RECOMMANDER SYSTEMS

Recommander Systems are algorithms aimed at suggesting relevant items to users (items = movies to watch, songs to listen, text to read, products to buy, ...)

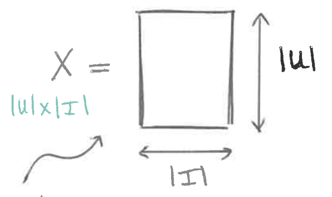
Ex: Youtube, Spotify, Amazon, Netflix, ...

There are two major paradigms for recommander systems: (i) collaborative filtering and (ii) content based methods.

- Collaborative Filtering (CF) methods.

These are based solely on past interactions recorded between users and items. These interactions are stored in a "user-item interaction" matrix X.

user $u \in U$ = set of users
item $i \in I$ = set of items
$|U|$ = # of users
$|I|$ = # of items



A sparse matrix

The main idea is that past-interactions are sufficient to detect similar users and items and to make recommendations (fill the missing values in X)

CF methods are divided into two sub-categories:

(a) Memory based approaches: no model; based on nearest neighbour searches

(b) Model based: assumes the existence of an underlying generative model that explains the user-item interactions (matrix factorization techniques).

CF suffer from the "cold-start" problem: impossible to recommend anything to new users or to recommend any new item to any user.

- Content-Based approaches.

These approaches use additional information about users and/or items.

Ex: (i) Movies: actors, directors, genre, duration, ...
(ii) User: age, sex, job, ...

↖ These are called user features & item features.

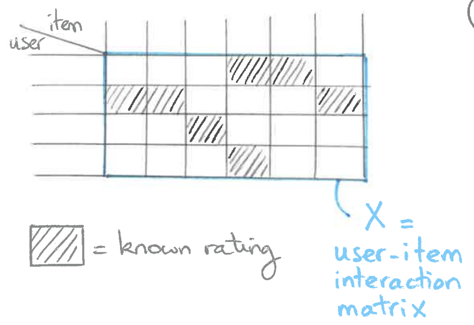
CB methods are divided into (i) classification problems (predict whether a user likes or not an item) and (ii) regression problems (predict the rating of an item). They suffer less from the cold-start problem.

I. COLLABORATIVE FILTERING

I.1. Memory-based approaches.

This category corresponds to neighborhood-based collaborative filtering methods. For a certain user, user-based k-nearest neighbour methods first identify a set of similar users, and then recommend items based on what items those similar users have purchased. Similarly, item-based k-nearest neighbour methods first identify a set of similar items for each of the items a user has purchased, and then recommend items based on those similar items.

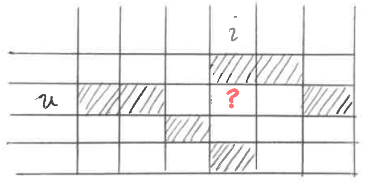
Goal: Fill missing ratings



▨ = known rating

First-Attempt:

User-User Collaborative Filtering vs Item-Item Collaborative Filt.



Estimate $\hat{r}_{ui}$ of rating r_{ui} ?

$$\hat{r}_{ui} = \frac{1}{|U_i^+|} \sum_{\substack{v \in U_i^+ \\ v \neq u}} r_{vi}$$

vs

$$\hat{r}_{ui} = \frac{1}{|I_u^+|} \sum_{\substack{j \in I_u^+ \\ j \neq i}} r_{uj}$$

$U_i^+ = \{u \in U : (u,i) \in S\}$

S = set of known ratings

$I_u^+ = \{i \in I : (u,i) \in S\}$

= set of items that user u has rated

x Better: consider a neighborhood of (u,i).

$\mathcal{N}(u;i)$ = neighborhood of u
[users similar to u that have also rated item i]

$\mathcal{N}(i;u)$ = neighborhood of i
[rated by u and similar to i]

$$\hat{r}_{ui} = \frac{1}{k} \sum_{v \in \mathcal{N}(u;i)} r_{vi}$$

$$\hat{r}_{ui} = \frac{1}{k} \sum_{j \in \mathcal{N}(i;u)} r_{uj}$$

Consider the k nearest neighbours only ($\to$ hyperparameter)

$\Rightarrow$ We need to define what it means for users to be similar, and for items to be similar.

x Option 1: Jaccard similarity

$$\text{sim}(u,v) = \frac{|r_u \cap r_v|}{|r_u \cup r_v|} = \frac{\text{\# items that both } u \,\&\, v \text{ have rated}}{\text{total \# of items rated by } u \,\&\, v \text{ together}}$$

similarity between two users u & v
(similarly define $\text{sim}(i,j)$ between two items i and j)

= "Intersection over Union".

$\hookrightarrow$ Not taking into account how users liked items; only how many they have in common.

x Option 2: (Adjusted) cosine similarity (CS)

$$\text{sim}(u,v) = \cos(r_u, r_v) = \frac{\langle r_u, r_v \rangle}{\|r_u\| \cdot \|r_v\|} = \frac{\sum_{i \in I_{u,v}} r_{ui} r_{vi}}{\sqrt{\sum_{i \in I_{u,v}} r_{u,i}^2} \sqrt{\sum_{i \in I_{u,v}} r_{v,i}^2}},$$

where $I_{u,v}$ = set of items both rated by u and v.
The adjusted CS substracts the users' average ratings.

x Option 3: Pearson similarity

$$\text{sim}(u,v) = \frac{\sum_{i \in I_{u,v}} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{u,v}} (r_{u,i} - \bar{r}_u)^2 \sum_{i \in I_{u,v}} (r_{v,i} - \bar{r}_v)^2}}$$

$\downarrow$ Rating predictions:

$$\boxed{\hat{r}_{ui} = \frac{\sum_{v \in \mathcal{N}(u;i)} \text{sim}(u,v)\, r_{vi}}{\sum_{v \in \mathcal{N}(u;i)} \text{sim}(u,v)}}$$

or

$$\boxed{\hat{r}_{ui} = \frac{\sum_{j \in \mathcal{N}(i;u)} \text{sim}(i,j)\, r_{uj}}{\sum_{j \in \mathcal{N}(i;u)} \text{sim}(i,j)}}$$

user-based CF

item-based CF

I.2. Explicit Matrix Factorization using SVD.

To gain some insight on how matrix factorization models work, we first review how PCA & SVD work.

Let X be our matrix of observations (the user-interaction matrix in our case), of size $n \times d$ ($n = |U|$ = number of users ; $d = |I|$ = number of items).

Let $\underset{(n\times d)}{X} = \underset{(n\times d)}{U}\,\underset{(d\times d)}{\Lambda}\,\underset{(d\times d)}{V^t}$ = SVD decomposition of X of rank $d < n$.

- $\underset{(n\times d)}{U}$ = has orthonormal columns $U^tU = I_d$
- $\underset{(d\times d)}{V}$ = has orthonormal columns $V^tV = I_d$
- $\underset{(d\times d)}{\Lambda}$ = diagonal matrix of singular values $= \operatorname{diag}(\lambda_1, \dots, \lambda_d)$.

↳ Recall that the j-th principal component of X is given by the eigenvector of the sample covariance matrix, associated with the j-th largest eigenvalue, see UL : PCA.

Assuming that the colums of X are centered, the sample covariance matrix $\underset{(d\times d)}{S} = \frac{1}{n}X^tX = \frac{1}{n}V\Lambda^2V^t$

$\Rightarrow$ (eigenvalue - eigenvector pairs) $= \left(\frac{1}{n}\lambda_i^2,\ v_i\right)$

i-th column of $\underset{(d\times d)}{V} = \begin{pmatrix} | & & | \\ v_1 & \cdots & v_d \\ | & & | \end{pmatrix}$

$$\boxed{\text{The PCs of } X \text{ are contained in the colums of } V, \text{ obtained from the SVD decomposition } X = U\Lambda V^t.}$$

↑ PCs have an interpretation in terms of low rank approximation of X. Indeed, it can be shown that the best (with respect

to Frobenius norm) rank r approximation to X is given by $X^* = U\Lambda^*V^t$, where Λ^* is obtained from Λ, setting the smallest $\lambda_{r+1}, \dots, \lambda_d$ singular values to zero.

Then $X^* = (Xv_1)v_1^t + \dots + (Xv_r)v_r^t = \begin{pmatrix} - & \hat{x}_1^t & - \\ & \vdots & \\ - & \hat{x}_n^t & - \end{pmatrix}$.

The j-th row of this matrix is

$\hat{x}_j = \langle x_j, v_1\rangle v_1 + \dots + \langle x_j, v_r\rangle v_r$

↑ The best rank r approximation is obtained as a linear combination of the first r principal components

Ex: suppose that X = collection of images.
Then $v_1, \dots, v_d$ are called eigenfaces. Each original image in X can be reconstructed from a linear combination of the eigenfaces.

↳ Each vector v_i / eigenface represent one specific aspect underlying the data. In an ideal world,

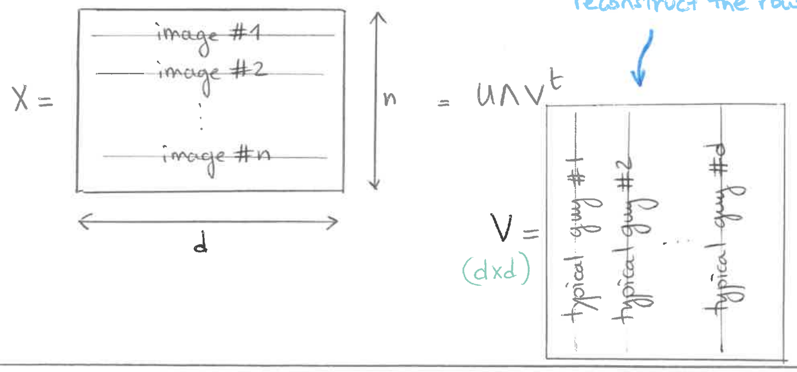
$v_1 \approx$ a typical elder person

$v_2 \approx$ a typical glasses wearer

$v_3 \approx$ a typical sad looking person, ... /...

⇓

colums of V are used to reconstruct the rows of X



⇒ If instead of a matrix of images we have a user-item interaction matrix, in an ideal world, the interpretation of the columns of V would be :

context of movie rating

v_1 = typical action movie fan

v_2 = typical romance movie fan

v_3 = typical sci-fy movie fan, .../...

$$\underset{(n \times d)}{X} = \begin{bmatrix} \text{— user 1 —} \\ \text{— user 2 —} \\ \vdots \\ \text{— user } n \text{ —} \end{bmatrix} = U \Lambda V^t \; ; \quad \underset{(d \times d)}{V} = \begin{array}{cccc} \begin{bmatrix} \text{action movie fan} & \text{romance mov. fan} & \text{sci-fi movie fan} & \cdots \end{bmatrix} \\ \uparrow v_1 \quad \uparrow v_2 \quad \uparrow v_3 \quad \uparrow v_d \end{array}$$

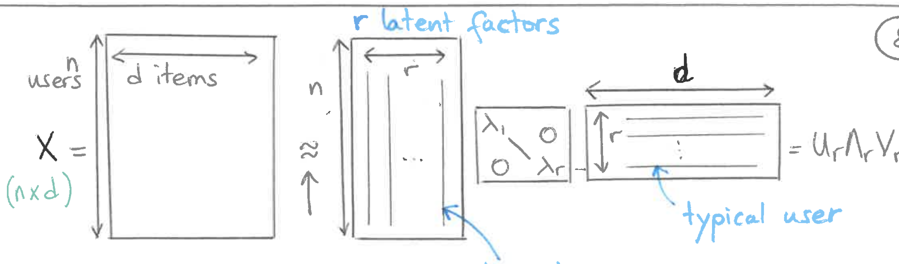
And similarly, taking the transpose of X,

$$\underset{(d \times n)}{X^t} = \begin{bmatrix} \text{— movie 1 —} \\ \text{— movie 2 —} \\ \vdots \\ \text{— movie } d \text{ —} \end{bmatrix} = V \Lambda U^t \; ; \quad \underset{(n \times d)}{U} = \begin{bmatrix} \text{typical action movie} & \text{typical romance mov} & \text{typical sci-fi mov} & \cdots \end{bmatrix}$$

The columns of U contain the eigenvalues of XX^t, related to the sample covariance matrix of X^t.

In this case, PCA does not reveal typical faces or typical users, but typical movies. These typical movies can be used to build back the original movies.

↳ SVD decomposition reveals all typical users and typical movies in one go; a very powerful tool.



keep only r PCs
$\parallel\!\!\!$
& set $\lambda_{r+1}, \dots, \lambda_d$ to zero ; or keep only the first r columns of U and V (denoted U_r, V_r)

⚠ X is usually sparse, with many missing values, and the SVD decomposition is not defined

A rating $\hat{r}_{ui}$ in X (entry corresponding to user u, item i) is expressed as a dot product between a vector $p_u \in \mathbb{R}^d$ and $q_i \in \mathbb{R}^d$ (up to a scaling factor) :

$$\hat{r}_{ui} = \langle p_u, q_i \rangle = p_u^t q_i .$$

$\equiv$ " user-item " interaction.

$$= \sum_{f \in \text{latent factor}} \text{affinity of } u \text{ for } f \times \text{affinity of } i \text{ for } f$$

I.3. Explicit Matrix Factorization: ALS, SGD, ..

The user-item interaction matrix contains many missing entries, so its SVD decomposition is undefined & cannot be computed. An easy fix would be to replace each missing value with a row mean, column mean, or 0, and compute the SVD decomposition of the filled matrix X. A better approach would be to solve the optimization problem

$$\min_{p_u, q_i} \Big\{ \sum_{r_{ui}\in X} (r_{ui} - p_u^t q_i)^2 + \lambda_p \sum_u \|p_u\|^2 + \lambda_q \sum_i \|q_i\|^2 \Big\}$$

$$\underset{\displaystyle \mathcal{L}(p_u, q_i)}{\parallel}$$

add regularization terms to prevent overfitting.

The are two popular methods for minimizing this criterion: Alternating Least Squares (ALS) and Stochastic Gradient Descent (SGD).

x Alternating Least Squares (ALS).

For ALS minimization, we hold one set of latent vectors constant (q_i, say), and optimize w.r.t. p_u (a simple LS problem). With this updated value of p_u, we hold it constant, and optimize w.r.t. q_i. And Repeat.

↳ derivation.

$$\frac{\partial \mathcal{L}}{\partial p_u} = -2 \sum_i (r_{ui} - \hat{p}_u^t q_i)^2 q_i^t + 2 \lambda_p \hat{p}_u^t = 0$$

$$\Leftrightarrow \quad - \underbrace{\sum_i (r_{ui} - \hat{p}_u^t q_i)^2 q_i^t}_{} + \lambda_p \hat{p}_u^t = 0$$

$$\sum_i r_{ui} q_i^t = (r_{u1} \cdots r_{ud}) \begin{pmatrix} - q_1^t - \\ \vdots \\ - q_d^t - \end{pmatrix} =: \underset{(1\times d)}{r_u^t} \; \underset{(d\times r)}{Q}$$

$$p_u^t \sum_i q_i q_i^t = p_u^t Q^t Q$$

$$\Leftrightarrow \hat{p}_u^t (Q^t Q + \lambda_p I_r) = r_u^t Q$$

$$\Leftrightarrow \hat{p}_u^t = r_u^t Q (Q^t Q + \lambda_p I_r)^{-1}$$

$$\Leftrightarrow \boxed{\hat{p}_u = (Q^t Q + \lambda_p I_r)^{-1} Q^t r_u}$$

And similarly,

$$\frac{\partial \mathcal{L}}{\partial q_i} = -2 \sum_u (r_{ui} - \underset{\displaystyle q_i^t p_u}{\underset{\parallel}{p_u^t \hat{q}_i}}) p_u^t + 2 \lambda_q \hat{q}_i^t = 0$$

Introducing $\underset{(n\times r)}{P} := \begin{pmatrix} - p_1^t - \\ \vdots \\ - p_n^t - \end{pmatrix}$ and $\underset{(n\times 1)}{r_i} := \begin{pmatrix} r_{u_1 i} \\ \vdots \\ r_{u_n i} \end{pmatrix}$,

we get

$$\boxed{\hat{q}_i = (P^t P + \lambda_q I_r)^{-1} P^t r_i}$$

↖ The ridge solution, see SL: RR AND LASSO

x Stochastic Gradient Descent

Here we take the derivative of the loss function with respect to each variable in the model. Samples at taken one at a time, or in batch.

We consider the more general model:

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u^t q_i \, ..$$

interaction term

global mean

user bias = some users tend to rate higher than others on average

item bias = some items are more popular than others

The loss becomes:

$$\mathcal{L}(p_u, q_i, b_u, b_i, \mu) = \sum_{r_{ui}\in X} (r_{ui} - (\mu + b_u + b_i + p_u^t q_i))^2 + \lambda_{pb} \sum_u \|b_u\|^2 + \lambda_{qb} \sum_i \|b_i\|^2 + \lambda_{pf} \sum_u \|p_u\|^2 + \lambda_{qf} \sum_i \|q_i\|^2$$

We need the gradient w.r.t. each parameter.

- $$\frac{\partial \mathcal{L}}{\partial b_u} = -\cancel{2}\underbrace{\left(r_{ui} - (\mu + b_u + b_i + p_u^t q_i)\right)}_{\overset{!!}{\varepsilon_{ui}} \text{ = current prediction error}} + \cancel{2}\lambda_{pb} b_u = 0$$

$$\frac{\partial \mathcal{L}}{\partial b_u} = 0 \iff \lambda_{pb} b_u - \varepsilon_{ui} = 0$$

- $$\frac{\partial \mathcal{L}}{\partial b_i} = 0 \iff \lambda_{qb} b_i - \varepsilon_{ui} = 0$$
- $$\frac{\partial \mathcal{L}}{\partial p_u} = 0 \iff \lambda_{pf} p_u - \varepsilon_{ui} q_i = 0$$
- $$\frac{\partial \mathcal{L}}{\partial q_i} = 0 \iff \lambda_{qf} q_i - \varepsilon_{ui} p_u = 0$$

The SGD updates are:

- $b_u \leftarrow b_u + \eta(\varepsilon_{ui} - \lambda_{pb} b_u)$
- $b_i \leftarrow b_i + \eta(\varepsilon_{ui} - \lambda_{qb} b_i)$
- $p_u \leftarrow p_u + \eta(\varepsilon_{ui} q_i - \lambda_{pf} p_u)$
- $q_i \leftarrow q_i + \eta(\varepsilon_{ui} p_u - \lambda_{qf} q_i)$

I.4. Implicit Matrix Factorization.

With implicit feedback, we do not have ratings anymore, we have user's preferences for items (e.g. a like on Instagram). The criterion to optimize must be modified accordingly. The following criterion was proposed in Collaborative Filtering for Implicit Feedback Datasets, Hu, Koren & Volinsky.

We introduce a set of binary variables d_{ui}, which indicates the preference of user u to item i. The d_{ui} are obtained by binarizing the r_{ui} values:

$$d_{ui} = \begin{cases} 1 & \text{if } r_{ui} > 0 \\ 0 & \text{otherwise} \end{cases}$$

If user u consumed item i ($r_{ui} > 0$), we have an indication that u likes it ($d_{ui} = 1$). On the other hand, if u never consumed i, we believe no preference ($d_{ui} = 0$).

However, values $d_{ui} = 0$ are associated with low confidence: not taking any positive action on an item can have many other reasons than not liking it (unaware, or unable to consume it since too pricy).

In addition $d_{ui} = 1$ can be the result of many $\neq$ reasons: gift to a friend even if they dislike the product, watching a TV show because staying on the same channel as before...

In general, as r_{ui} grows, we have a stronger indication that the user likes it.

$\Rightarrow$ Introduce a set of variables c_{ui} which indicate/measure our confidence in observing d_{ui}. The authors take

$$c_{ui} = 1 + \alpha r_{ui}$$

α: hyperparameter

r_{ui}: e.g. r_{ui} = # of clicks for item i. The more clicks, the more likely user u likes i.

$\Downarrow$

$$\mathcal{L} = \sum_{\forall (u,i)} c_{ui}(d_{ui} - p_u^t q_i)^2 + \lambda_p \sum_u \|p_u\|^2 + \lambda_q \sum_i \|q_i\|^2$$

(use ALS)

I.5. Sparse LInear Methods (SLIM).

All methods described above correspond to Matrix Factorization methods since they factorize the user-item interaction matrix X into a product of low-rank user factors and item factors. Similarly, SLIM learns a sparse matrix W by solving an $\ell_1 + \ell_2$-norm regularized optimization problem.

MF methods approximate X using $P\,Q^t$

user & item factors

SLIM = a special case setting $P \equiv X$, and $W \equiv Q^t$

SLIM does not learn user representation
$\Rightarrow$ this will simplify the learning process.
(& information is fully preserved)

to be learned, a $(d \times d)$ matrix

$$\boxed{\begin{aligned} &\underset{\substack{W \\ (d\times d)}}{\text{minimize}} \quad \frac{1}{2}\|X - XW\|_F^2 + \frac{\beta}{2}\|W\|_F^2 + \lambda\|W\|_1 \qquad X: (n\times d) \\ &\text{subject to} \quad W \geq 0 \\ &\qquad\qquad\quad \operatorname{diag}(W) = 0 \end{aligned}}$$

$\|.\|_F$ = Frobenius norm

$\|W\|_1 = \sum_{i,j} |w_{ij}|$

- ℓ_1-penalization ensures a sparse solution.
- $\operatorname{diag}(W) = 0$ ensures that x_{ij} is not used to compute $\tilde{x}_{ij} := (\tilde{X})_{ij}$; $\tilde{X} := XW$.

↳ the optimization problem can be decoupled into a set of sub-problems

$$\underset{w_j}{\text{minimize}} \quad \frac{1}{2}\|x_j - X w_j\|_2^2 + \frac{\beta}{2}\|w_j\|_2^2 + \lambda\|w_j\|_1$$

subject to $\begin{cases} w_j \geq 0 \\ w_{jj} = 0 \end{cases}$

j-th column of W

An elastic net problem (use e.g. CDA)

II. LIGHTFM: LEARNING WITH SIDE INFORMATION

A simple idea to incorporate side information: one-hot-encode each (user and/or item) feature into an attribute space, and assume that each attribute has its own latent vector.

The user vector is then decomposed into two components.

$$p_u + \sum_{a \in \mathcal{N}(u)} s_a$$

set of attributes of user u.

All p_u & s_a are learned

× Light FM model.

In the above decomposition, the user vector is the sum of two parts. LightFM treats everything as side information, or features. If we want to have a specific vector for each user, one must one-hot-encode that as a single feature for that user.

↳ latent representation of user u is $p_u := \sum_{a \in \mathcal{N}(u)} s_a$

———— " ———— of item i is $q_i := \sum_{\alpha \in \mathcal{M}(i)} u_\alpha$

Remark: (i) Without side information, one-hot-encoding users (and/or item) yields

$$\underset{(n\times n)}{\begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix}} \underset{(n\times r)}{\begin{pmatrix} — & s_1^t & — \\ & \vdots & \\ — & s_n^t & — \end{pmatrix}}.$$

aka the "feature matrix"

to be learned

(ii) With side information, consider

$$\begin{array}{c|c} \updownarrow n & \left(\begin{array}{cc:c} 1 & 0 & \\ 0 & 1 & * \end{array}\right) \\ & \xleftrightarrow{n} \quad \xleftrightarrow{\text{length of the set of user features}} \end{array}$$

This matrix is passed on to LightFM for training in user_features. The default value for user_features is nothing is passed on is the Id matrix.

Each feature is also described by a scalar bias term β_a^u & β_α^I:

$b_u = \sum_{a \in \mathcal{W}(u)} \beta_a^u \quad ; \quad b_i = \sum_{\alpha \in \mathcal{W}(i)} \beta_\alpha^I$.

x <u>LightFM model</u> is $\boxed{\hat{r}_{ui} = f(p_u^t q_i + b_u + b_i)}$

This model is able to compute recommendations for new users and items.

In the original paper (<u>Metadata Embeddings for User and Item Cold-start Recommendations</u>, Maciej Kula (2015)), the author is interested in predicting binary data: positive interactions denoted 1, and negative interactions denoted 0. For this reason, the autor takes $f(x) = \frac{e^x}{1+e^x}$ = sigmoid function.
An identity function would work well in predicting ratings.

<u>Remark</u>: If the feature sets consist solely of indicator variables for each user and item, LightFM reduces to the standard matrix factorization approach.

x <u>Criterion</u>
We discuss three criteria + learning algorithms to fit a LightFM model.

(a) <u>Likelihood</u>.
The author originally proposed to maximize the likelihood of the data:

$$\mathcal{L}(p_u, q_i, b_u, b_i) = \prod_{(u,i)\in\mathcal{S}_+} \hat{r}_{ui} \prod_{(u,i)\in\mathcal{S}_-} (1 - \hat{r}_{ui}).$$

set of positive interactions
—"— negative —"—

$\mathcal{L}$ is maximized using SGD + AdaGrad.

(b) <u>Learning to Rank – WARP / k-OS WARP</u>.
This technique and the one presented next are novel alternatives to ALS and SGD, in the context of implicit feedback.
WARP stands for <u>W</u>eighted <u>A</u>pproximate-<u>R</u>ank <u>P</u>airwise loss, and was introduced in <u>WSABIE: Scaling Up To Large Vocabulary Image Annotation</u>, J. Weston, S. Bengio & N. Usunier (2011).
It was introduced in the context of image annotation, where each image x is given a set of possible annotations $\{y\}$ to be ranked. Each image is assumed to has only one true annotation.
Specifically, let $\varphi(x) \in \mathbb{R}^d$ denote a scoring function for each annotation y associated with image x:

$$\varphi(x) = \begin{pmatrix} \varphi_1(x) \\ \vdots \\ \varphi_d(x) \end{pmatrix} \in \mathbb{R}^d$$

What we called r_{xy} before, and $\hat{r}_{xy}$ for their estimated versions.

Let $\text{rank}_y(\varphi(x))$ denote the rank of annotation y given by $\varphi(x)$:

$$\boxed{\text{rank}_y(\varphi(x)) := \sum_{j\neq y} \mathbb{1}(\varphi_y(x) \leqslant \varphi_j(x))} \text{ —— } (*)$$

top rank is zero

The rank is transformed into a loss via a function L:

$L(\text{rank}_y[\varphi(x)]) =$ loss for ranking true annotation y for user x using φ. (assuming there is only one true annotation)

where

$$L(k) := \sum_{j=1}^{k} \alpha_j \;, \quad \alpha_1 \geqslant \alpha_2 \geqslant \dots \geqslant 0$$

→ $\alpha_1 = \dots = \alpha_d = \frac{1}{d-1}$ minimizes the mean rank

→ $\alpha_1 = 1$; $\alpha_{j>1} = 0$ optimizes the propotion of top-ranked correct labels.

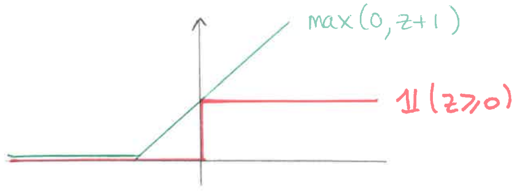
→ larger values of α in the top positions optimize the top k in the ranked list (good for optimizing precision at k).

We see from (*) that $\displaystyle\sum_{j\neq y} \frac{\mathbb{1}(\varphi_y(x) \leqslant \varphi_j(x))}{\text{rank}_y[\varphi(x)]} = 1$,

so that

$$L(\text{rank}_y[\varphi(x)]) = \left(\sum_{j\neq y} \frac{\mathbb{1}(\varphi_y(x) \leqslant \varphi_j(x))}{\text{rank}_y[\varphi(x)]}\right) \times L(\text{rank}_y[\varphi(x)])$$

$$= \sum_{j\neq y} L(\text{rank}_y[\varphi(x)]) \frac{\mathbb{1}(\varphi_y(x) \leqslant \varphi_j(x))}{\text{rank}_y[\varphi(x)]} .$$



We consider the convex surrogate given by the hinge loss to upper-bound the term $\mathbb{1}(\varphi_j(x) - \varphi_y(x) \geqslant 0)$

$$\text{loss} \leq \sum_{j\neq y} L(\text{rank}^1_y[\varphi(x)]) \frac{\max(0, \varphi_j(x) - \varphi_y(x) + 1)}{\text{rank}^1_y[\varphi(x)]}$$

where we replaced the $\text{rank}_y[\varphi(x)]$ functions by the margin-penalized rank of y:

$$\text{rank}^1_y[\varphi(x)] = \sum_{j\neq y} \mathbb{1}(\underbrace{1 + \varphi_j(x)}_{\text{margin of size } 1} > \varphi_y(x)) \quad (\geqslant \text{rank}_y[\varphi(x)])$$

The expected risk of φ is:

$$R(\varphi) := \mathbb{E}_{(X,Y)}\, \ell(Y, \varphi(X)),$$

where

$$\ell(y, \varphi(x)) := \sum_{j\neq y} L(\text{rank}^1_y[\varphi(x)]) \frac{\max(0, \varphi_j(x) - \varphi_y(x) + 1)}{\text{rank}^1_y[\varphi(x)]} .$$

Criterion to minimize.

The authors suggest the following sampling procedure

(i) Select a pair (x, y) according to $\mathbb{P}$

(ii) For the chosen pair (x, y) select a violating label $\bar{y}$ such that $1 + \varphi_{\bar{y}}(x) > \varphi_y(x)$.

Indeed, note that

$$R(\varphi) = \mathbb{E}_{X,Y}\left[\left\{\sum_{j\neq Y} L(\text{rank}^1_Y[\varphi(X)])\max(0, \varphi_j(X)-\varphi_Y(X)+1)\right\} \times \frac{1}{\text{rank}^1_Y \varphi(X)}\right]$$

$$= \mathbb{E}_{X,Y}\,\mathbb{E}_{\bar{Y}|X,Y}\left\{L(\text{rank}^1_Y[\varphi(X)])\max(0,\varphi_{\bar{Y}}(X)-\varphi_Y(X)+1)\right\}$$

$$= \mathbb{E}_{X,Y,\bar{Y}}\left\{L(\text{rank}^1_Y[\varphi(X)]\max(0,\varphi_{\bar{Y}}(X)-\varphi_Y(X)+1)\right\}$$

where

$$\mathbb{P}(\bar{Y}=j \mid X=x, Y=j) = \begin{cases} \dfrac{1}{\text{rank}^1_y[\varphi(x)]} & \text{if } 1+\varphi_j(x) > \varphi_y(x) \\ 0 & \text{otherwise} \end{cases}$$

$\uparrow$

sums to 1 since

$$\sum_{j\neq y} \mathbb{P}(\bar{Y}=j \mid X=x, Y=y) = \sum_{j:\, 1+\varphi_j(x)>\varphi_y(x)} \frac{1}{\text{rank}^1_y[\varphi(x)]} = 1$$

Goal = minimize a function $R(\varphi)$ expressed as the expected value of another one $\Rightarrow$ we can use the Robbins & Monro (1951) procedure to update the parameters β of the model

$$\beta \leftarrow \beta - \gamma_t \frac{\partial}{\partial\beta}\left\{L(\text{rank}^1_y[\varphi(x)]\max(0, \varphi_{\bar{y}}(x) - \varphi_y(x) + 1)\right\}$$

$\uparrow$ expensive to compute since $\varphi_i(x)$ must be calculated $\forall i$

$\uparrow$ Parameters governing the expression of φ.

Remarks: (i) To select $\bar{y}$ randomly, we must know in advance with labels are violators, i.e. we must compute $\varphi_i(x)$ $\forall i$. To get around this issue, the authors suggest to sample labels i uniformly with replacement until a violating label is found.

(ii) Following the procedure above, assuming there are $k = \text{rank}^1_y[\varphi(x)]$ violating labels, the number of trials N_k in the sampling step thus follows a geometric distribution with probability of success $\frac{k}{d-1}$.

Thus $\mathbb{E}\, N_k = \frac{d-1}{k}$, and we may approximate $\text{rank}^1_y[\varphi(x)]$ by $\left\lfloor \frac{d-1}{N} \right\rfloor$; N number of trials.

Since there are precisely k points that satisfy $1+\varphi_j(x) > \varphi_y(x)$.

(iii) In the original paper, it is assumed that each image is assigned to a unique label. In the context of collaborative filtering, this amounts in assuming that each user is associated with a single positive item. WARP thus ignores the fact that there are multiple positive items per user, and treat those items independently (sample a positive example $(x, \boxed{y})$ at random).

$\uparrow$ not unique

A modification of WARP, called k-os WARP (see Learning to Rank Recommendations with the k-order statistic loss, J. Weston, H. Yee, R. J. Weiss) uses the k-th positive example for any user as a basis for pairwise updates. Specifically:

- → Select a user u at random
- → Draw randomly K positive items for that user, and compute the score $\varphi_i(u)$ for each of the picked items.
- → Select as the positive example the k-th largest in the list.

hyperparameter

(c) Learning to Rank – BRP.

An alternative to WARP is Bayesian Personalized Ranking (BPR), introduced in the context of item recommendation with implicit feedback. The optimization criterion is derived from the maximum posterior estimator of the model parameters. Similar to WARP, we iteratively sample triplets (u, i, j) of seen i and unseen j items of a randomly picked user u. The procedure goes as follows:

↳ If an item i has been viewed by user u, we say that i is a positive item. We write $(u,i) \in S$

↳ If u has not viewed item j, we assume that u prefers i over j, and we write $i >_u j$.

↳ Consider the training set

$$\mathcal{L}_s := \{(u,i,j) \mid i \in I_u^+ , \; j \in I \setminus I_u^+\}$$

No preference between two items, or btw seen items, or btw to unseen items

$I_u^+ := \{i \in I \mid (u,i) \in S\}$

I = set of all items

↳ Likelihood function of the learning sample $\mathcal{L}_s$:

$$\ell(\theta \mid \mathcal{L}_s) = \prod_{(u,i,j) \in \mathcal{L}_s} p(i >_u j \mid \theta)$$

Assuming a prior $p(\theta)$ on θ, the posterior distribution is proportional to $\ell(\theta \mid \mathcal{L}_s)\, p(\theta)$.

$$\boxed{\prod_{(u,i,j) \in \mathcal{L}_s} \underbrace{p(i >_u j \mid \theta)} \underbrace{p(\theta)}}$$

$:= \sigma(\hat{x}_{uij}(\theta))$, where

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

&

$\hat{x}_{uij}(\theta)$ = arbitrary real-valued function parametrized by θ, capturing the relationship between user u and items i and j. The authors suggest to take $\hat{x}_{uij} = \hat{x}_{ui} - \hat{x}_{uj}$, so that we can apply a standard collaborative filtering model that predicts $\hat{x}_{ul}$, $\forall l \in I$

$\mathcal{N}(0, \Sigma_\theta)$ with $\Sigma_\theta = \lambda_\theta \underline{\underline{I}}$ diagonal

And indeed, user u prefers item i over j iff $\hat{x}_{uij} \geq 0$, i.e. $\hat{x}_{ui} \geq \hat{x}_{uj}$, i.e. iff $\sigma(\hat{x}_{uij}) \geq \frac{1}{2}$

The MAP estimator is then

$$\boxed{\hat{\theta}_{MAP} \in \underset{\theta}{\operatorname{argmax}} \sum_{(u,i,j) \in \mathcal{L}_s} \log \sigma(\hat{x}_{uij}(\theta)) - \lambda_\theta \|\theta\|^2}$$

Since the criterion is differentiable, we can use a gradient descent algorithm for maximization:

$$\theta \leftarrow \theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \frac{\partial}{\partial \theta} \hat{x}_{uij} + \lambda_\theta \theta \right)$$

observation (u,i,j) is chosen randomly.

The expression $\frac{\partial}{\partial\theta}\hat{x}_{uij}$ depends on the model considered.

- Ex: Matrix Factorization

Goal = estimation of a matrix X of dimension $|U| \times |I|$

How = using the approximation $X \approx WH^t$, for low-rank matrices W and H.

$|U| \times r$ $|I| \times r$

Then $\hat{x}_{ui} = \langle w_u , h_i \rangle = \sum_{\ell=1}^{r} w_{u\ell} h_{i\ell}$

row of W column of H

[The model parameters are $\theta = (W, H)$]

We obtain

$$\frac{\partial}{\partial\theta}\hat{x}_{uij} = \begin{cases} h_{i\ell} - h_{j\ell} & \text{if } \theta = w_{u\ell} \\ w_{u\ell} & \text{if } \theta = h_{i\ell} \\ -w_{u\ell} & \text{if } \theta = h_{j\ell} \\ 0 & \text{otherwise} \end{cases}$$

- Remark: Analogies to AUC optimization

Reminder: Probabilistic interpretation of (ROC) AUC

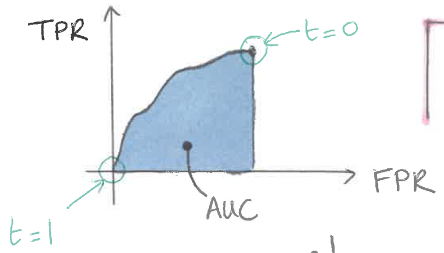
Notation: $\mathcal{L}_n = \{(X_1,Y_1),\dots,(X_n,Y_n)\}$ our learning sample, $Y_i \in \{0,1\}$, used to construct a learning function f_n, which classifies a new observed point x as 1 if $f_n(x) \geq t$, and as 0 otherwise, for some threshold t.

- TPR (aka recall / sensitivity) is

$$\frac{TP}{P} \approx \mathbb{P}(f_n(X) \geq t \mid Y=1, \mathcal{L}_n) = \bar{F}_1(t) = \int_t^1 f_1(u)\,du$$

- FPR (aka 1- specificity) is

$$\frac{FP}{N} \approx \mathbb{P}(f_n(X) \geq t \mid Y=0, \mathcal{L}_n) = \bar{F}_0(t) = \int_t^1 f_0(u)\,du$$



$$\boxed{AUC = \int_1^0 \bar{F}_1(t)\, d\bar{F}_0(t)}$$

With $\bar{F}_1(t) = \int_t^1 f_1(u)\,du$ & $d\bar{F}_0(t) = -f_0(t)\,dt$,

we observe that

$$AUC = \int_0^1 \int_t^1 f_1(u)\,du\, f_0(t)\,dt = \iint_{◺} f_0(u) f_1(t)\,du\,dt$$

$$= \mathbb{P}(f_n(X_1) \geq f_n(X_0) \mid \mathcal{L}_n, Y_0 = 0, Y_1 = 1),$$

where $(X_0, Y_0 = 0)$ and $(X_1, Y_1 = 1)$ are randomly chosen pairs.

In other words, the AUC corresponds to the probability that the score $f_n(X_1)$ is larger than $f_n(X_0)$, for a randomly generated sample $X_1 \mid Y_1 = 1$ from category 1, and a randomly generated sample $X_0 \mid Y_0 = 0$ from category 0.

In our context, $AUC(u) = \frac{1}{|I_u^+||I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \setminus I_u^+} \mathbb{1}(\hat{x}_{ui} \geq \hat{x}_{u}$

Empirical Probability

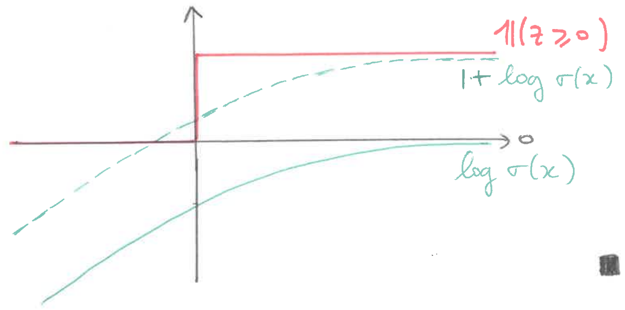
The average AUC is $\text{AUC} = \frac{1}{|U|} \sum_{u \in U} \text{AUC}(u)$

$$= \sum_{(u,i,j) \in \mathcal{D}_S} \alpha_u \underbrace{\mathbb{1}(\hat{x}_{ui} \geqslant \hat{x}_{uj})}_{\parallel} ;$$

with $\alpha_u = \frac{1}{|U| |I_u^+| |I \setminus I_u^+|}$.

$$\mathbb{1}(\hat{x}_{uij} \geqslant 0)$$

Instead of an indicator function, BRP uses the differentiable loss $\log \sigma(x)$.


$\mathbb{1}(z \geqslant 0)$
$1 + \log \sigma(x)$
0
$\log \sigma(x)$