Lec04-2. Model-based Collaborative Filtering

contents

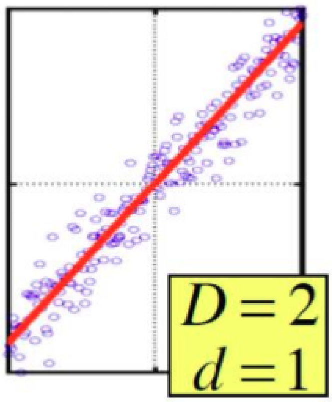
1. Basic Latent Factor Models
2. Incorporating User and Item Biases
3. Incorporating Implicit Feedback

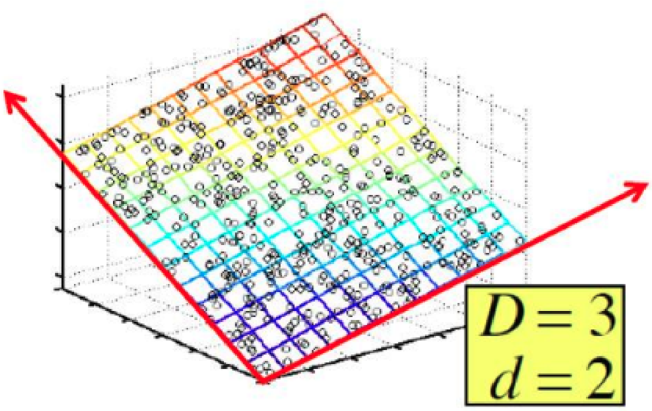
---

1. Basic Latent Factor Models

Dimensionality Reduction

- Given data in a **high D-dimensional space**, we project data into a **low d-dimensional subspace.**





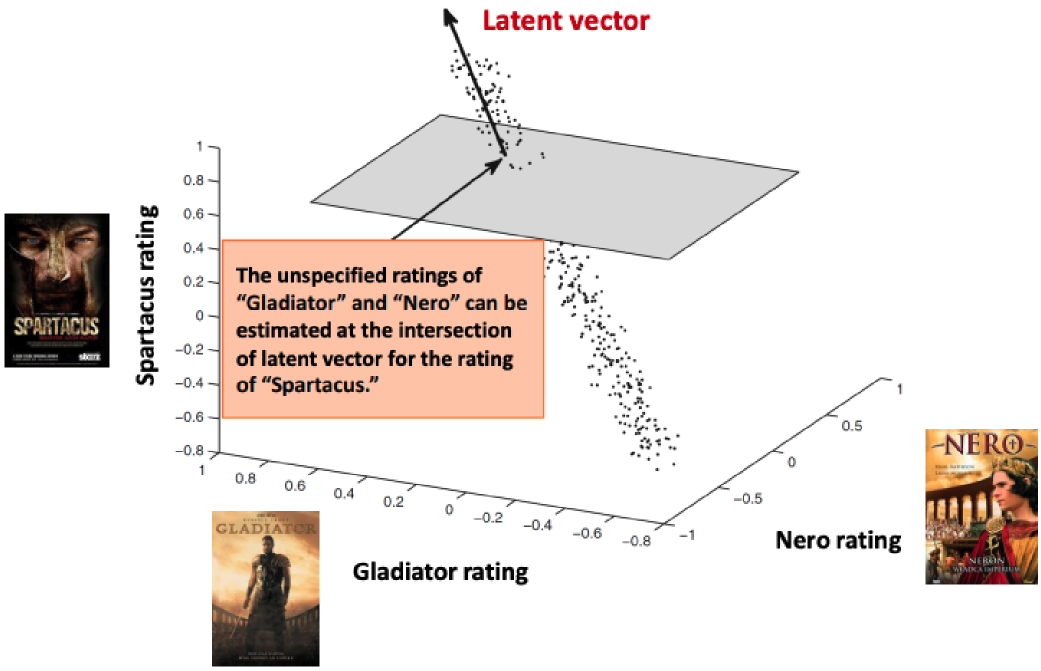
- How to represent **the axes of subspace** effectively?
  - Essential to preserve important features of data.

Why is Dimensionality Reduction?

- Key observation : **substantial portions of rows and columns** in the rating matrix are **highlt correlated.**
    - The rating matrix has **redundant rows and columns.**
    - The **fully specified low-rank approximation** can be determined with **a small subset of the entries** in the original matrix.
- Dimensionality reduction is used to **rotate the axis to remove redundant pairwise correlations.**
    - The axis represents a **latent factor** for users and items.

Geometric View : Latent Factor Models

- The **ratings of the three movies** are highly **correlated.**
    - It can be arranged along a **1-dimensional line.**



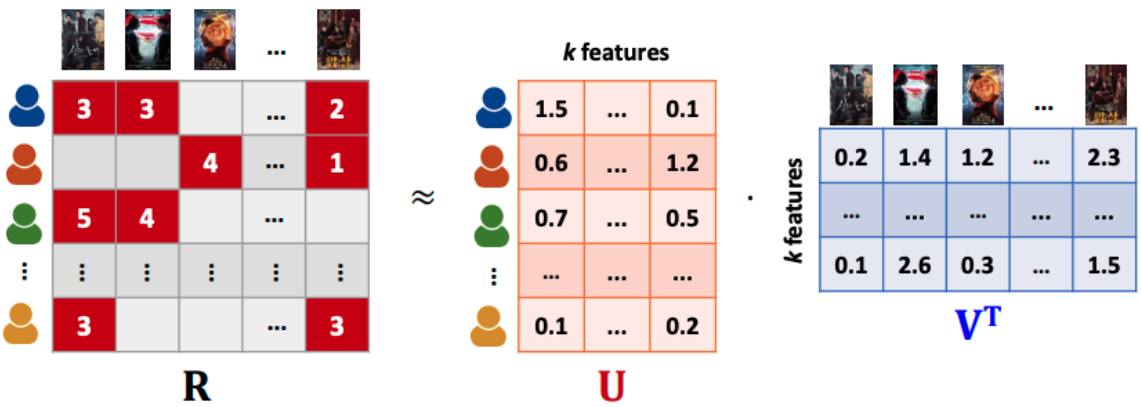
What are Latent Factor Models?

- Find a set of latent vectors, **minimizing the distant of user ratings from the hyperplane defined by the latent factors.**
    - **Capture the underlying redundancies** in the correlation structure of the data and **reconstruct all the missing values in one shot.**

- Note : If the data do not have any correlations or redundancies, the latent factor model does not work well.
- Matrix factorization is **a general solution to approximate a given matrix** for dimensionality reduction.
    - The latent vectors are **not always mutually orthogonal**.
    - SVD is the representative method of matrix factorization, in which basis vectors are orthogonal to each other.

What is Matrix Factorization?

- Given a matrix $R \in R^{m*n}$, it can be approximately expressed in the product of low rank-k factors, $k << min(m, n)$.
    - **U : latent user matrix (m x k matrix)**
        - Each user is represented by a latent vector(1 x k vector).
    - **V : latent item matrix (n x k matrix)**
        - Each item is represented by a latent vector(1 x k vector).



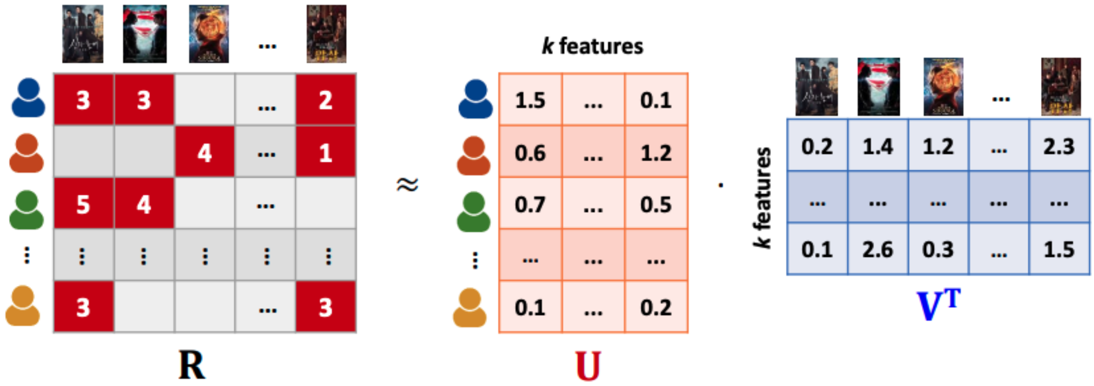
Objective Function for MF

- Factorize a matrix R into two latent matrices **U** and **V**.
    - The matrix R is approximated as a product of UV^T
    - **Note : missing ratings are ignored for model training.**

$$\min_{\mathbf{U},\mathbf{V}} \sum_{u=1}^{m} \sum_{i=1}^{n} y_{ui}\left(r_{ui} - \mathbf{u}_u \mathbf{v}_i^{\mathrm{T}}\right)^2$$

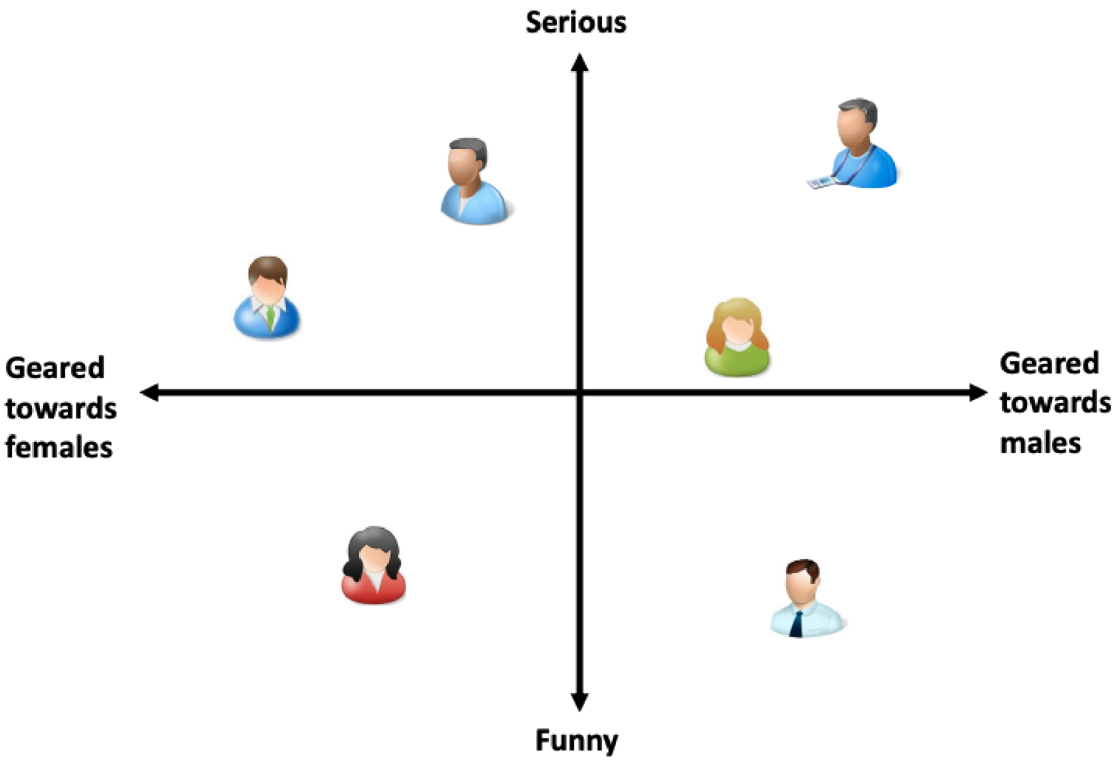
$$y_{ui} = \begin{cases} 1 & if\ r_{ui} \text{ exists} \\ 0 & \text{otherwise} \end{cases}$$

$\mathbf{u}_u$: u-th row of $\mathbf{U}$ $\quad$ $\mathbf{v}_i$: i–th row of $\mathbf{V}$



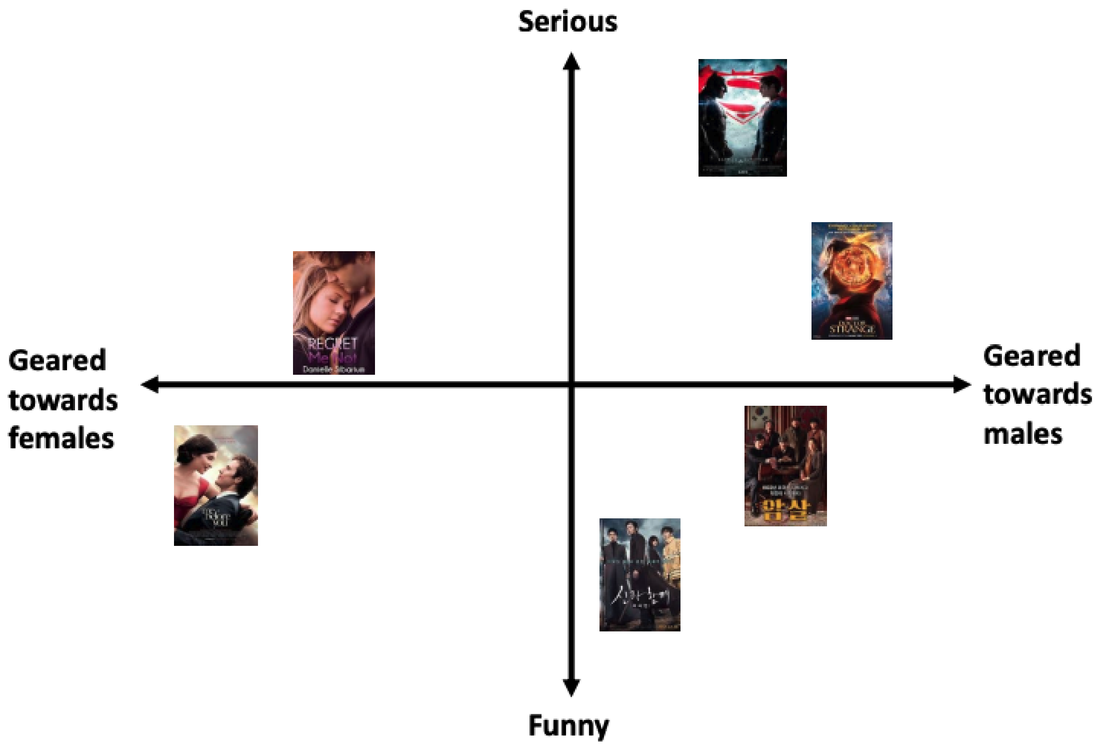
Conceptual View: Latent User Vectors

- Representing users in a latent feature space
  - Note: two factors may not be orthogonal.



Conceptual View: Latent Item Vectors

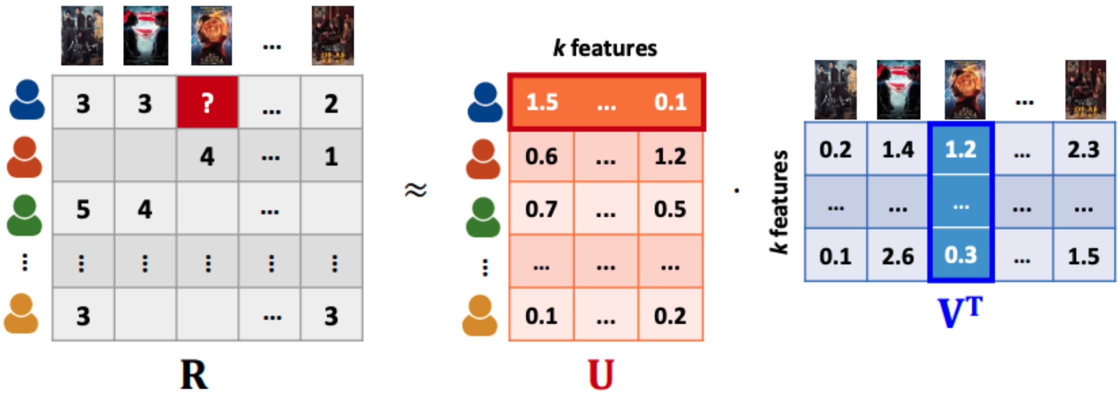
- **Representing items in a latent feature space**
  - **Note: two factors may not be orthogonal.**



Predicting Missing Ratings

- Estimate the rating of user u to item i as the product of the **latent user vector** u_u and the **latent item vector** v_i.

$$\hat{r}_{ui} = \mathbf{u}_u \mathbf{v}_i^{\mathrm{T}}$$



Unconstrained MAtrix Factorization

- Formulate an optimization problem for **U** and **V**.
  - There are no constraints on two matrices **U** and **V**.

$$\min \frac{1}{2} \sum_{(u,i)\in\mathcal{S}} e_{ui}^2 = \frac{1}{2} \sum_{(u,i)\in\mathcal{S}} \left(r_{ui} - \mathbf{u}_u \mathbf{v}_i^{\mathrm{T}}\right)^2$$

$\mathcal{S}$: a set of observed user-item pairs in $\mathbf{R}$

- Minimize **the sum of the squared error** between the observed value and the predicted value for the entry (u, i).
  - $e_{ui} = r_{ui} - \hat{r_{ui}}$, where $\hat{r_{ui}} = u_u v_i^T$

Recap: Gradient Descent (GD)

learning rate: controlling the step size

Randomly initialize parameters $\mathbf{w}^{0}$,

Repeat

$$\mathbf{w}^{\mathbf{t+1}} = \mathbf{w}^{\mathbf{t}} - \alpha \left.\frac{dE}{d\mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}^{\mathbf{t}}}$$

Until the stopping condition is satisfied

- Fixed number of iterations
- $|E(\mathbf{w}^{\mathbf{t+1}}) - E(\mathbf{w}^{\mathbf{t}})|$ is very small.

Computing Partial Derivatives

➢ **How to compute the partial derivative of E for u_{uq} and v_{iq}?**

$$\frac{\partial E}{\partial u_{uq}} = \sum_{i:(u,i)\in\mathcal{S}} \left(r_{ui} - \sum_{s=1}^{k} u_{us} \cdot v_{qs} \right) (-v_{iq}) \quad \forall u \in \{1, \dots, m\}, q \in \{1, \dots, k\}$$

$$= \sum_{i:(u,i)\in\mathcal{S}} (e_{ui}) (-v_{iq}) \quad \forall u \in \{1, \dots, m\}, q \in \{1, \dots, k\}$$

$$\frac{\partial E}{\partial v_{iq}} = \sum_{u:(u,i)\in\mathcal{S}} \left(r_{ui} - \sum_{s=1}^{k} u_{us} \cdot v_{is} \right) (-u_{uq}) \quad \forall i \in \{1, \dots, n\}, q \in \{1, \dots, k\}$$

$$= \sum_{u:(u,i)\in\mathcal{S}} (e_{ui}) (-u_{uq}) \quad \forall i \in \{1, \dots, n\}, q \in \{1, \dots, k\}$$

Training with GD

➢ **Input: rating matrix $\mathbf{R}$ and learning rate α**

Randomly initialize two matrices $\mathbf{U}$ and $\mathbf{V}$.

Let $\mathcal{S}$ = $\{(u,i): r_{ui}$ is observed$\}$.

Repeat

Compute each error e_{ui} as the observed entry of $\mathbf{R} - \mathbf{U}\mathbf{V}^{\mathbf{T}}$.

For each user-component pair (u,q) do

$$u_{uq}^{(t+1)} = u_{uq}^{(t)} - \alpha \sum_{i:(u,i)\in\mathcal{S}} e_{ui} \cdot (-v_{iq})$$

For each item-component pair (i,q) do

$$v_{iq}^{(t+1)} = v_{iq}^{(t)} - \alpha \sum_{u:(u,i)\in\mathcal{S}} e_{ui} \cdot (-u_{uq})$$

Until the stopping condition is satisfied

Training with GD

- **How to perform the updates using matrix representations?**

- **Compute the error matrix $\mathbf{E} = \mathbf{R} - \mathbf{U}\mathbf{V}^{\mathbf{T}}$.**
  - The unobserved entries of $\mathbf{E}$ are set to 0.

- **The updates can be computed as follows.**

$$\mathbf{U} = \mathbf{U} + \alpha\mathbf{E}\mathbf{V}$$
$$\mathbf{V} = \mathbf{V} + \alpha\mathbf{E}^{\mathbf{T}}\mathbf{U}$$

  - It can be executed to convergence.

Training with SGD

- **It is specific to the observed entry $(\boldsymbol{u}, \boldsymbol{i}) \in \mathcal{S}$.**
  - Update the relevant of $2k$ entries rather than $(mk + nk)$ entries.

$$e_{ui} = r_{ui} - \mathbf{u}_u\mathbf{v}_i^{\mathbf{T}}$$
$$u_{uq} \leftarrow u_{uq} + \alpha e_{ui} v_{iq} \quad \forall q \in \{1, \dots, k\}$$
$$v_{iq} \leftarrow v_{iq} + \alpha e_{ui} u_{uq} \quad \forall q \in \{1, \dots, k\}$$

- **It can be rewritten in vectorized form.**

$$\mathbf{u}_u \leftarrow \mathbf{u}_u + \alpha e_{ui}\mathbf{v}_i$$
$$\mathbf{v}_i \leftarrow \mathbf{v}_i + \alpha e_{ui}\mathbf{u}_\mathbf{u}$$

- **SGD is preferable when the data size is very large.**

Adding Regularization Terms

➢ **For model training, we introduce two terms.**
- The goodness of fit is to reduce **the prediction error**.
- The regularization term is used to alleviate the **overfitting** problem.

Least square problems

$$\min \frac{1}{2} \sum_{(u,i)\in\mathcal{S}} e_{ui}^2 = \frac{1}{2} \sum_{(u,i)\in\mathcal{S}} \left(r_{ui} - \mathbf{u}_u \mathbf{v}_i^{\mathrm{T}}\right)^2$$

⬇

Goodness of fit **Regularization**

$$\min \frac{1}{2} \sum_{(u,i)\in\mathcal{S}} \left(r_{ui} - \mathbf{u}_u \mathbf{v}_i^{\mathrm{T}}\right)^2 + \frac{\lambda}{2}\left(\sum_{u=1}^{m} \|\mathbf{u}_u\|^2 + \sum_{i=1}^{n} \|\mathbf{v}_i\|^2\right)$$

Computing Partial Derivatives

➢ **Interestingly, we can obtain almost the same results except for two terms λu_{uq} and λv_{iq}.**

$$\frac{\partial E}{\partial u_{uq}} = \sum_{i:(u,i)\in\mathcal{S}} \left(r_{ui} - \sum_{s=1}^{k} u_{us} \cdot v_{qs}\right)(-v_{iq}) + \lambda u_{uq} \quad \forall u \in \{1,\dots,m\}, q \in \{1,\dots,k\}$$

$$= \sum_{i:(u,i)\in\mathcal{S}} (e_{ui})(-v_{iq}) + \lambda u_{uq} \quad \forall u \in \{1,\dots,m\}, q \in \{1,\dots,k\}$$

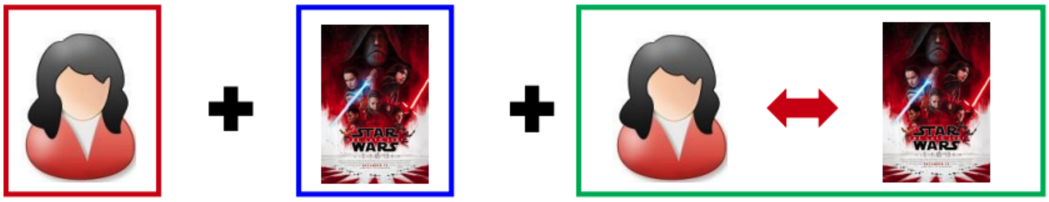
$$\frac{\partial E}{\partial v_{iq}} = \sum_{u:(u,i)\in\mathcal{S}} \left(r_{ui} - \sum_{s=1}^{k} u_{us} \cdot v_{is}\right)(-u_{uq}) + \lambda v_{iq} \quad \forall i \in \{1,\dots,n\}, q \in \{1,\dots,k\}$$

$$= \sum_{u:(u,i)\in\mathcal{S}} (e_{ui})(-u_{uq}) + \lambda v_{iq} \quad \forall i \in \{1,\dots,n\}, q \in \{1,\dots,k\}$$

2. Incorporating User and Item Biases

Modeling User Bias and Item Bias(Biased MF)

- The user rating consists of four parts:
  - μ : **global average** for all ratings
  - o_u : **user bias** for user ratings, p_i : **item bias** for item ratings
  - $u_u v_i^T$: **interaction** between user u and item i



Baseline predictor

Characterizes inherent properties of users and items.

Another predictor

Characterizes the matching between users and items.

- **Mean rating:** $\mu = 3.5$
- **Alice is a critical reviewer.**
  - Your ratings are 1 star lower than the mean: $o_u = -1.0$
- **Star Wars is a popular movie.**
  - This gets a rating of 0.5 higher than the average movie: $p_i = +0.5$
- **Predicted rating for Alice on Star Wars:** $3.5 - 1.0 + 0.5 = 3.0$

$$r_{ui} = \underbrace{\mu}_{\text{Overall mean rating}} + \underbrace{o_u}_{\text{Bias for Alice}} + \underbrace{p_i}_{\text{Bias for Star Wars}} + \underbrace{\mathbf{u}_u \mathbf{v}_i^{\mathbf{T}}}_{\text{Residual User-Movie interaction}}$$

Objective Function with Biases

- Assume that the matrix $\mathbf{R}$ is mean-centered by subtracting the global mean μ from the rating matrix.

Goodness of fit

$$\min \frac{1}{2} \sum_{(u,i)\in\mathcal{S}} \left(r_{ui} - (o_u + p_i + \mathbf{u}_u \mathbf{v}_i^{\mathrm{T}})\right)^2$$

Regularization

$$+\frac{\lambda}{2}\left(\sum_{u=1}^{m} \|\mathbf{u}_u\|^2 + \sum_{i=1}^{n} \|\mathbf{v}_i\|^2 + \sum_{u=1}^{m} \|o_u\|^2 + \sum_{i=1}^{n} \|p_i\|^2\right)$$

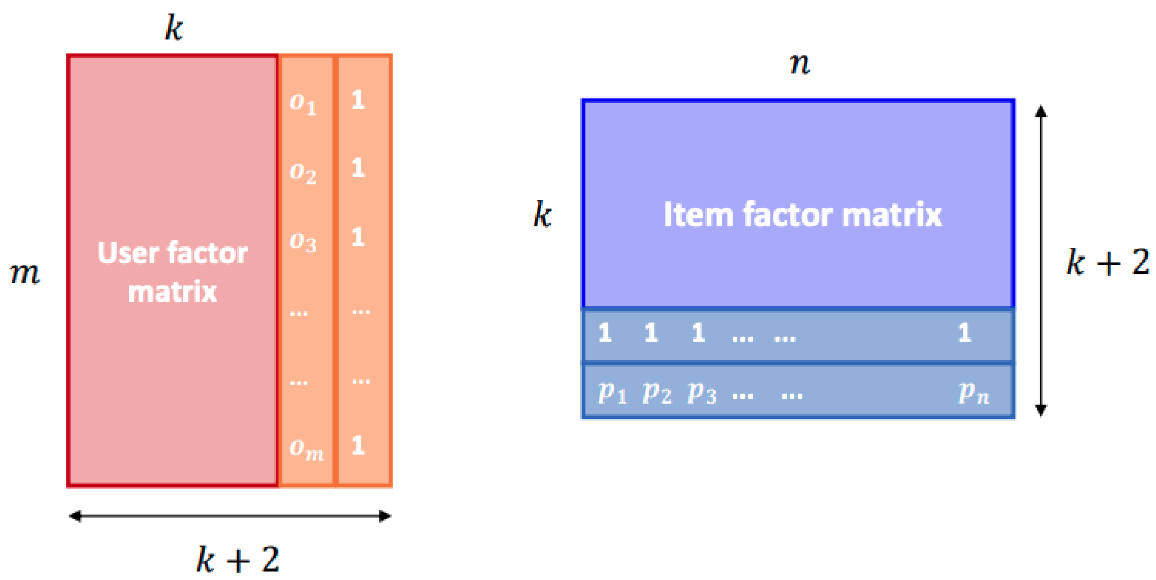
- Train with the SGD to find optimal parameters.
  - o_u, p_i, $\mathbf{u}_u$, and $\mathbf{v}_i$ are treated as the parameters to be learned.

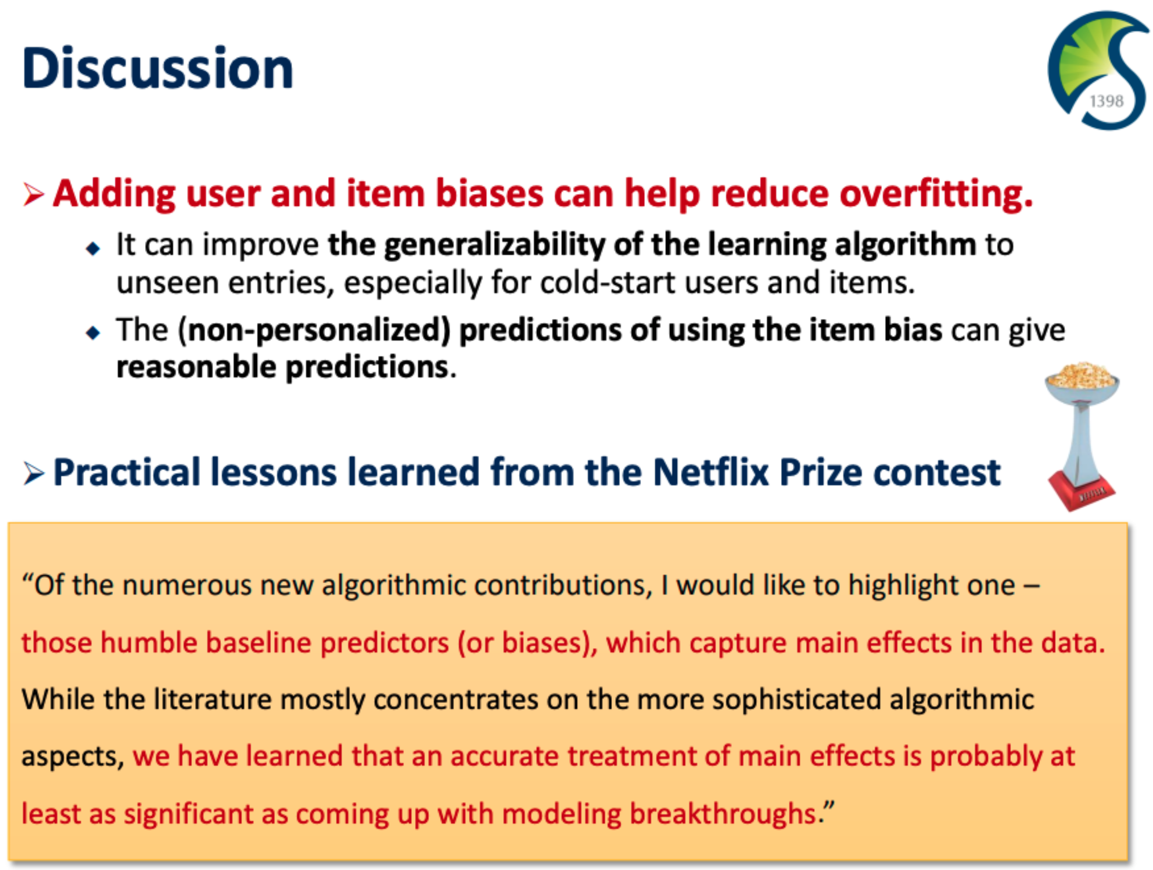
o_u : # user (m개), p_i : # item (n개), $u_u v_i^T$: (m+n) x k

Tricks to Incorporate Biases

- Instead of having seperate biases, we create larger factor matrices of size m x (k+2) and n x (k+2).

- Set **the last column** of the user factor matrix to **all 1s**.
- Set **the second last column** of the item factor matrix to **all 1s**.





---

3. Incorporating Implicit Feedback

How to Derive Implicit Feedback?

- For explicit feedback, we can derive implicit feedback.
    - Implicit feedback is captured by the identity of the rated items, regardless of actual rating values.
- **What if using two different item factor matrices?**
    - We have **explicit** and **implicit** item factors.
    - **User factors** are derived as **a linear combination of implicit item factors.**

Buildin an Implicit Feedback Matrix

- **An implicit feedback matrix is defined by the binary matrix.**
  - If a rating is observed, it is 1. otherwise, it is 0.
- **Then, it is normalized so that the L2-norm of each row is 1.**
  - Each nonzero entry in the matrix **F** is $1/\sqrt{|\mathcal{I}_u|}$, where $\mathcal{I}_u$ is a set of items rated by user u.

5	3		2
4		3	
	4		3
4			
	3		3

R

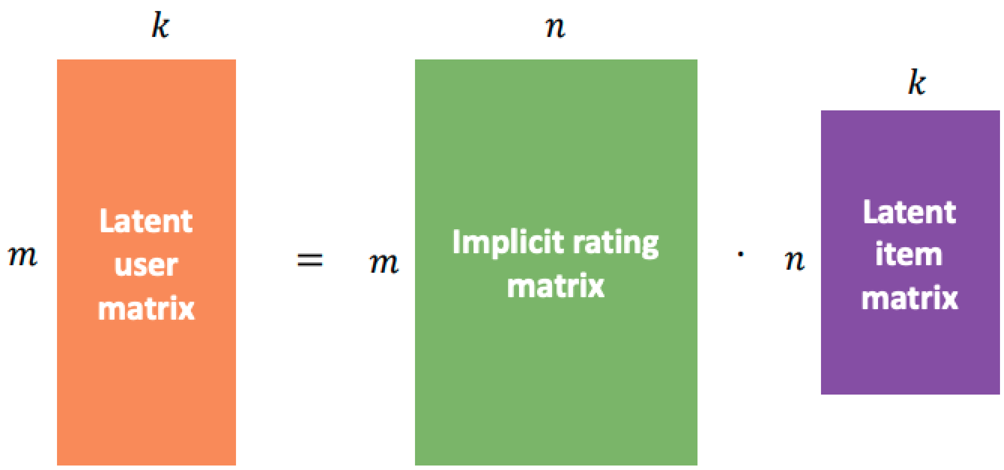
$1/\sqrt{3}$	$1/\sqrt{3}$	0	$1/\sqrt{3}$
$1/\sqrt{2}$	0	$1/\sqrt{2}$	0
0	$1/\sqrt{2}$	0	$1/\sqrt{2}$
$1/\sqrt{1}$	0	0	0
0	$1/\sqrt{2}$	0	$1/\sqrt{2}$

F

Representing Explicit Feedback

- **Assume that U is computed by a combination of implicit matrix F and implicit latent item matrix Y.**
  - The variables in Y encode **the propensity of each factor item** to contribute to implicit feedback.

$$\mathbf{R} \approx \mathbf{U}\mathbf{V}^{\mathbf{T}} = [\mathbf{F}\mathbf{Y}]\mathbf{V}^{\mathbf{T}}$$



Asymmetric Factor Models

➢ **How do we compute $\hat{r}_{ui}$?**

$$\hat{r}_{ui} = \sum_{s=1}^{k} [\mathbf{FY}]_{us} \cdot v_{is}$$

➢ **Objective function**

$$\min \frac{1}{2} \sum_{(u,i)\in\mathcal{S}} (r_{ui} - \hat{r}_{ui})^2 = \frac{1}{2} \sum_{(u,i)\in\mathcal{S}} \left(r_{ui} - \sum_{s=1}^{k} [\mathbf{FY}]_{us} \cdot v_{is} \right)^2$$

- It has two parameters $\mathbf{Y}$ and $\mathbf{V}$.
- It is trained with the gradient descent method.

Pros : Asymmetric Factor Models

- It often **provides better results** than existing latent factor models.
  - Two users will have **similar user factors** if **they have rated similar items**, regardless of their rating values.
  - It can **reduce the redundancy in user factors** by deriving them as linear combinations of item factors.
- It also supports explainability.
  - We rewrite the factorization $[FY]V^T$ as $F[YV^T]$.
  - The item-to-item prediction matrix $[YV^T]_{ij}$ tell us how much the act of rating item i contributes to the predicted rating of item j.
  - This type of explainability is inherent to **item-centric models.**

Cons: Asymmetric Factor Models

- **Deriving user factors from the identities of rated items may be an extreme case of using implicit feedback.**
  - It **does not discriminate between two users** who have **rated the same set of items** but have **very different ratings**.

- **Solution: the implicit user factor matrix $\mathbf{FY}$ is only used to adjust the explicit user factor matrix $\mathbf{U}$.**
  - That is, $\mathbf{FY}$ is added to $\mathbf{U}$ before multiplying with $\mathbf{V}^{\mathbf{T}}$.

$$\mathbf{R} \approx (\mathbf{U} + \mathbf{FY})\mathbf{V}^{\mathbf{T}}$$

Explicit user factors Implicit user factors

SVD++(=Biased MF + Asymmetric Factor Model)

- It can be view as combining the unconstrained matrix factorization model and the asymmetric factorization model.
  - The term **SVD++ is slightly misleading** because the basis vectors **are not orthogonal**. It lossely implies latent factor models.

- **How do we compute $\hat{r}_{ui}$?**
  - Note: it contains user and item biases in $\mathbf{U}$ and $\mathbf{V}$.

$$\hat{r}_{ui} = \sum_{s=1}^{k+2} (u_{us} + [\mathbf{FY}]_{us}) \cdot v_{is} = \sum_{s=1}^{k+2} \left(u_{us} + \sum_{h \in \mathcal{I}_u} \frac{y_{hs}}{\sqrt{\mathcal{I}_u}} \right) \cdot v_{is}$$

$\mathcal{I}_u$: a set of items rated by user u

  - The first term is computed by $\mathbf{UV}^{\mathbf{T}}$.
  - The second term is computed by $[\mathbf{FY}]\mathbf{V}^{\mathbf{T}}$.

Objective Function for SVD++

➢ **SVD++ has three parameters U, V, and Y with biases.**

Goodness of fit

$$\min \frac{1}{2} \sum_{(u,i)\in\mathcal{S}} \left(r_{ui} - \sum_{s=1}^{k+2} \left(u_{us} + \sum_{h\in\mathcal{I}_u} \frac{y_{hs}}{\sqrt{\mathcal{I}_u}} \right) \cdot v_{is} \right)^2$$

Regularization

$$+\frac{\lambda}{2} \sum_{s=1}^{k+1} \left(\sum_{u=1}^{m} u_{us}^2 + \sum_{i=1}^{n} v_{is}^2 + \sum_{i=1}^{n} y_{is}^2 \right)$$

- $(k+2)$th column of **U** contains only 1s.
- $(k+1)$th column of **V** contains only 1s.
- Last two columns of **Y** contain only 0s.

Training with SGD

➢ **Use the partial derivative to derive update rules.**

$$\hat{r}_{ui} = \sum_{s=1}^{k+2} \left(u_{us} + \sum_{h\in\mathcal{I}_u} \frac{y_{hs}}{\sqrt{\mathcal{I}_u}} \right) \cdot v_{is}$$

$$e_{ui} = r_{ui} - \hat{r}_{ui}$$

$$u_{uq} \leftarrow u_{uq} + \alpha\left(e_{ui} \cdot v_{iq} - \lambda \cdot u_{uq}\right) \quad \forall q \in \{1, \dots, k+2\}$$

$$v_{iq} \leftarrow v_{iq} + \alpha\left(e_{ui} \cdot \left(u_{us} + \sum_{h\in\mathcal{I}_u} \frac{y_{hs}}{\sqrt{\mathcal{I}_u}} \right) - \lambda \cdot v_{iq}\right) \quad \forall q \in \{1, \dots, k+2\}$$

$$y_{hq} \leftarrow y_{hq} + \alpha\left(\frac{e_{ui} \cdot v_{iq}}{\sqrt{\mathcal{I}_u}} - \lambda \cdot y_{hq}\right) \quad \forall q \in \{1, \dots, k+2\}, \forall h \in \mathcal{I}_u$$

➢ **Update U, V, and Y with biases with SGD.**

Training with SGD

- It can be rewritten in vectorized form.

$$\mathbf{u}_u \leftarrow \mathbf{u}_u + \alpha(e_{ui}\mathbf{v}_i - \lambda\mathbf{u}_u)$$
$$\mathbf{v}_i \leftarrow \mathbf{v}_i + \alpha\left(e_{ui}\cdot\left(u_{us} + \sum_{h\in\mathcal{I}_u}\frac{y_{hs}}{\sqrt{\mathcal{I}_u}}\right) - \lambda\mathbf{v}_i\right)$$
$$\mathbf{y}_h \leftarrow \mathbf{y}_h + \alpha\left(\frac{e_{ui}\cdot\mathbf{v}_i}{\sqrt{\mathcal{I}_u}} - \lambda\mathbf{y}_h\right) \quad \forall h\in\mathcal{I}_u$$

- These updates may be applied to the rows of **U**, **V**, and **Y**.