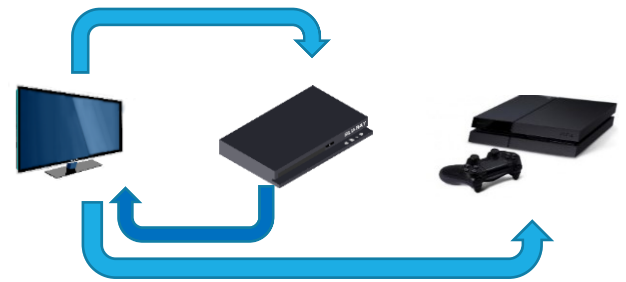
Recommender Systems

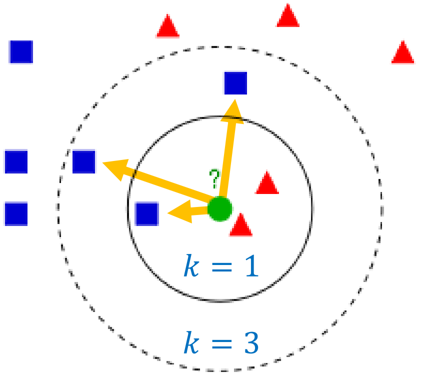
Lecture 5

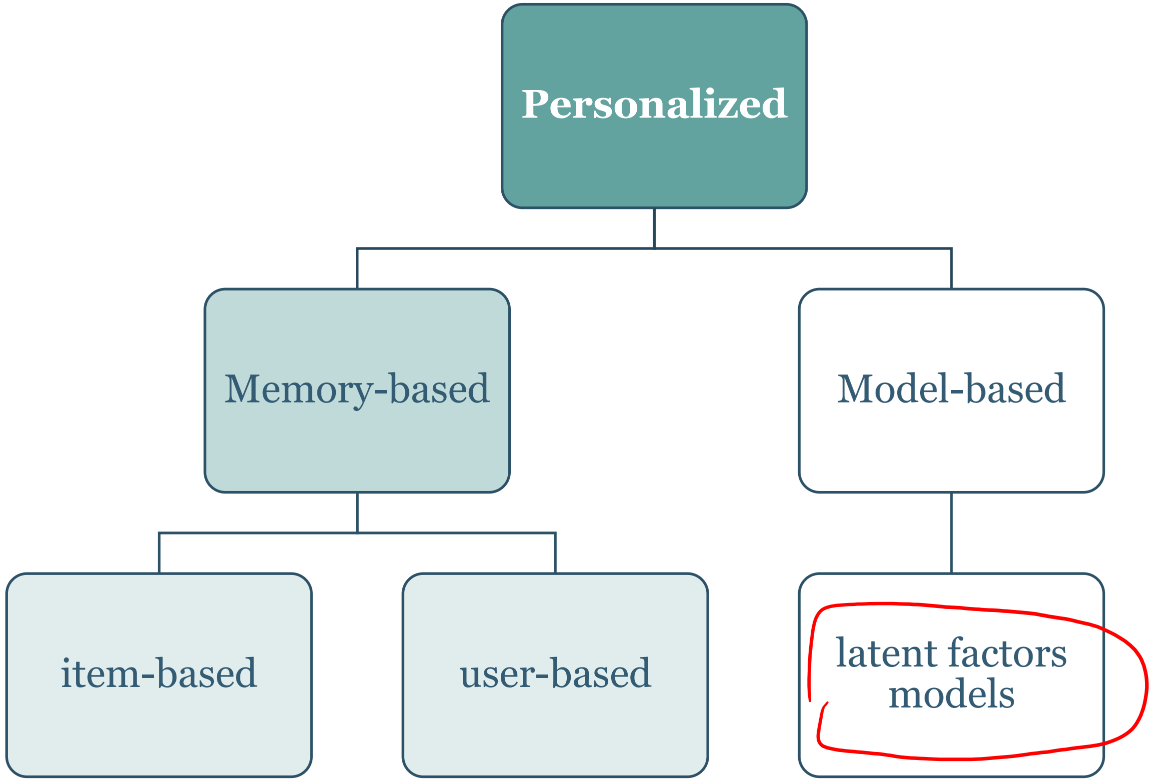
Previous lecture

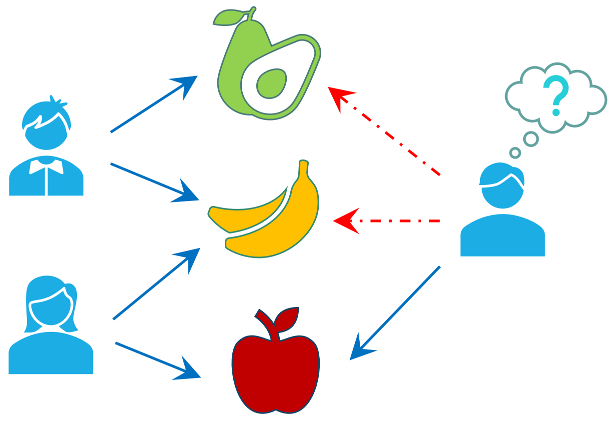
Association Rules



kNN







Previous lecture: neighborhood formation

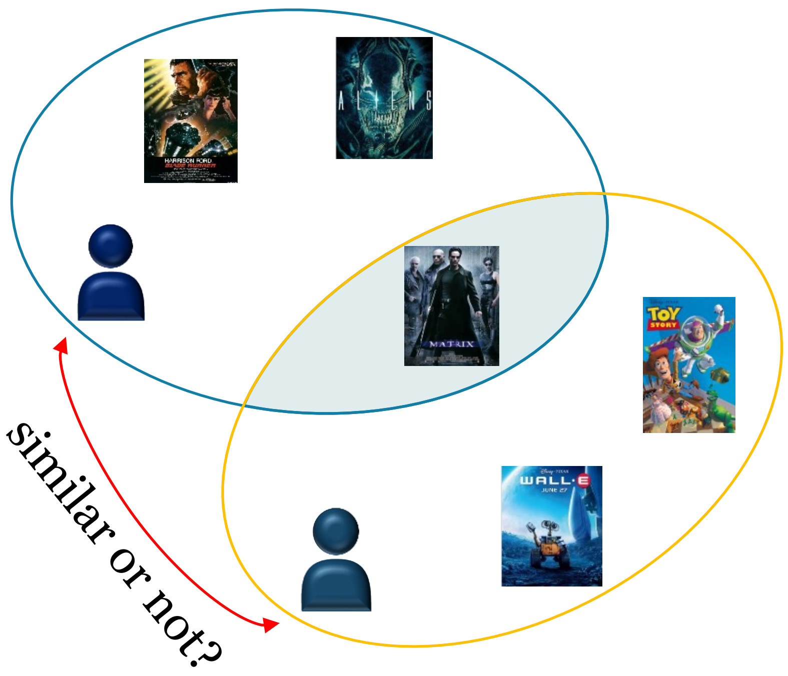
1. sample w.r.t. observed ratings: $\mathcal{N}_i(u)$ or $\mathcal{N}_u(i)$
2. sample n entities, s.t. k $\ll n \ll N$
    - randomly
    - by recency
3. select top-k most similar
    - what similarity?

Choosing between user-based and item-based:

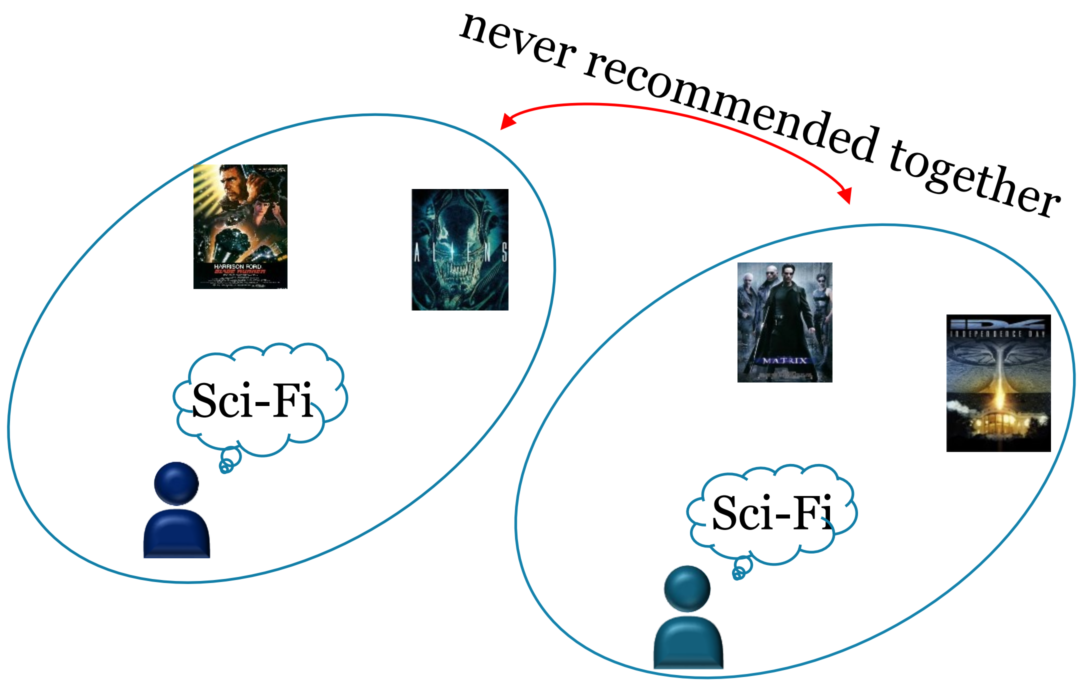
- # users vs. # items
- system dynamics
- explainability vs. serendipity

Previous lecture: limited coverage problems

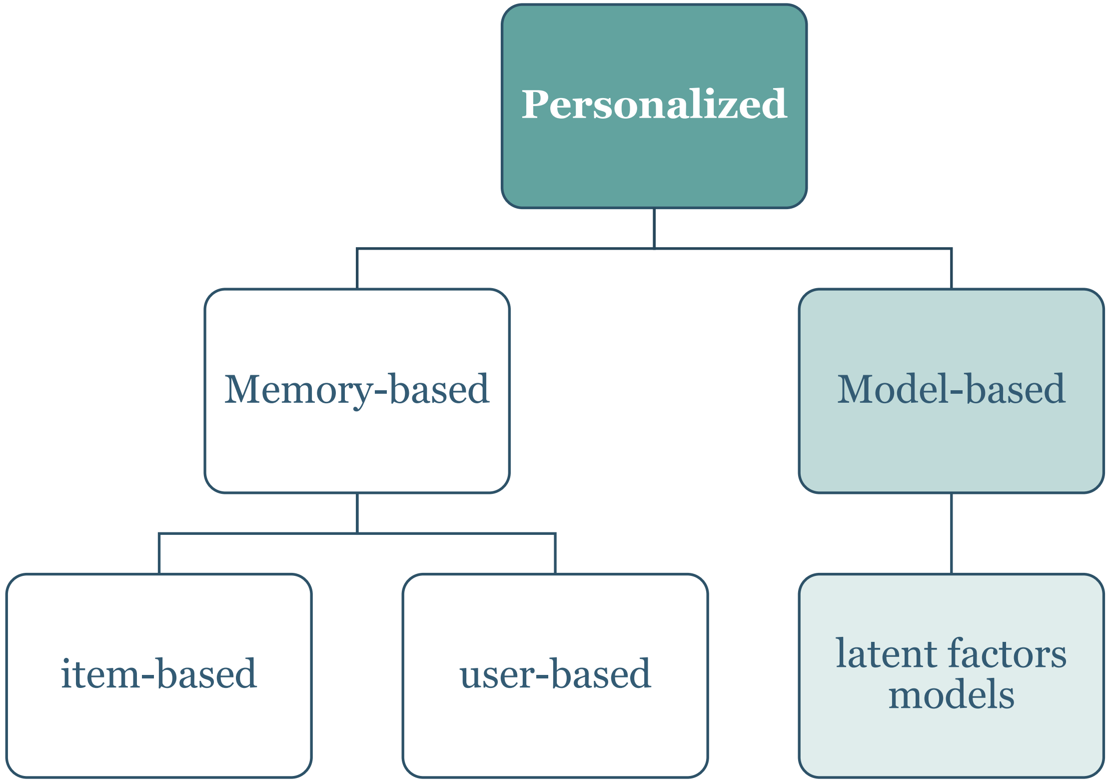
Unreliable correlations



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

$$A \approx PQ^T$$

$$A \approx R$$

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

$$A \approx e a^T \qquad e = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad a = \begin{bmatrix} 4 \\ 3 \\ 5 \\ 5 \end{bmatrix}$$

$$P = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

$$Q = \begin{bmatrix} 4 & 0 \\ 3 & 0 \\ 5 & 0 \\ 0 & 4 \\ 0 & 5 \end{bmatrix}$$

A general view on latent factors models

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

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

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

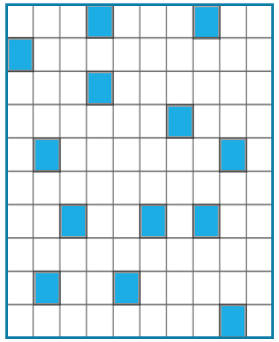
$$\mathcal{L}(A, R) \rightarrow \min$$

$R = PQ^T$

$r_{ij} = p_i^T q_j$

A

M users



N items

known entries

unknown entries

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")

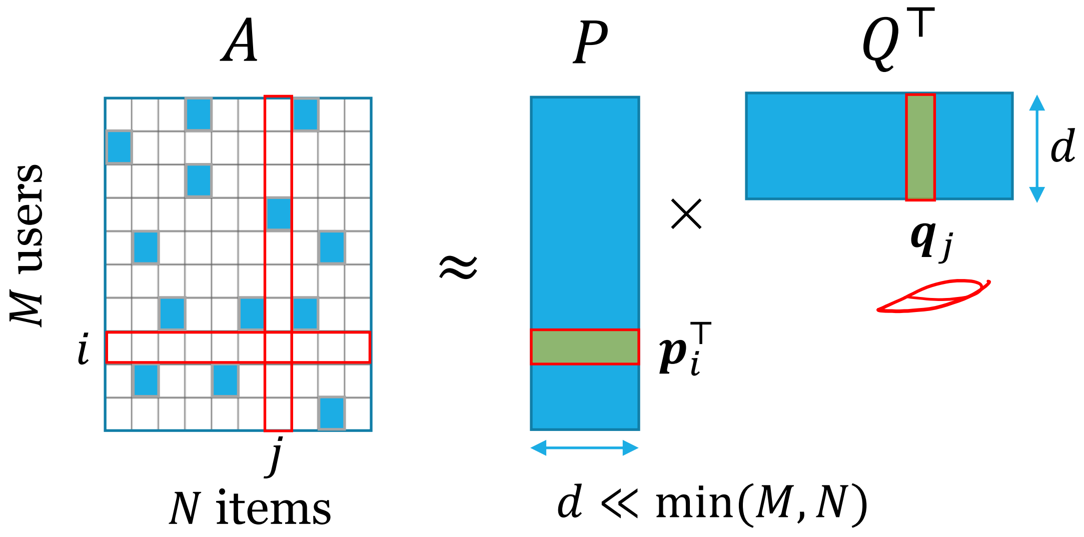
$$A_{full} = R + E, \qquad R = PQ^{\top}$$

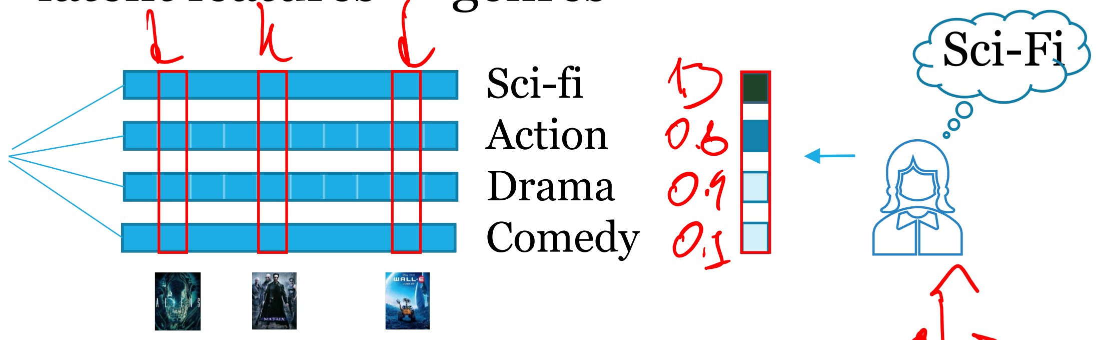
Predicted utility of item j for user i:

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

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

Simplistic view on latent features





How to find P and Q ?

$$\begin{bmatrix} 0 \\ 0 \\ 2.5 \\ 5 \end{bmatrix}$$

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

Singular Value Decomposition

Quick reminder:

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

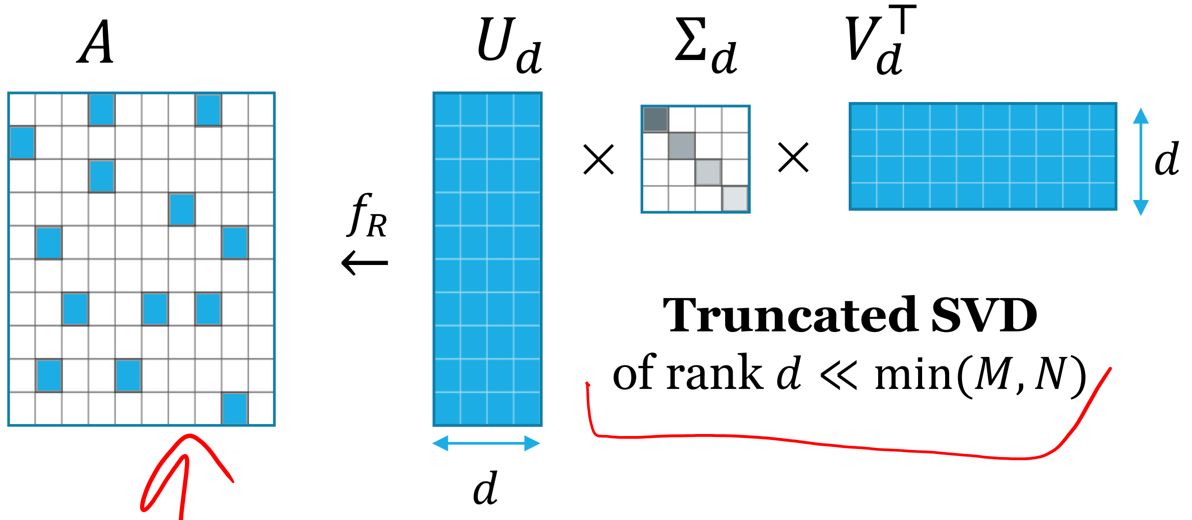
$$U \in \mathbb{R}^{M\times M}, \qquad V \in \mathbb{R}^{N\times N}$$

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

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

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

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



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

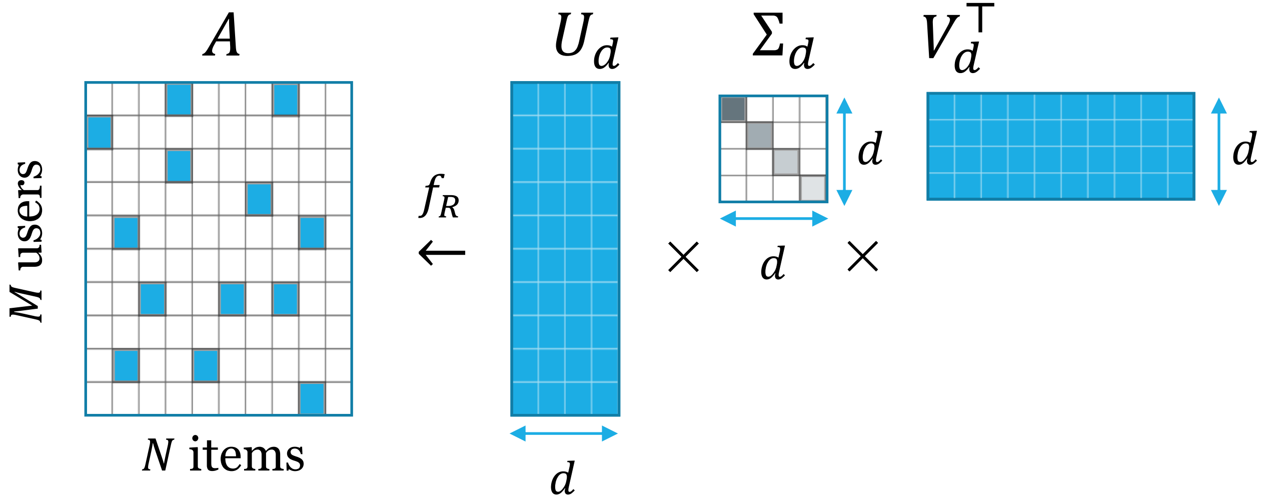
Low-rank approximation task:

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

$$R = U_d \Sigma_d V_d^\top, \quad \|A - R\|_F^2 = \sum_{i=d+1} \sigma_i^2 \, ?$$

Is it directly applicable here?

PureSVD model for CF



Let's impute zeros in place of unknowns!

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

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

Relevance score prediction:

$$A_0 V_d V_d^\top = U \Sigma V^\top V_d V_d^\top =$$

$$\Sigma = \begin{bmatrix} \Sigma_d & 0 \\ 0 & 0 \end{bmatrix} \qquad V = [V_d \; V_\perp]$$

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

$$V_d^\top V_d = I \qquad V_\perp^\top V_d = 0$$

$$R = A_0 V_d V_d^\top$$

$$= U_d U_d^\top A_0$$

$$RV = A_0 V = U\Sigma$$

Explaining recommendations

Which one is more explainable?

$R = PQ^\top$

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

$A = U\Sigma V^\top$

$V^\top a = \sum_{i \in N_u} v_i$

$v_j^\top \left(\sum_{i \in N_u} v_i\right) = \sum_{i \in N_u} v_j^\top v_i$

0.3 0.1 0.01

minmax

Which model does PureSVD resemble?

$r = Ca$

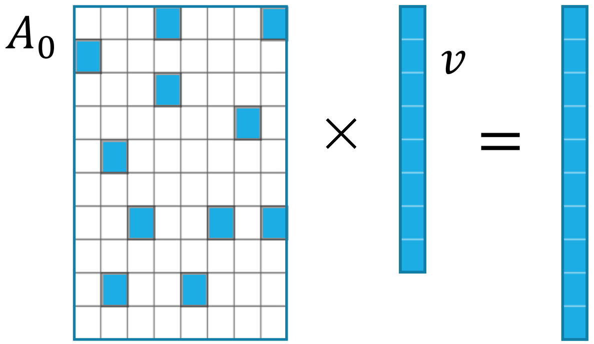
$C = A^\top A - \operatorname{diag}(\operatorname{diag} A^\top A)$

PureSVD computation

$Av \quad A^2v \ \cdots$

- 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

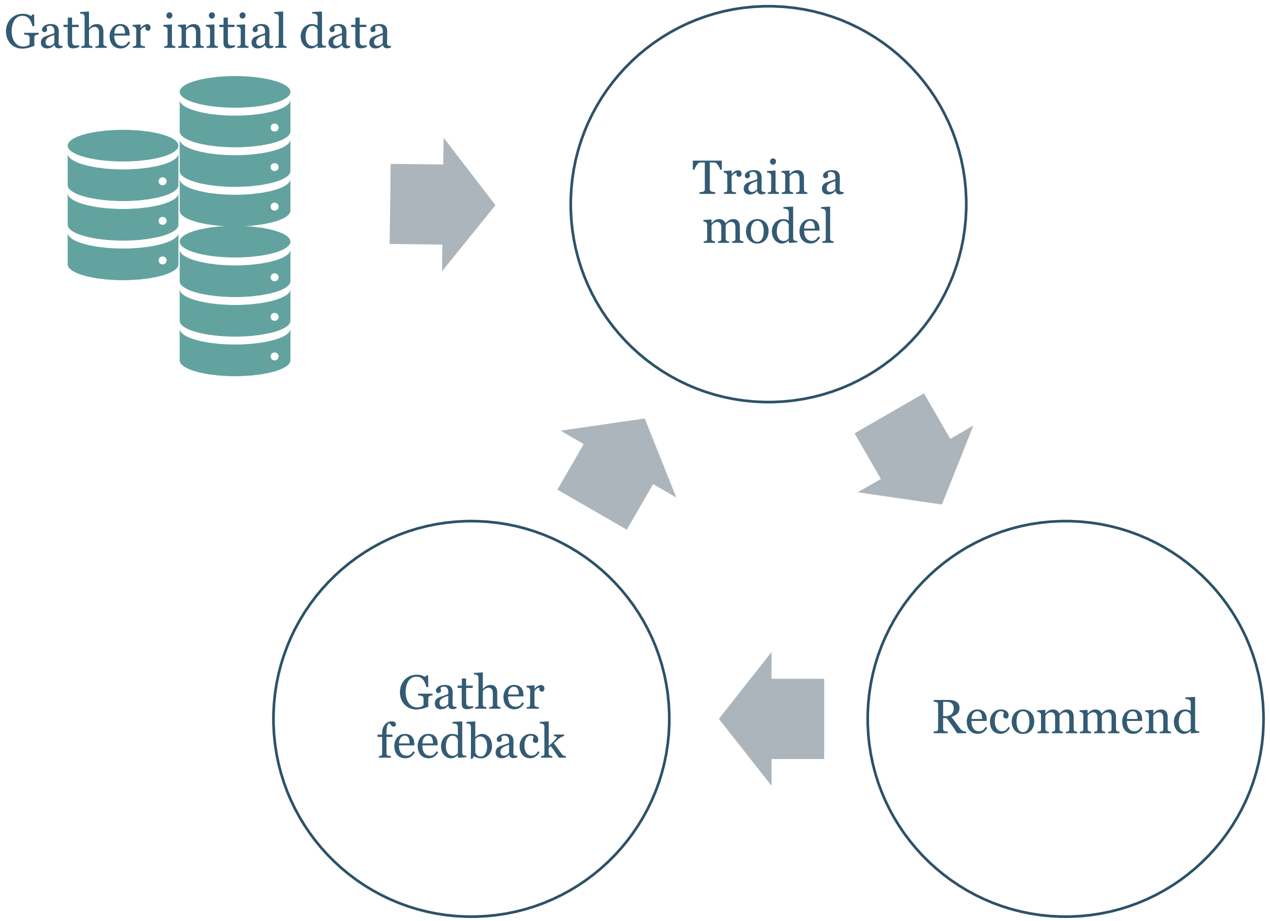
Research:

- "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

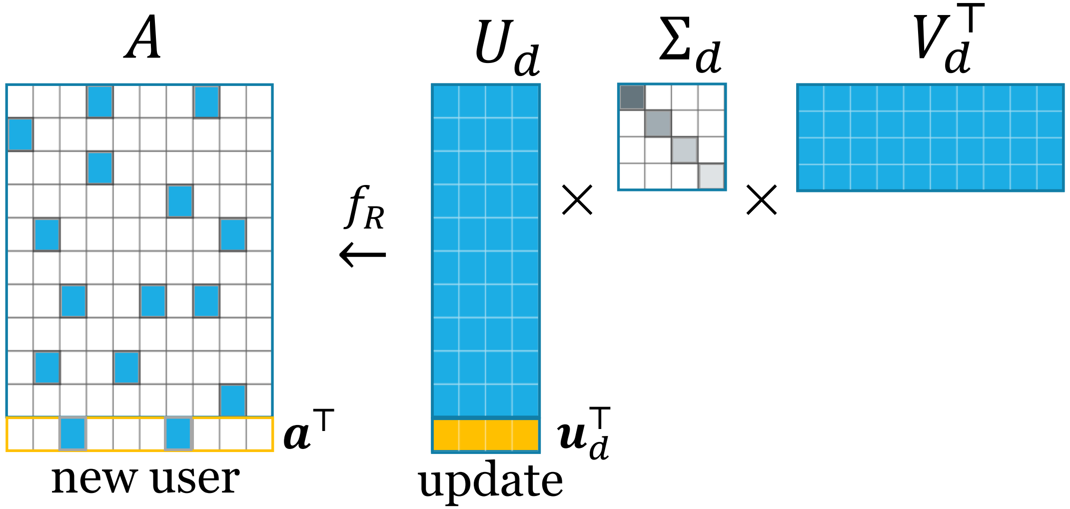
Generate random matrix $\Omega \in \mathbb{R}^{n\times(k+p)}$
$Y \leftarrow A\Omega$
$Q \leftarrow \mathrm{QR}(Y)$ $\triangleright$ QR decomposition of Y
for $i \leftarrow 1$ to q **do**
  $Y \leftarrow A^T Q$
  $Q \leftarrow \mathrm{QR}(Y)$
  $Y \leftarrow AQ$
  $Q \leftarrow \mathrm{QR}(Y)$
end for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

[*] 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.

$\|A - R\|_F^2$

$\|\dot{A} - \dot{R}\|_F^2$

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]

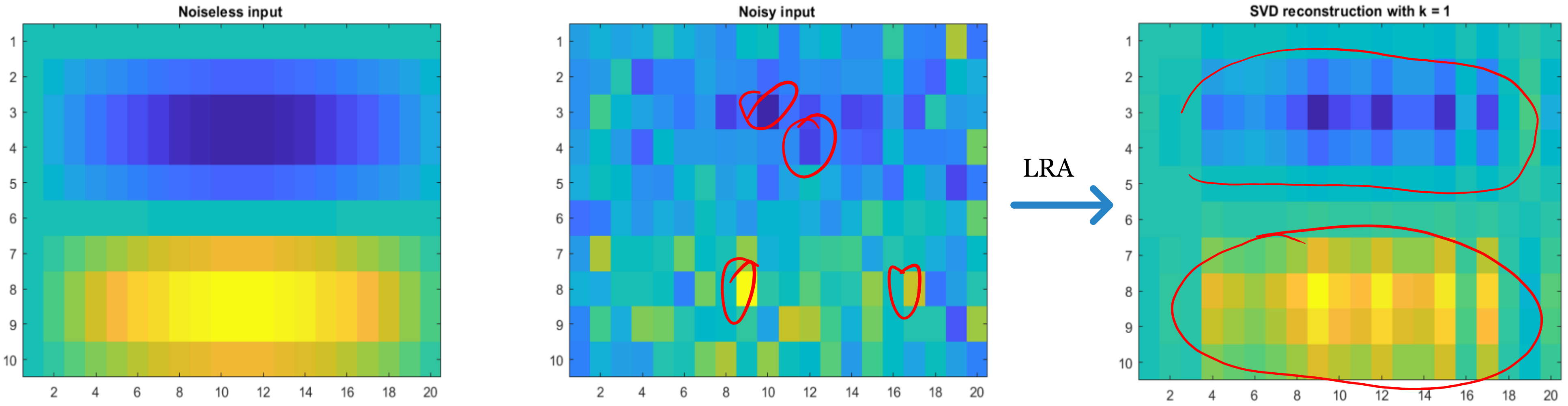
$A \to \frac{A + \Delta A}{\Delta t} \approx \frac{dA}{dt}$

$A(t) = \int_0 \dot{A}\, dt$

CF as low-rank approximation task

Horror

Romantic



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

$$\overset{R \;+\; E}{A = \boldsymbol{e}\cdot\boldsymbol{a}^\top + \epsilon B} \qquad [B]_{ij} \approx \mathcal{N}(0,1)$$

?	3	5	5
4	?	5	5

$$\sigma_1^2(A) \le M\|\boldsymbol{a}\|^2 + \epsilon^2 N, \qquad \sigma_k^2(A) \approx \epsilon^2 N, k \gg 1 \qquad Bx \approx 0$$

$$\sigma_1^2(A) = \lambda_1(AA^T) = \lambda_1\left(e a^T a e^T + \underbrace{\epsilon e a^T B^T + \epsilon B a e^T}_{} + \underbrace{\epsilon^2 BB^T}_{}\right)$$

$$[BB^T]_{ij} = \sum_k \underbrace{b_{ik} b_{jk}}_{} \approx N$$

$$b_{ik} b_{jk} \approx \begin{cases} 1, & i=j \\ 0, & i \neq j \end{cases}$$

$$= \lambda_1\left(\underbrace{\|a\|^2 e e^T}_{} + \epsilon^2 N I\right)$$

$$\lambda_1(G) = \max_{\|u\|=1} u^T G u$$

$$\|a\|^2 \underbrace{u^T e e^T u}_{} = \|a\| (u^T e)^2 \le \|a\|^2 \overset{1}{(u^T u)} \overset{M}{(e^T e)} = M\|a\|^2$$

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
  - $\|A - U_d \Sigma_d V_d^\top\|_F = \sqrt{\sigma_{d+1}^2 + \cdots + \sigma_{\min(M,N)}^2}$
- 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

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

$$[\hat{A}_0]_{ij} = \begin{cases} a_{ij} - \alpha, & ij \text{ known} \\ 0 & \text{otherwise} \end{cases}$$

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

$u_1 \quad u_2 \quad u_3 \qquad u_d$

Let's evaluate!

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

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

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

How to avoid explicitly forming it?

$$Av = \underbrace{\hat{A}_0 v}_{O(M)} + \underbrace{\alpha\, e_M (e_N^\top v)}$$

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