Classification

Recognition of MNIST digit images

Introduction

This project is to implement and evaluate classification algorithms. The classification task is to recognize a 28 x 28 grayscale handwritten digit image and identify it as a digit among 0, 1, 2, ..., 9. We will do the following three tasks.

- Implement logistic regression, train it on the MNIST digit images and tune hyper parameters.
- Implement single hidden layer neural network, train it on the MNIST digit images and tune hyper parameters such as the number of units in the hidden layer.
- Use a publicly available convolutional neural network package, train it on the MNIST digit images and tune hyper parameters.

Data Set

For the training of our classifiers, we will use the MNIST dataset. The MNIST database is a large database of handwritten digits that is commonly used for training and testing in the field of machine learning. The database contains 60,000 training images and 10,000 testing images. Each image has 28x28 grayscale pixels, each pixel is represented by a number between 0 and 255 indicating the intensity of the pixel. The input images are reshaped to a matrix with each row corresponding to an image and each row has 28 x 28 values corresponding to the pixels. The values are then scaled to lie between 0 and 1.

Logistic Regression

We use 1-of-K coding scheme t = [t1, ..., tk] for our multiclass classification task. Our multiclass logistic regression model is represented in the form,

$$p\left(\mathcal{C}_k|\mathbf{x}\right) = y_k\left(\mathbf{x}\right) = \frac{\exp\left(a_k\right)}{\sum_j \exp\left(a_j\right)}$$

The cross-entropy error function for multiclass classification problem seeing a training sample x is,

$$E\left(\mathbf{x}\right) = -\sum_{k=1}^{K} t_k \ln y_k$$

The gradient of the error function is

$$\nabla_{\mathbf{w}_j} E\left(\mathbf{x}\right) = \left(y_j - t_j\right)\mathbf{x}$$

We use stochastic gradient descent which uses first order derivatives to update W to find the optimum of the error,

$$\mathbf{w}_j^{t+1} = \mathbf{w}_j^{t} - \eta\nabla_{\mathbf{w}_j} E\left(\mathbf{x}\right)$$

The following are the tuned parameters,

Learning rate = 0.00001
Num. of iteration = 5000
Misclassification rate on the 60000 train set is 6.5717%
Misclassification rate on the 10000 test set is 7.38%

Single hidden layer Neural Network

We have used a neural network with one hidden layer. The feed forward propagation is,

$$z_j = h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + b_j^{(1)}\right)$$

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + b_k^{(2)}$$

$$y_k = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

The backpropagation is done as follows,

$$\delta_k = y_k - t_k$$

$$\delta_j = h'(z_j) \sum_{k=1}^{K} w_{kj} \delta_k$$

The gradient of the error function would be

$$\frac{\partial E}{\partial w_{ji}^{(1)}} = \delta_j x_i, \qquad \frac{\partial E}{\partial w_{kj}^{(2)}} = \delta_k z_j$$

Having the gradients, we will use stochastic gradient descent to train the neural network.

The following are the tuned parameters,

Learning rate = 0.00001
Num. of iteration = 5000
Activation function for hidden layers = Sigmoid function
Misclassification rate on the 60000 train set is 1.0333%
Misclassification rate on the 10000 test set is 2.46%

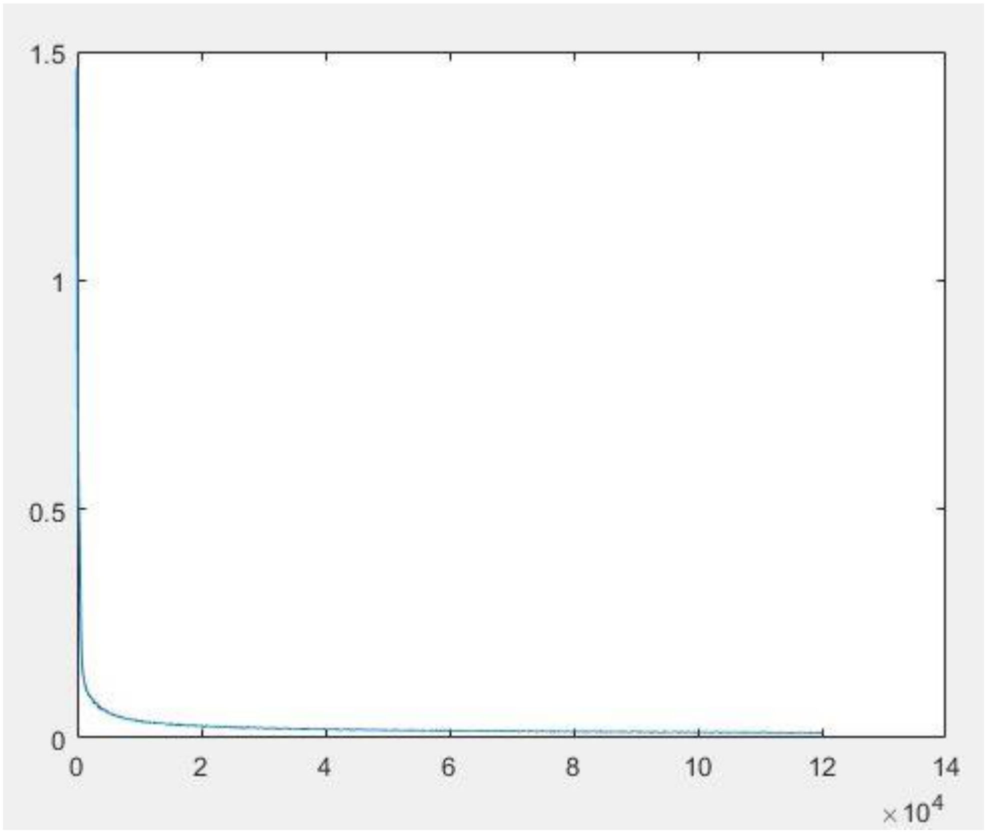
Convolutional Neural Network

For the convolutional neural network, we have used the Deep Learn Toolbox found here, https://github.com/rasmusbergpalm/DeepLearnToolbox

We have used a convolutional neural network with five layers. The first layer is the input layer. The second layer is a convolutional layer with 5 output maps and kernel size 5. The third layer is a sub sampling layer with a scale of 2. The fourth layer is convolutional layer with 12 output maps and kernel size 5. The fifth layer is a sub sampling layer with a scale of 2.

With this convolutional neural network with 100 epochs of the data, we get a classification error rate of 1.08% on the test images.

The following diagram shows the classification error rate,


1.5
1
0.5
0
0
2
4
6
8
10
12
14
×10^4