Redes Neurais Artificiais

Exercício 5 - Máquinas de Aprendizado Extremo - ELM

Vítor Gabriel Reis Caitité - 2016111849

1/17/2021

Função que calcula a saída de uma rede ELM

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

```
YELM<-function(xin, Z, W, par){
  n<-dim(xin)[2]

  # Adiciona ou não termo de polarização
  if(par == 1) {
    xin<-cbind(1, xin)
  }
  H<-tanh(xin%*%Z)
  y_hat<-sign(H %*% W)
  return(y_hat)
}
```

Treinamento da rede ELM

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

```
library("corpcor")

trainELM <- function(xin, yin, p, par){
  n <- dim(xin) [2] # Dimensão da entrada

  #Adiciona ou não o termo de polarização
  if(par == 1){
    xin<-cbind(1,xin)
    Z<-replicate(p, runif(n+1, -0.5, 0.5))
  }
  else{
    Z<-replicate(p, runif(n, -0.5, 0.5))
  }
  H<-tanh(xin %*% Z)

  W<-pseudoinverse(H)%*%yin
  #W<-(solve(t(H) %*% H) %*% t(H)) %*% yin

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

Enunciado Exercício 5

O objetivo dos exercícios desta semana é ter um primeiro contato com classificadores não lineares, especificamente ELMs. Qualquer linguagem de programação pode ser utilizada, embora os códigos apresentados sejam para R. Utilizando o pacote em R mlbench, devem ser geradas as seguintes bases de dados:

• mlbench.2dnormals(200)

• mlbench.xor(100)

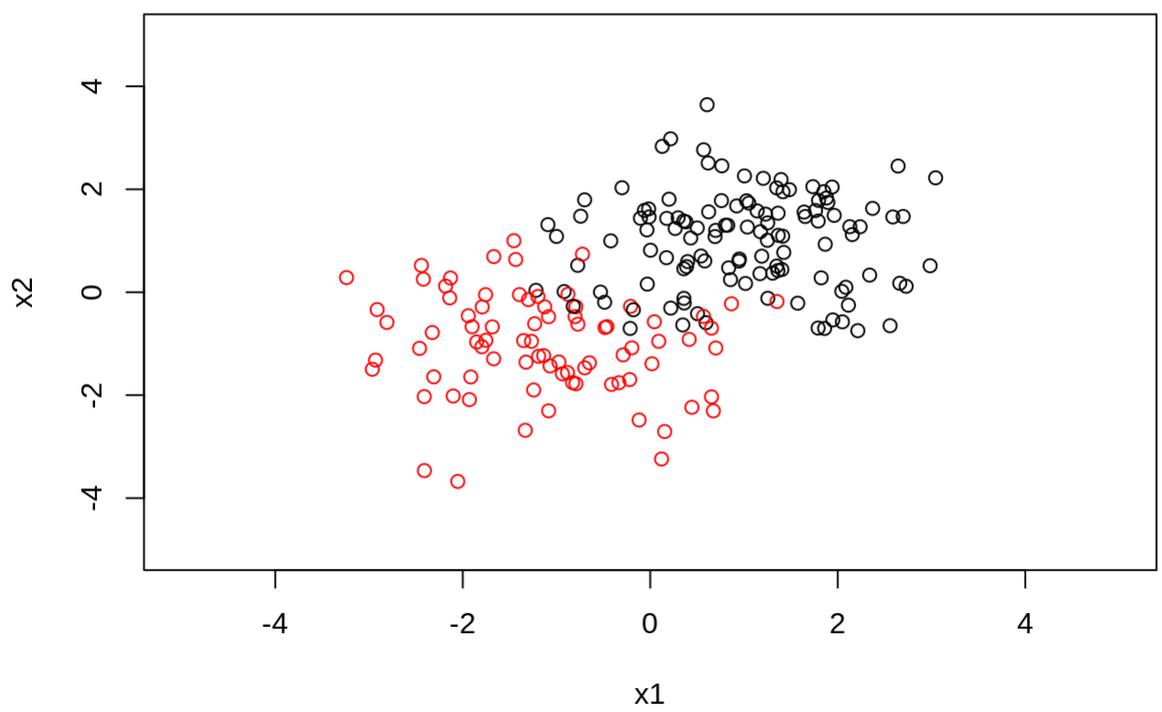
• mlbench.circle(100)

• mlbench.spirals(100,sd = 0.05)

Em seguida, para cada uma das bases, devem ser construídas diferentes ELMs (pelo menos 3 para cada base de dados) variando-se o número de neurônios, por exemplo p = 5, 10, 30. Como as bases de dados são bidimensionais, é possível visualizar a superfície de separação obtida para cada uma das ELMs, que deve ser mostrada no relatório. A superfície pode ser obtida conforme os códigos fornecidos no exercício anterior. Pede-se que, para cada uma das bases, seja feita uma breve discussão a respeito de qual número de neurônios levou aos melhores resultados. Neste primeiro momento, a avaliação da qualidade dos resultados será exclusivamente visual. O relatório deve ser em pdf, incluindo as figuras, a discussão e os códigos utilizados.

Base de dados mlbench.2dnormals

```
# Dados do problema:

rm(list=ls())
library(mlbench)
source("~/Documents/UFMG/9/Redes Neurais/exemplos/trainELM.R")
source("~/Documents/UFMG/9/Redes Neurais/exemplos/YELM.R")

data <- mlbench.2dnormals(200)

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



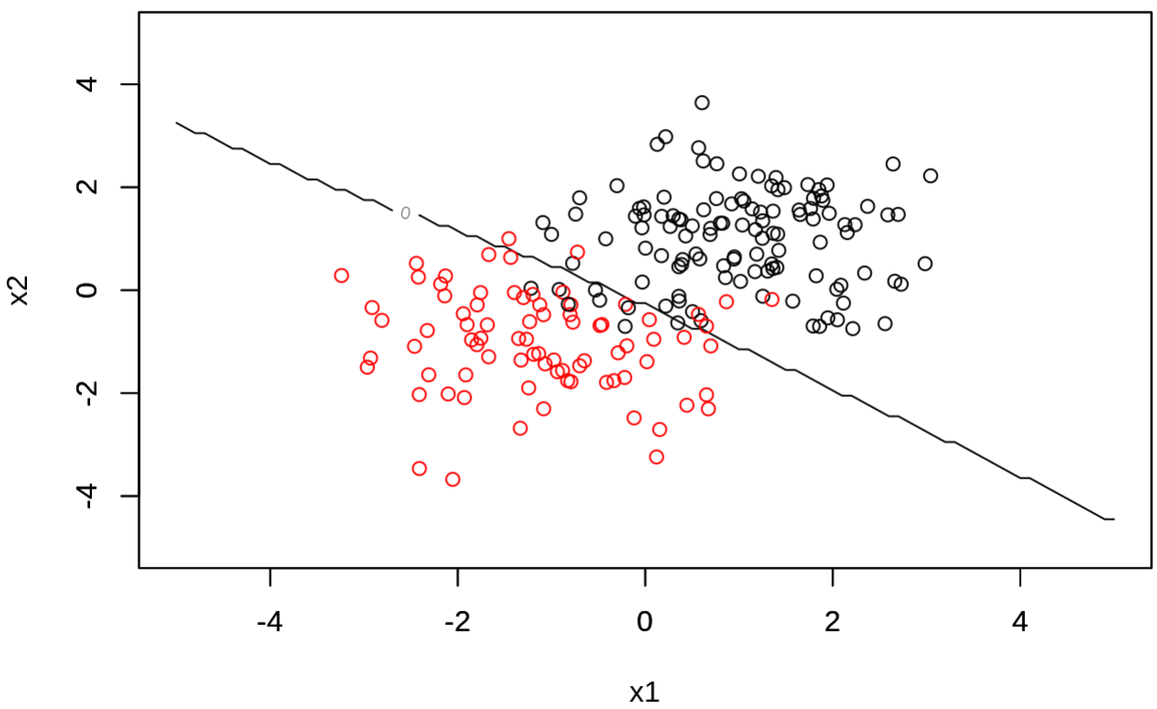

```
yin <- rep(0,200)
for (count in 1:length(class)) {
  if (class[count] == 1 ){
    yin[count] = -1
  }
  else if(class[count] == 2){
    yin[count] = 1
  }
}
```

ELM com número de neurônios p = 5:

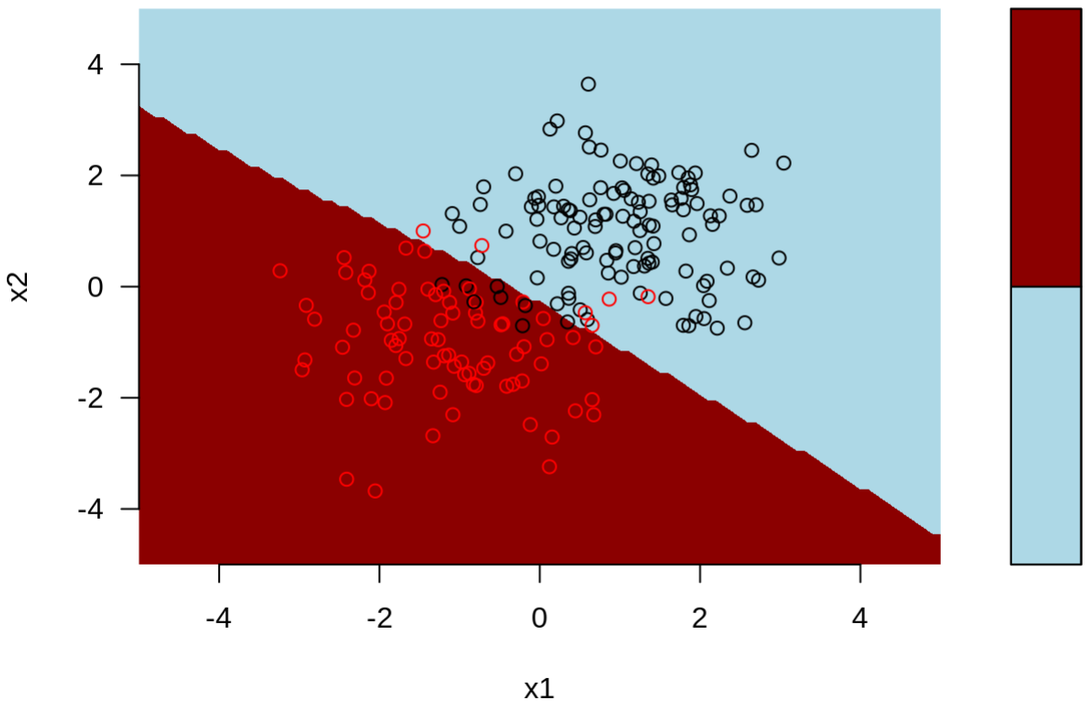
```
p<-5
retlist<-trainELM(xin,yin,p,1)

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

y_hat <- YELM(xin, Z, W, 1)
err_train <- sum((yin - y_hat)^2)/4 #divide-se por 4 pelo fato de a distância entre -1 e
 1 ser 2
paste("Erros:", err_train)
```

```
## [1] "Erros: 14"
```

```
# 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]<-YELM(x1x2, Z, W, 1)
  }
}

contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-5,5), ylim=c(-5,5), xlab = "x1", ylab
 = "x2")
par(new=T)
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 = "x
1", ylab = "x2", col = cols, axes=F, plot.axes = { points(xin[,1], xin[,2],col=data$clas
ses, xlim=c(-5,5), ylim=c(-5,5)); axis(1); axis(2) })
```

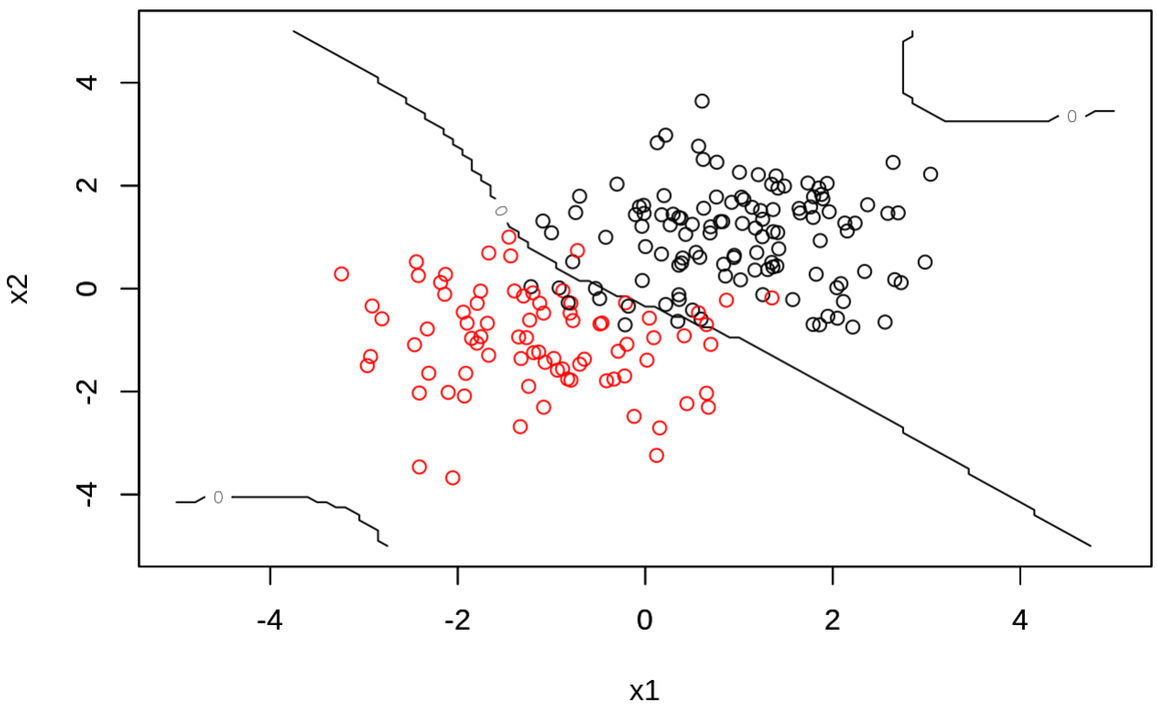


ELM com número de neurônios p = 10:

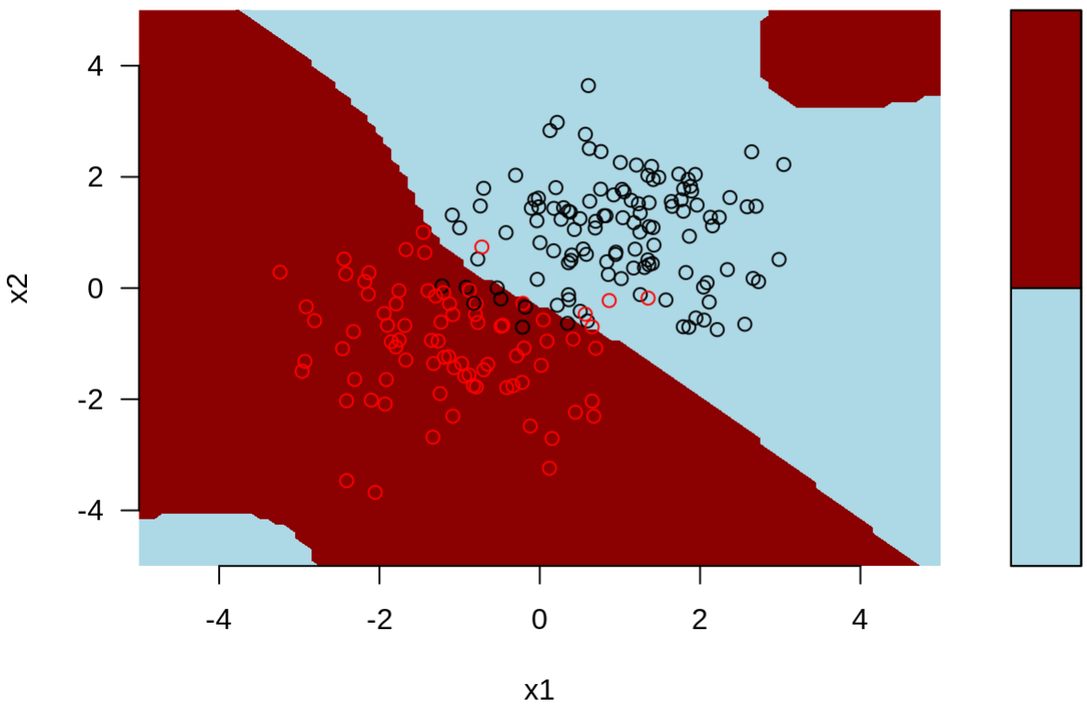
```
p<-10
retlist<-trainELM(xin,yin,p,1)

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

y_hat <- YELM(xin, Z, W, 1)
err_train <- sum((yin - y_hat)^2)/4 #divide-se por 4 pelo fato de a distância entre -1 e
 1 ser 2
paste("Erros:", err_train)
```

```
## [1] "Erros: 13"
```

```
# 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]<-YELM(x1x2, Z, W, 1)
  }
}

contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-5,5), ylim=c(-5,5), xlab = "x1", ylab
 = "x2")
par(new=T)
plot(xin[,1], xin[,2], col=data$classes, xlim=c(-5,5), ylim=c(-5,5), ylab = "", xlab =
"")
```




```
filled.contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-5,5), ylim=c(-5,5), xlab = "x
1", ylab = "x2", col = c('lightblue', 'darkred'), axes=F, plot.axes = { points(xin[,1],
 xin[,2],col=data$classes, xlim=c(-5,5), ylim=c(-5,5)); axis(1); axis(2) })
```

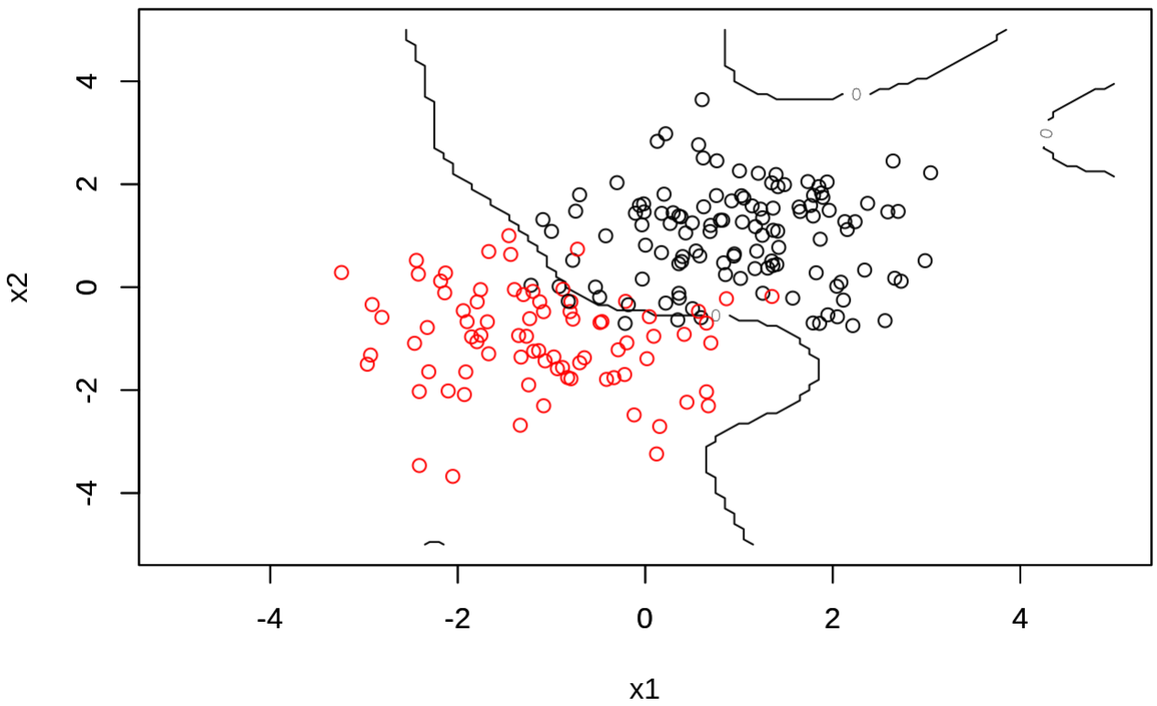


ELM com número de neurônios p = 30:

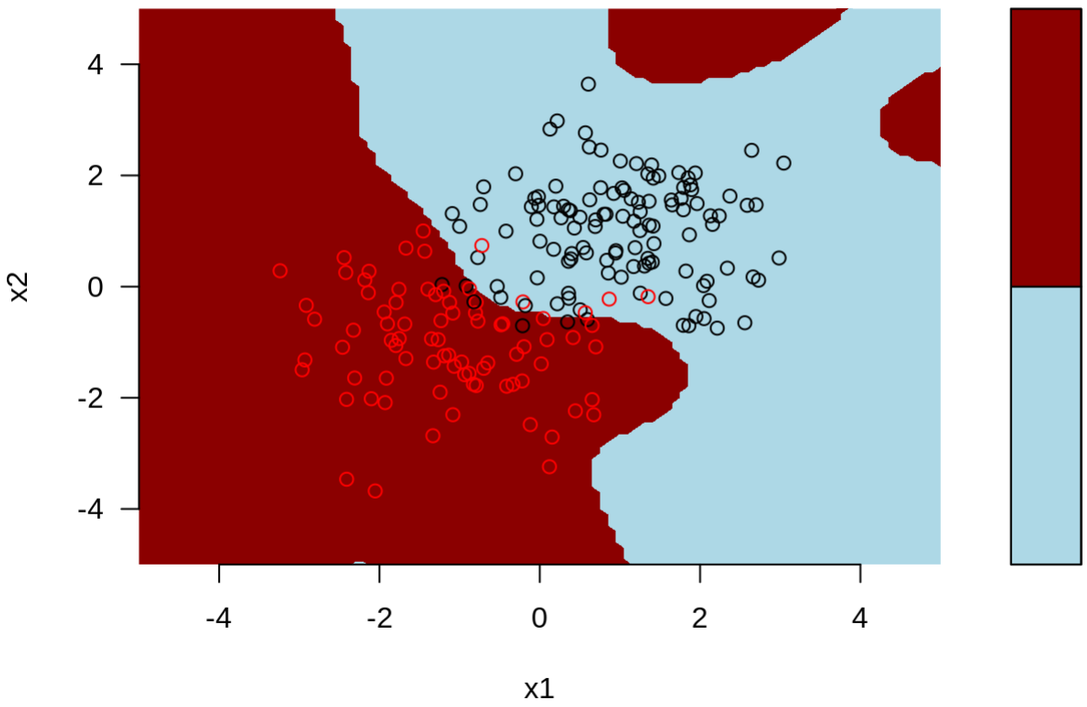
```
p<-30
retlist<-trainELM(xin,yin,p,1)

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

y_hat <- YELM(xin, Z, W, 1)
err_train <- sum((yin - y_hat)^2)/4 #divide-se por 4 pelo fato de a distância entre -1 e
 1 ser 2
paste("Erros:", err_train)
```

```
## [1] "Erros: 11"
```

```
# 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]<-YELM(x1x2, Z, W, 1)
  }
}

contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-5,5), ylim=c(-5,5), xlab = "x1", ylab
 = "x2")
par(new=T)
plot(xin[,1], xin[,2], col=data$classes, xlim=c(-5,5), ylim=c(-5,5), ylab = "", xlab =
"")
```




```
filled.contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-5,5), ylim=c(-5,5), xlab = "x
1", ylab = "x2", col = c('lightblue', 'darkred'), axes=F, plot.axes = { points(xin[,1],
 xin[,2],col=data$classes, xlim=c(-5,5), ylim=c(-5,5)); axis(1); axis(2) })
```

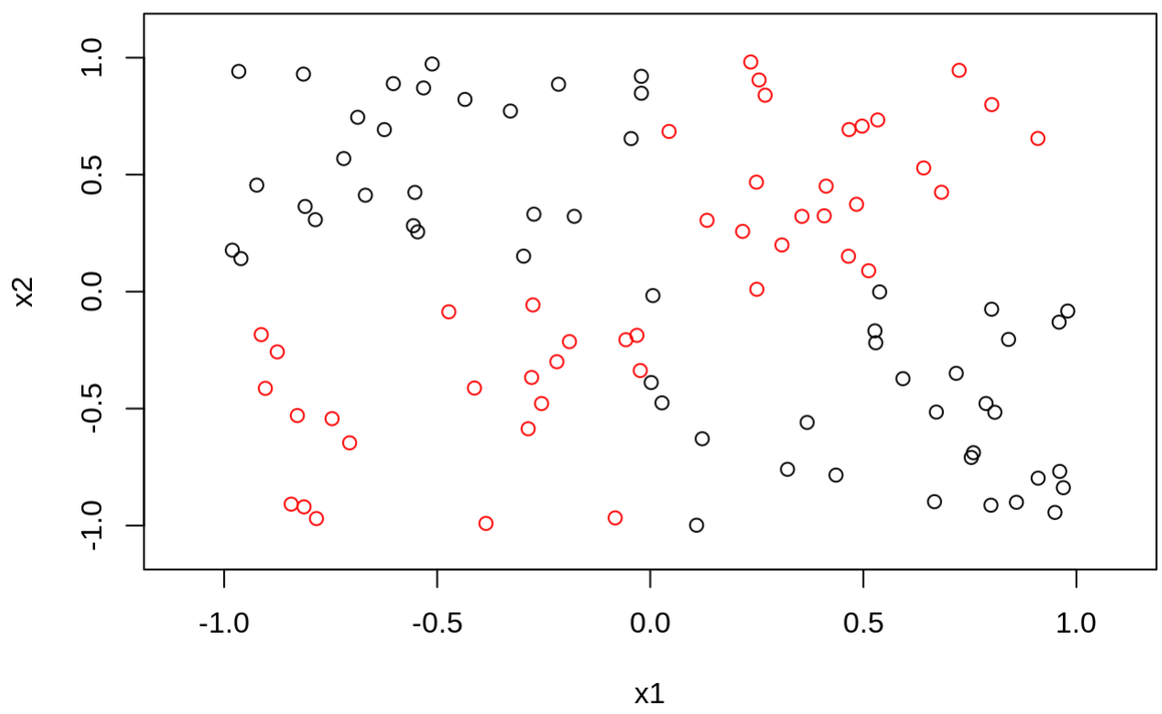


Discussão:

Pôde-se notar que ao aumentar o número de neurônios não necessariamente o erro melhorou. O erro um pouco mais alto para os dados dessa base de dados se deve principalmente por existir alguns dados de classes diferentes cuja a separação espacial praticamente não existe, como pode ser notado nas figuras anteriores. Fazendo uma análise estritamente visual para essa classe de dados, aparentemente pode-se inferir que o classificador gerado com uma ELM de 5 neurônios está o mais próximo do ideal. Enquanto isso, nos casos onde o número de nerônios foi 10 e 30 já se pode notar sinais de overfitting.

Base de dados mlbench.xor

```
rm(list=ls())
library(mlbench)
source("~/Documents/UFMG/9/Redes Neurais/exemplos/trainELM.R")
source("~/Documents/UFMG/9/Redes Neurais/exemplos/YELM.R")

data <- mlbench.xor(100)
xin<-data[["x"]]
class<-data[["classes"]]
plot(xin[,1], xin[,2], col=data$classes, xlim=c(-1.1, 1.1), ylim=c(-1.1, 1.1), ylab = "x
2", xlab = "x1")
```



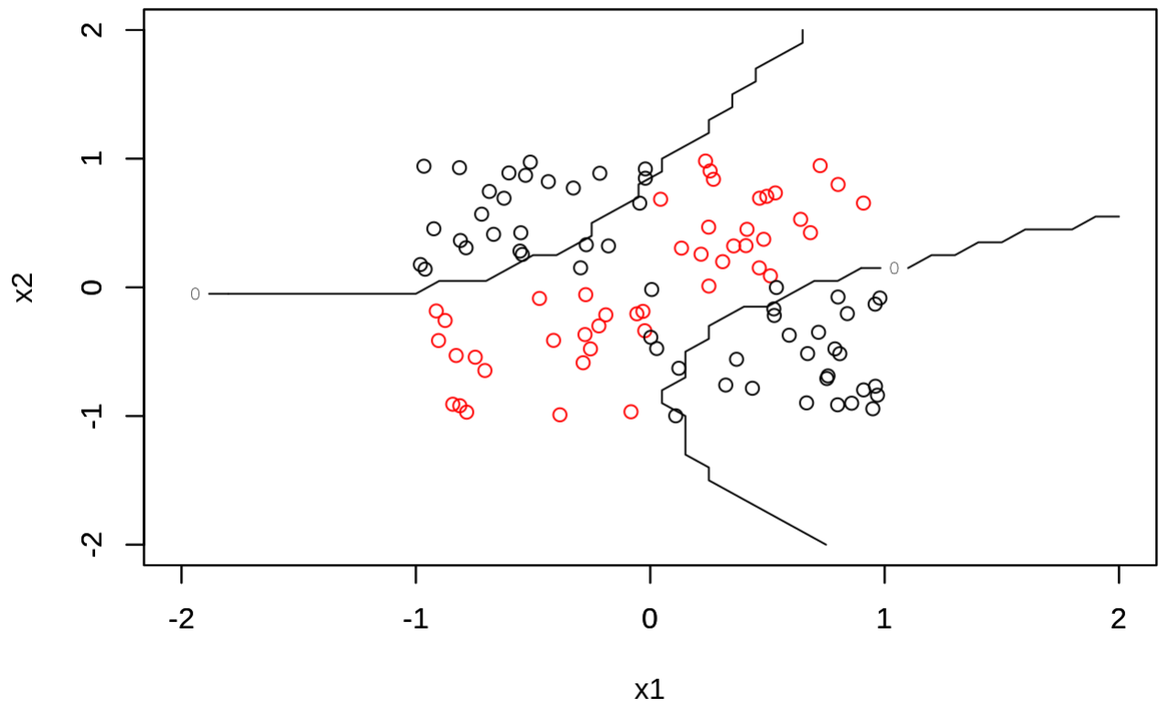

```
yin <- rep(0,100)
for (count in 1:length(class)) {
  if (class[count] == 1 ){
    yin[count] = -1
  }
  else if(class[count] == 2){
    yin[count] = 1
  }
}
```

ELM com número de neurônios p = 5:

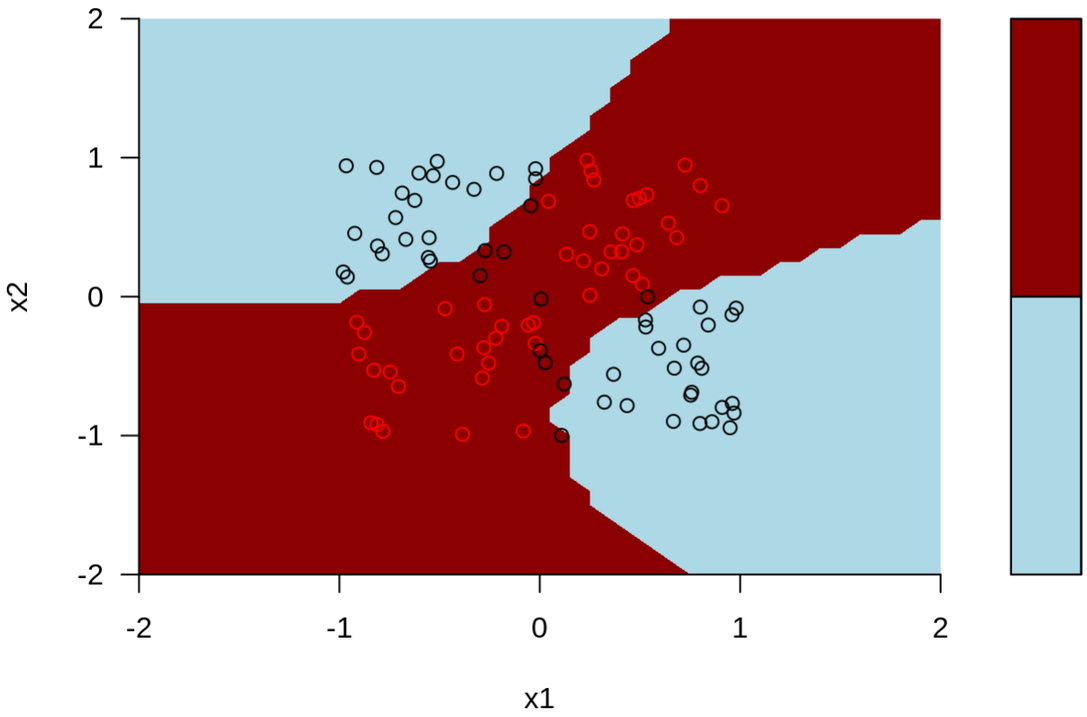
```
p<-5
retlist<-trainELM(xin,yin,p,1)

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

y_hat <- YELM(xin, Z, W, 1)
err_train <- sum((yin - y_hat)^2)/4 #divide-se por 4 pelo fato de a distância entre -1 e
 1 ser 2
paste("Erros:", err_train)
```

```
## [1] "Erros: 9"
```

```
# Plotando superfície
seqx1x2 <- seq(-2, 2, 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]<-YELM(x1x2, Z, W, 1)
  }
}

contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-2,2), ylim=c(-2,2), xlab = "x1", ylab
 = "x2")
par(new=T)
plot(xin[,1], xin[,2], col=data$classes, xlim=c(-2,2), ylim=c(-2,2), ylab = "", xlab =
"")
```




```
filled.contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-2,2), ylim=c(-2,2), xlab = "x
1", ylab = "x2", col = c('lightblue', 'darkred'), axes=F, plot.axes = { points(xin[,1],
 xin[,2],col=data$classes, xlim=c(-2,2), ylim=c(-2,2)); axis(1); axis(2) })
```

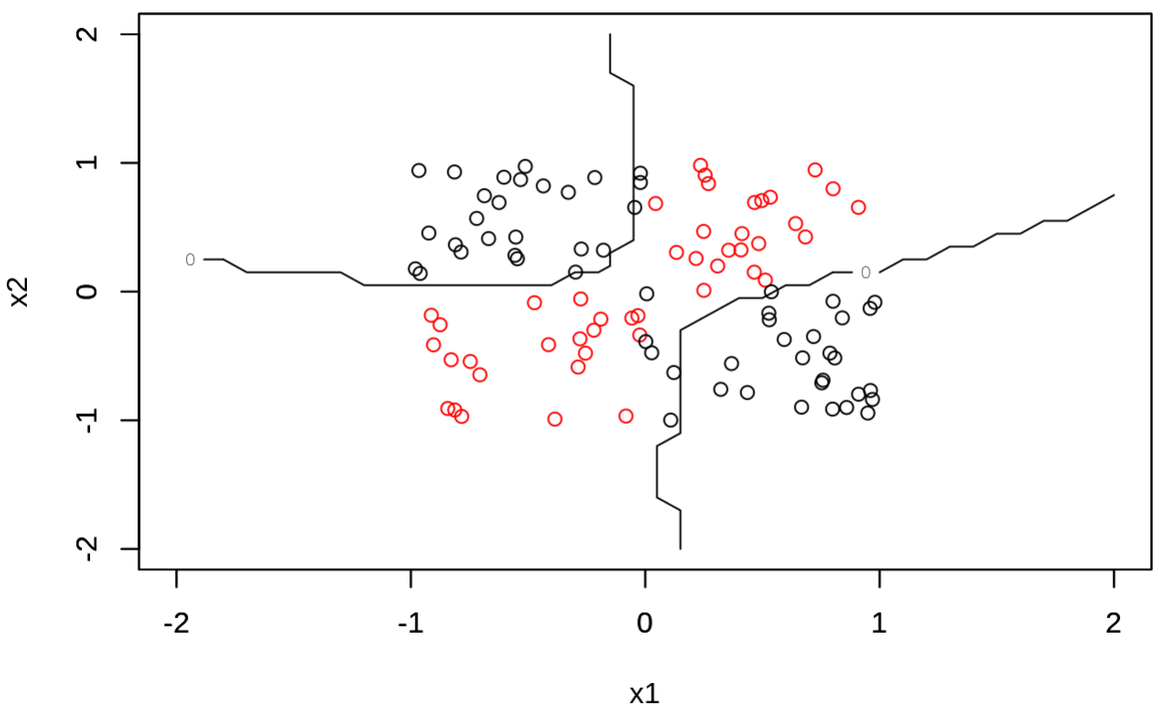


ELM com número de neurônios p = 10:

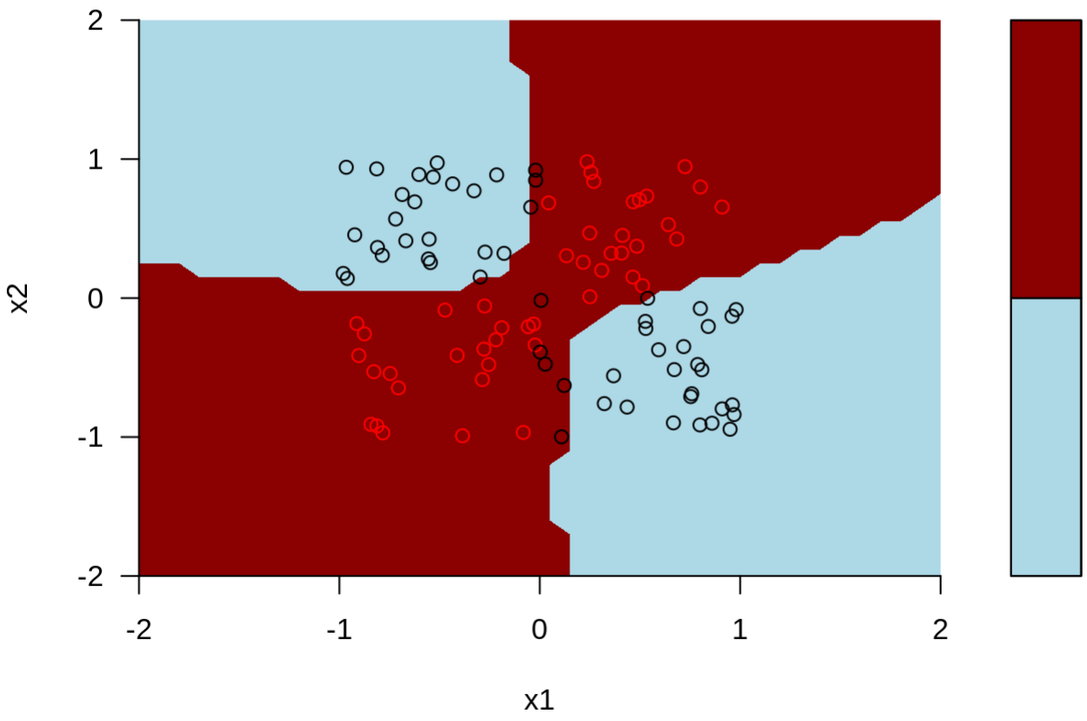
```
p<-10
retlist<-trainELM(xin,yin,p,1)

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

y_hat <- YELM(xin, Z, W, 1)
err_train <- sum((yin - y_hat)^2)/4 #divide-se por 4 pelo fato de a distância entre -1 e
 1 ser 2
paste("Erros:", err_train)
```

```
## [1] "Erros: 7"
```

```
# Plotando superfície
seqx1x2 <- seq(-2, 2, 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]<-YELM(x1x2, Z, W, 1)
  }
}

contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-2,2), ylim=c(-2,2), xlab = "x1", ylab
 = "x2")
par(new=T)
plot(xin[,1], xin[,2], col=data$classes, xlim=c(-2,2), ylim=c(-2,2), ylab = "", xlab =
"")
```




```
filled.contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-2,2), ylim=c(-2,2), xlab = "x
1", ylab = "x2", col = c('lightblue', 'darkred'), axes=F, plot.axes = { points(xin[,1],
 xin[,2],col=data$classes, xlim=c(-2,2), ylim=c(-2,2)); axis(1); axis(2) })
```

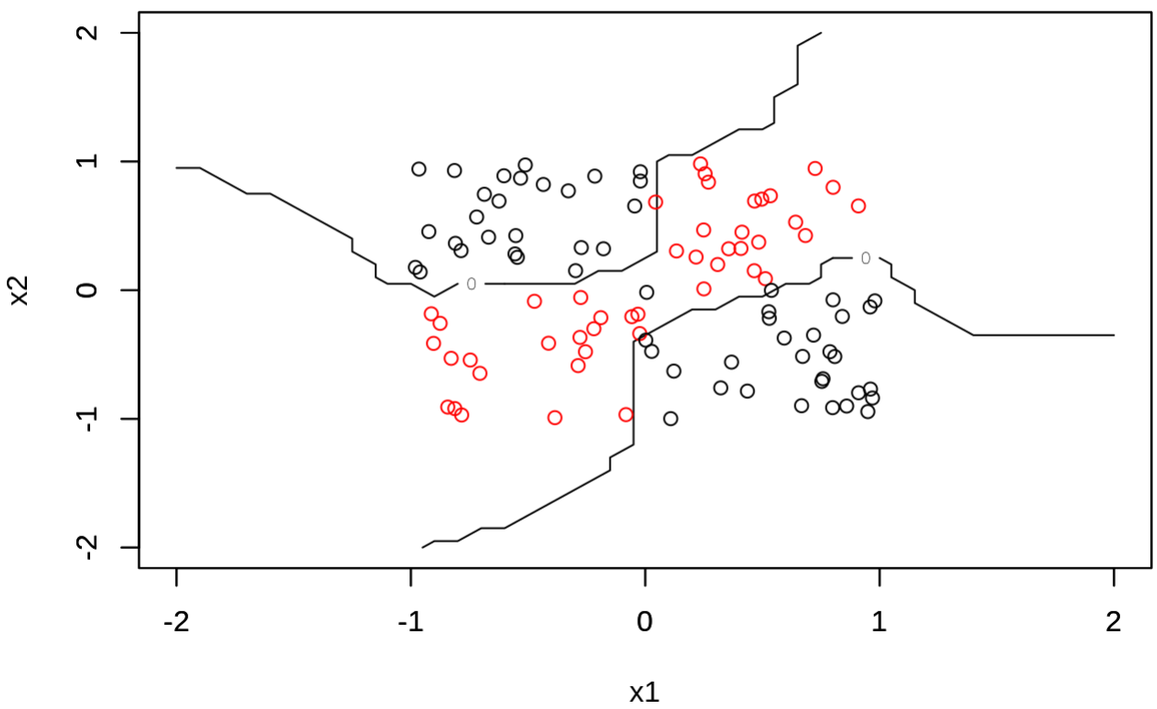


ELM com número de neurônios p = 30:

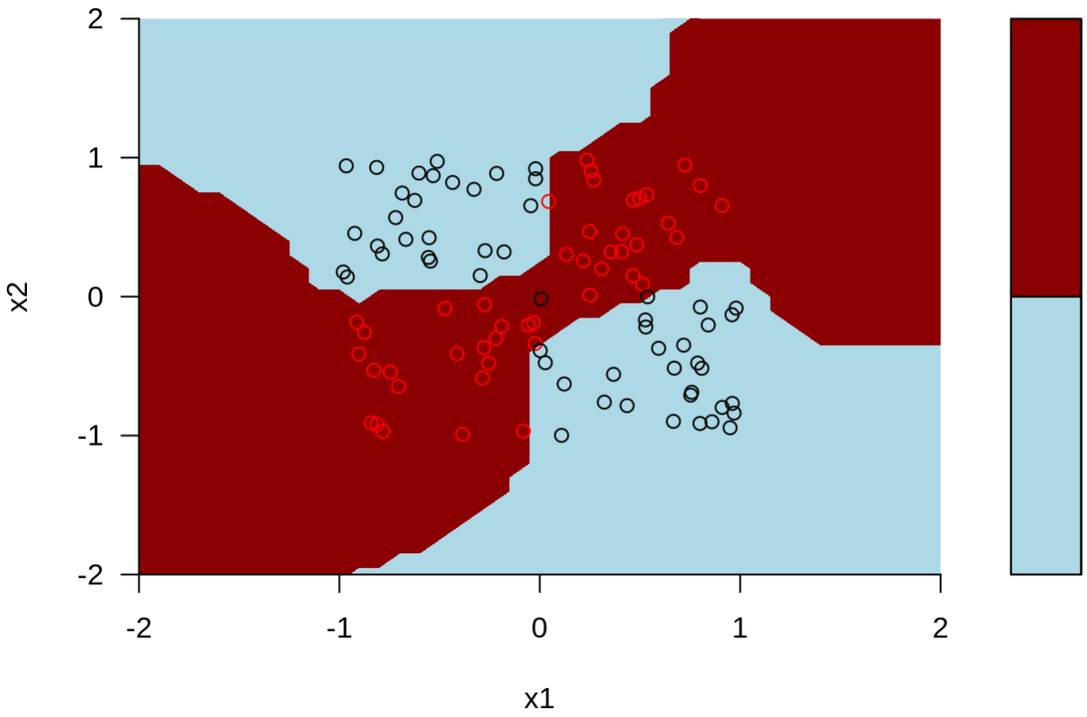
```
p<-30
retlist<-trainELM(xin,yin,p,1)

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

y_hat <- YELM(xin, Z, W, 1)
err_train <- sum((yin - y_hat)^2)/4 #divide-se por 4 pelo fato de a distância entre -1 e
 1 ser 2
paste("Erros:", err_train)
```

```
## [1] "Erros: 4"
```

```
# Plotando superfície
seqx1x2 <- seq(-2, 2, 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]<-YELM(x1x2, Z, W, 1)
  }
}

contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-2,2), ylim=c(-2,2), xlab = "x1", ylab
 = "x2")
par(new=T)
plot(xin[,1], xin[,2], col=data$classes, xlim=c(-2,2), ylim=c(-2,2), ylab = "", xlab =
"")
```




```
filled.contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-2,2), ylim=c(-2,2), xlab = "x
1", ylab = "x2", col = c('lightblue', 'darkred'), axes=F, plot.axes = { points(xin[,1],
 xin[,2],col=data$classes, xlim=c(-2,2), ylim=c(-2,2)); axis(1); axis(2) })
```

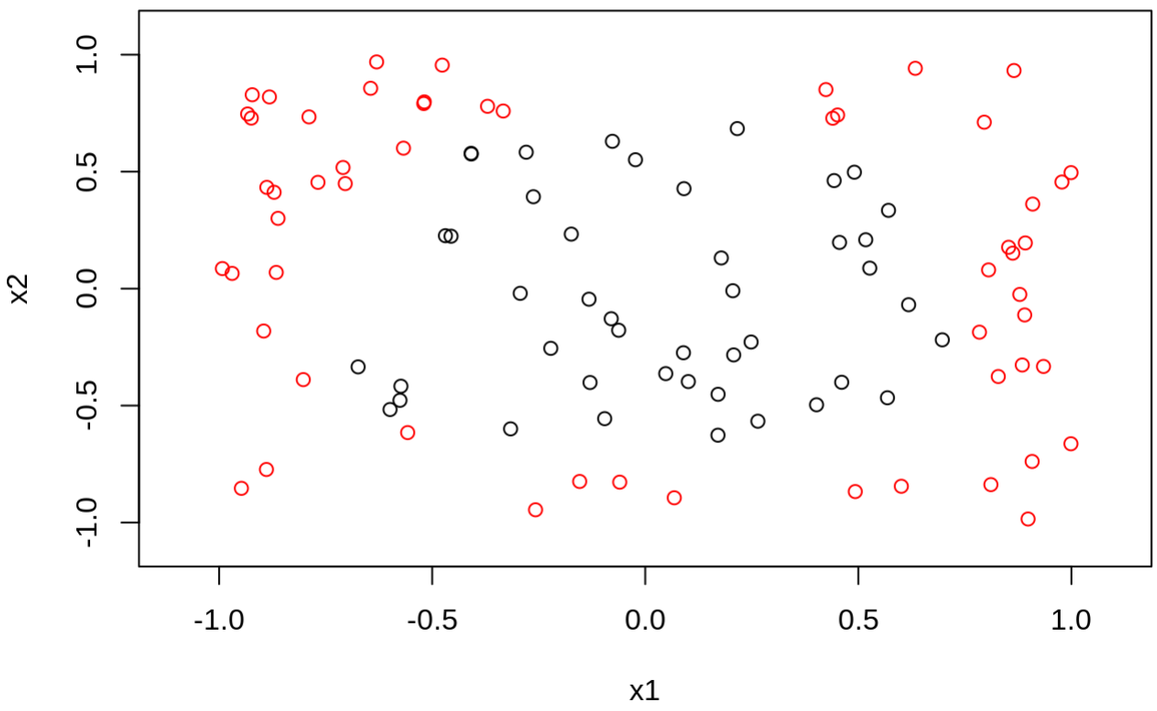


Discussão:

Visualmente pode-se notar que essa base de dados necessita de um classificador mais complexo que o anterior. Observou-se que o classificador gerado a partir de uma ELM de 5 neurônios não foi suficiente para classificar os dados como esperado (pode-se notar um número elevado de erros de classificação nesse caso). Já o classificador ELM com 10 neurônios, pode-se observar que ele gerou uma superfície de separação capaz de classificar os dados como esperamos. Por fim, a ELM com 30 neurônios, apesar de ser a que obteve menor número de erros, nela já é possível notar sinais de overfitting, ou seja o modelo se ajustou demais ao dados de treinamento. Isso pode acabar gerando problemas na classificação de diferentes dados.

Base de dados mlbench.circle

```
rm(list=ls())
library(mlbench)
source("~/Documents/UFMG/9/Redes Neurais/exemplos/trainELM.R")
source("~/Documents/UFMG/9/Redes Neurais/exemplos/YELM.R")

data <- mlbench.circle(100)
xin<-data[["x"]]
class<-data[["classes"]]
plot(xin[,1], xin[,2], col=data$classes, xlim=c(-1.1, 1.1), ylim=c(-1.1, 1.1), ylab = "x
2", xlab = "x1")
```



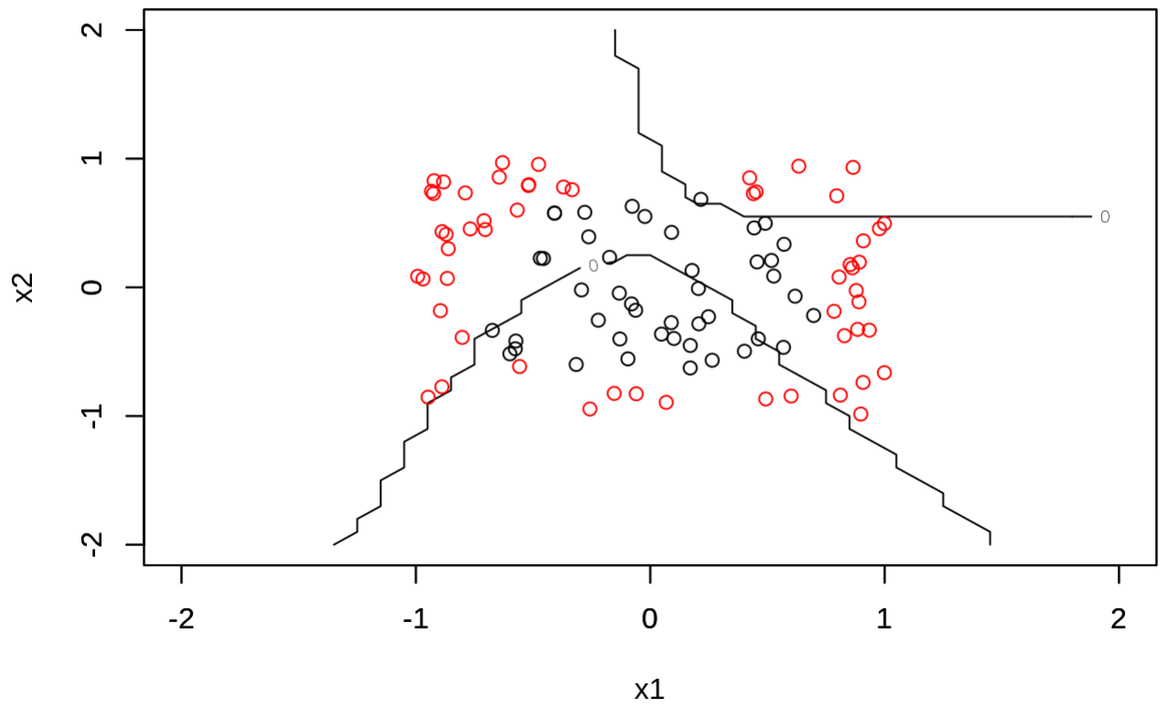

```
yin <- rep(0,100)
for (count in 1:length(class)) {
  if (class[count] == 1 ){
    yin[count] = -1
  }
  else if(class[count] == 2){
    yin[count] = 1
  }
}
```

ELM com número de neurônios p = 5:

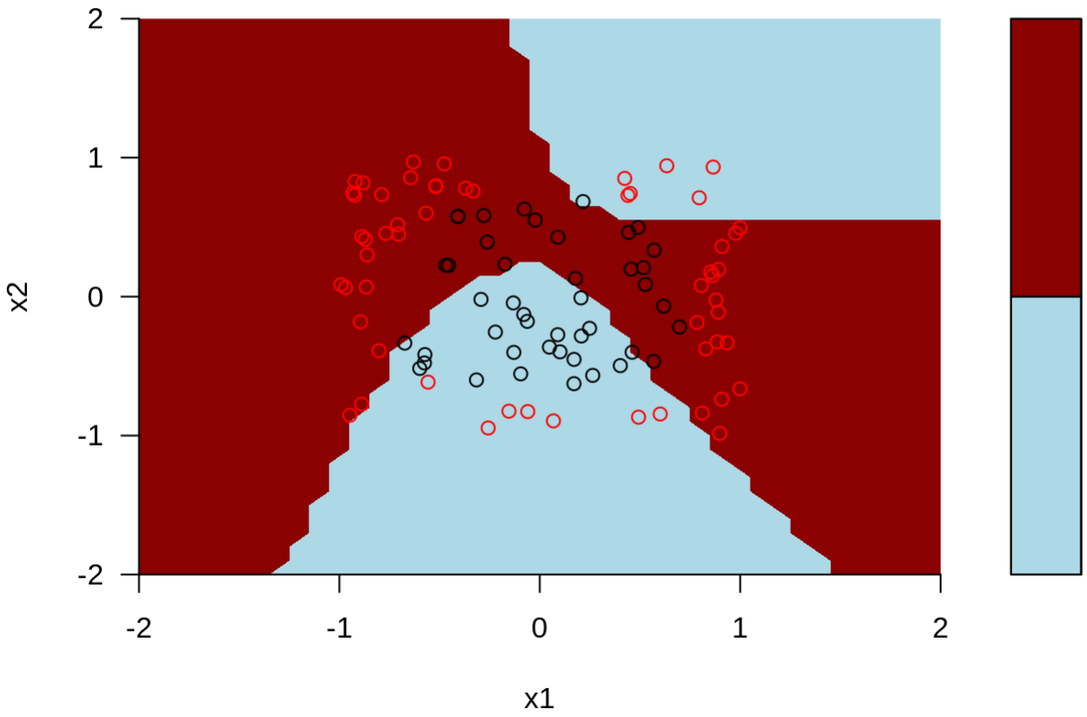
```
p<-5
retlist<-trainELM(xin,yin,p,1)

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

y_hat <- YELM(xin, Z, W, 1)
err_train <- sum((yin - y_hat)^2)/4 #divide-se por 4 pelo fato de a distância entre -1 e
 1 ser 2
print(err_train)
```

```
## [1] 33
```

```
# Plotando superfície
seqx1x2 <- seq(-2, 2, 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]<-YELM(x1x2, Z, W, 1)
  }
}

contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-2,2), ylim=c(-2,2), xlab = "x1", ylab
 = "x2")
par(new=T)
plot(xin[,1], xin[,2], col=data$classes, xlim=c(-2,2), ylim=c(-2,2), ylab = "", xlab =
"")
```




```
filled.contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-2,2), ylim=c(-2,2), xlab = "x
1", ylab = "x2", col = c('lightblue', 'darkred'), axes=F, plot.axes = { points(xin[,1],
 xin[,2],col=data$classes, xlim=c(-2,2), ylim=c(-2,2)); axis(1); axis(2) })
```

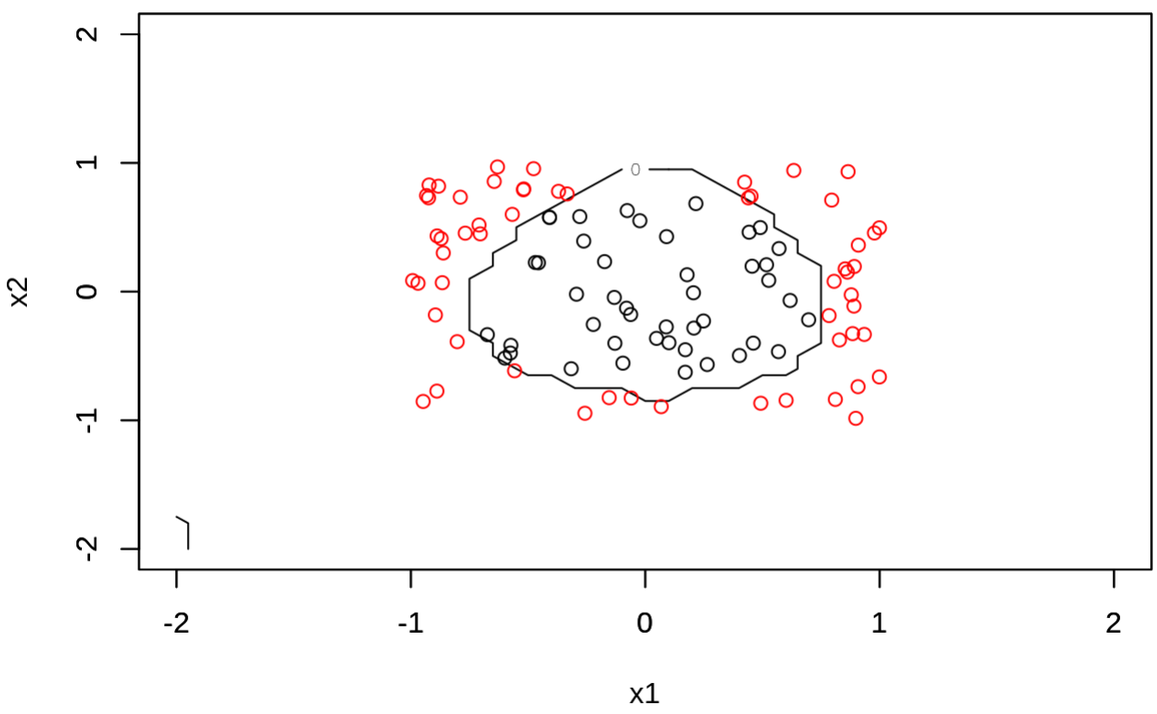


ELM com número de neurônios p = 10:

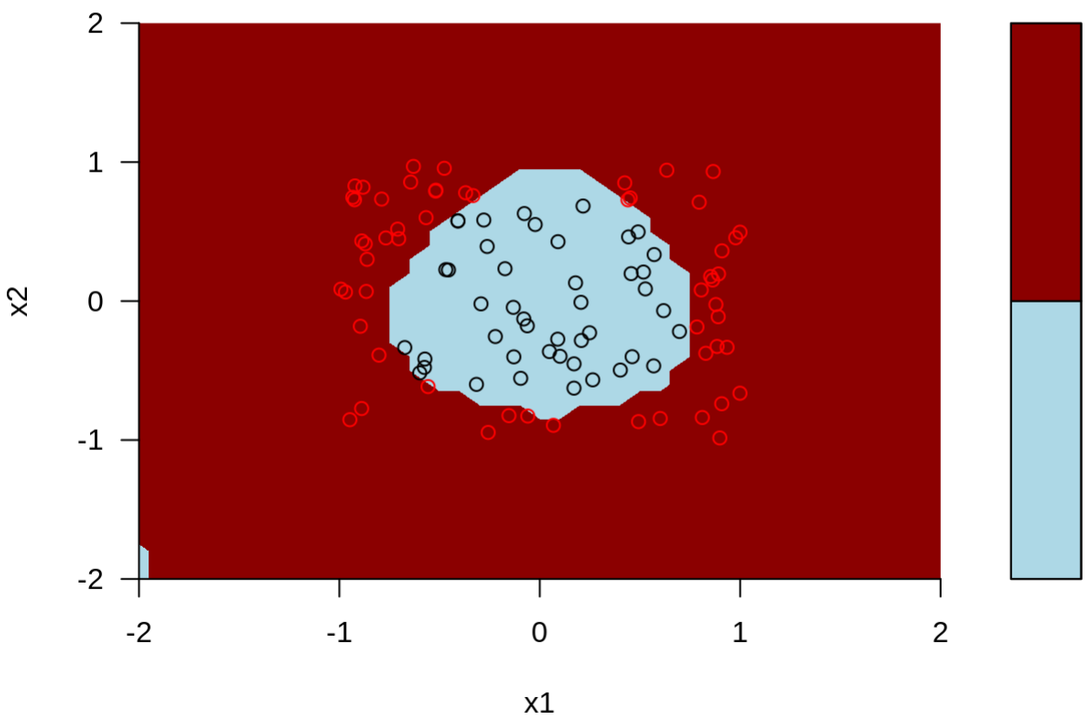
```
p<-10
retlist<-trainELM(xin,yin,p,1)

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

y_hat <- YELM(xin, Z, W, 1)
err_train <- sum((yin - y_hat)^2)/4 #divide-se por 4 pelo fato de a distância entre -1 e
 1 ser 2
paste("Erros:", err_train)
```

```
## [1] "Erros: 2"
```

```
# Plotando superfície
seqx1x2 <- seq(-2, 2, 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]<-YELM(x1x2, Z, W, 1)
  }
}

contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-2,2), ylim=c(-2,2), xlab = "x1", ylab
 = "x2")
par(new=T)
plot(xin[,1], xin[,2], col=data$classes, xlim=c(-2,2), ylim=c(-2,2), ylab = "", xlab =
"")
```




```
filled.contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-2,2), ylim=c(-2,2), xlab = "x
1", ylab = "x2", col = c('lightblue', 'darkred'), axes=F, plot.axes = { points(xin[,1],
 xin[,2],col=data$classes, xlim=c(-2,2), ylim=c(-2,2)); axis(1); axis(2) })
```

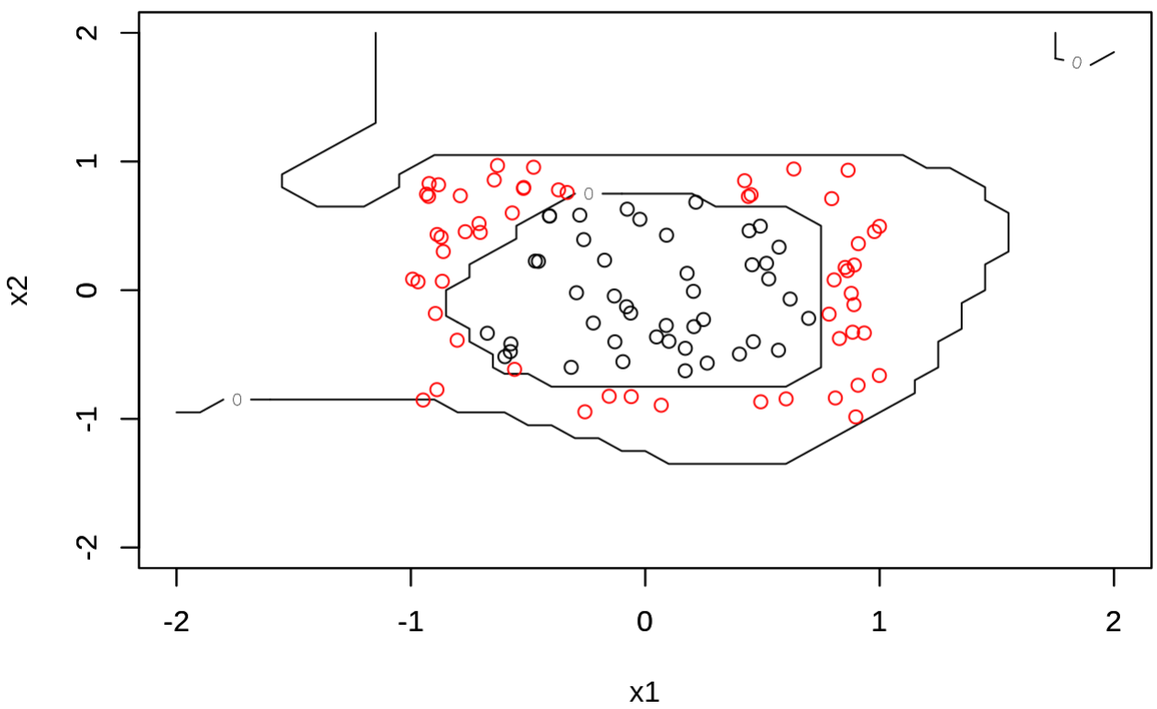


ELM com número de neurônios p = 30:

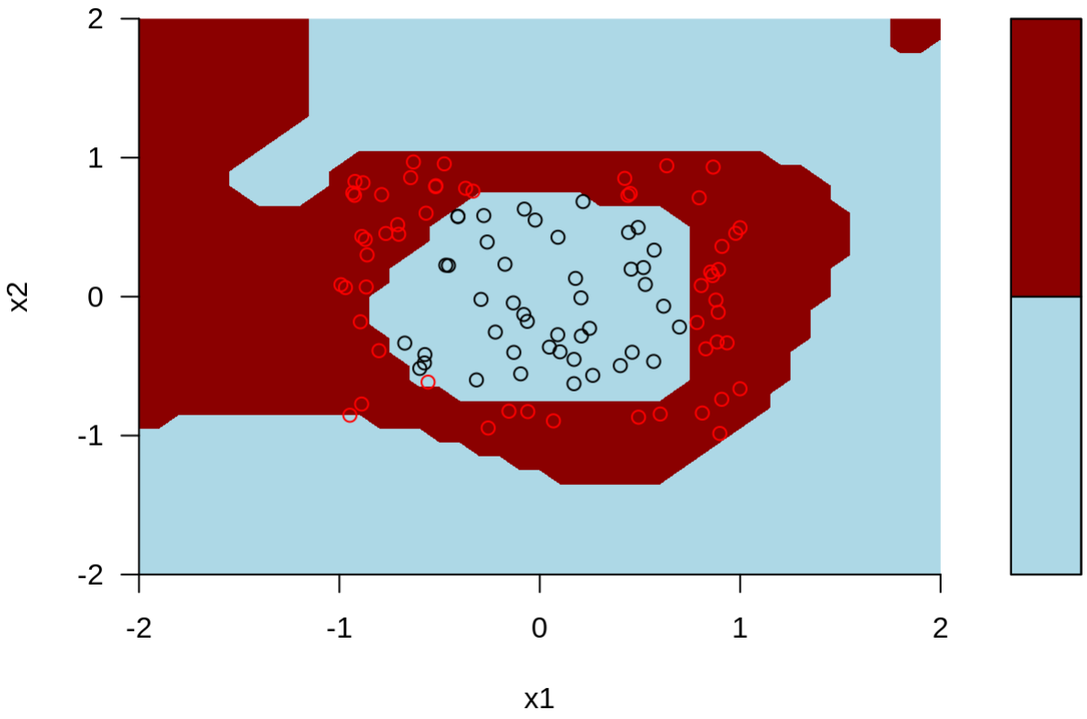
```
p<-30
retlist<-trainELM(xin,yin,p,1)

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

y_hat <- YELM(xin, Z, W, 1)
err_train <- sum((yin - y_hat)^2)/4 #divide-se por 4 pelo fato de a distância entre -1 e
 1 ser 2
paste("Erros:", err_train)
```

```
## [1] "Erros: 1"
```

```
# Plotando superfície
seqx1x2 <- seq(-2, 2, 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]<-YELM(x1x2, Z, W, 1)
  }
}

contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-2,2), ylim=c(-2,2), xlab = "x1", ylab
 = "x2")
par(new=T)
plot(xin[,1], xin[,2], col=data$classes, xlim=c(-2,2), ylim=c(-2,2), ylab = "", xlab =
"")
```




```
filled.contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-2,2), ylim=c(-2,2), xlab = "x
1", ylab = "x2", col = c('lightblue', 'darkred'), axes=F, plot.axes = { points(xin[,1],
 xin[,2],col=data$classes, xlim=c(-2,2), ylim=c(-2,2)); axis(1); axis(2) })
```

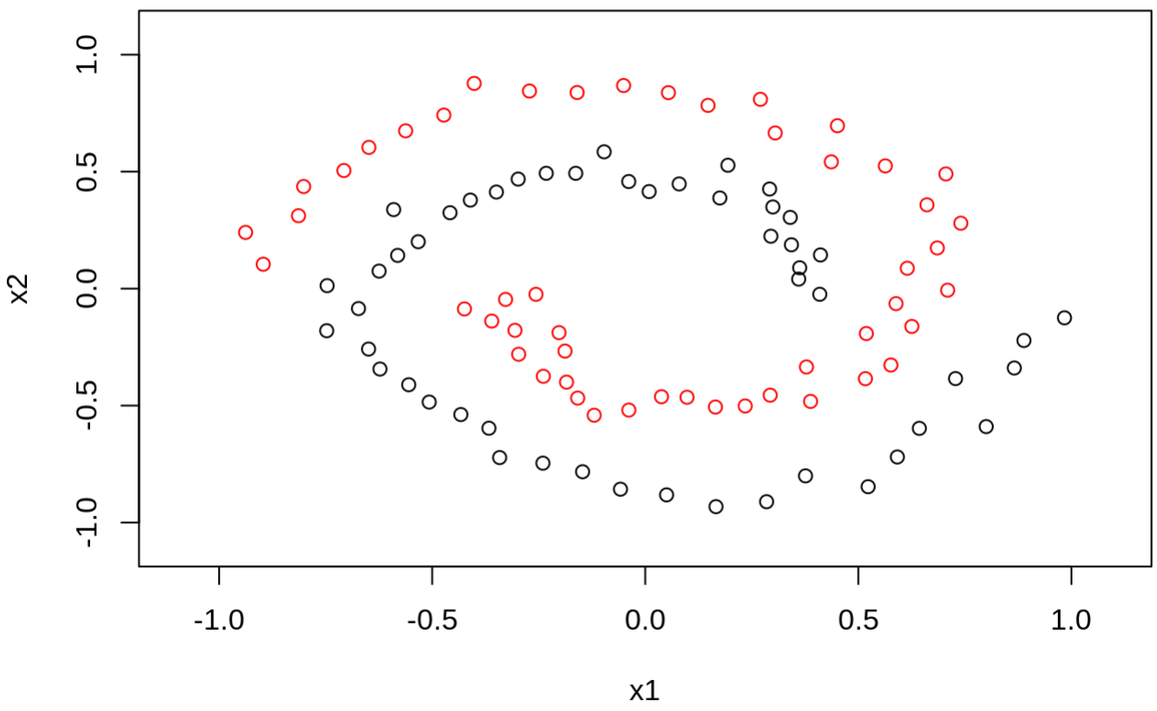


Discussão:

Observou-se que o classificador gerado a partir de uma ELM de 5 neurônios não foi suficiente para classificar os dados como esperado (pode-se notar um número elevado de erros de classificação nesse caso e uma superfície diferente da esperada). Já o classificador ELM com 10 neurônios, pode-se observar que ele gerou uma superfície de separação capaz de classificar os dados como esperamos e obteve um baixo número de erros. Por fim, a ELM com 30 neurônios, apesar de ser a que obteve menor número de erros, nela já é possível notar sinais de overfitting, ou seja o modelo se ajustou demais ao dados de treinamento. Isso pode acabar gerando problemas na classificação de diferentes dados.

Base de dados mlbench.spirals

```
rm(list=ls())
library(mlbench)
source("~/Documents/UFMG/9/Redes Neurais/exemplos/trainELM.R")
source("~/Documents/UFMG/9/Redes Neurais/exemplos/YELM.R")

data <- mlbench.spirals(100,sd = 0.05)
xin<-data[["x"]]
class<-data[["classes"]]
plot(xin[,1], xin[,2], col=data$classes, xlim=c(-1.1, 1.1), ylim=c(-1.1, 1.1), ylab = "x
2", xlab = "x1")
```



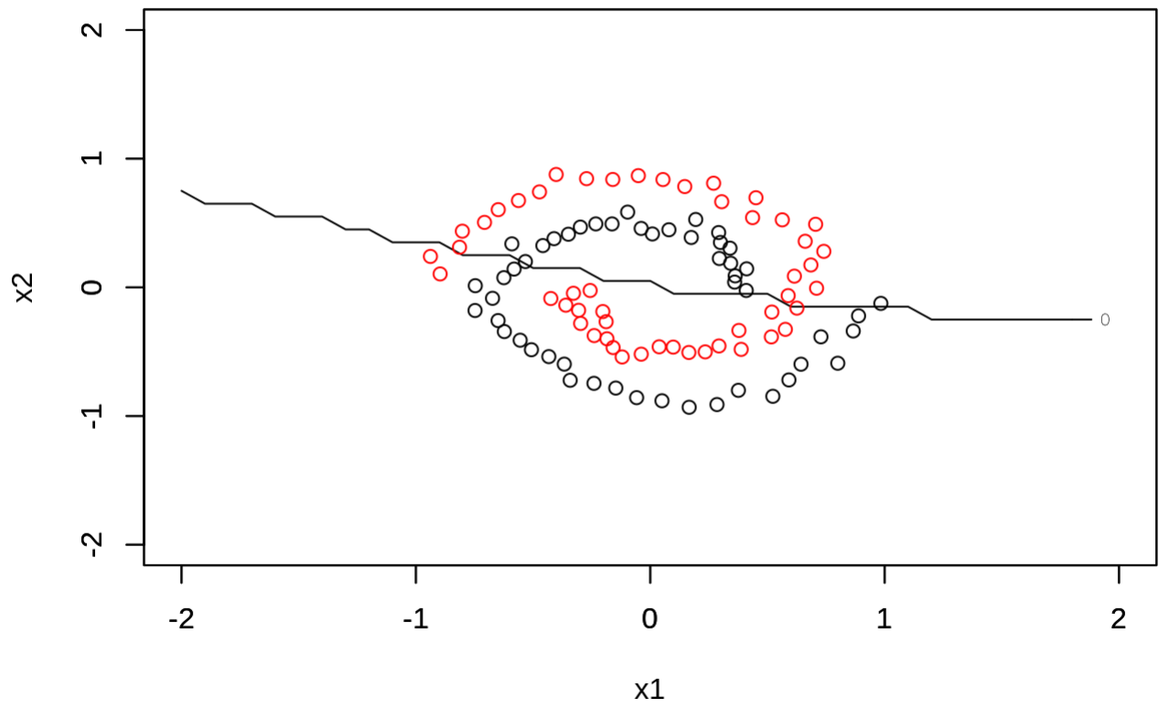

```
yin <- rep(0,100)
for (count in 1:length(class)) {
  if (class[count] == 1 ){
    yin[count] = -1
  }
  else if(class[count] == 2){
    yin[count] = 1
  }
}
```

ELM com número de neurônios p = 5:

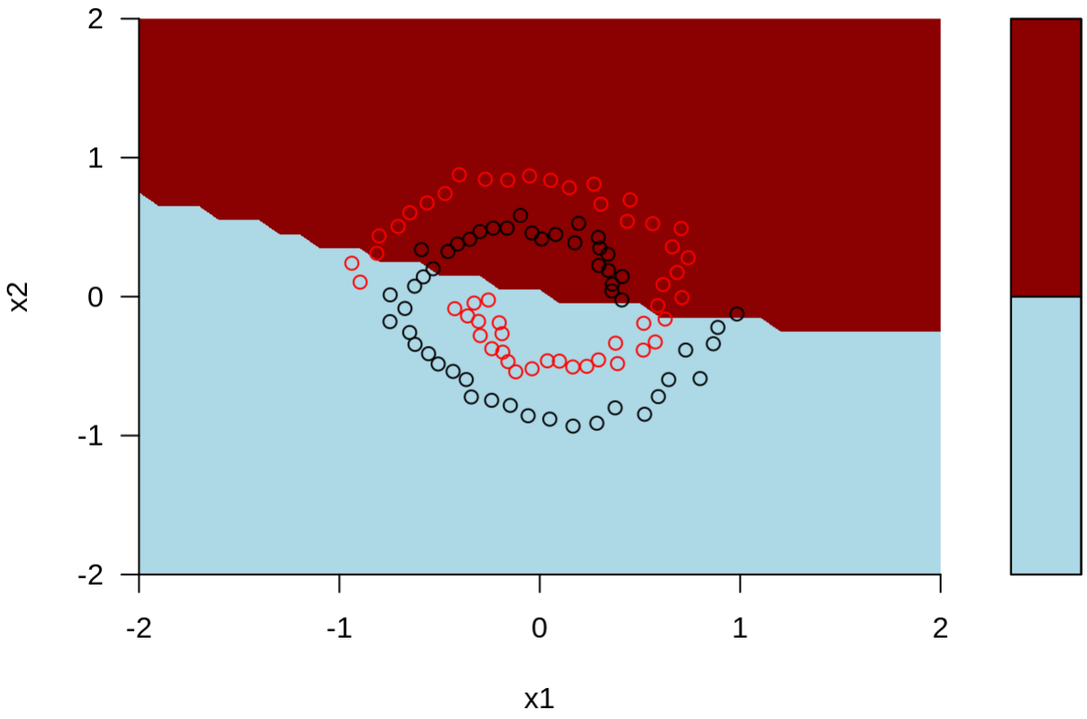
```
p<-5
retlist<-trainELM(xin,yin,p,1)

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

y_hat <- YELM(xin, Z, W, 1)
err_train <- sum((yin - y_hat)^2)/4 #divide-se por 4 pelo fato de a distância entre -1 e
 1 ser 2
print(err_train)
```

```
## [1] 50
```

```
# Plotando superfície
seqx1x2 <- seq(-2, 2, 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]<-YELM(x1x2, Z, W, 1)
  }
}

contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-2,2), ylim=c(-2,2), xlab = "x1", ylab
 = "x2")
par(new=T)
plot(xin[,1], xin[,2], col=data$classes, xlim=c(-2,2), ylim=c(-2,2), ylab = "", xlab =
"")
```




```
filled.contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-2,2), ylim=c(-2,2), xlab = "x
1", ylab = "x2", col = c('lightblue', 'darkred'), axes=F, plot.axes = { points(xin[,1],
 xin[,2],col=data$classes, xlim=c(-2,2), ylim=c(-2,2)); axis(1); axis(2) })
```

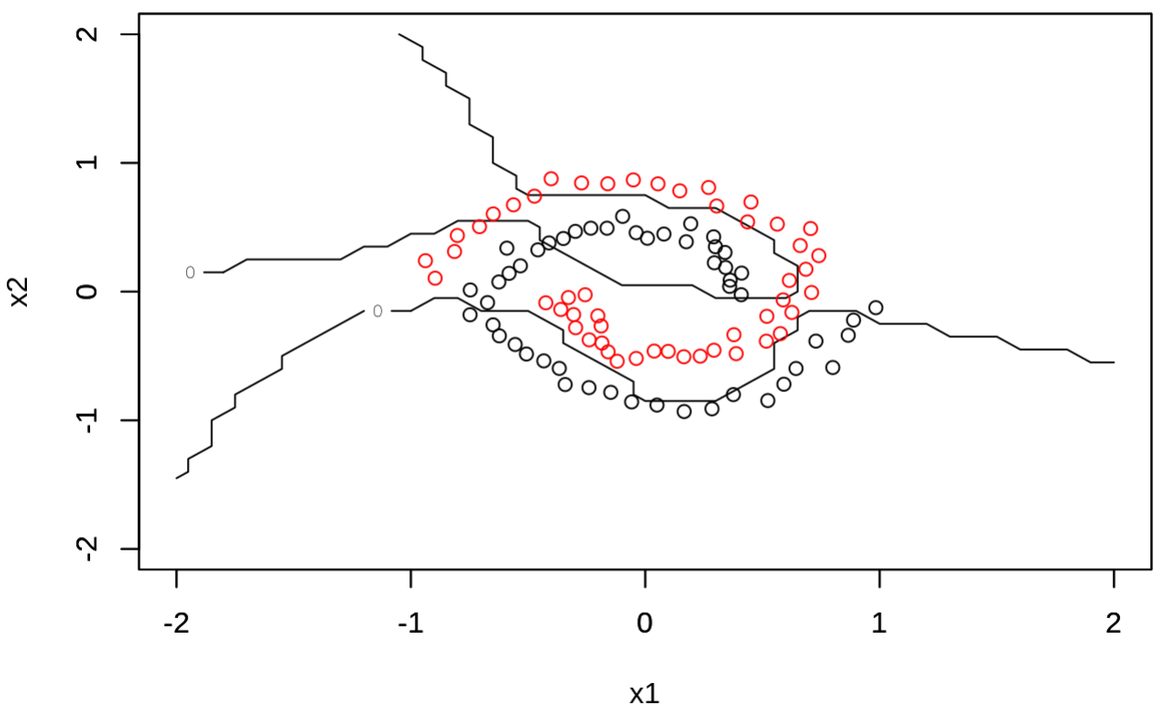


ELM com número de neurônios p = 10:

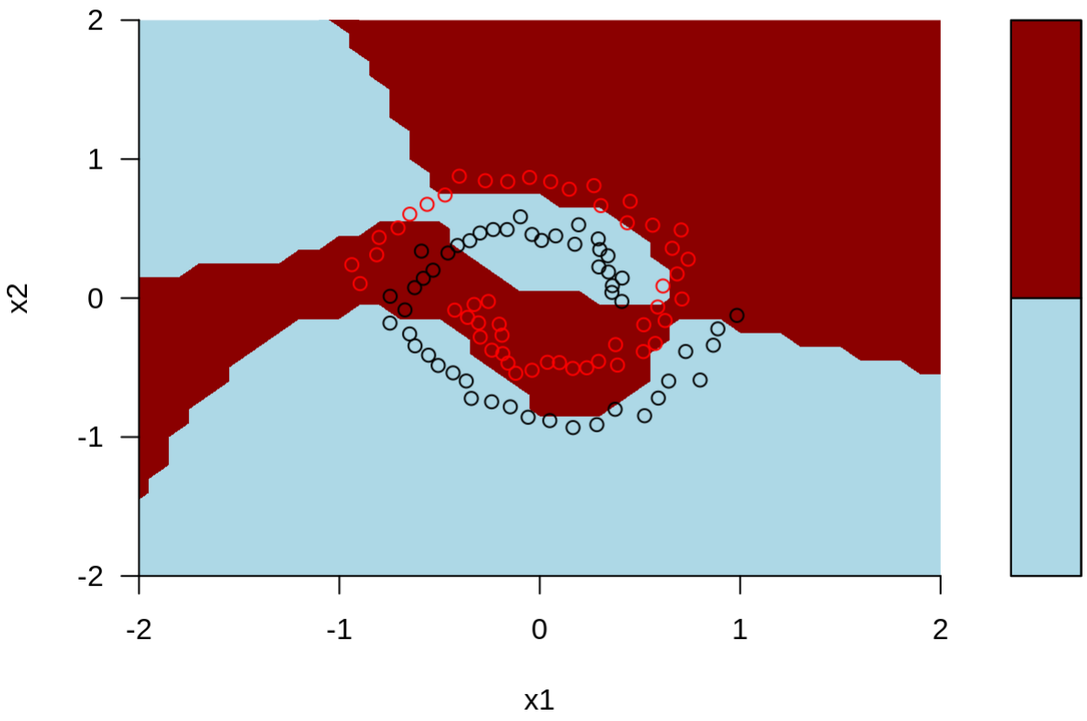
```
p<-10
retlist<-trainELM(xin,yin,p,1)

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

y_hat <- YELM(xin, Z, W, 1)
err_train <- sum((yin - y_hat)^2)/4 #divide-se por 4 pelo fato de a distância entre -1 e
 1 ser 2
paste("Erros:", err_train)
```

```
## [1] "Erros: 14"
```

```
# Plotando superfície
seqx1x2 <- seq(-2, 2, 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]<-YELM(x1x2, Z, W, 1)
  }
}

contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-2,2), ylim=c(-2,2), xlab = "x1", ylab
 = "x2")
par(new=T)
plot(xin[,1], xin[,2], col=data$classes, xlim=c(-2,2), ylim=c(-2,2), ylab = "", xlab =
"")
```




```
filled.contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-2,2), ylim=c(-2,2), xlab = "x
1", ylab = "x2", col = c('lightblue', 'darkred'), axes=F, plot.axes = { points(xin[,1],
 xin[,2],col=data$classes, xlim=c(-2,2), ylim=c(-2,2)); axis(1); axis(2) })
```

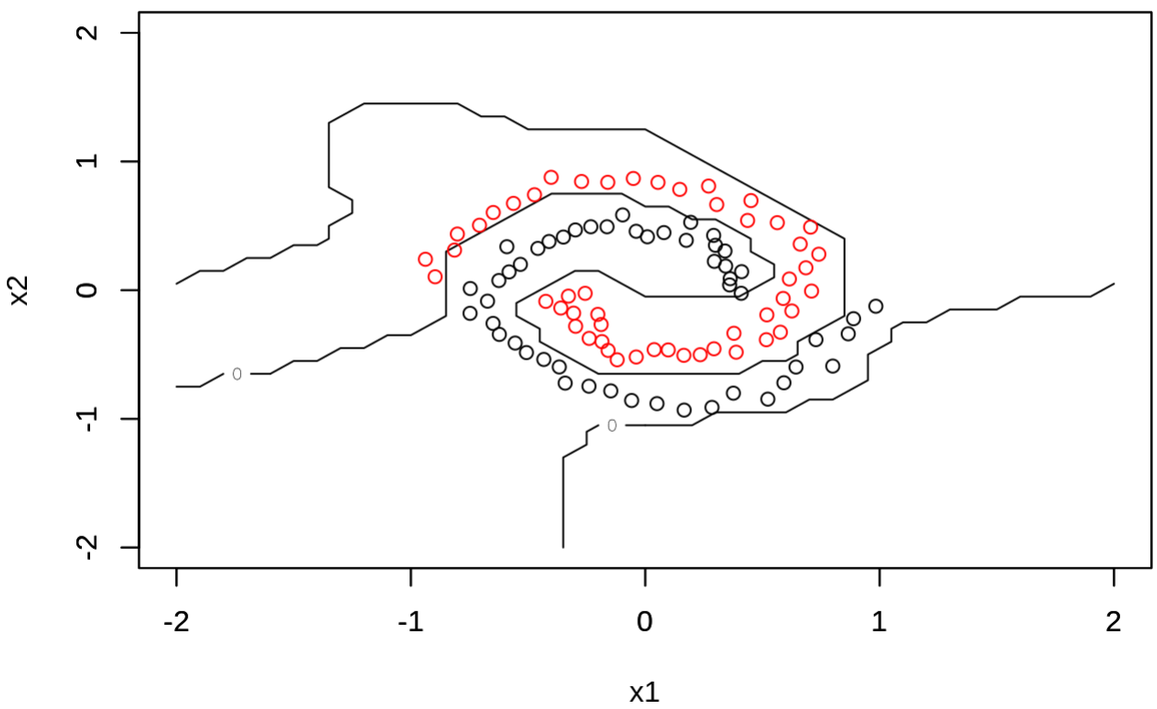


ELM com número de neurônios p = 30:

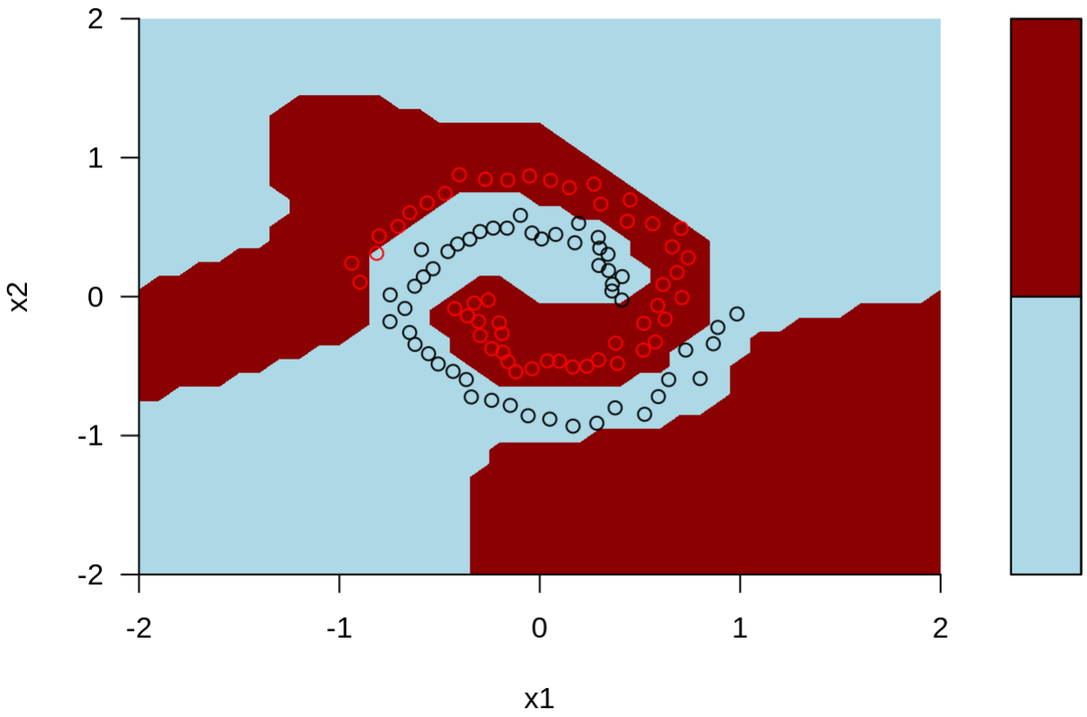
```
p<-30
retlist<-trainELM(xin,yin,p,1)

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

y_hat <- YELM(xin, Z, W, 1)
err_train <- sum((yin - y_hat)^2)/4 #divide-se por 4 pelo fato de a distância entre -1 e
 1 ser 2
paste("Erros:", err_train)
```

```
## [1] "Erros: 0"
```

```
# Plotando superfície
seqx1x2 <- seq(-2, 2, 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]<-YELM(x1x2, Z, W, 1)
  }
}

contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-2,2), ylim=c(-2,2), xlab = "x1", ylab
 = "x2")
par(new=T)
plot(xin[,1], xin[,2], col=data$classes, xlim=c(-2,2), ylim=c(-2,2), ylab = "", xlab =
"")
```




```
filled.contour(seqx1x2, seqx1x2, MZ, nlevels = 1, xlim=c(-2,2), ylim=c(-2,2), xlab = "x
1", ylab = "x2", col = c('lightblue', 'darkred'), axes=F, plot.axes = { points(xin[,1],
 xin[,2],col=data$classes, xlim=c(-2,2), ylim=c(-2,2)); axis(1); axis(2) })
```



Discussão:

Visualmente pode-se inferir que para esses dados necessita-se um classificador capaz de gerar uma superfície de separação que é bem mais complexa que as anteriores. No caso do classificador gerado a partir de uma ELM de 5 neurônios, pode-se notar que a superfície gerada não chegou nem perto de conseguir classificar os dados como esperado (pode-se notar um número elevado de erros de classificação nesse caso e uma superfície diferente da esperada). Para o classificador ELM com 10 neurônios, também não se teve uma solução satisfatória, apesar de já ser possível observar uma melhora significativa em comparação com o modelo com 5 neurônios. Por fim, a ELM com 30 neurônios foi capaz de gerar uma superfície de separação próxima ao esperado e um número de erros bem pequeno, considerada assim a solução mais próxima do ideal.