Neural Network

Supervised by: Shivan Shkwr

Done by:

- Abdulrahman Tawffeq
- Muhammed Salahadin

Introduction

Inspired by the actual biological neural network, the biological brain, something called an artificial neural network can be built as a kind of computational system. So, what is the neuron in the code? How does it receive inputs? How does it generate outputs?

Our brain does this. It receives all these inputs from light in the room that travel through our retina and into the brain and the signals then produce outputs that allow us to catch something or read some words. How can that process be simulated in software? And what types of outcomes can we generate?

So, we need to build and understand the perceptron. Then we can start to connect the perceptrons together to create more sophisticated systems that can begin to generate outputs based on more complex inputs. This idea of having some data that we want to make sense of. That data is an input to a machine learning algorithm. That algorithm is going to generate an output. So maybe the data is an image that a machine learning algorithm is going to guess is a cat or a dog? Or to be something else according to the type of the problem.

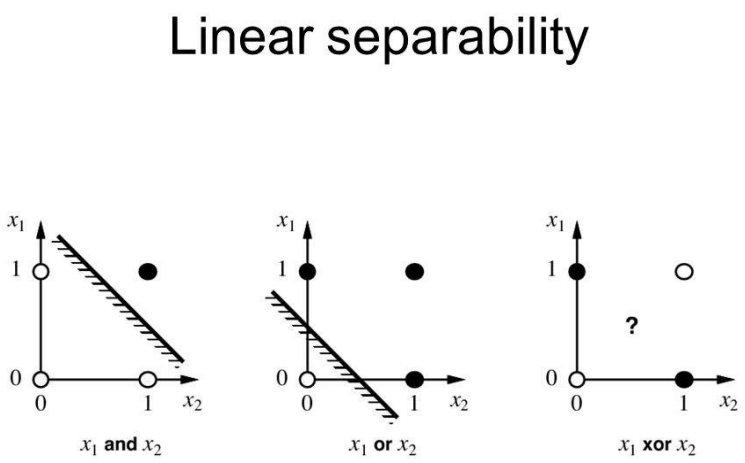
The Neuron

Simplest NN UML

Neuron
static learningRate = 0.1; - weights [ ]; - inputs [ ]; - error : floar;
+ constructor(numOfInputs: int); + setWeights(numOfInputs: int); + output(inputList [ ]): returns float; + train(inputList [ ], target: float) + static sigmoid(number: float): returns float + static error(target: flaot, output: float): returns float - dErrorOutput(target, output) - dOutputInput(output): returns float - dInputWeight(weightIndex, inputs): returns float - dErrorWeight(inputs, output, target, weightIndex): returns float + getWeights(): returns weights [ ] + getInputs(): returns inputs [ ]; + getError(): returns error float

But, the neural network that contains only one neuron is only helpful for the linearly separable problems like (And - Or - xor) gate. We need the multilayer neural network to be able to solve complex problems. So, we need to create the neural network class that will be using the neuron class inside its layers.

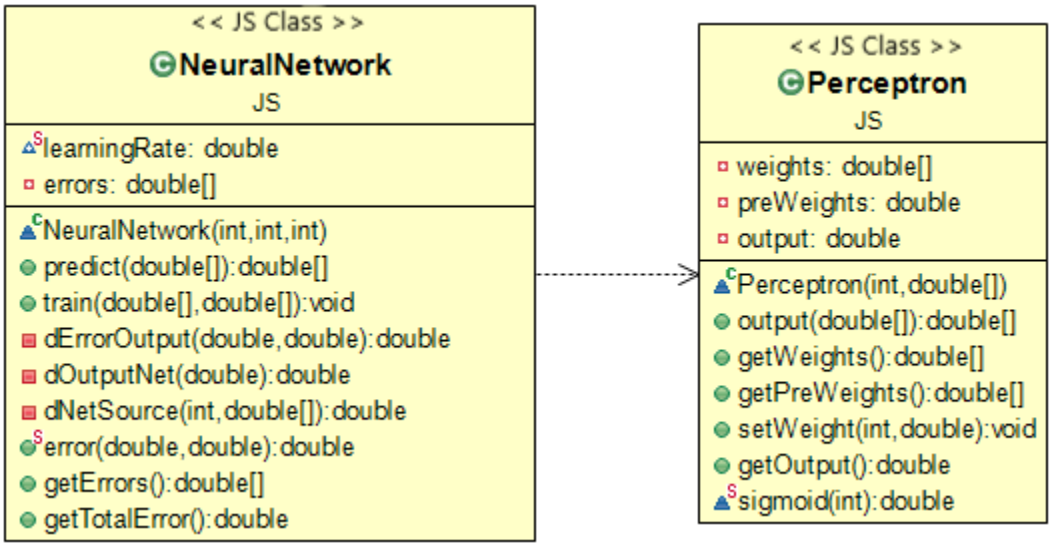
Examples:



Here the xor needs two lines to be solved, which means at least two neurons in the network.

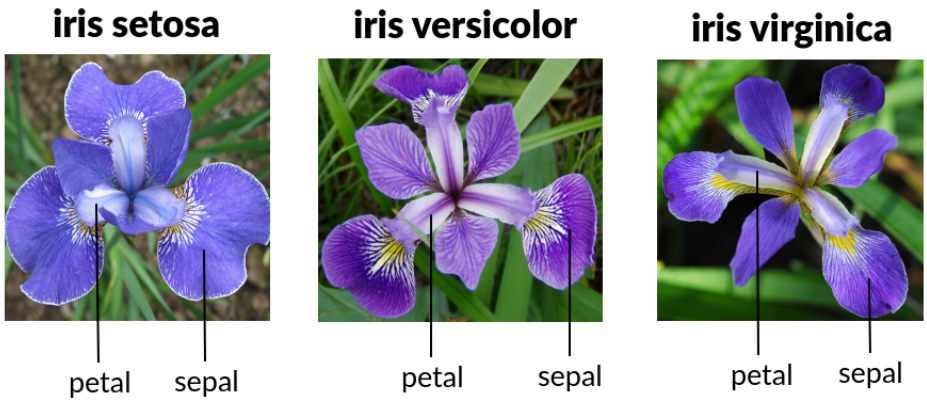
The number of neurons in the hidden layer could be specified by trial and error according to the complexity of the problem.

Multilayer NN UML



Iris Flower Type Prediction

We built the neural network and used a past data set that has information about the sepal and petal width and height of the iris flower. According to that, the type of iris flower will be Setosa, Virginica, or Visicolor.



The Neural Network Structure

Fully connects NN:

- Input Layer of 4 inputs => sepal.length sepal.width petal.length petal.width
- Hidden Layer of 4 Perceptrons based on trial and error
- Output Layer of 3 Perceptrons as we have three possible outputs => Setosa, Virginica, or Visicolor

Because the perceptron output ranges from 0 to 1 (Segmoid is used), we did data preprocessing to change the types to numbers:

- "Virginica": 0
- "Versicolor": 1

- "Setosa": 2

The output of the NN will be an array of 3 numbers, each from 0 to 1. We round the numbers to have an array of zeros and one, then the index of that one the flower type predicted by the machine.

Ex: [0, 0, 1] => index of 1 is 2 and 2 is Setos

Training the Neural Network

We separated the data randomly into training and testing sets. We train the neural network on that data set for 1000 epochs to get suitable results. Each epoch will train the NN of all rows in the training set (105 data).

After training the neural network on the data provided, the error was reduced and we saved the weights in a model to be used later without training it again.

Results

Iris Flower Type Prediction

Sepal Length

6.7

Sepal Width

3

Petal Length

5

Petal Width

1.7

Predict

Flower Type: Versicolor

So, the prediction is right for this test data that the machine did not train on it before ([6.7, 3, 5, 1.7], [1]), 1 is Versicolor