[Array 15 (2022) 100232](https://doi.org/10.1016/j.array.2022.100232)



Contents lists available at [ScienceDirect](http://www.elsevier.com/locate/array)

Array

journal homepage: [www.elsevier.com/locate/array](http://www.elsevier.com/locate/array)

An improved probability propagation algorithm for density peak clustering based on natural nearest neighborhood

Wendi Zuo [a](#_bookmark0), Xinmin Hou [a](#_bookmark0),[b](#_bookmark1),[c](#_bookmark2),[∗](#_bookmark3)

a *School of Data Science, University of Science and Technology of China, Hefei, Anhui 230026, China*

b *School of Mathematical Sciences, University of Science and Technology of China, Hefei, Anhui 230026, China*

c *CAS Key Laboratory of Wu Wen-Tsun Mathematics, University of Science and Technology of China, Hefei, Anhui 230026, China*

A R T I C L E I N F O A B S T R A C T

*Keywords:* Clustering Density peaks

Natural nearest neighborhood Probability propagation

Clustering by fast search and find of density peaks (DPC) (Since, 2014) has been proven to be a promising clustering approach that efficiently discovers the centers of clusters by finding the density peaks. The accuracy

of DPC depends on the cutoff distance (*𝑑𝑐* ), the cluster number (*𝑘*) and the selection of the centers of clusters.

Moreover, the final allocation strategy is sensitive and has poor fault tolerance. The shortcomings above

make the algorithm sensitive to parameters and only applicable for some specific datasets. To overcome the limitations of DPC, this paper presents an improved probability propagation algorithm for density peak clustering based on the natural nearest neighborhood (DPC-PPNNN). By introducing the idea of natural nearest neighborhood and probability propagation, DPC-PPNNN realizes the nonparametric clustering process and makes the algorithm applicable for more complex datasets. In experiments on several datasets, DPC-PPNNN is shown to outperform DPC, K-means and DBSCAN.

# Introduction

Clustering, also known as unsupervised classification, aims to divide datasets into subsets or clusters according to the similarity measure of the data sample (physical or abstract) such that the data samples within the subset or cluster have a high degree of similarity and that the data samples belonging to different subsets or clusters have a high degree of dissimilarity [[1](#_bookmark42)]. Currently, cluster analysis plays an important role in many fields such as social sciences, biology, pattern recognition, information retrieval and so on [[2](#_bookmark43)]. It is so useful in machine learning and data mining that many researchers have paid much attention to it. Over the past few decades, a number of excellent clustering algorithms have been developed for different types of applications. Typical algorithms include K-means [[3](#_bookmark44)] and K-medoids [[4](#_bookmark45)] based on partitioning, CURE [[5](#_bookmark46)] and BIRCH [[6](#_bookmark47)] based on hierarchy, DBSCAN [[7](#_bookmark48)] and OPTICS [[8](#_bookmark49)] based on density, WaveCluster [[9](#_bookmark50)] and STING [[10](#_bookmark51)] based on grids and statistical clustering [[11](#_bookmark52)] based on models.

In 2014, a density-based clustering algorithm DPC was given by Rodriguez and Laio [[12](#_bookmark53)] (Clustering by fast search and find of density peaks, Science, 344 (2014) 1492). The main idea of the algorithm DPC

is as follows: For each data point *𝑖*, we compute two quantities: its

local density *𝜌𝑖* and its distance *𝛿𝑖* from points of higher density. By mapping all the data points to the decision graph which takes *𝜌* and *𝛿*

as the two axes, we can recognize density peak points (cluster centers)

as points for which the values of *𝜌𝑖* and *𝛿𝑖* are anomalously large. After

the cluster centers have been found, each remaining point is assigned

to the same cluster as its nearest neighbor of higher density. The DPC algorithm is simple and efficient, and it can quickly find the high density peak points (cluster centers) without iteratively calculating the objective function. Moreover, it is suitable for cluster analysis on large- scale data. Although the DPC algorithm has obvious advantages over other clustering algorithms, it also has some shortcomings: the accuracy

of DPC depends on the cutoff distance (*𝑑𝑐* ), the cluster number (*𝑘*) and

the selection of the centers of clusters. Moreover, the final allocation

strategy is sensitive and has poor fault tolerance. Last, the algorithm has its basis on the assumptions that cluster centers are surrounded by neighbors with lower local density and that they are at a relatively large distance from any point with higher local density.

To avoid the deficiencies of DPC, in this paper, we present an improved probability propagation algorithm for density peak clustering based on natural nearest neighborhood (DPC-PPNNN). By introducing the concept of natural nearest neighborhood, we can avoid setting

*𝑑𝑐* manually. By calculating *𝛾* = *𝜌* × *𝛿* and *𝜃* = *𝜌* , we can select

*𝛿*

cluster centers automatically. Finally, the clustering algorithm based

on probability propagation can help us allocate all the remaining data points. By doing all these, DPC-PPNNN is suitable for more complex datasets and can distinguish two clusters that are close to each other.

∗ Corresponding author at: School of Data Science, University of Science and Technology of China, Hefei, Anhui 230026, China.

*E-mail addresses:* [zuowendi@mail.ustc.edu.cn](mailto:zuowendi@mail.ustc.edu.cn) (W. Zuo), [xmhou@ustc.edu.cn](mailto:xmhou@ustc.edu.cn) (X. Hou).

<https://doi.org/10.1016/j.array.2022.100232> Received 10 July 2022; Accepted 12 July 2022

Available online 18 July 2022

2590-0056/© 2022 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY-NC-ND license ([http://creativecommons.org/licenses/by-](http://creativecommons.org/licenses/by-nc-nd/4.0/) [nc-nd/4.0/](http://creativecommons.org/licenses/by-nc-nd/4.0/)).

The rest of this paper is organized as follows. In Section [2](#_bookmark6), we intro- duce the research progress related to the DPC algorithm. In Section [3](#_bookmark4), we briefly review the basic definitions and processes of the traditional DPC algorithm and reveal some problems within them. In Section [4.1](#_bookmark16), we introduce the basic concept and the algorithm of the natural nearest neighborhood. In Section [4.2](#_bookmark18), we show the improvement on the local density and the method of selecting cluster centers automatically. In Section [4.3](#_bookmark22), we propose the allocation algorithm based on probability propagation. In Section [5](#_bookmark27), we compare the DPC-PPNNN algorithm with other classical clustering algorithms using various synthetic and real-world datasets. In Section [6](#_bookmark40), we summarize the advantages and disadvantages of the DPC-PPNNN algorithm and point out the direction of our future research.

# DPC algorithm and analysis

* 1. *DPC algorithm*

The DPC algorithm presented by Rodriguez and Laio [[12](#_bookmark53)] has its ba- sis on the assumption that cluster centers are surrounded by neighbors with lower local density and that they are at a relatively larger distance from any point with higher local density. There are two quantities to

describe each data point *𝑖*: its local density *𝜌𝑖* and its distance *𝛿𝑖* from

points of higher density.

density for a data point *𝑖*: the cutoff distance method and the kernel The DPC algorithm provides two methods for calculating the local distance method. For a data point *𝑖*, its local density *𝜌𝑖* is defined as

# Related works

*𝜌𝑖* =

∑

*𝑖*≠*𝑗*

*𝜒* (*𝑑𝑖𝑗* − *𝑑𝑐* )*, 𝜒* (*𝑥*) =

{1*, 𝑥 <* 0

(1)

0*, 𝑥 >* 0

with the cutoff distance method, or

DPC has been proven to be an effective clustering strategy but it also has many limitations such as the arbitrary selection of density estimation metrics, clustering centers and the risk of error propagation.

*𝜌𝑖* =

∑

*𝑖*≠*𝑗*

exp

[ ( )2]

– *𝑖𝑗*

*𝑑*

*𝑑𝑐*

(2)

Over the past few years, many optimized variants of DPC have been proposed considering the following aspects:

The first aspect is improving the density measure of the DPC al- gorithm. Du, Ding and Jia [[13](#_bookmark54)] proposed DPC-KNN, which introduces the concept of K-nearest neighbors (KNN) to DPC and provides another option for computing the local density. They also employ PCA to reduce the dimensionality of data. However, the method still suffers from the limitations of DPC because it applies the same procedure in determining

the cluster centers and assigning the rest of the points. Liu, Wang, and

with the kernel distance method, where *𝑑𝑖𝑗* is the Euclidean distance

between data points *𝑖* and *𝑗*, *𝑑𝑐* (*>* 0), the cutoff distance, is the radius

the local density *𝜌𝑖* is positively correlated with the number of points of a point for scanning its neighborhood, which is set by the user. Thus, with a distance from *𝑖* less than *𝑑𝑐* . The most obvious difference be- tween the two methods is that for Eq. ([1](#_bookmark5)), *𝜌𝑖* is a discrete value, whereas for Eq. ([2](#_bookmark7)), it is a continuous value. However, for both methods, *𝜌𝑖* is sensitive to *𝑑𝑐* .

Subsequently, *𝛿𝑖* in DPC is defined as

Yu [[2](#_bookmark43)] proposed SNN-DPC, which estimates the local density based on the idea of shared nearest neighbor similarity. However, it still

*𝛿* = min

*𝑗*∶*𝜌𝑗 >𝜌𝑖*

*𝑖*

(*𝑑* ) (3)

suffers from the limitation of manually selecting the number of cluster centers. Mehmood et al. [[14](#_bookmark55)] proposed CFSFDP-HD, which applies a nonparametric density estimator (Kernel Density Estimation, KDE) to

*𝑖𝑗*

eliminate the reliance of DPC on the cut-off distance *𝑑𝑐* .

The second aspect is to automatically recognize the numbers of clusters and cluster centers. Liang and Chen [[15](#_bookmark56)] proposed 3DC, which introduces a divide-and-conquer strategy to determine the ideal number of clusters. However, it ignores the local structure of the datasets which may cause missing clusters. Xu, Wang and Deng [[16](#_bookmark57)] proposed DenPEHC, which could automatically detect all possible centers and build a hierarchy presentation for the dataset. Nevertheless, both 3DC and DenPEHC will aggravate the propagation of errors due to the hier- archical clustering strategy. Li, Ge, and Su [[17](#_bookmark58)] proposed an automatic clustering algorithm for determining the density of clustering centers. In this algorithm, it is considered that if the shortest distance between a potential cluster center and a known cluster center is less than the

cutoff distance *𝑑𝑐* , then the potential cluster center is a redundant

center. Otherwise, it is regarded as the actual center of another cluster.

The third aspect is to improve the assignment strategy to reduce error propagation. In most of the DPC variants, the idea of K-nearest neighbors is hybridized in the aggregation strategies. For instance, Zhou et al. [[18](#_bookmark59)] constructed the veins of clusters by connecting pairs with the highest similarity from the high-density regions to the cluster borders. The rest of the points are then assigned to the nearest veins. Xie et al. [[19](#_bookmark60)] proposed a two-step procedure for label propagation. The first strategy assigns non-outliers based on the search of the K- nearest neighbors starting from the density peaks. The second strategy assigns the other points using the fuzzy KNN technique. Geng et al. [[20](#_bookmark61)] proposed a KNN graph-based label propagation strategy to assign the remaining points. Liu, Wang and Yu [[2](#_bookmark43)] introduced a two-step alloca- tion method based on inevitably and possibly subordinate allocation of the noncenter points.

For the data point *𝑖* that has the highest local density *𝜌𝑖*, *𝛿𝑖* is conven-

tionally defined as

*𝛿𝑖* = max (*𝑑𝑖𝑗* ) (4)

*𝑗*

As shown in Eq. ([3](#_bookmark8)), *𝛿𝑖* is the minimum distance between point *𝑖* and points *𝑗* with local densities *𝜌𝑗 > 𝜌𝑖*.

After calculating the two quantities: *𝜌* and *𝛿*, we can recognize

of *𝜌𝑖* and *𝛿𝑖* are anomalously large by mapping all the data points to density peak points (cluster centers) as points for which the values the decision graph which takes *𝜌* and *𝛿* as the two axes. However,

sometimes we cannot select cluster centers from the decision graph correctly because they are too close to each other. Instead, we can sort

all the data points by *𝛾* defined in Eq. ([5](#_bookmark9)) and choose the largest *𝑘* (the

cluster number) data points as the cluster centers.

*𝛾𝑖* = *𝜌𝑖* × *𝛿𝑖* (5)

Since the cluster centers have been found, each remaining point is assigned to the same cluster as its nearest neighbor of higher density.

* 1. *Analysis*

Although the experimental results obtained with DPC have shown that DPC performs well in many instances, the following drawbacks are obvious.

distance (*𝑑𝑐* ). As shown in [Fig.](#_bookmark11) [1](#_bookmark11), there is a two-moon dataset. The red First, the accuracy of DPC depends on the setting of the cutoff

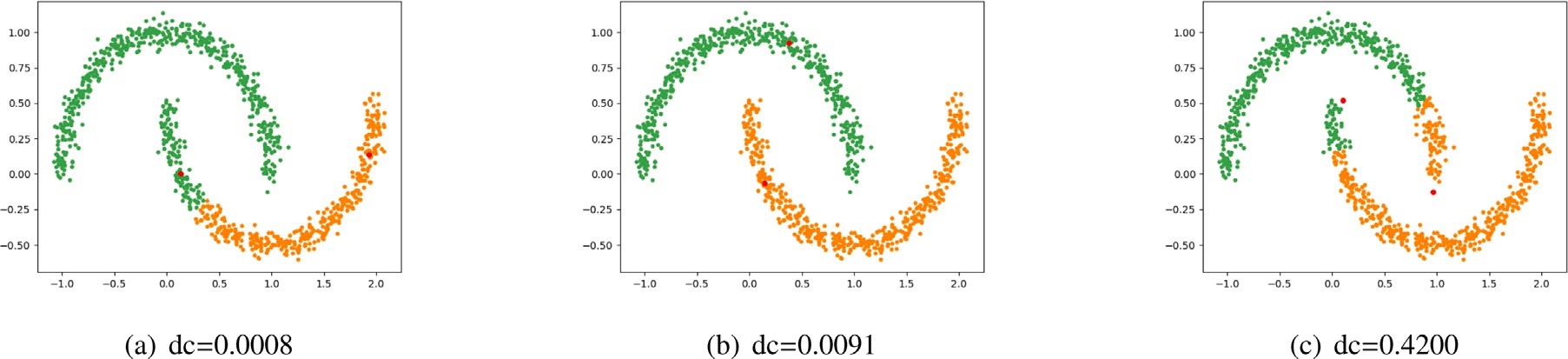
pictures in this article. When we change the value of *𝑑𝑐* , the clustering points are the cluster centers we select and the same for all the other

result of DPC can be greatly affected even for this simple dataset.

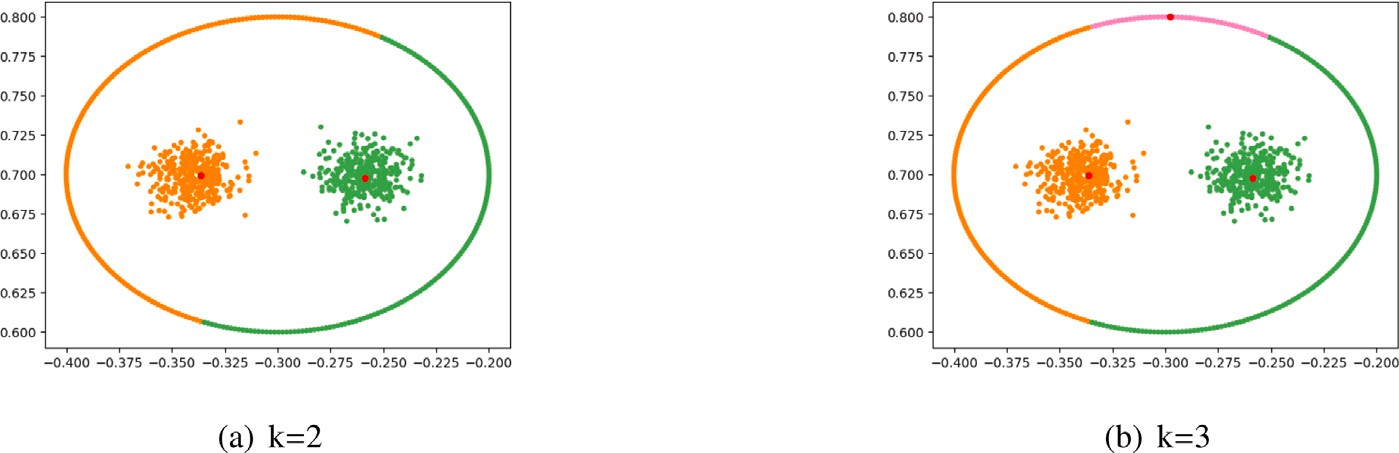
Second, the cluster number (*𝑘*) is difficult to determine. Actually,

time, not to mention choosing an ideal cluster number (*𝑘*) for DPC. we have little idea about the distribution of the dataset most of the

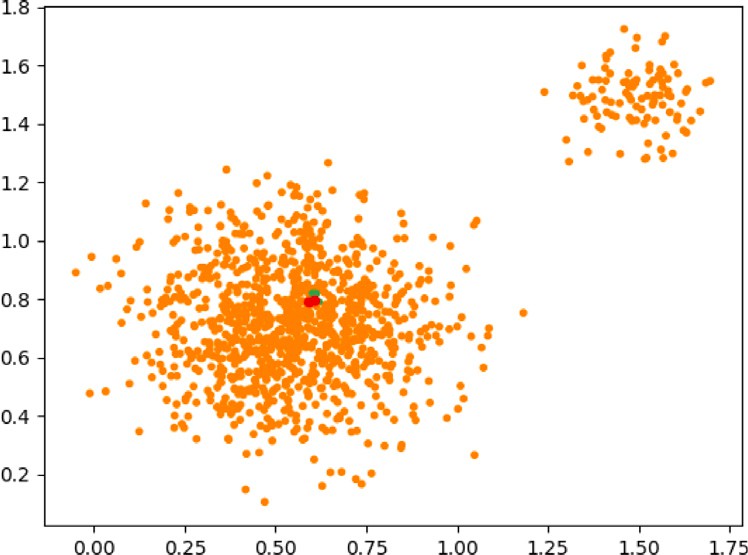
[Fig.](#_bookmark12) [2](#_bookmark12) shows the results of the traditional DPC algorithm on the Donut3



**/ig. 1.** Results of the traditional DPC algorithm on the two-moon dataset. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



**/ig. 2.** Results of the traditional DPC algorithm on the Donut3 dataset.



**/ig. 3.** Results of the traditional DPC algorithm on the dataset with two Gaussian clusters with very different densities.

dataset. In [Fig.](#_bookmark13) [2(a)](#_bookmark13), we recognize the outer ring as noise and let the cluster number be 2. However, clearly, the result cannot satisfy us. In [Fig.](#_bookmark15) [2(b)](#_bookmark15), we recognize the outer ring as a cluster and let the cluster number be 3. Even when we choose the correct cluster center in the outer ring, we cannot obtain the ideal result.

Third, it is difficult to recognize the centers of clusters. As shown

setting *𝑑𝑐* manually. By calculating *𝛾* = *𝜌* × *𝛿* and *𝜃* = *𝜌* , we can introducing the concept of natural nearest neighbors, we can avoid

select cluster centers automatically. Finally, the clustering algorithm based on probability propagation can help us allocate all the remaining data points. By doing all these, DPC-PPNNN can be suitable for more complex datasets and distinguish two clusters that are close to each other.

*𝛿*

# DPC-PPNNN

* 1. *Natural nearest neighborhood*

traditional *𝑘*-nearest neighbors. The NNN is a scale-free concept and The natural nearest neighborhood (NNN) [[21](#_bookmark62)] is very different from

does not require parameters in the selection of neighbors. The size of NNN of every data point may be different according to the distribution of the dataset. The following is a precise description of the natural neighborhood method through the definition of the relevant concepts.

Let *𝑋* = {*𝑥*1*, 𝑥*2*,* … *, 𝑥𝑛*} *⊂* R*𝑑* be a dataset. We use the Euclidean

distance *𝑑*(*𝑥𝑖, 𝑥𝑗* ) to measure the distance between points *𝑥𝑖* and *𝑥𝑗* . For a given integer *𝑘 >* 0, let *𝑛𝑛𝑘*(*𝑥𝑖*) be the *𝑘*th nearest neighbor of *𝑥𝑖* in

*𝑋* ⧵{*𝑥𝑖*}. Define the *𝑘*-*nearest neighborhood* of *𝑥𝑖* (*𝑁𝑁𝑘*(*𝑥𝑖*)) as the set of

*𝑘* nearest neighbors of *𝑥𝑖*, i.e.

*𝑗*=1

in [Fig.](#_bookmark14) [3](#_bookmark14), there is a dataset that has two clusters based on a Gaussian

*𝑁𝑁𝑘*(*𝑥𝑖*) = ∪*𝑘*

{*𝑛𝑛𝑗* (*𝑥𝑖*)}*.*

distribution with one of the densities much larger than the other. Even

The *𝑘*-*reverse nearest neighborhood* of *𝑥* (*𝑅𝑁𝑁* (*𝑥* )) is the set of points

when we choose the two data points with the largest *𝛾𝑖* as the cluster

centers, both of them will belong to the same cluster, which has a

*𝑖*

*𝑥𝑗* ∈ *𝑋* ⧵ {*𝑥𝑖*} with *𝑥𝑖* ∈ *𝑁𝑁𝑘*(*𝑥𝑗* ), i.e.

*𝑘 𝑖*

larger density. The reason for this phenomenon is that the difference in densities between the two clusters is so large that we can hardly recognize the centers of clusters regardless of no matter by the decision

graph or by sorting all the data points by *𝛾*.

Last, the final allocation strategy is sensitive and has poor fault

tolerance. As shown in [Fig.](#_bookmark15) [2(b)](#_bookmark15), even when we choose the three cluster centers correctly, we cannot obtain the ideal result. There are still some data points that cannot be allocated to the correct cluster.

To overcome the deficiencies mentioned above, in this article, we present an improved probability propagation algorithm for density peak clustering based on natural nearest neighbors (DPC-PPNNN). By

*𝑅𝑁𝑁𝑘*(*𝑥𝑖*) = {*𝑥𝑗* ∈ *𝑋* ⧵ {*𝑥𝑖*} | *𝑥𝑖* ∈ *𝑁𝑁𝑘*(*𝑥𝑗* )}*.*

The *𝑘*-natural nearest neighborhood of *𝑥𝑖* (*𝑁𝑁𝑁𝑘*(*𝑥𝑖*)) is defined as

*𝑁𝑁𝑁𝑘*(*𝑥𝑖*) = *𝑁𝑁𝑘*(*𝑥𝑖*) ∩ *𝑅𝑁𝑁𝑘*(*𝑥𝑖*)*.*

Clearly, *𝑁𝑁𝑁**𝑛*−1(*𝑥𝑖*) = *𝑋* ⧵ {*𝑥𝑖*} ≠ ∅ for any *𝑥𝑖* ∈ *𝑋*.

**Definition 1** (*Natural Nearest Neighborhood (𝑁𝑁𝑁 )*)**.** The least integer

*𝜆* with *𝑁𝑁𝑁𝜆*(*𝑥𝑖*) ≠ ∅ for every *𝑥𝑖* ∈ *𝑋* is called the natural eigenvalue of *𝑋*. We define *𝑁𝑁𝑁𝜆*(*𝑥𝑖*) as the natural nearest neighborhood of *𝑥𝑖*, denoted by *𝑁𝑁𝑁* (*𝑥𝑖*), for every *𝑥𝑖* ∈ *𝑋*,.

Let *𝑁𝑁𝑁𝑘*(*𝑋*) = {*𝑁𝑁𝑁𝑘*(*𝑥𝑖*) | *𝑥𝑖* ∈ *𝑋*}. Note that *𝑁𝑁𝑁𝑘*(*𝑋*) maybe

a multiset. Let

* 1. *Local density estimation and selection of cluster centers*

*𝑁𝑁𝑁* 0(*𝑋*) = {*𝑥𝑗* | *𝑥𝑗*

*𝑘*

∈ *𝑋* with *𝑁𝑁𝑁𝑘*(*𝑥𝑗* ) = ∅}*.*

require parameters in the selection of neighbors. The parameter *𝑑𝑐* in As shown in Section [4.1](#_bookmark16), NNN is a scale-free concept and does not

By [Definition](#_bookmark17) [1](#_bookmark17), the natural eigenvalue *𝜆* represents the number of

iterations in the process of searching the natural nearest neighborhood

*𝑁𝑁𝑁* (*𝑋*) = *𝑁𝑁𝑁𝜆*(*𝑋*). The natural eigenvalue *𝜆* is generally small

based on the NNN. For any data point *𝑥𝑖*, the new local density is DPC can be avoided by changing the definition of the local density

defined as

∑ [ ( *𝑑* )2]

when there is no outlier or noise (the data points make *𝜆* ∈ *𝛺*(ln *𝑛*)). In

the extremal case, *𝜆* can be the largest possible value *𝑛* − 1 if there is

*𝜌𝑖* =

*𝑥𝑗* ∈*𝑁𝑁𝑁* (*𝑥𝑖* )

exp −

*𝑖𝑗*

*𝜎𝑖*

*,* (6)

an outlier *𝑥𝑗* ∉ *𝑁𝑁𝑛*−2(*𝑥𝑖*) for any *𝑥𝑖* ≠ *𝑥𝑗* and *𝑥𝑖* ∈ *𝑋*.

where *𝜎* = max{*𝑑*(*𝑥 , 𝑥* ) | *𝑥*

∈ *𝑁𝑁𝑁* (*𝑥* )}. We define

To decrease the effect of outliers or noise in dataset *𝑋*, we in- troduce ln *𝑛* + ln *𝜆* as the threshold to control the number of itera-

tions when *𝑁𝑁𝑁* 0(*𝑋*) remains unchanged in its subsequent iteration,

*𝑖*

*𝛿* = min

*𝑗*∶*𝜌𝑗 >𝜌𝑖*

*𝑖*

( )

*𝑖𝑗*

*𝑑*

*𝑖 𝑗 𝑗 𝑖*

*𝑘*

i.e. 0 0

and for data point *𝑖* which has the highest local density *𝜌𝑖*, define

*𝑁𝑁𝑁𝑘* (*𝑋*) = *𝑁𝑁𝑁𝑘*+1(*𝑋*). Formally, we define the logarithmic

natural nearest neighborhood as follows:

*𝛿* = max

*𝑗*

*𝑖*

( )

*𝑖𝑗*

*𝑑*

*,*

least integer *𝜆* with **Definition 2** (*Logarithmic Natural Nearest Neighborhood*)**.** We call the

|{*𝑖* | *𝑖* ≤ *𝜆* − 1 with *𝑁𝑁𝑁* 0(*𝑋*) = *𝑁𝑁𝑁* 0 (*𝑋*)}| ≥ ln *𝑛* + ln *𝜆*

which are the same as in DPC.

Let *𝛾𝑖* = *𝜌𝑖* × *𝛿𝑖* as defined in Eq. ([5](#_bookmark9)). We can choose the candidate

centers based on a standard normal distribution with a one-sided 95%

confidence interval. The candidate centers are defined as

*𝑖 𝑖*+1

the logarithmic natural eigenvalue and define *𝑁𝑁𝑁𝜆*(*𝑥𝑖*) as the loga- rithmic natural nearest neighborhood of *𝑥𝑖*, denoted by *𝑁𝑁𝑁* (*𝑥𝑖*), for

*𝑥𝑖* ∈ *𝑋*.

By [Definition](#_bookmark19) [2](#_bookmark19), the logarithmic natural eigenvalue *𝜆* represents the

nearest neighborhood *𝑁𝑁𝑁* (*𝑋*), which will be bounded by ln *𝑛* + ln *𝜆*. number of iterations in the process of searching the logarithmic natural The logarithmic natural nearest neighborhood *𝑁𝑁𝑁* (*𝑋*) may contain

an empty set. The robust (logarithmic) natural nearest neighborhood

*𝐶𝑎𝑛𝐶𝑒𝑛𝑠* = {*𝑥𝑖* ∈ *𝑋* | *𝛾𝑖 > 𝐸*(*𝛾*) + 1*.*65Var(*𝛾*)}*,* (7)

where *𝐸*(*𝛾*) is the expectation of *𝛾*, Var(*𝛾*) is the variance of *𝛾*, 1.65 is

the 95% quantile of the standard normal distribution.

with low local density *𝜌𝑖* and small distance *𝛿𝑖*, but it may have some By the definition of CanCens, it does not contain the data points points with large *𝜌* and small *𝛿* or small *𝜌* and large *𝛿* which is not

suitable for being cluster centers. To eliminate those data points, we define

searching algorithm proposed in this paper, which is based on the NNN searching algorithm [[21](#_bookmark62)], is shown in Alg. [1](#_bookmark20).

*𝜃* = *𝜌𝑖*

*𝑖 𝛿𝑖*

(8)

**Algorithm 1** (Logarithmic) Natural Nearest Neighborhood Search Algorithm (NNN)

**Input:** A set of points *𝑋* = {*𝑥*1*, 𝑥*2*, ..., 𝑥𝑛*} *⊂* R*𝑑*

**Output:** The (logarithmic) natural eigenvalue *𝜆* = *𝑟* and *𝑁𝑁𝑁* (*𝑥𝑖*) =

*𝑁𝑁𝑁𝑟*(*𝑥𝑖*) for every *𝑥𝑖* ∈ *𝑋*

1: ∀*𝑥𝑖* ∈ *𝑋*, initially set *𝑁𝑁*0(*𝑥𝑖*) = ∅, *𝑅𝑁𝑁*0(*𝑥𝑖*) = ∅, *𝑁𝑁𝑁*0(*𝑥𝑖*) = ∅

2: *𝑟* = 1, flag=0, *𝑇* = 0

3: **while** flag=0 **do**

4: **for** ∀*𝑥𝑖* ∈ *𝑋* **do**

5: calculate *𝑛𝑛𝑟*(*𝑥𝑖*) = *𝑥𝑗*

6: Reset *𝑁𝑁𝑟*(*𝑥𝑖*) = *𝑁𝑁𝑟*−1(*𝑥𝑖*) ∪ {*𝑥𝑗* }

7: Reset *𝑅𝑁𝑁𝑟*(*𝑥𝑗* ) = *𝑅𝑁𝑁𝑟*−1(*𝑥𝑗* ) ∪ {*𝑥𝑖*}

8: Reset *𝑁𝑁𝑁𝑟*(*𝑥𝑖*) = *𝑁𝑁𝑟*(*𝑥𝑖*) ∩ *𝑅𝑁𝑁𝑟*(*𝑥𝑖*)

9: **end for**

10: calculate the set *𝑁𝑁𝑁* 0(*𝑋*)

for every *𝑥𝑖* ∈ *𝐶𝑎𝑛𝑐𝑒𝑛𝑠*. Note that the gap between *𝜌𝑖* and *𝛿𝑖* is large

enough for those data points with large *𝜌* and small *𝛿* or small *𝜌* and large *𝛿*. Therefore, we can select the final centers from Cancens based on standard normal distribution of *𝜃* with a two-sided 95% confidence

interval as shown in Eq. ([9](#_bookmark21)).

*𝐶𝑒𝑛𝑠*(*𝑋*) = {*𝑥𝑖* ∈ *𝐶𝑎𝑛𝐶𝑒𝑛𝑠* | |*𝜃𝑖* − *𝐸*(*𝜃*)| *<* 1*.*96Var(*𝜃*)}*,* (9)

where 1.96 is the 97.5% quantile of the standard normal distribution. In the Probabilistic Propagation Clustering Algorithm provided in Section [4.3](#_bookmark22), one cluster may have more than one center point in

*𝐶𝑒𝑛𝑠*(*𝑋*), so the allocation strategy of DPC cannot be used here. We

centers in *𝐶𝑒𝑛𝑠*(*𝑋*). need a new allocation strategy to cluster all the data points using the

* 1. *Probabilistic propagation clustering algorithm*

0 *𝑟* The idea of the probabilistic propagation clustering algorithm is

11: **if** *𝑁𝑁𝑁𝑟* (*𝑋*) remains unchanged **then**

12: *𝑇* ∶= *𝑇* + 1

13: **end if**

14: **if** *𝑇* ≥ ln *𝑟* + ln *𝑛* or ∅ ∉ *𝑁𝑁𝑁𝑟*(*𝑋*) **then**

15: flag=1

16: **else**

17: *𝑟* ∶= *𝑟* + 1

18: **end if**

19: **end while**

**Remrk.** NNN searching algorithm proposed by Zhu et al. [[21](#_bookmark62)] does not take outliers or noise into consideration. In contrast, by introducing

ln *𝑛* + ln *𝜆* as the threshold, the robust natural nearest neighborhood

searching algorithm we propose can recognize outliers and eliminate

their effect.

based on the spread of the epidemic in recent years. An infected man will infect the persons he contacts. Therefore, we first choose the data

point with the largest local density *𝜌* in *𝐶𝑒𝑛𝑠*(*𝑋*) as patient zero. In

nearest neighborhood (*𝑁𝑁𝑁* ). These neighbors will infect their natural addition, he will infect the persons he contacts with, that is his natural

nearest neighbors and the propagation will continue until there is no neighbor that can be infected. At this time, we recognize all the infected data points as a cluster. We call the processes of forming a cluster a round of propagations. Second, we select the data point that has the

largest local density among those data points not infected in *𝐶𝑒𝑛𝑠*(*𝑋*),

and repeat the process above until all the data points in *𝐶𝑒𝑛𝑠* have been

infected.

In the real world, some people will not become infected even when

other reasons. Therefore, in the algorithm, a data point *𝑥* ∈ *𝑋* is they contact with infected persons because they have antibodies or

infected with a certain probability *𝑝𝑥* = *𝐶* ⋅ (*𝑝*′ + *𝑝*′′), where *𝑝*′ (and *𝑝*′′)

*𝑥 𝑥 𝑥 𝑥*

denotes the rank of *𝑥* in the current round of propagation. Formally, in the current round of propagation, if point *𝑥* (to be checked) and the infected points can be ordered *𝑦*1*,* … *, 𝑦𝑖*−1*, 𝑥, 𝑦𝑖*+1*,* … *, 𝑦𝑘* with *𝜌*(*𝑦*1) ≤

… ≤ *𝜌*(*𝑦𝑖*−1) ≤ *𝜌*(*𝑥*) ≤ *𝜌*(*𝑦𝑖*+1) ≤ … ≤ *𝜌*(*𝑦𝑘*), then we define

*𝑝*′ = *𝑖* ;

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | No. of records | No. of attributes | No. of clusters |
| 2d-4c-no9 | 876 | 2 | 4 |
| 3-spiral | 312 | 2 | 3 |
| Aggregation | 788 | 2 | 7 |
| Cassini | 1000 | 2 | 3 |
| Complex9 | 3031 | 2 | 9 |
| Compound | 399 | 2 | 6 |
| Dartboard1 | 1000 | 2 | 4 |
| Jain | 373 | 2 | 2 |
| R15 | 600 | 2 | 15 |
| Shapes | 1000 | 2 | 4 |

**Table 1**

Synthetic datasets.

*𝑥 𝑘*

if point *𝑥* and the unchecked points in ∪*𝑘 𝑁𝑁𝑁* (*𝑦𝑗* ) can be ordered

*𝑗*=1*,𝑗*≠*𝑖*

*𝑧*1*,* … *, 𝑧𝑗*−1*, 𝑥, 𝑧𝑗*+1*,* … *, 𝑧*𝓁 with *𝜌*(*𝑧*1) ≤ … ≤ *𝜌*(*𝑧𝑗*−1) ≤ *𝜌*(*𝑥*) ≤ *𝜌*(*𝑧𝑗*+1) ≤

… ≤ *𝜌*(*𝑧*𝓁), then we define

*𝑝*′′ = *𝑗 .*

*𝑥* 𝓁

If point *𝑥* is not infected, we consider it having antibodies or

propagation process, we increase the probability *𝑝𝑥* of being infected to dead and that it will no longer be infected. At the beginning of the

propagation process, we decrease the probability *𝑝𝑥* of being infected ensure that the dissemination process will continue. At the end of the

to guarantee that the outliers or noise will not be classified into any cluster.

**Algorithm 2** Probabilistic propagation clustering algorithm (PP)

**Input:** A set of points *𝑋 ⊂* R*𝑑* , *𝜌* = *𝜌*(*𝑋*), *𝑁𝑁𝑁* (*𝑋*)

**Output:** Clustering results *𝐶𝐿𝑈* 1: Initially set *𝐶𝑒𝑛𝑠* = *𝐶𝑒𝑛𝑠*(*𝑋*) 2: *𝑛* = 1

3: *𝑥* = *𝑓 𝑖𝑛𝑑*(max *𝜌*(*𝐶𝑒𝑛𝑠*)); *𝐶𝑒𝑛𝑠* = *𝐶𝑒𝑛𝑠* ⧵ {*𝑥*};

*𝐶𝐿𝑈* (*𝑛*) = ∅*,* ; *𝐶𝐿𝑈𝑐𝑎𝑛* = *𝑁𝑁𝑁* (*𝑥*);

4: **while** *𝐶𝐿𝑈𝑐𝑎𝑛* ≠ ∅ **do**

5: **for** ∀*𝑦* ∈ *𝐶𝐿𝑈𝑐𝑎𝑛* **do**

6: calculate the probability of being infected for y: *𝑝𝑦*

7: **if** *𝑦* is infected (with probability *𝑝𝑦*) **then**

8: Reset *𝐶𝐿𝑈* (*𝑛*) = *𝐶𝐿𝑈* (*𝑛*) ∪ {*𝑦*}

9: Reset *𝐶𝐿𝑈𝑐𝑎𝑛* = *𝐶𝐿𝑈𝑐𝑎𝑛* ∪ *𝑁𝑁𝑁* (*𝑦*)

10: **if** *𝑦* ∈ *𝐶𝑒𝑛𝑠* **then**

11: Reset *𝐶𝑒𝑛𝑠* = *𝐶𝑒𝑛𝑠* ⧵ {*𝑦*}

12: **end if**

13: **else**

14: y does not belong to any cluster

15: **end if**

16: Reset *𝐶𝐿𝑈𝑐𝑎𝑛* = *𝐶𝐿𝑈𝑐𝑎𝑛* ⧵ {*𝑦*}

17: **end for**

18: **end while**

**Table 2**

Real-world datasets.

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | No. of records | No. of attributes | No. of clusters |
| Ecoli | 336 | 7 | 8 |
| Glass | 214 | 9 | 7 |
| Heart-statlog | 270 | 13 | 2 |
| Iono | 351 | 34 | 2 |
| Iris | 150 | 4 | 3 |
| Thy | 215 | 5 | 3 |
| Wdbc | 569 | 30 | 2 |
| Wine | 178 | 13 | 3 |

the K-means and DBSCAN algorithms are implemented in the sklearn library of Python and the DPC algorithm, without the ‘‘Halo’’ part, is based on the source code provided by the author since our datasets do not contain noise. All the results shown are the optimal results after argument tuning.

The synthetic datasets and real-world datasets used in the experi- ments are presented in [Tables](#_bookmark23) [1](#_bookmark23) and [2](#_bookmark25), respectively.

* 1. *Preliminaries*

To evaluate the performance of the clustering algorithm, three evaluation indices are introduced, namely, Adjusted Rand Index (ARI), Adjusted Mutual Information (AMI) and Fowlkes–Mallows index (FMI). The upper bound of the three indicators is 1, in other words, larger values of the indicators indicate better clustering results.

Before the tests, all of the real-world datasets should be adjusted by min–max normalization to eliminate the differences in the ranges of different dimensions, as shown in Eq. ([10](#_bookmark26)).

19: **if** *𝐶𝑒𝑛𝑠* ≠ ∅ **then**

20: *𝑛* = *𝑛* + 1

′

*𝑖𝑗*

*𝑥*

= *𝑥𝑖𝑗* − *𝑚𝑖𝑛*(*𝑥𝑗* )

*𝑚𝑎𝑥*(*𝑥𝑗* ) − *𝑚𝑖𝑛*(*𝑥𝑗* )

(10)

21: goto Step 3

22: **end if**

where *𝑥*

*𝑖𝑗*

is the original data in the *𝑖*th position of *𝑥𝑗* .

23: Assign each remaining data point to the cluster that has the largest sum of local densities among its natural nearest neighborhood

To more objectively reflect the actual results of various algorithms, we perform argument tuning on each of the algorithms, thereby en-

suring that their best performances are compared. For K-means, we

simply give the correct number of clusters. For DBSCAN, we use a grid

When the propagation process is over, there will be some data points that are not infected and are not outliers. We assign each remaining data point to the cluster that has the largest sum of local densities among the natural nearest neighborhoods containing it.

By introducing the idea of probabilistic propagation, we can make the cluster algorithm closer to reality. Moreover, we can distinguish

clustering algorithm is shown in Alg. [2](#_bookmark24). Denote *𝜌*(*𝐴*) = {*𝜌*(*𝑥*) | *𝑥* ∈ *𝐴*}. two clusters that are close to each other. The probabilistic propagation

# Experiment

We call the algorithms [1](#_bookmark20) and [2](#_bookmark24) together the DPC-PPNNN algo- rithm. To test the performance of the DPC-PPNNN algorithm, we use classical synthetic datasets and real-world datasets. Moreover, we take traditional DPC, K-means and DBSCAN as the control group, where

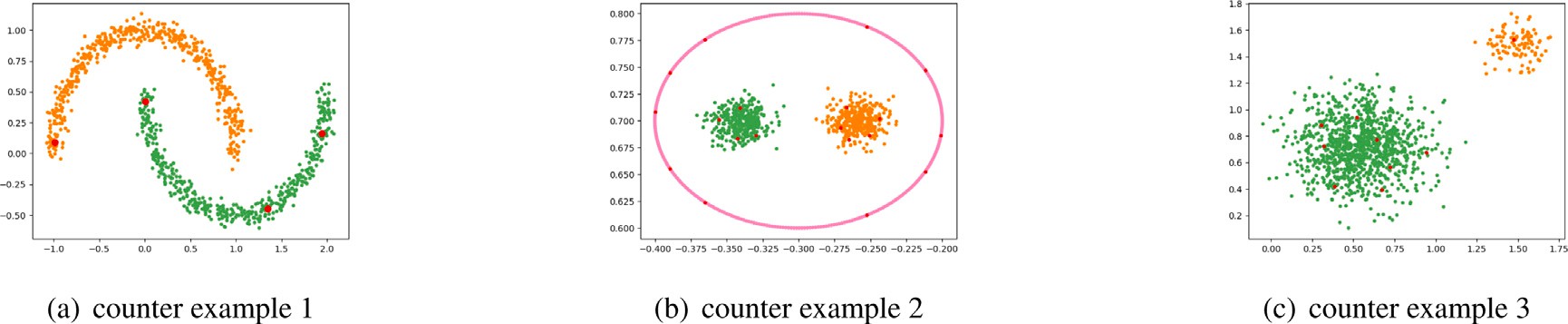
choose the optimal *𝑑𝑐* from 1% to 2% and apply the Gaussian kernel in search to find the best parameter configuration. For traditional DPC, we

the density estimation process.

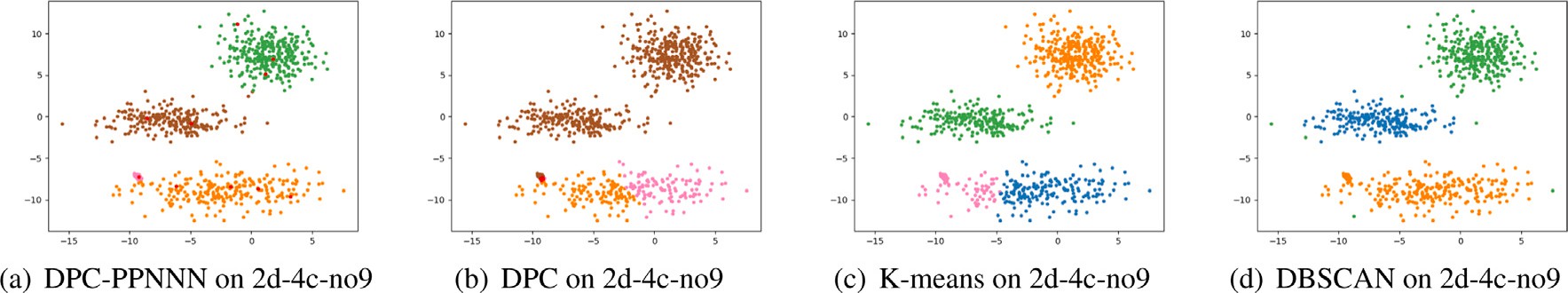
* 1. *Synthetic datasets*

In this part, we select a number of synthetic datasets that are widely used to test the performance of clustering algorithms. These datasets are different in terms of the distribution and numbers of points and clusters. They can simulate different situations to compare the performance of various clustering algorithms in different scenarios.

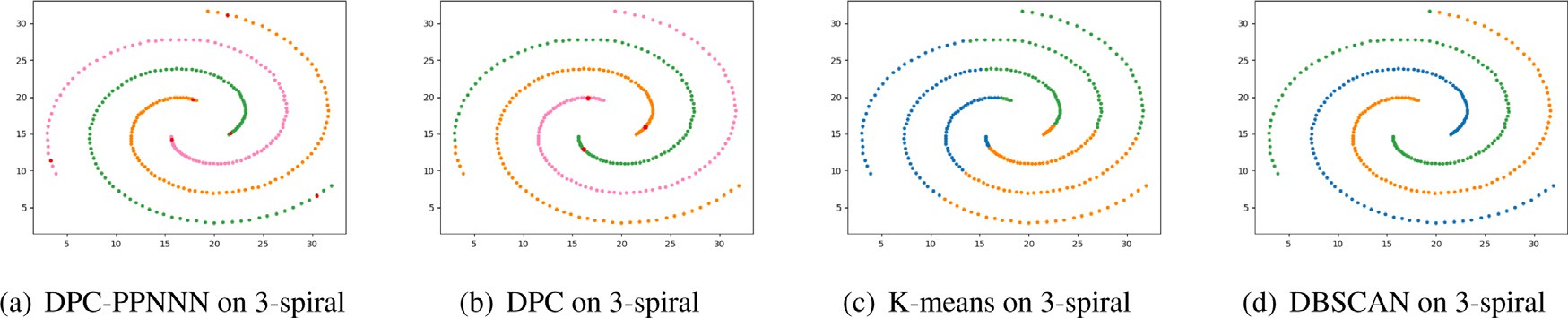
[Table](#_bookmark39) [3](#_bookmark39) shows the clustering results in terms of the ARI, AMI and FMI scores on all synthetic datasets listed in [Table](#_bookmark23) [1](#_bookmark23). The evaluation criterion with the highest score is marked in bold, and we can see that in most cases, the DPC-PPNNN obtains the highest score, except for



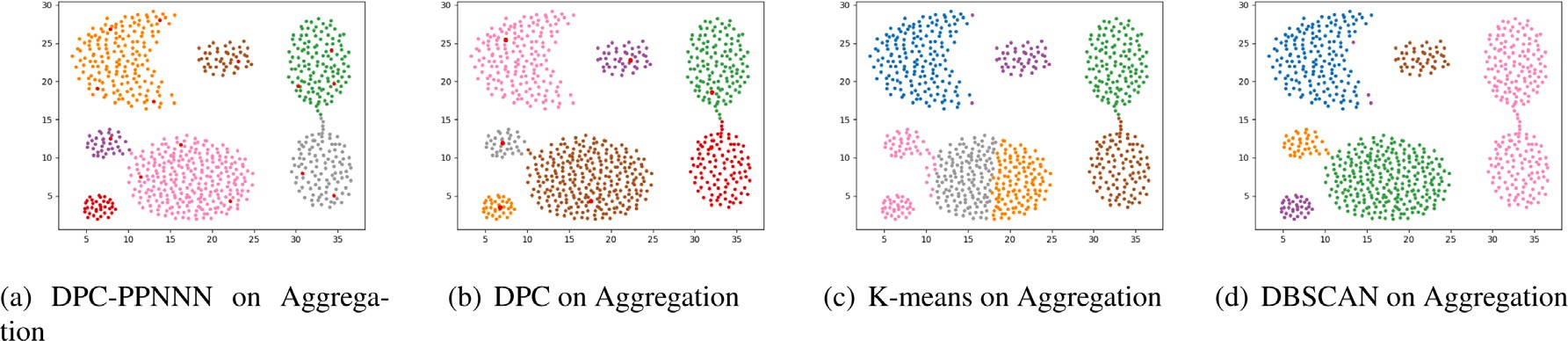
**/ig. 4.** The re-clustering results of the three counter examples in Section [3.2](#_bookmark10). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



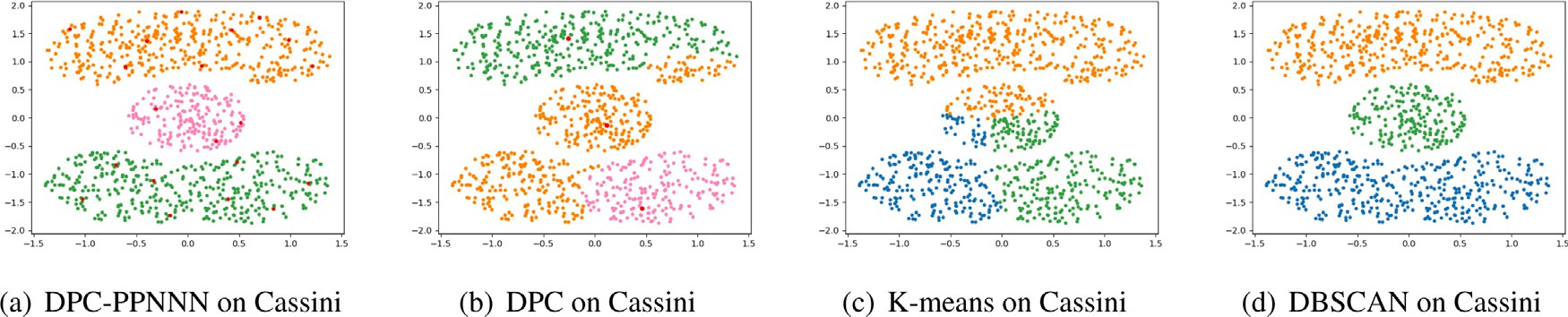
**/ig. 5.** The clustering results on 2d-4c-no9 by 4 algorithms.



**/ig. 6.** The clustering results on 3-spiral by 4 algorithms.



**/ig. 7.** The clustering results on Aggregation by 4 algorithms.



**/ig. 8.** The clustering results on Cassini by 4 algorithms.

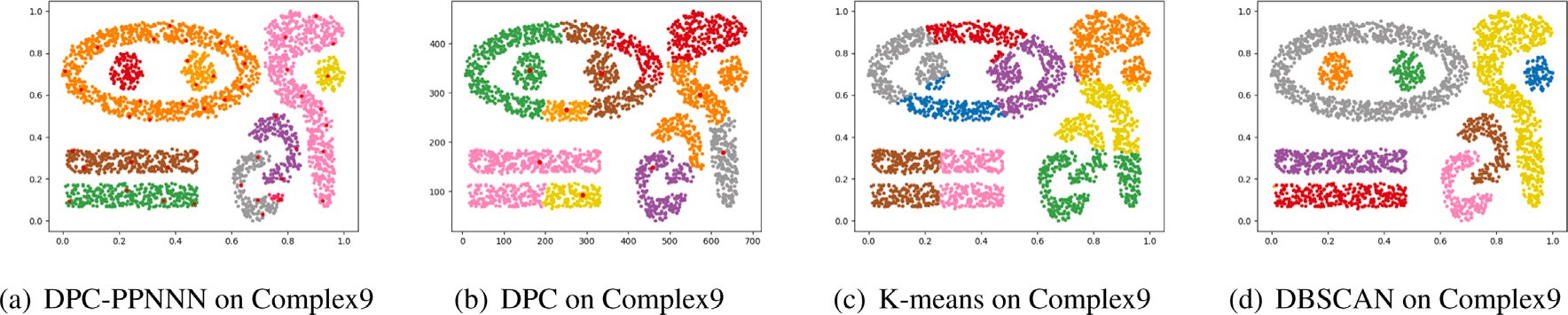
the Complex9 (where the score of DPC-PPNNN is a bit less than the DBSCAN).

In [Fig.](#_bookmark28) [4](#_bookmark28), we recluster the three counter examples in Section [3.2](#_bookmark10) by DPC-PPNNN. Clearly, the results are much better than the original DPC which implies that our algorithm can overcome the deficiencies of DPC. Next, we will show some typical clustering results on the synthetic datasets by DPC-PPNNN and the algorithms of the control group. The points with different colors in the figures are assigned to different

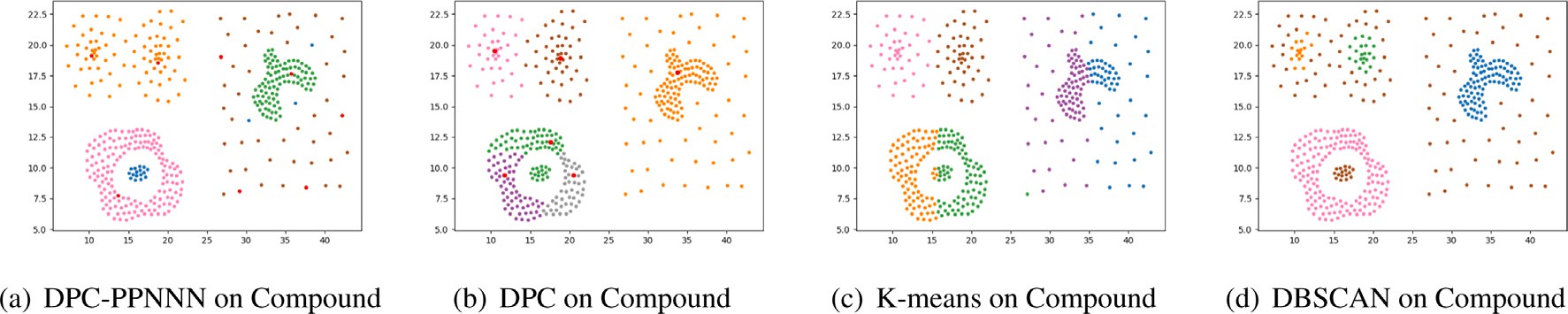
clusters. The cluster centers obtained from DPC and DPC-PPNNN are marked in red.

In [Fig.](#_bookmark29) [5](#_bookmark29), we can see that the 2d-4c-no9 dataset has four clusters, of which one cluster has a very high density. DPC-PPNNN succeeds in detecting all of them, while the other algorithms of the control group fail to do so.

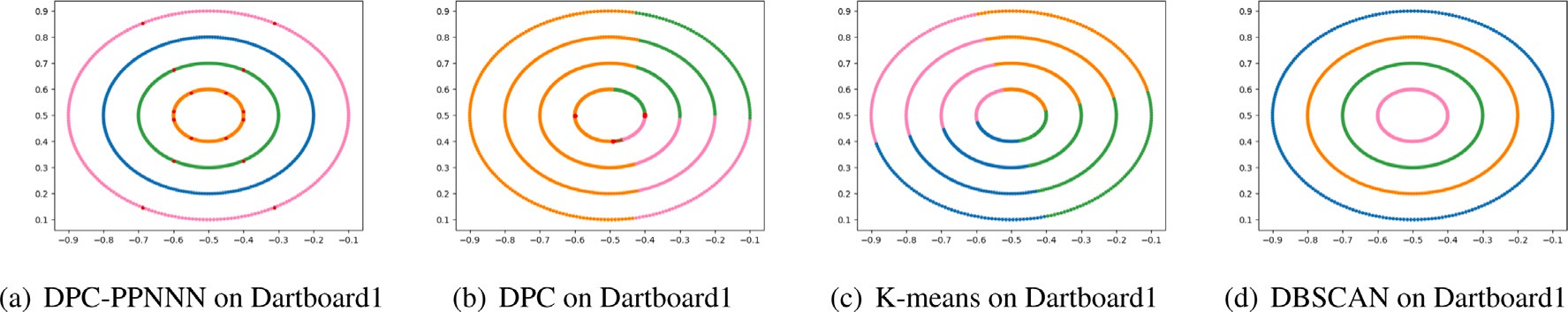
[Fig.](#_bookmark30) [6](#_bookmark30) shows the results of each algorithm on the 3-spiral dataset. DPC-PPNNN can identify the four clusters correctly. Furthermore, DPC



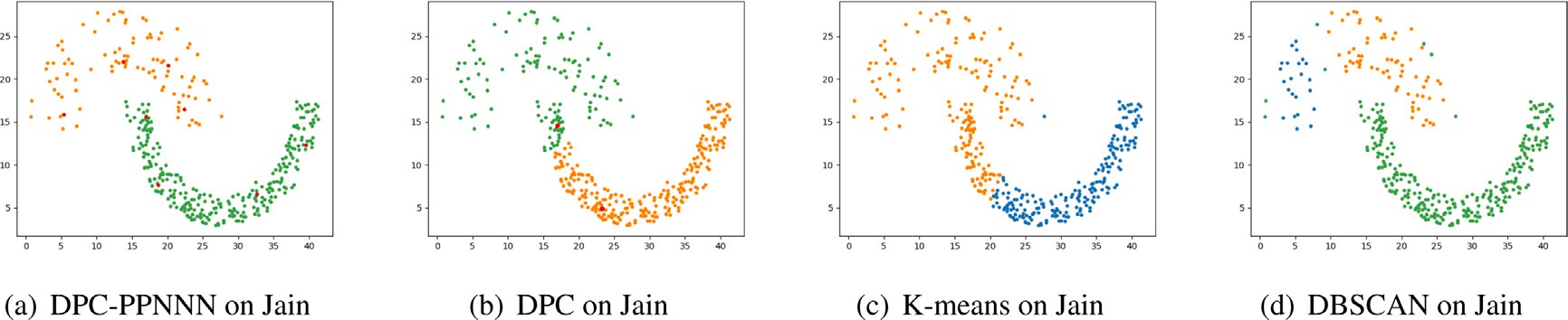
**/ig. 9.** The clustering results on Complex9 by 4 algorithms.



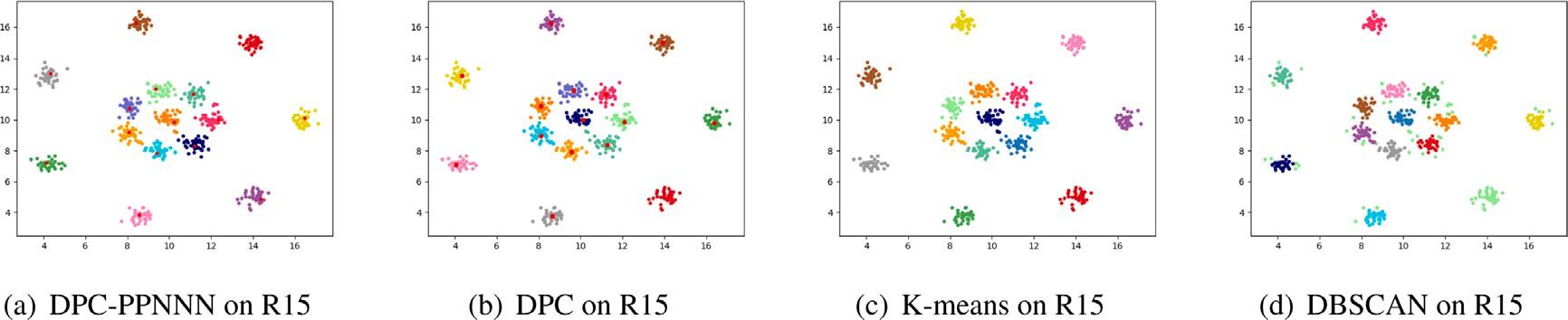
**/ig. 10.** The clustering results on Compound by 4 algorithms.



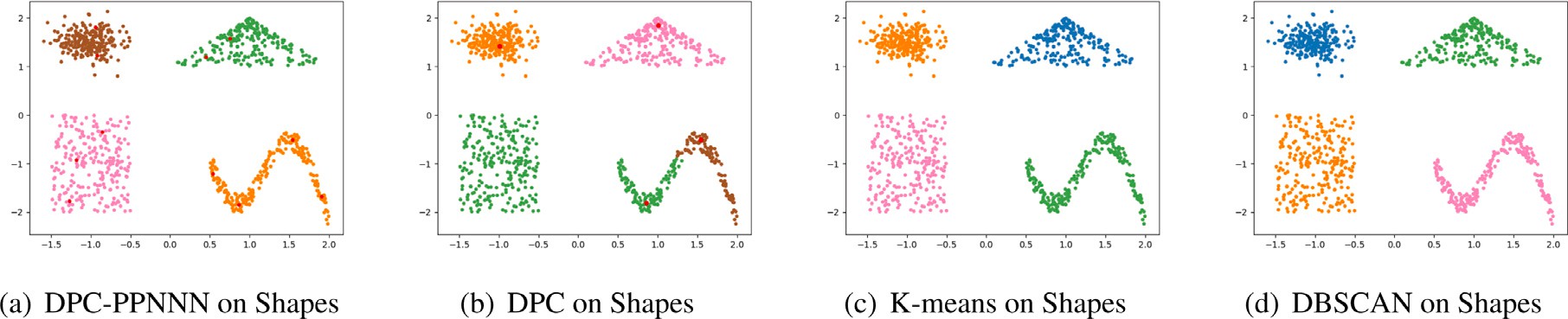
**/ig. 11.** The clustering results on Dartboard1 by 4 algorithms.



**/ig. 12.** The clustering results on Jain by 4 algorithms.



**/ig. 13.** The clustering results on R15 by 4 algorithms.



**/ig. 14.** The clustering results on Shapes by 4 algorithms.

and DBSCAN detect the four clusters mostly correctly with some wrong data points at the tail, perhaps because the arguments are not set

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| correctly, while KNN cannot recognize them at all. | Algorithm | ARI | AMI | FMI | ARI | AMI | FMI |
| The clustering results of the DPC-PPNNN and the other three algo- |  | 2d-4c-no9 |  |  | Compound |  |  |
| rithms of the control group on the Aggregation dataset are shown in | DPC-PPNNN(ours) | **0.9931** | **0.9898** | **0.9951** | **0.9375** | **0.9345** | **0.9543** |
| [Fig.](#_bookmark31) [7](#_bookmark31). The dataset can be detected correctly by DPC-PPNNN and DPC. | DPC [[12](#_bookmark53)] | 0.3336 | 0.5261 | 0.6108 | 0.5099 | 0.7542 | 0.6235 |
| K-means fails to do so because the dataset is somewhat complex for  K-means. DBSCAN cannot distinguish the two clusters on the right. | K-means [[3](#_bookmark44)]  DBSCAN [[7](#_bookmark48)] | 0.8952  0.8412 | 0.9007  0.8810 | 0.9241  0.8940 | 0.5364  0.8774 | 0.7128  0.8673 | 0.6410  0.9103 |
| From the clustering results in [Fig.](#_bookmark32) [8](#_bookmark32), we can see that DPC-PPNNN |  | 3-spiral |  |  | Dartboard1 |  |  |

**Table 3**

Performances of different clustering algorithms on different synthetic datasets.

and DBSCAN can detect the clusters in the Cassini dataset. Although DPC can find the correct cluster centers, it fails to allocate the other remaining data points correctly. The three clusters are not uniform in shape which leads to the wrong cluster results by K-means.

The [Fig.](#_bookmark33) [9](#_bookmark33) shows the results of the four algorithms on the Complex9 dataset. DBSCAN can recognize all the clusters successfully. While for DPC-PPNNN, there are some misclassified points at the tail of a cluster because these points are not connected tightly enough. DPC finds the correct cluster centers but it fails to allocate the other remaining data points correctly. K-means has poor performance.

[Fig.](#_bookmark34) [10](#_bookmark34) displays the results on the Compound dataset. DPC-PPNNN can detect most of the clusters correctly and can also recognize noise. DBSCAN is good at recognizing noise but it also misclassifies many data points as noise. DPC and K-means fail to recognize the clusters with noise.

As shown in [Fig.](#_bookmark35) [11](#_bookmark35), the Dartboard1 dataset has four concentric rings. Both DPC-PPNNN and DBSCAN have perfect performance. DPC cannot find the correct cluster centers because the densities of the data points have little difference. Moreover, K-means has poor performance. For the Jain dataset shown in [Fig.](#_bookmark36) [12](#_bookmark36), only DPC-PPNNN correctly identifies all clusters. DPC has not found the correct cluster centers because the difference between the densities of the two clusters is too large. The reason is the same for the poor performance of DBSCAN.

K-means is not suitable for the datasets with streamline shapes.

[Fig.](#_bookmark37) [13](#_bookmark37) displays the results on the R15 dataset. The distribution of points makes it the most straightforward dataset for all the algorithms. Although there are some small defects among them, all the algorithms can recognize both the clusters and centers.

The clustering shown in [Fig.](#_bookmark38) [14](#_bookmark38) demonstrates the ability to address the datasets with clusters of different shapes. DPC-PPNNN, K-means and DBSCAN all perform well. Only DPC cannot detect the clusters because there is a cluster with manifold structure.

* 1. *Read-world datasets*

In this section, 8 UCI datasets, as shown in [Table](#_bookmark25) [2](#_bookmark25), are used to demonstrate the performance of the DPC-PPNNN clustering algorithm. These datasets are different in terms of sample number, feature number and cluster number. As shown in [Table](#_bookmark41) [4](#_bookmark41), DPC-PPNNN performs almost the best among the test cases.

# Conclusions

In this paper, we proposed an improved probability propagation al- gorithm for density peak clustering based on natural nearest neighbors (DPC-NNN). The new algorithm does not require any parameters and can recognize cluster centers automatically. The final clustering process of the DPC-PPNNN motivated by the epidemic spread performs well especially for distinguishing two clusters that are close to each other.

However, since the algorithm is based on density and probability,

data points have similar *𝛿* or similar densities *𝜌*. For future work, we its performance is not good and stable enough, especially when all the

will work on solving these problems.

# Declaration of competing interest

The authors declare that they have no known competing finan- cial interests or personal relationships that could have appeared to influence the work reported in this paper.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| DPC-PPNNN(ours) | **1.0000** | **1.0000** | **1.0000** | **1.0000** | **1.0000** | **1.0000** |
| DPC [[12](#_bookmark53)] | 0.9521 | 0.9406 | 0.9680 | 0.0006 | 0.0130 | 0.3213 |
| K-means [[3](#_bookmark44)] | −0.0060 | −0.0055 | 0.3274 | −0.0030 | −0.0033 | 0.2470 |
| DBSCAN [[7](#_bookmark48)] | 0.9953 | 0.9918 | 0.9969 | **1.0000** | **1.0000** | **1.0000** |
| Aggregation Jain | | | | | | |
| DPC-PPNNN(ours) | 0.9927 | 0.9881 | 0.9943 | **1.0000** | **1.0000** | **1.0000** |
| DPC [[12](#_bookmark53)] | **0.9978** | **0.9956** | **0.9983** | 0.6438 | 0.5951 | 0.8502 |
| K-means [[3](#_bookmark44)] | 0.7624 | 0.8776 | 0.8159 | 0.3241 | 0.3677 | 0.7005 |
| DBSCAN [[7](#_bookmark48)] | 0.8719 | 0.9223 | 0.9051 | 0.9350 | 0.8476 | 0.9742 |
|  | Cassini |  |  | R15 |  |  |
| DPC-PPNNN(ours) | **1.0000** | **1.0000** | **1.0000** | **0.9928** | **0.9938** | **0.9932** |
| DPC [[12](#_bookmark53)] | 0.4886 | 0.5879 | 0.6736 | 0.9857 | 0.9885 | 0.9866 |
| K-means [[3](#_bookmark44)] | 0.5308 | 0.5400 | 0.7012 | **0.9928** | **0.9938** | **0.9932** |
| DBSCAN [[7](#_bookmark48)] | **1.0000** | **1.0000** | **1.0000** | 0.8307 | 0.8905 | 0.8416 |
|  | Complex9 |  |  | Shapes |  |  |
| DPC-PPNNN(ours) | 0.9375 | 0.9345 | 0.9543 | **1.0000** | **1.0000** | **1.0000** |
| DPC [[12](#_bookmark53)] | 0.3929 | 0.6666 | 0.4940 | 0.7439 | 0.8410 | 0.8145 |
| K-means [[3](#_bookmark44)] | 0.3461 | 0.6161 | 0.4517 | **1.0000** | **1.0000** | **1.0000** |
| DBSCAN [[7](#_bookmark48)] | **0.9998** | **0.9994** | **0.9998** | **1.0000** | **1.0000** | **1.0000** |

**Table 4**

Performances of different clustering algorithms on different real-world datasets. Algorithm ARI AMI FMI ARI AMI FMI

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Ecoli |  |  | Iris |  | |
| DPC-PPNNN(ours) | **0.5761** | 0.5614 | **0.7160** | **0.8680** | **0.8446** | **0.9114** |
| DPC [[12](#_bookmark53)] | 0.2660 | 0.3179 | 0.4361 | 0.5414 | 0.6780 | 0.7539 |
| K-means [[3](#_bookmark44)] | 0.4115 | **0.5921** | 0.5482 | 0.7163 | 0.7387 | 0.8112 |
| DBSCAN [[7](#_bookmark48)] | 0.1004 | 0.1275 | 0.5282 | 0.5681 | 0.7316 | 0.7715 |
|  | Glass |  |  | Thy |  |  |
| DPC-PPNNN(ours) | **0.2595** | 0.3450 | **0.5534** | **0.7522** | **0.6108** | **0.8827** |
| DPC [[12](#_bookmark53)] | 0.0651 | 0.0984 | 0.3516 | 0.2105 | 0.2301 | 0.7530 |
| K-means [[3](#_bookmark44)] | 0.1662 | 0.2942 | 0.3945 | 0.6283 | 0.5909 | 0.8546 |
| DBSCAN [[7](#_bookmark48)] | 0.2382 | **0.3577** | 0.5500 | 0.6715 | 0.4911 | 0.8606 |
| Heart-statlog Wdbc | | | | | | |
| DPC-PPNNN(ours) | 0.2380 | 0.2186 | 0.5432 | 0.7193 | 0.6054 | 0.8671 |
| DPC [[12](#_bookmark53)] | 0.0003 | 0.0002 | 0.7050 | 0.4705 | 0.4614 | 0.7860 |
| K-means [[3](#_bookmark44)] | **0.3488** | **0.2692** | **0.6755** | **0.7302** | **0.6226** | **0.8770** |
| DBSCAN [[7](#_bookmark48)] | 0.0861 | 0.1242 | 0.3417 | 0.2201 | 0.2618 | 0.6091 |
|  | Iono |  |  | Wine |  |  |
| DPC-PPNNN(ours) | 0.5287 | 0.4114 | 0.7935 | 0.7421 | 0.7268 | 0.8277 |
| DPC [[12](#_bookmark53)] | 0.0019 | 0.0013 | 0.7302 | 0.6990 | 0.7233 | 0.8006 |
| K-means [[3](#_bookmark44)] | 0.1776 | 0.1330 | 0.6053 | **0.8537** | **0.8400** | **0.9026** |
| DBSCAN [[7](#_bookmark48)] | **0.7214** | **0.6343** | **0.8779** | 0.4229 | 0.5209 | 0.6482 |

# Acknowledgments

The work was supported by National Natural Science Foundation of China (No. 12071453), the National Key R and D Program of China (2020YFA0713100), Anhui Initiative in Quantum Information Technologies, China (AHY150200), and the Innovation Program for Quantum Science and Technology, China (2021ZD0302904).

# References

1. [Tan PN, Steinback M, Kumar V. Introduction to data mining. Data mining. 2011.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb1)
2. [Liu R, Wang H, Yu X. Shared-nearest-neighbor-based clustering by fast search](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb2) [and find of density peaks. Inform Sci 2018;450:200–26.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb2)
3. Macqueen JB. Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, 1967. 1967.
4. [Kaufmann L, Rousseeuw PJ. Clustering by means of medoids. North-Holland;](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb4) [1987.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb4)
5. [Guha S, Rastogi R, Shim K. Cure: An efficient clustering algorithm for large](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb5) [databases. Inf Syst 1998;26:35–58.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb5)
6. [Zhang T, Ramakrishnan R, Livny M. Birch: An efficient data clustering method](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb6) [for very large databases. ACM SIGMOD Rec 1996;25:103–14.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb6)
7. [Ester M, Kriegel HP, Sander J, Xu X. A density-based algorithm for discovering](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb7) [clusters in large spatial databases with noise. AAAI Press; 1996.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb7)
8. [Ankerst M, Breunig MM, Kriegel HP, Sander J. Optics: Ordering points to](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb8) [identify the clustering structure. In: SIGMOD 1999, Proceedings ACM SIGMOD](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb8) [international conference on management of data, June 1-3, 1999. Philadelphia,](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb8) [Pennsylvania, USA; 1999.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb8)
9. [Sheikholeslami G, Chatterjee S, Zhang A. Wavecluster: A wavelet based clustering](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb9) [approach for spatial data in very large databases. VLDB J 2000;8:289–304.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb9)
10. [Wang W, Yang J, Muntz R. Sting: A statistical information grid approach to](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb10) [spatial data mining. VLDB J 1997;25:186–95.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb10)
11. [Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb11) [via the em algorithm. Proc R Stat Soc 1977;39:1–22.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb11)
12. [Rodriguez A, Laio A. Clustering by fast search and find of density peaks. Science](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb12) [2014;344:1492.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb12)
13. [Du M, Ding S, Jia H. Study on density peaks clustering based on k-nearest](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb13) [neighbors and principal component analysis. Knowl-Based Syst 2016;99:135–45.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb13)
14. [Mehmood R, Zhang G, Bie R, Dawood H, Ahmad H. Clustering by fast search](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb14) [and find of density peaks via heat diffusion. Neurocomputing 2016;208:210–7.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb14)
15. [Liang Z, Chen P. Delta-density based clustering with a divide-and-conquer](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb15) [strategy. Pattern Recognit Lett 2016;73:52–9.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb15)
16. [Xu J, Wang G, Deng W. DenPEHC: Density peak based efficient hierarchical](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb16) [clustering. Inform Sci 2016;373:200–18.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb16)
17. [Tao LI, Hongwei GE, Shuzhi SU. Density peaks clustering by automatic](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb17) [determination of cluster centers. Front Comput Sci Technol 2016;10:1614–22.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb17)
18. [Zhou Z, Si G, Zhang Y, Zheng K. Robust clustering by identifying the veins of](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb18) [clusters based on kernel density estimation. Knowl-Based Syst 2018;159:309–20.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb18)
19. [Xie JY, Grant WX, Philip W, Liu XH, Gao HC. Robust clustering by detecting](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb19) [density peaks and assigning points based on fuzzy weighted k-nearest neighbors.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb19) [Inform Sci 2016;354:19–40.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb19)
20. [Geng YA, Li Q, Zheng R, Zhuang F, He R. Recome: a new density-based clustering](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb20) [algorithm using relative knn kernel density. Inform Sci 2018;436–437:13–30.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb20)
21. [Zhu Q, Huang J, Feng J, Zhou X. A clustering algorithm based on natural nearest](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb21) [neighbor. J Comput Inf Syst 2014;10:5473–80.](http://refhub.elsevier.com/S2590-0056(22)00070-4/sb21)