International Journal of Information Technology and Knowledge Management
January-June 2011, Volume 4, No. 1, pp. 275-278

AN EFFICIENT PARTITIONING ALGORITHM TO FIND UN-EXPECTED BEHAVIOURAL DATA POINTS

K.Subramanian[1] & E.Ramaraj[2]

In Data Mining an outlier is an exception that deviate much from other observations in the multidimensional space. There are various approaches to detect outliers in the data set. Many different approaches are proposed by the researchers. In this paper three distance based outlier detection algorithms are taken and compared with the proposed algorithm.
Kewords: Data Mining – Multidimensional Space – Outlier

1. Introduction

Outlier detection is to the problem of finding patterns in data that do not match to expected behaviour. Outlier detection finds extensive use in a wide range of applications such as fraud detection in credit card usage, insurance or health care, intrusion detection in cyber security, fault detection in safety critical systems and military surveillance of enemy activities. This importance of outlier detection is due to the fact that outliers in data translate to significant (and often critical) actionable information in a wide range of application domains. For example, an anomalous traffic pattern in a computer network could mean that a hacked computer is sending out sensitive data to an illegal destination.

There are several approaches to outlier detection. The first approach is model-based outlier detection, where the data is assumed to follow a parametric (typically univariate) distribution [2].Such approaches do not work well in even moderately high dimensional spaces and finding the right model is often a difficult task. To overcome these limitations, researchers have turned to various non-parametric approaches that use a point's distance to its nearest neighbour as a measure of unusualness [1] [5] [7]. The following is a popular definition of distance-based outliers:"Outliers are the top n data points whose distance to the kth nearest neighbour is greatest [11]". While distance-based outlier detection has proven to be useful, the process continues to be time consuming.

Most of the existing algorithms have handled high dimensional datasets with low performances. The proposed algorithm is to overcome the problem and provide better performance than existing algorithms.

[1]Lecturer, J.J Arts and Science College, Pudukkottai, Tamilnadu

[2]Technology Advisor, Madurai Kamaraj University, Madurai, Tamilnadu

Email: [1]subjjcit@gmail.com, [2]eramaraj62@gmail.com

2. Existing Works

Many algorithms have been proposed to detect outliers. Knorr etal., [1] proposed the Nested-Loop (NL) algorithm to find outliers. In this algorithm, each data point in the data set is compared to various points in the data set to determine its M nearest neighbours. NL has quadratic complexity that makes all pair wise distance computations between the data points. The author suggested the use of spatial indexing structures such as R-trees and X-trees to find the nearest neighbours of each candidate point. It works well for low dimensional data sets. The index structures may lead to poor performance as the dimensionality increases [1] [2].

Bay and Schwabacher [3] presented an algorithm. It is based on NL and uses randomization and pruning rule with near linear time performance. The algorithm depends on the data ordering, which, as the authors in the paper state, can lead to a poor performance. The algorithm performs poorly when the data does not contain outliers.

Amol Ghoting etal., [4] presented an RBRP (Recursive Binning and Re-Projection) algorithm for fast mining of distance-based outliers. The algorithm facilitated fast convergence to a point's approximate nearest neighbours. Only a point's approximate nearest neighbours (and not its nearest neighbours) are needed for efficient distance-based outlier detection. The algorithm scales well to high-dimensional data sets with millions of data points, and outperforms the state-of-the-art distance-based outlier detection algorithms, often by over an order of magnitude. The algorithm scaled log-linearly as a function of the number of data points, and linearly as a function of the number of dimensions. In k-means algorithm partitions are generated by clustering the data. Each of the clusters can constitute a partition. However, this process requires specification of the number of clusters, and does not guarantee equal-frequency partitioning.

3. PROPOSED ALGORITHM

The proposed algorithm is divided into two phases. The first phase is called discard. This phase is used to partition the data set into partitions so that points that are close to each other in space are likely to be assigned to the same partitions. At each stage in the recursion, the proposed algorithms iteratively partition the data into 'g' partitions. This iterative partitioning is akin to the partitioning step employed in the k-means [8] algorithm. Essentially, it starts with 'g' random centers, and assigns each point to its closest center, creating 'g' partitions. Next, it finds 'g' centers for these 'g' partitions, and continues iteratively for a fixed number of iterations. Once it has finished with these iterations, for each of these partitions, it proceeds recursively if the size of the partition is greater than a user-defined threshold (partitionsize).

```
Procedure discard ( partition-size, g, n, D)
 begin
 c = {c1, c2, ...., cg}
 p = {p1, p2, ....., pg}
 for n iterations
 all p={}
 for each d in D
 j = Closest(c, d)
 Insert(d, j)
 end for
 c = {}
 recomputecenters(c, p)
 end for
 for each pi in p
 if size of pi > partition-size
 discard(partition-size, g, n, pi)
 else
 reorganize data points in pi
 add pi to B
 end if
 end for
 end
```

In procedure discard, the following inputs are required: partition-size, the maximum size of a partition; 'g', the number of partitions; n means the no. of iterations; D, the data points. Let c and p is assigned 'g' center points and partition of D respectively. The d is a data object in D. The procedure closest(c,d) is to return the index of the nearest elements in c to d. The insert(d,j) is to insert points d in j^{th} partition in p. In recomputeCenters(c, p), it inserts 'g' centers of partitions in p into c. The B is an output, that is, set of partitions.

The second phase process is called find-outlier. The given set D consists of 'n' number of data points and 'm' is number of data points required in each circle. Let icount be the number of points inside the small, the ocount be the number of points inside the ocount but not inside the icount. The data points can be easily observed. The proposed algorithm has some advantages over nested looping in avoiding distance calculation and gaining computational savings because there is a provision for an early exit, from the i-Loop, on satisfying the condition icount >m. It is possible that some points may not be tested. There is a provision for another early exit on the condition ocount + icount < m without the need to perform any distance calculations. In the worst case, distance calculation should be performed for those points that are stored in the "OList". This makes the proposed algorithm better than the nested looping algorithm.

```
Procedure find_outlier(D, n)
begin
 icount = 0
 ocount = 0
 for i = 1, …, n
 if x is inside the small square then
 icount = icount + 1
 if icount > m then
 exit from i loop (q is not an outlier, go to Next i)
 else
 if x is inside the large square then
 ocount = ocount + 1
 store x in NewArray
 next i
 if ocount + icount < m then
 add q in OList
go to end (q is an outlier)
 else
 for each point in NewArray, do
 if dist(q, x) ≤ d then
 add 1 to icount
 if icount < m then
 add q in OList
go to end (q is an outlier)
end
```

4. EXPERIMENTAL AND COMPARATIVE STUDY

This section investigates the efficiency of the new proposed algorithm, compared with NL and RBRP, when applied on different data sets to detect outliers. The proposed algorithm generates outputs that are identical to the outputs of NL. The performance of the proposed algorithm is reported in terms of CPU time. In these tests, three data sets which are obtained from the UCI Repository of Machine Learning Databases [5] have been tested. These are core histogram, letter and segmentation datasets.

Table 1
UCI Repository of Machine Learning Databases

S.No	Datasets	Dimension	No. of data points
1.	Segmentation	19	2310
2.	Letter	16	20000
3.	Core Histogram	32	68040

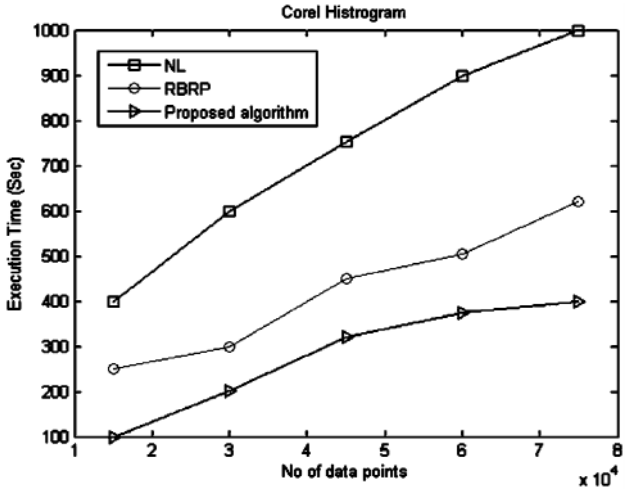

Fig. 1: Comparison of Execution Time between NL, RBRP and Proposed Algorithm using Corel Histogram Dataset

Table 2
Eexecution Time between NL, RBRP and Proposed Algorithm using Corel Histogram Dataset

Algorithm	No. of Data points				
	15000	30000	45000	60000	75000
NL	400	600	752	900	1000
RBRP	250	300	450	505	621
Proposed	100	202	320	375	400

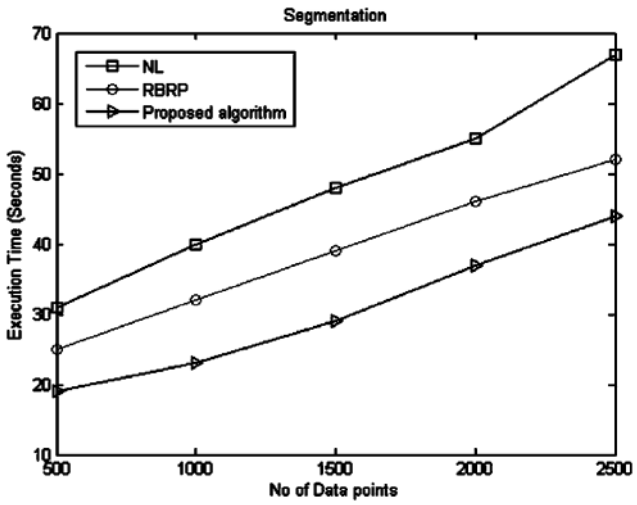

Fig. 2: Comparison of Execution Time Between NL, RBRP and Proposed Algorithm using Segmentation Dataset

Table 3
Execution Time between NL, RBRP and Proposed Algorithm using Segmentaion Dataset

Algorithm	No. of Data points				
	500	1000	1500	2000	2500
NL	31	40	48	55	67
RBRP	25	32	39	46	52
Proposed	19	23	29	37	44

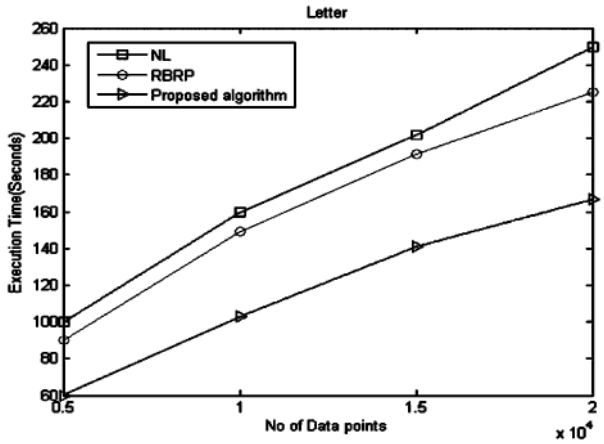

Fig. 3: Comparison of Execution Time between NL, RBRP and Proposed Algorithm using Letter Dataset

Table 4
Execution Time between NL, RBRP and Proposed Algorithm using Letter Dataset

Algorithm	No. of Data points			
	5000	10000	15000	20000
NL	100	160	202	250
RBRP	90	149	191	225
Proposed	60	103	141	167

5. CONCLUSION

The distance-based outlier detection methods differentiate an object as an outlier on the basis of the distance between the data points and its nearest neighbours. Despite the fact that they are simple to implement, they suffer exponential computational growth as most of them are founded on the principle that for each point q, there may be a need to calculate the distances between q and all data points in the dataset. The computational complexity is directly proportional to both the dimensionality of the data and the number of objects. In this paper, the proposed algorithm produces the output with fewer distance calculations than NL and RBRP algorithms. It is important to note that the proposed algorithm performs more comparison operations than NL and RBRP. The test results present a significant increase in efficiency over NL when applied to three bench-marked data sets.

REFERENCES

[1] Knorr, E., R. Ng, and V. Tucakov, 2000, "Distance based Outliers: Algorithms and Applications", VLDB Journal, 8(3-4), 237-253.

[2] Ramaswami, S., R. Rastogi and K. Shim, 2000, "Efficient Algorithm for Mining Outliers from Large Data Sets", Proc. ACM SIGMOD, pp. 427-438.

[3] Bay, S. and M. Schwabacher, 2003, "Mining Distance-based Outliers in Near Linear Time with Randomization and a Simple Pruning Rule", Proc. 9th ACM SIGKDD Int. Conf. Knowledge Discovey and Data Mining, ACM Press, pp. 29-38.

[4] Amol Ghoting, Srinivasan Parthasarathy, and Matthew Eric Otey, Fast Mining of Distance-Based Outliers in High-Dimensional Datasets.

[5] Blake, C. L. & C. J. Merz, 1998, UCI Repository of Machine Learning Databases, http://www.ics.uci.edu/mlearn/MLRepository.html, University of California, Irvine, Department of Information and Computer Sciences.

[6] E. Knorr and R. Ng., "Finding Intensional Knowledge of Distance-based Outliers", In VLDB, 1999.

[7] S. Ramaswamy, R. Rastogi, and K. Shim., "E_cient Algorithms for Mining Outliers from Large Datasets". In SIGMOD, 2000.