Long Term Project Report

(Recommendation System)

2016025478 서경욱

1. 개발 환경

- 파이썬 3.8.6

```
C:₩Users₩user>python --version
Python 3.8.6
```

2. 실행 방법

```
C:\Users\user\Desktop\4-1\데이터사이언스\Long term project_ Recommender System
(datascience) λ python recommender.py u1.base u1.test
```

- python recommender.py (training data path) (test data path)

```
matrix_factorizer = ALS_MatrixFactorizer(
    R=rating_matrix, nf=3, reg_param=0.1, epochs=20)
```

```
"""
R : rating matrix
nf : dim of latent matrix
_lambda : for normalization
epochs : max iteration
X user, Y item
"""
```

- ALS를 위한 hyperparameter 설정

3. Summary of Algorithm

본 과제에서는 각 영화에 대한 각 사용자의 평점 데이터를 기반으로 test data의 사용자의 영화에 대한 평점을 predict를 수행했다. 여러 approach들 중 Collaborative Filtering방식을 이용하여 다른 사람의 취향을 같이보며 predict했다. CF방식으로는 **Matrix Factorization** 방법을 사용했다.

Matrix Factorization 방법은 original rating matrix(user*item)을 **user latent factor matrix**(len(users) * nf) X, **item latent factor matrix**(len(items) * nf) Y라는 두 matrix로 나눈다. 구하고자 하는 predict matrix는 두 행렬을 곱해 평점 예측 행렬을 만든다. 각 X,Y를 곱한 행렬의 오차를 최소화 시키기위해 loss function을 정의하고 이를 minimize시키는 optimize를 통해 최적화된 X,Y, 그리고 둘의 곱인 R'을 구할 수 있다.

최적화를 수행하는 대표적인 알고리즘으로는 Gradient descent와 **Alternating Least Squeares(ALS)** 알고리즘이 있다.

$$\forall u_i : J(u_i) = ||R_{i\cdot} - u_i \times V^T||_2 + \lambda \cdot ||u_i||_2$$
$$\forall v_j : J(v_j) = ||R_{i\cdot} - U \times {v_j}^T||_2 + \lambda \cdot ||v_j||_2$$
$$u_i = (V^T \times V + \lambda I)^{-1} \times V^T \times R_{i\cdot}$$
$$v_j = (U^T \times U + \lambda I)^{-1} \times U^T \times R_{\cdot j}$$

과제의 구현에서는 ALS 알고리즘을 optimizer로 이용했다. ALS 알고리즘은 user latent factor matrix(여기에선 U), item latent factor matrix(여기에선 V) 두 행렬을 번갈아가며 학습시킨다. 하나의 행렬을 학습시킬 때 다른 하나의 행렬은 고정해놓고 최적화를 진행하는 방식이다. 여기서 λ는 model이 overfitting하지 않도록 방지하는 역할이다. ALS Matrix Factorizer를 구현하고 여러 parameter, 조건 등을 수정해가며 최적의 결과를 도출하고자 했다.

4. Detailed description of each function

```
if __name__ == '__main__':
    start = time.time()
    # check correct parameter
    if len(sys.argv) != 3:
        print("Insufficient arguments")
        sys.exit()

    training_data = sys.argv[1]
    test_data = sys.argv[2]

    train_df = pd.read_table(training_data, sep="\t", names=[
        'user_id', 'item_id', 'rating', 'time_stamp'])
    train_df.drop('time_stamp', axis=1, inplace=True)

    rating_matrix = train_df.pivot_table(
        'rating', index='user_id', columns='item_id').fillna(0).to_numpy()
```

- main의 첫번째 파트이다.
- 실행 parameter의 개수가 적절한지 확인 후 아니라면 exit한다.
- training data는 column의 이름이 명시되지 않은 형태이다. pandas의 read_table 함수를 이용하여 column들의 name을 정의하고 형태에 맞게 ₩t을 delimiter로 나눠 가져왔다.
- training data의 **time_stamp data**는 predict에 영향이 없다고 판단하여 제거해주었다.
- raiting_matrix를 가져온 training data를 행렬 형태로 만들어 저장한다. 각 cell의 값들은 영화에 대한 rating이다. 이 때 training data는 상당히 sparse하고 Nan이라는 결측치가 많다. ALS optimize를 위해서는 결측치가 존재해서는 안되므로 결측치를 0으로 채워주고 진행한다.
- **결측치 채우기는 0,1,2,3으로 진행해본 결과 0이 가장 결과가 좋았다.**

```
matrix_factorizer = ALS_MatrixFactorizer(
    R=rating_matrix, nf=3, reg_param=0.1, epochs=20)

print(" 학습 시작 ")
matrix_factorizer.fit()
res = matrix_factorizer.X.dot(matrix_factorizer.Y.T)

output_file = open("u" + str(sys.argv[1][1]) + ".base_prediction.txt", "w")
for i in range(res.shape[0]):
    for j in range(res.shape[1]):
        print(str(i+1)+'\t'+str(j+1)+'\t'+str(res[i][j]), file=output_file)

print("time:  ", str(time.time()-start))
```

- main의 두번째 파트이다.
- user의 편의에 맞게 nf(latent matrix의 차원), reg_param(regularizationi 위한 parameter), epochs(iteration 횟수)를 설정한다.
- matrix_factorizer 학습으로 해당 object의 X,Y를 train한다.
- 두 행렬의 곱으로 산출한 R'를 만들고 이를 과제 명세에 따른 형태로 저장한다.
- nf는 latent의 matrix의 차원으로 차원 수가 높을수록 기존 값들을 잘 반영할 것 같지만 아니다. 3,5,40,100,200 등의 값을 조정해가며 RMSE를 산출해본 결과 3이 best였다.

- reg_param 즉, λ는 여러 case들을 참조한 결과 0.1이 rule of thumb이라는 것을 알게 되었다.
- epochs는 ALS에서 거의 epoch 20내로 수렴한다는 여러 자료들을 확인하여 5, 10, 20에서 수행했다

```
class ALS_MatrixFactorizer():
    def __init__(self, R, nf, reg_param, epochs):
        """
        R : rating matrix
        nf : dim of latent matrix
        _lambda : for normalization
        epochs : max iteration
        X user, Y item
        """
        self.R = R
        self.nf = nf
        self._lambda = reg_param
        self.epochs = epochs

    def fit(self):
        self.X = np.random.normal(size=(self.R.shape[0], self.nf))
        self.Y = np.random.normal(size=(self.R.shape[1], self.nf))

        for epoch in range(self.epochs):
            start_epoch = time.time()
            self.optimize_X()
            self.optimize_Y()

            RMSE = self.train_RMSE()
            print("Epoch %d / train_RMSE = %.4f" % (epoch + 1, RMSE))

            print(str(epoch + 1) + " is     done!")
            print("time spent :      %d s" % (time.time()-start_epoch))
```

- init에서는 각 parameter를 초기화해준다.
- fit(학습)에서는 X(user), Y(item => movie)를 정규분포를 가지는 random한 값들로 채운 행렬로 초기화해준다.
- 각 epoch에서 X, Y를 순서대로 optimize하고 학습이 제대로 진행되고 있는지 보기위해 RMSE를 epoch마다 print하도록 했다.
- 기존에는 X,Y의 초기화에서 **np.random.random** 값으로 0에서 1사이 random한 값들로 초기화했었지만 **np.random.normal**이 RMSE측면 성능이 더 좋아서 이용했다.

```
def optimize_X(self):
    for i, Ri in enumerate(self.R):
        front = np.linalg.inv(
            np.dot(self.Y.T, np.dot(np.diag(Ri), self.Y)) + self._lambda * np.identity(self.nf))
        rear = np.dot(self.Y.T, np.dot(np.diag(Ri), self.R[i].T))
        self.X[i] = np.matmul(front, rear)

def optimize_Y(self):
    for j, Rj in enumerate(self.R.T):
        front = np.linalg.inv(
            np.dot(self.X.T, np.dot(np.diag(Rj), self.X)) + self._lambda * np.identity(self.nf))
        rear = np.dot(self.X.T, np.dot(np.diag(Rj), self.R[:, j]))
        self.Y[j] = np.matmul(front, rear)

def train_RMSE(self):
    Ri, Rj = self.R.nonzero()
    cost = 0
    for i, j in zip(Ri, Rj):
        cost += pow(self.R[i, j] - self.X[i, :].dot(self.Y[j, :].T), 2)
    return np.sqrt(cost/len(Ri))
```

- optimize_X, optimize_Y는 각각 2페이지에서의 수식을 구현하여 X,Y matrix를 최적화해주는 함수이다.
- train_RMSE는 학습 중간에 RMSE가 감소하는, 실제값과 예측값의 차이가 줄어가는지, 학습이 올바르게 되고 있는지 확인하기 위한 함수이다. for문에서 user i, item j에 대한 예측값과 original rating matrix R의 user i, item j의 값에 대한 차이로 RMSE를 구했다.

5. 실행 결과

- 실행 시간

```
time spent :      25 s
Epoch 4 / train_RMSE = 0.8994
4 is     done!
time spent :      25 s
Epoch 5 / train_RMSE = 0.8888
5 is     done!
time spent :      25 s
time:   134.3847725391388
```

(epoch 5)

```
time spent :      24 s
Epoch 9 / train_RMSE = 0.8692
9 is     done!
time spent :      25 s
Epoch 10 / train_RMSE = 0.8686
10 is     done!
time spent :      25 s
time:   265.50213646888733
```

(epoch10)

```
17 is     done!
time spent :      25 s
Epoch 18 / train_RMSE = 0.8688
18 is     done!
time spent :      24 s
Epoch 19 / train_RMSE = 0.8684
19 is     done!
time spent :      24 s
Epoch 20 / train_RMSE = 0.8680
20 is     done!
time spent :      25 s
time:   519.2554092407227
```

(epoch 20)

대략 한 **epoch당 25초**의 시간이 걸린다. 마지막 실행 시간은 전처리 + 학습시간 + 결과 저장 까지의 전체 시간이다.

모든 rating의 예측값을 3으로 채웠을 때 u1에 대해 RMSE가 1.272046이 나와 이보다 RMSE가 큰 case는 추가적인 tuning없이 다른 case를 진행했다. 결측치에 관해 따로 안적힌 case는 모두 0으로 넣어준 case다. (case가 많아 일부만 작성했다)

- ➔ nf=200, reg_param=0.1, epochs=5: RMSE: 1.573775
- ➔ nf=200, reg_param=0.1, epochs=5(결측치 2로 채우고) RMSE: 1.831184
- ➔ nf=200, reg_param=0.1, epochs=10: RMSE: 1.531105
- ➔ nf=200, reg_param=0.01, epochs=10 RMSE: 1.654429
- ➔ nf=200, reg_param=0.1, epochs=15 RMSE: 1.514845
- ➔ nf=200, reg_param=0.1, epochs=30 RMSE: 1.24962
- ➔ nf=200, reg_param=0.1, epochs=5(결측치 1로 채우고) RMSE: 2.584356
- ➔ nf=200, reg_param=0.01, epochs=10(결측치 1로 채우고) RMSE: 2.591225
- ➔ nf=100, reg_param=0.1, epochs=5 RMSE: 1.696909
- ➔ nf=3, reg_param=0.3, epochs=20 (결측치 3으로 채우기) RMSE: 1.124127 => 그런데 값이 다 3가까이로 되버려 의미가 없는 결과라 생각했다.
- ➔ nf=200, reg_param=0.1, epochs=15(결측치 3으로 채우기) RMSE: 1.248581
- ➔ nf=5, reg_param=0.1, epochs=5 RMSE: 1.132693
- ➔ nf=3, reg_param=0.1, epochs=5 **RMSE: 1.093744**
- ➔ nf=3, reg_param=0.1, epochs=10 **RMSE: 1.082229**
- ➔ nf=3, reg_param=0.1, epochs=20 **RMSE: 1.08237**

따라서 nf 3, reg_param 0.1, epochs 5,10,20에서 u1~u5에 대한 test를 진행했다.

➔ u1

```
C:\Users\user\Desktop\4-1\데이터사이언스\Long term project_ Recommender System
(datascience) λ pPA4.exe u1
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 62
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 1.092208
```

(epoch5)

```
C:\Users\user\Desktop\4-1\데이터사이언스\Long term project_ Recommender System
(datascience) λ pPA4.exe u1
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 49
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 1.088008
```

(epoch10)

```
C:\Users\user\Desktop\4-1\데이터사이언스\Long term project_ Recommender System
(datascience) λ pPA4.exe u1
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 45
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 1.078695
```

(epoch20) (**best**)

➔ u2

```
C:\Users\user\Desktop\4-1\데이터사이언스\Long term project_ Recommender System
(datascience) λ pPA4.exe u2
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 25
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 1.039095
```

(epoch5) (**best**)

```
C:\Users\user\Desktop\4-1\데이터사이언스\Long term project_ Recommender System
(datascience) λ pPA4.exe u2
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 22
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 1.047939
```

(epoch10)

```
C:\Users\user\Desktop\4-1\데이터사이언스\Long term project_ Recommender System
(datascience) λ pPA4.exe u2
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 39
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 1.043411
```

(epoch20)

➔ u3

```
C:\Users\user\Desktop\4-1\데이터사이언스\Long term project_ Recommender System
(datascience) λ pPA4.exe u3
the number of ratings that didn't be predicted: 13
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 8
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 0.988569
```

(epoch5)

```
C:\Users\user\Desktop\4-1\데이터사이언스\Long term project_ Recommender System
(datascience) λ pPA4.exe u3
the number of ratings that didn't be predicted: 13
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 17
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 0.98045
```

(epoch10)

```
C:\Users\user\Desktop\4-1\데이터사이언스\Long term project_ Recommender System
(datascience) λ pPA4.exe u3
the number of ratings that didn't be predicted: 13
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 26
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 0.9802119
```

(epoch20) (**best**)

➔ u4

```
C:\Users\user\Desktop\4-1\데이터사이언스\Long term project_ Recommender System
(datascience) λ pPA4.exe u4
the number of ratings that didn't be predicted: 10
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 18
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 0.9978479
```

(epoch5)

```
C:\Users\user\Desktop\4-1\데이터사이언스\Long term project_ Recommender System
(datascience) λ pPA4.exe u4
the number of ratings that didn't be predicted: 10
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 19
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 0.991329
```

(epoch10)

```
C:\Users\user\Desktop\4-1\데이터사이언스\Long term project_ Recommender System
(datascience) λ pPA4.exe u4
the number of ratings that didn't be predicted: 10
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 17
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 0.9893668
```

(epoch20) (**best**)

➔ u5

```
C:\Users\user\Desktop\4-1\데이터사이언스\Long term project_ Recommender System
(datascience) λ pPA4.exe u5
the number of ratings that didn't be predicted: 16
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 23
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 1.025008
```

(epoch5)

```
C:\Users\user\Desktop\4-1\데이터사이언스\Long term project_ Recommender System
(datascience) λ pPA4.exe u5
the number of ratings that didn't be predicted: 16
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 22
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 0.9979987
```

(epoch10) (**best**)

```
C:\Users\user\Desktop\4-1\데이터사이언스\Long term project_ Recommender System
(datascience) λ pPA4.exe u5
the number of ratings that didn't be predicted: 16
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 15
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 0.9998946
```

(epoch20)

-