Nonlinear Optimization I, Fall 2021
Homework 3
Due before lecture on 10/14
Please only in file formats .pdf, .zip, standard image formats (.jpg, .png, etc.).

Your submitted solutions to homeworks should be entirely your own work. Do not copy solutions from other students or any online source. You are allowed to discuss homework problems at a high-level with other students, but should carry out the execution of any thoughts/directions discussed independently, on your own. Feel free to cite any result presented in class without proof.

You can write solutions by hand or type them up (the LaTeX code for this pdf is on blackboard).

Q1. Compute the proximal operator prox_f for the following pair of functions.

(a) $f(x) = \begin{cases} 0 & \text{if } \|x\|_\infty \leq 1 \\ +\infty & \text{otherwise.} \end{cases}$

(b) $f(x) = \alpha\|x\|_3^3 = \sum_{i=1}^d \alpha|x_i|^3$

Q2. Consider a $\mu > 0$-strongly convex, continuously differentiable function[1] $f : \mathbb{R}^d \to \mathbb{R}$.

(a) For any fixed $\bar{x} \in \mathbb{R}^d$, show that every x with $f(x) \leq f(\bar{x})$ lies in the compact region given by

$$\|x - \bar{x}\|_2 \leq \frac{2\|\nabla f(\bar{x})\|_2}{\mu} \quad \text{and} \quad f(x) \geq f(\bar{x}) - \frac{1}{2\mu}\|\nabla f(\bar{x})\|_2^2.$$

(b) Conclude that a minimizer of f must exist.

(c) Conclude further that this minimizer is unique. (HW2-Q3(a) may be helpful here.)

Q3. Suppose you have a set of observations $\{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathbb{R}^d$ is some feature vector and $y_i \in \{\pm 1\}$ is a label. One formulation of the problem of training a Support Vector Machine to distinguish between observations with y_i positive and y_i negative is given by computing a vector $w \in \mathbb{R}^d$ minimizing

$$\min_{w \in \mathbb{R}^d} f(w) = \sum_{i=1}^n \max\{0, 1 - y_i \cdot x_i^T w\} + \frac{\lambda}{2}\|w\|_2^2.$$

(a) For any $w \in \mathbb{R}^d$, show that a subgradient of f can be computed as

$$\sum_{i=1}^n g_i + \lambda w \in \partial f(w)$$

where $g_i = \begin{cases} 0 & \text{if } y_i \cdot x_i^T w > 1 \\ -y_i x_i & \text{otherwise} \end{cases}$ for each i.

(b) In contrast to this simple subgradient computation, argue that computing $\text{prox}_{\alpha f}(0)$ is as hard as solving another Support Vector Machine problem instance.

[1]The existence and uniqueness results actually hold more broadly without needing to assume differentiability.

Q4. Consider the following LASSO optimization problem used to compute sparse approximate solutions to a linear system $Ax = b$:

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2} \|Ax - b\|_2^2 + \gamma \|x\|_1$$

for some $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $\gamma = 2$. For this exercise, first write a program that generates A and b with i.i.d. normally distributed entries (mean zero, variance one) and $n = 1000$, $m = 100$. Note that $n > m$ and so the system $Ax = b$ has infinitely many solutions (with probability one).

(a) For any x, verify that a subgradient of f at x is given by

$$g(x) := A^T(Ax - b) + \gamma \; sign(x)$$

where each component of $sign(x)$ is given by $sign(x)_i = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{otherwise.} \end{cases}$

(b) Implement and run 100 steps of the following Subgradient Method

$$x_{k+1} = x_k - \alpha_k g(x_k)$$

using $x_0 = 0$ and $\alpha_k = 1/\lambda_{max}(A^T A)$. Print out the objective value at each iteration. Verify your last iterate x_{100} is not sparse vector (having no entries that are exactly zero).

(c) Implement and run 100 steps of the Proximal Gradient Descent using $x_0 = 0$ and the same stepsize. Print out the objective value at each iteration. Verify your last iterate is a fairly sparse vector (having more zero entries than nonzero).

(d) Implement and run 100 steps of the Accelerated Proximal Gradient Method using $x_0 = 0$ and the same stepsize. Print out the objective value at each iteration. Verify your last iterate is a fairly sparse vector (having more zero entries than nonzero).

General Guidelines for Programming HW Problems: You can do programming assignments in any programming language you feel comfortable with (python, matlab, java, c/c++, haskell, etc). Programming questions will ask for you to solve a particular problem or describe particular settings to run an algorithm under. You must submit both your code and the requested output/plots from running your code. Grading will focus primarily on the quality of these outputs rather than of your code.