MATH-UA 253/MA-UY 3204 - Fall 2022 - Homework #3

This homework is a bit long, so start early!

Stochastic gradient descent. Consider the optimization problem:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in X, \end{array} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $X \subseteq \mathbb{R}^n$ is the domain of the optimization problem. If $X \neq \mathbb{R}^n$ and we apply gradient descent:

$$x_{n+1} = x_n - \alpha_n \nabla f(x_n), \tag{2}$$

it might not be the case that $x_n \in X$ for all n. Indeed, the minimum of f taken over all of $\mathbb{R}^n$ may not lie in X! This is a *constrained optimization problem*—we will learn more about constrained optimization in the second half of the class. A basic algorithm for solving this minimization problem is *projected gradient descent* (PGD). The idea is simple: after we take each gradient descent, we *project* x_{n+1} back onto X—that is, we set x_{n+1} to be the closest point to in X. Define:

$$\operatorname{proj}_X(x) = \arg\min_{y \in X} \|x - y\|_2. \tag{3}$$

Then PGD takes the following form:

$$x_{n+1} = \operatorname{proj}_X(x_n - \alpha_n \nabla f(x_n)). \tag{4}$$

Evaluating proj_X requires us to solve another optimization problem, which might be quite costly and complicated! However, for a variety of important but simple constraint sets, it is easy to project x onto X. The constraint set for the Thomson problem is one of the easy ones: the closest point to $x \in \mathbb{R}^n$ on the unit sphere $S^{n-1} = \{y \in \mathbb{R}^n : \|y\|_2 = 1\}$ is just:

$$\operatorname{proj}_{\mathbb{S}^{n-1}}(x) = \begin{cases} \frac{x}{\|x\|} & \text{if } x \neq 0, \\ \text{undefined} & \text{if } x = 0. \end{cases} \tag{5}$$

The Thomson problem. Consider a system of N particles at positions $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ (where $\boldsymbol{x}_i \in \mathbb{R}^3$), where particle i has charge q_i. The electrostatic potential energy stored in this system is:

$$U_E(N) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j \neq i}^{N} k_e \frac{q_i q_j}{r_{ij}}, \tag{6}$$

where $r_{ij} = |\boldsymbol{x}_i - \boldsymbol{x}_j|$. If we normalize the units so $k_e = 1$ and assume that $q_i = 1$ for all i, then $U_E(N)$ simplifies to:

$$U_E(N) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j \neq i}^{N} \frac{1}{r_{ij}}. \tag{7}$$

Now, assume that we have N electrons which are *constrained to lie on the surface of the unit sphere in* $\mathbb{R}^3$. The *Thomson problem* is the following:

$$\underline{\text{For } N > 0\text{, find the configuration of electrons on the unit sphere such that } U_E(N) \text{ is minimized.}}$$

That is, solve:

$$\begin{array}{lll} \text{minimize} & U_E(N) & \\ \text{subject to} & |\boldsymbol{x}_i| = 1, & i = 1, \ldots, N. \end{array} \tag{8}$$

For this problem, you should write a version of PGD to solve the Thomson problem. Comments:

- You will need to use line search to get your iteration to converge. It's recommended to just use backtracking line search.
- You will need a reasonable way of distributing points on the sphere. This is simple:
  1. For each i, choose $\boldsymbol{x}_i = (x_i, y_i, z_i)$ randomly such that $x_i, y_i, z_i \sim \mathcal{N}(0,1)$ independently and identically. On Tuesday (9/27) we will discuss line search in more detail.
  2. Set $\boldsymbol{x}_i := \boldsymbol{x}_i/|\boldsymbol{x}_i|$.

  You can do this in numpy as follows:

  ```
  X = np.random.randn(N, 3)
  X /= np.sqrt(np.sum(X**2, axis=1)).reshape(-1, 1)
  ```

Visualization. To visualize your solution, you have two options:

1. You can use matplotlib's mplot3d to make a 3D plot of the (x_i, y_i, z_i) coordinates (see this link for some ideas).
2. You can alternatively use PyVista to do the same.

Problem 1. Solve the Thomson problem for several choices of N where $3 \le N \le 14$, and verify that your results match the picture here:

`https://tracer.lcc.uma.es/problems/thomson/thomson.html`

Do this by visualizing the equilibrium distribution of nodes using the approach described in "Visualization" above, and also by comparing the resulting potential with what is listed in the table on the website linked above.

Problem 2. Solve the Thomson problem for a few choices of N greater than $N = 14$. The Thomson problem is *nonconvex*, and thus has numerous local minima. If you re-run your optimization algorithm for different initializations, you may get different minimizing values. Try re-running your solver for different starting configurations by selecting a new initial iterate randomly. If you like, you can use `np.random.seed` to make your results reproducible. Visualize each result in 3D, as discussed above. Are the final values of the electrostatic potential comparable, or are they very different?

Problem 3. The URL above also gives the following fit to the value of the potential for different N:

$$U_{\text{approx}}(N) = \frac{N^2}{2}\left(1 - aN^{-1/2} + bN^{-3/2}\right), \tag{9}$$

where $a = 1.10461$ and $b = 0.137$. For the N that you chose in Problem 2, visualize your combined results by making a plot where:

- The horizontal axis is N, and the vertical axis is U—both axes should be linear (so, use `plt.plot`).
- Plot U_{approx} using the formula above.
- Make a scatter plot of your results for comparison.

How well do your results match the predicted minimum electrostatic potential?