Coding Assignment-1
MA6100: Maths Behind Machine Learning

September 21, 2023

Given an arbitrary function $f : \mathbb{R}^n \to \mathbb{R}$ having gradient ∇f, the gradient descent algorithm is an iterative procedure used to search for a local minimum. The vanilla version works as follows: Fix a (sufficiently large) N and a small $\eta > 0$.

- Randomly choose an initial point $x_0 \in \mathbb{R}^n$
- For $k = 1 \ldots N$

$$x_{k+1} = x_k - \eta \nabla f(x_k)$$

- Output $(x_N, f(x_N))$

1. For each of the following functions, find the minimizer by setting the gradient to zero, and also using gradient descent (in python) with $N = 1000$ and $\eta = 0.02$. You can use `np.random.rand` [1] to select $x0$

   (a) $f(x) = x^\top Ax + b$, where $A = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & 0 \\ -1 & 0 & 1 \end{bmatrix}$, $b = [1]$

   (b) $f(x) = ||Ax - b||^2$, where $A = \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 1 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix}$

   (c) $f(x) = ||Ax - b||^2$, where $A = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 3 & 1 & 9 \\ 4 & 1 & 0 \\ 2 & 1 & 4 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ 3 \\ 1 \\ 0 \\ 9 \end{bmatrix}$

2. In this problem, your goal is solve $\min_{x \in \mathbb{R}^{10}} f(x)$, where $f(x) = x^\top Ax - 2b^\top x + c$. Generate A randomly using `sklearn.datasets.make_spd_matrix` [2], and generate b and c using `np.random.rand` function of numpy. Once you obtain A, b, c, keep them fixed for the entire problem. Solve this min. problem using the following different methods:

[1] https://numpy.org/doc/stable/reference/random/generated/numpy.random.rand.html

[2] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_spd_matrix.html

(a) Analytically, by setting gradient to zero. (You can verify that f is convex).

(b) Numerically using Gradient descent: Initialize $x = [\frac{1}{10}, \frac{1}{10}, \ldots, \frac{1}{10}]$. Step-size for Gradient Descent: $l_{GD} = \frac{1}{2\|A\|+\|b\|_2}$ where $\|A\|$ is the spectral norm of A. The Spectral norm of matrix A can be computed using `np.linalg.norm(A, ord=2)`. The norm of b can be computed using `np.linalg.norm(b)`. Run for 1000 iterations.

Make a plot of iterations (x-axis) vs objective value (at that iteration; y-axis) with gradient descent. And draw a horizontal line at the optimal objective value obtained using the analytical solution. Ideally, GD must converge to the horizontal line as iterations increase.