Assignment 4: Neural Networks

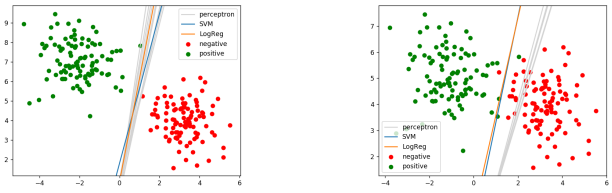
Chen Zongqi
Mat-Nr.1564832

December 5, 2017

1 Perceptron Learning

a) I set a flag to obtain a statement when there is still a wrong classified data. When the flag doesn't change which means every data is correctly classified, then break the loop.

b) We can see from Figure 1 blow, dataset 1 is linear separability dataset and dataset 2 is non-linear separability dataset. The result of misclassification rate we can see from Table 1. In linear separability dataset all of three algorithms perform prefect zero error rate, while in non-linear separability dataset our perceptron has worse performance than the others. The reason might be perceptron learning algorithm can not approach so precisely to the best decision boundary, Logistic regression can approach tiny step by step and Linear SVM can separate it in high dimension.



(a) Dataset 1 (b) Dataset 2

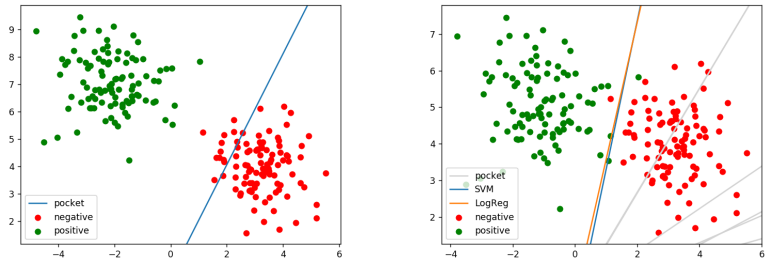
Figure 1: Rosenblatt's perceptron learning algorithm

Algorithm	Dataset 1	Dataset 2
Perceptron (best result)	0	11
Linear SVM (C=1)	0	3
Logistic regression	0	3

Table 1: Misclassification rates (train)

c) Done

d) See Figure 2 below, when we use pocket algorithm, the performance is worse than normal perceptron learning algorithm in both dataset 1 and dataset 2. That would be acceptable due to the principle of pocket algorithm is a approaching method to solve non-linear separability dataset. However pocket algorithm performs worse than normal algorithm in non-linear separability dataset 2, I think the reason depends on epoch number and sample set.



(a) Dataset 1 (b) Dataset 2

Figure 2: Pocket algorithm

2 Multi-Layer Feed-Forward Neural Networks

a) For FNN with zero hidden neurons, it would be like a line roughly drawing in the Figure. When hidden neurons are greater than zero, the line would have some curves depending on how many hidden neurons have. I think the decision boundary would have better performance when hidden neurons are more than 2.

b) See Figure 3, the decision boundary can separate quite better than 1 hidden neurons. When I run this several times, decision boundary may change little differently.

c) Done

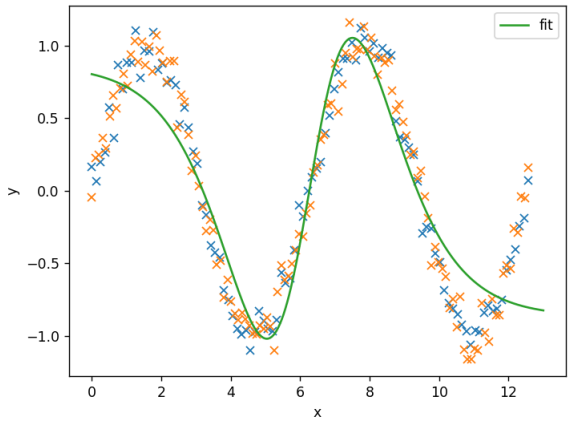
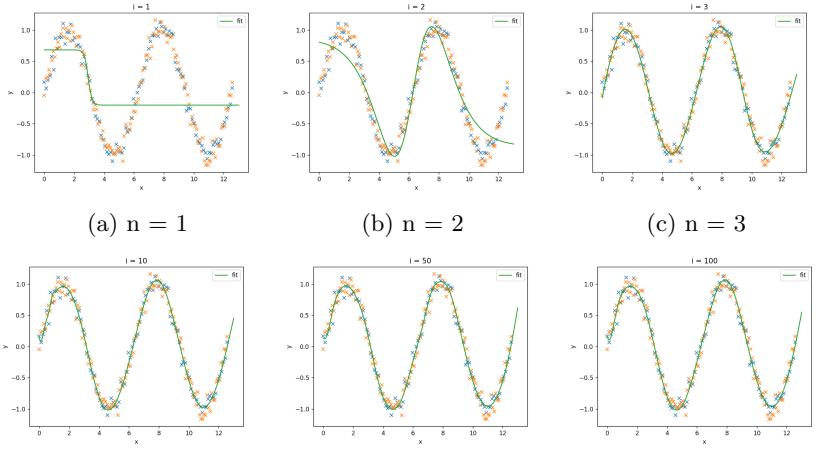


Figure 3: 2 hidden neurons

d) We can see the Figure 4, after n = 3 the decision boundary can fit our dataset better. Meanwhile after n = 3, the misclassification rates for test are nearly same about 0.011. The reason why it preforms stable due to we use meta-learning that find the best performance model in 10 experiments. Then we can avoid some bad model randomly occurred.

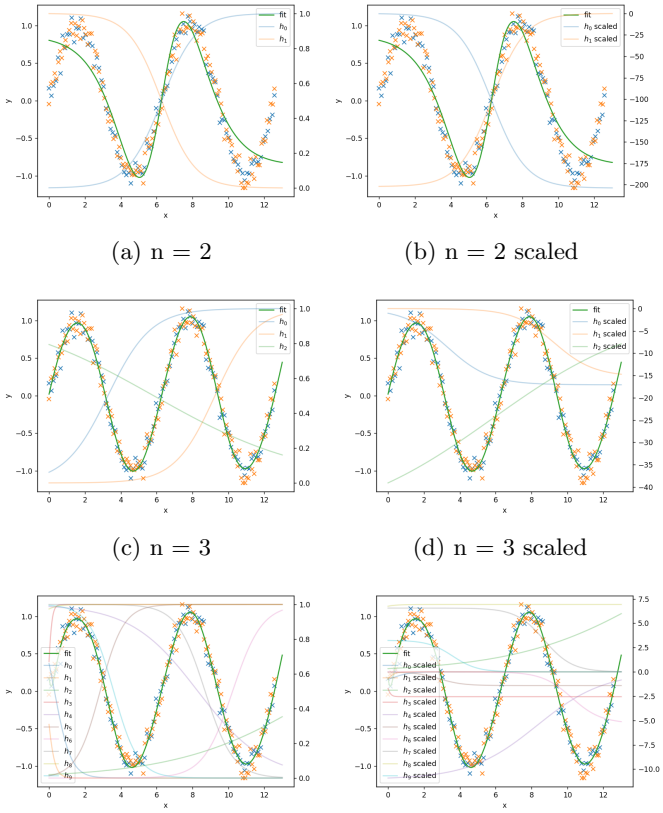


(a) n = 1 (b) n = 2 (c) n = 3

(d) n = 10 (e) n = 50 (f) n = 100

Figure 4: Improved training

e) From Figure 5, we can see different hidden neurons decision boundaries. When n = 2, two hidden neurons h_0 and h_1 represent intuitively which h_0 and h_1 can fit the decision boundary as left and right part. But when n is greater, the result might be more difficult to understand what the meaning of them.



(a) n = 2 (b) n = 2 scaled

(c) n = 3 (d) n = 3 scaled

(e) n = 10 (f) n = 10 scaled

Figure 5: Scaled

f) We can see from Figure 6, when we use ReLU instead of Logistic, the decision boundary looks quite like a wave. The line is not smoothly but curving.That would be more flexible for classifying. The equation of ReLU is $ReLU(x) = max\{0, x\}$ which ignores the value less than 0. Then when we do regression it would approach to minimum loss function value safely.

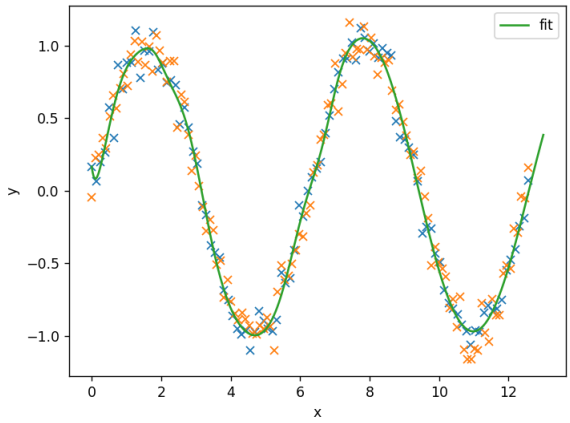
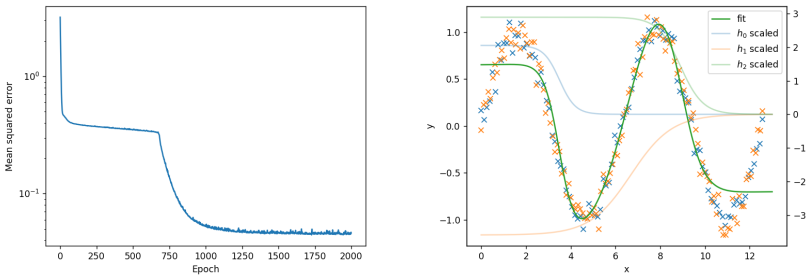


Figure 6: Activation with ReLU

g) Compared with scikit-learn, when n = 3 keras has lower misclassification rate than scikit-learn, result from Figure 7 below. However the difference between keras and scikit-learn is not so obviously. That might be caused by our small dataset.



(a) Training progress (b) Keras n = 3

Figure 7: Keras