CS289 Homework 6

SID: 24978491

April 21, 2015

1. Derivation of stochastic gradient updates of W_1 and W_2

Mean-squared Error:

The cost function is

$$J = \frac{1}{N}\sum_{i=1}^{n}\frac{1}{2}\sum_{k=1}^{n_{out}}\left(y_k^i - o_k^i\right)^2$$

where for a data point i, y_k^i is the label for the ith point in class k. x^i is a vector of features of data point i. o_k^i is the output of hidden layer. Denote the output of the first layer as v^i. We have

$$v^i = tanh\left(W^{(1)T}x^i\right)$$

$$o_k^i = \frac{1}{1+exp\left(-W_k^{(2)T}v^i\right)}$$

We know that

$$\frac{\partial J}{\partial o_k^i} = \frac{1}{N}\left(-y_k^i + o_k^i\right)$$

$$\frac{\partial o_k^i}{\partial v_j^i} = \frac{exp\left(-W_k^{(2)T}v^i\right)W_{jk}^{(2)}}{\left(1+exp\left(-W_k^{(2)T}v^i\right)\right)^2}$$

$$\frac{\partial o_k^i}{\partial W_{jk}^{(2)}} = \frac{exp\left(-W_k^{(2)T}v^i\right)v_j^i}{\left(1+exp\left(-W_k^{(2)T}v^i\right)\right)^2}$$

$$\frac{\partial v_j^i}{\partial W_{mj}^{(1)}} = \left(1-tanh\left(W_j^{(1)T}x^i\right)^2\right)x_m^i$$

Then we will get:

$$\frac{\partial J}{\partial W_{jk}^{(2)}} = \frac{\partial J}{\partial o_k^i}\frac{\partial o_k^i}{\partial W_{jk}^{(2)}} = \frac{1}{N}\left(-y_k^i + o_k^i\right)\frac{exp\left(-W_k^{(2)T}v^i\right)v_j^i}{\left(1+exp\left(-W_k^{(2)T}v^i\right)\right)^2}$$

$$\frac{\partial J}{\partial W_{mj}^{(1)}} = \frac{\partial J}{\partial o_k^i}\frac{\partial o_k^i}{\partial v_j^i}\frac{\partial v_j^i}{\partial W_{mj}^{(1)}} = \frac{1}{N}\left(-y_k^i + o_k^i\right)\frac{exp\left(-W_k^{(2)T}v^i\right)W_{jk}^{(2)}}{\left(1+exp\left(-W_k^{(2)T}v^i\right)\right)^2}\left(1-tanh\left(W_j^{(1)T}x^i\right)^2\right)x_m^i$$

Cross-Entropy Error:

The cost function is

$$J = \frac{1}{N}\sum_{i=1}^{n}\sum_{k=1}^{n_{out}} -\left(y_k^i ln\left(o_k^i\right) + \left(1-y_k^i\right) ln\left(1-o_k^i\right)\right)$$

where for a data point i, y_k^i is the label for the ith point in class k. x^i is a vector of features of data point i. o_k^i is the output of hidden layer. Denote the output of the first layer as v^i. We have

$$v^i = tanh\left(W^{(1)T}x^i\right)$$

$$o_k^i = \frac{1}{1+exp\left(-W_k^{(2)T}v^i\right)}$$

We know that

$$\frac{\partial J}{\partial o_k^i} = \frac{1}{N}\left(-\frac{y_k^i}{o_k^i} + \frac{1-y_k^i}{1-o_k^i}\right)$$

$$\frac{\partial o_k^i}{\partial v_j^i} = \frac{exp\left(-W_k^{(2)T}v^i\right)W_{jk}^{(2)}}{\left(1+exp\left(-W_k^{(2)T}v^i\right)\right)^2}$$

$$\frac{\partial o_k^i}{\partial W_{jk}^{(2)}} = \frac{exp\left(-W_k^{(2)T}v^i\right)v_j^i}{\left(1+exp\left(-W_k^{(2)T}v^i\right)\right)^2}$$

$$\frac{\partial v_j^i}{\partial W_{mj}^{(1)}} = \left(1-tanh\left(W_j^{(1)T}x^i\right)^2\right)x_m^i$$

Then we will get:

$$\frac{\partial J}{\partial W_{jk}^{(2)}} = \frac{\partial J}{\partial o_k^i}\frac{\partial o_k^i}{\partial W_{jk}^{(2)}} = \frac{1}{N}\left(-\frac{y_k^i}{o_k^i} + \frac{1-y_k^i}{1-o_k^i}\right)\frac{exp\left(-W_k^{(2)T}v^i\right)v_j^i}{\left(1+exp\left(-W_k^{(2)T}v^i\right)\right)^2}$$

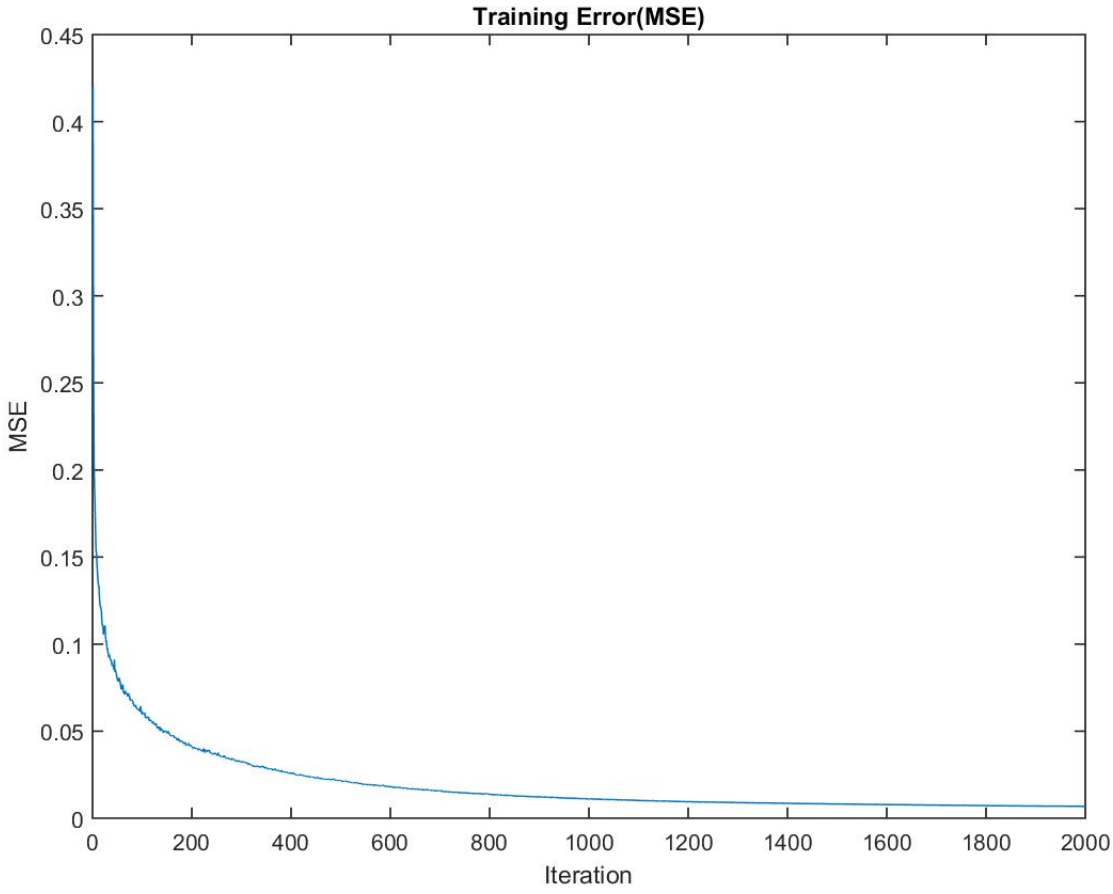
$$\frac{\partial J}{\partial W_{mj}^{(1)}} = \frac{\partial J}{\partial o_k^i}\frac{\partial o_k^i}{\partial v_j^i}\frac{\partial v_j^i}{\partial W_{mj}^{(1)}} = \frac{1}{N}\left(-\frac{y_k^i}{o_k^i} + \frac{1-y_k^i}{1-o_k^i}\right)\frac{exp\left(-W_k^{(2)T}v^i\right)W_{jk}^{(2)}}{\left(1+exp\left(-W_k^{(2)T}v^i\right)\right)^2}\left(1-tanh\left(W_j^{(1)T}x^i\right)^2\right)x_m^i$$

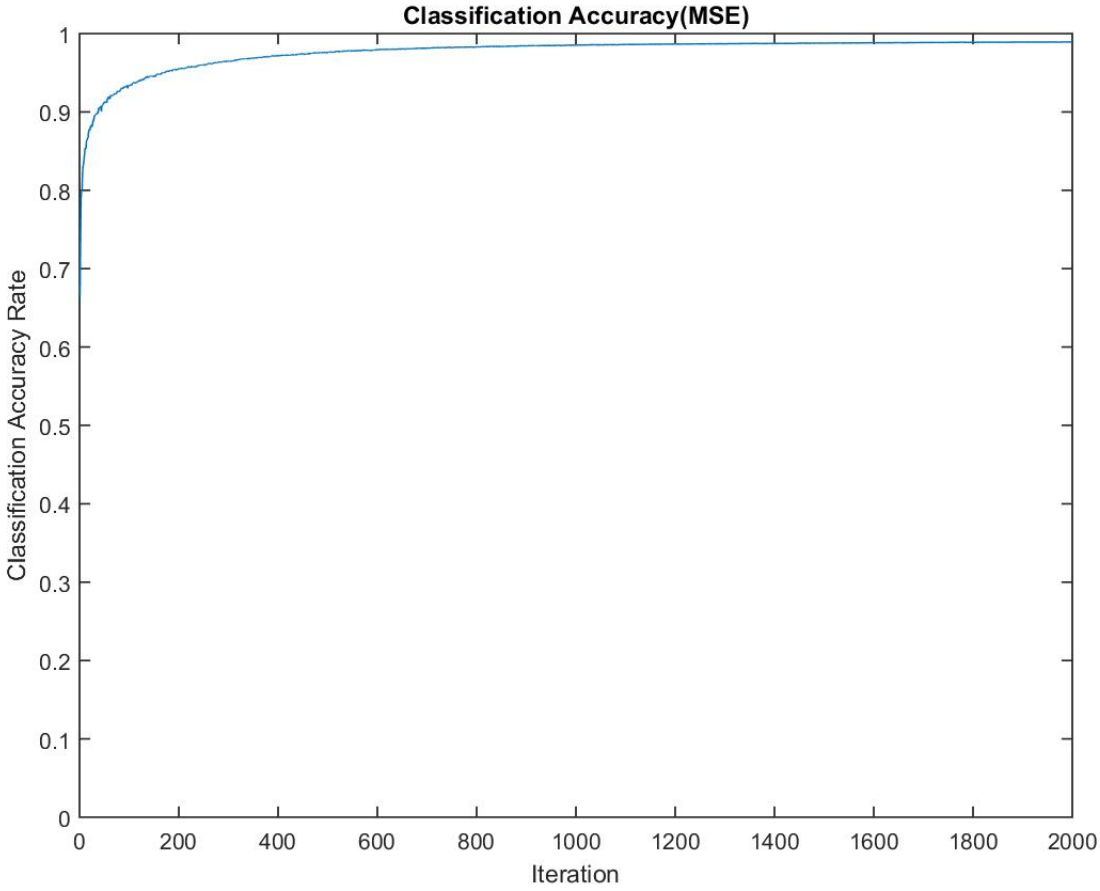
2. Training of Neural Network

When I trained the neural network, I have two stopping criteria. The first one is that the iteration times should not exceed 2000. The second stopping criteria is that when the training classification rate exceeds 0.999, the training will be terminated. The raw data of the digit data set are preprocessed in my program. First, the data is filtered so that each image has been processed as a black and white image. Second, each feature is standardized by first subtracting the mean and then divided by the standard deviation.My initial weights are distributed uniformly on $[-0.07, 0.07]$ and $[-0.17, 0.17]$ for the weights from input layer to hidden layer and for the weights from hidden layer to out layer. The Kaggle score is 0.9542, and the submission of Kaggle is from the model with

Mean-squared Error:

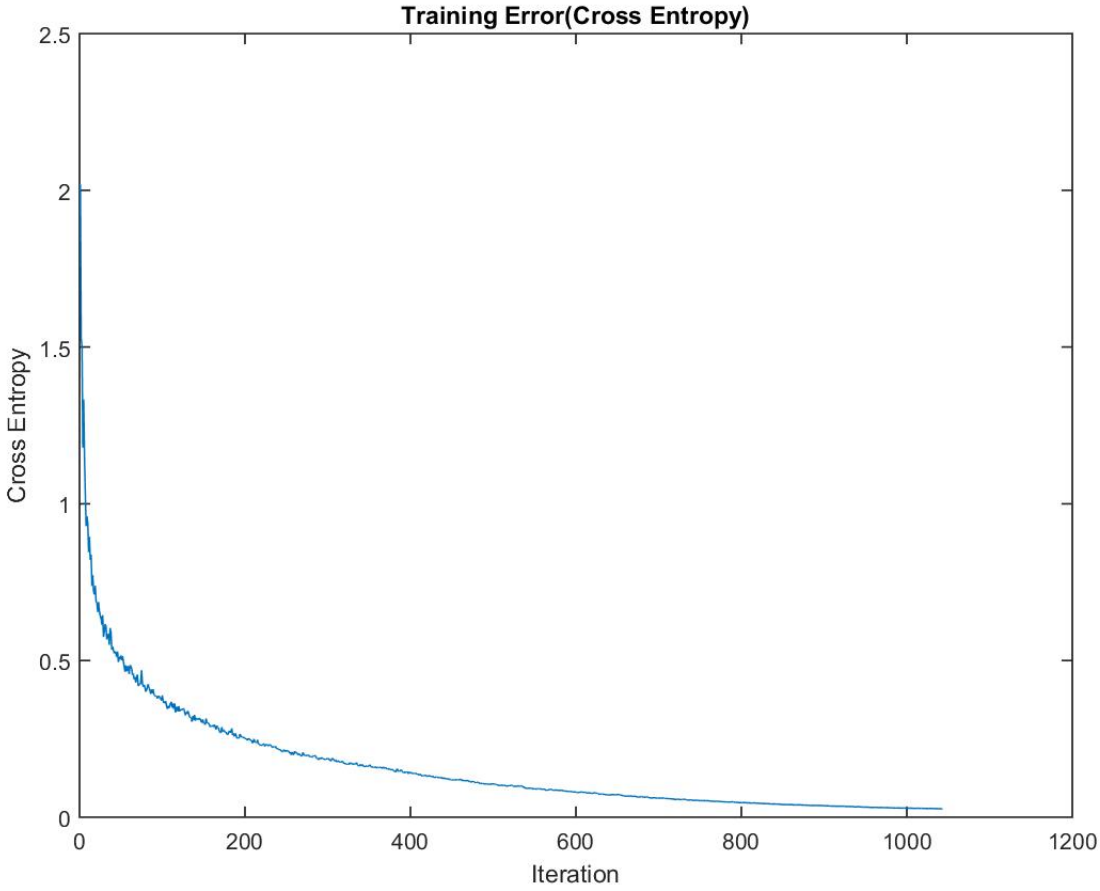
With mean-squared error as the objective to minimize, the learning rate is 0.005. To ensure the convergence of stochastic gradient algorithm, in each step, the step size is scaled by $\frac{1}{ln(0.005n+1)+1}$, where n is the iteration number. In each iteration, 500 data points are randomly selected to update the weight matrices. When the algorithm terminates, the training error is 0.0069. The classification accuracy of training data is 0.9892. 6000 points are randomly chosen from the original data set as validation data. The error rate of the validation data is only 0.055. The time for training is 1953.068620 seconds.The training error and classification accuracy on training set vs. iteration are plot as below:

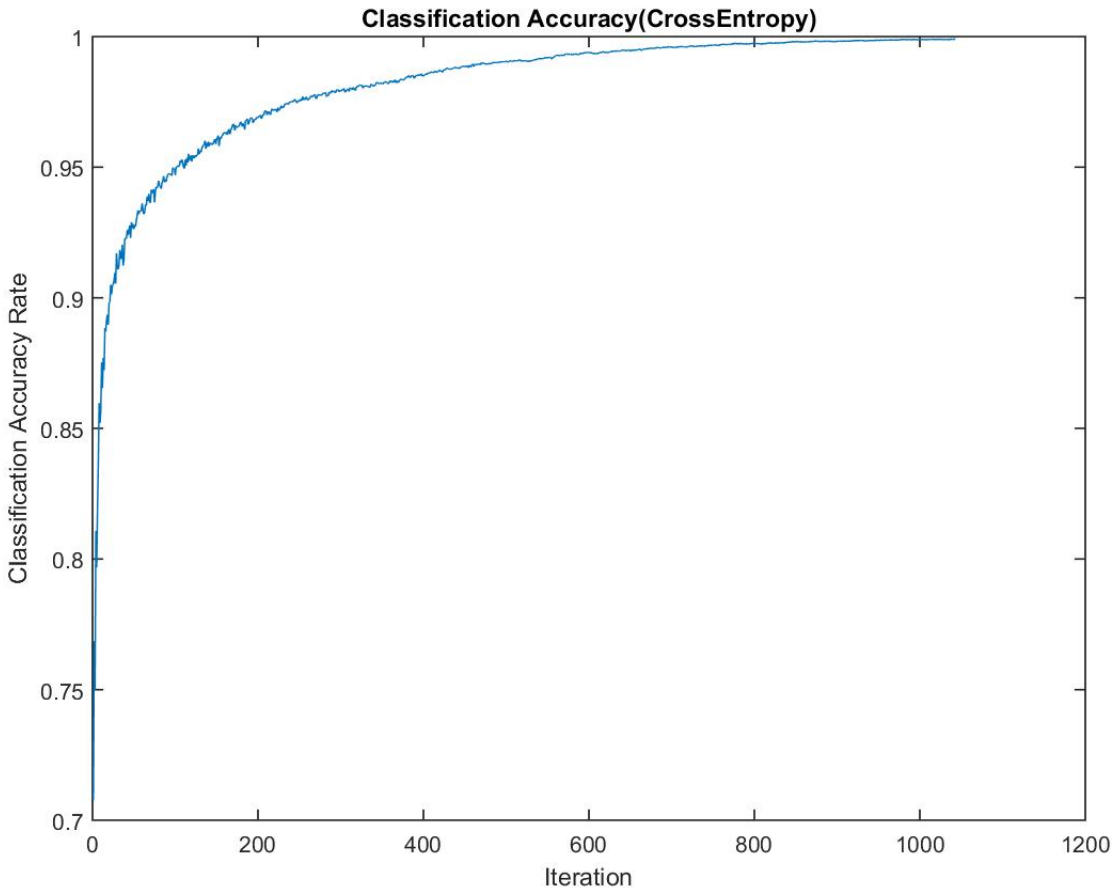




Cross-Entropy Error:

With mean-squared error as the objective to minimize, the learning rate is 0.04. To ensure the convergence of stochastic gradient algorithm, in each step, the step size is scaled by $\frac{1}{ln(0.005n+1)+1}$, where n is the iteration number. In each iteration, 500 data points are randomly selected to update the weight matrices. When the algorithm terminates, the training error is 0.0069. The classification accuracy of training data is 0.90. 6000 points are randomly chosen from the original data set as validation data. The error rate of the validation data is only 0.055. The time for training is 1953.068620 seconds.The training error and classification accuracy on training set vs. iteration are plot as below:


Training Error(Cross Entropy)
Cross Entropy
Iteration
2.5
2
1.5
1
0.5
0
0
200
400
600
800
1000
1200



From the results we can see that cross-entropy error provides faster convergence and higher accuracy rate. I think the reason for that is when the actual label and predicted label are different, the cost is higher and the gradient of cross-entropy can be higher.