Gradient Descent Exercise

February 26, 2024

Old Faithful is a geyser in Yellowstone National Park. The intervals between the geyser's eruptions can predicted based on the durations of its previous eruptions using linear regression. Specifically, we can predict the interval (in minutes) before the next eruption based on the duration (in seconds) of the previous eruption using a forumula of the form

$$interval = w_0 + w_1 * duration$$

Twelve data points from Old Faithful are included in the CSV file geyser.csv. The first column lists duration in minutes, and the second lists the intervals in seconds. The goal of this homework is to use gradient descent to find appropriate values for w_0 and w_1.

Part I

The Python script batchgrad.py implements batch gradient descent–mostly. However, it is missing a crucial function–the function computes the partial derivatives of the cost function. Using the formula we discussed in class, please implement this function.

Before you start coding, you should familiarize yourself with the code. Currently, it contains code that implements several crucial functions, including computing the cost function, reading in the data to a numpy array, and printing the results.

Once you have implemented the compute_grad function, run your code on the Old Faithful data. In a comment, state the equation you found and how many times you had to loop through the data before the equation was found. What would it predict as the interval for a 3 minute (180 second) duration?

Part II

If you finish batch gradient descent, try making a new version of your code that does stochastic gradient descent. In the previous section, when computing the derivatives, you should have computed the derivatives for the cost function over the entire data set. In stochastic gradient descent, we instead look at one data point at a time, calculating the derivative of the cost function for that point, and updating the weights accordingly.

Write a new script that implements stochastic gradient descent for the data in geyser.csv. You can (and should!) reuse code from batchgrad.py in this script. Again, report how many iterations your code ran (making sure to count iterations as complete loops through the data, not the number of times the weights were updated), as well as the final equation. Does it make the same prediction for a 3-minute duration?