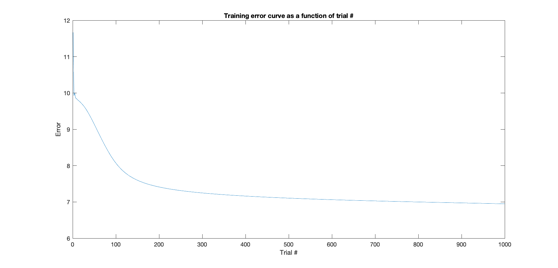
**Xiang Pan 潘翔**

**1900941520**

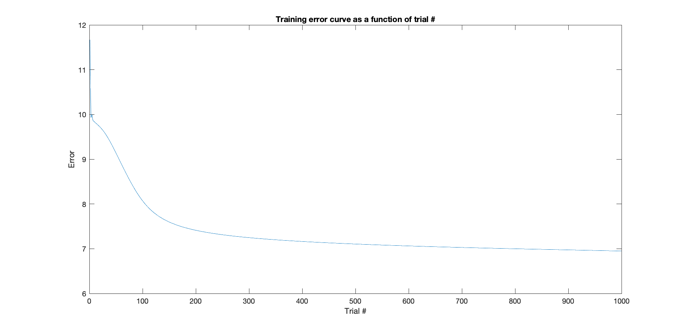
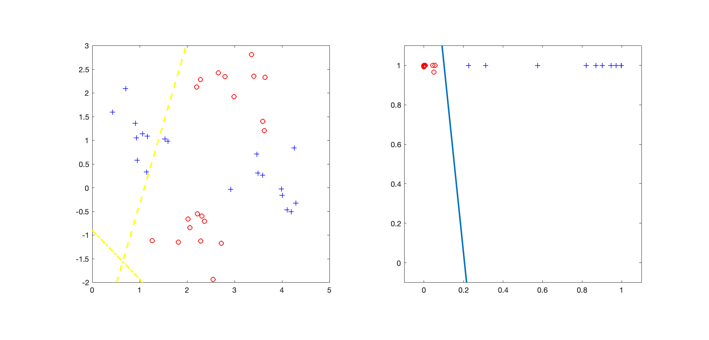
[**hover@hust.edu.cn**](mailto:hover@hust.edu.cn)

# Q1 PLA Result

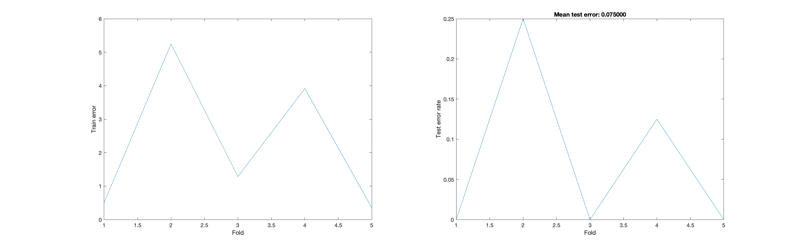


**Image 1.1 The Result and Train Loss of PLA**

# Q2 MLP Result

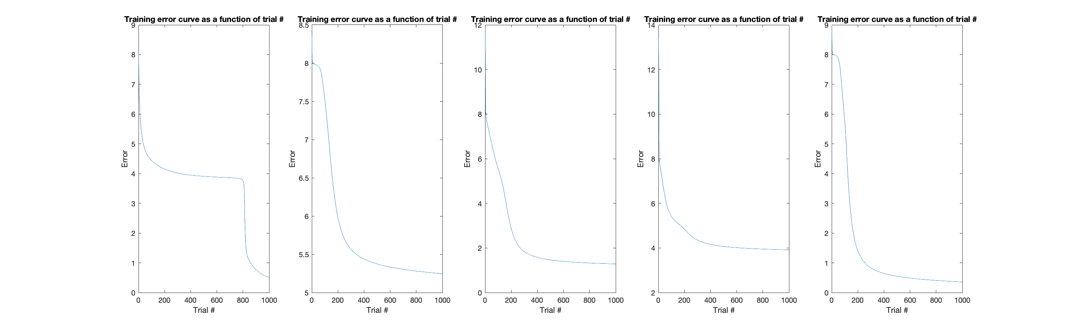
****

**Image 2.1 The Result and Train Loss of MLP**

****

# Q3 5-Fold

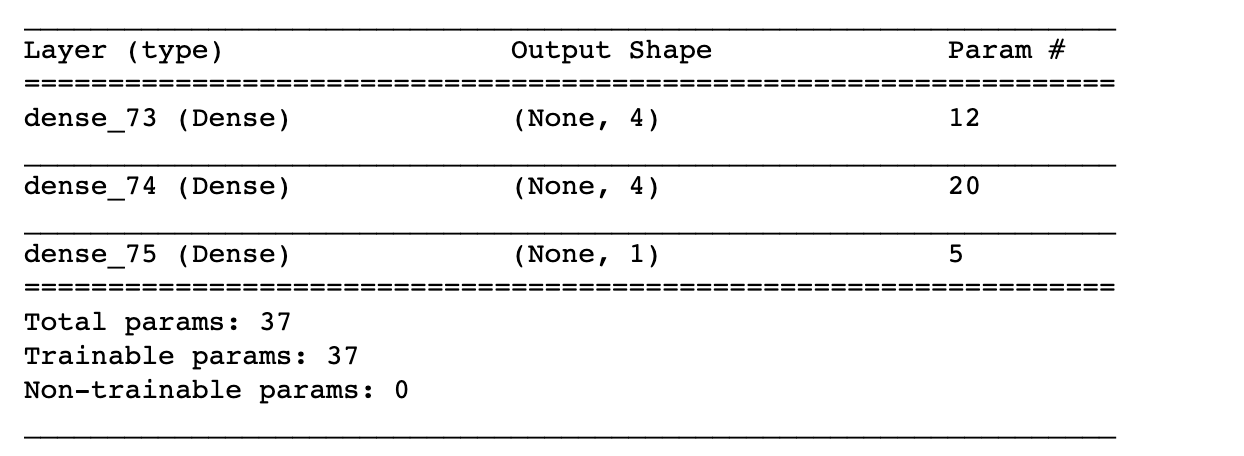
**Image 3.1 Train Loss of 5 Folds(****Classication)**

****

**Image 3.2 Train Error of 5 Folds(XOR)**

# QC: MLP for Classification

## Net1

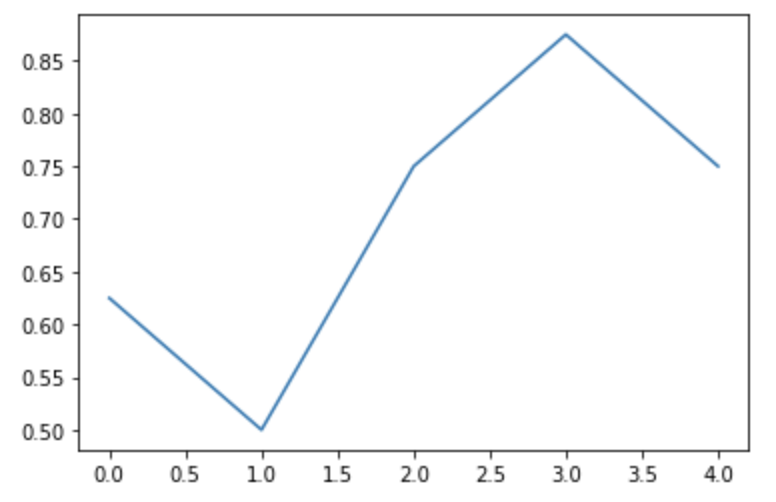


**Image C.1 Network Summary1**

MLP\_Split(0.8,0.2):0.875

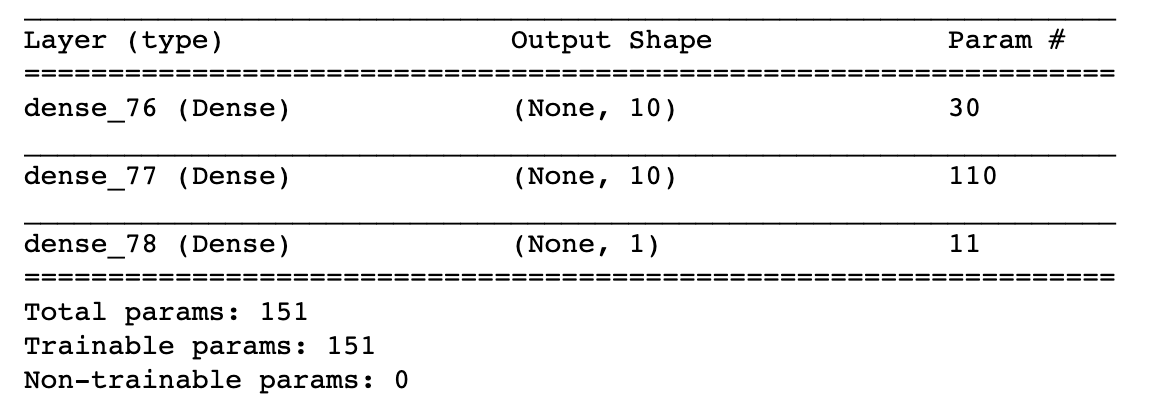
MLP\_overfit:1.0

MLP\_5\_Fold:



**Image C.2 Train Error of 5 Folds(Apple & Orange)**

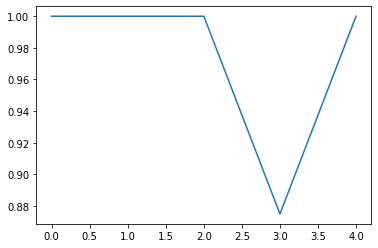
## Net2



**Image C.3 Network Summary2**

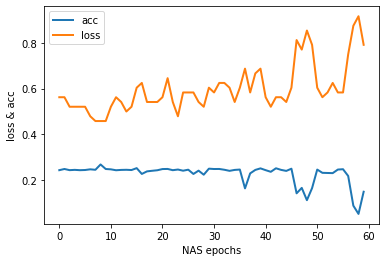
MLP\_Split(0.8,0.2):1.0

MLP\_5\_Fold:

****

**Image C.4 Train Error of 5 Folds(Apple & Orange)**

## MLP for Spirals:



**Image C.5 MLP of Spirals**

Because of the highly nonlinear of spirals’ data, the MLP cannot balance the simple network architecture and generalization. In fact, the highest acc is about 0.92, but the model will overfit in someway.

# NAS for MLP

MLP is the simplest NN, so it is very useful in understanding how NN works. In part.1 we try to adjust the hyperparameter to minimize the NN scale and maximize the performance of task. Here, I try to use NAS Method for smarter network design.

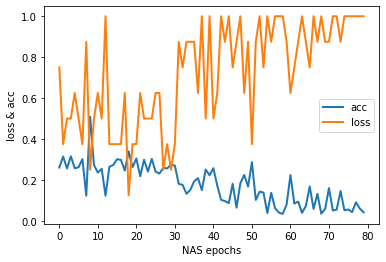
Here, I use the simplest violent search for MLP Network Architecture.

**Code C.1 NAS for MLP**

for neuron\_indx in range(neuron\_max // 10):

for layer\_indx in range(layer\_max):

model = basic\_model(layers, neurons)



**Image C.6 NAS for MLP Result**

From the search result, we can find the suitable model parameter and network architecture for the MLP.

**Code C.2 NN Configuration**

Neural Num=indexmax(acc) % earch\_step

Layer Num= indexmax(acc) /search\_step