λ-DBSCAN: Augmenting DBSCAN with Prior Knowledge

Joel Dierkes, Daniel Stelter, and Christian Braune

Otto von Guericke University
Institute for Simulation and Graphics
Magdeburg, Germany
{joel.dierkes,daniel.stelter,christian.braune}@ovgu.de

Abstract. State-of-the-art density based cluster algorithms offer remarkable speed and robustness. However, they do not allow the user to make local changes without affecting the global outcome. The user thus has to choose between clustering a local region well or keeping the global result. We present a new approach, λ-DBSCAN, which augments the DBSCAN algorithm to include local a priori knowledge. The parameters can be specified per observation, rather than globally, which enables the user to include local knowledge about the data without modifying other regions. Furthermore, we define regions in the data that should be affected by certain parameter choices, to reduce the workload for a user.

Keywords: prior knowledge · density-based clustering · DBSCAN.

1 Introduction

Finding relations in data is one of the most common problems in data analysis. For the unsupervised task of clustering the goal is to combine similar observations to clusters, whereas dissimilar observations should be in different clusters. However, the solution to this task can often be ambiguous, since in general there is no *perfect* clustering, i.e. a given ground truth, but rather multiple good ones. This is especially true for real life datasets, since normally no ground truth exists at all and all clusterings are open for interpretation by the viewer.

There exists a variety of cluster models, such as connectivity-, centroid-, and density-based models. Density models assume a cluster to be a dense region of observations bounded by less dense regions. The number of clusters does not have to be known *a priori*, and neither are any assumption made about their possible shapes. In contrast centroid-based clustering algorithms (like k-means [6]) implicitly assume that all points within a cluster can be properly represented by a single point. However, trivial density-based clustering algorithms, like DBSCAN [3], encounter difficulties if the non-dense regions of observations that divide dense regions do not clearly separate clusters. Several state-of-the-art density based cluster models, like OPTICS [1] or HDBSCAN [2], can successfully find clusters with varying densities. They do however tend to perform poor on datasets with close high density clusters in combination with distant low density ones [7].

Each approach to find clusters in a given data set must make assumptions about what a cluster actually is and how clusters should be separated. These assumptions are usually made implicitly by the choice of the clustering algorithm (e.g. k-means finds hyperspherical clusters, or single linkage clustering is suitable to follow lines in the data set) or explicitly by the choice of parameters for a clustering algorithm (e.g. the choice of the metric used to calculate (dis)similarites, or the choice of linkage method for hierarchical clustering). Another way to direct the clustering process would be to incorporate *a priori* knowledge into the clustering process.

However, even with *a priori* knowledge there might not be a sufficiently good choice of a clustering method and its parameters. This can be the case for datasets which show too large variety in different regions of the domain. Although there are algorithms which try to adapt to local properties of the data, they still can fail as the selected global parameters might only deliver good results for some (or even only for single) regions of the whole domain.

In this work we present λ-`DBSCAN` which is a modification of the `DBSCAN` algorithm presented in [3]. Our idea is to extend the algorithm slightly to allow the integration of prior knowledge of the data to the parameters. This enables us to change the way clusters are formed depending on the specific location of a data point, i.e., besides *global* prior knowledge the user is able to apply *local* prior knowledge as well.

2 Related Work

`DBSCAN` is a well studied density based cluster algorithm. Each observation is classified as either a core point, a border point, or noise. The algorithm defines a local neighborhood as a hypersphere of radius ε around each point and counts the number of other data points within this neighborhood to decide which class (core, border, or noise) a point belongs to.

$$N_\varepsilon(\boldsymbol{p}) = \{q \in D \mid dist(\boldsymbol{p}, \boldsymbol{q}) \leq \varepsilon\}. \tag{1}$$

A point $\boldsymbol{p}$ is considered to be a *core* point if and only if the size of its ε-neighborhood is at least $minPts$ or μ:

$$cores_{\varepsilon,\mu} = \{\boldsymbol{p} \in D \mid \mu \leq |N_\varepsilon(\boldsymbol{p})|\} \tag{2}$$

Thus, ε and μ are the two major parameters of `DBSCAN`.

Furthermore, an observation is a border point if at least one and less than μ observations are contained within its ε-neighborhood, otherwise it is noise

$$borderPoints_{\varepsilon,\mu} = \{\boldsymbol{p} \in D \mid 1 \leq |N_\varepsilon(\boldsymbol{p})| < \mu\} \tag{3}$$

$$\begin{aligned} noise_{\varepsilon,\mu} &= \{\boldsymbol{p} \in D \mid 0 = |N_\varepsilon(\boldsymbol{p})|\} \\ &= D \setminus (cores_{\varepsilon,\mu} \cup borderPoints_{\varepsilon,\mu}) \end{aligned} \tag{4}$$

To form clusters `DBSCAN` needs to decide which of the core and border points belong into the same cluster. For this [3] defines a set of relations to describe the

interaction between different points. The first is *directly density reachable*. Two points are directly density-reachable if their distance is less than ε and at least one of them is a core point. Secondly, two points $\boldsymbol{p}$ and $\boldsymbol{q}$ are *density reachable* if there exists a sequence of mutually directly density reachable points which starts with $\boldsymbol{p}$ and ends with $\boldsymbol{q}$. Clusters are then formed by the transitive closure of the density reachability relation.

Note here that $\boldsymbol{p}$ is in the ε-neighborhood of a point $\boldsymbol{q}$ if and only if $\boldsymbol{q}$ is in the ε-neighborhood of $\boldsymbol{p}$:

$$\boldsymbol{p} \in N_\varepsilon(\boldsymbol{q}) \Leftrightarrow \boldsymbol{q} \in N_\varepsilon(\boldsymbol{p}) \tag{5}$$

While `DBSCAN` enables the user to find clusters with similar densities, it struggles to produce good results if the densities within different clusters vary vastly. Other versions of the `DBSCAN` algorithms have been invented to cope with its limitations, such as `OPTICS` [1] or `HDBSCAN` [2].

`OPTICS` [1] (Ordering Points To Identify the Clustering Structure) is a density-based clustering algorithm designed to find patterns in spatial datasets. It generates a reachability plot, revealing clusters with varying density and hierarchies. `HDBSCAN` [2] (Hierarchical Density-Based Spatial Clustering of Applications with Noise) leverages a hierarchical approach combined with a density-based one. It creates a tree of cluster relationships which allows the algorithm to find clusters with varying densities.

A completely different approach able to cope with clusters of varying density would be spectral clustering [9] which turn clustering into a graph-based problem and leverages the eigenvalues and eigenvectors of the similarity matrix to identify clusters. It transforms the data into a lower-dimensional space using the similarity matrix and runs a cluster algorithm on these structures.

While these also find clusters with varying densities, they fail to produce good results in situations with close high-density clusters together with distant low-density ones. All of these extensions to `DBSCAN` have in common, that they use a set of global parameters which cannot be locally adapted. Although these parameter choices may have local influences they are usually not very intuitive to determine.

Another approach that is able to achieve comparable results is C-DBSCAN [14], which uses pairwise *must-link* and *cannot-link* constraints to force pairs of points into either the same cluster or into different clusters. *Cannot-link* constraints can be used to separate a point from another cluster that it would normally belong to. It could be seen as locally setting μ or ε to such values that each of the points involved in the constraint is not contained in the others ε-neighborhood. *Must-link* constraints on the other hand merge clusters that would not be combined by DBSCAN because no two points' ε-neighborhoods overlap sufficiently. This can merge clusters that are close to each other (which could be seen as modifying μ or ε again) but also those which are too dissimilar to each other to be found by purely distance- based approaches. For both types of constraints these have to be given for each involved pair of data points. Creating a large set of constraint may therefore be prohibitively expensive or counter-intuitive.

3 Implementation

Algorithm 1 Pseudocode of the λ-DBSCAN Algorithm

Input: DB: Database
Input: ε: Neighborhood size
Input: μ: MinPts
Input: $dist$: Distance function
Output: $label$: Points labels, initially undefined

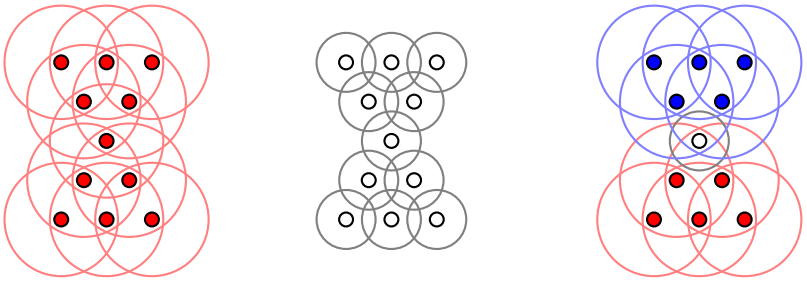
for $\boldsymbol{p} \in DB$ **do**
  if $label(\boldsymbol{p}) \neq$ undefined **then continue end if**
  Neighbors $N \leftarrow RangeQuery(DB, dist, \boldsymbol{p}, \varepsilon(\boldsymbol{p}))$
  if $|N| < \mu$ **then**
    $label(\boldsymbol{p}) \leftarrow$ Noise
    continue
  end if
  $c \leftarrow$ next cluster label
  $label(\boldsymbol{p}) \leftarrow c$
  Seed set $S \leftarrow N \setminus \{p\}$
  for $\boldsymbol{q} \in S$ **do**
    if $label(\boldsymbol{q}) =$ Noise **then** $label(\boldsymbol{q}) \leftarrow c$ **end if**
    if $label(\boldsymbol{q}) \neq$ undefined **then continue end if**
    Neighbors $N \leftarrow RangeQuery(DB, dist, \boldsymbol{q}, \varepsilon(\boldsymbol{p}))$
    $label(\boldsymbol{q}) \leftarrow c$
    if $|N| < \mu$ **then continue end if**
    $S \leftarrow S \cup N$
  end for
end for

When we want to extend DBSCAN in a way that allows us to incorporate prior knowledge (or different understandings of which points should be able to form a cluster and which not) the only parameters within the original algorithm are ε and μ and thus our first target for adjusting DBSCAN's clustering behaviour.

The λ-DBSCAN algorithm is based on the DBSCAN implementation. We still distinct between core, border, and noise points. The parameter ε which is used to determine which of these classes a point belongs to can be chosen for individual points rather than using one global value for each point, respectively. For the sake of this paper we will not alter the meaning of μ as our adaption of ε is sufficient. Technically a user of λ-DBSCAN could decide for each point individually which class a point belongs to (without calculating the membership to each class) but this will turn any practical application infeasible, of such a decision would have to be made for many points, points in a high-dimensional space, or both. Instead we select these parameters for whole regions of points, which enables the algorithm to be reasonably autonomous. The algorithm only relies on provided domain knowledge, if needed, except for one initial global choice of parameters.

The effect of this can be seen in Figure 1 where we see 11 points arranged on a grid of equilateral triangles such that for every choice of ε and μ DBSCAN must

either assign all points to the same cluster or to no cluster at all. Our Proposed algorithm λ-DBSCAN can however be parameterized in a way that the upper five points and the lower five points can be separated.



(a) DBSCAN (b) DBSCAN (smaller ε) (c) Our λ-DBSCAN

Fig. 1: Clustering of equidistant points with the classical DBSCAN and λ-DBSCAN.

Algorithm 1 outlines an implementation with a global μ and individual ε based on [15, Chapter 2.2]. The changes have been marked in Algorithm 1. Note that we use a function $\varepsilon(p)$ which returns individual εs for each data point. Hence the name λ-DBSCAN, coming from the Python unnamed `lambda` functions. Regarding a practical implementation each individual ε could be pre-computed and stored in a list. This only adds a linear complexity to the algorithm (w.r.t. to the complexity class of $\varepsilon(p)$ which will mostly be in $\mathcal{O}(1)$) and therefore does not impact the performance by much (cf. Table 1). However, this computation is only performed once before the clustering starts and will diminish with larger datasets.

It is important to notice here that Equation (5) does not necessarily hold anymore. If set accordingly, a point $\boldsymbol{p}$ could be in the ε-neighborhood of another point $\boldsymbol{q}$, without $\boldsymbol{q}$ being the in the ε-neighborhood of $\boldsymbol{p}$. We see three possible solutions:

1. Rely on the order in which the algorithm traverses the dataset.
2. Keep the original interpretation of Equation (5) intact and only consider $\boldsymbol{p}$ and $\boldsymbol{q}$ as *directly density reachable* if they are mutually contained in each other's ε-neighborhoods
3. Alter the interpretation of Equation (5) such that $\boldsymbol{p}$ and $\boldsymbol{q}$ as *directly density reachable* if either is contained in the other's ε-neighborhood

We discard the first approach, since the clustering should be independent of the iteration of the observations. We tested implementations 2. and 3., both yielded the same result for all datasets in this paper.

4 Evaluation

In the following, we compare results of λ-DBSCAN with selected algorithms on a variety of artificial and two real world datasets. Table 2 shows the parameters we used for each algorithm to cluster the synthetic datasets. Table 1 shows the runtimes and scores.

4.1 Synthetic Datasets

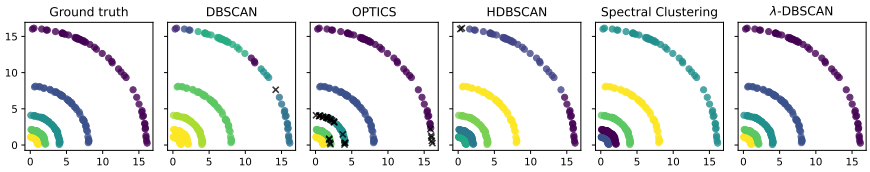


Fig. 2: Different clustering results for a dataset with clusters that form concentric rings. The number of observations per cluster are the same, the densities vary due to the different radii.

Rings: Figure 2 shows different cluster results for an artificial concentric ring data. All rings contain the same number of elements, which results in different densities because of their varying radii. Such datasets pose a challenge for density based cluster algorithms. In order to separate the two inner rings one has to choose a rather small value for ε which in turn leads to the points on the outermost ring to not be assigned to the same cluster. With the variation in density in side of the same clusters even HDBSCAN and OPTICS sometimes either detect noise where none should be or separates one cluster into two different clusters. Only spectral clustering yields a perfect result with the nearest neighbor affinity.

λ-DBSCAN allows us to define a ε parameter based on the location of the points. Since it is known a priori that the clusters get sparser the farther away they are from the origin, one can set the ε parameter to a value depending on the distance of the point to the origin – in this case to one third of the distance of a point to the origin. Spectral clustering however needed more than 28 times longer to compute this result than DBSCAN or λ-DBSCAN (see Table 1).

Spirals: Figure 3 displays cluster results for an artificial spiral data. The spirals are meet at the origin, which makes them indistinguishable for any basic density based cluster algorithms. The points are farther away from each other the further they are away from the origin, resulting in clusters with varying densities. DBSCAN and OPTICS both fail to yield a result comparable to the ground truth, not matter how the parameter are chosen. Spectral clustering finds three clusters (because it has to given the parameter choice) but those have virtually nothing

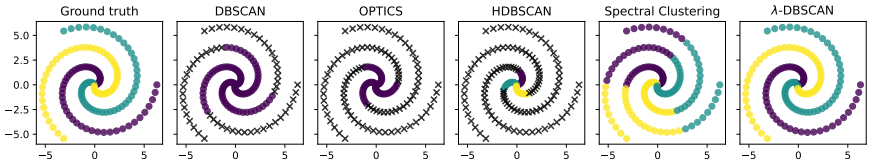


Fig. 3: A dataset with clusters that form concentric spirals. The number of observations per cluster are the same, the densities within each cluster decreases the further away a point is from the origin.

in common with the desired result. This stems from the underlying cluster model that is implicitly encoded in the algorithm itself.

λ-DBSCAN can use knowledge about local regions and thus separate the clusters around the origin. At the same time, it can use the knowledge about the spirals to solve the issue of varying densities in the clusters itself. Figure 3 and Table 2 show that the way we calculate the individual ε is rather simple. A more sophisticated choice might even lead to a perfect clustering result around the origin.

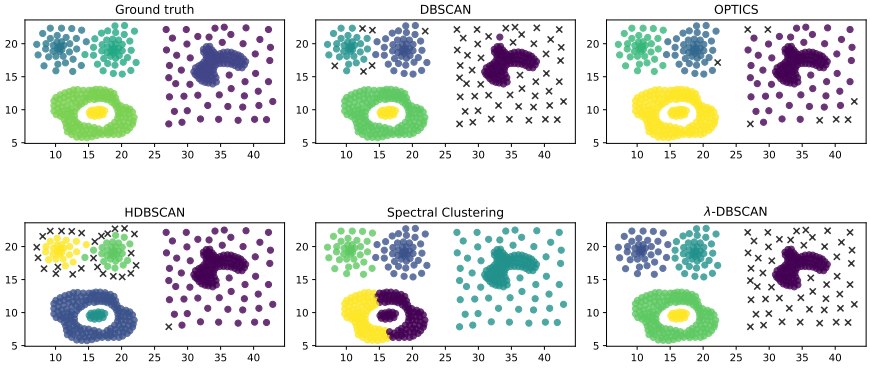


Fig. 4: The ground truth and five different clusterings of the compound dataset.

Zahn's Compound: The Compound dataset [4] represents a challenge for spectral clustering, since it can not clearly distinguish the two clusters where one cluster encapsulates another cluster with different densities. OPTICS fails to distinguish them at all. Since DBSCAN uses the ε and μ globally it fails to cluster the shapes without noise. λ-DBSCAN produces results better than DBSCAN while taking the same amount of time.

4.2 Real World Datasets

GPS pickup data: The GPS pickup dataset is a collection of roughly 2′800 GPS coordinates of pick-up and drop-off locations of customers of a ride pooling

Table 1: Comparison of the runtime and ARI scores for each algorithm and dataset. The PC used for the experiments features an AMD Ryzen 7 5700G processor and 32 GB DDR4-RAM.

Algorithm	Runtime in ms	ARI	Homogeneity	Completeness	V-measure
`Rings`					
DBSCAN	**1.3**	0.689	0.828	0.820	0.824
OPTICS	106.4	0.867	0.938	0.854	0.894
HDBSCAN	2.6	0.940	**1.000**	0.915	0.956
Spectral Clustering	36.7	**1.000**	**1.000**	**1.000**	**1.000**
λ-DBSCAN	**1.3**	**1.000**	**1.000**	**1.000**	**1.000**
`Spirals`					
DBSCAN	**1.0**	-0.008	0.000	0.000	0.000
OPTICS	63.3	-0.008	0.000	0.000	0.000
HDBSCAN	1.3	0.034	0.220	0.314	0.259
Spectral Clustering	13.7	0.065	0.072	0.072	0.072
λ-DBSCAN	1.2	**0.921**	**0.898**	**0.898**	**0.898**
`Zahn's Compund`					
DBSCAN	**1.4**	0.976	0.949	0.953	0.951
OPTICS	164.1	0.788	0.767	0.942	0.845
HDBSCAN	2.9	0.826	0.819	0.900	0.858
Spectral Clustering	83.2	0.585	0.776	0.809	0.792
λ-DBSCAN	1.5	**0.997**	**0.993**	**0.992**	**0.992**

company. The aim is to find a suitable selection of locations that represent a pick-up or drop-off location for a group of users – or in other terms: find clusters in the GPS data. The dataset contains groups of GPS points with varying densities and member counts, as well as different distances between the clusters.

While [7] finds clusters with varying densities and few points, they fail to split close clusters with high densities. The train station, for example, is one big cluster, while it could possibly be an aggregation of four to five close but separate clusters. None of the referenced algorithms in [7] can cluster these close bigger clusters effectively, leaving the customers with only one pick-up and drop-off location for a rather large area. λ-`DBSCAN` can select these regions in the data with appropriate ε and μ values.

Application in Flow Visualization We also applied λ-`DBSCAN` to a practical example in the field of flow visualization. We consider the *finite-time Lyapunov exponent (FTLE)* for a 2D flow. FTLE fields are scalar fields, its ridges are lines which one can see as a skeleton of the underlying flow. This skeleton divides the flow into multiple regions. Material inside the same region which is moved by the flow over time stays comparably close to each other over time. The extraction of FTLE ridges can greatly assist in many modern fields, e.g. path planning for autonomous underwater vehicles [13], tracking the movement of air pollution and its effects [10], and understanding medical phenomenons like aortic valve calcifications [8]. [16] recently proposed a new efficient and reliable attempt for

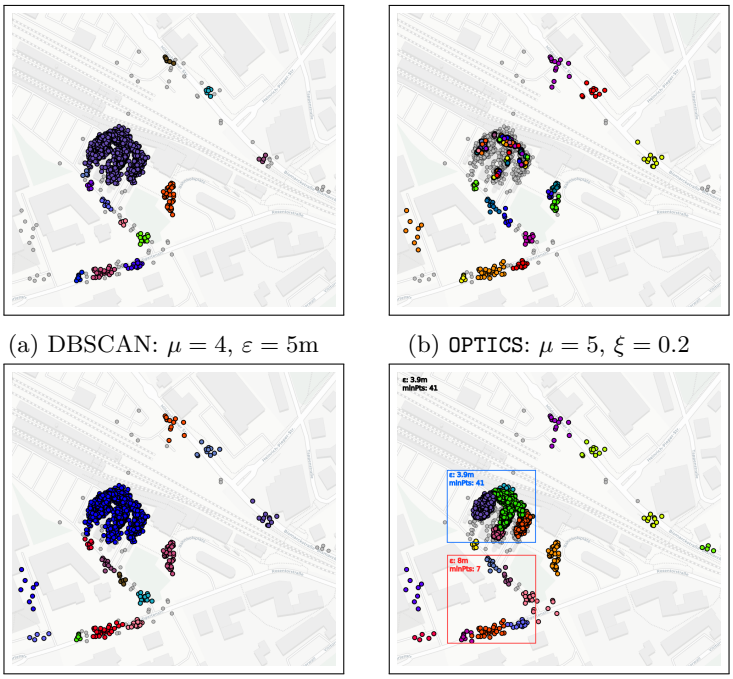
Table 2: Parameters for the artificial datasets.

Algorithm	Parameters
`Rings`	
DBSCAN	$\varepsilon = 1, \mu = 2$
OPTICS	$\mu = 26$
HDBSCAN	$\mu = 6$
Spectral Clustering	$k = 5$, affinity = nearest neighbors
λ-DBSCAN	$\varepsilon = \lvert p\rvert/3, \mu = 3$
`Spirals`	
DBSCAN	$\varepsilon = 0.5, \mu = 2$
OPTICS	$\mu = 10$
HDBSCAN	$\mu = 1$
Spectral Clustering	$k = 3$, affinity = nearest neighbors
λ-DBSCAN	$\mu = 3, \varepsilon = \begin{cases} \lvert p\rvert^{0.2} & \lvert p\rvert > 1 \\ 0.1 & else \end{cases}$
`Zahn's Compund`	
DBSCAN	$\varepsilon = 1.48, \mu = 3$
OPTICS	$\mu = 20$
HDBSCAN	$\mu = 7$
Spectral Clustering	$k = 5$, affinity = rbf
λ-DBSCAN	$\mu = 5, \varepsilon = \begin{cases} 1 & \lvert p - (16.5, 9.5)^T\rvert < 7 \vee p_0 > 25 \\ 0.0001 & 13.25 < p_0 < 17.5 \wedge 15 < p_1 \\ 2.5 & else \end{cases}$

sampling FTLE ridges with a population of particles. It results in a uniform sampling of these structures. Unfortunately, ridges in FTLE fields tend to be sharp and closely located to each other which leads to numerical difficulties. [16] point out the need for an appropriate clustering method in order to to detect outliers and to connect the correct ridges. We applied the cluster methods to an extraction example for a 2D flow around a cylinder inside a channel [5,12]. In this dataset the flow streams from left to right and is bounded by solid walls.

The FTLE ridge extraction resulting in 22,000 particles is shown in Figure 6. The particles create many lines on the left side of the cylinder, indicating a strong turbulence with thin coherent regions. Further to the right, there are fewer lines, thus, less turbulence. However, numerical problems lead to outliers and partly discontinuous structures.

Figure 6 also displays five different clusterings of the `ridges` dataset. `DBSCAN` with $\varepsilon = 0.0008$ and $\mu = 3$ does yield a meaningful clustering of ridges that are further away from the origin (approx. $x > 0.5$) with only some ridges merged. The ridges near the origin however can not be separated at all. Other parameter choices yield a better result for the ridges near the origin, but fracture the clusters further away. `OPTICS` with $\mu = 2$ and $\xi = 0.00001$ results in a lot of noise mislabeling, while other parameter choices merge many ridges into one. `HDBSCAN` with $\mu = 13$ does a good job of finding ridges in less turbulent regions, however



(a) DBSCAN: $\mu = 4$, $\varepsilon = 5$m

(b) `OPTICS`: $\mu = 5$, $\xi = 0.2$

(c) `HDBSCAN` ($\hat{\varepsilon}$): $\mu = 3$, $\hat{\varepsilon} = 5$m

(d) λ-`DBSCAN`: ε and μ shown in their respective regions.

Fig. 5: Different results for the GPS dataset. Parameters taken from [7] (except for λ-`DBSCAN`)

it also is unable to differentiate the lines in turbulent regions. Spectral clustering with $k = 50$ and the nearest neighbors affinity splits and joins lines almost at random, also merging the lines in the turbulent regions.

Given the prior knowledge that turbulent regions form around the origin and contain more close ridges, while less turbulent regions form further away from the origin and contain fewer ridges which are further spread apart, one can set the parameters for the λ-`DBSCAN` algorithm accordingly. A clustering with

$$\mu = 3,\ \varepsilon(x, y) = \begin{cases} 0.0035 & x < 0.5 \\ 0.008 & else \end{cases}$$

is seen at the bottom of Figure 6. The algorithm detects ridges around the origin, with only some being merged that are very close to each other.

5 Conclusion and Future Work

In this paper, we proposed a new algorithm for using local specific knowledge λ-`DBSCAN`. Effectively our algorithms allows the user to run different instances

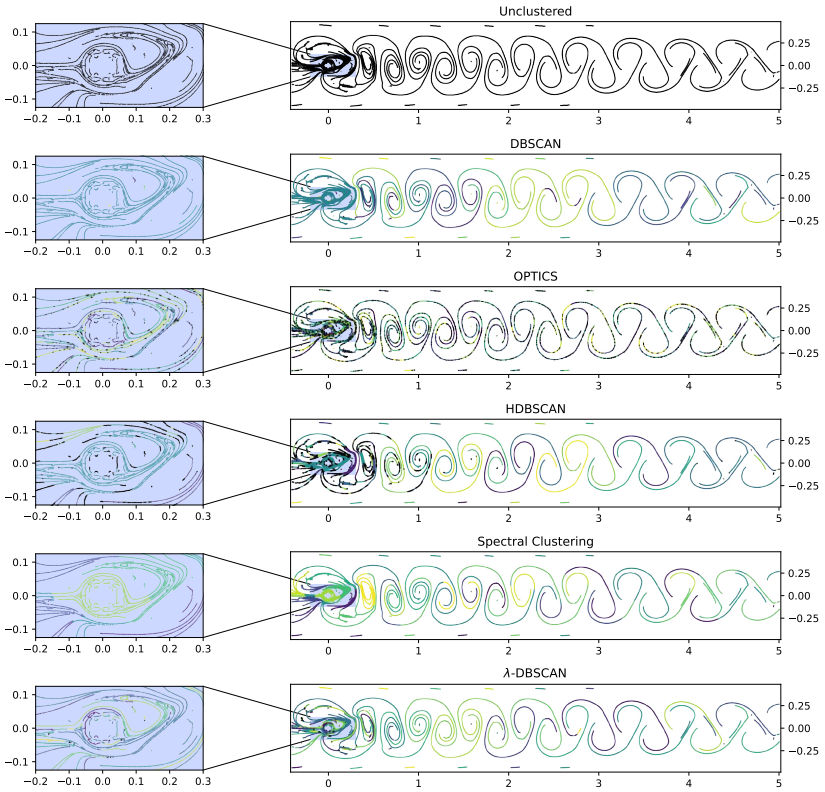


Fig. 6: The `ridges` dataset and five possible clusterings.

of the `DBSCAN` algorithm on different portions of the dataset while seamlessly merging their outcomes into a single result.

The speed of λ-`DBSCAN` enables the algorithm to be used in interactive applications and be a drop-in for a pure `DBSCAN` implementation since it is usually not slower while yielding at least the same – if not better – results. A program that leverages the user as an oracle who marks regions that need parameter changes comes to mind allowing some form of semi-supervised clustering. The cluster result could update in real time (only re-cluster the parts that changed) and thus allow processing big datasets.

The initial regions in which different parameters for λ-`DBSCAN` might be needed could also be automatically be identified by running k-means with a sufficiently large k and use the density around the so-found proto-clusters to initialize the function $\varepsilon(\boldsymbol{p})$.

Source Code The implementation of our algorithm is based on scikit-learn's [11] implmention of DBSCAN. Our changes are available at

`https://visual2.cs.ovgu.de/pubres/lambda-dbscan`.

References

1. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: Optics: Ordering Points to Identify the Clustering Structure. In: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data. p. 4960. SIGMOD '99, Association for Computing Machinery, New York, NY, USA (1999)
2. Campello, R.J.G.B., Moulavi, D., Sander, J.: Density-Based Clustering Based on Hierarchical Density Estimates. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds.) Advances in Knowledge Discovery and Data Mining. pp. 160–172. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
3. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. p. 226231. KDD'96, AAAI Press, Portland, Oregon (1996)
4. Fränti, P., Sieranoja, S.: K-Means Properties on Six Clustering Benchmark Datasets. Applied Intelligence **48**(12), 4743–4759 (2018), `http://cs.uef.fi/sipu/datasets/`
5. Günther, T., Gross, M., Theisel, H.: Generic Objective Vortices for Flow Visualization. ACM Transactions on Graphics (Proc. SIGGRAPH) **36**(4), 141:1–141:11 (2017)
6. MacQueen, J., et al.: Some Methods for Classification and Analysis of Multivariate Observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. vol. 1, pp. 281–297. Oakland, CA, USA (1967)
7. Malzer, C., Baum, M.: A Hybrid Approach to Hierarchical Density-Based Cluster Selection. In: 2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI). pp. 223–228 (2020)
8. Mutlu, O., Salman, H.E., Yalcin, H.C., Olcay, A.B.: Fluid Flow Characteristics of Healthy and Calcified Aortic Valves Using Three-Dimensional Lagrangian Coherent Structures Analysis. Fluids **6**(6), 203 (2021)
9. Ng, A., Jordan, M., Weiss, Y.: On Spectral Clustering: Analysis and an Algorithm. In: Dietterich, T., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Information Processing Systems. vol. 14. MIT Press (2001),
10. Nolan, P.J., Foroutan, H., Ross, S.D.: Pollution Transport Patterns Obtained Through Generalized Lagrangian Coherent Structures. Atmosphere **11**(2), 168 (2020)
11. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
12. Popinet, S.: Free Computational Fluid Dynamics. ClusterWorld **2**(6) (2004)
13. Ramos, A., García-Garrido, V., Mancho, A., Wiggins, S., Coca, J., Glenn, S., Schofield, O., Kohut, J., Aragon, D., Kerfoot, J., et al.: Lagrangian Coherent Structure Assisted Path Planning for Transoceanic Autonomous Underwater Vehicle Missions. Scientific reports **8**(1), 4575 (2018)
14. Ruiz, C., Spiliopoulou, M., Menasalvas, E.: C-DBSCAN: Density-Based Clustering with Constraints. Rough Sets, Fuzzy Sets, Data Mining and Granular Computing. LNCS(4482), 2007
15. Schubert, E., Sander, J., Ester, M., Kriegel, H.P., Xu, X.: DBSCAN Revisited: Why and How You Should (Still) Use DBSCAN. ACM Trans. Database Syst. **42**(3) (July 2017)
16. Stelter, D., Wilde, T., Rössl, C., Theisel, H.: A Particle-Based Approach to Extract Dynamic 3D FTLE Ridge Geometry (under review)