EE-627-A-HW8

Jiazhen Hong
10427454
5/5/2018

Part I:

1. For the provided data file "re_u.data", fix the "numIterations=20" and use different "rank" size, rank=5, 10, 20, 30 and test the MSE values.

Solution I:

Rank	MSE values
5	0.6149152887478032
10	0.4715729694187941
20	0.2921528974390666
30	0.18219919696543654

Part II:

2. For the fixed "rank=20", and use different numIterations=2, 5,10, 20, 30 and test the MSE values.

Solution II:

NumIterations	MSE values
2	0.485122545373162
5	0.33756685956979166
10	0.3076796572076697
20	0.28988589585387026
30	0.2862405185431791

Part III:

3. For the fixed "rank=20" and "numIterations=20", take different size of data. i.e., 2000, 5000, 10000, 20000, 50000, 100000

Solution III:

Size of data	MSE values
2000	0.29010457580821279081
5000	0.2906575130875049575
10000	0.29083349721840124
20000	0.292600761964985
50000	0.29219743426361544
100000	0.29182316687590737

Part IV:

For the above 3 different scenarios, from your observation, how MSE is changed related to the parameters? That is, which factor, the rank size, or the numIterations size, or the data size will change the MSE value more significantly?

Solution IV:

As above result shows, we know:
Firstly, when size of data and numIterations is stable, MSE values will decrease with rank size increasing.
Secondly, when size of data and rank size is stable, MSE values will decrease with numIterations increasing.
Thirdly, when numIterations and rank size is stable, MSE values is almost stalbe.

It is obvious shows that the MSE is principal influence by size of data and numIterations.

Code shows following:

```
import pyspark
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.classification import LogisticRegressionWithSGD
from pyspark.mllib.tree import DecisionTree
from pyspark.mllib.recommendation import ALS, MatrixFactorizationModel,
```

```
data=sc.textFile("re_u.data.txt")
```

```
print(data.count())
```

```
100000
```

```
ratings = data.map(lambda l: l.split(',')).map(lambda l: Rating(int(l[0
print(ratings.count())
```

```
100000
```

```
rank = 20
numIterations = 20
model = ALS.train(ratings, rank, numIterations)
```

```
testdata = ratings.map(lambda p: (p[0], p[1]))
predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r
ratesAndPreds = ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predic
```

```
MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
print("Mean Squared Error = " + str(MSE))
```

```
Mean Squared Error = 0.29212449655500833
```

```
import pyspark
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.classification import LogisticRegressionWithSGD
from pyspark.mllib.tree import DecisionTree
from pyspark.mllib.recommendation import ALS, MatrixFactorizationMode
```

```
data=sc.textFile("re_u.data.txt")
```

```
#print(data.count())
```

```
file=open('pData.txt','w')
data1=open('re_u.data.txt')
```

```
i=1
while(i<1000001):
    a = data1.readline()
    file.write(''.join(a))
    i = i + 1
data1.close()
file.close()
```

```
pData=sc.textFile('pData.txt')
```

```
ratings = pData.map(lambda l: l.split(',')).map(lambda l: Rating(int(
print(ratings.count())
```

```
100000
```

```
rank = 20
numIterations = 20
model = ALS.train(ratings, rank, numIterations)
```

```
testdata = ratings.map(lambda p: (p[0], p[1]))
predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]),
ratesAndPreds = ratings.map(lambda r: ((r[0], r[1]), r[2])).join(pred
```

```
MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
print("Mean Squared Error = " + str(MSE))
```

```
Mean Squared Error = 0.29182316687590737
```