Estimation Aspects of Signal Spectral Components Using Neural Networks

Viorel Nicolau and Mihaela Andrei

Department of Electronics and Telecommunications,
"Dunarea de Jos" University of Galati 47 Domneasca St.,
Galati 800008, Galati, Romania
{viorel.nicolau,mihaela.andrei}@ugal.ro

Abstract. Many neural network models have been mathematically demonstrated to be universal approximators. For accurate function approximation, the number of samples in the training data set must be high enough to cover the entire input data space. But this number increases exponentially with the dimension of the input space, increasing the space- and time-complexity of the learning process. Hence, the neural function approximation is a complex task for problems with high dimension of the input space, like those based on signal spectral analysis. In this paper, some aspects of neural estimation of signal spectral components are discussed. The goal is to find a feed-forward neural network (FFNN) model for estimating spectral components of a signal, with computational complexity comparable with Fast Fourier Transform (FFT) algorithm, but easier to implement in hardware. Different FFNN architectures, with different data sets and training conditions, are analyzed. A butterfly-like FFNN (BFFNN) was proposed, which has much less weight connections and better performance than FFNN.

Keywords: Feed-forward neural networks, neural estimator, spectral components, signal processing, FFT algorithm.

1 Introduction

Considering the approximation problem with neural networks (NN), the main design objective is to find a neural network which is a good approximator to some desired input-output mapping. Many neural network models have been mathematically demonstrated to be universal approximators.

The related results include proofs for the conventional multilayer perceptron [5], the radial basis function (RBF) NN [16], the rational function NN [13].

The feed-forward neural networks (FFNN), with a variety of activation functions, can be used as universal approximators [4], [5], [12], [8]. For accurate function approximation and good generalization, many aspects must be taken into account, regarding neural network topology [2], [18], training data set selection, and learning algorithm [3], [6].

For problems with small dimension of the input space, the main learning concern is related to over-training, which can lead to poor generalization capability of the

V.E. Balas et al. (Eds.): Soft Computing Applications, AISC 195, pp. 521–530.
springerlink.com

network and this fact must be considered when the neural network is designed and trained. In addition, the network size must be bounded, much less than the number of training samples. Otherwise, the network just memorizes the training samples, resulting in poor generalization [6].

If the dimension of the input space is big, the time complexity of the learning process increases, and good approximation capability is hard to be reached for the entire domain of interest in the input space [1]. The role of input dimension on function approximation is studied in [10], for various norms, and target sets using linear combinations of adjustable computational units. It results that for good approximation, input upper bounds must decrease, as the input space dimension increases, which means that the smoothness of function being approximated would have to be increased.

The theory of function approximation by neural networks is the basis to many neural applications such as pattern recognition, data compression, time series prediction, process state estimation, adaptive control, etc.

In signal processing, there are many ANN applications combined with spectral analysis, using Fast Fourier Transform (FFT) algorithm. Frequently, the problems have small dimension of the input space, and frequency spectra are used as input data for ANN. In this case, neural networks are used as classifiers of signal spectral components computed with FFT [14], [15].

If the input space dimension is high, ANN applications focus on few spectral components, like in power industry, where FFNN are widely used for harmonics analysis and prediction [20], [11]. FFT computation using cellular neural networks (CNN) was studied in [17], [19], where cell neighborhood is defined from functional rather than topological point of view. In CNN, each cell is connected only to its n-nearest neighborhood cells.

In this paper, some aspects of neural estimation of signal spectral components are discussed. The goal is to find a FFNN model to estimate several spectral components, with computational complexity comparable with FFT algorithms. Such FFNN spectral estimator has advantages over FFT algorithm, even if the model is implemented on sequential machines. It is easier to implement than FFT algorithm, which requires specialized processors with bit-reversed addressing mode.

Different FFNN architectures, with different data sets and training conditions, are analyzed. A new FFNN was proposed, with local connections of the inputs to the hidden layers, forming input butterflies like in radix-2 FFT algorithm.

The paper is organized as follows. In section 2 some elements of signal spectral analysis are discussed. In section 3, aspects of neural approximation and generalization, along with neural prediction techniques are presented. Section 4 describes some simulation results based on different neural architectures, training data sets and noise conditions. Conclusions are presented in section 5.

2 Elements of Signal Spectral Analysis

A physical signal $x(t)$ has finite energy, and it is referred as a finite energy process. If all process characteristics are time invariant, then the process is called stationary. If a

stationary process has its mean values equals with corresponding temporal mean values across of a representative process instance, then the process is called ergodic. Signal types as ergodic processes are studied in this paper.

Let $x_k(t)$ be a representative instance of an ergodic process $x(t)$ into finite time interval T, as shown in Fig. 1.

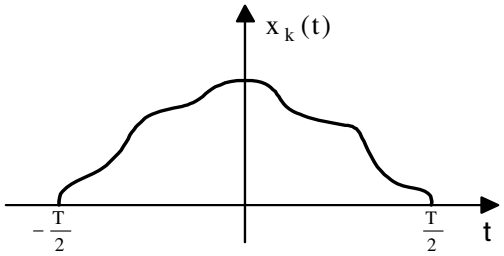

Fig. 1. Representative instance $x_k(t)$ of the ergodic process $x(t)$

The energy of $x_k(t)$ instance is:

$$E = \int_{-\infty}^{\infty} x_k^2(t)\,dt = \frac{1}{2\pi}\int_{-\infty}^{\infty} \left|X_k(j\omega)\right|^2 d\omega, \tag{1}$$

where $X_k(j\omega)$ is the continuous-time Fourier transform of $x_k(t)$. If the process energy is finite, then the integrals in (1) are bounded.

In signal spectral analysis, frequency spectra are desired, along with spectral component characteristics, such as magnitude of spectral components, and signal energy or power distribution into the spectrum. Frequency spectra are computed, according to input signal type, continuous- or discrete-time signal. For continuous-time signals $x(t)$, integral transforms, like continuous-time Fourier transform (CTFT), are used to compute frequency spectra, which are complex continuous functions in the frequency domain. CTFT analytical expression is:

$$X(j\omega) = \int_{-\infty}^{\infty} x(t)\cdot e^{-j\omega t}\,dt \tag{2}$$

Digital signal processing is based on discrete-time signals $x(n)$, as measurements of physical continuous-time signals $x(t)$, at discrete moments of time, $t = nT_S$, where T_S = sampling period, and n = sampling time index (integer). The samples are obtained with the sampling frequency $f_S = 1/T_S$. In this case, frequency spectra are computed with discrete transforms, like discrete-time Fourier transform (DTFT), which are also complex continuous functions in the frequency domain. DTFT analytical expression is obtained by changing integral symbol into infinite series with discrete moments of time:

$$X(j\omega) = \sum_{n=-\infty}^{\infty} T_S \cdot x(nT_S)\cdot e^{-j\omega\cdot nT_S} = \sum_{n=-\infty}^{\infty} x[n]\cdot e^{-j\omega\cdot n} \tag{3}$$

DTFT is a continuous periodic function, with a period of $\omega_S = 2\pi f_S = 2\pi/T_S$. This can be easily verified in (3), by changing variable ω with $\omega + \omega_S$. In general, DTFT provides an approximation of CTFT, being applied on infinite input data series. If $f_S \to \infty$, then $T_S \to dt$ and DTFT $\to$ CTFT. But in practical applications it is impossible doing computations on infinite series. Even for finite series with very long sequences of input data, the computations are very difficult.

Discrete Fourier Transform (DFT) is a discrete transform for spectral analysis of finite-domain of discrete-time functions. It is a particular type of DTFT, applied on a time-window with a finite number (N) of input samples of a longer input data sequence. As a result, DFT computes only N different frequency components, denoted X_k, equally spaced into a finite frequency range, corresponding to the first period of DTFT frequency domain, [0, ω_S]. DFT is an approximation of DTFT on this frequency domain, which gets better as the number N increases.

DFT analytical expression is obtained by sampling DTFT in frequency domain, for the finite input sequence, denoted x_n, n = 0 … N-1:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-j2\pi \cdot \frac{k}{N} \cdot n}, \quad k = 0 \ldots N-1 \tag{4}$$

Fast Fourier Transform (FFT) is an efficient class of algorithms to compute DFT and its inverse (it is not another discrete transform). FFT computes more quickly the frequency components of the discrete spectrum than DFT expressed in (4). Considering a finite input sequence of N samples, the computing complexity of DFT is $O(N^2)$, which means that the number of operations needed to obtain the result is proportional with N^2. By contrast, FFT algorithms have $O(N \cdot log_2 N)$. The difference in speed is important, especially for big values of N, when computational time can be reduced by several orders of magnitude (e.g. if $N = 1024 = 2^{10}$, then $O(N^2) = 1048576$, while $O(N \cdot log_2 N) = 10240$).

There are many forms of FFT algorithms, with decimation-in-time and decimation-in-frequency. Radix-2 FFT algorithms need the length of input sequence (N) to be a power of 2. Other FFT algorithms need N to be equal to the product of mutual prime numbers (e.g. 9·7·5 = 315 or 5·16 = 80). In addition, FFT algorithms for real input data are about 60% effort of the same sized complex data FFT.

In general, a radix-n FFT algorithm recursively breaks down the DFT of composite size $N = n \cdot m$ into n smaller DFTs of size m, where n is the radix number. By far the most commonly used radix-2 FFT is the Cooley–Tukey algorithm, which successively breaks a DFT of size N into 2 smaller DFTs of size N/2. In this case, the shape of the data-flow diagram is called butterfly diagram. At its basis, the butterfly diagram takes 2 inputs (x_1, x_0) and generates 2 outputs (y_1, y_0).

A decimation-in-time FFT algorithm with the length of input sequence $N = 2^p$, based on primitive N-th root of unity $\omega = exp(-j \cdot 2\pi / N)$ uses butterfly forms:

$$\begin{cases} y_0 = x_0 + x_1 \cdot \omega^k \\ y_1 = x_0 - x_1 \cdot \omega^k \end{cases}, \tag{5}$$

where k = 0 … N-1 and it depends on the part of the transform to be computed.

3 Aspects of Neural Estimation Using FFNNs

Neural network models are specified by the net topology, node characteristics, and training or learning rules. Static neural network (SNN) models, like FFNN, are characterized by neuron equations without memory, meaning that their outputs are functions only of the current inputs. SNNs implement nonlinear transformations of the form $\mathbf{y} = G(\mathbf{x})$, with input vectors $\mathbf{x} \in \Re^n$, and output vectors $\mathbf{y} \in \Re^m$, where n and m represent the vector dimensions of $\mathbf{x}$ and $\mathbf{y}$, respectively [7]. In this paper, FFNN models are used for neural estimation of signal spectral components.

There are many aspects which must be solved when using FFNNs, regarding the optimal network topology, learning algorithms to deal with local minima, the conditions to achieve good generalization, and efficacy in scaling to larger problems [7]. The most practical concerns are: the network size, time complexity of learning and network ability to generalize.

The network size is important, but in general it is not known for a given problem. If the network is too small, it will not be able to give a good model of the problem. If the network is too big, it will give more solutions which are consistent with training data set, but they are all poor approximations of the problem [2].

The training process consists in finding the correct set of network parameters (weights and biases), which produces the desired mapping for the training data set. Time complexity of learning is directly connected with the complexity of the problem. If the dimension of the input space is very large, then it is unlikely to determine if the correct set of weights and biases can be found in a reasonable amount of time [3]. This is the case when trying to approximate a function like FFT, with a huge dimension of the input space.

One way to reduce the number of network weights is to use local connections of the inputs to the hidden layers, so that individual neurons in the hidden layers to process local regions of the input space [7].

Neural estimation of several spectral components of a signal is a case of approximation problem, with big dimension of the input space. The inputs are N consecutive signal samples of finite input sequence, denoted x_n, n = 0 … N-1, and the outputs represent estimated amplitude of the desired spectral components. Unlike neural approximation of FFT, which tries to find an approximation into the entire input space, spectral estimation takes into account a much smaller input region, for signals with bounded variations in frequency and amplitude.

At every k moment of time, the time-window contains a finite number (N) of time-delayed samples, which form the input sequence of FFNN, as illustrated in Fig.2. The input sequence x_n was denoted according with time moment considered: $x_0 = x(k-N+1)$, …, $x_{N-1} = x(k)$.

It can generally be assumed that any real data will have some degree of uncertainty, generated by systematic or random errors. The estimates take into account the uncertainty in the target data or the uncertainty in the input data. Therefore, it is necessary for any learning system to be able to cope with such uncertainty, producing valid estimates based on real data sets. In this paper it is assumed that there is noise on the input data sets, generated by measurement system.

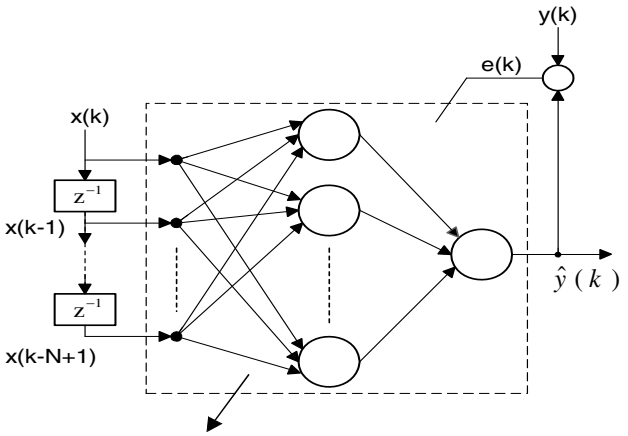

Fig. 2. Spectral component estimation using feed-forward neural networks

For every step k, the estimation error $e(k)$ is computed based on the computed and estimated values of spectral component amplitudes.

The performance goal has to be small enough to assure good estimation performance, but not too small to avoid loss of generalization. The mean squared error *mse* was chosen as performance function:

$$mse = E[e^2(k)] = \frac{\sum_{i=1}^{M} e^2(k)}{M}, \tag{6}$$

where M is the number of vectors in data sets.

4 Experimental Results

For simulations, signal data sets are generated, as sequences of samples from noisy signals with different narrow frequency bandwidth. In spectral analysis, time-windows with $N = 32$ samples are used, as inputs to FFNN and also to FFT algorithm, which computes the desired frequency spectra. The sampling frequency was chosen as f_S = 3200 Hz, resulting a frequency resolution of deltaf = 100 Hz.

Using FFT algorithm, the computed frequency spectra have 32 spectral components, of which only 16 are used, into the frequency domain [0, 1500] Hz. They form the desired frequency spectra in supervised learning process.

Due to large dimension of the input space, $\Re^N$, approximation of the entire FFT function is a very complex task. Hence, a smaller region of interest into the input space has to be selected, taking signals with narrow frequency bandwidth and bounded variations of magnitude. In this case, noisy signals in 2 narrow frequency intervals (3 components) are generated, I_1 = [100, 300], and I_2 = [700, 900] Hz.

The signals are series of sinusoidal functions, with variable number of spectral components, N_{rsc} = 1, 2 or 3, with discrete frequencies randomly selected in intervals. Also, the amplitudes and phases are bounded and randomly selected between the limits: $A_j \in [1, 3]$ and $[-\pi/2, \pi/2]$, respectively. The signal analytical expression is:

$$x(t) = \sum_{j=1}^{N_{rsc}} A_j \cdot \sin(2\pi \cdot f_j t + \varphi_j) \tag{7}$$

The signal parameters are: N_{rsc}, A_j, f_j, and φ_j, $j = 1 \ldots N_{rsc}$. For every parameter set with chosen N_{rsc}, a data sequence is generated. Time horizon includes $N_{tw} = 10$ distinct time-windows. Hence, each data sequence contains $N \cdot N_{tw} = 320$ samples.

In addition, signals can be affected by different types of noise (measurement, conversion, digital processing), which in general is considered as additive noise *z(t)*, resulting a noisy signal: *xz(t) = x(t) + z(t)*. In this paper, additive noise is considered with limited variations and uniform distribution. The signal-noise ratio is *SNR* = 20. An example of data set, with- and without noise, is shown in Fig. 3.

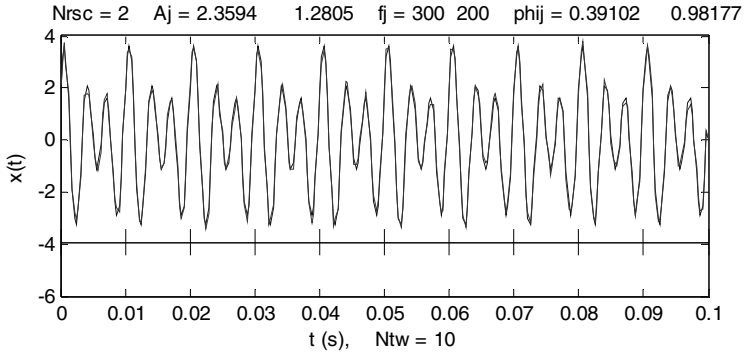

Fig. 3. Data sequence generated with 2 spectral components, f_{j1} = 300 Hz and f_{j2} = 200 Hz

For FFNN training and testing, each input data sequence is organized as a matrix of type *[N, N_{tw}]*, where each column contains *N* samples of time-windows. For example, the samples of first time-window of data sequence, presented in Fig. 3, are illustrated on the left side in Fig. 4. The noisy samples are illustrated with continuous line and the noiseless values are drawn with dotted line. On the right side, spectral components are represented with circle and diamonds marks, respectively.

For every frequency interval, I_1 and I_2, 2 training sets, with N_{rsc} = 1 and 2, respectively, and 3 testing sets, with N_{rsc} = 1, 2 and 3 are generated. The training sets contains a number of N_{lds} = 50 data sequences, with N_{tw} = 10 time-windows each. So, each training set contains a number of learning vectors N_{lv} = 500. Similarly, the testing sets contains a number of N_{tds} = 5 data sequences, with a total number of testing vectors N_{tv} = 50.

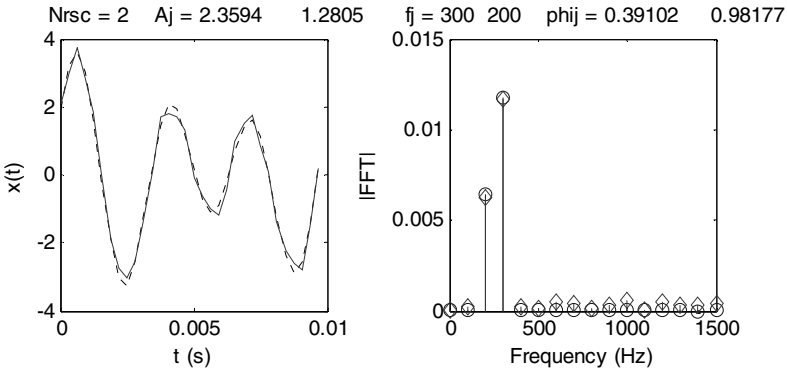

Fig. 4. Samples of first time-window, along with its computed FFT magnitude

All neural networks were trained using Levenberg-Marquardt back-propagation algorithm. During network learning and testing, the same performance criterion was used, the mean squared error (*mse*) of spectral component estimation related to initial noisy frequency spectrum, denoted *msel* and *mset*, respectively.

Two different FFNN architectures were tested. The first type of FFNN architecture is a standard FFNN with one N-dimensional input, one hidden layer with N_{hn} neurons, and one N/2-dimensional output, as illustrated in Fig. 5. The neurons in hidden layer have tansig transfer functions, and output neurons have poslin transfer functions, so that only positive values to be generated for spectral component amplitudes. Two feed-forward neural networks were generated with different number of neurons in hidden layer, N_{hn} = 32, and 64, respectively.

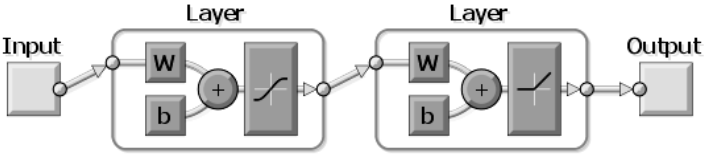

Fig. 5. First type of FFNN architecture, with one N-dimensional input

The second type of FFNN architecture is a butterfly-like FFNN (BFFNN) with two N/2-dimensional inputs, two hidden layers with $N_{hn1} = N_{hn2} = N/2$ neurons, connected separately with the inputs, and one N/2-dimensional output, as illustrated in Fig. 6. The neurons in hidden layers have linear transfer functions, and output neurons have poslin transfer functions.

The N input data samples are separated in two N/2-dimensional inputs, in the same manner as in radix-2 FFT algorithm to form butterfly diagram. The samples with even index are grouped into the first neural input, and the odd index samples form the second neural input. In this way, better performance than FFNN is obtained with less effort (much less weight connections).

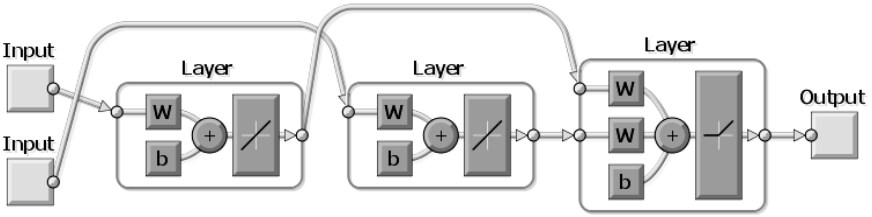

Fig. 6. Butterfly-like FFNN architecture, with two N/2-dimensional inputs

The training and testing results for 3 NNs are represented in Table 1. The parameter *mset1* represents the testing performance for testing data sets with N_{rsc} = 1, while *mset2* is for testing data sets with N_{rsc} = 2. It can be observed that butterfly-like FFNN has much better performance than FFNN, and *mse* has similar values in both frequency ranges.

Table 1. Network Performance for Different Neural Predictors

Freq. Range	Nrsc	FFNN with Nhn = 32			FFNN with Nhn = 64			Butterfly-like FFNN		
		msel $\cdot 10^{-6}$	mset1 $\cdot 10^{-6}$	mset2 $\cdot 10^{-6}$	msel $\cdot 10^{-6}$	mset1 $\cdot 10^{-6}$	mset2 $\cdot 10^{-6}$	msel $\cdot 10^{-6}$	mset1 $\cdot 10^{-6}$	mset2 $\cdot 10^{-6}$
I_1	1	5.89	6.244	3.246	5.93	6.215	5.81	0.331	0.369	7.494
	2	3.29	6.461	3.342	3.02	6.31	3.01	1.38	2.218	0.845
I_2	1	5.99	6.310	3.560	6.12	6.227	9.665	0.331	0.379	7.108
	2	3.81	6.799	3.483	3.28	6.477	3.345	1.32	0.378	0.768

5 Conclusions

In this paper, some aspects of neural estimation of signal spectral components are discussed. Different FFNN architectures, with different data sets and training conditions, are analyzed. A butterfly-like FFNN (BFFNN) was proposed, which has much less weight connections and better performance than FFNN.

References

[1] Barron, A.R.: Approximation and estimation bounds for artificial neural networks. Mach. Learn. 14(1), 115–133 (1994)
[2] Baum, B., Haussler, D.: What size net gives valid generalization? Neural Computation 1, 151–160 (1989)
[3] Blum, A., Rivest, R.L.: Training a 3-node neural network is NP-complete. In: Proc. of Computational Learning Theory Conference (COLT), pp. 9–18 (1988)
[4] Hornik, K., Stinchcombe, K.M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks 2, 359–366 (1989)
[5] Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural Networks 4, 251–257 (1991)

[6] Huang, S.C., Huang, Y.F.: Bounds on the number of hidden neurons in multilayer perceptrons. IEEE Transactions on Neural Networks 2(1), 47–55 (1991)
[7] Hush, D.R., Horne, B.G.: Progress in Supervised Neural Networks. IEEE Signal processing Magazine 10(1), 8–39 (1993), doi:10.1109/79.180705
[8] Ilin, R., Kozma, R., Wetbos, P.J.: Beyond Feedforward Models Trained by Backpropagation: A Practical Training Tool for a More Efficient Universal Approximator. IEEE Trans. on Neural Networks 19(6), 929–937 (2008)
[9] Lippmann, R.P.: An Introduction to Computing with Neural Nets. IEEE ASSP Magazine 4, 4–22 (1987), doi:10.1109/MASSP.1987.1165576
[10] Kainen, P.C., Kurkova, V., Sanguineti, M.: Dependence of Computational Models on Input Dimension: Tractability of Approximation and Optimization Tasks. IEEE Trans. on Information Theory 58(2), 1203–1214 (2012)
[11] Keerthipala, W.W.L.: Low Tah Chong, and Tham Chong Leong Artificial neural network model for analysis of power system harmonics. In: Proc. of IEEE Int. Conf. on Neural Networks, vol. 2, pp. 905–910 (1995)
[12] Leshno, M., Lin, V.Y., Pinkus, A., Schocken, S.: Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural Networks 6, 861–867 (1993)
[13] Leung, H., Haykin, S.: Rational function neural network. Neural Comput. 5, 928–938 (1993)
[14] Nakayama, K., Kaneda, Y., Hirano, A.: A Brain Computer Interface Based on FFT and Multilayer Neural Network. Feature Extraction and Generalization. In: Proc. of Int. Symp. on Intelligent Signal Processing and Comm. Systems (ISPACS 2007), pp. 826–829 (2007)
[15] Onishi, S., Tanaka, A., Hasegawa, H., Kinoshita, K., Kishida, S.: Construction of Individual Identification System using Voice in 3-lyered Neural Networks. In: Proc. of Int. Symp. on Intelligent Signal Processing and Comm. Systems (ISPACS 2009), pp. 635–637 (2009)
[16] Park, J., Sandberg, I.W.: Universal approximation using radial-basis-function networks. Neural Comput. 3, 246–257 (1991)
[17] Perko, M., Fajfar, I., Tuma, T., Puhan, J.: Fast Fourier Transform Computation using a Digital CNN Simulator. In: 5th IEEE Int. Workshop on Cellular Neural Networks Abd Their Applications, pp. 230–235 (1998)
[18] Segee, B.E.: Using spectral techniques for improved performance in ANN. In: Proc. of IEEE Int. Conf. on Neural Networks, pp. 500–505 (1993), doi:10.1109/ICNN.1993.298608
[19] Tamayo, O.M., Pineda, J.G.: Filtering and spectral processing of 1-D signals using cellular neural networks. In: Int. Symp. on Circuits and Systems (ISCAS 1996), vol. 3, pp. 76–79 (1996)
[20] Wu, X., He, W., Zhang, Z., Deng, J., Li, B.: The harmonics analysis of power system based on Artificial Neural Network. In: World Automation Congress (WAC 2008), pp. 1–4 (2008)