Reduction in Execution Cost of *k*-Nearest Neighbor Based Outlier Detection Method

Sanjoli Poddar(✉) and Bidyut Kr. Patra(✉)

National Institute of Technology Rourkela, Rourkela, Odisha, India
sanjoli0511@gmail.com, patrabk@nitrkl.ac.in

Abstract. Outlier detection is an important task as it leads to the discovery of critical information in a variety of the application domains. The variants of *k*-nearest neighbor based outlier detection method have been successfully applied over decades. However, these approaches have high execution time as they compute a score (known as outlier score) for each data point. In this paper, we propose a method to reduce the execution time of *k*-nearest neighbor based algorithms. Proposed method quickly identifies the data points which are normal and therefore outlier score for such points need not be computed in further processing. The proposed method is generic and can be applied to improve the execution efficiency of many density-based and distance-based outlier detection methods. Proposed work is compared with other existing methods and the result shows that the proposed work outperforms other methods.

Keywords: Density based outlier detection method
k-nearest neighbor · LOF · Execution time

1 Introduction

Outliers are the observations which deviates so much from the other observations as to arouse suspicions that it was generated from a different mechanism [1]. Efficient mining of the data is very important as it finds its application in various domains such as credit card fraud analysis, intrusion detection system, medical field, marketing etc. Both supervised and unsupervised learning methods are used to identify the outliers [8]. In unsupervised learning, no prior knowledge about the data set is known. This makes the unsupervised outlier detection methods very popular over supervised approach. Popular unsupervised algorithms include clustering techniques, distance-based outlier, density based-outlier and *k*-nearest neighbor based methods.

Among the unsupervised learning algorithms, density based outlier detection methods are very popular and efficient for identifying the outliers. The main idea behind density based methods is to compute outlier score for each data point and declare the points with high scores as outlier points. In order to compute the outlier score, *k*-nearest neighbors of each data point are computed and subsequently use their statistics according to the individual algorithm. Popular density based

D. Ghosh et al. (Eds.): ICMC 2018, CCIS 834, pp. 53–60, 2018.
https://doi.org/10.1007/978-981-13-0023-3_6

outlier detection method includes LOF (local outlier factor) [5], COF (connectivity outlier factor) [11], INFLO. The distance based method is also found to be using k-nearest neighbor information [9].

The outlier score corresponding to normal data point is of limited use especially when the objective is to mine out the outliers. As density and distance based approaches calculates the outlier score for every the data point, it makes the method inefficient. The problem increases in many folds with increasing the size of the data set.

In this paper, we propose a method to improve the execution time of k-nearest neighbor based outlier detection methods. In proposed method, we introduce a novel measure termed as *devToMean* to identify the normal points for which outlier scores are not required to compute in further processing. The novel measure ensures that none of the outlier points is identified as normal points. Having filtered these normal points, we only compute outlier score of the remaining data points based on the individual algorithm. The experiments are carried out both on synthetic as well as real datasets and the results show a significant improvement in execution cost over the other methods.

The rest of the paper is organized as follows. Section 2 describes state-of-the-art works in this direction. We describe proposed work in Sect. 3. Experimental results and analysis are reported in Sect. 4. We conclude our paper in Sect. 5.

2 Related Work

The broad application of the outlier detection has made the literature very rich. Widely popular outlier detection techniques include statistical approach, distance-based, density-based, rule-based, neighborhood based, *etc.* [8].

Distance and density based outlier detection techniques are widely used when no prior knowledge about the dataset is known unlike statistical approach. Knorr and Ng [2] proposed first distance-based outlier detection technique. It uses the distance parameter to find the outliers. The notion of the distance based algorithm is further extended to k-nearest neighbor distance or statistics [9]. In [9], it uses the relative location of an object to its neighbor to determine the degree to which object deviates from its neighbors. Thus the objects with the higher LDOF score (Eq. 1) are regarded as outliers.

$$LDOF_k(x_p) = \frac{d_{x_p}}{D_{x_p}} \tag{1}$$

where, $D_{x_p} = \frac{1}{k(k-1)} \sum_{x_i, x_{i'} \in kNN(x_p)} dist(x_i, x_{i'})$, d_{x_p} is the average distance from point x_p to all its k nearest neighbors, kNN is the k-nearest neighbors of the point x_p excluding the point x_p itself.

Density based outlier detection techniques use local information/statistics of each data point for computing outlier score of the point. Some of the significant works done in this area includes LOF (local outlier factor) [5], COF (connectivity based outlier factor) [11], INFLO [7], *etc.* In the popular LOF method [5],

local reachibility density of a point is computed using the statistics of k-nearest neighbors of the point. Finally, it computes a score called *lof* which is the average of the ratio of local reachibility density of a point p to the density of the point p's nearest neighbors. If the factor lof is close to 1 then the point is considered as normal. If the value of the lof $\gg$1, then it is declared as an outlier. INFLO [7] outlier detection approach addresses the problem of LOF in a dataset with variable density over the feature space. It considers both k-neighbors and reverse k-neighbors statistics while computing outlier score [3,4,6,10].

All the methods discussed above are proven to be very powerful and efficient in term of finding outliers. However, these approaches have high execution time as they compute outlier score for each data point. This problem becomes severe with the increase of size of the dataset. Few methods are reported to improve the execution time but they are specific to particular density based outlier detection technique while compromising the accuracy (precision) of the method.

Some of the work done to improve upon the density based outlier detection methods include LOF' [6]. Authors argued that the MinPts-dist is sufficient to find the density of a point. Subsequently, the basic formula (Eq. 1) for computing the outlierness of a point is altered. In FastLOF [10], author argue that a good estimation is fair enough for normal data points but precise nearest neighbors are required for the outliers. To reduce the execution time of k-nearest neighbor search, data set is randomly divided into data chunks and search is performed only within a single chunk for each point in it. Subsequently, approach takes a decision that which data points can be further considered to find the outliers and other are safely pruned (removed). However, in this case all outlier points may not be detected. Other pruning strategies are also developed to reduce the execution time of density and distance based approaches [3,4]. Basic idea of these research is to identify the normal points and prune them for further processing. In [4], k-means clustering method is applied over the dataset and subsequently, pruning strategy is applied to individual cluster. The points within a cluster are pruned (removed) if they locate close to the centriod (within the radius of the cluster). Finally, LDOF is applied to remaining points in the dataset. However, the genuine outlier point located close to the centriod of a cluster can be considered as normal point and it can be pruned in this approach. Another major drawback of the pruning approach is that pruning a normal point may change the characteristics of its neighbors (*i.e., normal to outlier point*). Therefore, it increases the false positive rate. Reduction of execution time of these approaches is achieved at the cost of accuracy of outlier detection mechanism.

3 Proposed Work

We address the problem of computation overhead involved in the methods discussed in previous section in a novel way. In those methods, outlier score is calculated for each data point. Intuitively, outlier scores corresponding to the normal points are not of significant use and hence this calculation can be avoided. In this proposed approach, we quickly identify most of the normal points and computation of the outlier score for these points are not performed in subsequent

step. It can be noted that we do not prune these points unlike pruning strategy. Therefore, accuracy of the individual method is not compromised in this approach and it leads to reduce the execution cost of the density and distance based outlier detection methods.

Our proposed approach works in two phases. In the first phase, a linear clustering method (k-means) is applied to partition the data into a number of chunks (cluster) and centroid point of each of the cluster is computed. We aim to mark all normal data points from each of these cluster. It can be intuitively said that the normal points lie in dense region, hence its deviation (density deviation) from its neighbors is small, whereas outliers lie in the sparse region and its deviation from its neighbors would be more as compared to normal points. We use this assumption in order to identify the normal points within each cluster. We introduce a metric termed as *devToMean* for each point within a cluster. Let C_i be a cluster and m_{C_i} be the mean of C_i. The *devToMean* of a point $x \in C_i$ is the ratio of deviation of the point x from mean point to the average deviation of its neighbors from the mean of the cluster. This is computed using the following Eq. 2.

$$devToMean(x) = \frac{||x - m_{C_i}||}{\frac{1}{k}\sum_{x_i \in k-NN(x)} ||x_i - m_{C_i}||} \tag{2}$$

The *devToMean* determines how much an object deviates from the mean of the cluster to which it belongs with respect to its neighbors. If this deviation of a point is similar to that of its neighbors, then the value of *devToMean* is close to 1 and the point is considered as a normal point. The value of $devToMean$ for outlier point can be high ($>> 1$) (outlier point far away from mean point) or close to 0 (outlier point close to the mean).

We mark the normal points ($devToMean \approx 1$) and avoid computation of outlier score for these points in next phase. The marked points are not pruned (removed) from the dataset. Remaining unmarked points are sorted based on their *devToMean* values. We apply 10 percentile rule to find probable outliers in the dataset. We select the *top ten (*10*) percentile* and *bottom ten (*10*) percentile* of these sorted unmarked points as probable outliers and investigate them in the next phase. The idea of this rule is that there are few outlier points compared to the normal points and the value of $devToMean$ for genuine outlier is either close to 0 or very high ($>> 1$).

The second stage involves the calculation of the outlier score of the points obtained using 10 percent rule. For computing the outlier score, one can use any popular distance or density based approach discussed in Sect. 2. While computing the k-nearest neighbor statistics of these selected points, all data points are used including the marked normal points. Therefore, accuracy of the outlier detection method is not compromised and results obtained by our approach will be the same as that of the original approach applied on the same dataset. The proposed approach is depicted in Algorithm 1. Finally, a ranking list is made for all the outlier score and top-n outliers are selected.

Algorithm 1. DevToMean outlier detection

Input: D, *num-clust*, k
Output: *outlier-score*
1: $Clusters \leftarrow k\text{-}meansClustering(D, num\text{-}clust)$
2: **for** each $c_i \in Clusters$ **do**
3: **for** each $p_j \in c_i$ **do**
4: Compute $devToMean(p_j)$
5: Mark the point p_j if $devToMean(p_j) = 1$.
6: **end for**
7: **end for**
8: $POutlier \leftarrow$ Filter the unmarked points by applying 10 percentile rule.
9: **for** each $p_i \in POutlier$ **do**
10: Compute outlier score of p_i using entire dataset D.
11: **end for**

As the outlier score corresponding to a very less number of the data point is calculated, the execution time of the proposed algorithm is quite less as compared to the time taken by the original algorithm. The number of clusters to be provided in the first phase is very important factor. The number of clusters should be such that the size of the cluster is neither too big nor too small. If the size of a cluster is too small, then the genuine outlier point might get overlooked as it would contain the value of *devToMean* factor close to 1. Also, if the size of the cluster is too large, then the reduction in time complexity would be quite small as large volume of data set would be examined to calculate the value of *devToMean* factor. Thus, specifying the appropriate number of clusters according to the size of the data set is very important. For testing purpose, we consider that the minimum size of the cluster to be 100.

4 Experimental Results

The proposed method is tested on synthetic as well as real data set. The synthetic data set was uniformly distribution within a region. We also injected few outliers to the dataset. We took one classification dataset named *Cover Type Data* from UCI machine learning repository and converted into One class data with few injection of outlier points from other classes.

We introduce a metric termed as *speed up factor* S which measures the percentage of decrease in the execution time of the proposed method from the execution time of the original approach. The metric is normalized by the maximum decrease in execution time over various input sizes of a dataset. Let t^m_{dev} and t^m_o be the execution time of proposed approach and original approach while both of them applied on a subset of size m of a dataset, respectively. The speed up factor S_m is computed as follows.

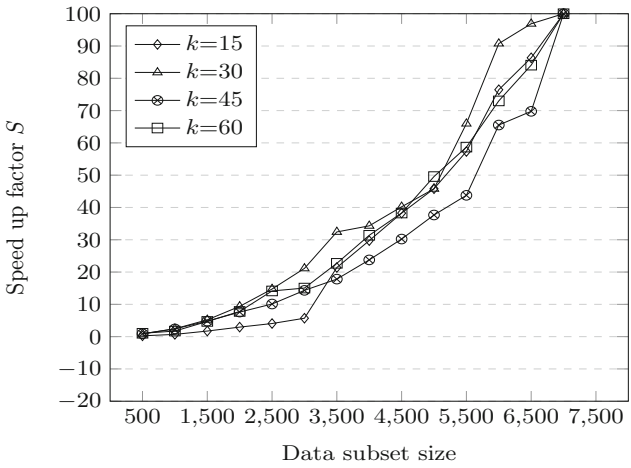

Fig. 1. Speedup factor S with varying data set and parameter k for LOF

$$S_m = \frac{t_o^m - t_{dev}^m}{\max_l\{t_o^l - t_{dev}^l\}} \times 100 \tag{3}$$

where, l is the size of a data subset. The minimum value of $S_m = 0$, when proposed approach takes exactly the same time ($t_o^m = t_{dev}^m$) as that of the original approach.

We speed LOF (density based outlier detection algorithm) up using our approach and speed up factor of the proposed approach is plotted in varying data size of the synthetic dataset in Fig. 1. It can be easily inferred from the plot that the proposed method's efficiency increases with increase in size of the data set in terms of execution cost. For the considered size of the data set, the reduction in time is more than 50% for higher values of k. We achieved a significant reduction in execution time of the LOF method over increasing the size of the dataset. This shows that proposed approach is effective in large size data. The popular distance based outlier detection method LDOF [9] speed-ed up using proposed approach and reported in Fig. 2. Similar trends are observed.

Few works are reported to reduce the execution time of the density based outlier detection techniques using pruning strategies while some of them modified the method for finding outlier detection method [4,6,10]. We compare our proposed method with FastLOF which belongs to first category on real Cover Type dataset. Results are recorded in Fig. 3. The results clearly show that our proposed method outperforms FastLOF. It can be noted that comparison with other pruning based approaches discussed in Sect. 2 are not reported here as these methods cannot produce exactly the same detection accuracy as that of the original approaches.

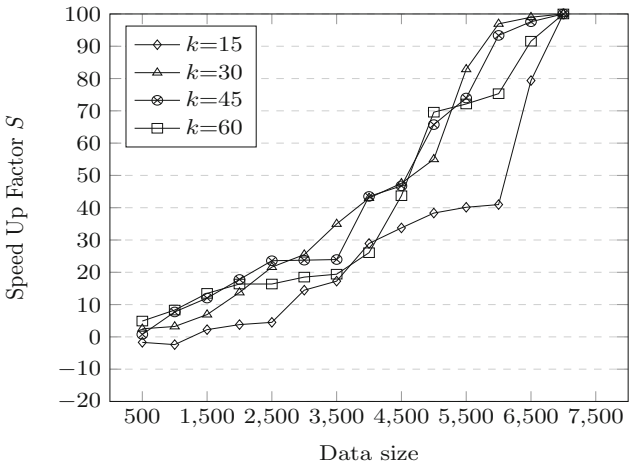

Fig. 2. Efficiency factor S with varying data set and parameter k for LDOF

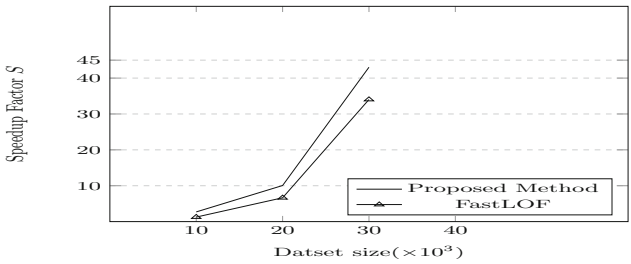

Fig. 3. Comparison of the proposed approach with FastLOF [10] with $k = 70$.

5 Conclusion

In this paper, we proposed a framework to speed up a set of popular outlier detection methods which compute outlier score for each data point using k-nearest neighbor statistics. We introduced a metric $devToMean$ which quickly identifies normal points and computation of outlier scores for these points are avoided in decision making. It is observed from experimental results that the proposed framework is very effective for large dataset and for grater value of the parameter k. In future, we can further reduce the execution time in speeding up the identification process of normal points ($devToMean$).

References

1. Hawkins, D.M.: Identification of Outliers, vol. 11. Chapman and Hall, London (1980)
2. Knorr, E.M., Ng, R.T.: A unified notion of outliers: properties and computation. In: Proceedings of the Third International Conference on Knowledge Discovery and Data Mining 1997 (KDD 1997), pp. 219–222 (1997)

3. Pamula, R.: Data Pruning Based Outlier Detection (Doctoral Dissertation) (2015). http://gyan.iitg.ernet.in/handle/123456789/631
4. Pamula, R., Deka, J.K., Nandi, S.: An outlier detection method based on clustering. In: Proceeding of International Conference on Emerging Applications of Information Technology, Kolkata, India (2011)
5. Breunig, M.M., Kriegel, H.-P., Ng, R.T, Sander, J.: LOF: identifying density-based local outliers. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data 2000 (SIGMOD 2000), pp. 93–104. ACM (2000)
6. Chiu, A.L., Fu, A.W.: Enhancements on local outlier detection. In: Proceedings of Seventh International Conference on Database Engineering and Applications Symposium, pp. 298–307. IEEE (2003)
7. Jin, W., Tung, A.K.H., Han, J., Wang, W.: Ranking outliers using symmetric neighborhood relationship. In: Ng, W.-K., Kitsuregawa, M., Li, J., Chang, K. (eds.) PAKDD 2006. LNCS (LNAI), vol. 3918, pp. 577–593. Springer, Heidelberg (2006). https://doi.org/10.1007/11731139_68
8. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. (CSUR) **41**(3), 15 (2009)
9. Zhang, K., Hutter, M., Jin, H.: A new local distance-based outlier detection approach for scattered real-world data. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS (LNAI), vol. 5476, pp. 813–822. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01307-2_84
10. Goldstein, M.: FastLOF: an expectation-maximization based local outlier detection algorithm. In: Proceeding of 21st International Conference on Pattern Recognition 2012 (ICPR 2012), pp. 2282–2285 (2012)
11. Tang, J., Chen, Z., Fu, A.W., Cheung, D.W.: Enhancing effectiveness of outlier detections for low density patterns. In: Chen, M.-S., Yu, P.S., Liu, B. (eds.) PAKDD 2002. LNCS (LNAI), vol. 2336, pp. 535–548. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-47887-6_53