Data Engineering: Spark ML subteam

Report on comparisons of speed between Spark ML and Sci-Kit Learn

Introduction

Spark ML

Spark ML is Apache Spark's in-built machine learning library, to be used on DataFrames. It is the successor to Spark's MLlib, and works faster due to the usage of DataFrames. Spark ML is particularly useful due to its various high level machine learning tools, including machine learning models and algorithms, data processing, and pipelines, which allow models to be constructed via an API that allows users to chain together various models, processing, and featurization steps into one conjugate model. It is mainly useful due to its easy convenience and usability with Python (pyspark), Java, and Scala.

Sci-Kit Learn

Sci-Kit Learn is a popular machine learning library for Python. It is probably the most popular machine learning library on Python, and provides a great deal of data visualization tools as well. It provides a similar features to Spark ML. It is mostly implemented using various libraries that compile to and run C/C++ to achieve optimal non-distributed data performance.

Question

Aims and Goals

Due to the near universal consensus that Sci-Kit Learn is a very useful library for machine learning library and that Spark's distributed system is one of the best available, we thought it would be prudent to compare their performances on machine learning models, especially related to time and accuracy.

Description

The dataset being used is a financial dataset, containing the details of opening price, closing price, high and low, and volume of trades for 851,264 spread out for variety of data. For both Spark (done via Python using Pyspark for parity) and Sci-Kit Learn, the aim of the experiment was to predict the trading volume as a function of the opening, closing, high, and low prices. This was relatively small dataset, especially given the resources of the server being used to process the data.

For the Spark ML part of the testing, the standard process was used; combining the columns into vectors via vector assembler, as Spark ML only works on vector inputs. After all the data was processed, the time to fit and train the model was calculated in the normal fashion (by measuring the difference of the start and end times), in order to get an measure of the time taken. Additional data was recordings were taken, by trying to edit the maximum regularization parameter in a loop, to see the effect of this parameter on times and RMS (root mean square error). This allowed a variety of different samples.

Sci-Kit was used in a similarly simple fashion, by constructing a linear regression model object, and training it with the the appropriate data, including a random split.

Results

Surprisingly, the results showed at Sci-Kit Learn (running on a non distributed system) is orders of magnitudes faster than Spark ML. From the various trials on Spark ML, it seems that the average time ranges from 8.2 to 8.9 seconds to train the model (although there were a couple outliers). This was quite high compared to Sci-Kit Learn, which took approximately 0.06878 seconds to train the model. Although both tests, Sci-Kit used a random data split of approximately 80/20, although this should not account for the massive differences in the results.

Additionally, the accuracy did seem to be superior with Sci-Kit, although not to the same extent as Spark ML. The RMS (root mean squared error) for the Spark ML model was 12426465.3241 and the Sci-Kit learn model was 12226663.45223931, for a difference of 199801.872.

Both of these results are surprising, but not unexpected when we looked deeper.

Explanation

The System

The most important thing to note is that the dataset itself is quite small; only about 850,000 samples, with only 4 features and a scalar output. This is small dataset, especially considering Spark's speciality is terabytes (and petabytes) of data. Thus, as a result, a lot the processing work that Spark does is essentially wasted as it is not too difficult to store this dataset in memory. This involves the work gone into creating workers and the workflow distribution is somewhat of a waste.

That being said, from our research it seem like Sci-Kit performs better for datasets that can be stored in memory. As this dataset is small, it seems that Sci-Kit is clearly the superior option. However, due to this fact, we still wanted to experiment with a larger dataset to see how this fact scales, and at which point does Spark become faster (as it is used in industrial level applications, Spark must clearly be faster).

Experiment 2

Aims, Goals, and Description

The aim of this experiment was quite simple. We wished to test larger data sets, in order to see how Spark ML and Sci-Kit scale. In order to manufacture our datasets, we used Sci-Kit's make_regression function, in order to create a large relatively linear dataset. After some trial and error (mainly related to creating datasets that were too large), we tested out multiple datasets: 1 million entries, 5 million entries, 10 million entries, 20 million entries, 100 million entries (each with 10 features/columns). After creating the dataset, we saved it as a CSV.

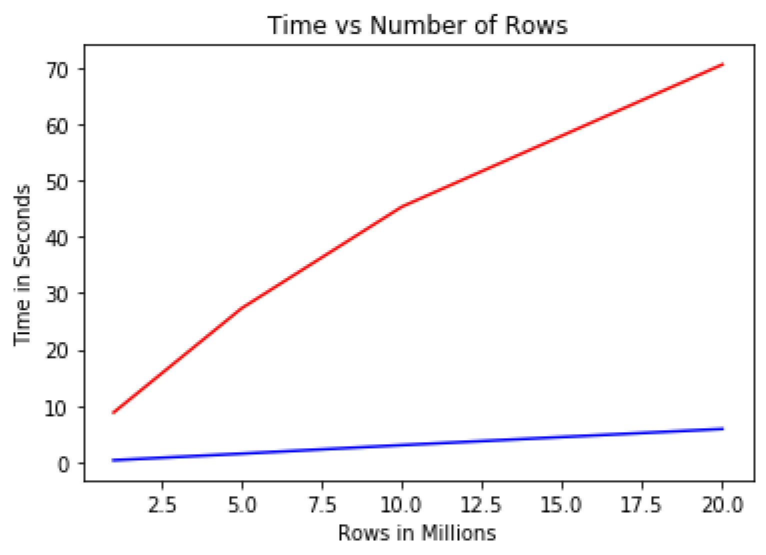
From this, it was a simple process. We used an 80/20, train/test split, and trained each of the data sets, using both Sci-Kit Learn and Spark. However, we did not measure the reading and loading times of the data. For the record, using Hadoop's file system also made Spark considerably faster in loading and reading the data, which took a longer time in Sci-Kit. Additionally, Spark supports better file formats, including Parquet, which would make this process even faster; however we did not use this in order to even the tests across both systems.

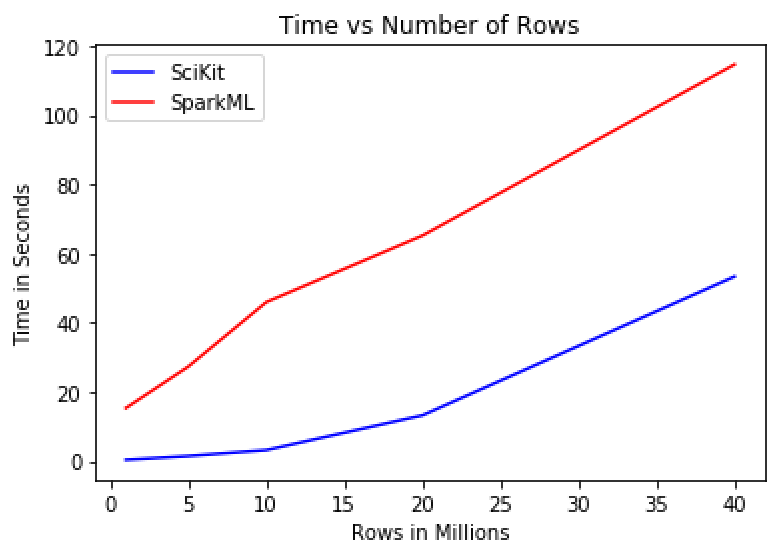
Most importantly, we used Pyspark instead of Scala. The main reason for this was due to the fact that we did not have a way to run Scala code on the server. Additionally, the Scala Zepplin notebooks are not very user-friendly. This aspect is important due to the fact Scala could possibly be faster than Python, for use in Spark. The sources on this were mixed. However, we thought it would be worth it to mention.

The times were compared and graphed for each test.

Results

The graph of the results is shown below:





As is evident from this graph, it is quite clear to see that the time taken for Sci-Kit, increases at a non-linear rate, while Spark ML, seems to increase at a somewhat logarithmic rate. This matches expectations as it is logical that the overhead required for Spark would result in the overall time requirement being lower as more rows are added. Additionally, it seems that Sci-Kit is still faster despite the increase in rows. However, at around 20 million rows it seems like Sci-Kit starts becoming much slower. We predict that this is due to the fact that beyond 20 million entries, the data can no longer be stored in memory. As for accuracy, both Sci-Kit and Spark ML were equally accurate within a small margin of error. While creating our datasets, we used a noise factor of 1, which means that it would be very difficult for any model to have a root mean squared error (RMSE) less than 1. For all of our models, in both packages and with all sized datasets, the RMSE were approximately equal to 1 within a small, negligible, margin of error. We can conclude from this that accuracy is not lost in Sci-Kit, even though the training time was shorter.

Conclusion

To conclude, in terms of comparing the sheer efficiency and ability of the ML algorithms, Sci-Kit learn seems to be much better than Spark ML. This is to be expected, as Sci-Kit was a library developed explicitly for machine learning purposes, whereas Spark ML was added to the Spark Framework as an expansion. Furthermore, Spark ML does seem to be better for huge, industry level data sets, with a lot more complicated data processing steps and a greater volume of data, as it seems to increase a logarithmic rate. In a more practical environment, due to many preprocessing steps usually involved in Machine Learning, there is reason to believe that Spark could be more efficient for use in a project with large amounts of data. .Using Scala and/or Parquet would probably net better results. However, in terms of pure machine learning performance, it does seem that with the exception of massive, industrial-level datasets, Sci-Kit Learn is superior to Spark ML.