PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e de Informática

Marcos Antonio Lommez Candido Ribeiro[1]

Lista Extra #1

Inteligência Artificial

[1]Aluno de Graduação em Ciência da Computação – tonilommez@hotmail.com

Todo o código gerado pode ser encontrado no link a seguir

Link para o código

1) Implemente o algoritmo Backpropagation para resolver as funções AND, OR and XOR com n entradas booleanas.

Nestes experimentos, você deverá investigar:
1) A importância da taxa de aprendizado
2) A importância do bias
3) A importância da função de ativação. Investigue pelo menos 2 funções
A sua lista deverá conter todas as explicações da implementação e os resultados dos testes realizados.
Coloque prints das entradas e saídas e explique o que foi observado.
Ao final, disponibilize o código desenvolvido

Para o presente problema foi construído um algoritmo flexível de RNA Backpropagation capaz de receber hiper-parâmetros de maneira a personaliza-lo

As seguinte funções de ativações foram preparadas:

```python
# Definindo as funções de ativação e suas derivadas
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def sigmoid_derivative(x):
    return sigmoid(x) * (1 - sigmoid(x))

def relu(x):
    return np.maximum(0, x)

def relu_derivative(x):
    return (x > 0) * 1

def tanh(x):
    return np.tanh(x)

def tanh_derivative(x):
    return 1 - np.tanh(x)**2
```

Todas as funções foram automatizadas, sendo os valores de testes gerados automaticamente com o uso de uma função "create_data"que recebe o tipo de operação, seja **AND**, **OR** ou **XOR**.

Similar a implementação do grid_search foi implementado um algoritmo de testagem de todas as combinações possíveis de hiper-parâmetros dentro de uma lista fornecida:

```python
def perform_experiments(functions, n_inputs_list, activations, learning_rates, use_bias_list, n_hidden_list, epochs_list):
    results_list = []

    for function in functions:
        for n_inputs in n_inputs_list:
            for activation in activations:
                for learning_rate in learning_rates:
                    for use_bias in use_bias_list:
                        for n_hidden in n_hidden_list:
                            for epochs in epochs_list:
                                predictions, accuracy = train_network(
                                    function, n_inputs, activation, learning_rate, use_bias, epochs, n_hidden
                                )
                                # Convert predictions from boolean to integer
                                predictions_numeric = predictions.astype(int)
                                # Retrieve expected values for the logical function
                                _, expected = create_data(function, n_inputs)
                                # Append the experiment details and accuracy to the results list
                                results_list.append([
                                    function.upper(), n_inputs, activation, learning_rate,
                                    use_bias, n_hidden, epochs, expected, predictions_numeric, accuracy
                                ])

    # Convert the results list to a NumPy array for better manipulation and visualization
    results_np_matrix = np.array(results_list, dtype=object)
    df = pd.DataFrame(results_np_matrix, columns=column_names)
    return df
```

Assim sendo, foi definida a seguinte tabela de hiper-parâmetros a testar apresentado na tabela, juntamento com sua saida em formato de tabela

```python
functions = ['and', 'or', 'xor']
n_inputs_list = [3]
activations = ['sigmoid', 'relu', 'tanh']
learning_rates = [0.1, 1]
use_bias_list = [True, False]
n_hidden_list = [2, 4]
epochs_list = [1000, 10000]
```

	Op	n_inputs	activation	learning_rate	bias	n_hidden	epochs	expected	predicted	accuracy
0	AND	3	sigmoid	0.1	True	2	1000	[[0, 0, 0, 0, 0, 0, 0, 1]]	[[0, 0, 0, 0, 0, 0, 0, 0]]	0.875
1	AND	3	sigmoid	0.1	True	2	10000	[[0, 0, 0, 0, 0, 0, 0, 1]]	[[0, 0, 0, 0, 0, 0, 0, 1]]	1.0
2	AND	3	sigmoid	0.1	True	4	1000	[[0, 0, 0, 0, 0, 0, 0, 1]]	[[0, 0, 0, 0, 0, 0, 0, 0]]	0.875
3	AND	3	sigmoid	0.1	True	4	10000	[[0, 0, 0, 0, 0, 0, 0, 1]]	[[0, 0, 0, 0, 0, 0, 0, 1]]	1.0
4	AND	3	sigmoid	0.1	False	2	1000	[[0, 0, 0, 0, 0, 0, 0, 1]]	[[0, 0, 0, 0, 0, 0, 0, 0]]	0.875
...	...	...	...	...	...	...	...	...	...	...
139	XOR	3	tanh	1	True	4	10000	[[0, 1, 1, 0, 1, 0, 0, 1]]	[[0, 1, 1, 0, 1, 0, 0, 1]]	1.0
140	XOR	3	tanh	1	False	2	1000	[[0, 1, 1, 0, 1, 0, 0, 1]]	[[0, 1, 1, 0, 1, 0, 0, 0]]	0.875
141	XOR	3	tanh	1	False	2	10000	[[0, 1, 1, 0, 1, 0, 0, 1]]	[[0, 1, 1, 0, 1, 0, 0, 0]]	0.875
142	XOR	3	tanh	1	False	4	1000	[[0, 1, 1, 0, 1, 0, 0, 1]]	[[0, 1, 1, 0, 1, 0, 0, 1]]	1.0
143	XOR	3	tanh	1	False	4	10000	[[0, 1, 1, 0, 1, 0, 0, 1]]	[[0, 1, 1, 0, 1, 0, 0, 1]]	1.0

Possuindo essa tabela é possível realizar analises sobre os dados, e para isso apliquei duas técnicas, sendo a primeira uma analise da acurácia media para cada uma das características usadas, como foi realizada uma analise combinatória completa entre todos os valores a média se torna assim um bom parâmetro para entender a sua influencia no resultado final, os valores encontram-se a seguir:

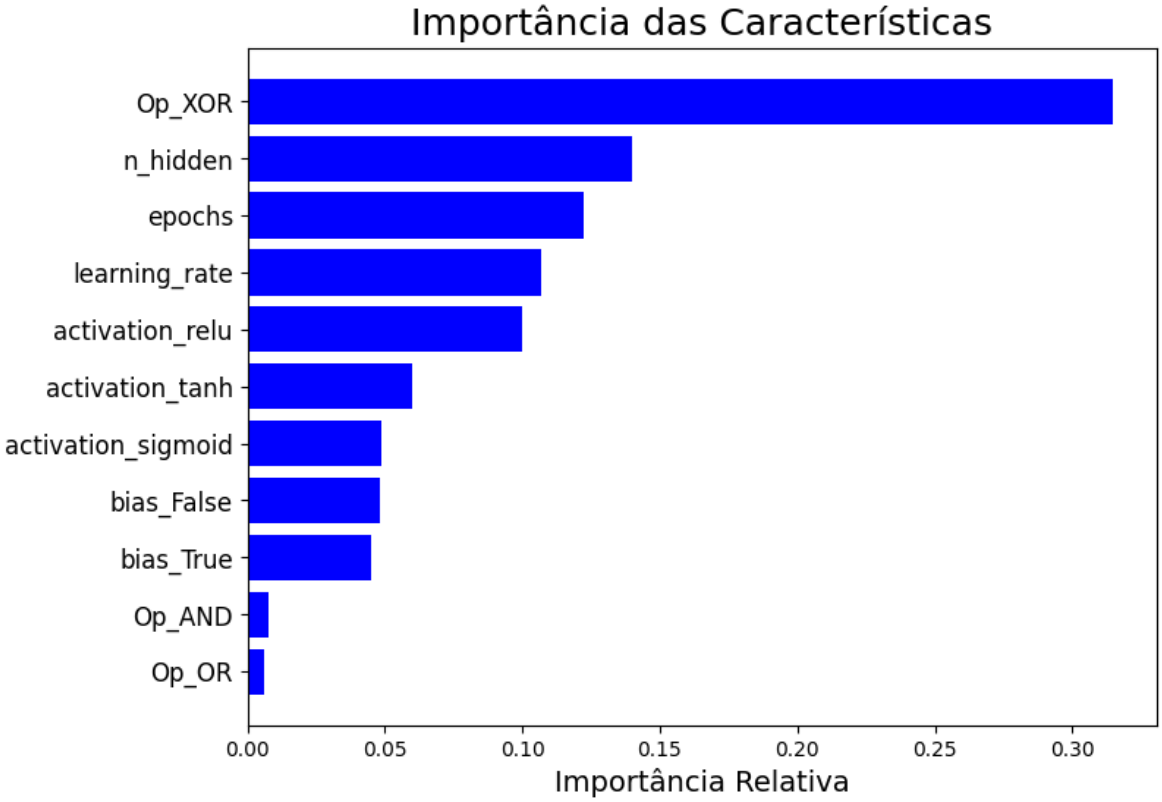
```
Op
AND     0.96875
OR      0.96875
XOR    0.791667
Name: accuracy, dtype: object

activation
relu       0.861979
sigmoid    0.914062
tanh       0.953125
Name: accuracy, dtype: object

learning_rate
0.1    0.901042
1.0    0.918403
Name: accuracy, dtype: object
```

```
bias
False    0.904514
True     0.914931
Name: accuracy, dtype: object

n_hidden
2    0.876736
4    0.942708
Name: accuracy, dtype: object

epochs
1000     0.899306
10000    0.920139
Name: accuracy, dtype: object
```

Nessa primeira analise percebemos que a função ativadora possui uma grande influencia no problema selecionado, tendo a função de tangente hiperbólica uma media impressionante de 95% de acurácia, mesmo influenciada por um treinamento com baixíssimo numero de épocas, alto learning_rate, sem bias e apenas 2 camadas de neurônios.

Por fim os outros parâmetros se mostraram pouco influentes isoladamente na definição da acurácia total, mas ainda influentes no modelo. Sendo a única exceção o numero de camadas ocultas, o que faz sentido porque camadas adicionam um nível de profundidade maior capaz de lidar com a operação xor que mais influencia negativamente os modelos.

Para enriquecer a analise podemos aplicar um treinamento de **Random Forest** para analisar nossos dados gerados e assim enxergarmos quais hiper-parâmetros mais influenciaram nosso modelo, através da analise da importância das classes geradas no Random Forest. Os resultados encontrados estão a seguir:



Diferentemente da analise de media de acurácia, a ordem dos parâmetros na arvore não pode ser analisada cruamente, porque não necessariamente reflete o peso das classes em relação a acurácia e sim na hora de gerar o valor de saída, sendo que esse peso da arvore pode ser para diminuir ou aumentar, não necessariamente apenas aumentar, assim como visto que a característica de mais importância é a operação XOR que diminui drasticamente a acurácia do modelo, ainda sim ser usado como um insight para entender o peso que cada parâmetro possui.

Agora explicando conceitualmente a importância da **taxa de aprendizado**, **bias** e da **função de ativação**:

Taxa de Aprendizado

A taxa de aprendizado é um dos fatores mais cruciais de uma rede neural, sendo ela a velocidade de aprendizado no qual uma formula é adaptada através do backpropagation. Em qualquer modelo baseado em alguma especie de regressão a taxa de aprendizado tem uma influencia decisiva se for estimada além do necessário ou menor do que o necessário, sendo seu valor padrão recomendado 0.001 na maioria dos casos.

Caso seu valor seja super estimado, o modelo poderá passar direto do ponto ideal e começar a crescer exponencialmente para fora, sempre pulando diretamente de um ponto negativo da derivada para um positivo, ou inversamente, ignorando o ponto mínimo. Por outro lado uma taxa de aprendizado subestimada ira fazer o modelo não chegar ao ponto de conversão, assim sendo aprendendo muito lentamente, o que não é um problema grave, mas ira gastar um tempo computacional demasiadamente grande, além de ser mais difícil de medir o ponto de conversão.

Bias

O bias tem o mesmo papel na formula gerada pelo modelo que o parâmetro "b"possui em uma equação, sendo esta uma constante que sobe ou desce no eixo Y a linha ou curva que estamos gerando. Assim sendo indispensável o uso do bias, porque este poderá gerar erros fatais caso os dados apresentados sejam "nebulosos"na região próxima ao numero 0, gerando erro de predição.

É importante dizer que o bias não deve ser confundido com o termo "viés"pois possui significado diferente neste contexto, mais como um nome próprio para esta variável matemática do que um significado real.

Função de Ativação

A função de ativação de uma rede neural tem o papel de decidir como os dados gerados pela rede irão ser demonstrados em sua saída. Como um exemplo simples a função sigmoide é criada especificamente para gerar uma súbita subida de 0 para 1, assim evitando valores como 0,5, isso permite as redes neurais trabalharem com modelos de classificação em vez de ser um modelo de regressão. Além de ter influencia direta em como os dados são treinados, devido a utilização de sua derivada no momento da retro propagação.

2) Investigue a razão do backpropagation utilizar uma função de ativação não linear, ao invés de uma linear como Percetron faz.

O algoritmo de backpropagation é capaz de utilizar uma ativação não linear devido ao fato de implementar uma estrategia de uso de atualização de pesos baseado na derivada da função de custo, o que significa que assim é possível utilizar uma formula quadrática ou ainda mais complexa e atualizar seus pesos gradualmente com a sequencia de épocas, por fim assim ultrapassando a limitação instrisecamente linear de operações como XOR.

Além disso o backpropagation também permite a utilização de camadas ocultas, o que permite que a formula gerada no final seja ainda mais complexa no sentido de ser capaz de lidar com dados mais complexos do que uma simples implementação. Com esse modelo de camadas é possivel ate mesmo diferenciar uma operação XOR usando formulas lineares.

Investigue o uso de pelo menos 3 funções de ativação normalmente utilizada em algoritmos e redes neurais. Qual a relevância da derivada desta função no treinamento do algoritmo?

No algoritmo realizado na questão (1) foi estudado o uso de 3 diferentes funções, sendo estas Sigmoid, Relu e Tahn. Todas cujo quais também são necessárias as utilizações de suas respectivas derivadas.

As derivadas são usadas porque assim é possível medir o grau de inclinamento da curva, para saber o quão distante a formula gerada esta da formula ideal de ser utilizada.