Neural Process Lett (2008) 28:63–79
DOI 10.1007/s11063-008-9082-0

Novel Neuronal Activation Functions for Feedforward Neural Networks

Mehmet Önder Efe

Published online: 21 August 2008

Abstract Feedforward neural network structures have extensively been considered in the literature. In a significant volume of research and development studies hyperbolic tangent type of a neuronal nonlinearity has been utilized. This paper dwells on the widely used neuronal activation functions as well as two new ones composed of sines and cosines, and a sinc function characterizing the firing of a neuron. The viewpoint here is to consider the hidden layer(s) as transforming blocks composed of nonlinear basis functions, which may assume different forms. This paper considers 8 different activation functions which are differentiable and utilizes Levenberg-Marquardt algorithm for parameter tuning purposes. The studies carried out have a guiding quality based on empirical results on several training data sets.

Keywords Activation functions · Dynamical system identification · Levenberg-Marquardt algorithm

1 Introduction

Artificial neural networks have been a useful tool in many disciplines especially in engineering. Many different forms of neural systems were introduced in the past and the goal of performance improvement in neural networks research has been a core issue that never ended. The efforts toward a better performing neural network have concentrated mainly on the following

- Structure selection, e.g. feedforward or feedback configurations
- Learning algorithm, e.g. error backpropagation, LM and so on
- Hybrid structures, e.g. fuzzy-neural, neuro-genetic and so on
- Type of neuron activation, e.g. sigmoid like functions, radial basis functions or others.

M. Ö. Efe (✉)
Department of Electrical and Electronics Engineering, TOBB Economics and Technology University, Söğütözü, Ankara, Turkey
e-mail: onderefe@etu.edu.tr

This paper focuses on the dependence of performance on the type of neuronal activation function. In the past, this was considered to some extent in [1–5]. In [1], traditional sigmoid activation function, logarithmic activation function (`log` in Table 1) and the arctangent type activation function (`atan` in Table 1) are comparatively considered with backpropagation type learning. Among these, the network with `atan` function was found to be the most successful. In [2], sigmoid, sinusoidal ($f(S) = \sin(\pi S)$) and Gaussian type ($f(S) = \exp(-S^2)$) activation functions are compared and among those, the sinusoidal one is seen to display most desirable characteristics. In [3], Wu et al. propose a trainable activation function based network structure. This approach performs a weighted sum of the derived activation functions, which are a differentiated sigmoid, a differentiated Gaussian and an exponential basis. Error backpropagation is used and it is seen that the proposed combination provides good performance at the cost of some computational intensity. In [4], a quadratic sigmoid function, which resembles radial basis functions, is utilized and better performance indications than sigmoid based networks are emphasized. An effort to adapt Type-2 fuzzy activation function for neural systems is presented in [5], where the utility based on synergy of approaches is the essence of the design. An in-depth discussion on methods and integrations can be found in [6]. Although, backpropagation method is frequently used in the cited body of research, due to its convergence speed, we choose Levenberg-Marquardt (LM) training algorithm for parameter tuning purposes [7].

To our best knowledge, this paper presents the most extensive comparison compared to the aforementioned references. Some of the references compare few activation functions yet some other focus on few types of training data sets. This paper considers eight different data sets, eight different activation functions with networks having fourteen different node numbers in the hidden layer. Under these conditions, the research conducted gives a clear idea about when an activation function is advisable.

The main contributions of the paper are as follows (i) the introduction of two novel activation functions described by $f(S) = a\sin(pS) + b\cos(qS)$ and $f(S) = \text{sinc}(S)$, (ii) the figures for guiding a designer which function to choose for a particular sort of a problem, and (iii) the diverse number of alternatives compared and (iv) the performance of the logarithmic and arctangent type of activation functions for mid sized networks. Regarding the hardware implementation, the remarkable performance of networks exploiting the activation function $f(S) = a\sin(pS) + b\cos(qS)$ especially for small sized networks is a prominent feature contributed by the current study.

In the next section, we describe the neuronal activation functions considered in this study. The considered data sets are described in the third section. The simulation results are presented in the fourth section and the concluding remarks are given at the end of the paper.

2 Neuronal Activation Functions

Feedforward neural network structure with sigmoid or hyperbolic tangent type neuronal nonlinearity is probably the most frequently used connectionist architecture utilized in many applications of engineering. The typical network connectivity with m input neurons, R hidden neurons and single output is depicted in Fig. 1. Although the considered network has a standard structure, the research on the discovery and optimization of neuronal details embodying a neuron is still an active field having a strong influence on the overall capabilities of a neural network. In other words, one type of neuron performs better while another does not. The goal of this paper is to draw a conclusion guiding the designers based on several sets of experiments.

Table 1 Adjustable Parameters and Number of Adjustable Parameters for Each Model

Label	Activation function	Number of adjustables associated to the network	Derivative computation
`tanh`	$f = \tanh(S)$	$(m+2)R+1$	$\frac{\partial f(S)}{\partial S} = 1 - f(S)^2$
`polyexp` See [2]	$f = \left(aS^2+bS+c\right)\exp\left(-\lambda S^2\right)$	$(m+2)R+1+4R$	$\frac{\partial f(a,b,c,S)}{\partial a} = S^2\exp\left(-\lambda S^2\right)$, $\frac{\partial f(a,b,c,S)}{\partial b} = S\exp\left(-\lambda S^2\right)$, $\frac{\partial f(a,b,c,S)}{\partial c} = \exp\left(-\lambda S^2\right)$ $\frac{\partial f(a,b,c,S)}{\partial S} = \left(-2\lambda S^3 - 2\lambda bS^2 + (2a-2\lambda c)S + b\right)\exp\left(-\lambda S^2\right)$
`quan` See [8]	$f = \frac{1}{2M+1}\sum_{k=-M}^{M}\tanh(S-\lambda k)$	$(m+2)R+1+R$	$\frac{\partial f(S,\lambda)}{\partial S} = \frac{-1}{2M+1}\sum_{k=-M}^{M}(\tanh(S-\lambda k))^2$ $\frac{\partial f(S,\lambda)}{\partial \lambda} = \frac{1}{2M+1}\sum_{k=-M}^{M}k(\tanh(S-\lambda k))^2$
`sinc`	$f = \begin{cases}\sin(\pi S)/\pi S & S \neq 0\\ 1 & S = 0\end{cases}$	$(m+2)R+1$	$\frac{\partial f(S)}{\partial S} = \begin{cases}(\cos(\pi S) - \mathrm{sinc}(S))/S & S \neq 0\\ 0 & S = 0\end{cases}$
`sincos`	$f = a\sin(pS) + b\cos(qS)$	$(m+2)R+1+4R$	$\frac{\partial f(a,b,p,q,S)}{\partial a} = \sin(pS)$, $\frac{\partial f(a,b,p,q,S)}{\partial b} = \cos(qS)$, $\frac{\partial f(a,b,p,q,S)}{\partial p} = aS\cos(pS)$ $\frac{\partial f(a,b,p,q,S)}{\partial q} = -bS\sin(qS)$, $\frac{\partial f(a,b,p,q,S)}{\partial a} = ap\cos(pS) - bq\sin(qS)$
`wave` See [2]	$f = \left(1-S^2\right)\exp\left(-\lambda S^2\right)$	$(m+2)R+1+R$	$\frac{\partial f(S,\lambda)}{\partial \lambda} = -S^2 f(S,\lambda)$, $\frac{\partial f(S,\lambda)}{\partial S} = 2S(\lambda S^2 - \lambda - 1)e^{-\lambda S^2}$
`atan` See [1]	$f = \mathtt{atan}(S)$	$(m+2)R+1$	$\frac{\partial f(S)}{\partial S} = \frac{1}{1+S^2}$
`log` See [1]	$f = \begin{cases}\ln(S+1) & S \geq 0\\ -\ln(-S+1) & S < 0\end{cases}$	$(m+2)R+1$	$\frac{\partial f(S)}{\partial S} = \frac{1}{1+\lvert S\rvert}$

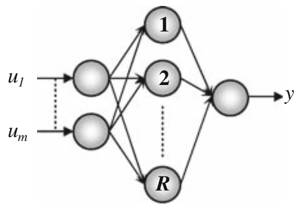

Fig. 1 A feedforward neural network architecture with single hidden layer, m inputs, R hidden neurons and single output

The net sum of the neurons in the hidden layer are computed using (1), where a superscript k denotes the discrete time index and w_{ij}s are the adjustable parameters of the hidden layer.

$$S_i^k = w_{i,j+1}^k + \sum_{j=1}^{m} w_{ij}^k u_j^k, \quad i = 1, 2, \ldots, R \tag{1}$$

The output of the ith hidden neuron is computed by using (2), where the function $f(\bullet)$ is the neuronal nonlinearity and is the entity under investigation here. The overall response (y) of the neural network is obtained through the use of (3), where the output neuron is a linear one having v_is as the adjustable parameters. The use of linear output neuron is deliberate as the output nonlinearity introduces reachability constraint as well as prior information about the bounds of output signal.

$$o_i^k = f\left(S_i^k\right) \tag{2}$$

$$y = v_{i+1}^k + \sum_{i=1}^{R} v_i^k o_i^k \tag{3}$$

The research problem in this paper is the following: Can an alternative form of neuronal activation function yield better results than networks with standard settings? Having this in mind, we utilize the set of activation functions listed in Table 1. In the leftmost column of the table, we give the label associated with each activation function. Briefly, the first row gives the hyperbolic tangent type of the activation, the second line describes a polynomial multiplied by a decaying exponential labeled as `polyexp`, the third line describes a neuronal activation function which is the sum of shifted hyperbolic tangents yielding a multilevel (quantized) activation. This is labeled as `quan`, and the number of different quanta is determined by the variable M, which can be set by the designer. The following one is a `sinc` function used frequently in signal analysis. The activation function given in the fifth row is a scaled sine at one adjustable frequency added to a scaled cosine at another adjustable frequency, which is labeled `sincos`. The activation function inspired from wavelets is given and is labeled `wave`, arctangent function is given next with `atan` label and lastly a logarithmic activation function is described with the label `log`.

The second column of the table describes the analytic form of the function and the number of adjustable parameters of a network for each case is given in the third and the associated derivatives are given in the rightmost column of the table. Clearly, if there is no extra parameter associated to the adopted neuron activation function, then the number of adjustable parameters are those from the weight matrices and the biases, which add up to $(m+2)R+1$ for the network illustrated in Fig. 1.

3 Data Sets

This section introduces the problems utilized to assess capabilities of the neural network models built individually with the aforementioned activation functions. Among the considered eight individual data sets, four contain synthetic data and four are from real life. Two of the real life data are classification examples and the remaining six sets are regression problems.

3.1 Data Sets 1–2: Bioreactor Benchmark Problem

Bioreactor benchmark problem is one of the problems considered for analyzing the effect of activation functions on the overall performance. The process is valuable as its dynamics consists of coupled and nonlinear differential equations, which display a rich set of behaviors containing limit cycles, attractors and repellers [9,10]. The bioreactor is a tank containing a mixture of water, biological cells and nutrients. The tank is fed by an inflow rate ($w^k \in [0, 2]$) and the same amount of mixture is removed from the tank, so that the reaction volume is kept constant. Though characterized by few state variables, the process is a good test bed for benchmarking the performance of numerical data based models and regression approaches. The state of the process is the amount of cells denoted by $c_1^k \in [0,1]$ and the amount of nutrients denoted by $c_2^k \in [0, 1]$. The two difference equations obtained after the discretization of the continuous time process are given in (4)–(5), where $\gamma = 0.48$ and $\beta = 0.02$ are the nutrient inhibition parameter the growth rate parameter, respectively [9–11].

$$c_1^{k+1} = c_1^k + \Delta\left(-c_1^k w^k + c_1^k\left(1 - c_2^k\right)e^{c_2^k/\gamma}\right) \tag{4}$$

$$c_2^{k+1} = c_2^k + \Delta\left(-c_2^k w^k + c_1^k\left(1 - c_2^k\right)e^{c_2^k/\gamma}\frac{1+\beta}{1+\beta-c_2^k}\right) \tag{5}$$

In above, $\Delta = 0.01\,\mathrm{s}$ is the discretization period. The performance of the considered neural network structure is evaluated in predicting the cell mass (c_1^k) under various operating conditions. An initial condition is chosen randomly and (6)–(7) are executed for 50 successive discrete time instants. This operation is repeated for 50 different initial conditions yielding a total of 2500 training samples. A similar strategy is followed for generating the validation data set containing 750 samples. The input vector of the network contains c_1^k, c_2^k, w^k and the auxiliary variables if any (See Fig. 1). The regression problem is to yield accurate predictions for the state variables individually.

3.2 Data Set 3

Although the bioreactor benchmark problem displays a quite rich set of dynamical properties, due to the discretization in (4)–(5), the network is guided dominantly by the previous response of the variable being predicted. For very small Δ, this becomes more apparent in (4)–(5). We therefore perform the tests on different models reported by many research studies. The model considered in [12] is given in (6). The training data is generated as explained for the bioreactor benchmark process. The input vector of the network in Fig. 1 contains x^k, w^k and the auxiliary variables if any. This model is a good candidate to investigate learning of nonlinearities influencing the behavior dominantly in a discrete map.

$$x^{k+1} = \frac{x^k}{1+\left(x^k\right)^2} + \left(w^k\right)^3 \tag{6}$$

For this model, the operating range is chosen as $x^k \in [-1, 1]$ and $w^k \in [-1, 1]$. The expression in (6) emphasizes that the next state of the system is dependent upon the current state and the current input nonlinearly. The regression problem is to obtain a neural network approximating the right hand side of (6) as much accurate as possible.

3.3 Data Set 4

The input/output relation of the third system is given by (7). Although (6) is from a frequently cited resource, the model below is a good example that clearly differentiates the activation functions in terms of the results obtained, which is the main motivation of including this process.

$$x^{k+1} = \frac{-x^k + \sin\left(2\pi x^k\right) w^k}{\sqrt{1 + \left(x^k\right)^2 + \sin\left(2\pi x^k\right)^2}} \tag{7}$$

The training and checking data sets are generated as described for the bioreactor benchmark process and the operating ranges and the network inputs are the same as in Sect. 3.2.

3.4 Data Set 5: Subsonic Cavity Flow

Aerodynamic flow modeling is a highly interesting engineering problem as the corresponding physics is involved directly with the Navier-Stokes equations, turbulence and complex flow geometries. One particular example is the flow past a rectangular cavity. A photo and the schematic view of the experimental setup considered in this paper are given in Fig. 2a,b. The goal of the modeling problem is to predict the floor pressures based on the pressure measurements from several critical locations of the cavity geometry [13–15]. These locations include the signal to the host computer (x_1), the output of the actuator (x_2), the measurement at the receptivity point (x_3), the measurement at cavity trailing edge (x_5) and the measurements before and after the rectangular cutout (x_4, x_7) are also depicted in Fig. 2a. The modeling strategies for such problems typically enjoy decomposition methods and these models are very likely to have many state variables and subject to inaccuracies. The neural network based modeling approach is an alternative technique that utilizes the local information to predict the behavior at the cavity floor (x_6).

The experimental setup is composed of a host computer having a digital signal processor, a filter to remove the spurious flow content at high frequencies, a Titanium diaphragm synthetic actuator and an air system providing the flow at a desired Mach number. Since the

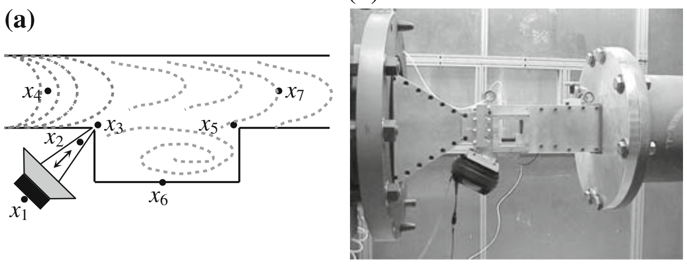

Fig. 2 (**a**) Schematic view and sensor locations on the cavity flow setup (**b**) the view of the experimental setup at The Ohio State University, GDTL

spectral view is rich at Mach 0.25, we consider the modeling problem based on data obtained under the conditions mentioned above.

The training and testing data sets have 2500 and 750 pairs, respectively. The data have been collected at a sampling frequency of 50 kHz. Half of the data have been obtained with a sinusoidal excitation at x_1 with frequency 3920 Hz and magnitude 4.06 V, and the other half are obtained again with a sinusoidal excitement of frequency 3250 Hz and magnitude 2.35 V. The experiments on this setup, data collection scheme, and some preliminary modeling results are discussed in detail in [13].

The neural network based modeling problem is to develop the function $f(\cdot)$ in (8) such that the response matches the real time data precisely and minimum amount of sensory information is exploited.

$$x_6^{k+1} = f(x_1^k, x_3^k, x_5^k, x_6^k, x_6^{k-1}) \tag{8}$$

3.5 Data Set 6: Servo Data Set from UCI Repository

The data provided in [16] describes a prediction problem for a servomechanism's rising time that varies according to several mechanical settings and gain selections. The data set is claimed to describe a highly nonlinear phenomenon motivating the research studies like the one presented here. The set is composed of a total of 167 instances of data, among which the first 100 are used for the training and the remaining 67 are reserved for testing set, which is used to stop training at the best level of generalization.

3.6 Data Set 7: Iris Data Set from UCI Repository

The data provided in [16] describes a classification problem for three classes of Iris plants. The input vector to the classifier contains sepal length, sepal width, petal length and petal width, all in centimeters. The output of the classifier is one of the three labels Iris Setosa, Iris Versicolour or Iris Virginica. In the training we assign -1 for the label Iris Setosa, 0 for the label Iris Versicolour and $+1$ for the label Iris Virginica and convert the classification problem into a tractable regression problem in between the three integers. Thresholding is possible after the training is finished, yet, the performance of the classifier is directly proportional to final RMSE value achieved.

In the UCI repository, 150 instances are provided. The training data has been selected in such a way that every class contributes 40 instances, adding up to 120 instances for three classes, and the remaining 30 instances have been used as testing data set.

3.7 Data Set 8: XOR Data

The most traditional data set used in the research of best neural network configuration is perhaps the XOR data. Since it contains binary entries, the data set describes a highly simple classification in between the two levels of voltage, namely High and Low. Clearly four lines of data are used for training and each particular trial is continued until the completion of maximum number of epochs, which is equal to 250 in this study, as there is no testing data to stop.

4 Simulation Results

During the performance comparison phase, we first choose $R = 2, 4, 5, 6$, which describe small sized networks. This is deliberate as we wish to compare the performance of simple structured networks, which are potentially useful for hardware implementations and are generally more preferable than those introducing computational intensity for better performance. Then we consider the cases with $R = 8$, 10, 12, 14, 16, 18, 20, 25, 40 and 100 to see how the performance is influenced as the number of neurons in the hidden layer increases.

Initially, all adjustable parameters of the network are assigned to small random numbers distributed uniformly in between -0.01 and $+0.01$. This is another deliberate choice as we would like to observe similar initial values for the Root Mean Squared Error (RMSE) levels obtained for the experiments carried out with a particular activation function. This would make it possible to see the effect of an activation function statistically over a set of experiments and remove the possible effect of "lucky" initial conditions.

The neural network described in Sect. 2 is trained 100 times for every one of the neuronal activation functions listed in Table 1, and the achieved RMSE value after the completion of each training trial is recorded. Every individual training trial is terminated after the completion of 250 epochs unless the checking routine stops the process. If the epoch error increases for 5 consecutive epochs, then the training is terminated before 250 epochs are completed. A pseudo code describing the whole process is given below.

Having finished the implementation of the pseudo code in Fig. 3, we have a set of 100 element arrays containing the final values of RMSEs. Denote one such generic array by $\mathrm{A} = \{r_1, r_2, \ldots, r_{100}\}$. Clearly, this array contains information about what could be the achievable best and what is a likely result of a training trial. In order to develop a descriptive idea from the content of array A, a 20-bin histogram of the array A is constructed and the mean of the distribution is computed then to describe the performance associated to a particular A array. Such a method is clearly an unbiased qualifier of a particular set of experiments. Note that, obtaining the mean of A directly would be an incorrect way to follow as there may be a large number in A, and this number, though it is very unlikely to emerge, might dominate the entire set of numbers available in the array A. Having obtained the mean of the distributions associated to each A, since there are 8 different types of activation functions, we follow a grading policy such that the activation function resulting in the best average value is assigned a grade 7 and that associated to the least average value is assigned a grade 0. For each one of the different hidden layer configurations, we obtain the Tables 2–15. For example, in Table 2, $R = 2$ and we see that for the first data set (c_1 prediction of biochemical process denoted by Eq. 4 in the first column), best result is obtained for the `sincos` type activation function and worst result is obtained with `quan` type activation, for which the relevant box is shaded gray. This is clear from the first numeric column of the table. The same interpretation is

```
% Pseudo Code
FOR R = 2,4,5,6,8,10,12,14,16,18,20,25,40,100
    FOR data set = 1 to 8
        FOR nonlinearity = {sincos, sinc, tanh, atan, log, wave, polyexp, quan}
            FOR trial = 1 to 100
                FOR Epoch = 1 to 250
                    Train The Network
                    IF Test Error Increases
                        Terminate this training trial
                    END
                END
                Write the final RMSE value to RMSE(trial)
            END
            SAVE the RMSE array for this particular nonlinearity
        END
    END
END
```

Fig. 3 A pseudo code for the collection of final RMSE arrays

Table 2 The results of grading procedure for $R = 2$

	Eq.(4)	Eq.(5)	Eq.(6)	Eq.(7)	Cavity	Servo	Iris	Xor
`sincos`	7 0.0083	7 0.0232	7 0.0843	7 0.1138	3 0.0921	5 0.5270	6 0.1532	5 0.0002
`sinc`	4 0.0083	6 0.0233	2 0.1331	5 0.1981	1 0.1146	2 0.6850	4 0.1735	6 0.0002
`tanh`	6 0.0083	5 0.0233	5 0.1251	4 0.2321	6 0.0784	3 0.6399	3 0.2012	3 0.0729
`atan`	5 0.0083	4 0.0234	6 0.1237	3 0.2346	7 0.0784	7 0.4597	7 0.1386	4 0.0629
`log`	3 0.0084	3 0.0237	3 0.1289	2 0.2488	5 0.0784	6 0.5226	5 0.1671	1 0.1780
`wave`	1 0.0124	1 0.0387	0 8.9616	0 0.4253	0 0.5814	0 153.6120	0 4.3915	7 0.0002
`quan`	0 0.0272	0 0.0464	4 0.1288	1 0.2875	4 0.0790	1 0.7524	2 0.2085	0 0.4739
`polyexp`	2 0.0097	2 0.0258	1 0.1629	6 0.1639	2 0.1115	4 0.5538	1 0.2086	2 0.0930

Table 3 The results of grading procedure for $R = 4$

	Eq.(4)	Eq.(5)	Eq.(6)	Eq.(7)	Cavity	Servo	Iris	Xor
`sincos`	4 0.0082	6 0.0223	7 0.0123	7 0.0413	7 0.0774	6 0.3800	5 0.1432	5 0.0002
`sinc`	5 0.0082	3 0.0226	5 0.0250	6 0.0610	4 0.0777	2 0.4428	3 0.1481	6 0.0001
`tanh`	7 0.0082	4 0.0224	6 0.0215	5 0.1361	6 0.0775	3 0.4298	7 0.1312	2 0.0051
`atan`	6 0.0082	5 0.0224	4 0.0313	4 0.1478	5 0.0775	7 0.2986	6 0.1321	1 0.0054
`log`	3 0.0082	7 0.0221	2 0.0549	2 0.1769	3 0.0778	5 0.3806	4 0.1462	3 0.0010
`wave`	1 0.0103	2 0.0228	0 0.5323	0 0.4699	0 522.0249	0 929.9038	0 0.2406	7 0.0001
`quan`	0 0.0235	0 0.0432	3 0.0346	1 0.3505	1 0.0855	1 0.6344	1 0.1890	0 0.1935
`polyexp`	2 0.0092	1 0.0288	1 0.1205	3 0.1754	2 0.0786	4 0.4051	2 0.1547	4 0.0003

valid for the remaining columns, each of which is dedicated to a particular data set shown on the top row of the table. Every cell in Table 2 contains a second number to describe the mean of the distribution of A array associated to that particular cell. That is to say, the best performer of the first column of Table 2 is `sincos` with a distribution average 0.0083, the worst performer is `quan` with a distribution average 0.0272. Clearly, these values guide the designer about positioning every single case among the others more precisely.

Regarding the other network configurations, a similar tabling is performed and the results are tabulated in Tables 3–15. Naturally, such a sorting is a good way of distinguishing the well performing and poor performing approaches. Having these data in front, three more tables are given to interpret the total behavior under three categories. In Table 16, the comparison of total grade of each case is presented. The first four data sets, namely the biochemical process (4)–(5) and the processes described by (6) and (7) are separated as synthetic data sets and the remaining four, i.e. cavity flow, servo data, Iris data and XOR table, are called the real (or real-time) data sets. These subsets are also compared among themselves to figure out the capabilities of neural networks, which might demonstrate excellence for one subset whereas performing poor for the other is a possibility as well.

Table 4 The results of grading procedure for $R = 5$

	Eq.(4)	Eq.(5)	Eq.(6)	Eq.(7)	Cavity	Servo	Iris	Xor
sincos	4 0.0082	5 0.0222	7 0.0114	7 0.0340	7 0.0772	4 0.3545	4 0.1415	5 0.0002
sinc	5 0.0082	3 0.0224	6 0.0132	6 0.0402	1 0.0835	2 0.3641	3 0.1426	6 0.0001
tanh	7 0.0081	4 0.0223	4 0.0165	5 0.0715	6 0.0773	5 0.3378	7 0.1305	1 0.0030
atan	6 0.0081	6 0.0220	5 0.0140	4 0.0869	5 0.0773	7 0.2433	6 0.1313	2 0.0029
log	3 0.0082	7 0.0219	2 0.0632	2 0.1408	3 0.0776	6 0.3369	5 0.1379	3 0.0008
wave	2 0.0090	2 0.0229	0 0.2465	1 0.3225	4 0.0775	0 1.5708e8	0 0.2324	7 0.0001
quan	0 0.0171	0 0.0394	3 0.0308	0 0.3430	0 0.0866	1 0.6304	1 0.1787	0 0.1643
polyexp	1 0.0093	1 0.0302	1 0.1007	3 0.1377	2 0.0786	3 0.3636	2 0.1471	4 0.0004

Table 5 The results of grading procedure for $R = 6$

	Eq.(4)	Eq.(5)	Eq.(6)	Eq.(7)	Cavity	Servo	Iris	Xor
sincos	3 0.0082	4 0.0221	7 0.0114	7 0.0335	7 0.0770	2 0.3337	2 0.1400	5 0.0001
sinc	5 0.0082	3 0.0223	5 0.0126	6 0.0352	4 0.0771	3 0.3242	3 0.1398	6 0.0001
tanh	7 0.0081	5 0.0220	4 0.0159	5 0.0535	6 0.0771	6 0.2717	7 0.1305	2 0.0024
atan	6 0.0081	6 0.0218	6 0.0123	4 0.0614	5 0.0771	7 0.2432	6 0.1313	1 0.0029
log	4 0.0082	7 0.0216	2 0.0342	3 0.1043	2 0.0774	5 0.2938	5 0.1375	3 0.0007
wave	1 0.0093	2 0.0228	0 0.1667	1 0.2838	3 0.0774	0 1.0321	0 0.3692	7 0.0001
quan	0 0.0183	0 0.0399	3 0.0258	0 0.3708	0 0.0870	1 0.5201	1 0.1747	0 0.1241
polyexp	2 0.0088	1 0.0296	1 0.0892	2 0.1344	1 0.0784	4 0.3174	4 0.1392	4 0.0003

Table 6 The results of grading procedure for $R = 8$

	Eq.(4)	Eq.(5)	Eq.(6)	Eq.(7)	Cavity	Servo	Iris	Xor
sincos	3 0.0082	4 0.0221	6 0.0113	7 0.0331	6 0.0768	2 0.3155	2 0.1393	5 0.0001
sinc	4 0.0082	3 0.0223	5 0.0123	6 0.0336	7 0.0767	3 0.2861	4 0.1372	6 0.0001
tanh	7 0.0081	5 0.0219	4 0.0177	5 0.0385	4 0.0769	5 0.2524	7 0.1295	2 0.0015
atan	6 0.0081	6 0.0216	7 0.0106	4 0.0436	5 0.0769	7 0.1612	6 0.1310	1 0.0015
log	5 0.0081	7 0.0213	2 0.0286	3 0.0717	2 0.0770	4 0.2611	5 0.1324	3 0.0006
wave	1 0.0098	2 0.0229	0 0.1440	1 0.1836	3 0.0769	1 0.4739	3 0.1375	7 0.0001
quan	0 0.0116	0 0.0367	3 0.0178	0 0.3694	0 0.0858	0 0.5530	0 0.1649	0 0.0913
polyexp	2 0.0088	1 0.0292	1 0.0861	2 0.1067	1 0.0781	6 0.2521	1 0.1405	4 0.0002

Table 7 The results of grading procedure for $R = 10$

	Eq.(4)	Eq.(5)	Eq.(6)	Eq.(7)	Cavity	Servo	Iris	Xor
sincos	3 0.0081	4 0.0221	6 0.0113	7 0.0328	5 0.0767	3 0.2999	1 0.1397	5 0.0001
sinc	4 0.0081	3 0.0223	4 0.0122	4 0.0438	7 0.0764	4 0.2591	3 0.1357	6 0.0001
tanh	6 0.0081	5 0.0219	2 0.0169	6 0.0334	4 0.0767	1 0.5531	7 0.1296	2 0.0013
atan	5 0.0081	6 0.0217	7 0.0106	5 0.0343	6 0.0766	7 0.1503	5 0.1318	1 0.0014
log	7 0.0081	7 0.0209	5 0.0115	3 0.0577	2 0.0768	5 0.2283	6 0.1298	3 0.0006
wave	1 0.0094	2 0.0229	0 0.1390	1 0.1730	3 0.0768	0 0.5990	4 0.1325	7 0.0001
quan	0 0.0138	0 0.0364	3 0.0162	0 0.3904	0 0.0830	2 0.4241	0 0.1656	0 0.0747
polyexp	2 0.0086	1 0.0279	1 0.1081	2 0.0906	1 0.0784	6 0.2262	2 0.1386	4 0.0002

Table 8 The results of grading procedure for $R = 12$

	Eq.(4)	Eq.(5)	Eq.(6)	Eq.(7)	Cavity	Servo	Iris	Xor
sincos	3 0.0082	4 0.0221	5 0.0113	5 0.0327	2 0.0766	3 0.2908	1 0.1403	5 0.0001
sinc	4 0.0082	3 0.0224	4 0.0122	4 0.0332	7 0.0763	5 0.2481	3 0.1345	6 0.0001
tanh	5 0.0081	5 0.0218	3 0.0149	7 0.0299	4 0.0765	1 0.5416	7 0.1295	2 0.0010
atan	6 0.0081	6 0.0217	7 0.0106	6 0.0302	3 0.0765	4 0.2677	4 0.1318	1 0.0011
log	7 0.0081	7 0.0210	6 0.0111	3 0.0515	5 0.0765	6 0.2143	6 0.1305	3 0.0005
wave	1 0.0089	2 0.0228	0 0.1306	1 0.1447	6 0.0764	0 0.5779	5 0.1314	7 0.0001
quan	0 0.0120	0 0.0362	2 0.0159	0 0.3489	0 0.0821	2 0.3675	0 0.1676	0 0.0743
polyexp	2 0.0085	1 0.0280	1 0.0954	2 0.0805	1 0.0783	7 0.2118	2 0.1396	4 0.0002

Table 9 The results of grading procedure for $R = 14$

	Eq.(4)	Eq.(5)	Eq.(6)	Eq.(7)	Cavity	Servo	Iris	Xor
sincos	3 0.0082	4 0.0221	5 0.0112	5 0.0329	2 0.0765	3 0.2754	2 0.1397	5 0.0001
sinc	4 0.0082	3 0.0224	4 0.0121	4 0.0335	7 0.0761	4 0.2536	3 0.1342	6 0.0001
tanh	6 0.0081	5 0.0219	3 0.0137	6 0.0300	4 0.0764	0 0.5883	6 0.1294	2 0.0009
atan	5 0.0081	6 0.0217	6 0.0111	7 0.0292	3 0.0765	7 0.1748	4 0.1317	1 0.0010
log	7 0.0081	7 0.0206	7 0.0108	3 0.0454	5 0.0763	5 0.2096	5 0.1294	3 0.0005
wave	2 0.0085	2 0.0229	0 0.1311	1 0.1164	6 0.0763	1 0.4773	7 0.1289	7 0.0001
quan	0 0.0092	0 0.0373	2 0.0150	0 0.3415	0 0.0823	2 0.3697	0 0.1709	0 0.0630
polyexp	1 0.0085	1 0.0272	1 0.1040	2 0.0800	1 0.0785	6 0.1964	1 0.1432	4 0.0002

Table 10 The results of grading procedure for $R = 16$

	Eq.(4)	Eq.(5)	Eq.(6)	Eq.(7)	Cavity	Servo	Iris	Xor
sincos	3 0.0082	4 0.0221	5 0.0112	5 0.0329	2 0.0765	4 0.2706	2 0.1410	5 0.0001
sinc	4 0.0082	3 0.0224	4 0.0120	3 0.0758	6 0.0761	5 0.2531	3 0.1339	6 0.0001
tanh	6 0.0081	5 0.0219	3 0.0132	6 0.0290	3 0.0764	0 0.7728	6 0.1295	2 0.0009
atan	5 0.0081	6 0.0218	6 0.0111	7 0.0283	4 0.0763	2 0.3260	4 0.1318	1 0.0009
log	7 0.0081	7 0.0206	7 0.0108	4 0.0422	7 0.0760	6 0.2016	5 0.1299	3 0.0005
wave	1 0.0088	2 0.0229	0 0.1258	1 0.1083	5 0.0763	1 0.3262	7 0.1291	7 0.0001
quan	0 0.0108	0 0.0353	2 0.0158	0 0.3597	0 0.0829	3 0.3194	0 0.1666	0 0.0610
polyexp	2 0.0084	1 0.0268	1 0.134	2 0.0760	1 0.0787	7 0.1726	1 0.1423	4 0.0002

Table 11 The results of grading procedure for $R = 18$

	Eq.(4)	Eq.(5)	Eq.(6)	Eq.(7)	Cavity	Servo	Iris	Xor
sincos	3 0.0082	4 0.0221	5 0.0112	5 0.0330	2 0.0765	4 0.2576	2 0.1413	5 0.0001
sinc	4 0.0081	3 0.0225	4 0.0120	3 0.0674	6 0.0760	5 0.2462	3 0.1343	6 0.0001
tanh	6 0.081	5 0.0218	3 0.0129	6 0.0302	3 0.0763	0 0.4932	6 0.1297	2 0.0008
atan	5 0.0081	6 0.0218	7 0.0107	7 0.0280	4 0.0763	1 0.3431	4 0.1322	1 0.0009
log	7 0.0081	7 0.0207	6 0.0107	4 0.0405	7 0.0757	6 0.1905	5 0.1300	3 0.0004
wave	1 0.0087	2 0.0228	0 0.1280	1 0.1076	5 0.0763	2 0.3205	7 0.1297	7 0.0001
quan	0 0.0096	0 0.0352	2 0.0155	0 0.3357	0 0.0811	3 0.2849	0 0.1698	0 0.0590
polyexp	2 0.0084	1 0.0270	1 0.1043	2 0.0795	1 0.0789	7 0.1567	1 0.1430	4 0.0002

Table 12 The results of grading procedure for $R = 20$

	Eq.(4)	Eq.(5)	Eq.(6)	Eq.(7)	Cavity	Servo	Iris	Xor
sincos	3 0.0082	4 0.0221	5 0.0111	5 0.0332	2 0.0764	3 0.2656	2 0.1405	5 0.0001
sinc	4 0.0082	3 0.0225	4 0.0119	4 0.0336	6 0.0760	4 0.2496	3 0.1339	6 0.0001
tanh	6 0.0081	5 0.0219	3 0.0132	6 0.0306	4 0.0763	0 0.7388	6 0.1302	2 0.0007
atan	5 0.0081	6 0.0218	6 0.0107	7 0.0283	5 0.0763	1 0.3448	4 0.1319	1 0.0008
log	7 0.0081	7 0.0207	7 0.0107	3 0.0385	7 0.0756	6 0.1834	5 0.1308	3 0.0004
wave	2 0.0082	2 0.0229	0 0.1278	1 0.1012	3 0.0763	2 0.2703	7 0.1292	7 0.0001
quan	0 0.0095	0 0.0347	2 0.0158	0 0.3450	0 0.0807	5 0.2362	0 0.1707	0 0.0545
polyexp	1 0.0084	1 0.0269	1 0.0876	2 0.0796	1 0.0792	7 0.1498	1 0.1436	4 0.0002

Table 13 The results of grading procedure for $R = 25$

	Eq.(4)	Eq.(5)	Eq.(6)	Eq.(7)	Cavity	Servo	Iris	Xor
sincos	3 0.0082	4 0.0223	5 0.0110	5 0.0331	3 0.0765	3 0.2567	2 0.1427	5 0.0001
sinc	4 0.0082	3 0.0225	4 0.0118	4 0.0335	6 0.0758	4 0.2355	3 0.1346	6 0.0001
tanh	6 0.0081	5 0.0221	3 0.0129	6 0.0300	4 0.0763	0 0.7967	6 0.1299	2 0.0007
atan	5 0.0081	6 0.0218	6 0.0109	7 0.0276	5 0.0762	1 0.4350	5 0.1322	1 0.0007
log	7 0.0081	7 0.0210	7 0.0106	3 0.0355	7 0.0751	6 0.1840	4 0.1324	3 0.0004
wave	2 0.0082	2 0.0229	0 0.1332	1 0.1052	2 0.0766	5 0.2055	7 0.1293	7 0.0001
quan	0 0.0107	0 0.0346	2 0.0153	0 0.3686	0 0.0818	2 0.2761	0 0.1777	0 0.0524
polyexp	1 0.0083	1 0.0258	1 0.0806	2 0.0750	1 0.0794	7 0.1521	1 0.1453	4 0.0002

Table 14 The results of grading procedure for $R = 40$

	Eq.(4)	Eq.(5)	Eq.(6)	Eq.(7)	Cavity	Servo	Iris	Xor
sincos	3 0.0082	2 0.0228	6 0.0110	4 0.0337	2 0.0764	2 0.2382	2 0.1462	5 0.0001
sinc	4 0.0081	5 0.0225	5 0.0116	3 0.0349	6 0.0763	4 0.2247	3 0.1349	6 0.0001
tanh	6 0.0081	4 0.0225	4 0.0123	6 0.0308	4 0.0763	1 0.5413	5 0.1313	2 0.0005
atan	5 0.0081	6 0.0220	3 0.0125	7 0.0280	5 0.0763	0 0.6297	4 0.1323	1 0.0006
log	7 0.0081	7 0.0218	7 0.0106	5 0.0313	7 0.0745	5 0.2210	6 0.1307	3 0.0004
wave	2 0.0082	3 0.028	0 0.1228	1 0.0937	3 0.0764	7 0.1589	7 0.1291	7 0.0001
quan	0 0.0120	0 0.0308	2 0.0148	0 0.4508	0 0.0802	3 0.2326	0 0.1905	0 0.0417
polyexp	1 0.0082	1 0.0247	1 0.0448	2 0.0665	1 0.0800	6 0.2050	1 0.1477	4 0.0002

Table 15 The results of grading procedure for $R = 100$

	Eq.(4)	Eq.(5)	Eq.(6)	Eq.(7)	Cavity	Servo	Iris	Xor
sincos	2 0.0082	3 0.0231	6 0.0108	5 0.0337	6 0.0765	3 0.2055	1 0.1523	5 0.0001
sinc	3 0.0082	7 0.0223	5 0.0115	3 0.0355	3 0.0796	5 0.1867	4 0.1372	6 0.0001
tanh	7 0.0081	2 0.0231	4 0.0128	6 0.0320	4 0.0766	1 0.5136	6 0.1313	2 0.0004
atan	4 0.0081	4 0.0230	3 0.0134	7 0.0295	5 0.0766	0 0.8380	5 0.1323	1 0.0005
log	5 0.0081	6 0.0223	7 0.0106	4 0.0346	7 0.0751	2 0.2273	2 0.1474	3 0.0003
wave	6 0.0081	5 0.0227	0 0.0946	1 0.0947	0 0.0831	6 0.0786	7 0.1305	7 0.0001
quan	0 0.0127	0 0.0282	2 0.0137	0 0.4689	1 0.0818	4 0.1893	0 0.1997	0 0.0370
polyexp	1 0.0082	1 0.0238	1 0.0382	2 0.0804	2 0.0797	7 0.0667	3 0.1446	4 0.0003

Table 16 Overall comparison of the results for the entire sets of data

$R =$	2	4	5	6	8	10	12	14	16	18	20	25	40	100	Average
sincos	47	47	43	37	35	34	28	29	30	30	29	30	26	31	34.0
sinc	30	34	32	35	38	35	36	35	34	34	34	34	36	36	34.5
tanh	35	40	39	42	39	33	34	32	31	31	32	32	32	32	34.6
atan	43	38	41	41	42	42	37	39	35	35	35	36	31	29	37.4
log	28	29	31	31	31	38	43	42	46	45	45	44	47	36	38.3
wave	9	10	16	14	18	18	22	26	24	25	24	26	30	32	21.0
quan	12	7	5	5	3	5	4	4	5	5	7	4	5	7	5.6
polyexp	20	19	17	19	18	19	20	17	19	19	18	18	17	21	18.6

The information conveyed through the Tables 2–15 is informative yet still implicit as each table is dedicated to a particular neural network configuration determined by the parameter R. In the following three tables, the collected information is evaluated for the goal of extracting a guiding knowledge for the practitioners of neural networks. For this purpose, in Table 16, the total number of grades for each approach is presented. For every studied value of R, the activation type with the highest grade is designated in a black box, and its follower is designated in a box shaded gray. The rightmost column gives the average grade over the rows of the table. In a sense, this average can be considered as the overall performance of each neuronal activation function.

Three important findings from Table 16 are as follows:

- For small networks that have few neurons, `sincos` type activation function is more advisable than the other alternatives.
- `log` type neuronal activation function has the highest average value, which is 38.3, and this activation function is the best for mid sized and large sized network structures.
- As opposed to the traditional choice, `tanh` type activation function demonstrates an average performance and it is poorer than `sincos` for small sized networks.

Briefly, if the number of neurons is a critical parameter, e.g. in analog implementations, then utilizing `sincos` type activation function is more advantageous, otherwise one's good choice would be to select the `log` type activation function as its performance is generally good.

When we confine ourselves to the results obtained with the synthetic data, we do not see to much difference from the major results drawn from Table 16. Following observations are visible from Table 17.

- Networks exploiting `sincos` type neuronal activation function maintain the good performance for small R.
- `atan` type neuronal activation function has the highest average value, which is 22.2, and this activation function is either the best or the second best in most of the cases. Noting that the `log` type activation function's average grade and its behavior for varying R indicates that the `log` type of activation function is still a powerful candidate.
- Similar to the previous table, for small R, `tanh` type activation function is still poorer than network structures exploiting `sincos` function.

Finally, the performances for the last four data sets have been considered separately in Table 18. These data sets include the cavity flow data, servo data, Iris data and XOR data. The values in Table 18 do not consistently point a specific type of neuronal activation. According to the results seen in the table, following remarks need emphasis.

- For small sized networks, `sincos` and `atan` type activation functions seem to be the best approaches and for the remaining R values, `sincos` behaves very similar to `tanh` type neuronal activation function utilizing network configurations.

Table 17 Overall comparison of the results for the synthetic data

$R =$	2	4	5	6	8	10	12	14	16	18	20	25	40	100	Average
`sincos`	28	24	23	21	20	20	17	17	17	17	17	17	15	16	19.2
`sinc`	17	19	20	19	18	15	15	15	14	14	15	15	17	18	16.5
`tanh`	20	22	20	21	21	19	20	20	20	20	20	20	20	19	20.1
`atan`	18	19	21	22	23	23	25	24	24	25	24	24	21	18	22.2
`log`	11	14	14	16	17	22	23	24	25	24	24	24	26	22	20.4
`wave`	2	3	5	4	4	4	4	5	4	4	5	5	6	12	4.8
`quan`	5	4	3	3	3	3	2	2	2	2	2	2	2	2	2.6
`polyexp`	11	7	6	6	6	6	6	5	6	6	5	5	5	5	6.1

Table 18 Overall comparison of the results for the real (or real-time) data

$R =$	2	4	5	6	8	10	12	14	16	18	20	25	40	100	Average
`sincos`	19	23	20	16	15	14	11	12	13	13	12	13	11	15	14.8
`sinc`	13	15	12	16	20	20	21	20	20	20	19	19	19	18	18.0
`tanh`	15	18	19	21	18	14	14	12	11	11	12	12	12	13	14.4
`atan`	25	19	20	19	19	19	12	15	11	10	11	12	10	11	15.2
`log`	17	15	17	15	14	16	20	18	21	21	21	20	21	14	17.9
`wave`	7	7	11	10	14	14	18	21	20	21	19	21	24	20	16.2
`quan`	7	3	2	2	0	2	2	2	3	3	5	2	3	5	2.9
`polyexp`	9	12	11	13	12	13	14	12	13	13	13	13	12	16	12.6

- For mid-sized networks, `sinc` and `atan` type functions perform well and `sinc` type activation function becomes the best in average, with a grade 18.0.
- Surprisingly, for large size networks, the best performing network structure is the one utilizing wave type activation function. For these networks, the table shows that `log` type activation function is a good alternative as well.
- Similar to the previous two tables, `tanh` type activation function demonstrates an average performance according to the table.

Concatenating the three tables, one sees that `sincos` type neuronal nonlinearity performs well for small R. Particularly for synthetic data, this type of neuronal activation function yields significantly better results than the others. Again for small sized networks, in average, `sinc` type activation function performs good for real data. The best performer and its follower are scattered in the table concerning the real data and considering the three tables together, one sees that the `sincos` type activation function would be a good choice if the small network size is a constraint.

As a last issue, a comparison of average computing times denoted by T_A in microseconds for the considered activation functions are presented in Table 19. Every activation function in the left column is presented an argument and the neuron output is computed for 2,000,000 randomly chosen input values. The elapsed times are measured and averaged to obtain the T_A value of the same row. According to the table, it is seen that the highest value is encountered with the `sinc` type and this is still affordable when the capabilities of today's microprocessors are taken into account.

Clearly, the values seen above put forth a tradeoff. For a fixed network structure, if the achievable smallest RMSE values for a given data set were the same for all eight activation functions, the designer's choice would be the one having smallest T_A value to minimize possible propagation delays caused by the neural network. On the other hand, under the same condition, the designer might choose the minimum-number-of-adjustable-parameters neural model to minimize the delays for on line learning applications. This discussion shows that the tradeoff in choosing the most preferable neuronal activation function is tightly coupled with the priorities of the designer as well as the nature of the problem in hand.

Table 19 A comparison of the computational complexity obtained on a Pentium II computer

Nonlinearity type	T_A
`atan`	0.3435
`wave`	0.3830
`polyexp`	0.3840
`tanh`	0.6160
`sincos`	1.1020
`quan`	2.7500
`log`	6.1090
`sinc`	104.3360

The major contribution of this paper is to illustrate that a neural network's representational performance could be made better if the activation scheme for a neuron is reinterpreted. Though not based upon an inspiration from biology, `sincos` and `sinc` type of neuron firing schemes are proposed and among them `sincos` is found to be very useful for small R.

5 Conclusions

This paper presents a comparison of neuronal activation functions utilized mostly in neural network applications. In order to render the results distinguishable, we consider eight different data sets and eight different neuronal activation functions for a given network configuration. The presented research covers single hidden layer networks having 2, 4, 5, 6, 8, 10, 12, 14, 16, 18, 20, 25, 40 and 100 hidden neurons for all experiments described above. Every training trial is repeated 100 times and the final value of the RMSE is recorded. Clearly, a total of 89,600 training trials, each composed of 250 epochs, carry significant information about the effect of neuronal activation function.

The first important conclusion drawn from the tables is that the networks having `sincos` type activation function, proposed in this paper, display better performance than its alternatives operated with small R. This is a substantially important finding letting the designers and engineers build more compact neural systems displaying the same performance of the traditional yet more complicated systems.

Second conclusion of the presented study is the consistently good performance of `log` type of activation function employing neural networks. Considering the widespread use of hyperbolic tangent (`tanh`) type neural networks, it is believed that this is another guiding conclusion for the practitioners of connectionist models.

In sum, for small sized networks, the usefulness of the neuronal activation function named `sincos` and described by $f(S) = a\sin(pS) + b\cos(qS)$ is one; for mid sized networks, usefulness of `atan` and `log` type of activation function instead of hyperbolic tangent function is another major contribution of this paper.

Future work of the author aims at implementing the best performing approaches for the identification and control of unmanned lightweight systems.

Acknowledgements This work was supported by TOBB Economics and Technology University, BAP Program, under contract no ETÜ-BAP-2006/04 and TÜBİTAK contract no 107E137.

References

1. Kamruzzaman J, Aziz SM (2002) A note on activation function in multilayer feedforward learning. In: Proc. of the 2002 int. joint. conf. on Neural Networks (IJCNN ’02), 12–17 May, vol 1, pp 519–523
2. Hara K ,Nakayamma K (1994) Comparison of activation functions in multilayer neural network for pattern classification. In: IEEE world congress on computational intelligence, 1994 IEEE in conf. on neural networks, 27 June–2 July, vol 5, pp 2997–3002
3. Wu Y, Zhao M, Ding X (1997) Beyond weights adaptation: a new neuron model with trainable activation function and its supervised learning. In: Int. conf. on neural networks, 9–12 June, vol 2, pp 1152–1157
4. Karaköse M, Akın E (2004) Type-2 fuzzy activation function for multilayer feedforward neural networks. In: IEEE int. conf. on systems, man and cybernetics, 10–13 Oct., vol 4, pp 3762–3767
5. Chiang C-C, Fu H-C (1992) A variant of second-order multilayer perceptron and its application to function approximations. In: Int. joint. conf. on neural networks (IJCNN ’92), 7–11 June, vol 3, pp 887–892
6. Jang J-SR, Sun C-T, Mizutani E (1997) Neuro-fuzzy and soft computing. PTR Prentice-Hall
7. Hagan MT, Menhaj MB (1994) Training feedforward networks with the Marquardt algorithm. IEEE Trans Neural Netw NN-5:989–993
8. Kretzschmar R, Büeler R, Karayiannis NB, Eggimann F (2000) Quantum neural networks versus conventional feedforward neural networks: an experimental study. In: Proc. of the 2000 IEEE signal processing society workshop on neural networks for signal processing X, Dec. 11–13, vol 1, pp 328–337
9. Efe MÖ, Abadoğlu E, Kaynak O (1999) A novel analysis and design of a neural network assisted nonlinear controller for a bioreactor. Int J Robust Nonlinear Control 9(11):799–815
10. Puskorius GV, Feldkamp LA (1990) Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks. IEEE Trans Neural Netw 5(2):279–297
11. Ungar LH (1997) A bioreactor benchmark for adaptive-network based process control. In: Miller WT III, Sutton RS, Werbos PJ (eds) Neural networks for control. MIT Press, pp 387–402
12. Narendra KS, Parthasarathy K (1990) Identification and control of dynamical systems using neural networks. IEEE Trans Neural Netw 1:4–27
13. Efe MÖ, Debiasi M, Yan P, Özbay H, Samimy M (2008) Neural network based generalization and modeling of subsonic cavity flow regimes. Int J Syst Sci 39(2):105–117
14. Samimy M, Debiasi M, Caraballo E, Malone J, Little J, Özbay H, Efe MÖ, Yan P, Yuan X, DeBonis J, Myatt JH, Camphouse RC (2004) Exploring strategies for closed-loop cavity flow control. In: 42nd AIAA Aerospace Sciences meeting and exhibit, January 5–8, Reno, Nevada, U.S.A., 2004 (Paper # AIAA 2004-0576)
15. Debiasi M, Samimy M (2004) Logic-based active control of subsonic cavity-flow resonance. AIAA J 42:1901–1909
16. Asuncion A, Newman DJ (2007) UCI machine learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. University of California, School of Information and Computer Science, Irvine, CA