Journal of Computer Science 3 (12): 944-947, 2007
ISSN 1549-3636
© 2007 Science Publications

A Fast Distance-Based Algorithm to Detect Outliers

Moh'd Belal Al- Zoubi and Nadim Obeid
Computer Information Systems Department, University of Jordan, Amman 11942, Jordan

Abstract: A fast distance-based algorithm for outlier detection will be proposed. It was found that the proposed algorithm reduced the number of distance calculations compared to the nested-loop algorithm. Test results were performed on different well-known data sets. The results showed that the proposed algorithm gave a reasonable amount of CPU time saving.

Keywords: Distance-based outliers, outlier detection, data mining

INTRODUCTION

There is a need for pre-processing of the raw data in many fields, such as data mining, information retrieval, machine learning and pattern recognition. Zhang et. al [1] argue for the importance of data pre-processing and present the following reasons: (1) real world data is impure; (2) high performance data mining systems require high quality data and (3) quality data yields high quality patterns. Therefore, developing efficient data-preprocessing techniques is a critical task that requires considerable research efforts.

Data pre-processing involves many tasks including detecting outliers, recovering incomplete data and correcting errors. These tasks often present themselves as less glamorous. However, they are more critical than further steps in many application areas [1].

Outlier detection is an important pre-processing task. It has many practical applications in several areas, such as fraud detection[2], identifying computer network intrusions and bottlenecks [3], criminal activities in E-commerce and detection of suspicious activities [4]. Knorr and NG [5] defined outliers as those data points (vectors) with values different from those of the remaining set of data. Different approaches have been proposed to detect outliers, and a good review can be found in [6].

One of the most popular approaches for detecting outliers is the distance-based approach [7-12]. In this approach, the distance of a point from its k nearest points (or neighbors) is calculated. If the neighboring points are relatively close, then the point is considered normal. However, if the neighboring points are far away, then the point is considered an outlier. One of the advantages of this approach is that no explicit distribution needs to be defined to detect outliers. Moreover, this approach can be applied to any feature space for which a distance measure can be defined [7-9]. Commonly, the Euclidean distance is used as the distance function. A detailed discussion on the usefulness, the meaning, and the knowledge of distance-based outliers with a description of the real-life application domains for which this notion of outlier is relevant, can be found in [7, 13, 14].

Given a distance measure on a feature space, the distance-based approach for outlier detection is defined as follows[5]. A point q in a data set is an outlier with respect to the parameters M and d, if there are less than M points within the distance d from q, where the values of M and d, are decided by the user. The problem with this approach is that it suffers from exponential computational growth as it is founded on the principle that for each point q, there may be a need to calculate the distances between q and all data points (objects) in the dataset. The computational complexity is directly proportional to both the dimensionality of the data and the number of objects.

Hence, it is beneficial to look for techniques that can manage to produce outputs identical to the existing ones, but can efficiently decide, at least, on the distances between points without calculation (i.e., with lower runtime) [2, 7] and/or with fewer distance calculations (i.e., distance calculations are performed between fewer points). In this paper, we propose an algorithm to detect outliers in a shorter time than most of the existing outliers-detecting algorithms. Given a

Corresponding Author: Moh'd Belal Al- Zoubi, Computer Information Systems Dept., University of Jordan, Amman 11942, Jordan, Tel: +962-5355000, Fax: +962-5354070

query point *q*, the proposed algorithm satisfies (a) as it can decide on the status of a large number of points, with respect to *q*, without the need to perform arithmetic operations. It also satisfies (b) because it only needs to perform an actual distance calculation on points that have been determined to fall within a particular area around *q*.

In the rest of this paper, we briefly present some of the existing approaches proposed to find outliers, then we present our algorithm and show how it may save calculations. After that, we present a section where we investigate the efficiency of the proposed algorithm and Nested Loop (NL) algorithm by running both algorithms on different data sets to detect outliers

RELATED WORK

Many algorithms have been proposed to find outliers efficiently. Knorr, Ng and Tucakov, in [7], propose the Nested-Loop (NL) algorithm to find outliers. In NL, each data point in the data set is compared to each point in the data set to determine its *M* nearest neighbors. NL has quadratic complexity as we must make all pairwise distance computations between the data points. Knorr et al. also suggested the use of spatial indexing structures such as R-trees and X-trees to find the nearest neighbors of each candidate point. This suggestion may work well for low-dimensional data sets. However, index structures can lead to poor performance as the dimensionality increases [8, 12].

In [8], Ramaswamy et al. modified the definition of outliers introduced in [7] and consider as outliers the top n points whose distances to their k^{th} nearest neighbors are the greatest. To detect outliers, a partition-based algorithm is presented that partitions the input points using a clustering algorithm and, then, prunes those partitions that cannot contain outliers. One shortcoming of this definition is that it only considers the distance to the k^{th} neighbor and ignores information about closer points [14].

Bay and Schwabacher [12] present an algorithm, which is based on NL and uses randomization and pruning rule with near linear time performance. However, the algorithm depends on the data ordering, which, as the authors in the paper state, can lead to a poor performance. In addition, the algorithm can perform poorly when the data does not contain outliers. In this paper, we propose a new algorithm to speed up NL. The proposed algorithm is presented in the next section.

PROPOSED ALGORITHM

Given a query point, *q*, if a circle (in two dimensional space), centered at *q*, is drawn with radius *d*, as shown in Fig. 1, then we count the number of data points inside the circle. If the count is less than *M*, then the point is considered an outlier, otherwise, the point is normal (not outlier). This can be done by checking whether each data point is inside (intersect) or outside the circle.

However, finding the points that are inside a circle may require a large amount of distance calculations. Instead, for each query point *q*, we first test the points inside the square that touches the interior of the circle, as shown in Fig. 2 (shaded area) and count the number of points (*countSmall*) inside the square. If *countSmal*l is greater than *M*, then *q* is not an outlier, and there is no need to test the rest of the points in the dataset. Otherwise, we count the number of points (*countLarge*) inside the shaded area of the larger square touching the outer side of the circle, as shown in Fig.3 and store these points (e.g., their coordinates) in an array for later processing. If the value of *countSmal*l, plus the value of *countLarge* is less than *M*, then *q* is an outlier. Otherwise, we perform distance calculations for the points that are stored in the array only.

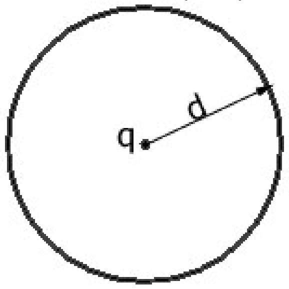

Fig. 1: A circle with radius *d*, centered at the query point *q*

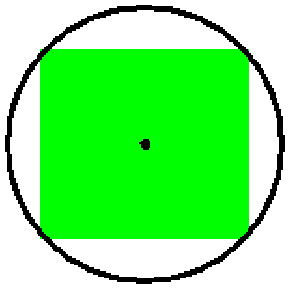

Fig. 2: A square (shaded) that touches the interior of the circle

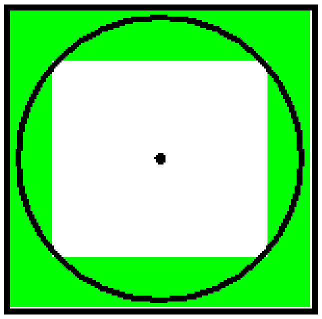
Fig. 3: A larger square touching the outer side of the circle

The proposed algorithms is as follows:
Given a set X of N number of data points, let *countSmall* be the number of points inside the small (shaded) square (shown in Fig. 2). Let *countLarge* be the number of points inside the large square but not inside the small square (shaded area in Fig. 3). Let *NewArray* be an empty array of type X.

```
For each query point, q, do
  Begin
    countSmall = 0
    countLarge = 0
    // The i-Loop
    For i = 1, …, N //for each point, xi
      If x is inside the small square
        then
          countSmall = countSmall + 1
      If countSmall > M
        then
          Exit from i loop (q is not an outlier,
            go to Next i)
        Else
          If x is inside the large square
            then
              countLarge = countLarge + 1
              store x in NewArray
      Next i
    If countLarge + countSmall < M
      then
        go to End (q is an outlier)
      Else
        // Perform distance calculations for the
        // points in the NewArray.
        For each point in NewArray, do
          If dist(q, x) <= d
            then
              add 1 to countSmall
    If countSmall < M then go to End (q is an
      outlier)
  End
```

It can be easily observed that the proposed algorithm has some advantages over the NL in avoiding distance calculation and gaining computational savings because:

1. There is a provision for an early exit, from the *i-Loop,* on satisfying the condition *"countSmall > M"*. It is possible that some *points* may not be tested.
2. There is a provision for another early exit on the condition *"countLarge + countSmall < M"* without the need to perform any distance calculations.
3. Distance calculation, in the worst case, is needed to be performed for those points that are stored in the *"NewArray"* only. This makes the proposed algorithm better than the NL algorithm.

RESULTS AND DISCUSSION

In this section, we will investigate the efficiency of the new proposed algorithm, compared with NL, when applied on different data sets to detect outliers. The proposed algorithm generates outputs that are identical to the outputs of NL. The choice of M and d is based on the heuristics discussed in [7]. The performance of the proposed algorithm is reported in terms of CPU time and percentage of savings compared to NL.

In our tests, five data sets which are obtained from the UCI Repository of Machine Learning Databases [15] have been tested. These are Breast, Letter, Pima, Segmentation and Wine data sets. The description of these data sets is shown in Table 1. N is the number of points, and D represents the dimensionality of data.

Table 1. Description of datasets

DataSet	N	D
Breast	699	10
Letter	20000	16
Pima	768	8
Segmentation	2310	19
Wine	178	13

For each data set, the value of M is the same for both the proposed algorithm and NL. The same applies for the values of d.

Table 2 shows the CPU run time for the Proposed algorithm and NL. It shows that the performance of the proposed algorithm has a significant speed improvement over NL in all cases.

Table 2: The CPU run time (in seconds) of the proposed algorithm and NL.

Data Set	NL	New
Breast	3.6	2.7
Letter	4147	2913
Pima	3.7	2.3
Segmentation	67	41
Wine	0.45	0.31

Table 3 shows the percentage CPU time savings obtained from the proposed algorithm compared to NL. It shows that good CPU time savings is achieved and, on average, up to 32.5% of the CPU time can be saved.

Table 3: percentage savings of the proposed algorithm compared to NL.

Data Set	%Savings
Breast	25
Letter	30
Pima	38
Segmentation	39
Wine	31

It is clear from the two tables above that the proposed algorithm gains significant CPU time savings over NL.

CONCLUSION

Distance-based outlier detection methods distinguish an object as an outlier on the basis of the distance between it and its nearest neighbors. 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 (object) q, there may be a need to calculate the distances between q and all data points (objects) in the dataset. The computational complexity is directly proportional to both the dimensionality of the data and the number of objects. In this paper, we have proposed an algorithm that produces the same output as NL with fewer distance calculations. It is important to note that the proposed algorithm performs more comparison operations than NL. However, tests have shown that these operations are trivial tasks for most compilers, and thus, they are less computationally demanding than arithmetic operations. The test results present a significant increase in efficiency over NL when applied to five bench-marked data sets.

REFERENCES

1. Zhang, S., C. Zhang and Q. Yang, 2003. Data Preparation for Data Mining. Applied Artificial Intelligence, 17(5-6): 375-381.
2. Bolton, R. and D. J. Hand, 2002. Statistical Fraud Detection: A Review, Statistical Science, 17(3): 235-255.
3. Lane, T. and C. E. Brodley. 1999. Temporal Sequence Learning and Data Reduction for Anomaly Detection, ACM Transactions on Information and System Security, 2(3): 295-331.
4. Chiu, A. and A. Fu, 2003. Enhancement on Local Outlier Detection. 7th International Database Engineering and Application Symposium (IDEAS03), pp. 298-307.
5. Knorr, E. and R. Ng, 1998. Algorithms for Mining Distance-based Outliers in Large Data Sets, Proc. the 24th International Conference on Very Large Databases (VLDB), pp. 392-403.
6. Hodge, V. and J. Austin, 2004. A Survey of Outlier Detection Methodologies, Artificial Intelligence Review, 22: 85–126.
7. Knorr, E., R. Ng, and V. Tucakov, 2000. Distance-based Outliers: Algorithms and Applications. VLDB Journal, 8(3-4): 237-253.
8. Ramaswami, S., R. Rastogi and K. Shim, 2000. Efficient Algorithm for Mining Outliers from Large Data Sets. Proc. ACM SIGMOD, pp. 427-438.
9. Angiulli, F. and C. Pizzuti, Outlier Mining in Large High-Dimensional Data Sets, 2005. IEEE Transactions on Knowledge and Data Engineering, 17(2): 203-215.
10. Acuna E. and C. Rodriguez, 2004. A Meta Analysis Study of Outlier Detection Methods in Classification, Technical paper, Department of Mathematics, University of Puerto Rico at Mayaguez, available at academic.uprm.edu/~eacuna/paperout.pdf. In proceedings IPSI 2004, Venice.
11. Shrestha, M., H. Hamilton and Y. Yao, 2006. The PDD Framework for Detecting Categories of Peculiar Data. Proc. 6th International Conf. on Data Mining (ICDM06), pp. 562-571.
12. 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.
13. Eskin, E., A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, 2002. A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data," Applications of Data Mining in Computer Security, Kluwer.
14. Angiulli, F., S Basta, and Pizzuti, 2006. Distance-Based Detection and Prediction of Outliers,. IEEE Transactions on Knowledge and Data Engineering, 18(2): 203-215..
15. 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.