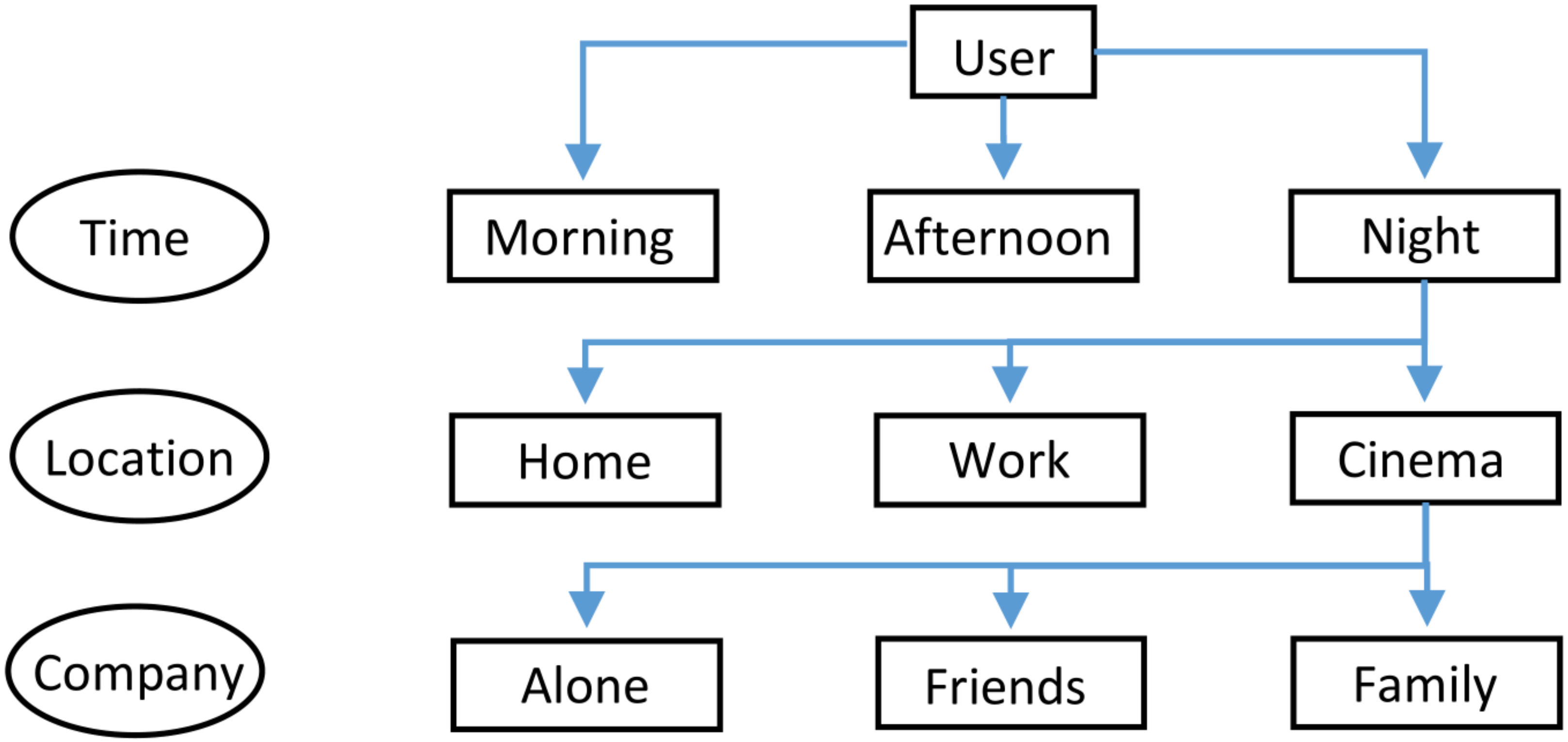
measures potential user's disappointment

Case study – which algorithm is better?



Read more: *"**Fifty shades of ratings: How to Benefit from Negative Feedback in Top-n Recommendations Tasks**"*, Evgeny Frolov and Ivan Oseledets. *Proceedings of the 10th ACM Conference on Recommender Systems, 2016,* pp. 91-98.

Tensors are also good for context-aware RecSys



Also: folksonomies, cross-domain RS, temporal models

Read more about tensors in recsys: "***Tensor methods and recommender systems***", Evgeny Frolov and Ivan Oseledets. *WIREs Data Mining Knowledge Discovery* 2017, vol. 7, issue 3.

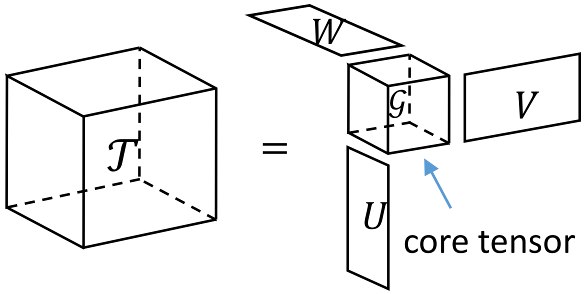
Tensor decompositions – higher order generalization of SVD

$$r_{ij} = \sum_{\alpha=1}^{R} \sigma_\alpha u_{i\alpha} v_{j\alpha}$$

Candecomp/Parafac (CP)

$$t_{ijk} = \sum_{\alpha=1}^{r} \lambda_\alpha u_{i\alpha} v_{j\alpha} w_{k\alpha}$$

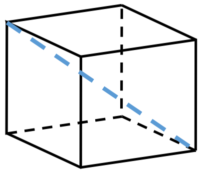
$$\mathcal{T} = [\![\boldsymbol{\lambda}; U, V, W]\!]$$



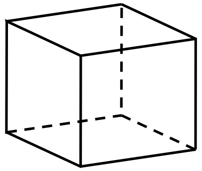
Tucker decomposition (TD)

$$t_{ijk} = \sum_{\alpha,\beta,\gamma=1}^{r_1,r_2,r_3} g_{\alpha\beta\gamma}\, u_{i\alpha}\, v_{j\beta}\, w_{k\gamma}$$

$$\mathcal{T} = \mathcal{G} \times_1 U \times_2 V \times_3 W$$



diagonal core with λ_α on diagonal



dense core of size $r_1 \times r_2 \times r_3$ with $g_{\alpha\beta\gamma}$ values

Tensor decompositions have much broader set of properties.

Rank can exceed $\max(M, N, K)$.

No generalization of Eckart-Young theorem.

Manifold of $rank\text{-}r$ tensors is **not closed**.

r_1, r_2, r_3- **multilinear rank**

Tensor product $\times_i$ is calculated with help of tensor matricization.

Two efficient algorithms: HOSVD, HOOI.

Key properties of tensor decompositions

Properties	TD	CP
Storage requirements	$dnr + {\color{red} r^d}$ curse of dimensionality	dnr
Uniqueness	No	Yes (under mild conditions)
Stability	stable	ill-posed*

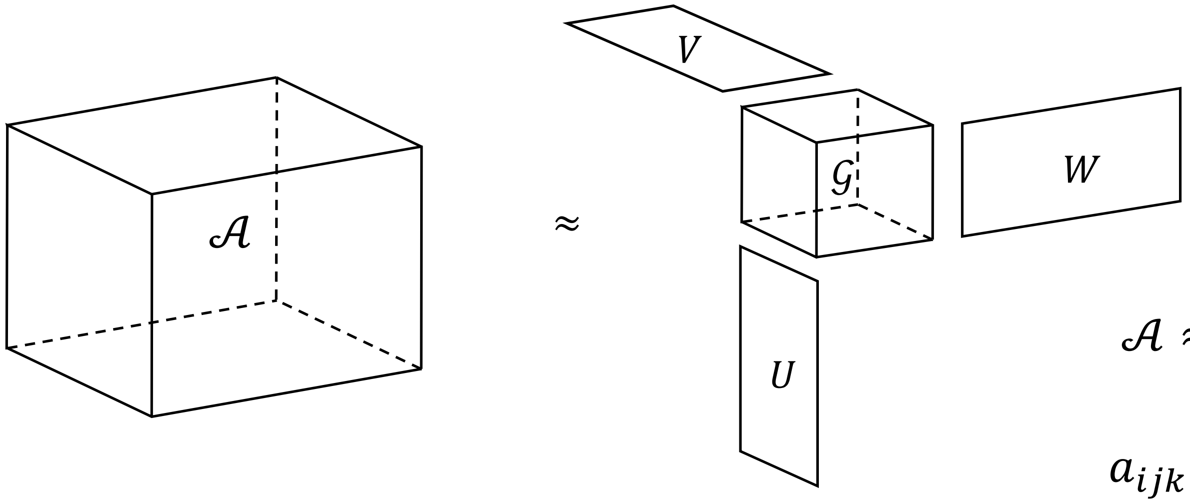
*V. De Silva and L.-H. Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem.

In recommender systems*:

Method	**Year**	**Type**	**Algorithm**
TagTR	2008	TD	HOSVD
Multiverse	2010	TD	SGD
PITF	2010	CP	SGD
TFMAP	2012	CP	SGD
GFF	2015	CP	ALS

***More examples in "Tensor Methods and Recommender Systems", E.Frolov, I.Oseledets** http://arxiv.org/abs/1603.06038

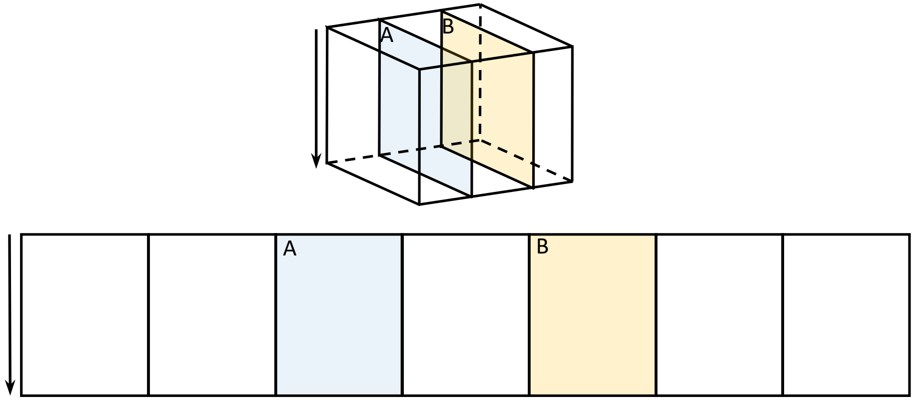
Tucker decomposition



$$\mathcal{A} \approx \mathcal{G} \times_1 U \times_2 V \times_3 W$$

$$a_{ijk} \approx \sum_{\alpha,\beta,\gamma} g_{\alpha\beta\gamma} u_{i\alpha} v_{j\beta} w_{k\gamma}$$

Tensor matricization (unfolding):



General ALS procedure (HOOI)

Initialize V, W randomly; require orthonormal columns

Repeat until convergence:

$U \leftarrow r_1$ left principal vectors of $(\mathcal{A} \times_2 V \times_3 W)^{(1)}$

$V \leftarrow r_2$ left principal vectors of $(\mathcal{A} \times_1 U \times_3 W)^{(2)}$

$W \leftarrow r_3$ left principal vectors of $(\mathcal{A} \times_1 U \times_2 V)^{(3)}$

$\mathcal{G} \leftarrow \mathcal{A} \times_1 U \times_2 V \times_3 W$

Your home task

- Download Movielens-10M dataset: http://files.grouplens.org/datasets/movielens/ml-10m.zip
- Try your favorite models on it
- Try comparing several models/configs via cross-validation

This is the dataset that will be used during the team competition!