Prediction of Solar Flares

Asish Kakumanu
50288695

Swapnika Pemmasani
50289464

1. Abstract

We predict solar flares by using the physical parameters provided by the Space-weather Helioseismic and Magnetic Imager Active Region Patches (SHARP) and various other data sources. SHARP contains various space weather quantities calculated from the photospheric vector magnetogram data. Achieving this require preprocessing and analysis on huge data which is where we use Apache Spark Framework which extends the capability of Hadoop File System and makes things much faster than Hadoop by providing an interface to program an entire cluster with parallelized data. In this instance, we load the data into a *hive repository* and later query using *spark SQL* to classify data based on their regions according to the maximum GOES magnitude of flares. Now, for each region we retrieve required parameters since the beginning of the flare and make a dataset with two splits viz., training and testing datasets. Now, we train a *ML Lib Random-Forests Algorithm* model on a different subsample for every iteration of this dataset and considering each as a tree node and aggregating to a decision tree. Now, we aggregate the predictions from its set of decision trees generated from each subsample to make a prediction of the occurrence of a certain class of flares on the basis of vertical current, peak, flux and in a given active region within the next 24 hours.

2. Problem Statement

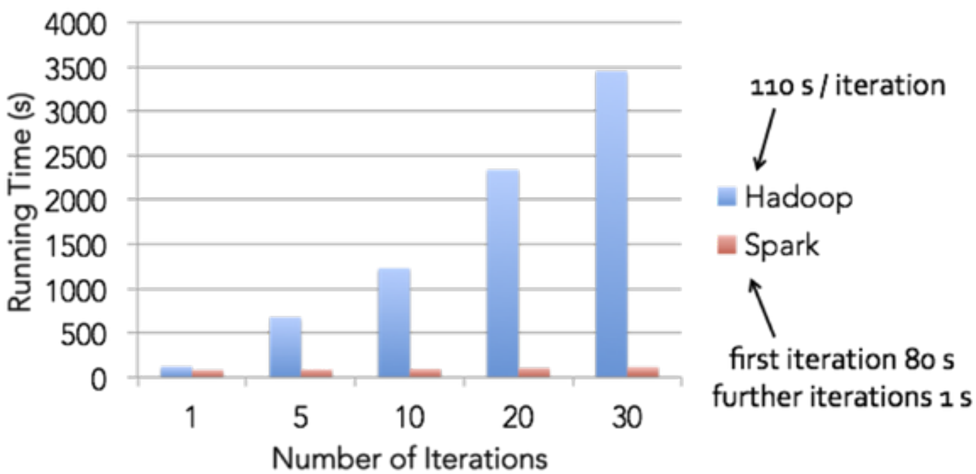
Solar Flares and *Coronal mass ejections* (CME) are no threat to humans on earth because of the earth's atmosphere. But, it's a major problem to our technology both on earth and in space. X- Class Solar flares and CME damage electric grids. They produce electrical currents in conductive material on the ground which can overload transformers and lead to a widespread blackout. Previously, in 1989, electricity was cut off to over 6 million people for 9 hours in Quebec. A clear analysis by *Nat Geo* showed that the CME which happened in 1921, if it happens today could lead more than 130 million people without power and 350+ transformers, major power grids would be at risk for permanent damage. Presently, we can give a heads up only 10-60 minutes before the geomagnetic storms as it takes a lot of time to analyze the data which is generated by SHARP (around 1TB data per day), which is too short to take any action. So, there is huge research going on this domain to predict solar storms or CME's 24-36 hours prior which will be useful to take preventive measures to minimize the effects of technology both on earth and in space, and thereby remain safe. Likewise, power grids on earth can be re-configured to provide extra grounding.

2.1 Tools Used

Apache Spark, Apache Hive Tables, Spark SQL, Apache MLLib

In this instance, we plan to use Apache Spark because it is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations. The main feature of Spark is its in-memory cluster computing that increases the processing speed of an application. Spark is designed to cover a wide range of iterative algorithms and interactive queries which helps in reduce the amount of time needed to analyze the data and make a prediction because of its simultaneous cluster computing system.

In comparison, the performance between Hadoop and Spark for an iterative computations is given below,



3. Solution

The use case can be solved by the following steps:

- Cleaning the data.
- Transforming data into feature vectors.
- Train the classification model

Spark machine learning API is divided into two packages called spark.mllib and spark.ml. The spark. mllib package contains the original API built on top of RDDs. On the other hand, the spark.ml package provides higher-level API built on top of Data Frames for constructing ML pipelines. To solve this use case, we are using spark MLLib.

3.1 Architecture

3.1.1 Data Collection

Data is collected from different resources namely NASA Open Data API, UCI ML Archive and NOAA (National Weather Service). This data is provided by the Space Weather Helioseismic and Magnetic Imager Active Region Patches (SHARP) which is by Stanford. It generates a terra byte amount of data a day. The attributes in the dataset are

- Start time of Flare
- Peak time of Flare
- End time of Flare
- Class of Flare
- X & Y Positions of Flare center

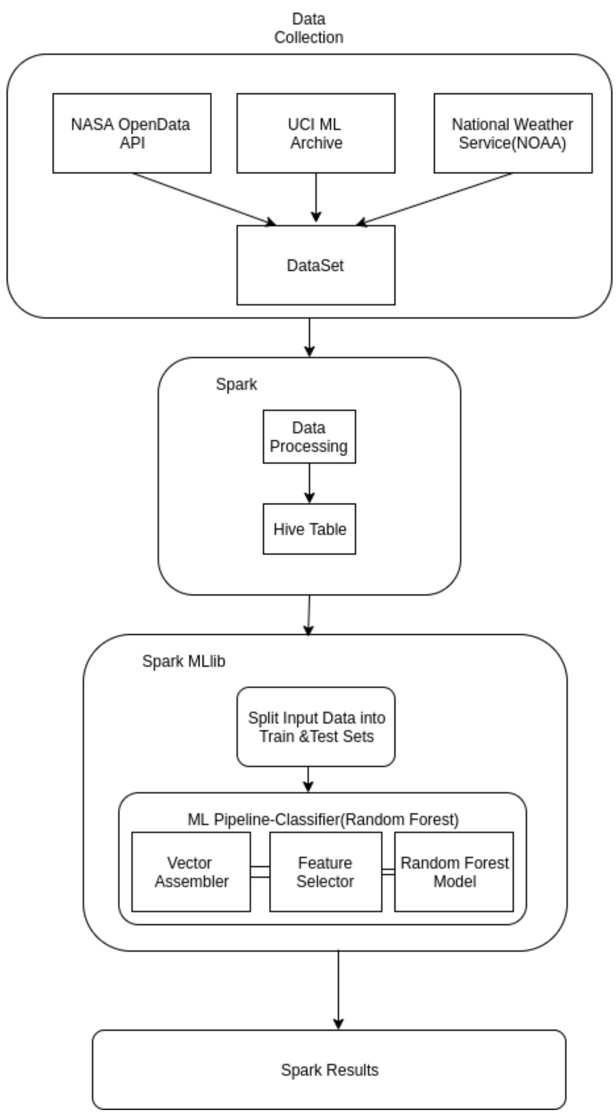
The start time of Flare is defined as the first four minutes. The peak time is the long-wavelength channel peak reaches a maximum which defines a class. Finally, the end time of flare is when the long channel reaches a half level between the peak and initial flare values.

3.1.2 Sample Data

Flare	Start time	Peak	End	Durs	Peak c/s	Total Counts	Sunward Detectors	Trigger	RHESSI Flare #
081102_2014	2-Nov-2008 20:14:55	20:15:07	20:16:12	78	3646	172685	n0 n3 n6 n1		
081211_1142	11-Dec-2008 11:42:14	11:42:51	11:45:31	197	2408	90103	n5 n1 n3 n4		8121110

3.1.3 Data Pipeline and Algorithm

This data is then made into a data frame and inserted into Hive tables. Now, we get the data from the beginning of the flare to classify the data based on their regions according to the maximum magnitude of flare. Now for each region, we query for required parameters and make a dataset with *two splits of 0.3 and 0.7* from the whole. Now, we combine a list of columns into a single vector column. Which is useful for combining features. This is used to train ML models which are based on decision trees. Post this step, we train a *MLLib Random Forest Algorithm* on different subsamples for every iteration of the dataset and considering each as a tree node and aggregate this to a decision tree. Now, we make a prediction using the decision tree generated from all the subsamples.



3.1.4 Implementation

3.1.4.1 Importing Data

Importing raw data from local file system to Spark RDD.

```
>>>Data = sc.textFile("file:///home/username/<filename>")
```

Type of data

```
>>>Type(data)
        <class ‘pyspark.rdd.RDD’>
```

Now, the raw data is split using map() function.

```
>>>df = data.map(lambda row:row.split(“ ”))
```

Now, the data is stored in a hive table using .saveAsTable()

```
>>>df.write.format(“orc”).saveAsTable(“solarFlares”)
```

3.1.4.2 Training the random forest

```
From pyspark.mllib.tree import RandomForest
From time import *
Start_time = time()
Model = RandomForest.trainClassifier(training_data, numClasses=2,
categoricalFeaturesInfo={}, numTrees=RF_NUM_TREES, featureSubsetStrategy=”auto”,
impurity=”gini”,maxDepth=RF_MAX_DEPTH, maxBins=RF_MAX_BINS,
seed=RANDOM_SEED)
End_time = time()
Elapsed_time = end_time – start_time
```

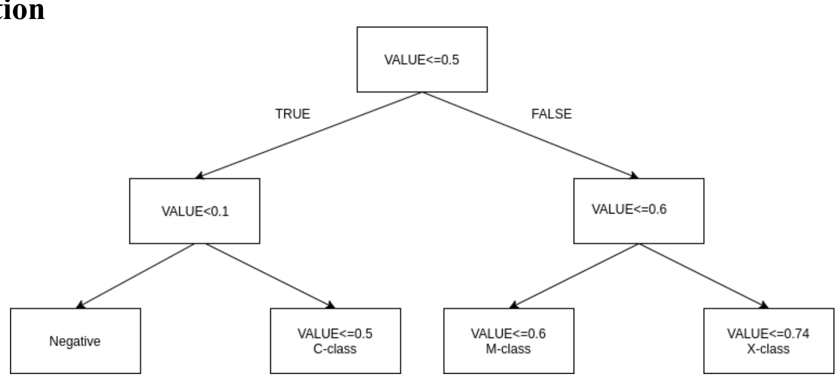
Prediction

```
Predictions = model.predict(test_data.map(lambda x:x.features))
Labels_and_predictions = test_data.map(lambda x:x.label).zip(predictions)
Acc = labels_and_predictions.filter(lambda x:x[0] == x[1].count() / float(test_data.count())
```

Evaluation

```
From pyspark.mllib.evaluation import BinaryClassificationMetrics
Metrics = BinaryClassificationMetrics(labels_and_predictions)
Metrics.areaUnderPR * 100
Metrics.areaUnderROC * 100
```

4. Visualization



5. Summary

To conclude, Spark helps to simplify the challenging and computationally intensive task of processing high volumes of real-time or archived data, seamlessly integrating relevant complex capabilities such as machine learning and many other algorithms. We observed that using the libraries in spark and map functions available, data can be processed efficiently. We also learned about spark ML package and used it to solve the use case on predicting solar flares.

6. References

1. Chang Liu, Na Deng, Jason T. L. Wang, Haimin Wang: Predicting Solar Flares Using Sdo/Hmi Vector Magnetic Data Product and Random Forest Algorithm (May 2017).
2. Ruizhe Ma, Soukaina Filali Boubrahimi, Shah Muhammad Hamdi, Rafal. A. Angryk: Solar Flare Prediction using Multivariate Time Series Decision Tree (Dec 2017).
3. Tarek A M Hamad, Zhiguang Wang, Alaa S. Al-Waisy: Deep Learning Technology for Predicting Solar Flares from (Geostationary Operational Environmental Satellite) Data.
4. Apache Spark Machine Learning Random Forest: https://mapr.com/blog/predicting-loan-credit-risk-using-apache-spark-machine-learning-random-forests/
5. Spark MLLib API Random Forest Algorithm as a Classifier: A Spark-based Approach: https://dzone.com/articles/classification-using-random-forest-with-spark-20
6. Random Forest using PySpark: https://jarrettmeyer.com/2017/05/04/random-forests-with-pyspark
7. Importing data into Hive tables using Spark and querying using Spark SQL: http://www.informit.com/articles/article.aspx?p=2756471&seqNum=5
8. Extracting, transforming and selecting features: Vector Assembler, StringIndexer, IndexerToString: https://spark.apache.org/docs/latest/ml-features.html#vectorassembler
9. Solar Flare Data National Centers for Environmental Information: https://www.ngdc.noaa.gov/stp/solar/solarflares.html
10. Knowledge extraction evolutionary learning: https://sci2s.ugr.es/keel/dataset.php?cod=1295
11. Nasa GBM Solar Flare List- Autonomous flare finder identifies all flares – GOES C-Class detected by GBM above 10 keV: https://data.nasa.gov/Space-Science/GBM-Solar-Flare-List/3dgk-yz3m
12. Solar Flare Machine Learning Repository Center for Machine Learning and Intelligent Systems: http://archive.ics.uci.edu/ml/datasets/solar+flare