CS 331 SPRING 2019

CS331- Introduction to Artificial Intelligence

Instructor: Dr. Yasir Mehmood

Assignment – 2

Deadline: 2nd May, 2019.

Honor Code

You are to do this assignment by yourself. All of the submitted code should be your own creation and a result of your own thought process. No cooperation is allowed under any circumstances. Any help outside of the course staff is prohibited. You are also given the responsibility of reporting any cooperation incidences to the course staff.

Please follow the below instructions, failure to do so will result in loss of marks.

Important Instructions

- This assignment is worth **5%** of your grade so start early and do **NOT** hardcode anything in your assignment.
- You're **NOT** allowed to use any pre-existing machine learning libraries. You need to implement the backpropagation algorithm yourself.
- This assignment is to be done in Python same environment as for the previous assignment and you are required to use **NUMPY** for this assignment where necessary.
- You are not allowed to use more than **2 epochs** for training the dataset.
- If your network is taking more than 8 minutes to train, you are doing something wrong. Your network must train under **8 minutes**.
- You have to take command line arguments for test and train data for example:

  For Training:

  ```
  python MyNetwork.py train train.txt train-labels.txt learningRate
  ```

  For Testing:

  ```
  python MyNetwork.py test test.txt test-labels.txt netWeights.txt
  ```

  Make sure this format is followed. Argument types are clear from their names.

- Visit the following link to download the datasets: https://drive.google.com/drive/folders/145Gc30kX0HhIAh1MbXU5uAQIGZqI9dFv?usp=sharing

Note:

Please make sure you **do not overfit** your network (by more than 2 epochs). We will be testing your network on a separate test data set and will also be training your network for separate testing as well.

Simple Neural Network

You are provided with a dataset of handwritten digits. The **txt files folder** contains four files; **train.txt, test.txt, train-labels.txt and test-labels.txt**. The dataset consists of 70,000 labeled 28x28 pixel grayscale images of hand-written digits. The dataset is split into 60,000 training images and 10,000 test images. There are 10 classes (one for each of the 10 digits). The task at hand is to train a model using the 60,000 training images and subsequently test its classification accuracy on the 10,000 test images.
The first file, **train.txt,** has 60,000 training samples and the second, **test.txt,** has 10,000 samples for you to test your code on whereas both the **train-labels.txt and test-labels.txt,** contains the labels for each of the images corresponding to each row of the **train.txt and test.txt respectively**. *Each sample is a handwritten digit represented by a 28 by 28 grayscale pixel image and is a feature vector of length 784 (*the input layer of the neural net contains 784 = 28 × 28 neurons). Each pixel is a value between 0 and 255, where 0 indicates white. The value of a label is the digit it represents. For instance, a label of value 8 means the sample represents the digit 8.

The format of each image in **train.txt and test.txt** is like the following (the square brackets included):

[0 255 0 0 255 198 187 0 0 … 0 0 255]

The network you're going to work with has **three** layers. *One input layer, one hidden layer and one output layer*. The activation function should be:

RADIAL BASIS FUNCTION

You will have to read up on this function your self and attempt to implement it as this has not been covered in class. (This is a substitute for the Sigmoid Function)

Below is a link that may prove useful for understanding RBF:
https://www.youtube.com/watch?v=OUtTI99uRf4

The output layer of the network contains 10 neurons. If the first neuron fires, i.e., has an output ≈1, then that will indicate that the network thinks the digit is a 0. If the second neuron fires then that will indicate that the network thinks the digit is a 1. And so on. A little more precisely, we number the output neurons from 0 through 9, and figure out which neuron has the highest activation value. If that neuron is, say, neuron number 6, then our network will guess that the input digit was a 6. And so on for the other output neurons.

For example: network output of [0.2, 0.4, 0.01, 0.22, 0.5, 0.8, 0.35, 0.11, 0.32, 0.1]
Means your network predicted: 5 (Highest value)

1. Choose any three different learning rates (The answer to "Why?" is given below).

2. Assign random weights over the links between the layers. These weights should be picked at random from an interval of [-1, +1]. You may also want to add biases to the neurons in each layer (but this is entirely up to you, all we want is how well your NN performs!).

3. Create a neural net of size [784, 30, 10]. Since the network has three layers, it means 784 neurons in the input layer, 30 in the hidden layer and 10 in the output layer. For each training example use backpropagation algorithm to calculate the gradient estimate. It consists of the following steps:

   - ❖ Feed forward the input to get activations of the output layer.
   - ❖ Calculate derivatives of the cost function for that input with respect to the activations of the output layer.
   - ❖ Calculate the errors for all the weights (and biases) of the neurons using Back propagation.
   - ❖ Update weights (and biases).

Fig. 1 Sample of images from data set

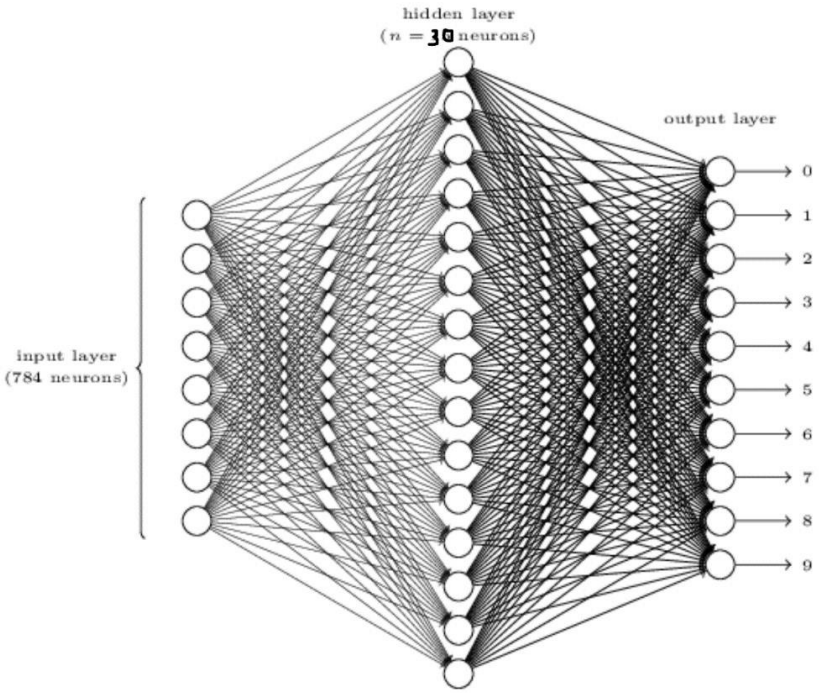

Fig. 2 Our three layer neural network

TASKS

1. Devise your program using **Cross Entropy** as your cost function and report model accuracy. Remember we did not study this cost function in the class. You will have to do some research on this part yourself.

2. After training your network (after 2 epochs) save your network weights in a file **netWeight.txt**. In the testing phase this file must be given as an argument (as mentioned in the instructions) and must be assigned to the network, because you want to test your network from the weights you have trained it on. You have to submit your weights as well.

3. Plot the execution *time versus learning rate graph* for at least three different learning rates.

4. Write a report of the methods employed to reach your results along with a description of your networks determined weight values, accuracies etc. and explanation of why you chose which technique to arrive at your current (approximate) accuracy. You will be marked on the quality of the content of your report so write anything you feel deserves credit. (Should not be more than half a page)

You should correctly and clearly show your output on the console. Any unclear output will not be awarded marks. Clearly state epoch number, accuracy, and error for each epoch with clear labelling of what each print statement means. You can additionally print number of correct classifications out of the total.

Sample Output:

Epoch Number 1 -------> 1005/10000 images correctly classified

Accuracy 10.05 % ---------- Error 89.95 %

Epoch Number 2 -------> 9000/10000 images correctly classified

Accuracy 90.00 % ---------- Error 10.00 %

^these are of course hypothetical values and can vary but just an illustration of your output on console.

Marking Criteria:

Accuracy	30
Time	30
Graphs	20
Report	15
Output	5

- Failing to follow the instructions will lead to loss of marks.

Files to Submit:

- **Network.py**
- **netWeights.txt**
- **Report**
- **Graphs**

Zip all the files in a folder named: YourRollNumber_Assignment2.zip

Both Network.py and netWeights.txt will go through moss before testing, and no plagiarism act will be tolerated.

Here are a few links to help you start:

https://iamtrask.github.io/2015/07/12/basic-python-network/

http://iamtrask.github.io/2015/07/27/python-network-part2/

https://www.youtube.com/watch?v=aircAruvnKk&t=9s

Good Luck! ☺