

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

## Exercício 5: Redes Neurais Artificiais

A partir do problema proposto nesta semana, foi realizada a divisão do problema em quatro partes referentes aos conjuntos de testes disponibilizados no enunciado do exercício. Para as implementações a seguir, foram implementados diferentes plots variando-se o número de neurônios por meio de um loop *for* assumindo os valores de 5, 1-, 15, 20, 25 e 30. A implementação da ELM pode ser vista logo abaixo:

```
<>>=
rm(list=ls())
library('RSNNS')
library('mlbench')

YELM <- function(xin, Z, W, par){
  n <- dim(xin)[2]
  if (par == 1)
    xin <- cbind(1, xin)
  H <- tanh(xin %*% Z)
  yhat <- sign(H %*% W)
  return(yhat)
}

treinaELM <-function(xin, yin, p, par) {
  n <- dim(xin)[2]
  if (par == 1) {
    xin <- cbind(1, xin)
    Z <- replicate(p, runif((n+1), -0.5, 0.5))
  }
  else
    Z <- replicate(p, runif((n+1), -0.5, 0.5))
  H <- tanh(xin %*% Z)
  W <- (solve(t(H) %*% H) %*% t(H)) %*% yin
  return(list(W, H, Z))
}
```

### 1. mlbench.2dnormals(200)

Para este primeiro conjunto de dados, foi realizado a seguinte implementação, que pode ser vista logo abaixo:

```
<>>=
# Caso 1: mlbench.2dnormals

normals = mlbench.2dnormals(200)
normals = cbind(normals[[1]], normals[[2]])
normals[,3] = (normals[,3] - 1.5) * 2
normalsall <- splitForTrainingAndTest(normals[,1:2], normals[,3], ratio = 0.3)
```

```

xin_normals <- normalsall$inputsTrain
yd_normals <- normalsall$targetsTrain
xin_test_normals <- normalsall$inputsTest
yteste <- normalsall$targetsTest

pvet <- seq(5, 30, 5)

for(pfor in pvet) {
  retlist <- treinaELM(xin_normals, yd_normals, pfor, 1)
  W_normals <- retlist[[1]]
  H_normals <- retlist[[2]]
  Z_normals <- retlist[[3]]
  yt_normals <- YELM(xin_test_normals, Z_normals, W_normals, 1)

  plot(normals, xlim = c(-3, 3), ylim = c(-4, 4))

  seqi <- seq(-4, 4, 0.2)
  seqj <- seq(-3, 3, 0.15)
  M <- matrix(0, nrow = length(seqi), ncol = length(seqj))
  ci <- 0

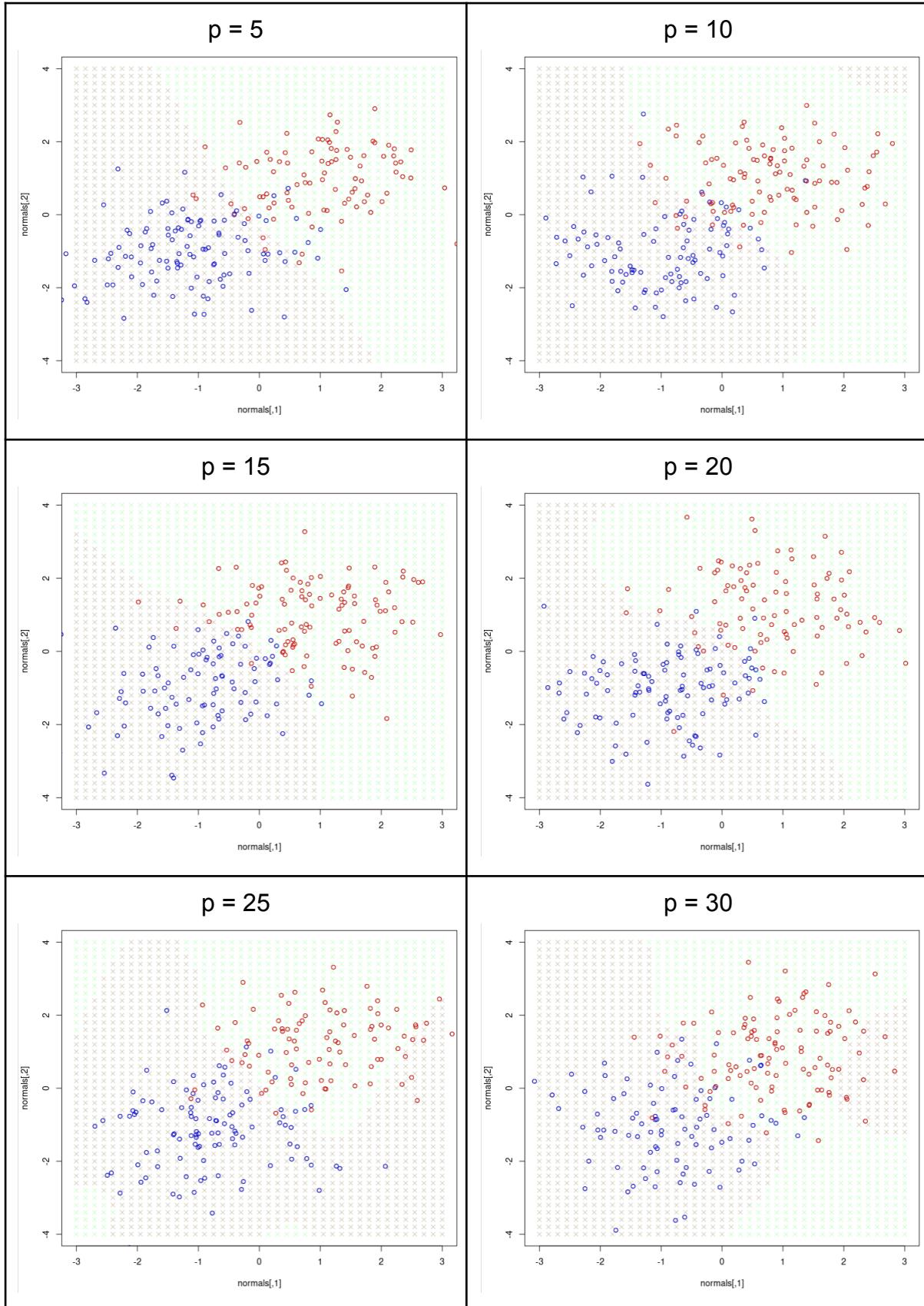
  for(i in seqi) {
    ci <- ci + 1
    cj <- 0
    for(j in seqj) {
      cj <- cj + 1
      X <- as.matrix(t(c(i,j)))
      M[ci,cj] <- YELM(X, Z_normals, W_normals, 1)
    }
  }

  for(i in 1:length(seqi)) {
    for(j in 1:length(seqj)) {
      if(M[i,j] == 1)
        points(seqj[i], seqi[j], pch = 4, col = "cornsilk3")
      else
        points(seqj[i], seqi[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")
  }
}

```

A seguir, são apresentadas as superfícies de separação para os diferentes valores de  $p$  mencionados mais acima:



Analizando-se as figuras acima, percebe-se que no momento que se aumenta o número de neurônios, é observado um maior detalhismo na separação das superfícies.

No primeiro caso, as separações com a utilização de um número relativamente pequeno de neurônios já apresentou resultados interessantes. Contudo, ao mesmo tempo que observamos um maior detalhismo das superfícies, foi observado também uma tendência ao overfitting nas respostas, e consequentemente resultados indesejados nas respostas dos modelos.

## 2. mlbench.xor(100)

Para este segundo exemplo, foi realizado a seguinte implementação:

```
<>>=
# Caso 2: mlbench.xor

xor = mlbench.xor(100)
xor = cbind(xor[[1]], xor[[2]])
xor[,3] = (xor[,3] -1.5) * 2

xorall <- splitForTrainingAndTest(xor[,1:2], xor[,3], ratio = 0.1)
xin_xor <- xorall$inputsTrain
yd_xor <- xorall$targetsTrain
xin_test_xor <- xorall$inputsTest
ytest <- xorall$targetsTest

pvet <- seq(5, 30, 5)

for(pfor in pvet){
  retlist <- treinaELM(xin_xor, yd_xor, pfor, 1)
  W_xor <- retlist[[1]]
  H_xor <- retlist[[2]]
  Z_xor <- retlist[[3]]
  yt_xor <- YELM(xin_test_xor, Z_xor, W_xor, 1)

  plot(xor, xlim = c(-1, 1), ylim = c(-1,1))

  seqi <- seq(-1, 1, 0.05)
  seqj <- seq(-1, 1, 0.05)
  M <- matrix(0, nrow = length(seqi), ncol = length(seqj))
  ci <- 0
```

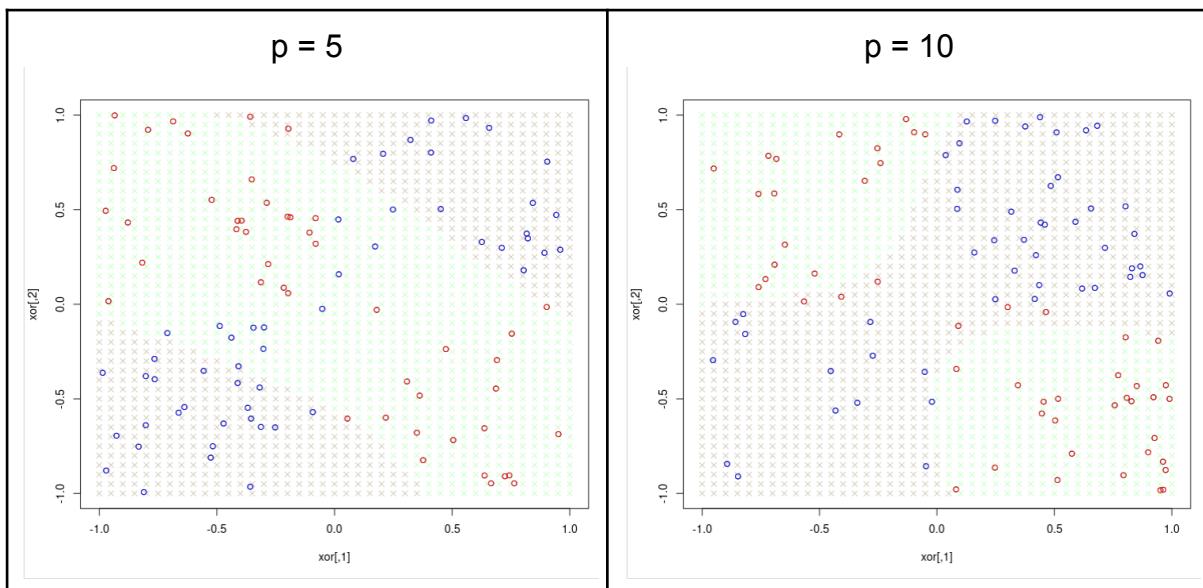
```

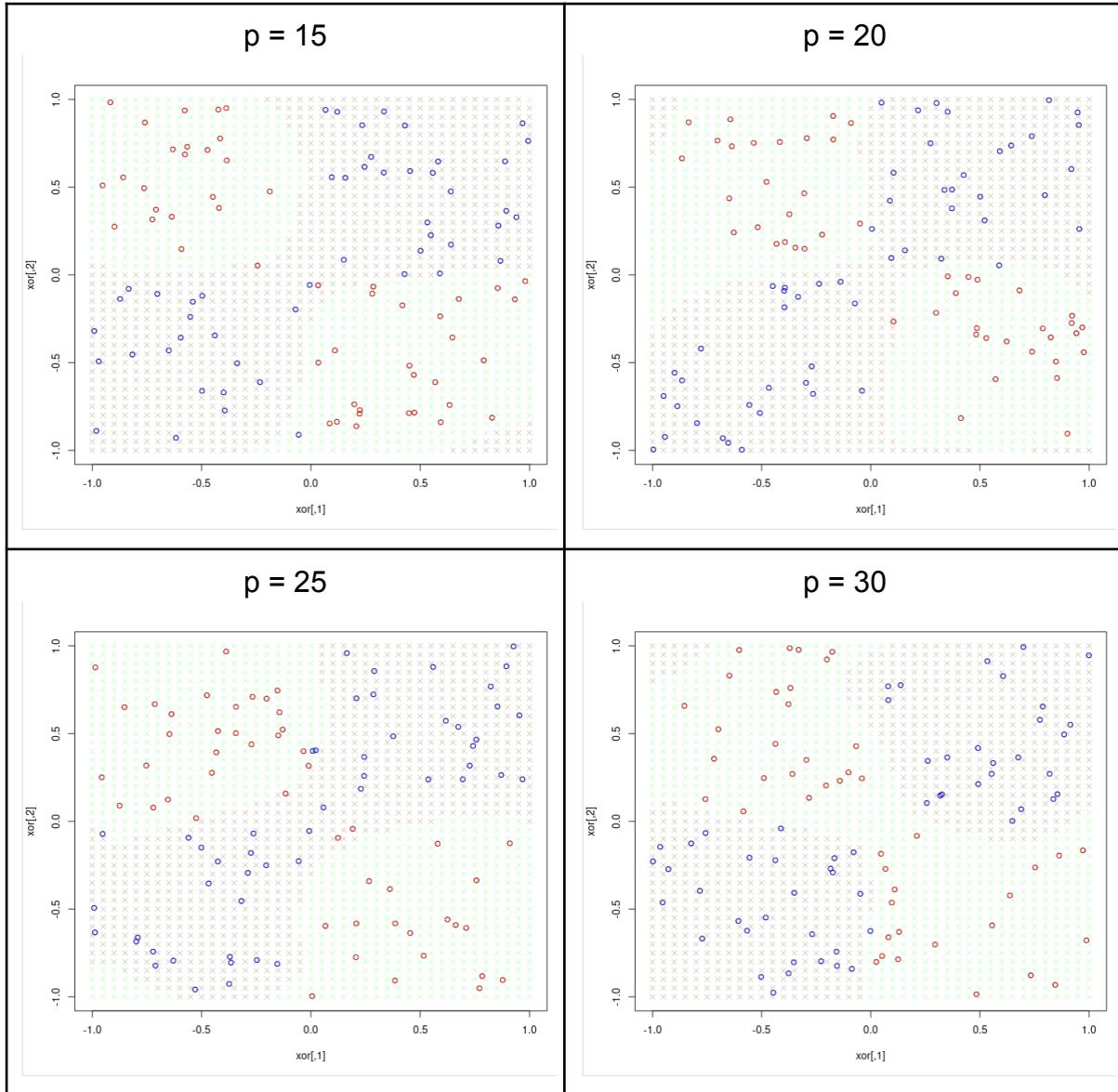
for(i in seqi){
  ci <- ci + 1
  cj <- 0
  for(j in seqj){
    cj <- cj + 1
    X <- as.matrix(t(c(i, j)))
    M[ci, cj] <- YELM(X, Z_xor, W_xor, 1)
  }
}

for(i in 1:length(seqi)){
  for(j in 1:length(seqj)){
    if(M[i,j] == 1)
      points(seqi[i], seqj[j], pch = 4, col = "cornsilk3")
    else
      points(seqi[i], seqj[j], pch = 4, col = "darkseagreen1")
  }
}

for(i in 1:100){
  if(xor[i,3] == 1)
    points(xor[i,1], xor[i,2], col = "blue")
  if(xor[i,3] == -1)
    points(xor[i,1], xor[i,2], col = "red")
}
@
```

A seguir, são apresentadas as superfícies de separação para os diferentes valores de  $p$  mencionados mais acima:





Diferentemente do primeiro exemplo, aumentar o número de neurônios no modelo trouxe um impacto muito positivo para a resposta, uma vez que tratava-se de um modelo de maior complexidade.

Com isso, uma implementação com um menor número de neurônios apresentou uma dificuldade maior na separação de superfícies, sobretudo na primeira e segunda figuras.

### 3. mlbench.circle(100)

Para este terceiro exemplo, foi realizado a seguinte implementação:

```

<>>=
# Caso 3: mlbench.circle

circle = mlbench.circle(100)
circle = cbind(circle[,1], circle[,2])
circle[,3] = (circle[,3] -1.5) * 2

circleall <- splitForTrainingAndTest(circle[,1:2], circle[,3], ratio = 0.3)
xin_circle <- circleall$inputsTrain
yd_circle <- circleall$targetsTrain
xin_test_circle <- circleall$inputsTest
yteste <- circleall$targetsTest

pvet <- seq(5, 30, 5)

for(pfor in pvet){
  retlist <- treinaELM(xin_circle, yd_circle, pfor, 1)
  W_circle <- retlist[[1]]
  H_circle <- retlist[[2]]
  Z_circle <- retlist[[3]]
  yt_circle <- YELM(xin_test_circle, Z_circle, W_circle, 1)

  plot(circle, xlim = c(-1,1), ylim = c(-1,1))

  seqi <- seq(-1, 1, 0.05)
  seqj <- seq(-1, 1, 0.05)
  M <- matrix(0, nrow = length(seqi), ncol = length(seqj))
  ci <- 0

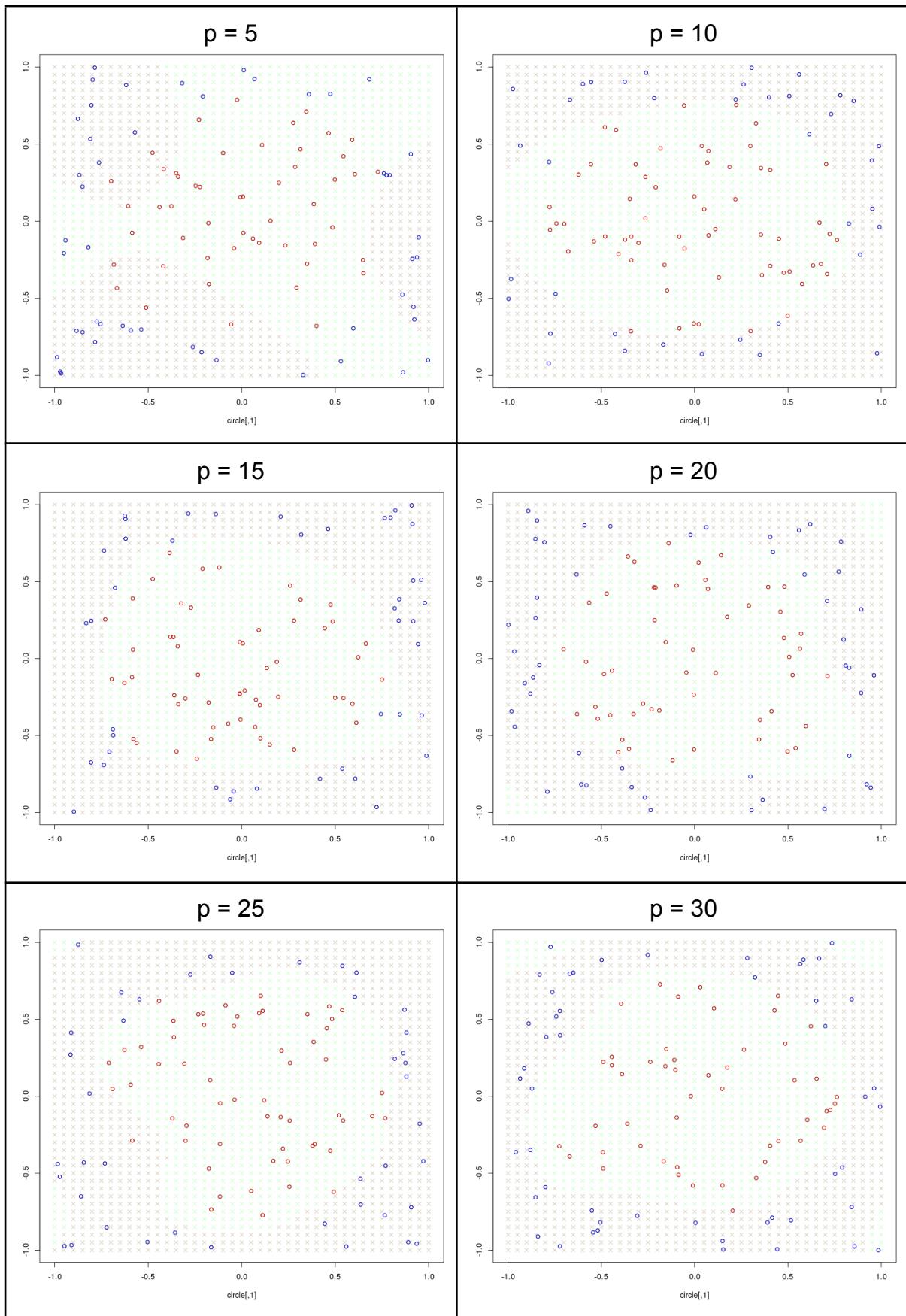
  for(i in seqi){
    ci <- ci + 1
    cj <- 0
    for(j in seqj){
      cj <- cj + 1
      X <- as.matrix(t(c(i,j)))
      M[ci,cj] <- YELM(X, Z_circle, W_circle, 1)
    }
  }

  for(i in 1:length(seqi)){
    for(j in 1:length(seqj)){
      if(M[i,j] == 1)
        points(seqj[i], seqi[j], pch = 4, col = "cornsilk3")
      else
        points(seqj[i], seqi[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")
  }
}

```

A seguir, são apresentadas as superfícies de separação para os diferentes valores de  $p$  mencionados mais acima:



Assim como no segundo exemplo, observou-se uma melhora gradual ao aumentar o número de neurônios do modelo até o valor de  $p = 25$ . Na resposta de  $p = 30$ , observa-se um início de tendência ao overfitting.

A diferença mais nítida dentre as respostas obtidas entre  $p = 5$  e  $p = 10$  (as duas primeiras imagens). No primeiro caso, ficou nítida a dificuldade do modelo para a resolução do problema, enquanto no intervalo entre  $p = 10$  e  $p = 25$ , o modelo se comportou de maneira satisfatória.

#### 4. mlbench.spirals(100, sd = 0.05)

Para este último caso, foi realizado a seguinte implementação:

```
<>>=
# Caso 4: mlbench.spiral

espiral = as.matrix(mlbench.spirals(100, 1, sd = 0.05))
espiral = cbind(espiral[,1], espiral[,2])
espiral[,3] = (espiral[,3]-1.5) * 2

espiralall <- splitForTrainingAndTest(espiral[,1:2], espiral[,3], ratio = 0.1)
xin_espiral <- espiralall$inputsTrain
yd_espiral <- espiralall$targetsTrain
xin_test_espiral <- espiralall$inputsTest
yteste <- espiralall$targetsTest

pvet <- seq(5, 30, 5)

for(pfor in pvet) {
  retlist <- treinaELM(xin_espiral, yd_espiral, pfor, 1)
  W_espiral <- retlist[[1]]
  H_espiral <- retlist[[2]]
  Z_espiral <- retlist[[3]]
  yt_espiral <- YELM(xin_test_espiral, Z_espiral, W_espiral, 1)

  plot(espiral, xlim = c(-1,1), ylim = c(-1,1))

  seqi <- seq(-1, 1, 0.05)
  seqj <- seq(-1, 1, 0.05)
  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(t(c(i,j)))
      M[ci,cj] <- YELM(X, Z_espiral, W_espiral, 1)
    }
  }
}
```

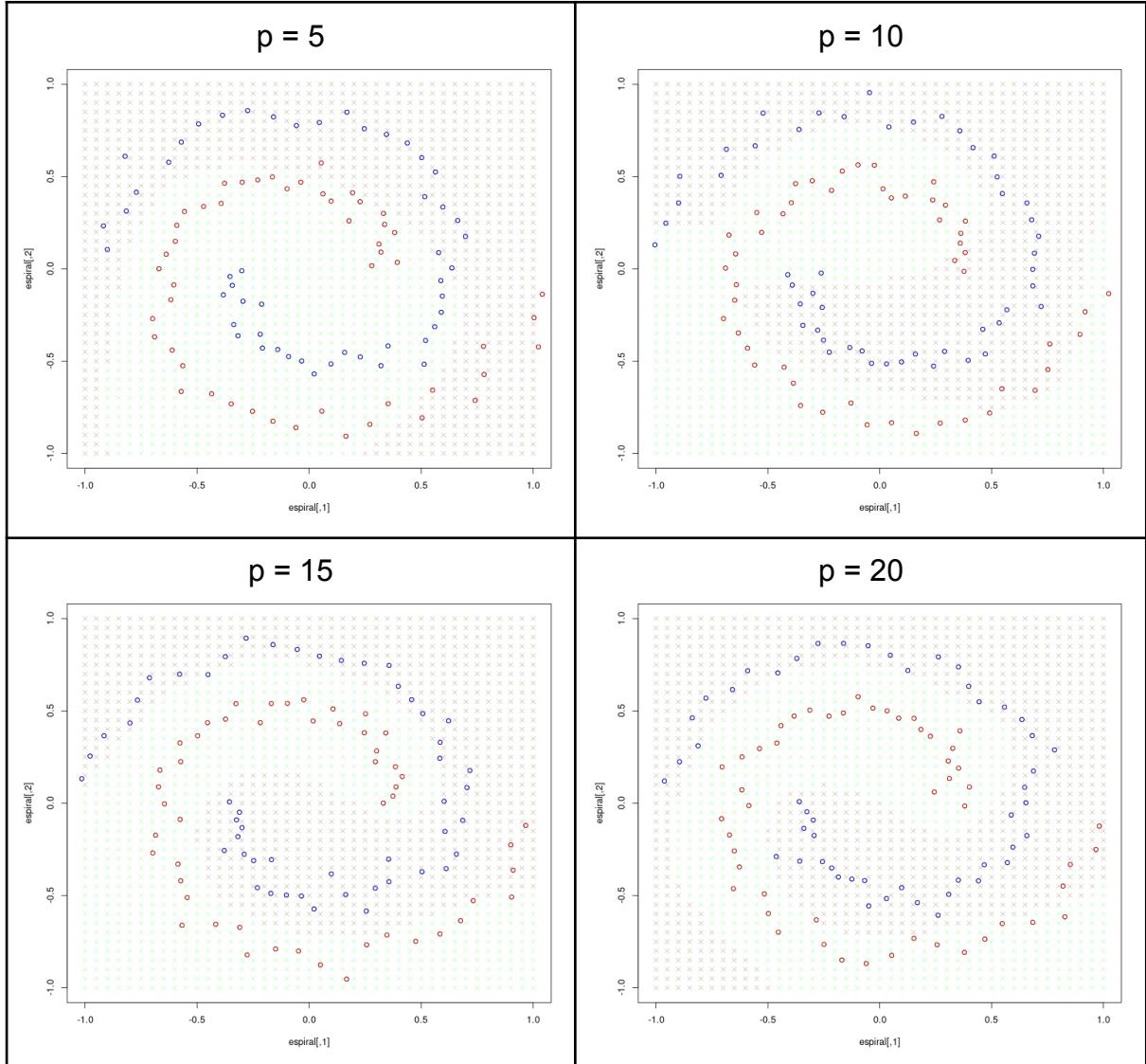
```

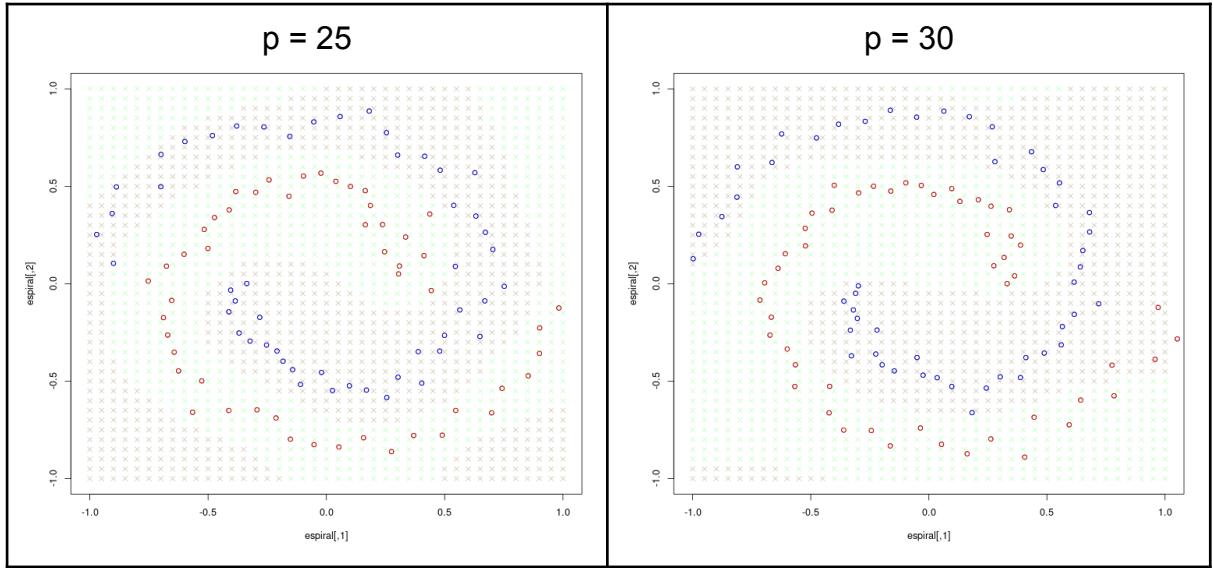
for(i in 1:length(seqi)){
  for(j in 1:length(seqj)){
    if(M[i,j] == 1)
      points(seqj[i], seqi[j], pch = 4, col = "cornsilk3")
    else
      points(seqj[i], seqi[j], pch = 4, col = "darkseagreen1")
  }
}

for(i in 1:100{
  if(espiral[i,3] == 1)
    points(espiral[i,1], espiral[i,2], col = "blue")
  if(espiral[i,3] == -1)
    points(espiral[i,1], espiral[i,2], col = "red")
}
}

```

A seguir, são apresentadas as superfícies de separação para os diferentes valores de  $p$  mencionados mais acima:





Neste último caso, tratando-se de um outro problema complexo a ser solucionado, foi necessário a utilização de um número maior de neurônios para realizar uma separação adequada.

A partir do segundo caso,  $p = 10$ , nota-se o início de um resultado aceitável para o modelo. Contudo, quanto mais neurônios eram adicionados melhor a resposta do modelo era encontrada, como podemos ver no último caso, com  $p = 30$ .