Part 1:


Part 2:

1. fixed numIterations = 20
   a. rank = 5

In [9]:
```
rank = 5
numIterations = 20
model = ALS.train(train_ratings, rank, numIterations)
```

In [10]:
```
testdata = train_ratings.map(lambda p: (p[0], p[1]))
predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
ratesAndPreds = train_ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
print("Mean Squared Error = " + str(MSE))
```

```
Mean Squared Error = 0.6187260909958737
```

   b. rank = 10

In [11]:
```
rank = 10
numIterations = 20
model = ALS.train(train_ratings, rank, numIterations)
```

In [12]:
```
testdata = train_ratings.map(lambda p: (p[0], p[1]))
predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
ratesAndPreds = train_ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
print("Mean Squared Error = " + str(MSE))
```

```
Mean Squared Error = 0.472247470360593
```

   c. rank = 20

In [13]:
```
rank = 20
numIterations = 20
model = ALS.train(train_ratings, rank, numIterations)
```

In [14]:
```
testdata = train_ratings.map(lambda p: (p[0], p[1]))
predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
ratesAndPreds = train_ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
print("Mean Squared Error = " + str(MSE))
```

```
Mean Squared Error = 0.29089591114428714
```

   d. rank = 30

In [15]:
```
rank = 30
numIterations = 20
model = ALS.train(train_ratings, rank, numIterations)
```

In [16]:
```
testdata = train_ratings.map(lambda p: (p[0], p[1]))
predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
ratesAndPreds = train_ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
print("Mean Squared Error = " + str(MSE))
```

```
Mean Squared Error = 0.18314214236371854
```

As it shows, as rank increased, the MSE decreased.

2. fixed rank = 20
   a. numIterations = 2

In [3]:
```
rank = 20
numIterations = 2
model = ALS.train(train_ratings, rank, numIterations)
```

In [4]:
```
testdata = train_ratings.map(lambda p: (p[0], p[1]))
predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
ratesAndPreds = train_ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
print("Mean Squared Error = " + str(MSE))
```

```
Mean Squared Error = 0.5235247099214314
```

   b. numIterations = 5

In [7]:
```
rank = 20
numIterations = 5
model = ALS.train(train_ratings, rank, numIterations)
```

In [8]:
```
testdata = train_ratings.map(lambda p: (p[0], p[1]))
predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
ratesAndPreds = train_ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
print("Mean Squared Error = " + str(MSE))
```

```
Mean Squared Error = 0.337948246332415
```

   c. numIterations = 10

In [9]:
```
rank = 20
numIterations = 10
model = ALS.train(train_ratings, rank, numIterations)
```

In [10]:
```
testdata = train_ratings.map(lambda p: (p[0], p[1]))
predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
ratesAndPreds = train_ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
print("Mean Squared Error = " + str(MSE))
```

```
Mean Squared Error = 0.30763536590666024
```

   d. numIterations = 20

In [13]:
```
rank = 20
numIterations = 20
model = ALS.train(train_ratings, rank, numIterations)
```

In [14]:
```
testdata = train_ratings.map(lambda p: (p[0], p[1]))
predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
ratesAndPreds = train_ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
print("Mean Squared Error = " + str(MSE))
```

```
Mean Squared Error = 0.29089591114428714
```

   e. numIterations = 30
      because numIteration is too large, the function cannot run properly.

As we see, as numIteration increased, the MSE decreased.

3. With fixed rank = 20, numIterations = 30

In [1]:
```
from pyspark import SparkContext
from pyspark.mllib.recommendation import ALS, MatrixFactorizationModel, Rating
sc = SparkContext.getOrCreate( )
sc.setLogLevel("WARN")
train_data_pre = sc.textFile('re_u.data')
train_data = sc.parallelize(train_data_pre.take(2000))
train_ratings = train_data.map(lambda l: l.split(',')).map(lambda l: Rating(int(l[0]), int(l[1]), float(l[2])))
```

In [3]:
```
rank = 20
numIterations = 20
model = ALS.train(train_ratings, rank, numIterations)
```

In [4]:
```
testdata = train_ratings.map(lambda p: (p[0], p[1]))
predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
ratesAndPreds = train_ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
print("Mean Squared Error = " + str(MSE))
```

| Size of data | Mean Squared Error |
|---|---|
| 2000 | 0.291120758082147923 |

| | |
|---|---|
| 5000 | 0.291259682781938134 |
| 10000 | 0.29208892894787620 |
| 20000 | 0.291487303979932457 |
| 50000 | 0.291021837001804823 |
| 100000 | 0.291294923919096340 |

As data shows, the size of data won't affect MSE.