Exercício 7.
Nome: Lucas Kou Kinoshita
RM: 2019021557
data: 26/04/2025

Implementação:

A implementação do k-médias e do treinamento com RBF seguiram o algoritmo elaborado em sala de aula e nas notas de aula.

```r
kmeans <- function(xin, k, maxit){
  N <- dim(xin)[1]
  n <- dim(xin)[2]
  iseq <- sample(N)
  ci <- as.matrix(xin[iseq[1:k], ])
  c_mais_prox <- matrix(nrow = 1, ncol = N)

  for(j in 1:maxit) {
    for(i in 1:N){
      xaug_mat <- matrix(xin[i,], nrow = k, ncol = n, byrow = TRUE)
      dxaug_ci <- (ci - xaug_mat)^2
      di_vec <- rowSums(dxaug_ci)
      c_mais_prox[i] <- which.min(di_vec)
    }

    for(l in 1:k){
      ickvet <- which(c_mais_prox == l)
      if (length(ickvet) > 0) {
        ci[l,] <- colMeans(xin[ickvet,])
      }
    }
  }

  return(list(center = ci, cluster = c_mais_prox))
}
```

Figura 1: implementação k-médias

```r
pdfnvar <- function(x, m, K) {
  n <- length(x)
  coef <- 1 / sqrt((2*pi)^n * det(K))
  expoente <- exp(-0.5 * t(x-m) %*% solve(K) %*% (x-m))
  return(coef * expoente)
}
```

Figura 2: implementação da gaussiana

```r
RBF <- function(xin, yin, p, r){
  radialnvar <- function(x, m, invK) exp(-0.5 * (t(x-m) %*% invK %*% (x-m)))

  N <- dim(xin)[1]
  n <- dim(xin)[2]

  xin <- as.matrix(xin)
  yin <- as.matrix(yin)

  xclust <- kmeans(xin, p, 10)

  m <- xclust$center
  covi <- r * diag(n)
  inv_covi <- (1/r) * diag(n)

  H <- matrix(nrow = N, ncol = p)
  for(j in 1:N){
    for(i in 1:p){
      mi <- m[i,]
      H[j,i] <- radialnvar(xin[j,], mi, inv_covi)
    }
  }

  Haug <- cbind(1, H)
  W <- pseudoinverse(Haug) %*% yin

  return(list(m, covi, r, W))
}
```

Figura 3: implementação do treinamento com RBF

```r
YRBF <- function(xin, modRBF){
  radialnvar <- function(x, m, invK) exp(-0.5 * (t(x-m) %*% invK %*% (x-m)))

  N <- dim(xin)[1]
  n <- dim(xin)[2]

  m <- modRBF[[1]]
  covi <- modRBF[[2]]
  inv_covi <- (1/modRBF[[3]]) * diag(n)
  W <- modRBF[[4]]
  p <- dim(m)[1]

  xin <- as.matrix(xin)
  H <- matrix(nrow = N, ncol = p)

  for(j in 1:N){
    for(i in 1:p){
      mi <- m[i,]
      H[j,i] <- radialnvar(xin[j,], mi, inv_covi)
    }
  }

  Haug <- cbind(1, H)
  Yhat <- Haug %*% W

  return(Yhat)
}
```

Figura 4: implementação do cálculo de resultados do RBF

Parte 1.

Assim como sugerido no enunciado do exercício, os conjuntos sintéticos de dados foram gerados com auxílio da biblioteca *mlbench.* Em seguida, treinadas pelo método *RBF(),* que utiliza o k-médias para determinação de (*p*) *clusters* com fator de espalhamento (*r*) nas gaussianas.

```
# Normals
normals <- mlbench.2dnormals(200)
xin_normals <- as.matrix(normals$x)
yin_normals <- as.numeric(normals$classes)

modelo_normals <- RBF(xin_normals, diag(1,2)[yin_normals,], p, r)
plot_classification(xin_normals, yin_normals, modelo_normals, "2D Normals")

# XOR
xor <- mlbench.xor(100)
xin_xor <- as.matrix(xor$x)
yin_xor <- as.numeric(xor$classes)

modelo_xor <- RBF(xin_xor, diag(1,2)[yin_xor,], p, r)
plot_classification(xin_xor, yin_xor, modelo_xor, "XOR")

# Circle
circle <- mlbench.circle(100)
xin_circle <- as.matrix(circle$x)
yin_circle <- as.numeric(circle$classes)

modelo_circle <- RBF(xin_circle, diag(1,2)[yin_circle,], p, r)
plot_classification(xin_circle, yin_circle, modelo_circle, "Circle")

# Spirals
spiral <- mlbench.spirals(100, cycles=1, sd=0.05)
xin_spiral <- as.matrix(spiral$x)
yin_spiral <- as.numeric(spiral$classes)

modelo_spiral <- RBF(xin_spiral, diag(1,2)[yin_spiral,], p, r)
plot_classification(xin_spiral, yin_spiral, modelo_spiral, "Spirals")
```

Figura 5: Geração de dados, treinamento e visualização

Os métodos *plot_classification()* e *criar_grid()* foram elaborados em um arquivo utils.R com o objetivo de auxiliar a visualização das superfícies de separação resultantes dos treinamentos. De forma:

```
plot_classification <- function(xin, yin, modelo, titulo) {
  grid <- criar_grid(xin)
  pred_grid <- YRBF(grid, modelo)
  pred_grid <- apply(pred_grid, 1, which.max) # Classe com maior ativação

  plot(xin[,1], xin[,2], col = yin, pch = 19, main = titulo, xlab = "", ylab = "")
  points(grid[,1], grid[,2], col = pred_grid, pch = ".", cex = 0.5)
}
```

Figura 6: implementação da visualização de superfícies

```r
criar_grid <- function(x, n = 200) {
  x1_range <- seq(min(x[,1]) - 0.5, max(x[,1]) + 0.5, length.out = n)
  x2_range <- seq(min(x[,2]) - 0.5, max(x[,2]) + 0.5, length.out = n)
  grid <- expand.grid(x1 = x1_range, x2 = x2_range)
  return(as.matrix(grid))
}
```

Figura 7: geração dos grids para visualização

Por fim, podemos visualizar os seguintes resultados para diferentes números de centros:

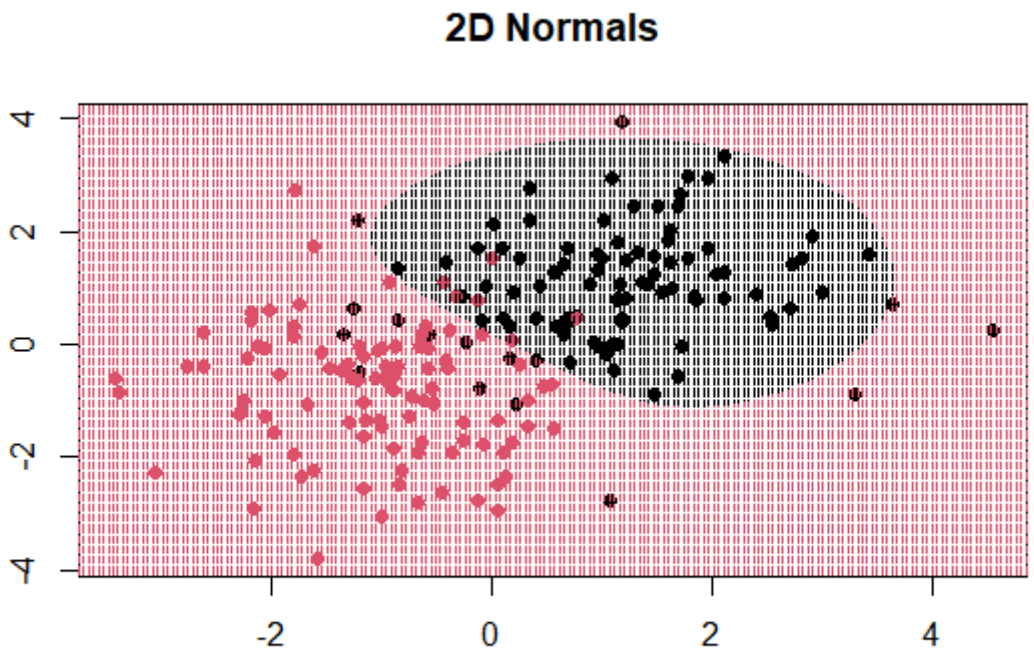

Figura 8: Superfície de separação para o problema das normais, k = 2

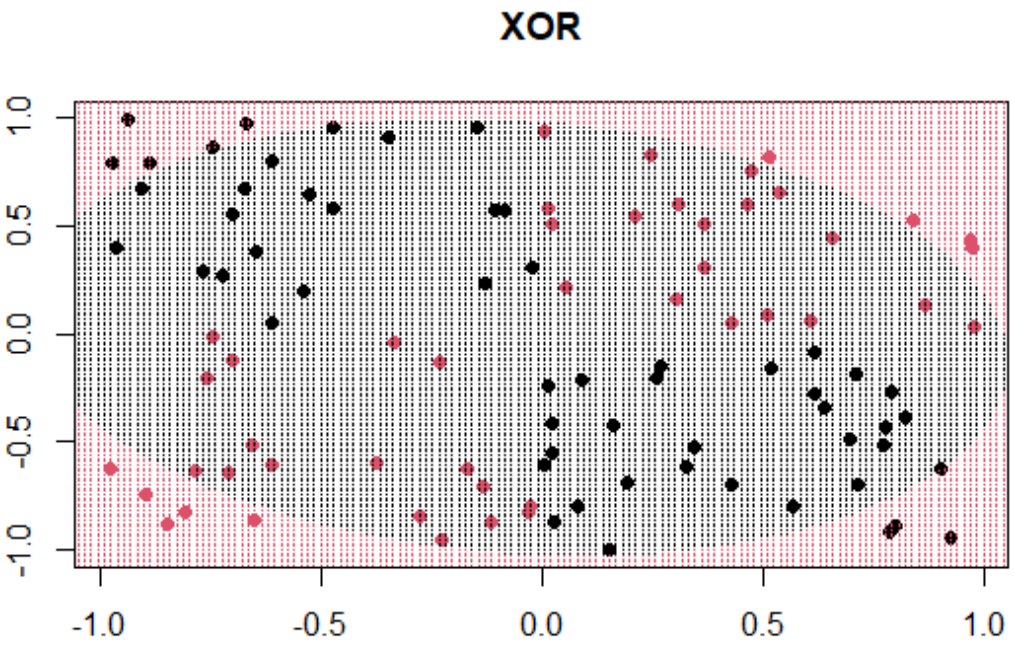

Figura 9: Superfície de separação para o problema XOR, k = 2

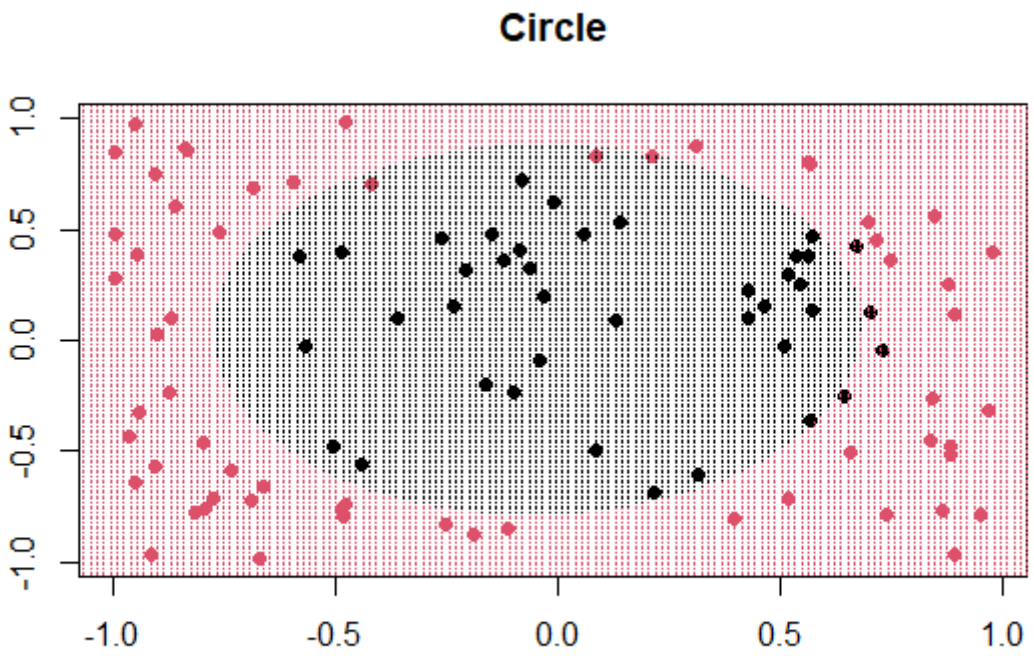

Figura 10: Superfície de separação para o problema do círculo, k = 2

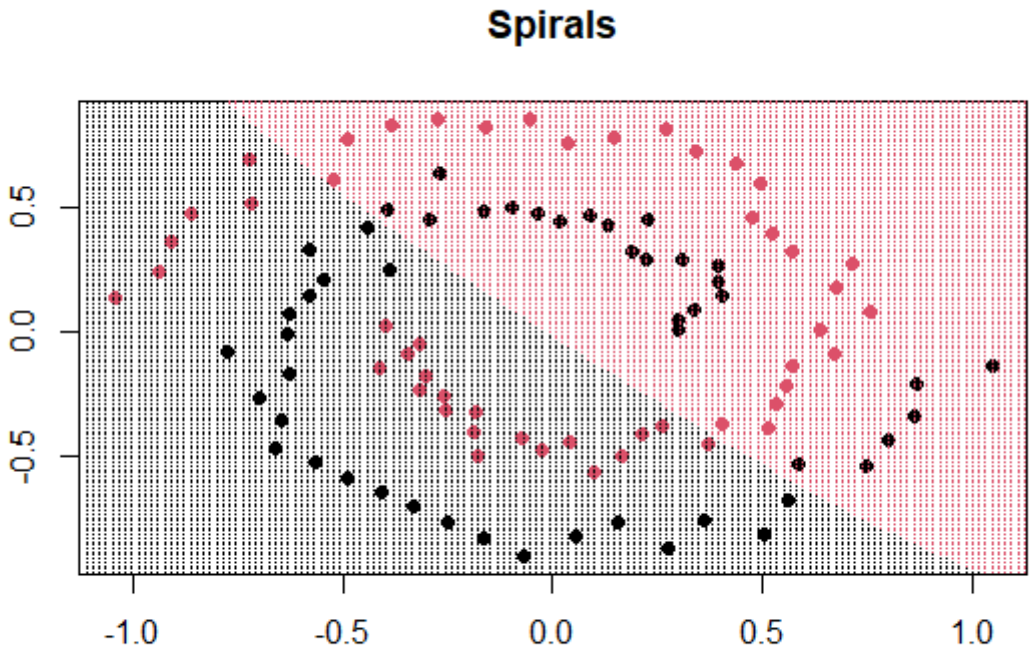

Figura 11: Superfície de separação para o problema das espirais, k = 2

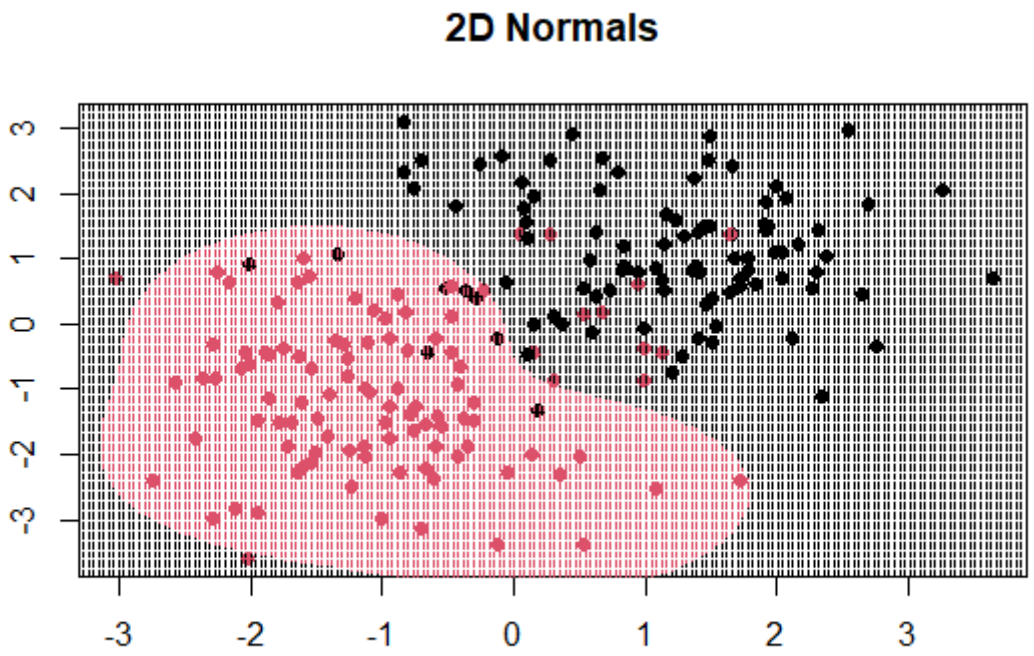

Figura 12: Superfície de separação para o problema das normais, k = 6

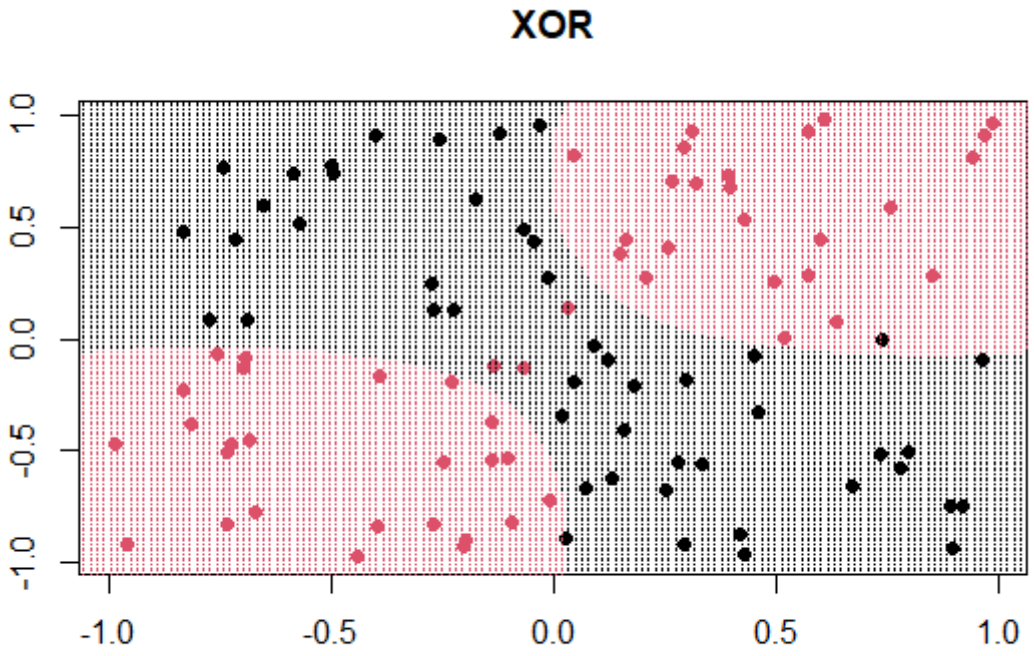

Figura 13: Superfície de separação para o problema XOR, k = 6

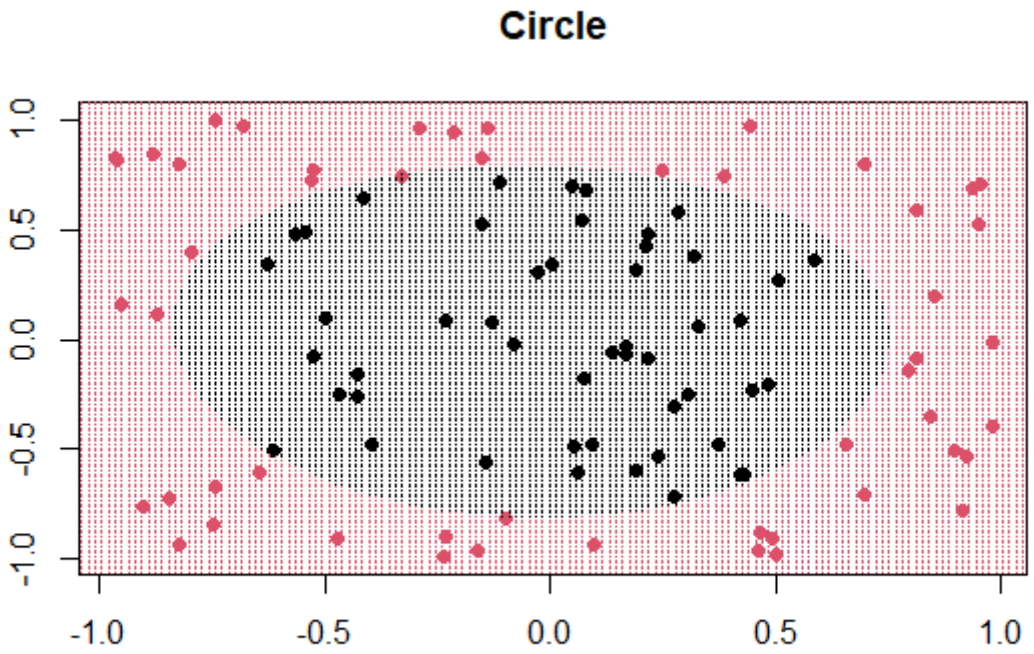

Figura 14: Superfície de separação para o problema do círculo, k = 6

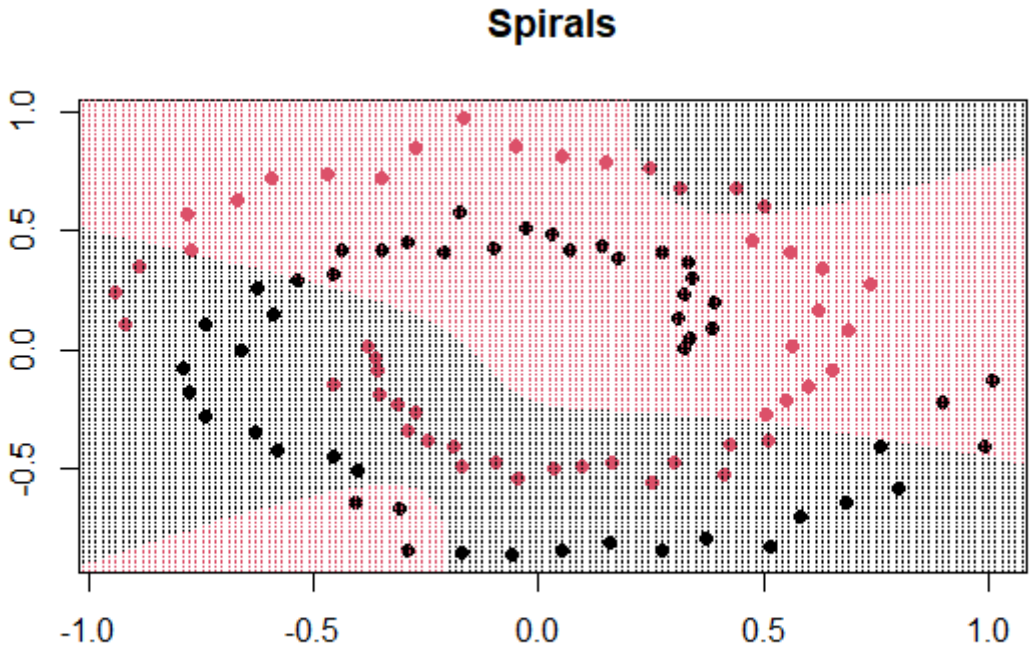

Figura 15: Superfície de separação para o problema das espirais, k = 6

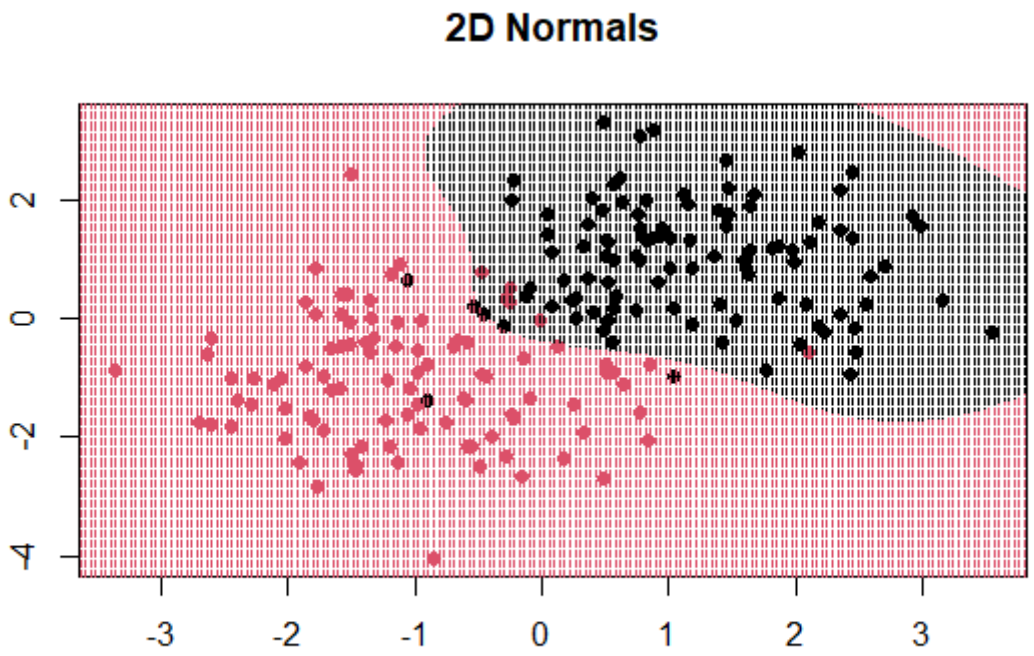

Figura 16: Superfície de separação para o problema das normais, k = 12

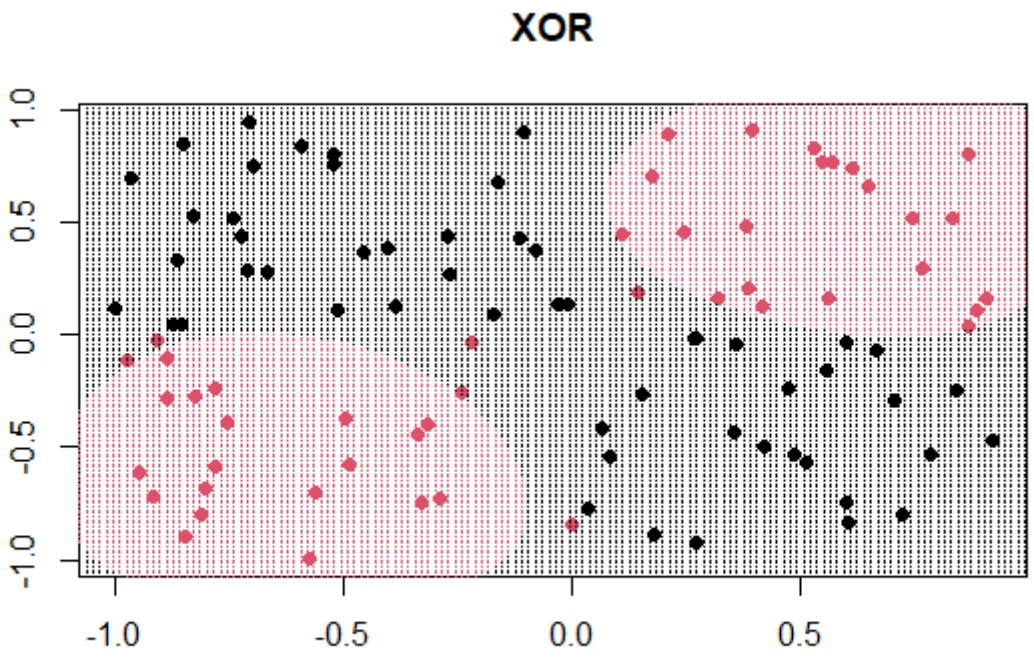

Figura 17: Superfície de separação para o problema XOR, k = 12

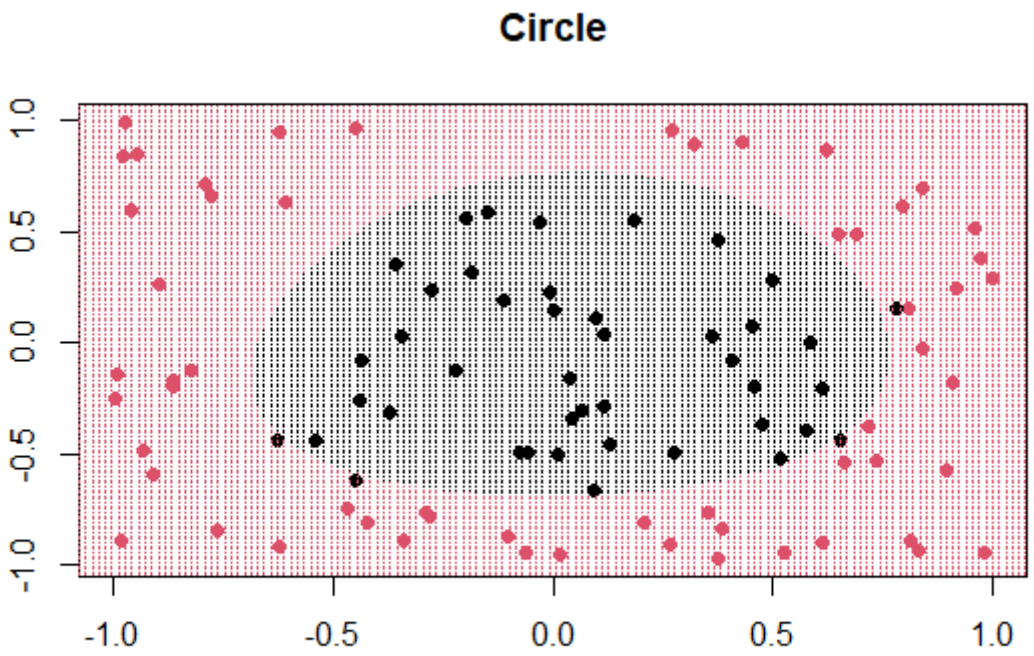

Figura 18: Superfície de separação para o problema do círculo, k = 6

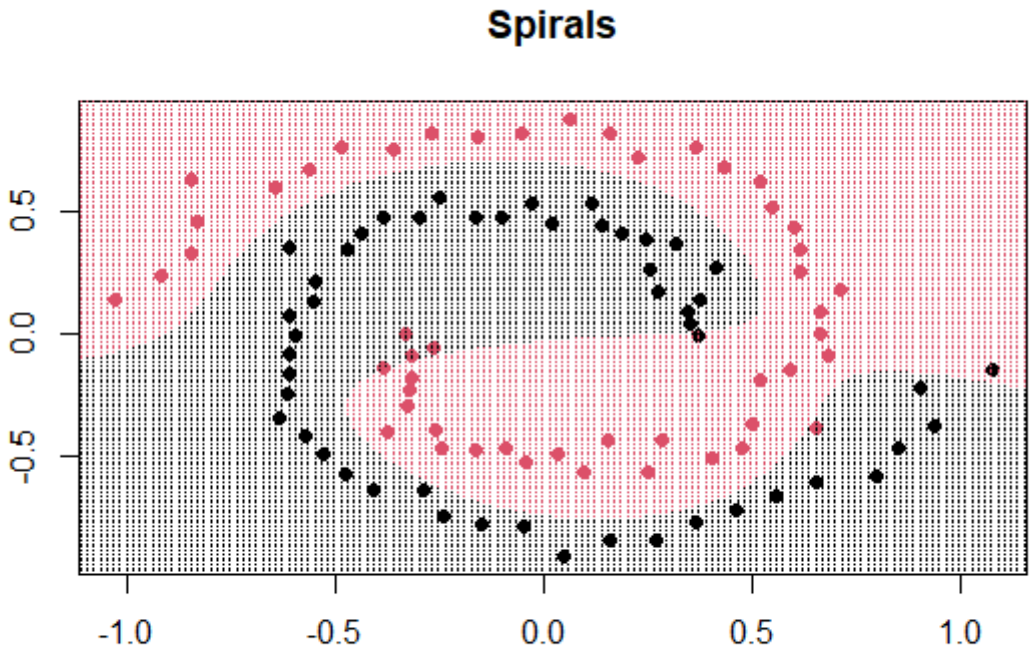

Figura 19: Superfície de separação para o problema das espirais, k = 12

Parte 2.

Partindo dos mesmos métodos implementados anteriormente, porém com os dados sintéticos gerados para o problema de aproximação da função *sinc*.

```
# sinc
x <- runif(100, -15, 15)
y <- -sin(x)/x + rnorm(100, 0 , 0.05)
xin <- as.matrix(x)
yin <- as.matrix(y)
p <- 2
r <- 1
modeloRBF <- RBF(xin, yin, p, r)
yhat <- YRBF(xin, modeloRBF)
plot(x, y, col = 'blue', main = 'sinc(x)', xlab = 'x', ylab = 'y')
points(x, yhat, col = 'red')
legend("topright", legend = c("Original data", "Aproximação"),
       col = c("blue", "red"), pch = c(1,1))
```

Figura 20: Implementação para o problema de aproximação

O desempenho de cada modelo é testado por meio de um segundo conjunto de testes, assim como sugerido no enunciado, à partir do MSE

```
xtest <- runif(50, -15, 15)
ytest <- -sin(xtest)/xtest + rnorm(50, 0, 0.05)
xtest <- as.matrix(xtest)
yhat <- YRBF(xtest, modeloRBF)

mse <- mean((ytest - yhat)^2)
print(mse)
```

Figura 21: Implementação do conjunto de teste e do MSE

Foram obtidos os seguintes resultados:

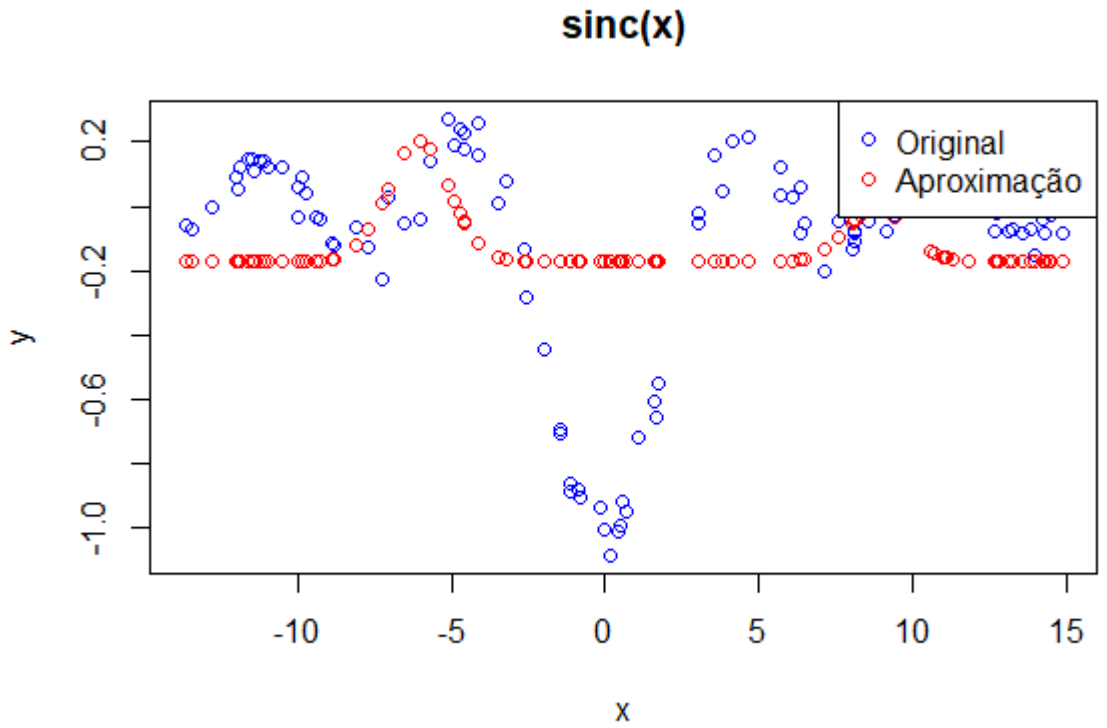

Figura 22: Aproximação com k = 2 e r = 1. MSE = 0.0866

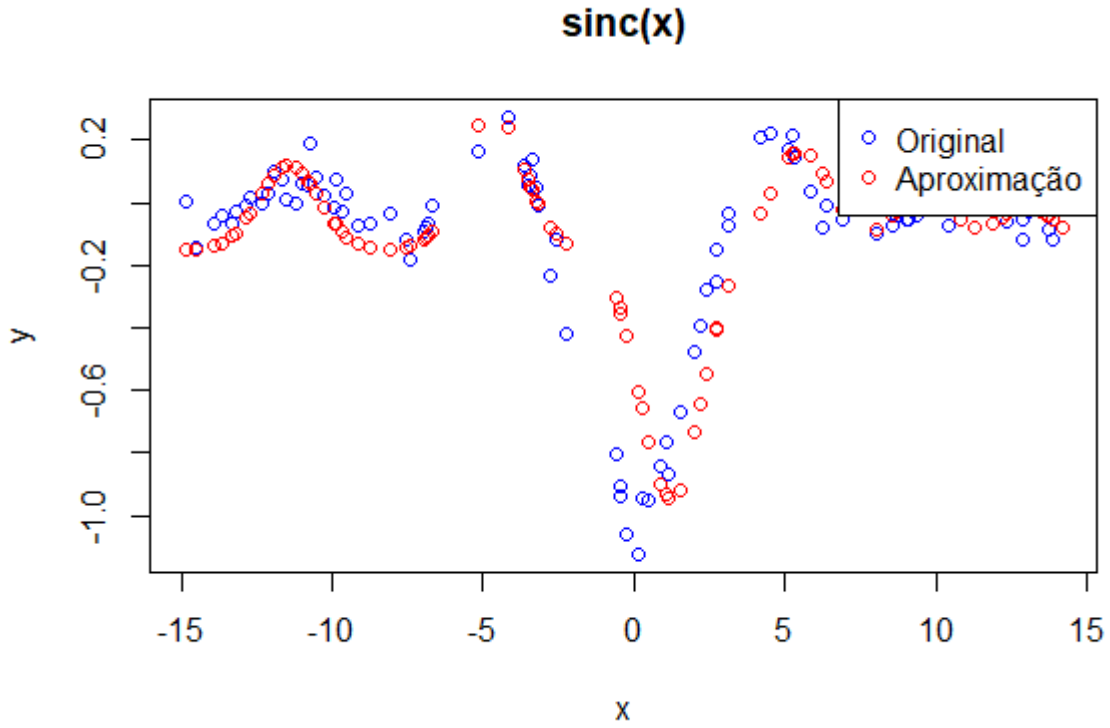

Figura 23: Aproximação com k = 6 e r = 1. MSE = 0.0437

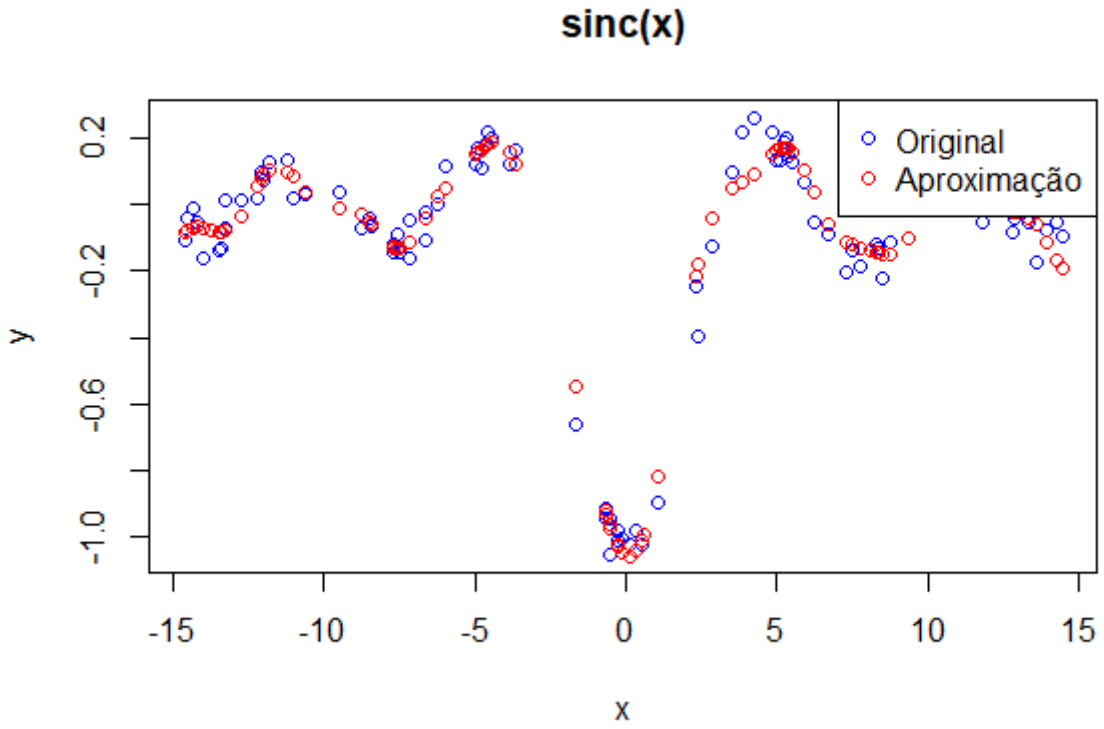

Figura 24: Aproximação com k = 16 e r = 1. MSE = 0.0045

Conclusão:

Através da visualização das superfícies de separação - para a primeira parte do exercício - e da sobreposição dos dados aproximados e valores de MSE - para a segunda parte -. Podemos assumir que, para os conjuntos de dados sintéticos gerados, os modelos com cerca de 10 ou mais centros utilizados no k-médias foram capazes de alcançar desempenhos satisfatórios.