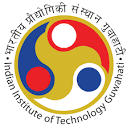
**INDIAN INSTITUTE OF TECHNOLOGY, GUWAHATI**

****

**DEPARTMENT OF MECHANICAL ENGINEERING**

**ME 674 (Soft Computing)**

**Coding Assignment 1**

**Artificial Neural Network**

**Submitted by: SADASHIV KESHAV**

**Roll No: 214103319**

**SUBJECT INSTRUCTOR: PROF. Sukhomay Pal**

**TABLE OF CONTENTS**

1. Problem Statement
2. Theory of the Problem
3. Selecting dataset from the problem
4. Applying Artificial Neuron Network to train the model
5. Checking for error in the model and the practical value
6. Coding for Artificial Neural Network with back propagation
7. Conclusion

**PROBLEM STATEMENT**

The present experimental study deals with natural convection through vertical cylinder .The experimental set up is designed and used to study the natural convection phenomenon from vertical cylinder in terms of heat transfer coefficient.

A vertical cylindrical rod of brass is taken and kept inside a wooden enclosure. The brass rod is heated with the help of heating coil wrapped around the brass rod. There are thermocouples placed at different points to measure the temperature at different locations of the brass rod. Heat Transfer by natural convection will take place from the rod. We will compare the power input given to the brass rod for heating and the heat transfer that will take place through natural convection. By comparison we would be able to find out the value of heat transfer coefficient (h).

**THEORY OF THE PROBLEM**

The power input given to the brass rod for heating can be calculated by the relation given below

Q=V\*I V-Applied Voltage

I-Applied Current

The heat transfer by natural convection can be calculated by the relation given below

Q=h\*A\*∆T

On comparing the above two equations we can calculate the value of heat transfer coefficient as follows

h= (V\*I) / ( h\*A\*∆T)

**Practical Setup of the Experiment**

** **

a) Brass Rod

b) Overall Experimental Setup

**Sample Calculation**

Heat Supplied = V\*I

= 220\*0.5

=110 watt

We also know that,

Heat Supplied=h\*AΔT

Thus

h\* π\*d\*l\*ΔT=110

h\* π\*0.038\*0.5\*51=110\*0.7

(Taking a loss factor of approximately 30%)

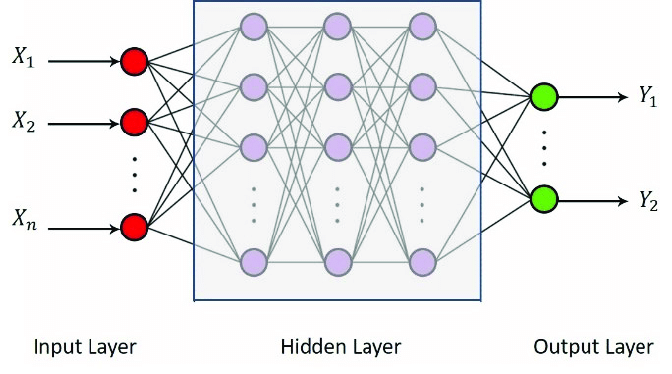
h=**25.29 W/m2k**

**Selecting Dataset from the Problem**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Voltage (V) | Current (I) | Efficiency(η) | Area(A) | Heat Transfer |
|  |  |  |  | Coefficient(h) |
| 220 | 0.5 | 0.7 | 0.059 | 51 |
| 210 | 0.6 | 0.72 | 0.059 | 52 |
| 215 | 0.55 | 0.75 | 0.06 | 54 |
| 217 | 0.58 | 0.68 | 0.065 | 52 |
| 206 | 0.62 | 0.69 | 0.064 | 53 |
| 205 | 0.56 | 0.71 | 0.059 | 50 |
| 211 | 0.6 | 0.69 | 0.071 | 61 |
| 212 | 0.58 | 0.64 | 0.063 | 58 |
| 225 | 0.64 | 0.67 | 0.073 | 55 |
| 230 | 0.59 | 0.69 | 0.071 | 59 |
| 211 | 0.61 | 0.7 | 0.069 | 56 |
| 219 | 0.6 | 0.55 | 0.062 | 55 |
| 220 | 0.6 | 0.71 | 0.064 | 58 |
| 216 | 0.7 | 0.69 | 0.054 | 57 |
| 210 | 0.65 | 0.68 | 0.051 | 53 |
| 225 | 0.68 | 0.72 | 0.049 | 54 |
| 215 | 0.64 | 0.69 | 0.055 | 55 |
| 210 | 0.8 | 0.62 | 0.069 | 55 |
| 215 | 0.76 | 0.65 | 0.065 | 50 |
| 220 | 0.55 | 0.71 | 0.069 | 52 |
| 214 | 0.61 | 0.62 | 0.064 | 56 |
| 218 | 0.65 | 0.67 | 0.055 | 58 |
| 215 | 0.64 | 0.66 | 0.058 | 60 |
| 222 | 0.62 | 0.65 | 0.057 | 55 |
| 220 | 0.64 | 0.62 | 0.055 | 58 |

**Training the Artificial Neural Network (ANN) for our problem**

For our given Problem statement we are considering 5 input neurons, 3 hidden neurons and 1 output neuron. Initially we are randomly assigning the weight value to U and V matrix.



We can calculate an output from a neural network by propagating an input signal through each layer until the output layer outputs its values.

We call this forward-propagation.

It is the technique we will need to generate predictions during training that will need to be corrected, and it is the method we will need after the network is trained to make predictions on new data.

We can break forward propagation down into three parts

1. Neuron Activation.
2. Neuron Transfer.
3. Forward Propagation

**Neuron Activation**

The first step is to calculate the activation of one neuron given an input.

The input could be a row from our training dataset, as in the case of the hidden layer. It may also be the outputs from each neuron in the hidden layer, in the case of the output layer.

Neuron activation is calculated as the weighted sum of the inputs.

Activation = sum (weight \* input) + bias

#### Neuron Transfer

Once a neuron is activated, we need to transfer the activation to see what the neuron output actually is.

Different transfer functions can be used. It is traditional to use log sigmoid function.

Output = 1 / (1 + e^ (-activation))

**Forward Propagation**

Forward propagating an input is straightforward.

We work through each layer of our network calculating the outputs for each neuron. All of the outputs from one layer become inputs to the neurons on the next layer.

### **Back Propagate Error**

The backpropagation algorithm is named for the way in which weights are trained.

Error is calculated between the expected outputs and the outputs forward propagated from the network. These errors are then propagated backward through the network from the output layer to the hidden layer, assigning blame for the error and updating weights as they go.

This part is broken down into two sections.

1. Transfer Derivative.
2. Error Backpropagation

#### Transfer Derivative

Given an output value from a neuron, we need to calculate its slope.

We are using the sigmoid transfer function, the derivative of which can be calculated as follows:

Derivative = output \* (1.0 - output)

#### Error Backpropagation

The first step is to calculate the error for each output neuron, this will give us our error signal (input) to propagate backwards through the network.

The error for a given neuron can be calculated as follows:

Error = (output - expected) \* transfer derivative (output)

This error calculation is used for neurons in the output layer. The error signal for a neuron in the hidden layer is calculated as the weighted error of each neuron in the output layer. Think of the error traveling back along the weights of the output layer to the neurons in the hidden layer.

The back-propagated error signal is accumulated and then used to determine the error for the neuron in the hidden layer, as follows

Error = (weight \* error) \* transfer derivative (output)

**Conclusion**

Total 25 input pattern has been fed to the ANN Model in which 17 has been used to train the model while rest of 8 input patterns has been fed for testing the model. Below is the normalized output value from the model and the normalized target output value. Corresponding error has also been shown.

**Output Value Target Value Error**

0.282471 0.422367 0.139896

0.144337 0.238770 0.094433

0.196395 0.482945 0.286550

0.126768 0.075676 0.051093

0.133820 0.012302 0.121518

0.257716 0.346878 0.089162

0.212607 0.175862 0.036745

0.218517 0.290028 0.071511

0.221922 0.235508 0.013586

If we decrease the tolerance limit then the model will be more accurate in predicting the output and the error will be less but it will take more iteration to do so and subsequently more time.