Data 612 - Project 5

Paul Perez

7/7/2020

Contents

The Recommender System's

This recommender system will use an Alternating Least Squares matrix factorization model from both the `recommenderlab` and `sparklyr` packages. We'll load the MovieLense dataset from the `recommenderlab` package, which is where our first model will be built in. We'll also load a transformed version of the dataset into Spark. We can see that using the `class()` function, that this MovieLense dataset is a `realRatingMatrix`.

RecommenderLab Model

```
rl_start <- Sys.time()

# Access Data
data("MovieLense")
rl_matrix <- as(MovieLense, "realRatingMatrix")

# Select Relevant Data
movie_ratings <- rl_matrix

# Split into Training & Test datasets using an 80/20 Ratio
set.seed(123)
eval_sets <- evaluationScheme(data = movie_ratings, method = "split", train = 0.8, given = 5, goodRatin

# RecommenderLab ALS Model
rl_als_model <- Recommender(getData(eval_sets, "train"), method = "ALS")

# Predict the ALS Recommender System on the Test set
rl_als_predict <- predict(object =  rl_als_model, newdata =  getData(eval_sets, "known"), n = 5, type =

# Evaluate
rl_evaluation <- calcPredictionAccuracy(x = rl_als_predict, data = getData(eval_sets, "unknown"))

rl_end <- Sys.time()
```

Spark Model

```
sp_start <- Sys.time()

# Connect to Spark
sc <- spark_connect(master = "local")

# Access Data - Transform into Spark Required Input
sp_data <- MovieLense %>%
  as(. , "data.frame") %>%
  mutate(user = as.numeric(user),
         item = as.numeric(item))

# Copy Mutated Matrix to Spark
sp_matrix <- copy_to(sc, sp_data, "sp_matrix", overwrite = TRUE)

# Split into Training & Test datasets using 80/20 Ratio
partitions <- sp_matrix %>%
  sdf_random_split(training = 0.8, testing = 0.2, seed = 123)

sp_train <- partitions$training
sp_test <- partitions$test

# Spark ALS Model
sp_als_model <- ml_als(sp_train, rating_col = "rating", user_col = "user", item_col = "item", max_iter =

# Predict the ALS Recommender System on the Test set
#sp_als_predict <- ml_predict(sp_als_model, sp_test)
sp_als_predict <- sp_als_model$.jobj %>%
  invoke("transform", spark_dataframe(sp_test)) %>%
  collect()

sp_end <- Sys.time()

# Disconnect from Spark
spark_disconnect(sc)
```

Table 1: Run Time Comparison

recommenderlab	sparklyr
41.8029	21.2591

Evaluation of Peformance

To evaluate the performance of the model creation on both the `recommenderlab` package and the `sparklyr` package, we can calculate the time it took for each model to run.

```
recommenderlab <- unclass(rl_end - rl_start)[1]
sparklyr <- unclass(sp_end - sp_start)[1]
time_df <- data.frame(recommenderlab, sparklyr)
time_df %>% kable(caption = "Run Time Comparison", align = 'c') %>% kable_styling("striped", full_width
```

Summary

While both recommender systems ran relatively quick due to the size of the dataset, `sparklyr` ran just short of half the time it took `recommenderlab` to run. As the size of this data and complexity of these recommendation systems increases, it makes sense to run on Apache Spark.This is an open-source distributed general-purpose cluster computing framework according to wikipedia, and I am using this verison of spark on my local device. I am sure the processing speed of the recommender systems would increase if I ran it on databricks or a stronger computer. Since I adapted an earlier recommender system built in R to run with `sparklyr`, I was limited to building out an ALS model, which I could compare to the `recommenderlab` model. If I decide to create another recommender system, I would utilize `PySpark` for programming it in Python.

Source Code GitHub Repository