Modern Recommender Systems and Their Applications

Evgeny Frolov

Research Scientist

Computational Intelligence Lab, Skoltech

evgeny.frolov@skoltech.ru

Outline

- Brief overview
    - a bit of history
    - case studies
    - recsys taxonomy
- Collaborative Filtering
    - latent factor models
    - Incorporating side information
    - Context-awareness
- Recent advances in ANN models
    - Autoencoders
    - Graph-based models
    - Sequential learning
- Current trends

What is a recommender system?

Examples:
- Amazon
- Netflix
- Pandora
- Last.fm
- etc.

Many different areas: e-commerce, news, social networks, tourism, entertainment, education…

Goal: predict user preferences given some prior information on user behavior.

Amazon's item-to-item approach

Iterative algorithm

```
For each item in product catalog, I1
  For each customer C who purchased I1
    For each item I2 purchased by
      customer C
      Record that a customer purchased I1
      and I2
  For each item I2
    Compute the similarity between I1 and I2
```

$$similarity(I_1, I_2) = \cos(p_1, p_2) = \frac{(p_1, p_2)}{\|p_1\|\|p_2\|}$$

p_k- one-hot vector of purchases of item k

+$2.93 billion to revenue after integration of recommendations

Amazon. com recommendations: **Item-to-item** collaborative filtering
G Linden, B Smith, J York - IEEE Internet computing, 2003
Cited by 7117 Related articles All 42 versions

Netflix prize story

October 2, 2006 - June 26, 2009

Contest: Given a database of movies rated by users, beat Netflix's recsys by at least 10%

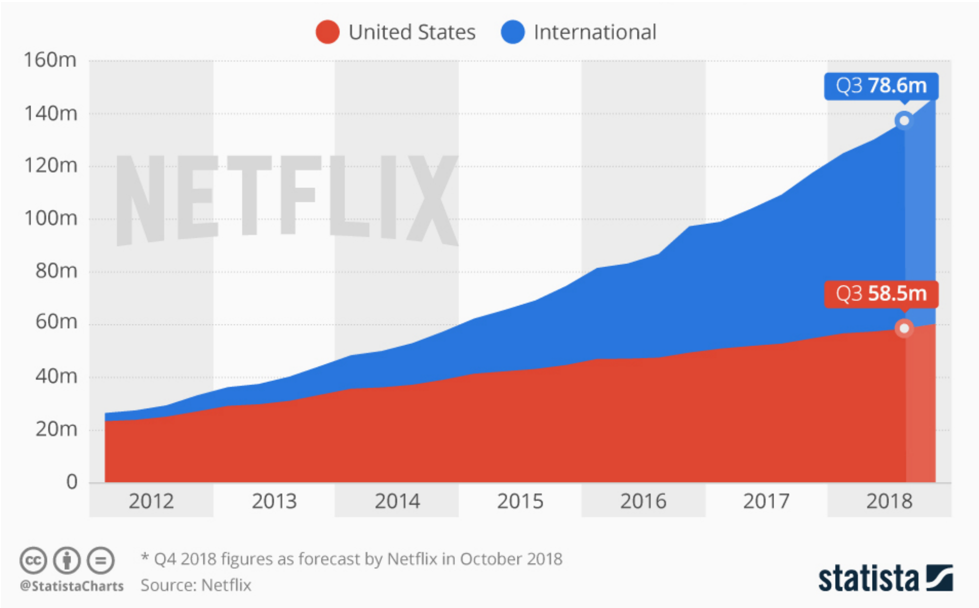
Award: **$1,000,000**

Key to success: ensemble of models.
Actual solution was never implemented!
https://www.techdirt.com/blog/innovation/articles/20120409/03412518422/why-netflix-never-implemented-algorithm-that-won-netflix-1-million-challenge.shtml

However, latent factors models based on **matrix factorization** gained popularity afterwards.

The good and the bad:
+ made recsys field much more visible
- shifted attention to wrong aspects (still recovering)

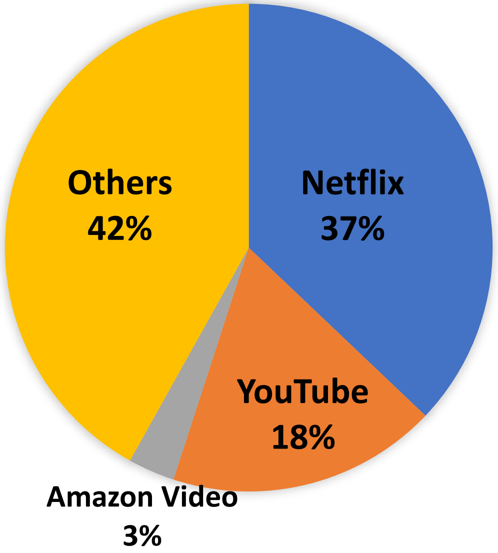
Netflix impact



- As of Q4 2019, more than 167M paid accounts
- 61M from the US.

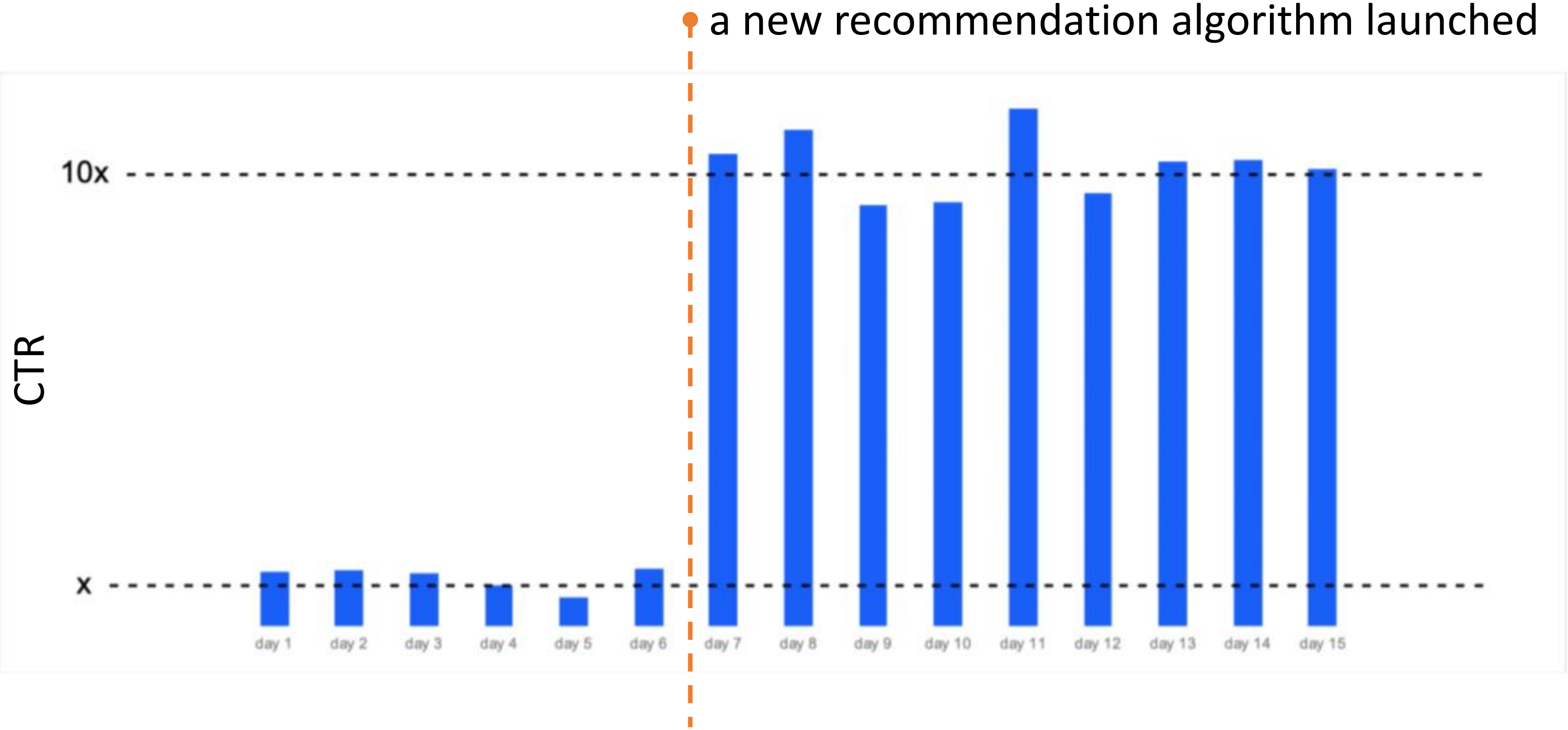
80% of what people watch comes from recommendations => $1 billion savings

Internet media-traffic share in North America (2018)



Sources:
http://dl.acm.org/citation.cfm?id=2843948
http://www.internetphenomena.com/tag/amazon-video/
https://www.businessofapps.com/data/netflix-statistics/
https://www.statista.com/statistics/250934/quarterly-number-of-netflix-streaming-subscribers-worldwide/

Ozon (Russian online retailer)



For every purchase, Ozon also offers an accompanying product. *«Harry Potter» problem.*

Previous algorithm: hand-crafted association rules. Required a lot of attention from the data analytics team.

Source: Pavel Pekichev’s talk and Yandex.Zen Meetup, June 28, 2019

[Дзен-митап: Рекомендательные системы изнутри]

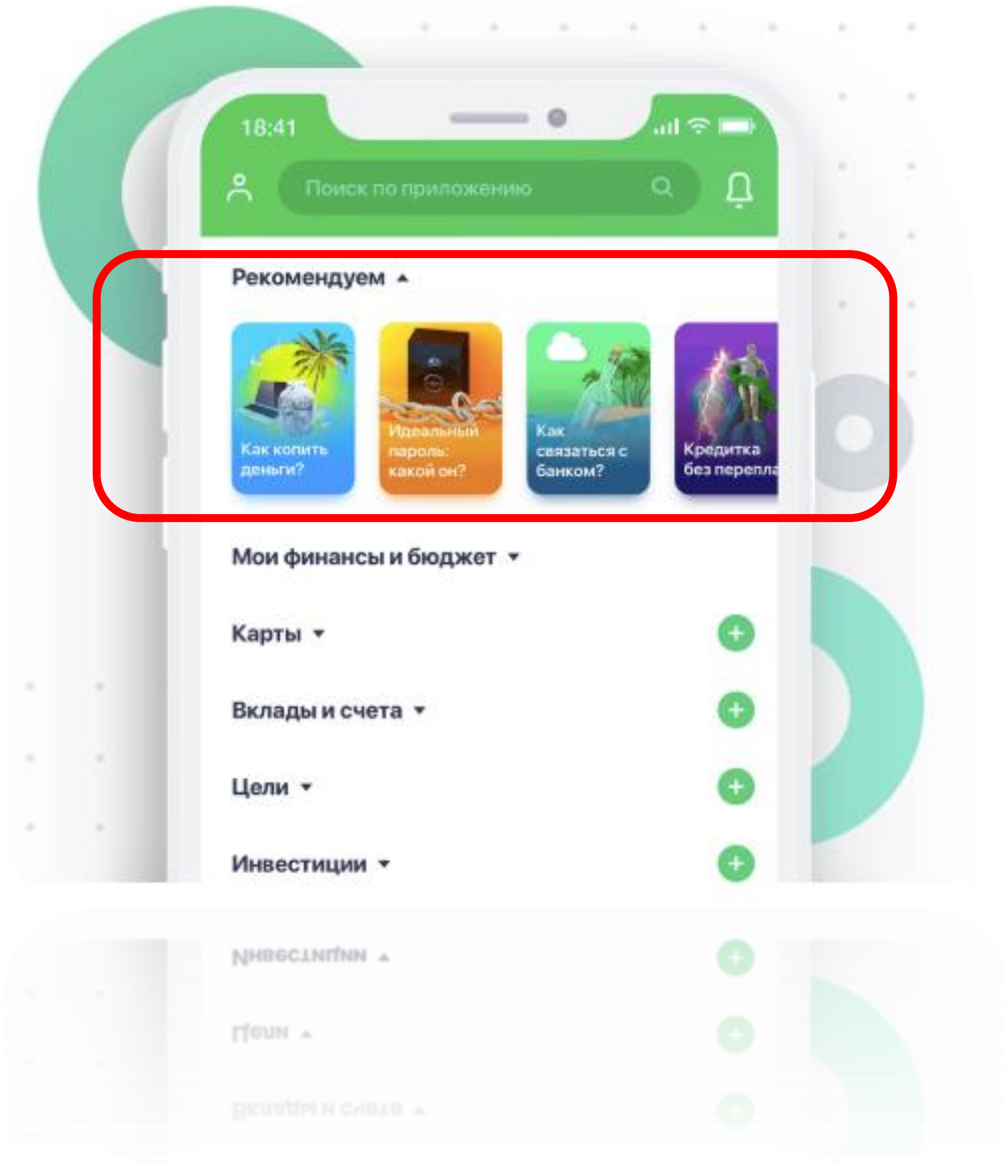
IKEA’s creative intelligence

- selling «inspirational shopping experience»
- intelligent assistance for finding good composition

Designer-driven add-to-cart recommendations
https://dl.acm.org/doi/10.1145/3298689.3346959

Photos taken at RecSys’19 conference

Sberbank Stories



Personalization of stories recommendations increases CTR, which:

- helps promoting bank services and products
- stimulates additional transactions via partner networks (e.g., cinema tickets, discount coupons, etc.)

Typical problems and challenges

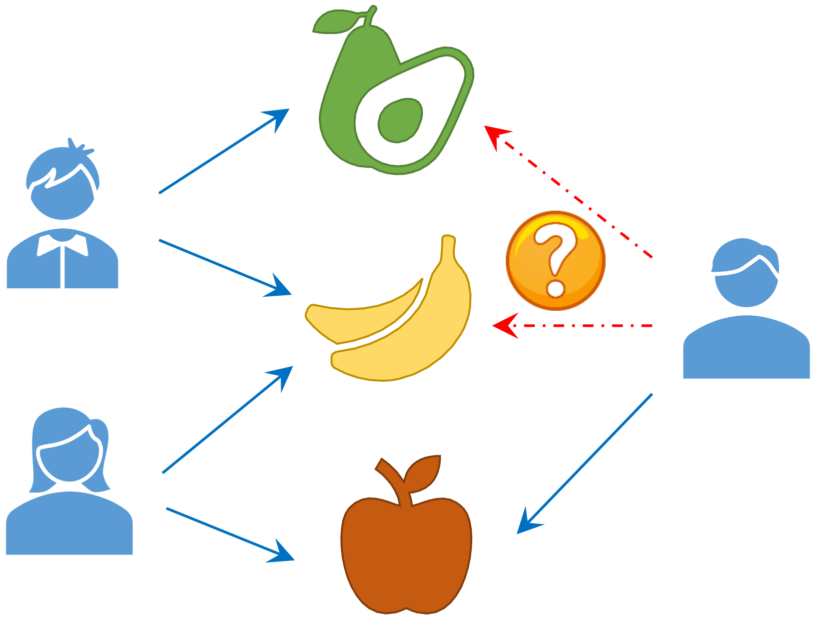
cold-start	• resolving recommendation uncertainty • finding representative items
missing values	• 99.99...% of unknowns • data is Missing Not at Random (MNAR)
debiasing	• popularity biases • causality and feedback loops
short head / long tail	• 5% of items may hold 40% of all interactions • recommending niche products
evaluation	• lack of standardization • offline evaluation vs. AB-tests
explanation	• why a product is recommended • why a user will like a product
complex models	• incorporating content and context information • multi-task learning
performance	• quick model computation • real-time recommendations

Recommender systems internals

Engine
- content-based
- collaborative filtering
- hybrid systems

Outcome
- “utility" score
- ranked list of items

Collaborative Filtering

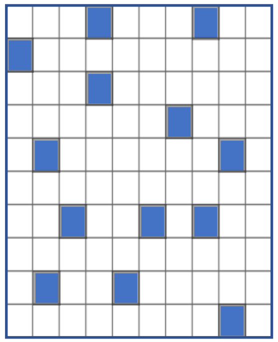


- **Non-personalized**
  - popularity models
  - baseline predictors
- **Personalized**
  - Memory-based — item-based, user-based
    - kNN-based models
    - some graph-based models
  - Model-based — latent factors models
    - Matrix/Tensor Factorization
    - Artificial Neural Networks

A general view on recommendation problem

utility matrix A

m users



n items

Incomplete data:

- known entries
- unknown entries

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

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

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)

top-n recommendations task:

$$\text{toprec}(i, n) \coloneqq \arg \max_{j}^{n} r_{ij}$$

Low-rank approximation with matrix factorization

Assumption:
there is a *small* number of common patterns in human behavior + *individual variations*

$$A_{full} = R + E$$

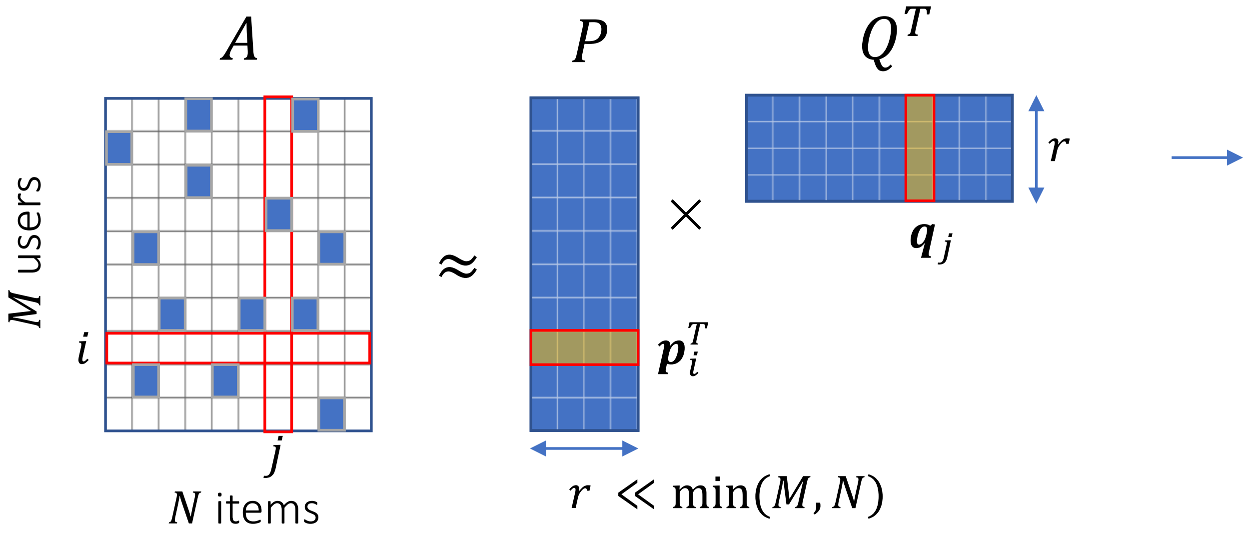
$$R = PQ^T$$

rows of P and Q give *embeddings* of users and items onto a latent feature space

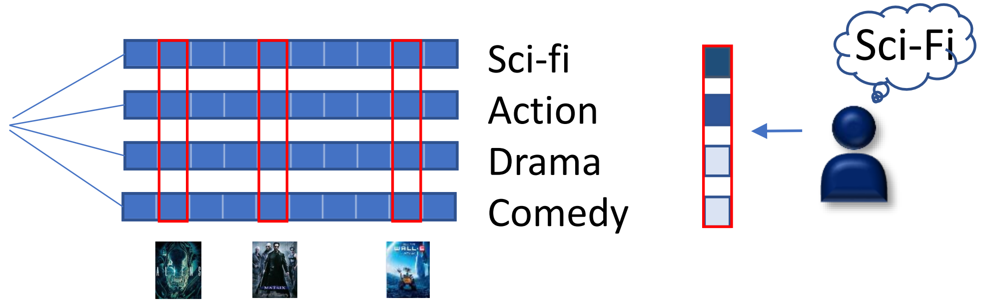
predicted utility of item j for user i

$$\boxed{r_{ij} \approx \boldsymbol{p}_i^T \boldsymbol{q}_j = \sum_{k=1}^{r} p_{ik} q_{jk}}$$

$\boldsymbol{p}_i$ - latent feature vector for user i
$\boldsymbol{q}_j$ - latent feature vector for item j



Simplistic view: latent features ↔ genres



Variations of MF approaches

- PureSVD – contentwise.com

$$\mathcal{L}(A,R) = \|A_0 - R\|_F^2, \; R = U\Sigma V^T = VV^T A_0, V^T V = I$$

- ALS + NN – Yandex.Zen

$$\mathcal{L}(A,R) = \frac{1}{2}\|W \odot (A - PQ^T)\|_F^2 + \frac{1}{2}\lambda(\|P\|_F^2 + \|Q\|_F^2)$$

- iALS – Ivi, Yandex.Music

$$\mathcal{L}(A,R) = \frac{1}{2}\|W \odot (S - PQ^T)\|_F^2 + \frac{1}{2}\lambda(\|P\|_F^2 + \|Q\|_F^2)$$

Why SVD still?

- Criteo – billion-scale recsys

Read More:
[SparkRSVD open-sourced by Criteo for large scale recommendation engines]

Github:
[criteo/Spark-RSVD: Randomized SVD of large sparse matrices on Spark]

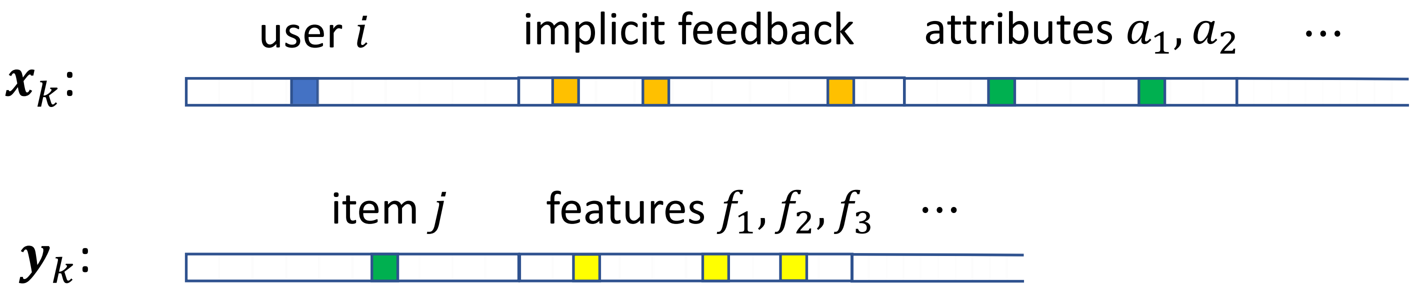
Generate random matrix $\Omega \in \mathbb{R}^{n\times(k+p)}$
$Y \leftarrow A\Omega$
$Q \leftarrow \mathrm{QR}(Y)$ $\triangleright$ QR decomposition of Y
for $i \leftarrow 1$ to q **do**
  $Y \leftarrow A^T Q$
  $Q \leftarrow \mathrm{QR}(Y)$
  $Y \leftarrow AQ$
  $Q \leftarrow \mathrm{QR}(Y)$
end for
$B \leftarrow Q^T A$
$\widetilde{Q}, \widetilde{R} \leftarrow \mathrm{QR}(B^T)$
SVD decomposition of $\widetilde{R} = \widetilde{V}\Sigma\widetilde{U}^T$
return $U = Q\widetilde{U}$

SVDFeature

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

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

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



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

Optimized with ALS, SGD.

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**

	User				Movie					Other Movies rated					Time	Last Movie rated					Target y	
	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...		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	$y^{(6)}$

HybridSVD

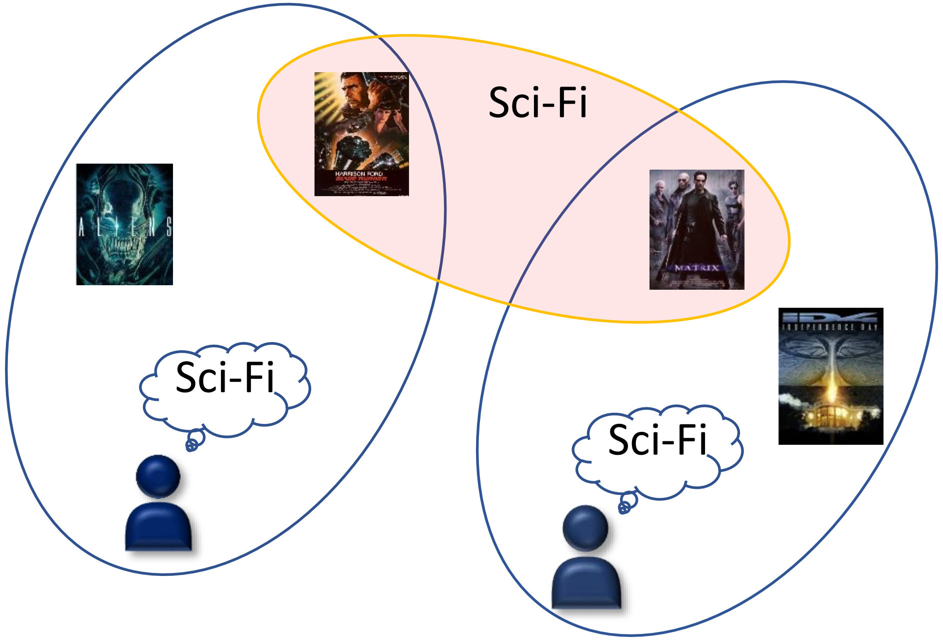
[Frolov, Oseledets 2018]

“Similarity” of users i and j depends on co-occurrence of items in their preferences.

$$G = AA^\top = U\Sigma^2 U^\top \quad \leftrightarrow \quad g_{ij} = a_i^\top a_j$$

Key idea: replace scalar products with a bilinear form.

$$\mathrm{sim}(i,j) \sim a_i^\top S a_j$$



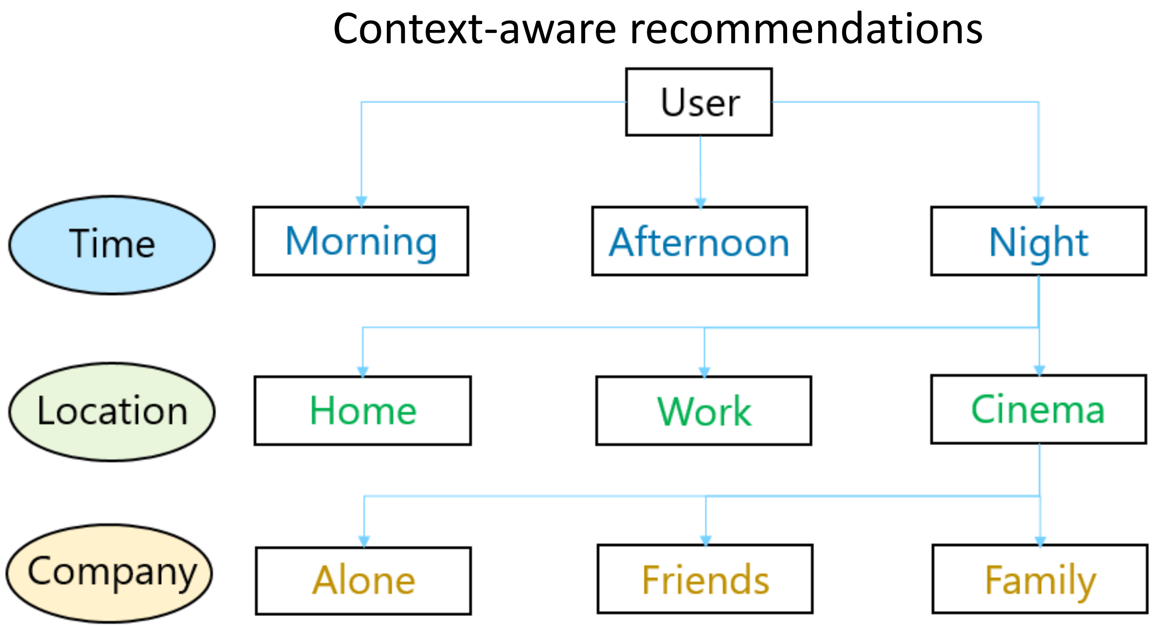
Creates “virtual” links based on side features.

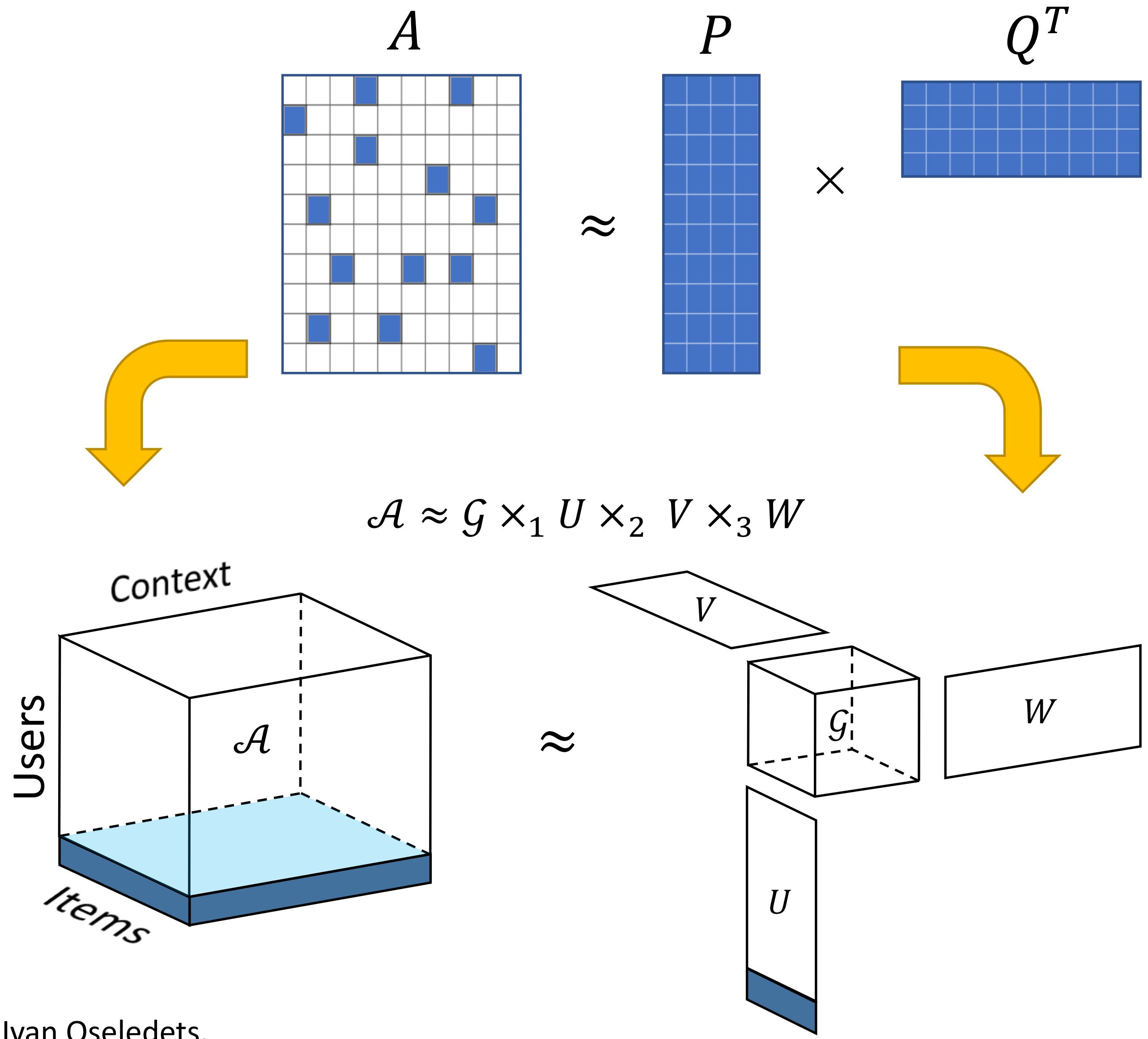
$$\begin{cases} ASA^\top = U\Sigma^2 U^\top \\ A^\top K A = V \Sigma^2 V^\top \end{cases}$$

Similarity matrix *S*

1			
	1	0.5	
	0.5	1	
			1

Contextual information and tensor factorization





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

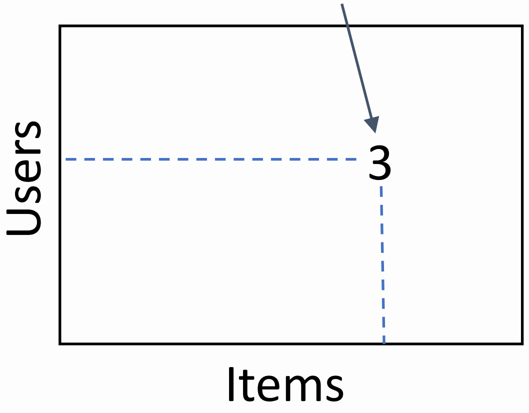
"Fifty shades of ratings"

[Frolov, Oseledets 2016]

Standard model

$$User \times Item \rightarrow Rating$$

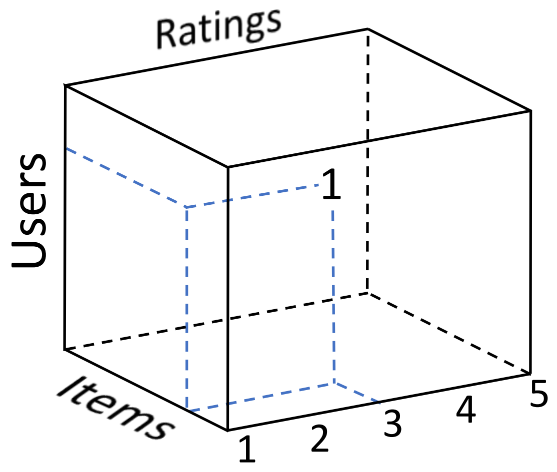
ratings are **cardinal** values



Technique: **Matrix factorization**

Collaborative Full Feedback model CoFFee*

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



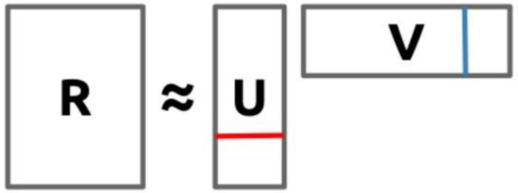
Technique: **Tensor Factorization**

based on Tucker Decomposition

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

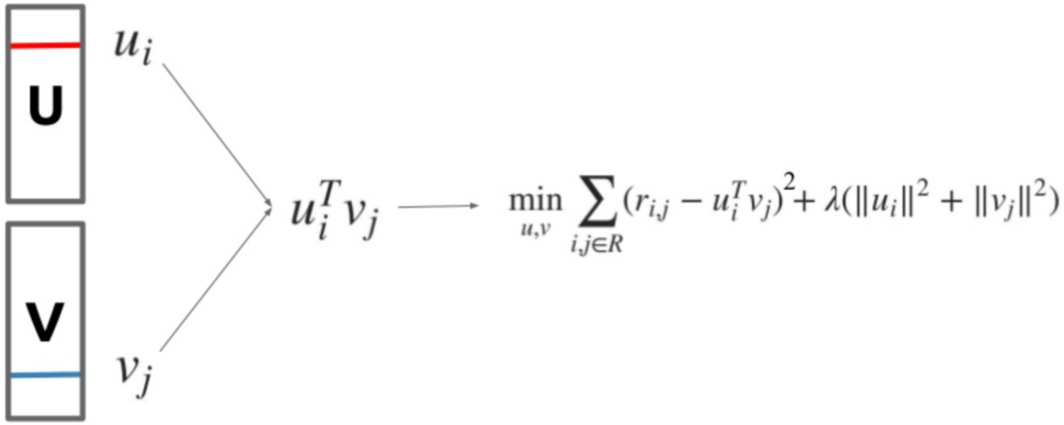
From matrix factorization to neural networks

A Matrix Factorization view



$$\min_{u,v} \sum_{i,j \in R} (r_{i,j} - u_i^T v_j)^2 + \lambda(\|u_i\|^2 + \|v_j\|^2)$$

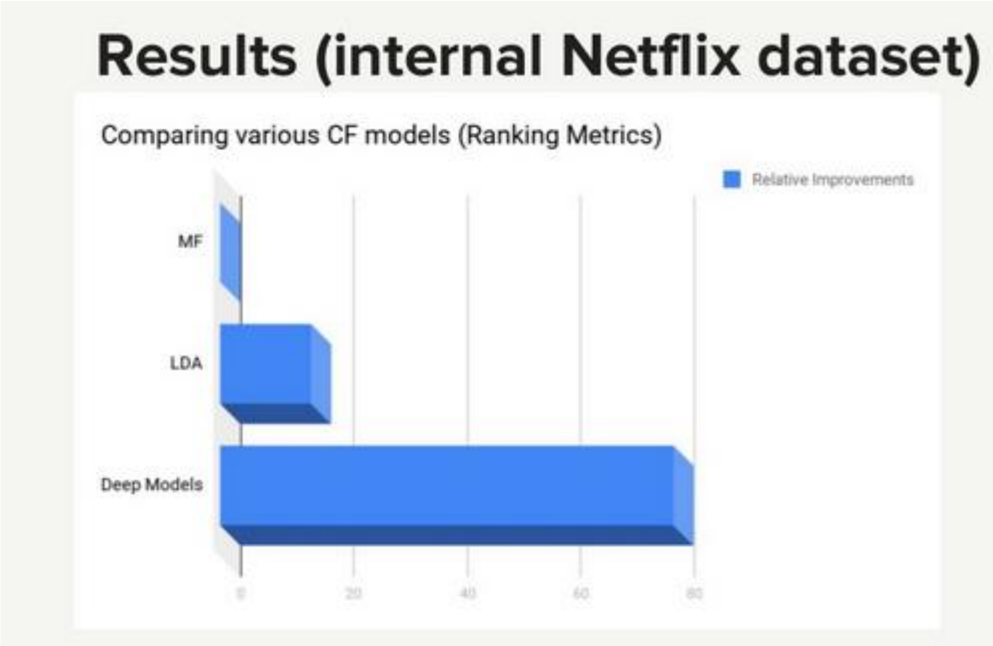
A Feed-Forward Network view



Source: https://www.slideshare.net/justinbasilico/recent-trends-in-personalization-a-netflix-perspective

A bit about "NN hype"

How it is presented



How it works in practice

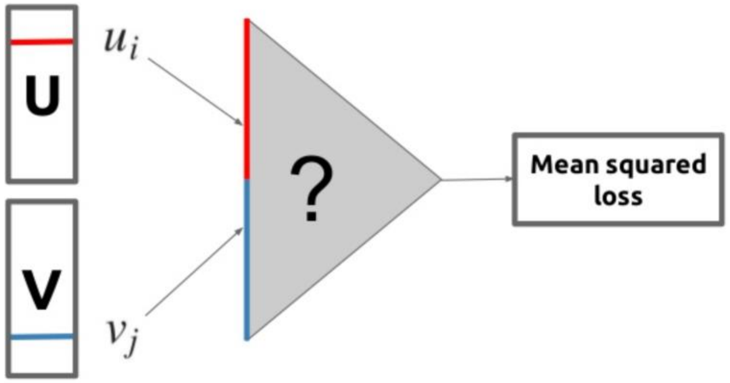
MF

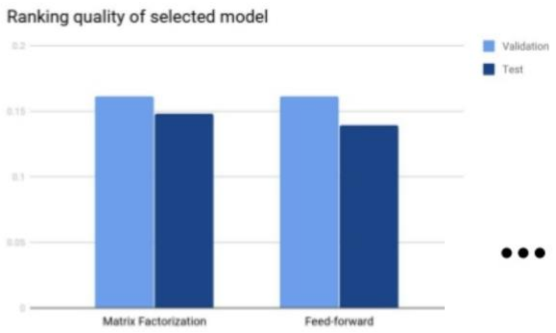
Method	movielens HR [%]	movielens ARHR [%]	movielens NDCG [%]	yahoo HR [%]	yahoo ARHR [%]	yahoo NDCG [%]	pinterest HR [%]	pinterest ARHR [%]	pinterest NDCG [%]
EigenRec	45.21	20.44	26.35	48.12	23.30	29.23	33.81	13.51	18.41
PureSVD	44.14	19.33	25.36	38.68	18.30	22.62	30.97	11.85	16.30
RP3b	34.87	15.02	19.66	41.51	17.82	22.94	27.01	8.07	12.45
SLIM	46.34	21.39	27.28	52.44	26.15	32.35	34.17	13.63	18.57
Mult-DAE	44.06	18.97	24.83	45.37	21.46	27.07	35.03	13.79	18.77
Mult-VAE	44.35	19.50	25.31	45.09	21.22	26.80	35.13	13.73	18.71

RecWalk: Nearly Uncoupled Random Walks for Top-N Recommendation
http://www.nikolako.net/papers/ACM_WSDM2019_RecWalk.pdf

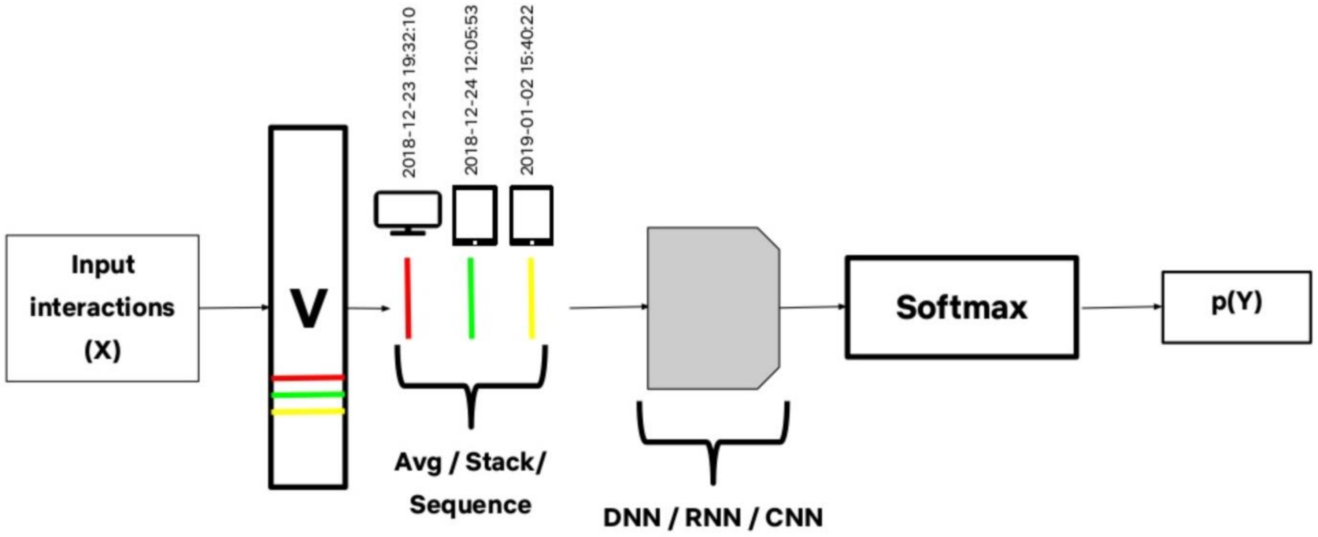
Netflix experience

... isn't always the best





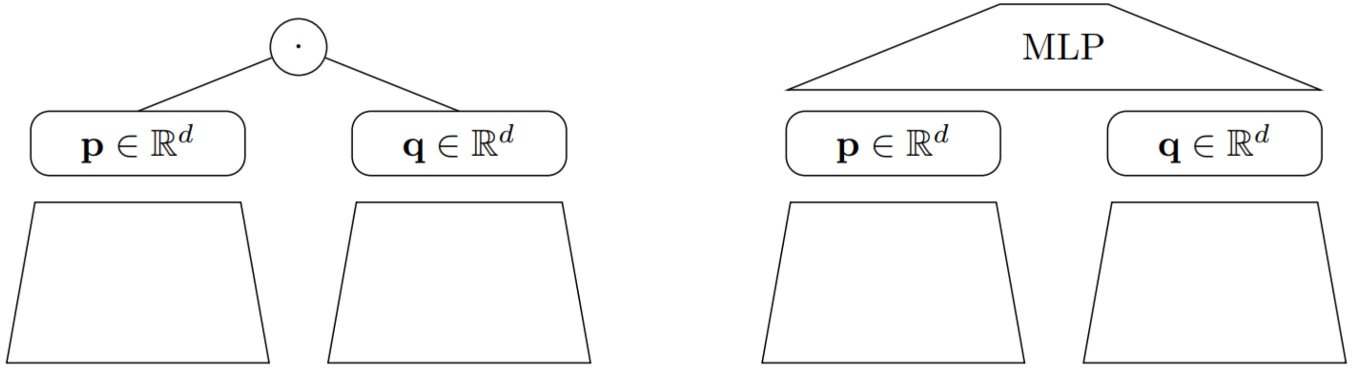
... but opens up many possibilities

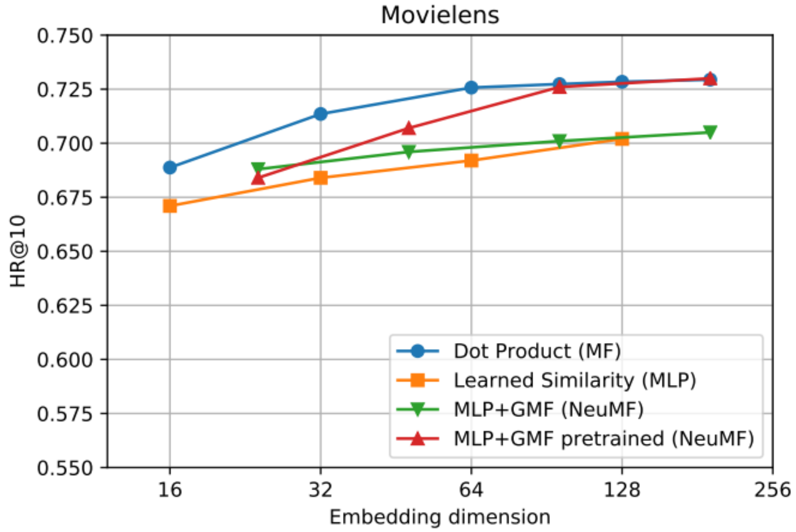


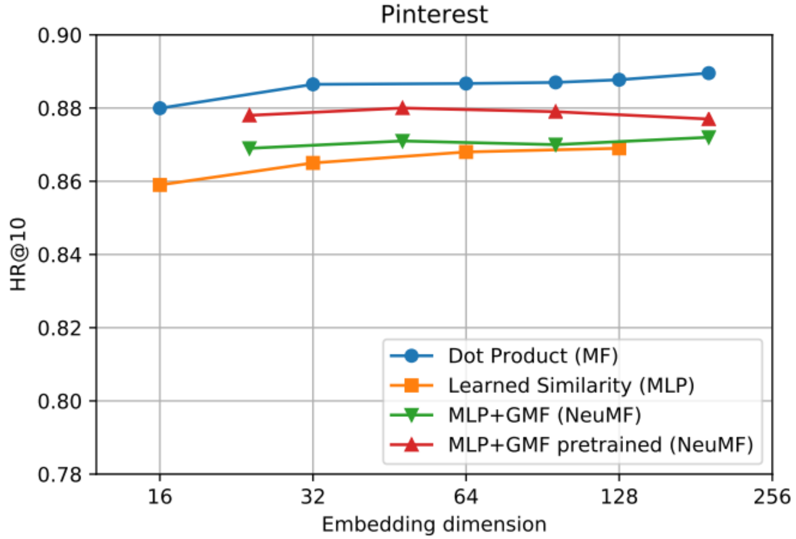
https://slideslive.com/38916930/recent-trends-in-personalization-a-netflix-perspective
https://www.slideshare.net/justinbasilico/recent-trends-in-personalization-a-netflix-perspective

MLP vs dot product

$$\phi^{\mathrm{dot}}(\mathbf{p}, \mathbf{q}) = \langle \mathbf{p}, \mathbf{q} \rangle \qquad \phi^{\mathrm{MLP}}(\mathbf{p}, \mathbf{q}) = \mathbf{f}_{W_l, \mathbf{b}_l}(\ldots \mathbf{f}_{W_1, \mathbf{b}_1}([\mathbf{p}, \mathbf{q}]) \ldots)$$



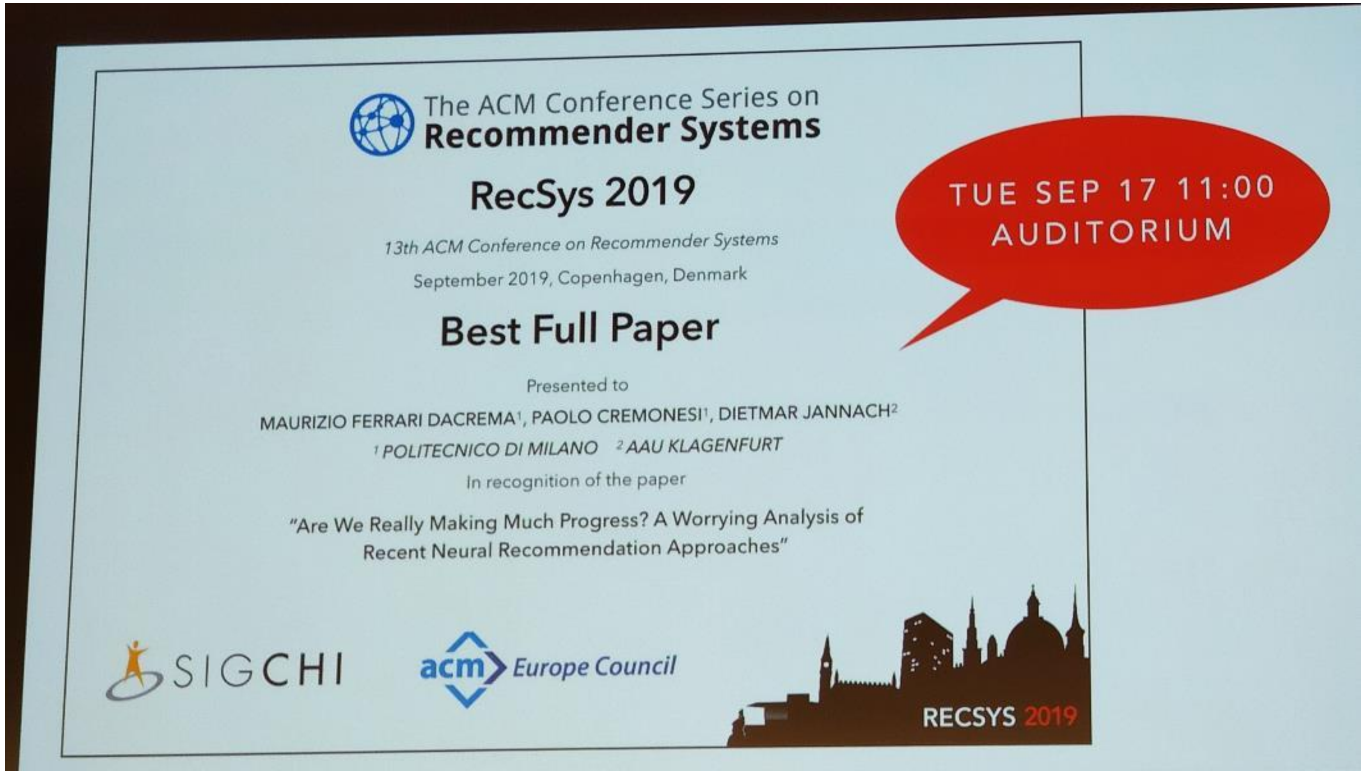




fully connected feed-forward networks seem to be not good at learning multiplicative relationships (like dot products)

Source: Rendle et al., Neural Collaborative Filtering vs. Matrix Factorization Revisited, 2020

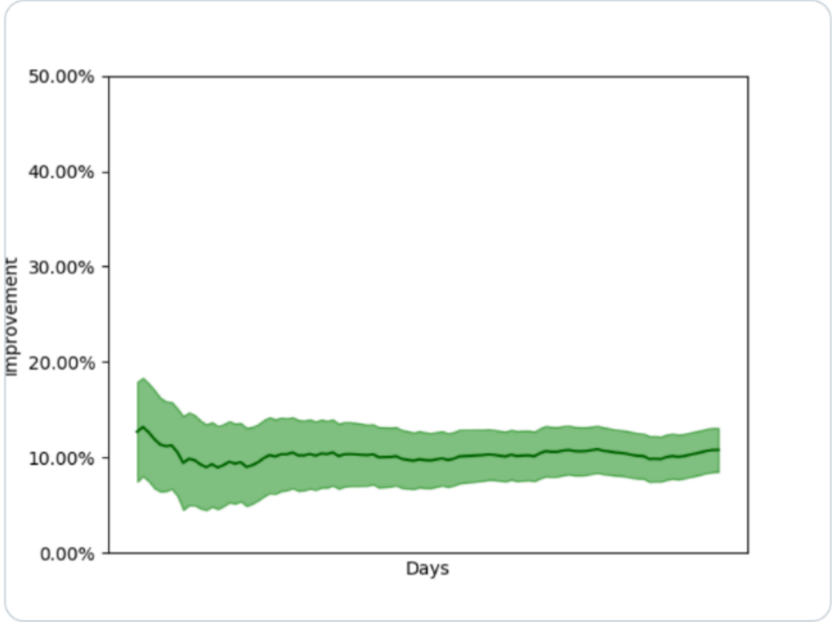
RecSys best Paper Award 2019



"Hot" discussion on twitter

Balázs Hidasi
@balazshidasi

This is the relative increase in revenue we got in an online A/B test (100+ days) measuring GRU4Rec against an already optimized recommender model (which already performs significantly better than simple baselines like MF of item based CF). (1/6) #recsys2019

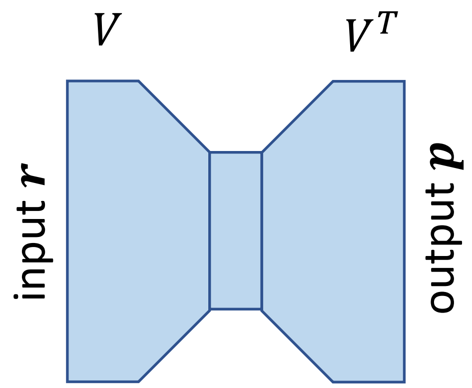


https://twitter.com/balazshidasi/status/1173885942400241664

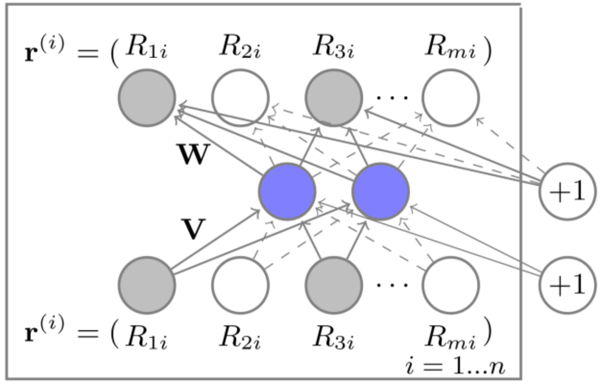
https://twitter.com/alexk_z/status/1173911139287216128

Autoencoders

Simple linear autoencoder: PureSVD $\boldsymbol{p} = VV^T\boldsymbol{r}$



In a general case:



$$\min_{\theta} \sum_{\mathbf{r} \in \mathbf{S}} ||\mathbf{r} - h(\mathbf{r};\theta)||_2^2 \qquad h(\mathbf{r};\theta) = f\left(\mathbf{W} \cdot g(\mathbf{V}\mathbf{r} + \boldsymbol{\mu}) + \mathbf{b}\right)$$

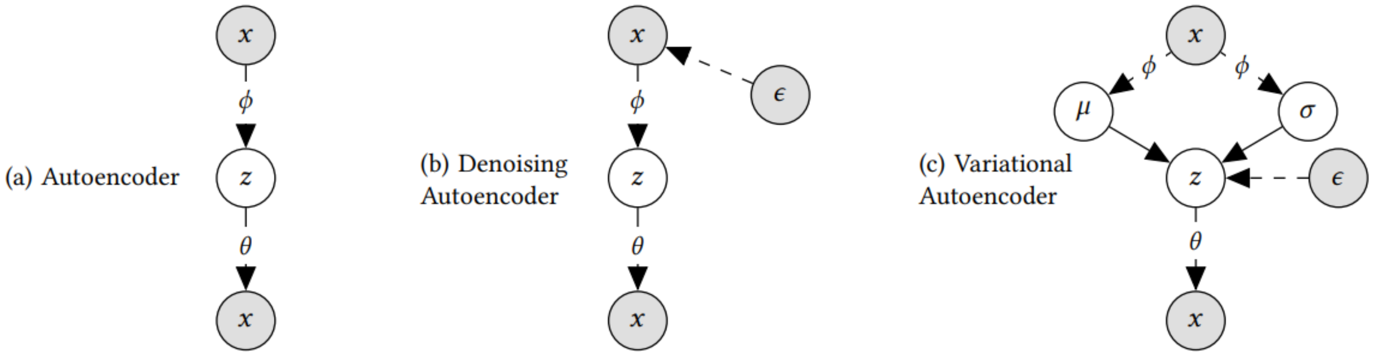
Doesn't work well out-off-the-box, need modifications e.g.:

- Regularization, dropout
- Adding noise in input (e.g., masking) or hidden layer (e.g., gaussian noise)
- Parametrization of hidden state
- Composite loss

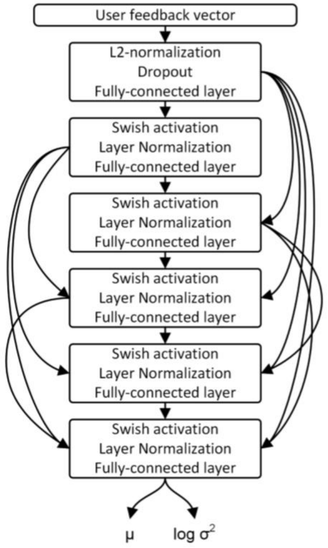
Image source: AutoRec: Sedhain et al. Autoencoders Meet Collaborative Filtering, 2015

Autoencoders evolution

AE→DAE→SDAE → MultDAE → MultVAE →RecVAE…



New architecture	Composite prior	$\beta(x)$ rescaling	Alternating training	Decoder w/o denoising	NDCG@100 ML-20M	NDCG@100 Netflix	NDCG@100 MSD
					0.426	0.386	0.319
✓					0.428	0.388	0.320
✓	✓				0.435	0.392	0.325
✓		✓			0.435	0.390	0.321
✓			✓	✓	0.427	0.387	0.319
✓	✓	✓			0.438	0.390	0.325
	✓	✓	✓	✓	0.420	0.380	0.308
✓		✓	✓	✓	0.434	0.383	0.321
✓	✓		✓	✓	0.437	0.392	0.323
✓	✓	✓	✓		0.441	0.391	0.322
✓	✓	✓	✓	✓	**0.442**	**0.394**	**0.326**

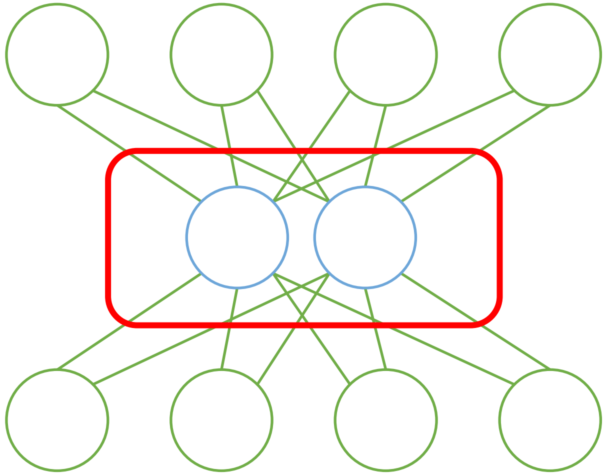


MultVAE [Liang et al. 2018] is used at Netflix, where *input dimensionality is relatively small.*

Generally, it is hard to maintain in large production environments.

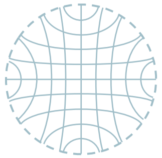
Hyperbolic geometry in Autoencoders

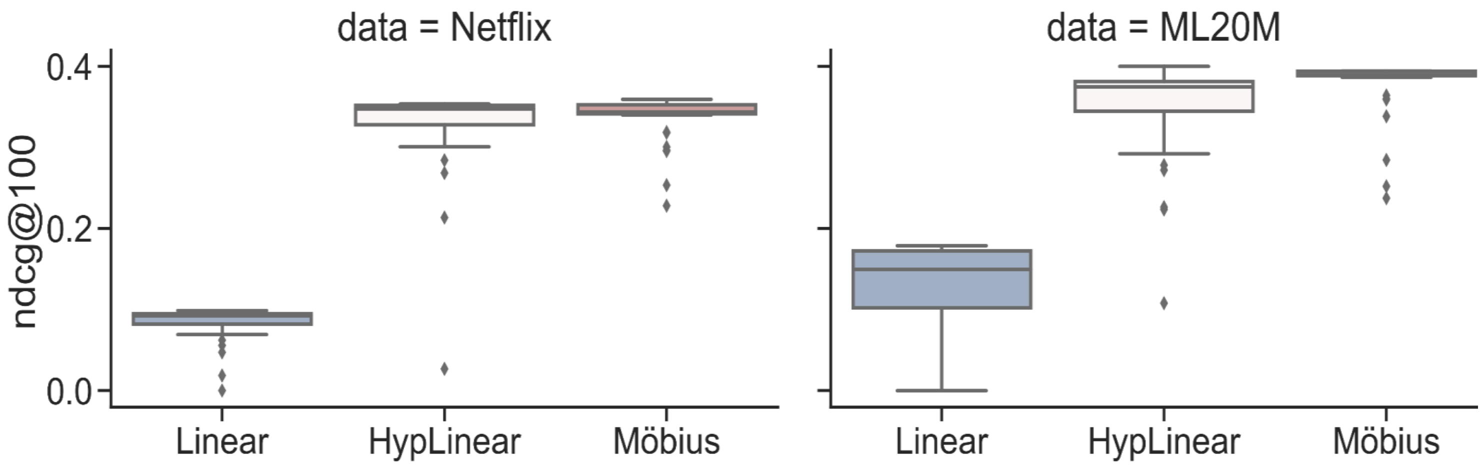
[Mirvakhabova and Frolov et al. 2020]



Key idea: replace linear operation with their hyperbolic counterparts

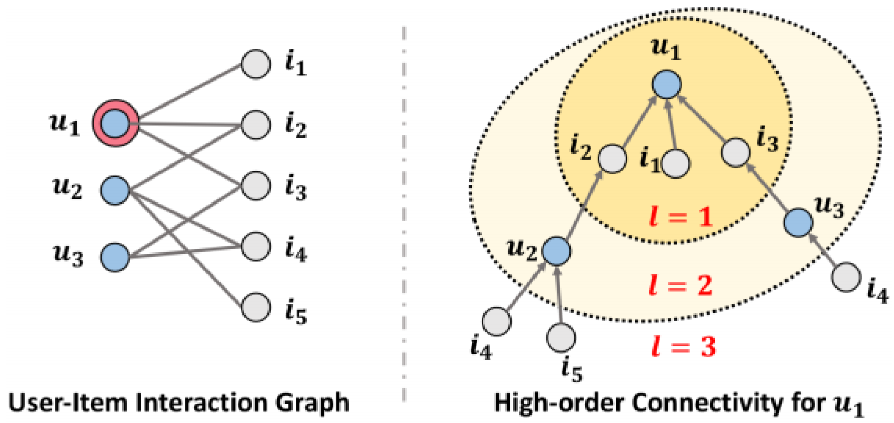
$$Wx + b \rightarrow W^{\otimes_c}(x) \oplus_c b$$

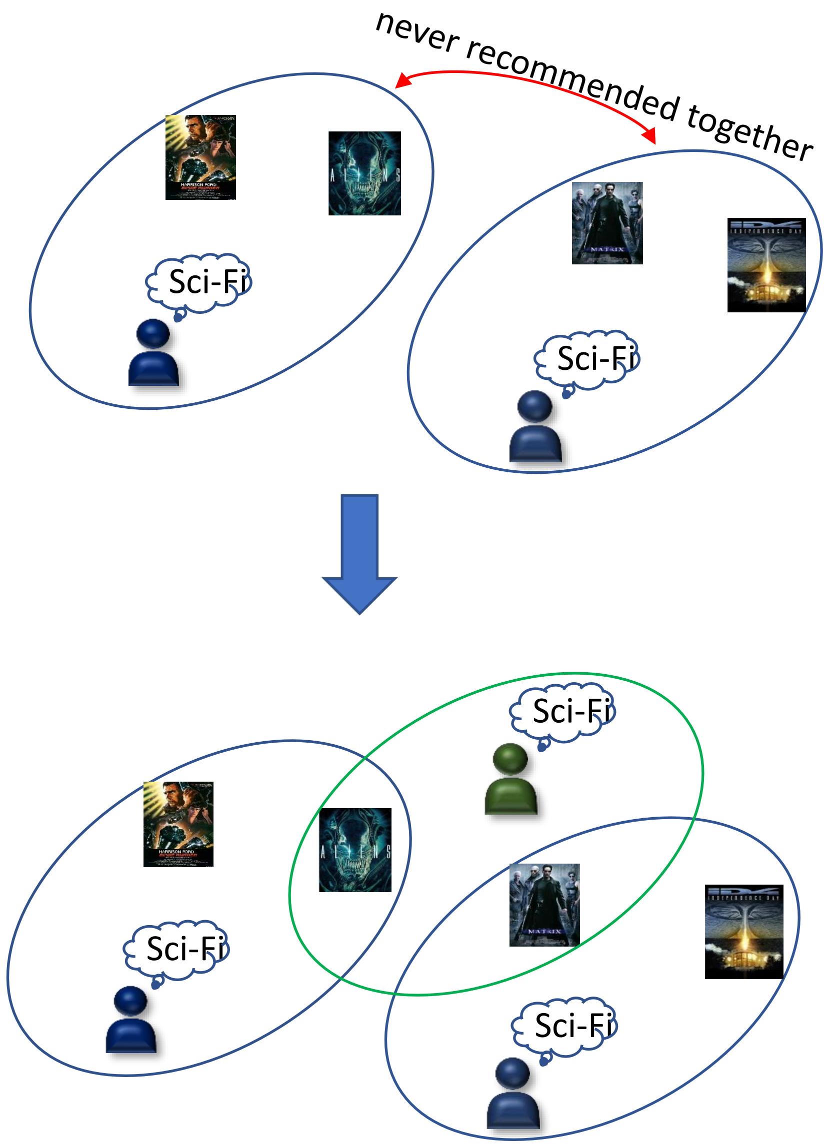




- implemented internally at Sberbank AI Lab
- being tested at Yandex, 100M scale

Graph-based models



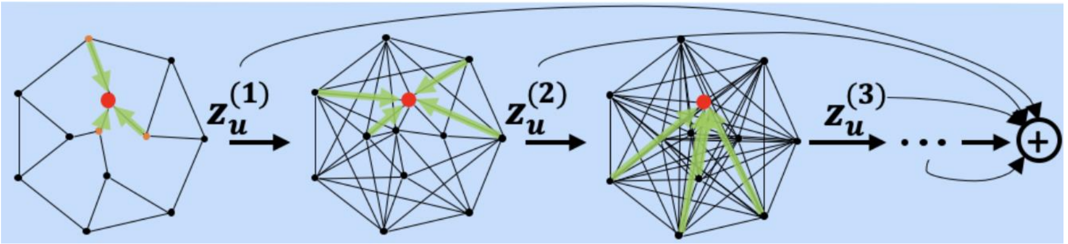


Possible approaches:

- Random-walk methods, e.g. personalized page-rank models RecWalk, Personalized Diffusions
- **Graph-convolutional neural networks**, e.g. NGCF

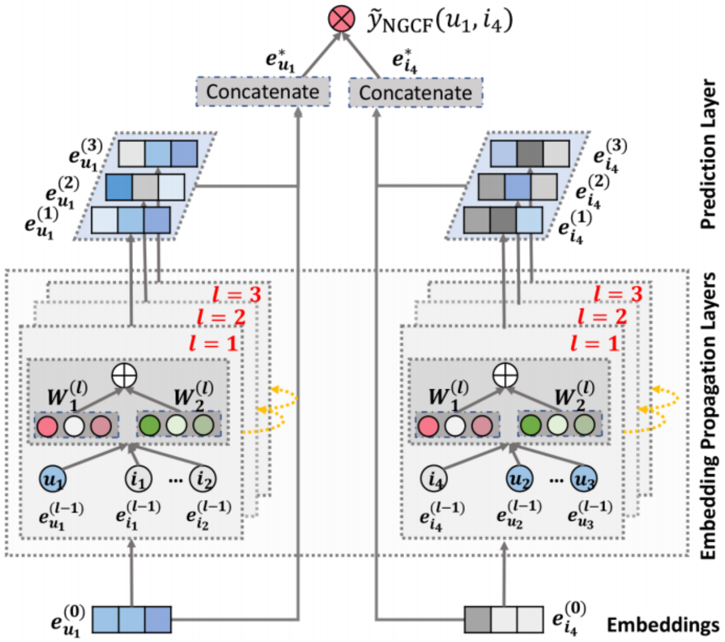
Graph-based NN models

Neighborhood aggregation: $z_u^{(l+1)} = \mathrm{AGG}(z_u^{(l)}, \{z_i^{(l)} : i \in \mathcal{N}_u\})$



Possible aggregations:
- weighted sum in GIN [Xu et al. 2018]
- LSTM aggregator in GraphSAGE [Hamilton et al 2017]
- bilinear interaction aggregator in BGNN [Zhu et al. 2020]
- …

$$z_u^{(l+1)} = z_u^{(l)} + \sum_{i \in N_u} \frac{1}{|\mathcal{N}_u|} z_i^{(l)} \qquad z_i^{(l+1)} = z_i^{(l)} + \sum_{u \in N_i} \frac{1}{|\mathcal{N}_i|} z_u^{(l)}$$



Pinterest is a famous adopter of GCN (PinSage/PinnerSage models):
- Paper: Graph Convolutional Neural Networks for Web-Scale Recommender Systems
- Blog: [PinSage: How Pinterest improved their recommendation system?]

Sequence-aware models

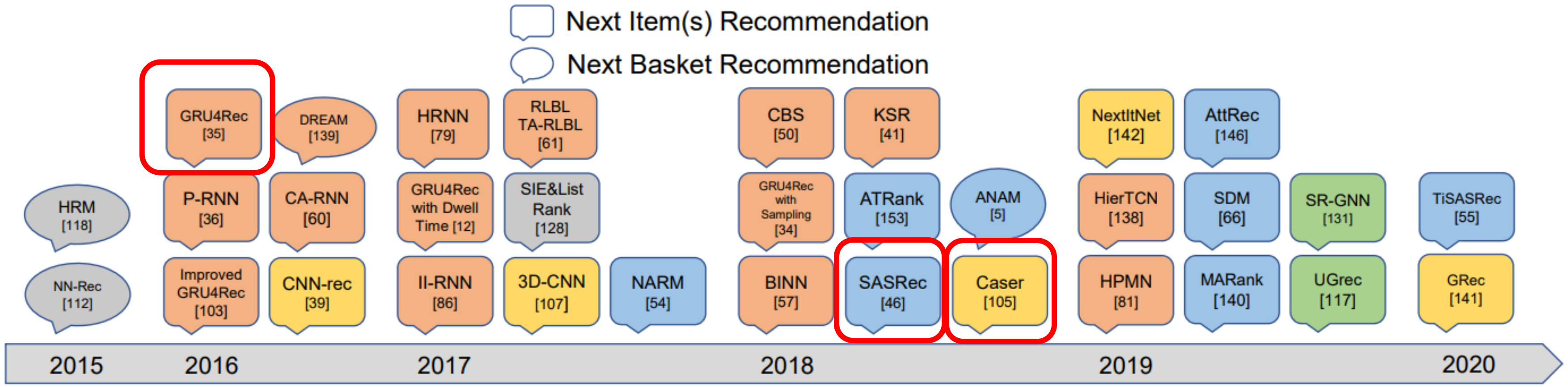
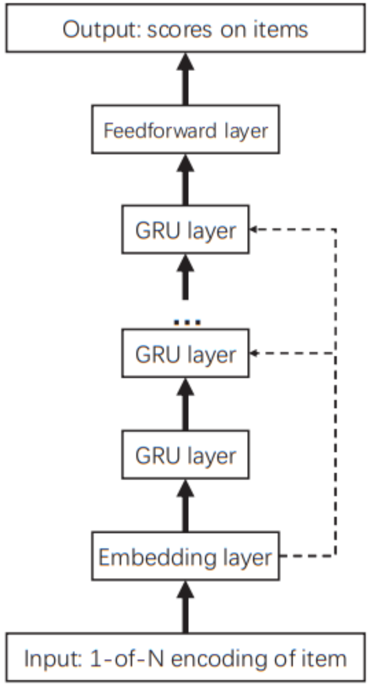


Fig. 8. Some recent and representative DL-based sequential recommendation models. Different colors indicate different DL techniques (grey: MLP; orange: RNN; yellow: CNN; blue: attention mechanism; green: GNN).

[Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations (arxiv.org)]

Sequence-aware models

Recurrent NN (GRURec)



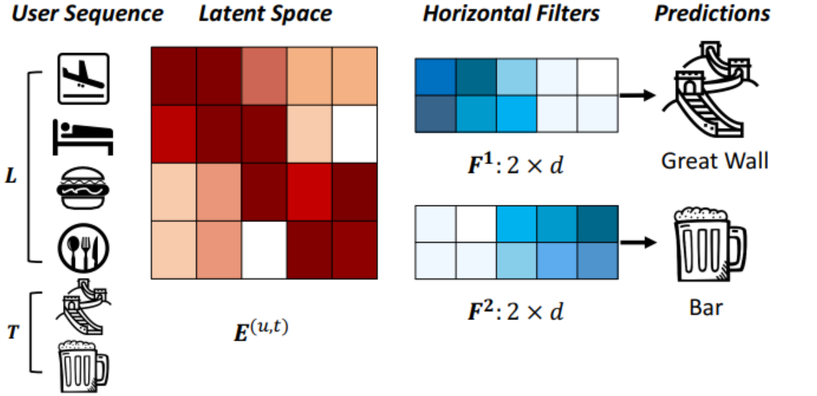
GRU: $\mathbf{h_t} = (1 - \mathbf{z_t})\mathbf{h_{t-1}} + \mathbf{z_t}\hat{\mathbf{h}}_\mathbf{t}$

$\mathbf{z_t} = \sigma(W_z\mathbf{x_t} + U_z\mathbf{h_{t-1}})$ as in standard RNN

$\hat{\mathbf{h}}_\mathbf{t} = \tanh\left(W\mathbf{x_t} + U(\mathbf{r_t} \odot \mathbf{h_{t-1}})\right)$

$\mathbf{r_t} = \sigma(W_r\mathbf{x_t} + U_r\mathbf{h_{t-1}})$

Convolutional NN (Caser)

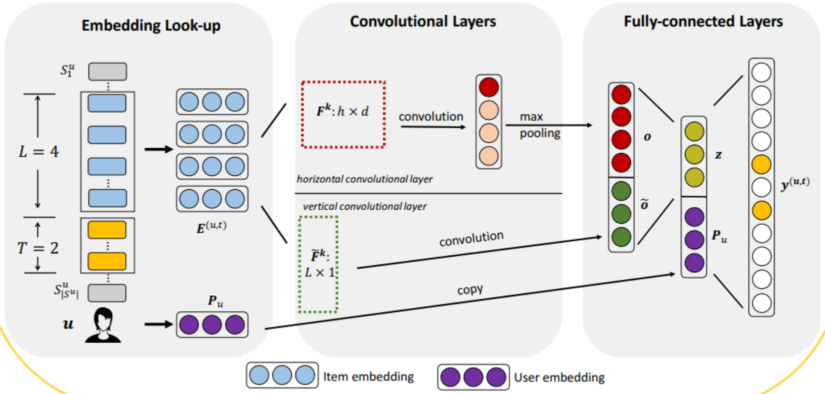


$$E^{(u,t)} = \begin{bmatrix} Q_{S^u_{t-L}} \\ \vdots \\ Q_{S^u_{t-2}} \\ Q_{S^u_{t-1}} \end{bmatrix}$$

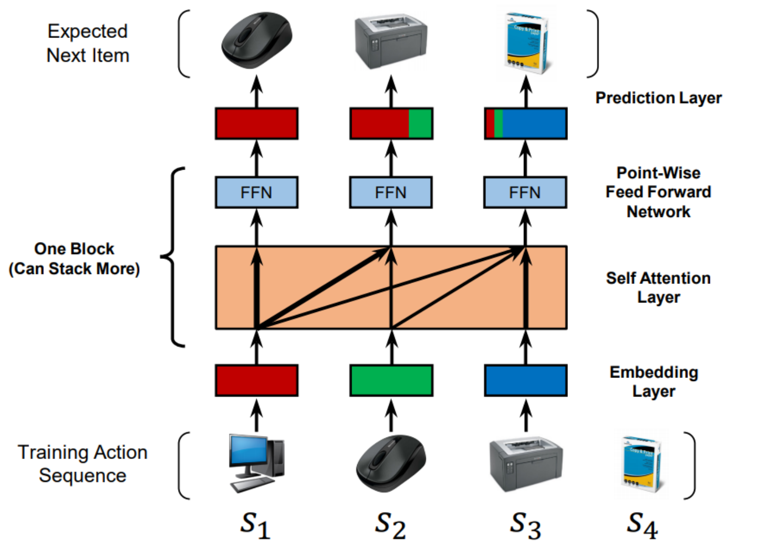
Horizontal convolution:

$$\mathbf{c}^k = \left[\mathbf{c}^k_1 \; \mathbf{c}^k_2 \cdots \mathbf{c}^k_{L-h+1} \right]$$

$$\mathbf{c}^k_i = \phi_c(\boldsymbol{E}_{i:i+h-1} \odot \boldsymbol{F}^k)$$



Self-Attention (SASRec)



$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}$$

$$\text{SA}(\widehat{\mathbf{E}}) = \text{Attention}(\widehat{\mathbf{E}}\mathbf{W}^Q, \widehat{\mathbf{E}}\mathbf{W}^K, \widehat{\mathbf{E}}\mathbf{W}^V)$$

$$\widehat{\mathbf{E}} = \begin{bmatrix} \mathbf{M}_{s_1} + \mathbf{P}_1 \\ \mathbf{M}_{s_2} + \mathbf{P}_2 \\ \dots \\ \mathbf{M}_{s_n} + \mathbf{P}_n \end{bmatrix}$$

interactions between Q_i and K_j for $j > i$ **are forbidden**

Why recsys is different from NLP

- BERT for NLP:
  - vocabulary size is 30K x 1024,
  - compute makes up almost the entire workload.
- Transformer-based recsys:
  - equivalent ‘vocabulary’ is 300M users and 12M items with dimension 64
  - don’t fit on a single GPU, heavily IO-bound.

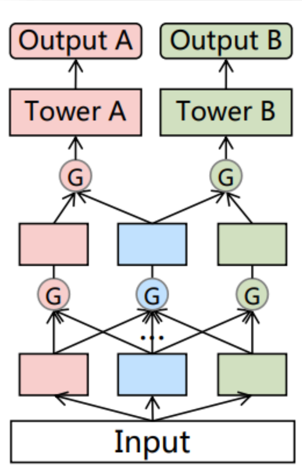
…inference for recommender systems in production still happens on CPU because GPUs don’t offer the same speedups that we see in other domains out of the box…

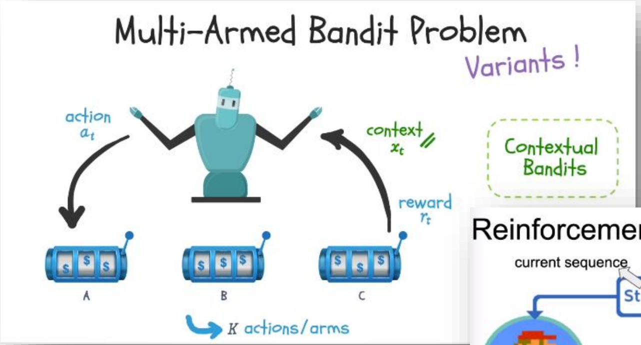
Even Oldridge, research scientist at NVidia

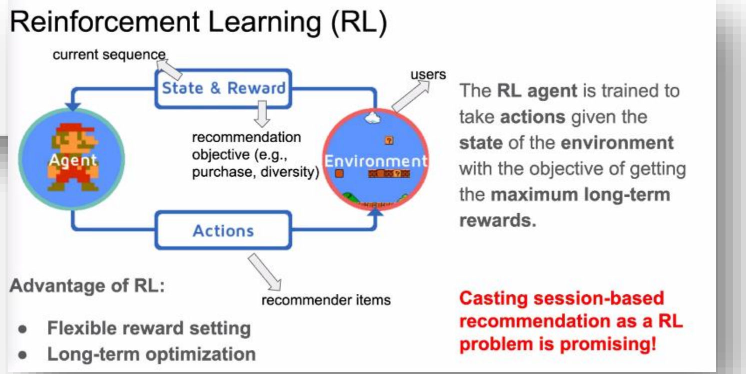
Source: Why isn’t your recommender system training faster on GPU?

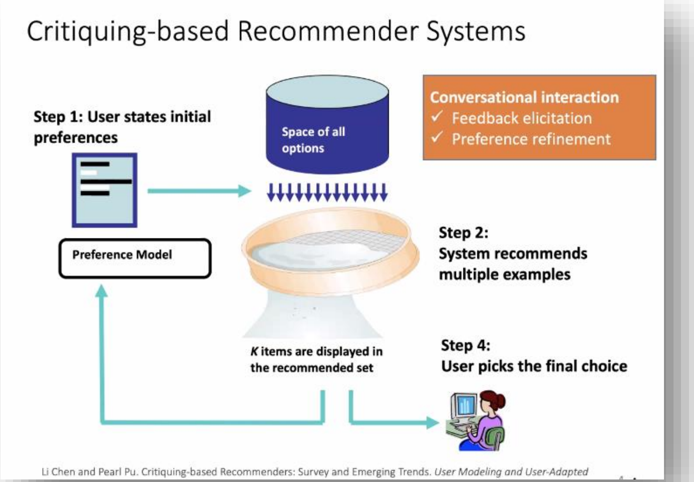
Current trends

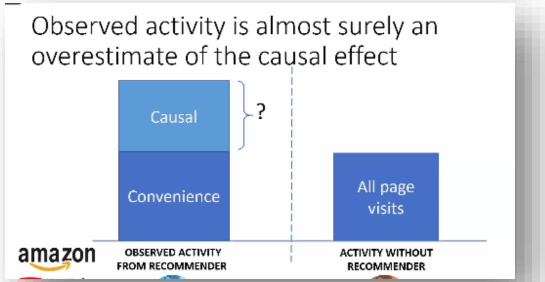
- Multi-task learning (RecSys 2020 best paper)
- Causality
- Fairness and debiasing
- Reinforcement Learning
- Conversational Recommenders, critiquing











Time for questions

Contact:
evgeny.frolov@skoltech.ru