**Introduction**

The third assignment of the course focuses on the implementation and results of two supervised learning models, a linear regressor and an artifical neural network (ANN) with one hidden layer. Both models are trained to fit a one dimensional input-output dataset, and their performance is evalued using the mean-squared error metric. Additionally, using hyperparameter adjustment, we observe the results of model complexity on learning ability.

**Implementation Results**

**Part A**

**Q) Is it sufficient to use a linear regressor or is it necessary to use an ANN with a single hidden layer? If it is latter, what will be the minimum number of hidden units?**

**A:** As will be seen in the later results in this report, it is necessary to use an artificial neural network with a single layer rather than a linear regressor, due to the fact that the distribution of the data is non-linear and linear regressor models cannot catch non-linear data. However, artifical neural networks can learn from data and fit accordingly.

**Q) What is a good value for the learning rate?**

**A:** Setting the learning rate as provided sufficient results.

**Q) How to initialize the weights?**

**A:** Weights were initialized using small random values from a normal distribution scaled by .

**Q) How many epochs should you use? How to decide when to stop?**

**A:** Since the regression task is computationally easy, we can experiment with large values of epochs. There are two choices for stopping. Either the training phase ends with reach the upper bound of epoch, or we set a threshold for error, such that when the error gets desirably small, the training phase ends itself. However, this mechanism **(early stopping)** is not implemented in our model.

**Q) Does normalization affect the learning process for this application?**

**A:** Yes, normalization significantly improves the learning process. Without normalization, the model failed to converge due to large gradients and unstable updates.

**Part B**

The table given below displays the details of the artifical neural network model that we have implemented for the regression task:

|  |  |
| --- | --- |
| **ANN Used** | 64 Hidden Layers |
| **Learning Rate** | 0.001 |
| **Range of initial weights** | Normal distribution scaled by 0.1 |
| **Number of Epochs** | 1000 |
| **When to Stop** | When epochs end (no early stopping) |
| **Is Normalization Used** | Yes |
| **Training Loss (averaged)** | 46.6088 |
| **Test Loss (averaged)** | 78.8576 |

The figures given below display the results of the artificial neural network with parameters provided above, compared with the linear regressor model, and the loss curves per epoch:

A comparison of graphs with numbers and symbols

AI-generated content may be incorrect.

Figure 1: Results of the regression task on both training and test sets.

A graph with red and orange lines

AI-generated content may be incorrect.

Figure 2: Training loss per epoch for both linear regressor and ANN regressor.

**Part C**

In this part, by using grid search, we have tested with multiple values of hyperparameters. The table given below displays the results of our tests with their respective hyperparameters:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Hidden Layers** | **Learning Rates** | **Epochs** | **Train MSE** | **Train STD** | **Test MSE** | **Test STD** |
| 2 | 0.01 | 1000 | 432.184 | 437.298 | 567.221 | 711.836 |
| 2 | 0.0005 | 1000 | 340.736 | 445.03 | 458.394 | 688.143 |
| 2 | 0.0001 | 1000 | 500.557 | 628.912 | 696.867 | 858.409 |
| 4 | 0.01 | 1000 | 350.489 | 479.92 | 497.099 | 744.636 |
| 4 | 0.0005 | 1000 | 332.782 | 452.1 | 448.083 | 698.826 |
| 4 | 0.0001 | 1000 | 445.564 | 532.514 | 615.564 | 767.223 |
| 8 | 0.01 | 1000 | 340.577 | 545.207 | 498.219 | 824.49 |
| 8 | 0.0005 | 1000 | 332.577 | 453.373 | 446.99 | 701.217 |
| 8 | 0.0001 | 1000 | 425.705 | 500.356 | 581.565 | 741.386 |
| 16 | 0.01 | 1000 | 355.668 | 607.332 | 530.888 | 902.679 |
| 16 | 0.0005 | 1000 | 333.684 | 454.212 | 447.215 | 702.938 |
| 16 | 0.0001 | 1000 | 417.586 | 481.648 | 560.524 | 727.018 |
| 32 | 0.01 | 1000 | 96.982 | 161.848 | 129.652 | 168.106 |
| 32 | 0.0005 | 1000 | 47.083 | 69.189 | 100.168 | 172.322 |
| 32 | 0.0001 | 1000 | 337.287 | 422.282 | 466.107 | 624.129 |

Inspecting the results of the parameter tuning yield that while increasing the number of neurons on the hidden layer increase the ability of capturing more complex features of the model, it may also lead to underfitting, hence it must be done with caution. On the other hand, learning rate adjustment is a crucial step in obtaining the optimal model. Decreasing the learning rate too much can also lead to underfitting, and increase computational costs, hence it must also be done with caution. The plots given below display results for various hidden layer numbers, and learning rates:

A graph with blue dots and a red line

AI-generated content may be incorrect.

**Figure 3:** Results for 2 hidden layer neurons and a learning rate of 0.005.

A graph with blue dots and a red line

AI-generated content may be incorrect.

**Figure 4:** Results for 4 hidden layer neurons and a learning rate of 0.005.

A graph with blue dots and a red line

AI-generated content may be incorrect.

**Figure 5:** Results for 8 hidden layer neurons and a learning rate of 0.005.

A graph with blue dots and a red line

AI-generated content may be incorrect.

**Figure 6:** Results for 16 hidden layer neurons and a learning rate of 0.005.

A graph with blue dots and a red line

AI-generated content may be incorrect.

**Figure 7:** Results for 32 hidden layer neurons and a learning rate of 0.005.