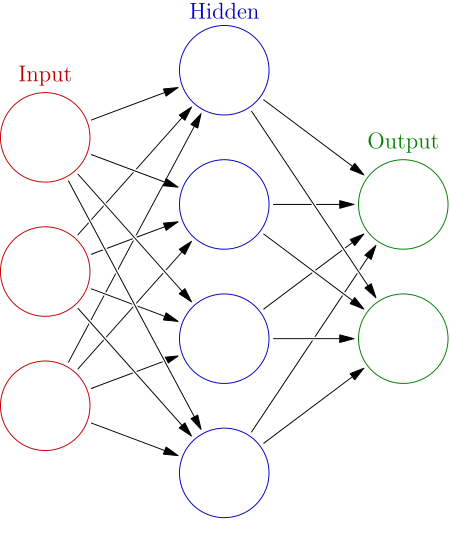
**PART I – Neural Networks**

**Intro**

A Neural Network is a machine learning model loosely based on the structure of interconnected neurons in the human brain. Since the biological brain is well-adapted to handling complex tasks, the inspiration for Artificial Neural Networks is that a model that mimics the brain’s structure and basic funcitoning would prove most capable at solving difficult machine learning problems.

Structurally, a Neural Network is a collection of interconnected units called Perceptrons. The perceptrons are arranged in different layers and the data flows from one layer to the next. 

In the most common type of neural network, all of the perceptrons in a given layer receive inputs from all the perceptrons in the previous layer. After receiving these inputs, the perceptron computes its value as follows:

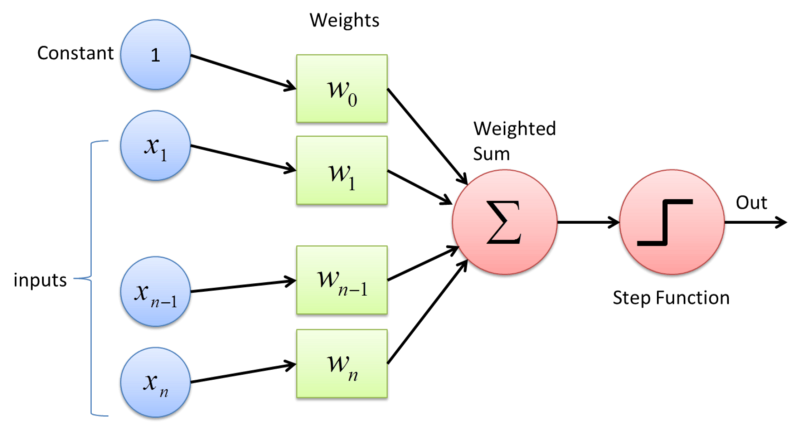
Let X = [x1, x2, x3..] be the inputs received from the previous layer.

[W1, w2, w3…] are the weights of the particular perceptron

Z = X.W + b = x1\*w1 + x2\*w2 + … Calculates the weighted sum of the inputs and adds a bias

H = sigmoid(z) = 1/1+exp(-z) [[use LaTeX here]]

An activation function is used to introduce non-linearity



**Functioning of a Neural Network**

**Vectorization**

In a neural network, the collection of weights of every unit in a layer are vectorized into a matrix [[**Th**]], in which each row holds all the weights of a single perceptron.

The values of all the units in a given layer are handled together as a vector

The biases in a given layer are also handled together as a vector.

The inputs and outputs are given and received in the form of a vector

**Notation**

L : the number of layers in the network

**Zij** : the weighted sum of the jth unit in layer I [[layer number is a superscript ]]

**Aij** : the activated value of the jth unit in layer i.

Superscript (m) : concerned with the mth example in the dataset.

[[**Th**]]: The matrices holding the weights between each layer

**Forward Propagation**

Forward propagation uses the parameters of the network to calculate an output vector from an input vector, by stepping through the computations layer by layer.

Given an input vector X:

Starting with A1 = X,

Repeat: l = 2 to L

A(l-1) = [1; A(l-1 )] Prepend 1 for bias

Z(l) = Theta(l-1)\*A(l-1) Comute the weighted sum Z

A(l) = sigmoid(Z(l)) Apply the activation function

Ends with output Y = A(L)

**Cost Function and Optimization**

Although many cost functions are possible, the squared error cost function is most common.

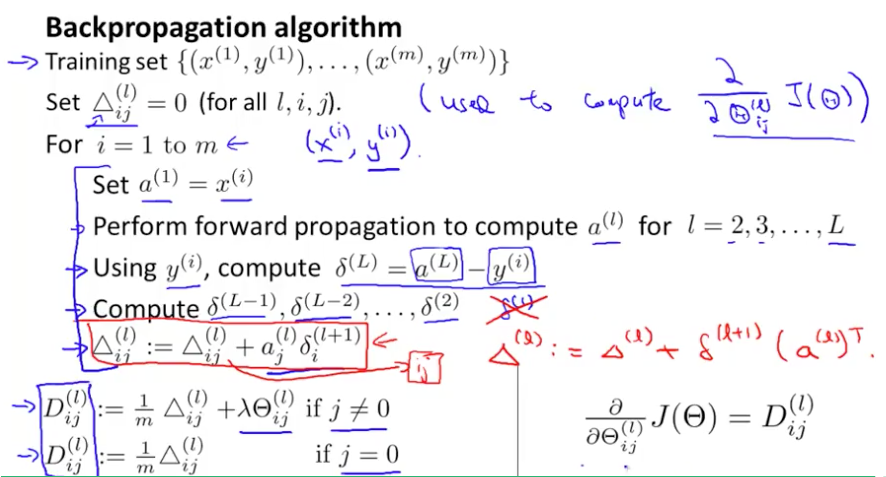
J = **Sigma**(j over all units) (a(L)(j) – Y(j) ) [[**^]]** 2

Training the neural network is the process of reducing this cost function by adjusting the parametrs **Th**. We use gradient descent for this purpose. Therefore, the gradient of the cost function is computed w.r.t every parameter **Th**(l, I, j) and we use the update step:

Th(lij) = Th(lij) – alpha\*Doa/doa(Th(lij)) J(Th)

The process of gradient descent is applied to a neural network as per the backpropagation algorithm.

**The Backpropagation Algorithm**



**One-Hot**

To build a Neural Network for multi-class classification, the labels of the data needs to be encoded into a format that the network can output. Most commonly, the data are one-hot coded. This implies that, for an n-class problem, each label is a n-dimesnional vector, with a 1 in the place of the correct output class and 0’s everywhere else.

For the Iris Data Set, this coding looks like:

‘Iris-setosa’ [1 0 0]

‘Iris-virginica’ [0 1 0]

‘Iris-versicolor’ [0 0 1]

**Experiment IIa**

**Objective**

To build a Neural Network that classifies the Iris Data Set, the shape of the Neural Network has to be selected. Find the right shape of the network for

* Case 1: 1 Hidden Layer (3 Layer Network)
* Case 2: 2 Hidden Layers (4 Layer Network)

… by varying the hidden layer size and choosing the one with highest classification accuracy.

**Procedure**

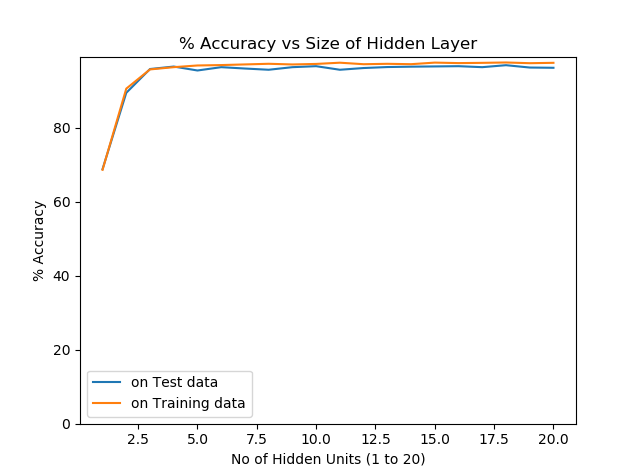
* An ANN class was created that handles Neural Networks of any shape. It wrapped up the functions of parameter initialization, Forward Propagation and Backpropagation.
* Created a function *basic\_fit()* that uses ANN class to train on the Iris Data Set, with 1000 iterations of backpropagation.
* Created a function *rept()* that, with a fixed shape of the hidden layer, trains 100 networks on the data set(using *basic\_run()*) and returns the average accuracy for the 100 trained networks. This step ensured that the results for a given shape are consistent across many runs.
* For Case 1 with one hidden layer, a function *linsearch()* was created which varies the size of this layer from 1 to 20 and gathers the metrics for each shape using *rept().*

For Case 2 with two hidden layers, a function *quadsearch()* was created which varies the sizes of the layers from 1 to 10 each and gathers the metrics for each shape using rept()

* Comparing the values of accuracy (on test data), the best Neural Network shape was chosen in each case.

**Results and Analysis**

For Case 1, where a 3 layered Neural Network was used, the following graph shows the variation of accuracy (on test data and training data) vs number of units in the hidden layer:



The network shape with the highest test accuracy was chosen.

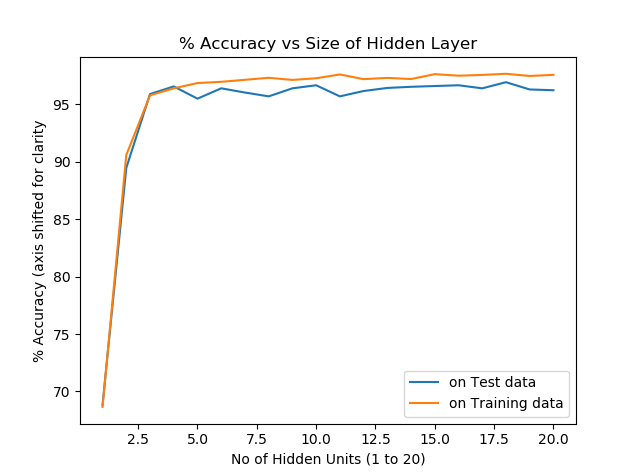
Highest Accuracy: 96.93% Hidden Layer Size: 18

Confusion Matrix: [10 0 0]

[0 9.69 0.82] *Decimal values are a result of averaging over 100 runs.*

[0 0.31 9.18]

The accuracy graph was redrawn with axes shifted, to better understand the trends with clarity.



**Observations Drawn**

* If we had chosen the H.L. size based on training accuracy, we would still get 18 units.
  + This indicates that a H.L. size of 18 is ideally suited to the problem since it outperforms other shapes in fitting to the data (evidenced by Training accuracy) and generalizing to test data.
* For the smallest hidden layers, the accuracy is relatively very less.
  + This indicates that these networks are underfitting the data.
* For higher values of H.L. size, we being to see testing accuracy fall below training accuracy. (although only slightly here)
  + This indicates that these Networks are overfitting the data

**Conclusions**

* For a given problem, there exists some shape of Neural Network that performs the best. It is worthwhile to search and find the right shape.
* If the Network is too complex for the problem, it might overfit the data to some degree.
* If the network is not complex enough to learn the patterns in the data, accuracy will be very low and the network underfits.