IT-562 : Recommendation Engine and Applications

PERFORMANCE REPORT

Collaborative Filtering:
Estimating SVD through Stochastic Gradient Descent

Team Name : Tune-In
Members:
201501021 - Roshan Shah
201501113 - Shrey Shah
201501243 - Kajal Gosaliya
201501244 - Rutul Patel

In this assignment, we have used SVD via Stochastic Gradient Descent using surprise library. There are 2 methods in our approach , MatrixFacto and inbuilt SVD method. The accuracy is calculated through RMSE (Root-Mean-Square Error).

Compute RMSE (Root Mean Squared Error).

$$\text{RMSE} = \sqrt{\frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2}.$$

Other parameters for performance include training dat fit time, and training data test time, and Memory occupied in RAM terms.

Dataset is of the form of tuples as:
(UserId, SongId, Rating)

We divide the data into 5 folds.To use different folds for training and testing.

We have 4 datasets of different sizes:
1k data points
10k data points
100k data points
1 million points

Comparison on based of the parameters is done.

Code Snippet

Matrix Factorization:

```
import numpy as np
import pandas as pd
import surprise
from surprise import Reader
from surprise import Dataset
from surprise.model_selection import cross_validate
from guppy import hpy

class MatrixFacto(surprise.AlgoBase):
    '''A basic rating prediction algorithm based on matrix factorization.'''
    skip_train=0
    def __init__(self, learning_rate, n_epochs, n_factors):

        self.lr = learning_rate  # learning rate for SGD
        self.n_epochs = n_epochs  # number of iterations of SGD
        self.n_factors = n_factors  # number of factors

    def train(self, trainset):
        '''Learn the vectors p_u and q_i with SGD'''

        print('Fitting data with SGD...')

        # Randomly initialize the user and item factors.
        p = np.random.normal(0, .1, (trainset.n_users, self.n_factors))
        q = np.random.normal(0, .1, (trainset.n_items, self.n_factors))

        # SGD procedure
        for _ in range(self.n_epochs):
            for u, i, r_ui in trainset.all_ratings():
                err = r_ui - np.dot(p[u], q[i])
                # Update vectors p_u and q_i
                p[u] += self.lr * err * q[i]
                q[i] += self.lr * err * p[u]
                # Note: in the update of q_i, we should actually use the
previous (non-updated) value of p_u.
                # In practice it makes almost no difference.

        self.p, self.q = p, q
        self.trainset = trainset

    def estimate(self, u, i):
        '''Return the estmimated rating of user u for item i.'''

        # return scalar product between p_u and q_i if user and item are
known,
        # else return the average of all ratings
        if self.trainset.knows_user(u) and self.trainset.knows_item(i):
```

```
            return np.dot(self.p[u], self.q[i])
        else:
            return self.trainset.global_mean

if __name__ == '__main__':
    df = pd.read_csv("data_1k.csv")
    reader = Reader(rating_scale=(1, 5))
    data = Dataset.load_from_df(df[['user_id', 'song_id', 'rating']], reader)
    algo = MatrixFacto(learning_rate=0.01, n_epochs=10, n_factors=10)
    cross_validate(algo,data,measures=['RMSE'],cv=5,verbose=True)
    h=hpy()
    print (h.heap())
```

```
:\Users\Roshan Shah\PycharmProjects\MusicRecommendation>python MF_1k.py
itting data with SGD...
itting data with SGD...
itting data with SGD...
itting data with SGD...
itting data with SGD...
valuating RMSE of algorithm MatrixFacto on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
MSE (testset)     2.0323  1.7271  1.8863  1.8255  1.9400  1.8822  0.1031
it time           0.41    0.33    0.59    0.36    0.33    0.40    0.10
est time          0.02    0.00    0.02    0.02    0.02    0.01    0.01

:\Users\Roshan Shah\PycharmProjects\MusicRecommendation>python MF_10k.py
itting data with SGD...
itting data with SGD...
itting data with SGD...
itting data with SGD...
itting data with SGD...
valuating RMSE of algorithm MatrixFacto on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
MSE (testset)     1.8735  1.9365  1.8350  1.8644  1.9673  1.8953  0.0489
it time           2.80    2.94    2.81    3.70    1.44    2.74    0.73
est time          0.03    0.03    0.03    0.02    0.03    0.03    0.01

:\Users\Roshan Shah\PycharmProjects\MusicRecommendation>python MF_100k.py
itting data with SGD...
itting data with SGD...
itting data with SGD...
itting data with SGD...
itting data with SGD...
valuating RMSE of algorithm MatrixFacto on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
MSE (testset)     1.5546  1.5599  1.5490  1.5500  1.5560  1.5539  0.0040
it time           34.33   29.39   29.69   20.84   21.44   27.14   5.20
est time          0.23    0.81    0.38    0.34    0.52    0.46    0.20
```

```
Evaluating RMSE of algorithm MatrixFacto on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    2.4481  2.4461  2.4485  2.4453  2.4506  2.4477  0.0019
Fit time          248.28  277.92  255.31  279.25  254.62  263.08  12.90
Test time         4.34    4.00    3.97    3.78    5.61    4.34    0.66

C:\Users\Roshan Shah\PycharmProjects\MusicRecommendation>
```

SVD (Inbuilt):

```
import pandas as pd
```

```
from surprise import Reader
from surprise import SVD
from surprise import Dataset
from surprise.model_selection import cross_validate
from guppy import hpy

if __name__ == '__main__':
    df = pd.read_csv("data_1k.csv")
    reader = Reader(rating_scale=(1, 5))
    data = Dataset.load_from_df(df[['user_id', 'song_id', 'rating']], reader)
    algo = SVD()
    cross_validate(algo,data,measures=['RMSE'],cv=5,verbose=True)
    h = hpy()
    print h.heap()
```

```
C:\Users\Roshan Shah\PycharmProjects\MusicRecommendation>python SVD_1k.py
Evaluating RMSE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.4175  1.4732  1.4332  1.4054  1.3756  1.4210  0.0322
Fit time          0.30    0.20    0.20    0.34    0.17    0.24    0.07
Test time         0.00    0.00    0.00    0.00    0.00    0.00    0.00

C:\Users\Roshan Shah\PycharmProjects\MusicRecommendation>python SVD_10k.py
Evaluating RMSE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.4274  1.4457  1.4443  1.4354  1.4485  1.4403  0.0078
Fit time          1.86    2.61    1.77    2.03    1.27    1.91    0.43
Test time         0.05    0.03    0.05    0.03    0.03    0.04    0.01

C:\Users\Roshan Shah\PycharmProjects\MusicRecommendation>python SVD_100k.py
Evaluating RMSE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.4655  1.4692  1.4777  1.4761  1.4639  1.4705  0.0055
Fit time          12.06   12.36   17.59   17.72   11.84   14.32   2.73
Test time         0.42    0.39    0.39    0.27    0.72    0.44    0.15

C:\Users\Roshan Shah\PycharmProjects\MusicRecommendation>python SVD_1m.py
Evaluating RMSE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.5257  1.5236  1.5264  1.5222  1.5246  1.5245  0.0015
Fit time          158.26  161.84  151.59  170.21  159.77  160.33  6.02
Test time         4.14    8.42    4.06    6.28    3.84    5.35    1.77
```

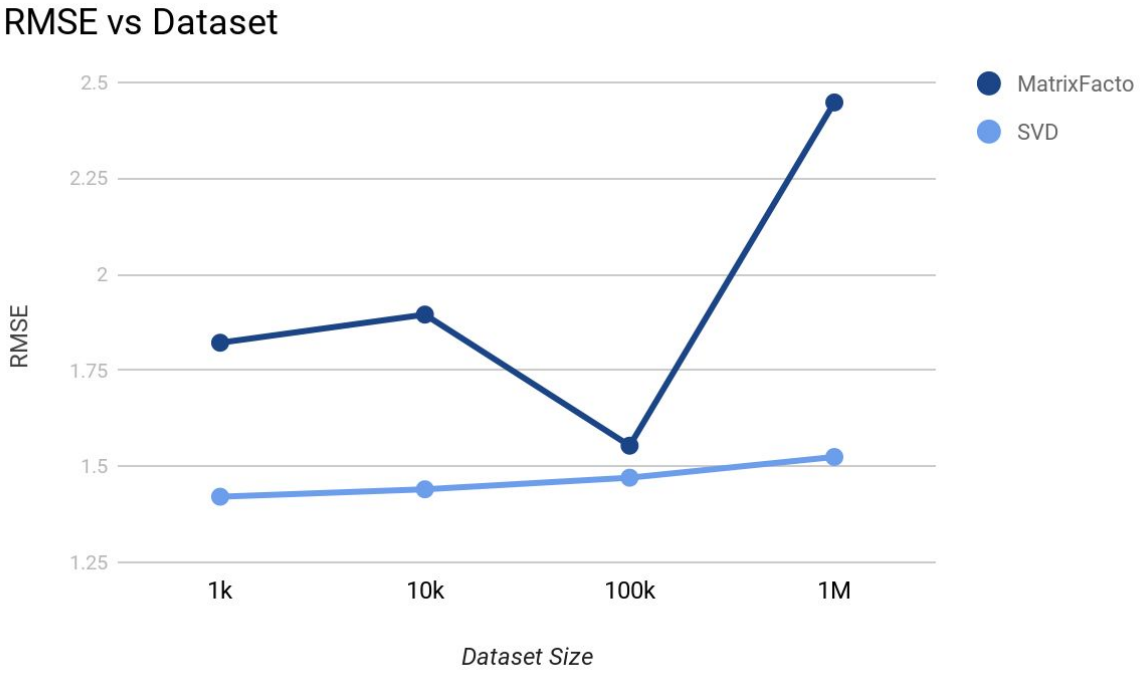
Performance Comparison

Matrix Facto:

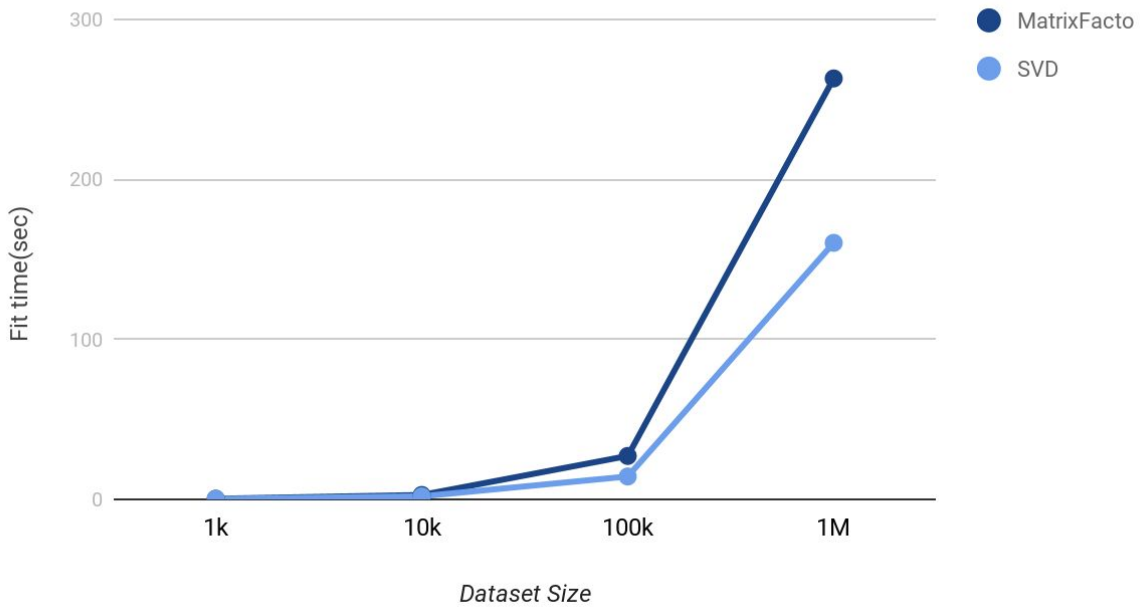
Parameters\Dataset	1k	10k	100k	1 Million
RMSE	1.822	1.8953	1.5539	2.4477
Fit time (sec)	0.40	2.74	27.14	263.08
Test time (sec)	0.01	0.03	0.46	4.34
Memory (MB)	19.049	19.008	27.256	109.704

SVD (Inbuilt):

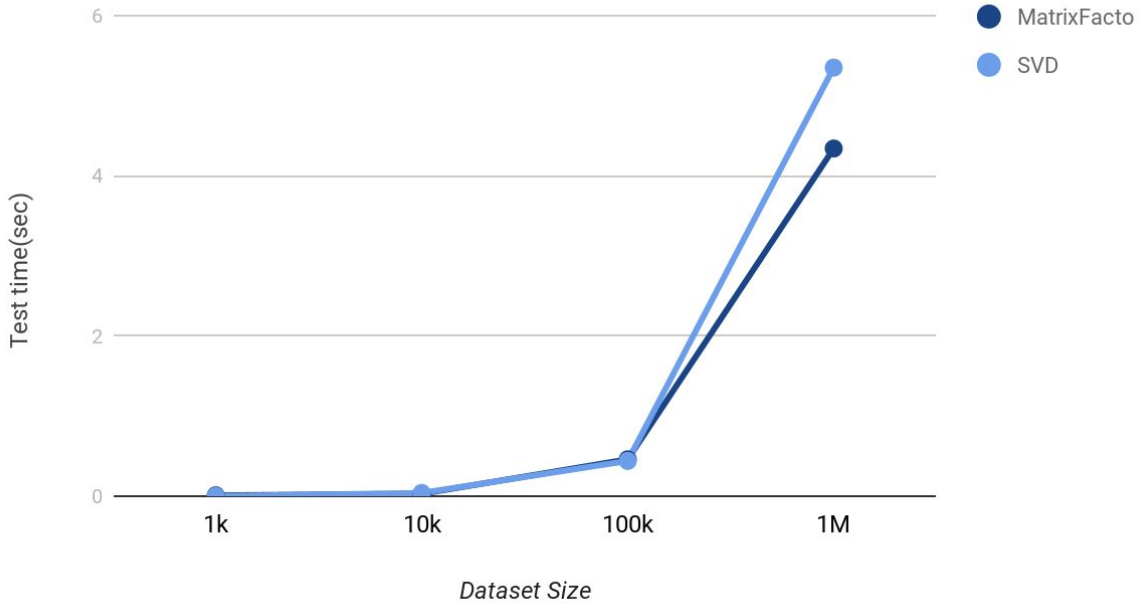
Parameters\Dataset	1k	10k	100k	1 Million
RMSE	1.4210	1.4403	1.4705	1.5245
Fit time (sec)	0.24	1.91	14.32	160.33
Test time (sec)	0.00	0.04	0.44	5.35
Memory (MB)	19.052	19.877	27.259	110.573

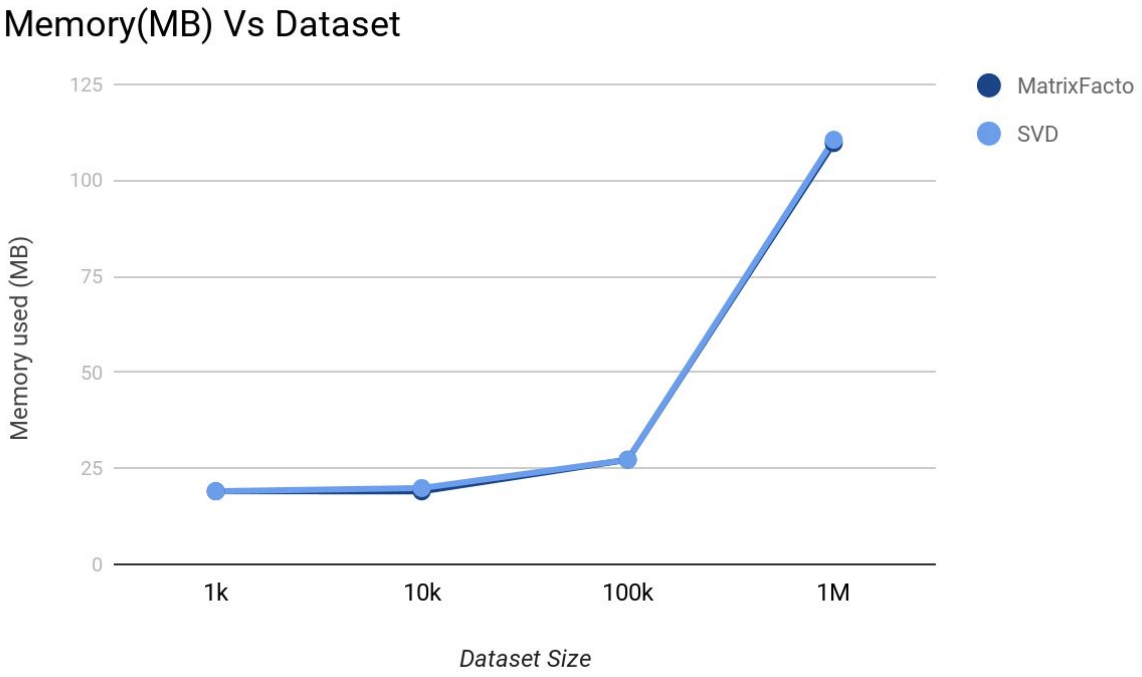


Fit time(sec) vs Dataset



Test time (sec) vs Dataset





From the above graphs , we can observe that as the dataset size increases, training time, test time, memory usage as well as RMSE increases.
