**A PROJECT REPORT**

**ON**

**“PREDICATE NUMBER USING ANN”**

**Submitted to**

**UNIVERSITY OF Mumbai**

**MADE BY**

**SHAVEJ SHAIKH ROLL NUMBER 49**

**SHAHNAWAZ SHAIKH ROLL NUMBER 50**



**DEPARTMENT OF COMPUTER ENGINEERING**

**WATUMULL COLLEGE OF ENGINEERING**

**UNIVERSITY OF MUMBAI**

Department of Computer Engineering

Watumull Institute

University of Mumbai

2018 - 2019

**MINI PROJECT**

## MINI PROJECT NAME

Submitted in partial requirement for the degree of Bachelor of Engineering in Computer Engineering by,

# SHAVEJ SHAIKH(49)

# SHAHNAWAZ SHAIKH(50)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# NAME OF PROJECT GUIDE

**(PROJECT GUIDE)**

### Examiners

#### \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**(Head of the Department)**

**Date:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Dr. Sunita Sharma**

**(Principal)**

**Date:**

**ABSTRACT:**

In this project, we explain our approach to the problem in detail , including the definitions and the mechanism of various algorithms used to model the scores in order to predict the output from given inputs and to find out the best algorithm .We used supervised algorithms in order to predict the results.

          Each row is a training example of three inputs and one output the neural network to be able to predict what the output should be for situation.

**Content**

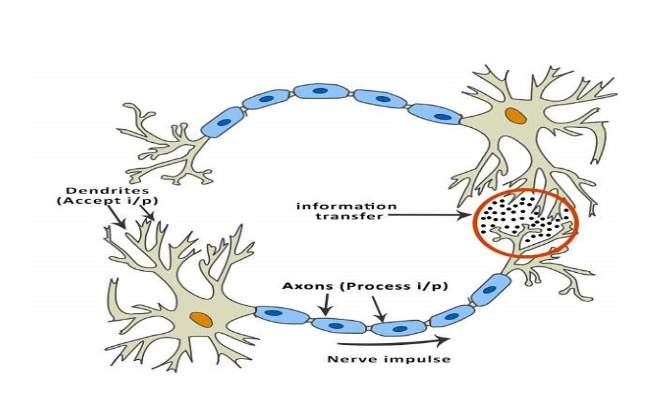
1. **UNDERSTANDING ARTIFICIAL NEURAL NETWORK**
   1. **What is neural network? 4**
   2. **Application of neural Network. 5**
2. **PERCEPTRON**
   1. **What is perceptron? 7**
   2. **Steps of perceptron 8**
3. **TRAINING** 
   1. **Adjusting of weight 10**
4. **SIGMOID NEURON**
   1. **Sigmoid function 11**
5. **BACKPROPAGATION**
   1. **Backpropagation 12**
6. **CONCLUSION**
   1. **Conclusion 13**

**REFERENCES 14**

**CHAPTER 1**

1.1 What is ANNs?

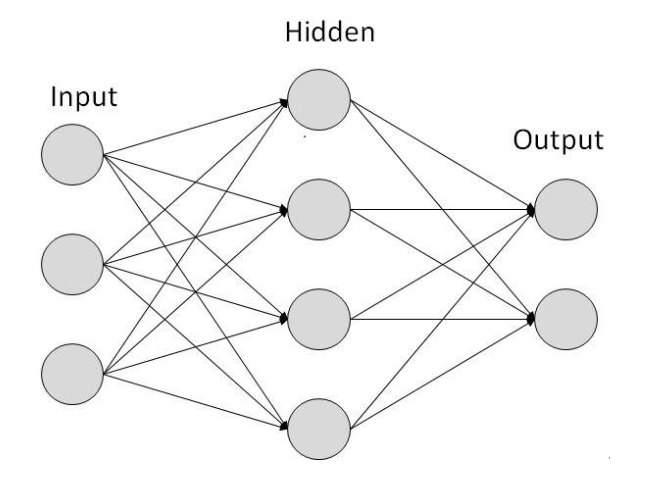
The idea of ANNs is based on the belief that working of human brain by making the right connections, can be imitated using silicon and wires as living neurons and dendrites. The human brain is composed of 86 billion nerve cells called neurons. They are connected to other thousand cells by Axons. Stimuli from external environment or inputs from sensory organs are accepted by dendrites. These inputs create electric impulses, which quickly travel through the neural network. A neuron can then send the message to other neuron to handle the issue or does not send it forward.



*Figure 1.1*

ANNs are composed of multiple nodes, which imitate biological neurons of human brain. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its activation or node value.

Each link is associated with weight. ANNs are capable of learning, which takes place by altering weight values. The following illustration shows a simple ANN .



*Figure 1.2*

1.2 Applications of Neural Networks

They can perform tasks that are easy for a human but difficult for a machine −

1. Aerospace − Autopilot aircrafts, aircraft fault detection.

2. Automotive − Automobile guidance systems.

3. Military − Weapon orientation and steering, target tracking, object discrimination, facial recognition, signal/image identification.

4. Electronics − Code sequence prediction, IC chip layout, chip failure analysis, machine vision, voice synthesis.

5. Financial − Real estate appraisal, loan advisor, mortgage screening, corporate bond rating, portfolio trading program, corporate financial analysis, currency value prediction, document readers, credit application evaluators.

6. Industrial − Manufacturing process control, product design and analysis, quality inspection systems, welding quality analysis, paper quality prediction, chemical product design analysis, dynamic modeling of chemical process systems, machine maintenance analysis, project bidding, planning, and management.

7. Medical − Cancer cell analysis, EEG and ECG analysis, prosthetic design, transplant time optimizer.

8. Speech − Speech recognition, speech classification, text to speech conversion.

9. Telecommunications − Image and data compression, automated information services, real-time spoken language translation.

10. Transportation − Truck Brake system diagnosis, vehicle scheduling, routing systems.

11. Software − Pattern Recognition in facial recognition, optical character recognition, etc.

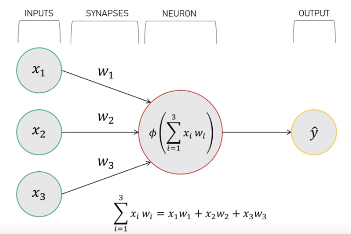
12. Time Series Prediction − ANNs are used to make predictions on stocks and natural calamities.

13. Signal Processing − Neural networks can be trained to process an audio signal and filter it appropriately in the hearing aids.

**CHAPTER 2**

**2.1** What is perceptron?

Perceptron is a single layer neural network and a multi-layer perceptron is called Neural Networks. Perceptron is a linear classifier (binary). Also it is used in supervised learning. It helps to classify the given input data. A normal neural network looks like this as we all know



*Figure 2.1*

The perceptron consists of 4 parts .

1. Input values or One input layer

2. Weights and Bias

3. Net sum

4. Activation Function

Steps:

1. Initialize the weights and the threshold. Weights may be initialized to 0 or to a small random value. In the example below, we use 0.
2. For each example *j* in our training set *D*, perform the following steps over the input {\displaystyle \mathbf {x} \_{j}}and desired output {\displaystyle d\_{j}}:
   1. Calculate the actual output:



{\displaystyle {\begin{aligned}y\_{j}(t)&=f[\mathbf {w} (t)\cdot \mathbf {x} \_{j}]\\&=f[w\_{0}(t)x\_{j,0}+w\_{1}(t)x\_{j,1}+w\_{2}(t)x\_{j,2}+\dotsb +w\_{n}(t)x\_{j,n}]\end{aligned}}} b. Update the weights:

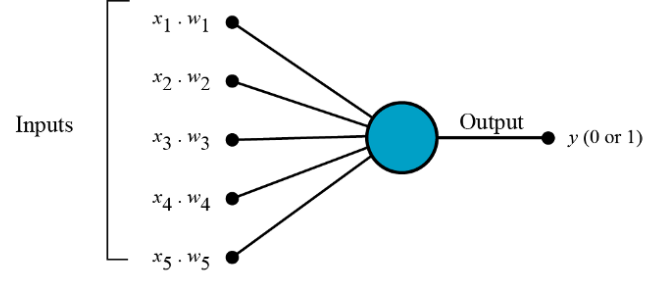


1. For [offline learning](https://en.wikipedia.org/wiki/Offline_learning), the second step may be repeated until the iteration error {\displaystyle {\frac {1}{s}}\sum \_{j=1}^{s}|d\_{j}-y\_{j}(t)|} is less than a user-specified error threshold {\displaystyle \gamma }, or a predetermined number of iterations have been completed, where *s* is again the size of the sample set.

The algorithm updates the weights after steps 2a and 2b. These weights are immediately applied to a pair in the training set, and subsequently updated, rather than waiting until all pairs in the training set have undergone these steps.

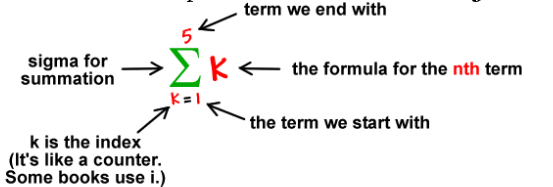
a. All the inputs x are multiplied with their weights w. Let’s call

it k.



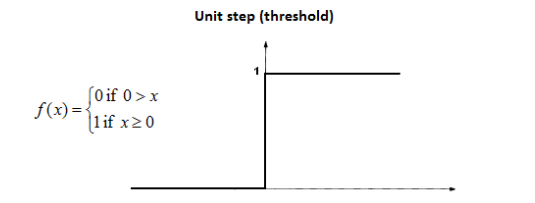
*Figure 2.2*

b. Add all the multiplied values and call them Weighted Sum.



c. Apply that weighted sum to the correct Activation Function.

For Example : Unit Step Activation Function.

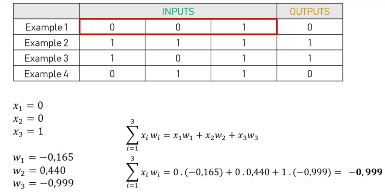


*Figure 2.3*

Weights shows the strength of the particular node.

A bias value allows you to shift the activation function curve up or down.In short, the activation functions are used to map the input between the required values like (0, 1) or (-1, 1).

Example :



**Chapter 3**

**3.1 Training**

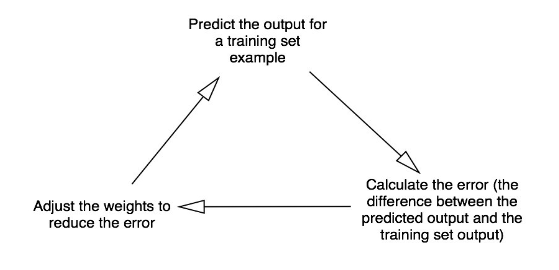
We will give each input a weight, which can be a positive or negative number. An input with a large positive weight or a large negative weight, will have a strong effect on the neuron’s output. Before we start, we set each weight to a random number. Then we begin the training process:

1. Take the inputs from a training set example, adjust them by the weights, and pass them through a special formula to calculate the neuron’s output.

2. Calculate the error, which is the difference between the neuron’s output and the desired output in the training set example.

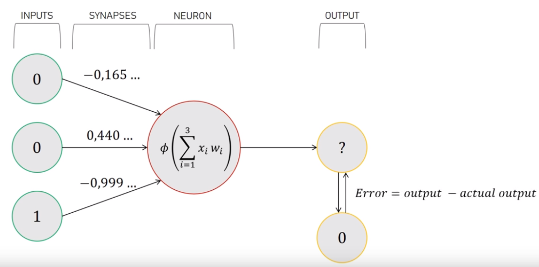
3. Depending on the direction of the error, adjust the weights slightly.

4. Repeat this process 10, 000 times.



*Figure 3.1*

Eventually the weights of the neuron will reach an optimum for the training set. If we allow the neuron to think about a new situation, that follows the same pattern, it should make a good prediction.



def train(self, training\_set\_inputs, training\_set\_outputs, number\_of\_training\_iterations):

for iteration in range(number\_of\_training\_iterations):

# Pass the training set through our neural network (a single neuron).

output = self.think(training\_set\_inputs)

# Calculate the error (The difference between the desired output and the predicted output).

error = training\_set\_outputs - output

# Multiply the error by the input and again by the gradient of the Sigmoid curve.

adjustment = dot(training\_set\_inputs.T, error \* self.\_\_sigmoid\_derivative(output))

# Adjust the weights.

self.synaptic\_weights += adjustment

# The neural network thinks.

def think(self, inputs):

# Pass inputs through our neural network (our single neuron).

inputs=inputs.astype(float)

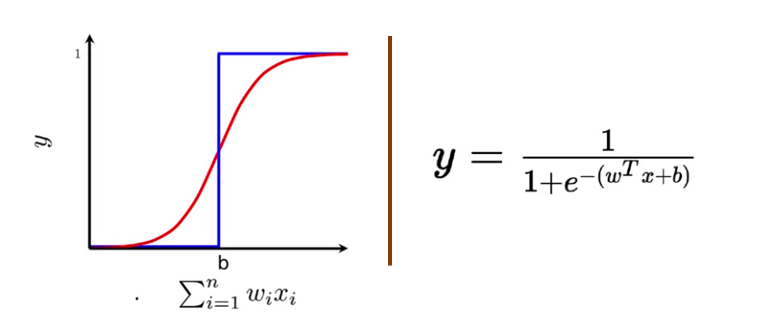
outputs=self.\_\_sigmoid(dot(inputs, self.synaptic\_weights))

return outputs

**CHAPTER 4**

4.1 Sigmoid Neuron

Introducing sigmoid neurons where the output function is much smoother than the step function. In the sigmoid neuron, a small change in the input only causes a small change in the output as opposed to the stepped output. There are many functions with the characteristic of an “**S**”shaped curve known as sigmoid functions. The most commonly used function is the logistic function.

****

*Figure 3.1*

We no longer see a sharp transition at the threshold **b**. The output from the sigmoid neuron is not 0 or 1. Instead, it is a real value between 0–1 which can be interpreted as a probability.

def \_\_sigmoid(self, x):

return 1 / (1 + exp(-x))

# The derivative of the Sigmoid function.

# This is the gradient of the Sigmoid curve.

# It indicates how confident we are about the existing weight.

def \_\_sigmoid\_derivative(self, x):

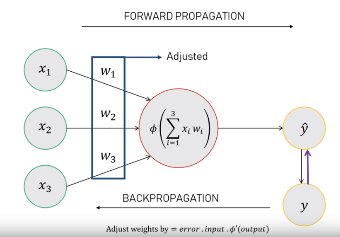
return x \* (1 - x)

# We train the neural network through a process of trial and error.

# Adjusting the synaptic weights each time.

**CHAPTER 5**

**5.1 BACKPROPAGATION**



Backpropagation is the central mechanism by which neural networks learn. It is the messenger telling the network whether or not the net made a mistake when it made a prediction.

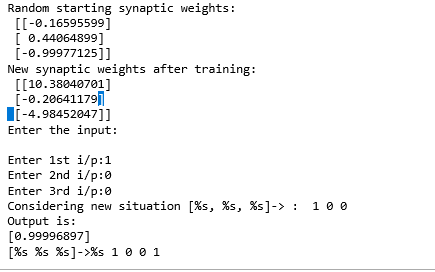
*To propagate* is to transmit something (light, sound, motion or information) in a particular direction or through a particular medium. When we discuss backpropagation in deep learning, we are talking about the transmission of information, and that information relates to the error produced by the neural network when they make a guess about data.

A neural network propagates the signal of the input data forward through its parameters towards the moment of decision, and then *backpropagates*information about the error through the network so that it can alter the parameters one step at a time.

You could compare a neural network to a large piece of artillery that is attempting to strike a distant object with a shell. When the neural network makes a guess about an instance of data, it fires, a cloud of dust rises on the horizon, and the gunner tries to make out where the shell struck, and how far it was from the target. That distance from the target is the measure of error. The measure of error is then applied to the angle of and direction of the gun (parameters), before it takes another shot.

Backpropagation takes the error associated with a wrong guess by a neural network, and uses that error to adjust the neural network’s parameters in the direction of less error. How does it know the direction of less error?

AFTER BACKPROPAGATION:



**CONCLUSION**

The aim of these project to create number system using simple perceptron algorithm. Traditional computer programs normally can’t learn. What’s amazing about neural networks is that they can learn, adapt and respond to new situations. Just like the human mind.

**REFERENCE**

<https://medium.com/technology-invention-and-more/how-to-build-a-simple-neural-network-in-of-python-code-cc8f23647ca1>

<https://en.wikipedia.org/wiki/Artificial_neural_network>