PROPOSAL 6: [ML] [CEP] PREDICTIVE ANALYTICS WITH ONLINE DATA FOR WSO2 MACHINE LEARNER (ML)

(dananjayamahesh@gmail.com)

I am Mahesh Dananjaya, student from university of Moratuwa, the leading technical university of the country. I am majoring electronic and telecommunication engineering and I have been very enthusiastic to machine learning and big data analysis and also tried to implement more optimized and high performance platforms to implement the networking, telecommunication, signal processing and information processing.

I have done lots of projects related to machine intelligence and information processing and data analysis. Some of my projects include development of SOPC (System on Programmable Chip) for real time speech recognition on reconfigurable hardware, Artificial neural network implementation for pattern recognition in speech signals, intrusion detection application based on cluster analysis and several other projects in SVM, Regressions, Discriminant analysis (DA) with Fisher discriminant and ML, factor analysis for real world applications, principle component analysis (PCA) to reduce dimensionality in network information for information security analysis. And I have been engaging with the Big Data analysis research with the Dr. Prathapasinghe Dharmawansa from Stanford University (Currently in University of Moratuwa). My area is to combine IOT, Information Processing and machine learning areas. I have been specializing multivariate analysis in my mathematics area.

1 INTRODUCTION

Machine learning and Big Data analysis is becoming a next breakthrough of information processing and data analysis. Many organizations carried out many researches to bring the big data analytics a reality and to build up some commercial level applications. Nowadays there are many machine learning algorithms are there to come up with various solutions to build up some intelligent information analysis to usual data flow. Most of the network data is passing through high speed network lines as streams. Therefore the today's trend is to extend prevailing data analysis algorithms to support streaming and to provide the same level of predictive data analytics to stream data also. Company like WSO2 is trying to use those Big Data technologies and Machine Learning algorithms with their current product such as Data Analysis servers (DAS) and CEP (Complex Event Processors) with Siddhi

processor by supporting streaming of data. The use of machine learning algorithms for data analytics with the stream data support is an emerging technology.

2 PROBLEM STATEMENT

Even though WSO2 ML (Machine Learner) can be used to analyses cluster of data as well as standalone data and it is still not supporting stream of data. Therefore it cannot be directly used with the other products such as CEP (Complex Event Processor). And also WSO2 ML (Machine Learner) is using Apache Spark MLLib for their machine learning algorithms. Spark MLLib is supporting its machine learning algorithms for streaming k-means and other generalized linear models with the spark streaming by breaking of streaming data into mini batches. But still that technology is supported only with the Scala API. Therefore it cannot be directly used with the WS02 machine learning package.

3 PPROPOSAL

Proposed solution is to develop Java API for WSO2 ML with streaming data support and with spark MLLib k-mean and generalized linear models algorithms. This can be done with the Spark MLLib k-mean clustering and GLM (Generalized Linear Model) algorithms and by periodically retraining the model with the newly arriving sequential data. This will be implemented as a Siddhi extension that can operate directly on incoming streams that can be used to perform a predictive analysis of data/event streams coming from the WS02 CEP (Complex Event Processor). Thus we can use this for real time streamline data analysis by providing better predictions.

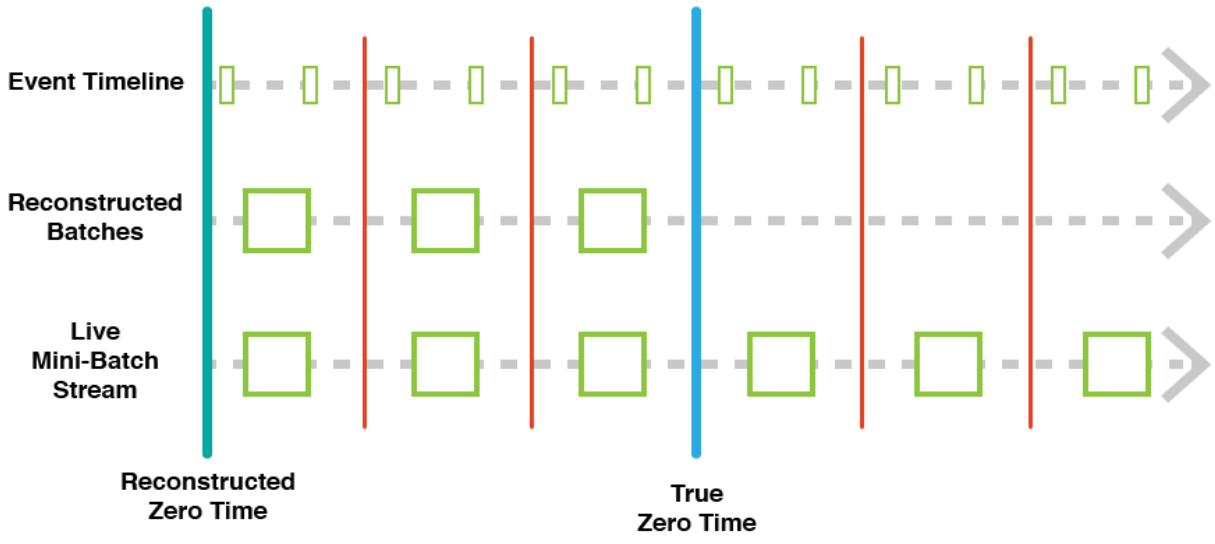
4 METHODOLOGY

What we are going to do basically is that we can uses the same Apache Spark MLLib for machine learning algorithms which we use for standalone data to analyze them and periodically retrain the model with upcoming data come as data streams from several applications such as WS02 CEP (Complex Event Processor)’s event streams. Therefore in this API I have to develop the same behavior inside apache spark which we have to get mini-batches of data from data coming as streams periodically by breaking of streaming data into mini batches and retrain the model. We use mini-batch learning in my incremental learning approach since these algorithms operates as stochastic gradient descents so that any learning with new data can be done on top of the previously learned models.

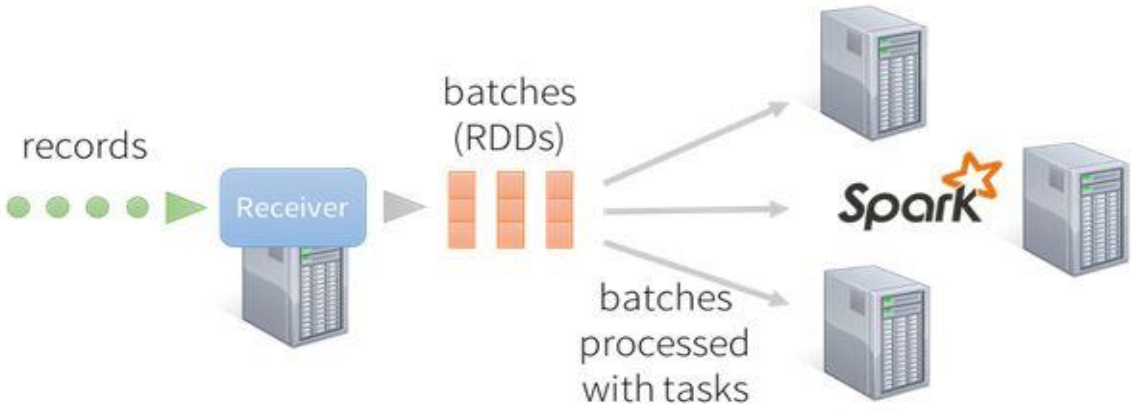
Then we have to save the previous model parameters and re-train the model again with the next set of mini-batch of data. Therefore in the process we should have to follow some basic steps. First we need to break the streams into several pieces of data sets which called the mini-batches and train a model repeatedly with those mini-batches. After each training run, we can save the model information (such as weights, intercepts for regression and cluster centers for k-mean clustering). As WS02 proposal page shows we use the first approach and try to implement incremental learning algorithms based on the mini batches of data taken from the data streams.

4.1 Mini-Batches from Streaming Data

First task is to taking pieces of data bundles out of streaming data which is known as mini-batches to retrain model. Initially we looked into how spark streaming taking the mini batches from streaming data. Basically Spark Streaming use the modified version of the Lambda Architecture, known as the Kappa Architecture. In this architecture the output of a fast, and often imprecise, streaming layer is joined with the output of a slow, but precise, batch layer. Typically this takes the form of a series of scheduled batch processes that replace the output of a continually running stream processor with a more accurate equivalent. This provides a guarantee that any inaccurate data introduced by the stream processor will be replaced by accurate data within a small window. In Spark Mini-Batch process, it defines several other important parameters to coordinate the mini-batches. They are Zero Time and Batch interval. The **batch interval** is the interval at which events that have been collected are processed. The **zero time** is defined as the same time throughout the entire distributed system, allowing Spark Streaming to use it to short-circuit computation of outputs from before inputs were available.



Spark MLLib can then it its machine learning algorithms to with those mini batches. This image shows how the spark mini batch sampling is happening. With the help of these methods, we can train models again with newly arriving data, keeping the characteristics learned with the previous data.



In the reference research paper of [1] in the references shows that how the mini batches and the mini batch size b with the convergence time T can be used for the Stochastic Gradient Descent (SGD) technique which can be effectively used for large scale optimization problems including predictive data analysis. Mini batch training needs to be employed to reduce the communication cost. However, an increase in mini batch size typically decreases the rate of convergence. Since we have to use spark MLLib to get the retraining model and we have the previous models saved in the program we can go for the SGD based optimization techniques to find the incremental model for any learning algorithms, initially k-mean clustering and Generalized Linear Regression models (GLM).

Therefore initially what we have to do is separate streaming data into set of data bundles called mini batch with the appropriate mini batch size **b** which will be a paramount important parameters to avoid

4.2 Running the Spark ML Algorithms and Save the Model parameters with Streaming Data.

For each mini batch we can use spark MLLib to run the relevant machine learning algorithm, in our case it is k mean clustering and generalized linear regression model (GLM). We have to run the algorithm and save the model parameters and when the next batch comes run the

algorithm and get the new parameters and then we can combine the models and build the new model with this incremental algorithms.

If we take the K mean clustering algorithm that is implemented on the spark streaming is elaborated below.

4.3 Periodically Train the models with the previous models.

For the periodically retraining of the model can be achieved by several ways. As the [1] research paper suggest and most of the people are using we can use mini-batch training with some stochastic gradient descent optimization techniques. So we can train the model periodically with mini batches.

Mini Batch Generalized Linear Regression

Especially when we comes to the Generalized Linear Regression we can use the stochastic gradient descent techniques to find the new model based on the past models also. As the spark streaming algorithms do referring to reference [5], we can use stochastic gradient descent methods for incremental learning model with GLS. We need to keenly look into Regularization of the parameters and along with various parameters associated with stochastic gradient descent such as Step Size, Number of Iterations and Mini batch fraction to control some issues regarding data horizon (how quickly a most recent data point becomes a part of the model) and data obsolescence (how long does it take a past data point to become irrelevant to the model.

Description of how the algorithms run with the references [8], [9], [10] and [11]. There those references describes the how it works and how we can use it with mini batch learning algorithms. Reference [8] shows clearly and simply how the mini batch size can be selected and how it can be parameterized b (B- batch size).

Mini Batch K Mean Clustering

When we comes to mini batch k mean clustering we can use the model parameters such as cluster center to periodically retrain the model. Referring to [4] spark streaming algorithm and Scala API we can write as following.

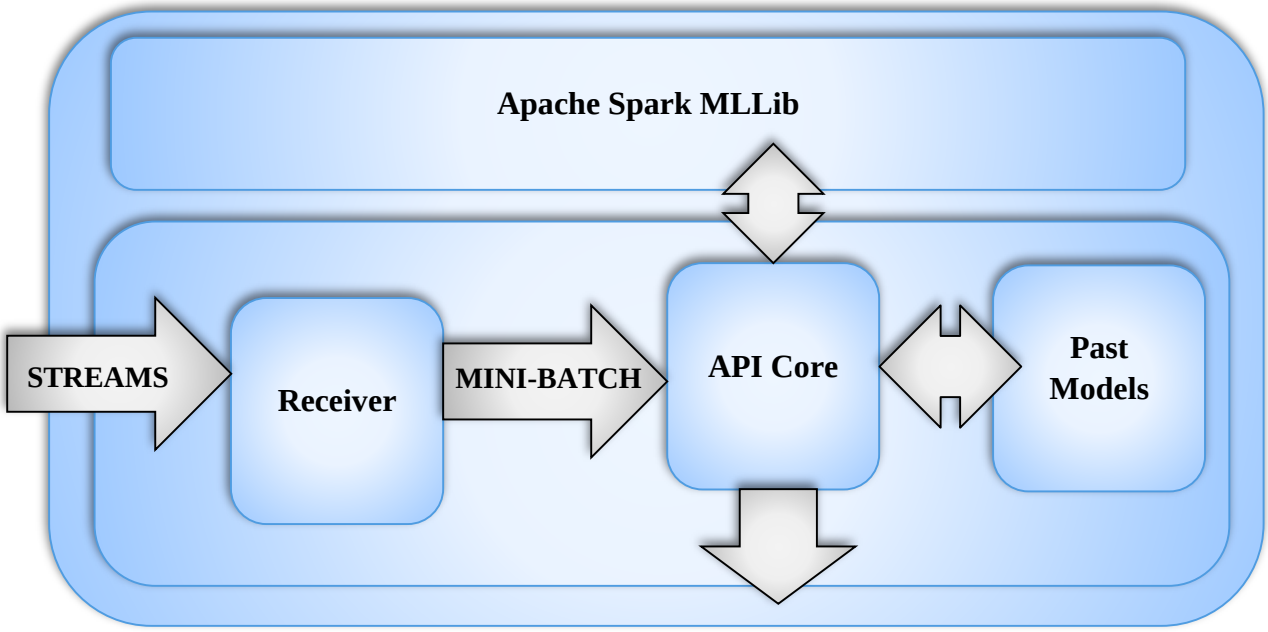
we may want to estimate clusters dynamically, updating them as new data arrive we need to develop the API to support streaming k mean clustering with mini batches, with parameters to control the decay (or "forgetfulness") of the estimates. The algorithm uses a generalization of the mini-batch k-means update rule. For each batch of data, we assign all points to their nearest cluster, compute new cluster centers, then update each cluster using:

$$C_{t+1} = \frac{C_t n_t \alpha + X_t m_t}{n_t \alpha + m_t}$$

Where, $n_{t+1} = n_t + m_t$

Where C_t*the previous center for the cluster is*, $n_t$is the number of points assigned to the cluster thus far, x_t is the new cluster center from the current batch, and m_t is the number of points added to the cluster in the current batch. The decay factor α can be used to ignore the past: with α=1 all data will be used from the beginning; with α=0 only the most recent data will be used. This is analogous to an exponentially-weighted moving average. These decaying factors make more control upon data horizon and data obsolesces to control the mini batch process.

5 ARCHITECTURE OVERVIEW



Streaming data in accepted inside the core. Those streams may be coming from the WSO2 CEP (Complex Event Processor), data streams coming out from the Siddhi processor, or else it can be another streamline of data. Then the stream of data I break into mini batches and passes into the API core where the algorithms run and store the models derived with the past models. Apache Spark is used to run the machine learning algorithms that we need to run for each and every mini batch of data.

6 Time Line of the Project

April 22- May 22	Community bonding period - Getting familiar with the platform and discussing implementation methods.
May 23 – June 18	Implementing streaming generalized linear regression and test for streams
June 19 - June 21	Implementing a part of streaming k-means
June 21-June 28	Midterm Evaluation
June 28-July 17	Implementing the other part of the streaming k means clustering and test and test streaming GLR further with streams.
July 18-August 12	Integrating The Java API with the CEP Siddhi processor streams and test with the CEP event streams and other streams.
August 13– August 16	Final Documentation of the project
August 16-August 24	Submit Codes and documentations and evaluations

7 References

[1] https://www.cs.cmu.edu/~muli/file/minibatch_sgd.pdf
[2] https://databricks.com/blog/2015/01/28/introducing-streaming-k-means-in-spark-1-2.html
[3]http://crediblecode.net/z-featureditems/featured-1/sparking-insights
[4]http://spark.apache.org/docs/latest/mllib-clustering.html#streaming-k-means
[5]http://spark.apache.org/docs/latest/mllib-linear-methods.html#streaming-linear-regression

[6] https://papers.nips.cc/paper/4432-better-mini-batch-algorithms-via-accelerated-gradient-methods.pdf

[7] https://databricks.com/blog/2015/01/28/introducing-streaming-k-means-in-spark-1-2.html

[8] http://stats.stackexchange.com/questions/140811/how-large-should-the-batch-size-be-for-stochastic-gradient-descent

[9] http://www.cs.cmu.edu/~muli/file/minibatch_sgd.pdf

[10] http://arxiv.org/pdf/1106.4574.pdf

[11] http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf