Chapter 15

Density-based Clustering

The clustering methods like K-means or Expectation-Maximization are suitable for finding ellipsoid-shaped clusters, or at best convex clusters. However, for non-convex clusters, such as those shown in Figure 15.1, these methods have trouble finding the true clusters, since 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 non-convex or shape-based clusters.

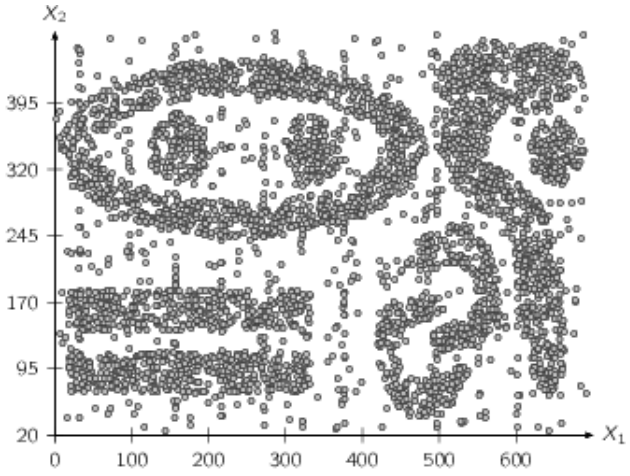


Figure 1.1: Density-based Clusters

Figure 15.1: Density-based Dataset

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. Define a ball of radius ϵ around a point $\mathbf{x}$, called the ϵ-*neighborhood* of $\mathbf{x}$, as follows

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

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, i.e., $\delta(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2$. 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 neighborhood. In other words, $\mathbf{x}$ is a core point if $|N_\epsilon(\mathbf{x})| \geq minpts$, 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, i.e., it has $|N_\epsilon(\mathbf{x})| < minpts$, but it belongs to the neighborhood of some core point $\mathbf{z}$, i.e., $\mathbf{x} \in N(\mathbf{z})$. Finally, if a point is neither a core nor a border point, then it is called a *noise point* or an outlier.

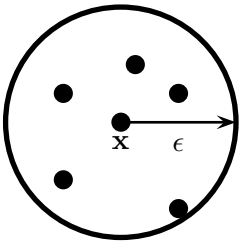


Figure 15.2: Neighborhood of a Point

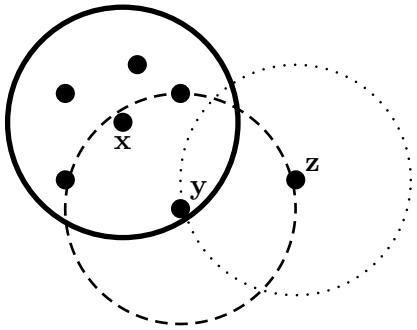


Figure 15.3: Core, Border and Noise Points

Example 15.1: Figure 15.2 shows the ϵ-neighborhood of the point $\mathbf{x}$, using the Euclidean distance metric. Figure 15.3 shows the three different types of points, using $minpts = 6$. Here $\mathbf{x}$ is a core point since $|N_\epsilon(\mathbf{x})| = 6$. $\mathbf{y}$ is a border point, since $|N_\epsilon(\mathbf{y})| = 3$, but it is reachable from $\mathbf{x}$. 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 of points, $\mathbf{x} = \mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_l = \mathbf{y}$, such that $\mathbf{x}_i$ is directly density reachable from $\mathbf{x}_{i-1}$. 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. Finally, 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 $\mathbf{z}$. We can now define a density-based cluster as a maximal set of density connected points.

Algorithm 15.1: Density-based Clustering Algorithm

```
DBSCAN (D, ϵ, minpts) :
Cores = ∅
foreach x ∈ D do
    // Find the core points
    if N_ϵ(x) ≥ minpts then
        Cores = Cores ∪ {x}
k = 0
foreach x ∈ Cores, such that x is unmarked do
    k = k + 1
    DENSITYCONNECTED (x, k)
C = {C_i}_{i=1}^k, where C_i = {x ∈ D : x has cluster id i}
Noise = {x ∈ D : x is unmarked}
Border = D − (C ∪ Noise)
return C, Border, Noise

DENSITYCONNECTED (x, k):
Mark x with current cluster id k
foreach y ∈ N_ϵ(x) do
    Mark y with current cluster id k
    if y ∈ Cores and y is unmarked then
        DENSITYCONNECTED (y, k)
```

Algorithm 15.1 shows the pseudo-code for the DBSCAN algorithm. Initially, all the points are unmarked. First, it computes the neighborhood $N_\epsilon(\mathbf{x})$ for each point

$\mathbf{x}$ in the dataset $\mathbf{D}$, and checks if it is a core point (lines 2 - 4). Next, starting from each unmarked core, the method recursively finds all other points density connected to it; all such points belong to the same cluster (line 9). Some border point may be reachable from core points in more than one cluster. Such points may either be arbitrarily assigned to one of the clusters, or they may be assigned to all of them. Those points that do not belong to any cluster are treated as outliers or noise.

The density-based clustering algorithm can also be described 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 (i.e., each of them is in the 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.

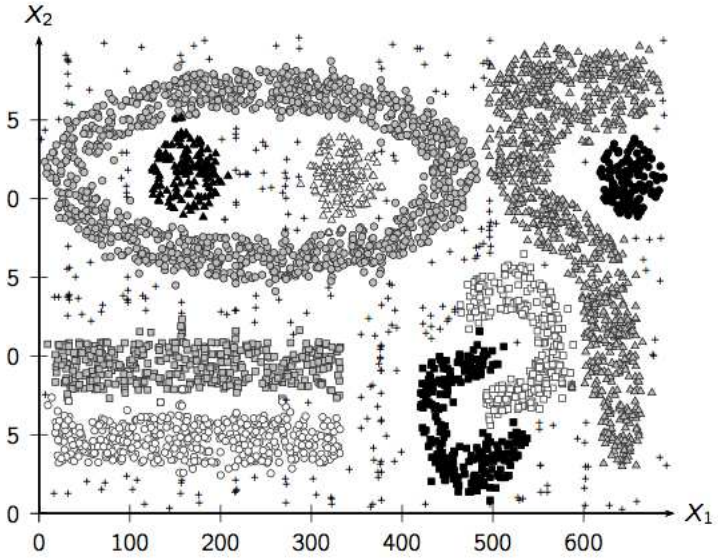
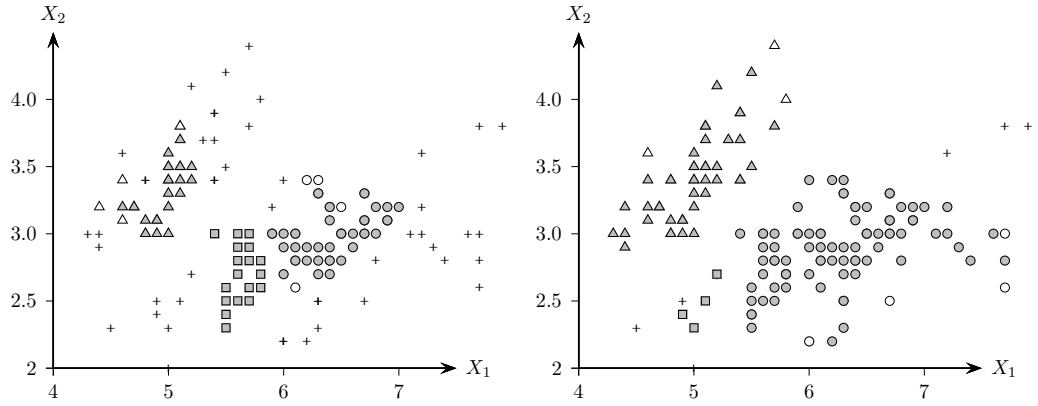


Figure 15.4: Density-based Clusters

Example 15.2: Figure 15.4 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, with $k = 9$ clusters. Cluster are shown using different symbols and shading, and noise points are shown as plus symbols.



(a) $\epsilon = 0.2$, *minpts* = 5 (b) $\epsilon = 0.325$, *minpts* = 5

Figure 15.5: Denclue Clustering: Iris 2D Dataset

Example 15.3: Figure 15.5 shows the clusterings obtained via DBSCAN on the Iris two-dimensional dataset, for two different parameter settings. Figure 15.5a shows the clusters obtained with $\epsilon = 0.2$ and *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 15.5b shows the clusters obtained with a larger value of radius ($\epsilon = 0.325$), with *minpts* = 5. Two clusters are found, corresponding to the two dense regions of points.

Computational Complexity The main effort in density-based clustering is to compute 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 maximal density connected clusters. Thus, the overall complexity is $O(n^2)$ in the worst-case.

15.2 Kernel Density Estimation

The density based clustering described above is a special case of kernel density estimation. The main goal of density estimation is to find the dense regions of points, which is essentially the same as clustering. Kernel density estimation is a non-parametric technique that does not assume any fixed probability model of the clusters, as in the case of K-means or model-based clustering via the EM algorithm. Instead, kernel density estimation tries to 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 $x_1, x_2, \cdots, x_n$ be a random sample drawn from the underlying probability density function $f(x)$, which is assumed to be unknown. Let $F(x)$ denote the cumulative distribution function. For a continuous distribution it is given as

$$F(x) = \int_{-\infty}^{x} f(x)\, dx,$$

subject to the condition that $\int_{-\infty}^{\infty} f(x)\, dx = 1$.

Notice how the probability density $f(x)$ is the derivative of the cumulative distribution $F(x)$, whereas $F(x)$ is the integral of $f(x)$. Let $\hat{F}(x)$ be the estimate of $F(x)$ at x. The cumulative distribution can be directly estimated from the data by counting how many points are less than or equal to x

$$\hat{F}(x) = \frac{|\{x_i | x_i \leq x\}|}{n}$$

Let $\hat{f}(x)$ denote the estimate of $f(x)$ at x. We can estimate $\hat{f}(x)$ by considering a window of width h centered at x, and then computing the derivative at x

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

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, since 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.

The probability density $\hat{f}(x)$ can be estimated using an alternate formulation. For any closed interval $[a, b]$, the cumulative probability distribution is given as

$$P_{ab} = \int_{a}^{b} f(x)\, dx$$

Given the n points $x_1, x_2, \cdots, x_n$, P_{ab} gives the probability that a point belongs to the closed interval $[a, b]$. Assuming that the points are independently and identically distributed according to $f(x)$, then the probability that exactly m points belong to the closed interval is given by the *binomial distribution*

$$f(m \mid \ n, P_{ab}) = \binom{n}{m} (P_{ab})^m (1 - P_{ab})^{n-m}$$

The expected number of points that fall in the interval $[a, b]$, is then given as

$$E[m] = nP_{ab}$$

Since the binomial distribution has a very sharp peak at the expected value, letting $E[m] = k$, we can estimate P_{ab} as

$$\hat{F}_{ab} = \frac{k}{n} \tag{15.2}$$

Now, assuming that the closed interval $[a, b]$ centered at x is small, i.e., assuming that width of the interval $h = b - a$ is small, then another way to estimate P_{ab} is as follows

$$\hat{F}_{ab} = \int_a^b f(x)\, dx \simeq \hat{f}(x)h \tag{15.3}$$

Combining (15.2) and (15.3), we have

$$\hat{f}(x)h = \frac{k}{n} \implies \hat{f}(x) = \frac{k/n}{h} = \frac{k}{nh} \tag{15.4}$$

Combining the above with (15.1) we see that

$$\frac{k}{n} = \hat{F}\left(x + \frac{h}{2}\right) - \hat{F}\left(x - \frac{h}{2}\right)$$

which equates the fraction of points in the closed interval with the difference between the cumulative distribution at the two ends of that interval. In other words (15.1) and (15.4) are equivalent, though the latter is more intuitive, since it gives the density estimate as the ratio of the fraction of the points in the region to the volume of the region.

Discrete Kernel The density estimate $\hat{f}(x)$ from (15.4) can also be re-written in terms of a **kernel function** as shown below

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

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

$$K(z) = \begin{cases} 1 & \text{If } |z| \leq \frac{1}{2} \\ 0 & \text{Otherwise} \end{cases} \tag{15.6}$$

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, since

$$\begin{aligned} \left|\frac{x-x_i}{h}\right| \leq \frac{1}{2} &\implies -\frac{1}{2} \leq \frac{x-x_i}{h} \leq \frac{1}{2} \\ &\implies -\frac{h}{2} \leq x - x_i \leq \frac{h}{2} \\ &\implies -x - \frac{h}{2} \leq x_i \leq -x + \frac{h}{2} \\ &\implies x + \frac{h}{2} \geq x_i \geq x - \frac{h}{2} \end{aligned}$$

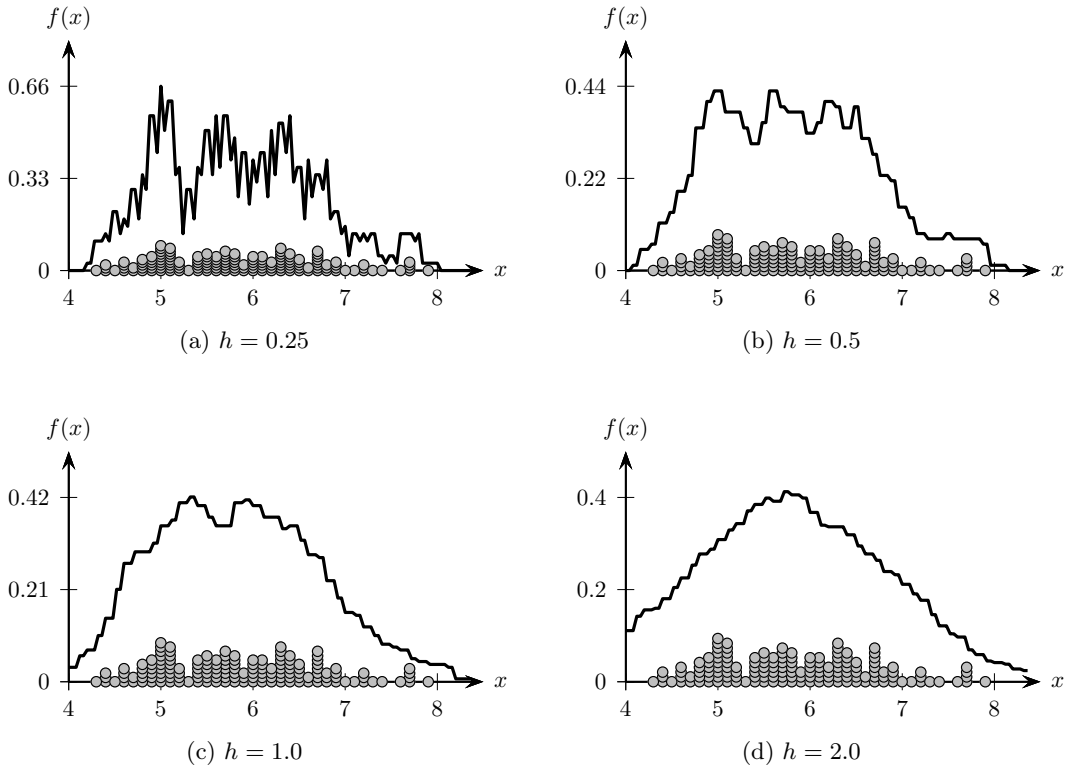
Example 15.4: Figure 15.6 shows the kernel density estimates using the Discrete kernel for different values of the influence parameter h, for the one-dimensional Iris dataset comprising the `sepal length` attribute. The x-axis also plots the $n = 150$ data points. Since several points have the same value, they are shown stacked, where the stack height corresponds to value frequency.

When h is small, as shown in Figure 15.6a, 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 15.6d). We can observe that the Discrete kernel yields a non-smooth (or jagged) density function.

Gaussian Kernel The width h is a parameter which 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. Furthermore, the kernel function in (15.6) 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.

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

$$K(z) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{z^2}{2}\right\} \tag{15.7}$$



(a) $h = 0.25$ (b) $h = 0.5$

(c) $h = 1.0$ (d) $h = 2.0$

Figure 15.6: Kernel Density Estimation: Discrete Kernel (varying h)

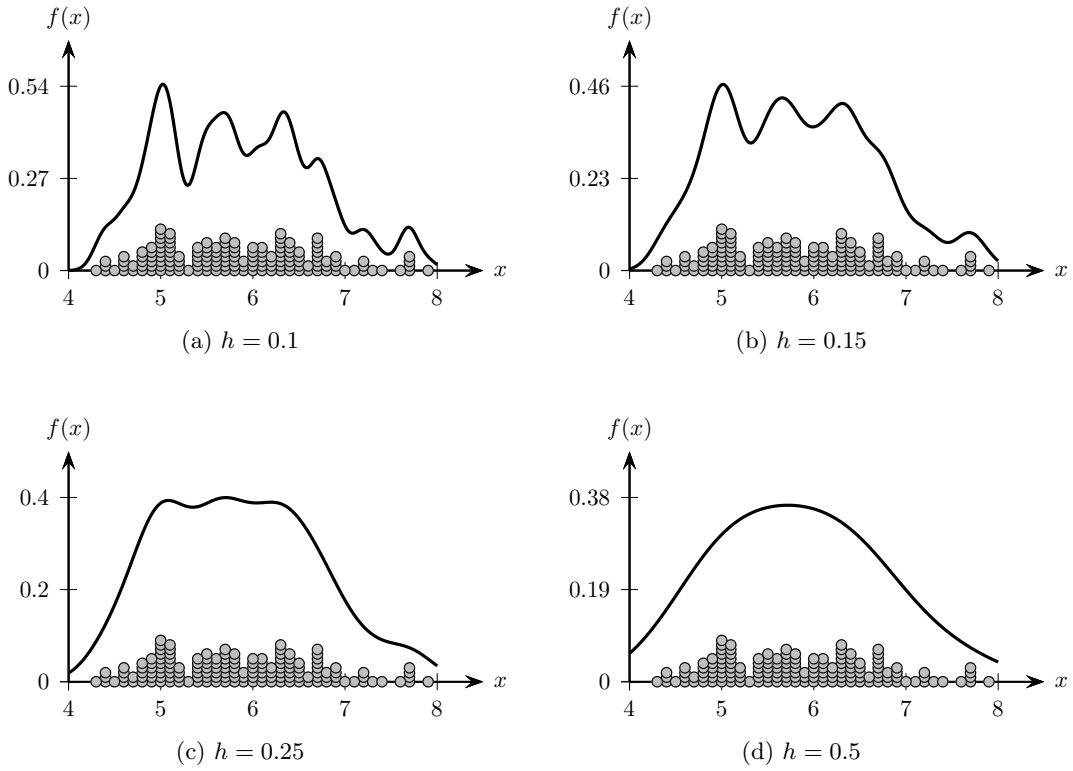
Thus we have

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

Here x (the center of the window) acts as the mean of the distribution, and h acts as the standard deviation of the distribution.

Example 15.5: Figure 15.7 shows the univariate density function for the one-dimensional Iris dataset, using the Gaussian kernel, 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 clearly 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 15.6, we can clearly see that the Gaussian kernel yields much smoother estimates, without discontinuities.



(a) $h = 0.1$ (b) $h = 0.15$

(c) $h = 0.25$ (d) $h = 0.5$

Figure 15.7: Kernel Density Estimation: Gaussian Kernel (Varying h)

15.2.2 Multivariate Density Estimation

To estimate the probability density at a d-dimensional point $\mathbf{x} = (x_1, x_2, ...x_d)$, we define the d-dimensional "window" as a hypercube in d-dimensions with edge-length h, i.e., a window centered at $\mathbf{x}$ with width 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

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \tag{15.8}$$

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

$$K(\mathbf{z}) = \begin{cases} 1 & \text{If } |z_j| \leq \frac{1}{2}, \text{ for all dimensions } j = 1, \cdots, d \\ 0 & \text{Otherwise} \end{cases} \tag{15.9}$$

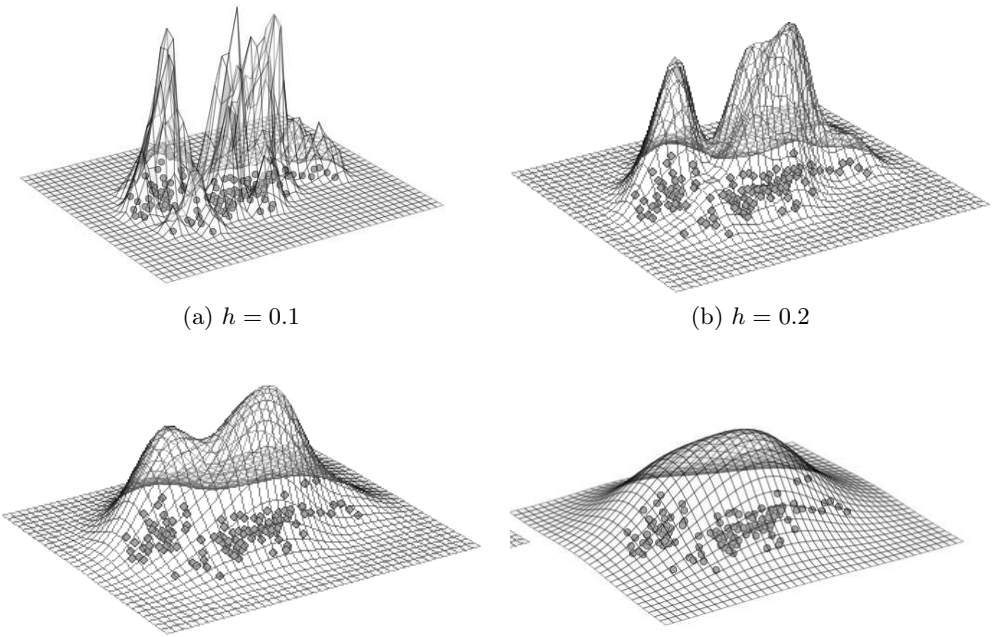
For $\mathbf{z} = \frac{\mathbf{x}-\mathbf{x}_i}{h}$, we see that the kernel computes the number of points within the hypercube of width h centered at $\mathbf{x}$, since $K(\frac{\mathbf{x}-\mathbf{x}_i}{h}) = 1$ if and only if $|\frac{x_j - x_{ij}}{h}| \leq \frac{1}{2}$ for all dimensions j.

Further, the d-dimensional Gaussian kernel (with $\mathbf{\Sigma} = \mathbf{I}_d$) is given as

$$K(\mathbf{z}) = \frac{1}{(2\pi)^{d/2}} \exp\left\{-\frac{\mathbf{z}^T\mathbf{z}}{2}\right\} \tag{15.10}$$

Putting $\mathbf{z} = \frac{\mathbf{x}-\mathbf{x}_i}{h}$, we have

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



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

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

Figure 15.8: Density Estimation: 2D Iris Dataset (varying h)

Example 15.6: Figure 15.8 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.

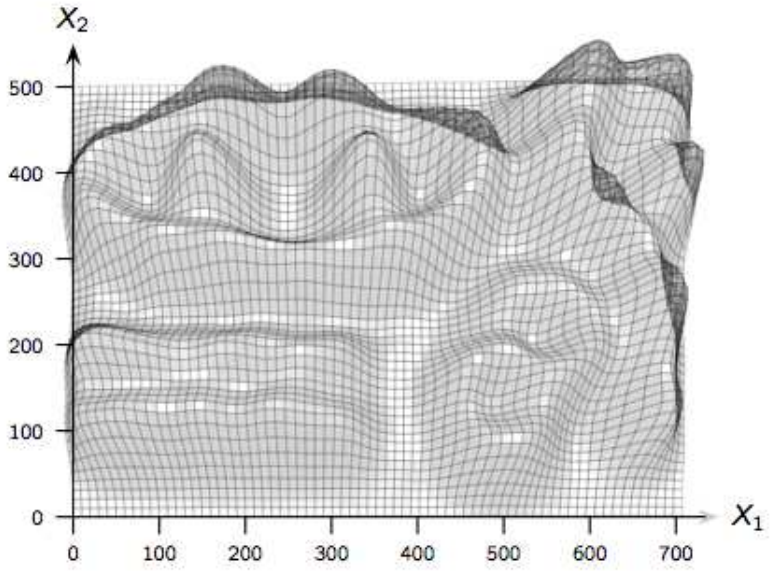


Figure 15.9: Density Estimation: Density-based Dataset

Example 15.7: Figure15.9 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 density estimation formulation above we implicitly fixed the volume of the hybercube by fixing the edge length h. The kernel function was used to find out the number 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 hypercube to vary to accommodate those k points in (15.4). This approach is called the k-nearest neighbor (KNN) approach to density estimation. Like kernel density estimation, KNN density estimation is also a non-parametric approach.

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

$$\hat{f}(\mathbf{x}) = \frac{k}{n \, \text{vol}_d(h_{\mathbf{x}})} = \frac{k}{n(h_{\mathbf{x}})^d}$$

where $h_{\mathbf{x}}$ is the distance from $\mathbf{x}$ to its k-th nearest neighbor. In other words, the width of the hypercube is now a variable, which depends on $\mathbf{x}$ and the chosen value k. As before, we assume that the d-dimensional hypercube is centered at $\mathbf{x}$.

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 those 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 maximum 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 maximum.

The gradient at a point $\mathbf{x}$ can be computed as the multivariate derivative of the probability density estimate in (15.8), i.e.,

$$\nabla \hat{f}(\mathbf{x}) = \frac{\partial}{\partial \mathbf{x}} \hat{f} = \frac{1}{nh^d} \sum_{i=1}^{n} \frac{\partial}{\partial \mathbf{x}} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \tag{15.11}$$

For the Gaussian kernel (15.10), 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

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

Substituting the above in (15.11), the gradient at a point $\mathbf{x}$ is given as

$$\nabla \hat{f}(\mathbf{x}) = \frac{1}{nh^{d+2}} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \cdot (\mathbf{x}_i - \mathbf{x}) \tag{15.12}$$

The above equation can be thought of as having two parts. A vector $(\mathbf{x}_i - \mathbf{x})$ and a scalar *influence* value $K(\frac{\mathbf{x}_i-\mathbf{x}}{h})$. Thus, the gradient is the net vector obtained as the weighted sum of all other points in terms of the kernel function K. For

each point $\mathbf{x}_i$, we first compute the direction away from $\mathbf{x}$, i.e., the vector $(\mathbf{x}_i - \mathbf{x})$. Next, we scale the magnitude of this vector using the Gaussian kernel function as the influence $K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)$. Finally, the $\nabla \hat{f}(\mathbf{x})$ vector is the net influence at $\mathbf{x}$, as illustrated in Figure 15.10.

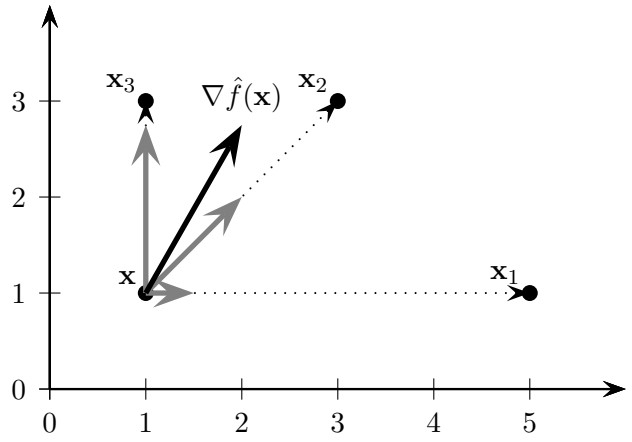


Figure 15.10: The Gradient Vector

We say that a point $\mathbf{x}$ is *density attracted* to another point $\mathbf{x}^*$ if a gradient ascent process started at $\mathbf{x}$ converges to $\mathbf{x}^*$. That is, there exists a sequence of points $\mathbf{x} = \mathbf{x}_0 \rightarrow \mathbf{x}_1 \rightarrow \ldots \rightarrow \mathbf{x}_m$, such that $\|\mathbf{x}_m - \mathbf{x}^*\| \leq \epsilon$, and each intermediate point is obtained after a small move in the direction of the gradient vector

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \delta \cdot \nabla \hat{f}(\mathbf{x}_t) \tag{15.13}$$

where $\delta > 0$ is the step size.

Center-defined Cluster A cluster $C \subseteq \mathbf{D}$, is called a *center defined cluster* $C \subseteq \mathbf{D}$ 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,

$$\hat{f}(\mathbf{x}^*) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x}^* - \mathbf{x}_i}{h}\right) \geq \xi \tag{15.14}$$

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, $\hat{f}(\mathbf{x}_i^*) \geq \xi$.

Algorithm 15.2: DENCLUE Algorithm

```
   DENCLUE (D, h, ξ, ε)        :
1  A ← ∅
   // find density attractors
2  foreach x ∈ D do
4      x* ← FINDATTRACTOR(x, h, ε)
5      if f̂(x*) ≥ ξ then
7          A ← A ∪ {x*}
9          N(x*) ← N(x*) ∪ {x}
11 M = {A ⊆ A : ∀x*_i, x*_j ∈ A, x*_i and x*_j are density reachable}
12 C = ∅
   // density-based clusters
14 foreach A ∈ M do
15     foreach x* ∈ A do C = C ∪ N(x*)
17     C = C ∪ C
18 return C

   FINDATTRACTOR (D, h, ε):
20 t ← 0
21 x_0 ← x
22 repeat
23     if Gradient Ascent then
25         x_{t+1} ← x_t + δ · ∇f̂(x_t)
26     else
28         x_{t+1} ← (Σ_{i=1}^n K((x_t − x_i)/h) · x_t) / (Σ_{i=1}^n K((x_t − x_i)/h))
29     t ← t + 1
30 until ‖x_t − x_{t−1}‖ ≤ ε
32 return x_t
```

3. Any two density attractors $\mathbf{x}_i^*$ and $\mathbf{x}_j^*$ are *density reachable*, i.e., there exists a path from $\mathbf{x}_i^*$ to $\mathbf{x}_j^*$, such that for all points $\mathbf{y}$ on the path, $\hat{f}(\mathbf{y}) \geq \xi$.

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 by invoking the `FindAttractor` routine (line 4). If the density at $\mathbf{x}^*$ is above the minimum density threshld ξ, the attractor is added to the set of attractors $\mathcal{A}$ (line 7). The method also maintains the set of all points $N(\mathbf{x}^*)$ attracted to each attractor $\mathbf{x}^*$ (line 9). In the second step, DENCLUE finds all the maximal subsets of attractors $A \subseteq \mathcal{A}$, such that any pair of attractors in A is density-reachable from each other (line 11);

further, no more attractors can be added to A without destroying this property. The set of all these maximal subsets $\mathcal{M}$ defines the density-based clusters. Each final cluster C comprises all points $\mathbf{x} \in \mathbf{D}$ that are density attracted to some attractor in A, obtained as the union of all the $N(\mathbf{x}^*)$ sets for each $\mathbf{x}^* \in A$ (lines 14 - 17).

The `FindAttractor` method (lines 20 - 32) implements a hill-climbing process. The standard approach is to use gradient-ascent (line 25). Starting from a given $\mathbf{x}$, we iteratively update it using the gradient-ascent update rule (15.13). However, this approach can be slow to converge. Instead, one can directly optimize the move direction by setting the gradient (15.12) to zero

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

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

$$\mathbf{x}_{t+1} = \frac{\sum_{i=1}^{n} K\left(\frac{\mathbf{x}_t-\mathbf{x}_i}{h}\right) \mathbf{x}_i}{\sum_{i=1}^{n} K\left(\frac{\mathbf{x}_t-\mathbf{x}_i}{h}\right)} \tag{15.15}$$

where t denotes the current iteration. 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 (line 28) results in much faster convergence of the hill-climbing process, and is the preferred approach.

For faster influence computation, it is possible to compute the kernel values for only the nearest neighbors of each point $\mathbf{x}_t$. That is, we can index the points in the dataset $\mathbf{D}$ using a spatial index structure (e.g., a k-d Tree), 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 (e.g., we can set $z = 3$ for a two-dimensional dataset, without loosing any accuracy in the density evaluation). Let $B(\mathbf{x}_t, r)$ denote the set of all points in $\mathbf{D}$ that lie within a ball of radius ϵ centered at $\mathbf{x}_t$. The nearest-neighbors update rules can then be expressed as

$$\mathbf{x}_{t+1} = \frac{\sum_{\mathbf{x}_i \in B(\mathbf{x}_t, r)} K\left(\frac{\mathbf{x}_t-\mathbf{x}_i}{h}\right) \mathbf{x}_i}{\sum_{\mathbf{x}_i \in B(\mathbf{x}_t, r)} K\left(\frac{\mathbf{x}_t-\mathbf{x}_i}{h}\right)} \tag{15.16}$$

When the data dimensionality is not high, this can results in significant speedup. However, the effectiveness deteriorates rapidly with increasing number of dimensions. This is due to two effects. The first is that finding $B(\mathbf{x}_t, r)$ 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.

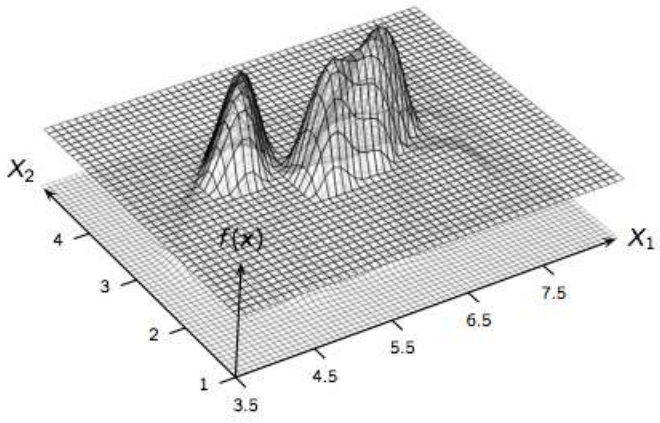


Figure 15.11: DENCLUE: Iris 2D Dataset

Example 15.8: Figure 15.11 shows the DENCLUE clustering for the 2D Iris dataset, with $h = 0.2$ and $\xi = 0.08$, using a Gaussian kernel. The clustering is obtained by thresholding the probability density function in Figure 15.8b at $\xi = 0.08$. The two peaks correspond to the two final clusters.

Example 15.9: Figure 15.12 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 $k = 8$ clusters. The figure is obtained by slicing the density function at the density value ξ, so that only the regions above that value are plotted. All the clusters are correctly identified, with the exception of the two semi-circular 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

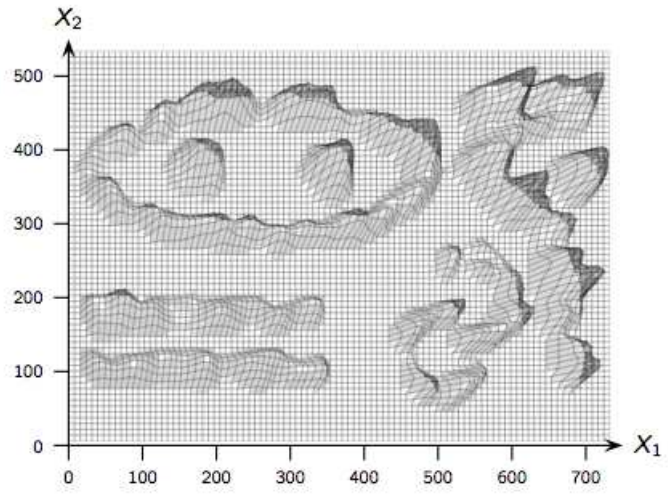


Figure 15.12: DENCLUE: Density-based Dataset

$h = \epsilon$ and $\xi = minpts$, then using a 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 an appropriate 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. In addition, new clusters may emerge if the peak density satisfies the lower ξ value.

Computational Complexity The computational complexity of 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 $O(n^2t)$. It is assumed that for reasonable values of h and ξ, there are only a few density attractors, i.e., $|\mathcal{A}| = m \ll n$. 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. When the dimensionality is small, the use of a spatial index can reduce the complexity of finding all the attractors to $O((n \log n)t)$.

15.4 Annotated References