Exercício 1 - Aplicação perceptron simples

Rodrigo Machado Fonseca - 2017002253

October 23, 2021

1 Introdução

Neste trabalho nós utilizaremos a o algoritmo Perceptron Simples para classificar se um tumor é benigno ou maligno de acordo com a base de dados *Breast Cancer*[1].

2 Perceptron Simples

A rede perceptron classifica baseado na função limiar que consiste na seguinte função:

$$\begin{cases} w1 * x1 + w2 * x2 \geq \theta, & 1 \\ w1 * x1 + w2 * x2\theta, & 0 \end{cases} \tag{1}$$

$$\begin{cases} w1, w2 & pesos \\ x1, x2 & coordenadas\ \ de\ \ x \\ \theta & limiar\ \ de\ \ separacao \end{cases}$$

Os parâmetros que estamos interessados em descobrir são os pesos e o valor de θ. Mas note que da forma que a equação 1 está disposta fica impossível utilizar o percpetron, pois após a multiplicação da entrada pelos pesos iremos comparar com o limiar, mas note que também não sabemos o limiar. Portanto, iremos fazer um pequeno ajuste na equação.

$$\begin{cases} w1 * x1 + w2 * x2 - \theta \geq 0, & 1 \\ w1 * x1 + w2 * x2 - \theta < 0, & 0 \end{cases} \tag{2}$$

Note que esta pequena alteração nos permite descobrir treinar a função de modo a descobrir os pesos e o limiar. Porque, basta colocarmos uma entrada padrão -1 que será multiplica por um valor aleatório de θ e agora valor de comparação da função degrau será centrado no zero. Portanto, o θ será atualizado assim como os pesos e no final a nossa função irá retornar $\theta, w1, w2$, respectivamente.

Portanto, agora iremos preparar os dados, implementar a função *train_perceptron* e a função *calculate_perceptron_output*. Porém devo ressaltar que utilizaremos um número de peso arbitrário de modo que o algoritmo torne-se geral e possa ser aplicado independente da dimensão da entrada.

```
> train_perceptron <- function(xin,yd,eta,tol,maxepocas,par){
+
+   ## Definition of  functions parameters
+
+   #yd: output
+   #xin: input
+   #eta: learning rate
+   #tol: maximum error tolerance
+   #maxepoch -> maximum number of iterations
+   #par: par=1, if p = 1, enter the term polarization
+
+   ## Dimensions of input matrix
+   n_row <-dim(xin)[1]
+   n_col <-dim(xin)[2]
+
+   ## term polarization and weights
+
+   if (par==1){
+
+     # initialize weight matrix
+     wt<-as.matrix(runif(n_col+1)-0.5)
+     xin<-cbind(-1,xin)
+   }
+   else{
+
+     # initialize weight matrix
+     wt<-as.matrix(runif(n_col)-0.5)
```

```
+   }
+
+   ## control variables
+
+   # vector error per epoch
+   evec <- matrix(ncol = 1, nrow = 1)
+
+   # variblas comparative
+   nepoch <-1
+   error_epoch <-tol+1
+
+
+   while ((nepoch < maxepocas) && (error_epoch > tol))
+   {
+     # sum square error
+     ei2<-0
+
+     # random string
+     xseq<-sample(n_row, replace = FALSE)
+
+     for (i in 1:n_row)
+     {
+
+       # random sequence
+       irand<-xseq[i]
+
+       # output
+       yhati<- as.double((xin[irand, ] %*% wt) >= 0)
+       # error
+       ei<-yd[irand]-yhati
+
+       # delta
+       dw<-eta * ei * xin[irand,]
+
+       #Update wights
+       wt<-wt + dw
+
+       #erro acumulado
```

```
+       ei2<-ei2+ei*ei
+     }
+
+     #mean square error (MSR)
+     evec <- rbind(evec, as.matrix(ei2/n_row, ncol = 1, nrow= 1))
+
+     #save error per epoch
+     error_epoch <- ei2/n_row
+
+     # Update number of epochs
+     nepoch <-nepoch +1
+   }
+
+   r_list <-list(wt, evec, nepoch)
+   return(r_list)
+
+ }

>   calculate_perceptron_output <- function(xvec,w,par){
+   if(par==1){
+       xvec<-cbind(-1,xvec)}
+
+   u<-xvec %*% w
+   y<-1.0*(u>=0)
+   return ((as.matrix(y)))
+ }
```

3 Preparação dos Dados

Na base *Breast Cancer*, estão disponíveis nesta base 699 amostras, das quais 458 (65.5%) correspondem a tumores benignos e 241 (34.5%) a tumores malignos.

Para utilizarmos esses dados vamos tratá-los de modo que torne-se possível a aplicação do Perceptron Simples. A priori, iremos remover os id's, pois eles não tem nenhuma correlação com o tipo de câncer. Em seguida, iremos excluir o valores NA's (não disponível). Por fim, iremos transformar benigno para 1 e maligno para 0, para ficar coerente a função degrau.

Coluna	Atributo	Domínio
1.	Sample code number	id number
2.	Clump Thickness	1 - 10
3.	Uniformity of Cell Size	1 - 10
4.	Uniformity of Cell Shape	1 - 10
5.	Marginal Adhesion	1 - 10
6.	Single Epithelial Cell Size	1 - 10
7.	Bare Nuclei	1 - 10
8.	Bland Chromatin	1 - 10
9.	Normal Nucleoli	1 - 10
10.	Mitoses	1 - 10
11.	Class:	(2 for benign, 4 for malignant)

Table 1: Atributos da base de dados Breast Cancer

```
> # Carregar os dados
> data <- as.matrix(read.table("breast-cancer-wisconsin.data", sep = ','))
> # Remover coluna de ids
> data <- data[,-1]
> # Converter para matrix numerica
> data <- apply(data, c(1,2), as.numeric)
> # Remover linhas com NAs
> rows_to_remove <- apply(data, 1, function(x){any(is.na(x))})
> data <- data[!rows_to_remove,]
> # Obter amostras x e y
> x <- data[,1:9]
> y <- data[,10]
> # Rotular as amostras das Classes com o valor de 0 (maligno) e 1 (benigno)
> y[y == 4] <- 0
> y[y == 2] <- 1
```

4 Treinamento

Para fazer o treinamento iremos separar o conjunto de dados entre treinamento e teste. O conjunto de treinamento será utilizado para encontrar os pesos da nossa rede neural. O conjunto de teste será utilizado para analisar a capacidade do nosso classificador distinguir entre benigno e maligno.

Esses dados devem ser escolhidos de forma aleatório de modo a não enviesar o estudo.

```
> library('RSNNS')
> train_routine <- function(x, y, eta, tol, maxepochs, par, number_repeat){
+   #y: Output variable array.
+   #x: Input variable array.
+   #eta: Learning rate.
+   #tol: Maximum error tolerance.
+   #maxepochs -> Maximum number of iterations per epoch.
+   #par: par=1, if p = 1, enter the term polarization
+   #number_repeat: Number of analysis repetitions.
+
+ accuracy <- matrix(nrow = number_repeat, ncol = 1)
+ for (i in 1:number_repeat)
+ {
+   # Select test and training samples
+   xy_all <- splitForTrainingAndTest(x,y,ratio = 0.3)
+   x_train <- xy_all$inputsTrain
+   y_train <- xy_all$targetsTrain
+   x_test <- xy_all$inputsTest
+   y_test <- xy_all$targetsTest
+
+   # Training
+   retlist <- train_perceptron(x_train, y_train, eta, tol, maxepochs, par)
+
+   # Testing
+   wt <- retlist[[1]]
+   y_hat <- calculate_perceptron_output(x_test, wt, par)
+   accuracy[i,] <- 1 - (t(y_test - y_hat) %*% (y_test - y_hat))/length(y_test)
+ }
+
+ return(accuracy)
+ }
```

5 Resultados

Agora que já possuíamos nossa rotina de trinamento podemos fazer o experimento. Iremos separa 70 % para treino e 30 % para teste. Iremos repetir o experimento 20 vezes e após isso calcular a média da acurácia e o seu desvio padrão.

```
> accuracy <- train_routine(x, y, 0.1, 0.0001, 1000, 1, 20)
> print("Acurácia Média")

[1] "Acurácia Média"

> mean(accuracy)

[1] 0.977561

> print("Desvio Padrão")

[1] "Desvio Padrão"

> sd(accuracy)

[1] 0.01513064
```

6 Discussão

Com esse experimento, pode-se concluir que foi possível construir uma rede Perceptron Simples capaz de classificar corretamente as amostras mais de 95% das vezes. Esse valor é bastante satisfatório, o que indica que o algoritmo de treinamento foi implementado corretamente e que uma rede Perceptron Simples é capaz de resolver o problema da base de dados *Breast Cancer*. Além disso, ao escolhermos o conjunto de treino e teste de forma aleatório, também atualizando os pesos de forma aleatória garantimos a replicabilidade deste experimento.

References

[1] https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29