CHAPTER 15 Density-based Clustering

The representative-based clustering methods like K-means and expectation-maximization are suitable for finding ellipsoid-shaped clusters, or at best convex clusters. However, for nonconvex clusters, such as those shown in Figure 15.1, these methods have trouble finding the true clusters, as two points from different clusters may be closer than two points in the same cluster. The density-based methods we consider in this chapter are able to mine such nonconvex clusters.

15.1 THE DBSCAN ALGORITHM

Density-based clustering uses the local density of points to determine the clusters, rather than using only the distance between points. We define a ball of radius ϵ around a point $\mathbf{x} \in \mathbb{R}^d$, called the *$\epsilon$-neighborhood* of $\mathbf{x}$, as follows:

$$N_\epsilon(\mathbf{x}) = B_d(\mathbf{x}, \epsilon) = \{\mathbf{y} \mid \delta(\mathbf{x}, \mathbf{y}) \le \epsilon\}$$

Here $\delta(\mathbf{x}, \mathbf{y})$ represents the distance between points $\mathbf{x}$ and $\mathbf{y}$, which is usually assumed to be the Euclidean distance, that is, $\delta(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2$. However, other distance metrics can also be used.

For any point $\mathbf{x} \in \mathbf{D}$, we say that $\mathbf{x}$ is a *core point* if there are at least $minpts$ points in its ϵ-neighborhood. In other words, $\mathbf{x}$ is a core point if $|N_\epsilon(\mathbf{x})| \ge minpts$, where $minpts$ is a user-defined local density or frequency threshold. A *border point* is defined as a point that does not meet the $minpts$ threshold, that is, it has $|N_\epsilon(\mathbf{x})| < minpts$, but it belongs to the ϵ-neighborhood of some core point $\mathbf{z}$, that is, $\mathbf{x} \in N_\epsilon(\mathbf{z})$. Finally, if a point is neither a core nor a border point, then it is called a *noise point* or an outlier.

Example 15.1. Figure 15.2a shows the ϵ-neighborhood of the point $\mathbf{x}$, using the Euclidean distance metric. Figure 15.2b shows the three different types of points, using $minpts = 6$. Here $\mathbf{x}$ is a core point because $|N_\epsilon(\mathbf{x})| = 6$, $\mathbf{y}$ is a border point because $|N_\epsilon(\mathbf{y})| < minpts$, but it belongs to the ϵ-neighborhood of the core point $\mathbf{x}$, i.e., $\mathbf{y} \in N_\epsilon(\mathbf{x})$. Finally, $\mathbf{z}$ is a noise point.

We say that a point $\mathbf{x}$ is *directly density reachable* from another point $\mathbf{y}$ if $\mathbf{x} \in N_\epsilon(\mathbf{y})$ and $\mathbf{y}$ is a core point. We say that $\mathbf{x}$ is *density reachable* from $\mathbf{y}$ if there exists a chain

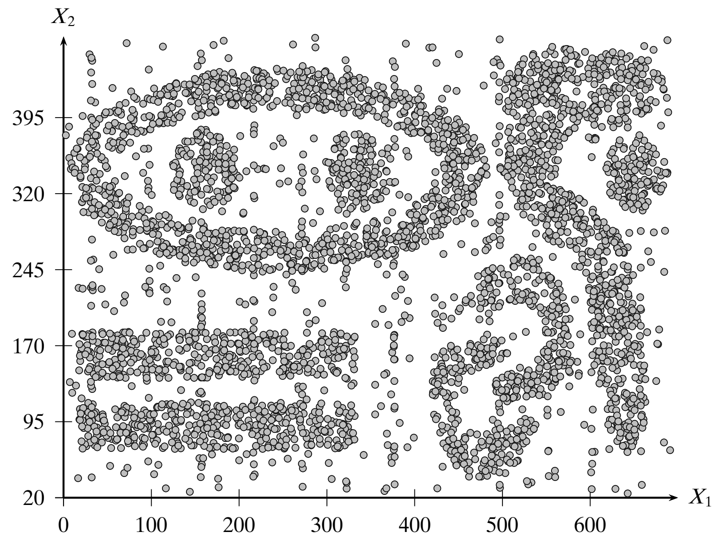


Figure 15.1. Density-based dataset.

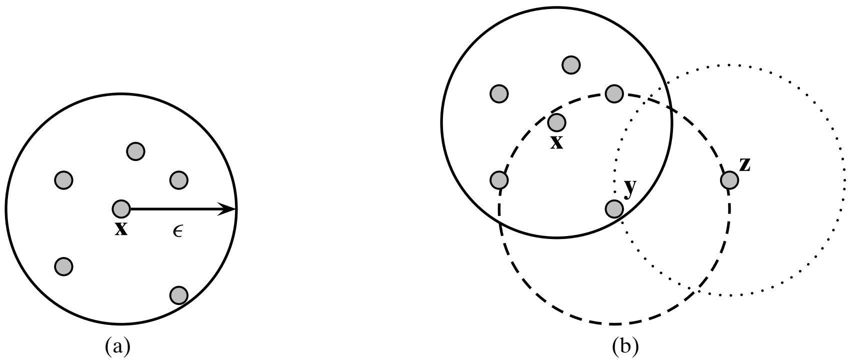


Figure 15.2. (a) Neighborhood of a point. (b) Core, border, and noise points.

of points, $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_l$, such that $\mathbf{x} = \mathbf{x}_0$ and $\mathbf{y} = \mathbf{x}_l$, and $\mathbf{x}_i$ is directly density reachable from $\mathbf{x}_{i-1}$ for all $i = 1, \dots, l$. In other words, there is set of core points leading from $\mathbf{y}$ to $\mathbf{x}$. Note that density reachability is an asymmetric or directed relationship. Define any two points $\mathbf{x}$ and $\mathbf{y}$ to be *density connected* if there exists a core point $\mathbf{z}$, such that both $\mathbf{x}$ and $\mathbf{y}$ are density reachable from $\mathbf{z}$. A *density-based cluster* is defined as a maximal set of density connected points.

The pseudo-code for the DBSCAN density-based clustering method is shown in Algorithm 15.1. First, DBSCAN computes the ϵ-neighborhood $N_\epsilon(\mathbf{x}_i)$ for each point $\mathbf{x}_i$ in the dataset $\mathbf{D}$, and checks if it is a core point (lines 2–5). It also sets the cluster id $id(\mathbf{x}_i) = \emptyset$ for all points, indicating that they are not assigned to any cluster. Next, starting from each unassigned core point, the method recursively finds all its density connected points, which are assigned to the same cluster (line 10). Some border point

ALGORITHM 15.1. Density-based Clustering Algorithm

```
DBSCAN (D, ϵ, minpts):
1  Core ← ∅
2  foreach x_i ∈ D do // Find the core points
3      Compute N_ϵ(x_i)
4      id(x_i) ← ∅ // cluster id for x_i
5      if N_ϵ(x_i) ≥ minpts then Core ← Core ∪ {x_i}
6  k ← 0 // cluster id
7  foreach x_i ∈ Core, such that id(x_i) = ∅ do
8      k ← k + 1
9      id(x_i) ← k // assign x_i to cluster id k
10     DensityConnected (x_i, k)
11 C ← {C_i}_{i=1}^k, where C_i ← {x ∈ D | id(x) = i}
12 Noise ← {x ∈ D | id(x) = ∅}
13 Border ← D \ {Core ∪ Noise}
14 return C, Core, Border, Noise

DensityConnected (x, k):
15 foreach y ∈ N_ϵ(x) do
16     id(y) ← k // assign y to cluster id k
17     if y ∈ Core then DensityConnected (y, k)
```

may be reachable from core points in more than one cluster; they may either be arbitrarily assigned to one of the clusters or to all of them (if overlapping clusters are allowed). Those points that do not belong to any cluster are treated as outliers or noise.

DBSCAN can also be considered as a search for the connected components in a graph where the vertices correspond to the core points in the dataset, and there exists an (undirected) edge between two vertices (core points) if the distance between them is less than ϵ, that is, each of them is in the ϵ-neighborhood of the other point. The connected components of this graph correspond to the core points of each cluster. Next, each core point incorporates into its cluster any border points in its neighborhood.

One limitation of DBSCAN is that it is sensitive to the choice of ϵ, in particular if clusters have different densities. If ϵ is too small, sparser clusters will be categorized as noise. If ϵ is too large, denser clusters may be merged together. In other words, if there are clusters with different local densities, then a single ϵ value may not suffice.

Example 15.2. Figure 15.3 shows the clusters discovered by DBSCAN on the density-based dataset in Figure 15.1. For the parameter values $\epsilon = 15$ and $minpts = 10$, found after parameter tuning, DBSCAN yields a near-perfect clustering comprising all nine clusters. Cluster are shown using different symbols and shading; noise points are shown as plus symbols.

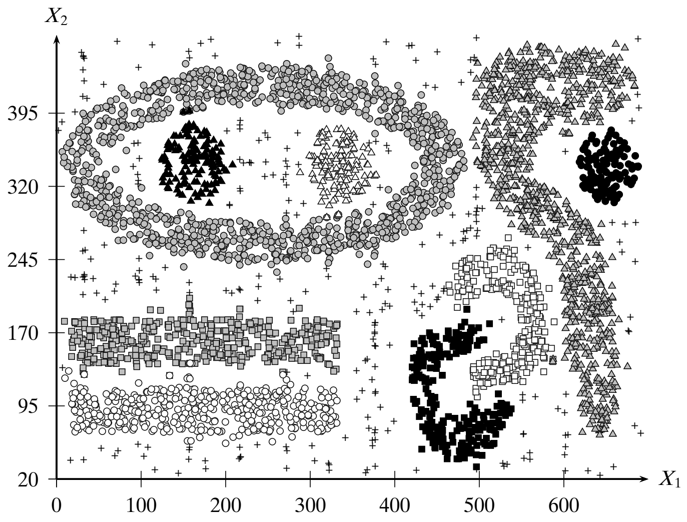


Figure 15.3. Density-based clusters.

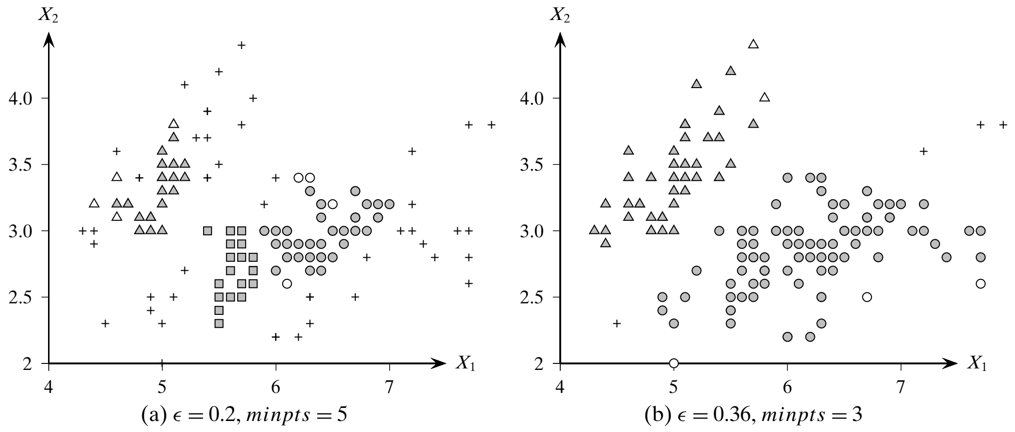


Figure 15.4. DBSCAN clustering: Iris dataset.

Example 15.3. Figure 15.4 shows the clusterings obtained via DBSCAN on the two-dimensional Iris dataset (over `sepal length` and `sepal width` attributes) for two different parameter settings. Figure 15.4a shows the clusters obtained with radius $\epsilon = 0.2$ and core threshold $minpts = 5$. The three clusters are plotted using different shaped points, namely circles, squares, and triangles. Shaded points are core points, whereas the border points for each cluster are showed unshaded (white). Noise points are shown as plus symbols. Figure 15.4b shows the clusters obtained with a larger value of radius $\epsilon = 0.36$, with $minpts = 3$. Two clusters are found, corresponding to the two dense regions of points.

For this dataset tuning the parameters is not that easy, and DBSCAN is not very effective in discovering the three Iris classes. For instance it identifies too many points (47 of them) as noise in Figure 15.4a. However, DBSCAN is able to find the two main dense sets of points, distinguishing `iris-setosa` (in triangles) from the other types of Irises, in Figure 15.4b. Increasing the radius more than $\epsilon = 0.36$ collapses all points into a single large cluster.

Computational Complexity

The main cost in DBSCAN is for computing the ϵ-neighborhood for each point. If the dimensionality is not too high this can be done efficiently using a spatial index structure in $O(n \log n)$ time. When dimensionality is high, it takes $O(n^2)$ to compute the neighborhood for each point. Once $N_\epsilon(\mathbf{x})$ has been computed the algorithm needs only a single pass over all the points to find the density connected clusters. Thus, the overall complexity of DBSCAN is $O(n^2)$ in the worst-case.

15.2 KERNEL DENSITY ESTIMATION

There is a close connection between density-based clustering and density estimation. The goal of density estimation is to determine the unknown probability density function by finding the dense regions of points, which can in turn be used for clustering. Kernel density estimation is a nonparametric technique that does not assume any fixed probability model of the clusters, as in the case of K-means or the mixture model assumed in the EM algorithm. Instead, it tries to directly infer the underlying probability density at each point in the dataset.

15.2.1 Univariate Density Estimation

Assume that X is a continuous random variable, and let $x_1, x_2, \ldots, x_n$ be a random sample drawn from the underlying probability density function $f(x)$, which is assumed to be unknown. We can directly estimate the cumulative distribution function from the data by counting how many points are less than or equal to x:

$$\hat{F}(x) = \frac{1}{n}\sum_{i=1}^{n} I(x_i \le x)$$

where I is an indicator function that has value 1 only when its argument is true, and 0 otherwise. We can estimate the density function by taking the derivative of $\hat{F}(x)$, by considering a window of small width h centered at x, that is,

$$\hat{f}(x) = \frac{\hat{F}\left(x + \frac{h}{2}\right) - \hat{F}\left(x - \frac{h}{2}\right)}{h} = \frac{k/n}{h} = \frac{k}{nh} \tag{15.1}$$

where k is the number of points that lie in the window of width h centered at x, that is, within the closed interval $[x - \frac{h}{2},\ x + \frac{h}{2}]$. Thus, the density estimate is the ratio of the fraction of the points in the window (k/n) to the volume of the window (h). Here h plays the role of “influence.” That is, a large h estimates the probability density over a large window by considering many points, which has the effect of smoothing the estimate. On the other hand, if h is small, then only the points in close proximity to x are considered. In general we want a small value of h, but not too small, as in that case no points will fall in the window and we will not be able to get an accurate estimate of the probability density.

Kernel Estimator

Kernel density estimation relies on a *kernel function* K that is non-negative, symmetric, and integrates to 1, that is, $K(x) \geq 0$, $K(-x) = K(x)$ for all values x, and $\int K(x)dx = 1$. Thus, K is essentially a probability density function. Note that K should not be confused with the positive semidefinite kernel mentioned in Chapter 5.

Discrete Kernel The density estimate $\hat{f}(x)$ from Eq. (15.1) can also be rewritten in terms of the kernel function as follows:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

where the **discrete kernel** function K computes the number of points in a window of width h, and is defined as

$$K(z) = \begin{cases} 1 & \text{If } |z| \leq \frac{1}{2} \\ 0 & \text{Otherwise} \end{cases} \tag{15.2}$$

We can see that if $|z| = |\frac{x-x_i}{h}| \leq \frac{1}{2}$, then the point x_i is within a window of width h centered at x, as

$$\begin{aligned} \left|\frac{x - x_i}{h}\right| \leq \frac{1}{2} \text{ implies that} & -\frac{1}{2} \leq \frac{x_i - x}{h} \leq \frac{1}{2}, \text{ or} \\ & -\frac{h}{2} \leq x_i - x \leq \frac{h}{2}, \text{ and finally} \\ & x - \frac{h}{2} \leq x_i \leq x + \frac{h}{2} \end{aligned}$$

Example 15.4. Figure 15.5 shows the kernel density estimates using the discrete kernel for different values of the influence parameter h, for the one-dimensional Iris dataset comprising the `sepal length` attribute. The x-axis plots the $n = 150$ data points. Because several points have the same value, they are shown stacked, where the stack height corresponds to the frequency of that value.

When h is small, as shown in Figure 15.5a, the density function has many local maxima or modes. However, as we increase h from 0.25 to 2, the number of modes decreases, until h becomes large enough to yield a unimodal distribution, as shown in Figure 15.5d. We can observe that the discrete kernel yields a non-smooth (or jagged) density function.

Gaussian Kernel The width h is a parameter that denotes the spread or smoothness of the density estimate. If the spread is too large we get a more averaged value. If it is too small we do not have enough points in the window. Further, the kernel function in Eq. (15.2) has an abrupt influence. For points within the window ($|z| \leq \frac{1}{2}$) there is a net contribution of $\frac{1}{hn}$ to the probability estimate $\hat{f}(x)$. On the other hand, points outside the window ($|z| > \frac{1}{2}$) contribute 0.

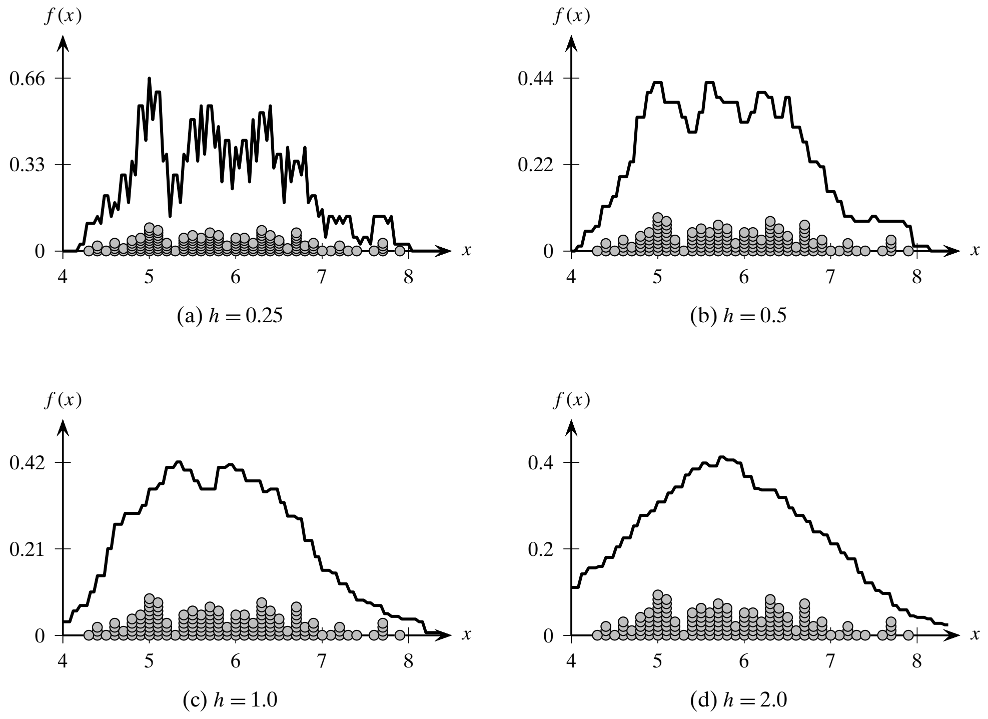


Figure 15.5. Kernel density estimation: discrete kernel (varying h).

Instead of the discrete kernel, we can define a more smooth transition of influence via a Gaussian kernel:

$$K(z) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{z^2}{2}\right\}$$

Thus, we have

$$K\left(\frac{x - x_i}{h}\right) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{(x - x_i)^2}{2h^2}\right\}$$

Here x, which is at the center of the window, plays the role of the mean, and h acts as the standard deviation.

Example 15.5. Figure 15.6 shows the univariate density function for the 1-dimensional Iris dataset (over `sepal length`) using the Gaussian kernel. Plots are shown for increasing values of the spread parameter h. The data points are shown stacked along the x-axis, with the heights corresponding to the value frequencies.

As h varies from 0.1 to 0.5, we can see the smoothing effect of increasing h on the density function. For instance, for $h = 0.1$ there are many local maxima, whereas for $h = 0.5$ there is only one density peak. Compared to the discrete kernel case shown in Figure 15.5, we can clearly see that the Gaussian kernel yields much smoother estimates, without discontinuities.

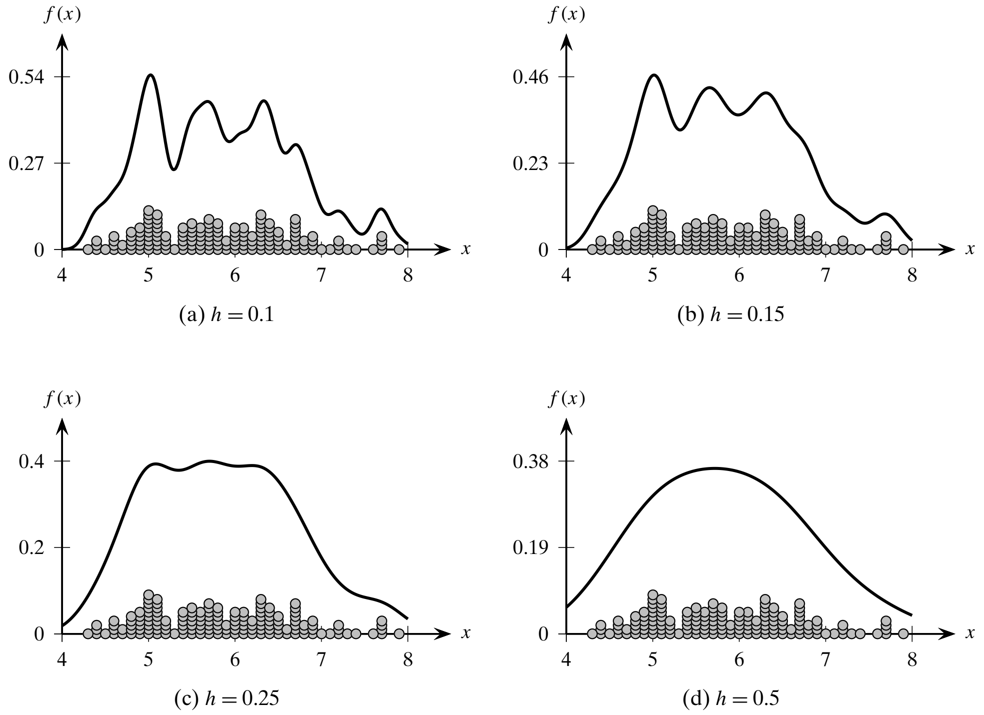


Figure 15.6. Kernel density estimation: Gaussian kernel (varying h).

15.2.2 Multivariate Density Estimation

To estimate the probability density at a d-dimensional point $\mathbf{x} = (x_1, x_2, \ldots, x_d)^T$, we define the d-dimensional "window" as a hypercube in d dimensions, that is, a hypercube centered at $\mathbf{x}$ with edge length h. The volume of such a d-dimensional hypercube is given as

$$\text{vol}(H_d(h)) = h^d$$

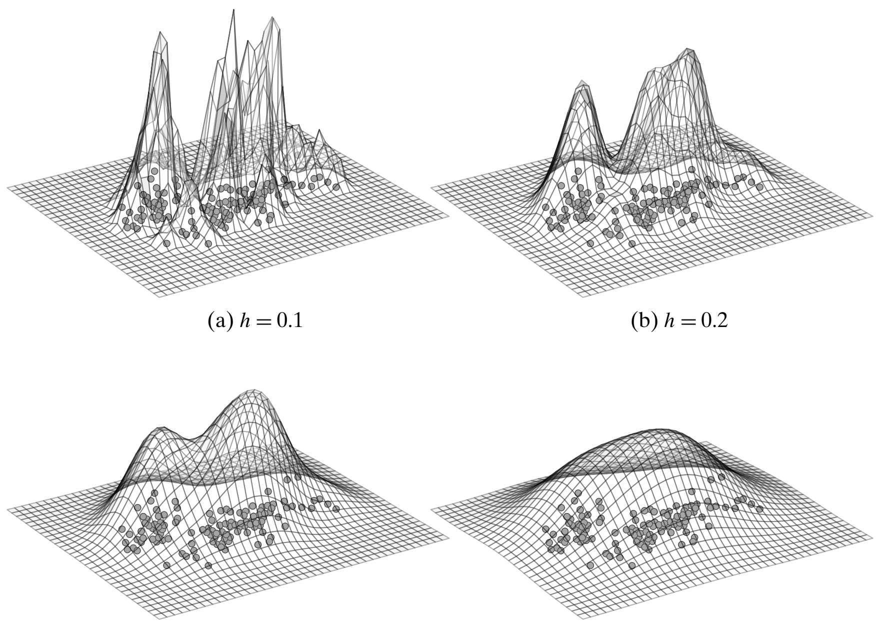
The density is then estimated as the fraction of the point weight lying within the d-dimensional window centered at $\mathbf{x}$, divided by the volume of the hypercube:

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \tag{15.3}$$

where the multivariate kernel function K satisfies the condition $\int K(\mathbf{z})d\mathbf{z} = 1$.

Discrete Kernel For any d-dimensional vector $\mathbf{z} = (z_1, z_2, \ldots, z_d)^T$, the discrete kernel function in d-dimensions is given as

$$K(\mathbf{z}) = \begin{cases} 1 & \text{If } |z_j| \le \frac{1}{2}, \text{for all dimensions } j = 1, \ldots, d \\ 0 & \text{Otherwise} \end{cases}$$



(a) $h = 0.1$ (b) $h = 0.2$

(c) $h = 0.35$ (d) $h = 0.6$

Figure 15.7. Density estimation: 2D Iris dataset (varying h).

For $\mathbf{z} = \frac{\mathbf{x}-\mathbf{x}_i}{h}$, we see that the kernel computes the number of points within the hypercube centered at $\mathbf{x}$ because $K(\frac{\mathbf{x}-\mathbf{x}_i}{h}) = 1$ if and only if $|\frac{x_j - x_{ij}}{h}| \le \frac{1}{2}$ for all dimensions j. Each point within the hypercube thus contributes a weight of $\frac{1}{n}$ to the density estimate.

Gaussian Kernel The d-dimensional Gaussian kernel is given as

$$K(\mathbf{z}) = \frac{1}{(2\pi)^{d/2}} \exp\left\{-\frac{\mathbf{z}^T\mathbf{z}}{2}\right\} \tag{15.4}$$

where we assume that the covariance matrix is the $d \times d$ identity matrix, that is, $\boldsymbol{\Sigma} = \mathbf{I}_d$. Plugging $\mathbf{z} = \frac{\mathbf{x}-\mathbf{x}_i}{h}$ in Eq. (15.4), we have

$$K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) = \frac{1}{(2\pi)^{d/2}} \exp\left\{-\frac{(\mathbf{x}-\mathbf{x}_i)^T(\mathbf{x}-\mathbf{x}_i)}{2h^2}\right\}$$

Each point contributes a weight to the density estimate inversely proportional to its distance from $\mathbf{x}$ tempered by the width parameter h.

Example 15.6. Figure 15.7 shows the probability density function for the 2D Iris dataset comprising the `sepal length` and `sepal width` attributes, using the Gaussian kernel. As expected, for small values of h the density function has several local maxima, whereas for larger values the number of maxima reduce, and ultimately for a large enough value we obtain a unimodal distribution.

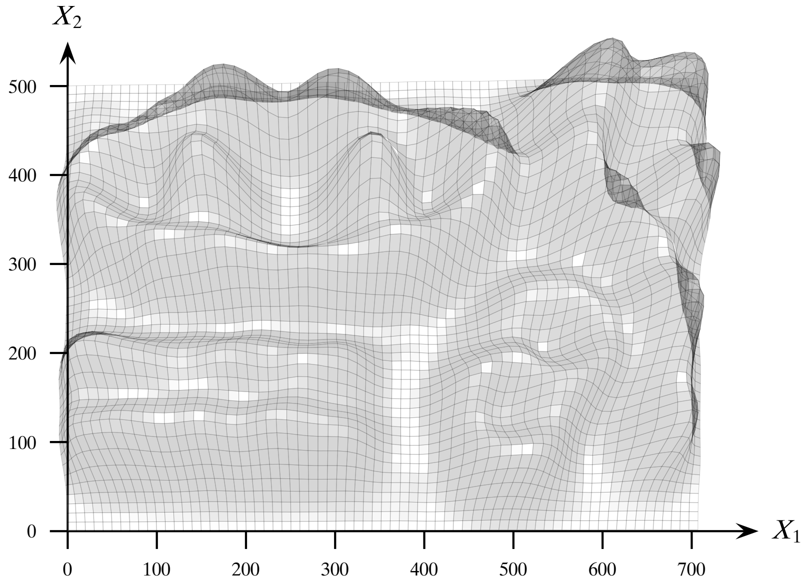


Figure 15.8. Density estimation: density-based dataset.

Example 15.7. Figure 15.8 shows the kernel density estimate for the density-based dataset in Figure 15.1, using a Gaussian kernel with $h = 20$. One can clearly discern that the density peaks closely correspond to regions with higher density of points.

15.2.3 Nearest Neighbor Density Estimation

In the preceding density estimation formulation we implicitly fixed the volume by fixing the width h, and we used the kernel function to find out the number or weight of points that lie inside the fixed volume region. An alternative approach to density estimation is to fix k, the number of points required to estimate the density, and allow the volume of the enclosing region to vary to accommodate those k points. This approach is called the k nearest neighbors (KNN) approach to density estimation. Like kernel density estimation, KNN density estimation is also a nonparametric approach.

Given k, the number of neighbors, we estimate the density at $\mathbf{x}$ as follows:

$$\hat{f}(\mathbf{x}) = \frac{k}{n\,\mathrm{vol}(S_d(h_{\mathbf{x}}))}$$

where $h_{\mathbf{x}}$ is the distance from $\mathbf{x}$ to its kth nearest neighbor, and $\mathrm{vol}(S_d(h_{\mathbf{x}}))$ is the volume of the d-dimensional hypersphere $S_d(h_{\mathbf{x}})$ centered at $\mathbf{x}$, with radius $h_{\mathbf{x}}$ [Eq. (6.4)]. In other words, the width (or radius) $h_{\mathbf{x}}$ is now a variable, which depends on $\mathbf{x}$ and the chosen value k.

15.3 DENSITY-BASED CLUSTERING: DENCLUE

Having laid the foundations of kernel density estimation, we can develop a general formulation of density-based clustering. The basic approach is to find the peaks in the density landscape via gradient-based optimization, and find the regions with density above a given threshold.

Density Attractors and Gradient

A point $\mathbf{x}^*$ is called a *density attractor* if it is a local maxima of the probability density function f. A density attractor can be found via a gradient ascent approach starting at some point $\mathbf{x}$. The idea is to compute the density gradient, the direction of the largest increase in the density, and to move in the direction of the gradient in small steps, until we reach a local maxima.

The gradient at a point $\mathbf{x}$ can be computed as the multivariate derivative of the probability density estimate in Eq. (15.3), given as

$$\nabla \hat{f}(\mathbf{x}) = \frac{\partial}{\partial \mathbf{x}} \hat{f}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} \frac{\partial}{\partial \mathbf{x}} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) \tag{15.5}$$

For the Gaussian kernel [Eq. (15.4)], we have

$$\begin{aligned}\frac{\partial}{\partial \mathbf{x}} K(\mathbf{z}) &= \left(\frac{1}{(2\pi)^{d/2}} \exp\left\{-\frac{\mathbf{z}^T\mathbf{z}}{2}\right\}\right) \cdot -\mathbf{z} \cdot \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \\ &= K(\mathbf{z}) \cdot -\mathbf{z} \cdot \frac{\partial \mathbf{z}}{\partial \mathbf{x}}\end{aligned}$$

Setting $\mathbf{z} = \frac{\mathbf{x}-\mathbf{x}_i}{h}$ above, we get

$$\frac{\partial}{\partial \mathbf{x}} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) = K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) \cdot \left(\frac{\mathbf{x}_i-\mathbf{x}}{h}\right) \cdot \left(\frac{1}{h}\right)$$

which follows from the fact that $\frac{\partial}{\partial \mathbf{x}}\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) = \frac{1}{h}$. Substituting the above in Eq. (15.5), the gradient at a point $\mathbf{x}$ is given as

$$\nabla \hat{f}(\mathbf{x}) = \frac{1}{nh^{d+2}} \sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) \cdot (\mathbf{x}_i - \mathbf{x}) \tag{15.6}$$

This equation can be thought of as having two parts. A vector $(\mathbf{x}_i - \mathbf{x})$ and a scalar *influence* value $K(\frac{\mathbf{x}-\mathbf{x}_i}{h})$. For each point $\mathbf{x}_i$, we first compute the direction away from $\mathbf{x}$, that is, the vector $(\mathbf{x}_i - \mathbf{x})$. Next, we scale it using the Gaussian kernel value as the weight $K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)$. Finally, the vector $\nabla \hat{f}(\mathbf{x})$ is the net influence at $\mathbf{x}$, as illustrated in Figure 15.9, that is, the weighted sum of the difference vectors.

We say that $\mathbf{x}^*$ is a *density attractor* for $\mathbf{x}$, or alternatively that $\mathbf{x}$ is *density attracted* to $\mathbf{x}^*$, if a hill climbing process started at $\mathbf{x}$ converges to $\mathbf{x}^*$. That is, there exists a sequence of points $\mathbf{x} = \mathbf{x}_0 \rightarrow \mathbf{x}_1 \rightarrow \ldots \rightarrow \mathbf{x}_m$, starting from $\mathbf{x}$ and ending at $\mathbf{x}_m$, such that $\|\mathbf{x}_m - \mathbf{x}^*\| \leq \epsilon$, that is, $\mathbf{x}_m$ converges to the attractor $\mathbf{x}^*$.

The typical approach is to use the gradient-ascent method to compute $\mathbf{x}^*$, that is, starting from $\mathbf{x}$, we iteratively update it at each step t via the update rule:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \delta \cdot \nabla \hat{f}(\mathbf{x}_t)$$

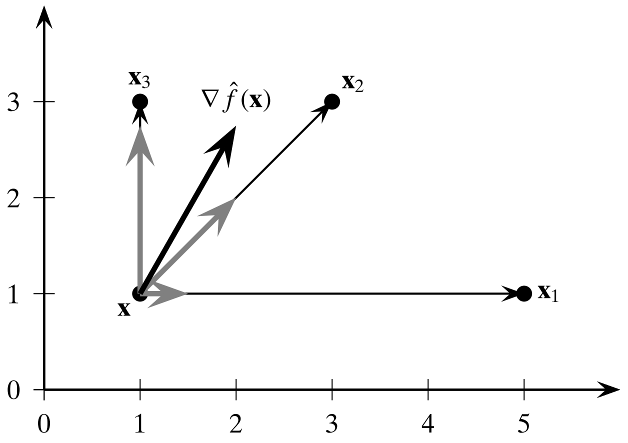


Figure 15.9. The gradient vector $\nabla\hat{f}(\mathbf{x})$ (shown in thick black) obtained as the sum of difference vectors $\mathbf{x}_i - \mathbf{x}$ (shown in gray).

where $\delta > 0$ is the step size. That is, each intermediate point is obtained after a small move in the direction of the gradient vector. However, the gradient-ascent approach can be slow to converge. Instead, one can directly optimize the move direction by setting the gradient [Eq. (15.6)] to the zero vector:

$$\begin{aligned}\nabla\hat{f}(\mathbf{x}) &= \mathbf{0}\\ \frac{1}{nh^{d+2}}\sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)\cdot(\mathbf{x}_i-\mathbf{x}) &= \mathbf{0}\\ \mathbf{x}\cdot\sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) &= \sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)\mathbf{x}_i\\ \mathbf{x} &= \frac{\sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)\mathbf{x}_i}{\sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)}\end{aligned}$$

The point $\mathbf{x}$ is involved on both the left- and right-hand sides above; however, it can be used to obtain the following iterative update rule:

$$\mathbf{x}_{t+1} = \frac{\sum_{i=1}^{n} K\left(\frac{\mathbf{x}_t-\mathbf{x}_i}{h}\right)\mathbf{x}_i}{\sum_{i=1}^{n} K\left(\frac{\mathbf{x}_t-\mathbf{x}_i}{h}\right)} \tag{15.7}$$

where t denotes the current iteration and $\mathbf{x}_{t+1}$ is the updated value for the current vector $\mathbf{x}_t$. This direct update rule is essentially a weighted average of the influence (computed via the kernel function K) of each point $\mathbf{x}_i \in \mathbf{D}$ on the current point $\mathbf{x}_t$. The direct update rule results in much faster convergence of the hill-climbing process.

Center-defined Cluster

A cluster $C \subseteq \mathbf{D}$, is called a *center-defined cluster* if all the points $\mathbf{x} \in C$ are density attracted to a unique density attractor $\mathbf{x}^*$, such that $\hat{f}(\mathbf{x}^*) \geq \xi$, where ξ is a user-defined minimum density threshold. In other words,

$$\hat{f}(\mathbf{x}^*) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x}^* - \mathbf{x}_i}{h}\right) \geq \xi$$

Density-based Cluster

An arbitrary-shaped cluster $C \subseteq \mathbf{D}$ is called a *density-based cluster* if there exists a set of density attractors $\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_m^*$, such that

1. Each point $\mathbf{x} \in C$ is attracted to some attractor $\mathbf{x}_i^*$.
2. Each density attractor has density above ξ. That is, $\hat{f}(\mathbf{x}_i^*) \geq \xi$.
3. Any two density attractors $\mathbf{x}_i^*$ and $\mathbf{x}_j^*$ are *density reachable*, that is, there exists a path from $\mathbf{x}_i^*$ to $\mathbf{x}_j^*$, such that for all points $\mathbf{y}$ on the path, $\hat{f}(\mathbf{y}) \geq \xi$.

DENCLUE Algorithm

The pseudo-code for DENCLUE is shown in Algorithm 15.2. The first step is to compute the density attractor $\mathbf{x}^*$ for each point $\mathbf{x}$ in the dataset (line 4). If the density at $\mathbf{x}^*$ is above the minimum density threshold ξ, the attractor is added to the set of attractors $\mathcal{A}$. The data point $\mathbf{x}$ is also added to the set of points $R(\mathbf{x}^*)$ attracted to $\mathbf{x}^*$

ALGORITHM 15.2. DENCLUE Algorithm

```
DENCLUE ($\mathbf{D}, h, \xi, \epsilon$):
1  $\mathcal{A} \leftarrow \emptyset$
2  foreach $\mathbf{x} \in \mathbf{D}$ do // find density attractors
4      $\mathbf{x}^* \leftarrow$ FINDATTRACTOR($\mathbf{x}, \mathbf{D}, h, \epsilon$)
5      if $\hat{f}(\mathbf{x}^*) \geq \xi$ then
7          $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathbf{x}^*\}$
9          $R(\mathbf{x}^*) \leftarrow R(\mathbf{x}^*) \cup \{\mathbf{x}\}$
11 $\mathcal{C} \leftarrow \{$maximal $C \subseteq \mathcal{A} \mid \forall \mathbf{x}_i^*, \mathbf{x}_j^* \in C, \mathbf{x}_i^*$ and $\mathbf{x}_j^*$ are density reachable$\}$
12 foreach $C \in \mathcal{C}$ do // density-based clusters
13     foreach $\mathbf{x}^* \in C$ do $C \leftarrow C \cup R(\mathbf{x}^*)$
14 return $\mathcal{C}$

FINDATTRACTOR ($\mathbf{x}, \mathbf{D}, h, \epsilon$):
16 $t \leftarrow 0$
17 $\mathbf{x}_t \leftarrow \mathbf{x}$
18 repeat
20     $\mathbf{x}_{t+1} \leftarrow \frac{\sum_{i=1}^{n} K\left(\frac{\mathbf{x}_t - \mathbf{x}_i}{h}\right) \cdot \mathbf{x}_t}{\sum_{i=1}^{n} K\left(\frac{\mathbf{x}_t - \mathbf{x}_i}{h}\right)}$
21     $t \leftarrow t + 1$
22 until $\|\mathbf{x}_t - \mathbf{x}_{t-1}\| \leq \epsilon$
24 return $\mathbf{x}_t$
```

(line 9). In the second step, DENCLUE finds all the maximal subsets of attractors $C \subseteq \mathcal{A}$, such that any pair of attractors in C is density-reachable from each other (line 11). These maximal subsets of mutually reachable attractors form the seed for each density-based cluster. Finally, for each attractor $\mathbf{x}^* \in C$, we add to the cluster all of the points $R(\mathbf{x}^*)$ that are attracted to $\mathbf{x}^*$, which results in the final set of clusters $\mathcal{C}$.

The FINDATTRACTOR method implements the hill-climbing process using the direct update rule [Eq. (15.7)], which results in fast convergence. To further speed up the influence computation, it is possible to compute the kernel values for only the nearest neighbors of $\mathbf{x}_t$. That is, we can index the points in the dataset $\mathbf{D}$ using a spatial index structure, so that we can quickly compute all the nearest neighbors of $\mathbf{x}_t$ within some radius r. For the Gaussian kernel, we can set $r = h \cdot z$, where h is the influence parameter that plays the role of standard deviation, and z specifies the number of standard deviations. Let $B_d(\mathbf{x}_t, r)$ denote the set of all points in $\mathbf{D}$ that lie within a d-dimensional ball of radius r centered at $\mathbf{x}_t$. The nearest neighbor based update rule can then be expressed as

$$\mathbf{x}_{t+1} = \frac{\sum_{\mathbf{x}_i \in B_d(\mathbf{x}_t, r)} K\left(\frac{\mathbf{x}_t - \mathbf{x}_i}{h}\right) \mathbf{x}_i}{\sum_{\mathbf{x}_i \in B_d(\mathbf{x}_t, r)} K\left(\frac{\mathbf{x}_t - \mathbf{x}_i}{h}\right)}$$

which can be used in line 20 in Algorithm 15.2. When the data dimensionality is not high, this can result in a significant speedup. However, the effectiveness deteriorates rapidly with increasing number of dimensions. This is due to two effects. The first is that finding $B_d(\mathbf{x}_t, r)$ reduces to a linear-scan of the data taking $O(n)$ time for each query. Second, due to the *curse of dimensionality* (see Chapter 6), nearly all points appear to be equally close to $\mathbf{x}_t$, thereby nullifying any benefits of computing the nearest neighbors.

Example 15.8. Figure 15.10 shows the DENCLUE clustering for the 2-dimensional Iris dataset comprising the `sepal length` and `sepal width` attributes. The results were obtained with $h = 0.2$ and $\xi = 0.08$, using a Gaussian kernel. The clustering is obtained by thresholding the probability density function in Figure 15.7b at $\xi = 0.08$. The two peaks correspond to the two final clusters. Whereas `iris setosa` is well separated, it is hard to separate the other two types of Irises.

Example 15.9. Figure 15.11 shows the clusters obtained by DENCLUE on the density-based dataset from Figure 15.1. Using the parameters $h = 10$ and $\xi = 9.5 \times 10^{-5}$, with a Gaussian kernel, we obtain eight clusters. The figure is obtained by slicing the density function at the density value ξ; only the regions above that value are plotted. All the clusters are correctly identified, with the exception of the two semicircular clusters on the lower right that appear merged into one cluster.

DENCLUE: Special Cases

It can be shown that DBSCAN is a special case of the general kernel density estimate based clustering approach, DENCLUE. If we let $h = \epsilon$ and $\xi = minpts$, then using a

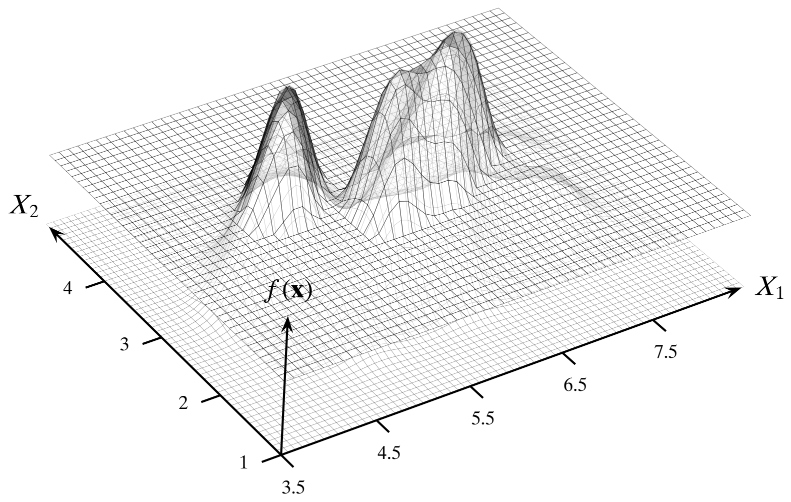


Figure 15.10. DENCLUE: Iris 2D dataset.

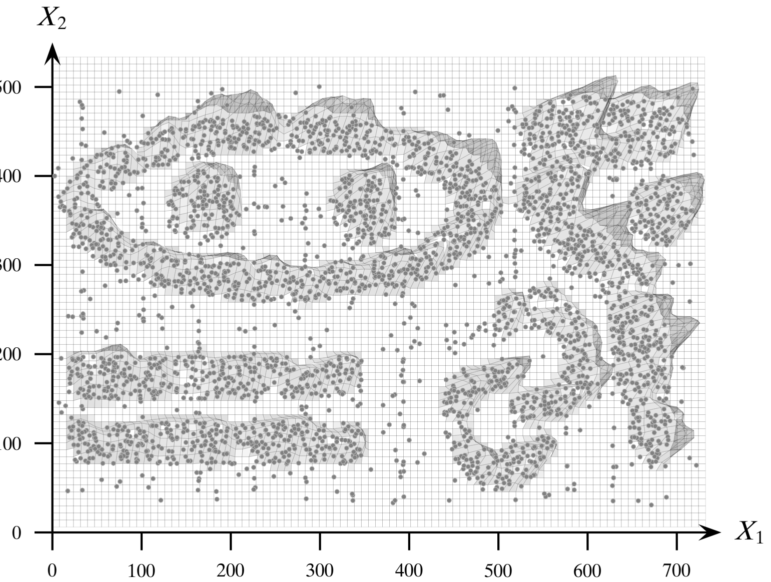


Figure 15.11. DENCLUE: density-based dataset.

discrete kernel DENCLUE yields exactly the same clusters as DBSCAN. Each density attractor corresponds to a core point, and the set of connected core points define the attractors of a density-based cluster. It can also be shown that K-means is a special case of density-based clustering for appropriates value of h and ξ, with the density attractors corresponding to the cluster centroids. Further, it is worth noting that the density-based approach can produce hierarchical clusters, by varying the ξ threshold.

For example, decreasing ξ can result in the merging of several clusters found at higher thresholds values. At the same time it can also lead to new clusters if the peak density satisfies the lower ξ value.

Computational Complexity

The time for DENCLUE is dominated by the cost of the hill-climbing process. For each point $\mathbf{x} \in \mathbf{D}$, finding the density attractor takes $O(nt)$ time, where t is the maximum number of hill-climbing iterations. This is because each iteration takes $O(n)$ time for computing the sum of the influence function over all the points $\mathbf{x}_i \in \mathbf{D}$. The total cost to compute density attractors is therefore $O(n^2t)$. We assume that for reasonable values of h and ξ, there are only a few density attractors, that is, $|\mathcal{A}| = m \ll n$. The cost of finding the maximal reachable subsets of attractors is $O(m^2)$, and the final clusters can be obtained in $O(n)$ time.

15.4 FURTHER READING

Kernel density estimation was developed independently in Rosenblatt (1956) and Parzen (1962). For an excellent description of density estimation techniques see Silverman (1986). The density-based DBSCAN algorithm was introduced in Ester et al. (1996). The DENCLUE method was proposed in Hinneburg and Keim (1998), with the faster direct update rule appearing in Hinneburg and Gabriel (2007). However, the direct update rule is essentially the *mean-shift* algorithm first proposed in Fukunaga and Hostetler (1975). See Cheng (1995) for convergence properties and generalizations of the mean-shift method.

Cheng, Y. (1995). “Mean shift, mode seeking, and clustering.” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 17 (8): 790–799.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). “A density-based algorithm for discovering clusters in large spatial databases with noise.” *In Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining* (pp. 226–231), edited by E. Simoudis, J. Han, and U. M. Fayyad. Palo Ato, CA: AAAI Press.

Fukunaga, K. and Hostetler, L. (1975). “The estimation of the gradient of a density function, with applications in pattern recognition.” *IEEE Transactions on Information Theory*, 21 (1): 32–40.

Hinneburg, A. and Gabriel, H.-H. (2007). “Denclue 2.0: Fast clustering based on kernel density estimation.” *In Proceedings of the 7th International Symposium on Intelligent Data Analysis* (pp. 70–80). New York: Springer Science+Business Media.

Hinneburg, A. and Keim, D. A. (1998). *“An efficient approach to clustering in large multimedia databases with noise.” In Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining* (pp. 58–65), edited by R. Agrawal and P. E. Stolorz. Palo Alto, CA: AAAI Press.

Parzen, E. (1962). On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33 (3): 1065–1076.

Rosenblatt, M. (1956). "Remarks on some nonparametric estimates of a density function." *The Annals of Mathematical Statistics*, 27 (3): 832–837.

Silverman, B. (1986). *Density Estimation for Statistics and Data Analysis*. Monographs on Statistics and Applied Probability. Boca Raton, FL: Chapman and Hall/CRC.

15.5 EXERCISES

Q1. Consider Figure 15.12 and answer the following questions, assuming that we use the Euclidean distance between points, and that $\epsilon = 2$ and $minpts = 3$

(a) List all the core points.

(b) Is a directly density reachable from d?

(c) Is o density reachable from i? Show the intermediate points on the chain or the point where the chain breaks.

(d) Is density reachable a symmetric relationship, that is, if x is density reachable from y, does it imply that y is density reachable from x? Why or why not?

(e) Is l density connected to x? Show the intermediate points that make them density connected or violate the property, respectively.

(f) Is density connected a symmetric relationship?

(g) Show the density-based clusters and the noise points.

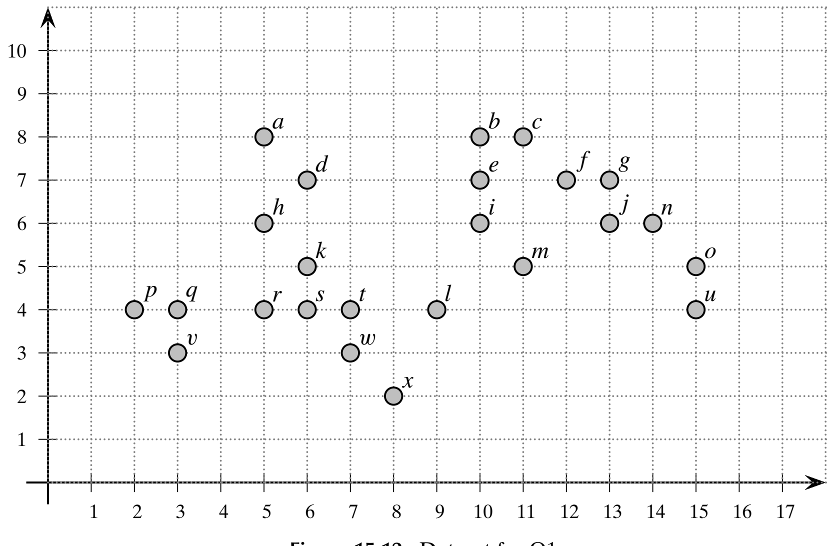


Figure 15.12. Dataset for Q1.

Q2. Consider the points in Figure 15.13. Define the following distance measures:

$$L_\infty(\mathbf{x}, \mathbf{y}) = \max_{i=1}^{d} \left\{ |x_i - y_i| \right\}$$

$$L_{\frac{1}{2}}(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{d} |x_i - y_i|^{\frac{1}{2}} \right)^2$$

$$L_{\min}(\mathbf{x},\mathbf{y}) = \min_{i=1}^{d}\{|x_i - y_i|\}$$

$$L_{pow}(\mathbf{x},\mathbf{y}) = \Big(\sum_{i=1}^{d} 2^{i-1}(x_i - y_i)^2\Big)^{1/2}$$

(a) Using $\epsilon = 2$, $minpts = 5$, and L_∞ distance, find all core, border, and noise points.
(b) Show the shape of the ball of radius $\epsilon = 4$ using the $L_{\frac{1}{2}}$ distance. Using $minpts = 3$ show all the clusters found by DBSCAN.
(c) Using $\epsilon = 1$, $minpts = 6$, and $L_{\min}$, list all core, border, and noise points.
(d) Using $\epsilon = 4$, $minpts = 3$, and L_{pow}, show all clusters found by DBSCAN.

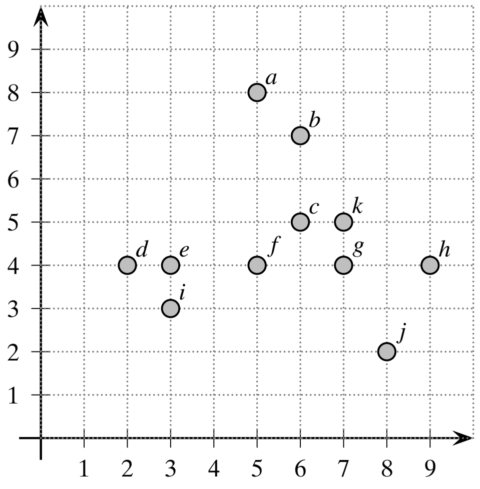


Figure 15.13. Dataset for Q2 and Q3.

Q3. Consider the points shown in Figure 15.13. Define the following two kernels:

$$K_1(\mathbf{z}) = \begin{cases} 1 & \text{If } L_\infty(\mathbf{z},\mathbf{0}) \le 1 \\ 0 & \text{Otherwise} \end{cases}$$

$$K_2(\mathbf{z}) = \begin{cases} 1 & \text{If } \sum_{j=1}^{d} |z_j| \le 1 \\ 0 & \text{Otherwise} \end{cases}$$

Using each of the two kernels K_1 and K_2, answer the following questions assuming that $h = 2$:
(a) What is the probability density at e?
(b) What is the gradient at e?
(c) List all the density attractors for this dataset.

Q4. The Hessian matrix is defined as the set of partial derivatives of the gradient vector with respect to $\mathbf{x}$. What is the Hessian matrix for the Gaussian kernel? Use the gradient in Eq. (15.6).

Q5. Let us compute the probability density at a point x using the k-nearest neighbor approach, given as

$$\hat{f}(x) = \frac{k}{nV_x}$$

where k is the number of nearest neighbors, n is the total number of points, and V_x is the volume of the region encompassing the k nearest neighbors of x. In other words, we fix k and allow the volume to vary based on those k nearest neighbors of x. Given the following points

$$2, 2.5, 3, 4, 4.5, 5, 6.1$$

Find the peak density in this dataset, assuming $k = 4$. Keep in mind that this may happen at a point other than those given above. Also, a point is its own nearest neighbor.