Final Project Paper, DS-GA-1004, Spring 2019

Anshul Garg

Ag6516@nyu.edu

INTRODUCTION

Starting with the project, I built a recommender system, which recommends audio tracks to users based on their history. I prepared datasets, trained ALS (alternating least square) model on a hyperparameter grid, and cross-validated them. I also evaluated a baseline model, as well as applied alternative model solutions using ALS methods on my original model.

To reduce the computation time on Dumbo cluster, I reduced the training set to keep only the users which are present in validation or test files. Therefore, I only worked with 110,000 users. I created 1000 partitions for the same. This is potentially problematic as the number of available labels (i.e. number of tracks listened) for these users have been split among train, validation and test files. This in result reduced the data available for training the model. But, I had to necessarily train on this set of users for better evaluation. I fit the indexer to the training set (either the full or subsampled one, depending on which was used, as both contain all the validation and test users as well, and I skipped new tracks as those are not predictable enough).

IMPLEMENTATION

The following steps explain how I built and evaluated the model:

Step1. **Indexing:** I gave numeric indices to the alphanumeric strings for the user_id and item_ids in the original files (cf_train, cf_validation and cf_test). I have implemented Pipeline which further has String Indexer inside it. This helped to reduce memory requirement of model training and evaluation.

Step2. **Model training:** I used Spark implementation of ALS algorithm for training a bi-linear latent factor model (pyspark.ml.recommendation.ALS class).

Step3. **Hyper-paramter search:** Hyper-parameter settings that were considered are:
• rank (dimension) of the latent factors: (4, 8, 16),
• regularization parameter: (0, 0.1, 1, 10), and
• alpha (scaling parameter for count data): (0.1, 0.5, 1.0)

For the rest of hyper-parameters, the default values (in Spark) were used. I used root mean squared error (RMSE) as the evaluation criterion to choose among the different hyper-parameters and thereby, choose the best model. This was used over ranking-based metrics to reduce computation time.

Step4. **Evaluation on test:** For evaluating the trained model on the test set, I used Regression Evaluator on the validation dataset to calculate the RMSE. I performed more detailed analysis in the Extension used.

The final parameters of my ALS model are :
1. maxIter = 3
2. regParam= 1
3. userCol= "user_num"
4. itemCol= "track_num"
5. ratingCol ="count"
6. implicitPrefs=True
7. coldStartStrategy="drop"
8. alpha = 0.5
9. rank = 4

EXTENSION

- I have implemented the Alternative model formulations extension. I have implemented Log Compression on count value, as I read that it stands out the best amongst the dropcount and upcount if considered for testing model.
- I have trained different models using this method using training dataset and evaluated them using the RegressionEvaluator on the validation dataset to find the RMSE.
- The model created using log compression method works best with the data and gives the least RMSE (root mean squared error) on the validation data set as shown in the evaluation results.

CONCLUSION

A basic music recommender system can be trained efficiently on a cluster using Spark implementation of ALS method. I used a training set with 110,000 users. I observed that the log compression on count gives best precision for recommendation based on Ranking Metrics evaluation. For exact details about the results, please refer to my Logs_combined report.

EVALUATION

- **Evaluation on base implementation training via validation data**

rank: 4
alpha: 0.5
regularParam: 1
Root mean square error (RMSE): 8.159281940361623

- **Evaluation on log compression on count implementation training via validation data**

rank: 4
alpha: 0.5
regularParam: 1
Root mean square error (RMSE): 7.71668081677277

Issues Faced:

- The data given was huge being said it was the Big Data.
- Training and testing models took a huge amount of time.
- The DUMBO cluster I used for the project was having a few issues in the past and in the present due to which I was unable to get the results as quickly as possible.

Contributions:

I worked on this project alone and was responsible for the following tasks:

- Subsampling the dataset
- Training, training and preprocessing the Model
- Resolving Issues faced during running codes I wrote
- Developing the Pipeline
- Evaluation and Extension Implementation
- At the end, documenting the log file, the test results and drafting a report for the project.