Exercício 4
Nome: Lucas Kou Kinoshita
RM: 2019021557
data: 06/04/2025

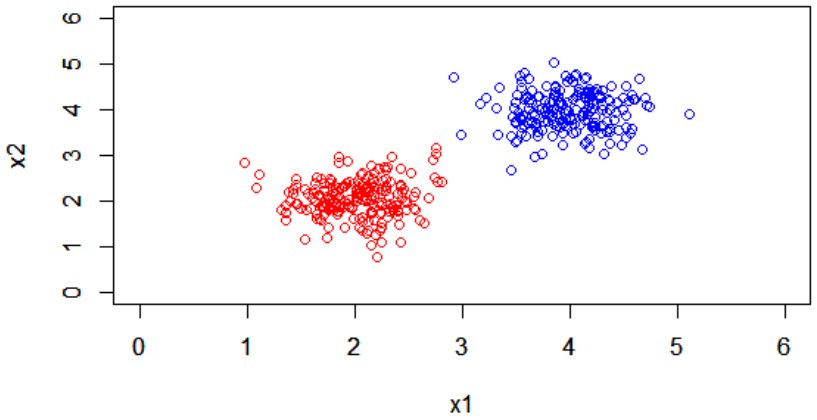
Exercício 1:

Inicialmente, define-se a função responsável pelo treinamento do modelo a partir do perceptron simples.

```r
perceptron <- function(X, y, eta, tol, maxepocas, par) {
  N <- nrow(X) # Número de linhas de X
  n <- ncol(X)
  error_curve <- numeric(maxepocas) # Vetor para armazenar o erro por época

  # inicializando pesos
  if(par == 1){
    w <- as.matrix(runif(n+1) - 0.5)
    X <- cbind(1,X)
  } else {
    w <- as.matrix(runif(n) - 0.5)
  }

  nepocas <- 0
  erroepoca <- tol + 1 # Inicializa erro acima do limite para entrar no loop

  while (erroepoca > tol && nepocas < maxepocas) {
    xseq <- sample(N) # Embaralha os índices
    ei2 <- 0 # Inicializa erro da época

    for (i in xseq) {
      yhat <- 1.0*(X[i, ] %*% w) >= 0
      erro <- y[i] - yhat
      dw <- eta * erro * X[i, ] # Atualização dos pesos
      w <- w + dw
      ei2 <- ei2 + erro^2 # Acumula erro quadrático
    }

    error_curve[nepocas] <- ei2 / N # Erro médio quadrático por época
    nepocas <- nepocas + 1
  }

  list(weights = w, error = error_curve[1:(nepocas - 1)])
}
```

Geram-se os dados sintéticos assim como indicados no enunciado

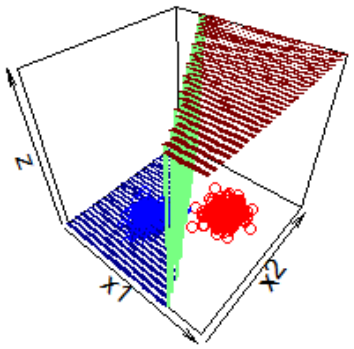
```r
s1 <- 0.4
s2 <- 0.4
nc <- 200
xc1 <- matrix(rnorm(nc*2), ncol = 2)*s1 + t(matrix((c(2,2)), ncol = nc, nrow = 2))
xc2 <- matrix(rnorm(nc*2), ncol = 2)*s2 + t(matrix((c(4,4)), ncol = nc, nrow = 2))

plot(xc1[,1], xc1[,2], xlim = c(0,6), ylim = c(0,6), xlab = 'x1', ylab = 'x2', col = 'red')
par(new=TRUE)
plot(xc2[,1], xc2[,2], xlim = c(0,6), ylim = c(0,6), xlab = 'x1', ylab = 'x2', col = 'blue')
```



Os dados gerados são usados para o treinamento do modelo, resultando na obtenção da curva de erro e dos valores de peso (*w*), os quais serão usados para a visualização da superfície de separação..

```
xall <- rbind(xc1, xc2)
yall <- rbind(matrix(0, ncol = 1, nrow = nrow(xc1)),
              matrix(1, ncol = 1, nrow = nrow(xc2)))
retlist<- perceptron(xall, yall, 0.1, 0.1, 100, 1)

final_weights <- retlist$weights
error_curve <- retlist$error
```

```
seqx1x2 <- seq(0,6, 0.2)
npgrid <- length(seqx1x2)
M <- matrix(nrow = npgrid, ncol = npgrid)
ci <- 0
for(x1 in seqx1x2){
  ci <- ci + 1
  cj <- 0
  for(x2 in seqx1x2){
    cj<- cj + 1
    xin <- as.matrix(cbind(1, x1,x2))
    M[ci,cj] <- 1*((xin %*% final_weights) >= 0)
  }
}

par(new=FALSE)
ribbon3D(seqx1x2, seqx1x2,
         xlab = 'x1', ylab = 'x2',
         xlim=c(0,6), ylim = c(0,6), M, colkey = F)

scatter3D(xc1[,1], xc1[,2],
          xlab = 'x1', ylab = 'x2',
          matrix(0, nrow = dim(xc1)[1]),
          add = T, col = 'blue',
          colkey = F)
scatter3D(xc2[,1], xc2[,2],
          xlab = 'x1', ylab = 'x2',
          matrix(0, nrow = dim(xc1)[1]),
          add = T, col = 'red',
          colkey = F)
```



Exercício 2:

Agora, diferentemente do exercício anterior, os dados sintéticos são separados em dados de treinamento e teste, com proporção de 7:3.

```
s1 <- 0.4
s2 <- 0.4
nc <- 200
xc1 <- matrix(rnorm(nc*2), ncol = 2)*s1 + t(matrix((c(2,2)), ncol = nc, nrow = 2))
xc2 <- matrix(rnorm(nc*2), ncol = 2)*s2 + t(matrix((c(4,4)), ncol = nc, nrow = 2))

plot(xc1[,1], xc1[,2], xlim = c(0,6), ylim = c(0,6), xlab = 'x1', ylab = 'x2', col = 'red')
par(new=TRUE)
plot(xc2[,1], xc2[,2], xlim = c(0,6), ylim = c(0,6), xlab = 'x1', ylab = 'x2', col = 'blue')

ntrain <- nc*0.7

seqc1 <- sample(nc)
xc1treina <- xc1[seqc1[1:ntrain],]
yc1treina <- matrix(0, nrow=ntrain)

seqc2<-sample(nc)
xc2treina <- xc2[seqc2[1:ntrain],]
yc2treina <- matrix(1, nrow = ntrain)

xc1teste <- xc1[seqc1[(ntrain + 1): nc],]
yc1teste <- matrix(0, nrow=(nc-ntrain))
xc2teste <- xc2[seqc2[(ntrain + 1): nc],]
yc2teste <- matrix(1, nrow = (nc-ntrain))
```

Em seguida, o modelo é treinado e testado com os conjuntos de dado apropriados.

```
xall <- rbind(xc1treina, xc2treina)
yall <- rbind(matrix(0, ncol = 1, nrow = nrow(xc1treina)),
              matrix(1, ncol = 1, nrow = nrow(xc2treina)))

retlist<- perceptron(xall, yall, 0.1, 0.1, 100, 1)
final_weights <- retlist$weights
error_curve <- retlist$error

xinteste <- rbind(xc1teste, xc2teste)
yteste <- rbind(yc1teste, yc2teste)

xinteste <- cbind(xinteste, 1)
yt <- 1*((xinteste %*% final_weights) >= 0)

acuracia <- 1 - (t(yteste - yt) %*% (yteste - yt))/nrow(yteste)
```

Resultando na seguinte acurácia e matriz de confusão.

```
> cat("Matriz de Confusão (Teste):\n")
Matriz de Confusão (Teste):
> print(table(Predito = yt, Verdadeiro = yteste))
       Verdadeiro
Predito  0  1
      0 60  0
      1  0 60
```

Exercício 3:

Primeiramente, o dataset é importado e as matrizes de interesse (para treino e teste com proporção de 7:3) e rótulos de saída são definidos.

```
data(iris)

xc1 <- as.matrix(iris[1:50, 1:4])
xc2 <- as.matrix(iris[51:150, 1:4])

ntrain <- 35
seqc1 <- sample(50)

plot(iris, col = c("red", "green3", "blue")[unclass(iris$Species)])

xc1treina <- xc1[seqc1[1:ntrain],]
yc1treina <- matrix(0, nrow=ntrain)
seqc2<-sample(50)
xc2treina <- xc2[seqc2[1:ntrain],]
yc2treina <- matrix(1, nrow = ntrain)

xc1teste <- xc1[seqc1[(ntrain + 1): 50],]
yc1teste <- matrix(0, nrow=(50-ntrain))
xc2teste <- xc2[seqc2[(ntrain + 1): 50],]
yc2teste <- matrix(1, nrow = (50-ntrain))
```

Em seguida, usando a mesma função para o treino do modelo a partir do perceptron simples usada nos exercícios anteriores, obtêm-se os pesos (w) (a partir dos quais o conjunto de testes será validado) e a curva de erro.

```
xin <- as.matrix(rbind(xc1treina, xc2treina))
yd <- rbind(yc1treina, yc2treina)
xinteste <- as.matrix(rbind(xc1teste, xc2teste))
yteste <- rbind(yc1teste, yc2teste)


retlist <- perceptron(xin, yd,0.1, 0.01, 1000, 1)
wt <- retlist$weights
error_curve <- retlist$error

# Aproximação
xinteste <- cbind(1, xinteste)
yt <- 1*((xinteste %*% wt) >= 0)
```

```
> acuracia <- 1 - (t(yteste - yt) %*% (yteste - yt))/nrow(yteste)
> cat("Acurácia (Teste):", round(acuracia, 4), "\n")
Acurácia (Teste): 1
> cat("Matriz de Confusão (Teste):\n")
Matriz de Confusão (Teste):
> print(table(Predito = yt, Verdadeiro = yteste))
       Verdadeiro
Predito  0  1
      0 15  0
      1  0 15
```

Por fim, cria-se um loop de 100 repetições do processo de treinamento para obter o valor média e as variâncias dos conjuntos de treino e teste

```r
for (r in 1:reps) {
  # Define dados
  xc1 <- as.matrix(iris[1:50, 1:4])        # Setosa
  xc2 <- as.matrix(iris[51:150, 1:4])      # Versicolor + Virginica

  seqc1 <- sample(50)
  seqc2 <- sample(100)

  # Treino
  xc1treina <- xc1[seqc1[1:ntrain],]
  yc1treina <- matrix(0, nrow=ntrain)
  xc2treina <- xc2[seqc2[1:ntrain],]
  yc2treina <- matrix(1, nrow=ntrain)
  # Teste
  xc1teste <- xc1[seqc1[(ntrain + 1): 50],]
  yc1teste <- matrix(0, nrow=(50 - ntrain))
  xc2teste <- xc2[seqc2[(ntrain + 1): 100],]
  yc2teste <- matrix(1, nrow=(100 - ntrain))

  # Conjuntos
  xin <- rbind(xc1treina, xc2treina)
  yd <- rbind(yc1treina, yc2treina)

  xinteste <- rbind(xc1teste, xc2teste)
  yteste <- rbind(yc1teste, yc2teste)

  # Treinamento
  retlist <- perceptron(xin, yd, eta = 0.1, tol = 0.01, maxepocas = 1000, par = 1)
  wt <- retlist$weights

  # Acurácia treino
  xitreino <- cbind(1, xin)
  ytreino_pred <- 1*((xitreino %*% wt) >= 0)
  acc_treino[r] <- 1 - (t(yd - ytreino_pred) %*% (yd - ytreino_pred)) / nrow(yd)

  # Acurácia teste
  xiteste <- cbind(1, xinteste)
  yteste_pred <- 1*((xiteste %*% wt) >= 0)
  acc_teste[r] <- 1 - (t(yteste - yteste_pred) %*% (yteste - yteste_pred)) / nrow(yteste)
}
```

```
> # Resultados
> cat("Treinamento:\n")
Treinamento:
> cat("Média da acurácia:", round(mean(acc_treino), 4), "\n")
Média da acurácia: 1
> cat("Variância da acurácia:", round(var(acc_treino), 6), "\n\n")
Variância da acurácia: 0

>
> cat("Teste:\n")
Teste:
> cat("Média da acurácia:", round(mean(acc_teste), 4), "\n")
Média da acurácia: 0.9948
> cat("Variância da acurácia:", round(var(acc_teste), 6), "\n")
Variância da acurácia: 0.000273
```

Exercício 4:

De forma semelhante ao exercício 3, os dados foram importados para o script, mas devido a organização, a lógica de separação e tratamento dos dados foi alterada, além disso o número máximo de épocas para o treinamento do modelo foi reduzido de 1000 para 100 em função do tempo de processamento.

```
# Load BreastCancer
data("BreastCancer")
df <- BreastCancer
df <- df[, -1]  # drop 'Id'
df <- na.omit(df)

# Convert factor columns to numeric (except 'Class')
df[, 1:9] <- lapply(df[, 1:9], function(x) as.numeric(as.character(x)))
df$class <- ifelse(df$class == "malignant", 1, 0)

# Matrix version
X <- as.matrix(df[, 1:9])
y <- as.matrix(df$class)
N <- nrow(X)

ntrain <- nrow(df)*0.7  # you can adjust this
reps <- 100
acc_treino <- numeric(reps)
acc_teste <- numeric(reps)
```

```r
for (r in 1:reps) {
  # Shuffle indices
  idx <- sample(N)

  # Train/Test split
  train_idx <- idx[1:ntrain]
  test_idx <- idx[(ntrain + 1):N]

  xin <- X[train_idx, ]
  yd <- y[train_idx]
  xinteste <- X[test_idx, ]
  yteste <- y[test_idx]

  # Train perceptron
  retlist <- perceptron(xin, yd, eta = 0.1, tol = 0.01, maxepocas = 100, par = 1)
  wt <- retlist$weights

  # Train accuracy
  xitreino <- cbind(1, xin)
  ytreino_pred <- 1 * ((xitreino %*% wt) >= 0)
  acc_treino[r] <- 1 - mean((yd - ytreino_pred)^2)

  # Test accuracy
  xiteste <- cbind(1, xinteste)
  yteste_pred <- 1 * ((xiteste %*% wt) >= 0)
  acc_teste[r] <- 1 - mean((yteste - yteste_pred)^2)
  print(r)
}

# Resultados
cat("Treinamento:\n")
cat("Média da acurácia:", round(mean(acc_treino), 4), "\n")
cat("Variância da acurácia:", round(var(acc_treino), 6), "\n\n")

cat("Teste:\n")
cat("Média da acurácia:", round(mean(acc_teste), 4), "\n")
cat("Variância da acurácia:", round(var(acc_teste), 6), "\n")
```

```
Treinamento:
> cat("Média da acurácia:", round(mean(acc_treino), 4), "\n")
Média da acurácia: 0.9578
> cat("Variância da acurácia:", round(var(acc_treino), 6), "\n\n")
Variância da acurácia: 0.001545

>
> cat("Teste:\n")
Teste:
> cat("Média da acurácia:", round(mean(acc_teste), 4), "\n")
Média da acurácia: 0.9513
> cat("Variância da acurácia:", round(var(acc_teste), 6), "\n")
Variância da acurácia: 0.001488
```

Conclusão:

Podemos verificar através da superfície de separação plotada no primeiro exercício e pelas acurácias médias obtidas nos exercícios subsequentes que o algoritmo para o treinamento do modelo com o uso do perceptron simples foi bem sucedido.

Ademais, os baixos valores para as variâncias das acurácias mostra que o modelo é efetivo para diferentes separações e conjuntos de dados.