Neural Network Theory and Applications

Homework Assignment 2

I. Problem one

Similar to the conventional MLP, we can calculate the $\triangle u_{kj}(n)$s and $\triangle v_{kj}(n)$s of the network for both online learning and batch learning.

a) Online learning mode

$$\delta_j^k(n) = \begin{cases} e_j^K(n) f'\left(v_j^K(n)\right) & \text{for neuron } j \text{ in output layer K} \\ f'\left(v_j^k(n)\right) \sum_{i=1}^{N_{k+1}} \delta_j^{k+1}(n)\left(2u_{k+1,ji} x_{kj} + v_{k+1,ji}\right) & \text{for neuron } j \text{ in hidden layer k} \end{cases}$$

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

Thus, according to the delta rule:

$$\triangle u_{kj}(n) = \eta_1 \delta_j^k(n) x_{k-1}^2$$

$$\triangle v_{kj}(n) = \eta_2 \delta_j^k(n) x_{k-1}$$

b) Batch learning mode

For the batch learning mode, the values $\triangle U_{kj}(n)$ and $\triangle V_{kj}(n)$ are the average of correction $\triangle u_{kj}(n)$ and $\triangle v_{kj}(n)$ during one epoch of training, and will only update after that epoch.

$$\triangle U_{kj}(n) = \frac{1}{N} \sum_{t=1}^{N} \triangle u_{kjt}(n)$$

$$\triangle V_{kj}(n) = \frac{1}{N} \sum_{t=1}^{N} \triangle v_{kjt}(n)$$

II. Problem two

In this assignment, I use Java to implement the Back Propagation algorithm, and the codes are under the directory workspace\.

- The main data structure for the neural network is the *NeuronNetwork* class in the *network* package.(Figure 1)

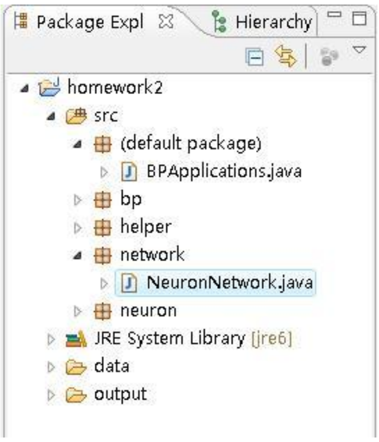

Figure 1. NeuronNetwork class

- The main implementation of BP algorithm is the *BPAlgorithmForMLQP* class. At the same time, the *OnlineBPAlgorithm* class and the *BatchBPAlgorithm* class, which are the subclasses of the *BPAlgorithmForMLQP* class, realize the online and batch BP algorithm separately.

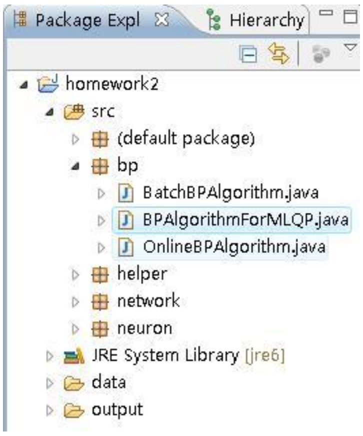

Figure 2. BPAlgorithmForMLQP class

III. Problem three

a) Online Learning Mode:

1) Case 1:

✓ uLearningRate = 1, vLearningRate = 1

✓ initial values: (u = v for each input)

Layer	hidden layer		output layer									
u	No. 1	No. 2	No. 1	No. 2	No. 3	No. 4	No. 5	No. 6	No. 7	No. 8	No. 9	No.10
values	-0.1	0.0	-0.5	-0.4	-0.3	-0.2	-0.1	0.0	0.1	0.2	0.3	0.4
bias	0.2											

✓ NumOfEpoches = 1000. (If we run the training data once, we call it an *epoch*. However, we do update weights after each single training data.)

✓ The training result: (The blank space between two colored areas is the decision boundaries)

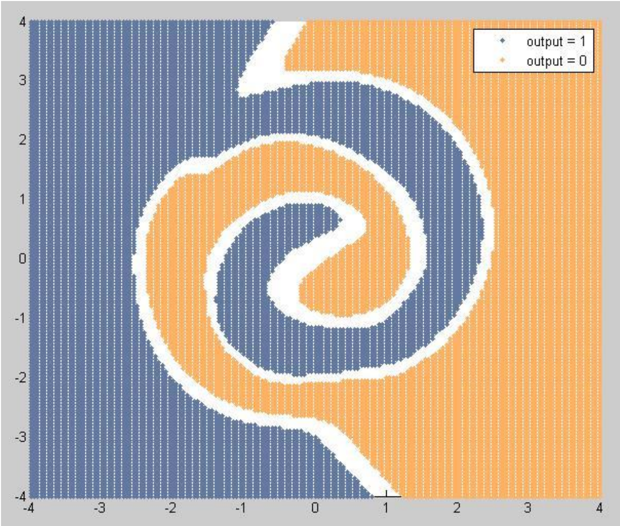

Figure 3. case 1: two spiral grid

2) Case 2:

- ✓ uLearningRate = 0.1, vLearningRate = 0.1
- ✓ initial values: (u = v for each input)

Layer	hidden layer		output layer									
u	No. 1	No. 2	No. 1	No. 2	No. 3	No. 4	No. 5	No. 6	No. 7	No. 8	No. 9	No.10
values	-0.1	0.0	-0.5	-0.4	-0.3	-0.2	-0.1	0.0	0.1	0.2	0.3	0.4
bias	0.2											

- ✓ NumOfEpoches = 1000.
- ✓ The training result: (The blank space between two colored areas is the decision boundaries)

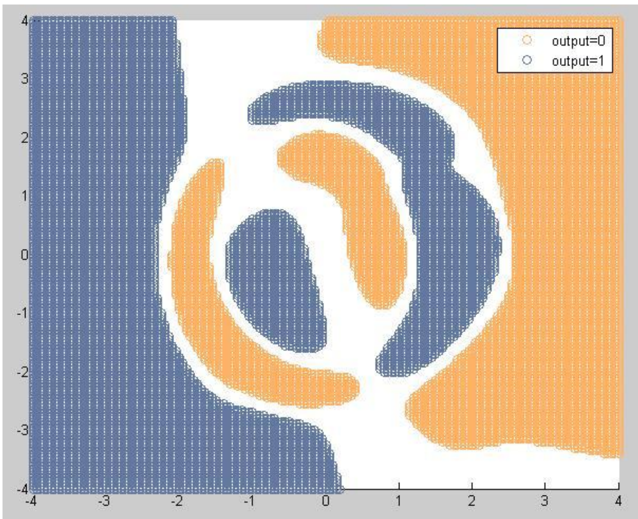

Figure 4. case 2: two spiral grid

3) Case 3:

- ✓ uLearningRate = 1.0, vLearningRate = 1.0
- ✓ initial values: (u = v for each input)

Layer	hidden layer		output layer									
u	No. 1	No. 2	No. 1	No. 2	No. 3	No. 4	No. 5	No. 6	No. 7	No. 8	No. 9	No.10
values	-1.0	0.0	-5.0	-4.0	-3.0	-2.0	-1.0	0.0	1.0	2.0	3.0	4.0
bias	0.2											

- ✓ NumOfEpoches = 1000.
- ✓ The training result: (The blank space between two colored areas is the decision boundaries)

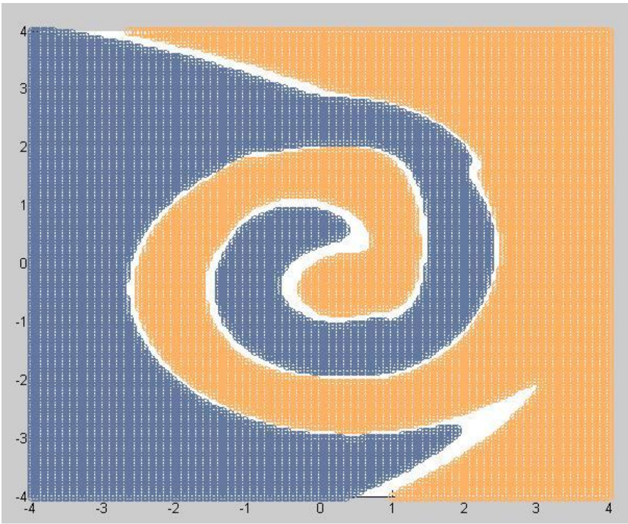

Figure 5. case 2: two spiral grid

4) MSEs compared

Here, I compare the MSEs of different learning rates, namely learning rate = 0.1, 0.5 and 1.0. The experiments are almost the same as case 1, except that I may use different learning rates. From Figure 5 below, we can see that smaller learning rate may lead to lower convergence speed. The network converges at epochs = 200 ~400 when the learning rate is 1 and 0.5, while the MSE is still declining even after 1000 epochs when learning rate is 0.1.

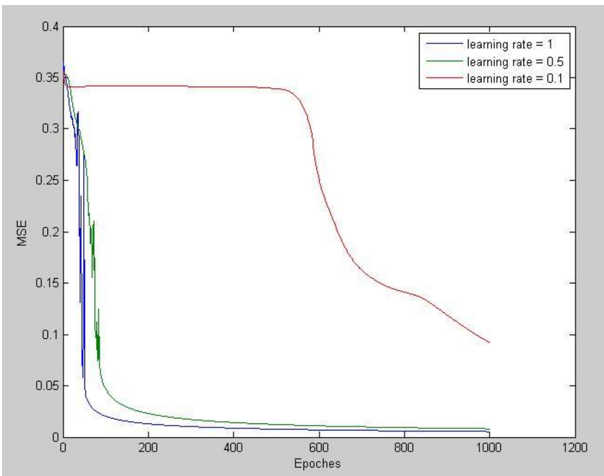

Figure 6. MSEs of different learning rate

b) Batch Learning Mode

In our batch mode, I use 25 training data as a batch. In Figure 7, I compare the two learning mode to see their convergence speed. According to the results, we can see that compared to online learning mode, the batch learning has a much slower convergence speed.

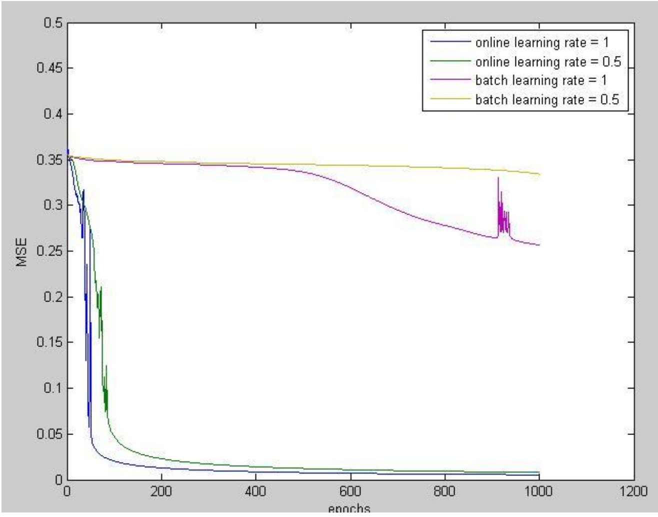

Figure 7. online and batch learning compared

IV. Problem four

The neural network I used in this problem is a three-layer network: one input layer, one hidden layer and one output layer. The hidden layer is composed of ten units. I've used different learning rates, and tried different initial values, but I still didn't get a satisfying result. The main reasons may be:

The input data are not always within the range from 0.0 to 1.0. So, I consider that we may need a way of normalizing the input data. However, I didn't get any good idea of normalization. For those input which are among the range from 0 to 1000, I simply divide each input by 1000.

- Online learning mode
  - ✓ initial values: (u = v for each input)

Layer	hidden layer	output layer
u	from No.1 to No.19	from No.1 to No.10
values	-1.0 -0.9 -0.8 -0.7 … 0.0 … 0.8 0.9 1.0	-0.5 -0.4 … 0 … 0.4 0.5
bias	0.2	

  - ✓ NumOfEpoches = 100000.
  - ✓ MSEs when different learning rates are used:

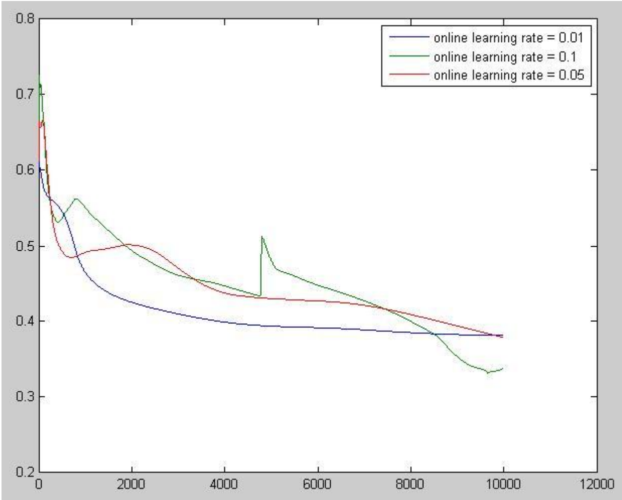

Figure 8. MSEs for different learning rate

V. Problem five

a) Online Learning Mode:

5) Case 1:
   - ✓ uLearningRate = 1, vLearningRate = 1
   - ✓ initial values: (u = v for each input)

Layer	hidden layer		output layer					
u	No.1	No.2	No.1	No.2	No.3	No.4	No.5	No.6
values	-0.1	0.0	-0.3	-0.2	-0.1	-0.0	0.1	0.2
bias	0.2							

   - ✓ NumOfEpoches = 1000.
   - ✓ The training result:

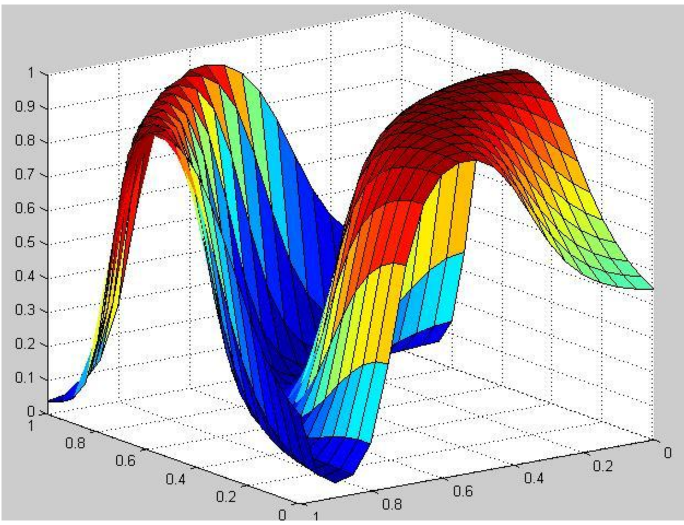

Figure 9. case 1: function surf

6) Case 2:
   - ✓ uLearningRate = 0.1, vLearningRate = 0.1
   - ✓ initial values: (u = v for each input)

Layer	hidden layer		output layer					
u	No.1	No.2	No.1	No.2	No.3	No.4	No.5	No.6
values	-0.1	0.0	-0.3	-0.2	-0.1	-0.0	0.1	0.2
bias	0.2							

- ✓ NumOfEpoches = 1000.
- ✓ The training result:

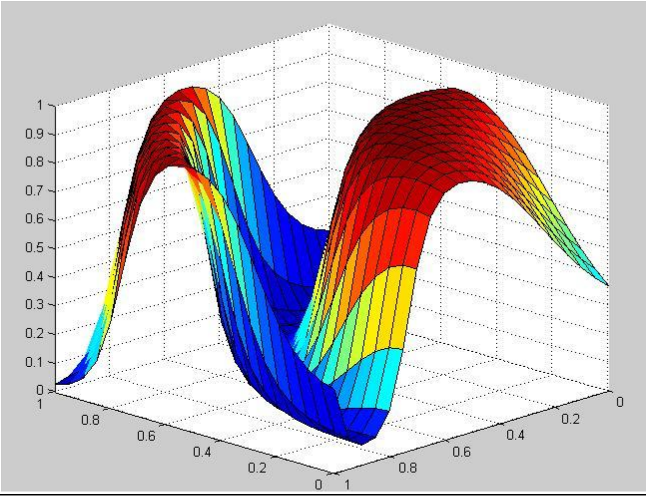

Figure 10. case 2: function surf

7) Case 3:

- ✓ uLearningRate = 1.0, vLearningRate = 1.0
- ✓ initial values: (u = v for each input)

Layer	hidden layer		output layer					
u	No.1	No.2	No.1	No.2	No.3	No.4	No.5	No.6
values	-1.0	0.0	-3.0	-2.0	-1.0	0.0	1.0	2.0
bias	0.2							

- ✓ NumOfEpoches = 1000.
- ✓ The training result: (The blank space between two colored areas is the decision boundaries)

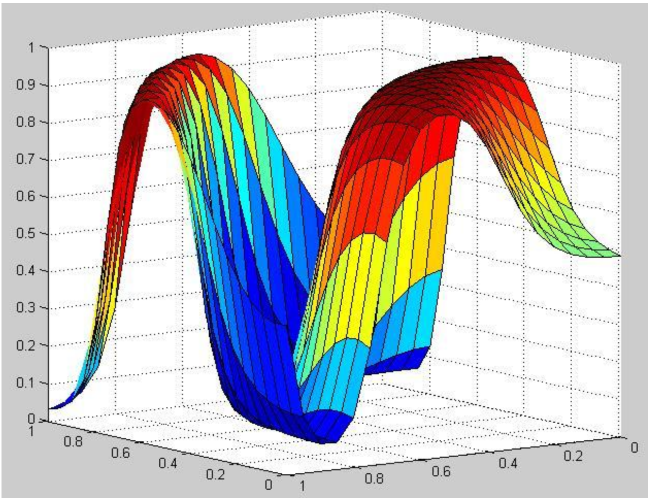

Figure 11. case 3: function suf

8) MSEs compared

Here, I compare the MSEs of the above three cases.

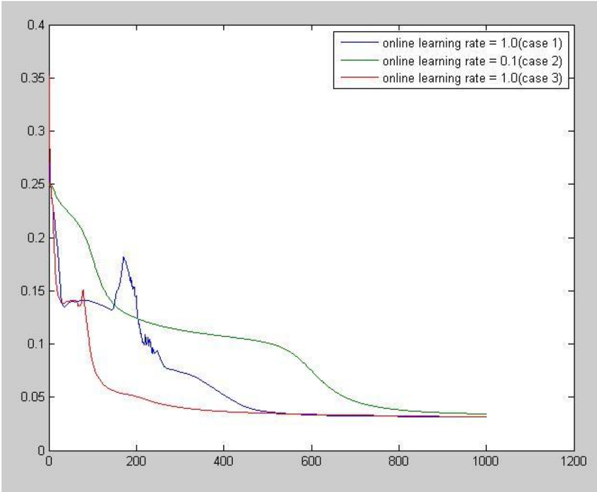

Figure 12. three cases' MSEs

b) Batch Learning Mode

In our batch mode, I use 25 training data as a batch.

9) Case 4:

- ✓ uLearningRate = 1.0, vLearningRate = 1.0
- ✓ initial values: (u = v for each input)

Layer	hidden layer		output layer					
u	No.1	No.2	No.1	No.2	No.3	No.4	No.5	No.6
values	-1.0	0.0	-3.0	-2.0	-1.0	0.0	1.0	2.0
bias	0.2							

- ✓ NumOfEpoches = 1000.
- ✓ The training result: (The blank space between two colored areas is the decision boundaries)

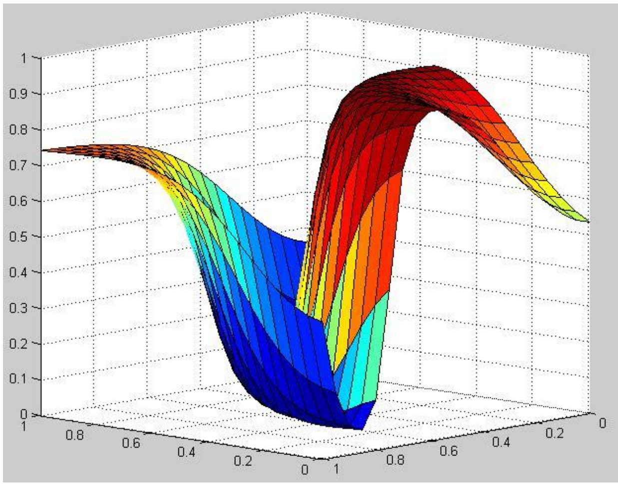

Figure 13. case 4: function suf

In Figure 13, I compare the two learning modes (learning rate = 1.0) to see their convergence speed.

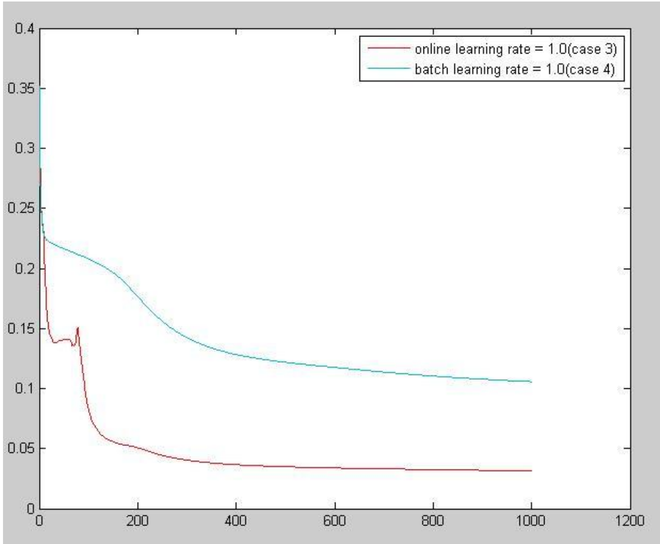

Figure 14. online and batch learning compared