CSL603– Machine Learning
Lab 3

Due on 3/11/2017 11.55pm

Instructions: Upload to your moodle account one zip file containing the following. Please do not submit hardcopy of your solutions. In case moodle is not accessible email the zip file to the instructor at ckn@iitrpr.ac.in. Late submission is not allowed without prior approval of the instructor. You are expected to follow the honor code of the course while doing this homework.

1. **You are allowed to work in pairs for this lab. However, this does not mean only one team member works on the lab, while the other only gets to add his/her name. Both the team members should know the inner workings of the code. I and the TAs reserve the right to question you if required. If you are found unable to explain the code, you will receive no points for the lab.**
2. This lab must be implemented in Matlab/Python.
3. A neatly formatted PDF document with your answers for each of the questions in the homework. You can use latex, MS word or any other software to create the PDF.
4. Include a separate folder named as 'code' containing the scripts for the homework along with the necessary data files. Ensure the code is documented properly.
5. Include a README file explaining how to execute the scripts.
6. Name the ZIP file using the following convention rollnumber(s)hwnumber.zip

---

In this lab, you will be experimenting with multi-layer perceptron. The second part of the lab takes time to train. So, it is advisable to get started early. Please contact the IT department to obtain a licensed version of Matlab that can be installed on your machine for implementing this lab. You are welcome to implement this lab in Python as well.

1. This is a warm up exercise for problem 2. You will experiment with a simple 2 dimensional 3 class classification problem to visualize decision boundaries learned by a MLP. Our goal is to study the changes to the decision boundary and the training error with respect to parameters such as number of training iterations, number of hidden layer neurons and finally the learning rate. The Matlab script l31.m in the l3.zip file provides the necessary skeletal framework for performing these experiments. Before we start the experiments,

complete the following functions in the script. The locations in the script where your code needs to be added have been marked as 'TO DO'.

i. [w v trainerror] = mlptrain(X, Y, H, eta, nEpochs) that outputs the weights of the connections between input and hidden layer units - w and between hidden and output layers units - v, given the training data X, training labels Y, the number of hidden layer units H, the learning rate eta, and the number of training epochs nEpochs. We will assume cross entropy as the error function, sigmoid as the activation function for the hidden layer units and softmax function for the output layer units. The output of the next work will be the probability of classification.

ii. Compute the training error using the learned weights at the end of each epoch. The error value is stored in the variable trainerror(epoch) that is returned by the mlptrain function.

iii. ydash = mlptest(X, w, v) that determines the outputs of the output layer units of the MLP, given the data X, the learned weights w and v. This function essentially performs a forward pass on the learned MLP using the data.

iv. If you have correctly implemented the above functionalities, we are all set to run some experiments. Let us fix the number of epochs to be 1000, the learning rate to be 0.01 and study the relation between the training error and the number of hidden layer units. Vary the number of hidden layer units in powers of 2, starting from 2 and going up till 64. What are your observations? Include in the report a single graph that includes the plots for training error against number of epochs for the different choices of hidden layer units. Also include separate plots for the decision boundaries for three choices of hidden layer units - 2, 4 and 64. What are your observations from these plots?

2. In this question, you will implement a basic neural network to predict the steering angle the road image for a self-driving car application that is inspired by Udacity's Behavior Cloning Project [1]. This dataset has been compiled from Udacity's simulator. However, we have preprocessed the original images to transform them to greyscale images of size 32x32. This is roughly the same size of an MNIST image. You should be able to run these experiments on your machines. The accompanying dataset folder (steering) contains the training images. The data.txt files contains the image name and the corresponding steering angle. The following are the specifications for the first task of this lab.

   a. The network architecture should be 1024 (input)-128-64-1 (there is a possibility that this architecture may change. So your implementation should have the ability to tune these parameters)

   b. Use a numerically stable implementation of sigmoid, such as scipy.special.expit that uses something like the following

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

```
            z = exp(x)
            return z / (1 + z)
```

c. All the hidden layers will employ sigmoid activation function, while the output layer will not have any activation function.
d. The implementation should have the ability to tune/set the learning rate, minibatch size, and the number of training iterations.
e. Sum of squares error should be used as the loss function at the output layer.
f. The minibatches must proceed sequentially through the data. The data should be shuffled initially before the start of the training.
g. Dropout will be implemented for all layers. The implementation should have the ability to set the dropout percentage.
h. The weights of each layer should be initialized by uniformly spacing over the range [-0.01, 0.01]. The bias terms should be initialized to 0.
i. The dataset provided to you should be split into training/validation according to 80:20 ratio.

For grading purposes, we want you to generate the following plots. Note that we should be able to recreate these plots using your implementation.

i. A plot of sum of squares error on the training and validation set as a function of training iterations (for 50 iterations) with a learning rate of 0.001. (no dropout, minibatch size of 64).
ii. A plot of sum of squares error on the training and validation set as a function of training iterations (for 50 iterations) with a fixed learning rate of 0.001 for three minibatch sizes – 32, 64, 128.
iii. A plot of sum of squares error on the training and validation set as a function of training iterations (for 50 iterations) with a learning rate of 0.001, and dropout probability of 0.5 for the first, second and third layers.
iv. A plot of sum of squares error on the training and validation set as a function of training iterations (for 50 iterations) with different learning rates – 0.001, 0.0005, 0.0001 (no drop out, minibatch size – 64)

Discuss your inferences/observations from these plots on the training/tuning neural networks.

The second task of the lab is a competition. You are allowed to perform the following modifications to the networks to try better models with the constraint that these have to be implemented from scratch. (some of them are easy to implement, while some might take a significant amount of time)

- Implement convolution layers
- Implement a different activation function
- Implement Max Pooling layers
- Implement and try a different optimizer (such as Adam or RMS Prop)
- Experiment with varying number of layers and nodes in a layer
- Anything else that you might like to try…

Design and run your experiments to keep track of the best model obtained so far. Two days prior to the submission deadline, I will post a set of test images, and ask you to submit the predicted steering angle for each image. You will provide a text file as part of your submission. Each line in the text file should correspond to the steering angle predicted for the corresponding image. You must also give include a description of your approach for this part of the lab. The final model resulting in the predictions should also be shared, along with the code to generate the predictions using this model for the test dataset. This part of the lab is a competition, and the scores will be normalized relative to the winning team. A maximum of 10 points will be awarded for this task.

An important aspect of machine learning is reproducibility of the results presented in a paper/report. Therefore, we will run your code to see if the results are closely matching with what you have presented in the report. Any deviation beyond a reasonable threshold will be considered as fudging of results and will invite severe penalty.

Reference

[1] https://github.com/udacity/CarND-Behavioral-Cloning-P3