Parallelized ALS on Apache Spark

Note: to run the source file, open a terminal shell, change directory to the directory of ALSbySpark.py file, then run the following command on the shell.

```
pyspark ALSbySpark.py
```

Background Information:

- A **SparkContext** is the main entry point for Spark to function representing the connection to a Spark cluster. It can be used to create RDDs, accumulators, and broadcast variables on that cluster. Only one SparkContext may be active per JVM [1].
- In randomSplit, the random **seed** is any valid 32-bit integer. It is used to control the sequence generation of pseudo-random numbers. Every unique seed value results in the same sequence.
- The implementation of ALS in Spark MLlib has the following parameters:
  numBlocks is the number of blocks used to parallelize computation (set to -1 to auto-configure).
  rank is the number of latent factors in the model.
  iterations is the number of iterations to run.
  lambda specifies the regularization parameter in ALS.
  implicitPrefs specifies whether to use the explicit feedback ALS variant or one adapted for implicit feedback data.
  alpha is a parameter applicable to the implicit feedback variant of ALS that governs the baseline confidence in preference observations.

Code and Algorithm:

We have commented the code so that the algorithm is clear. The code here is based largely on the code by Dianes [2].

Part I: Loading and Parsing Data into Spark RDDs:

We work at first on the small data set, available on the MovieLens link by the name ml-latest-small.zip

```
import os
from pyspark import SparkContext, SparkConf

sc = SparkContext(appName="PythonRecommender")
# Parse small dataset into RDD
small_ratings_raw_data = sc.textFile('hdfs:/user/hadoop/ml-latest-
small/ratings.csv')
small_ratings_raw_data_header = small_ratings_raw_data.take(1)[0] # header

# filter out header
small_ratings_data = small_ratings_raw_data.filter(lambda line:
line!=small_ratings_raw_data_header)\
    .map(lambda line: line.split(",")).map(lambda tokens:
(tokens[0],tokens[1],tokens[2])).cache()
# repeat with movies file

small_movies_raw_data = sc.textFile('hdfs:/user/hadoop/ml-latest-small/movies.csv')
small_movies_raw_data_header = small_movies_raw_data.take(1)[0]

small_movies_data = small_movies_raw_data.filter(lambda line:
line!=small_movies_raw_data_header)\
    .map(lambda line: line.split(",")).map(lambda tokens:
(tokens[0],tokens[1])).cache()
```

Part II: Evaluating Model Parameters:

To choose our model parameters, we use trial and error on a set of estimated ranks [4, 8, 12]. For each of those ranks, we build a model and evaluate it using RMSE on the validation set. By the end of this step, we choose the rank with the smallest error and test our model with the chosen parameters on the testing set. We output the RMSE.

```
# Split ratings into training, validation, and testing sets
training_RDD, validation_RDD, test_RDD = small_ratings_data.randomSplit([6, 2, 2],
seed=0L)
validation_for_predict_RDD = validation_RDD.map(lambda x: (x[0], x[1]))
test_for_predict_RDD = test_RDD.map(lambda x: (x[0], x[1]))

# Training phase
from pyspark.mllib.recommendation import ALS
import math

seed = 5L
iterations = 10 # most ALS algorithms converge after 8-12 iterations
regularization_parameter = 0.1
ranks = [4, 8, 12] # array of rank values to evaluate
errors = [0, 0, 0]
err = 0
tolerance = 0.02

min_error = float('inf')
best_rank = -1
best_iteration = -1

from time import time
t0=time()
for rank in ranks:
    model = ALS.train(training_RDD, rank, seed=seed, iterations=iterations,
                      lambda_=regularization_parameter)
    predictions = model.predictAll(validation_for_predict_RDD).map(lambda r: ((r[0],
r[1]), r[2]))
    rates_and_preds = validation_RDD.map(lambda r: ((int(r[0]), int(r[1])),
float(r[2]))).join(predictions)
    error = math.sqrt(rates_and_preds.map(lambda r: (r[1][0] - r[1][1])**2).mean())
    errors[err] = error
    err += 1
    print 'For rank %s the RMSE is %s' % (rank, error)
    if error < min_error:
        min_error = error
        best_rank = rank

t1 = time() - t0

# Testing the selected model
model = ALS.train(training_RDD, best_rank, seed=seed, iterations=iterations,
                      lambda_=regularization_parameter)
predictions = model.predictAll(test_for_predict_RDD).map(lambda r: ((r[0], r[1]),
r[2]))
rates_and_preds = test_RDD.map(lambda r: ((int(r[0]), int(r[1])),
float(r[2]))).join(predictions)
error = math.sqrt(rates_and_preds.map(lambda r: (r[1][0] - r[1][1])**2).mean())

# 3 print statements
```

Part III: building and using the recommender

After choosing our parameters, we should now be ready to work on the complete dataset, named ml-20m.zip on the MovieLens repository. After running this code, however, we run out of memory as the data size is too huge for one machine. We thus continue the work on the small dataset that we used earlier.

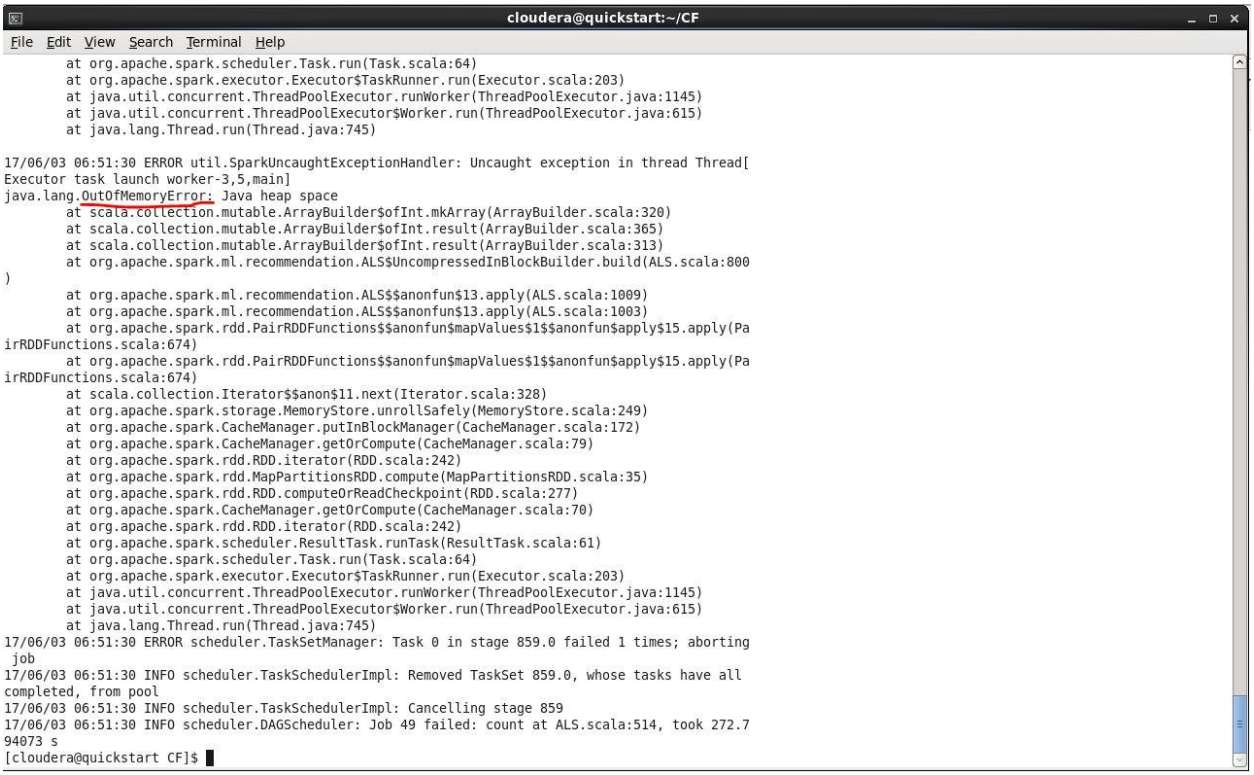
```
# Load the complete dataset file
complete_ratings_raw_data = sc.textFile('hdfs:/user/hadoop/ml-latest-
small/ratings.csv')
complete_ratings_raw_data_header = complete_ratings_raw_data.take(1)[0]

# Parse
complete_ratings_data = complete_ratings_raw_data.filter(lambda line:
line!=complete_ratings_raw_data_header)\
    .map(lambda line: line.split(",")).map(lambda tokens:
(int(tokens[0]),int(tokens[1]),float(tokens[2]))).cache()

training_RDD, test_RDD = complete_ratings_data.randomSplit([7, 3], seed=0L)

complete_model = ALS.train(training_RDD, best_rank, seed=seed,
                           iterations=iterations, lambda_=regularization_parameter)

test_for_predict_RDD = test_RDD.map(lambda x: (x[0], x[1]))

predictions = complete_model.predictAll(test_for_predict_RDD).map(lambda r: ((r[0],
r[1]), r[2]))
rates_and_preds = test_RDD.map(lambda r: ((int(r[0]), int(r[1])),
float(r[2]))).join(predictions)
error = math.sqrt(rates_and_preds.map(lambda r: (r[1][0] - r[1][1])**2).mean())

print 'For testing data the RMSE is %s' % (error)

# Making recommendations
complete_movies_raw_data = sc.textFile('hdfs:/user/hadoop/ml-latest-
small/movies.csv')
complete_movies_raw_data_header = complete_movies_raw_data.take(1)[0]

# Parse
complete_movies_data = complete_movies_raw_data.filter(lambda line:
line!=complete_movies_raw_data_header)\
    .map(lambda line: line.split(",")).map(lambda tokens:
(int(tokens[0]),tokens[1],tokens[2])).cache()

complete_movies_titles = complete_movies_data.map(lambda x: (int(x[0]),x[1]))

# 1 print statement

def get_counts_and_averages(ID_and_ratings_tuple):
    nratings = len(ID_and_ratings_tuple[1])
    return ID_and_ratings_tuple[0], (nratings, float(sum(x for x in
ID_and_ratings_tuple[1]))/nratings)

movie_ID_with_ratings_RDD = (complete_ratings_data.map(lambda x: (x[1],
x[2])).groupByKey())
movie_ID_with_avg_ratings_RDD =
movie_ID_with_ratings_RDD.map(get_counts_and_averages)
movie_rating_counts_RDD = movie_ID_with_avg_ratings_RDD.map(lambda x: (x[0],
x[1][0]))

#Adding new user ratings
new_user_ID = 0 # Assume userID is 0 because it is not used in the MovieLens dataset
```

```
# The format of each line is (userID, movieID, rating)
new_user_ratings = [
     (0,260,4), # Star Wars (1977)
     (0,1,3), # Toy Story (1995)
     (0,16,3), # Casino (1995)
     (0,25,4), # Leaving Las Vegas (1995)
     (0,32,4), # Twelve Monkeys (a.k.a. 12 Monkeys) (1995)
     (0,335,1), # Flintstones, The (1994)
     (0,379,1), # Timecop (1994)
     (0,296,3), # Pulp Fiction (1994)
     (0,858,5) , # Godfather, The (1972)
     (0,50,4) # Usual Suspects, The (1995)
    ]
new_user_ratings_RDD = sc.parallelize(new_user_ratings)
complete_data_with_new_ratings_RDD =
complete_ratings_data.union(new_user_ratings_RDD)

from time import time

t2 = time()
new_ratings_model = ALS.train(complete_data_with_new_ratings_RDD, best_rank,
seed=seed,
                              iterations=iterations,
lambda_=regularization_parameter)
t3 = time() - t2
# 1 print statement

# Getting top recommendations
new_user_ratings_ids = map(lambda x: x[1], new_user_ratings) # get just movie IDs
# keep those not on the ID list
new_user_unrated_movies_RDD = (complete_movies_data.filter(lambda x: x[0] not in
new_user_ratings_ids).map(lambda x: (new_user_ID, x[0])))

# Use the input RDD, new_user_unrated_movies_RDD, with
new_ratings_model.predictAll() to predict new ratings for the movies
new_user_recommendations_RDD =
new_ratings_model.predictAll(new_user_unrated_movies_RDD)

new_user_recommendations_rating_RDD = new_user_recommendations_RDD.map(lambda x:
(x.product, x.rating))
new_user_recommendations_rating_title_and_count_RDD = \

new_user_recommendations_rating_RDD.join(complete_movies_titles).join(movie_rating_c
ounts_RDD)
new_user_recommendations_rating_title_and_count_RDD.take(3)
# flat this down in order to have (Title, Rating, Ratings Count)
new_user_recommendations_rating_title_and_count_RDD = \
    new_user_recommendations_rating_title_and_count_RDD.map(lambda r: (r[1][0][1],
r[1][0][0], r[1][1]))
# get highest rated recommendations for new user, filtering out movies with less
than 25 ratings.
top_movies = new_user_recommendations_rating_title_and_count_RDD.filter(lambda r:
r[2]>=25).takeOrdered(25, key=lambda x: -x[1])

# 1 print statement

print 'The best model was trained with rank %s' % best_rank
print "Model selection took %s seconds" % round(t1,3)
print 'For testing data the RMSE is %s' % (error)

print "There are %s movies in the complete dataset" %
(complete_movies_titles.count())
```

```
print "New model trained in %s seconds" % round(t3,3)

print ('TOP recommended movies (with more than 25 reviews):\n%s' %
        '\n'.join(map(str, top_movies)))
```

Results:

Below is the error that resulted when we attempted to work on the complete dataset.



```
cloudera@quickstart:~/CF
File  Edit  View  Search  Terminal  Help
        at org.apache.spark.scheduler.Task.run(Task.scala:64)
        at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:203)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
        at java.lang.Thread.run(Thread.java:745)

17/06/03 06:51:30 ERROR util.SparkUncaughtExceptionHandler: Uncaught exception in thread Thread[
Executor task launch worker-3,5,main]
java.lang.OutOfMemoryError: Java heap space
        at scala.collection.mutable.ArrayBuilder$ofInt.mkArray(ArrayBuilder.scala:320)
        at scala.collection.mutable.ArrayBuilder$ofInt.result(ArrayBuilder.scala:365)
        at scala.collection.mutable.ArrayBuilder$ofInt.result(ArrayBuilder.scala:313)
        at org.apache.spark.ml.recommendation.ALS$UncompressedInBlockBuilder.build(ALS.scala:800
)
        at org.apache.spark.ml.recommendation.ALS$$anonfun$13.apply(ALS.scala:1009)
        at org.apache.spark.ml.recommendation.ALS$$anonfun$13.apply(ALS.scala:1003)
        at org.apache.spark.rdd.PairRDDFunctions$$anonfun$mapValues$1$$anonfun$apply$15.apply(Pa
irRDDFunctions.scala:674)
        at org.apache.spark.rdd.PairRDDFunctions$$anonfun$mapValues$1$$anonfun$apply$15.apply(Pa
irRDDFunctions.scala:674)
        at scala.collection.Iterator$$anon$11.next(Iterator.scala:328)
        at org.apache.spark.storage.MemoryStore.unrollSafely(MemoryStore.scala:249)
        at org.apache.spark.CacheManager.putInBlockManager(CacheManager.scala:172)
        at org.apache.spark.CacheManager.getOrCompute(CacheManager.scala:79)
        at org.apache.spark.rdd.RDD.iterator(RDD.scala:242)
        at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:35)
        at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:277)
        at org.apache.spark.CacheManager.getOrCompute(CacheManager.scala:70)
        at org.apache.spark.rdd.RDD.iterator(RDD.scala:242)
        at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:61)
        at org.apache.spark.scheduler.Task.run(Task.scala:64)
        at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:203)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
        at java.lang.Thread.run(Thread.java:745)
17/06/03 06:51:30 ERROR scheduler.TaskSetManager: Task 0 in stage 859.0 failed 1 times; aborting
 job
17/06/03 06:51:30 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 859.0, whose tasks have all
completed, from pool
17/06/03 06:51:30 INFO scheduler.TaskSchedulerImpl: Cancelling stage 859
17/06/03 06:51:30 INFO scheduler.DAGScheduler: Job 49 failed: count at ALS.scala:514, took 272.7
94073 s
[cloudera@quickstart CF]$
```

After working on the small dataset, below are the results we got from 10 iterations.

```
The best model was trained with rank 4
Model selection took 35.94 seconds
For testing data the RMSE is 0.9367179771
```

```
There are 9125 movies in the complete dataset
New model trained in 3.065 seconds
TOP recommended movies (with more than 25 reviews):
(u'Modern Times (1936)', 4.268370683587567, 32)
(u'All About Eve (1950)', 4.1706884648936313, 38)
(u'Ran (1985)', 4.1165700163034877, 26)
(u'Brokeback Mountain (2005)', 4.1126334784820049, 29)
(u'Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)', 4.0602242298053639, 39)
(u'"Amelie (Fabuleux destin d\'Am\xe9lie Poulain', 4.0536008861144301, 125)
(u'Cinema Paradiso (Nuovo cinema Paradiso) (1989)', 4.0508416908552594, 46)
(u'There Will Be Blood (2007)', 4.0497488592850335, 26)
(u'His Girl Friday (1940)', 4.0437585576730175, 26)
(u'"Godfather: Part II', 4.0326643555481905, 135)
(u'"Producers', 4.0247270429594995, 33)
(u'Bringing Up Baby (1938)', 4.0168114434401527, 30)
(u'Hoop Dreams (1994)', 4.0126942444854548, 61)
(u'Adaptation (2002)', 4.0096967225907321, 42)
(u'Dog Day Afternoon (1975)', 3.9987964229597304, 29)
(u'"Lives of Others', 3.9903235100656596, 37)
(u'"Third Man', 3.9885580974375157, 38)
(u'And Your Mother Too (Y tu mam\xe1 tambi\xe9n) (2001)', 3.9866808653725068, 26)
(u'Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)', 3.9863469271988543, 105)
(u'Chinatown (1974)', 3.9845113416458746, 76)
(u'Raging Bull (1980)', 3.9830222616675859, 50)
(u'Annie Hall (1977)', 3.9766657577933864, 80)
(u'On the Waterfront (1954)', 3.9673148581941096, 29)
(u'"Grapes of Wrath', 3.9661698674156045, 31)
(u'Rear Window (1954)', 3.9645839714755673, 92)
```

We also tried 12 iterations:

```
The best model was trained with rank 4
Model selection took 36.914 seconds
For testing data the RMSE is 0.935162742481
```

```
There are 9125 movies in the complete dataset
New model trained in 3.284 seconds
TOP recommended movies (with more than 25 reviews):
(u'Modern Times (1936)', 4.2935086603217449, 32)
(u'All About Eve (1950)', 4.1707397576995744, 38)
(u'Ran (1985)', 4.1408008979925466, 26)
(u'Cinema Paradiso (Nuovo cinema Paradiso) (1989)', 4.1000152514152424, 46)
(u'Brokeback Mountain (2005)', 4.0991349869959839, 29)
(u'Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)', 4.0809584315379546, 39)
(u'"Amelie (Fabuleux destin d\'Am\xe9lie Poulain', 4.0787828162007074, 125)
(u'There Will Be Blood (2007)', 4.0496231899278445, 26)
(u'"Lives of Others', 4.0414547754827481, 37)
(u'"Godfather: Part II', 4.0280455499852668, 135)
(u'His Girl Friday (1940)', 4.0262622612906735, 26)
(u'"Producers', 4.0182730010449923, 33)
(u'Bringing Up Baby (1938)', 4.0172495576667036, 30)
(u'"Third Man', 4.0024150850946985, 38)
(u'Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)', 3.995944782933412
5, 105)
(u'Dog Day Afternoon (1975)', 3.9937910381051087, 29)
(u'Adaptation (2002)', 3.9912081169295672, 42)
(u'Annie Hall (1977)', 3.9904001756343797, 80)
(u'Hoop Dreams (1994)', 3.986711607032476, 61)
(u'Chinatown (1974)', 3.9843605746652671, 76)
(u'On the Waterfront (1954)', 3.9805074713340467, 29)
(u'And Your Mother Too (Y tu mam\xe1 tambi\xe9n) (2001)', 3.9803660790555266, 26)
(u'Rear Window (1954)', 3.9730867895707678, 92)
(u'Raging Bull (1980)', 3.972831421313292, 50)
(u'Casablanca (1942)', 3.9664849408199645, 117)
```

With 14 iterations, we obtained the following results:

```
The best model was trained with rank 4
Model selection took 36.754 seconds
For testing data the RMSE is 0.934249157748
```

```
There are 9125 movies in the complete dataset
New model trained in 4.96 seconds
TOP recommended movies (with more than 25 reviews):
(u'Modern Times (1936)', 4.2883052750561594, 32)
(u'All About Eve (1950)', 4.1684996453592129, 38)
(u'Ran (1985)', 4.1498545133796512, 26)
(u'Cinema Paradiso (Nuovo cinema Paradiso) (1989)', 4.1250677967874898, 46)
(u'Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)', 4.0957679577857604, 39)
(u'"Amelie (Fabuleux destin d\'Am\xe9lie Poulain', 4.083596732997302, 125)
(u'Brokeback Mountain (2005)', 4.0780203437037681, 29)
(u'"Lives of Others', 4.0724156189711067, 37)
(u'There Will Be Blood (2007)', 4.059619573519444, 26)
(u'Bringing Up Baby (1938)', 4.0226707057564148, 30)
(u'"Godfather: Part II', 4.0176624489867434, 135)
(u'"Producers', 4.0147672829471581, 33)
(u'"Third Man', 4.01171071262808, 38)
(u'His Girl Friday (1940)', 4.0111395881579188, 26)
(u'Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)', 4.0085414617474502, 105)
(u'Annie Hall (1977)', 4.0024468176286732, 80)
(u'On the Waterfront (1954)', 3.9920714181908998, 29)
(u'Chinatown (1974)', 3.9884509191494031, 76)
(u'Dog Day Afternoon (1975)', 3.9858205335504682, 29)
(u'And Your Mother Too (Y tu mam\xe1 tambi\xe9n) (2001)', 3.9805262869476472, 26)
(u'Rear Window (1954)', 3.9790381352639175, 92)
(u'Adaptation (2002)', 3.9775982018823846, 42)
(u'Casablanca (1942)', 3.9682525492155438, 117)
(u'Hoop Dreams (1994)', 3.9608692642878673, 61)
(u'Raging Bull (1980)', 3.9585650592931096, 50)
```

With 16 iterations

```
The best model was trained with rank 4
Model selection took 35.593 seconds
For testing data the RMSE is 0.933858465136
```

```
There are 9125 movies in the complete dataset
New model trained in 3.792 seconds
TOP recommended movies (with more than 25 reviews):
(u'Modern Times (1936)', 4.2776469906376091, 32)
(u'All About Eve (1950)', 4.1653620473373749, 38)
(u'Ran (1985)', 4.1511343791257271, 26)
(u'Cinema Paradiso (Nuovo cinema Paradiso) (1989)', 4.1351434873960429, 46)
(u'Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)', 4.1057228391569138, 39)
(u'"Lives of Others', 4.0897362138016149, 37)
(u'"Amelie (Fabuleux destin d\'Am\xe9lie Poulain', 4.0831622515608021, 125)
(u'There Will Be Blood (2007)', 4.0706326121783407, 26)
(u'Brokeback Mountain (2005)', 4.0527941842666166, 29)
(u'Bringing Up Baby (1938)', 4.0277173106558815, 30)
(u'"Third Man', 4.0204239205761318, 38)
(u'Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)', 4.017892536601277, 105)
(u'"Producers', 4.0163569802397419, 33)
(u'Annie Hall (1977)', 4.0130612045664424, 80)
(u'"Godfather: Part II', 4.0060811674462737, 135)
(u'On the Waterfront (1954)', 4.0012918768943511, 29)
(u'His Girl Friday (1940)', 3.9983095153797934, 26)
(u'Chinatown (1974)', 3.9923281362983647, 76)
(u'Rear Window (1954)', 3.9828484399107995, 92)
(u'And Your Mother Too (Y tu mam\xe1 tambi\xe9n) (2001)', 3.9825726215355433, 26)
(u'Dog Day Afternoon (1975)', 3.9801163658587582, 29)
(u'Adaptation (2002)', 3.9701616614820341, 42)
(u'Casablanca (1942)', 3.9686783731547717, 117)
(u'Vertigo (1958)', 3.9610872601032097, 69)
(u'Network (1976)', 3.9580159490205915, 38)
```

For 18 iterations, we obtained the results below.

```
The best model was trained with rank 4
Model selection took 37.051 seconds
For testing data the RMSE is 0.933713410043
```

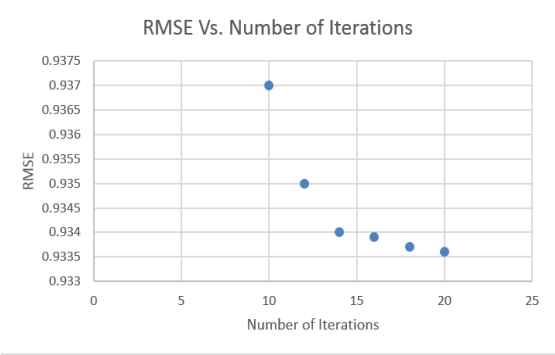
```
There are 9125 movies in the complete dataset
New model trained in 4.101 seconds
TOP recommended movies (with more than 25 reviews):
(u'Modern Times (1936)', 4.2679496525413336, 32)
(u'All About Eve (1950)', 4.1615660688846754, 38)
(u'Ran (1985)', 4.1486841635674176, 26)
(u'Cinema Paradiso (Nuovo cinema Paradiso) (1989)', 4.1359315085378254, 46)
(u'Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)', 4.1124288394264576, 39)
(u'"Lives of Others', 4.099716263944714, 37)
(u'"Amelie (Fabuleux destin d\'Am\xe9lie Poulain', 4.0815948220192029, 125)
(u'There Will Be Blood (2007)', 4.0805629678306614, 26)
(u'Bringing Up Baby (1938)', 4.0303072831025224, 30)
(u'"Third Man', 4.0298761817417468, 38)
(u'Brokeback Mountain (2005)', 4.0272470220072432, 29)
(u'Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)', 4.024212447707848, 105)
(u'Annie Hall (1977)', 4.0221765353837604, 80)
(u'"Producers', 4.0190563245121034, 33)
(u'On the Waterfront (1954)', 4.007728134703668, 29)
(u'Chinatown (1974)', 3.9957784082593615, 76)
(u'"Godfather: Part II', 3.994315921176895, 135)
(u'His Girl Friday (1940)', 3.9874902819814881, 26)
(u'And Your Mother Too (Y tu mam\xe1 tambi\xe9n) (2001)', 3.9850015469962767, 26)
(u'Rear Window (1954)', 3.9845060292758148, 92)
(u'Dog Day Afternoon (1975)', 3.9752173477848638, 29)
(u'Vertigo (1958)', 3.9726858708290953, 69)
(u'Casablanca (1942)', 3.9676545949435553, 117)
(u'Adaptation (2002)', 3.967237417408823, 42)
(u'Network (1976)', 3.9654763680514016, 38)
```

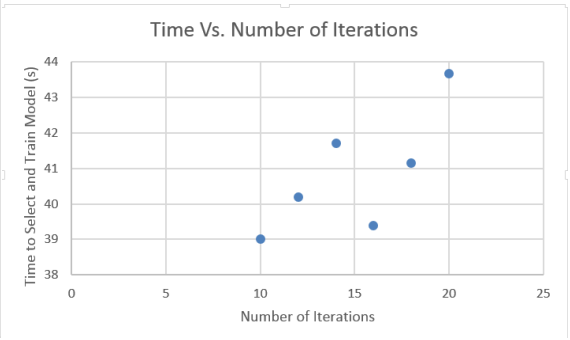
For 20 iterations, we obtained the results below.

```
The best model was trained with rank 4
Model selection took 39.155 seconds
For testing data the RMSE is 0.933614079342
```

```
There are 9125 movies in the complete dataset
New model trained in 4.525 seconds
TOP recommended movies (with more than 25 reviews):
(u'Modern Times (1936)', 4.2584815160841778, 32)
(u'All About Eve (1950)', 4.1564556494377385, 38)
(u'Ran (1985)', 4.1438137728856397, 26)
(u'Cinema Paradiso (Nuovo cinema Paradiso) (1989)', 4.1305334312733439, 46)
(u'Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)', 4.1168880819536362, 39)
(u'"Lives of Others', 4.1068597129874167, 37)
(u'There Will Be Blood (2007)', 4.0897963200817671, 26)
(u'"Amelie (Fabuleux destin d\'Am\xe9lie Poulain', 4.0798477425419239, 125)
(u'"Third Man', 4.0389972711688005, 38)
(u'Annie Hall (1977)', 4.0295710146315908, 80)
(u'Bringing Up Baby (1938)', 4.0290699614048275, 30)
(u'Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)', 4.0279367771749133, 105)
(u'"Producers', 4.0208681691496384, 33)
(u'On the Waterfront (1954)', 4.0111707895569593, 29)
(u'Brokeback Mountain (2005)', 4.0013017760388685, 29)
(u'Chinatown (1974)', 3.9980962405449154, 76)
(u'And Your Mother Too (Y tu mam\xe1 tambi\xe9n) (2001)', 3.9872516352364116, 26)
(u'Rear Window (1954)', 3.9841172316337792, 92)
(u'"Godfather: Part II', 3.9828332054294662, 135)
(u'Vertigo (1958)', 3.9810389844176665, 69)
(u'His Girl Friday (1940)', 3.979056311258145, 26)
(u'Cool Hand Luke (1967)', 3.9766515518882577, 46)
(u'Network (1976)', 3.9721733209880181, 38)
(u'Dog Day Afternoon (1975)', 3.9682427121080015, 29)
(u'Adaptation (2002)', 3.966520325017588, 42)
```

Below are graphs showing the effect of the number of iterations on the RMSE and the running time of the code.





Conclusion:

With increasing number of iterations in the ALS algorithm, the RMSE decreases approaching 0. However, although the running time shows an unexpected trend, we believe this is because the machine could have been unequally loaded with other tasks during operation of the code.

References

[1] "SparkContext (Spark 2.0.2 JavaDoc)", Spark.apache.org. [Online]. Available: https://spark.apache.org/docs/2.0.2/api/java/org/apache/spark/SparkContext.html. [Accessed: 01- Jun- 2017].

[2] J. Dianes, "Building a Movie Recommendation Service with Apache Spark & Flask - Part 1 | Codementor", Codementor.io, 2015. [Online]. Available: https://www.codementor.io/jadianes/building-a-recommender-with-apache-spark-python-example-app-part1-du1083qbw. [Accessed: 01- Jun- 2017].