Low-Rank Matrix Modelling for Missing Data

Chapter Intended Learning Outcomes:

(i) Understand the basic concepts of recommender system

(ii) Know standard techniques in realizing a recommender system

(iii) Understand the importance of low-rank matrix in recommender system and related applicability

Recommender System Basics

A recommender system refers to a system that is capable of predicting the preference of a set of items for a user, and then recommend the top item(s).

In the past, we may not need recommender systems because people used to shop in a physical store with limited available items.

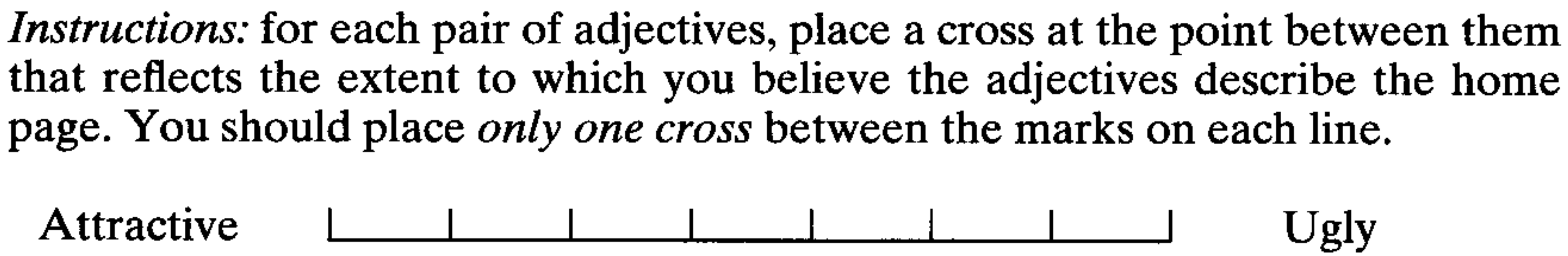
Nowadays, the internet allows people to access enormous resources online.

A new problem arose as people had a hard time selecting the items/information they actually want to buy/get. While e-commerce Web sites can use input about a customer’s interests to generate a list of recommended items.

User preference is commonly expressed as rating.

Rating can be in the form continuous and discrete data:

- Continuous ratings are specified on a continuous scale, which may correspond to the level of like or dislike of the item at hand, e.g., semantic scale:



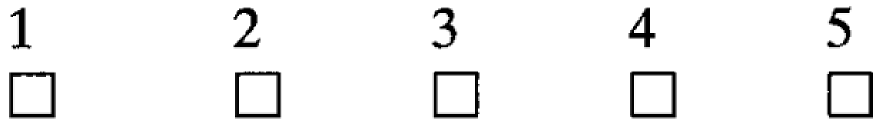
Instructions: for each pair of adjectives, place a cross at the point between them that reflects the extent to which you believe the adjectives describe the home page. You should place *only one cross* between the marks on each line.

Attractive
Ugly

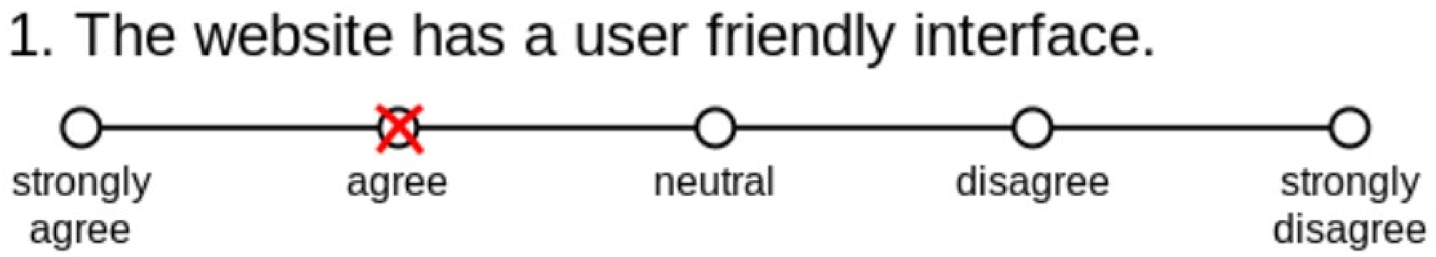
The rating can be easily converted to a number in a range, e.g., $[-5, 5]$.

It might create a burden on the user to select from an infinite number of possibilities.

- Interval-based ratings are often drawn from a 5-point scale, although other scales are also possible. Examples can be integer values from 1 to 5 or from -2 to 2, i.e., the rating values are typically equidistant.



- Ordinal ratings typically refer to ordered categorial values, e.g., 5 choices of "Strongly Disagree", "Disagree", "Neural", "Agree", and "Strongly Agree" which can map to 1, 2, 3, 4, and 5, although different distances are allowed.



Source:https://sebastiansauer.github.io/Likert-not-metric/

While for the forced choice case, neutral option is absent.

- Binary ratings allow only two options, corresponding to positive and negative responses. In case where the user is neutral, he may not specify a rating if allowed.

  It can be considered as a special case of interval-based and ordinal ratings.

- Unary ratings allow a user to specify a positive preference for an item but there is no mechanism to specify a negative preference, e.g., “Like” button on Facebook.

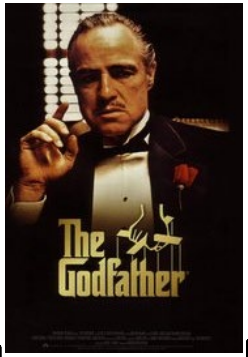
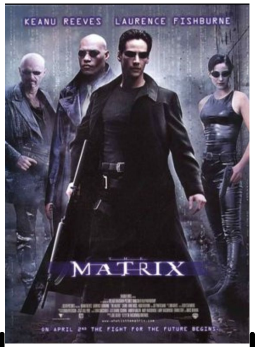
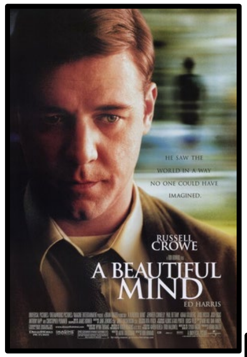
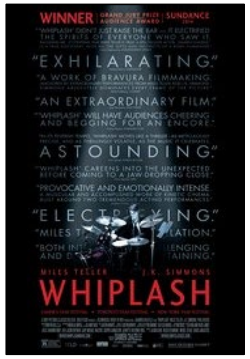
  The act of a customer buying an item can also be considered a positive vote for an item. While if he has not bought the item, it does not necessarily indicate a dislike for it.

Recommendation systems are all around us: Youtube, Amazon, Netflix, Facebook and many other such web services.

A famous example is Netflix Prize whose goal was to accurately predict user preferences for films based on all user previous ratings. 1 million could be obtained if the recommender algorithm outperformed Netflix’s internal CineMatch algorithm in offline tests by 10%.

The ratings are 1, 2, 3, 4 and 5 where 5 means “strongly recommend”.

A database of over 100 million movie ratings made by 480,189 users in 17,770 films is provided, which corresponds to the task of completing a matrix with around 99% missing entries. Note the users or films are not provided.

						• • •
Alice	1			4		
Bob		2	5			
Carol			4	5		
Dave	5				4	
⋮						

Recommend Alice to watch "Beauty and the Beast"?

Review of Least Squares for Linear Regression Model

The linear regression model is given as:

$$y = \theta_0 + \theta_1 x_1 + \cdots + \theta_L x_L + e = \boldsymbol{\theta}^T \boldsymbol{x} + e \tag{1}$$

where $\boldsymbol{\theta} = [\theta_0, \theta_1, \cdots, \theta_L]^T$ contains the parameters to be estimated, $\boldsymbol{x} = [x_0, x_1, \cdots, x_L]^T$ with $x_0 = 1$ being the known regressor, e is zero-mean additive disturbance or noise, and y is the observed value.

Using the system concept, $\boldsymbol{x}$ is the input vector while y is the observed output which is a dependent variable.

To find $\boldsymbol{\theta}$, generally we need $N \geq L+1$ observations:

$$y_n = \boldsymbol{\theta}^T \boldsymbol{x}_n + e_n, \quad n = 1, \cdots, N, \quad \boldsymbol{x}_n = [1, x_{n,1}, \cdots, x_{n,L}]^T$$

A standard problem formulation is to use least squares (LS):

$$\hat{\boldsymbol{\theta}} = \arg\min_{\tilde{\boldsymbol{\theta}}} J(\tilde{\boldsymbol{\theta}}), \quad J(\tilde{\boldsymbol{\theta}}) = \frac{1}{2}\sum_{n=1}^{N}\left(y_n - \tilde{\boldsymbol{\theta}}^T \boldsymbol{x}_n\right)^2 \tag{2}$$

where $\tilde{\boldsymbol{\theta}}$ is the optimization variable of $\boldsymbol{\theta}$. To find the solution, it is easier to express $J(\tilde{\boldsymbol{\theta}})$ using vector and matrix representation.

Let

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,L} \\ 1 & x_{2,1} & \cdots & x_{2,L} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{N,1} & \cdots & x_{N,L} \end{bmatrix} \quad \text{and} \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

We have:

$$J(\tilde{\boldsymbol{\theta}}) = 0.5(\boldsymbol{y} - \boldsymbol{X}\tilde{\boldsymbol{\theta}})^T(\boldsymbol{y} - \boldsymbol{X}\tilde{\boldsymbol{\theta}}) = 0.5||\boldsymbol{y} - \boldsymbol{X}\tilde{\boldsymbol{\theta}}||_2^2$$

Expanding $J(\boldsymbol{\theta})$ yields:

$$J(\tilde{\boldsymbol{\theta}}) = 0.5\left(\boldsymbol{y}^T\boldsymbol{y} - 2\boldsymbol{y}^T\boldsymbol{X}\tilde{\boldsymbol{\theta}} + \tilde{\boldsymbol{\theta}}^T\boldsymbol{X}^T\boldsymbol{X}\tilde{\boldsymbol{\theta}}\right)$$

Differentiating $J(\tilde{\boldsymbol{\theta}})$ w.r.t. $\tilde{\boldsymbol{\theta}}$ and then setting the resultant expression to 0, we get:

$$\left.\frac{dJ(\tilde{\boldsymbol{\theta}})}{d\tilde{\boldsymbol{\theta}}}\right|_{\tilde{\boldsymbol{\theta}}=\hat{\boldsymbol{\theta}}} = -\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\theta}} = 0 \Rightarrow \boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\theta}} = \boldsymbol{X}^T\boldsymbol{y}$$

Hence the LS solution is:

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{3}$$

Is it the global and unique solution?

Apart from using (2), we can use the gradient descent to find the solution in an iterative manner:

$$\hat{\boldsymbol{\theta}}(k+1) = \hat{\boldsymbol{\theta}}(k) - \mu \boldsymbol{\nabla}(J(\hat{\boldsymbol{\theta}}(k))) \qquad (4)$$

where $\mu > 0$ is step-size parameter while $\boldsymbol{\nabla}(J(\hat{\boldsymbol{\theta}}(k)))$ is the gradient vector, which contains the derivatives of $(J(\hat{\boldsymbol{\theta}}(k)))$ w.r.t. the estimates $\hat{\theta}_0(k), \hat{\theta}_1(k), \cdots, \hat{\theta}_L(k)$, at the kth iteration.

The algorithm is terminated when the solution converges, i.e., $\hat{\boldsymbol{\theta}}(k+1) \approx \hat{\boldsymbol{\theta}}(k)$. For the linear model (1), (4) is:

$$\hat{\boldsymbol{\theta}}(k+1) = \hat{\boldsymbol{\theta}}(k) + \mu \boldsymbol{X}^T \left(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\theta}}(k) \right) = \hat{\boldsymbol{\theta}}(k) + \mu \sum_{n=1}^{N} \left(y_n - \tilde{\boldsymbol{\theta}}(k)^T \boldsymbol{x}_n \right) \boldsymbol{x}_n$$

or

$$\hat{\theta}_l(k+1) = \hat{\theta}_l(k) + \mu \sum_{n=1}^{N} \left(y_n - \hat{\boldsymbol{\theta}}^T(k)\boldsymbol{x}_n \right) x_{n,l}, \quad l = 0, 1, \cdots, L \qquad (5)$$

Example 1

Given N samples of y_n which has the form of:

$$y_n = \theta_0 + \theta_1 n + e_n, \quad n = 1, \cdots, N$$

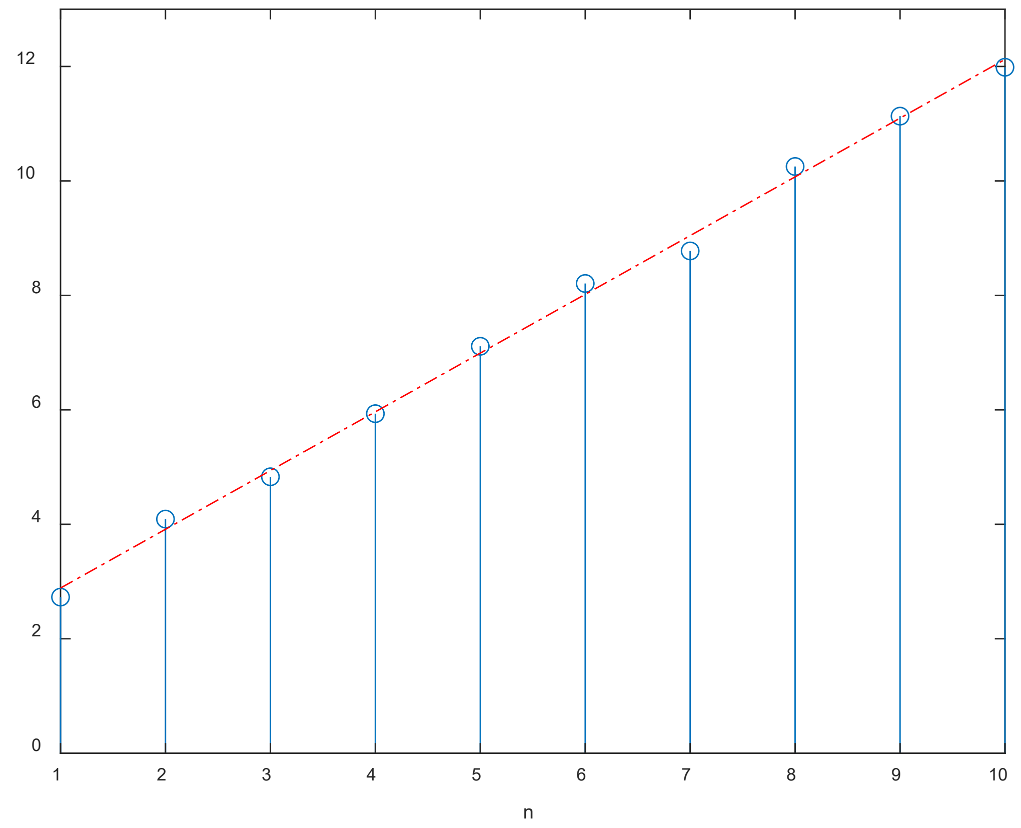
Find the LS estimates of θ_0 and θ_1.

In this case, there is only one variable $x = n$, and (3) is:

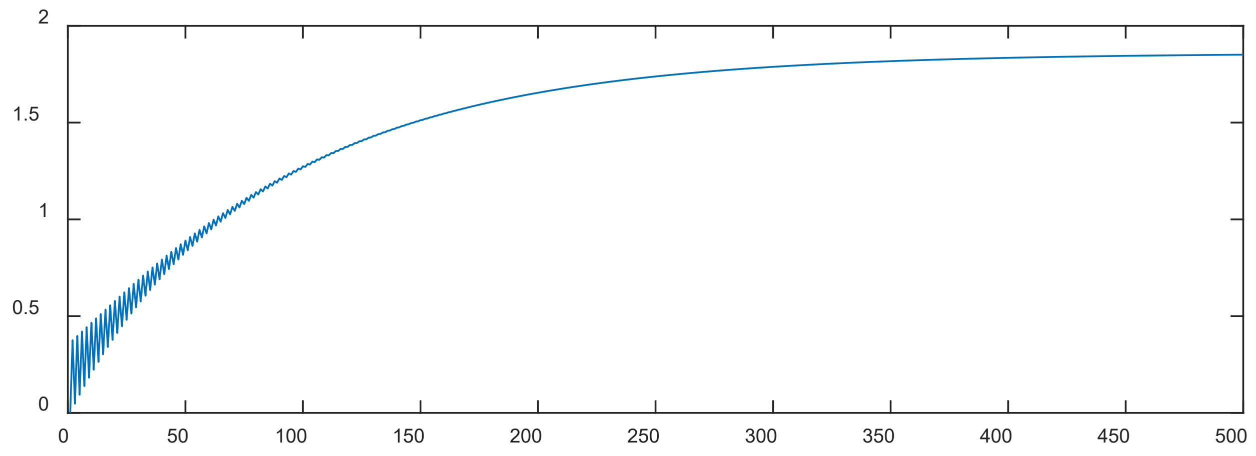
$$\begin{bmatrix} \hat{\theta}_0 \\ \hat{\theta}_1 \end{bmatrix} = \left(\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ \vdots & \vdots \\ \vdots & \vdots \\ 1 & N \end{bmatrix}^T \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ \vdots & \vdots \\ \vdots & \vdots \\ 1 & N \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ \vdots & \vdots \\ \vdots & \vdots \\ 1 & N \end{bmatrix}^T \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

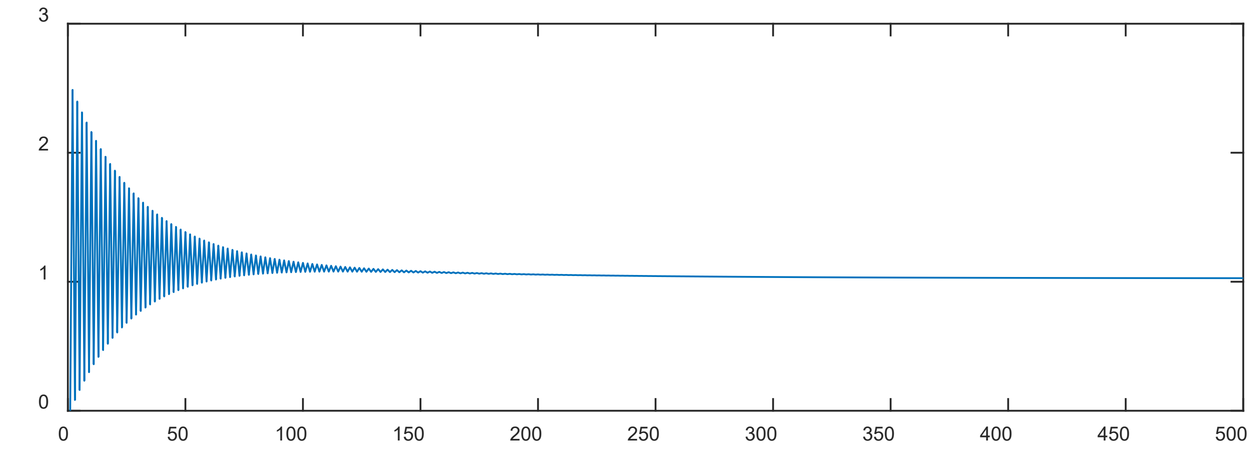
MATLAB illustration with $\theta_0 = 2$, $\theta_1 = 1$ and $N = 10$ while e_n is a zero-mean white Gaussian noise of power $\sigma_e^2 = 0.04$:

```
est =
    1.8596
    1.0264
```



For gradient descent, we choose $\mu = 0.005$, $\hat{\theta}_0(0) = \hat{\theta}_1(0) = 0$:





How about if setting μ smaller or larger?

Comparison between (3) and (4):

1. No need to choose μ	1. Need to choose μ
2. No initialization is needed	2. Initialization is needed
3. No iteration is required	3. Iterations are needed
4. Valid only for linear model or quadratic $J(\boldsymbol{\theta})$	4. More general as it works for any forms of $J(\boldsymbol{\theta})$
5. Need matrix inversion which may be a problem for large L	5. Work well even for large L
6. $\boldsymbol{X}^T\boldsymbol{X} \in \mathbb{R}^{(L+1)\times(L+1)}$ can be ill-conditioned or even non-invertible if $N < L+1$	

Regularization is a tool to impose *a priori* information on the structure of the solution. A common regularization for linear LS can be formulated as a constrained optimization problem:

$$\text{minimize}: \quad J(\tilde{\boldsymbol{\theta}}) = \frac{1}{2}\sum_{n=1}^{N}\left(y_n - \tilde{\boldsymbol{\theta}}^T \boldsymbol{x}_n\right)^2, \quad \text{subject} \quad \text{to}: \quad ||\tilde{\boldsymbol{\theta}}||_2^2 \leq \rho$$

which can equivalently be written as:

$$\text{minimize}: \quad L(\tilde{\boldsymbol{\theta}}, \lambda) = \frac{1}{2}\left(\sum_{n=1}^{N}\left(y_n - \tilde{\boldsymbol{\theta}}^T \boldsymbol{x}_n\right)^2 + \lambda||\tilde{\boldsymbol{\theta}}||_2^2\right) \qquad (6)$$

The first term measures the model misfit while the second term quantifies the size of the parameter vector norm. Formulation (6) is known as ridge regression or ℓ_2 regularization, which can be used to avoid ill-conditioning and overfitting and/or reduce mean square error (MSE).

Similar to (2), differentiating $L(\tilde{\boldsymbol{\theta}}, \lambda)$ w.r.t. $\tilde{\boldsymbol{\theta}}$ and then setting the resultant expression to 0, we get:

$$\left(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_{L+1}\right)\hat{\boldsymbol{\theta}} = \left(\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^T + \lambda \boldsymbol{I}_{L+1}\right)\hat{\boldsymbol{\theta}} = \sum_{n=1}^{N} y_n \boldsymbol{x}_n = \boldsymbol{X}^T \boldsymbol{y} \qquad (7)$$

where $\boldsymbol{I}_{L+1}$ is the $(L+1) \times (L+1)$ identity matrix.

Even $\boldsymbol{X}^T\boldsymbol{X}$ or $\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^T$ is ill-conditioned, $\left(\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^T + \lambda \boldsymbol{I}_L\right)$ will become well-conditioned for a non-zero value of $\lambda > 0$.

Due to the presence of λ, the regularized LS estimate is a biased solution, i.e., $\mathbb{E}\{\hat{\boldsymbol{\theta}}\} \neq \boldsymbol{\theta}$. Roughly speaking, if we have $N \to \infty$ observations, $\hat{\boldsymbol{\theta}} \neq \boldsymbol{\theta}$. On the other hand, the standard LS estimate (3) is an unbiased estimate.

Consider estimating a scalar θ. The MSE of the estimate $\hat{\theta}$ is:

$$\mathrm{MSE}(\hat{\theta}) = \mathbb{E}\{(\hat{\theta}-\theta)^2\} = \mathrm{var}(\hat{\theta}) + (\mathrm{bias}(\hat{\theta}))^2$$

where

$$\mathrm{var}(\hat{\theta}) = \mathbb{E}\{(\hat{\theta}-\mathbb{E}\{\hat{\theta}\})^2\}$$

and

$$\mathrm{bias}(\hat{\theta}) = \mathbb{E}\{\hat{\theta}\} - \theta$$

A biased estimator means $\mathrm{bias}(\hat{\theta}) \neq 0$ while an unbiased estimator means $\mathrm{MSE}(\hat{\theta}) = \mathrm{var}(\hat{\theta})$. Interestingly, the MSE of biased estimator can be less than that of unbiased estimator.

Example 2

Consider estimation of a single constant θ_0 given:

$$y_n = \theta_0 + e_n, \quad n = 1, \cdots, N$$

The standard LS solution can easily be computed as:

$$\hat{\theta}_0 = \frac{1}{N} \sum_{n=1}^{N} y_n$$

It is because:

$$\left. \frac{d \sum_{n=1}^{N} \left(y_n - \tilde{\theta}_0 \right)^2}{d \tilde{\theta}_0} \right|_{\tilde{\theta}_0 = \hat{\theta}_0} = 2 \sum_{n=1}^{N} \left(y_n - \hat{\theta}_0 \right) (-1) = 0 \Rightarrow \hat{\theta}_0 = \frac{1}{N} \sum_{n=1}^{N} y_n$$

While (7) gives:

$$\hat{\theta}_0^b = \frac{1}{N+\lambda}\sum_{n=1}^{N} y_n = \frac{N}{N+\lambda}\hat{\theta}_0$$

The tradeoff between MSE and unbiasedness is illustrated with a simple MATLAB simulation: $\theta_0 = 1$, $N = 10$ and e_n is i.i.d. Gaussian noise with power $\sigma_e^2 = 1$.

That is, an unbiased estimate gives a larger MSE while a biased estimate gives a smaller MSE.

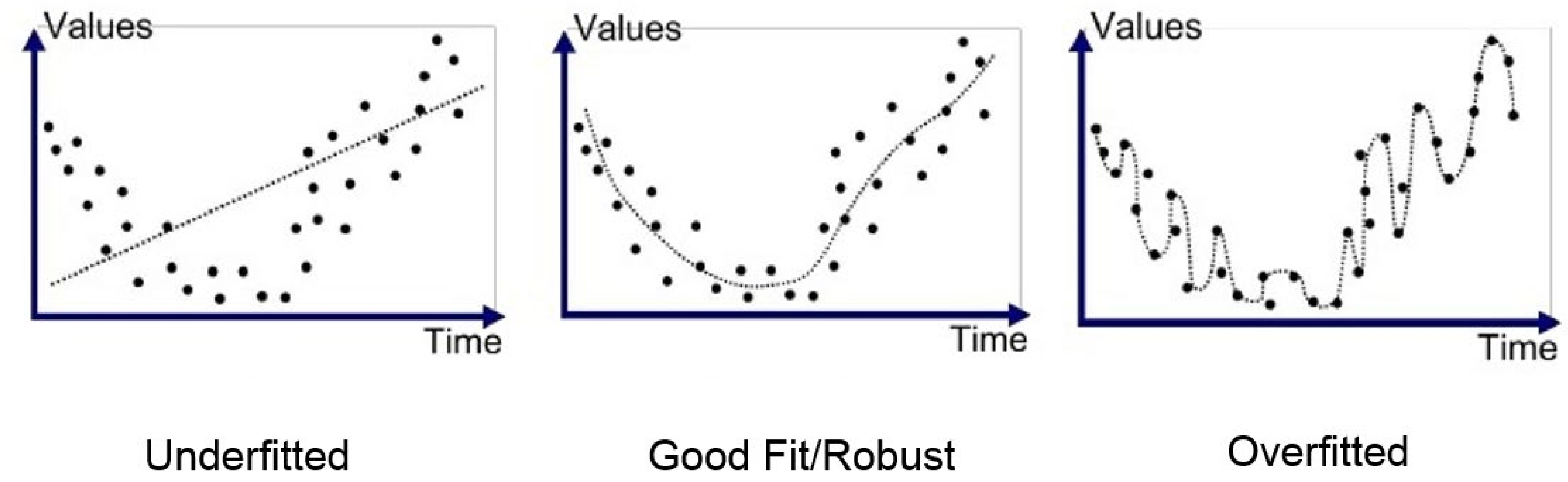
Note that 10000 independent runs are used to compute the MSE empirically.

```
>> N=10;
lambda=1;
theta=1;
```

```
for k=1:10000
    y=theta*ones(N,1)+randn(N,1);
    ls(k)=mean(y);
    rls(k)=ls(k)*N/(N+lambda);
end
[mean(ls) mean(rls)]

    1.0000    0.9091

[mean((ls-theta).^2) mean((rls-theta).^2)]

    0.0996    0.0906
```

In machine learning, when there are many variables or features, the training data can be fit very well with capturing the noise, such that $J(\hat{\boldsymbol{\theta}}) \approx 0$.

However, the model may perform badly for another set of data, which is referred to as overfitting. To address it, one direction is to reduce the variable number, while another is to use regularization whose basic idea is to decrease the magnitudes/values of all (or some) elements in $\hat{\boldsymbol{\theta}}$ and thus their impact on predicting y_n is reduced.



Source: https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76

Conventionally, regularization is not required for the bias term θ_0 in linear regression because it is intercept measuring the mean of y when $x_1 = \cdots = x_L = 0$, and (6) becomes:

$$\text{minimize}: \quad L(\tilde{\boldsymbol{\theta}}, \lambda) = \frac{1}{2}\left(\sum_{n=1}^{N}\left(y_n - \tilde{\boldsymbol{\theta}}^T \boldsymbol{x}_n\right)^2 + \lambda \sum_{l=1}^{L} \tilde{\theta}_l^2\right)$$

For the gradient descent algorithm, (5) is modified as:

$$\hat{\theta}_0(k+1) = \hat{\theta}_0(k) + \mu \sum_{n=1}^{N}\left(y_n - \hat{\boldsymbol{\theta}}^T(k)\boldsymbol{x}_n\right) x_{n,0}$$

$$\hat{\theta}_l(k+1) = \hat{\theta}_l(k) + \mu\left(\sum_{n=1}^{N}\left(y_n - \hat{\boldsymbol{\theta}}^T(k)\boldsymbol{x}_n\right) x_{n,l} - \lambda\hat{\theta}_l(k)\right), \quad l = 1, \cdots, L$$

while (7) is slightly modified as:

$$\left(\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^T + \lambda \mathrm{diag}(0, 1, \cdots, 1)\right)\hat{\boldsymbol{\theta}} = \sum_{n=1}^{N} y_n \boldsymbol{x}_n$$

Recommender System Approaches

Consider the following movie rating prediction example:

	Alice	Bob	Carol	Dave
Beauty and the Beast	5	5	0	0
Titanic	5	?	?	0
P.S. I Love You	?	4	0	?
Mission: Impossible	0	0	5	4
John Wick	0	0	5	?

Just a simple example with 5 movies and four users. For presentation simplicity, a rating of 0 is allowed.

The first three movies and the last two might be classified as romance and action categories, respectively.

Using "human" intelligence, what will you recommend for the users?

In fact, the "missing" ratings can be estimated by machine learning algorithms.

Two main approaches are content-based filtering and collaborative filtering.

Content-based Filtering

In the first approach, it is assumed that the features of each item or user are available. Suppose for each movie we have already obtained two features $0 \le x_i \le 1$, $i = 1, 2$, which measure the romance and action components, respectively:

	1. Alice	2. Bob	3. Carol	4. Dave	x_1 Romance	x_2 Action
1.Beauty and the Beast	5	5	0	0	0.9	0
2.Titanic	5	?	?	0	1.0	0.01
3.P.S. I Love You	?	4	0	?	0.99	0
4.Mission: Impossible	0	0	5	4	0.1	1.0
5.John Wick	0	0	5	?	0	0.9

Each movie is characterized by features x_1 and x_2, e.g., the first movie is characterized by the feature vector:

$$\boldsymbol{x}^{(1)} = [x_0^{(1)} \quad x_1^{(1)} \quad x_2^{(1)}]^T = [1 \quad 0.9 \quad 0]^T$$

where $x_0^{(1)} = 1$ is used for the bias term as in (1).

Now we have 5 feature vectors $\boldsymbol{x}^{(1)}$, $\boldsymbol{x}^{(2)}$, $\boldsymbol{x}^{(3)}$, $\boldsymbol{x}^{(4)}$ and $\boldsymbol{x}^{(5)}$ where their first elements are 1.

Suppose each user is characterized by parameter vector to be determined, e.g., the vector for the first user Alice is:

$$\boldsymbol{\theta}^{(1)} = [\theta_0^{(1)} \quad \theta_1^{(1)} \quad \theta_2^{(1)}]^T$$

The rating of user j on movie i is modelled as:

$$\left(\boldsymbol{\theta}^{(j)}\right)^T \boldsymbol{x}^{(i)} = \theta_0^{(j)} x_0^{(i)} + \theta_1^{(j)} x_1^{(i)} + \theta_2^{(j)} x_2^{(i)} = \theta_0^{(j)} + \theta_1^{(j)} x_1^{(i)} + \theta_2^{(j)} x_2^{(i)}$$

We expect $\left(\boldsymbol{\theta}^{(j)}\right)^T \boldsymbol{x}^{(i)}$ is close to the known entry and it is used for predicting the unrated entry, e.g.,

$$\left(\boldsymbol{\theta}^{(1)}\right)^T \boldsymbol{x}^{(1)} \approx 5$$

We may start with guessing:

$$\boldsymbol{\theta}^{(1)} = [\theta_0^{(1)} \quad \theta_1^{(1)} \quad \theta_2^{(1)}]^T = [0 \quad 5 \quad 0]^T$$

We get:

$$\left(\boldsymbol{\theta}^{(1)}\right)^T \boldsymbol{x}^{(1)} = 0 \times 1 + 5 \times 0.9 + 0 \times 0 = 4.5 \approx 5$$

$$\left(\boldsymbol{\theta}^{(1)}\right)^T \boldsymbol{x}^{(2)} = 0 \times 1 + 5 \times 1 + 0 \times 0.01 = 5$$

$$\left(\boldsymbol{\theta}^{(1)}\right)^T \boldsymbol{x}^{(4)} = 0 \times 1 + 5 \times 0.1 + 0 \times 1 = 0.5 \approx 0$$

$$\left(\boldsymbol{\theta}^{(1)}\right)^T \boldsymbol{x}^{(5)} = 0 \times 1 + 5 \times 0 + 0 \times 0.9 = 0$$

which align with the observed ratings.

Also, the prediction seems reasonable:

$$\left(\boldsymbol{\theta}^{(1)}\right)^T \boldsymbol{x}^{(3)} = 0 \times 1 + 5 \times 0.99 + 0 \times 1 = 4.95$$

In fact, computing $\boldsymbol{\theta}^{(j)}$ can be cast as a LS estimation problem:

$$\min_{\boldsymbol{\theta}^{(j)}} \frac{1}{2} \sum_{i:r(i,j)=1} \left(y^{(i,j)} - \left(\boldsymbol{\theta}^{(j)}\right)^T \boldsymbol{x}^{(i)} \right)^2$$

where $r(i,j) = 1$ if user j has rated movie i, and $y^{(i,j)}$ is the rating given by user j on movie i, which is defined only when $r(i,j) = 1$. For example, $\boldsymbol{\theta}^{(1)}$ is computed by minimizing:

$$\left(y^{(1,1)} - \left(\boldsymbol{\theta}^{(1)}\right)^T \boldsymbol{x}^{(1)} \right)^2 + \left(y^{(2,1)} - \left(\boldsymbol{\theta}^{(1)}\right)^T \boldsymbol{x}^{(2)} \right)^2 + \left(y^{(4,1)} - \left(\boldsymbol{\theta}^{(1)}\right)^T \boldsymbol{x}^{(4)} \right)^2 + \left(y^{(5,1)} - \left(\boldsymbol{\theta}^{(1)}\right)^T \boldsymbol{x}^{(5)} \right)^2$$

How to compute $\boldsymbol{\theta}^{(1)}$? What is $\boldsymbol{\theta}^{(1)}$? How about $\boldsymbol{\theta}^{(2)}$, $\boldsymbol{\theta}^{(3)}$, $\boldsymbol{\theta}^{(4)}$?

With regularization, we have:

$$\min_{\boldsymbol{\theta}^{(j)}} \frac{1}{2}\left[\sum_{i:r(i,j)=1}\left(y^{(i,j)}-\left(\boldsymbol{\theta}^{(j)}\right)^T\boldsymbol{x}^{(i)}\right)^2+\lambda\sum_{l=1}^{L}\left(\theta_l^{(j)}\right)^2\right]$$

where L is the number of features, and here $L=2$.

According to gradient descent, $\boldsymbol{\theta}^{(j)}$ is obtained via:

$$\theta_l^{(j)}(k+1)=\theta_l^{(j)}(k)+\mu\sum_{i:r(i,j)=1}\left(y^{(i,j)}-\left(\boldsymbol{\theta}^{(j)}(k)\right)^T\boldsymbol{x}^{(i)}\right)x_l^{(i)},\quad l=0$$

$$\theta_l^{(j)}(k+1)=\theta_l^{(j)}(k)+\mu\left(\sum_{i:r(i,j)=1}\left(y^{(i,j)}-\left(\boldsymbol{\theta}^{(j)}(k)\right)^T\boldsymbol{x}^{(i)}\right)x_l^{(i)}-\lambda\theta_l^{(j)}(k)\right),l\neq 0$$

To estimate all $\boldsymbol{\theta}^{(j)}$, we generalize to J users, and here $J = 4$:

$$\min_{\boldsymbol{\theta}^{(1)} \cdots \boldsymbol{\theta}^{(J)}} J(\boldsymbol{\theta}^{(1)} \ \cdots \ \boldsymbol{\theta}^{(J)})$$

where

$$J(\boldsymbol{\theta}^{(1)} \ \cdots \ \boldsymbol{\theta}^{(J)}) = \frac{1}{2}\left[\sum_{j=1}^{J} \sum_{i:r(i,j)=1} \left(y^{(i,j)} - \left(\boldsymbol{\theta}^{(j)}\right)^T \boldsymbol{x}^{(i)}\right)^2 + \lambda \sum_{j=1}^{J}\sum_{l=1}^{L}\left(\theta_l^{(j)}\right)^2\right]$$

which contains $J(L+1)$ parameters to be determined.

On the other hand, content-based filtering can also be performed if the relevant features of each user are available.

User profiles might be obtained from answers provided on a suitable questionnaire or by other means, e.g., suppose we ask each user how much they like romance and action films.

The user profiles are generated accordingly:

$$\boldsymbol{\theta}^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}, \boldsymbol{\theta}^{(2)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}, \boldsymbol{\theta}^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} \text{ and } \boldsymbol{\theta}^{(4)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$$

Alice and Bob like romance films but hate action films.

Carol and Dave hate romance films but love action films.

Similarly, for parameter vector of movie i, $\boldsymbol{x}^{(i)}$:

$$\min_{\boldsymbol{x}^{(i)}} \frac{1}{2} \left[\sum_{j:r(i,j)=1} \left(y^{(i,j)} - \left(\boldsymbol{\theta}^{(j)} \right)^T \boldsymbol{x}^{(i)} \right)^2 + \lambda \sum_{l=1}^{L} \left(x_l^{(i)} \right)^2 \right]$$

and we only sum over all j where there are data for movie i.

To estimate all $\boldsymbol{x}^{(i)}$, we compute for I users, and here $I=5$:

$$\min_{\boldsymbol{x}^{(1)} \cdots \boldsymbol{x}^{(I)}} J(\boldsymbol{x}^{(1)} \ \cdots \ \boldsymbol{x}^{(I)})$$

where

$$J(\boldsymbol{x}^{(1)} \ \cdots \ \boldsymbol{x}^{(I)}) = \frac{1}{2}\left[\sum_{i=1}^{I} \sum_{j:r(i,j)=1} \left(y^{(i,j)} - \left(\boldsymbol{\theta}^{(j)}\right)^T \boldsymbol{x}^{(i)}\right)^2 + \lambda \sum_{i=1}^{I} \sum_{l=1}^{L} \left(x_l^{(i)}\right)^2\right]$$

which contains IL parameters to be determined.

Music Genome Project, which is used for the internet radio service Pandora.com, was a successful realization:

Each song in the Music Genome Project was scored by trained music analyst based on hundreds of distinct musical characteristics. These attributes capture not only a song's musical identity but also many significant qualities that are relevant to understanding listeners' musical preferences.

To summarize, prior knowledge of item or user features is required in content-based filtering. These item or user profiles allow associating users with matching products.

However, this approach requires gathering external information that might not be available or easy to collect.

Collaborative Filtering

It does not require item and user profiles because it can learn the features for itself.

In practice, it is difficult to obtain “romance” and “action” components particularly for large number of items, and two features are insufficient, or difficult to obtain user profiles.

	1. Alice	2. Bob	3. Carol	4. Dave	x_1 Romance	x_2 Action
1.Beauty and the Beast	5	5	0	0	?	?
2.Titanic	5	?	?	0	?	?
3.P.S. I Love You	?	4	0	?	?	?
4.Mission: Impossible	0	0	5	4	?	?
5.John Wick	0	0	5	?	?	?

Recall that if $\boldsymbol{x}^{(1)} \cdots \boldsymbol{x}^{(I)}$ are given, $\boldsymbol{\theta}^{(1)} \cdots \boldsymbol{\theta}^{(J)}$ are obtained:

$$\min_{\boldsymbol{\theta}^{(1)} \cdots \boldsymbol{\theta}^{(J)}} \frac{1}{2}\left[\sum_{j=1}^{J} \sum_{i:r(i,j)=1} \left(y^{(i,j)} - \left(\boldsymbol{\theta}^{(j)}\right)^T \boldsymbol{x}^{(i)}\right)^2 + \lambda \sum_{j=1}^{J} \sum_{l=1}^{L} \left(\theta_l^{(j)}\right)^2\right] \quad (8)$$

While if $\boldsymbol{\theta}^{(1)} \cdots \boldsymbol{\theta}^{(J)}$, we can compute $\boldsymbol{x}^{(1)} \cdots \boldsymbol{x}^{(I)}$:

$$\min_{\boldsymbol{x}^{(1)} \cdots \boldsymbol{x}^{(I)}} \frac{1}{2}\left[\sum_{i=1}^{I} \sum_{j:r(i,j)=1} \left(y^{(i,j)} - \left(\boldsymbol{\theta}^{(j)}\right)^T \boldsymbol{x}^{(i)}\right)^2 + \lambda \sum_{i=1}^{I} \sum_{l=1}^{L} \left(x_l^{(i)}\right)^2\right] \quad (9)$$

How can we solve the problem when $\boldsymbol{x}^{(1)} \cdots \boldsymbol{x}^{(I)}$ and $\boldsymbol{\theta}^{(1)} \cdots \boldsymbol{\theta}^{(J)}$ are not known?

The idea of collaborative filtering is to randomly initialize $\boldsymbol{x}^{(1)} \cdots \boldsymbol{x}^{(I)}$, and solve $\boldsymbol{\theta}^{(1)} \cdots \boldsymbol{\theta}^{(J)}$ using (8), then refine $\boldsymbol{x}^{(1)} \cdots \boldsymbol{x}^{(I)}$ using (9), and so on.

Interestingly, alternating updates of (8) and (9) work and the algorithm will converge to a reasonable set of parameters.

All observed user ratings are utilized in a collaborative manner to produce the user and item profiles.

For efficient computation, we can combine (8) and (9):

$$\min_{\boldsymbol{\theta}^{(1)} \cdots \boldsymbol{\theta}^{(J)}, \boldsymbol{x}^{(1)} \cdots \boldsymbol{x}^{(J)}} J(\boldsymbol{\theta}^{(1)} \ \cdots \ \boldsymbol{\theta}^{(J)}, \boldsymbol{x}^{(1)} \ \cdots \ \boldsymbol{x}^{(J)})$$

And the cost function is now (ignore the scalar of 0.5):

$$\sum_{(i,j):r(i,j)=1} \left(y^{(i,j)} - \left(\boldsymbol{\theta}^{(j)}\right)^T \boldsymbol{x}^{(i)} \right)^2 + \lambda \left[\sum_{j=1}^{J} \sum_{l=1}^{L} \left(\theta_l^{(j)}\right)^2 + \sum_{i=1}^{I} \sum_{l=1}^{L} \left(x_l^{(i)}\right)^2 \right] \quad (10)$$

Note that:

$$\sum_{(i,j):r(i,j)=1}\left(y^{(i,j)}-\left(\boldsymbol{\theta}^{(j)}\right)^T\boldsymbol{x}^{(i)}\right)^2=\sum_{j=1}^{J}\sum_{i:r(i,j)=1}\left(y^{(i,j)}-\left(\boldsymbol{\theta}^{(j)}\right)^T\boldsymbol{x}^{(i)}\right)^2$$
$$=\sum_{i=1}^{I}\sum_{j:r(i,j)=1}\left(y^{(i,j)}-\left(\boldsymbol{\theta}^{(j)}\right)^T\boldsymbol{x}^{(i)}\right)^2$$

which means summing all pairs (i,j) for which $r(i,j)=1$, i.e., summing over every movie rated by all contributed users.

In this formulation, we are able to find $\boldsymbol{x}^{(1)}\cdots\boldsymbol{x}^{(I)}$, $\boldsymbol{\theta}^{(1)}\cdots\boldsymbol{\theta}^{(J)}$ simultaneously with initializing all of them via gradient descent:

$$\theta_l^{(j)}(k+1) = \theta_l^{(j)}(k) + \mu \left(\sum_{i:r(i,j)=1} \left(y^{(i,j)} - \left(\boldsymbol{\theta}^{(j)}(k)\right)^T \boldsymbol{x}^{(i)} \right) x_l^{(i)} - \lambda \theta_l^{(j)}(k) \right)$$

$$x_l^{(i)}(k+1) = x_l^{(i)}(k) + \mu \left(\sum_{j:r(i,j)=1} \left(y^{(i,j)} - \left(\boldsymbol{\theta}^{(j)}(k)\right)^T \boldsymbol{x}^{(i)} \right) \theta_l^{(j)} - \lambda x_l^{(i)}(k) \right)$$

Here, we remove $x_0^{(l)}$ and $\theta_0^{(l)}$, and regularize all parameters because if a bias term is needed, one $x_l^{(l)}\theta_l^{(l)}$ will learn for it automatically.

Upon convergence, will $\boldsymbol{x}^{(1)} \cdots \boldsymbol{x}^{(I)}$, $\boldsymbol{\theta}^{(1)} \cdots \boldsymbol{\theta}^{(J)}$ converge to a unique solution? Why?

The predicted rating for user j on movie i is computed as

$$\left(\boldsymbol{\theta}^{(j)}\right)^T \boldsymbol{x}^{(i)}$$

For the above example, we represent it using a matrix:

$$\boldsymbol{Y} = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & ? \end{bmatrix}$$

More generally, we can express Y in terms of $\left(\boldsymbol{\theta}^{(j)}\right)^T \boldsymbol{x}^{(i)}$:

$$\boldsymbol{Y} = \begin{bmatrix} \left(\boldsymbol{\theta}^{(1)}\right)^T \boldsymbol{x}^{(1)} & \left(\boldsymbol{\theta}^{(2)}\right)^T \boldsymbol{x}^{(1)} & \cdots & \left(\boldsymbol{\theta}^{(J)}\right)^T \boldsymbol{x}^{(1)} \\ \left(\boldsymbol{\theta}^{(1)}\right)^T \boldsymbol{x}^{(2)} & \left(\boldsymbol{\theta}^{(2)}\right)^T \boldsymbol{x}^{(2)} & \cdots & \left(\boldsymbol{\theta}^{(J)}\right)^T \boldsymbol{x}^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ \left(\boldsymbol{\theta}^{(1)}\right)^T \boldsymbol{x}^{(I)} & \left(\boldsymbol{\theta}^{(2)}\right)^T \boldsymbol{x}^{(I)} & \cdots & \left(\boldsymbol{\theta}^{(J)}\right)^T \boldsymbol{x}^{(I)} \end{bmatrix} \in \mathbb{R}^{I \times J}$$

with some unknown $\left\{\left(\boldsymbol{\theta}^{(j)}\right)^T \boldsymbol{x}^{(i)}\right\}$.

It is observed that Y can be expressed as product of two matrices:

$$\boldsymbol{Y} = \boldsymbol{X}^T\boldsymbol{\Theta} \tag{11}$$

where

$$\boldsymbol{\Theta} = [\boldsymbol{\theta}^{(1)}, \ \cdots \ , \ \boldsymbol{\theta}^{(J)}] \in \mathbb{R}^{L\times J} \text{ and } \boldsymbol{X} = [\boldsymbol{x}^{(1)}, \ \cdots \ , \ \boldsymbol{x}^{(I)}] \in \mathbb{R}^{L\times I}$$

What are the ranks of Θ and X? What is the rank of Y?

Now we may rewrite (10) in a more compact form:

$$\min_{\boldsymbol{\Theta},\boldsymbol{X}} \sum_{\text{observed } y^{(i,j)}} \left(y^{(i,j)} - \left(\boldsymbol{\theta}^{(j)}\right)^T \boldsymbol{x}^{(i)}\right)^2 + \lambda\left[\sum_{j=1}^{J} ||\boldsymbol{\theta}^{(j)}||_2^2 + \sum_{i=1}^{I} ||\boldsymbol{x}^{(i)}||_2^2\right] \tag{12}$$

Instead of using gradient descent, we may solve (12) as in (6)-(7).

Again, we may not be able to find the global solution of Θ and X as the cost function contain local minima.

However, if we fix Θ (or X), a unique for X (or Θ) can be determined.

That is, we can perform the estimation of Θ and X in an alternating manner until parameter convergence. Note that it is different from the one-step solution in (7).

We refer this procedure to as alternating LS (ALS), and the algorithm is presented as follows:

```
Input: observed $y^{(i,j)}$, $L$, $\lambda$
    initialize $X$
    repeat
        for $j = 1, \cdots, J$ do:
            $\boldsymbol{\theta}^{(j)} = \left( \sum_{\text{observed } y^{(i,j)}} \boldsymbol{x}^{(i)} \left( \boldsymbol{x}^{(i)} \right)^T + \lambda \boldsymbol{I} \right)^{-1} \sum_{\text{observed } y^{(i,j)}} y^{(i,j)} \boldsymbol{x}^{(i)}$
        end for

        for $i = 1, \cdots, I$ do:
            $\boldsymbol{x}^{(i)} = \left( \sum_{\text{observed } y^{(i,j)}} \boldsymbol{\theta}^{(j)} \left( \boldsymbol{\theta}^{(j)} \right)^T + \lambda \boldsymbol{I} \right)^{-1} \sum_{\text{observed } y^{(i,j)}} y^{(i,j)} \boldsymbol{\theta}^{(j)}$
        end for
    until convergence
Output: $X^T \Theta$
```

As the number of features $L << \min(I, J)$, we can view collaborative filtering as low-rank matrix completion problem.

Note that there is no need to care about what the feature vectors $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \cdots$, correspond to romance, action, comedy, …, although we may do investigation on this, if not impossible. This approach is also known as latent factor modelling where $\{\boldsymbol{x}^{(i)}\}$ and $\{\boldsymbol{\theta}^{(j)}\}$ are latent feature vectors.

This formulation can resolve Θ and $\boldsymbol{X}$ without knowing the user and item profiles. That is, it is domain-free.

We may expect the users A and B might be similar if $||\boldsymbol{\theta}^{(A)} - \boldsymbol{\theta}^{(B)}||_2^2$ is small.

On the other hand, we may measure the similarity between two movies C and D by checking $||\boldsymbol{x}^{(C)} - \boldsymbol{x}^{(D)}||_2^2$.

The low-rank matrix completion problem can also be formulated and solved as follows.

Let $\boldsymbol{X}_\Omega \in \mathbb{R}^{n_1 \times n_2}$ be a matrix with missing entries:

$$[\boldsymbol{X}_\Omega]_{ij} = \begin{cases} \boldsymbol{X}_{ij} = x_{i,j}, & \text{if } (i,j) \in \Omega \\ 0, & \text{otherwise} \end{cases}$$

where Ω is a known subset of the complete set of entries while the unknown entries are assumed zero.

Low-rank matrix completion refers to finding $\boldsymbol{M} \in \mathbb{R}^{n_1 \times n_2}$, given the incomplete observations $\boldsymbol{X}_\Omega$ with the low-rank information of $\boldsymbol{X}$, which can be mathematically formulated as:

$$\min_{\boldsymbol{M}} \ \text{rank}(\boldsymbol{M}), \quad \text{s.t.} \ \|\boldsymbol{M}_\Omega - \boldsymbol{X}_\Omega\|_F \leq \epsilon_F, \quad \epsilon_F \geq 0$$

That is, among all matrices best fit with the observed entries, we look for the one with minimum rank.

However, it is difficult because the rank minimization problem is computationally infeasible especially for large data size.

A practical solution is to approximate the rank by nuclear norm for convex optimization:

$$\min_{\boldsymbol{M}} \ \|\boldsymbol{M}\|_*, \quad \text{s.t.} \ \|\boldsymbol{M}_\Omega - \boldsymbol{X}_\Omega\|_F \leq \epsilon_F$$

where $||\boldsymbol{M}||_*$ equals the sum of singular values of $\boldsymbol{M}$. This approach can automatically determine the rank but the complexity of nuclear norm minimization is high.

On the other hand, low-rank matrix factorization is computationally simpler:

$$\min_{\boldsymbol{U},\boldsymbol{V}} J(\boldsymbol{U},\boldsymbol{V}) := \|(\boldsymbol{U}\boldsymbol{V})_\Omega - \boldsymbol{X}_\Omega\|_F^2 \tag{13}$$

where $\boldsymbol{U} \in \mathbb{R}^{n_1 \times r}$ and $\boldsymbol{V} \in \mathbb{R}^{r \times n_2}$, which is basically the same as (12) with $\lambda = 0$, but we need to know r.

Recall $\|\cdot\|_F$ denotes the element-wise ℓ_2-norm:

$$\|\boldsymbol{X}_\Omega\|_F = \left(\sum_{(i,j)\in\Omega} |\boldsymbol{X}_{ij}|^2 \right)^{1/2}$$

and here only the observed entries are involved.
To solve (13), we apply alternating minimization strategy:

$$\boldsymbol{V}^{k+1} = \arg\min_{\boldsymbol{V}} \ \|(\boldsymbol{U}^k\boldsymbol{V})_\Omega - \boldsymbol{X}_\Omega\|_F^2$$

and

$$\boldsymbol{U}^{k+1} = \arg\min_{\boldsymbol{U}} \ \|(\boldsymbol{U}\boldsymbol{V}^{k+1})_\Omega - \boldsymbol{X}_\Omega\|_F^2$$

where the algorithm is initialized with $\boldsymbol{U}^0$, and $\boldsymbol{U}^k$ represents the estimate of $\boldsymbol{U}$ at the kth iteration.

For a fixed $\boldsymbol{U}$, consider solving $\boldsymbol{V}$:

$$\min_{\boldsymbol{V}} \ J(\boldsymbol{V}) := \|(\boldsymbol{U}\boldsymbol{V})_\Omega - \boldsymbol{X}_\Omega\|_F^2$$

Note that $(\cdot)^k$ is dropped for notational simplicity.
Denoting the ith row of $\boldsymbol{U}$ and the jth column of $\boldsymbol{V}$ as $\boldsymbol{u}_i^T$ and $\boldsymbol{v}_j$, where $\boldsymbol{u}_i, \boldsymbol{v}_j \in \mathbb{R}^r$, $i = 1, \cdots, n_1$, $j = 1, \cdots, n_2$, the problem can be rewritten as:

$$\min_{\boldsymbol{V}} \; J(\boldsymbol{V}) := \sum_{(i,j)\in\Omega} \left(\boldsymbol{u}_i^T \boldsymbol{v}_j - \boldsymbol{X}_{ij}\right)^2$$

Since $J(\boldsymbol{V})$ is decoupled w.r.t. $\boldsymbol{v}_j$, it is equivalent to solving the following n_2 independent subproblems:

$$\min_{\boldsymbol{v}_j} \; J(\boldsymbol{v}_j) := \sum_{i\in\mathcal{I}_j} \left(\boldsymbol{u}_i^T \boldsymbol{v}_j - \boldsymbol{X}_{ij}\right)^2, \; j = 1, \cdots, n_2$$

where $\mathcal{I}_j = \{j_1, \cdots, j_{|\mathcal{I}_j|}\} \subseteq \{1, \cdots, n_1\}$ denotes the set containing the row indices for the jth column in Ω. Here, $|\mathcal{I}_j|$ stands for the number of elements in $\mathcal{I}_j$.

For example, consider $\boldsymbol{X}_\Omega \in \mathbb{R}^{4\times 3}$:

$$\boldsymbol{X}_\Omega = \begin{bmatrix} 0 & \times & 0 \\ \times & 0 & \times \\ 0 & \times & \times \\ \times & 0 & \times \end{bmatrix}$$

For $j = 1$, the $(2,1)$ and $(4,1)$ entries are observed, and thus $\mathcal{I}_1 = \{2,4\}$. Similarly, $\mathcal{I}_2 = \{1,3\}$ and $\mathcal{I}_3 = \{2,3,4\}$. Combining the results yields $\sum_{j=1}^{n_2} |\mathcal{I}_j| = |\Omega|$.

Define $\boldsymbol{U}_{\mathcal{I}_j} \in \mathbb{R}^{|\mathcal{I}_j|\times r}$ containing the $|\mathcal{I}_j|$ rows indexed by $\mathcal{I}_j$:

$$\boldsymbol{U}_{\mathcal{I}_j} = \begin{bmatrix} \boldsymbol{u}_{j_1}^T \\ \vdots \\ \boldsymbol{u}_{j_{|\mathcal{I}_j|}}^T \end{bmatrix}$$

and

$$\boldsymbol{b}_{\mathcal{I}_j} = [\boldsymbol{X}_{j_1 j}, \cdots, \boldsymbol{X}_{j_{|\mathcal{I}_j|} j}]^T \in \mathbb{R}^{|\mathcal{I}_j|}$$

then we obtain:

$$\min_{\boldsymbol{v}_j} \; J(\boldsymbol{v}_j) := \|\boldsymbol{U}_{\mathcal{I}_j}\boldsymbol{v}_j - \boldsymbol{b}_{\mathcal{I}_j}\|_2^2 \tag{14}$$

Following (2) to (3), $\boldsymbol{v}_j$ is determined as:

$$\boldsymbol{v}_j = \left(\boldsymbol{U}_{\mathcal{I}_j}^T \boldsymbol{U}_{\mathcal{I}_j}\right)^{-1} \boldsymbol{U}_{\mathcal{I}_j}^T \boldsymbol{b}_{\mathcal{I}_j} \tag{15}$$

Having the same structure in $\boldsymbol{U}^{k+1} = \arg\min_{\boldsymbol{U}} \ \|(\boldsymbol{U}\boldsymbol{V}^{k+1})_\Omega - \boldsymbol{X}_\Omega\|_F^2$, the ith row of $\boldsymbol{U}$ is updated by

$$\min_{\boldsymbol{u}_i^T} \ \|\boldsymbol{u}_i^T \boldsymbol{V}_{\mathcal{J}_i} - \boldsymbol{b}_{\mathcal{J}_i}^T\|_2^2 \qquad \text{or} \qquad \min_{\boldsymbol{u}_i} \ \|\boldsymbol{V}_{\mathcal{J}_i}^T \boldsymbol{u}_i - \boldsymbol{b}_{\mathcal{J}_i}\|_2^2$$

The solution is:

$$\boldsymbol{u}_i = \left(\boldsymbol{V}_{\mathcal{J}_i}\boldsymbol{V}_{\mathcal{J}_i}^T\right)^{-1}\boldsymbol{V}_{\mathcal{J}_i}\boldsymbol{b}_{\mathcal{J}_i}$$

Here $\mathcal{J}_i = \{i_1, \cdots, i_{|\mathcal{J}_i|}\} \subseteq \{1, \cdots, n_2\}$ is the set containing the column indices for the ith row in Ω.

Using previous example, only $(1, 2)$ entry is observed for $i = 1$, and thus $\mathcal{J}_1 = \{2\}$. Similarly, $\mathcal{J}_2 = \{1, 3\}$, $\mathcal{J}_3 = \{2, 3\}$ and $\mathcal{J}_4 = \{1, 3\}$. Here, $\boldsymbol{V}_{\mathcal{J}_i}^{k+1} \in \mathbb{R}^{r\times|\mathcal{J}_i|}$ contains $|\mathcal{J}_i|$ columns indexed by $\mathcal{J}_i$ and $\boldsymbol{b}_{\mathcal{J}_i}^T = [\boldsymbol{X}_{ii_1}, \cdots, \boldsymbol{X}_{ii_{|\mathcal{J}_i|}}]^T \in \mathbb{R}^{|\mathcal{J}_i|}$. We still have $\sum_{i=1}^{n_1} |\mathcal{J}_i| = |\Omega|$.

This low-rank matrix completion algorithm is summarized as:

```
**Input**: $X_\Omega$, $\Omega$ and $r$
    **Initialize:** randomly initialize $\boldsymbol{U}^0 \in \mathbb{R}^{n_1 \times n_2}$
    Determine $\{\mathcal{I}_j\}_{j=1}^{n_2}$ and $\{\mathcal{J}_i\}_{i=1}^{n_1}$
    **for** $k = 0, 1, \cdots$ **do**
        **for** $j = 1, \cdots, n_2$ **do** ($\boldsymbol{U}^k$ is fixed):
            $\boldsymbol{v}_j^{k+1} = \left(\boldsymbol{U}_{\mathcal{I}_j}^{k}{}^T \boldsymbol{U}_{\mathcal{I}_j}^k\right)^{-1} \boldsymbol{U}_{\mathcal{I}_j}^{k}{}^T \boldsymbol{b}_{\mathcal{I}_j}^k$
        **end for**
        **for** $i = 1, \cdots, n_1$ **do** ($\boldsymbol{V}^k$ is fixed):
            $\boldsymbol{u}_i^{k+1} = \left(\boldsymbol{V}_{\mathcal{J}_i}^k \boldsymbol{V}_{\mathcal{J}_i}^{k}{}^T\right)^{-1} \boldsymbol{V}_{\mathcal{J}_i}^k \boldsymbol{b}_{\mathcal{J}_i}$
        **end for**
    **until** convergence
    **end for**
```

Output: $\boldsymbol{M} = \boldsymbol{U}^{k+1}\boldsymbol{V}^{k+1}$

It can be revealed that this is the same as the ALS with $\lambda = 0$.
Using the provided MATLAB code with $r = 2$:

```
X = [5  5  0  0  ;
      5  0  0  0  ;
      0  4  0  0  ;
      0  0  5  4  ;
      0  0  5  0  ];
omega = [1  2  4  5  6  8  9  10  11  13  14  15  16  17  19];
Omega = zeros(5, 4);
Omega(omega) = 1;
X_Omega = X.*Omega;
rank = 2;
maxiter = 100;
[M ,RMSE] = LP2(X,X_Omega,Omega,rank,maxiter);

>> Omega
Omega =
     1      1      1      1
      1     0     0     1
      0     1     1     0
      1     1     1     1
      1     1     1     0
>> M
M =
    5.0000    5.0000         0    0.0000
    5.0000    5.0000   -0.0000         0
    4.0000    4.0000    0.0000    0.0000
    0.0000    0.0000    5.0000    4.0000
    0.0000    0.0000    5.0000    4.0000
```

That is:

$$\boldsymbol{M} = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & 5 & 0 & 0 \\ 4 & 4 & 0 & 0 \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 4 \end{bmatrix} \quad \text{versus} \quad \boldsymbol{X} = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & ? \end{bmatrix}$$

It is also important to note that since we do not directly solve

(13), only a local solution is guaranteed. To ensure a good solution, we can run the algorithm a few times, and choose the solution with minimum $\|\boldsymbol{M}_\Omega - \boldsymbol{X}_\Omega\|_F^2$.

Although collaborative filtering is domain-free and requires no user and item profiles, it suffers from cold start problem, i.e., inability to handle new user/item.

Suppose adding a new user in our example:

	Alice	Bob	Carol	Dave	Eve
Beauty and the Beast	5	5	0	0	?
Titanic	5	?	?	0	?
P.S. I Love You	?	4	0	?	?
Mission:	0	0	5	4	?

Impossible					
John Wick	0	0	5	?	?

Can you estimate the ratings for Eve?

That is, we need to compute $\boldsymbol{\theta}^{(5)}$ as well.

According to (10), to find $\boldsymbol{\theta}^{(5)}$, we should only minimize:

$$\lambda \sum_{j=1}^{J} \sum_{l=1}^{L} \left(\theta_l^{(j)}\right)^2 \quad \text{or} \quad \lambda \sum_{l=1}^{L} \left(\theta_l^{(5)}\right)^2$$

because there is no $y^{(i,j)}$ for Eve. This minimization will simply, for our case, gives:

$$\theta_1^{(5)} = \theta_2^{(5)} = 0$$

and hence

$$\boldsymbol{Y} = \begin{bmatrix} 5 & 5 & 0 & 0 & 0 \\ 5 & ? & ? & 0 & 0 \\ ? & 4 & 0 & ? & 0 \\ 0 & 0 & 5 & 4 & 0 \\ 0 & 0 & 5 & ? & 0 \end{bmatrix}$$

We do not recommend Eve any movies and this may not be reasonable.

To get a more reasonable result, we may follow the idea in the PCA, i.e., we operate on the matrix with zero-mean rows:

$$\overline{\boldsymbol{Y}} = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.67 & -1.67 & 3.34 & ? & ? \end{bmatrix} \text{ with } \quad \boldsymbol{\mu} = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.67 \end{bmatrix}$$

That is, after completing $\overline{Y}$, we add $\boldsymbol{\mu}$ in the final result.

Although we still get $\theta_1^{(5)} = \theta_2^{(5)} = 0$, the column vector for Eve will become $\boldsymbol{\mu}$.

The best we may do is to use average rating for unrated user. To assess the performance of a recommender system, we may use MSE:

$$\mathrm{MSE}(\hat{y}^{(i,j)}) = \mathbb{E}\left\{\left(\hat{y}^{(i,j)} - y_{\mathrm{unobserved}}^{(i,j)}\right)^2\right\}$$

Here, $y_{\mathrm{unobserved}}^{(i,j)}$ represents the ground truth value of the unobserved rating. In practice, we can use the empirical MSE:

$$\mathrm{MSE}(\hat{y}^{(i,j)}) = \frac{1}{N_{\text{unobserved}}} \sum_{\text{unobserved } (i,j)} \left(\hat{y}^{(i,j)} - y^{(i,j)}_{\text{unobserved}} \right)^2 \qquad (16)$$

where $N_{\text{unobserved}}$ is the number of missing entries in the recommender system matrix. Famous datasets such as MovieLens and Netflix are commonly used for evaluation. MovieLens 100 K dataset can be downloaded at:

https://grouplens.org/datasets/movielens/

- 100,000 ratings from 943 users on 1,682 movies, i.e., $\boldsymbol{X}_\Omega \in \mathbb{R}^{1682 \times 943}$ with around 94% missing entries
- Rating is from 1 to 5
- Each user has rated at least 20 movies

To compute (16), we can use the cross-validation idea. For example, we can separate Ω into 2 sets Ω_1 and Ω_2 of 50,000 ratings such that $\Omega = \Omega_1 \cup \Omega_2$ and $\Omega_1 \cap \Omega_2 = \varnothing$.

Suppose the estimate based on $\boldsymbol{X}_{\Omega_1}$ is $\boldsymbol{M}$. We can use the difference between $\boldsymbol{M}_{\Omega_2}$ and $\boldsymbol{X}_{\Omega_2}$ to compute the empirical MSE.

Suppose:

$$\boldsymbol{X} = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & ? \end{bmatrix} \Rightarrow \boldsymbol{X}_{\boldsymbol{\Omega}_1} = \begin{bmatrix} 5 & ? & 0 & ? \\ ? & ? & ? & 0 \\ ? & 4 & ? & ? \\ 0 & ? & 5 & ? \\ ? & 0 & 5 & ? \end{bmatrix} \text{ and } \boldsymbol{X}_{\boldsymbol{\Omega}_2} = \begin{bmatrix} ? & 5 & ? & 0 \\ 5 & ? & ? & ? \\ ? & ? & 0 & ? \\ ? & 0 & ? & 4 \\ 0 & ? & ? & ? \end{bmatrix}$$

$$\boldsymbol{X}_{\Omega_1} = \begin{bmatrix} 5 & ? & 0 & ? \\ ? & ? & ? & 0 \\ ? & 4 & ? & ? \\ 0 & ? & 5 & ? \\ ? & 0 & 5 & ? \end{bmatrix} \Rightarrow \boldsymbol{M} = \begin{bmatrix} 5 & 4 & 0 & 1 \\ 5 & 1 & 1 & 0 \\ 5 & 4 & 0 & 0 \\ 0 & 1 & 5 & 5 \\ 0 & 0 & 5 & 5 \end{bmatrix}$$

The MSE based on $\boldsymbol{M}_{\Omega_2}$ and $\boldsymbol{X}_{\Omega_2}$ is then:

$$\frac{(4-5)^2+(1-0)^2+(5-5)^2+(0-0)^2+(1-0)^2+(5-4)^2+(0-0)^2}{7}$$

Nearest Neighborhood

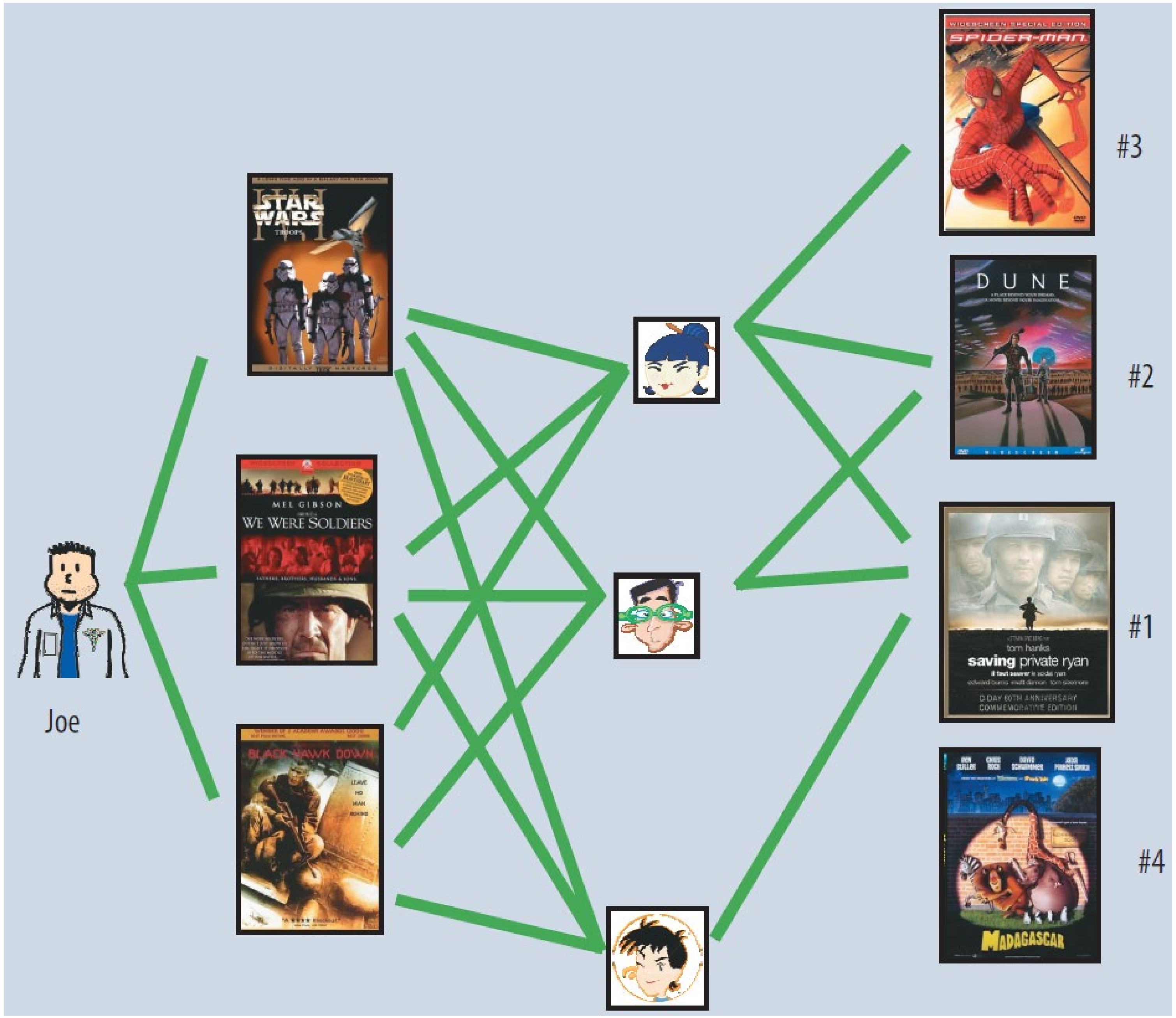
It can also be classified as the earliest collaborative filtering approach, which expects to be inferior to matrix factorization.

Neighborhood methods are centered on computing the relationships between users or items, i.e., either useroriented or item-oriented.

Because similar users have similar patterns of rating behaviors, and similar items receive similar ratings.

User-based method: users liked same types of products then they are most probably similar and they might like same type of products.

For a particular user, we want to find his nearest neighbors and recommend him products liked by them.


Joe
#3
#2
#1
#4

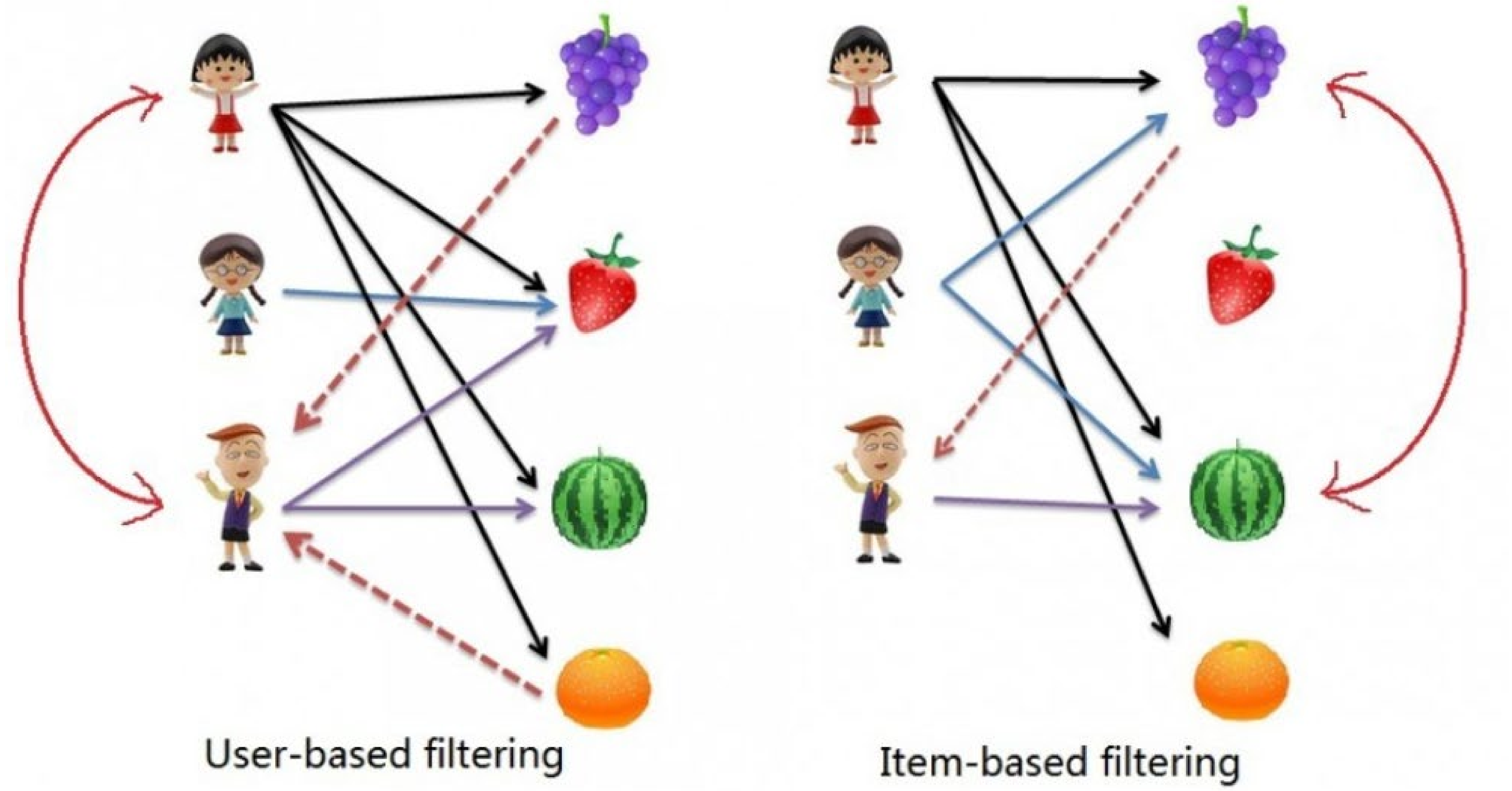
What will you recommend for Joe? First priority? Second priority?

That is, the ratings provided by similar users to Joe are used to make recommendation for Joe.

The predicted ratings of Joe can be computed as weighted average values of the peer-group ratings for each item.

Item-based method: Similar items are rated in a similar way by the same user and we predict a user's preference for an item based on ratings of "neighboring" items by the same user.

Consider item B, the first step is to determine a set of S items most similar to B. The weighted average of the ratings in S, specified by user A, can be used to predict rating of A for B.



Source: https://medium.com/@cfpinela/recommender-systems-user-based-and-item-based-collaborative-filtering-5d5f375a127f

Suppose users *A* and *B* rate N products and the ratings are expressed as length-N vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. To measure the similarity between *A* and *B*, one possibility is to use the Euclidean distance:

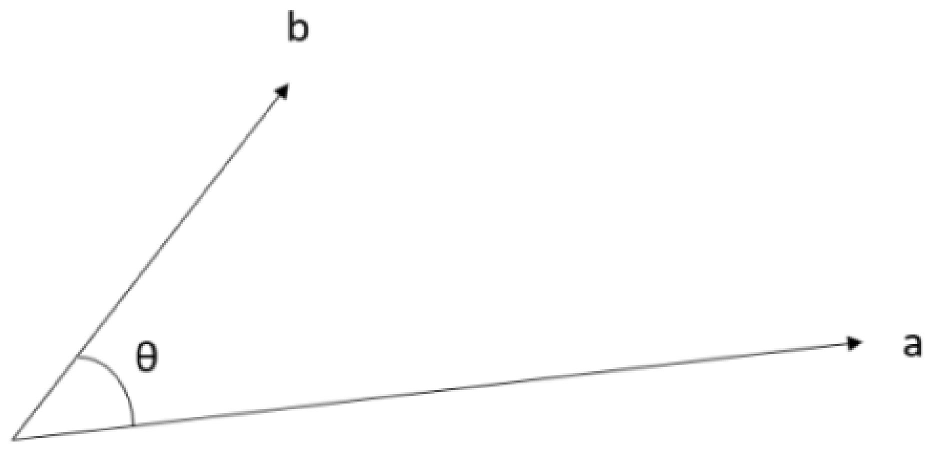
$$||\boldsymbol{a}-\boldsymbol{b}||_2 = \sqrt{(a_1-b_1)^2+\cdots+(a_N-b_N)^2}$$

The most similar case occurs when $||\boldsymbol{a}-\boldsymbol{b}||_2=0$, meaning that $\boldsymbol{a}=\boldsymbol{b}$ but we may not know its maximum value.

It is more common to use cosine similarity and Pearson correlation coefficient.

Cosine similarity between $\boldsymbol{a}$ and $\boldsymbol{b}$ is:

$$\mathrm{Cosine}(\boldsymbol{a},\boldsymbol{b}) = \frac{\boldsymbol{a}^T\boldsymbol{b}}{||\boldsymbol{a}||_2\cdot||\boldsymbol{b}||_2} = \frac{\sum_{n=1}^N a_n b_n}{\sqrt{\sum_{n=1}^N a_n^2}\sqrt{\sum_{n=1}^N b_n^2}} \tag{17}$$



The name is due to the geometric relationship:

$$\boldsymbol{a} \cdot \boldsymbol{b} = ||\boldsymbol{a}||_2 \cdot ||\boldsymbol{b}||_2 \cos(\theta)$$

We see that $\boldsymbol{a}$ and $\boldsymbol{b}$ are most similar when $\cos(\theta) = 1$, i.e., they are in the same direction while they are most dissimilar when $\cos(\theta) = -1$.

Hence the cosine similarity is bounded by:

$$-1 \leq \text{cosine}(\boldsymbol{a}, \boldsymbol{b}) \leq 1$$

Pearson correlation coefficient for $\boldsymbol{a}$ and $\boldsymbol{b}$ is:

$$\mathrm{Pearson}(\boldsymbol{a},\boldsymbol{b}) = \frac{\sum_{n=1}^{N}(a_n-\bar{a})(b_n-\bar{b})}{\sqrt{\sum_{n=1}^{N}(a_n-\bar{a})^2}\sqrt{\sum_{n=1}^{N}(b_n-\bar{b})^2}} \qquad (18)$$

where

$$\bar{a} = \frac{1}{N}\sum_{n=1}^{N} a_n \qquad \text{and} \qquad \bar{b} = \frac{1}{N}\sum_{n=1}^{N} b_n$$

That is, (18) is the cosine similarity between the zero-mean versions of $\boldsymbol{a}$ and $\boldsymbol{b}$. Hence it also has values between 1 and -1.

As some users normally give higher ratings on the products but some people rate the products lower, the Pearson correlation coefficient can take into account for the biases of users.

In the practical scenarios, it is not possible for users *A* and *B* to rate all N products. Hence (17) and (18) should be modified accordingly to include the product ratings both users have rated.

Suppose $N = 5$, and user *A* has rated products 1, 2, 3 and 4, while user *B* has rated products 3, 4 and 5. Then (18) is calculated as:

$$\text{Pearson}(\boldsymbol{a}, \boldsymbol{b}) = \frac{(a_3 - \bar{a})(b_3 - \bar{b}) + (a_4 - \bar{a})(b_4 - \bar{b})}{\sqrt{\sum_{n=3}^{4}(a_n - \bar{a})^2}\sqrt{\sum_{n=3}^{4}(b_n - \bar{b})^2}}$$

where

$$\bar{a} = \frac{1}{4}\sum_{n=1}^{4} a_n \qquad \text{and} \qquad \bar{b} = \frac{1}{3}\sum_{n=3}^{5} b_n$$

To present in a general form, we let $\mathcal{I}_A$ and $\mathcal{I}_B$ be the sets of item indices for which ratings are provided by users *A* and *B*, respectively.

The mean ratings are:

$$\bar{a} = \frac{1}{|\mathcal{I}_A|} \sum_{n \in \mathcal{I}_A} a_n \qquad \text{and} \qquad \bar{b} = \frac{1}{|\mathcal{I}_B|} \sum_{n \in \mathcal{I}_B} b_n$$

and (18) becomes:

$$\mathrm{Pearson}(\boldsymbol{a}, \boldsymbol{b}) = \frac{\sum_{n \in \mathcal{I}_A \cap \mathcal{I}_B} (a_n - \bar{a})(b_n - \bar{b})}{\sqrt{\sum_{n \in \mathcal{I}_A \cap \mathcal{I}_B} (a_n - \bar{a})^2} \sqrt{\sum_{n \in \mathcal{I}_A \cap \mathcal{I}_B} (b_n - \bar{b})^2}} \tag{19}$$

After obtaining the similarity measure, we can perform product recommendation based on weighted average.

Suppose user *A* is closest to his peer-group members *B*$_1$, …, *B*$_K$, and they have provided ratings for item j, as $r_{1,j},\cdots,r_{K,j}$.

The rating of user *A* for item j can be predicted as:

$$\hat{r}_{a,j}=\frac{\sum_{k=1}^{K}\mathrm{Sim}(\boldsymbol{a},\boldsymbol{b}_k)\cdot r_{k,j}}{\sum_{k=1}^{K}|\mathrm{Sim}(\boldsymbol{a},\boldsymbol{b}_k)|} \tag{20}$$

where Sim means cosine similarity or correlation coefficient. To account for individual variations, we can use the mean-centered version when correlation coefficient is applied:

$$\hat{r}_{a,j}=\bar{a}+\frac{\sum_{k=1}^{K}\mathrm{Sim}(\boldsymbol{a},\boldsymbol{b}_k)\cdot(r_{k,j}-\bar{b}_k)}{\sum_{k=1}^{K}|\mathrm{Sim}(\boldsymbol{a},\boldsymbol{b}_k)|} \tag{21}$$

Example 3

Consider the following user-item matrix where the rating is based on a 7-point scale. According to the user-based method, perform product recommendation for user 3 using Pearson correlation coefficient.

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	7	6	7	4	5	4
User 2	6	7	?	4	3	4
User 3	?	3	3	1	1	?
User 4	1	2	2	3	3	4
User 5	1	?	1	2	3	3

That is, we need to predict user 3 ratings on items 1 and 6.

Item-Id ⇒ / User-Id ⇓	1	2	3	4	5	6	Mean Rating	Cosine$(i,3)$ (user-user)	Pearson$(i,3)$ (user-user)
1	7	6	7	4	5	4	5.5	0.956	0.894
2	6	7	?	4	3	4	4.8	0.981	0.939
3	?	3	3	1	1	?	2	1.0	1.0
4	1	2	2	3	3	4	2.5	0.789	-1.0
5	1	?	1	2	3	3	2	0.645	-0.817

First, we need to compute the similarity between user 3 and all other users. For example, the values of $\text{Cosine}(1,3)$ and $\text{Pearson}(1,3)$ are computed as:

$$\text{Cosine}(1,3) = \frac{6*3+7*3+4*1+5*1}{\sqrt{6^2+7^2+4^2+5^2}\cdot\sqrt{3^2+3^2+1^2+1^2}} = 0.956$$

$$\text{Pearson}(1,3) =$$

$$= \frac{(6-5.5)*(3-2)+(7-5.5)*(3-2)+(4-5.5)*(1-2)+(5-5.5)*(1-2)}{\sqrt{0.5^2+1.5^2+(-1.5)^2+(-0.5)^2}\cdot\sqrt{1^2+1^2+(-1)^2+(-1)^2}}$$

$$= 0.894$$

As all ratings are positive, the cosine similarity values are all positive, while Pearson correlation coefficient is more discriminative because its sign provides similarity and dissimilarity information.

If we choose top-2 closest users to user 3, both measures indicate users 1 and 2. Applying (20) with the Pearson correlation coefficient, the predicted values are:

$$\hat{r}_{3,1} = \frac{7 \times 0.894 + 6 \times 0.939}{0.894 + 0.939} = 6.49 \approx 6$$

and

$$\hat{r}_{3,6} = \frac{4 \times 0.894 + 4 \times 0.939}{0.894 + 0.939} = 4$$

Hence item 1 should be prioritized over item 6 in recommending user 3.

However, for item 6, the predicted rating of 4 is higher than all observed ratings, because of the bias caused by the peer users 1 and 2 – more optimistic with more positive ratings.

Based on (21), the results are:

$$\hat{r}_{3,1} = 2 + \frac{(7-5.5)\times 0.894 + (6-4.8)\times 0.939}{0.894+0.939} = 3.35 \approx 3$$

and

$$\hat{r}_{3,6} = 2 + \frac{(4-5.5)\times 0.894 + (4-4.8)\times 0.939}{0.894+0.939} = 0.86 \approx 1$$

The priority of recommending item 1 is still higher than that of item 6.

But mean-centering process enables a better relative prediction because 3.35 and 0.86 are comparable with the highest and lowest observed ratings.

In item-based approach, peer groups are constructed in terms of items rather than users because we want to compute the similarities between items.

Let $\mathcal{U}_i$ and $\mathcal{U}_j$ be the indices of the set of users who have rated items i and j.

We use the adjusted cosine similarity to measure the similarity between items i and j:

$$\mathrm{AdjCosine}(i,j) = \frac{\sum_{u\in\mathcal{U}_i\cap\mathcal{U}_j} s_{u,i}\cdot s_{u,j}}{\sqrt{\sum_{u\in\mathcal{U}_i\cap\mathcal{U}_j} s_{u,i}^2}\cdot\sqrt{\sum_{u\in\mathcal{U}_i\cap\mathcal{U}_j} s_{u,j}^2}} \tag{22}$$

where $s_{u,i}$ is the mean-centered rating for user u on item i as in (19). Note that (22) is not Pearson correlation coefficient because mean centering is still performed on rows.

Suppose there are K closest items, say, 1 to K, to target item j, for which user *A* has provided ratings.

The predicted rating $\hat{r}_{a,j}$ of user *A* on item j can be computed using the weighted average:

$$\hat{r}_{a,j} = \frac{\sum_{k=1}^{K} \mathrm{AdjCosine}(k,j) \cdot r_{a,k}}{\sum_{k=1}^{K} |\mathrm{AdjCosine}(k,j)|} \tag{23}$$

For example, in a movie recommendation system, the item peer group will typically be movies of a similar genre. The rating history of the same user on such movies is a very reliable predictor of the interest of that user.

Example 4

Repeat Example 3 using the item-based method with the adjusted cosine similarity.

Item-Id ⇒ / User-Id ⇓	1	2	3	4	5	6
1	1.5	0.5	1.5	-1.5	-0.5	-1.5
2	1.2	2.2	?	-0.8	-1.8	-0.8
3	?	1	1	-1	-1	?
4	-1.5	-0.5	-0.5	0.5	0.5	1.5
5	-1	?	-1	0	1	1
Cosine$(1, j)$ (item-item)	1	0.735	0.912	-0.848	-0.813	-0.990
Cosine$(6, j)$ (item-item)	-0.990	-0.622	-0.912	0.829	0.730	1

As in the user-based method, the average rating of each item in the rating matrix is first subtracted from each rating to create a mean-centered matrix.

The missing ratings of user 3 correspond to items 1 and 6. Hence the similarity of the columns for items 1 and 6 needs to be computed.

For example, the adjusted cosine similarity between items 1 and 3 is computed as:

$$\begin{aligned}\mathrm{AdjCosine}(1,3) &= \frac{1.5\times 1.5+(-1.5)\times(-0.5)+(-1)\times(-1)}{\sqrt{1.5^2+(-1.5)^2+(-1)^2}\cdot\sqrt{1.5^2+(-0.5)^2+(-1)^2}}\\ &= 0.912\end{aligned}$$

From the adjusted cosine similarities, items 2 and 3 are most similar to item 1, while items 4 and 5 are closest to item 6.

Hence we have:

$$\hat{r}_{3,1} = \frac{3 \times 0.735 + 3 \times 0.912}{0.735 + 0.912} = 3$$

and

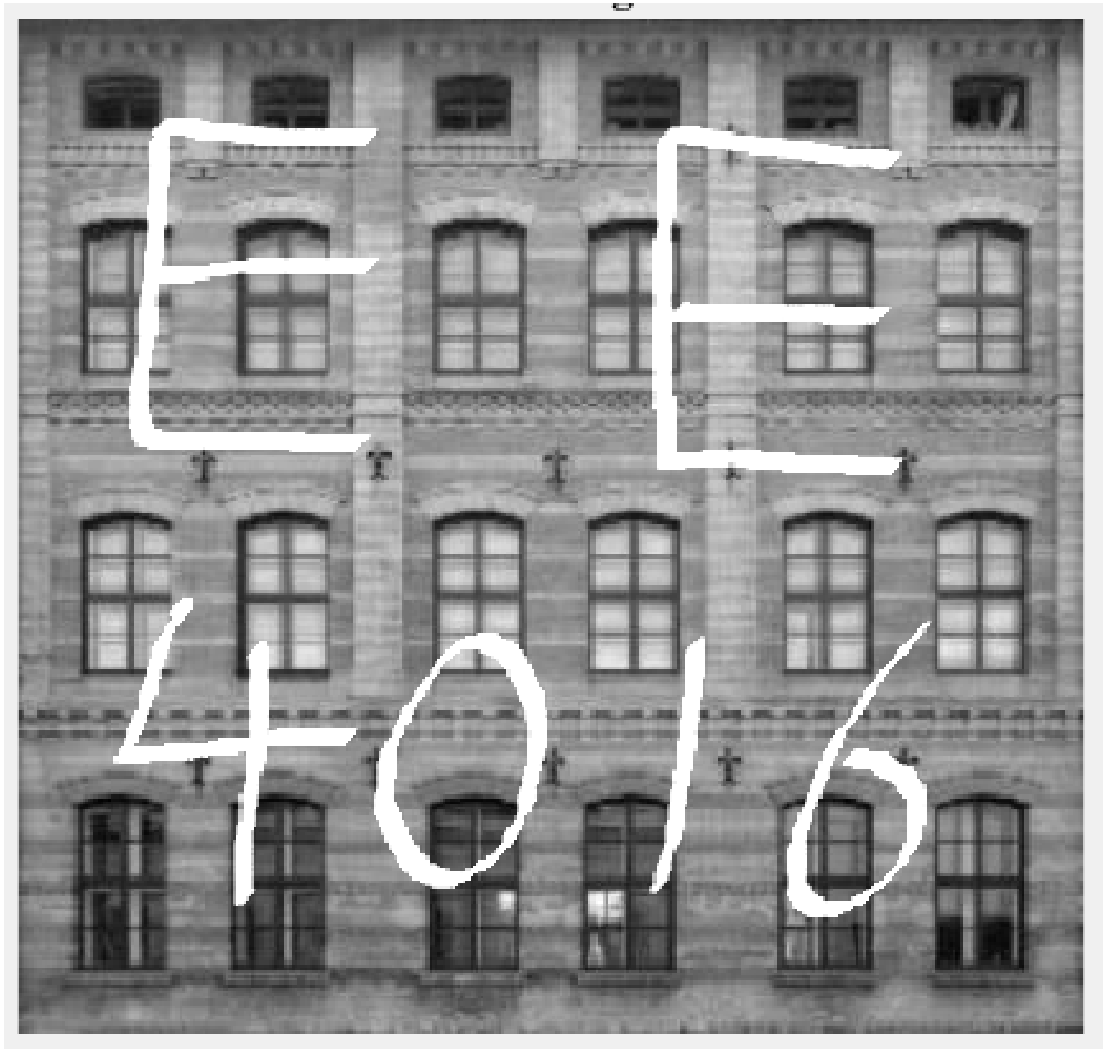
$$\hat{r}_{3,6} = \frac{1 \times 0.829 + 1 \times 0.730}{0.829 + 0.730} = 1$$

Since the predicted ratings use the ratings of user 3 himself, the results tend to be more consistent with the other ratings of this user.

Note that although Examples 3 and 4 give the same results after rounding, it is possible that user-based and item-based methods provide different recommendations.

Application to Image Inpainting

Consider an image with missing/distorted pixels. Is it possible to recover it?



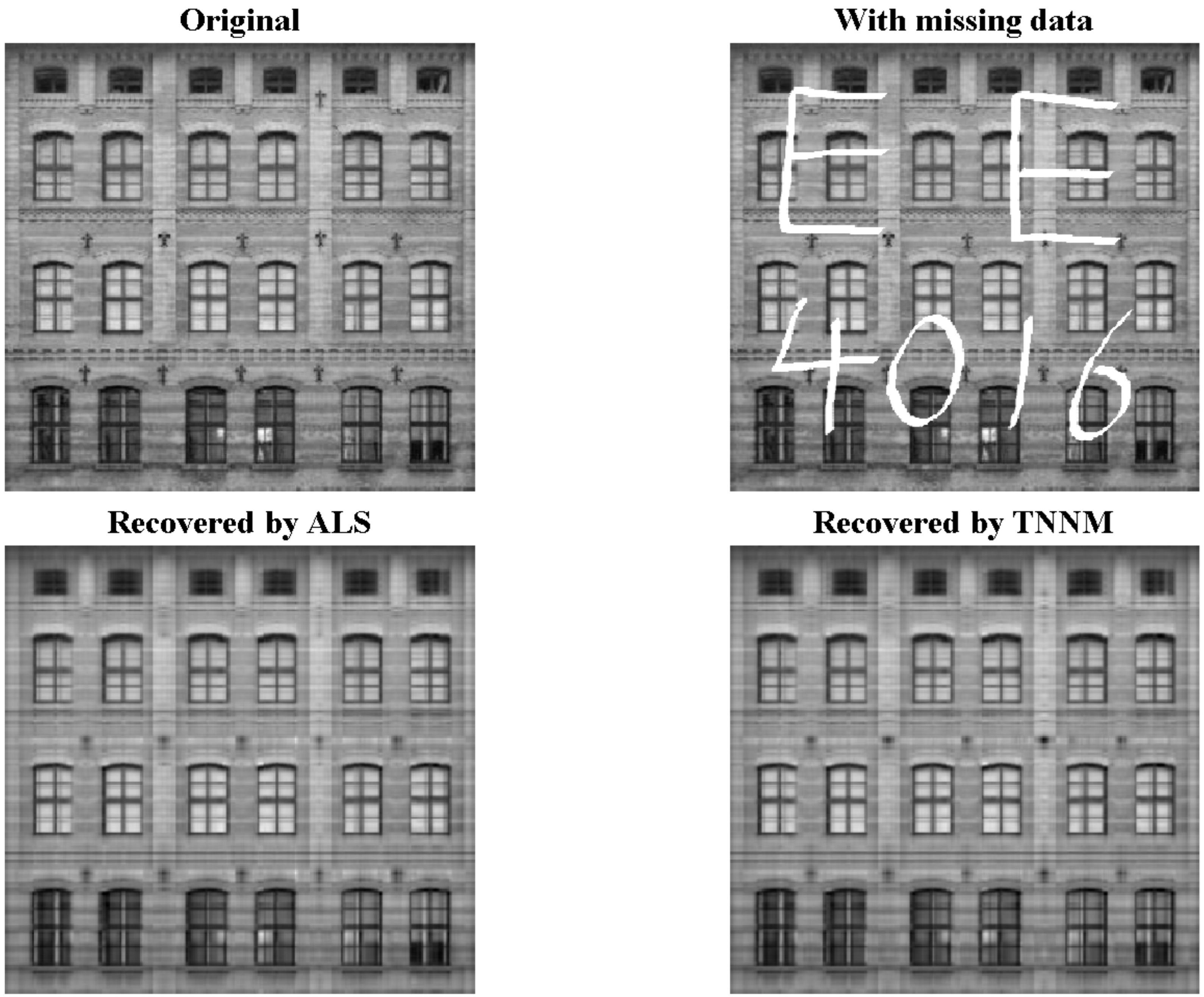
The task of recovering such an image is called image inpainting.

If the image data can be approximated as a low-rank matrix, matrix completion methods can be applied for its restoration.

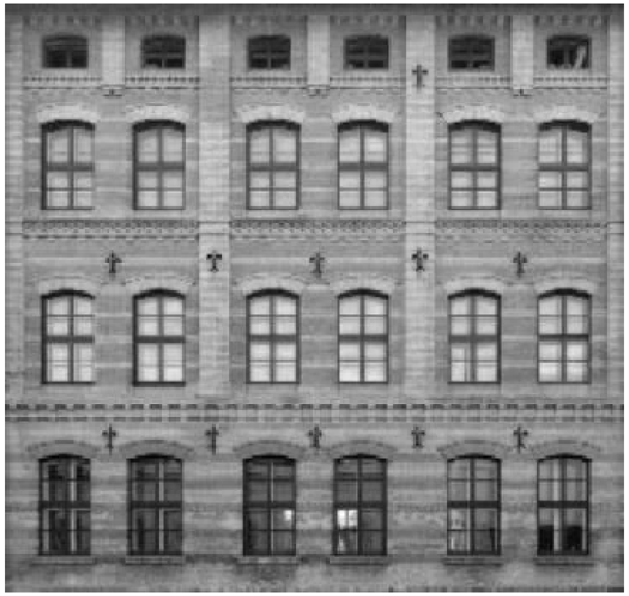
In this image, the distorted pixels correspond to “EE4016”.

We apply ALS with $\lambda = 0$ and nuclear norm minimization on the image. A noisy version with zero-mean Gaussian noise is also investigated.

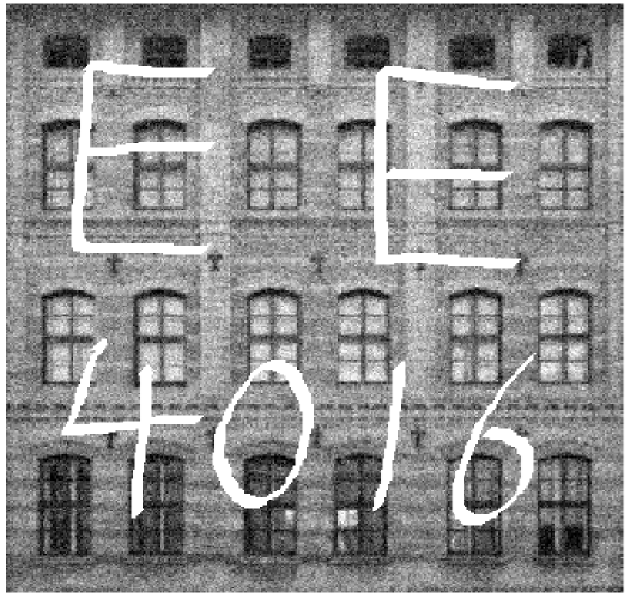
In ALS, we use rank $r = 7$ while for nuclear norm minimization, we start minimization from the 8th largest singular values because the first 7 singular values contain the image features, which is referred to as truncated nuclear norm minimization.


Original
With missing data
EE
4016
Recovered by ALS
Recovered by TNNM

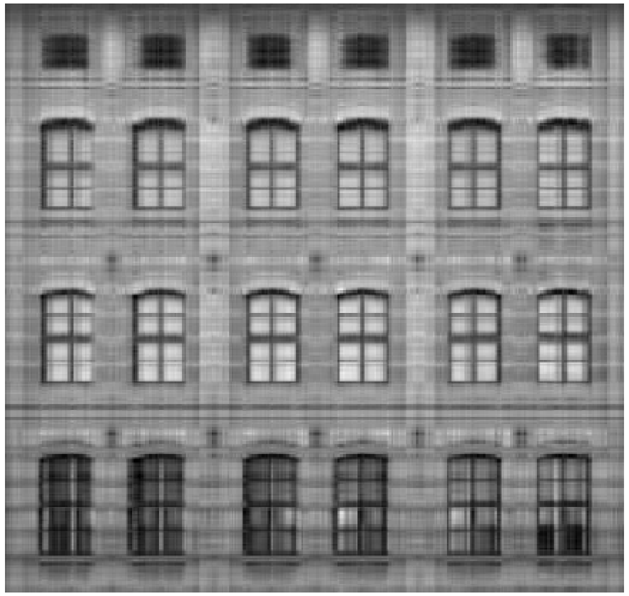
Original



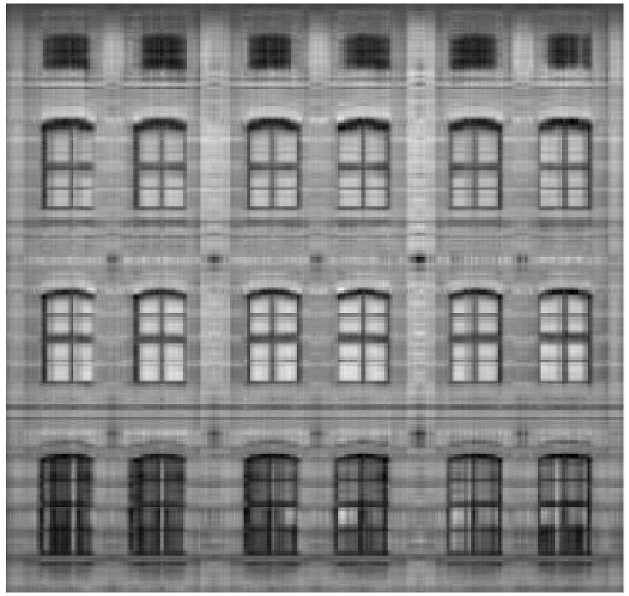
With missing data at SNR = 12dB



Recovered by ALS



Recovered by TNNM



References:

1. S. Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*, Academic Press, 2015
2. C. C. Aggarwal, *Recommender Systems: The Textbook*, Springer, 2016
3. YouTube videos by Andrew Ng, including https://www.youtube.com/watch?v=giIXNoiqO_U
4. Y. Koren, R. Bell and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30-37, 2009
5. http://stanford.edu/~rezab/classes/cme323/S15/notes/lec14.pdf
6. W.-J. Zeng and H. C. So, "Outlier-robust matrix completion via ℓ_p-minimization," *IEEE Transactions on Signal Processing*, vol. 66, pp. 1125-1140, 2018
7. Y. Hu, D. Zhang, J. Ye, X. Li and X. He, "Fast and accurate matrix completion via truncated nuclear norm regularization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 9, pp. 2117-2130, 2013