Introduction to Deep Learning, Fall 2018

Homework on Unit 1: 100 Total Points

Due: September 16th, 11:59PM

Submissions are required electronically through Sakai. Non-code is required in pdf format, and code is required in a Jupyter notebook.

Because the computational costs of this homework are rather low, you should be able to do everything on your local machine. However, later homeworks will require greater compute power, so you should try to get code running on the Duke Compute Cluster *now*.

Problem 1: Understanding Backpropagation (25 points)

Backpropagation is a critical concept to understanding how deep networks are actually learned. It is important to get a grounding in how it actually works by working through examples of backprop in practice. Towards that end, please do the following:

(a) Set up logistic regression as a network (i.e. as in lecture 2, slide 12), and derive the gradient on the parameters by using backprop. Note that there is only one key step here, which is accounting for the sigmoid function correctly.

(b) Set up a multi-layer perceptron with a single hidden layer (i.e. as in lecture 2, slide 32), and derive the gradients on both layers using backprop.

(c) Show how the results in (a) and (b) will change when using a softmax (multi-class) setup rather than a binary result.

Problem 2: Algorithmic Implementation of a Multi-Class Logistic Regression *without Tensorflow* (30 Points)

To gain a greater understanding of how neural networks are really working in code, it is necessary to implement simple networks by hand to build this understanding. Therefore, please do the following using python functions and numpy rather than Tensorflow.

(a) Set up a logistic regression network, and learn it on MNIST using stochastic gradient descent.

Notes:

(1) Don't use other neural network toolkits (e.g. pytorch). If you want to use a package beyond standard numpy, which is all that's necessary, ask a TA.

(2) It is perfectly acceptable to use Tensorflow's methods to pull the MNIST data and create mini-batches, but do not use it for the networks or optimization.

Problem 3: Algorithmic Implementation of a Multi-Layer Perceptron *with Tensorflow* (25 Points)

As in problem 2, but now you are free to use Tensorflow.

(a) Set up a logistic regression network, and learn it on MNIST using stochastic gradient descent.

(b) Set up an MLP with a single hidden layer (you can choose the number of hidden nodes) and learn it on MNIST using stochastic gradient descent.

(c) Set up an MLP with two hidden layers (i.e. lecture 2, slide 55)

Problem 4: Performance Comparison (20 points)

We want to compare the differing networks to determine what's best for recognizing MNIST digits. To do this, split the data into training data and validation data. Use the training data to learn the network and then estimate performance on the validation data Then please answer the following:

(a) Did your implementations and Tensorflow's implementations from problems 2 and 3 perform the same?
(b) What is the validation accuracy from the multi-class logistic regression?
(c) What is the validation accuracy from the multi-class MLP with a single hidden layer? If you change the number of nodes in the hidden layer, how susceptible is the hold out performance?
(d) What is the validation accuracy from the multi-class MLP with two hidden layer? If you change the number of nodes in the hidden layers, how susceptible is the hold out performance?
(e) Do you match my reported accuracies (lecture 2, slide 58)?

Problem 5: Bookkeeping (5 points)

(a) Did you successfully run code on the cluster?
(b) How many hours did this assignment take you? (There is **NO** correct answer here, this is just an information gathering exercise)
(c) Verify that you adhered to the Duke Community Standard in this assignment (https://studentaffairs.duke.edu/conduct/about-us/duke-community-standard). (I.E. write "I adhered to the Duke Community Standard in the completion of this assignment")