RNA de Múltiplas Camadas

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 múltipla. O algoritmo implementado, foi testado para o problema XOR.**

***Palavras-chave*—Xor, RNA, Camada Múltipla**

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, no presente trabalho foi criado uma MLP (Multi-Layer Perceptron) ou seja, rede neural de múltiplas camadas a partir de um modelo matemático capaz de aprender e reajustar seus pesos e convergir á uma saída desejada - Figura 1. As principais características desse modelo são:

- Fluxo de sinal Unidirecional.
- Uma camada de estrada.
- Uma ou mais camadas escondidas.
- Uma camada de saída.

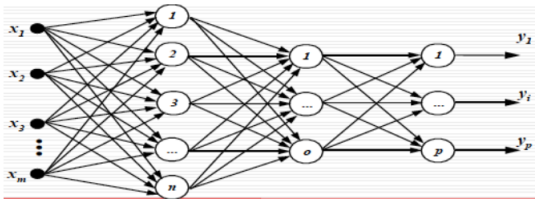

Fig. 1. Rede neural MLP

Uma vantagem da utilização de redes neurais de múltiplas camadas é que com duas ou mais camadas ocultas, a rede é capaz de aproximar qualquer função não linear, mesmo que descontínua. Um exemplo é a função XOR, cuja convergência não é atingida quando executada em uma rede neural de camada simples por exemplo. Nesse contexto, o presente trabalho busca resolver esse problema em uma MLP.

II. Implementação

A Figura 2 mostra a representação ilustrativa da rede testada com a nomeação das variáveis utilizadas na codificação. Nela, é possível identificar as conexões e seus respectivos pesos e a saída de cada neurônio.

A implementação do algoritmo de Redes Neurais de camadas múltiplas se deu a partir de uma classe chamada MLP.

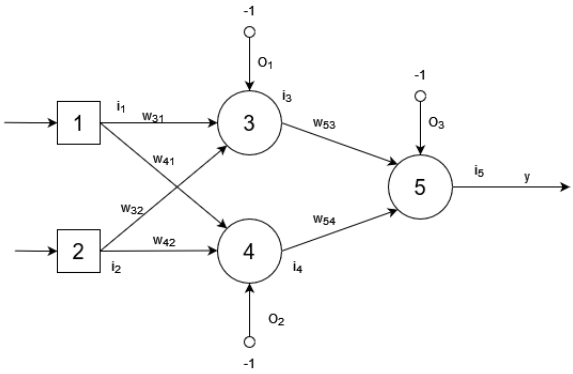

Fig. 2. Rede Neural e variáveis utilizadas

Essa classe possui como atributos: a taxa de aprendizagem, o número máximo de épocas, definição dos bias, o vetor de pesos, a tolerância, as entradas e saídas e o número de épocas já executadas.

As duas funções que compõe essa classe são: *activation(inputs)* e *fit()*. 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* é dividido em duas etapas principais: A propagação e a retro-propagação. A propagação- da camada de entrada para a camada de saída- onde é calculada a saída e o erro da rede a partir dos pesos em cada camada, semelhante àquilo que é feito no Perceptron. E por fim a retro-propagação - da camada de saída para a camada de entrada - em que é feito o reajuste dos pesos para as n camadas da rede onde cada conexão pode contribuir na saída.

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 a tolerância, $tol = 0.1$ (Critério de parada, é comparado com o erro obtido na época).
- O bias de todos os neurônios da rede é -1.
- Os pesos são valores aleatórios de -0.5 a 0.5 para todas

as conexões da rede.

- Entradas e saídas: valores correspondentes à lógica X-OU (Fig 3).
- 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. 3. Tabela lógica OR e XOR (últmas colunas)

III. Resultados

Utilizando uma rede com apenas uma camada oculta e dois neurônios (Fig. 3), foram coletados dados de 100 experimentos. Por se tratar de valores para os pesos aleatórios, nem sempre o algoritmo converge dentro do número máximo de épocas definidos. Em média, a convergência do algoritmo se deu em aproximadamente 36%. Dessas convergências, a média na quantidade de épocas que atingiram o objetivo foi de 51.

A seguir é feita a análise de um exemplo de como se deram os experimentos realizados. Nesse foi definido os seguintes pesos aleatórios (Fig 4).

```
Pesos escolhidos : {'w31': -0.919810747592088, 'w3O1': -0.927026152592780
8, 'w41': 1.6383332381880966, 'w4O2': 0.03333124034658634, 'w32': -0.11997
679019807837, 'w42': 2.009677285348276, 'w53': 0.776165843086989, 'w54':
3.0011497484504903, 'w5O3': -3.773971763474527}
```

Fig. 4. Pesos utilizados

A figura 5 faz referência à esse mesmo experimento em que houve a convergência da função X-OR com a saída: 0 1 1 0.

```
------------ EPOCA 35------------
         Saída : 0       Erro : 0
         Saída : 1       Erro : 0
         Saída : 0       Erro : 1
         Saída : 0       Erro : 1
---------------------------------
------------ EPOCA 36------------
         Saída : 0       Erro : 0
         Saída : 1       Erro : 0
         Saída : 1       Erro : 0
         Saída : 0       Erro : 0
---------------------------------
```

Fig. 5. Resultado obtido na lógica X-OU

É possível perceber que dentro de 36 épocas o algoritmo convergiu ao resultado esperado para a função X-OR.

IV. Conclusão

A função X-OU não é linearmente separável, por isso não é possível sua convergência apenas com o Perceptron. Por causa disso a necessidade da utilização da MLP que se mostrou bastante eficiente em convergir para um valor satisfatório para determinados pesos.