Universidade Federal de Minas Gerais
Aluno: Giovanni Martins de Sá Júnior
Matrícula: 2017001850

Exercício 7: Redes Neurais Artificiais

Este exercício tem como base implementar uma rede neural RBF com o objetivo de classificar regiões a partir de centros e raios. As bases de dados usadas foram:

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

Para as implementações a seguir, foram gerados vários plots variando o número de centros (k) a partir de um for para que assumisse os valores 5, 10, 15, 20, 25 e 30. As funções usadas nas soluções de cada base de dados é mostrado a seguir:

```
install.packages("mlbench")
install.packages("kernlab")
install.packages("MLmetrics")

library("mlbench")
library("kernlab")
library("MLmetrics")
library("RSNNS")

rm(list=ls())

YRBF <- function(xin, modRBF){
  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]
  n <- dim(xin)[2]
  m <- as.matrix(modRBF[[1]])
  covlist <- modRBF[[2]]
  p <- length(covlist)
  W <- modRBF [[3]]
  xin <- as.matrix(xin)

  H <- matrix(nrow = N, ncol = p)
  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)
}

trainRBF <- function(xin, yin, p){
  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]
n<-dim(xin)[2]
xin <- as.matrix(xin)
yin <- as.matrix(yin)
xclust<-kmeans(xin, p)
m <- as.matrix (xclust$centers)
covlist <- list()

for ( i in 1:p)
{
  ici <- which(xclust$cluster == i )
  xci <- xin [ici, ]
  if(n==1){
    covi <- var(xci)
  }
  else{
    row <- dim(xci)[1];
    if(is.null(row)){
      row <- 0
    }
    if(row > 1){
      covi <- cov(xci)
    }
    else{
      covi <- cov(matrix(c(xci, xci), nrow = 2))
    }
  }
  covlist [[i]] <- covi
}

H <- matrix(nrow = N, ncol = p)
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 <- (solve(t(Haug) %*% Haug)%*% t(Haug))%*% yin
return (list(m, covlist, W, H))
}
```

Caso 1: mlbench.2dnormals

Para este caso, o código usado pode ser visualizado abaixo:

```
acuracia<-c()
kvet<-seq(5, 30, 5)

for (kfor in kvet)
{
  for(i in 1:10){
    normals=mlbench.2dnormals(200)
    normals=cbind(normals[[1]], normals[[2]])
    normals[,3]=(normals[,3]-1.5)*2
    normals_all<-splitForTrainingAndTest(normals[,1:2],normals[,3],ratio=0.3)
    xin_normals<-normals_all$inputsTrain
    yd_normals<-normals_all$targetsTrain
    xin_test_normals<-normals_all$inputsTest
    ytest<-normals_all$targetsTest

    retlist<-trainRBF(xin_normals, yd_normals, kfor)
    yhat_test<-sign(YRBF(xin_test_normals, retlist))
    acuracia[i]<-Accuracy(ytest, yhat_test)
    yhat_train<-sign(YRBF(xin_normals, retlist))
    acuracia_train<-Accuracy(yd_normals, yhat_train)
  }

  mean(acuracia)
  sd(acuracia)
  plot(normals, xlim=c(-3,3), ylim=c(-4,4))
  seq_i<-seq(-4,4,0.2)
  seq_j<-seq(-3,3,0.15)
  Mat<-matrix(0,nrow=length(seq_i), ncol=length(seq_j))
  c_i<-0

  for(i in seq_i){
    c_i<-c_i+1
    c_j<-0
    for(j in seq_j)
    {
      c_j<-c_j+1
      x <- as.matrix(t(c(i,j)))
      Mat[c_i,c_j]<-sign(YRBF(x,retlist))
    }
  }
```

```
for(i in 1:length(seq_i)){
  for(j in 1:length(seq_j))
  {
    if(Mat[i,j]==1){
      points(seq_j[i],seq_i[j],pch=4,col="cornsilk3")
    }
    else{
      points(seq_j[i],seq_i[j],pch=4,col="darkseagreen1")
    }
  }
}
for(i in 1:200){
  if(normals[i,3]==1){
    points(normals[i,1],normals[i,2],col="blue")
  }
  if(normals[i,3]==-1){
    points(normals[i,1],normals[i,2],col="red")
  }
}
```

Usando k=5, a superfície de separação foi obtida como mostrado na figura abaixo:

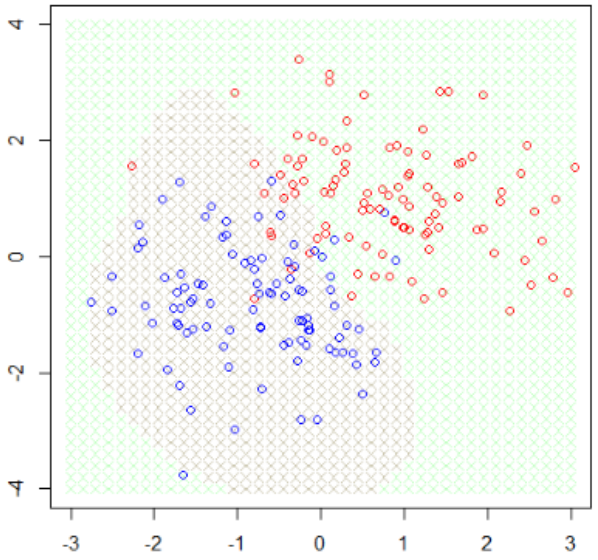


Figura 1.1. k = 5

Para k=10, a superfície se configura como mostrado a seguir:

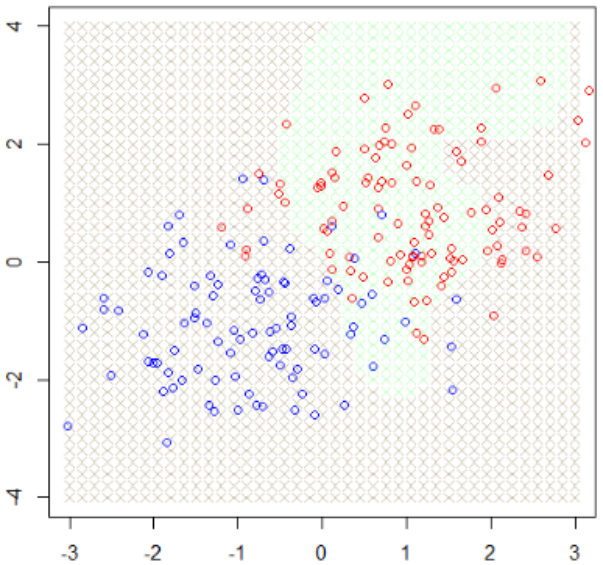


Figura 1.2. k = 10

Para k=15:

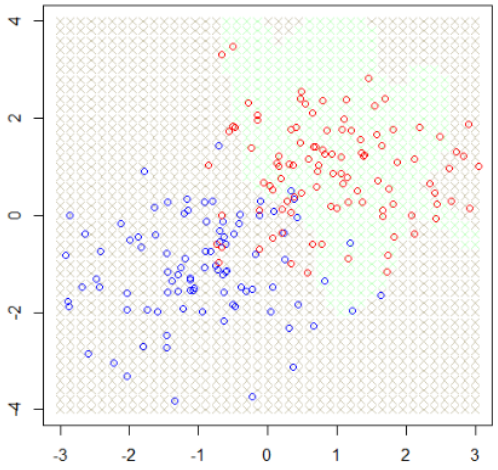


Figura 1.3. k = 15

Para k=20:

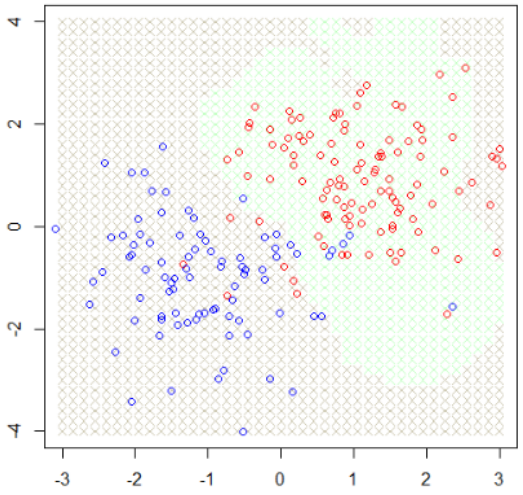


Figura 1.4. k = 20

Para k = 25:

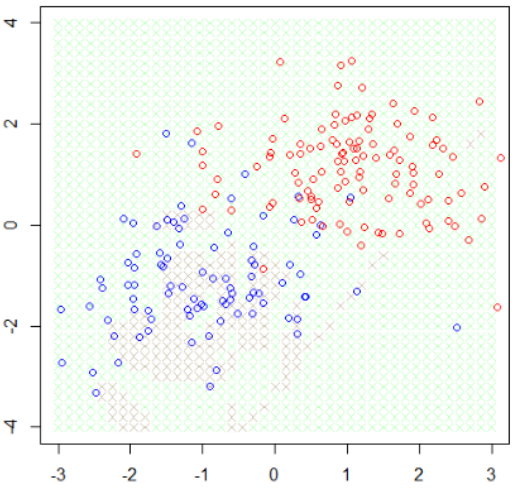


Figura 1.5. k = 25

Para k = 30:

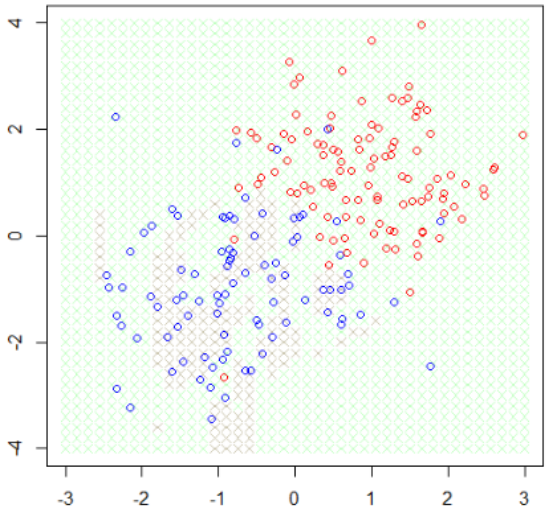


FIgura 1.6. k = 30

Analisando então as imagens acima, é notável que conforme o número de centros a solução tende a piorar, isso é devido a simplicidade da solução e ao aumentar o valor de 'k' a solução tende a apresentar overfitting, ou seja, se aproximar do erro e distanciar da solução desejada, neste caso a melhor representação foi com k=5.

Caso 2: mlbench.xor

Para este caso, o código usado pode ser visualizado abaixo:

```
acuracia<-c()
kvet<-seq(5, 30, 5)
for (kfor in kvet)
{
  for(i in 1:10){
    xor=mlbench.xor(100)
    xor=cbind(xor[[1]], xor[[2]])
    xor[,3]=(xor[,3]-1.5)*2
    xor_all<-splitForTrainingAndTest(xor[,1:2],xor[,3],ratio=0.3)
    xin_xor<-xor_all$inputsTrain
    yd_xor<-xor_all$targetsTrain
    xin_test_xor<-xor_all$inputsTest
    ytest<-xor_all$targetsTest

    retlist<-trainRBF(xin_xor, yd_xor, kfor)
    yhat_test<-sign(YRBF(xin_test_xor, retlist))
    acuracia[i]<-Accuracy(ytest, yhat_test)
    yhat_train<-sign(YRBF(xin_xor, retlist))
    acuracia_train<-Accuracy(yd_xor, yhat_train)
  }
  mean(acuracia)
  sd(acuracia)
  plot(xor, xlim=c(-1,1), ylim=c(-1,1))
  seq_i<-seq(-1,1,0.05)
  seq_j<-seq(-1,1,0.05)
  Mat<-matrix(0,nrow=length(seq_i), ncol=length(seq_j))
  c_i<-0

  for(i in seq_i){
    c_i<-c_i+1
    c_j<-0
    for(j in seq_j)
    {
      c_j<-c_j+1
      x <- as.matrix(t(c(i,j)))
      Mat[c_i,c_j]<-sign(YRBF(x,retlist))
    }
  }
```

```
  for(i in 1:length(seq_i)){
    for(j in 1:length(seq_j))
    {
      if(Mat[i,j]==1){
        points(seq_j[i],seq_i[j],pch=4,col="cornsilk3")
      }
      else{
        points(seq_j[i],seq_i[j],pch=4,col="darkseagreen1")
      }
    }
  }
  for(i in 1:100){
    if(normals[i,3]==1){
      points(xor[i,1],xor[i,2],col="blue")
    }
    if(normals[i,3]==-1){
      points(xor[i,1],xor[i,2],col="red")
    }
  }
}
```

Usando k=5, a superfície de separação foi obtida como mostrado na figura abaixo:

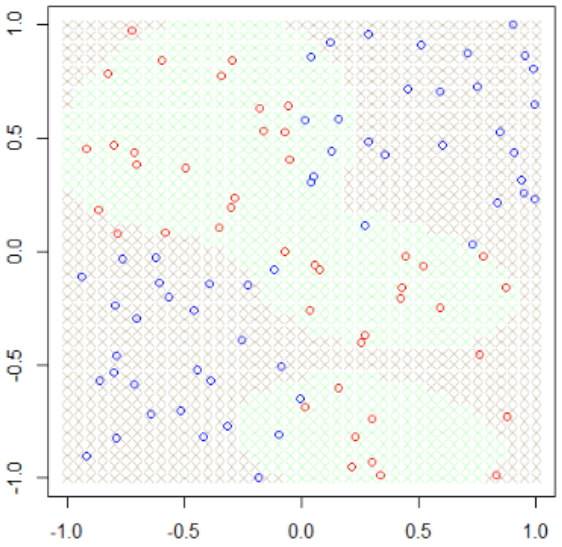


Figura 2.1. k = 5

Para k=10, a superfície se configura como mostrado a seguir:

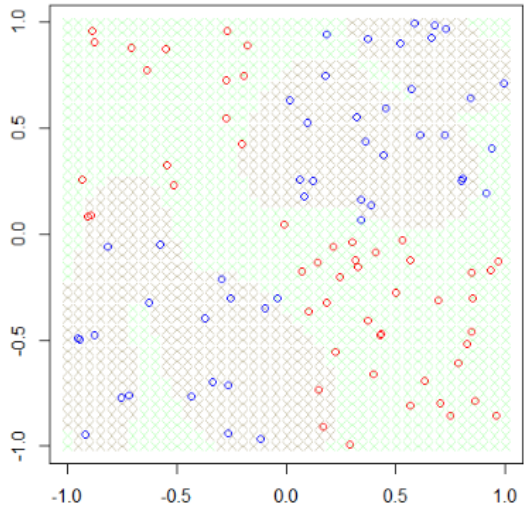


Figura 2.2. k = 10

Para k=15:

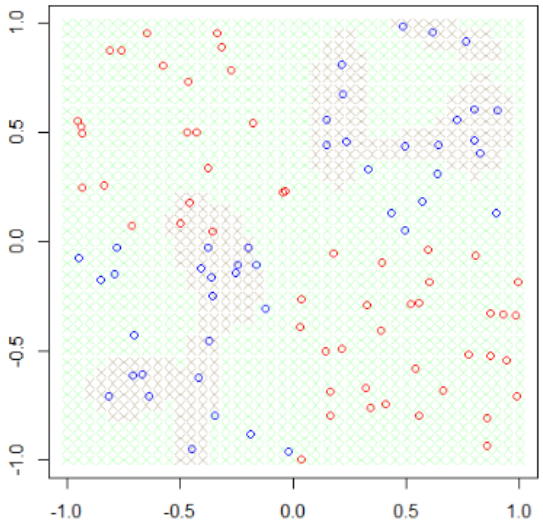


Figura 2.3. k = 15

Para k=20:

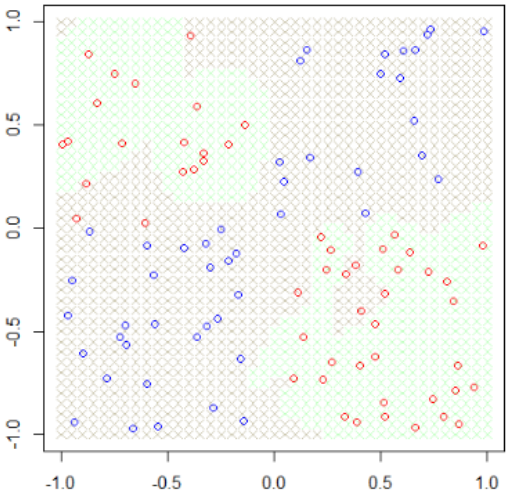


Figura 2.4. k = 20

Para k=25:

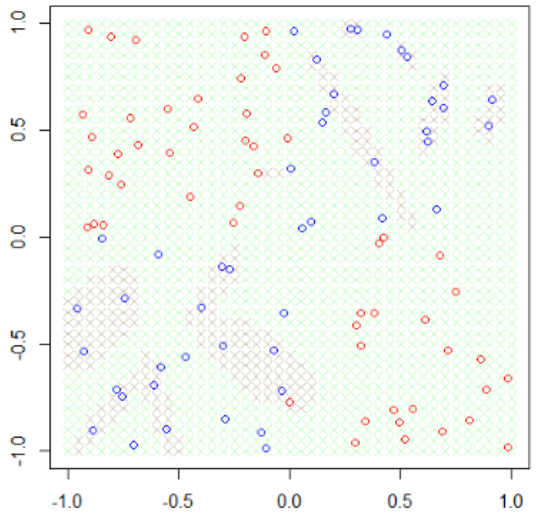


Figura 2.5. k = 25

Para k=30:

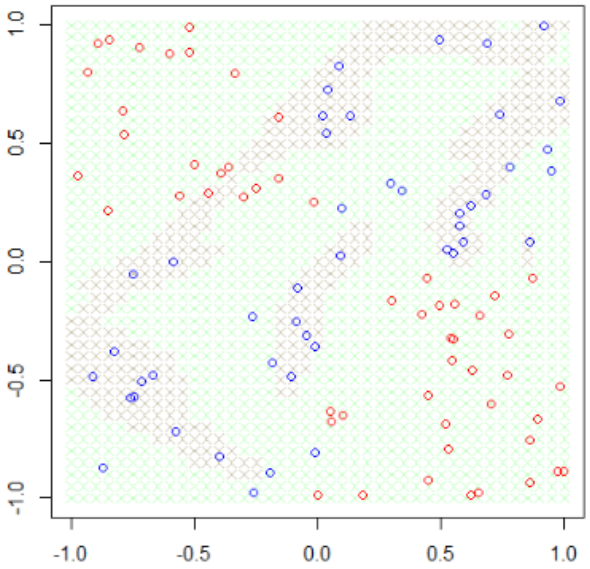


Figura 2.6. k = 30

Assim como no caso anterior, ao aumentar o número de centros a solução tende a piorar pois vai apresentando overfitting. Porém neste caso, por se tratar de um problema mais complexo que o anterior, para se obter uma melhor solução é necessário um número de centros maiores, sendo k=20 a melhor solução.

Caso 3: mlbench.circle

Para este caso, o código usado pode ser visualizado abaixo:

```
for (kfor in kvet)
{
  for(i in 1:10){
    circle=mlbench.circle(100)
    circle=cbind(circle[[1]], circle[[2]])
    circle[,3]=(circle[,3]-1.5)*2
    circle_all<-splitForTrainingAndTest(circle[,1:2],circle[,3],ratio=0.3)
    xin_circle<-circle_all$inputsTrain
    yd_circle<-circle_all$targetsTrain
    xin_test_circle<-circle_all$inputsTest
    ytest<-circle_all$targetsTest

    retlist<-trainRBF(xin_circle, yd_circle, kfor)
    yhat_test<-sign(YRBF(xin_test_circle, retlist))
    acuracia[i]<-Accuracy(ytest, yhat_test)
    yhat_train<-sign(YRBF(xin_circle, retlist))
    acuracia_train<-Accuracy(yd_circle, yhat_train)
  }
  mean(acuracia)
  sd(acuracia)
  plot(circle, xlim=c(-1,1), ylim=c(-1,1))
  seq_i<-seq(-1,1,0.05)
  seq_j<-seq(-1,1,0.05)
  Mat<-matrix(0,nrow=length(seq_i), ncol=length(seq_j))
  c_i<-0
  for(i in seq_i){
    c_i<-c_i+1
    c_j<-0
    for(j in seq_j)
    {
      c_j<-c_j+1
      x <- as.matrix(t(c(i,j)))
      Mat[c_i,c_j]<-sign(YRBF(x,retlist))
    }
  }
```

```
  for(i in 1:length(seq_i)){
    for(j in 1:length(seq_j))
    {
      if(Mat[i,j]==1){
        points(seq_j[i],seq_i[j],pch=4,col="cornsilk3")
      }
      else{
        points(seq_j[i],seq_i[j],pch=4,col="darkseagreen1")
      }
    }
  }
  for(i in 1:100){
    if(circle[i,3]==1){
      points(circle[i,1],circle[i,2],col="blue")
    }
    if(circle[i,3]==-1){
      points(circle[i,1],circle[i,2],col="red")
    }
  }
}
```

Usando k=5, a superfície de separação foi obtida como mostrado na figura abaixo:

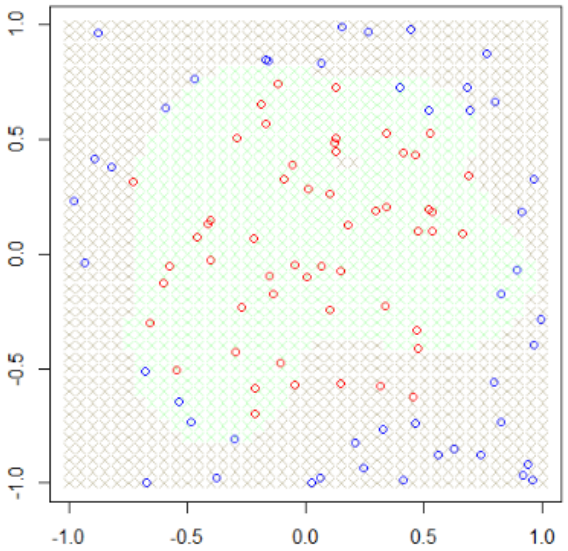


Figura 3.1. k=5

Para k=10, a superfície se configura como mostrado a seguir:

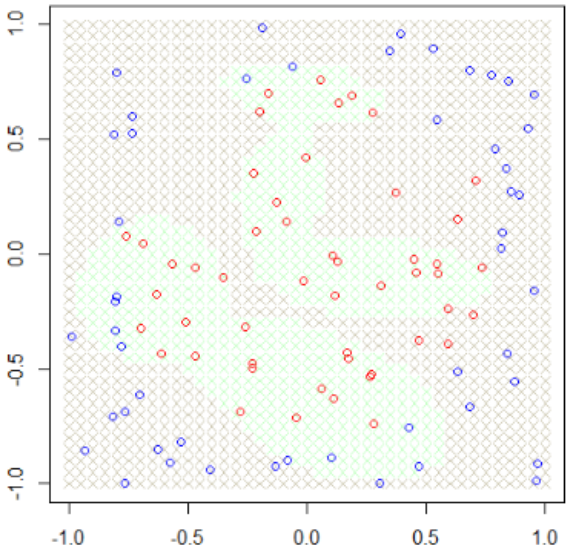


Figura 3.2. k=10

Para k=15:

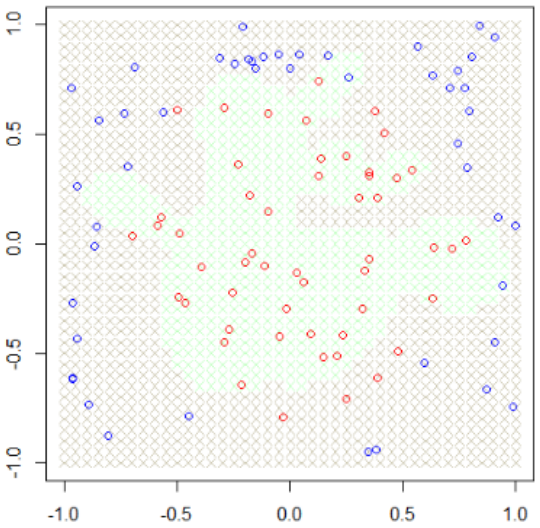


Figura 3.3. k=15

Para k=20:

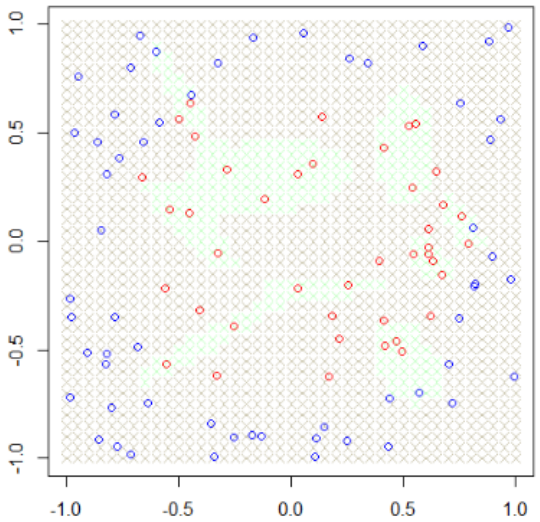


Figura 3.4. k= 20

Para k=25:

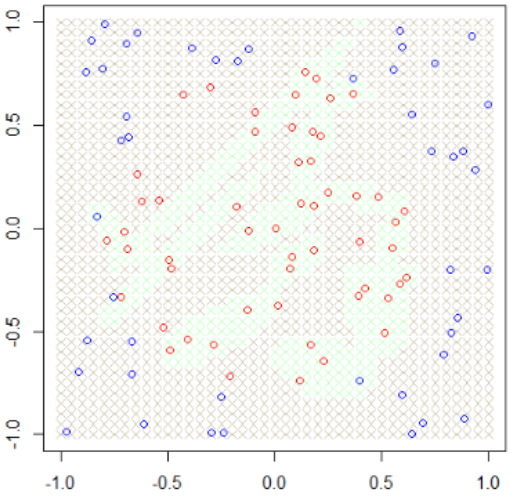


Figura 3.5. k= 25

Para k=30:

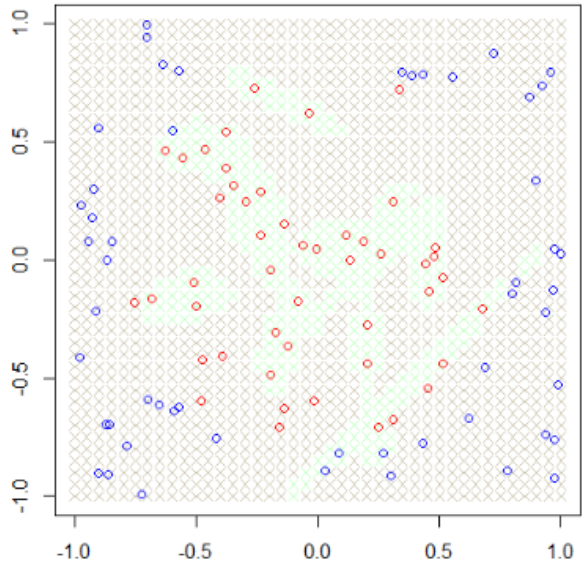


Figura 3.6. k= 30

Neste caso todas as soluções visualmente aparentam ser agradáveis no quesito de englobar os dados apresentados na região correta, porém conforme se aumenta o número de centros a solução tende a apresentar overfitting, com destaque para k=5 que apresenta ser uma solução menos complexa, bem definida e satisfatória.

Caso 4: mlbench.spirals

```
for (kfor in kvet)
{
  for(i in 1:10){
    spirals=mlbench.spirals(100)
    spirals=cbind(spirals[[1]], spirals[[2]])
    spirals[,3]=(spirals[,3]-1.5)*2
    spirals_all<-splitForTrainingAndTest(spirals[,1:2],spirals[,3],ratio=0.3)
    xin_spirals<-spirals_all$inputsTrain
    yd_spirals<-spirals_all$targetsTrain
    xin_test_spirals<-spirals_all$inputsTest
    ytest<-spirals_all$targetsTest

    retlist<-trainRBF(xin_spirals, yd_spirals, kfor)
    yhat_test<-sign(YRBF(xin_test_spirals, retlist))
    acuracia[i]<-Accuracy(ytest, yhat_test)
    yhat_train<-sign(YRBF(xin_spirals, retlist))
    acuracia_train<-Accuracy(yd_spirals, yhat_train)
  }
  mean(acuracia)
  sd(acuracia)
  plot(spirals, xlim=c(-1,1), ylim=c(-1,1))
  seq_i<-seq(-1,1,0.05)
  seq_j<-seq(-1,1,0.05)
  Mat<-matrix(0,nrow=length(seq_i), ncol=length(seq_j))
  c_i<-0
  for(i in seq_i){
    c_i<-c_i+1
    c_j<-0
    for(j in seq_j)
    {
      c_j<-c_j+1
      x <- as.matrix(t(c(i,j)))
      Mat[c_i,c_j]<-sign(YRBF(x,retlist))
    }
  }
```

```
  for(i in 1:length(seq_i)){
    for(j in 1:length(seq_j))
    {
      if(Mat[i,j]==1){
        points(seq_j[i],seq_i[j],pch=4,col="cornsilk3")
      }
      else{
        points(seq_j[i],seq_i[j],pch=4,col="darkseagreen1")
      }
    }
  }
  for(i in 1:100){
    if(spirals[i,3]==1){
      points(spirals[i,1],spirals[i,2],col="blue")
    }
    if(spirals[i,3]==-1){
      points(spirals[i,1],spirals[i,2],col="red")
    }
  }
}
```

Usando k=5, a superfície de separação foi obtida como mostrado na figura abaixo:

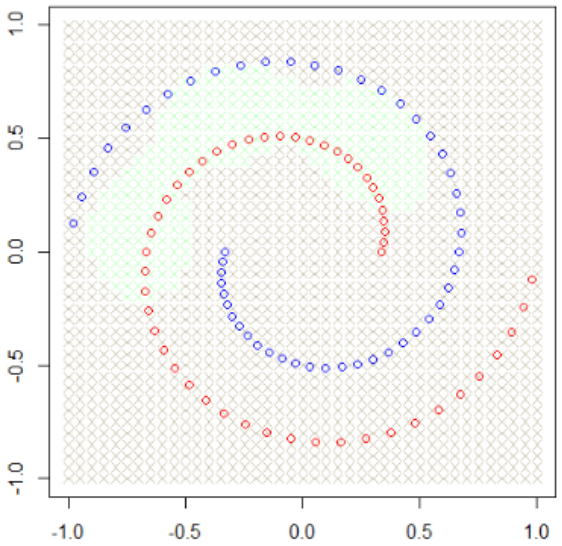


Figura 3.1. k = 5

Para k=10:

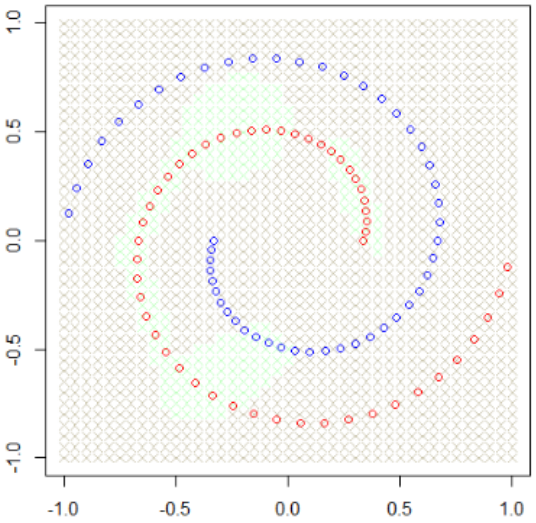


Figura 3.2. k = 5

Para k=15:

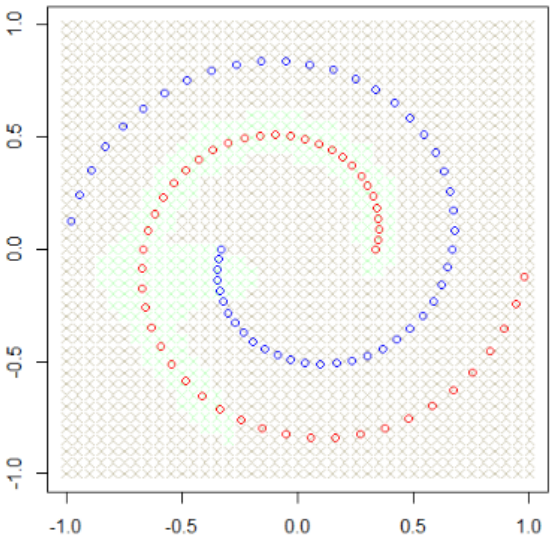


Figura 3.3. k = 10

Para k=20:

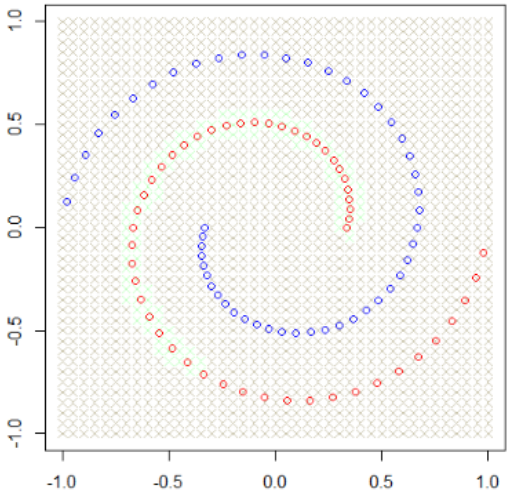


Figura 3.4. k = 20

Para k=25:

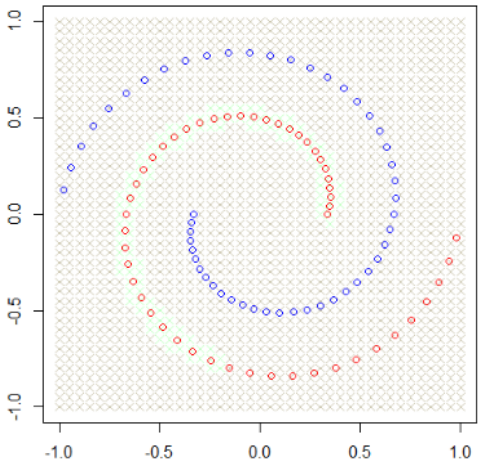


Figura 3.5. k = 25

Para k=30:

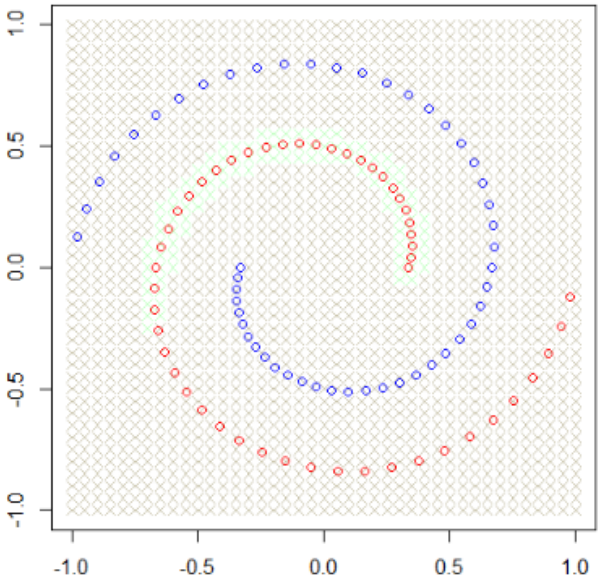


Figura 3.6. k = 30

Analisando então as imagens acima, é notável que devido a complexidade, redes com pouco número de centros têm dificuldade em resolver o problema, entretanto até os últimos casos sofreram com a representação mesmo apresentando um resultado melhor que os primeiros.

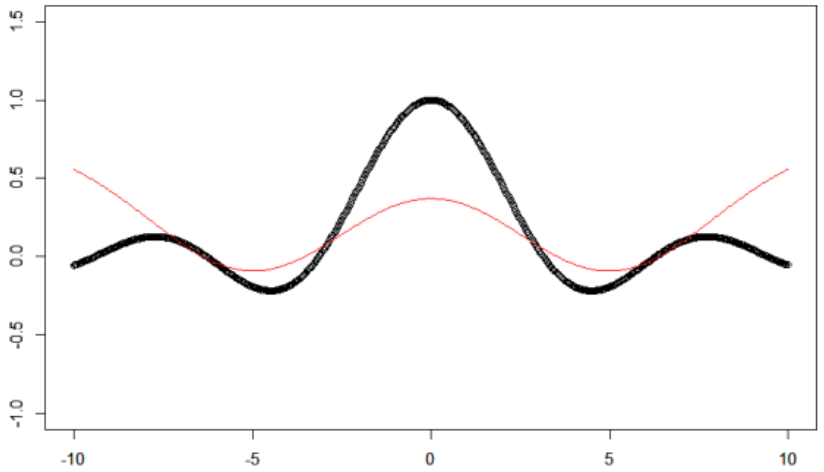
Parte 2: Aproximação da função sin(x)

Para este caso, o código usado pode ser visualizado abaixo:

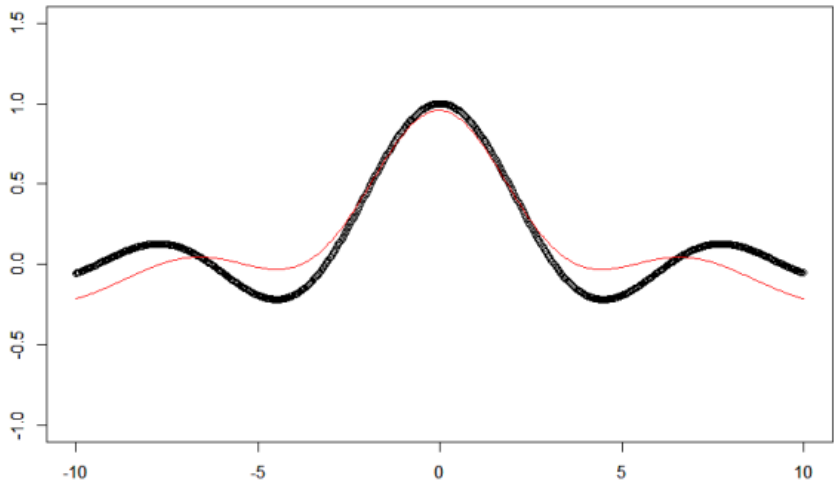
```
xin<-matrix(seq(-10.00001,10.00001,0.01*pi),ncol=1)
yin<-matrix((sin(xin)/xin),ncol=1)

modRBF<-trainRBF(xin,yin,5)
xrange<-matrix(seq(-10,10,0.01*pi),ncol=1)

yhat_teste<-YRBF(xrange,modRBF)

plot(xin,yin,type='b', xlim=c(-10,10),ylim=c(-1,1.5),xlab="x", ylab="f(x),yhat")
par(new=T)
plot(xrange,yhat_teste,col='red',type='l', xlim=c(-10,10),ylim=c(-1,1.5))
```

Sendo que o número de centros foi alterado de forma manual. Os resultados obtidos podem ser visualizado como mostrado abaixo:

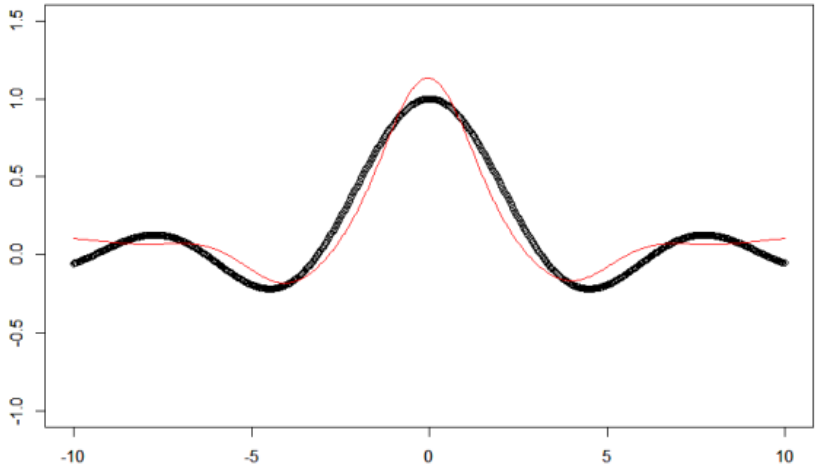
Número de centros = 2.



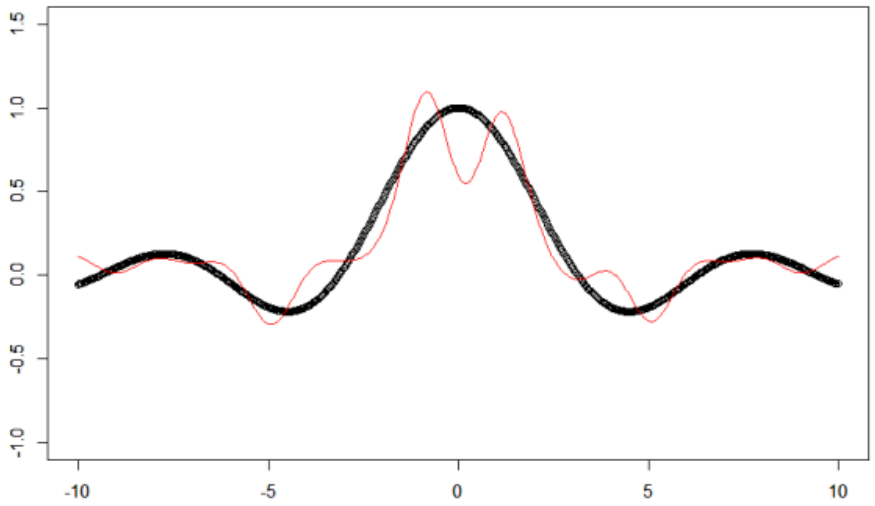
Número de centros = 3.



Número de centros = 5.



Número de centros = 10.



É notável que redes com número de centros pequenos sofre com a representação da função uma vez que se configura com uma complexidade maior (como foi muito bem representado no primeiro caso). Em contrapartida, redes com um número de centros muito grandes sofrem com problemas de overfitting (como mostrado no último caso). Logo é necessário saber escolher um valor que atenda a complexidade da rede porém não sofra com problemas de overfitting, que é o caso do segundo e do terceiro último caso.