Universidade Federal de Minas Gerais

Escola de Engenharia

Reconhecimento de Padrões - 2021/1 - ERE

ELT135

Exercício 1 - Treinamento Perceptron

Autores:
Vítor Gabriel Reis Caitité

Email:
vcaitite@ufmg.br

30 de maio de 2021

Sumário

Lista de Figuras

1 Introdução

1.1 Objetivo

O objetivo desse exercício é aplicar o modelo "preceptron simples"estudado na disciplina a um problema prático referente a um conjunto de dados disponibilizado publicamente e que envolve o reconhecimento do tipo de tumor de mama (maligno ou benigno). O *database* foi obtido da University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg [2].

1.2 Dados

Basicamente o arquivo contendo os dados contêm 1 coluna de Id (apenas um identificador) seguida de 9 colunas contendo os vetores de entrada x, e por fim, uma última coluna contendo a classificação do tumor correspondente ao rótulo y daquela amostra. Foram disponibilizados 699 dados.

```
 #  Attribute                     Domain
-- ----------------------------------------
 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)
```

Figura 1: Organização dos dados.

1.3 Perceptron Simples

Como se sabe, o perceptron simples pode ser utilizado para dividir duas classes linearmente separáveis. Ou seja, o modelo de classificador desenvolvido com esse perceptron de camada única (cuja a rede está mostrada na Figura 2) é na verdade um classificador linear. Dependendo da dimensão o problema esse classificador representa uma reta, um plano ou um hiperplano no espaço de entrada. Como estamos lidando com um problema com 9 variáveis de entrada, então o classificador gerado a partir do perceptron simples representará um hiperplano nesse espaço de entrada, tentando separar os dados linearmente [1].

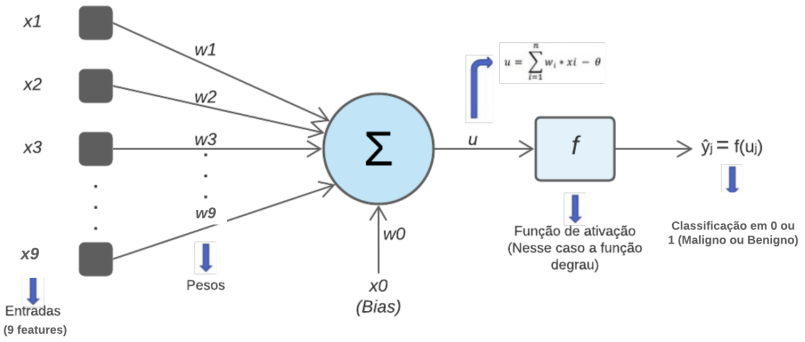

Figura 2: Rede Perceptron Simples. Fonte: Imagem produzida pelo autor

O algoritmo de treinamento do perceptron simples apresentado durante a disciplina baseia-se na aplicação da Equação 1 sobre os dados de treinamento até que o erro global atinja um critério de parada ou complete-se o número máximo de iterações passado como parâmetro.

$$w(t+1) = w(t) + \eta * e(t) * x(t) \tag{1}$$

onde:

- w(t) - valores d vetor de pesos no instante t;
- e(t) - valor do erro no instante t;
- x(t) - vetor de entradas no instante t;
- η - passo de treinamento.

2 Desenvolvimento

Inicialmente, buscou-se entender mais sobre os dados do problema e por se tratar de uma pequena quantidade de dados decidiu - se plotar gráficos das amostras usando apenas 2 dimensões de cada vez. Isso foi feito somente no intuito de tentar visualizar uma possível separação espacial. Esses gráficos podem ser vistos na Figura 3.

OBS: O problema será resolvido utilizando todas as 9 dimensões, nesse ponto buscou-se apenas investigar e mostrar uma possível separação espacial.

Como pode ser observado nos gráficos, no geral os dados apresentam uma boa separação espacial, o que inicialmente pode nos indicar que o modelo perceptron simples será capaz de realizar uma boa classificação.

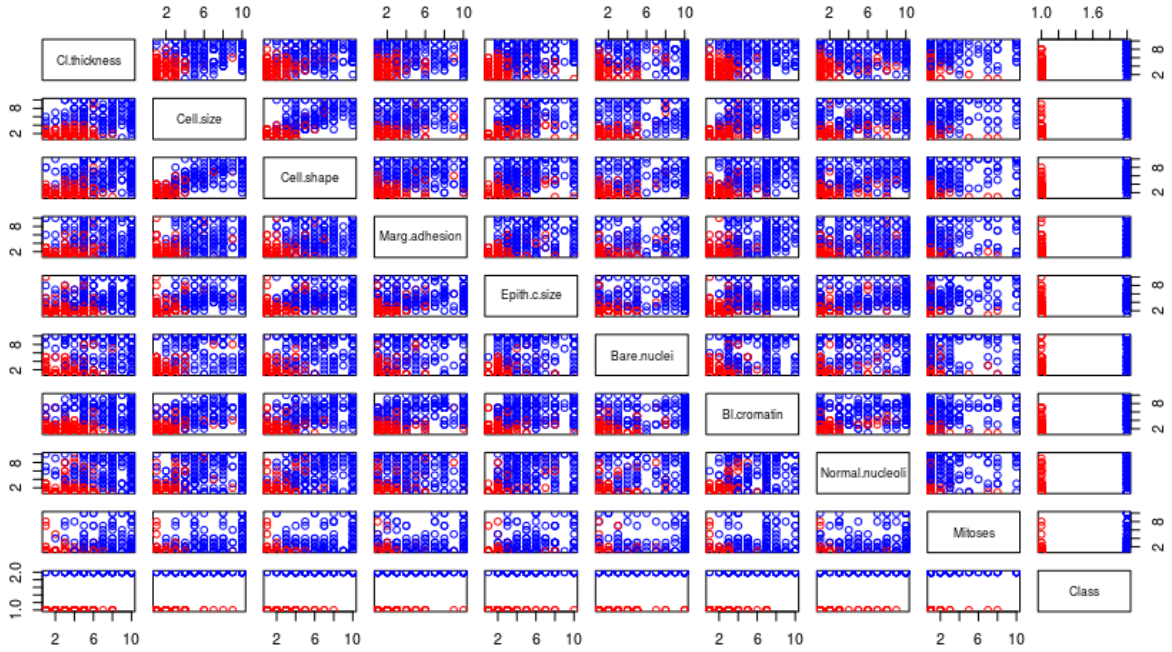

Figura 3: Amostras das 2 classes do conjunto de dados para todos os pares de variáveis de entrada. Fonte: Imagem produzida pelo autor utilizando a função de plot do R.

2.1 Tratamento Inicial

O primeiro passo realizado foi carregar os dados e armazená-los. Estes dados receberão um tratamento inicial para eliminação dos dados faltantes, representados pelo string NA. Além disso, rotulou-se as amostras com valor de 0 (malígno) e 1 (benígno).

```
rm(list=ls())
source("~/Documents/list_1/trainPerceptron.R")
source("~/Documents/list_1/yperceptron.R")
library(caret)

# Carregando base de dados:
```

```
path <- file.path("~/Documents/UFMG/10/Reconhecimento de padr es/list/
    list_1/database", "BreastCancer.csv")
data <- read.csv(path)

# Rotular as amostras das Classes com o valor de 0 (maligno) e 1 (
    benigno).
data$Class <- ifelse(data$Class=="malignant", 0, 1)

# Tratamento de dados faltantes - Remove-se linhas contendo NA
data <- data[complete.cases(data),]
```

Listing 1: Carregamento e tratamento dos dados

2.2 Treinamento e Teste

Nessa etapa foi treinado um perceptron simples, utilizando a rotina de treinamento de perceptron mostrada no início desse documento e a base de dados do item anterior, para classificar um tumor em maligno ou benigno. Foram utilizados 70% dos dados para treino e 30% para teste. Além disso, foram realizados 20 execuções diferentes de treinamento e teste, e os valores de acurácia (de treinamento e teste), foram apresentados na forma de média ± desvio_padrão. Os resultados e o script desenvolvido podem ser vistos abaixo.

```
# Removendo coluna ID
data <- data[,2:11]

# Realiza pelo 20 execucoes diferentes
accuracy_train <- rep(0, 20)
accuracy_test <- rep(0, 20)
for(execution in 1:20){
  # Selecionar aleatoriamente 70% das amostras para o conjunto de
      treinamento e 30% para o conjunto de teste
  partition <- createDataPartition(1:nrow(data),p=.7)
  train <- as.matrix(data[partition$Resample1,])
  test <- as.matrix(data[- partition$Resample1,])
  x_train <- as.matrix(train[, 1:(ncol(train)-1)])
  y_train <- as.matrix(train[, ncol(train)])
  x_test <- as.matrix(test[, 1:(ncol(train)-1)])
  y_test <- as.matrix(test[, ncol(train)])

  # Treinando modelo:
  retlist <- trainPerceptron(x_train, y_train, 0.1, 0.01, 1000, 1)
  W<- retlist[[1]]

  # Calculando acuracia de treinamento
```

```
22    y_hat_train <- as.matrix(yperceptron(x_train, W, 1), nrow = length_
          train, ncol = 1)
23    accuracy_train[execution]<-1-((t(y_hat_train-y_train) %*% (y_hat_
          train-y_train))/length(y_train))
24
25    # Calculando acuracia de Teste:
26    y_hat_test <- as.matrix(yperceptron(x_test, W, 1), nrow = length_test
          , ncol = 1)
27    accuracy_test[execution]<-1-((t(y_hat_test-y_test) %*% (y_hat_test-y_
          test))/length(y_test))
28  }
29
30  # Media das acuracias
31  mean_accuracy_train <- mean(accuracy_train) * 100
32  mean_accuracy_test <- mean(accuracy_test) * 100
33
34  # Desvio Padrao das acuracias
35  sd_accuracy_train <- sd(accuracy_train) * 100
36  sd_accuracy_test <- sd(accuracy_test) * 100
37
38  # Printing Result
39  print(paste("Acuracia media de treinamento do modelo: ", mean_accuracy_
        train, "%", "  ", sd_accuracy_train, "%"))
40  print(paste("Acuracia media de teste do modelo: ", mean_accuracy_test,
        "%", "  ", sd_accuracy_test, "%"))
```

Listing 2: Treinamento e aplicação do perceptron simples

2.3 Resultados

O resultado encontrado ao se rodar o algoritmo acima está mostrado na Figura 4 abaixo.

```
> source('~/Documents/UFMG/10/Reconhecimento de padrões/list/list_1/perceptron_application.R')
[1] "Acurácia media de treinamento do modelo:  97.1398747390397 % ± 0.663994340597906 %"
[1] "Acurácia media de teste do modelo:  96.4460784313726 % ± 1.3761838204863 %"
```

Figura 4: Resultado encontrado.

Como pode-se notar obteve-se uma acurácia de treinamento e teste de aproximadamente 97,1% e 96,4%. Essa acurácia elevada se deve principalmente a boa separação espacial dos dados, que possibilita que o classificador gerado a partir do perceptron simples, gere um hiperplano no espaço de entrada dos dados e seja capaz de separar linearmente as duas classes de dados.

3 Anexo - Funções Utilizadas

3.1 Treinamento Perceptron Simples

```
trainPerceptron <- 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 ) >= 0 )
      ei<-yd[irand]- yhati
      dw<-as.vector(eta) * as.vector(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)
}
```

Listing 3: Função de treinamento de um perceptron simples em R

3.2 Resposta do Perceptron Simples

```
yperceptron <- function(xvec, w, par){
  # xvec: vetor de entrada
  # w: vetor de pesos
```

```
4    # par: se adiciona ou nao o vetor de 1s na entrada
5    # yperceptron: resposta do perceptron
6    if ( par==1){
7      xvec<-cbind ( 1 , xvec )
8    }
9    u <- xvec %*% w
10   y <- 1.0 * (u>=0)
11   return(as.matrix(y))
12 }
```

Listing 4: Função que calcula a resposta de um perceptron simples em R

Referências

[1] Prof. Antônio de Pádua Braga. Aprendendo com exemplos: Princıpios de redes neurais artificiais e de reconhecimento de padrões. Notas de aula de disciplina.

[2] O. L. Mangasarian and W. H. Wolberg. "cancer diagnosis via linear programming". SIAM News, Volume 23, Number 5, September 1990. pp 1 18.