Artificial Neural Network Regressor

CS 550 Assignment II

Emirhan Koç
Bilkent University
Electrical and Electronics Engineering
emirhan.koc@bilkent.edu.tr

I. Introduction

In this assignment, it is asked to implement an artificial neural network (ANN) regressor and learn its weights by implementing the backpropagation algorith. This implementation supports an ANN for linear regression (no hidden layer) and an ANN with single hidden layer. After implementing ANNs, it is tested on two datasets contains 1D input features and 1D outputs. Evaluations on these datasets are displayed and discussed in the further sections. Each dataset is divided into training and test sets and evaluations are performed on both of them. In this implementation, it is tried different choices of parameters/methods such as learning rate, momentum, number of hidden layer and number of hidden units and activation functions.

II. Models

A. Linear Regression Model

In linear regression, an d-dimensional vector x is mapped to scalar value $\hat{y}$ which denotes the prediction of true value y. Here, W is also a d-dimensional weight vector and b is the bias term. The objective is to minimize the difference between predicted value $\hat{y}$ and the true value y.

$$\hat{y} = W^T * x \tag{1}$$

where the objective is to find a W to minimize error defined below.

$$\frac{\sum_{i=1}^{M}(y_i - \hat{y_i})^2}{2M} \tag{2}$$

B. ANN Regressor

In thisp part, an ANN with one hidden layer with different hidden units are built. Here, two sets of weights are created such that first one is from input to hidden layer, and the second one is from hidden layer to output layer.

$$\begin{aligned} z^{[1]} &= W^{[1]} * a^{[1]} + b^{[1]} \\ a^{[2]} &= \sigma(z^{[1]}) \\ z^{[2]} &= W^{[2]} * a^{[2]} + b^{[2]} \\ a^{[3]} &= \sigma(z^{[2]}) \\ \hat{y} &= a^{[3]} \end{aligned} \tag{3}$$

Eq.3 is the generan form of forward pass in single layer hidden unit and σ is the sigmoid activation function. In our experiments, different activation functions are tried such as ReLU, tanh. Similarly, our aim is to find proper weight vectors in each layer to minimize error. In this structure, binary cross entropy and mean-squared error is used as loss metric. In Eq.4, binary cross-entropy loss is defined where y is true value and p is the prediction value.

$$L(y, p) = -(y \log(p) + (1 - y) \log(1 - p)) \tag{4}$$

III. Best Configurations for Dataset

In this part, I have experimented both of the architecture on two different dataset using different hyperparameters such as learning rates, hidden layer and unit sizes, loss metric and weight initialization. Below, evaluations for both of the dataset are presented.

A. Dataset I

ANN:ANN Regressor with single hidden layer, 16 units
Selected activation function: tanh
Selected loss function: Mean Squared Error
Learning rate: 0.01
Range of initial weights: (-0.1 to 0.1) Uniform distribution
Number of epochs: 2000
When to stop: When reached pre-defined epoch count
Is momentum used: Yes, a=0.1
Is normalization used: Yes, the data is standardized
Stochastic or batch learning: SGD is preferred
Training loss (averaged over training instances): 0.021 mean, 0.0308 std
Test loss (averaged over test instances): 0.05 mean, 0.07 std

B. Dataset II

ANN:ANN Regressor with single hidden layer, 8 units
Selected activation function: tanh
Selected loss function: Mean Squared Error
Learning rate: 0.01
Range of initial weights: (-0.1 to 0.1) Uniform distribution
Number of epochs: 16000
When to stop: When reached pre-defined epoch count
Is momentum used: Yes, a=0.1
Is normalization used: Yes, the data is standardized
Stochastic or batch learning: SGD is preferred
Training loss (averaged over training instances): 0.096 mean, 0.22 std

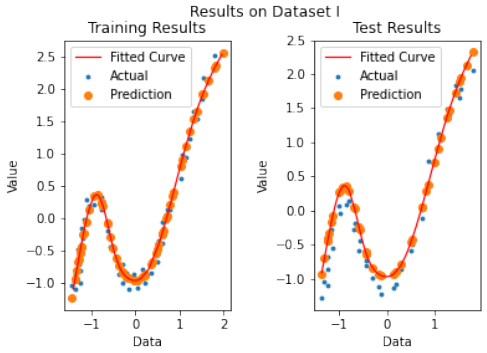

Fig. 1. ANN with selected hyperparameters in first dataset

Test loss (averaged over test instances): 0.87 mean, 1.34 std

Fig. 2. ANN with selected hyperparameters in second dataset

IV. Effect of Hidden Layer and Hidden Units

In this part, it is observed effects of hidden layer with non-linearity, hidden unit size, choice of activation function and initialization in power of prediction. Firstly, it is seen that linear regressor with no hidden layer is not successfull compared to ANN regressor since the data is highly non-linear.It is also seen that increased hidden unit number effect the ability of model to capture pattern in data. In my implementation, mean squared loss is decreased from 2 hidden units to 16 hidden units. Also, when the size of hidden units is increased, the number of weights and computational cost is increased. In the figure below, you can see the abovementioned effects in both first and second datasets.

Below is the MSE values for the first dataset:

Size	Linear	2	4	8	16
Train	0.51	0.14	0.02	0.02	0.02
Test	0.44	0.15	0.05	0.058	0.054

Below is the MSE values for the second dataset:

Size	Linear	2	4	8	16
Train	0.92	0.33	0.3	0.09	0.14
Test	1.05	0.76	1.06	0.87	0.85

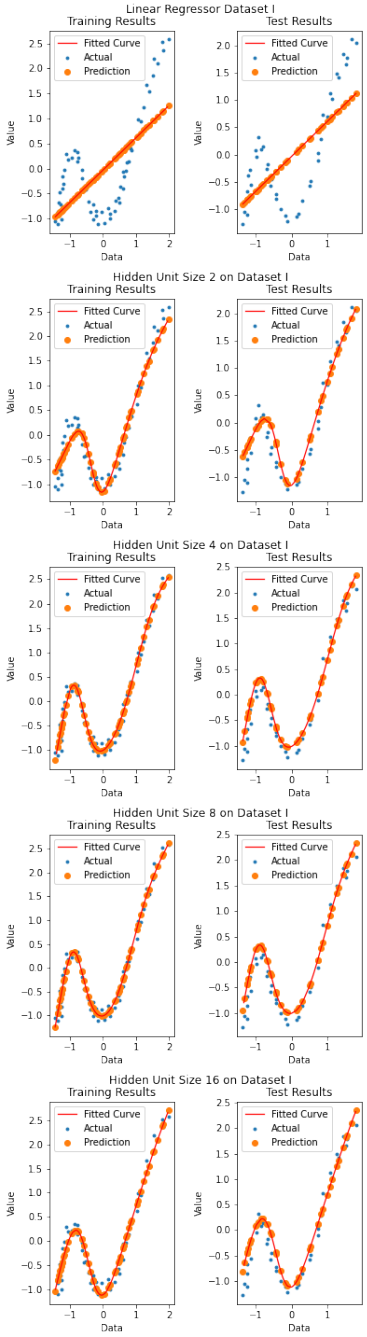

Fig. 3. Train and Test Results in Dataset I

V. Effect of Learning Rate (Part D)

In this part, we investigated how learning rate effect the learning procedure and how thresholding changes the epoch to stop. Also, in this part, I used sigmoid as activation function.According to results obtained from different runs of

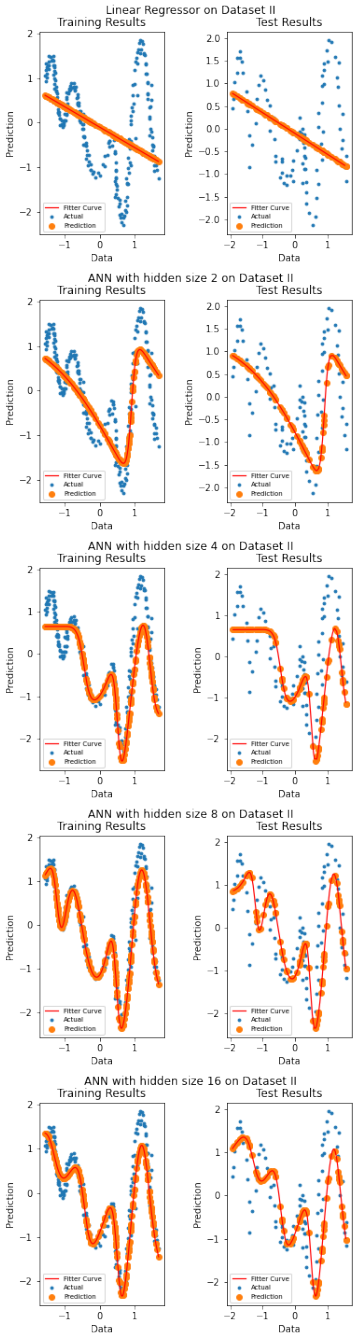

Fig. 4. Train and Test Results in Dataset II

different values of learning rates, high and low values of learning rate effect learning negatively and increases computational cost such that converges become slower and harder. Using the finding in Part C, it is seen that the model works best for learning rate value 0.01.

VI. Effect of Momentum (Part E)

In this part, we investigated how momentum changes learning and its effects on mean squared error loss. Basically, momentum utilizes the information of the gradient of the previous pass and accumulates over iterations. It is usefull in particular in stochastic gradient descent as the gradient direction changes drastically in each iteration with zigzag behaviour. In that sense, accumulation of two consequtive gradient value will help to total gradient to flow in one direction mostly and convergence will be faster.

TABLE I
Effect of Learning Rate in Dataset II

Learning Rate	Loss	Threshold	Epoch
1.0	0.767	0.8	252
0.1	0.167	0.17	2592
0.01	0.089	0.9	16124
0.001	0.14	0.14	11982
0.0001	0.99	1.0	255

TABLE II
Effect of Momentum in Dataset II

Case	Loss	Threshold	Epoch	Momentum
Momentum	0.078	0.8	14516	0.2
No Momentum	0.89	0.9	15521	None

VII. Effect of Batch and Stochastic Gradient(Part F)

In this part, we investigated the effect of batch and stochastic gradient in time for convergence, ability of learning. In stochastic gradient descent method, each instance of the data is given to model separetly and backpropagation is directly applied to this instance and weights are updated.As each instance of the data is passed forward and backward, fluctuation of gradient are occur much and frequently. Herein, this causes the model to find global minimum harder and requires more epochs to find it. Yet, it is advantageous compared to batch learning because high volume of data in forward and backward propagation requires high volume of memory and but increases speed of convergence as it can be implemented in vectorized fashion.

TABLE III
Effect of Batch in Dataset II

Case	Loss	Threshold	Epoch
Batch	0.078	0.8	11298
Stochastic	0.089	0.9	15521

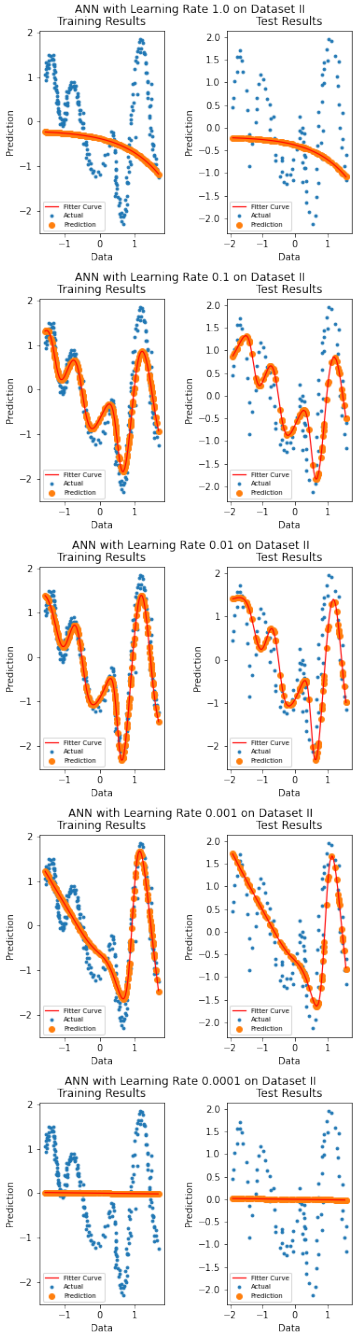

Fig. 5. Train and Test Results in Dataset II

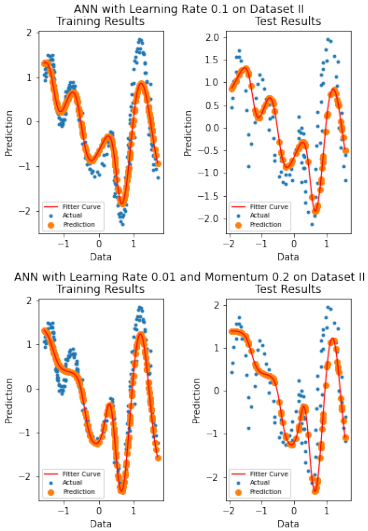

Fig. 6. Train and Test Results in Dataset II

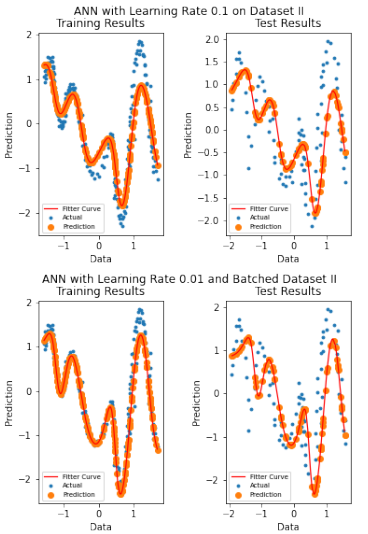

Fig. 7. Train and Test Results in Dataset II: Upper:SGD, Bottom: Batch