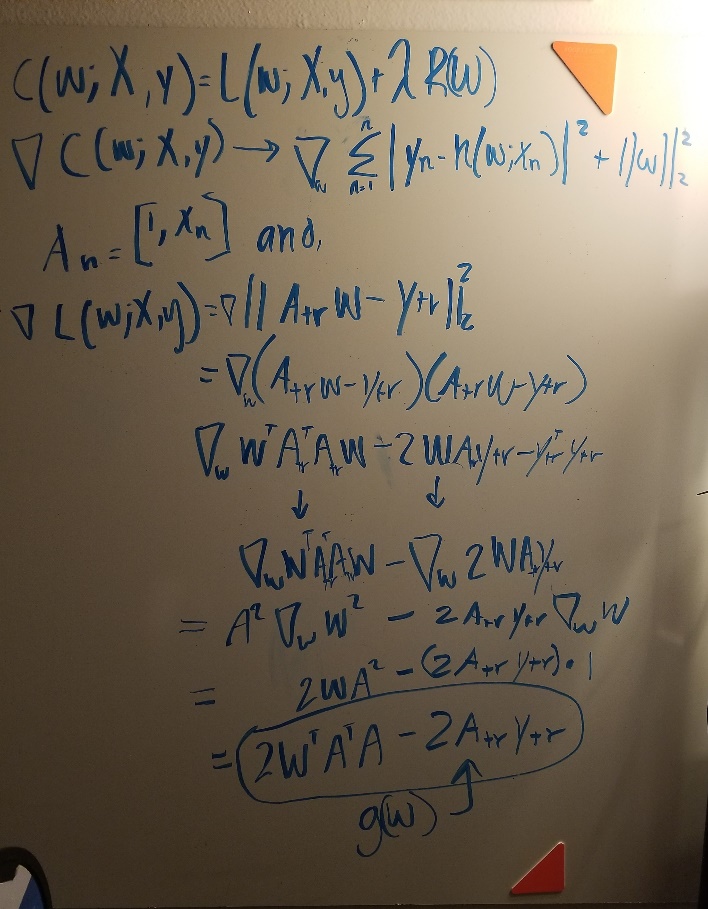
AI EXP : HW3 : Linear Regression

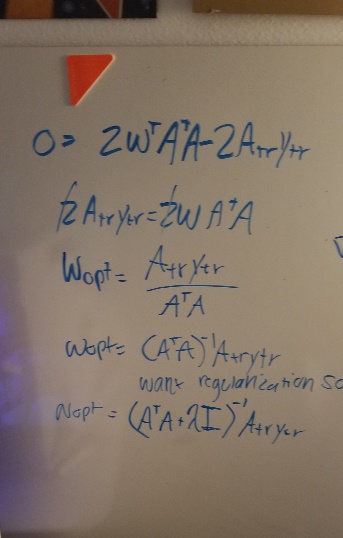
Alex Lamarche

**Problem 1: Setting up the functions**

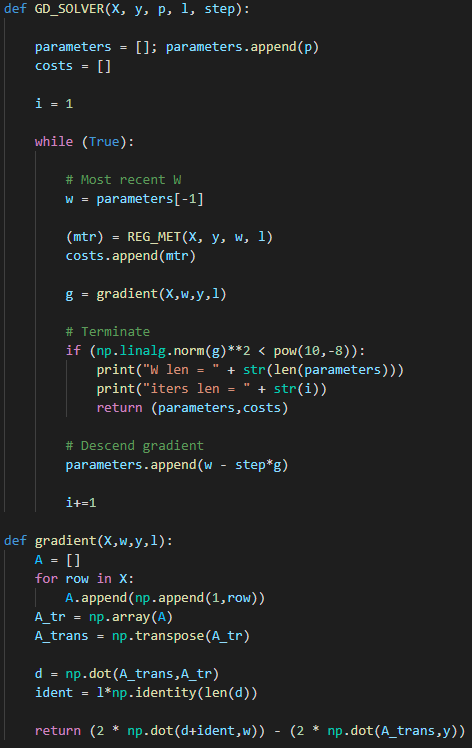
1.A - Scripts included or check the GitHub link here : <https://github.com/ajl3545/AI_HW3>

1.B – Derivation of g(w) the gradient function

1.C Derivation of g(Wopt) the gradient function whose value is: g(w) = 0



1.D – Gradient Algorithm presentation and effects of α and τ:

*Discussion –* α represents the *step* that must be taken towards the minimal value of the computed gradient. Each of the w values changes with the gradient descent iterations - towards an optimal set of parameters. By subtracting the previous set of w parameters by the current gradient “direction,” computed by gradient(W,x,y,z), the algorithm gets closer to a solution that minimizes the cost function. Step is a multiplier that is used to get closer to the minimum of the gradient function. Increasing the step could lead to overfitting and would overshoot on the gradient.

τ, a threshold per se, is used to compute how close the parameters get us to an optimal regression line. First, we step towards the minimum, then we check how close we are. τ ensures that we are close to the minimal value. If the summation of the squares of the parameters (the cost at that parameter iteration) is lower than τ, then that means the most recent w parameters have- as closely as possible and within reason – calculated a minimal cost.

\*ignore the print statements\*

**Problem 2 – Linear Regression (true function is affine)**

2.A <https://github.com/ajl3545/AI_HW3/blob/main/problem2.py> or check out included file

2.B – Present the results of CF\_SOLVER on training and testing data and show MSE for each:

Training data:

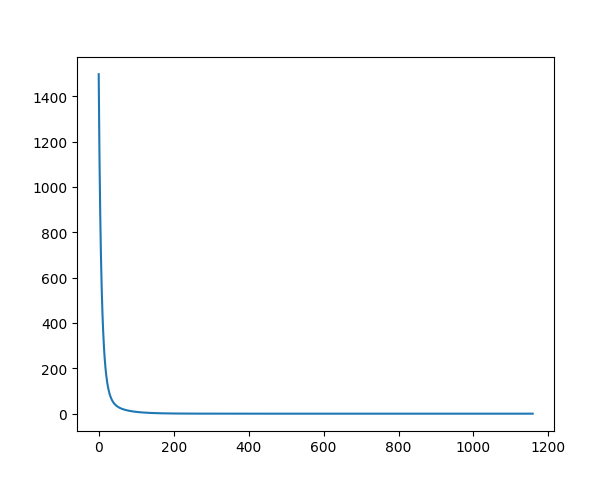


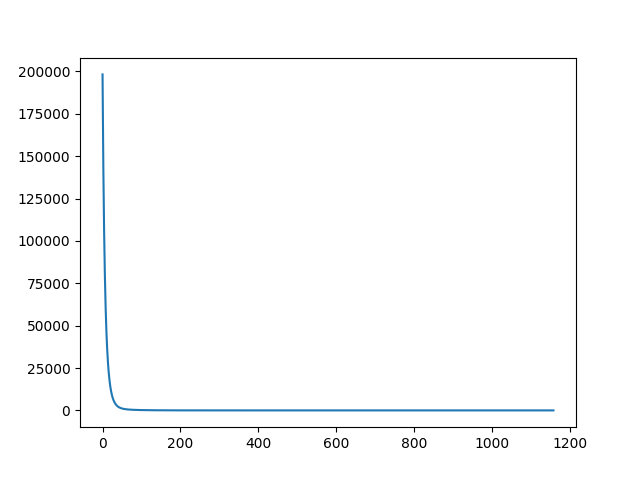
Testing data:



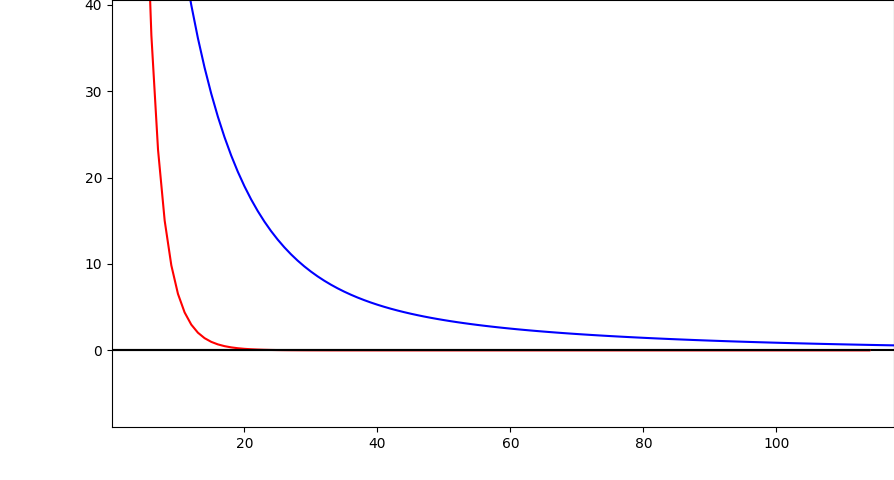
The Results: barely a difference, with the Wopt values nearly being the same. The testing data however, had more data and took less time to run. The training data took nearly 10x more iterations to come up with a result.

(cont.)

2.C – **Figure 1**. Regression objective(y) versus GD iteration (x). As the iteration progresses along the x axis, the cost metric reduces – which is expected since we want the cost to be minimal.

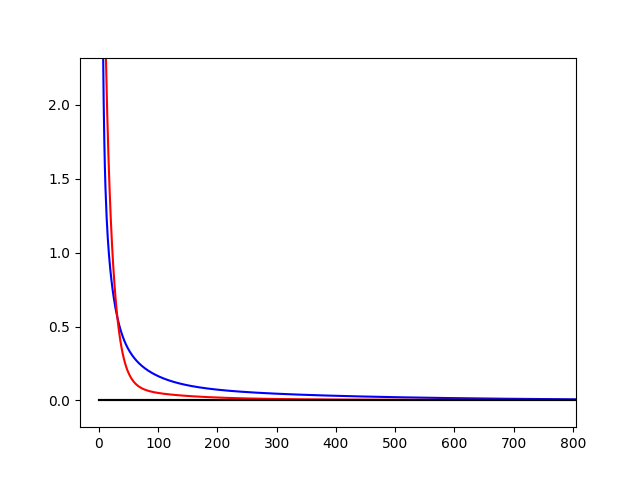
2.D – **Figure 2.** Euclidean norm (y) versus the iteration (x). Total # of iterations = 1158 on training data:

2.E – **Figure 3.** The metric lines (black) for both testing and training overlap since they are so small. Plotted are the MSE values for training (blue), testing, (red), and the benchmark MSE values (black) calculated by CF\_SOLVER. Both MSE’s converge close enough to the benchmarks to consider calculation successful.



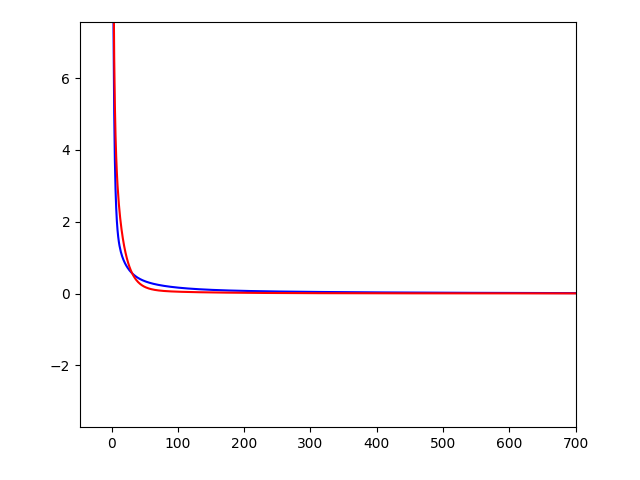
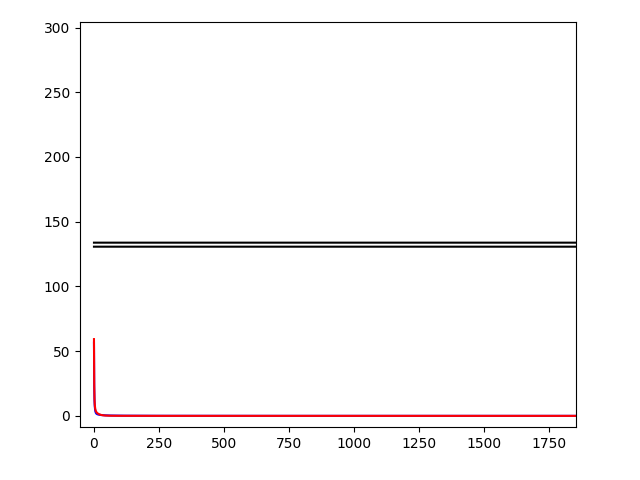
2.F – The benchmarks were meaningful since they predict a successful linear regression. Since the benchmark values aim to be at zero, the MSE lines must align with the benchmark as the iterations grow along the X axis. The MSE’s are both aligning with the benchmark.

2.G – **Figure 4.** There isn’t even 10 lines in the testing data to compare with… I still managed to run the plots. It seems that when the dataset is lower, more iterations are needed to converge to a minimum. Less data means more effort to calculate a relationship between variables. When there is more data, the linear relationship becomes more evident more quickly:



(cont.)

2.H – **Figure 5.** With lambda = 2. There were nearly 5k iterations on the training data and over 20k iterations on the testing data. I assume that with a large enough lambda, the data overfits or overshoots? The benchmark data is out of whack and doesn’t represent a proper value at all. Affecting the regularization skews the accuracy:



**Problem 3 - Linear Regression (true function is affine + noise)**

3.A – Check out the GitHub link : or see attached file