Percepetron de Camada Simples para saídas discretas

Prof. Leandro Augusto

Problema da porta lógica AND

É muito comum encontrar em problemas que envolvem RNA utilizar em um primeiro experimento as portas lógicas. Isso deve-se a origem das pesquisas neste tipo de assunto, onde se tinha como pretenção a substituição dos dispositivos eletrônicos portas lógicas por neurônios artificiais, de maneira a tornar os computadores inteligentes.

No entanto, este estudo não teve evolução neste sentido e ficou o uso de portas lógicas como uma referência para mostrar que uma implememetação de RNA funciona de maneira adequada, ou seja, aprende.

No problema a seguir, utilizaremos a porta lógica AND (ou E). Trata-se de um problema com 4 exemplares (objetos ou registros) caracteizados por dois atributos descritivos (variáveis) e um atributo classificatório (saída desejada), sendo todos atributos binários.

Esta base tem como característica que a saída desejada só será 1 quando os dois atributos descritivos forem também 1.

A seguir serão apresentados mais detalhes sobre o dataset.

Leitura e análises exploratória do dataset

Como em todo processo de Mineração de Dados, a primeira etapa do processo é sempre a carga de um dataset.

Para este caso de exemplo com o Perceptron, o dataset a ser usado será a porta lógica AND (E), como dito antes, cuja construção será apresentada a seguir. É importante destacar que em cada exemplar foi adicionado um valor constante, 1, referente ao bias (discutido em aula).

```
x1<-c(0,0,1)
x2<-c(0,1,1)
x3<-c(1,0,1)
x4<-c(1,1,1)

X<-rbind(x1,x2,x3,x4)
print(X)
```

```
##    [,1] [,2] [,3]
## x1    0    0    1
## x2    0    1    1
## x3    1    0    1
## x4    1    1    1
```

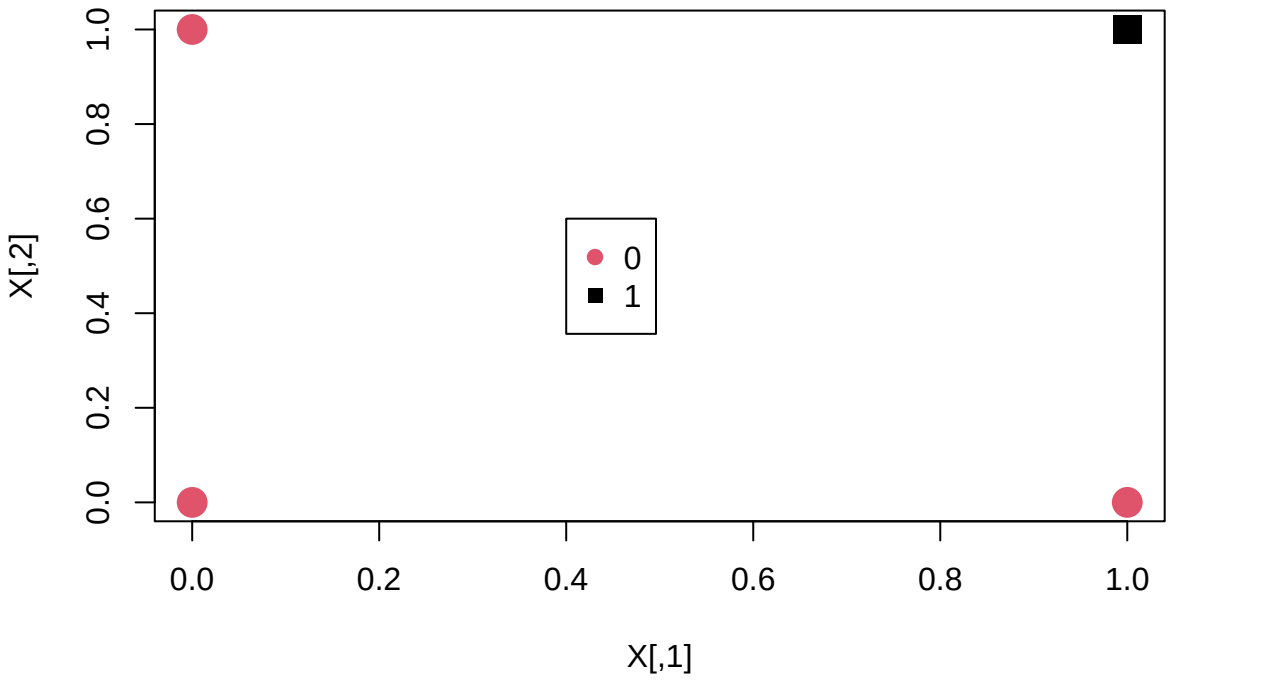
Como mencionado antes, a saída desejada deste dataset só será 1, quando os atributos descritivos (exceto o bias) também for 1. Portanto, a saída desejada deste conjunto é definida como:

```
Y<-c(0,0,0,1)
print(Y)
```

```
## [1] 0 0 0 1
```

Visualizando o dataset para o entendimento de algumas caracerísticas dos dados:

```
plot(X,type="n")
points(x1[1],x1[2],pch=c(19),cex=2,col="2")
points(x2[1],x2[2],pch=c(19),cex=2,col="2")
points(x3[1],x3[2],pch=c(19),cex=2,col="2")
points(x4[1],x4[2],pch=c(15),cex=2,col="1")
legend(0.4,0.6,legend=c("0","1"),col=c(2,1), pch=c(19,15))
```



Parametrização da Rede

Para esa rede neural é preciso inicializar os valores dos pesos e definir um valor para a taxa de aprendizagem. Os pesos serão inicializados com valor 0, exceto para o bias.

```
W<-c(0,0,1)
```

O valor de eta será ajustado como sendo 0.1:

```
eta<-0.1
```

Por fim, precisa ser definido o número máximo épocas como critério de parada:

```
epoca_max<-20
```

Apenas como forma de monitorar o treinamento da rede, mais especificamente o erro de aprendizagem, durante o treinamento será monitorado o erro a cada iteração (apresentação de entrada do conjunto de treinamento) e, por isso, criaremos uma variável para armazenar essa medida:

```
erro_ite<-rep(0,dim(X)[1])
```

E outra variável para guardar o erro total por época de aprendizagem (uma rodada completa de todo conjunto de treinamento):

```
erro_total<-rep(0,epoca_max)
```

Treinamento da Rede

Agora, finalmente, reliza-se o treinamento do neurônio

```
for(epoca in 1:epoca_max){
  for(i in 1:dim(X)[1]) {
   v<-sum(X[i,]*W)
   if(v>0){
     y_calc<-1
   }else{
     y_calc<-0
   }
   erro<-Y[i]-y_calc
   delta<-(eta*erro*X[i,])
   W<-W+delta
   erro_ite[i]<-erro^2
  }
  erro_total[epoca]<-sum(erro_ite)
  if(sum(erro_ite)==0){
   break
  }
}
```

Análise do desempenho da rede

É interessante, em nível de informação, saber o resultado de alguns parâmetros do treinamento da rede, por exemplo:

o número de épocas efetivamente usado no treinamento:

```
print(paste("Número de épocas usadas no treinamento",epoca,"de um máximo de",epoca_max))
```

```
## [1] "Número de épocas usadas no treinamento 12 de um máximo de 20"
```

O valor final do peso sináptico:

```
print(W)
```

```
## [1]  0.2  0.1 -0.3
```

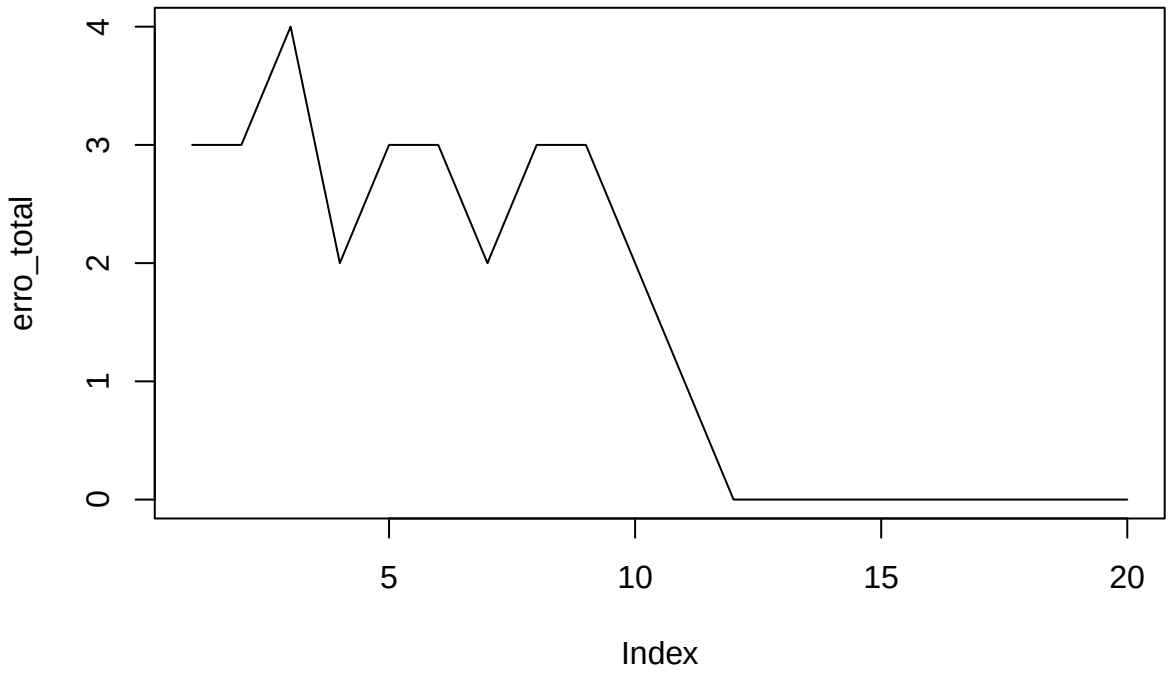
E o erro total da rede:

```
print(erro_total)
```

```
##  [1] 3 3 4 2 3 3 2 3 3 2 1 0 0 0 0 0 0 0 0 0
```

ou de forma gráfica

```
plot(erro_total,type="l")
```



Para este primeiro caso, a análise do desempenho da rede será feita de forma geométrica com o objetivo de verificar o resultados final do treinamento.

Os valores finais (após treinamento) do peso indica os parâmetros do hiperplano de separação entre as classes do conjunto de dados.

```
print(W)
```

```
## [1]  0.2  0.1 -0.3
```

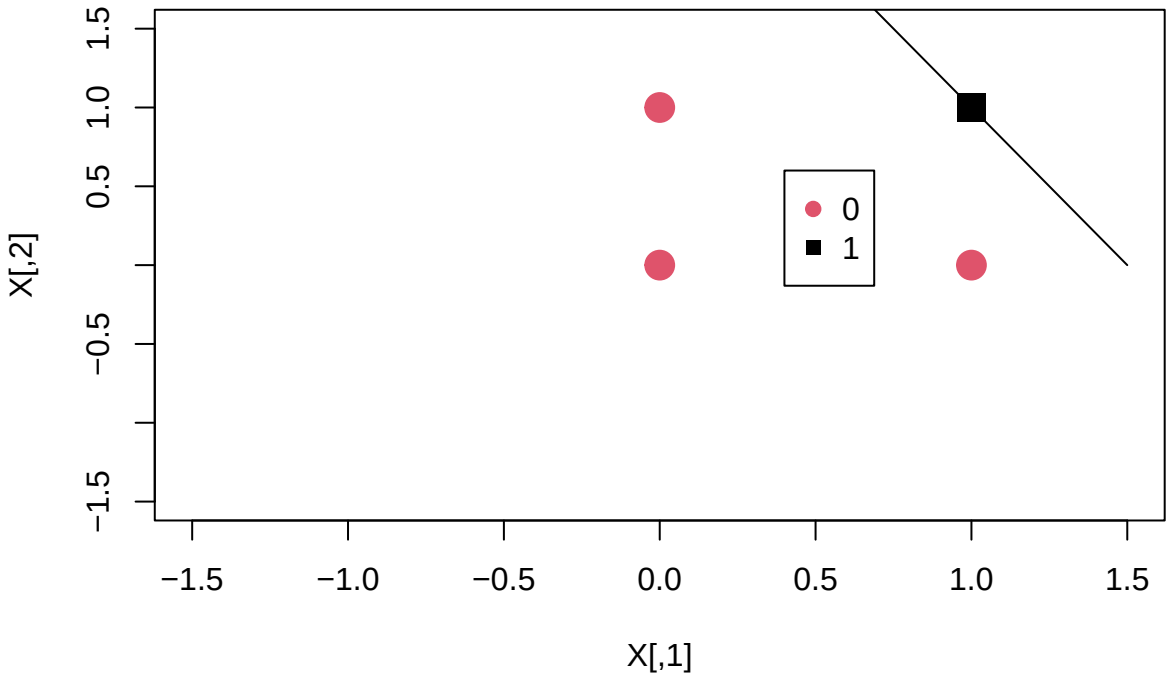
Que traduzidos graficamente, teremos então o conjunto de treinamento e os pesos sinápticos traduzidos em hiperplano de separação.

```
plot(X,type="n",xlim=c(-1.5,1.5 ), ylim=c(-1.5,1.5))
points(x1[1],x1[2],pch=c(19),cex=2,col="2")
points(x2[1],x2[2],pch=c(19),cex=2,col="2")
points(x3[1],x3[2],pch=c(19),cex=2,col="2")
points(x4[1],x4[2],pch=c(15),cex=2,col="1")
legend(0.4,0.6,legend=c("0","1"),col=c(2,1), pch=c(19,15))

x_1<- -(W[3]/W[1])

x_2<- -(W[3]/W[2])

segments(x_1,0,0,x_2)
```



O resultado esperado para o gráfico acima é ter, além do conjunto de treinamento, um segmento de reta, representando um hiperplano de separação. Este hiperplano deve separar o conjunto de treinamento em suas classes. Caso o hiperplano não separe as classes corretamente, dizemos que há um erro de classificação. Contudo, a visualização permite apenas uma interpretação qualitativa e limita-se a conjunto de dados com até 3 atributos (variáveis), o que não é comum em casos reais. Portanto, a seguir é proposto um novo desafio ainda com análise de desempenho qualitativo e, nas próximas atividades será abordado a análise quantitativa do desempenho da Rede Neural.

Exercício

Case 1: Dataset AND

Para o experimento apresentado anteriormente, faça as alterações de valores dos seguintes parâmetros e avalie os resultados de época e hiperplano de separação:

a) Altere os valores de pesos iniciais para

```
W<-c(0.1,0.1,1)
```

b) Com os valores de pesos iniciais do primeiro experimento (apresentado anteriormente durante a aula), diminua o valor da taxa de aprendizagem eta para:

```
eta<-0.01
```

c) Com o valor do eta reduzido, altere o valor do peso conforme sugestão abaixo e verifique os medidores de desempenho:

```
W<-c(0.1,0,1)
```

Case 2: Dataset Iris

O dataset Iris trata-se de uma base onde através das medidas sob as sépalas e pétalas da planta Iris seja capaz de construir uma modelagem preditiva capar de reconhecer automaticamente as espécies da planta em: Setosa, Versicolor ou Virginica.

Para este case, simplificaremos o problema estudando apenas duas espécies da planta, Setosa e Virginica. E também, deve-se utilizar aqui apenas as medidas de sépalas (sepal).

Dica, como o dataset está disponível no R, a carga é feita simplesmeste com o comando:

```
data("iris")
head(iris)
```

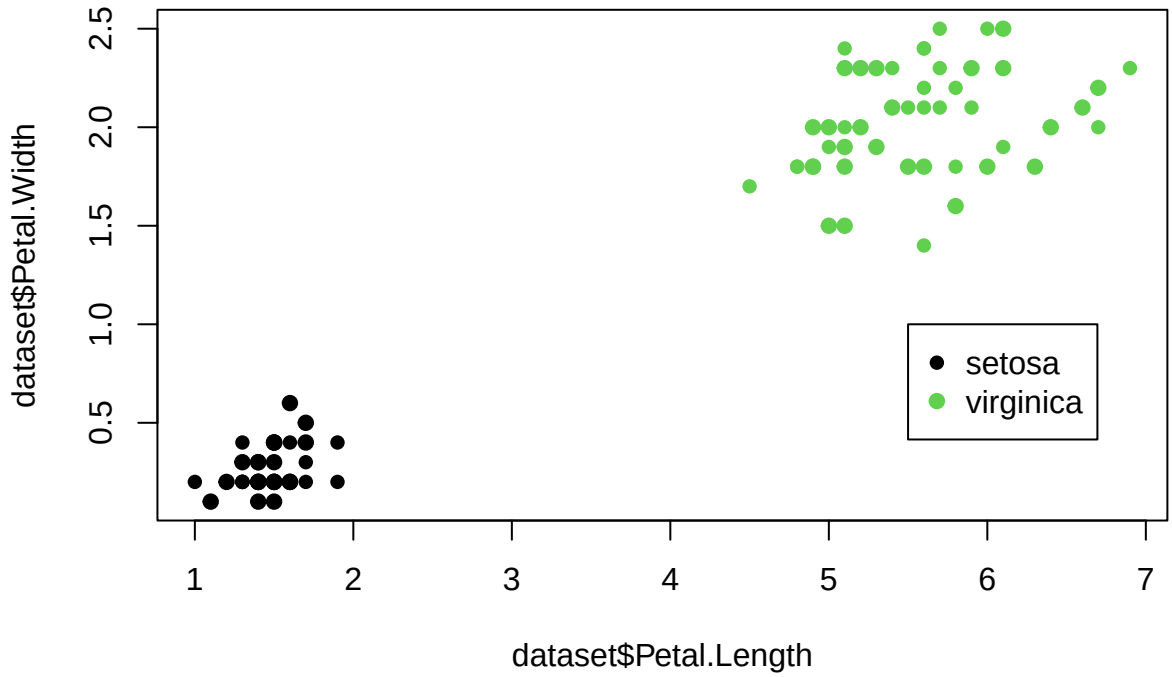
```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

A separação e visualização do dataset a ser utilizado no experimento (Sepal.Lenght, Sepa.Width, Species={setosa e virginica}) são apresentados a seguir:

```
dataset<-iris[which(iris$Species!="versicolor"),][1:2-3]
head(dataset)
```

```
##   Petal.Length Petal.Width Species
## 1          1.4         0.2  setosa
## 2          1.4         0.2  setosa
## 3          1.3         0.2  setosa
## 4          1.5         0.2  setosa
## 5          1.4         0.2  setosa
## 6          1.7         0.4  setosa
```

```
plot(dataset$Petal.Length,dataset$Petal.Width,pch=c(16,19),col=dataset$Species)
legend(5.5,1,legend=c("setosa","virginica"),col=c(1,3),pch=c(16,19))
```



Importante notar que, a rede neural só trabalha com valores numéricos. O atributo classificatório do dataset Iris, Espécie, é do tipo categórico. Portanto, será necessário realizar a transformação no atributo para numérico e binário (0 e 1). Uma sugestão de transformação pode ser feita como segue usando a função mapvalues do pacote plyr:

```r
library("plyr")
classes<-as.numeric(dataset$Species)
classes_bin<-mapvalues(classes,from=c(1,3),to=c(0,1))
print((classes_bin))
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Como dica final, sugere-se que normalize os valores dos atributos (variáveis)

```r
x1<-dataset[,1]/max(dataset[,1])
x2<-dataset[,2]/max(dataset[,2])
```

Portanto, pede-se:

a) Escolher os melhores parâmetros para o treinamento da rede;
b) Apresentar os resultados finais de ajustes de parâmetros do treinamento da rede;
c) Apresentação dos resultados de forma geométrica (visualização dos dados com a fronteira de separação sobreposta).
d) Refaça todos os itens acima (a-c) agora, considerando, apenas as classes versicolor e virginica.