Single Layer Neural Network and Multilayer Neural Network

Single Layer Neural Network:

Neural networks also known as neural nets is a type of algorithm in machine learning and artificial intelligence that works the same as the human brain operates. The artificial neurons in the neural network depict the same behaviour of neurons in the human brain. Neural networks are used in risk analysis of business, forecasting the sales, and many more. Neural networks are adaptable to changing inputs so that there is no need for designing the algorithm again based on the inputs. In this article, we'll discuss single-layered neural network with its syntax and implementation of the neuralnet() function in R programming. Following function requires neuralnet package.

Types of Neural Networks

Neural Networks can be classified into multiple types based on their depth activation filters, Structure, Neurons used, Neuron density, data flow, and so on. The types of Neural Networks are as follows:

1. Perceptron
2. Feed Forward Neural Networks
3. Convolutional Neural Networks
4. Radial Basis Function Neural Networks
5. Recurrent Neural Networks
6. Sequence to Sequence Model
7. Modular Neural Network

Depending upon the number of layers, there are two types of neural networks:

1. Single Layered Neural Network: A single layer neural network contains input and output layer. The input layer receives the input signals and the output layer generates the output signals accordingly.
2. Multilayer Neural Network: Multilayer neural network contains input, output and one or more than one hidden layer. The hidden layers perform intermediate computations before directing the input to the output layer.

Single Layered Neural Network

A single-layered neural network often called perceptrons is a type of feed-forward neural network made up of input and output layers. Inputs provided are multi-dimensional. Perceptrons are acyclic in nature. The sum of the product of weights and the inputs is calculated in each node. The input layer transmits the signals to the output layer. The output layer performs computations. Perceptron can learn only a linear function and requires less training output. The output can be represented in one or two values(0 or 1).

Implementation in R

R language provides neuralnet() function which is available in neuralnet package to perform single layered neural network.

Syntax:

neuralnet(formula, data, hidden)

Parameters:

formula: represents formula on which model has to be fitted

data: represents dataframe

hidden: represents number of neurons in hidden layers

To know about more optional parameters of the function, use below command in console: help(“neuralnet”)

Multi Layered Neural Networks in R Programming

A series or set of algorithms that try to recognize the underlying relationship in a data set through a definite process that mimics the operation of the human brain is known as a Neural Network. Hence, the neural networks could refer to the neurons of the human, either artificial or organic in nature. A neural network can easily adapt to the changing input to achieve or generate the best possible result for the network and does not need to redesign the output criteria.

Types of Neural Network

Neural Networks can be classified into multiple types based on their Layers and depth activation filters, Structure, Neurons used, Neuron density, data flow, and so on. The types of Neural Networks are as follows:

- Perceptron
- Multi-Layer Perceptron or Multi-Layer Neural Network
- Feed Forward Neural Networks
- Convolutional Neural Networks
- Radial Basis Function Neural Networks
- Recurrent Neural Networks
- Sequence to Sequence Model
- Modular Neural Network

Multi-Layer Neural Network

To be accurate a fully connected Multi-Layered Neural Network is known as Multi-Layer Perceptron. A Multi-Layered Neural Network consists of multiple layers of artificial neurons or nodes. Unlike Single-Layer Neural networks, in recent times most

networks have Multi-Layered Neural Network. The following diagram is a visualization of a multi-layer neural network.

Explanation: Here the nodes marked as “1” are known as bias units. The leftmost layer or Layer 1 is the input layer, the middle layer or Layer 2 is the hidden layer and the rightmost layer or Layer 3 is the output layer. It can say that the above diagram has 3 input units (leaving the bias unit), 1 output unit, and 4 hidden units(1 bias unit is not included).

A Multi-layered Neural Network is a typical example of the Feed Forward Neural Network. The number of neurons and the number of layers consists of the hyperparameters of Neural Networks which need tuning. In order to find ideal values for the hyperparameters, one must use some cross-validation techniques. Using the Back-Propagation technique, weight adjustment training is carried out.

Formula for Multi-Layered Neural Network

Suppose we have x_n inputs(x_1, x_2….x_n) and a bias unit. Let the weight applied to be w_1, w_2…..w_n. Then find the summation and bias unit on performing dot product among inputs and weights as:

$$r = \Sigma^{m}_{i=1} w_i x_i + bias$$

On feeding the r into activation function F(r) we find the output for the hidden layers. For the first hidden layer h^1, the neuron can be calculated as:

$$h_1^1 = F(r)$$

For all the other hidden layers repeat the same procedure. Keep repeating the process until reach the last weight set.

Implementing Multi-Layered Neural Network in R

In R Language, install the neuralnet package to work on the concepts of Neural Network. The neuralnet package demands an all-numeric matrix or data frame. Control the hidden layers by mentioning the value against the hidden parameter of the neuralnet() function which can be a vector for many hidden layers. Use the set.seed() function every time to generate random numbers.

Example: Use the neuralnet package in order to fit a linear model. Let us see the steps to fit a Multi-Layered Neural network in R.

Step 1: The first step is to pick the dataset. Here in this example, let’s work on the Boston dataset of the MASS package. This dataset typically deals with the housing values in the fringes or suburbs of Boston. The goal is to find the medv or median values of the houses occupied by its owner by using all the other available continuous variables. Use the set.seed() function to generate random numbers.