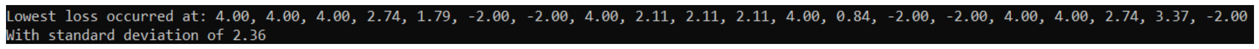
a.

Batch size: 1

python -m free_response.q3 1

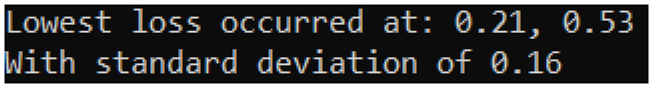


```
Lowest loss occurred at: 4.00, 4.00, 4.00, 2.74, 1.79, -2.00, -2.00, 4.00, 2.11, 2.11, 2.11, 4.00, 0.84, -2.00, -2.00, 4.00, 4.00, 2.74, 3.37, -2.00
With standard deviation of 2.36
```

Standard Deviation: 2.36

Batch Size:10

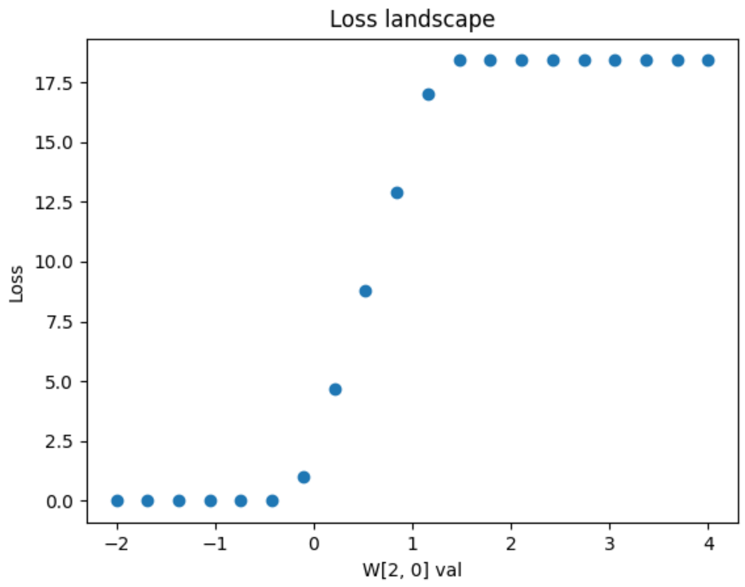
python -m free_response.q3 10



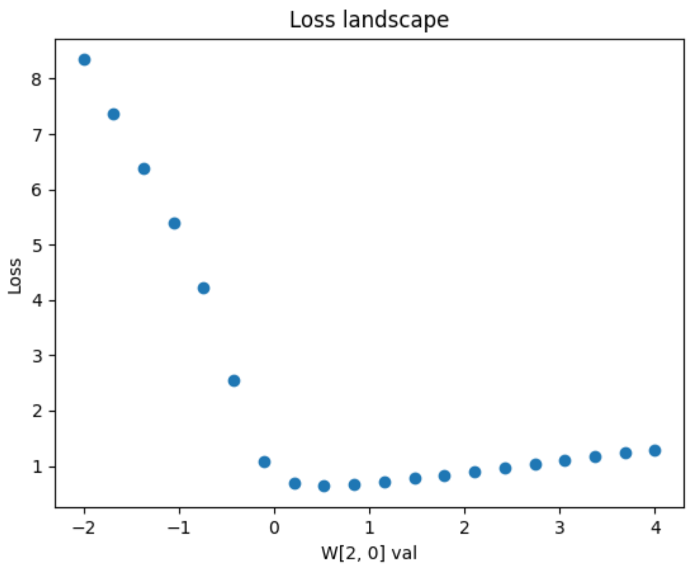
```
Lowest loss occurred at: 0.21, 0.53
With standard deviation of 0.16
```

Standard Deviation: 0.16

I chose these two batch sizes as they are extremes in terms of the amount of data used to compute the gradient, in order to obtain the stark difference in standard deviations between the batch sizes.



For a batch size of 1, each weight update is based on the gradient estimated from only one example at a time. Since each example can have a different gradient, the gradient estimates can be very different from each other, leading to high variability in the estimated gradients. Therefore, the results can be very noisy, and the loss values obtained after each epoch can vary widely, leading to a large standard deviation.



For a batch size of 10, each weight update is based on the gradient estimated from 10 examples at a time. Since we are averaging the gradients from 10 examples, the resulting gradient estimate is more stable and less noisy compared to the case of a batch size of 1. Thus, the updates are less noisy, and the loss values obtained after each epoch are more consistent, leading to a smaller standard deviation.

b.

It is seen that the batch size can have a significant impact on the optimization process. When using batch gradient descent, the entire dataset is used to compute the gradients at each step, this can be slow but provides a more accurate estimate of the gradients. Using a batch size of 1 corresponds to stochastic gradient descent, where only a single example is used to compute the gradients. Using a larger batch size corresponds to mini-batch gradient descent, where the gradients are computed using a subset of the dataset. A larger batch size generally leads to smoother and less noisy loss landscapes, as seen in the lower standard deviations for batch sizes larger than 1. The larger batch sizes can result in slower convergence due to the increased computational cost per iteration. Additionally, using smaller batch sizes can help the optimization algorithm escape from local minima and find better global optima. Thus, there is a trade-off between the speed and stability of the optimization process when selecting a batch size for gradient descent.