Nonlinear Optimization 1, Fall 2023 - Homework 3
Due at 3:30PM on Thursday 9/29 (Gradescope)

Your submitted solutions to assignments should be your own work. While discussing homework problems with peers is permitted, the final work and implementation of any discussed ideas must be executed solely by you. Acknowledge any source you consult.

Problem 1 - Computing proximal operators

Compute the proximal operator prox_f for the following functions.

(a) (ℓ_1**-norm**) $f(x) = \|x\|_1 = \sum_{i=1}^n |x_i|$.

(b) (**Indicator** ℓ_∞**-ball**) $f(x) = \begin{cases} 0 & \text{if } \|x\|_\infty \leq 1, \text{or} \\ +\infty & \text{otherwise.} \end{cases}$

(c) (ℓ_3**-norm cubed**) $f(x) = \alpha\|x\|_3^3 = \alpha \sum_{i=1}^n |x_i|^3$.

Problem 2 - The Moreau envelope is slick

Given any closed convex continous function $f\colon \mathbb{R}^d \to \mathbb{R} \cup \{\infty\}$, we define its Moreau envelope as the function given by

$$\widehat{f}(x) = \min_{y \in \mathbb{R}^d} f(y) + \frac{1}{2}\|y - x\|^2.$$

Then, $\mathrm{prox}_f(x)$ is the unique y attaining the above minimum.

(a) Prove that f and $\widehat{f}$ have the same minimum value. Moreover prove that they also have the same minimizers.

(b) Prove that $\widehat{f}$ is convex and everywhere finite.

(c) Prove that $\mathrm{prox}_f(x)$ depends continuously on x.

(d) Prove that $\partial\widehat{f}(x) = \{x - \mathrm{prox}_f(x)\}$, so $\widehat{f}$ is continously differentiable.

Problem 3 - The forward-backward method is not always the best choice

Consider a dataset of observations represented as $\{(x_i, y_i)\}_{i=1}^n$, where each $x_i \in \mathbb{R}^d$ denotes a feature vector and the corresponding $y_i \in \{\pm 1\}$ serves as its label. One prevalent approache to derive a classifier from this dataset is through Support Vector Machines (SVM). The training process for an SVM entails solving the following optimization problem:

$$\min_{w \in \mathbb{R}^n} f(w) \quad \text{where} \quad f(x) := \sum_{i=1}^n \max\{0, 1 - y_i x_i^\top w\} + \frac{\lambda}{2}\|w\|^2.$$

(a) For any $w \in \mathbb{R}^d$, show that there exists subgradient $\xi \in \partial f(w)$ that has the form

$$\xi = \sum_{i=1}^n g_i + \lambda w \quad \text{where} \quad g_i = \begin{cases} 0 & \text{if } y \cdot x_i^\top w > 1, \\ -y_i x_i & \text{otherwise} \end{cases} \quad \text{for all } i \in [n].$$

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

Problem 4 - A regularization rodeo

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

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

for a given $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $\lambda = 2$.

(a) Write a program that generates random A and b with i.i.d. normally distibuted entries $N(0, 1)$ with $n = 1000$ and $m = 100$. Note that this means that the system $Ax = b$ has infinitely many solutions.

(b) For any x, verify that

$$g(x) = A^\top(Ax - b) + \lambda\,\mathrm{sign}(x) \qquad \text{where} \qquad \mathrm{sign}(x)_i = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{if } x = 0, \\ -1 & \text{otherwise} \end{cases} \quad \text{for all } i \in [n].$$

satisfies $g(x) \in \partial f(x)$.

(c) Implement and run 100 steps of Subgradient Descent, i.e.,

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

on you random problem using $x_0 = 0$ and stepsize $\alpha_k = 1/\lambda_{\max}(A^\top A)$. Verify your last iterate is not a sparse vectors (having no zero entries).

(d) Implement and run 100 steps of the Forward-Backward Method using same stepsize and initial point. Verify that the last iterate is fairly sparse (having more zero than nonzero entries). How does the loss function of the last iterate compare with the one you got in (c)?

(e) Implement and run 100 steps of the Accelerated Forward-Backward Method using same stepsize and initial point. Verify that the last iterate is fairly sparse (having more zero than nonzero entries). How does the loss function of the last iterate compare with the ones you got in (c) and (d)? Consider making a plot.