Movie Recommendation Two Ways

ALS and K-Means Predictions in PySpark

Run Programs

1. Store u.data and u.item on Hadoop Distributed File System in a folder called input.
2. Execute Alternating Least Squares recommendation using :
   spark-submit --driver-memory 512m MovieLensALS.py hdfs://localhost:9000/input
3. Execute KMeans recommendation using:
   spark-submit --driver-memory 512m MovieLensKM.py hdfs://localhost:9000/input

Alternating Least Squares

In this program, we use ALS to perform collaborative filtering. We want to break a matrix of user ratings up into a smaller set of factors and this is accomplished by optimizing a matrix factorization problem. ALS has the advantage of being easy to parallelize which makes it a great candidate for Spark.

ALS is trained using three parameters, including rank, iterations, and lambda. Rank is the number of features or latent factors used by the model. Iterations is the number of iterations of ALS to run so that the algorithm converges to a reasonable solution. Lambda specifies the regularization parameter.

The program includes a loop to iterate through several options for the available parameters. We test rank of 8 and 12, lambdas of 0.1 and 10, and number of iterations of 10 and 20. By calculating the RMSE for every combination of parameters, we can determine the optimal model uses 8 latent factors, a low regularization parameter of 0.1, and 20 iterations. By reviewing the RMSEs of each model, we see that the regularization parameter of 0.1 always yields better results than a high parameter of 10, but the number of iterations and rank both have minimal impact on model performance.

Below is the output of the program:

```
Got 100000 ratings from 943 users on 1682 movies.
Ratings data was partitioned into training: 60024, validation: 20435, test: 19541.
RMSE  = 0.952735 for the model trained with rank = 8, lambda = 0.1, and numIter = 10.
RMSE  = 0.945888 for the model trained with rank = 8, lambda = 0.1, and numIter = 20.
RMSE  = 3.701949 for the model trained with rank = 8, lambda = 10.0, and numIter = 10.
RMSE  = 3.701949 for the model trained with rank = 8, lambda = 10.0, and numIter = 20.
RMSE  = 0.953673 for the model trained with rank = 12, lambda = 0.1, and numIter = 10.
RMSE  = 0.953495 for the model trained with rank = 12, lambda = 0.1, and numIter = 20.
RMSE  = 3.701949 for the model trained with rank = 12, lambda = 10.0, and numIter = 10.
RMSE  = 3.701949 for the model trained with rank = 12, lambda = 10.0, and numIter = 20.
The best model was trained with rank = 8 and lambda = 0.1, and numIter = 20, and its RMSE on
the test set is 0.953738.
The best model improves the baseline by 15.49%.
```

K-Means Clustering

K-Means clustering is a form of unsupervised learning that assigns data points to groups by optimizing the total distance between each point and the center of a cluster. In this program, we use K-Means to assign every movie in the dataset to a cluster based on it's genre vector. We can then predict a rating for a user for a movie based on the user's average rating for other movies in that cluster.

K-Means is trained using a given number of clusters. We can make this number a parameter in order to determine the optimal number of clusters. By calculating the within cluster sum of squares for each cluster configuration, we see that the best model has 174 clusters.

Below is the output of the program:

Got 100000 ratings from 943 users on 1682 movies for 19 genres.
Movies data was partitioned into training: 1010, validation: 336, test: 336.
SSE = 336 for the model with 2 clusters
SSE = 263 for the model with 3 clusters
SSE = 253 for the model with 4 clusters
SSE = 222 for the model with 5 clusters
SSE = 210 for the model with 6 clusters
SSE = 223 for the model with 7 clusters
SSE = 213 for the model with 8 clusters
SSE = 194 for the model with 9 clusters
SSE = 212 for the model with 10 clusters
…
SSE = 52 for the model with 100 clusters
SSE = 40 for the model with 125 clusters
SSE = 33 for the model with 150 clusters
SSE = 29 for the model with 174 clusters
The best model has 174 clusters, its training SSE is 29 and its SSE on the test set is 26.
The test RMSE is 0.751470 and the baseline RMSE is 0.838106.
The best model improves the baseline by 10.34%.

Comparing Models

We evaluate the performance of both models based on the Root Mean Square Error, which compares the predicted ratings to the actual ratings. We created a naïve baseline for measuring performance where the predicted rating is simply the average rating for all movies. K-Means has 10.34% improvement over the baseline while ALS has 15.49% improvement. ALS is the better performing model.