Universidade Federal de Minas Gerais

Exercício ELM – Unidade 2

Redes Neurais Artificiais

Daniel Nogueira Junqueira – 2021072244

danijnog@ufmg.br

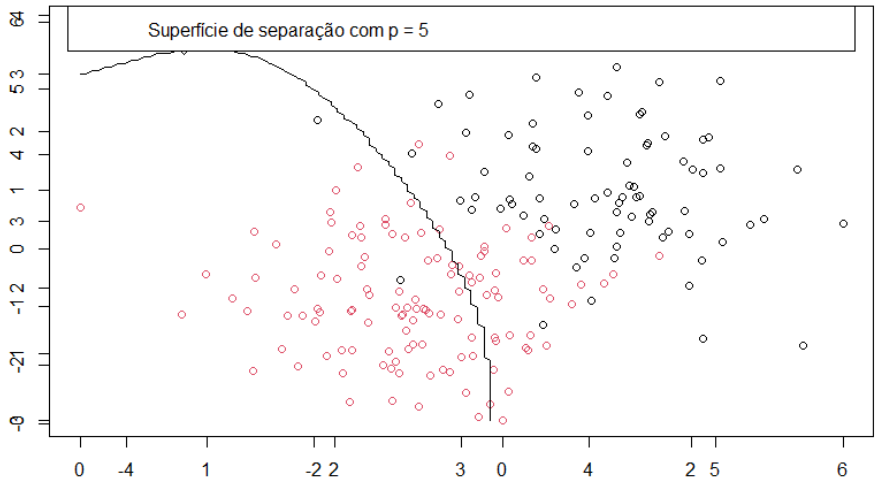
Esse documento tem como finalidade mostrar a utilização de ELMs como classificadores não lineares de um conjunto de dados.

Abaixo consta as bases de dados geradas de acordo com o enunciado do trabalho e o gráfico das ELMs contendo a separação linear das amostras.

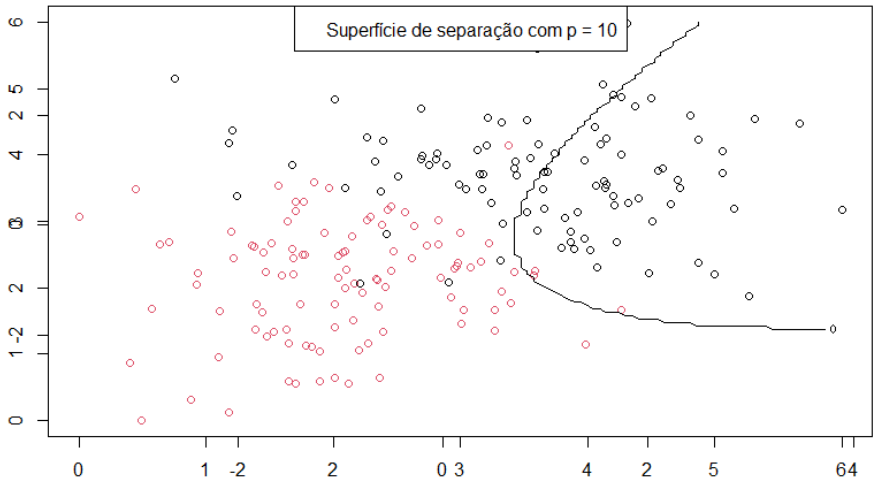
Em cada base de dados, é utilizado a seguinte quantidade de neurônios para efeitos de comparação: 5, 10, e depois 30.

1. Base de dados: *mlbench.2dnormals(200)*

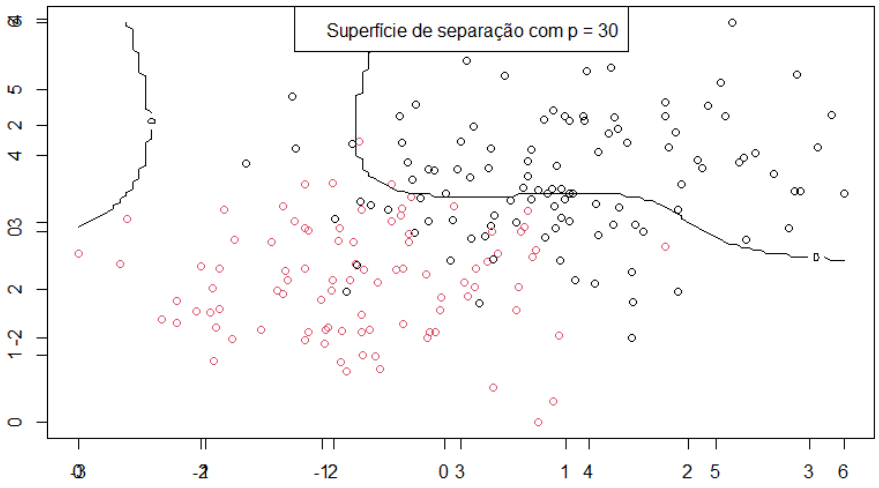
Utilizando primeiramente 5 neurônios, obtemos a seguinte separação:



10 neurônios



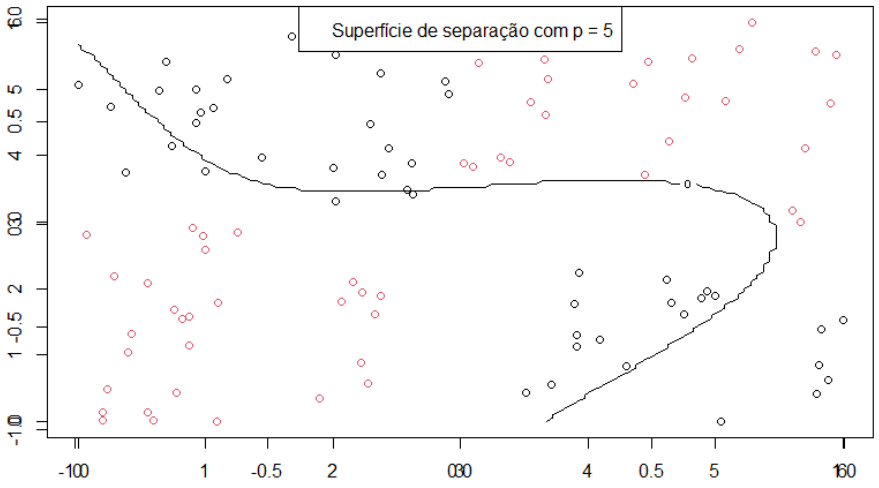
30 neurônios



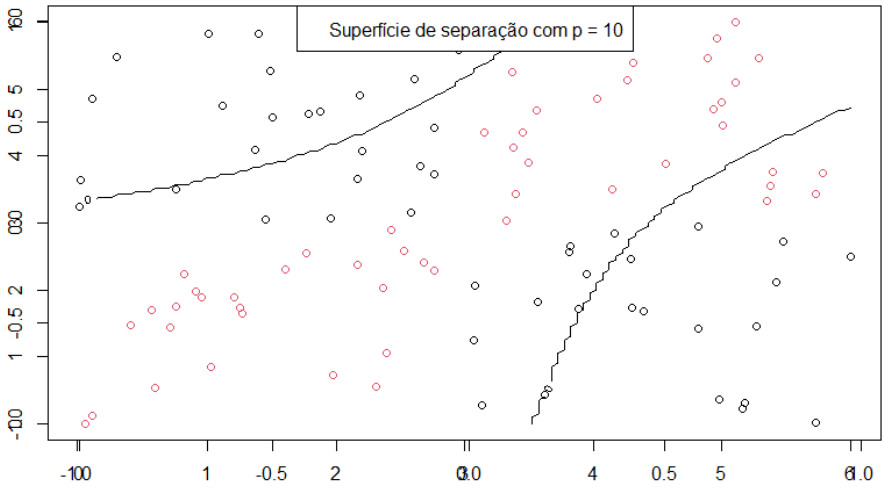
Conclusão: A partir dos 3 gráficos, chegamos a conclusão de que a quantidade de neurônios que melhor projeta uma superfície de separação é na faixa de 5. Quando aumentamos muito a quantidade de neurônios, é comum ocorrer *overfitting* do modelo.

2. Base de dados: *mlbench.xor(100)*

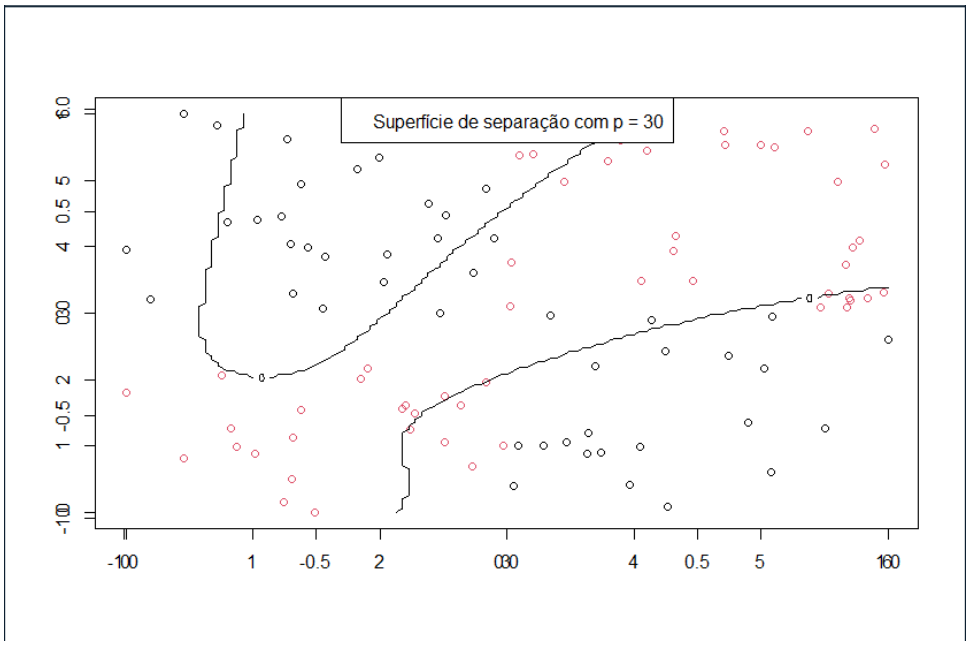
5 neurônios:



10 neurônios:



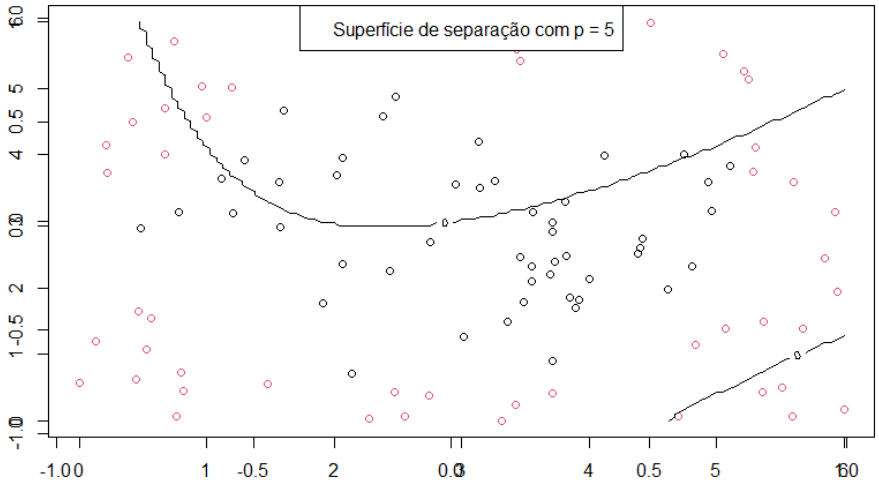
30 neurônios



Conclusão: Nesse caso, com uma base de dados xor, ao aumentarmos a quantidade de neurônios é perceptível que o modelo se adapta melhor, apesar de ainda encontrar dificuldades para projetar a superfície de separação.

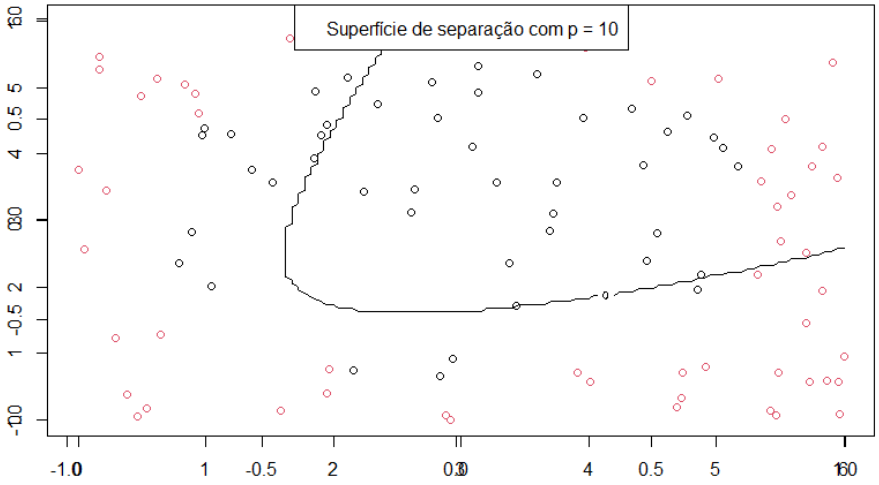
3. Base de dados: *mlbench.circle(100)*

5 neurônios

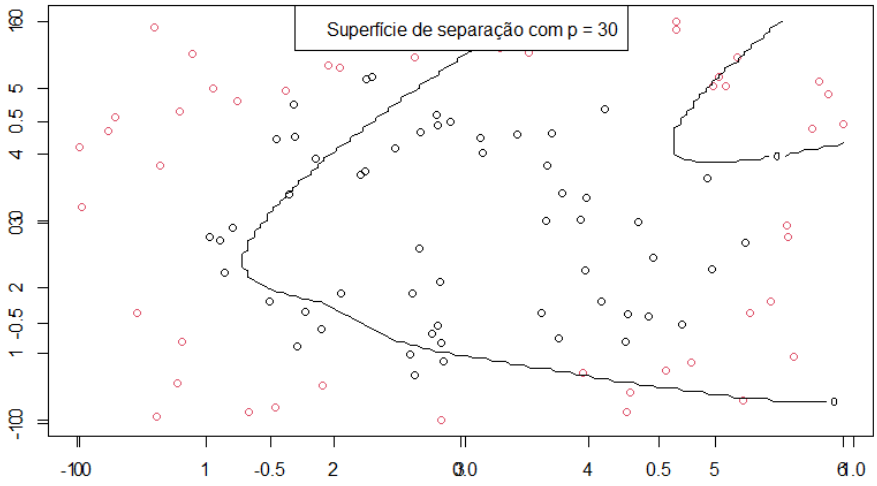


---

10 neurônios



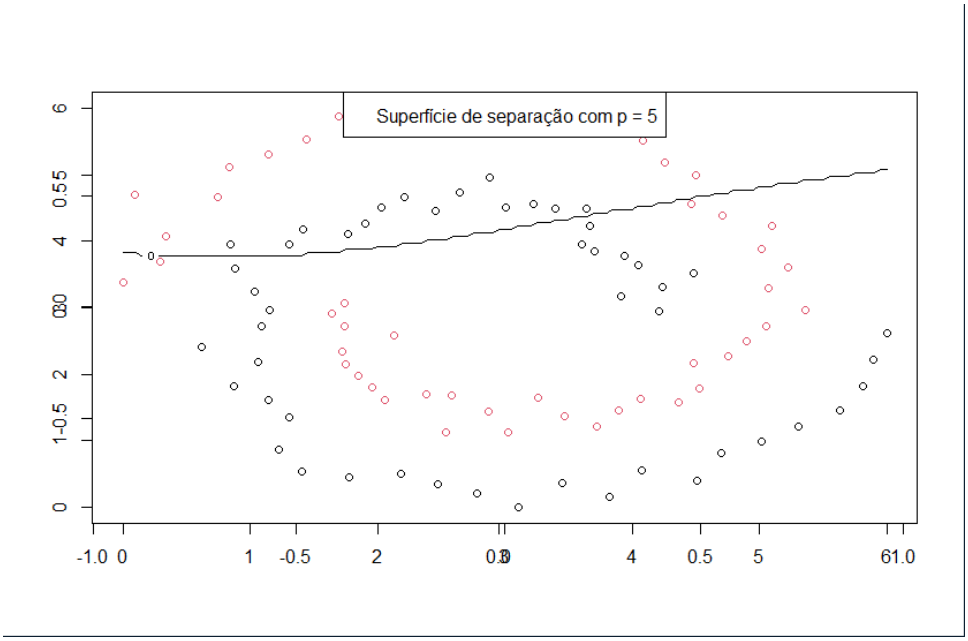
30 neurônios



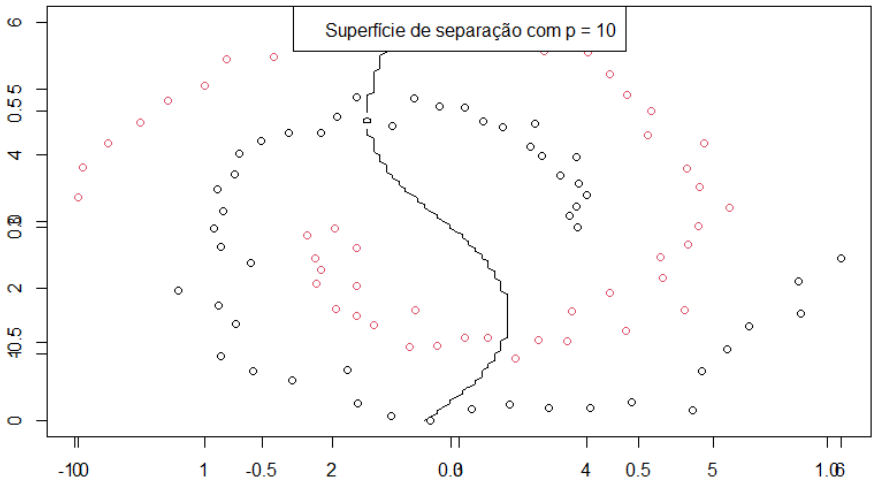
Conclusão: O modelo, mesmo com um aumento na quantidade de neurônios, apesar de haver uma melhoria na projeção da superfície de contorno ainda não ficou muito bem projetada, e se aumentarmos ainda mais a quantidade de neurônios pode acabar ocorrendo *overfitting* do modelo.

4. Base de dados: *mlbench.spirals(100,sd = 0.05)*

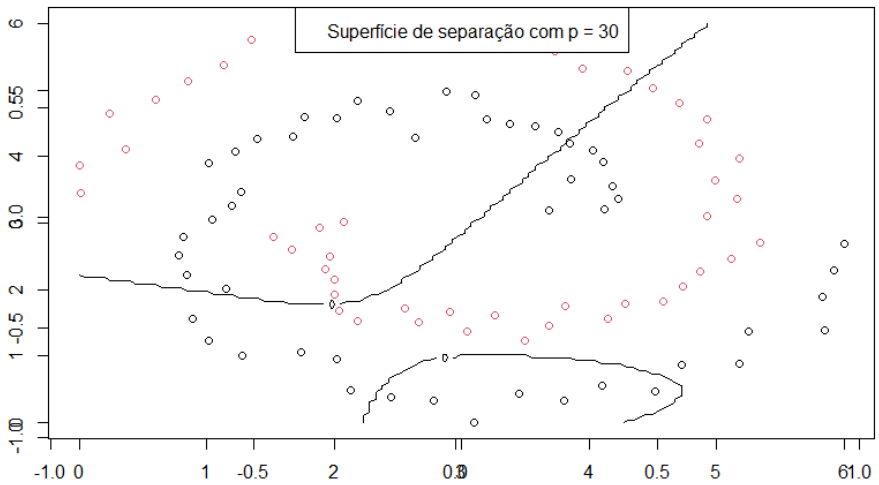
5 neurônios



10 neurônios



30 neurônios



Conclusão: Ao aumentarmos a quantidade de neurônios é perceptível uma melhoria na previsão do modelo, apesar de também não ser perfeito para realizar a projeção da superfície de separação. Podemos concluir também que ao aumentarmos demais a quantidade de neurônios acaba ocorrendo um overfitting do modelo.

5. Código utilizado

```r
# Carregue a biblioteca mlbench
library(mlbench)

# Gere o conjunto de dados
data <- mlbench.spirals(100,sd = 0.05)

# Extraia as amostras de entrada (xall) e os rótulos de saída (yall)
xall <- data$x
yall <- as.numeric(data$classes)

# Visualize o conjunto de dados
plot(data)

# Crie a matriz de pesos aleatórios Z para a camada oculta
p <- 60
Z <- replicate(p, runif(3, -0.5, 0.5))

# Normalize os dados de entrada (xall)
xall <- scale(xall)

# Adicione um termo de polarização (bias) à matriz xall normalizada
Xaug <- cbind(replicate(dim(xall)[1], 1), xall)

# Calcule a saída da camada oculta aplicando a função tangente hiperbólica
H <- as.matrix(tanh(Xaug %*% Z))

# Calcule os pesos da camada de saída da ELM
W <- pseudoinverse(H) %*% yall

# Visualize o separador linear
seqi <- seq(0, 6, 0.05)
seqj <- seq(0, 6, 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
    xg <- c(1, i, j)
    Hg <- as.matrix(tanh(xg %*% Z))
    M[ci, cj] <- sign(Hg %*% W)
  }

  plot(data)
  par(new = T)
  # Desenhe o separador linear
  contour(seqi, seqj, M, xlim = c(0, 6), ylim = c(0, 6), nlevels = 0)

  legend('top', legend = ('Superfície de separação com p = 30'))
}
```

OBS: para cada base de dados, foi alterado a linha “5” do modelo, gerando uma nova base de dados e projetando a ELM novamente.