TUM EI 70360: Machine Learning and Optimization
Fall 2023
Lecturer: Reinhard Heckel
Teaching Assistant: Tobit Klug

Problem Set 6
Issued: Tuesday, Nov. 21, 2023
Due: Thursday, Nov. 30, 2023

Problem 1 (Convergence of gradient descent and stochastic gradient method)**.** Which of the following statements on gradient descent (GD) and the stochastic gradient method (SGM) applied to minimizing a convex and differentiable function f are true, without making any assumption on f other than that f has a minimum?

(a) If the stepsize of GD is chosen constant and sufficiently small, GD converges to a minimizer of f.

(b) If the stepsize of SGM is chosen constant and sufficiently small, SGM converges to a minimizer of f.

(c) There is a fixed choice of stepsizes α_k such that the average of the iterates of SGM, weighted by the stepsizes, converges to a minimizer of f.

Problem 2 (Stochastic subgradient methods, 6pts)**.** In this problem you will implement the subgradient and stochastic subgradient methods for minimizing the convex but nondifferentiable function

$$J(\mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, \langle \mathbf{w}, \mathbf{x}_i \rangle + b) + \frac{\lambda}{2} ||\mathbf{w}||^2$$

where $L(y,t) = \max\{0, 1 - yt\}$ is the hinge loss. This corresponds to the optimal soft margin hyperplane.

1. Determine $J_i(\mathbf{w}, b)$ such that
$$J(\mathbf{w}, b) = \sum_{i=1}^{n} J_i(\mathbf{w}, b).$$

2. Determine a subgradient $\mathbf{u}_i$ of each J_i with respect to the variable $\boldsymbol{\theta} = [b \ \mathbf{w}^T]^T$. A subgradient of J is then $\sum_{i=1}^{n} \mathbf{u}_i$.

   Note that for a convex function $f \colon \mathbb{R}^d \to \mathbb{R}$ that is convex, but not necessarily differentiable, a sub-gradient at θ is a vector $\mathbf{g} \in \mathbb{R}^d$ such that
$$f(\theta') \geq f(\theta) + \left\langle \theta' - \theta, \mathbf{g} \right\rangle \qquad \text{for all } \mathbf{x}' \in \mathbb{R}^d.$$

If f is differentiable at θ then the sub-gradient at θ is unique and equal to the gradient at θ. The sub-gradient is a natural generalization of the gradient of a convex function. For example, the function $f(\theta) = \max\{\langle \theta, \mathbf{x} \rangle, 0\}$ is not differentiable at $\theta = 0$. However, a sub-gradient at $\theta = 0$ is given by 0. At all points where $\langle \theta, \mathbf{x} \rangle$ is positive, f is differentiable and thus the sub-gradient is equal to $\mathbf{x}$; at all points where $\langle \theta, \mathbf{x} \rangle$ is negative, f is also differentiable and the sub-gradient is equal to 0.

3. Download the nuclear.csv file from the course website. The variables x and y contain training data for a binary classification problem. The variables correspond to the total energy and tail energy of waveforms produced by a nuclear particle detector. The classes correspond to neutrons and gamma rays, which allows the two particle types to be distinguished. This is a somewhat large data set (n = 20,000), and subgradient methods are appropriate given their scalability.

   Implement the subgradient method for minimizing J and apply it to the nuclear data. Hand in two figures: One showing the data as a scatter plot with the learned linear decision boundary, the other showing J as a function of iteration number. Also report the estimated hyperplane parameters and the minimum achieved value of the objective function.

   - Some advice: Debugging goes faster if you look at a subsample of the data. Fixing the random number generator seed is also helpful so that errors become reproducible. To do this in numpy, execute the command `numpy.random.seed(0)`.
   - Use $\lambda = 0.001$. Since this is a linear problem in low dimension, we don't need much regularization.
   - Use a step size of $\alpha_j = 100/j$, where j is the iteration number.
   - To compute the subgradient of J, write a function to find the subgradient of J_i and then sum those results.
   - Since the objective will not be monotone decreasing, determining a good stopping rule can be tricky. Just look at the graph of the objective function and "eyeball it" to decide when the algorithm has converged.

4. Now implement the stochastic subgradient method, which is like the subgradient method, except that your step direction is a subgradient of a random J_i, not J. Be sure to cycle through all data points before starting a new loop through the data. Report/hand in the same items as in part (c).

   - Use the same λ, stopping strategy, and α_j as in part 3. Here j indexes the number of times you have cycled (randomly) through all the data.
   - Your plot of J versus iteration number will have roughly n times as many points as in part (c) since you have n updates for every one update of the full subgradient method.

5. Comment on the (empirical) rate of convergence of the stochastic subgradient method relative to the subgradient method. Explain your findings.