Homework 1

陈子轩 (120033910005)

April 2021

Problem 1

Online learning form

Suppose the network has a total of K layers

Forward Propagation:

$$net_{kj} = \sum_{i=1}^{N_{k-1}} (u_{kji}x_{k-1,i}^2 + v_{kji}x_{k-1,i}) + b_{kj}$$

$$x_{kj} = f(net_{kj})$$

$$e_{Kj} = d_{Kj} - x_{Kj}$$

$$Loss = \frac{1}{2}\sum_{j\in C} e_{Kj}^2$$

Backward Propagation:

for $k = K$

$$\delta_{kj} = \frac{\partial Loss}{\partial x_{kj}} = -e_{kj} = x_{Kj} - d_{Kj}$$

for $K > k-1 \geq 1$

$$\delta_{k-1,i} = \sum_j \frac{\partial Loss}{\partial x_{kj}} \frac{\partial x_{kj}}{\partial net_{kj}} \frac{\partial net_{kj}}{\partial x_{k-1,i}} = \sum_j \delta_{kj} f'(net_{kj})(2u_{kji}x_{k-1,i} + v_{kji})$$

for $K \geq k \geq 1$

$$\frac{\partial Loss}{\partial u_{kji}} = \frac{\partial Loss}{\partial x_{kj}} \frac{\partial x_{kj}}{\partial u_{kji}} = \delta_{kj} f'(net_{kj}) x_{k-1,j}^2$$

$$\frac{\partial Loss}{\partial v_{kji}} = \frac{\partial Loss}{\partial x_{kj}} \frac{\partial x_{kj}}{\partial v_{kji}} = \delta_{kj} f'(net_{kj}) x_{k-1,j}$$

$$\frac{\partial Loss}{\partial b_{kj}} = \frac{\partial Loss}{\partial x_{kj}} \frac{\partial x_{kj}}{\partial b_{kj}} = \delta_{kj} f'(net_{kj})$$

Batch learning form

Denote the batch size as m and the dimension of the kth layer as n_k. Suppose X_k is a batch form of the kth layer's feature. X_k is a $n_k \times m$ matrix. Therefore the weight matrix U_k and V_K are both $n_{k-1} \times n_k$. b_k is the bias vector in the form of $n_k \times 1$. $1_{1\times m}$ is a $1 \times m$ all 1 vector.

Forward Propagation:

$$N_k = U_k^T(X_{k-1} \circ X_{k-1}) + V_k^T X_{k-1} + b_k 1_{1\times m}$$

$$X_k = f(N_k)$$

$$E_K = D_K - X_K$$

$$Loss = \frac{1}{2} Tr(E_K^T E_K)$$

Backward Propagation:
for $k = K$

$$\Delta_k = \frac{\partial Loss}{\partial X_k} = -E_k = X_k - D_k$$

for $K > k-1 \geq 1$

$$\Delta_{k-1} = \frac{\partial Loss}{\partial X_k}\frac{\partial X_k}{\partial N_k}\frac{\partial N_k}{\partial X_{k-1}} = \Delta_k f'(N_k)(2U_k^T X_{k-1} + V_k^T)$$

Problem 2

We implemented a two-layer MLQP using Pytorch. The dimension of hidden layer is set to 512. The on-line BP version is achieved by setting $batchsize = 1$. We did online learning at 3 different learning rate: 0.0001, 0.0005 and 0.001. We train the network for 200 epochs and the result is shown below.

From Figure 1 we can see that as the learning rate increases, the running time decreases a little bit. The MLQP model achieves highest accuracy at learning rate= 0.0001(over 95%). As the learning rate increases, the loss curve and the accuracy curve oscillates more and more violently.

Problem 3

As shown in Figure 2, we randomly divide the original data set into four pieces and combine them into 4 different sub-problems. We trained four MLQP networks separately and constructed two min-max modular networks, as is shown in Figure 3

We train the network for 100 epoches at learning rate 0.0002. The result is shown in Figure 4. We can see that the decision boundary of the second min-max network is obviously better than the first one, although it cost a bit more time.

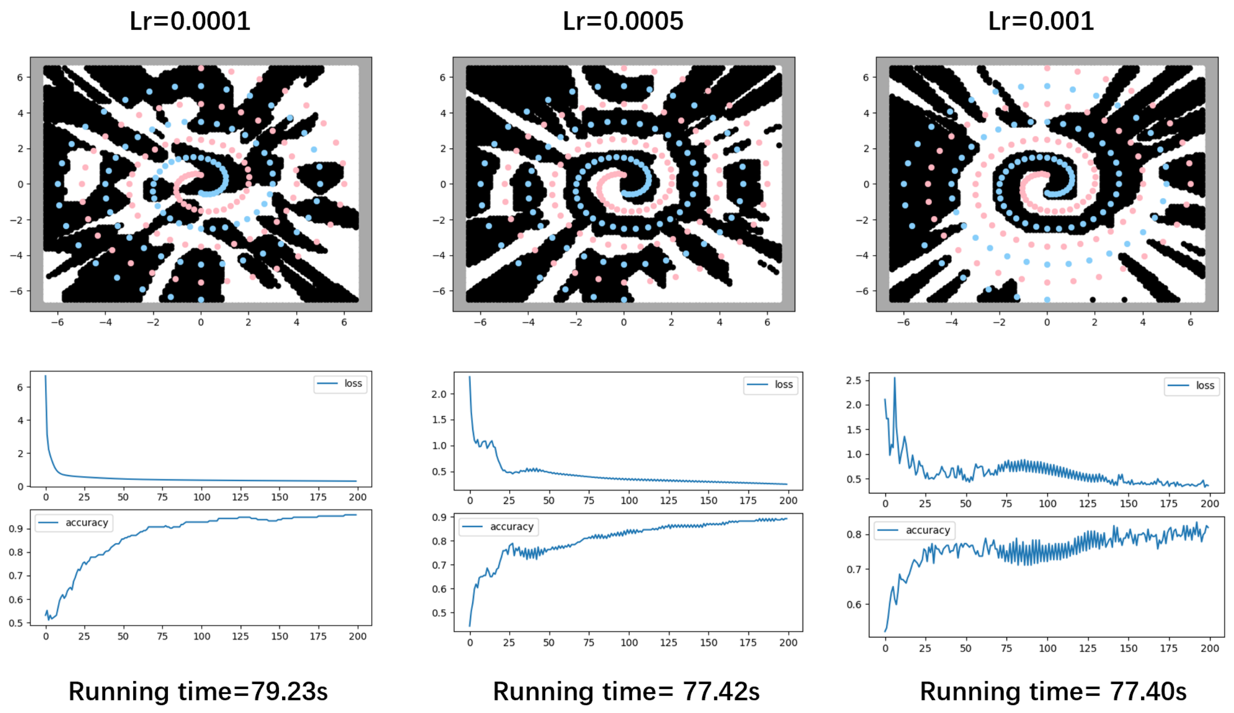

图 1: result of problem 2

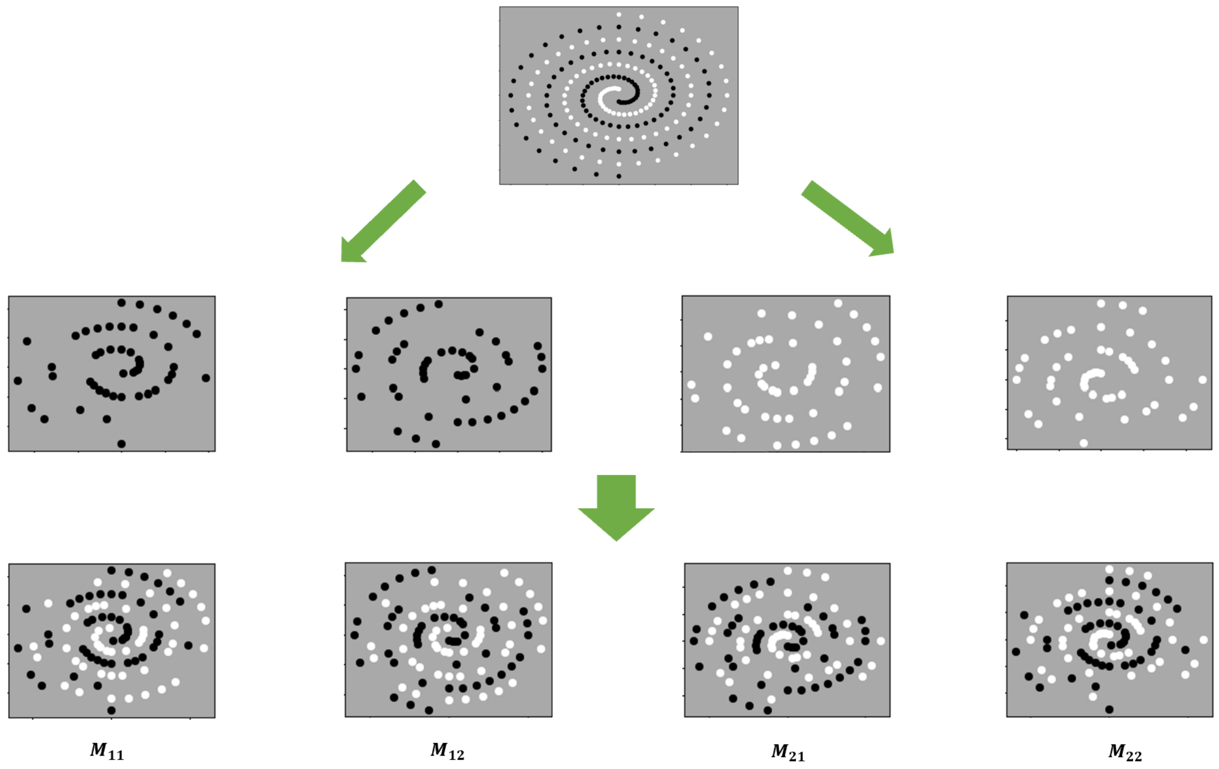

图 2: Sub-problems division

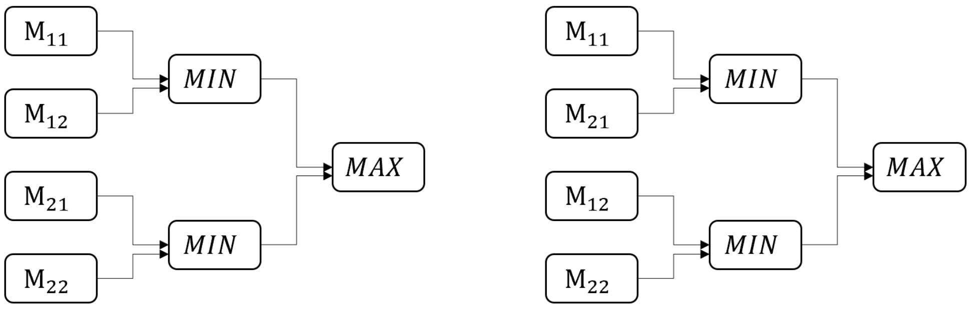

图 3: Two forms of min-max modular network

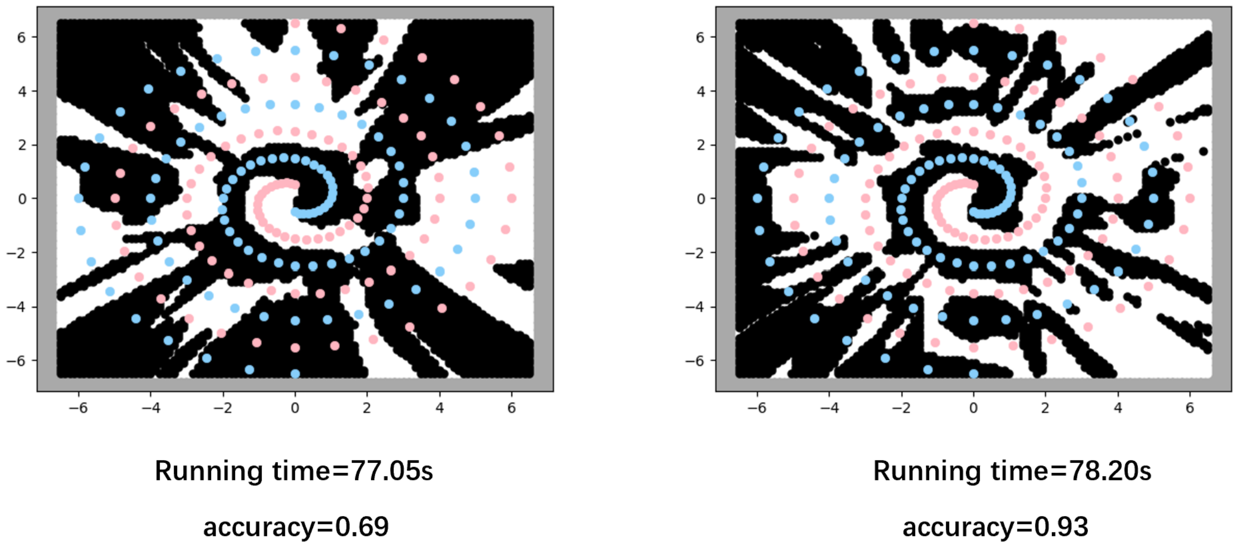

图 4: Decision boundaries and training time of two min-max modular networks