**PROJECT STATUS**

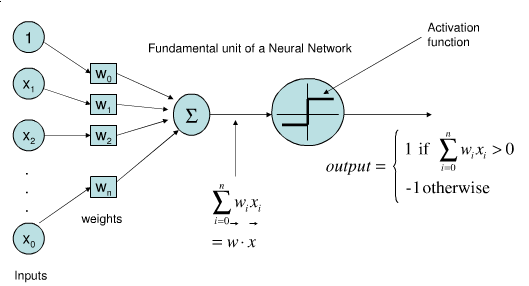
**Problem Statement:**

Gender Recognition system from audio files using FFT with Artificial Neural Networks.

**Status:**

**6. Training**

Using Artifcial Neural Network to train the model for classification problem.

  
Illustration 1: Artificial Neural Network

The description of the architecture i am using for classification are as follows:

1. 1 input layer , Hidden layer, 1 Output Layer.

2. using 12 input attributes and 1 output attributes.

3. activation function for input and hidden layer is “relu”.

4. activation function for output layer is “sigmoid”.

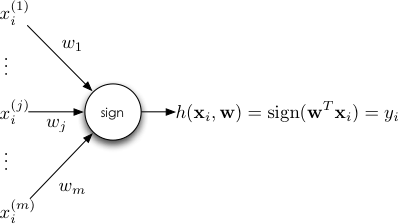
5. using 10 perceptron in each input and hidden layer.

6. for optimisation algorthm is used “adam”.

**Q. What is Perceptron?**

**Ans :**

a computer model or computerized machine devised to represent or simulate the ability of the brain to recognize and discriminate.



**Q.What is activation Function?**

**Ans :**

In artificial neural networks, the activation function of a node defines the output of that node given an input or set of inputs. A standard computer chip circuit can be seen as a digital network of activation functions that can be "ON" (1) or "OFF" (0), depending on input.

**ReLu activation Function:**

A unit employing the rectifier is also called a rectified linear unit (ReLU).

A smooth approximation to the rectifier is the analytic function

 **F(x) = Log(1 + exp(x))**

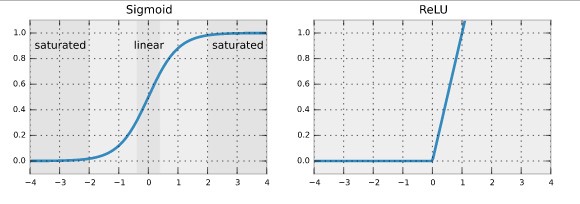
which is called the **softplus** function

**Sigmoid activaton Function:**

It is used in neural networks to give logistic neurons real-valued output that is a smooth and bounded function of their total input. It also has the added benefit of having nice derivatives which make learning the weights of a neural network easier.

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

**Graphical Representation**



**Implementation**

**Modules used:**

import numpy as np

import pandas as pd

import h5py

from sklearn.model\_selection import train\_test\_split

from keras.models import load\_model

from keras.models import Sequential

from keras.layers import Dense, Dropout

from keras import optimizers

from sklearn.utils import shuffle

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix

**Using python keras module for layer creation and compilation:**

# Initialising the ANN

classifier = Sequential()

# Adding the input layer and the first hidden layer

classifier.add(Dense(output\_dim = 10, init = 'uniform', activation = 'relu', input\_dim = 12))

# Adding the second hidden layer

classifier.add(Dense(output\_dim = 10, init = 'uniform', activation = 'relu'))

# Adding the third hidden layer

classifier.add(Dense(output\_dim = 10, init = 'uniform', activation = 'relu'))

# Adding the output layer

classifier.add(Dense(output\_dim = 1, init = 'uniform', activation = 'sigmoid'))

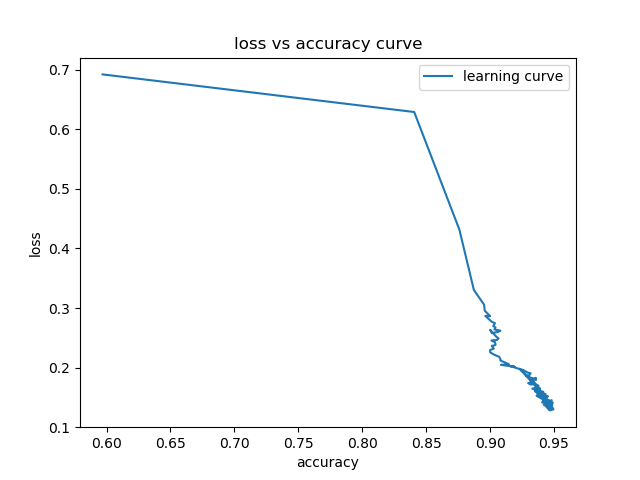
# Compiling the ANN

classifier.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

# Fitting the ANN to the Training set

history = classifier.fit(x\_train, y\_train, batch\_size = 20, nb\_epoch = 200)

**Learnng Curve:**

****

**7. Testing and Evaluation**

**Q.What is Confusion Matrix?**

**Ans :**

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

For the above Model :

cm =