

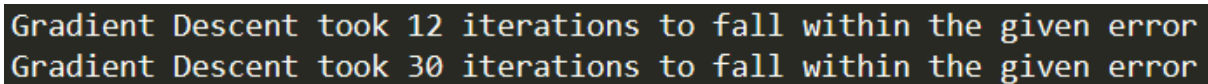
CS 416 Assignment 1

Task 1.

The first task is to perform Gradient Descent on a variety of functions. Function one is defined by a transformation by the 5D Hilbert matrix, further functions are in 2 Dimensions. Implementation uses the Torch library, allowing a backtrack post computation of the function, calculating the gradient between the two states. This gradient is then used to determine the subsequent step for the next iteration.

Each function is computed by its own definition, that takes the Torch Tensor x as an input, and returns $f(x)$, and the gradient between the transformation points. Each function is computed by a sequence of relevant Torch functions. Gradient Descent takes as input a starting step s ; α and β used as constant parameters for deciding the step in the next iteration; a starting vector x ; an error value after which the Gradient Descent terminates; and the function to be computed.

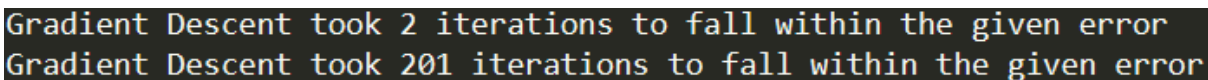
Task 1.1 asks for the number of iterations before termination over Function 1, given starting parameters. The two cases differ on values of α and β , the first where both are 0.5, and the second at 0.1. Naturally the second takes longer to converge, as the iterative steps are smaller. Results are shown in Figure 1.



```
Gradient Descent took 12 iterations to fall within the given error
Gradient Descent took 30 iterations to fall within the given error
```

Figure 1: Iterations in Task 1.1

Task 1.2 compares number of iterations to minimise two similar convex 2D functions. The difference is that the first has a coefficient of y of 2, and the second of $1/100$. Otherwise the Gradient Descent parameters are equivalent. Results show that the second function takes significantly longer to minimise via Gradient Descent; this is likely due to the plane being significantly shallower, and since we differentiate with respect to x , the coefficient of y has no effect on the step size. This results in it taking significantly longer for the step to become small enough that we converge on the minimum. Results are shown in Figure 2.



```
Gradient Descent took 2 iterations to fall within the given error
Gradient Descent took 201 iterations to fall within the given error
```

Figure 2: Iterations in Task 1.2

Task 2.

The goal for Task 2 was to use Gradient Descent to find a best estimator function, in this case a line, for classifying a 2D set of data. We use Torch functions to compute the average error when given a matrix X and its correct classifications. Gradient Descent is then implemented, with the goal of reducing this average error, by altering the parameters w_1 , w_2 and w_3 . As opposed to working towards a maximum error as in Task 1, we instead run the Gradient Descent for a specified number of iterations, as a control for balancing running time and required

accuracy. Additionally, the step size on each iteration is fixed, a larger step value results in faster convergence, but sacrifices a degree of accuracy, and gives significantly lower accuracy at fewer iterations. The default is 0.1, which gives smooth but slow convergence on our dataset. Finally, we rearrange the weight equation in order to plot lines for each step of gradient descent, to visualise the convergence. We plot each line according to $y = \frac{-w_1x - w_3}{w_2}$. The final vector after 20000 iterations, and the separation it represents, are shown below.

```
tensor([ 1.9824,  6.8053, -20.8522], requires_grad=True)
```

Figure 3: Final value of w

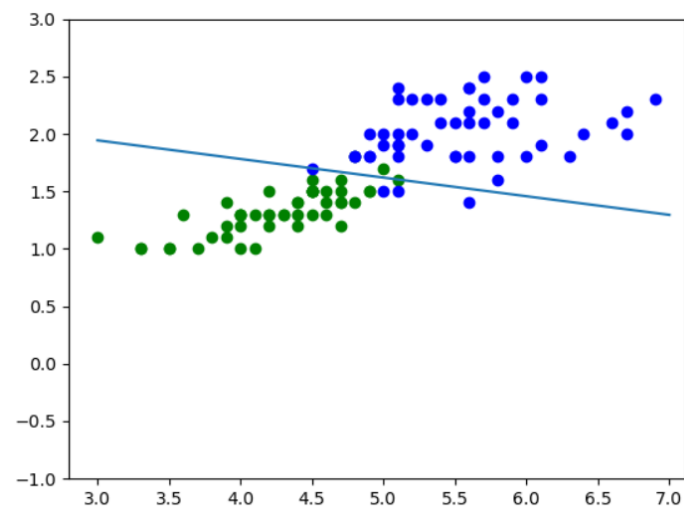


Figure 4: Final separator output