ECE 6270, Spring 2021

Homework #4

Due Thursday, March 4, at 11:59pm Suggested Reading: B&V, Sections 9.2 (again) and 9.5.

1. Prepare a one paragraph summary of what we talked about since the last assignment. I do not want just a bulleted list of topics, I want you to use complete sentences and establish context (Why is what we have learned relevant? How does it connect with other classes?). The more insight you give, the better.

2. Provide feedback to your peers on Homework #3 in Canvas.

3. **1D optimization.** Consider the function

$$f(x) = \begin{cases} 25x^2 & x < 1 \\ x^2 + 48x - 24 & 1 \le x \le 2 \\ 25x^2 - 48x + 72 & x > 2 \end{cases}$$

   (a) Prove that f is strongly convex with parameter $m = 2$ and is M-smooth with $M = 50$.

   (b) What is the global minimizer of f? Justify your answer.

   (c) Run the gradient descent with $\alpha_k = 1/50$, the heavy ball method with $\alpha_k = 1/8$ and $\beta_k = 2/3$, and Nesterov's method with $\alpha_k = 1/50$ and $\beta_k = (k-1)/(k+2)$. In all cases initialize at $x_0 = 3$. Plot the function value versus the iteration counter for each method.

   (d) Repeat part (3c), tuning the parameters of each algorithm to achieve the fastest rate of convergence. Make a plot of the function value versus the iteration counter. What are the best parameters for each algorithm?

4. **Backtracking convergence.** Recall that when using backtracking to select a step size to move from $\boldsymbol{x}_k$ in direction $\boldsymbol{d}_k$, we start with $\alpha = \bar{\alpha}$, and then iteratively decrease α by factor of $\rho < 1$ until we satisfy the Armijo condition

$$f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k) \ \le \ f(\boldsymbol{x}_k) + c_1 \alpha \langle \boldsymbol{d}, \nabla f(\boldsymbol{x}_k) \rangle, \qquad (1)$$

   where $0 < c_1 < 1$ is some user-defined parameter.

   (a) Show that if f is M-smooth and $\boldsymbol{d}_k$ is a descent direction, then when

$$\alpha \ \le \ 2(c_1 - 1) \frac{\langle \boldsymbol{d}_k, \nabla f(\boldsymbol{x}_k) \rangle}{M \|\boldsymbol{d}_k\|_2^2}$$

   the Armijo condition (1) must hold. [Hint: Recall condition (iv) from Problem 5 of Homework 3.]

(b) Use this to derive an upper bound on the number of backtracking iterations (i.e., the number of times we have to multiply α by ρ) assuming $\bar{\alpha} = 1$ and $\boldsymbol{d}_k = -\nabla f(\boldsymbol{x}_k)$. Your answer should depend on ρ, c_1, and M.

5. **Proximal minimization.** Let f be a convex function that is M-smooth but *not* strongly convex. In this problem we will show that we can always form a strongly convex perturbation of f that will be close to f. In particular, suppose that we have an estimate $\boldsymbol{x}_0$ that we think is in the rough neighborhood of $\boldsymbol{x}^\star$ (e.g., if we have an upper bound on $\|\boldsymbol{x}^\star\|_2$, then $\boldsymbol{x}_0 = \boldsymbol{0}$ is a possible choice). Consider the function

$$f_\delta(\boldsymbol{x}) = f(\boldsymbol{x}) + \frac{\delta}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2.$$

(a) Let $\boldsymbol{x}_\delta^\star$ denote a minimizer of f_δ. Is $\boldsymbol{x}_\delta^\star$ unique?

(b) Argue that if $\boldsymbol{x}_0$ is a minimizer of f, then $\boldsymbol{x}_\delta^\star = \boldsymbol{x}_0$.

(c) Prove that $f(\boldsymbol{x}_\delta^\star) - f(\boldsymbol{x}^\star) \leq \frac{\delta}{2}\|\boldsymbol{x}^\star - \boldsymbol{x}_0\|_2^2$. Thus, if δ and/or $\boldsymbol{x}_0$ are chosen appropriately, $\boldsymbol{x}_\delta^\star$ does almost as good of a job at minimizing f as $\boldsymbol{x}^\star$.

(d) The modified function f_δ is now both M_δ-smooth as well as strongly convex (with parameter m_δ). Determine the parameters M_δ and m_δ.

6. **Socially distanced robots.** In class we saw that a way of making a swarm of robots meet as the same location was to have the robots move in the steepest descent direction to the cost

$$\sum_{n=1}^{N} \sum_{m \in \mathcal{N}_n} \|\boldsymbol{p}^{(n)} - \boldsymbol{p}^{(m)}\|_2^2$$

where $\boldsymbol{p}^{(n)}$ the position of robot n and $\mathcal{N}_n$ is the set containing the indices off all of robot n's neighbors. If we assume that the neighborhood relationship is symmetric, i.e., $m \in \mathcal{N}_n \Leftrightarrow n \in \mathcal{N}_m$ we showed that he steepest descent update law becomes

$$\boldsymbol{p}_{k+1}^{(n)} = \boldsymbol{p}_k^{(n)} - \alpha_k \sum_{m \in \mathcal{N}_n} (\boldsymbol{p}_k^{(n)} - \boldsymbol{p}_k^{(m)}),$$

where $\boldsymbol{p}_k^{(n)}$ is the position of robot n at time step k, and $\alpha_k > 0$ is the step size at time k.

(a) If the robots are socially distant, however, they should not try to meet at the same location, but end up a distance $\Delta > 0$ away from each other, which can be encoded through the cost

$$\sum_{n=1}^{N} \sum_{m \in \mathcal{N}_n} \left(\|\boldsymbol{p}^{(n)} - \boldsymbol{p}^{(m)}\|_2^2 - \Delta^2\right)^2.$$

What is the corresponding so-called formation controller that results from a gradient descent step on this new objective function?

(b) Implement this formation controller. Initialize your problem using

```
import numpy as np
np.random.seed(2021)
p = np.random.uniform(0,50,[2,10])
```

In the above, p is a 2×10 array that represents the initial locations of 10 robots that have been placed at random in a 50×50 square. In your controller, use a fixed step size (you will need to tune this to ensure convergence) and set $\Delta = 6$. Note that your initialization begins with some robots that are not respecting social distancing! Watch what happens to those robots over the first few iterations. Turn in a plot of the configuration that your algorithm converges to.

(c) Now consider the formation controller that would arise by using Nesterov's method to minimize this objective function. Write down the update rule. Next implement this controller on the same scenario as in the previous part. Again, used a fixed step size for α_k. Set $\beta_k = (k-1)/(k+2)$. Compare the results to what you obtained using simple gradient descent.

(d) We could also derive a controller for this problem using Newton's method. Comment on the benefits and drawbacks of using Newton's method in this context.

7. **Smoothed data fitting.** Suppose that we observe a sequence of N observations given by $y_n = x_n + e_n$, where e_n represents noise, and consider the case where x_n represents samples of some underlying smooth function, so that we do not expect x_n to be too different from x_{n-1} and x_{n+1}. One approach to "denoising" the observations y_n to obtain a smoothed estimate of x_n is to solve the optimization problem

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}} \ \frac{1}{2}\sum_{n=1}^{N}(x_n - y_n)^2 + \frac{\lambda}{2}\sum_{n=1}^{N-1}(x_{n+1} - x_n)^2.$$

(a) Calculate the gradient of this objective function. Write your answer in vector notation (in terms of the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbb{R}^N$). Your answer should involve the tri-diagonal matrix

$$\boldsymbol{D} = \begin{bmatrix} 1 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & 2 & -1 & \\ & & & & -1 & 2 & -1 \\ & & & & & -1 & 1 \end{bmatrix}.$$

(b) Calculate the Hessian matrix for this objective function.

(c) Write down the update equation for Newton's method. Simplify your answer as much as you can, but you may again leave things in terms of $\boldsymbol{D}$ (i.e., do not worry about explicitly calculating $\boldsymbol{D}^{-1}$).

(d) Do you think that Newton's method would be a scalable solution to this problem when N is very large? Why or why not?

8. **More logistic regression.** Recall the logistic regression problem from Homework 3. In this problem we will continue where we left off and evaluate the performance of new algorithms that we have learned about since the last homework.

   (a) Last time we implemented gradient descent using a bisection algorithm to choose the optimal α (up to some tolerance) at each iteration. Now try replacing this linear search with a standard backtracking algorithm. Experiment with the parameters c_1 and ρ. Report the c_1 and ρ that work best. Report how many gradient steps are now required when using this line search. Also report the total number of iterations taken by the combined backtracking searches. How does this compare to the results you obtained last time using bisection?

   (b) Next implement a solver for this problem using the heavy ball method. First try setting $\alpha \approx 0.001$ and $\beta_k \approx 0.95$ and run the code with these fixed step sizes. Play around with α and β to see if you can get any improvement. Report the values of α and β that work best, and how many iterations the method requires. Then try using backtracking (using the same parameters as before) to adjust α at each iteration (you can leave β fixed here). Now how many iterations are required?

   (c) Next implement a solver using Nesterov's method. Try both using a fixed stepsize of $\alpha \approx 0.001$ as well as backtracking. Use the rule of thumb $\beta_k = (k-1)/(k+2)$. Report how many iterations are required for both versions.

   (d) Next implement a solver using Newton's method setting α using backtracking (be sure to start with $\bar{\alpha} = 1$). Report the number of Newton steps required.

   (e) Finally, implement a solver using the BFGS method, again setting α using backtracking with $\bar{\alpha} = 1$. Report the number of quasi-Newton steps required.