DLP_LAB2_310605001_王盈驊

tags: `deep learning`

Introduction

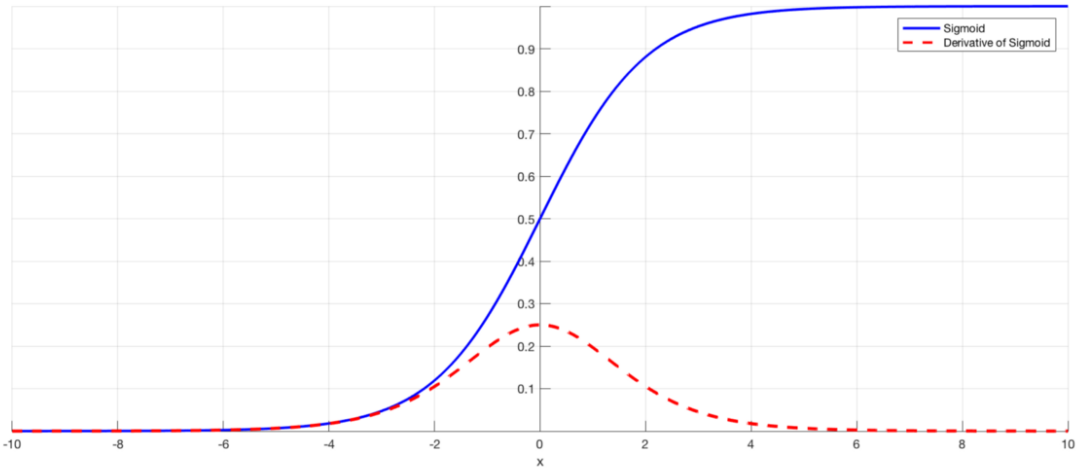
This work will implement a two hidden layers neural network with forwarding pass and back propagation only use Numpy and other standard libraries, the deep learning framework is not allowed to use in this homework. In the part of forward pass, we use sigmoid function to be our active function, In part of back propagation, we use chain rule and gradient descent to complete the work. The details will be introduced below.

Experiment setups

A. Sigmoid functions

The sigmoid functions as activate function on neural network, this function can conquer nonlinear problems, likes XOR. The derivation formula and graph of sigmoid are shown below.

Since all equations needed for the derivative of the Sigmoid function are already found during the feedforward step it saves us a ton of computations, and this is one of the benefits of using the Sigmoid function as an activation function in the layers of a neural network.



B. Neural network

Implementation Details:

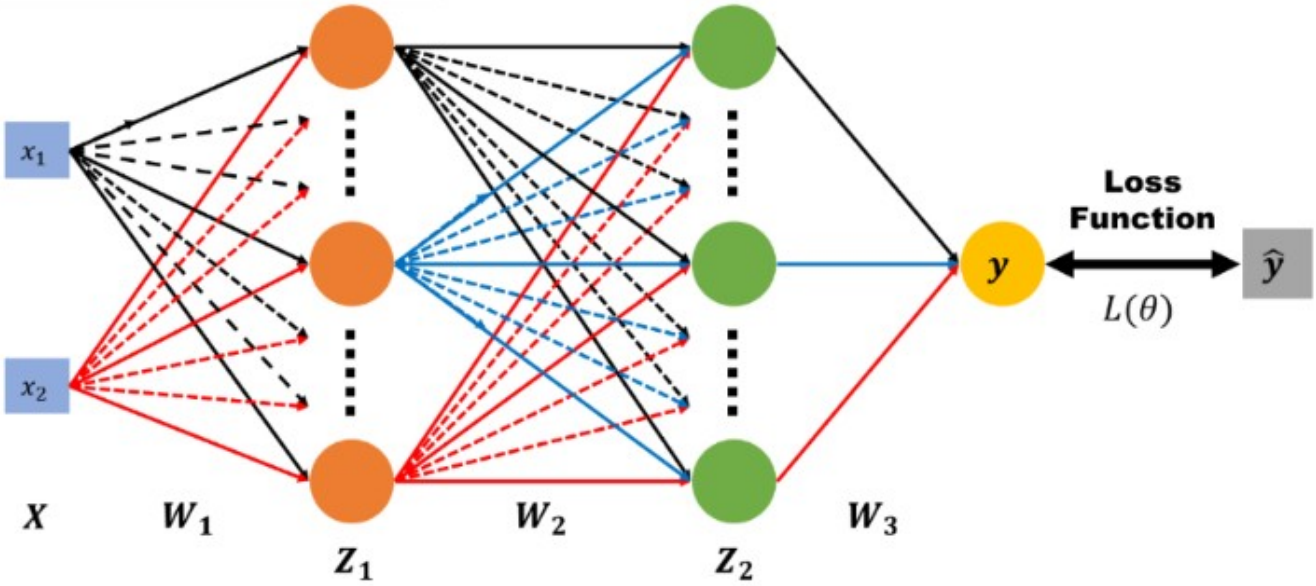


Figure 2. Forward pass

- In the figure 2, we use the following definitions for the notations:
  1. x_1, x_2 : *nerual network inputs*
  2. X : $[x_1, x_2]$
  3. y : *nerual network outputs*
  4. $\hat{y}$: *ground truth*
  5. L(θ) : *loss function*
  6. W_1, W_2, W_3 : *weight matrix of network layers*

- Here are the computations represented:
  $\boldsymbol{Z_1 = \sigma(XW_1)}$ $\boldsymbol{Z_2 = \sigma(Z_1W_2)}$ $\boldsymbol{y = \sigma(Z_2W_3)}$

- In the equations, the $\boldsymbol{\sigma}$ is sigmoid function that refers to the special case of the **logistic** function and defined by the formula:
  $$\boldsymbol{\sigma}(\mathrm{x}) = \frac{1}{1+e^{-x}}$$

Neural networks are set of algorithms inspired by the functioning of human brian.

- ==Input units== : The activity of the input units represents the raw information that is fed into the network. this also called input layer.
- ==Hidden units== : The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units. this also called hidden layer.
- ==Output units== : The behaviour of the output units depends on the activity of the hidden units and the weights between the hidden and output units. this also called output layer.
- ==Active function== : The purpose of the activation function is to introduce non-linearity into the output of a neuron. This is important because most real world data is nonlinear. Every activation

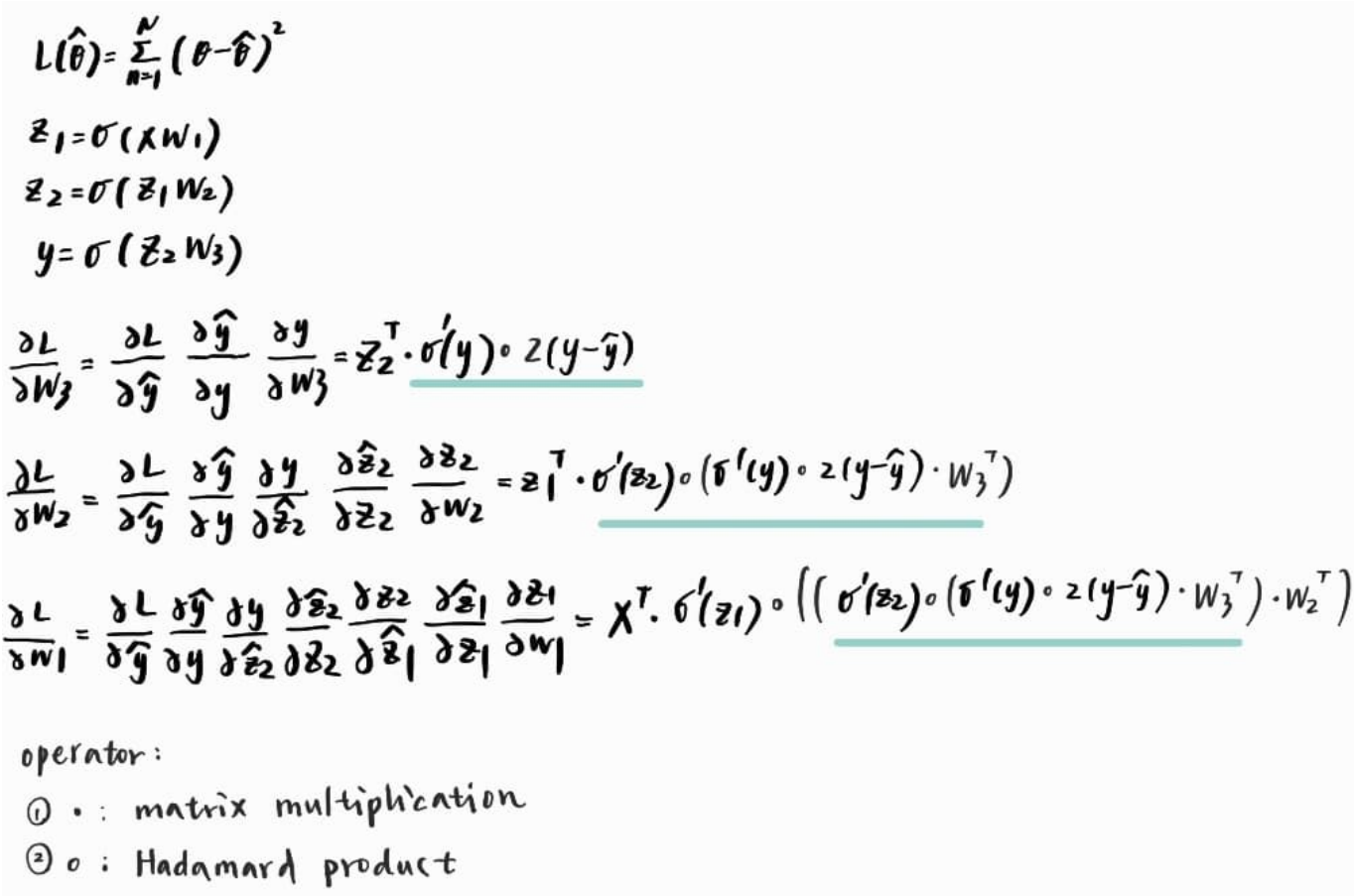
function (or non-linearity) takes a single number and performs a certain fixed mathematical operation on it. There are several activation functions you may encounter in practice: Sigmod, tanh, ReLU...

C. Backpropagation

- Loss function: In this work, we use Mean-Square Error as loss function.
- gradient descent: There's an important parameter η which scales the gradient and thus controls the step size. In machine learning, it is called learning rate and have a strong influence on performance

$$p_{n+1} = p_n - \eta \nabla f(p_n)$$

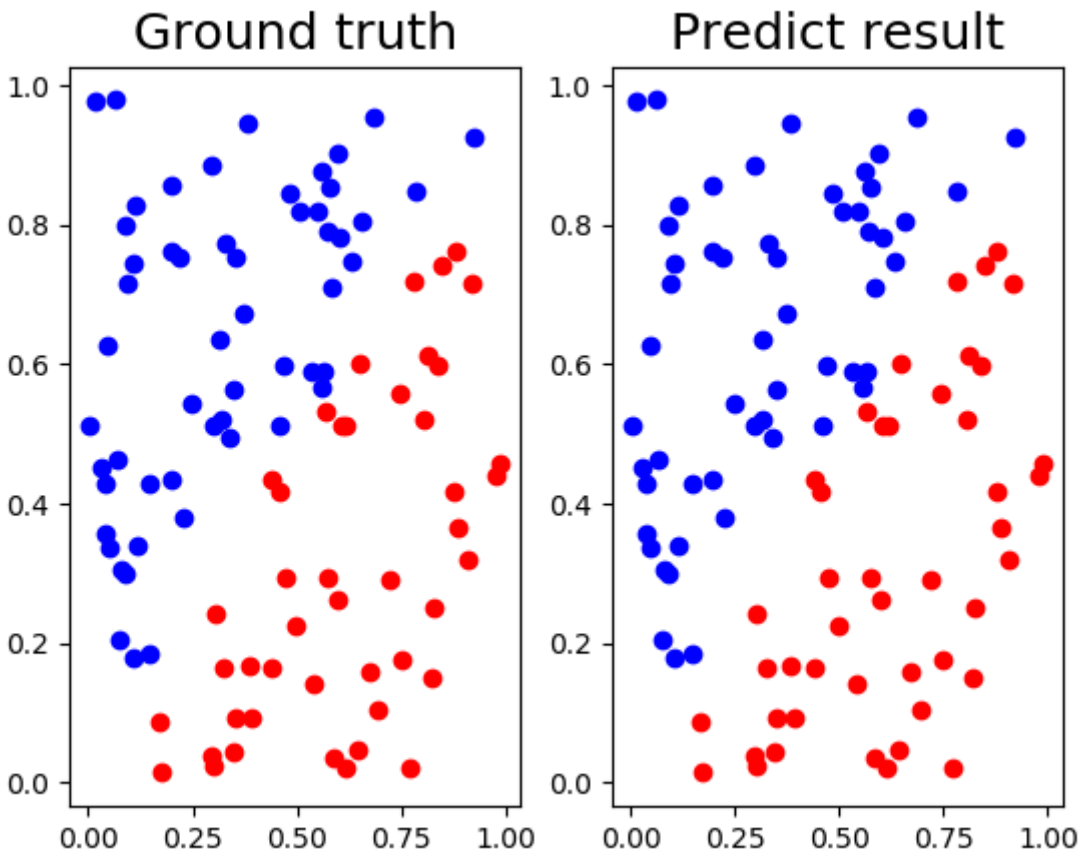
- chain rule: We use chain rule to get gradient of every weight and scale with learning rate to modify weight.

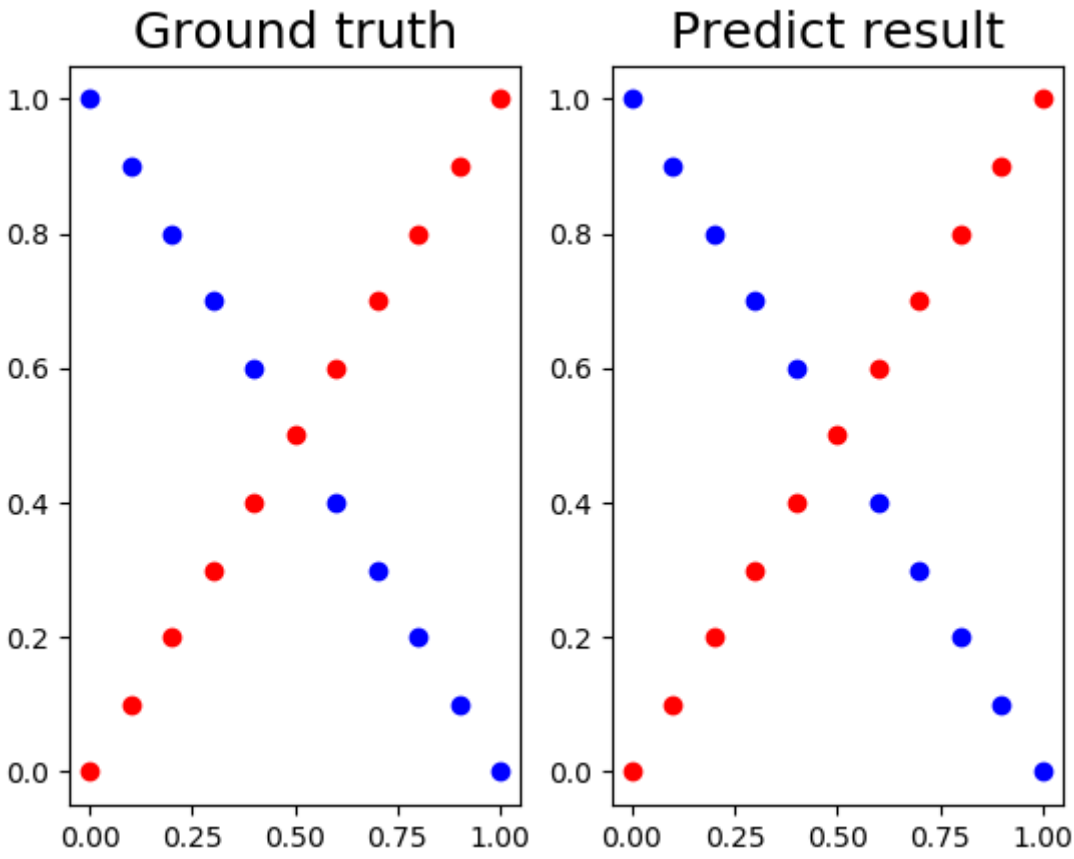


Results of your testing

A. Screenshot and comparison figure

1. linear

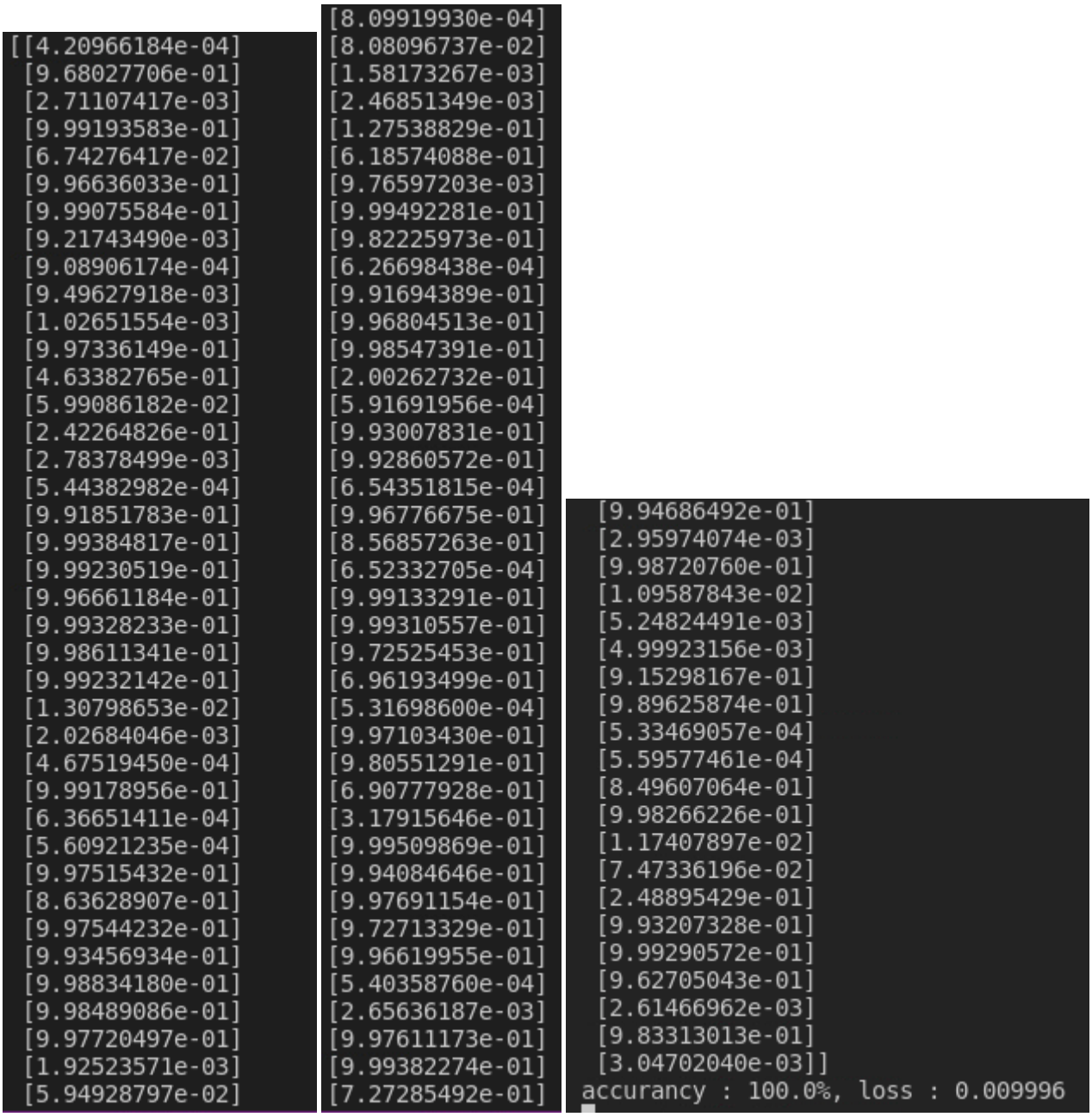




2. XOR

B. Show the accuracy of your prediction

1. ==linear data== : accuracy 100%

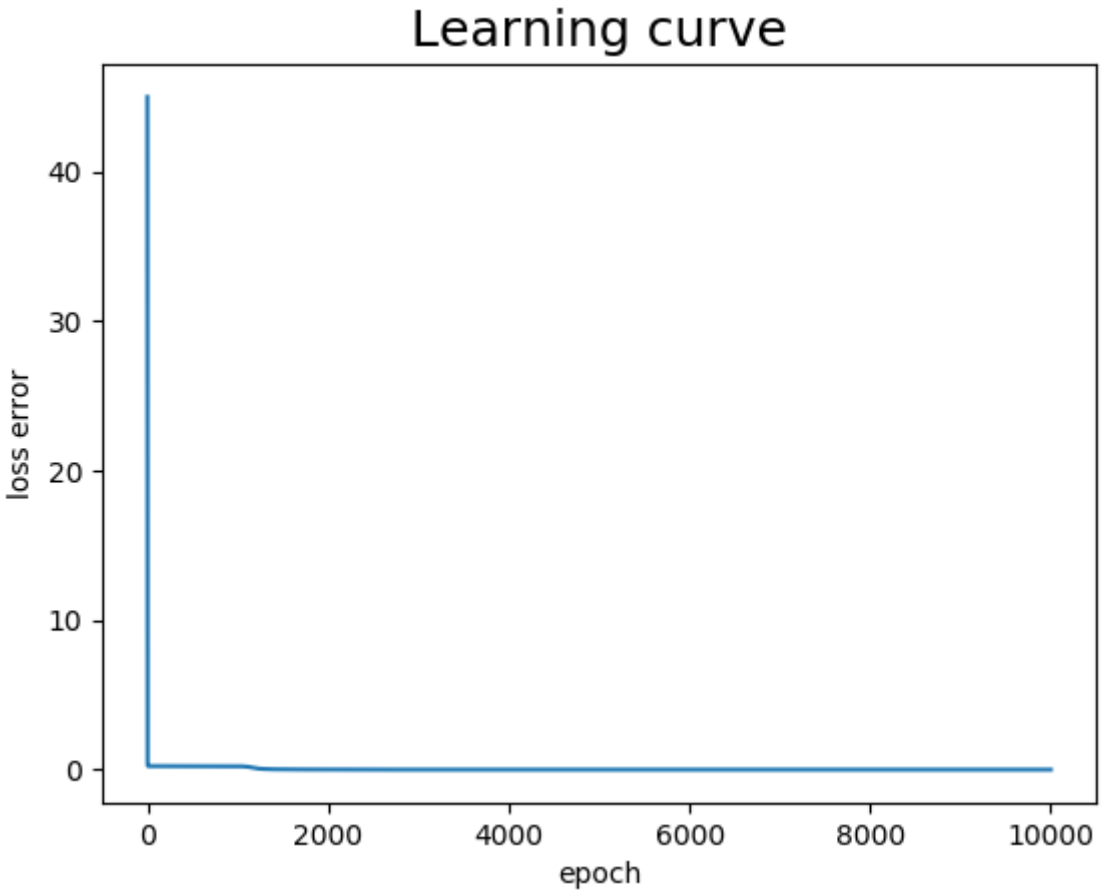


2. ==XOR data== : accuracy 100%

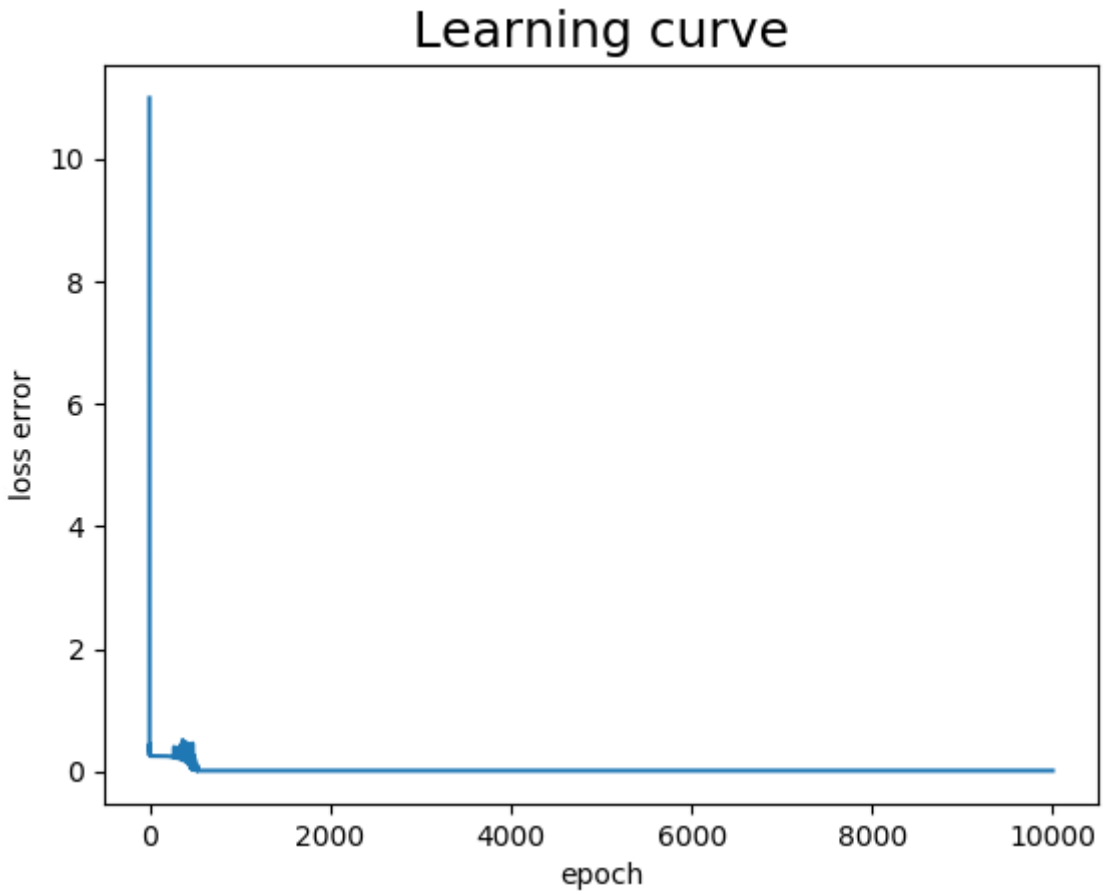
```
[[0.00436719]
 [0.97280763]
 [0.02090975]
 [0.97232855]
 [0.06099143]
 [0.97065677]
 [0.09449886]
 [0.95885126]
 [0.09270912]
 [0.68874875]
 [0.06899492]
 [0.04457432]
 [0.74821348]
 [0.02768852]
 [0.98544072]
 [0.01769212]
 [0.99127726]
 [0.01202818]
 [0.99174142]
 [0.00878233]
 [0.99162215]]
accurancy : 100.0%, loss : 0.009262
```

C. Learning curve (loss, epoch curve)

1. linear data



2. XOR data



D. anything you want to present

1. parameter of linear data
    - hidden_layer1_size = 5
    - hidden_layer2_size = 5
    - epochs = 10000
    - learning_rate = 0.01
2. parameter of XOR data
    - hidden_layer1_size = 5
    - hidden_layer2_size = 5
    - epochs = 10000
    - learning_rate = 0.8

Discussion

A. Try different learning rates

1. linear data

learning rate	**0.1**	**0.01**	**0.003**
accuracy	99%	100%	100%

learning rate	0.1	0.01	0.003
convergence rate	fast	fast	slow

2. XOR data

learning rate	0.8	0.01
accuracy	99%	52.4%

B. Try different numbers of hidden units

1. linear data

hidden units	(2,2)	(5,5)
accuracy	66.7%	100%
convergence rate	slow	fast

2. XOR data

hidden units	(2,2)	(5,5)
accuracy	66.7%	100%

C. Try without activation functions

1. linear data

activation functions	no	yes
accuracy	45.0%	100%

2. XOR data

activation functions	no	yes
accuracy	52.0%	100%

D. Anything you want to share

Extra

Implement different optimizers.

Implement different activation functions.

Implement convolutional layers.