Redes Neurais Artificiais

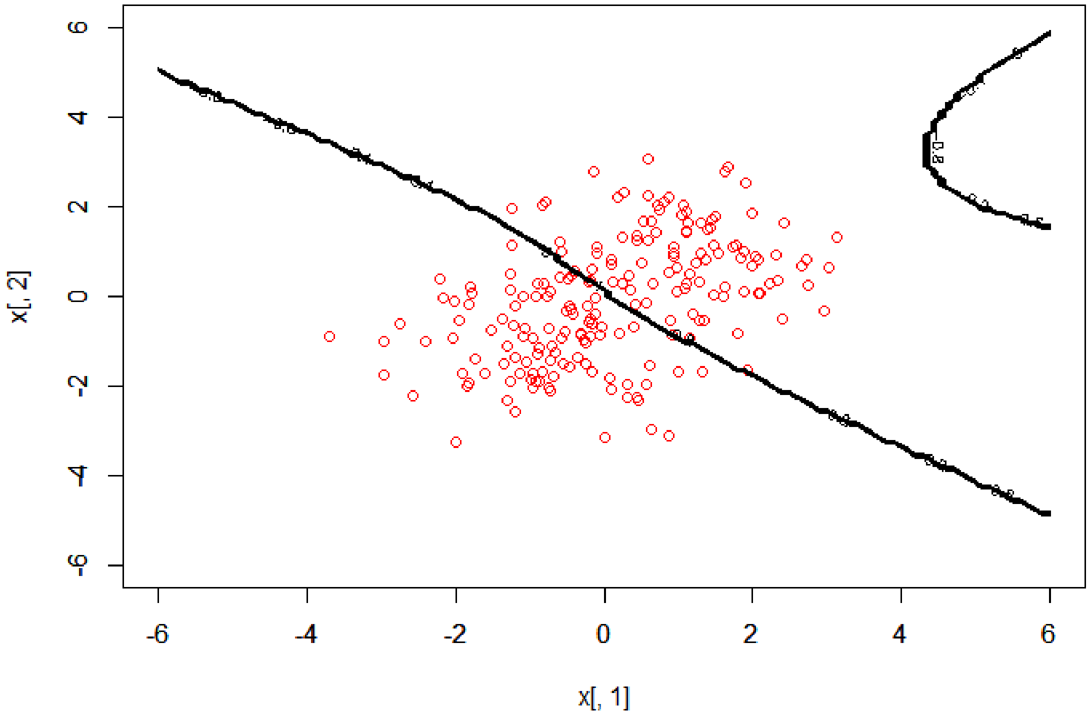
André Costa Werneck, Matrícula: 2017088140

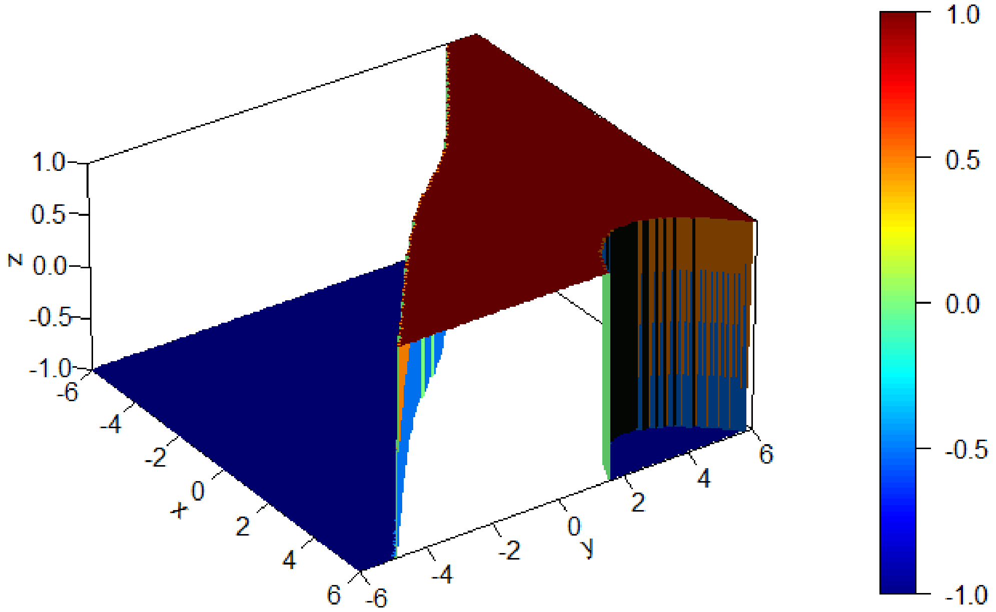
LISTA 5

08/05/2022

1) *Base 2dnormals e:*

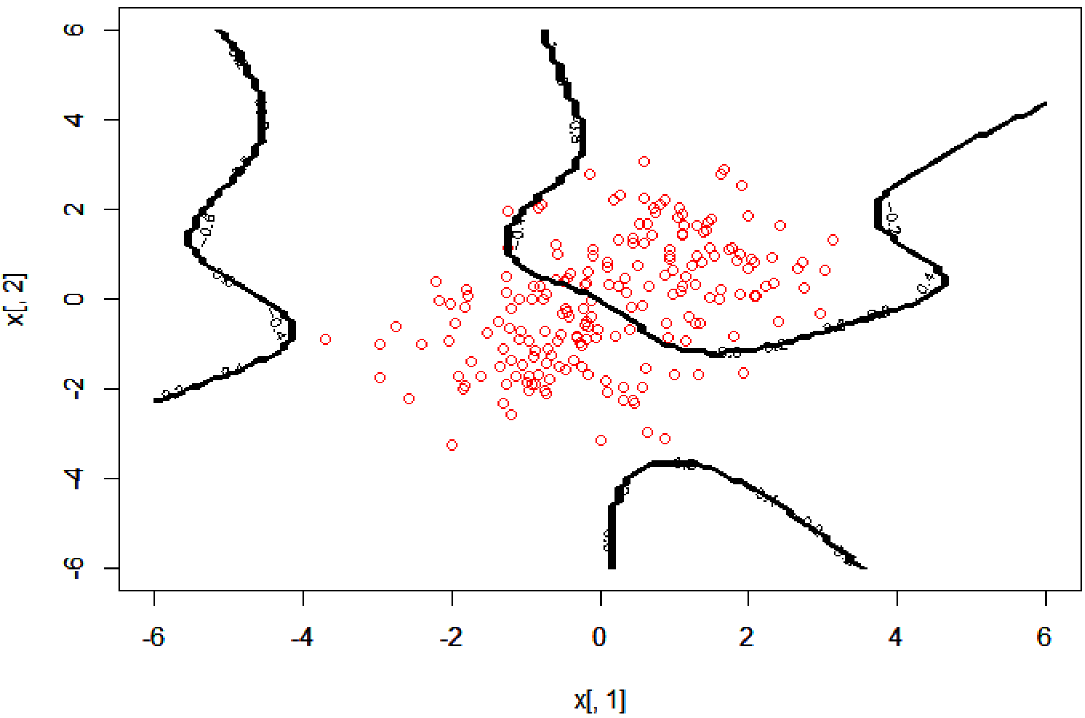
- *P = 5 neurônios*

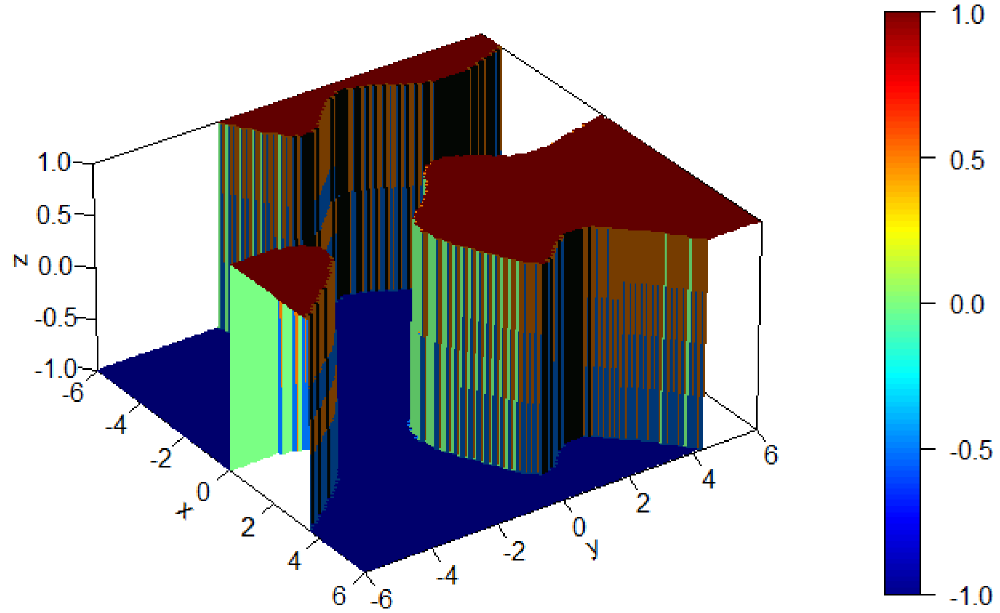




Acurácia = [1] 0.94

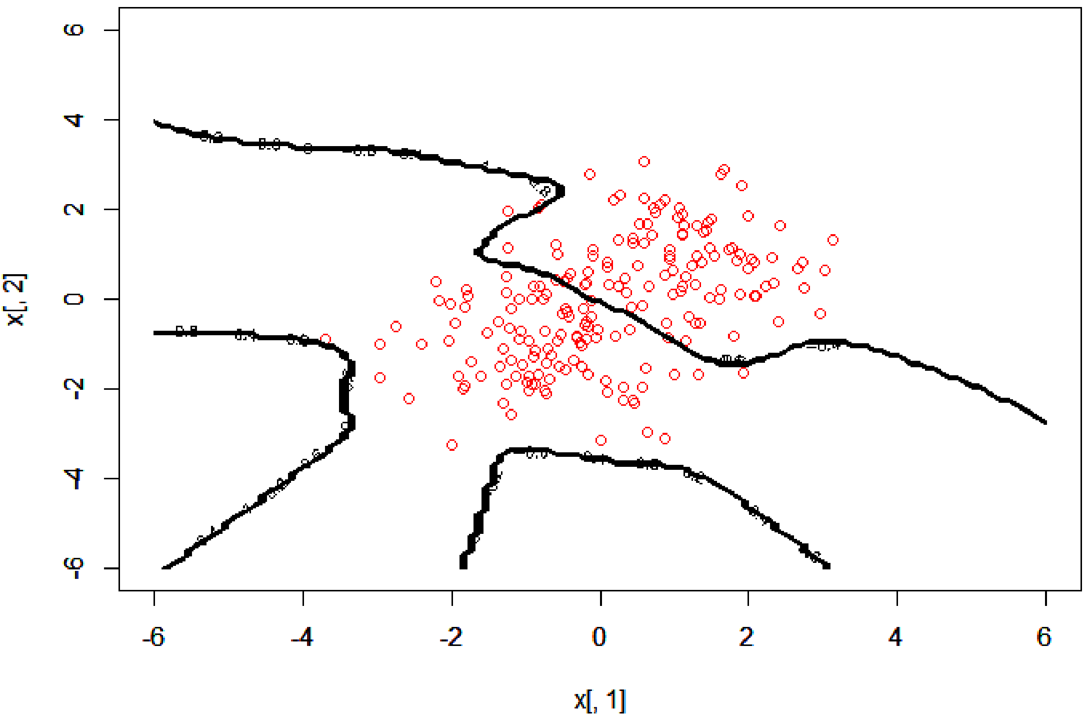
- *P = 20*

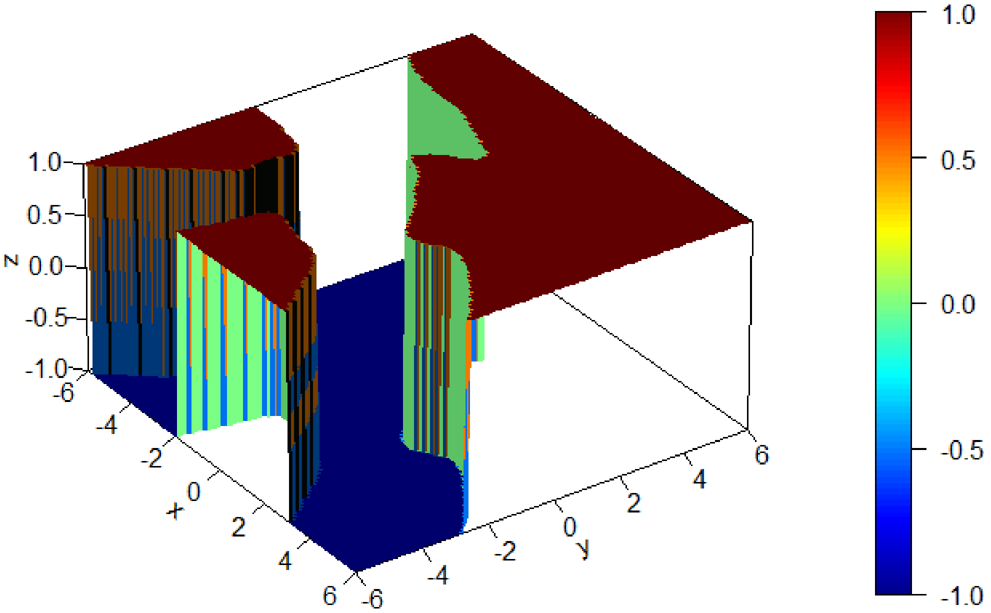




Acurácia = [1] 0.95

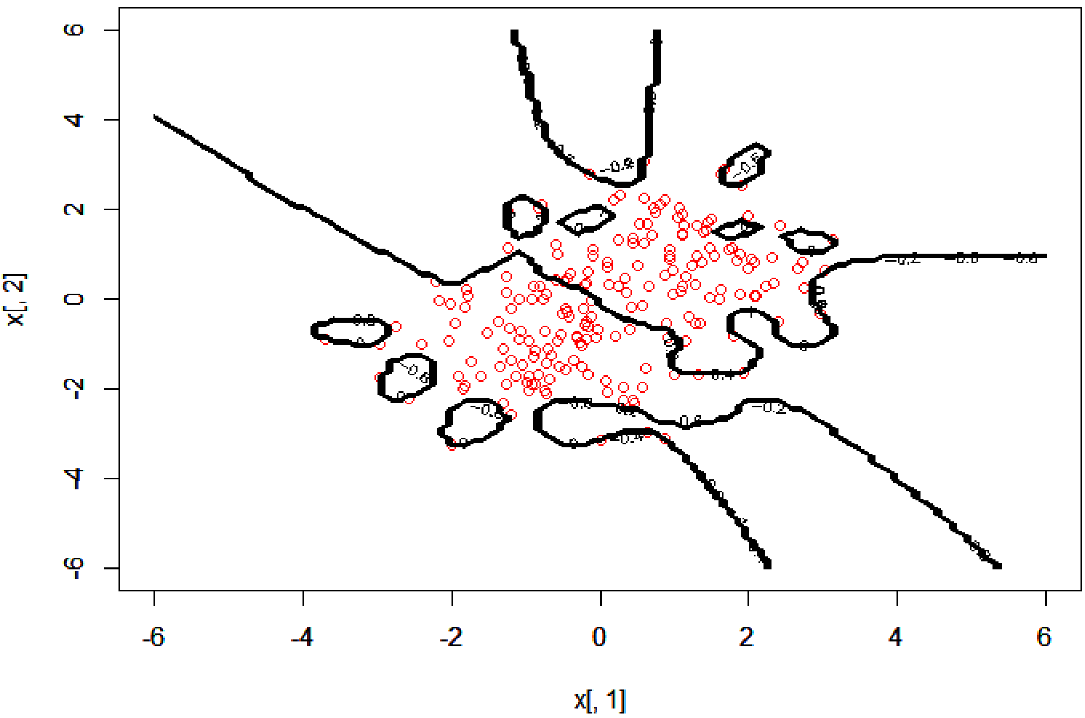
- *P = 50*

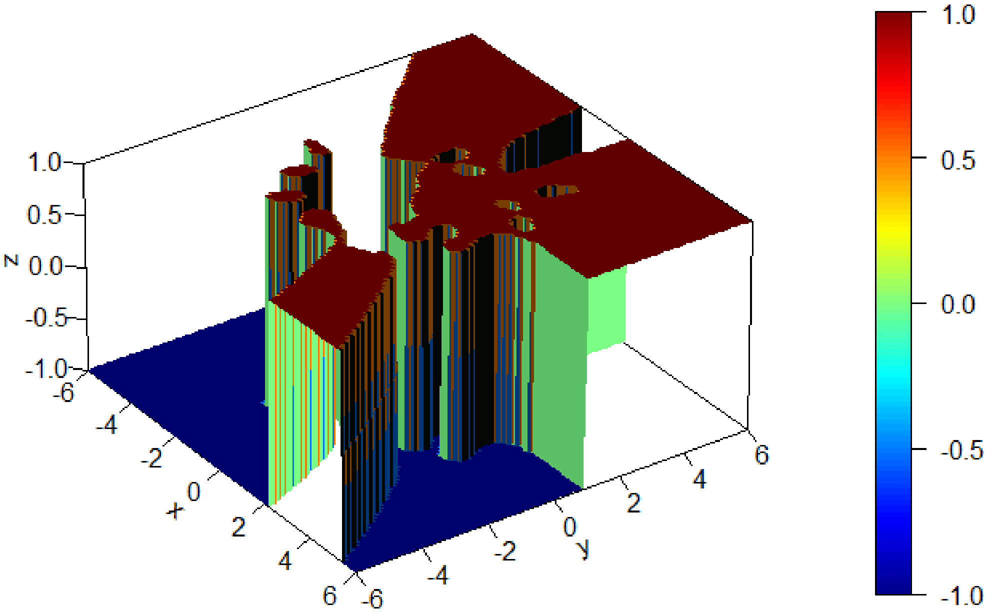




Acurácia = [1] 0.94

- *P = 100*





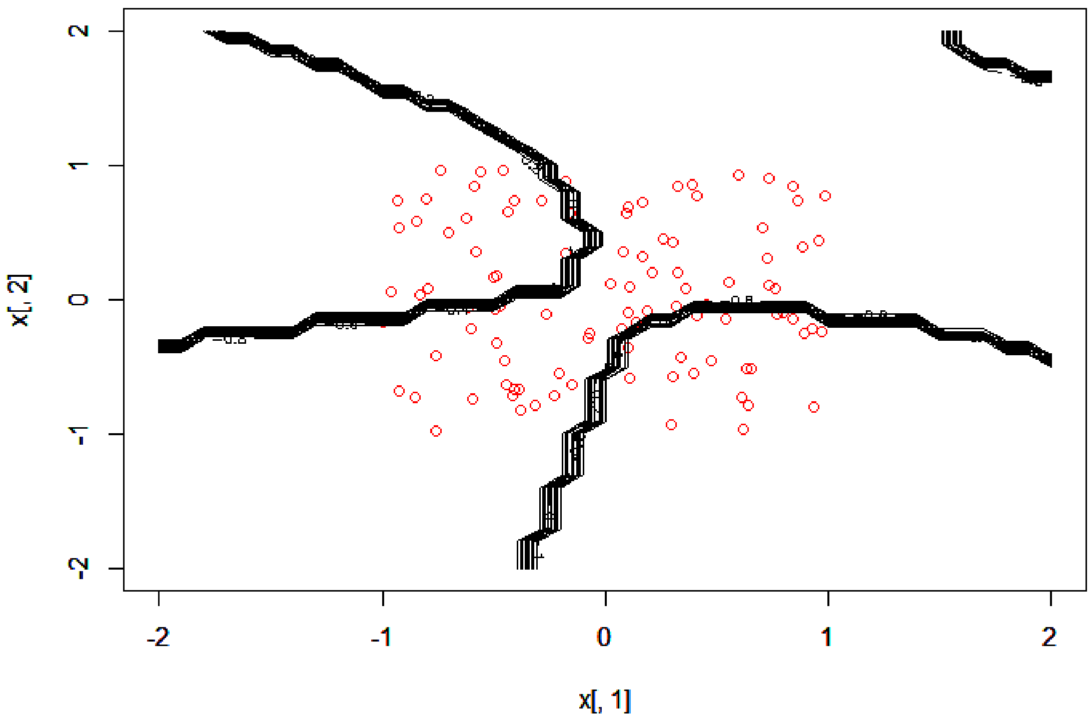
Acurácia = [1] 0.96

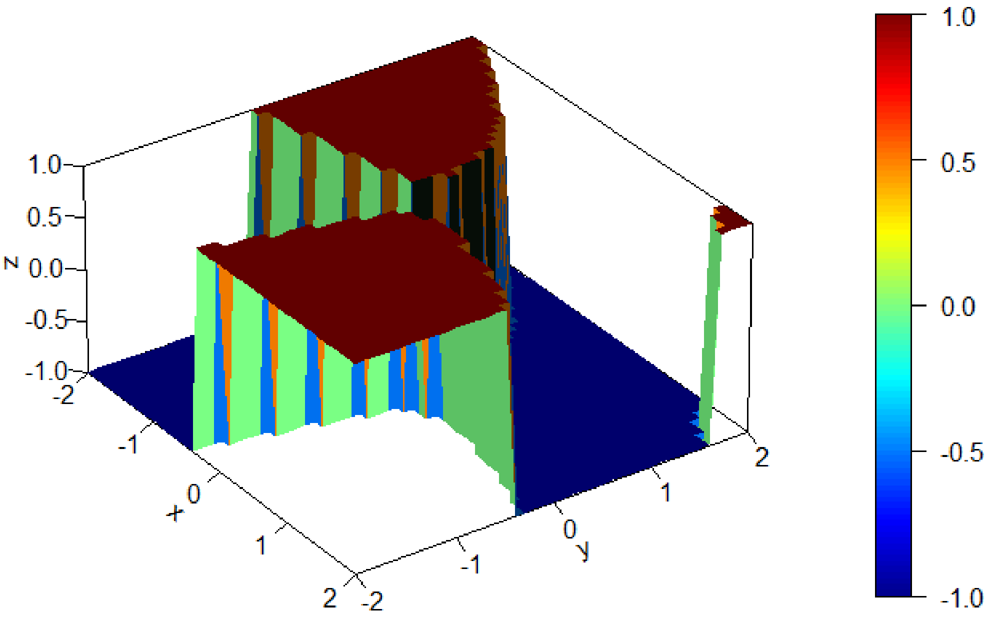
<u>Conclusão para base 2dnormals</u>

Foi observado no experimento feito acima, no qual se treinou uma base bidimensional normal com uma ELM com diferentes números de neurônios e 2 classes, diferentes resultados graças a variação do número de neurônios. Como o problema apresenta apenas 2 classes, um número de neurônio pequeno, como 5, apresentou uma curva de separação muito semelhante a uma reta e foi satisfatório, obtendo visualmente uma classificação justa e uma acurácia calculada de 94%, comprovando o escrito anteriormente. Ao se aumentar o número de neurônios, a gaussiana de separação foi alterando de formato e começou a buscar os pontos de fronteira com mais exatidão apresentando boa separação para p=20 e 50 neurônios, mas já com um sinal de overfitting para p=100. Vale ressaltar, também, que para p=5 a classificação obtida foi boa pois o problema era simples e bidimensional. Provavelmente, para problemas de classificação mais complexos, a separação não seria tão boa com um número baixo de neurônios.

2) Base XOR e:

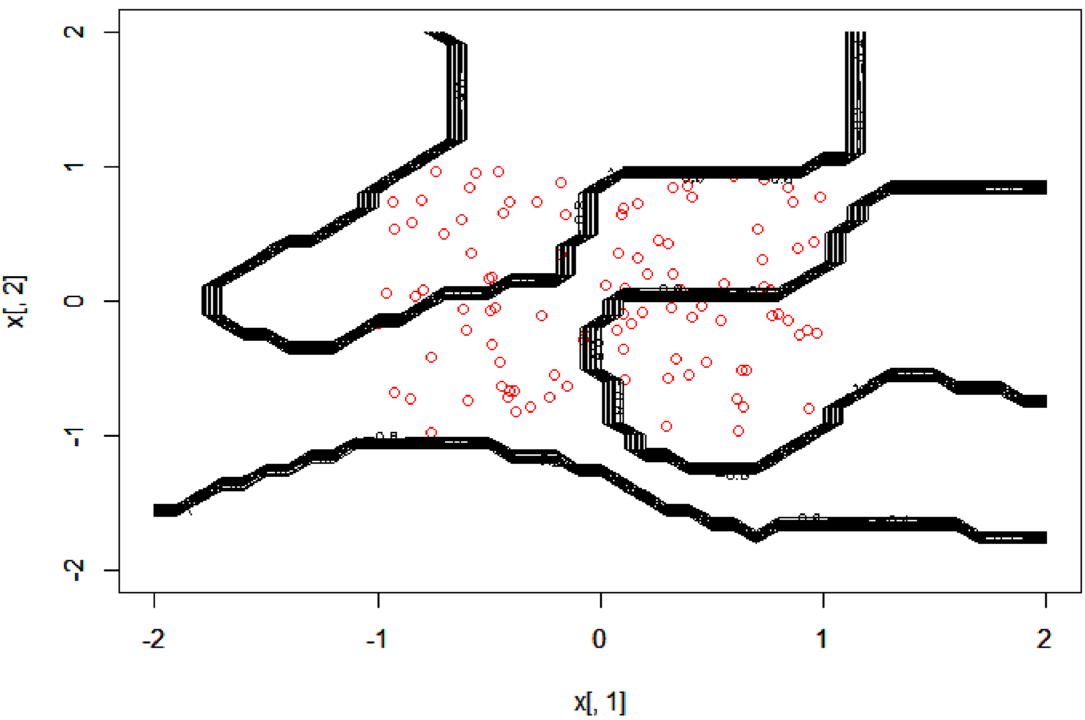
- *P = 5*

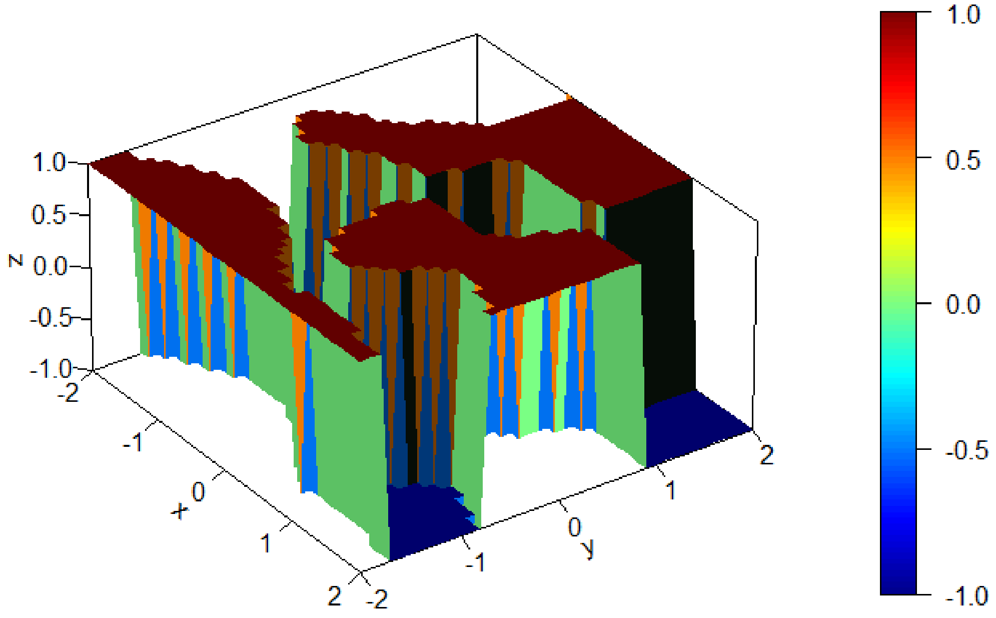




Acurácia = [1] 0.93

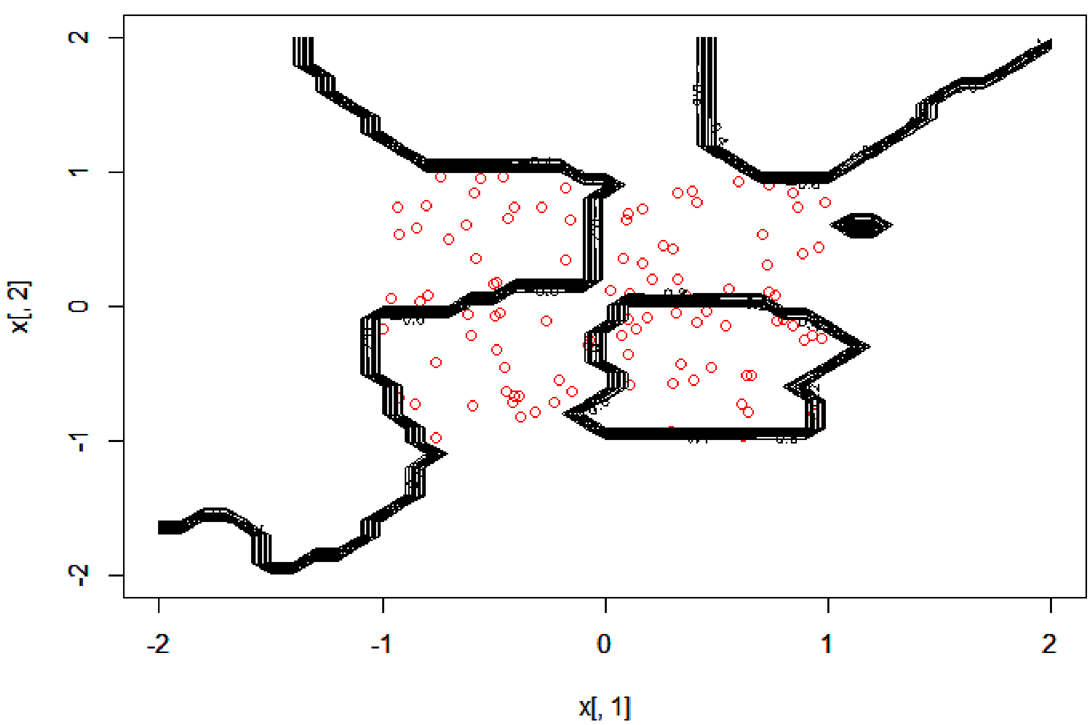
- *P = 20*

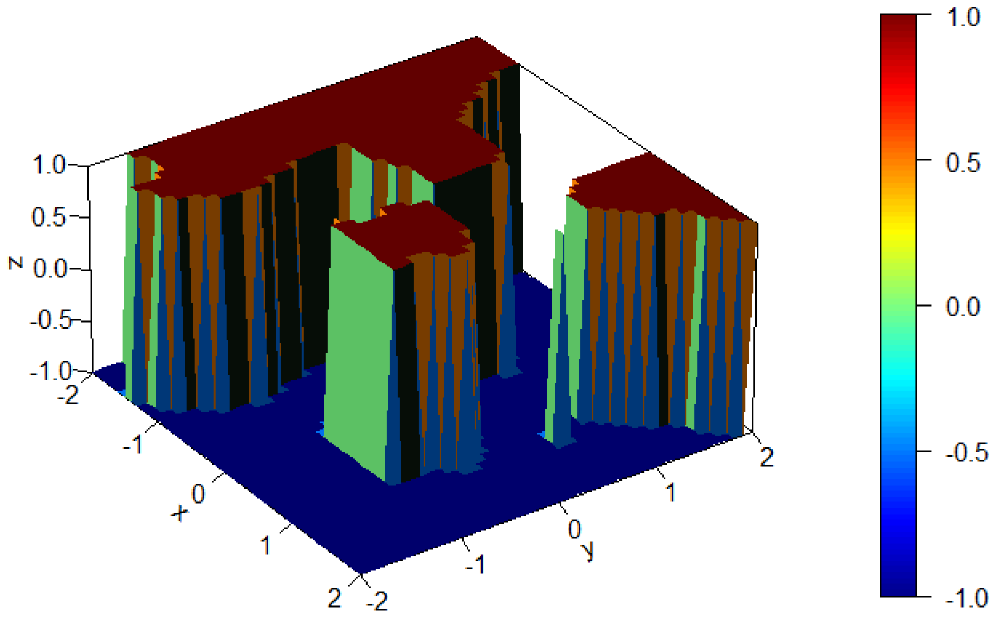




Acurácia = [1] 0.99

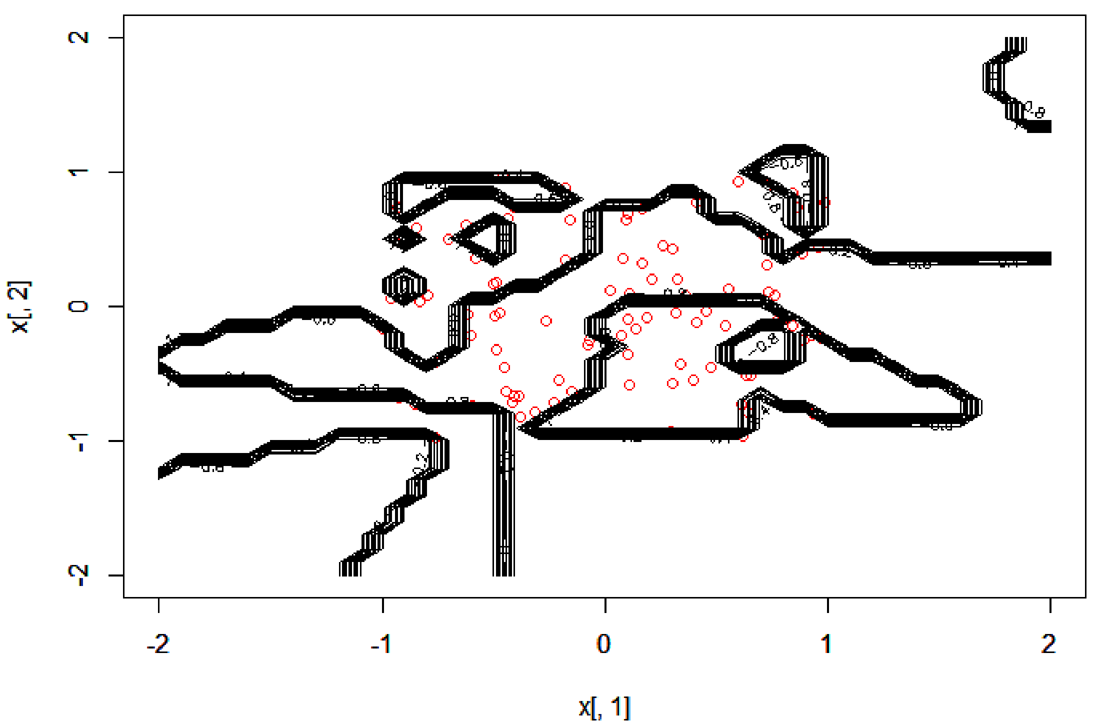
- *P = 50*

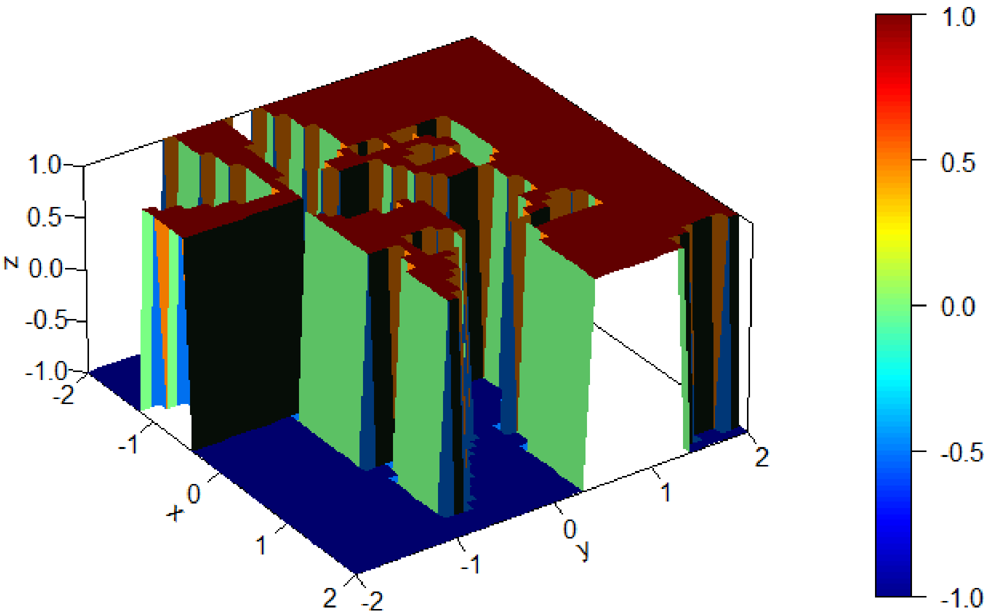




Acurácia = [1] 1

- *P = 100*





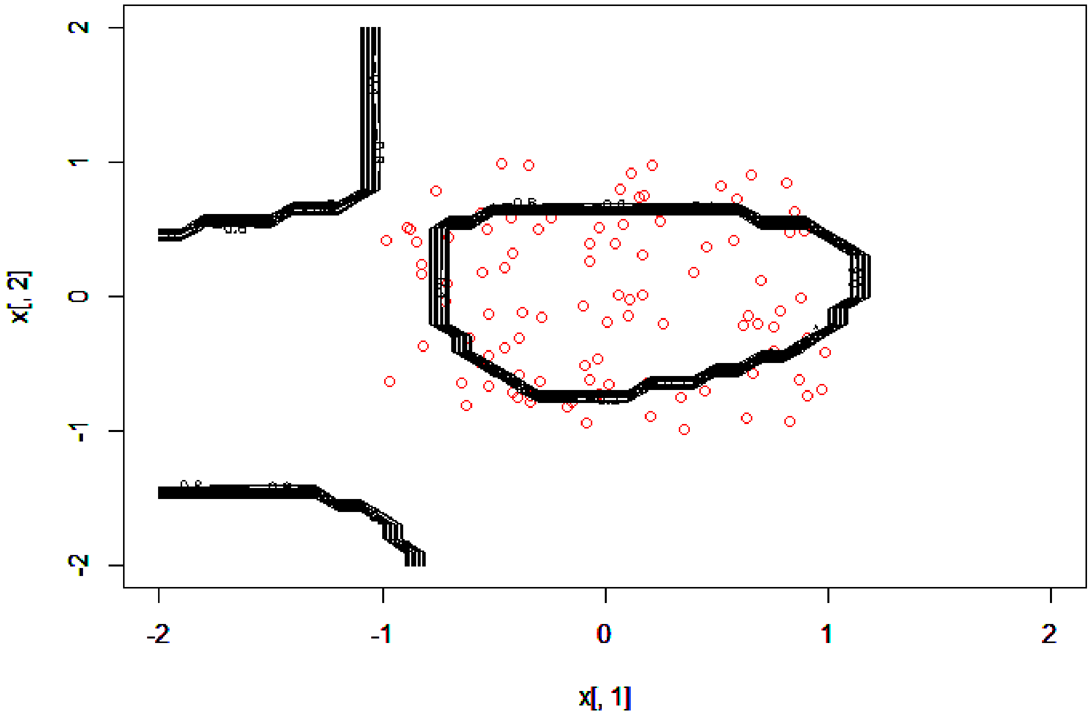
Acurácia = [1] 1

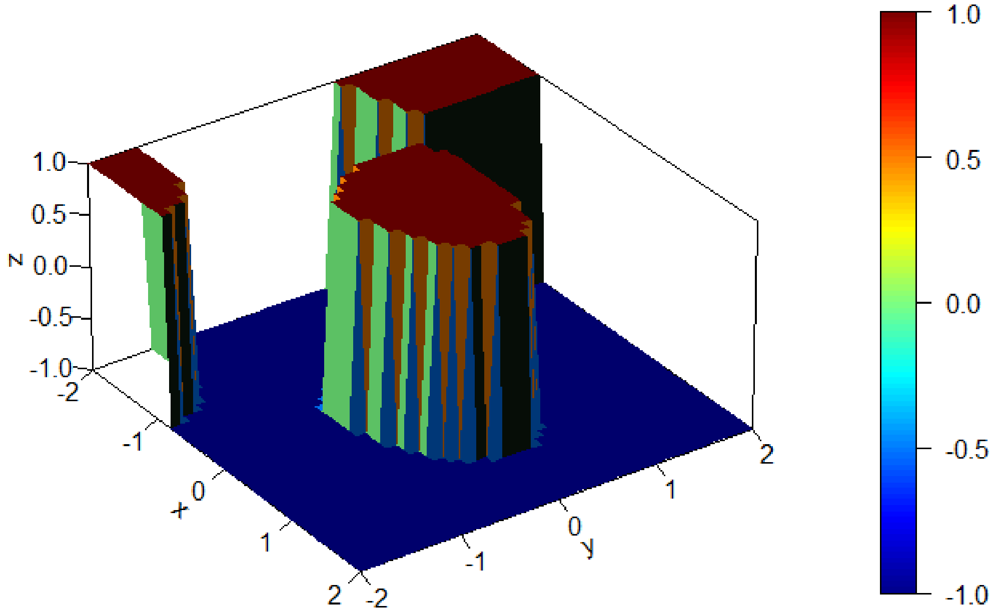
Conclusão para base XOR

Para a base XOR, o resultado do experimento teve conclusão semelhante ao experimento com a base 2dnormals. O problema XOR é um pouco mais complexo e, visualmente, apresentou sinais de overfitting mais cedo, me surpreendendo. Com p=20, já se observa uma curva que busca passagem muito próxima aos pontos de fronteira. Não é em vão que a acurácia saiu de 93% com p=5 para 99% e 100% com os outros valores de p, representando, se somado às análises visuais das superfícies de separação, um sinal claro de overfitting. Dessa forma, para um problema de classificação XOR, um número de neurônios mais baixo, entre 5 e 20, pode ser a melhor solução.

3) Base Circle e:

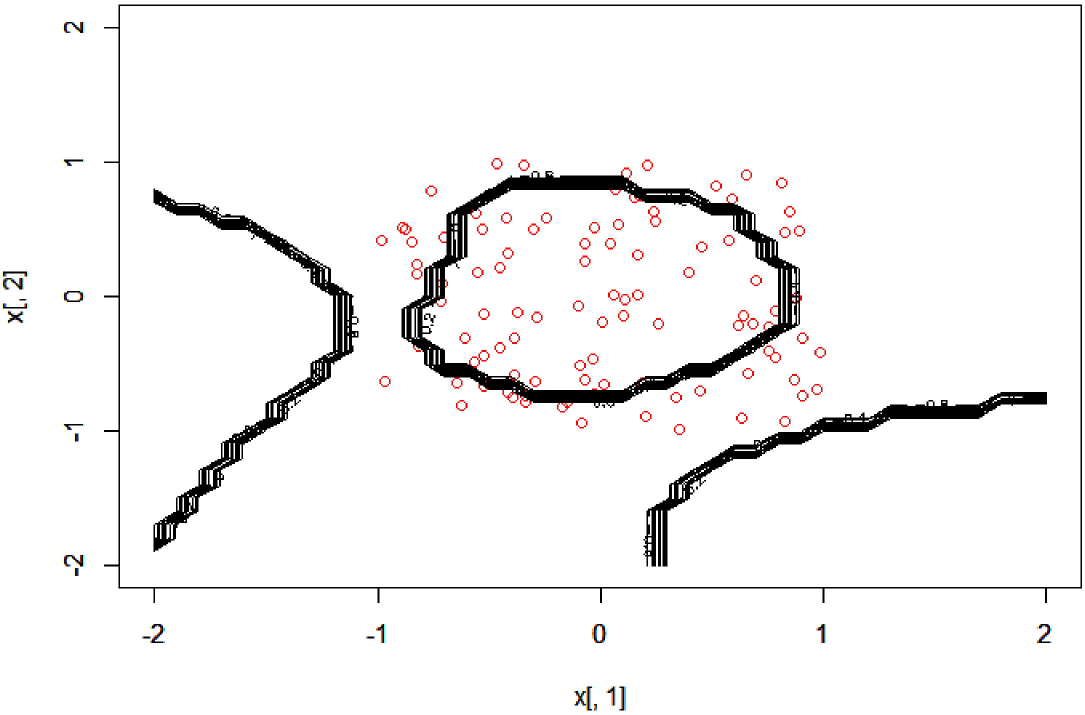
- *P = 5*

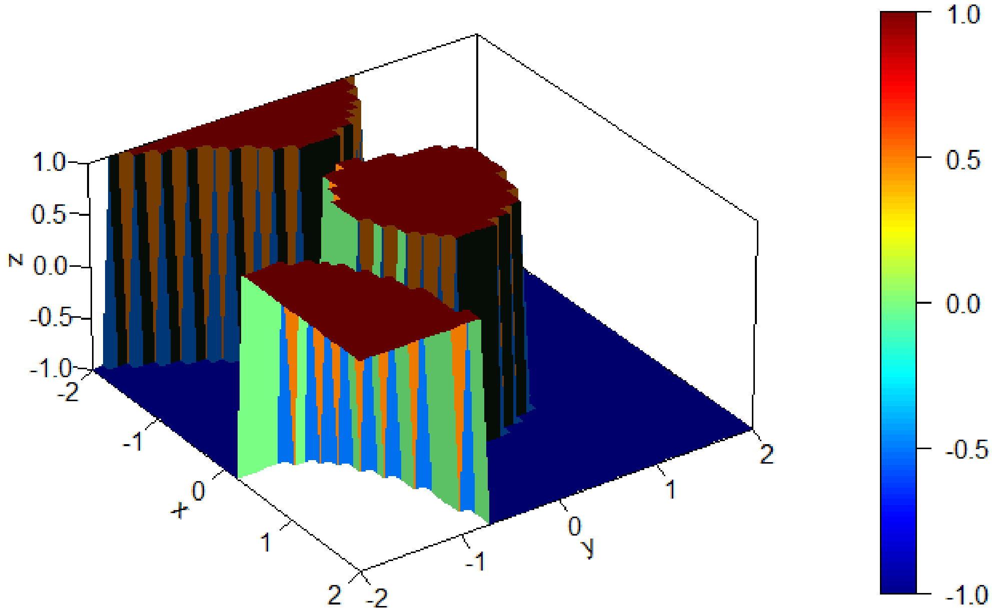




Acurácia = [1] 0.92

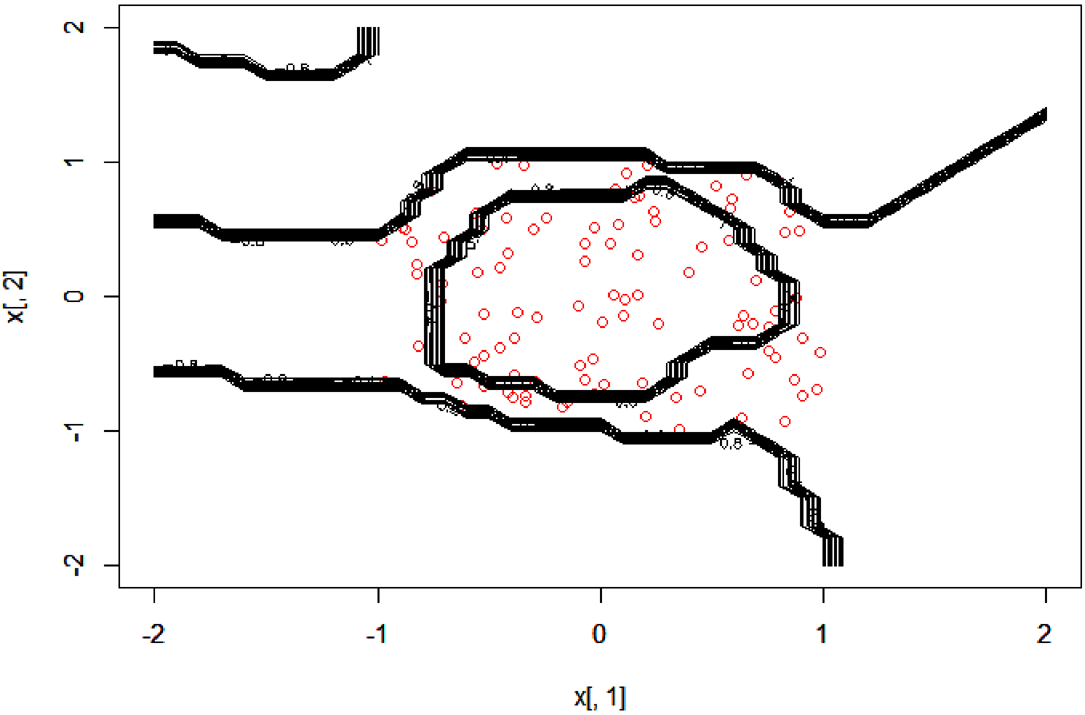
- *P = 20*

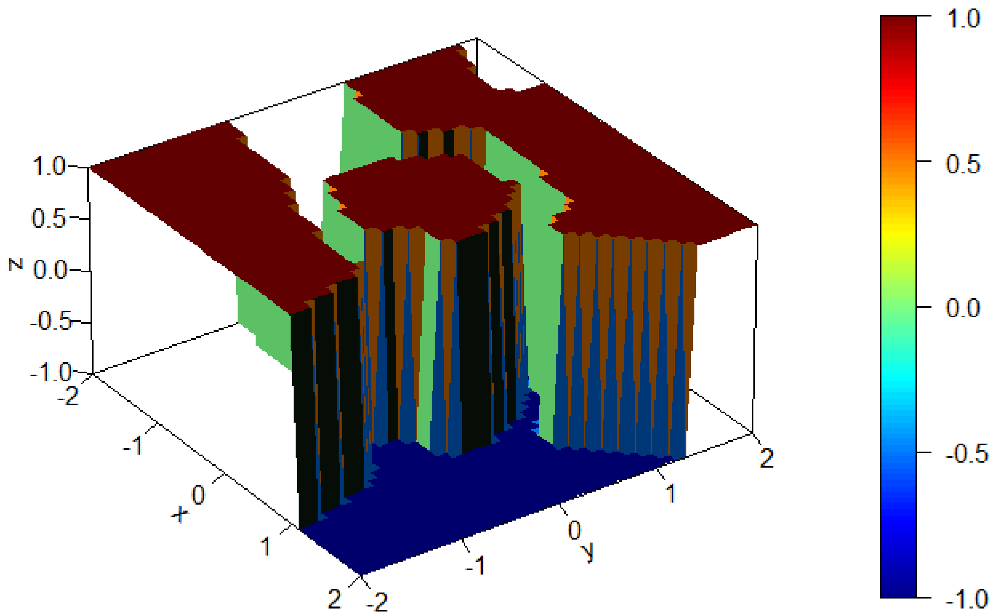




Acurácia = [1] 0.98

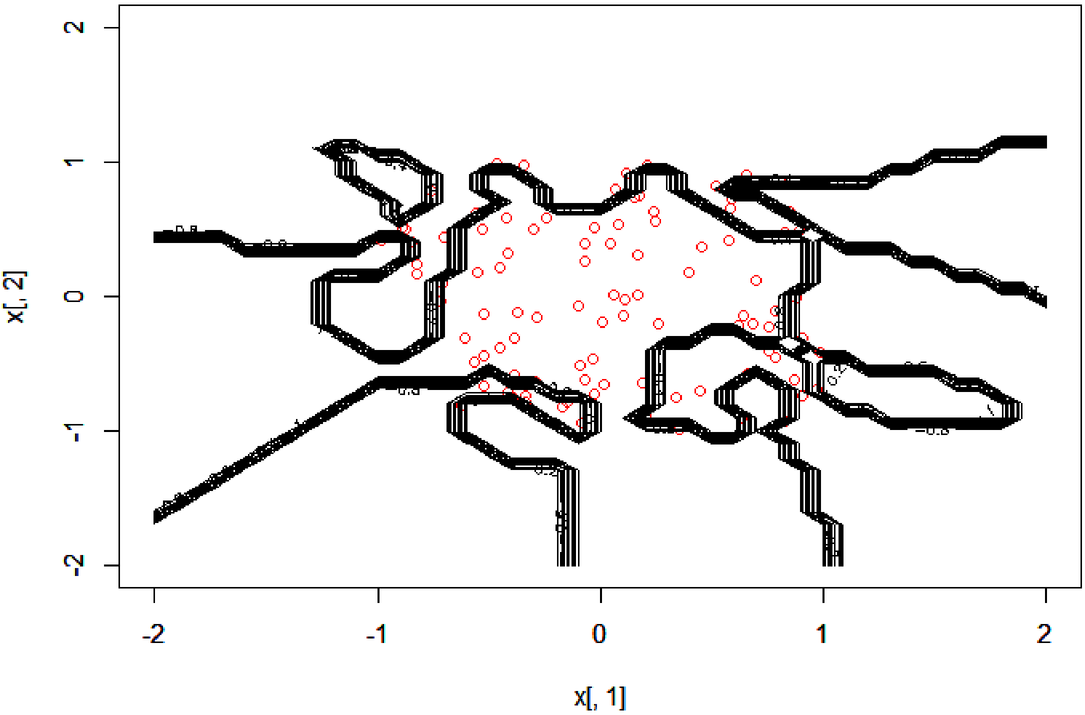
- *P = 50*

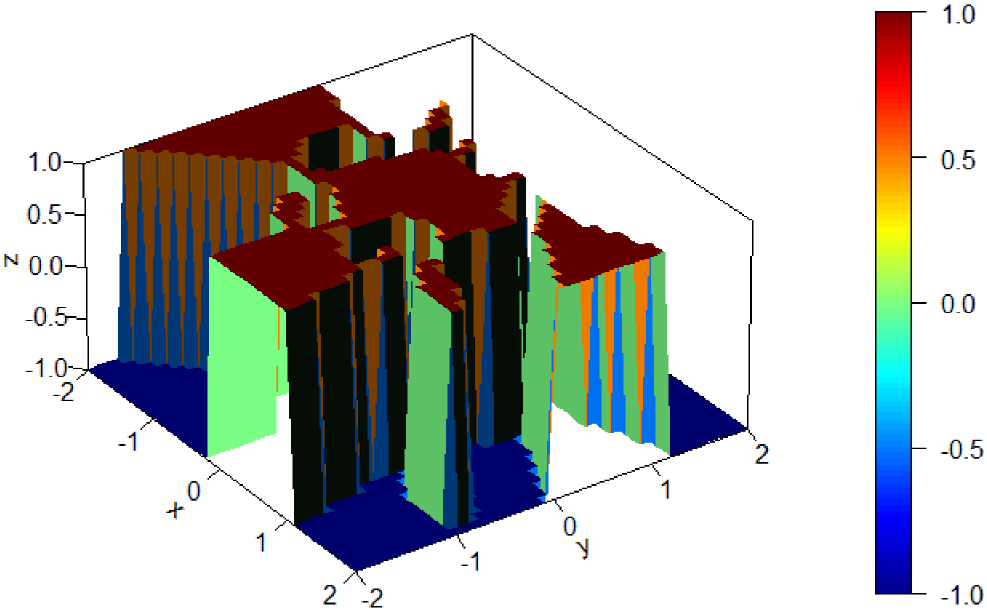




Acurácia = [1] 1

- *P = 100*





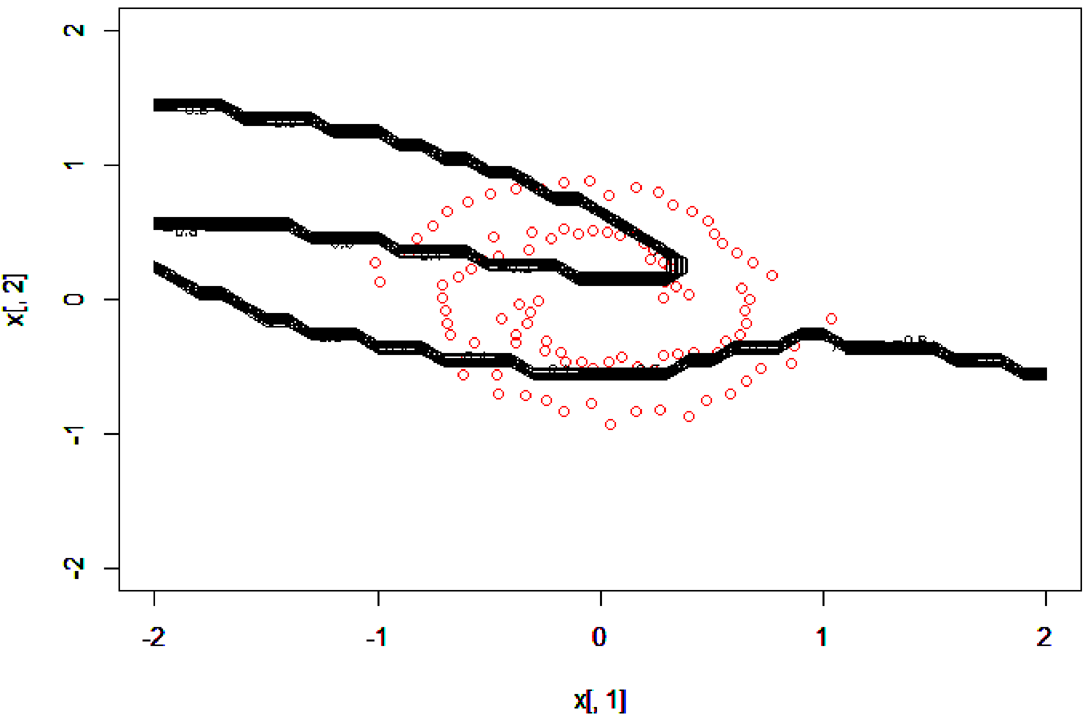
Acurácia = [1] 1

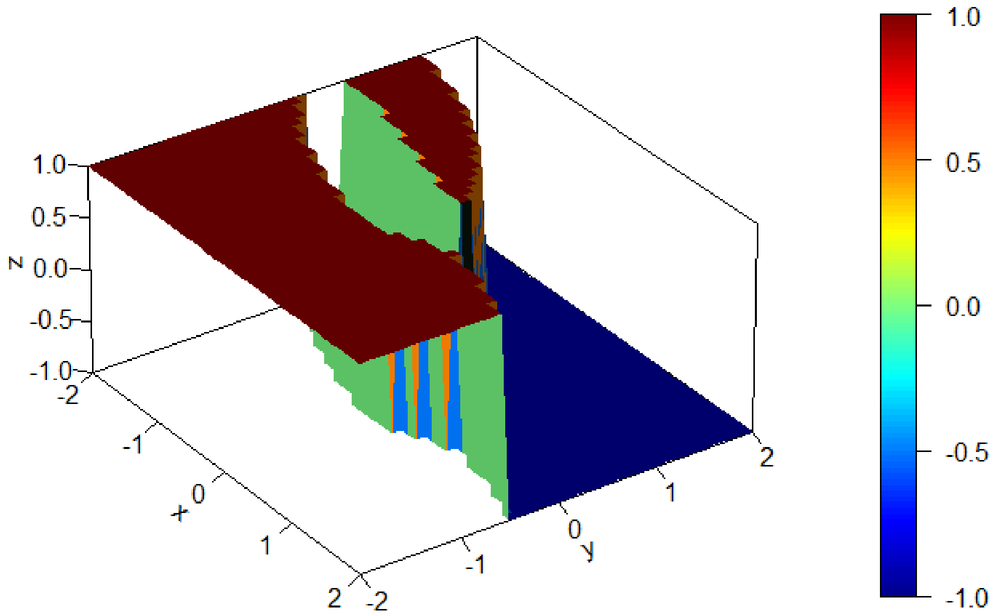
<u>Conclusão para base Circle</u>

Para a base Circle, a conclusão se difere das bases anteriores apenas no fato de que o overfitting demorou um pouco mais para acontecer. Observa-se que a intenção é obter uma gaussiana radial, circular, de separação e classificação dos dados. Ela é obtida com uma certa clareza visual até p=50 neurônios. Para p = 100, a superfície de classificação já não se parece mais com uma circunferência e isso é um sinal de overfitting, uma vez que ela passa exatamente pelos pontos fronteiriços e ignora o erro inerente e saudável ao experimento.

4) Base Spirals e:

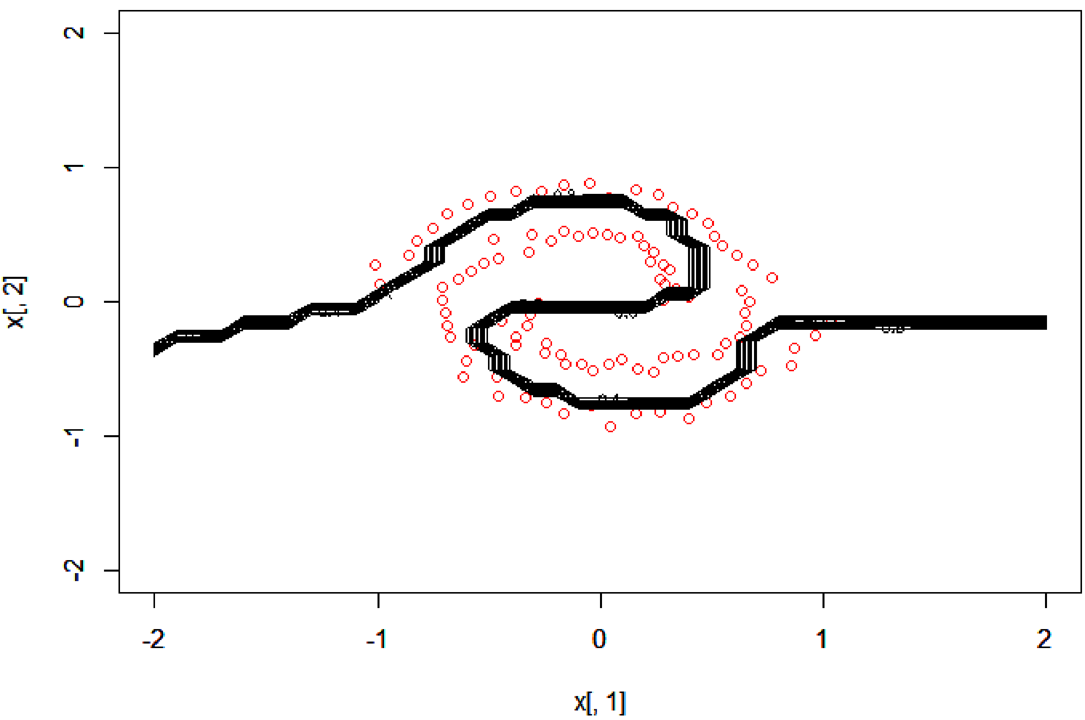
- *p = 5*

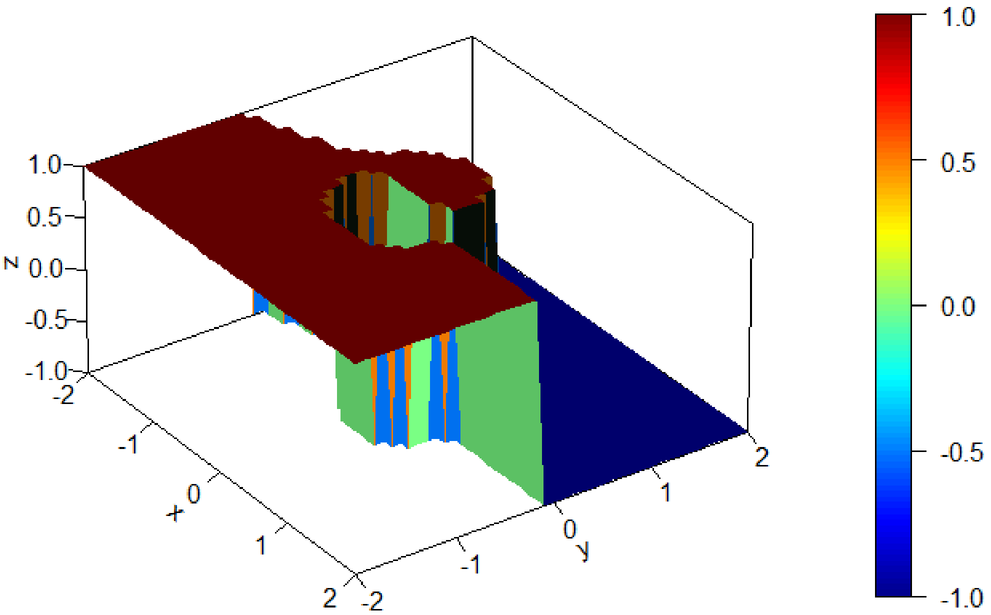




Acurácia = [1] 0.74

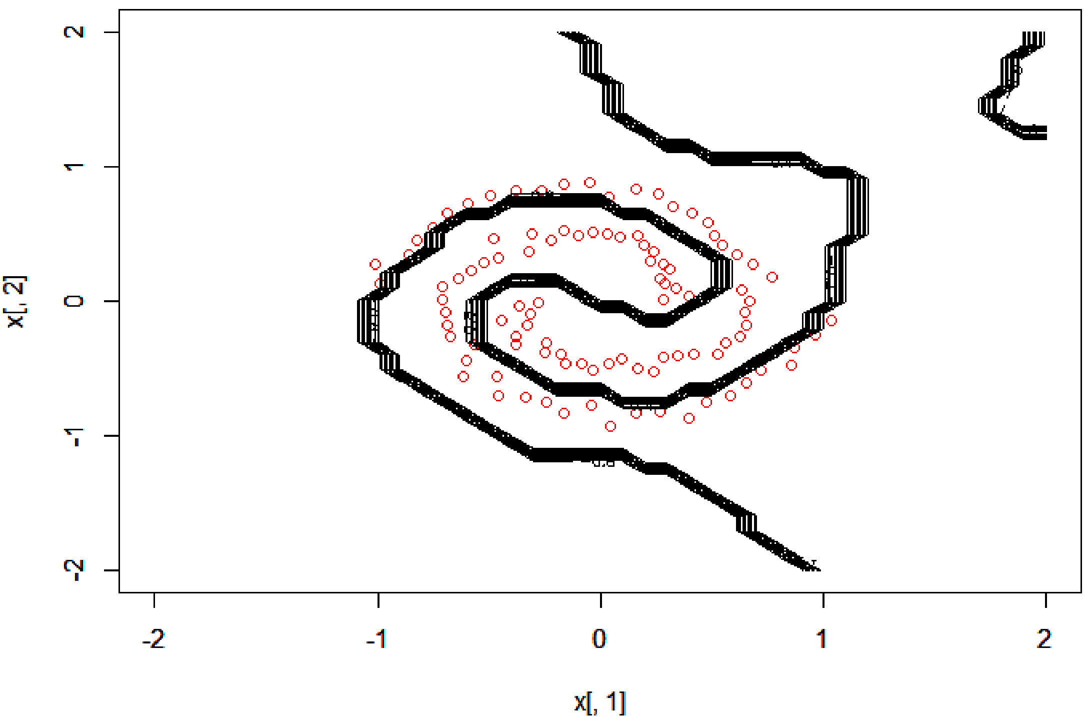
- *p = 10*

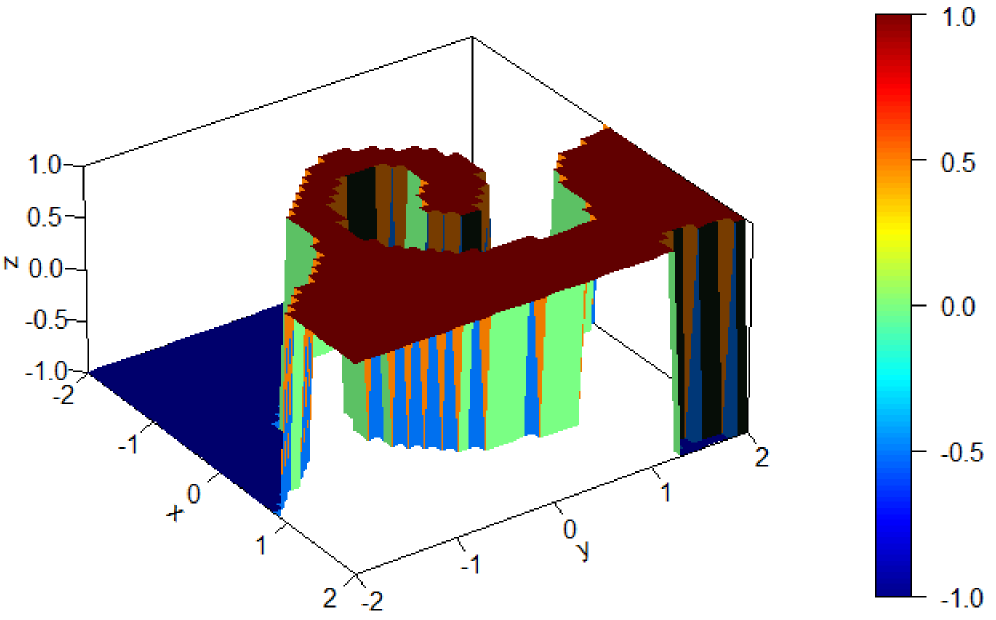




Acurácia = [1] 0.97

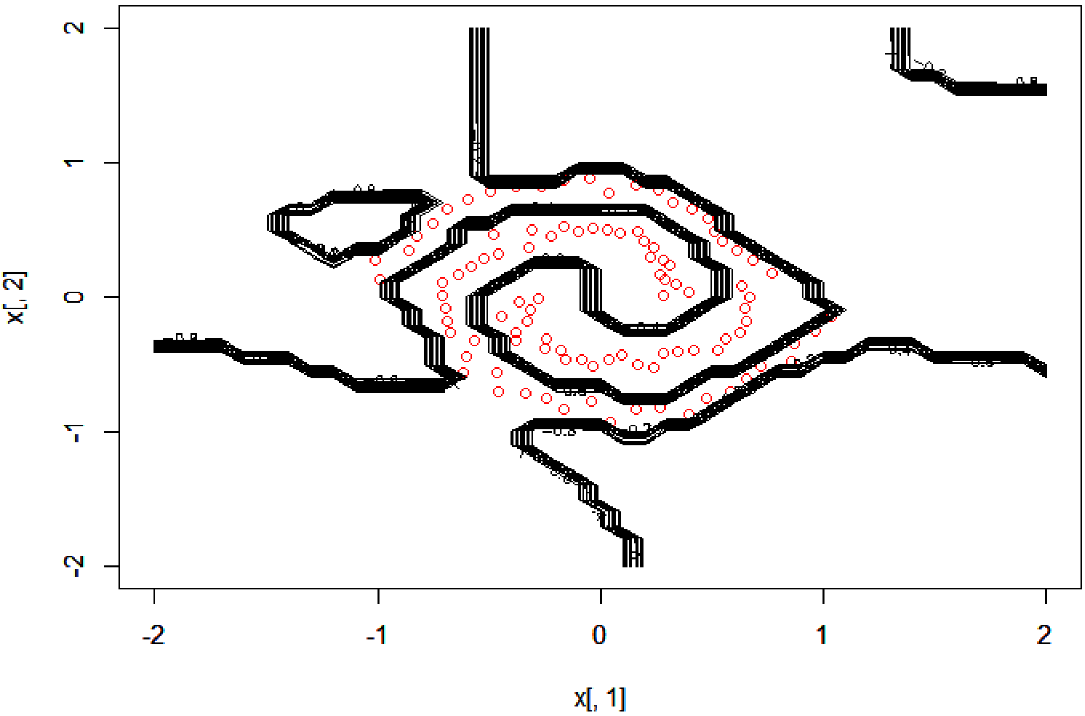
- *p = 20*

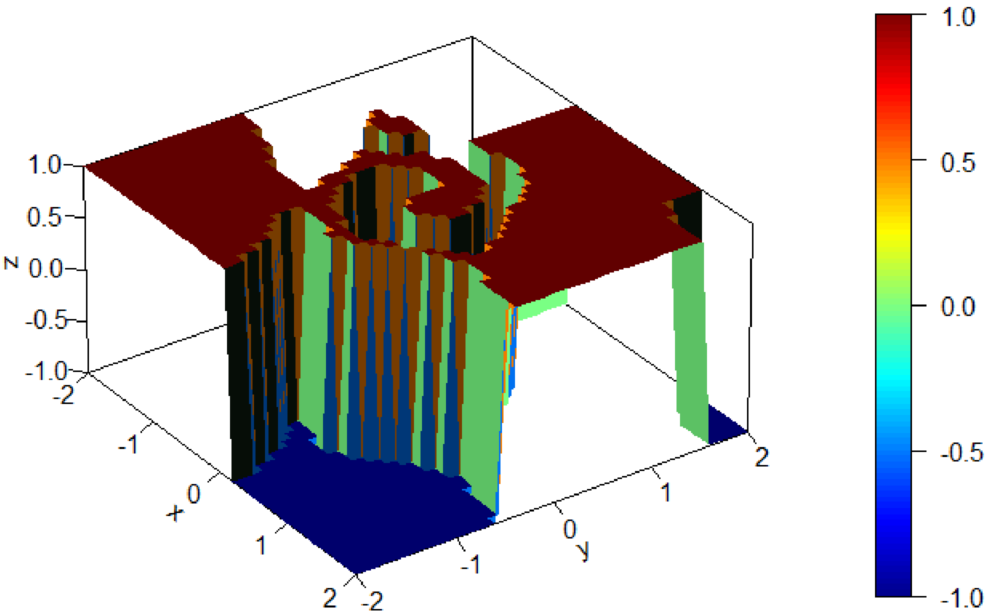




Acurácia = [1] 1

- *p = 50*





Acurácia = [1] 1

<u>Conclusão para a base Spirals</u>

Para a base Spirals, o experimento apresentou algumas diferenças relevantes. De primeira, para um p = 5, como pode ser observado acima, o modelo de ELM não conseguiu aprender muito bem e classificou apenas 74% das amostras corretamente. Entretanto, provando a enorme sensibilidade desse problema ao número de neurônios, observou-se uma convergência rápida e muita diferença de aprendizado para o teste com p=10, 20 e 50 neurônios. A acurácia saltou de 74% para 97% e 100%, respectivamente, e a superfície foi ficando mais com a cara de uma espiral, como deveria ser. Entretanto, semelhantemente ao problema da base Circles, com um número mais alto de neurônios, como 50, por exemplo, a gaussiana de separação vai perdendo a forma espiral e começa a apresentar sinais claros de overfitting. Dessa forma, conclui-se que essa base foi a que mais apresentou sensibilidade ao número de neurônios e foi a mais difícil de ser aprendida, uma vez que com um P baixo, o modelo não consegue apresentar uma solução apropriada.

ANEXOS

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

```
<<echo=FALSE,results=hide>>=
# gerando os dados de entrada
rm(list = ls())

library('mlbench')
library('corpcor')
library('caret')
library('plot3D')

cd0 <- mlbench.2dnormals(200)

cd1 <- mlbench.xor(100)

cd2 <- mlbench.circle(100)

cd3 <- mlbench.spirals(100,sd = 0.05)

# funcao ELM
treinaELM <- function(Xin, Yin, p){
  #pega as dimensoes

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

  # coloca 1 para termo de polarizacao

  Xaug <- cbind(Xin,1)
  #cria Z aleatoriamente
  Z <- replicate(p,runif((n+1),-0.5,0.5))

  # calcula H
  H <- tanh(Xaug %*% Z)

  #calcula pseudoinversa
  Hinv <- pseudoinverse(H)

  # calcula w
  W <- Hinv %*% Yin
```

```
  return(list(W,H,Z))

}
# funcao para calcular saida da ELM
YELM <- function(Xin,Z,W){
  n <- dim(Xin)[2]
  Xaug <- cbind(Xin,1)
  H <- tanh(Xaug %*% Z)
  Yhat <- sign(H%*%W) # retorna -1,0 ou 1 de acordo com o sinal do numero calculado
  return(Yhat)

}

@

\begin{itemize}

\item Base "2dnormals"
\newline

<<echo=FALSE>>=

x <- as.matrix(cd3$x)
y <- as.matrix(cd3$classes)
class(y) <- "numeric"
y[y==2]<-(-1)
y[y==1]<- (1)

@

<<echo=false>>=

# selecionando aleatoriamente 70% das amostras de treino

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))
}

@

<<echo=false>>=
# training the model
p <- 50
model <- treinaELM(x,y,p)

W <- model[[1]]
H <- model[[2]]
Z <- model[[3]]

# calculando Yhat de treino

yhattrain <- YELM(x,Z,W)

acc <- y - yhattrain

acc <- length(acc[acc==0])/dim(y)[1]
accTrain <- as.numeric(confusionMatrix(factor(y),factor(yhattrain))$overall[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]<- YELM(X,Z,W)

  }

}

plot(x[,1],x[,2],col = 'red', xlim = c(-2,2), ylim = c(-2,2))
par(new=T)
#plot(Xblue[,1],Xblue[,2],col = 'blue', xlim = c(0,6), ylim = c(0,6),xlab = '',ylab = '')
#par(new=T)
@

\begin{figure}[!htb]
\begin{center}
<<echo=F,fig=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='')
@
\end{center}
\caption{Contorno da Superficie em 2D}
\label{Contorno da Superficie em 2D}
\end{figure}

\begin{figure}[!htb]
\begin{center}
<<echo=F,fig=TRUE>>=
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)
print(accTrain)
```

```
@
\end{center}
\caption{Contorno da Superficie em 3D}
\label{Contorno da Superficie em 3D}
\end{figure}



\end{itemize}

\end{document}
```