International Journal of Grid and High Performance Computing
Volume 16 • Issue 1

An Approach for Detecting Local Outliers in Grid Queries

Shuang Li, Hunan International Economics University, Changsha, China
Xiaoguo Yao, Hunan International Economics University, Changsha, China*

ABSTRACT

The density local outlier factor algorithm (LOF) needs to calculate the distance matrix for k-nearest neighbor search. The algorithm has high time complexity and is not suitable for the detection of large-scale data sets. A local outlier detection algorithm is proposed based on grid query (LOGD). In the algorithm, the k other data points closest to the data point in the target grid must be in the target grid or in the nearest neighboring grid of the target grid, it is used to improve the neighborhood query operation of the LOF algorithm, the calculation amount of the LOF algorithm is reduced in the neighborhood query. Experimental results show that the proposed LODG algorithm can effectively reduce the time of outlier detection under the condition, the detection accuracy of the original LOF algorithm is basically the same.

KEYWORDS

Distance matrix, Grid, k-nearest neighbors, Local outlier factor (LOF), Memory, Neighborhood query

1. INTRODUCTION

Outlier detection is one of the basic tasks of data mining. Its purpose is to eliminate noise or discover potential and meaningful knowledge*(Li X, Lv J, Yi Z., 2018)*. After long-term development, outlier detection has been widely used in fraud detection, intrusion detection, and abnormal natural climate discovery.

Outlier detection can be roughly divided into five categories based on distribution, bias, clustering, distance, and density*(Peng T. & Yang N. Y.,2018; Zhu L. & Qiu Y. Y.,2017; Xu H. L. & Tang S., 2017; Marateb H R & Rojas-Martínez M,2012)*. The representative density-based algorithm is the LOF algorithm *(Breunig M M & Kriegel H P, 2000)*, local outlier factor (*lof*) is used to represent the degree of local outlier of an object. The larger the *lof* value, the higher the degree of outlier of the data. However, the LOF algorithm also has huge defects, that is, it needs to calculate the distance matrix to determine the k nearest neighbor distance, but a large amount of memory consumption makes outlier detection of large-scale data sets unacceptable. In response to this problem, Lee J et al. used a grid to divide the data *(Lee J & Cho N W, 2016)*, and then calculated the local outliers between the centroid points of each grid. Zhang J. et al. proposed a dense grid method *(Zhang J.*

DOI: 10.4018/IJGHPC.336474 *Corresponding Author

& Sun Z. H., 2011), the data in the dense cell grid was eliminated first, the *lof* value of each data is calculated in the remaining data. Although the above two algorithms speed up the calculation, the spatial distribution characteristics of the data is not considerd, and the detection accuracy rate needs to be improved. Rundensteiner E A proposed a fast branch reduction strategy TOLF that does not need to calculate *knn (Yan Y Z & Cao L, 2017)*. This method performs well on multiple data sets. Kim D et al. proposed the DILOF algorithm *(Na G S & Kim D, 2018)*. Inthis method, a new sampling method is used to avoid data distribution assumptions, while new distance approximation techniques are use to speed up detection. In addition, some scholars have proposed clustering to exclude non-outliers before performing outlier detection *(Yin N. & Zhang L.., 2017; Cao K. Y. & Luan F. J.,2017; Wang J. H. & Jin P., 2015)*, adding information entropy to determine attribute weights, and reducing the impact of attribute differences on outlier detection*(Wang J. H. & Zhao X. X.,2013; Xin L. L. & He W., 2015; Hu C. P. & Qin X. L., 2010)*, using parallel computing *(Bai M & Wang X,2016; Wang X. T. & Shen D. R., 2016)*, introduction of square neighborhoods *(Jie C. M. & Liu H. J.,2012)*, introduction of rough sets*(Yuan Z. & Feng S., 2018)*, and use of average density *(Zhou P. & Cheng Y. Y., 2017)*.

Although many published algorithms have been proposed in recent years to improve the LOF, these algorithms improve the detection efficiency by reducing the number of data points or attributes, and the influence of the filtered data points is ignored on outliers. Aiming at this problem, in this paper, an outlier detection algorithm LOGD is proposed based on grid query. The data is mapped to the grid, and the grid can remember the relative position information between the data points. By querying the adjacent grid of the target grid, the k nearest neighbors of the data points are queried in the target grid, the distance calculations between data are reduced in the LOF algorithm when performing k-nearest neighbor query, and the local outlier value of each data is finally calculated. This algorithm can reduce the amount of calculation and improve the detection speed, while still having the same accuracy rate as the original LOF algorithm.

2. BASIC CONCEPTS OF THE LOF ALGORITHM

Local outlier factor (LOF) algorithm is a popular method for outlier detection in data mining. It is based on the concept of local density and compares the density of an instance to its neighbors to identify outliers *(Breunig M. M.,2000; Tang J., 2015; Wen J.,2013; Saeed H.,2017)*.

The basic idea behind LOF is that outliers are instances that have significantly lower density compared to their neighbors. LOF measures the degree of outlierness for each instance by examining the local density ratio. It calculates the LOF score for an instance by comparing its local reachability density (LRD) to the LRDs of its k-nearest neighbors. If an instance has a much lower LRD than its neighbors, it is considered an outlier.

Definition 1: (*k*-th distance of object *p: k-distance (p)*). For any natural number *k*, dataset *D*, define the *k*-th distance of *p* as the distance between *p* and some object *o*, and record it as *k- distance (p)*, where the object *o* meets the following conditions:

(1) At least *k* objects $o' \in D\backslash\{p\}$ satisfy: $d(p, o') \leq d(p, o)$
(2) At most *k-1* objects $o' \in D\backslash\{p\}$ satisfy: $d(p, o') < d(p, o)$

Where the distance between object *p* and object *o* is written as *d (p, o)*.

Definition 2: (*k*-th distance neighborhood of object *p:* $N_k(p)$) . Given the *k-distance(p)* of data object *p*, the *k*-th distance neighborhood of object *p* is the set of data points that its distance from *p* does not exceed *k-distance(p)*:

$$N_k(p) = \{q \mid d(p, q) \leq k\text{-distance}(p)\} \quad (1)$$

Therefore, the number of *k*-th neighbor points of the object *p* is $| N_k(p) | \geq k$.

Definition 3: (The reachable distance of object *p* relative to object *o: reach-dist(p, o)*). The reachable distance of object *p* from object *o* to *k* is the maximum value between *k* nearest neighbors' distance of *p* and the distance from *p* to *o*, which is defined as follows:

$$reach\text{-}dist\ (p, o) = max\ \{k\text{-}distance(p), d(p, o)\} \tag{2}$$

Definition 4: (Local reachable density of object *p:* $lrd_k(p)$). The local reachable density of object *p* is the inverse of the average reachable density of object *p* from the point to *p* in the neighborhood of the *k*-th distance, it is expressed as Equation (3):

$$lrd_k\left(p\right) = 1 / \left(\frac{\sum_{o \in N_k(p)} reach - disp_k\left(p, 0\right)}{\left|N_k\left(p\right)\right|} \right) \tag{3}$$

Definition 5: (local outlier factor of object *p:* $lof_k(p)$). The local outlier factor of object *p* represents the outlier degree of object *p*, it is expressed in equation (4):

$$lof_k\left(p\right) = \sum_{o \in N_k(p)} \frac{lrd_k\left(o\right)}{lrd_k\left(p\right)} / \left|N_k\left(p\right)\right| \tag{4}$$

If the *lof* value of the object *p* is larger, the outlier degree of the object *p* is higher, and the more likely it is an outlier. On the contrary, the probability that the object *p* belongs to a certain cluster is higher.

Local Outlier Grid Detection (LOGD) algorithm is a variation of the LOF algorithm that incorporates the concept of grid-based partitioning. LOGD divides the data space into a grid structure and calculates the local density of instances within each grid cell. It then applies the LOF algorithm to identify outliers based on the density information obtained from the grid cells.

The relationship between LOF and LOGD is that LOGD is an extension or modification of the LOF algorithm that incorporates grid-based partitioning to improve efficiency and scalability. By dividing the data space into grid cells, LOGD reduces the number of distance calculations required, making it more efficient for large datasets. However, the fundamental concept of measuring local density and comparing it to the neighbors remains the same in both LOF and LOGD.

In summary, LOF is a general outlier detection algorithm that measures outlierness based on local density, while LOGD is a specific variation of LOF that incorporates grid-based partitioning for enhanced efficiency.

3. LOGD ALGORITHM

The computational efficiency of the Local Outlier Factor (LOF) algorithm is primarily determined by the neighborhood query operation. This study proposes the utilization of a grid structure to store and retrieve data location information within neighboring grids, thereby optimizing the neighborhood query process in the LOF algorithm. The algorithmic steps can be outlined as follows:

- Grid Partitioning: The dataset is divided into a grid structure to efficiently store the characteristics of data distribution in different regions.

- Grid-based Neighbor Search: The grid structure is leveraged to perform neighbor queries, enabling rapid identification of data points within the vicinity of a given point of interest.
- Local Density Calculation: The local density of each data point is computed based on the number of neighboring points found within a predefined distance threshold.
- LOF Computation: The LOF for each data point is calculated by comparing its local density with that of its neighbors, providing a measure of its outlying behavior within its local neighborhood.

By integrating grid-based optimization, the LOF algorithm aims to enhance its scalability and efficiency in identifying local outliers within large datasets. This approach can significantly reduce the time complexity associated with the neighborhood query operation, making the LOF algorithm more practical for real-world applications involving extensive data analysis.

3.1 Meshing

Given a *d*-dimensional data set *D*, the number of data is *N*. Let the value of the **i**-th dimension be in the interval $R_{gi} = [l_i, h_i]$ $(i = 1,2,\ldots, d)$, then $S = R_{g1} \times R_{g2} \times \ldots \times R_{gd}$ is the *d*-dimensional data space. Each dimension of the data space is divided into equal-length and disjoint intervals. Thereby, grid cells are formed. On each dimension, these grids are left-closed and right-open. In this way, the data space is divided into $\prod num_i$ super-rectangular grid cells of equal volume (num_i is the number of intervals in the i-th dimension of the data space). The grid side length is given in equation (5):

$$length = a \times \sqrt[d]{\prod_{i=1}^{d} \left(h_i - l_i\right) / N} \tag{5}$$

where *a* is the grid control factor *(Han L. Z. & Qian X. Z.,2018)*, it is used to control the grid size. After several experiments, the meshing effect in this paper is better when $a = 1.7$. The number of grids in each dimension is Equation (6):

$$num = \left\lfloor \frac{h_i - l_i}{length} \right\rfloor \tag{6}$$

In this way, each dimension of the data space is divided into grids of equal length and disjointness. These grids are left-closed and right-opened in each dimension.

3.2 Data Binning

Data binning, also known as data discretization, involves mapping each object in a dataset to the corresponding grid. For a given data object $X = (x_1, x_2,\ldots, x_n)$, the subscript of the grid in each dimension can be calculated using Equation (7). This process partitions the continuous attribute values into discrete intervals, effectively organizing the data into a grid structure for efficient storage and retrieval.

$$index = \left\lfloor \frac{x_i - l_i}{Len} \right\rfloor \tag{7}$$

Binning is a crucial preprocessing step in data analysis and mining, as it enables the handling of large datasets by reducing the computational complexity associated with continuous data. By discretizing the data into grids, the binning process facilitates neighborhood queries, density

calculations, and outlier detection in algorithms such as the Local Outlier Factor (LOF). Additionally, binning can aid in visualizing and interpreting data patterns, especially in multidimensional spaces, by simplifying the representation of continuous data into a discrete form.

The use of data binning has wide-ranging applications in various fields, including machine learning, data warehousing, and spatial data analysis. It provides a systematic approach to organizing and processing data, contributing to the efficiency and interpretability of analytical tasks. As datasets continue to grow in size and complexity, the role of data binning becomes increasingly significant in optimizing data analysis processes.

3.3 Optimize Neighborhood Queries

LOGD is the improvement of the LOF algorithm, it is mainly reflected in the optimization of neighborhood queries. This step is divided into the following two steps:

- Query grid G_i in order ($i = 1, 2, ..., n$, where n is the number of grids). If there are no data points in the grid, query the next grid.
- If the number of data in the query grid G_i is not 0, then G_i is used as the center grid, the adjacent grids of the G_i grid is queried, and then the query grids are merged all into the grid G (As shown in Figure 1, the dark gray grid is G_i, and the dark and light gray grids are merged into grid G). If the number of data contained in grid G is less than $K + 1$ (K is the k-th nearest neighbor parameter), the grid G is used as the center to continue to query the neighboring grids of grid G, and then all the query grids are combined as the new grid G (see Figure 2, the dark gray grid is G_i, the medium gray is the grid G formed by the previous grid merge, all the gray grids are the second grid merge to form a new grid G). The number of data points is calculated in grid G. If the number of data points is less than $K + 1$, the above query step is repeated. If the number of data points is greater than $K + 1$, calculate the distance between each data d_j in the G_i grid and all the data in G, and find the k nearest neighbor data points of d_j, and record the k nearest

Figure 1. First grid query

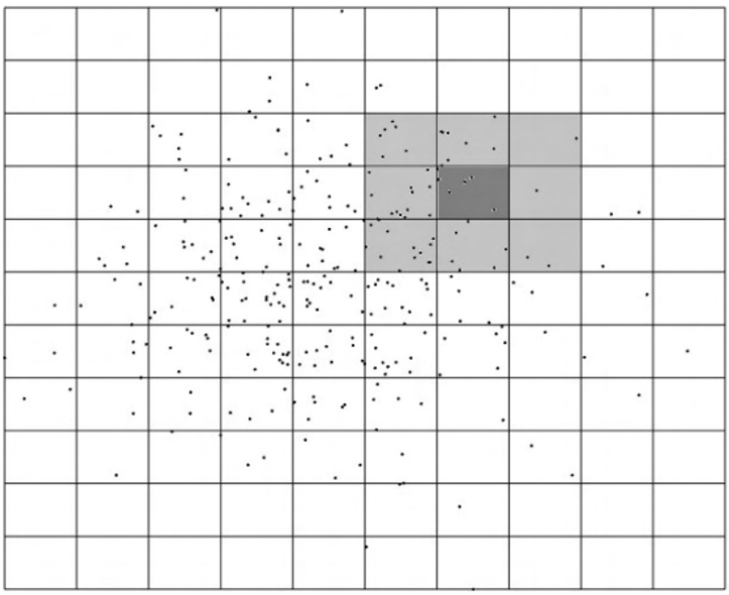
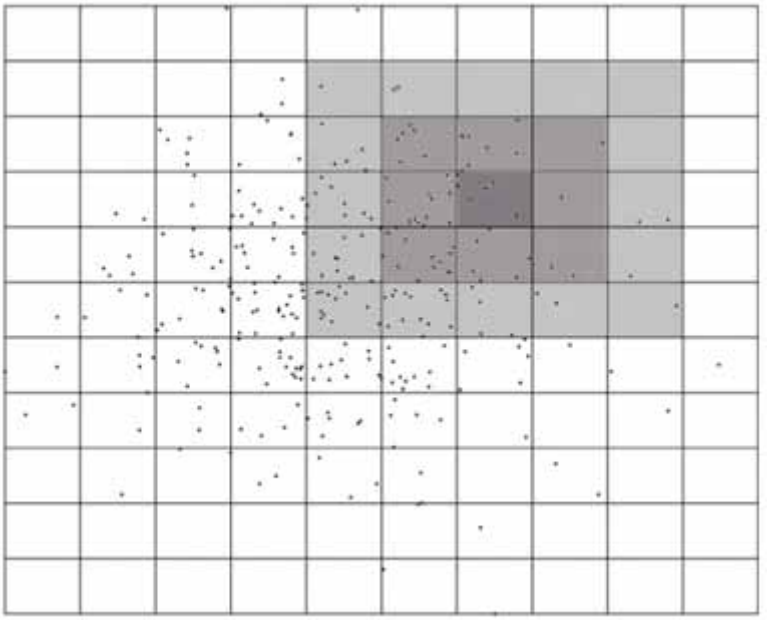


Figure 2. Second grid query



neighbor distance of data point d_j as *k-distance*(d_j), *k-distance*(d_j) is the numbered set of each data point in the neighborhood of $N_k(d_j)$ and *k*-distance.

In this step, adjacent meshes are defined in two-dimensional data as a group of meshes with common edges or vertices, and they appear as a set of hyperrectangles with common faces and vertices in a super rectangle, The grid can be up to 3^d-*1*

3.4 Find Outliers

To identify outliers, the Local Outlier Factor (LOF) value is computed for each data point based on formulas (3) and (4). The LOF quantifies the outlying behavior of a data point in its local neighborhood. If the LOF value of a data point exceeds a predefined threshold, denoted as ε, the data point is classified as an outlier; otherwise, it is considered a non-outlier.

The LOF threshold, ε, is a critical parameter that influences the sensitivity of the outlier detection process. Setting an appropriate ε value is essential for accurately distinguishing outliers from non-outliers. A higher ε value may result in a more lenient outlier classification, while a lower ε value can lead to a stricter classification, potentially identifying fewer outliers.

Furthermore, the LOF threshold ε can be determined through various methods, including statistical analyses, domain knowledge, or through iterative experimentation to achieve the desired outlier detection performance.

In practical applications, the identification of outliers through the LOF algorithm and the determination of an optimal LOF threshold ε play a crucial role in anomaly detection, fraud detection, and quality control across diverse domains such as finance, healthcare, and manufacturing. Additionally, the flexibility in setting the LOF threshold allows for customization based on specific requirements and the characteristics of the dataset, contributing to the adaptability and effectiveness of the outlier detection process.

3.5 Local Outlier Grid Detection (LOGD) Algorithm Detailed Description

The pseudo code of the local outlier detection algorithm based on grid query is as follows.

```
1.  input: D, K, ε;
2.  numbered for each data point;
3.  put data ∈ D into grids;
4.  for Gi in Grid // Grid for all grid collections
5.  if the number of data in Gi: NGI> 0
6.  G0 = Gi;
7.  while 1
8.  query the adjacency grids of Gi;
9.  merge Gi and G′i adjacency grids as G;
10. if the number of data in G: NG> = K + 1
11. break;
    else
12. make G as Gi;
13. calculate distance between data in G0 and G;
14. record: k-distance (d), j N (k d) j and data number
15. calculate lrd (k p); i // pi is the data object in dataset D
16. calculate lof (k p); i
17. for p in D
18. if lof (p) i> ε
19. p will be marked as outlier;
20. output all outliers;
```

Here's the pseudo code for the local outlier detection algorithm based on grid query:

```
function LocalOutlierFactor(GridSize, MinPts, Dataset):
    // Step 1: Divide the dataset into a grid structure
    Grid = DivideIntoGrids(Dataset, GridSize)

    // Step 2: Compute the local density for each data point
    for each data point X in Dataset:
        Neighbors = GridQuery(Grid, X, GridSize)
        if |Neighbors| < MinPts:
            X.local_density = LOW
        else:
            X.local_density = ComputeLocalDensity(X, Neighbors)

    // Step 3: Compute the Local Outlier Factor (LOF) for each
data point
    for each data point X in Dataset:
        Neighbors = GridQuery(Grid, X, GridSize)
        X.LOF = ComputeLOF(X, Neighbors)

    return Dataset
```

In this pseudo code, the DivideIntoGrids function divides the dataset into a grid structure based on the specified GridSize. The GridQuery function retrieves the neighboring data points within a specific

grid size. The ComputeLocalDensity function calculates the local density of a data point based on its neighbors, while the ComputeLOF function computes the Local Outlier Factor for each data point.

This pseudo code outlines the basic steps of the local outlier detection algorithm based on grid query, where the local density and LOF are computed for each data point to identify outliers within the dataset.

4. LOGD ALGORITHM PERFORMANCE ANALYSIS

The performance analysis of LOGD algorithm is mainly explained from three aspects: algorithm innovation point, time complexity analysis, and experimental results.

4.1 Algorithm Innovation

The core idea of the LOGD algorithm is mainly reflected in the use of the unique memory of the grid to optimize the neighborhood query operation of the LOF algorithm, which is specifically reflected in the following two aspects.

(1) Only the distance between all the data in the grid covered by the target grid and its adjacent grids need to be considered, and the distance between the data in the target grid and all the data in the entire data set need not be considered. Greatly reduces the amount of calculations in neighborhood queries.
(2) Each time a target grid is determined, and then its adjacent grids are queried until the stop query condition is satisfied, this is as a grid operation. The calculation of the distance between the data in this grid operation is only to calculate the distance between the data in the target grid and all the data involved in this grid operation, not the distance between all the data in this operation. This also reduces the amount of calculation for distance.

4.2 Algorithm Time Complexity Analysis

Suppose *d*-dimensional data *D*, the number of data is *N*, and the number of divided grids in each dimension is m, then the average number of data in each grid is $n = N/md$. In a multi-dimensional case, each mesh has a maximum of 3^d-*1* adjacent meshes. In the worst case, each grid and its adjacent grids need to be queried once, but only the distance between the data in the central grid and all the data in the query grid is calculated, so the time complexity is *O(f (N, d, m))* at this time, then there is the following formula:

$$O(f(N, d, m)) < 3^d \times n \times n \times m^d$$
$$O(f(N, d, m)) < 3^d \times (N / m^d) \times (N / m^d) \times m^d$$
$$O(f(N, d, m)) < (3 / m)^d \times N^2$$

And because the number of grids is the same in each dimension, let $HL = h_i\text{-}li$, then thre is the following formula:

$$Len = \frac{a \times HL}{\sqrt[d]{N}}, m = \frac{HL}{Len} = \frac{\sqrt[d]{N}}{a}$$

m is put into the above inequality, there is the following formula:

$$O(f(N, d, m)) < (3a)^d \times N$$

Its time complexity is *O(kN)*, which is linear time complexity.

4.3 Experimental Results

In order to verify the effectiveness of the algorithm in this paper, the LOF and gridLOF *(Lee J & Cho N W, 2016)* algorithms were compared with the LOGD algorithm. Four data sets were selected in the experiment. The relevant attributes of the data set are shown in Table 1. All the algorithms in this article are implemented and processed with MATLAB R2016a tool. The experimental environment is: CPU is Intel i5 2.50 GHz; memory is 8 GB; OS is Windows 7.

After several experimental comparisons, the parameters of each algorithm with the best detection results are shown in Table 2, where *k* represents the *k*-th neighbor, *e* represents the local outlier factor threshold, and *Gn* represents the number of grids in each dimension in the grid-LOF algorithm.

When the optimal parameters are selected, the detection effect of each algorithm is shown in Figures 3-6.

For the analysis of Figures 3 ~ 6, for the Ag data set with a more evenly distributed data set, each algorithm can detect outliers, and the detection results of the LOF algorithm and the LOGD algorithm are basically the same. For data set Jain with uneven density distribution, all algorithms can also detect outliers, but since the *k*-th nearest neighbor parameter of the LOF algorithm is global-oriented, when the data distribution density is greatly different, the LOF algorithm can easily identify the sparse density cluster interior points as outliers, the grid-LOF algorithm performs a *k*-nearest neighbor query on the grid centroid, and the grid centroid is less affected by the data density. This also results in the LOF and LODG algorithms that there is better detection accuracy in the Jain dataset than the grid-LOF algorithm, but the false detection rate is also high. For the pb and data4 datasets, the amount of data in the data4 dataset is greater than the pb dataset, but the commonality between these two datasets is that the boundary between clusters is not obvious in the case of LOF and LOGD, if the local outlier threshold *e* is selected to be too small, the false detection rate is high, and if it is selected to be too large, the detection accuracy is reduced. The grid-LOF algorithm does not consider the spatial distribution of data points in the grid. The cluster edge grid may contain outliers and intra-cluster points. Once the centroid is calculated by the cluster edge grid, it is determined to be an outlier, its Intra-cluster points can be misidentified as outliers. If the calculated centroid is determined as an

Table 1. Experimental dataset related attributes

data set	number	cluster	attribute	noise	noise rate/%
Ag	888	7	2	39	4.39
Jain	423	2	2	26	6.15
pb	350	3	2	36	10.29
data4	5 050	4	2	59	1.17

Table 2. Various algorithm parameters

data set	LOF		LOGD		grid-LOF		
	k	*e*	*k*	*e*	*k*	*e*	*Gn*
Ag	10	1.291	10	1.291	8	1.275	30
Jain	11	1.279	11	1.279	14	1.280	27
pb	6	1.235	6	1.235	12	1.229	50
data4	20	1.549	20	1.549	9	1.305	60

Figure 3. Effects of various algorithms in the Ag dataset

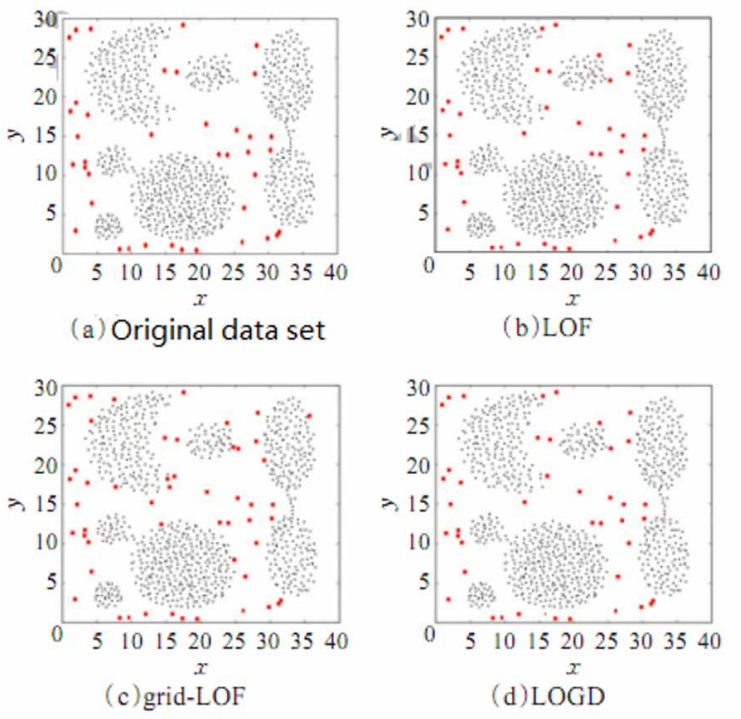


Figure 4. Effects of various algorithms in the Jain dataset

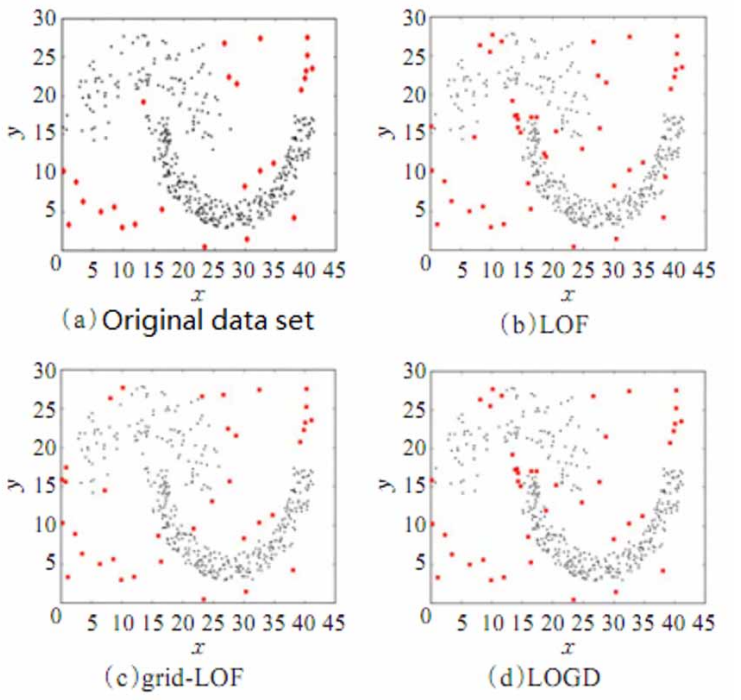


Figure 5. Effects of various algorithms in the pb dataset

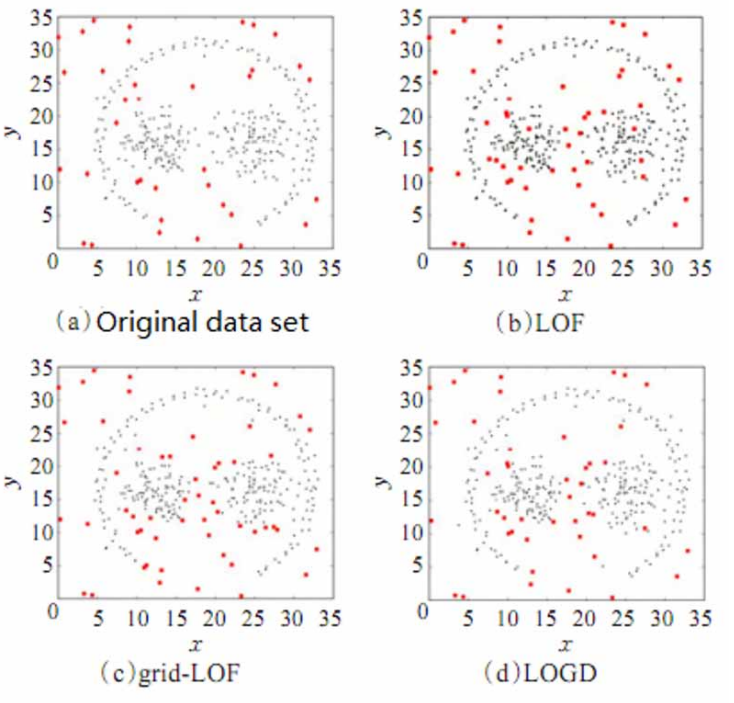
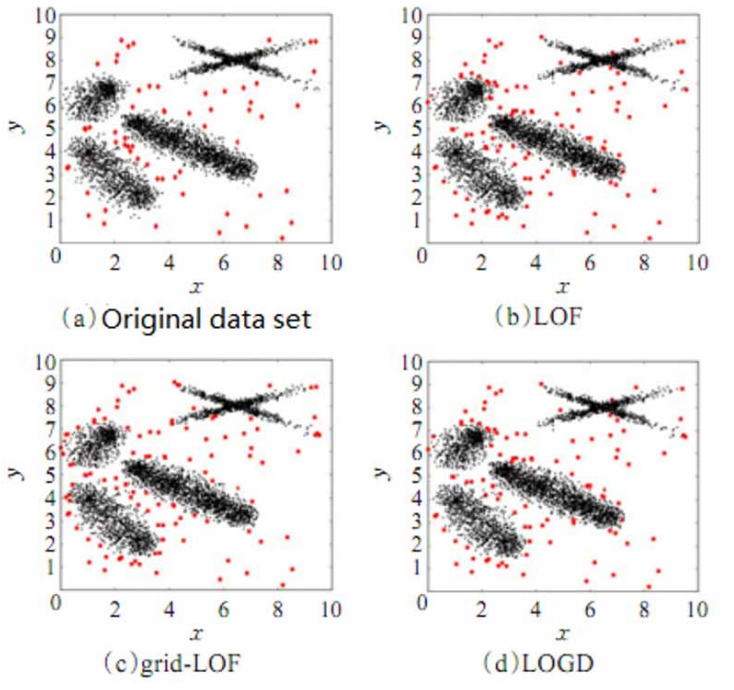


Figure 6. Effects of various algorithms in the data4 dataset



intra-cluster point, outliers in the cluster edge grid will be determined as intra-cluster points, which will also cause misdetection. In the case where the boundaries between clusters are not obvious, all three algorithms do not solve them well.

Here are the accuracy indicators of three algorithms for measuring LOF, grid-LOF and LOGD:

$$Accuracy = \frac{correctly\ found\ outliers}{actual\ outliers\ in\ the\ dataset} \times 100\%$$

$$False\ detection\ rate = \frac{number\ of\ incorrectly\ detected\ outliers}{number\ of\ actually\ detected\ outliers} \times 100\%$$

On these four data sets, the detection accuracy and false detection rate of each algorithm are shown in Figures 7 and 8.

Figure 7. Accuracy of each algorithm on four data sets

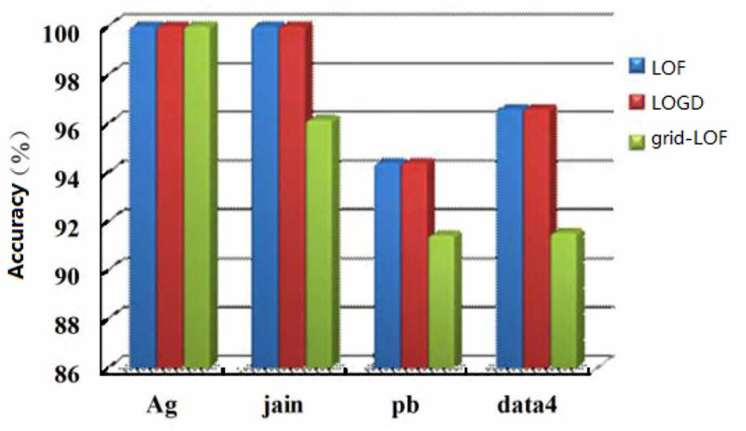
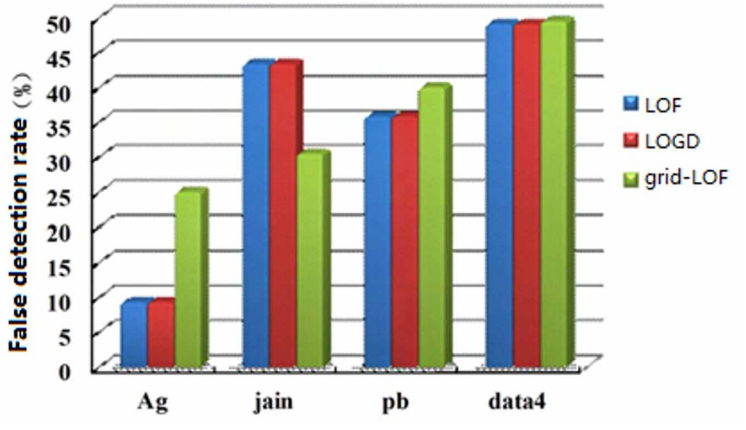


Figure 8. False detection rate of each algorithm on four data sets



In Figures 7 and 8, it can be seen that the LOF and LOGD algorithms have higher detection accuracy than the grid-LOF algorithm in the detection accuracy rate, but on the Jain dataset with uneven density distribution of the data set, although the detection accuracy of the LOF and LOGD algorithm is high, but the false detection rate is also high.

In addition to the accuracy and false detection rate to measure the accuracy of the algorithm, the efficiency of the algorithm needs to be measured in terms of the algorithm's running time. As are shown in Figure 9, the running time of each algorithm on four data sets is compared.

By analyzing Figure 9, the running time of the LOF algorithm is the longest. For the LOGD algorithm, the running time is greatly reduced due to the improved k nearest neighbor query operation. The running time of the grid-LOF algorithm is closely related to the number of grids. When the grid-LOF detection accuracy is high, the number of grids in each dimension is also large, and the corresponding running time is also long. On the data4 data set in Figure 9, the running time of the LOGD algorithm is shorter than the grid-LOF algorithm. Therefore, the LOGD algorithm performs best in the comprehensive detection accuracy and running time.

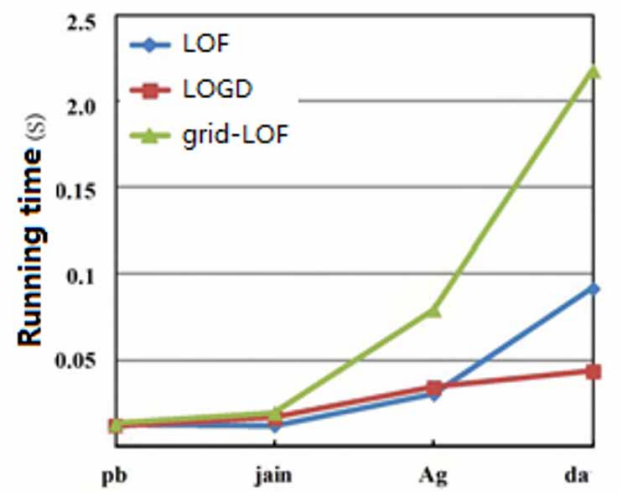
5. CONCLUSION

In this paper, we propose an outlier detection algorithm called Local Outlier Grid Detection (LOGD) based on grid query. The algorithm involves mapping the data to a grid structure, which enables the grid to retain the relative position information between data points. By querying the adjacent grids of a target grid, we can retrieve the k nearest neighbors of data points within the target grid. This query approach reduces the number of distance calculations required between data points when performing the k-nearest neighbor search in the LOF algorithm.

Furthermore, we recalculate the size of the local outlier for each data point using the queried k nearest neighbors within the target grid. This recalibration helps in reducing the computation overhead and significantly improves the detection speed, while maintaining the same level of accuracy as the original LOF algorithm.

However, a notable challenge in the LOGD algorithm is the repeated calculation of distances between data points in adjacent grids. To address this issue, we propose utilizing a suitable data structure to store the distance matrix between data points in adjacent grids. This will be a subject of further investigation and study in our future work.

Figure 9. Run time of each algorithm on four data sets



Overall, the proposed LOGD algorithm offers a more efficient and scalable approach to outlier detection by leveraging grid-based partitioning and optimizing the distance calculation process. The experimental results demonstrate its effectiveness in reducing computational complexity while maintaining the detection accuracy of the original LOF algorithm.

ACKNOWLEDGMENT

This work is sponsored by 2022 Hunan Province General Higher Education Teaching Reform Research Project (Xiang Jiao Tong [2022] No. 248) PBL Teaching Model Reform and Practice of New Media Operation Technology (HNJG-2022-1163), China. In 2022, Hunan International Economics University first-class undergraduate course (Hunan International Economics University (2023) No. 15) "New Media Operation Technology" online first-class course (19).2023 Education Reform Project of Hunan International Economics University: Research on Hybrid Teaching Mode of "New Media Operation Technology" based on STEM Education Concept, Hunan International Economics University, No.43, No.55 [2023].

CONFLICT OF INTEREST STATEMENT

We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled.

REFERENCES

Bai, M., Wang, X., Xin, J., & Wang, G. (2016). An efficient algorithm for distributed density-based outlier detection on big data. *Neurocomputing*, *181*(C), 19–28. doi:10.1016/j.neucom.2015.05.135

Breunig, M. M., Kriegel, H. P., Ng, R. T., & Sander, J. (2000). LOF: identifying densitybased local outliers. *In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. (pp.93-104). Dallas, Texas, USA: ACM. doi:10.1145/342009.335388

Cao, K. Y., Luan, F. J., & Sun, H. L., e al. (2017). Density-based local outlier detection on uncertain data. *Chinese Journal of Computers*, *40*(10), 2231–2244. doi:10.11897/SP.J.1016.2017.02231

Han, L. Z., Qian, X. Z., & Luo, J. (2018). Multi-density clustering algorithm DBSCAN based on region division. *Jisuanji Yingyong Yanjiu*, *35*(6), 1668–1671. doi:10.3969/j.issn.1001-3695.2018.06.015

Hu, C. P., & Qin, X. L. (2010). A Density-Based Local Outlier Detecting Algorithm. *Journal of Computer Research and Development*, *47*(12), 2110–2116.

Jie, C. M., Liu, H. J., & Zhu, Q. S. (2012). Square symmetric neighborhood based local outlier detection algorithm. *Jisuanji Yingyong Yanjiu*, *29*(2), 472–474. doi:10.3969/j.issn.1001-3695.2012.02.018

Lee, J., & Cho, N. W. (2016). Fast outlier detection using a grid-based algorithm. *PLoS One*, *11*(11), e0165972. doi:10.1371/journal.pone.0165972 PMID:27832163

Li, X., Lv, J., & Yi, Z. (2018). An efficient representation-based method for boundary point and outlier detection. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(1), 51–62. doi:10.1109/TNNLS.2016.2614896 PMID:27775542

Marateb, H. R., Rojas-Martínez, M., Mansourian, M., Merletti, R., & Mañanas Villanueva, M. A. (2012). Outlier detection in high-density surface electromyographic signals. *Medical & Biological Engineering & Computing*, *50*(1), 79–89. doi:10.1007/s11517-011-0790-7 PMID:21698432

Na, G. S., Kim, D., & Yu, H. (2018). *DILOF: effective and memory efficient local outlier detection in data streams*. The 24th ACM SIGKDD International Conference, London, United Kingdom. (pp.1993-2002). DOI: . 3220022 doi:10.1145/3219819

Peng, T., Yang, N. Y., & Xu, Y. B. (2018). An Outlier Detection Method Based on Ranking and Clustering in Bi-typed Heterogeneous Network. *Tien Tzu Hsueh Pao*, *46*(2), 281–288. doi:10.3969/j.issn.0372-2112.2018.02.004

Saeed, H. (2017). LOF+: Fast outlier detection using the local correlation integral. *Information Systems*, *64*, 1–14.

Tang, J., Liu, Z., Liao, S., & Su, Z. (2015). A clustering approach for local outlier detection. *Information Sciences*, *316*, 1–17.

Wang, J. H., & Jin, P. (2015). Outliers detecting based on rough reduction and grid. *Computer Engineering and Applications*, *51*(3), 133–137. doi:10.1201/b18565-28

Wang, J. H., Zhao, X. X., & Zhang, G. Y. (2013). NLOF:A New Density-based Local Outlier Detecting Algorithm. *Computer Science*, *40*(8), 181–185. doi:10.3969/j.issn.1002-137X.2013.08.038

Wang, X. T., Shen, D. R., & Bai, M. (2016). BOD:An Efficient Algorithm for Distributed Outlier Detection. *Chinese Journal of Computers*, *39*(1), 36–51. doi:10.11897/SP.J.1016.2016.00036

Wen, J. (2013). Density-based local outlier detection. *Pattern Recognition*, *46*(7), 1914–1931.

Xin, L. L., He, W., & Yu, J. (2015). An outlier detection algorithm based on density difference. [Engineering Science]. *Journal of Shandong University*, *45*(3), 7–14. doi:10.6040/j.issn.1672-3961.1.2014.182

Xu, H. L., Tang, S., & Mao, R.i, et al. (2017). Various Pivots Based Outlier Detection Algorithm in Metric Space. *Chinese Journal of Computers*, *40*(12), 2839–2855. doi:10.11897/SP.J.1016.2017.02839

Yan, Y. Z., Cao, L., & Rundensteiner, E. A. (2017). Scalable top-n local outlier detection. *The 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. doi:10.1145/3097983.3098191

Yin, N., & Zhang, L. (2017). Research on Application of Outlier Mining Based on Hybrid Clustering Algorithm in Anomaly Detection. *Computer Science*, *44*(5), 116–119. doi:10.11896/j.issn.1002-137X.2017.05.021

Yuan, Z., & Feng, S. (2018). Outlier detection algorithm based on neighborhood value difference metric. *Jisuanji Yingyong*, *38*(7), 1905–1909. doi:10.11772/j.issn.1001-9081.2017123028

Zhang, J., Sun, Z. H., & Yang, M. (2011). Fast Incremental Outlier Mining Algorithm Based on Grid and Capacity. *Journal of Computer Research and Development*, *48*(5), 823–830.

Zhou P., Cheng Y. Y. (2017). An Improved LOF Outlier Detection Algorithm. *Computer Technology and Development*, *27*(12), 115-118. doi:. 1673-629X.2017.12.02510.3969/j.issn

Zhu L., Qiu Y. Y., & Yu S. (2017). A Fast kNN-Based MST Outlier Detection Method. *Chinese Journal of Computers*, *40*(12): 2856-2870. doi:. 1016. 2017.0285610.11897/SP.J

Shuang Li is studying for Ph.D in business management from Lyceum Of The Philippines University Manila Campus. She received her Master's degree in Applied Economics from Hunan University. Currently, she is a lecturer at the School of Information and Mechatronic Engineering, Hunan International Economics University, Her research interests include e-commerce, business management.

Xiaoguo Yao, Corresponding author, He is in Hunan International Economics University, Changsha, China. He is graduated from Lanzhou University of Technology with a master's degree, intermediate automation engineer, His research direction focuses on image processing, computer vision, artificial intelligence and other research.