International Journal of Scientific Research in Computer Science and Engineering

Research Paper

ISSN 2320-7639

Volume-1, Issue-2, Mar- Apr-2013

Available online at www.isroset.org

Outlier Detection Based on Clustering Over Sensed Data Using Hadoop

V. Jain

Institute of Engineering & Technology, Devi Ahilya Vishwavidyalaya Indore, India

Received: 16 Mar 2013 Revised: 22 Mar 2013 Accepted: 18 Apr 2013 Published: 30 Apr 2013

Abstract- Outliers are regarded as noisy data in statistics, has turned out to be an important problem which is being researched in diverse fields of research and application domains. Many outlier detection techniques have been developed specific to certain application domains, while some techniques are more generic. Outlier detection aims to find patterns in data that do not conform to expected behaviour. It has extensive use in a wide variety of applications such as military surveillance for enemy activities, intrusion detection in cyber security, fraud detection for credit cards, insurance or health care and fault detection in safety critical systems. In our work, we investigate that there is need to develop an outlier detection solution for large amount of sensed data facts to optimize the processing of data mining. Sensed data is the output of sensor nodes consisting the real values after sensing. Existing solutions provide outlier detection only for static datasets and using clustering algorithms for normal data size. In our work, we have developed an outlier detection system which performs outlier detection of Intel sensed dataset using clustering algorithms DBScan and K-Means. Experimental study has been performed using java application and hadoop system.

Keywords: Outlier detection, Clustering, Hadoop.

1. INTRODUCTION

Extraction of information is an important aspect in data mining field which not only results in driving force for business strategies but also a prominent factor in the detection of frauds and any unusual events and activities. This extraction process is easily possible with small size dataset but as soon as the size of dataset tends to increase, the complexity starts to arises, and in that large amount of dataset which is generally expected to be scattered and unstructured. It is not always possible that every information is relevant. It may contain some data which is not of interest for a particular user but may be useful for others. Such immaterial data is called outlier.

Hence, outliers are those data objects that shows deviated behaviour from normal data objects [2]. To find these deviated patterns we need some techniques that are called outlier detection techniques. There are various techniques that finds outlier in the given dataset, It is easily possible with the small dataset but if volume of data is too large which do not fit into single disk [8], variety is more and unstructured, if it is then, so many overhead starts to appear like, memory starts to run out of the bound, processing and execution time increases, efficiency and accuracy gets reduces.

To overcome these overheads, Hadoop is widely adopted to support distributed applications [7]. Hadoop is the most popular alternative to the best of our knowledge, thus we in our work finding outlier using Hadoop. Hadoop is able to handle the large files easily as it has its own dedicated storage and processing unit i.e. HDFS and MapReduce respectively [5]. HDFS stores the datasets and MapReduce processes the data stored at HDFS. Hadoop uses MapReduce framework which is the processing unit, it is scalable, reliable and fault tolerant. This framework splits the large data into the chunks and then process all the small chunks simultaneously through map function using <key, value> pair and output of this map function is given as input to reduce function which merges the small processed units and generates a final output.

In our work for detecting outliers we are using DBScan algorithm. This algorithm clusters the data on basis of density which requires two parameters as input i.e. Eps and Minpts. Eps defines the radius or the neighbourhood of cluster and Minpts defines the minimum number of points in the cluster. The dataset we are using in our work is from Intel laboratory Berkeley, US in which data is collected through 54 sensors deployed in the lab [1]. This is wireless sensor data which is a real time data. Real time data are more vulnerable and prone to be maliciously altered thus it is highly recommended to find outlier and discard the noise data from it. Based on experiments, we found that K-Mean works better than DBScan algorithm if computation time is major concern whereas if we concern about outliers and noise than DBScan is the better choice.

In our study, we also found that WEKA tool [12] could not handle a large dataset.

2. BACKGROUND

BigData is growing rapidly day by day. BigData is set of large dataset and large size creates complexity and difficulty for processing, storage, transfer, visualization and analysis. Due to high scale and large size of collection, it is very difficult for processing and mining purpose. Traditional processing algorithms and architectures are not perfectly suitable for BigData thus it incorporates the set of techniques and technologies that require new forms of integration to uncover large hidden values from large datasets that are diverse, complex, and of a massive scale.

2.1 Hadoop

Hadoop is open-source software that enables reliable, scalable, distributed computing on clusters of inexpensive servers. Hadoop provides its services using MapReduce Framework [16]. Hadoop is used for processing, storing and analyzing massive amounts of distributed unstructured data. As a distributed file storage subsystem, Hadoop Distributed File System (HDFS) was designed to handle petabytes and exabyte of data distributed over multiple nodes in parallel.

2.2 HDFS Architecture

The Hadoop Distributed File System (HDFS) is the file system component of the Hadoop framework. HDFS is designed and optimized to store data over a large amount of low cost hardware in a distributed fashion.

2.3 MapReduce

Map Reduce is a software framework for distributed processing of large data sets on computer clusters [11]. It is first developed by Google. Map Reduce is intended to facilitate and simplify the processing of vast amounts of data in parallel on large clusters of commodity hardware in a reliable, fault-tolerant manner. As shown in Fig. 1, this framework works on two methods i.e. map() function and reduce() function. MapReduce framework receives input and generates output in <*key*, *value*> pair.

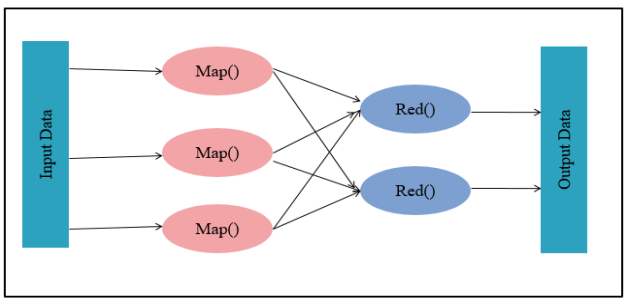

Fig. 1: MapReduce Framework

2.4 Clustering

Clustering is a division of data into groups of similar objects. Each group, called cluster, consists of objects that are similar between themselves and dissimilar to objects of other groups. Representing data by fewer clusters necessarily loses certain fine details, but achieves simplification. It represents many data objects by few clusters, and hence, it models data by its clusters. Data modeling puts clustering in a historical perspective rooted in mathematics, statistics, and numerical analysis. From a machine learning perspective clusters correspond to hidden patterns, the search for clusters is unsupervised learning, and the resulting system represents a data concept. Therefore, clustering is unsupervised learning of a hidden data concept. Data mining deals with large databases that impose on clustering analysis additional severe computational requirements. These challenges led to the emergence of powerful broadly applicable data mining clustering methods surveyed below.

2.5 DBSCAN CLUSTERING ALGORITHM

DBSCAN (Density Based Spatial Clustering of Applications with Noise) algorithm clusters the dataset on basis of density. Data objects within cluster have higher density as compared to those which are outside the clusters. Data objects those are outside the cluster are considered as noise. As shown in Fig. 2, cluster formation is done on basis of two parameters i.e. radius (Eps) and minimum points (MinPts).With this understanding, we can describe core, border and noise points in a given data set.

Core points: A point is a core point if the number of points within a given radius (Eps) has more than specified no. of points (MinPts). This point exists inside the cluster.

Border points: A border point is not a core point, but falls within the neighbourhood of a core point.

Noise points: A noise point is any point that is neither a core point nor a border point.

About DBSCAN Algorithm

1. It labels all points as core, border or noise points.
2. Eliminate noise points.
3. Make each group of connected core points into a separate cluster.
4. Assign each border point to one of the clusters of its associated core points.

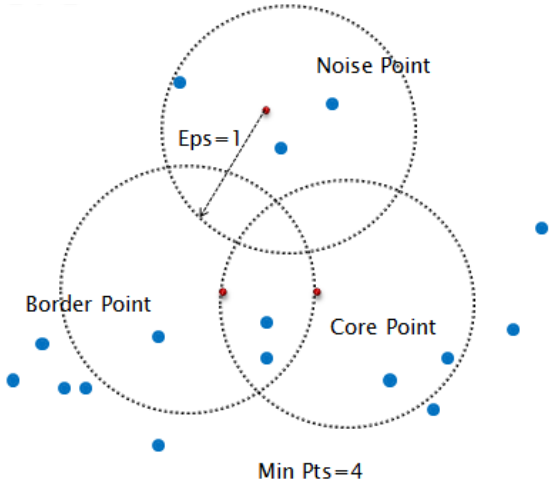

Fig.2: Border and Core Points Representation of DBScan

2.6 K-MEANS CLUSTERING ALGORITHM

K-Mean clustering is a centroid based partitioning technique. This algorithm aims to partition the n observation into *k* clusters in which each observation belongs to the cluster. The no. of clusters generated depends on the user defined value i.e. *k*. An objective function is used to assess the partitioning quality so that objects within the cluster are similar to one another but dissimilar to objects in other cluster. K-Means Clustering is a technique which can be used to provide a structure to unstructured data so that valuable information can be extracted. The key to the implementation of the K-Means algorithm using hadoop is the design of the Mapper and Reducer routines.

3. RELATED WORK

Outlier detection in wireless sensor networks (WSNs) is the process of identifying those data instances that deviate from the rest of data patterns based on certain measures. WSNs are vulnerable to faults and malicious attacks; this in turn causes inaccurate and unreliable sensor readings. Traditional outlier detection techniques are not directly applicable to wireless sensor networks due to their particular requirements, dynamic nature and resource limitations. In this research they proposed a novel in-network knowledge discovery approach that provides outlier detection in WSNs. Zhao et al. [10], numerous applications requires the management of spatial data i.e. data related to space. Increasingly large amount of data are obtained from satellite images, x-ray, crystallography or other automatic equipment. Therefore, automated knowledge discovery becomes more and more important in spatial database. The task considered in this paper is class identification i.e. the grouping of the objects of a database into meaningful subclasses. In this paper they used distance based and density based algorithms in which k-medoid and DBSCAN algorithms. First determines k-representatives minimizing the objective function. Second, assign each object to the clusters with its representative "closest" to the considered object. Through density based notion they formalize an intuitive notion of "clusters" and "noise" in a database D of points of some k-dimensional space S. The key idea behind is that for each point of a cluster the neighbourhood of a given radius has to contain at least a minimum no. of points i.e. the density in the neighbourhood has to exceed some threshold. Wang and Su [3] proposed an improved version of K-Means algorithm.

Fu et al. [9], The enlarging volumes of information emerging by the progress of technology, makes clustering of very large scale of data a challenging task. In order to deal with the problem, many researchers try to design efficient parallel clustering algorithms. In this paper, adapted K-Means algorithm in MapReduce framework which is implemented by hadoop to make the clustering method applicable to large scale data. By applying proper <key, value> pairs, the proposed algorithm can be parallel executed effectively.
Firstly it randomly selects *k* objects from the whole objects which represents initial clusters centre. Each remaining object is assign to the cluster to which it is the most similar, based on the distance between the object and the cluster centre. The new mean for each cluster is then calculated. This process iterates until the criterion function converges.

Map-function: The input dataset is stored on HDFS as a sequence file of <key, value> pairs, each of which represents a record in the dataset. The key is the offset of these records in bytes, and the value is a string of the content of this record.

Combine- function: After each map task we apply a combiner to combine the intermediate data of the same map task. Combine function contains partial sum of the values of the points assigned to the same cluster.

Reduce- function: The input of the reduce function is the data obtained from the combine function of each host. Reduce function contains sum of all the samples and compute the total no. of samples assigned to the same clusters.
Zhao et al [6] proposed an efficient parallel k-means clustering algorithm based on MapReduce and their study confirmed that it scales well and can handle large datasets efficiently.

4. METHODOLOGY

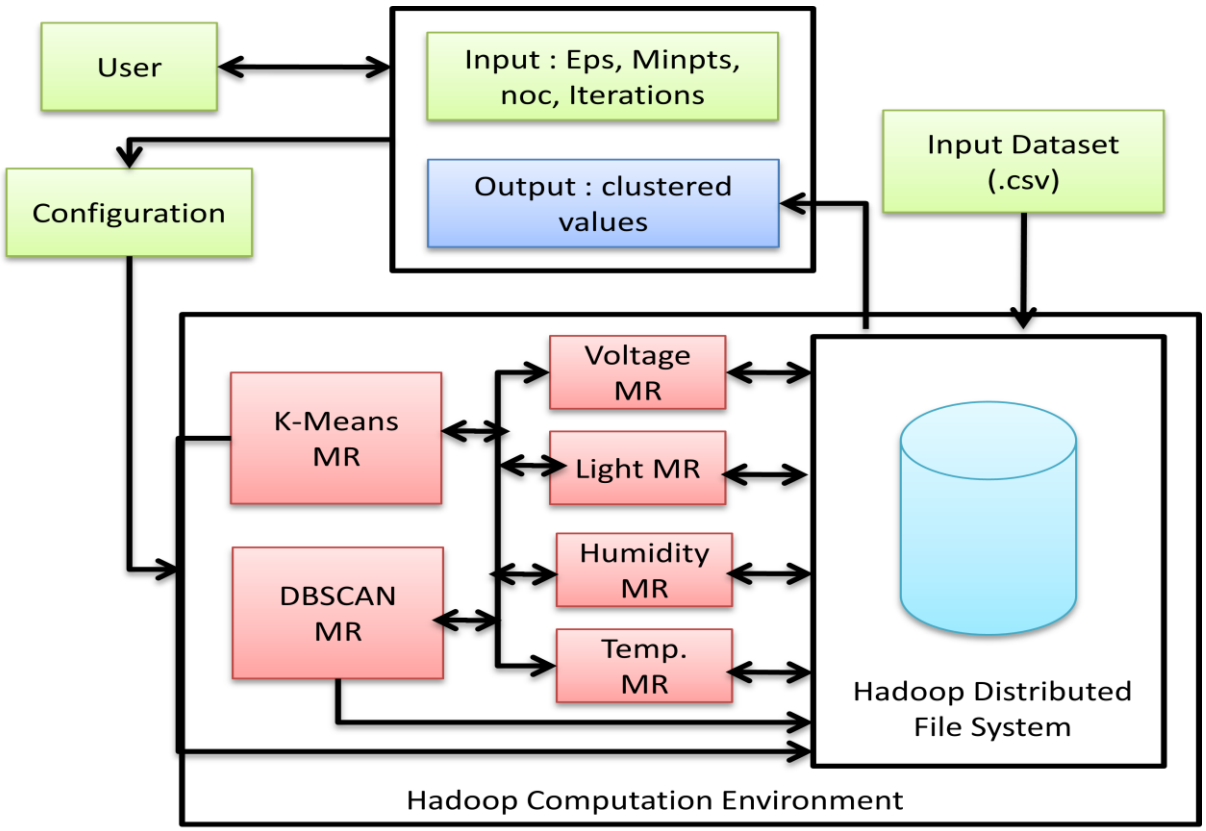

Fig. 1 : Our methodology for Outlier detection

The Block representation consist the major components of proposed solution. Here, a brief explanation against the understanding of block representation is cited below:

User gives minimum value for cluster and epsilon parameters as input and load Intel dataset to HDFS. It supports .csv format and upload file into input folder of HDFS. Initially, any one Map and Reduce function (MR) from voltage, humidity, light and temperature has been called with DBSCAN MR to prepare the clusters and store on slave machines. Configuration components help to establish synchronization between DBSCAN MR and HDFS with input parameters. Similar to above steps, K-Means MR is also involved with voltage, humidity, light and temperature MR to obtain the outliers and rewrite to HDFS. At last HDFS generates results and forward to command user interface in terms of clustered value. A computation time for all MR has been evaluated on single machine and multimode machine to observe the performance of proposed solution. Finally, all the results are compared with performance of Weka tool and simple java applications.

5. IMPLEMENTATION & RESULTS

A java based application has been developed to implement DBSCAN and K-means to obtain the outliers of the existing dataset. This application has been executed on simple JVM platform on single node. The objective behind this implementation is to measure the computation time and performance of outlier detection from large data on single matching with traditional technology. Weka tool has been used to explore the knowledge and outliers from selected dataset. Similar to section one DBSCAN and K-means algorithm is implemented to obtain the outliers from the Intel dataset. Although, K-means does not support outlier detection but can help to evaluate the difference between two clustering algorithms in terms of performance and clustering approach. Hadoop Computation environment has been used to implement parallel and distributed processing with high performance objective [4]. Similar with above sections, DBSCAN and K-Means algorithms has been implemented to obtain the clusters and outliers from dataset. It has implemented with single node configuration and multimode configuration. Table 1 and Table 2 shows the results of the experiment performed on intel dataset for outlier detection using K-means and DBScan algorithm by varying different dataset sizes and using Java application, hadoop based single node cluster and multi-node clusters.

Computation Time (in seconds)												
Parm	Voltage						Light					
DSize	DB J	KM J	DB S	KMS	DBM	KM M	DBJ	KMJ	DB S	KMS	DBM	KM M
1K	<1	<1	10	9	9	10	<1	<1	9	10	9	10
10K	<1	<1	10	9	10	10	<1	<1	10	10	10	10
100K	1	<1	10	9	10	10	1	1	10	9	10	9
1M	32	4	35	10	13	13	11	4	17	9	16	15
50M	1180	175	10	10	12	12	436	192	17	9	14	13
100M	--	--	10	10	9	9	--	--	17	9	13	9

Table 1: Computation time for DBScan & KMeans algorithm for voltage & light

Computation Time (in seconds)												
Param	Humidity						Temperature					
DSize	DB J	KMJ	DB S	KMS	DBM	KM M	DBJ	KMJ	DBS	KMS	DBM	KM M
1K	<1	<1	22	9	23	20	<1	<1	10	10	19	17
10K	<1	<1	10	10	20	16	<1	<1	9	10	19	17
100K	1	1	21	10	21	14	1	1	10	10	19	15
1M	16	4	22	10	19	13	15	4.5	21	10	25	20
50M	648	180	21	10	17	9	680	185	21	9	14	13
100M	--	--	21	10	15	10	--	--	21	9	14	9

Table 2: Computation time for DBScan & KMeans algorithm for humidity & temperature

6. DISCUSSION

This work concludes that K-mean is good for clustering purpose but have certain drawbacks like fix number of cluster and incapable for outlier detection. Subsequently, DBScan execution and computation time is similar with K-Means but gives better result in terms of arbitrary shape clusters and outliers detection. Implementation of both algorithms on java platform was efficient solution for small data sample but gives poor results for large data size. In addition, it also gives poor computation time for average data sample. Proposed solution is also implemented and tested with Weka tool and gives similar results with java implementation. However, it was not able to handle large data processing and could not handle dataset beyond 30MB. Implementation of application with single node hadoop computing environment gives better and hurdles results then above two techniques. It is good for large data sample but becomes exhaust for very large data sample due to absence of slave machine. Multiple node Hadoop computation environments conclude that deployment of multiple slave machine help for deployment of multiple mapper function and distribution of data processing on other servers. It concludes that this solution is good for large data sample but performs poorly for small dataset and in comparison to single node because of distribution overhead.

REFERENCES

[1] M. Ester, H.P. Kriegel, J. Sander, X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise",KDD-96 Proceedings, , German, pp.226-231, 1996.

[2] K. Narita, H. Kitagawa, “Outlier Detection for Transaction Databases Using Association Rules”, In Proceedings of the Ninth International Conference on Web-Age Information Management,Washington, pp. 373-380, 2008.

[3] J. Wang, X. Su, "An improved K-Means clustering algorithm," 2011 IEEE 3rd International Conference on Communication Software and Networks, China, pp. 44-46, 2011.

[4] M. Bhandarkar, "MapReduce programming with apache Hadoop", IEEE International Symposium on Parallel & Distributed Processing (IPDPS), Atlanta-GA, pp.1-1, 2010.

[5] M. Ding, L. Zheng, Y. Lu, L. Li, S. Guo, and M. Guo, “More convenient more overhead: the performance evaluation of Hadoop streaming”, In Proceedings of the ACM Symposium on Research in Applied Computation (RACS), USA, pp. 307-313, 2011.

[6] W. Zhao, H. Ma and Q. He, “Parallel K-Means Clustering Based on MapReduce”, Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, Springer Berlin Heidelberg, pp. 674-679, 2009.

[7] Feng Wang, Jie Qiu, Jie Yang, Bo Dong, Xinhui Li, Ying Li, “Hadoop high availability through metadata replication”. In Proceedings of the first international workshop on Cloud data management (CloudDB '09). ACM- USA, pp. 37-44, 2009.

[8] R. Leonardo, F, Cordeiro, "Clustering very large

multi-dimensional datasets with MapReduce", ACM SIGKDD international conference on Knowledge discovery and data mining, USA, pp.690-698, 2011.Y. X. Fu, W. Z. Zhao, H. F. Ma, "Research on Parallel DBSCAN Algorithm Design Based on MapReduce", Advanced Materials Research, Vols.301, Issue.303, pp. 1133-1138, 2011.

[9] W. Zhao, H.Ma, Q. He, "Parallel K-Means Clustering Based on MapReduce", In Proceedings of the 1st International Conference on Cloud Computing , Springer-Verlag, Berlin, pp. 674-679, 2009.

[10] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Sixth symposium on Operating Systems design and implementation (OSDI), San Francisco, CA, pp. 213-220, 2004.

[11] M.F. Hornick, E. Marcadé, S. Venkayala, "Java Data Mining: Strategy, Standard, and Practice: A Practical Guide for Architecture, Design, and Implementation", Morgan Kaufmann, canada,pp.1-544, 2010.

