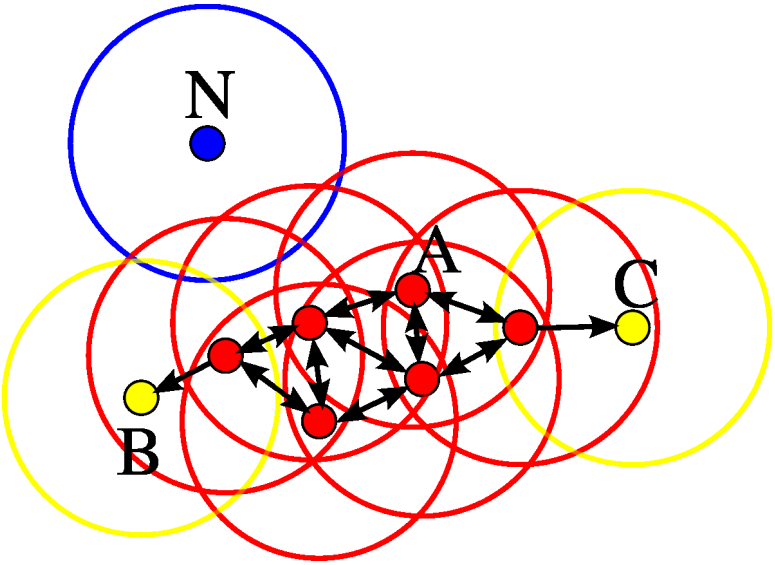
DBSCAN

Density-based spatial clustering of applications with noise (**DBSCAN**) is a data clustering algorithm proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996. It is a density-based clustering algorithm because it finds a number of clusters starting from the estimated density distribution of corresponding nodes. DBSCAN is one of the most common clustering algorithms and also most cited in scientific literature.[1] OPTICS can be seen as a generalization of DBSCAN to multiple ranges, effectively replacing the ε parameter with a maximum search radius.

Basic idea

DBSCAN's definition of a cluster is based on the notion of *density reachability*. Basically, a point q is *directly density-reachable* from a point p if it is not farther away than a given distance ε (i.e., is part of its ε-neighborhood) and if p is surrounded by sufficiently many points such that one may consider p and q to be part of a cluster. q is called *density-reachable* (note the distinction from "*directly* density-reachable") from p if there is a sequence $p_1, \ldots, p_n$ of points with $p_1 = p$ and $p_n = q$ where each p_{i+1} is directly density-reachable from p_i.



Points at A are core points. Points B and C are *density-reachable* from A and thus *density-connected* and belong to the same cluster. Point N is a noise point that is neither a core point nor density-reachable. (MinPts=3 or MinPts=4)

Note that the relation of density-reachable is not symmetric. q might lie on the edge of a cluster, having insufficiently many neighbors to count as dense itself. This would halt the process of finding a path that stops with the first non-dense point. By contrast, starting the process with p *would* lead to q (though the process would halt there, q being the first non-dense point). Due to this asymmetry, the notion of *density-connected* is introduced: two points p and q are density-connected if there is a point o such that both p and q are density-reachable from o. Density-connectedness *is* symmetric.

A cluster, which is a subset of the points of the database, satisfies two properties:

1. All points within the cluster are mutually density-connected.
2. If a point is density-connected to any point of the cluster, it is part of the cluster as well.

Algorithm

DBSCAN requires two parameters: ε (eps) and the minimum number of points required to form a cluster (minPts). It starts with an arbitrary starting point that has not been visited. This point's ε-neighborhood is retrieved, and if it contains sufficiently many points, a cluster is started. Otherwise, the point is labeled as noise. Note that this point might later be found in a sufficiently sized ε-environment of a different point and hence be made part of a cluster.

If a point is found to be a dense part of a cluster, its ε-neighborhood is also part of that cluster. Hence, all points that are found within the ε-neighborhood are added, as is their own ε-neighborhood when they are also dense. This process continues until the density-connected cluster is completely found. Then, a new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise.

Pseudocode

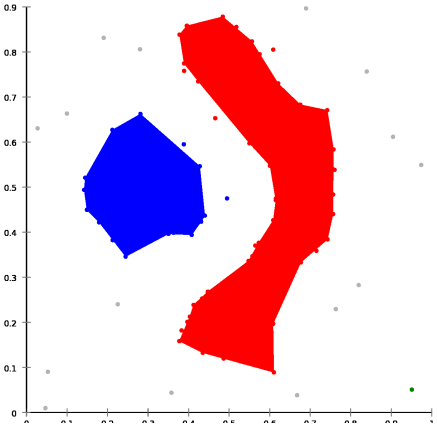
```
DBSCAN(D, eps, MinPts)
   C = 0
   for each unvisited point P in dataset D
      mark P as visited
      NeighborPts = regionQuery(P, eps)
      if sizeof(NeighborPts) < MinPts
         mark P as NOISE
      else
         C = next cluster
         expandCluster(P, NeighborPts, C, eps, MinPts)

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

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

Complexity

DBSCAN visits each point of the database, possibly multiple times (e.g., as candidates to different clusters). For practical considerations, however, the time complexity is mostly governed by the number of regionQuery invocations. DBSCAN executes exactly one such query for each point, and if an indexing structure is used that executes such a neighborhood query in $O(\log n)$, an overall runtime complexity of $O(n \cdot \log n)$ is obtained. Without the use of an accelerating index structure, the run time complexity is $O(n^2)$. Often the distance matrix of size $(n^2 - n)/2$ is materialized to avoid distance recomputations. This however also needs $O(n^2)$ memory.

Advantages

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



DBSCAN can find non-linearly separable clusters. This dataset cannot be adequately clustered with k-means or Gaussian Mixture EM clustering.

5. DBSCAN is designed for use with databases that can accelerate region queries, e.g. using an R* tree.

Disadvantages

1. The quality of DBSCAN depends on the distance measure used in the function regionQuery(P, ε). The most common distance metric used is Euclidean distance. Especially for high-dimensional data, this metric can be rendered almost useless due to the so-called "Curse of dimensionality", making it difficult to find an appropriate value for ε. This effect, however, is also present in any other algorithm based on Euclidean distance.
2. DBSCAN cannot cluster data sets well with large differences in densities, since the minPts- ε combination cannot then be chosen appropriately for all clusters.

See the section below on extensions for algorithmic modifications to handle these issues.

Parameter estimation

Every data mining task has the problem of parameters. Every parameter influences the algorithm in specific ways. For DBSCAN, the parameters ε and *minPts* are needed. The parameters must be specified by the user.

- *MinPts*: As a rule of thumb, a minimum *minPts* can be derived from the number of dimensions *D* in the data set, as $minPts \geq D+1$.
  The low value of $minPts=1$ does not make sense, as then every point on its own will already be a cluster. With $minPts=2$, the result will be the same as of hierarchical clustering with the single link metric, with the dendrogram cut at height ε (however, DBSCAN is substantially faster, because it does not compute the full dendrogram and can use indexes).
  However, larger values are usually better for data sets with noise and will yield more significant clusters. The larger the data set, the larger the value of *minPts* should be chosen.
- ε: The value for ε can then be chosen by using a k-distance graph, plotting the distance to the $k=minPts$ nearest neighbor. Good values of ε are where this plot shows a strong bend: if ε is chosen too small, a large part of the data will not be clustered; whereas for a too high value of ε, clusters will merge and the majority of objects will be in the same cluster.

OPTICS can be seen as a generalization of DBSCAN that replaces the ε parameter with a maximum value that mostly affects performance. *MinPts* then essentially becomes the minimum cluster size to find. While the algorithm is much easier to parameterize than DBSCAN, the results are a bit more difficult to use, as it will usually produce a

hierarchical clustering instead of the simple data partitioning that DBSCAN produces.

Recently, one of the original authors of DBSCAN has revisited DBSCAN and OPTICS, and published a refined version of hierarchical DBSCAN (HDBSCAN*), which no longer has the notion of border points.

Generalization

Generalized DBSCAN or **GDBSCAN** is a generalization by the same authors to arbitrary "neighborhood" and "dense" predicates. The ε and minpts parameters are removed from the original algorithm and moved to the predicates. For example on polygon data, the "neighborhood" could be any intersecting polygon, whereas the density predicate uses the polygon areas instead of just the object count.

Extensions

Various extensions to the DBSCAN algorithm have been proposed, including methods for parallelization, parameter estimation and support for uncertain data. The basic idea has been extended to hierarchical clustering by the OPTICS algorithm. DBSCAN is also used as part of subspace clustering algorithms like PreDeCon and SUBCLU. HDBSCAN is a revisited version of DBSCAN and OPTICS, from which informative clusters can be extracted using heuristics.

Availability

An implementation of DBSCAN is available in the ELKI framework. Notice that this implementation is not optimized for speed but for extensibility. Thus, this implementation can use various index structures for sub-quadratic performance and supports various distance functions and arbitrary data types, but it may be outperformed by low-level optimized implementations on small data sets.

scikit-learn includes a Python implementation of DBSCAN for arbitrary Minkowski metrics, based on kd-trees and ball trees.

GNU R contains DBSCAN in the "fpc" package with support for arbitrary distance functions via distance matrixes. However it does not have index support (and thus has quadratic runtime complexity).

References

[1] (http://academic.research.microsoft.com/CSDirectory/paper_category_7.htm) Most cited data mining articles according to Microsoft academic search; DBSCAN is on rank 24, when accessed on: 4/18/2010

Further reading

- Domenica Arlia, Massimo Coppola. "Experiments in Parallel Clustering with DBSCAN". *Euro-Par 2001: Parallel Processing: 7th International Euro-Par Conference Manchester, UK August 28–31, 2001, Proceedings*. Springer Berlin.
- Hans-Peter Kriegel, Peer Kröger, Jörg Sander, Arthur Zimek (2011). "Density-based Clustering" (http://wires.wiley.com/WileyCDA/WiresArticle/wisId-WIDM30.html). *WIREs Data Mining and Knowledge Discovery* **1** (3): 231–240. doi: 10.1002/widm.30 (http://dx.doi.org/10.1002/widm.30).

Article Sources and Contributors

DBSCAN *Source*: http://en.wikipedia.org/w/index.php?oldid=586638217 *Contributors*: 3mta3, AlanUS, AlexAlex, Angelo.romano, Arrakis3k, Chire, Dicklyon, Evert Mouw, Fnielsen, Fortdj33, Gareth Jones, Hbisgin, Hirsutism, Hpxchan, Ijon, Justinbozonier, Lewian, Lingeek, Mathiasl26, Nurnur, Peterbvolk, Pgr94, Pyfan, Qwertyus, R'n'B, Rjwilmsi, WikHead, Wombat, 94 anonymous edits

Image Sources, Licenses and Contributors

File:DBSCAN-Illustration.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:DBSCAN-Illustration.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Chire

File:DBSCAN-density-data.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:DBSCAN-density-data.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Chire

License