Report

Artificial Neural Networks

Amit Pingale - NUID 001898697

Nitesh Machireddy - NUID 001893245

Artificial Neural Networks

Artificial neural nets mimic the way the biological brain works and tries to teach a program how to recognize and memorize patterns and classify a dataset. ANNs are the most popular way of achieving machine learning. In this particular project, we are using a neural network with two hidden layers to classify images of handwritten digits into their respective digits.

Perceptron

Perceptron is a single layer neural network and a multi-layer perceptron is called Neural Networks. Perceptron is a linear classifier (binary).

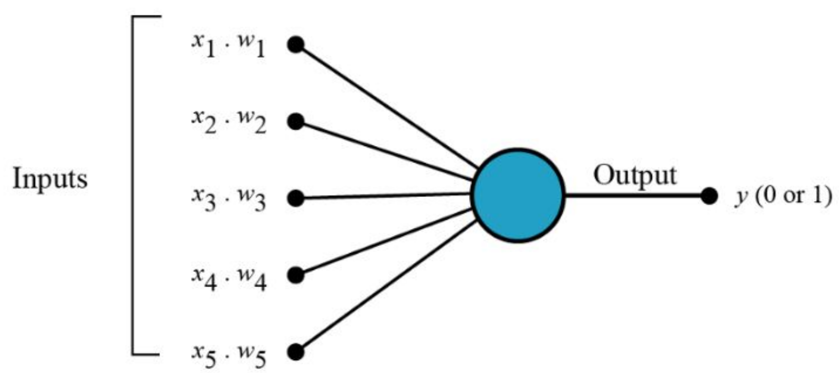
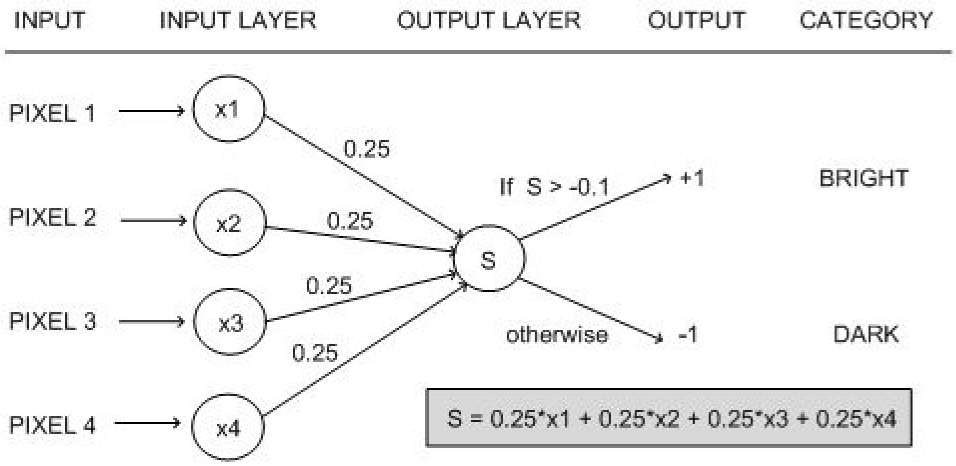


Fig: Multiplying inputs with weights for 5 inputs

The perceptron consists of 4 parts

1. Input values or One input layer
2. Weights and Bias
3. Net sum
4. Activation Function

Working of Perceptron



Activation of Perceptron

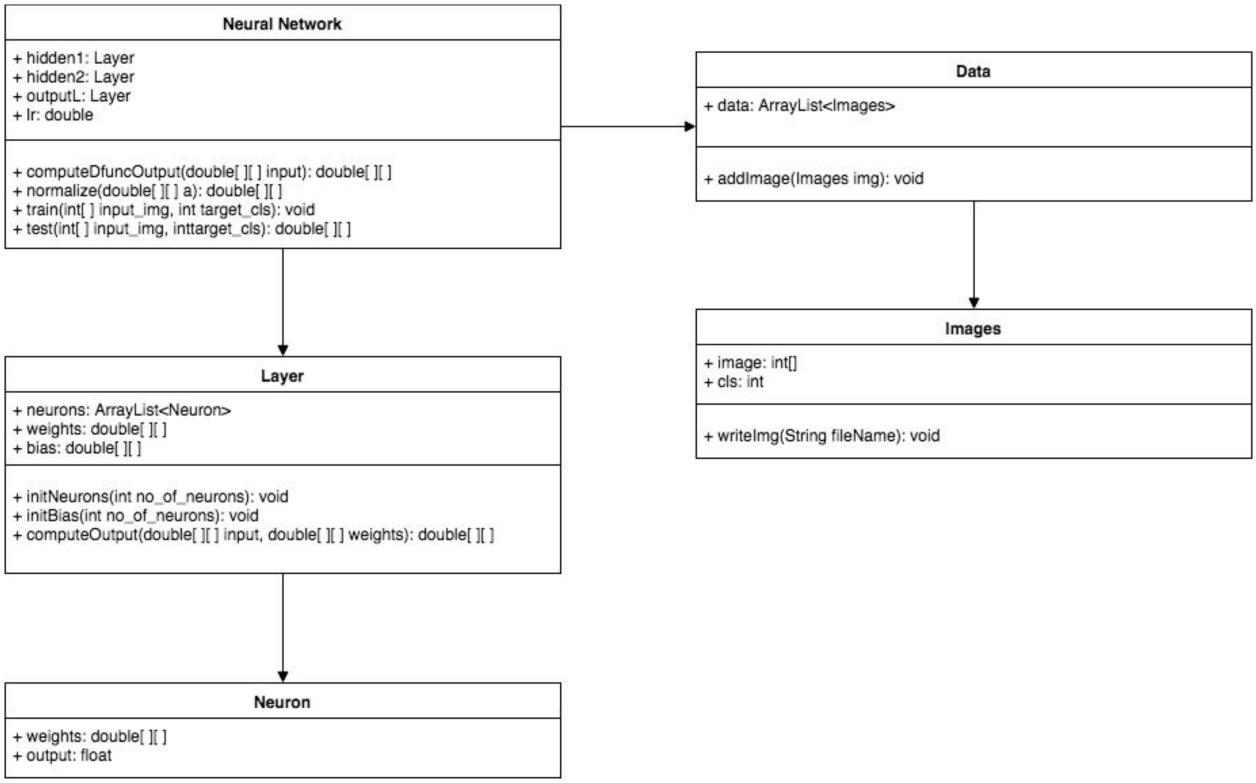
The activation function of a perceptron is used to calculate the output of the neuron considering the inputs and weights.

In our case, it is Sigmoid function

S from the image above is added to the bias b and the output of the perceptron is calculated by computing the sigmoid of the sum

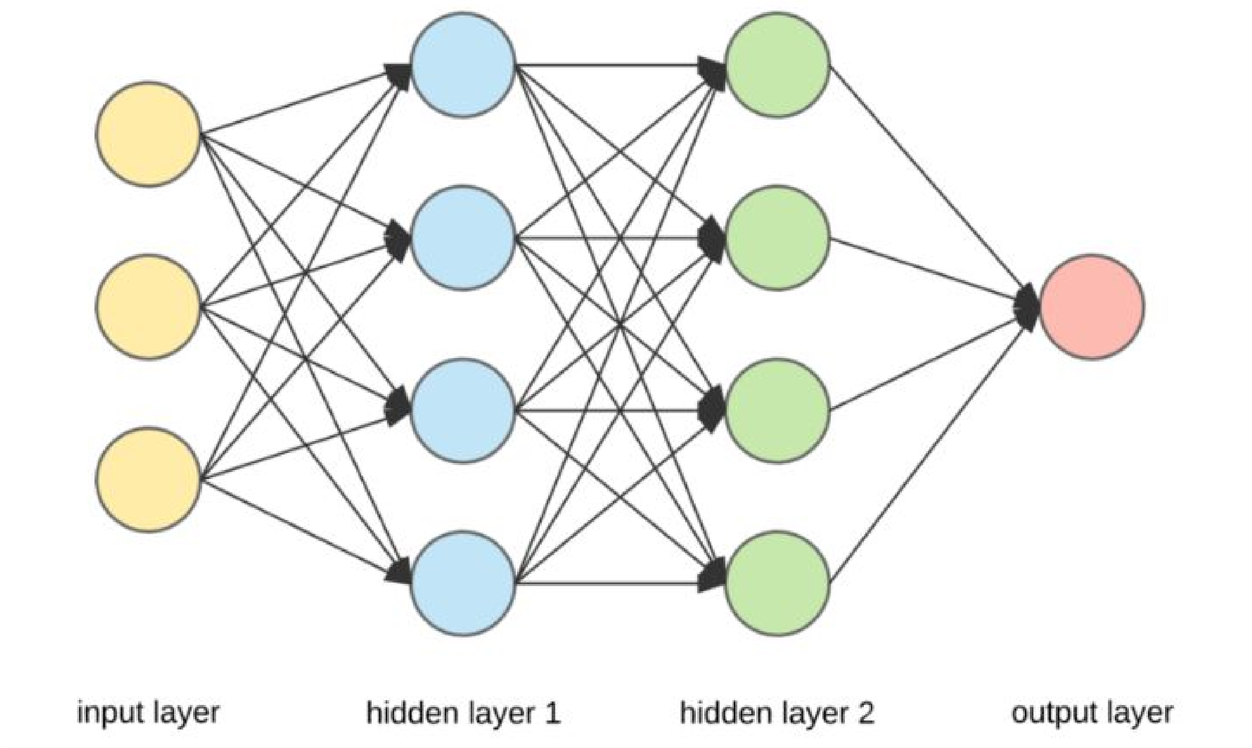
Output = Sigmoid(S+b)

Neural Network Architecture



Training Neural Net

Our neural Network has Four layers including input



Input → Hidden1 → Hidden2 → Output

Input layer has 783 inputs

Hidden 1 has 300 Neurons

Hidden 2 has 100 Neurons

Output has 10 Neurons

Back Propagation

Backpropagation is the process of figuring out by how much the weights are to be adjusted in order to minimize the error.

Back propagation is done in three steps in our project

1. Compute error
2. Compute Gradient
3. Adjust weights

Compute Error:

Error for output layer is calculated by

E = (Target output - actual output)

But for the hidden layers, these errors would be redistributed to all the neurons as per the weights

That is,

$E_H = (W_{HO})^\wedge * E_O$

$(W_{HO})^\wedge$ is the transpose of weight matrix from hidden to output

E_O is the error at output layer

Compute Gradient:

Gradient is computed by the backpropagation formula

G_H = dSigmoid(Outpus) *E_O*lr

G_H is the gradient at hidden layer

dSigmoid is the derivative of sigmoid which is sigmoid*(1-sigmoid)

E_O is the output error

lr is the learning rate

Adjust Weights:

All the weights are then adjusted by a product of **gradient and previous layer outputs**

The code of this process is shown below:

```
//Compute error
double[][] outErrors = Matrix.subtract(targets, outOutputs);

//Compute Gradients
double[][] gradients = computeDfuncOutput(outOutputs);
gradients = Matrix.elemMultiply(gradients, outErrors);
gradients = Matrix.scalarMultiply(gradients, lr);

//Compute deltas
double[][] hidden2_T = Matrix.transpose(h2outputs);
double[][] weight_h20_deltas = Matrix.multiply(gradients, hidden2_T);

//Adjust weights
outputL.setWeights(Matrix.add(weight_h20_deltas, outputL.getWeights()));
outputL.setBias(Matrix.add(outputL.getBias(), gradients));
```

This process is repeated for all layers and this training process is run for all the available data entries. Thus we end up with the actual weights of the network that achieves the minimum error.

Dataset used for training Neural Network

The MNIST dataset is a collection of pixel values of handwritten digits. The dataset comprises of 42,000 28x28 pixel images. Total pixel values in an image is 783 which is saved in .csv file.

The data set is divided in the ratio of 9:1 for training and testing purpose.

- Number of images used for training = 38,000
- Number of images used for testing = 4,000

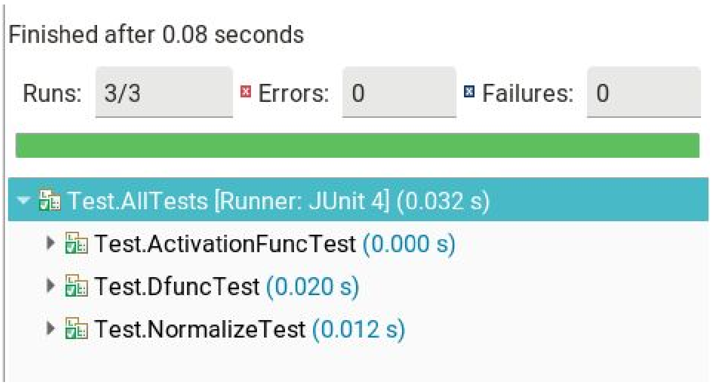
Results

The network consistently achieves about **80% or more accuracy** while recognizing the hand written images.

The confusion matrix looks shown below.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	367	0	1	1	0	52	2	1	4	2
1	0	399	1	2	0	22	1	2	15	0
2	4	6	254	14	6	57	3	3	24	1
3	0	0	6	324	0	78	3	1	5	4
4	2	0	3	1	319	59	1	0	7	30
5	4	0	0	7	3	306	5	1	1	7
6	3	1	8	1	7	33	337	0	8	2
7	1	1	5	2	4	11	1	335	4	37
8	2	8	2	8	2	101	2	0	246	8
9	4	0	1	1	15	48	1	12	3	314

Test Cases


Finished after 0.08 seconds
Runs: 3/3
Errors: 0
Failures: 0
Test.AllTests [Runner: JUnit 4] (0.032 s)
Test.ActivationFuncTest (0.000 s)
Test.DfuncTest (0.020 s)
Test.NormalizeTest (0.012 s)