OPTICS algorithm

Ordering points to identify the clustering structure (**OPTICS**) is an algorithm for finding density-based clusters in spatial data. It was presented by Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel and Jörg Sander. Its basic idea is similar to DBSCAN, but it addresses one of DBSCAN's major weaknesses: the problem of detecting meaningful clusters in data of varying density. In order to do so, the points of the database are (linearly) ordered such that points which are spatially closest become neighbors in the ordering. Additionally, a special distance is stored for each point that represents the density that needs to be accepted for a cluster in order to have both points belong to the same cluster. This is represented as a dendrogram.

Basic idea

Like DBSCAN, OPTICS requires two parameters: ε, which describes the maximum distance (radius) to consider, and $MinPts$, describing the number of points required to form a cluster. A point p is a *core point* if at least $MinPts$ points are found within its ε-neighborhood $N_\varepsilon(p)$. Contrary to DBSCAN, OPTICS also considers points that are part of a more densely packed cluster, so each point is assigned a *core distance* that describes the distance to the $MinPts$ th closest point:

$$\text{core-distance}_{\varepsilon,MinPts}(p) = \begin{cases} \text{UNDEFINED} & \text{if } |N_\varepsilon(p)| < MinPts \\ \text{distance to the } MinPts\text{-th closest point} & \text{otherwise} \end{cases}$$

The *reachability-distance* of another point o from a point p is the distance between o and p, or the core distance of p:

$$\text{reachability-distance}_{\varepsilon,MinPts}(o,p) = \begin{cases} \text{UNDEFINED} & \text{if } |N_\varepsilon(p)| < MinPts \\ \max(\text{core-distance}_{\varepsilon,MinPts}(p), \text{distance}(p,o)) & \text{otherwise} \end{cases}$$

If p and o are nearest neighbors, this is the $\varepsilon' < \varepsilon$ we need to assume in order to have p and o belong to the same cluster.

Both the core-distance and the reachability-distance are undefined if no sufficiently dense cluster (w.r.t. ε) is available. Given a sufficiently large ε, this will never happen, but then every ε-neighborhood query will return the entire database, resulting in $O(n^2)$ runtime. Hence, the ε parameter is required to cut off the density of clusters that is no longer considered to be interesting and to speed up the algorithm this way.

The parameter ε is strictly speaking not necessary. It can be set to a maximum value. When a spatial index is available, it does however play a practical role when it comes to complexity. It is often claimed that OPTICS abstracts from DBSCAN by removing this parameter, at least to the amount of only having to give a maximum value.

Pseudocode

The basic approach of OPTICS is similar to DBSCAN, but instead of maintaining a set of known, but so far unprocessed cluster members, a priority queue (e.g. using an indexed heap) is used.

```
OPTICS(DB, eps, MinPts)
   for each point p of DB
      p.reachability-distance = UNDEFINED
   for each unprocessed point p of DB
      N = getNeighbors(p, eps)
      mark p as processed
      output p to the ordered list
```

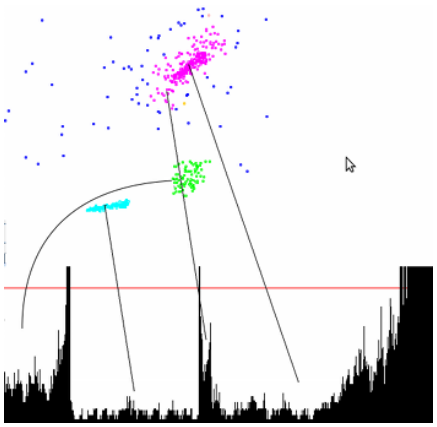
```
       Seeds = empty priority queue
       if (core-distance(p, eps, Minpts) != UNDEFINED)
          update(N, p, Seeds, eps, Minpts)
          for each next q in Seeds
             N' = getNeighbors(q, eps)
             mark q as processed
             output q to the ordered list
             if (core-distance(q, eps, Minpts) != UNDEFINED)
                update(N', q, Seeds, eps, Minpts)
```

In update(), the priority queue Seeds is updated with the ε-neighborhood of p and q, respectively:

```
update(N, p, Seeds, eps, Minpts)
   coredist = core-distance(p, eps, MinPts)
   for each o in N
      if (o is not processed)
         new-reach-dist = max(coredist, dist(p,o))
         if (o.reachability-distance == UNDEFINED) // o is not in Seeds
             o.reachability-distance = new-reach-dist
             Seeds.insert(o, new-reach-dist)
         else               // o in Seeds, check for improvement
             if (new-reach-dist < o.reachability-distance)
                o.reachability-distance = new-reach-dist
                Seeds.move-up(o, new-reach-dist)
```

OPTICS hence outputs the points in a particular ordering, annotated with their smallest reachability distance (in the original algorithm, the core distance is also exported, but this is not required for further processing).

Extracting the clusters



Using a reachability plot to identify clusters

Using a *reachability-plot* (a special kind of dendrogram), the hierarchical structure of the clusters can be obtained easily. It is a 2D plot, with the ordering of the points on the x-axis and the reachability distance on the y-axis. Since points belonging to a cluster have a low reachability distance to their nearest neighbor, the clusters show up as valleys in the reachability plot. The deeper the valley, the denser the cluster.

The image on the right illustrates this concept. In its upper half, an artificial example of a database consisting of two-dimensional, spatial points is shown. The lower part shows the reachability plot as computed by OPTICS. The black lines link some clusters to their respective valleys. The horizontal red line is an example on how to obtain a clustering. Each valley it crosses is made a cluster of its own. If the line was moved down, more clusters would emerge, especially for the topmost cluster, which features varying densities.

Note that deriving clusters in such a way yields the same result on core points of running DBSCAN on the data with ε set to the chosen reachability-distance threshold. The assignment of non-core points to neighboring clusters is non-deterministic in DBSCAN, too.

The blue points in this image are considered noise, and no valley is found in their reachability plot. This is subject to the ε parameter, which bounds the density of clusters.

A more advanced analysis does not use a specific value of ε, but instead looks for spikes that separate clusters. This can be used to obtain a hierarchical clustering that cannot be achieved by a single DBSCAN run.

Complexity

Like DBSCAN, OPTICS processes each point once, and performs one ε-neighborhood query during this processing. Given a spatial index that grants a neighborhood query in $O(\log n)$ runtime, an overall runtime of $O(n \cdot \log n)$ is obtained. The authors of the original OPTICS paper report an actual constant slowdown factor of 1.6 compared to DBSCAN. Note that the value of ε might heavily influence the cost of the algorithm, since a value too large might raise the cost of a neighborhood query to linear complexity.

In particular, choosing $\varepsilon > \max_{x,y} d(x,y)$ (larger than the maximum distance in the data set) is possible, but will obviously lead to quadratic complexity, since every neighborhood query will return the full data set. Even when no spatial index is available, this comes at additional cost in managing the heap. Therefore, ε should be chosen appropriately for the data set.

Extensions

OPTICS-OF is an outlier detection algorithm based on OPTICS. The main use is the extraction of outliers from an existing run of OPTICS at low cost compared to using a different outlier detection method.

DeLi-Clu, Density-Link-Clustering combines ideas from single-linkage clustering and OPTICS, eliminating the ε parameter and offering performance improvements over OPTICS.

HiSC is a hierarchical subspace clustering (axis-parallel) method based on OPTICS.

HiCO is a hierarchical correlation clustering algorithm based on OPTICS.

DiSH is an improvement over HiSC that can find more complex hierarchies.

Availability

Implementations of OPTICS, OPTICS-OF, DeLi-Clu, HiSC, HiCO and DiSH are available in the ELKI data mining framework (with index acceleration). An incomplete and slow implementation can be found in the Weka extensions. The MRC National Institute for Medical Research provides a C reimplementation of OPTICS [1] without index support.

References

[1] http://mathbio.nimr.mrc.ac.uk/wiki/Software#OPTICS

Article Sources and Contributors

OPTICS algorithm *Source*: http://en.wikipedia.org/w/index.php?oldid=579675207 *Contributors*: 1ForTheMoney, Alvin Seville, Arrakis3k, Bolinstephen, Chire, Gareth Jones, Hirsutism, Ivionday, John of Reading, Mild Bill Hiccup, Pgr94, Slowmo0815, 21 anonymous edits

Image Sources, Licenses and Contributors

Image:OPTICS.png *Source*: http://en.wikipedia.org/w/index.php?title=File:OPTICS.png *License*: Public Domain *Contributors*: Slowmo0815

License