KTH ID2223
Scalable Machine Learning and Deep Learning
Lab 1 - Linear Regression with SparkML

Fannar Magnusson (fannar@kth.se)
Thorsteinn Thorri Sigurdsson (ttsi@kth.se)

November 19, 2017

1 Introduction

The goal of the homework is to implement a scalable machine learning pipeline in Spark, using RDDs and the Dataframe API. We use the Million Song Dataset[1] to do predictions on what year a track is from depending on a number of features. We use Spark's implementation of Linear Regression, as well as implementing our own Spark abstraction (Linear Regression with Gradient Descent) to do the predictions and calculate the root-mean-square error.

2 Solution

We wrote our solution in Scala using the SparkML library, using IntelliJ as the development environment. We followed the suggested structure from the skeleton code provided.[2]

2.1 Task 1

In task one we are simply becoming familiar with the dataset and query the data using both RDDs and the Dataframe API. The number of songs in the subset of the dataset we are working with is 6724. The first column in the dataset is the release year of the song, the next columns are various features of the song.

2.2 Task 2

In task two we did transformations on the dataset to prepare it before we can do training on it. We do a number of transformations in a pipeline

[1]`https://labrosa.ee.columbia.edu/millionsong/`
[2]`https://github.com/id2223/vt17-lab1`

to transform a DataFrame of raw data to a two column DataFrame. The first column is the label (year of the track shifted so min year is 0) and the second column is a vector of doubles of the song's features (three features in our use case). The *transformers* used in our pipeline include for example a `RegexTokenizer` that splits the input data (comma-separated values) by commas, `array2vector` transformer, `VectorSlicer` that extracts the year (label) into a new column and a `Vector2DoubleUDF` transformer.

2.3 Task 3

In task three we start doing predictions on the data with basic Linear Regression. We split our dataset 80/20 - training/test data randomly. We add a Linear Regression *estimator* to out pipeline stages. We then use our Linear Regression estimator to create a model by fitting the training data. We evaluate the model and print the RMSE from the training summary. We finally use the model to do predictions on the testing dataset. We try using different parameters (maximum iterations and regularization parameter) for the Linear Regression and evaluate the model.

2.4 Task 4

In task four we use a `ParamGridBuilder` and `CrossValidator` to build combinations of a *maxIterations* and *regParam* to find what are the best parameters (given a number of options) for the Linear Regression.

2.5 Task 5

As we have only been using three features so far, we might want to extend that to improve the learning model. We can do that using a `PolynomialExpansion` *transformer*. The transformer gives us a new vector of features that we then use, along with CrossValidator to find a best model again.

2.6 Task 6

In task six we finish defining the `MyLinearRegressionImpl`, which is our own Spark abstraction implementation of Linear Regression with Gradient Descent. We implement task three again, now using our new Linear Regression estimator and compare to Spark's one.

3 Results

For the small Million Song Dataset (containing 6724 songs) we get root-mean-square error of 17.324597689471393 in task 3, using basic Linear Regression, maxIter = 10 and regParm = 0.1 In task 4 we find that maxIterat = 10 and regParam = 0.1 is the best combination of parameters. In task five, using the polynomial features, we get root-mean-square error of 17.160216935522577, which is slightly better than using the basic features, the best parameters in that case are maxIterations = 50 and regParam = 0.1. Finally using our `MyLinearRegressionImpl` we get root-mean-square error of 19.05829283065441.