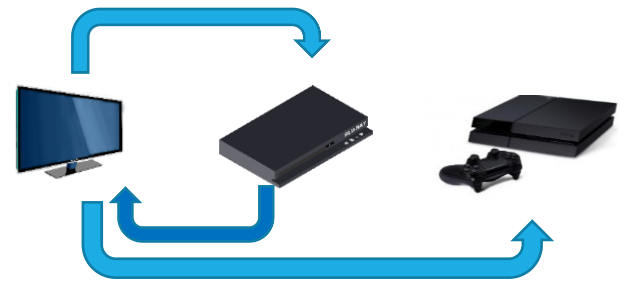
Recommender Systems

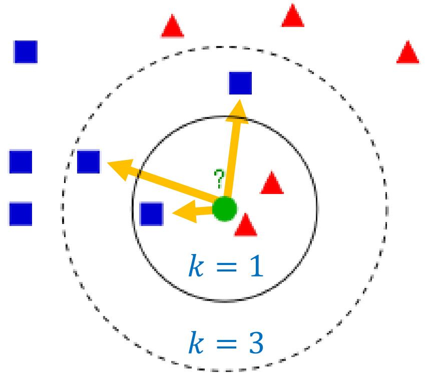
Lecture 5

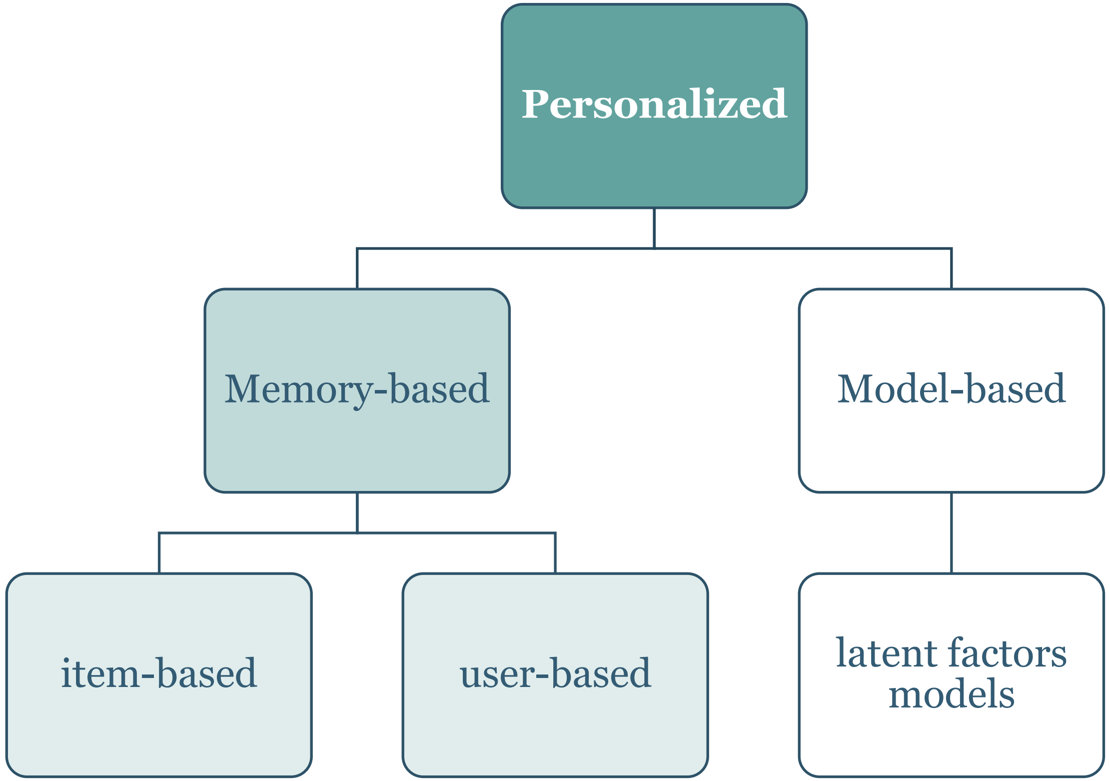
Previous lecture

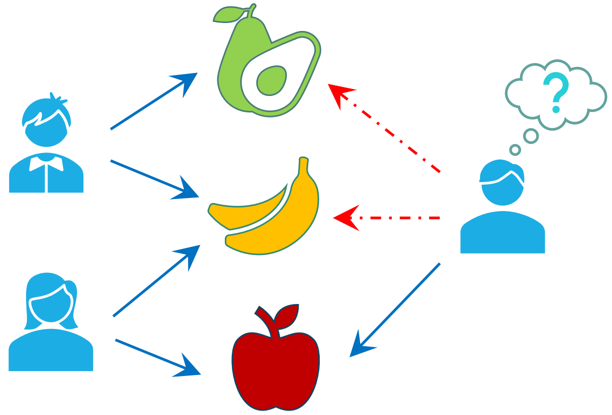
Association Rules



kNN







Previous lecture: neighborhood formation

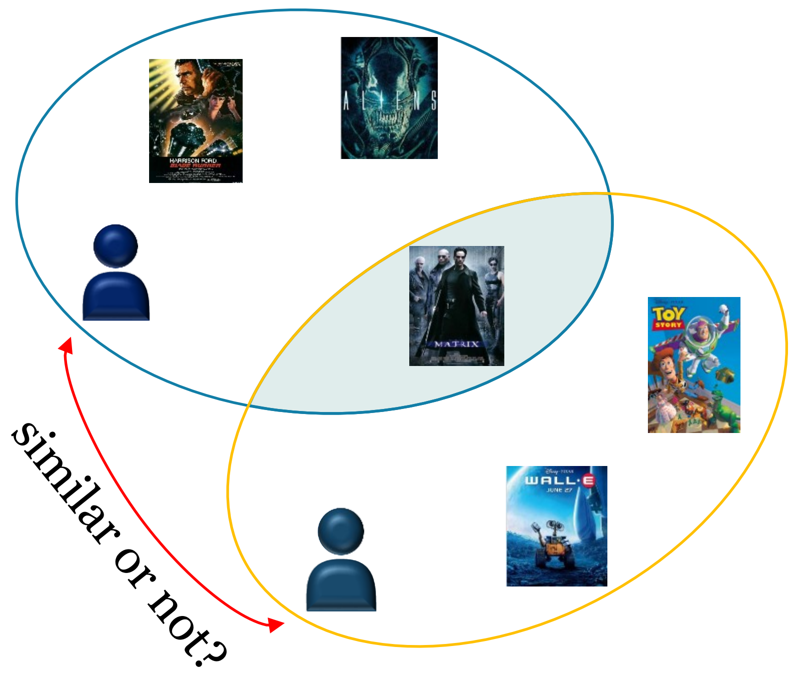
1. sample w.r.t. observed ratings: $\mathcal{N}_i(u)$ or $\mathcal{N}_u(i)$
2. sample n entities, s.t. k $\ll n \ll N$
   - randomly
   - by recency
3. select top-k most similar
   - what similarity?

Choosing between user-based and item-based:

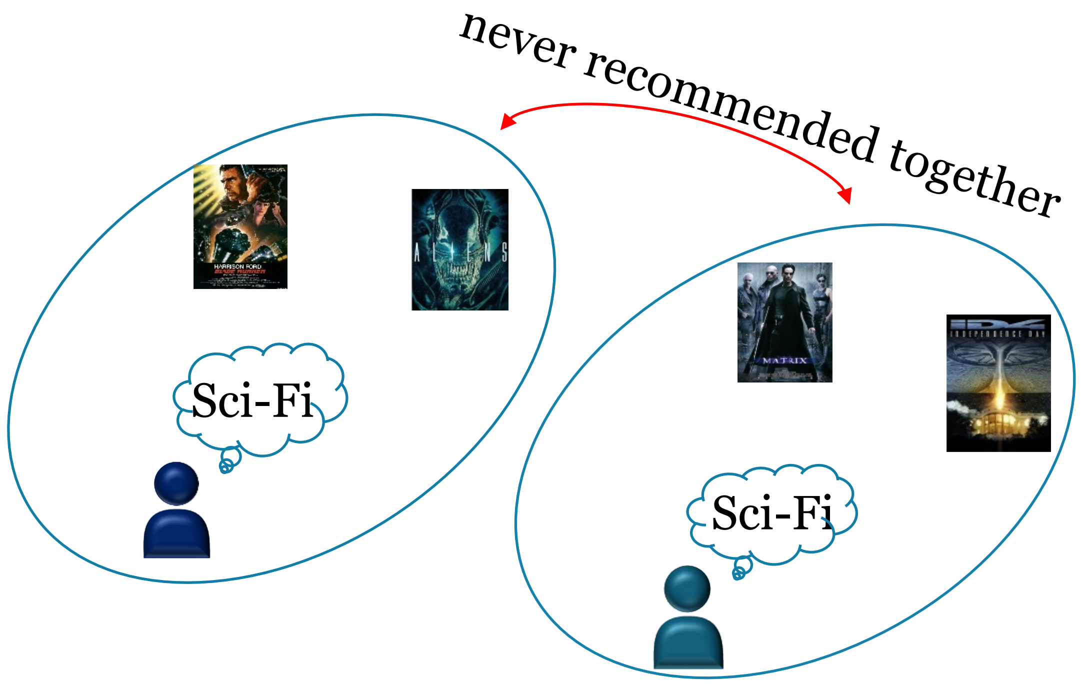
- # users vs. # items
- system dynamics
- explainability vs. serendipity

Previous lecture: limited coverage problems

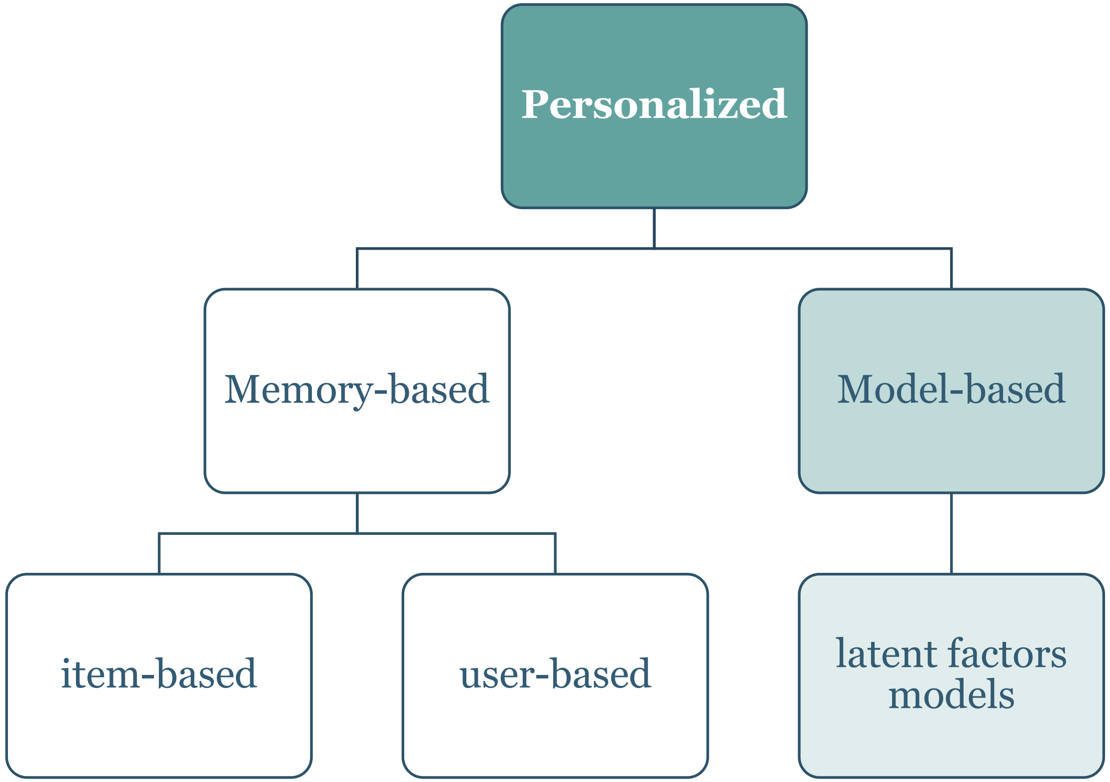
Unreliable correlations



Weak generalization



Today's Lecture



- Low-rank approximation for CF
  - PureSVD
  - Recommendation vs matrix completion
- Revisiting popularity bias

Low rank representation

?	3	5	5
4	?	5	5

4	3	?		
?	3	5		
4	?	5		
			4	5
			?	5

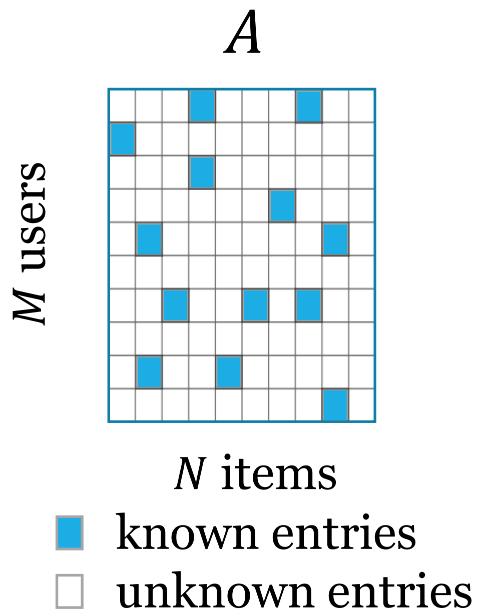
A general view on latent factors models

- **Task**: find utility (relevance) function f_R :

$$f_R\colon \text{Users} \times \text{Items} \to \text{Relevance score}$$

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

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



Components of the model:

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

What is the form of R and $\mathcal{L}$?

Intuition behind MF

Assumption: observed interactions can be explained via

- a *small* number of common patterns in human behavior
- + individual variations (including random factors and "unknown unknowns")

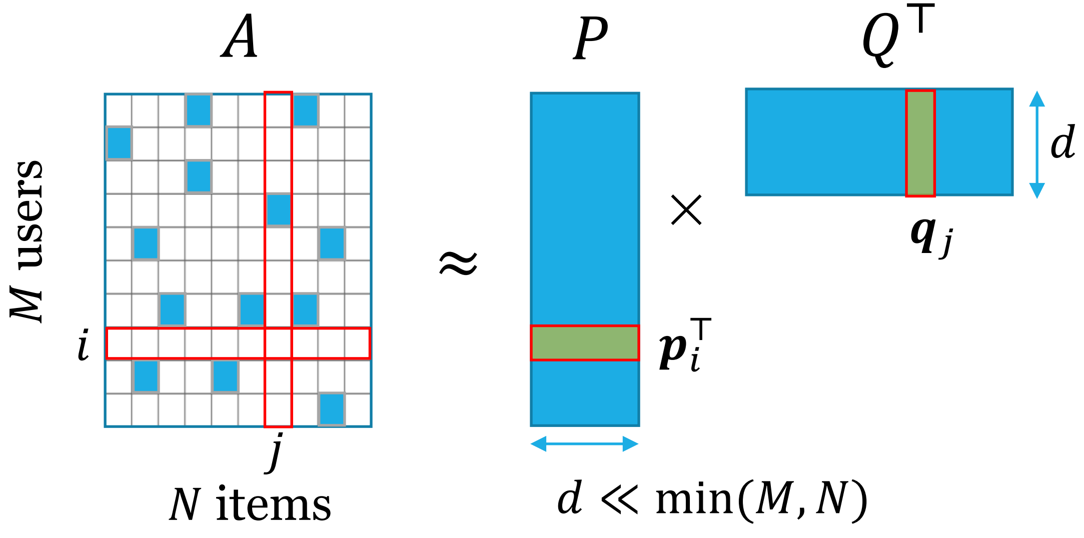
$$A_{full} = R + E, \qquad R = PQ^\top$$

Predicted utility of item j for user i:

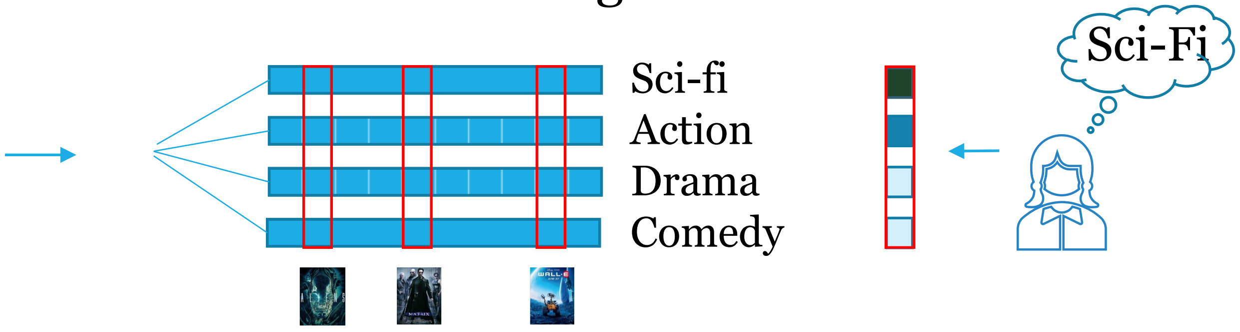
$$r_{ij} \approx \boldsymbol{p}_i^\top \boldsymbol{q}_j = \sum_{k=1}^{d} p_{ik} q_{jk}$$

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

Simplistic view on latent features



latent features $\leftrightarrow$ genres



How to find P and Q ?

rows of P – user embeddings
rows of Q – item embeddings

Singular Value Decomposition

Quick reminder:

$$A = U\Sigma V^\top$$

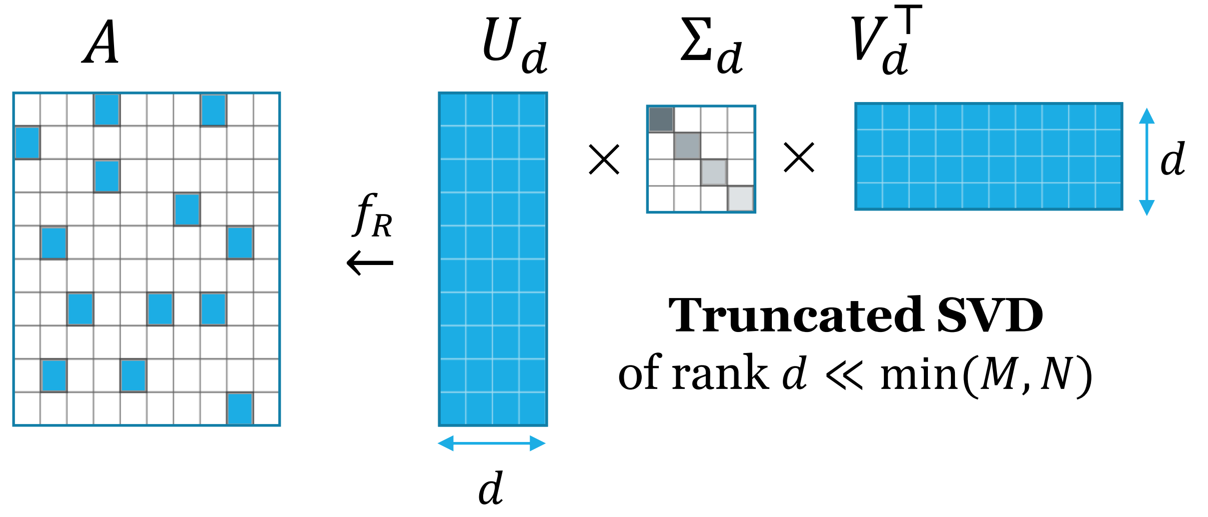
$$U \in \mathbb{R}^{M\times M}, \qquad V \in \mathbb{R}^{N\times N}$$

$$U^\top U = I_M, \qquad V^\top V = I_N$$

$\Sigma \in \mathbb{R}^{M\times N}$ - diagonal, with $[\Sigma]_{kk} = \sigma_k$:

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(M,N)} \geq 0$$

$$\sigma_k(A) = \sqrt{\lambda_k(A^\top A)} = \sqrt{\lambda_k(AA^\top)}$$



Truncated SVD
of rank $d \ll \min(M,N)$

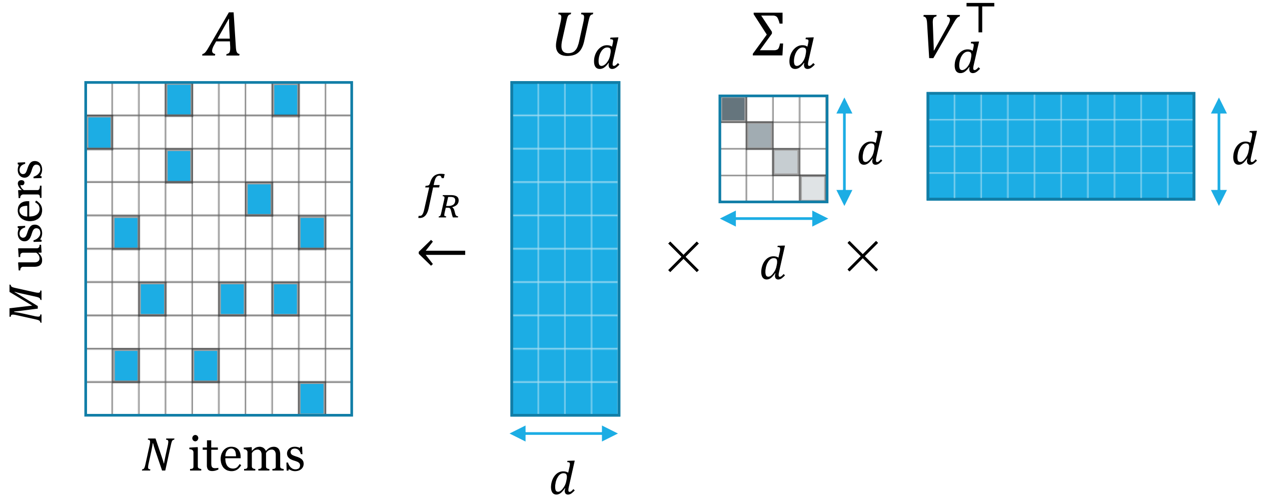
Low-rank approximation task:

$$\|A - R\|_F^2 \to \min, \;\; \text{s.t. } \operatorname{rank}(R) = d$$

$$R = U_d\Sigma_d V_d^\top, \quad \|A - R\|_F^2 =$$

Is it directly applicable here?

PureSVD model for CF



Relevance score prediction:

$$A_0 V_d V_d^\top =$$

Let’s impute zeros in place of unknowns!

$$A_0 = U \Sigma V^\top, \qquad [A_0]_{ij} = \begin{cases} a_{ij}, & \text{if known} \\ 0, & \text{otherwise} \end{cases}$$

$$R = U_d \Sigma_d V_d^\top$$

Explaining recommendations

Which one is more explainable?

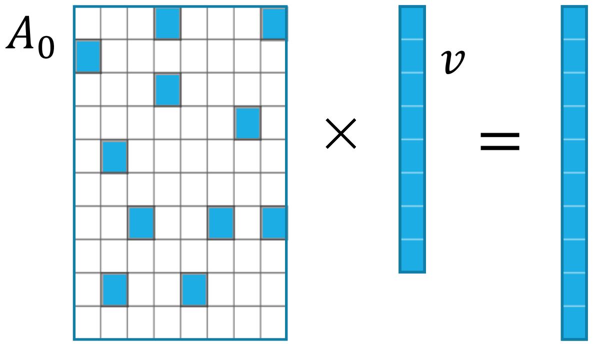
$$\boldsymbol{r} = Q\boldsymbol{p} \qquad \text{vs.} \qquad \boldsymbol{r} = VV^\top\boldsymbol{a}$$

Which model does PureSVD resemble?

PureSVD computation

- Efficient computation with Lanczos algorithm
  - iterative process
  - requires only sparse matvec (fast with CSR format)
  - complexity $O(nnz \cdot d) + O\big((M+N) \cdot d^2\big)$

    nnz – number of non-zeros of A_0



- Efficient implementations in Python:
  - SciPy Sparse svds,
  - Scikit-Learn TruncatedSVD.
- Core functionality is also implemented in Spark.

Billion-scale computations with SVD

In practice, in distributed setups, randomized SVD is used.

Examples:

- Criteo* https://github.com/criteo/Spark-RSVD
- Facebook's randomized SVD implementation https://research.fb.com/fast-randomized-svd

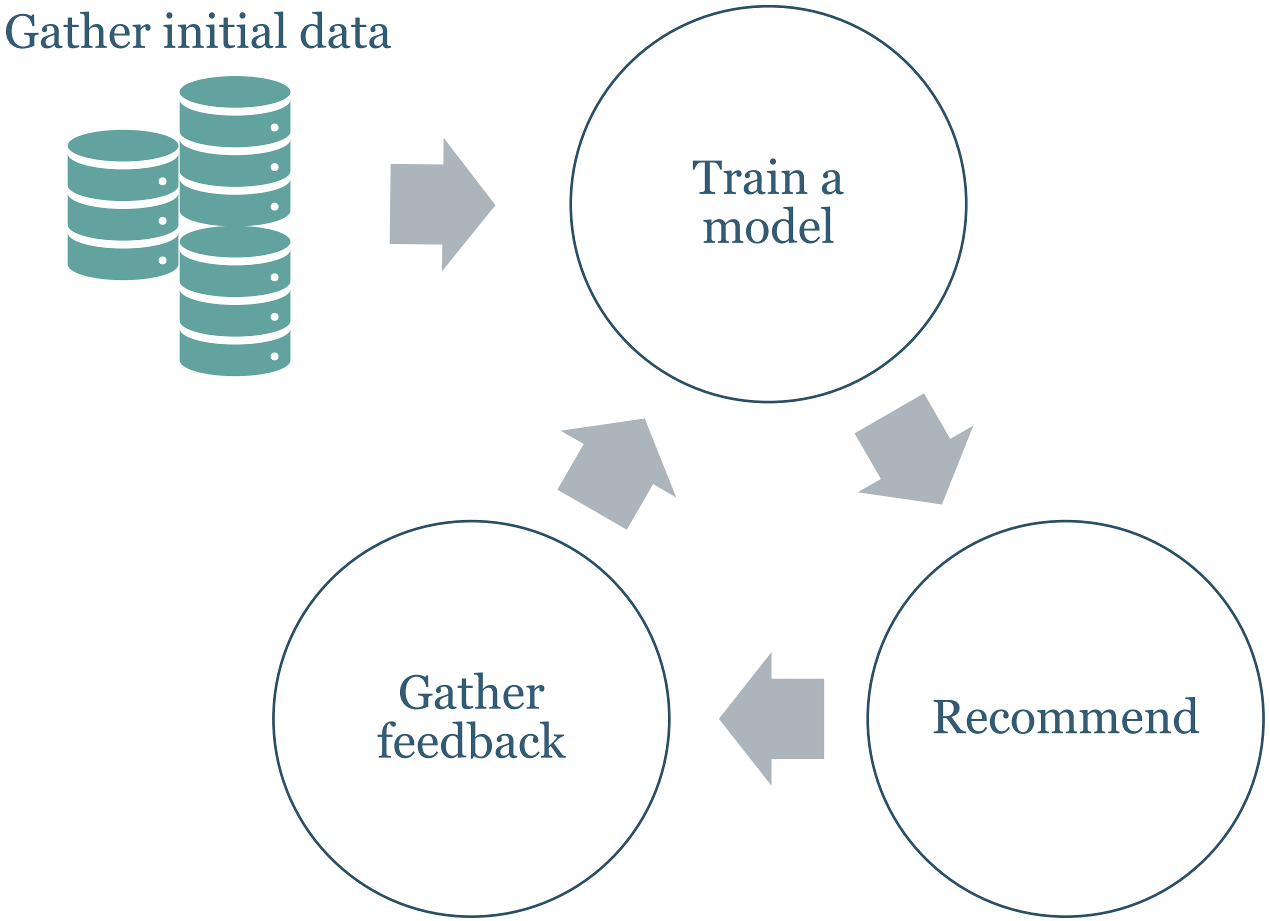
Research:

- “out-of-memory” SVD [Kabir 2017]
- communication-avoiding algebra [Demmel 2008]
- DeepMind’s attempt to adapt to modern hardware (GPU, TPU) via game-theoretic approach https://www.deepmind.com/blog/game-theory-as-an-engine-for-large-scale-data-analysis

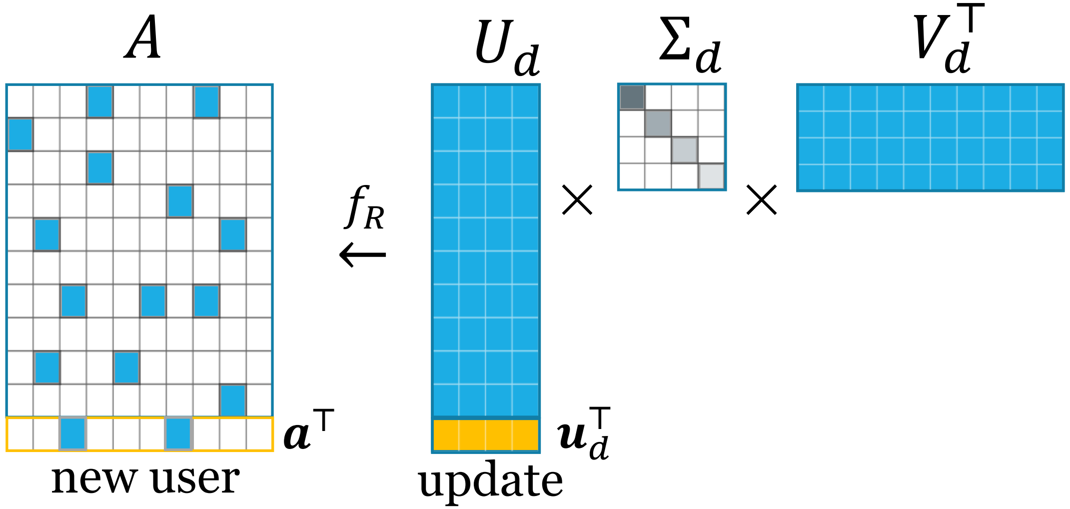
Generate random matrix $\Omega \in \mathbb{R}^{n\times(k+p)}$
$Y \leftarrow A\Omega$
$Q \leftarrow \mathrm{QR}(Y)$ ▷ QR decomposition of Y
for $i \leftarrow 1$ to q **do**
  $Y \leftarrow A^T Q$
  $Q \leftarrow \mathrm{QR}(Y)$
  $Y \leftarrow AQ$
  $Q \leftarrow \mathrm{QR}(Y)$
end for
$B \leftarrow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

*Read more: SparkRSVD open-sourced by Criteo for large scale recommendation engines

Lifecycle of a recsys model



PureSVD – recommending online



*folding-in technique**

Finding a warm-start user representation:

$$\left\| \boldsymbol{a}_0^\top - \boldsymbol{u}^\top \Sigma V^\top \right\|_2^2 \to \min$$

new user embedding

$$\boldsymbol{u}^\top = \boldsymbol{a}_0^\top V \Sigma^{-1}$$

Prediction:

$$\boldsymbol{r}^\top = \boldsymbol{u}_d^\top \Sigma_d V_d^\top = \boldsymbol{a}_0^\top V_d V_d^\top$$

$$\boldsymbol{r} = V_d V_d^\top \boldsymbol{a}_0$$

- convenient for evaluation
- complexity $\sim O(Nd)$
- enables real-time recommendations

*G. Furnas, S. Deerwester, and S. Dumais, “Information Retrieval Using a Singular Value Decomposition Model of Latent Semantic Structure,” Proceedings of ACM SIGIR Conference, 1988

On-stream and incremental learning

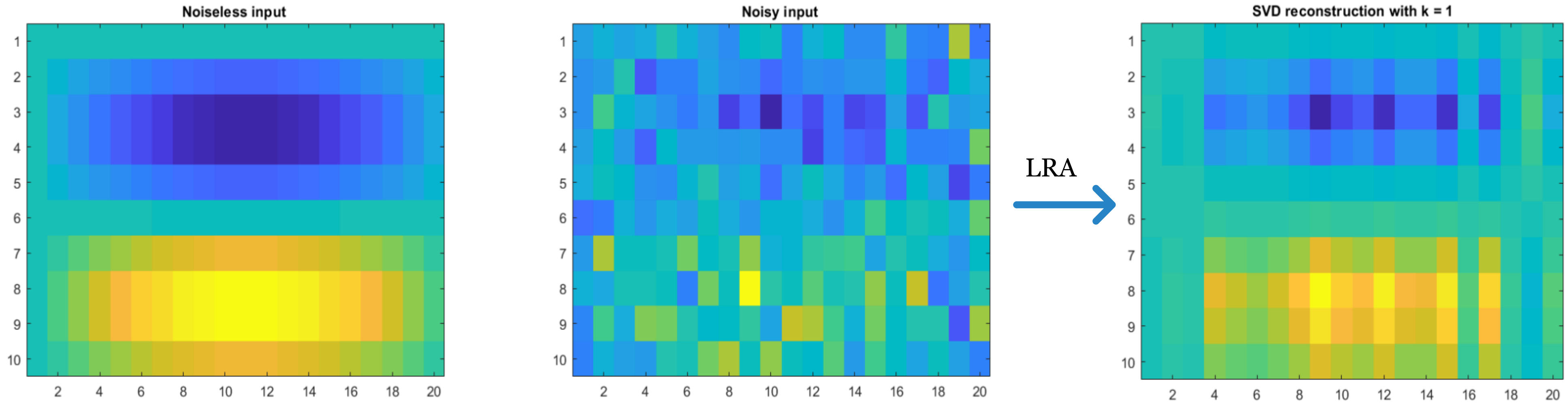
Incremental learning:

- *Adding new users/items:*
  - rank-1 updates (see G. Golub, C. Van Loan, “Matrix Computations”)
  - M. Brand "Incremental singular value decomposition of uncertain data with missing values.", 2002.

Adding new interactions:

- *via projector splitting approach* [Lubich & Oseledets 2013]:
  - example in recsys: [Olaleke et al. 2021]
- *streaming:*
  - Method of frequent directions [Ghashami et al. 2016]
  - Zoom SVD [Jang et al. 2018]

CF as low-rank approximation task



unknown in reality

Images source: Khoshrou, Abdolrahman, and Eric J. Pauwels. "Regularisation for PCA-and SVD-type matrix factorisations." *arXiv preprint arXiv:2106.12955* (2021).

Approximation with irreducible noise

$$A = \boldsymbol{e} \cdot \boldsymbol{a}^\top + \epsilon B$$

$$\sigma_1^2(A) \leq M\|\boldsymbol{a}\|^2 + \epsilon^2 N, \qquad \sigma_k^2(A) \approx \epsilon^2 N, k \gg 1$$

?	3	5	5
4	?	5	5

Practical consequences

- larger # of items – harder pattern discovery task

- even in the simplest case singular values won't become 0
  - let's check it!

- no simple choice of the optimal rank of the decomposition
  - $\left\|A - U_d \Sigma_d V_d^{\top}\right\|_{\mathrm{F}} = \sqrt{\sigma_{d+1}^2 + \cdots + \sigma_{\min(M,N)}^2}$

- doesn't mean you can't get close to zero RMSE on trainset

Data centering (PCA style)

Observations:

- *today*: in PureSVD, values are highly biased towards 0
- *previously*: a signal is carried mostly by baseline estimators

How does it affect rating prediction?

Strategy:

- value imputation → mean shifted matrix
- akin to data centering in PCA

Spectrum of mean-shifted ratings matrix

$$A = \hat{A}_0 + \alpha \boldsymbol{e}_M \boldsymbol{e}_N^\top$$

$$\sigma_1^2(A) = \sigma_1^2\left(\hat{A}_0\right) + \alpha^2 MN, \qquad \sigma_k^2(A) \approx \sigma_1^2\left(\hat{A}_0\right), k \gg 1$$

Let's evaluate!

Note: we have a dense (and potentially huge) matrix

$$A = \hat{A}_0 + \alpha\, \boldsymbol{e}_M \boldsymbol{e}_N^\top$$

Example: $M = 1\_000\_000$ users and $N = 100\_000$ items would require $\approx$ 745 Gb of RAM

How to avoid explicitly forming it?

Practical consequences

- SVD for CF is:
  - NOT pure matrix completion
  - NOT pure dimensionality reduction

- common PCA-like preprocessing may spoil data representation

- rating prediction doesn’t make sense
  - recommendations can still be good!
  - we can treat rating values more flexibly

Mitigating popularity bias

Top singular vector and popularity bias

- We saw that for a simple rank-1 approximation, top-singular value is driven by the common row of ratings
$$A = \boldsymbol{e} \cdot \boldsymbol{a}^\top + \epsilon B$$

- More generally, top singular triplet is likely to capture signal mostly from popular items and active users

- Idea: remove it ☺

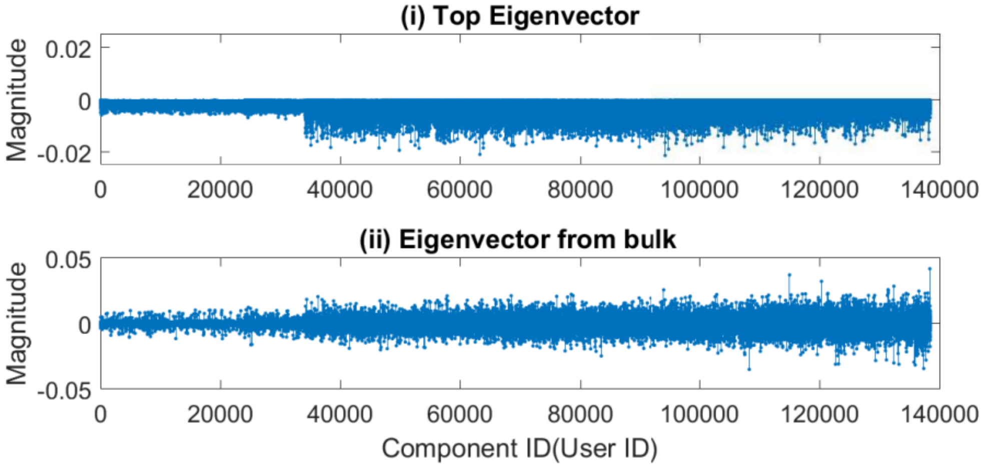


Image source: Khawar, Farhan, and Nevin L. Zhang. "Matrix Factorization Equals Efficient Co-occurrence Representation." *arXiv preprint arXiv:1808.09371* (2018).

The effect of removing top singular component

Method	NDCG@50	Recall@50	D@50	Time(min.)
(a)SVD($k = 20$)	0.60597	0.40434	1574	34.8
(b)SVD($k = 19$)	0.60168	0.40088	2139	35.4
(c)SVD($k = 1$)	0.42106	0.19704	290	20.8
SVD($k = 100$)	0.59912	0.37539	2368	88
WRMF($k = 20$, $\lambda = 10^{-3}$)	0.60678	0.40904	1861.6	214

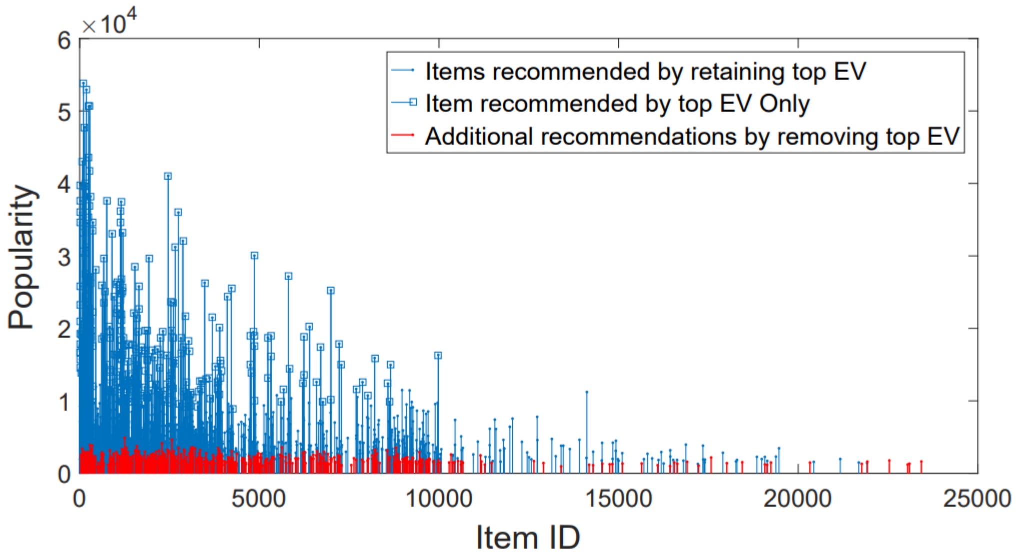


Figure 3: Removing v_H increases diversity by including non-popular items.

Results from: Khawar, Farhan, and Nevin L. Zhang. "Matrix Factorization Equals Efficient Co-occurrence Representation." *arXiv preprint arXiv:1808.09371* (2018).

Data normalization in PureSVD

- Common observation:
  - interactions data approximately follows power-law or zipf-like distributions

- What effect does it have on covariance matrix?
  - $\boldsymbol{a}_i^\top \boldsymbol{a}_j$

- Idea: normalization inversely proportional to popularity

$$\tilde{A} = AD^{f-1}, \qquad [D]_{ii} = \|\bar{\boldsymbol{a}}_i\|$$