Assignment II-MLQP

Yifeng Chen 5140309458

March 8, 2017

1 Derivation

MLQP, an abbreviation for Multilayer Quadratic Perceptron,is an structure proposed by Prof.Bao-Liang Lu. The structure is defined as follows:

$$X_k = f(U_k X_{k-1}^2 + V_k X_{k-1} + b_k) \tag{1.1}$$

where X_k is the output of the k^{th} layer, U_k,V_k,b_k are the parameters of the k^{th} layer,while f is the activation function.Suppose H_k is the number of perceptrons in the k^{th} layer,and M is the number of samples,then the size of the parameters are $U_k : H_k \times H_{k-1}$,$V_k : H_k \times H_{k-1}$,b_k : H_k(For the first layer, H_0 is exactly the input dimension).The notation we use is as follows:

$$\begin{aligned} P_k &= U_k X_{k-1}^2 + V_k X_{k-1} + b_k \\ Q_k &= U_k X_{k-1}^2 \\ L_k &= V_k X_{k-1} \\ I_k &= X_{k-1}^2 \end{aligned} \tag{1.2}$$

1.1 Batch Learning

Batch learning techniques generate the best predictor by learning on the entire training data set at once.For the k^{th} layer,we have:

$$\begin{aligned} \frac{\partial X_k}{\partial X_{k-1}} &= \frac{\partial X_k}{\partial P_k}\frac{\partial P_k}{\partial Q_k}\frac{\partial Q_k}{\partial I_k}\frac{\partial I_k}{\partial X_{K-1}} + \frac{\partial X_k}{\partial P_k}\frac{\partial P_k}{\partial L_k}\frac{\partial L_k}{\partial X_{k-1}} \\ \frac{\partial X_k}{\partial U_k} &= \frac{\partial X_k}{\partial P_k}\frac{\partial P_k}{\partial Q_k}\frac{\partial Q_k}{\partial U_k} \\ \frac{\partial X_k}{\partial V_k} &= \frac{\partial X_k}{\partial P_k}\frac{\partial P_k}{\partial L_k}\frac{\partial L_k}{\partial V_k} \\ \frac{\partial X_k}{\partial b_k} &= \frac{\partial X_k}{\partial P_k}\frac{\partial P_k}{\partial b_k} \end{aligned} \tag{1.3}$$

Applying a function to a matrix is equivalent to applying the function to its every element.Then we can get:

$$
\begin{aligned}
g(x) &= \frac{e^{-x}}{(1+e^{-x})^2} \\
\frac{\partial X_k}{\partial X_{k-1}} &= 2U^T g(P_k) X_{k-1} + V^T g(P_k) \\
\frac{\partial X_k}{\partial U_k} &= g(P_k)(X_{k-1}^2)^T \\
\frac{\partial X_k}{\partial V_k} &= g(P_k) X_{k-1}^T \\
\left(\frac{\partial X_k}{\partial b_k}\right)_i &= \sum_{j=1}^{M} g(P_k)_{i,j} (i = 1 \ldots H_k)
\end{aligned}
\tag{1.4}
$$

For the final F^{th} layer,we will get the final scores X_F of $H_F \times M$,M is the number of samples(as defined before).And we also have the expected output D.According to the cost function:$\epsilon_{av} = \frac{1}{2M}\sum_{n=1}^{M} e_n^2 = \frac{1}{2M}\sum_{n=1}^{M}(D - X_F)_n^2$:

$$
\frac{\partial \epsilon_{av}}{\partial X_F} = \frac{1}{M}\sum_{n=1}^{M} e_n \frac{\partial \epsilon_{e_n}}{\partial X_F} = -\frac{1}{M}\sum_{n=1}^{M}(D - X_F) \tag{1.5}
$$

Now we get the gradients first from 1.5,and then we can use 1.4 to do BP.

1.2 Online learning

Online learning is a method in which data becomes available in a sequential order and is used to update our best predictor for future data at each step.It is nearly the same as the way we do batch learning,except that M = 1(M is number of samples).So 1.3,1.4 still holds by setting M=1,which is:

$$
\begin{aligned}
g(x) &= \frac{e^{-x}}{(1+e^{-x})^2} \\
\frac{\partial X_k}{\partial X_{k-1}} &= 2U^T g(P_k) X_{k-1} + V^T g(P_k) \\
\frac{\partial X_k}{\partial U_k} &= g(P_k)(X_{k-1}^2)^T \\
\frac{\partial X_k}{\partial V_k} &= g(P_k) X_{k-1}^T \\
\left(\frac{\partial X_k}{\partial b_k}\right) &= g(P_k)
\end{aligned}
\tag{1.6}
$$

For the final F^{th} layer,we will get the final scores X_F of $H_F \times 1$.And we also have the expected output D.According to the cost function:$\epsilon = \frac{1}{2}e_n^2 = \frac{1}{2}(D - X_F)^2$:

$$
\frac{\partial \epsilon_{av}}{\partial X_F} = e = -(D - X_F) \tag{1.7}
$$

Now we get the gradients first from 1.7,and then we can use 1.6 to do BP.

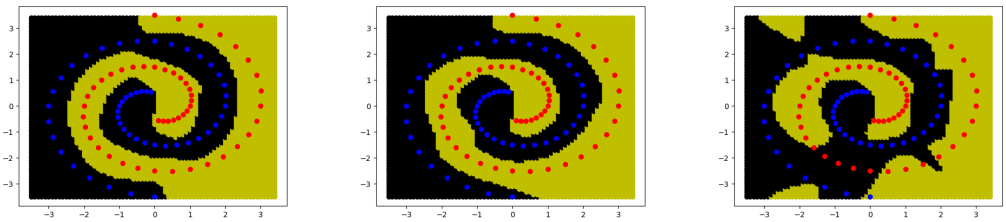
2 Train & Test

Using merely single hidden layer with the number of perceptrons set as 10,we run this model upon data of two spirals,trying to separate them.To show the influence of learning rate,we choose three different learning rates:1e-3,1e-2,1e-1.We define the time for the model to finally converge as train time.The results are as follows:

Table 2.1: Influence of Learning Rate on Model Performance

Learning rate	Train Time(s)	Accuracy on Train	Accuracy on Test
1e-1	1.21996903419	1.0	1.0
1e-2	7.48591518402	1.0	1.0
1e-3	47.5390191078	0.97	0.96

Unsurprisingly, 1e-1 converges most quickly.In my implementation,i use momentum to update parameters,which gives a better tolerance for large learning rates.However,when learning rate is set to 1e-3,the model falls into some local optimal and fails to predict all answers correctly.



(a) learning rate = 1e-1 (b) learning rate = 1e-2 (c) learning rate = 1e-3

Figure 2.1: Separatrices Contrast

As shown in Figure 2.1,when learning rate is set to 1e-1,1e-2,the hyperplane perfectly separates these two spirals from each other.However,when learning rate is set to 1e-3,the separatrix looks extremely coarse in contrast to the former ones.Since a smoother separatrix usually indicates a stronger ability of generalization,the model trained using appropriate learning rate tends to be more robust.