l-DBSCAN : A Fast Hybrid Density Based Clustering Method

P. Viswanath, Rajwala Pinkesh
Department of Computer Science and Engineering,
Indian Institute of Technology - Guwahati,
Guwahati - 781039, India.
{viswanath, rajwala}@iitg.ernet.in

Abstract

Density based clustering techniques like DBSCAN can find arbitrary shaped clusters along with noisy outliers. A severe drawback of the method is its huge time requirement which makes it a unsuitable one for large data sets. One solution is to apply DBSCAN using only a few selected prototypes. But because of this the clustering result can deviate from that which uses the full data set. A novel method proposed in the paper is to use two types of prototypes, one at a coarser level meant to reduce the time requirement, and the other at a finer level meant to reduce the deviation of the result. Prototypes are derived using leaders clustering method. The proposed hybrid clustering method called l-DBSCAN is analyzed and experimentally compared with DBSCAN which shows that it could be a suitable one for large data sets.

1. Introduction

Density based clustering techniques like DBSCAN (Density Based Clustering of Applications with Noise) [2] can discover consistent arbitrary shaped clusters along with detection of noisy outliers. This is in contrast with k-means type clustering methods which finds compact and spherical shaped clusters. Single-linkage method [3] can find arbitrary shaped clusters, but apart from its huge time requirements, it is a very sensitive method to noise. In this perspective, density based methods are attractive. But this also suffers from huge computational requirements. For DBSCAN, the time complexity is $O(n^2)$ where n is size of the data set. One way to overcome this problem is to build an index over the data set like a R^*-tree index. But this solution is suitable only when the dimensionality of the data is low.

Hybrid clustering methods are recently used [1] to overcome the problems with large data sets. The basic technique is to first find suitable prototypes from the large data set and then to apply the clustering method using only the prototypes. These kind of schemes can be seen as approximate methods where the solution can deviate from that of the original method (which uses the entire data set) based on the representative power (quality) of the prototypes. Number of prototypes is also important. Small number of prototypes are in favor of reducing the computational requirements whereas large number of prototypes are required to see that the solution is closer to that of the original method. It seems that a novel way is to use atleast two types of prototypes, one is at a coarser level meant to reduce the computational requirements, and the other at a finer level to ensure smaller approximation errors.

The paper tries to improve on DBSCAN by following a hybrid clustering scheme. It first derives a two level hierarchy of prototypes by employing *leaders clustering method* [4] with two different threshold values. Leaders clustering method is a fast method which runs in linear time of the input data set size. These prototypes are subsequently carefully used in deriving density based clusters. We analyze pros and cons of the proposed hybrid scheme along with experimental evidences. The proposed method is called *l*-DBSCAN, where the *l* stands for *leaders*.

The paper is organized as follows. Section 2 reviews DBSCAN method, Section 3 reviews leaders clustering method along with a modified leaders method called *coarse-fine-leaders* to derive two types of leaders. *l*-DBSCAN is described in Section 4. Experimental results are given in Section 5 and Section 6 gives some of the conclusions.

2. DBSCAN

DBSCAN groups the data points which are dense and connected into a single cluster. Density at a point is given by m/nV, where m is the number of points out of n input data points that are falling in a small region around the point and V is the volume of the region. The region is assumed to be a hyper sphere of radius ϵ at the point and hence the threshold density can be specified by a parameter *MinPts*,

Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06)
0-7695-2521-0/06 $20.00 © 2006 IEEE

Algorithm 1 DBSCAN($\mathcal{D}$, ϵ, *MinPts*)

```
{Each cluster is given an identifier cid }
cid = 0;
for each pattern x in D do
  if x is not marked as "seen" then
    Mark x as "seen";
    Find N_ε(x; D);
    if card(N_ε(x; D)) < MinPts then
      Mark each pattern of N_ε(x; D) as "noise";
    else
      cid = cid +1;
      Mark each pattern of N_ε(x; D) with cid;
      for each pattern y ∈ N_ε(x; D) and y is not
      marked as "seen" do
        Mark y as "seen";
        Find N_ε(y; D);
        if N_ε(y; D) ≥ MinPts then
          Mark each pattern of N_ε(y; D) with cid;
          If any pattern of N_ε(y; D) is marked as
          "noise" then remove that mark;
        end if
      end for
    end if
  end if
end for
Output all patterns of D along with cid or "noise" mark;
```

minimum number of points required to be present in the region to make it dense. For given ϵ and *MinPts* values, DBSCAN finds a dense point in the input set and expands it by merging neighboring dense regions together. A point can be either a dense (core point) or a non-dense point (non-core point). A non-dense point might be a border point of a dense region or a noisy pattern.

Algorithm 1 describes DBSCAN where $\mathcal{D}$ is the input data set, $N_\epsilon(x;\mathcal{D})$ is the sub-set of patterns in $\mathcal{D}$ that are present in hyper-sphere of radius ϵ at x ($x \in \mathcal{D}$). $card(N_\epsilon(x;\mathcal{D}))$ is the cardinality of the set. The algorithm marks each pattern of $\mathcal{D}$ with a cluster identifier (*cid*) which gives the cluster to which the pattern belongs or gives a mark *"noise"* indicating that the pattern is a noisy one. One additional mark *"seen"* is used to distinguish between the patterns which are processed from that which are not. Note that a pattern which is initially marked as *"noise"* can later on become a border point of a cluster and hence the *"noise"* mark can be deleted.

It can be seen that the time consuming step in DBSCAN is in finding $N_\epsilon(x;\mathcal{D})$ which can take $O(n)$ time where $x \in \mathcal{D}$ and $\mid \mathcal{D} \mid = n$. Hence the time complexity of the method is $O(n^2)$. On the otherhand, if we derive k leaders first by applying leaders clustering method and subsequently applying DBSCAN only with leaders has total time complexity of $O(n + k^2)$. Normally for large data sets with appropriate setting of parameters for the leaders method, k can be much smaller than n. Hence the hybrid scheme can be a faster one.

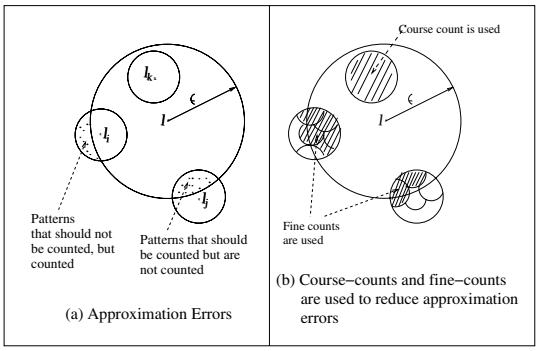

Figure 1. Approximation errors.

3. Leaders

For a given threshold distance τ, leaders method [4] works as follows. It maintains a set of leaders $\mathcal{L}$, which is initially empty and is incrementally built. For each pattern x in $\mathcal{D}$ if there is a leader $l \in \mathcal{L}$ such that distance between x and l is less than τ, then x is assigned to the cluster represented by l. Note that even if there are many such leaders, only one (the first encountered one) is chosen (because of this the cluster represented by a leader is of semi-spherical shape). If there is no such a leader then x itself becomes a new leader and is added to $\mathcal{L}$. Along with each leader a count indicating number of patterns that are grouped under the leader is also stored. So the output of leaders method is a set of pairs $\mathcal{L}^\star = \{(l, count(l)) \mid l \in \mathcal{L}\}$.

The hybrid clustering scheme also uses same values of ϵ and *MinPts* but uses $\mathcal{L}^\star$ instead of $\mathcal{D}$. So the hybrid method primarily outputs clusters of leaders (*i.e.,* a partition of $\mathcal{L}$) which can further be mapped into clusters of input data patterns by expanding (replacing) each leader by the patterns in $\mathcal{D}$ for which it is the representative. The key difference between the hybrid scheme and DBSCAN is in finding $card(N_\epsilon(l;\mathcal{L}^\star))$. If $N_\epsilon(l;\mathcal{L}^\star) = \{l_1, \ldots, l_m\}$, then $card(N_\epsilon(l;\mathcal{L}^\star))$ is taken to be $\sum_{i=1}^{m} count(l_i)$, the sum of counts of leaders which are present in $N_\epsilon(l;\mathcal{L}^\star)$.

A major source of error because of which the hybrid scheme's output can differ from that of the DBSCAN is when a leader l which is dense according to $\mathcal{L}^\star$ can be non-dense according to $\mathcal{D}$. That is, it is possible that $card(N_\epsilon(l;\mathcal{L}^\star)) > MinPts$ whereas $card(N_\epsilon(l;\mathcal{D})) \leq MinPts$. Or a non-dense leader according to $\mathcal{L}^\star$ can be dense according to $\mathcal{D}$.

Let *leader-cluster*(l) is the set of patterns in $\mathcal{D}$ which are grouped under leader l according to the leaders algorithm. There can be a leader $l_i \in N_\epsilon(l; \mathcal{L}^\star)$, but some patterns in *leader-cluster*(l_i) can be present outside of $N_\epsilon(l; \mathcal{D})$. See Figure 1(a). Because of these kind of leaders a non-dense leader can become dense. Opposite of this problem is when a leader l_j which is not in $N_\epsilon(l; \mathcal{L}^\star)$ can have some patterns of *leader-cluster*(l_j) present in $N_\epsilon(l; \mathcal{D})$. Because of this a dense leader can become non-dense. These approximation errors can be reduced by reducing the τ (threshold distance) value and becomes zero as $\tau \to 0$. But as τ value is reduced the number of leaders *i.e.,* $\mid \mathcal{L} \mid$ increases and hence the computational advantage because of the hybrid scheme reduces. Note that there is no approximation error added due to leaders like l_k in Figure 1(a).

Approximation errors as discussed above is due to leaders that are near to the boundary of the hyper-sphere (with radius ϵ at l). So, one can have leaders derived from two threshold values τ_f and τ_c such that $\tau_f < \tau_c$, and by choosing carefully between these two types of leaders one can reduce the computational requirements and also the approximation errors. The basic idea is to use leaders with threshold τ_c called *coarse-leaders* in the middle portions of the hyper-sphere and leaders with threshold τ_f called *fine-leaders* in the boundary regions.

We first present the hierarchical leaders clustering method called *coarse-fine-leaders* followed by the hybrid density based clustering method *l*-DBSCAN.

The method *coarse-fine-leaders*($\mathcal{D}$, τ_f, τ_c) (see Algorithm 2) first partitions $\mathcal{D}$ according to leaders method using threshold τ_c. This gives leaders called *coarse-leaders* along with respective counts. The set of patterns represented by each *coarse-leader* is again partitioned using leaders algorithm, but with threshold τ_f. In the method these two types of leaders are derived simultaneously. The output of the method, *i.e.,* $\mathcal{L}^{\star\star}$ is a set of triplets where the first member is a *coarse-leader*, the second member is its count called *coarse-count* and the third member is a set consisting of *fine-leaders* grouped under the *coarse-leader* along with their count values called *fine-counts*. For a *coarse-leader* l, *fine-leaders*(l) represents the set of *fine-leaders* grouped under l. The time complexity of the method is $O(n)$ only as that of leaders method.

4. *l*-DBSCAN

The proposed hybrid clustering method *l*-DBSCAN works similar to DBSCAN but uses only the set of all fine-leaders $\mathcal{L}^\star = \bigcup_{\forall l} \mathcal{L}_l^\star$ in deriving the clusters. To reduce the cost of finding *card*$(N_\epsilon(l; \mathcal{L}^\star))$ a modified method called *l-card*$(N_\epsilon(l; \mathcal{L}^\star))$ is given in Algorithm 3. Since a coarse-leader (like l_j in Figure 1(b)) can lie outside of the hyper-sphere of radius ϵ at l for which a fine-leader can lie within the hyper-sphere, the algorithm first finds all coarse-leaders that lie within the radius of $\epsilon + \tau_c$ at l (because, a *coarse-leader* which lies outside of this hyper-sphere can not have any patterns that could lie in the hyper-sphere of radius ϵ at l). For coarse-leaders like l_k in Figure 1(a) the coarse-count is used. For others the fine-count of fine-leaders which lie within the hyper-sphere is used.

Algorithm 2 Coarse-fine-leaders($\mathcal{D}$, τ_f, τ_c)

$\mathcal{L} = \emptyset$;
for each $x \in \mathcal{D}$ **do**
  Find a $l \in \mathcal{L}$ such that *Distance*$(x, l) < \tau_c$
  if there is no such l or when $\mathcal{L} = \emptyset$ **then**
    $\mathcal{L} = \mathcal{L} \cup \{x\}$;
    coarse-count(x) = 1;
    fine-leaders(x) = $\{x\}$;
    fine-count(x) = 1;
  else
    coarse-count(l) = *coarse-count*(l) + 1;
    Find a $l_f \in$ *fine-leaders*(l) such that *Distance*$(x, l_f) < \tau_f$;
    if there is no such $l_f \in$ *fine-leaders*(l) **then**
      fine-leaders(l) = *fine-leaders*(l) $\cup \{x\}$;
      fine-count(x) = 1;
    else
      fine-count(l_f) = *fine-count*(l_f) + 1;
    end if
  end if
end for
Output $\mathcal{L}^{\star\star} = \{(l, \textit{coarse-count}(l), \mathcal{L}_l^\star) \mid l \in \mathcal{L}\}$, where $\mathcal{L}_l^\star = \{(l_f, \textit{fine-count}(l_f)) \mid l_f \in \textit{fine-leaders}(l)\}$;

5. Experimental Results

Experiments are done both with synthetic and real-world data sets. Synthetic data set used is of two dimensional one as shown in Figure 2 consisting of 4000 patterns. For this, DBSCAN with $\epsilon = 16$ and *MinPts* = 10 finds the two banana shaped clusters and also finds noisy outliers. We used one real world data set called *pendigits* data from UCI machine learning repository (URL: http://www.ics.uci.edu/ mlearn/MLRepository.html). For this only the training data is used without class-labels. The data is of 16 dimensions and has 7494 patterns. ϵ and *MinPts* are so chosen that the clustering output of DBSCAN is as close to that of the actual classification (according to the class-labels given in the training set) as possible. Distance measure used is Euclidean distance.

l-DBSCAN is compared with the standard DBSCAN with respect to clustering output and time requirements. *l*-DBSCAN and DBSCAN are run with same ϵ and *MinPts*

Algorithm 3 *l-card*$(N_\epsilon(l; \mathcal{L}^\star))$

$\{\mathcal{L}^{\star\star}$, τ_f and τ_c are given as global parameters for this algorithm$\}$
Find the set of all coarse-leaders present within the radius of $\epsilon + \tau_c$ at l. That is, Find $N_{\epsilon+\tau_c}(l; \mathcal{L}^{\star\star})$;
total-count = 0;
for each coarse-leader $l_c \in N_{\epsilon+\tau_c}(l; \mathcal{L}^{\star\star})$ **do**
  if *Distance*(l, l_c) + $\tau_c < \epsilon$ **then**
    total-count = *total-count* + *coarse-count*(l_c);
  else
    for each fine-leader $l_f \in$ *fine-leaders*(l_c) **do**
      if *Distance*$(l, l_f) < \epsilon$ **then**
        total-count = *total-count* + *fine-count*(l_f);
      end if
    end for
  end if
end for
return(*total-count*);

but the additional parameters τ_f and τ_c, which are used only with *l*-DBSCAN are varied. τ_c values used are ϵ and $\epsilon/2$, whereas τ_f values used are $\tau_c/2$, $\tau_c/4$ and 0 (*zero*). Note that with $\tau_f = 0$, *l*-DBSCAN and DBSCAN gives exactly same clustering results, since there are no approximation errors. But the actual time taken by *l*-DBSCAN can be much smaller that by DBSCAN because $\tau_c > 0$. Rand-Index is used to compare the clustering outputs of the two methods. All noisy patterns are grouped under a separate cluster. Table 1 summarizes the results where the last column gives the computational advantage with respect to time, which gives ratio of actual time taken by *l*-DBSCAN and DBSCAN.

Some observations from the results are as follows. With synthetic data set, with $\tau_c = \epsilon$ and $\tau_f = \epsilon/2$, *l*-DBSCAN deviates from DBSCAN considerably (Rand-Index = 0.981). This is primarily because noisy patterns according to DBSCAN as shown in Figure 2 are added to the near-by cluster by *l*-DBSCAN. This is because, with $\tau_f = \epsilon/2$, *l*-DBSCAN has greater approximation errors, hence is less sensitive to noise. For pendigits data also a similar deviation of the clustering results can be observed. With other values of τ_f the results are very good. It can be seen that, for both data sets, with $\tau_c = \epsilon/2$ and $\tau_f = \epsilon/8$, *l*-DBSCAN gives same clustering result as that of DBSCAN but consumes considerably very less time (with synthetic data, DBSCAN takes 1563.7 seconds whereas *l*-DBSCAN takes only 65.7 seconds).

6. Conclusions

The paper presented a novel scalable hybrid clustering scheme to get density based arbitrary shaped clusters. First two types of prototypes are derived using leaders clustering method. These prototypes are carefully used by the density based clustering method DBSCAN. The proposed method is called *l*-DBSCAN which is considerably faster than DBSCAN using the entire data set. With suitable selection of parameters, *l*-DBSCAN can find same clustering result as that found by DBSCAN but takes considerably less time than that taken by DBSCAN.

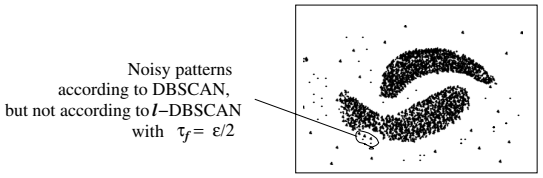

Figure 2. Synthetic Data Set.

Synthetic Data Set. $\epsilon = 16$, *MinPts* = 10			
τ_c	τ_f	Rand-Index	Time Ratio
ϵ	$\epsilon/2$	0.981	0.003
ϵ	$\epsilon/4$	0.993	0.015
ϵ	0	1	0.121
$\epsilon/2$	$\epsilon/4$	0.993	0.007
$\epsilon/2$	$\epsilon/8$	1	0.042
$\epsilon/2$	0	1	0.355
Pendigits Data Set. $\epsilon = 300$, *MinPts* = 30			
τ_c	τ_f	Rand-Index	Time Ratio
ϵ	$\epsilon/2$	0.972	0.012
ϵ	$\epsilon/4$	0.991	0.030
ϵ	0	1	0.411
$\epsilon/2$	$\epsilon/4$	0.991	0.054
$\epsilon/2$	$\epsilon/8$	1	0.083
$\epsilon/2$	0	1	0.538

Table 1. *l*-DBSCAN Vs. DBSCAN.

References

[1] E. Y. Cheu, C. K. Kwoh, and Z. Zhou. On the two-level hybrid clustering algorithm. In *Proceedings of International Conference on Artificial Intelligence in Science and Technology*, pages 138–142, 2004.

[2] M. Ester, H. P. Kriegel, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd ACM SIGKDD*, pages 226–231, Portand, Oregon, 1996.

[3] A. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[4] H. Spath. *Cluster Analysis Algorithms for Data Reduction and Classification*. Ellis Horwood, Chichester, UK, 1980.