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

# Exercício 6: Redes Neurais Artificiais

A partir do problema proposto no exercício, foi possível dividir o problema em duas partes, referentes às seguintes amostras de dados:

- Breast Cancer;
- Statlog(Heart).

## 1. Breast Cancer

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

```
<<>>=
# Caso 1: Breast Cancer

neuronios <- c(5, 10, 30, 50, 100, 110, 180, 200, 300, 400, 600)
k <- 0
acuracia1 <- matrix(0, nrow = 11, ncol = 10)
acuracia2 <- matrix(0, nrow = 11, ncol = 10)

for(i in neuronios) {
  k <- k + 1
  for(j in 1:10) {
    bcdata <- read.csv("~/Projetos/Trabalhos-Faculdade/Redes\ Neurais\ Artificiais/Exercicios/06/wpbc.csv")
    bcdata$id <- NULL
    bcdata$X <- NULL
    bcdata$class[bcdata$diagnosis == "M"] <- (-1)
    bcdata$class[bcdata$diagnosis == "B"] <- 1
    bcdata$diagnosis <- NULL

    trainpercentage <- 70

    bcdataM <- as.matrix(bcdata[bcdata$class == (-1),])
    bcdataB <- as.matrix(bcdata[bcdata$class == 1,])
    nrowsM  <- length(bcdataM[,2])
    ntrainM <- as.integer(trainpercentage / 100*nrowsM)
    nrowsB  <- length(bcdataB[,2])
    ntrainB <- as.integer(trainpercentage / 100*nrowsB)

    seqM <- sample(nrowsM)
    seqB <- sample(nrowsB)

    xcMtrain <- bcdataM[seqM[1:ntrainM], 1:30]
    xcMtest <- bcdataM[seqM[(ntrainM + 1):nrowsM], 1:30]
    xcBtrain <- bcdataB[seqB[1:ntrainB], 1:30]
    xcBtest <- bcdataB[seqB[(ntrainB + 1):nrowsB], 1:30]
```

```
    yMtrain <- matrix(-1, ntrainM)
    yMtest <- matrix(-1, nrows - ntrainM)
    yBtrain <- matrix(1, ntrainB)
    yBtest <- matrix(1, nrowsB - ntrainB)

    xin <- as.matrix(rbind(xcMtrain, xcBtrain))
    yd <- rbind(yMtrain, yBtrain)

    retlist <- treinaELM(xin, yd, i, 1)
    W <- retlist[[1]]
    H <- retlist[[2]]
    Z <- retlist[[3]]

    xin_test <- as.matrix(rbind(xcMtest, xcBtest))
    y_hat <- YELM(xin_test, Z, W, 1)
    y_test <- rbind(yMtest, yBtest)
    y_test <- (y_test + 1) / 2
    y_hat <- (y_hat + 1) / 2

    acuracia1[k,j] <- 1 - (t(y_test - y_hat) %*% (y_test - y_hat)) / length(xin_test[,1])

    y_hat1 <- YELM(xin, Z, W, 1)
    yd <- (yd + 1) / 2
    y_hat1 <- (y_hat1 + 1) / 2

    acuracia2[k,j] <- 1 - (t(yd - y_hat1) %*% (yd - y_hat1)) / length(xin[,1])
  }
}

r1 <- c()
d1 <- c()
r2 <- c()
d2 <- c()
```

```
for(i in 1:11) {
  r1[i] <- mean(acuracia1[i,])
  d1[i] <- sd(acuracia1[i,])
  r2[i] <- mean(acuracia2[i,])
  d2[i] <- sd(acuracia2[i,])
}

plot(neuronios, r1, type = "l", col = "blue", ylim = c(0,1))
lines(neuronios, r2, col = "red")

res_f <- data.frame(neuronios, r1, d1, r2, d2)
@
```

Primeiramente, foi implementado um loop com o intuito de percorrer os valores dos neurônios e em seguida, um segundo loop for de dez repetições com o intuito de gerar dez dados aleatórios para cada neurônio. Com isso, é calculado o desvio e o resultado, como pode ser visto mais abaixo:

| | neuronios | r1 | d1 | r2 | d2 |
|---|---|---|---|---|---|
| 1 | 5 | 0.8750000 | 0.03163726 | 0.8879093 | 0.016074047 |
| 2 | 10 | 0.9156977 | 0.00685181 | 0.9214106 | 0.006696021 |
| 3 | 30 | 0.9488372 | 0.01521040 | 0.9677582 | 0.004866961 |
| 4 | 50 | 0.9494186 | 0.01425440 | 0.9760705 | 0.005473722 |
| 5 | 100 | 0.9470930 | 0.01608644 | 0.9803526 | 0.005283672 |
| 6 | 110 | 0.9401163 | 0.02436780 | 0.9861461 | 0.005966696 |
| 7 | 180 | 0.8813953 | 0.01801388 | 0.9924433 | 0.003562251 |
| 8 | 200 | 0.8598837 | 0.02518634 | 0.9954660 | 0.003096405 |
| 9 | 300 | 0.6401163 | 0.10998638 | 0.7403023 | 0.170849656 |
| 10 | 400 | 0.5930233 | 0.17349021 | 0.6526448 | 0.251279559 |
| 11 | 600 | 0.5377907 | 0.16817530 | 0.6098237 | 0.176000065 |

**Figura 1.** Tabela resultante da implementação

Em seguida, é calculado a média e desvio padrão, gerando-se um gráfico que mede a precisão do treinamento e do teste.

```
> mean(acuracia1)
[1] 0.8262156
> sd(acuracia1)
[1] 0.1689039
> mean(acuracia2)
[1] 0.882757
> sd(acuracia2)
[1] 0.1717461
```

**Figura 2.** Precisão do Teste (Acuracia1) e Precisão do Treinamento (Acuracia2)

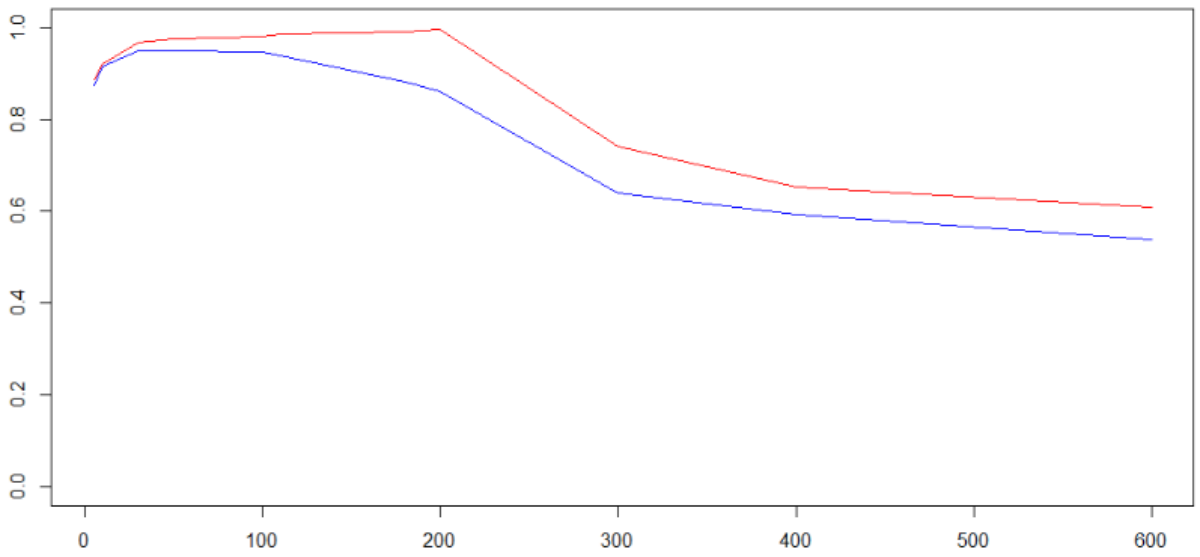


**Figura 3.** Gráfico comparativo

Ao observarmos a precisão de cada neurônio da função ELM, notou-se um tendência de melhorar da precisão até uma determida quantidade de neurônios, e a partir deste valor, a precisão tende a  cair.

No exemplo anterior, a melhor eficiência foi observada com 50 neurônios, vinculada a um precisão de cerca de 95%. No caso do treinamento com Perceptron, a precisão foi de cerca de 82%. Com isso, constatou-se que o primeiro modelo foi mais efetivo na resolução do problema.

## 2. Statlog (heart)

Para este segundo exemplo, foi realizado uma implementação similar a este segundo caso, porém agora, utilizando-se uma base de dados Statlog(heart). Assim, tem-se a seguinte implementação:

```
<<>>=
# Caso 2: Statlog(Heart)

neuronios <- c(5, 10, 30, 50, 100, 110, 180, 200, 300, 400, 600)
k <- 0
acuracia1 <- matrix(0, nrow = 11, ncol = 10)
acuracia2 <- matrix(0, nrow = 11, ncol = 10)

for(i in neuronios) {
  k <- k + 1
  for(j in 1:10) {
    bcdata <- read.csv("~/Projetos/Trabalhos-Faculdade/Redes\ Neurais\ Artificiais/Exercicios/06/heart.csv")
    bcdata$result[bcdata$class == "present"] <- (-1)
    bcdata$result[bcdata$class == "absent"] <- 1
    bcdata$class <- NULL
    bcdata <- as.matrix(bcdata)
    bcdata[,1:13] <- normalize(bcdata[,1:13])
    bcdata <- as.data.frame(bcdata)
    train_percentage <- 70
    bcdataM <- as.matrix(bcdata[bcdata$result == (-1),])
    bcdataB <- as.matrix(bcdata[bcdata$result == 1,])

    nrowsM  <- length(bcdataM[,2])
    ntrainM <- as.integer(train_percentage / 100 * nrowsM)
    nrowsB  <- length(bcdataB[,2])
    ntrainB <- as.integer(train_percentage / 100 * nrowsB)

    seqM <- sample(nrowsM)
    seqB <- sample(nrowsB)

    xcMtrain <- bcdataM[seqM[1:ntrainM],]
    xcMtest  <- bcdataM[seqM[(ntrainM+1):nrowsM],]
    xcBtrain <- bcdataB[seqB[1:ntrainB],]
    xcBtest  <- bcdataB[seqB[(ntrainB+1):nrowsB],]
    yMtrain  <- matrix(-1, ntrainM)
    yMtest   <- matrix(-1, nrowsM - ntrainM)
    yBtrain  <- matrix(1, ntrainB)
    yBtest   <- matrix(1, nrowsB - ntrainB)
```

```
    xin <- as.matrix(rbind(xcMtrain, xcBtrain))[,1:13]
    yd  <- rbind(yMtrain, yBtrain)

    retlist <- treinaELM(xin, yd, i, 1)

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

    xin_test <- as.matrix(rbind(xcMtest, xcBtest))[,1:13]
    y_hat <- YELM(xin_test, Z, W, 1)
    y_test <- rbind(yMtest, yBtest)
    y_test <- (y_test + 1) / 2
    y_hat <- (y_hat + 1) / 2

    acuracia1[k,j] <- 1 - (t(y_test - y_hat) %*% (y_test - y_hat)) / length(xin_test[,1])

    y_hat1 <- YELM(xin, Z, W, 1)
    yd <- (yd + 1) / 2
    y_hat1 <- (y_hat1 + 1) / 2
    acuracia2[k, j] <- 1 - (t(yd - y_hat1) %*% (yd - y_hat1)) / length(xin[,1])
  }
}

r1 <- c()
d1 <- c()
r2 <- c()
d2 <- c()

for(i in 1:11){
  r1[i] <- mean(acuracia1[i,])
  d1[i] <- sd(acuracia1[i,])
  r2[i] <- mean(acuracia2[i,])
  d2[i] <- sd(acuracia2[i,])
}

plot(neuronios, r1, type = "l", col = "blue", ylim = c(0,1))
lines(neuronios, r2, col = "red")

res_f <- data.frame(neuronios, r1, d1, r2, d2)
@
```

Assim , como no exemplo anterior, foi implementado um loop com o intuito de percorrer os valores dos neurônios e em seguida, um segundo loop for de dez repetições com o intuito de gerar dez dados aleatórios para cada neurônio. Com isso, é calculado o desvio e o resultado, como pode ser visto mais abaixo:

| | neuronios | r1 | d1 | r2 | d2 |
|---|---|---|---|---|---|
| 1 | 5 | 0.7567901 | 0.05762786 | 0.7608466 | 0.04216606 |
| 2 | 10 | 0.8123457 | 0.05929354 | 0.8349206 | 0.02415646 |
| 3 | 30 | 0.8160494 | 0.04884827 | 0.8708995 | 0.01767192 |
| 4 | 50 | 0.7086420 | 0.05940768 | 0.7851852 | 0.08833145 |
| 5 | 100 | 0.5296296 | 0.04635798 | 0.5592593 | 0.08546243 |
| 6 | 110 | 0.4987654 | 0.05179303 | 0.5560847 | 0.07200650 |
| 7 | 180 | 0.5419753 | 0.07686776 | 0.5724868 | 0.06996231 |
| 8 | 200 | 0.5197531 | 0.08094557 | 0.5322751 | 0.09996127 |
| 9 | 300 | 0.5345679 | 0.11508663 | 0.5460317 | 0.11835779 |
| 10 | 400 | 0.5530864 | 0.09890251 | 0.5539683 | 0.08982770 |
| 11 | 600 | 0.5098765 | 0.14781054 | 0.5232804 | 0.11905611 |

**Figura 4.** Tabela resultante da implementação

Em seguida, é calculado a média e desvio padrão, gerando-se um gráfico que mede a precisão do treinamento e do teste.

```
> mean(acuracia1)
[1] 0.6164983
> sd(acuracia1)
[1] 0.1459542
> mean(acuracia2)
[1] 0.6450216
> sd(acuracia2)
[1] 0.1523473
```

**Figura 5.** Precisão do Teste (Acuracia1) e Precisão do Treinamento (Acuracia2)

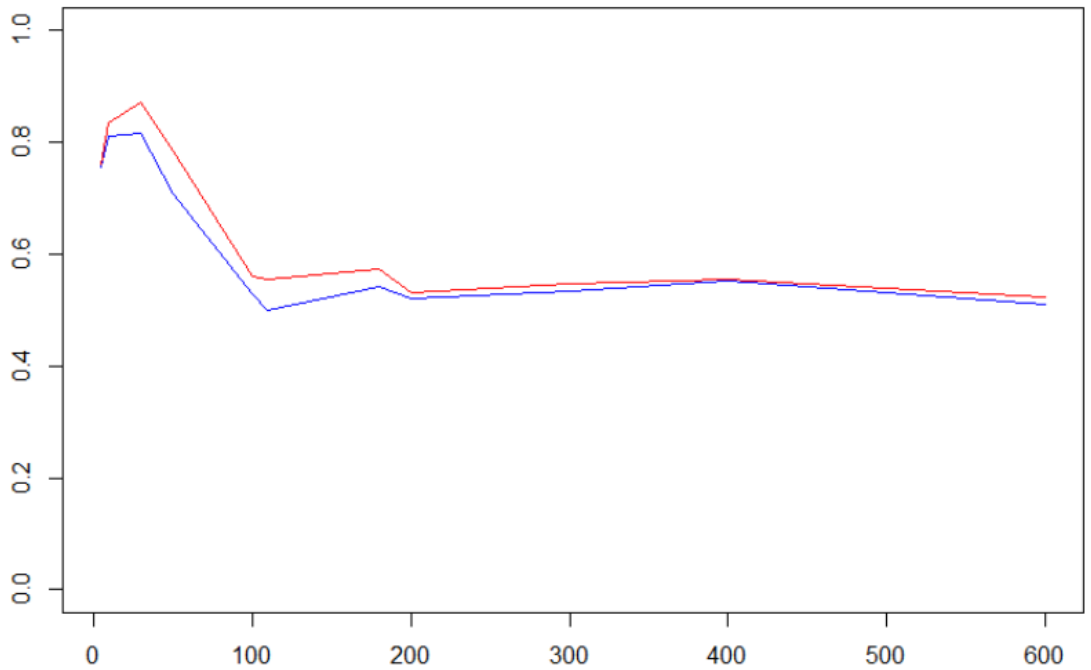


**Figura 6.** Gráfico comparativo

Ao se observar a precisão vinculado o número de neurônios associados com a função ELM, nota-se que como no primeiro caso, ocorre uma tendência de aumentar a precisão até um determinado número de neurônios, e a partir disso, passar a diminuir.

Neste segundo caso, a melhor eficiência com obtida com 30 neurônios, tendo um precisão de cerca de 81 %, e a partir disso, perdendo eficiência. Com treinamento de perceptron, a precisão obtida foi de 64%, que diante da aplicação com ELM, mostrou-se mais eficiente, que atingiu cerca de 61% de precisão. Além disso, a precisão do perceptron foi maior que a do ELM na maioria dos casos.