UNIVERSIDADE FEDERAL DE MINAS GERAIS

DISCIPLINA: Redes Neurais Artificial
PROFESSORES: Frederico Gualberto Ferreira Coelho
ALUNO: Antonio Carlos da Anunciação – 2018019443

TRABALHO PRATICO 3
Aplicação do Adaline para Problemas de Classificações:

Exercício 1: Dada distribuições normais no espaço **R2**, ou seja, duas distribuições com duas variáveis cada, **(X_1, X_2)**, gerando um conjunto de dados com duas classes, caracterizadas como $\mathcal{N}$**([2, 2],** σ **= 0.4)** e $\mathcal{N}$**([4, 4],** σ **= 0.4)**, como pode ser visualizado na **FIG01**. Com o tamanho das amostras ***nc*** igual a 200 para cada classe.

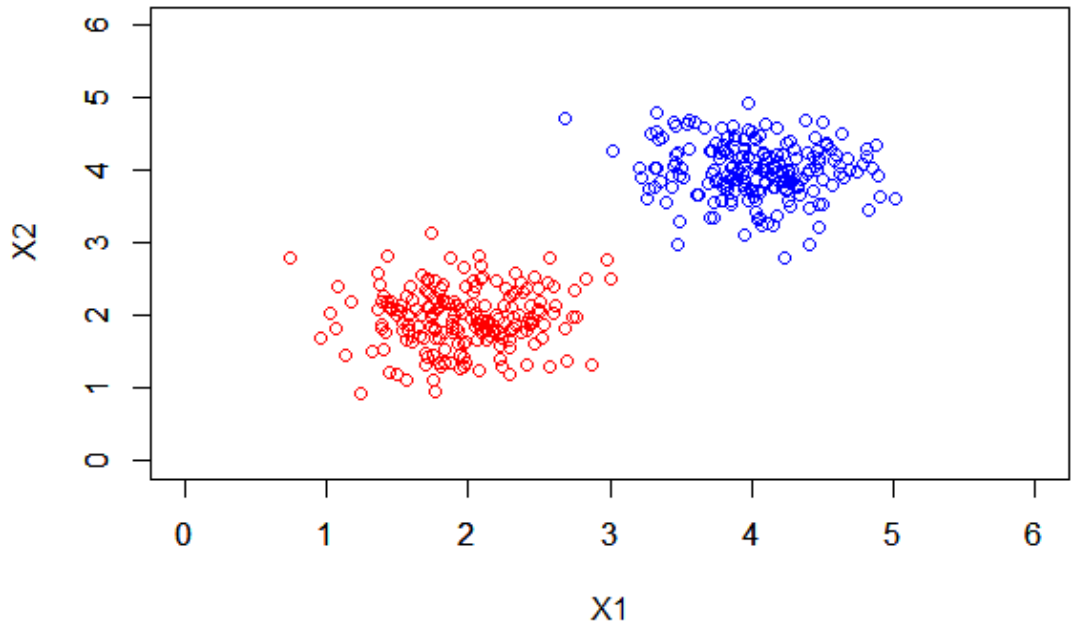

FIG01: Distribuição dos Dados

Neste trabalho será treinado um classificador linear do tipo **Adaline** para resolver o problema de classificação dos dados acima. As saídas para rotulação dos dados da rede será 1 para uma classe e -1 para a outra classe. Os dados serão separados em dois conjuntos, conjunto de treinamento com 90% dos dados e um conjunto de testes com 10% dos dados de forma aleatória.

Resultados:

Parâmetros de Rede:
(Yhat = **a** + **b**x_1 +**c**x_2)

a	**b**	**c**
-2.716	**0.4004**	**0.4579**

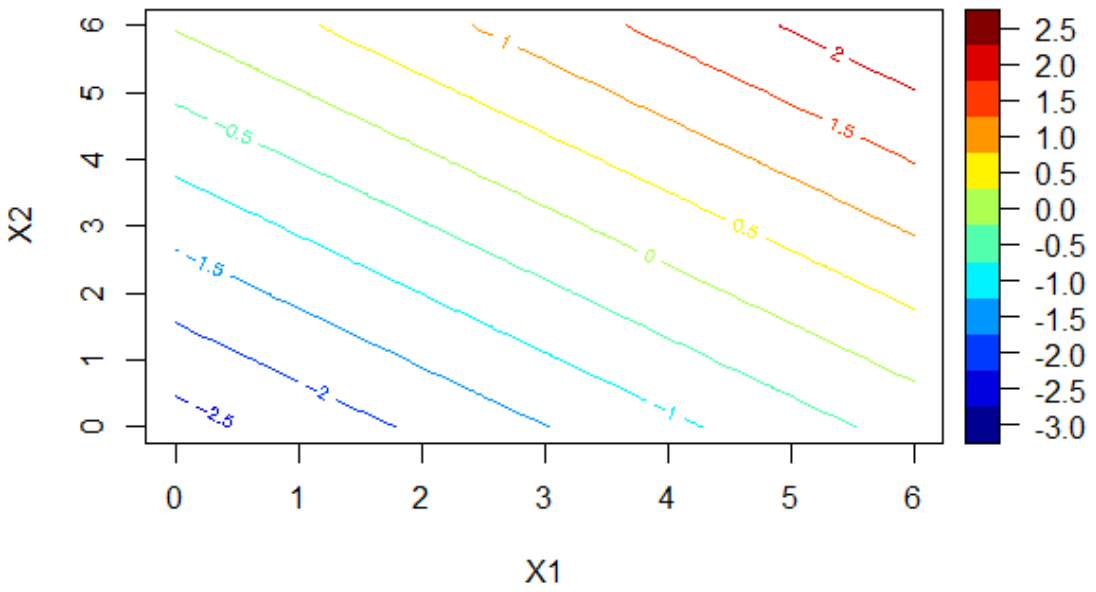

FIG02: Curvas de Níveis

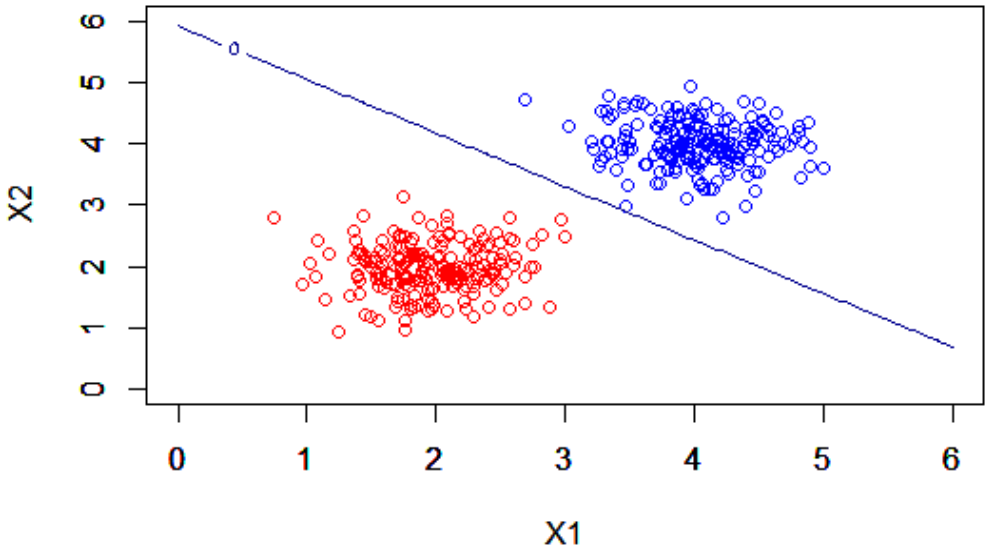

FIG03: Curva de Separação Linear

Podemos inferir que o Adaline gerou uma ótima superfície de separação, a qualidade de superfície de separação se deve ao fato dos dados terem uma boa separabilidade, ou seja, a dispersão dos dados é baixa, desvio padrão pequeno, isto contribuiu para a convergência do método.

A seguir temos os resultados do erro final de predição utilizando os dados de teste e o erro durante o processo de aprendizagem.

Erro Médio Quadrático: **0.0788912**

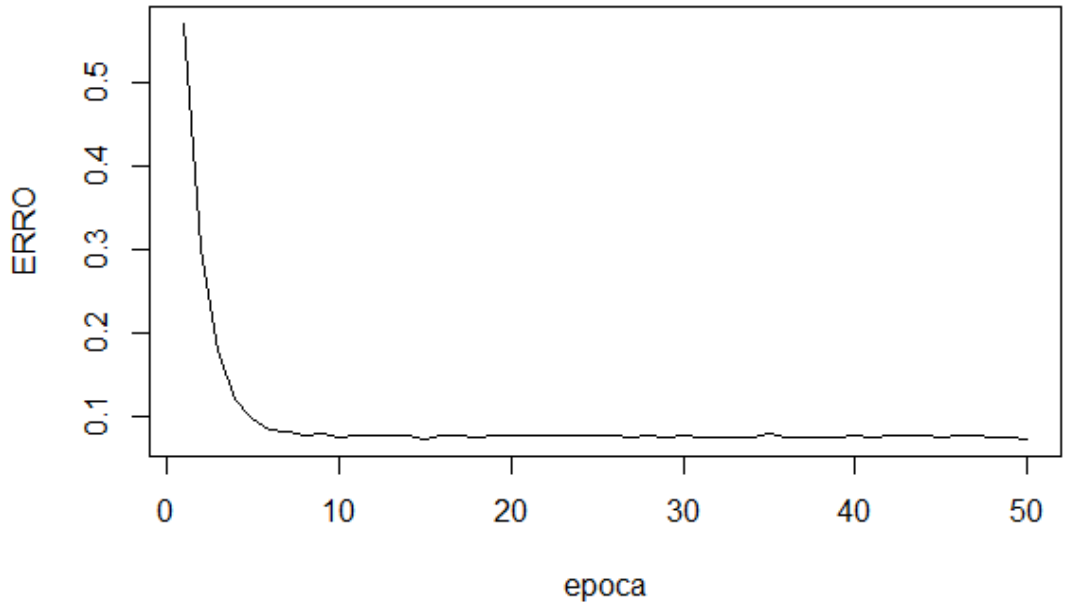

FIG04: Erro no Processo de Aprendizado

Exercício 2: Agora os dados de entrada serão amostrados de quatro gaussianas como mostrado na **FIG05**. As classes estão separadas por cores, conforme indicado.

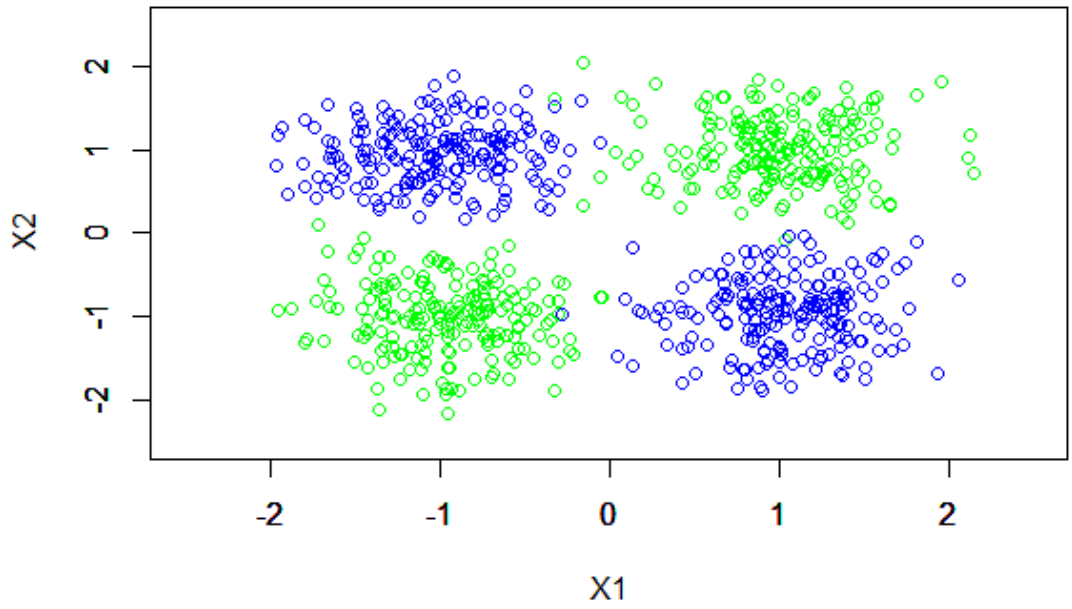

FIG05: Distribuição dos Dados

Para este problema os dados não são linearmente separáveis, mas como pode-se notar visualmente existe uma óbvia separação entre eles. Para podermos classificar corretamente esses dados eles serão divididos em dois subgrupos com duas classe cada um desses subgrupos, um subgrupo separado verticalmente no eixo **X1 = 0**, gerando duas classes, **classe -1** para **X1 < 0** e **classe 1** para **X1 > 0**, é usada a mesma estratégia para separamos os horizontalmente em **X2 = 0**, e novamente **X2 < 0 classe -1** e **X2 > 0**, **classe 1**. A classificação final será o produto das classificações horizontais e verticais. Desse modo no final teremos **Classe -1 para a cor Azul e Classe 1 para a cor Verde**.

Separando os dados para o processo de aprendizagem:

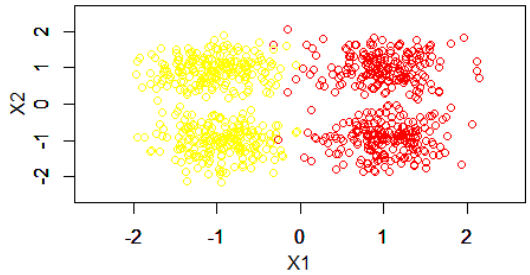

FIG06: Separação em X1

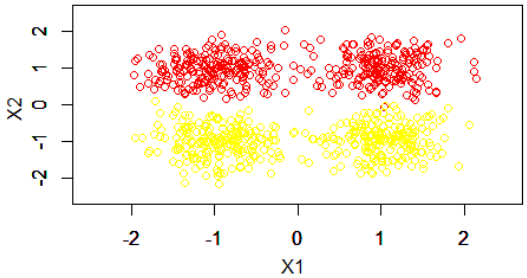

FIG07: Separação em X2

Uma vez separados serão treinadas duas Redes Adaline, cada uma especialista em classificar seu grupo.

Parâmetros Vertical – X1		
a	**b**	**c**
0.01255	**0.8842**	**0.00541**

Parâmetros Horizontal – X2		
a	**b**	**c**
0.02203	**0.00646**	**0.89288**

Erro durante o no processo de aprendizagem dos Classificaadores:

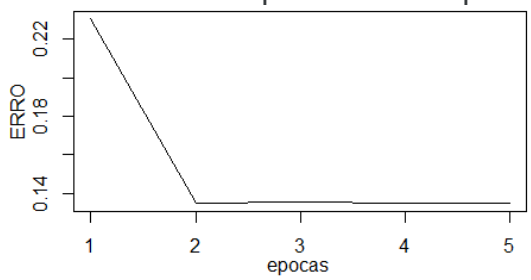

FIG08: Erro de Aprendizagem Vertical – X1

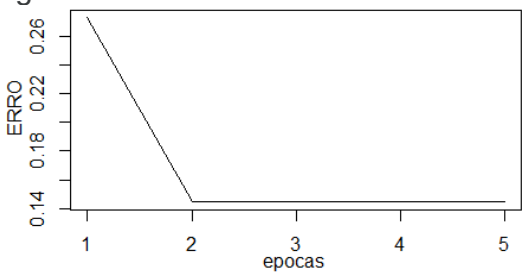

FIG09: Erro de Aprendizagem Horizontal – X2

A classificação final é obtida multiplicando os resultados das classificações individuais de cada subgrupo, gerando a superfície de separação indicado na **FIG10**, abaixo:

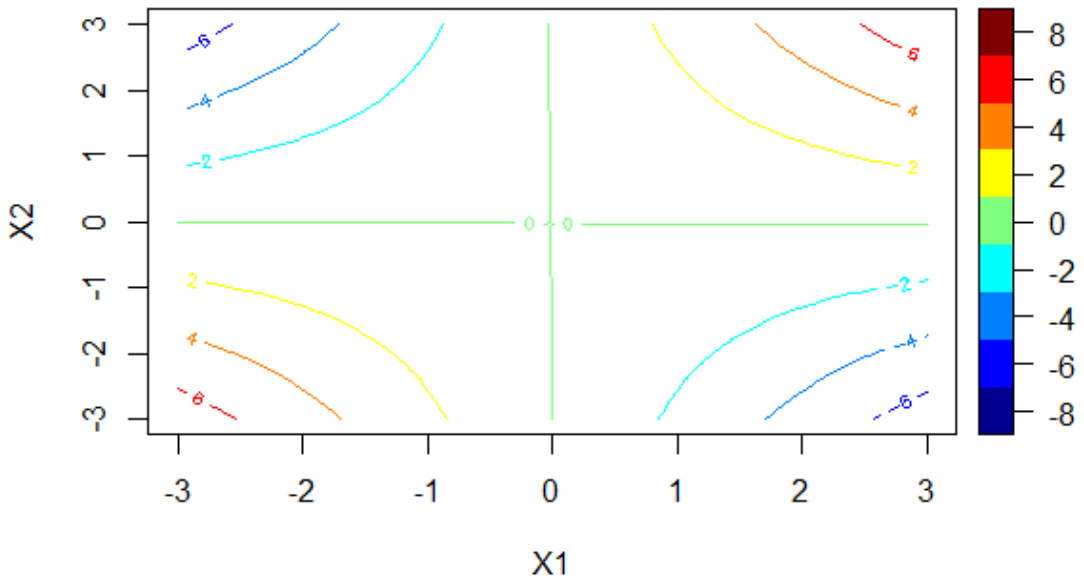

FIG10: Superfície de Classificação

A **FIG10** mostrar as curvas de níveis, nela podemos inferir as classificações dos dados originais, **Verdes** e **Azuis**, onde as regiões das curvas de níveis **menores que zero** indica que ali é a **Classe Azul**, e valores **maiores que zero Classe Verde**.

Erro Médio Quadrático: **3.406399**

A aplicação do modelo retornou um Erro Médio Quadrático maior que a aplicação linearmente separável do exercício 1, isto se deve ao fato de haver uma maior região de interseções entre os dados, porém observando-se a **FIG11** logo abaixo podemos ver que o modelo apresentou uma curva de separação satisfatória.

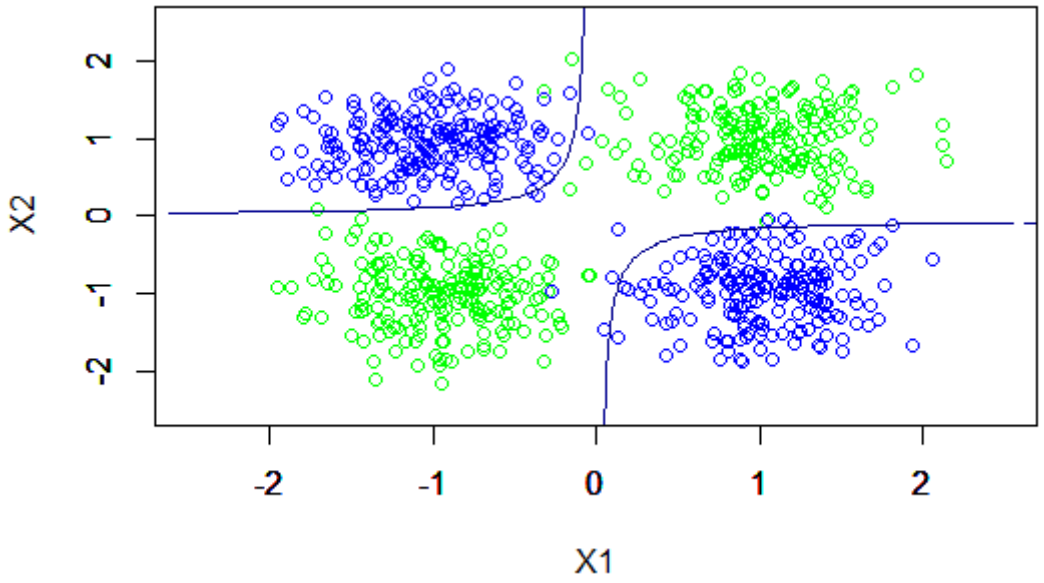

FIG11: Curvas de Separação

Por fim mostramos que o Adaline pode ser usado para alguns problemas de classificação de dados não-linearmente separáveis, se estes dados estiverem distribuídos de determinadas maneiras.

Códigos:

Exercício 1

```
rm(list=ls())
library('plot3D')

# Definiçao dos dados em duas dimensões:
# Declarando as variancias:
s1 <- s2 <- 0.4
nc<-200

# Criando os dados:
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))

trainadaline <- function(xin, yd, eta, tol, maxepocas, par) {
 dimxin <- dim(xin)
 N <- dimxin[1]
 n <- dimxin[2]

 if(par == 1) {
  wt <- as.matrix(runif(n+1)-0.5)
  xin <- cbind(1, xin)
 }
 else wt <- as.matrix(runif(n)-0.5)

 nepocas <- 0
 eepoca <- tol + 1

 evec <- matrix(nrow = 1, ncol = maxepocas)
 while((nepocas < maxepocas) && (eepoca > tol)) {
  ei2 <- 0
  xseq <- sample(N)
  for(i in 1:N) {
   irand <- xseq[i]
   yhati <- 1.0*((xin[irand, ] %*% wt))
   ei <- yd[irand]-yhati
   dw <- eta*ei*xin[irand,]
   wt <- wt + dw
   ei2 <- ei2 + ei*ei
  }
  nepocas <- nepocas + 1
  evec[nepocas] <- ei2/N

  eepoca <- evec[nepocas]
 }
 retlist <- list(wt, evec[1:nepocas])
 return(retlist)
}

# Definição dos dados de treinamento e de teste:
Mtrain<- 0.9*dim(xc1)[1]
seqc1 <- sample(dim(xc1)[1])
seqc2 <- sample(dim(xc2)[1])

xc1train <- xc1[seqc1[(1:Mtrain)], ]
xc1tst <- xc1[seqc1[((Mtrain+1):dim(xc1)[1])], ]
xc1Class <- matrix(nrow = dim(xc1train)[1], ncol = 1,-1)
xc1Classtst <- matrix(nrow = dim(xc1tst)[1], ncol = 1,-1)

xc2train <- xc2[seqc2[(1:Mtrain)], ]
xc2tst <- xc2[seqc2[((Mtrain+1):dim(xc2)[1])], ]
xc2Class <- matrix(nrow = dim(xc2train)[1], ncol = 1,1)
xc2Classtst <- matrix(nrow = dim(xc2tst)[1], ncol = 1,1)

xcTrain <- rbind(xc1train, xc2train)
xcClass <- rbind(xc1Class, xc2Class)

xcTst <- rbind(xc1tst, xc2tst)
xcClassTst <- rbind(xc1Classtst, xc2Classtst)
```

```
eta <- 0.01
tol <- 10^-3
maxepocas <- 10^3
par <- 1

dataTrain <- trainadaline(xcTrain, xcClass, eta, tol, maxepocas, par)

seqi<-seq(0, 6, 6/100)
seqj<-seq(-0, 6, 6/100)

Class <-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
  Class[ci,cj]<- c(1,i,j)%*%dataTrain[[1]]
 }
}

# Plotando os graficos
plot(xc1[,1],xc1[,2], col = 'red', xlim = c(0,6), ylim = c(0,6), xlab = 'X1', ylab = 'X2')
par(new=T)
plot(xc2[,1],xc2[,2], col = 'blue', xlim = c(0,6), ylim = c(0,6), xlab = '', ylab = '')
par(new=T)
contour2D(Class, seqi, seqj, colkey = NULL, xlim = c(0,6), ylim = c(0,6), xlab = '', ylab = '', levels= 0, axis = F)

pesos <- dataTrain[[1]]
validation <- cbind(cbind(1,xcTst)%*%pesos,xcClassTst)
erro <- (validation[,2]-validation[,1])^2
erro_mQ <- mean(erro)

plot(dataTrain[[2]][1:50], type = 'l', xlab = 'epoca', ylab = 'ERRO')
```

Exercício 2

```
rm(list=ls())
library('plot3D')

trainadaline <- function(xin, yd, eta, tol, maxepocas, par) {
 dimxin <- dim(xin)
 N <- dimxin[1]
 n <- dimxin[2]

 if(par == 1) {
  wt <- as.matrix(runif(n+1)-0.5)
  xin <- cbind(1, xin)
 }
 else wt <- as.matrix(runif(n)-0.5)

 nepocas <- 0
 eepoca <- tol + 1

 evec <- matrix(nrow = 1, ncol = maxepocas)
 while((nepocas < maxepocas) && (eepoca > tol)) {
  ei2 <- 0
  xseq <- sample(N)
  for(i in 1:N) {
   irand <- xseq[i]
   yhati <- 1.0*((xin[irand, ] %*% wt))
   ei <- yd[irand]-yhati
   dw <- eta*ei*xin[irand,]
   wt <- wt + dw
   ei2 <- ei2 + ei*ei
  }
  nepocas <- nepocas + 1
  evec[nepocas] <- ei2/N

  eepoca <- evec[nepocas]
 }
 retlist <- list(wt, evec[1:nepocas])
```

```
  return(retlist)
}

# Criando os dados:
s1 <- s2 <- 0.4
nc <- 200
x11<-matrix(rnorm(nc*2), ncol=2)*s1 + t(matrix((c(-1,-1)), ncol=nc, nrow=2))
x12<-matrix(rnorm(nc*2), ncol=2)*s1 + t(matrix((c(1,1)), ncol=nc, nrow=2))
x21<-matrix(rnorm(nc*2), ncol=2)*s2 + t(matrix((c(-1,1)), ncol=nc, nrow=2))
x22<-matrix(rnorm(nc*2), ncol=2)*s2 + t(matrix((c(1,-1)), ncol=nc, nrow=2))
X <- rbind(x11, x12, x21, x22)

# Plotando os graficos
plot(x11[,1],x11[,2], col = 'green', xlim = c(-2.5,2.5), ylim = c(-2.5,2.5), xlab = 'X1', ylab = 'X2')
par(new=T)
plot(x12[,1],x12[,2], col = 'green', xlim = c(-2.5,2.5), ylim = c(-2.5,2.5), xlab = '', ylab = '')
par(new=T)
plot(x21[,1],x21[,2], col = 'blue', xlim = c(-2.5,2.5), ylim = c(-2.5,2.5), xlab = '', ylab = '')
par(new=T)
plot(x22[,1],x22[,2], col = 'blue', xlim = c(-2.5,2.5), ylim = c(-2.5,2.5), xlab = '', ylab = '')

# Separando os dados em um grid vertical e outra horizontal:
XVerticalC1 <- rbind(x11, x21) # Classe -1
XVerticalC1Class <- matrix(-1, nrow = dim(XVerticalC1)[1], ncol = 1)
XVerticalC2 <- rbind(x12, x22) # Classe +1
XVerticalC2Class <- matrix(1, nrow = dim(XVerticalC2)[1], ncol = 1)
XVertical <- rbind(XVerticalC1, XVerticalC2)
XVerticalClass <- rbind(XVerticalC1Class, XVerticalC2Class)

XHorizontalC1 <- rbind(x11, x22) # Classe -1
XHorizontalC1Class <- matrix(-1, nrow = dim(XHorizontalC1)[1], ncol = 1)
XHorizontalC2 <- rbind(x21, x12) # Classe +1
XHorizontalC2Class <- matrix(1, nrow = dim(XHorizontalC2)[1], ncol = 1)
XHorizontal <- rbind(XHorizontalC1, XHorizontalC2)
XHorizontalClass <- rbind(XHorizontalC1Class, XHorizontalC2Class)

# Separando dados em um grupo de teste e outro de treino:
MtrainX<- 0.9*dim(X)[1]
seqX <- sample(dim(X)[1])

XtrainVertical <- XVertical[seqX[(1:MtrainX)], ]
XTestVertical <- XVertical[seqX[((MtrainX+1):dim(X)[1])], ]
XClassTrainVertical <- XVerticalClass[seqX[(1:MtrainX)], ]
XClassTestVertical <- XVerticalClass[seqX[((MtrainX+1):dim(X)[1])], ]

XtrainHorizontal <- XHorizontal[seqX[(1:MtrainX)], ]
XTestHorizontal <- XHorizontal[seqX[((MtrainX+1):dim(X)[1])], ]
XClassTrainHorizontal <- XHorizontalClass[seqX[(1:MtrainX)], ]
XClassTestHorizontal <- XHorizontalClass[seqX[((MtrainX+1):dim(X)[1])], ]

# Parametros de controle do algoritmo:
eta <- 0.01
tol <- 0.001
maxepocas <- 1000
par <- 1

vertical_classify <- trainadaline(XtrainVertical, XClassTrainVertical, eta, tol, maxepocas, par)
Horizontal_classify <- trainadaline(XtrainHorizontal, XClassTrainHorizontal, eta, tol, maxepocas, par)
pesos_verticais <- as.matrix(vertical_classify[[1]])
pesos_horizontais <- as.matrix(Horizontal_classify[[1]])

# Classificação final (Classificação Vertical X horizontal):

seqi<-seq(-3, 3, 6/100)
seqj<-seq(-3, 3, 6/100)

Class <-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
    Class[ci,cj]<- (c(1,i,j)%*%pesos_verticais)*(c(1,i,j)%*%pesos_horizontais)
  }
}

plot(x11[,1],x11[,2], col = 'green', xlim = c(-2.5,2.5), ylim = c(-2.5,2.5), xlab = '', ylab = '')
par(new=T)
plot(x12[,1],x12[,2], col = 'green', xlim = c(-2.5,2.5), ylim = c(-2.5,2.5), xlab = '', ylab = '')
par(new=T)
plot(x21[,1],x21[,2], col = 'blue', xlim = c(-2.5,2.5), ylim = c(-2.5,2.5), xlab = '', ylab = '')
par(new=T)
plot(x22[,1],x22[,2], col = 'blue', xlim = c(-2.5,2.5), ylim = c(-2.5,2.5), xlab = '', ylab = '')
par(new=T)
contour2D(Class, seqi, seqj, colkey = NULL, xlim = c(-2.5,2.5), ylim = c(-2.5,2.5), xlab = 'X1', ylab = 'X2', levels = -0.1:0.1)


plot(vertical_classify[[2]][0:5], type='l', xlab = 'epocas', ylab = 'ERRO')
plot(Horizontal_classify[[2]][0:5], type='l', xlab = 'epocas', ylab = 'ERRO')

XTest <- rbind(XTestVertical, XTestHorizontal)
XTest <- cbind(1, XTest)
XTestClass <- rbind(as.matrix(XClassTestVertical), as.matrix(XClassTestHorizontal))
erroT <- cbind((XTest%*%pesos_verticais)*(XTest%*%pesos_horizontais),XTestClass)
erroMQ <- mean((erroT[2]-erroT[1])^2)
```