# RNA para camada simples

Inteligência Computacional

Sávio Rodrigues
*Design de gestão e informática*
*CEFET-MG - Campus V*
Divinópolis, Brasil
saviorodrigues012@gmail.com

***Resumo*—O presente trabalho visa a implementação de uma rede Neural de Camada simples. O algoritmo implementado, foi testado para os problemas OR e XOR.**

***Palavras-chave*—Or, Xor, Perceptron, RNA, Camada Simples**

## I. Introdução

O conexionismo é a criação de máquinas inteligentes a partir da reprodução e interconexão dos elementos de processamento do cérebro de animais. Nesse contexto, foi criado o neurônio artificial a partir de um modelo matemático simplificado do processamento existente em um neurônio biológico. O modelo de Neurônio de McCuloch e Pitts (MCP ) se tornou a base para todos os neurônios utilizado atualmente, caracterizado por receber entradas binárias e apenas uma única saída ponderada por pesos mostrado na figura 1. O Perceptron possui essencialmente essas características.

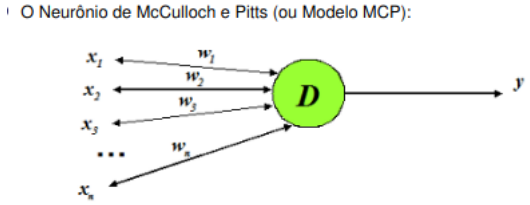


Fig. 1. Neurônio MCP

Outro aspecto importante na implementação do Perceptron é a função de ativação, ou seja, o processamento responsável pelo resultado da saída do neurônio. No presente trabalho, foi utilizado a função de ativação degrau por ser requerido saídas binárias.

## II. Implementação

A implementação do algoritmo de Redes Neurais de camada simples se deu a partir de uma classe chamada Perceptron. Essa classe possui como atributos a taxa de aprendizagem, o número máximo de épocas, o vetor de pesos e o theta (grau de ativação).

As duas funções que compõe essa classe são: *activation(inputs)* e *fit(inputs, y, yd)*. A *activation* recebe como parâmetro as variáveis de entrada e é responsável por retornar o valor 1 ou 0 de acordo com a ativação da função de transferência escolhida - função degrau. A *Fit* recebe como parâmetro as variáveis de entrada, o y - valor retornado na função de ativação- e o y esperado. Seu objetivo é aplicar o reajuste dos pesos a partir do erro obtido, seguindo a seguinte formulação:

$$w_i(p+1) = w_i(p) + \alpha x_i(p) e(p)$$

Em que $\alpha$ é a taxa de aprendizagem e $e(p)$ o erro obtido.

A taxa de aprendizagem é um valor constante e positivo menor que a unidade, que determina a velocidade de ajuste dos pesos da rede.

Os experimentos realizados, possuem as seguintes características:

- Na implementação, foi considerado $\alpha = 0.1$.
- Os pesos são valores aleatórios de $-0.5$ a $0.5$ para as duas entradas.
- Entradas e saídas: valores correspondentes à lógica OU a posteriormente X-OU ( Fig 2 ).
- O número máximo de épocas é um valor fixo de 100.

| Variáveis de Entrada | | E | OU | X-OU |
|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_1 \cap x_2$ | $x_1 \cup x_2$ | $x_1 \oplus x_2$ |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |

Fig. 2. Tabela lógica OR e XOR (últmas colunas)

## III. Resultados

### A. OU

O experimento com a função OU se mostrou bastante estável durante a realização dos testes. Mesmo trabalhando com números aleatórios de pesos, em menos de 15 épocas já foi o suficiente para a convergência do resultado. Isso se explica pelo fato de a lógica OU ser linearmente separável, garantindo portanto, a convergência dos resultados. A figura 3 ilustra o resultado de um dos testes realizados com essa função, que atingiu e convergência com 5 épocas.

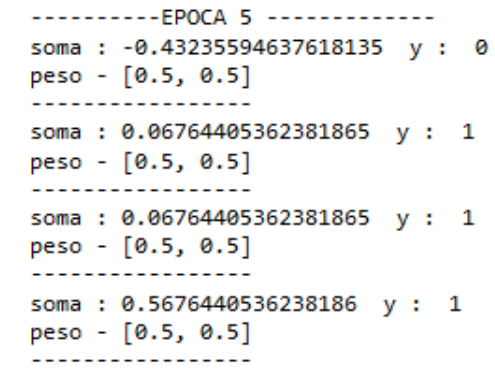


Fig. 3. Resultado obtido na lógica X-OU

### *B. X-OU*

Ao realizar esse experimento, o algoritmo não converge pra um resultado satisfatório. Isso se dá pelo fato de que a função X-OU não é linearmente separável, condição primordial para a convergência do algoritmo. Nesse sentido, em todas as execuções o algoritmo atinge o número máximo de épocas estabelecidas. A figura 4 faz referencia ao último valor impresso durante a execução do algoritmo na centésima época, sem que haja a convergência para os valores esperados.

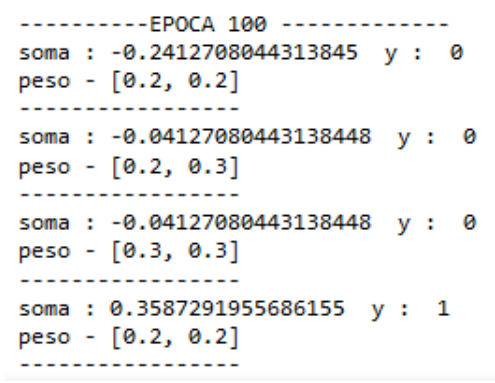


Fig. 4. Resultado obtido na lógica X-OU