Introduction to Deep Learning, Fall 2018

Homework on Unit 2: 50 Total Points (Half-Sized Homework)

Due: September 28th, 11:59PM

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

The overarching goal of this homework is to build a mathematical understanding and practical intuition of stochastic gradient methods, which will be used extensively over the duration of the course. How effective your optimization method is can drastically impact the effectiveness of the network and the computational costs.

Problem 1: Understanding Gradient Descent via Majorization-Minimization (20 Points)
A common viewpoint of (stochastic) gradient descent is to view it as a "majorization-minimization" approach. Succinctly, this means that you find a simple, easy-to-evaluate function that *upper bounds* the true objective function, which you then minimize. First, we will derive why a smooth gradient upper bounds the first order approximation of the function we showed in the lectures, and then we'll examine when the system is expected to improve the objective

(a) Using the fundamental theorem of calculus,

$$F(\boldsymbol{y}) = F(\boldsymbol{x}) + \int_0^1 (\boldsymbol{y}-\boldsymbol{x})^{\mathrm{T}} \nabla F\big(\boldsymbol{x} + t(\boldsymbol{y}-\boldsymbol{x})\big) dt,$$

and the fact that the gradient is smooth with Lipschitz constant L, prove the form of the Lipschitz gradient upper bound on the function from the lectures

$$F(\boldsymbol{y}) \leq F(\boldsymbol{x}) + [\nabla F(\boldsymbol{x})]^{\mathrm{T}}(\boldsymbol{y}-\boldsymbol{x}) + \frac{L}{2}\|\boldsymbol{y}-\boldsymbol{x}\|_2^2$$

(b) Using the form of the upper bound for stochastic gradient descent given in the literature for $E[F(\boldsymbol{w}_{k+1})] - F(\boldsymbol{w}_k)$, under what regimes will:

1. A step size of $\alpha_k = \frac{1}{L}$ be optimal?
2. A step size of $\alpha_k = \frac{1}{2L}$ be optimal?
3. A step size of $\alpha_k = \frac{1}{10L}$ be optimal?

Problem 2: Practical Implementations of Optimization Functions (27 Points)
In addition to the mathematical intuition in problem 1, it is important to see what the practical implications of such functions are. There are numerous ways to implement such functions, but there are built-in Tensorflow functions that can do this for you.

Using the built in functions, i.e. (tf.train.GradientDescentOptimizer, tf.train.MomentumOptimizer, tf.train.AdamOptimizer), try out each function on your most complex MLP from HW 1. You can use any dataset you like, but MNIST is fine.

Specifically, do *at least* the following:

(a) Try multiple (>3) step-sizes for SGD. Determine the "optimal" constant step size.
(b) Vary the step-size and momentum settings in SGD w/ momentum. Determine the "optimal" settings.
(c) Use the default settings of the Adam algorithm
(d) Vary the step-size of the Adam algorithm to see the change in how effective it works

After that, for each setting, please answer the following:

(a) How did each algorithm qualitatively work?
(b) How hard was it to tune your step sizes and settings?
(c) How did each algorithm perform in validation performance estimates?

Problem 3: Bookkeeping (3 points)

(a) How many hours did this assignment take you? (There is **NO** correct answer here, this is just an information gathering exercise)
(b) 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")