UNIVERSIDADE FEDERAL
DE MINAS GERAIS

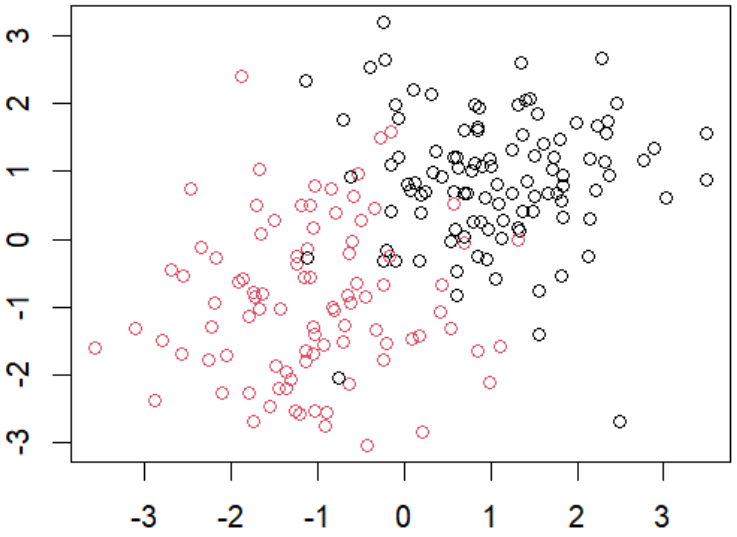
ATIVIDADE AVALIATIVA 3
TREINAMENTO ADALINE E PERCEPTRON

ALUNO: FELIPPE VELOSO MARINHO
MATRÍCULA: 2021072260
DISCIPLINA: REDES NEURAIS ARTIFICIAIS

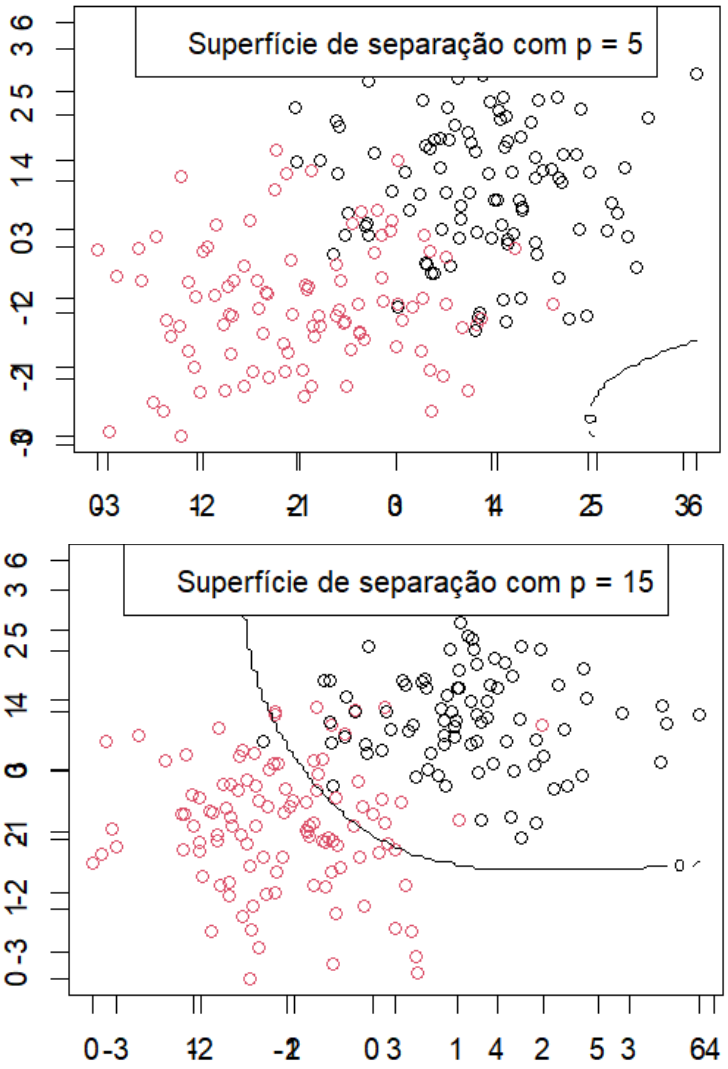
As bases de dados deve ser gerada utilizando:
1 mlbench.2dnormals(200)
2 mlbench.xor(100)
3 mlbench.circle(100)
4 mlbench.spirals(100,sd = 0.05)

Para cada uma dessas devem ser construídas diferentes ELMs (3 para cada)
Para cada uma deve-se variar para p=5, p=10 e p=30.
Refletir qual qtd de neurônios levou melhores resultados para cada,

- 1 mlbench.2dnormals(200)
- A função essencialmente cria dois grupos de pontos distribuídos aleatoriamente no espaço bidimensional, seguindo uma distribuição normal.

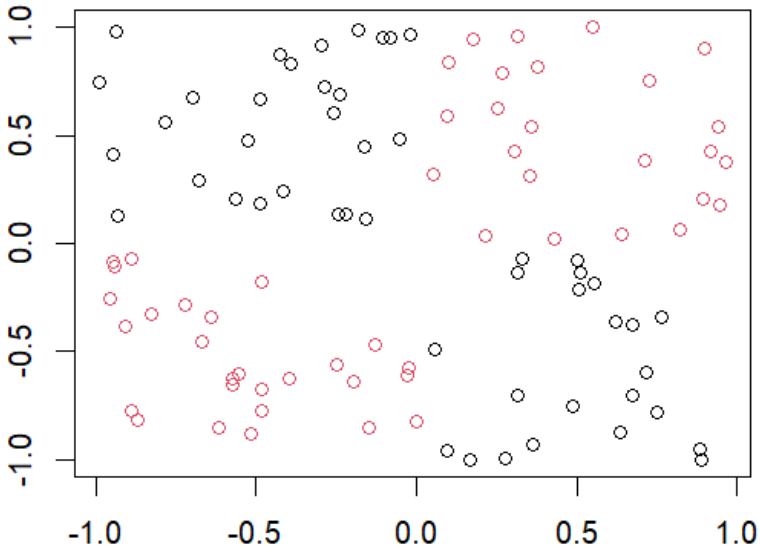


Utilizando este conjunto de dados para p = 5 temos uma superfície de separação com resultado bem abaixo do esperado. Os melhores resultados que obtive ficaram entre 15 e 20 neurônios porém com uma acurácia relativamente baixa para a quantidade de pontos utilizados.

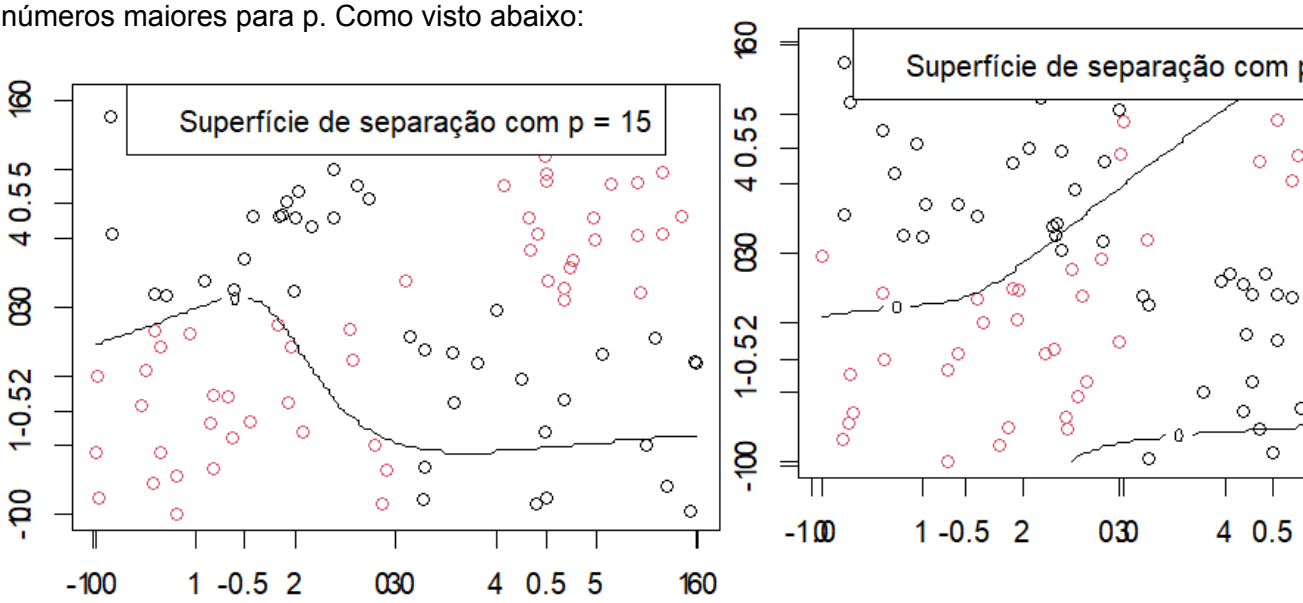


- 2 mlbench.xor(100)

- O conjunto de dados XOR é bem conhecido no contexto de problemas de classificação binária. Ele gera amostras em um plano onde as classes formam um padrão de exclusão mútua (XOR). Ou seja, são duas classes que não podem ser separadas linearmente.

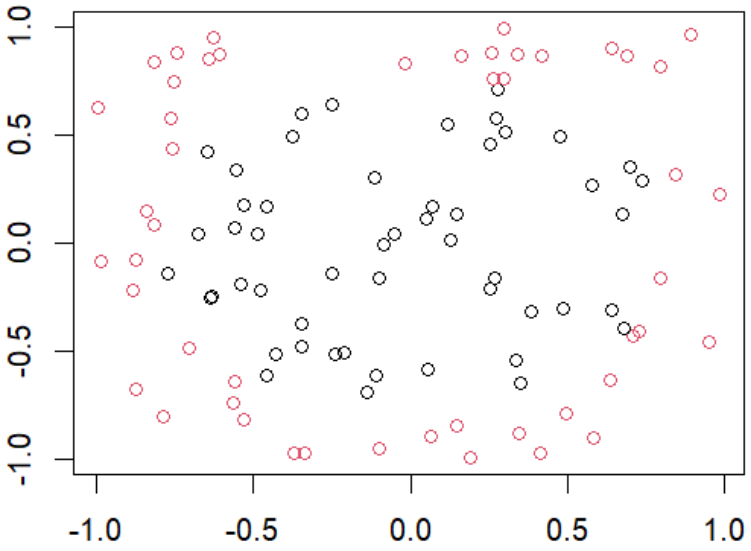


Repetindo o teste para essa base de dados, obtivemos os melhores resultados para números maiores para p. Como visto abaixo:

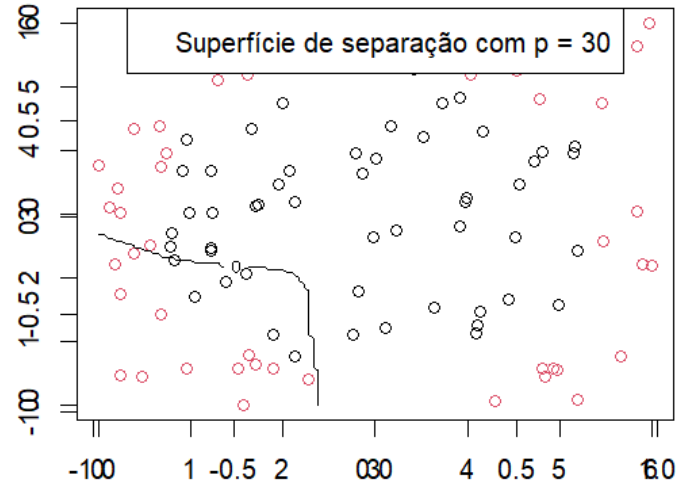


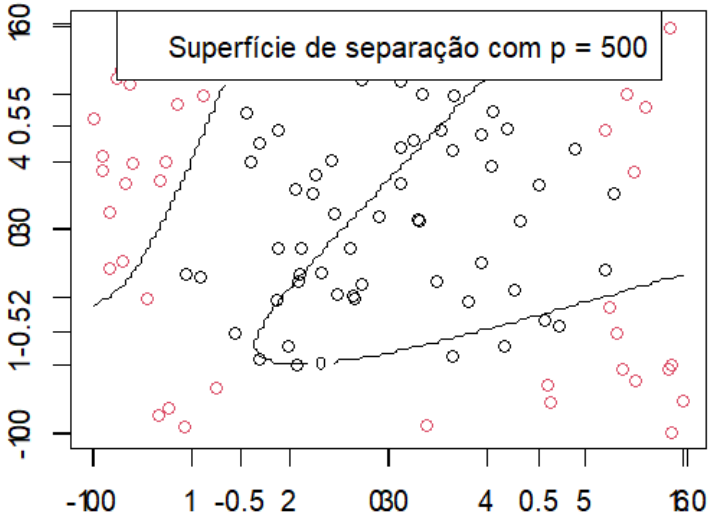
- 3 mlbench.circle(100)

- Este conjunto de dados gera duas classes distribuídas em torno de um círculo. Cada classe é definida por pontos que se encontram em torno de um círculo com centro e raio específicos.

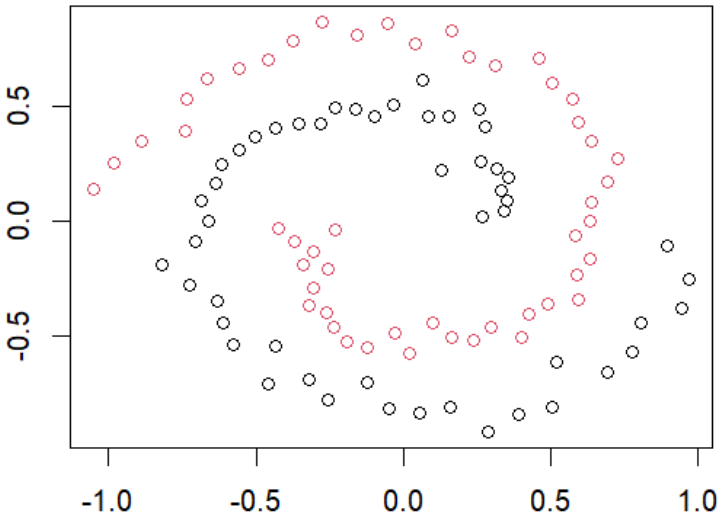


Para essa base de dados obtivemos um resultado curioso. Em todos os valores de p, não obtive muitas variações na delimitação da superfície de separação. Aparentemente para esse tipo de problema o modelo desenvolvido não se comportou bem, tendo uma acurácia bem baixa.

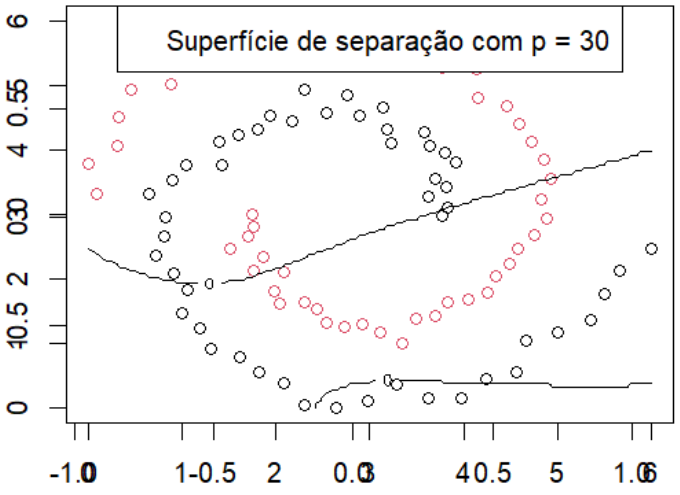


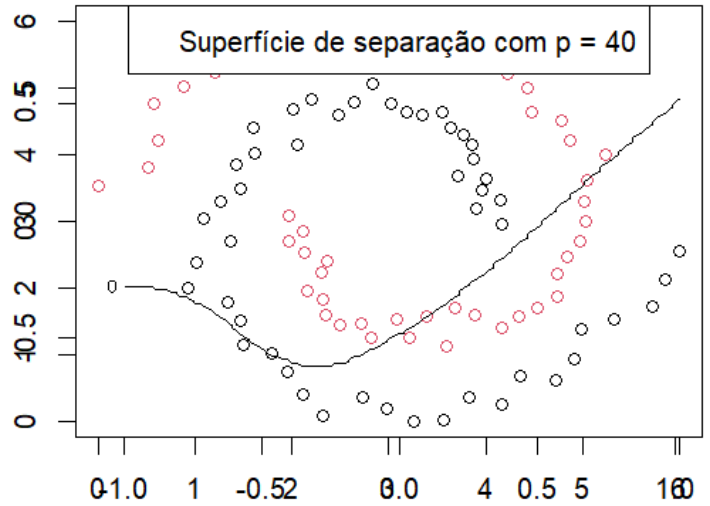


- 4 mlbench.spirals(100,sd = 0.05)
- Aqui, o conjunto de dados cria duas classes que seguem a forma de espirais. Cada classe representa uma espiral em duas dimensões. O parâmetro sd controla a dispersão dos pontos ao longo das espirais.



Para esse conjunto de dados tivemos um resultado semelhante ao conjunto anterior, com o modelo não se comportando muito bem. Possivelmente o modelo poderia ser adaptado para melhor funcionamento nos diferentes problemas, porém para análise proposta, é o suficiente. A superfície de separação para valores de p abaixo 30 se mostrou underfitting porém mesmo com mais números o modelo não se comporta bem.





Código:

```
# importando o pacote mlbench
library(mlbench)
library(MASS)

# Defina os conjuntos de dados que serão utilizados
datasets <- list(
```

```
  mlbench.2dnormals(200),
  mlbench.xor(100),
  mlbench.circle(100),
  mlbench.spirals(100, sd = 0.05)
)

# Defina o número de neurônios na camada oculta (p)
p_values <- c(5, 10, 30)

# Adapte a partir daqui

# Gere o conjunto de dados
#data <- mlbench.2dnormals(200)
#data <- mlbench.xor(100)
#data <- mlbench.circle(100)
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 <- 40
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 = paste('Superfície de separação com p =', p))
}
```