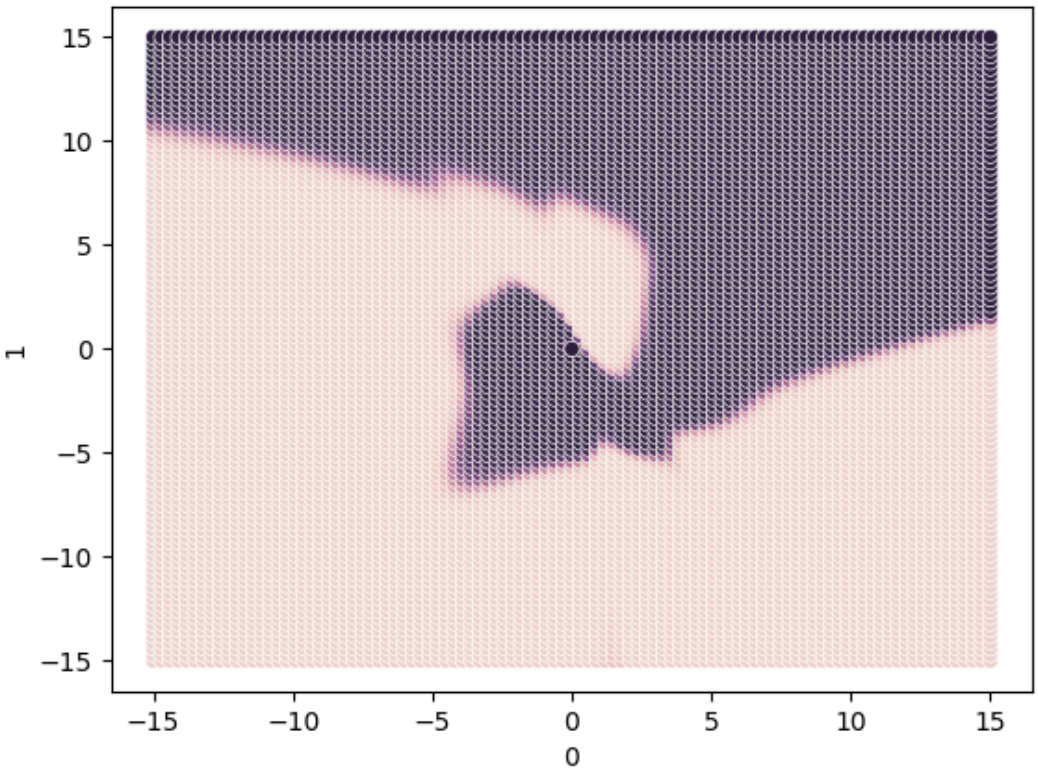
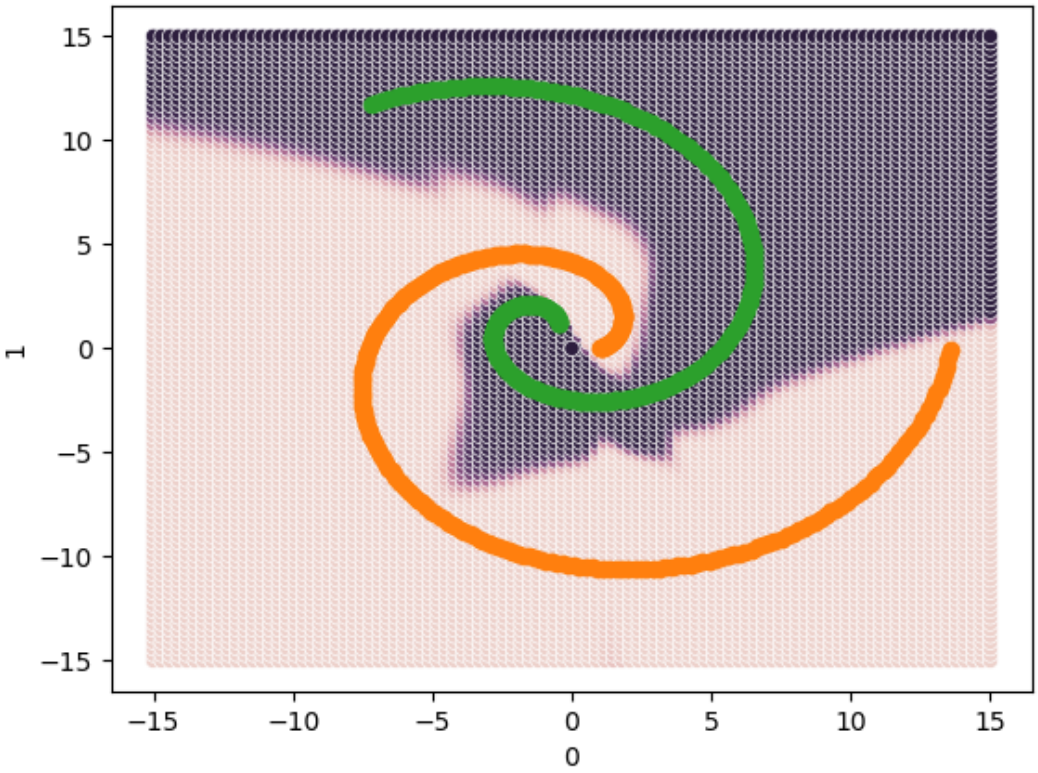
- I have **two hidden layers and one output layer** which of each has **12 neurons**, **4 neurons** and **1 neuron (output)** in order.
- In first hidden layer takes 2 dimension vector as input and products 12 dimension vector **(12 neurons)** as output. In second hidden layer takes 12 dimension vector as input and products 4 dimension vector **(4 neurons)** as output. In output layer, takes 4 dimension vector as input and products scalar value **(1 neuron)**. Then, apply sigmoid function to this output value of MLP. This is the final predicted scalar value which is bounded in [0,1].
- In every layer I use **weight** and **bias** which of sizes depend on input dimension and output dimension.
- I use **xavier initializer** (uniform distribution) for initializing every weights and biases between layers.
- In every layer **learning lates are 0.6**.
- One can easily find my hyperparameter setting from "def __init__(self):" method of class MLP
- I utilize **sigmoid function** as activation function in every hidden layers and output layer.
- I update my parameter for **1000000 times**, in each of which update is progressed by **fully stochastic** method.
- This is the output visualization



- This is the output visualization overlapped with training samples.



- If seaborn and matplotlib package are installed in one's local, mymlp.py script produces above two pictures automatically in the same directory of mymlp.py.