Programming 2 for ECSE 6965

Lingyu Zhang, RIN:661950104

Mar 16, 2018

Theoretical part

The objective of this problem is that we utilize the Neural network learning method to compute the optimal weight and bias as the parameters of multi-class regression. For this problem, we have the training dataset which contains 50000 samples, each sample is a $28 * 28$ vector, so we construct the data as $784 * 1$ vector For the label y, we construct it as one-of-k coding, if label is 2, then y is denoted as below

$$y = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

For forward propagation, the computation process is as below:

$$H^1 = Relu\Big((W^1)^t x + W_0^1\Big)$$

$$H^2 = Relu\Big((W^2)^t H^1 + W_0^2\Big)$$

$$\hat{y} = \sigma_M\Big((W^3)^t H^2 + W_0^3\Big)$$

We use squared loss function for the output. The loss function is represented as below

$$L = \frac{1}{2}(y - \hat{y})^t (y - \hat{y})$$

For backpropagation, the computation process is as below: Since Y is a $K * 1$ vector, the output has K notes, then for the output layer, we use sigmoid function σ_m as the activation function.

$$\nabla \hat{y}[m] = \frac{\partial\Big(\frac{1}{2}(y[m] - \hat{y}[m])^t(y - \hat{y}[m])\Big)}{\partial \hat{y}[m]} = -(y[m] - \hat{y}[m])$$

For $m - th$ sample, we use $z_3[m]$ to denote the output of the second hidden layer.

$$z_3[m] = (W^3)^t H^2[m] + W_0^3$$

$$\hat{y}[m] = \sigma_m(z_3[m]) = \begin{bmatrix} \sigma_m(z_3[m][1]) \\ \sigma_m(z_3[m][2]) \\ ... \\ \sigma_m(z_3[m][K]) \end{bmatrix}$$

where

$$z_3[m][k] = (W_k^3)^t H^2[m] + W_0^3[k]$$

$$\sigma_m(z_3[m][k]) = \frac{exp((W_k^3)^t H^2[m] + W_0^3[k])}{\sum_{j=1}^{K} exp((W_j^3)^t H^2[m] + W_0^3[j])}$$

Therefore

$$\nabla W^3 = \frac{\partial \hat{y}[m]}{\partial W^3} \nabla \hat{y}[m] = \sum_{k=1}^{10} \frac{\partial \hat{y}[m][k]}{\partial W^3} \nabla \hat{y}[m][k] = \sum_{k=1}^{10} \begin{bmatrix} \frac{\partial \hat{y}[m][k]}{\partial W_1^3} & \frac{\partial \hat{y}[m][k]}{\partial W_2^3} & ... & \frac{\partial \hat{y}[m][k]}{\partial W_{10}^3} \end{bmatrix} \nabla \hat{y}[m][k]$$

Now we compute $\frac{\partial \hat{y}[m][k]}{\partial W_i^3}$

$$\frac{\partial \hat{y}[m][k]}{\partial W_i^3} = \frac{\partial \sigma_M(z_3[m][k])}{\partial W_i^3} = \frac{\partial \frac{exp(z_3[m][k])}{\sum_{j=1}^{K} exp(z_3[m][j])}}{\partial W_i^3}$$

where

$$\frac{\partial \sigma_M(z_3[m][k])}{\partial W_i^3} = \begin{cases} \sigma_M(z_3[m][i])(1 - \sigma_M(z_3[m][i]))H^2[m] & \text{if } i = \text{k} \\ -\sigma_M(z_3[m][k])\sigma_M(z_3[m][i])H^2[m] & \text{else} \end{cases}$$

Now we compute ∇W_0^3

$$\nabla W_0^3 = \frac{\partial \hat{y}[m]}{\partial W_0^3} \nabla \hat{y}[m]$$

$$\frac{\partial \hat{y}[m]}{\partial W_0^3} = \begin{bmatrix} \frac{\partial \hat{y}[m][1]}{\partial W_0^3} & \frac{\partial \hat{y}[m][2]}{\partial W_0^3} & ... & \frac{\partial \hat{y}[m][K]}{\partial W_0^3} \end{bmatrix}$$

Now we compute $\frac{\partial \hat{y}[m][i]}{\partial W_0^3}$

$$\frac{\partial \hat{y}[m][i]}{\partial W_0^3} = \begin{bmatrix} \frac{\partial \hat{y}[m][i]}{\partial W_0^3[1]} \\ \frac{\partial \hat{y}[m][i]}{\partial W_0^3[2]} \\ ... \\ \frac{\partial \hat{y}[m][i]}{\partial W_0^3[K]} \end{bmatrix}$$

Now we compute $\frac{\partial \hat{y}[m][i]}{\partial W_0^3[k]}$

$$\frac{\partial \hat{y}[m][i]}{\partial W_0^3[k]} = \frac{\partial \sigma_M(z_3[m][i])}{\partial W_0^3[k]}$$

where

$$\frac{\partial \sigma_M(z_3[m][i])}{\partial W_0^3[k]} = \begin{cases} \sigma_M(z_3[m][k])(1 - \sigma_M(z_3[m][k])) & \text{if } i = \text{k} \\ -\sigma_M(z_3[m][i])\sigma_M(z_3[m][k]) & \text{else} \end{cases}$$

Now we compute ∇H^2

$$\nabla H^2 = \frac{\partial \hat{y}[m]}{\partial H^2} \nabla \hat{y}[m] = \sum_{k=1}^{K} \frac{\partial \hat{y}[m][k]}{\partial H^2} \nabla \hat{y}[m][k]$$

$$= \sum_{k=1}^{K} \left[\hat{y}[m][k] W_k^3 - \hat{y}[m][k] \sum_{j=1}^{K} \hat{y}[m][j] W_j^3 \right] \nabla \hat{y}[m][k]$$

From second hidden layer to the first hidden layer

$$(H^2[m])^{N_2 * 1} = ReLu((W^2)^t H^1[m] + W_0^2)$$

Let $(z_2[m])^{N_2 * 1} = (W^2)^t H^1[m] + W_0^2$ then we have $(H^2[m])^{N_2 * 1} = ReLu(z_2[m])$ therefore

$$\nabla W^2 = \frac{\partial H^2}{\partial W^2} \nabla H^2 = \sum_{k=1}^{N_2} \frac{\partial ReLu(z_2[m][k])}{W^2} \nabla H^2[m][k]$$

$$\nabla W^2 = \sum_{k=1}^{N_2} \left[\frac{\partial ReLu(z_2[m][k])}{W_1^2} \quad \frac{\partial ReLu(z_2[m][k])}{W_2^2} \quad \cdots \quad \frac{\partial ReLu(z_2[m][k])}{W_{N_2}^2} \right] \nabla H^2[m][k]$$

For the i-th element

$$\frac{\partial ReLu(z_2[m][k])}{W_i^2} = \begin{cases} H^1[m] & \text{if } i = k \text{ and } z_2[m][k] > 0 \\ 0 & \text{else} \end{cases}$$

Now we compute ∇W_0^2

$$\nabla W_0^2 = \frac{\partial H^2}{\partial W_0^2} \nabla H^2 = \sum_{k=1}^{N_2} \frac{\partial ReLu(z_2[m][k])}{W_0^2} \nabla H^2[m][k]$$

$$\nabla W_0^2 = \sum_{k=1}^{N_2} \begin{bmatrix} \frac{\partial ReLu(z_2[m][k])}{W_0^2[1]} \\ \frac{\partial ReLu(z_2[m][k])}{W_0^2[2]} \\ \cdots \\ \frac{\partial ReLu(z_2[m][k])}{W_0^2[N_2]} \end{bmatrix} \nabla H^2[m][k]$$

For the i-th element

$$\frac{\partial ReLu(z_2[m][k])}{W_0^2[i]} = \begin{cases} 1 & \text{if } i = k \text{ and } z_2[m][k] > 0 \\ 0 & \text{else} \end{cases}$$

Now we compute $\triangledown H^1$

$$\triangledown H^1[m] = \frac{\partial H^2}{\partial H^1} \triangledown H^2 = \sum_{k=1}^{N_2} \frac{\partial ReLu(z_2[m][k])}{H^1} \triangledown H^2[m][k]$$

For the k-th element

$$\frac{\partial ReLu(z_2[m][k])}{\partial H^1} = \begin{cases} W_k^2 & \text{if } z_2[m][k] > 0 \\ 0 & \text{else} \end{cases}$$

From the first hidden layer to the input layer

$$(H^1[m])^{N_1 * 1} = ReLu((W^1)^t X[m] + W_0^2)$$

Let $(z_1[m])^{N_1 * 1} = (W^1)^t X[m] + W_0^1$ then we have $(H^1[m])^{N_1 * 1} = ReLu(z_1[m])$ Therefore

$$\triangledown W^1 = \frac{\partial H^1}{\partial W^1} \triangledown H^1 = \sum_{k=1}^{N_1} \frac{\partial ReLu(z_1[m][k])}{W^1} \triangledown H^1[m][k]$$

$$\triangledown W^1 = \sum_{k=1}^{N_1} \left[\frac{\partial ReLu(z_1[m][k])}{W_1^1} \quad \frac{\partial ReLu(z_1[m][k])}{W_2^1} \quad \cdots \quad \frac{\partial ReLu(z_1[m][k])}{W_{N_1}^1} \right] \triangledown H^1[m][k]$$

For the i-th element

$$\frac{\partial ReLu(z_1[m][k])}{W_i^1} = \begin{cases} X[m] & \text{if } i = k \text{ and } z_1[m][k] > 0 \\ 0 & \text{else} \end{cases}$$

Now we compute $\triangledown W_0^1$

$$\triangledown W_0^1 = \frac{\partial H^1}{\partial W_0^1} \triangledown H^1 = \sum_{k=1}^{N_1} \frac{\partial ReLu(z_1[m][k])}{W_0^1} \triangledown H^1[m][k]$$

$$\triangledown W_0^1 = \sum_{k=1}^{N_1} \begin{bmatrix} \frac{\partial ReLu(z_1[m][k])}{W_0^1[1]} \\ \frac{\partial ReLu(z_1[m][k])}{W_0^1[2]} \\ \cdots \\ \frac{\partial ReLu(z_1[m][k])}{W_0^1[N_2]} \end{bmatrix} \triangledown H^1[m][k]$$

For the i-th element

$$\frac{\partial ReLu(z_1[m][k])}{W_0^1[i]} = \begin{cases} 1 & \text{if } i = k \text{ and } z_1[m][k] > 0 \\ 0 & \text{else} \end{cases}$$

Then for each iteration, we update the parameters as below:

$$W^3_{new} = W^3 - \eta \nabla W^3$$

$$W^3_{0,new} = W^3_0 - \eta \nabla W^3_0$$

$$W^2_{new} = W^2 - \eta \nabla W^2$$

$$W^2_{0,new} = W^2_0 - \eta \nabla W^2_0$$

$$W^1_{new} = W^1 - \eta \nabla W^1$$

$$W^1_{0,new} = W^1_0 - \eta \nabla W^1_0$$

Model architecture

The whole model consists of one input layer, two hidden layers and one output layer. For hidden layer, we use ReLu function and for output layer, we use softmax function. The input layer has 784 nodes, the first hidden layer has 100 nodes, the second hidden layer has 100 nodes, the output layer has 10 nodes.

Experimental part

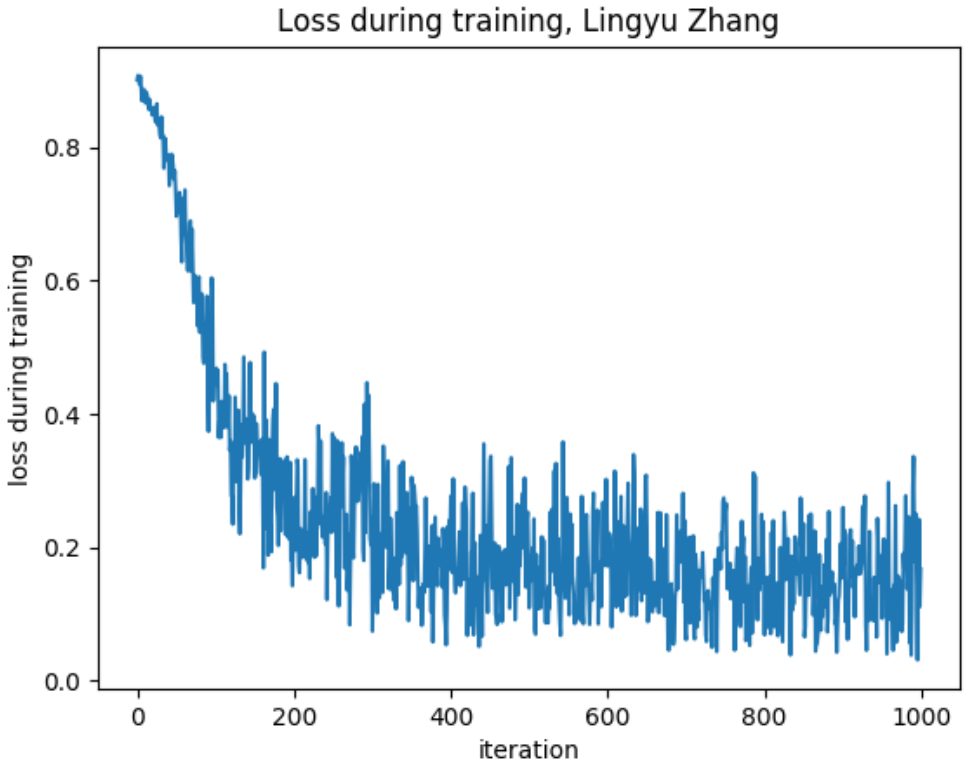
Firstly we initialize the parameters, the weight matrix is initialized by normal distribution with mean = 0 and standard derivation = 0.1 For the bias vector, we initialized them as zero vectors. We tune the hyper-parameters and the optimal ones are as below: The loss has already converged after one training epoch, so we choose the number of training epochs as 1, we firstly use a learning rate and then tune it based on the speed of convergence.

$$\text{learning rate } \eta = 0.2$$

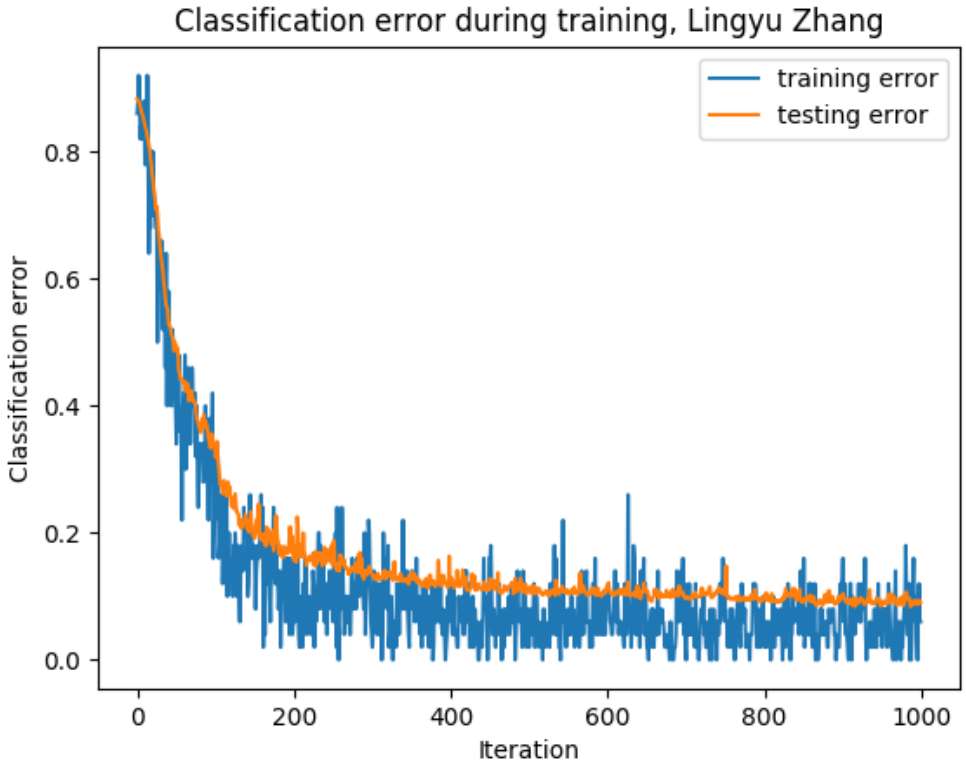
$$\text{batch size} = 50$$

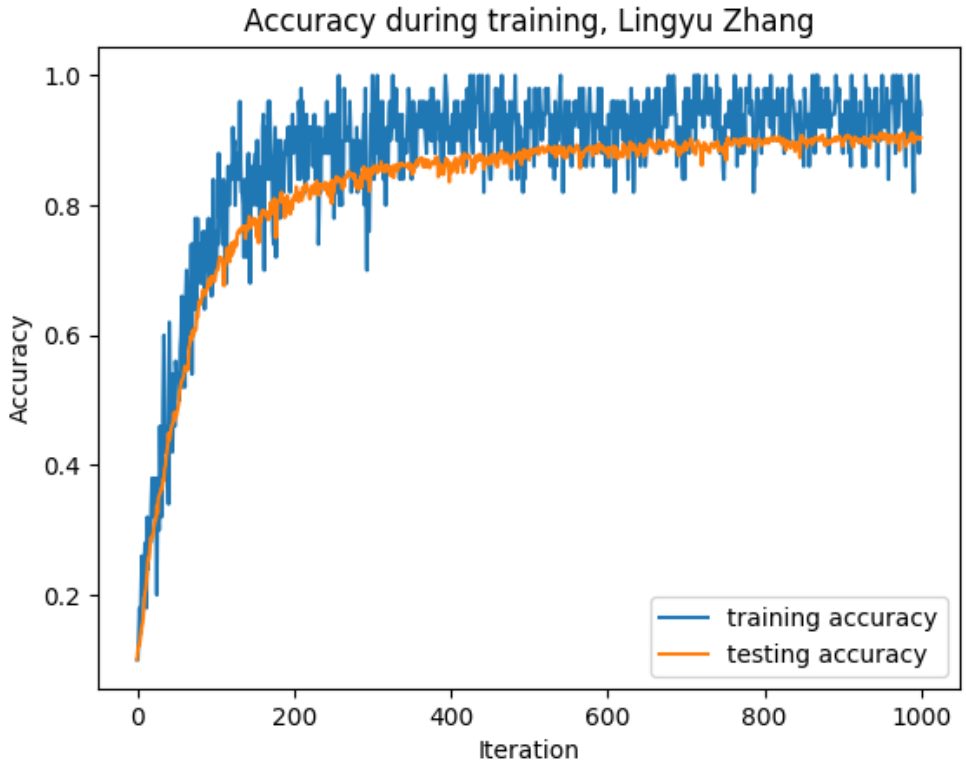
$$\text{training epochs} = 1$$

Below is the training loss over iterations



We can find that the loss is going down with the training iteration increases and finally converged. Below is the prediction accuracy and prediction error plot for training set and testing set over iterations


Classification error during training, Lingyu Zhang
training error
testing error
Classification error
0.0
0.2
0.4
0.6
0.8
0
200
400
600
800
1000
Iteration



We can find that the prediction accuracy for both training set and testing set is increasing with the training process going on, and finally converged. And the training accuracy is slightly higher than the testing accuracy, which is reasonable.

Classification result

Belos is the confusion matrix for the testing result

```
[[477   0   1   2   0   9   4   2   3   4]
 [  0 521   2   3   0   4   1   1   0   2]
 [  2   4 432  14   6   5   8  14   5   7]
 [  1   0   4 478   0  21   0   6   5   6]
 [  1   5   2   1 399   2   3   5   0  66]
 [  3   5   2  25   5 398   8   5   3   3]
 [  2   1   5   0   4  19 453   0   0   1]
 [  2   4   6  11   4   2   0 489   1   8]
 [  0   7   8  25   0  26   3   7 422  17]
 [  5   2   0   6   3   2   0  23   1 436]]
```

Therefore,

$$error(digit = 0) = 1 - \frac{477}{477 + 0 + 1 + 2 + 0 + 9 + 4 + 2 + 3 + 4} = 4.9800\%$$

$$error(digit = 1) = 1 - \frac{521}{0 + 521 + 2 + 3 + 0 + 4 + 1 + 1 + 0 + 2} = 2.4344\%$$

$$error(digit = 2) = 1 - \frac{432}{2 + 4 + 432 + 14 + 6 + 5 + 8 + 14 + 5 + 7} = 13.078\%$$

$$error(digit = 3) = 1 - \frac{478}{1 + 0 + 4 + 478 + 0 + 21 + 0 + 6 + 5 + 6} = 8.2533\%$$

$$error(digit = 4) = 1 - \frac{399}{1 + 5 + 2 + 1 + 399 + 2 + 3 + 5 + 0 + 66} = 2.3\%$$

$$error(digit = 5) = 1 - \frac{398}{3 + 5 + 2 + 25 + 5 + 398 + 8 + 5 + 3 + 3} = 17.5619\%$$

$$error(digit = 6) = 1 - \frac{453}{2 + 1 + 5 + 0 + 4 + 19 + 453 + 0 + 0 + 1} = 6.5979\%$$

$$error(digit = 7) = 1 - \frac{489}{2 + 4 + 6 + 11 + 4 + 2 + 0 + 489 + 1 + 8} = 7.2106\%$$

$$error(digit = 8) = 1 - \frac{422}{0 + 7 + 8 + 25 + 0 + 26 + 3 + 7 + 422 + 17} = 18.05\%$$

$$error(digit = 9) = 1 - \frac{436}{5 + 2 + 0 + 6 + 3 + 2 + 0 + 23 + 1 + 436} = 8.7866\%$$

$$avg(error) = 1 - \frac{477 + 521 + 432 + 478 + 399 + 398 + 453 + 489 + 422 + 436}{5000} = 9.9\%$$

We can find that the average classification error is 9.9%