Car Classification Using Artificial Neural Network

Sucheta Chauhan*, Prof. Prema K. V.**

FET, Mody Institute of Technology & Science, Lakshmangarh-332311 (Distt. Sikar), Rajasthan, INDIA
(*sucheta_chauhan2008@yahoo.com)
(**premakv.et@mitsuniversity.ac.in)

***Abstract*-** Car may be classified by a number of different standards and targets. Even, a broad classification is difficult, because a vehicle may fit into multiple categories. The proposed work provides a computer-based tool capable of classifying cars, as closely as possible to classifications performed by skilled operators. Such a tool is capable of extracting a number of numerical parameters characterizing the cars in areas like Value for Money, Design and Function and On the Road Performance. Such parameters are, then, used for training and learning an Artificial Neural Network (ANN) with the aim of classifying them by using different training and learning functions which reflects its performance level. From the results, it is revealed that TRAINLM, LEARNGDM & LOGSIG give comparatively good performance for this problem. So it would be easy to classify a car, whether it is family car or sports car or any other car within one second.

***Index Terms*-** Artificial Neural Networks (ANN), Backpropagation Learning Algorithm (BPP), Car parameters, Multilayer Feedforward Neural Network (MFNN).

I. INTRODUCTION

Basically, cars can be categorized into various types on the basis of their shape, size, mechanical specifications, wheels [10], performance, seat fabrics etc. [6]. We can't classify all the cars on the basis of one single formula. A particular car model can be categorized as a sports car as well as a convertible. Cars are classified on the basis of their manufacturing specifications like: Medium Cars, Family Cars, People Movers, Sports Cars, Luxury SUVs, All-Terrain 4WDs. Apart from this there are three key areas which are considered, being Value for Money, Design and Function and On the Road Performance.

Within these areas there are up to nineteen Car parameters:

1) Value for money-Pricing, Running cost etc.
2) Design & Function- Ergonomics, Innovation, Safety etc.
3) On Road Performance- Fuel Consumption etc.

Each criterion is weighted – critical, high, medium or low – according to their importance to buyers in the relevant market segment. Each criterion is given an importance weighting score between 1 and 0.4, with 1 being the Critical, with 0.8 being the high, with 0.6 being the average and with 0.4 being the low importance weighting for each criterion. An individual can purchase a car with its own satisfaction in all areas. Each vehicle's overall weighting reflects its position within the class and should only be compared within its category.

Classification is a variable technique referred with assigning data cases to one of a fixed number of possible classes [5]. A Multilayer Feedforward Neural Network with Backpropagation Algorithm is used for car classification.

Artificial Neural Network is a network or circuit of artificial neurons/nodes, i.e. processing units like neurons in the brain. The signals are transmitted by means of connection links. Links are associated with weights which are multiplied by net input. Output is calculated by applying activation function to the net input. **Feedforward Neural Network** has a single layer of weights where the inputs are directly connected to the outputs. In this network, all the neurons are directed towards the front. Each neuron on the layer is connected to another neuron on the next layer without feedback connection [2]. **Multilayer Feedforward Neural Network** consists of a set of inputs at the input layer, one or more hidden layers, and an output layer with output nodes where error (Desired-Actual Output) is computed. The input data are to be sent in the forward direction from one layer to another layer up to the output layer. These types of networks are related to Multilayer Perceptron (MLP) Network. Here in figure 1: a Multilayer Perceptron Network with 2 neurons on input layer, 3 neurons on single hidden layer and 1 neuron on output layer is shown [1].

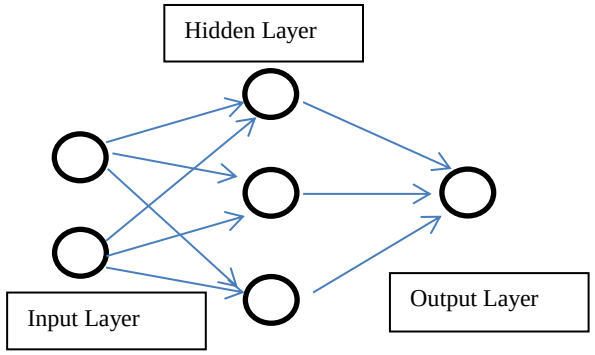

Figure 1: multilayer feed forward network architecture.

This error, in turn, can be used to modify the weights associated with the units. Desired output is known only at output layer so error is computed at the output layer only and then has to tune the hidden layer to get minimum error. Because we know the desired output only at the output layer but not on the every hidden layer that's why Backpropagation algorithm came into picture [9].

II. METHODOLOGY

Backpropagation Learning:

It's a commonly used method for training the artificial neural network to minimize the gradient.

Steps in BPP algorithms:
Step 1: Randomly initialize the weights and biases.
Step 2: feed the training sample.
Step 3: Propagate the inputs forward; compute the net input and output of each unit in the hidden and output layers.
Step 4: back propagate the error to the hidden layer.
Step 5: update weights and biases to reflect the propagated errors. Training and learning functions are mathematical procedures used to automatically adjust the network's weights and biases.
Step 6: terminating condition.

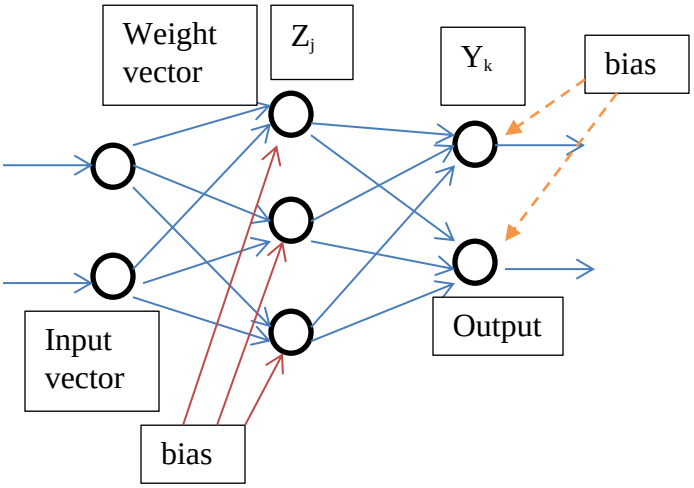

Figure 2: Error propagation through hidden layer

Propagate the inputs forward:
Each hidden unit (Z_j) sums its weighted input signals.

$$Z_{-inj} = \sum_i w_{ij} X_i + w_{oj}$$

Where, w_{ij} is the weight between hidden and input layer; X_i is the input training vector; and w_{oj} is the bias of the unit. Applying Activation function: $Z_j = f(Z_{-inj})$
And sigmoid function is calculated as :

$$f(Z_{-inj}) = 1/(1+e^{-Z_{-inj}})$$

And this signal to the output layer.
Each output unit (Y_k, k=1….m) sums its weighted input signals.

$$Y_{-ink} = \sum_j w_{jk} Z_j + w_{ok}$$

And output signal after applying activation function:

$$Y_k = f(Y_{-ink})$$

Back propagate the error:
When reaching the Output layer, the error is computed and propagated backwards to hidden.
For a unit k in the output layer the error is computed by a formula: $\delta_k = (T_k - Y_k) f(Y_{-ink})$

δ_k is error at output unit Y_k
Each hidden unit Zj sums its delta input from above layer inputs.

$$\delta_{-inj} = \sum_k \delta_k w_{jk}$$

The error is calculated as: $\delta_j = \delta_{-inj} f(Z_{-inj})$

δ_j is Error at hidden unit Zj.

Update weights and biases:
Weights are updated by the following equations

$$\Delta w_{ij} = \alpha \delta_j X_i \text{ and } \Delta w_{jk} = \alpha \delta_k Z_j$$

Where, α is learning rate
Biases are updated by the following equations:

$$\Delta w_{oj} = \alpha \delta_j \text{ and } \Delta w_{ok} = \alpha \delta_k$$

so $w_{ij(new)} = w_{ij(old)} + \Delta w_{ij}$ and $w_{jk(new)} = w_{jk(old)} + \Delta w_{jk}$
And biases are:

$$w_{oj(new)} = w_{oj(old)} + \Delta w_{oj} \text{ and } w_{ok(new)} = w_{ok(old)} + \Delta w_{ok}$$

Terminating Conditions:
Networks run until at least one of the following Termination conditions were satisfied: Maximum Epoch as specified is reached or Given Minimum Gradient reached or Performance Goal met [3].

III. PROPOSED MODEL

The Project development process for an ANN has seven steps.
Step 1(Accumulation of Data):
The data to be used for training the network are accumulated.
There are 19*552 size of input data matrix: as shown in Table 1. There are 6 classes like: Midsize car, Family, People mover, Sports car, Luxury car and 4 wheel Drive cars. Each Class has 92 samples which distinguish one class from another class.
Step 2 (Choosing Training and testing Data)
From the available data, training and testing data sets are distinguished. From the available input datasets; 168 samples are arbitrarily selected for training and 384 samples for testing (i.e. in ratio of 30:70). See in Table 2.
Step 3 (Choosing Network Architecture):
Network has one input Layer, one hidden layer and one output layer. So it's a 2 layer Feedforward network. Here 19 neurons at input layer, 3 neurons at the hidden layer and 6 neuron at output layer that shows 6 car classes.
Step 4 (Parameter Adjustment and Weight Initialization):
There are parameters (momentum constant, learning rate, maximum epochs, goal etc.) for adjusting the network to the expected level of performance and arbitrarily initializes the network weights [11].
Step 5 (Training):
The network training is done until the calculated outputs are within the margin of the known outputs.
Step 6 (Testing):
Now after successful training of network, it's ready for testing.
Step 7 (Implementation):
So now the network can reproduce the desired output for given inputs like those in the training set and also produces the output for unseen datasets [7].
We are calculating the performance efficiency by taking the following parameters:
1. Convergence Rate.
2. Epochs taken to converge the network.
3. The calculated MSE (Mean Squared Error).

IV. RESULTS AND ANALYSIS

Neural Network Toolbox in MATLAB has various training and learning functions.
We have applied TRAINLM and TRAINSCG training functions.

TABLE 1: Input and Class Samples

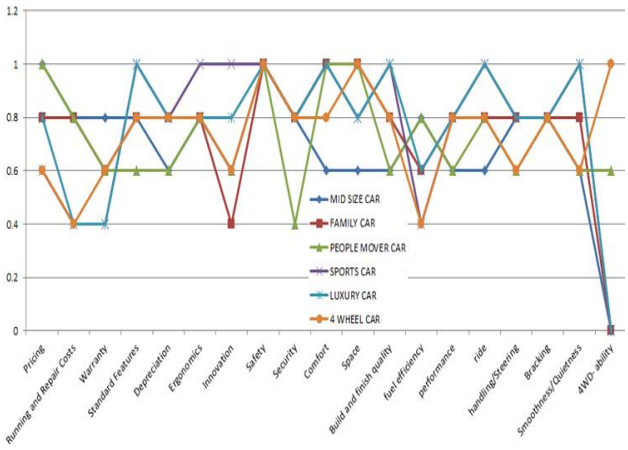

TABLE 2: Training & Testing Data Size

Input Data Size	Training set	Testing set	No. of classes	Attribute type
19*552	19*168	19*384	6	Real

TABLE 3: Summarizes the Information About The Convergence Of The Neural Network By Taking Account Different Parameters On Same Dataset

Training function	Learning function	Transfer function	No. of Epochs	MSE
TRAINLM	LEARNGDM	LOGSIG	10	0.00722
TRAINSCG	LEARNGDM	LOGSIG	32	0.029
TRAINLM	LEARNGD	LOGSIG	9	0.00952
TRAINSCG	LEARNGD	LOGSIG	31	0.028

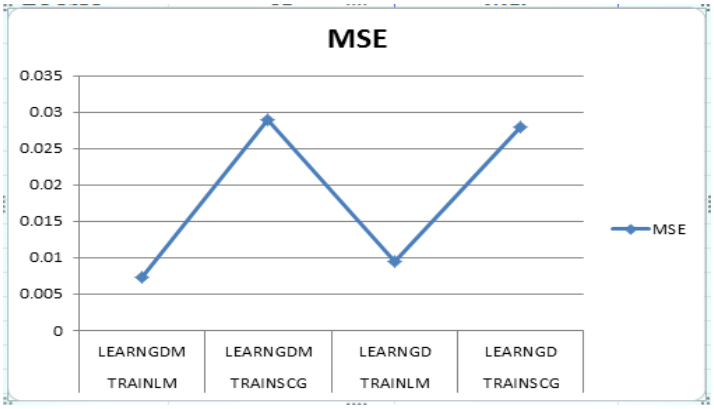

Figure 3: Mean Square Error representation for various parameters where MSE is minimum when I used TRAINLM & LEARNGDM parameters.

Where, TRAINLM (Train Levenberg-Marquardt) is used to train the network (update bias and weights) according to Levenberg-Marquardt Optimization and TRAINSCG (Train scaled conjugate gradient) is used to train the network (update bias and weights) according to scaled conjugate gradient method [4].
For learning purpose, we have applied LEARNGDM and LEARNGD learning functions. Where, LEARNGDM is used to adapt the network according to Gradient Descent Momentum weight & bias learning function and LEARNGD is used to adapt the network according to Gradient Descent weight & bias learning function. Their effect on MSE (Mean Square Error) is shown in Table 3.

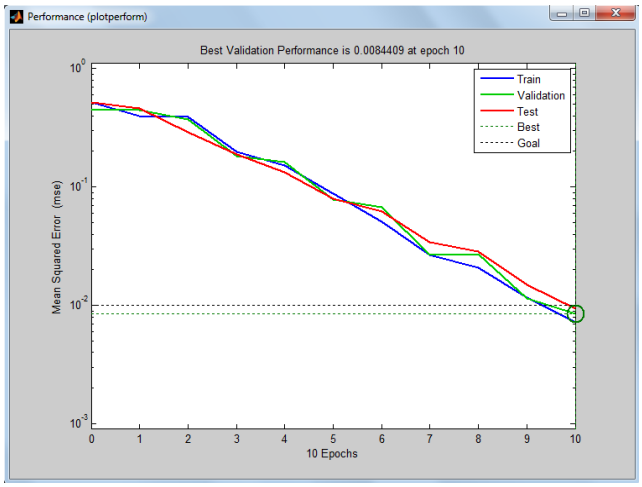

Figure 4: Network convergence for training function TRAINLM and learning function and Transfer function LEARNGDM and LOGSIG respectively. The network is converged with 10 epochs and with MSE as 0.00722

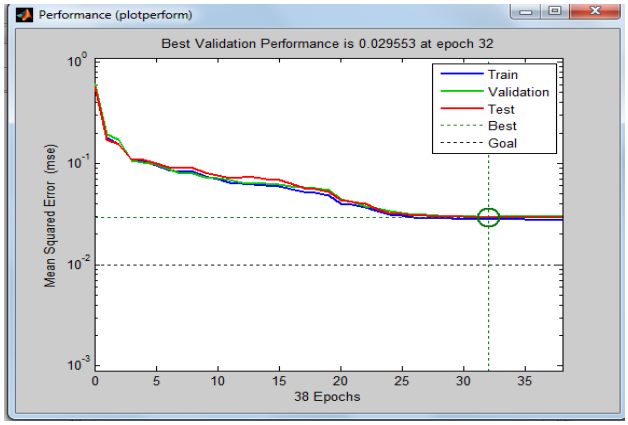

Figure 5: network convergence for training function TRAINSCG and learning and Transfer function LEARNGDM and LOGSIG respectively. The network is converged with 32 epochs and with MSE as 0.029.

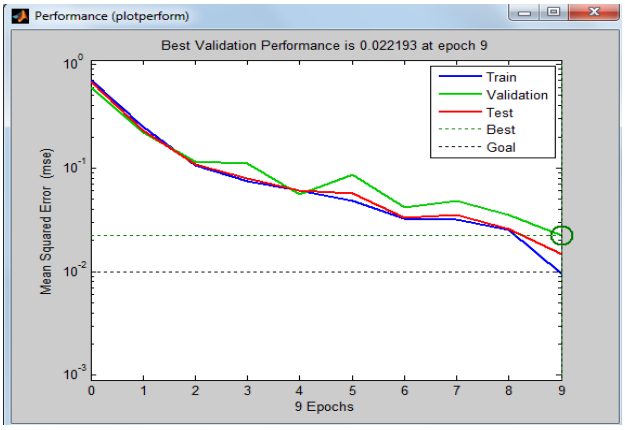

Figure 6: network convergence for training function TRAINLM and learning and Transfer function LEARNGD and LOGSIG respectively. The network is converged with 9 epochs and with MSE as 0.00952.

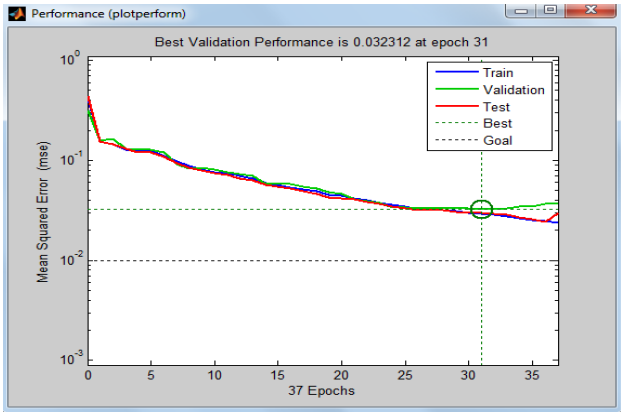

Figure 7: network convergence for training function TRAINSCG and learning and Transfer function LEARNGD and LOGSIG respectively. The network is converged with 31 epochs and with MSE as 0.028.

V. CONCLUSION

A Multilayer Perceptron using Backpropagation Algorithm can be used as a remedy for classification problems with full compound of Training, Learning and Transfer Functions. In proposed work it's found that the combination of TRAINLM, LOGSIG & LEARNGDM worked comparatively better, and the efficiency can be improved by dimensionality reduction using PCA (Principal Component Analysis) which is the future scope of this paper.

ACKNOWLEDGMENT

Sucheta Chauhan takes immense pleasure in thanking to my Guide **"Prof. Prema K.V."**, Head of Department of Computer Science and Engineering; for inspiring, directing, and motivating.

REFERENCES

[1] S. Haykin,"Multilayer Perceptron" in *Neural Networks and Learning Machines* (3rd Ed.): Publishing as Pearson Prentice Hall, 2009, ch. 4, pp. 122-139.

[2] Yegnanarayana, "Feedforward Neural Networks" in *Artificial Neural Networks*, New Delhi, India: Prentice- Hall of India, 1999, ch. 4, pp. 88-141.

[3] S N Sivanandam, S Sumathi, S N Deepa, "Back Propagation Network (BPN)" in *Introduction To NEURAL NETWORKS Using MATLAB 6.0*: Tata McGraw Hill Education Private Limited: New Delhi,2006, ch. 8, pp. 185-193.

[4] Rojalina Priyadarshini; "Functional Analysis of Artificial Neural Network for Dataset Classification", Special Issue of IJCCT Vol. 1 Issue 2, 3, 4; 2010 for International Conference [ACCTA-2010], 3-5 August 2010.

[5] Guoqiang Peter Zhang "Neural Network for Classification- A Survey 2000" IEEE Transactions on systems, man and cybermetics- part c: applications and reviews, Vol 30, No 4.

[6] R. Furferi, L. Governi; "Neural Network based classification of car seat fabrics" *International Journal of Mathematical Models and Methods in Applied Sciences*; Issues 3, Vol. 5, 2011.

[7] V. Arulmozhi, "Classification task by using Matlab Neural Network Tool Box- A Beginner's View" *International Journal of Wisdom Based Computing*, Vol. 1(2), August 2011.

[8] Prema K.V.,"A Multilayer Neural Network Classifier", *Journal of Computer Society of India*, Volume 35, Issue no: 1, January-March 2005.

[9] Jerzy Jackowski, ICSENG '05 Proceedings of the 18th *International Conference on System Engineering*, IEEE Computer Society Washington, DC, USA ©2005, pp. 212-217.

[10] Linhui Liu, "Realization and Application Research of BP Neural Network Based on MATLAB", International Seminar on Future Bio Medical Information Engineering (FBIE 2008), pp. 130- 133.

[11] Danu Pranantha "Neural Network in Business Intelligence": Lifelong Learning Journal (2009) – http://danupranantha.wordpress.com/2009/10/25/neuralnetwork-in-business-intelligence-part-3

AUTHORS

Sucheta Chauhan pursuing M.Tech. (Computer Science) from MITS, Lakshmangarh and email address: sucheta_chauhan2008@yahoo.com

Prof. Prema K.V., HOD of Computer Science department in FET, MITS, Lakshmangarh and email address: premakv.et@mitsuniversity.ac.in