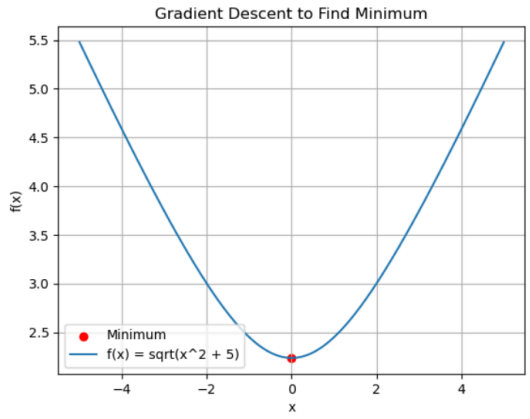
Britney Zhu

AMS 595.01 Fall 2025

Project #6 Report

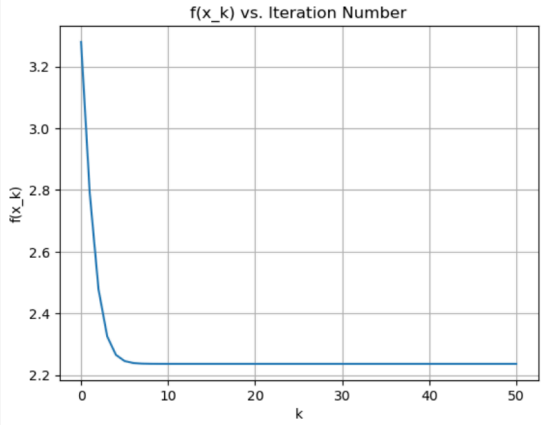
Part 1: Implementing Gradient Descent Algorithm

The purpose of this part of the project is to see the gradient descent algorithm play out. We want to visualize the gradient descent optimization algorithm on the function f(x) = sqrt(x^2+5). Because it is gradient descent, we have to do an iteration method to find the minimum by moving the directions of the gradient descent. Here we compared the different step sizes we used in our algorithm.



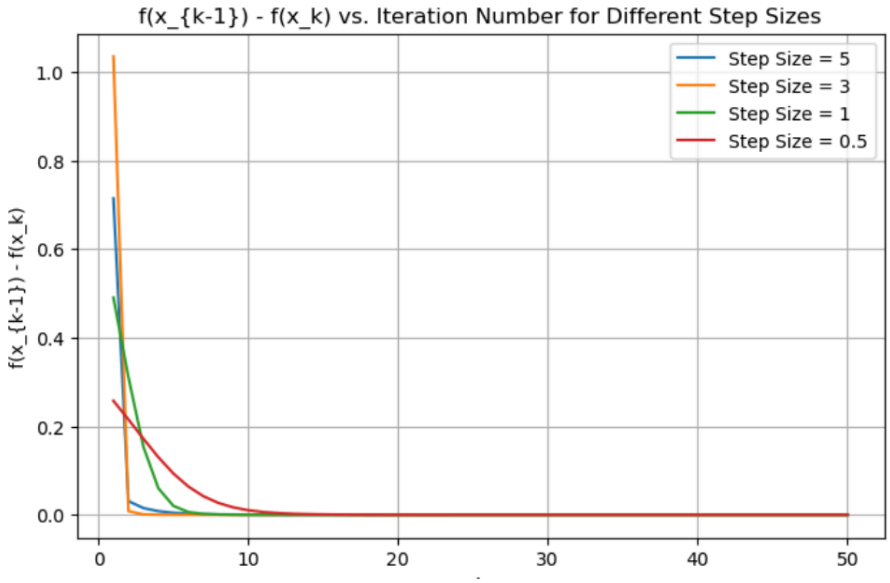
First we plotted the function sqrt(x^2+5), with its minimum at x=0, fx =sqrt(5) which is approx. 2.2360679775.

```
Final x_T = 5.3130210921243 61e-13
Final f(x_T) = 2.23606797749979
```



Then we created a gradient descent function. Here, we used step size 1 with initial guess 7/5+1 (I did 7 as my last ID number is 7). We can see that the value of fx decreases and gets closer to the final value of 2.236067977 as the number of iterations (k) increases. And x_T to the 5.3 e-13 makes sense since it's close to 0.

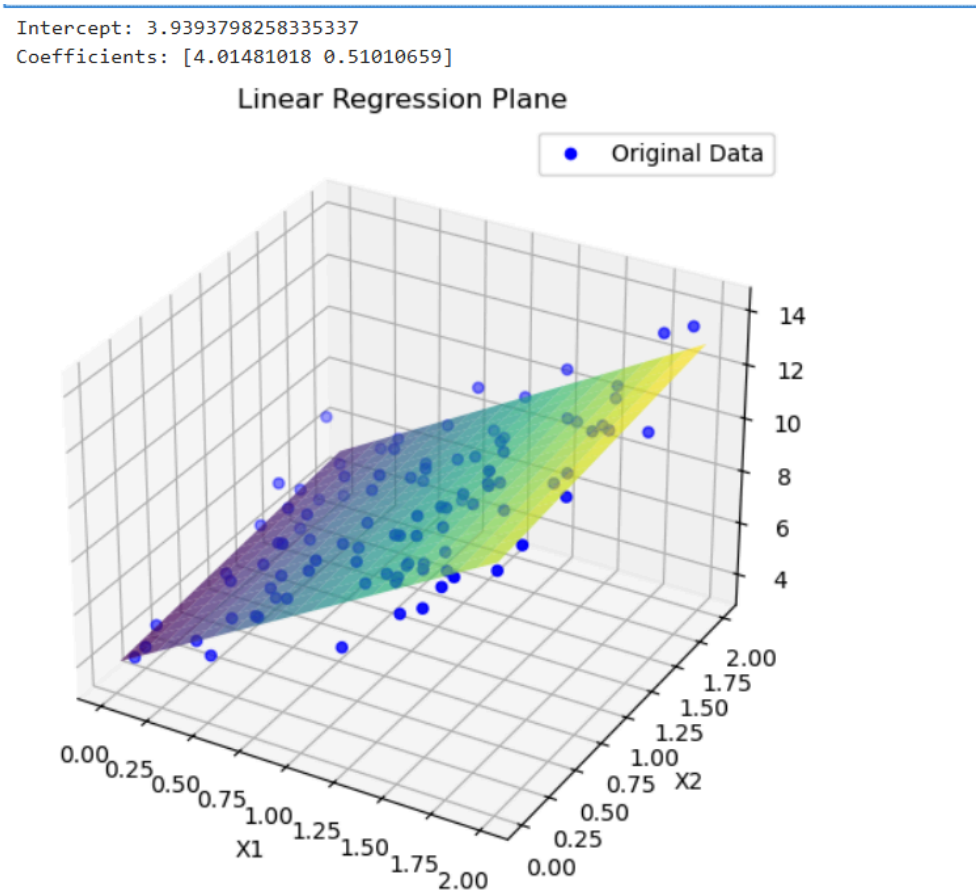
Now we make a graph with different step sizes and compare the results.



We can see that step 1 and 0.5 converge slowly but are more smooth. We got the difference = 0 results later than in step 5 and 3. The bigger the step size, the faster it gets to 0. The graph however is not as smooth as if you do step 1 or 0.5.

Part 2: Linear Regression with Numpy and Graphing with Matplotlib

The purpose of this part of the project is to create a graph of a linear regression. However, this time there are 2 x values instead of 1 x value. Thus the dimension of the graph needs to be 3D, involving the x,y, and z planes.

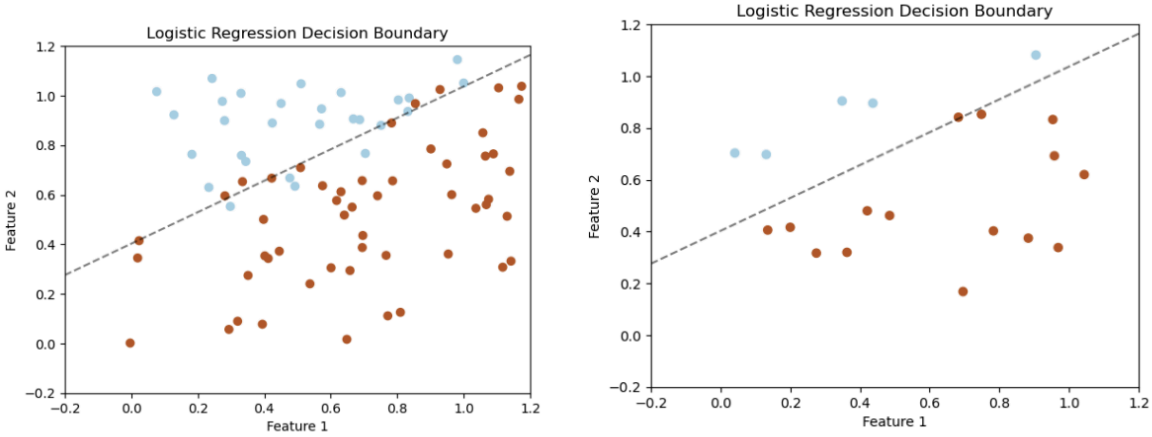


By generating random data points using my ID by setting the seed to 67, we are able to create our own unique data points. Here we can see the regression plane with our original data points as

well. I got my y-intercept: 3.9393798258335337 with my coefficients: [4.01481018, 0.51010659].

Part 3: Logistic Regression with Numpy and Graphing with Matplotlib

For the 3rd part, the purpose is to have a better understanding of the logistic regression model's behavior and its parameter optimization process. We tested by tuning the parameters "learning_rate" and "num_epochs". And we also commented on each of the lines to show what the code is doing. I tested using: learning_rate = 0.5, num_epochs = 600. I got Training Set Accuracy: 0.90 and Test Set Accuracy: 0.95.

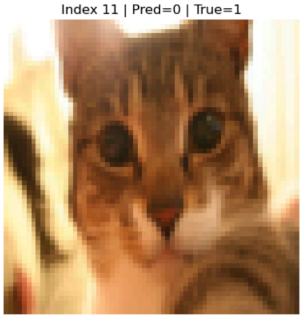


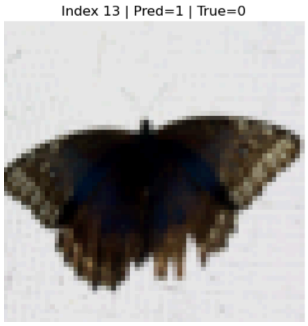
The left graph shows the training data vs testing data. We can see that it is accurate as most of the yellow falls into the yellow range and most of the blue falls into the blue range. Since the training data has more samples data, there will be some blue that falls below the contour line in the yellow zone.

Part 4: Image Binary Classification

This is a classic example of image binary classification. We are given 2 datasets, training and testing data. I used the deep learning model MLP neural network as my torch package could not be installed. The professor indicated it is okay to use other deep learning models. Using the original data first, we are able to see the accuracies of the training and testing data. "Training

Accuracy: 0.7847, Test Accuracy: 0.6000” with the indices of the incorrect predictions as “Misclassified Test Indices: [6 10 11 13 17 18 19 20 23 26 28 29 30 33 38 40 42 44 46 48]”. And the first 4 images that were incorrectly predicted are:





This shows the model predicted the cats as not a cat which is why the pred = 0, but the true is =1. Whereas the butterfly is supposed to be true = 0, however the model predicted pred as 1 thinking it is a cat.

Now we have to modify the data, using only the training data. This time there is no more testing data, but we are working with 167 training data. We have to split the training data into testing and training data. This means less sample size. Here we see the training data is more accurate now, specifically Training Accuracy: 0.9820, Test Accuracy: 0.5238. The testing accuracy got a little less accurate. And the first 4 misclassified images here are:

The explanation of these solutions is the same as I said about the original data. The purpose of this part of the project is to show how deep learning model is used to classify image binary, but however is not 100% accurate everytime.