Kosuke Takahashi

Liu

CS596

October 4th, 2019

Homework Assignment 2 Write-up

Algorithm:

The algorithm that was used to solve this linear regression assignment was using gradient descent. I decided to write my algorithm using gradient descent because it was what the given code alluded to me using. To write the gradient descent, I needed to calculate the cost which ended up being “﻿(1/2\*m) \* np.sum(np.square(residualError-y))” and I needed to calculate the new theta value which ended up being “﻿theta - (1/m)\*alpha\*(X.T.dot((gradient - y)))”. I had to constantly calculate the new theta value as the algorithm iterates through the set of data hence the reason why theta is always subtracted by itself. In the gradient descent method, it takes in three different vector inputs, ‘X’, ‘y’, and ‘theta’ and calculates accordingly using ‘alpha’ and ‘numIterations’. ‘alpha’ is a variable that acts as the learning rate of this machine learning algorithm and ‘numIterations’ is the number of times that it will iterate through the for loop. This gradient descent algorithm allows convergence to the solution while taking into account ‘alpha’ and ‘numIterations’.

Results:

My initial test for this code was setting ‘ALPHA’ to 0.01 and ‘MAX\_ITER’ to 100 in main\_ha2.py. This resulted in a generally normal descent. The slope did not seem too steep, but as expected, with more iterations, the cost goes down.

A screenshot of a social media post

Description automatically generated

With my next test, I increased ‘MAX\_ITER’ to 500 to see where the curve will start to flat out. When I ran the code, the graph that was outputted showed around where the curve started to flat out which was around 220 iterations. It was not completely flat as there is a small slope. But with the new input values, I was able to tell that initially, cost reduces faster per iteration, but after around 75 iterations, that rate slows down.

A screenshot of a cell phone

Description automatically generated

For my next test, I increased ‘ALPHA’ to 0.5 while keeping ‘MAX\_ITER’ at 500. The results were very drastic as there was a straight line heading down as cost went to 0 for the initial iteration. Then there is a minute curve that round out the graph.

A screenshot of a cell phone

Description automatically generated

For my last test, I decreased ‘ALPHA’ to 0.005 while keeping ‘MAX\_ITER’ at 500. The results looked similar to the first test as the curve did not start to flat out until later through the iterations. The curve started to flat out at around 400. This helped me make a solution to the relation of ‘ALPHA’ and ‘MAX\_ITER’

A screenshot of a cell phone

Description automatically generated

Each test helped me deduce the relationship between ‘ALPHA’ and ‘MAX\_ITER’ and how it affects the cost-iteration relationship as well. As ‘ALPHA’ decreases, there are more iterations that is needed for the curve to reach its maximum number of iterations. But for all the examples, the cost reduces quicker in through the initial iterations, but after that it slows down.

Extra:

While researching on how to solve this linear regression assignment, I came across the library “scikit-learn” and “sklearn” as they are useful tools in machine learning as it helps normalize and standardize datasets. But for this assignment it was not needed, but seeing how prevalent these libraries are, I will most likely research deeper into these libraries for future assignments.