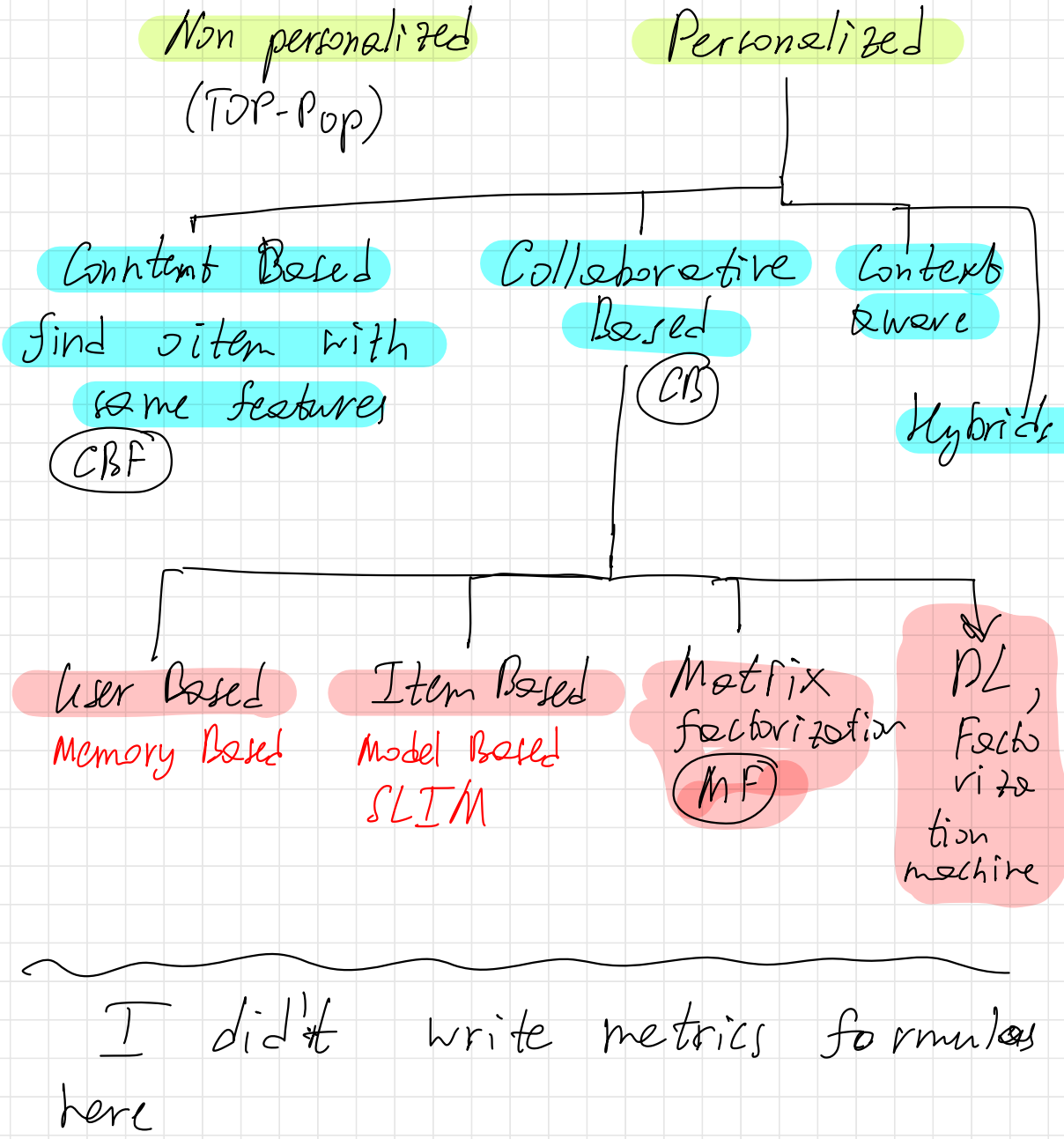
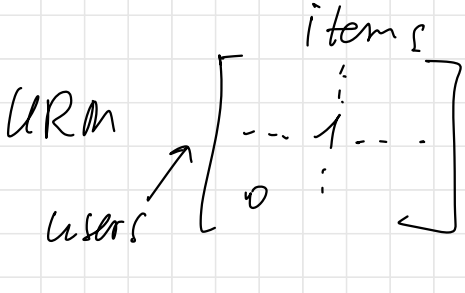
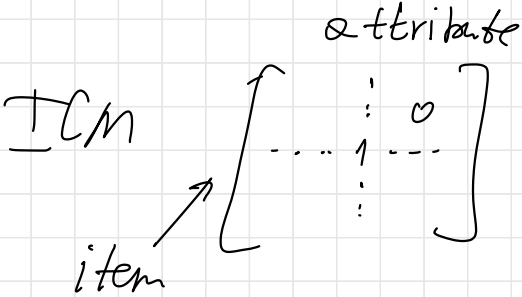


I didn't write metrics formulas here





$\{0, 1\}$ – implicit parameter (interacted or not)

$[0-5]$ – explicit parameter, (how user evaluated recommendation)

Dot product $\vec{i} \cdot \vec{j} = \sum_k i_k \cdot j_k = s_{ij}$ ↑ similarity

Cosine similarity $\frac{\vec{i} \cdot \vec{j}}{|\vec{i}| \cdot |\vec{j}|} = \cos\theta$ normalized

θ - angle between vector $\vec{i}, \vec{j}$

Shrink term $\frac{\vec{i} \cdot \vec{j}}{|\vec{i}| \cdot |\vec{j}| + t}$

if interacted only in one item/attribute

⇒ will be not relevant (some kind of regularization)

Content Based Filtering (ICn)

item

item $\begin{bmatrix} - & 0{,}2 & \\ 0{,}1 & - & \\ & - & - \end{bmatrix}$

Similarity matrix for item

similar, with empty diagonal

Rating prediction based on Item Sim Matr normalised

$$r_{ui} = \frac{\sum_j r_{uj} s_{ij}}{\sum_j s_{ij}}$$

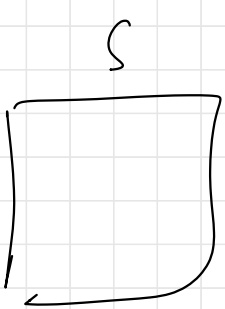
user u put rating for item j

Item

from Similarity Matr.

$$\tilde{r}_u = r_u \cdot S$$

$$\tilde{R} = R \cdot S$$ - for all users



$$r_{ui} = \frac{\sum_{j=KNN(i)} r_{uj} s_{ij}}{\sum_{j=KNN(i)} s_{ij}}$$

only if j in to k nearest to i

TF-IDF (Term Frequency – inverse document frequency)

$$TF_{a,i} = \frac{N_{a,i}}{N_i}$$ often = 1

$N_{a,i}$ – appearances of attribute a in item i

N_i – number of attributes of item i

$$IDF_a = \log \frac{N_{ITEMS}}{N_a}$$

N_{ITEMS} – total number of items

N_a – number of items with attribute a

Collaborative filtering (URM)

User Based similarity

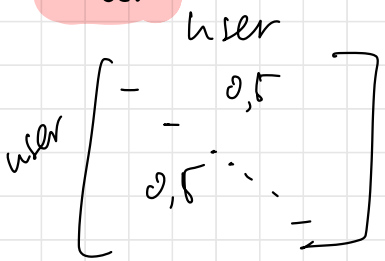
Cosine similarity ↗

$$\frac{\bar{r}_u \cdot \bar{r}_v}{|\bar{r}_u| \cdot |\bar{r}_v| + t}$$

rating of user v to item i

$$r_{ui} = \frac{\sum_{v \in KNN(u)} r_{vi} \cdot s_{uv}}{\sum_{v \in KNN(u)} s_{uv}}$$

S_{uv}



But when we use explicit rating we have take into account, that different users put their rating differently => remove bias effect.

from Cosine to Pearson correlation

1 similar, -1 not similar

$$\bar{r}_u = \frac{\sum_i r_{ui}}{N_u + t} \qquad \bar{r}_v = \frac{\sum_i r_{vi}}{N_v + t}$$

how much user u like item i compared to his average opinion

$$s_{uv} = \frac{\sum_i (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_i (r_{ui} - \bar{r}_u)^2 \cdot \sum_i (r_{vi} - \bar{r}_v)^2} + t}$$

Prediction

$$r_{ui} - \bar{r}_u = \frac{\sum_{v \in KNN(u)} (r_{vi} - \bar{r}_v) s_{uv}}{\sum_{v \in KNN(u)} s_{uv}} \Rightarrow$$

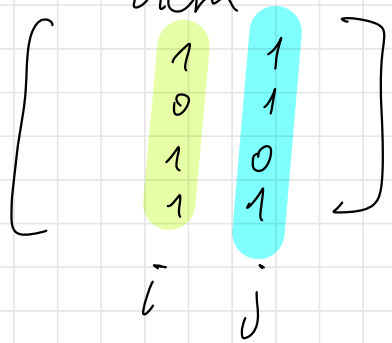
$$r_{ui} = \frac{\sum_{v \in KNN(u)} (r_{vi} - \bar{r}_v) s_{uv}}{\sum_{v \in KNN(u)} s_{uv}} + \bar{r}_u$$

do you like i more or less

user average opinion

Item based Collaborative filtering

Item similarity (URM)



$$s_{ij} = \frac{\sum_u r_{ui} \cdot r_{uj}}{\sqrt{\sum_u r_{ui}^2 \cdot \sum_u r_{uj}^2}}$$

$$r_{ui} = \frac{\sum_{j \in KNN(i)} r_{uj} \cdot s_{ij}}{\sum_{j \in KNN(i)} s_{ij}}$$

prediction

$$\tilde{r}_u = r_u \cdot S$$

$$\tilde{R} = R \cdot S$$

for explicit ratings

$$\bar{r}_u = \frac{\sum_i r_{ui}}{N_u + t}$$

user bias

from cosine to adjusted cosine

same

$$s_{ij} = \frac{\sum_u (r_{ui} - \bar{r}_u)(r_{uj} - \bar{r}_u)}{\sqrt{\sum_u (r_{ui} - \bar{r}_u)^2 \cdot \sum_u (r_{uj} - \bar{r}_u)^2}}$$

Comparison User Based and Item Based

User Based

$$r_{ui} = \frac{\sum_{v \in KNN(u)} r_{vi} \cdot s_{uv}}{\sum_{v \in KNN(u)} s_{uv}}$$

similarity between users: cosine similarity

Item Based

$$r_{ui} = \frac{\sum_{j \in KNN(i)} r_{uj} \cdot s_{ij}}{\sum_{j \in KNN(i)} s_{ij}}$$

similarity between items: cosine similarity

as Content based, but without knowing attributes

For explicit ratings

$$r_{ui} = \frac{\sum_{v \in KNN(u)} (r_{vi} - \bar{r}_v) \, s_{uv}}{\sum_{v \in KNN(u)} s_{uv}}$$

Pearson correlation coefficient (remove user bias)

$$r_{ui} = \frac{\sum_{j \in KNN(i)} (r_{ui} - \bar{r}_u) \, s_{ij}}{\sum_{j \in KNN(i)} s_{ij}}$$

cosine adjusted (remove user bias

Model Based / MEMORY Based

Model: $\begin{cases} f(URM) & \text{Collaborative} \\ f(ICM) & \text{Content Based} \end{cases}$

estimating ratings $g(\text{model}, \text{user profile})$

recommendation only for users from data set

user profile $\in$ URM $\rightarrow$ Memory Based

user profile $\notin$ URM $\rightarrow$ Model Based

↳ if come new user, we have to recompute similarities

↳ can make recommendation for new users

Item based CF - MODEL BASED

- $S_{II} = \mathbf{f}(URM)$
- estimated ratings = $\mathbf{g}(S_{II}, \vec{r}_u)$

Model Based

Item Based

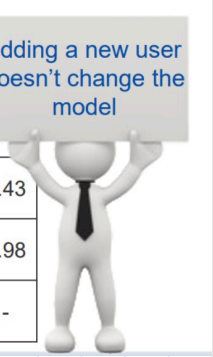
$$\tilde{r}_{ui} = \frac{\sum_{j \in KNN(i)} r_{uj} \cdot s_{ji}}{\sum_{j \in KNN(i)} s_{ji}}$$

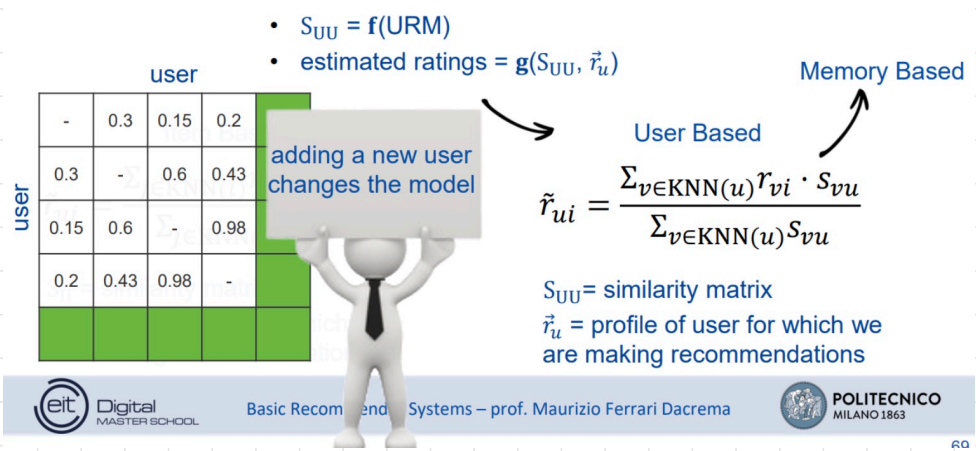
S_{II} = similarity matrix

$\vec{r}_u$ = profile of user for which we are making recommendations

doesn't change significantly II matrix

item	item			
	-	0.3	0.15	
	0.3	-	0.6	0.43
	0.15	0.6	-	0.98
	0.2	0.43	0.98	-





More significantly change model, because we will have new column and row $\Rightarrow$ **Memory based**

But in Item based, if we add new item $\Rightarrow$ we also have to recompute S_{II}, Because there will be new column.

Association Rules

$P(\text{item } i \mid \text{history of ratings}) = P(i \mid j_1, j_2, j_3, \ldots)$

implicit rating case: $P(i|j) \approx \frac{\#\langle i,j\rangle}{\#\langle j\rangle} = s_{ij}$

$$P(i|j) \approx \frac{\#\langle i,j\rangle}{\#\langle j\rangle + t} = s_{ij}$$

By using frequency based approach, we measure how frequently happen that a user likes both item i and item j. by knowing that user liked item j.

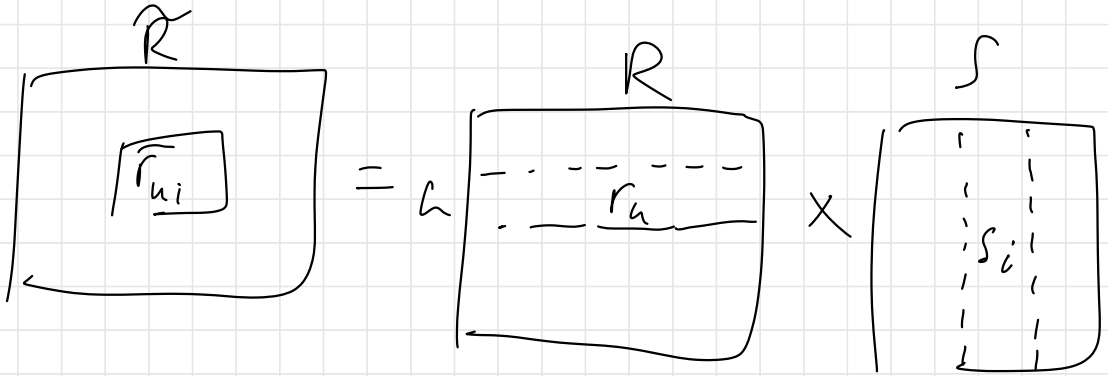
The conditional probability, that a user will like item i, by knowing that he/she previously liked item j. is equivalent to count, how many times a user liked both item i and item j. divided by the number

of times, he or she liked item j.

this is new definition of s_{ij}: similarity.

ITEM BASED collaborative filtering as an Optimization problem (SLIM)

$$\tilde{r}_{ui} = \sum_j r_{uj} s_{ji} \qquad \tilde{R} = RS \qquad S \in R^{(|I| \times |I|)}$$



Instead heuristic to compute S_{II}, we can learn it from data.

$\tilde{R} = RS$ ML model s_{ij} - parameter that we will learn

As loss function we will use MSE, because other aren't differentiable

SLIM – sparse Linear method for Top-N Recommender systems

$$E(S) = \sum (r_{ui} - \tilde{r}_{ui})^2$$

$u,i \in R^+$ — only existing ratings

$$\tilde{r}_{ui} = \sum_j r_{uj} s_{ij}$$ Not normalised

$$E(S) = \| R - \underbrace{RS}_{\tilde{R}} \|_2$$

$$S^* = \underset{S}{\operatorname{argmin}} \|R - RS\|_2 = \underset{S}{\operatorname{argmin}} E(S)$$

But if $S = I \Rightarrow$ degenerate solution

$\Rightarrow$ add regularization weight

$$E(S) = \|R - RS\|_2 + \lambda \|S\|_2 \qquad \lambda > 0$$

$$\|S\|_2 = \sum_{ij} s_{ij}^2$$

$\lambda \to 0 \Rightarrow S = I$

$\lambda \to \infty \Rightarrow S = 0$

↑ as sparse as possible

$$\|S\|_1 = \sum_{ij} |s_{ij}| \quad - \quad L_1 \text{ regularization}$$

$$E(S) = \|R - RS\|_2 + \lambda_2 \|S\|_2 + \lambda_1 \|S\|_1$$

$$\lambda_1, \lambda_2 \geq 0$$

SLIM

Lasso (L1)	Ridge (L2)
Absolute value of the parameter	**Squared** value of the parameter
Can force the parameter to be **zero**	Shrinks the values of the less important parameters **towards zero**.
Can lead to a **sparse model**	Does not lead to a sparse model
Not differentiable, can be solved numerically (gradient descent)	**Differentiable**, can be solved analytically

$$\frac{\partial E(S)}{\partial S} = -2(R - RS)R^T + 2\lambda_2 S + \lambda_1 \frac{\partial L_1}{\partial S}$$

$$\frac{\partial L_1}{\partial s_{ij}} = \begin{cases} -1, & s_{ij} < 0 \\ 0, & s_{ij} = 0 \\ 1, & s_{ij} > 0 \end{cases} = sign(s_{ij})$$

Steps of S updating

- Sample $u, i, r_{ui} \in R^+$ (existing ratings)
- Compute Gradient

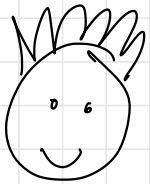
$$\frac{\partial E(S)}{\partial s_{i*}} = -2(r_{ui} - r_u s_{i*}) r_u^T + 2\lambda_2 s_{i*} + \lambda_1 sign(s_{i*})$$

(s_{i*}: i-th column)

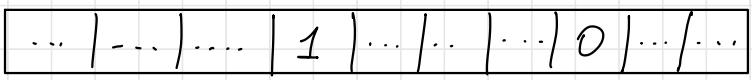
- Update parameters $s_{i*} = s_{i*} - \eta \cdot \frac{\partial E(S)}{\partial s_{i*}}$ (η: learning rate)
- Repeat

BPR Bayesian Personalized Ranking

Optimize pairwise ranking



u



Item with interactions

item i

item j

Item without interaction

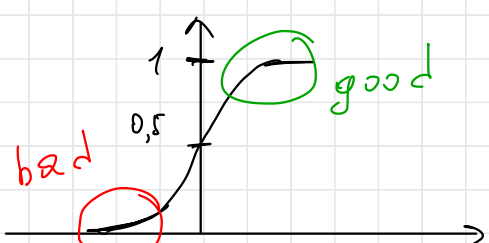
we want

$\tilde{r}_{ui} > \tilde{r}_{uj}$

maximize $P(\tilde{r}_{ui} > \tilde{r}_{uj} \mid u)$

$x_{uij} = \tilde{r}_{ui} - \tilde{r}_{uj}$ $\quad P(\tilde{r}_{ui} > \tilde{r}_{uj} \mid u) \cong \sigma(x_{uij})$

$$\sigma(x_{uij}) = \frac{1}{1 + e^{-x_{uij}}}$$



θ - model parameters

i - item relevant for user u

j - item non relevant for user u

$$P(\tilde{r}_u(\theta) > \tilde{r}_{uj}(\theta) \mid u) \equiv \sigma(x_{uij}(\theta))$$

Assume that probability for each pair of items are independent (but they are not in real)

$$\underset{\theta}{\operatorname{argmax}} \prod_{u,i,j} P(\tilde{r}_{ui}(\theta) > \tilde{r}_{uj}(\theta) \mid u)$$

maximise log likelihood

$$\underset{\theta}{\operatorname{argmax}} \log\left[\prod_{u,i,j} P(\tilde{r}_{ui}(\theta) > \tilde{r}_{uj}(\theta) \mid u) \right] =$$

$$= \underset{\theta}{\operatorname{argmax}} \sum_{u,i,j} \log\left(P(\tilde{r}_{ui}(\theta) > \tilde{r}_{uj}(\theta) \mid u)\right) =$$

$$= \underset{\theta}{\operatorname{argmax}} \sum_{u_{ij}} \log \sigma(x_{u_{ij}}(\theta)) \iff$$

$$\underset{\theta}{\operatorname{argmin}} \; -\sum_{u_{ij}} \log \sigma(x_{u_{ij}}(\theta)) + \underbrace{\lambda \|\theta\|_2}_{\text{regularization}}$$

$$\frac{\partial}{\partial \theta} \left\{ -\sum_{u_{ij}} \log \sigma(x_{u_{ij}}(\theta)) \right\}$$

$$\ln' x = \frac{1}{x}$$

$$= -\sum \frac{\partial}{\partial \theta} \log(\sigma(x_{u_{ij}}(\theta))) =$$

$$= -\sum \frac{1}{\sigma(x_{u_{ij}}(\theta))} \sigma(x_{u_{ij}}(\theta)) (1 - \sigma(x_{u_{ij}}(\theta))) \frac{\partial x_{u_{ij}}(\theta)}{\partial \theta}$$

$$\frac{\partial}{\partial x} \frac{1}{1+e^{-x}} = \frac{-(1+e^{-x})'}{(1+e^{-x})^2} =$$

$$= \frac{e^{-x}}{(1+e^{-x})^2} = \frac{e^{-x}}{1+e^{-x}} \quad \frac{1}{1+e^{-x}} = \frac{1}{e^x+1} \cdot \sigma(x) =$$

$$= \sigma(-x)\,\sigma(x) = (1-\sigma(x))\,\sigma(x)$$

$$\frac{e^{-x}+1-1}{e^{-x}+1} = 1 - \frac{e^{-x}}{e^{-x}+1} = (1-\sigma(x))$$

$$x_{uij} = r_u \cdot s_{i*} - r_u s_{j*} = r_u (s_{i*} - s_{j*})$$

$$\frac{\partial x_{uij}}{\partial s_{i*}} = r_u^T \qquad \frac{\partial x_{uij}}{\partial s_{j*}} = -r_u$$

OPTIMIZE BPR (Missing as negative)

↓

MAXIMIZE AUC in the ROC curve

- Sample data point u, i, j

  $u, i \in R^+$

  $u, j \in R^-$

- Compute gradients $\frac{\partial E(s)}{\partial s_{i*}} = \frac{e^{-x_{ui,j}}}{1+e^{-x_{ui,j}}} r_u^T$

- Update parameters $s_{i*} = s_{i*} - \eta \frac{\partial E(s)}{\partial s_j}$

- Repeat

WARP (Weighted Approximate-Rank Pairwise) loss.

Idea: When sampling u, i, j check if model predictions would already produce the right ranking, i.e. $\tilde{r}_{ui} > \tilde{r}_{uj}$. If so, draw another j. The goal is to look for training samples that violate (нарушать) the correct ranking

BPR exhibits a very strong popularity bias

- Popular items will be sampled a lot, their predictions will tend to be high
- Unpopular items will be sampled rarely, their predictions will tend to be low

Offline evaluation often uses test data splits that exhibit popularity bias (the same of the training data).

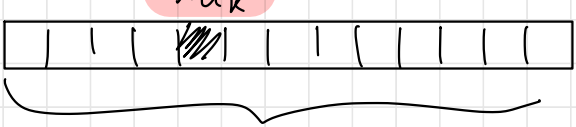
BPR may look good offline but perform poorly online

Matrix factorization

In CB we don't have content attributes.

Idea: each user and item will be associated to set of latent features which are learned from the collaborative data (also called latent factors, or embeddings)

User preferences

$\vec{x_u}$ = 

x_{uk} - how much user u likes feature k.

Item description

$\vec{y_i}$ = 

y_{ik}: how much feature k is important for item

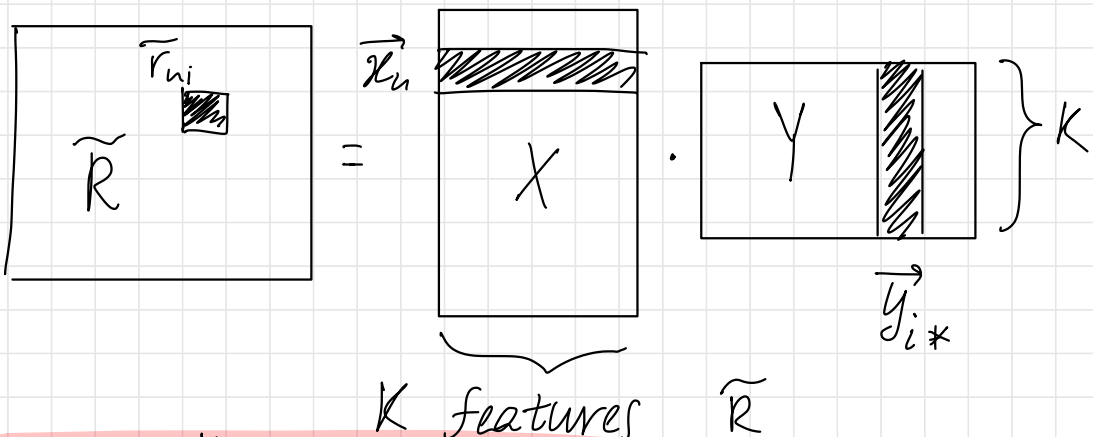
$$\tilde{r}_{ui} = \sum_k x_{uk} \cdot y_{ik} = \vec{x}_u \cdot \vec{y}_i$$

estimated rating for user u on item i

$X \in \mathbb{R}^{|U| \times K}$: user-feature matrix

$Y \in \mathbb{R}^{K \times |I|}$: feature-item matrix

$$\tilde{R} = X \cdot Y$$



$$X^*, Y^* = \underset{X,Y}{\operatorname{argmin}} \| R - \tilde{R} \|_2 \quad \text{where } \tilde{R} = XY$$

(one of the possible loss function)

$X - |U| \times |K|$, $Y - |K| \times |I|$

in SLIM and BPR we had $|U| \times |U|$ params

now we have $|K| \cdot (|U| + |I|)$ - which usuall much less then $|U| \cdot |U|$

K - hyperparameter

Number of Latent Factors

How many latent factors? Common values are between 10 and 200

• Large number of latent factors • Few ratings	• Small number of latent factors • Many ratings
• Risk of overfitting • Poor scalability of the model to large data sets	• Personalization capabilities of the system are reduced • Biased towards popular items • For $K = 1$ we have Top Popular

Regularization: $X^*, Y^* = \underset{X,Y}{\operatorname{argmin}} \|R - XY\|_2 +$

$+ \lambda_x \|X\|_2 + \lambda_y \|Y\|_2$

gradient: $\frac{d}{dx_u}\left((r_{ui} - x_u y_{i*})^2 + \lambda_x x_u^2\right) =$

$= -2(r_{ui} - x_u y_{i*}) y_{i*}^T + 2\lambda_x x_u$

- Sample data point u, i, r_{ui}
- Compute gradient on X:

$$\frac{\partial E(X,Y)}{\partial x_u} = -2(r_{ui} - x_u y_{i*}) y_{i*}^T + 2\lambda_x x_u$$

- Compute gradient on Y:

$$\frac{\partial E(X,Y)}{\partial y_{i*}} = -2(r_{ui} - x_u y_{i*}) x_u^T + 2\lambda_y y_{i*}$$

- Update parameters X

$$x_u = x_u - \ell \frac{\partial E(X,Y)}{\partial x_u}$$

- Update parameters Y

$$y_{i*} = y_{i*} - \ell \frac{\partial E(X,Y)}{\partial y_{i*}}$$

- Repeat

Since most of the URM values are zeros, it's important to decide how to consider these values, when choosing the best feature matrices X and Y

- Missing as random. We assume that it's equally likely that a user will like or dislike an item, which hasn't been rated (50/50 - 0/1)

Funk SVD: minimize error function only on the non-zero elements → missing as random

- Missing-as-negative. It's more likely for a user to dislike an item that has not been rated yet. (optimise Precis, Recall)

Pure SVD: minimize the error fnction on all the matrix → Missing-as-negative

Alternating Least Squares (ALS)

$$X^*, Y^* = \underset{X, Y}{\operatorname{argmin}} \|R - XY\|_2 + \lambda_x \|X\| + \lambda_y \|Y\|$$

loss - degree 4, gradient degree 3

difficult to find zeros.

ALS - reduces the loss to quadratic and gradient to linear.

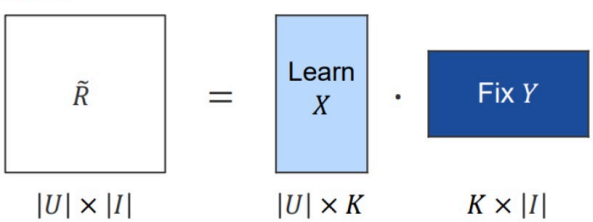
Alternating Least Squares (ALS)

At the beginning, the matrices X and Y are filled with random values

Fix X and learn Y

Fix Y and learn X

continue until the improvements are irrelevant (or until the error is minimized)



Iteratively update only one of matricies and each iteration update new one.

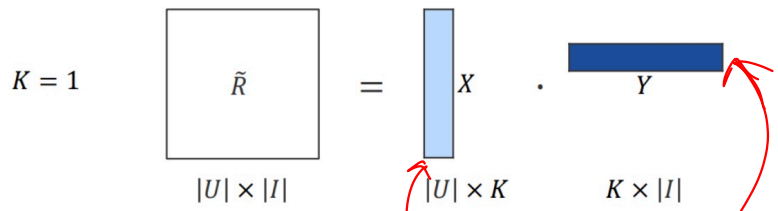
Funk SVD

The latent factors are not computed all at once.

- At the beginning $K = 1$
  X and Y are tuned until the error is minimized

Fix X and learn Y

Fix Y and learn X

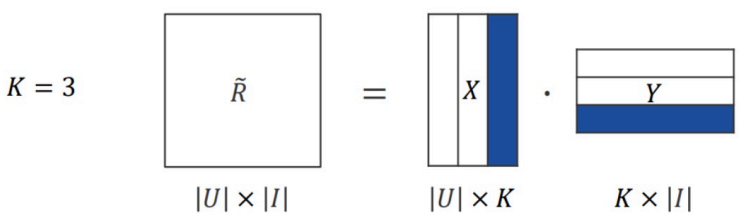


first feature

Funk SVD

The latent factors are not computed all at once.

- At the beginning $K = 1$, X and Y are tuned until the error is minimized
- K is increased and the method tunes again X and Y
- K is increased until the desired value



√ Learning on latent factor at time is efficient and reduces overfitting

X Missing as Random Assumption

X Well for rating prediction, but poorly for Top-N (sample zero ratings)

X No latent factors for new users or new items ⇒ retrain model

SVD ++

In Funk SVD $\tilde{r}_{ui} = \sum_k x_{uk} \cdot y_{ik}$

SVD ++ $\tilde{r}_{ui} = \underbrace{\mu + b_u + b_i}_{\text{global effects, learned}} + \sum_k x_{uk} \cdot y_{ik}$

$\mu \in \mathbb{R}$ - mean rating, overall average rating

$b_u \in \mathbb{R}^{|U|}$ - user bias, average rating a user gave to all the rated items

$b_i \in \mathbb{R}^{|I|}$ - item bias, average rating an item received

SVD++ can be better with explicit ratings than Funk SVD, because it consider global effects

$X \cdot Y + \mu + b_u + b_i$

$\mu^*, b_u^*, b_i^*, X^*, Y^* = \operatorname{argmin} \|R - \tilde{R}\|_2 + \lambda_x \|X\|_2 + \lambda_y \|Y\|_2$

$\mu^*, b_u^*, b_i^*, X^*, Y^*$ + ...

this model has user bias, item bias

$$K(|U|+|I|) + |U|+|I| + 1 \text{ parameters}$$

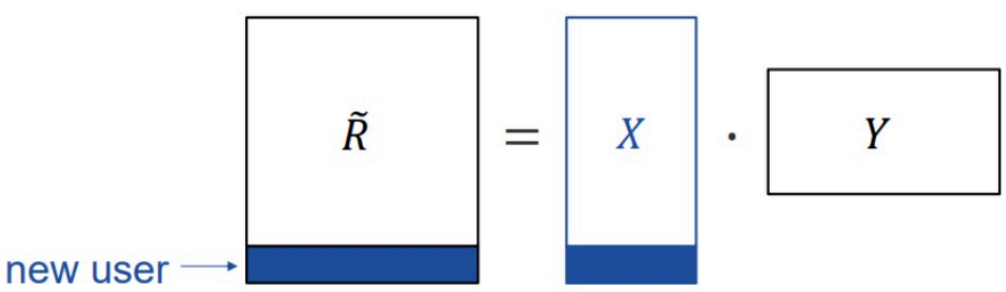
(μ)

Not practical to optimize with ALS, so it is solved with stochastic Gradient Descent.

Not Model Based

✗ New user → need to recompute X

✗ New item → need to recompute Y



Adapting SVD++ (Asymmetric SVD)

How to solve problem of new users and items — define the user factors as function of the items they interacted with.

$$x_{uk} = \sum_j r_{uj} \cdot z_{jk} \qquad z \in \mathbb{R}^{|I| \times |k|}$$

the user preference for a feature is estimated based on:

r_{uj}: how much user u likes item j (ask the user for some initial preferences)

z_{jk}: how much the feature k is present in the item j

---

approximate x_{uk} for new users.

$$x_{uk} = \sum_j r_{uj} z_{jk} = \vec{r_u} \cdot \vec{z_k}$$

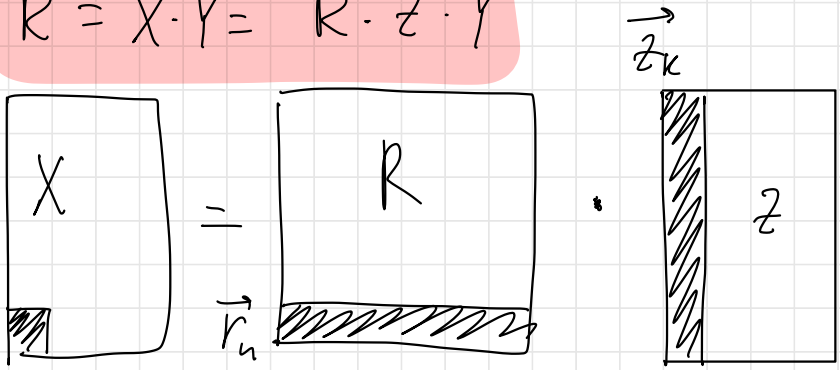
$\vec{r}_u =$ | | 3 | | 5 | | 2 | |

We ask the user some preferences

Movie	Titanic	Iron Man	Shrek	Finding Nemo
Liking	3	5	2	4

$$X = R \cdot Z$$

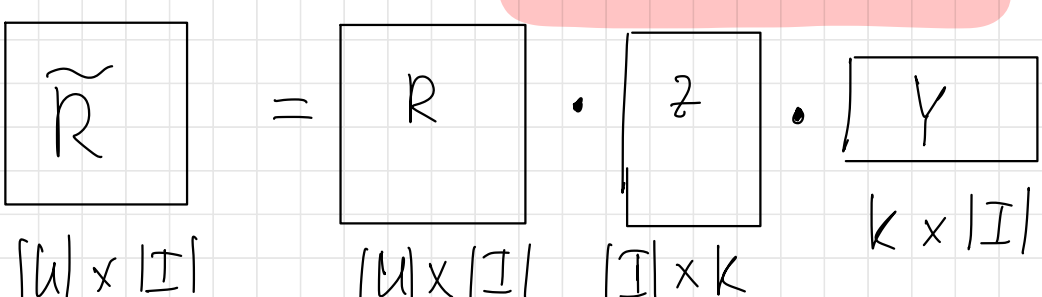
$$\tilde{R} = X \cdot Y = R \cdot Z \cdot Y$$



x_{uk}

$|U| \times K$ $\quad$ $|U| \times |I|$ $\quad$ $|I| \times K$

$$\tilde{r}_{ui} = \sum_k x_{uk} \cdot y_{ki} \rightarrow \tilde{r}_{ui} = \sum_k \left(\sum_j r_{uj} z_{jk} \right) y_{ki}$$



$|U| \times |I|$ $\quad$ $|U| \times |I|$ $\quad$ $|I| \times K$ $\quad$ $K \times |I|$

r_{uj}: recorded user preference for item j

z_{jk}: latent feature preference factor

y_{ki}: how much latent feature k is present in item I

$2k|I|$ parameters

Asymmetric SVD with global effects

$$\tilde{r}_{ui} = \mu + b_i + b_u + \sum_k \left(\sum_j r_{uj} z_{jk} \right) y_{ki}$$

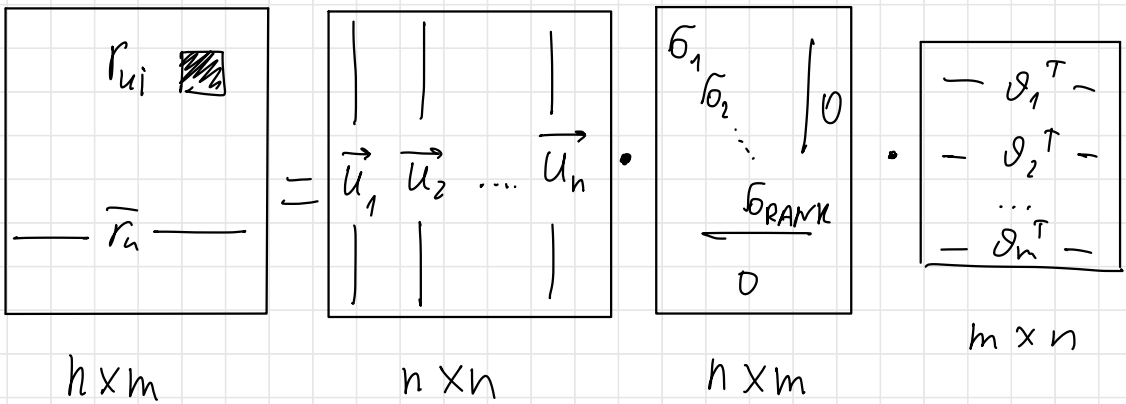
$2K|I| + |I| + |U| + 1$ parameters

$$\tilde{R} = R \cdot Z \cdot Y =$$

not similar matrix

$$= R \quad S \quad (|I| \times |I|)$$

Pure SVD Singular Value Decomposition



$n \times m$ $\quad$ $n \times n$ $\quad$ $n \times m$ $\quad$ $m \times m$

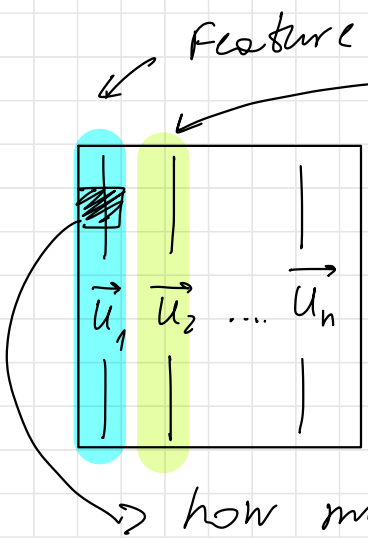
$$R = U \Sigma V^T$$

n users, m items

U: left singular vectors $\quad U^T U = I \quad U U^T = I$

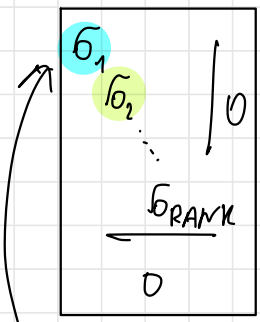
Columns

- Eigenvectors of R (left)
- Hierarchically arranged by their ability to describe the variance in columns of R.



→ how much user 2 is interested in feature 1.

Σ: Singular values, diagonal matrix, rank (R) elements.

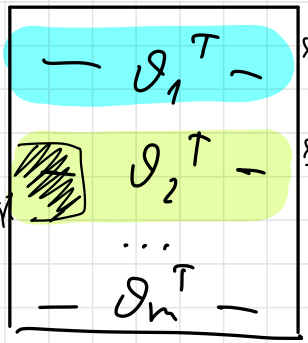


Singular values of R.

- non negative
- hierarchically arranged.

$\sigma_1 \geq \sigma_2 \ldots \geq \sigma_{Rank(R)} \geq 0$

strength of each feature in R



$m \times m$

V: Right singular vectors

Orthogonal $V^T V = V V^T = I$

Rows of V^T

- Eigenvectors of R (right)
- Hierarchically arranged by their ability to describe the variance of rows of R.

how much feature 2 describe item 1

Truncated SVD

$R = U \cdot \Sigma \cdot V^T \approx U_k \Sigma_k V_k^T$

first K highest singular values

$U_K \Sigma_K V_K^T =$ $\vec{u}_1$ $\vec{u}_2$ $\cdot$ K { σ_1 σ_2 } $\cdot$ $\vec{v}_1^T$ $\vec{v}_2^T$

$n \times K$ $\quad K \times K$ $\quad K \times m$

Frobenius norm: $\|A\|_f = \sqrt{\sum_{i,j} a_{ij}^2}$ → all values

$\|A\|_F^2 = \sum_k \sigma_k^2(A)$ sum of squares of singular values

The best low-dimensional approximation of matrix A by a rank K under the Frobenius norm is given by truncated SVD

The approximation error is given by the $K+1$ singular value

$$\|A - A_K\|_F = \|A - U_K \Sigma_K V_K^T\|_F = \sigma_{K+1}(A)$$

if we apply SVD decomposition on R we obtain user and item factors

But, this will not work for recommendation

The full rank SVD decomposition is exact, we will perfectly reconstruct R, missing interactions included => Overfitting.

Pure SVD uses a truncated SVD decomposition with K components:

- Approximate error of R allows the model to generalize
- Contains the main features and reduces overfitting

$U_K \in \mathbb{R}^{|U| \times K}$: user-feature matrix

$\Sigma_K \in \mathbb{R}^{K}$: singular values

$V_K^T \in \mathbb{R}^{K \times |I|}$: feature-item matrix

$$R \approx \tilde{R} = U_K \Sigma_K V_K^T$$

if $P = U\Sigma$ $\quad R = U\Sigma V^T \quad | \cdot V$

$V V^T = I$

$V^T = V^{-1}$

$RV = U\Sigma = P$

$R = U\Sigma V^T = P V^T = RVV^T = R(S)$

similarity

Pure SVD equivalent to an item based CF with $S = V \cdot V^T$

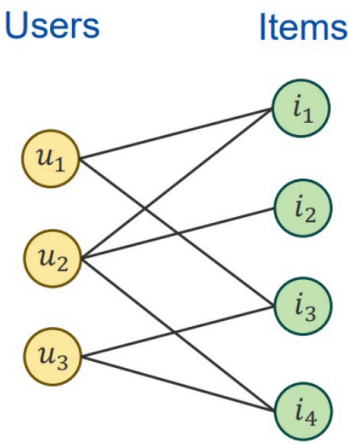
By using truncated SVD, PureSVD optimizes the Frobenious norm

The Frobenious norm accounts for all values of R, including the missing ratings which are treated as zeros

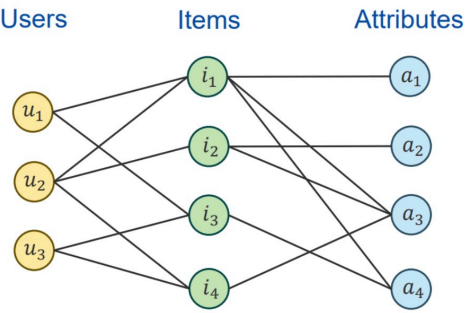
PureSVD relies on the Missing-as-Negative Assumption

Graph Based

Bipartite User-Item Graph



Tripartite User-Item-Attributes Graph



A graph is defined as $G = (N, E)$

N the set of nodes

E the set of edges

Collaborative Model

$$N = U \cup I$$
$$E = \{u, i \mid u, i \in R^+\}$$

collaborative

Hybrid Content-Collaborative Model

$$N = U \cup I \cup A$$
$$E = \{u, i \mid u, i \in R^+\} \cup \{i, a \mid i, a \in C^+\}$$

content based

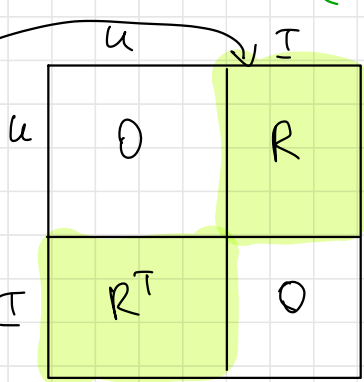
Adjacency Matrix Notation

Adjacency Matrix $G \in \mathbb{R}^{|N| \times |N|}$, $g_{xy} = \begin{cases} 1, & \text{node } x \text{ is connected to node } y \\ 0, & \text{otherwise} \end{cases}$

- **Undirected** Graph
  G is symmetric, $g_{xy} = g_{yx}$

- **Directed** Graph
  G is asymmetric, $g_{xy} = 1, g_{yx} = 0$

User Item edges

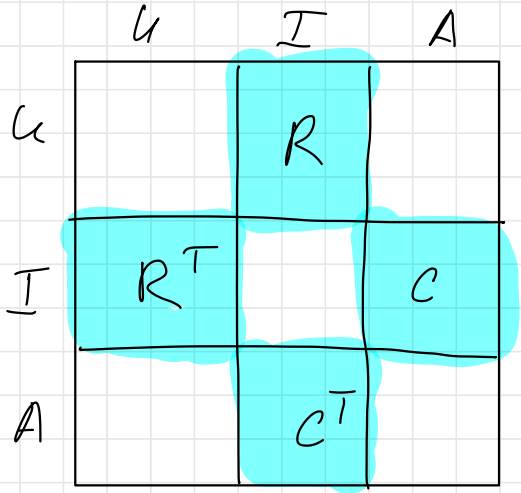


$G =$

$$G = \begin{bmatrix} 0 & R \\ R^T & 0 \end{bmatrix}$$

Item-User edges

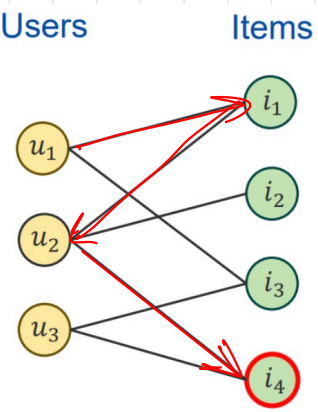
$$G = \begin{bmatrix} 0 & R & 0 \\ R^T & 0 & C \\ 0 & C^T & 0 \end{bmatrix}$$



Random walk

$u_1 \rightarrow i_1 \rightarrow u_2 \rightarrow \boxed{i_4}$

Recommendation
(at 3 hops)
(steps)



We will compute probability to transition between two nodes and a probability distribution on being in each node

We need two things:

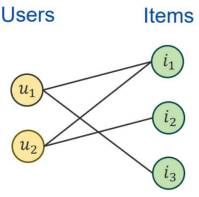
- A **transition probability** matrix, $P \in \mathbb{R}^{|N| \times |N|}$, $\sum_y p_{xy} = 1, \forall\, x$
- A **state probability** vector (row), $\Pi \in \mathbb{R}^{|N|}$, $\sum_i \Pi_i = 1$

Given $D \in \mathbb{R}^{|N|}$, d_x the number of outgoing edges from node x (its **degree**):

$$d_x = \sum_{y \in N} g_{xy} \qquad p_{xy} = \frac{g_{xy}}{d_x}$$

Symmetric!

$$G = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad D = \begin{bmatrix} 2 & 2 & 2 & 1 & 1 \end{bmatrix}$$



$$P = \begin{bmatrix} 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1/2 & 1/2 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Not symmetric

State Probability

The probability of being in state i can be computed, given:

- The state we were in at the previous hop (0)
- The probability to jump from it to i

probability of being on state 1 and go to state i

probability being in state 2 and go to state i

$$\Pi_i^{(1)} = \Pi_1^{(0)} p_{1i} + \Pi_2^{(0)} p_{2i} + \cdots = \sum_{j \in N} \Pi_j^{(0)} p_{ji}$$

$$\Pi^{(h)} = \Pi^{(h-1)} \cdot P$$

hoop

Repeat as needed

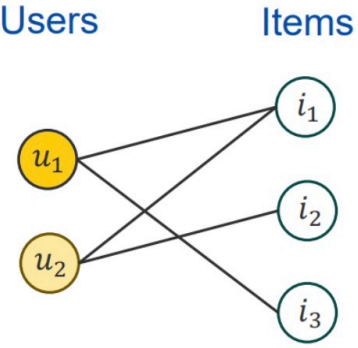
$$\Pi^{(0)} = [1 \quad 0 \quad 0 \quad 0 \quad 0]$$

$$\Pi^{(1)} = \Pi^{(0)} \cdot P = \left[0 \quad 0 \quad \frac{1}{2} \quad 0 \quad \frac{1}{2}\right]$$

$$\Pi^{(2)} = \Pi^{(1)} \cdot P = \left[\frac{3}{4} \quad \frac{1}{4} \quad 0 \quad 0 \quad 0\right]$$

$$\Pi^{(3)} = \Pi^{(2)} \cdot P = \left[0 \quad 0 \quad \frac{4}{8} \quad \frac{1}{8} \quad \frac{3}{8}\right]$$

$$\Pi^{(4)} = \Pi^{(3)} \cdot P = \left[\frac{10}{16} \quad \frac{6}{16} \quad 0 \quad 0 \quad 0\right]$$



after few steps we can use this scores and rank them

$$P = \begin{bmatrix} 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1/2 & 1/2 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\tilde{r}_{ui} = \Pi^{(3)}_{|u|+i}$$

Recommendations i_1, i_2, i_3

Number of hops h usually 3, 5, 7

- if h is high I travel too far from target user
- Over smoothing - tend to be popularity - based

State Probability: Steady State

Every graph has a state Π such that:

$$\Pi = \Pi \cdot P$$

The state Π is called a **steady state** and is an **eigenvector of P**

If the random walk continues for $h \to \infty$

- We will reach the steady state regardless for the starting state
- For *n*-partite graphs we will not, we will find a *periodic* sequence of states

Page-Rank: Random Walk with Restart

Infinite random walk with probability $1 - \gamma$ to restart from the initial state (user)

$$\Pi = \gamma\Pi \cdot P + (1-\gamma)\Pi^{(0)}$$

Random walk ↑ ↑ Restart

The choice of γ allows to control how **personalized** it is

Π is found by solving a linear system of equations, but is expensive

$\gamma \to 1 \Rightarrow$ steady state

$\gamma \to 0 \Rightarrow$ we don't moving and don't predict

Computing P^3 is expensive (and dense) but:

- We only need a specific meta-path: $U \to I \to U \to I$
- P is a block matrix, we can compute only the portion we need

$$P = \begin{bmatrix} 0 & P_{UI} \\ P_{IU} & 0 \end{bmatrix} \qquad P_{UI} = \text{diag}\left[\frac{1}{D_U}\right] \cdot R$$

$$P_{IU} = \text{diag}\left[\frac{1}{D_I}\right] \cdot R^T$$

$U \to I \to U$, $\in [0,1]$ $|U| \times |U|$

$$P = \begin{bmatrix} 0 & P_{UI} \\ P_{IU} & 0 \end{bmatrix}$$

$$P^2 = \begin{bmatrix} P_{UI} \cdot P_{IU} & 0 \\ 0 & P_{IU} \cdot P_{UI} \end{bmatrix}$$

$I \to U \to I$ $\in [0,1]$ $|I| \times |I|$

$$P^3 = \begin{bmatrix} 0 & P_{UI} P_{IU} P_{UI} \\ P_{IU} P_{UI} P_{IU} & 0 \end{bmatrix}$$

$I \to U \to I \to U$

$U \to I \to U \to I$

(we need only this to predict)

The meta-path we are interested in is $U \to I \to U \to I$
In the following Π_I is the state probability for item nodes, Π_U for user nodes

$$\Pi^{(3)} = \Pi^{(0)} \cdot P^3 = \Pi^{(0)} \cdot \begin{bmatrix} 0 & P_{UI} \cdot P_{IU} \cdot P_{UI} \\ \cancel{P_{IU} \cdot P_{UI} \cdot P_{IU}} & 0 \end{bmatrix}$$

$$\Pi_I^{(3)} = \Pi_U^{(0)} \cdot P_{UI} \cdot P_{IU} \cdot P_{UI}$$

This is an item-item similarity

This is the (starting) user profile, $\Pi_U^{(0)}$ acts as a one-hot encoding

The meta-path we are interested in is $U \to I \to U \to I$

In the following Π_I is the state probability for item nodes, Π_U for user nodes

$$\Pi^{(3)} = \Pi^{(0)} \cdot P^3 = \Pi^{(0)} \cdot \begin{bmatrix} 0 & P_{UI} \cdot P_{IU} \cdot P_{UI} \\ P_{IU} \cdot P_{UI} \cdot P_{IU} & 0 \end{bmatrix}$$

$$\Pi_I^{(3)} = \Pi_U^{(0)} \cdot P_{UI} \cdot P_{IU} \cdot P_{UI} = \Pi_U^{(0)} \cdot P_{UI} \cdot S$$

similarity matrix Item-Item

$$S_{ij} = [P_{IU} \cdot P_{UI}]_{ij} = \sum_{u \in U} \frac{r_{ui}\, r_{uj}}{d_i\, d_j}$$

For implicit interactions P^3 is the Cosine similarity without shrinkage

In P_α^3 the transition probabilities are elevated to a power α

$$P_{UI} = \left(\text{diag}\left[\frac{1}{D_U}\right] \cdot R \right)^\alpha$$

$$P_{IU} = \left(\text{diag}\left[\frac{1}{D_I}\right] \cdot R^T \right)^\alpha$$

$$S_{ij} = \sum_{u \in U} \left(\frac{r_{ui}\, r_{uj}}{d_i\, d_j} \right)^\alpha$$

If $\alpha > 0$ small transition probabilities and similarities are attenuated

- P_α^3 tends to exhibit a strong popularity bias

$$P_{UI} = \left(\text{diag}\left[\frac{1}{D_U}\right] \cdot R \right)$$

$$P_{IU} = \left(\text{diag}\left[\frac{1}{D_I}\right] R^T \right)$$

RP^3_β tries to reduce the popularity bias

- **Problem**: it is (too) easy for a random walk to end up in items with high degree
- **Idea**: penalize the similarity of items with a high degree. Divide the similarity by the popularity of the destination item elevated to a power β

$$S_{ij} = \frac{1}{d_j^{\beta}} \sum_{u \in U} \left(\frac{r_{ui}\, r_{uj}}{d_i\, d_j} \right)^{\alpha}$$

Adding Side Information

However, if we add both item attributes A_I and user attributes A_U we encounter problematic meta-paths:

- $U \rightarrow I \rightarrow U \rightarrow I$
- $U \rightarrow I \rightarrow A_I \rightarrow I$
- $U \rightarrow A_U \rightarrow U \rightarrow I$ ← New (desired) meta-path

- $U \rightarrow I \rightarrow U \rightarrow A_U$
- $U \rightarrow A_U \rightarrow U \rightarrow A_U$

← These meta-paths do not end in the right node!

Deep learning recommender system

Basic Layer

In a typical fully-connected layer, one computes a weighted average of the input, adds a bias and then applies a non-linear activation function f

$$x_o = f_o(x_i \cdot W_o + b_o)$$

n : Input size
$x_i \in \mathbb{R}^n$: Input data
$W \in \mathbb{R}^{n \times m}$: Weight matrix

m : Output size
$x_o \in \mathbb{R}^m$: Output data
$b \in \mathbb{R}^m$: Bias vector

The parameters

Binary Cross Entropy

If we use implicit ratings and the model computes probabilities:

$$\underset{\theta}{\operatorname{argmin}} -\frac{1}{N}\sum^{N}\left[r_{ui}\log \tilde{p}_{ui}(\theta) + (1 - r_{ui})\log\left(1 - \tilde{p}_{ui}(\theta)\right)\right]$$

r_{ui}: Ground truth label (1/0)
$\tilde{p}_{ui}(\theta)$: Estimated probability that the label is 1, or $\tilde{p}_{ui}(\theta) = \sigma\left(\tilde{r}_{ui}(\theta)\right)$
θ : Parameters of the model
N : Number of samples the loss is computed on

This loss function can be applied to most of the recommender algorithms we discussed in the previous lectures (SLIM, MF…)

How do we sample the data?

- If we use only positive interactions, $u, i \in R^+$, we do not learn to rank
- If we use all ground truth, there are too many negatives! **Need to subsample**, for example draw a sample among positives or negatives with 50% probability

Autoencoders

Autoencoders for Recommendation

- The **input data** is the full user profile r_u
- The **embedding** will be a compressed representation of the user interests, (hopefully) sufficient to reconstruct their interactions
- The **reconstructed input** contains the predictions of the model for all items, which we rank and use to decide what to recommend

r_u — input

$e_u = g_e(r_u)$ — embedding after encoder

$\tilde{r}_u = g_d(e_u)$ — reconstructed input from decoder.

Encoder, Decoder will both reconstruct ratings for all users ⇒ collaborative filtering (Model Based)

Autoencoders as Item-Item Similarity Models

Consider a shallow autoencoder with no hidden layers and embedding size K

If $f = I$, $b_e, b_d = \bar{0}$ and share the parameters of encoder and decoder $W_d = W_e^T$:

$$e_u = f_e(r_u \cdot W_e + b_e) = r_u \cdot W_e$$

$$\tilde{r}_u = f_d(e_u \cdot W_d + b_d) = e_u \cdot W_d = r_u \cdot W_e \cdot W_d = r_u \cdot \boxed{W_e \cdot W_e^T}$$

Factorized **symmetric** item-item similarity

$W_e \in \mathbb{R}^{|I| \times K}$, $W_d \in \mathbb{R}^{K \times |I|}$

EASE^R

EASE^R is an item-based similarity collaborative filtering model

$$S^* = \underset{S}{\operatorname{argmin}} \|R - RS\|_F + \lambda \|S\|_F$$

$$s.t. \ \operatorname{diag}(S) = 0$$

1. To find the optimal S we set the gradient of the loss to zero (it is a function of $\vec{\gamma}$)

$$S^*(\vec{\gamma}) = \left(R^T \cdot R + \lambda I_{|I|}\right)^{-1} \cdot \left(R^T \cdot R - \operatorname{diagMat}(\vec{\gamma})\right)$$

2. To find the multipliers that satisfy the constraints, we solve $\operatorname{diag}\left(S(\vec{\gamma})\right) = 0$

$$\mathrm{P} = \left(R^T \cdot R + \lambda I_{|I|}\right)^{-1}$$

$$S^*(\vec{\gamma}) = P \cdot \left(P^{-1} - \lambda I_{|I|} - \operatorname{diagMat}(\vec{\gamma})\right) = I_{|I|} - P \cdot \left(\lambda I_{|I|} + \operatorname{diagMat}(\vec{\gamma})\right)$$

$$\boxed{\vec{\gamma} = \vec{1} \oslash \operatorname{diag}(P)}$$

3. The optimal S that will also satisfy the constraints is:

$$S^* = I_{|I|} - P \cdot \operatorname{diagMat}\left(\vec{1} \oslash \operatorname{diag}(P)\right)$$

$\oslash$: Element-wise division

- EASE^R is fast and highly effective
- Computing $R^T \cdot R$ and the inverse required for P is very memory intensive

Why does EASE^R refer to autoencoders?

The loss function aims to minimize the Frobenius norm of $R - RS$, hence a perfect solution would be an S such that $R = RS$

To quote the paper: "*This is done by reproducing the input as its output, as is typical for autoencoders.*" despite EASE^R not having their typical architecture

Denoising Autoencoders

One of the issues when training autoencoders is that the input space (user profiles) is very sparse

- **Risk:** the encoder might create poor embeddings for new user profiles
- **Risk:** the decoder may not know how to reconstruct correctly portions of the embedding space

Denoising Autoencoders

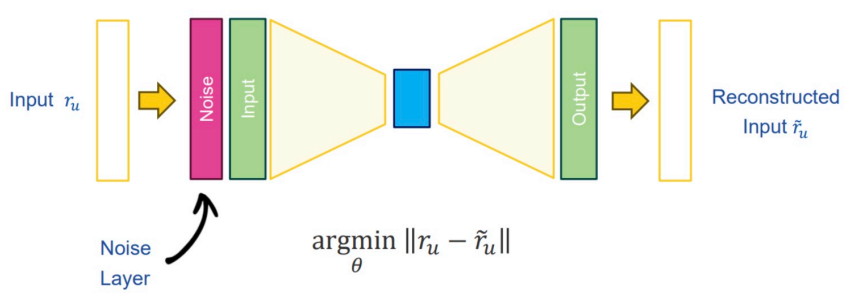
Idea: train the autoencoder to reconstruct the correct user profile from a noisy one

Types of noise Salt & Pepper:

- Randomly remove (dropout) a certain portion of the positive interactions
- Randomly add a (very) small number of positive interactions

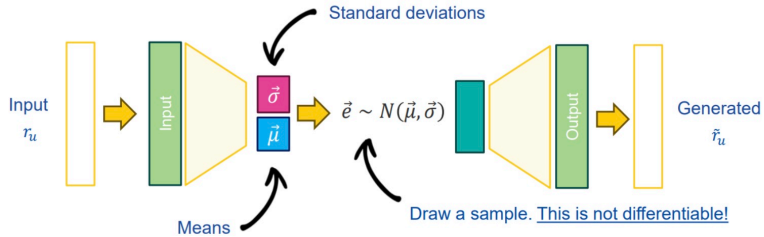
Both encoder and decoder will see many more user profiles

Denoising Autoencoders



VAE

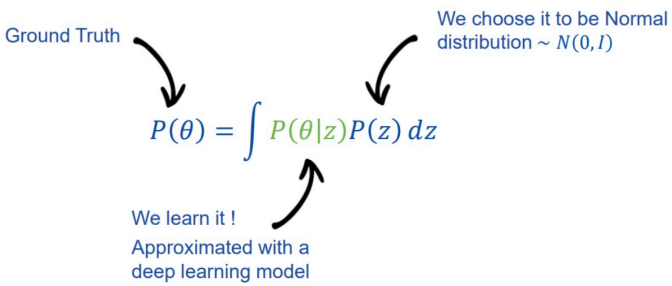
Variational Autoencoders



Reparametrization trick: $\vec{e} = \vec{\mu} + \vec{\sigma} \odot \vec{\epsilon}$

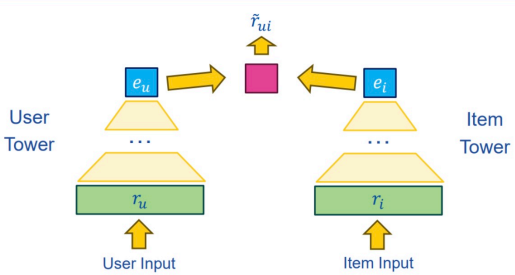
Where $\vec{\epsilon} \sim N(0,1)$, stochastic but we don't learn it

Variational Autoencoders



TWO-TOWER MODEL

Two-Tower Model



Consider a two-tower model with no hidden layers, embedding size K and both user and item input one-hot encoded x_u, x_i.

$$e_u = f^U(x_u \cdot W^U + b^U)$$
$$e_i = f^I(x_i \cdot W^I + b^I)$$

$$\tilde{r}_{ui} = e_u \cdot e_i$$

$x_u \in \mathbb{R}^{|U|}$, $x_i \in \mathbb{R}^{|I|}$

$W^U \in \mathbb{R}^{|U| \times K}$, $W^I \in \mathbb{R}^{|I| \times K}$

Two-Tower Model as… Matrix Factorization

Consider a two-tower model with no hidden layers, embedding size K and both user and item input one-hot encoded x_u, x_i. If $f = I$ and $b^U, b^I = \bar{0}$:

$$e_u = f^U(x_u \cdot W^U + b^U) = x_u \cdot W^U = W_u^U$$
$$e_i = f^I(x_i \cdot W^I + b^I) = x_i \cdot W^I = W_i^I$$

$$\tilde{r}_{ui} = e_u \cdot e_i = W_u^U \cdot W_i^I$$

User and Item latent factors

$x_u \in \mathbb{R}^{|U|}$, $x_i \in \mathbb{R}^{|I|}$

$W^U \in \mathbb{R}^{|U| \times K}$, $W^I \in \mathbb{R}^{|I| \times K}$

Two-Tower Model as… Matrix Factorization

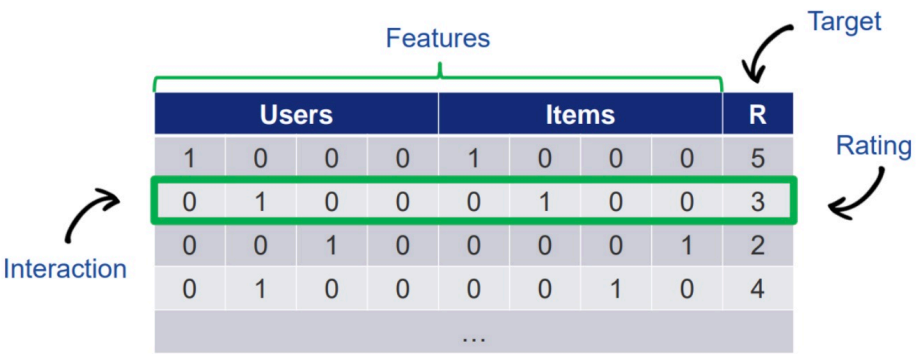
A two-tower model with one-hot encoded input is **memory based**

We cannot integrate new users or new interactions without retraining it

The issue can be overcome if one uses other encodings, for example the entire user profile. The model becomes equivalent to Asymmetric SVD

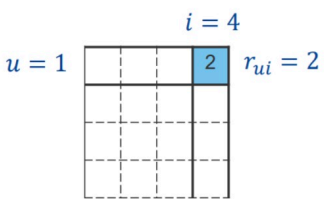
Factorization machines

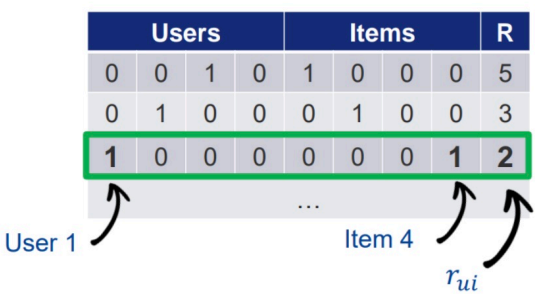
Input Data Representation



Input Data Representation: Example

Different format, same information as the URM:





The Model to Estimate Ratings

$$\tilde{r}^{(k)} = \omega_0 + \sum_{i=1}^{n} \omega_i \, x_i^{(k)} + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \omega_{ij} \, x_i^{(k)} x_j^{(k)}$$

index of row (user-item interaction)

i, j: Index of the columns

Users				Items				R
1	0	0	0	1	0	0	0	5
0	1	0	0	0	1	0	0	3
0	0	1	0	0	0	0	1	2

w_0, w_i, w_{ij} – to learn

$\omega_0 \in \mathbb{R}$ is the intercept of the model function

- it represents the global bias
- if we estimate only ω_0, we obtain the average of all the ratings

$$\tilde{r}^{(k)} = \omega_0 + \sum_{i=1}^{n} \omega_i \, x_i^{(k)} + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \omega_{ij} \, x_i^{(k)} x_j^{(k)}$$

n: number of columns/features ($|U| + |I|$)

$\omega_i \in \mathbb{R}$: models the strength of the i-th input feature

- This component is the linear combination of the variables

Given a row k, the equation can be:

$\tilde{r}^{(k)} = w_0 + w_u + w_i$

$\tilde{r}_{ui} = \mu + b_u + b_i$ ← equation to Global Effects

Quadratic Term

$$\tilde{r}^{(k)} = \omega_0 + \sum_{i=1}^{n} \omega_i \, x_i^{(k)} + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \omega_{ij} \, x_i^{(k)} x_j^{(k)}$$

Notice this

$\omega_{i,j}$ models the interaction between i-th and j-th features

- Assumption: matrix is symmetric $\omega_{ij} = \omega_{ji}$
- In theory one could add third degree terms, forth and so on…

$$\underset{\omega}{\operatorname{argmin}}\ E(\omega) = \underset{\omega}{\operatorname{argmin}} \left\| r^{(k)}, \tilde{r}^{(k)} \right\|$$

$$\tilde{r}^{(k)} = \omega_0 + \boldsymbol{\omega} \cdot \boldsymbol{x}^{(k)} + \boldsymbol{x}^{(k)\,T} \cdot \boldsymbol{W} \cdot \boldsymbol{x}^{(k)}$$

$\omega_0 \in \mathbb{R}$

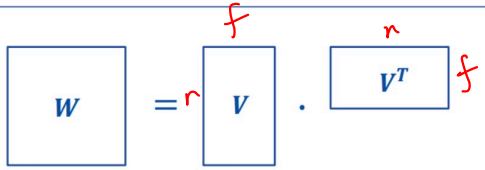
$\boldsymbol{\omega} \in \mathbb{R}^n$

$W \in \mathbb{R}^{n \times n}$

A lot!!!

Number of parameters: $1 + n + \frac{n^2 - n}{2}$

Factorizing W



$$\omega_{ij} = \boldsymbol{v}_i \cdot \boldsymbol{v}_j = \sum_{h=0}^{f} \boldsymbol{v}_{ih} \boldsymbol{v}_{jh}$$

$W \in \mathbb{R}^{n \times n}$

$V \in \mathbb{R}^{n \times f}$

where $f << n$

f = #latent factors

Final Factorization Machine Model

$$\tilde{r}^{(k)} = \omega_0 + \sum_{i=1}^{n} \omega_i\, x_i^{(k)} + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \boldsymbol{\omega_{ij}}\, x_i^{(k)} x_j^{(k)}$$

$$\boldsymbol{\omega_{ij}} = \boldsymbol{v}_i \cdot \boldsymbol{v}_j = \sum_{h=0}^{f} \boldsymbol{v}_{ih} \boldsymbol{v}_{jh}$$

$$\tilde{r}^{(k)} = \omega_0 + \sum_{i=1}^{n} \omega_i\, x_i^{(k)} + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{\boldsymbol{h=0}}^{\boldsymbol{f}} \boldsymbol{v_{ih} v_{jh}}\, x_i^{(k)} x_j^{(k)}$$

Final Number of Parameters

$$\tilde{r}^{(k)} = \omega_0 + \sum_{i=1}^{n} \omega_i \, x_i^{(k)} + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{h=0}^{f} v_{ih} v_{jh} \, x_i^{(k)} x_j^{(k)}$$

$$\tilde{r}^{(k)} = \omega_0 + \boldsymbol{\omega} \cdot \boldsymbol{x}^{(k)} + \boldsymbol{x}^{(k)\,T} \cdot \boldsymbol{V} \cdot \boldsymbol{V}^{\boldsymbol{T}} \cdot \boldsymbol{x}^{(k)}$$

Number of parameters: $1 + \mathrm{n} + nf$

Much more manageable!

Equivalence with SVD++

$$\tilde{r}^{(k)} = \omega_0 + \boldsymbol{\omega}_i + \boldsymbol{\omega}_u + \boldsymbol{V}_i \cdot \boldsymbol{V}_{\boldsymbol{u}}^{\boldsymbol{T}}$$

If we use Factorization Machines with only collaborative data, we obtain a model which is equivalent to SVD++

We need to do more!