Multilayer Perceptron network for automatic Driving vehicle

Behnam Ghiaseddin[1], Omid Rahmani Seryasat[2], Javad Haddadnia[3*]

[1,] Department of Electrical Engineering, Takestan branch, Islamic Azad University, Takestan, Iran
[2,] Department of Electrical Engineering, Takestan branch, Islamic Azad University, Takestan, Iran
[3.] Associate Professor, Electrical and Computer Engineering Department, Hakim Sabzevari University & Center for Research of Advanced Medical Technologies, Sabzevar University of Medical Sciences, Sabzevar, Iran
[*]Corresponding author: Haddadnia@sttu.ac.ir

Abstract: This study aims to determine a special model of MLP Neural Network in controlling the movement of a moving object (e.g. an Automobile) Controlling the movement of an automobile means moving it in a specified path with the ability of controlling two following parameters:1)Steer, 2) Speed The main idea of this work comes from that as the moving object locates in the turns of a path, it has to reduce its speed preventing from departure from the path. On the other hand, the more displacing from zero degree, the more decrease in the speed is required. Accepting the mentioned logic, the outputs (Speed and Steer) have different behaviors and can hardly be determined by a single Neural Network. In such situations, a normal MLP Neural Network includes a hidden layer with a sigmoid activation function. The best of sigmoid activation functions is Hyperbolic Tangent Sigmoid (known as TANSIG in MATLAB), and the output layer includes linear activation functions. The new idea is differing the activation functions of the output layer's Nodes. This change in functions must include this logic that speed must increase as the path is straight and must reduce as the path faces a turn. If the first layer of the Neural Network determines the degree of turn in the path, the second layer must determine the speed of the moving object and the steer. As the turns of the path changes to either of the sides (Right or Left), the steering must change value to one of these sides as the speed reduces anyway. For this reason, the speed's activation function has to generate the maximum in the middle of its values and should also reduce its value by moving to the sides, as the steer's activation function remains linear. For aiming this goal, the Gaussian activation function for the speed control is advised. The goal of this study is to design such Neural Network and survey its results.
[Behnam Ghiaseddin, Omid Rahmani Seryasat, Javad Haddadnia. **Multilayer Perceptron network for automatic Driving vehicle**. *Life Sci J* 2013;10(5s):400-404] (ISSN:1097-8135). http://www.lifesciencesite.com. 72

Keywords: Artificial Neural Network, Movement Control, Multi Layer Perceptron, Activation Function, Gaussian function.

1. Introduction

An artificial neural network is able to analyze and learn patterns and use this learning in acting with the least difference from desired action in different kinds of situations. A human brain is made of 1011 nodes connecting to each other with connection terminals named dendrites. Each node can be connected to one or more other nodes. These connections can be in single or double directions.

The transportation of data is done by electrochemical discharging of a node. The electrochemical signal is transported through the dendrites to the other nodes and human body's neural network. Changing the value of this discharge is called learning. When the human starts learning some act, the value of the electrochemical discharges will change in continual loops to reach an optimal value. This act of learning can be simulated with some blocks as the nodes connecting to each other in an apparent order. The value of electrochemical signals can be simulated as a gain of connection of these blocks. These gains are called "Weights". In learning process, there are these weights that change and when they reach an optimal value, the neural network has been trained. A simple Multi Layer Perceptron (MLP) artificial neural network is shown in figure 1.

In a MLP nodes are separated in different layers, connected to all nodes in the neighbor layer. In each layer on each node, we have an activation function which acts on the input value of each node.

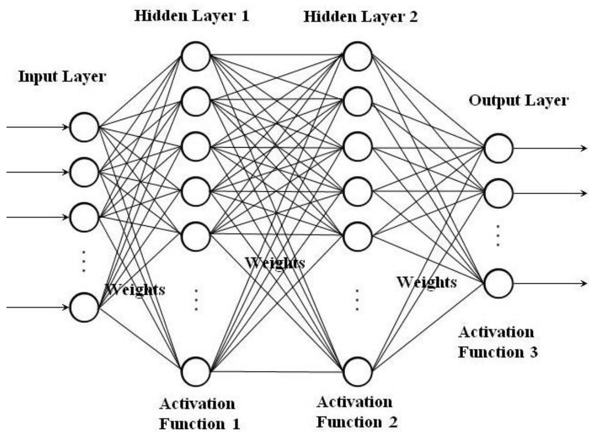


Figure 1. A three-layer artificial neural network

The transportation of data is done in a single direction from the first layer at the left to the last at the right. The input value of each node is the summery of the values of all connections going to that node.

2. Problem definition

In learning how to control a moving object, there are several elements to be mentioned. The image of the path already contains most of the data needed to analyze the path and finding the rules of control. Viewing the path, the driver can decide how much to steer or how much to accelerate or decelerate.

The above mentioned two elements are called “movement elements” which are used in a complete movement controlling process of a moving object. An image from a path either contains details and useless information. The controlling system must usually abandon these details or we can say that the controlling system should filter the noises of the path.

For teaching the Neural Network to control this object, some patterns of movement should be taken, so by learning these patterns the artificial Neural Network would be able to learn how to control the object. After training the Neural Network, we can use the learned weights on the same MLP to control the object’s movement.

The inputs of MLP Neural Network would be the images from the path and the output would be the movement elements.

The problem is that the nodes on the output layer of the MLP Neural Network contain two different activation functions which have to be formulized separately.

3. Architecture of the Neural Network

The Neural Network designed for the specified reason mentioned above, is similar to a normal MLP Neural Network. Only the activation function in the one of the output Nodes is changed from the linear function to Gaussian activation function.

Figure 2. shows a schematic model of such Neural Network.

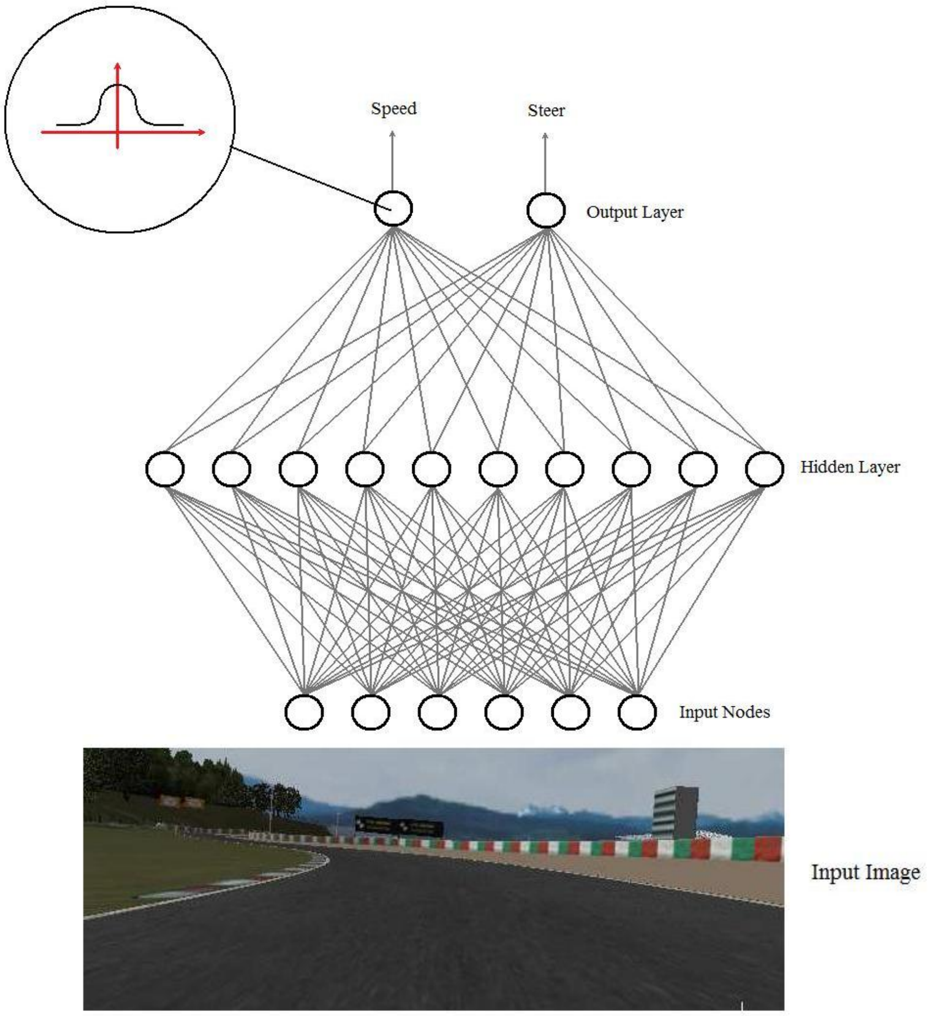


Figure 2. Architecture of the Neural Network

4. Training method

The training method, usually used for multi layer Perceptron artificial neural networks is Back Propagation of Errors (BPE). The main goal of BPE algorithm is to minimize an error function and adjust the parameters of the artificial neural networks. This error function is as written below:

$$J_g = \frac{1}{2}\sum_{i=1}^{N}(d_i - d_i^M)$$

Where M shows the output layer, N indicates the number of nodes in the output layer, di is the value of the desired ith output where diM is the actual ith output of the neural network.
For adjusting the weights of the network, the changes in the value of Wijk must be in the way of reduction of the value of the error function Jg.
Wijk-new= Wijk-old +ΔWijk

$$\Delta W_{ij}^k = \eta_1 \frac{\partial J_g}{\partial W_{ij}^k}$$

Where ηl is a positive value in the range of (0 1) and is called learning rate. There would be:

$$\Delta W_{ij}^k = -\eta_1 \frac{\partial J_g}{\partial O_j^{k+1}} \cdot \frac{\partial O_j^{K+1}}{\partial net_j^{k+1}} \cdot \frac{\partial net_j^{k+1}}{\partial W_{ij}^k}$$

Now defining a new value δjk as below:

$$\delta_j^k = -\frac{\partial J_g}{\partial O_j^k} \cdot \frac{\partial O_j^k}{\partial net_j^k}$$

Therefore the change in value of Wijk would be:

$$\Delta W_{ij}^k = \eta_1 \delta_j^{k+1} O_i^k$$

This way the error function of the network will reduce in loops until there would be small changes in the network's weights. In this case the neural network has learnt the logic of the problem and the trained weights can be used for giving outputs for some new inputs other than patterns.

4.1. Weight adjustments with Gaussian activation function.

The weight of a connection is adjusted by an amount proportional to the product of an error signal δ, on the unit k receiving the input and the output of the unit j sending this signal along the connection:

$$\Delta W_{jk}^p = \eta\, \delta_k^p y_j^p$$

If the unit is an output unit, the error signal is given by

$$\delta_o^p = (d_o^p - y_o^p) \,.\, f'(x_o^p)$$

Take as the activation function F the 'Gaussian' function as defined

$$y^p = f(x^p) = a.e^{-\frac{(x-b)^2}{2.c^2}}$$

In this case the derivative is equal to:

$$f'(x^p) = \frac{\partial}{\partial x^p}\left(a.e^{-\frac{(x-b)^2}{2.c^2}}\right)$$
$$= a.\left(-\frac{2(x-b)}{2.c^2}\right).e^{-\frac{(x-b)^2}{2.c^2}}$$
$$= a.e^{-\frac{(x-b)^2}{2.c^2}}.\frac{b-x}{c^2}$$
$$= y^p.\frac{b-x}{c^2}$$

such that the error signal for an output unit can be written as:

$$\delta_o^p = (d_o^p - y_o^p)\,.\,y_o^p.\left(\frac{b-x}{c^2}\right)$$

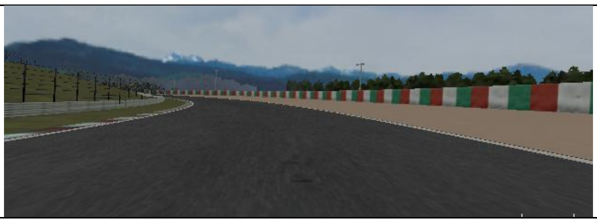
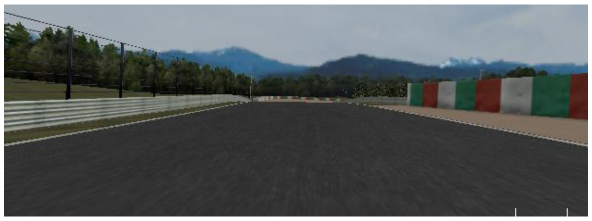
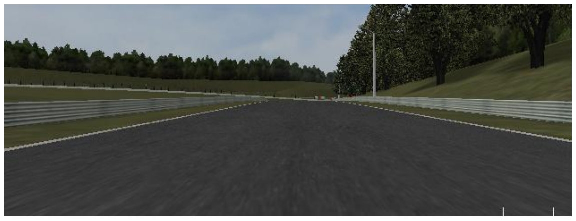
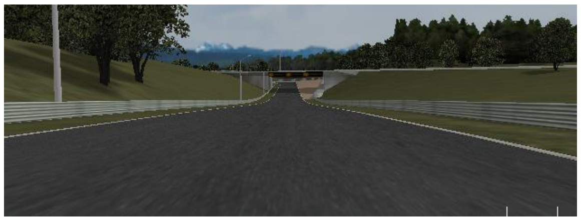
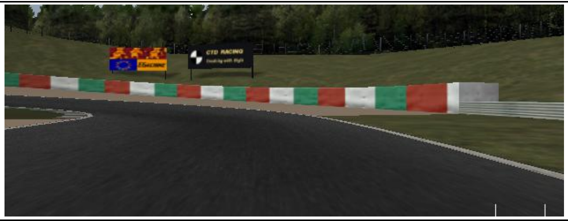
The error signal for a hidden layer is determined recursively in terms of error signals of the units to which it directly connects and the weights of those connections. For the Gaussian activation function:

$$\delta_h^p = f'(x_h^p).\sum_{o=1}^{N_o}\delta_o^p.\omega_{ho} = y^p.\frac{b-x}{c^2}.\sum_{o=1}^{N_o}\delta_o^p.\omega_{ho}$$

5. Results and Comparison

After the network has been trained, in order to evaluate the results and reaction of the trained Neural Network to a new input, some new images from the path were given to the network. Table 1 shows some of these images indicating the actual and desired outputs of the network.

Table 1. 5 examples for the images of the path and the Neural Network's results

No.	Input Picture	Steering	Speed	
1		-1.5	43	Desired Value
		-1.34	44.56	Actual Value
2		0.1	57	Desired Value
		0.11	55.81	Actual Value
3		0	51	Desired Value
		-0.01	53.23	Actual Value
4		0	68	Desired Value
		0.02	65.02	Actual Value
5		-5.2	43	Desired Value
		-4.98	44.79	Actual Value

As can be seen in the table, the trained MLP Neural Network has an acceptable result in steering and speeding. Table 2 indicates the errors of the values shown in table 1.

Table 2. Errors in the patterns shown in table 1

No.	Percent of the steering error	Percent of the Speed error
1	1.6	1.95
2	0.1	1.5
3	0.1	2.8
4	0.2	3.7
5	2.2	2.2

The results show that in steering, the more the steering is required, the more error will occur but the speed results follow no logic and they have acceptable error values.

6. Conclusions

It has been determined throughout the present research that a Neural Network of the simple Multi-Layer Perceptron type with 2 layers and a Gaussian activation function in the output layer on the Node which indicates the speed parameter is a better choice to be employed to take control of the movement of a moving object in a specified path.

In further expansion of this project, the comparison of the different Neural Networks will be determined.

Refrences

1. S. Haykin, Neural Networks A Comprehensive Foundation, Prentice Hall International, Inc., 1994.
2. Yu Hen Hu, Jenq-neng Hwang, Handbook of Neural Network Signal Processing, Electrical Engineering and Signal Processing Series, CRC Press LLC, 2002.
3. Nikola K. Kasabov, Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering, A Bradford Book, The MIT Press Cambridge, Massachusetts, Second Printing, 1998.
4. Anna Booth, Using Neural Networks to Improve Behavioral Realism in Driving Simulation Scenarios, TRL Limited, Wokingham, UK, 2007.
5. J. B. Siddharth Jonathan, Arvind Chandrasekhar, T. Srinivasan, Sentient Autonomous Vehicle Using Advanced Neural Net Technology, Department of Computer Science and Engg, Sri Venkateswara College of Engineering, Sriperumbudur, India, 2004
6. Julio K. Rosenblatt, DAMIN: A Distributed Architecture for Mobile Navigation, Robotics Institute Carnegie Mellon University, Pittsburgh, Pennsylvania, 9 October 1997.
7. S. Baluja, Evolution of an Artificial Neural Network Based Autonomous Land Vehicle Controller, IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, Vol. 26, No. 3, June 1996.
8. Parag H. Batavia, Dean A. Pomerleau, Charles E. Thorpe, Applying Advanced Learning Algorithms to ALVINN, Robotics Institute Carnegie Mellon University, Pittsburgh, Pennsylvania, 9 October 1996.
9. John Hancock, Chuck Thorpe, ELVIS: Eigenvectors for Land Vehicle Image System, Robotics Institute Carnegie Mellon University, Pittsburgh, Pennsylvania, 1995.
10. Todd M. Jochem, Dean A. Pomerleau, Charles E. Thorpe, Vision Guided Lane Transition, Robotics Institute Carnegie Mellon University, Pittsburgh, Pennsylvania, 1995.
11. Todd M. Jochem, Dean A. Pomerleau, Charles E. Thorpe, Vision-based Neural Network Road and Intersection Detection and Traversal, Robotics Institute Carnegie Mellon University, Pittsburgh, Pennsylvania, 1995.
12. Dean A. Pomerleau, Neural Networks Vision for Robot Driving, Carnegie Mellon University, School of Computer Science, 1995.

3/18/2013