Outlier Detection

March 2nd, 2021

Marco Ligia
Technological University Dublin
Ireland
B00139079@mytudublin.ie

Abstract

This paper focuses on distance-based outliers detection methods, reviewing basic principle of the classical approach, and one of its improvements that was developed for detecting outliers in very large datasets – sequential algorithm iOrca. The latter is using an indexing technique that makes the mining of outliers in sufficiently large data sets up to an order of magnitude faster compare to traditional approach that requires $O(n^2)$ computation time.

Keywords: Outliers, distance-based methods, iOrca algorithm, indexing technique, very large datasets.

1. Introduction

Outlier is the term used to denote an item that differs significantly from the overall average in a set or combination of data. One of the most important question in the problem of identifying outliers is an understanding of why the outlier should be detected. In this sense, the background of detecting outliers becomes more important than the method of identification itself [1].
The question of the best method to approach an outlier comes to mind as soon as we start dealing with the outlier identification problem, but there is no a certain rule that guarantees that a specific method is the best to detect an outlier. Firstly, it is important to identify the reason why of identifying the outlier [2]. So, the background of detecting outliers helps in defining the method itself.
Nonetheless, there are a number of methods to approach the outliers:

- **Sorting and Plotting.** Sorting the data is an easy but efficient way to spot uncommon values. Simple sorting of data sheet for each variable would make identification of unusually high or low values easier.
- **Statistical tests.** As a rule, statistical tests are used for individual features and extreme values are captured (Extreme-Value Analysis). For example, Z-value or Kurtosis measure can be used to identify if the dataset contains outliers. Here Kurtosis is the average of the Z-scores, each taken to the 4th power. Kurtosis is a measure of whether the data are heavy-tailed or light-tailed relative to a normal distribution. That is, data sets with high kurtosis tend to have heavy tails, or outliers. Data sets with low kurtosis tend to have light tails, or lack of outliers. The Z-values are typically used to introduce the cut-offs for outliers.
- Z-value: $Z_i = \frac{|x_i - \mu|}{\sigma}$; kurtosis $C = \frac{1}{n}\sum_{i=1}^{n} Z_i^4$.

- **Model tests.** The idea is to build a model that describes the data (for example, linear regression model). Points that strongly deviate from the model (at which the model is very wrong) are anomalies, or outliers.
- **Machine learning methods.** Many machine learning techniques show poor performance if outliers are not considered. To avoid this kind of issue it is possible dropping them from the data, fixed threshold values at some reasonable point, or manipulating the data. But in turn, quality of many such methods, for example the Clustering method is measured by its ability to detect some or all of the hidden patterns. Clustering can also be used to detect anomalies [3].
- **Iterative methods.** Methods that consist of iterations, at each of which a group of "especially suspicious objects" is removed. For example, in the n-dimensional feature space, you can remove the convex hull of our points-objects, considering its representatives as outliers. As a rule, the methods of this group are quite time consuming.
- **Metric methods.** Based on the number of publications, these are the most popular methods among researchers. They postulate the existence of some metric in the object space, which helps to find outliers/anomalies. It is intuitively clear that an outlier has few neighbours, while a typical point has many. Therefore, a good measure of anomaly can be, for example, the “distance to the k-th neighbour” (as in the Local Outlier Factor method), the density in the proximity of a point, etc. Specific metrics can be used in these methods, such as Mahalanobis distance. The latter parameter is typically used if you are dealing with detecting multivariate outliers. This is because the Mahalanobis is based on the data distribution obtained in observations [4], [5].

2. Distance-based outlier detection method.

Distance-based outlier detection method belongs to the last group of methods from the previous chapter, the Metric methods. This method could be seen as reviewing the neighbourhood of a data point, which is defined by a given radius (distance). A data point can be considered as outlier when its neighbourhood does not have enough other objects.

Distance-based outlier detection is one of the most popular algorithms in the field of unsupervised learning [3]. This technique has different versions that differ on the measures used, such as the number of nearest neighbours, radius and thresholds.

The outputs are outliers that can be thought of as labels or scores (based on neighbours and distance).

Approaching this problem more formally, then first it is defined an anomaly score for every object *y* in the data D as the distance to the k-th closest neighbour of y in D or, it can be defined as sum of the distances of the k closest neighbours of y in D, where D is distance. The anomaly detection issue can be formalised as: predetermined parameters t and k, identify the top t objects in the dataset regarding their anomaly scores [6].

In this form the statement of the outlier detection task was used in many research fields to find anomalies such as Seismology, astronomy, cosmonautics and more [6].

However, the issue of distance-based outlier detection has not an easy solution that is efficient for big datasets because of the possible quadratic time complexity. The traditional nested loop distance-based algorithm requires $O(n^2)$ (with n = |D|) computation time. This is infeasible for large datasets. Because of it, distance-based methods have a number of “improvements”, such as **Orca** method, which keep the t-th biggest outlier score and set it as a cut off threshold. In [5], it is presented an idea of mapping the existing data in order to split them into multiple cells first, called hyper rectangles. Not whole dataset was stored. instead, only the corners of the minimum bounding

rectangles (MBR were preserved to save the disc space. In the algorithm, all MBRs were pruned every time the check for condition on outliers was due. This was done because MBRs cannot contain the nearest neighbours of the test point. However, the complexity of this approach increased exponentially with the dimension of the dataset [6]. This leaves the approach inefficient for large datasets again.

3. Improvement of the existing algorithm – iORCA – fast outlier detection using indexing

In the Orca algorithm mentioned in the previous section, the pruning rate is achieved by carefully selecting the "cut-off threshold". When this threshold increases slowly, it produces repetitive comparisons for inliers. In order to improve this aspect of iOrca (indexed Orca), the algorithm need to be such as:

- the threshold is updated faster,
- the structure of the data is organized in a way that the nearest neighbours are determined for each object in a fixed time,
- there are no nonessential disk accesses executions, to make sure the previous steps are completed
- the index is obtained rapidly and it is not necessary to keep the entire data in memory.

For the cut-off updates to run faster, the outliers should be processed as early as possible. Hence, in the proposed indexing algorithm, firstly a random object R from D as a reference object is selected, and then is computed the distance of all the other objects in distance D from R. Subsequently, the objects in the dataset are set based on the increasing distance from R.

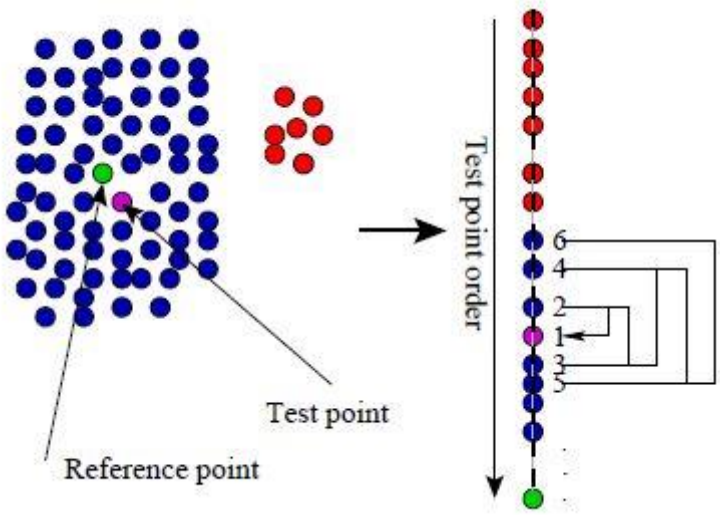

Figure 1: Explanation of the indexing process.

In Figure 1, the figure on the left shows a dataset, where the "normal" elements are shown in blue, and the anomalies are in red. The reference and test objects are also illustrated. On the right side of Figure 1 it is possible to see how the test points are reordered. The most distant data points from the reference point are executed first.

Figure 1 illustrates the index for a dataset. The initial data is represented on left side. The initial objects are represented by the blue objects, the anomalies are represented by the red objects. The green point at the centre is the reference object, which has been selected at random. The ordered objects are shown on the right side. The right side of the Figure 1 represents the index, which is formed as list of objects, in one dimension, organized based on the distance of all the dataset's object measured from the green point (i.e., the distances are sorted from largest to shortest). This way they avoid going through the whole unordered set. The points are tested along this index order. This is done to achieve the following effect:
The random choice of R, makes it probable that R will be one of the inlier objects when the number of inliers is higher than outliers (just because there are more inliers than outliers). Hence, there is a better chance that the points farthest from R will be the outliers. Because the processing of points is organised in a way that the distance to R is decreasing, the probability of the outliers being processed first will be high. This leads to a faster growth of the cutoff threshold. So, the first requirements stated above is satisfied. Re-ordering the dataset for testing is advantageous for more than that though. For ordered points, it is intuitively clear that iOrca can be interrupted earlier than other techniques by following the Stopping rule:

$$\|x_t - y\| + \left\|x_{(n-k+1)} - y\right\| < c;$$

where y is the reference point, c is the current cutoff threshold defined by user, x_t is any tested point, and $x_{(n-k+1)}$ is the k-th nearest neighbour of y. $\|*\|$ is the 2-norm of a vector. If the inequality is satisfied, then iOrca can be terminated [6].

For this method, upon indexing the distances are pre-computed and available in memory. In order to ensure that all checks on all conditions hold is quick and produces little overhead. To satisfy the second requirement stated earlier, the search the nearest neighbours of a test point should be done in a constant amount of time. For this, the index is applied again. The attempt to find the nearest neighbours does not start from the start of the dataset for each test object, but it starts from the position where it is located on the index and subsequently slowly "spreading" out [6].
Figure 1 presents one of the test objects T (given in magenta) in the dataset. It is represented the position where T is located along the index. So, the attempt to find the nearest neighbour starts out from the object T by with first checks made at the closest two objects (top and bottom), subsequently keeping searching before the score of that object becomes lower than the cut-off (when this is satisfied, the object is pruned). Alternatively, the test object has a score which is above the *t*-th largest anomaly (in this instance the anomalies and the threshold will be updated). As the index sorts the objects projected in one dimension, it is probable that if x_i is closer to x_t than x_j along the index, for a test point x_t, then it will hold when the actual distances are computed, or, if stated formally,

$$\|x_t - y\| > \|x_i - y\| < \left\|x_j - y\right\|, \text{ and from this it can be deduced that}$$

$$\|x_t - x_i\| > \left\|x_t - x_j\right\|.$$

This results in a fast search of the nearest neighbours, as later was demonstrated by experimental study on very large datasets. The suggested method tests each point by traversing the dataset objects following a spiral order. Testing each point, individually, considering the entire dataset makes "I/O expensive". Hence, it is suggested to process the test objects in groups. This way they can be checked at the same time using a spiral search order for the whole test group. The indexing

scheme allows to do this by using a group of index distances (namely the space between the test objects and y). Firstly, calculating the mean of the distance, and secondly performing the spiral search on the index, beginning from this mean value is located. Seeing as how after sorting, the test groups should be similar, checking these groups with the same spiral order is better in terms of speeding up the search as well.
Finally, the indexing intuitively easy, and not expensive to calculate. It is completely stored in memory at runtime as it just includes *n* floating objects numbers and the connected integer indices. iOrca (short for indexed Orca) was estimated to be upto a level of magnitude faster than Orca and it is cheaper in terms of disk space. The important moment is the choice of R which is critical in determining the effectiveness of the pruning [6].

Input: k, t, b, D
Output: O_k, the set of top t outliers in D
Initialization: $c \leftarrow 0$, $O_k \leftarrow \emptyset$, $L = \mathbf{BuildIndex}(D)$;
while $B \leftarrow$*get_next_block*$(D, b, L) <> \emptyset$ **do**
    // process points as specified in $L.id$
  if *(Lemma 4.1 holds for* $B(1)$*)* **then** Terminate;
  else
    $\mu = \mathbf{findAvg}(L(B))$;
    $startID = \mathbf{find}(L >= \mu)$;
    $order = \mathbf{spiralOrder}(L.id, startID)$;
    forall the $b \in B$ **do** $N_k(b) \leftarrow \emptyset$;
    forall the $i = 1$ *to* $|D|$ **do**
      $x = \mathbf{getFromfile}(order(i), D)$;
      forall the $b \in B, b \neq x$ **do**
        if $|N_k(b)| < k$ *or* $dist(b, x) <$
        $maxdist(b, N_k(b))$ **then**
          $N_k(b) \leftarrow \text{Update\_nbors}(N_k(b), x, k)$;
          $\delta_k(b) \leftarrow \max\{\|b - y\| : y \in N_k(b)\}$;
          if $\delta_k(b) < c$ **then** $B \leftarrow B \setminus b$;
        end
      end
    end
    for b=*1 to* B **do**
      $newO \leftarrow newO \bigcup [b; \quad \delta_k(b); \quad N_k(b)]$;
    end
    $O_k \leftarrow \text{Find\_Top\_t}(newO \cup O_k, t)$;
    $c \leftarrow \min\{\delta_k(y, D) : y \in O\}$;
  end
end

Fig. 2. Pseudocode of Algorithm 2: Indexed Orca (iOrca) – [6]

Complexity analysis was performed on the proposed algorithm. In order to get the index, it is needed to make *n* distance calculations, each calculation requires O(d) time. After running the experiment it is observed that the ordering requires O(n log n) time, generating a total running time of O(n log n+nd). Considering the n floating Object numbers needed for the index, the time needed is O(n). The disk space is also preserved, and even more so if iOrca is terminated early [6].

4. Experiments.

iOrca produced exactly the same outliers on the tested datasets, as those obtained from Orca. Hence the accuracy was not contested at the time. The testing was performed on the following real datasets:

1. Covertype: consist of 581,012 rows and 10 attributes
2. Landsat : consist of 275,465 rows and 60 attributes
3. MODIS: consist of 15,900,968 rows and 7 attributes
4. CarrierX: consist of 97,814,864 rows and 19 attributes.

The following default values for the parameters were used: block size, b = 1000, t = 30 and k = 5.

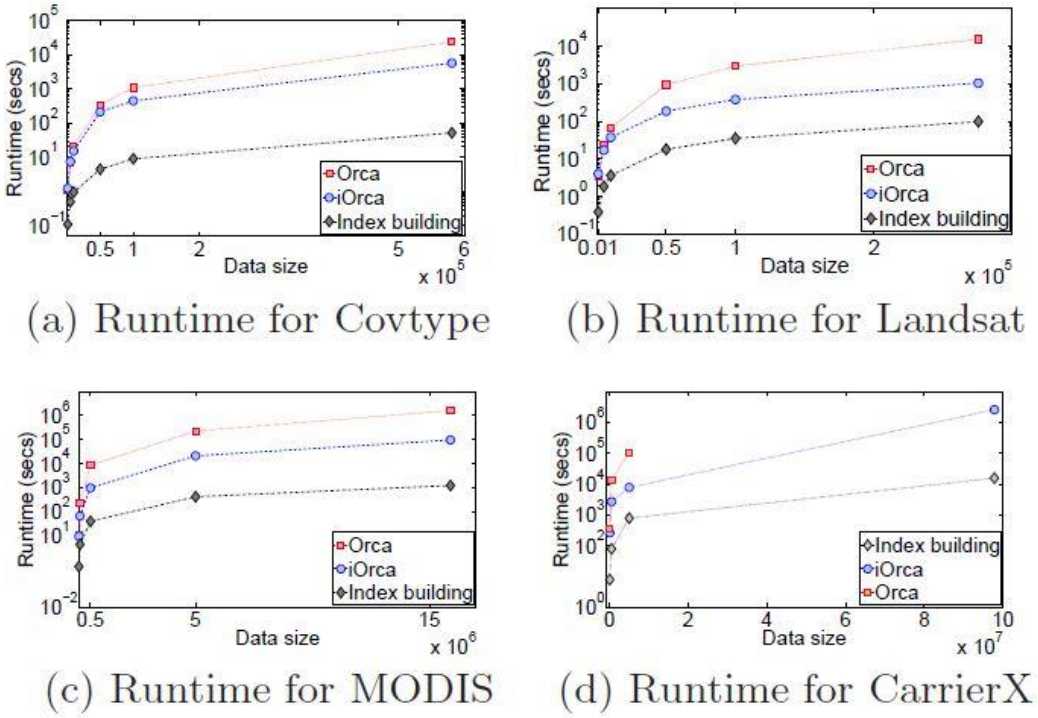

Figure 3: Figure illustrates running time for the 4 datasets described previously. It is visible that for each dataset, iOrca running time is better than Orca [6]

Figure 3 presents the outcomes for run-time about the abovementioned datasets. Figure 3 presents the outcomes for the four datasets. From the four figures a, b, c, d it is possible to see the running time (in log scale) along the y axis for Orca (red points), iOrca (blue points) and the index building phase of iOrca (black points). The portions of the dataset chosen randomly lied on the x axis. From the figures it is possible to derive that iOrca method outperformed the Orca method; “the drop in running time is from 2 to 17 times compared to Orca” [6].
The decrease in runtime is the result of the following factors:

1. the indexing technique sorts the objects in a way that the furthest points from a common reference object are executed before. As these objects are the possible anomalies, the threshold escalates faster in comparison with Orca, so that the inliers are pruned faster

2. the spiral spread is an advantage to spot the closest neighbours for every single test object faster than processing the dataset following the initial sorting
3. the early stopping method allows to prevent checking the whole datasets.

iOrca was repeatedly tested and provided a benchmarking for various indexed schemes, yet its performance was rarely contested (Fig.4). Empirical outcomes have proved that it is more precise than previous distance-based definitions of anomalies. Various researcher commended its build index process as very simple and virtually not having cost time for building indexing scheme. The only reported problem was iORCA's pivot selection method [7]. However, responding to this issue new Algorithms still used probabilistic or approximate outliers density approach.

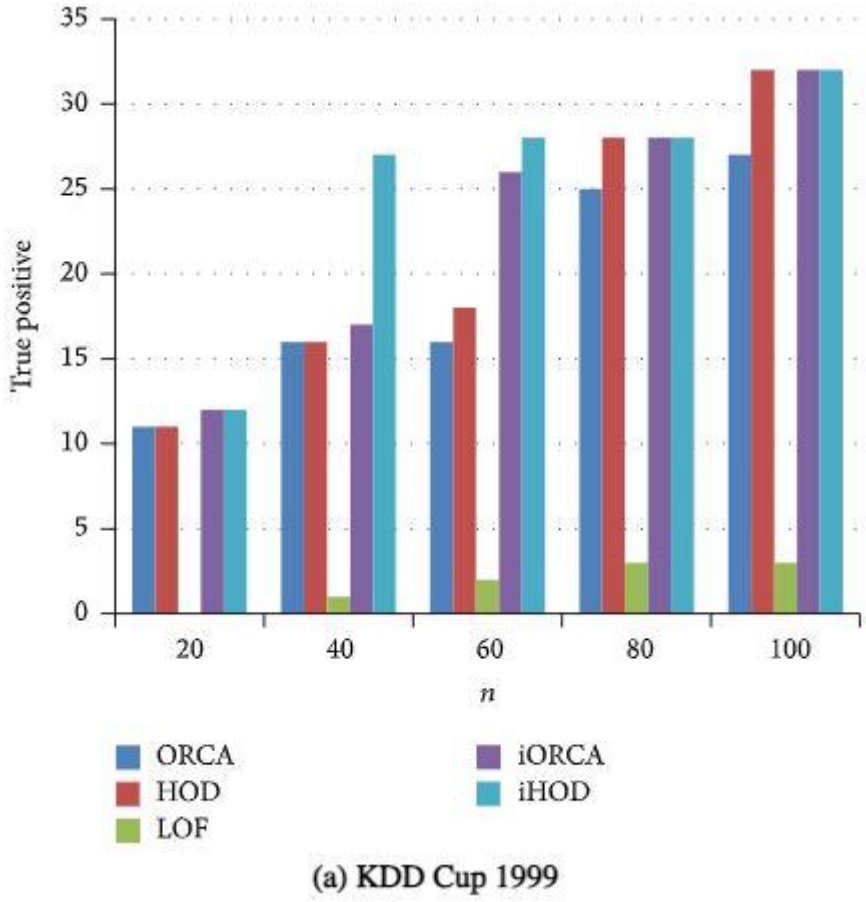

Fig.4 – Comparison by "True positive values" with various values n of five algorithms for KDD dataset [7].

5. Conclusions

In this paper methods of distance-based outlier detection were considered. As it is follows from the sources, the distance-based methods can be significantly improved via correctly defined indexing scheme that allows for early and sufficient pruning of dataset. On a minus-side, the algorithm makes several probability-based choices, from which is can be deducted that there is a possibility that the testing may go wrong. However, the tests performed by the authors on a very large datasets all demonstrated the superiority of the scheme of indexed Orca (iOrca), and at the same a good agreement of the known benchmarks.

References

[1] Ninghao Liu, Donghwa Shin, Xia Hu, Contextual Outlier Interpretation, Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18), 2018.

[2] T. Fuertes, Outliers: looking for a needle in a haystack, 2016, https://quantdare.com/outliers-looking-for-a-needle-in-a-haystack/

[3] Lewison, Outliers detection with Isolation Forest, https://towardsdatascience.com/outlier-detection-with-isolation-forest-3d190448d45e

[4] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, Jörg Sander, LOF: Identifying Density-Based Local Outliers, ACM SIGMOD RecordMay 2000.

[5] Knorr & Ng, Algorithms for Mining Distance-Based Outliers in Large Datasets, Proceedings of the 24th VLDB Conference New York, USA, 1998.

[6] Bhaduri, K., Matthews, B., & Gianella, C., Algorithms for speeding up distance-based outlier detection, Proceedings of the 17th ACM SIGKDD international conference on Knowledge, 2011.

[7] Xu,H., Mao,R., Liao, H., Zhang, H., Lu, M., & Chen, G., Index Based Hidden Outlier Detection in Metric Space, Scientific Programming, vol. 2016, Article ID 8048246, 14 pages, 2016. https://doi.org/10.1155/2016/8048246.