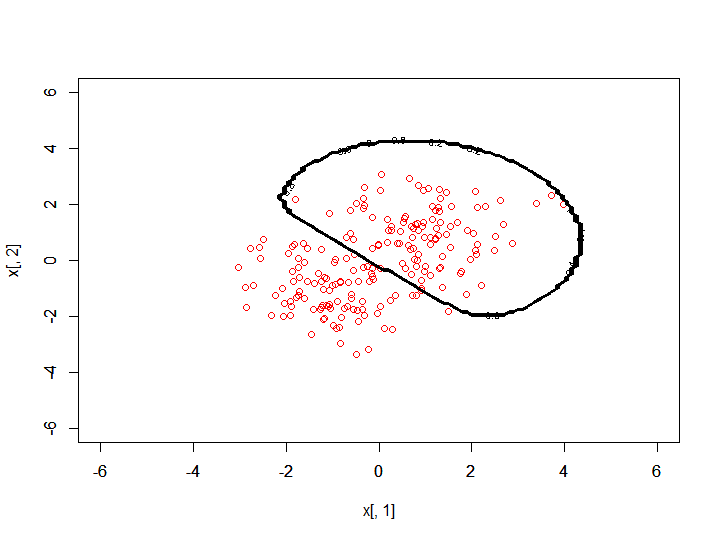
**Redes Neurais Artificiais**

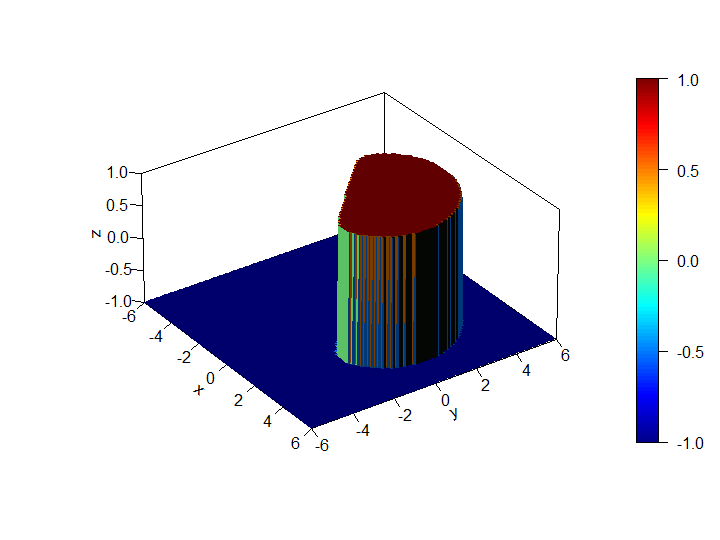
**André Costa Werneck, Matrícula: 2017088140 LISTA 7**

**31/05/2022**

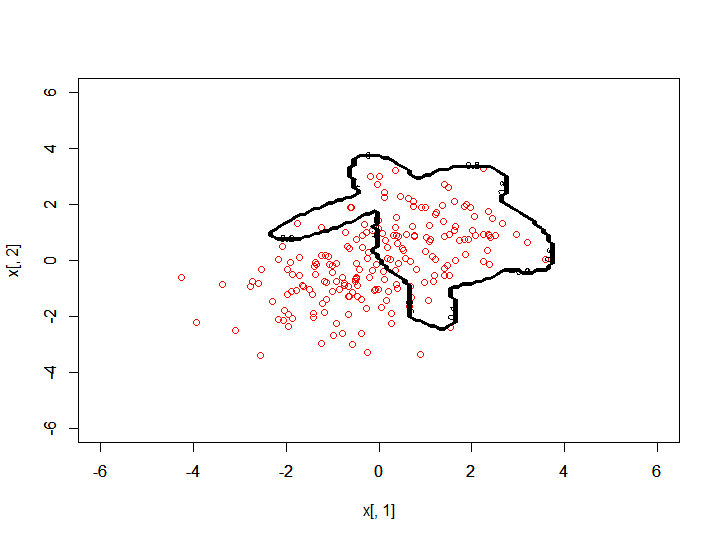
1. *Base 2dnormals e:*

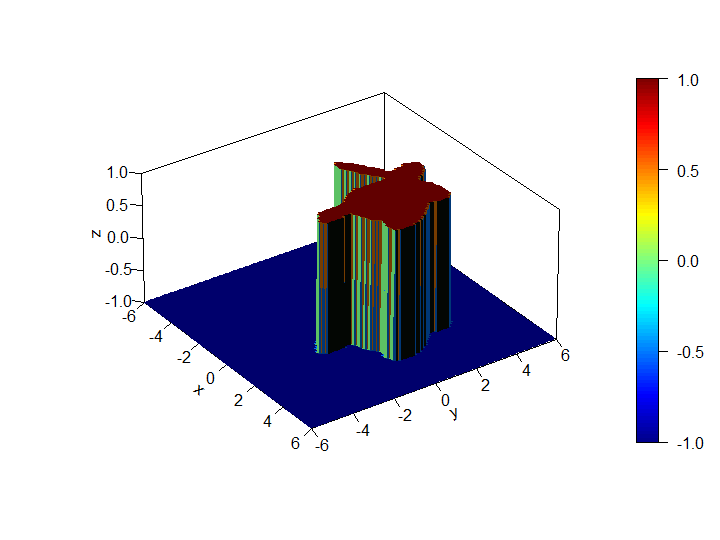
* *P = 2 centros*





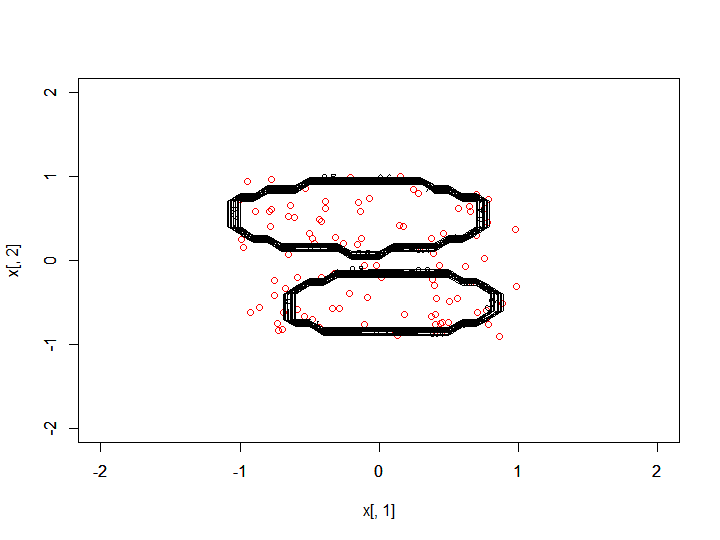
**P = 20**

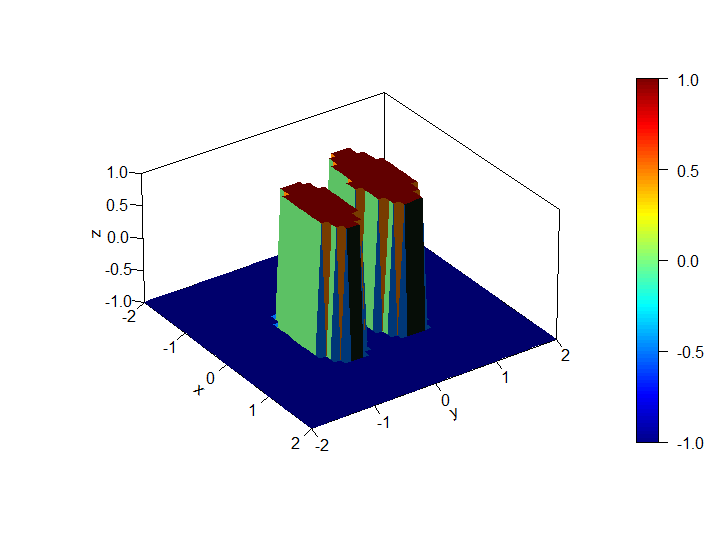




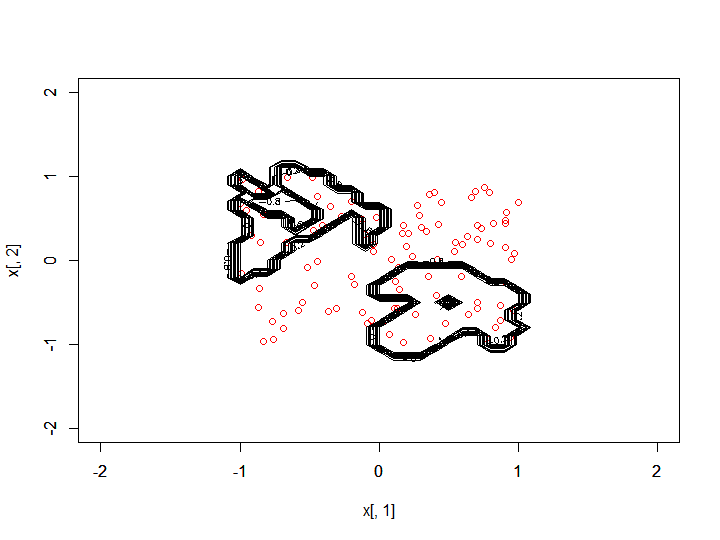
1. *Base xor e:*

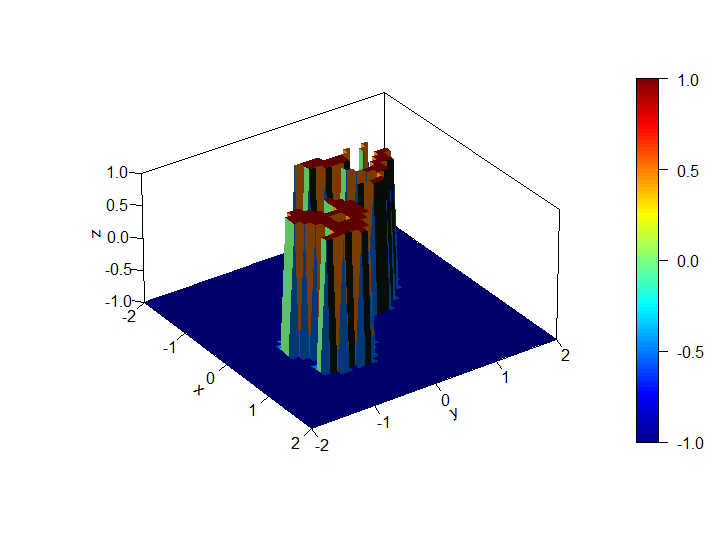
* *P = 2 centros*





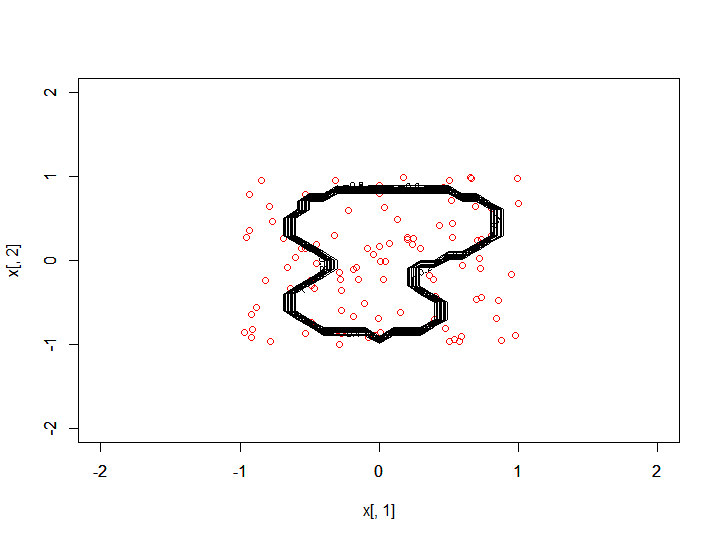
**P = 20**

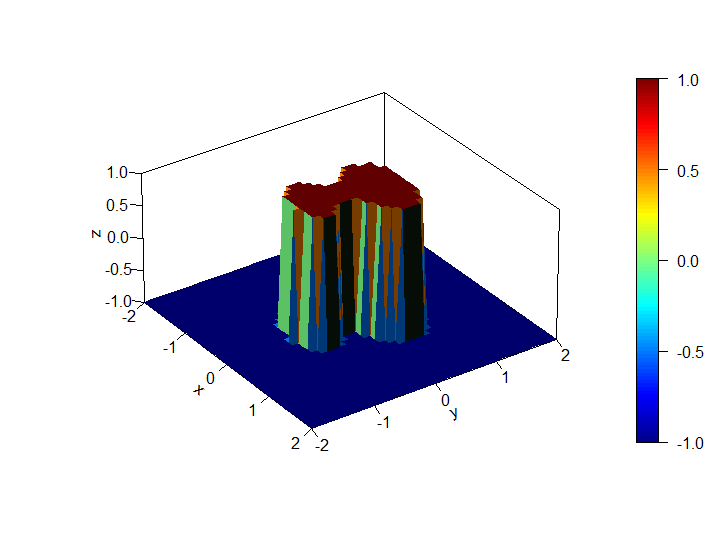




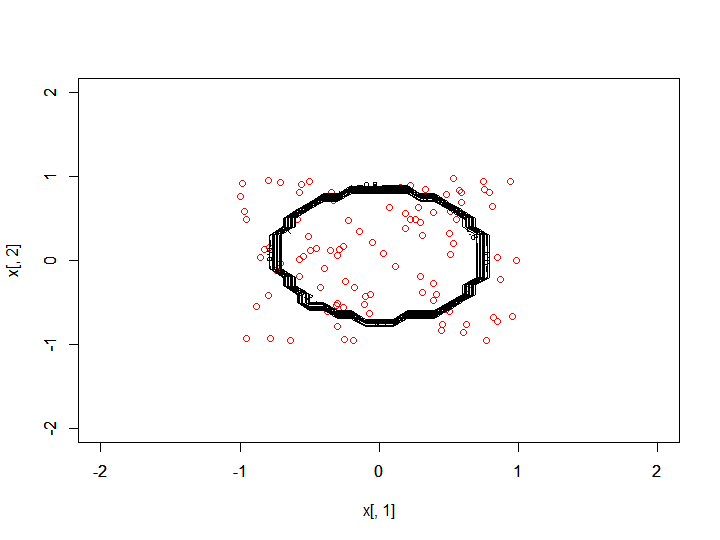
1. *Base circle e:*

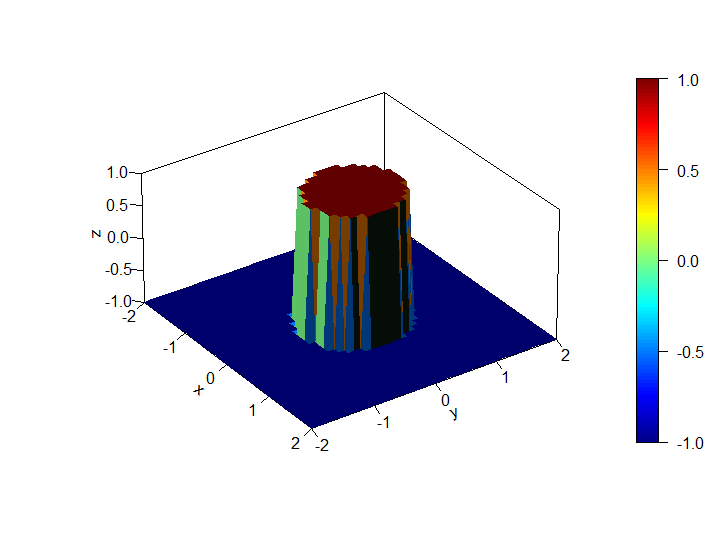
* *P = 2 centros*





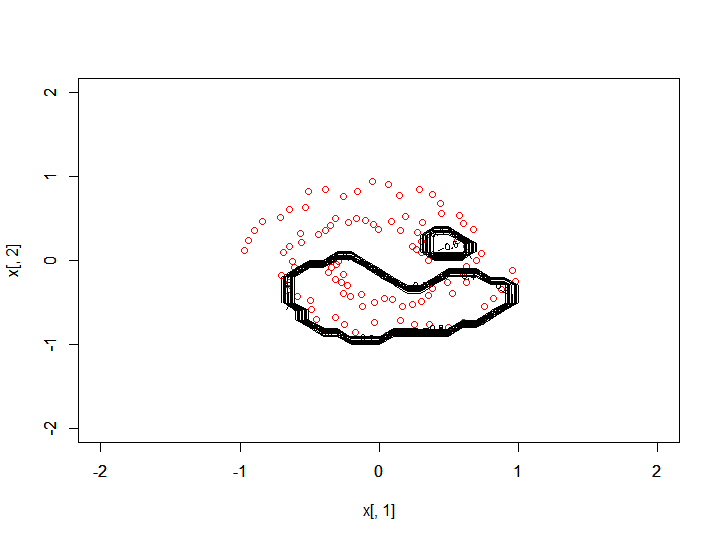
**P = 1**

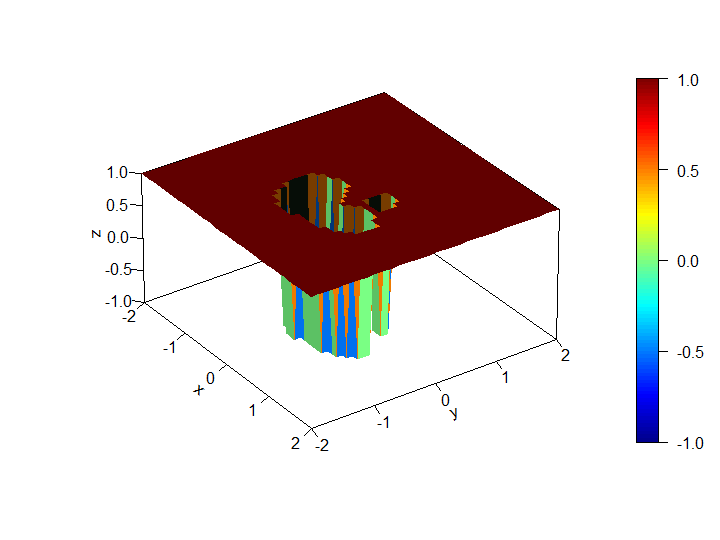




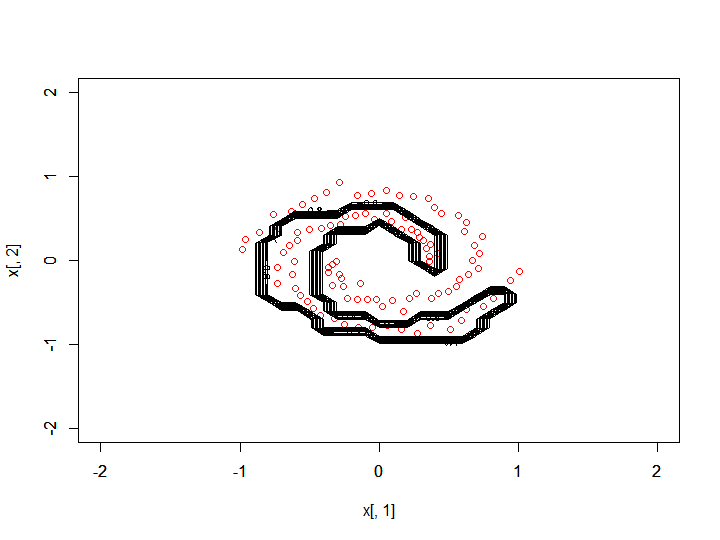
1. *Base spirals e:*

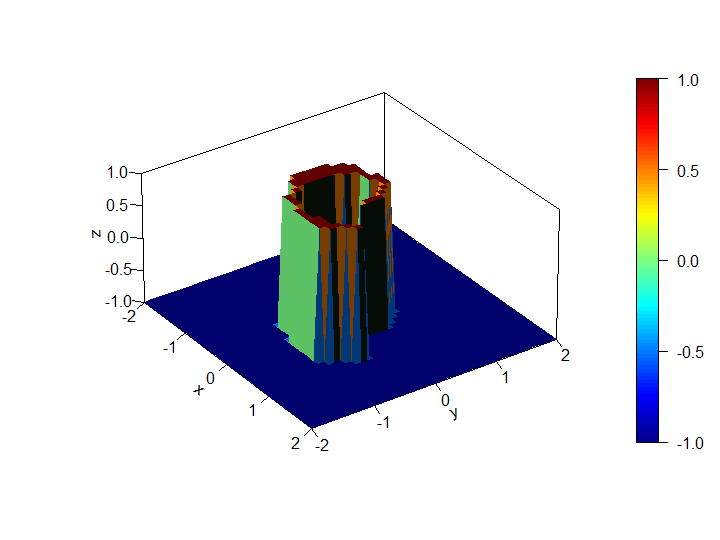
* *P = 5 centros*





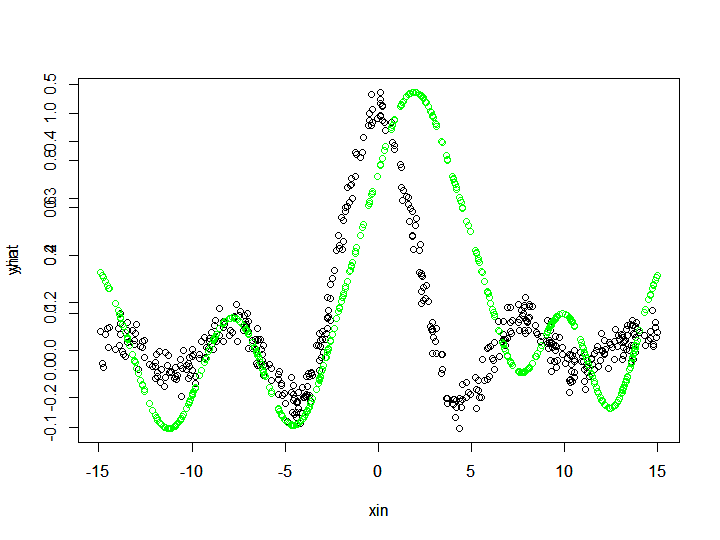
**P=20**





**FUNÇAO SINC**

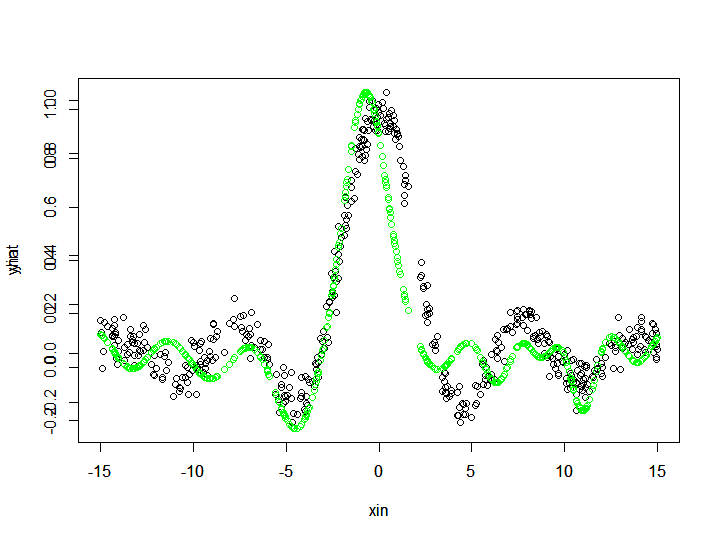
**P= 5**



**MSE [,1]**

**[1,] 0.05515065**

**P = 9**



**MSE [,1]**

**[1,] 0.02184937**

**ANEXOS**

O código para realização dos exercícios 1, 2, 3 e 4 se encontra devidamente documentado, através dos comentários, em anexo:

rm(list=ls())

library('mlbench')

library('plot3D')

cd0 <- mlbench.spirals(100,sd = 0.05)

#pdfnvar<-function(x,m,K,n) ((1/(sqrt((2\*pi)^n\*(det(K)))))\*exp(-0.5\*(t(x-m) %\*% (solve(K)) %\*% (x-m))))

trainRBF <- function(xin,yin,p){

# aplica a PDF nos dados -> OBRENÇAO DA MATRIX H

pdfnvar<-function(x,m,K,n) ((1/(sqrt((2\*pi)^n\*(det(K)))))\*exp(-0.5\*(t(x-m) %\*% (solve(K)) %\*% (x-m))))

# pega as dimensoes de entrada

N <- dim(xin)[1]

n <- dim(xin)[2]

xin <- as.matrix(xin)

yin <- as.matrix(yin)

# aplica clustering nos dados com k-means nativo do R

dataclustering <- kmeans(xin,p)

centers <- as.matrix(dataclustering$centers) # armazena os centros encontrados

covlist <- list()

# estima matriz de covariancia para cada um dos centros

for(i in 1:p){

ici <- which(dataclustering$cluster == i)

xci <- xin[ici,]

if (n == 1)

covi <- var(xci)

else

covi<-cov(xci)

covlist[[i]] <- covi

}

#determina matriz H

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

for (j in 1:N) {

for(i in 1:p){

mi<-centers[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(H,1)

# calcula W usando a pseudoinversa de Haug e sem regularizacao

W <- (solve(t(Haug)%\*%Haug) %\*% t(Haug)) %\*% yin

return(list(centers,covlist,W,H))

}

YRBF <- function(xin,modelRBF,binary){

# declara a funcao gaussiana radial

pdfnvar<-function(x,m,K,n) ((1/(sqrt((2\*pi)^n\*(det(K)))))\*exp(-0.5\*(t(x-m) %\*% (solve(K)) %\*% (x-m))))

# pega as dimensoes de entrada

N <- dim(xin)[1]

n <- dim(xin)[2]

xin <- as.matrix(xin)

centers<-as.matrix(modelRBF[[1]])

covlist <- modelRBF[[2]]

p <- length(covlist) # numero de funcoes radiais

W <- modelRBF[[3]]

#determina matriz H

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

for (j in 1:N) {

for(i in 1:p){

mi<-centers[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(H,1)

Yhat <- Haug %\*% W

if(binary==TRUE)

return(sign(Yhat))

else

return(Yhat)

}

x <- as.matrix(cd0$x)

y <- as.matrix(cd0$classes)

class(y) <- "numeric"

y[y==2]<- (-1)

y[y==1]<- (1)

separeTrainAndTest <- function(x,y,percTrain){

xin <- x

yin <- y

indexTreino <- sample(dim(xin)[1])

Xtrain <- xin[indexTreino[1:(dim(xin)[1]\*percTrain)],]

Ytrain <- as.matrix(yin[indexTreino[1:(dim(xin)[1]\*percTrain)],])

Xtest <- xin[indexTreino[((dim(xin)[1]\*percTrain)+1):dim(xin)[1]],]

Ytest <- as.matrix(yin[indexTreino[((dim(xin)[1]\*percTrain)+1):dim(xin)[1]],])

return(list(Xtrain,Ytrain,Xtest,Ytest))

}

# training the model

p <- 20

model <- trainRBF(x,y,p)

yhat <- YRBF(x,model,binary = TRUE)

MSE <- (t(y-yhat) %\*% (y-yhat))/dim(y)[1]

acc <- y - yhat

acc <- length(acc[acc==0])/dim(y)[1]

seqi <- seq(-2,2,0.1)

seqj <- seq(-2,2,0.1)

M <- matrix(0,nrow = length(seqi),ncol = length(seqj))

ci <- 0

for (i in seqi) {

ci<-ci+1

cj<-0

for (j in seqj) {

cj<-cj+1

X<-as.matrix(cbind(i,j))

M[ci,cj]<- YRBF(X,model,binary = TRUE)

}

}

plot(x[,1],x[,2],col = 'red', xlim = c(-2,2), ylim = c(-2,2))

par(new=T)

contour(seqi,seqj,M,xlim = c(-2,2),ylim = c(-2,2),xlab= '', ylab='')

persp3D(seqi,seqj,M,counter = T,theta=55,phi=30,r=40,d=0.1,expand=0.5,ltheta=90,lphi=180,shade=0.4,ticktype='detailed',nticks=5)

# sinc -----

rm(list=ls())

xin <- runif(500,-15,15)

yin <- sin(xin)/xin + rnorm(500,0,0.05)

plot(xin,yin)

trainRBF <- function(xin,yin,p){

# aplica a PDF nos dados -> OBRENÇAO DA MATRIX H

pdfnvar<-function(x,m,K,n) ((1/(sqrt((2\*pi)^n\*(det(K)))))\*exp(-0.5\*(t(x-m) %\*% (solve(K)) %\*% (x-m))))

# pega as dimensoes de entrada

N <- dim(xin)[1]

n <- dim(xin)[2]

xin <- as.matrix(xin)

yin <- as.matrix(yin)

# aplica clustering nos dados com k-means nativo do R

dataclustering <- kmeans(xin,p)

centers <- as.matrix(dataclustering$centers) # armazena os centros encontrados

covlist <- list()

# estima matriz de covariancia para cada um dos centros

for(i in 1:p){

ici <- which(dataclustering$cluster == i)

xci <- xin[ici,]

if (n == 1)

covi <- var(xci)

else

covi<-cov(xci)

covlist[[i]] <- covi

}

#determina matriz H

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

for (j in 1:N) {

for(i in 1:p){

mi<-centers[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(H,1)

# calcula W usando a pseudoinversa de Haug e sem regularizacao

W <- (solve(t(Haug)%\*%Haug) %\*% t(Haug)) %\*% yin

return(list(centers,covlist,W,H))

}

YRBF <- function(xin,modelRBF,binary){

# declara a funcao gaussiana radial

pdfnvar<-function(x,m,K,n) ((1/(sqrt((2\*pi)^n\*(det(K)))))\*exp(-0.5\*(t(x-m) %\*% (solve(K)) %\*% (x-m))))

# pega as dimensoes de entrada

N <- dim(xin)[1]

n <- dim(xin)[2]

xin <- as.matrix(xin)

centers<-as.matrix(modelRBF[[1]])

covlist <- modelRBF[[2]]

p <- length(covlist) # numero de funcoes radiais

W <- modelRBF[[3]]

#determina matriz H

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

for (j in 1:N) {

for(i in 1:p){

mi<-centers[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(H,1)

Yhat <- Haug %\*% W

if(binary==TRUE)

return(sign(Yhat))

else

return(Yhat)

}

p <- 30

model <- trainRBF(as.matrix(xin),as.matrix(yin),p)

yhat <- YRBF(as.matrix(xin),model,binary = FALSE)

MSE <- (t(as.matrix(yin)-yhat) %\*% (as.matrix(yin)-yhat))/dim(as.matrix(yin))[1]

plot(xin,yin,col = 'black')

par(new=T)

plot(xin,yhat,col = 'green')