BIOS 611: Homework 5

Cassi Chen

2025-10-22

Task 1: Clusters and the Gap Statistic

Part 1a: Write a function to generate data

```
generate_hypercube_clusters <- function(n, k, side_length, noise_sd = 1.0) {
  data <- data.frame()

  for (i in seq_len(n)) {

    # center point
    center_point <- rep(0, n)
    center_point[i] <- side_length

    # cluster points
    cluster_points <- matrix(
      rnorm(k * n, mean = 0, sd = noise_sd),
      nrow = k, ncol = n
    )
    cluster_points <- cluster_points + rep( center_point, each = nrow(cluster_points))

    # append to df
    cluster_df <- data.frame(cluster_points)
    cluster_df$cluster <- i
    data <- rbind(data, cluster_df)
  }

  return(data)
}
```

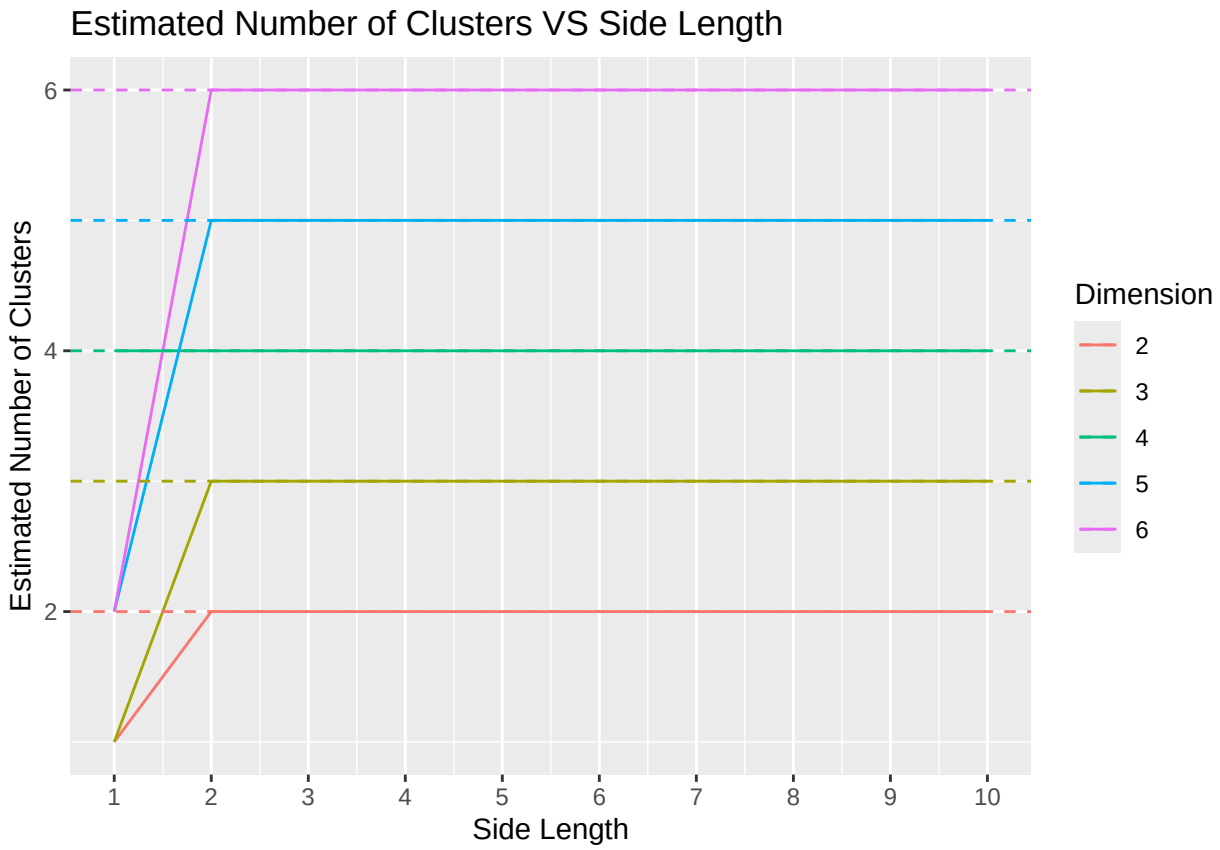
Part 1b: Run a simulation

```
# parameters
dimensions <- c(6, 5, 4, 3, 2)
side_lengths = 10:1
k <- 100
noise_sd <- 1.0
results <- data.frame()

# simulation
for (d in dimensions) {
  for (side_length in side_lengths) {
```

```
    # data
    data <- generate_hypercube_clusters(d, k, side_length, noise_sd)

    # clusgap
    clustered <- clusGap(data,
                         FUN = kmeans,
                         K.max = 10,
                         nstart = 20,
                         iter.max = 50)

    # optimize
    best_k <- maxSE(clustered$Tab[, "gap"],
                    clustered$Tab[, "SE.sim"],
                    method = "firstSEmax")

    # results
    results <- rbind(results, data.frame(
      dimension = d,
      side_length = side_length,
      clusters_guess = best_k
    ))
  }
}
```

Part 1c: Visualize the simulation results

```
# plot
plot <- ggplot(results,
               aes(x = side_length,
                   y = clusters_guess,
                   color = factor(dimension))) +
  geom_line() +
  geom_hline(aes(yintercept = dimension, color = factor(dimension)),
             linetype = "dashed") +
  scale_x_continuous(breaks = sort(unique(results$side_length))) +
  labs(
    title = "Estimated Number of Clusters VS Side Length",
    x = "Side Length",
    y = "Estimated Number of Clusters",
    color = "Dimension"
  )
plot
```




```
# save plot
ggsave(
  filename = "kmeans_clusters.png",
  plot = plot,
)
```

```
## Saving 6.5 x 4.5 in image
```

In our simulations, the gap statistic method doesn't consistently begin to reduce its estimate of the number of clusters. In fact for all 6 dimensions, at side length 2, the estimate number of clusters is equal to the true number of clusters. It remains true up to side length 10.

Task 2: Spectral Clustering on Concentric Shells

Part 2a: Write a function to generate data

```
generate_shell_clusters <- function(n_shells, k_per_shell, max_radius, noise_sd = 0.1) {

  data <- data.frame()
  radiuses <- seq(1, max_radius, length.out = n_shells)

  for (i in seq_len(n_shells)) {

    # sphere
    r <- radiuses[i]
    xyz <- matrix(rnorm(k_per_shell * 3), ncol = 3)
    xyz <- xyz / sqrt(rowSums(xyz^2))
```

```
    xyz <- xyz * r
    x <- xyz[,1]
    y <- xyz[,2]
    z <- xyz[,3]

    # df
    cluster_df <- data.frame(x = x, y = y, z = z, cluster = i)
    data <- rbind(data, cluster_df)
  }

  return(data)
}
```

Part 2b: Generate a sample dataset and create an interactive 3D scatter plot to confirm structure

```
# plot
sample <- generate_shell_clusters(n_shells = 3, k_per_shell = 50, max_radius = 4)
plot <- plot_ly(sample, x = ~x, y = ~y, z = ~z, color = ~factor(cluster), type = "scatter3d", mode = "ma
plot
```

```
saveWidget(plot, "3D_scatter.html", selfcontained = TRUE)
```

Part 2c: Implement the spectral clustering algorithm.

```
wrapper <- function(x, k, d_threshold) {
  n <- nrow(x)

  # build a similarity graph
  dist_matrix <- as.matrix(dist(x, method = "euclidean"))
  A <- (dist_matrix < d_threshold) * 1.0
  diag(A) <- 0

  # compute the normalized laplacian matrix
  D <- diag(rowSums(A))
  D_d <- diag(D)
  D_d[D_d == 0] <- 1e-10
  D_inv_sqrt <- diag(1 / sqrt(D_d))
  L <- D - A
  L_sym <- D_inv_sqrt %*% L %*% D_inv_sqrt

  # eigen-decomposition
  eigen_result <- eigen(L_sym, symmetric = TRUE)
  eigenvectors <- eigen_result$vectors[, (n - k + 1):n, drop = FALSE]
  eigenvalues <- eigen_result$values

  # cluster the eigenvectors
  kmeans_cluster <- kmeans(eigenvectors, centers = k, nstart = 20, iter.max = 50)

  # clusters <- kmeans_result$cluster
  return(kmeans_cluster)
}
```

Part 2d: Run a simulation to find the point at which the shells become too close for the algorithm to distinguish

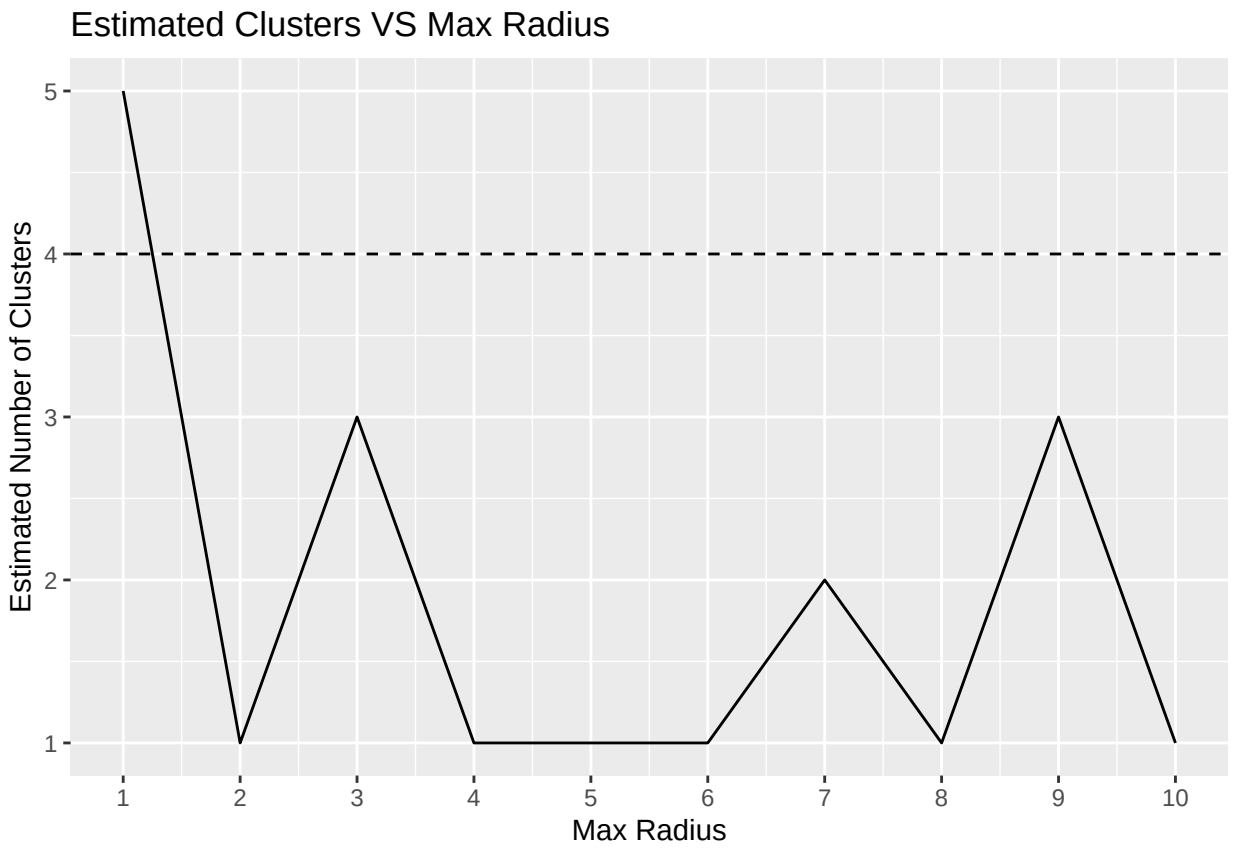
```
# parameters
max_radii <- seq(10, 0.5, by = -1)
d_threshold <- 1
n_shells <- 4
k_per_shell <- 100
noise_sd <- 0.1

# results
results <- data.frame()

for (max_radius in max_radii) {

  # data
  data <- generate_shell_clusters(n_shells, k_per_shell, max_radius, noise_sd)[, c("x", "y", "z")]

  # Apply Gap Statistic with spectral clustering

  clustered <- clusGap(data,
                       FUN = function(x, k) wrapper(x, k, d_threshold),
                       K.max = 6)

  # Find optimal number of clusters
  best_k <- maxSE(clustered$Tab[, "gap"],
                  clustered$Tab[, "SE.sim"],
                  method = "firstSEmax")

  # Store results
  results <- rbind(results, data.frame(
    max_radius = max_radius,
    clusters_guess = best_k,
    clusters_true = n_shells
  ))
}
```

Part 2e: Visualize and interpret your results

```
plot <- ggplot(results, aes(x = max_radius, y = clusters_guess)) +
  geom_line() +
  geom_hline(aes(yintercept = clusters_true), linetype = "dashed") +
  scale_x_continuous(breaks = sort(unique(results$max_radius))) +
  labs(title = "Estimated Clusters VS Max Radius",
       x = "Max Radius",
       y = "Estimated Number of Clusters")

# Save the plot
ggsave(
  filename = "spectral_clusters.png",
  plot = plot
)
```

```
## Saving 6.5 x 4.5 in image
```

```
plot
```



The algorithm increases estimated number of clusters as max radius decreases. We expect as max radius decreases the estimated number of clusters decreases. However, in our case, this is not true. We can assume that the algorithm may be sensitive to other variables, like how close the shells are to each other. Using a different threshold may change these results. Smaller threshold (0.8 etc) makes the algorithm more sensitive since the shells are closer to each other. Thus, we can try to make the threshold larger (1.2) for better results.