Multilayer Neural Network for Image Classification

I. Prepare your dataset

In this experiment, you need to implement a multilayer neural network for image classification using the MNIST dataset. To use that data, first download the images from this link http://yann.lecun.com/exdb/mnist/. The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. There are four files that contains both the images and labels for training and test. Download these four file:

- train-images-idx3-ubyte.gz: training set images (9912422 bytes)
- train-labels-idx1-ubyte.gz: training set labels (28881 bytes)
- t10k-images-idx3-ubyte.gz: test set images (1648877 bytes)
- t10k-labels-idx1-ubyte.gz: test set labels (4542 bytes)

after downloading the files, uncompress them and use whatever languages that you like to read the images and labels. You should get the images like what is shown in the following figure. Please confirm that the image are correctly read out from the files by visualizing them as images.

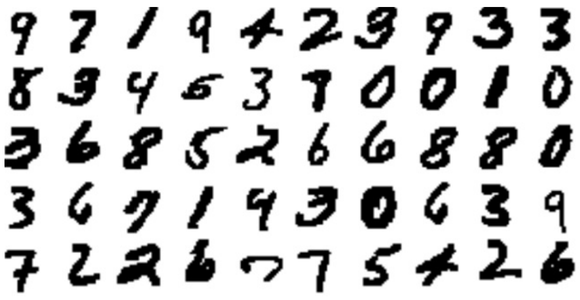

Fig. 1. Example images from MNSIT data.

II. Compile the code

Design a following 3 layer neural network. The input data is vectors with a dimensionality of 784. The first hidden layer has 300 neuron and the second hidden layer has 100 neurons. The output layer has 10 classes, which corresponding to the 10 classes of digits that you obtained from the MNSIT data.

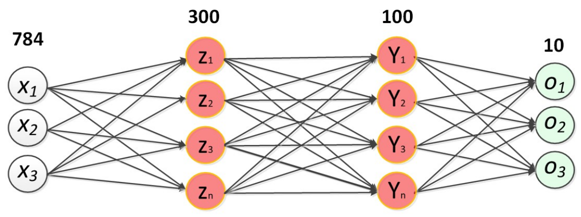

Fig. 2. Network Architecture.

The math equations that you need are as follows. Suppose the input data is $\boldsymbol{X} = \{\boldsymbol{x}_n\}_{n=1}^N$ of 784 dimensional vectors. Since there are three layers, therefore there are three weight matrices, i.e., $\boldsymbol{W}_1 : [300 \times 784]$, $\boldsymbol{W}_2 : [100 \times 300]$ $\boldsymbol{W}_3 : [10 \times 100]$ and thee sets of bias $\boldsymbol{b}_1 \in R^{300}$, $\boldsymbol{b}_2 \in R^{100}$, $\boldsymbol{b}_3 \in R^{10}$ that have to be learned. You can choose whatever activation functions that you like. Common activations functions and their derivatives are as follows:

- Sigmoid activation function:

$$g(x) = \frac{1}{1+\exp(x)} \quad g'(x) = g(x)(1-g(x))$$

- Hyperbolic Tangent function:

$$g(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)} \quad g'(x) = 1 - g^2(x)$$

- Rectified Linear Unit (ReLu) function

$$g(x) = \max(0,x) \quad g'(x) = \begin{cases} 1 & x > 0 \\ 0 & \text{otherwise} \end{cases}$$

The forward procedure are as follows:

$$\begin{aligned}\mathbf{z} &= g(\boldsymbol{W}_1\boldsymbol{x}^T + \boldsymbol{b}_1)\\ \mathbf{y} &= g(\boldsymbol{W}_2\boldsymbol{z}^T + \boldsymbol{b}_2)\\ \mathbf{o} &= \boldsymbol{W}_3\boldsymbol{y}^T + \boldsymbol{b}_3\end{aligned}$$

after that, we apply softmax function to obtain the probabilities as follows.

$$a_i = \frac{\exp o_i}{\sum_{j=1}^{n} \exp o_j}.$$

Suppose the true labels are $\boldsymbol{t} = (t_1, ..., t_{10})$, NOTE this is a one-hot encoded vector. That means there is only one nonzero element and all others are zeros. The cross entropy loss is defined as

$$\mathcal{L} = -\sum_{i=1}^{B}\sum_{j=1}^{10} t_i \ln a_i$$

B is the batch size. The entire data should be divided into batches and forward each batch of data into the network. The backpropagation equation for the loss function is

$$\frac{\partial \mathcal{L}}{\partial o_i} = a_i - t_i$$

After that, use what you have learned in the course to compute the $\nabla\boldsymbol{W}_1$, $\nabla\boldsymbol{W}_2$, and $\nabla\boldsymbol{W}_3$ based on the the back-propagation procedure. Then use the stochastic gradient descent algorithm to update the three weight matrices.

After learning, pass the test data into the network and test the accuracy. Compare your accuracy with other benchmark accuracies on the official website http://yann.lecun.com/exdb/mnist/.