Lab2 Back-propagation

Deep Learning and Practice

資科工碩1 游騰德 310551054

Introduction

This lab is to implement a simple neural network by only NumPy and standard Python libraries. The network to be implemented includes 1 input layer, 2 hidden layers, and 1 output layer. During training, the network forwards an input to the output layer, calculates the loss, and then updates the weights by back-propagation. The process is repeated until the results is satisfactory or the loss is converged. The purpose of the network is to separate the generated data into 2 classes. Please refer to the following for further implementing details.

Experiment setups

Sigmoid functions

Sigmoid function is one kind of activation functions applied before a neuron outputs the value. The definition of sigmoid function is written bellow:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

And the derivative of the sigmoid function is

$$\sigma'(x) = x * (1 - x)$$

The sigmoid function can make sure that the output value of a neuron is with 0 and 1. If the output value of sigmoid function is near 0 or 1, the gradient vanishing problem may exist.

Neural network

The network contains 4 layers. The input data is a 2 dimensional vector, and the output data is a 1 dimensional vector. The 2 hidden layers contains 4 neurons each. There are a total of 24 weights in this network. The network forwards an input to the output layer and the prediction 0 or 1 regarding the difference between the output value to 0 and 1.

Back-propagation

The weights of each neurons are updated during the back-propagation procedure. The weights are updated according to the loss, which is the difference between the ground truth value and the output value. In order to decrease the loss, we need to find the partial derivative respecting to each weight, so that we can discover the value to adjust the weights.

Output layer

Suppose O is a neuron in output layer and w is a weight of O. We want to find how w affects the the output of O as well as the loss E. As we know, w is multiplied by the output of the previous neuron, H, and then passed through activation function, follow by the loss function.Therefore, to find how w affects E, we find 3 values: the partial derivative of the loss function, the derivative of activation function, and the partial derivative of the input of O over w. We then multiply the 3 values to get the partial derivative of E over w, which is how w affect E.

Hidden Layers

As for hidden layers, the input part and the activation part remains the same. The main difference is that the loss function is not applied to the neuron output. So instead of finding the partial derivative of the loss function, we find the sum of all partial derivatives of E over the output, which is the first value. The calculation of the rest 2 values are the same as those in output layers. As the aforementioned method, we multiply the 3 values to see how a weight affects E.

Note that during the back propagation process, the weights are updated backward from the output layer to the input layer. The partial derivative of E over a neuron's output can be calculated and save within that neuron itself. By applying Chain Rule, the saved value can be used to update the weights of the neurons in the previous layer.

Learning rate

After we know how w affect E, we can update the weights by tuning it according to the derivative value. However, to avoid making dramatic changes, we set a rate to diminish the correction value. The rate is called learning rate.

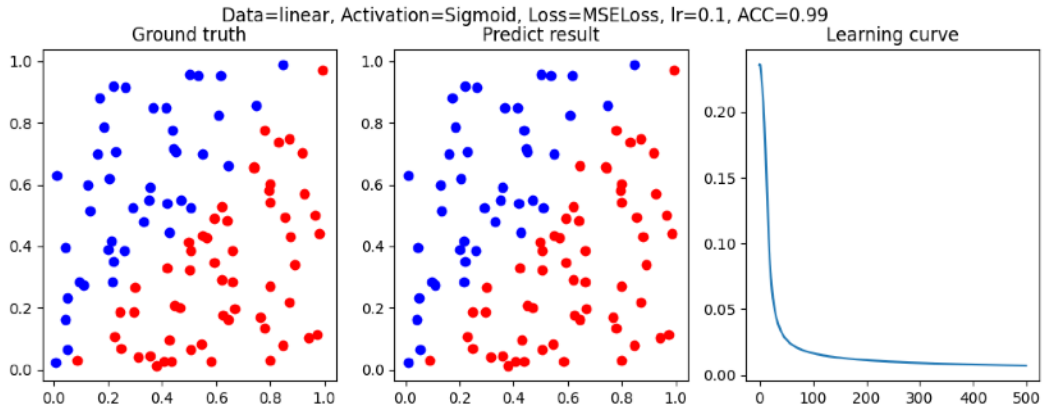
Results of testing

The plots below are the comparison figure and the learning curve of each test. Configurations of each test are listed in the title of each figure, follow by the accuracy (ACC) on the dataset.

The screenshots below are the trained network of each test. For each neuron, the weights, activation function, and the current output is printed.

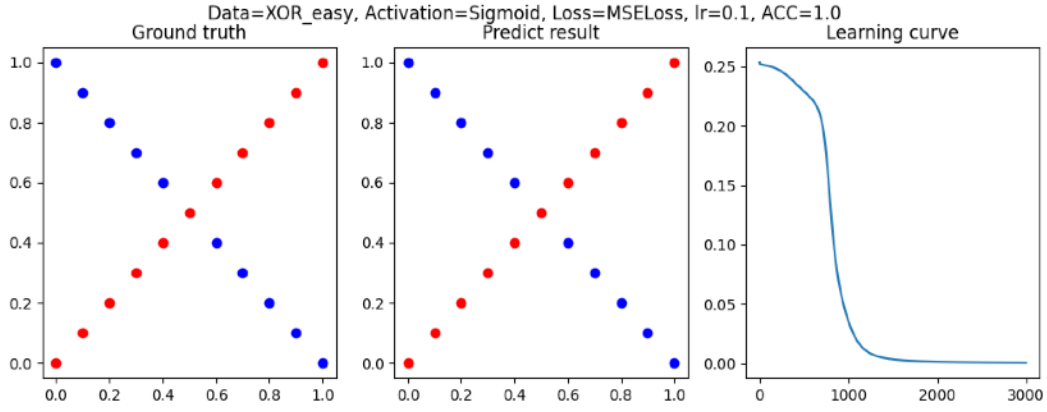
In this test of XOR easy dataset w/ MAE loss, the loss took more epochs than the other tests to be converged. Nevertheless, the results accuracy is still not as good as the others.

Linear dataset w/ MSE loss



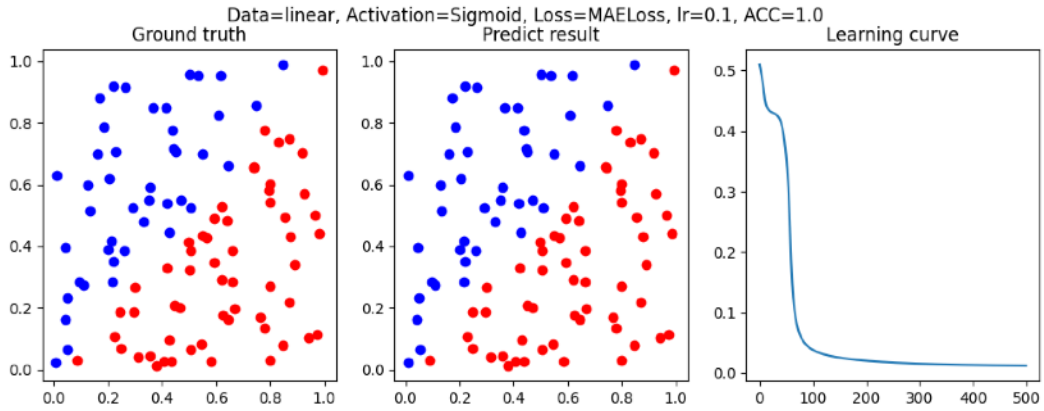

```
FCNet
Layer 0
Neuron [] Sigmoid 0.9167146119578825
Neuron [] Sigmoid 0.7022642408197249
Bias 0.0
Layer 1
Neuron [3.1504819772475647, -2.0896408061245078] Sigmoid 0.8054344728669421
Neuron [5.200453172970554, -5.373498183026413] Sigmoid 0.7298212304610096
Neuron [-6.507730069628558, 5.944505487639805] Sigmoid 0.1429357585945506
Neuron [-0.9809432865024469, -0.12220960765858674] Sigmoid 0.2718872450106569
Bias 0.0
Layer 2
Neuron [0.19273650799358133, 1.0180523061086517, -2.6076192003020617, 0.7666443348325851] Sigmoid 0.6756718217069636
Neuron [0.7264887230072253, 0.7899256928134115, -1.984207855317255, 0.07143890257845198] Sigmoid 0.7104257436268443
Neuron [1.229566742100599, 4.3889895551404825, -5.996479082868699, -0.8344118523832598] Sigmoid 0.9572843848475334
Neuron [-1.1410628204486097, -4.139607056596056, 5.520580060105527, 0.907950801993920l] Sigmoid 0.051942594815300655
Bias 0.0
Layer 3
Neuron [-1.771049355182317, -1.0571436099097316, -6.659077280381821, 8.291292457405156] Sigmoid 0.00037372012476136744
Bias 0.0
```

XOR easy dataset w/ MSE loss



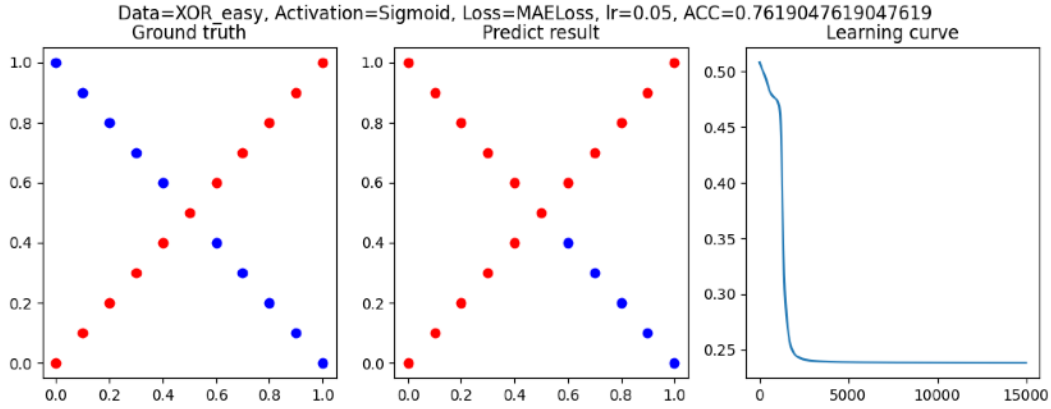

```
FCNet
Layer 0
Neuron [] Sigmoid 1.0
Neuron [] Sigmoid 0.0
Bias 0.0
Layer 1
Neuron [-5.2711683456595555, 6.329497006109792] Sigmoid 0.00511134454337921
Neuron [6.763071503343693, -6.201114802707604] Sigmoid 0.9988456599794703
Neuron [-3.959597880131821, -3.796479264781385] Sigmoid 0.0187138935542379
Neuron [-4.096169657094694, 3.346579575188232] Sigmoid 0.016364039468330833
Bias 0.0
Layer 2
Neuron [5.522957033304295, -4.99035580104601, 2.149486477922855, 3.5888229180927946] Sigmoid 0.007710647623076534
Neuron [-4.6123004651767845, 2.6174255209369903, -4.430135667887593, 0.035599072459379294] Sigmoid 0.9247359885188111
Neuron [2.3918105055062324, -9.166378994559707, -2.302396303884866, 3.6663735441409937] Sigmoid 0.0001087160251227l324
Neuron [-0.037566213307902524, 0.9404601899971226, 2.624224525944833, -1.0075641847442944] Sigmoid 0.7254789934486471
Bias 0.0
Layer 3
Neuron [-5.901880641891243, 7.744438920088182, 10.564915238435963, -1.0620885801812605] Sigmoid 0.9982505080713839
Bias 0.0
```

Linear dataset w/ MAE loss




```
FCNet
Layer 0
Neuron [] Sigmoid 0.9167146119578825
Neuron [] Sigmoid 0.7022642408197249
Bias 0.0
Layer 1
Neuron [-4.243307912216955, 3.702950100107773] Sigmoid 0.21594546620695151
Neuron [8.230681915937401, -8.117707648021598] Sigmoid 0.8634694607005825
Neuron [-4.6933998173304685, 4.1889093599120315] Sigmoid 0.20411248355410833
Neuron [-0.21445652260092543, 1.3124639179774378] Sigmoid 0.6737293383493418
Bias 0.0
Layer 2
Neuron [2.3767191354499166, -3.5729980426790107, 1.6457498347776474, 0.5748798143740282] Sigmoid 0.1360254639613028
Neuron [-5.10529013878976, 8.781479574653549, -4.729314396757269, -0.5745234944102217] Sigmoid 0.9941047152087636
Neuron [1.2578043320248748, -4.487734057479773, 3.0175058451725025, 1.1335360912122692] Sigmoid 0.09763379075544704
Neuron [-1.6547902044792009, 3.575749756007265, -2.650203919703675, -0.3195957114576919] Sigmoid 0.8780342551639266
Bias 0.0
Layer 3
Neuron [5.7243782956391795, -10.737782037564267, 6.368611334122549, -3.7759954887356573] Sigmoid 3.4076656555503758e-06
Bias 0.0
```

XOR easy dataset w/ MAE loss

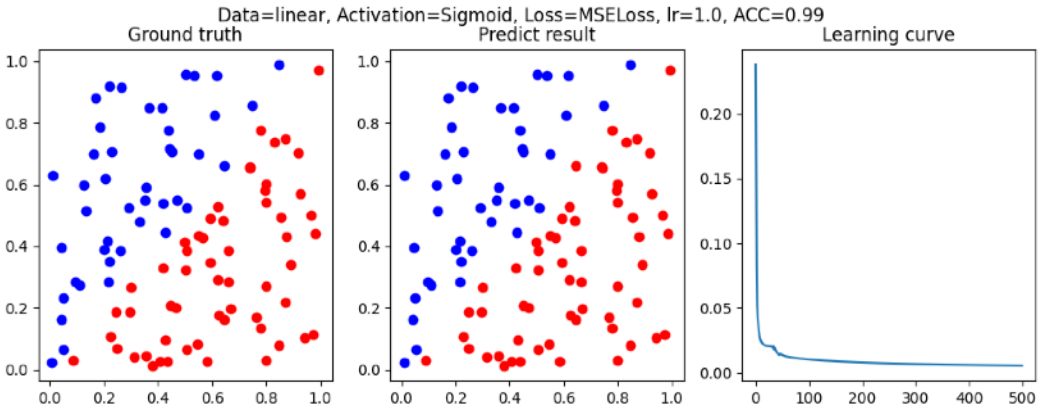


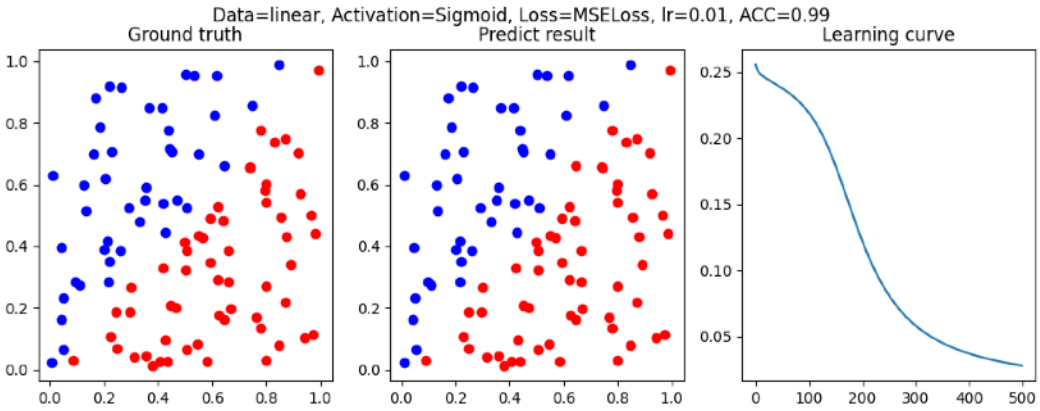

```
FCNet
Layer 0
Neuron [] Sigmoid 1.0
Neuron [] Sigmoid 0.0
Bias 0.0
Layer 1
Neuron [1.870284344581865, -1.2831766710330814] Sigmoid 0.8664911607932767
Neuron [-4.0426494723943245, 1.0172441781910436] Sigmoid 0.01724818930278741
Neuron [2.886004125165131, -2.6395732061825425] Sigmoid 0.9471502179375448
Neuron [-7.299056237862479, 8.418159852338777] Sigmoid 0.0006757197196315059
Bias 0.0
Layer 2
Neuron [-0.039560337260300574, 0.6798450027268356, -1.0183747705608244, -0.9189368067037329] Sigmoid 0.27136116432939617
Neuron [1.7923428662282286, -2.6927650512657637, 3.58923026329798, -6.650671655788538] Sigmoid 0.9926205546017258
Neuron [0.7474398210750801, -1.7661070096666756, -0.3607677913115003, -1.9971798242100245] Sigmoid 0.5681078574782591
Neuron [-2.679951415249536, 2.341082678484989, -3.858283988448108, 8.292984754581084] Sigmoid 0.0026499911299225124
Bias 0.0
Layer 3
Neuron [0.22904221482763265, 8.979048381588676, 2.397465816026887, -11.678229213285197] Sigmoid 0.9999665695779343
Bias 0.0
```

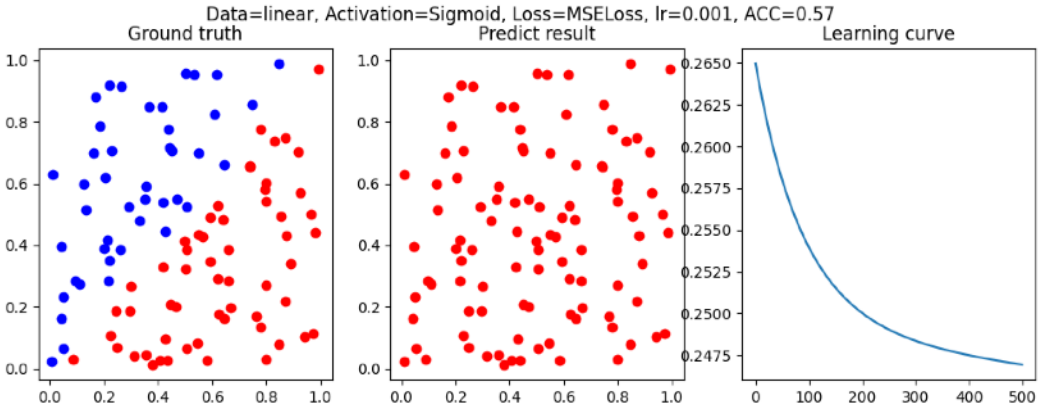
Discussion

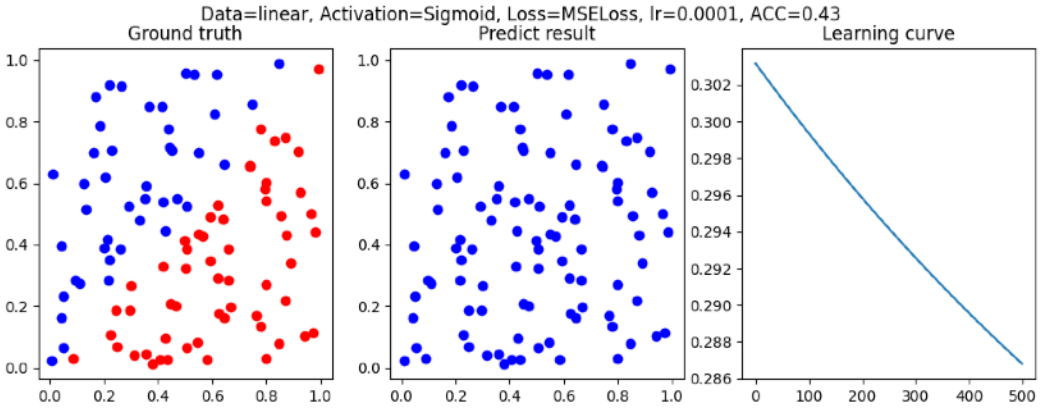
Try different learning rates

The smaller the learning rate is, the more epochs the network takes to converge. The last 2 tests are not even converged yet after 500 epochs of training.







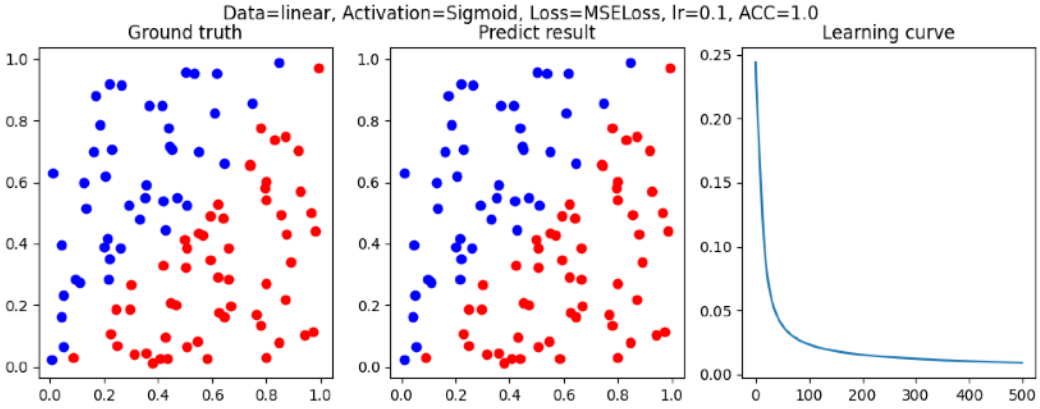


Try different numbers of hidden units

In this comparison, we can see that the more hidden layers, the more epochs the network takes to converge. Additionally, the computational time increased significantly while the number of hidden layers increased. All hidden layers added has 4 neurons each.

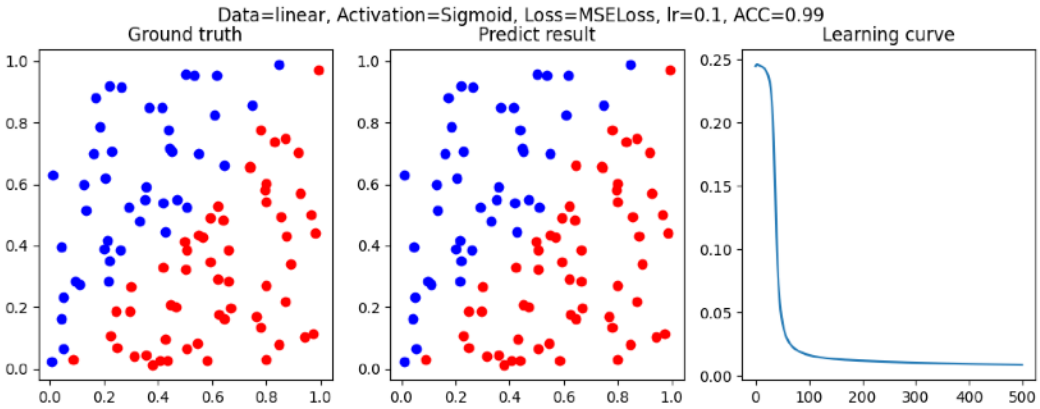
---

1 hidden layer

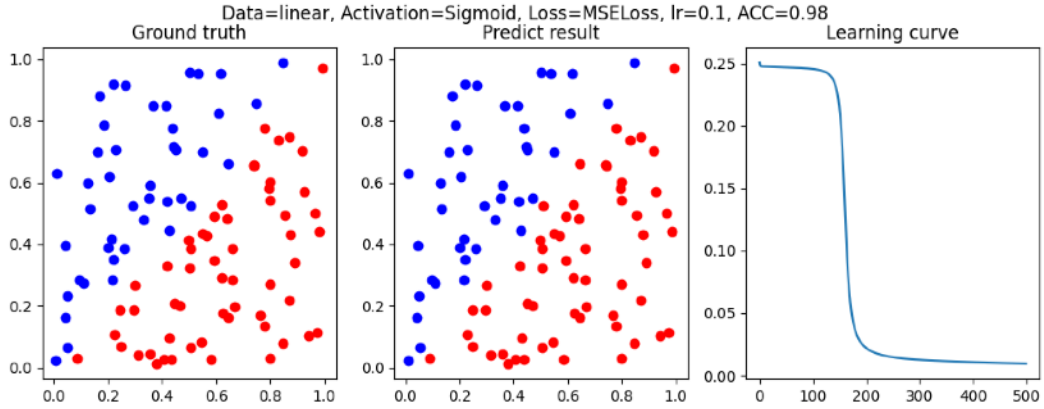


---

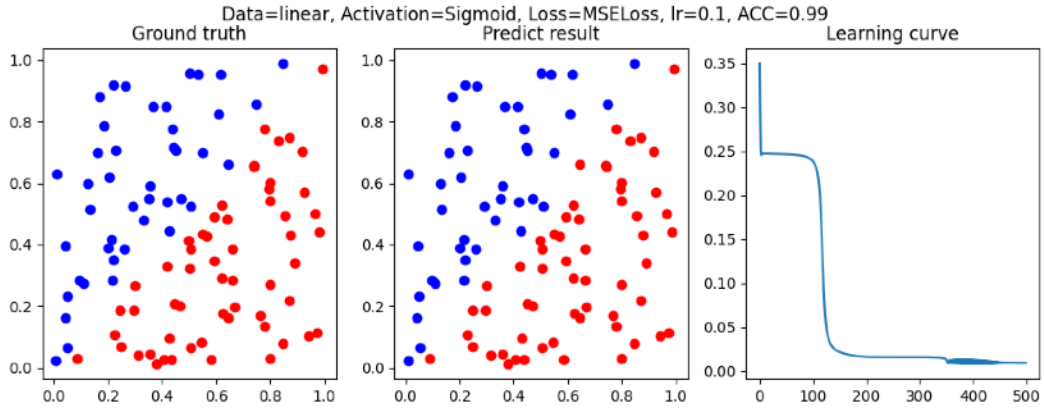
3 hidden layers



4 hidden layers

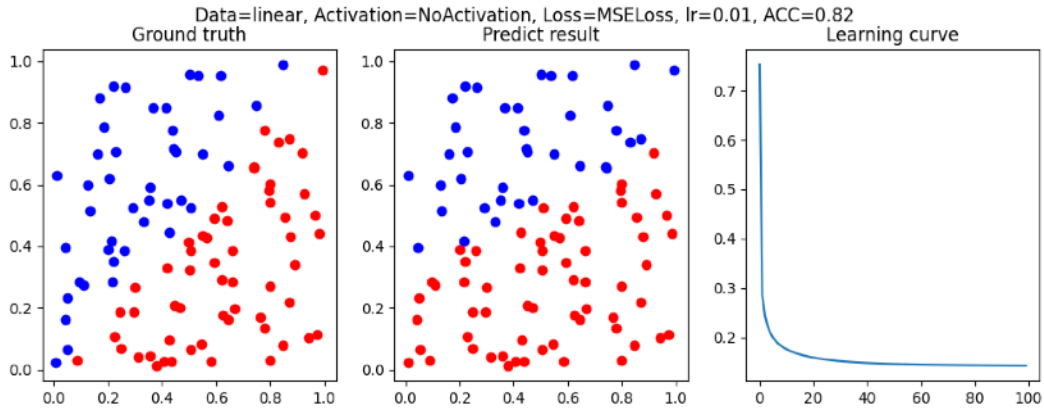


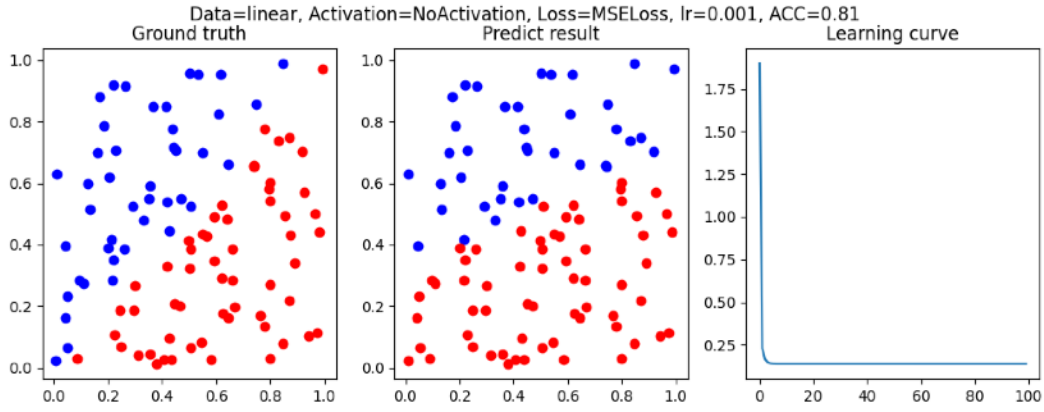
5 hidden layers

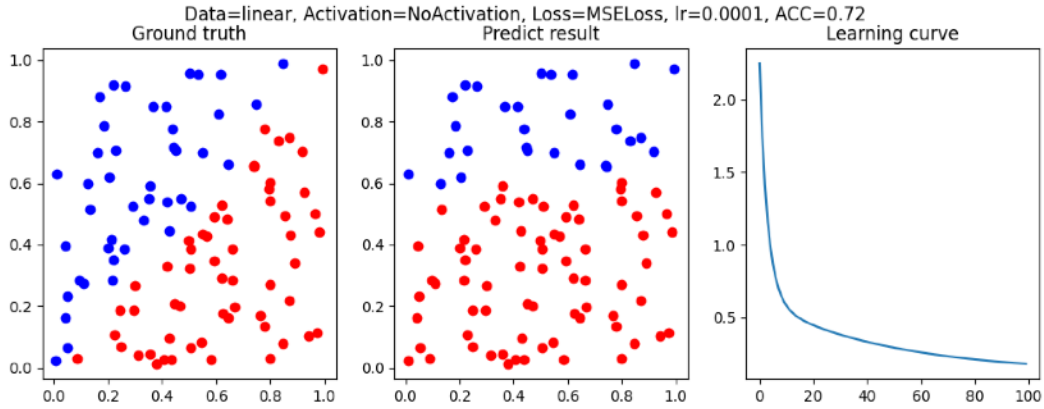


Try without activation functions

In this compassion, I found that if the learning is set to 0.1, the value of some neuron output will become too large, causing the weight to be set to nan. Also, the accuracy is worse in comparison with tests with sigmoid function applied.



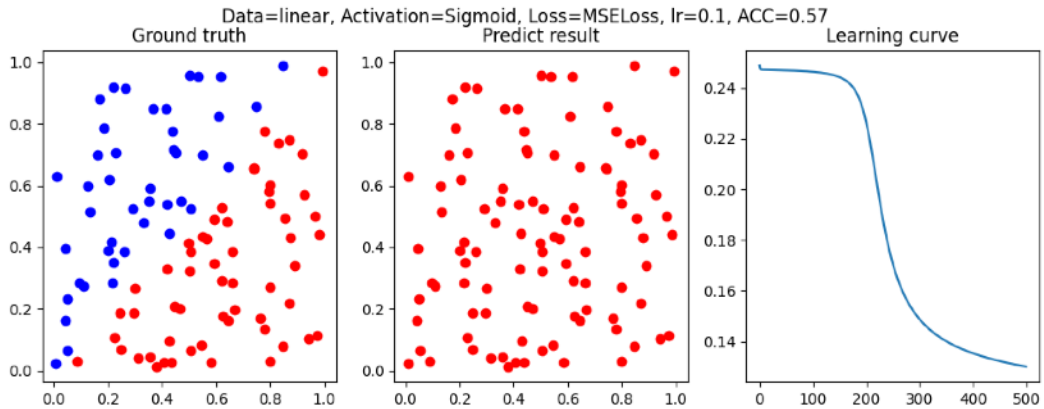




Things to share

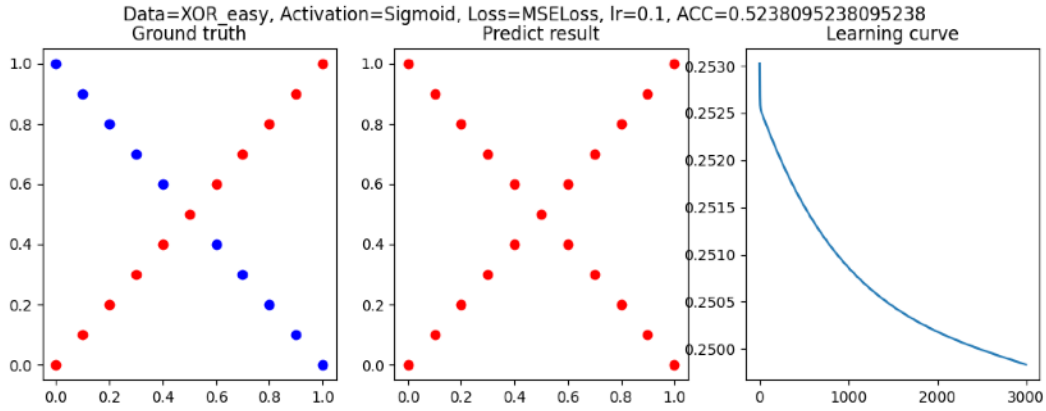
During training, I found that the initial weighs to each neuron should be randomly assigned to a reasonable value. Despite the loss is still decreasing, the network will learn nothing if all the weights are initialized to be 0. The weight in the same layer will be updated through the same derivative calculation, so the weight values are always the same. The results are shown below if the weights are all set to 0 in the beginning of the training.

Linear dataset w/ MSE loss (weights initialize to 0)



```
FCNet
Layer 0
Neuron [] Sigmoid 0.9167146119578825
Neuron [] Sigmoid 0.7022642408197249
Bias 0.0
Layer 1
Neuron [-5.535523803222838, 4.386211313269787] Sigmoid 0.11982047429788442
Neuron [-5.535523803222838, 4.386211313269787] Sigmoid 0.11982047429788442
Neuron [-5.535523803222838, 4.386211313269787] Sigmoid 0.11982047429788442
Neuron [-5.535523803222838, 4.386211313269787] Sigmoid 0.11982047429788442
Bias 0.0
Layer 2
Neuron [-2.353562763838254, -2.353562763838254, -2.353562763838254, -2.353562763838254] Sigmoid 0.2445260029485633
Neuron [-2.353562763838254, -2.353562763838254, -2.353562763838254, -2.353562763838254] Sigmoid 0.2445260029485633
Neuron [-2.353562763838254, -2.353562763838254, -2.353562763838254, -2.353562763838254] Sigmoid 0.2445260029485633
Neuron [-2.353562763838254, -2.353562763838254, -2.353562763838254, -2.353562763838254] Sigmoid 0.2445260029485633
Bias 0.0
Layer 3
Neuron [-3.914080743897866, -3.914080743897866, -3.914080743897866, -3.914080743897866] Sigmoid 0.021282521444546208
Bias 0.0
```

Linear dataset w/ MSE loss (weights initialize to 0)

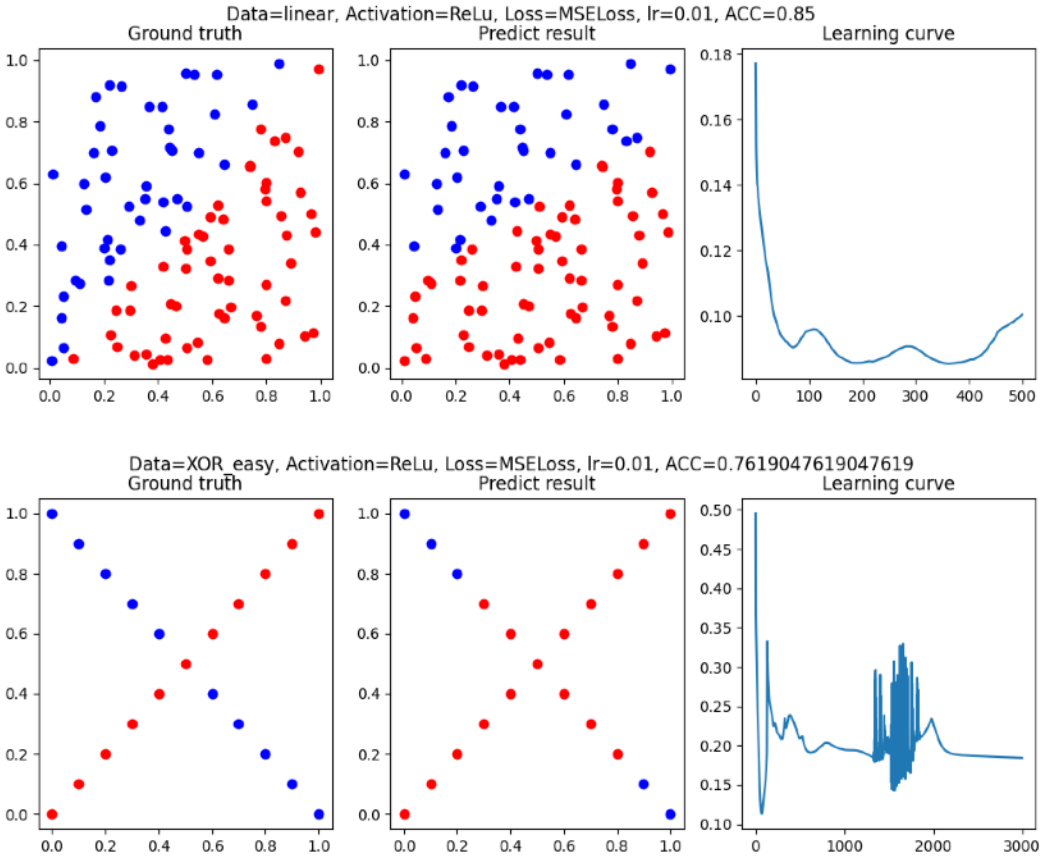



```
FCNet
Layer 0
Neuron [] Sigmoid 1.0
Neuron [] Sigmoid 0.0
Bias 0.0
Layer 1
Neuron [0.25622468473597687, 0.22615178127565788] Sigmoid 0.5636249132934149
Neuron [0.25622468473597687, 0.22615178127565788] Sigmoid 0.5636249132934149
Neuron [0.25622468473597687, 0.22615178127565788] Sigmoid 0.5636249132934149
Neuron [0.25622468473597687, 0.22615178127565788] Sigmoid 0.5636249132934149
Bias 0.0
Layer 2
Neuron [-0.662050268294137, -0.662050268294137, -0.662050268294137, -0.662050268294137] Sigmoid 0.18363182295567662
Neuron [-0.662050268294137, -0.662050268294137, -0.662050268294137, -0.662050268294137] Sigmoid 0.18363182295567662
Neuron [-0.662050268294137, -0.662050268294137, -0.662050268294137, -0.662050268294137] Sigmoid 0.18363182295567662
Neuron [-0.662050268294137, -0.662050268294137, -0.662050268294137, -0.662050268294137] Sigmoid 0.18363182295567662
Bias 0.0
Layer 3
Neuron [-0.12756685327086092, -0.12756685327086092, -0.12756685327086092, -0.12756685327086092] Sigmoid 0.4757119162141777
Bias 0.0
```

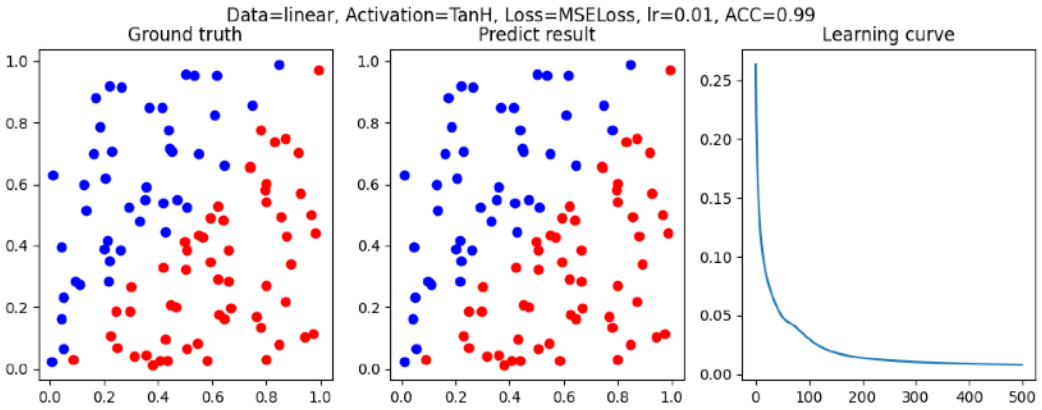
Extra

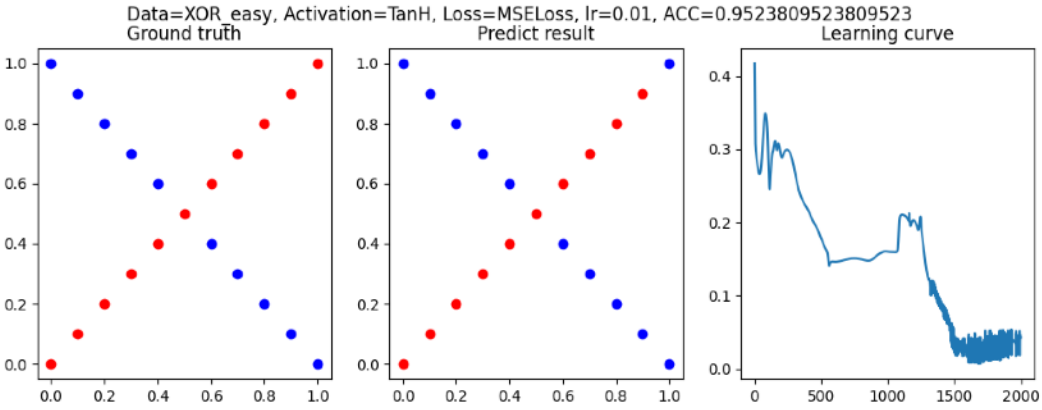
Implement different activation functions

ReLu



TanH




Data=XOR_easy, Activation=TanH, Loss=MSELoss, lr=0.01, ACC=0.9523809523809523
Ground truth
Predict result
Learning curve
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
0.4
0.3
0.2
0.1
0.0
0
500
1000
1500
2000