Planar Robot Arm Performance: Analysis with Feedforward Neural Networks

Abraham Antonio López Villarreal, Samuel González-López, Luis Arturo Medina Muñoz
Technological Institute of Nogales Sonora
Mexico
samuelgonzalezlopez@gmail.com
lmedina@utnogales.edu.mx

ABSTRACT: *The purpose of this paper is to define the best training algorithm and activation function during the training process of a two degree of freedom planar robot arm that can be used to approach SCARA applications. A feed-forward neural network with two hidden layers was trained with several training algorithms such as Levenberg-Marquardt, Bayesian Regularization and others, and the activation functions such as symmetric sigmoid, logarithmic sigmoid and linear transfer to compare the resulting error and look for optimal performance.*

Keywords: Feedforward neural network, Planar robot arm, Training algorithm comparison, Activation function comparison

Received: 14 December 2016. Revised 24 January 2017, Accepted 31 January 2017

1. Introduction

The work on this paper consists in the design, structure and simulation of a feed-forward neural network, applied to the training of a two link, two degree of freedom planar robot arm, during the learning process in trajectory following. The computational implementation and simulation is developed in Matlab 2015b with the help of Simulink and the Neural Network Toolbox, in which it is possible to define and structure different neural networks, training algorithms and activation functions, and customize each section of the neural network to achieve the desired error.

Once the neural network has been created, the simulation is run with different training algorithms and activation functions in a comparative study, to determine which combination works best in an inverse kinematics problem applied to a two degree of freedom planar robot arm. In [4] a comparative study was carried out to select an appropriate neural network, the authors identifieddifferentlearning algorithms, including the Error Back Propagation (EBP)algorithm, the Levenberg Marquardt (LM) algorithm, and therecently developed Neuron-by-Neuron (NBN) algorithm. Similarly, our work performs a comparative analysis to identify those topologies of neural networks with better performance for the training of a robot arm.

2. Kinematics –Method

As defined in Inverse Kinematics – Basic Methods Paper [1], forward kinematics is based on the manipulation of a structure by changing the joint angles from the structure, while inverse kinematics do the opposite, movement is based on the direct manipulation with the end of the structure (Figure 1), and the joint angles are obtained from the endpoint of the effector.

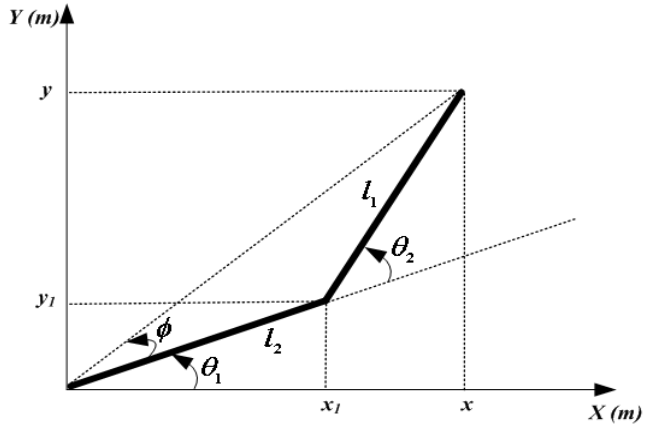

Figure 1. DoF robot arm structure and the angles defined with inverse kinematics

On this problem, the angles are obtained with the use of inverse kinematics algebraic methods, defined by the use of formulas (1) and (2).

$$\vartheta_1 = Tan^{-1}(x/y) - Tan^{-1}\left(\frac{\alpha_2 Sin\vartheta_2}{\alpha_1 + \alpha_2 Cos\vartheta_2}\right)$$

$$\vartheta_2 = Cos^{-1}\left(\frac{x^2 + y^2 - \alpha_1^2 - \alpha_2^2}{2\alpha_1\alpha_2}\right)$$

3. Neural Network Structure and Topology

The topology of the neural network used for this experiment is shown in (Figure 2). A feed-forward network, with two input, two hidden layers with 10 neurons for each layer and two output was implemented. The number of hidden layers and neurons remained constant during the experiment to isolate these variables, and only the activation functions and training algorithms were adjusted to compare results based on MSE.

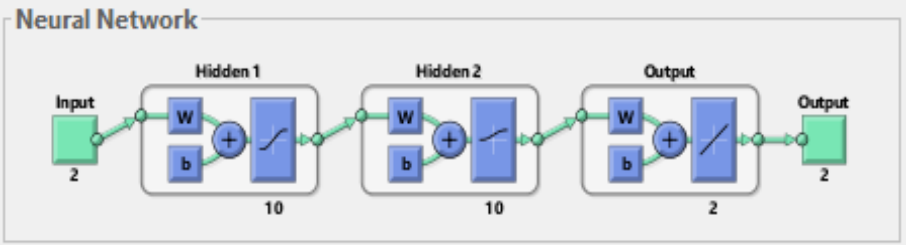

Figure 2. Topology of the Neural Network used for this experiment

As described during the introduction of this paper, the main objective is to train the artificial neural network with several training algorithms and compare results to define which one works best for this type of applications. Eleven different algorithms were

selected as candidates for this trial, their programming instruction and description for Matlab is shown in (Table 1).

Training Algorithm	Description
trainlm	Levenberg-Marquardt backpropagation
trainbr	Bayesian Regularization backpropagation
trainscg	Scaled conjugate gradient backpropagation
trainbfg	BFGS quasi-Newton backpropagation
traincgb	Conjugate gradient backpropagation with Powell-Beale restarts
traingd	Gradient descent backpropagation
traingda	Gradient descent with adaptive lr backpropagation
traingdm	Gradient descent with momentum
traingdx	Gradient descent w/momentum &adaptive lr backpropagation
trainoss	One step secant backpropagation
trainrp	RPROP backpropagation

Table 1. Training algorithms used for this experiment

Each training algorithm was programmed in the neural network and the test and simulation of the planar robot arm was run at least four times, and the activation functions were adjusted as described in (Table 2). Transfer Function 1 has connections from the first hidden layer and on to the second hidden layer. Transfer Function 2 has connections from the second hidden layer and on to the output layer, and Transfer Function 3 is the result of the output layer. Transfer Function 3 remained constant during the experiment as Purelin, which is the recommended activation function for linear and nonlinear problems.

TransferFcn 1	TransferFcn 2	TransferFcn 3
symmetric sigmoid	symmetric sigmoid	linear transfer
symmetric sigmoid	logarithmic sigmoid	linear transfer
logarithmic sigmoid	symmetric sigmoid	linear transfer
logarithmic sigmoid	logarithmic sigmoid	linear transfer

Table 2. Transfer Functions used on each hidden layer and output layer

The training data for this experiment is distributed in a planar environment and goes from 0 to π (Figure 3). The minimum and maximum reach of the robot arm is determined by the total length of each link and can be obtained by Lmin=L1-L2 and Lmax=L1+L2. For this particular problem, L1 is set to 10, while L2 is set to 5. Each point dataset is placed 3^{o} apart of each other, or $2\pi/60$.

During the training process and MSE validation, the error measurement and other critical parameters were obtained via the on-screen interface from Matlab, and the planar robot arm was simulated to visualize how well it can follow a pre-defined trajectory that doesn't necessarily contain the same data points that were used during the training process, as a way to validate the capability of the neural network to adapt to un-trained data. An example of the simulation is shown in (Figure 4).

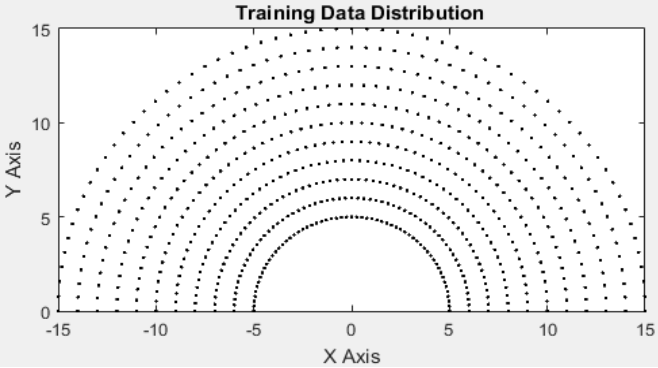

Figure 3. Training data distribution

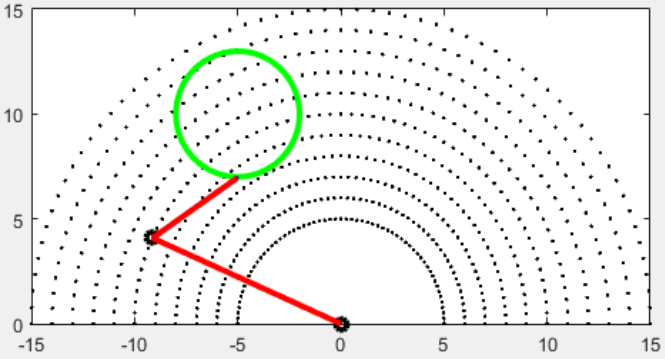

Figure 4. Simulation of the planar robot arm during trajectory follows

4. Results for comparison

During the training process, a few other parameters were set constant to isolate as many variables as possible. The error (MSE) was set to 0.0001, the number of epochs was set to 1000, and the training algorithms and transfer functions were adjusted as described before. The results are shown in (Table 3).

Training Algorithm	TransferFcn 1	TransferFcn 2	MSE	Epochs	Gradient	MU
trainlm	tansig	tansig	0.0000993	75	0.0003010	0.0000100
trainlm	tansig	logsig	0.0000969	87	0.0042900	0.0000010
trainlm	logsig	tansig	0.0000966	106	0.0133000	0.0000010
trainlm	logsig	logsig	0.0000981	73	0.0009800	0.0000010
trainbr	tansig	tansig	0.0000983	184	0.0108000	50.0000000
trainbr	tansig	logsig	0.0000895	35	0.0104000	5.0000000
trainbr	logsig	tansig	0.0000980	62	0.0511000	5.0000000
trainbr	logsig	logsig	0.0000961	90	0.0126000	5.0000000
trainscg	tansig	tansig	0.0059500	99	0.0399000	

trainscg	tansig	logsig	0.0137000	98	0.0325000	
trainscg	logsig	tansig	0.0065200	231	0.0168000	
trainscg	logsig	logsig	0.6269000	49	0.0199000	
trainbfg	tansig	tansig	0.0024300	139	0.0161000	
trainbfg	tansig	logsig	0.0038400	158	0.0190000	
trainbfg	logsig	tansig	0.0030700	142	0.0107000	
trainbfg	logsig	logsig	0.0028200	179	0.0021200	
traincgb	tansig	tansig	0.0102000	119	0.0209000	
traincgb	tansig	logsig	0.0019000	243	0.0029900	
traincgb	logsig	tansig	0.0470000	68	0.0923000	
traincgb	logsig	logsig	0.0103000	226	0.0094300	
traingd	tansig	tansig	0.0952000	1000	0.0811000	
traingd	tansig	logsig	0.1180000	1000	0.0851000	
traingd	logsig	tansig	0.2440000	1000	0.1040000	
traingd	logsig	logsig	0.2520000	1000	0.1060000	
traingda	tansig	tansig	0.0828000	79	0.5800000	
traingda	tansig	logsig	0.1620000	93	0.8220000	
traingda	logsig	tansig	0.1180000	91	0.8100000	
traingda	logsig	logsig	0.2360000	89	0.6840000	
traingdm	tansig	tansig	2.4800000	9	26.1000000	
traingdm	tansig	logsig	0.8440000	12	6.2200000	
traingdm	logsig	tansig	2.4400000	10	16.8000000	
traingdm	logsig	logsig	1.6000000	14	5.3800000	
traingdx	tansig	tansig	0.0820000	122	0.3520000	
traingdx	tansig	logsig	0.0663000	133	0.2960000	
traingdx	logsig	tansig	0.1160000	129	0.7140000	
traingdx	logsig	logsig	0.1100000	133	0.2450000	
trainoss	tansig	tansig	0.0272000	82	0.0518000	
trainoss	tansig	logsig	0.0155000	87	0.0352000	
trainoss	logsig	tansig	0.0529000	104	0.0653000	
trainoss	logsig	logsig	0.1520000	49	0.1750000	
trainrp	tansig	tansig	0.0063600	402	0.0074000	
trainrp	tansig	logsig	0.0090500	216	0.0106000	
trainrp	logsig	tansig	0.0087300	128	0.0135000	
trainrp	logsig	logsig	0.0170000	63	0.0396000	

Table 3. MSE results and measurements for comparison

5. Conclusions

The experiment shows some interesting results for the various training algorithms applied during the training process. As we can see in the results chart, there are only two algorithms that reached the proposed MSE, which are the Levenberg-Marquardt and Bayesian Regularization Backpropagation, all of the remaining algorithms show poor performance and higher error. The best result was achieved by a combination of Bayesian Regularization algorithm and tansig / logsig / purelin transfer functions, which resulted in an MSE of 0.0000895 in just 35 epochs. And we can consider this as the optimal combination during the training process of the 2DoF planar robot arm for SCARA applications.

References

[1] Lukas, Barinka., Roman, Berka (2002). Inverse Kinematics –Basic Methods. Department of Computer Science & Engineering Czech Technical University, 2002.

[2] Pedro, Ochoa., Hernandez, Ivan., Ramírez, Julio (2007). Simulation and Animation of a 2 Degree of Freedom Planar Robot Arm Based on Neural Networks, Fourth Congress of Electronics, Robotics and Automotive Mechanics, p. 488 –493, 2007.

[3] The Mathworks Inc, Matlab User Guide, R2016B.

[4] David, Hunter., Hao, Yu., Michael, Pukish., Janusz, Kolbusz., Bogdan, M., Wilamowski, Selection of Proper Neural Network Sizes and Architectures.