Redes Neurais Artificiais

Exercício 7 - Redes RBF

Vítor Gabriel Reis Caitité - 2016111849

02/08/2021

Função que calcula a saída de uma rede RBF

Abaixo está a função que calcula a saída de uma rede RBF.

```
# Função que calcula a saı´da de uma rede RBF.
library("corpcor")

YRBF <- function(xin, modRBF){
  ####### Função radial Gaussiana ########
  pdfnvar<-function(x,m,K,n){
    if (n==1) {
      r<-sqrt(as.numeric(K))
      px<-(1/(sqrt(2*pi*r*r))) * exp(-0.5 *((x-m)/r)^2)
    }
    else {
      px<-((1/(sqrt((2*pi)^n * (det(K))))) * exp (-0.5 * (t(x-m) %*% (solve(K)) %*% (x-
m))))
    }
  }
  ########################################
  N <- dim(xin)[1] # número de amostras
  n <- dim(xin)[2] # dimensão de entrada (deve ser maior que 1)
  m <- as.matrix(modRBF[[1]])
  covlist <- modRBF[[2]]
  p <- length(covlist) # Número de funções radiais
  W <- modRBF [[3]]

  xin <- as.matrix(xin) # garante que xin seja matriz

  H <- matrix(nrow = N, ncol = p)
  # Calcula matriz H
  for (j in 1:N) {
    for (i in 1:p) {
      mi <- m[i, ]
      covi <- covlist[i]
      covi <- matrix(unlist(covlist[i]), ncol = n, byrow = T) + 0.001 * diag(n)
      H[j,i] <- pdfnvar(xin[j, ], mi, covi, n)
    }
  }

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

Treinamento da rede RBF

A função abaixo é uma implementação possível, em R, para o algoritmo de treinamento de uma rede RBF. Essa função é utilizada na resolução dos exercícios da lista.

```
# Função de treinamento de uma rede RBF.
library("corpcor")

trainRBF <- function(xin, yin, p){
  ####### Função radial Gaussiana ########
  pdfnvar<-function(x,m,K,n){
    if (n==1) {
      r<-sqrt(as.numeric(K))
      px<-(1/(sqrt(2*pi*r*r)))*exp(-0.5 *((x-m)/r)^2)
    }
    else {
      px<-((1/(sqrt((2*pi)^n * (det(K))))) * exp (-0.5 * (t(x-m) %*% (solve(K)) %*% (x-
m)))) #eq 6.5
    }
  }
  ########################################
  N<-dim(xin)[1] # número de amostras
  n<-dim(xin)[2] # dimensão de entrada (deve ser maior que 1)

  xin <- as.matrix(xin) # garante que xin seja matriz
  yin <- as.matrix(yin) # garante que yin seja matriz

  # Aplica o algoritmo kmeans para separar os clusters
  xclust<-kmeans(xin, p)

  # Armazena vetores de centros das funções:
  m <- as.matrix (xclust$centers)
  covlist <- list()

  # Estima matrizes de covariância para todos os centros:
  for ( i in 1:p)
  {
    ici <- which(xclust$cluster == i )
    xci <- xin [ici, ]
    if(n==1){
      covi <- var(xci)
    }
    else{
      covi <- cov(xci)
    }
    covlist [[i]] <- covi
  }

  H <- matrix(nrow = N, ncol = p)
  # Calcula matriz H
  for (j in 1:N) {
    for (i in 1:p) {
      mi <- m[i, ]
      covi <- covlist[i]
      covi <- matrix(unlist(covlist[i]), ncol = n, byrow = T) + 0.001 * diag(n)
      H[j,i] <- pdfnvar(xin[j, ] , mi, covi, n)
    }
  }

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

  return (list(m, covlist, W, H))
}
```

Enunciado Exercício 5

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 (pacotes equivalentes em python ou outra linguagem podem ser utilizados):

• mlbench.2dnormals(200)

• mlbench.xor(100)

• mlbench.circle(100)

• mlbench.spirals(100,sd = 0.05)

Para cada uma das bases, devem ser testados pelo menos três valores de k (número de centros) para a função de k-médias, e devem ser construídas redes RBF conforme mostrado nas vídeo-aulas. Assim como feito para o exercício de ELMs, a superfície de separação obtida para cada uma das redes deve ser mostrada no relatório.

A segunda parte desta atividade consiste em construir uma rede RBF para aproximar a função sinc acrescida de um ruído gaussiano:

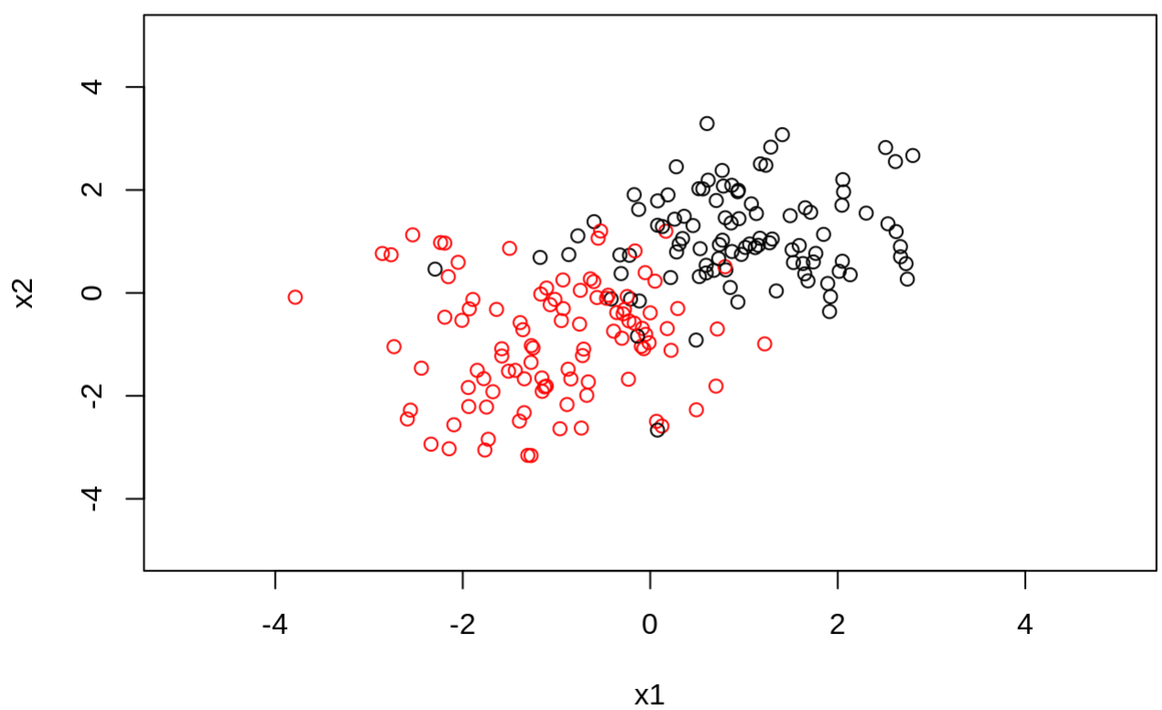
• sinc(x) = sin(x)/x

Assim como feito para a primeira parte, devem ser ajustadas, pelo menos 3 redes RBF, com diferentes números de centros. As 3 redes devem ter seu desempenho comparado em um segundo conjunto de 50 amostras (gerado da mesma forma que o primeiro). A métrica a ser usada é o erro quadrático médio.

Parte 1 - Base de dados mlbench.2dnormals

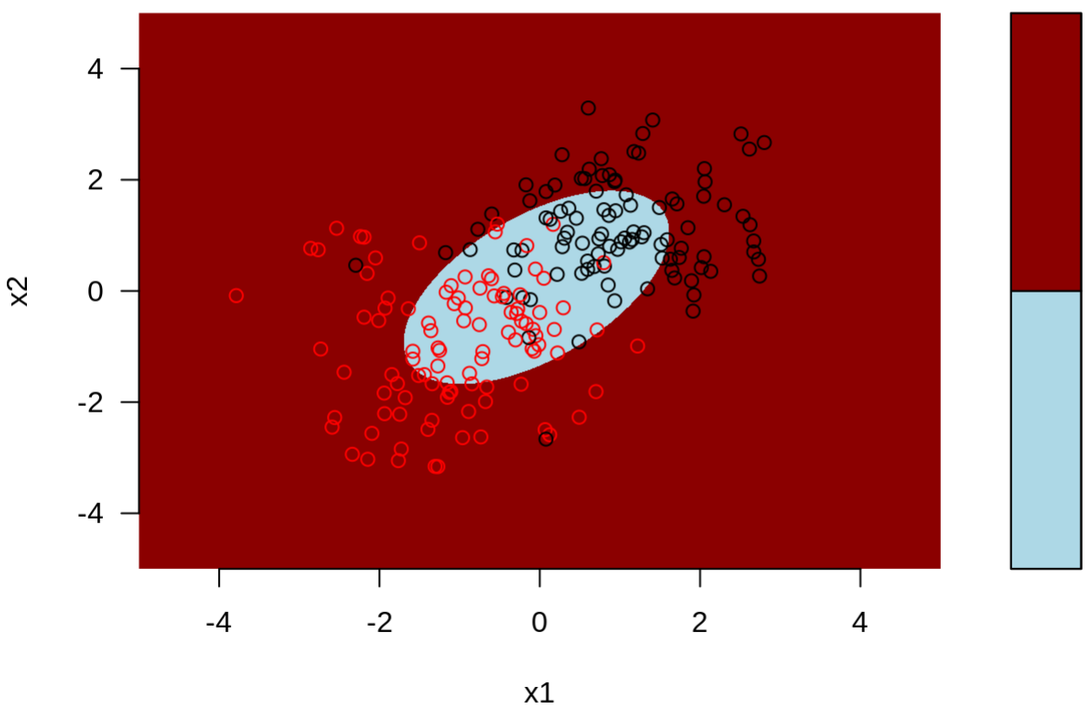
Para essa base de dados buscou-se utilizar os valores de k como 1, 2 e 10. Afim de se poder observar um solução com underfitting (para k=1), uma solução visualmente boa (para k = 2) e outra solução que já apresenta sinais de overfitting (para k=10). As superfícies de separação obtidas para cada uma das 3 redes serão mostradas abaixo.

```
rm(list = ls())
source(file = "/home/vitor/Documents/UFMG/9/Redes Neurais/exemplos/trainRBF.R")
source(file = "/home/vitor/Documents/UFMG/9/Redes Neurais/exemplos/YRBF.R")
library(mlbench)

data <- mlbench.2dnormals(200)

xin<-matrix(data[["x"]], ncol = 2)
class<-matrix(data[["classes"]], ncol=1)
plot(xin[,1], xin[,2], col=data$classes, xlim=c(-5, 5), ylim=c(-5, 5), ylab = "x2", xlab
 = "x1")
```

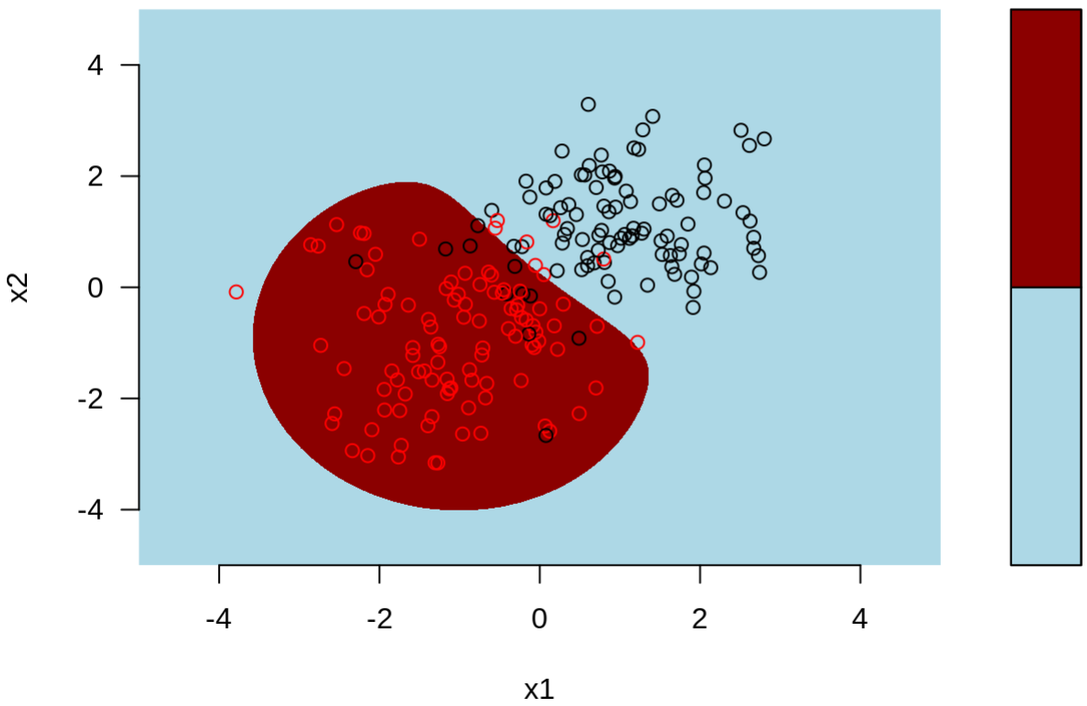


```
yin <- rep(0,200)
for (count in 1:length(class)) {
  if (class[count] == 1 ){
    yin[count] = -1
  }
  else if(class[count] == 2){
    yin[count] = 1
  }
}

pvector<-c(1,2,10)
for (p in pvector) {
  modRBF<-trainRBF(xin,yin,p)

  # Plotando superfície
  seqx1x2 <- seq(-5, 5, 0.1)
  MZ<-matrix(nrow = length(seqx1x2), ncol = length(seqx1x2))
  cr<-0
  for (i in 1:length(seqx1x2)) {
    for (j in 1:length(seqx1x2)) {
      cr<-cr+1
      x1<-seqx1x2[i]
      x2<-seqx1x2[j]
      x1x2<-matrix(cbind(x1,x2), nrow = 1)
      MZ[i,j]<-YRBF(x1x2, modRBF)
    }
  }

  #contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-5,5), ylim=c(-5,5), xlab = "x1", y
lab = "x2")
  #par(new=T)
  print(paste("Classificação utilizando valor k = ", p))
  #plot(xin[,1], xin[,2], col=data$classes, xlim=c(-5,5), ylim=c(-5,5), ylab = "", xlab
 = "")
  cols = c('lightblue', 'darkred')
  filled.contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-5,5), ylim=c(-5,5), xlab =
"x1", ylab = "x2", col = cols, axes=F, plot.axes = { points(xin[,1], xin[,2],col=data$cl
asses, xlim=c(-5,5), ylim=c(-5,5)); axis(1); axis(2) })
}
```

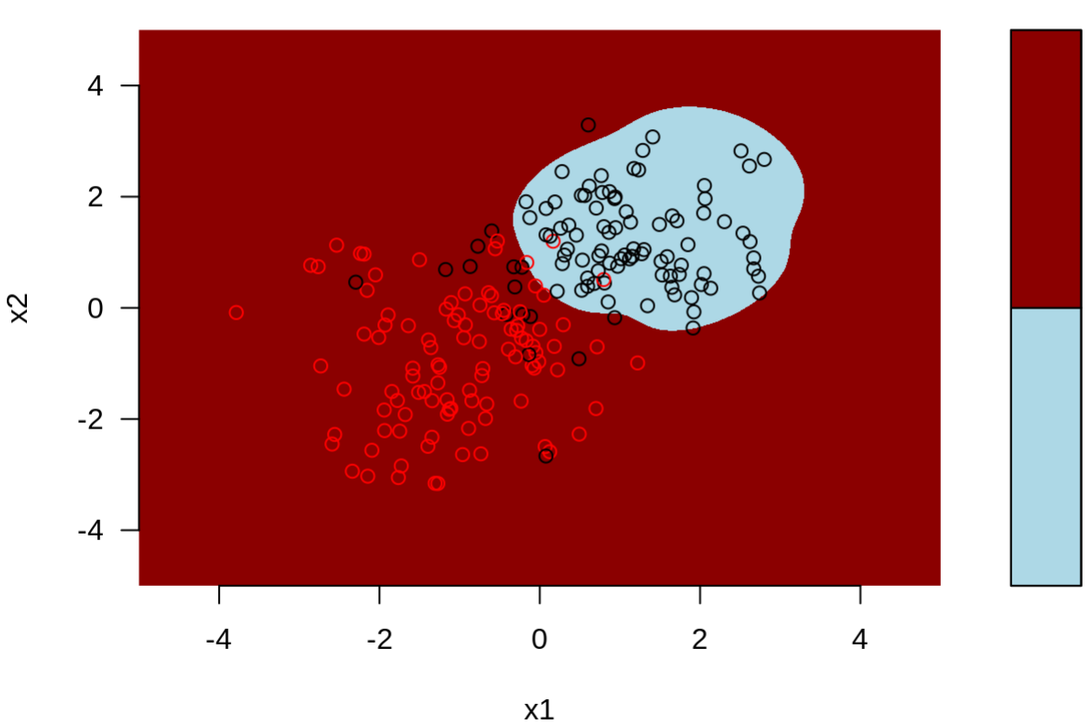
```
## [1] "Classificação utilizando valor k =  1"
```



```
## [1] "Classificação utilizando valor k =  2"
```



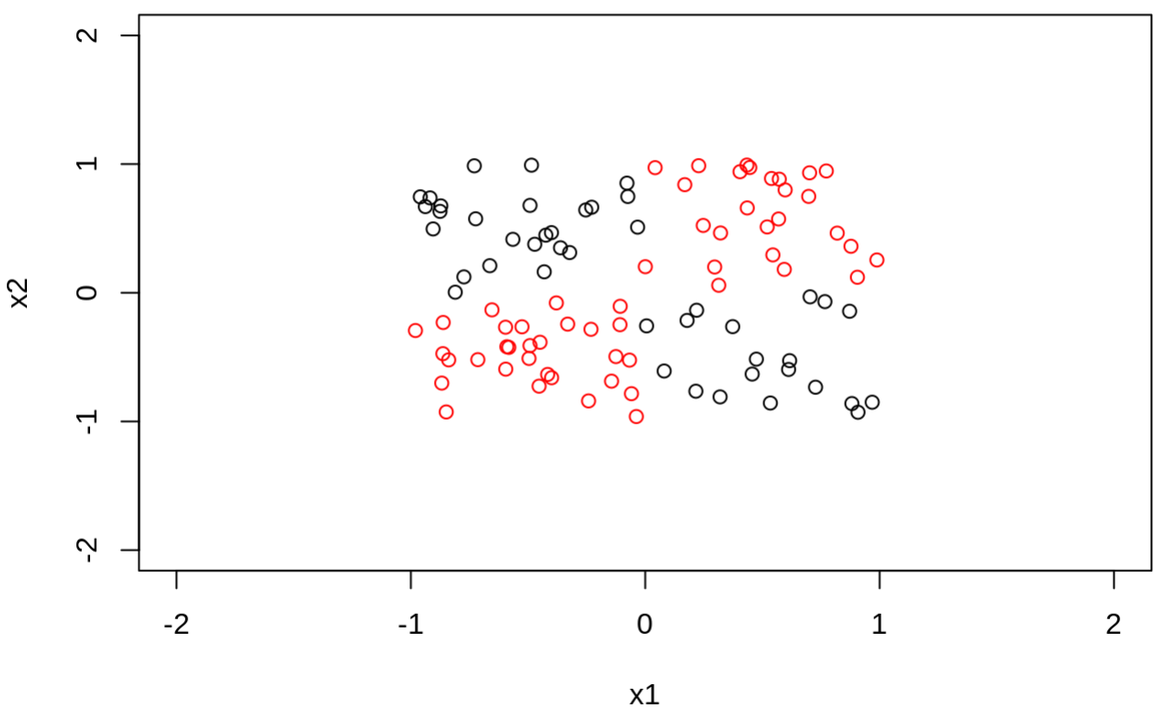

```
## [1] "Classificação utilizando valor k =  10"
```



Parte 1 - Base de dados mlbench.xor

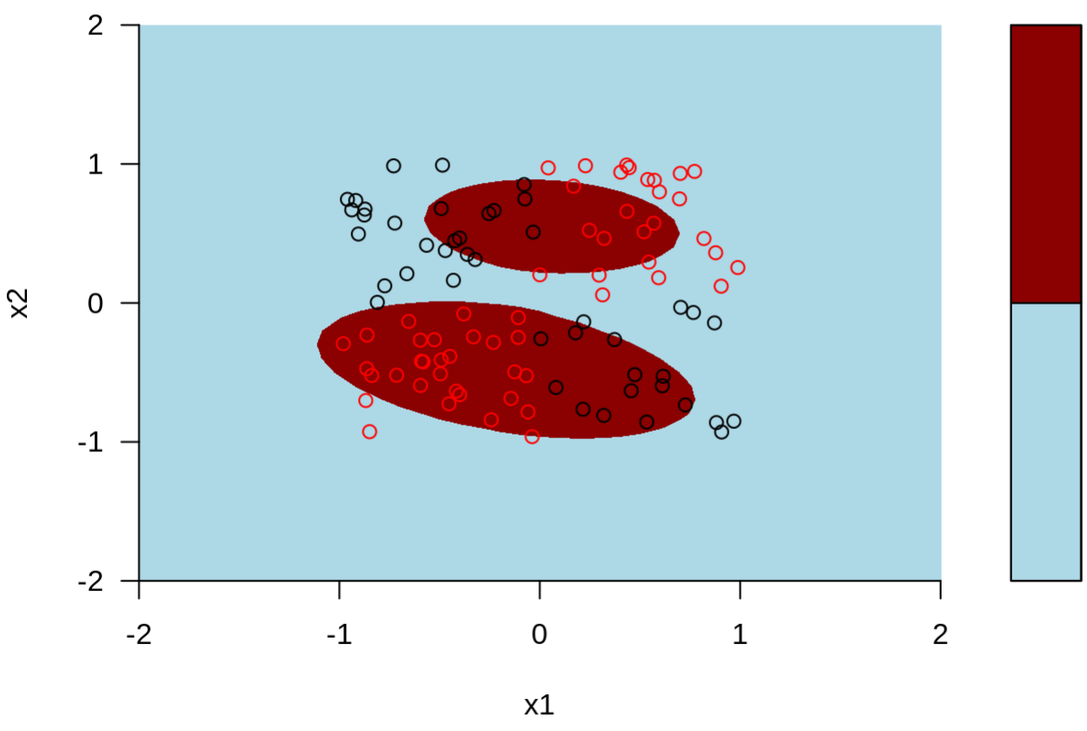
Já para a base de dados dados mlbench.xor utilizou-se os valores de k como 2, 4 e 10. Isso foi feito afim de se poder observar um solução com underfitting (para k=2), uma solução visualmente boa (para k = 4) e outra solução que já apresenta sinais de overfitting (para k=10). As superfícies de separação obtidas para cada uma das 3 redes serão mostradas abaixo.

```
rm(list = ls())
source(file = "/home/vitor/Documents/UFMG/9/Redes Neurais/exemplos/trainRBF.R")
source(file = "/home/vitor/Documents/UFMG/9/Redes Neurais/exemplos/YRBF.R")
library(mlbench)

data <- mlbench.xor(100)

xin<-matrix(data[["x"]], ncol = 2)
class<-matrix(data[["classes"]], ncol=1)
plot(xin[,1], xin[,2], col=data$classes, xlim=c(-2, 2), ylim=c(-2, 2), ylab = "x2", xlab
 = "x1")
```

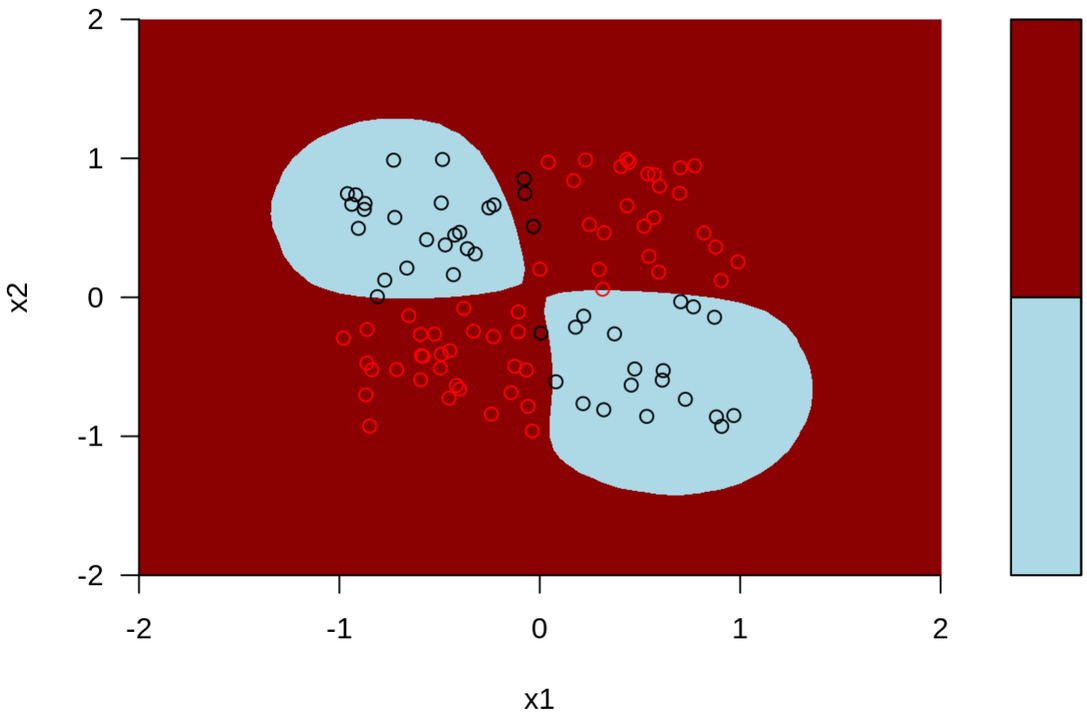


```
yin <- rep(0,100)
for (count in 1:length(class)) {
  if (class[count] == 1 ){
    yin[count] = -1
  }
  else if(class[count] == 2){
    yin[count] = 1
  }
}

pvector<-c(2,4,10)
for (p in pvector) {
  modRBF<-trainRBF(xin,yin,p)

  # Plotando superfície
  seqx1x2 <- seq(-5, 5, 0.1)
  MZ<-matrix(nrow = length(seqx1x2), ncol = length(seqx1x2))
  cr<-0
  for (i in 1:length(seqx1x2)) {
    for (j in 1:length(seqx1x2)) {
      cr<-cr+1
      x1<-seqx1x2[i]
      x2<-seqx1x2[j]
      x1x2<-matrix(cbind(x1,x2), nrow = 1)
      MZ[i,j]<-YRBF(x1x2, modRBF)
    }
  }

  #contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-5,5), ylim=c(-5,5), xlab = "x1", y
lab = "x2")
  #par(new=T)
  print(paste("Classificação utilizando valor k = ", p))
  #plot(xin[,1], xin[,2], col=data$classes, xlim=c(-2,2), ylim=c(-2,2), ylab = "", xlab
 = "")
  cols = c('lightblue', 'darkred')
  filled.contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-2,2), ylim=c(-2,2), xlab =
"x1", ylab = "x2", col = cols, axes=F, plot.axes = { points(xin[,1], xin[,2],col=data$cl
asses, xlim=c(-5,5), ylim=c(-5,5)); axis(1); axis(2) })
}
```

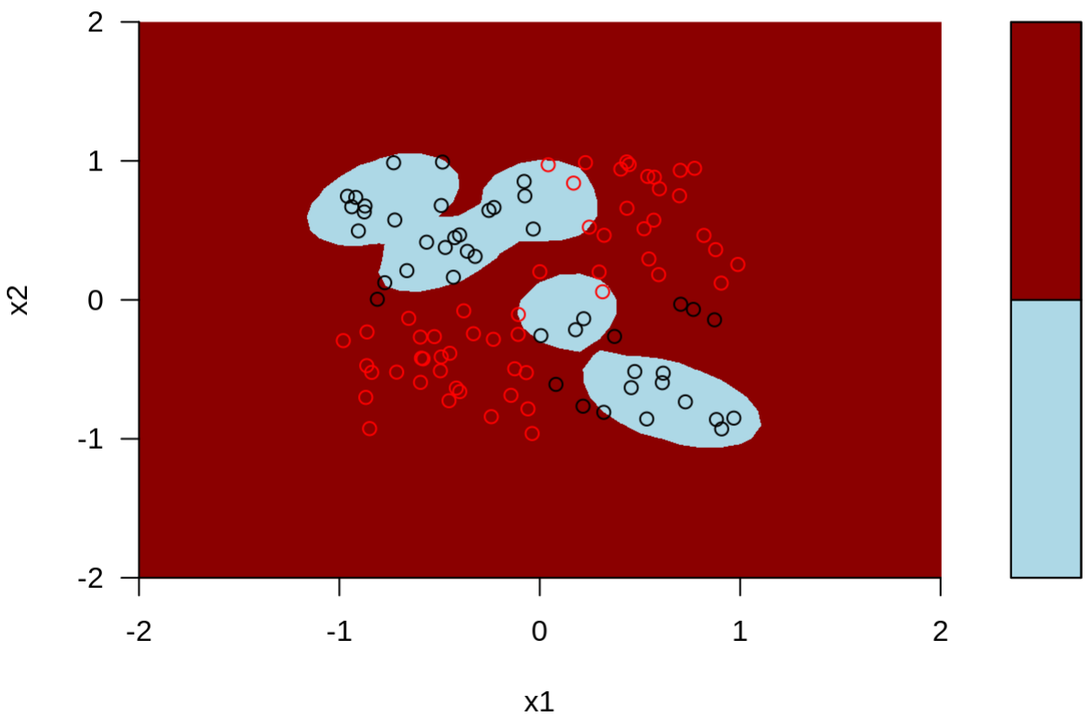
```
## [1] "Classificação utilizando valor k =  2"
```



```
## [1] "Classificação utilizando valor k =  4"
```



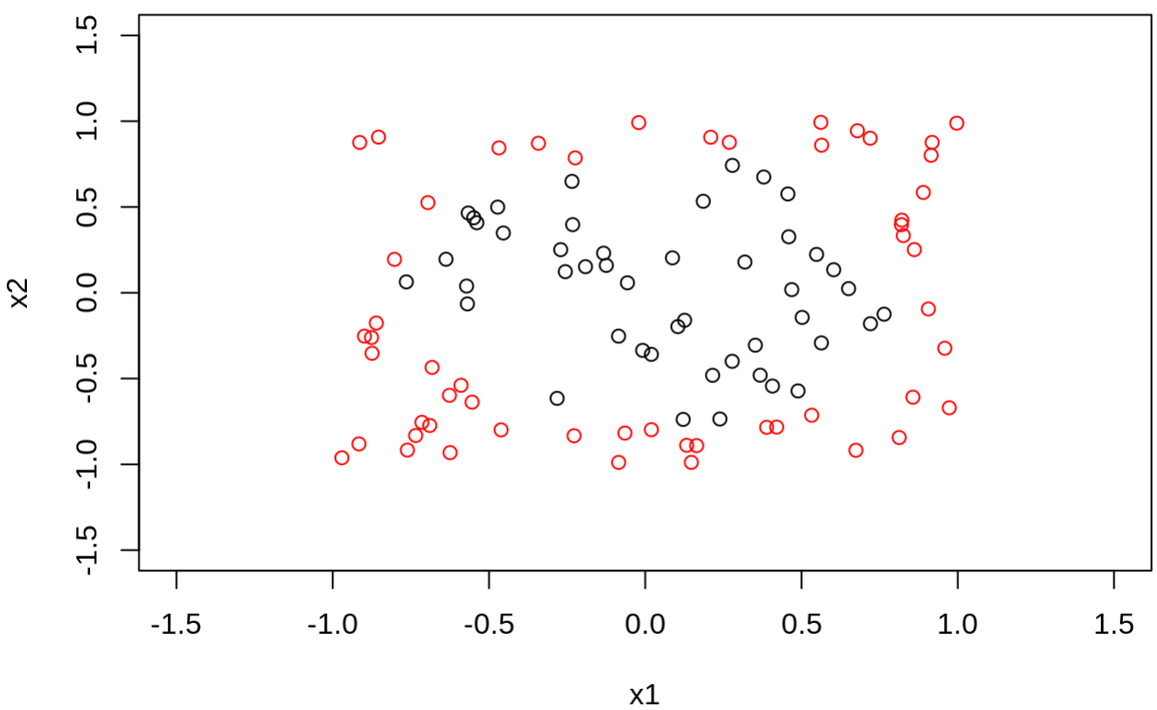

```
## [1] "Classificação utilizando valor k =  10"
```



Base de dados mlbench.circle

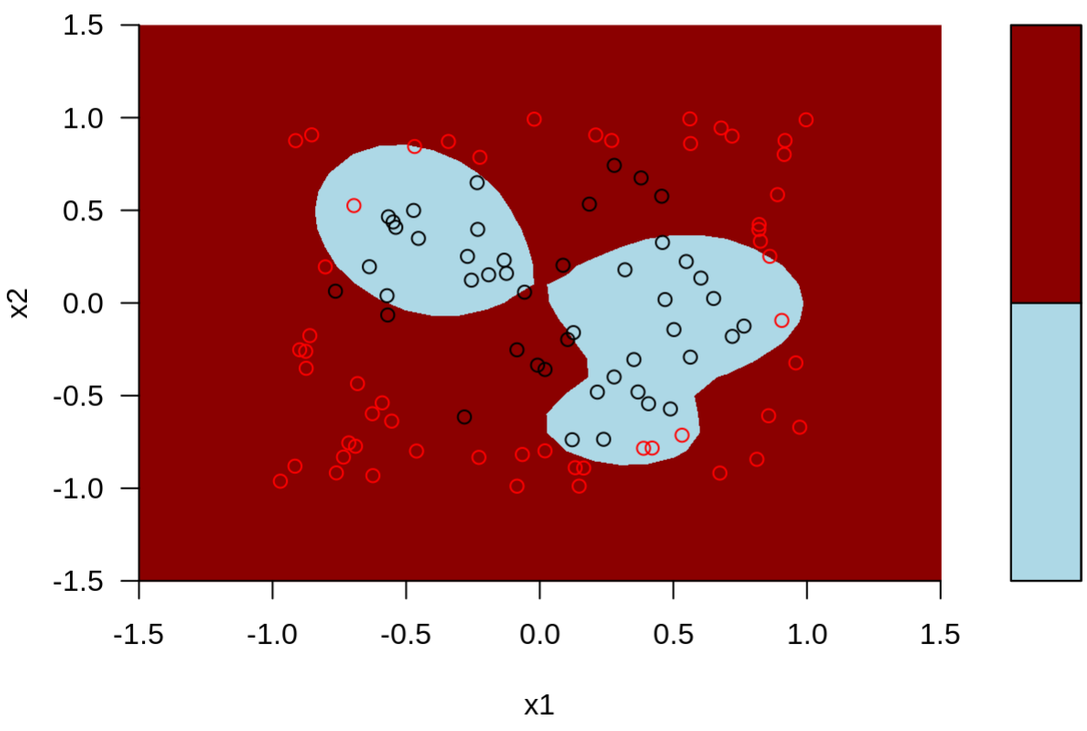
Para a base de dados dados mlbench.circle utilizou-se os valores de k como 5, 10 e 20. Isso foi feito afim de se poder observar um solução com underfitting (para k=5), uma solução visualmente boa (para k = 10) e outra solução que já apresenta sinais de overfitting (para k=20). As superfícies de separação obtidas para cada uma das 3 redes serão mostradas abaixo.

```
rm(list = ls())
source(file = "/home/vitor/Documents/UFMG/9/Redes Neurais/exemplos/trainRBF.R")
source(file = "/home/vitor/Documents/UFMG/9/Redes Neurais/exemplos/YRBF.R")
library(mlbench)

data <- mlbench.circle(100)

xin<-matrix(data[["x"]], ncol = 2)
class<-matrix(data[["classes"]], ncol=1)
plot(xin[,1], xin[,2], col=data$classes, xlim=c(-1.5, 1.5), ylim=c(-1.5, 1.5), ylab = "x
2", xlab = "x1")
```

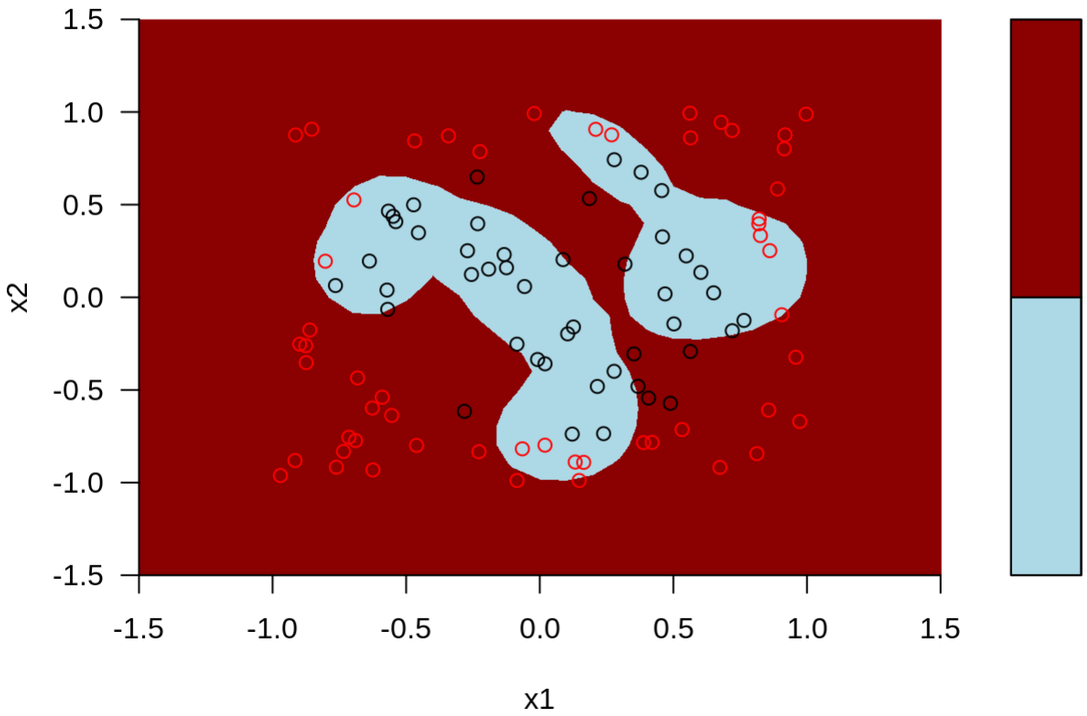


```
yin <- rep(0,100)
for (count in 1:length(class)) {
  if (class[count] == 1 ){
    yin[count] = -1
  }
  else if(class[count] == 2){
    yin[count] = 1
  }
}

pvector<-c(5,10,20)
for (p in pvector) {
  modRBF<-trainRBF(xin,yin,p)

  # Plotando superfície
  seqx1x2 <- seq(-5, 5, 0.1)
  MZ<-matrix(nrow = length(seqx1x2), ncol = length(seqx1x2))
  cr<-0
  for (i in 1:length(seqx1x2)) {
    for (j in 1:length(seqx1x2)) {
      cr<-cr+1
      x1<-seqx1x2[i]
      x2<-seqx1x2[j]
      x1x2<-matrix(cbind(x1,x2), nrow = 1)
      MZ[i,j]<-YRBF(x1x2, modRBF)
    }
  }

  #contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-5,5), ylim=c(-5,5), xlab = "x1", y
lab = "x2")
  #par(new=T)
  print(paste("Classificação utilizando valor k = ", p))
  #plot(xin[,1], xin[,2], col=data$classes, xlim=c(-1,1), ylim=c(-1.5,1.5), ylab = "", x
lab = "")
  cols = c('lightblue', 'darkred')
  filled.contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-1.5,1.5), ylim=c(-1.5,1.5),
 xlab = "x1", ylab = "x2", col = cols, axes=F, plot.axes = { points(xin[,1], xin[,2],col
=data$classes, xlim=c(-5,5), ylim=c(-5,5)); axis(1); axis(2) })
}
```

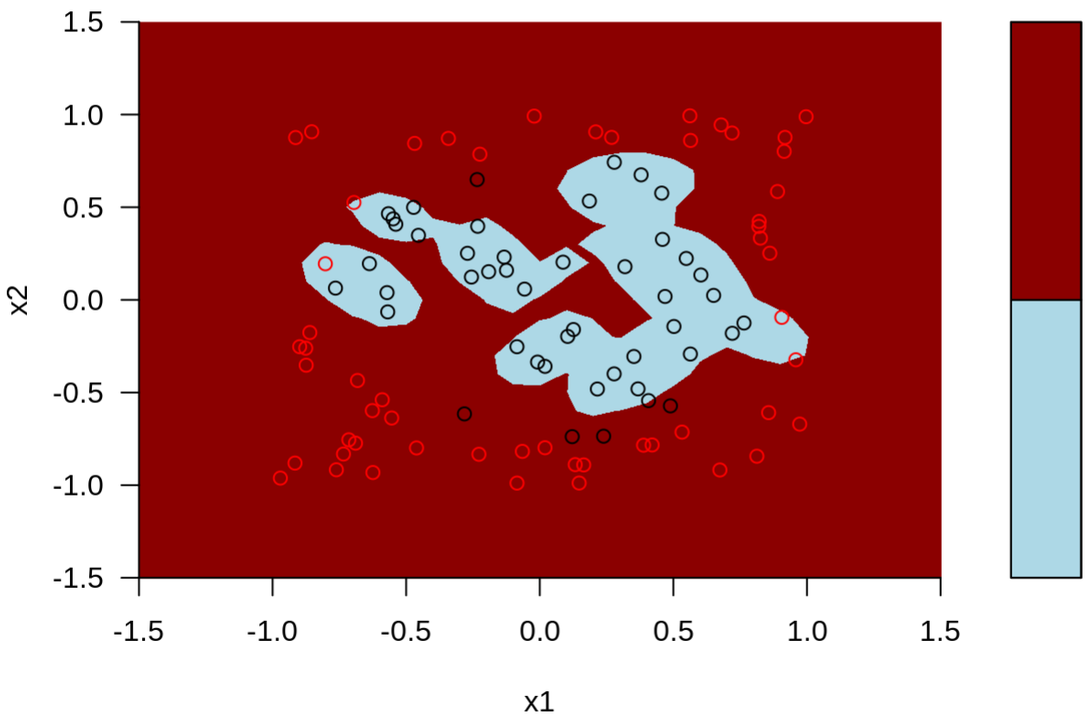
```
## [1] "Classificação utilizando valor k =  5"
```



```
## [1] "Classificação utilizando valor k =  10"
```

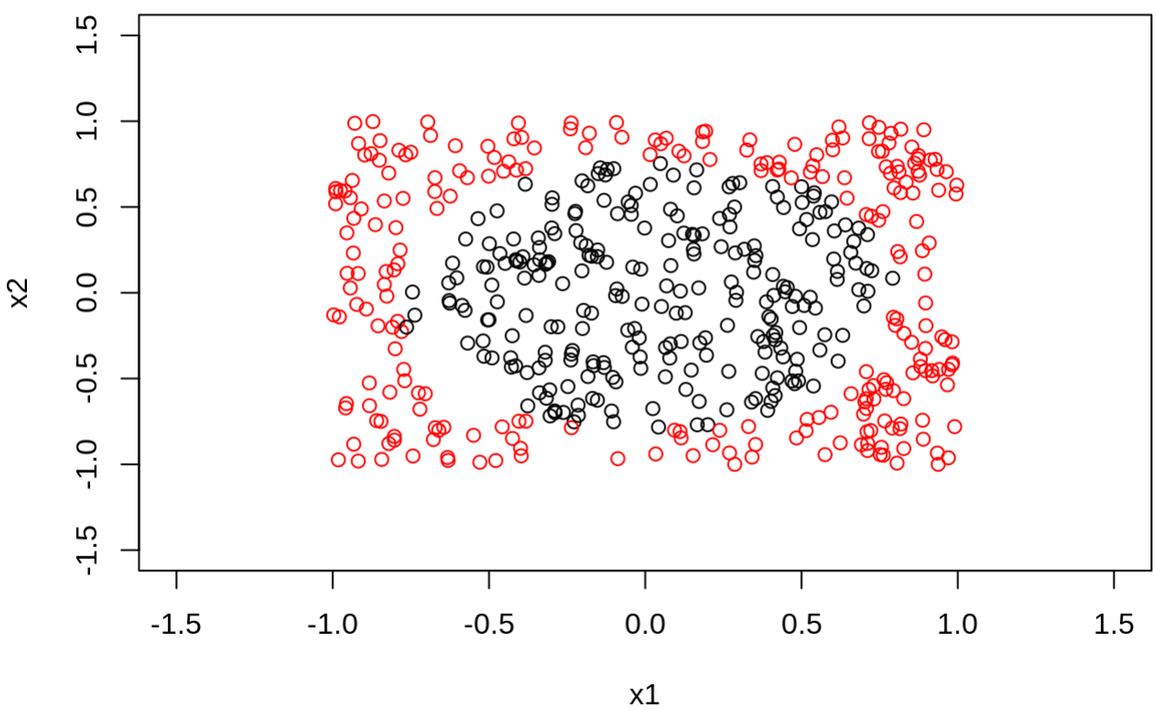



```
## [1] "Classificação utilizando valor k =  20"
```



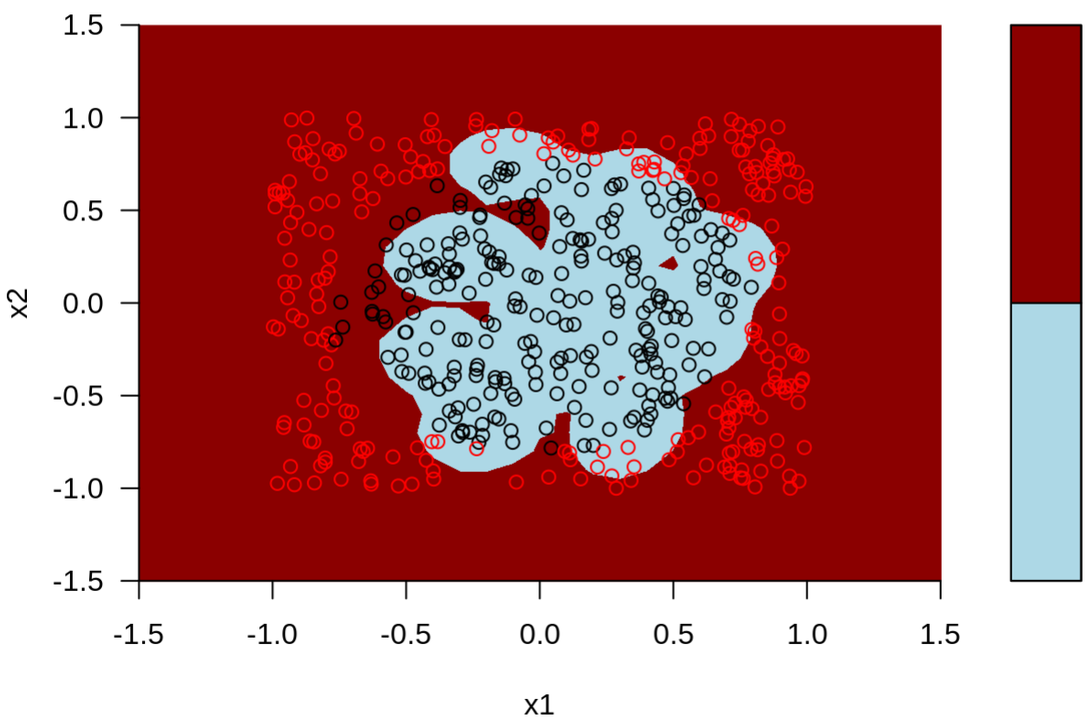
Para essa base de dados decidiu-se investigar um pouco a influência da quantidade de dados no treinamento. Como pode-se ver abaixo ao rodar o algoritmo de treinamento para 500 amostras (e k=15) obtém-se uma superfície mais próxima do esperado anteriormente.

```
rm(list = ls())
source(file = "/home/vitor/Documents/UFMG/9/Redes Neurais/exemplos/trainRBF.R")
source(file = "/home/vitor/Documents/UFMG/9/Redes Neurais/exemplos/YRBF.R")
library(mlbench)

data <- mlbench.circle(500)

xin<-matrix(data[["x"]], ncol = 2)
class<-matrix(data[["classes"]], ncol=1)
plot(xin[,1], xin[,2], col=data$classes, xlim=c(-1.5, 1.5), ylim=c(-1.5, 1.5), ylab = "x
2", xlab = "x1")
```



```
yin <- rep(0,100)
for (count in 1:length(class)) {
  if (class[count] == 1 ){
    yin[count] = -1
  }
  else if(class[count] == 2){
    yin[count] = 1
  }
}

pvector<-c(15)
for (p in pvector) {
  modRBF<-trainRBF(xin,yin,p)

  # Plotando superfície
  seqx1x2 <- seq(-5, 5, 0.1)
  MZ<-matrix(nrow = length(seqx1x2), ncol = length(seqx1x2))
  cr<-0
  for (i in 1:length(seqx1x2)) {
    for (j in 1:length(seqx1x2)) {
      cr<-cr+1
      x1<-seqx1x2[i]
      x2<-seqx1x2[j]
      x1x2<-matrix(cbind(x1,x2), nrow = 1)
      MZ[i,j]<-YRBF(x1x2, modRBF)
    }
  }

  #contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-5,5), ylim=c(-5,5), xlab = "x1", y
lab = "x2")
  #par(new=T)
  print(paste("Classificação utilizando valor k = ", p))
  #plot(xin[,1], xin[,2], col=data$classes, xlim=c(-1,1), ylim=c(-1.5,1.5), ylab = "", x
lab = "")
  cols = c('lightblue', 'darkred')
  filled.contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-1.5,1.5), ylim=c(-1.5,1.5),
 xlab = "x1", ylab = "x2", col = cols, axes=F, plot.axes = { points(xin[,1], xin[,2],col
=data$classes, xlim=c(-5,5), ylim=c(-5,5)); axis(1); axis(2) })
}
```

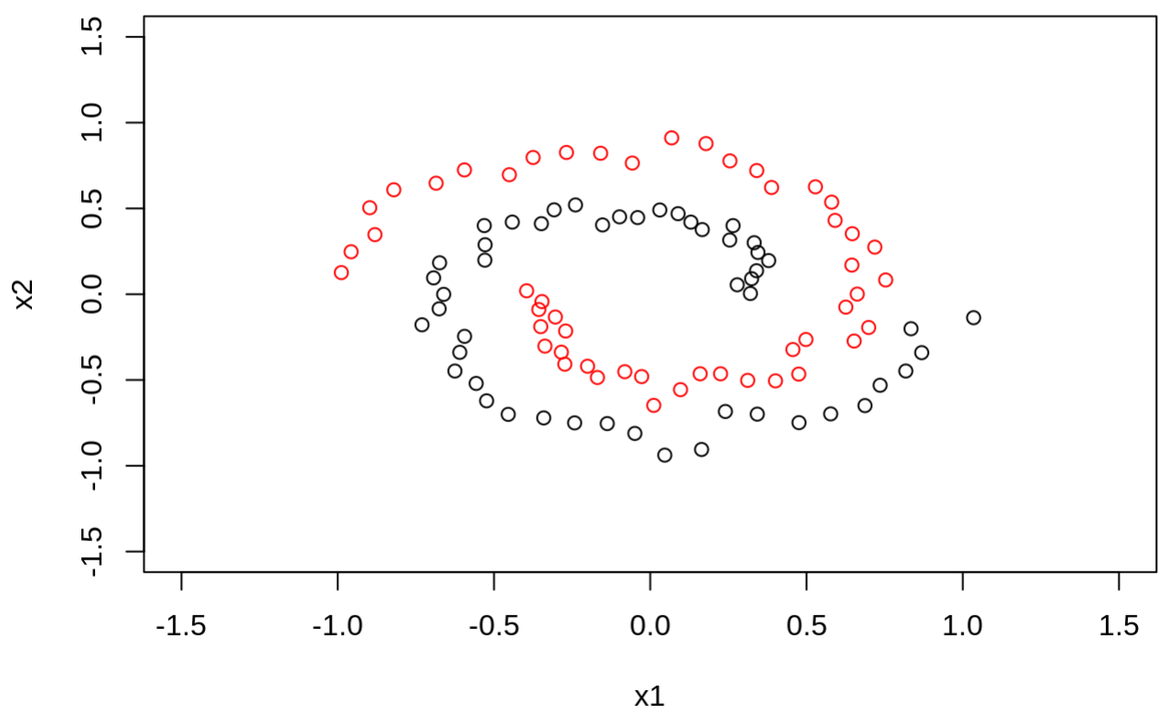
```
## [1] "Classificação utilizando valor k =  15"
```



Parte 1 - Base de dados mlbench.spirals

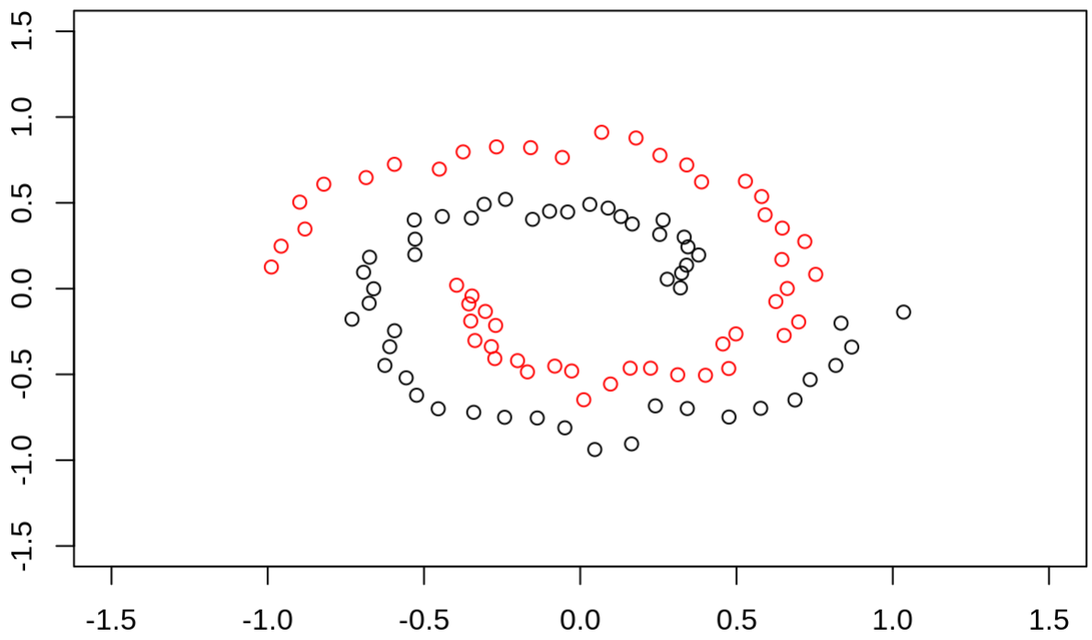
Por fim, para a base de dados dados mlbench.spirals utilizou-se os valores de k como 15, 20 e 25. Isso foi feito afim de se poder observar um solução com underfitting (para k=15), uma solução visualmente boa (para k = 20) e outra solução que já apresenta sinais de overfitting (para k=25). As superfícies de separação obtidas para cada uma das 3 redes serão mostradas abaixo.

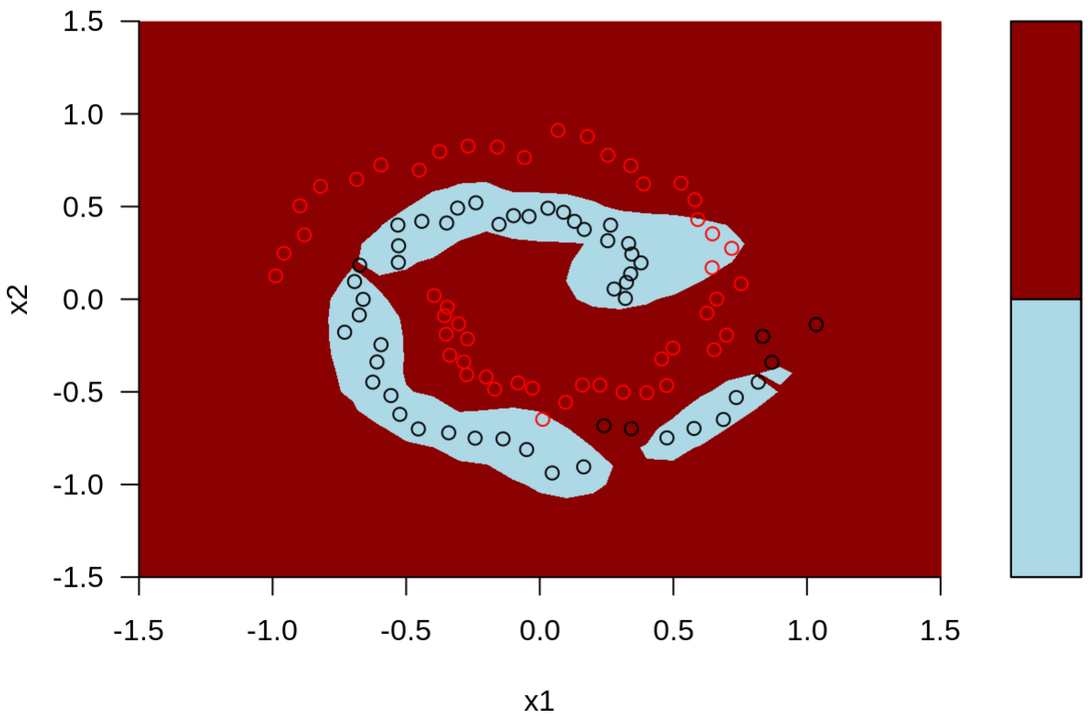
```
rm(list = ls())
source(file = "/home/vitor/Documents/UFMG/9/Redes Neurais/exemplos/trainRBF.R")
source(file = "/home/vitor/Documents/UFMG/9/Redes Neurais/exemplos/YRBF.R")
library(mlbench)

data <- mlbench.spirals(100,sd = 0.05)

xin<-matrix(data[["x"]], ncol = 2)
class<-matrix(data[["classes"]], ncol=1)
plot(xin[,1], xin[,2], col=data$classes, xlim=c(-1.5, 1.5), ylim=c(-1.5, 1.5), ylab = "x
2", xlab = "x1")
```



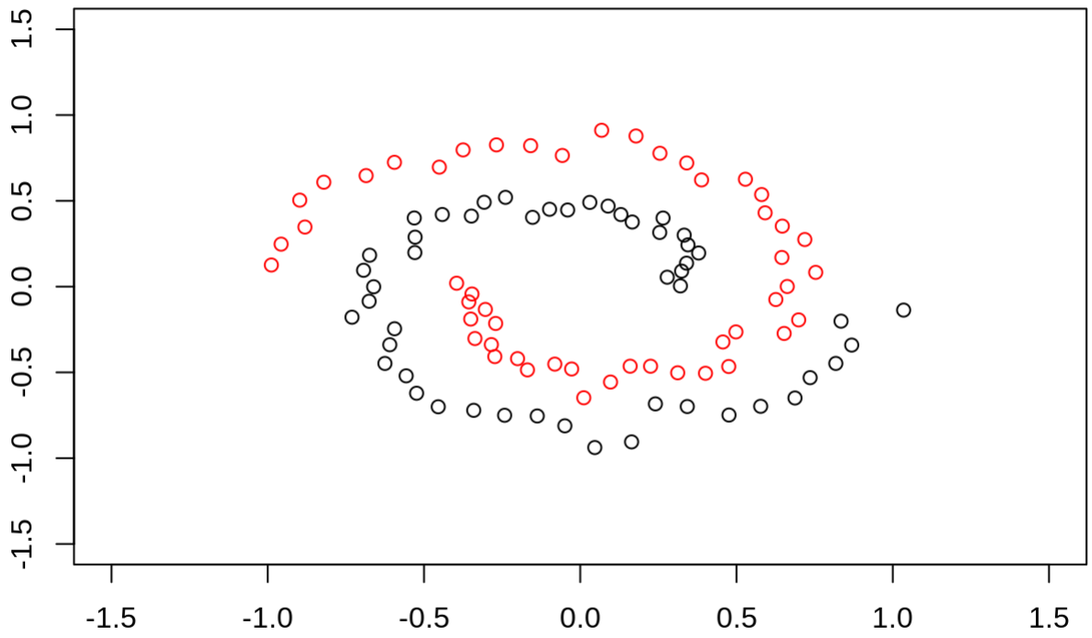
```
yin <- rep(0,100)
for (count in 1:length(class)) {
  if (class[count] == 1 ){
    yin[count] = -1
  }
  else if(class[count] == 2){
    yin[count] = 1
  }
}

pvector<-c(15,20,25)
for (p in pvector) {
  modRBF<-trainRBF(xin,yin,p)

  # Plotando superfície
  seqx1x2 <- seq(-5, 5, 0.1)
  MZ<-matrix(nrow = length(seqx1x2), ncol = length(seqx1x2))
  cr<-0
  for (i in 1:length(seqx1x2)) {
    for (j in 1:length(seqx1x2)) {
      cr<-cr+1
      x1<-seqx1x2[i]
      x2<-seqx1x2[j]
      x1x2<-matrix(cbind(x1,x2), nrow = 1)
      MZ[i,j]<-YRBF(x1x2, modRBF)
    }
  }

  #contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-5,5), ylim=c(-5,5), xlab = "x1", y
lab = "x2")
  #par(new=T)
  print(paste("Classificação utilizando valor k = ", p))
  plot(xin[,1], xin[,2], col=data$classes, xlim=c(-1.5,1.5), ylim=c(-1.5,1.5), ylab = ""
, xlab = "")
  cols = c('lightblue', 'darkred')
  filled.contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-1.5,1.5), ylim=c(-1.5,1.5),
 xlab = "x1", ylab = "x2", col = cols, axes=F, plot.axes = { points(xin[,1], xin[,2],col
=data$classes, xlim=c(-5,5), ylim=c(-5,5)); axis(1); axis(2) })
}
```

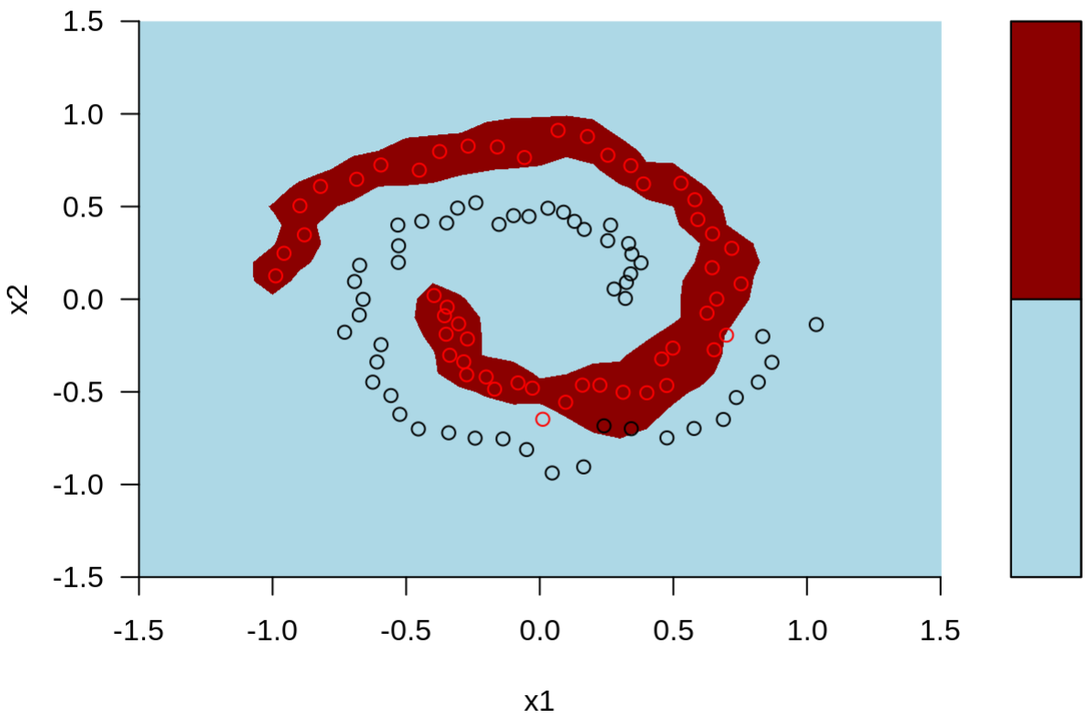
```
## [1] "Classificação utilizando valor k =  15"
```



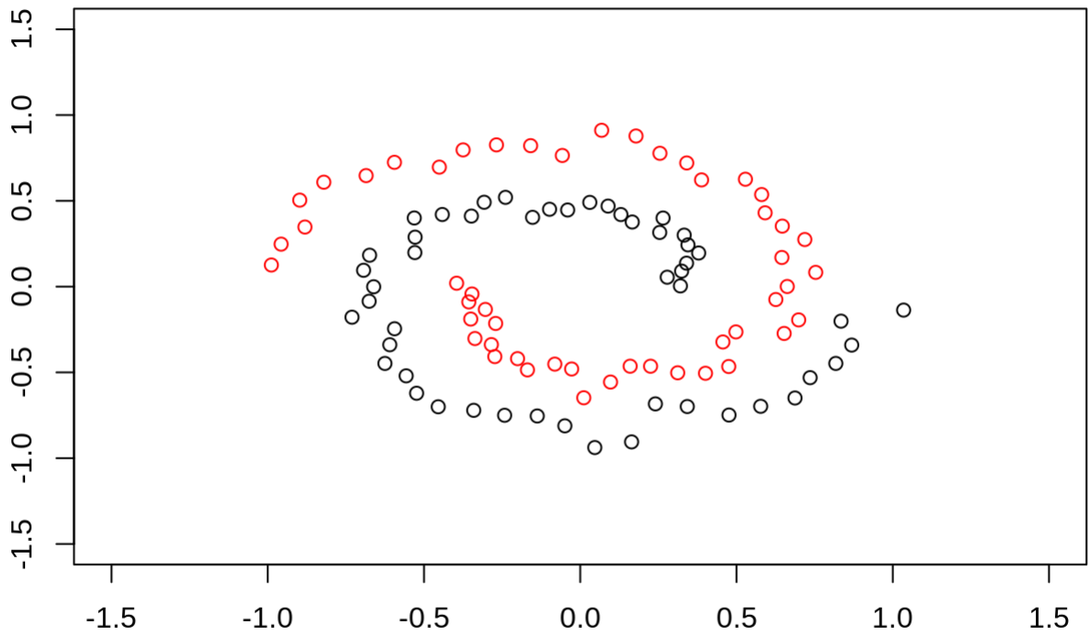


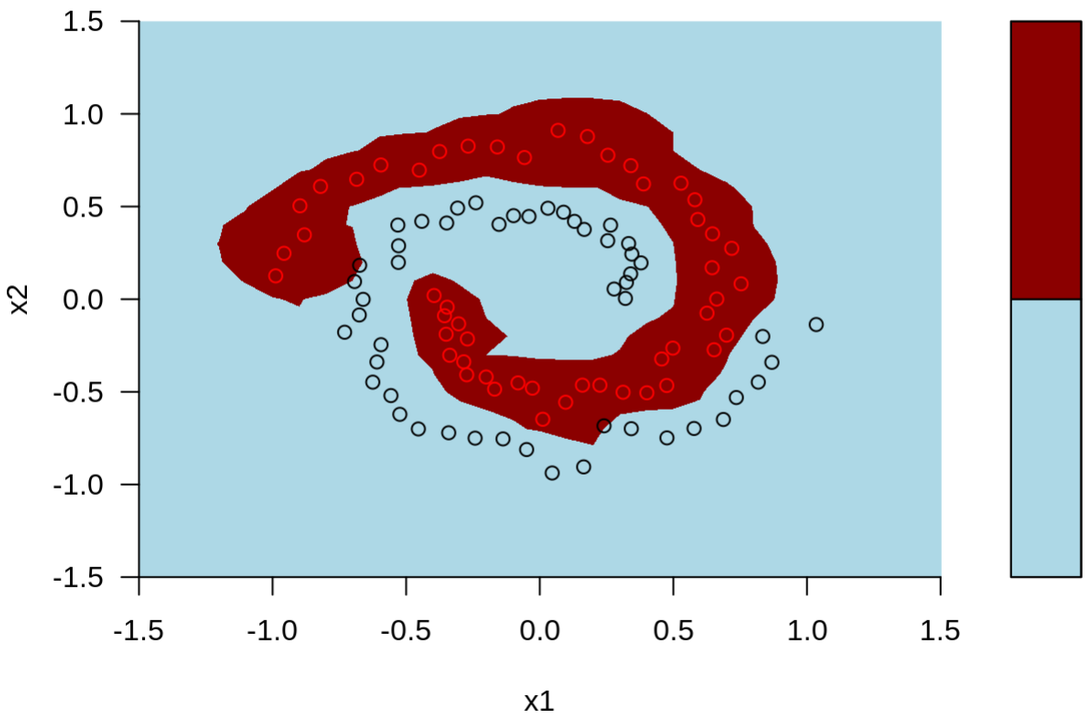

```
## [1] "Classificação utilizando valor k =  20"
```






```
## [1] "Classificação utilizando valor k =  25"
```





Parte 2 - Aproximação da função sinc

Como pode ser visto abaixo construiu-se uma rede RBF para aproximar a função sinc acrescida de um ruído gaussiano. Assim como feito para a primeira parte, foram ajustadas, 3 redes RBF, com diferentes números de centros (3, 10 e 15). As 3 redes tiveram seu desempenho comparado em um segundo conjunto de 50 amostras (gerado da mesma forma que o primeiro). A métrica usada foi o erro quadrático médio

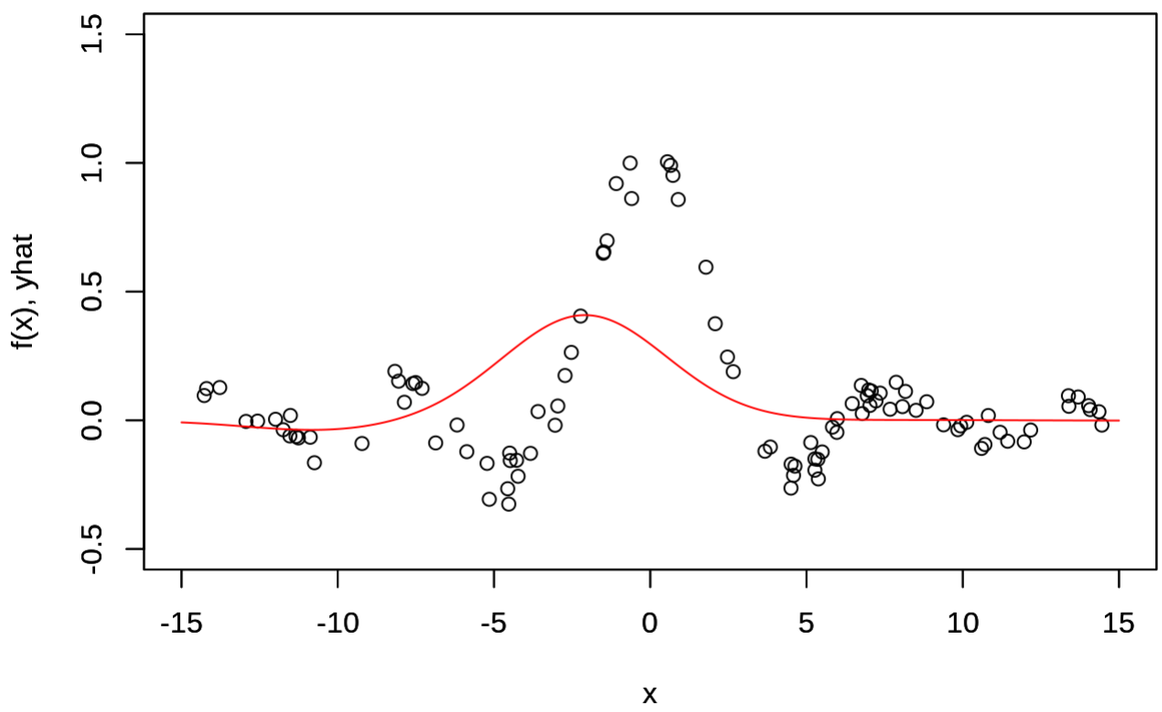
```
rm(list = ls())
source(file = "/home/vitor/Documents/UFMG/9/Redes Neurais/exemplos/trainRBF.R")
source(file = "/home/vitor/Documents/UFMG/9/Redes Neurais/exemplos/YRBF.R")

x <- matrix(runif (100, -15, 15), ncol=1)
y <- matrix(sin(x)/x + rnorm(100, 0, 0.05), ncol=1)

x_test <- matrix(runif (50, -15, 15), ncol=1)
y_test <- matrix(sin(x_test)/x_test + rnorm(50, 0, 0.05), ncol=1)

pvector<-c(3,8,15)
for (p in pvector) {
  modRBF <- trainRBF(x, y, p)
  yhat_test <- YRBF(x_test, modRBF)
  mse <- sum((yhat_test - y_test)^2)/length(y_test)
  print(paste("Erro quadrático médio: MSE = ", mse))
}
```

```
## [1] "Erro quadrático médio: MSE =  0.03624870165011"
## [1] "Erro quadrático médio: MSE =  0.0191921533581178"
## [1] "Erro quadrático médio: MSE =  0.013720920573641"
```

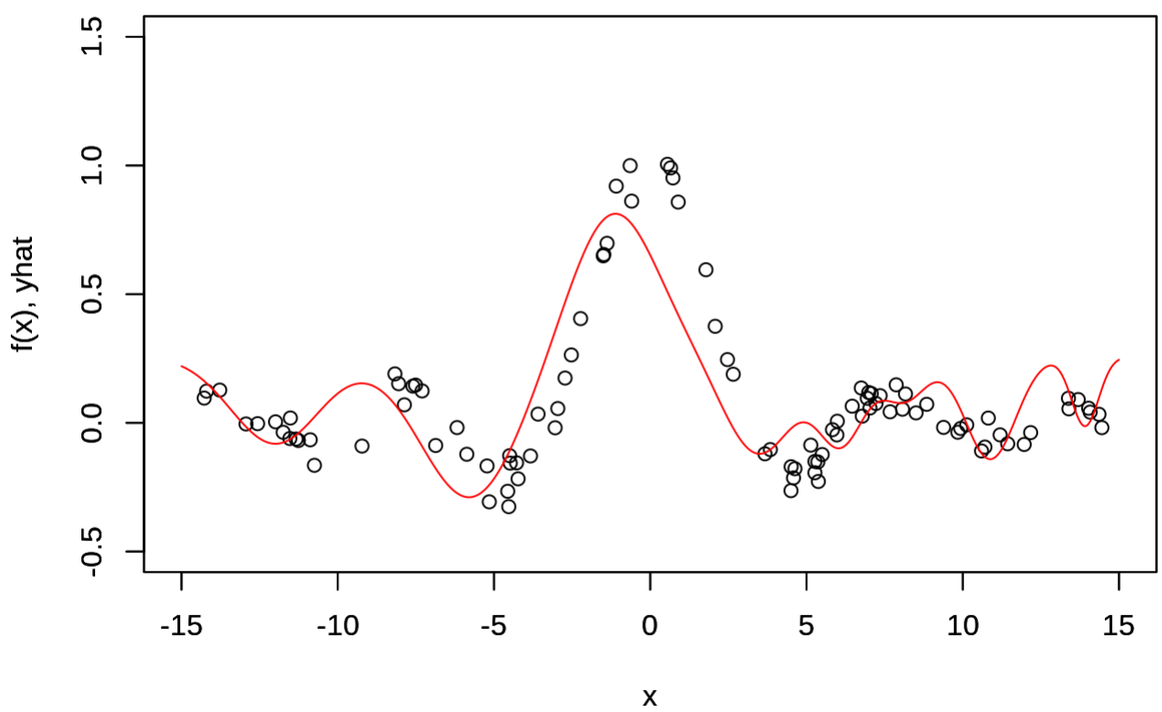
Afim de realizar uma observação mais visual, foi ainda plotado as curvas resultantes de cada modelo (para k=3, 8 e 15) para valores de x entre -15 e 15.

```
rm(list = ls())
source(file = "/home/vitor/Documents/UFMG/9/Redes Neurais/exemplos/trainRBF.R")
source(file = "/home/vitor/Documents/UFMG/9/Redes Neurais/exemplos/YRBF.R")

x <- matrix(runif (100, -15, 15), ncol=1)
y <- matrix(sin(x)/x + rnorm(100, 0, 0.05), ncol=1)

pvector<-c(3,8,15)
for (p in pvector) {
  modRBF <- trainRBF(x, y, p)
  xrange <- matrix(seq(-15, 15, 0.01), ncol = 1)
  yhat <- YRBF(xrange, modRBF)

  ############## Plots: ###############
  print(paste("Aproximação utilizando valor k = ", p))
  plot(x, y, xlim = c(-15, 15), ylim = c(-0.5,1.5), xlab = "x", ylab = "f(x), yhat", col
 = "black")
  par(new=T)
  plot(xrange, yhat, xlim = c(-15, 15), ylim = c(-0.5,1.5), xlab = "x", ylab = "f(x), yh
at", col = "red", type = "l")
}
```

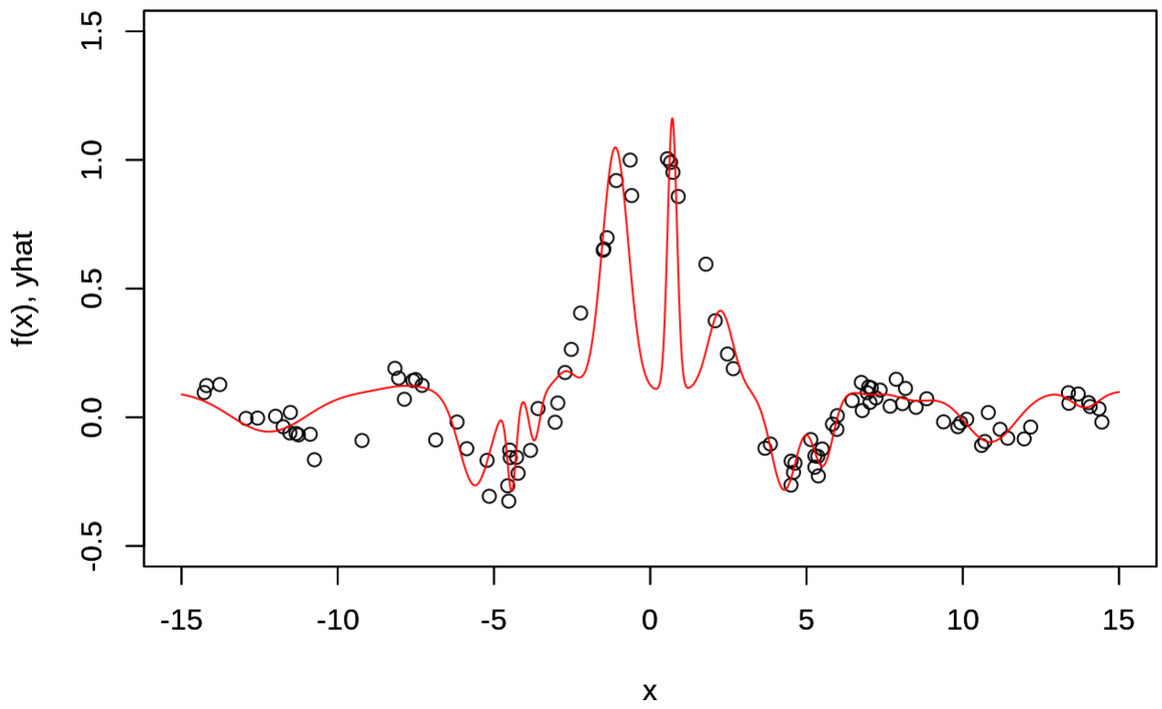
```
## [1] "Aproximação utilizando valor k =  3"
```




```
## [1] "Aproximação utilizando valor k =  8"
```




```
## [1] "Aproximação utilizando valor k =  15"
```


f(x), yhat
1.5
1.0
0.5
0.0
-0.5
-15
-10
-5
0
5
10
15
x