Spatial data mining

DBRS: A Density-Based Spatial Clustering Method with Random Sampling

Bence Novak
2007

Basic definitions

- Clustering
  - Divide the data points into clusters (groups) so that
    - points in the same cluster as similar as possible
    - points in different clusters as different as possible
  - Other words:
    - minimizing the within cluster distance
    - maximizing the between cluster distance
  - In spatial data:
    - find regions with high point intensity
    - separated by areas with low intensity

Introduction

- In this presentation, we are interested in clustering in spatial datasets with the following four features:
  - the clusters may be of widely varying shapes
    - e.g. addresses for historical and current students at a university, apartment buildings (roughly rectangular), along major bus lines (long thin lines).
  - clusters may be of varying densities
    - e.g. supermarket customers across the whole country, for metropolitan areas, clusters may be very dense, on the countryside the clusters are more sparsely distributed.

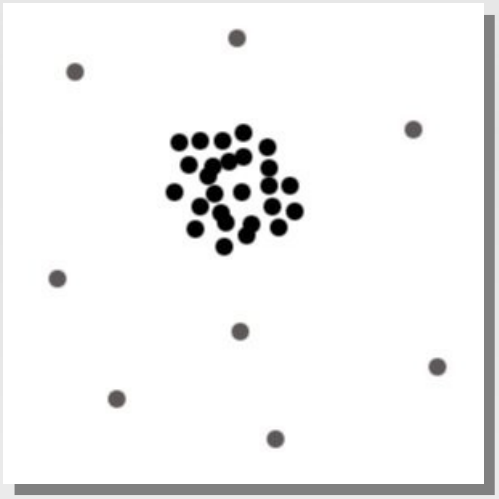
Introduction cont.

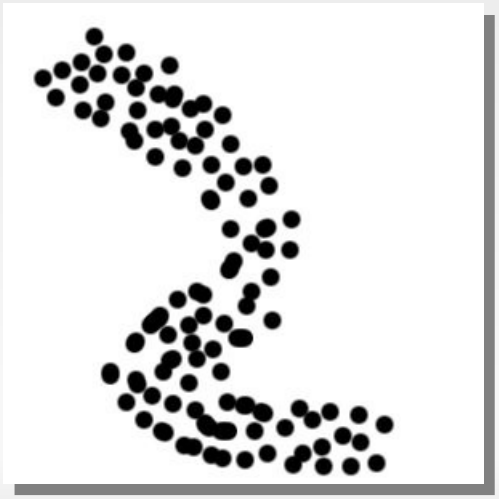
- the spatial dataset may have significant non-spatial attributes
  - e.g. in image processing the procedure for region-based segmentation compares a pixel with its neighbours, region growing is heavily based not only the location of the pixel but also the non-spatial attributes.
- we are interested in very large datasets, including those of at least 100 000 points.

α-approximate density based clustering

- The goal is to find clusters until the probability that any cluster of at least *MinPts* points has not been found is at most α.
- In practice, not every application requires complete and accurate results. E.g. if a construction company wants to distribute its advertisement to clusters of similar residences according to the residence type (non-spatial attribute) and location (spatial), then an α-approximate clustering may be satisfactory.

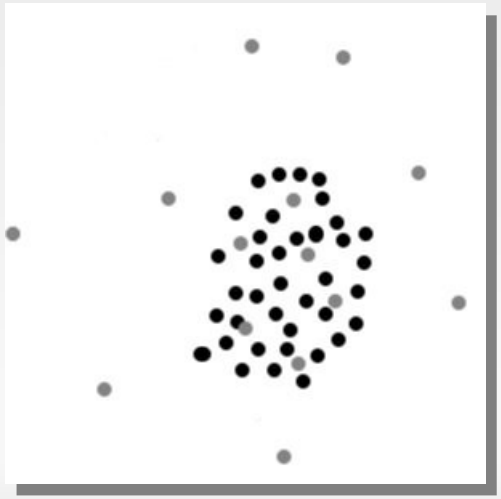
DBRS





- Density-Based clustering with Random Sampling (DBRS) can discover density-based clusters with noise (first picture).
- It can follows clusters of many shapes (second picture).

DBRS



- DBRS scales well on clusters with very dense neighborhoods (first picture).
- It also handles non-spatial attributes along with spatial attributes by paying attention to the purity of a neighborhood. It can avoid creating clusters of points with different values for non-spatial attributes even though they are close to each other.

DBRS

- DBRS repeatedly picks an unclassified point at random and examines it neighborhood. If the neigh. is sparsely populated or the purity of the points in the neigh. is too low, the point is classified as noise.
- Otherwise, if any point is a part of a known cluster this is joined to that cluster.
- If neither of these, a new cluster is begun with this neighborhood.
- DBRS discovers multiple clusters concurrently.
- We can use one heuristic to save large amount of time (large databases), and with it DBRS can perform α-approximate clustering on very large datasets.

CLARANS

- CLARANS is an other spatial clustering algorithm.
- To find clustering, it finds a medoid (center of the cluster) for each of *k* clusters.
- The process can be described as searching a graph where every node is a potential solution and each node is represented by a set of *k* medoids. Two nodes are neighbors if their sets differ by only one object.
- Steps of the process:
  - select an arbitrary possible clustering node *current*
  - randomly pick a neighbor of *current* and comparing the quality of clusterings at *current* and the neighbor node

CLARANS cont.

- swap, if there is an improvement in the clustering quality (the number of tired neighbors is restricted by *maxneighbor*)
- if swap happens, CLARANS moves to the neighbor's node and the process is started again, otherwise the current clustering produces a local optimum
- if the local optimum found, it starts with new randomly selected node in search for a new local optimum
- the number of local optima to be searched is bounded by *numlocal*

- Problems:
  - it assumes that the object to be clustered are all stored in main memory
  - the run time: without considering *numlocal* the computational complexity is $\Omega(kn^2)$ where *n* is the size of the dataset (large databases)

DBSCAN

- DBSCAN was the first density-based spatial clustering method
- key idea is that to define a new cluster or extend an existing one, a neighborhood around a point of a given radius *(Eps)* must contain at least a minimum number os points *(MinPts)*
- the density in the neighborhood is determined by the choice of a distance function for *p* and *q* denoted by *dist(p,q)*
- it uses an efficient spatial access data structure, called R*-tree
- the average case time complexity is O(*n* log *n*)

Definitions for DBSCAN

- Given a dataset D, a distance function *dist*, and parameters *Eps* and *MinPts* the following definitions are used to define DBSCAN:
  - the ***Eps-neighborhood*** of a point p, denoted by $N_{Eps}(p)$, is defined by $N_{Eps}(p) = \{ q \in D \mid dist(p,q) \leq Eps\}$
  - a point p is ***directly density-reachable*** from a point q with respect to *Eps and MinPts* if $p \in N_{Eps}(q)$ *and* $|N_{Eps}(q)| \geq MinPts$
  - a point p is ***density-reachable*** from a point q with respect to *Eps* and *MinPts* if there is a chain of points $p_1,...,p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i
  - a point p is ***density-connected*** to a point q with respect to *Eps* and *MinPts* if there is a point o such that both p and q are density-reachable from o with respect to *Eps* and *MinPts*

Definitions for DBSCAN cont.

- let *D* be a set of data points, a ***density-based cluster*** *C* with respect to *Eps* and *MinPts* is a non-empty subset of *D* satisfying the following conditions:
  - $\forall p,q$: if $p \in C$ and q is density-reachable from p w.r.t *Eps* and *MinPts*, then $q \in C$
  - $\forall p,q \in C$: p is density-connected to q with respect to *Eps* and *MinPts*

DBSCAN algorithm

- If *Eps* and *MinPts* are defined, it starts to group points from an arbitrary point *q*.
- First find its neighborhood (all points that are directly density reachable from *q*), performing a ***region query***, a look up in the R*-tree.
- If the neighborhood is sparse, it contains fewer than *MinPts* points, then point *q* is labeled as noise.
- Otherwise, a cluster is created and all point in *q*'s neighborhood are placed in this cluster. Then the neighborhoods of **all** *q*'s neighbors are examined to see if they can be added to the cluster.

DBSCAN algorithm cont.

- If a cluster cannot be extended, DBSCAN chooses another arbitrary ungrouped point and repeats the process.
- This iterated until all points have been placed in cluster or labeled as noise. For a dataset with *n* points, *n* region queries are required.
- Advantage:
  - it can follow the shape of the clusters and it only requires the distance function and the two input parameters
  - gives extremely good results and it is efficient in many datasest

DBSCAN algorithm cont.

- Problems:
  - if a dataset has clusters of widely varying densities, DBSCAN is not able to handle it efficiently
  - if non-spatial attributes play a role in determining the desired result, DBSCAN is not appropriate, because it does not take into account any non-spatial attributes
  - DBSCAN is not suitable for finding approximate clusters in very large datasets

Definitions for DBRS

- Given a dataset *D,* a symmetric distance function *dist*, parameters *Eps* and *MinPts*, and a property *prop* defined w.r.t. one or more non-spatial attributes, the following definitions used to define DBRS:
  - The ***matching neighborhood*** of a point *p*, denoted by $N'_{Eps}(p)$, is defined as $N'_{Eps}(p) = \{ q \in D \mid dist(p,q) \leq Eps$ and $p.prop = q.prop \}$
  - DBRS uses a parameter called *MinPur*, to control the purity of the neighborhood (non-spatial attributes).
  - ***core point:*** a point whose matching neighborhood is dense enough (has at least *MinPts* and over *MinPur* percentage of matching neighbors)
  - ***border point:*** a neighbor of a core point, that is not a core point itself
  - ***noise:*** other than core points and border points

Definitions for DBRS

- Two points *p* and *q* are ***directly purity-density-reachable*** w.r.t. *Eps, MinPts* and *MinPur* from each other if
  - $p \in N'_{Eps}(q)$ and $q \in N'_{Eps}(p)$;
  - $|N'_{Eps}(q)| \geq MinPts$ or $|N'_{Eps}(p)| \geq MinPts$;
  - $|N'_{Eps}(q)| / |N_{Eps}(q)| \geq MinPur$ or $|N'_{Eps}(p)| / |N_{Eps}(p)| \geq MinPur$.
  - This is a symmetric relations for two core points as well as one core and one border point, but it is not symmetric for two border points.
- A point *p* and a point *q* are ***purity-density-reachable(PD-reachable)*** w.r.t. *Eps, MinPts,* and *MinPur* from each other, denoted by *PD(p,q),* if there is a chain of points $p_1,...,p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly purity-density-reachable from p_i.
- Let *D* be a dataset of points. A ***purity-density-based cluster*** *C* is a non-empty subset of *D* satisfying the following condition:
  - $\forall p,q \in D$: if $p \in C$ and *PD(p,q)* holds, then $q \in C$
  - It is obvious that for $\forall p,q \in C$, *PD(p,q)* holds.

DBRS

- Look at a cluster as a minimum number of core points (skeletal points) and their neighborhoods. To find a cluster, it is sufficient to perform region queries on the skeletal points.
- We cannot identify skeletal points before examining the dataset. Instead, we can randomly select sample points, find their neighborhoods, and merge them if they intersect.
- In many cases DBRS will find the same clusters as DBSCAN. When two groups have a common border point, DBRS will identify the two groups as one cluster. DBSCAN will separate two groups into two clusters, and the common point will be assigned to the firstly discovered cluster.

DBRS algorithm

- We need the following: *D* is the dataset, *Eps* and *MinPts* are global density parameters, *MinPur* is the minimum fraction of points in neighborhood with property *prop* required to define a cluster.
- It starts with an arbitrary point *q* and finds its matching neighborhood. In the algorithm, the region query `D.matchingNeighbors(q,Eps,prop)` finds this matching neighborhood, which is called `qseeds` (line 4).
- If the matching neighbors of *q* satisfies the *MinPts* and *MinPur* parameters, the *q* is a core point, otherwise *q* is noise or border, but it is tentatively classified as noise (line 6).

DBRS algorithm

- **Algorithm DBRS(D, Eps, MinPts, MinPur, prop)**

```
(1) ClusterList = Empty;
(2) while (!D.isClassified())
(3) { Select one unclassified point q from D;
(4)   qseeds = D.matchingNeighbors(q, Eps, prop);
(5)   if ((|qseeds| < MinPts) or (qseed.pur < MinPur))
(6)        q.clusterID = -1; /*q is noise or a border point */
(7)   else
(8)      { isFirstMerge = True;
(9)        C_i = ClusterList.firstCluster;
(10)       while (C_i != Empty) /* compare qseeds to all
  existing clusters */
(11)       { if ( hasIntersection(qseeds, C_i))
(12)            if (isFirstMerge)
```

DBRS algorithm cont.

```
(13)            { newC_i = C_i.merge(qseeds);
(14)              isFirstMerge = false; }
(15)            else
(16)            { newC_i = newC_i.merge(Ci);
(17)              ClusterList.deleteCluster(C_i); }
(18)          C_i = ClusterList.nextCluster;
(19)        } // while != Empty
(20)     if (isFirstMerge) /* no intersection with any existing cluster */
(21)     { Create a new cluster C_j from qseeds;
(22)       ClusterList = ClusterList.addCluster(C_j); }
(23)   }}
```

DBRS algorithm cont.

- The clusters are organised in a list called `ClusterList`. If `qseeds` intersects with a single existing cluster, DBRS merges `qseeds` into this cluster. If `qseeds` intersects with two or more existing clusters, the algorithm merges `qseeds` and those clusters together (line 11-18).
- Otherwise, a new cluster is formed from `qseeds` (line 20-22).
- After examining the neighborhood of one point, the algorithm selects another arbitrary, unclassified point and repeats the procedure. The procedure iterated until every data point is clustered or is labeled as noise.

DBRS algorithm cont.

- Crucial difference from DBSCAN:
  - once DBRS has labeled a *q*'s neighbors as a part of a cluster, it does not examine the neighborhood for each of these neighbors. It can lead a significant time saving for dense clusters
  - the region query, the most time-consuming part of the algorithm can be answered in O(log *n*) time using R*-tree, so in the worst case, where every point in the dataset is noise, DBRS performs *n* region queries and the time complexity is O*(n* log *n)*, and if any clusters are found, it will perform fewer region queries, so the computational time in the worst case equals the DBSCAN computational time in the average case
  - with a heuristic stopping condition, DBRS can be used for α -approximate density clustering to handle very large datasets more efficiently

Theoretical Comparison

- In the following slides, we compare DBRS and DBSCAN from theoretical view w.r.t. the neighborhood graphs they generate during the clustering.
  - The ***neighborhood graph*** for a spatial relation *neighbor* is a graph *G=(V,E)* with the set of vertices *V* and the set of directed edges *E* such that each vertex corresponds to an object of the database and two vertices *v1* and *v2* are connected iff *neighbor(v1,v2)* holds. Given different *neighbor relations,* a neighborhood graph can be directed or undirected.
  - A neighborhood graph (or subgraph) is ***connected*** iff for any pair of vertices in the graph (or subgraph) there is an undirected path joining the vertices.
  - A directed neighborhood graph (or s.g.) is ***strongly connected*** iff for any two nodes *p* and *q* with *neighbor(p,q)* holding, there is a directed path from *p* to *q.*

Theoretical Comparison cont.

- There are some lemmas helping the comparison, all of them have a proof, but they are quite long and not so difficult, so I will only mention the basic idea of the proofs.
- **Lemma 1.:**
  - If the density-reachable relation is the *neighbor* relation, a connected neighborhood (sub-)graph represents a cluster generated by DBSCAN.
- **Proof:**
  - We must prove that:
    - a cluster defined by DBSCAN (slide 13) is represented in a connected neighborhood (sub-)graph
    - only the points belonging to the same cluster represented in the same connected neighborhood graph

Theoretical Comparison cont.

- From Lemma 1, given n points, the clustering process can be viewed abstractly as constructing neighborhood graphs.
- Each time a core point is found, the algorithm finds the directly density-reachable relation between the core point and its neighbors.
- The directly density-reachable relation holding for the two objects corresponds to the directed edge between the two corresponding vertices in the neighborhood graph.
- Each cluster is constructed as a subgraph. If there are k clusters, the graph has k connected subgraph.

Theoretical Comparison cont.

- **Lemma 2.:**
  - If the density-reachable relation is the *neighbor* relation, DBSCAN's clustering process corresponds to constructing the strongly connected neighborhood graph.
- **Proof:**
  - From Lemma 1, we know each cluster generated by DBSCAN is represented as a connected neighborhood subgraph. To prove the connected neighborhood graph is strongly connected, we need to prove that there is a directed path connecting two vertices *p* and *q* iff *neighbor(p,q)* holds. The proof requires two parts:
    - if *neighbor(p,q)* holds, there is a directed part connected v_p and v_q
    - only if *neighbor(p,q)* holds, there is a directed path from v_p to v_q

Theoretical Comparison cont.

- In the following lemma, we prove that a neighborhood graph is also connected by applying PD-reachable (Slide 18) as the *neighbor* relation.
- **Lemma 3.:**
  - If the PD-reachable relation is the *neighbor* relation, the neighborhood graph generated by DBRS is connected.
- **Proof:**
  - We must prove two parts:
    - a cluster defined by DBRS is represented in a connected neighborhood graph
    - only the points belonging to the same cluster are represented in the same connected neighborhood graph

Theoretical Comparison cont.

- Unless all points are noise, constructing a strongly recommended neighborhood graph is more expensive than constructing an undirected connected graph, fewer points are checked for their neighborhood.
- In DBRS, for any two PD-reachable points, there is at least one undirected path connecting them. In DBSCAN, for any two directly-density reachable core points, there are always two directed path (edges) connecting them.
- Regardless of whether the connectivity is directed or undirected, all connected points should belong to the same cluster.
- In the worst case (all points are noise), the costs of constructing the two neighborhood graphs are the sam because no directed or undirected edges are generated.

Conclusion

- DBRS aims to reduce the number of region queries for datasets with varying densities.
- It scales very well on the high-density clusters.
- DBRS deals with both spatial and non-spatial attributes.
- It can take into account of a property related to non-spatial attribute(s), by means of a purity threshold, when finding the matching neighborhood.
- To increase the efficiency of clustering on large datasets we can use a heuristic that can reduce run time significantly at the cost of missing a probabilistically controlled number of clusters.

Conclusion

- The DBRS approach still needs improvement.
- First, our implementation must be improved by incorporating R*-trees or other specialized data structures for performing region queries.
- Secondly, the algorithm may miss joining certain clusters(Point 1 and Point 8 should be in one cluster).