ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 6, June 2015

Distributed Computing Based Methods for Anomaly Analysis in Large Datasets

Remya G[1], Anuraj Mohan[1]

Assistant Professor, Dept of CSE, NSS College of Engineering, Palakkad, India [1]

Abstract: Anomaly detection is considered as one among the important domain in data mining. Both supervised and unsupervised learning methods are used in anomaly detection task. In this paper emphasis is given to distance based prediction of anomalies. We studied the traditional methods which involves index-based, nested-loop and cell-based approaches towards anomaly detection. As the size of the datasets become very large the task of detecting anomalies becomes computationally complex. Having the push towards big data mining, it will become more necessary to adopt existing anomaly detection algorithms to various distributed computing platforms. Our paper is based on a survey on the different strategies that can be adopted for anomaly analysis using distributed computing techniques. First we studied the concept of anomaly detection solving set, a subset of the input data set representing a model that can be used to predict anomalies. The solving set is defined using necessary number of points that helps in the detection of the top anomalies by taking into consideration only a subset of all the pair wise distances from the data set. Then we analysed the possibility of using Map Reduce framework for performing anomaly analysis. A MapReduce based solving set algorithm for anomaly detection using Hadoop framework is also proposed.

Keywords: Anomaly, Distributed Computing, Map Reduce, Hadoop

I. INTRODUCTION

An anomaly is a data object that is very much different from the compared normal objects as if it were developed using a unnatural methodology. Motivation for anomaly analysis involves fraud detection, customer segmentation, customized marketing, medical treatment etc. Both supervised and unsupervised methods are used in anomaly analysis task. Using supervised learning techniques for anomaly analysis is less efficient as the main challenge involves is the presence of imbalance classes. Unsupervised methods uses the basic principle that an anomaly is expected to be far away from any groups of normal objects. To improve the quality of anomaly detection, one can get help from models for normal objects learned from unsupervised methods which we usually refer as semi-supervised learning. In a proximity based method an object is an anomaly if the nearest neighbors of the object are far away, i.e., the distance of the object is significantly large from the distance between most of the other objects in the same data set. Two major types of proximity-based anomaly detection techniques are distance-based and density-based methods. A top-m distance-based anomaly in a data set is an object having weight greater than the m^{th} largest weight, where the weight of a data set object is computed as the sum of the distances from the object to its m nearest neighbors.

Distributed computing can be defined to the use of distributed systems to solve computing problems. In distributed computing, a problem is divided into many sub tasks, each task can be assigned to one or more computers, which can communicate to each other by message passing and can finally generate a single result. Distributed algorithms are a classification of Parallel algorithms, which are executed concurrently, with different processors executing various parts of the algorithm and the data and operations at one processor, is abstracted from the other. One of the major issues in designing and implementing distributed algorithms is to maintain proper co-ordination between parts of the algorithm and to handle the system failures and unreliable communications media.

Map Reduce [16] is a distributed computing framework developed at Google for the analytics of large heterogeneous data that have been divided over many computers. Its programming model allows the user to neglect about many of the issues associated with distributed computing: splitting up and assigning the input to various systems, scheduling and running computation tasks on the available nodes and coordinating the necessary communication between tasks. Map Reduce framework uses terms of key-value pairs as input, which are generated from an input file by user-configurable rules. Map Reduce uses a very simple programming abstraction, which in its most common form, requires its user to design only two functions - map and reduce. Hadoop is an open source framework which uses the Map Reduce programming model for writing and running distributed applications that process large amounts of data.

II. RELATED WORK

The concept of distance-based anomaly relies on the notion of the neighbourhood of a point, typically, the k nearest neighbours, and has been first introduced by Knorr and Ng [1], [2]. The authors present two algorithms, the first one is a nested loop algorithm that runs in O(kN2) time, while the next one is a cell-based algorithm that has a polynomial time complexity with respect to the number of points of the data set, but exponential with respect to the number of dimensions of the data set. On the other hand, the nested loop approach is impractical when anomalies in large data sets have to be mined. In [3], a new definition of distance-based anomaly that takes into account the whole

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 6, June 2015

neighbourhood by considering, for each point p, the sum of the proximities from its n nearest neighbours, is proposed. An analogous definition of anomaly based on the k-nearest neighbours has been used in [4] for unsupervised anomaly detection to detect intrusions in unlabeled data. A complete discussion on the need, the understanding, and the intentional information of distance-based anomalies, as well as the relevant areas in which the anomaly concept can be applied, can be found in [5], [6] .A parallel version of nested loop approach for anomaly detection is introduced in[7]. Dutta et al. [8] proposed algorithms for the distributed computation of principal components and top-k anomaly detection. The concept of anomaly detection solving set, a subset of the input data set representing a model that can be used to predict anomalies, is defined in [9]. A distributed method for detecting distance-based anomalies in very large data sets is proposed in [11]. The possibility of using Map Reduce parallel programming model in many machine learning and data mining algorithms for big data mining is studied in [12]. A Hadoop Map Reduce based tool for anomaly analysis is implemented by Pranab Ghosh in[13]

A. Definition of the Tasks

In the following, we assume that any data set is a finite subset of a given metric space.
Anomaly Weight: Given an object p∈D, the weight wk (p, D) of p in D is the sum of the distances from p to its k nearest neighbors in D.
Top n Anomalies: Let T be a subset of D having size n. If there not exist objects x ∈ T and y in (D \T) such that wk(y, D)>wk(x, D), then T is said to be the set of the top n anomalies in D. In such a case, w*=minx ∈ t wk(x, D) is said to be the weight of the top nth anomaly, and the objects in are said to be the top n anomalies in D [3].
Anomaly Detection Solving Set [9]: An anomaly detection solving set S is a subset of D such that, for each y E (D \ S), it holds that wk(y, S) ≤ w*, where w* is the weight of the top nth anomaly in D. It is found that a solving set S always defines the set T of the top n anomalies in D and, moreover, it can be used to predict unknown anomalies.

B. Solving Set Algorithm [9]

At each iteration the algorithm compares each object with a selected small subset of the complete data set objects called candidate set, and stores their k nearest neighbours with respect to the candidate set. From these stored neighbours, an upper limit to the actual weight of each object can thus be calculated. The actual weights of the candidate objects will be identified as they are compared with every object in the data set. An object with upper limit weight lower than the n^{th} highest weight associated with a candidate object are called non active as these objects cannot belong to the top-n anomalies, while the others are called active. Random objects are selected as candidates in the beginning. At each subsequent iterations candidate set is built by selecting, among the active objects of the data set not already inserted during the previous iterations and the objects having the highest current weight upper bounds. During the computation, if an object becomes non active, then it need not be considered for adding into the candidate set, because it cannot be an anomaly. As the algorithm progresses new objects, more precise weights are computed and will result in the rise of non-active objects. The algorithm terminates when no more objects have to be observed.

C. Distributed SolvingSet Algorithm [11]

The Distributed Solving Set algorithm adopts the same procedure of the Solving Set algorithm. It consists of a main cycle executed by a controller node, which iteratively performs the following two tasks: 1) the basic computation, which is done by every node in parallel; and 2) the combining of the incomplete results returned by each node after completing its job. The computation is done by estimating of the anomaly weight of each object and of a global lower bound for the weight, below which points are sure to be non-anomalies. Alternate local and global information is considered for iteratively refining the above estimates.
It can be noted while when that several mining algorithms deal with distributed data set by computing local models which are aggregated in a general model as a final step in the controller node, the Distributed Solving Set algorithm is dissimilar, since it computes the true global information through iterations where only selected global data and all the local data are involved.

The basic operation executed at each node consists in the following steps:

1) The current solving set objects are received along with the current lower bound for the weight of the top nth anomaly,
2) Compare them with the local objects.
3) Extract a new set of local candidate objects (the data points with the top weights, according to the current calculations together with the list of local nearest neighbors with respect to the solving set.
4) Determining the number of local active objects, that is the data points which are having weight not smaller than the current lower bound.

The comparison is performed in many distinct cycles, in order to avoid unnecessary computations. The above data are used in the synchronization step by the controller node to generate a new set of global candidates to be used in the following iteration, and for each of them the true list of distances from the nearest neighbors, to compute the new lower bound for the weight.

D. MapReduce model for distance based anomaly analysis[13].

We essentially need to find the k nearest neighbors for a data point and find its total distance to the k nearest neighbors and use that as the anomaly score. A MapReduce programming model can be deployed for this purpose by [13]. The Map and Reduce parts of MapReduce are both defined with respect to data represented as (key, value) pairs. Map function takes as input (key, value) pair and returns a (key, value) pair as output. The logic needed to process the input will be defined inside the map function: Map (k1, v1) → list (k2, v2).The Map operation will be applied to every pair in the input in parallel. Then the Map

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 6, June 2015

Reduce framework internally does a shuffle and sort phase in which the same key from all lists will be grouped together, and one group will be assigned for each key. The Reduce function is then applied is to each such group in parallel, which will finally produce a list of values which belongs to the same domain: Reduce (k2, list (v2)) → list (v3). Each Reduce call will either produces either a value or a null return. The final result will be the consolidated results from all the calls.

Two MapReduce jobs for performing the anomaly detection task is proposed in [13]

The jobs of the task are

1) A MapReduce job to find pair wise distance between all data points.
2) A MapReduce job which finds the k nearest neighbors of an object and find the weight of the object with respect to the k nearest neighbors.

The immediate issue that comes for this MapReduce is how to divide up the work of pairing up entities and calculate the distance for each pair. The idea is to partition each set of objects and pair up the partitions from each object. The partitioning can be done by hashing the object Id.The Id of each object is hashed. For each object type we get a set of hash values. All the possible combination of hash values for the two object types can be taken. For each hash value pair, we pair up the objects from each type falling in the two hash values. The distance computation between objects for each hash value pair is distributed among the reducers. The mapper output key is a function of two hash values. The following function is used to generate the mapper output key

key = (hash(SID) % hashBucketCount) *
hashBucketCount + hash(TID) % hashBucketCount

where SID = Source object ID ,TID = Target object ID and hashBucketCount = Number of hash buckets, which is configurable. To distribute the load uniformly across reducers through Hadoop's default reducer partitioner, hashBucketCount should be chosen properly. For example, if the value chosen is 10, there will be 200 unique values for the key. The values for a given mapper key will contain instances of objects of both types. Since we have to pair up instances of one type entity with instances of another type, we need a way to segregate the instances, so instances of one type appears before the instances of the other type in the list of values when the reducer gets invoked. This will enable easier nested loop join. The key defined earlier will be used as the base part of the key. The key is enhanced so that it will ensure that that the source entities will appear before the target entities. The mapper of Average Distance has the first object ID along with the distance as the key and the distance and the second object ID as the value. A secondary sorting by the distance on the key can be performed. The first object ID is the base part of the key and the distance is the key extension. In the reducer, for a given entity we get all the other objects sorted by distance. The reducer retains the first k neighbouring objects and finds the average distance to them.

There are number of ways to detect anomalies from the result, as follows

1) Select the top n after sorting the result by descending order of the average distance
2) Select all with average distance over a predefined threshold after sorting the result by the descending order of the average distance.

III. MAPREDUCE BASED SOLVING SET ALGORITHM

We propose to modify the solving set algorithm so as to fit into the MapReduce framework of Hadoop. To design the algorithm in Hadoop, the concept of chaining MapReduce jobs is used. Chaining means executing multiple MapReduce jobs one after the other. In this method, there will be multiple stages of MapReduce in which there will be an Initial MapReduce Stage, a number of intermediate stages and a final stage. The numbers of intermediate stages are user controlled.

A. Initial Stage

The initial MapReduce phase takes as input the dataset, outputs a candidate set (initially random).The output from the initial MapReduce phase is fed into the intermediate stage.

B. Intermediate Stage

The intermediate stage contains the following modules

1) A supervisor MapReduce module which takes as input the candidate objects, stores it's true as well as upper bound weights and generates new candidate objects by selecting objects with top n upper bound weight.
2) The Similarity MapReduce module which takes as input, output from supervisor MapReduce module and finds the distance between the candidate set objects and all other objects which is stored in the HDFS.
3) The output from similarity MapReduce module is fed into Compute Weight MapReduce module which finds K nearest neighbors of each candidate objects and thereby calculates its true weight. This module also finds the upper bound weights of all the objects and makes the objects with upper bound weight less than the least true weight, non-active.ie they cannot be considered as anomalies and are excluded in further MapReduce stages. The module emits as output objects along with its weights. In the next intermediate phase the result of ComputeWeight MapReduce module fed as input to the supervisor MapReduce module which updates the weight information which it stored about the candidate objects and emits top n upper bound weighed objects as the output. In the successive intermediate stages new candidates will be generated and the supervisor dynamically changes the candidate objects by comparing the weights. When the supervisor MapReduce module receives the same candidate set as it generated in the previous iteration the algorithm terminates.

C. Final Stage

The output from the final intermediate stage is fed into the final Map Reduce stage which writes the anomaly objects to a file that can be accessed by the user.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 6, June 2015

IV. EXPERIMENTS

The Distributed solving set algorithm is coded in Java and the communications are done through the Java libraries implementing the TCP sockets. As experimental platform, we used 6 workstations, each equipped with a Intel 2 GHz processor and 2 GB of RAM, interconnected by an Ethernet network. The dataset used in the experiment is Poker which is obtained from the real data set Poker Hands, available at UCI repository, by removing the class label. The Poker consists of 1000000 instances of10 attributes. For Map Reduce implementation of anomaly detection a Hadoop cluster consisting of 6 nodes were set up. Experimental results showed that the distributed strategies are more efficient than single node nested loop approaches.

The Map Reduce based strategies holds the following advantages over the distributed solving set algorithm.

1) It can resolve scalability issues that comes with anomaly detection over big data due to use of Hadoop system which is highly scalable
2) The reliability of the system is high as the data is stores in the HDFS

V. CONCLUSION

We aimed at parallelizing various strategies that can are used in anomaly detection task. We first discussed the basic techniques used in distance based anomaly detection. The solving set algorithm and the distributed solving set algorithm for anomaly detection is studied and implemented. The possibility of using MapReduce model using Hadoop for anomaly detection is studied and a nested loop based anomaly detection algorithm is implemented in Hadoop framework. We also propose a MapReduce based solving set algorithm for solving the anomaly detection task. We conclude that the MapReduce based strategies may be more suitable for anomaly detection in big datasets. We aim to learn more machine learning techniques like SVM and parallelize them so as to work with MapReduce model. We also aim to develop distributed algorithms for statistical based and clustering based anomaly detection methods and compare the results with distance based methods.

REFERENCES

[1] E. Knorr and R. Ng, "Algorithms for Mining Distance-Based Outliers in Large Datasets," Proc. Int'l Conf. Very Large Databases (VLDB '98), pp. 392-403, 1998.
[2] E. Knorr, R. Ng, and V. Tucakov, "Distance-Based Outlier: Algorithms and Applications," VLDB J., vol. 8, nos. 3-4, pp. 237-253, 2000.
[3] F.Angiulli and C.Pizzuti, "Outlier Mining in Large High-Dimensional Data Sets," IEEE Trans. Knowledge and Data Eng.,vol. 2, no. 17, pp. 203-215, Feb. 2005.
[4] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data," Applications of Data Mining in Computer Security, Kluwer, 2002.
[5] E. Knorr and R. Ng, "Finding Intentional Knowledge of Distance-Based Outliers," Proc. Int'l Conf. Very Large Databases (VLDB '99), pp. 211-222, 1999.
[6] E. Knorr, R. Ng, and V. Tucakov, "Distance-Based Outlier: Algorithms and Applications," VLDB J., vol. 8, nos. 3-4, pp. 237-253, 2000.
[7] Distributed and Parallel Databases, 12, 5–26, 2002 Kluwer Academic Publishers. Manufactured in The Netherlands. Parallel Mining of Outliers in Large Database.
[8] H. Dutta, C. Giannella, K.D. Borne, and H. Kargupta, "Distributed Top-K Outlier Detection from Astronomy Catalogs Using the DEMAC System," Proc. SIAM Int'l Conf. Data Mining (SDM), 2007.
[9] Angiulli.F, Basta.S, and Pizzuti.C (Feb 2006), "Distance-Based Detection and Prediction of Outliers," IEEE Trans. Knowledge and Data Eng., vol. 18, no. 2, pp. 145-160.
[10] Angiulli.F, Basta.S, Lodi.S, and Sartori.C (2010), "A Distributed Approach to Detect Outliers in very Large Data Sets," Proc. 16th Int'l Euro-Par Conf. Parallel Processing (Euro-Par), pp. 329-34.
[11] Angiulli.F, Basta.S, Lodi.S, and Sartori.C(2013), "A Distributed Strategies for Mining Outliers in Large Data Sets," IEEE Trans. Knowledge and Data Eng.Vol 25.Nov 7,pp 1520-1532
[12] Xindong Wu1, 2, Xingquan Zhu3, Gong-Qing Wu2, Wei Ding, "Data Mining with Big Data". IEEE Trans. Knowledge and Data Eng.jan 2014.pp 97-107
[13] https://github.com/pranab
[14] Apache Hadoop. http://hadoop.apache.org/.
[15] http://archive.ics.uci.edu/ml/datasets/Poker+Hand
[16] Dean, J. and Ghemawat, S. (2004), 'MapReduce: Simplified data processing on large Clusters'. Proceedings of the Sixth Symposium of Operating Systems Design and Implementation (OSDI 2004), CA, USA, pp. 137-150.