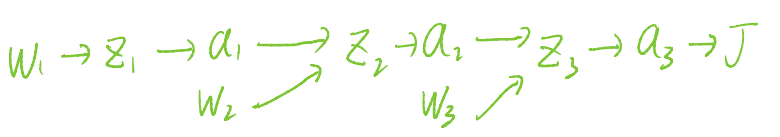
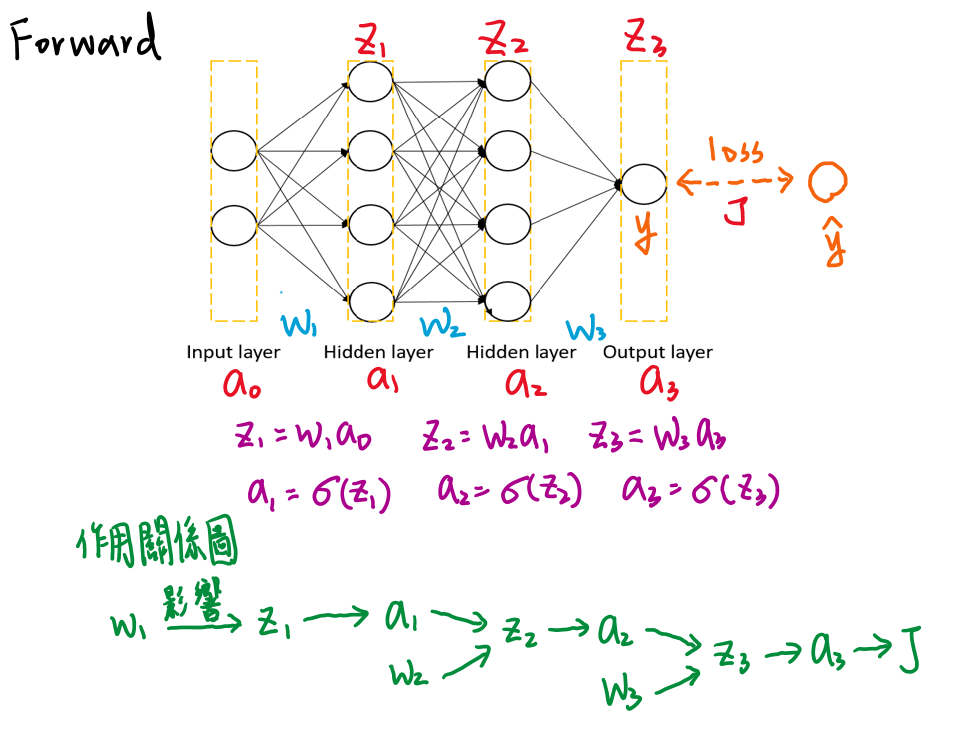
**312554012 王偉誠 Lab1 : back-propagation**

**1. Introduction**

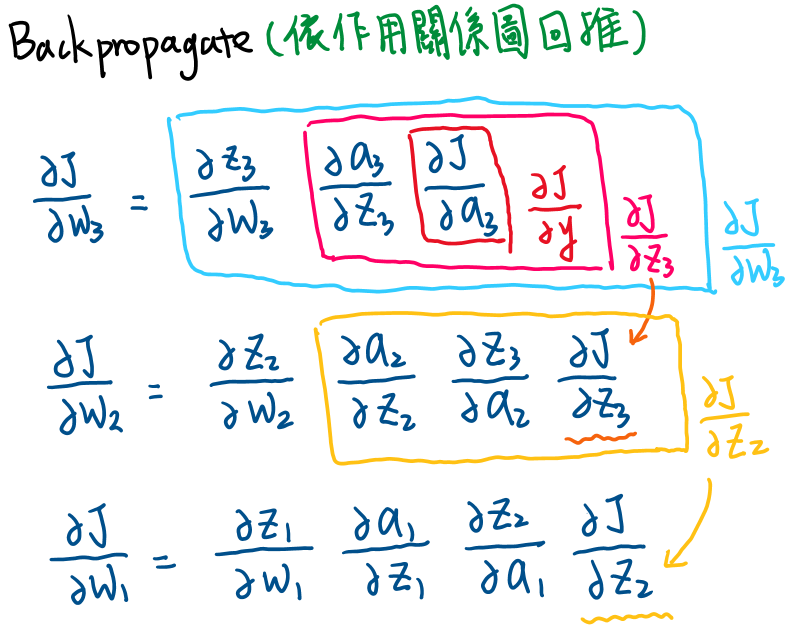
**A. LAB objective**

This LAB involves constructing a fully connected neural network with two hidden layers. The objective is to classify input data by feeding it with linear and XOR data.

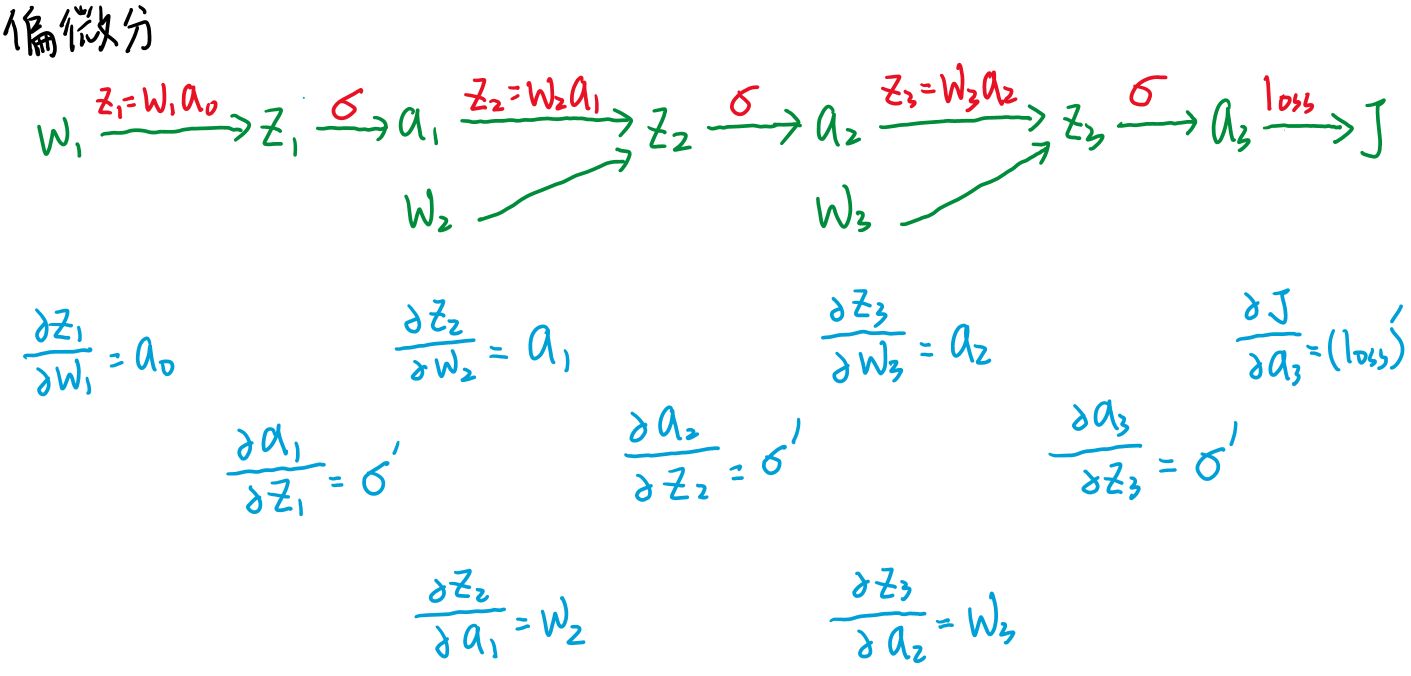
1. **Fully connected NN**
2. Based on the diagram of relationships, we can derive the forward computation equations and the diagram illustrating the functional relationships among variables.



1. Furthermore, we can utilize the chain rule to derive the backpropagation formula.



1. Compute the partial derivatives for each variable.



1. Since the input data consists of k data points, denoted as x2\*k (where x2\*k represents the calculated value of a0 as mentioned above) , the output data is represented as y1\*k ( where y1\*k represents the calculated value of a3 as mentioned above)

∴ a1n\*k = w1n\*2 \* a02\*k

　　　 a2n\*k = w2n\*n \* a1n\*k

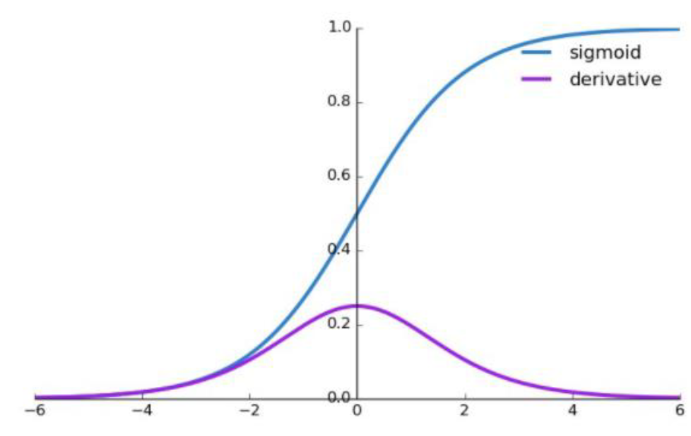
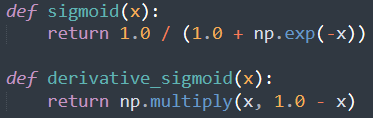
　　　　　 a31\*k = w31\*n \* a2n\*k

→ w1n\*2 , w2n\*n , w31\*n  (can get weight dimension)

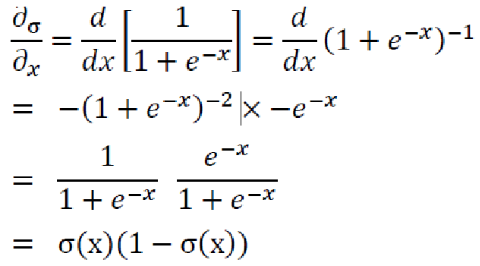
**2. Experiment setups:**

**A. Sigmoid functions**

The sigmoid function is one of the activation functions used to address non-linear problems by employing a non-linear equation. It is suitable for tackling classification problems involving datasets such as XOR, which require non-linear classification boundaries.  
Here are the graphs of the sigmoid function and its derivative:



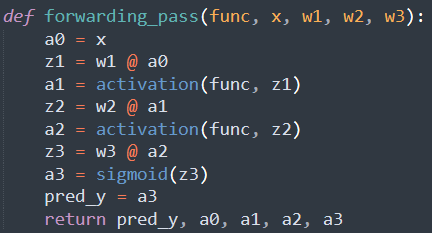
推導過程：



**B. Neural network**

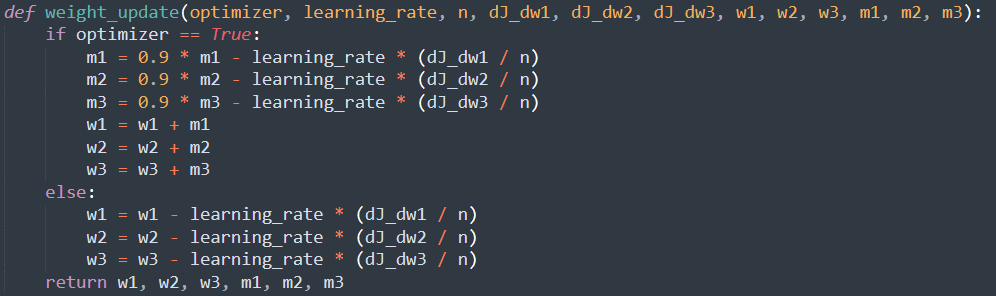
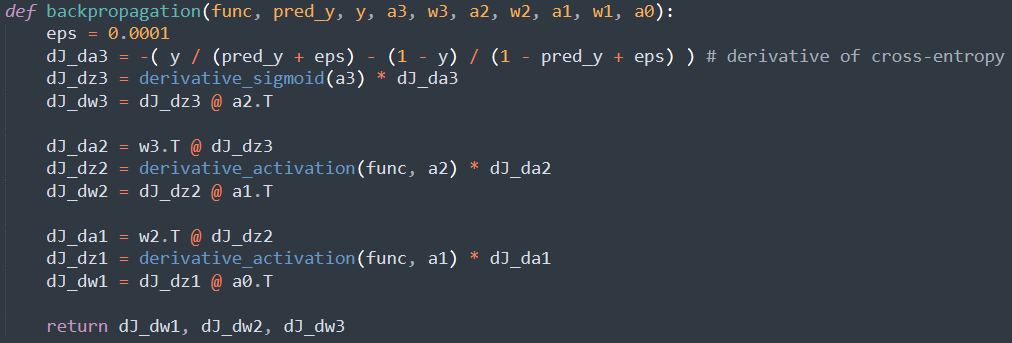
According to the fully connected NN diagram mentioned above, we have implemented two hidden layers, each consisting of 10 neurons. The operation of the NN is as follows: Initially, the weights w1, w2, and w3 are randomly initialized, and their dimensions can be determined using the calculations provided in Introduction B(4). Through the forwarding pass, we obtain the output, and then we perform backpropagation to compute the gradients and update the weights. The learning rate affects the magnitude of the weight updates. By iterating this process repeatedly, we can train the model.

The forwarding process of the NN can be implemented according to the diagram in Introduction B(1).



1. **Backpropagation**

The implementation of the backpropagation process follows the diagrams provided in Introduction B(2) and B(3), allowing us to achieve the desired outcome. In this case, the cross-entropy loss function is employed. When computing the cross entropy or its derivative, an epsilon term is incorporated to prevent division by zero or logarithm of zero errors.   
(I use the Momentum Optimizer. In the extra part, we will discuss the details in depth.)

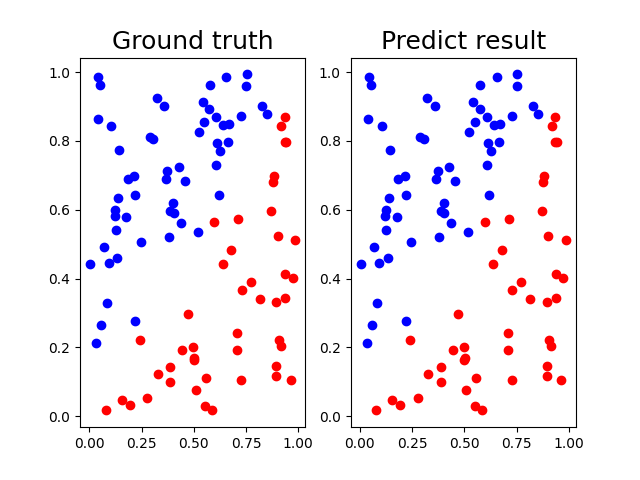


**3. Results of your testing**

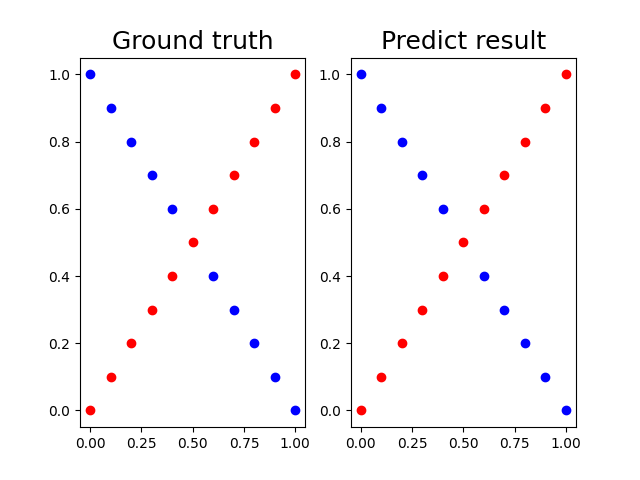
**A. Screenshot and comparison figure**

Both layers consist of 10 neurons each, learning rate = 0.1

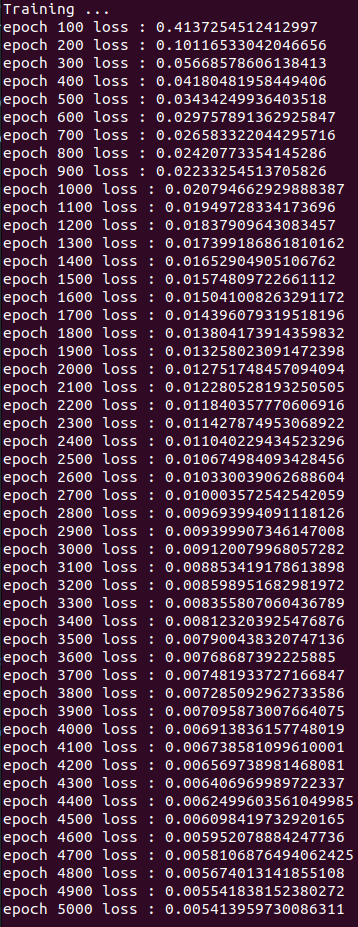
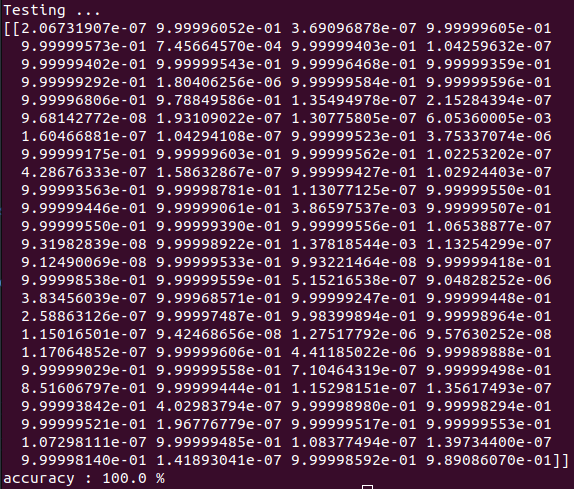
Linear data: accuracy = 100%

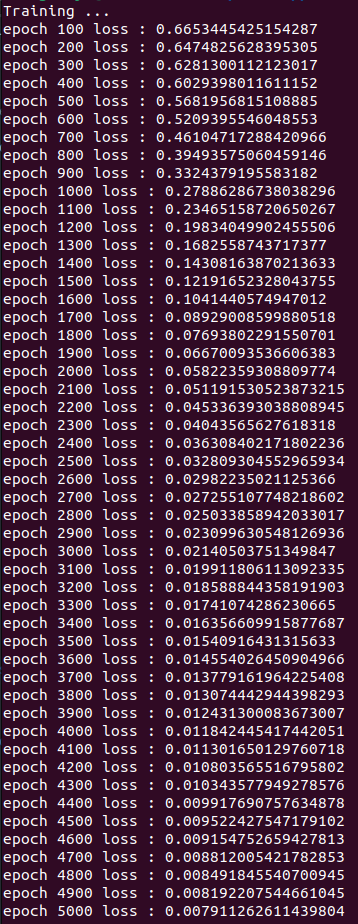
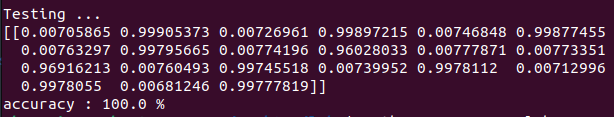


XOR data: accuracy = 100%



**B. Show the accuracy of your prediction**

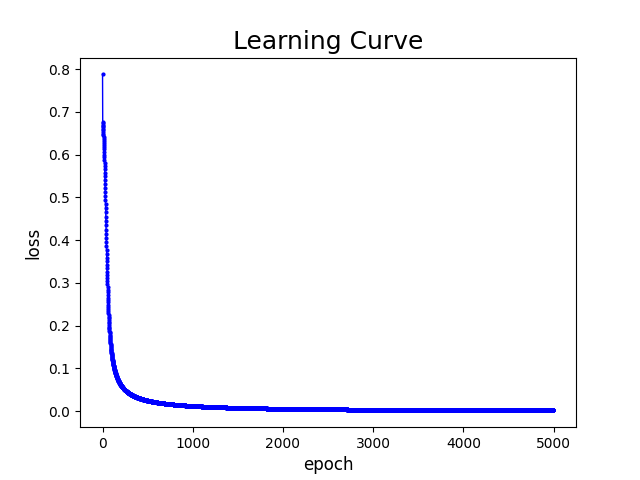
**Linear training** **Linear testing**

**XOR training** **XOR testing**

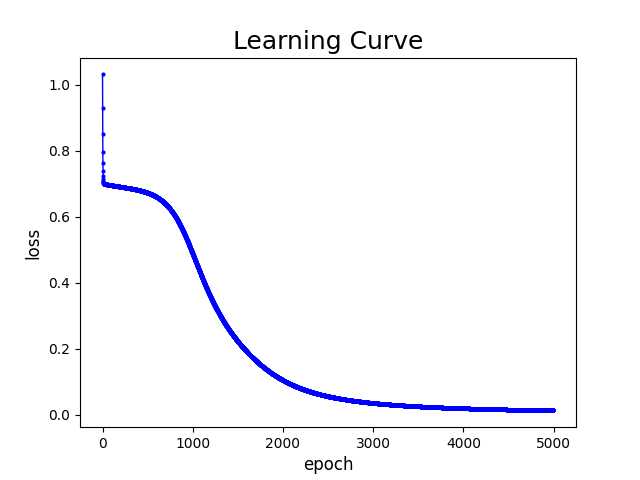
1. **Learning curve (loss, epoch curve)**

Both layers consist of 10 neurons each, learning rate = 0.1

Linear data: accuracy = 100%



XOR data: accuracy = 100%



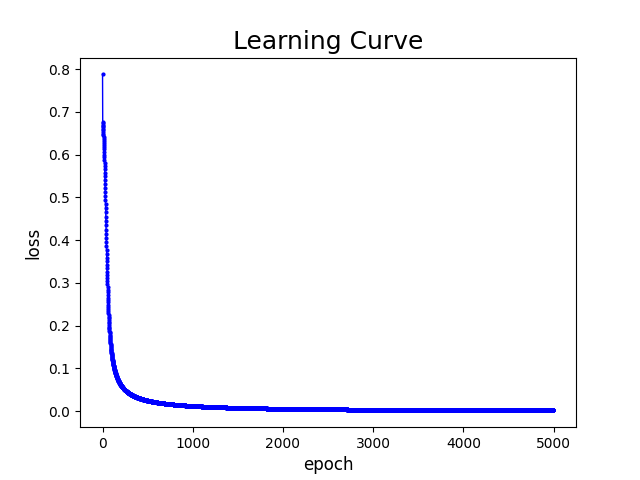
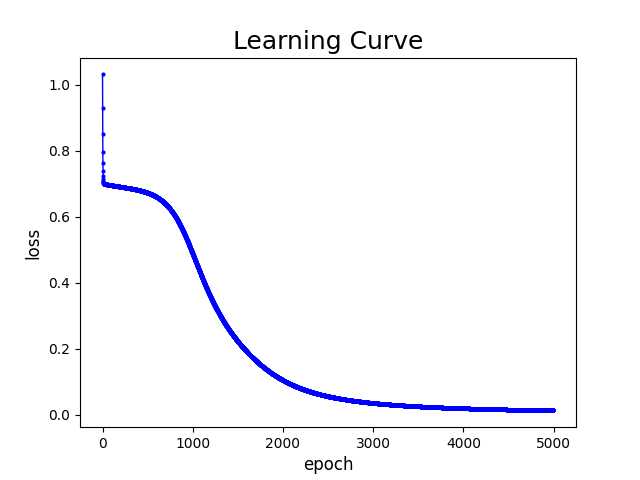
**4. Discussion**

**A. Try different learning rates**

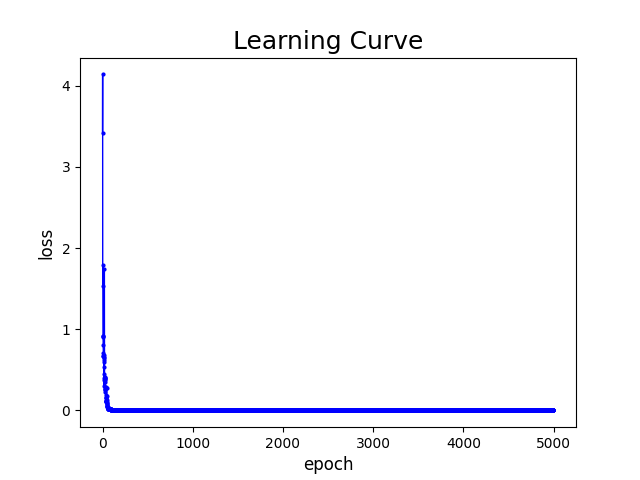
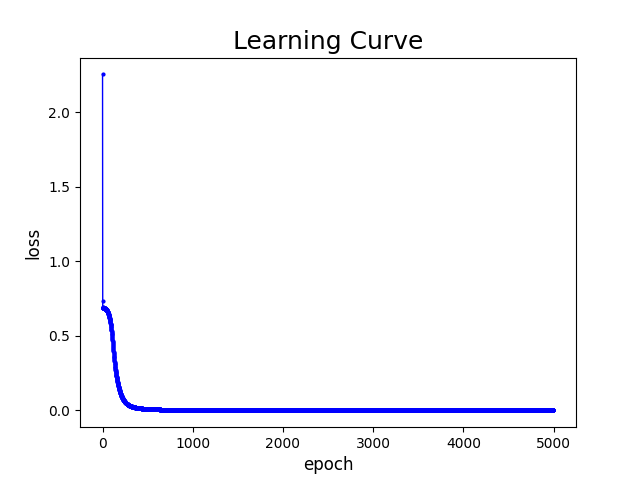
When the learning rate is larger, the weight updates have a larger magnitude, which allows the loss to converge more quickly. Below are the learning curves for learning rates of 0.1, 1, and 0.01.

**Linear data** **XOR data**

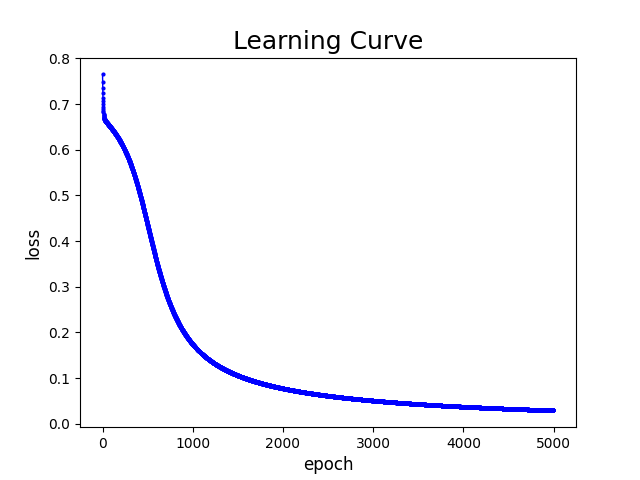
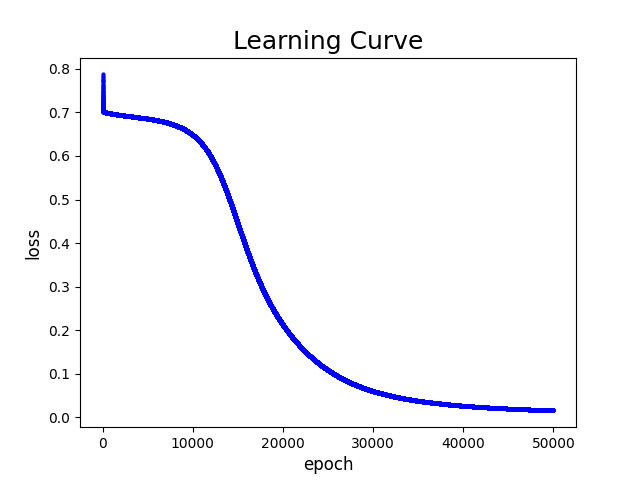
Origin: learning rate = 0.1

learning rate = 1

learning rate = 0.01

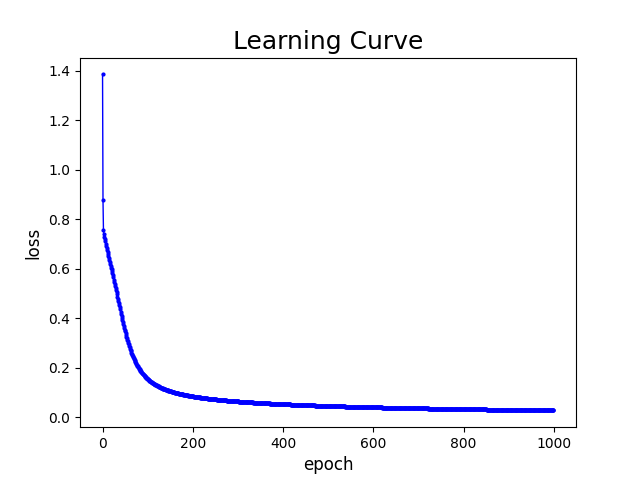
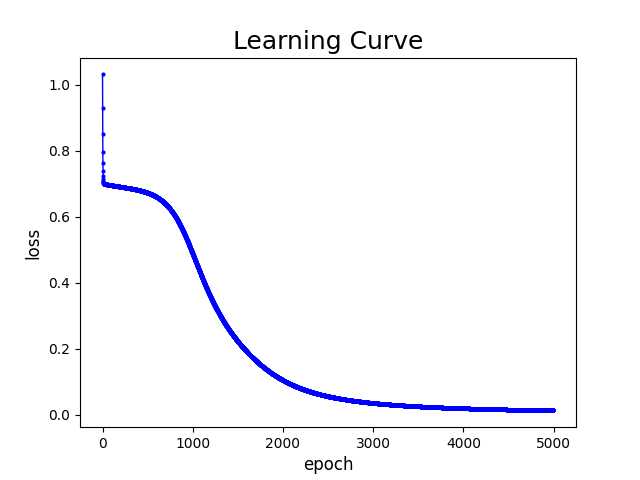
 

**B. Try different numbers of hidden units**

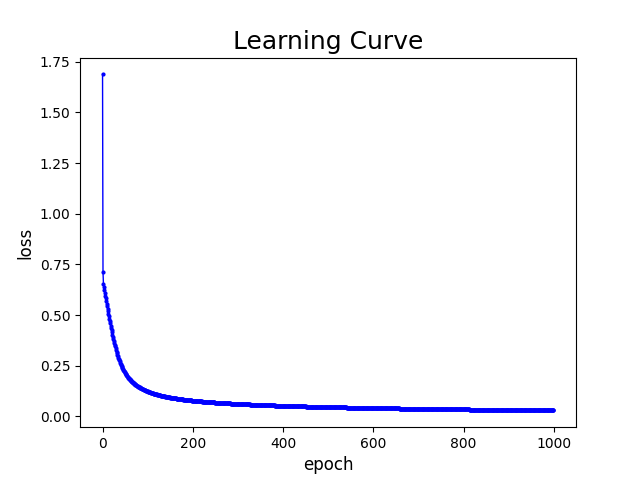
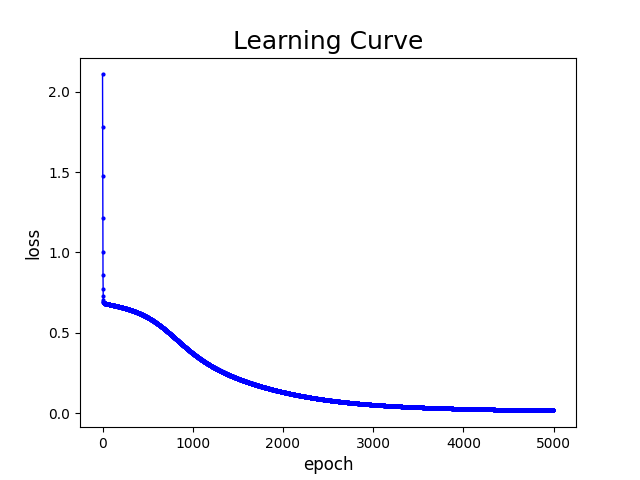
When the size of W (nxn) is changed, the total number of hidden units will also change. As n increases, the number of hidden units increases as well. It is observed that as n becomes larger, resulting in more hidden units, the loss decreases at a faster rate. Below are the learning curves for n values of 10, 15, and 5.

**Linear data** **XOR data**

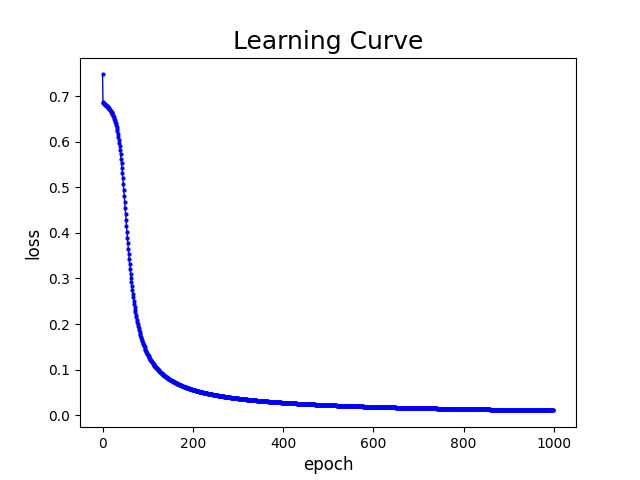
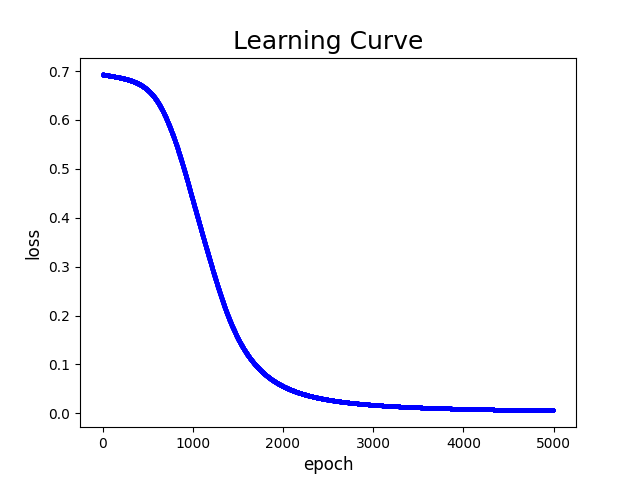
Origin: n = 10

n = 15

n = 5

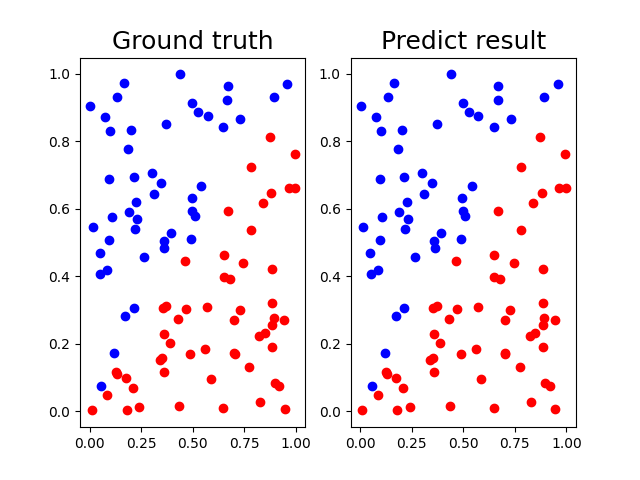
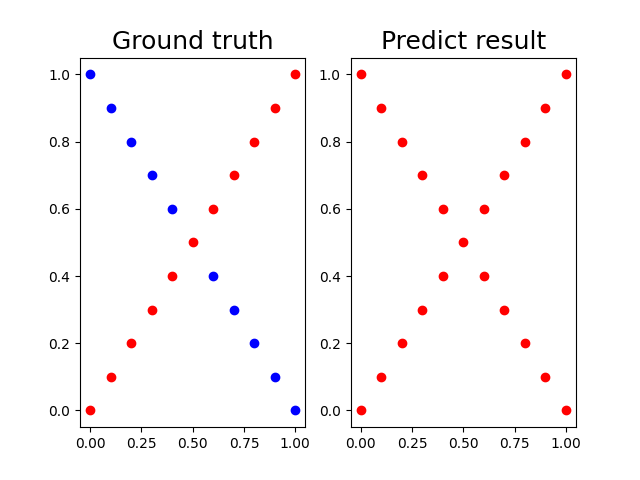
 

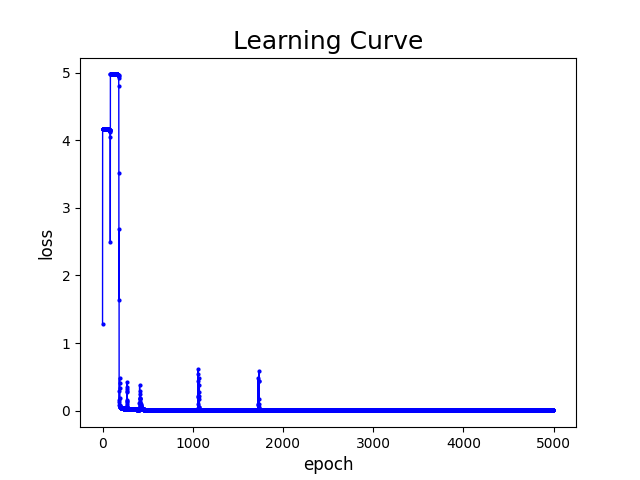
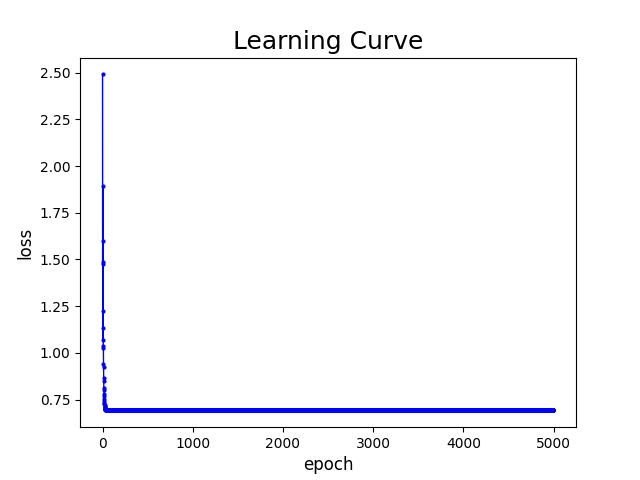
**C. Try without activation functions**

Without an activation function, the accuracy of XOR classification significantly decreases. This is because the absence of an activation function hinders the network's ability to effectively handle nonlinear data.

In the learning curve, it is evident that for linear data, using an activation function such as sigmoid leads to less pronounced loss oscillation compared to the case without an activation function. However, for XOR data, the loss appears to stagnate around 0.6920093402625231 and does not decrease further, indicating that the network struggles to properly classify the XOR data without an activation function.

**Linear data** **XOR data**

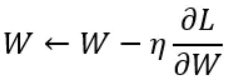
 

**5. Extra**

**A. Implement different optimizers.**

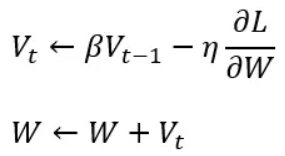
Origin: Stochastic Gradient Decent (SGD)



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Different optimizer: Momentum Optimizer

This optimizer simulates the concept of physical momentum. It adjusts the learning rate differently in the dimensions of the same direction. The learning rate increases when the direction remains the same, and decreases when the direction changes. The following diagram illustrates the update formula for the momentum optimizer. In this formula, Vt represents the update velocity, which is influenced by the previous update. If the gradient in the previous step is in the same direction as the current gradient, the update velocity increases (gradient amplification). If the directions are different, the update velocity decreases (gradient attenuation). The parameter β can be thought of as air resistance or ground friction in physical momentum. It is typically set to 0.9 to control the influence of Vt-1 on Vt.

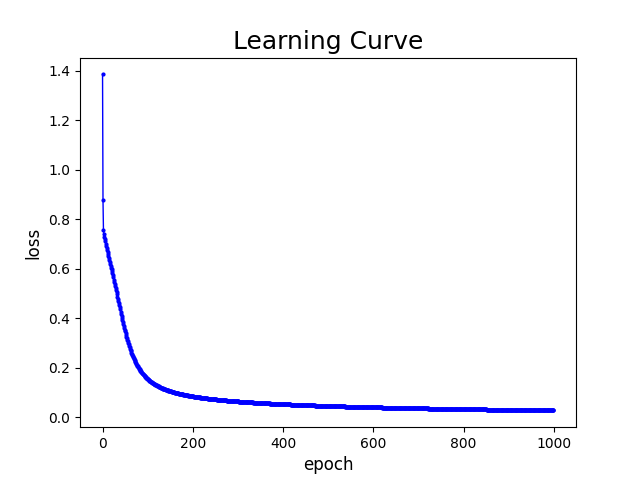
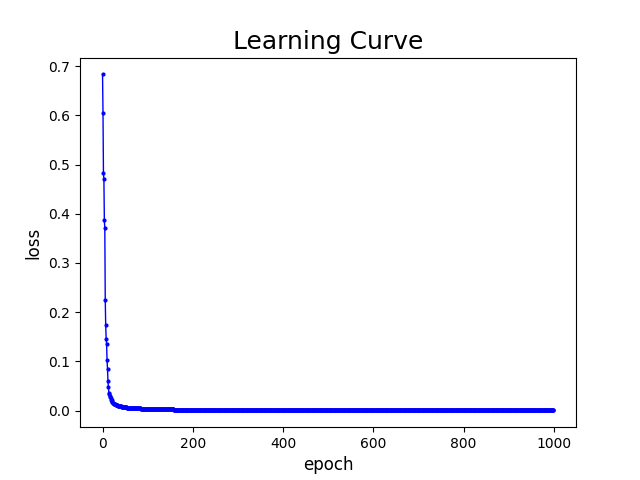


\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

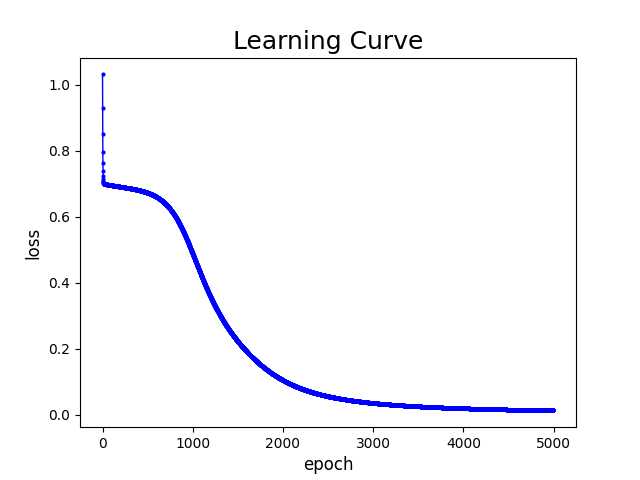
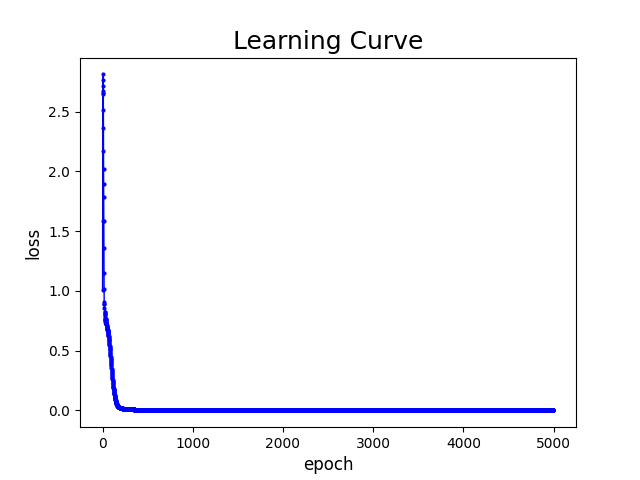
When running the neural network with two layers of 10 neurons each and a learning rate of 0.1, it is observed that the use of the Momentum Optimizer results in faster convergence of the loss compared to using Stochastic Gradient Descent (SGD).

**Stochastic Gradient Decent (SGD)** **Momentum Optimizer**

Linear data

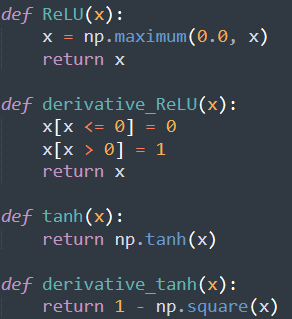
XOR data

**B. Implement different activation functions.**

In the NN diagram provided in Introduction B(1), if we replace the activation functions of a1 and a2 with different functions, we need to ensure that the output layer a3 remains unchanged as sigmoid(z3). This is because the sigmoid activation function guarantees that the output values are bounded between 0 and 1, which is crucial for maintaining consistency during the backpropagation process.

ReLU, both layers consist of 10 neurons each, using learning rate = 0.1 and learning rate = 1

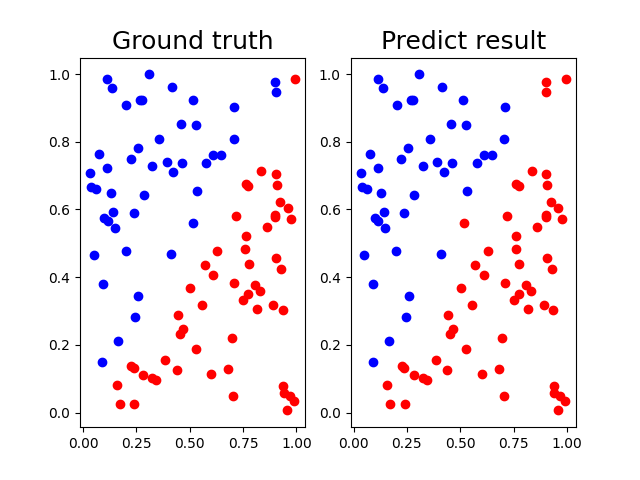
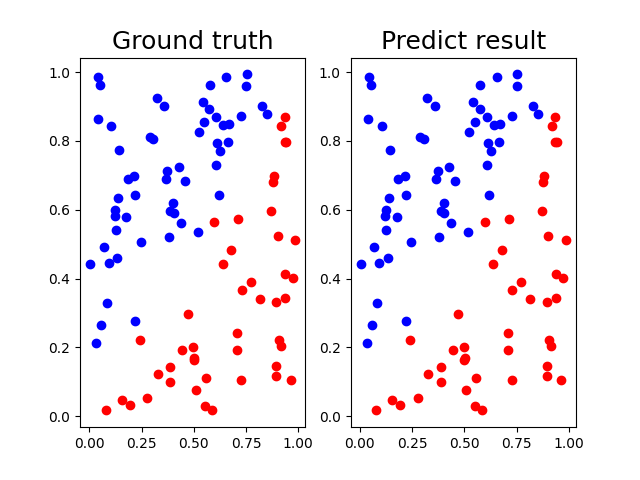


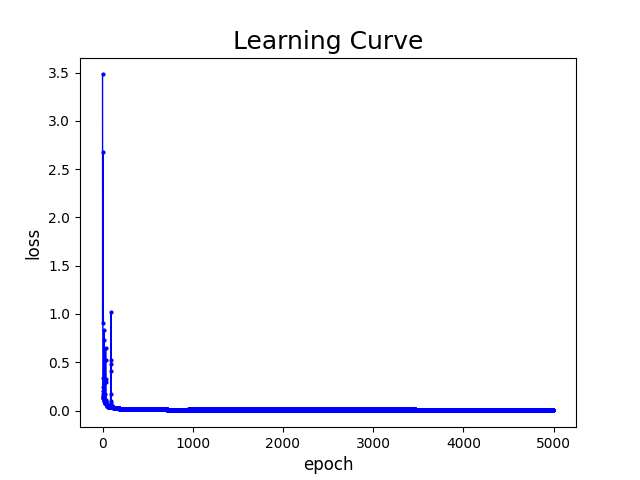
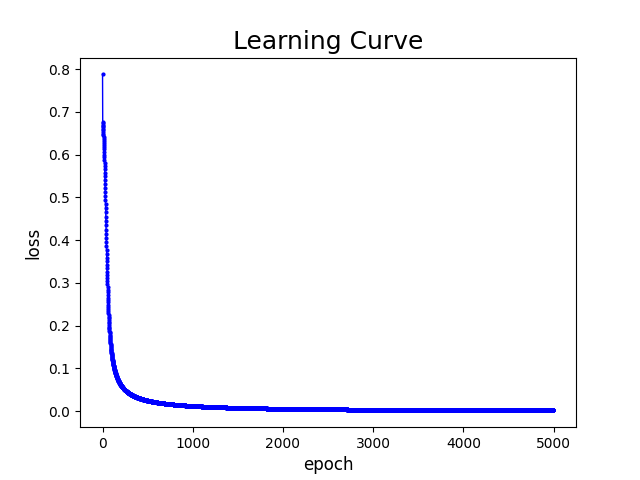
At a learning rate of 0.1, both activation functions perform reasonably well. Although ReLU exhibits some oscillation in the early stages of training, it eventually converges. However, at a learning rate of 1, the sigmoid activation function outperforms ReLU. ReLU exhibits significant oscillation, particularly when handling XOR data, and also yields high loss for linear data.

Therefore, it is crucial not to set the learning rate too high during training, as this can prevent the model from finding the optimum point.

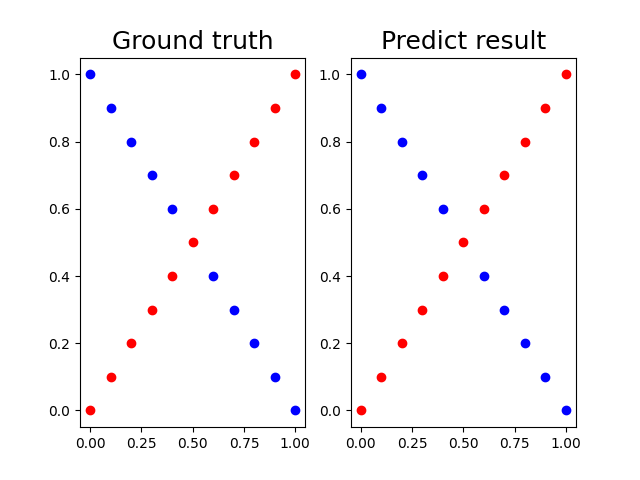
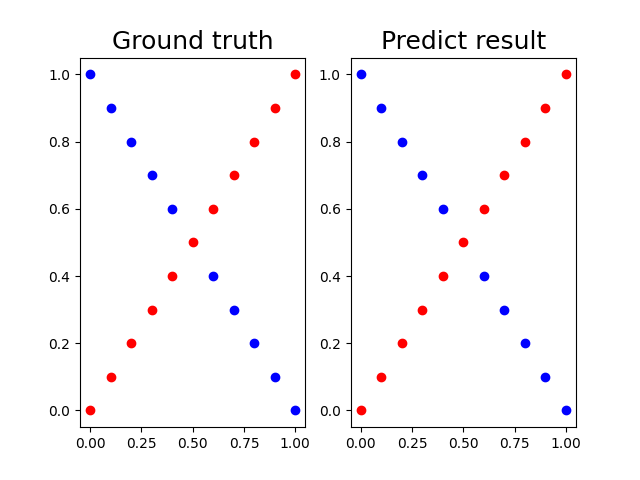
**sigmoid ReLU**

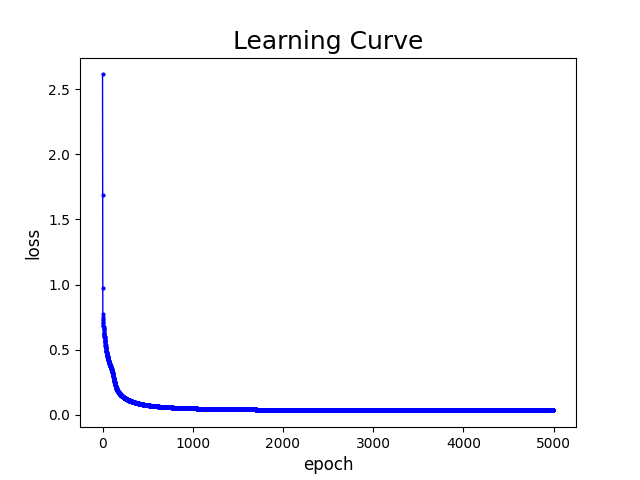
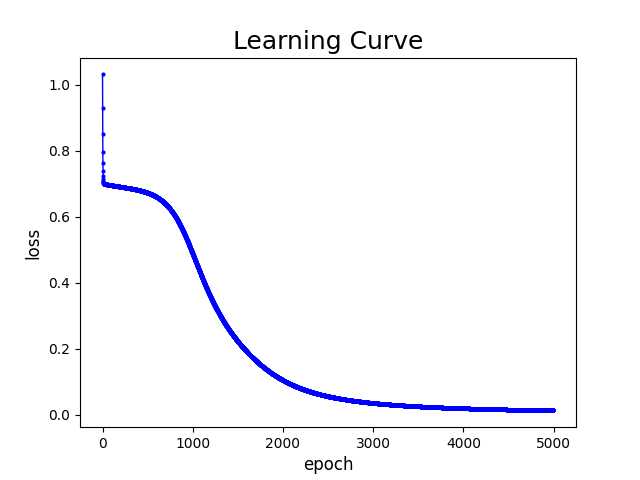
learning rate = 0.1 on linear data:





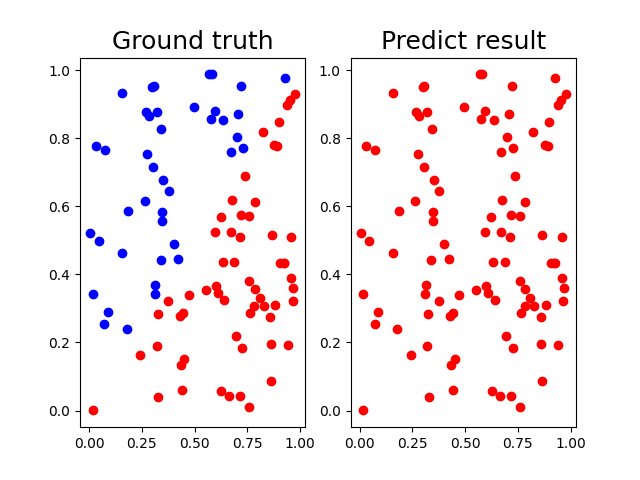
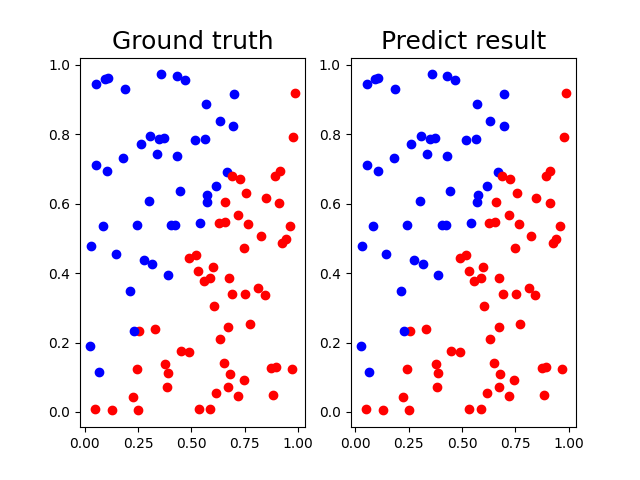
learning rate = 0.1 on XOR data:

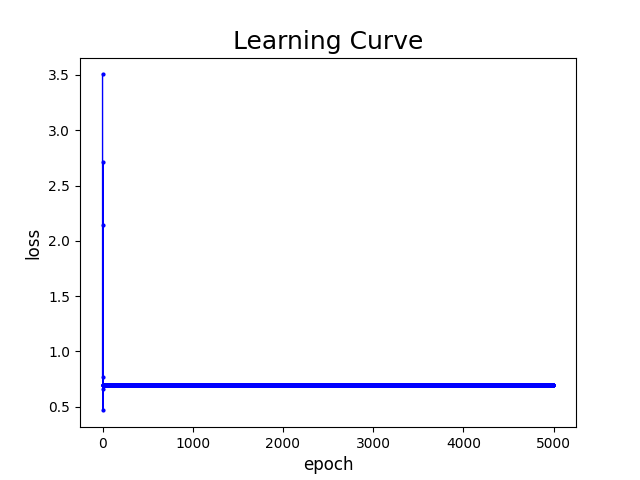
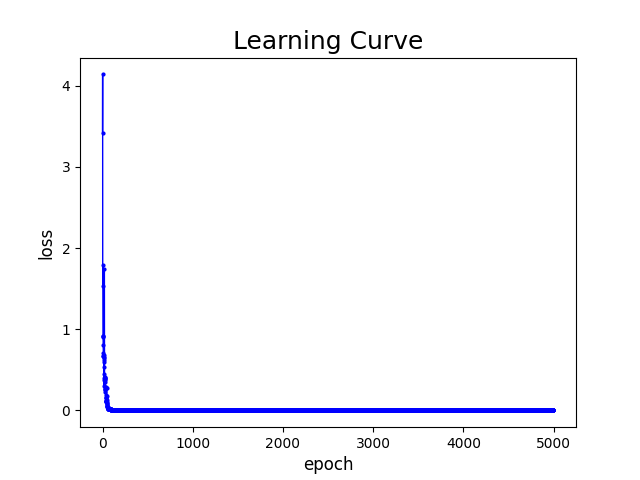




**sigmoid ReLU**

learning rate = 1 on linear data:





learning rate = 1 on XOR data:

