Ejercicios Propuestos

Juan Rubio Cobeta

October 16, 2025

Ejercicio 1:

Realiza una descripción razonada de la aplicación práctica del teorema.

Solución:

El teorema fundamental del Escalamiento Multidimensional (MDS) Clasico es de gran importancia práctica porque ofrece una solución algebraica directa a un problema geométrico: la creación de un "mapa" o configuración espacial a partir, únicamente, de una matriz de distancias.

En esencia, permite a un analista convertir una tabla de disimilaridades (por ejemplo, entre productos, ciudades o especies) en una visualización que revela la estructura latente en los datos. Su utilidad se puede resumir en tres funciones clave:

1. **Transformación de Distancias a Productos Escalares:**
   El teorema proporciona el método para convertir la información de una matriz de distancias D en una matriz de productos escalares centrada B. Esto se logra mediante la fórmula de doble centrado:

   $$B = HAH, \quad \text{donde } a_{rs} = -d_{rs}^2/2$$

   Este paso es fundamental, ya que las coordenadas de los puntos se pueden derivar algebraicamente de los productos escalares, pero no directamente de las distancias. El teorema nos da el puente necesario para hacerlo.

2. **Herramienta de Diagnóstico:**
   El teorema establece que las distancias son perfectamente euclideas si y solo si la matriz B es semidefinida positiva. En la práctica, esto nos permite verificar la naturaleza de nuestros datos: si al calcular los valores propios de B todos resultan ser no negativos, sabemos que es posible una representación perfecta. Si aparecen valores propios negativos, el mapa será necesariamente una aproximación.

3. **Método Constructivo para Obtener Coordenadas:**
   Finalmente, el teorema es constructivo: nos da la receta para encontrar las coordenadas del mapa. Mediante la descomposición espectral de la matriz B, la configuración de puntos X se obtiene directamente:

   $$X = V\Lambda^{1/2}$$

   Aquí, V es la matriz de vectores propios y Λ la matriz diagonal de valores propios. Además, la magnitud de estos valores propios (λ_i) indica la importancia de cada dimensión, ayudando a decidir cuántas dimensiones usar para una visualización efectiva.

Ejercicio 2:

La siguiente tabla recoge las distancias entre las siguientes 10 ciudades Europeas: Londres, Estocolmo, Lisboa, Madrid, París, Amsterdam, Berlin, Praga, Roma y Dublin. Escribe el código necesario usando R, para obtener los vectores propios normalizados ($v_i' v_i = \lambda_i$) para la matriz B de productos escalares centrados asociada.

Solución:

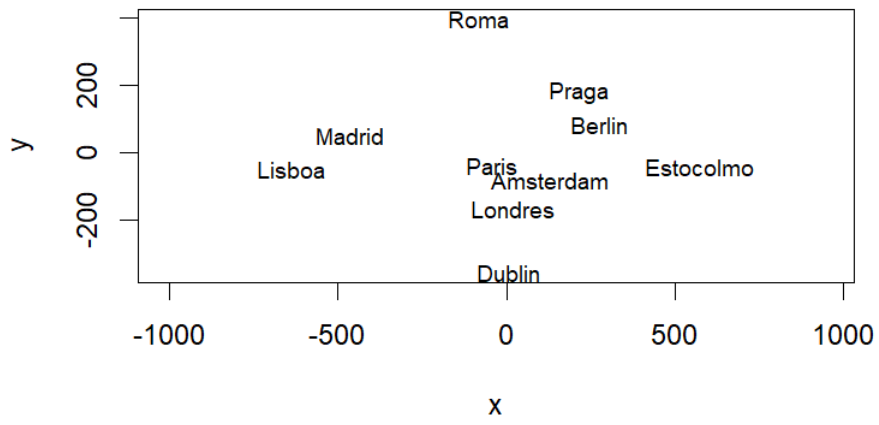
```
# 1. Definir los datos de entrada
ciudades <- c("Londres", "Estocolmo", "Lisboa", "Madrid", "Paris",
              "Amsterdam", "Berlin", "Praga", "Roma", "Dublin")
distancias <- as.matrix(
  data.frame(
    c1 = c(0, 569, 667, 530, 141, 140, 357, 396, 570, 190),
    c2 = c(569, 0, 1212, 1043, 617, 446, 325, 423, 787, 648),
    c3 = c(667, 1212, 0, 201, 596, 768, 923, 882, 714, 714),
    c4 = c(530, 1043, 201, 0, 431, 608, 740, 690, 516, 622),
    c5 = c(141, 617, 596, 431, 0, 177, 340, 337, 436, 320),
    c6 = c(140, 446, 768, 608, 177, 0, 218, 272, 519, 302),
    c7 = c(357, 325, 923, 740, 340, 218, 0, 114, 472, 514),
    c8 = c(396, 423, 882, 690, 337, 272, 114, 0, 364, 573),
    c9 = c(570, 787, 714, 516, 436, 519, 472, 364, 0, 755),
    c10 = c(190, 648, 714, 622, 320, 302, 514, 573, 755, 0)
  )
)

# 2. Aplicar el algoritmo de MDS Clasico
n <- nrow(distancias)
A <- -0.5 * distancias^2
I <- diag(n) # Matriz identidad
ones <- matrix(1, nrow = n, ncol = n)
H <- I - (1/n) * ones
B <- H %*% A %*% H


# 3. Obtener los valores y vectores propios de B
descomposicion_B <- eigen(B)
valores_propios <- descomposicion_B$values
vectores_propios_norm1 <- descomposicion_B$vectors

# 4. Normalizar los vectores propios segun la condicion v_i' * v_i = lambda_i
Lambda_sqrt <- diag(sqrt(pmax(valores_propios, 0)))
vectores_propios_normalizados <- vectores_propios_norm1 %*% Lambda_sqrt


# 5. Mostrar el resultado final
colnames(vectores_propios_normalizados) <- paste0("v", 1:n)
rownames(vectores_propios_normalizados) <- ciudades
cat("--- Matriz B de productos escalares centrados ---\n")
print(round(B, 2))
cat("\n--- Vectores propios normalizados (v_i' * v_i = lambda_i) para la matriz B ---\n")
print(round(vectores_propios_normalizados, 2))
```

Ejercicio 3:

Usando R, construye un fichero de sintaxis mediante el cual se efectúe MDS clásico sobre los datos de las ciudades de la Tabla 1, determinando la dimensión adecuada para la representación. Compara los resultados con los obtenidos en el Ejemplo anterior.

Solución:

```
  dimnames(distancias) <- list(ciudades, ciudades)

# --- PASO 1: Realizar el MDS Clasico ---.
mds_resultado <- cmdscale(distancias, k = length(ciudades) - 1, eig = TRUE)

# --- PASO 2: Determinar la dimension adecuada ---
valores_propios <- mds_resultado$eig
# Imprimimos los valores propios para inspeccionarlos
# Vemos que los dos primeros son mucho mas grandes que el resto.
cat("---- Valores Propios (Eigenvalues) ----\n")
print(valores_propios)

varianza_explicada <- valores_propios[valores_propios > 0] / sum(valores_propios[valores_prop
varianza_acumulada <- cumsum(varianza_explicada)
bondad_ajuste <- data.frame(
  Dimension = 1:length(varianza_explicada),
  Varianza_Explicada = round(varianza_explicada * 100, 2),
  Varianza_Acumulada = round(varianza_acumulada * 100, 2)
)
print(bondad_ajuste)
# Observacion: Con 2 dimensiones ya explicamos mas del 90% de la varianza,
# lo que indica que una representacion en 2D es muy adecuada.
barplot(valores_propios,
        main = "Grafico de Sedimentacion (Scree Plot)",
        xlab = "Dimension (Componente)",
        ylab = "Valor Propio (Eigenvalue)",
        names.arg = 1:length(valores_propios),
        col = "steelblue")

# --- PASO 3: Representacion grafica en la dimension elegida (k=2) ---
coordenadas <- mds_resultado$points[, 1:2]
x <- coordenadas[, 1]
y <- coordenadas[, 2]
y <- -y
plot(x, y,
     type = "n", # No dibuja los puntos, solo crea el lienzo
     main = "Mapa MDS de Ciudades Europeas (k=2)",
     xlab = "Coordenada 1",
     ylab = "Coordenada 2",
     asp = 1)
text(x, y, labels = ciudades, cex = 0.8)
```



Respecto a la comparación, ambos gráficos demuestran con éxito la principal fortaleza del MDS clásico: transforman una matriz de distancias en un "mapa" bidimensional que es notablemente similar a un mapa geográfico real.

El primer gráfico, basado en el conjunto de datos `eurodist`, ofrece una visión más amplia de Europa, con una orientación que se alinea aproximadamente con un mapa convencional (por ejemplo, Estocolmo en el noreste y Lisboa en el suroeste).

El segundo gráfico tiene el mismo éxito al preservar las posiciones relativas (por ejemplo, agrupando correctamente a Madrid con Lisboa y a París con Londres). Sin embargo, su orientación general es diferente, pareciendo estar rotado en comparación con el primer gráfico (por ejemplo, Roma se sitúa en la parte superior). Esta diferencia es totalmente esperada, ya que la solución que proporciona el MDS es única solo hasta isometrías (rotaciones y reflexiones).

Ejercicio 4:

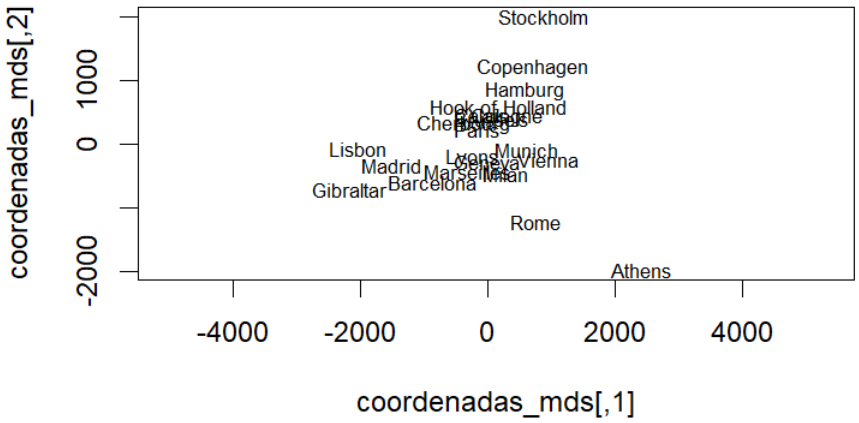
Sobre los datos de `eurodist`, construir un fichero de sintaxis en R que permita realizar MDS clásico después de transformar las distancias mediante el procedimiento de la constante aditiva. Construye la representación en dos dimensiones y compárala con la anterior. ¿Cuántas dimensiones resultan necesarias para explicar los datos?

Solución:

Creo que hay un atributo en alguna función relacionada con MDS que ya hace la modificación. Yo he intentado seguir los pasos de la teoría:

```
library(stats)
datos <- eurodist
nombres_ciudades <- attr(datos, "Labels")

Delta <- as.matrix(datos)
n <- nrow(Delta)

Delta_sq <- Delta^2

A_original <- -0.5 * Delta_sq

I <- diag(n)
ones <- matrix(1, nrow = n, ncol = n)
H <- I - (1/n) * ones

B_original <- H %*% A_original %*% H

eigen_B_original <- eigen(B_original)
lambda_min <- min(eigen_B_original$values)

c_aditiva <- 0
if (lambda_min < 0) {
  c_aditiva <- -2 * lambda_min
}

Delta_sq_corregida <- Delta_sq
Delta_sq_corregida[upper.tri(Delta_sq_corregida, diag = FALSE)]
<- Delta_sq_corregida[upper.tri(Delta_sq_corregida, diag = FALSE)] + c_aditiva
Delta_sq_corregida[lower.tri(Delta_sq_corregida, diag = FALSE)]
<- Delta_sq_corregida[lower.tri(Delta_sq_corregida, diag = FALSE)] + c_aditiva
diag(Delta_sq_corregida) <- 0

A_corregida <- -0.5 * Delta_sq_corregida
B_corregida <- H %*% A_corregida %*% H

eigen_B_corregida <- eigen(B_corregida)

num_dim_plot <- 2
coordenadas_mds <- eigen_B_corregida$vectors[, 1:num_dim_plot] %*%
  diag(sqrt(pmax(0, eigen_B_corregida$values[1:num_dim_plot])))
```

```
rownames(coordenadas_mds) <- nombres_ciudades
coordenadas_mds[, 2] <- -coordenadas_mds[, 2]

plot(coordenadas_mds, type = "n", asp = 1)
text(coordenadas_mds, labels = nombres_ciudades, cex = 0.7)
```



Como podemos ver en el gráfico, dos dimensiones serían necesarias para explicar los datos. Comparándola con la anterior de los apuntes, se ve claramente la idea del ajuste: Aunque se pierde claridad ya que todas las ciudades están mas juntas, se gana seguridad en el sentido de que no tenemos ciudades como Atenas que se salen de la pantalla en el gráfico original. En el modificado, está ajustado para que sea más preciso.