Carceptron: Prediction of Car Purchasing using Backpropagation Neural Network

Ravi Tomar[1], Aditi Chauhan[2], Hardik Kachhwaha[3], Prashant Kumar Gautam[4]
[1,2,3,4]University of Petroleum and Energy studies
Dehradun, uttarakhand, India
[1]rtomar@ddn.upes.ac.in
[2]aditi97chauhan@gmail.com
[3]kachhwahahardik96@gmail.com
[4]kumar.gautamprashant14@stu.upes.ac.in

Abstract

We make so many predictions on a very little fact. Sometimes, we are wrong quite often. We can't afford to make wrong prediction or decision where we have to invest. Like cars where we are investing a lot and there are many fishes in the sea type of fact here because we have more than 300 brands of car and in those we have different categories of them. So, it's bit confusing what to do and where to invest. Carceptron, it's all about making accurate prediction, which will help us to invest on a right car. In this paper we have used C programming language and implemented the Backpropagation algorithm using Multilayer Perceptron in neural networks and used the car evaluation dataset for analysis and predicting whether the car is good for buying or not the basis of the input data by user.

KEYWORDS

Perceptron, Carceptron, Backpropagation

1. INTRODUCTION

Car is one of the huge investments that one makes in his life. With number of options in hand, car purchase is a very confusing and tough task. Selecting the right choice according to the needs, features and budget with more number of available brands and models in market. This sometimes can lead to wrong decisions. Hence, there is a need of a concept which can predict the car on the basis of buyer's requirement and produce the best suitable output. Our report proposes the concept of Carceptron which will make accurate decisions by implementing Backpropagation artificial neural networks [1, 2]. Some of the most important factors which effects the car purchase are been considered and there response is been trained using Backpropogation algorithm, which analyze the information and will tell the customer about the car whether it is feasible to purchase the car or not.

Artificial Neural Networks (ANN) are used for processing of information like our biological Nervous system works. Neural Networks can be used in various sectors and has many applications such as speech recognition, pattern recognition, classification of data, image analysis etc. [3, 4]. Training is compulsory when we have to pass a random function using Artificial Neural Network Training means when we have values generated from a function are not accurate or approx.

2. PROBLEM OVERVIEW

In today's world we have more than millions of features, brands and models present in the automobile world and selecting the best possible car with all the needed facilities within the budget in very difficult and a tough task as many options are available today, and when you plan to buy a car you need to do a lot of research, many conditions are to be kept in mind like budget, safety conditions, boot space, comfort etc, and if you do not do proper research work then it can lead to loss of money which is not affordable . People today are so busy in their life that it is very difficult to review all the feature, brands and models of the car available in the market due to busy schedule and unawareness.

3. BASICS OF NEURAL NETWORKS

Artificial Neural Networks are relatively crude electronic models based on the neural structure of the brain [5]. Neural networks are arranged in layers. Each individual layer is made up of nodes. Each layer is connected to the next layer and nodes are all interconnected. Each hidden layer consists of sigmoidal activation function which helps network to stabilize. [6-8]

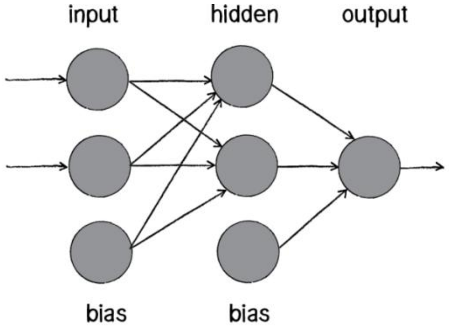


Fig 1: Neural Network

This forms a complete network which consists of an input layer, hidden layers and an output layer. At input layer, we take pattern as input. Hidden layer processes it and finally we get output at output layer.

The perceptron: A perceptron is the simplest neural network possible: a computational model of a single neuron [6]. It consists of one or more inputs, a processor, and a single output [6].

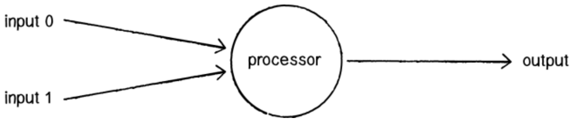


Fig 2: Perceptron

One of the important features of a neural network is its ability to learn. There are different types of learning:

Supervised learning: The system where input and output is provided for training[7].
Unsupervised learning: In this system in which only the input is provided, and a certain pattern has to be found out within the set of inputs by the network [7].
Reinforced Learning: It is similar to supervised learning in that some feedback is given, however instead of providing a target output a reward is given based on how well the system performed[7].

4. LITERATURE REVIEW

According to the Muller et al. [2] there are two main reason for NN investigation, first is to try to get an understanding on how human brain function and second is desire to build machines that are capable for solving complex problems that sequentially operating computers were unable to solve. It has also been said by some of the researchers that errors are handled better by NN than traditional computers programs. So predicting the result of a dataset which has huge number of attributes and undefined output can be made easy using neural network .It is a more efficient way for predicting and classifying data rather than doing it with any of the traditional methods. Results that they get using neural networks are encouraging. That is why NN are becoming so popular.

5. PROPOSED WORK

This section will address the idea behind the Carceptron, people get too confused while buying any car from the varieties which fall in their budget. The proposed model is calculating the condition of the car and categorizing the cars into four different categories.

Firstly, there are six attributes on which any car is decided to buy or not which are number of doors, buying cost, maintenance cost, safety, luggage boot, number of people accommodation. These attributes are assigned a random weight and multiplied with a bias function for every six attributes there are six nodes generated in hidden layer for every node the sigmoid function is calculated. Then similarly, last layer output layer is calculated by same.
The assigned weights to the nodes are random values so there must be an error or deviation from the original values. So error is calculated in the each node by using formula

Error = (output of feedforward)*(1-output of feedforward)*(1-output of feedforward)

Then errors on hidden layer is calculated by using the formula

Error = output*(1-output)*(error of output layer)*(weight)

Then feed forward is run again for the accurate results. Here, there is proper implementation of multilayer perceptron using artificial neural network the last accurately obtained results are the accurate/expected outputs after the removal of errors.

6. ALGORITHM

```
Step 1-  Normalize data to set between 0 and 1, using
         (ch[i]-xmin )/(xmax - xmin )
Step 2-  for every dataset in the training set
          Input the dataset to the network
         Transfer the input forward to the next layer
through the network:
for every single layer in the network
for all the node in each layer
     1.  calculate the random number using
         rand()/(double)RAND_MAX
     2.  Calculate the sum of the inputs each node in the
         layer
     3.  Add the bias to the sum of inputs of each layer
     4.   Calculate the activation function for the node
         using1/(1+pow(2.71,(-1)*sum));
     End
Step 3-   Propagating the errors backward through the
         network layer by layer for each node in the
         output layer  calculating the error using  the
         formula

(output of feed forward)*(1-output of feed forward)*(1-
output of feed forward)

for all hidden layers in the network  and each  node in the
layer
     1.  Calculate the node's error using
         output*1-output*error of output layer*weight
     2.  Updating each node's weight in the network using
         the formula

((random1[1][j])+(l*erroroutputlayer[i][0]*outputofnetwok
[i][1]));
End
```

(i) Feed-forward in the network

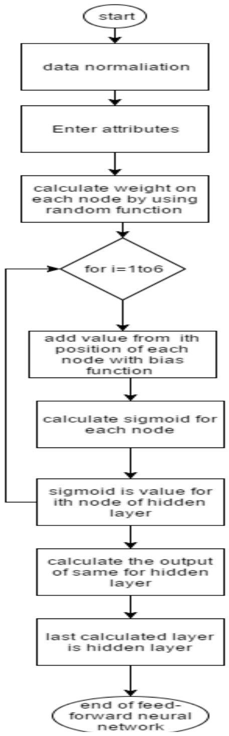

start
data normaliation
Enter attributes
calculate weight on each node by using random function
for i=1to6
add value from ith position of each node with bias function
calculate sigmoid for each node
sigmoid is value for ith node of hidden layer
calculate the output of same for hidden layer
last calculated layer is hidden layer
end of feed-forward neural network

Fig 3: Feedforward Flowchart

(ii) Backpropogation implementation in the network

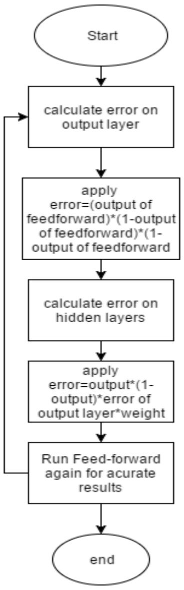
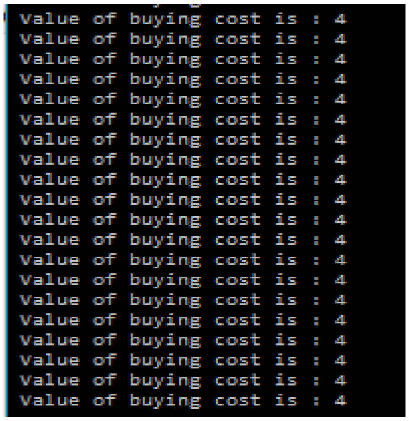

Start
calculate error on output layer
apply error=(output of feedforward)*(1-output of feedforward)*(1-output of feedforward
calculate error on hidden layers
apply error=output*(1-output)*error of output layer*weight
Run Feed-forward again for acurate results
end

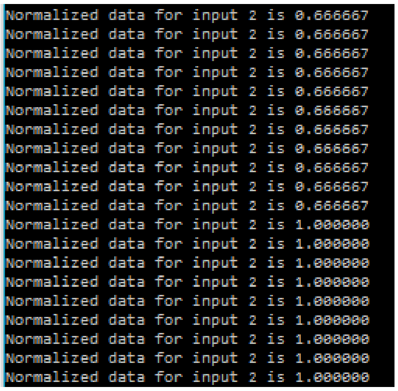
Fig 4: Backpropogation Flowchart

7. IMPLEMENTATION



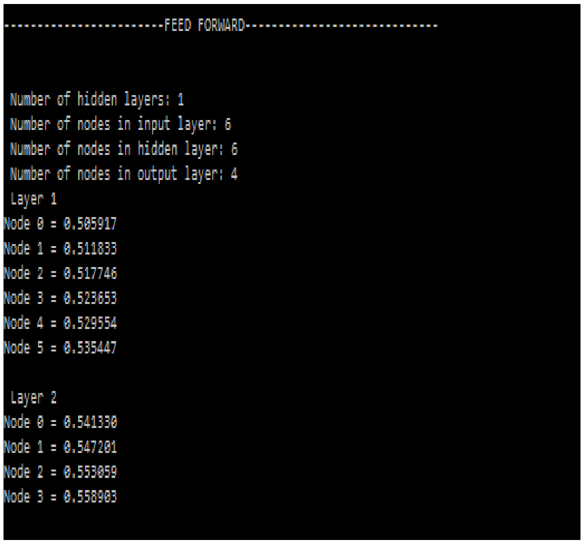
```
Value of buying cost is : 4
Value of buying cost is : 4
Value of buying cost is : 4
Value of buying cost is : 4
Value of buying cost is : 4
Value of buying cost is : 4
Value of buying cost is : 4
Value of buying cost is : 4
Value of buying cost is : 4
Value of buying cost is : 4
Value of buying cost is : 4
Value of buying cost is : 4
Value of buying cost is : 4
Value of buying cost is : 4
Value of buying cost is : 4
Value of buying cost is : 4
Value of buying cost is : 4
Value of buying cost is : 4
Value of buying cost is : 4
Value of buying cost is : 4
```

Fig 5: Data input in the network



```
Normalized data for input 2 is 0.666667
Normalized data for input 2 is 0.666667
Normalized data for input 2 is 0.666667
Normalized data for input 2 is 0.666667
Normalized data for input 2 is 0.666667
Normalized data for input 2 is 0.666667
Normalized data for input 2 is 0.666667
Normalized data for input 2 is 0.666667
Normalized data for input 2 is 0.666667
Normalized data for input 2 is 0.666667
Normalized data for input 2 is 0.666667
Normalized data for input 2 is 1.000000
Normalized data for input 2 is 1.000000
Normalized data for input 2 is 1.000000
Normalized data for input 2 is 1.000000
Normalized data for input 2 is 1.000000
Normalized data for input 2 is 1.000000
Normalized data for input 2 is 1.000000
Normalized data for input 2 is 1.000000
Normalized data for input 2 is 1.000000
```

Fig 6: Data Normalization output



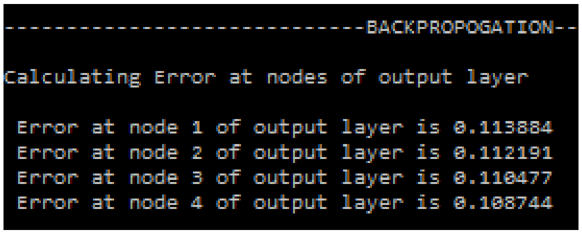
```
-----------------------FEED FORWARD----------------------------



 Number of hidden layers: 1
 Number of nodes in input layer: 6
 Number of nodes in hidden layer: 6
 Number of nodes in output layer: 4
 Layer 1
Node 0 = 0.505917
Node 1 = 0.511833
Node 2 = 0.517746
Node 3 = 0.523653
Node 4 = 0.529554
Node 5 = 0.535447

 Layer 2
Node 0 = 0.541330
Node 1 = 0.547201
Node 2 = 0.553059
Node 3 = 0.558903
```

Fig. 7: Feed forward output of the network

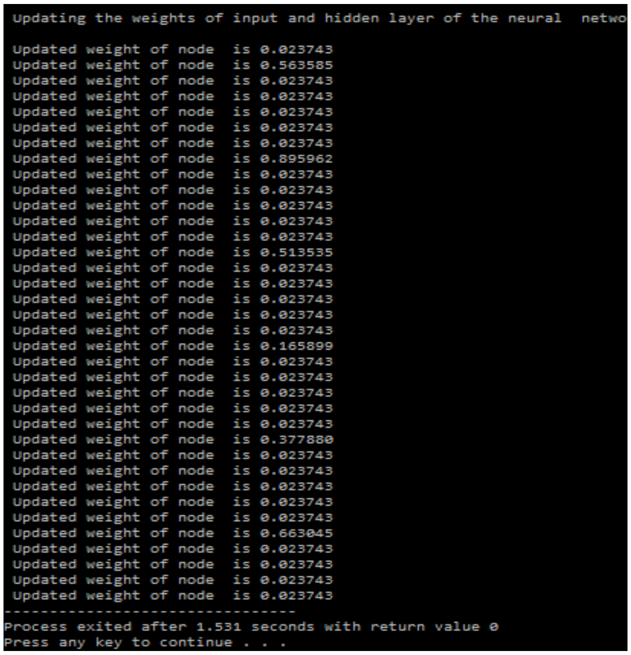


```
-----------------------------BACKPROPOGATION--

Calculating Error at nodes of output layer

 Error at node 1 of output layer is 0.113884
 Error at node 2 of output layer is 0.112191
 Error at node 3 of output layer is 0.110477
 Error at node 4 of output layer is 0.108744
```

Fig 8: Backpropogation output of the network



```
 Updating the weights of input and hidden layer of the neural  netwo

 Updated weight of node  is 0.023743
 Updated weight of node  is 0.563585
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.895962
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.513535
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.165899
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.377880
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.663045
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
 Updated weight of node  is 0.023743
--------------------------------
Process exited after 1.531 seconds with return value 0
Press any key to continue . . .
```

Fig. 9: Updated weights of the network

8. CONCLUSION

We have been able to complete our Paper work successfully to full satisfaction .As proposed we have implemented backpropogation neural network algorithm on our dataset. We were able to generate a code for the system. In the course of completion we have obtained a sound knowledge over general programming logic and C programming environment.

9. REFERENCES

[1] Md. Ali Hossain, Md. Mijanur Rahman, Uzzal Kumar Prodhan, Md. Farukuzzaman Khan, 2013. Implementation of Backpropagation Neural Network for Isolated Bangla Speech Recognition. International Journal of Information Sciences and Techniques (IJIST), Volume 3, Issue 4.

[2] Muller, J. Reinhardt, M. T. Strickland, 1995. Neural Networks: An Introduction, Physics of Neural Networks. Springer.

[3] www.ijsrp.org/research-paper-1212/ijsrp-p1240.pdf.

[4] Micheline Kamber, Jiawei Han, Jianpei, Data mining: Concepts and Techniques, Third Edition.

[5] S. B. Maind, P. Wankar, 2014. Basic of Artificial Neural Network. International Journal on Recent and Innovation Trends in Computing and Communication, Volume 2, Issue 1, pp. 96-100.

[6] http://natureofcode.com/book

[7] http://www.theprojectspot.com

[8] www.pages.cs.wisc.edu/~bolo/shipyard/neural/local.html.