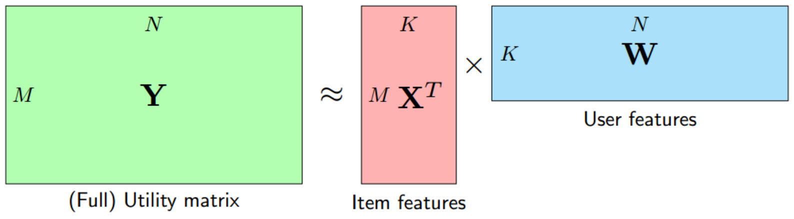
Matrix Factorization Collaborative Filtering

Introduction

- Both content-based and Neighborhood-based have their own disadvantages
  - Content-based: We have to build the item profile and build the model for each user to make (item features, # users), so we cannot recognize any relations of users
  - Neighborhood-based: We only use users or items-based at one time

$$\mathbf{Y} \approx \hat{\mathbf{Y}} = \mathbf{X}^T\mathbf{W}$$



Hình 19.1: Matrix factorization. Ma trận utility $\mathbf{Y} \in \mathbb{R}^{M\times N}$ được phân tích thành tích của hai ma trận $\mathbf{X} \in \mathbb{R}^{M\times K}$ và $\mathbf{W} \in \mathbb{R}^{K\times N}$.

- In Matrix Factorization, we don't need to build (item features, # items), MFCF train the $W \in \mathbb{R}^{K\times N}$ = User features = (item features, # users) at the same time with $X \in \mathbb{R}^{K\times M}$ = Item features = (item features, # items)
- By this way, we are trying to decompose $Y \in \mathbb{R}^{M\times N} = X^T W$, K is selected such that $K \ll M, N$, so **X** and **W** have the rank < K, so MFCF can be called **low-rank matrix factorization**
- The main idea behind MFCF is existing the latent features depicting the correlation between items and users
  - In the movie recommender system, latent features may be "hinh su", "chinh tri", "hanh dong", …or anything that we cannot name. Items will obsess or Users will like some latent features. In one users or items, the larger the value of latent feature is, the more likely items will be in or users like this latent feature
  - Collaborative Filtering, in the narrow sense, is a method of making prediction of ratings of users/ items by collecting the ratings of the other users. In general sense, CF is the process of filtering for information (ratings) using techniques involving the collaboration (interaction) among the multiple users
  - MFCF can be classified into CF because MFCF is the process of optimizing the **X** when **W** is fixed and optimizing **W** when **X** is fixed (**Content-based Process**), so MFCF is the collaborations of users when items are fixed and collaborations of items when users are fixed
- In the real datasets, amount of users and items are very large
  - NBCF does not need training process but it need to find the similarity matrix and find K most matching users/ items in every predictions
  - MFCF may be time-consuming in training process but predicting ratings may be less time-consuming than NBCF because we just need to $X^T W$, so MFCF may be more appropriate for big datasets than former approach
- MFCF helps to save the memory because we just need to save **X** and **W** so we need to store K(M + N) values, while we need to save M^2 or N^2 values to store the sim matrix in NBCF

Model

- Rating of user n on item m: $y_{mn} = \mathrm{x}_m^T \mathrm{w}_n + \mathrm{b}_m + \mathrm{d}_n$ in which $\mathrm{b} = \begin{bmatrix} b_1 & b_2 & \dots & b_M \end{bmatrix}$ is bias of optimizing **X** when **W** is fixed and $\mathrm{d} = \begin{bmatrix} d_1 & d_2 & \dots & d_N \end{bmatrix}$ is bias of optimizing **W** when **X** is fixed
- Cost function: $L(X, W, b, d) = \underbrace{\frac{1}{2s}\sum_{n=1}^{N}\sum_{m:r_{mn}=1}\left(\mathrm{x}_m^T \mathrm{w}_n + b_m + d_n - y_{mn}\right)^2}_{Data\ loss} + \underbrace{\frac{\lambda}{2}\left(\|\mathrm{X}\|_F^2 + \|W\|_F^2\right)}_{\mathrm{Re}\,gularization\ loss}$ in which
  - r_{mn} = 1 if item n is rated by user n
  - s: number of ratings
  - y_{mn}: ratings of user n on item m which is not normalized
- Optimizing (**W**, d) of cost function when (**X**, b) is fixed

  - For each user, We only get the item profile the user already rated $\hat{X}_n \in \mathbb{R}^{K\times s_n}$, $w_n \in \mathbb{R}^K$: column #n in W , the rating user rated $\hat{y}_n \in \mathbb{R}^{s_n}$, $\hat{\mathrm{b}}_n \in \mathbb{R}^{s_n}$, $d_n \in \mathbb{R}$, s_n: number of items the user rated

$$L_1\left(\mathrm{w}_n, d_n\right) = \frac{1}{2s}\left\|\hat{\mathrm{X}}_n^T \mathrm{w}_n + \hat{\mathrm{b}}_n + d_n - \hat{\mathrm{y}}_n\right\|_2^2 + \frac{\lambda}{2}\left\|\mathrm{w}_n\right\|_2^2$$

$$\rightarrow \begin{cases} \nabla_{\mathrm{w}_n} L_1 = \frac{1}{s}\hat{\mathrm{X}}_n\left(\hat{\mathrm{X}}_n^T \mathrm{w}_n + \hat{\mathrm{b}}_n + d_n - \hat{\mathrm{y}}_n\right) + \lambda \mathrm{w}_n \rightarrow \mathrm{w}_n = \mathrm{w}_n - \nabla_{\mathrm{w}_n} L_1 \\ \nabla_{d_n} L_1 = \frac{1}{s} 1^T \left(\hat{\mathrm{X}}_n^T \mathrm{w}_n + \hat{\mathrm{b}}_n + d_n - \hat{\mathrm{y}}_n\right) \rightarrow d_n = d_n - \nabla_{d_n} L_1 \end{cases}$$

- Optimizing (**X**, b) of cost function when (**W**, d) is fixed
  - For each item, we only get the user items are already rated $\hat{\mathrm{W}}_m \in \mathbb{R}^{K\times s_m}$, $\mathrm{x}_m \in \mathbb{R}^K$: column #m in X , the rating user rated $\hat{y}_m \in \mathbb{R}^{s_m}$, $b_m \in \mathbb{R}$, $\hat{\mathrm{d}}_m \in \mathbb{R}^{s_m}$, s_m :number of user items are rated

$$L_2\left(\mathrm{x}_m, b_m\right) = \frac{1}{2s}\left\|\hat{\mathrm{W}}_m^T \mathrm{x}_m + \hat{\mathrm{d}}_m + b_m - \hat{y}_m\right\|_2^2 + \frac{\lambda}{2}\left\|\mathrm{x}_m\right\|_2^2$$

$$\rightarrow \begin{cases} \nabla_{\mathrm{x}_m} L_2 = \frac{1}{s}\hat{\mathrm{W}}_m\left(\hat{\mathrm{W}}_m^T \mathrm{x}_m + \hat{\mathrm{d}}_m + b_m - \hat{y}_m\right) + \lambda \mathrm{x}_m \rightarrow \mathrm{x}_m = \mathrm{x}_m - \nabla_{\mathrm{x}_m} L_2 \\ \nabla_{b_m} L_2 = \frac{1}{s} 1^T \left(\hat{\mathrm{W}}_m^T \mathrm{x}_m + \hat{\mathrm{d}}_m + b_m - \hat{y}_m\right) \rightarrow b_m = b_m - \nabla_{b_m} L_2 \end{cases}$$

Discussion

- Nonenegative Matrix Factorization: Based on the Matrix Factorization, *users* and *items* are linked together by the latent features, so the larger the latent features are, the larger the features of *users* or *items* are. Intuitively, the latent features in *user* matrix or *item* matrix which is non-negative and equal 0 describe the irrelevance to the responding user or item. Therefore, the user or item matrix should be the non-negative matrix and several values of these matrices are 0
- Incremental Matrix Factorization: the inference process of Matrix Factorization is fast but the training process took a lot of time, especially to train the large dataset. Moreover, the utility matrix change all the time because of new user or item as well as new ratings, so the parameters need to be changed quickly most of the time in training process. Hence, the slow training process must be repeated to update the parameters. Incremental MF can solve this problem