May Zhang
A17452487

In this project, we are trying to find the parameters w that best fit the (x,y) data pairs with the function f_w. We did this by first transforming the nonlinear function into a linear expression using the Jacobian matrix, and using least squares to determine the parameters that minimize the sum of squared errors.

tanh function

Derivative calculations

To transform the nonlinear function into linear expressions, we need to calculate the Jacobian matrix, which is the gradient of the function with respect to parameters w.

dfw/dw1 = ϕ(w2x1 + w3x2 + w4x3 + w5)
dfw/dw2 = w1 * ϕ'(w2x1 + w3x2 + w4x3 + w5) * x1
dfw/dw3 = w1 * ϕ'(w2x1 + w3x2 + w4x3 + w5) * x2
dfw/dw4 = w1 * ϕ'(w2x1 + w3x2 + w4x3 + w5) * x3
dfw/dw5 = w1 * ϕ'(w2x1 + w3x2 + w4x3 + w5)
dfw/dw6 = ϕ(w7x1 + w8x2 + w9x3 + w10)
dfw/dw7 = w6 * ϕ'(w7x1 + w8x2 + w9x3 + w10) * x1
dfw/dw8 = w6 * ϕ'(w7x1 + w8x2 + w9x3 + w10) * x2
dfw/dw9 = w6 * ϕ'(w7x1 + w8x2 + w9x3 + w10) * x3
dfw/dw10 = w6 * ϕ'(w7x1 + w8x2 + w9x3 + w10)
dfw/dw11 = ϕ(w12x1 + w13x2 + w14x3 + w15)
dfw/dw12 = w11 * ϕ'(w12x1 + w13x2 + w14x3 + w15) * x1
dfw/dw13 = w11 * ϕ'(w12x1 + w13x2 + w14x3 + w15) * x2
dfw/dw14 = w11 * ϕ'(w12x1 + w13x2 + w14x3 + w15) * x3
dfw/dw15 = w11 * ϕ'(w12x1 + w13x2 + w14x3 + w15)
dfw/dw16 = 1

The derivative matrix is Dr(w) = 2 * Dr(w).T @ r(w) = Dr(w) = 2 * Df(w).T @ r(w).

Stopping Criterion

The algorithm can be stopped when a minimal residual or smaller optimality condition residual is achieved. We observed consistent training loss after approximately 5-6 iterations when implementing the tanh function to define f_w. This consistency was confirmed by graphing iterations against training loss.

The curve below demonstrates a sharp decrease in training loss at the beginning of the training and gradually goes stable as the iterations go on. Determining the interpretation of "small" is challenging, given the training loss can range from 0 to infinity, unlike the error rate that is confined between 0-100%. Therefore, it's more effective to establish the stop criterion once the training loss no longer declines. We empirically did the stopping criterion by checking the magnitude of the difference between the consecutive parameters W. When the magnitude of difference is below the threshold, the training loss is barely declining. So we are using the small residual criterion to reach the point that we almost solved the equations for r_w. This further suggests that the training loss function has reached a local minimum at this stage. As a result, trying new initializers would be helpful to possibly approach different local minima.

I further checked the optimality condition residual and verified that the magnitude of the formula provided in the textbook is actually small enough.

First, I initialized lambda at the value of 10^-5, and tried different initialization W. The error metric I chose is mean squared error (MSE). The error metric for random initialization varies each time I run the code, but in general it seems to work better than setting all numbers to a single value. By testing with various initialized values, starting with uniformly random values from [-1,1] seems to have a relatively low MSE (2.659e-7).

Parameter W	Uniform random [-1,1]	Uniform random [-2,2]	[1] * 16	Standard normal distribution	[2] * 16
MSE	2.659e-7	0.027	0.101	0.012	0.030

Table1. The mean squared error of the training data when starts at different initialization values.

Training loss and different initializations

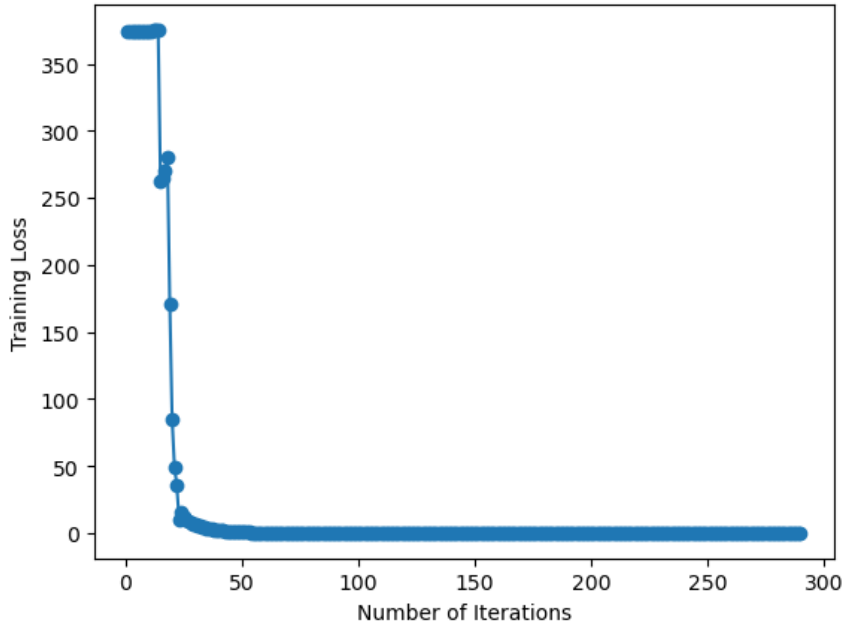

Figure1. Training loss v. Number of Iterations with f_w defined by tanh function, W initializes with uniformly random [-1,1], and lambda initializes at 10^-5.

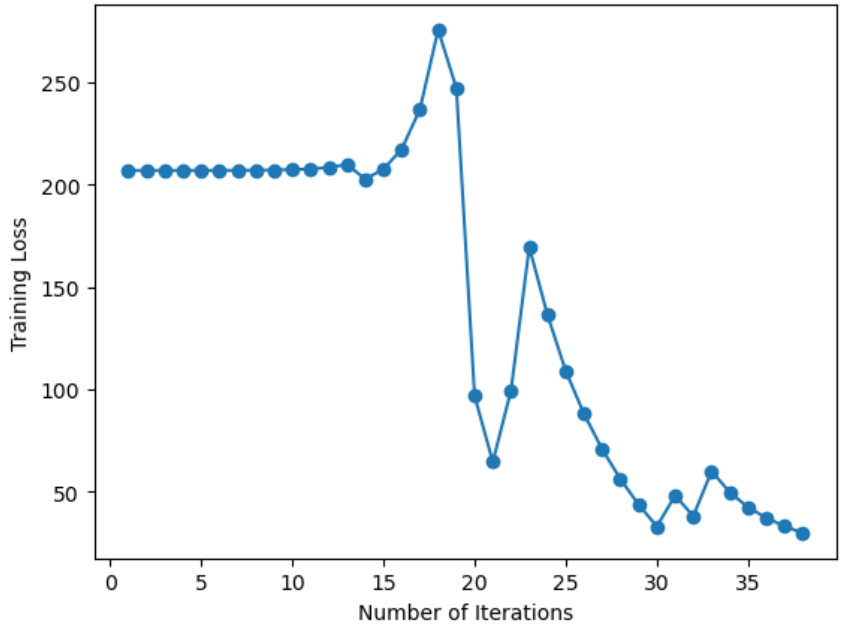

Figure2. Training loss v. Number of Iterations with f_w defined by tanh function, W initializes with uniformly random [-2,2], and lambda initializes at 10^-5.

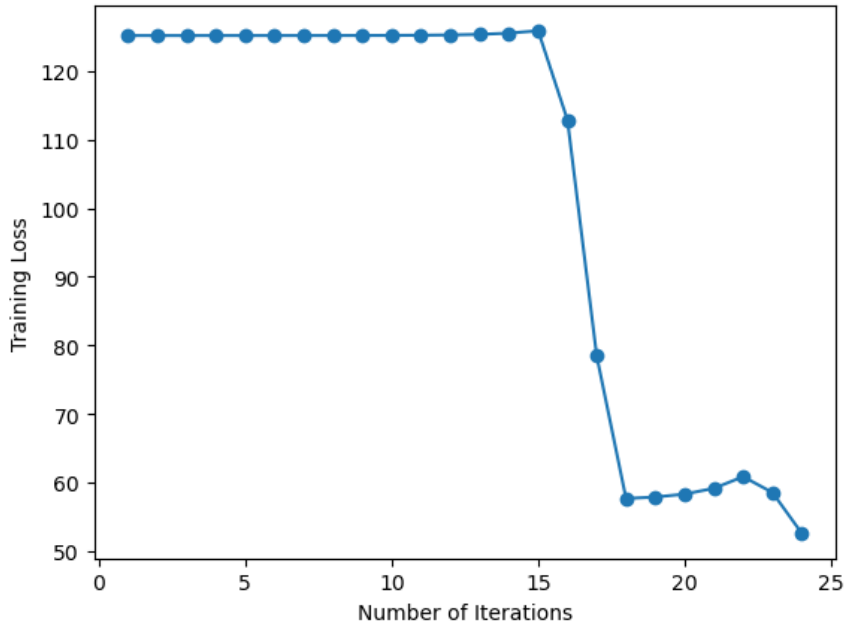

Figure3. Training loss v. Number of Iterations with f_w defined by tanh function, W initializes with [1] * 16, and lambda initializes at 10^-5.

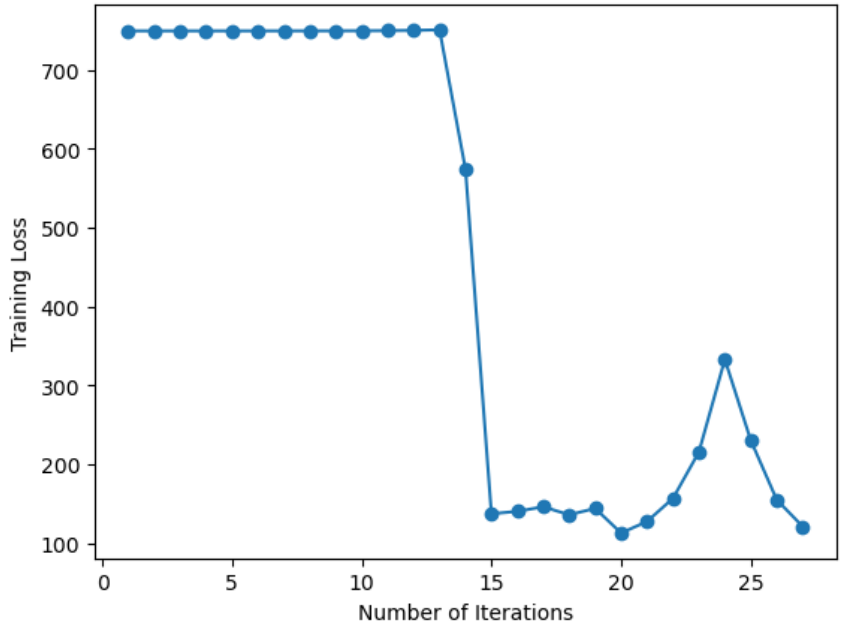

Figure4. Training loss v. Number of Iterations with f_w defined by tanh function, W initializes with standard normal distribution, and lambda initializes at 10^-5.

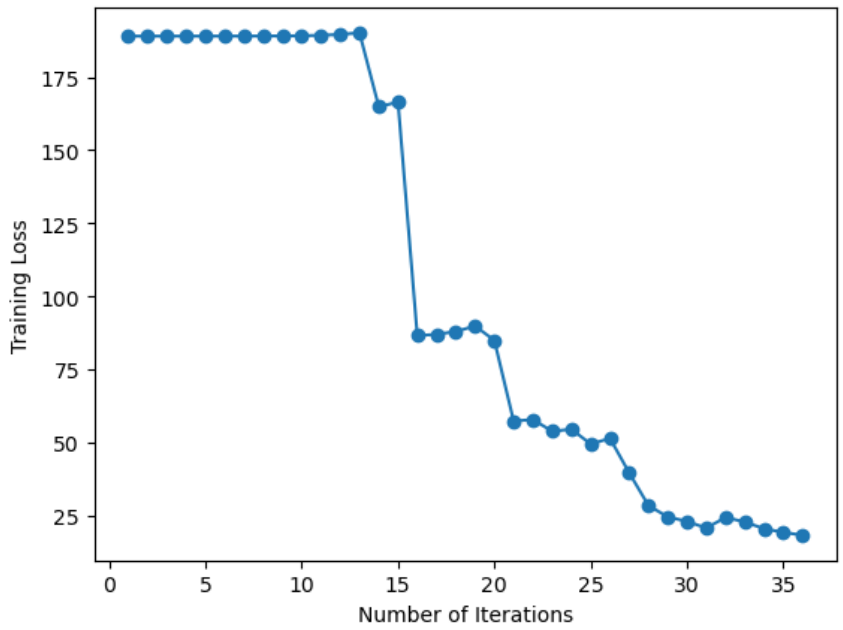

Figure5. Training loss v. Number of Iterations with f_w defined by tanh function, W initializes with [2] * 16, and lambda initializes at 10^-5.

Setting the initialized W at different points results in a different pattern of training loss curve and a different final W. In general, they have the pattern of a short curve that barely has a change in training loss, a curve or slope that sharply decreases in training loss, and a relatively stable curve that barely declines in training loss. Except for the uniformly random initialized point, it seems that others haven't really reached the stabilized point yet as they met the standard of that the decline between points is smaller than the given threshold.

The difference in the pattern of the curve suggests different paths were taken for each W to eventually reach its local minimum. In some of the initialization, the values of the final W are similar while in the other it seems like they are not going to the same minimizer W. This suggests that they might converge to different local minima. In conclusion, by choosing different initialized weights, we might reach a different value of weights in the end that minimize the value of r_w by running the Levenberg-Marquardt algorithm.

After setting the parameter initialization at [2] * 16 (since it has a relatively low MSE and the value is fixed in each trial unlike the random values), I varied the initialized value of lambda.

Training loss and different lambda values

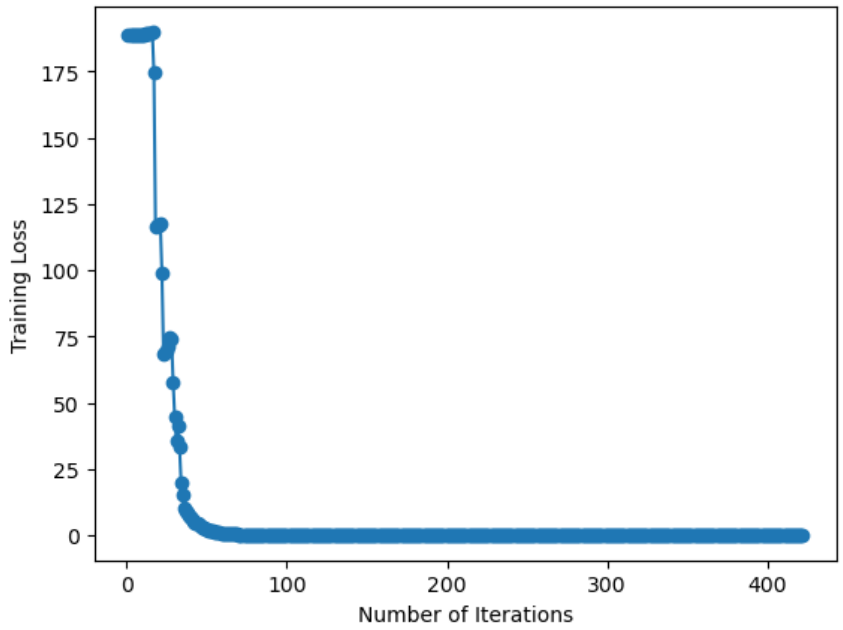

Figure6. Training loss v. Number of Iterations with f_w defined by tanh function, W initializes with [2] * 16, and lambda initializes at 10^-6.

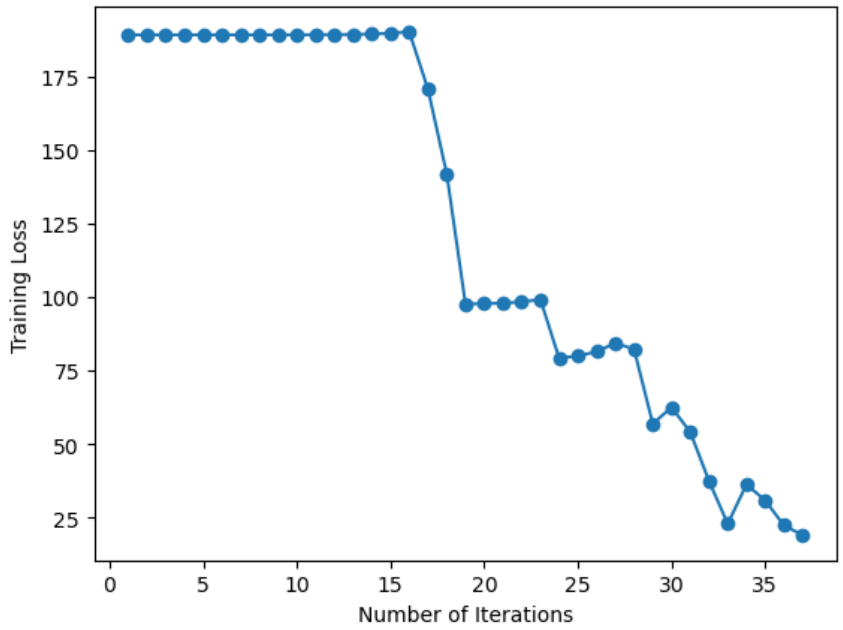

Figure7. Training loss v. Number of Iterations with f_w defined by tanh function, W initializes with [2] * 16, and lambda initializes at 1.1*10^-6.

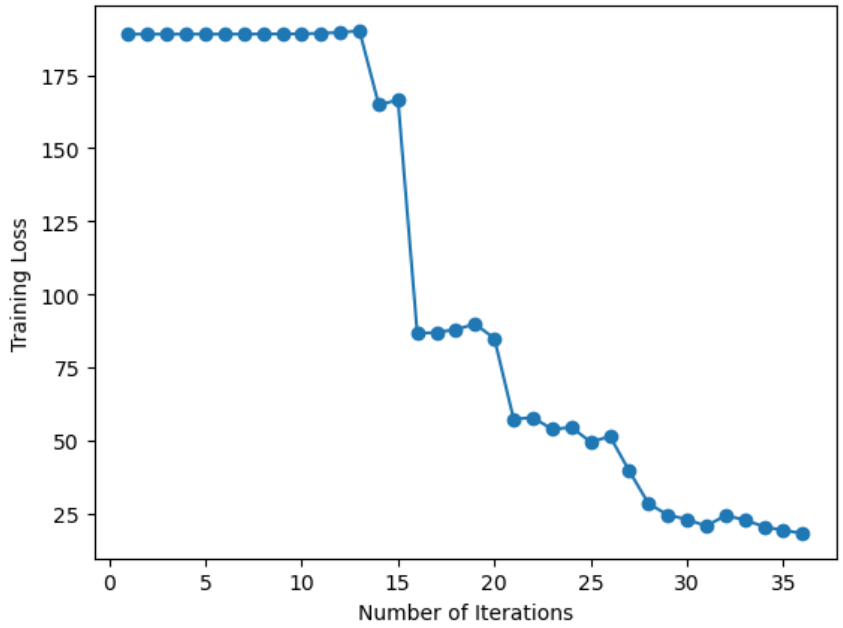

Figure8. Training loss v. Number of Iterations with f_w defined by tanh function, W initializes with [2] * 16, and lambda initializes at 10^-5. (Same as the figure above but now we are looking at different lambda values).

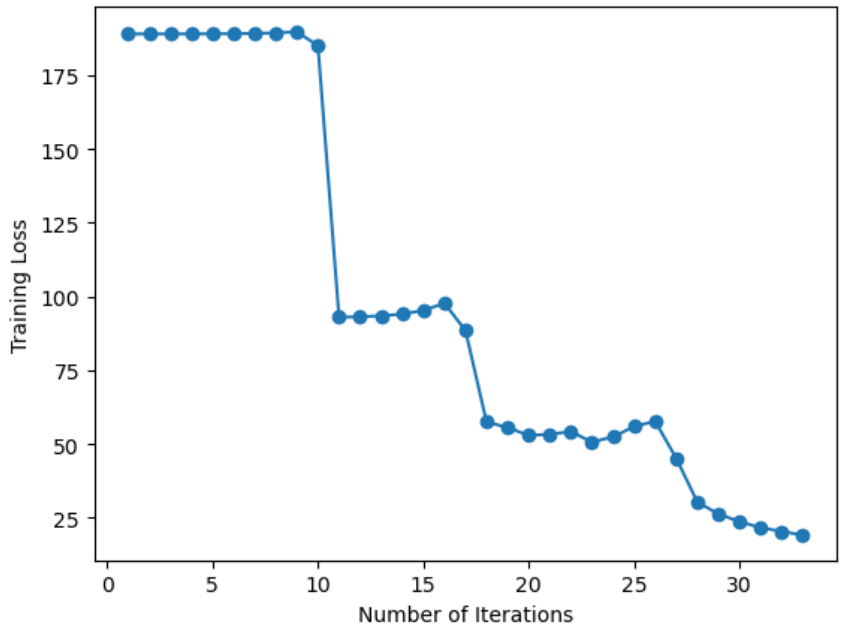

Figure9. Training loss v. Number of Iterations with f_w defined by tanh function, W initializes with [2] * 16, and lambda initializes at 10^-4.

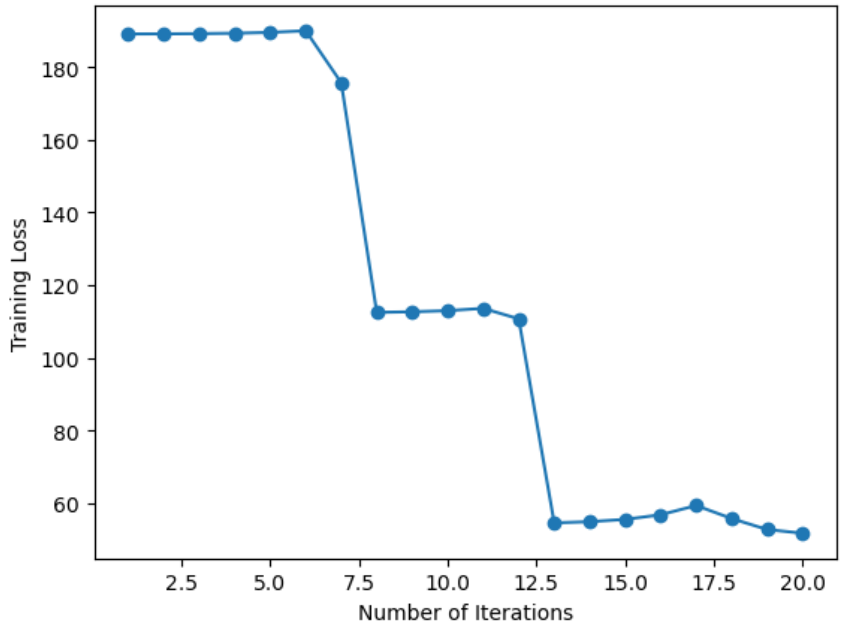

Figure10. Training loss v. Number of Iterations with f_w defined by tanh function, W initializes with [2] * 16, and lambda initializes at 10^-3.

Lambda	***10^-6***	***1.1*10^-6***	***10^-5***	***10^-4***	***10^-3***
MSE	***2.517e-7***	***0.0022***	***0.0299***	***0.1007***	***0.1007***

Table2. The mean squared error of the training data with different lambda values.

By setting the lambda initialization value to 10^-6, we immediately have a sharp decrease in training loss after a few training iterations, and reach a significantly lower training loss value.

In general, the trend is that with a smaller lambda value, the mean square error gets smaller, the training curve is smoother with a more curvy decline rather than stair-like sharp decrease, and it generally needs more steps to reach the final value of weights and a longer time to run the code. This is reasonable that with a small lambda value, we tend to take baby-step in each step so a longer time is needed but it's also more likely to reach the minima more accurately instead of overpassing the minima with a big step.

Testing the neural network with different gamma

We will use the weights derived from training when initialized W is [2] * 16 and the lambda value is 10^-5, which has a MSE of 0.003.

Gamma	1	2	3	4	5
MSE	0.005	0.719	7.203	21.644	50.146

Table3. The mean squared error of the test data with different gamma values.

I tested the gamma value from 1 to 5. In general, with a larger gamma value, the mean squared error is also larger. When gamma = 1, the MSE is only slightly higher than the MSE in the training data, suggesting that the model generalized well to the test data. It is reasonable that the model is better at predicting when the range of the test data is within [-1,1] since we train the data generated uniformly at random within the range of [-1,1]. When the gamma value gets larger, predicting the value of the data involves extrapolation so there is a higher error rate when predicting these values. The error rate increases drastically since the product of x1 and x2 amplifies the error when the data gets large.

Another non-linear function

The nonlinear function I chose is x1^2 + x2^2 + x3^2 since I would like to see if the square of an element would make any difference to the effect of the training. To implement it, I simply have to generate a new set of training data, and add the parameters X_train, Y_train to the Levenberg-Marquardt algorithm so that a new training set can be put into the model.

Training the neural network with different initialized weights (new nonlinear function)

Parameter W	Uniform random [-1,1]	Uniform random [-2,2]	[1] * 16	Standard normal distribution	[2] * 16
MSE	0.085	0.090	0.090	0.0890	0.090

Table4. The mean squared error of the training data starts at different initialization values with the new function.

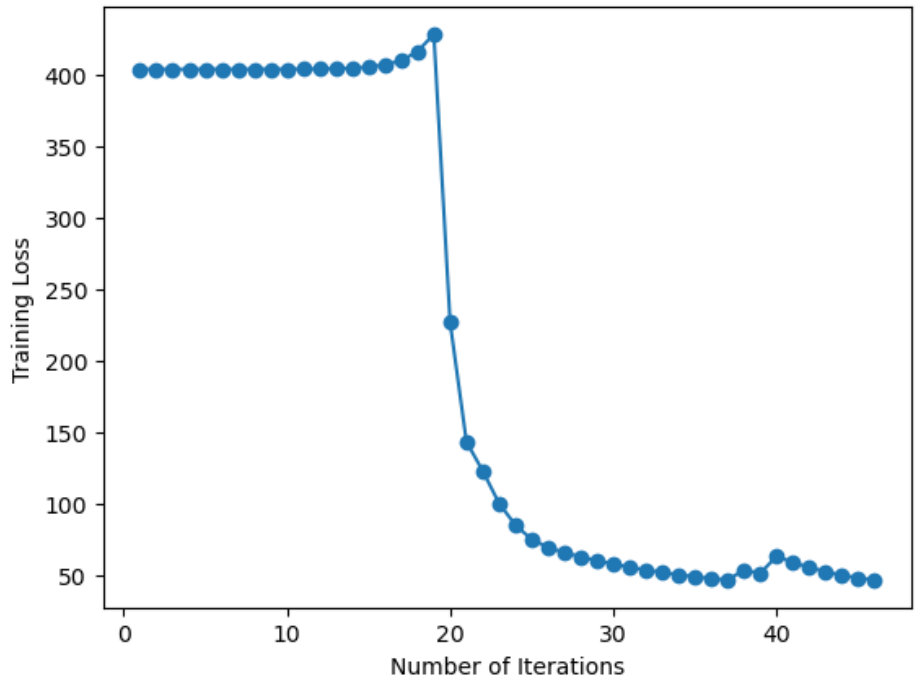

Figure11. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-5.

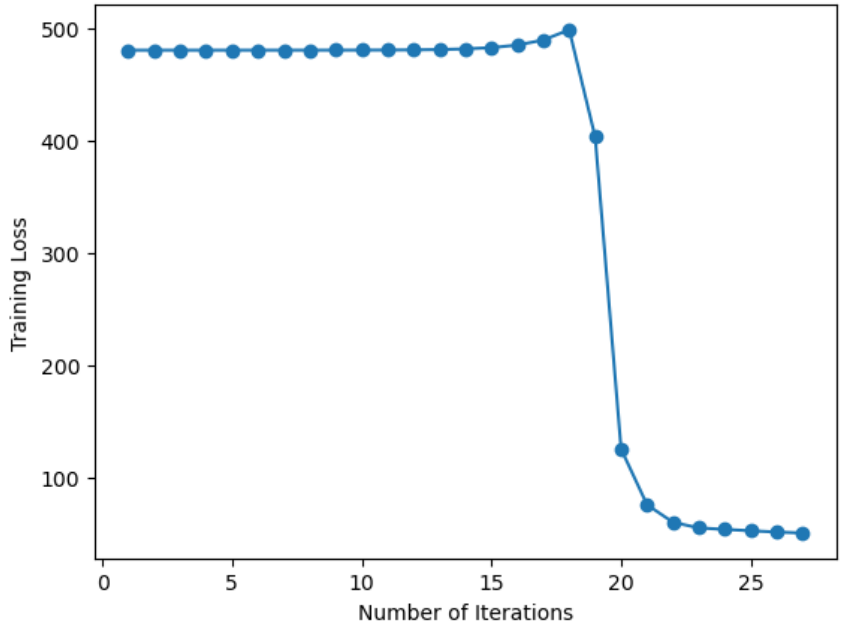

Figure12. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-2,2], and lambda initializes at 10^-5.

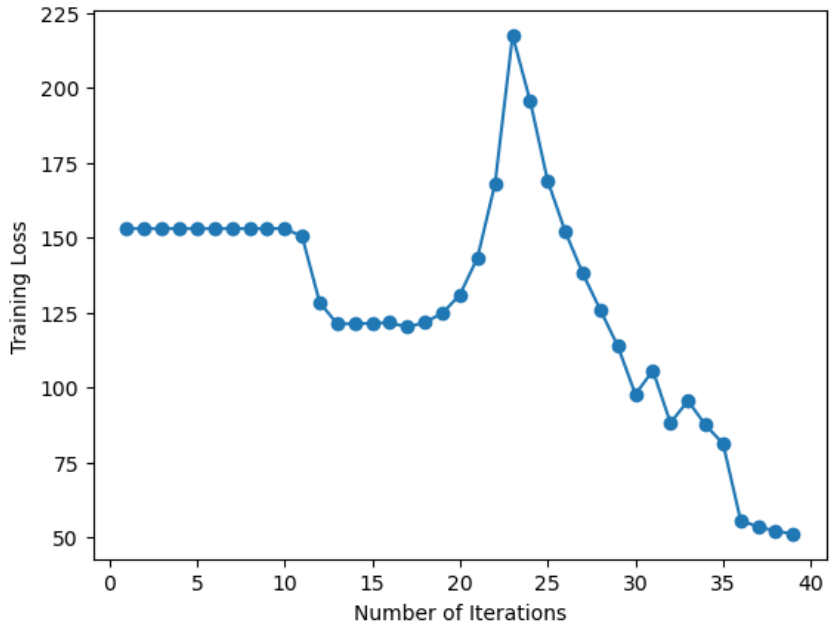

Figure13. Training loss v. Number of Iterations with f_w, W initializes with [1] *16, and lambda initializes at 10^-5.

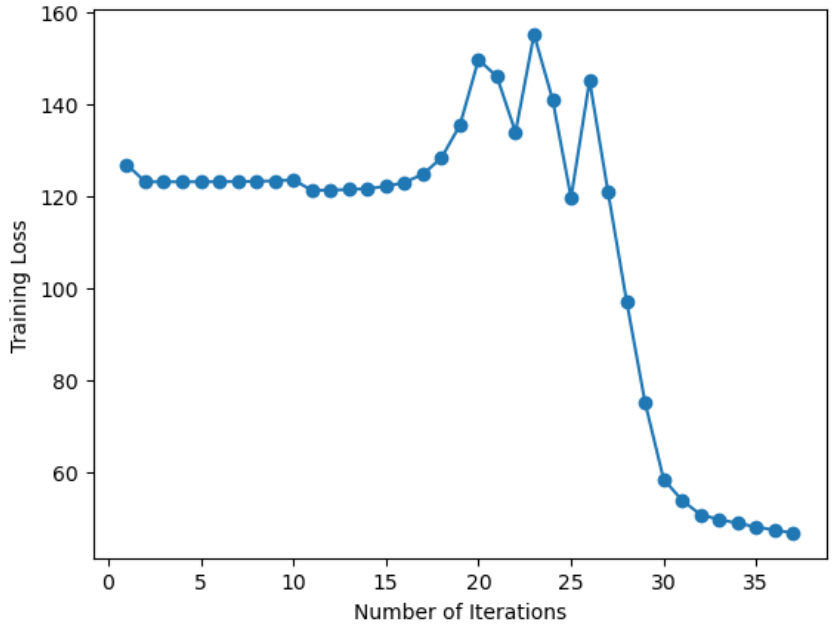

Figure14. Training loss v. Number of Iterations with f_w, W initializes with [2] *16, and lambda initializes at 10^-5.

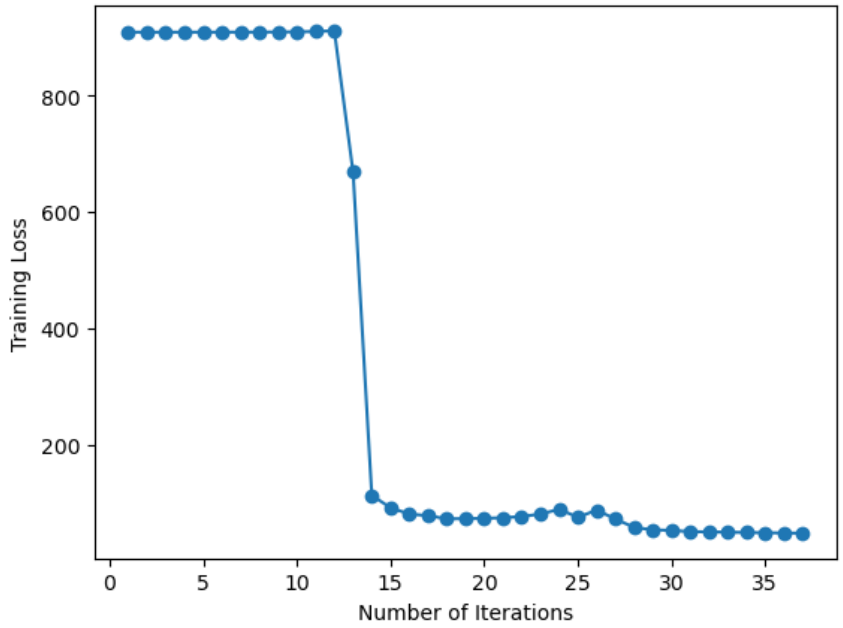

Figure15. Training loss v. Number of Iterations with f_w, W initializes with standard normalized distribution of weights, and lambda initializes at 10^-5.

In general, the mean square error for this function is fairly consistent with different initialization values. The training process works the best with the uniformly distributed initialization in [-1,1] with a smoothly declining training curve and a relatively low MSE. W initialized with random data seem to have a more smooth decline curve while there might be more fluctuation when the elements are all set at one value. With different initialization W, we will reach different local minima. An example is shown below:

Final parameter W with an initialization of uniformly random [-1,1]:
[3.03057816 0.36707326 -0.02826774 -0.74183031 -0.86577897 2.56363363
 -0.97459889 -0.01257275 0.06025727 -0.85901154 2.18585599 0.56981983
  0.02143498 0.83350798 -0.71243634 5.52356667]

Final parameter W with an initialization of uniformly random [-2,2]:
[-2.03107922e+00 -4.83758138e-01 -1.04910375e+00 -1.64402434e-02
  9.16668925e-01 -2.52827797e+00 -5.07716832e-01 6.73867499e-01
  2.35914170e-02 6.69069704e-01 2.24044610e+00 -1.04099327e+00
  2.93294472e-02 -2.40506730e-03 -7.86000410e-01 4.72727899e+00]

Training the neural network with different lambda (new nonlinear function)

Next, we will fix the initialized weights at the uniformly distributed random data from [-1,1] range, and see how different lambda values affect the training.

Lambda	***10^-5***	***10^-4***	***10^-3***	***10^-2***	***1***
MSE	***0.085***	***0.085***	***0.085***	***0.085***	***0.085***

Table5. The mean squared error of the training data with different lambda values.

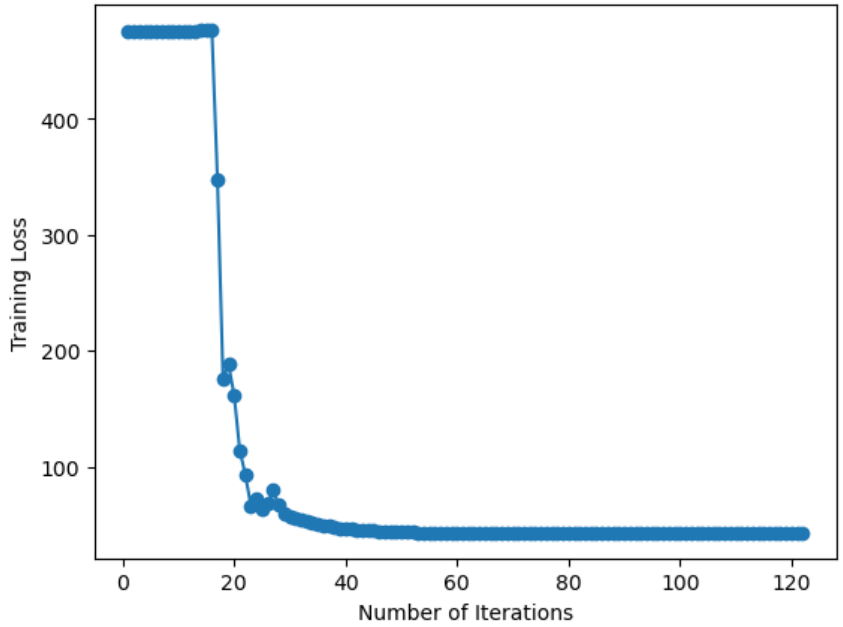

Figure16. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-5.

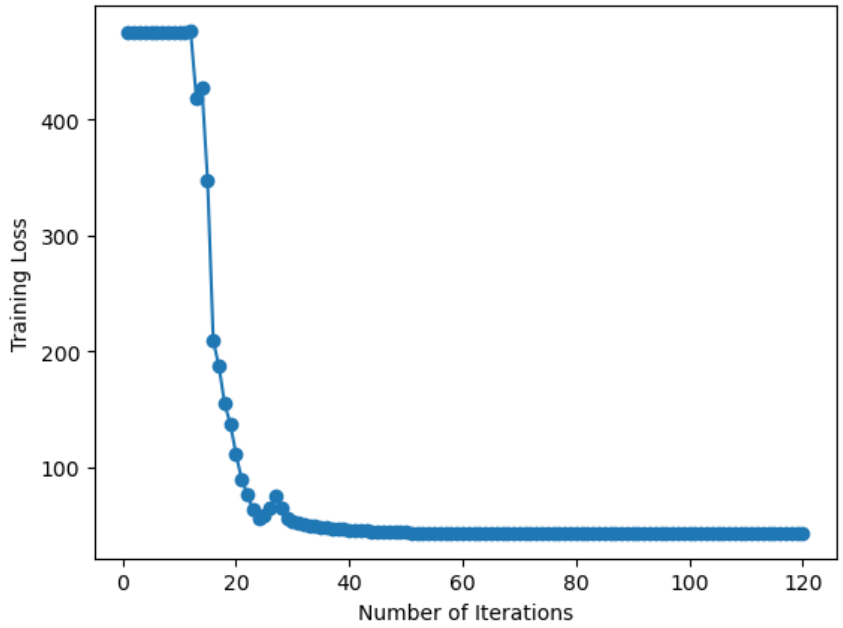

Figure17. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-4.

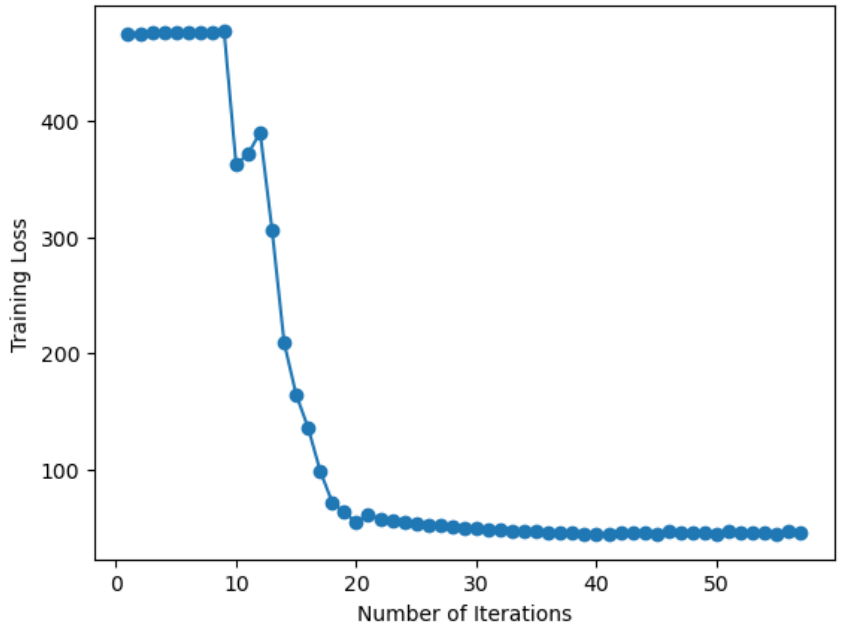

Figure18. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-3.

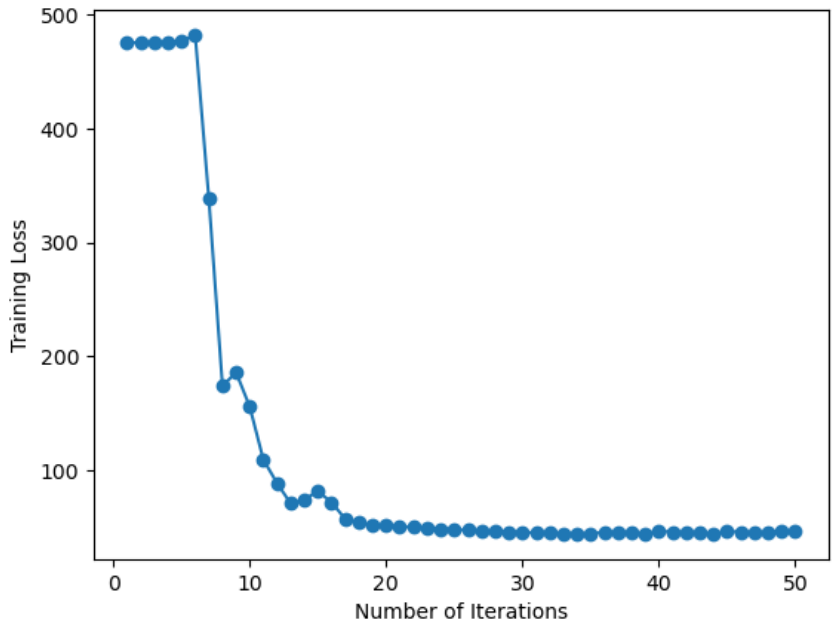

Figure19. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-2.

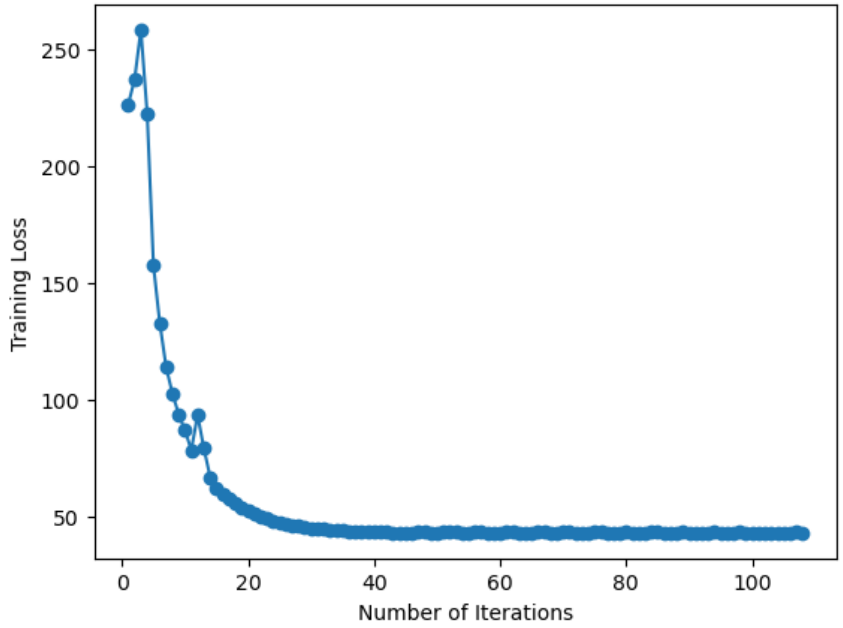

Figure20. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 1.

For all the training curves, they show a trend of a short stable line that has little changes in training loss, and then a sharp but curvy decrease in training loss, and at last a stable curve where there is little decline in the training loss. By changing the lambda value, the

error rate by MSE doesn't change a lot, but the number of steps to reach the final value might vary based on different lambda values. However, a larger lambda value doesn't necessarily imply a shorter number of iterations.

Testing the neural network with different gamma (new nonlinear function)

For the variation of the value of gamma, we will use the model trained with an initialization of the uniformly distributed random data and lambda set at 10^-5. The training error calculated by MSE is 0.089.

Gamma	1	2	3	4	0.5
MSE	0.092	2.680	13.574	62.352	0.066

Table6. The mean squared error of the test data with different gamma values.

I generated the test data with different gamma and found that the error rate of the test data with gamma = 1 is fairly low and close to the error rate of the training data. This is reasonable since we train the data generated uniformly at random within the range of [-1,1]. This suggests that the model generalized well to the test data within the same range. However, as the gamma increases, the MSE increases dramatically. This is reasonable as the model might not be able to extrapolate the nonlinear function well, especially considering it's the sum of the squares of each element and the actual function would increase a lot more as the value increases. However, when gamma = 0.5, the error rate of the test data could be even lower than the training data. In general, the model can well generalize to the test data with the range of [-1,1], but as gamma increases, the error rate would increase drastically.

Noisy training data

In general, after the noise is added to the data, the error calculated by the mean squared error (MSE) goes much higher compared to the data without noises.

Training the neural network with different initialized weights with epsilon =1

Parameter W	Uniform random [-1,1]	Uniform random [-2,2]	[1] * 16	Standard normal distribution	[2] * 16
MSE	0.306	0.306	0.472	0.304	0.360

Table7. The mean squared error of the training data starts at different initialization values with the new function.

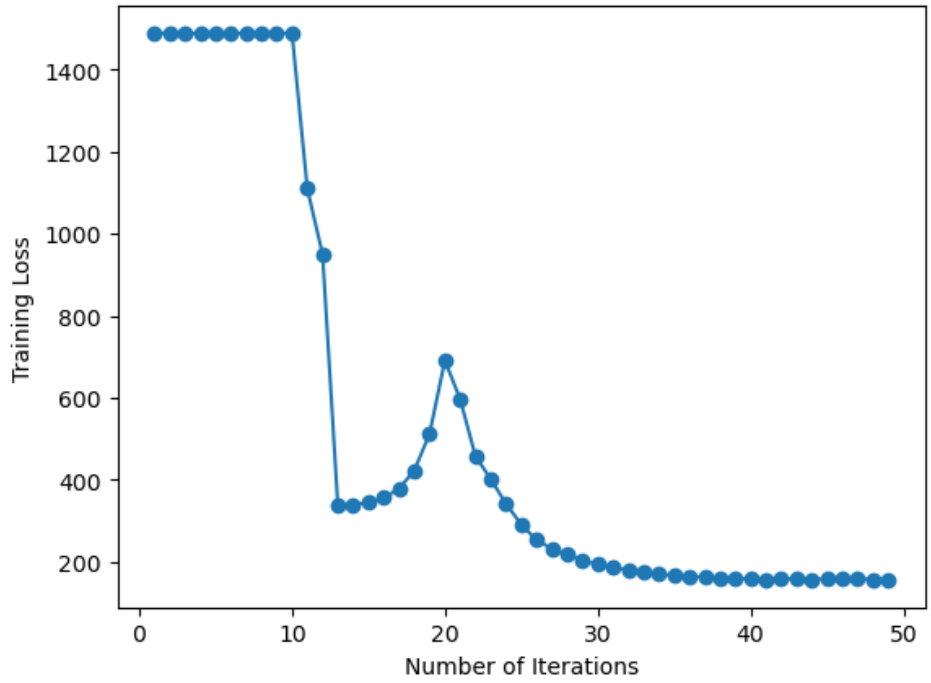

Figure21. Training loss v. Number of Iterations with noisy data, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-5.

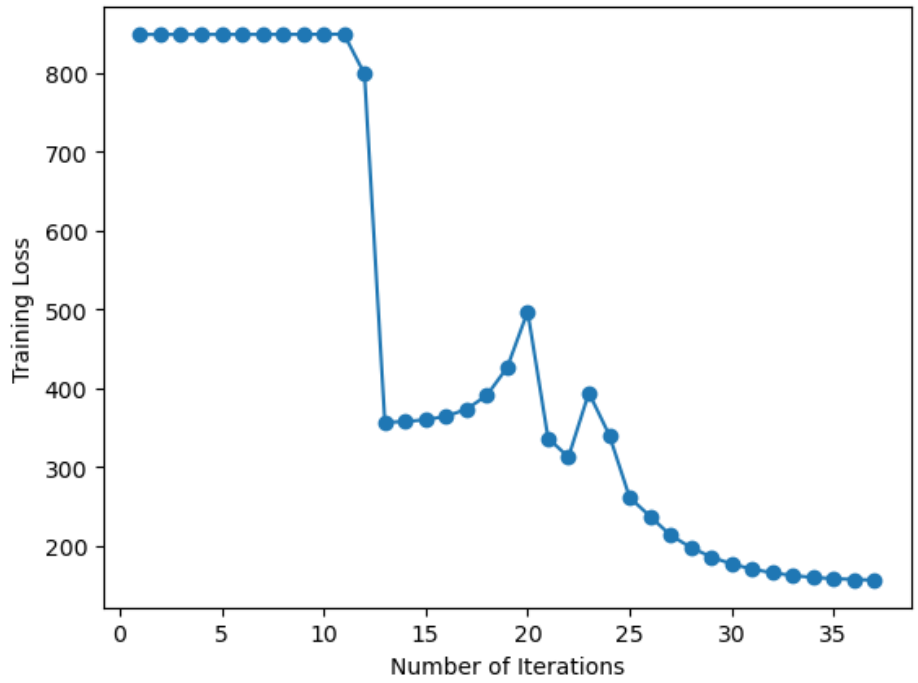

Figure22. Training loss v. Number of Iterations with noisy data, W initializes with uniformly random value from [-2,2], and lambda initializes at 10^-5.

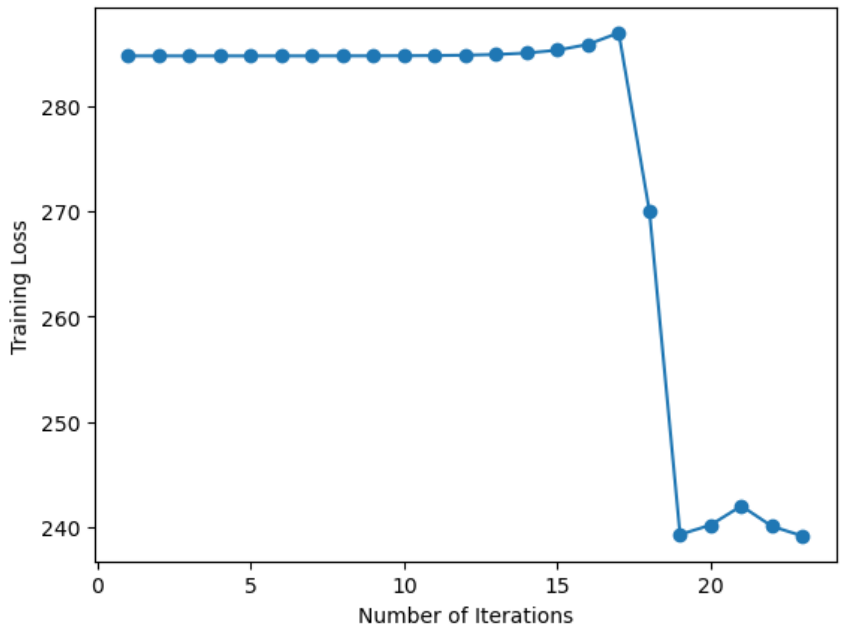

Figure23. Training loss v. Number of Iterations with noisy data, W initializes with [1] * 16, and lambda initializes at 10^-5.

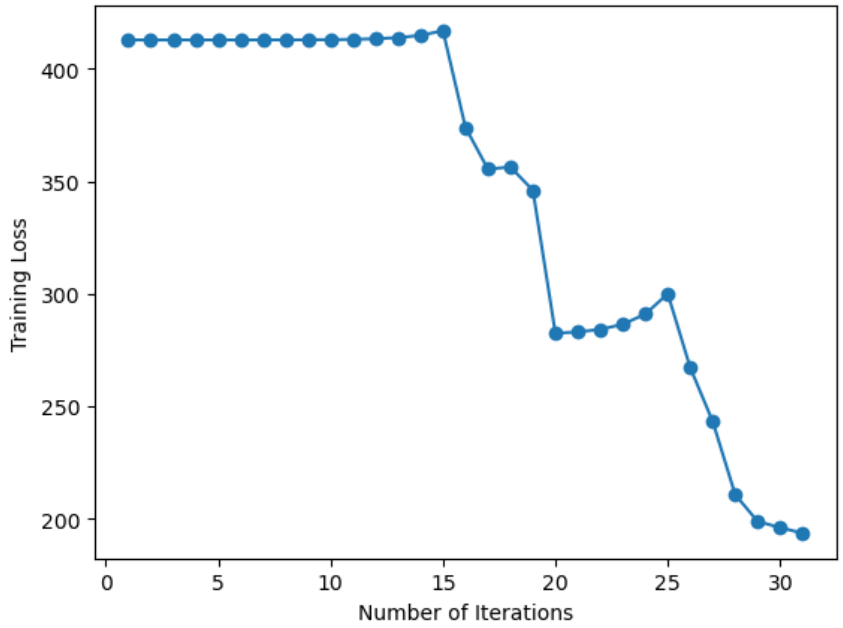

Figure24. Training loss v. Number of Iterations with noisy data, W initializes with [2] * 16, and lambda initializes at 10^-5.

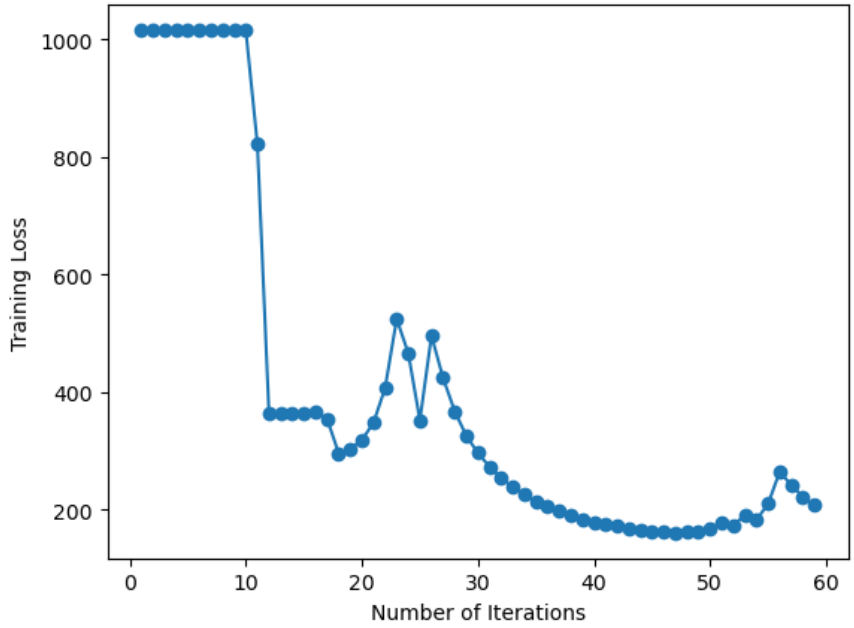

Figure25. Training loss v. Number of Iterations with noisy data, W initializes with standard normalized distribution, and lambda initializes at 10^-5.

Training the neural network with different lambda with epsilon =1

Lambda	***10^-5***	***10^-4***	***10^-3***	***10^-2***	***0.1***
MSE	***0.366***	***0.321***	***0.350***	***0.348***	***0.321***

Table8. The mean squared error of the training data with different lambda values.

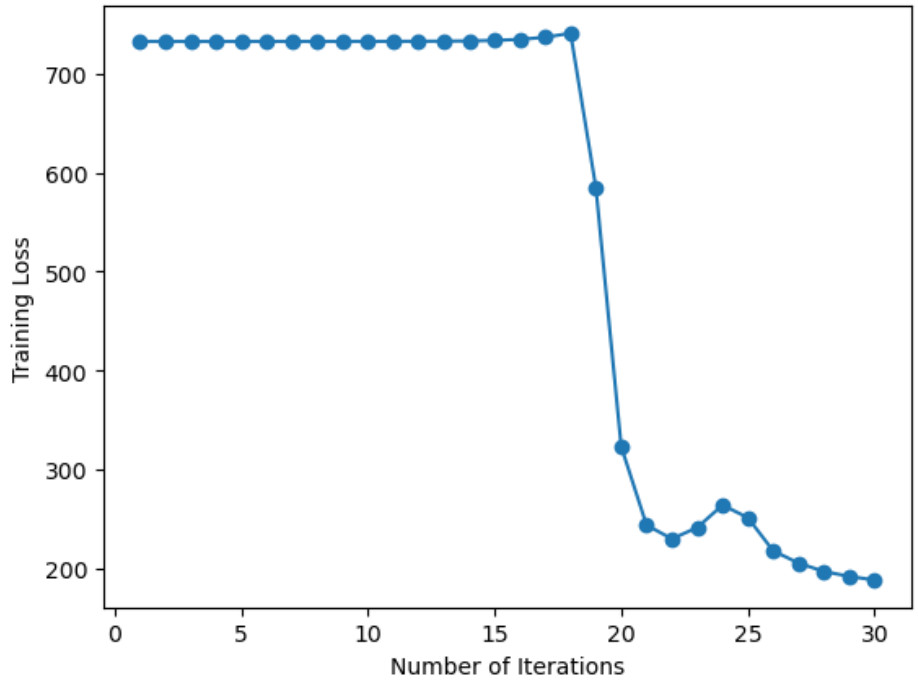

Figure26. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-5.

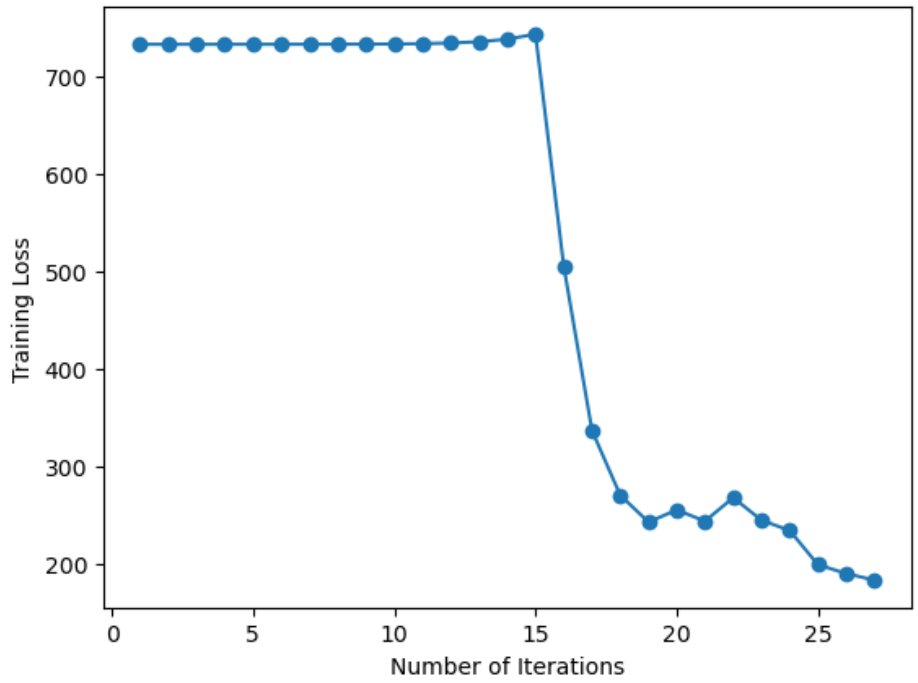

Figure27. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-4.

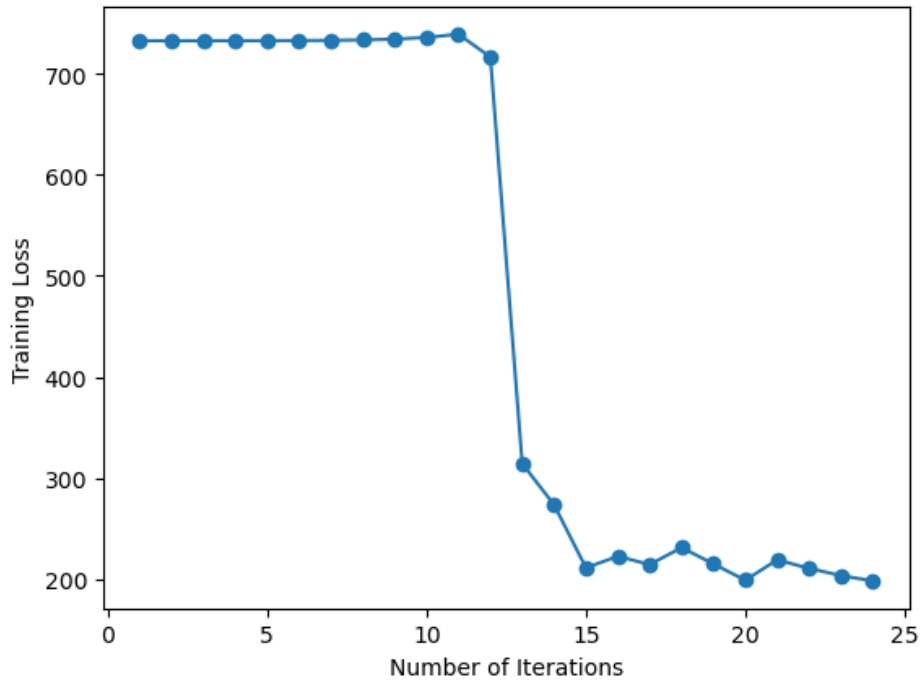

Figure28. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-3.

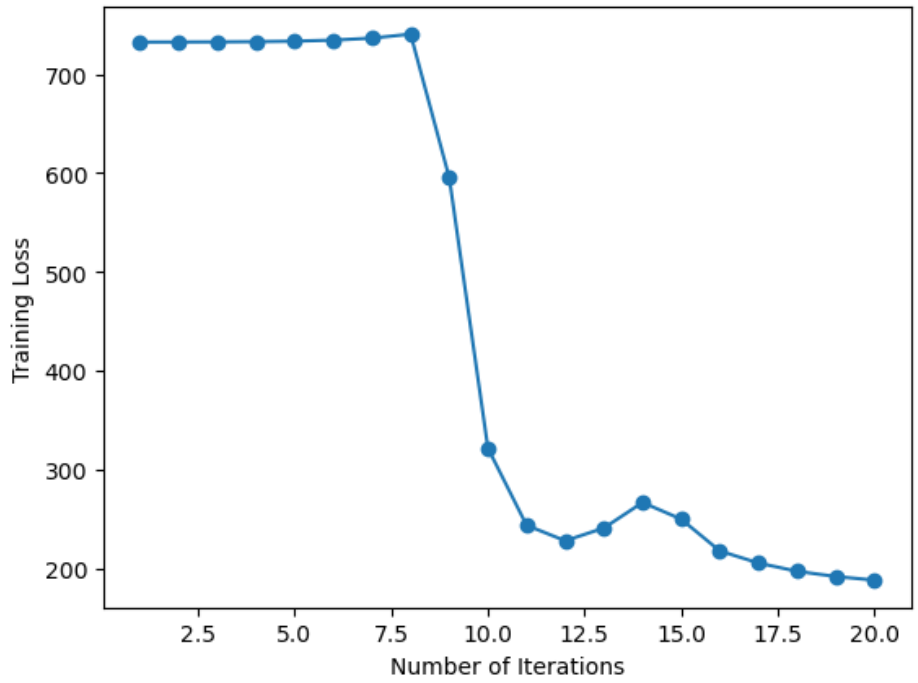

Figure29. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-2.

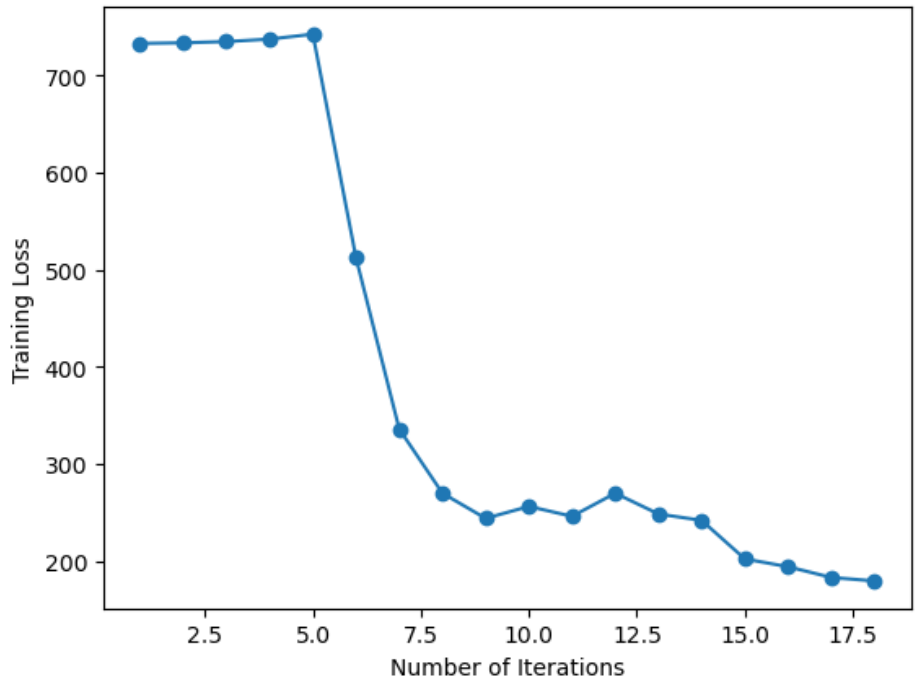

Figure30. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 0.1.

The data generally have a relatively smooth declining curve at different values of lambda with a bit fluctuation. There aren't many changes in the MSE with the change of lambda values. Generally less steps are needed with a larger lambda value.

Testing the neural network with different gamma with epsilon =1

We will use the weights derived from training when initialized W is uniformly random data from [-1,1] and the lambda value is 10^-5, which has a MSE of 0.348.

Gamma	1	2	3	4	5
MSE	0.432	1.276	9.581	25.243	48.710

Table9. The mean squared error of the test data with different gamma values.

Training the neural network with different initialized weights with epsilon =2

Parameter W	Uniform random [-1,1]	Uniform random [-2,2]	[1] * 16	Standard normal distribution	[2] * 16
MSE	1.291	1.313	1.427	1.321	1.296

Table10. The mean squared error of the training data starts at different initialization values with the new function.

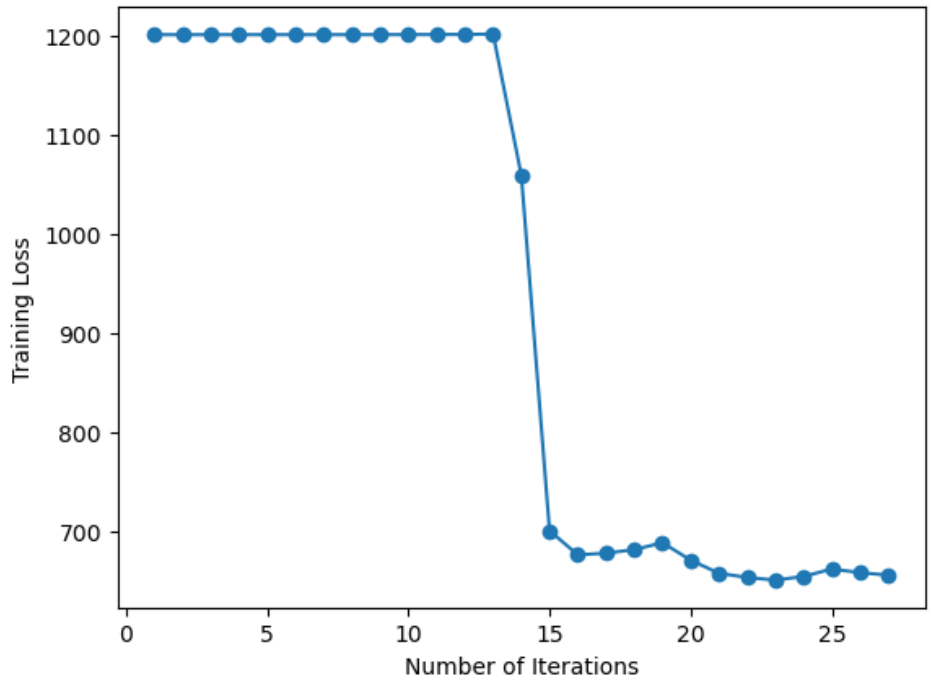

Figure31. Training loss v. Number of Iterations with noisy data, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-5.

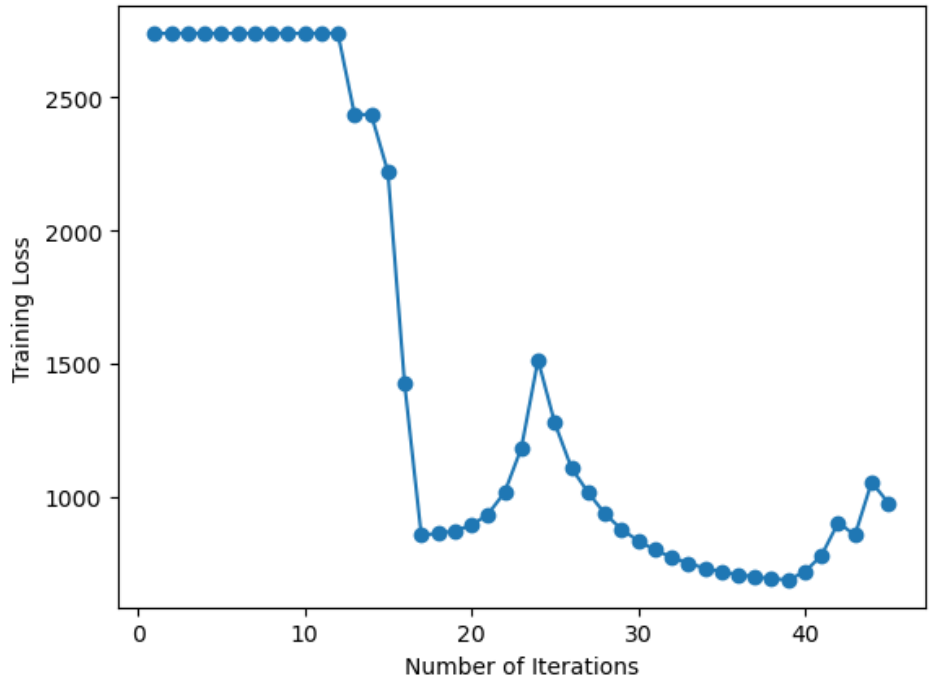

Figure32. Training loss v. Number of Iterations with noisy data, W initializes with uniformly random value from [-2,2], and lambda initializes at 10^-5.

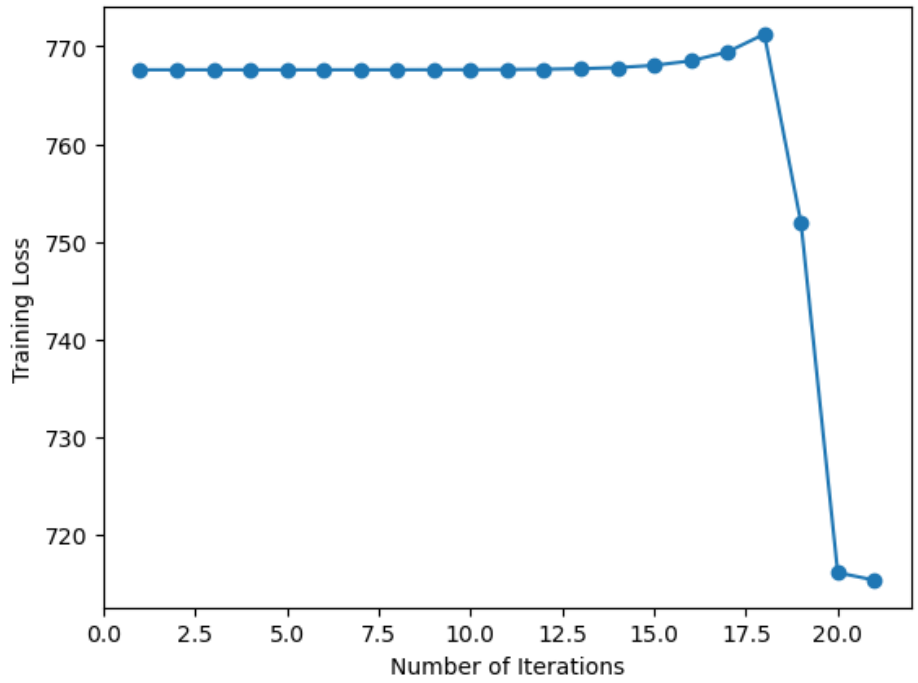

Figure33. Training loss v. Number of Iterations with noisy data, W initializes with [1] * 16, and lambda initializes at 10^-5.

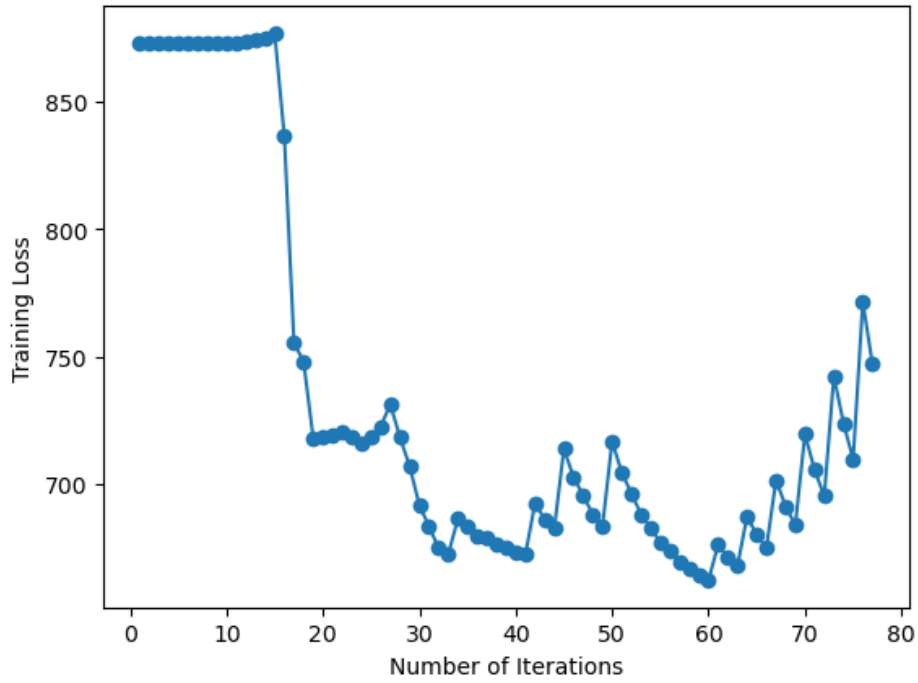

Figure34. Training loss v. Number of Iterations with noisy data, W initializes with [2] * 16, and lambda initializes at 10^-5.

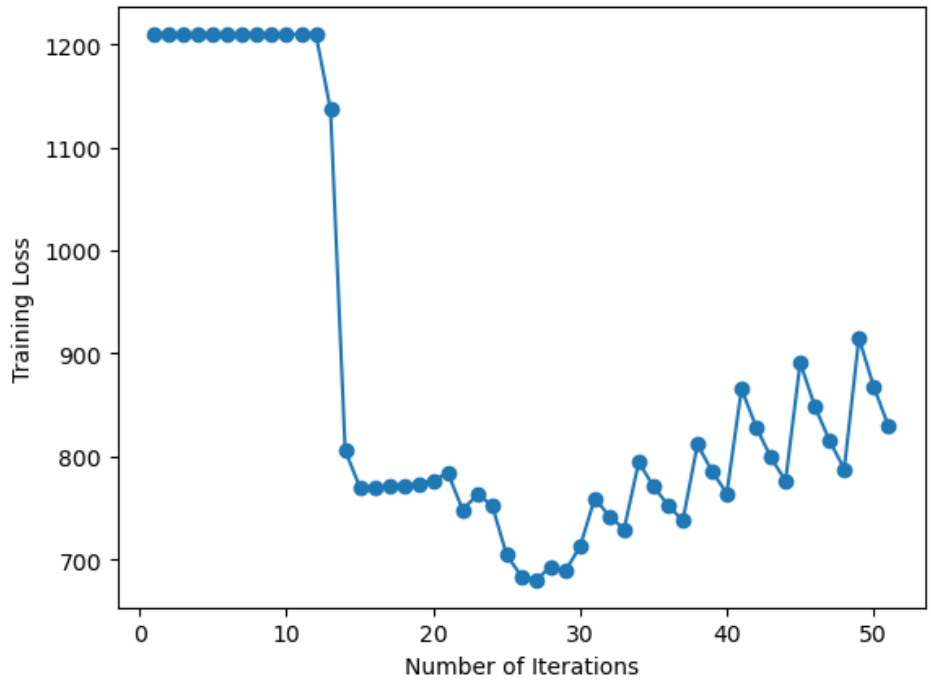

Figure35. Training loss v. Number of Iterations with noisy data, W initializes with standard normalized distribution, and lambda initializes at 10^-5.

Training the neural network with different lambda with epsilon = 2

Lambda	***10^-5***	***10^-4***	***10^-3***	***10^-2***	***0.1***
MSE	***1.291***	***1.276***	***1.277***	***1.291***	***1.291***

Table11. The mean squared error of the training data with different lambda values.

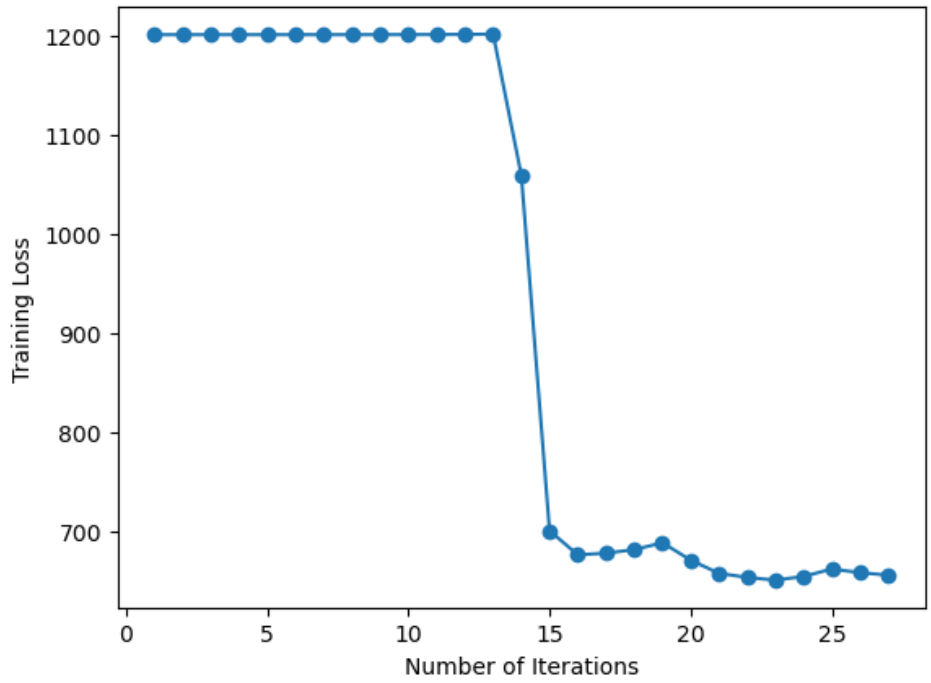

Figure36. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-5.

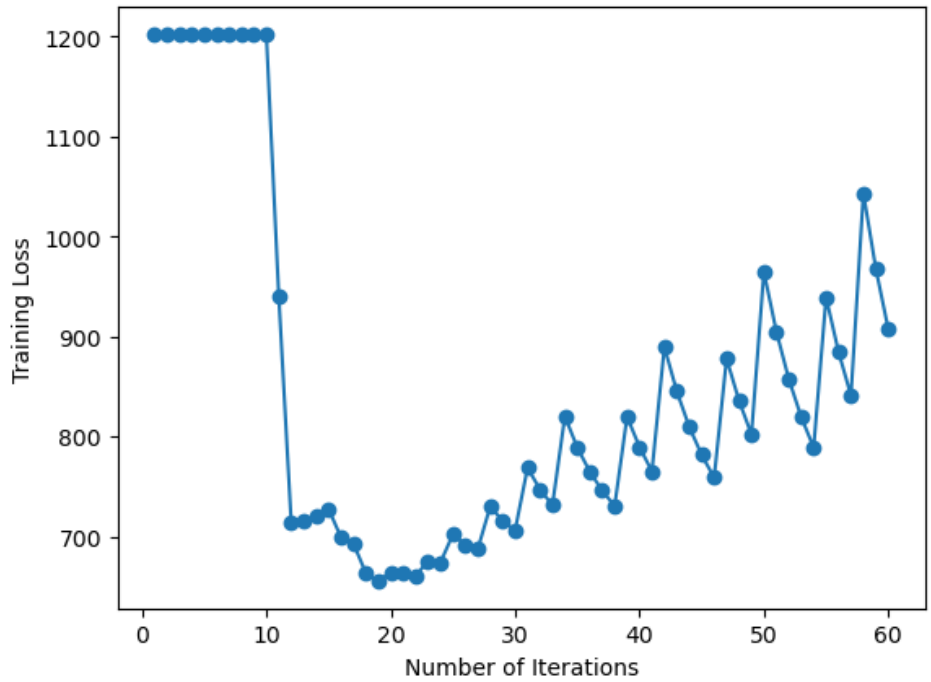

Figure37. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-4.

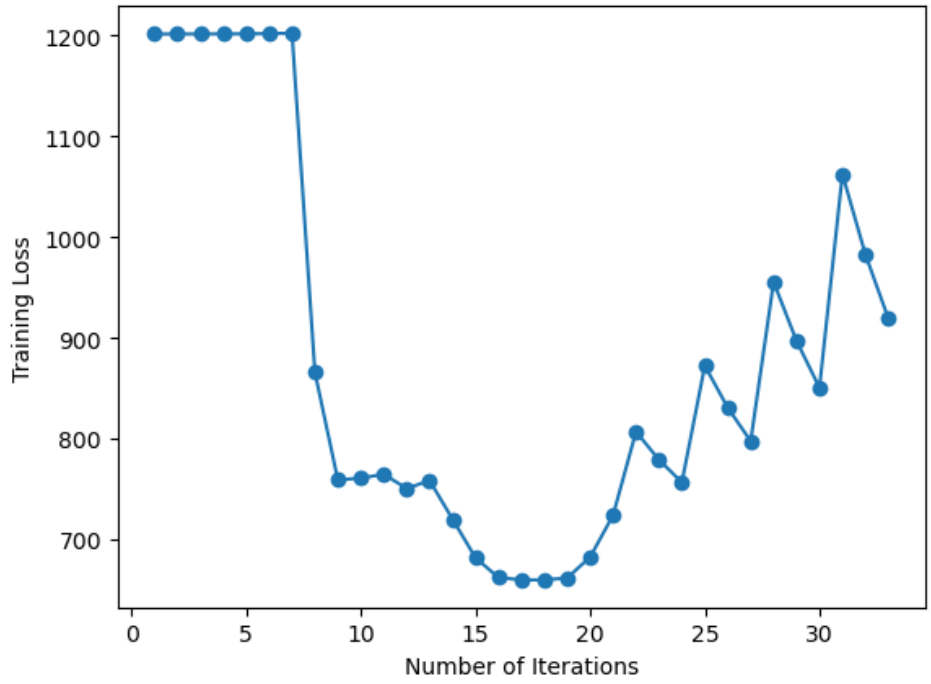

Figure38. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-3.

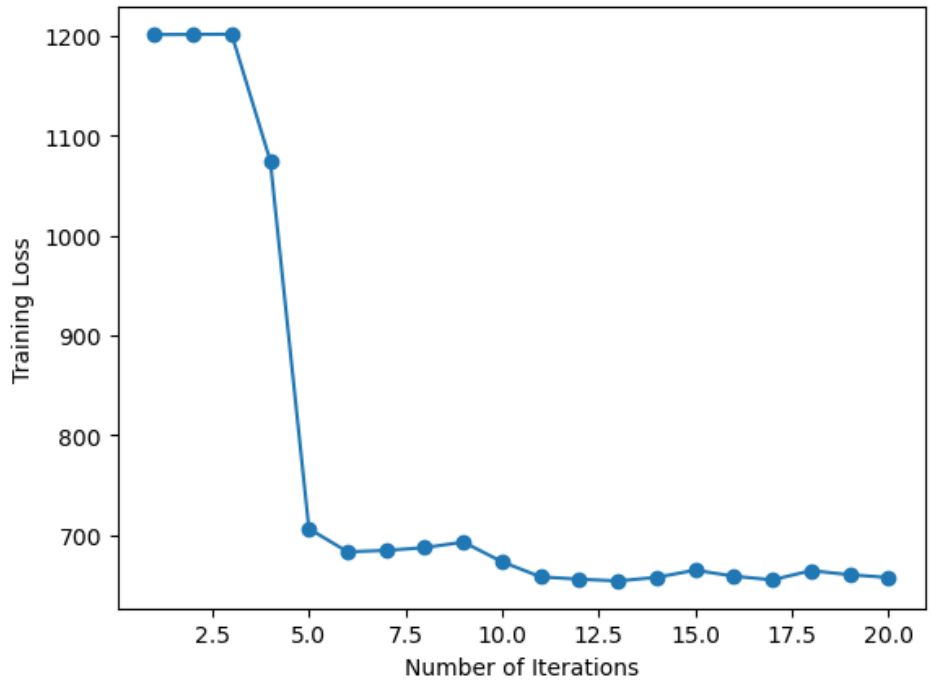

Figure39. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-2.

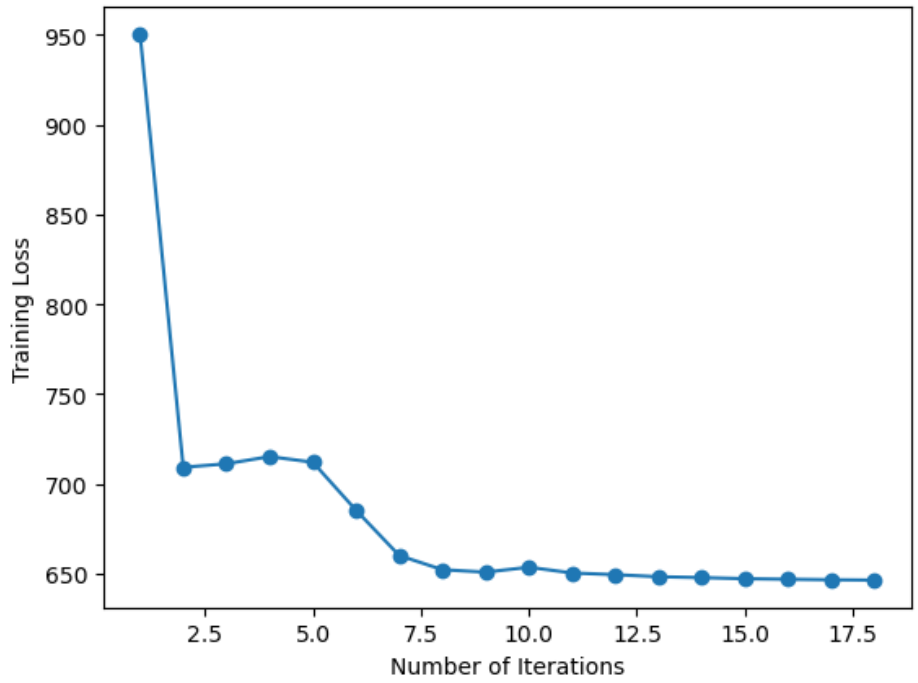

Figure40. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 0.1.

The data generally show a declining curve at different values of lambda with a bit fluctuation. There aren't many changes in the MSE with the change of lambda values. Generally less steps are needed with a larger lambda value.

Testing the neural network with different gamma with epsilon = 2

We will use the weights derived from training when initialized W is uniformly random data from [-1,1] and the lambda value is 10^-5, which has a MSE of 1.291.

Gamma	1	2	3	4	5
MSE	1.373	2.696	8.424	27.894	67.156

Table12. The mean squared error of the test data with different gamma values.

Training the neural network with different initialized weights with epsilon =5

Parameter W	Uniform random [-1,1]	Uniform random [-2,2]	[1] * 16	Standard normal distribution	[2] * 16
MSE	8.337	8.276	7.978	8.293	8.006

Table13. The mean squared error of the training data starts at different initialization values with the new function.

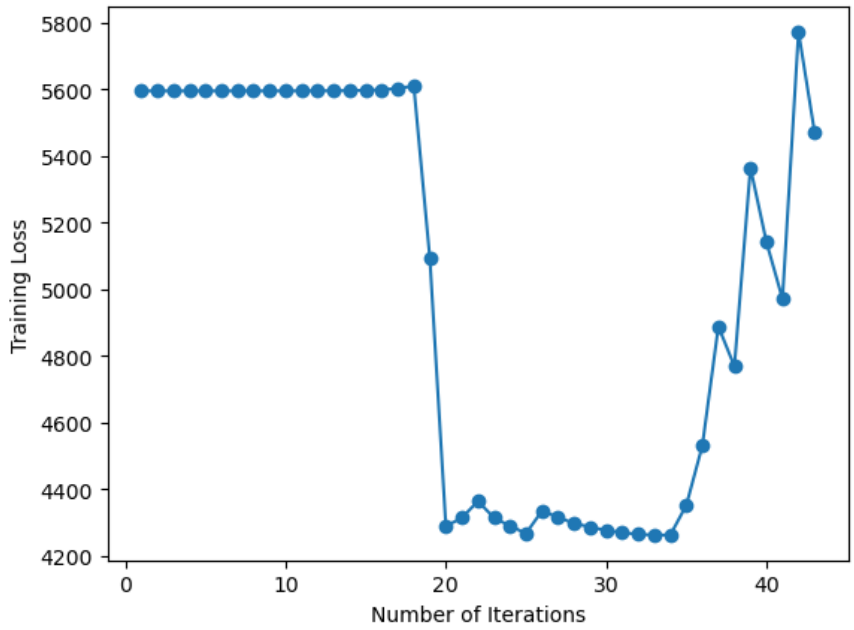

Figure31. Training loss v. Number of Iterations with noisy data, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-5.

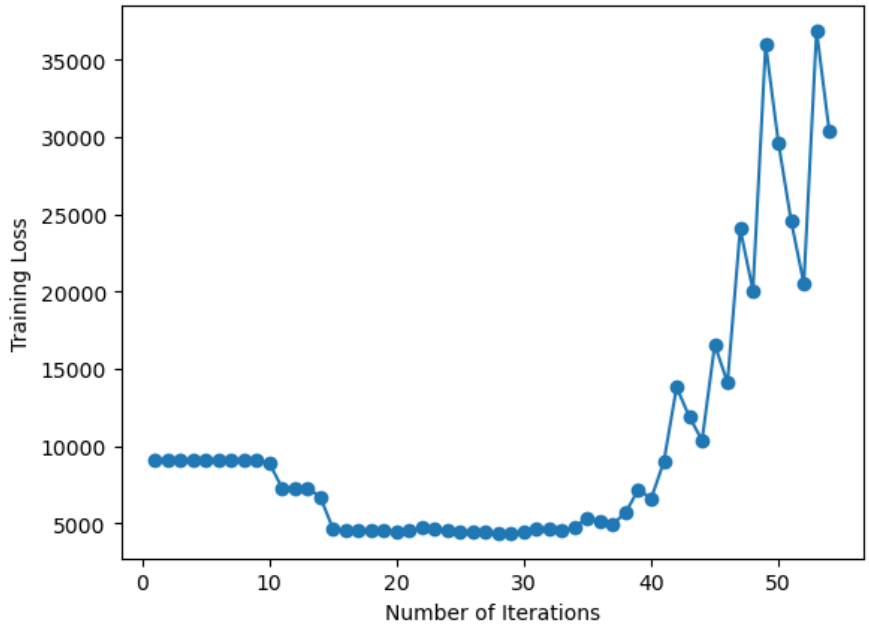

Figure32. Training loss v. Number of Iterations with noisy data, W initializes with uniformly random value from [-2,2], and lambda initializes at 10^-5.

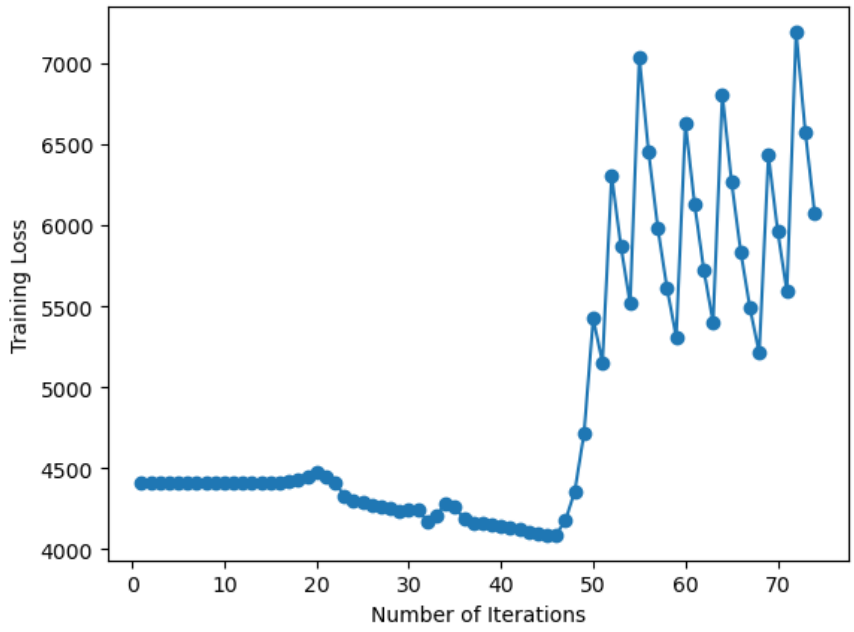

Figure33. Training loss v. Number of Iterations with noisy data, W initializes with [1] * 16, and lambda initializes at 10^-5.

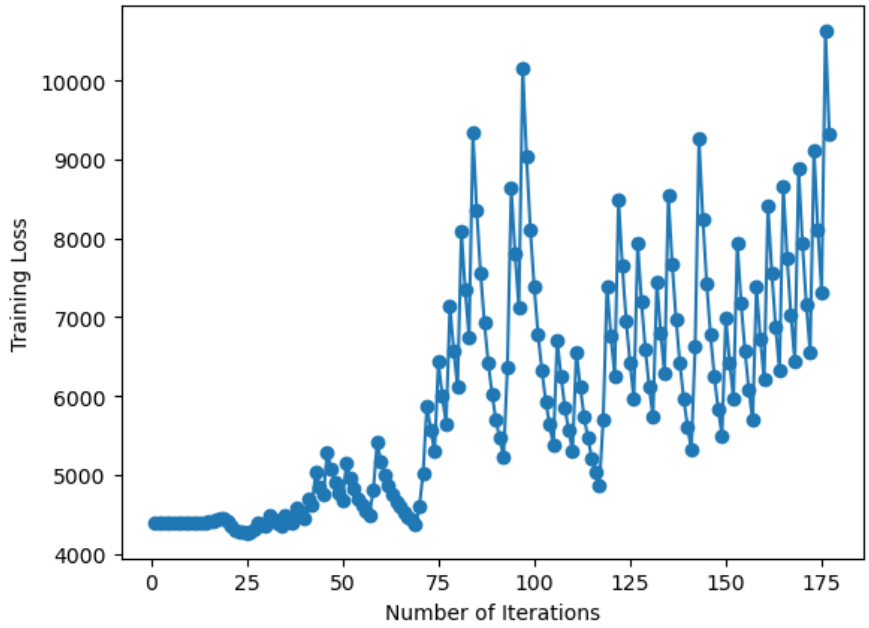

Figure34. Training loss v. Number of Iterations with noisy data, W initializes with [2] * 16, and lambda initializes at 10^-5.

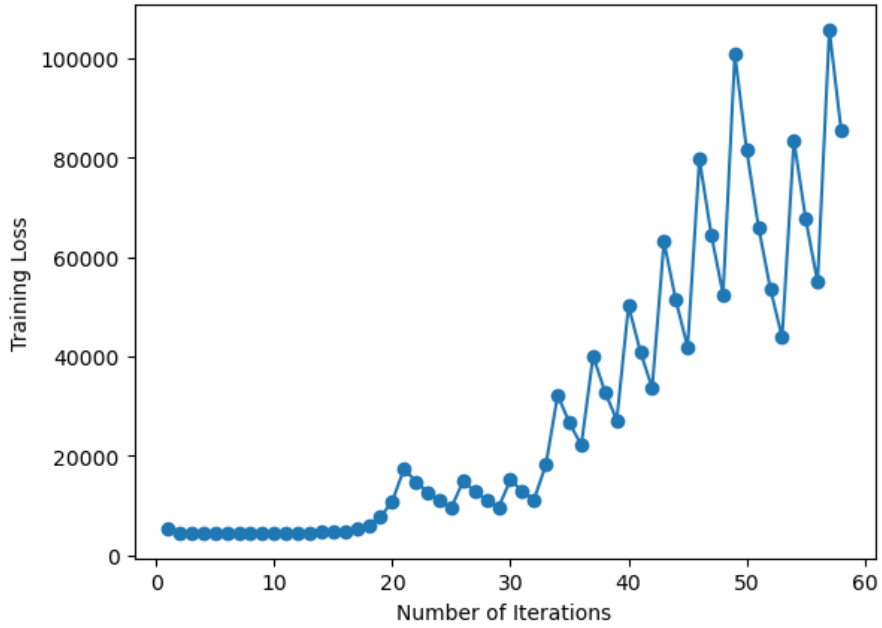

Figure35. Training loss v. Number of Iterations with noisy data, W initializes with standard normalized distribution, and lambda initializes at 10^-5.

The value of MSE is relatively similar with different initializations, and larger compared to epsilon =2, a lot of fluctuations and oscillations are shown in the curve, and the training loss is rising sometimes.

Training the neural network with different lambda with epsilon = 5

Lambda	***10^-5***	***10^-4***	***10^-3***	***10^-2***	***0.1***
MSE	***8.337***	***8.070***	***8.247***	***8.337***	***8.338***

Table14. The mean squared error of the training data with different lambda values.

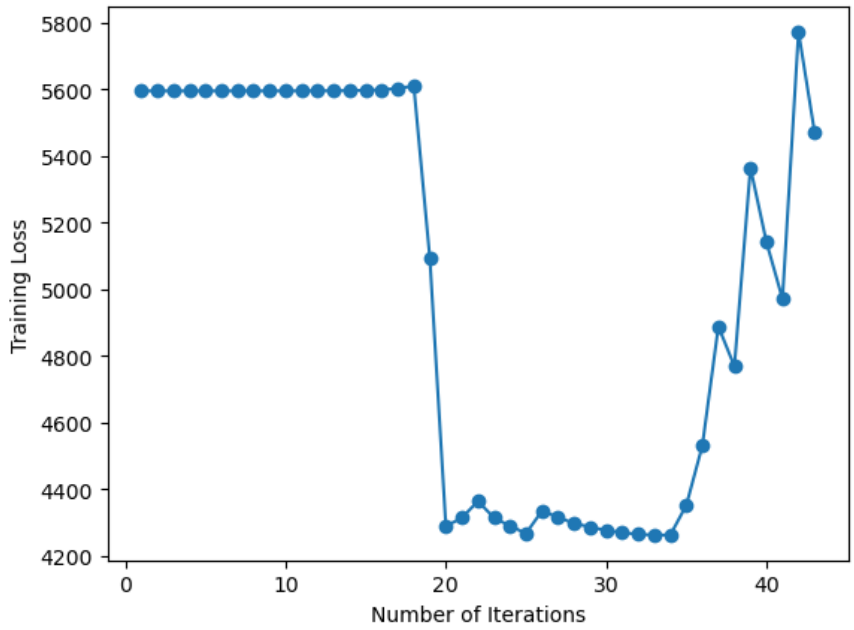

Figure36. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-5.

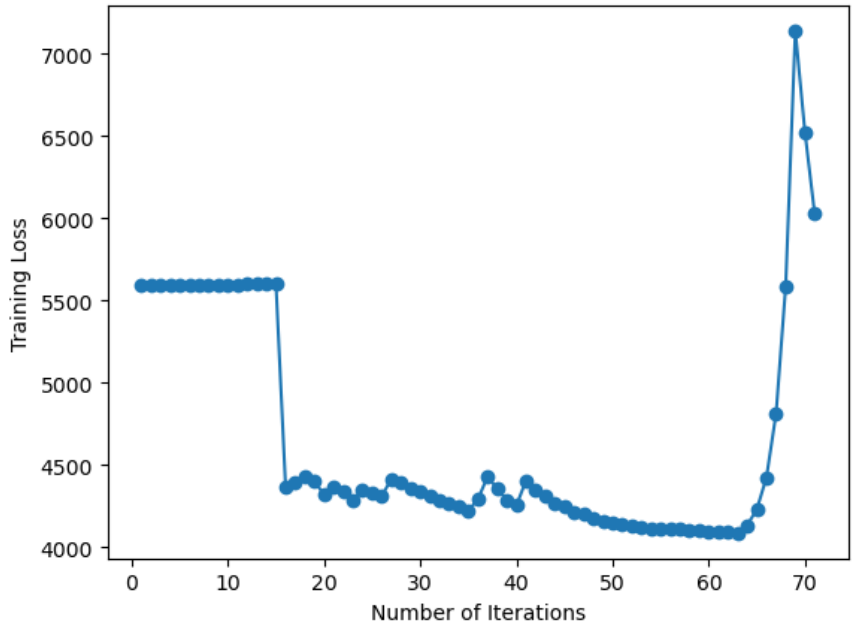

Figure37. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-4.

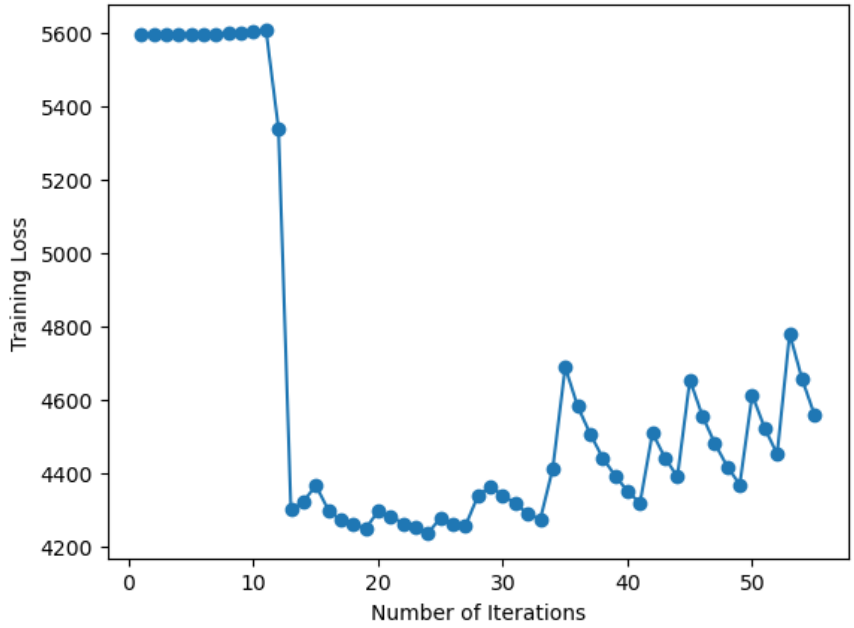

Figure38. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-3.

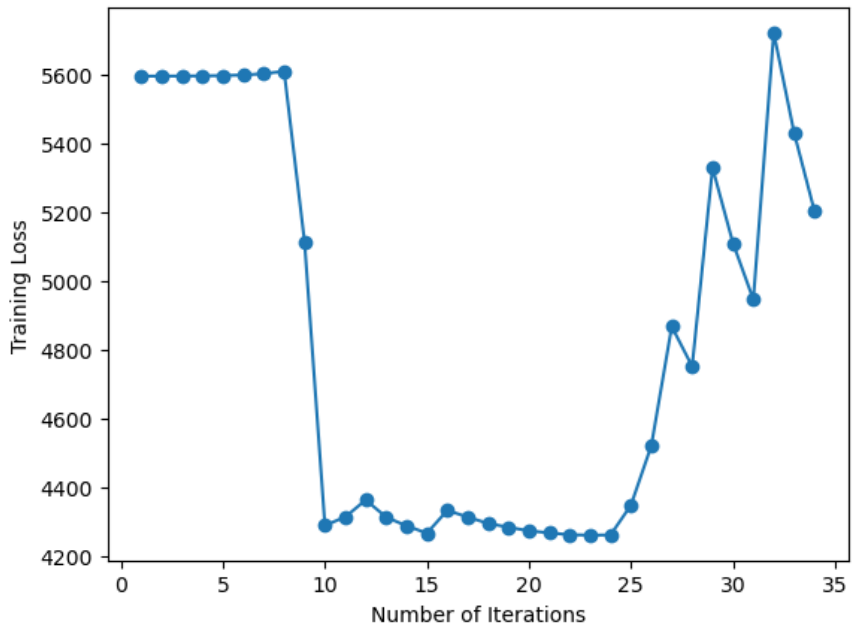

Figure39. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 10^-2.

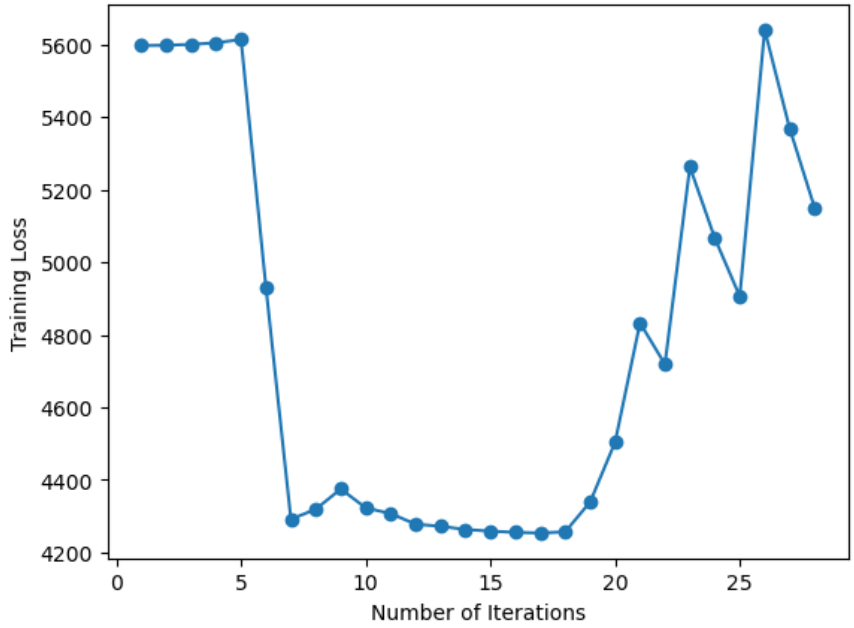

Figure40. Training loss v. Number of Iterations with f_w, W initializes with uniformly random value from [-1,1], and lambda initializes at 0.1.

The data generally have plateaus at the beginning and the middle and a sharp declining curve, but then it starts to rise with oscillations.

Testing the neural network with different gamma with epsilon =5

We will use the weights derived from training when initialized W is uniformly random data from [-1,1] and the lambda value is 10^-5, which has a MSE of 1.291.

Gamma	1	2	3	4	5
MSE	8.281	9.857	20.980	50.229	87.170

Table15. The mean squared error of the test data with different gamma values.

In general, we still see the trend that different initializations lead to different minimizers, though most of the time the MSE is still similar. The lambda doesn't change the MSE a lot but the pattern of the training loss curve might change, sometimes with more fluctuations when the lambda is large (it doesn't always happen though). The MSE increases with the increase of gamma as seen in other sets of data.

With a larger epsilon, there is more noise and thus randomness in the data, and this leads to a larger error rate. This is reasonable since with more noise it would be harder for the model to learn the pattern of the original function.

Contour plot