Assignment 6 - Gradient Descent Report

EE22B175

Overview

The `gradDescent` function is a Python implementation of the gradient descent optimization algorithm for finding the local minimum of a given function. This algorithm is widely used in numerical optimization to iteratively update a set of parameters in order to minimize a cost or objective function.

The algorithm supports both 1-dimensional and 2-dimensional problems. In 1-dimensional problems, it finds the local minimum for a function of a single variable, while in 2-dimensional problems, it finds the local minimum for a function of two variables.

Function Signature

```
gradDescent(startingX, learningRate, func, xDerivative, xLims, stepCount, problemName,
leeway = 3, startingY = None, yDerivative = None, yLims = None,
numberOfPoints = 300, azimuth = -80)
```

Parameters

- `startingX` : Initial value for the independent variable X.
- `learningRate` : The step size for adjusting the X value in each iteration.
- `func` : The function to be minimized.
- `xDerivative` : The derivative of the function with respect to X.
- `xLims` : A list specifying the lower and upper bounds for the X variable.
- `stepCount` : The number of iterations for the gradient descent.
- `problemName` : A string identifier for saving the animation and plots.
- `leeway` : A parameter to adjust the plotting limits (optional, default is 3).
- `startingY` : Initial value for the independent variable Y for 2D problems (optional).
- `yDerivative` : The derivative of the function with respect to Y for 2D problems (optional).
- `yLims` : A list specifying the lower and upper bounds for the Y variable for 2D problems (optional).
- `numberOfPoints` : The number of points used for plotting (optional, default is 300).
- `azimuth` : The azimuth angle for 3D plots (optional, default is -80).

Restrictions

- The gradient descent function defined above can take any function as input as long as it is **differentiable**. This means that the function must have a well-defined derivative for all possible values of its input(s), at least within the limits specified.
- In addition to being differentiable, the function should also be **continuous**. This means that there should be no gaps or sudden jumps in the function's output.
- The function should be **defined** within the specified range of values. If the function is not defined at a particular point, the gradient descent algorithm may not be able to converge to a minimum.

Algorithm Steps

1. The function initializes the metadata for saving the animation.
2. It creates a `PillowWriter` object for saving the animation with specified frames per second (FPS) and metadata.
3. For 1D problems (where `startingY` is not provided):
    - It generates a linear space of X values within the specified limits.
    - Evaluates the function at these X values to obtain corresponding Y values.
    - Sets up a plot to visualize the function.
    - Initializes the starting point and trajectory lists.
    - Iterates through the specified number of steps:
        - Applies periodic boundary conditions if needed.
        - Updates the X value by moving it in the opposite direction of the gradient.
        - Computes the corresponding Y value.
        - Appends the new values to the trajectory lists.
        - Saves frames for the animation.
    - Prints and displays the local minimum found.
4. For 2D problems (where `startingY` is provided):
    - It creates a grid of X and Y values within specified limits.
    - Evaluates the function over the grid to obtain Z values.
    - Sets up a 3D plot to visualize the function.
    - Initializes the starting point and trajectory lists.
    - Iterates through the specified number of steps:
        - Handles periodic boundary conditions if necessary.
        - Updates X and Y values based on gradient descent.
        - Calculates the corresponding Z value.
        - Appends the new values to their respective trajectory lists.
        - Saves frames for the animation.
    - Prints and displays the local minimum found.

Clever Implementation

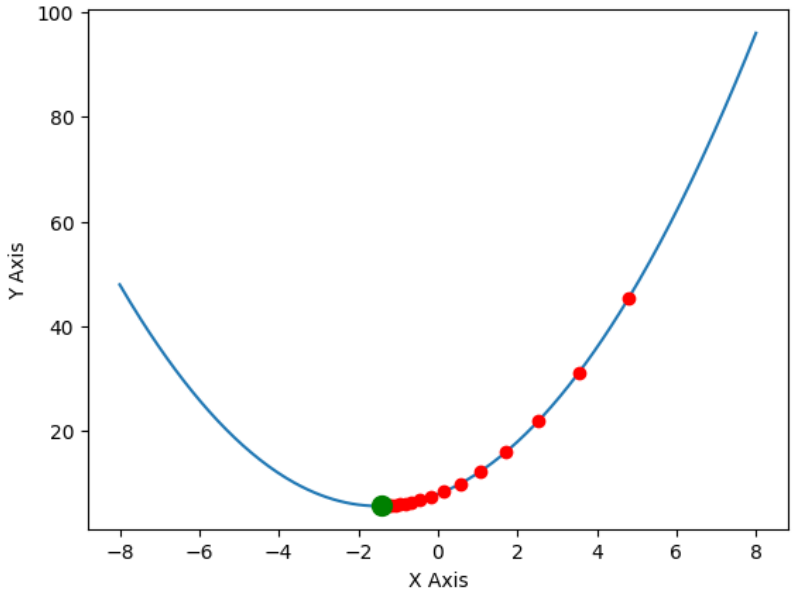
- To take care of needing to look for the best solution, optimum, only in the range specified, every time the value of the independent variable(s) gets updated by the Gradient Descent Algorithm, its value is brought to be within the limits of search that have been specified.
- This makes the algorithm more reliable and also takes care of overflow that may occur when large values have to be calculated, that is if we know that the "search area" does not contain any values that are too large to overflow.

Problems

The large Green Dot in each of the graphs represents the minimum found using Gradient Descent,
while the Red and/or Black points are the points through which the algorithm has traversed.

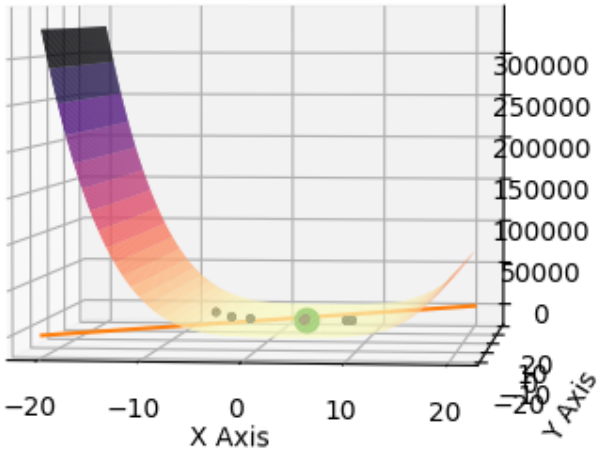
Problem 1 - Simple 1D Function

The function is $y = x^2 + 3x + 8$, and its derivative, with respect to x, is $2x + 3$. The minimum of this function is at $x = -1.5$ and $y = 5.75$, the point $(-1.5, 5.75)$. Using gradient descent, the local minima obtained is very close the the actual minima. As it depends on the staring value, which is randomized, the learning rate, and the number of steps, if these knobs are adjusted properly it could be made to practically converge to the global minimum.



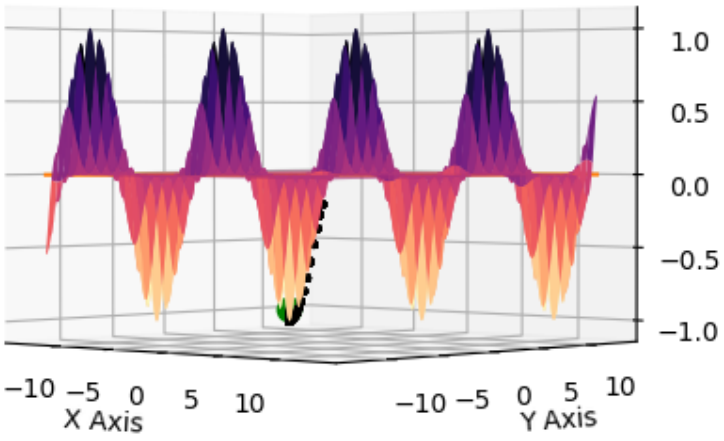
Problem 2 - 2D Polynomial

The function is $f(x, y) = z = x^4 - 16x^3 + 96x^2 - 256x + y^2 - 4y + 262$, and its derivative with respect to x is $f_x = 4x^3 - 48x^2 + 192x - 256$, its derivative with respect to y is $f_y = 2y - 4$.



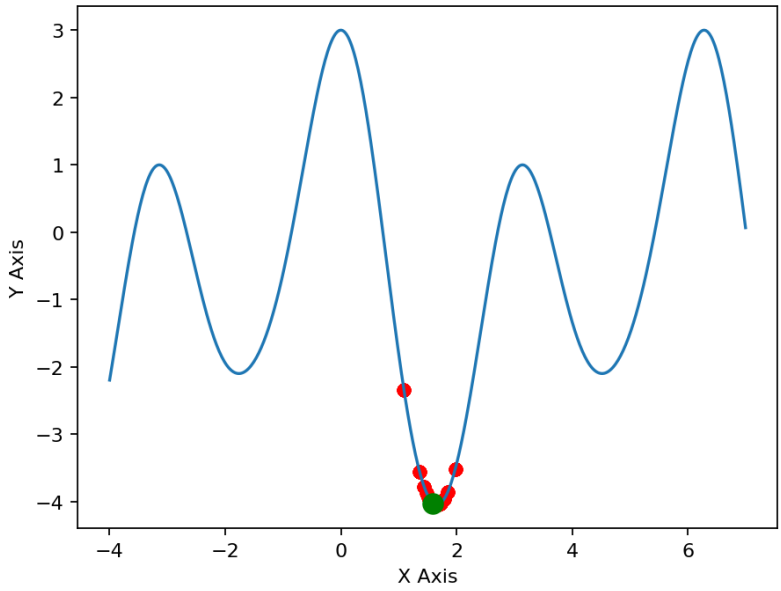
Problem 3 - 2D Function

The function is $f(x, y) = z = e^{-(x-y)^2} \cdot \sin(y)$, and its derivative with respect to x is $f_x = -2 \cdot e^{-(x-y)^2} \cdot \sin(y) \cdot (x - y)$, its derivative with respect to y is $f_y = e^{-(x-y)^2} \cdot \cos(y) + 2 \cdot e^{-(x-y)^2} \cdot \sin(y) \cdot (x - y)$. The given function is a periodic function and hence, the minimum has to be looked for only between $-\pi$ and π in the x and y.



Problem 4 - 1D Trigonometric

The function is $y = \cos(x)^4 - \sin(x)^3 - 4\sin(x)^2 + \cos(x) + 1$, and its derivative, with respect to x, is $-4\sin(x)\cos(x)^3 - 3\cos(x)\sin(x)^2 - 8\sin(2x) - \sin(x)$. The minimum of this function is at $x = 1.662$ and $y = -4.045$, the point $(1.662, -4.045)$.



Link to a Google Drive Folder Containing Animations as GIFs of the Gradient Descent Algorithm

Observations

- The `gradDescent` function is a versatile tool for visualizing and finding local minima of functions, whether they are 1D or 2D.
- Gradient descent is a greedy algorithm, which means that it does not have any knowledge of the global minimum of the function. It is therefore possible for gradient descent to get stuck in a local minimum.
- The learning rate is an important hyperparameter in gradient descent.
- The animation generated helps in understanding the optimization process.

Limitations

- The algorithm's effectiveness depends on the choice of learning rate and the function's characteristics. If the learning rate is too large, the algorithm may diverge, and if it's too small, it may converge very slowly.
- Gradient descent is a local search algorithm, which means that it can only find local minima. It is therefore important to initialize the algorithm with a good guess for the minimum.
- Gradient descent is sensitive to the noise in the function. If the function is noisy, the algorithm may converge to a spurious minimum.
- Gradient descent can be slow to converge, especially for high-dimensional functions.
- For more complex functions or high-dimensional problems, finding the right learning rate can be challenging.

Workarounds

- Experiment with different learning rates to find the optimal rate for convergence.
- Use more advanced optimization algorithms, like Adam or L-BFGS, for complex or high-dimensional problems.
- To avoid getting stuck in a local minimum, multiple initial guesses can be used. Additionally, momentum and adaptive learning rate techniques can be used to improve the performance of the algorithm.
- To reduce the sensitivity to noise, the function can be smoothed or regularized.