baseline model

November 6, 2019

```
In [2]: import pandas as pd
        import numpy as np
        import random
        import matplotlib.pyplot as plt
        from surprise import Dataset
        from surprise import accuracy
        from surprise import Reader
        import time
        import sklearn
        from sklearn.model_selection import train_test_split
        from random import shuffle
        import seaborn as sns
        from surprise.prediction_algorithms.baseline_only import BaselineOnly
```

```
In [3]: train_ratings = pd.read_csv("train_data.csv")
        test_ratings = pd.read_csv("test_data.csv")
```

```
In [4]: print('train_minimum_rating',min(train_ratings['rating']))
        print('train_maximum_rating',max(train_ratings['rating']))
        print('test_minimum_rating',min(train_ratings['rating']))
        print('test_maximum_rating',max(train_ratings['rating']))
```

```
('train_minimum_rating', 0.5)
('train_maximum_rating', 5.0)
('test_minimum_rating', 0.5)
('test_maximum_rating', 5.0)
```

```
In [18]: def coverage(threshold1, threshold2,prediction):
             start_time=time.time()
             predictions = pd.DataFrame(prediction)
             pred = predictions.groupby('uid')
             df1= pred.apply(lambda x: x.sort_values(by=["est"],ascending=False))
             df2=df1.reset_index(drop=True)
             df3 = df2.groupby('uid').head(10)

             s1=df3[df3['r_ui'] > threshold1].groupby('uid')['r_ui'].count().reset_index()
             s2 = df3.pivot_table(index=['uid'],aggfunc='size').reset_index()
```

```
            s2.columns = ['uid','counts']

            df=pd.merge(s1, s2, on='uid')

            # #number of high true rating(larger than 4) devided by top N predictions
            df['rate']=df['r_ui']/df['counts']

            user_coverage=float(sum(df['rate']> threshold2))/df3['uid'].nunique()

            item=df3.groupby('iid').apply(lambda x: x.sort_values(by=["est"],ascending=False)

            s=item[item['r_ui'] > threshold1].groupby('iid')['r_ui'].count().reset_index()
            ss = item.pivot_table(index=['iid'],aggfunc='size').reset_index()
            ss.columns = ['iid','counts']
            dff=pd.merge(s, ss, on='iid')
            dff['rate']=dff['r_ui']/dff['counts']
            item_coverage=float(sum(dff['rate']> threshold2))/df3['iid'].nunique()

            catalog_coverage = float(df3['iid'].nunique())/predictions['iid'].nunique()
            end_time=time.time()
            duration=end_time-start_time
            return user_coverage, item_coverage, catalog_coverage,duration
```

```
In [5]: reader = Reader(rating_scale=(0.5, 5))
        ratings = Dataset.load_from_df(train_ratings, reader)
```

```
In [6]: raw_ratings = ratings.raw_ratings
        random.seed(42)
        shuffle(raw_ratings)
```

```
In [7]: ratings.raw_ratings = raw_ratings
```

```
In [8]: copy_ratings=ratings
```

```
In [11]: test_rating = Dataset.load_from_df(test_ratings, reader)
         test_raw_rating=test_rating.raw_ratings
```

Fit a baseline model

```
In [14]: print('Using SGD')
         bsl_options = {'method': 'sgd',
                        'learning_rate': .00005,
                        }
         algo = BaselineOnly(bsl_options=bsl_options)

         algo.fit(ratings.build_full_trainset())
```

```
Using SGD
Estimating biases using sgd...
```

```
Out[14]: <surprise.prediction_algorithms.baseline_only.BaselineOnly at 0x10978bbd0>
```

```
In [15]: test_set = ratings.construct_testset(test_raw_rating)
```

```
In [16]: test_result = algo.test(test_set)
```

RMSE and MAE on test set using baseline model

```
In [17]: test_rmse = accuracy.rmse(test_result)
         test_mae = accuracy.mae(test_result)
```

```
RMSE: 0.9691
MAE:  0.7611
```

user-coverage,item-coverage, catalog, running time on the test set

```
In [19]: coverage(4.0, 0.5,test_result)
```

```
Out[19]: (0.17297840281265695,
          0.11483253588516747,
          0.8365577051367579,
          21.256478786468506)
```