Phân cụm DBSCAN

Le Nhat Tung

Contents

1 Giới thiệu về DBSCAN

1.1 Thuật toán phân cụm DBSCAN là gì?

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) là một thuật toán phân cụm dựa trên mật độ. Được giới thiệu bởi Martin Ester, Hans-Peter Kriegel, Jörg Sander, và Xiaowei Xu vào năm 1996, DBSCAN đã trở thành một trong những thuật toán phân cụm phổ biến nhất trong học máy không giám sát.

Khác với K-Means dựa trên khoảng cách Euclidean và yêu cầu xác định trước số cụm, DBSCAN nhóm các điểm dựa trên mật độ của chúng trong không gian dữ liệu. Điều này giúp DBSCAN có nhiều ưu điểm:

- Không cần biết trước số lượng cụm

- Phát hiện được cụm có hình dạng tùy ý, không chỉ cụm hình cầu
- Có khả năng phát hiện và loại bỏ nhiễu (outliers)
- Hiệu quả với bộ dữ liệu lớnf

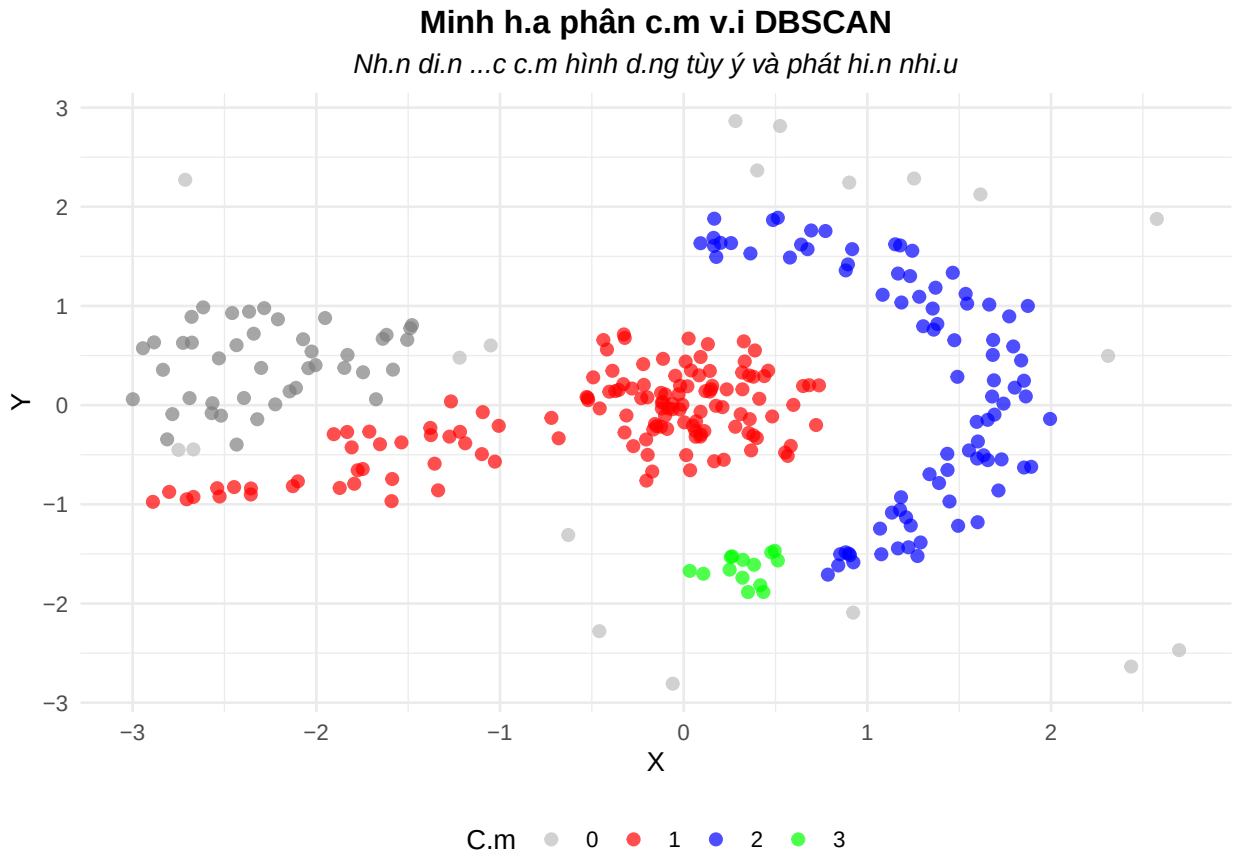

Figure 1: Minh hoa DBSCAN voi cac cum khac nhau

1.2 Các khái niệm cơ bản trong DBSCAN

DBSCAN dựa trên một số khái niệm cơ bản:

1. **Epsilon (ε)**: Bán kính vùng lân cận của một điểm
2. **MinPts**: Số lượng điểm tối thiểu cần có trong vùng lân cận ε
3. **Điểm lõi (Core point)**: Điểm có ít nhất MinPts điểm (bao gồm chính nó) trong vùng lân cận ε
4. **Điểm biên (Border point)**: Điểm nằm trong vùng lân cận ε của điểm lõi nhưng không phải là điểm lõi
5. **Điểm nhiễu (Noise point)**: Điểm không phải là điểm lõi hay điểm biên

1.3 Nguyên lý hoạt động của DBSCAN

Thuật toán DBSCAN hoạt động theo các bước sau:

1. **Khởi tạo**: Chọn một điểm ngẫu nhiên chưa được thăm
2. **Mở rộng cụm**:
    - Nếu điểm đó là điểm lõi (có ít nhất MinPts điểm trong vùng bán kính ε), tạo một cụm mới
    - Thêm tất cả các điểm trong vùng lân cận ε vào cụm
    - Tiếp tục mở rộng cụm theo cách tương tự cho các điểm lõi mới được thêm vào
3. **Tiếp tục**: Chọn một điểm chưa thăm khác và lặp lại quá trình
4. **Kết thúc**: Khi tất cả các điểm đã được thăm, các điểm không thuộc cụm nào được xác định là nhiễu

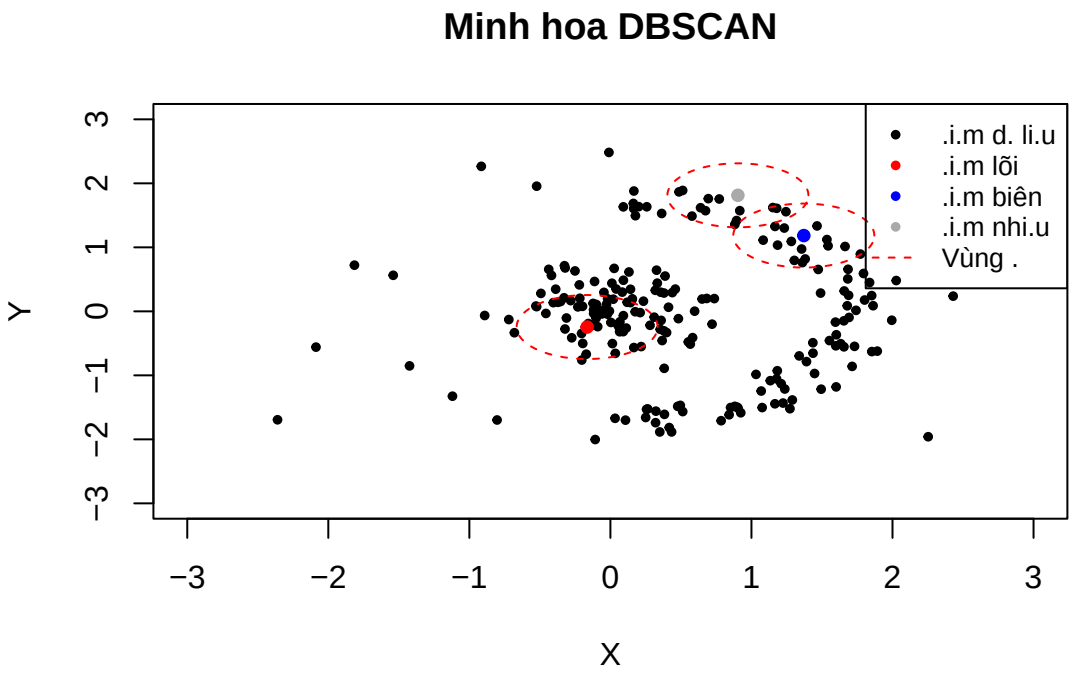

Figure 2: Minh hoa nguyen ly DBSCAN

2 Cách xác định tham số cho DBSCAN

Một trong những thách thức khi sử dụng DBSCAN là việc xác định hai tham số quan trọng: ε (epsilon) và MinPts. Không giống như K-Means, nơi chúng ta cần chọn K, DBSCAN đòi hỏi phải xác định mức độ "đông đúc" của các cụm.

2.1 Xác định MinPts

MinPts xác định số lượng điểm tối thiểu cần có trong vùng lân cận để một điểm được coi là điểm lõi.

Hướng dẫn chọn MinPts: - Một quy tắc thông thường là chọn MinPts $\geq$ D + 1, trong đó D là số chiều của dữ liệu - Với dữ liệu 2 chiều, MinPts = 4 thường là lựa chọn tốt - Với dữ liệu nhiều chiều hơn hoặc khi có nhiễu, có thể tăng MinPts lên (5-10) - MinPts càng lớn, thuật toán càng ổn định nhưng có thể bỏ qua các cụm nhỏ

2.2 Xác định Epsilon (ε)

Epsilon xác định bán kính vùng lân cận. Đây thường là tham số khó xác định nhất và có ảnh hưởng lớn đến kết quả phân cụm.

2.2.1 1. Phương pháp k-distance graph

Phương pháp k-distance graph là cách phổ biến nhất để xác định ε phù hợp:

1. Với mỗi điểm, tính khoảng cách đến điểm thứ k gần nhất (k = MinPts - 1)
2. Sắp xếp các khoảng cách này theo thứ tự tăng dần
3. Vẽ đồ thị k-distance
4. Tìm "điểm gẩy" (elbow point) - điểm mà tại đó đường cong có sự thay đổi đáng kể
5. Chọn ε tại điểm gẩy này

```
# Cài đặt và nạp các gói cần thiết
library(dbscan)  # Cho thuật toán DBSCAN
library(factoextra)  # Cho trực quan hóa
library(fpc)  # Cho đánh giá phân cụm
library(ggplot2)  # Cho vẽ biểu đồ

# Tạo dữ liệu mẫu cho DBSCAN
set.seed(123)
# Tạo một cụm hình tròn
n1 <- 100
t <- runif(n1, 0, 2*pi)
r <- runif(n1, 0, 0.8)
x1 <- r * cos(t)
y1 <- r * sin(t)
# Tạo một cụm hình bán nguyệt
n2 <- 100
t <- runif(n2, -pi/2, pi/2)
r <- runif(n2, 1.5, 2)
x2 <- r * cos(t)
y2 <- r * sin(t)
# Thêm một số điểm nhiễu
n3 <- 20
x3 <- runif(n3, -2.5, 2.5)
y3 <- runif(n3, -2.5, 2.5)
# Kết hợp dữ liệu
x <- c(x1, x2, x3)
y <- c(y1, y2, y3)
synthetic_data <- data.frame(x = x, y = y)

# Tính khoảng cách k-nearest neighbors
k <- 4
knn_dists <- kNNdist(synthetic_data, k = k)

# Sắp xếp khoảng cách và vẽ đồ thị
eps_candidates <- sort(knn_dists)
```

```r
plot(eps_candidates, type = "l",
     xlab = "Điểm dữ liệu (đã sắp xếp)",
     ylab = paste("Khoảng cách đến điểm thứ", k, "gần nhất"),
     main = "Phương pháp k-distance")

# Tìm điểm gẩy (có thể bằng thuật toán hoặc quan sát)
# Ví dụ đơn giản: tìm điểm có độ cong lớn
eps_diff <- diff(eps_candidates, differences = 2)
eps_index <- which.max(eps_diff)
eps_value <- eps_candidates[eps_index]

# Đánh dấu điểm gẩy
points(eps_index, eps_candidates[eps_index], col = "red", pch = 19)
text(eps_index, eps_candidates[eps_index],
     labels = paste("  ", round(eps_candidates[eps_index], 2)),
     pos = 4, col = "red")

# Vẽ đường thẳng tại điểm gẩy
abline(h = eps_candidates[eps_index], col = "red", lty = 2)
```

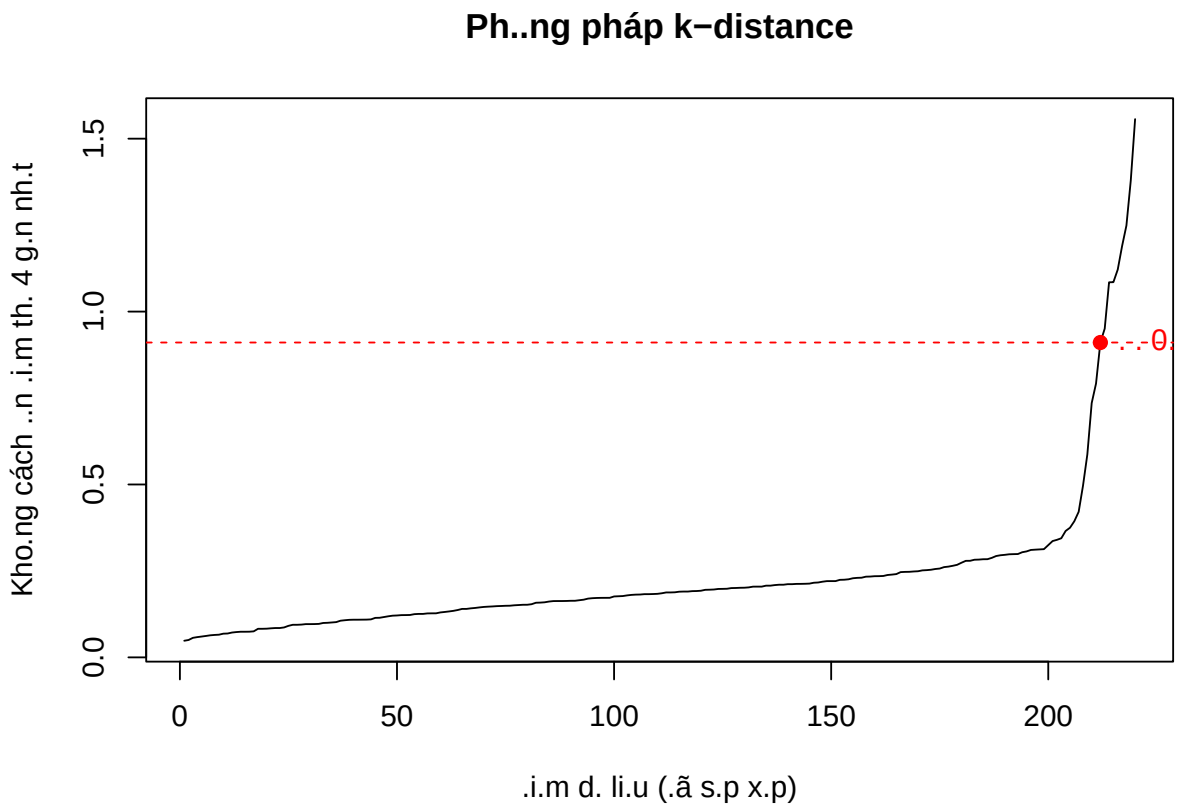

Figure 3: Do thi k-distance de xac dinh epsilon

2.3 So sánh các tham số khác nhau

Để hiểu rõ hơn về ảnh hưởng của các tham số, có thể thử nghiệm với nhiều cặp (ε, MinPts) khác nhau:

```
# Tạo lưới các tham số
eps_values <- c(0.2, 0.5, 0.8)
minPts_values <- c(3, 5)

# Tạo lưới plot
par(mfrow = c(length(minPts_values), length(eps_values)))
par(mar = c(4, 4, 2, 1))

# Áp dụng DBSCAN với các tham số khác nhau
for (minPts in minPts_values) {
  for (eps in eps_values) {
    # Áp dụng DBSCAN
    db <- dbscan::dbscan(synthetic_data, eps = eps, minPts = minPts)
    # Vẽ kết quả
    plot(synthetic_data, col = db$cluster + 1, pch = 20, cex = 0.8,
         main = paste("  =", eps, ", MinPts =", minPts),
         xlab = "X", ylab = "Y")

    # Hiển thị số lượng cụm và điểm nhiễu
    n_clusters <- max(db$cluster)
    n_noise <- sum(db$cluster == 0)
    legend("topright",
           legend = c(paste("Cụm:", n_clusters),
                      paste("Nhiễu:", n_noise)),
           cex = 0.7, bty = "n")
  }
}
```

3 Đánh giá chất lượng phân cụm DBSCAN

Đánh giá chất lượng phân cụm DBSCAN có một số khác biệt so với đánh giá K-Means, do đặc tính dựa trên mật độ và khả năng phát hiện nhiễu của DBSCAN.

3.1 Đánh giá nội tại (Internal Evaluation)

3.1.1 1. Silhouette Coefficient

Chỉ số Silhouette vẫn có thể được sử dụng cho DBSCAN, nhưng cần lưu ý rằng điểm nhiễu (có nhãn cụm = 0) sẽ không được đưa vào tính toán.

```
# Áp dụng DBSCAN
library(dbscan)
db_result <- dbscan::dbscan(synthetic_data, eps = 0.5, minPts = 5)

# Tính chỉ số Silhouette cho các điểm KHÔNG phải nhiễu
non_noise <- which(db_result$cluster > 0)
```

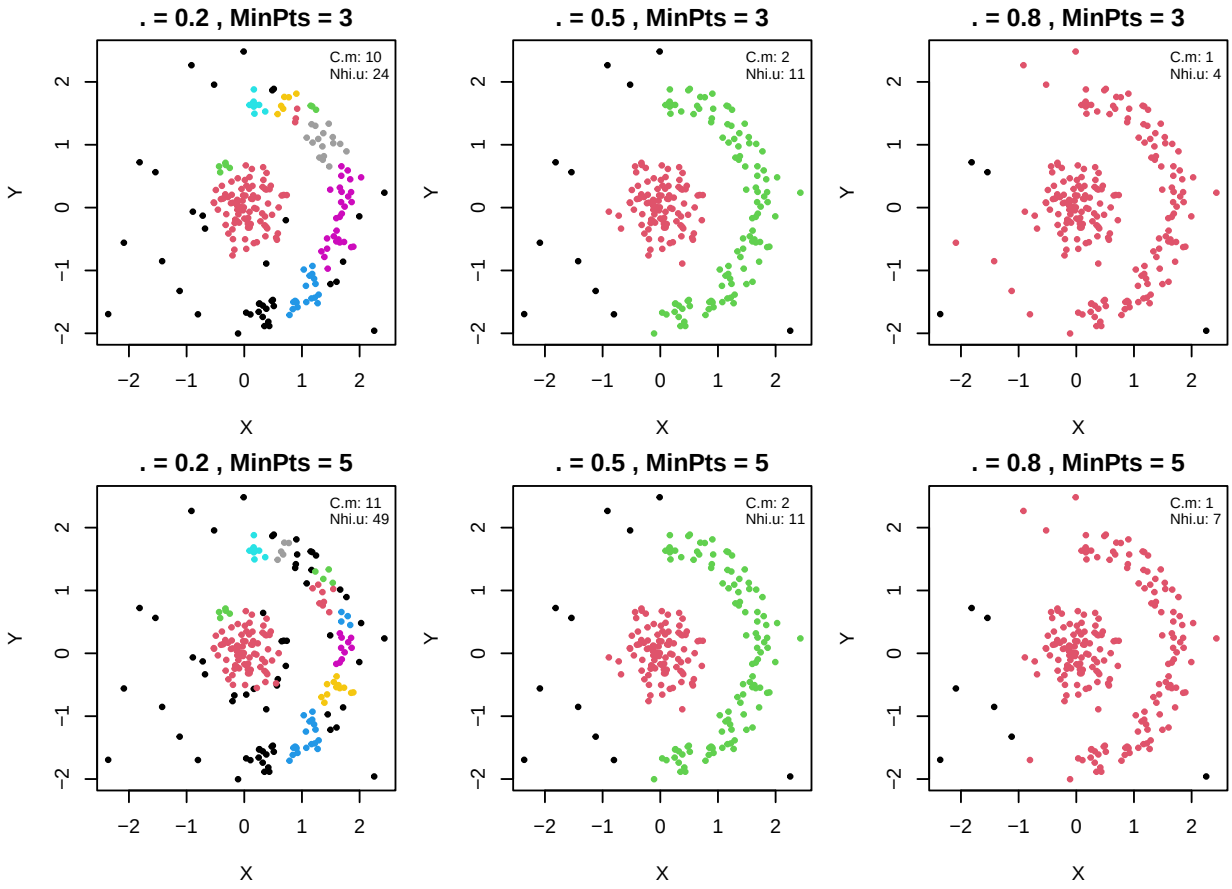

Figure 4: So sanh ket qua voi cac tham so khac nhau

```r
# Kiểm tra xem có nhiều hơn 1 cụm thực sự không
if (length(unique(db_result$cluster[non_noise])) > 1) {

  # Tính silhouette
  sil <- cluster::silhouette(db_result$cluster[non_noise],
                             dist(synthetic_data[non_noise, ]))

  # Vẽ biểu đồ Silhouette
  plot(sil, main = "Silhouette plot cho DBSCAN",
       col = as.numeric(factor(db_result$cluster[non_noise])))

  # Trung bình Silhouette
  avg_sil <- mean(sil[, 3])
  cat("Điểm Silhouette trung bình:", round(avg_sil, 3), "\n")

} else {
  cat("Không thể tính Silhouette: chỉ tìm thấy 1 cụm (không tính nhiễu)\n")
}
```

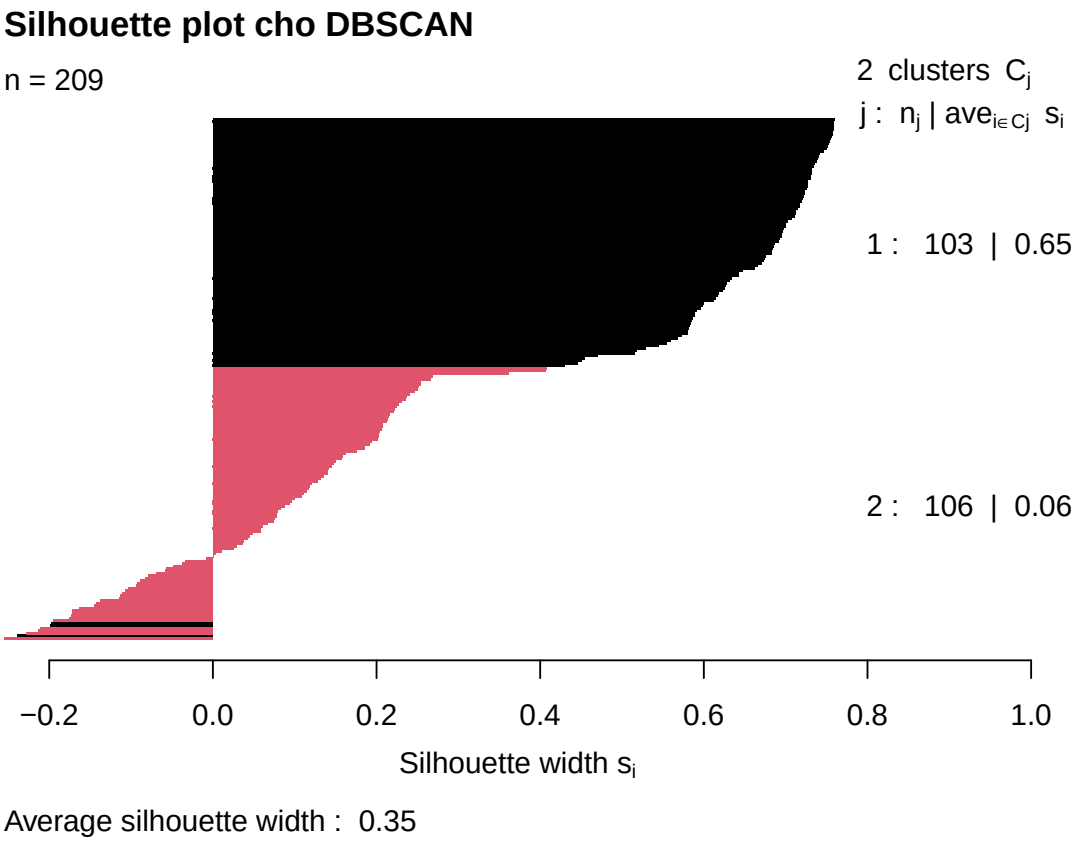

Figure 5: Chỉ số Silhouette cho DBSCAN

```
## Điểm Silhouette trung bình: 0.353
```

3.1.2 2. Tỷ lệ nhiễu (Noise Ratio)

Một chỉ số đặc biệt cho DBSCAN là tỷ lệ điểm nhiễu. Tỷ lệ này không nên quá cao hoặc quá thấp: - Tỷ lệ cao: Có thể ε quá nhỏ hoặc MinPts quá lớn - Tỷ lệ thấp: Có thể ε quá lớn, gom các cụm riêng biệt lại với nhau

```
# Tính tỷ lệ nhiễu
noise_ratio <- sum(db_result$cluster == 0) / length(db_result$cluster)
cat("Tỷ lệ nhiễu:", noise_ratio, "\n")
```

```
## Tỷ lệ nhiễu: 0.05
```

3.1.3 3. Mật độ các cụm (Cluster Density)

Một chỉ số quan trọng khác là mật độ của các cụm, được đo bằng số điểm trên đơn vị thể tích:

```
# Tính mật độ các cụm
cluster_density <- numeric(max(db_result$cluster))
for (i in 1:max(db_result$cluster)) {
  cluster_points <- synthetic_data[db_result$cluster == i, ]
  # Ước lượng thể tích bằng tích của phạm vi trên mỗi chiều
  volume <- prod(apply(cluster_points, 2, function(x) diff(range(x))))
  # Mật độ = số điểm / thể tích
  cluster_density[i] <- nrow(cluster_points) / volume
}

# Hiển thị mật độ các cụm
cat("Mật độ các cụm:", cluster_density, "\n")
```

```
## Mật độ các cụm: 39.45652 10.731
```

3.1.4 4. Davies-Bouldin Index và Calinski-Harabasz Index

Các chỉ số này cũng có thể được sử dụng cho DBSCAN, nhưng cần điều chỉnh để loại bỏ các điểm nhiễu:

```
# Tính Calinski-Harabasz Index (chỉ cho các điểm không phải nhiễu)
if (length(unique(db_result$cluster[non_noise])) > 1) {
  ch_index <- calinhara(synthetic_data[non_noise, ],
                        db_result$cluster[non_noise])
  cat("Calinski-Harabasz Index:", ch_index, "\n")
} else {
  cat("Không thể tính CH Index: chỉ tìm thấy 1 cụm (không tính nhiễu)\n")
}
```

```
## Calinski-Harabasz Index: 57.57721
```

3.2 Đánh giá ngoại tại (External Evaluation)

Nếu có nhãn thực tế, các chỉ số đánh giá ngoại tại như Adjusted Rand Index (ARI) và Normalized Mutual Information (NMI) có thể được sử dụng. Cần lưu ý rằng DBSCAN phân loại một số điểm là nhiễu (nhãn 0), nên cần xử lý phù hợp.

```r
# Giả sử chúng ta có nhãn đúng (ground truth)
# Trong thực tế, chúng ta cần dữ liệu có nhãn
# Ở đây, chúng ta tạo nhãn giả dựa trên cách tạo dữ liệu
true_labels <- c(rep(1, n1), rep(2, n2), rep(0, n3))  # 0 đại diện cho nhiễu

# Tính Adjusted Rand Index
library(fossil)
adj_rand <- adj.rand.index(true_labels, db_result$cluster)
cat("Adjusted Rand Index:", adj_rand, "\n")
```

```
## Adjusted Rand Index: 1
```

```r
# Tính Normalized Mutual Information
library(aricode)
nmi <- NMI(true_labels, db_result$cluster)
cat("Normalized Mutual Information:", nmi, "\n")
```

```
## Normalized Mutual Information: 0.8219241
```

4 Trực quan hóa kết quả DBSCAN

Trực quan hóa là một phần quan trọng để hiểu kết quả phân cụm DBSCAN, đặc biệt là hình dạng của các cụm và vị trí của các điểm nhiễu.

```r
# Trực quan hóa kết quả DBSCAN
fviz_cluster(list(data = synthetic_data, cluster = db_result$cluster),
             palette = c("black", "red", "blue"),  # màu đen cho nhiễu
             geom = "point",
             ellipse = FALSE,
             ggtheme = theme_minimal(),
             main = "Kết quả phân cụm DBSCAN")
```

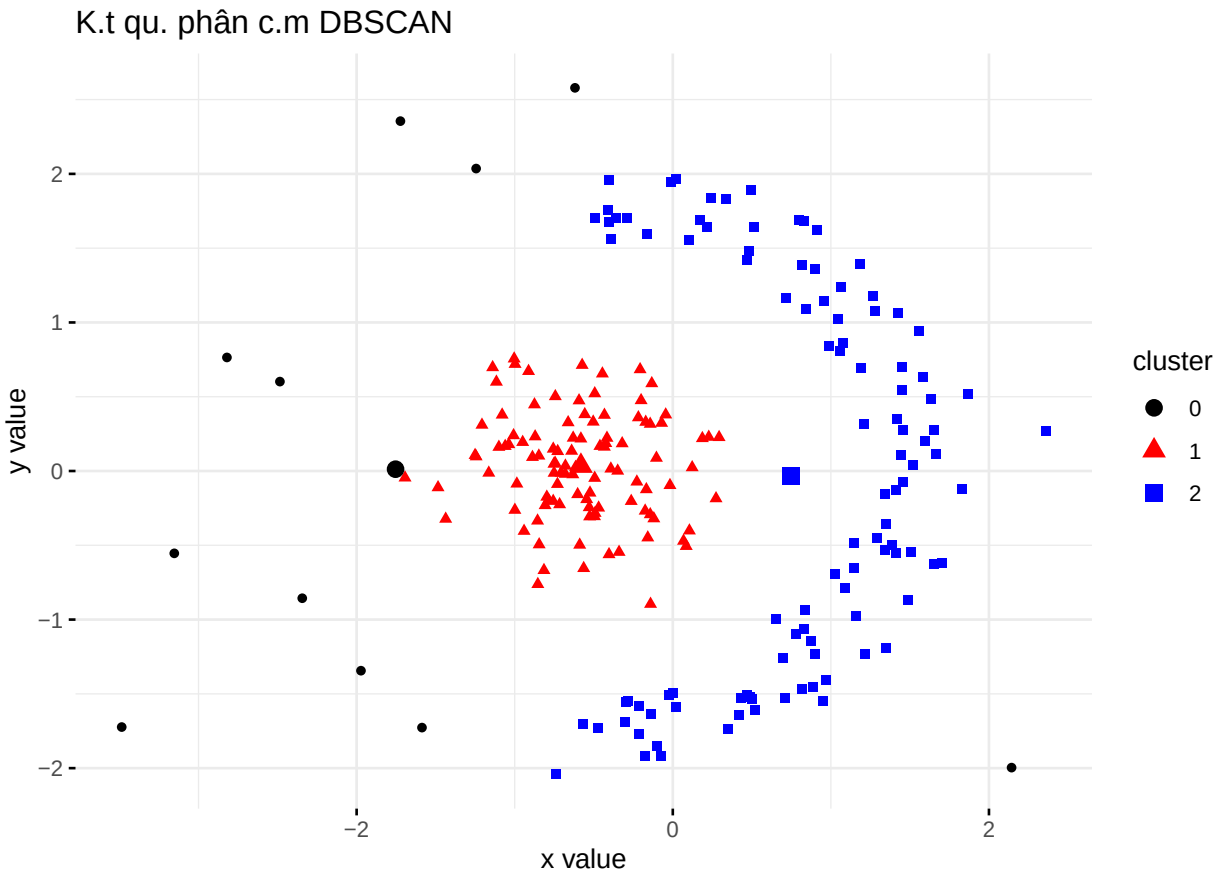

Figure 6: Ket qua phan cum DBSCAN