2802ICT Programming Assignment 2

Due: 11:59pm, Sunday 17 May 2020

Total Mark: 15

Task:

In this assignment, you are to implement a popular machine learning algorithms, i.e. **Neural Networks**, using **Python**. You will train and test a simple neural network with the datasets provided to you and experiment with different settings of hyper parameters. The datasets provided to you is a subset of the popular MNIST hand written digits database (URL) in which a sample of it is given below:

Figure 1: Sample images in MNIST

The training and test sets are provided to you together with this assignment in the following files:

Training samples: TrainDigitX.csv.gz
Training labels: TrainDigitY.csv.gz
Test samples: TestDigitX.csv.gz
Test labels: TestDigitY.csv.gz
Test samples with no labels provided: TestDigitX2.csv.gz

There are 50,000 training samples in TrainDigitX.csv.gz, 10,000 test samples in TestDigitX.csv.gz, and another 5,000 test sample in TestDigitX2.csv.gz. Each sample is a handwritten digit represented by a 28 by 28 greyscale pixel image. Each pixel is a value between 0 and 1 with a value of 0 indicating white. Each sample used in the dataset (a row in TrainDigitX.csv, TestDigitX.csv, or TestDigitX2.csv.gz) is therefore a feature vector of length 784(28x28=784). TrainDIgitY.csv.gz and TestDigitY.csv.gz provide labels for samples in TrainDigitX.csv.gz and TestDigitX.csv.gz, respectively. The value of a label is the digit it represents, e.g., a label of value 8 indicates the sample represents the digit 8.

Besides the python implementation, you should also submit a **report** discussing about your implementation and experiments, which should address the following three parts:

Part 1: Manual calculation for a small neural network

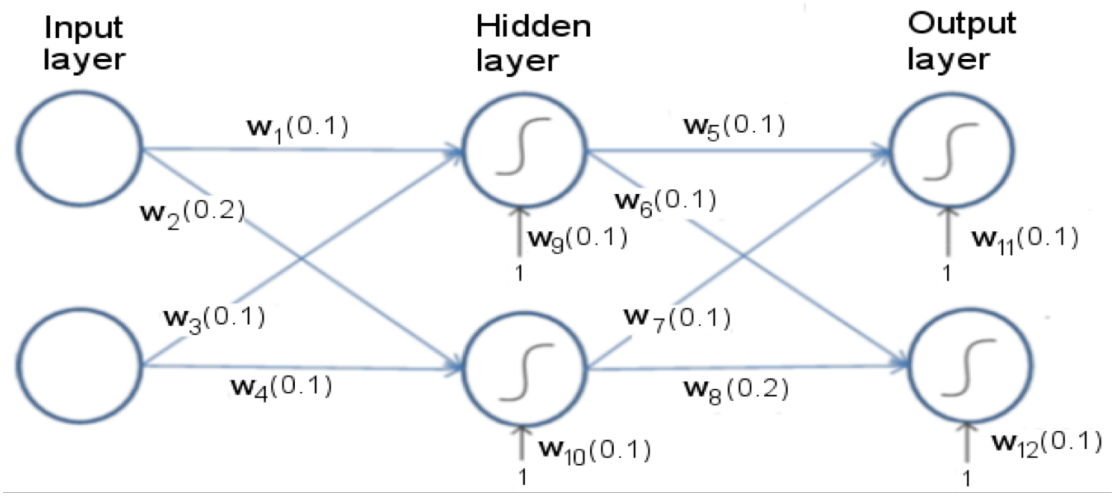

Figure 2: A Small Neural Network.

Before writing your code, you should work on the neural network shown in Fig. 2 by hand. The small network has two neurons in the input layer, two neurons in the hidden layer and two neurons in the output layer. Weights and biases are marked on the figure. W_9, w_{10}, w_{11}, w_{12} are weights for the bias term. Note that there are no biases for the neurons in the input layer. There are two training samples: X_1 = (0.1, 0.1) and X_2 = (0.1, 0.2). The label for X_1 is 0, so the desired output for X_1 in the output layer should be Y_1 = (1, 0). The label for X_2 is 1, so the desired output for X_2 in the output layer should be Y_2 = (0, 1). You should update the weights and biases for this small neural net using stochastic gradient descent with backpropagation. You should use batch size of 2 and a learning rate of 0.1. You should update all the weights and biases only once MANUALLY. Show your working and results in your report. And then use this as a test case to test your code which you will implement in Part 2.

Part 2: Implementation in Python

Your task is to implement a neural network learning algorithm that creates a neural network classifier based on the given training datasets. Your network has 3 layers: an input layer, one hidden layer and an output layer. You should name your main file as neural_network.py which accepts seven arguments. Your code should be run on the command line in the following manner:

> python neural_network.py NInput NHidden NOutput TrainDigitX.csv.gz TrainDigitY.csv.gz TestDigitX.csv.gz PredictDigitY.csv.gz

where the meanings of each argument are:

NInput: number of neurons in the input layer

NHidden: number of neurons in the hidden layer

NOutput: number of neurons in the output layer
trainset.csv.gz: the training set
trainset_label.csv.gz: labels associated with the training set
testset.csv.gz: the test set
testset_predict.csv.gz: predicted labels for the test set

You should set the default value of number of epochs to 30, size of mini-batches to 20, and learning rate to 3, respectively.

Note: you can use numpy.loadtxt() and numpy.savetxt() methods to read from or write to a csv.gz. There is no need to un-compress the file before use. To making plots, you can use the Python library 'matplotlib'.

The nonlinearity used in your neural net should be the basic sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

The main steps of training a neural net using stochastic gradient descent are:

- Assign random initial weights and biases to the neurons. Each initial weight or bias is a random floating-point number drawn from the standard normal distribution (mean 0 and variance 1).
- For each training example in a mini-batch, use backpropagation to calculate a gradient estimate, which consists of following steps:
  1. Feed forward the input to get the activations of the output layer.
  2. Calculate derivatives of the cost function for that input with respect to the activations of the output layer.
  3. Calculate the errors for all the weights and biases of the neurons using backpropagation.
- Update weights (and biases) using stochastic gradient descent:

$$w \to w - \frac{\eta}{m} \sum_{i=1}^{m} error_i^w$$

where m is the number of training examples in a mini-batch, $error_i^w$ is the error of weight w for input i, and η is the learning rate.

- Repeat this for all mini-batches. Repeat the whole process for specified number of epochs. At the end of each epoch evaluate the network on the test data and display its accuracy.

For this part of the assignment, use the quadratic cost function:

$$C(w,b) = \frac{1}{2n}\sum_{i=1}^{n} ||f(x_i) - y_i||^2$$

where

w: weights

b: biases

n: number of test instances

x_i: i^{th} test instance vector

y_i: i^{th} test label vector, i.e. if the label for x_i is 8, then y_i will be [0,0,0,0,0,0,0,0,1,0]

f(x): Label predicted by the neural network for an input x

For this hand written digits recognition assignment, we will encode the output (a number between 0 and 9) by using 10 output neurons. The neuron with the highest activation will be taken as the prediction of the network. So the output number y has to be represented as a list of 10 numbers, all of them being 0 except for the entry at the correct digit.

You should do the following:

1. Create a neural net of size [784, 30, 10]. This network has three layers: 784 neurons in the input layer, 30 neurons in the hidden layer, and 10 neurons in the output layer. Then train it on the training data with the following settings: epoch = 30, minibatch size = 20, η = 3.0

   Test your code on the test data (TestDigitX.csv.gz) and make plots of test accuracy vs epoch. Report the maximum accuracy achieved. Also run you code with the second test set (TestDigitX2.csv.gz) and output your predictions to PredictDigitY2.csv.gz. You should upload both PredictTestY.csv.gz and PredictTestY2.gz in your submission.
2. Train new neural nets with the same settings as in (1) above but with η = 0.001, 0.1, 1.0, 10, 100. Plot test accuracy vs epoch for each η on the same graph. Report the maximum test accuracy achieved for each η. Remember to create a new neural net each time so its starts learning from scratch.
3. Train new neural nets with the same settings as in (1) above but with mini-batch sizes = 1, 5, 10, 20, 100. Plot maximum test accuracy vs mini-batch size. Which one achieves the maximum test accuracy? Which one is the slowest?
4. Try different hyper-parameter settings (number of epochs, η, and mini-batch size, etc.). Report the maximum test accuracy you achieved and the settings of all the hyper parameters.

Note: Once you implemented your neural network learning algorithm, you should compare its computed values with the manually calculated values you did by hand in Part 1 for the small network shown in Fig. 2 to verify its correctness. To verify your implementation, make sure that the weights and biases output by the learned network after one epoch are the same as those you calculated manually in Part 1. Train this small network for 3 epochs and output the weights and biases after each epoch. Report this in your report. Note that you need to do this before you run your network on the MNIST dataset.

Part 3: An alternate cost function

Now replace the quadratic cost function by a cross entropy cost function

$$C(w,b) = -\frac{1}{n}\sum_{i=1}^{n} y_i \ln\left[f(x_i)\right] + (1-y_i)\ln[1-f(x_i)]$$

Train a neural net with the specifications as in Part 2. What is the test maximum accuracy achieved?

What to hand in:

You should hand in your well commented python scripts and the two prediction files: PredictTestY.csv.gz and PredictTestY2.gz, as well as a report explaining your algorithms and the experimental results with respect to Part 1 to Part 3. Submission should be done through the link provided in L@G by the due date.

Marking scheme:

Marks are allocated as follows:

(a) Part 1 (3 marks)

(b) Part 2 (6 marks)

(c) Part 3 (3 marks)

(d) Report quality (3 marks)

About collaboration:

You are encouraged to discuss the assignment with your fellow students, but eventually this should be your own work. Anyone caught plagiarizing will be penalized.