DLP HW1

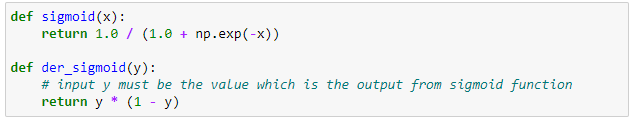
多工所碩一 楊奕儒

ID：312553038

1. **Introduction**

In this lab, I implemented a simple neural network with forward and backward pass using two hidden layers to classify input data.

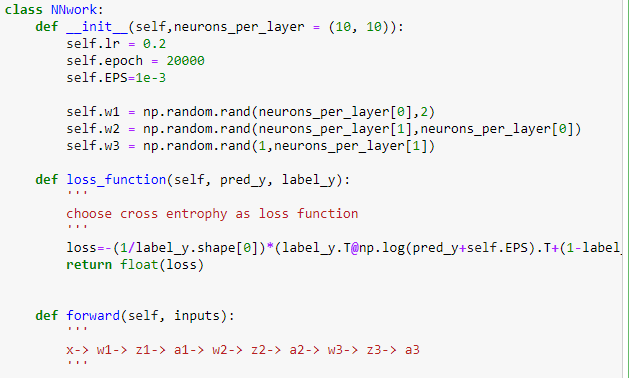
1. **Experiment setups**
2. **Sigmoid functions**



1. **Neural network**

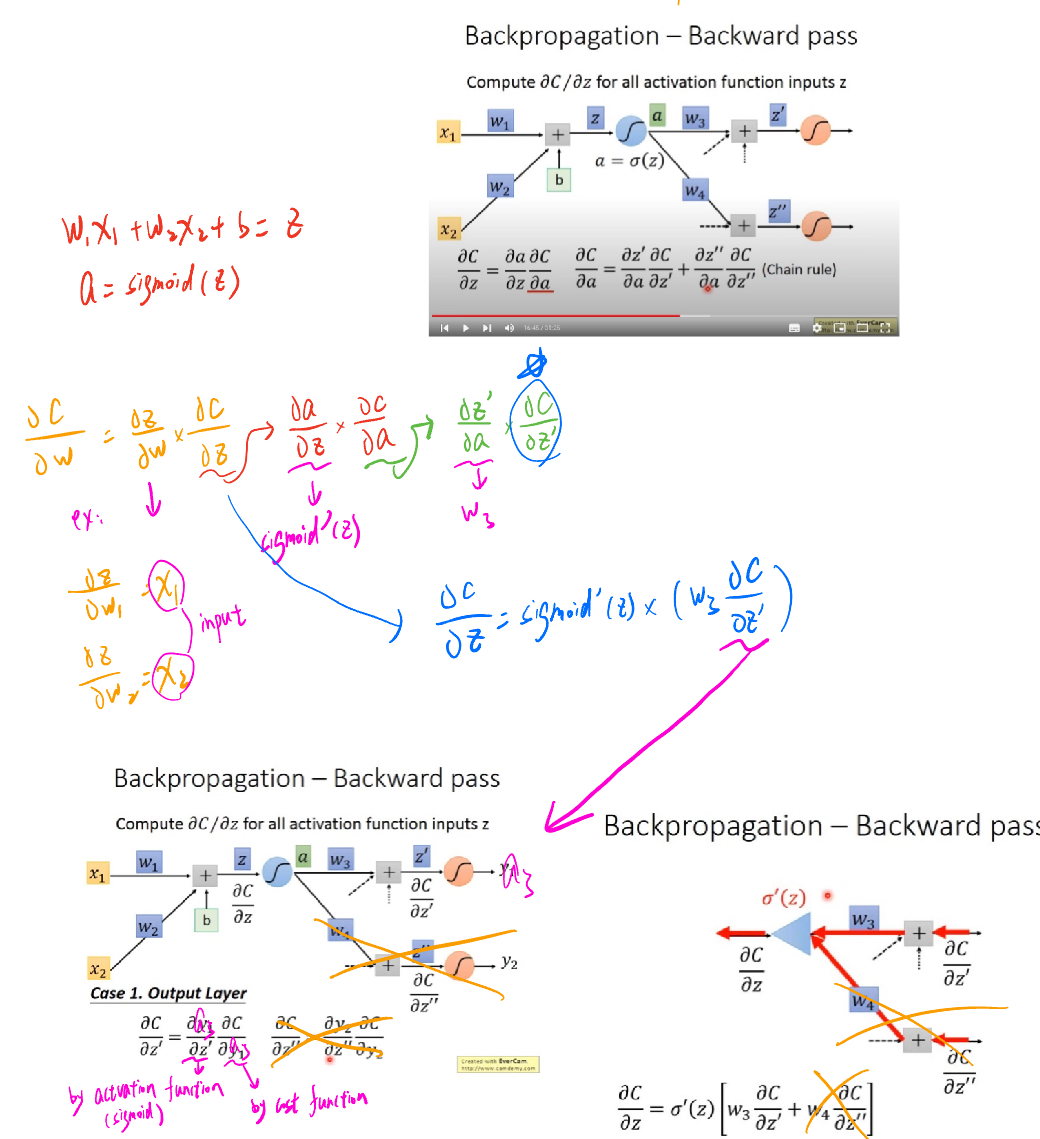
In my neural network, there are ten neurons in both two hidden layers , and the learning rate is set to 0.2.

Also, I chose cross entropy as loss function to calculate loss.



1. **Backpropagation**

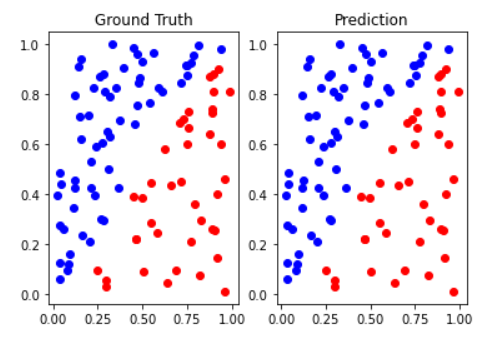
Based on these pictures below, I calculated the gradient and update the weight between layers.

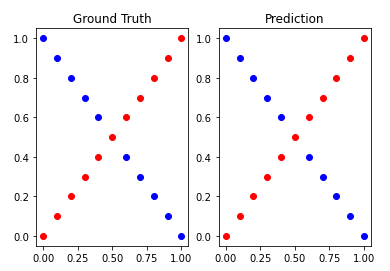


**(3) Results of your testing**

**(a) Screenshot and comparison figure**

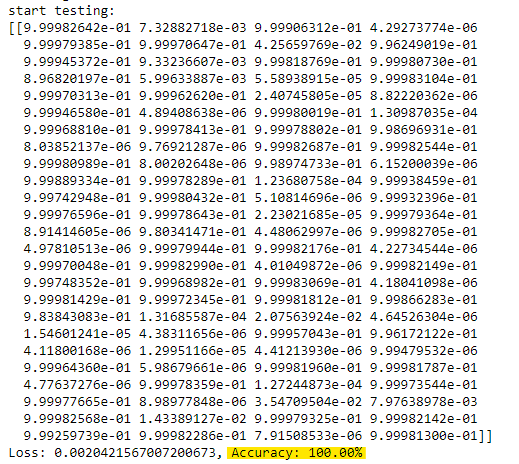
Linear： XOR：



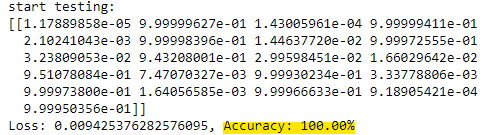


**(b) Show the accuracy of your prediction**

Linear：

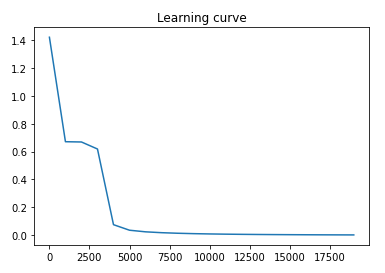


XOR：

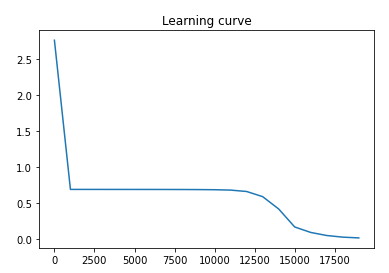


1. **Learning curve (loss, epoch curve)**

Linear：



Xor：



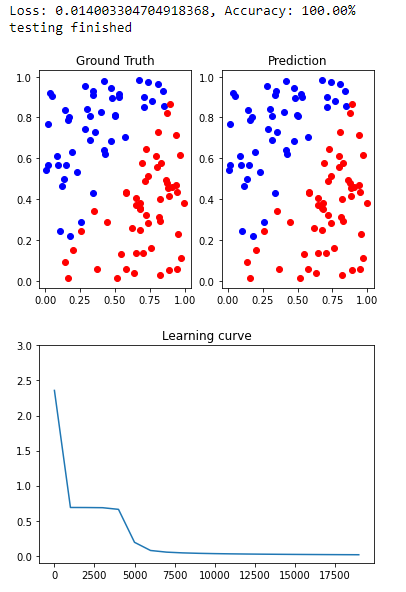
1. **Anything you want to present**

Based on the output, we can see that both linear and XOR input data achieve 100% accuracy. However, in the XOR input case, the loss remains constant for a certain period of time before eventually converging.

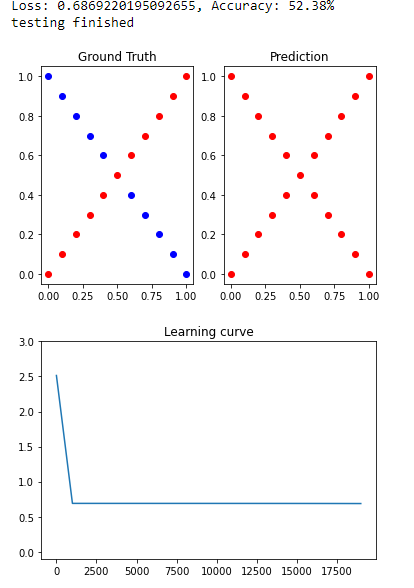
1. **Discussion**
2. **Try different learning rates**

After lowering the learning rate to 0.1, we observed that in the linear case, the loss still converges and the accuracy reaches 100%. However, in the XOR case, the loss does not converge. I suspect that the gradients are getting stuck in a local minimum, so we need to increase the learning rate in order for it to converge.

Linear：



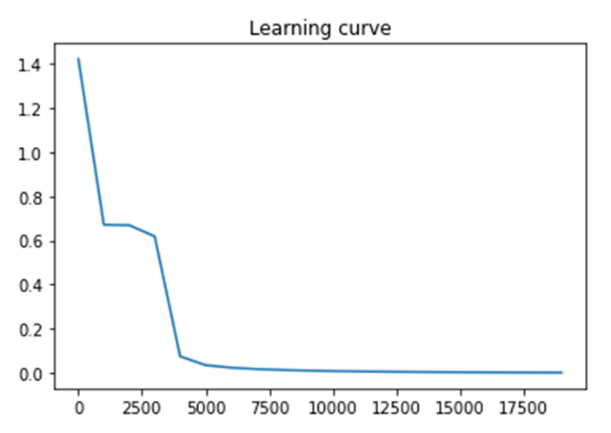
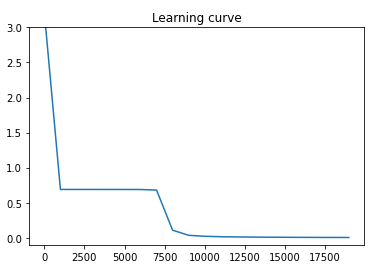
XOR：



1. **Try different numbers of hidden units**

Linear:

After doubling the number of neurons in each layer, we observed that although the loss can still converge, it requires more epochs to achieve convergence in linear case.

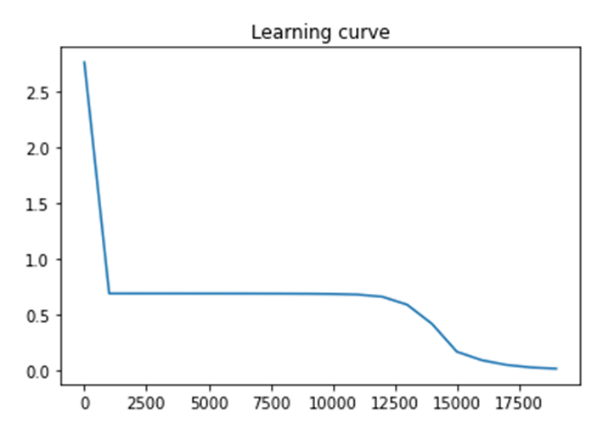
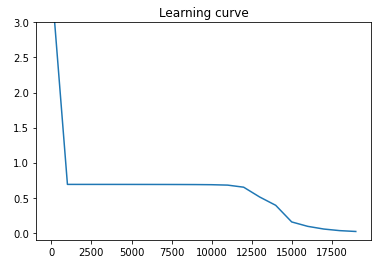


XOR：

10 neurons in each layer 20 neurons in each layer

XOR：

After doubling the number of neurons in each layer, we observed that the number of epochs required to converge did not change significantly compared to the linear case.



10 neurons in each layer 20 neurons in each layer

1. **Try without activation functions**

If I remove the activation functions from my neural network, the output may generate negative values. Consequently, when calculating the loss using cross entropy, the logarithm function cannot handle negative values, resulting in an incorrect loss calculation.

Reference：

https://www.youtube.com/watch?v=ibJpTrp5mcE&list=PLJV\_el3uVTsPy9oCRY30oBPNLCo89yu49&index=12&t=1193s