Fall 2012

Motion Learning Using The Neural Network

Priyank Shah
San Jose State University

Motion Learning Using The Neural Network

A Thesis
Presented to
The Faculty of the Department of Computer Science
San José State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
by
Priyank K Shah
September 2012

SAN JOSE STATE UNIVERSITY

The Designated Thesis Committee Approves the Thesis Titled

Motion Learning Using The Neural Network

by
Priyank K Shah

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

October 2012

Dr. Chris Tseng, Department of Computer Science	Date
Dr. Soon Tee Teoh, Department of Computer Science	Date
Mr. Grinish Engineer, Software Engineer(Cisco Systems)	Date

Acknowledgement

I would like to express my sincere gratitude to my project advisor Dr. Tseng for guidance, cooperation and time in making my project a success. I would also like to thank my committee members Dr. Teoh and Mr. Grinish Engineer for their support and patience. Also, sincere thanks to my family and friends for constant moral support.

ABSTRACT

One of the most promising topics of research in the field of artificial intelligence is the application of data captured from human motion using sensors processed with various algorithms to achieve successful data analysis. This project aims to design and develop a method to judge human motion and allows the users to see the score while they are performing motion. The Neural Network is trained to follow human expert scores on all players' motion profiles as compared with master player. This Neural Network is trained for its best performance by changing the number of iterations and the dataset passes through the network. Its sets the time period to train the Neural Network and sets the gradient so that error generated is a minimal or changing mean square error. The user data derived from a Yoga performance or any other motion was collected from Kinect sensors' data reports. The data collected contained X, Y and Z rotational and positional points. Kinect sensors captures 20 joints. The master data and user data were used as input to the Dynamic Time Warping algorithm. It compensated for the speed and time between the master and user data and gave a value that suggests the relative similarity of both motion profiles. The output of this algorithm was fed to Neural Network that was trained with a human judgment expert's data on motion profiles. This project is an attempt to train a Neural Network that will eventually judge like an expert and determine the success level that a user's motion profile exhibits.

Table of Contents

List of Tables

List of Figures

Introduction

Recently, several papers have been published on motion learning and how to improve it. The main goal of this motion learning research was to achieve a very accurate motion judgment as close to human judge as possible. Today, the number of people using various motion sensors for daily use is increasing.

There are several other studies that support that human motion collected from sensor data can be used to study motion learning patterns, artificial intelligence and to predict future motion.

An artificial Neural Network is a computational model derived from a biological concept of a Neural network. The network is formed of neurons that are interconnected to solve a specific network. The artificial neural network is configured for a specific application, data classification, motion learning, pattern recognition through a learning process. Training is done by various Neural Network algorithms and the adjustments of the training parameters.

Motion learning is a basic operation in the selection of significant segments of video signals. In this project, an efficient neural network system is proposed for motion detection from the static background. This method mainly consists of three parts: Collecting Motion Data using Kinect; processing data to remove speed and time variants using Dynamic Time Warping algorithm; feeding the data obtained from DTW to the Neural Network Backpropagation algorithm to the trained network; and finally, using this trained Network to judge human motion.

The goal of the project is motion learning and the application of the Neural Network algorithm to better judge human motion. Motion learning is implemented by using a forward Neural Network that has been trained accordingly. During training, the network is trained to associate output with input patterns. When the network is used, it identifies the input pattern and tries to output the associated output pattern. The power of neural networks comes to life when a pattern that has no output associated with it, is given as an input. In this case, the network gives the output that corresponds to a taught input pattern that is least different from the given pattern. The input data is taken from Kinect sensors. The captured data is taken as input to the Dynamic Time Warping algorithm, the output of which is fed to Backpropagation

Neural Network. With the help of this trained Network, a person is judged in real time, which accordingly can be used to improve the motion. In this project, the trained Network gives score as a person actions or movements, such as Yoga. The scored judgment can help a person improve future movements.

2. Literature Review

In this section we discuss various concepts like the Dynamic Time Warping algorithm, the Backpropagation algorithm, the parameters to train Neural Network and the Matlab Toolbox.

Dynamic Time Warping

Dynamic time warping (DTW) is used for measuring two sequences which may vary in time and speed. The DTW algorithm is kind of dynamic programming approach. It forms a matrix and takes minimum path so that two wave forms match the original wave form [11].

The DTW algorithm [11]

```
int DTWDistance(char s[1..n], char t[1..m]) {
   declare int DTW[0..n, 0..m]
   declare int i, j, cost

   for i := 1 to m
      DTW[0, i] := infinity
   for i := 1 to n
      DTW[i, 0] := infinity
   DTW[0, 0] := 0

   for i := 1 to n
      for j := 1 to m
         cost:= d(s[i], t[j])
```

```
        DTW[i, j] := cost + minimum(DTW[i-1, j  ],    // insertion
                                    DTW[i  , j-1],    // deletion
                                    DTW[i-1, j-1])    // match


   return DTW[n, m]
}
```

Specifications of algorithms

Char s[1...n] , char t[1...m] -- input data for the DTW algorithm.

Here i and j are taken as row and column.

Initial conditions

In the initial condition i from 1 to m are specified as infinity similarly, j from 1 to m are specified as infinity. Also, most importantly, DTW [0, 0] is taken (0, 0). Cost d(s[i], t[j]) is d(x, y) is a distance between symbols, i.e. d(x, y) = | x - y |. Now DTW is calculated as cost plus minimum value from (i-1,j) (i, j-1),(i-1,j-1) . The algorithm forms a matrix. Once a matrix is formed, it uses a backtracking algorithm to form the points and plot the graph.

Example of DTW algorithm

Here cost is considered as 1.

	i/j	B	C	A	D	B	C
	0	1	2	3	4	5	6
A	1	1	2	3	4	5	6
C	2	2	2	3	4	5	6
B	3	3	3	3	4	5	6
D	4	4	4	4	4	5	6
A	5	5	5	5	5	5	6
C	6	6	6	6	6	6	6

AS shown from the arrows, we can back trace and form the graph. Here the matrix is formed starting from the left hand corner, if the alphabet matched 1 is added to cell (i-1)(j-1). If they are not equal than greater of cell (i-1)(j) or (i)(j-1) is taken to fill the next cell.

Kinect Data

Joints captured by Kinect motion sensor

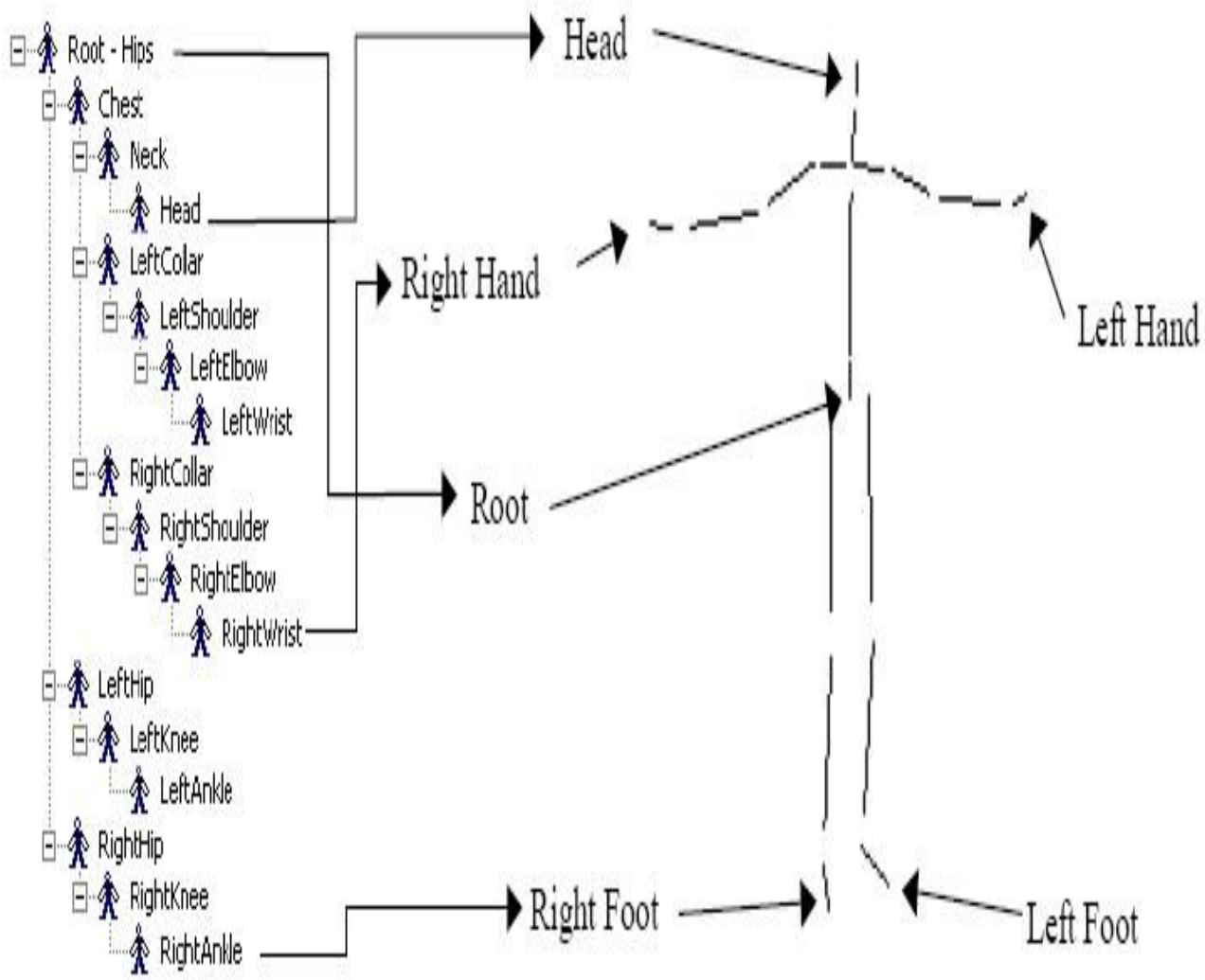


Fig 1: Kinect captured points [16]

Kinect gives 6 data points for a particular joint with X, Y and Z rotation and position. In this project, we only captured the X, Y and Z rotation, because taking position into consideration for calculating the Neural Network is potentially inaccurate as X , Y and Z position varies as it is based on the height of a person. So, for the training data the X, Y and Z rotation details were

taken. The problem of a person having different height was eliminated since the rotation does not come into picture if person is having different height.

In this project, one study is of master yoga posture data files and the other is data file for student data. They were fed into the DTW algorithm to get 9 points of 3 joints of a right shoulder. The collected data from Kinect was fed to the DTW before inputing to the Neural Network. It was fed into the DTW to get around 150 points of frame data into a single point (since the Neural Network can't handle too many inputs). In our case, we considered 3 joints, therefore a total of 9 data points were used to feed the Neural Network.

Sample 1 frame data from Kinect

AnkleRight: 0.09563106 -0.6953287 3.490368

AnkleLeft: -0.07453345 -0.7462188 3.511517

KneeRight: 0.1040133 -0.4055198 3.463443

KneeLeft: -0.1025893 -0.4003633 3.476388

HipRight: 0.08443671 0.03220186 3.538582

HipLeft: -0.07134762 0.03197961 3.537938

ShoulderRight: 0.1783416 0.429853 3.630216

ShoulderLeft: -0.1617673 0.4396111 3.647002

ElbowRight: 0.422052 0.4634577 3.648444

ElbowLeft: -0.4061069 0.4651955 3.662422

WristRight: 0.6802884 0.4880981 3.60269

WristLeft: -0.62367 0.4694732 3.643325

This data is the X, Y and Z rotation.

Neural Network

The Neural Network refers to a network or circuit of biological neurons. An artificial Neural Network is composed of interconnecting neurons that are used to solve real biological neural problems or an artificial intelligence problem [9].

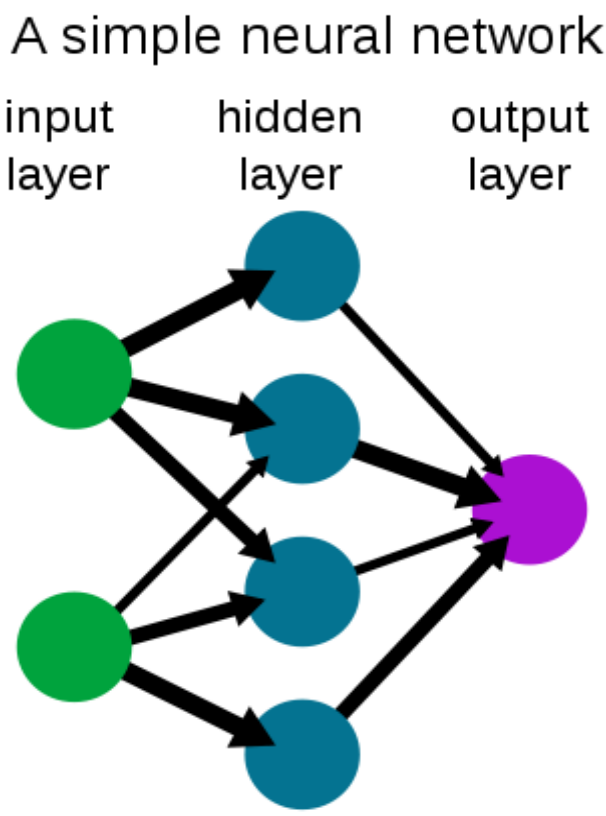


Fig 2: Neural Network Layers[9]

An artificial Neural Network is an adaptive system which adapts to changes based on external and internal information that flows through the network [2]. In a Neural Network, simple nodes are connected together to form network nodes. There can be any number of inputs, hidden and output layers in the network. The Neural Network is trained to predict what the behavior will be if similar or different information is fed to the Network [2].

Backpropagation algorithm

Backpropagation is a method of training an artificial neural network so as to minimize the objective function [10]. It is a machine learning method. It requires a dataset to train the network to a desired output. There are two ways to train the Neural Network using Backpropagation, incremental and batch learning. The weight update is followed immediately after each propagation for incremental learning. However, the batch learning weight update occurs after many propagations.

Backpropagation involves two phases:

Phase 1: Propagation

Each propagation involves a forward propagation of a training pattern's input through the Neural Network and backward propagation of the propagation's output activations through the Neural Network using the training pattern’s target [15].

Phase 2: Weight update

For each weight-synapse: follow the multiplication of its output delta and input activation to get the gradient of the weight and bring the weight in the opposite direction of the gradient by subtracting a ratio of it from the weight [15].

Matlab Neural Network Toolbox

The work flow of the Neural Network design process contains steps of collecting data, creating the network, configuring the network, initializing weights, training and validating the network. The Neural Network toolbox uses a network object to store all the information that the user defines while training the network. There are many algorithms that are built in the toolbox like Levenberg-Marquardt, Bayesian Regularization and Resilient backpropagation. The training of Neural Network is limited based on various parameters: the like Minimum Gradient Magnitude, Maximum Training Time, the Minimum Performance Value, and the Maximum number of epoch among others. The trained Network can be analyzed by a graph generated by a toolbox like Performance, Training state, Error Histogram and Regression.

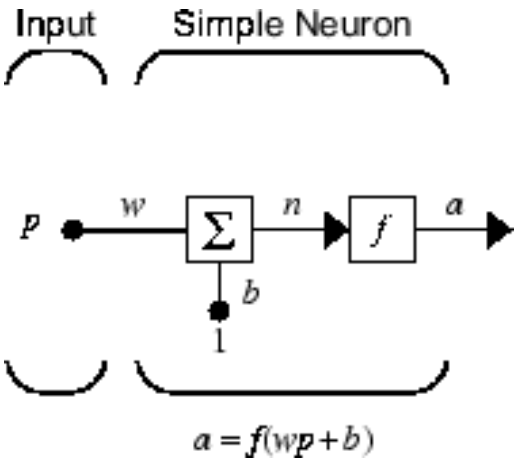


Figure 3: Simple Neuron [12]

Neural Network training parameters

Epoch - An epoch is a single pass through the entire data set. It iterates through the process by providing the network with input and updating the weights. The Linear network can be trained to perform a linear classification with the function train. The train function takes a set of input vectors and calculates the network bias increments and weight due to each of the inputs according to learnp. Based on the sum of all these corrections the network is adjusted [12]. Either train or adapt is used to give epoch.
example: net.trainParam.epochs = 100;

Mean squared error (MSE) – MSE gives the average of squares of errors. Error is the difference between the value implied by an estimator from the quantity to be estimated. MSE is a risk function, corresponding to the expected value of the squared error loss or quadratic loss [17]. The output layer of the Network learns to match the associated target vectors with a minimal mean squared error [12]. Either train of epoch is used to set the mean squared error.
example: net.trainParam.goal = 0.003;

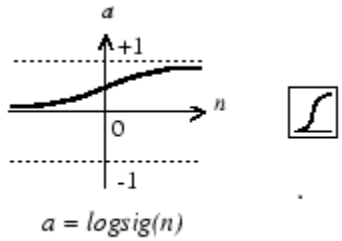
Sigmoid function - A sigmoid function is an 'S' shaped curve produced by a mathematical function. The Sigmoid function includes tansig, logsig and purelin and many others. A sigmoid function is real- valued and differentiable, having either a non-negative or non-positive first derivative [18].The Sigmoid function is required for both the hidden layer and output layers.

Implementation

The human motion data was collected using Microsoft Kinect. The Single joint data contained 6 points representing the X, Y and Z positional and rotational axis. We only considered the X, Y and Z rotational axis, since different users will have different heights; and comparing the master profile with user profile in terms of positional axis would give inaccurate results. The three joints of user's motion profile were taken into account. The Dynamic Time Warping Algorithm and the Neural Network code were implemented in Matlab.

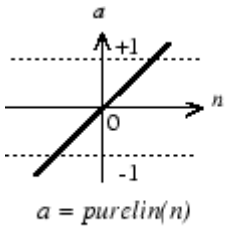
First, the data was collected from the human expert to train the Neural Network by using the Feedforward Backpropagation algorithm. This trained Network was used to predict scores of the user's motion profiles' as compared to those of the master player. The User's motion profiles were captured via Kinect Sensors. The data collected from Kinect contained over 100 frames that represented each joint positions for just 5 seconds of motion. Since the Neural Network couldn't handle this huge amount of data, the Dynamic Time Warping algorithm was used. The Master data and User data were used as input to the Dynamic Time Warping algorithm. It compensated for the speed and time between the Master and User's data. The value as closer to zero suggests the closeness of both the motion profiles. The value was close to zero suggested that both motion profile were similar. The output of the Dynamic Time Warping algorithm contained three values for each joint pertaining to X, Y and Z rotational axis. The output of this algorithm was fed to the Neural Network that was trained with human judgment expert data on motion profiles. It eventually gave scores that depended on how close a user was compared to the master. A transfer function, like the tansig, logsig and purelin function, was taken as input for hidden layer of Neurons. The natural finite bounds for tansig, logsig and purelin are [-1,1], [0,1] and [-infinity, + infinity] respectively.

Below are the curves of these three functions:



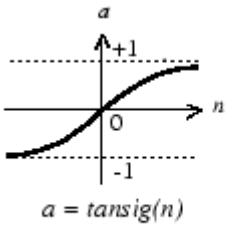
Log-Sigmoid Transfer Function

Figure 3: logsig



Linear Transfer Function

Figure 4: purelin



Tan-Sigmoid Transfer Function

Figure 5: tansig

A different number of hidden layers of Neurons were considered and coupled with a different transfer function for training. To complete the training of the Network mainly the performance (goal) and epoch were taken into account. The trained Neural Network was tested with both untrained data means, data not taken into account for training and trained data means. data used in training the Neural Network.

In this project the feed - forward backpropagation network was used to train the Network. The command to create the backpropagation network is:

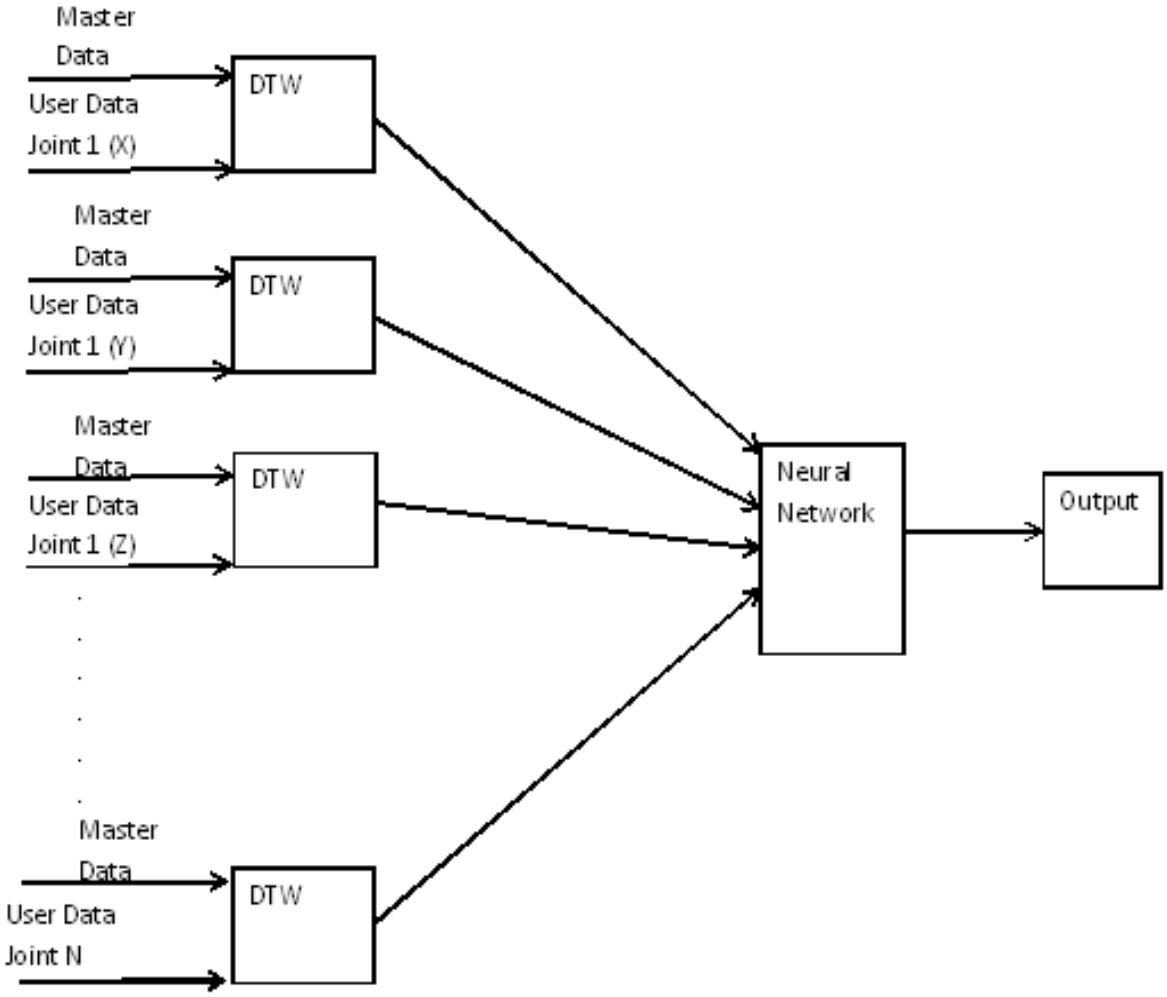
net = newff (PR, [S1 S2..... SN1], {TF1 TF2...TFN1},BTF,BLF, PF)

where net = newff creates a new network, PR is R*2 matrix of minimum and maximum values for R input elements, Si is size of i^{th} layer for N1 layers, TFi is transfer function of the ith layer, BTF is backpropagation network training function, BLF is backpropagation weight/bias learning function and PF is performance function [19].

Example net = newff([0 10], [5 1], {'tansig' ' purelin'});

In this case, the two layer feed-forward network was created. There were five hidden layer neurons with tansig as a transfer function and one output layer neuron with purelin as sigmoid function. The network input layer ranged from [0 to 10].

Flow Diagram of Project



Results

Tables Description

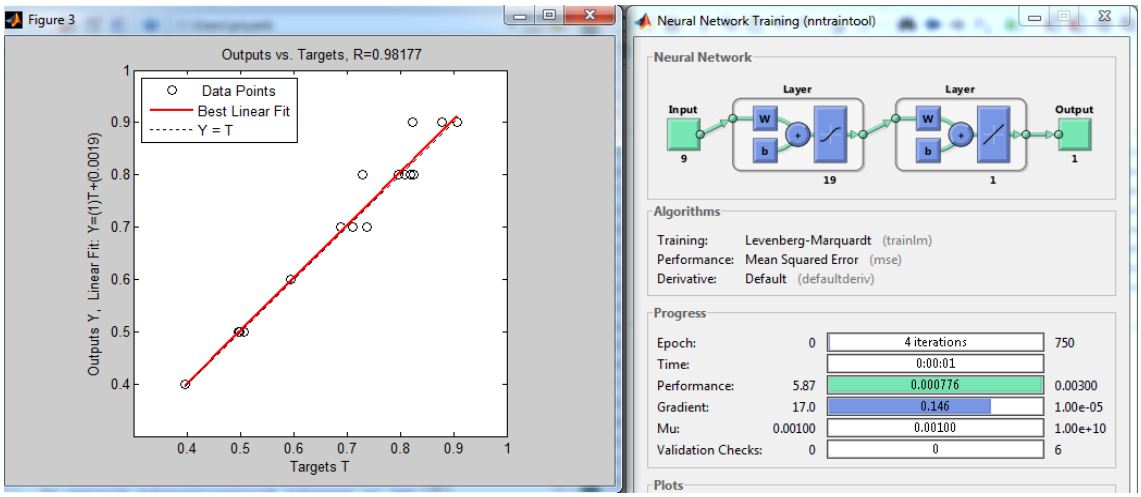
This tables represents different scenarios taken into consideration to test the Neural Network. The applied parameters considered to stop Neural Network training once the best performances was achieved were: Sigmoid function, Mean Square Error, Gradient and number of neurons used. The Master and user profile were compared for three joints.

Here, the first column represents a human judge score that is given by an expert. The second column represents the output of application. The third column represent difference between human judge score and the application output that shows the similarity of the output of the Neural Network. The fourth column represents the number of neurons used in the training application. The Neural Network sigmoid function represents the functions that are used to train the Network. The goal column represents the mean square error. The epoch tells the number of times the data is passed to the network. The gradient column represents the value set, so that the error generated is a minimum or acceptable value. The last column's average difference represents the sum of the difference between a human judge and the Neural Network output divide by the total number of motion profiles.

Here, each table represents different kind of criteria that can be used to stop training after a certain expected performance is reached. Parameters are taken into consideration, as well as the average difference of error generated when these parameters were taken to train the network. The Lower the average difference indicates that network has been trained properly enough to give the closest possible score of user's motion profile to that of the human judge.

Scenario 1: Data not used in Neural Network training

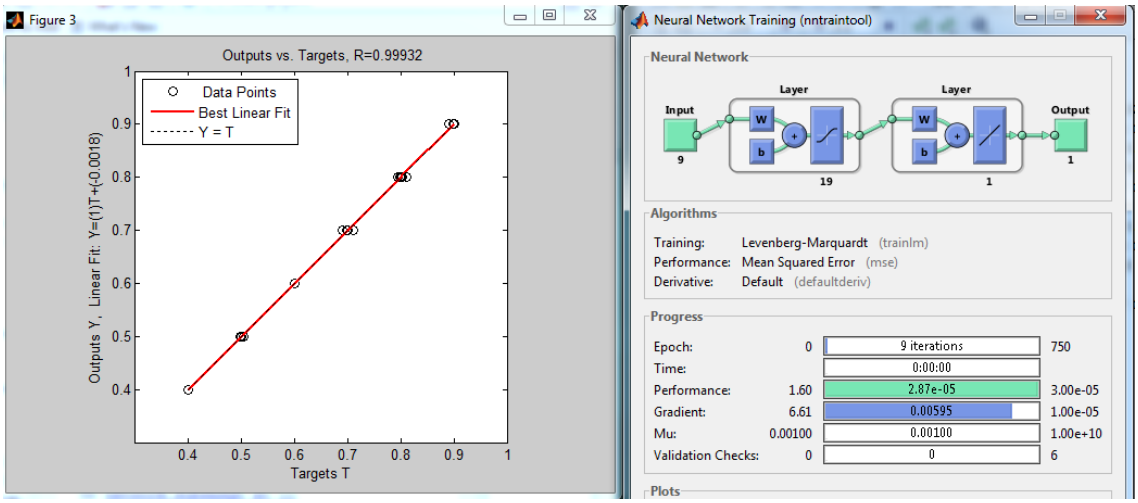
1) Here, the goal is kept 0.003, the tansig is used as a sigmoid function and the number of neurons is kept as 19 for training the Neural Network. The average difference of errors got was 0.0832.



Human Judge Score	NN output	Difference	NN neurons used	NN sigmoid function	goal	epoch	Average Difference
0.7	0.746	0.046	19	tansig	0.003	750	0.0832
0.8	0.664	0.136					
0.7	0.723	0.023					
0.8	0.548	0.252					
0.9	0.887	0.013					
0.7	0.8399	0.139					
0.7	0.642	0.058					
0.7	0.742	0.058					
0.5	0.468	0.032					
0.5	0.458	0.042					
0.4	0.433	0.033					
0.6	0.489	0.111					
0.5	0.592	0.092					
0.8	0.767	0.033					
0.7	0.724	0.024					
0.7	0.687	0.013					
0.7	0.731	0.031					
0.5	0.498	0.002					

Table 1: Result of untrained data for goal 0.0003

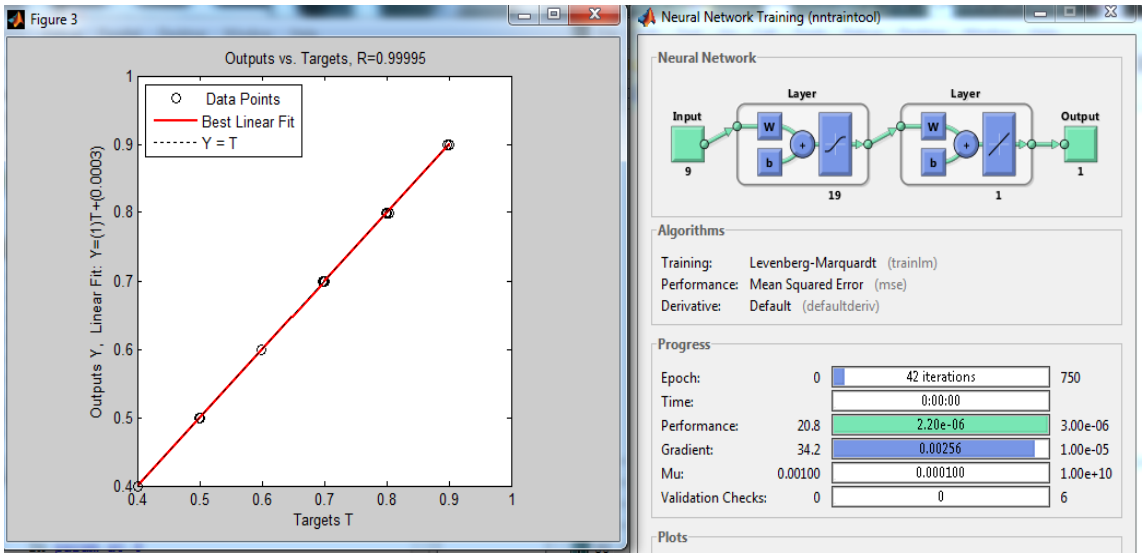
2) Here, the goal is increased 0.00003 to try to decrease the average difference of errors, tansig is used as the sigmoid function and the number of neurons is kept as 19.



Human Judge Score	NN output	Difference	NN neurons used	NN sigmoid function	goal	epoch	Average Difference of errors
0.7	0.740	0.040	19	tansig	0.00003	750	0.0376
0.8	0.714	0.086					
0.7	0.706	0.006					
0.8	0.684	0.016					
0.9	0.912	0.012					
0.7	0.699	0.001					
0.7	0.686	0.014					
0.7	0.696	0.004					
0.5	0.459	0.041					
0.5	0.534	0.034					
0.4	0.417	0.017					
0.6	0.571	0.029					
0.5	0.564	0.036					
0.8	0.686	0.114					
0.7	0.695	0.005					
0.7	0.704	0.004					
0.7	0.672	0.028					
0.5	0.566	0.066					

Table 2: Result of untrained data for goal 0.00003

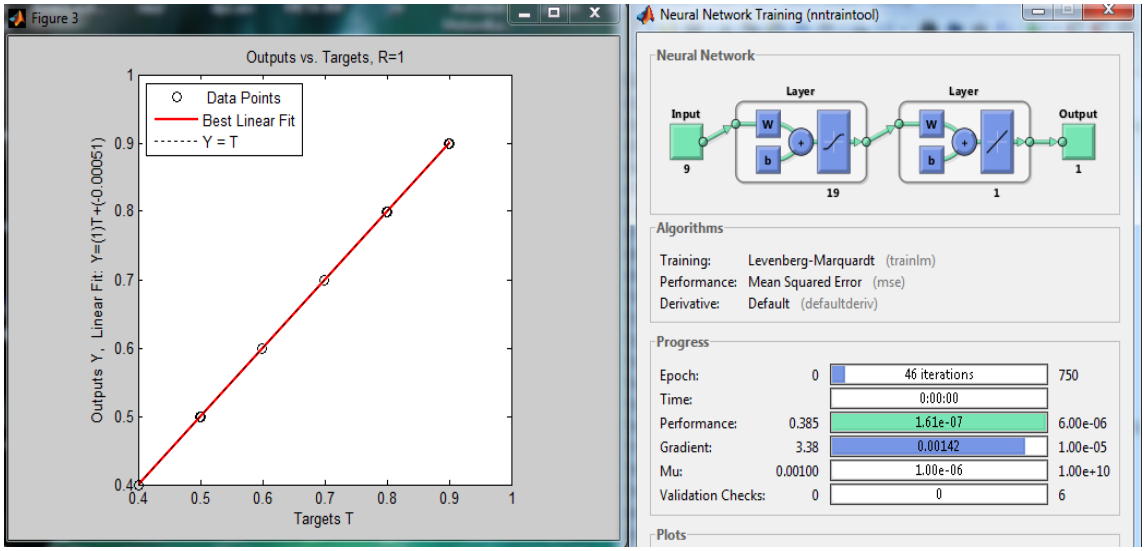
3) Here, the goal is decreased by an additional 0.000003 to improve output of neural network, tansig is used as a sigmoid function and number of neurons is kept as 19.



Human Judge Score	NN output	Difference	NN neurons used	NN sigmoid function	goal	epoch	Average Difference Of errors
0.7	0.744	0.044	19	tansig	0.000003	750	0.0252
0.8	0.793	0.006					
0.7	0.692	0.008					
0.8	0.818	0.018					
0.9	0.966	0.036					
0.7	0.702	0.002					
0.7	0.675	0.025					
0.7	0.706	0.006					
0.5	0.518	0.018					
0.5	0.507	0.007					
0.4	0.394	0.006					
0.6	0.549	0.051					
0.5	0.514	0.014					
0.8	0.747	0.053					
0.7	0.654	0.046					
0.7	0.736	0.064					
0.7	0.685	0.015					
0.5	0.533	0.067					

Table 3: Result of untrained data for goal 0.000003

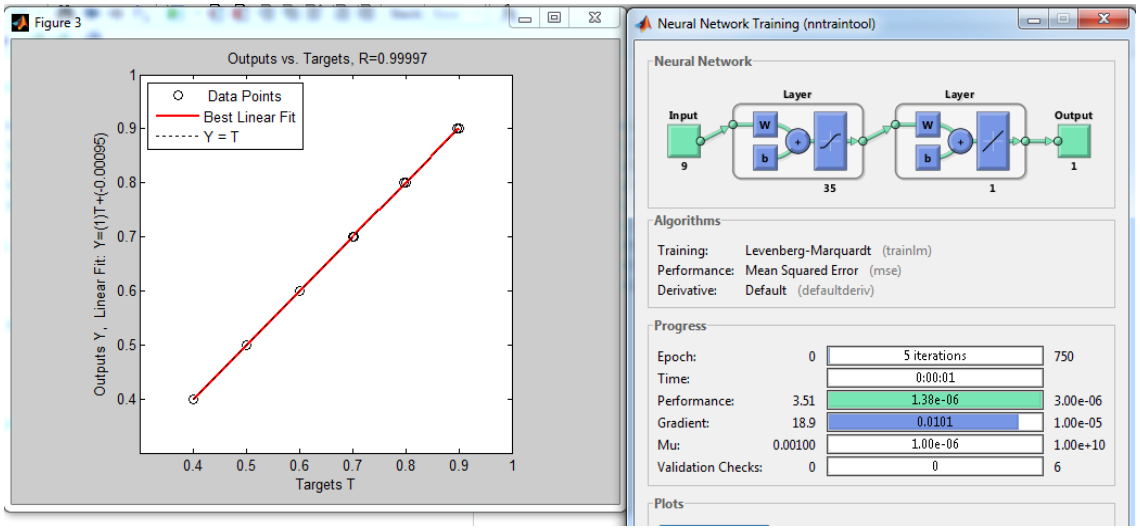
4) Here, we still have decreased further to 0.000006 , tansig is used as sigmoid function and number of neurons is kept as 19. Here we can see the average difference of error is increased compared to 0.000003. So we stopped here decreasing the goal.



Human Judge Score	NN output	Difference	NN neurons used	NN sigmoid function used	goal	epoch	Average difference of errors
0.7	0.749	0.049	19	tansig	0.000006	750	0.0928
0.8	0.773	0.027					
0.7	0.705	0.005					
0.8	0.754	0.046					
0.9	0.884	0.016					
0.7	0.747	0.047					
0.7	0.705	0.05					
0.7	0.680	0.020					
0.5	0.682	0.182					
0.5	0.694	0.194					
0.4	0.721	0.321					
0.6	0.868	0.268					
0.5	0.588	0.088					
0.8	0.709	0.090					
0.7	0.720	0.020					
0.7	0.744	0.044					
0.7	0.609	0.090					
0.5	0.614	0.114					

Table 4: Result of untrained data for goal 0.000006

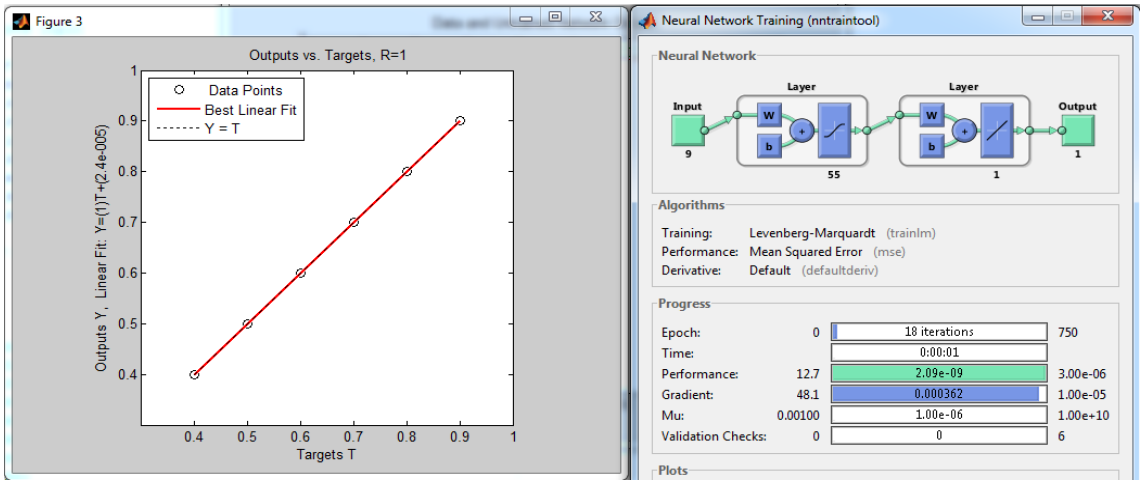
5) Here, the goal is kept 0.000003, tansig is used as the sigmoid function and the number of neurons is increased to 35. But, we can see that by increasing the number of neurons the average difference of errors increased.



Human Judge Score	NN output	Difference	NN neurons used	NN sigmoid function	goal	epoch	Average difference of errors
0.7	0.744	0.044	35	tansig	0.000003	750	0.042
0.8	0.803	0.003					
0.7	0.698	0.002					
0.8	0.564	0.236					
0.9	0.915	0.015					
0.7	0.754	0.054					
0.7	0.696	0.004					
0.7	0.692	0.008					
0.5	0.491	0.011					
0.5	0.470	0.030					
0.4	0.396	0.004					
0.6	0.506	0.090					
0.5	0.534	0.066					
0.8	0.689	0.111					
0.7	0.717	0.017					
0.7	0.679	0.021					
0.7	0.708	0.008					
0.5	0.532	0.032					

Table 5: Result of untrained data for 35 hidden layer neurons

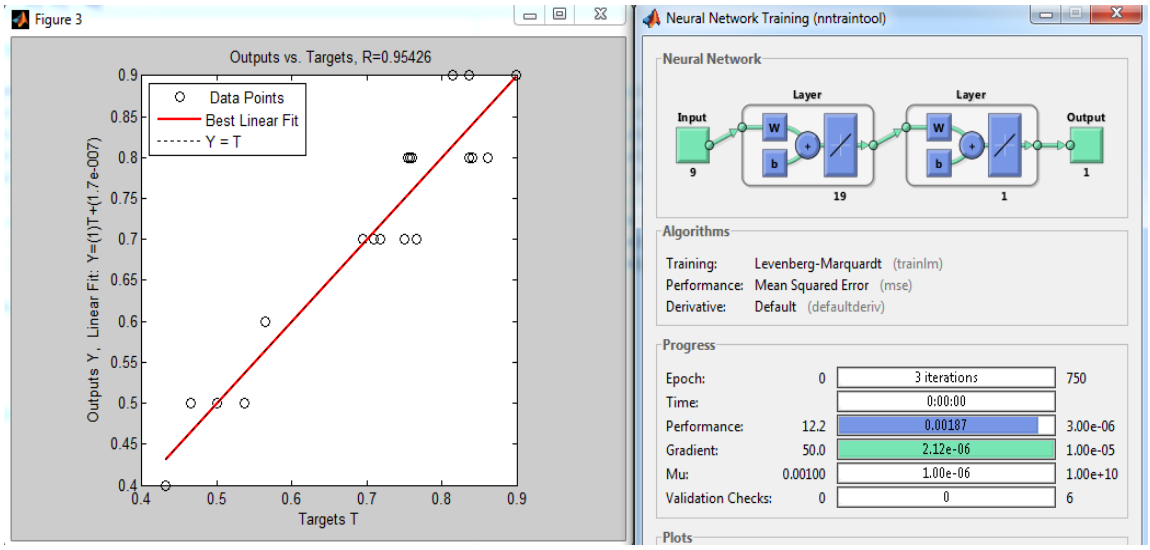
6) Here, the goal is kept 0.000003, tansig is used as a sigmoid function and number of neurons is increased to 55. But, we can see that increasing the number of neurons averages the difference of errors increased.



Human Judge Score	NN output	Difference	NN neurons used	NN sigmoid function	goal	epoch	Average difference of errors
0.7	0.742	0.042	55	tansig	0.000003	750	0.056944
0.8	0.781	0.019					
0.7	0.697	0.003					
0.8	0.477	0.323					
0.9	0.934	0.034					
0.7	0.640	0.060					
0.7	0.711	0.011					
0.7	0.707	0.007					
0.5	0.527	0.027					
0.5	0.534	0.034					
0.4	0.510	0.110					
0.6	0.545	0.055					
0.5	0.555	0.055					
0.8	0.720	0.080					
0.7	0.738	0.038					
0.7	0.632	0.070					
0.7	0.674	0.025					
0.5	0.532	0.032					

Table 6: Result of untrained data for 55 hidden layer neurons

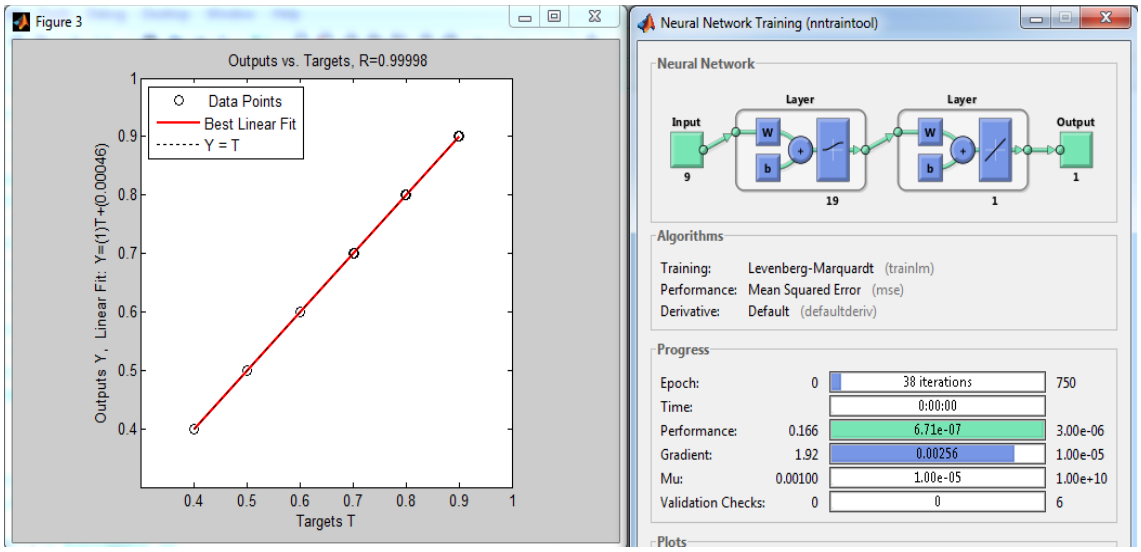
7) Here, the goal is kept 0.000003, the number of neurons is 19 and purelin is used as a sigmoid function. Here the error increases compared to the tansig function.



Human Judge Score	NN output	difference	NN neurons used	NN sigmoid function used	goal	epoch	Average difference of errors
0.7	0.698	0.002	19	purelin	0.000003	750	0.074722
0.8	0.521	0.270					
0.7	0.758	0.058					
0.8	0.724	0.070					
0.9	0.838	0.062					
0.7	0.822	0.122					
0.7	0.694	0.006					
0.7	0.759	0.059					
0.5	0.382	0.118					
0.5	0.513	0.013					
0.4	0.487	0.013					
0.6	0.446	0.154					
0.5	0.684	0.184					
0.8	0.845	0.045					
0.7	0.809	0.109					
0.7	0.729	0.029					
0.7	0.710	0.010					
0.5	0.534	0.034					

Table 7: Result of untrained data for purelin as the sigmoid function

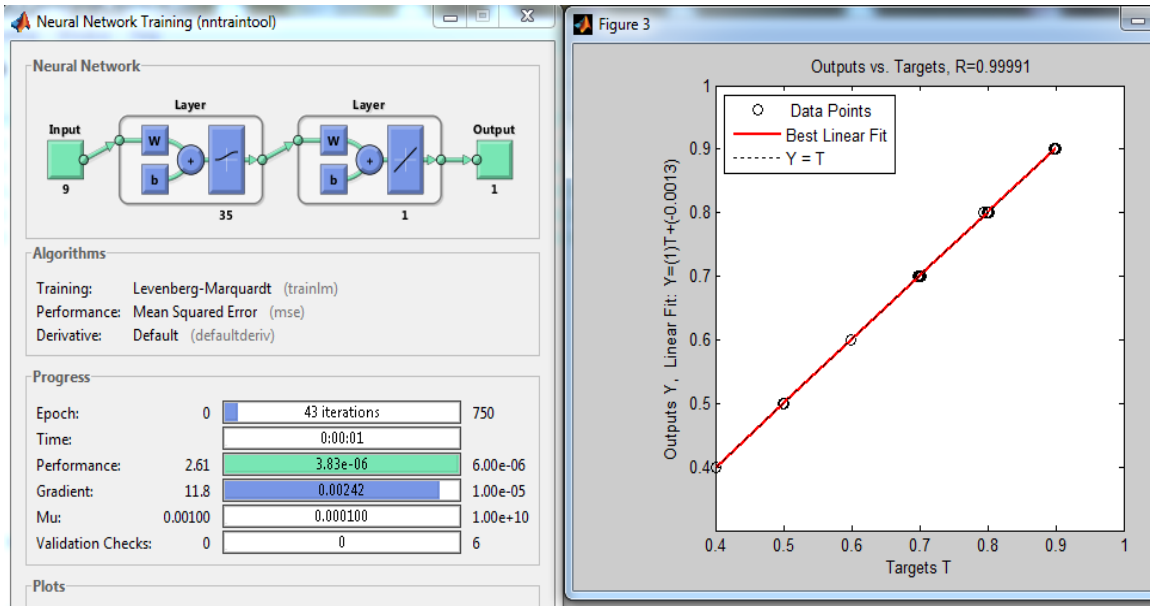
8) Here, the goal is kept 0.000003, the number of neurons is 19 and logsig is used as sigmoid function. Here, the error increases compared to the tansig function.



Human Judge Score	NN output	Difference	NN neurons used	NN sigmoid function	goal	epoch	Average difference of errors
0.7	0.738	0.038	19	logsig	0.000003	750	0.056833
0.8	0.824	0.024					
0.7	0.695	0.005					
0.8	0.619	0.180					
0.9	0.918	0.018					
0.7	0.621	0.070					
0.7	0.702	0.002					
0.7	0.644	0.046					
0.5	0.401	0.099					
0.5	0.422	0.078					
0.4	0.514	0.114					
0.6	0.602	0.002					
0.5	0.464	0.036					
0.8	0.629	0.170					
0.7	0.621	0.070					
0.7	0.671	0.030					
0.7	0.695	0.005					
0.5	0.564	0.036					

Table 8: Result of untrained data for logsig as a sigmoid function

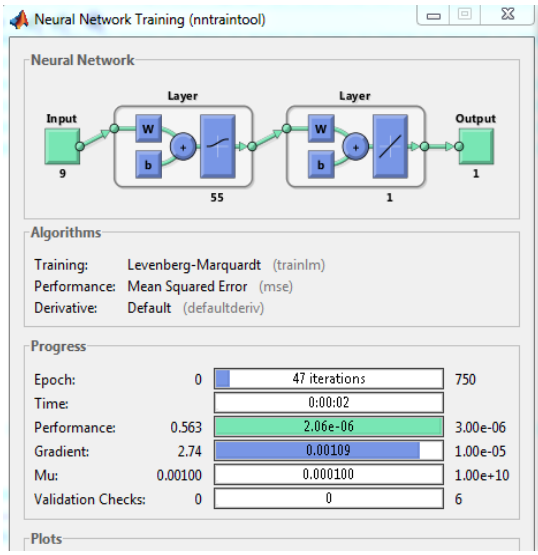
9) Here, the goal is kept 0.000003,the number of neurons is increased to 35 and logsig is used as a hidden layer sigmoid function. Here error increases compared to the tansig function.

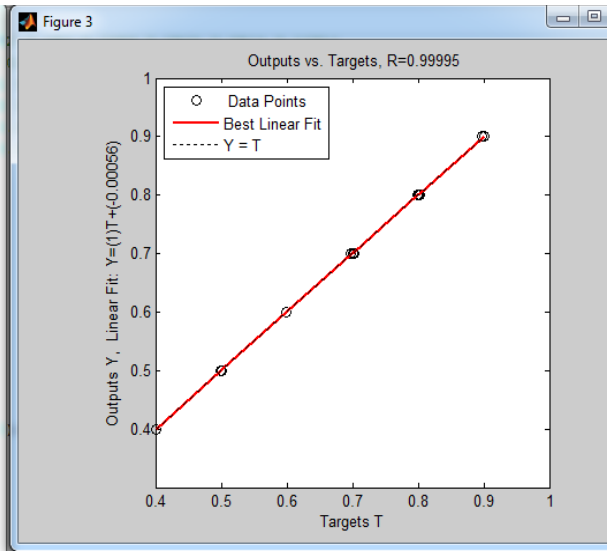


Human Judge Score	NN output	Difference	NN neurons used	NN sigmoid function	goal	epoch	Average Difference of errors
0.7	0.735	0.035	35	logsig	0.000003	750	0.0733
0.8	0.757	0.043					
0.7	0.707	0.007					
0.8	0.469	0.330					
0.9	0.906	0.006					
0.7	0.587	0.113					
0.7	0.722	0.078					
0.7	0.664	0.036					
0.5	0.380	0.120					
0.5	0.462	0.038					
0.4	0.400	0					
0.6	0.377	0.233					
0.5	0.533	0.033					
0.8	0.653	0.147					
0.7	0.699	0.001					
0.7	0.632	0.068					
0.7	0.723	0.023					
0.5	0.516	0.016					

Table 9: Result of untrained data for logsig as a sigmoid function and 35 as hidden layer neurons

10) Here, the goal is kept 0.000003,the number of neurons is increased to 55 and logsig is used as a hidden layer sigmoid function. Here, the error increases compared to the tansig function.

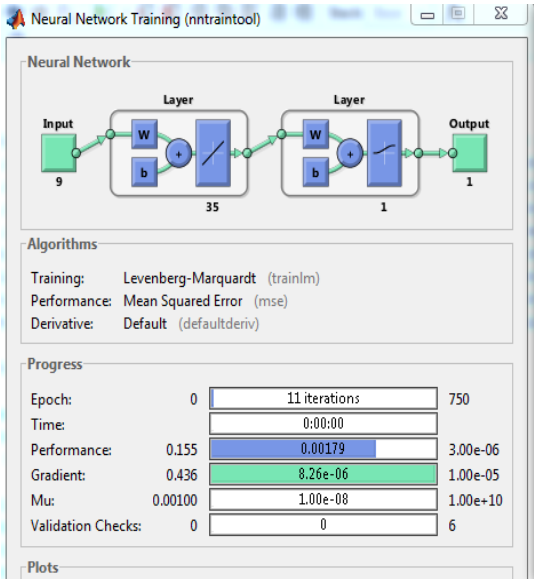


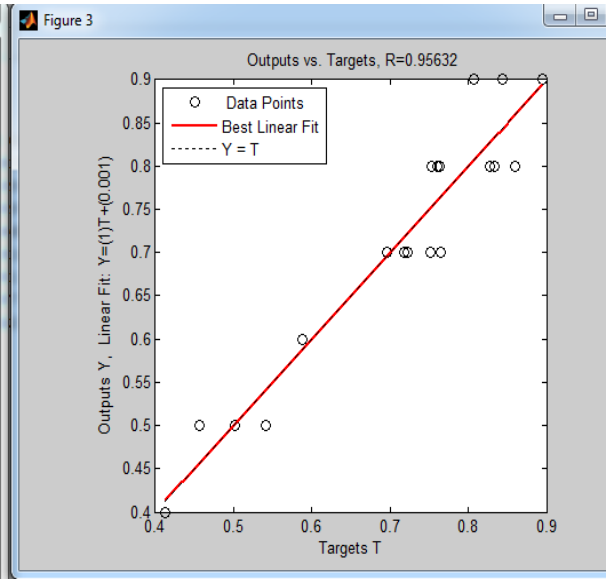


Human Judge Score	NN output	Difference	NN neurons used	NN sigmoid function	goal	epoch	Average Difference
0.7	0.743	0.043	55	logsig	0.000003	750	0.039
0.8	0.766	0.034					
0.7	0.701	0.001					
0.8	0.598	0.102					
0.9	0.934	0.066					
0.7	0.632	0.068					
0.7	0.683	0.017					
0.7	0.691	0.009					
0.5	0.470	0.030					
0.5	0.467	0.033					
0.4	0.461	0.039					
0.6	0.550	0.050					
0.5	0.525	0.025					
0.8	0.699	0.101					
0.7	0.675	0.025					
0.7	0.689	0.011					
0.7	0.710	0.010					
0.5	0.540	0.040					

Table 10: Result of Untrained data for logsig as a sigmoid function and 55 as hidden layer neurons

11) Here, the goal is kept 0.000003, the number of neurons is increased to 35 and purelin is used as a hidden layer sigmoid function. Here, the error increases compared to the tansig function.

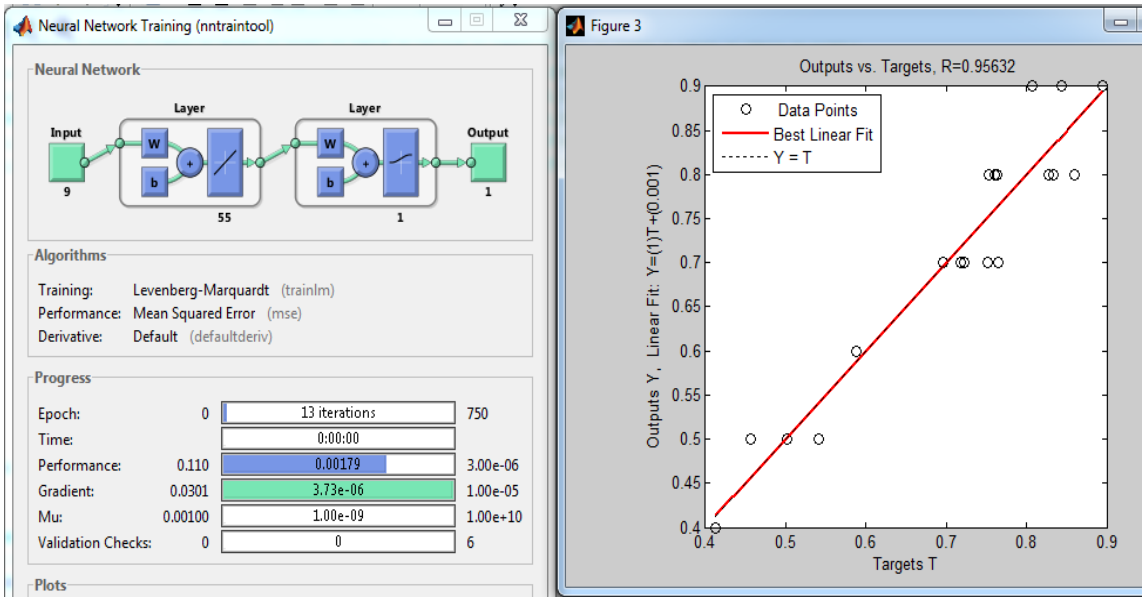




Human Judge Score	NN output	Difference	NN neurons used	NN sigmoid function	goal	epoch	Average difference of errors
0.7	0.698	0.002	35	purelin	0.000003	750	0.087
0.8	0.466	0.334					
0.7	0.759	0.041					
0.8	0.735	0.065					
0.9	0.833	0.067					
0.7	0.855	0.155					
0.7	0.713	0.013					
0.7	0.766	0.034					
0.5	0.343	0.157					
0.5	0.499	0.001					
0.4	0.502	0.102					
0.6	0.427	0.167					
0.5	0.693	0.193					
0.8	0.848	0.048					
0.7	0.807	0.107					
0.7	0.746	0.046					
0.7	0.721	0.021					
0.5	0.513	0.013					

Table 11: Result of Untrained data for purelin as a sigmoid function and 35 as hidden layer neurons

12) Here, the goal is kept 0.000003, the number of neurons is increased to 55 and logsig is used as a hidden layer sigmoid function. Here, the error increases as compared to the tansig function.



Human Judge Score	NN output	Difference	NN neurons used	NN sigmoid function	goal	epoch	Average difference of errors
0.7	0.698	0.002	55	purelin	0.000003	750	0.0901
0.8	0.466	0.334					
0.7	0.759	0.059					
0.8	0.735	0.065					
0.9	0.833	0.067					
0.7	0.855	0.155					
0.7	0.713	0.013					
0.7	0.766	0.066					
0.5	0.343	0.157					
0.5	0.499	0.001					
0.4	0.502	0.102					
0.6	0.427	0.173					
0.5	0.693	0.193					
0.8	0.848	0.048					
0.7	0.807	0.107					
0.7	0.746	0.046					
0.7	0.721	0.021					
0.5	0.513	0.013					

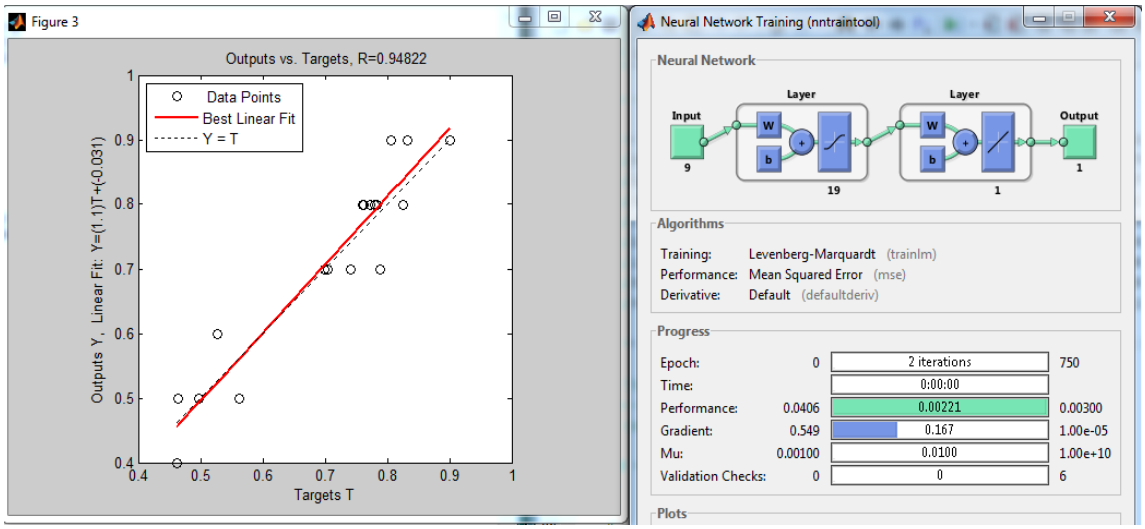
Table 12: Result of Untrained data for purelin as a sigmoid function and 55 as hidden layer neurons

Testing with data used in training

These tables represent the changing goal , the number of neurons and sigmoid function to trained neural networks until the best result is achieved. The tables represents different parameters used to stop the training of Neural Network, until best result was not achieved. Here, the data is taken that was already used in training.

Scenario 2 : The data of user's motion profile was used in training the Network

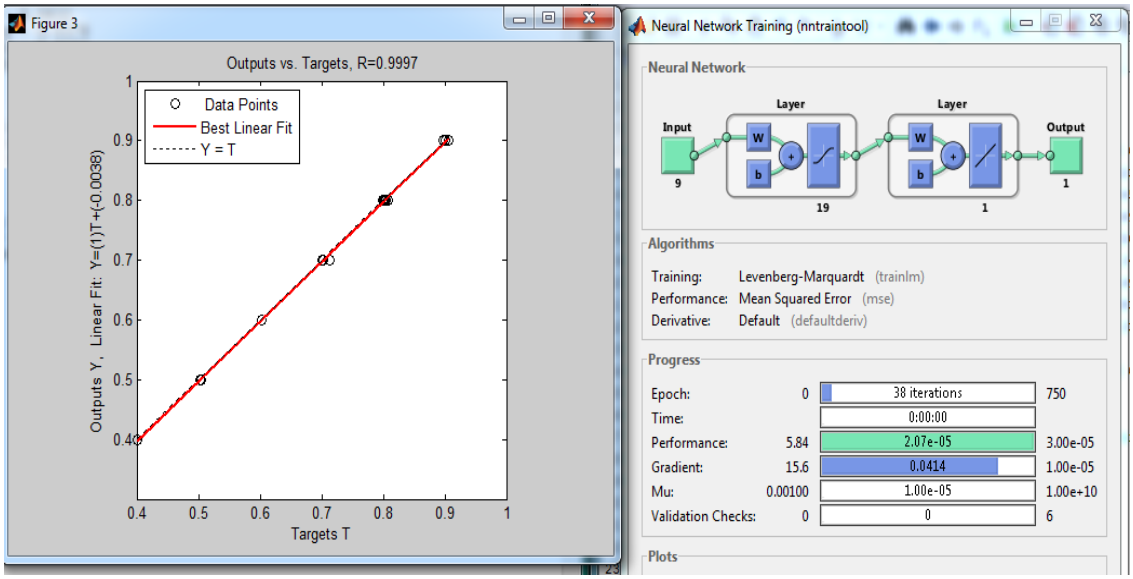
1) Here, the goal is kept 0.003, tansig is used as the sigmoid function and the number of neurons is kept as 19. The NN output is compared to a human judge score. The network training is stopped as the performance is reached. Here, the network performed better compared to the untrained data.



Human Judge Score	NN output	Difference	NN neurons used	NN sigmoid function	goal	epoch	Average difference of errors
0.7	0.740	0.040	19	tansig	0.003	750	0.0256
0.7	0.703	0.003					
0.9	0.900	0.000					
0.8	0.783	0.017					
0.7	0.787	0.013					
0.8	0.779	0.020					
0.6	0.526	0.074					
0.4	0.461	0.061					
0.5	0.497	0.003					

Table 13: Result of trained data for goal for goal 0.003

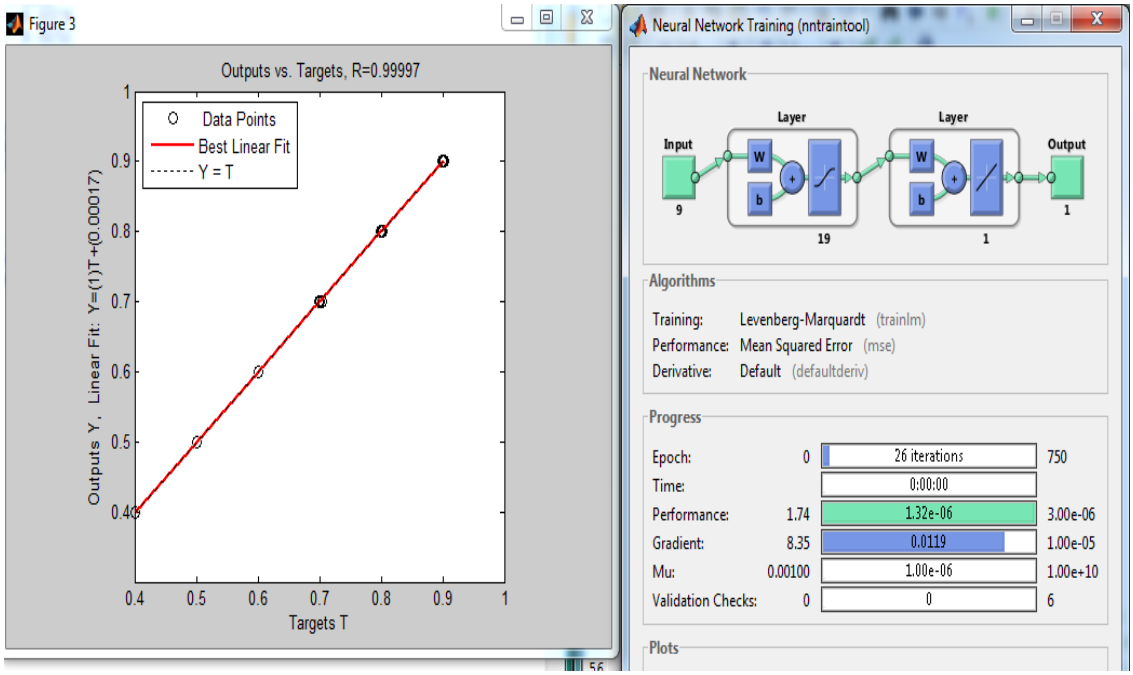
2) Here, the goal is increased 0.00003 to try to decrease an average difference of errors, tansig is used as the sigmoid function and the number of neurons is kept as 19. We can see that the average difference in errors between the human judge and NN output is decreased.



Human Judge Score	NN output	Difference	NN neurons used	NN sigmoid function	goal	epoch	Average difference of errors
0.7	0.700	0.0	19	tansig	0.00003	750	0.00144
0.7	0.701	0.001					
0.9	0.897	0.003					
0.8	0.802	0.002					
0.7	0.700	0.0					
0.8	0.801	0.001					
0.6	0.602	0.002					
0.4	0.401	0.001					
0.5	0.503	0.003					

Table 14: Result of trained data for goal 0.00003

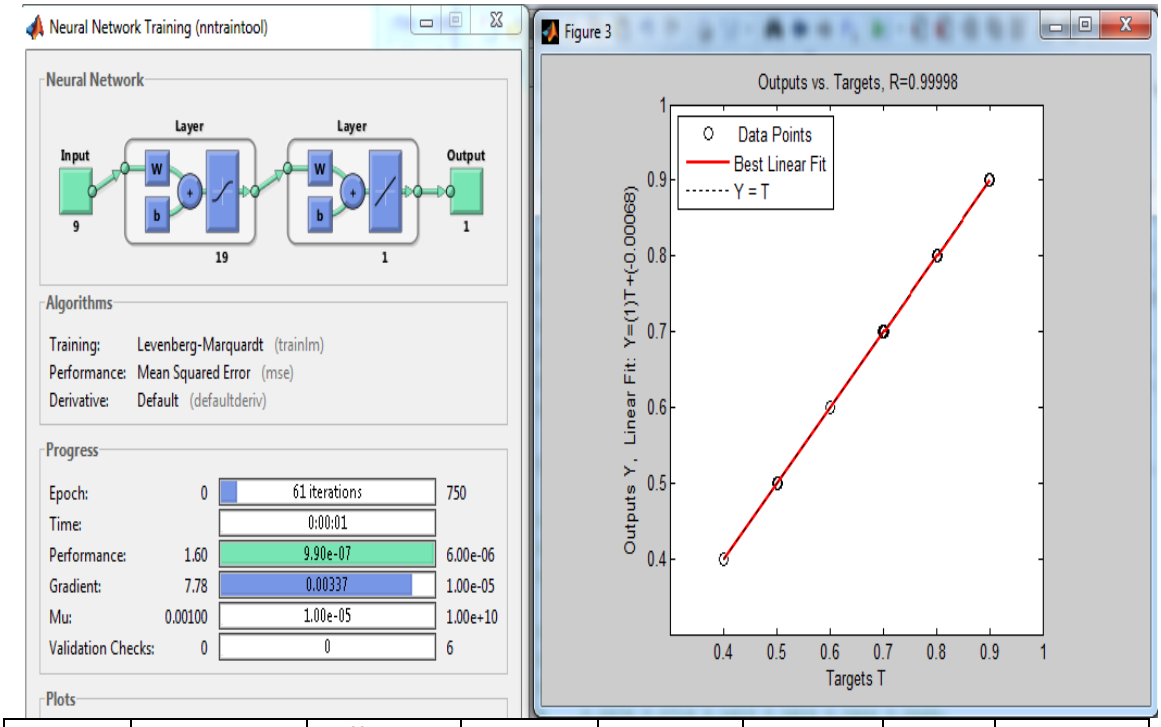
3) Here, the goal is decreased more than 0.000003 to improve the output of the neural network, tansig is used a as sigmoid function and the number of neurons is kept as 19. Here, the average difference of errors reached is at its lowest. But compared to untrained data, the trained data error is less.



Human Judge Score	NN output	Difference	NN neurons used	NN sigmoid function used	goal	epoch	Average difference of errors
0.7	0.700	0.0	19	tansig	0.000003	750	0.00033
0.7	0.700	0.0					
0.9	0.898	0.002					
0.8	0.799	0.001					
0.7	0.700	0.0					
0.8	0.800	0.0					
0.6	0.600	0.0					
0.4	0.400	0.0					
0.5	0.500	0.0					

Table 15: Result of trained data for goal 0.000003

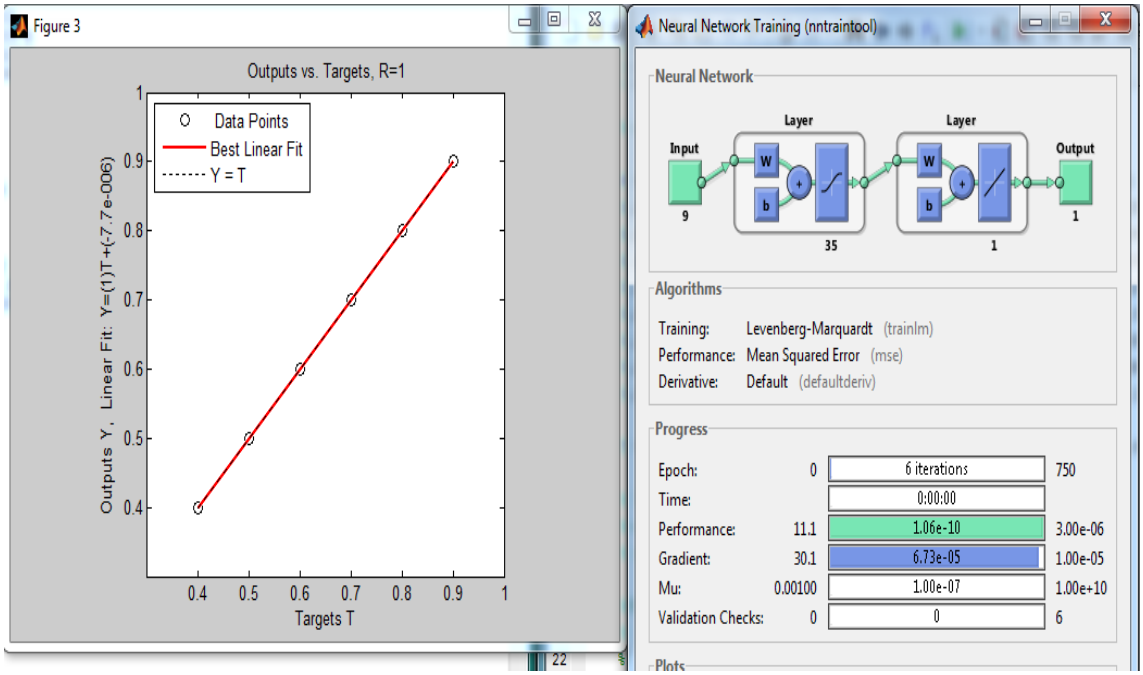
4) Here, we still further decreased the goal to 0.000006 , tansig is used as a sigmoid function and the number of neurons is kept as 19. Here, we can see the average difference of errors is increased compared to 0.000003. So, here we stop here decreasing the goal.



Human Judge Score	NN output	Difference	NN neurons used	NN sigmoid function used	goal	epoch	Average difference of errors
0.7	0.699	0.001	19	tansig	0.000006	750	0.00077
0.7	0.701	0.001					
0.9	0.899	0.001					
0.8	0.801	0.001					
0.7	0.701	0.001					
0.8	0.800	0.0					
0.6	0.599	0.001					
0.4	0.399	0.001					
0.5	0.500	0.00					

Table 16: Result of trained data for goal 0.000006

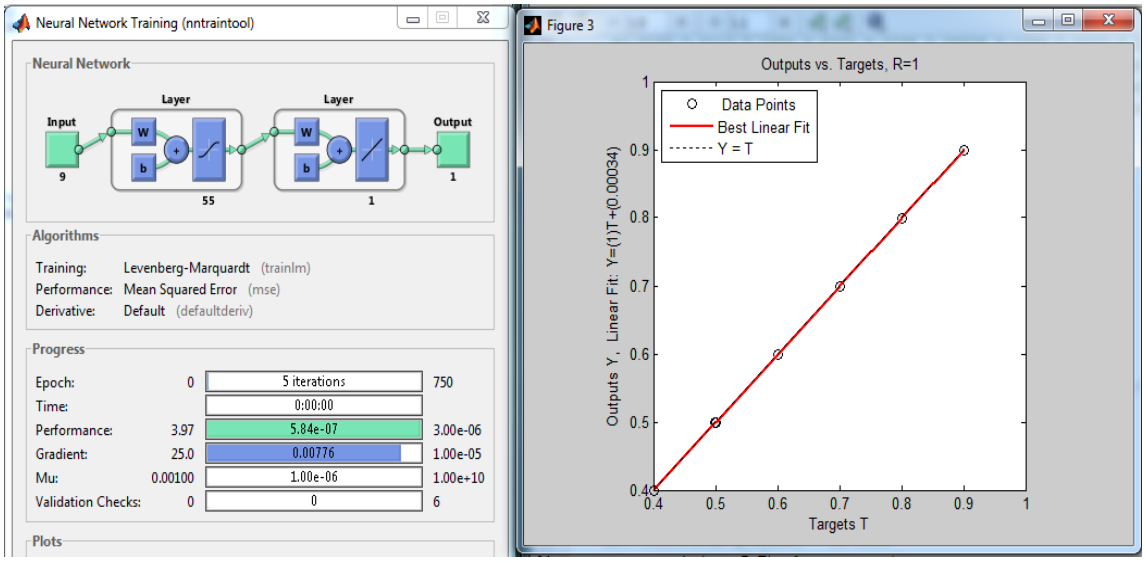
5) Here, we still further try to decrease the average difference by changing the goal and keeping the training function same. However, we can see the average difference in error has decreased and has factored at zero.



Human Judge Score	NN output	Difference	NN neurons used	NN sigmoid function used	goal	epoch	Average difference of errors
0.7	0.700	0.0	35	tansig	0.000003	750	0
0.7	0.700	0.0					
0.9	0.900	0.0					
0.8	0.800	0.0					
0.7	0.700	0.0					
0.8	0.800	0.0					
0.6	0.600	0.0					
0.4	0.400	0.0					
0.5	0.500	0.0					

Table 17: Result of Untrained data for 35 hidden layer neurons

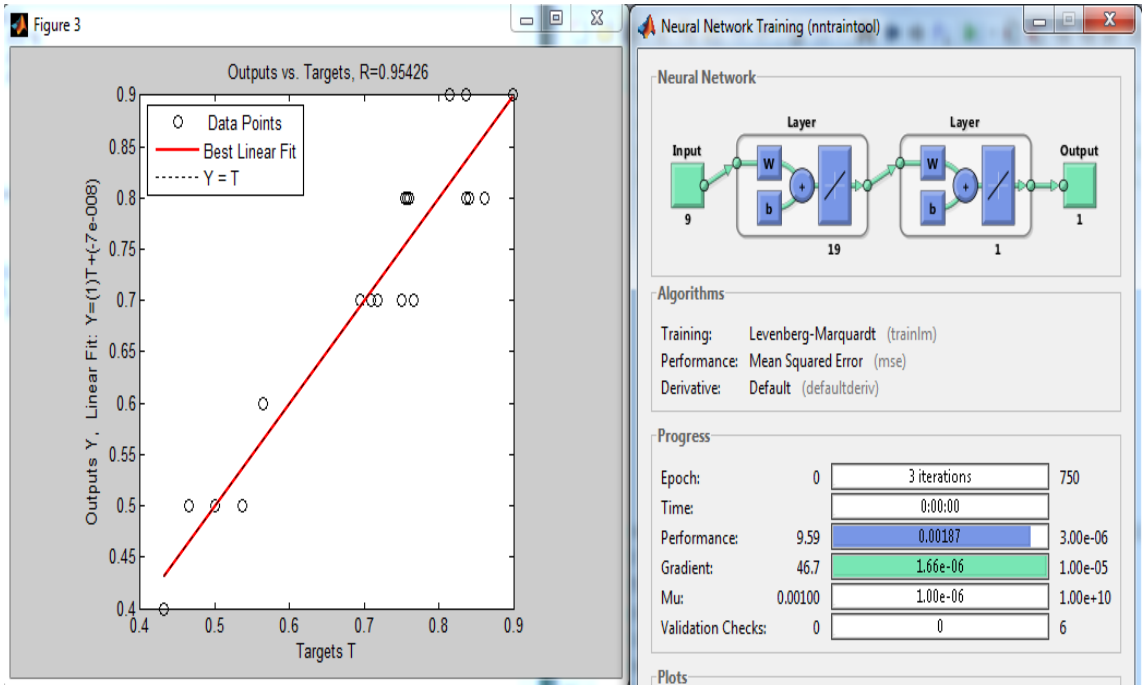
6) Here, we still try to decrease the average difference by changing the goal and keeping the training function same. However, we can see the average difference in errors is increased, so we stopped changing goal, here.



Human Judge Score	NN output	Difference	NN neurons used	NN sigmoid function used	goal	epoch	Average difference of errors
0.7	0.701	0.001	55	tansig	0.000003	750	0.00066
0.7	0.701	0.001					
0.9	0.900	0.0002					
0.8	0.801	0.001					
0.7	0.701	0.0013					
0.8	0.801	0.001					
0.6	0.600	0.0004					
0.4	0.400	0.0006					
0.5	0.499	0.001					

Table 18: Result of untrained data for 55 hidden layer neurons

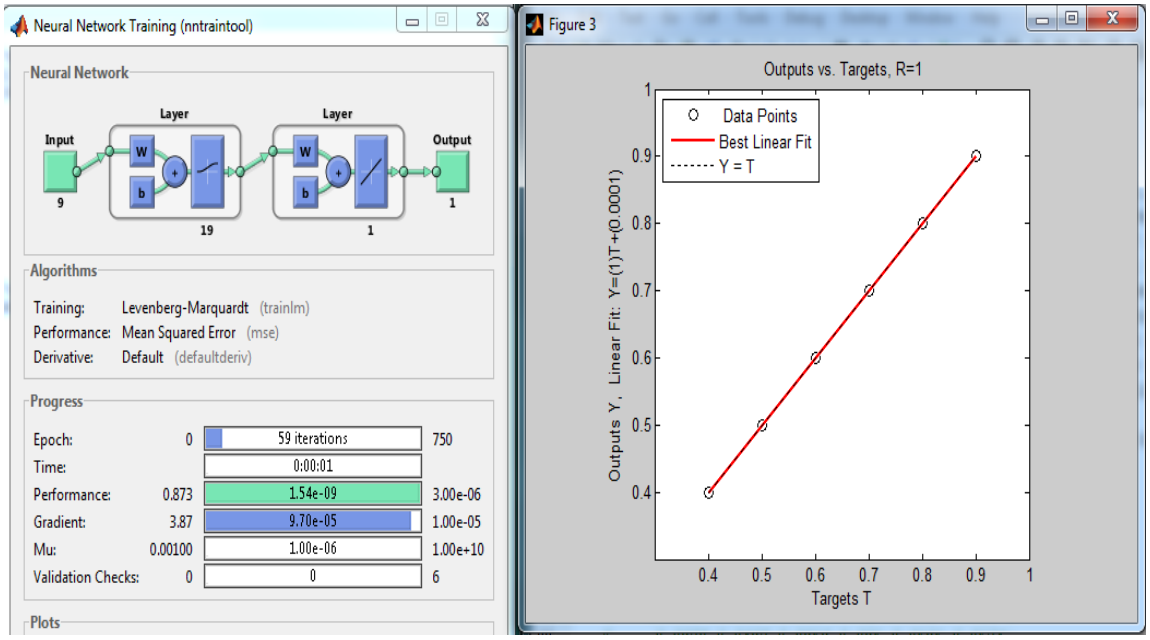
7) Here, the goal is kept 0.000003, the number of neurons is 19 and purelin is used as the sigmoid function. Here, the error increased compared to tansig function, but decreases compared to untrained data.



Human Judge Score	NN output	Difference	NN neurons used	NN sigmoid function used	goal	epoch	Average difference of errors
0.7	0.717	0.017	19	purelin	0.000003	750	0.02955
0.7	0.709	0.009					
0.9	0.898	0.002					
0.8	0.754	0.046					
0.7	0.750	0.050					
0.8	0.759	0.041					
0.6	0.565	0.035					
0.4	0.431	0.031					
0.5	0.465	0.035					

Table 19 : Result of Untrained data for purelin as sigmoid function

8) Here, the goal is kept 0.000003, the number of neurons is 19 and logsig is used as the sigmoid function. Here, the error increased compared to the tansig function, but it is low compared to the untrained data.



Human Judge Score	NN output	Difference	NN neurons used	NN sigmoid function used	goal	epoch	Average difference of errors
0.7	0.700	0.00	19	logsig	0.000003	750	0.00044
0.7	0.700	0.00					
0.9	0.899	0.001					
0.8	0.799	0.001					
0.7	0.700	0.0					
0.8	0.800	0.0					
0.6	0.599	0.001					
0.4	0.399	0.001					
0.5	0.500	0.0					

Table 20 : Result of untrained data for logsig as sigmoid function

Combined output of all tests

Scenario 1 : Data not used in the Neural Network training

Number of Neurons used	Sigmoid function used	Mean Square Error	Epoch	Average difference of errors
19	tansig	0.003	750	0.0832
19	tansig	0.00003	750	0.0376
19	tansig	0.000003	750	0.0252
19	tansig	0.000006	750	0.0928
35	tansig	0.000003	750	0.042
55	tansig	0.000003	750	0.056944
19	purelin	0.000003	750	0.074722
35	purelin	0.000003	759	0.087
55	purelin	0.000003	750	0.0901
19	logsig	0.000003	750	0.056833
35	logsig	0.000003	750	0.0733
55	logsig	0.000003	750	0.039

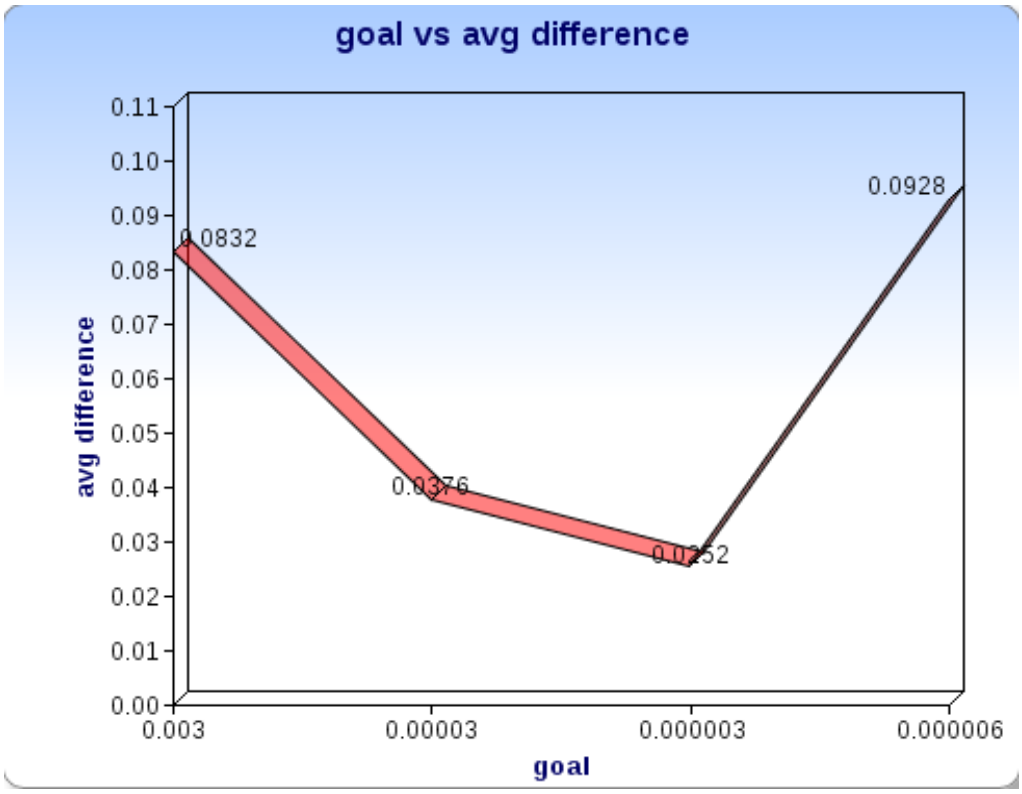
Scenario 2 : Data of the user's motion profile that was used in training the Network

Number of Neurons used	Sigmoid function used	Mean Square Error	Epoch	Average difference of errors
19	tansig	0.003	750	0.0256
19	tansig	0.00003	750	0.00144
19	tansig	0.000003	750	0.00033
19	tansig	0.000006	750	0.00077
35	tansig	0.000003	750	0
55	tansig	0.000006	750	0.00066
19	purelin	0.000003	750	0.02955
19	logsig	0.000003	750	0.00044
35	logsig	0.000003	750	0.00088
55	logsig	0.000003	750	0.001
35	purelin	0.000003	750	0.02733
55	purelin	0.000003	750	0.02733

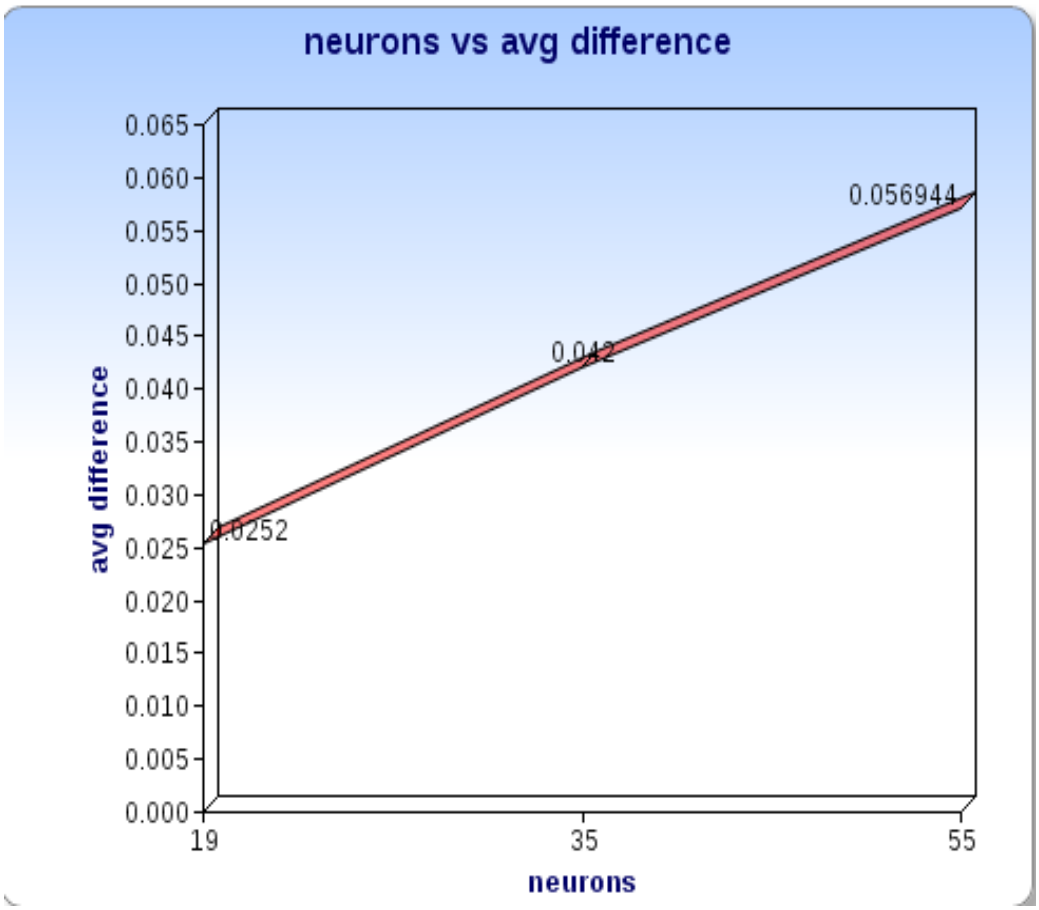
Charts

These charts represent the average differnce of errors versus different criteria that were used to stop network training after a certain expected performance is reached. The parameters used were: the Mean Square Error (goal), the gradient, the number of neurons and the sigmoid function.Here, the charts generated are from data not used in training the Neural Network.

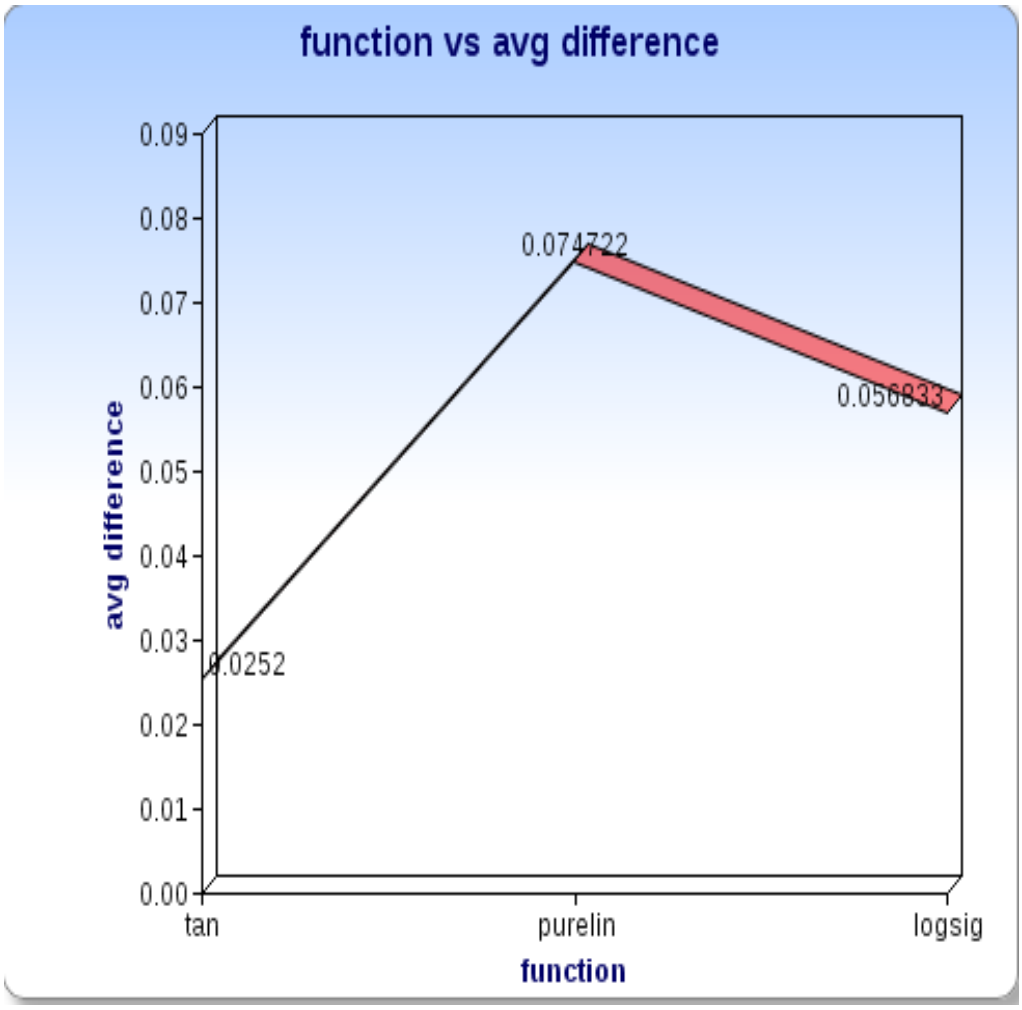
1) This chart is of an average difference of errors versus mean square error. This graph suggests that we achieved the best performance when the goal is 0.000003. Here, the average difference of error increases after goal of 0.000003 is reached. Here, the average difference is high when goal is 0.003, but decreased when we reduced the mean square error to 0.000003 and after that it starts increase.



2) This graph is of the number of neurons used in training the Network versus the average difference of errors. The graph suggests that as the number of neurons is increased from 19, the average difference of errors will increase. It indicates that the best performance happens when the number of hidden neurons is 19.



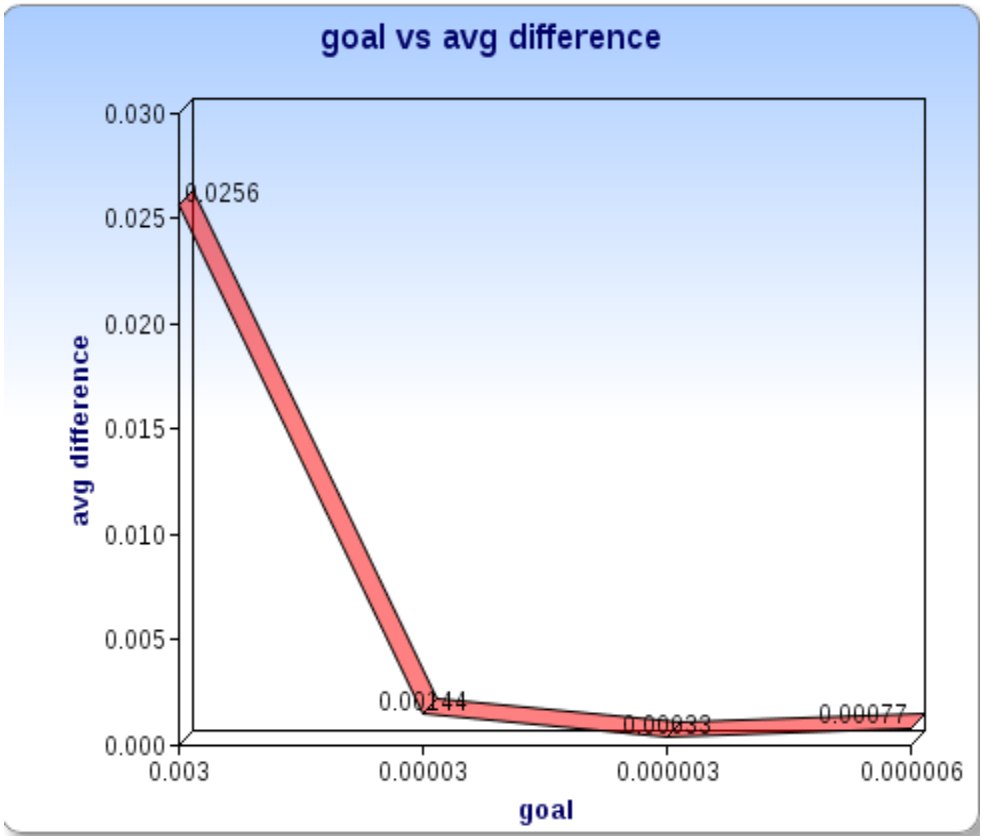
3) Here, the graph is of the function used to trained the network versus the average difference of errors in all the readings. The graph suggests that the tansig function is best to use as the average error is the lowest as compared to the purelin function and the logsig function.



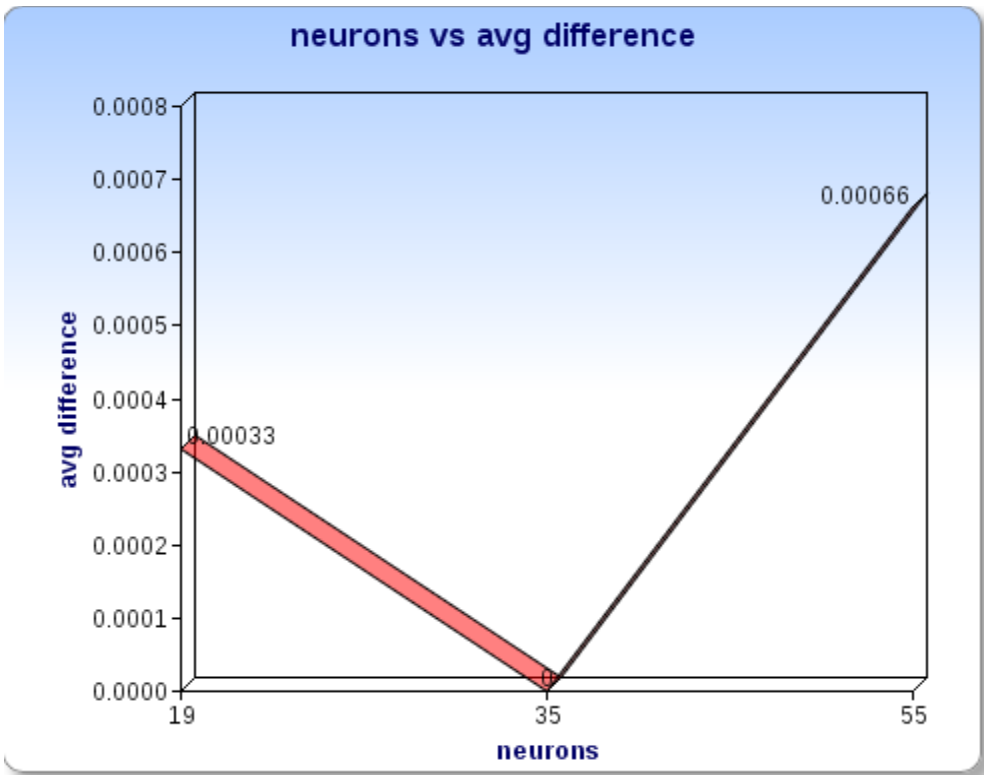
Trained Data

Here the neural network is tested with the data used in training.

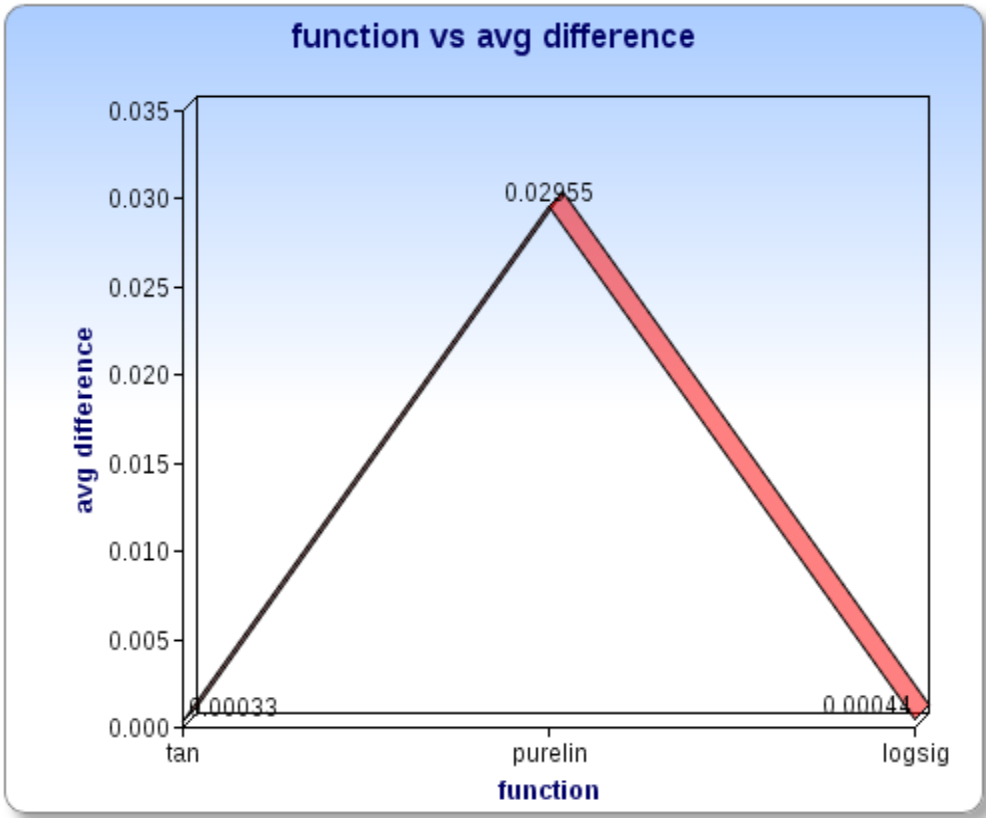
1) The chart below is of the goal versus the average difference of errors. It shows that when the goal is 0.000003, the neural network is at its best.



2) This chart is the number of neurons versus the average difference errors. Here, the error decreases as we increase the number of neurons beyond 35. So the best number of hidden neurons to be used is 35.



3) This graph is of the function used to trained the network versus the average difference errors in of all readings of data used in training the network. The graph suggests that the tansig function is best the to use since the error is low.



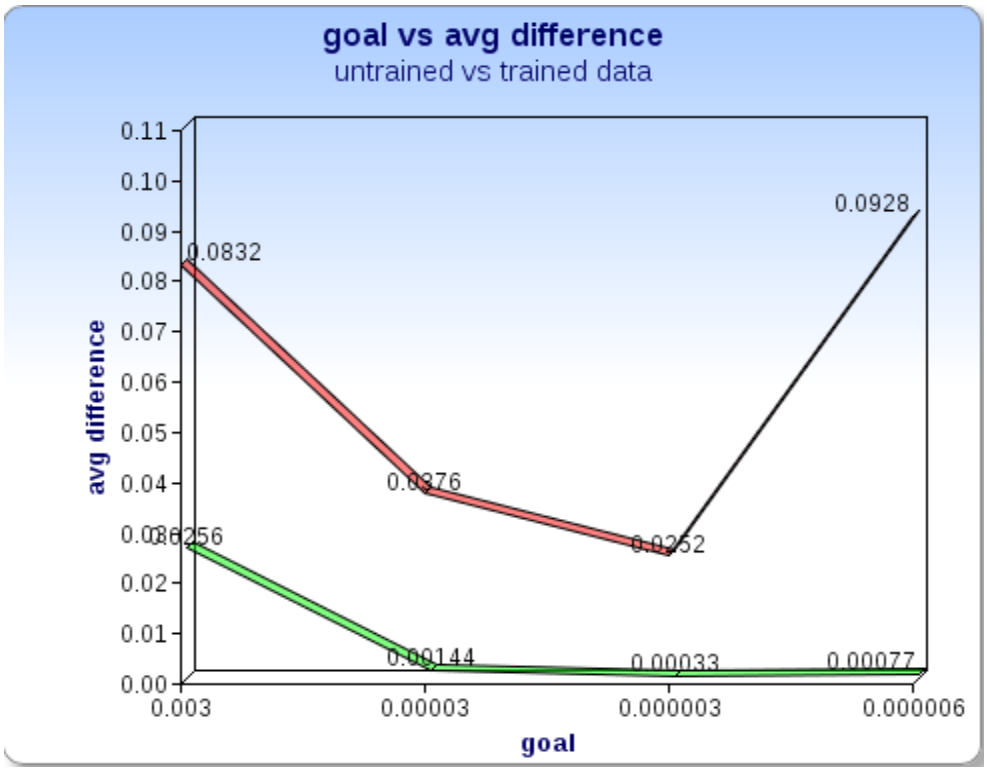
Comparing trained and untrained data

The charts below compare the Neural Network performance of trained data versus untrained data.

1) The chart represents the mean square error considered during training the Network versus the average difference of errors obtained while testing this Network with trained and untrained data. The Neural Network performed better in the case of data that is used in training the network.

Red line - Untrained Data

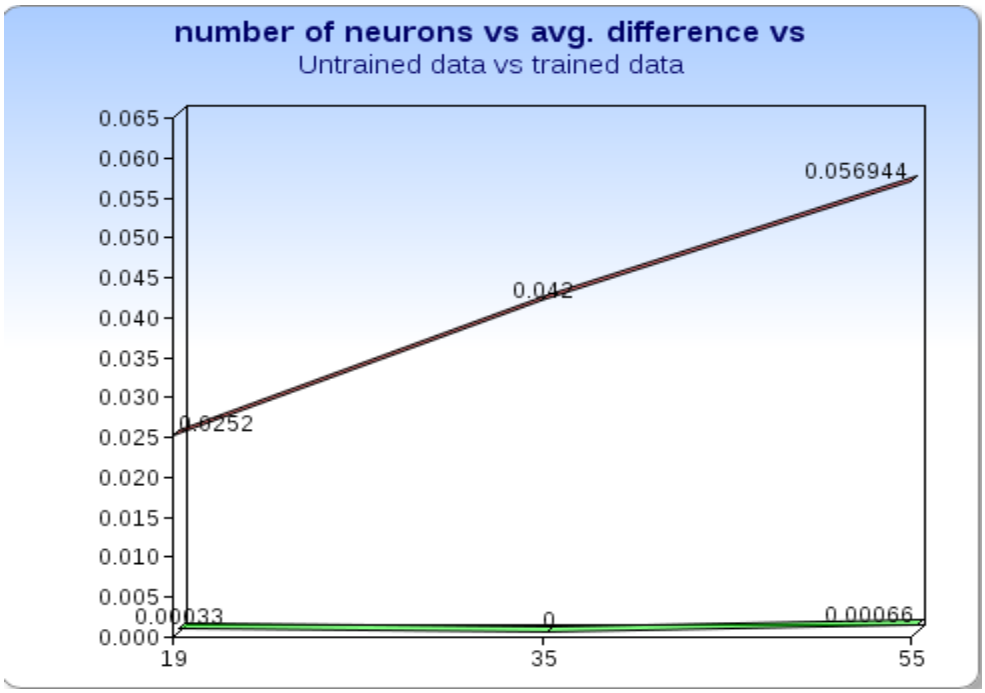
Green Line - Trained Data



2) The chart below represents the number of neurons considered during training the Network against the average difference of errors obtained while testing this trained Network with trained and untrained data. Here, we can conclude that the average difference of errors was low with trained data compared to that of untrained data.

Red line - Untrained Data

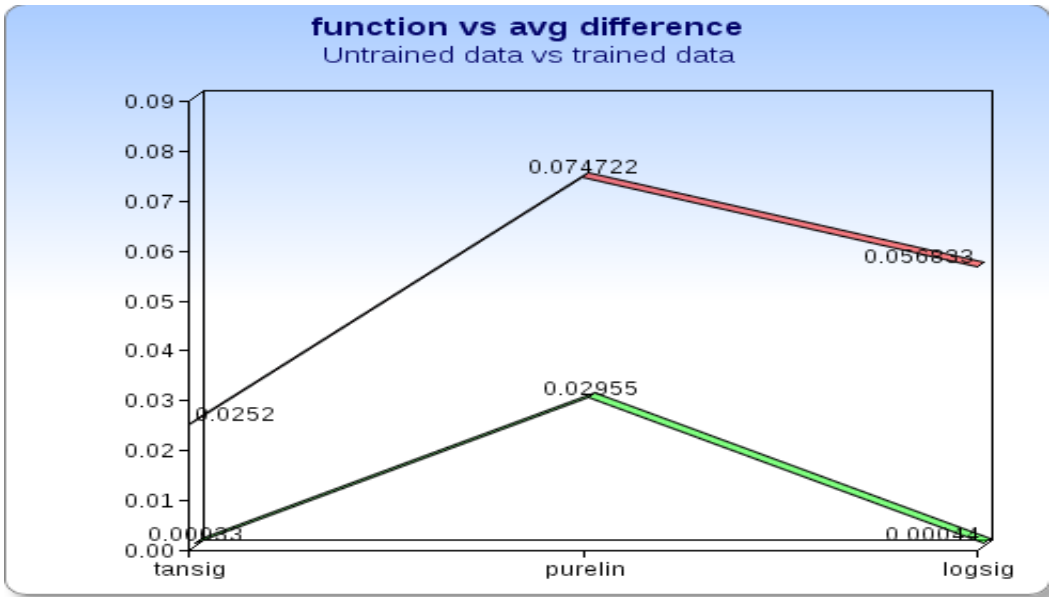
Green Line - Trained Data



3) The chart below represents the various sigmoid functions used during training the Neural Network versus the average difference of errors obtained while testing this Neural Network with trained and untrained data. Here, we can conclude that the average difference of errors was low with trained data compared to that of untrained data.

Red line - Untrained Data

Green Line - Trained Data



Technical Analysis

The Neural Network's best output is achieved when the sigmoid function is set to tansig. This is because tansig has a range from -1 to 1, while purelin has a range from negative infinity to positive infinity. Our input to the Neural Network's hidden layer neuron is in a range from [-1,1]. Similarly, tansig gives better results compared to logsig, since logsig has natural finite bounds of [0,1] and the logsig sigmoid function has slower rate of learning compared to the tansig sigmoid function. The table shows this as logsig takes 59 iterations to reach performance goal of 0.000003, while tansig takes only 26 iterations to reach the same performance goal. Additionally, when the number of neurons is set to 19 as one of the stopping parameters, the average difference of errors in the user's motion profile is minimal. Also, the output of the application is close to human judgment on the user's motion profiles when Neural Network is tested with data used in training the Network compared to untrained data.

Limitation

There are certain limitations in my proposed method. The neural network does not behave as needed if all the joints are taken into consideration. Also, this method is used if speed is not the measure to consider in giving a score.

Future Work

There is room for additional research on an algorithm better than Dynamic Time Warping for comparing two motion files. More joints can be taken into account for a better comparison of motion profiles. Moreover, an algorithm could be developed which tells the user the amount of degrees to bend a joint for better motion. Human motion sensors could be placed on a user. This would allow for more accurate data and joints that could be considered in measuring the closeness of user's motion profile.

Conclusion

In this project, I have presented a method to judge human motion. It allows the users to see the score while they are performing motion. User data performing motion was collected using Kinect sensors which contained X, Y and Z rotational axis for each joint. Only rotational axis were taken into account as different profile have different height due to which considering positional axis points for joints can generate inaccurate results. Master data and user data was used as input to the Dynamic Time Warping algorithm. It compensated for the speed and time between master and user data and gave a value suggested the similarity of both motion profiles. The output of Dynamic Time Warping algorithm is single value suggesting how close are both the motion profile. The value as close to zero suggest that they are similar. The output of this algorithm was fed to Neural Network that was trained with human judgment expert's data on motion profiles. For training the Network FeedForward Backpropagation algorithm was used. The Network gave the best output when tansig was used as the sigmoid function with the number of hidden layer neurons was 19. This trained network eventually behaved like an expert by commenting on how good or bad a user's motion profile exhibited.

References

[1] P. Latha, L. Ganesan, N. Ramaraj, and P. V. Hari Venkatesh, "Detection of moving images using the neural network."

[2] Christos Stergiou and Dimitrios Siganos, "Neural networks."

[3] Felty Timothy, Dynamic Time Warping retrieved from "http://www.mathworks.com/matlabcentral /fileexchange/6516-dynamic-time-warping"

[4] ByungRae Cha, Binod Vaidya, Young Kim, "Improvement of Neural Network Learning for Gesture Recognition of Game Contents using BPN, BPNX, and LM Algorithms."

[5] Roska, T; Boros, T; Thiran,P. ; Chua,L.O., "Detecting simple motion using cellular neural network"

[6] Christopher M.Bishop, Neural Networks for Pattern Recognition.

[7] Laurene V.Fausett, Fundamentals of Neural Network.

[8] Brian D. Ripley, Pattern recognition and neural networks.

[9] Neural Network retrieved from "http://en.wikipedia.org/wiki/Neural_network"

[10] Backpropagation retrieved from "http://en.wikipedia.org/wiki/Backpropagation"

[11] Dynamic Time Warping algorithm retrieved from "http://en.wikipedia.org/wiki/Dynamic time_warping"

[12] M.Beale, M.Hagan, H.Demuth, "neural Network Toolbox User's Guide"

[13] P.Senin, "Dynamic Time Warping Algorithm Review"

[14] Motion Karaoke retrieved from "https://sites.google.com/site/motionkaraoke/"

[15] Backpropagation retrieved from "http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html"

[16] BVH retrieved from "http://www.mindfiresolutions.com/BVH-biovision-hierarchy.htm"

[17] MSE retrieved from "http://en.wikipedia.org/wiki/Mean_squared_error"

[18] Sigmoid function from "http://en.wikipedia.org/wiki/Sigmoid_function"

[19] Neural Network Toolbox retrieved from "http://radio.feld.cvut.cz/matlab/toolbox/mmet/newff.html"