EE2703: Assignment 5 - Gradient Descent

Atreya Vedantam, EE22B004

22nd December 2022

1 Gradient Descent Function

The gradient descent function that I have written works for an n-dimensional function. The function takes in as inputs:

- `guess` which is a list of components that together correspond to a single point in N-dimensional space. This is the starting point of the Gradient Descent Algorithm
- `f` which is the objective function to be optimized
- `fgrad` which is the gradient of the function
- `ranges` which is a n-list of two values each corresponding to the minimum possible value that a component can take and the maximum possible value that a component can take
- `a` which is the learning rate of the algorithm (set to 0.1)
- `iterations` which is the *maximum* number of steps that the algorithm performs. Note that this is only to ensure a breaking of the `while` loop in case the relative error never drops below the tolerance (breaking while loops is always a good programming practice)
- `tol` which is the tolerance for relative error which is used to break the `while` loop

In addition I have also written a `plotFunc` function which given the function, a range of plotting and a resolution of plotting, generates the plot along with the gradient descent steps performed on it. Plots from that function are shown below.

1.1 Restrictions on possible input functions

The function must be continuous and differentiable (at least in the range of interest). This is because the gradient of a function is only defined for such functions.

1.2 Approach

Start with the initial guess (call it $\mathbf{x_0}$). Perform iteratively

$$\mathbf{x_{i+1}} = \mathbf{x_i} - \nabla_{\mathbf{x}}\mathbf{f}(\mathbf{x}) \times \alpha$$

I have chosen the learning rate $\alpha = 0.1$. At each iteration I check for the value of each component of $\mathbf{x}$. If it lies beyond the allowed range (given by the list `ranges`), I set it to either the maximum or the minimum of the range depending on its value. Now I check for the relative error of the functional values $\mathbf{f}(\mathbf{x_i})$ and $\mathbf{f}(\mathbf{x_{i+1}})$.

The expectation is that these functional values approach the same limiting value as the algorithm converges (Cauchy condition). Hence if it falls below a `tolerance` then I terminate the algorithm and pronounce the $\mathbf{x_i}$ the optimal solution. If for some reason this convergence doesn't occur, I don't want to be stuck in an infinite `while` loop, so there's a maximum number of iterations that the algorithm will perform otherwise (set to 10000 in these problems).

2 Problem 1

This is a one dimensional quadratic function and hence has only one minimum. Hence our choice of starting point has little consequence. The figure below illustrates the function and the gradient descent points in red.

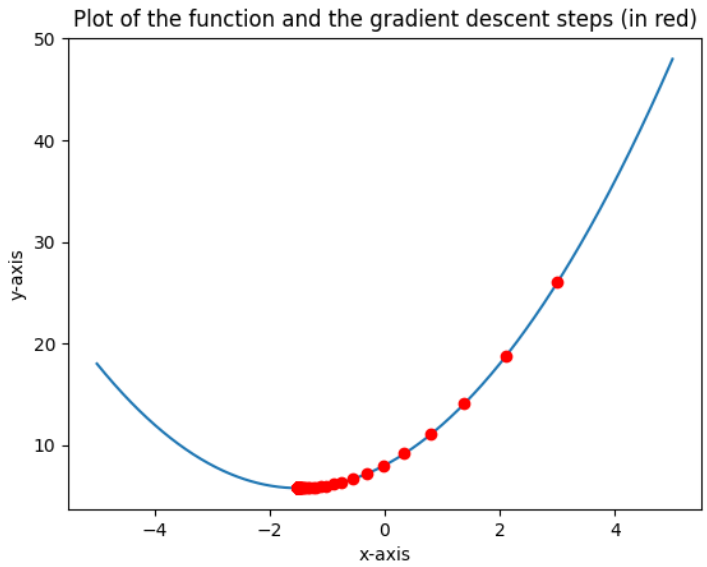

Figure 1: Gradient Descent performed on a quadratic one-dimensional function

3 Problem 2

This is a two dimensional polynomial. The range of values to search for is also given. Plotting the function gives us an idea of where to start. I have chosen to start at $(6, 2)$. The function is mostly flat around the minima and hence it the algorithm converges to roughly the same point despite using slightly different initial points. It becomes very steep on both sides along the x-axis but it maintains its strong convexity, thereby ensuring the convergence. Below is the plot for the second function.

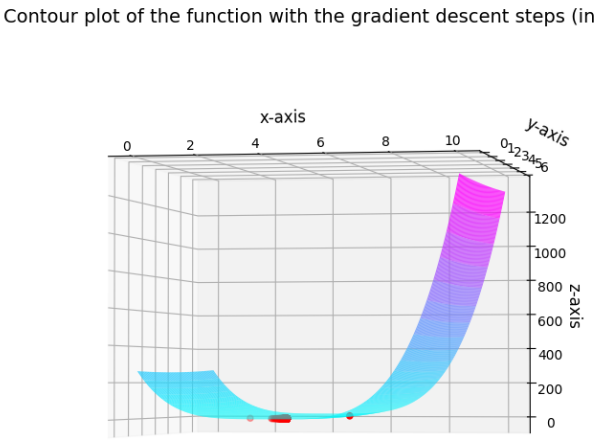

Figure 2: Gradient Descent performed on a polynomial two-dimensional function

4 Problem 3

This is a two-dimensional function with some periodicity. The x-range is already specified. I have chosen the y-range to be $(-5, 10)$ because that contains the global minimum (observe the plot). A guess of $(0, 0)$ seems to be a good guess - enough to tip the gradient descent algorithm into the minimum but not too close to it to be seen as spoonfeeding. The resulting plot is quite satisfactory. The algorithm begins at a functional value of roughly 0 and descends inside the 'hole' as seen.

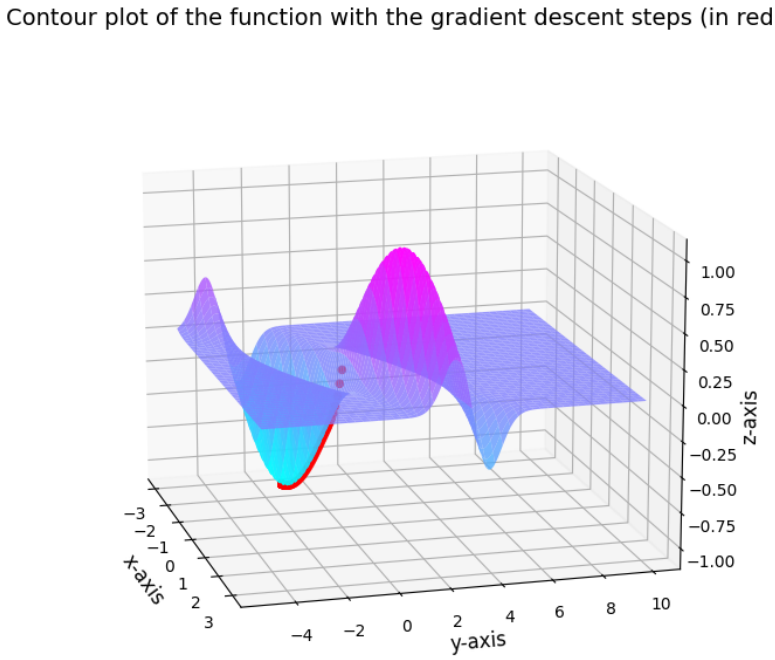

Figure 3: Gradient Descent performed on a two dimensional function with a periodicity

5 Problem 4

We are not given the gradient of the function here. I thought of using numerical differentiation but sir mentioned that it was not allowed. Hence I manually computed its derivative. A guess of 3 seems to be the most appropriate. A slightly greater guess pushes the algorithm into a local minimum on the other side. The plot below may help convey this point better.

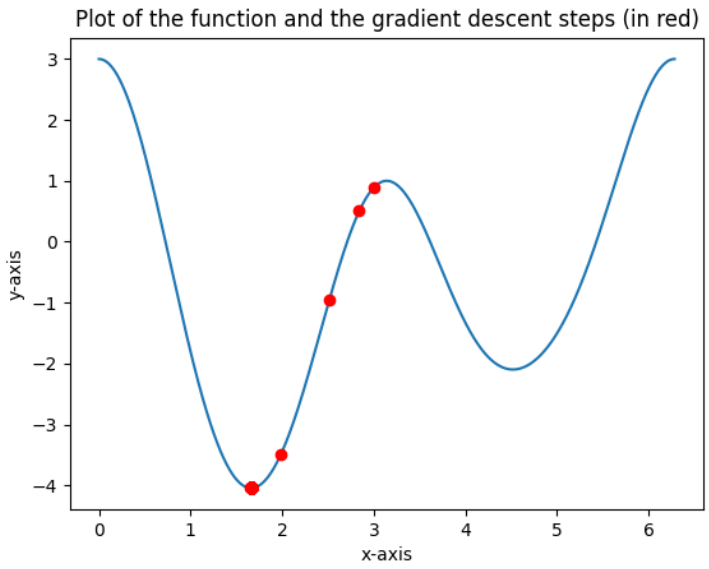

Figure 4: Gradient Descent applied on a one-dimensional periodic function

6 Results

The final optima for each of the four functions along with their functional values are:

- The minimum occurs at x = [-1.5] and f(x) = [5.75]
- The minimum occurs at x = [3.98882448 2.] and f(x, y) = 2.000000015597948
- The minimum occurs at x = [-1.57079633 -1.57079633] and f(x, y) = -1.0
- The minimum occurs at x = [1.66166081] and f(x) = [-4.04541205]