11-785 Fall 2017
Homework 01 (v2)

Introduction

For this homework assignment you will be working with the MNIST data, MNIST is an acronym for Modified NIST and NIST is an acronym for National Institute of Standards and technology, MNIST is a database of handwritten digits. The NIST special database 3 and special database 1 were originally created by NIST by gathering handwriting samples from census bureau workers and high school students respectively. MNIST was later created by Yann LeCun and others by combining the two databases, increasing the size of the images from 20x20 to 28x28 and centering them around the "center of mass of the pixels" by taking the average location of the pixels weighted by intensity. The data consists of 60000 training examples and 10000 testing examples (To test accuracy after training on the training images) and labels for each of the images, each of the images is 28x28 and rescale, and the labels are unsigned bytes ranging from 0 to 9. Further information can be found at http://yann.lecun.com/exdb/mnist/.

This assignment should be done without the use of a machine learning framework in order to gain and understanding of the internals of neural networks.

Task 1:

In this task you will implement a basic neural network for MNIST. There are several requirements for this task.

1. You must use sigmoid activation.
   You may want to use a numerically stable implementation of sigmoid, such as scipy.special.expit, which uses something like:

```
def sigmoid(x):
    if x < 0:
        a = exp(x)
        return a / (1 + a)
    else:
        return 1 / (1 + exp(-x))
```

2. You must be able to set the learning rate and the number of training iterations.
3. The network must have a (784)-128-64-10 architecture.
4. You must classify based on the highest activation in the last layer.
5. Each iteration must make a full pass over the data.
6. Initialize the weights within a layer by uniformly spacing over the range [-0.01, 0.01]. For example if you had 3 weights in layer, you would initialize them to -0.01, 0, and 0.01. If you had 5 weights in the next layer they would be initialized to -0.01, -0.006, -0.002, 0.002, 0.006, and 0.01. Note that since the layers all have an even number of neurons, we

will not actually have a weight of 0 in our architecture (why would this be a bad idea?) Biases must initially be set to 0.

7. You must use cross entropy cost.

Things to keep in mind:

1. You will need to save the total input to each neuron in order to back propagate your gradients.
2. You will need to accumulate the back propagated error in order to adjust weights and biases at the end of each iteration.

For grading please plot (matplotlib is a good resource for this) and submit your costs and accuracies for 50 iterations with learning rate 0.001. It should go without saying that discussing these accuracies or costs is prohibited.

Task 2:

Congratulations, you have created your first neural net, unfortunately, as you may have noticed passing through all 60000 training examples each iteration is rather slow so in this task you will implement minibatches. Requirements for this task:

1. The minibatches must proceed sequentially through the data
2. You must be able to adjust the minibatch size.

For grading please plot and submit your costs and accuracies for 50 iterations with learning rate 0.001 for minibatches of size 10, 100 and 500.

Task 3:

Implement RMSprop. Keep in mind the learning rate should update at the end of each iteration.

Task 4:

Part 1. Implement dropout. Requires customizable dropout percentage.
Part 2. Implement batch normalization. Requires customizable scale and offset.

If please submit a file named test.sh which trains your network for 10 epochs on the training data then tests it network on the testing data and outputs only the accuracy of the network (in percentage points) as a decimal.
For grading submit all of your source files.