Report

IT-562 Recommendation Engine

Team-Kitkat

The algorithm we have used here to calculate the RMSE and MAE value is SVD. We have used the inbuilt implementation of SVD from surprise library. So here we have divided our data into 5 folds i.e. we have 5 parts of data out of which 4 will be used to train the dataset and the remaining ones will be used for testing and calculate RMSE and MAE value. Initially, we have calculated RMSE and MAE value for 1000 tuples then for 10000, 21633, 100000 and finally for 836006. The code snippet that we have used is as below.

**CODE:**

import os

import surprise

from guppy import hpy

from surprise import BaselineOnly

from surprise import Dataset

from surprise import Reader

from surprise import SVD

from surprise.model\_selection import cross\_validate

# path to dataset file

if \_\_name\_\_ == '\_\_main\_\_':

file\_path = os.path.expanduser('C:/Users/admin/Desktop/Data/music/ratings\_digital\_music(full).csv')

reader = Reader(line\_format='user item rating timestamp', sep=',')

data = Dataset.load\_from\_file(file\_path, reader=reader)

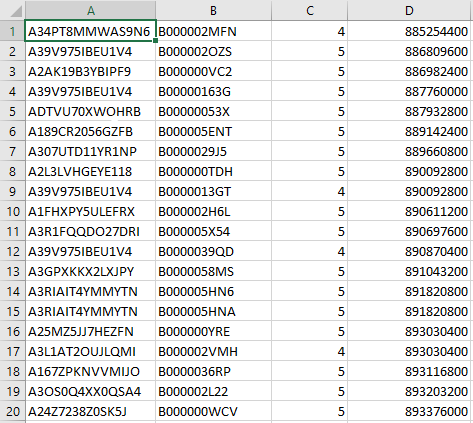
algo = SVD()

cross\_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)

h = hpy()

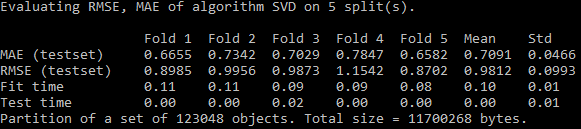
print h.heap()

In the above code, we have taken input from a CSV file. The dataset that we have used has user\_id, song\_id, ratings and timestamp as their columns.

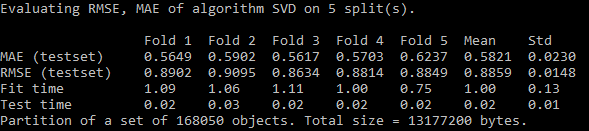


**Output:**

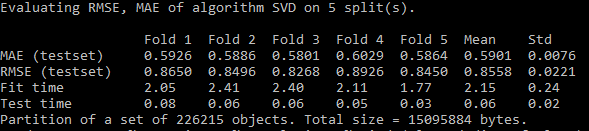
For 1000 tuples.



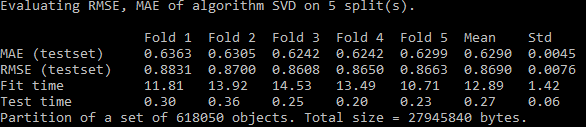
For 10000 Tuples.



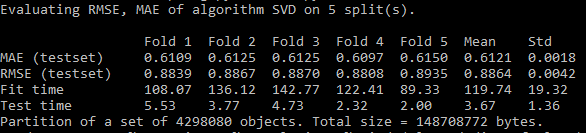
For 21633 tuples



For 100000 tuples



For 836006 tuples



In all the Graphs below 1(on x-axis) corresponds to 1000 tuples, Similarly

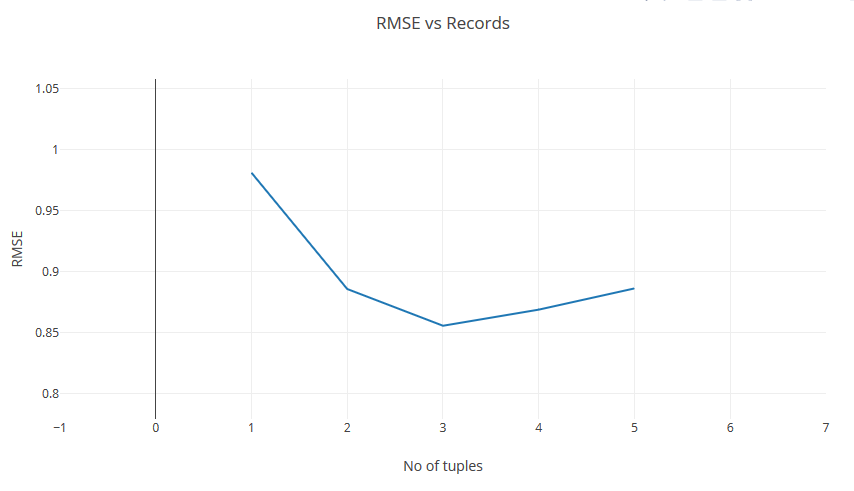
2- 10000

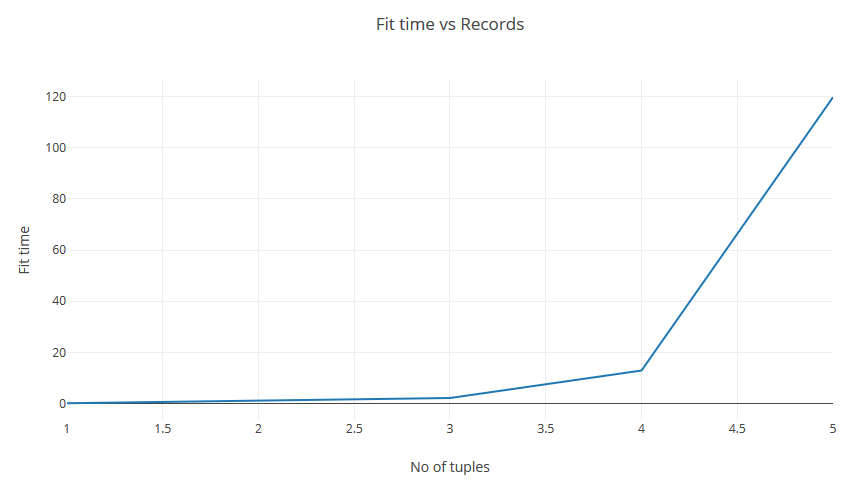
3 - 21633

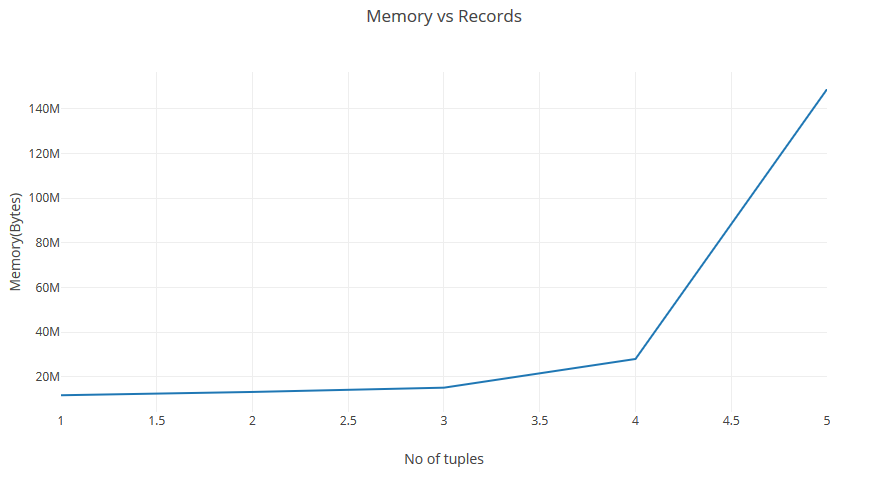
4 - 100000

5 - 836006

So, while plotting graphs we will take mean RMSE into consideration.







Thus from above results we can conclude that it is not necessary that with the increase in training data the error would decrease factors like number of folds, how the partition of data is done also plays a very important role.

**Continued on next page**

**Code using basic matrix factorization:**

import os

import surprise

import numpy as np

from guppy import hpy

from surprise import BaselineOnly

from surprise import Dataset

from surprise import Reader

from surprise import SVD

from surprise.model\_selection import cross\_validate

class MatrixFacto(surprise.AlgoBase):

skip\_train = 0

'''A basic rating prediction algorithm based on matrix factorization.'''

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):

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

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]

self.p, self.q = p, q

self.trainset = trainset

def estimate(self, u, i):

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\_\_':

file\_path = os.path.expanduser('C:/Users/admin/Desktop/Data/music/ratings\_digital\_music(full).csv')

reader = Reader(line\_format='user item rating timestamp', sep=',')

data = Dataset.load\_from\_file(file\_path, reader=reader)

data.split(2) # split data for 2-folds cross validation

algo = MatrixFacto(learning\_rate=.01, n\_epochs=10, n\_factors=10)

surprise.evaluate(algo, data, measures=['RMSE'])

h = hpy()

print h.heap()

|  |  |  |
| --- | --- | --- |
| Number of tuples | Mean RMSE | Memory(bytes) |
| 1000 | 0.9990 | 11838096 |
| 10000 | 0.9689 | 13667552 |
| 21633 | 1.8696 | 17524672 |
| 100000 | 0.9282 | 32376128 |
| 836006 | 1.0987 | 233852548 |

**PS:-**Also, the latter code compiled with some deprecated warning and when it was compiled for the last dataset it specifically gave ‘**RuntimeWarning: overflow encountered in multiply**’

So it is not certain whether the code is suitable for large dataset or not.

Also seeing the RMSE values and comparing both the implementation it can be concluded that using inbuilt SVD()**[uses SGD**] is more preferable rather than implementing custom svd().