DCF: An Efficient and Robust Density-Based Clustering Method

Joshua Tobin

School of Computer Science & Statistics
Trinity College Dublin, Dublin, Ireland
Email: tobinjo@tcd.ie

Mimi Zhang

School of Computer Science & Statistics
Trinity College Dublin, Dublin, Ireland
Email: mimi.zhang@tcd.ie

Abstract—Density-based clustering methods have been shown to achieve promising results in modern data mining applications. A recent approach, Density Peaks Clustering (DPC), detects modes as points with high density and large distance to points of higher density, and hence often fails to detect low-density clusters in the data. Furthermore, DPC has quadratic complexity. We here develop a new clustering algorithm, aiming at improving the applicability and efficiency of the peak-finding technique. The improvements are threefold: (1) the new algorithm is applicable to large datasets; (2) the algorithm is capable of detecting clusters of varying density; (3) the algorithm is competent at deciding the correct number of clusters, even when the number of clusters is very high. The clustering performance of the algorithm is greatly enhanced by directing the peak-finding technique to discover modal sets, rather than point modes. We present a theoretical analysis of our approach and experimental results to verify that our algorithm works well in practice. We demonstrate a potential application of our work for unsupervised face recognition.

Index Terms—Clustering, Density Peaks, Modal Sets.

I. INTRODUCTION

Density Peaks Clustering (DPC) [1] is a popular and intuitive mode-seeking clustering algorithm, in which cluster centers are selected as points of high local density with a relatively large distance from any point of higher local density. For each data point, two quantities are computed: its local density, defined as the number of neighbors within a threshold distance, and its distance from points of higher density. Centers are chosen as the points with maximal product of the local density and distance to the closest point of higher density, henceforth referred to as the peak-finding criterion. After the cluster centers have been found, each remaining point is assigned to the same cluster as its nearest neighbor of higher local density.

DPC is one of a number of prominent mode-seeking clustering algorithms. Alternative methods include MeanShift [2], [3] and QuickShift [4]. MeanShift uses the Kernel Density Estimate (KDE) to iteratively update candidate centers until convergence. In the QuickShift method, each sample is repeatedly moved to its nearest neighbor with higher local density if one exists within a τ -radius ball until termination. As such, in both approaches each point terminates at a local mode of the density. All points that have terminated at the same mode are assigned to the same cluster. While the three methods share the same allocation procedure, DPC avoids the need to explicitly maximise the density field for each data point, as in MeanShift,

and the need to specify a threshold distance for which two instances should be not clustered together, as in QuickShift.

A key drawback of the DPC procedure is that the points with maximal values of the peak-finding criterion are often poor representations of the clusters. This is most prevalent when the data contain both high- and low-density clusters. The DPC criterion often leads to multiple points from high-density clusters being erroneously selected as centers, while true clusters of lower density are ignored. Previous attempts to improve the detection of low-density modes involve incorporating information about points of lower local density into the density [5] and the distance [6]. However, these approaches are susceptible to variations in the underlying density estimation. To formulate a robust mode-finding procedure, we direct the DPC procedure to identify locally high density regions of the dataset.

QuickShift++ [7], [8] improves on both MeanShift and QuickShift by modelling locally high-density regions using cluster cores. Cluster cores extend the concept of point modes to sets of points of arbitrary shape, size and density level. Cluster cores are parametrized by $\beta \in (0, 1)$, which determines how much the density can fluctuate within a cluster. Using cluster cores instead of point modes reduces the risk of selecting multiple centers from a high-density cluster. Furthermore, cluster cores are less sensitive to the chance variation that occurs in the empirical density estimate and better capture locally high-density regions.

We introduce Density Core Finding (DCF), a novel clustering method that uses the peak-finding criterion to estimate these cluster cores. The peak-finding criterion is computed for each point, and the instance with maximum value is selected as a center. The cluster core containing this center is then found, and all instances belonging to it are removed from consideration as potential centers. The algorithm continues detecting cluster cores until no more remain in the data. The allocation procedure is unchanged from DPC, each non-center point is allocated to the same cluster as its nearest neighbor of higher local density. The benefits offered by DCF are illustrated in Figure 1.

In our theoretical analysis, we discuss the necessary conditions for DCF to be guaranteed to recover all of the true modes in the data, as well as the difficulties posed by the peak-finding criterion. To our knowledge, this is the first

Fig. 1: An illustration containing two clusters, one high-density cluster with two modes and one low-density cluster. Left: DPC incorrectly selects both centers from the first cluster, as the peak-finding criterion favours high-density points. Right: Cluster cores, represented by dashed lines, better represent the cluster centers.

work to give guarantees on mode recovery for an algorithm using the peak-finding framework. We demonstrate that DCF recovers clusters of the same quality as QuickShift++, while being significantly faster in execution. The improved quality of the clustering results from DCF compared to a number of competitor density-based clustering methods, including DPC, QuickShift++ and DBSCAN, is illustrated using a range of real-world datasets. Finally, we demonstrate the superiority of DCF over competitor methods in a popular application of density-based clustering: large-scale face recognition.

The remainder of the paper is organised as follows: in Section II we survey the field of density-based clustering; in Section III we describe DCF in detail; in Section IV we analyze the mode-recovery performance of the DCF method; in Section V we present an experimental evaluation; in Section VI we demonstrate an application of DCF; finally, in Section VII we conclude and discuss potential avenues for future research.

II. RELATED WORK

Density-based clustering is a nonparametric approach to clustering where clusters are taken to be high-density regions of the unknown density. Density-based clustering methods, including DBSCAN [9] and MeanShift [2], do not make assumptions of the form of the underlying density of the dataset. Consequently, density-based clusters are not necessarily groups of points that have low within-cluster dissimilarity, and thus can be of any shape.

Density-based clustering methods can be broadly segregated into two categories, (1) hill-climbing methods, such as DPC, MeanShift, QuickShift, and QuickShift++, and (2) density-level set approaches. Density-level set methods imagine clusters as the sets of points resulting from a cut through the probability density function at a certain density level. Performing these cuts induce connected regions in the feature space where the density is greater than the cut threshold. These points in each region form the density-based clusters. This form of clustering was first developed in [10]. There exists an extensive literature further developing level set methods [11]–[15]. These works discuss how mutual k -NN graphs estimate level sets, and provide guarantees on the consistency of this

estimation. Guidance is also provided in [11] regarding the optimal choice of k for identifying clusters in the data.

DBSCAN operates as a level set algorithm by categorizing points as core points (in a cluster) or outlying points (outside all clusters) based on a density threshold. Level set methods are not well suited to clusters of different densities. In such a case, determining the suitable cut level becomes impossible, either ignoring low-density clusters or incorrectly merging high-density clusters. OPTICS [16] is an attempt to improve the performance of DBSCAN for problems of this type. As in DBSCAN, OPTICS finds core samples of high density and builds clusters from them. OPTICS produces a hierarchical ordering of the data, based on distances from points to clusters, from which final clusterings can be extracted. Hierarchical DBSCAN (HDBSCAN) [17], creates a hierarchy of DBSCAN clusterings for all values of the density threshold. A key development of this approach is the extraction of clusters at different densities using an optimization algorithm applied to the cluster hierarchy.

Extensions of the DPC algorithm abound in the literature. Most research aims to replace the original density estimate, which has $O(n^2)$ complexity, with a less intensive estimation method. Recent attempts include kernel density estimation [18], nearest neighbor-based density functions [19], [20], and density estimates using the ratio of maximal distance to average distance in the set of nearest neighbors [21]. The scale of the density estimate has great influence on the selection of centers in the DPC algorithm. The selection of centers from the decision graph has been automated, using criteria such as the silhouette index and the generalized value distribution [18], [22], [23]. As discussed above, existing attempts to improve the performance of DPC on clusters of different densities are vulnerable to errors arising from small fluctuations, and the problem of reliably detecting such clusters remains.

The notion of cluster cores used in this work was first developed in [7]. The authors provide an alternate notion of clusters, called modal-sets. The procedure they introduce estimates the modal-sets by searching the instances in descending order of local density. For each instance, the level set at the level of its local density is found. If the subset of the level set containing the point is disconnected from all previous modal-set estimates, it is accepted as a modal-set. Once all points have been assessed, the points not contained in modal-sets are assigned to the cluster of their nearest modal-set. The authors provide consistency guarantees on the recovery of true modal-sets in the data. A subsequent work develops QuickShift++ [8], which improves on the procedure of [7] in several ways. Firstly, the concept of cluster cores extend modal-sets by allowing the empirical density within a cluster core to vary. This better accounts for fluctuations in the density observed in real data. The allocation of QuickShift++ is also improved, using a hill-climbing procedure to allocate points to the same cluster as their nearest neighbor of higher local density.

Cluster cores have obvious benefits for the DPC procedure, resolving the issues caused by clusters of varying density. Incorporating cluster cores into the DPC procedure also leads

Fig. 2: A 2-D simulated example containing two clusters, one high-density cluster (the upper moon) and one low-density cluster (the lower moon). Left: DPC incorrectly selects both centers from the high-density cluster. Right: DCF uses the darkened points as the respective cluster cores, resolving the error in clustering.

to improvements over the QuickShift++ method. Firstly, the peak-finding criterion provides a more efficient ordering for the search of level sets than empirical density alone. Coupled with the termination criteria to be introduced later, we demonstrate that the number of assessments completed is reduced on average by 98%.

III. OUR METHOD

An example of the operation of DCF is illustrated in Figure 2. The figure depicts a dataset containing two clusters, a high-density cluster (top) and a low-density cluster (bottom). The DPC method, as seen on the left of the figure, proceeds by searching for the points with maximal values of the peak-finding criterion, selecting first the blue peak, the point of maximum density. DPC erroneously selects the second center also from the top cluster. The allocation mechanism incorrectly assigns all points in the bottom cluster. For this example, seven points in the top cluster have larger value of the peak-finding criterion than the maximum value in the bottom cluster.

The DCF procedure can be seen on the right of the figure. We first select the instance of maximum density as the first peak. However, DCF proceeds to compute the cluster core associated with this point, the highlighted blue points, and remove all elements of the core from consideration as centers. Of those remaining, the point with maximal value of the peak-finding criterion is in the bottom cluster. The associated cluster core is visible in red. As this cluster core is disconnected from the first cluster core, it is accepted as a valid cluster core. The algorithm's termination procedure is invoked when assessing a third center. The third center is selected as before; however, as the cluster core associated with this point contains all of the instances in the dataset, the algorithm terminates.

A. Notation and definitions

We consider a dataset $\mathbf{X}$ consisting of n data points in $\mathbb{R}^d$ drawn from an unknown density f with compact support $\mathcal{X}$. In this work, we take $\|\cdot\|$ to be the Euclidean norm. We use

TABLE I: The notation used in this work.

Notation	Explanation
$\mathbf{X}$	A set of n points in $\mathbb{R}^d$.
f	The unknown density.
$\mathcal{X}$	The compact support of f .
$r_k(x)$	The distance from a point x to its k -th nearest neighbor.
$f_k(x)$	The k -NN density estimate at a point x .
$b(x)$	The nearest point to x with higher local density.
$\delta_k(x)$	The distance from x to $b(x)$.
$\gamma_k(x)$	The peak-finding criterion.
M	A cluster core.
β	The fluctuation parameter for cluster cores.
G	The mutual k -NN graph of $\mathbf{X}$.
$G(\lambda)$	The λ -level set of G .
$\mathcal{M}$	The set of estimated cluster cores.
$\hat{\mathcal{C}}$	The set of clusters.

a k -nearest neighbors (k -NN) density estimator. For a data point $x \in \mathbf{X}$, let $r_k(x)$ be the distance between x and its k -th nearest neighbor. The density estimate used is a simple functional of the distance $r_k(x)$.

Definition 1. For every $x \in \mathbb{R}^d$, let $r_k(x)$ denote the distance from x to its k -th nearest neighbor in $\mathbf{X}$. The density estimate is given as

$$f_k(x) := \frac{k}{n \cdot v_d \cdot r_k(x)^d},$$

where v_d is the volume of the unit sphere in $\mathbb{R}^d$.

Guarantees on the quality of this density estimate are available in [24]. As well as a density estimate, the peak-finding criterion requires the distance from each point to its nearest neighbor of higher density:

Definition 2. For the point $x = \arg \max_{x \in \mathbf{X}} f_k(x)$, we define the quantity

$$\delta_k(x) = \max_{x' \in \mathbf{X}} \|x - x'\|.$$

For the remaining points, let $b(x) = \arg \min_{x' \in \mathbf{X}} \{\|x - x'\| : f_k(x) < f_k(x')\}$, i.e. the nearest neighbor of x with higher density. Define the distance to the nearest neighbor of higher local density as

$$\delta_k(x) = \|x - b(x)\|.$$

Next, we formalize the definition of the peak-finding criterion:

Definition 3. Taking $f_k(x)$ and $\delta_k(x)$ as defined above, we define the peak-finding criterion, $\gamma_k(x)$ as

$$\gamma_k(x) = f_k(x) \cdot \delta_k(x).$$

Following [8], we define the cluster core with respect to a fixed fluctuation parameter β . In what follows, a connected set refers to a set which cannot be divided into two disjoint non-empty closed sets.

Definition 4. Let $0 < \beta < 1$. A closed and connected set $M \subset \mathcal{X}$ is a cluster core if M is a maximal connected subset of $\{x \in \mathcal{X} : f(x) \geq (1 - \beta) \cdot \max_{x' \in M} f(x')\}$.

Varying the fluctuation parameter β determines the magnitude of the cluster cores. It is also shown in [8] that if M_1, M_2 are distinct cluster cores of f then $M_1 \cap M_2 = \emptyset$, namely cluster cores do not overlap.

To estimate the cluster cores, we use the level sets of the mutual k -NN graph. The mutual k -NN graph, G , is an undirected graph with the points of $\mathbf{X}$ as vertices and an edge between x and x' if $\|x - x'\| \leq \min\{r_k(x), r_k(x')\}$. The level sets of G are defined as

Definition 5. Let $G(\lambda)$ denote the λ -level of the mutual k -NN graph G with vertices $\{x \in \mathbf{X} : f_k(x) \geq \lambda\}$ and an edge between x and x' if an edge exists between x and x' in G .

It has been shown in [25] that the λ -level connected sets of f are well approximated by the connected components of $G(\lambda)$. As the value of λ decreases, the connected components of $G(\lambda)$ are hierarchically nested.

B. DCF Algorithm

Algorithm 1: DCF ALGORITHM

Input: Neighborhood parameter k , fluctuation parameter β .

Output: A set of clusters $\hat{\mathcal{C}}$

Initialisation: $\hat{\mathcal{M}} = \emptyset$, $Assessed = \emptyset$, G_C a directed graph with points $\{x_1, \dots, x_n\}$ as vertices and no edges.

- 1: Sort the x 's in decreasing order of γ_k values.
- 2: Find $x = \arg \max_{x \in \mathbf{X}} \gamma_k(x)$.
- 3: Define $\lambda := f_k(x)$.
- 4: Let $A_\beta(x)$ be the connected component of $G(\lambda - \beta\lambda)$ containing x .
- 5: Add $A_\beta(x)$ to $\hat{\mathcal{M}}$ and to $Assessed$.
- 6: **repeat**
- 7: Find $x = \arg \max_{x \in \mathbf{X}} \{\gamma_k(x) : x \notin Assessed\}$.
- 8: Define $\lambda := f_k(x)$.
- 9: Let $A_\beta(x)$ be the connected component of $G(\lambda - \beta\lambda)$ containing x .
- 10: Add $A_\beta(x)$ to $Assessed$.
- 11: **if** $A_\beta(x)$ is disjoint from all cluster cores in $\hat{\mathcal{M}}$ **then**
- 12: Add $A_\beta(x)$ to $\hat{\mathcal{M}}$.
- 13: **end if**
- 14: **until** $\mathbf{X} \subseteq Assessed$.
- 15: **for** $i = 1$ to n **do**
- 16: If x_i is not in a cluster core, add an edge to G_C from x_i to $b(x_i)$.
- 17: **end for**
- 18: **for** each cluster core $M \in \hat{\mathcal{M}}$ **do**
- 19: let C_M be the points $x \in \mathbf{X}$ whose directed path in G_C terminates in M .
- 20: Add the cluster $C_M \cup M$ to $\hat{\mathcal{C}}$.
- 21: **end for**
- 22: **return** $\hat{\mathcal{C}}$

The DCF algorithm takes as input the dataset $\mathbf{X}$ and uses parameters k and β to return the final set of clusters $\hat{\mathcal{C}}$. Initially, the set of estimated cluster cores $\hat{\mathcal{M}} = \emptyset$ and the cluster assignment graph G_C is initialised with vertices as the points of $\mathbf{X}$ and no edges. DCF computes the peak-finding criterion for each point and selects the instance x with maximal value (Lines 1-2). The density level is set to $\lambda - \beta\lambda$ where $\lambda = f_k(x)$. The level set graph $G(\lambda - \beta\lambda)$ is found, and the connected component $A_\beta(x)$ containing x is the first cluster core and is added to $\hat{\mathcal{M}}$. All points from $A_\beta(x)$ are added to $Assessed$ and thus excluded from further consideration (Lines 3-5).

Next, the instance x with maximal value of the peak-finding criterion yet to be assessed is selected. The density level is set to $\lambda - \beta\lambda$ where $\lambda = f_k(x)$ (Lines 7-8). The level set graph $G(\lambda - \beta\lambda)$ is updated and the connected component of $G(\lambda - \beta\lambda)$ containing x , namely $A_\beta(x)$, is found. Firstly, all points in $A_\beta(x)$ are added the set $Assessed$ and hence excluded from future consideration as cluster cores (Lines 9-10). Then, if $A_\beta(x)$ is disjoint from all sets in $\hat{\mathcal{M}}$, then $A_\beta(x)$ is added to $\hat{\mathcal{M}}$ (Lines 11-13). The algorithm proceeds adding will be treated as a cluster core and cluster cores to $\hat{\mathcal{M}}$ until the termination criteria is met, i.e., all points have been added to $Assessed$ (Line 14).

After the set of estimated cluster cores has been returned, edges are added to the graph G_C from each non-core point x to $b(x)$ (Lines 15-17). The points in a cluster core M together with all the vertices that have paths terminating in M form a cluster that is added to $\hat{\mathcal{C}}$ (Lines 18-22). Proceeding in this way, each sample point will be assigned to a unique cluster.

IV. ANALYSIS OF DCF

DCF selects the first point in the ordered sequence (of remaining points) to identify the cluster core and then removes all the points in the cluster core from the ordered sequence. As DCF assesses only a small fraction ($\approx 2\%$) of the data points, and the scale of the distance $\delta_k(x)$ is problem specific, it is not guaranteed that DCF will detect all the cluster cores.

A. Peak-Finding and Mode Recovery

Here, we provide guarantees on the detection of disconnected clusters with DCF. We describe a problematic case in which DCF will fail to detect the true clusters in the data, and subsequently demonstrate why such a case is unlikely to occur in real datasets. Finally, we provide guarantees related to the termination criterion used in DCF.

Proposition 1. Any cluster that corresponds to a connected component in the mutual k -NN graph will be recovered by DCF.

Proof. If the cluster corresponds to a connected component in the mutual k -NN graph G , the related cluster core is a sub-graph of the connected component and hence is disconnected with other components in the graph. The if statement in Line 11 of Algorithm III-B will always be satisfied. According to the definition of a mutual k -NN graph, the points of the cluster

that are not in the cluster core will always be linked to a point in the cluster core by a directed path. $\square$

It is immediately clear from Prop. 1 that

Corollary 1. *If all clusters are mutually disconnected in the graph G , then DCF will recover the exact clustering.*

We here describe a scenario in which DCF fails to detect the true clusters in the data. Consider the termination point x_T , with density $f_k(x_T) = \lambda_T$, and distance to its nearest neighbor of higher local density $\delta_k(x_T)$. The peak-finding criterion value at x_T is $\gamma_k(x_T)$. When $\lambda_T - \beta\lambda_T$ is less than the minimum density level in the data. The level set graph $G(\lambda_T - \beta\lambda_T)$ contains all the points in $\mathbf{X}$, and hence the algorithm terminates. Consider the illustrative example in Figure 3. Here we see a connected component $A_\beta(x_T)$ containing x_T . As the density level of all points in $A_\beta(x_T)$ is at least $\lambda_T - \beta\lambda_T$, once x_T is assessed the algorithm will terminate. The point x^* with density $f_k(x^*) = \lambda^*$, will only be assessed by DCF if $\gamma_k(x^*) > \gamma_k(x_T)$. As $\lambda^* > \lambda_T$, we require

$$\delta_k(x^*) > \frac{\lambda_T}{\lambda^*} \delta_k(x_T), \quad (1)$$

for x^* to be assessed. As the quantity $\delta_k(x_T)$ is not bounded in general, we cannot guarantee that all modes will be assessed in the DCF procedure.

It should be noted that the scenario described above is unusual in the peak-finding context. Previous work has highlighted the inability of the peak-finding criterion to detect low-density clusters. To understand why this scenario is unlikely in practice, we adapt results from studies of mutual k -NN graphs to show that the probability of there being a path in the connected graph between x_T and any other cluster decreases as the distance $\delta_k(x_T)$ increases [11].

The distance between sets $A, B \subseteq \mathbb{R}^d$ is $d(A, B) = \inf\{\|x - y\| : x \in A, y \in B\}$. We denote the closest connected component to $A_\beta(x_T)$ as $A'_\beta(x_T)$. We assume that there exists $\tilde{\beta} > 0$ such that $d(A_\beta(x_T), A'_\beta(x_T)) \geq u_T > 0$ for some u_T representing the lower bound between the components.

Fig. 3: An illustration of the difficulties posed by the peak-finding criterion. Note that the termination point x_T has potentially unbounded $\delta_k(x_T)$, inhibiting detection of the cluster core containing x^* .

Note that $u_T \leq \delta_k(x_T)$. We require a lower bound on the probability mass of balls of radius u_T around points in $A_{\tilde{\beta}}(x_T)$

$$\rho_T \leq \inf_{x \in A_{\tilde{\beta}}(x_T)} \mu(B(x, u_T)),$$

where $\mu(\cdot)$ is the Lebesgue measure on $\mathbb{R}^d$. As the distance from $A_{\tilde{\beta}}(x_T)$ to all other clusters increases, the value of ρ_T increases. We denote by $r_k(A) = \max_{x \in A} r_k(x)$ the maximal k -NN radius of the points in A . Finally, it is required to denote by $\mathcal{D}_\beta$ the event in which $\|f_k(x) - f(x)\| \leq \tilde{\beta}\lambda_T$ for all sample points in $A_{\tilde{\beta}}(x_T) \cup A'_\beta(x_T)$.

Proposition 2 (Adapted from Prop. 6 of [11]). *Let $\mathcal{I}_\beta$ denote the event that the subgraph of samples in $A_{\tilde{\beta}}(x_T)$ is isolated in $G(\lambda_T - \beta\lambda_T)$. Then, given $\beta < \tilde{\beta}$, $k < \frac{\rho_T n}{2} - 2 \log(\mu(A_{\tilde{\beta}}(x_T))n)$ we obtain*

$$\begin{aligned} \Pr((\mathcal{I}_\beta)^c) &\leq \Pr(r_k(A_{\tilde{\beta}}(x_T)) \geq u_T) + \Pr(\mathcal{D}_\beta) \\ &\leq \exp\left(-\frac{n-1}{2} \left(\frac{\rho_T}{2} - \frac{k-1}{n-1}\right)\right) + \Pr(\mathcal{D}_\beta). \end{aligned}$$

The proof is similar to that given in [11] and is thus omitted for brevity. From this result, we see that the probability of the set being connected in the graph decreases as the quantity ρ_T , and thus u_T , increases. To interpret this result in the context of DCF, consider fixing the pairwise distances of points within $A_\beta(x_T)$, and increasing $\delta_k(x_T)$. We see that as the magnitude of $\delta_k(x_T)$ increases, the magnitude of u_T increases accordingly, and the probability of x_T being connected to the remainder of the graph decreases.

There are several statements regarding the termination level that can be guaranteed.

Proposition 3. *If DCF terminates at x_T with termination density level $\lambda_T - \beta\lambda_T$, $\lambda_T - \beta\lambda_T$ is at least as low as the lowest dip in density between clusters in $\mathbf{X}$.*

Proof. Following the discussion above, if x_T is the termination point, then $\lambda_T - \beta\lambda_T$ is at most the minimum local density of the points in the data. The result follows. $\square$

While obvious, this statement ensures that the termination density level is low when there is a reasonable degree of separation between clusters in the data.

Proposition 4. *If DCF assesses a point x with $f_k(x) = \lambda < \lambda_T$ and x lies in the same connected component of G as x_T , the connected component of $G(\lambda - \beta\lambda)$ containing x will not be accepted as a cluster core.*

Proof. Taking the set of points in the connected component of G containing x and x_T to be $\mathbf{X}'$ and the connected component of $G(\lambda - \beta\lambda)$ to be $A_\beta(x)$. For $A_\beta(x)$ to be accepted as an estimated cluster core, we require $A_\beta(x)$ to be disjoint from all cluster cores in $\widehat{\mathcal{M}}$. But $\lambda < \lambda_T$, hence $G(\lambda_T - \beta\lambda_T) \subseteq G(\lambda - \beta\lambda)$. As x_T is the termination point, $G(\lambda_T - \beta\lambda_T)$ is connected and $\mathbf{X}' \subseteq G(\lambda_T - \beta\lambda_T)$. Hence, $A_\beta(x)$ is not disjoint from any cluster core in $\widehat{\mathcal{M}}$. $\square$

This guarantees that no modes are missed even if the termination density level is high.

B. Simulated Experiments

In order to demonstrate that DCF does not, in practice, ignore modes due to extreme values of $\delta_k(x_T)$ at termination, we compare DCF and QuickShift++ on a range of simulated datasets.

The analysis in the previous section indicates that the performance of DCF depends on the degree of separation of clusters. To assess the impact of separation on our method, we generate data as proposed in [26], [27]. Gaussian components are sampled so that their means satisfy the following inequality:

$$\forall_{i \neq j} : \|m_i - m_j\| \geq c \max_{i,j} \{\text{tr}(\Sigma_i), \text{tr}(\Sigma_j)\},$$

where c controls the degree of separation of the clusters. For each mixture component, we fix the eccentricity (the ratio of the largest to the smallest eigenvalue of the covariance matrix) as one. We generate mixtures of five components with data sizes ranging from $n = 1000$ to $n = 2000$ in increments of 20, data of dimension $d = \{2, 5\}$, and degrees of separation ranging from $c = 0$ (low) to $c = 5$ (high) in increments of 0.05. For each configuration we repeat 10 times. Three of the generated datasets are shown in Figure 4. For both DCF and QuickShift++, we assess $k = \{40, 50\}$ and $\beta = \{0.3, 0.5, 0.7\}$.

In Figure 5, the similarities in performance of DCF and QuickShift++ are shown. In the left plot, we compare the number of cores recovered from the data as the separation parameter c increases. The number of cores returned by DCF tracks the number returned by QuickShift++. In fact, the number of cores returned by DCF is fractionally higher than that of QuickShift++ for a large range of values of the parameter c . This is a direct result of the different approaches to the ordering of data points. This result demonstrates the ability of DCF to recover modes in the data. The quality of the clusterings returned by both methods, measured by the Adjusted Rand Index (ARI), were within 1% for each value of c and, as a result, are not shown.

On the right of Figure 5, we include an analysis of the fraction of points assessed by DCF for three values of $\beta = \{0.3, 0.5, 0.7\}$. Recall that QuickShift++ assesses every instance in $\mathbf{X}$. For each of the three values, we see that the proportion of instances assessed by DCF is less than 2%, an impressive result given the quality of the clusterings returned. This highlights the ability of the peak-finding criterion to determine modes in the data.

C. Complexity Analysis

The most computation-intensive task is creating the mutual k -NN graph which requires $O(nk \log(n))$ operations on average. Another major computational burden is finding, for each point, its nearest neighbor of higher density. For the points which do not have a point of higher density in their neighbors, this requires $O(n)$ operations. In practice, the proportion of instances without a point of higher density in

Fig. 4: Three of the generated datasets used in Section IV-B, with separation values $c = 0.2, 1.5$ and 4.2 respectively.

Fig. 5: Left: The number of cluster cores (#Cores) recovered from each dataset as a function of the separation parameter c for both methods. These results are averaged over the different data sizes, dimensions, repetitions, and parameter values. Right: Plotted is the proportion of instances assessed as potential modes (Prop. Assessed) in DCF for values of $\beta = \{0.3, 0.5, 0.7\}$ as a function of the separation parameter c .

their neighbors is typically less than 1%, as long as k is not too small. As such, for the vast majority of instances, this requires $O(k)$ operations. Assessing each cluster core requires $O(nk)$ operations. The assignment mechanism requires $O(n)$ operations. As such, we see that the complexity of DCF is near linear in n and k .

V. EVALUATION

The performance of DCF on real-world datasets is evaluated using two well known measures of clustering quality and compared to a range of competitor methods. All experiments have been conducted on a PC running Debian 10 (Buster), consisting of 24 cores and 24GB of RAM. Our method is implemented in Python; its source code, and code to reproduce the below experiments is available online.¹

A. Experimental Set-up

To evaluate the clusterings produced by DCF and competitor methods, we adopt two widely used external indices: Adjusted Rand Index (ARI) and Adjusted Mutual Information (AMI), each comparing clusterings to ground truth labels available from the data. For both metrics, a larger value indicates a higher-quality clustering. Due to space limitations, the formal definitions of these metrics are omitted, and can be found in [28].

¹<https://github.com/tobinjo96/DCFcluster> (GitHub Repository)

DCF is assessed on five real-world datasets collected from the UCI Machine Learning Repository [29] and the Phonemes dataset [30]. Details of the datasets can be found in Table II. Instances with missing values are removed.

To assess the performance of DCF on these datasets, we apply the following competitor density-based clustering algorithms:

- QuickShift++ (QSP) [8] implemented in Python and C++.
- Density Peaks Clustering (DPC) [1] implemented in R.
- Adaptive Density Peaks Clustering (ADP) [18] implemented in R.
- Comparative Density Peaks Clustering (CDP) [6] implemented in Matlab.
- DBSCAN (DBS) implemented in Python in the SciKit library [31].
- HDBSCAN (HDB) [17] implemented in Python and C++.

Of these algorithms, DPC, ADP and CDP require the number of clusters to be specified in advance. For all experiments, the true number of clusters is provided as an input to these algorithms. Where possible, all algorithms are executed in parallel on all cores of the machine.

B. Results

Results for clustering six real-world datasets are presented in Table III. For each method, we present the clustering with the highest value of the ARI and AMI. DCF achieves the best clustering, in terms of AMI, for every dataset examined and the best for four of the six datasets, in terms of ARI. QuickShift++ achieves similar performance to DCF, reflecting the ability of cluster cores to model high-density regions in the data. The slight advantage gained by DCF in AMI is likely a result of searching for modes using the peak-finding criterion, rather than in descending order of density. CDP and DPC outperform DCF in ARI on the Dermatology and Letter Recognition datasets respectively. As ARI penalises false positive clusters, the results for CDP and DPC are enhanced by the true number of clusters being provided as an input. The comparatively poor performance of DBSCAN reflects the inability of level sets to capture clusters of different densities in the data. Furthermore, the optimisation algorithm applied in HDBSCAN to extract clusters from the DBSCAN hierarchy fails to improve the clustering results.

The average run time, in seconds, for each of the methods is presented in Table IV. For the small datasets, we see that DCF runs marginally slower than QuickShift++, DBSCAN and HDBSCAN. The magnitude of difference is negligible and unlikely to hinder the use of DCF in applications. We see that for larger datasets, DCF has the fastest, or near fastest, run time. The difference is most pronounced for the Letter Recognition dataset, the dataset with the most points in our experiments. The quadratic complexity of DPC, ADP and CDP results in execution times that impede repeated experimentation. Furthermore, DCF achieves a 33% speed up compared to QuickShift++ for this dataset, as the search is restricted to points with large values of the peak-finding criterion. To support the run time analysis, the proportion of instances

TABLE II: Characteristics of the real-world datasets.

Source	Name	Instances	Dim	Classes
[29]	Dermatology	358	34	6
[29]	Ecoli	336	7	8
[29]	Glass	214	9	6
[29]	Letter Recognition	20000	16	26
[29]	Page Blocks	5743	10	5
[30]	Phonemes	4509	256	5

TABLE III: Quality of clusterings for the real-world datasets.

Dataset	Metric	DCF	QSP	DPC	ADP	CDP	DBS	HDB
Derm.	ARI	0.72	0.70	0.22	0.59	0.73	0.44	0.47
	AMI	0.78	0.78	0.45	0.73	0.75	0.63	0.66
Ecoli	ARI	0.73	0.73	0.55	0.72	0.51	0.50	0.40
	AMI	0.68	0.68	0.50	0.65	0.55	0.48	0.41
Glass	ARI	0.31	0.30	0.20	0.26	0.25	0.25	0.25
	AMI	0.42	0.40	0.27	0.38	0.38	0.38	0.37
Letter R.	ARI	0.20	0.20	0.22	0.10	0.13	0.07	0.02
	AMI	0.59	0.58	0.53	0.33	0.42	0.46	0.45
Page B.	ARI	0.46	0.46	0.39	0.38	0.42	0.32	0.33
	AMI	0.30	0.30	0.27	0.26	0.29	0.18	0.20
Phonemes	ARI	0.76	0.76	0.71	0.70	0.56	0.44	0.36
	AMI	0.83	0.80	0.79	0.75	0.66	0.61	0.57

TABLE IV: Average run time of the assessed clustering methods for the real-world datasets.

Dataset	DCF	QSP	DPC	ADP	CDP	DBS	HDB
Derm.	0.07	0.03	4.65	2.5	0.32	0.01	0.02
Ecoli	0.06	0.02	2.54	1.67	0.13	0.00	0.02
Glass	0.03	0.05	0.61	0.24	0.11	0.00	0.00
Letter R.	12.79	19.21	2430.84	1002.42	372.14	19.94	25.53
Page B.	0.73	1.61	123.27	43.26	14.59	1.23	0.68
Phonemes	7.21	8.79	1627.81	57.33	43.22	15.26	11.42

Fig. 6: The proportion of instances assessed as modes by DCF for each of the six datasets and for all assessed parameter values. Note the log scaling of the y-axis.

visited by DCF is presented in Figure 6. This proportion is almost always less than 1% for Letter Recognition, Page Blocks and Phonemes, the three larger datasets in our analysis.

C. Analysis of the Parameter Space

DCF achieves the best results across the datasets when optimal values are used for the parameters. We now demonstrate that excellent performance is achieved for a wide range of parameter values. DCF has the following parameters: (1) k , the

Fig. 7: For each clustering algorithm, we show the clustering quality as a function of the input parameters. The ARI is shown in black, and the AMI in blue. Note that for DCF, we present the clustering quality as a function of both k and β . For the first column, $\beta = 0.4$ for all datasets. For the second column, k is set to the optimal value for each dataset (in the order of the datasets, $k = \{15, 15, 15, 20, 85, 115\}$).

number of neighbors when building the k -NN graph and (2) β , the amount of variation in density allowed within a cluster core. The first two columns of Figure 7 show how the ARI and AMI varies with changes in k and β . The remaining columns show changes in the ARI and AMI with respect to changes in the primary parameter according to guidance provided by the authors.

We see immediately that the performance of DCF is robust to choices of k and β . The results remain consistent as k is increased. For all datasets except Letter Recognition, choosing $k \approx \sqrt{n}$ returns high quality results. The quality of the clusterings remain also remarkably consistent as the variation permitted within the cluster core increases from $\beta = 0.1$ to $\beta = 0.9$. As such, it is recommended to proceed with $\beta = 0.4$ for general applications. Analysis of QuickShift++ reveals similar results, highlighting the robustness of clustering results when high-density regions are represented with cluster cores. The methods that apply the peak-finding criterion, particularly ADP and CDP, also display highly consistent results as the values of the bandwidth (h) and threshold distance (d_c) increase respectively. These results indicate that the peak-finding criterion is a reliable measure for mode detection.

VI. FACE RECOGNITION

Face recognition has become a central problem in deep learning in recent years [32]. However, the majority of work

relies on large labelled datasets for use in training. As the number and volume of datasets increases, effective unsupervised face recognition methods will be required. DCF is applied to two large image datasets to demonstrate its ability to perform unsupervised face recognition. We use a sample of the MS-Celeb-1M data [33] consisting of 17,146 identities, each with roughly 100 images. The YouTube Face dataset (YTB-Faces) [34] is another benchmark image dataset. We use the sample of this dataset in [35] of 155,282 frames with 1,595 identities. For both datasets, we apply DCF to numerical features extracted using a trained CNN.² Sample images from the datasets are presented in Figure 8 and details of the datasets can be found in Table V.

To assess the performance of DCF on these datasets, we apply the following competitor algorithms:

- QuickShift++ (QSP) [8] implemented in Python and C++.
- OPTICS (OPT) implemented in Python in the SciKit library [31].

These algorithms were chosen as comparators as they are scalable to large datasets and do not require the number of clusters to be provided as an input. As the number of clusters is not likely to be provided for such an application, scalable adaptations of DPC (e.g. [36]) are not suitable for comparison.

²<https://github.com/y1-1993/learn-to-cluster>

(a) MS-Celeb-1M

(b) YTB-Faces

Fig. 8: Three samples from two clusters present in each of the face recognition datasets.

TABLE V: Characteristics of the face recognition datasets.

Source	Name	Instances	Dim	Classes
[33]	MS-Celeb-1M	1,160,507	256	17,146
[34]	YTB-Faces	155,282	256	1,595

TABLE VI: Quality of clusterings for the face recognition datasets.

Dataset	Metric	DCF	QSP	OPT
MS-Celeb-1M	ARI	0.90	0.83	-
	AMI	0.96	0.92	-
YTB-Faces	ARI	0.69	0.52	0.06
	AMI	0.91	0.88	0.15

TABLE VII: Average run time of the assessed clustering methods for face recognition image datasets.

Dataset	DCF	QSP	OPT
MS-Celeb-1M	13202.00	39212.14	-
YTB-Faces	2212.59	4338.95	29631.24

A. Results

The results of the experiments for both datasets are given in Table VI. DCF achieves exceptional results for both datasets. MS-Celeb-1M and YTB-Faces are challenging datasets due to the large size and extremely large number of clusters. DCF achieves the highest quality clustering for both ARI and AMI. The impressive results of both DCF and QuickShift++ indicate that cluster cores are well suited to the problem

Fig. 9: Analysis of effects of CPF parameters k and β for face detection datasets. The ARI is shown in black and the AMI in blue.

of face detection. DCF shows slightly better performance. As clusters in face recognition data tend to be relatively well separated, the peak-finding criterion quickly traverses disconnected components in the mutual k -NN graph.

We report the average run time, in seconds, for each of the methods in Table VII. The peak-finding criterion leads to dramatically reduced run time for DCF compared to QuickShift++. For MS-Celeb-1M, DCF is approximately 67% faster than QuickShift++ and approximately 49% faster for YTB-Faces. OPTICS did not complete within 24 hours for MS-Celeb-1M, and hence was not included.

We repeat the analysis of the parameter space completed in Section V for the face recognition datasets. Again we see that DCF is robust for a large range of values for k and β . As the size of the dataset increases, we observe that the optimal value for k is approximately $\log(n)$. Choosing $k \approx \log n$ instead of $\sqrt{n}$, has the added benefit of reducing the computation required to construct the k -NN graph.

VII. CONCLUSION

This article introduced DCF, an improved algorithm for clustering using modal sets and density peaks. We showed that modelling high-density regions of the data using cluster cores achieves better results than using modes, and that the peak finding criterion effectively locates cluster cores in the data. Theoretical guarantees on the performance of DCF were provided, supported by results from simulation studies. Extensive experimental results further demonstrated that DCF has excellent performance. It achieves superior results over a broad range of benchmark datasets and indices when compared to popular competitor methods. DCF is both efficient and robust, gracefully scaling to big datasets and achieving exceptional performance over a broad range of parameter values. Finally, we showed the ability of DCF to perform unsupervised face recognition, a challenging contemporary clustering problem.

REFERENCES

- [1] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.
- [2] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, Aug. 1995.
- [3] D. Comaniciu and M. Peter, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 281–288, Jan. 2002.
- [4] A. Vedaldi and S. Soatto, "Quick Shift and Kernel Methods for Mode Seeking," in *Computer Vision – ECCV 2008*, ser. Lecture Notes in Computer Science, D. Forsyth, P. Torr, and A. Zisserman, Eds. Berlin, Heidelberg: Springer, 2008, pp. 705–718.
- [5] B. Chen, K. M. Ting, T. Washio, and Y. Zhu, "Local contrast as an effective means to robust clustering against varying densities," *Machine Learning*, vol. 107, no. 8, pp. 1621–1645, Sep. 2018.
- [6] Z. Li and Y. Tang, "Comparative density peaks clustering," *Expert Systems with Applications*, vol. 95, pp. 236–247, Apr. 2018.
- [7] H. Jiang and S. Kpotufe, "Modal-set estimation with an application to clustering," in *Artificial Intelligence and Statistics*. PMLR, Apr. 2017, pp. 1197–1206.
- [8] H. Jiang, J. Jang, and S. Kpotufe, "Quickshift++: Provably Good Initializations for Sample-Based Mean Shift," in *International Conference on Machine Learning*. PMLR, Jul. 2018, pp. 2294–2303.
- [9] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96. Portland, Oregon: AAAI Press, Aug. 1996, pp. 226–231.
- [10] J. A. Hartigan, *Clustering Algorithms*. Wiley, 1975.
- [11] M. Maier, M. Hein, and U. von Luxburg, "Optimal construction of k-nearest-neighbor graphs for identifying noisy clusters," *Theoretical Computer Science*, vol. 410, no. 19, pp. 1749–1764, Apr. 2009.
- [12] I. Steinwart, "Adaptive Density Level Set Clustering," in *Proceedings of the 24th Annual Conference on Learning Theory*. JMLR Workshop and Conference Proceedings, Dec. 2011, pp. 703–738.
- [13] B. N. Li, C. K. Chui, S. Chang, and S. H. Ong, "Integrating spatial fuzzy clustering with level set methods for automated medical image segmentation," *Computers in Biology and Medicine*, vol. 41, no. 1, pp. 1–10, Jan. 2011.
- [14] S. Kpotufe and U. von Luxburg, "Pruning nearest neighbor cluster trees," *arXiv:1105.0540 [cs, stat]*, May 2011.
- [15] S. Balakrishnan, S. Narayanan, A. Rinaldo, A. Singh, and L. Wasserman, "Cluster Trees on Manifolds," *Advances in Neural Information Processing Systems*, vol. 26, 2013.
- [16] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," *ACM SIGMOD Record*, vol. 28, no. 2, pp. 49–60, Jun. 1999.
- [17] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-Based Clustering Based on Hierarchical Density Estimates," in *Advances in Knowledge Discovery and Data Mining*, ser. Lecture Notes in Computer Science, J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, Eds. Berlin, Heidelberg: Springer, 2013, pp. 160–172.
- [18] X.-F. Wang and Y. Xu, "Fast clustering using adaptive density peak detection," *Statistical Methods in Medical Research*, vol. 26, no. 6, pp. 2800–2811, Dec. 2017.
- [19] M. Du, S. Ding, and H. Jia, "Study on density peaks clustering based on k-nearest neighbors and principal component analysis," *Knowledge-Based Systems*, vol. 99, pp. 135–145, 2016.
- [20] J. Xie, H. Gao, W. Xie, X. Liu, and P. Grant, "Robust clustering by detecting density peaks and assigning points based on fuzzy weighted K-nearest neighbors," *Information Sciences*, vol. 354, pp. 19–40, 2016.
- [21] J. Hou and M. Pelillo, "A new density kernel in density peak based clustering," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, Dec. 2016, pp. 468–473.
- [22] J. Ding, X. He, J. Yuan, and B. Jiang, "Automatic clustering based on density peak detection using generalized extreme value distribution," *Soft Computing*, vol. 22, no. 9, pp. 2777–2796, May 2018.
- [23] Y. Wang, D. Wang, X. Zhang, W. Pang, C. Miao, A.-H. Tan, and Y. Zhou, "McDPC: Multi-center density peak clustering," *Neural Computing and Applications*, 2020.
- [24] S. Dasgupta and S. Kpotufe, "Optimal rates for k-NN density and mode estimation," *Advances in Neural Information Processing Systems*, vol. 27, 2014.
- [25] K. Chaudhuri, S. Dasgupta, S. Kpotufe, and U. von Luxburg, "Consistent Procedures for Cluster Tree Estimation and Pruning," *IEEE Transactions on Information Theory*, vol. 60, no. 12, pp. 7900–7912, Dec. 2014.
- [26] S. Dasgupta, "Learning Mixtures of Gaussians," in *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, ser. FOCS '99. USA: IEEE Computer Society, Oct. 1999, p. 634.
- [27] J. J. Verbeek, N. Vlassis, and B. Kröse, "Efficient greedy learning of Gaussian mixture models," *Neural Computation*, vol. 15, no. 2, pp. 469–485, Feb. 2003.
- [28] H. Xiong, J. Wu, and J. Chen, "K-Means Clustering Versus Validation Measures: A Data-Distribution Perspective," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 318–331, 2009.
- [29] D. Dua and C. Graff, "UCI Machine Learning Repository," University of California, Irvine, School of Information and Computer Sciences, 2019.
- [30] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Science & Business Media, Aug. 2009.
- [31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.
- [32] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "CosFace: Large Margin Cosine Loss for Deep Face Recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 5265–5274.
- [33] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, "MS-Celeb-1M: A Dataset and Benchmark for Large-Scale Face Recognition," *arXiv:1607.08221 [cs]*, Jul. 2016.
- [34] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," in *2011 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2011, pp. 529–534.
- [35] L. Yang, X. Zhan, D. Chen, J. Yan, C. C. Loy, and D. Lin, "Learning to Cluster Faces on an Affinity Graph," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 2293–2301.
- [36] S. Sieranoja and P. Fränti, "Fast and general density peaks clustering," *Pattern Recognition Letters*, vol. 128, pp. 551–558, Dec. 2019.