Namal Institute Mianwali

Artificial Neural Networks

Muhammad Asad	BSCS-2019-01
Kiran Zafar	BSCS-2019-36

Abstract:

A training model is a dataset used to train a machine learning algorithm. It is made up of sample output data as well as the equivalent sets of input data that have an impact on the outcome. The training model is used to process the input data via the algorithm in order to compare the processed output to the sample output. The model is modified based on the results of this association.

"Model fitting" is the term for this iterative procedure. The precision of the model is dependent on the correctness of the training or validation dataset.

Firstly, this report explains the basic neural network and then it describes how an artificial neural network is formed. Secondly, it clearly explains that how we can train our machine to recognize hand written Roman numerals and then check its validation that whether it recognizes the alphabets properly or not by doing experiments.

Table of Contents:

Contents

1. Problem Statement:

We are given with a hand-written Roman numerals (from 1 to 10) dataset, we have to design and train the best Multi-layer Perceptron (using scikitLearn library) that reports the best training and validation accuracies.

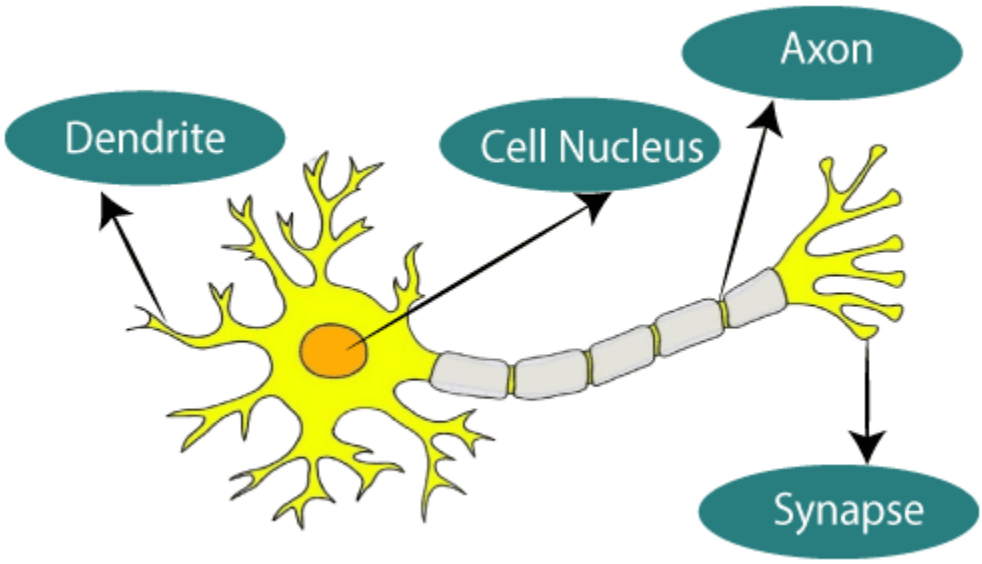
2. What is Artificial Neural Network (ANN)?

The term "artificial neural network" refers to a sub-field of artificial intelligence influenced by biology and fashioned after the brain. A computational network based on biological neural networks that construct the structure of the human brain is known as an artificial neural network. Artificial neural networks, like human brains, have neurons that are coupled to each other in various layers of the networks. Nodes are the name for these neurons.

2.1 Basic structure of an ANN:

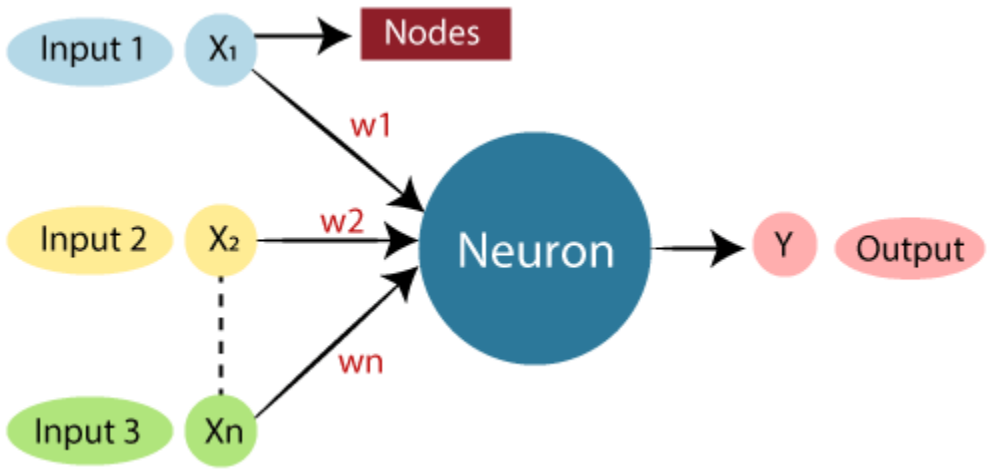
The premise behind ANNs is that by forming the appropriate connections, the human brain may be replicated using silicon and wires as real neurons and dendrites.

The human brain is made up of 86 billion neurons, which are nerve cells. Axons connect them to thousands of other cells. Dendrites accept stimuli from the outside world as well as inputs from sensory organs. Electric impulses are generated as a result of these inputs, and they travel fast across the neural network. A neural network of brain is shown below:



ANNs are made up of numerous nodes that resemble actual neurons in the brain. The neurons are linked together and interact with one another. The nodes can accept data as input and perform simple operations on it. These operations produce a result that is passed on to other neurons. Each node's output is referred to as its activation or node value.

Each link has a weight connected with it. ANNs can learn by changing their weight values, The following illustration shows a simple ANN.



The relationship between a biological and an artificial neural network is as follows:

Biological Neural Network	Artificial Neural Network
Dendrites	Inputs
Cell Nucleus	Nodes
Synapse	Weights
Axon	Output

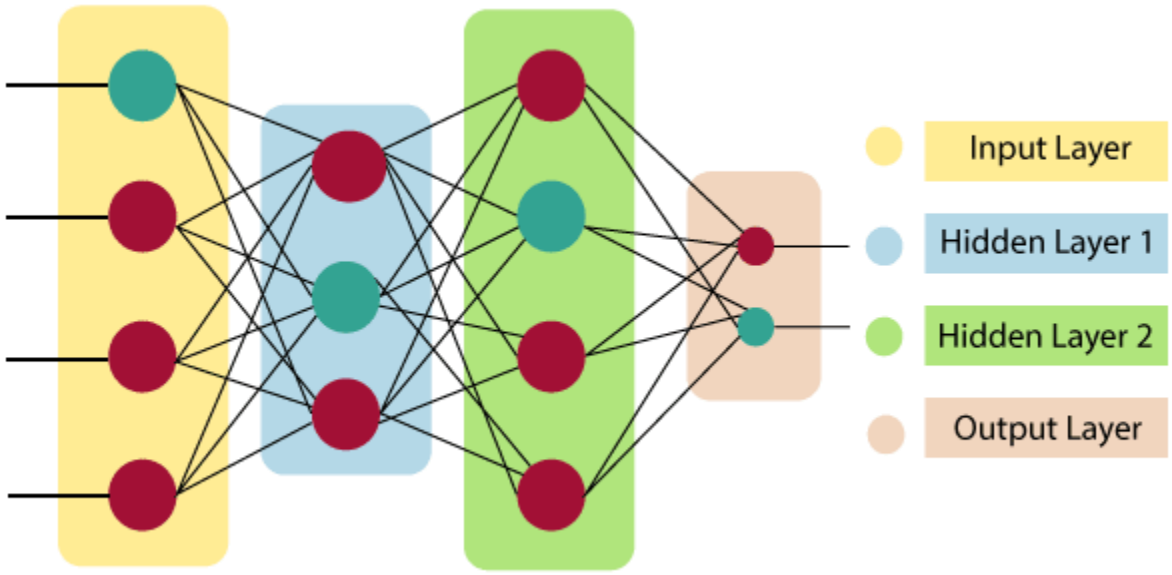
Consider the example of a digital logic gate that receives an input and produces an output to better understand the artificial neural network. The "OR" gate accepts two inputs. If one or both inputs are "On," the output will be "On." If both inputs are

"Off," the output will be "Off." In this case, the output is determined by the input. Our brains don't work in the same way. Because our brain's neurons are "learning," the outputs to inputs connection is constantly changing.

2.2 The architecture of an ANN:

To know the concept of artificial neural network architecture, we must first learn what a neural network is made up of. To describe a neural network that is made up of a large number of artificial neurons called units that are stacked in layers. Let's take a look at the various layers that may be found in an artificial neural network.

The Artificial Neural Network is made up of three layers:



2.2.1 Input layer:

It accepts inputs in a variety of formats specified by the programmer, as the name implies.

2.2.2 Hidden Layer:

Between the input and output layers is a hidden layer. It does all the maths to uncover hidden features and patterns.

2.2.3 Output layer:

Via the hidden layer, the input is transformed into output, which is then conveyed using this layer.

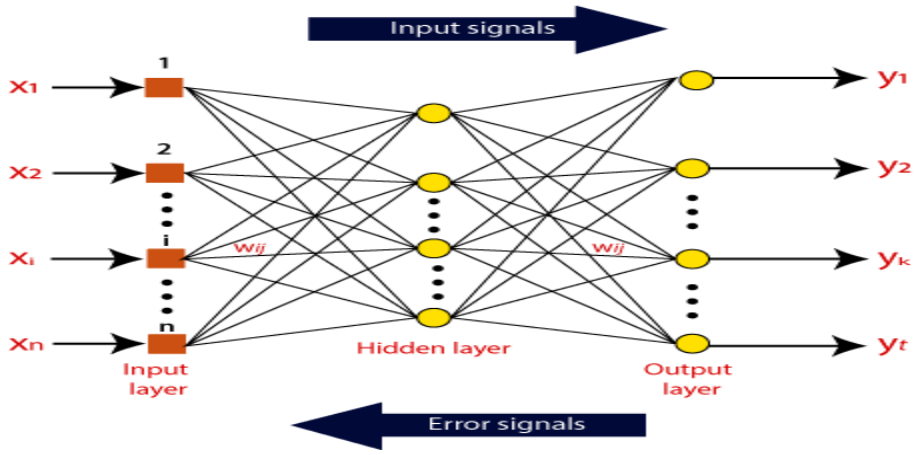
The weighted total of the inputs is computed by the artificial neural network, which incorporates a bias. The transfer function is used to express the computation.

$$\sum_{i=1}^{n} Wi * Xi + b$$

To produce the output, the weighted total is supplied as an input to an activation function. Activation functions determine whether or not a node should fire. Those who are fired are the only ones who make it to the output layer. There are several activation functions that can be used depending on the type of task we're doing.

2.3 How do artificial neural networks work?

The artificial neurons form the nodes of an artificial neural network, which is best described as a weighted directed graph. The directed edges with weights represent the relationship between neuron outputs and neuron inputs. The Artificial Neural Network gets the input signal in the form of a pattern and a picture in the form of a vector from an external source. The notations x(n) are then used to mathematically allocate these inputs for each n number of inputs.



After that, each input is multiplied by the weights assigned to it (these weights are the details utilised by the artificial neural networks to solve a specific problem). These weights usually describe the strength of the connections between neurons inside the artificial neural network in broad terms. Within the processing unit, all of the weighted inputs are summed.

If the weighted total is 0, bias is used to make the output non-zero or something else is used to scale up to the system's response. Bias has the same input as weight, and both are equal to one. The total of weighted inputs can be anything from 0 to positive infinity in this case. A specified maximum value is benchmarked, and the total of weighted inputs is processed through the activation function to keep the response within the bounds of the desired value.

The set of transfer functions employed to achieve the desired output is referred to as the activation function. There are several types of activation functions, but the most common are linear or non-linear sets of functions. The Binary, linear, and Tan hyperbolic sigmoidal activation functions are some of the most often utilized sets of activation functions. Let's take a closer look at each of them:

2.3.1 Binary:

The output of a binary activation function is either a one or a zero. A threshold value has been set up to accomplish this. If the net weighted input of neurons is more than one, the activation function's ultimate output is one; otherwise, the output is zero.

2.3.2 Sigmoidal Hyperbolic:

The Sigmoidal Hyperbola function is commonly referred to as a "S" curve. To approximate output from the actual net input, the tan hyperbolic method is applied. The following is the definition of the function:

F(x) = (1/1 + exp (-????x))

Where “????” is considered the Steepness parameter.

3. Functions:

The following functions are used:

3.1 MLPClassifier:

Multiclass classification can be done using Logistic Regression’s one-vs-rest technique, which allows you to specify the numerical solver, which is set to an acceptable regularization strength by default. Multiclass classification can also be done with MLPClassifier. A feed forward artificial neural network model, the

multilayer perceptron (MLP), maps input data sets to a collection of acceptable outputs. An MLP is made up of numerous layers, each of which is fully connected to the one before it. Except for the nodes of the input layer, the nodes of the layers are neurons with nonlinear activation functions. One or more nonlinear hidden layers may exist between the input and output layers.

Parameters:

- ***hidden_layer_sizes:*** This option allows us to determine the number of layers and nodes that the Neural Network Classifier should have. The number of nodes at the ith position is represented by each element in the tuple, where I is the tuple's index. As a result, the length of the tuple represents the number of hidden layers in the neural network.
- ***max_iter:*** The number of epochs is indicated.
- ***activation:*** The activation function for the hidden layers.
- ***learning_rate_init:*** The initial learning rate used which is 0.001 by default. It controls the step-size in updating the weights.

3.1.1 .fit () function:

It maps the parameters given in MLPClassifier on y_train for training data.

3.1.2 .predict () function:

It predicts the accuracy of trained data.

3.2 Matplotlib:

Matplotlib is a Python package that allows you to create static, animated, and interactive visualizations. Matplotlib makes simple things simple and difficult things possible. In our project matplotlib is used to read images.

3.3 OS:

It is used for images path as this function gives us the last part of the path which may be a folder or a file name

4. Experiments:

4.1 Experiment 01:

In our first experiment, we counted the dark pixels of training images and trained on desired output using MLPClassifier function of sklearn. We iterated it for 200 iterations and on each iteration, the learning loss was *2.1+*. The overall accuracy

of trained data with x_train was 18.2%. We used the following filters of MLPClassifier function:

hidden_layer_sizes = (10, 10, 10)
activation = 'logistic'
learning_rate_init = 0.001
random_state = 1
verbose = True
max_iter = 200
n_iter_no_change = 200

4.2 Experiment 02:

In this experiment, we have cropped 20x20 image from the center of an image (as size of each input/image is different) which is converted to 2D array. Then, we converted this array into one single row (list). After reshaping we have given this array into .fit () function parameter for training which resulted into accuracy of 86% which is a best experiment with lowest loss. Filters of MLPClassifier for this experiment are given below:

hidden_layer_sizes = (500, 200, 100),
activation = 'logistic',
learning_rate_init = 0.001,
random_state = 1,
verbose = True,
max_iter = 500,
n_iter_no_change = 500

5. Conclusion:

From the above experiments, we can conclude that the first experiment was not satisfactory as it was giving overall accuracy of just 18.2 % that was the more loss in training data. In second experiment, by changing logic some parameters of MLPClassifier, we came up with the accuracy of 86% with lowest loss which we can say the best experiment.