DBSCAN Clustering

Humaun Farid Sohag

2025-08-13

#DBSCAN for Sample data-1

#Step 1: Install and Load Required Packages

```
# You may need to install these packages if you haven't already.
 #install.packages("ggplot2")
 #install.packages("ggforce")

library(ggplot2)
library(ggforce)
```

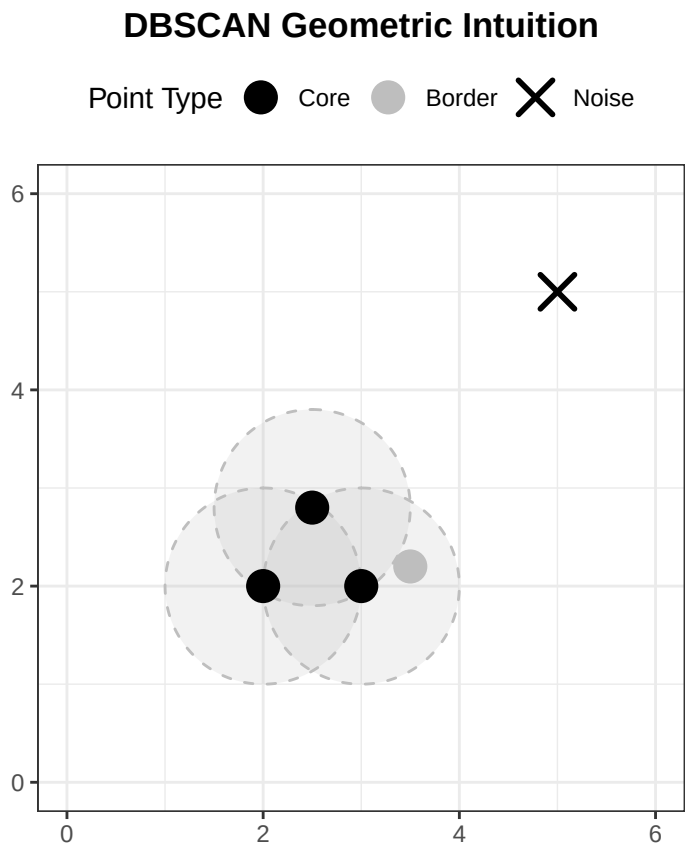
#Step 2: Define the data points and parameters

```
# Combine all points into a single data frame with a 'type' column.
points_df <- data.frame(
  x = c(2, 3, 2.5, 3.5, 5),
  y = c(2, 2, 2.8, 2.2, 5),
  type = factor(c("Core", "Core", "Core", "Border", "Noise"),
                levels = c("Core", "Border", "Noise"))
)

# Define epsilon and create a data frame for the core point circles
epsilon <- 1.0
core_points_circles <- data.frame(
  x0 = c(2, 3, 2.5),
  y0 = c(2, 2, 2.8),
  r = epsilon
)
```

#Step 3: Create the plot using ggplot2

```
ggplot() +
  # Draw the epsilon circles for core points
  geom_circle(
    data = core_points_circles,
    aes(x0 = x0, y0 = y0, r = r),
```

```
  color = "gray",
  fill = "gray",
  alpha = 0.2,
  linetype = "dashed"
) +

# Draw the data points
geom_point(
  data = points_df,
  aes(x = x, y = y, shape = type, color = type),
  size = 5,
  stroke = 1.5 # Makes shapes like 'x' thicker
) +

# Set custom colors and shapes to match the Python plot
scale_color_manual(values = c(Core = "black", Border = "gray", Noise = "black")) +
scale_shape_manual(values = c(Core = 16, Border = 16, Noise = 4)) + # 16 is a solid circle, 4 is an '

# Set axis limits and ensure aspect ratio is equal so circles are not distorted
coord_fixed(xlim = c(0, 6), ylim = c(0, 6)) +

# Add titles and labels
labs(
  title = "DBSCAN Geometric Intuition",
  x = NULL, y = NULL, color = "Point Type", shape = "Point Type"
) +

# Apply a clean theme and add a grid
theme_bw() +
theme(
  plot.title = element_text(hjust = 0.5, face = "bold"),
  legend.position = "top"
)
```



DBSCAN for Sample data-2

Step 1: Install and Load Required Packages

```
# You may need to install these packages if you haven't already.
#install.packages("mlbench")
#install.packages("dbscan")
#install.packages("ggplot2")

library(mlbench)
library(dbscan)
library(ggplot2)
```

Step 2: Generate Synthetic Data

```
# The mlbench.spirals() function can create a similar non-linear dataset.
# We'll use two spirals, which serve a similar purpose to the 'moons' dataset.
# `set.seed()` is the R equivalent of Python's `random_state`.
set.seed(42)
moons_data <- mlbench.spirals(n = 300, cycles = 1, sd = 0.1)
X <- moons_data$x
```

Step 3: Standardize the Data

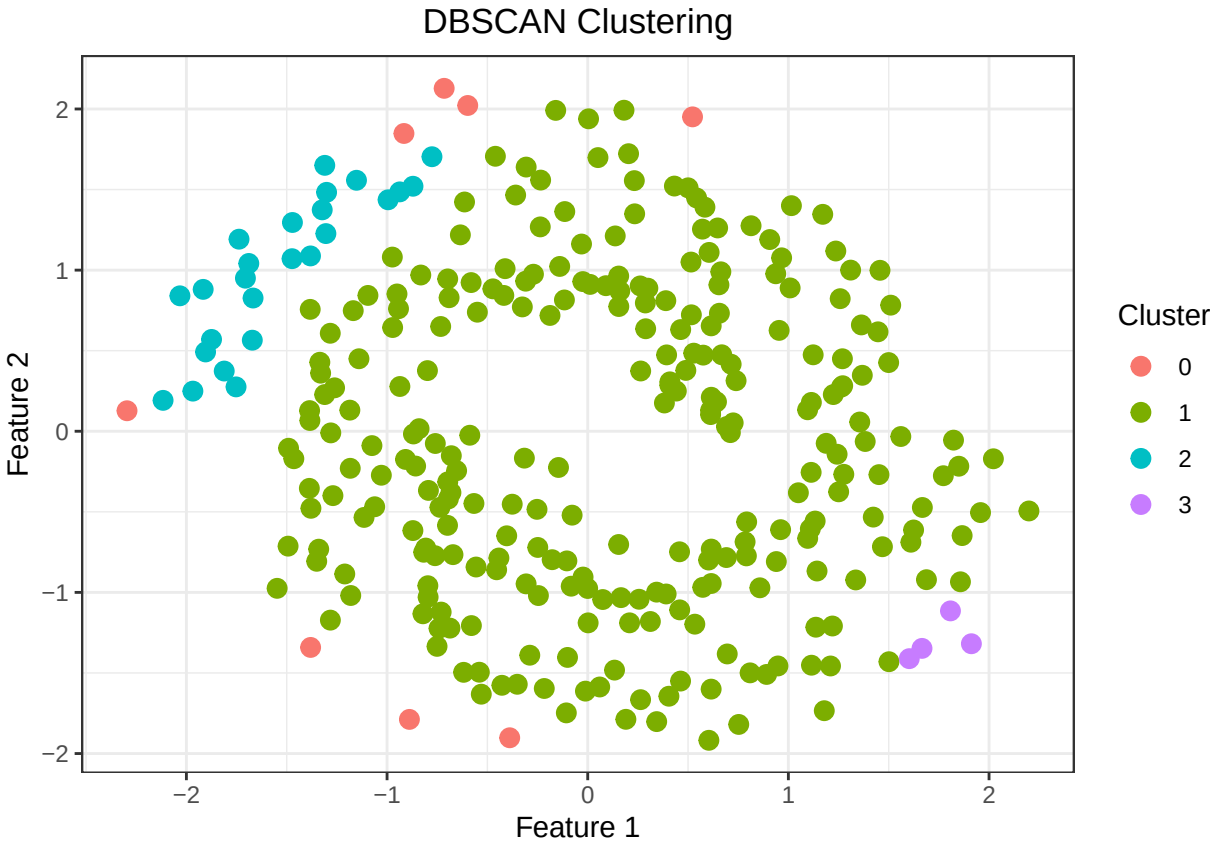
```r
# The scale() function in R standardizes the data (mean=0, sd=1),
# similar to StandardScaler in scikit-learn.
X_scaled <- scale(X)
```

Step 4: Apply DBSCAN Algorithm

```r
# Use the dbscan() function.
# - eps = 0.3: Defines the radius of the neighborhood.
# - minPts = 5: The minimum number of points required for a core point.
dbscan_result <- dbscan(X_scaled, eps = 0.3, minPts = 5)
```

Step 5: Plot the Results

```r
# We use ggplot2 for a high-quality visualization.
# First, create a data frame for plotting.
plot_data <- data.frame(
  Feature1 = X_scaled[, 1],
  Feature2 = X_scaled[, 2],
  Cluster = as.factor(dbscan_result$cluster) # Convert cluster numbers to a factor for coloring
)

# Create the plot
ggplot(plot_data, aes(x = Feature1, y = Feature2, color = Cluster)) +
  geom_point(size = 3) +
  labs(
    title = "DBSCAN Clustering",
    x = "Feature 1",
    y = "Feature 2",
    color = "Cluster"
  ) +
  theme_bw() + # A clean black and white theme
  theme(plot.title = element_text(hjust = 0.5)) # Center the plot title
```


DBSCAN Clustering
Feature 2
Feature 1
2
1
0
−1
−2
−2
−1
0
1
2
Cluster
0
1
2
3