Volume 3, Issue 3, March 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Enhanced DBSCAN Outlier Detection

Priyamvada Paliwal
Software Engineering
I T M University
Sector 23-A, Gurgaon , India.

Meghna Sharma
Astt. Prof. Dept. of CS
I T M University
Sector 23-A, Gurgaon, India.

Abstract - Most real-world databases include a certain amount of exceptional values, generally termed as "outliers".The DBSCAN algorithm can identify clusters in large spatial data sets by looking at the local density of database elements, using only one input parameter. This paper presents a comprehensive study of Outlier Detection and DBSCAN , algorithm The salient of this paper to present enhanced DBSCAN algorithm with its implementation with the complexity. And there are also additional features described with this algorithm for finding outliers

Keywords – Density-based spatial clustering of applications with noise, Threshold Distance, Minimum Points, Cluster, Density.

I. Introduction

Outlier detection is currently very active area of research in data set mining community. Finding outliers in a collection of patterns is a very well-known problem in the data mining field. An outlier is a pattern which is dissimilar with respect to the rest of the patterns in the dataset. Proposed Method for outlier detection uses hybrid approach. Purpose of approach is first to apply clustering algorithm that is k means which partition the dataset into number of clusters and then find outliers from the section [10] from the each resulting clusters using distance based method. The principle of outliers finding depend on the threshold. Threshold is set by user. The main objective of the second stage is a finding out the objects, which are far away from their cluster centroids from the section[3]. In proposed approach, two techniques are combining to efficiently find the outlier from the data set. The experimental results using real dataset demonstrate that proposed method takes less computational cost and performs better than the distance based method. Proposed algorithm efficiently prunes of the safe cells (inliers) and save huge number of extra calculations**.** Outliers may introduce skew or complexity into models of the data, making it difficult, if not impossible, to fit an accurate model to the data in a computationally feasible manner. For example, statistical measures of the data may be skewed because of erroneous values, or the noise of the outliers may obscure the truly valuable information residing in the data set from the approach suggested by [2]. Accurate and efficient removal of outliers may greatly enhance the performance of statistical and data mining algorithms and techniques. Detecting and eliminating such outliers as a pre-processing step for other techniques is known as data cleaning. As can be seen, different domains have different reasons for discovering outliers: They may be noise that we want to remove.

II. Outlier Detection With The Help Of Dbscan Algorithm

A clustering algorithm that can do everything that DBSCAN can do is not yet available Various new clustering algorithms appear occasionally. DBSCAN has been modified to great extent recently and used to derive a new procedure to calculate EPS (threshold distance) which are most important parameters from the section[5]. The density-based clustering algorithm presented is different from the classical Density-Based Spatial Clustering of Applications with Noise (DBSCAN), and has the following advantages: first, Greedy algorithm substitutes for R*-tree in DBSCAN(Density based spatial clustering of application with noise) to index the clustering space so that the clustering time cost is decreased to great extent and I/O memory load is reduced as well; second, the merging condition to approach to arbitrary- shaped clusters from [12] is designed carefully so that a single threshold can distinguish correctly all clusters in a large spatial dataset though some density-skewed clusters live in it from the outliers in [4]. Finally, authors investigate a robotic navigation and test two artificial datasets by the proposed algorithm to verify its effectiveness and efficiency.

```
DBSCAN(D, eps, MinPts(minimum points))
  C = 0
  for each unvisited point P in dataset D
    mark P as visited
    NeighborPts = regionQuery(P, eps)
    if sizeof(NeighborPts) < MinPts
      mark P as NOISE
    else
      C = next cluster
      expandCluster(P, NeighborPts, C, eps, MinPts)

expandCluster(P, NeighborPts, C, eps, MinPts)
   add P to cluster C
   for each point P' in NeighborPts
      if P' is not visited
         mark P' as visited
         NeighborPts' = regionQuery(P', eps)
         if sizeof(NeighborPts') >= MinPts
            NeighborPts = NeighborPts joined with NeighborPts'
      if P' is not yet member of any cluster
         add P' to cluster C

regionQuery(P, eps)
return all points within P's eps-neighborhood
```

A. Complexity :

DBSCAN visits each point of the database, possibly multiple times (e.g., as candidates to different clusters) from the section [8]. For practical considerations, however, the time complexity is mostly governed by the number of region Query invocations. DBSCAN executes exactly one such query for each point, and if an indexing structure is used from [12] that executes such a neighborhood query in Ologn, an overall runtime complexity of O(n.logn) is obtained . Without the use of an accelerating index structure, the run time complexity is O(n2). Often the distance matrix of size (n2 – n)/2 is materialized to avoid distance recomputations. This however also needs O(n2)memory.

III. Implementation Of Dbscan Algorithm

Follow the below given path to run the code (i.e. netbeans):
1) Create a project name "DBSCAN1“
2) Under which create a package name as "dbscan",
3) It will show now path DBscan1-->src-->dbscan & now create a class as
same name as above given java files.
4) Under dbscan package create 5 classes -->dbscan.java,Gui ,java,Point.java Utility.java, DB.java
5) After saving all & try to run GUI.java where main() class is defined.

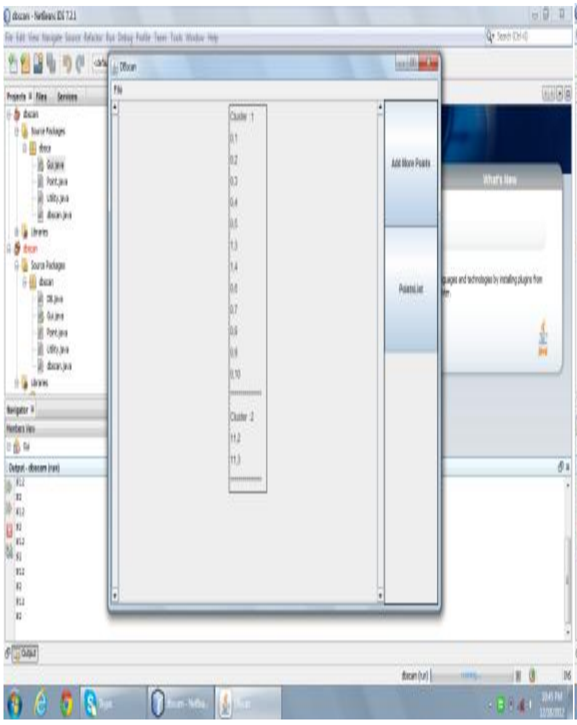
A. This section will focus on three primary components

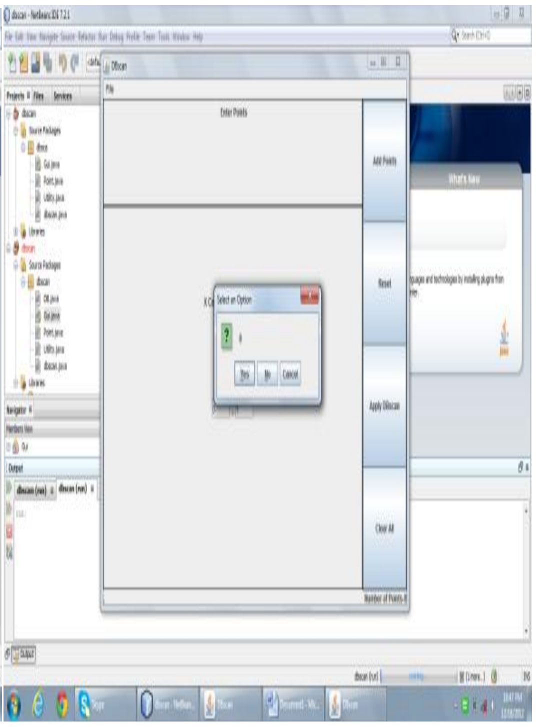
- CODE
- SCREENSHOTS OF DBSCAN ALGORITHM
- ADDITIONAL FEATURES

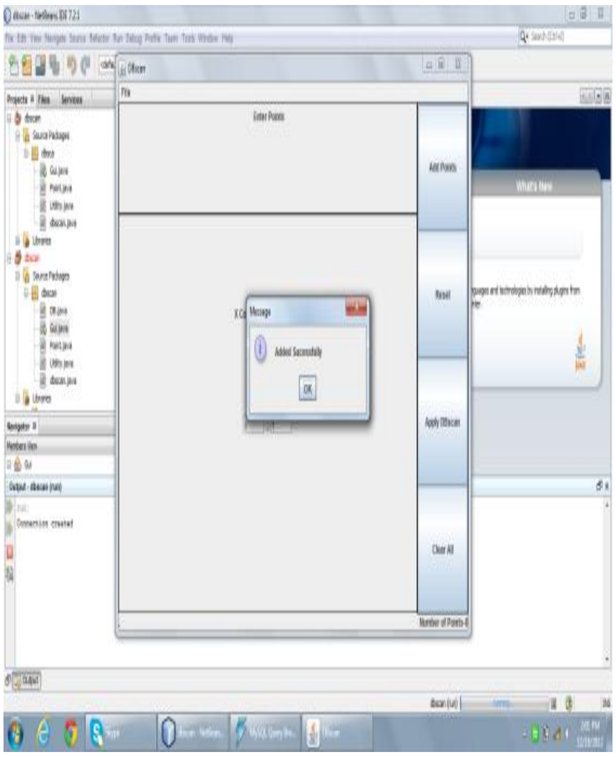
A.1 CODE

```
public static void insertF()
{
String x=tfx.getText();
String y=tfy.getText();
JOptionPane.showConfirmDialog(null,x);
double t=0,c=0;
if(x!=null && (!x.equals("")))
{
t= Double.parseDouble(x);

}
if(y!=null && (!y.equals("")))
{
c=Double.parseDouble(y);

}
DB d=new DB();
Connection con=d.getCont();
try
{
PreparedStatement ps=con.prepareStatement("insert into define values(?,?)");
ps.setDouble(1, t);
ps.setDouble(2, c);
double i=ps.executeUpdate();
if(i==1)
{
JOptionPane.showMessageDialog(null,"Added Sucessfully");
}
}
catch(Exception e)
{
JOptionPane.showMessageDialog(null,"Exception sorry");
}
}
```

A.2 Screenshots :

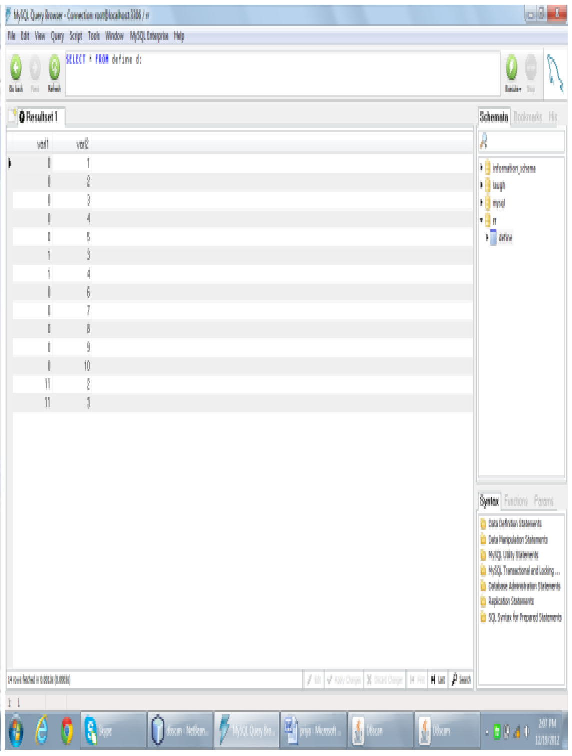






Added Sucessfully

```
Connection created
```


MySQL Query Browser - Connection: root@localhost:3306 / rr
File Edit View Query Script Tools Window MySQL Enterprise Help
Go back
Next
Refresh
Resultset 1
Schemata
Bookmarks
History
information_schema
laugh
mysql
rr
define
Syntax
Functions
Params
Data Definition Statements
Data Manipulation Statements
MySQL Utility Statements
MySQL Transactional and Locking ...
Database Administration Statements
Replication Statements
SQL Syntax for Prepared Statements

```
SELECT * FROM define d;
```

var1	var2
0	1
0	2
0	3
0	4
0	5
1	3
1	4
0	6
0	7
0	8
0	9
0	10
11	2
11	3

A.3 ADDITIONAL FEATURES ADDED:

1) DBSCAN does not require one to specify the number of clusters in the data a priori, as opposed to k-means.
2) DBSCAN can find arbitrarily shaped clusters. It can even find a cluster completely surrounded by (but not connected to) a different cluster. Due to the Minimum Points parameter, the so-called single-link effect (different clusters being connected by a thin line of points) is reduced.
3) DBSCAN has a notion of noise.
4) DBSCAN requires just two parameters and is mostly insensitive to the ordering of the points in the database. (However, points sitting on the edge of two different clusters might swap cluster membership if the ordering of the points is changed, and the cluster assignment is unique only up to isomorphism.)

IV. CONCLUSIONS

DBSCAN algorithm here considers only point objects but it could be extended for other spatial objects like polygons. Applications of DBSCAN to high dimensional feature spaces should be investigated and radius generation for this high dimensional data also has to be explored. It also fails to detect clusters with varied density. DBSCAN algorithm here considers only point objects but it could be extended for other spatial objects like polygons. Applications of DBSCAN to high dimensional feature spaces should be investigated and radius generation for this high dimensional data also has to be explored. It also fails to detect clusters with varied density.

Acknowledgment

We would like to give our sincere gratitude to our guide Ms. Meghna Sharma who guided us to pursue this topic and helped us to complete this topic. We would also like to thank her for providing us remarkable suggestions and constant encouragement. I deem it my privilege to have carried out my research work under her guidance.

REFERENCES

[1] Barnett, V., Lewis, T. (1995). *Outliers in Statistical Data*. Wiley, 3rd Edition.

[2] C. Aggarwal, P. Yu. Outlier detection for high dimensional data. In: Proc of SIGMOD’01, 2001:37-46

[3] M. M. Breunig, H. P. Kriegel, R. T. Ng, J. Sander. LOF: identifying density-based local outliers. In: Proc of SIGMOD’00, 2000:93-104

[4] S. Ramaswamy, R. Rastogi, S. Kyuseok. Efficient algorithms for mining outliers from large data sets. In: Proc of SIGMOD’00, 2000: 427-438.

[5] M. F. Jiang, S. S. Tseng, C. M. Su. Two-phase clustering process for outliers detection. Pattern Recognition Letters, 2001,
22(6/7): 691-700.

[6] Garfinkel, S. and H. Holtzman, *Understanding RFId Technology*, in *RFID: Applications, Security, and Privacy*, S. Garfinkel and B. Rosenberg, Editors. 2005, Addison Wesley. p. 15-36.

[7] Gray, J., Szalya, A.S. (2002) .The World-Wide Telescope, an Archetype for Online Science. MSR Technical Report MSR- TR-2002-75.

[8] Wikipedia (http://en.wikipedia.org/wiki/DBSCAN#Algorithm)

[9] Asif, Z. and Mandviwalla, M. (2005). Integrating the Supply Chain with RFID: A Technical and Business Analysis. Communications of the Association for Information Systems, 15, 393 – 427. Bolton, R. and Hand, D. (2002). Statistical fraud detection: A review. Statistical Science, 17(3), 235–255.

[10] F. Angiulli, C. Pizzuti. Fast outlier detection in high dimensional spaces. In: Proc of PKDD’02, 2002

[11] K. Yamanishi, J. Takeuchi, G. Williams. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. In: Proc of KDD’00, 2000: 320~325.

[12] Henrik Bäcklund (henba892), Anders Hedblom (andh893), Niklas Neijman (nikne866).A Density-Based Spatial Clustering of Application with Noise