Applying Genetic Algorithm in Architecture and Neural Network Training

Mahshid Kaviani[1] Seyed Majid MirRokni[2*]
[1]MSc student, Yazd University
[2]Space Physics Group, Department of Physics,Faculty of Science, yazd University

Abstract
The structure of Artificial Neural Network with ANN symbol is not a predefined program. Therefore, architecture and proper training of ANN have significant effect on the results. ANN architecture of consists determination of a number of optimal neurons in hidden layers creating a challenging puzzle that makes us use the trial and error method. ANN training is an optimization process to determine the optimal values of weights and biases. The classic optimization algorithms provides us with the optimal local values, wherein calculation time is increased in multivariable problems and the rate of convergence is reduced to optimal value. Applying Genetic Algorithm with symbol GA offers an appropriate solution for these challenges. GA is optimized through random search technique. It simultaneously considers various responses and reduces chance of convergence to a local optimum. Therefore, by an intelligent search method values close to the optimum point are found. The objective of using convergence in this case study is to predict the daily average temperature in 2009 which is obtained through some parameters such as pressure, vapor pressure and relative humidity. GA is used in this study, to avoid the trial and error methods in the architecture of ANN and to achieve absolute optimum values in a short time, for determining the number of hidden layer neurons as well as the optimal values of weights and biases in ANN training. The results show that the GA approach can be replaced with trial and error methods in determining optimal state of ANN architecture, and to increase speed, accuracy and efficiency of the training.

Key Words:
Artificial Neural Network, Genetic Algorithm, optimization algorithms, trial and error, ANN architecture.

1. Introduction

Artificial Neural Network with ANN symbol is a powerful tool for approximation of functions; this characteristic relates the target and input data. every ANN consists of one or more neurons that receive input data of $x1, x2, ..., xp$ with weights and biases $w1, w2, ..., wp$ and taking into account the biase (b) a stimulus function of ($\int$) is exerted on the results in exchange for every neuron summing them together. The result of this operation is the output neuron (O) (Fig 1.1) [14].

If ANN is created with one neuron, it may not have the ability to approximate nonlinear functions. To overcome this limitation, multilayer ANN is used being composed of several layers containing one or more neurons.
In the input layer, the data are entered into each neuron having initial weights and biases; in addition, after performing the operations, the output of every Perceptron is considered as an input for each neuron of the next layer, so that it may finally reach the last layer. The number of neurons should equal to the number of outputs of the problem ater intermediate layers (hidden), in the last layer or the output layer, (Fig1.2) [15, 2, 18]. We can use ANN approach to approximate function; ANN in here first determines the input data and the ANN target. Then it creates ANN of weights and biases for producing output data (O); the error is determined based on the difference between the target data and the output data. During the training process, these weights and biases are optimized so that the error rate is reduced until it reaches an acceptable value.
Therefore, ANN acts as a function by specifying the optimal values of weights and biases. Thus, if it receives some input data, it will produce an output proportional to the target [5, 19]. On the other hand, the ANN structure is not a pre-determined program, so the architecture of ANN and its training should be considered when the appropriate input is selected for the target. Architecture of ANN structure consists of the creation of different parts of raw network such as the type of stimulus function, the determination of the optimal number of hidden layers and the optimal number of neurons in each layer. In the training phase, the optimal values of weights and biases being unknown parameters of approximation function problem, are determined through an optimization approach. Therefore, in designing ANN, different optimization problems are posed. Solving these optimization problems is important for special applications since speed, accuracy and efficiency of ANN depend on the complexity of the problem and ANN's structure [4].
Architecture of ANN significantly affects directly the two important factors of publicity and training time [1]. Correct selection of ANN architecture increases responding time and thus; improves system performance

Manuscript received June 5, 2017
Manuscript revised June 20, 2017

across the system. Generally, the more complicated is the considered system, the larger the ANN's size should be (number of hidden layers and neurons in each layer will be larger). If the size of ANN is considered too small for the intended application, it will never be able to approximate the desired function. in this case, great errors are produced. On the other hand, when the size of ANN is large, the learning process for the training data can well be performed, however, in case of the new data that have not been trained, this size does not provide appropriate output and consequently, ANN is not publicized and over fitting occurs [3]. Also, the larger gets the size of the ANN, the more time is spent on computing. Usually, understanding and extracting information from a smaller model are easier and need less time. Thus, the size of ANN should be as small as possible, but large enough to provide results with high accuracy and reliability [4, 1, 16, 10, 3].

Finding the optimum architecture for ANN is a complex process being considered as an art rather than a science. Usually, the optimal number of hidden layers and neurons in each layer of ANN are determined by trial and error method. It should be mentioned that a lot of time needs to be spent so that, all cases can be examined by trial and error method; resorting to this method makes the problem solving more difficult.

In ANN training mainly classical methods are used. Classic algorithms are usually more effective for solving small problems with limited number of variables. However, when faced with large number of variables, the classical methods will not work efficiently. Applying classical training algorithm one should regard that if the number of neurons is high, the computation time for training will be increased and the rate of convergence to the optimal value will drop significantly [9, 1].

Therefore, finding the correct architecture of the ANN proportional to the problem on one hand, and finding the values of the weights and biases in the training process for problem solving on the other hand are problems that arise in designing ANN.

Applying evolutionary optimization, genetic algorithm (GA) can replace trial and error methods to determine optimal ANN architecture, and increase the speed, accuracy and efficiency of the training phase [13, 8, 6]. GA approach is evolved from the living organisms. This approach is used for multi variable optimization and has the ability to solve extensive range of optimization problems in discrete and continuous states through the use of just three simple genetic operations (Selection, Cross Over and Mutation). Therefore, it is useful for solving optimization problems associated with chaotic complicated systems. GA is optimized by random search technique, so it regards several points simultaneously in research space and reduces the chance of convergence to a local optimum [7]. Therefore, by an intelligent search method, it can find values close to the absolute optimum. Hence, by combining the GA and ANN, we can achieve optimal amounts in ANN architecture in less time. The advantage of this approach - beside the less time of execution- is that by one time execution of algorithm, all optimal points can be achieved and designer's job in making decisions is facilitated.

Penfold et al (1990) [9] used two methods of Back Propagation and GA for training ANN. The results of that research showed that the combination of ANN and GA may reduce errors, speed up calculations and provide a better model. Zhang and Muhlenbein (1993) [4] applied GA and optimized the structure, training weights and biases at the same time. Blanco et al (2000) [1] used GA for selecting the appropriate architecture and finding the minimum number of required neurons in the ANN. Arifovic and Gencay (2001) [11] used GA to select the input, determine the number of hidden layers as well as the relational structures between input and output layers which finally resulted in the determination of ANN training. Ileana et al (2004) [10] used GA approach for optimization of ANN dimensions (number of hidden layers and number of neurons in each layer).

Therefore, in designing ANN by GA two general approaches can be followed. In the first approach, GA is used only in the architecture of ANN structure where training can be conducted by classical methods. In the second approach, GA is used both for architecture and training ANN [16].

Predicting the daily average temperature, the case study of ANN targets in the present study included Yazd synoptic station (selected stations in 2009) being considered by the Meteorological Parameters. Therefore, the second approach has used GA to determine the optimal number of neurons and to determine the optimal values of the weights and biases during the training phase. In data and methods section, application of GA is investigated for this purpose. Then details on the methods of using this approach in the present study are discussed. Finally, the results of study are presented.

2.Data and Methods

To predict the average daily temperature in 2009, data of synoptic station with average daily pressure, water vapor pressure, and relative humidity were used within 29 years (1980 to 2008) in Yazd (selected stations). Input and target data have been normalized as follows:

$$X_N = a \times \frac{X_i - X_{\min}}{X_{\max} - X_{\min}} - b$$

In this relation X_i is the raw variable, $X_{\min}$ is the minimum and $X_{\max}$ is the maximum values of the sample

being studies. X_N shows normalized value of Xi; a and b depending on the type of problem are determined in a way that limit raw data to reach the considered ranges. In this study values of 0.8 and 1, have been chosen for variables a and b, respectively, so that the data can be placed in the interval (0.1, 0.9).

After normalization, the selected data should be divided into training and testing data. The training data clearly explains the different aspects of the system for ANN. Test data should be used to ensure accuracy of ANN and improve its generalizability for new data, [12, 14].

After preparing the data, the ANN must be designed. In this study GA has been used for determining the optimal number of neurons in the hidden layers as well as determining the optimal values of the weights and biases in ANN training process. It can be said that GA is a programming technique in MATLAB software that uses the genetic evolution of living organisms to solve optimization problems. For this purpose, the evolution of living beings, has been simulated through computational blocks. In this approach, each chromosome represents a point in the search space, being a possible answer to the mentioned problem. These chromosomes (responses) are formed from fixed number of genes (variables), having discrete or continuous values. For representing chromosomes, various coding is used. Decoding them, we can obtain the answer. Sets of chromosomes make up a population. With the application of selected genetic operators, crossovers and mutations on any population, a new population is formed by the same number of chromosomes creating the next generation.

In order to solve the problem using GA, the fitness function must be defined for that problem. For each chromosome, this function represents a number that represents the competence or ability of that chromosome. In the last generations, the more appropriate chromosome is the answer of optimization problem and due to GA features, we can hope that it is close to Absolute optimal. In the present study, ANN approach is combined with GA as follows:

1- Number of variables (neurons, weights and biases) and their variation domains, population in each generation, the maximum number of generations, the number of parents in crossovers and mutations, and fitness function are determined. Then, each chromosome is defined in a way that one part includes variables (genes) with discrete values (representing the number of neurons in each hidden layer) and the other part includes variables with continuous values (representing the values of the weights and biases). At this stage, one random initial population is created with this specification.
2- One raw ANN is created and the number of hidden layers as well as the stimulus function are determined in it.
3- ANN architecture and training weights and biases are specified using values of the genes in the determined population. ANN is trained through the normalized input data, dimensions, weights and biases that GA has proposed for that; fitness function for each chromosome in this population is calculated.
4- To create the next generation, genetic operators such as selection, crossover and mutation are used.
5- Generation of previous population has combined genetic operators; fitness functions of all chromosomes are calculated.
6- After sorting the chromosomes based on the best values of fitness function, and by elitism of the population in each generation, the best chromosomes in each generation are separated and a new generation is created. In this stage, a value of one is added to the number of generations and the above steps are repeated until the number of generations reaches its maximum value. In the present study, the maximum number of generations have stopped the condition. Fig 2.1 shows these operations.

After implementation of GA and ANN training, to predict the temperature the statistical indicators, MSE, RMSE and fitness diagrams, correlation and error coefficient, have been used for evaluating the performance of ANN.

3. Discussion

In the present study, data of average relative humidity, average daily pressure and temperature and average daily water vapor's pressure for input data as well as average daily temperature for target data were collected in a 29-year statistical period (1980 to 2009) in the MATLAB software environment. After normalization, data are divided into two parts of training and testing. In this study 70% of the data was randomly assigned for ANN training and 30% for the test.

After defining data for ANN, the architecture stage and training begins; in this stage GA is used. According to the steps in Figure 2.1, parameters of problem must be defined for the GA.

Table 3.1 shows GA parameters such as the number of neurons, weights and biases (Nvar-Nerons and Nvar-Ws & Bs) and their variation domains (VarMin-Nerons, VarMax-Nerons, VarMin-W & B and VarMax-W & B), the number of population in each generation (PopSize), the maximum number of generations (MaxGeneration), number of parents in the intersection operations (Nc) and

mutation (Nm) and fitness function (Cost-Fcn). The present study considered a hidden layer for ANN, so the number of related variables neurons (Nvar-Nerons) equals one; however, due to the relationship between the number of neurons, weights and biases, it is realized that the number of variables related to weights and biases (Nvar-Ws & Bs) in each chromosome is different. Thus, Nvar-Ws & Bs are obtained while creating the initial population for each chromosome through the relation written in the table. Quantities for the ANN with a hidden layer are obtained from the following relations; the quantities are: N (IW) (number of weights in the input layer), N (LW) (number of weights in the hidden layer), N(B1) (Number of biases for the hidden layer) and N(B2) (Number of biases for output layer):

$$N(IW) = Num(Neurons) \times Num(Inputs)$$
$$N(LW) = Num(Neurons)$$
$$N(B1) = Num(Neurons)$$
$$N(B2) = Num(Outputs)$$

In this relationship Num (Neurons) is the number of neurons, Num (Inputs) indicates the number of inputs, and Num (Outputs) shows the number of outputs. PopSize parameter specifies proposed chromosome number (initial number of population). To produce the next generations, some of the GA chromosomes of initial population are randomly selected to Cross Over Operation and mutations. Parameters Nc and Nm show the number of parents for crossover and mutation operations of the initial population, respectively. Being stop condition of GA in this research, MaxGeneration parameter indicates the maximum number of generations . In many problems, criterion for evaluating the performance of ANN is average square error with symbol of MSE. Therefore, in this study, the MSE index has been considered for fitness function GA (Cost-Fcn) .

In the present study, GA has been used simultaneously for determining the optimal number of neurons, optimal values of weights and biases, so in the initial population, each chromosome proposes number of neurons for raw ANN. Then, raw ANN is created by this number of neurons using library function of newff in the MATLAB software. In this hidden layer, the stimulus function of logsig has been considered for hidden layer neurons as well as the stimulus function of purelin for neurons in the output layer

At this stage, according to the relation mentioned in table 3.1, Nvar-Ws & Bs has been specified while GA can produce values for weights and biases in a definite range. At this stage, each proposed GA chromosome in the initial population has been introduced to the raw ANN and ANN has been trained based on the proposed number of neurons, weights and biases. Then, fitness function for each chromosome is calculated and the population is rearranged based on the lowest values of MSE. For the next generations, some of chromosomes have been selected with the number of Nc and Nm as parents to operate crossover and mutation.

After the crossover and mutation operations, new chromosomes come together with previous generations; fitness function is calculated for all of them and as the number of PopSize, best chromosomes are used with the lowest values of MSE to create a new generation .

Figure 3.1 shows the values of the fitness function of the best chromosome in each generation. In this figure, horizontal axis shows the number of generations and vertical axis indicates the minimum values of MSE for ANN in each generation. According to the condition of the algorithm's end, the best chromosome in last generation is the answer to the problem. Therefore, the number of neurons, values of weights and biases are introduced by this chromosome to the ANN . ANN for the test data acts like a function after the optimum values for the number of neurons the weights and biases of training data are determined, , .

By Post processing methods, the performance of ANN can be evaluated during the training phase and after that. To evaluate the performance of ANN Table 3.2 shows the values of MSE, RMSE for training data, and test data and all output data of ANN.

4. Conclusion

ANN structure is not a pre-determined program and optimal values of weights and biases are determined in the training process being based on the input data. Therefore, by choosing the appropriate input data and correct architecture, ANN finds a better training and can have a better performance in predicting new data; the more accurate information on system features is provided to ANN, clearer the effects of complexity of nonlinear relationships gets and ANN cangain the ability of approximation of the dominant function on data. It should be noted that ANN is not a miracle, but if used wisely can create striking results.

Designing ANN involves difficult issues of optimization. The problems that arise in designing ANN are finding the correct architecture of ANN and set of weights and biases to solve the problem. To determine the optimal ANN architecture a lot of time is usually spent on calibration through trial and error method;the classical methods in training ANN obtain locally optimal values of weights and biases. Therefore, in this study, ANN gas been designed to use GA to predict the temperature. The results of the study shows that GA can simultaneously regard possible answers . For this reason, it is possible to achieve optimal values in ANN architecture and train it in less time. The

advantage of this approach is that in addition to less time of implementation, only one execution of the optimization algorithm can achieve all optimal points ; consequently, the job of designer in decision making gets easier. Therefore, The GA approach can replace trial and error methods in determining optimal state of ANN architecture, and increase the speed, accuracy and efficiency in the training phase.

References

[1] A. Blanco, M. Delgado, M.C. Pegalajar, A genetic algorithm to obtain the optimal recurrent neural network, International Journal of Approximate Reasoning, 23(2000), 67-83.

[2] A. Kaur, H. Singh, Artificial Neural Networks in Forecasting Minimum Temperature. IJECT., 2(2011), 101-105.

[3] A. Kopel, X. H. Yu, Optimize Neural Network Controller Design Using Genetic Algorithm, Proceeding of 7thWorld Congress on Intelligent Control and Automation, June 25-27, Chongqing, China, 2008, 2012-2016.

[4] B. T. Zhang, H. Muhlenbein, Evolving Optimal Neural Networks Using Genetic Algorithms with Occam's Razor, Complex Systems, 7(1993), 199-220.

[5] C. M. Roadknight, and Co Author, Modeling Complex Environmental Data, IEEE Trans. Neural Networks, 8(1997), 852-861.

[6] D. E. Goldberg, Genetic Algrithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[7] E. D. Judith, Neural Network Architectures, V AN NOSTRAND REINHOLD, New York. 1990.

[8] G. F. Miller, P. M. Todd, S. U. Hegde, Designing Neural Networks using Genetic Algorithms, proceedings of Third International Conference on Genetic Algorithms, 1989, 379-384.

[9] H. B. Penfold, O. F. Diessel, and M. W. Bentink, A genetic breeding algorithm which exhibits self-organizing in neural networks, International Symposium on A.I. Applications and Neural Networks, 1990, 293-296.

[10] I. Ileana, C. Rotar, A. Incze, The optimization of feed forward Neural Networks structure using Genetic Algorithms, proceedings of International Conference on Theory and Applications of Mathematics and Informatic, ICTAMI., 2004, 223-234.

[11] J. Arifovic, R. Gencay, Using genetic algorithms to select architecture of feedforward artificial neural network, Physica A, 289 (2001), 574-594.

[12] J.M. Zurada, Introduction to Artificial Neural Network Systems, West Publishing Company, 1992.

[13] K. P., Sankar, Selection of Optima Set of Weights and biases in a Layered Network Using Genetic Algorithms, Information Science, 80(1994:), 213-234.

[14] L. Fausett, Fundamental of Neural Networks: Architectures, Algorithms and Applications. Prentice Hall, 1994.

[15] Paras, and Co Author, A Feature Based Neural Network Model for Weather Forecasting. World Academy of Science,Engineering and Technology, 10(2007), 66-73.

[16] S.Mizuta, and Co Author, Structure Design of Neural Networks Using Genetic Algorithm, Complex Systems, 13(2001), 161-175.

[17] S. Panigrahi, Y. Karali, H. S. Behera, Normalize Time Series and Forecast using Evalutionary Neural Network, International Journal of Engineering Research and Technology, 2(2013), 2518-2522.

[18] Shereef, I. Kadar., S. S. Baboo, A New Weather Forecasting Technique using Back Propagation Neural Network with Modified Levenberg-Marquardt Algorithm for Learning. IJCSI., 8(2011), 153-160.

[19] Yuval, Neural Network Training for Prediction of Climatological Time Series, Regularized by Minimization of the Generalized Cross-Validation Function, Monthly Weather Review, 128(2000), 1456-1473.

Tables

Table 3.1: GA Parameters

GA Parameters	Valuse
Nvar-Nerons	1
Nvar-Ws & Bs	N(IW)+N(LW)+N(B1)+N(B2)
VarMin-Nerons : VarMax-Nerons	1 : 5
VarMin-W & B : VarMax-W & B	-1 : 1
PopSize	140
MaxGeneration	100
Nc	112
Nm	7
Cost-Fcn	MSE

Table 3.2: values of MSE, RMSE for training data, and test data and all output data of ANN

Datas / Assessment Standard	Train Data	Test Data	All Data
MSE	0.002	0.002	0.002
RMSE	0.041	0.051	0.044
R	0.98	0.97	0.98

Figures

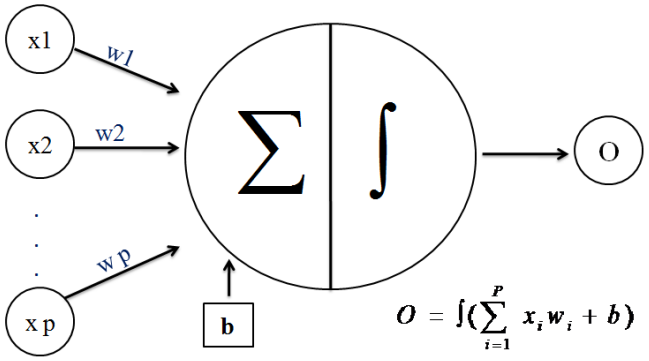

Figure 1.1: Structure of Neuron

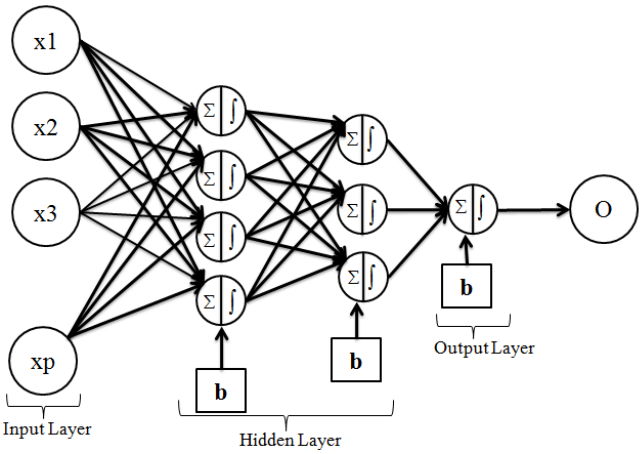

Figure 1.2: Structure of Multi-Layer Perceptron Networks

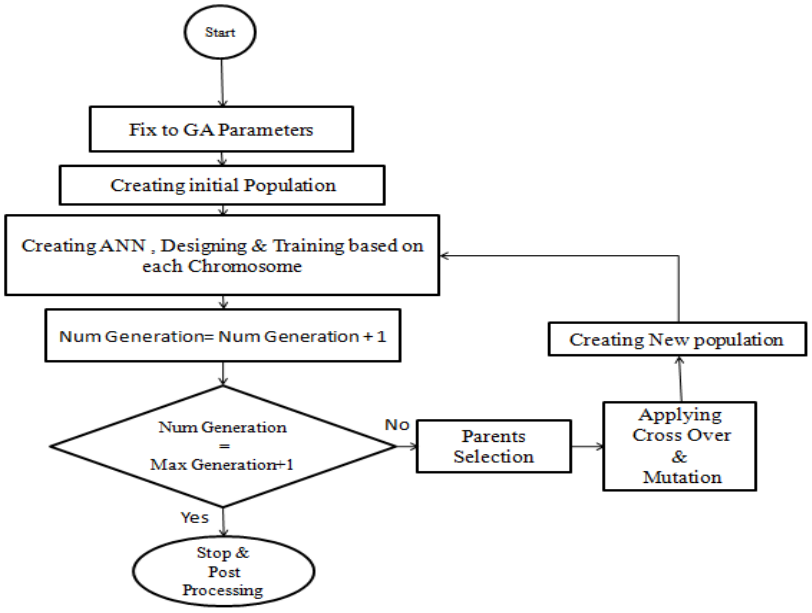

Figure 2.1: view of GA-ANN approach

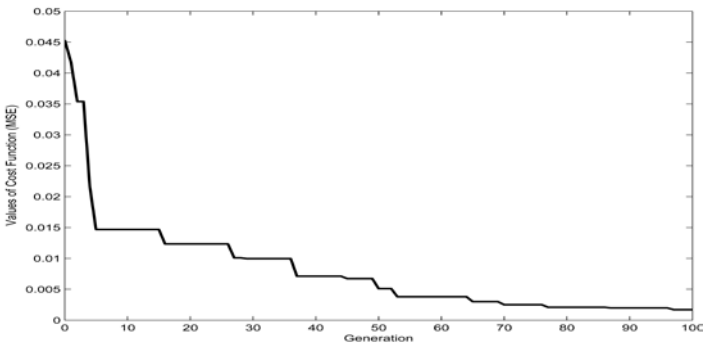

Figure 3.1: values of the fitness function of the best chromosome in each generation

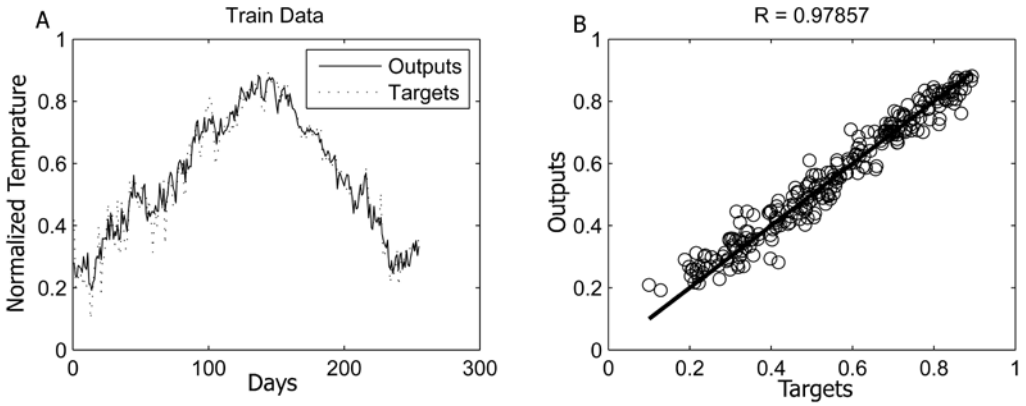

Figure 3.2: (A) Fitting the output data on the target data for the training data; b) Correlation coefficient between output and target values for the training data

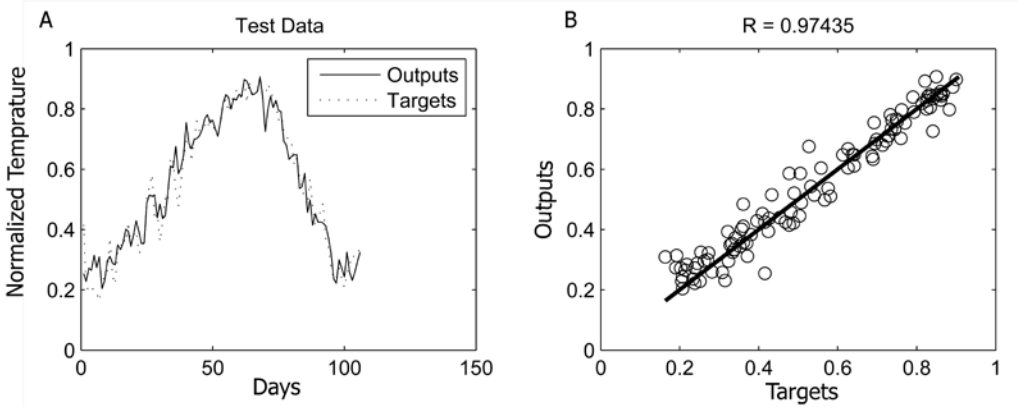

Figure 3.3: (similar to Figure 3.2 but allocated to the test data)

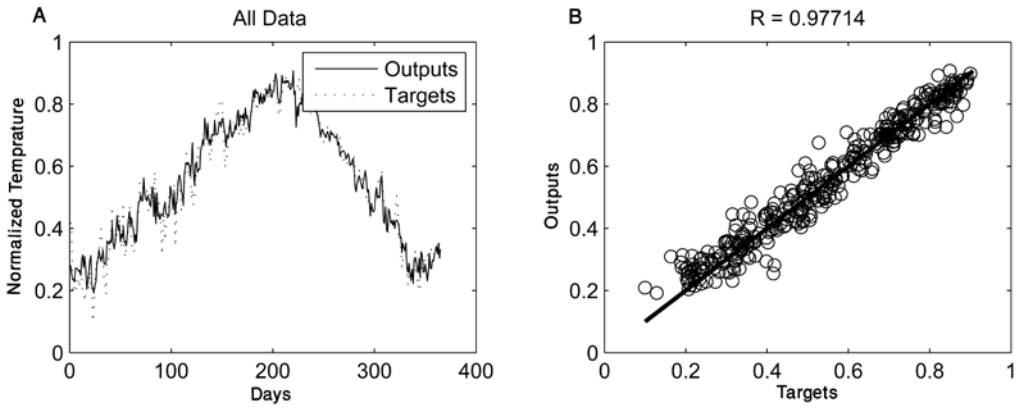

Figure 3.4: (similar to Figure 3.2 but allocated to all the data)