Outlier Detection using Distributed Mining Technology in Large Database

Mr.B.Murugananthan, Ms. Ankita Dubey
Assistant Professor (Sr.G), Department of Computer Science and Engineering, SRM University, India
M.Tech Scholar, Department of Computer Science and Engineering, SRM University, India

ABSTRACT *- In many data analysis tasks, a large number of variables are being recorded or sampled. One of the first steps towards obtaining a coherent analysis is the detection of outlaying observations. A distributed approach is presented to detect distance-based outliers, based on the concept of outlier detection solving set. Data objects, which are different from or inconsistent with the remaining set of data, are called outliers. Many data mining techniques exists that attempt to find patterns that occur frequently in the data in which outliers are treated as noise that needs to be removed from a dataset. It is worth to notice that this is a unique peculiarity of distributed data mining technique, since other distributed methods for outlier detection are not able to return a model of the data that can be used for predicting novel outliers. In this paper, we analyze and detect the distance-based outliers in the data set. Priority is given to reduce the processing time of the supervisor node. This is done by simultaneous data computation by local nodes. It also deals with the drawbacks of the centralized system such as complete failure of the system and security.*

Keywords *- Clustering, Distance-based outlier, Data mining, Outlier detection.*

1. Introduction

Data mining, also called as data surfing, Knowledge discovery, knowledge extraction, data/pattern analysis, data archeology, information harvesting, data dredging, and business intelligence, is the process of extraction of interesting, and potentially useful patterns or knowledge from large amount of data.

The application of Data mining can be used to address many real world challenges. It includes variety of fields like, government security, science research, and business. The applications range from insurance, banking, retail, astronomy, medicine, to detection of criminals and terrorists by analyzing the data mined. Some of the earliest successful applications of data mining are marketing research and credit-card-fraud detection. It is achieved by studying a consumer's purchasing behavior. When there is an abnormal pattern in the customer's transaction, we can detect anomaly.

Data mining attempts to discover hidden rules underlying the data in contrast to an expert system which uses traditional methods. Data mining uses different techniques like artificial intelligence, neural networks, and sophisticated statistical tools such as cluster analysis to detect trends, patterns, and relationships, which would have been hard to detect otherwise .

There are many types of data mining techniques; Predictive modeling, Descriptive modeling, Pattern mining, Anomaly detection. Descriptive modeling; also called clustering, divides data into groups. But with clustering the proper groups are not known in advance; instead the patterns discovered by analyzing and understanding the data are used to determine and classify into groups. As an example, an advertiser could analyze a general population in order to classify potential customers into different clusters and then develop separate advertising campaigns targeted to each group. In fraud detection, clustering is used to identify groups of individuals with similar purchasing and expenditure patterns.

Anomaly detection which is another version of clustering—that uses to find data instances that are unfamiliar and do not fit any established pattern. An example of anomaly detection is Fraud detection. Fraud detection is a problem for predictive modeling, because it has low accuracy and can quickly become out of date. But in anomaly detection concentration on modeling is done in order to understand what is normal behavior. Which gives us an idea about unusual transactions. Anomaly detection is also used for intrusion detection.

Traditional data mining algorithms are designed assuming that data are centralized in a single memory hierarchy. Moreover, these algorithms are designed to be executed by a single processor. There by, many researchers proposes parallel data mining (PDM) and distributed data mining (DDM) algorithms[1.].

The effect of constant factors and decrease execution times could be reduced dramatically by parallel processing. When we mine data from distributed sources, the fragmentation of data set is done in many local data sets, which are generated at different nodes of a network. The advantages of such solutions are simplicity and feasibility with established technology. Moreover, the discovery of anomalies must be carried out timely, because preventive actions must be taken as early as possible.

Data objects that do not comply with the general behavior or model of the data, and are grossly different form or inconsistent with the remaining set of data are called outliers. In the case of fraud detection, the outliers may be of particular interest because they indicate fraudulent activity. Hence, outlier detection and analysis is an interesting data mining task. Which is also referred to as outlier mining or outlier analysis.

Outlier detection approaches focuses on discovering patterns that occur infrequently in the data, as opposed to many traditional data mining techniques that attempt to find patterns that occur frequently in the data. Outliers are frequently treated as noise that needs to be removed from a dataset in order for a specific model or algorithm to succeed.

In this work, we follow the definition given as a top-n distance based outlier in a data set is an object having weight not smaller than the nth largest weight, where the weight of a data set object is computed as the sum of the distances from the object to its k nearest neighbors.

2. Related work

The outlier detection task can be very time consuming and recently there has been an increasing interest in parallel/distributed methods for outlier detection. Parallel versions, called PENL, of the basic NL algorithm have been proposed by Hung and Cheung [6.]. PENL is based on a definition of outlier employed a distance-based outlier is a point. Bay's algorithm, which is based on a definition of distance-based outlier coherent with the one used here. Moreover, this parallel version does not deal with the drawbacks of the centralized version, which is sensitive to the order and to the distribution of the data set.

Defining outliers by their distance to neighboring data points has been shown to be an effective non-parametric approach to outlier detection. A fast algorithm for mining distance-based outliers exist [2.]. As we process more data points, the algorithm finds more extreme outliers, and the cut-off increases giving us improved pruning efficiency. The state-of-the art distance-based outlier detection algorithm, uses the NL algorithm with a pre-processed data set. The algorithm facilitates fast convergence to a point's approximate nearest neighbors. RBRP improves upon the scaling behavior of the state-of-the-art by employing an efficient pre-processing step that allows for fast determination of approximate nearest neighbors. We validated its scaling behavior on several real and synthetic data sets. RBRP consistently outperforms ORCA, the state-of-the-art distance-based outlier detection algorithm, often by an order of magnitude.

A fast outlier detection strategy for distributed high-dimensional data sets with mixed attributes relies on an intuitive anomaly score for categorical attributes and at the same time efficiently handles sparse continuous data [3.]; this method is fast, scalable, and uses two data passes. Datasets that are distributed amongst different geographical sites which results as low communication and synchronization overhead. It is experimentally shown to be highly scalable with the number of points and the number of attributes in the dataset.

A new definition of distance-based outlier and an algorithm, called HilOut [4.], designed to efficiently detect the top n outliers of a large and high-dimensional data set. The algorithm consists of two phases: the first provides an approximate solution with temporal cost and spatial cost and the second calculates the exact solution with a final scan. Given a set of N data points or objects and the number n of expected outliers, find the top n objects that are considerably dissimilar or inconsistent with respect to the remaining data.

A distributed method (called Distributed SolvingSet) [5.] to detect distance-based anomalies is presented. Anomaly detection refers to the task of identifying abnormal or inconsistent patterns from a data set. While outliers may seem as undesirable entities in a data set, but identifying them gives us many important information.

In the existing system, the request from query node is sent to the super node (Global candidate). Super node is having its own data set, where it searches for the data. then sends the query to other sub nodes (Local nodes) These sub nodes

are also having their own data sets, in which they search for the data searched by super node. After completion of the search, all the nodes detect outliers and then removes it from the data set and create one index of the data. Later when data computation is done all the nodes communicate to each other and match their data set to remove the duplicate data from the data set. For which the LazyDistributedSolvingSet algorithm [1.] is used. And then they send it back to the super node which then forwards the whole search result to the user.

2.1 LazyDistributedSolvingSet Algorithm

A generic approach to Lazy distributed set solving [1.] that permits exploitation of the increasing availability of clustered and/or multi-processor machines. From the analysis accomplished in the preceding section, it follows that the total amount TD of data transferred linearly increases with the number of employed nodes. Though in some scenarios the linear dependence of the amount of data transferred may have little impact on the execution time and on the speedup of the method and, also, on the communication channel load, this kind of dependence is in general undesirable, since in some other scenarios relative performances could sensibly deteriorate when the number of nodes increases. In order to remove this dependency, we describe in this section a variant of the basic DistributedSolvingSet algorithm [5.] previously introduced. With this aim, an incremental procedure is pursued. Several iterations are accomplished: during each of them only a subset of the nearest neighbors' distances [2.], starting from the smallest ones, is sent by each local node to the supervisor node. At each iteration, the supervisor node collects the additional distances, puts them together with the previously received ones, and checks whether additional distances are needed in order to determine the true weight associated with the candidate objects.

2.1.1 ALGORITHM

1. DSS = $\emptyset$;
2. OUT = $\emptyset$;
3. d = $\sum_{i=1}^{l} d_I$;
4. for each node $N_i \in N$
5. NodeInit($\left\lceil m\frac{di}{d}\right\rceil$,$C_i$);
6. C = $\bigcup_{i=0}^{l} C_i$;
7. act = d;
8. minOUT = 0;
9. while (C ≠ $\emptyset$) {
10. DSS = DSS $\bigcup C$;
11. For each node $N_i \in N$
12. NodeComp (minOut, C , act, $\lceil\frac{k}{l}\rceil$+1, $LNNC_i$, LC_i, act_i);
13. act =$\sum_{i=1}^{l} act_i$;
14. repeat
15. for each q ϵ C {
16. NNC[q] =get_k_NNC(NNC[q] $\bigcup(\bigcup_{i=1}^{l} LNNCi[q])$);
17. u_NNC[q] = 0;
18. nodes[q] = $\emptyset$;
19. If $\exists$ j s.t. $\min_i$ { last($LNNC_i$[q])} = NNC[q][j] {
20. u_NNC[q] = k – j;
21. cur_last[q] = NNC[q][k];
22. nodes[q] =$\bigcup_{i=1}^{l} N_i$ s.t. last ($LNNC_i$[q]) $\in$ NNC[q];
23. }
24. }
25. For each node $N_i \in U_q \in_C$ nodes[q]
26. NodeReq (u_NNC, cur_last, nodes, $LNNC_i$);
27. }
28. Until $U_q \in_C$ nodes[q] = $\emptyset$;
29. For each q $\in$ C
30. updateMax(Out,(q,Sum (NNC[q])));
31. minOUT =Min(OUT);
32. C = $\emptyset$;
33. For each p $\in \bigcup_{i=1}^{l} LC_i$
34. C = C U {p};
35. }

end

procedure NodeComp$_i$(minOUT, C, act, k_0, $LNNC_i$, LC_i , act_i) {

….

33. $LNNC_i$ = sort($LNNC_i$);
34. $rLNNC_i$ = get_l($LNNC_i$, k – k_0);
35. $LNNC_i$ = get_f($LNNC_i$, k_0);
36. store(act_i , C_i, $rLNNC_i$);

}

Procedure NodeReq$_i$(u_NNC,cur_last,nodes, $LNNC_i$){

37. load ($rLNNC_i$);
38. for each (q such that $N_i \in$ nodes[q]){
39. $LNNC_i$[q] = get_f($rLNNC_i$[q], $\lceil\frac{u\_NNC[q]}{|nodes[q]|}\rceil$ +1, cur_last[q]);
40. $rLNNC_i$[q] = get_l($rLNNC_i$[q], |$rLNNC_i$[q]| - |$LNNC_i$[q]|);
41. }
42. store($rLNNC_i$);

}

3. Proposed work

In the previous system, for every keyword being searched, we used to get a whole large set of result, which includes the result in which we are interested and also similar domain.

As shown in the fig.3.1, the system computes the true global model through iterations where only selected global data and all the local data are involved. The computation is driven by the estimate of the outlier weight of each data point and of a global lower bound for the weight, below which it might be non-outliers.

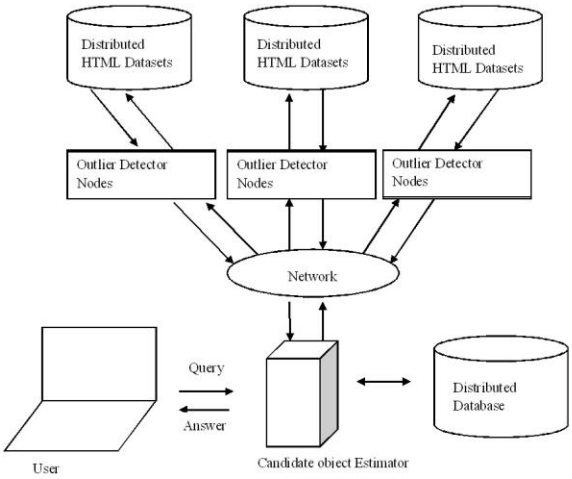


figure 3.1 system architecture

The above estimates are iteratively refined by considering alternatively local and global information. Finally, this global candidates is broadcast to all nodes and used by each node to goes over its own dataset and detect outliers.

Number of the active data points in the local data set, is set to the local data set size. We assume that each node owns a unique subset of points (rows) of the entire dataset (horizontally distributed). The main concept is that each node computes its own local model using only its own dataset. Each node also adds up the continuous vectors of all data points corresponding to all categorical sets.

Each node obtains the pruned candidate sets by deleting the sets that are frequent, and the sets that are infrequent and have infrequent subsets. The cut-off threshold in unlikely to converge to such a large value quickly, not just for a uniformly distributed data set, but for any arbitrary data set. Distributed Solving set delivers near-linear scaling behaviour only when the cut-off distance can quickly converge to a large value. This can occur only when the data set has a large number of outlying points.

After each node has computed and shared all necessary values, each node independently goes over its local dataset and selects the outliers. k nearest neighbor' distances of the candidate objects inserts into the data structure. The Greatest distances in neighbor nodes to be possibly returned in subsequent calls of the procedure NodeReq, leaves in neighbor nodes only the k^{th} smallest distances there included, and stores in the local memory the number of active objects, the candidates coming from the local node, and the data structures. Then outliers removed from neighbor nodes datasets by executing the function.

3.1 One Shot Method

To obtain a "one-shot" merging method of the local information with some approximation guarantees, the Key Operation is to repeatedly forming clusters from the nearest neighbors. It is designed to handle very large data sets. It assumes that clusters are normally distributed around a centroid in a Euclidean space. One shot method includes following things:

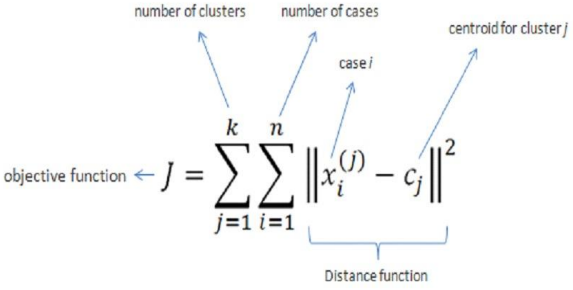
- Removing outliers from the local candidate's extraction and the results will be displayed in global candidate.
- In this method, we use semantic search for searching a word and a user must be authorized then only the request will passed to the local candidate's.
- We have to search a word from the dataset and the same word is used for clustering process. Clustering is the process of partitioning into clusters. This is done such that patterns in the same cluster are alike and patterns belonging to two different clusters are different.
- We use k-means clustering algorithm because it reduces time complexity and improve the performance of clustering using centroids.

For example, user can search for a word "Apple" shows the result relevant to Food items-apple and Electronic device-apple. From these results, we have to cluster both the results separately.

3.1.1 K-means Clustering

K-Means is simplest learning algorithm for clustering analysis. The aim of K-Means algorithm is to find the best division of n observations in k clusters, so that the distance between the cluster's members and its corresponding centroid, representative of the cluster, is minimized. Here, the goal is to partition the n observation into k sets S_i, i=1, 2, ..., k in order to minimize the within-cluster sum of squares (WCSS),defined:

$$\sum_{j=1}^{k}\sum_{i=1}^{n} \| x_i^j - c_j \|^2$$



where, term $\| x_i^j - c_j \|$ provides the distance between an entity point and the cluster's centroid. Given below are the steps of the algorithm:

1. Set the centroid of the initial group. This step can be done by different methodologies. One among this is to assign random values for the centroids of all groups. Another method is to use the values of K different entities as being the centroids.
2. Next we assign each entity to the cluster that has the nearest centroid. To find the cluster with the most similar centroid, it must calculate the distance between all the entities and each centroid.
3. Next we recalculate the values of the centroids. Now the centroid's fields are updated. This is done by taking the average of the values of the entities' attributes that are part of the cluster.
4. Repeat steps 2 and 3 iteratively until entities can no longer change groups.

The K-Means is a computationally efficient greedy technique, being the most popular representative-based clustering algorithm.

Algorithm 1: K-Means Algorithm

Input: $E = \{e_1, e_2, \ldots, e_n\}$ (set of entities to be clustered)
k (number of clusters)
$MaxIters$ (limit of iterations)
Output: $C = \{c_1, c_2, \ldots, c_k\}$ (set of cluster centroids)
$L = \{l(e) \mid e = 1, 2, \ldots, n\}$ (set of cluster labels of E)

```
foreach c_i ∈ C do
    c_i ← e_j ∈ E (e.g. random selection)
end
foreach e_i ∈ E do
    l(e_i) ← argminDistance(e_i, c_j) j ∈ {1...k}
end

changed ← false;
iter ← 0;
repeat
    foreach c_i ∈ C do
        UpdateCluster(c_i);
    end
    foreach e_i ∈ E do
        minDist ← argminDistance(e_i, c_j) j ∈ {1...k};
        if minDist ≠ l(e_i) then
            l(e_i) ← minDist;
            changed ← true;
        end
    end
    iter++;
until changed = true and iter ≤ MaxIters ;
```

Algorithm 1: K-means clustering

The phrase “one-shot learning” has been used to describe our ability – as humans – to correctly recognize and understand objects (e.g. images, words) based on very few training examples. Successful one-shot learning requires the learner to incorporate strong contextual information into the learning algorithm (e.g. information on object categories for image classification or “function words” used in conjunction with a novel word and referent in word-learning). Variants of oneshot learning have been widely studied in literature on cognitive science, language acquisition (where a great deal of relevant work has been conducted on “fast-mapping”), and computer vision. Many recent statistical approaches to one-shot learning, which have been shown to perform effectively in a variety of examples, rely on hierarchical Bayesian models. In this article, we propose a simple latent factor model for one-shot learning with continuous outcomes. We propose effective methods for one-shot learning in this setting, and derive risk approximations that are informative in an asymptotic regime where the number of training examples n is fixed (e.g. n = 2) and the number of contextual features for each example d diverges. These approximations provide insight into the significance of various parameters that are relevant for oneshot learning. One important feature of the proposed one-shot setting

is that prediction becomes "easier" when d is large, i.e. when more context is provided.

4. Conclusion

The proposed solution produces an overall speedup close to linear w.r.t. the number of computing nodes. When the number of nodes increase, the proposed system scales accordingly in the case of computation of coordinator node and data transmission. We have seen a virtually optimal speed-up on standard and usual benchmarks, that is, the speed-up nearly matched the number of processes or processors. The main application of this work reflects in parallel data mining (PDM) and faster search engine and a more accurate result. The future work of this work include semantic search.

References

[1.] Fabrizio Angiulli, Senior Member, IEEE, Stefano Basta, Stefano Lodi, and Claudio Sartori, "Distributed Strategies for Mining Outliers in Large Data Sets" , IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 7, july 2013.

[2.] A. Ghoting, S. Parthasarathy, and M.E. Otey, "Fast Mining of Distance-Based Outliers in High-Dimensional Datasets," Data Mining Knowledge Discovery, vol. 16, no. 3, pp. 349-364, 2008.

[3.] A. Koufakou and M. Georgiopoulos, "A Fast Outlier Detection Strategy for Distributed High-Dimensional Data Sets with Mixed Attributes," Data Mining Knowledge Discovery, vol. 20, pp. 259-289, 2009.

[4.] 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.

[5.] Srinivasa Rao, Divakar Ch , Govardhan A , " An Optimized Approach for Discovering Anomalies in Distributed Data Mining ", IJARCSSE , vol. 3,no. 3,march 2013.

[6.] Edward Hung, David W. Cheung, "Parallel Mining of Outliers in Large Database", Distributed and Parallel Databases, 12, 5–26, 2002.