Resumo sobre Perceptron e Redes Neurais

João Pedro Ferreira da Cruz[1]

[1]Departamento de Computação – Universidade Federal de Sergipe (UFS)

pedro.dacruz@dcomp.ufs.br,jpfdc1120@academico.ufs.br

1. Introdução

Redes neurais artificiais são estruturas da computação que foram feitas com base na junção de vários neurônios. Dessa forma, cada neurônio dessa rede possui semelhança com neurônios humanos. Na figura 1 têm-se exemplo simples que exemplifica as camadas de uma rede neural artificial.

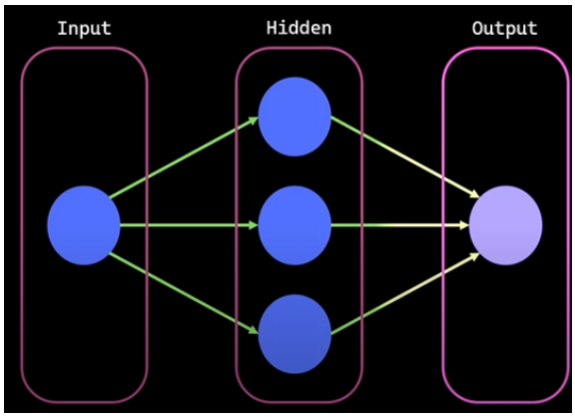

Figura 1. Estrutura de básica de uma rede neural

A entrada, a informação sendo passada e interpretada, e a saída (lembrando que tudo isso são números). Após a passagem pelo output, ele pode ser repassado para um outro neurônio. O processo que calcula a saída de uma rede neural a partir das entradas é chamado de *"Feedforward"*.

Ao chegar no neurônio de saída, é preciso (num modelo supervisionado) ver o *Loss*, ou a perda, que compara o valor esperado com o valor que foi obtido.

Ao avistar um possível erro, o *"Backpropagation"* é feito para corrigi-lo na cadeia de neurônios, ajustando os pesos das funções, e esse ciclo continua até o usuário determinar. Esse encerramento pode ocorrer por vezes que os dados passam pela rede(épocas) ou por tolerância, medida atraves da acurácia do modelo, por exemplo, quando um modelo chegar numa acurácia de 90% e esse foi o desejado, então o modelo para o ciclo[Zhang et al. 2023].

Em síntese:

- *Feedforward* = Entrada -> saída
- *Backpropagation* = Saída -> Entrada

2. Perceptron

O que ocorre em um Perceptron simples é a conexão de x dados com um neurônio. Dessa forma 2.

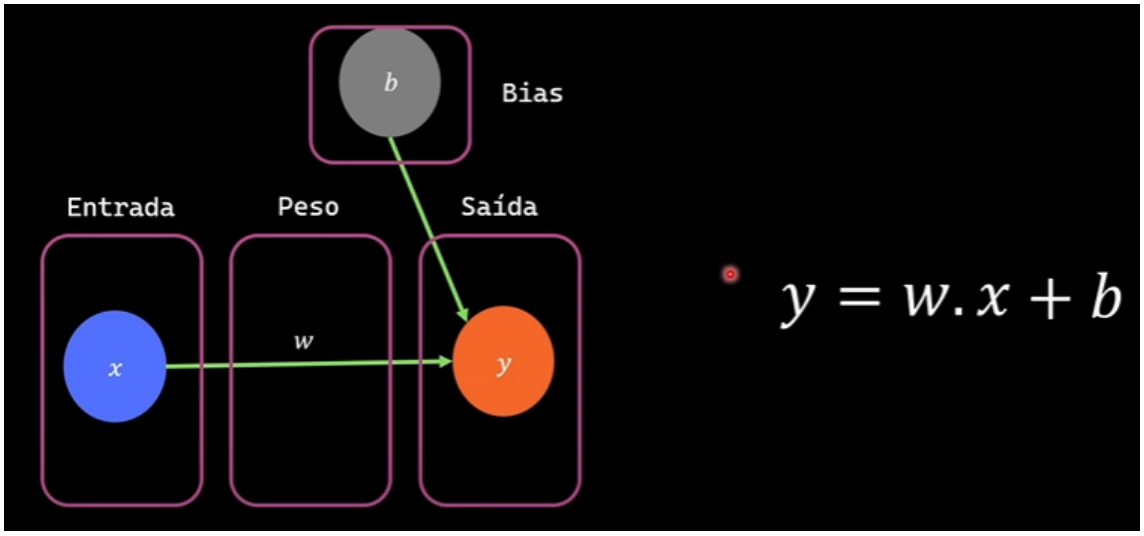

Figura 2. Rede Neural desmontada

Esse Bias, ou viés, é um número constante atrelado ao ”b”da formula, sendo o ”w”o peso um número e o ”x”o valor da entrada ou do dado e o ”y”sendo o neurônio que nesse caso é um *Perceptron*.

Então ao abordar problemas mais simples usamos esse mesmo modelo, o Perceptron simples, que possui apenas uma camada de saída, sem camadas ocultas.

2.1. Funções de ativação

Funciona como uma chave de liga e desliga, por exemplo, ao observar características de um aluno foi constatado que ele passou no ano letivo, para ele ser aprovado as suas características tem de convergir para serem favoráveis a sua aprovação, tendo essas características números, como por exemplo ”Aluno Esforçado: 0.1/1”,ao soma-las, elas tem de ser maior ou igual a um determinado número, então, ele terá o resultado 1, se ele não tiver chances de ser aprovado ele terá o resultado 0, tendo assim o resultado desse neurônio um desses números, esse resultado pode ser expresso pelo nome de *”Threshold”*, que é justamente a função que determina qual o resultado[Zhang et al. 2023].

3. Problema da conjunção ou da representação da porta AND com um Perceptron

No caso do problema instigado pelo docente Leonardo Nogueira Matos, o problema da conjunção ou da porta ”AND”, ele é um problema que é dito como linearmente separável, ou seja, é possível uma separação por uma única linha reta. Por exemplo, num gráfico com vários pontos vermelhos e azuis(vermelhos e azuis são dados de dois tipos feitos para exemplificação), se isso for possível através de uma reta, ele é linearmente

separável, se não, ele necessita de algo como uma parábola ou uma curva, então o modelo mais adequado é o de redes neurais multicamadas com função de ativação não-linear.

Ao mergulhar nos Perceptron, é possível achar uma diferença nas suas saídas, pois ao explorar modelos mais antigos, vê-se que eles são comumente treinados para atingirem resultados bipolares e não binários(codificação one-hot), ou seja, com números de 1 para verdadeiro e -1 para falso, assim melhorando a sua performance no início do modelo, mas tendo ela normalizada em um tempo maior[Davey et al. 2004]. Porém, usarei o modelo mais atual com os símbolos binários.

3.1. Implementação

Então, ao esquematizar a implementação, ele fica dessa forma:

```
andInput = [1,1]
# Nesse caso ele esta passando por cada casa do "a" e do "b" e ao passar ele anota o
    quanto
# errou e corrige cada peso

def perceptron(andInput):
  a = [1,1,0,0]
  b = [1,0,1,0]
#output desejado
  x = [1,0,0,0]
  weights = [0.1,0.8] # peso
  threshold = 1
  lr = 0.9
  i = 0
#colocando as epocas
  epochs = 0
  while epochs <= 2:
    i = 0
    print('Pesos obtidos'+ str(weights))
    while i < 4:
        soma = a[i]*weights[0] + b[i]*weights[1]
      #activationFunction
        out = activation(soma, threshold)
      #ajuste se teve um erro
        if (out != x[i]):
            error = x[i] - out
        #Rosenblatts perceptron learning algorithm
            weights[0] = weights[0] + lr * error * a[i]
            weights[1] = weights[1] + lr * error * b[i]
        i = i + 1
    epochs = epochs + 1

  soma = andInput[0]*weights[0] + andInput[1]*weights[1]
#somente depuracao
  print('Pesos obtidos'+ str(weights))
  return activation(soma,threshold)

print('Resultado desse input ' + str(andInput) + ' : ' +str(perceptron(andInput)))
```

Com exceção da chave de ativação que já foi previamente explicada, o código está indentado e comentado, assim facilitando a leitura e o entendimento.

Referências

Davey, N., Frank, R., Hunt, S., Adams, R., and Calcraft, L. (2004). High capacity associative memory models - binary and bipolar representation. *Applied Soft Computing - ASC*.

Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press. https://D2L.ai.