MATH 126 Winter 18: Homework 2

posted on Jan. 19; due by Jan. 26

1. [10 points] Consider the following problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} \boldsymbol{c}^\top \boldsymbol{x}$$
$$\text{subject to } \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b} \text{ for all } \boldsymbol{A} \in \mathcal{A},$$

where $\mathcal{A} \subseteq \mathbb{R}^{p\times d}$ is the set

$$\mathcal{A} = \{\boldsymbol{A} \in \mathbb{R}^{p\times d} \ : \ \bar{A}_{ij} - V_{ij} \le A_{ij} \le \bar{A}_{ij} + V_{ij}, \ i = 1,\dots,p, \ j = 1,\dots,d\}$$

for given matrices $\bar{\boldsymbol{A}}$ and $\boldsymbol{V}$ (with $V_{ij} \ge 0$ for all i, j). Express this problem as an LP.

2. [3 points each] Consider a gradient descent method with a constant step size s:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - s\nabla f(\boldsymbol{x}_k), \quad k = 0, 1, \dots$$

for solving a quadratic problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} \left\{ f(\boldsymbol{x}) \equiv \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{p}^\top \boldsymbol{x} \right\}$$

where $\boldsymbol{Q} \in \mathbb{S}^d_{++}$, and we denote μ and L to be the smallest and largest eigenvalues of $\boldsymbol{Q}$ for $0 < \mu < L$.
(1) What is the closed-form optimal solution $\boldsymbol{x}_*$ of the problem?
(2) Show that

$$||\boldsymbol{x}_{k+1} - \boldsymbol{x}_*||_2 \le ||\boldsymbol{I} - s\boldsymbol{Q}||_2 ||\boldsymbol{x}_k - \boldsymbol{x}_*||_2. \tag{a}$$

(3) Considering (a), what is the condition of step size s that guarantees a sequence $\{\boldsymbol{x}_k\}$ to converge to $\boldsymbol{x}_*$ (*i.e.*, $||\boldsymbol{x}_k - \boldsymbol{x}_*||_2 \to 0$ as $k \to \infty$)?
(4) Considering (a), what is the optimal step size $\hat{s}$ (*i.e.*, the step size that has the fastest rate of convergence)?
(5) Specify a (two-dimensional) worst-case function f of a gradient descent method with the optimal step size $\hat{s}$ (*i.e.*, find f that satisfies $||\boldsymbol{x}_{k+1} - \boldsymbol{x}_*||_2 = ||\boldsymbol{I} - \hat{s}\boldsymbol{Q}||_2 ||\boldsymbol{x}_k - \boldsymbol{x}_*||_2$).

3. [5 points each] Consider the quadratic problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} \left\{ \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{p}^\top \boldsymbol{x} \right\} \tag{P}$$

where $\boldsymbol{Q}$ is the 50×50 Hilbert matrix defined by

$$Q_{i,j} = \frac{1}{i + j - 1}, \quad i, j = 1, \dots, 50,$$

and $\boldsymbol{p} = (1, 1/2, 1/3, \ldots, 1/50)^\top$. The matrix can be constructed via the MATLAB command `Q = hilb(50)`. Implement the (MATLAB) code for the following five methods efficiently (*i.e.*, avoid redundant computations for fair run time comparison), and compare their performance with the initial vector $\boldsymbol{x}_0 = (1, 2, 3, \ldots, 50)^\top$ and a stopping criteria $||\nabla f(\boldsymbol{x})|| \leq 10^{-3}$. (Hint: The optimal value is $p_* = -0.5$, so check whether your algorithm generates a sequence $f(\boldsymbol{x}_k)$ that reaches to p_*.)

- MATLAB linear system solver[1] (`linsolve` or backslash command),
- Gradient descent methods with
  - Exact line search
  - Backtracking line search (initial $s = 1$, $\alpha = 0.4$, $\beta = 0.5$)
  - Constant step
- Conjugate gradient (CG) method
  (Its MATLAB code is available at https://en.wikipedia.org/wiki/Conjugate_gradient_method.)

(1) Present the step size rule for the exact line search of the gradient method for solving (P). Describe your appropriate[2] choice of the constant step size for the gradient method for solving (P).

(2) Report the run time of MATLAB linear system solver. Report the total number of iterations and run time to reach given tolerance for the four iterative optimization algorithms. Also report the per-iteration run time[3] for the four cases. (Hint: Use the MATLAB `tic/toc` commands to check the run time.)

(3) Compare the four iterative optimization algorithms (except the MATLAB solver) in terms of the total number of iterations and run time to reach given tolerance, and the per-iteration run time, with a brief reasoning for such results (*e.g.*, state a reason why one algorithm has shorter per-iteration run time compared to other algorithms, state why one algorithm took more iterations to reach same tolerance).

[1] Note that this will perform the best among the algorithms considered in Problem 3 for this toy (small-dimensional) problem, but not necessarily for the large-dimensional problems.

[2] Your *appropriate* step size should guarantee a descent update at every iteration.

[3] For simplicity, let's denote the per-iteration run time to be $\frac{\text{(Run time)}}{\text{(Total number of iterations)}}$, ignoring the fact that some algorithms compute some operations before iterating (*i.e.*, before the first iteration).