Gradient Descent Optimization and Cost Function Visualization on a Noisy Linear Dataset

Procedure

1. Generate Sample Data
    - Create a synthetic dataset using numpy's random functions. This dataset should resemble a linear trend with some added noise.
      Generate Data Points:
        - Decide on the number of data points. Let's assume $m = 100$ for this example.
        - Decide on a linear function for the underlying trend, e.g., $y = wx$. Let's take $w = 4$.
        - Use numpy to generate random noise. The randn function is useful as it returns samples from the "standard normal" distribution.
        - Multiply the noise by a factor to decide its magnitude. For instance, a factor of 1.5 will produce a moderate amount of noise.
        - Create a linearly spaced set of x values.
        - Calculate the corresponding y values based on the linear function and add the noise.
        - Plot the generated data to visualize the linear trend.
2. Initialize Parameters for Gradient Descent
    - Define the learning rate, number of iterations, and initial parameter values for the linear regression model.
3. Implement Gradient Descent
    - Write a function named compute_cost that calculates the mean squared error of your model given current parameter values.
    - Write a function named gradient_descent that will adjust the parameter values using the gradient descent algorithm.
    - Using the above functions, compute the optimal parameters for the given synthetic dataset.
4. Visualize the Results
    - Plot the linear regression line with the optimal parameters on the same graph as your data points.
    - Plot the cost history over iterations to understand the convergence of the gradient descent algorithm.
5. Discussion
    - Discuss the importance of the learning rate. What happens if it's too high or too low?
    - How does the number of iterations affect the result? Is there a point where increasing the number of iterations doesn't provide much benefit?