UNIVERSIDADE FEDERAL DE MINAS GERAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Exercício Computacional 1: Perceptron

Aluno: *Pedro Vinícius A. B. de Venâncio*

Disciplina: *Redes Neurais Artificiais (EEE950)* – Professor: *Antônio de Pádua Braga*
Data: *Setembro de 2019*

Introdução

Este relatório tem como objetivo desenvolver[1] o modelo *perceptron*, tendo em vista sua importância histórica e suas diversas aplicações.

Perceptron

Desenvolvido por Frank Rosenblatt e inspirado em trabalhos anteriores de Warren McCulloch e Walter Pitts, o *perceptron* é um neurônio artificial que mapeia um vetor de entrada $x \in \mathbb{R}$ em uma única saída binária $f(x)$. O mapeamento é realizado através de uma regra bastante simples (1):

$$\hat{y} = f(x) = \begin{cases} 1, \text{ se } w \cdot x + b \geq threshold \\ 0, \text{ caso contrário} \end{cases} \tag{1}$$

onde w é um vetor de pesos real que reflete a importância de cada entrada, b é uma constante denominada viés e *threshold* é um limiar para classificação.

```
# Activation function
activation <- function(X, W, limiar = 0.5){
  z = t(X %*% t(W))
  # Heaviside function
  Y_hat = sapply(z > limiar, as.numeric)
  return(Y_hat)
}
```

Em termos gerais, o *perceptron* pode ser visto como o tipo mais simples de rede neural do tipo *feedforward*, isto é, que possui fluxo contínuo de operações da entrada pra camada seguinte. Durante o treinamento, a regra de atualização do vetor de pesos w na época atual t é dada por (2):

$$w(t+1) = w(t) + \eta e(t) x(t) \tag{2}$$

onde $e(t)$ é a diferença entre a saída esperada e a saída prevista pelo modelo $(y - \hat{y})$ no instante t e η é uma taxa de aprendizagem na direção minimizante desse erro.

```
# Perceptron algorithm
perceptron <- function(X, Y, eta = 0.01, max_epochs = 100){
```

[1] Todas as implementações foram feitas em linguagem R.

```
  # Number of features
  n_features <- dim(X)[2]
  # Number of epochs
  epoch <- 1

  # Random initial weights
  W <- t(rnorm(n_features))
  # Predictions with initial weights
  Y_hat <- activation(X, W)

  while(sum(Y_hat - Y) != 0 || epoch < max_epochs){

    # Weights updates
    W <- W + eta*((Y - Y_hat) %*% X)
    # Predictions with current weights
    Y_hat <- activation(X, W)
    # Update number of epochs
    epoch <- epoch + 1

  }
  return(W)
}
```

Além disso, sua equação de decisão $\sum_j w_j x_j = b$ corresponde à equação geral de um hiperplano, o que lhe permite uma notável capacidade de separação linear [1]. No espaço $\mathbb{R}^2$, por exemplo, a equação de separação se torna $w_1 x_1 + w_2 x_2 = b$, ou seja, uma reta em sua equação reduzida $y = mx + n$, onde n é o coeficiente linear dado por $n = -\frac{b}{w_2}$ e m é o coeficiente angular $m = -\frac{w_1}{w_2}$.

```
# Plot decision boundary
decision_boundary <- function(X, W){
  # Linear coefficient
  intercept <- -W[1]/W[3]
  # Angular coefficient
  slope <- -W[2]/W[3]
  # Equation of a line
  classifier <- slope*sort(X[,2]) + intercept
  lines(sort(X[,2]), classifier)
}
```

Problema de Classificação

Um problema de classificação consiste em atribuir uma classe a cada valor de entrada x, tendo, portanto, sua saída $f(x)$ definida por uma quantidade limitada de valores discretos. Em outras palavras, estamos tentando mapear variáveis de entrada em categorias.

O modelo *perceptron* é capaz de resolver alguns desses problemas, tendo como premissa para tal a separabilidade linear dos dados. Dessa forma, um conjunto de 300 amostras foi gerado para visualizar seu funcionamento (Figura 1). Os dados podem ser divididos em dois grupos distintos:

1. 150 amostras com média $\mu_1 = 500$ e desvio padrão $\sigma_1 = 120$ no eixo das abscissas.
2. 150 amostras com média $\mu_2 = 600$ e desvio padrão $\sigma_2 = 120$ no eixo das ordenadas.

```
# Synthetic linearly separable data points

# Interesting case studies: set seed as 3, 7, 10, 19
set.seed(3)
# Number of samples
n_samples <- 150
x <- seq(0, 2*n_samples, length = n_samples)

# Samples generated by a normal distribution
x1 <- rnorm(n_samples, mean = 650, sd = 120)
x2 <- rnorm(n_samples, mean = 600, sd = 120)

# Scatter plot
plot(x, x2, xlab = TeX('x_1'), ylab = TeX('x_2'), col = 'blue')
points(x1, x, col = 'red')
```

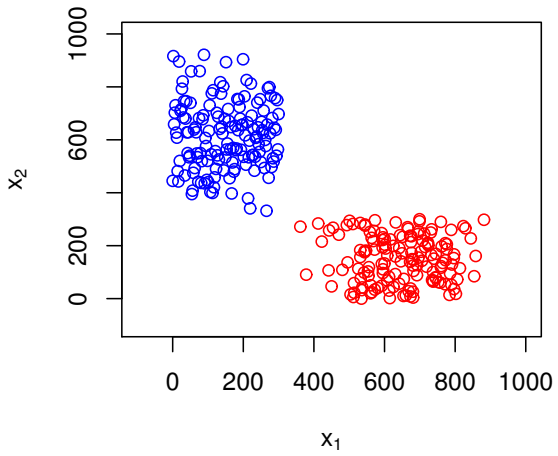

Figura 1: Base de dados sintéticos e linearmente separável.

Os parâmetros utilizados para o treinamento do modelo foram $\eta = 0.01$ e $max\_epochs = 10$. No entanto, foram necessárias apenas 2 épocas para o algoritmo atingir um erro absoluto igual a zero. Através da superfície de separação gerada, corrobora-se que o *perceptron* foi capaz de separar os dados em dois grupos visualmente distintos (Figura 2).

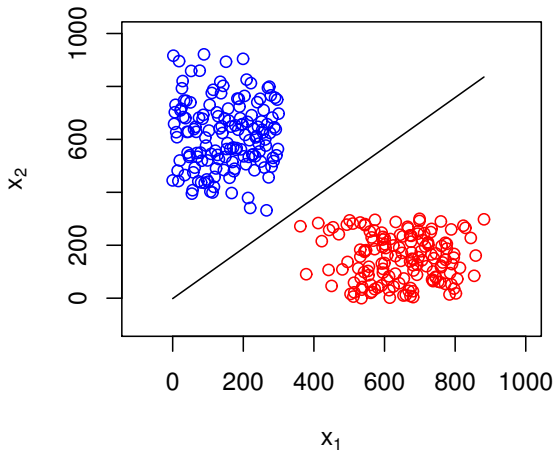

Figura 2: Superfície de separação gerada pelo *perceptron*.

Com o intuito de avaliar o desempenho do modelo também para dados reais, o *dataset Wisconsin Breast Cancer* foi considerado como *benchmark*. Ele compreende 9 *features* calculadas a partir de imagens dos núcleos celulares presentes na mama de 683 pacientes. O objetivo é utilizar o *perceptron* para classificar os casos dos pacientes em "M"(*malignant*) e "B"(*benign*).

Para tal fim, os dados foram divididos em duas partições aleatórias: conjunto de treinamento e conjunto de teste, sendo o primeiro responsável por conter 70% das amostras e o segundo por conter os 30% restantes. O processo de treinamento foi realizado com um taxa de aprendizagem $\eta = 0.01$ e um número máximo de épocas $max\_epochs = 100$.

```
# Splitting data for training and test
proportion <- 0.7
set.seed(10)
perm <- sample(dim(X)[1])
train_index <- perm[1:round(proportion*n_samples)]
test_index <- perm[round(proportion*n_samples + 1):n_samples]
X_train <- X[train_index,]
Y_train <- Y[train_index]
X_test <- X[test_index,]
Y_test <- Y[test_index]
```

Uma vez que os dados estão no plano $\mathbb{R}^9$, a visualização da superfície de separação se torna inviável para avaliar o desempenho do modelo. Assim, métricas de avaliação numéricas, como sensibilidade, especificidade e erro devem ser consideradas (Tabela 1).

threshold	Sensibilidade	Especificidade	Erro Quadrático Médio
0.5	0.9367	0.9761	3.9×10^{-2}

Tabela 1: Métricas de avaliação de desempenho.

A matriz de confusão também é uma boa alternativa para casos com dimensões maiores, apresentando os números de falsos positivos, falsos negativos, verdadeiros positivos e verdadeiros negativos (Tabela 2).

		Actual	
		M	B
Predicted	M	123	5
	B	3	74

Tabela 2: Matriz de confusão.

Diante dos valores apresentados, pode-se afirmar que o desempenho do *perceptron* para esse problema foi bastante satisfatório. Tanto o percentual de classificações corretas para os que possuem o tumor maligno (especificidade) quanto para aqueles que possuem o tumor benigno (sensibilidade) foram acima de 93%. Além disso, o erro quadrático médio calculado foi pequeno, visto que apenas 8 pacientes de 205 (conjunto de teste) foram classificados incorretamente.

Conclusão

Apesar de hoje ser mais comum utilizar outros modelos de redes neurais mais complexos, o *perceptron* permite uma compreensão clara do funcionamento de uma rede neural em termos matemáticos para problemas de classificação, sendo uma excelente introdução.

Referências

[1] A. d. P. Braga, *Redes neurais artificiais: teoria e aplicações*. Livros Técnicos e Científicos, 2000.