SGD Algorithm to predict movie ratings

There will be some functions that start with the word "grader" ex: grader_matrix(), grader_mean() those function definition.

Every Grader function has to return True.

1. Download the data from here
2. The data will be of this format, each data point is represented as a triplet of user_i

user_id	movie_id	rating
77	236	3
471	208	5
641	401	4
31	298	4
58	504	5
235	727	5

Task 1

Predict the rating for a given (user_id, movie_id) pair

Predicted rating $\hat{y}_{ij}$ for user i, movied j pair is calcuated as $\hat{y}_{ij} = \mu + b_i + c_j + u_i^T v_j$, here we using SGD algorithm with the optimization problem for N users and M movies is defined as

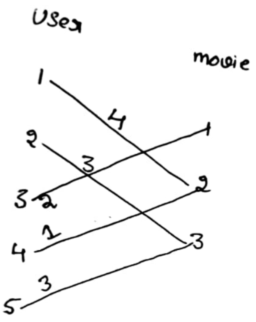
$$L = \min_{b, c, \{u_i\}_{i=1}^N, \{v_j\}_{j=1}^M} \quad \alpha \Big(\sum_j \sum_k v_{jk}^2 + \sum_i \sum_k u_{ik}^2 + \sum_i b_i^2$$
$$\sum_{i,j \in \mathcal{I}^{\text{train}}} (y_{ij} - \mu - b_i - c_j - u_i^T v_j)^2$$

- μ : scalar mean rating
- b_i : scalar bias term for user i
- c_j : scalar bias term for movie j
- u_i : K-dimensional vector for user i
- v_j : K-dimensional vector for movie j

*. We will be giving you some functions, please write code in that functions only.

*. After every function, we will be giving you expected output, please make sure that you get that o

1. Construct adjacency matrix with the given data, assuming its [weighted un-directed bi-partite](#) rating given by user to the movie



the Adjacency matrix

$$\begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 0 & 4 & 0 \\ 2 & 0 & 0 & 3 \\ 3 & 2 & 0 & 0 \\ 4 & 0 & 1 & 0 \\ 5 & 0 & 0 & 3 \end{array}$$

you can construct this matrix like $A[i][j] = r_{ij}$ here i is user_id, j is movie_id and r_{ij} is rating giv

Hint : you can create adjacency matrix using [csr_matrix](#)

2. We will Apply SVD decomposition on the Adjaceny matrix [link1](#), [link2](#) and get three matrices if A is of dimensions $N \times M$ then
   U is of $N \times k$,
   $\sum$ is of $k \times k$ and
   V is $M \times k$ dimensions.

   *. So the matrix U can be represented as matrix representation of users, where each row u_i

   *. So the matrix V can be represented as matrix representation of movies, where each row v movie.

3. Compute μ , μ represents the mean of all the rating given in the dataset.(write your code in c
4. For each unique user initilize a bias value B_i to zero, so if we have N users B will be a N di corresponds to the bias term for i^{th} user (write your code in def initialize())
5. For each unique movie initilize a bias value C_j zero, so if we have M movies C will be a M will corresponds to the bias term for j^{th} movie (write your code in def initialize())
6. Compute dL/db_i (Write you code in def derivative_db())

7. Compute dL/dc_j(write your code in def derivative_dc()
8. Print the mean squared error with predicted ratings.

```
for each epoch:
    for each pair of (user, movie):
        b_i =  b_i - learning_rate * dL/db_i
        c_j =  c_j - learning_rate * dL/dc_j
predict the ratings with formula
```

$$\hat{y}_{ij} = \mu + b_i + c_j + \text{dot\_product}(u_i, v_j)$$

9. you can choose any learning rate and regularization term in the range 10^{-3} to 10^2
10. **bonus**: instead of using SVD decomposition you can learn the vectors u_i, v_j with the help of

Task 2

As we know U is the learned matrix of user vectors, with its i-th row as the vector ui for user i. Each
for a particular user.

The question we'd like to investigate is this: do our computed per-user features that are optimized
to do with gender?

The provided data file user_info.csv contains an is_male column indicating which users in the data
given the features U?

> **Note 1** : there is no train test split in the data, the goal of this assignment is to give an intution
> factorization with the help of SGD and application of truncated SVD. for better understanding
> check netflix case study.
>
> **Note 2** : Check if scaling of U, V matrices improve the metric

20/1591773525000/00484516897554883881/03543900857199698311/1PHFdJh_4gIPiLH5Q4UErH8GK71hTrzI

```
--2020-06-10 07:20:18--  https://doc-0k-0g-docs.googleusercontent.com/docs/securesc/45
Resolving doc-0k-0g-docs.googleusercontent.com (doc-0k-0g-docs.googleusercontent.com).
Connecting to doc-0k-0g-docs.googleusercontent.com (doc-0k-0g-docs.googleusercontent.c
HTTP request sent, awaiting response... 200 OK
Length: 12073 (12K) [text/plain]
Saving to: ‘user_info.csv.txt’

user_info.csv.txt   100%[===================>]  11.79K  --.-KB/s    in 0s
```

Reading the csv file

```
)2/1591769175000/00484516897554883881/03543900857199698311/1-1z7iDB52cB6_JpO7Dqa-eOYSs-mivp
```

```
-2020-06-10 06:06:55--  https://doc-0g-0g-docs.googleusercontent.com/docs/securesc/459
esolving doc-0g-0g-docs.googleusercontent.com (doc-0g-0g-docs.googleusercontent.com)..
onnecting to doc-0g-0g-docs.googleusercontent.com (doc-0g-0g-docs.googleusercontent.cc
ITTP request sent, awaiting response... 200 OK
ength: 880367 (860K) [text/csv]
aving to: ‘ratings_train.csv’

atings_train.csv    100%[===================>] 859.73K  --.-KB/s    in 0.006s

020-06-10 06:06:55 (130 MB/s) - ‘ratings_train.csv’ saved [880367/880367]
```

```
data1=pd.read_csv('user_info.csv.txt')
```

```
import pandas as pd
data=pd.read_csv('ratings_train.csv')
data.head()
```

	user_id	item_id	rating
0	772	36	3
1	471	228	5
2	641	401	4
3	312	98	4
4	58	504	5

```
data.shape
```

```
(89992, 3)
```

Create your adjacency matrix

```
from scipy.sparse import csr_matrix
```

```
adjacency_matrix = csr_matrix((data.rating.values,(data.user_id.values,data.item_id.values
```

```
adjacency_matrix.shape
```

```
(943, 1681)
```

Grader function - 1

```
def grader_matrix(matrix):
  assert(matrix.shape==(943,1681))
  return True
grader_matrix(adjacency_matrix)
```

```
True
```

SVD decompostion

Sample code for SVD decompostion

```
from sklearn.utils.extmath import randomized_svd
import numpy as np
matrix = np.random.random((20, 10))
U, Sigma, VT = randomized_svd(matrix, n_components=5,n_iter=5, random_state=None)
print(U.shape)
print(Sigma.shape)
print(VT.T.shape)
```

```
(20, 5)
(5,)
(10, 5)
```

Write your code for SVD decompostion

```
# Please use adjacency_matrix as matrix for SVD decompostion
# You can choose n_components as your choice
from sklearn.utils.extmath import randomized_svd
import numpy as np
matrix = np.random.random((943, 1681))
U, Sigma, VT = randomized_svd(adjacency_matrix, n_components=50,n_iter=50, random_state=No
print(U.shape)
print(Sigma.shape)
print(VT.T.shape)
```

```
(943, 50)
(50,)
(1681, 50)
```

Compute mean of ratings

```
def m_u(ratings):
    '''In this function, we will compute mean for all the ratings'''
    # you can use mean() function to do this
    # check this (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFr


    return np.mean(ratings)


mu=m_u(data['rating'])
print(mu)
```

```
3.529480398257623
```

Grader function -2

```
def grader_mean(mu):
  assert(np.round(mu,3)==3.529)
  return True
mu=m_u(data['rating'])
grader_mean(mu)
```

```
True
```

Initialize B_i and C_j

Hint : Number of rows of adjacent matrix corresponds to user dimensions(B_i), number of column dimensions (C_j)

```
def initialize(dim):
    '''In this function, we will initialize bias value 'B' and 'C'.'''
    # initalize the value to zeros
    # return output as a list of zeros
    temp=np.zeros(dim)



    return temp


dim=943 # give the number of dimensions for b_i (Here b_i corresponds to users)
b_i=initialize(dim)


dim=1681 # give the number of dimensions for c_j (Here c_j corresponds to movies)
c_j=initialize(dim)
```

Grader function -3

```
def grader_dim(b_i,c_j):
  assert(len(b_i)==943 and np.sum(b_i)==0)
  assert(len(c_j)==1681 and np.sum(c_j)==0)
  return True
grader_dim(b_i,c_j)
```

```
True
```

Compute dL/db_i

```
def derivative_db(user_id,item_id,rating,U,V,mu,alpha):
    db=2*alpha*b_i[user_id]-2*(rating -mu-b_i[user_id]-c_j[item_id]- np.dot(U[user_id].T,V

    return db
```

Grader function -4

```
def grader_db(value):
    assert(np.round(value,3)==-0.931)
    return True
U1, Sigma, V1 = randomized_svd(adjacency_matrix, n_components=2,n_iter=5, random_state=24)
# Please don't change random state
# Here we are considering n_componets = 2 for our convinence
alpha=0.01
value=derivative_db(312,98,4,U1,V1,mu,alpha)
grader_db(value)
```

```
True
```

Compute dL/dc_j

```
def derivative_dc(user_id,item_id,rating,U,V,mu):
    dc=2*alpha*c_j[item_id]-2*(rating -mu-b_i[user_id]-c_j[item_id]- np.dot(U[user_id].T,V


    return dc
```

Grader function - 5

```
def grader_dc(value):
    assert(np.round(value,3)==-2.929)
    return True
U1, Sigma, V1 = randomized_svd(adjacency_matrix, n_components=2,n_iter=5, random_state=24)
# Please don't change random state
# Here we are considering n_componets = 2 for our convinence
r=0.01
value=derivative_dc(58,504,5,U1,V1,mu)
```

```
grader_dc(value)
```

```
True
```

Compute MSE (mean squared error) for predicted ratings

for each epoch, print the MSE value

```
 for each epoch:

     for each pair of (user, movie):

         b_i =  b_i - learning_rate * dL/db_i

         c_j =  c_j - learning_rate * dL/dc_j

 predict the ratings with formula
```

$$\hat{y}_{ij} = \mu + b_i + c_j + \text{dot\_product}(u_i, v_j)$$

```
from sklearn.metrics import mean_squared_error
mu=m_u(data["rating"])
y=data["rating"]
mes_y=[]
for epoch in range(0,50):
  y_i_j=[]
  for i in range(0,data.shape[0]):
    user_id1=data["user_id"][i]
    item_id1=data["item_id"][i]
    rating1=data['rating'][i]
    learing_rate=0.01
    b_i[user_id1]=b_i[user_id1]-learing_rate*(2*alpha*b_i[user_id1]-2*(rating1 -mu-b_i[use
    c_j[item_id1]=c_j[item_id1]-learing_rate*(2*alpha*c_j[item_id1]-2*(rating1 -mu-b_i[use
    y_i_j_temp=mu+b_i[user_id1]+c_j[item_id1]+np.dot(U[user_id1],VT[:,item_id1])
    y_i_j.append(y_i_j_temp)
  mes=mean_squared_error(y,y_i_j)
  print("for epoch = " ,epoch,"mes = ",mes)
  mes_y.append(mes)
```

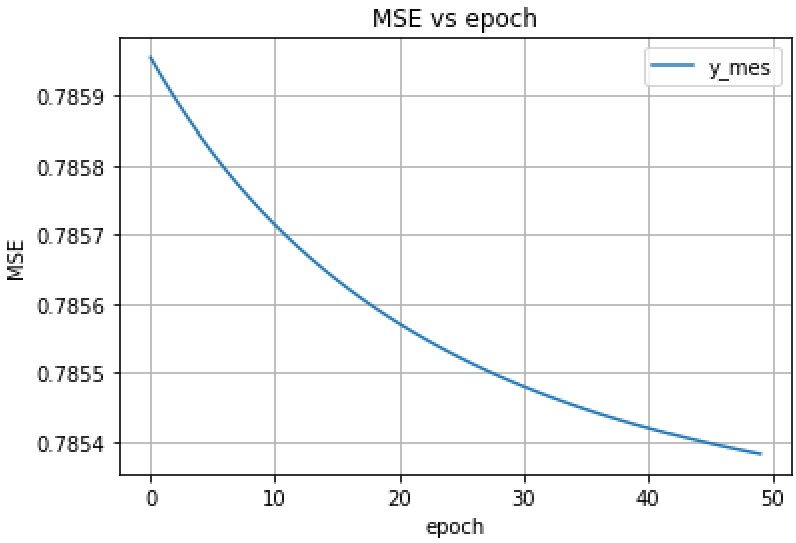
```
for epoch =  0 mes =  0.7859543895733097
for epoch =  1 mes =  0.7859236375626233
for epoch =  2 mes =  0.78589471555363
for epoch =  3 mes =  0.7858674751222213
for epoch =  4 mes =  0.7858417837327881
for epoch =  5 mes =  0.7858175226412596
for epoch =  6 mes =  0.7857945851210896
for epoch =  7 mes =  0.7857728749560172
for epoch =  8 mes =  0.7857523051542519
for epoch =  9 mes =  0.7857327968472996
for epoch =  10 mes =  0.78571427834341
for epoch =  11 mes =  0.7856966843110432
for epoch =  12 mes =  0.7856799550720919
for epoch =  13 mes =  0.7856640359880974
for epoch =  14 mes =  0.785648876925541
for epoch =  15 mes =  0.7856344317886024
for epoch =  16 mes =  0.7856206581096715
for epoch =  17 mes =  0.7856075166894547
for epoch =  18 mes =  0.7855949712797976
for epoch =  19 mes =  0.7855829883034146
for epoch =  20 mes =  0.7855715366055886
for epoch =  21 mes =  0.7855605872336466
for epoch =  22 mes =  0.7855501132406321
for epoch =  23 mes =  0.785540089510108
for epoch =  24 mes =  0.7855304925994631
for epoch =  25 mes =  0.785521300599456
for epoch =  26 mes =  0.7855124930080492
for epoch =  27 mes =  0.7855040506168415
for epoch =  28 mes =  0.7854959554086377
for epoch =  29 mes =  0.7854881904648812
for epoch =  30 mes =  0.785480739881845
for epoch =  31 mes =  0.785473588694609
for epoch =  32 mes =  0.78546672280798
for epoch =  33 mes =  0.7854601289336145
for epoch =  34 mes =  0.7854537945326822
for epoch =  35 mes =  0.7854477077635089
for epoch =  36 mes =  0.7854418574336799
for epoch =  37 mes =  0.7854362329561642
for epoch =  38 mes =  0.7854308243090566
for epoch =  39 mes =  0.785425621998589
for epoch =  40 mes =  0.7854206170250959
for epoch =  41 mes =  0.7854158008516555
for epoch =  42 mes =  0.7854111653751581
for epoch =  43 mes =  0.7854067028995784
for epoch =  44 mes =  0.7854024061112529
```

Plot epoch number vs MSE

- epoch number on X-axis
- MSE on Y-axis

```
#plot log_loss vs epoch
import matplotlib.pyplot as plt
a=list(range(0,50))
plt.plot(a,mes_y,label='y_mes')
plt.xlabel("epoch")
plt.ylabel("MSE")
plt.title("MSE vs epoch")
plt.legend()
```

```
plt.grid()
plt.show()
```



Task 2

```
data1=pd.read_csv('user_info.csv.txt')
data1.head()
```

	user_id	age	is_male	orig_user_id
0	0	24	1	1
1	1	53	0	2
2	2	23	1	3
3	3	24	1	4
4	4	33	0	5

Training model logistic regression

As we know U is the learned matrix of user vectors, with its i-th row as the vector ui for user i. Each for a particular user.

The question we'd like to investigate is this: do our computed per-user features that are optimized to do with gender?

The provided data file user_info.csv contains an is_male column indicating which users in the data given the features U?

```
from sklearn import linear_model
```

```
Y=data1['is_male']
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
 early_stopping=False, epsilon=0.1, eta0=0.0001,
 fit_intercept=True, l1_ratio=0.15, learning_rate='constant',
 loss='log', max_iter=1000, n_iter_no_change=5, n_jobs=None,
 penalty='l2', power_t=0.5, random_state=15, shuffle=True,
 tol=0.001, validation_fraction=0.1, verbose=2, warm_start=False)


clf = linear_model.SGDClassifier(alpha=0.0001,eta0=0.0001,max_iter=1000,loss='log',penalty
clf.fit(U,Y)
```

```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
              early_stopping=False, epsilon=0.1, eta0=0.0001,
              fit_intercept=True, l1_ratio=0.15, learning_rate='optimal',
              loss='log', max_iter=1000, n_iter_no_change=5, n_jobs=None,
              penalty='l2', power_t=0.5, random_state=None, shuffle=True,
              tol=0.001, validation_fraction=0.1, verbose=0, warm_start=False)
```

```
user_male_pred = clf.predict(U)
```

confusion_matrix

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
def plot_confusion_matrix(test_y, predict_y):
 C = confusion_matrix(test_y, predict_y)

 A =(((C.T)/(C.sum(axis=1))).T)

 B =(C/C.sum(axis=0))
 plt.figure(figsize=(20,4))

 labels = [0,1]
 # representing A in heatmap format
 cmap=sns.light_palette("blue")
 plt.subplot(1, 3, 1)
 sns.heatmap(C, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)
 plt.xlabel('Predicted Class')
 plt.ylabel('Original Class')
 plt.title("Confusion matrix")

 plt.subplot(1, 3, 2)
 sns.heatmap(B, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)
 plt.xlabel('Predicted Class')
 plt.ylabel('Original Class')
 plt.title("Precision matrix")

 plt.subplot(1, 3, 3)
 # representing B in heatmap format
 sns.heatmap(A, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)
```

```
plt.xlabel('Predicted Class')
plt.ylabel('Original Class')
plt.title("Recall matrix")

plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarnin
  import pandas.util.testing as tm
```

```
print('Train confusion_matrix')
plot_confusion_matrix(Y,user_male_pred)
```

```
Train confusion_matrix
```