The Efficient Way of Detecting Anomalies in Large Scale Streaming Data

Lighari, Sheeraz Niaz; Hussain, Dil muhammed Akbar

Published in:
University of Sindh Journal of Information and Communication Technology (USJICT)

Publication date:
2018

Document Version
Publisher's PDF, also known as Version of record

University of Sindh Journal of Information and Communication Technology (USJICT)
Volume 2, Issue 3, July 2018
ISSN-E: 2523-1235, ISSN-P: 2521-5582
Website: http://sujo.usindh.edu.pk/index.php/USJICT/

The Efficient Way of Detecting Anomalies in Large Scale Streaming Data

Sheeraz Niaz Lighari, Dil Muhammad Akbar Hussain

Department of Energy Technology, Aalborg University
snl@et.aau.dk, akh@et.aau.dk

Abstract: These days many companies has marketed the big data streams in numerous applications including industry, Internet of Things and telecommunication. The stream of data produced by these applications may contain the values which are not normal. These values are called as anomalies. A lot of work has been done in anomaly detection to the batch data but detecting anomalies from streaming data nevertheless remains a largely available issue. In streaming data, the tasks related to find out the anomalies has become challenging with the passage of time because of the dynamic changes in data, which are produced by different methods applied in data streaming infrastructures. In the process of anomaly detection, first of all, it is required to know the way of finding the normal behavior of data and then it is easy to know the dynamic behavior or change in the data. In this context, clustering is a very prominent technique. The application of clustering method is very common to analyze the static data but in the field of data mining, it is key a problem especially on the streaming data. In this paper, we are applying streaming version of KMeans clustering algorithm for anomaly detection. The algorithm is analyzed both on single and distributed environments. Furthermore, we are investigating the stream of data to know various factors such as accuracy, anomaly detection time, true positive rate, and false positive rate. The data stream used in our analysis is generated from Kddcup99 dataset which is largely used in the field of intrusion detection.

Keywords: Batch data, Streaming data, Clustering, KMeans, and Anomaly detection

I. INTRODUCTION

The data stream belong to any application anyway it includes numerous inquiries to be tended to, for example, What kind of data is?, Is it critical or not?, Does it contain any value which isn't normal. In this paper, later is the primary focal point of our exploration.

Before we proceed to analyze the data, first we need to understand the data stream. It is a stream of data which is organized as succession of objects. When data is accessed from the data stream, these objects are read in a sequence. The reading of objects can be done one time or multiple times. In the effect, it is difficult to maintain all the objects in memory at a time. Therefore, every object must be investigated once while analyzing the data stream. Furthermore, utilization of memory should be limited as the new objects of data are constantly produced. Hence, the new objects should be processed on time immediately after the generation. Due to these requirements, the errors always generated while analyzing the data streams. Moreover, it is more challenging to analyze the values that are deviating from the original data because they are required to be processed soon after the generation.

Let us discuss what the anomalies are before the process of anomaly detection. The anomalies are the deviations from the normal pattern. They are usually alerts in the form of malicious activities, network attacks, faults and inaccuracies. Anomalies are given different names like outliers, intrusions or malwares etc.

If we talk about the online systems. They are producing the data continuously. The amount of data generated by them is so large that traditional methods are unable to monitor them. Furthermore, complexity of data makes them even more challenging for monitoring. The data monitoring systems must be equipped with a feature of anomaly detection. Anomaly detection is very necessary specially for the critical infrastructures.

In this paper, we propose an anomaly detection method based on KMeans Stream Clustering which is also called as online KMeans clustering. The KMeans comes with two flavors offline and online. The offline KMeans is trained once by the existing dataset, whereas in proper anomaly detection, we need to retrain the model after arrival of new data, in order to reduce the false positive rate. The retraining of the model is the main feature of online KMeans. One of the version of online KMeans is to use the mini-batches [1]. In this type of training, the model is trained with multiple subsets of the dataset instead of training it with multiple iterations. Moreover, in this type of KMeans, the cluster centers are recomputed after every mini-batch. This type of technique can be applied in the streaming model. In Streaming KMeans of apache spark MLlib, the cluster centers are updated by arrival of new time window of the data. For making it an adaptive model, a new parameter is

specified called as half life. On the basis of half life parameter, the points are decided weak or strong from the incoming batch of data. The half life controls, after how many data points or time windows the previous impacts center computation only to the half, providing you the control to tune the "forgetfulness" of your model [2]. We further discuss the streaming frameworks used in the project which are as in the following:

A. Apache Spark

It is an open source tool. Conventionally, Hadoop and MapReduce engines are built on acyclic data flow which is not suitable for real-time applications. In acyclic data flow, the data is read from disk or storage and then stored back after the processing. This way of reading and writing is time-consuming and also expensive in computation.

Sparks RDD's (Resilient Distributed Datasets) handles this issue effectively. RDD's are saved in memory, which takes less time as compared to read and write from the disk. They also able to remake themselves from DAG (Dynamic Acyclic Graph) in the situation of failure.

B. Spark Streaming

The idea of Spark Streaming [3] is depicted in the figure no.1. The Spark streams are divided into the batches at the user-specified intermissions. These batches are then considered as RDD's, and then they are dispatched to the spark cluster.

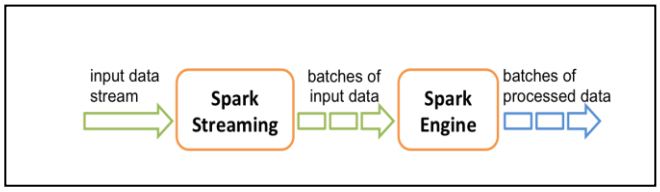

Figure 1. Spark Streaming

The data stream in spark is represented as DStream. In DStream, every RDD is contained in the specific interval of time as shown in the figure no.2.

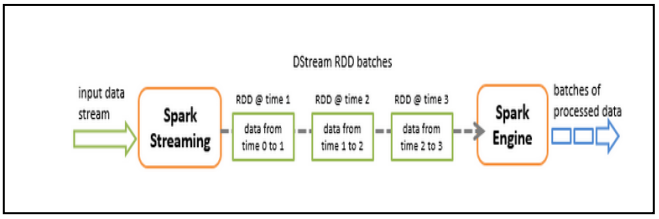

Figure 2. Spark Dstream

C. Apache Kafka

Apache kafka is a distributed messaging system. It is capable of streaming the data in real-time. It is working on the model called as Publish-Subscribe model. The kafka can scale out easily because of its distributed nature [6].
Kafka contains many components, which take part in streaming, and they are Kafka producer, Kafka consumer, Kafka topic, Kafka broker, and Zookeeper.
Kafka topics: They are the group of messages. It is actually a stream of publishing the messages.
Kafka producer: A process publishes the messages on the topics. The messages are first dispatched to the kafka broker, then they are send to topics for publishing.
Kafka consumer: It is a process, which is responsible to subscribe the messages to topics. Consumer gets the messages from the topics when it has to consume the message.
Kafka Broker: It keeps the messages to a particular amount of time, which are send to him by the producer, and then kafka consumer reads the messages from the broker which are in the form of topics.
Zookeeper: It is used to coordinate between the brokers. Zookeeper is also responsible to send the information to producer and consumers that the brokers is not working.

II. RELATED WORK

In literature, we see some examples of anomaly detection or intrusion detection based on signature rule based techniques. The authors in [7] present the anomaly detection based on the signatures and then it discuss the evolution of rule-based technique from the signature-based technique. Moreover, in the reference [8] the anomaly detection is performed using fuzzy means in addition with the signature rules. The signature based or rule-based techniques work best with only known anomalies but for both known and unknown anomalies, we need another technique, which is called Machine learning. There are two kinds of machine learning; supervised and unsupervised. The anomaly detection based on the supervised algorithms is presented in the reference [9]. The anomaly detection using both supervised and unsupervised algorithms is applied in the reference [10]. Furthermore, the authors in the reference [11] proposes the anomaly detection model based on rule based and unsupervised learning algorithms. In this paper, we are mainly focusing on the unsupervised algorithm KMeans clustering because of its wide usage in the problems of anomaly detection.

The authors in reference [12] propose a cluster-based anomaly detection working on the distributed environment of Hadoop and MapReduce. The authors in reference [13] use context based clustering with KMeans to detect the anomalies and it is also implemented on the Hadoop

framework. The clustering using KMeans or Streaming KMeans is also a part of a tool called as Apache Mahout [14] but it performs in batch mode, which is a time consuming, and it does not fit to the process of the real-time clustering.

All the techniques described above are based on Hadoop and MapReduce frameworks. Hadoop is primarily designed for batch data where the data exists before the processing. It fails at processing of data, which comes incessantly real time. In Hadoop, for new incoming data it requires to rebuild the model with all existing data, which is the wasteful technique. Hence, Hadoop is not appropriate for real-time processing. Similarly HBase [15] and BashReduce [16] undergoes with the same problem. Moreover, there are other streaming processing tools like apache storm [17] and apache S4 [18] but apache spark outperforms with respect to speed [19]. Hence, it was the main reason to use apache spark as our tool of the experiment. Furthermore, it is fault-tolerant and scalable.

We have performed our experiments on VMware environment, which is performed on single machine in [20], but we are implementing it with more than one machine which gives the flavor of distributed work. We have also compared the time taken by single machine with the distributed environment. Furthermore, we are analyzing other parameters like accuracy, anomaly detection time, false positive rate and true positive rate while [20] only calculates the accuracy.

III. PROPOSED MODEL

In the proposed model as in figure no.3, first of all, the stream of data is generated using the Kafka producer from the dataset Kddcup99. The data reaches at the spark engine in the form of Kafka queues or partitions. Each newly arrived data instance is tested by anomaly detection model to check its possibility of being an anomalous instance. After checking the data status, the model can be retrained with either of two methods. In first method, the model is retrained with only newly arrived instance, which saves time, and in second method the model is retrained with the existing data and newly arriving data instance which takes time and suitable for real-time applications. The first method is used by apache spark Streaming KMeans as in our work while mini-batch algorithms use the second method. After checking the anomalies, the accuracy rate of the algorithm is calculated along with other parameters like false positive rate, true positive rate, and anomaly detection time. All the parameters are computed both on the single and distributed mode.

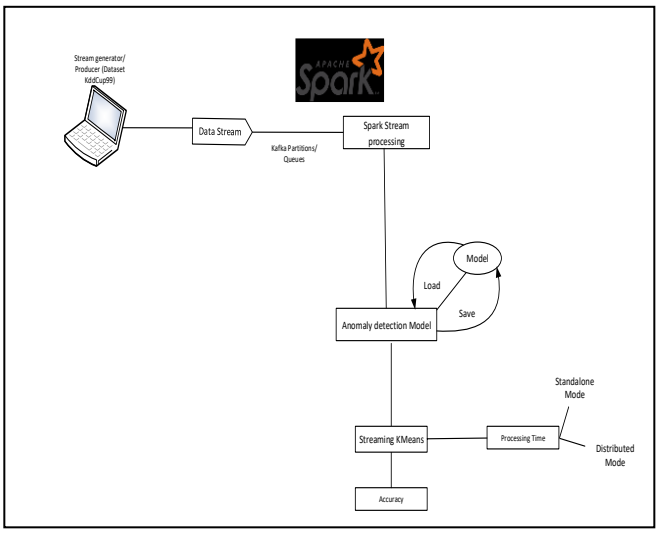

Figure 3. Streaming data anomaly detection model

IV. ALGORITHM FOR ANOMALY DETECTION

1. Setting up the Spark Cluster
2. Reading the input Stream from Kafka Cluster
3. Normalize the input stream by standard deviation and mean.
   (a) Mean is broadcaste
   (b) Standard deviation is broadcasted
4. Apply the KMeans model on normalized streaming data
   (a) Streaming Kmeans model is broadcasted
5. DistancetoCentroid is calculated
6. The distance exceeds the threshold, It is assumed to be an anomaly.

The cluster is premediated in Figure No.3 utilizing the apache spark. The stream produced on KddCup99 dataset is getting through the Kafka Queues. The information stream is isolated in little units called as the DStream. To prepare the information and foresee the anomalies from the streaming data, we set numerous changes and activities. At each progression of the change and activity, a piece of the information is put away in memory and the outcomes are moved to the following Dstream unit for additional preparation. In additional preparation process, the surge of information is gotten by the model and after that it proceeds to the anomaly detection process.

During the process of anomaly detection, the spark DStream model gets the stream from the cluster of Kafka. The DStream contains four key groups of attacks in the dataset like U2R, R2L, DoS and Probe. The principle calculation of

the DStream begins to process mean with mapped and reduced transformations. From mean, the square of the distinction is figured and from the distinction, the difference is figured. Similarly from distinction, the standard deviation is computed. After every one of these computations, the values are then standardized using standard deviation and mean. Subsequent to playing out the normalization procedure, the KMeans streaming model is made and introduced. The model is then connected to information which was normalized in the previous step. After at that point, it is the way toward figuring the DistancetoCentroid for each element. At that point the code defines the threshold. On the off chance that the DistancetoCentroid of the component surpasses the threshold, at that point it is reflected as an anomaly.

Besides, in the proposed work, the procedure of anomaly detection begins by utilizing feature engineering which helps to remove the four attack labels rather than 21 present in the dataset. In the process of Stream clustering, the clusters are updated when new data arrives in the Kafka queues.

V. STREAM OF DATA GENERATED FROM KDDCUP99

There 42 different labels of the KDDCup99 dataset. The attributes from 1 to 9 are the fields defining the the tcp connection [21]. The attributes from 32 to 42 are used to evaluate the attacks in the dataset. Among these attributes, the attribute 41 is used to identify the type of the attack. There are 4 different categories of the attacks in the dataset as specified in table no.1.

0,1032,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,511,511,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,smurf.
0,1032,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,511,511,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,smurf.
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,237,18,0.00,0.00,1.00,1.00,0.08,0.07,0.00,255,18,0.07,0.07,0.00,0.00,0.00,0.00,1.00,1.00,neptune.
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,215,7,1.00,1.00,0.00,0.00,0.03,0.06,0.00,255,7,0.03,0.07,0.00,0.00,1.00,1.00,0.00,0.00,neptune.
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,135,11,1.00,1.00,0.00,0.00,0.08,0.07,0.00,255,11,0.04,0.06,0.00,0.00,1.00,1.00,0.00,0.00,neptune.
0,1032,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,511,511,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,smurf.
19,209,450,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,3,6,0.00,0.00,0.00,0.00,1.00,0.00,0.50,215,218,1.00,0.00,0.00,0.01,0.00,0.00,0.00,0.00,normal.
0,959,331,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,2,0.00,0.00,0.00,0.00,1.00,0.00,1.00,87,48,0.55,0.05,0.01,0.00,0.00,0.00,0.00,0.00,normal.
0,1032,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,511,511,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,smurf.
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,246,17,1.00,1.00,0.00,0.00,0.07,0.06,0.00,255,16,0.06,0.07,0.00,0.00,1.00,1.00,0.00,0.00,neptune.
Processed a total of 46297 messages

VI. TYPES OF ANOMALIES

In the table no.1, we are calculating the anomaly type from the stream of data. The analysis result show that most of the anomalous data instances are DoS (Denial of Service) instances.

Table I. Types of Anomalies

Index	DoS	Probe	U2R	R2L	Normal
1	15	0	0	0	8
2	79	0	0	0	20
3	9	0	0	0	1
4	4	0	0	0	0
5	2	0	0	0	1
6	1	0	0	0	1
7	2	0	0	0	1
8	3	0	0	0	0
9	5	0	0	0	2
10	6	0	0	0	2

VII. RESULTS

The results of the experiments performed in our work are described in table no.2 and table no.3. The table no.2 contains the results of single mode which consists of 1 Master and 1 Worker. The table no.3 exhibits the results of the distributed mode which includes 1 Master and 2 Workers. In the table no.2, if we calculate the average accuracy of the model, it is 76%. Interestingly, some of results showing the 100% accuracy which is because of the no false positives at the indexes of 4 and 8. Furthermore, the table no.2 shows the minimum anomaly detection or processing time of 49seconds, which is shown in the index no.1. On contrast, in the table no.3, which is the case of distributed mode, where minimum processing time is 33seconds and the average accuracy, is 68%. In this mode, we also observed the 100% accuracy at the index no. 7 and 9. Although, there is a slight difference of accuracy rate both in single and distributed mode but the distributed mode showing a good amount of the reduction in the processing time.

If we further say the reason behind the 100% accuracy rate, it may be due to the detected anomalies and real anomalies, which are same at number to the particular instance of data.

Table II. 1 Master and 1 Worker (Single Mode)

Index	Accuracy	Anomaly detection time (Seconds)	Detected Anomalies	Real Anomalies	True positive
1	.65	49.019343011	23	15	15
2	.79	57.92875465	99	79	79
3	0.9	64.316454181	10	9	9
4	1.0	70.170019339	4	4	4
5	.66	74.259295139	3	2	2
6	.5	77.41612176	2	1	1
7	.66	80.249553008	3	2	2
8	1.0	83.804295379	3	3	3
9	.71	86.803457384	7	5	5
10	.75	89.542611188	8	6	6
Index	**False positive**	**Positives**	**Negatives**	**True Positive Rate**	**False Positive Rate**
1	8	23	24	0.65	0.33
2	20	99	100	.79	.2
3	1	10	11	0.9	.09
4	0	4	5	1.0	0.0
5	1	3	4	.66	.25
6	1	2	3	.5	.33
7	1	3	4	.66	.25
8	0	3	4	.66	.25
9	2	7	8	.71	.25
10	2	8	9	.75	.22

Table III. 1 Master 2 Workers (Distributed Mode)

Index	Accuracy	Anomaly detection time (Seconds)	Detected Anomalies	Real Anomalies	True positive
1	.84	33.276976865	25	21	21
2	.78	43.750534611	65	51	51
3	.5	51.004885917	4	2	2
4	.6	57.031494693	5	3	3
5	.5	62.526298527	2	1	1
6	.9	67.899759682	10	9	9
7	**1.0**	72.342786019	3	3	3
8	**.2**	76.306710162	5	1	1
9	**1.0**	80.787546223	8	8	8
10	**.5**	84.916666357	2	1	1
Index	**False positive**	**Positives**	**Negatives**	**True Positive Rate**	**False Positive Rate**
1	4	25	26	.84	.15
2	14	65	71	.78	.19
3	2	4	10	.5	.2
4	2	5	9	.6	.22
5	1	2	6	.5	.16
6	1	10	12	.9	.08
7	0	3	9	1.0	0.0
8	4	5	15	.2	.26
9	0	8	9	1.0	0.0
10	1	2	7	.5	.14

VIII. CONCLUSION

In this paper, we have implemented the anomaly detection of streaming data using Streaming KMeans. The experiments were performed using the apache kafka and apache spark. We produced the stream of dataset from Kddcup99 using the API's of apache kafka producer. To detect the anomalies, we designed a kafka consumer based on the apache spark for processing both on the single and distributed mode. In the experiment, we firstly computed the rate of accuracy of the Streaming KMeans model then we calculated the anomaly detection time both on the single and distributed mode. The model is showing 76% and 68% accuracy rates respectively on single and standard mode. At some instances, it has also exhibited the 100% accuracy rate because of same amount of detected anomalies and real anomalies or having no false positive rate at some instances of data. Furthermore, in the case of anomaly detection we saw a considerable reduction in the processing time when the process is distributed to the multiple machines. In our case, it is up to 33seconds when we are using 1 master and 2 worker which is 49seconds when it is 1 master and 1 worker.

REFERENCES

[1]. Sculley D , "Web-scale k-means clustering", In: Proceedings of the 19th international conference on World wide web. ACM 2010 S. 1177-1178

[2]. https://www.inovex.de/blog/online-offline-machine-learning-network-anomaly-detection/

[3]. https://spark.apache.org/streaming/

[4]. https://hortonworks.com/hadoop-tutorial/introduction-spark-streaming/

[5]. https://mapr.com/blog/spark-streaming-hbase/

[6]. Kreps, Jay, Neha Narkhede, and Jun Rao. "Kafka: A distributed messaging system for log processing." in Proceedings of the NetDB, pp. 1-7, June 2011.

[7]. W. Lee , J. Stolfo , "A framework for constructing features and models for intrusion detection systems," ACM Trans. Inf. Syst. Sec., 2000

[8]. S. Bridges, B. Vaughn, "Fuzzy Data Mining and Genetic Algorithms Applied to Intrusion Detection," Proceedings of the National Information Systems Security Conference (NISSC), Baltimore,MD, October, 2000

[9]. Govinda, Manish, "A framework for fast and efficient cybersecurity," Conference on Advances in Computing & Communications, ICACC 2016, 6-8 September 2016, Cochin, India

[10]. Lighari, S. N. and Hussain, D. M. A, "Testing of algorithms for anomaly detection in Big data using apache spark", 1 Sep 2017 2017 9th International Conference on Computational Intelligence and Communication Networks (CICN). IEEE, p. 97-100 4 p. (International Conference on Computational Intelligence and Communication Networks (CICN)).

[11]. Lighari, S. N. & Hussain, D. M. A, "Hybrid model of rule based and clustering analysis for big data security", 1 Nov 2017 2017 First International Conference on Latest trends in Electrical Engineering and Computing Technologies (INTELLECT). IEEE, p. 1-5 5 p

[12]. L. Yu and Z. Lan, "A scalable, non-parametric anomaly detection framework for hadoop," in Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference. acm, 2013, p. 22.

[13]. M. Gupta, A. B. Sharma, H. Chen, and G. Jiang, "Context-aware time series anomaly detection for complex systems", Published 2013

[14]. Apache mahout. [Online]. Available: https://mahout.apache.org/

[15]. Apache hbase. [Online]. Available: https://hbase.apache.org/

[16]. http://www.linux-mag.com/id/7407/

[17]. Storm-distributed and fault-tolerant realtime computation.[Online].Available: http://storm.incubator.apache.org/

[18]. s4. [Online]. Available: http://incubator.apache.org/s4

[19]. M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. Mccauley, M. Franklin, S. Shenker, and I. Stoica, "Fast and interactive analytics over hadoop data with spark", August 2012

[20]. Padma, Priya, and Chitturi, "Spark for data science cook book", Packt 2016

[21]. Preeti, Sudheer, "Analysis of Kdd Dataset attributes-Class wise intrusion detection", ICRTC 2015