ECSE 4965/6965

Introduction to Deep Learning

Program Assignment 3

Due date: March 20, 11:59pm

In this programming assignment, you will implement techniques to learn a deep (two hidden layers) neural network (NN) for image digit classification on the MNIST dataset (https://en.wikipedia.org/wiki/MNIST_database). The NN will take an image of a hand-written numerical digit as input and classify it into one of 10 classes corresponding to digits 0 to 9 respectively. Given the training data, you will follow the equations in the lecture notes to train the NN using the back propagation method and then evaluate its performance on the given testing data. The NN has one input layer (784 nodes), two hidden layers with 100 nodes for each layer, and one output layer with 10 nodes, corresponding to the 10 digit classes respectively. For training, you will learn the parameters $\mathbf{\Theta}$=[$\mathbf{W}_1$, $\mathbf{W}_{1,0}$, $\mathbf{W}_2$, $\mathbf{W}_{2,0}$, $\mathbf{W}_3$,$\mathbf{W}_{3,0}$] by minimizing the squared loss for the output nodes, where $\mathbf{W}_1$ and $\mathbf{W}_2$ are the weight matrices for the hidden layers, $\mathbf{W}_3$ is the weight matrix for the output layer, and $\mathbf{W}_{1,0}$,$\mathbf{W}_{2,0}$, $\mathbf{W}_{3,0}$ are the corresponding bias vectors.

Specifically, given the training data **D**={**x**[m], ***y***[m]}, m=1,2, …, M, where **x**[m] is a grayscale image of 28x28 (a 784x1 vector) and ***y***[m] is output vector that follows the 1 of K encoding, with kth element of being 1 and the rest being 0. Use ReLU activation function for the hidden nodes and the softmax (multi-class sigmoid) function for the output. You will implement the following methods to learn the parameters $\mathbf{\Theta}$

1. Implement in Tensorflow the back propagation method to solve for $\mathbf{\Theta}$ iteratively using all data in **D**. Each time, randomly choose a mini-batch of 50 data from **D**. For each point in the mini-batch, perform forward propagation and back propagation to compute the gradients of weight matrix and weight bias vector for each layer. Then update the weights and bias using the average gradient of the gradients computed from all the 50 samples. Initialize weight matrix to small different values and bias vectors all to zero. Iteratively update each **W** with an appropriate learning rate until convergence. Save $\mathbf{\Theta}$ using pickle.dump functions (see below for details).

2. Using the given testing dataset **T**, evaluate the performance of your trained NN by computing the classification error for each digit as well as the average classification errors for all digits. The classification error for each digit is computed as the ratio of incorrect classification to the total number images for that digit. Plot the average training classification error, average testing classification error, and value of the loss function after each parameters update.

3. Do not use Tensorflow's back propagation function. You however may use it to verify your implementation. Submit your Tensorflow code via LMS, along with the required classification error plots, and the saved parameters $\mathbf{\Theta}$.

Data format

The training data **D** contains 50000 training images (00001.jpg – 50000.jpg) in train_data folder, and their labels are stored in train_label.txt with each raw being the label of the corresponding image.

The testing data **T** contains 10000 testing images (00001.jpg – 10000.jpg) in test_data folder, and their labels are stored in test_label.txt.

Load images and normalize

Load the images in the order

Convert it to vector and normalize it to [0,1] by dividing the intensity of each pixel by 255

The following link teaches how to load images:

http://learningtensorflow.com/lesson3/

Output

Use the code below to save the learned parameters Theta in the following format

$\mathbf{\Theta} = [\mathbf{W}_1, \mathbf{W}_{1,0}, \mathbf{W}_2, \mathbf{W}_{2,0}, \mathbf{W}_3, \mathbf{W}_{3,0}]$

$\mathbf{\Theta}$ is a list objects. $\mathbf{W}_1$ and $\mathbf{W}_2$ are the matrix of the weights for the hidden layers, $\mathbf{W}_3$ is the matrix of the weights for the output layer, and $\mathbf{W}_{1,0}, \mathbf{W}_{2,0}, \mathbf{W}_{3,0}$ are the corresponding bias.

```
--------------------
import pickle
filehandler = open("nn_parameters.txt","wb")

pickle.dump(Theta, filehandler, protocol=2)

filehandler.close()

-----------------
```

Batch computing

To speed up the training, you can simultaneously compute the gradients of 50 samples at the same time using 3-D tensor. For example, given two tensors A^{N*p*q} and B ^{N*q*m}, tf.matmul(A,B) will compute matrix multiplication N time and return a tensor with shape N*p*m. More details and example can be found at:

https://www.tensorflow.org/api_docs/python/tf/matmul