Pattern Recognition 137 (2023) 109314

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Feature weighting in DBSCAN using reverse nearest neighbours

Stiphen Chowdhury[a], Na Helian[b], Renato Cordeiro de Amorim[c,*]

[a] School of Computing and Information Science, Anglia Ruskin University, East Road, Cambridge CB1 1PT, UK
[b] Department of Computer Science, University of Hertfordshire, Hatfield AL10 9AB, UK
[c] School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe CO4 3SQ, UK

ARTICLE INFO

Article history:
Received 11 May 2021
Revised 20 December 2022
Accepted 8 January 2023
Available online 12 January 2023

Keywords:
Density-based clustering
Reverse nearest neighbour
DBSCAN

ABSTRACT

DBSCAN is arguably the most popular density-based clustering algorithm, and it is capable of recovering non-spherical clusters. One of its main weaknesses is that it treats all features equally. In this paper, we propose a density-based clustering algorithm capable of calculating feature weights representing the degree of relevance of each feature, which takes the density structure of the data into account. First, we improve DBSCAN and introduce a new algorithm called DBSCANR. DBSCANR reduces the number of parameters of DBSCAN to one. Then, a new step is introduced to the clustering process of DBSCANR to iteratively update feature weights based on the current partition of data. The feature weights produced by the weighted version of the new clustering algorithm, W-DBSCANR, measure the relevance of variables in a clustering and can be used in feature selection in data mining applications where large and complex real-world data are often involved. Experimental results on both artificial and real-world data have shown that the new algorithms outperformed various DBSCAN type algorithms in recovering clusters in data.

1. Introduction

The digital universe is growing in size every year. This wealth of data being generated are usually stored in digital media, hence offering huge potential for its automatic mining. Raw data by itself are unlikely to be useful, and given its size they are very expensive to label. Clustering algorithms follow the unsupervised learning framework, which does not require labeled data to learn from. Given a data set Y containing n points, a clustering algorithm will produce a set of clusters so that the points assigned to a given cluster are similar according to some measure. These algorithms have been applied as a dominant data analysis tool in diverse fields including medicine, marketing, bioinformatics, image processing, computer security, geography, physics, and astronomy (see for instance [1], and references therein).

There are indeed different approaches clustering algorithms may employ. Arguably, the most popular approaches are partitional, hierarchical, and density-based. Algorithms following the partitional approach produce a set of K disjoint clusters $S = \{S_1, S_2, \ldots, S_K\}$ whose sum of cardinalities equals the cardinality of the data set itself — That is, $\left|\bigcup_{S_l \in S} S_l\right| = n$. Hierarchical algorithms go a step further by also producing information about the relationships between clusters, usually at a high computational cost. Density-based algorithms define clusters as areas of higher density, which allows clusters with arbitrary shapes. In this paper we focus on density-based algorithms. We direct readers interested in other approaches to the many sources in the literature (see for instance [2,3], and references therein).

DBSCAN [4] is a classic example of density-based algorithm, which remains very relevant [5]. It recovers clusters in a two-step approach: (i) it identifies the *core* points, that is, a set of high-density points; (ii) it forms clusters from these *core* points by grouping reachable points. Reachability is defined in such a way that no two points of different high-density regions, separated by a contiguous low density region, are reachable from each other. This definition makes DBSCAN intuitive and rather popular among the density-based clustering algorithms.

Unfortunately, DBSCAN does have shortcomings. Among these we have: (i) it requires two parameters with no obvious method to determine their optimum values. In fact, this algorithm is rather sensitive to these parameters; (ii) it is not particularly suitable for data sets containing clusters with widely different densities; (iii) it treats all features equally regardless of their contribution to the clustering. Various algorithms have attempted to address the shortcomings (i) and (ii), for details see Section 2.

We find (i) and (ii) to be important, but (iii) is particularly interesting. This is an issue because in real-world data sets it is un-

* Corresponding author.
E-mail addresses: stiphen.chowdhury@aru.ac.uk (S. Chowdhury), n.helian@herts.ac.uk (N. Helian), r.amorim@essex.ac.uk (R. Cordeiro de Amorim).

https://doi.org/10.1016/j.patcog.2023.109314

likely that all features will be relevant. In fact, even among relevant features there may be different degrees of relevance. Hence, density-based clustering algorithms may benefit from taking into account the relevance of each feature during the clustering procedure. This princip has been successfully applied to partitional algorithms and hierarchical algorithms (see for instance [6–8], and references therein), but no such large research effort has been done in density-based clustering algorithms.

The main contribution of this paper is two-fold. We first introduce DBSCANR, a density-based clustering algorithm that uses reverse nearest neighbour to address shortcomings (i) and (ii). Second, we extend this work by introducing automatic feature weights. These feature weights are used to model the degree of relevance of each feature, and can be seen as a generalisation of feature selection. The latter either selects or deselects a particular feature. Feature weights assign a degree of relevance to each feature, a factor between zero and one.

2. Related work

Unfortunately, there is no precise widely accepted definition for the term *cluster*. A loose definition often employed is that a cluster is a compact set of similar points. Clearly, clusters may have different cardinalities, shapes and densities. Density-based clustering algorithms aim at discovering high-density regions that are separated from each other by contiguous regions of lower density [9]. It is intuitive to assign the term cluster to such high-density areas. These algorithms rely heavily on a density estimation function, but they do not usually make assumptions regarding the number of clusters in a data set, or the data distribution. This can lead to the identification of arbitrarily shaped clusters. In this section we describe some of the key algorithms related to our research, giving emphasis to those we experimentally compare with.

DBSCAN is often considered the most popular density-based clustering algorithm. As such, it can detect clusters of different cardinalities and shapes. Given a data set Y containing n points y_i, each described over V features, DBSCAN begins by assigning each $y_i \in Y$ into one of three categories: (i) core; (ii) (directly) reachable; (iii) outlier. A core point is a $y_i \in Y$ with at least *minPts* points within a distance of ϵ. In other words, let

$$d(y_i, y_j) = \sum_{v=1}^{V} (y_{iv} - y_{jv})^2, \tag{1}$$

and

$$N_\epsilon(y_i) = \{y_j \in Y : d(y_i, y_j) \leq \epsilon\}. \tag{2}$$

The point $y_i \in Y$ is a core point iff $|N_\epsilon(y_i)| \geq minPts$, where *minPts* is a user-defined threshold. A point y_j is said to be directly reachable from y_i iff y_i is a core point and $y_j \in N_\epsilon(y_i)$. A point y_j is reachable from y_i if there is a path of points $y_i, \ldots, y_j$ where each point is directly reachable from the previous. Outliers are points that are unreachable from any other point in the data set. DBSCAN produces a clustering using the definitions above and following three simple steps: (i) for each $y_i \in Y$, compute $N_\epsilon(y_i)$ and identify the set of core points; (ii) for each core point, identify all reachable and directly reachable points; (iii) assign each non-core point (excluding outliers) to its connected cluster.

DBSCAN produces a clustering based on three things: ϵ, *minPts*, and the distance function in use. Under usual conditions ϵ is inversely proportional to the number of clusters. A high ϵ leads to larger neighbourhoods and by consequence a lower number of clusters, while a low ϵ has the opposite effect. It is often stated that density-based clustering algorithms are capable of recovering clusters of arbitrary shapes. This is a very tempting thought, which may lead to some disregarding the importance of selecting an appropriate distance or similarity measure. This measure is the key to produce homogeneous clusters as it defines homogeneity, so it has an impact on the actual clustering. Most likely the impact will not be as obvious as if one were to apply an algorithm such as k-means [10] (where the distance in use leads to a clear bias towards a particular cluster shape, which is something that can also be exploited [11]). However, the impact of this selection will still exist at a more local level. If this was not the case, DBSCAN would produce the same clustering regardless of the distance measure in place.

OPTICS [12] still requires two parameters, *minPts* and ϵ, but it manages to address DBSCAN's inability to deal with clusters of different densities. It does so by taking into consideration the distance between core points and the *minPts*th nearest point when calculating the reachability distances. This essentially allows OPTICS to identify clusters in data of varying density. One should note that a higher ϵ incurs more computational cost [13].

ISDBSCAN [14] has pioneered the use of reverse nearest neighbour (RNN) [15] in DBSCAN-based algorithms. Let us first make some important definitions. The nearest neighbour of $y_i \in Y$ is the point $y_j \in Y$ with the lowest distance to y_i, with $y_i \neq y_j$. With this, we can now define the k-neighbourhood of y_i as the set $NN_k(y_i)$ containing the k-nearest points to y_i, with $y_i \notin NN_k(y_i)$. We can now make an important definition we will use later on.

Definition 1. The reverse k-neighbourhood of a point $y_i \in Y$ is given by

$$RNN_k(y_i) = \{y_j \in Y : y_i \in NN_k(y_j)\}. \tag{3}$$

ISDBCAN calculates the k-influence space $IS_k(y_i) = NN_k(y_i) \cap RNN_k(y_i)$. Note that $NN_k(y_i) \neq \emptyset$ but there is no such guarantee for $RNN_k(y_i)$. However, this RNN-based approach allows the algorithm to capture local densities in different regions of the data space, leading to the recovery of clusters having heterogeneous densities. In addition, ISDBSCAN attempts to lower the difficulty of using DBSCAN by removing one of its parameters, ϵ, leaving only k (the number of nearest neighbours) as a parameter.

ISDBSCAN performs the clustering task in two-steps. First, it attempts to identify all outliers in a given data set. It does so by calculating the k-influenced outlierness of a point y_i given by

$$INFLO_k(y_i) = \sum_{y_j \in IS_k(y_i)} \frac{den_k(y_j)}{|IS_k(y_i)| den_k(y_i)}, \tag{4}$$

where $den_k(y_i) = \frac{1}{d(y_i, y_t)}$ and y_t is the kth-neighbour of y_i. Second, the clustering algorithm is applied to the residual data set. This algorithm builds a cluster based on the density of y_i if $|IS_k(y_i)| \geq 2/3k$. This was the best threshold identified by its authors. Of course, it is fair to assume that a different threshold may be found in experiments on different data sets. Hence, one may even argue that this threshold is in fact a parameter with no clear method to identify its optimal value. Such thought leads ISDBSCAN to have the same number of parameters as DBSCAN.

RNN-DBSCAN [16] aims at reducing the number of parameters of DBSCAN by adapting ISDBSCAN's RNN_k-based density estimation. Thus, RNN-DBSCAN is able to recover clusters with different degrees of density by setting a single parameter, k. Unlike ISDBSCAN, the density of a point is determined by a special combination of nearest neighbourhood and reverse nearest neighbourhood instead of the influence space. Given $y_i, y_j \in Y$ there are three scenarios for connectivity.

1. The point y_j is *directly density-reachable* from y_i if $y_j \in NN_k(y_i)$ and $|RNN_k(y_i)| \geq k$.
2. A point y_j is *density-reachable* from y_i if there exists a sequence of points $C = (y_i, \ldots, y_j)$, such that each of these points is directly density-reachable from the previous, and $|RNN_k(y_t)| > k$.

3. The point y_j is *density-connected* to point y_i, if there is a point $y_t \in Y$ such that both y_i and y_j are density-reachable from y_t.

Using the above, RNN-DBSCAN defines a cluster using a simple definition: any two points $y_i, y_j \in Y$ belong to the same cluster if they are density-reachable or density-connected. RNN-DBSCAN indeed requires a single parameter to tackle the problem of variable density clusters, however, it does so considering all the features to be equally relevant.

Adaptive DBSCAN (ADBSCAN) [17] is a recent advancement in density-based clustering that requires two parameters, k and *noise_percent* (the prior estimate of the noise ratio of the data set). This algorithm automatically discovers the number of clusters by initially building a nearest neighbour graph, and eventually dividing the data set into subgraphs. In the latter, two vertices are considered to be *subgraph core* points if they are the nearest neighbour to each other. The density of a point $y_i \in Y$ is give by

$$\rho(y_i) = \log\left[\frac{\sum_{j=1}^{k} d(y_i, y_i^j)}{k}\right], \tag{5}$$

where y_i^k is the kth nearest neighbour of y_i. The original authors then specify a criteria for a subgraph to be a core subgraph based on the existence of a subgraph core point, the value of *noise_percent*, the average, quartile, and standard deviation obtained with (5). ADBSCAN is indeed an enhancement of DBSCAN as it seems to act well on data sets with large density variations. However, it introduces a new parameter to do so.

Density Peak Clustering (DPC) [18] has recently gained popularity [19–22], due to its effectiveness and intuitive distance threshold parameter (with a suggested standard value). The general idea behind DPC is that cluster centres are high-density points that are surrounded by lower-density neighbours, and that these centres have a high relative distance to other points of higher density. DPC identifies K clusters and automatically assign points to them. The local density of point $y_i \in Y$ is the number of neighbours adjacent to y_i within a user-defined cutoff distance d_c.

As popular as it may be, DPC is not without weaknesses. Hence, it has been a target of numerous extensions. DPC-DBFN [23] improves clustering recovery by calculating local densities using a fuzzy kernel rather than a crisp kernel. Once the cluster centre is identified, before the label assignment, a new step is introduced to the DPC clustering process to form the high-density regions called cluster backbones. This is constructed by labelling a data point as a dense point, border point or noisy point. A point y_i is a dense point if its density is equal to or higher than the average density over all points in the data set. Otherwise, y_i is either a border point or a noise point depending on the variance of the distance between points. DPC-DBFN improves DPC to find clusters with various densities, shapes, and sizes, however, it introduces a new controlling parameter to distinguish border points from noise points.

The above clustering methods enhance DBSCAN, however, they treat all the features equally regardless of their degree of relevance, which can have a detrimental impact on the clustering. One could argue that nowadays the most interesting data sets are high-dimensional. In this type of data meaningful clusters often appear to be discovered in a particular subset of features rather than on all the available features [24,25]. A common solution is to apply a feature selection algorithm before the clustering. However, this introduces two issues: (i) it assumes that all clusters have the same relevant features; (ii) it assumes that all selected features are equally relevant. Both issues go considerably against intuition. In real-world data sets, it is perfectly possible to have a set of relevant features in which the relevance of each of them differs.

3. DBSCANR and W-DBSCANR

3.1. DBSCANR

In this section we introduce our density-based clustering algorithm, DBSCANR. Very much like DBSCAN (for details, see Section 2), DBSCANR needs to determine whether a point $y_i \in Y$ is *core* or *directly reachable*. In the case of DBSCANR this is determined using reverse nearest neighbour, $RNN_k(y_i)$ (see Definition 1).

Definition 2. A point $y_i \in Y$ is said to be a *core* point if

$$|RNN_k(y_i)| \geq k. \tag{6}$$

Notice that we use the quantity of reverse nearest neighbours as the density of a point. So, the density of $y_j \in Y$ is higher than that of $y_i \in Y$ if $|RNN_k(y_j)| > |RNN_k(y_i)|$. We can now make another important definition for DBSCANR.

Definition 3. A point $y_j \in Y$ is said to be directly-reachable from a point $y_i \in Y$, with $y_i \neq y_j$ if

1. y_i is a *core* point,
2. $y_j \in RNN_k(y_i)$.

The key idea of our method is that, each point in a cluster has to comprise of at least a given minimum number of points (k) in its reverse nearest neighbour. This way reverse nearest neighbourhood estimates density of a point by discarding those that do not consider the query point as their nearest neighbour. We find the above definitions of *core* and *directly reachable* entities to be more robust than those used by DBSCAN, and our experiments support this statement.

Given the basic definitions above, we can now go further and introduce other new key definitions for our method.

Definition 4. A point $y_j \in Y$ is *density-reachable* from $y_i \in Y$ with respect to k, if there exists a sequence of points $C = (y_i, \ldots, y_j)$, such that each element is directly-reachable from the previous.

Density reachability is the transitive closure of direct reachability. Any point other than *core* can not be mutually density and direct reachable, leading to the asymmetry illustrated in Fig. 1c. This figure shows the more interesting asymmetric case of this definition in a $2D$ vector space, which measures distance using (1). Within cluster S_c, two *core* entities are density-reachable from each other. The same can not be said for the entities that are not *core*. The following definition relates those *non-core* entities to the *core* entities they are density-reachable from.

Definition 5. A point $y_j \in Y$ is *density-connected* to $y_i \in Y$ with respect to k, if both y_i and y_j are density-reachable from $y_t \in Y$.

Density-connectivity is a symmetric relation (see Fig. 1b). Similar to the approach taken by DBSCAN, a DBSCANR cluster is a set of density-connected points holding the maximality with respect to density-reachability.

Definition 6. The purpose of any clustering algorithms is to split a data set Y containing n entities $y_i \in \mathbb{R}^V$ into K clusters $S = \{S_1, S_2, \ldots, S_K\}$. Here, we are particularly interested in hard-clustering so that a given point y_i can be assigned to a single cluster $S_c \in S$, and $\sum_{l=1}^{K} |S_l| = n$. Our final clustering satisfies the following conditions:

1. $\forall y_i, y_j \in Y$: if $y_i \in S_c$ and y_j is density-reachable from y_i wrt. k, then $y_j \in S_c$. (Maximality)
2. $\forall y_i, y_j \in S_c$, y_i is density-connected to y_j wrt. k. (Connectivity)

Given k, we can recover a cluster in a two-step process. First, select the *core* point from the data set with the highest density

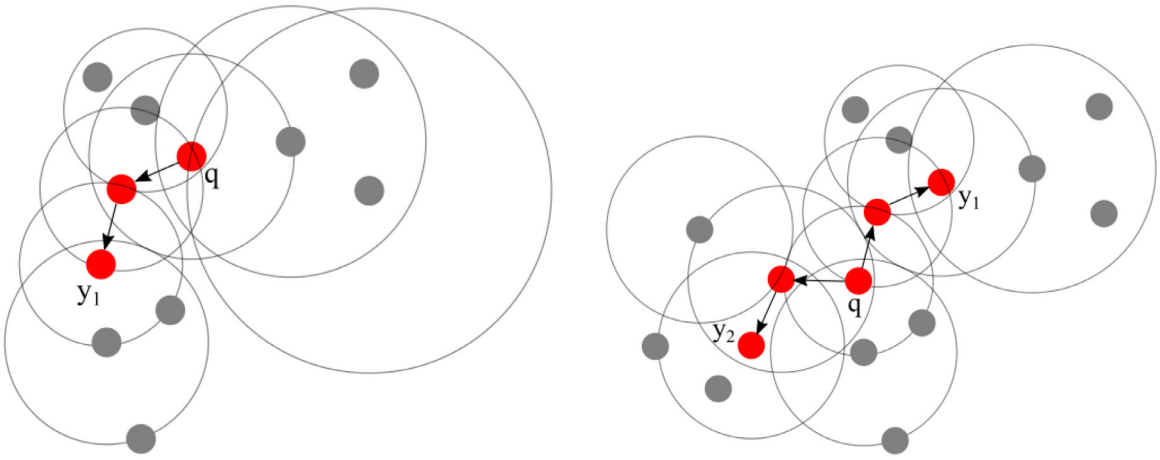


Fig. 1. Reverse nearest neighbour based density-reachability and density-connectivity. (a) y_1 is density-reachable from q; q is not density-reachable from y_1. (b) y_1 and y_2 are density-connected by q.

(see Definition 2), using it to retrieve all related density-reachable points. Second, assign the latter to the cluster of the *core* point.

DBSCANR starts with the highest density core point $y_i \in Y$ from a sequence of core points C. If there is more than one point with the same highest density, then one of them is selected uniformly at random. Afterwards, DBSCANR retrieves all points that are density-reachable from y_i wrt. k. This method, iteratively, recovers all clusters comprising the *core* points. Finally, each point that does not satisfy the condition for *core* point (see Definition 2) will be assigned to the cluster of its nearest *core* point. Although we use only global values for k, DBSCANR recovers clusters of different densities and shapes simultaneously (see Definition 6).

Algorithm 1 : DBSCANR (Y, k).

Input
Y: Data set.
k: Minimum number of points.
Output
S : A clustering $S = \{S_1, S_2, \cdots, S_K\}$

1: Set $S \leftarrow \emptyset$ and $C \leftarrow \emptyset$.
2: Add each point in Y to C (as per definition 2).
3: Identify the point $q \in C$ with the highest density, and remove q from C.
4: S_c = RecoverCluster (C, q, k)
5: If $|S_c| \geq k$
Add S_c to S
6: Remove each point in S_c from C
Repeat steps 3 to 6 until $|C|$ has converged.
7: Assign each unclustered point to the cluster of its nearest *core* point.

We can formalise the whole algorithm as follows.

In the above, the quantity of nearest neighbours, k, is a user-defined parameter. The quantity of clusters, K, is automatically found by the algorithm.

There are reasons why using reverse nearest neighbours makes our algorithm superior to others. Notice that the k-nearest neighbourhood of a point usually contains k points. However, with reverse nearest neighbour no such guarantee exists as local densities are taken into account. This is particularly helpful when attempting to identify clusters with very different local densities. For instance, Fig. 2 illustrates the neighbourhood using coloured circles.

Algorithm 2 : RecoverCluster (C, q, k).

Input
C: *Core* point vector.
q: A point of high density.
k: Minimum number of points.
Output
S_c : A cluster

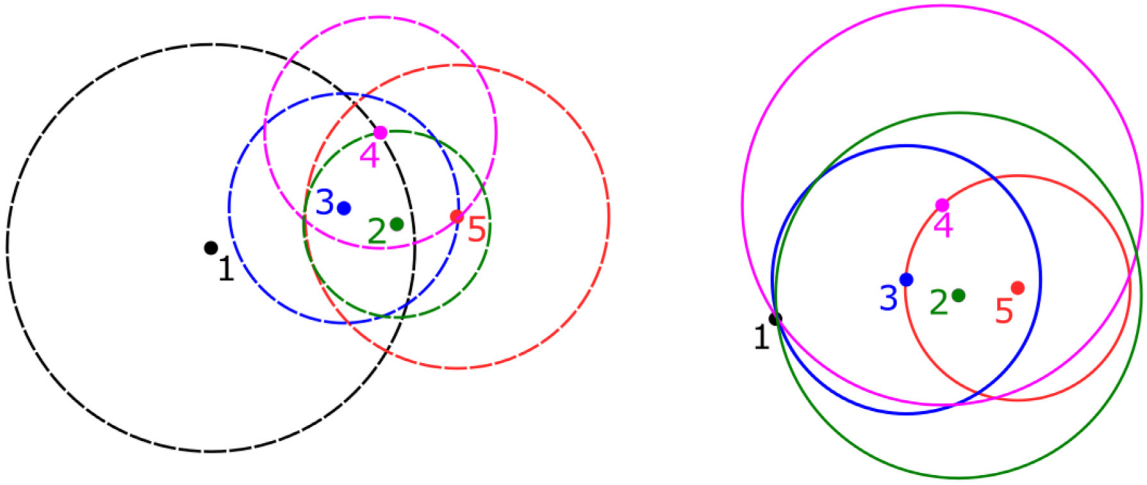
1: Set *seeds* $\leftarrow \emptyset$ and $S_c \leftarrow \emptyset$.
2: For each $y_i \in C$
If $y_i \in RNN_k(q)$ and y_i has never been assigned to S_c
Add y_i to *seeds*.
3: Add q to S_c, and remove q from *seeds* (if $q \in$ *seeds*).
4: For each $y_i \in$ *seeds*
If y_i has never been assigned to S_c
Set $q \leftarrow y_i$.
Repeat steps 2 to 4 until $|seeds| = 0$.
5: For each $y_i \in S_c$
Add to S_c all points in $NN_k(y_i)$ that are not in C and have never been assigned to S_c.

Algorithm 3 : W-DBSCANR (Y, k, β).

Input
Y: Data set.
k: Minimum number of points.
β: Weight exponent.
Output
S : A clustering $S = \{S_1, S_2, \ldots, S_K\}$
W : A set of weight vectors $W = \{w_1, w_2, \ldots, w_K\}$

1: Set K to be the number of clusters in the clustering produced by Algorithm 1.
2: Set $w_{cv} \leftarrow \frac{1}{V}$, for $c = 1, 2, \ldots, K$ and $v = 1, 2, \ldots, V$.
3: S = UpdateClustering (Y, k, S, W).
4: Update feature weights for each cluster (as per Equation 17).
5: Repeat steps 3 and 4 until $|S|$ has converged.

In Fig. 2(a), the black circle of point 1 along with other coloured circles of point 2, 3, 4, and 5 corresponds to the k-nearest neighbourhood at $k = 3$. $NN_k(1) = \{2, 3, 4\}$, $NN_k(2) = \{3, 4, 5\}$, $NN_k(3) = \{2, 4, 5\}$, $NN_k(4) = \{2, 3, 5\}$ and $NN_k(5) = \{2, 3, 4\}$.



(a) (b)

Fig. 2. The neighbourhood are shown in different colours (a) k-nearest neighbourhood at $k = 3$ (b) reverse k-nearest neighbourhood at $k = 3$.

Algorithm 4 : UpdateClustering (Y, k, S, W).

Input
Y: Data set.
k: Minimum number of points.
S: A clustering.
W: A set of weight vectors.
Output
S : A clustering $S = \{S_1, S_2, \cdots, S_K\}$

1: Set $C \leftarrow \emptyset$.
2: Add each weighted *core* point in Y to C (as per definition 7).
3: Identify the point $q \in C$ with the highest density, and remove q from C.
4: S_c = RecoverCluster (C, q, k, W)
5: If $|S_c| \geq k$
Add S_c to S
6: Remove each point in S_c from C
Repeat steps 3 to 6 until $|C|$ has converged.
7: Assign each unclustered point to the cluster of its nearest weighted *core* point.

Algorithm 5 : RecoverCluster (C, q, k, W).

Input
C: Weighted *core* point vector.
q: Weighted *core* point with the highest density.
k: Minimum number of points.
W: A set of weight vectors $W = \{w_1, w_2, \ldots, w_K\}$.
Output
S_c : A weighted cluster

1: Set *seeds* $\leftarrow \emptyset$ and $S_c \leftarrow \emptyset$.
2: For each $y_i \in C$
If $y_i \in RNN_k^W(q)$ and y_i has never been assigned to S_c
Add y_i to *seeds*.
3: Add q to S_c, and remove q from *seeds* (if $q \in$ *seeds*).
4: Identify $y_i \in$ *seeds*, such that y_i has not been assigned to any cluster. Set $q \leftarrow y_i$. Repeat steps 2 to 4 until $|seeds| = 0$.
5: For each $y_i \in S_c$
Add to S_c all points in $NN_k^W(y_i)$ that are not in C and have not been assigned to a cluster.

In Fig. 2(b), the coloured circles represent the reverse k-nearest neighbourhoods of each point at $k = 3$. $RNN_k(1) = \emptyset$, $RNN_k(2) = \{1, 3, 4, 5\}$, $RNN_k(3) = \{1, 2, 4, 5\}$, $RNN_k(4) = \{1, 2, 3, 5\}$ and $RNN_k(5) = \{2, 3, 4\}$. Since point 2, 3, 4, and 5 do not have point 1 as their neighbour, the reverse nearest neighbour set of point 1 is empty, hence no circle around point 1 in Fig. 2(b). It is interesting to note that the empty reverse nearest neighbour set of point 1 can be associated with the separation of two widely variable clusters. This indicates that the reverse nearest neighbour can be used to identify the border point of a widely variable density cluster such as point 1, without the need of any special combination, while still maintaining the competitiveness in clustering recovery when compared with DBSCAN and its state-of-the-art counterparts. Hence, our algorithm can find naturally meaningful clusters rather than clusters that fit a certain static neighbourhood query. Unlike ISDBSCAN and RNN-DBSCAN, we used only RNN for our neighbourhood calculation rather than any special combination of the nearest neighbourhood and its reverse counterpart.

When two clusters of widely variable densities are separated by very narrow sparse regions, recovering cluster borders may become difficult. To address this issue only core points are clustered in the initial clustering recovery step of our algorithm. We take the view that cluster borders are surrounded by non-core or border points, in the final clustering recovery step we assigned the border points to its nearest core neighbour cluster. Within the same cluster if the density varies, this cluster extension strategy includes all the points rather than assigning the non-core points as outliers just because it does not meet the clustering definition based on a special neighbourhood search condition.

DBSCANR requires a single user-defined parameter, and it is able to recover clusters of different densities. However, very much like its competitors DBSCANR still treats all features equally.

3.2. Weighted DBSCANR (W-DBSCANR)

In most pattern recognition tasks different features may have different degrees of relevance, and this certainly applies to clustering. Even if we assume that all features in a given data set are relevant, there may be different degrees of relevance. Given a cluster $S_l \in S$, one can set the weight of a feature v to be inversely proportional to the dispersion of v within S_l [26]. In other words, features that are more compact within a cluster are more discrim-

Table 1
The synthetic data sets we experiment with.

			No noise features		+ 50% noise features		+ 100% noise features	
Data set	Entities n	Clusters K	Features V	Noise Features	Features V	Noise Features	Features V	Noise Features
Aggregation	788	7	2	0	3	1	4	2
Grid	655	2	2	0	3	1	4	2
D31	3100	31	2	0	3	1	4	2
Flame	240	2	2	0	3	1	4	2
Mixed	1479	5	2	0	3	1	4	2
Pathbased	300	3	2	0	3	1	4	2
R15	600	15	2	0	3	1	4	2
Spiral	312	3	2	0	3	1	4	2
Toy	373	2	2	0	3	1	4	2
Diamonds	800	2	2	0	3	1	4	2

Table 2
The real-world data sets we experiment with.

Data set	Entities n	Clusters K	Features V
Banknote	1372	2	4
Iris	150	3	4
Ecoli	336	8	7
Seeds	210	3	7
BreastC.	699	2	9
BreastT.	106	6	9
Liver	583	2	9
Wine	178	3	13
Leaf	340	30	14
Zoo	101	7	16
Parkinsons	195	2	22
Leuk	72	3	39
TeachingA.	151	3	56
Soya	47	4	58
Libras	360	15	90
ALLAML	72	2	7129
Carcinom	174	11	9182
CLL-SUB	111	3	11340
Colon	62	2	2000
GLIOMA	50	4	4434
Lung	203	5	3312
Lymphoma	96	9	4026
ORL	400	40	1024
Prostate-GE	102	2	5966
Tox-171	171	4	5748

Table 3
The real-world data sets with added noise we experiment with.

			+ 50% noise features		+ 100% noise features	
Data sets	Entities n	Clusters K	Features V	Noise Features	Features V	Noise Features
Banknote	1372	2	6	2	8	4
Iris	150	3	6	2	8	4
Ecoli	336	8	11	4	14	7
Seeds	210	3	11	4	14	7
BreastC.	699	2	14	5	18	9
BreastT.	106	6	14	5	18	9
Liver	583	2	14	5	18	9
Wine	178	3	20	7	26	13
Leaf	340	30	21	7	28	14
Zoo	101	7	24	8	32	16
Parkinsons	195	2	33	11	44	22
Leuk	72	3	59	20	78	39
TeachingA.	151	3	84	28	112	56
Soya	47	4	87	29	116	58
Libras	360	15	135	45	180	90

inatory than those that are less compact. We considerably expand the above in order to introduce, perhaps for the first time, feature weighting to a density-based clustering algorithm. Given $y_i, y_j \in S_l$ we can calculate their distance using

$$d^W(y_i, y_j) = \sum_{v=1}^{V} w_{lv}^{\beta}(y_{iv} - y_{jv})^2, \tag{7}$$

where β is a user-defined parameter, and w_{lv} is the weight of feature v at cluster S_l. Clearly, the balanced use of (7) for density estimation requires each weight to be non-negative and $\sum_{v=1}^{V} w_{lv} = 1$ for each cluster $S_l \in S$. Hence, the weighted $k-$neighbourhood of y_i is the set $NN_k^W(y_i)$ containing the $k-$nearest points to y_i, calculated using (7), with $y_i \notin NN_k^W$. Notice that in this case the l in (7) represents the cluster y_i belongs to. The above allow us to revisit our definition of reverse k-neighbourhood (RNN_k), and present its weighted version.

$$RNN_k^W(y_i) = \{y_j \in Y : y_i \in NN_k^W(y_j)\}. \tag{8}$$

Now, we are ready to make some important definitions for our algorithm.

Definition 7. A point y_j is weighted directly density-reachable from a point y_i with respect to k and β, if

1. $y_j \in RNN_k^W(y_i)$
2. $|RNN_k^W(y_i)| \geqslant k$ (i.e. y_i is a weighted *core point*)

Definition 8. A point y_j is weighted density-reachable from a point y_i, if there exists a sequence of points $C^W = (y_i, \ldots, y_j)$, such that each element is weighted directly density-reachable from the previous.

Definition 9. A point y_j is weighted density-connected to a point y_i, if both y_i and y_j are weighted density-reachable from a common point y_t.

Weighted density-connectivity is a symmetric relation. We now introduce the notion of weighted density-based cluster. Similar to DBSCAN, a weighted density-based cluster can now be defined as a set of weighted density-connected points which hold maximality with respect to weighted density-reachability.

Definition 10. Weighted clusters are a partition of a data set Y containing n entities $y_i \in \mathbb{R}^V$ into K non-empty disjoint clusters $S = \{S_1, S_2, \ldots, S_K\}$. Here, we are particularly interested in hard-clustering so that a given point y_i can be assigned to a single cluster $S_c \in S$. Thus, the final clustering is a maximal set of weighted density-connected entities subject to $S_k \cap S_l = \emptyset$ for $k, l = 1, 2, \ldots, K$ and $k \neq l$ satisfying the following conditions :

1. $\forall y_i, y_j \in Y$: if $y_i \in S_c$ and y_j is weighted density-reachable from y_i with respect to k and β, then $y_j \in S_c$. (Maximality)
2. $\forall y_i, y_j \in S_c$: y_i is weighted density-connected to y_j with respect to k and β. (Connectivity)

Our proposed clustering algorithm recovers weighted cluster in two steps. First, it identifies the weighted *core* points with the

Table 4
Results of the experiments on the original synthetic data sets (no noise features have been added). We measure cluster recovery using the Adjusted Rand index (ARI), F-Measure (FM), Normalised Mutual Information (NMI) and Accuracy (Acc).

Data set	CVI	DBSCAN (k/ϵ)	OPTICS (k/m_{cr})	ISDBSCAN (k)	RNN-DBSCAN (k)	ADBSCAN (k/p)	DPC-DBFN (k)	DBSCANR (k)	W-DBSCANR (k/β)
Aggregation	ARI	0.9834 (10/0.06)	0.9082 (7/0.14)	0.9911 (12)	0.9966 (13)	0.9065 (29/0.1)	0.9927 (30)	0.9978 (12)	**1 (17/1.4)**
	FM	0.9915 (11/0.06)	0.7016 (7/0.15)	0.9947 (12)	0.9982 (13)	0.0784 (29/0.1)	0.9957 (24)	0.9988 (12)	**1 (17/2.2)**
	NMI	**1 (18/0.06)**	0.9381 (7/0.15)	0.9958 (9)	0.9958 (8)	0.9435 (29/0.1)	0.9884 (33)	0.9957 (12)	**1 (17/2.2)**
	Acc	**1 (16/0.06)**	0.9489 (7/0.15)	0.9987 (9)	0.9987 (13)	0.9374 (29/0.1)	0.9962 (24)	0.9987 (12)	**1 (17/2.2)**
D31	ARI	0.8224 (42/0.04)	0.4649 (27/0.09)	-	0.8591 (35)	-	**0.9541 (83)**	0.9354 (35)	0.9445 (33/1.8)
	FM	0.9445 (42/0.04)	0.0859 (11/0.14)	-	0.948 (35)	-	**0.9774 (83)**	0.9681 (35)	0.9735 (15/3.2)
	NMI	**0.9917 (27/0.03)**	0.3686 (11/0.14)	-	0.9656 (35)	-	0.9678 (83)	0.9574 (35)	0.964 (16/3.4)
	Acc	**0.9943 (27/0.03)**	0.1246 (11/0.14)	-	0.9741 (35)	-	0.9774 (83)	0.9681 (35)	0.9735 (15/3.2)
Flame	ARI	0.9715 (14/0.12)	0.8969 (7/0.08)	0.9497 (8)	**0.9881 (8)**	0.0128 (41/0)	0.9666 (6)	0.9833 (8)	0.9833 (7/2.4)
	FM	0.9896 (14/0.12)	0.9601 (7/0.08)	0.9774 (8)	0.9942 (8)	0.4138 (41/0)	0.991 (6)	**0.9955 (8)**	**0.9955 (7/1.5)**
	NMI	**1 (3/0.06)**	**1 (20/0.04)**	0.9621 (8)	**1 (8)**	0.0437 (53/0.5)	0.9355 (27)	0.9635 (8)	**1 (2/1.1)**
	Acc	**1 (3/0.06)**	**1 (15/0.05)**	0.9957 (8)	**1 (8)**	**1 (41/0.05)**	0.9917 (6)	0.9958 (8)	**1 (2/1.1)**
Grid	ARI	0.6377 (4/0.07)	0.8585 (5/0.07)	0.9397 (7)	0.9397 (7)	0.1031 (197/0.45)	0.5207 (299)	0.9457 (4)	**1 (4/1.1)**
	FM	0.6724 (5/0.07)	0.64 (30/0.02)	0.9843 (7)	-	0.2132 (197/0.45)	0.8609 (299)	**0.9859 (4)**	**0.9859 (4/3.3)**
	NMI	0.8829 (7/0.07)	0.4288 (21/0.02)	0.8995 (7)	-	0.1185 (179/0.45)	0.4832 (299)	**0.9075 (4)**	**0.9075 (4/3.3)**
	Acc	0.8958 (7/0.07)	0.9398 (5/0.07)	0.9847 (7)	-	0.4225 (77/0.5)	0.8611 (299)	**0.9863 (4)**	**0.9863 (4/3.3)**
Mixed	ARI	**1 (2/0.05)**	0.9998 (5/0.3)	**1 (23)**	0.9989 (36)	**1 (43/0)**	0.6125 (29)	**1 (14)**	**1 (14/1.9)**
	FM	**1 (2/0.05)**	0.7987 (7/0.23)	**1 (23)**	0.9998 (36)	0.4 (43/0)	0.4831 (31)	**1 (14)**	**1 (12/4.2)**
	NMI	**1 (2/0.04)**	0.9982 (7/0.23)	**1 (12)**	**1 (14)**	**1 (43/0)**	0.6128 (29)	**1 (14)**	**1 (12/4.2)**
	Acc	**1 (2/0.04)**	0.9993 (7/0.23)	**1 (12)**	**1 (13)**	0.9491 (43/0)	0.7904 (31)	**1 (14)**	**1 (12/4.2)**
Pathbased	ARI	0.8948 (9/0.08)	0.7867 (9/0.1)	0.8819 (12)	0.9065 (6)	0.7505 (32/0.5)	0.5756 (96)	**0.959 (6)**	**0.959 (6/1.6)**
	FM	0.8116 (3/0.06)	0.6281 (9/0.11)	0.959 (12)	0.9671 (6)	0.226 (29/0.2)	0.824 (95)	**0.987 (6)**	0.9771 (8/1.9)
	NMI	0.956 (9/0.08)	**1 (4/0.22)**	0.8947 (12)	0.9336 (6)	0.7357 (32/0.5)	0.6044 (96)	**1 (1)**	**1 (4/2.3)**
	Acc	0.9898 (9/0.08)	**1 (4/0.22)**	0.9726 (12)	0.9862 (6)	**1 (25/0)**	0.82 (95)	**1 (1)**	**1 (1/3.2)**
R15	ARI	0.9893 (30/0.05)	0.8682 (8/0.22)	0.9743 (26)	0.9857 (30)	0.9098 (42/0.1)	**0.9964 (39)**	0.9928 (22)	0.9929 (30/1.8)
	FM	0.995 (30/0.05)	0.8949 (8/0.24)	0.9882 (26)	0.9933 (30)	0.0485 (43/0.15)	**0.9983 (39)**	0.9967 (22)	0.9967 (30/3.3)
	NMI	**1 (6/0.02)**	0.9916 (8/0.24)	**1 (12)**	**1 (12)**	0.9666 (42/0.1)	0.9971 (39)	0.9942 (22)	0.9942 (30/3.3)
	Acc	**1 (6/0.02)**	0.9943 (8/0.24)	**1 (12)**	**1 (12)**	**1 (43/0.15)**	0.9983 (39)	0.9967 (22)	0.9967 (7/2.2)
Spiral	ARI	**1 (2/0.04)**	0.5617 (5/0.21)	**1 (5)**	**1 (2)**	**1 (26/0)**	0.1487 (4)	**1 (2)**	**1 (2/1.3)**
	FM	**1 (2/0.04)**	0.7367 (12/0.02)	**1 (5)**	**1 (2)**	0 (26/0)	0.5542 (4)	**1 (2)**	**1 (2/1.2)**
	NMI	**1 (2/0.04)**	**1 (9/0.07)**	**1 (5)**	**1 (2)**	**1 (26/0)**	0.204 (19)	**1 (2)**	**1 (2/1.2)**
	Acc	**1 (2/0.04)**	**1 (9/0.07)**	**1 (5)**	**1 (2)**	**1 (26/0.05)**	0.5545 (4)	**1 (2)**	**1 (2/1.2)**
Toy	ARI	0.967 (13/0.06)	**1 (32/0.07)**	**1 (17)**	0.9917 (15)	**1 (36/0.25)**	0.8008 (43)	**1 (16)**	**1 (12/1.6)**
	FM	0.8301 (17/0.1)	0.9067 (18/0.12)	**1 (17)**	0.9991 (15)	0 (36/0.25)	0.9331 (43)	**1 (16)**	**1 (11/2.5)**
	NMI	**1 (8/0.05)**	**1 (10/0.29)**	**1 (17)**	**1 (15)**	**1 (36/0.25)**	0.7198 (43)	**1 (10)**	**1 (2/4)**
	Acc	**1 (8/0.05)**	**1 (3/0.49)**	**1 (17)**	**1 (15)**	**1 (36/0.25)**	0.9517 (43)	**1 (2)**	**1 (2/1.4)**
Diamonds	ARI	0.9975 (12/0.06)	0.9751 (27/0.07)	-	0.995 (36)	**1 (39/0.05)**	**1 (41)**	0.995 (22)	**1 (9/1.5)**
	FM	0.9994 (12/0.06)	0.9937 (27/0.08)	-	0.9987 (36)	**1 (39/0.05)**	**1 (41)**	0.9987 (22)	**1 (10/1.3)**
	NMI	**1 (35/0.09)**	**1 (33/0.07)**	-	**1 (12)**	**1 (39/0.05)**	**1 (41)**	**1 (2)**	**1 (2/1.6)**
	Acc	**1 (12/0.06)**	**1 (2/0.6)**	-	**1 (12)**	**1 (35/0)**	**1 (41)**	**1 (2)**	**1 (2/1.5)**

largest number of similar *core* points in its neighbourhood. Second, it retrieves all points that are weighted density reachable from y_i.

3.2.1. Calculating feature weights in W-DBSCANR

Feature weighting, can be thought of as a generalization of feature selection. Under this view, feature selection assigns a binary weight. A weight of one means the feature is selected, and a weight of zero means the feature is deselected. Feature weighting assigns a value, usually in the interval $[0, 1]$, to each feature. In our model, the higher this value is for a particular feature, the more relevant the feature is. In fact, we go further and assign a weight to each feature at each cluster. Feature weighting is a rather intuitive approach because even among relevant features there may be different degrees of relevance. That is, a feature v may have different degrees of relevance at different clusters. Also, feature weights can be used as a starting point for feature selection (see for instance [27], and references therein).

In order to calculate feature weights we introduce a new step to DBSCANR. This allows us to iteratively update each feature weight at each cluster based on the current partition. In the first iteration we set each feature weight, w_{cv}, to $\frac{1}{V}$ so that all feature weights have the same value to start from.

With the above, we can recover K clusters from the first iteration of our algorithm, and represent this clustering using graphs. Let G be a graph with K components $G_{(1)}, G_{(2)}, \ldots, G_{(K)}$, so that the vertices of $G_{(c)}$ (with $1 \leq c \leq K$) represent the data points of a cluster $S_c \in S$. Given $G_{(c)}$, we can generate V graphs $G_{(c,1)}, G_{(c,2)}, \ldots, G_{(c,V)}$, so that each edge of $G_{(c,v)}$ (with $1 \leq v \leq V$) is the feature-wise distance between its endvertices calculated using

$$d^W_{cv}(y_{iv}, y_{jv}) = \frac{d^W(y_i, y_j)}{w^\beta_{cv}}. \tag{9}$$

The equation above ensures a fair distribution of $d^W(y_i, y_j)$ over each feature v. Notice that $d^W(y_i, y_j)$ is calculated over all features. However, the division by w^β_{cv} ensures the degree of relevance of a feature v at cluster $S_c \in S$ is taken into account. A lower weight leads to a higher distance, and by consequence a less compact cluster.

The above requires a precise definition of compactness. Given a graph $G_{(c,v)}$, representing the feature v at cluster $S_c \in S$, we can calculate its compactness based on the edges of its minimum spanning tree (MST), $G^*_{(c,v)}$, after removing all vertices of degree one. Let

$$u_{ijc} = \begin{cases} 1, \text{ if there exists an edge between } y_i, y_j \in S_c \text{ in } G^*_{(c,v)} \\ 0, \text{ otherwise.} \end{cases} \tag{10}$$

Table 5
Results of the experiments on the original real-world data sets (no noise features have been added). We measure cluster recovery using the Adjusted Rand index (ARI), F-Measure (FM), Normalised Mutual Information (NMI) and Accuracy (Acc).

Data set	CVI	DBSCAN (k/ϵ)	OPTICS (k/m_{cr})	ISDBSCAN (k)	RNN-DBSCAN (k)	ADBSCAN (k/p)	DPC-DBFN (k)	DBSCANR (k)	W-DBSCANR (k/β)
Banknote	ARI	0.0216 (12/0.21)	0.0214 (42/0.1)	-0.0109 (34)	-0.0081 (46)	0.0789 (181/0.5)	0.1216 (178)	0.8433 (14)	**0.8513 (14/2.5)**
	FM	0.4162 (33/0.18)	0.3584 (40/0.06)	0.4029 (34)	0.4062 (42)	0.2062 (92/0)	0.6112 (178)	0.9586 (14)	**0.9868 (35/2.6)**
	NMI	**1 (10/0.03)**	**1 (26/0.09)**	0.0835 (34)	0.0867 (42)	0.0635 (196/0.2)	0.1936 (251)	**1 (12)**	**1 (11/3.7)**
	Acc	**1 (10/0.03)**	**1 (4/0.56)**	0.5037 (39)	0.5106 (45)	0.6306 (181/0.5)	0.6786 (178)	**1 (6)**	**1 (2/1.2)**
BreastC.	ARI	0.8526 (10/0.94)	0.8502 (42/0.17)	0.6505 (45)	0.0404 (8)	0.7828 (179/0.4)	0.2948 (299)	0.8452 (34)	**0.8661 (36/1.2)**
	FM	0.8244 (8/0.58)	0.0576 (20/0.11)	0.912 (45)	0.3964 (8)	0.006 (179/0.4)	0.771 (299)	0.9568 (34)	**0.9674 (38/1.5)**
	NMI	**1 (3/0.25)**	0 (10/0.11)	0.8399 (28)	0.0299 (9)	0.6912 (179/0.4)	0.3915 (299)	0.787 (34)	0.8265 (38/1.5)
	Acc	**1 (3/0.25)**	0.6176 (10/0.11)	0.9725 (28)	0.6617 (8)	0.9882 (179/0.45)	0.7725 (299)	0.9599 (34)	0.97 (38/1.5)
BreastT.	ARI	0.152 (2/0.48)	0.0148 (4/0.08)	0.0976 (5)	0.2892 (3)	0.3905 (20/0.35)	0.4087 (30)	0.2773 (3)	**0.4532 (3/2)**
	FM	0.2339 (2/0.24)	0.1503 (4/0.08)	0.3673 (5)	0.3977 (3)	0.0842 (20/0.35)	0.5704 (23)	0.5305 (3)	**0.5831 (3/1.4)**
	NMI	0.4241 (2/0.24)	0.1687 (4/0.08)	0.5478 (5)	0.629 (3)	0.5548 (32/0.25)	0.5472 (24)	0.7102 (1)	**0.7846 (1/2.9)**
	Acc	0.3333 (2/0.24)	0.2692 (4/0.08)	0.541 (5)	0.4891 (3)	0.5 (21/0.45)	0.5943 (23)	0.7222 (2)	**0.766 (2/3)**
Ecoli	ARI	-0.0084 (4/0.19)	0.4217 (14/0.03)	-	-	0.4221 (36/0.15)	**0.6581 (3)**	0.4206 (3)	0.5398 (3/1.7)
	FM	0.1607 (4/0.09)	-	-	-	0.1162 (52/0)	**0.5115 (2)**	0.4517 (3)	0.4988 (3/1.5)
	NMI	**0.6687 (4/0.09)**	-	-	-	0.4978 (52/0)	0.6268 (3)	0.6423 (1)	0.6423 (1/1.1)
	Acc	**0.7765 (4/0.09)**	-	-	-	0.6465 (52/0)	0.756 (3)	0.5863 (3)	0.6726 (3/1.5)
Iris	ARI	0.628 (5/0.25)	0.6063 (13/0.03)	0.4607 (12)	0.4008 (6)	0.886 (40/0)	0.851 (21)	0.8681 (7)	**0.9222 (5/1.5)**
	FM	0.8434 (5/0.13)	0.8266 (13/0.04)	0.6178 (12)	0.4662 (6)	0.4122 (42/0.05)	0.9466 (21)	0.9533 (7)	**0.9733 (5/2.1)**
	NMI	0.9583 (7/0.13)	0.893 (2/0.22)	0.7111 (12)	0.654 (6)	0.8862 (40/0)	0.8366 (21)	0.8498 (7)	**1 (3/3.7)**
	Acc	0.9899 (7/0.13)	0.9615 (2/0.22)	0.7197 (12)	0.554 (6)	0.8929 (40/0)	0.9467 (21)	0.9533 (7)	**1 (3/1.4)**
Leaf	ARI	-	0.0058 (2/0.08)	-	-	0.2681 (21/0.3)	0.3646 (2)	0.4096 (2)	**0.4136 (2/1.6)**
	FM	-	-	-	-	0.0076 (20/0.3)	0.5596 (5)	0.5327 (2)	**0.5796 (3/2.5)**
	NMI	-	-	-	-	0.6472 (21/0.3)	0.7064 (2)	0.8876 (1)	**0.9238 (1/2.8)**
	Acc	-	-	-	-	0.0833 (17/0.45)	0.5353 (2)	0.7283 (1)	**0.7816 (1/2.8)**
Leuk72	ARI	**0.8947 (2/3.4)**	-	0.7439 (16)	0.8264 (3)	0.743 (28/0.3)	0.8809 (1)	0.8809 (3)	0.8809 (2/1.1)
	FM	**0.9635 (2/1.7)**	-	0.9055 (16)	0.9439 (3)	0.4366 (28/0.3)	0.9568 (1)	0.9574 (4)	0.9574 (4/1.1)
	NMI	**1 (2/0.92)**	-	0.9366 (10)	0.9437 (3)	0.7227 (28/0.3)	0.8593 (1)	**1 (1)**	**1 (1/1.4)**
	Acc	**1 (2/0.92)**	-	0.9828 (10)	0.9848 (3)	0.7826 (28/0.35)	0.9583 (1)	**1 (1)**	**1 (1/1.4)**
Libras	ARI	0.0369 (4/1.42)	0.0014 (2/0.17)	-	0.2676 (4)	0.2861 (37/0.15)	0.1346 (39)	0.3752 (5)	**0.4141 (5/1.4)**
	FM	0.2827 (4/0.71)	-	-	0.3701 (4)	0.0509 (37/0.15)	0.3102 (6)	0.473 (5)	**0.4783 (5/5)**
	NMI	0.8451 (4/0.71)	-	-	0.6401 (4)	0.5629 (37/0.15)	0.4461 (6)	0.7992 (1)	**0.8462 (2/1.3)**
	Acc	0.6545 (4/0.71)	-	-	0.4231 (4)	0.4545 (37/0.15)	0.3194 (6)	0.5851 (2)	**0.7297 (2/1.3)**
Liver	ARI	0.0521 (9/0.08)	0.0565 (11/0.12)	0.0072 (46)	0.032 (9)	-0.0003 (179/0.1)	0.0387 (174)	0.0327 (5)	**0.076 (10/1.2)**
	FM	0.5399 (2/0.82)	0.5404 (4/0.29)	0.4753 (49)	0.5343 (13)	0.1338 (162/0.45)	**0.5473 (174)**	0.5399 (5)	0.5424 (5/3.7)
	NMI	0.0499 (4/0.02)	0.0141 (11/0.13)	0.0063 (46)	0.0084 (13)	0.0078 (162/0.45)	0.0163 (263)	**0.4421 (1)**	0.2766 (1/4.5)
	Acc	0.6822 (41/0.13)	0.7794 (6/0.15)	0.6149 (49)	0.6521 (9)	0.06 (92/0.5)	0.7547 (1)	**0.875 (1)**	0.7778 (3/2.5)
Parkinsons	ARI	0.1877 (7/0.65)	-0.0069 (4/0.08)	0.0834 (10)	0.2473 (5)	0 (89/0)	0.3595 (2)	0.2967 (8)	**0.3891 (4/1.9)**
	FM	0.601 (10/0.47)	0.4282 (4/0.1)	0.5705 (10)	0.6434 (5)	0.1299 (89/0.25)	**0.7381 (2)**	0.6822 (8)	0.7328 (5/1.5)
	NMI	0.6616 (5/0.25)	0 (2/0.32)	0.2718 (10)	0.2626 (5)	0 (89/0)	0.2736 (11)	**1 (1)**	**1 (3/4.2)**
	Acc	0.9167 (5/0.25)	0.7526 (4/0.1)	0.7832 (10)	0.8177 (5)	0.7541 (89/0)	0.8308 (2)	**1 (1)**	**1 (1/1.6)**
Seeds	ARI	0.4916 (18/0.5)	0.4202 (15/0.06)	0.3855 (12)	-	0.4083 (40/0.15)	0.7664 (2)	0.6132 (3)	**0.7909 (9/2.8)**
	FM	0.7641 (18/0.25)	0.6222 (26/0.04)	0.516 (12)	-	0.3829 (57/0.35)	0.9135 (2)	0.8537 (3)	**0.9176 (9/4.4)**
	NMI	0.9368 (16/0.23)	0.9596 (4/0.09)	0.5773 (12)	-	0.482 (55/0.35)	0.7343 (2)	**1 (1)**	**1 (1/1.7)**
	Acc	0.9836 (16/0.23)	0.988 (4/0.09)	0.6519 (12)	-	0.2349 (59/0.4)	0.9143 (2)	**1 (1)**	**1 (1/1.5)**
Soya	ARI	**1 (2/6.38)**	-	0.9776 (5)	0.9776 (4)	**1 (22/0.15)**	0.5952 (9)	**1 (2)**	**1 (2/1.2)**
	FM	**1 (2/3.19)**	-	0.9868 (5)	0.9868 (4)	**1 (22/0.15)**	0.8506 (9)	**1 (2)**	**1 (2/1.6)**
	NMI	**1 (2/3.19)**	-	**1 (7)**	**1 (5)**	**1 (22/0.15)**	0.7623 (9)	**1 (1)**	**1 (1/1.4)**
	Acc	**1 (2/3.19)**	-	**1 (4)**	**1 (2)**	0.5 (22/0.15)	0.8298 (9)	**1 (1)**	**1 (1/1.4)**
TeachingA.	ARI	0.022 (11/3.01)	0.0092 (4/0.02)	0.0182 (9)	-	0.0026 (50/0.45)	**0.0714 (1)**	0.0119 (3)	0.0339 (10/1.5)
	FM	0.3359 (6/1.44)	0.1795 (4/0.05)	0.3885 (9)	-	0 (34/0.2)	0.3985 (54)	0.4024 (6)	**0.4665 (10/5)**
	NMI	0.4051 (3/0.02)	0.2415 (3/0.02)	0.0611 (9)	-	0.0499 (50/0.45)	0.1112 (1)	0.3062 (1)	**0.5328 (5/2.6)**
	Acc	0.5455 (3/0.02)	0.619 (3/0.02)	0.4552 (9)	-	0.6667 (45/0.45)	0.4702 (1)	0.6 (2)	**0.8 (5/2.6)**
Wine	ARI	0.4497 (17/0.95)	-	0.5635 (9)	0.3738 (3)	0.3967 (32/0.4)	0.8465 (46)	0.7123 (3)	**0.8672 (4/1.7)**
	FM	0.6939 (23/0.51)	-	0.817 (9)	0.5526 (3)	0.2134 (32/0.05)	**0.9514 (46)**	0.9015 (3)	0.9506 (5/2.3)
	NMI	**1 (7/0.38)**	-	0.9057 (9)	0.5871 (3)	0.4164 (35/0.2)	0.8237 (46)	0.8044 (2)	**1 (2/2.1)**
	Acc	**1 (7/0.38)**	-	0.969 (9)	0.6433 (3)	**1 (39/0.25)**	0.9494 (46)	0.8989 (3)	**1 (2/2.1)**
Zoo	ARI	0.8142 (2/2.05)	**0.9106 (4/0.25)**	0.4972 (6)	0.7696 (3)	0.543 (32/0)	0.4185 (31)	0.8203 (4)	0.7357 (5/1.7)
	FM	0.7298 (2/1.02)	0.5506 (4/0.27)	0.6973 (6)	0.7222 (3)	0 (12/0.5)	0.4842 (19)	**0.7484 (4)**	0.7462 (5/1.3)
	NMI	0.9279 (2/1.02)	**0.9645 (3/0.34)**	0.8379 (6)	0.8885 (3)	0.7621 (32/0)	0.5554 (31)	0.8543 (4)	0.8566 (5/1.3)
	Acc	0.9231 (2/1.02)	0.9831 (3/0.34)	0.7831 (6)	0.8817 (3)	**1 (12/0.5)**	0.5644 (31)	0.8614 (4)	0.8416 (5/1.3)

We can then measure the compactness of $G_{(c,v)}$ with

$$\mathcal{C}_{cv} = \max\left\{\sum_i \sum_j u_{ijc} \frac{d^W(y_i, y_j)}{w_{cv}^{\beta}}\right\}. \tag{11}$$

Now that we have a measure of compactness, we would like to minimise it over all clusters and features. Let $\hat{w}_{cv}$ be the weight found in the previous iteration (or $\hat{w}_{cv} = V^{-1}$, if in the first iteration). The optimal w_{cv} is that which minimises

$$\sum_{c=1}^{K}\sum_{v=1}^{V} w_{ck}^{\beta} \max\left\{\sum_i \sum_j u_{ijc} \frac{d^W(y_i, y_j)}{\hat{w}_{cv}^{\beta}}\right\} = \sum_{c=1}^{K}\sum_{v=1}^{V} w_{cv}^{\beta} \mathcal{C}_{cv}. \tag{12}$$

Table 6
Results of the experiments on the high-dimensional data sets. We measure cluster recovery using the Adjusted Rand index (ARI), F-Measure (FM), Normalised Mutual Information (NMI) and Accuracy (Acc).

Data set	CVI	DBSCAN (k/ϵ)	OPTICS (k/m_{cr})	ISDBSCAN (k)	RNN-DBSCAN (k)	ADBSCAN (k/p)	DPC-DBFN (k)	DBSCANR (k)	W-DBSCANR (k/β)
ALLAML	ARI	0.1653 (3/17.23)	-	0.0833 (11)	-	0.05 (29/0.15)	**0.2037 (7)**	-0.035 (2)	0.1171 (1/1.2)
	FM	0.4836 (3/17.23)	-	0.4805 (11)	-	0 (22/0.25)	**0.6535 (7)**	0.2516 (2)	0.55 (3/1.3)
	NMI	0.3632 (3/16.99)	-	0.6255 (7)	-	0.078 (29/0.15)	0.2491 (5)	0.1204 (2)	**1 (1/1.2)**
	Acc	0.8684 (3/16.99)	-	0.931 (7)	-	0.1822 (29/0.15)	0.75 (5)	0.7059 (2)	**1 (1/1.2)**
Carcinom	ARI	0.2975 (2/20.91)	-	-	-	0.5734 (29/0.4)	0.1986 (21)	0.4885 (2)	**0.5981 (2/2.3)**
	FM	0.4542 (2/20.91)	-	-	-	0 (8/0.5)	0.3506 (2)	0.5777 (2)	**0.6616 (2/2.3)**
	NMI	**0.906 (2/19.85)**	-	-	-	0.7033 (34/0.3)	0.4097 (3)	0.8412 (1)	0.8741 (1/2.3)
	Acc	**0.8667 (2/19.85)**	-	-	-	0 (8/0.5)	0.408 (21)	0.7292 (1)	0.7619 (1/2.3)
CLL-SUB	ARI	-0.0129 (2/15.6)	-	-	-	-	**0.2297 (2)**	0.0341 (2)	0.0794 (2/2.1)
	FM	0.1411 (2/18.29)	-	-	-	-	**0.6081 (10)**	0.3985 (2)	0.4424 (2/1.9)
	NMI	0.312 (2/15.6)	-	-	-	-	**0.4682 (2)**	0.2358 (2)	0.3011 (2/1.4)
	Acc	0.5185 (2/18.29)	-	-	-	-	0.6036 (10)	0.5472 (2)	**0.6296 (2/2.1)**
Colon	ARI	0.013 (4/14.9)	-	-0.0101 (12)	0.1307 (3)	-0.0178 (29/0.4)	0.1911 (4)	0.1052 (4)	**0.2327 (3/2.9)**
	FM	0.5133 (6/18.53)	-	0.5016 (12)	0.5581 (3)	0.36 (24/0.3)	0.654 (24)	0.5914 (4)	**0.6903 (3/2.9)**
	NMI	0.2443 (3/13.25)	-	0.0024 (12)	0.195 (3)	0.1064 (28/0.25)	0.1743 (4)	0.0664 (4)	**0.3674 (1/2.7)**
	Acc	0.6364 (3/13.25)	-	0.5167 (12)	0.7241 (3)	0.5 (24/0.3)	0.7419 (4)	0.6935 (4)	**0.7778 (1/2.7)**
GLIOMA	ARI	0.3967 (2/15.86)	-	-	-	0.3235 (8/0.45)	0.3527 (2)	0.4841 (3)	**0.5128 (2/1.4)**
	FM	0.5536 (3/15.6)	-	-	-	0 (8/0.45)	0.6607 (13)	0.7498 (3)	**0.775 (2/1.4)**
	NMI	0.7368 (2/15.86)	-	-	-	0.492 (11/0.5)	0.5018 (1)	0.9069 (2)	**0.9069 (2/1.7)**
	Acc	0.7143 (2/12.48)	-	-	-	0.7273 (25/0)	0.66 (13)	0.9524 (2)	**0.9524 (2/1.7)**
Lung	ARI	0.2603 (2/11.46)	-	0.263 (10)	-	0.4969 (51/0.5)	0.3588 (2)	0.4413 (3)	**0.6364 (5/1.1)**
	FM	0.3624 (2/11.1)	-	0.6315 (10)	-	0.0012 (51/0.5)	0.5568 (20)	0.4891 (3)	**0.7012 (5/1.1)**
	NMI	**0.7337 (4/9.32)**	-	0.6578 (10)	-	0.5758 (60/0)	0.4369 (20)	0.5559 (3)	0.6922 (5/1.1)
	Acc	0.7535 (2/11.1)	-	0.7634 (10)	-	0.8333 (60/0)	0.6897 (4)	0.7438 (3)	**0.8438 (2/1.1)**
Lymphoma	ARI	0.2824 (2/25.63)	-	-	-	**0.6461 (17/0)**	0.4346 (1)	-0.0115 (2)	0.2843 (2/1.5)
	FM	0.3322 (2/25.63)	-	-	-	0.0055 (17/0)	0.4291 (12)	0.3606 (2)	**0.5596 (2/1.5)**
	NMI	0.7506 (2/25.63)	-	-	-	**0.7831 (17/0)**	0.5303 (1)	0.7626 (2)	0.7686 (2/2.5)
	Acc	0.4912 (2/26.68)	-	-	-	**1 (17/0)**	0.5729 (1)	0.6047 (2)	0.6364 (2/1.8)
ORL	ARI	0.1328 (3/4.79)	-	-	-	0.4027 (31/0.15)	0.1863 (35)	0.4377 (2)	**0.4575 (3/1.9)**
	FM	0.4667 (3/4.79)	-	-	-	0.0214 (29/0.25)	0.3302 (24)	0.6005 (3)	**0.6059 (3/1.7)**
	NMI	0.7961 (3/4.79)	-	-	-	0.7957 (31/0.15)	0.5971 (35)	**0.9397 (1)**	0.9302 (2/1.2)
	Acc	0.5063 (3/4.79)	-	-	-	0.125 (29/0.25)	0.3575 (24)	0.7526 (1)	**0.8071 (2/1.4)**
Prostate-GE	ARI	0.0359 (5/16.09)	-	0.0204 (21)	0.0003 (5)	-	0.0586 (21)	0.0216 (2)	**0.0805 (2/1.6)**
	FM	0.4534 (2/20.9)	-	0.5024 (17)	0.4195 (5)	-	**0.5971 (23)**	0.4218 (5)	0.5567 (6/1.1)
	NMI	**1 (2/9.85)**	-	0.0089 (13)	0.0367 (5)	-	0.0762 (21)	0.1678 (1)	0.1871 (2/1.2)
	Acc	**1 (2/9.85)**	-	0.5517 (17)	0.53 (5)	-	0.6275 (21)	0.6429 (2)	0.7222 (2/1.4)
TOX-171	ARI	0.1497 (4/14.21)	-	0.1234 (7)	-	-	**0.179 (2)**	0.0075 (2)	0.1517 (3/1.1)
	FM	0.2617 (4/14.84)	-	0.4109 (7)	-	-	**0.4514 (2)**	0.1893 (2)	0.4358 (3/1.1)
	NMI	**0.4812 (3/12.76)**	-	0.4063 (7)	-	-	0.2639 (2)	0.3508 (1)	0.3944 (2/3.1)
	Acc	**0.6071 (5/13.38)**	-	0.5455 (7)	-	-	0.462 (2)	0.56 (1)	0.566 (2/1.3)

We can minimise the above, subject to $\sum_{v=1}^{V} w_{cv} = 1$ for $c = 1, \ldots, K$, with the Lagrange function

$$\mathcal{L} = \sum_{v=1}^{V} w_{cv}^{\beta} \mathcal{C}_{cv} + \lambda \left(1 - \sum_{v=1}^{V} w_{cv} \right), \tag{13}$$

whose derivative with respect to w_{ck} is

$$\frac{\partial \mathcal{L}}{\partial w_{cv}} = \beta w_{cv}^{\beta-1} \mathcal{C}_{cv} - \lambda. \tag{14}$$

Equating the above to zero leads to

$$\left(\frac{\lambda}{\beta} \right)^{\frac{1}{\beta-1}} = w_{cv} \mathcal{C}_{cv}^{\frac{1}{\beta-1}} \iff w_{cv} = \left(\frac{\lambda}{\beta \mathcal{C}_{cv}} \right)^{\frac{1}{\beta-1}}. \tag{15}$$

Summing the above over all features leads to

$$\sum_{v=1}^{V} \left(\frac{\lambda}{\beta \mathcal{C}_{cv}} \right)^{\frac{1}{\beta-1}} = 1 \iff \left(\frac{\lambda}{\beta} \right)^{\frac{1}{\beta-1}} = \frac{1}{\sum_{v=1}^{V} \left(\frac{1}{\mathcal{C}_{cv}} \right)^{\frac{1}{\beta-1}}}. \tag{16}$$

Finally we have

$$w_{cv} = \frac{1}{\sum_{u=1}^{V} \left[\frac{\mathcal{C}_{cv}}{\mathcal{C}_{cu}} \right]^{\frac{1}{\beta-1}}}. \tag{17}$$

We are now ready to present our feature weighted density-based clustering method W-DBSCANR as follows:

4. W-DBSCANR complexity

Since W-DBSCANR is an extension of the DBSCANR algorithm, and DBSCANR needs to calculate the k-nearest neighbours of each point, if n is the cardinality of the data set, the direct implementation of W-DBSCANR has $O(n^2)$ time complexity. The time complexity of W-DBSCANR depends on the following five parts: 1) the time complexity of finding k-nearest neighbours. The key issue of DBSCAN-type clustering methods is identifying each point type, which is a k-nearest neighbour problem. DBSCANR is not any different. Therefore, improving the k-nearest neighbour's complexity will improve the computational complexity of DBSCANR and of W-DBSCANR. Many techniques were proposed to improve the runtime of the nearest neighbour query. For instance, Kd-tree [28], semi-convex hull tree [29], and trinary-project tree [30] are some examples to name but a few. Most of the proposed algorithms degenerate in higher dimensional space [31]. If Kd-tree is used for the nearest neighbour query, the time complexity of finding the reverse nearest neighbours of each point is $O(n \log n)$ [32]. DBSCANR computes all pairwise distances to determine the core and non-core points, which requires $O(n^2)$. 2) the time complexity of finding the core points. Determining the core point requires $O(n)$ time given the nearest neighbours have already been calculated. 3) the time complexity of clustering the core points. If c is the number of core points and r is the number of core points in the reverse nearest neighbour of the c points, then clustering core points requires

Table 7
The results of our experiments on the synthetic data sets with 50% added noise features. We measure cluster recovery using the Adjusted Rand index (ARI), F-Measure (FM), Normalised Mutual Information (NMI) and Accuracy (Acc).

Data set	CVI	DBSCAN (k/ϵ)	OPTICS (k/m_{cr})	ISDBSCAN (k)	RNN-DBSCAN (k)	ADBSCAN (k/p)	DPC-DBFN (k)	DBSCANR (k)	W-DBSCANR (k/β)
Aggregation	ARI	0.8428 (11/0.15)	0.8583 (27/0.03)	-	-	0.7619 (45/0.2)	0.7382 (13)	0.7216 (8)	**0.8705 (8/2.1)**
	FM	0.6525 (11/0.15)	0.6041 (9/0.11)	-	-	0.083 (53/0.4)	0.7054 (10)	0.6434 (9)	**0.9975 (19/1.2)**
	NMI	0.9013 (11/0.15)	0.9702 (9/0.11)	-	-	0.7868 (45/0.2)	0.8087 (13)	0.9053 (1)	**0.9924 (19/1.2)**
	Acc	0.9029 (11/0.15)	0.9866 (9/0.11)	-	-	**1 (40/0)**	0.8414 (10)	0.7551 (7)	0.9975 (19/1.2)
D31	ARI	0.2285 (13/0.08)	0.0571 (53/0)	-	-	-	0.2955 (9)	0.288 (9)	**0.2998 (10/2.6)**
	FM	0.3234 (13/0.08)	0.2158 (7/0.14)	-	-	-	0.3671 (10)	**0.3852 (9)**	**0.3995 (10/2.6)**
	NMI	0.8015 (19/0.08)	**0.9059 (5/0.2)**	-	-	-	0.6702 (14)	0.8716 (3)	**0.9157 (3/4.6)**
	Acc	0.5632 (19/0.08)	**0.7725 (5/0.2)**	-	-	-	0.3771 (10)	0.6675 (3)	**0.7981 (3/1.2)**
Flame	ARI	0.5742 (15/0.22)	0.3878 (28/0.03)	-	-0.0017 (7)	0.0193 (37/0)	0.6777 (63)	0.0078 (7)	**0.9833 (8/1.1)**
	FM	0.8816 (15/0.22)	0.7905 (26/0.03)	-	0.5638 (7)	0.1779 (42/0.15)	**0.9056 (63)**	0.5814 (7)	0.8214 (4/2.8)
	NMI	**1 (8/0.13)**	0.6519 (28/0.03)	-	0.1633 (7)	0.0467 (42/0.4)	0.5534 (63)	0.1572 (7)	**1 (1/2.2)**
	Acc	**1 (8/0.13)**	0.9348 (28/0.03)	-	0.5975 (8)	0.6456 (37/0)	0.9125 (63)	0.6 (6)	**1 (1/2.2)**
Grid	ARI	0.8277 (6/0.12)	0.8082 (17/0.04)	**0.888 (21)**	-	0.7171 (34/0.45)	0.5785 (3)	0.1533 (4)	0.8867 (7/4.9)
	FM	0.5229 (20/0.21)	0.5127 (8/0.1)	0.9698 (21)	-	0.3719 (40/0)	0.872 (3)	0.6297 (4)	**0.9764 (9/1.2)**
	NMI	0.2735 (6/0.12)	0.68 (2/0.34)	0.8655 (9)	-	0.6727 (34/0.45)	0.5619 (3)	**1 (1)**	**1 (2/1.7)**
	Acc	0.949 (9/0.1)	0.9316 (16/0.05)	0.9776 (9)	-	0.8824 (34/0.5)	0.8809 (3)	**1 (1)**	**1 (2/1.7)**
Mixed	ARI	0.6509 (41/0.16)	0.8594 (5/0.21)	0.8435 (14)	0.864 (11)	0.89 (44/0.2)	0.7132 (16)	0.8809 (7)	**1 (8/1.5)**
	FM	0.4335 (13/0.12)	0.3199 (9/0.16)	0.6209 (14)	0.5593 (11)	0.2344 (44/0.2)	0.5041 (16)	0.6326 (7)	**1 (18/1.1)**
	NMI	0.7429 (25/0.12)	0.9126 (2/0.43)	0.8368 (12)	0.8153 (11)	0.8143 (44/0.2)	0.6062 (16)	0.8257 (9)	**1 (18/1.1)**
	Acc	0.8842 (26/0.12)	0.8876 (9/0.16)	0.9094 (12)	0.9147 (11)	0.9731 (53/0.5)	0.8465 (16)	0.9182 (7)	**1 (18/1.1)**
Pathbased	ARI	0.5674 (6/0.12)	0.3361 (38/0.02)	0.1511 (18)	-	0.6217 (38/0.45)	0.3689 (131)	0.1663 (5)	**0.671 (5/1.3)**
	FM	0.5546 (8/0.13)	0.5375 (38/0)	0.3372 (18)	-	0.2692 (48/0.5)	0.697 (131)	0.5377 (5)	**0.7407 (31/1.1)**
	NMI	0.7612 (2/0.02)	0.7867 (3/0.26)	0.2863 (18)	-	0.6394 (42/0.5)	0.3835 (4)	**1 (2)**	**1 (2/3.7)**
	Acc	0.8161 (8/0.13)	0.8 (4/0.2)	0.4808 (18)	-	**1 (55/0.15)**	0.7067 (131)	**1 (2)**	**1 (2/2.2)**
R15	ARI	0.1331 (10/0.1)	0.2404 (10/0.14)	0.2352 (9)	-	0.2938 (33/0)	0.3647 (6)	0.2447 (7)	**0.9893 (8/1.3)**
	FM	0.4238 (10/0.1)	0.405 (9/0.2)	0.4742 (9)	-	0.0947 (34/0.15)	0.5208 (28)	0.4846 (7)	**0.995 (13/1.1)**
	NMI	0.6464 (10/0.1)	0.8872 (6/0.25)	0.713 (9)	-	0.7511 (33/0)	0.6767 (7)	0.888 (2)	**0.9914 (7/1.2)**
	Acc	0.5243 (10/0.1)	0.7456 (6/0.25)	0.4789 (9)	-	**1 (33/0)**	0.505 (44)	0.7062 (4)	0.995 (7/1.2)
Spiral	ARI	0.0179 (10/0.14)	0.3519 (49/0.01)	-0.0001 (7)	-	0.0017 (36/0.5)	0.035 (83)	0.0104 (4)	**1 (4/1.2)**
	FM	0.3187 (15/0.19)	0.3813 (48/0)	0.2325 (7)	-	0.1373 (37/0.1)	0.4436 (87)	0.3101 (4)	**0.8394 (6/1.3)**
	NMI	**1 (5/0.07)**	0.8269 (4/0.09)	0.0303 (7)	-	0.0414 (36/0)	0.0577 (8)	0.7987 (2)	**1 (2/1.8)**
	Acc	**1 (5/0.07)**	0.9429 (4/0.09)	0.3662 (7)	-	**1 (36/0)**	0.4423 (87)	0.8 (2)	**1 (2/1.8)**
Toy	ARI	**0.8249 (14/0.17)**	0.6834 (32/0.05)	0.1105 (12)	-0.0358 (10)	0.3125 (29/0.3)	0.4261 (240)	0.6258 (7)	0.779 (7/2)
	FM	0.6944 (20/0.2)	0.475 (32/0.06)	0.5142 (12)	0.4095 (9)	0.2952 (123/0.5)	0.8015 (240)	0.5077 (6)	**1 (15/1.2)**
	NMI	0.854 (8/0.13)	**1 (3/0.3)**	0.1363 (12)	0.0461 (10)	0.2627 (29/0.15)	0.3844 (3)	**1 (3)**	**1 (1/2.4)**
	Acc	0.9926 (8/0.13)	**1 (3/0.3)**	0.7707 (12)	0.6935 (9)	0.8107 (29/0.3)	0.8338 (5)	**1 (1)**	**1 (1/2.4)**
Diamonds	ARI	0.8368 (17/0.15)	0.0418 (17/0.05)	0.0001 (11)	-	0.0015 (43/0.5)	0.8233 (9)	0.9311 (6)	**0.995 (7/1.3)**
	FM	0.9551 (17/0.15)	0.337 (36/0.02)	0.348 (11)	-	0.3311 (38/0)	0.9537 (9)	0.9825 (6)	**1 (39/1.2)**
	NMI	**1 (35/0.16)**	**1 (2/0.47)**	0.0327 (11)	-	0.0139 (114/0.5)	0.7738 (9)	**1 (1)**	**1 (1/2.4)**
	Acc	**1 (5/0.05)**	**1 (2/0.47)**	0.5051 (11)	-	**1 (85/0.4)**	0.9538 (9)	**1 (1)**	**1 (1/2.1)**

$O(c \log c + rc)$ time. 4) the time complexity of clustering the unclustered points to their nearest core points. If there are l unclustered points and l is fairly less than the cardinality of the data set, then clustering l points requires $O(l)$ time. 5) the time complexity of updating feature weights for each cluster. If t is the number of iterations required for steps 3 and 4 of Algorithm 3, m is the number of features, and n is the data points, updating feature weights for each cluster require $O(tmn)$ time. Thus the time complexity of W-DBSCANR is $O(n^2)$.

5. Set up of experiments

In this section we describe the set up of our experiments. We experiment with both real-world and synthetic data sets, with and without added noise features.

The real-world data sets we experiment with were obtained from the popular UCI machine learning repository [33] and scikit-feature selection repository [34], for details see Table 2. These data sets have no missing values, or features with a range of zero. From some of these data sets we have generated two others by adding $\lceil 0.5V \rceil$ and V noise features, respectively. Here, a noise feature is one composed entirely of within-domain uniformly random noise. We have added noise features so that we can evaluate how the algorithms we experiment with perform under such conditions.

The synthetic data sets we experiment with were also also obtained online [35], for details see Table 1. We generated two extra data sets from each of these in a similar way to what we did regarding the real-world data sets.

We have normalised all the data sets we experiment with using

$$y_{iv} = \frac{y_{iv} - \bar{y}_v}{\text{range}(y_v)}, \tag{18}$$

We opted for (18) rather than the z-score because the former is biased towards features under a unimodal distribution. Such features are inclined to have a lower standard deviation (when compared to multimodal features) which leads to higher z-score. Hence, features with a unimodal distribution are likely to have a higher contribution to the clustering than features with a multimodal distribution. However, multimodal features are those that are usually of particular interest during clustering.

The algorithms we experiment with require parameters, we have set those according to the below. In all cases we attempted to identify the best possible parameters for each of the algorithms. All algorithms are deterministic, so the results in our tables are the best we could find.

1. DBSCAN: We experimented with k from 3 to 50 in steps of 1, and ϵ from the minimum pairwise to maximum pairwise distances for each data set in steps of 0.01.

Table 8
The results of our experiments on the synthetic data sets with 100% added noise features. We measure cluster recovery using the Adjusted Rand index (ARI), F-Measure (FM), Normalised Mutual Information (NMI) and Accuracy (Acc).

Data set	CVI	DBSCAN (k/ϵ)	OPTICS (k/m_{cr})	ISDBSCAN (k)	RNN-DBSCAN (k)	ADBSCAN (k/p)	DPC-DBFN (k)	DBSCANR (k)	W-DBSCANR (k/β)
Aggregation	ARI	0.5994 (12/0.23)	0.0755 (8/0.04)	0.5602 (9)	-	0.6433 (48/0.5)	0.628 (7)	0.6257 (4)	**0.7407 (5/1.5)**
	FM	0.3918 (12/0.23)	0.1824 (6/0.06)	0.3723 (9)	-	0.075 (40/0.05)	0.4803 (3)	0.469 (4)	**0.9988 (13/1.2)**
	NMI	0.7414 (11/0.2)	0.7958 (6/0.06)	0.7 (9)	-	0.6079 (48/0.5)	0.6601 (7)	0.8693 (2)	**0.9957 (13/1.2)**
	Acc	0.7636 (11/0.2)	0.7068 (6/0.06)	0.6917 (9)	-	**1 (41/0)**	0.7373 (7)	0.7857 (2)	0.9987 (13/1.2)
D31	ARI	-	0.0056 (10/0.02)	-	-	-	**0.1553 (4)**	0.0848 (4)	0.0291 (3/1.1)
	FM	-	0.0704 (3/0.2)	-	-	-	**0.2243 (80)**	0.1909 (4)	0.181 (3/2.4)
	NMI	-	0.7699 (3/0.2)	-	-	-	0.4651 (4)	0.8067 (1)	**0.9299 (1/1.2)**
	Acc	-	0.4959 (3/0.2)	-	-	-	0.2371 (4)	0.4946 (1)	**0.7589 (1/1.2)**
Flame	ARI	0.1367 (15/0.28)	0.0284 (6/0.07)	-0.011 (9)	-0.0245 (6)	0.0397 (34/0.15)	**0.5031 (41)**	0.0539 (3)	0.1028 (3/1.6)
	FM	0.5364 (17/0.3)	0.3902 (7/0.05)	0.417 (9)	0.3843 (6)	0 (34/0.15)	**0.8339 (41)**	0.4816 (3)	0.5794 (3/2.1)
	NMI	**1 (10/0.23)**	0 (3/0.15)	0.0822 (9)	0.0256 (6)	0.0672 (34/0.15)	0.4304 (40)	**1 (1)**	**1 (2/4.2)**
	Acc	**1 (6/0.17)**	0.6453 (7/0.05)	0.6343 (9)	0.588 (6)	0.6667 (57/0.5)	0.8583 (40)	**1 (1)**	**1 (1/1.6)**
Grid	ARI	0.7089 (9/0.21)	0.3819 (40/0.03)	0.02 (11)	0.0182 (9)	0.5282 (45/0.5)	0.2213 (299)	0.467 (4)	**0.7664 (4/3.5)**
	FM	0.5692 (23/0.31)	0.4289 (40/0.03)	0.4071 (11)	0.4142 (6)	0 (43/0.45)	0.7356 (299)	0.8273 (4)	**0.9749 (8/1.2)**
	NMI	0.2795 (8/0.19)	**1 (3/0.19)**	0.0662 (11)	0.0672 (6)	0.5063 (55/0.5)	0.2637 (7)	**1 (1)**	**1 (1/1.7)**
	Acc	0.9529 (14/0.19)	**1 (3/0.19)**	0.5974 (11)	0.5951 (6)	0.8136 (45/0.5)	0.7359 (299)	**1 (1)**	**1 (1/1.4)**
Mixed	ARI	0.5203 (35/0.24)	0.028 (6/0.08)	-	-	0.033 (53/0.5)	0.631 (7)	0.4079 (4)	**0.6919 (4/3.9)**
	FM	0.2916 (35/0.24)	0.1301 (33/0.01)	-	-	0.0952 (54/0.1)	0.3883 (7)	0.4124 (4)	**1 (12/1.1)**
	NMI	0.6022 (35/0.24)	0.8031 (5/0.11)	-	-	0.0767 (53/0.5)	0.5414 (7)	0.7331 (2)	**1 (12/1.1)**
	Acc	0.8435 (30/0.23)	0.6444 (5/0.11)	-	-	0.8049 (53/0.5)	0.8134 (7)	0.74 (2)	**1 (12/1.1)**
Pathbased	ARI	0.3964 (10/0.23)	0.0854 (7/0.05)	0.2568 (9)	-	0.4454 (33/0.45)	0.2887 (71)	0.3684 (6)	**0.4717 (17/1.1)**
	FM	0.5259 (41/0.37)	0.1846 (6/0.08)	0.622 (9)	-	0.1684 (33/0.45)	0.6884 (72)	0.5313 (6)	**0.7513 (11/1.3)**
	NMI	0.7956 (2/0.05)	0.823 (2/0.22)	0.3761 (9)	-	0.501 (33/0.45)	0.257 (71)	0.7755 (1)	**1 (1/4.3)**
	Acc	0.8012 (10/0.23)	0.9231 (2/0.22)	0.6947 (9)	-	**1 (33/0.45)**	0.6933 (71)	0.7273 (1)	**1 (1/4.3)**
R15	ARI	0.2323 (2/0.21)	0.1799 (6/0.07)	-	-	0.2216 (40/0.05)	0.1735 (38)	0.2053 (4)	**0.3546 (5/1.5)**
	FM	0.4498 (2/0.21)	0.2813 (2/0.34)	-	-	0.0639 (42/0.1)	0.3916 (18)	0.4224 (5)	**0.9967 (22/1.1)**
	NMI	0.9183 (3/0.07)	0.8954 (2/0.34)	-	-	0.6896 (42/0.1)	0.4921 (4)	0.7437 (2)	**0.9942 (22/1.1)**
	Acc	0.7292 (3/0.07)	0.8563 (2/0.34)	-	-	0.0623 (40/0.05)	0.3533 (31)	0.664 (2)	**0.9967 (22/1.1)**
Spiral	ARI	**0.0181 (12/0.25)**	0.0083 (5/0.06)	-	-	0.0023 (37/0.35)	0.0175 (203)	0.0037 (3)	0.0172 (3/3)
	FM	0.2547 (12/0.25)	0.1892 (9/0.04)	-	-	0.1034 (38/0)	0.4198 (147)	0.3353 (3)	**1 (8/1.1)**
	NMI	0.7808 (4/0.12)	0.2856 (3/0.15)	-	-	0.0463 (38/0)	0.0538 (10)	0.1388 (1)	**1 (8/1.1)**
	Acc	0.6923 (4/0.12)	0.5789 (2/0.28)	-	-	**1 (38/0)**	0.4327 (203)	0.4583 (2)	**1 (8/1.1)**
Toy	ARI	**0.7162 (15/0.27)**	0.5753 (51/0.04)	0.0783 (7)	0.0881 (5)	0.6485 (30/0.5)	0.6997 (291)	0.1568 (3)	0.578 (5/2)
	FM	0.6424 (10/0.27)	0.4625 (42/0.03)	0.4393 (7)	0.4895 (5)	0.2817 (31/0)	0.8967 (291)	0.3937 (5)	**1 (8/1.2)**
	NMI	0.4987 (12/0.27)	0.3622 (3/0.22)	0.0379 (8)	0.1043 (5)	0.5085 (30/0.5)	0.5598 (291)	**1 (2)**	**1 (1/2.9)**
	Acc	0.9544 (9/0.23)	0.8742 (42/0.03)	0.758 (7)	0.7622 (5)	0.9091 (30/0.5)	0.9249 (291)	**1 (2)**	**1 (1/1.2)**
Diamonds	ARI	0.2398 (30/0.25)	0.0186 (12/0.03)	0.0007 (9)	-	0.0004 (64/0.5)	**0.6356 (254)**	0.0764 (4)	0.3328 (4/3)
	FM	0.6616 (30/0.25)	0.3345 (17/0.03)	0.3374 (9)	-	0.3525 (46/0)	0.8986 (254)	0.5862 (4)	**1 (44/1.2)**
	NMI	**1 (17/0.19)**	**1 (2/0.25)**	0.0202 (8)	-	0.0164 (46/0)	0.5349 (254)	**1 (2)**	**1 (1/2.4)**
	Acc	**1 (16/0.19)**	**1 (2/0.25)**	0.5072 (9)	-	0.5038 (67/0.2)	0.8988 (254)	**1 (1)**	**1 (1/1.6)**

2. OPTICS: We used a method OPTICS (AutoCl) [36] in order to obtain flat partition. We set the speed control parameter of OPTICS (AutoCl), ϵ to ∞. The minimum number of points k was chosen from 3 to 50 in steps of 1, and minimum cluster ratio m_{cr} was choosen from 0.01 to 5 in steps of 0.01.
3. ISDBSCAN, RNN-DBSCAN AND DBSCANR: These algorithms have a single parameter k. We experiment with values of k from 3 to 50 as in [14,16]. As in the other algorithms, the selected k was that which produced the best cluster recovery.
4. ADBSCAN and DPC-DBFN: Both of them require a user-defined k. However, ADBSCAN demands one additional parameter, the percentage of noise in the data set. We experiment with the parameters within the allowed range based on the authors recommendation. That is, values from 0 to 0.5 in steps of 0.05.
5. W-DBSCANR: We have two parameters minimum number of points, k and weight exponent, β. We maintain the same range of values of k from 3 to 50 in steps of 1 and β was chosen from 1.1 to 5 in step of 0.1.

We evaluated cluster recovery using four commonly used Clustering Valuation Indices (CVI): the Adjusted Rand Index (ARI) [37], Normalised Mutual Information (NMI) [38], F-measure (FM), and Accuracy (Acc) [39]. The range of ARI lies between -1 and 1, and that of all other metrics is between 0 and 1. A higher value indicates better clustering recovery.

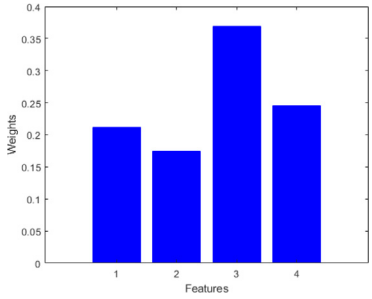


Fig. 3. Average per feature (over the three clusters) of the feature weights obtained by W-DBSCANR on the Iris data set.

6. Experiments

6.1. Experiment on original data sets without noise features

In this section, we discuss the results of our experiments on the original data sets without noise features (see Tables 1 and 2). Table 4 presents the results of the eight algorithms we experiment with, including our proposed two algorithms, on the ten original synthetic data sets. All of our results are presented in terms of four evaluation metrics (see Section 5), and all algorithms were

Table 9
The results of our experiments on the real-world data sets with 50% added noise features. We measure cluster recovery using the Adjusted Rand index (ARI), F-Measure (FM), Normalised Mutual Information (NMI) and Accuracy (Acc).

Data set	CVI	DBSCAN (k/ϵ)	OPTICS (k/m_{cr})	ISDBSCAN (k)	RNN-DBSCAN (k)	ADBSCAN (k/p)	DPC-DBFN (k)	DBSCANR (k)	W-DBSCANR (k/β)
Banknote	ARI	0.0414 (28/0.24)	0.0088 (6/0.08)	-0.001 (11)	0.0071 (8)	0.0385 (92/0.5)	**0.0578 (2)**	0.0089 (3)	0.0535 (3/1.9)
	FM	0.3758 (10/0.3)	0.3577 (12/0.04)	0.3597 (11)	0.3777 (8)	0.2055 (67/0)	0.6157 (2)	0.3819 (8)	**0.9311 (17/1.1)**
	NMI	**1 (12/0.17)**	**1 (4/0.21)**	0.0259 (11)	0.037 (8)	0.0502 (171/0.5)	0.0781 (2)	0.058 (3)	**1 (2/2.4)**
	Acc	**1 (12/0.17)**	**1 (4/0.21)**	0.556 (11)	0.5664 (8)	0.59 (92/0.5)	0.6224 (2)	0.7778 (2)	**1 (2/1.9)**
BreastC.	ARI	0.8424 (9/0.76)	0.01 (2/0.12)	0.8303 (50)	0.816 (12)	0.8394 (166/0.5)	0.8241 (266)	0.8609 (48)	**0.8717 (32/1.5)**
	FM	0.8152 (8/0.85)	-	0.9518 (50)	0.9428 (12)	0.0116 (115/0.45)	0.9505 (266)	0.961 (48)	**0.9658 (36/2.7)**
	NMI	0.7133 (11/0.82)	-	0.8977 (20)	0.8087 (18)	0.7371 (166/0.5)	0.7445 (266)	0.7756 (48)	**1 (1/1.4)**
	Acc	0.9927 (3/0.42)	-	0.9855 (20)	0.967 (23)	0.9735 (142/0.45)	0.9542 (266)	0.9642 (48)	**1 (1/1.4)**
BreastT.	ARI	0.1181 (2/0.57)	-	-	-	0.2495 (27/0.25)	**0.286 (37)**	0.1917 (2)	0.2842 (2/1.1)
	FM	0.1822 (2/0.57)	-	-	-	0.0535 (28/0)	0.4311 (33)	0.361 (2)	**0.4447 (3/2)**
	NMI	0.3535 (2/0.57)	-	-	-	0.4064 (27/0.35)	0.4121 (37)	**0.528 (1)**	**0.528 (1/1.1)**
	Acc	0.2921 (2/0.57)	-	-	-	**0.7143 (26/0.4)**	0.4623 (32)	0.5333 (1)	0.5333 (1/1.1)
Ecoli	ARI	0.2715 (6/0.36)	-	-	-	0.0803 (42/0.05)	0.2642 (132)	0.298 (3)	**0.51 (2/1.2)**
	FM	0.1629 (6/0.36)	-	-	-	0.0092 (42/0.05)	0.3296 (2)	0.33 (3)	**0.3304 (3/2.5)**
	NMI	0.4654 (3/0.22)	-	-	-	0.2088 (41/0.3)	0.3705 (2)	0.5608 (2)	**0.7252 (1/1.4)**
	Acc	**0.619 (6/0.36)**	-	-	-	0.4712 (42/0.05)	0.5089 (227)	0.5506 (3)	0.6154 (1/1.4)
Iris	ARI	0.5286 (2/0.42)	0.5281 (8/0.08)	-	0.4437 (5)	**0.5584 (39/0.2)**	0.5394 (10)	0.4869 (5)	0.4986 (3/1.3)
	FM	0.6129 (5/0.3)	0.6132 (13/0.07)	-	0.5013 (5)	0.2222 (34/0.05)	**0.7001 (20)**	0.5185 (5)	0.5185 (4/2.2)
	NMI	0.7627 (9/0.3)	0.7505 (15/0.07)	-	0.6735 (5)	0.7201 (39/0.2)	0.6863 (7)	0.762 (1)	**1 (2/2.5)**
	Acc	0.875 (9/0.3)	0.7505 (15/0.07)	-	0.5986 (5)	**1 (35/0.35)**	0.72 (14)	0.8889 (1)	**1 (1/4.4)**
Leaf	ARI	-	-	-	-	0.0925 (20/0)	**0.1352 (32)**	0.0031 (1)	0.1295 (2/1.8)
	FM	-	-	-	-	0.0135 (20/0)	0.3117 (2)	0.1948 (1)	**0.3294 (2/3.9)**
	NMI	-	-	-	-	0.4516 (19/0.35)	0.506 (2)	0.6945 (1)	**0.8158 (1/1.4)**
	Acc	-	-	-	-	0.3333 (20/0)	0.3147 (2)	0.4257 (1)	**0.5889 (1/1.4)**
Leuk	ARI	0.8741 (6/2.42)	-	0.8809 (13)	0.8437 (8)	0.8459 (23/0.35)	0.5613 (3)	0.8809 (3)	**0.9186 (2/2.3)**
	FM	0.9578 (6/2.42)	-	0.9568 (13)	0.942 (8)	0.1959 (23/0.35)	0.7916 (3)	0.9568 (3)	**0.9714 (4/1.4)**
	NMI	0.942 (5/2.28)	-	0.9337 (7)	0.9392 (2)	0.8078 (23/0.35)	0.6358 (3)	0.8684 (2)	**1 (1/1.4)**
	Acc	0.9844 (5/2.28)	-	0.9818 (7)	0.9833 (2)	0.9444 (23/0.35)	0.7917 (3)	0.9583 (3)	**1 (1/1.4)**
Libras	ARI	-	-	-	0.106 (2)	0.0795 (29/0.5)	0.1565 (2)	0.2422 (2)	**0.2673 (2/1.2)**
	FM	-	-	-	0.2825 (2)	0.013 (38/0.15)	0.316 (1)	0.3689 (2)	**0.3806 (2/2.7)**
	NMI	-	-	-	0.4404 (2)	0.3536 (31/0.4)	0.4248 (2)	**0.8053 (1)**	0.7238 (1/1)
	Acc	-	-	-	0.3089 (2)	0.5455 (36/0.35)	0.3222 (1)	**0.6364 (1)**	0.5833 (1/1)
Liver	ARI	**0.0397 (5/0.59)**	0.0369 (7/0.12)	0.0282 (19)	0.0364 (7)	0.0327 (176/0.05)	0.0327 (2)	0.0327 (3)	0.0339 (3/1.3)
	FM	**0.5399 (2/0.99)**	0.4105 (7/0.12)	0.5274 (49)	0.5321 (49)	0.1099 (176/0.05)	**0.5399 (2)**	**0.5399 (3)**	**0.5399 (3/1.2)**
	NMI	0.3456 (2/0.14)	0.0166 (2/0.16)	0.0066 (20)	0.0066 (20)	0.0057 (176/0.05)	0.0057 (2)	0.0361 (2)	**1 (1/1.4)**
	Acc	0.75 (2/0.14)	0.0166 (2/0.16)	0.636 (49)	0.6431 (5)	0.7788 (176/0.05)	0.7564 (1)	0.6667 (1)	**1 (1/1.4)**
Parkinsons	ARI	0.0545 (5/1.1)	-	0.1157 (7)	0.0434 (4)	-0.0197 (73/0.1)	**0.178 (98)**	0.0977 (4)	0.1762 (5/1.2)
	FM	0.4723 (4/1.13)	-	0.568 (7)	0.5118 (4)	0.2832 (73/0.1)	**0.6917 (98)**	0.5377 (4)	0.5582 (4/1.5)
	NMI	**1 (2/0.78)**	-	0.1894 (7)	0.0434 (4)	0.0255 (61/0.05)	0.2322 (73)	0.2943 (1)	**1 (1/3.9)**
	Acc	**1 (2/0.78)**	-	0.8168 (7)	0.7579 (4)	0.5 (54/0)	0.7436 (1)	0.7692 (4)	**1 (1/1.7)**
Seeds	ARI	0.3019 (5/0.43)	-	-	-	0.0147 (34/0.05)	**0.6228 (53)**	0.4235 (3)	0.4924 (3/2.7)
	FM	0.4877 (5/0.43)	-	-	-	0.1027 (34/0.05)	**0.8594 (52)**	0.7076 (3)	0.8568 (8/1.1)
	NMI	0.7612 (2/0.17)	-	-	-	0.098 (34/0.05)	0.6079 (52)	0.821 (1)	**1 (1/1.5)**
	Acc	0.7522 (12/0.49)	-	-	-	0.3665 (34/0.05)	0.8571 (52)	0.9333 (1)	**1 (1/1.5)**
Soya	ARI	**1 (2/3.89)**	-	0.934 (7)	0.9598 (4)	**1 (22/0)**	0.6794 (14)	**1 (3)**	**1 (2/1.9)**
	FM	**1 (2/3.89)**	-	0.9793 (7)	0.9722 (4)	0.3333 (22/0)	0.811 (10)	**1 (3)**	**1 (2/1.4)**
	NMI	**1 (2/3.89)**	-	**1 (5)**	**1 (3)**	**1 (22/0)**	0.7243 (14)	**1 (2)**	**1 (1/1.4)**
	Acc	**1 (2/3.64)**	-	**1 (5)**	**1 (3)**	0.5 (22/0)	0.8298 (14)	**1 (2)**	**1 (1/1.4)**
TeachingA.	ARI	0.0431 (9/2.46)	-	-	0.016 (6)	0.0026 (31/0)	0.0147 (3)	0.0252 (5)	**0.0435 (3/5)**
	FM	0.2976 (19/2.58)	-	-	0.2736 (6)	0.1574 (29/0.4)	0.3989 (2)	0.408 (5)	**0.4687 (9/1.1)**
	NMI	**1 (2/1.63)**	-	-	0.0937 (6)	0.0512 (31/0)	0.0354 (3)	0.1228 (1)	**1 (1/1.7)**
	Acc	**1 (2/1.63)**	-	-	0.4 (6)	0.5455 (29/0.4)	0.4305 (3)	0.5 (2)	**1 (1/1.7)**
Wine	ARI	0.2088 (4/0.81)	-	0.3968 (8)	-	-0.0013 (39/0.1)	**0.4367 (3)**	0.0959 (2)	0.4334 (3/2.2)
	FM	0.369 (4/0.81)	-	**0.7423 (8)**	-	0.2104 (53/0.1)	0.639 (2)	0.3494 (3)	0.6792 (2/4.1)
	NMI	0.7612 (2/0.56)	-	0.7473 (8)	-	0.0365 (53/0.1)	0.5787 (3)	0.7515 (1)	**1 (1/1.9)**
	Acc	0.7368 (5/0.72)	-	0.9016 (8)	-	0.3829 (47/0.1)	0.6517 (2)	0.6923 (1)	**1 (1/1.9)**
Zoo	ARI	0.8702 (2/1.55)	-	-	-	0.7829 (28/0.3)	0.6095 (1)	0.401 (2)	**0.9224 (2/1.9)**
	FM	0.7389 (2/1.55)	-	-	-	0.0756 (28/0.3)	0.5937 (3)	0.463 (2)	**0.764 (3/2.5)**
	NMI	**0.9508 (2/1.55)**	-	-	-	0.8321 (28/0.3)	0.7147 (1)	0.7025 (1)	0.9175 (3/1.2)
	Acc	**0.9551 (2/1.55)**	-	-	-	0.5 (14/0.5)	0.6634 (1)	0.5743 (2)	0.9109 (3/2.5)

given a good set of parameters (for details see Section 5). Hence, there are $10 \times 4 = 40$ results. Under these conditions W-DBSCANR reached the highest overall scores in 30 of the 40 cases with a highest ARI and FM in 7, and highest NMI and Acc in 8 of the 10 data sets. In the only data sets it did not (D31 and R15) the difference in ARI was negligible. We should note that both DBSCANR and DBSCAN have the second best overall score of 22 and 20 respectively

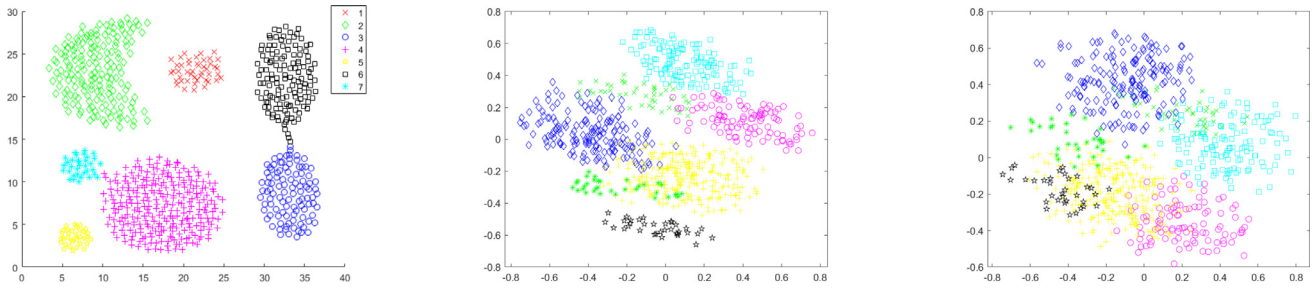
In Table 5, we present our experiments on 15 original real-world data sets using 4 evaluation metrics, leading to $15 \times 4 = 60$ results. In this set of experiments W-DBSCANR presented the highest score in 42 cases of the 60, under the same conditions of our previous experiment. W-DBSCANR outperforms all other algorithms under experiment in 11 data sets with respect to ARI, NMI and FM, and in 9 of the 15 data sets when FM is used. The two exceptions were the Ecoli and the Zoo data sets where W-DBSCANR falls behind in all 4 evaluation measures. DBSCAN and DBSCANR did reach the second best possible overall score, the latter using only a single parameter.

Table 10
The results of our experiments on the real-world data sets with 100% added noise features. We measure cluster recovery using the Adjusted Rand index (ARI), F-Measure (FM), Normalised Mutual Information (NMI) and Accuracy (Acc).

Data set	CVI	DBSCAN (k/ϵ)	OPTICS (k/m_{cr})	ISDBSCAN (k)	RNN-DBSCAN (k)	ADBSCAN (k/p)	DPC-DBFN (k)	DBSCANR (k)	W-DBSCANR (k/β)
Banknote	ARI	0.0192 (14/0.39)	0.008 (3/0.14)	0.0105 (9)	0.0032 (8)	0.0107 (73/0.5)	**0.0443 (2)**	0.0069 (4)	0.0079 (2/1.4)
	FM	0.376 (11/0.43)	-	0.363 (11)	0.3557 (8)	0 (71/0.45)	0.6037 (2)	0.3974 (4)	**0.9585 (14/1.1)**
	NMI	**1 (3/0.16)**	-	0.027 (11)	0.0234 (8)	0.0193 (80/0.35)	0.0523 (2)	0.1028 (2)	**1 (11/1.1)**
	Acc	**1 (3/0.16)**	-	0.5701 (9)	0.5551 (8)	**1 (92/0)**	0.6071 (2)	0.6 (1)	**1 (1/4.8)**
BreastC.	ARI	0.8576 (5/0.9)	-	0.8524 (47)	-	0.8497 (97/0.5)	0.7973 (234)	**0.8661 (15)**	0.8606 (17/1.1)
	FM	0.6414 (22/1.16)	-	0.9558 (49)	-	0.1704 (115/0.05)	0.9414 (234)	**0.9622 (15)**	0.961 (6/1.3)
	NMI	0.5523 (8/1.05)	-	0.8454 (11)	-	0.7501 (97/0.5)	0.6828 (234)	**1 (1)**	**1 (1/2)**
	Acc	0.9897 (6/0.73)	-	0.9789 (11)	-	0.9634 (97/0.5)	0.9471 (230)	**1 (1)**	**1 (1/1.7)**
BreastT.	ARI	0.1091 (2/0.91)	-	0.0657 (5)	-	0.1963 (26/0)	0.1773 (41)	0.0561 (2)	**0.2586 (2/2.2)**
	FM	0.1665 (2/0.91)	-	0.2868 (4)	-	0.0815 (25/0.45)	**0.4335 (2)**	0.2068 (2)	0.2672 (2/1.8)
	NMI	0.4529 (2/0.6)	-	0.4526 (4)	-	0.3334 (26/0)	0.3306 (6)	0.4179 (1)	**0.7443 (1/1.2)**
	Acc	0.4286 (2/0.6)	-	0.5111 (4)	-	0.75 (25/0.45)	0.4623 (2)	0.4167 (1)	**0.7895 (1/1.2)**
Ecoli	ARI	-0.0156 (2/0.41)	-	0.0661 (6)	-	0.1919 (37/0.45)	0.118 (2)	0.0072 (3)	**0.2011 (2/1.4)**
	FM	0.058 (4/0.55)	-	0.162 (6)	-	0.0087 (37/0.45)	0.3059 (2)	0.2212 (3)	**0.315 (3/3)**
	NMI	**0.6648 (2/0.41)**	-	0.279 (6)	-	0.2908 (37/0.35)	0.2303 (2)	0.1846 (3)	0.6475 (1/1.3)
	Acc	0.4706 (2/0.41)	-	0.4601 (6)	-	0.5419 (37/0.35)	0.4494 (2)	0.4107 (3)	**0.5588 (1/1.3)**
Iris	ARI	0.5084 (3/0.49)	0.5543 (6/0.07)	0.3628 (8)	0.4868 (4)	0.5312 (53/0.2)	0.5596 (7)	0.5341 (4)	**0.5715 (4/1.8)**
	FM	0.6097 (7/0.49)	0.5568 (6/0.09)	0.472 (8)	0.5305 (4)	0.6027 (42/0.1)	**0.8059 (3)**	0.7591 (4)	0.7591 (4/4.5)
	NMI	**0.821 (12/0.51)**	0.7584 (8/0.09)	0.6685 (8)	0.7036 (4)	0.6964 (53/0.2)	0.6867 (5)	0.7085 (1)	0.7999 (2/1.2)
	Acc	**0.9375 (12/0.51)**	0.6757 (6/0.09)	0.584 (8)	0.6408 (4)	0.5 (51/0.2)	0.8067 (3)	0.8667 (1)	0.8182 (1/3.9)
Leaf	ARI	-	-	-	-	0.018 (18/0.4)	0.0936 (29)	0.0024 (1)	**0.1368 (4/1)**
	FM	-	-	-	-	0.0104 (18/0.4)	0.2147 (23)	0.1789 (1)	**0.2896 (4/1)**
	NMI	-	-	-	-	0.3637 (18/0.4)	0.432 (23)	0.6801 (1)	**0.6888 (1/2.4)**
	Acc	-	-	-	-	0.1667 (18/0.4)	0.2559 (23)	0.42 (1)	**0.4444 (1/1.1)**
Leuk	ARI	0.679 (9/2.96)	-	0.2866 (5)	0.7929 (2)	0.6123 (29/0.4)	0.7131 (3)	**0.8809 (3)**	**0.8809 (3/1.4)**
	FM	0.7879 (2/2.77)	-	0.4256 (5)	0.908 (2)	0.1704 (29/0.4)	0.9044 (3)	0.9568 (3)	**0.9714 (4/1.1)**
	NMI	**1 (2/2.37)**	-	0.7873 (5)	0.9389 (3)	0.589 (29/0.4)	0.7066 (3)	**1 (1)**	**1 (1/1.3)**
	Acc	**1 (2/2.37)**	-	0.7941 (5)	0.9831 (3)	0.9091 (29/0.4)	0.9028 (3)	**1 (1)**	**1 (1/1.3)**
Libras	ARI	0.0248 (3/3.63)	-	-	-	0.0346 (44/0)	**0.1345 (2)**	0.0838 (2)	0.0976 (2/1.3)
	FM	0.1599 (3/3.63)	-	-	-	0.0094 (44/0)	0.2697 (16)	0.3051 (2)	**0.3219 (2/1.3)**
	NMI	0.5557 (3/3.63)	-	-	-	0.3117 (44/0)	0.4076 (2)	0.6427 (1)	**0.8434 (1/1.1)**
	Acc	0.3435 (3/3.63)	-	-	-	**0.75 (44/0)**	0.3028 (2)	0.4833 (1)	0.6863 (1/4.5)
Liver	ARI	0.0418 (16/0.84)	-	**0.0502 (13)**	0.0299 (12)	0.0283 (145/0.5)	0.0181 (14)	0.0327 (3)	0.0327 (3/1.3)
	FM	0.539 (66/1.09)	-	0.5345 (50)	0.533 (28)	0.1311 (132/0.4)	0.5036 (284)	**0.5399 (3)**	**0.5399 (3/1.1)**
	NMI	0.0166 (7/0.73)	-	0.0114 (13)	0.0069 (6)	0.0145 (145/0.5)	0.0248 (37)	0.274 (1)	**1 (1/1.6)**
	Acc	0.7483 (23/0.8)	-	0.6508 (13)	0.6425 (6)	0.0086 (132/0.5)	0.753 (1)	0.6667 (1)	**1 (1/1.6)**
Parkinsons	ARI	0.0325 (5/1.61)	-	0.0405 (7)	-0.0407 (3)	0.0478 (39/0.5)	0.1009 (98)	0.0407 (1)	**0.1043 (3/1.3)**
	FM	0.4684 (2/1.64)	-	0.4532 (7)	0.4139 (3)	0.2126 (42/0.4)	0.6367 (98)	0.4109 (3)	**0.7013 (4/1.1)**
	NMI	0.1341 (4/1.64)	-	0.1487 (7)	0.0293 (3)	0.0117 (39/0.5)	0.114 (98)	**1 (1)**	0.3163 (1/3.8)
	Acc	0.7712 (5/1.61)	-	0.7778 (7)	0.7249 (3)	0.765 (39/0.5)	0.7282 (2)	**1 (1)**	0.8205 (8/1.1)
Seeds	ARI	0.0804 (12/0.78)	-	0.0337 (6)	-	0.0308 (30/0.35)	**0.3905 (101)**	0.0227 (2)	0.242 (3/1.3)
	FM	0.3825 (22/0.86)	-	0.3269 (6)	-	0 (25/0.4)	0.7091 (54)	0.2569 (2)	**0.8581 (6/1.2)**
	NMI	**0.8656 (6/0.65)**	-	0.1174 (6)	-	0.1123 (30/0.35)	0.4102 (76)	0.4773 (1)	0.8572 (1/4.2)
	Acc	0.9474 (6/0.65)	-	0.4792 (6)	-	**1 (28/0)**	0.7048 (54)	0.6364 (1)	0.9 (1/4.2)
Soya	ARI	0.9598 (2/4.52)	-	**1 (8)**	0.9131 (4)	0.3809 (10/0)	0.7207 (3)	**1 (3)**	**1 (2/1.1)**
	FM	0.9722 (2/4.52)	-	**1 (8)**	0.9646 (4)	0.0985 (9/0.25)	0.875 (7)	**1 (3)**	**1 (2/1.1)**
	NMI	**1 (2/4.11)**	-	**1 (6)**	**1 (3)**	0.6298 (9/0.25)	0.8303 (3)	**1 (3)**	**1 (1/1.6)**
	Acc	**1 (2/4.11)**	-	**1 (6)**	**1 (3)**	**1 (10/0)**	0.8723 (7)	**1 (3)**	**1 (1/1.6)**
TeachingA.	ARI	0.0101 (17/3.26)	-	-	-	0.0032 (31/0)	**0.053 (61)**	0.0146 (2)	0.0508 (3/4)
	FM	0.261 (5/3.12)	-	-	-	0.1234 (30/0.2)	**0.4552 (51)**	0.345 (4)	0.428 (6/1.1)
	NMI	**0.7403 (2/2.61)**	-	-	-	0.0599 (28/0.4)	0.0572 (61)	0.2192 (1)	0.677 (1/2.9)
	Acc	**0.8333 (2/2.61)**	-	-	-	0.5833 (27/0.5)	0.457 (45)	0.4444 (1)	0.6667 (1/1.7)
Wine	ARI	0.0969 (7/1.18)	-	-	-	0.0109 (54/0)	0.2755 (1)	0.2744 (3)	**0.3044 (3/2.1)**
	FM	0.4144 (7/1.18)	-	-	-	0 (39/0.3)	0.5487 (79)	0.625 (3)	**0.6664 (2/4.4)**
	NMI	**1 (2/0.83)**	-	-	-	0.0753 (54/0)	0.3292 (1)	0.3606 (3)	0.8626 (1/4.9)
	Acc	**1 (2/0.83)**	-	-	-	0.4035 (54/0)	0.5843 (2)	0.6461 (3)	0.9333 (1/2.2)
Zoo	ARI	0.8077 (2/1.91)	-	0.2707 (4)	-	0.7558 (21/0)	0.8291 (2)	0.6785 (2)	**0.9606 (2/4.2)**
	FM	0.6954 (2/1.91)	-	0.4045 (4)	-	0.085 (21/0)	0.6591 (2)	0.7016 (2)	**0.77 (3/2.9)**
	NMI	**0.9533 (2/1.75)**	-	0.738 (4)	-	0.8051 (21/0)	0.7882 (2)	0.8261 (2)	0.9162 (1/3.2)
	Acc	**0.9429 (2/1.75)**	-	0.6441 (4)	-	0.8929 (18/0.3)	0.8218 (2)	0.7327 (2)	0.901 (3/2.9)

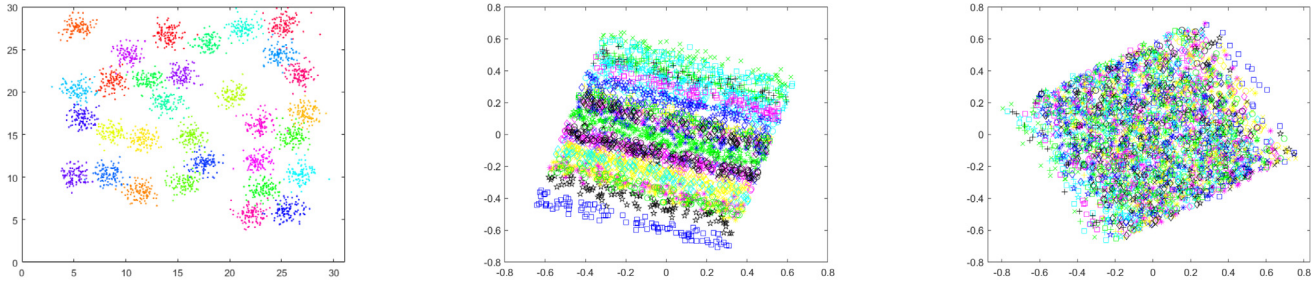
The results we present in this section support our claim that density-based algorithms can benefit from feature weighting. Let us analyse the case of a particular data set a bit further. Figure 3 presents the feature weights obtained by our method, averaged over the three clusters there are in the Iris data set. We can see a higher weight in features 3 and 4 (petal length and petal width) than features 1 and 2 (sepal length and sepal width). These results are very much supported by the literature in partitional clustering algorithms (see for instance [26]).

Given popular density-based algorithms tend to degenerate in higher dimensional space [31], we experiment further with data sets with a much higher number of features than points. Table 6 presents the results on 10 well-known high-dimensional data sets, again under four clustering evaluation indices. W-DBSCANR has the best performance in 21 cases, while DPC-DBFN (the algorithm in second place) has the best results in 8 cases.



(a) Original **(b)** +50% noise **(c)** +100% noise

Fig. 4. Clustering using *true* labels shown on the plane of the first two principal components (a) Aggregation original data set. (b) Aggregation data set with one noise feature. (c) Aggregation data set with two noise features.



(a) Original **(b)** +50% noise **(c)** +100% noise

Fig. 5. Clustering using *true* labels shown on the plane of the first two principal components (a) D31 original data set. (b) D31 data set with one noise feature. (c) D31 data set with two noise features.

6.2. Experiments on data sets with added noise features

In this section, we present the results of our experiments with the data sets to which we added noise features. We find this set of experiments rather important because we can be certain these data sets have irrelevant features. Hence, we are interested in the behaviour of density-based clustering algorithms in this scenario. More specifically, we show the superiority of W-DBSCANR (in terms of cluster recovery) over the other algorithms we experiment with. First, we show the catastrophic effect noise features can have on data sets. Figures 4 and 5 show plots over the first and second principal components of the Aggregation and D31 data sets, respectively. In these, it is quite clear that the original data sets (those with no added noise features) have clear clusters. However, as the number of added noise features increases these clusters become less and less clear. This has a direct effect on cluster recovery as our experiments in this section demonstrate.

Tables 7 and 8 present the results of our experiments on the synthetic data sets adding 1 ($V \times 0.5$) or 2 ($V \times 1$) noise features respectively, since these data sets are two dimensional. In this set of experiments, W-DBSCANR reached the highest expected ARI in $9 + 5 = 14$ data sets (9 when adding one noise feature, and 5 when adding two noise features), and DBSCAN reached the highest ARI in $2 + 2 = 4$ data sets. Given the presence of noise features in the original data set, unsurprisingly, ISDBSCAN and RNN-DBSCAN could not reach the highest ARI for any of the data sets, and could not recover the *true* number of clusters for most artificial data sets with noise feature under experiment (hence the dashes). We were, of course, happy to see that our W-DBSCANR reached the highest possible score of 1 in most data sets. Overall W-DBSCANR dominates with highest score in $35 + 32 = 67$ of the 80 cases. DBSCANR takes the second place with the highest score in $8 + 8 = 16$ cases.

Tables 9 and 10 present the results on the noise versions of the 15 real-world data sets we experiment with. We use the same noise model of before, adding 50% and 100% noise features to each data set (for details see Table 3). W-DBSCANR had results higher by about 0.15 in average (that is an increase of about 27%) when compared to the second best performing algorithm, DBSCAN. W-DBSCANR also reached the highest overall score in $42 + 35 = 77$ of the $60 + 60 = 120$ cases. The result for DBSCAN (the second best algorithm) was $15 + 16 = 31$. However, the total average cluster recovery of DBSCAN across all 15 data sets and 4 measures is only 1% higher than the proposed DBSCANR. Given the latter has only one parameter to be optimised (DBSCAN has two), we are tempted to claim DBSCANR is still rather competitive. Notice that the OPTICS, ISDBSCAN and RNN-DBSCAN ceased to find the *true* number of clusters in some of the data sets and therefore we were forced to put dashes under score and parameters.

7. Conclusion and future work

Feature selection has a long history in the machine learning community. However, even among relevant features there may be different degrees of relevance. With this in mind this paper introduces, perhaps for the first time, the use of feature weights to density-based clustering algorithms. Our method, W-DBSCANR, is capable of generating a set of weights modeling the degree of relevance of features. In fact, it goes a step further by allowing the intuitive idea that a given feature may have different degrees of relevance at different clusters. Clearly, as a clustering method it does the above without requiring labelled samples.

Our experiments clearly demonstrate that W-DBSCANR outperforms other popular and new density-based clustering algorithms (for details see Section 6). We have demonstrated this is the case on a number of data sets with and without added noise features, high-dimensional or not, real-world and synthetic. Our evaluation made use of four measures, these being the Adjusted Rand Index, F-Measure, Normalised Mutual Information, and the usual classification Accuracy. However, this is not to say our algorithm has

no limitations. For instance, W-DBSCANR has two parameters (the same number DBSCAN and most others have). One of these, β, helps define how much higher the weight of a compact feature should be in comparison to features that are less compact. That is, with a high β the standard deviation of the weights of a particular feature within a cluster are lower than with a low β. Although β seems somewhat stable (its optimal value is usually between 1.1 and 2.5) it would be better to have a method to estimate it. The same can be said for the other parameter, that is k, the number of nearest neighbours.

Another limitation with our algorithm (and with the vast majority of feature weighting algorithms in partitional clustering [7]) is that feature weights are calculated in isolation. This is problematic when relevance is not found at a particular feature but instead in a group of features. We envisage that it should be possible to deal with this problem by grouping features (rather than points) in the data pre-processing stage. Of course, research is needed to find the exact way this should be done. The third main limitation of our algorithm is that feature weights are always used in distance calculations, even if the weight itself is negligible. It may be of benefit to perform some level of feature selection before applying our algorithm, or any other feature weighting method. Our future work will address the limitations above.

Declaration of Competing Interest

There is no conflict of interest.

Data availability

Data will be made available on request.

References

[1] A.K. Jain, Data clustering: 50 years beyond k-means, Pattern Recognit. Lett. 31 (8) (2010) 651–666.
[2] B. Mirkin, Clustering: A Data Recovery Approach, Chapman and Hall/CRC, 2016.
[3] F. Murtagh, P. Contreras, Algorithms for hierarchical clustering: an overview, Wiley Interdiscip. Rev. Data MiningKnowl. Discov. 2 (1) (2012) 86–97.
[4] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: Kdd, Vol. 96, 1996, pp. 226–231.
[5] E. Schubert, J. Sander, M. Ester, H.P. Kriegel, X. Xu, Dbscan revisited, revisited: why and how you should (still) use DBSCAN, ACM Trans. Database Syst. (TODS) 42 (3) (2017) 1–21.
[6] I. Niño Adan, D. Manjarres, I. Landa-Torres, E. Portillo, Feature weighting methods: a review, Expert Syst. Appl. 184 (2021) 115424.
[7] R.C. De Amorim, A survey on feature weighting based k-means algorithms, J. Classif. 33 (2) (2016) 210–242.
[8] E. Hancer, B. Xue, M. Zhang, A survey on feature selection approaches for clustering, Artif. Intell. Rev. 53 (6) (2020) 4519–4545.
[9] H.-P. Kriegel, P. Kröger, J. Sander, A. Zimek, Density-based clustering, Wiley Interdiscip. Rev. Data MiningKnowl. Discov. 1 (3) (2011) 231–240.
[10] J. MacQueen, 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, 1967, pp. 281–297. Oakland, CA, USA.
[11] M. Zampieri, R.C.d. Amorim, Between sound and spelling: combining phonetics and clustering algorithms to improve target word recovery, in: International Conference on Natural Language Processing, Springer, 2014, pp. 438–449.
[12] M. Ankerst, M.M. Breunig, H.-P. Kriegel, J. Sander, Optics: ordering points to identify the clustering structure, in: ACM Sigmod record, Vol. 28, ACM, 1999, pp. 49–60.
[13] P. Berkhin, A survey of clustering data mining techniques, in: Grouping Multidimensional Data, Springer, 2006, pp. 25–71.
[14] C. Cassisi, A. Ferro, R. Giugno, G. Pigola, A. Pulvirenti, Enhancing density-based clustering: parameter reduction and outlier detection, Inf. Syst. 38 (3) (2013) 317–330.
[15] F. Korn, S. Muthukrishnan, Influence sets based on reverse nearest neighbor queries, in: ACM Sigmod Record, Vol. 29, ACM, 2000, pp. 201–212.
[16] A.C. Bryant, K.J. Cios, RNN-DBSCAN: a density-based clustering algorithm using reverse nearest neighbor density estimates, IEEE Trans. Knowl. Data Eng. (2017).
[17] H. Li, X. Liu, T. Li, R. Gan, A novel density-based clustering algorithm using nearest neighbor graph, Pattern Recognit. 102 (2020) 107206.
[18] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.
[19] Y. Ren, N. Wang, M. Li, Z. Xu, Deep density-based image clustering, Knowl. Based Syst. 197 (2020) 105841.
[20] J. Zheng, S. Wang, D. Li, B. Zhang, Personalized recommendation based on hierarchical interest overlapping community, Inf. Sci. 479 (2019) 55–75.
[21] X. Xu, S. Ding, H. Xu, H. Liao, Y. Xue, A feasible density peaks clustering algorithm with a merging strategy, Soft Comput. 23 (13) (2019) 5171–5183.
[22] L. Bai, X. Cheng, J. Liang, H. Shen, Y. Guo, Fast density clustering strategies based on the k-means algorithm, Pattern Recognit. 71 (2017) 375–386.
[23] A. Lotfi, P. Moradi, H. Beigy, Density peaks clustering based on density backbone and fuzzy neighborhood, Pattern Recognit. 107 (2020) 107449.
[24] H. Liu, L. Yu, Toward integrating feature selection algorithms for classification and clustering, IEEE Trans. Knowl. Data Eng. 17 (4) (2005) 491–502.
[25] X. Chen, Y. Ye, X. Xu, J.Z. Huang, A feature group weighting method for subspace clustering of high-dimensional data, Pattern Recognit. 45 (1) (2012) 434–446.
[26] R.C. De Amorim, B. Mirkin, Minkowski metric, feature weighting and anomalous cluster initializing in k-means clustering, Pattern Recognit. 45 (3) (2012) 1061–1075.
[27] R.C. de Amorim, Unsupervised feature selection for large data sets, Pattern Recognit. Lett. 128 (2019) 183–189.
[28] Y. Chen, L. Zhou, Y. Tang, J.P. Singh, N. Bouguila, C. Wang, H. Wang, J. Du, Fast neighbor search by using revised kd tree, Inf. Sci. 472 (2019) 145–162.
[29] Y. Chen, L. Zhou, N. Bouguila, B. Zhong, F. Wu, Z. Lei, J. Du, H. Li, Semi-convex hull tree: fast nearest neighbor queries for large scale data on GPUs, in: 2018 IEEE International Conference on Data Mining (ICDM), IEEE, 2018, pp. 911–916.
[30] J. Wang, N. Wang, Y. Jia, J. Li, G. Zeng, H. Zha, X.-S. Hua, Trinary-projection trees for approximate nearest neighbor search, IEEE Trans. Pattern Anal. Mach. Intell. 36 (2) (2013) 388–403.
[31] Y. Chen, L. Zhou, S. Pei, Z. Yu, Y. Chen, X. Liu, J. Du, N. Xiong, KNN-BLOCK DBSCAN: fast clustering for large-scale data, IEEE Trans. Syst. Man Cybern.Syst. 51 (6) (2019) 3939–3953.
[32] R.A. Brown, Building a balanced kd tree in O(kn log n) time, arXiv preprint arXiv:1410.5420(2014).
[33] D. Dua, C. Graff, UCI machine learning repository, 2017. http://archive.ics.uci.edu/ml.
[34] J. Li, K. Cheng, S. Wang, F. Morstatter, R.P. Trevino, J. Tang, H. Liu, Feature selection: a data perspective, ACM Comput. Surv. (CSUR) 50 (6) (2017) 1–45.
[35] P. Fränti, et al., Clustering datasets, 2015. http://cs.uef.fi/sipu/datasets/.
[36] J. Sander, X. Qin, Z. Lu, N. Niu, A. Kovarsky, Automatic extraction of clusters from hierarchical clustering representations, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2003, pp. 75–87.
[37] L. Hubert, P. Arabie, Comparing partitions, J. Classif. 2 (1) (1985) 193–218.
[38] A. Strehl, J. Ghosh, Cluster ensembles—a knowledge reuse framework for combining multiple partitions, J. Mach. Learn. Res. 3 (Dec) (2002) 583–617.
[39] M. Wu, B. Schölkopf, A local learning approach for clustering, Adv. Neural Inf. Process. Syst. 19 (2006).

Stiphen Chowdhury holds a first class BSc (Hons) in Business Information Systems from the University of East London, as well as a MSc in Software Engineering with distinction, and a PhD in Computer Science (2021) both from the University of Hertfordshire. He is currently a Lecturer at the Anglia Ruskin University.

Na Helian has 20 years research experience around UK, Singapore, Japan and China in data mining, grid computing, data storage, etc. She has 47 journal papers (including ACM Publications, IEEE Transactions, etc) and 26 conference proceeding papers published in the above areas. Her research background on data storage contributed to the following research funding: EPSRC project (GR/S96487): Grid-Oriented Storage (GOS): Next Generation Data Storage System Architecture for the Grid Computing Era.

Renato Cordeiro de Amorim is a Senior Lecturer in Computer Science and AI at the University of Essex. He has published a number of papers introducing novel methods following the unsupervised and semi-supervised learning frameworks, with applications in fields such as security, biosignal processing and general data mining. His research is funded by the Royal Society and Innovate UK.