MatrixFactorization

October 6, 2018

0.1 1. Gii thiu v phng pháp Recommendation system

0.1.1 1. 1. Tm quan trng ca recommendation system

Trong cuc sng hàng ngày chúng ta thng thy nhng tình hung khá tình c khi các h thng ln có kh nng c và hiu s thích ca ngi dùng và hin th nhng thông tin mà ngi dùng quan tâm rt chun xác. Chng hn nh:

- Facebook có kh nng hin th trên newfeed nhng trng thái ca nhng ngi mà bn quan tâm.
- Youtube có th t ng nhy sang nhng video mà bn có kh nng yêu thích da trên nhng gì mà bn ang xem.
- Amazon có th a ra nhng cun sách cùng loi vi nhng cun sách mà bn ã mua hoc rating cao.
- Google có th a qung cáo v mt vt phù hp vi nhng gì bn tìm kim gn ây.

Nu không có các thut toán recommendation, tri nghim ngi dùng s kém hn vì thông tin mà h thc s quan tâm không c a ra úng thi im trong khi thông tin không cn thit c a ra nhiu hn. Hu qu là ngi dùng cm thy b làm phin và nhiu lon thông tin. Trong lnh vc marketing h thng recommendation li càng tr nên quan trng. Mi sn phm c a n úng ngi tiêu dùng có nhu cu s làm tng doanh thu, gim chi phí thi gian, chi phí qung cáo và giúp ngi tiêu dùng s hu c th mình cn. Trong lnh vc gii trí nh video, game, truyn online,... ngi dùng s t hài lòng cao khi tìm c úng loi hình gii trí yêu thích d dàng. Dù ch ra i trong 10 nm tr li ây, song hành cùng thi kì bùng n internet nhng có th nói `recommendation system` là mt lnh vc nghiên cu sôi ng. Nó ã to ra mt cuc cách mng thay i hành vi mua sm, hành vi gii trí, chin lc kinh doanh,... trên toàn cu. Hàng trm triu các h kinh doanh nh l ang hng li t nó thông qua khai thác ngun khách hàng vô tn t tài nguyên mng và bin kênh bán hàng này thay th các kênh truyn thng. Lnh vc này ng thi là chìa khóa mu cht giúp các công ty công ngh Google, Facebook, Amazon, Microsoft,... tr thành nhng tp oành hàng u th gii. Chính vì th `recommendation system` luôn c các công ty kinh doanh trên nn tng online u t nghiên cu và phát trin to ra mt h thng thông minh nhm nâng cao tri nghim khách hàng và ti u hóa ngun tài nguyên.

0.1.2 1.2. Phng pháp recommendation

Bên trên chúng ta ã bit vai trò ca `recommendation system` i vi vic phát trin ca lnh vc internet và kinh doanh online. Tuy nhiên thc s bài toán `recommendation system` là gì? C s ca phng pháp `recommendation system` ra sao chúng ta vn cha thc s hiu rõ. Theo nh ngha t wikipedia thì `recommendation system` là mt nhánh nh ca lnh vc *h thng chit lc thông tin* (information filtering system) c s dng d báo mc yêu thích thông qua rating ca mt ngi dùng (user) cho mt sn phm (item). a ra c sn phm phù hp nht n ngi dùng òi hi các h thng phi da trên thông tin ã rating ca sn phm, thông tin ngi dùng, thông tin v sn phm xây dng thut toán ti u. Da trên hàm loss function tính ra sai s d báo ca thut toán và tìm ra mt phng pháp có mc d báo chun xác nht. Có rt nhiu các thut toán

khác nhau c s dng trong `recommendation system` nhng v c bn chúng bao gm 2 phng pháp chính: `Collaborative filtering` và `Content based filtering`. im khác bit c bn gia 2 phng pháp này là:

- Collaborative filtering: Da trên mi quan h tng quan v mt hành vi tiêu dùng hoc c trng sn phm tìm ra các users hoc items có chung c tính, s thích. T ó da trên nhng thông tin mà nhóm ngi dùng hoc sn phm liên quan gn nht ã rating ánh giá sn phm mà mt ngi dùng c th cha rating. Tuy nhiên nhc im ca thut toán này là a ra d báo v rating mà không hoàn toàn hiu v user, item mà hoàn toàn da trên quan sát v mc tng ng gia các nhóm users, items d báo. Thut toán c s dng trong các bài toán này ch yu là k-nearest neighbor tìm ra nhóm tng ng và ma trn h s tng quan c s dng o lng mc gn gi v mt hành vi hay c tính phân nhóm.
- Content based filtering: Da trên nhng thông tin và ni dung liên quan n sn phm nh nhà sn xut, th loi, nm sn xut, công dng, c tính,... hoc da trên thông tin ca ngi dùng nh gii tính, tui, ngành ngh,... a ra d báo v rating ca ngi i vi sn phm ó. Thut toán này ch n thun là các phng trình hi qui gia các chiu c tính ca sn phm hoc ngi dùng i vi im rating mà không tn dng c tng quan v mt hành vi gia nhng nhóm ngi dùng hay c trng sn phm nh `Collaborative filtering`. Trong thc t hành vi ca ngi dùng li cho thy rt ging nhau nu thuc cùng mt nhóm chng hn nh các nhóm nhc thin, nhc vàng, nhc tr, nhc thiu nhi s phù hp vi ngi già, ngi trung niên, ngi tr, thiu nhi. Không xem xét c các yu t tng quan theo nhóm là mt hn ch ln ca content based filtering.

Mi thut toán u có u, nhc im khác nhau và mc hiu qu trong d báo mc yêu thích các cp (user, item) (*ngi dùng, sn phm*) cng khác nhau tùy thuc vào tp d liu. Nhng các thut toán u có im chung ó là s dng d liu mà ngi dùng ã rating i vi các sn phm làm c s d báo rating cho các sn phm cha c ánh giá. Vic này cng ging nh chúng ta chi trò chi in s vào *ma trn tin ích* (utility matrix). Mt chiu ca ma trn ng vi users và chiu còn li ng vi items. Các ô trên ma trn th hin giá tr rating ca user tng ng lên item. Nh vy s có nhng ô ã c rating bi ngi dùng và các ô còn li cha c rating. Quá trình gii bài toán cng ging nh vic chúng ta i gii ma trn ti nhng ô còn thiu sao cho sai s cui cùng gia d báo và thc t là nh nht.

0.1.3 1.3. Gii thiu thut toán matrix factorization

Trong thut toán matrix factorization chúng ta gi nh c trng ca item c th hin qua ma trn $\mathbf{I}$ và hành vi ca ngi dùng c th hin qua ma trn $\mathbf{U}$. Vi mi dòng ca ma trn $\mathbf{I}$ là mt c trng n (*latent feature*) ca sn phm và mi ct ca $\mathbf{U}$ là mc yêu thích ca mt ngi dùng i vi c trng n tng ng. Các c trng n này có th coi nh nhng nhân t chính c tng hp t nhiu thông tin liên quan n sn phm tng t nh thành phn chính trong phép phân tích thành phn chính `PCA`. c trng ca sn phm th m c th hin qua vector dòng $\mathbf{i_m}$ và hành vi ca ngi dùng th n c th hin qua vector ct $\mathbf{u_n}$. Khi ó giá tr d báo mc yêu thích ca mt ngi dùng n lên mt sn phm m s là tích ca 2 vector $\mathbf{i_m}$ và $\mathbf{u_n}$:

$$y_m n = \mathbf{i_m u_n}$$

c lng ca ma trn tin ích $\hat{\mathbf{Y}}$ s c biu din theo các ma trn hành vi $\mathbf{I}$ và ma trn ngi dùng $\mathbf{U}$ nh sau:

$$\hat{\mathbf{Y}} \approx \begin{bmatrix} \mathbf{i}_1\mathbf{u}_1 & \mathbf{i}_1\mathbf{u}_2 & \dots & \mathbf{i}_1\mathbf{u}_N \\ \mathbf{i}_2\mathbf{u}_1 & \mathbf{i}_2\mathbf{u}_2 & \dots & \mathbf{i}_2\mathbf{u}_N \\ \dots & \dots & \ddots & \dots \\ \mathbf{i}_M\mathbf{u}_1 & \mathbf{i}_M\mathbf{u}_2 & \dots & \mathbf{i}_M\mathbf{u}_N \end{bmatrix} = \begin{bmatrix} \mathbf{i}_1 \\ \mathbf{i}_2 \\ \dots \\ \mathbf{i}_M \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_N \end{bmatrix} = \mathbf{IU}$$

0.1.4 1.4. Thut toán gradient descent

Gi s rng chúng ta ã có thông tin v các ma trn **U** và **I** iu chúng ta cn thc hin bây gi là coi các dòng ca mi **I** là mt item profile và mi ct ca **U** là mt user profile. Gi s $\mathbf{I} \in \mathbb{R}^{M \times K}$, $\mathbf{U} \in \mathbb{R}^{K \times N}$, $\mathbf{Y} \in \mathbb{R}^{M \times N}$. Thông thng ta s chn s c trng n nh hn s lng sn phm và ngi dùng. Khi ó Y s c biu din di dng tích ca 2 ma trn có rank nh hn (*Low-rank Matrix factorization*):

$$\hat{\mathbf{Y}} = \mathbf{IU}$$

Hàm loss function ca thut toán chính là chun Frobenius norm v lch gia **Y** và $\hat{\mathbf{Y}}$ nh sau:

$$\mathcal{L}(\mathbf{I}, \mathbf{U}) = \frac{1}{2s}||\mathbf{Y} - \hat{\mathbf{Y}}||_F^2$$

tránh hin tng overfiting h s hiu chnh bc 2 (*l2 - reguralization*) c a thêm vào:

$$\mathcal{L}(\mathbf{I}, \mathbf{U}) = \frac{1}{2s}||\mathbf{Y} - \hat{\mathbf{Y}}||_F^2 + \frac{\lambda_1}{2}||\mathbf{I}||_F^2 + \frac{\lambda_2}{2}||\mathbf{U}||_F^2 \tag{1.4.1}$$

Nu coi **U** c nh và cn ti u **I**. Bài toán `Matrix factorization` s tng ng vi ti u hàm loss function:

$$\mathcal{L}(\mathbf{I}) = \frac{1}{2s}||\mathbf{Y} - \hat{\mathbf{Y}}||_F^2 + \frac{\lambda_1}{2}||\mathbf{I}||_F^2$$

Nu coi **I** c nh và cn ti u **U**. Hàm loss function s có dng:

$$\mathcal{L}(\mathbf{U}) = \frac{1}{2s}||\mathbf{Y} - \hat{\mathbf{Y}}||_F^2 + \frac{\lambda_2}{2}||\mathbf{U}||_F^2$$

Ta nhn thy hàm loss function u là nhng hàm li. Vic tìm nghim ti u có th da trên bài toán ti u li bc 2 (*Quadratic Programming*) hoc cách n gin hn là thông qua thut toán `gradient descent`. Chúng ta s s dng thut toán `Stochastic gradient descent` cp nht ln lt tng im d liu trên toàn b d liu, sau ó lp li quá trình này. i vi trng hp ma trn sn phm (**I**) c nh, ta s cn cp nht ma trn ngi dùng (**U**) theo phng gradient descent. Mi mt lt cp nht, mt ngi dùng u c la chn. Da trên thông tin v nhng sn phm mà ngi dùng u ã rating. Vector gradient descent c tính toán cp nht giá tr ca vector **u** tng ng. Quá trình này tip tc cho n khi toàn b các vector users c cp nht. Tng t nh vy i vi trng hp c nh ma trn ngi dùng c nh và cp nht ma trn sn phm.

Ti u ma trn ngi dùng:

n gin hóa quá trình tính toán, ta có th biu din hàm loss function theo tng loss function ca tng user nh sau:

$$\mathcal{L}(\mathbf{U}) = \frac{1}{2s}\sum_{n=1}^{N}\sum_{m:r_{mn}=1}(y_{mn} - i_m.u_n)^2 + \frac{\lambda_1}{2}||\mathbf{U}||_F^2$$

Trong ó r_{mn} là phn t thuc ma trn rating $R \in \mathbb{R}^{M \times N}$ có giá tr 0 hoc 1. $r_{mn} = 1$ ánh du sn phm m ã c rating bi user n và bng 0 trong trng hp cha c rating. iu kin $r_{mn} = 1$ trong hàm loss function là lc ra nhng sn phm ã c rating bi user n. Khi ó nu coi $\hat{\mathbf{I}}_n$ là ma trn các sn phm ã c rating ca user n và $\hat{y}_n$ là vector kt qu rating tng ng thì hàm loss function i vi user n c vit gn nh sau:

$$\mathcal{L}(\mathbf{U}|\sqcap\int\rceil\nabla = \backslash) = \frac{1}{2s}\sum_{m:r_{mn}=1}(y_{mn} - \mathbf{i_m}.\mathbf{u_n})^2 + \frac{\lambda_1}{2}||\mathbf{u_n}||^2 = \frac{1}{2s}||\hat{y}_n - \hat{\mathbf{I}}_n.\mathbf{u_n}||^2 + \frac{\lambda_1}{2}||\mathbf{u_n}||^2$$

o hàm ca nó tng ng:

$$\frac{\partial \mathcal{L}(\mathbf{U}|\sqcap\int\rceil\nabla = \backslash)}{\partial \mathbf{u_n}} = -\frac{1}{s}\hat{\mathbf{I}}_n^T(\hat{y}_n - \hat{\mathbf{I}}_n.\mathbf{u_n}) + \lambda_1 \mathbf{u_n}$$

Công thc cp nht nghim cho mi ct ca ma trn ngi dùng:

$$\mathbf{u}'_\mathbf{n} = \mathbf{u_n} - \theta(-\frac{1}{s}\hat{\mathbf{I}}_m^T(\hat{y}_n - \hat{\mathbf{I}}_n.\mathbf{u_n}) + \lambda_1 \mathbf{u_n})$$

Ti u ma trn sn phm:
Hoàn toàn tng t ta cng có i vi ma trn sn phm, hàm loss function i vi item = m:

$$\mathcal{L}(\mathbf{I}|\rangle\sqcup\rceil\Updownarrow = \Updownarrow) = \frac{1}{2s}\sum_{n:r_{mn}=1}(y_{mn} - \mathbf{i_m}.\mathbf{u_n})^2 + \frac{\lambda_1}{2}||\mathbf{i_m}||^2 = \frac{1}{2s}||\hat{y}_m - \mathbf{i_m}.\hat{\mathbf{U}}_m||^2 + \frac{\lambda_1}{2}||\mathbf{i_m}||^2$$

o hàm tng ng i vi mi item s là:

$$\frac{\partial \mathcal{L}(\mathbf{I}|\rangle\sqcup\rceil\Updownarrow = \Updownarrow)}{\partial \mathbf{i_m}} = -\frac{1}{s}(\hat{y}_n - \mathbf{i_m}.\hat{\mathbf{U}}_m)\hat{\mathbf{U}}_m^T + \lambda_1 \mathbf{i_m}$$

Công thc cp nht nghim cho mi dòng ca ma trn sn phm:

$$\mathbf{i}'_\mathbf{m} = \mathbf{i_m} - \theta(-\frac{1}{s}(\hat{y}_n - \mathbf{i_m}.\hat{\mathbf{U}}_m)\hat{\mathbf{U}}_m^T + \lambda_1 \mathbf{i_m})$$

0.2 2. Xây dng code thut toán

0.2.1 2.1. Thc hành trên b d liu movie length 1M

Chúng ta s thc hin phng pháp `matrix factorization` trên b d liu Movie length 1M gm 1 triu các lt ratings cho khong 4000 b phim c thu thp t 6000 ngi dùng. tin phù hp vi thut toán ã xây dng, các x lý d liu s c thc hin trên ma trn. Load d liu u vào nh sau:

```
In [1]: import pandas as pd
        import numpy as np
        columns = ['user_id', 'item_id', 'rating', 'timestamp']
        movie_length = pd.read_csv('ml-1m/ratings.dat', header = 0, \
                                   names = columns, sep = '::', engine = 'python')
        movie_length = movie_length.sort_values(['user_id', 'item_id'])
        movie_length.head()
```

```
Out[1]:     user_id  item_id  rating  timestamp
        39        1        1       5  978824268
        24        1       48       5  978824351
        38        1      150       5  978301777
        43        1      260       4  978300760
        22        1      527       5  978824195
```

Kích thc các d liu và s lng users, items

```
In [2]: print('Data movie length shape: %s'%str(movie_length.shape))
        print('No customers: %s'%str(np.unique(movie_length.iloc[:, 0]).shape[0]))
        print('No movies: %s'%str(np.unique(movie_length.iloc[:, 1]).shape[0]))
```

```
Data movie length shape: (1000208, 4)
No customers: 6040
No movies: 3706
```

Thng kê mô t s tn sut rating ca các users:

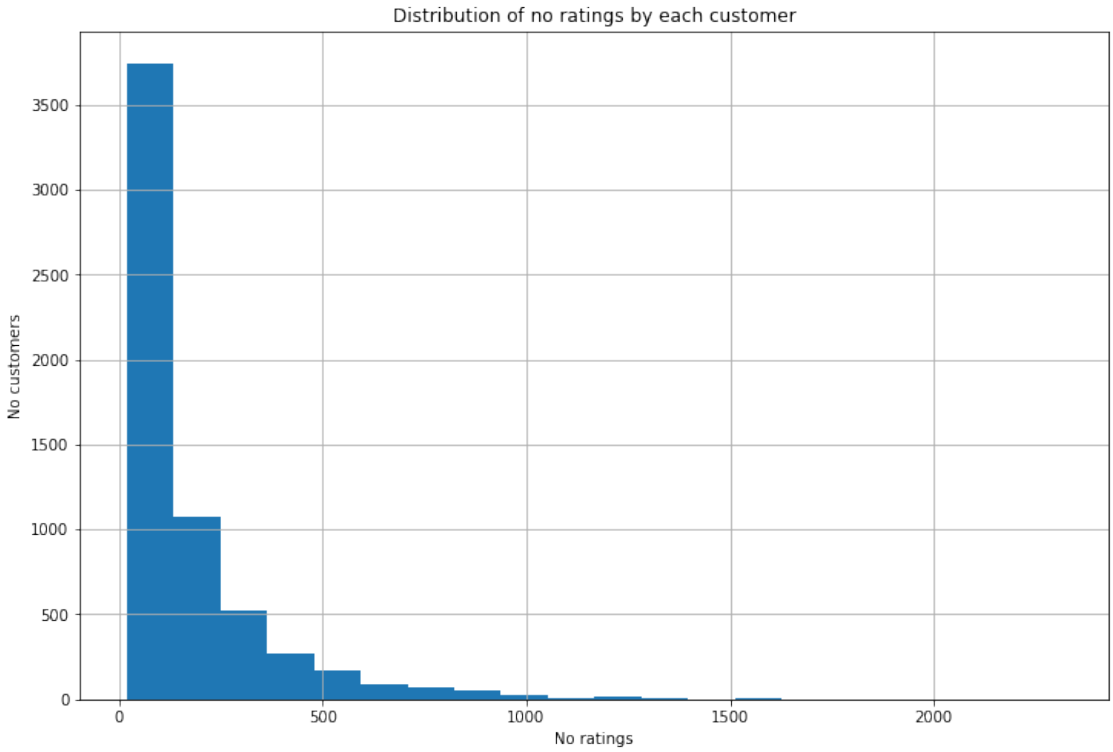
```
In [3]: movie_length['user_id'].value_counts().describe()
```

```
Out[3]: count    6040.000000
        mean      165.597351
        std       192.747126
        min        20.000000
        25%        44.000000
        50%        96.000000
        75%       208.000000
        max      2314.000000
        Name: user_id, dtype: float64
```

- Bình quân mt user rate tng cng 165 b phim.
- User thp nht rate 20 b phim và user nhiu nht rate 2314 b phim.
- Khong s lng b phim rating ph bin ca mt user là t 44 b ti 208 b phim (chim 50%).

```
In [4]: import matplotlib.pyplot as plt
        %matplotlib inline
        movie_length[['user_id', 'item_id']].groupby(['user_id']).count().\
        hist(bins = 20, figsize = (12, 8))
        plt.title('Distribution of no ratings by each customer')
        plt.xlabel('No ratings')
        plt.ylabel('No customers')
```

```
Out[4]: Text(0,0.5,'No customers')
```



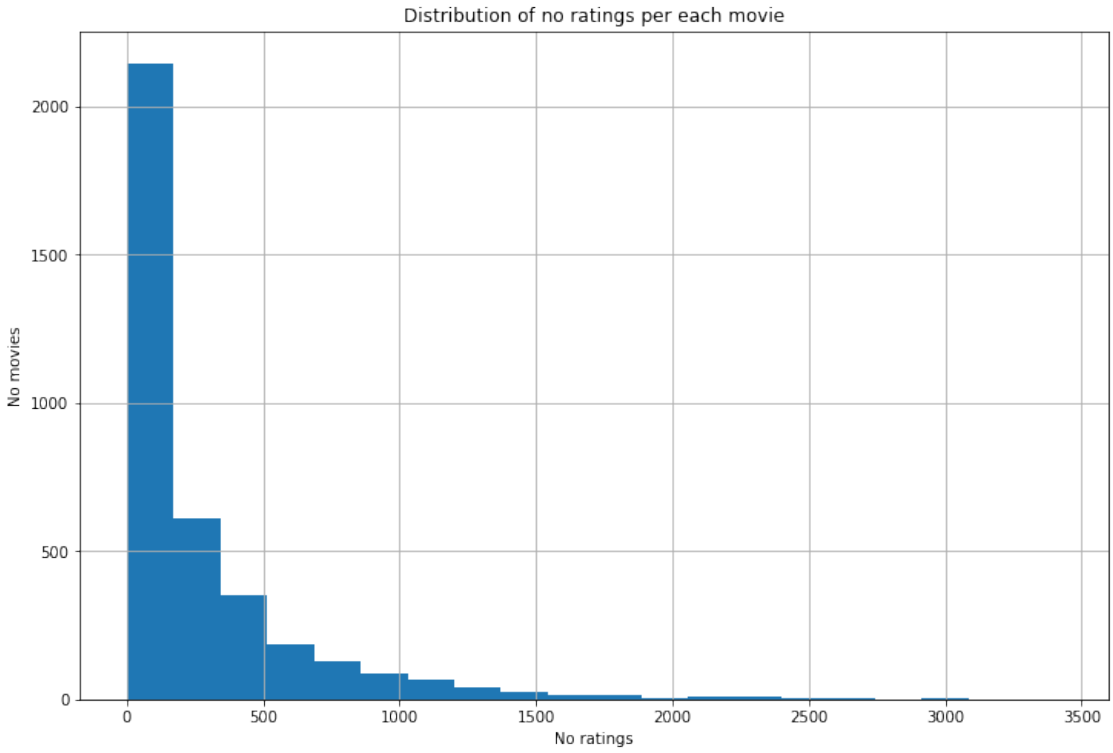
Nhìn vào phân b ca s lng user theo mc rating ta có th thy mu ca chúng ta có hin tng không cân bng khi có nhiu user rating rt ít và nhiu user rating nhiu hn. Tuy nhiên ây không phi là bài toán classification nên vic mu có kích thc mt cân bng cng không nh hng ti mc chính xác ca thut toán. Hn na thut toán `matrix factorization` xây dng hàm loss function riêng l cho tng user nêu vic user này rating bao nhiêu sn phm không nh hng n kt qu d báo rating ca user khác. Hoàn toàn tng t ta cng thng kê c s lng các user rating i vi tng b phim.

```
In [5]: movie_length['item_id'].value_counts().describe()
```

```
Out[5]: count    3706.000000
        mean      269.888829
        std       384.046815
        min         1.000000
        25%        33.000000
        50%       123.500000
        75%       350.000000
        max      3428.000000
        Name: item_id, dtype: float64
```

```
In [6]: movie_length[['user_id', 'item_id']].groupby(['item_id']).count().\
        hist(bins = 20, figsize = (12, 8))
        plt.title('Distribution of no ratings per each movie')
        plt.xlabel('No ratings')
        plt.ylabel('No movies')
```

Out[6]: Text(0,0.5,'No movies')



Mt vài ánh giá: * S ln 1 b phim c rating ít nht là 1 ln. * S ln 1 b phim c rating nhiu nht là 3428 ln. * Mc rating ph bin ca mt b phim là t 33 n 123 ln.

0.3 2.2. Thut toán matrix factorization

0.3.1 2.2.1. Các hàm trong thut toán

u tiên ta s tin hành chia mu train và test theo t l sao cho s lng rating trong tp train chim 2/3 s lng các lt rating. Cách chia mu hp lý nht là m bo t l s lng ratings xut hin trong tp train i vi s lng ratings xut hin trong tp test ca cùng mt user là bng nhau. Cách chia này m bo s công bng i vi các user khi không có user nào có quá nhiu d liu train và d liu test ít hoc d liu train quá ít nhng d liu test li quá nhiu. Giá tr c d báo t mô hình mà d liu train quá ít s thng không chun xác và làm sai lch kt qu kim tra sai s trên test.

```
In [7]: #declare split_rate for train/total ratings
        split_rate = 2/3

        def split_train_test(dataset):
            gb = dataset.groupby('user_id')
            ls = [gb.get_group(x) for x in gb.groups]
            items = [x for x in gb.groups]
            index_size = [{'i': i, 'index':gb.groups[i], 'size':len(gb.groups[i])} for i in ite
            index_train = pd.Int64Index([])
```

```
        index_test = pd.Int64Index([])
        for x in index_size:
            np.random.shuffle(x['index'].values)
            le = int(x['size']*split_rate)
            index_train = index_train.append(x['index'][:le])
            index_test = index_test.append(x['index'][le:])
        train = dataset.iloc[index_train].values
        test = dataset.iloc[index_test].values
        #minus id to 1 to index start from 0
        train[:, 0] -= 1
        train[:, 1] -= 1
        test[:, 0] -= 1
        test[:, 1] -= 1
        return train, test

train, test = split_train_test(movie_length)
```

Tin hành xây dng thut toán:

Các bin chính cho thut toán d báo bao gm: * n_users: S lng user (chính là N trong thut toán). * n_items: S lng item (chính là M trong thut toán). * K: S lng nhân t n c s dng (giá tr K trong thut toán). * theta: Tham s θ cp nht h s trong thut toán gradient descent. * split_rate: T l chia mu train/test. * lamda: Tham s hiu chnh ca thành phn hiu chnh `l2 - regularization` (n gin thit lp $\lambda_1 = \lambda_2$). * I: Ma trn sn phm. * U: Ma trn ngi dùng.

Lu ý rng các ma trn $\mathbf{I}, \mathbf{U}$ c xây dng da trên các *nhân t n* (latent feature) nên ban u ta cha xác nh c các nhân t này và phi khi to giá tr ngu nhiên cho chúng. S lng nhân t n K là mt giá tr tùy ý ta có th la chn. Theo Matrix Factorization For Recommendation System thì khi s lng nhân t n càng nhiu thut toán càng chính xác hn nhng cng làm gia tng chi phí tính toán. ng thi nhng mô hình c xây dng da trên nhân t n ã c tinh luyn có mc khác bit ln cng a ra kt qu chính xác hn các vic to ra các nhân t n ngu nhiên.

```
In [8]: n_users = np.max(train[:, 0] + 1) #plus one because index start from 0
        n_items = np.max(train[:, 1] + 1)
        n_ratings = train.shape[0]
        print('N user dimesion: %s'%n_users)
        print('M item dimesion: %s'%n_items)
        print('S Number of rating: %s'%n_ratings)
        K = 2
        theta = 0.75
        lamda = 0.2
        #Inititalize random matrix according to Gauss distribution
        I = np.random.randn(n_items, K)
        U = np.random.randn(K, n_users)
```

```
N user dimesion: 6040
M item dimesion: 3952
S Number of rating: 664826
```

```
In [9]: import scipy.sparse as sparse
        #Rating matrix
```

```
        Y = np.zeros(shape = (n_items, n_users))
        print('Y utility matrix shape: %s'%str(Y.shape))
        Y = sparse.coo_matrix((train[:, 2], (train[:, 1], train[:, 0])),\
                              shape = (n_items, n_users), dtype = np.float).toarray()
```

```
Y utility matrix shape: (3952, 6040)
```

Không phi hoàn toàn các giá tr trên ma trn **Y** u c rating. Vì vy ma trn **R** c to ra nhm ánh du các v trí c rating ca **Y** bng giá tr 1 và cha c rating bng 0.

```
In [10]: R = sparse.coo_matrix((np.ones((n_ratings,)), (train[:, 1], train[:, 0])),\
                               shape = (n_items, n_users)).toarray()
```

thut toán `gradient descent` hi t nhanh hn chúng ta cn chun hóa ma trn **Y** v giá tr kì vng bng 0 bng cách tr i mi giá tr rating trong vector rating ca mt user vi trung bình ca vector rating ó.

```
In [11]: def standardize_Y(Y):
             sum_rating = Y.sum(axis = 0)
             u_rating = np.count_nonzero(Y, axis = 0)
             u_mean = sum_rating/u_rating
             for n in range(n_users):
                 for m in range(n_items):
                     if Y[m, n] != 0:
                         Y[m, n] -= u_mean[n]
             return Y, u_mean

         Y_stad, u_mean = standardize_Y(Y)
```

Sau khi chun hóa ma trn **Y** thì phn quan trng nht là áp dng thut toán `gradient descent` ti u hóa các h s ca ma trn **U**, **I**. Da trên lý thuyt v thut toán ã xây dng mc **1.4** xây dng các hàm s update ma trn:

Thut toán gradient descent cho ma trn ngi dùng:

```
In [12]: def updateU(U):
             for n in range(n_users):
             # Matrix items include all items is rated by user n
                 i_rated = np.where(Y_stad[:, n] != 0)[0] #item's index rated by n
                 In = I[i_rated, :]
                 if In.shape[0] == 0:
                     U[:, n] = 0
                 else:
                     s = In.shape[0]
                     u_n = U[:, n]
                     y_n = Y_stad[i_rated, n]
                     grad = -1/s * np.dot(In.T,(y_n-np.dot(In, u_n))) + lamda*u_n
                     U[:, n] -= theta*grad
             return U
```

Thut toán gradient descent cho ma trn sn phm:

```
In [13]: def updateI(I):
             for m in range(n_items):
             # Matrix users who rated into item m
                 i_rated = np.where(Y_stad[m, :] != 0)[0] #user's index rated into m
                 Um = U[:, i_rated]
                 if Um.shape[1] == 0:
                     I[m, :] = 0
                 else:
                     s = Um.shape[1]
                     i_m = I[m, :]
                     y_m = Y_stad[m, i_rated]
                     grad = -1/s * np.dot(y_m - np.dot(i_m, Um), Um.T) + lamda*i_m
                     I[m, :] -= theta*grad
             return I
```

Xây dng hàm d báo ma trn Y:

Da trên ma trn **U** và **I** ta có th tính toán ma trn d báo ca **Y** là $\hat{\mathbf{Y}}$ theo công thc *(1.4.0)* và xây dng c hàm *pred()*. Các kt qu d báo cn c chuyn hóa ngc li rating bng cách cng thêm trung bình rating ca mi user vào các giá tr ratings thuc cùng 1 user. Mt s kt qu s vt quá min giá tr ca rating là [1, 5] và khi ó s c gán li v 2 u mút 1 hoc 5. Ma trn thu c s tha mãn ti mt ô ca $\hat{\mathbf{Y}}$ là kt qu d báo rating ca ngi dùng i vi sn phm tng ng. Do ta ch cn ánh giá trn **Y** trên nhng cp (user,item) ã c rating nên trong hàm *pred_train_test()* ta cn da vào ma trn rating **R** thay th nhng v trí cha c rating bng 0. Lý do hàm s này c t tên là *pred_train_test()* ó là chúng ta cng có th thc hin tng t cho tp test khi thay th giá tr cha rating bng 0.

```
In [14]: def pred(U, I):
             #predict utility matrix base on formula Y_hat = I.U
             Y_hat = np.dot(I, U)
             #invert to forecast values by plus user's mean ratings
             for n in range(n_users):
                 Y_hat[:, n] += u_mean[n]
             #convert to interger values because of rating is integer
             Y_hat = Y_hat.astype(np.int32)
             #replace values > 5 by 5 and values < 1 by 1
             Y_hat[Y_hat > 5] = 5
             Y_hat[Y_hat < 1] = 1
             return Y_hat

         def pred_train_test(Y_hat, R):
             #replace values have not yet rated by 0
             Y_pred = Y_hat.copy()
             Y_pred[R == 0] = 0
             return Y_pred
```

Xây dng hàm loss function:

Hàm loss function c xây dng da trên công thc *(1.4.1)* nh sau:

```
In [15]: def loss(Y, Y_hat):
             error = Y-Y_hat
```

```
            loss_value = 1/(2*n_ratings)*np.linalg.norm(error, 'fro')**2 + \
            lamda/2*(np.linalg.norm(I, 'fro')**2 + np.linalg.norm(U, 'fro')**2)
            return loss_value
```

S dng ma trn $\hat{\mathbf{Y}}$ d báo trên tp test

```
In [16]: Y_test = sparse.coo_matrix((test[:, 2], (test[:, 1], test[:, 0])), \
                                    shape = (n_items, n_users), dtype = np.float).toarray()
         R_test = sparse.coo_matrix((np.ones(test.shape[0]), (test[:, 1], test[:, 0])), \
                                    shape = (n_items, n_users), dtype = np.float).toarray()
```

Xây dng hàm tính RMSE:

Sau khi tính c ma trn d báo $\hat{\mathbf{Y}}$ trên tp test kt hp vi ma trn tin ích $\mathbf{Y}$ ca tp test ã bit ta s tính c *RMSE* trên tp test nh sau:

```
In [17]: import math
         def RMSE(Y_test, Y_pred):
             error = Y_test - Y_pred
             n_ratings = test.shape[0]
             rmse = math.sqrt(np.linalg.norm(error, 'fro')**2/n_ratings)
             return rmse
```

Xây dng vòng lp ti u chính:

Sau khi ã thit k c các hàm tính toán *loss function, RMSE* và các hàm ti u *gradient descent* ta s tin hành xây dng vòng lp ti u cp nht các ma trn $\mathbf{U}$ và $\mathbf{I}$ và ánh giá hiu qu ca mi bc lp thông qua giá tr ca *loss function* và *RMSE*.

```
In [19]: def fit(Umatrix, Imatrix, Ytrain, Ytest, n_iter, log_iter):
             for i in range(n_iter):
                 #update U and I
                 Umatrix = updateU(Umatrix)
                 Imatrix = updateI(Imatrix)
                 #calculate Y_hat
                 Y_hat = pred(Umatrix, Imatrix)
                 #calculate Y_hat_train by replace non ratings by 0
                 Y_pred_train = pred_train_test(Y_hat, R)
                 #calculate loss function
                 loss_value = loss(Ytrain, Y_pred_train)
                 #calculate Y_pred on test dataset
                 Y_pred_test = pred_train_test(Y_hat, R_test)
                 #calculate RMSE
                 rmse = RMSE(Ytest, Y_pred_test)
                 if i % log_iter == 0:
                     print('Iteration: {}; RMSE: {}; Loss value: {}'.format(i, rmse, loss_value
             return Y_hat, Y_pred_test
         Y_hat, Y_pred = fit(Umatrix = U, Imatrix = I, Ytrain = Y, Ytest = Y_test, n_iter = 100
```

```
Iteration: 0; RMSE: 1.0978996819873132; Loss value: 367.43589641528513
Iteration: 10; RMSE: 1.0972517727989382; Loss value: 371.61999405456453
```

```
Iteration: 20; RMSE: 1.0967013607569651; Loss value: 376.1721780533387
Iteration: 30; RMSE: 1.096062264477316; Loss value: 380.4612541939793
Iteration: 40; RMSE: 1.0953942145953313; Loss value: 384.2559773463809
Iteration: 50; RMSE: 1.0948442303904773; Loss value: 387.81426251576477
Iteration: 60; RMSE: 1.0945337212732151; Loss value: 390.15217556003586
Iteration: 70; RMSE: 1.0943198545615616; Loss value: 391.6867939243047
Iteration: 80; RMSE: 1.0941863369527691; Loss value: 392.6789203399845
Iteration: 90; RMSE: 1.0940269119371213; Loss value: 393.30221888979895
```

0.3.2 2.2.2. Xây dng class MF

Da trên các hàm ã x lý *(2.2.1)* ta s thit k class MF có chc nng x lý d liu, fiting model, ánh giá kt qu model và a ra các recommend cho khách hàng v sn phm nh sau:

Class Data x lý d liu:

```
In [20]: class Data(object):
             """
             This class used to manage data.
             Two arguments:
             dataset: pandas data frame include user_id, item_id and rating
             split_rate: number train ratings/ total ratings
             """
             def __init__(self, dataset, split_rate):
                 self.dataset = dataset
                 self.split_rate = split_rate
                 self.train, self.test = self.split_train_test(self.dataset)
                 self.Ytrain, self.Rtrain = self.utility_matrix(self.train)
                 self.Ytest , self.Rtest  = self.utility_matrix(self.test)
                 self.Ystad,  self.u_mean = self.standardize_Y(self.Ytrain)
                 self.n_users = np.max(self.train[:, 0] + 1) #plus one because index start fro
                 self.n_items = np.max(self.train[:, 1] + 1)
                 self.n_ratings = self.train.shape[0]

             def split_train_test(self, dataset):
                 "split train and test"
                 gb = dataset.groupby('user_id')
                 ls = [gb.get_group(x) for x in gb.groups]
                 items = [x for x in gb.groups]
                 index_size = [{'i': i, 'index':gb.groups[i], 'size':len(gb.groups[i])} for i
                 index_train = pd.Int64Index([])
                 index_test = pd.Int64Index([])
                 for x in index_size:
                     np.random.shuffle(x['index'].values)
                     le = int(x['size']*split_rate)
                     index_train = index_train.append(x['index'][:le])
                     index_test = index_test.append(x['index'][le:])
                 train = dataset.iloc[index_train].values
```

```
        test = dataset.iloc[index_test].values
        #minus id to 1 to index start from 0
        train[:, 0] -= 1
        train[:, 1] -= 1
        test[:, 0] -= 1
        test[:, 1] -= 1
        return train, test

    def utility_matrix(self, data_mtx):
        "create Y and R matrix"
        Y = np.zeros(shape = (n_items, n_users))
        Y = sparse.coo_matrix((data_mtx[:, 2], (data_mtx[:, 1], data_mtx[:, 0])), \
                              shape = (n_items, n_users), dtype = np.float).toarray()
        R = sparse.coo_matrix((np.ones((data_mtx.shape[0],)), (data_mtx[:, 1], data_mt
                              shape = (n_items, n_users)).toarray()
        return Y, R

    def standardize_Y(self, Y):
        "standard data to mean ratings of each user = 0"
        sum_rating = Y.sum(axis = 0)
        u_rating = np.count_nonzero(Y, axis = 0)
        u_mean = sum_rating/u_rating
        for n in range(n_users):
            for m in range(n_items):
                if Y[m, n] != 0:
                    Y[m, n] -= u_mean[n]
        return Y, u_mean
```

Class model xây dng và ánh giá model:

```
In [21]: class Model():
             """
             This class manage update U and I matrix, predict and evaluate error
             Four arguments:
             data: instance from Data class which supplies the data for model
             theta: learning rate
             lamda: regularization parameter
             K: number of latent factors
             """
             def __init__(self, data, theta, lamda, K):
                 self.data = data
                 self.theta = theta
                 self.lamda = lamda
                 self.K = K
                 self.I = np.random.randn(data.n_items, K)
                 self.U = np.random.randn(K, data.n_users)
```

```
def updateU(self):
    for n in range(self.data.n_users):
    # Matrix items include all items is rated by user n
        i_rated = np.where(self.data.Ystad[:, n] != 0)[0] #item's index rated by
        In = self.I[i_rated, :]
        if In.shape[0] == 0:
            self.U[:, n] = 0
        else:
            s = In.shape[0]
            u_n = self.U[:, n]
            y_n = self.data.Ystad[i_rated, n]
            grad = -1/s * np.dot(In.T,(y_n-np.dot(In, u_n))) + self.lamda*u_n
            self.U[:, n] -= self.theta*grad

def updateI(self):
    for m in range(self.data.n_items):
    # Matrix users who rated into item m
        i_rated = np.where(self.data.Ystad[m, :] != 0)[0] #user's index rated int
        Um = self.U[:, i_rated]
        if Um.shape[1] == 0:
            self.I[m, :] = 0
        else:
            s = Um.shape[1]
            i_m = self.I[m, :]
            y_m = self.data.Ystad[m, i_rated]
            grad = -1/s * np.dot(y_m - np.dot(i_m, Um), Um.T) + self.lamda*i_m
            self.I[m, :] -= self.theta*grad

def pred(self, I, U):
    #predict utility matrix base on formula Yhat = I.U
    Yhat = np.dot(I, U)
    #invert to forecast values by plus user's mean ratings
    for n in range(self.data.n_users):
        Yhat[:, n] += self.data.u_mean[n]
    #convert to interger values because of rating is integer
    Yhat = Yhat.astype(np.int32)
    #replace values > 5 by 5 and values < 1 by 1
    Yhat[Yhat > 5] = 5
    Yhat[Yhat < 1] = 1
    return Yhat

def pred_train_test(self, Yhat, R):
    #replace values have not yet rated by 0
    Y_pred = Yhat.copy()
    Y_pred[R == 0] = 0
    return Y_pred

def loss(self, Y, Yhat):
```

```
        error = Y-Yhat
        n_ratings = np.sum(Y != 0)
        loss_value = 1/(2*n_ratings)*np.linalg.norm(error, 'fro')**2 +\
        lamda/2*(np.linalg.norm(self.I, 'fro')**2 + \
                 np.linalg.norm(self.U, 'fro')**2)
        return loss_value

    def RMSE(self, Y, Yhat):
        error = Y - Yhat
        n_ratings = np.sum(Y != 0)
        rmse = math.sqrt(np.linalg.norm(error, 'fro')**2/n_ratings)
        return rmse
```

Xây dng class MF qun lý model và data:

```
In [22]: class MF():
             """
             This class used to manage model and data
             Two main arguments:
             data: control the data
             model: control the functions which execute model
             """
             def __init__(self, data, model, n_iter, print_log_iter):
                 self.data = data
                 self.model = model
                 self.n_iter = n_iter
                 self.print_log_iter = print_log_iter
                 self.Y_pred_train = None
                 self.Y_pred_test = None
                 self.Yhat = None

             def fit(self):
                 for i in range(self.n_iter):
                     #update U and I
                     self.model.updateU()
                     self.model.updateI()
                     #calculate Y_hat
                     self.Yhat = self.model.pred(self.model.I, self.model.U)
                     #calculate Y_pred_train by replace non ratings by 0
                     self.Y_pred_train = self.model.pred_train_test(self.Yhat, self.data.Rtrain
                     self.Y_pred_test  = self.model.pred_train_test(self.Yhat, self.data.Rtest)
                     if i % self.print_log_iter == 0:
                         print('Iteration: {}; RMSE: {}; Loss value: {}'.\
                               format(i, self.model.RMSE(self.data.Ytest, self.Y_pred_test),\
                                      self.model.loss(self.data.Ytrain, self.Y_pred_train)))

             def recommend_for_user(self, user_id, k_neighbors):
                 recm = np.concatenate((np.arange(1, self.Y_pred_test.shape[0]+1).reshape(-1,
```

```
                                        self.Y_pred_test[:, user_id - 1].reshape(-1, 1)), axis
                recm.sort(axis = 0)
                print('Top %s item_id recommended to user_id %s: %s'%\
                      (k_neighbors, user_id, str(recm[-k_neighbors:, 0])))
```

```
In [23]: data = Data(dataset = movie_length, split_rate = 2/3)
         model = Model(data = data, theta = 0.75, lamda = 0.1, K = 2)
         mf = MF(data = data, model = model, n_iter = 100, print_log_iter = 10)
         mf.fit()
```

```
Iteration: 0; RMSE: 1.2230236685162748; Loss value: 681.8471565701848
Iteration: 10; RMSE: 1.1780718373263668; Loss value: 286.40240426148296
Iteration: 20; RMSE: 1.087117979096145; Loss value: 518.2976746384784
Iteration: 30; RMSE: 1.08441027204107; Loss value: 534.4106319526331
Iteration: 40; RMSE: 1.08385608972465; Loss value: 546.1164972768984
Iteration: 50; RMSE: 1.0828157171073196; Loss value: 559.6753058301773
Iteration: 60; RMSE: 1.079754860855491; Loss value: 575.3332305811103
Iteration: 70; RMSE: 1.0750987045005322; Loss value: 590.821120508004
Iteration: 80; RMSE: 1.0709110406988624; Loss value: 602.5417655852891
Iteration: 90; RMSE: 1.0685236770672915; Loss value: 608.9641965725858
```

Ta nhn thy kt qu ca hàm `loss function` và `RMSE` gim dn sau các vòng lp. iu ó cho thy thut toán `gradient descent` ã phát huy tác dng trong vic làm gim sai s d báo. Tuy nhiên ôi khi chúng ta s gp tình hung `loss function` và `RMSE` tng dn. Có nhiu nguyên nhân dn ti iu này chng hn nh h s `learning rate` và `regularization` c thit lp quá cao làm thut toán nhy ra khi cc tr toàn cc hoc cng có th các `loss function` ch tng tm thi và gim sau ó nghim di chuyn qua im cc tr a phng. i vi kh nng 1 chúng ta cn iu chnh li các h s `learning rate` và `regularization` thut toán di chuyn úng hng ti nghim ti u. Kh nng th 2 không quá nghiêm trng bi thut toán có th di chuyn ti nghim ti u ngay sau ó.

Recommend 10 sn phm tim nng nht cho user_id = 200:

```
In [24]: mf.recommend_for_user(user_id = 200, k_neighbors = 10)
```

```
Top 10 item_id recommended to user_id 200: [3943 3944 3945 3946 3947 3948 3949 3950 3951 3952]
```

0.4 2.3. Tài liu tham kho

1. Recommendation System - Stanford
2. Collaborative Filtering Youtube - Stanford
3. Recommendation System - Machine Learning - Andrew Ng
4. Matrix Factorization - Machine Learning C Bn - Tiep Huu Vu
5. Matrix Factorization techniques for recommender systems
6. Learning from Incomplete Ratings Using Non-negative Matrix Factorization
7. Matrix Factorization - Albert Au Yeung
8. Algorithms for Non-negative Matrix Factorization - Daniel D.Lee and H. Sebastian Seung