Project Report for Data Structure

Simple implementation of xor

using neural network

錢宇倫
通訊二
110503511

Department of Communication Engineering,

September 28, 2022

Contents

1 Compile Result

```
whimy@LAPTOP-3EHNOILN:/mnt/c/Users/User/Desktop/110503511_assignment_1/xor_in_nn$ make
cc -g -Wall -Werror -c main.c
cc -pthread -lpthread -o backprop main.o layer.o neuron.o -lm
whimy@LAPTOP-3EHNOILN:/mnt/c/Users/User/Desktop/110503511_assignment_1/xor_in_nn$ gcc main.c -lm layer.c neuron.c
```

figure.1 - compile

2 Run Code

Setting background...

```
whimy@LAPTOP-3EHNOILN:/mnt/c/Users/User/Desktop/110503511_assignment_1/xor_in_nn$ ./a.out
File open successed.
Setting the number of Layers in Neural Network:4
Setting the number of neurons in layer[1]:2
Setting the number of neurons in layer[2]:4
Setting the number of neurons in layer[3]:4
Setting the number of neurons in layer[4]:1

Created Layer: 1
Number of Neurons in Layer 1: 2
Neuron 1 in Layer 1 created
Neuron 2 in Layer 1 created

Created Layer: 2
Number of Neurons in Layer 2: 4
Neuron 1 in Layer 2 created
Neuron 2 in Layer 2 created
Neuron 3 in Layer 2 created
Neuron 4 in Layer 2 created

Created Layer: 3
Number of Neurons in Layer 3: 4
Neuron 1 in Layer 3 created
Neuron 2 in Layer 3 created
Neuron 3 in Layer 3 created
Neuron 4 in Layer 3 created

Created Layer: 4
Number of Neurons in Layer 4: 1
Neuron 1 in Layer 4 created
```

figure.2 - background setting

```
Neural Network Created Successfully...

Setting the learning rate: 0.15

Setting the number of training examples: 4
```

figure.3 - background setting

Initializing weight...

```
Initializing weights...
0:w[0][0]: 0.369542
1:w[0][0]: 0.026532
2:w[0][0]: 0.120254
3:w[0][0]: 0.461154
0:w[0][1]: 0.859829
```

figure.4 - Initialization

Feed input and desired output of each training example

```
Enter the Inputs for training example[0]:
0 0

Enter the Inputs for training example[1]:
0 1

Enter the Inputs for training example[2]:
1 0

Enter the Inputs for training example[3]:
1 1

Enter the Desired Outputs (Labels) for training example[0]:
0

Enter the Desired Outputs (Labels) for training example[1]:
1

Enter the Desired Outputs (Labels) for training example[2]:
1

Enter the Desired Outputs (Labels) for training example[3]:
0
```

figure.5 - Feed input and desired output

Show the training process if user enter 'y'

```
Would you like to see the detailed process?[y/n]:y
Input: 0.000000
Input: 0.000000
Output: 1
MAE: 0.931585
MSE: 0.867850

Input: 0.000000
Input: 1.000000
Output: 1
MAE: 0.010321
MSE: 0.000213
```

figure.6 - Visible training process

Show the training process if user enter n

```
Would you like to see the detailed process?[y/n]:n
Training completed!
You can see the detailed training process in a csv file named 'train.csv'

Now you can enter some examples to test-->
Enter input's number of bits(Enter 0 to exit the test):
```

figure.7 - Invisible training process

Testing First, enter the number of your input's bit.
Second, enter your n-bits input.

```
Training completed!
You can see the detailed training process in a csv file named 'train.csv'

Now you can enter some examples to test-->
Enter input's number of bits(Enter 0 to exit the test):2
Enter the input:11
Output: 1

Enter input's number of bits(Enter 0 to exit the test):5
Enter the input:10101
Output: 1
```

figure.8 - Testing

Exit the code Enter 0 to exit the code.

```
Enter input's number of bits(Enter 0 to exit the test):0
whimy@LAPTOP-3EHNOILN:/mnt/c/Users/User/Desktop/110503511_assignment_1/xor_in_nn$
```

figure.9 - Exit

3 Analysis

MAE(Mean-Absolute Error) of training

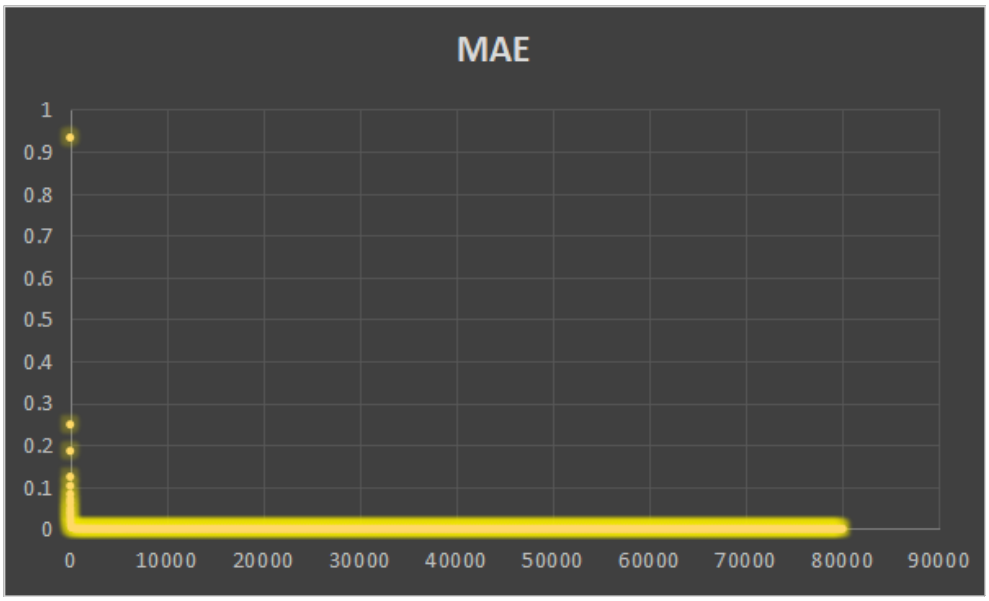

figure.10 - Graph of MAE

MSE(Mean-Square Error) of training

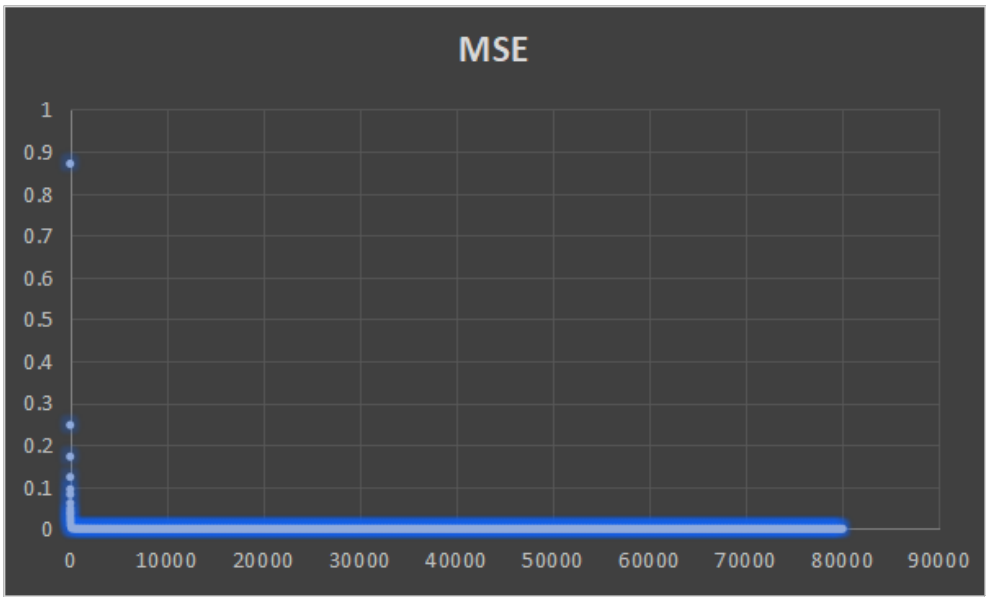

figure.11 - Graph of MSE

4 Discussion

About activation function...
In this case, we use two types of act.funcion:

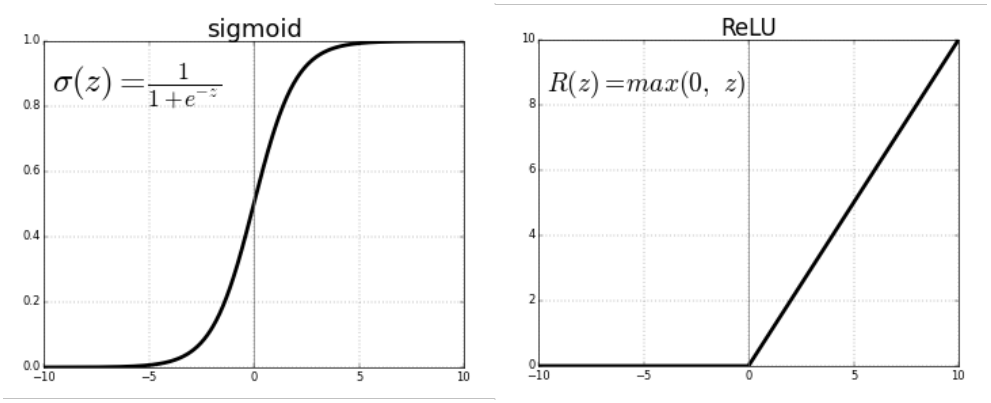

figure.12 - Graphs of each act,function

About analysis...
After the training process, a csv file will be created.
It save each MAE(Mean-Absolute Error) and MSE(Mean-Square Error).
Here are math equations of each loss function:

$$MAE = \frac{\sum_{i=1}^{n} |ActualOutput - DesiredOutput|}{n} \tag{1}$$

$$MSE = \frac{\sum_{i=1}^{n} (ActualOutput - DesiredOutput)^2}{n} \tag{2}$$

5 To-Improve

About code...
Add more Error-proofing.
Use less memory
Make the UI better.

References

[1] Code reference(GitHub)

[2] Building Neural Network Framework in C using Backpropagation

[3] Simple neural network implementation in C

[4] Neural networks and back-propagation explained in a simple way

[5] Activation Functions in Neural Networks