SPATIAL OUTLIER DETECTION USING IMPROVED Z-SCORE TEST

VAIBHAV SURI
Student, CSE Department, GITAM University, Visakhapatnam, India
vaibhavsuri1@gmail.com

P. SOURABH
Student, CSE Department, GITAM University, Visakhapatnam, India
psourabh92@gmail.com

T. NIHAR
Student, CSE Department, GITAM University, Visakhapatnam, India
nihartirupati92@gmail.com

KALYAN NETTI
Senior Scientist, National Geophysical Research Institute, Hyderabad, India
netti_kalyan@ngri.res.in

Abstract:

Outlier detection is a prominent technique in data mining since it has various vital applications. Outlier detection can be used to eliminate noise or study the specific observations in data that are dissimilar to those in its neighborhood. The identification of spatial outliers can be used to reveal hidden but valuable knowledge in many applications [1]. The z-score test has long been used to detect outliers in data. In this work, a modified version of the z-score test is proposed which can lead to a reduction of the time-complexity of the traditional z-score test by a value of *n* each iteration, where *n* is the number of spatial points in the data set which have not been discovered as outliers.. Using the outlier threshold value to calculate the value of the modified z-score function rather than calculating the z-score value for each observation in the dataset makes this possible.

Keywords: *spatial outlier, modified z-score test, spatial outlier detection algorithm, z-score test*

1. INTRODUCTION

Grubbs defined outlier as: “An outlier is something which is an outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs" [2]. Spatial outliers are observations in a spatial dataset that are significantly different from their neighborhood, though not completely different from the entire given set of data. They, when identified may cause instability among the data and may lead to the discovery of some unexpected or undesirable patterns.

The identification of spatial outliers can be used to reveal hidden but valuable knowledge in many applications. For example, it can help locate extreme meteorological events such as tornadoes and hurricanes, identify aberrant genes or tumor cells, discover highway traffic congestion points, pinpoint military targets in satellite images, determine possible locations of oil reservoirs, and detect water pollution incidents.

2. RELATED WORKS

A z-score (also known as z-value, standard score, or normal score) is a measure of the divergence of an individual experimental result from the most probable result, the mean. *Z* is expressed in terms of the number of standard deviations from the mean value.

$$z = \frac{X - \mu}{\sigma} \tag{1}$$

X = Experimental Value

μ = *Mean*

σ = *Standard Deviation*

It is apparent that the measure of divergence provided by the z-score provides a mechanism to determine the magnitude by which an observation "diverges" from the other observations of the dataset and if found to be large enough, the observation can be deemed an outlier [3].

The traditional z-statistic has also been previously used in spatial outlier detection algorithm which also involves identifying the *K* nearest neighbors of a data object however involves computing the value of the traditional z-statistic test for each data object and then comparing it to the threshold value (*θ*) in order to evaluate whether the data object under scrutiny is an outlier or not.

3. PROBLEM DEFINITION

The database consists of a set of n spatial points, X= {x_1,x_2,...,x_n} The function *f*(x_i,x_j) is defined as the relationship function mapping from X*x*X to R (real number set) which is used to determine the relationship between any two spatial points x_i and x_j The implementer depending upon his/her requirements can choose the relationship function, which may be a Euclidean distance function, Manhattan distance function, etc. The function *c*(x_i) is defined as the comparison function for a spatial point i. The K nearest neighbors of a spatial point x_i are denoted as *KNBR*(x_i) . The threshold value is denoted as θ and the current comparator value calculated from the threshold value is denoted as *CURRENT*.

4. PROPOSED ALGORITHM

In the previous mechanisms, the comparison function value is calculated for each observation in the dataset and these values are then used to calculate the z-score value for each observation in the dataset. The z-score value for each observation is then compared with the threshold value (θ) to determine whether it is an outlier or not. However in this proposed algorithm after calculating the comparison function values, these values are used along with the threshold value (θ) to get the comparator value *CURRENT*. The *CURRENT* value is then compared with the comparison function value of each observation to determine if the observation is an outlier or not. Hence instead of calculating the z-score value of each observation, in the proposed algorithm the *CURRENT* value needs to be calculated only once during each iteration.

Algorithm

Step 1: For the given spatial data set $X = \{x_1, x_2, \dots, x_n\}$ find the K nearest neighbors of each spatial point $x_i \in X$ and store their indexes in $KNBR(x_i)$.

Step 2: For each point $x_i \in X$, compute $c(x_i)$ as

$$c(x_i) = \frac{1}{k}\sum_{j \in knbr(x_i)} f(x_i, x_j) \tag{2}$$

Step 3: Compute the mean (μ) and standard deviation (σ) for the set $\{c(x_1), c(x_2), \dots, c(x_n)\}$

Compute the value of $CURRENT$ as

$$CURRENT = \theta \times \sigma + \mu$$

where θ is the threshold value

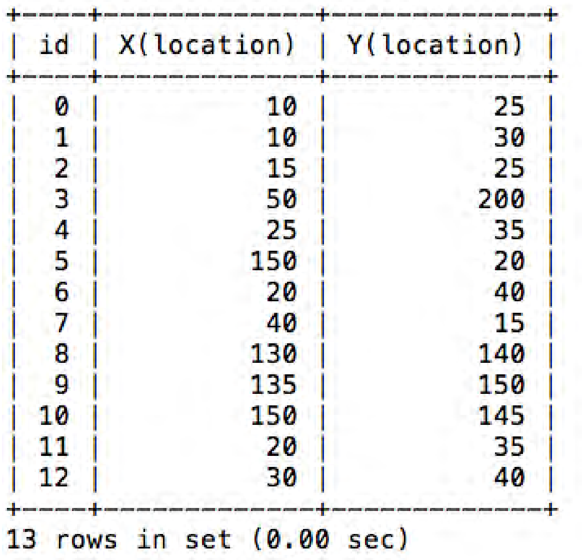
Step 4: For each $x_i \in X$, if $c(x_i) \geq CURRENT$, then x_i is an outlier.

Step 5: Ignoring the discovered outliers, find the current K nearest neighbors of each $x_i \in X$, and repeat steps 2, 3, 4 and 5 until no more outliers are found in Step 4.

5. IMPLEMENTATION OF THE PROPOSED ALGORITHM

This example illustrates the outlier detection using the algorithm described above. We consider an example that consists of Cartesian data, i.e. the spatial points are specified in the terms of Cartesian co-ordinates. For the ease of understanding we only consider the 1st Quadrant of the Cartesian system here. This example was implemented using the MySQL database management system, which features an in-built spatial extender, and the user interface was written in Java.

The dataset, in the form of a relation with a 2-dimensional attribute titled 'location', used for this example implementation is shown in Fig [1]



```
+----+-------------+-------------+
| id | X(location) | Y(location) |
+----+-------------+-------------+
|  0 |          10 |          25 |
|  1 |          10 |          30 |
|  2 |          15 |          25 |
|  3 |          50 |         200 |
|  4 |          25 |          35 |
|  5 |         150 |          20 |
|  6 |          20 |          40 |
|  7 |          40 |          15 |
|  8 |         130 |         140 |
|  9 |         135 |         150 |
| 10 |         150 |         145 |
| 11 |          20 |          35 |
| 12 |          30 |          40 |
+----+-------------+-------------+
13 rows in set (0.00 sec)
```

Fig [1]

Fig [2] illustrates the cluster formation of different spatial points from the given data. In this figure there are two clusters formed and each one is represented by a different color so as to distinguish among the clusters formed

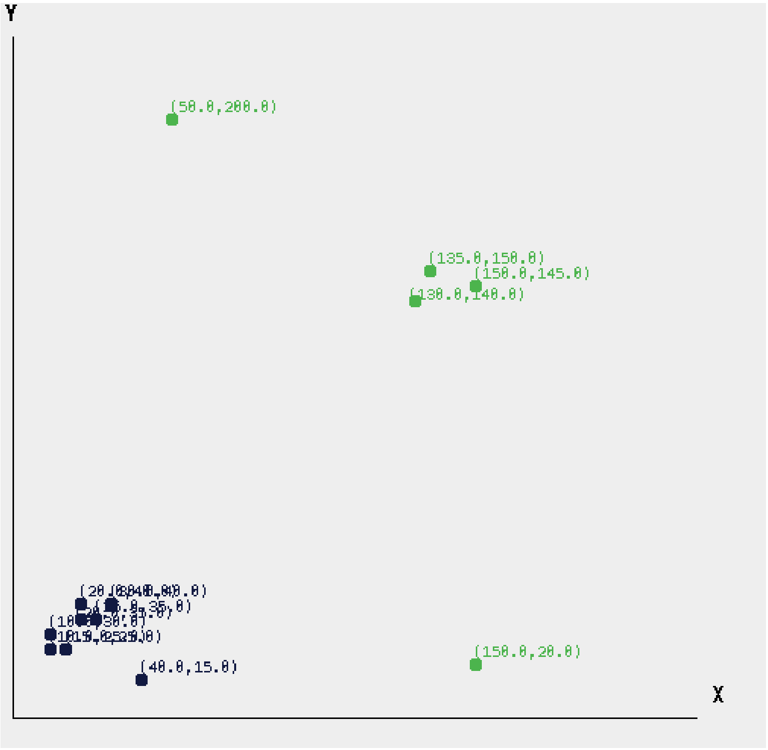


Fig [2]

Now we subject the above data to the outlier detection algorithm described in order to detect the outliers in it. The Fig [3] below shows the outliers present in the data, the encircled points (encircled only to illustrate) are the outliers for the given data set since they are significantly different from the other observations of the dataset.

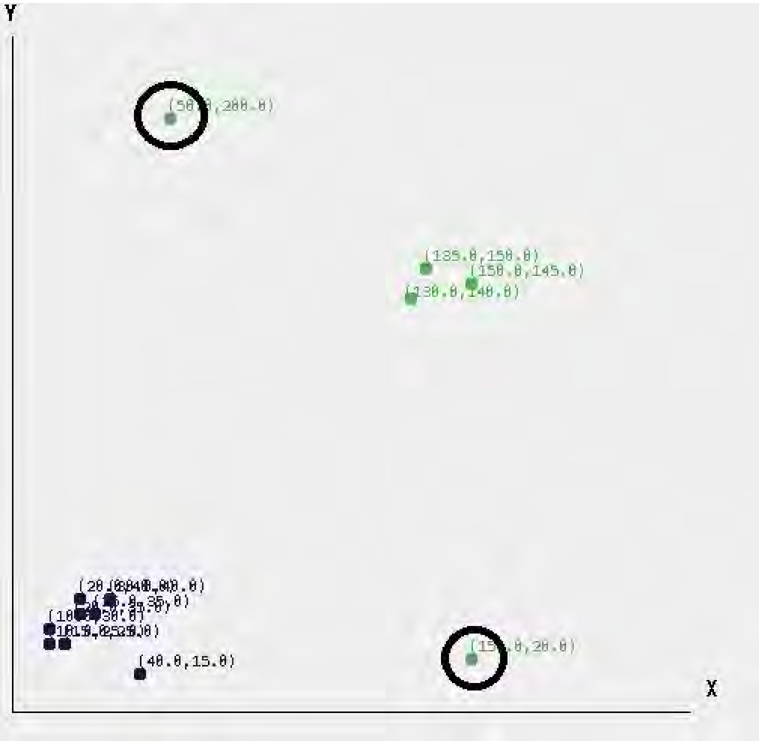


Fig [3]

Different operations can be performed (eg: deletion, modification, specialized testing, etc.) on the outliers identified depending upon the context where it is being applied. In this example we identify the outliers and eliminate them, which is used in noise removal.

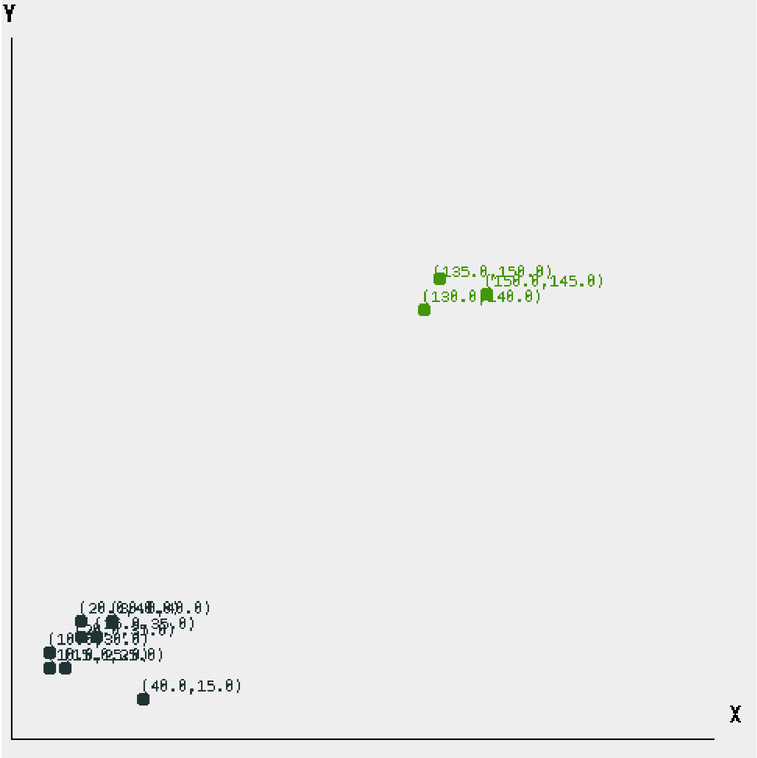
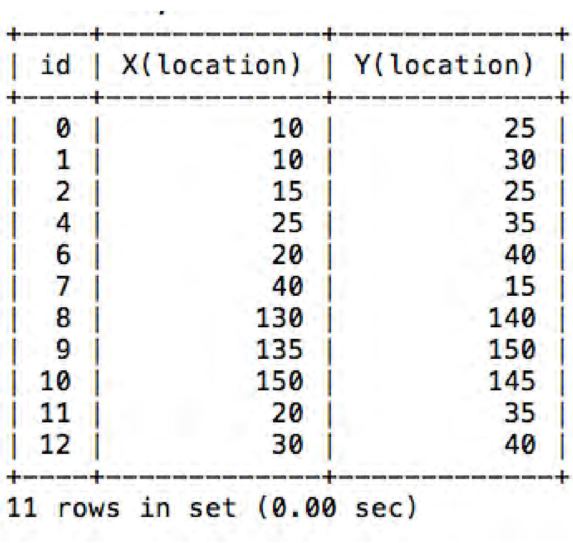


Fig [4]



```
+----+-------------+-------------+
| id | X(location) | Y(location) |
+----+-------------+-------------+
|  0 |          10 |          25 |
|  1 |          10 |          30 |
|  2 |          15 |          25 |
|  4 |          25 |          35 |
|  6 |          20 |          40 |
|  7 |          40 |          15 |
|  8 |         130 |         140 |
|  9 |         135 |         150 |
| 10 |         150 |         145 |
| 11 |          20 |          35 |
| 12 |          30 |          40 |
+----+-------------+-------------+
11 rows in set (0.00 sec)
```

Fig [5]

Thus after elimination of the outliers the obtained clustered spatial data is as shown in Fig[4]. The figure shows that the proposed algorithm correctly identifies the spatial outliers in the given data set. The relation/table after the outlier removal is shown in Fig[5].

9. Conclusion

We present an algorithm for outlier detection in spatial databases, which depends upon the K-nearest neighbors of each spatial point of the data set. It utilizes the neighborhood configuration of each spatial point to compute the value of a comparison function for each respective spatial point. A modified version of the z-statistic is used to evaluate the value for a single comparator, unlike in previous mechanisms that required evaluation of the z-score value for each observation. It is quite apparent from this and the previously detailed explanation that in this algorithm the overhead of calculation for each iteration is reduced from $n \times constant$ to $constant$, where *n* denotes the number of spatial points in the data set that have not been discovered as outliers. Though this is quite large in and of itself, considering spatial datasets are meant to contain a large amount of observations, this value can increase massively when the number of outliers is more.

10. References

[1] http://www.springerreference.com/docs/html/chapterdbid/62601.html

[2] Grubbs, F. E, "Procedures for detecting outlying observations in samples", Technometrics 11 (1), February 1969

[3] https://www.ctspedia.org/do/view/CTSpedia/OutLier