Columbia University
IEOR4742 – Deep Learning for OR & FE (Hirsa)
Assignment 2 – Due 11:40 am on Thursday March 5th, 2020

Problem 1 (Impact of different number of layers, different activation functions and optimization on learning): The code *logistic_regression_multi_layer.ipynb* is a 2-layer logistic regression. The goal is to extend it to 4- & 6-later logistic regression with different neurons and activation function for each layer utilizing different optimization routine to assess their impact on accuracy. Consider the following six feedforward architectures:

(1) 1st layer: sigmoid, 2nd layer: tanh, 3rd layer: reLU, 4th layer: sigmoid, 5th layer: leaky reLU, 6th layer: tanh

(2) 1st layer: tanh, 2nd layer: sigmoid, 3rd layer: sigmoid, 4th layer: reLU, 5th layer: tanh, 6th layer: leaky reLU

(3) 4 layers all leaky reLU

(4) 4 layers all sigmoid

(5) 6 layers all leaky reLU

(6) 6 layers all sigmoid

- Groups 1-8 optimizer choice: `tf.train.GradientDescentOptimizer`
- Groups 1-2 – 50 neurons for each layer in all those 6 architectures
- Groups 3-4 – 100 neurons for each layer in all those 6 architectures
- Groups 5-6 – 200 neurons for each layer in all those 6 architectures
- Groups 7-8 – 300 neurons for each layer in all those 6 architectures
- Groups 9-16 optimizer choice: `tf.train.RMSPropOptimizer`
- Groups 9-10 – 50 neurons for each layer in all those 6 architectures
- Groups 11-12 – 100 neurons for each layer in all those 6 architectures
- Groups 13-14 – 200 neurons for each layer in all those 6 architectures
- Groups 15-16 – 300 neurons for each layer in all those 6 architectures
- Groups 17-22 optimizer choice: `tf.train.AdamOptimizer`
- Groups 17-18 – 50 neurons for each layer in all those 6 architectures
- Groups 19-20 – 100 neurons for each layer in all those 6 architectures
- Groups 21 – 200 neurons for each layer in all those 6 architectures
- Groups 22 – 300 neurons for each layer in all those 6 architectures

In all those cases, what is the exact number of parameters we are trying to learn? Assess and conclude

Problem 2 (`CIFAR-10`): Repeat **Problem 1** for `CIFAR-10` dataset.

Problem 3 (visualization of the lost function): Use the first & second architecture and the optimization routine you were assigned in **Problem 1** to assess the loss function by interpolation, namely the path traveled through the loss function from the starting point to the optimal point obtained by the optimizer (extension of *example_interpolation_1.ipynb*)