Large Scale Efficient Clustering Using DBSCAN and Ensemble Techniques

D. Pradeep Kumar[1(✉)], B. J. Sowmya[1], R. Hanumantharaju[1], Anita Kanavalli[1], S. Seema[1], and K. N. Shreenath[2]

[1] Department of Computer Science and Engineering, M S Ramaiah Institute of Technology, Bengaluru, India
{Pradeepkumard,sowmyabj,hmrcs,anithak,seemas}@msrit.edu
[2] Department of Computer Science and Engineering, Siddaganga Institute of Technology, Tumkur, India
shreenathk_n@sit.ac.in

Abstract. Data clustering techniques are unsupervised machine learning techniques used in the field of data mining. Different clustering technique operate and perform differently based on the characteristics of input dataset. DBSCAN is a popular clustering technique with great ability to cluster datasets of arbitrary shapes and different sizes and densities using the principle of density estimation and noise removal. Density based clustering has proven to be very efficient and has found numerous applications across several domains. But DBSCAN does not perform well with clusters of similar densities and has a worst case run-time complexity of O(n^2) for high dimensional data. In this paper we try to overcome the drawbacks by developing SuperCube based Accelerated Density Based Spatial Clustering algorithm that detects the clusters by performing a dimensionality reduction that is, transforming high dimensional complex data into a lower dimensionality data and uses a unique combination of a virtual grid and employs representative points which reduces the time complexity to O (n log n). Results from the experiments with Datasets of varying sizes and dimensions are presented which proves that the proposed algorithm performs with greater accuracy and effectiveness.

Keywords: SuperCube · Ensemble techniques · Density based spatial clustering of applications · Superimposing

1 Introduction

Clustering is one of the most widely used methods and one of many effective data analysis techniques for unsupervised learning. It is a helpful approach that attempts to assemble the input data set with respect to any similarities into a list of finite ranges of semantically compatible groups. These algorithms, especially hierarchical algorithms, density-based algorithms, partitioned algorithms, graph-based algorithms, combinational algorithms, model-based algorithms and grid-based algorithms, can be loosely categorized into seven

A. K. Luhach et al. (Eds.): ICAICR 2020, CCIS 1393, pp. 622–636, 2021.
https://doi.org/10.1007/978-981-16-3660-8_58

groups. Density-based algorithms are noted for their concise description and hence the comparatively straightforward implementation of these varieties of algorithms. Another vital feature of this algorithm is that, even in an outlier data set, it is able to discover clusters of varying shapes and different sizes and does not require users to determine the number of clusters. DBSCAN defines denser area clusters and low density regions are classified as noise or outliers, providing high-quality output that depends on two specified parameters, Eps and MinPts. The basic principle behind such a cluster finding algorithm is that the neighbor points of the specified radius Eps would consist of the smallest number of points (MinPts) for each point in the cluster. An item is automatically picked by DBSCAN and evaluated only once. The neighborhood of the object is analyzed in such a manner that a cluster to which objects can be added later is created if it satisfies the minimum requirement for forming a cluster, and if the neighborhood objects do not fulfil the lowest threshold criterion, it is declared a noisy object. Many clustering tools are unable to classify clusters of arbitrary shapes, so DBSCAN has the bonus of being able to recognize a cluster of arbitrary shapes. The drawbacks of DBSCAN include the complexity of the worst case, tending to $O(n^2)$. Additionally, spatial indexing methods do not work efficiently for higher dimensional data, the runtime complexity grows from O (n logn) to O (n^2) for high dimensions. To overcome these issues, the SuperDBSCAN algorithm which is a combination of the SuperCube based Accelerated DBSCAN algorithm along with PCA (Principal Component Analysis) is proposed. This algorithm runs with a temporal complexity of O (n log n), even for higher dimensional data and uses a unique combination of a virtual grid, which is imposed on the input data and employs representative points, this significantly reduces the number of comparisons that need to be made, which translates into a significant run time speed up of up to 52.57% when compared to other proposed improvements. The PCA algorithm attempts to derive a low-dimensional set of features from a much larger set while still preserving as much variance as possible and also visualizing higher dimensional data. Another criticism of the DBSCAN algorithm has been that it is sensitive to its input parameters and MINPTS. The Super DBSCAN algorithm eliminates the need for the MINPTS parameter thus making it more accessible to non-expert users. The major contribution of this article is summarized as follows:

1. To reduce High dimensional to low dimensional data.
2. To identify and classify similar density clusters.
3. It maintains 100% accuracy of the original DBSCAN algorithm
4. It achieves a significant speed up in run time when compared to other improvements proposed for the DBSCAN algorithm.
5. It eliminates the need for the MINPTS input parameter, making the algorithm more user-friendly for non-expert users.

2 Literature Survey

DBSCAN was originally introduced by Ester and Kriegel, which was used as an underlying algorithm in many applications. A large amount of work has been done in the area of DBSCAN like parallel computing, dimensionality reductions, cloud computing, etc. to enhance and increase the accuracy and efficiency of DBSCAN clustering method. A research had been carried out to develop an algorithm which could improve the density calculations on the data set and thereby decreasing accuracy loss.

[1] Algorithm makes use of a ranked retrieval technique called WAND in order to improve the results of DBSCAN clustering. It further works by reducing the invoking times of WAND. [2] Another approach introduces in the modification of DBSCAN algorithm that was P- DBSCAN, used for the detection of geo tagged photos using the concept of adaptive density for fast convergence towards high density regions. In this method, the density threshold is also used as one of the factors for efficient working of DBSCAN. [3] Another methodology has been done in Linear DBSCAN calculation dependent on area delicate hashing. The fundamental territories of this paper incorporate including more info boundaries and lacking of over-simplification. Not at all like the first DBSCAN, this procedure utilizes LSH which requires quicker locale inquiry for k neighbors of an information point. This paper incorporates a seed point determination strategy, which depends on impact space and neighborhood closeness, to choose some seed focuses rather than all the area during bunch development is utilized in this work. Therefore, the quantity of district inquiries can be diminished, in this way guaranteeing better calculation precision. [4] An equal DBSCAN bunching calculation was likewise executed in past works for treatment of the huge scope information preparing utilizing the huge information handling stage called Spark. Flash, as another age of quick broadly useful motor for huge scope information handling, gives strong circulated dataset reflection for information stockpiling that dispenses with the requirement for middle outcomes to be shipped off the disseminated document frameworks, and thus it improves continuous information preparing. [5] presents the dimensionality decrease strategies and their appropriateness for different kinds of information and application zones. In this paper they have clarified the Linear Dimension Reduction Technique (LDRTs) and Non-Direct Dimension Reduction Technique (NLRDTs) which thusly have a few administered, solo and semi-managed methods to accomplish low dimensionality. When done a similar report among LDRT and NLRDT can presume that LDRT requires less calculation force and cost. [6] Myat Cho Mon Oo and Thandar Thein have proposed a proficient prescient examination framework for high dimensional enormous information by improving adaptable irregular forest calculation on Apache Spark stage. The viability of the proposed framework is inspected on five-genuine world datasets and results showed that the proposed framework accomplishes profoundly serious execution contrasted and RF calculation actualized by Spark MLib [7]. In this paper, a strategy for coordinating dull information is proposed and Nonnegative grid factorization is applied to the bunching gathering model dependent on dim information. From the start the distinctive base grouping results are acquired by utilizing different bunching designs and NMF is applied to get coordinated outcomes which fills in as appeared in the underneath graph. Test results show that the strategy beats other bunching group procedures [8]. This examination presents an exhaustive

investigation of DBSCAN calculation and the improved adaptation of DBSCAN calculation with its usage utilizing mat lab. In this cycle, the information has been isolated impeccably, at that point new testing strategy will applied so as to decrease the thickness of thick information and get information with just a single thickness circulation (meager information), the consequences of examining were viable. They have utilized three strategies to get the normal yield which resembles the figure underneath. [9] Ll Meng'ao and MENG Dongxue, *GU Songyuan, LIU Shufen proposed a paper so as to conquer the time cost of the calculation dependent on lattice cell based calculation. This strategy has been checked tentatively that DBSCAN calculation dependent on matrix cells shows higher exactness and lower time multifaceted nature. [10] In this paper the specialists have proposed a method that decreases the time unpredictability of dbscan calculation contrasted with the first calculation with a period intricacy of O(N). They have proposed a calculation with LSH where It looks for the surmised Nearest Neighbor Points rather than the exact Nearest Neighbor Points and LSH restores a few purposes of the circle whose middle is p and the range is $(1 + \varepsilon)$d. ε is doled out by the clients and $\varepsilon > 0$. [11] This paper which focuses on a new method to solve dimensionality problems where clustering is integrated with correlation measure to produce good feature subset. Evaluated computational time and accuracy of proposed method with Relief and IG methods. Relief (Kira and Rendell, 1992) present a feature selection approach based on instance based attribute ranking scheme called RELIEF algorithm. It deals with incomplete, noisy and multiclass datasets. [12] This paper clarifies about another closeness measure that represents excess of information that are dissipated a similar way from a given point. The run-time assessment shows that both the separation based and common neighbor based thickness gauges have a similar quadratic time unpredictability as for dataset size. Examination of commotion impacts showed that both DBSCAN and CI-DBSCAN have a similar power to the clamor. [13] The archive joins thickness tops grouping and gravitational pursuit strategy to upgrade information bunching which proposes an altered component of choosing the cut-off separation.

[2] The introduced strategy chooses the enhanced starting bunch communities then the age of the underlying populace is executed dependent on the arrangement of applicant groups. They give an improved instrument on boundary choice which will expand the exhibition of grouping. [14] This paper clarifies about building up another DBSCAN calculation for assessing the quantity of groups by advancing a probabilistic cycle, in particular DBSCAN-Martingale, which includes arbitrariness in the choice of thickness boundaries and lessen the quantity of emphases needed to separate all bunches. This work fundamentally centers around logical recipe for the normal number of separated bunches per DBSCAN-acknowledgment by outlining with the time administrator and age in thickness based grouping. [15] The archive clarifies about another upgraded DBSCAN-Martingale probabilistic cycle calculation for assessing the quantity of groups, which includes arbitrariness in the choice of thickness boundaries and lessen the quantity of emphases needed to remove all bunches. The tests acknowledged in this work, in the covering bunch altering issue one can decrease the quantity of between group edges by covering at least two bunches. At that point, it might be smarter to embed vertex in more than one group than to eliminate edges from that vertex.

Grouping calculations are alluring for the assignment of class recognizable proof in spatial information bases. In any case, the notable calculations experience the ill effects of serious downsides when applied to huge spatial information bases, most of which could be the expanding time unpredictability and finding the groups for high dimensional information.

3 Design

The idea aims at finding a better and suitable technique for the clustering purpose in regard with DBSCAN. The propose a method to reduce high dimensional data to low dimensional data and thereby reducing the time complexity. The aim of this work is to design such an algorithm which could be used to increase the efficiency of finding clusters of high dimensional datasets using the DBSCAN approach. Also, it reduces the time complexity for the same. It defines all the necessary modules used for this approach of clustering, for example, with respect to the data analysis for handling high dimensional data is done using PCA technique which projecting the high dimensional data in a space for lower dimensional subspace, data pre-processing techniques involves outlier detection, handling missing values of attributes, redundant data and similar density data. The initial step in our work includes the data pre-processing. Data pre-processing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that the pre-process our data before feeding it into our model. Standardize our dataset in such a manner that obtain equal values for the given data, that is convert the strings to integers (Fig. 1).

Unsupervised Learning a Machine Learning technique that has been pre-owned in our idea. This technique has various applications such as segmenting datasets by some shared attributes. Detecting anomalies that do not fit to any group, and simplifying datasets by aggregating variables with similar attributes. The objective of clustering is to find different groups within the elements in the data. To do so, clustering algorithms find the structure in the data so that elements of the same cluster (or group) are more similar to each other than to those from different clusters. Ensuing the data pre-processing takes place the data is split into two sets: the training and the testing data. It is based on a number of points with a specified radius ε and there is a special label assigned to each datapoint. The process of assigning this label is the following:

- It is a specified number (MinPts) of neighbour points. A core point will be assigned if there is this MinPts number of points that fall in the ε radius.
- A border point will fall in the ε radius of a core point, but will have less neighbours than the MinPts number.
- Every other point will be noise points.

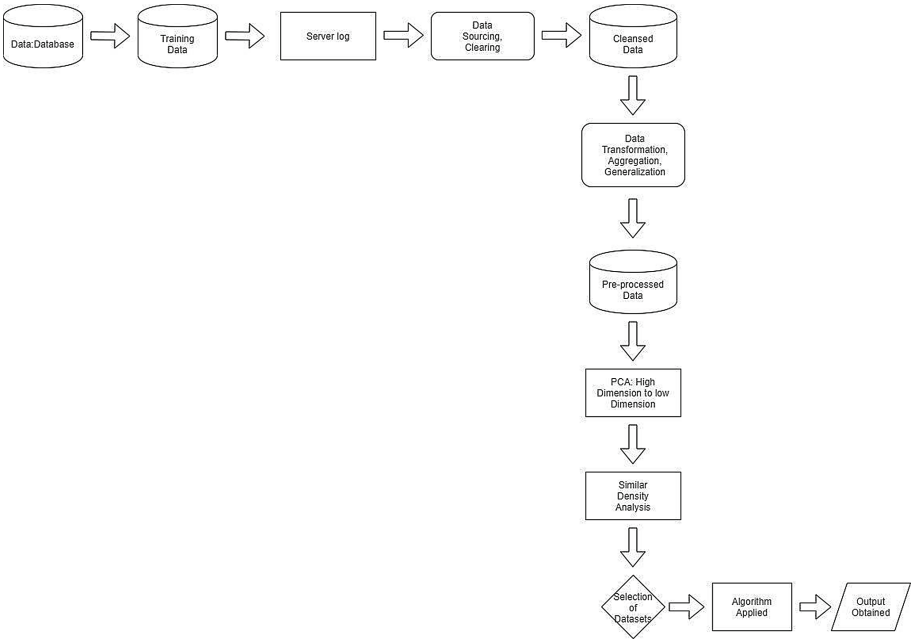

Fig. 1. Proposed framework for the efficient clustering

4 Implementation and Results

Usage of a novel methodology known as SuperCube based Accelerated Density Based Spatial Clustering for Applications, in this algorithm. Here, our algorithm executes with a time complexity of O(nlogn), even for higher dimensional data. It unstratified a virtual cube on the given data set and uses the idea of representative points to reduce the number of comparisons. It provides significant run time speed up when compared to other proposed improvements and eliminates the need for one input parameter MINPTS, thus making it easier to use for naive users.

A series of steps have been followed to obtain accurate results. Commencing the process of data processing, followed by merging conditions and then the superimposing supercube to choosing the representation points.

Pre-processing

Perform a pre-processing step on the input data and sort the data set according to one of the dimensions. For example, a two dimensional data set is first sorted according to the X coordinates and then the result is sorted according to the Y coordinates. Hence this approach of data sorting is extensible for any dimension. This pre-processing speeds up the supercube allocation as explained below.

The Merging Condition
As explained below the algorithm checks if the distance between two points is less than the input parameter. If it is, then the two supercubes that these two points belong to are merged to belong to the same cluster.

Superimposing Super Cubes
One of the key ideas of this proposed algorithm is the formation of the Supercubes. These Supercubes are contributory in providing us an execution speed improvement via the original DBSCAN algorithm. A supercube is an n-dimensional analogue of a square (n = 2) and a cube (n = 3). A Supercube has all the properties that a cube has in three dimensional space, but in n dimensional space, it is a closed, compact, convex figure whose 1-skeleton consists of groups of opposite parallel line segments aligned in each of the space's dimensions, perpendicular to each other and of the same length. So in 1-D a Supercube is a line segment, a square in 2-D, a cube in 3-D, a tesseract in 4-D and so on. Supercubes for the first four dimensions are shown in Fig. 3. Based on the dimensionality of the data, overlay a virtual Supercube on the points such that the length of the space diagonal of the Supercube is.

Deciding the area in space where the 2-D grid needs to be constructed. For this define the boundary by taking the minimum and maximum of X and Y coordinates respectively. Once the scope of the grid is defined, construct the boxes and superimpose this grid on the original data set. The key idea for the construction of this grid is that, construct it in such a way that every point in a particular box is guaranteed to belong to the same cluster. This provides us a significant speed up, as instead of checking each and every point against each point in the data set as in the original DBSCAN algorithm, just need to check if any one point in the box satisfies the merging condition, if it does all the points belonging to that particular box are guaranteed to satisfy the merging condition too. This property is guaranteed by the virtue of creation of the boxes. Choose the diagonal of each box to be equal to the parameter that is input to the algorithm, thus create boxes of length = breadth = $/\sqrt{2}$. Therefore, the maximum distance between any two points inside the box is never greater than, consequently they belong to the same cluster. This approach scales with dimensions. For two dimensional data set construct a grid consisting of flat boxes, for three dimensional data, create a grid of cubes where the length of the space diagonal is equal to and so on.

Choosing Representative Points
For n dimensional data, need 2(n + 1) representative points. For sake of explanation, consider two dimensional data. Hence for each face of the grid, define eight Representative Points. These points are labelled as follows:

- Top
- Top Right
- Right
- Bottom Right
- Bottom
- Bottom Left

- Left
- Top Left

These points are the boundary points of the box.

For example, the Top point is the top-most point inside the box. Use a token ring approach to distribute points to these eight positions which is explained here. To decide which box does the point belong to, divide the point by $/\sqrt{2}$ in each dimension to obtain the corresponding band in which the point lies. The intersection of these bands gives us the box in which the point lies. In case the grid does not start from the origin, perform a origin shift transformation. If any points are found to lie exactly on the edge of the box round up the division. As the first point is entered into a new box, all of these eight positions are initialized to that point. An ideal position with respect to representative points is defined as a case where representative point lie on the edge of the box as shown in Fig. 2. Now whenever a new point is encountered, each position calculates the Euclidean distance between the ideal position and the current point and the point being examined. If a new point is found to have a smaller distance, the corresponding representative point is updated with the new point. Obviously multiplicity is allowed, that is one point can represent multiple positions. This process is repeated for all the points belonging to a particular box.

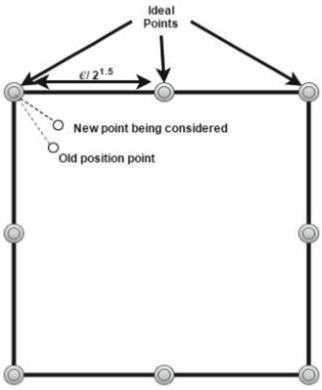

Fig. 2. The distance between the new point and the ideal position is compared with the distance between the representative point and the ideal position, if the former is lesser the new point is updated as the new representative point.

Depth First Search

Now to implement the main clustering; traverse the entire grid in a depth first fashion. Start with the box closest to the shifted origin. At any given point of time, compare the two closest representative points between boxes. For example, in a two-dimensional space, begin by checking if the top representative point of the current box i is within an - distance of the Bottom representative point of the top neighbor box j. Traversal in a clockwise fashion. If the distance between corresponding points is less than, the

merging condition is said to be satisfied and all the points in both these boxes are labeled to belong to the same cluster. If the merging condition fails, the next box in the DFS traversal is checked and the cluster IDs are not updated.

Layering

Two points belong to the same cluster if the distance between them is less then, the location of the points inside the boxes are not known. Therefore, it is entirely possible that the distance between points in two consecutive boxes can be less. That is, consider only neighboring boxes in our depth first traversal, the j-layer; bound to miss points that are at a distance which is less than. Consequently, the accuracy of the algorithm suffers. To overcome this problem, introduce a concept of layering. If the traversal for the j-layer box returns a failure, check for the (j + 1) th layer box in the same direction, subject to the condition that the distance between the representative points is less than. Check the $(j + 1)^{th}$ layer box only for non-diagonal neighbors which again reduces the number of iterations. This can be done because the grids are designed in such a way that the diagonal of each box is equal to the value of, therefore two points in the diagonal direction cannot be at a distance less than and not lie in consecutive boxes.

Density based clustering techniques especially DBSCAN have found innumerable applications across domains. The Traditional DBSCAN algorithm has a time complexity of $O(n^2)$ which reduces to O (n log n) when spatial indexing. However, the complexity again rises to $O(n^2)$ for high dimensional data. Here, proposed a new algorithm, Super DBSCAN, to overcome the drawbacks of DBSCAN. This algorithm runs with a temporal complexity of O (n log n) which uses a unique combination of a virtual grid, that is imposed on the input data and employs representative points which reduces the number of comparisons that need to be made and in turn reduces the time complexity. Our projected algorithm is an easy but efficient and effective algorithm and it boosts the performance of DBSCAN in adjacent clusters with different densities.

Algorithm 1:
Principle Component Analysis
PCA algorithm(X, k): top k eigenvalues/eigenvectors
X = N × m is a data matrix,
every point in data xi = column vector, I =1 to m
- X mean x is subtracted from each column vector xi in the X
- Σ XXT … covariance matrix of X
- { λi , ui }i=1..N = eigenvectors/eigenvalues of Σ ... λ1 ≥ λ2 ≥ … ≥ λN
- Return { λi , ui }i=1..k

% top k principal components

Algorithm 2:
SuperCube creation and initialization
function: GENERATE-SuperCUBE(data set D,)
input:data set D containing all input points and user input parameter
output: SuperCubeDetailsNbPos : coordinates are considered as key and its updated position details as value
foreach dimension i in n **do**
//n is number of dimension of data set D which is globally defined iBand ← generate steps of √2 by incrementally increasing from shifted origin to boundary of the Supercube container for every axis ;
End
foreach value v in every iBand **do**
SuperCubeList ← generate all the combination of Supercube for v;
end
foreach SuperCubePoint k in SuperCubeList **do**
Position ← IDENTIFY-POSITION-OF-SuperCUBE(k); SuperCubeDetailsNbPos ← APPEND(k , NEIGHBOURING-POINTS(k), Position);
End
return SuperCubeDetailsNbPos;

Algorithm 3:
Computation of Representative Points
function: COMPUTE-REPRESENTATIVE-POINTS(data set D,)
input: data set D containing all input points and user input parameter
output: SuperCubeDetailsRepPts: coordinates are considered as key and its updated points as value
Foreach data point j in data set D containing data points **do** IdentifiedSuperCube ← FIND-CORRESPONDING-SuperCUBE-OF-POINT(j);
Status ← CHECK-VISITED(IdentifiedSuperCube)
if Status is not visited **then** // visiting for the first time
initialize all the $2^{(n+1)}$ representative points to the current j point; // n is number of dimension for data set D which is globally defined

end
else
foreach representative point l ∈ set($2^{(n+1)}$ representative points) **do**
compare euclidean distance between the ideal point-and-the current point represented by a and ideal point-and-representative point l represented by b ;
if a < b **then**
updated representative data point l ← current data point j ;
end
end
end
SuperCubeDetailsRepPts ← insert the current point j in IdentifiedSuperCube with its updated $2^{(n+1)}$ representative points ;
End
return SuperCubeDetailsRepPts;

Algorithm 4:
Clustering mechanism for SuperCube
function: CLUSTERING-FUNCTION(SuperCubeDetails, CurrentSuperCube)
input: CurrentSuperCube:Supercube under consideration and SuperCubeDetails:dictionary with Supercube coordinate as key and its updated neighbouring points, representative points, position details as value
output: ClusterList = contains list of complete number of points and cluster in every cluster **if** CurrentSuperCube is not visited **then**
if CurrentSuperCube is opened **then**
foreach NeighbouringSuperCube of CurrentSuperCube **do**
if distance between corresponding representative points < for NeighbouringSuperCube in jth layer **then**
$ClusterID_{NeighbourSuperCube} \leftarrow ClusterID_{CurrentSuperCube}$; //merging cluster
Mark CurrentSuperCube as visited;
CLUSTERING-FUNCTION(NeighbourSuperCube, SuperCubeDetails); // recursive call
End
else
if the NeighbouringSuperCube is not a diagonal Supercube **then**
if distance metric between respective representative data points < for NeighbouringSuperCube in (j + 1)th layer of the Supercube **then**
ClusterIDNeighbourSuperCube ← ClusterIDCurrentSuperCube; // merging cluster
Mark CurrentSuperCube as visited; CLUSTERING-FUNCTION(NeighbourSuperCube, SuperCubeDetails); // recursive call
end
else
check for the next NeighbouringSuperCube;
end
end
else
check for the next NeighbouringSuperCube ; // distance greater than , merging condition violated
end
end
end
end
else
do nothing; // check for the next Supercube in next function call
end
end
else
do nothing; // check for the next Supercube in next function call
end
return ClusterList;

Algorithm 5:
SuperCube based Accelerated DBSCAN
Function: SuperCUBE-BASED-ACCELERATED-DBSCAN(data set D,)
input:data set D containing all input points, user input parameter indicating the density of the clusters needed.
Output: Reduced Data obtained from PCA shifted to LastClusterList.
LastClusterList = aggregation of individual ClusterList for each Supercube initialize
CurrentClusterID=1; SuperCubeDetails1 ← GENERATE-SuperCUBE(data set D,);
SuperCubeDetails2 ← COMPUTE-REPRESENTATIVE-POINTS(data set D,);
SuperCubeDetails ← SuperCubeDetails1 + SuperCubeDetails2
foreach Supercube k in SuperCubeDetails **do**
if k is not visited and k has some points **then**
mark k as visited;
ClusterIDk ← CurrentClusterID;
LastClusterList = LastClusterList + CLUSTERING-FUNCTION(SuperCubeDetails, current Supercube k,)
CurrentClusterID = CurrentClusterID+1;
end
else
pass to the next Supercube in SuperCubeDetails
end
end
return LastClusterList;

To check the time and the efficiency of the proposed SCA-DBSCAN algorithm, conduct multiple experiments. To verify the results, run the DBSCAN algorithm and the FastDBSCAN algorithm on the same data sets with the same input parameters and compare the accuracy of the results as well as the time taken by each algorithm to achieve those results. Run the algorithm on three different data sets that represent three different types of data: sparsely distributed data (synthetic data set 1), tightly coupled data and data with arbitrary shape that classical partition based algorithms fail to identify (synthetic data set 2). After performing the proposed algorithm on the obtained data sets can observe the following results: DATA SET 1.

On comparing both the algorithms, can conclude that the number of clusters formed are 5 with an accuracy of 0.166 for the original DBSCAN while, the proposed algorithm gives us an accuracy of 0.85714 and the time taken is 0.722 s with the number of points obtained individually for each cluster is 50, which results in a tightly coupled data as shown in Fig. 3.

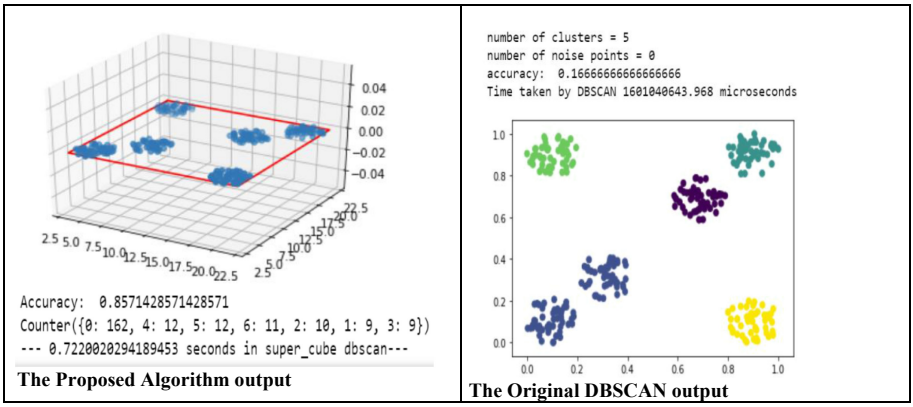

Fig. 3. Output for dataset 1

On comparing both the algorithms the number of clusters formed is 1 with an accuracy of 0.0667 for the original DBSCAN while the proposed algorithm gives us an accuracy of 1.0 and the time taken is 0.3517 s with the number of points obtained is 116, which results in a sparsely distributed data as shown in Fig. 4.

On comparing both the algorithms the number of clusters formed is 0 with an accuracy of 0.0 for the original DBSCAN while the proposed algorithm gives us an accuracy of 0.9714 and the time taken is 1.062 s with the number of points obtained is 1, which results in a sparsely coupled data and as shown in Fig. 5.

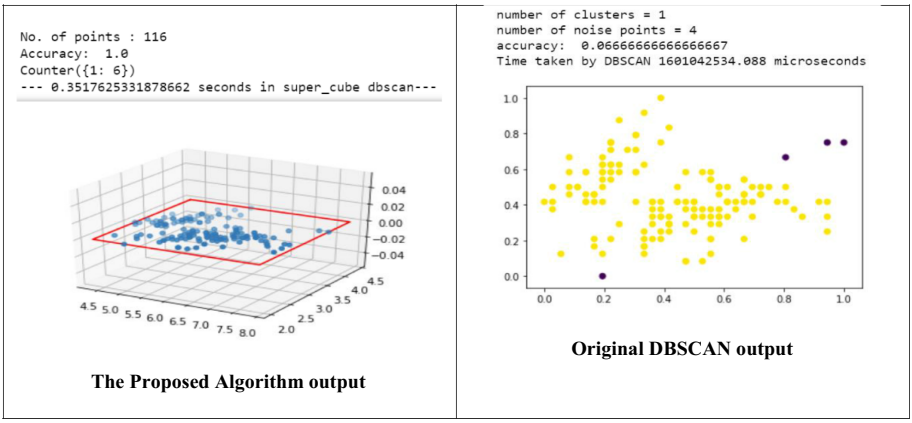

Fig. 4. Output for dataset 2

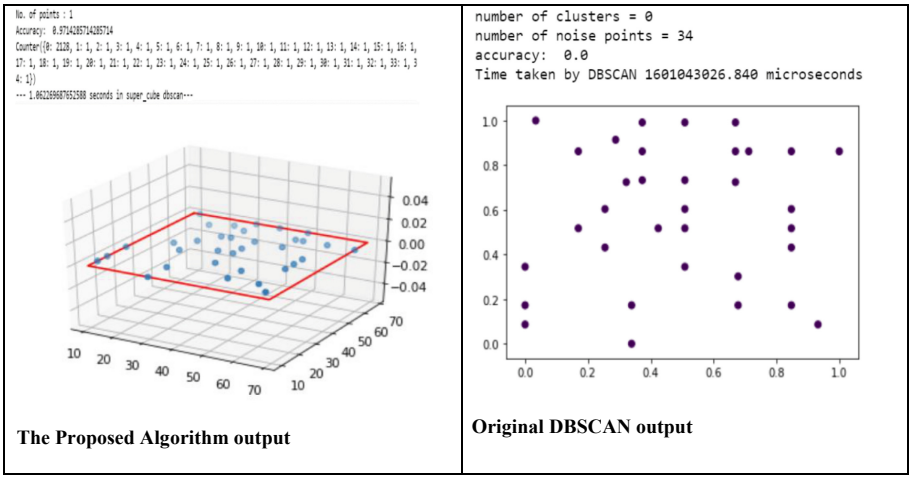

Fig. 5. Output for dataset 3

On comparing both the algorithms the number of clusters formed is 3 with an accuracy of 0.156 for the original DBSCAN while the proposed algorithm gives us an accuracy of 0.8 and the time taken is 0.627 s with the number of points obtained as 50 for cluster 1, 2 and 3 while cluster 4 consists of 42 points, which results in an arbitrary size and shaped data and as shown in Fig. 6.

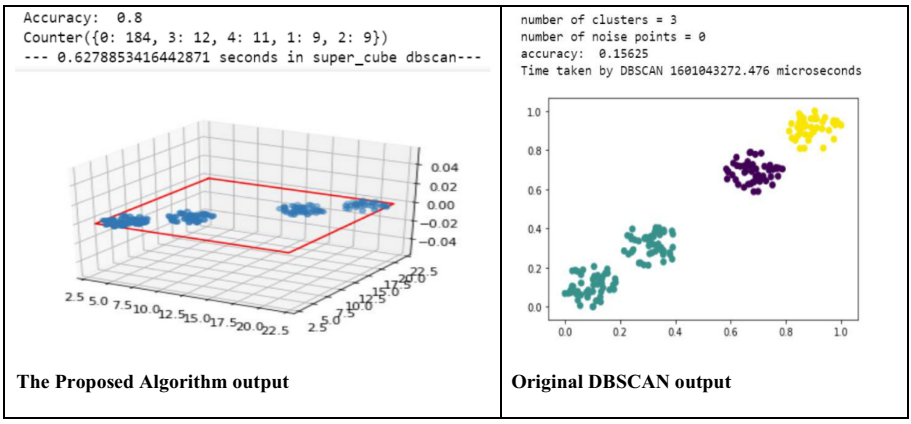

Fig. 6. Output for dataset 4

5 Conclusion

Established an algorithm that centralizes around the drawbacks of a common machine learning technique known as DBSCAN. Through this user a naive user can competently lay hands on understanding the over comings of DBSCAN through a transparent algorithm that have implemented. Conducted a basic comparison with the supercube DBSCAN algorithm and the original DBSCAN where the accuracy and the time complexity are displayed clearly. The main concept of the DBSCAN algorithm is to locate regions of high density that are separated from one another by regions of low density. So it would suggest that while dealing with spatial clusters of different density, size and shape, it could be challenging to detect the cluster of points. The task can be even more complicated if the data contains noise and outliers. DBSCAN tends to be slower when working with large datasets since computation on similar datasets becomes time consuming as the datasets increase. Our research shows how it could be improvised or modified in a way that it identifies outliers easily with anomaly detection and also handles the large datasets thereby reducing the time complexity and also reducing the dimensionality. Aim at incorporating Dimensional reduction so as to increase the cluster performance for our work. Our work proposes a simple yet improvised version of DBSCAN which not only reduces the time complexity but increases the effectiveness and efficiency of the clustering altogether.

References

1. Mehta, J., Mathur, V., Sanjay, S.: HCA-DBSCAN: SuperCube based accelerated density based spatial clustering for applications with noise. IEEE (2019)
2. Zhang, Y., Wang, X., Li, B., Chen, W., Wang, T., Lei, K.: Dboost: a fast algorithm for DBSCAN-based clustering on high dimensional data. In: Bailey, J., Khan, L., Washio, T., Dobbie, G., Huang, J.Z., Wang, R. (eds.) Advances in Knowledge Discovery and Data Mining. LNCS (LNAI), vol. 9652, pp. 245–256. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31750-2_20

3. Schikuta, E.: Grid clustering: an efficient hierarchical clustering method for very large data sets. In: Proceedings of 13th International Conference on Pattern Recognition, vol. 2, pp. 101–105 (1996)
4. Kisilevich, S., Mansmann, F., Keim, D.: P-DBSCAN: a density based clustering algorithm for exploration and analysis of attractive areas using collections of geo-tagged photos. In: Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research and Application (pp. 1–4) (2010)
5. Huang, F., et al.: Research on the parallelization of the DBSCAN clustering algorithm for spatial data mining based on the spark platform. Remote Sens. **9**(12), 1301 (2017)
6. He, Q., Gu, H.X., Wei, Q., Wang, X.: A novel DBSCAN based on binary local sensitive hashing and binary-KNN representation (2017)
7. Ayesha, S., Hanif, M.K., Talib, R.: Overview and comparative study of dimensionality reduction techniques for high dimensional data. Inf. Fusion **59**, 44–58 (2020)
8. Oo, M.C.M., Thein, T.: An efficient predictive analytics system for high dimensional big data. J. King Saud Univ.-Comput. Inf. Sci. (2019)
9. Ye, W., Wang, H., Yan, S., Li, T., Yang, Y.: Nonnegative matrix factorization for clustering ensemble based on dark knowledge. Knowl.-Based Syst. **163**, 624–631 (2019)
10. Kumari, A., Shrivastava, V., Pandey, A.: Reduction of DBSCAN time complexity for data mining using parallel computing techniques (2019)
11. Meng'Ao, L., Dongxue, M., Songyuan, G., Shufen, L.: Research and improvement of DBSCAN cluster algorithm. In: 2015 7th International Conference on Information Technology in Medicine and Education (ITME), pp. 537–540. IEEE (2015)
12. Wu, Y., Guo, J., Zhang, X.: A linear DBSCAN algorithm based on LSH (2007)
13. Chormunge, S., Jena, S.: Correlation based feature selection with clustering for high dimensional data. J. Electr. Syst. Inf. Technol. **5**(3), 542–549 (2018)
14. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density- based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, AAAI Press, 1996, pp. 226–231 (1996)
15. Guha, S., Rastogi, R., Shim, K.: CURE: a~ efficient clustering algorithms for large databases. In: Proceedings of ACM SIGMOD International Conference on Management of Data, Seattle, WA, 1998, pp. 73–84 (1998)
16. https://www.geeksforgeeks.org/dimensionality-reduction/?ref=lbp
17. A fast clustering algorithm to cluster very large categorical data sets in data mining, In: Proceedings of SIG- OD Workshop on Research Issues on Data Mining and Knowledge Discovery, Tech. Report 97–07, UBC, Dept. of CS (1997)
18. https://stackabuse.com/dimensionality-reduction-in-python-withr/