CHAPTER 6
A MODIFIED DBSCAN ALGORITHM

6.1 Introduction

The hierarchical and partitional clustering algorithms are not significant in producing the clusters of variable densities. The density-based clustering algorithms have been effective in this direction. In this chapter, we propose, a novel approach to improve the computational complexity of DBSCAN algorithm. We achieve this using the prototypes produced from the squared error clustering method such as $K$-means.

6.2 Terminologies

We first discuss the useful terminologies for understanding the proposed method.

6.2.1 DBSCAN Method

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [29] is one of the renowned density based clustering technique to cluster the data of arbitrary shapes. A cluster in this model is described as a linked region that exceeds a given density threshold [118]. The functioning of DBSCAN is directed by two well known definitions, namely, density-reachability and density-connectability, which depend on two predefined parameter values known as the size of the neighborhood ($\varepsilon$) and the minimum number of points in the neighborhood ($N_{\text{min}}$). For the sake of defining a cluster of DBSCAN, the following definitions [22] are used. Let the Euclidean distance between two points $x$ and $y$ is denoted by $d(x, y)$.

Definition 1. ($\varepsilon$-neighborhood): The $\varepsilon$-neighborhood of $x$ with respect to the given set $S$ of $n$ data points is denoted by $N_{\varepsilon}(S, x)$ and is defined as the set of all points from
6.2.1 DBSCAN Method

$S$ which are at less than or equal to $\varepsilon$-distance from $x$. \textit{i.e.} $N_\varepsilon(S, x) = \{ y \in S \mid d(x, y) \leq \varepsilon \}$. 

\textbf{Definition 2. (Core point):} A point $x \in S$ is said to be a core point if $|N_\varepsilon(S, x)| \geq N_{\text{min}}$.

\textbf{Definition 3. (Border point):} A point $x \in S$ is a border point if $|N_\varepsilon(S, x)| < N_{\text{min}}$ and $x \in N_\varepsilon(S, y) \forall y$ is a core point.

\textbf{Definition 4. (Directly density-reachable):} A point $x$ is said to be directly density-reachable from a point $y$ with respect to the given set $S$, neighborhood value $\varepsilon$ and the minimum limit on the number of points $N_{\text{min}}$ if the below two conditions are satisfied.

1. $x \in N_\varepsilon(S, y)$
2. $|N_\varepsilon(S, y)| \geq N_{\text{min}}$

\textbf{Definition 5. (Density-reachable):} A point $x$ is said to be density-reachable from a point $y$ if there exists a sequence $x_1, x_2, \ldots, x_i$ where $x_1 = x$ and $x_i = y \forall x_l$ is directly density-reachable from $x_{l+1}$ for $l = 1, 2, \ldots, i-1$.

\textbf{Definition 6. (Density-connected):} Two points $x$ and $y$ are said to be density-connected with respect to $\varepsilon$ and $N_{\text{min}}$ if there exists a point $z$ such that $x$ and $y$ are density-reachable from $z$ with respect to $\varepsilon$ and $N_{\text{min}}$.

Then a cluster is defined as a set of density-connected points that is maximal with respect to density-reachability.

\textbf{Definition 7. (Cluster):} Given a set $S$ of $N$ data points. A cluster $C$ with respect to $\varepsilon$ and $N_{\text{min}}$ is a non-empty subset of $S$ such that the following conditions are satisfied.

1. $\forall x, y \in S$ if $x \in C$ and $y$ is density-reachable from $x$ with respect to $\varepsilon$ and $N_{\text{min}}$ then $y \in C$
2. $\forall x, y \in C$, $x$ and $y$ are density-connected with respect to $\varepsilon$ and $N_{\text{min}}$

It can easily be seen from the definition 7 that a cluster contains minimum of $N_\varepsilon$ points. In DBSCAN, we begin with a random point $x$ and finds all the points which are density-reachable from $x$ with respect to $\varepsilon$ and $N_{\text{min}}$. If $x$ is a core point, then the formation of a cluster is completed with respect to $\varepsilon$ and $N_{\text{min}}$. It is obvious to
note that no points are density-reachable from \( x \) when \( x \) is a border point in case of which DBSCAN begins with an unclassified point to repeat the same process. The two parameters \( \varepsilon \) and \( N_{\text{min}} \) direct the notion of DBSCAN and decide the quality of clusters. These two parameters are used in a global way in DBSCAN. In order to form high quality clusters, it is very crucial to select the efficient values for these two parameters. DBSCAN visits each point of the database, possibly multiple times. Usually the DBSCAN-based methods have the time complexity \( O(n^2) \).

6.2.2 Prototype

We define a prototype \( p \) of the given set \( S \) as a representative point for a group of points of \( S \). In the proposed method, the prototypes are chosen from the \( K \)-means algorithm which generates a partition of the given data so as to minimize the squared error. Let \( S = \{X_1, X_2, \ldots, X_n\} \) is the given set of \( n \) data points where each point \( X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d}) \in \mathbb{R}^d \) is of dimension \( d \) and \( k \) be the required number of clusters \( C_1, C_2, \ldots, C_k \). Then, the \( K \)-means is applied with \( k' (> k) \) number of desired clusters \( C_1, C_2, \ldots, C_{k'} \). Then we consider the centroids \( c_1, c_2, \ldots, c_{k'} \) of these clusters as the prototypes to represent the points of their corresponding clusters. In the proposed method, we initially form \( k' \) number of prototypes where \( k' \) is assumed to be sufficiently larger than the number of clusters \( k \). The following Figures 6.1(a-d) shows 30, 20, 10 and 5 prototypes formed in the cluster-inside-cluster data. All the prototypes except in Figure 6.1(d) contain the points of same cluster. In Figure 6.1(d) all the five prototypes have the points of multiple clusters. This indicates that the number of prototypes should be chosen sufficiently large in order to avoid the amalgamation of points of multiple clusters into a prototype.
6.3 Algorithm Based on Squared Error Clustering

Given a set $S$ of $n$ data points, we first choose a number $k'$ which is sufficiently large than number of clusters. The squared error clustering method is applied on the given data to form $k'$ number of sub-clusters, say, $\{P_1, P_2, \ldots, P_{k'}\}$. The corresponding centroids $\{p_1, p_2, \ldots, p_{k'}\}$ are chosen as the prototypes. Then, we apply the DBSCAN algorithm on the given data by avoiding the unnecessary distance computations with the help of prototypes produced by squared error clustering. The key feature of this method is as follows. If a point $x$ of a prototype $p$ is within the $\varepsilon$-neighborhood of a point $y$ of another prototype $q$, then all the points represented by both of these prototypes $p$ and $q$ belongs to the same cluster. The main scheme of the proposed
method is as follows. Initially, all the \( k' \) prototypes are marked as unclassified. We start randomly with a sub-cluster \( P_m \) represented by the prototype \( p_m \). Add the sub-cluster \( P_m \) to cluster \( C_j \) (\( j \) equals to 1 initially) and mark its prototype \( p_m \) as classified. Then, the Euclidean distances of all the points \( x_i \in P_m \) from \( p_m \) are calculated. We now start with the point \( x_i \in P_m \) whose distance from \( p_m \) is maximum and compute its \( \varepsilon \)-neighborhood \( N_\varepsilon (S, x_i) \). At this moment, distances from the point \( x_i \) to all the prototypes (except \( p_m \)) are calculated to apply the DBSCAN on the points whose prototype is closer to \( x_i \). This computation is needed to deal only with the points of closer prototypes and to skip the points of far prototypes. Assume, \( p_n \) is the closer prototype of \( x_i \). Then we find whether there is any point \( y \in P_n \) such that \( y \) is directly density-reachable (see definition 4) from \( x_i \) with respect to \( \varepsilon \) and \( N_{min} \). If so, then add the whole sub-cluster \( P_n \) to \( C_j \) and mark \( p_n \) as classified. Now, we consider an unclassified prototype \( p \), which is closer to the point \( x_i \) to repeat the same process. On the other side, when no point of \( P_n \) is directly density-reachable from \( x_i \), then we can ignore \( x_i \) since we have considered the prototypes with respect to \( x_i \) in sorted order. After ignoring \( x_i \), we consider the next farthest point of \( P_m \) from its prototype \( p_m \) and repeat the same process for \( t \) number of farthest points of \( P_m \) from \( p_m \). With this, the initially considered prototype \( P_m \) is exhausted. If any new sub-clusters are amalgamated to the cluster \( C_j \) during this iteration with respect to the sub-cluster \( P_m \), then the same process is repeated for each of this newly merged sub-cluster to proceed further. The stage of complete formation of a cluster can be identified as follows. If at least one of the \( k' \) prototypes is unclassified and no points of any unclassified prototype are directly density-reachable from the points of any classified prototypes. In this case we increment the value of \( j \) by 1 and the points of next coming cluster are stored in \( C_j \). The whole process is terminated when all the \( k' \) prototypes are classified. Unlike other DBSCAN methods, here the value of \( N_{min} \) is fixed at 1. Because, if a point \( y \) of another prototype \( P_n \) is directly density-reachable from \( x_i \), all the other points of \( P_n \) and \( P_m \) belong to the same cluster. The algorithm is formally presented as follows.
Algorithm MDBSCAN \((S, t)\)

**Input:** A set \(S\) of \(n\) points and the number of clusters \(k\);

**Output:** The set \(C_i\) of clusters

**Functions and variables used:**
- \(t\): Maximum number of points of any prototype to apply the DBSCAN.
- \(\varepsilon\): A neighborhood constant.
- \(N_{\text{min}}\): Limit of the minimum number of points for the DBSCAN.
- \(SE(S, k')\): A function to produce the \(k' (> k)\) sub-clusters \(\{P_1, P_2, \ldots, P_{k'}\}\) using squared error clustering.
- \(P\): A set to store the \(k'\) sub-clusters \(\{P_1, P_2, \ldots, P_{k'}\}\).
- \(p\): A set to store the \(k'\) prototypes \(\{p_1, p_2, \ldots, p_{k'}\}\) which are the centroids of \(\{P_1, P_2, \ldots, P_{k'}\}\).
- \(Sort(x_i)\): A function to sort the points \(x_i\).
- \(d(x, y)\): Euclidean distance between the points \(x\) and \(y\).
- \(\varepsilon\)-nbhd\((x)\): A function to find the \(\varepsilon\) - neighborhood of the point \(x\).
- \(C_k\): Set of \(k\) clusters. /* initially empty */
- \(i, j, k, l, q, t, count\): Temporary variables /* \(k, l\) are 1 and \(count\) is 0 initially */
- \(Temp_i\): Temporary set.

**Step 1:** Call squared error clustering function \(SE(S, k')\) to find a set of \(k'\) sub-clusters \(\{P_1, P_2, \ldots, P_{k'}\}\).

**Step 2:** Find the \(k'\) prototypes \(\{p_1, p_2, \ldots, p_{k'}\}\) as the centroids of \(\{P_1, P_2, \ldots, P_{k'}\}\).

**Step 3:** Mark all the prototypes \(p_1, p_2, \ldots, p_{k'}\) as unclassified.

**Step 4:** Randomly start with a sub-cluster \(P_m\) with an unclassified prototype \(p_m\).

**Step 5:** Add \(P_m\) to \(C_q\), i.e. \(C_q \leftarrow C_q \cup P_m\). Then mark \(p_m\) as classified.

**Step 6:** Calculate \(d(x_i, p_m) \forall x_i \in P_m\) and sort these distances in descending order using \(Sort()\);

**Step 7:** Start with a point \(x_i \in P_m\) whose distance from \(p_m\) is maximum.
\textit{count} \leftarrow \textit{count} + 1;

\textbf{Step 8:} Calculate \( d(x_i, p_j) \forall p_j \in p-\{p_m\} \) and sort the prototypes in ascending order using Sort();

\textbf{Step 9:} Find the closer unclassified prototype (say \( p_n \)) to \( x_i \).

\textbf{Step 10:} Find the \( \varepsilon \)-neighborhood of \( x_i \) by calling the function \( \varepsilon\text{-nbhd} (x_i) \).

\textbf{Step 11:} If there exist at least one point \( y \in P_n \) such that \( y \) is directly density-reachable from \( x_i \) with respect to \( \varepsilon \) and \( N_{\text{min}} \), then

\[
\{ \\
\quad \text{Add the sub-cluster } P_n \text{ to } C_q \text{ and } \text{Temp}_i \cdot \text{i.e. } \text{Temp}_i \leftarrow \text{Temp}_i \cup P_n; C_q \leftarrow C_q \cup P_n; \\
\quad \text{Mark } p_n \text{ as classified.} \\
\quad \text{count} \leftarrow \text{count} + 1; \\
\quad \text{Repeat from step 9 with next closer prototype to } x_i. \\
\}
\]

\textbf{Else}

\[
\{ \\
\textbf{If} \text{ count } < t, \text{ then} \\
\quad \text{Repeat from step 7 with new } x_i (\in P_m) \text{ value taken as the } (\text{count}+1)^{\text{th}} \text{ farthest point from } p_m. \\
\textbf{Else} \\
\{ \\
\textbf{If} (\text{Temp}_i \neq \emptyset) \text{ then} \\
\quad \{ \\
\quad \quad l \leftarrow l + 1; \\
\quad \quad \text{Repeat steps 6 to 11 for every sub-cluster of } \text{Temp}_i. \\
\quad \} \\
\textbf{Else} \\
\{ \\
\} 
\} 
\]
If all the $k'$ prototypes are classified then go to step 12;

Else

{ 
    $q \leftarrow q + 1$;
    Go to step 4;
}

Step 12: Output the clusters and Exit ();

**Time Complexity:** The initial phase of squared error clustering algorithm runs in $O(k'nt)$ time, where $n$ is the number of points, $k'$ is the required number of sub-clusters and $t$ is the number of iterations. Since the $k'$ value is fixed here, it requires $O(nt)$ time. Assume $m$ is the maximum number of points for any $k'$ sub-clusters. The distances from the points of all $k'$ sub-clusters are computed with respect to only their prototypes (centroids) which require $O(mk')$. All the points of $k'$ sub-clusters are sorted with respect to their distances from the corresponding prototypes in $O(k'm \log m)$ time. Then we have computed the distances from $t$ number of points of all the sub-clusters to the $k'$ prototypes. This is needed $O(k't^2)$ computation time. The $\varepsilon$-neighborhood is calculated for $t$ number of points of all the $k'$ sub-clusters with respect to only their neighboring sub-clusters (say $q$). The computation time for this task is $O(k'tqm)$ time. Here, all the values of $k'$, $m$, $t$ and $q$ are very small compared to $n$. Therefore, the overall time complexity of the proposed method is maximum of $\{O(nt), O(k'tqm)\)$. 


6.3.1 Experimental Results

We tested the proposed algorithm on various artificial and biological data sets. In order to compare with the proposed scheme, we implemented few existing techniques, namely, I-DBSCAN [166], DBCMAM [121], VDBSCAN [167] and KFWDBSCAN [168]. For the sake of visualization, initially the proposed and the existing methods were experimented on six artificial data sets. Then, the algorithms were tested on various biological data sets using the error rate [93]. Finally, the results of four gene expression data were compared using the runtime of the algorithms.

6.3.2 Results of Artificial Data

We used four artificial data sets namely, two-spiral, cluster-inside-cluster, crescent-full moon and outlier data for the experimentation. Initially, the squared error clustering was applied on all the above data sets with the number of prototypes 25. The prototypes produced for all the above data are shown in the Figures 6.2(a), 6.2(c), 6.2(e) and 6.2(g) respectively. Then, the proposed method was experimented on all the above illustrated data sets with the help of the prototypes produced by squared error clustering method. The algorithm successfully produced the desired clusters as shown in the Figures 6.2(b), 6.2(d), 6.2(f) and 6.2(h) respectively.
Figure 6.2: Results of two-spiral, cluster-inside-cluster, crescent-full moon and outlier data; (a, c, e, g) initial prototypes; (b, d, f, h) result of the proposed method
6.3.3 Results of Real World Data

Here, we compare the results of eight biological data sets [32], namely, iris, wine, statlog heart, breast tissue, Pima Indians diabetes, cloud, blood transfusion and yeast using the error rate (ER) [93] which is very efficient to evaluate the clusters of complex supervised/trained data. It is defined as follows

\[ ER = \frac{Number\ of\ misclassified\ objects}{Total\ number\ of\ objects} \times 100\% \]  

(6.1)

It can be seen from the definition that lesser values of ER indicate more quality of the clusters. The comparison results of the proposed method with I-DBSCAN [166], DBCAMM [121], VDBSCAN [167] and KFWDBSCAN [168] using the above error rate (ER) are depicted in Table 6.1. It is easy to observe from the values of the ER that the proposed algorithm produces efficient results than that of the existing.
Table 6.1: Results of biological data using error rate (ER)

<table>
<thead>
<tr>
<th>Data</th>
<th>Data Size</th>
<th>Misclassified Patterns</th>
<th>ER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>[166]</td>
<td>[121]</td>
</tr>
<tr>
<td>Iris</td>
<td>150</td>
<td>22</td>
<td>14</td>
</tr>
<tr>
<td>Wine</td>
<td>178</td>
<td>23</td>
<td>12</td>
</tr>
<tr>
<td>Statlog Heart</td>
<td>270</td>
<td>44</td>
<td>27</td>
</tr>
<tr>
<td>Breast Tissue</td>
<td>106</td>
<td>23</td>
<td>5</td>
</tr>
<tr>
<td>Pima Indians Diabetes</td>
<td>768</td>
<td>193</td>
<td>41</td>
</tr>
<tr>
<td>Cloud</td>
<td>1024</td>
<td>98</td>
<td>35</td>
</tr>
<tr>
<td>Blood Transfusion</td>
<td>748</td>
<td>56</td>
<td>21</td>
</tr>
<tr>
<td>Yeast</td>
<td>1484</td>
<td>87</td>
<td>19</td>
</tr>
</tbody>
</table>
6.3.4 Results of Gene Expression Data

Finally, we considered four gene expression datasets for the experimentation, namely Yeast cell-cycle, Sporulation, Lymphoma and Diauxic [33]. The proposed and the existing methods namely, I-DBSCAN, DBCAMM, VDBSCAN and KFDBSCAN were applied on these four multi-dimensional gene expression data sets. The results were compared using the run time of these methods. It is obvious to note from the comparison results shown in Figures 6.3(a-d) that the proposed method has less computational complexity in case of most of the gene expression data sets.

![Figure 6.3: Run time comparison of MDBSCAN with I-DBSCAN, DBCAMM, VDBSCAN and KFDBSCAN on genome data, (a) Yeast cell-cycle; (b) Sporulation; (c) Lymphoma; (d) Diauxic](image-url)
6.4 Conclusion

A novel approach has been proposed to speed up the DBSCAN algorithm using the sub-cluster prototypes produced by the squared error clustering method. The proposed method is experimented on different artificial and biological data. The results are compared with various existing techniques in terms of error rate and run time. All the comparison results depicted the effectiveness of the proposed modified DBSCAN method over I-DBSCAN, DBCAMM, VDBSCAN and KFWDBSCAN.