Treinamento Perceptron Simples

Matheus Araujo - 2013066265

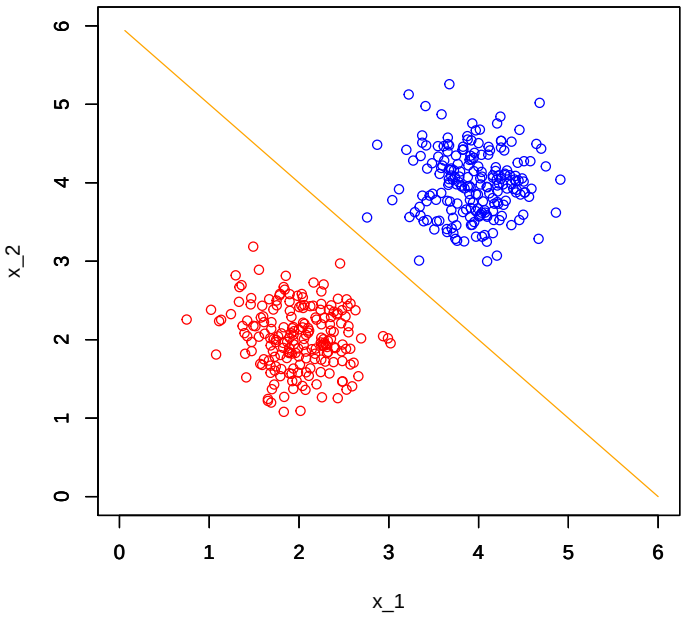
1 Treinamento Perceptron Simples

Nessa atividade, a técnica de *Perceptron Simples* será aprimorada através de treinamento. Em vez de usarmos pesos calculados para o vetor de pesos da Rede Neural, eles serão calculados por uma função de treinamento.

1.1 Geração de dados

Inicialmente, os dados são gerados conforme o código a seguir:

```
>   rm(list = ls())
>   library('plot3D')
>   source('treinap.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],col='red',xlim=c(0,6),ylim=c(0,6),xlab='x_1',ylab='x_2')
>   par(new=T)
>   plot(xc2[,1],xc2[,2],col='blue',xlim=c(0,6),ylim=c(0,6),xlab='',ylab='')
>   x1_reta<-seq(6/100,6,6/100)
>   x2_reta<- -x1_reta+6
>   par(new=T)
>   plot(x1_reta,x2_reta,type='l',col='orange',xlim=c(0,6),ylim=c(0,6),xlab='',ylab='')
```



1.2 Treinamento

Agora os valores para o vetor de pesos são calculados pela função *treinap* no arquivo *treinap.R*.

```
>   X = rbind(xc1,xc2)
>   yc1<-matrix(0,nrow=nc,ncol=1)
>   yc2<-matrix(1,nrow=nc,ncol=1)
>   yd<-rbind(yc1,yc2)
>   w = treinap(X, yd, 0.1, 0.01, 100, 1)
```

1.3 Função de treinamento

A função *treinap* é apresentada a seguir:

```
treinap <- function (xin, yd, eta, tol, maxepocas, par)
{
  dimxin<-dim(xin)
  N<-dimxin[1] # numero de amostras
  n<-dimxin[2] # dimensao de entrada

  # adiciona ou nao termo de polarizacao ao vetor de treinamento w
```

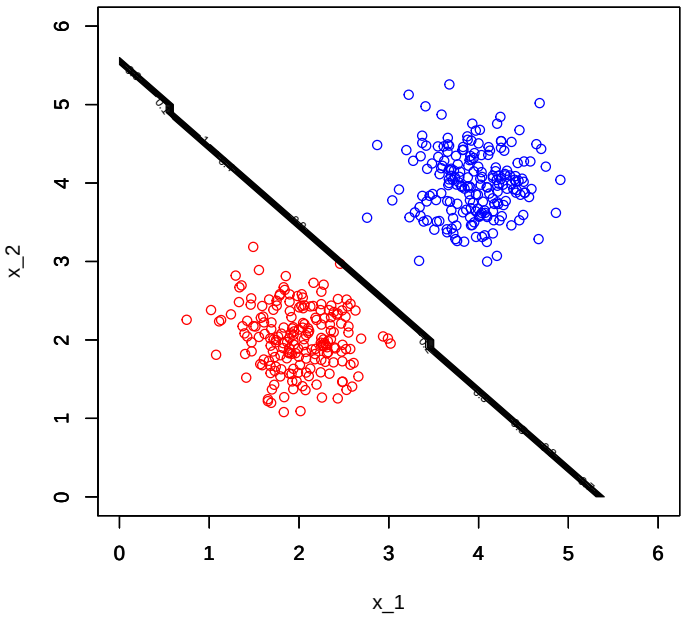
```
  if (par==1) {
    wt<-as.matrix(runif(n+1)-0.5)
    xin<-cbind(-1,xin)
  }
  else {
    wt<-as.matrix(runif(n)-0.5)
  }

  nepocas<-0 # contador de epocas
  eepoca<-tol+1 # acumulador de erro de epocas
  evec<-matrix(nrow=1,ncol=maxepocas) # vetor de erro de epocas

  # laco principal de treinamento
  while((nepocas<maxepocas) && (eepoca>tol)) {
    ei2<-0
    xseq<-sample(N) # sequencia aleatoria de treinamento

    for(i in 1:N) {

      irand<-xseq[i] # amostra dado da sequencia aleatoria

      yhati<-1.0*((xin[irand,]%*%wt) >= 0) # calcula saida do perceptron

      ei<-yd[irand]-yhati
      dw<-eta*ei*xin[irand,]
      wt<-wt+dw # ajusta vetor de pesos
      ei2<-ei2+ei*ei # acumula erro por epoca
    }

    nepocas<-nepocas+1
    evec[nepocas]<-ei2/N
    eepoca<-evec[nepocas]
  }

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

1.4 Resultados

O resultado da função de treinamento é mostrado a seguir:

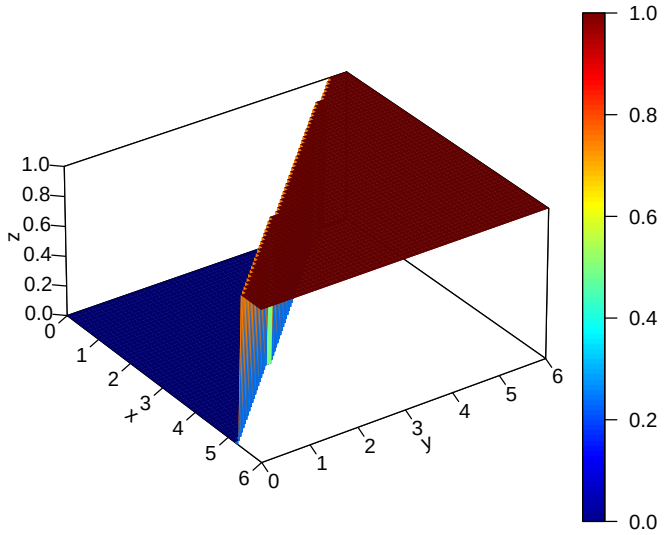
```
>   seqi<-seq(0,6,0.1)
>   seqj<-seq(0,6,0.1)
>   M<-matrix(0,nrow=length(seqi),ncol=length(seqj))
>   ci<-0
```

```
>   for(i in seqi) {
+     ci<-ci+1
+     cj<-0
+     for(j in seqj) {
+       cj<-cj+1
+       M[ci,cj]<-1*(i*w[[1]][2]+j*w[[1]][3]>=w[[1]][1])
+     }
+   }
>   plot(xc1[,1],xc1[,2],col='red',xlim=c(0,6),ylim=c(0,6),xlab='x_1',ylab='x_2')
>   par(new=T)
>   plot(xc2[,1],xc2[,2],col='blue',xlim=c(0,6),ylim=c(0,6),xlab='',ylab='')
>   par(new=T)
>   contour(seqi,seqj,M,xlim=c(0,6),ylim=c(0,6),xlab='',ylab='')
```



Visão em 3D:

```
>     persp3D(seqi,seqj,M,counter=T,theta=55,phi=30,r=40,d=0.1,expand=0.5,ltheta=90,
+             lphi=180,shade=0.4,ticktype="detailed",nticks=5)
```


z
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
x
0
1
2
3
4
5
6
y
1.0
0.8
0.6
0.4
0.2
0.0