Deep Learning and Practice Lab0 Report

1. Introduction

In this lab, we should build a deep neural network to implement XOR regression problem. Toward my work, I implement the layers which follow the idea of pytorch. The layers are cascading between each other and have *forward* and *backward* operations. Moreover, whole layer should inherit the nn.Module class which is usual father class defined by myself. The architecture of my neural network is shown in Figure 1, and the screen shot of model idea is illustrated in Figure 2.

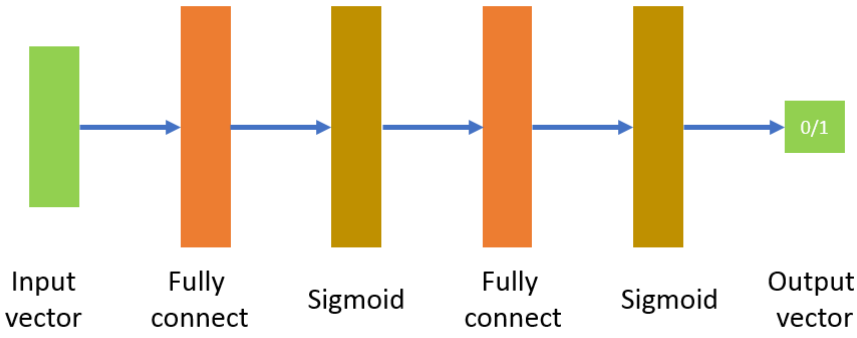

Figure 1. The structure of my network to solve XOR problem

```
class Module(object):
    def __init__(self):
        pass

    def __call__(self, x):
        return self.forward(x)

    def forward(self, *args):
        pass

    def step(self, *args):
        pass
```

Figure 2. The idea of fundamental nn.Module

2. Experiment setups

In my experiment, the sigmoid function is adopted after every fully connected layer. The neural network structure is shown in Figure 1 which is a common deep neural network, and the number of layer is 1. Additionally, I implement the bonus part whose number of layer is 2.

In the process of back-propagation, the order and input of derivation in equation

is very important. Especially, we should consider the value changing between activation function. The equation 1 shows the function of sigmoid backward derivation.

$$\nabla \mathrm{Sigmoid}(\mathrm{x}) = \frac{\partial C}{\partial h} * \mathrm{Sigmoid}(\mathrm{x}) * (1 - \mathrm{Sigmoid}(\mathrm{x})) \quad (1)$$

In the equation 1, the $\frac{\partial C}{\partial h}$ is the derivation which is passed from the next layer. To be notice, the x is the input of sigmoid operation during forward operation.

$$\frac{\partial h}{\partial W} = \frac{\partial C}{\partial h} * x \quad (2)$$

The equation 2 shows the derivation of fully convolution. In the fully connected layer, the derivation of bias is like the derivation of weight matrix. The process is just multiplying the backward result which is passed from next layer with the input tensor x. The detail can be checked in my code.

3. Results

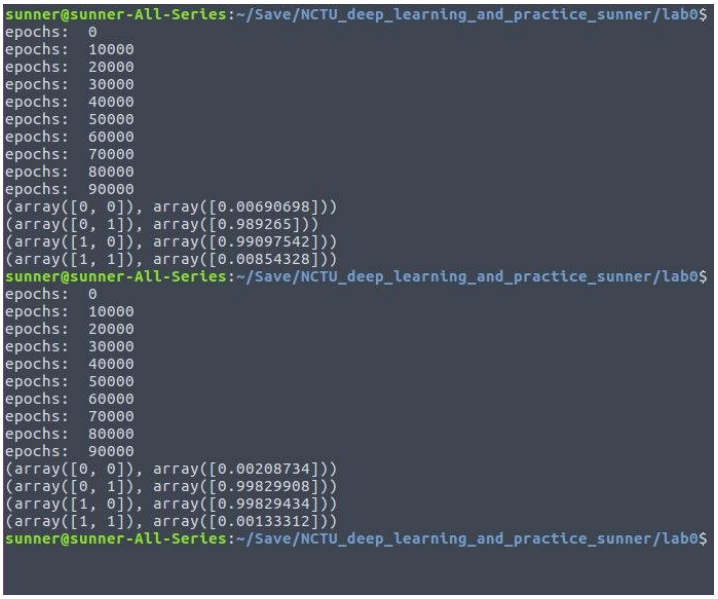

Figure 3. The testing result of lab0 and bonus

In Figure 3, The testing results are shown. I follow the format which is shown in the pdf description file and print the testing value. Even though the extreme small value cannot be obtained after 100000 training. However, we can easily determine the result if the value is close to zero or not. The positive result will be judge if the value is close to 1, and the negative result will be judge if the value is close to 0. As the result, the XOR result can be performed by this model.

4. Discussion

In this lab, we try to implement the simple deep neural network from scratch. Additionally, we adopt the sigmoid function whose derivation can be gotten easily. By my implementation, the XOR result can be obtained clearly.

There're two direction which can improve the result. First, we can increase the number of hidden layer neuron. In my experiment, the number of hidden is 3 (In bonus, the number in second layer is 10). If we increase the neuron, the performance might raise. Second, we can add more layer and try to transform the data to the higher feature space.

```
sunner@sunner-All-Series:~/Save/NCTU_deep_learning_and_practice_sunner/lab0$ python3 bonus.py
epochs:  0
epochs:  10000
epochs:  20000
epochs:  30000
epochs:  40000
epochs:  50000
epochs:  60000
epochs:  70000
epochs:  80000
epochs:  90000
(array([0, 0]), array([0.00706701]))
(array([0, 1]), array([0.9988315]))
(array([1, 0]), array([0.99269903]))
(array([1, 1]), array([0.00087952]))
sunner@sunner-All-Series:~/Save/NCTU_deep_learning_and_practice_sunner/lab0$ python3 bonus.py
epochs:  0
epochs:  10000
epochs:  20000
epochs:  30000
epochs:  40000
epochs:  50000
epochs:  60000
epochs:  70000
epochs:  80000
epochs:  90000
(array([0, 0]), array([0.00114212]))
(array([0, 1]), array([0.99868498]))
(array([1, 0]), array([0.99863011]))
(array([1, 1]), array([0.00150918]))
sunner@sunner-All-Series:~/Save/NCTU_deep_learning_and_practice_sunner/lab0$
```

Figure 4. The testing result after increasing the number of neuron

The Figure 4 shows the result that use more neuron in each layer. The upper side is the result that use 10 neurons, and the lower side use 64 neurons. the Figure 5 shows the result that use 3 layers to train the model. Conversely, these results don't improve a lot after these changing. What's worse, the more number of neuron or layers you adopt, the more time you should spend to train and inference.

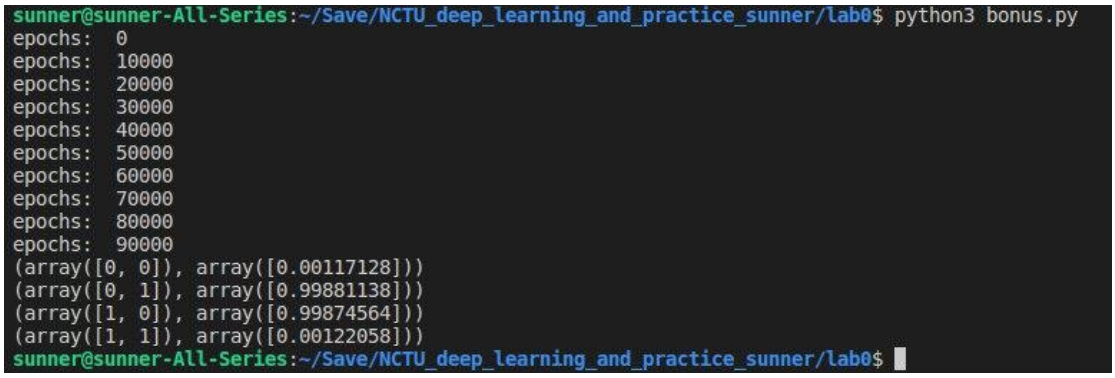

Figure 5. The testing result after adopting 3 hidden layers

The last approach is that the more training iteration can be used, but we should also consider the overfitting problem when increasing iteration technique is adopted.