CS554 Introduction to Machine Learning Single and Multi Layer Perceptron Regression

Mert Erkol

May 10, 2023

1 Introduction

In this study Single Layer Perceptron(SLP) and Multi Layer Perceptron(MLP) with different number of hidden units fitted to the given custom 1D dataset and their results evaluated with Mean Square of Error.

2 Background

The goal is to fit a model to given custom data that generalizes the problem and predicts next outcomes when new input is given. In this section SLP and MLP networks, MSE loss function, sigmoid activation and backpropagation are explained in detail.

2.1 Single Layer Perceptron

SLP is a neural network architecture that can have multiple input and output units but it has no hidden units between them an example of SLP can seen in the figure 1. SLP has 1 layer of computational node and produce the linear combination the weights, bias and the given input.

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{1}$$

where y is the output of the nodes, f is the selected activation function which maps the linear output to a non-linear output, w_i is the i_th weight of the network, x_i is the i_th feature of the given input and b is the bias of the layer [1].

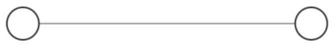


Figure 1: SLP with 1 input unit and 1 output unit

2.2 Multi Layer Perceptron

MLP is also an artificial neural network architecture very similar to SLP. But instead of 1 computational layer of nodes it can have multiple layers called hidden layers and then calculates these linear combinations between input to hidden and hidden to output layers. An example architecture with 1 hidden layer can seen below in figure 2

$$h_j = f\left(\sum_{i=1}^{n} w_{ji}^{(1)} x_i + b_j^{(1)}\right) \tag{2}$$

where h_j the output of hidden layer neurons is calculated just like SLP before but now we have 2 different weight matrices w_{ji} and b_j for the first layer

$$y_k = f\left(\sum_{j=1}^{m} w_{kj}^{(2)} h_j + b_k^{(2)}\right) \tag{3}$$

where y_k is the predicted output of hidden layers. w_{kj} and b_K are the weights and bias for the output layer.

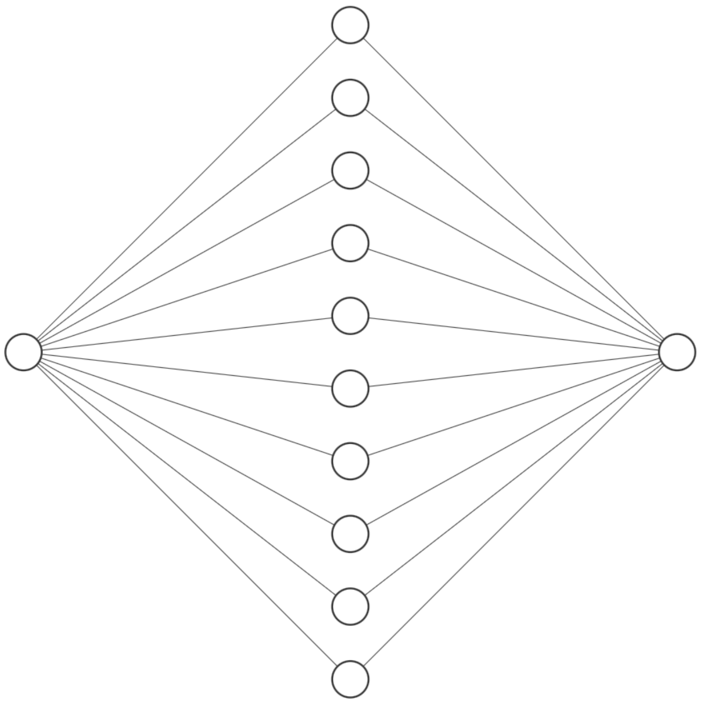


Figure 2: MLP with 1 input unit, 1 hidden layer and 10 hidden units and 1 output unit

2.3 Activation function

An activation function is a mathematical operation that is applied to an output of a neuron. It provides non-linearity to the neural network. There are several activation functions is available but for this work sigmoid activation function has been used as follows

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

where x is the either output of the neuron or the input

2.4 Loss function

The Mean Square Error(MSE) is a common loss function for evaluating regression. It calculates the distance between the prediction and the desired value squares and calculates the mean for all data points.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{5}$$

where n is the number of samples in the dataset. $\hat{y}_i$ is the $i_t h$ prediction and y_i is the $i_t h$ desired output.

2.5 Backpropagation

The backpropagation is a supervised optimization method for artificial neural networks. It helps us to calculate the gradients for weight and biases to minimize the loss function(5). After the forward pass as mentioned in one of the networks (1) total loss is calculated according to the loss formula in (5). For gradient descent algorithm , gradients can be calculated with backpropagation for weight and bias updates. The update rule for weights can be defined as (6)

$$w_{new} = w_{old} + \eta \triangle w \tag{6}$$

where w_{new} is the new weight parameter, w_{old} is the current weight parameter, η is the hyperparameter learning rate, $\triangle w$ is the gradient of the weight parameters.

The $\triangle w$ can be calculated as:

$$\triangle w = -\frac{\partial L}{\partial w} \tag{7}$$

where ∂ is the partial derivative, L is the loss function

3 Methodology

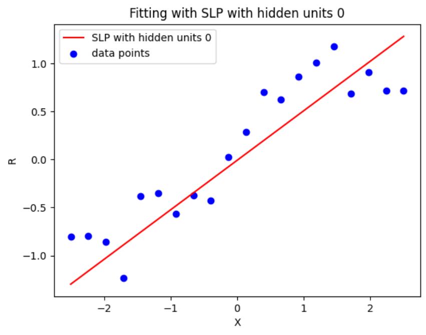
To complete the project Python@3.9 has been used as a programming language. NumPy library is used for matrix operations such as addition, vectorization, dot product. The structure is follows, Read the train and test data, train the SLP and multiple MLP networks, evaluate them on test data by calculating MSE. For MLP training multiple combinations of learning rate(lr) and iteration size has been tested and it converges to best results at when lr is equal to 0.01 and 5000 iterations for training. MLP structure is follows forward, sigmoid activation and forward again. SLP structure is only 1 forward layer without any activation function.

4 Results

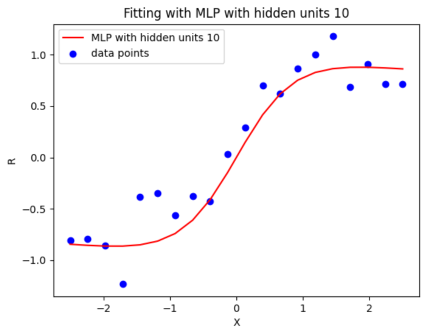
In this section models that fitted on training data and loss versus complexity models can be seen

5 Discussion

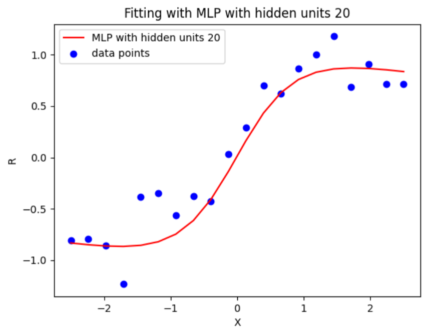
As seen in the figure 4 the MLP model with 10 hidden units in 3b is the best network selection for the given dataset. It minimizes the error very nicely and it also the power of generalization. The test and train errors on models 3c and 3d is lower than the 3b but the error difference is not high enough and we are starting to overfit the training dataset. So in this case we select the model MLP with 10 hidden units. Also the significant improvement can be seen when we go from SLP to any MLP model that is because the optimal solution for this dataset is not linear and we can train non-linear models with models with hidden layers like MLPs.



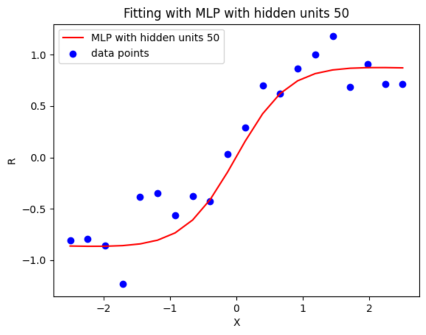
(a) SLP network



(b) MLP with hidden = 10



(c) MLP with hidden = 20



(d) MLP with hidden = 50

Figure 3: Networks fitted on training data

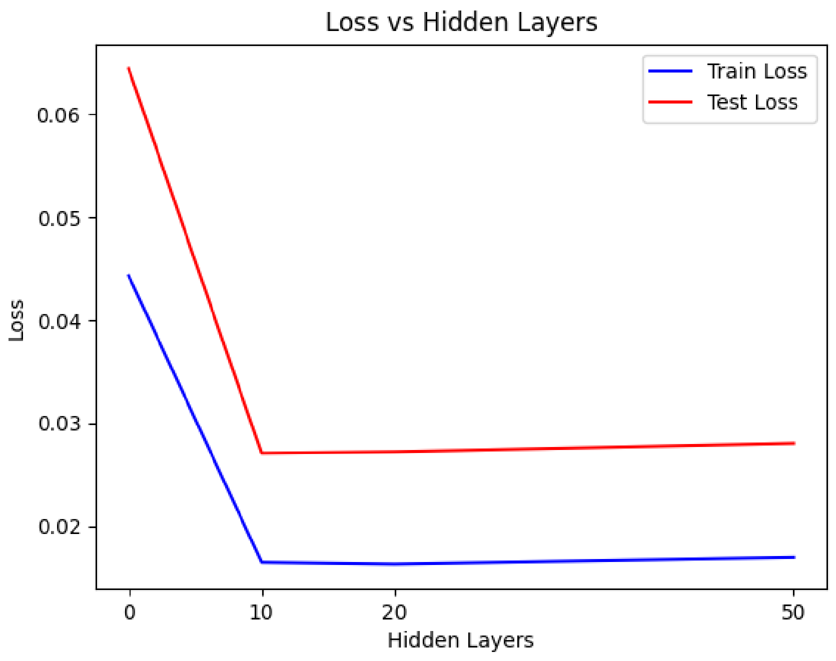


Figure 4: Caption

6 Conclusion

In this study SLP and MLP network architectures implemented from scratch, optimized with gradient descent and evaluated with MSE. According to the results and the general idea from discussion section. Selecting the correct architecture, optimum hyperparameters, network size, loss function and optimum optimization algorithm are very crucial decisions need to be decided for best performing model. While selecting the best performing model any researcher must remember that lowest error does not mean best performance, the importance of generalization is significant. We must prevent the model from overfitting.

References

[1] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2014.