The Relationship of DBSCAN to Matrix Factorization and Spectral Clustering

Erich Schubert[1], Sibylle Hess[2], and Katharina Morik[2]

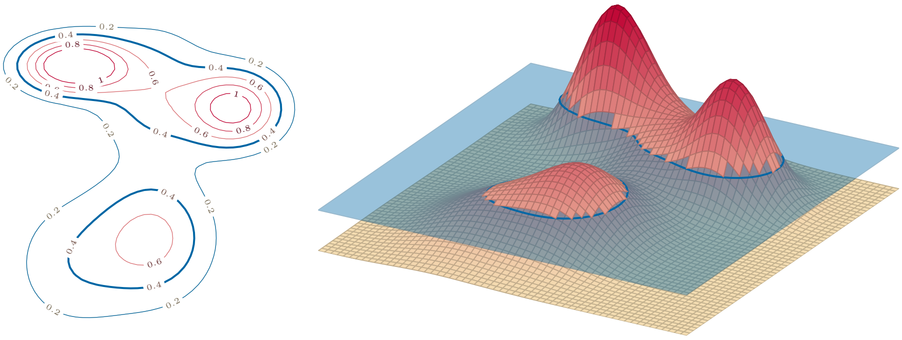
[1] Heidelberg University, Germany
schubert@informatik.uni-heidelberg.de
[2] TU Dortmund, Germany
{sibylle.hess,katharina.morik}@tu-dortmund.de

Abstract. DBSCAN is a popular approach for density-based clustering. In this short “work in progress” paper, we want to present an interpretation of DBSCAN as a matrix factorization problem, which introduces a theoretical connection (but not an equivalence) between DBSCAN and Spectral Clustering (SC).
While this does not yield a faster algorithm for DBSCAN, establishing this relationship is a step towards a more unified view of clustering, by identifying further relationships between some of the most popular clustering algorithms.

1 Introduction

The concept of density-based clustering was popularized by the seminal algorithm DBSCAN [5, 8, 9] and has sparked the development of a wide array of density-based clustering methods such as OPTICS [1], LSDBC [2], and HDBSCAN* [3].

The core idea of DBSCAN is the notion of density-connected sets of points, illustrated in Fig. 1. As in kernel density estimation, we interpret the data as samples drawn from a probability density function (pdf). In this example, we use a model composed of three Gaussian distributions. In Fig. 1a we plot the density contour lines, while Fig. 1b visualizes the pdf, and a cut at a density of 0.4, which yields two clusters (“islands”) in this plot. If we increased the density threshold to 0.6, the larger cluster would split into two clusters, at 0.8 the smaller cluster would disappear.



(a) Contour plot

(b) Density model with level set

Fig. 1: Density level set model

The DBSCAN algorithm is a database oriented technique to find such clusters, formulated around the concept of ε-radius queries (corresponding to using a uniform kernel of radius ε for density estimation). The algorithm begins at any unprocessed data point, and as long as the current point satisfies a density criterion (e.g., it has at least *minPts* neighbors), all the points neighbors are added to the same cluster. The algorithm then continues by expanding the cluster at each dense neighbor point. Once all expansions have been performed, the cluster is complete, and the algorithm can continue at any other unprocessed point to find another cluster. Non-dense points are either "noise", or "border" points, details of which can be found in [5, 8, 9].

With data index structures, a database system may be able to accelerate the ε-radius queries, and thus the algorithm. Without acceleration, the complexity of DBSCAN is that of computing all pairwise distances, i.e., $O(n^2)$ run-time. DBSCAN however does not need to store the entire distance matrix, and only uses $O(n)$ memory for object labels, a seed list of unprocessed objects, and the neighbors of a single object at a time.

Spectral clustering (SC) is based on the idea of finding the best normalized cut of the adjacency matrix [6]. Given the symmetric adjacency weight matrix $W \in \mathbb{R}^{n\times n}$ of n data points x_i, the normalized cut objective aims at finding a partition into k sets (clusters) $C_1 \ldots C_k$ solving the following optimization problem:

$$\min_{C_1,\ldots,C_k} \sum\nolimits_{s=1}^{k} \frac{\sum_{i\in C_s} \sum_{j\notin C_s} W_{ij}}{\sum_{i\in C_s} \sum_{j\in DB} W_{ij}}$$

which is the cost of cutting all the edges between different clusters, normalized by the total edge weight of each cluster. Like many clustering problems, this is NP-hard to solve exactly. Spectral clustering is based on a relaxed version of this optimization problem [6].

To find the clustering solution, we need to find the smallest eigenvalues and eigenvectors of the graph Laplacian. Eigenvectors to a zero eigenvalue indicate connected components (which obviously indicate optimal graph cuts, with zero cut edges). But for spectral clustering it is common to use k additional eigenvectors (corresponding to the k smallest eigenvalues) as a projection of the data (also referred to as "spectral embedding"), then SC uses k-means on this projection instead of the original data. For establishing the relationship to DBSCAN, we will use only these optimum cuts with zero cut edges, and do not need to consider the issues with k-means clustering here.

Different versions of the graph Laplacian exist, such as the unnormalized $L = D - W$ (where D is the diagonal containing the weight sums of each point), the normalized symmetric $L_{\text{sym}} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$ [7]; and the normalized random-walk $L_{\text{rw}} = D^{-1} L = I - D^{-1} W$ [10]. The normalized versions aim at creating more balanced clusters with respect to the edge weight, by normalizing the weights according to the node degrees.

The eigenvectors of the Laplacian are usually found based on singular value decomposition (SVD), approximated with power iterations, but could also be solved via nonnegative matrix factorization (NMF) [4]. The complexity of the factorization is $O(n^3)$, although in some cases on sparse graphs we can obtain speedups to $O(n^2)$. This makes SC one of the slower popular clustering techniques, and makes it difficult to scale this approach to large data sets.

2 DBSCAN as Matrix Factorization

While the original DBSCAN algorithm is a database oriented technique, we can also interpret it as a graph algorithm. We first consider only so-called "core" points $C \subseteq \mathit{DB}$, on which we will get a stronger result. In the standard DBSCAN setup, core points are exactly those points that have at least *minPts* neighbors within a distance of ε, i.e., points x with the binary predicate $\mathrm{core}(x) := |\mathrm{RangeQuery}(\mathit{DB}, \mathit{dist}, x, \varepsilon)| \geq \mathit{minPts}$. In the following, we use the symbol $C := \{x \in \mathit{DB} \mid \mathrm{core}(x)\}$ to denote the *subset* containing only core points. The core point graph then is defined by the edges

$$e_{\text{core}}(x_i, x_j) := 1 \text{ if } \mathit{dist}(x_i, x_j) \leq \varepsilon \wedge \mathrm{core}(x_i) \wedge \mathrm{core}(x_j)$$

and 0 otherwise. We use a numeric representation of the edges, rather than a set notation, because of the normalization and matrix notations used in spectral clustering.

Clusters in DBSCAN correspond to connected components in C. To reconstruct the full clustering on *DB*, we assign points $x \in \mathit{DB} \setminus C$ to a cluster if they are within distance ε of at least one core point in the cluster; otherwise it is labeled as "noise". This assignment of "border" points is not necessarily unique (c.f., [5, 9]), but it is not very common that a point is neighbor to two different clusters and not a core point itself.

Connected components arise in spectral clustering when factoring the graph Laplacian, as eigenvectors with an eigenvalue of 0. Each connected component is a cut with cost 0, and thus an optimal cluster (the k-means step in spectral clustering is only necessary to find further substructures). But the connected components of e_{core} are *exactly* the DBSCAN cluster cores, and hence we get the DBSCAN results except for noise and border points. If we next consider the full DBSCAN clusters, the possibility that a border point connects two clusters however becomes problematic. The full reachability graph of DBSCAN is (note the $\vee$ – it is sufficient if one of the points is core):

$$e_{\text{symmetric}}(x_i, x_j) := 1 \text{ if } \mathit{dist}(x_i, x_j) \leq \varepsilon \wedge \big(\mathrm{core}(x_i) \vee \mathrm{core}(x_j)\big)$$

Spectral Clustering (SC) is based on the notion of minimum cuts. In the situation where a border point connects two clusters, this will usually yield a minimum cut, i.e., spectral clustering will choose the desired cluster split.

In the logic of DBSCAN, the graph is however not symmetric. Only from core points we can reach neighbors, and this is also reflected in later methods such as OPTICS. From this point of view, the following edge definition is more appropriate:

$$e_{\text{asymmetric}}(x_i \to x_j) := 1 \text{ if } \mathit{dist}(x_i, x_j) \leq \varepsilon \wedge \mathrm{core}(x_i)$$

If we use, e.g., power iterations to approximate the eigenvectors of the Laplacian, we do not run into major problems even for a non-symmetric matrix (an exact solution may in the worst case involve complex eigenvalues and eigenvectors because of the asymmetry; but it is easy to see that the random-walk interpretation of power iterations means we can find clusters as sets that are closed under random walks). Non-core nodes x_j are simply "dead ends" in a random walk. Non-core points that are reachable from more than one cluster will be non-zero in more than one eigenvector. Formally, this means the vectors no longer are orthogonal, but for the purpose of clustering this is not problematic. Intuitively, an eigenvector of this matrix is a set of points such that when following all edges, we obtain the same set ($A \cdot x = x$). Finding the eigenvectors is, of course, a matrix factorization.

3 Spectral Density-Based Clustering

This edge graph we use here is a subtle variation of a common graph used in spectral clustering: the ε-neighborhood graph [6] places an edge whenever two points have a distance of at most ε. Here, based on DBSCAN, we modify this with a minimum density requirement, by omitting edges where the points are not "dense".

It is easy to see that most of the graph will be identical – in particular the graph restricted to the cluster core. As the differences are fairly subtle, we cannot expect this approach to produce very different results except that we do get more completely disconnected points (in particular for larger values of *minPts*). For users of spectral clustering, this small modification may be worth exploring.

While the ε-neighbor graph is usually unweighted, alternatives such as the kNN graph are often weighted by their distance. Here, we can incorporate ideas from OPTICS clustering (a hierarchical extension of DBSCAN, that no longer needs a fixed ε threshold). The key notion of OPTICS is the (asymmetric!) reachability, which can be defined as

$$\text{reachability}(x_i \to x_j) := \max\{\mathit{dist}(x_i, x_j), \mathit{minPts}\text{-dist}(x_i)\}$$

where *minPts*-dist is the distance to the *minPts* nearest neighbor. However, for spectral clustering we need a similarity and not a distance matrix, and we thus need to also choose an appropriate transform, e.g., the Gaussian similarity of bandwidth σ:

$$s(x_i \to x_j) := \tfrac{1}{\sqrt{2\pi\sigma^2}} \exp -\tfrac{|\,\text{reachability}(x_i \to x_j)|^2}{2\sigma^2}.$$

We expect this to yield an interesting hybrid method: Because the *minPts*-nearest neighbors all have the same weight, the approach will behave more like unweighted spectral clustering on "dense" structures, and more like the kNN graph on sparse points. In particular for larger values of *minPts*, this introduces a smoothing property previously found beneficial in OPTICS. Yet, the difference may often be too small to have a measurable impact on common evaluation metrics.

4 Complexity

Finding the edge graph, unfortunately, is already as expensive as running the original DBSCAN or OPTICS algorithms. Since these run in worst-case $O(n^2)$, we cannot expect any of above variations to outperform the original methods. In fact, just storing the edge graph will consume more memory than either DBSCAN or OPTICS even when using a sparse matrix, and thus the original algorithms remain the preferred choice.

If we intend to use spectral clustering, we can however incorporate the density-based ideas from DBSCAN and OPTICS more easily. When using the DBSCAN-graph in spectral clustering, it is a good idea to build a separate edge graph only for each DBSCAN cluster. We can then easily run spectral clustering only on each connected component separately, which is more efficient. Furthermore, the construction of the similarity graph can then be accelerated using a database index.

With the reachability-graph in spectral clustering, we will not as obviously get such computational benefits. Experience in OPTICS shows that it may be worth truncating the neighbors at a certain threshold, and the use of the exponential function in the Gaussian similarity also yields a similarity that drops off quickly to effectively zero.

When ignoring neighbors outside of a $\varepsilon_{\max}$ radius, OPTICS can be index accelerated. We can employ the same technique in spectral clustering, by choosing an $\varepsilon_{\max}$ query radius that is, e.g., $\varepsilon_{\max} = 5\sigma$, where the density has already reduced by a factor of $3.7 \cdot 10^{-6}$. This will produce a more sparse graph, which may be faster to compute.

5 Conclusions

In this short paper, we established a connection (not an equivalence) between density-based DBSCAN clustering, matrix factorization, and spectral clustering: the clusters found by DBSCAN correspond (*except* for border and noise points) to optimal cuts of the neighborhood graph of the core points *only*, and hence, to spectral clusters.

While this does not yield a faster clustering algorithm – computing the graph is as expensive as running DBSCAN – the theoretical relationship of these methods is interesting. In future work, we plan to study consequences of applying other factorization strategies onto this matrix to improve DBSCAN results, but also to integrate the notion of density into that of spectral clustering. We would like to investigate the feasibility of doing the opposite connection: reformulate spectral clustering in a DBSCAN-like way in order to find more efficient, potentially index accelerated, algorithms to accelerate the rather expensive spectral clustering algorithms without storing the large graph explicitly.

Bibliography

[1] M. Ankerst, M. M. Breunig, H. Kriegel, and J. Sander. “OPTICS: Ordering Points To Identify the Clustering Structure”. In: *ACM SIGMOD*. 1999, pp. 49–60.

[2] E. Biçici and D. Yuret. “Locally Scaled Density Based Clustering”. In: *ICANNGA*. 2007, pp. 739–748.

[3] R. J.G. B. Campello, D. Moulavi, and J. Sander. “Density-Based Clustering Based on Hierarchical Density Estimates”. In: *PAKDD*. 2013, pp. 160–172.

[4] C. H. Q. Ding and X. He. “On the Equivalence of Nonnegative Matrix Factorization and Spectral Clustering”. In: *SIAM SDM*. 2005, pp. 606–610.

[5] M. Ester, H. Kriegel, J. Sander, and X. Xu. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *ACM KDD*. 1996, pp. 226–231.

[6] U. von Luxburg. “A tutorial on spectral clustering”. In: *Statistics and Computing* 17.4 (2007), pp. 395–416.

[7] A. Y. Ng, M. I. Jordan, and Y. Weiss. “On Spectral Clustering: Analysis and an algorithm”. In: *NIPS*. 2001, pp. 849–856.

[8] J. Sander, M. Ester, H. Kriegel, and X. Xu. “Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications”. In: *Data Min. Knowl. Discov.* 2.2 (1998), pp. 169–194.

[9] E. Schubert, J. Sander, M. Ester, H. Kriegel, and X. Xu. “DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN”. In: *ACM TODS* 42.3 (2017), 19:1–19:21.

[10] J. Shi and J. Malik. “Normalized Cuts and Image Segmentation”. In: *IEEE TPAMI* 22.8 (2000), pp. 888–905.