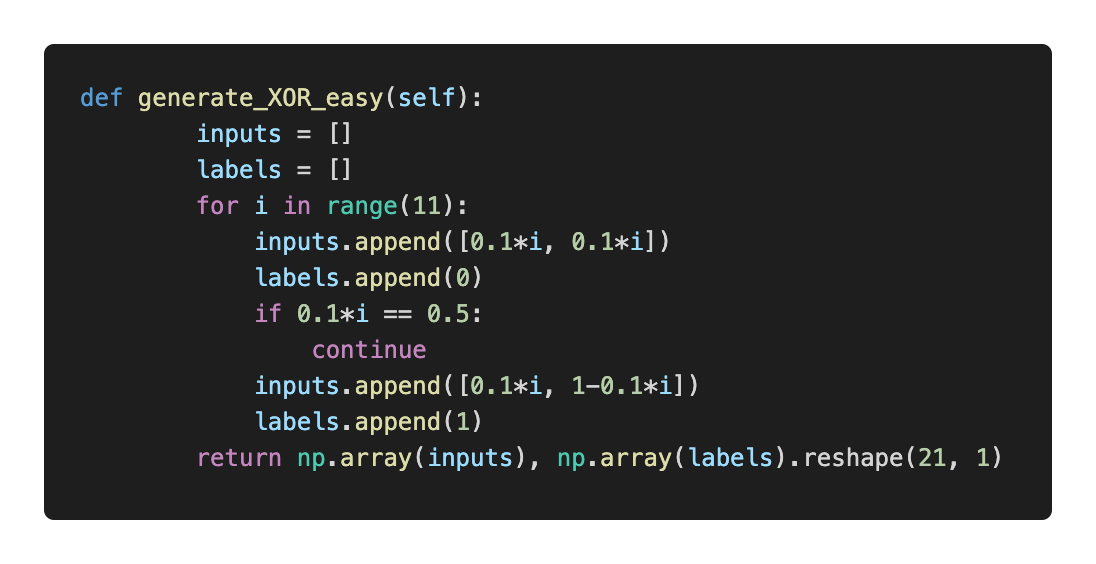
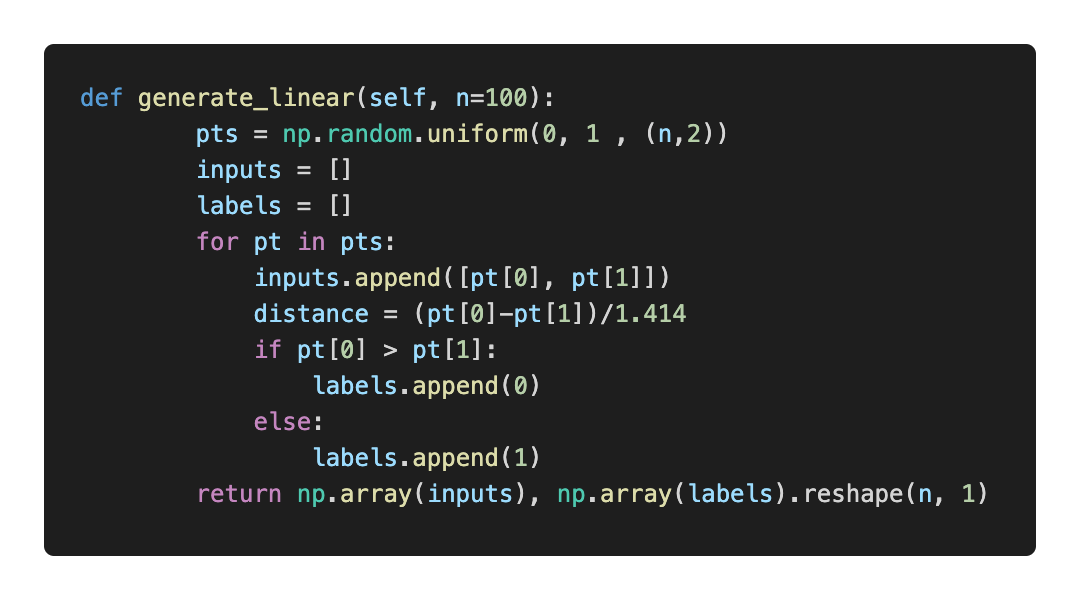
**Lab1: back-propagation Report**

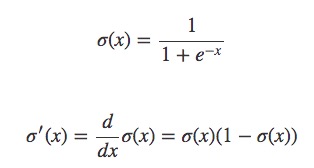
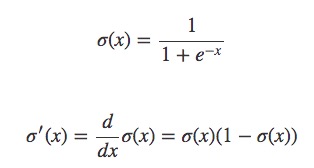
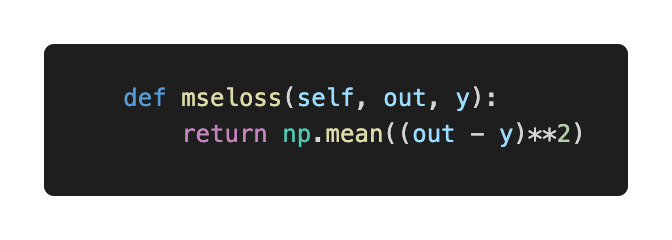
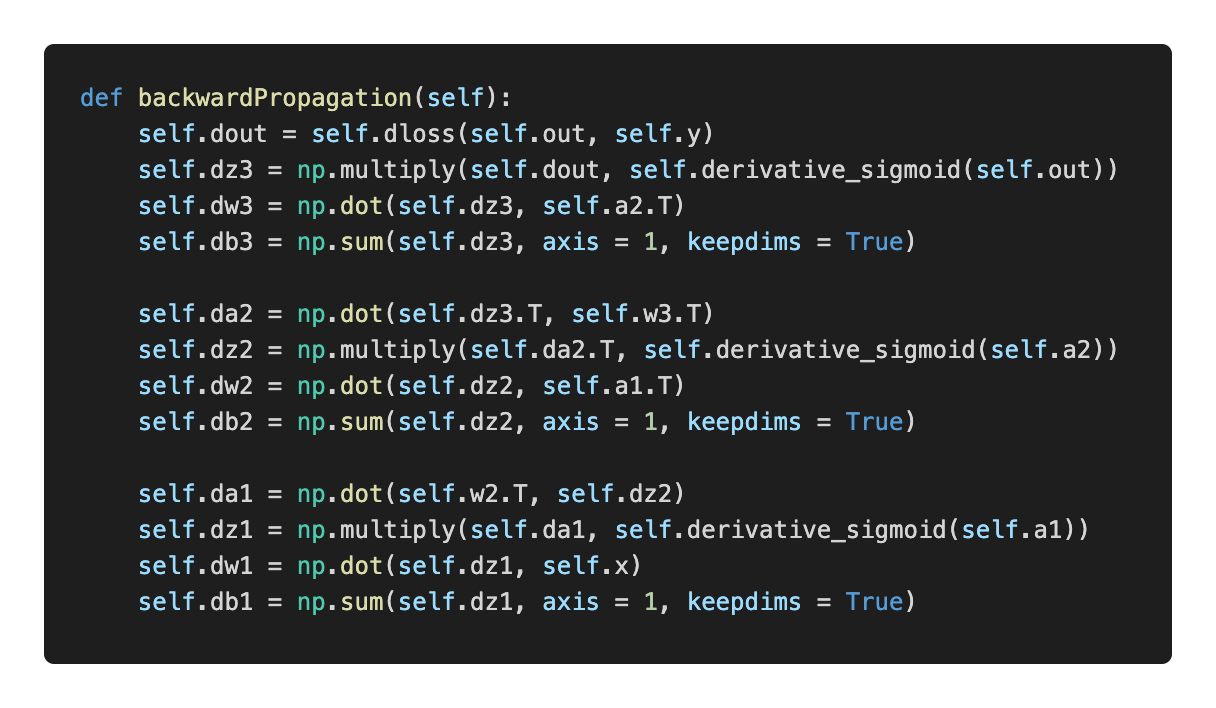
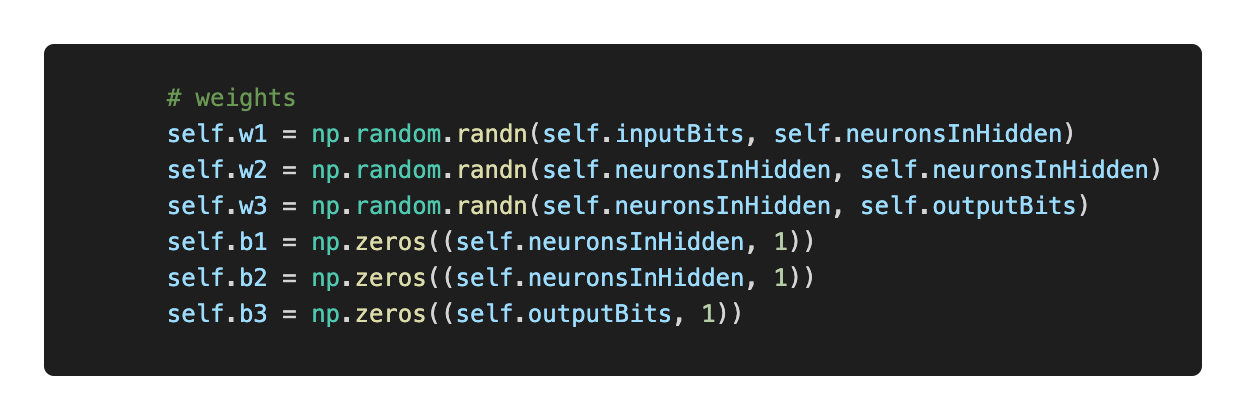
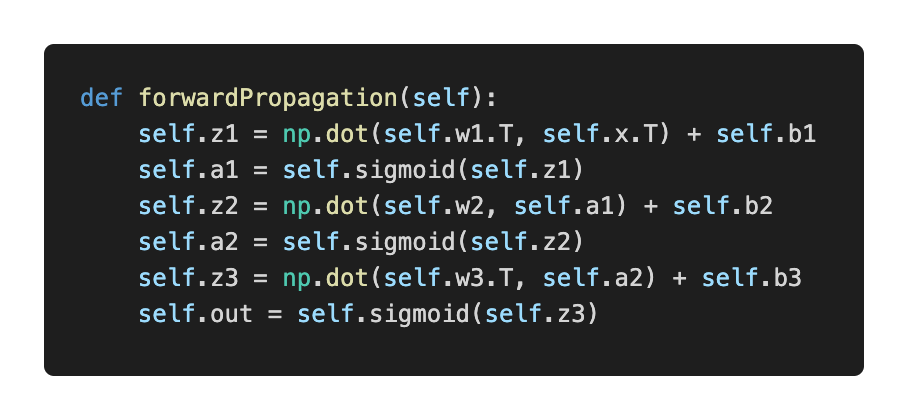
**311605014 鄧奕辰**

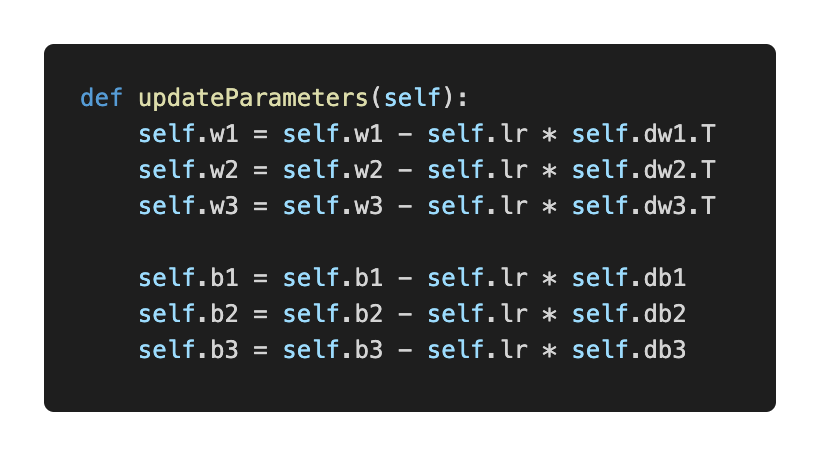
1. Introduction

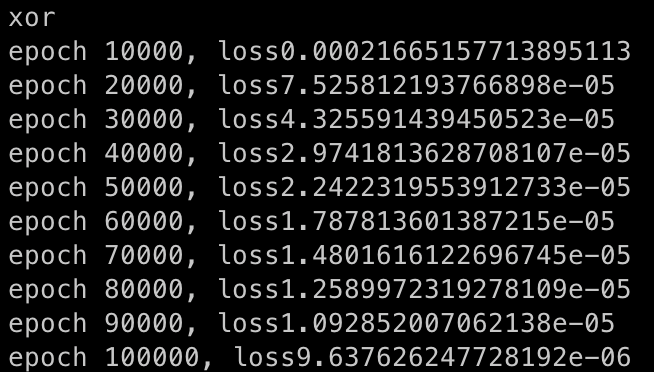
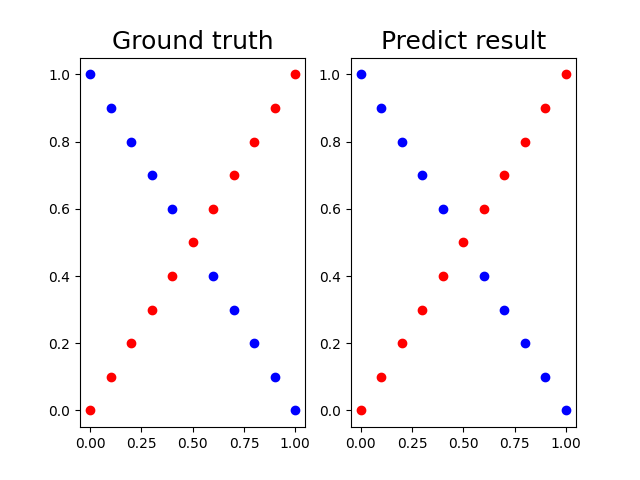
In this project, we implement the neural network with forward and backward pass using two hidden layers. Using two kinds of data, xor and linear, to classify the input data. In this report, it will show how the neural network update the weights, and the comparison of different learning rates and different numbers of hidden units.

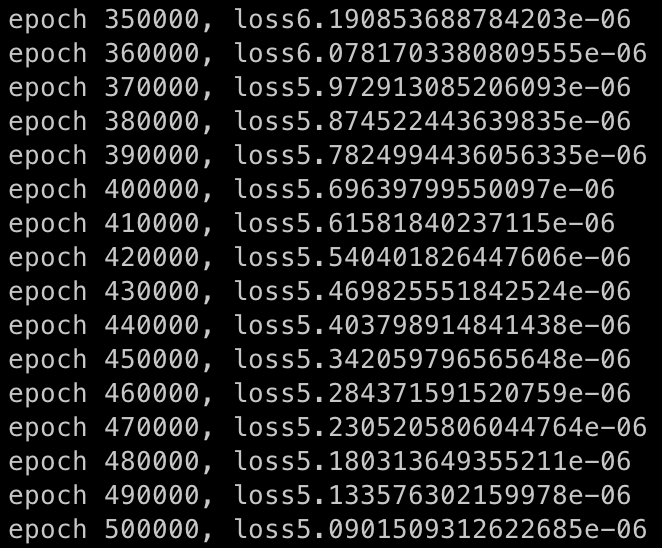
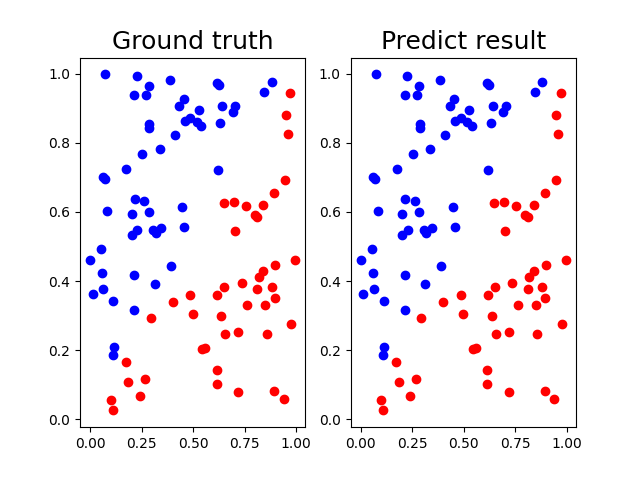
1. Dataset

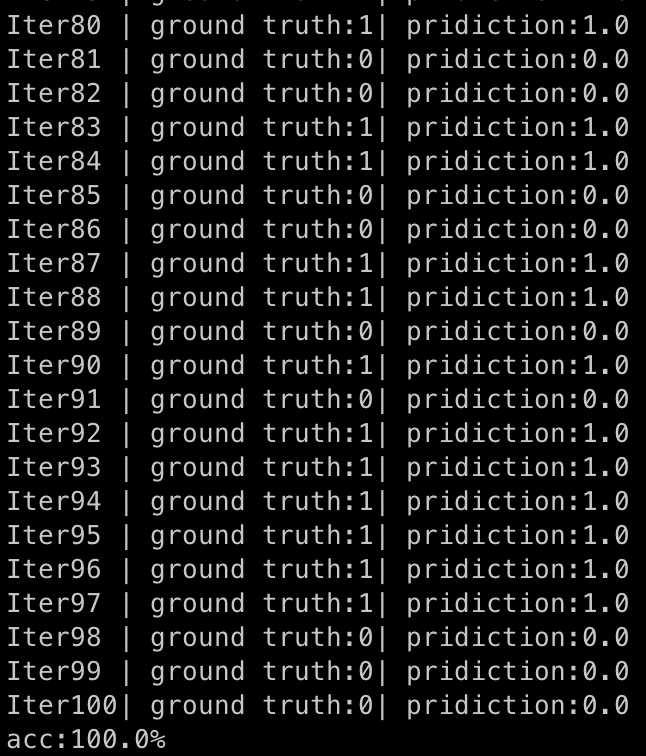
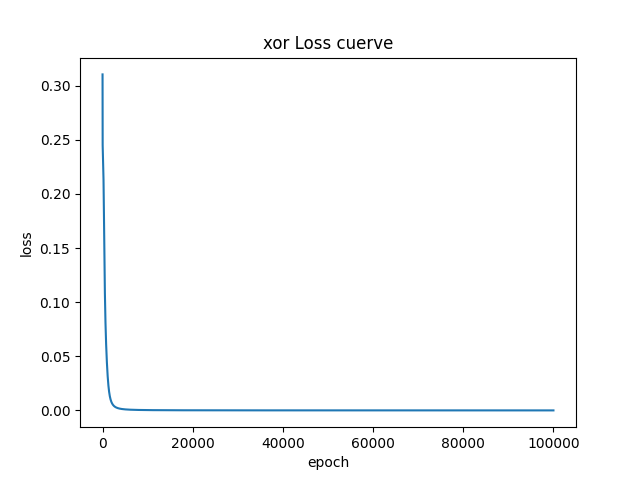
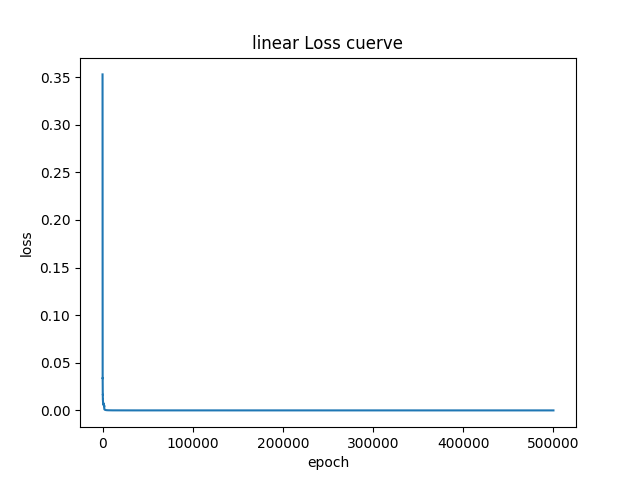
* XOR
* Linear

1. Experiment setups
2. Sigmoid functions and derivative of sigmoid function   
   Use sigmoid function as activation function.
3. Neural network  
   Neural network with two hidden layers, 2-bits input layer and 1-bit output layer. Each neuron has a weight and bias, and go through activation function. Derive as Z = σ(wX + b).  
   Loss function will be MSE loss :  
   
4. Backpropagation   
   The model initializes with random weight parameters.  
     
   forwardPropagation function implement a very classic calculation, each neurons sum the previous output and pass through a activate function “sigmoid” and calculate the loss. The loss will be passed through back propagation by chain rule. 

After the backward propagation, the gradient of weight and bias are calculated, and all the weight will be updated by learning rate times gradient.  


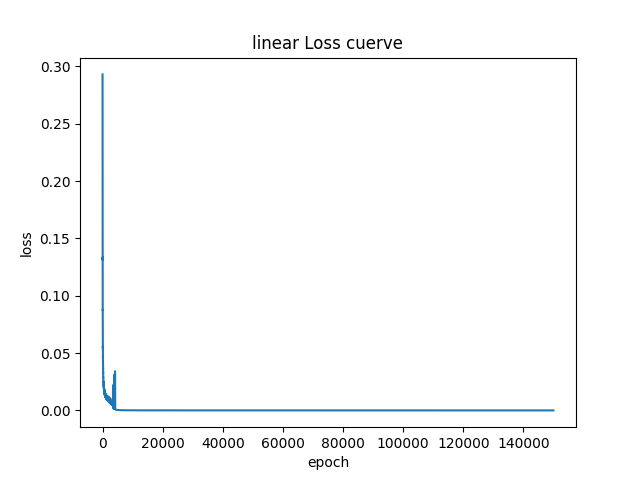
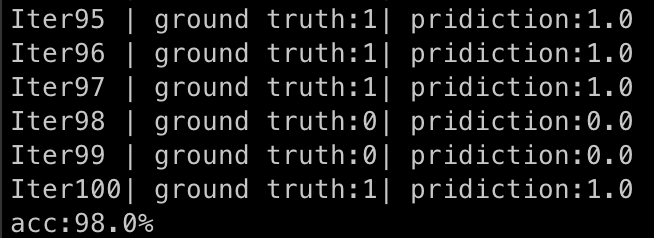
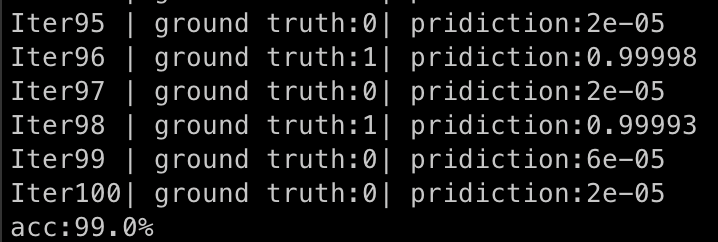
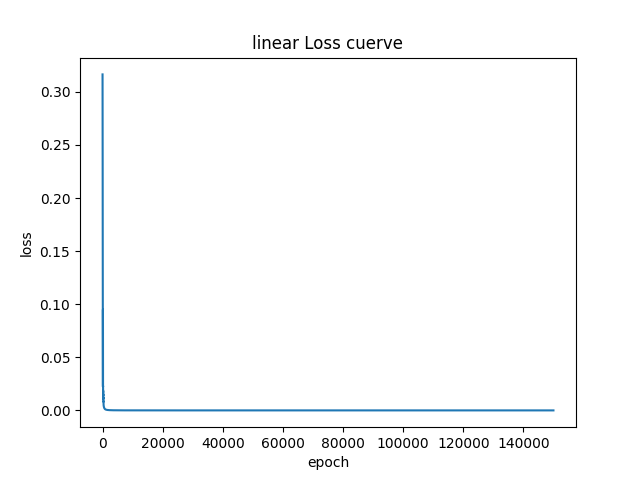
1. Results of your testing
2. Screenshot and comparison figure  
   XOR train with 100000 epochs:   
     
   

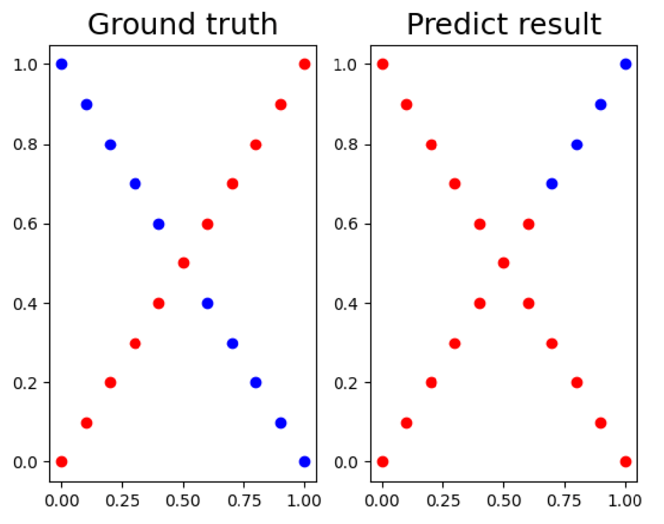
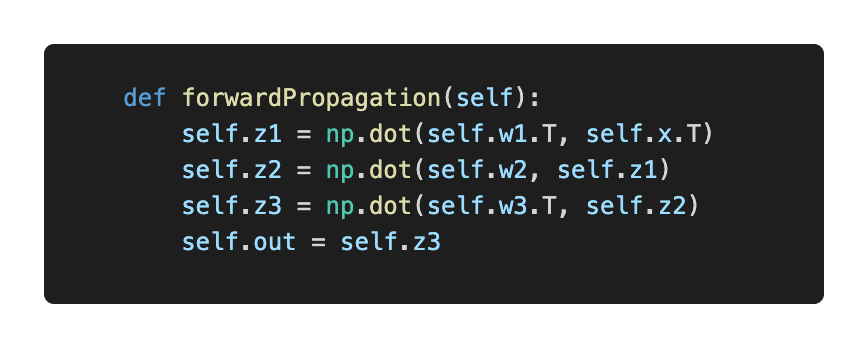
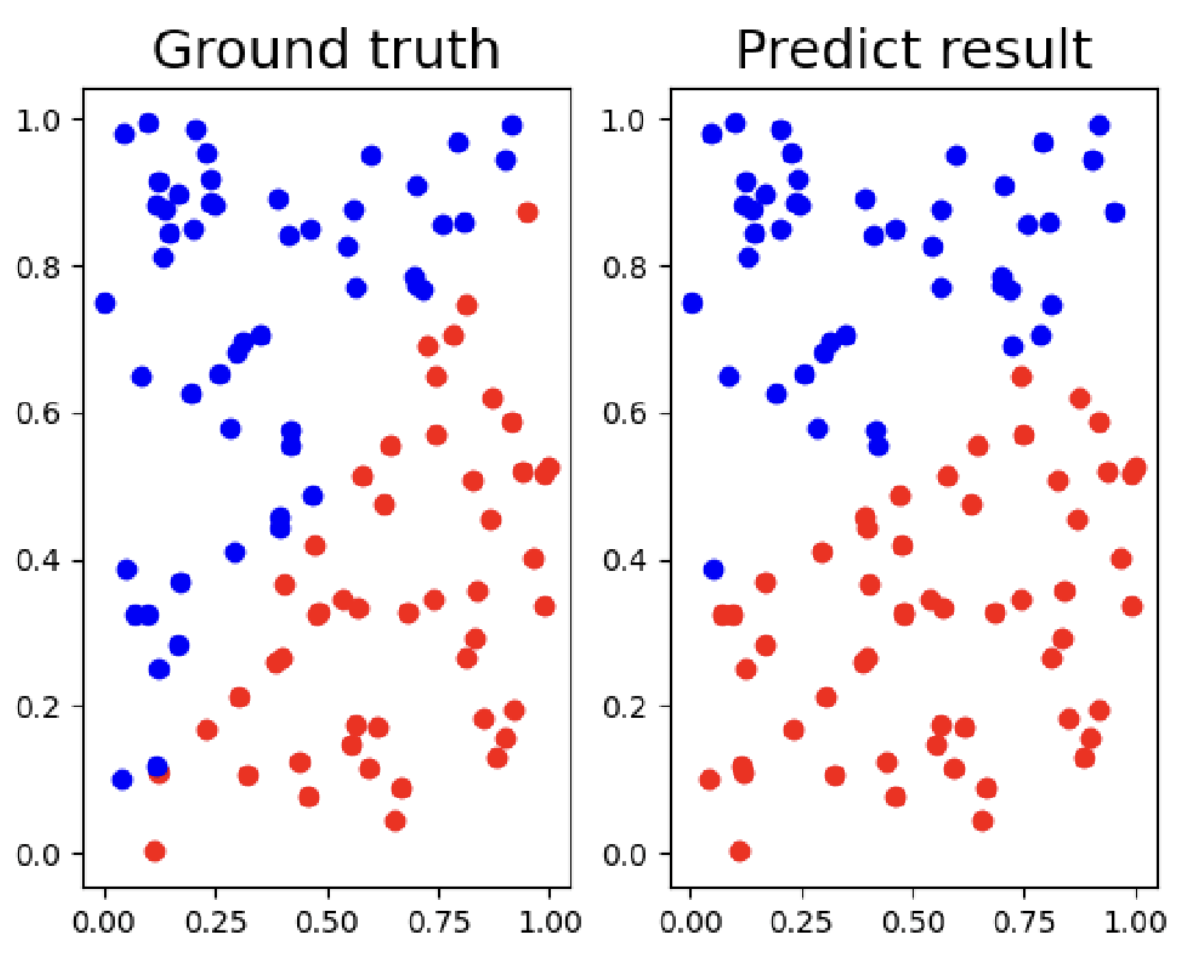
Linear train with 500000 epochs:  
  


1. Show the accuracy of your prediction   
   XOR test:  
     
   Linear test:  
   
2. Learning curve   
   XOR:  
     
   Linear:  
   
3. Discussion   
   A. Try different learning rates

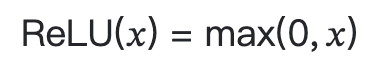
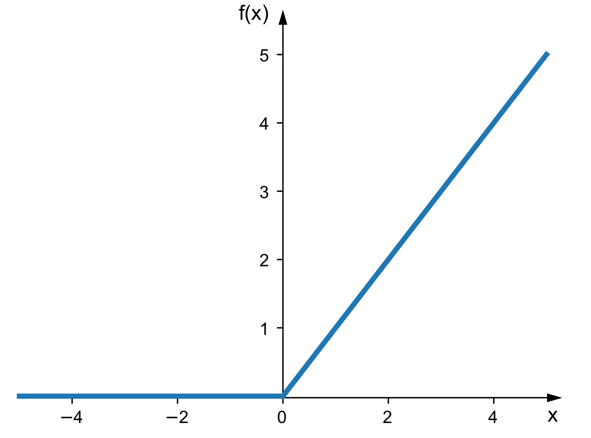
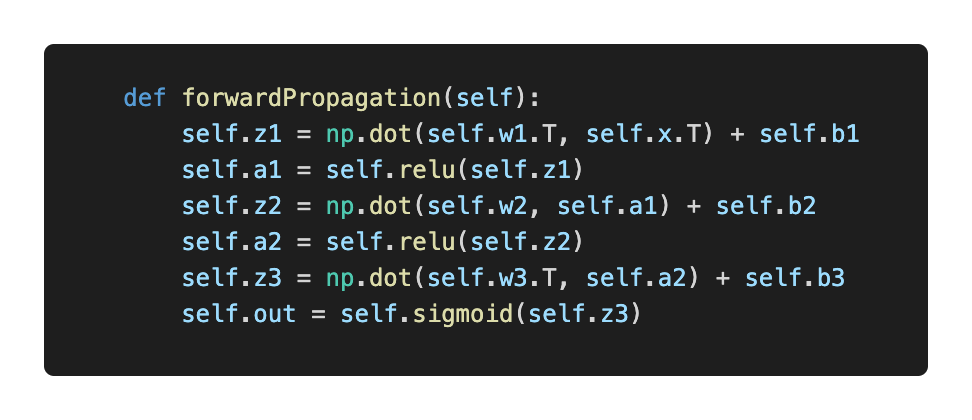
Using linear as example, I set learning rate = 0.1 and epoch = 15000. The testing accuracy achieve 100%.  
Using linear as example, I set learning rate = 0.05 and epoch = 12000. The testing accuracy only achieve 97%.  
Using linear as example, I set learning rate = 0.5 and epoch = 10000. The testing accuracy only achieve 98%, because of overshooting.  
Using linear as example, I set learning rate = 0.5 and epoch = 5000. The testing accuracy achieve 100%.

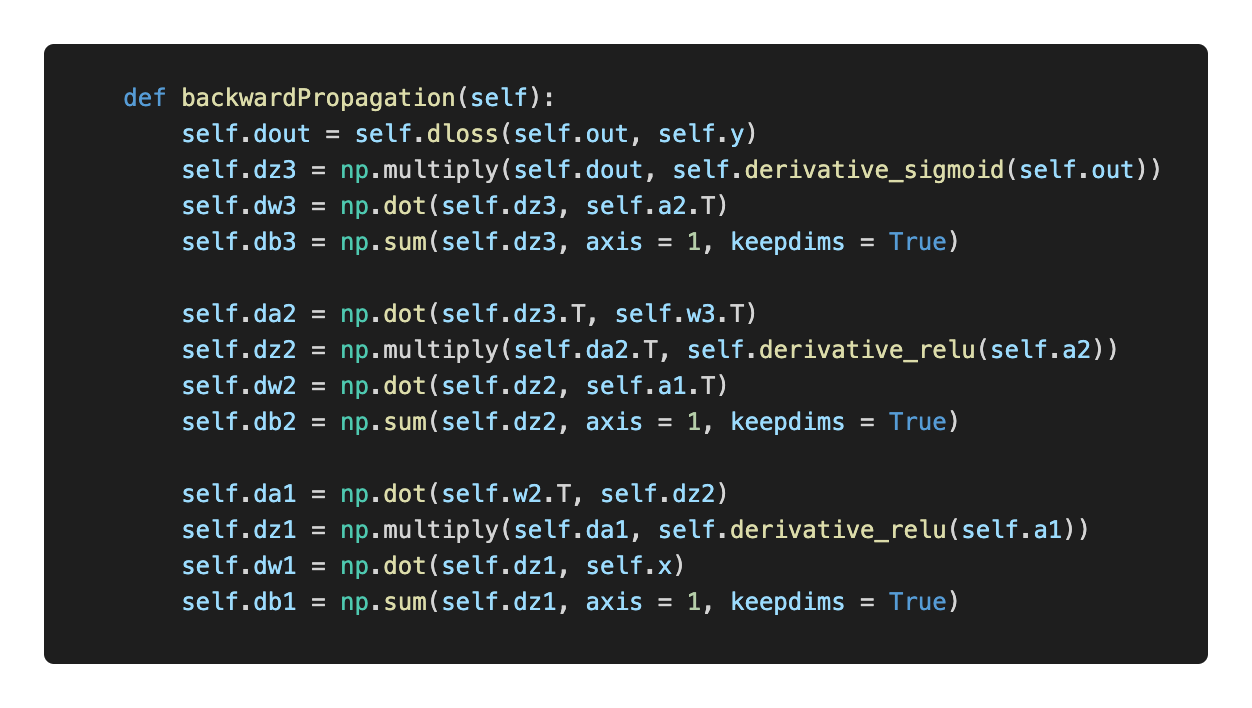
Based on the experiments conducted above, it was found that it is necessary to set the learning rate to an appropriate value and, if necessary, gradually reduce it to enable the model to converge to the global minimum.

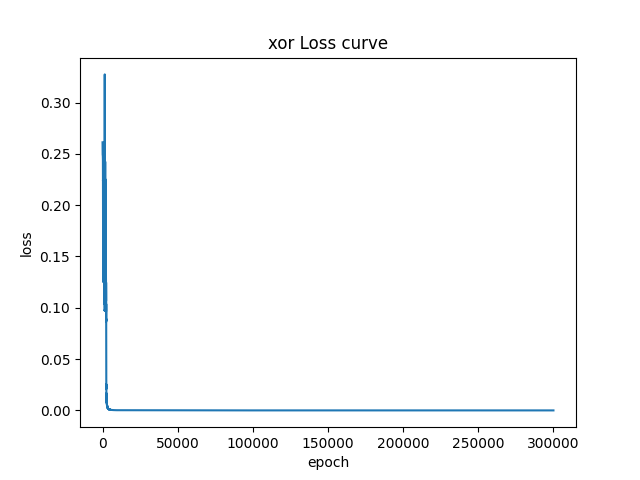
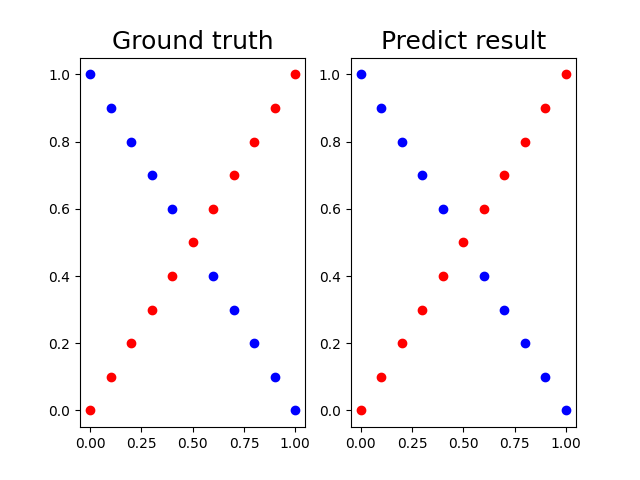
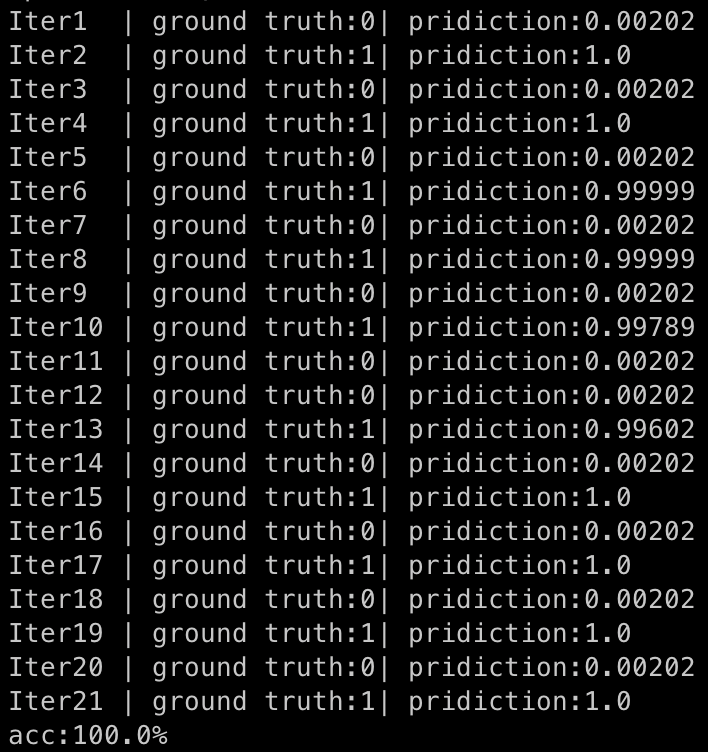
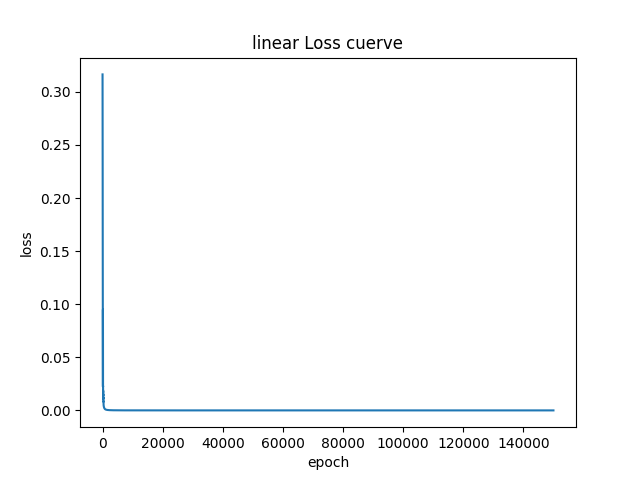
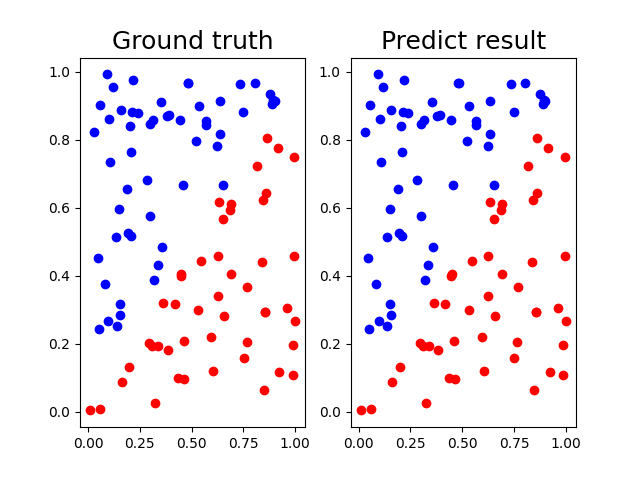
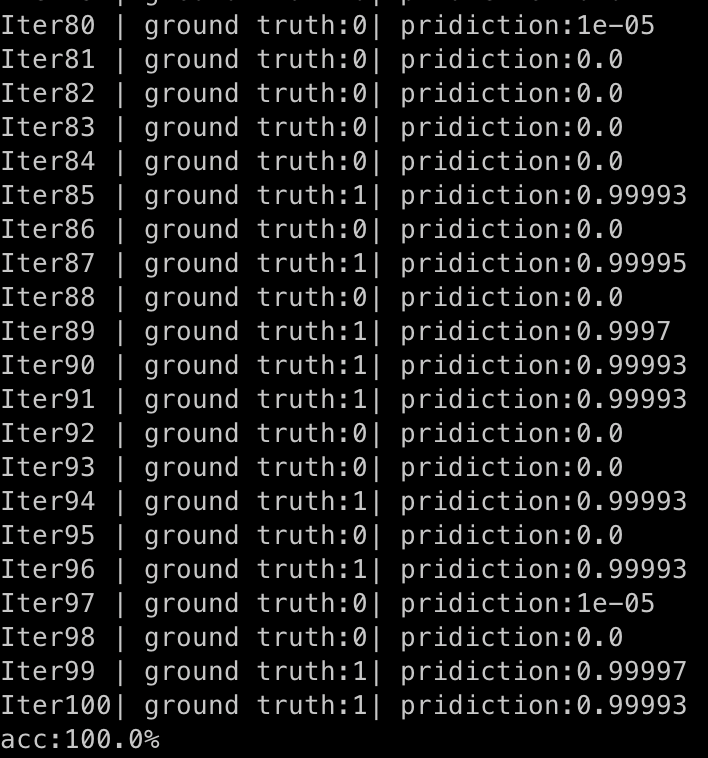
1. Try different numbers of hidden units   
   For the pictures shown above, all the hidden layers were set to 4 units.  
   Using linear as example, I set two hidden layers with 10 units, learning rate = 0.1, epoch = 15000.  
     
   Using linear as example, I set two hidden layers with 2 units, learning rate = 0.1, epoch = 15000.  
     
     
     
   To summarize the above experiments, increasing the number of neurons in the hidden layer does not necessarily lead to better performance. In fact, it can increase the difficulty of convergence. However, when dealing with more complex features and tasks in harder datasets, increasing the distribution of neurons can result in better performance.

C. Try without activation functions   
  
  
If I don’t use any activation function in a neural network, it would just become a linear regression model which can simplified as (X\*W).  
Moreover, the hidden layers would not learn any non-linear relationship in the data.  
To summarize the XOR experiments without activation, it’s impossible to draw a single line to describe the relationship of XOR, therefore the result of XOR will have a bad accuracy. However, in the linear data which has a nearly linear relationship, therefore it has an ordinary performance. 

1. Extra
2. Implement different optimizers
3. Implement different activation functions

Use ReLU activation function, shown as below. For the first and second activation function, I use ReLU activation function, then through sigmoid to output.   
  
  




* XOR train with 30000 epoch:  
    
  
* XOR testing:   
  The testing result shows that, relu activation with sigmoid activation still have a good performance.   
  
* Linear train with 25000 epoch:  
    
  
* Linear test:  
  The testing result shows that, relu activation with sigmoid activation still have a good performance.  
  

1. Implement convolutional layers