Neural Network Laboratory Work – 3

Ravinthiran Partheepan

Multi-Layer Perceptron

- A **multilayer perceptron** (MLP) is a class of feedforward **neural network** .
- MLP utilizes a supervised learning technique called backpropagation for training.
- Its multiple layers and non-linear activation seperates Multilayer perceptron from a linear **perceptron**.
- It can distinguish data that is not linearly separable.

Multi Layer Perceptron Layers

- An **MLP** consists of at least three layers of nodes:
- an input layer,
- a hidden layer
- an output layer.
- Except for the input nodes, each node is a **neuron** that uses a nonlinear activation function
- **Nonlinear activation function** – They allow the model to create complex mappings between the network's inputs and outputs

Adding Linear Transfer Function

- def linear(x):
  return x

- def d_linear(x):
  return(x)

- class Layer:
  activationFunctions = {
  'tanh': (tanh, d_tanh),
  'sigmoid': (sigmoid, d_sigmoid),
  'relu': (relu, d_relu),
  'linear': (linear, d_linear)

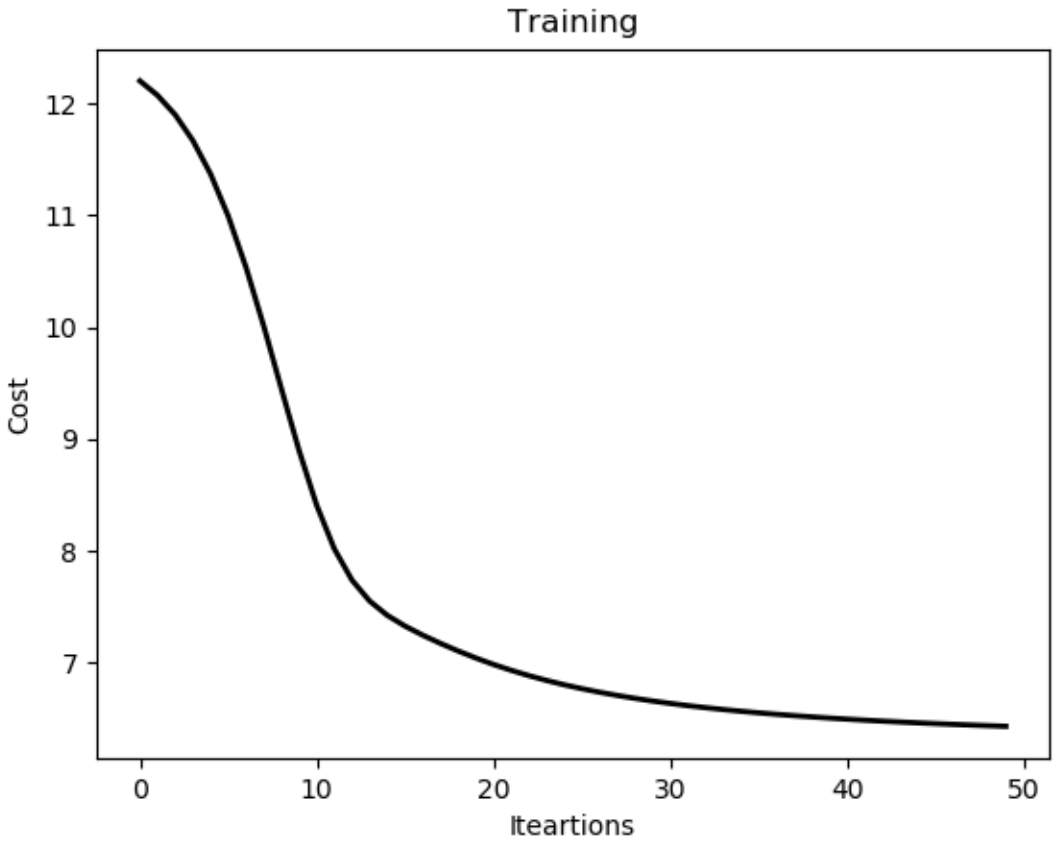
  }

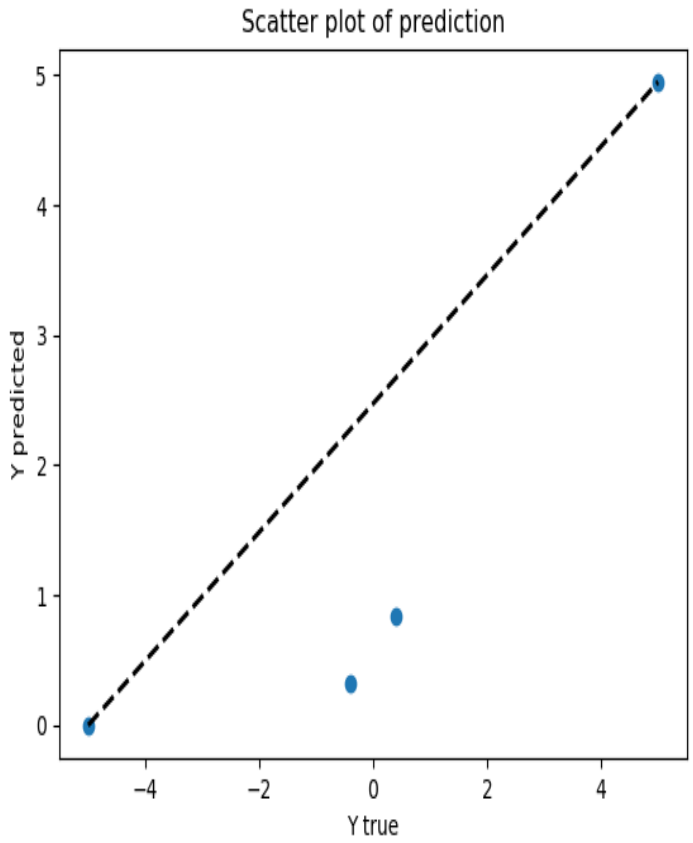
Linear Transfer & Sigmoid Function

- Learning Speed - 0.6
- Iteration / epoch - An **iteration** is a measure of the number of times all of the training vectors are used once to update the weights.
- Epochs defined in this experiment - 50
- Sigmoid Transfer Function - It was used between the hidden and output layers. For computing the variation in weight values between the hidden and output layers
- it exists between (0 to 1).
- Sigmoid Function - 1 / (1 + exp(-x)) (Scales the value from 0 to 1)

Linear Transfer VS Sigmoid

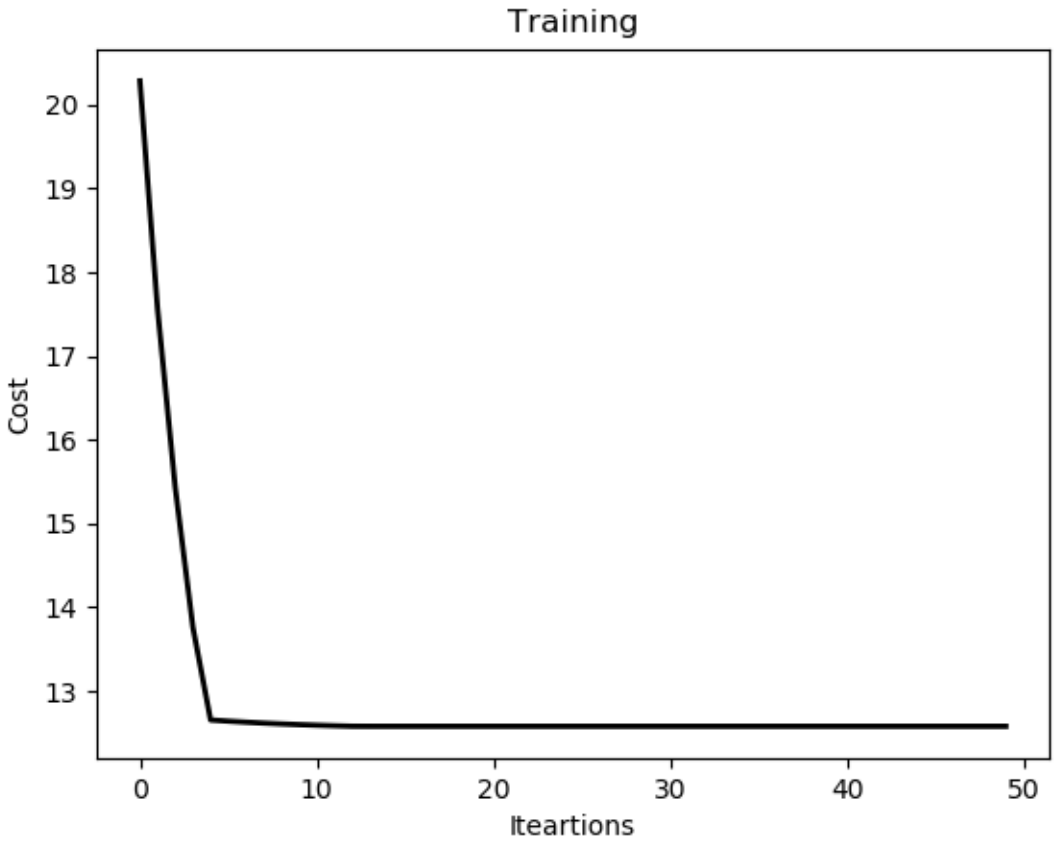
- Pearsons correlation: 0.880, p value 0.120
- MSE: 6.430
- RMSE: 2.536

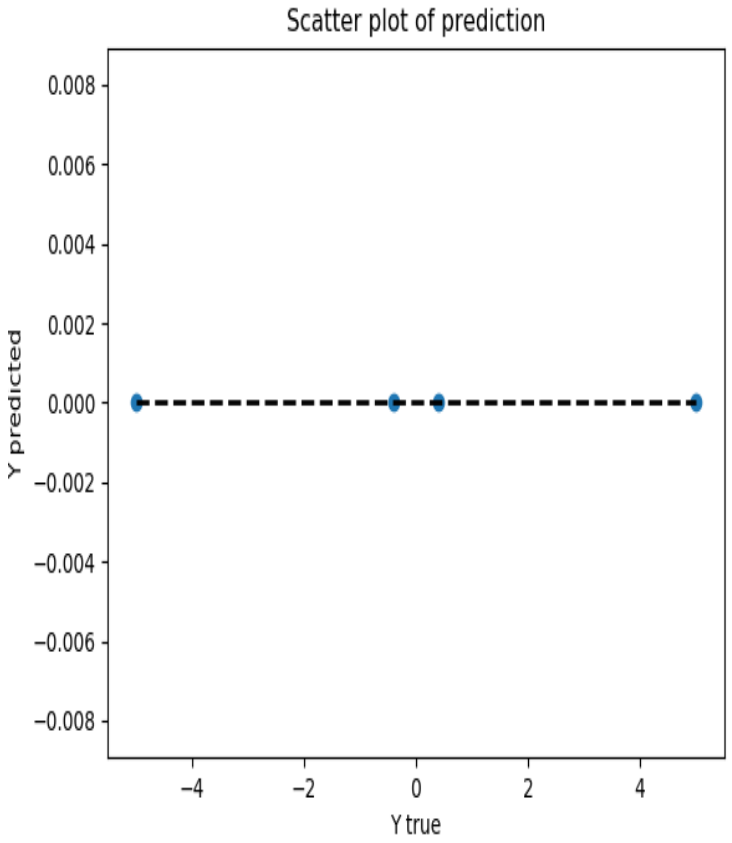




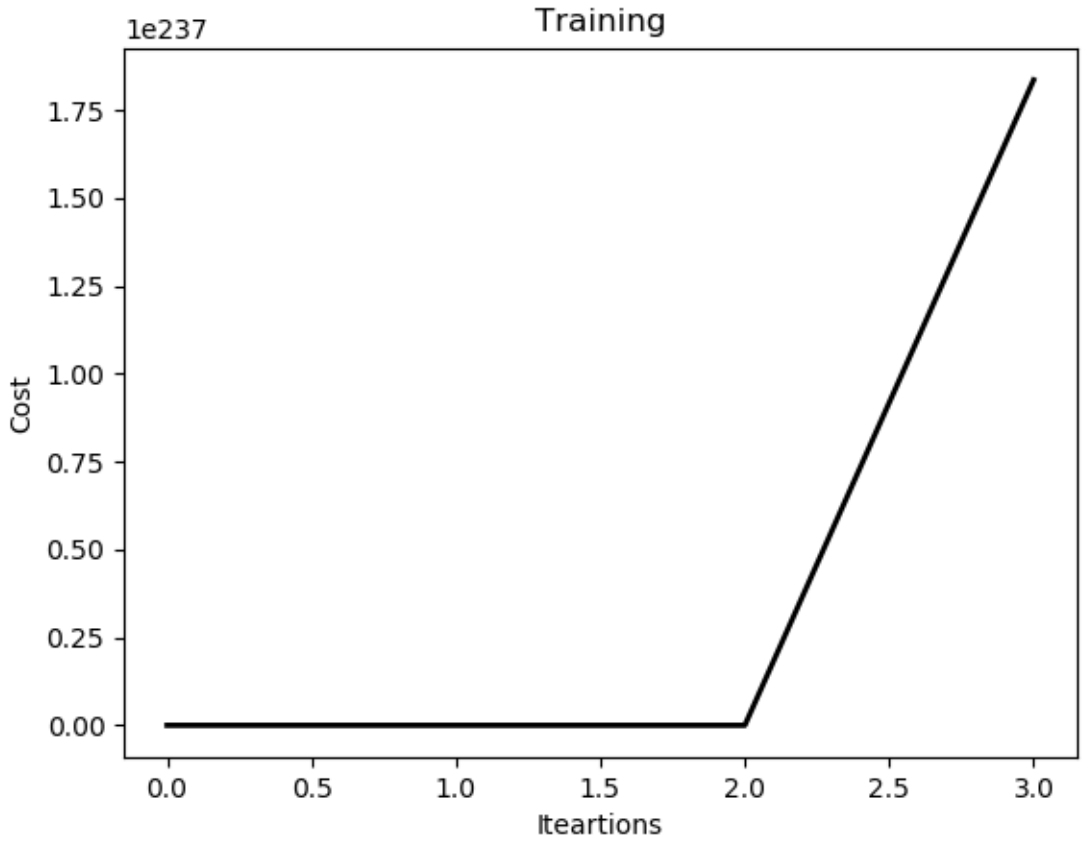
Sigmoid Transfer VS ReLu

- Pearsons correlation: nan, p value nan
- MSE: 12.580
- RMSE: 3.547





Linear Transfer Function



Task – 2 and 3

Selection of Neurons in Hidden Layer

- The number of hidden nodes should be less than twice the size of the nodes in the input layer.
- The number of hidden nodes should be 2/3 the size of input nodes, plus the size of the output node.
- For this experiment, the input layer is 18. Then the hidden nodes should be less than (Input_Layer $<$ 36)
- Number of hidden Nodes used = absolute_Value(18*2/3+1) = 13

RMSE and MSE Calculation

- def mse(y, a):
  return (y - a)**2

- def d_mse(y, a):
  return -2 * (y - a)

- MSE = sum((actual_output - self.target)**2)/N (i.e, Actual Output – Target)

- RMSE = Sqrt(sum(Target – ActualOutput)^2) / N

- **Root-mean-square error** (**RMSE**) is used to measure the differences between values predicted by a model and the values observed.

- The Pearson **correlation coefficient** can be used to summarize the strength of the linear relationship between two data samples.

- The Pearson's **correlation coefficient** is calculated as the covariance of the two variables divided by the product of the standard deviation of each data **sample**.

Parkinson – RMSE,MSE, Correlation

- Motor detoriation progression MSE: 1.452
- Motor detoriation progression RMSE: 1.205

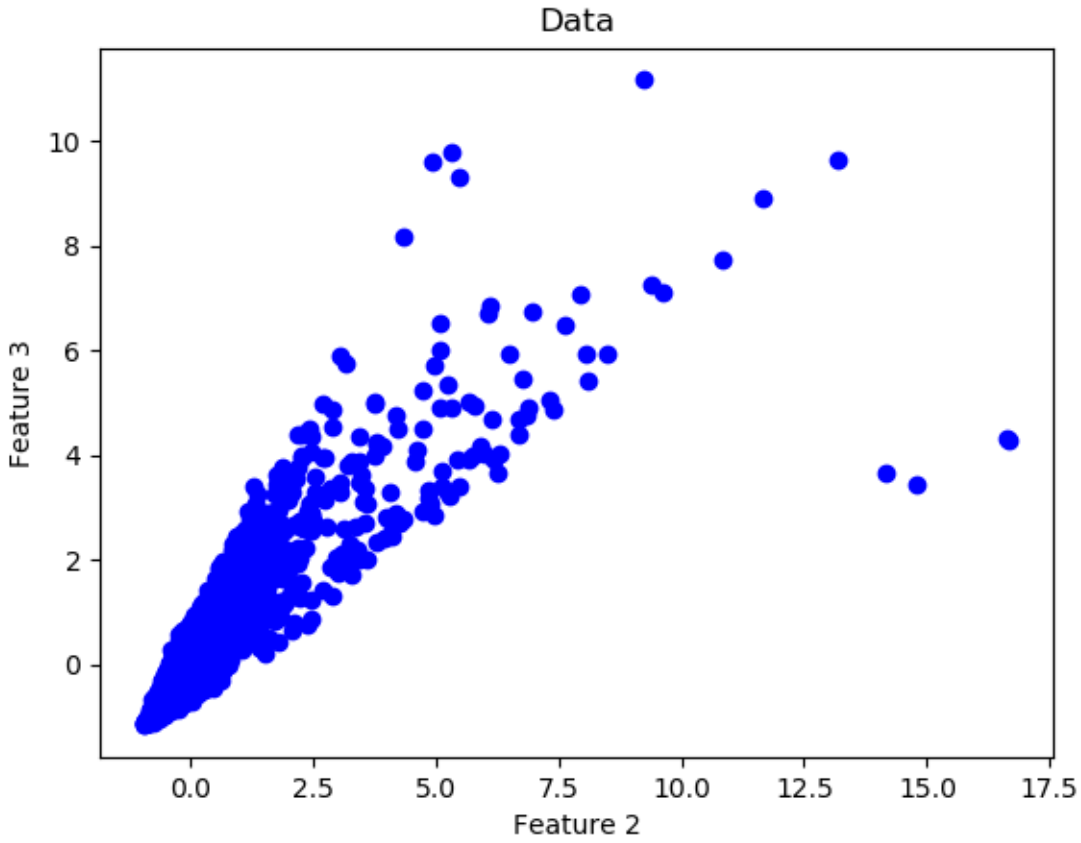
- MSE for general detoriation progression : 1.379
- RMSE for general detoriation progression: 1.174

- **Pearsons correlation for motor detoriation progression : 0.274**, p value 0.095 (correlation = pearsonr(x.ravel(), y2.ravel())
- **Pearsons correlation for general detoriation: 0.310**, p value 0.112(correlation = pearsonr(x.ravel(), y1.ravel())

- Data: patients ill with Parkinson Disease

- Number of input patients: 5875
- **Number of input features: 18**

Scatter Plot for the Predicted Values



Thank you

- Any Queries?