Universidade Federal de Minas Gerais

Nome: Guilherme Vinícius Amorim

Matrícula: 2017089081

Data: 09/2020

Exercício 7

O objetivo dos exercícios desta semana é implementar e testar uma rede neural RBF, com seleção automática de centros e raios, usando a técnica de *k*-médias. Para os primeiros testes, de classificação, devem ser geradas as seguintes bases de dados, utilizando o pacote em R *mlbench*:

• *mlbench.2dnormals(200)*
• *mlbench.xor(100)*
• *mlbench.circle(100)*
• *mlbench.spirals(100,sd = 0.05)*

O primeiro passo da implementação foi carregar os dados e, por ser um problema de classificação, as saídas tiveram que ser ajustadas para -1 e 1. Após o ajuste dos valores de saída, 30% dos dados foram divididos para serem testados enquanto 70% dos dados foram separados para treinamento a partir da função *splitForTrainingAndTest()* fornecida pela *library* 'RSNNS'.

Após esses passos iniciais, os passos finais foram utilizar as rotinas treinaRBF() e YRBF() para treinamento e cálculo do Yhat. Como a aplicação era um problema de classificação, uma função de ativação do tipo f(x) = tanh(x) foi aplicada ao resultado fornecido pelo treinamento da rede RBF.

As duas imagens abaixo contemplam a rotina responsável por carregar os dados das 4 bases de testes e fazer o ajuste dos valores de saída, além da rotina responsável pelo cálculo da rede RBF, assim como a construção de um *grid* para a solução alcançada:

```r
ams1 <- mlbench.2dnormals(200)
ams2 <- mlbench.xor(100)
ams3 <- mlbench.circle(100)
ams4 <- mlbench.spirals(100, sd = 0.5)

X1 <- as.matrix(ams1$x)
Y1 <- as.matrix(c(ams1$classes))
X2 <- as.matrix(ams2$x)
Y2 <- as.matrix(c(ams2$classes))
X3 <- as.matrix(ams3$x)
Y3 <- as.matrix(c(ams3$classes))
X4 <- as.matrix(ams4$x)
Y4 <- as.matrix(c(ams4$classes))

for(i in seq(1, length(Y1), 1)){
  if(Y1[i] == 2)
    Y1[i] <- -1
}

for(i in seq(1, length(Y2), 1)){
  if(Y2[i] == 2)
    Y2[i] <- -1
}

for(i in seq(1, length(Y3), 1)){
  if(Y3[i] == 2)
    Y3[i] <- -1
}

for(i in seq(1, length(Y4), 1)){
  if(Y4[i] == 2)
    Y4[i] <- -1
}
```

Figura 1: Rotina em R para carregar os dados.

```r
pRBF <- treinaRBF(x1train,y1train, p)
Y1hat<-tanh(YRBF(x1test, pRBF))
error1 <- sum((y1test - Y1hat)^2)/length(y1test)*4

seq1x1x2 <- seq(min(X1), max(X1), 0.1)
lseq1 <- length(seq1x1x2)
MZ1 <- matrix(nrow = lseq1, ncol = lseq1)
for (i in 1:lseq1) {
  for (j in 1:lseq1) {
    x1 <- seq1x1x2[i]
    x2 <- seq1x1x2[j]
    x1x2 <- matrix((cbind(x1,x2)), nrow = 1)
    MZ1[i,j] <- tanh(YRBF(x1x2, pRBF))
  }
}

contour(seq1x1x2, seq1x1x2, MZ1, nlevels = 1, ylim = c(max(X1), min(X1)),  xlim = c(max(X1), min(X1)))
par(new =T)
plot(ams1, ylim = c(max(X1), min(X1)),  xlim = c(max(X1), min(X1)))
```

Figura 2: Rotina em R responsável pelos cálculos da rede RBF, assim como a criação de um *grid* que ilustra o resultado alcançado.

Base de dados 2D *normals*

Para os dados da base 2D *normals*, foram utilizados k = 1, 2 e 9 centros. Segue abaixo os resultados obtidos:

```
Error:   3.968174 using 1 centers
Error:   1.504915 using 4 centers
Error:   1.574171 using 9 centers
```

Figura 3: *Mean Square Error* para k = 1, 4 e 9.

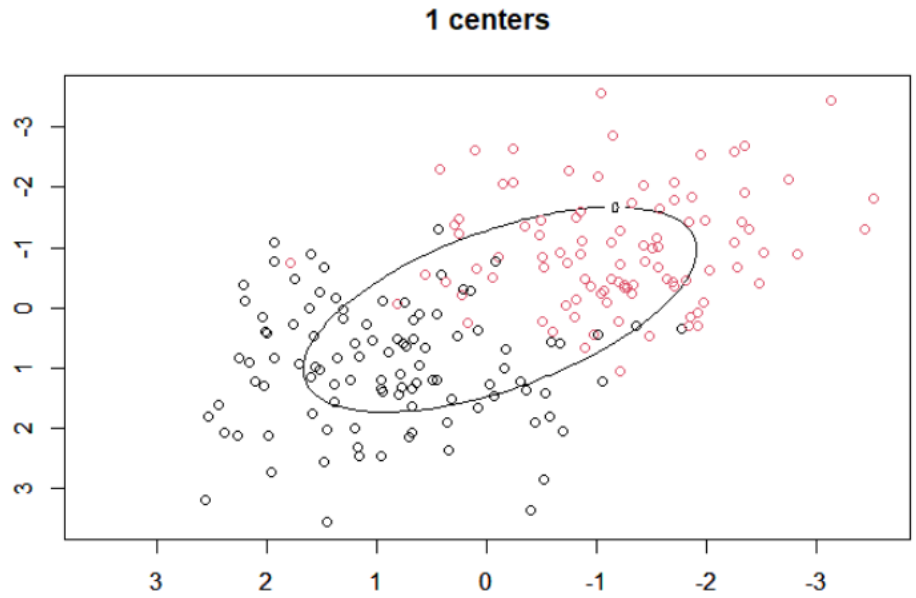

Figura 4: Superfície de separação para k = 1.

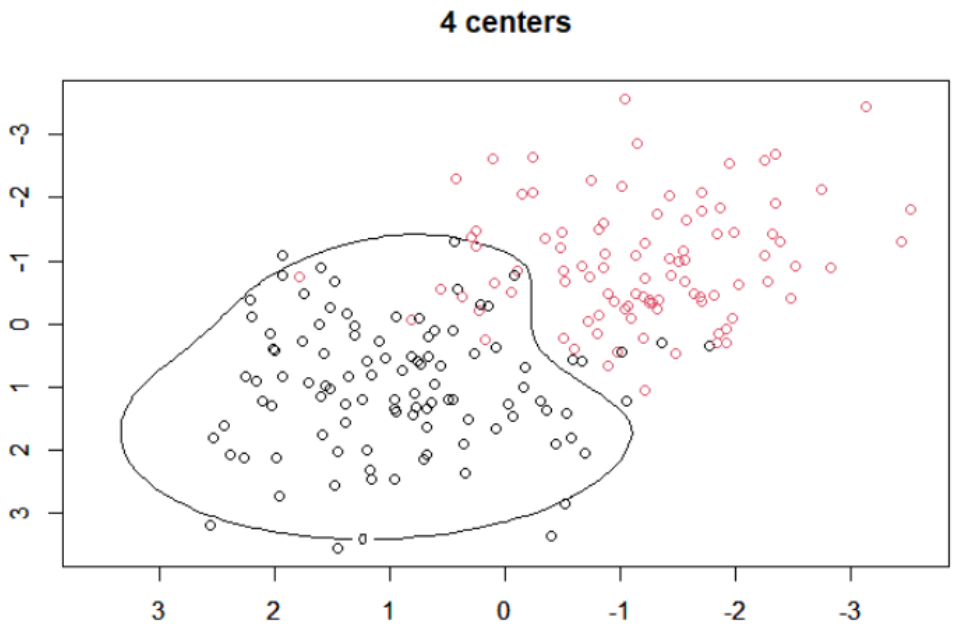

Figura 5: Superfície de separação para k = 4.

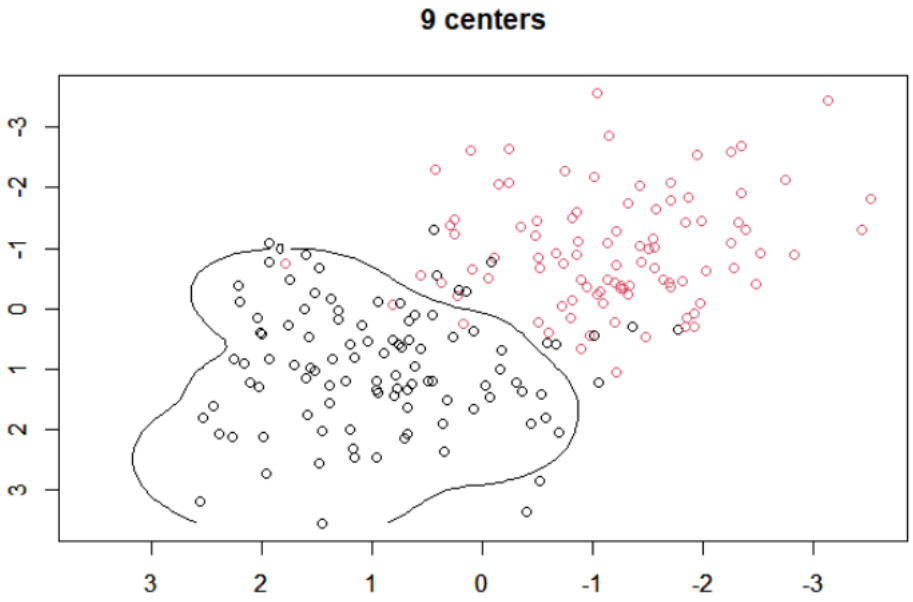

Figura 6: Superfície de separação para k = 9.

Observa-se que os dados dessa base de dados são facilmente separáveis. Assim, k = 4 já oferece um bom resultado de classificação.

XOR

Para os dados da base XOR, foram utilizados k = 1, 9 e 16 centros. Segue abaixo os resultados obtidos:

```
Error:  3.993409 using 1 centers
Error:  2.543713 using 9 centers
Error:  2.563235 using 16 centers
```

Figura 7: *Mean Square Error* para k = 1, 9 e 16.

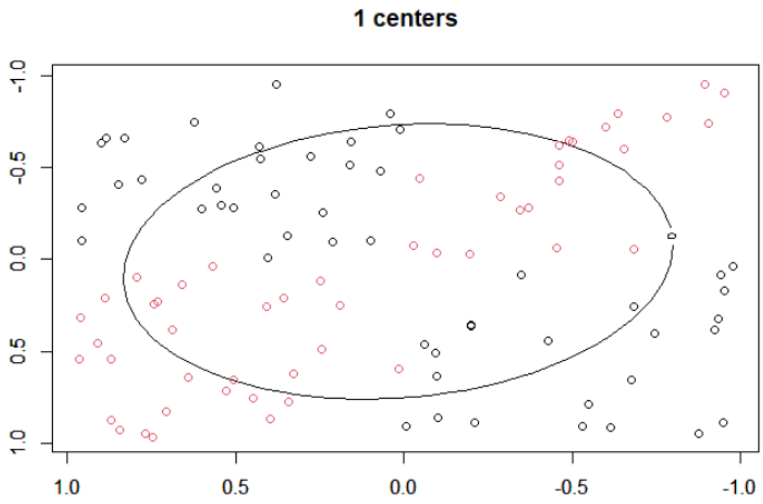

Figura 8: Superfície de separação para k = 1.

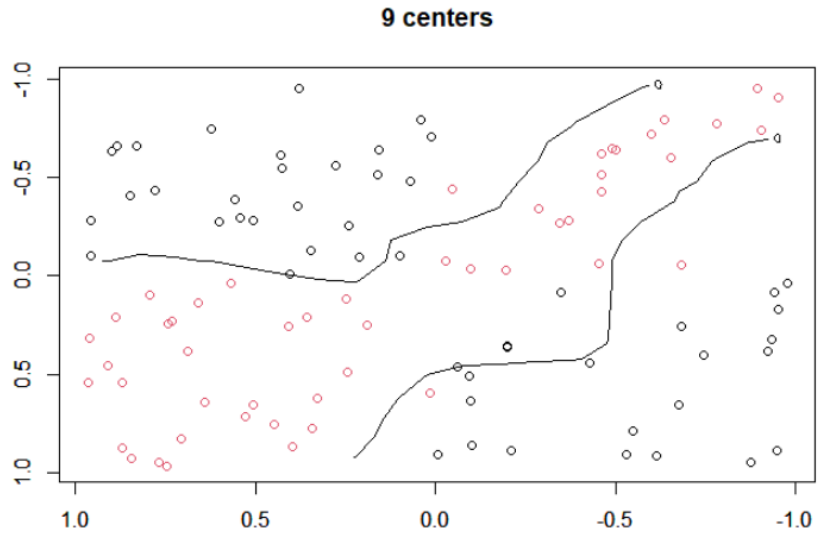

Figura 9: Superfície de separação para k = 9.

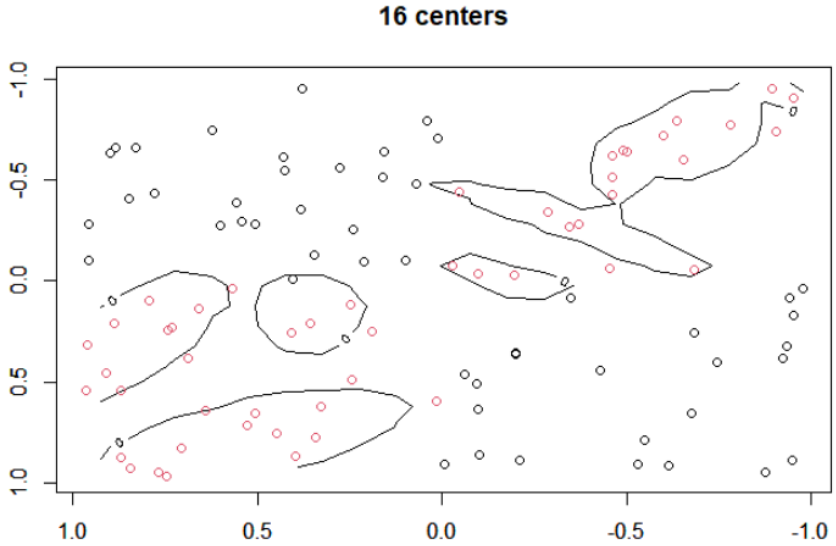

Figura 10: Superfície de separação para k = 16.

Observa-se que os dados dessa base de dados são bem classificados para k = 9, enquanto para k = 16 observa-se um *overfit*.

Circle

Para os dados da base *circle*, foram utilizados k = 4, 9 e 16 centros. Segue abaixo os resultados obtidos:

```
Error:   3.756265 using 4 centers
Error:   3.444837 using 9 centers
Error:   3.037217 using 16 centers
```

Figura 11: *Mean Square Error* para k = 4, 9 e 16.

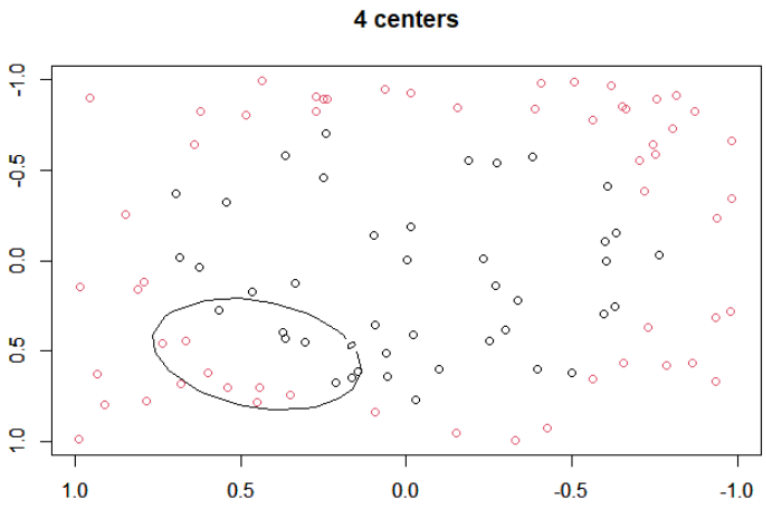

Figura 12: Superfície de separação para k = 4.

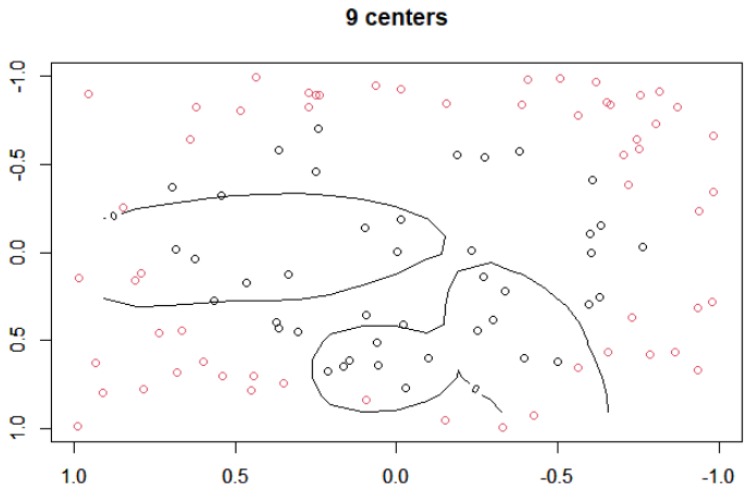

Figura 13: Superfície de separação para k = 9.

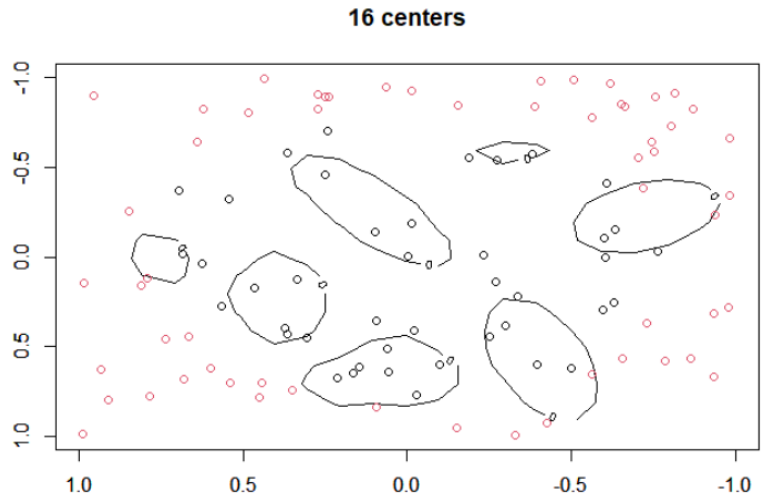

Figura 14: Superfície de separação para k = 16.

Observa-se que os dados dessa base de dados fornecem um *mean square error* com pouca dependência de k. Além disso, percebe-se que k = 4 e k = 9 não são suficientes para uma boa separação dos dados, tendo k = 16 o melhor resultado.

Spirals

Para os dados da base *spirals*, foram utilizados k = 4, 9 e 16 centros. Segue abaixo os resultados obtidos:

```
Error:  3.759427 using 4 centers
Error:  3.97498 using 9 centers
Error:  3.846341 using 16 centers
```

Figura 15: *Mean Square Error* para k = 4, 9 e 16.

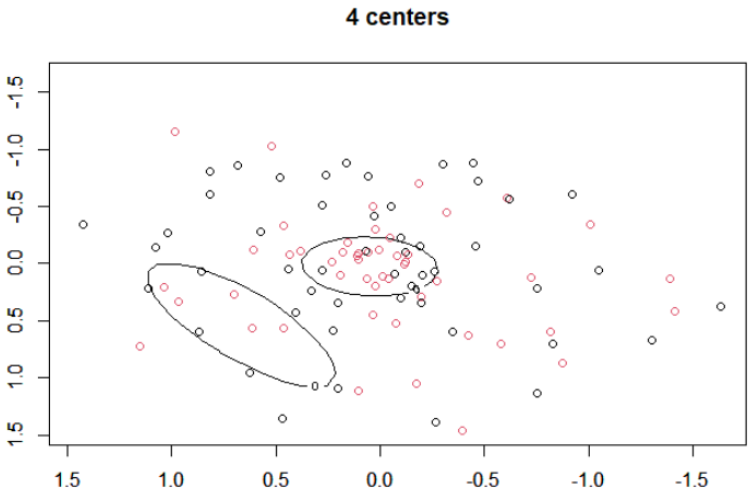

Figura 16: Superfície de separação para k = 4.

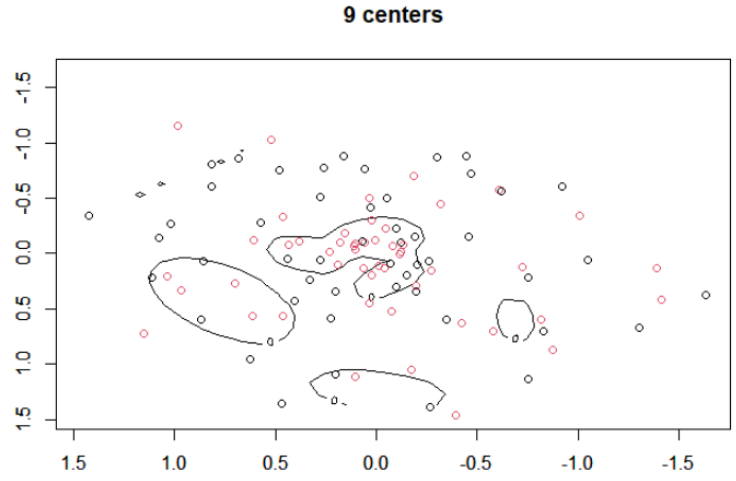

Figura 17: Superfície de separação para k = 9.

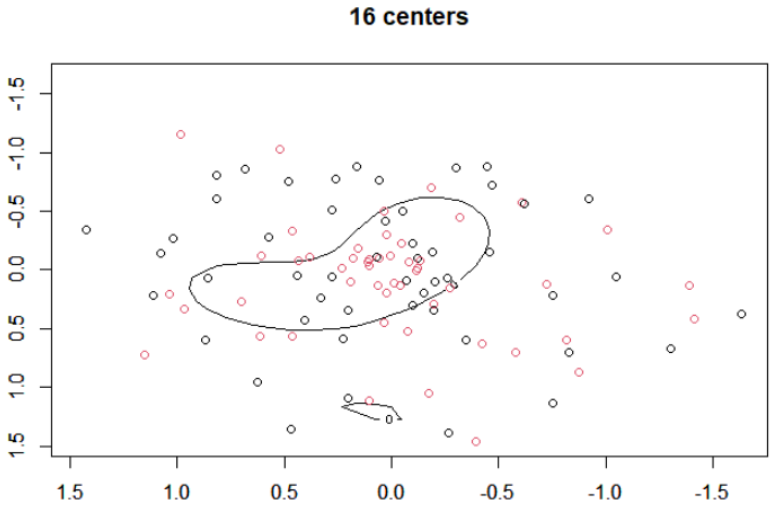

Figura 14: Superfície de separação para k = 16.

Observa-se que os dados dessa base de dados fornecem um *mean square error* com pouca dependência de k. Além disso, nem mesmo k = 16 foi suficiente para se obter uma boa separação dessa base de dados.

Parte 2

Nessa parte, as redes RBFs são utilizadas para se aproximar a função sinc acrescida de um ruído gaussiano. Para tanto, foi utilizado valore de k entre 1 e 10. O desempenho foi medido a partir de um segundo conjunto de 50 amostras (gerado da mesma forma que o primeiro). A métrica a ser usada é o erro quadrático médio. Segue abaixo a rotina que realiza tudo que foi descrito acima:

```
## Data
X <- matrix(runif(100, -15, 15), ncol = 1, nrow = 100)
Y <- matrix((sin(X)/X + rnorm(100, 0, 0.05)), ncol = 1, nrow = 100)

Xtest <- matrix(runif(50, -15, 15), ncol = 1, nrow = 50)
Ytest <- matrix((sin(Xtest)/Xtest + rnorm(50, 0, 0.05)), ncol = 1, nrow = 50)

error_vec<-c()
for (p in 1:10) {

  modRBF <- treinaRBF(X, Y, p)

  Yhat <- YRBF(Xtest, modRBF)
  error <- sum((Ytest - Yhat)^2)/length(Ytest)

  error_vec<-c(error_vec,error)
}

plot(1:10, error_vec, type = 'b')
```

Figura 15: Rotina em R para a aproximação da função sinc(x).

O seguinte gráfico relaciona o *mean square error* para valores de k entre 1 e 10.

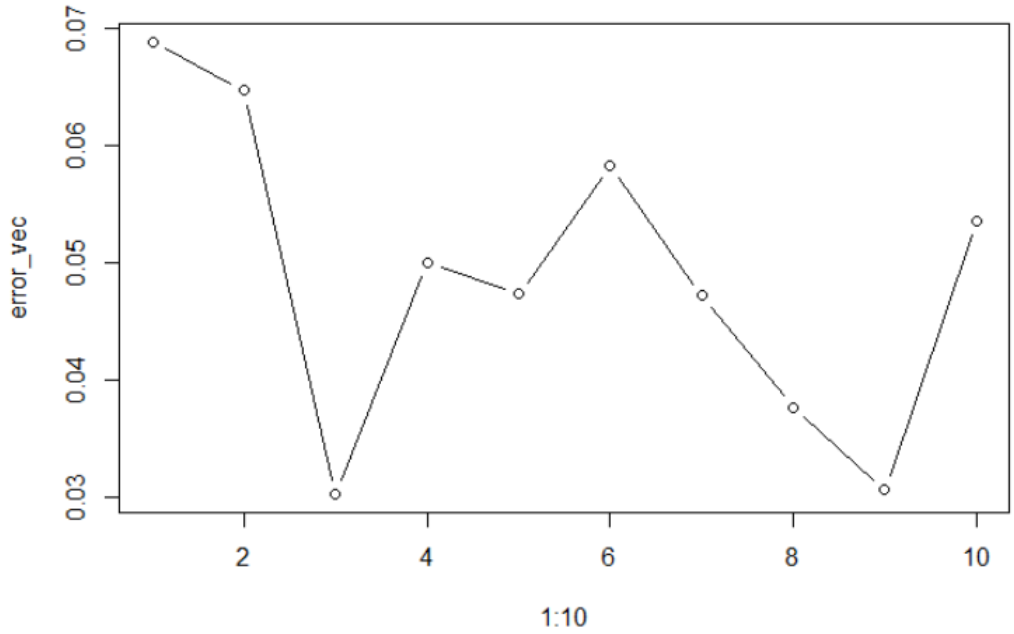

Figura 16: *Mean Square Error* para valores de k entre 1 e 10.

Segue abaixo a função sinc reconstruída para k = 3 (número de centros com o menor erro):

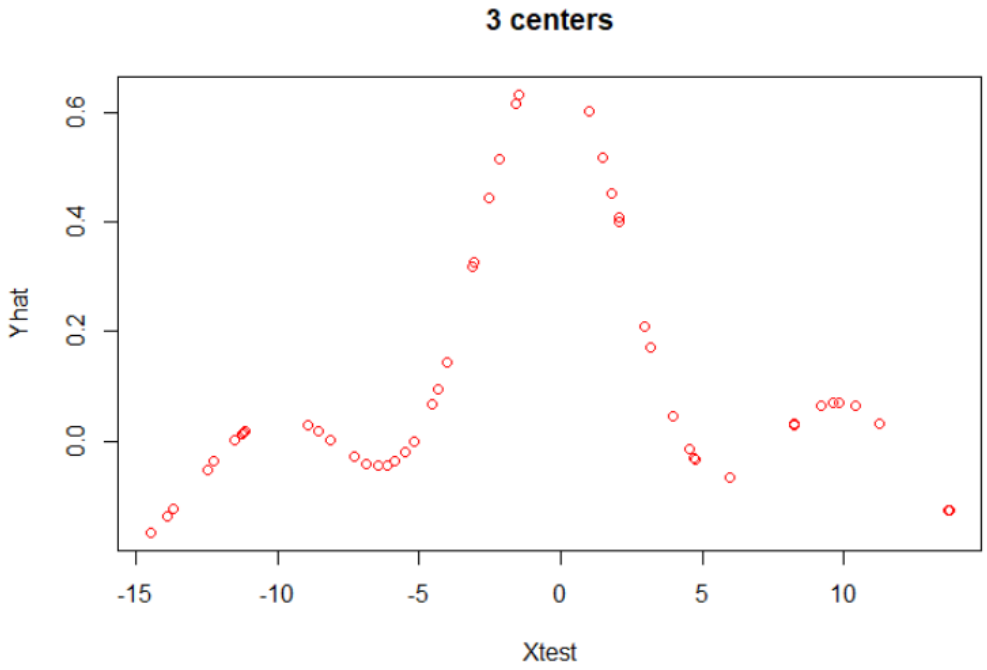

Figura 17: Reconstrução da função sinc a partir de uma rede RBF com k = 3.