Improved density peak clustering for large datasets

Vincent Courjault-Rade, Ludovic d'Estampes, Stéphane Puechmorel

Improved density peak clustering for large datasets

Vincent Courjault-Radé

Université fédérale de Toulouse, ENAC

F-31055 Toulouse, France

and

Ludovic d'Estampes

Université fédérale de Toulouse, ENAC

F-31055 Toulouse, France

and

Stéphane Puechmorel

Université fédérale de Toulouse, ENAC

F-31055 Toulouse, France

August 12, 2016

Abstract

Clustering is the usual way of classifying data when there is no a priori knowledge, especially about the number of classes. Within the frame of big data analysis, the computational effort needed to perform the clustering task may become prohibitive and motivated the construction of several algorithms or the adaptation of existing

ones, as the well known K-means algorithm [12]. Recently, Rodriguez and Laio [17] proposed an algorithm that clusters efficiently by fast searching local density peaks that are sufficiently distant one from the others. However it is able to work on small datasets only and is highly sensitive to the value of tunable parameters. In this paper we propose Improved Density Peak Clustering (IDPC), a new algorithm designed for large datasets based on [17] which corrects the shortcomings mentioned above. Thanks to our Cover Map (CM) procedure iterated with a decreasing locally-adaptive window (ICMDW), we are able to build both a localisation map and a multidimensional density map. The nature of the density map, which fits perfectly with the approach of [17], allows us to compute the different steps with much less operations. It carries unsensitive parameters, supports last improvements on cluster centers selection and potentially allows new improvements.

Keywords: Clustering algorithm; density-based clustering; large datasets;

1 Introduction

Clustering is a commonly used tool that aims to identify similar samples in a dataset. It can be viewed both from the machine learning and the statistics point of view, with classical algorithms pertaining to each domain. When considered from the algorithmic standpoint, the complexity of the clustering problem is known to be NP hard, even for the usual K-means, when the number of clusters is not fixed.

Presented in [17], the innovative fast density peak detection (FDPC) algorithm is able to cope with non-convex clusters and gives a convenient decision rule to find the correct number of clusters. As in the mean shift algorithm [3], clusters centers are defined as local density maxima. More specifically, a cluster center is defined as a point surrounded by neighbours with lower local density and away from any other local maxima. In order to detect them, a distance d_c is first selected and is used to define neighborood. Densities ρ_i are calculated at each data point x_i by counting the number of sample points closer than

d_c to x_i (1). Next, for each data point, the nearest point with a higher density is found and the distance δ_i separating them is recorded (2). Cluster centers are then defined as the points x_i that have both high density ρ_i and high distance δ_i. Other points receive the same label than their nearest neighbor of higher density (3).

This innovative method has promising results but shows some limitations. In step (1), the distances between all sample points are calculated, leading to an $O(N^2)$ complexity and memory footprint, not to mention other steps. This issue limits it to small datasets. Furthermore, choosing a good threshold distance d_c is non-trivial and critical since it has a major impact on the final clustering.

Our new algorithm aims to overcome the drawbacks mentioned above. The general idea of IDPC is to start by the construction of a density map trough the procedure Iterated Cover Map with a Decreasing Window (ICMDW), reducing the dimension N of the problem, and then to use the localisation information it provides to construct a very fast version of steps (2) and (3).

In the second Section we review the work related to [17]. In the third Section we detail the implementation of the ICMDW algorithm. In the fourth Section we explain how the results provided by ICMDW are used to greatly reduce the computations in cluster centroid selection and labelisation. And finally, we present experimental results in the fifth Section.

2 Related work

2.1 Fast Density Peak Clustering

Let $(x_i)_{i=1...N}$ be the sample points. In the first step of FDPC the Euclidean distances d_{ij} between each data points are computed. Then a distance d_c is chosen to finally compute, for each x_i, the number of data points that are closer than d_c to the point x_i. Local densities

are then obtained according to the formula:

$$\rho_i := \rho(x_i) = \sum_{j=1}^{N} (\mathbf{1}_{\mathbb{R}^-}(d_{ij} - d_c)).$$

Please note that the expression is of the form $\sum_{j=1}^{N} K(x_i - x_j)$, where K is the so-called kernel function, that reduces here to the characteristic function of a ball $1_{B(0,d_c)}$.
In the second step, the minimum distance between the point x_i and any other point with higher density is calculated as $\delta_i = \min_{j:\rho_j > \rho_i}(d_{ij})$. Finally, cluster centers are found as outliers in the decision graph formed by (δ, ρ), where outliers are the points sharing anomalously large density-delta values. Conventionally, for the point x_k with highest density, $\delta_k = \max_j(d_{ij})$.

Finally, other points are assigned to the same cluster than their Nearest Point with Higher Density (NHPD). Its algorithmic is detailed in Algorithm 1.

Many researchers worked on this algorithm in order to improve and/or use it on a specific

Algorithm 1 FDPC

procedure $FDPC(d_c)$
  for all $(i, j) \in \{1, ..., N\}^2$ **do**
    $d_{ij} \leftarrow \text{distance}(x_i, x_j)$
  end for
  for all $i \in \{1, ..., N\}$ **do**
    $\rho_i \leftarrow \sum_{j=1}^{N} (\mathbf{1}_{x<0}(d_{ij} - d_c))$
  end for
  for all $j \in \{1, ..., N\}$ **do**
    $\text{nhpd}_j \leftarrow \text{argmin}_{k:\rho(x_k)>\rho(x_j)}(d_{jk})$
    $\delta_j \leftarrow d_{j,\text{nhpd}_j}$
  end for
  Cluster centers $\leftarrow$ outliers in decision graph
  $\forall j$, $\text{Label}_j \leftarrow \text{Label}_{\text{nhpd}_j}$
end procedure

problem. It has been used successfully on clustering of electricity consumption behaviour [22], text clustering [10], unsupervised acoustic subword units discovery pompage [25], batch process modelling and on-line monitoring [16], medical data [7] and it has been

slightly modified to work on image segmentation [1, 2, 18], outliers detection [4] and hyper spectral band selection [6]. K-means and FDPC have been compared in [15] and FDPC has shown more accurate than K-means.

However, FDPC has proved to be slower than K-means and suffers from three other shortcomings:

- First of all, due to its inherent computational complexity, it cannot handle large data sets.
- Secondly, the choice of d_c has to be made empirically whereas it highly impacts every steps of the clustering algorithm.
- Finally, no decision rule is given to determine the outliers in the decision graph, leading to a manual or poor automatic selection of the cluster centers.

2.2 Improvements of FCDP

To overcome the limitations underlined above, a lot of work was done to derive improved versions of the original algorithm. Most of them deals only with one issue, due to the application:

Ability to handle large datasets

An extension of FDPC [9] has been proposed in order to manage large taxi fleet datasets. Data sample is first projected to a density image which is processed to obtain the densities ρ_i and the contours of clusters. This method works well on large datasets, but for 2-dimensional data only.

Choice of d_c

The baseline choice of d_c is to take the one that produce an average density count equal to 2% of N ($\bar{\rho} = 2\%N$)[4]. A non-trivial entropy-based choice of d_c has been brought in [20], but it is computationally difficult to perform, even more in large scale contexts. There

exist two methods that replace the density estimation (1) by a non-parametric one, avoiding the choice of d_c. In [21] the local density is estimated with a non-parametric multivariate kernel estimator. For each point x_i, the local density is calculated throughout the formula

$$\rho_i = \frac{1}{n\prod_{l}^{d} h_l}.\sum_{j=1}^{n} K(\frac{x_{i1}-x_{j1}}{h_1},...,\frac{x_{id}-x_{jd}}{h_d}),$$

where d is the dimension, K is a Gaussian kernel and $\{h_1,...,h_d\}$ are the bandwidths which are automatically and locally chosen. Moreover an automatic cluster centroid selection method is also developed through maximizing an average silhouette index. In [14], density estimation is done with an another non-parametric density estimator, based on the heat equation, which is more accurate but computationally expensive.

Cluster centers selection

In [11] data points are arranged according to their $\gamma_i = \delta_i.\rho_i$ values in descending order and the first m points are selected. Then centers are the points that possess a higher γ_i than the turning-point defined as

$$\text{argmax}_{i\in\{1,...,m\}}(\frac{k_i^1}{k_1^{i-1}}),\ \text{where } k_i^n = \frac{\gamma_{i+n}-\gamma_i}{n}.$$

Regarding [13], cluster centers are supposed to possess large δ_i ($\delta_i > 2\text{Var}(\delta)$ in [13], $\delta_i > 3\text{Var}(\delta)$ in [4]) and large ρ_i ($\rho_i > \text{mean}(\rho)$). Furthermore, a merge step is added: two clusters are merged if it exists a point x belonging to one cluster and a point y belonging to the other one that are closer than d_c. [23] also merges recursively the clusters based on their relative inter-connectivity and relative closeness.

An another d_c-based algorithm is presented in [8]. The concepts of DBSCAN [5] (density reachable, core objects..) are coupled with the divide-and-conquer strategy to produce a better cluster centers selection. Potential centroids obtained by the decision graph are

tested recursively and ignored if there is density-reachability between it and any other cluster centroid.

A k-nearest neighbors (KNN) based algorithm [19] is proposed as a rigorous way to detect cluster centers automatically. KNN is used to define $\rho_i = \frac{[K]}{\sum_{j=1} K d_{ij}}$. The product $\gamma = \rho.\delta$ is proven to follow a fat tailed distribution and a statistical test is proposed to find its outliers, which will define the cluster centers.

Other improvements

In [24], KNN is used to improve the assignment rule of the remaining points, right after the cluster centers detection step. First, outlier points (with a large $\delta_i^K = \max_{i \in \mathrm{KNN}_i}(d_{ij})$) are deleted. Secondly, remaining points are assigned through a restrictive strategy called "strategy 1". Roughly speaking, the label of a centroid x_c is propagated to the unlabelled points closer than $\sum_{j \in KNN_c} d_{cj}/K$ and those newly labelled points propagates recursively the label by the same way. Unassigned points by the strategy 1 are managed by the second strategy. The key aspect of strategy 2 is to learn the probability p_i^c that a point x_i belongs to cluster c, then assign the point i to its most similar cluster c with the highest value of p_i^c.

In [26], a derivative of FDPC that can manage uneven density datasets is proposed. The differences occurs for densities ρ_i and distances δ_i that are replaced by quantities with a similar behavior. The notion of mutual attraction of two points is introduced and is given by

$$f_{ij} = \min(f_{i \to j}, f_{j \to i}), \text{ where } f_{i \to j} = e^{-\frac{d(i,j)^2}{2\sigma^2}}.$$

The parameter σ, controlling the width of the kernel function, is locally calculated by

$$\sigma_i = \frac{\sum_{j \in J} d(i,j)}{Card(J)}, \text{ where } J = \{j | d(i,j) \leq d_c\}.$$

The densities ρ_i are then replaced by the influence scores

$$\rho_i = \frac{\sum_{j \in A} f_{i,j}}{card(A)}$$

and the δ_i are calculated with $\delta_i = \min_{j:\rho_j > \rho_i} \frac{d(i,j)}{min(\sigma_i, \sigma_j)}$.

3 Building density map and localisation map

In order to address the issue of large databases, instead of evaluating the densities at every single point $x_i \in N \times P$, we propose to evaluate the densities ρ_i at a reduced number of points covering the domain of the data. Data sample is randomized and pseudo normalized (normalized but with a small part of the dataset). The densities ρ_i are calculated with an adaptive windows d_c^g, smaller when the local density is high. To do so, the procedure Cover Map (CM) is iterated with a decreasing window size d_c^g. At each iteration g the results obtained from the last CM iteration are used to perform a cheaper CM than the one that would be directly performed with d_c^g.

3.1 Procedure Cover Map

For a given fixed window d_c, the Cover Map procedure covers the domain of the dataset with balls of radius equal to d_c.

The first observation is used to construct the first ball $B_1(x_1, d_c)$. Then for each observation x_i it checks if the data point belongs to at least one existing ball. If yes, then those balls increase their density count by one. Else, a new ball $B(x_i, d_c)$ is created (see algorithm 2).

Thanks to this procedure we obtain a subset of points, possessing much fewer points that the original one, with their associated local density ρ_i covering the whole domain. But it remains the difficult choice of d_c, and the reduced set of balls can still be too large for the second step.

Algorithm 2 Cover Map

procedure $CM(d_c)$
  $S \leftarrow B_1(x_1, d_c)$ ▷ Create the first ball
  $\rho_1(B_1(x_1, d_c)) \leftarrow 1$
  for all $x_i, i > 1$ **do**
    $\overrightarrow{d} \leftarrow \text{distance}(x_i, S)$
    $J \leftarrow \{j | \overrightarrow{d}[j] < d_c\}$
    if $J \neq \emptyset$ **then**
      $\forall j \in J, \rho_j(B_j) \leftarrow \rho_j(B_j) + 1$
    else
      $S \leftarrow S \cup B_k(x_i, d_c)$ ▷ Create new ball
      $\rho_k(B_k(x_i, d_c)) \leftarrow 1$
    end if
  end for
  return $\{S, \{\rho_k\}_k\}$
end procedure

3.2 Iterative CM procedure with a decreasing window (ICMDW)

A modified version of the CM procedure described above is iterated with a decreasing window $d_c^g = d_c^{g-1}.R,\ R \in]0,1[$. At each consecutive CM iteration it uses the results of the last one in order to reduce the number of distances to be calculated. The modified CM algorithm mainly differs in the construction and use of a family-tree F_{tree} which holds the localisation information of the balls, allowing much less computations. This algorithm creates a collection of balls with different size $\{B_j^g\}_{g,j}$, the corresponding local densities and the family-tree F_{tree}. The various terms used in the sequel are summarized below.

Terminology:

Terminal ball: A terminal ball is a ball that contains less than $N_{\min}$ data points.

Attached: A ball $B^g(x, d_c \times R)$ is attached to a previous generation ball $B^{g-1}(y, d_c)$ if $d(x, y) < d_c.(R + 1)$.

Daughters of B:$D(B)$, the daughters of the ball B are the next generation balls attached to it.

Mother of $\mathbf{B^g(x, d_c)}$: $M(B^g(x, d_c))$, the mother of B is the closest ball of generation $g-1$

containing x.

Mother of x: When the current generation is g, the mother of the data point x is the closest ball of generation $g-1$ containing x. It is denoted by $M(x)$.

Familly-tree: The family-tree contains the balls created at each generation and their links.

We enforce that if a data point x_i belongs to a ball B^g of generation g then it cannot belong to a ball of next generation $g+1$ unattached to B^g. Formally, if $x_i \in B_j^g$ and $x_i \in B_k^{g+1}$ then B_k^{g+1} is attached to B_j^g.

This implies that when we check a new observation at generation g in the CM procedure, we do not need to calculate the distances between it and all existing balls (their number can be very high), but we only need to compute the distances between it and the few balls of generation g attached to the mother ball (generation $g-1$) of x_i.

The first step of ICMDW (see Algorithm 3) is a standard CM iteration with a very large initial d_c^1. Then the modified CM algorithm (see Algorithm 4) is iteratively applied with a smaller window $d_c^2 = d_c^1.R$, where $R \in]0,1[$ is the coefficient which controls the decreasing speed of the window. At a given iteration g, the modified CM procedure works as follow. For each x_i it first constructs J, the set of indexes of existing balls to which x_i belongs to. But for that purpose, instead of comparing x_i with the whole set S of existing balls, it only searches among the reduced set s. s is constructed by loading the mother $m = M(x_i)$ of x_i and then by loading $s = D(m)$, the daughters balls of m. J can then be obtained by calculating the distances between x_i and the reduced set of balls s only. If $J \neq \emptyset$, the balls pointed by J increase their density count by one. Else, a new ball $B(x_i, d_c^g)$ is created and F_{tree} is updated by indicating to which balls $B(x_i, d_c^g)$ is attached. For this update, which happens only when a ball is created, we need to compute the distance between x_i and all the current generation existing balls to retain those closer than $d_c^g.(1+R)$. Finally, J is stored to be able to load its mother and to know if x_i belongs to terminal balls in the

next CM iteration. After all observations x_i has been processed, the balls with less than $N_{\min}$ points are set terminals and data points that belongs to only terminal balls are frozen (not used in the next iterations), reducing the number of data points processed at next the rounds. The frozen points from the last round are nevertheless used one last time. Indeed, data points frozen at iteration g are used at iteration $g+1$ to increase the density count of the ρ_i^{g+1}, but not to create new balls. This procedure is iterated and stops when there remain only terminal balls.

Density counts are re-evaluated through a last pass of the dataset. To do so, for each point x_i, we find the balls of first generation to which x_i belongs to, increase their density count by one, load the daughters of the closer one and repeat the process until x_i doesn't belong to any of the daughter balls. From those operations we can also obtain cc_i, for each data-point x_i, the closest and smaller terminal ball containing it (used for the labelisation phase). This re-evaluation is necessary since some balls that are created lately comes after the drawing of many points that should belong to them, implying an artificial small density count. Finally densities are rescaled with respect to the volume of the balls.

In [4] d_c is chosen such as $\bar{\rho} = 2\%N$. However, here it is the width of the window d_c^g which is indirectly defined by $N_{\min}$. Since we work in a large scale context and since the window is adaptive, we can choose $N_{\min}$ much lower than $2\%N$. From our experience, we propose $N_{\min} = 0.1\%N$ but it can be modified. It can be raised for faster but less accurate results, or parsimoniously lowered for expensive but more accurate results. Note that the accuracy is not of much importance for the cluster centers selection, but is determinant for the labelisation phase.

$R \in]0,1[$ controls the decreasing speed of the radius. When R is small, the balls will decrease faster implying better performances, but will increase the risks of creating balls that possess low ρ_i while the area may be dense. To overcome this problem $N_{\min}$ can be raised but it will imply a loss in details since many balls will not be deployed. Setting

Algorithm 3 Iterated CM With Decreasing Window

procedure ICMDW(d_c, $N_{\min}$, R)
  $d_c^1 \leftarrow R$
  $\{\{B_k^1\}_k, \{\rho_k^1\}_k\} \leftarrow \mathrm{CM}(d_c^1)$
  $g \leftarrow 1$
  while it remains at least a non-terminal ball **do**
    $g \leftarrow g + 1$
    $d_c^g \leftarrow d_c^{g-1}.R$
    $\{F_{\text{tree}}, \{\rho_k^g\}_k, S^g, TB^g\} \leftarrow \mathrm{CM}_{\text{modif}}(d_c, N_{\min}, R, F_{\text{tree}}, g, S^{g-1})$
  end while
  ▷ Density re-evaluation:
  for all $x_i \in \{1, ..., N\}$ **do**
    $s \leftarrow \{B_k^1\}_k$
    $g \leftarrow 1$
    while $J \neq \emptyset$ **do**
      $\overrightarrow{d} \leftarrow \mathrm{distance}(x_i, s)$
      $J \leftarrow \{j | \overrightarrow{d}[j] < d_c^g\}$
      $\forall j \in J$, $\rho_j^g \leftarrow \rho_j^g + 1$
      $cc_i \leftarrow B_{J[1]}^g$
      $g \leftarrow g + 1$
      $s \leftarrow D(J[1])$
    end while
  end for
  $\{\rho_k^g\} \leftarrow \{\rho_k^g / \mathrm{Volume}^g\}$
  return $\{F_{\text{tree}}, \{\rho_k^g\}_{g,k}, \{S^g\}_g, \{TB^g\}_g, \{cc_i\}_{i \in \{1,...,N\}}\}$
end procedure

Algorithm 4 CM modified

procedure $CM_{\text{MODIF}}(d_c,\, N_{\min},\, R,\, F_{\text{tree}},\, g)$
  $S \leftarrow \emptyset$
  for all not frozen x_i **do**
    $m \leftarrow M(x_i)$ ▷ Load mother of x_i
    $s \leftarrow D(m)$ ▷ Load daughters of M
    $\overrightarrow{d} \leftarrow \text{distance}(x_i, s)$
    $J \leftarrow \{j | \overrightarrow{d}[j] < d_c\}$
    if $J \neq \emptyset$ **then**
      $\forall j \in J, \rho_j^g(B_j^g) \leftarrow \rho_j^g(B_j) + 1$
    else
      $S^g \leftarrow S^g \cup B_k^g(x_i, d_c)$ ▷ Create new ball
      $\rho_k^g(B_k^g(x_i, d_c)) \leftarrow 1$
      $L \leftarrow \{l | \text{distance}(B_k(x_i, d_c), S^{g-1}[l]) < d_c.(1+R))\}$
      Update F_{tree}^g ▷ Record that $B(x_i, d_c)$ is attached to $\{B_l\}_{l \in L}$
      $J \leftarrow k$
    end if
    Store J in F_{tree}^g
  end for
  for all x_i frozen at last iteration **do**
    $m \leftarrow M(x_i)$ ▷ Load mother of x_i
    $s \leftarrow D(m)$ ▷ Load daughters of M
    $\overrightarrow{d} \leftarrow \text{distance}(x_i, s)$
    $\forall j \in J, \rho_j^g(B_j^g) \leftarrow \rho_j^g(B_j^g) + 1$
  end for
  $TB^g \leftarrow \{B_l^g | \rho_l^g(B_l^g) < N_{\min}\}_l$
  return $\{F_{\text{tree}}, \{\rho_k^g\}_k, S^g, TB^g\}$
end procedure

$R = \frac{3}{4}$ appeared to be a good trade-off according to the conducted experiments.

In order to quickly cover the domain with few balls in the first iteration, d_c^1 can be set to 1. It is justified by the fact that, since the dataset is normalized, $d_c^1 = 1$ is equal to the standards deviations $\{\sigma_j\}_{j=1,\dots,p}$ in each dimension. Moreover the choice d_0 has little impact then we do not need an accurate normalization. Therefore, we can only pseudo normalize the dataset, that is to say, estimate $\{\hat{\mu_j}\}_{j=1,\dots,p}$ then $\{\hat{\sigma_j}\}_{j=1,\dots,p}$ with a small subset part of the sample only and compute $x_{ij} = \frac{x_{ij}}{\hat{\sigma_j}}, \forall (i,j) \in N \times P$.

4 Cluster centers selection and labelisation using F_{tree}

This step mainly consists in finding, for each terminal ball, the closest one with higher density. Even if the number of points to be treated is decreased, it can remain too large to be directly processed by the standard peak detection presented in [17]. Instead of computing all the distances between all terminal balls, we use the localisation information provided by F_{tree} to compute distances for a few of them only.

This part is based on the following observation. Consider the k^{th} ball B_k^{G-1} of generation $G-1$ (where G is the last generation) and also consider its daughters $A_k^G = D(B_k^{G-1})$. We define C_k^{G-1} the "champion" of B_k^{G-1}, the ball of A_k^G who has the higher density. Then, excepted for the champion C_k^{G-1}, every daughter ball $B_j^G \in A_k^G$ possess a neighbor of higher density that is also belonging to A_k^G. For these ones, we can restrict the search to those few sister balls. However, for the champion C_k^G, we need to search farther. To do so, we apply quite similar steps between the champions: We first search among the sisters balls of B_k^{G-1} and find B_l^{G-1} (if it exists), the closest one which contains a higher density champion. We then search for the NHPD among the daughter balls of B_l^{G-1}. If there isn't any sister ball containing a higher density champion, we have to search farther again with the same procedure.

The complete algorithm of cluster center selection and labelisation consists in the fol-

lowing steps:

1) Champions election (see Algorithm 5)

This procedure constructs two new objects that share the same structure than F_{tree}: For each ball B_k^g, $\rho_{\text{champ},k}^g := \rho_{\text{champ}}^g(B_k^g)$ provides the density of its champion. $c_k^g \in \{0, ..., G-1\}$ indicate the champion's level of B_k^g. First, the c_k^g are initialized according to their generation ($c_k^g = G - g$). Then we begin at generation $G-1$. For each ball B_k^{G-1}, we find among its daughters the ball B_c^G which hold the highest density then we set the champion's level c_c^G of B_c^G to $G - (G-1) = 1$ and update the champion-densities by setting $\rho_{\text{champ}}^{G-1}(B_k^{G-1}) \leftarrow \rho^G(B_c^G)$. Afterwards, for each ball B_k^{G-2} of the generation $G-2$, we find among its daughters the ball B_c^{G-1} which hold the highest champion-density then set the champion's level of **the champion** carried by B_c^{G-1} to $G - (G-2) = 2$ and update the champion-densities by setting $\rho_{\text{champ}}^{G-2}(B_k^{G-2}) \leftarrow \rho_{\text{champ}}^{G-1}(B_c^{G-1})$. This procedure is repeated for generations $G-3, ..., 1$.

2) Constructions of δ_i

For each terminal ball, the Tree-Dive procedure (see Algorithm 6) is applied in order to obtain its NHPD and the distance δ separating them. For a level c champion ball B, this algorithm executes the following steps. It first load the ball B_i^{G-c} that has promoted B to the level c. Afterwards, the mother $B_k^{G-c-1} = M(B_i^{G-c})$ of B_i^{G-c} and then the daughters $s = D(B_k^{G-c-1})$ of B_k^{G-c-1} are loaded. $B_j^{G-c} \in s$, the closest one with higher champion-density than $\rho(B)$, is retained. There always exist a ball with higher champion's density, otherwise its champion's level should be at least $c+1$. If B_j^{G-c} is a terminal ball then the algorithm stops, if not the daughters of B_j^{G-c} are loaded and the closest one from B with higher champion-density is selected. If it is terminal then it stops, if not it keep diving into the tree until it falls into a terminal ball with higher density. The distance $\delta(B)$ separating this terminal ball and B is finally returned.

3) Cluster center selection and labelisation

$\gamma_i^g = \delta_i^g.\rho_i^g$ are computed, sorted and the turning-point-based decision rule from [11] presented in related work is applied to obtain the cluster centers. The selected centroids take each an unique label and the remaining balls receive the same label than their NHPD. Finally, each data point are labelled as the closest terminal ball containing it.

The complete procedure of IDPC is summarized in Algorithm 7.

Algorithm 5 Champions election

procedure ChampElect(F_{tree}, $\{\rho_k^g\}_{g,k}$, $\{B_k^g\}_{g,k}$)
  for $g \in \{G-1, ..., 1\}$ **do**
    for all B_l^g **do**
      if B_l^g is terminal **then**
        $\rho_{\text{champ},l}^g(B_l^g) \leftarrow \rho_l^g(B_l^g)$
        $c_l^g \leftarrow G - g$ ▷ Upgrade champions level
      else
        $B_j^{g+1} \leftarrow \text{argmax}_{B_i^{g+1} \in D(B_l^g)}(\rho_{\text{champ},i}^{g+1}(B_i^{g+1}))$
        $\rho_{\text{champ},l}^g(B_l^g) \leftarrow \rho_{\text{champ},j}^{g+1}(B_j^{g+1})$
        $c_j^{g+1} \leftarrow G - g$
      end if
    end for
  end for
  return $(\{\rho_{\text{champ},k}^g\}_{g,k}, \{c_k^g\}_{k,g})$
end procedure

5 Experimental results

The generated dataset is composed by 1024508 bidimensional points in the rectangular domain $[-300, 300] \times [-300, 300]$. It contains 13 clusters, its distribution can be seen in the figure 1. The ICMDW has been applied with $N_{\min} = 0.1\%N$ and $R = \frac{3}{4}$.

It produced a set of 9224 balls (figure 2), including the 4622 terminal balls (figure 3), all generations confounded. The total number of generations was 16.

Algorithm 6 Tree-dive

procedure TREE-DIVE(B,F_{tree}, c, $\{\rho^g_{\text{champ,k}}\}_{g,k}$)
  $T \leftarrow False$
  Load B_M ▷ mother of the ball that promoted B to level c
  while $T == False$ **do**
    Load $s \leftarrow D(B_M)$ ▷ daughters of B_M
    $s^c_+ \leftarrow \{B^c_i | \rho^c_{\text{champ},i}(B^c_i) > \rho(B)\}$
    $j \leftarrow \text{argmin}_i(\text{distance}(B, s_+[i]))$
    if B^c_j is a terminal Ball **then**
      $T \leftarrow True$
    else
      $B_M \leftarrow B^c_j$
      $c \leftarrow c + 1$
    end if
  end while
  nhpd $\leftarrow B^c_j$
  $\delta \leftarrow \text{distance}(B, B^c_j)$
  return $\{\delta, \text{nhpd}\}$
end procedure

The density map is represented in the figure 4 over two different points of view. To construct this graphic we have created a grid of points covering the domain and associated, for each point of the grid, the density of the closest terminal ball containing it, to finally plot the surface. The density is well represented however there is some moderatly dense balls wich are supposed to carry a null or very low density. This problem that we call aliasing occurs in low density areas located near an abrupt peak. An aliasing ball is a ball that has been created lately (at a given generation), preventing it to increase its density count with the points that have been drawn before its creation. Therefore it is set terminal and get back its density at the density reevaluation phase. Even if the density map doesn't reflect perfectly the true density, the aliasing can only create relatively low density balls (because of the large width of the window d^g_c) near a much higher density peak. So its impact on clustering is almost inexistent.

The cluster centers can now be selected. We can see in the decision graph (figure 5) 13 points standing out the others. They are automatically detected with the turning-point

Algorithm 7 IDPC

procedure IDPC
$N_{\min} \leftarrow 0.1\%N$
$d_c^1 \leftarrow 1$
Pseudo Normalize
Randomize
$\{\{B_k^g\}_{g,k}, F_{\text{tree}}, \{\rho_k^g\}_{g,k}, \{TB^g\}_g, \{cc_i\}_i\} \leftarrow ICMDW(d_c^1, N_{\min}, R)$
$\{\{\rho^g_{\text{champ},k}\}_{g,k}, \{c_k^g\}_{k,g}\} \leftarrow \text{CHAMPELECT}(F_{\text{tree}}, \{B_k^g\}_{g,k}, \{\rho_k^g\}_{g,k})$
for all (g,k) **do**
$\{\delta_k^g, \text{nhpd}_{g,k}\} \leftarrow \text{TREE-DIVE}(B_k^g, F_{\text{tree}}, c_k^g, \{\rho^g_{\text{champ,k}}\}_{g,k})$
end for
$\gamma \leftarrow \text{sort}(\delta.\rho)$
for $i \leftarrow \{1, ..., m\}$ **do**
$k_i^1 \leftarrow \frac{\gamma_i - \gamma_1}{i-1}$
$k_1^{i-1} \leftarrow \frac{\gamma_{i+1} - \gamma_i}{1}$
end for
$tp \leftarrow \text{argmax}_{i \in \{1,...,m\}}(\frac{k_i^1}{k_1^{i-1}})$ ▷ Turning-point
for all $j \leq tp$ **do**
$\text{Label}[j] \leftarrow j$
end for
for all $j \geq tp + 1$ **do** ▷ Labelise Balls
$\text{Label}[j] \leftarrow \text{Label}[\text{nhpd}_j]$
end for
for $i \leftarrow \{1, ..., N\}$ **do** ▷ Labelise datapoints
$\text{Label}[i] \leftarrow \text{Label}[cc_i]$
end for
return Label
end procedure

procedure. The sorted gamma values are represented in the figure 5 and the selected cluster centers are represented in red and their location and the corresponding clustering results obtained by propagation of the label are showed in the figure 6.

Note that we didn't have generated any flat clusters (like an uniform) because the original algorithm, and so our, isn't able to deal with. In those situations, the algorithm may detect several peaks for only one real cluster. In a future work, we will benefit from the reduced set of pretender centroids and their localisation information provided by our algorithm to develop a better cluster center selection. To do so we might not only take into account the distance separating two peaks but rather the density connectivity between them. In other

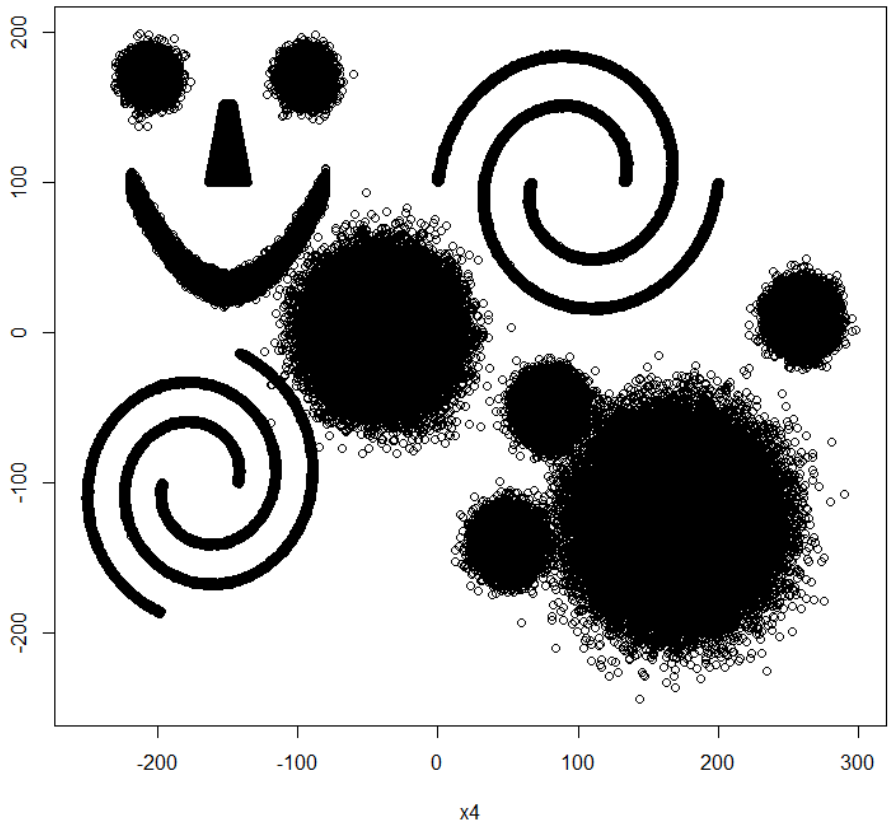


Figure 1. Generated dataset

words, searching if there is a dense trajectory to reach one peak from an another one.

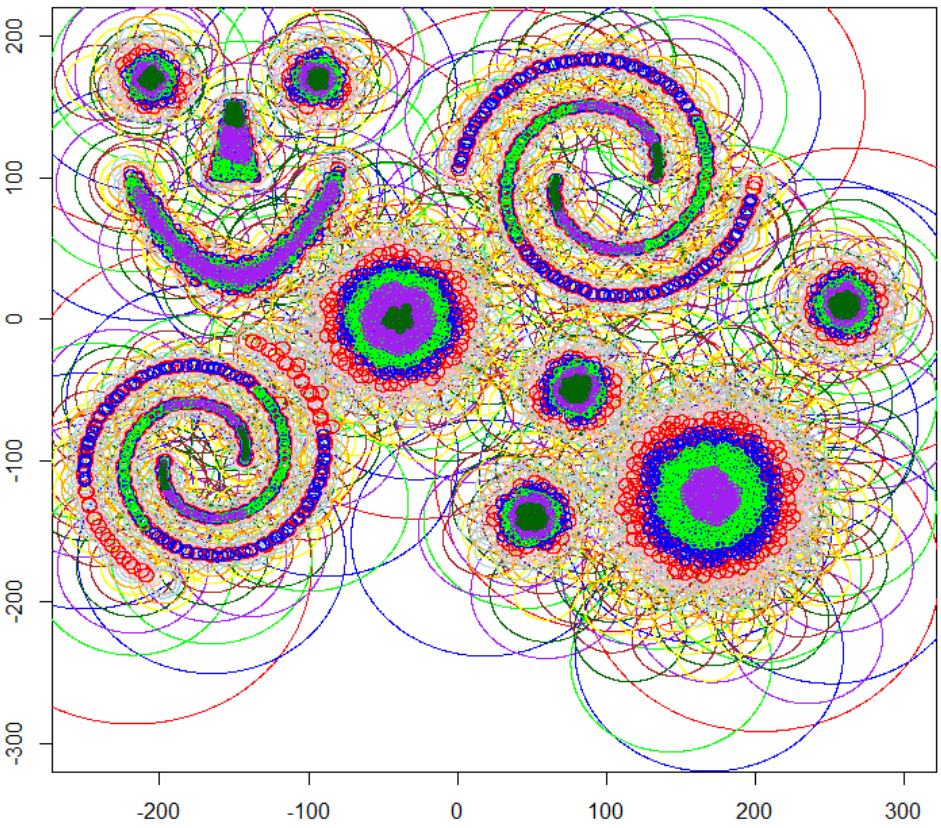


Figure 2. All the balls

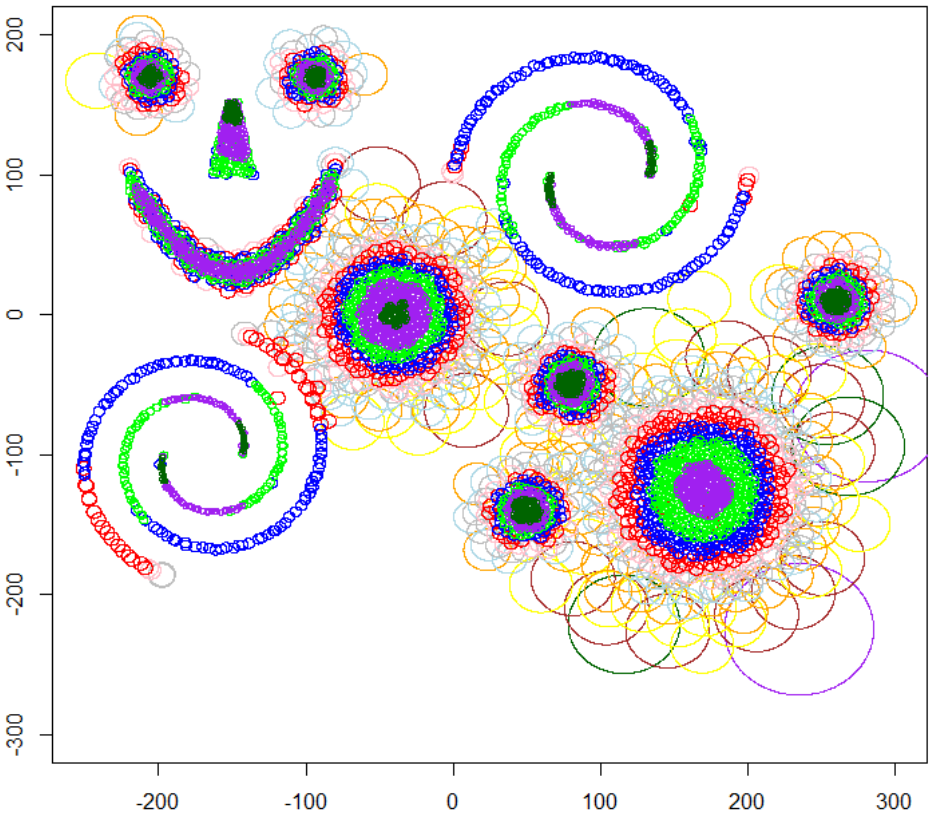


Figure 3. Terminal balls only

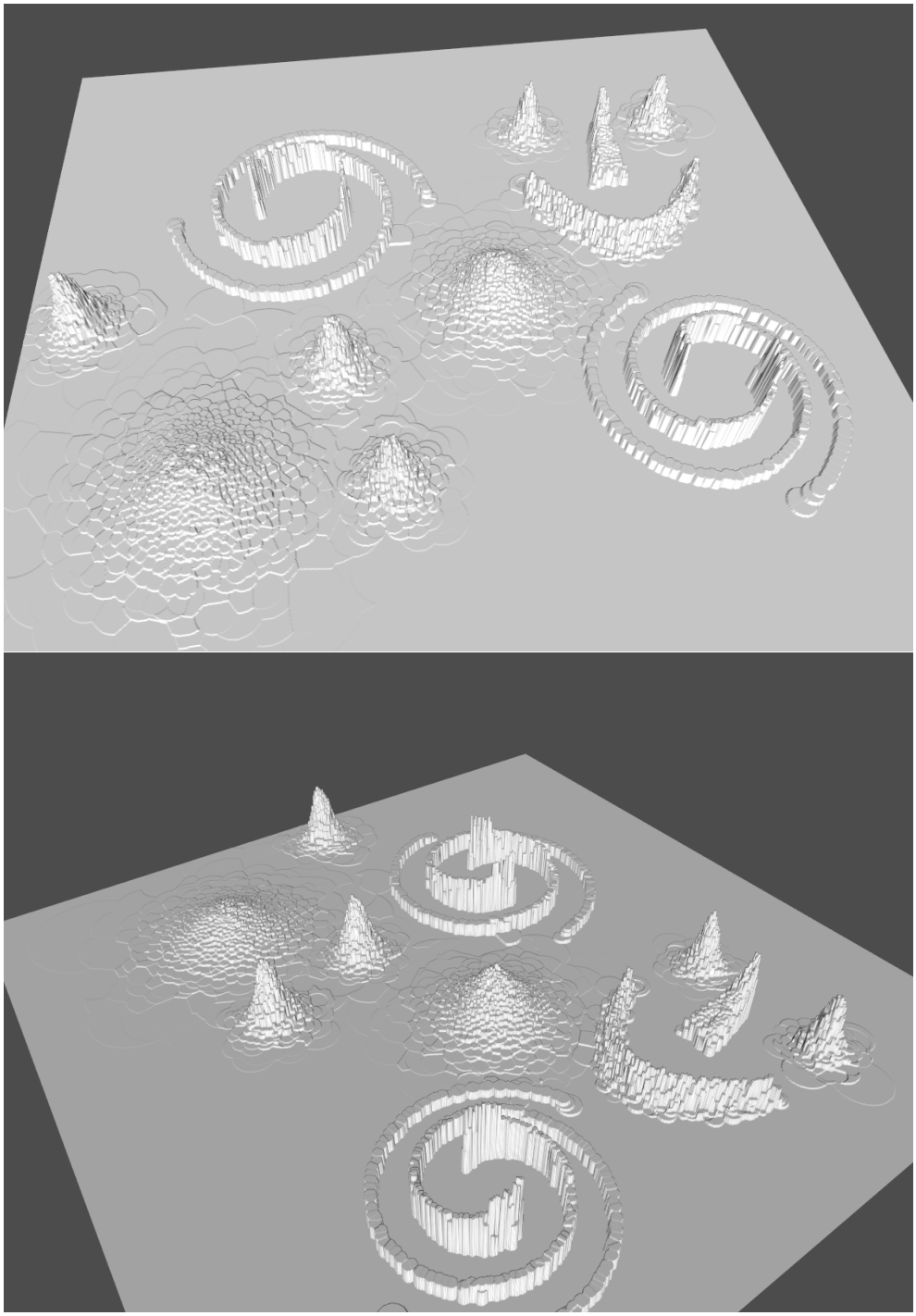
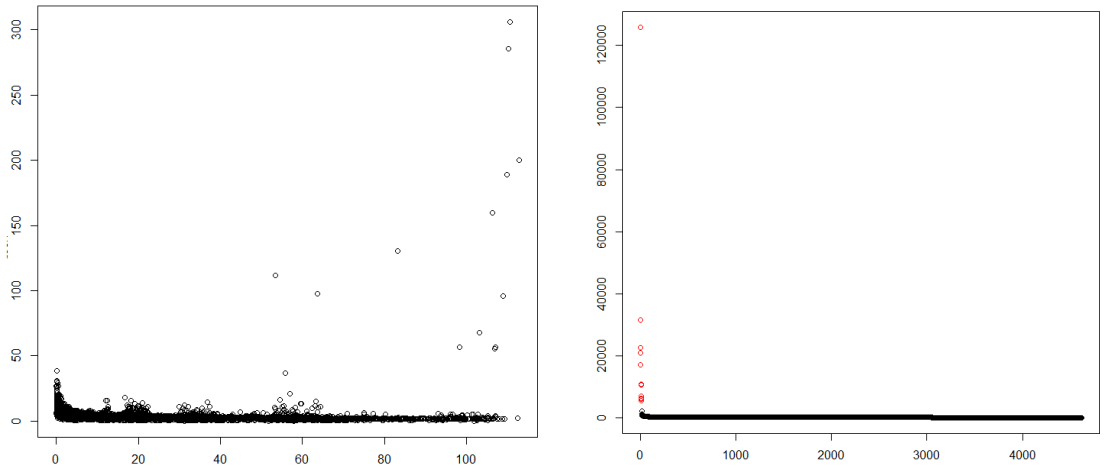
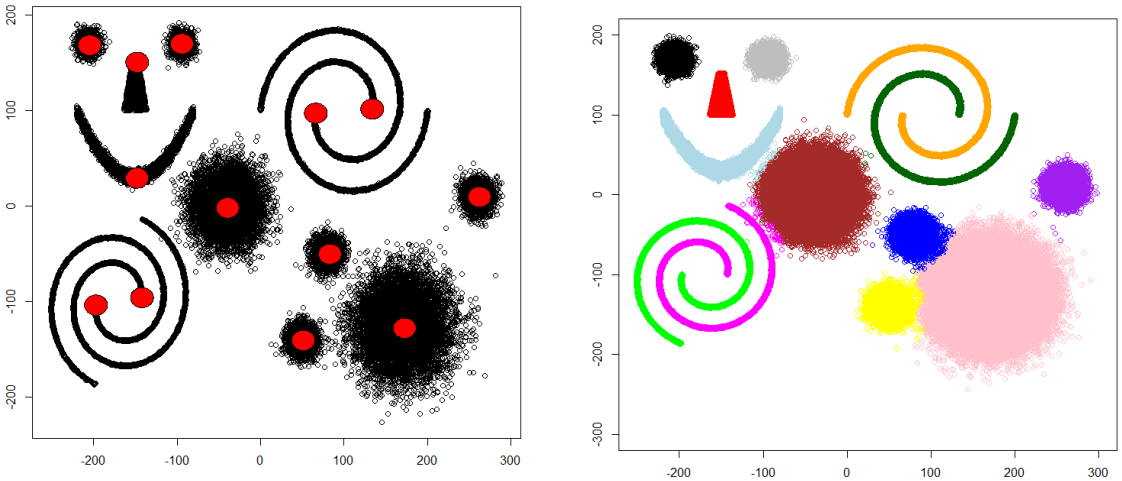


Figure 4. Density map



Decision graph Turning point

Figure 5. Center selection



Cluster centers Labelled dataset

Figure 6. Clustering results

6 Conclusion and future work

Many extensions of FDPC have been developed but none is able to deal with multidimensional and large datasets that arise in many pratical data analysis applications. In this paper we have presented an efficient algorithm able to manage those kind of data.

For that purpose, we have designed an iterative procedure which creates both localisation and density map which gives, in a computationaly efficient way, the densities ρ_i and the distances δ_i.

It is adaptive, the results being barely influenced by the few free parameters to be tuned, so that the automatic choice is often satisfactory. This last point is a dramatic improvement over most of the existing algorithms that tend to depend critically on the initial conditions.

An important remaining work to be performed is the recoding of the procedure in a compiled language with optimized data structures, from which a gain of at least an order of magnitude is expected. Moreover we aim to develop a better cluster selection unlocked by our intermediate results, able to manage flat distributions. Finally, we expect to delete the aliasing in the density map.

References

[1] Dongxia Chang, Yao Zhao, Lian Liu, and Changwen Zheng. A dynamic niching clustering algorithm based on individual-connectedness and its application to color image segmentation. *Pattern Recognition*, 2016.

[2] Zhensong Chen, Zhiquan Qi, Fan Meng, Limeng Cui, and Yong Shi. Image segmentation via improving clustering algorithms with density and distance. *Procedia Computer Science*, 55:1015–1022, 2015.

[3] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995.

[4] Haizhou Du et al. Robust local outlier detection. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 116–123. IEEE, 2015.

[5] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[6] Sen Jia, Guihua Tang, and Jie Hu. Band selection of hyperspectral imagery using a weighted fast density peak-based clustering approach. In *International Conference on Intelligent Science and Big Data Engineering*, pages 50–59. Springer, 2015.

[7] Shuai Li, Xiaofeng Zhou, Haibo Shi, and Zeyu Zheng. An efficient clustering method for medical data applications. In *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2015 IEEE International Conference on*, pages 133–138. IEEE, 2015.

[8] Zhou Liang and Pei Chen. Delta-density based clustering with a divide-and-conquer strategy: 3dc clustering. *Pattern Recognition Letters*, 73:52–59, 2016.

[9] Dongchang Liu, Shih-Fen Cheng, and Yiping Yang. Density peaks clustering approach for discovering demand hot spots in city-scale taxi fleet dataset. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 1831–1836. IEEE, 2015.

[10] Peiyu Liu, Yingying Liu, Xiuyan Hou, Qingqing Li, and Zhenfang Zhu. A text clustering algorithm based on find of density peaks. In *2015 7th International Conference on Information Technology in Medicine and Education (ITME)*, pages 348–352. IEEE, 2015.

[11] Chun-lai Ma, Tao Ma, and Hong Shan. A new important-place identification method. In *Computer and Communications (ICCC), 2015 IEEE International Conference on*, pages 151–155. IEEE, 2015.

[12] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[13] Rashid Mehmood, Rongfang Bie, Hussain Dawood, and Haseeb Ahmad. Fuzzy clustering by fast search and find of density peaks. In *2015 International Conference on Identification, Information, and Knowledge in the Internet of Things (IIKI)*, pages 258–261. IEEE, 2015.

[14] Rashid Mehmood, Guangzhi Zhang, Rongfang Bie, Hassan Dawood, and Haseeb Ahmad. Clustering by fast search and find of density peaks via heat diffusion. *Neurocomputing*, 2016.

[15] Li Miao. Comparative analysis of two clustering algorithms: K-means and fsdp (fast search and find of density peaks). 2015.

[16] Yan Qin, Chunhui Zhao, and Furong Gao. An iterative two-step sequential phase partition (itspp) method for batch process modeling and online monitoring. *AIChE Journal*, 62(7):2358–2373, 2016.

[17] Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, 2014.

[18] Yong Shi, Zhensong Chen, Zhiquan Qi, Fan Meng, and Limeng Cui. A novel clustering-based image segmentation via density peaks algorithm with mid-level feature. *Neural Computing and Applications*, pages 1–11.

[19] Guangtao Wang and Qinbao Song. Automatic clustering via outward statistical testing on density metrics.

[20] Shuliang Wang, Dakui Wang, Caoyuan Li, Yan Li, and Gangyi Ding. Clustering by fast search and find of density peaks with data field. *Chinese Journal of Electronics*, 25(3):397–402, 2016.

[21] Xiao-Feng Wang and Yifan Xu. Fast clustering using adaptive density peak detection. *Statistical methods in medical research*, page 0962280215609948, 2015.

[22] Yi Wang, Qixin Chen, Chongqing Kang, and Qing Xia. Clustering of electricity consumption behavior dynamics toward big data applications.

[23] Zhang WenKai and Li Jing. Extended fast search clustering algorithm: Widely density clusters, no density peaks.

[24] Juanying Xie, Hongchao Gao, Weixin Xie, Xiaohui Liu, and Philip W Grant. Robust clustering by detecting density peaks and assigning points based on fuzzy weighted k-nearest neighbors. *Information Sciences*, 354:19–40, 2016.

[25] Jia Yu, Lei Xie, Xiong Xiao, Eng Siong Chng, and Haizhou Li. A density peak clustering approach to unsupervised acoustic subword units discovery. In *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 178–183. IEEE, 2015.

[26] Ruisheng Zhang, Huiyi Ma, Qidong Liu, and Zhili Zhao. An improved fast search clustering algorithm based on kernel density. In *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, pages 689–693. IEEE, 2015.