Evolving Neural Network CMAC and its Applications

Oleg Rudenko, Oleksandr Bessonov and Oleksandr Dorokhov
Kharkiv National University of Economics, Nauka Ave 9a, 61166 Kharkiv, Ukraine
E-mail: aleks.dorokhov@meta.ua

Keywords: neural network, training, nonlinear object, identification, adaptive control

Received: April 21, 2018

The conventional neural network (NN) CMAC (Cerebellar Model Articulation Controller) can be applied in many real-world applications thanks to its high learning speed and good generalization capability. In this paper, it is proposed to utilize a neuro-evolutional approach to adjust CMAC parameters and construct mathematical models of nonlinear objects in the presence of the Gaussian noise. The general structure of the evolving NN CMAC (ECMAC) is considered. The paper demonstrates that the evolving NN CMAC can be used effectively for the identification of nonlinear dynamical systems. The simulation of the proposed approach for various nonlinear objects is performed. The results proved the effectiveness of the developed methods.

Povzetek: Razvit je postopek za evolucijsko iskanje najbolj prilagojene CMAC (Cerebellar Model Articulation Controller) nevronske mreže za probleme z Gaussovim šumom.

1 Introduction

Using a mathematical model of the cerebellar cortex developed by D. Marr [1] in 1975 J. Albus proposed a model describing the motion control processes that occur in the cerebellum, which was subsequently implemented in the neural network controller for controlling the robot - arm, which he called CMAC - Cerebellar Model Articulation Controller [2, 3]. Ease of implementation and a good network of approximating properties have ensured its wide usage not only in the tasks of controlling the robotic arm in real time, but also to solve many other practical problems [4-14].

However, it should be noted that in designing a network CMAC a number of difficulties in the selection of parameters such as the number of levels and the quantization levels, the shape of the receptive field, the type of applied information hashing algorithm and training. These parameters have a significant impact on the accuracy and speed of CMAC network, and therefore, the determination of the optimal values of these parameters is an important practical problem. In this article, for eliminating the drawbacks of traditional methods of synthesis and functioning ANN CMAC we provide the use of a new class of networks - evolving ANN (EANN) in which, in addition to traditional learning it is used another fundamental form of adaptation - evolution, realized by applying the evolutionary computation [15-18].

The use of two forms of adaptation in EANN - evolution, and training, allowing to change the network structure, its parameters and learning algorithms without external intervention, make the network data most suitable for work in non-stationary conditions and uncertainty about the properties of the object under study and the conditions of its functioning.

The main advantage of using evolutionary algorithms (EA) as learning algorithms is that many ANN parameters can be encoded in the genome and determined in parallel. Moreover, unlike most optimization algorithms designed to solve a problem, EA operate with a multitude of solutions - the population, which allows reaching a global minimum, without getting stuck in the local ones. In this case, information about each individual of the population is encoded in a chromosome (genotype), and the solution (phenotype) is obtained after evolution (selection, crossing, mutation) by decoding.

Among EAs that are stochastic and include evolutionary programming, evolutionary strategies, genetic algorithms, genetic programming, in particular, programming with gene expression, genetic algorithms (GA) are the most common [19,20]. GA abstract the fundamental processes of Darwinian evolution: natural selection and genetic changes due to recombination and mutation.

2 Neural network CMAC

The modification of the network proposed by Albus is shown in Figure 1. The network consists of the input, hidden and output layers, labeled L1, L2, L3, respectively, and uses two basic conversions:

$$S: X \Rightarrow A, \quad (1)$$

$$P: A \Rightarrow y, \quad (2)$$

where X - N-dimensional space of continuous input signals; A - n-dimensional space associations; y - a one-dimensional output.

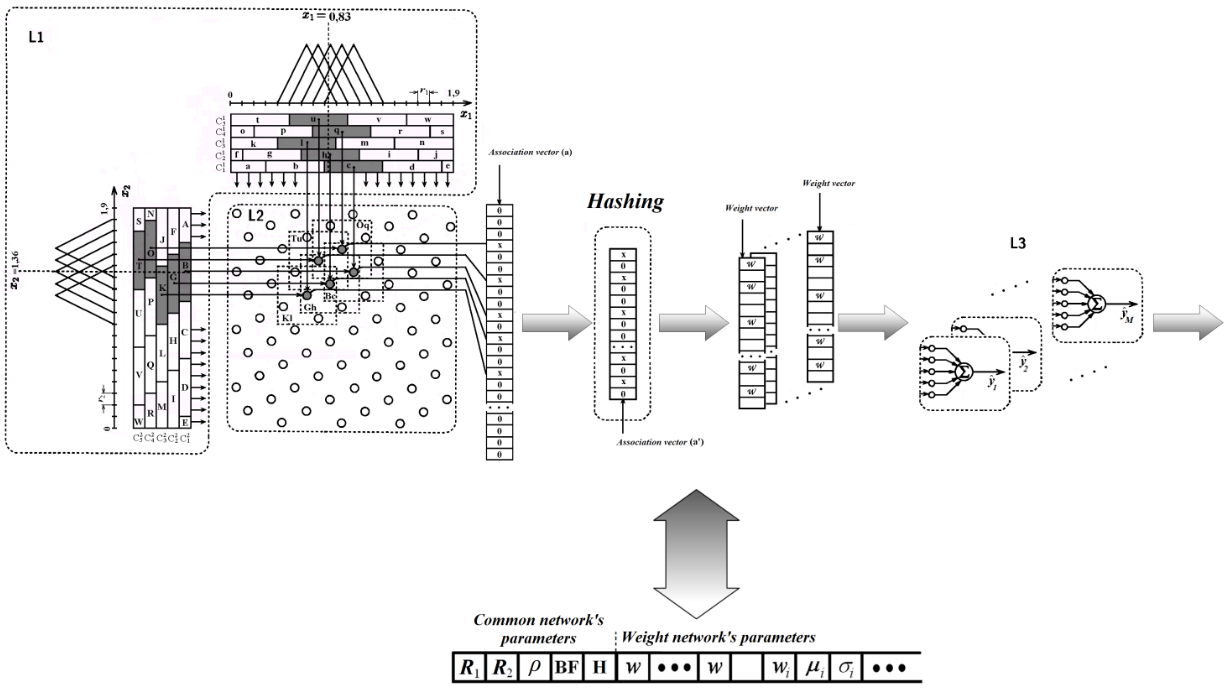
L1
L2
L3
Hashing
Association vector (a)
Association vector (a')
Weight vector
Common network's parameters
Weight network's parameters
R_1 R_2 ρ BF H w ••• w w_i μ_i σ_i •••

Figure 1: Albus network.

Converting $S \Longrightarrow A$, in turn, consists of two transformations:

$$X \Longrightarrow M \tag{3}$$

$$M \Longrightarrow A, \tag{4}$$

where M - the space of binary variables.

The principle of the network operation as an associative memory is as follows. Approximated function *y = f (x)* is given to a limited number of points (argument values) **x** constituting N-dimensional space of the input signals. This space is divided into subspaces M formed the input signals **x**(*i*) ($i=\overline{1,M}$).

A number of subspaces M impacts the accuracy of the network and number of utilized memory cells. Therefore, on the one hand, side it should be big enough to ensure good approximation capabilities of the network and on the other hand side, it should be not too big to save some memory. In constructing the cerebellum model Albus proceeded from the fact that the appearance of the excitation signal activates its a certain area of the cerebellum, or receptive field, characterized by a parameter ρ.

Therefore, storage of values of **y**(*i*) (network output signal) corresponding to **x**(*i*) ($i=\overline{1,M}$), used *ρ* memory cells, the number of which is constant for all vectors of the input signals on the network. At receipt of the input signal ***x***(*i*) a signal ***y***(*i*) appears at network output, which is the sum of *ρ* addressable cells content.

Associative CMAC properties manifest themselves in the form of used addressing, which is based on a special coding input information and called hash coding or hashing [21-23].

3 Encoding information in CMAC

Information coding in the network means that to each N-dimensional input vector x(i) an n-dimensional association vector a(i), is assigned and stored in virtual memory.

Elements of **a**(*i*) can take the values from the interval [0, 1] (in the papers cited above it is assumed that these elements take the values 0 or 1). Thus only ρ << n elements of the vector have non-zero values, i.e. only ρ memory elements are active.

A continuous plurality of input signals by sampling (at the level of quantization) is converted into discrete. Thus to represent the *i*-th input signal components R_i quantization levels used with the appropriate quantization step r_i ($i=\overline{1,N}$). It should be noted that the accuracy of the system identification depends substantially on the size of the quantization step, and loss of stability is possible in digital automated control systems with an incorrect choice of this parameter.

Each stage is characterized by a corresponding association matrix A_i $\left(i=\overline{1,\rho}\right)$, only one element of which is different from zero.

Construction associations vector as follows. For a given total number of input signals association matrix A_i of each quantization stage ($i=\overline{1,\rho}$) are formed. The columns of these matrixes form association vectors $\boldsymbol{a}_i$ ($i=\overline{1,\rho}$).

The dimension of these vectors, *n*, equal to the sum of all elements of the matrices A_i ($i=\overline{1,\rho}$) and can be calculated by the formula:

$$n = \left]\rho\left(\frac{R-1}{\rho}+1\right)^N\right[, \quad (5)$$

where R - the number of used levels for quantizing input signals; N - the dimension of the input vector; $]\bullet[$ - means rounded to the nearest whole number.

Since all ρ matrices A_i ($i=\overline{1,\rho}$) have only one non-zero element, from the n components of the vector $\boldsymbol{a}(i)$ only ρ are non-zero.

The quantization region is arranged in such a way, that any of them relating to the adjacent stages have not more than (ρ - 1)-th connection. This corresponds to a restriction on the maximum total number of cells equal to (ρ - 1) used for storing two different vectors of the input signals in which their recognition is still possible.

4 Selecting the basic functions of neurons

Selecting the basic functions of neurons in L1 layer significantly affects the approximating properties of CMAC network.

As already noted, the traditional CMAC performs piecewise constant approximation, that is a consequence of the usage of neurons with a rectangular activation function.

When choosing rectangular basis functions computational cost will be minimal. Also, CMAC networks widely use B-splines as basis functions.

B-splines undoubted advantage is the possibility of recurrent calculating in both the splines in accordance with the formula [24-27]:

$$B_{n,j}(x) = \left[\frac{x-\lambda_{j-n}}{\lambda_{j-1}-\lambda_{j-n}}\right]B_{n-1,j-1}(x) + \left[\frac{\lambda_j - x}{\lambda_j - \lambda_{j-n+1}}\right]B_{n-1,j}(x) \quad (6)$$

and their derivatives δ -order:

$$^{(\delta)}B_{n,j}(x) = \frac{(n-1)}{(n-\delta-1)}\left[\frac{x-\lambda_{j-n}}{\lambda_{j-1}-\lambda_{j-n}}\right]^{(\delta)} B_{n-1,j-1}(x) + $$
$$+\frac{(n-1)}{(n-\delta-1)}\left[\frac{\lambda_j - x}{\lambda_j-\lambda_{j-n+1}}\right]^{(\delta)} B_{n-1,j}(x) \quad (7)$$

Here:

$$B_{0,j}(x) = \begin{cases} 1, \text{if } \ x \in [\lambda_{j-1}, \lambda_j); \\ 0, \text{otherwise}; \end{cases}$$

$$^{(0)}B_{0,j}(x) = \begin{cases} 1, \text{if } \ x \in [\lambda_{j-1}, \lambda_j); \\ 0, \text{otherwise}; \end{cases}$$

$$^{(\delta)}B_{n,j}(x) = \left[\frac{^{(\delta-1)}B_{n-1,j-1}(x)}{\lambda_{j-1}-\lambda_{j-n}}\right] - \left[\frac{^{(\delta-1)}B_{n-1,j}(x)}{\lambda_j - \lambda_{j-n+1}}\right] \quad (8)$$

where λ_j - j-th spline's node (center of the quantization field).

Thus, after determining the active slot $(\lambda_{j-1}, \lambda_j]$ for the zero order B-spline, these expressions can be used to obtain the values of all nonzero B-spline of higher order and, if appropriate, their derivatives.

Note that traditional CMAC uses zero order B-spline. Selection of the first order B-spline leads to the triangular membership function, and selection of the fourth-order B-spline leads to membership function similar to the Gaussian.

The CMAC network also uses Gaussian activation function of the form:

$$\Phi_i(x_j) = \exp\left\{-\frac{(\boldsymbol{x_j} - \boldsymbol{\mu}_i)^2}{\sigma^2}\right\} \quad (9)$$

As a basis one can use trigonometric functions, for example, cosine:

$$\Phi_i(x_j) = \begin{cases} \cos\left(\frac{\pi}{\rho r_j}(x_j - \lambda_i)\right) \text{if } \ x_j \in (\lambda_i - \frac{\rho r_j}{2}, \lambda_i + \frac{\rho r_j}{2}]; \\ 0 \qquad \text{otherwise}, \end{cases} \quad (10)$$

where λ_i - i-th center of the quantization field; r_j - quantization step of j-th component of the input signal.

However, it should be noted that although the most commonly used Gaussian membership functions also allow a very simple calculation of derivatives and have the property of a local activation, it is difficult to allocate clearly enough their activation boundary, which is often important for the implementation of the network that used, for example, scaling basis functions.

In order to eliminate this disadvantage, one can use a modified Gaussian function that has the following form:

$$\Phi_i(x) = \begin{cases} \exp\left\{-\frac{(\lambda_2 - \lambda_1)^2/4}{(x-\lambda_1)(\lambda_2 - x)}\right\} \text{if } \ x \in (\lambda_1, \lambda_2); \\ 0 \qquad \text{otherwise.} \end{cases} \quad (11)$$

As seen from the expression (11), the function is strictly defined in the range (λ_1, λ_2), which simplifies the process of scaling basis functions when the network parameters such as R and ρ are changing.

5 Network training

Defining the network parameters, i.e. in the general case defining a vector $\theta(k)$, that includes all network parameters (weights, parameters of basis functions, etc.) is accomplished by training with the teacher.

The training criterion can be presented as follows:

$$F[e(k)] = \sum_{i=1}^{k} \rho(e(i)), \quad (12)$$

where $\rho(e(i))$ - some loss function.

Gradient network training algorithm has the following form:

$$\hat{\theta}(k)=\hat{\theta}(k-1)+\gamma(k)\frac{\partial F(e(k))}{\partial\theta_j}, \quad (13)$$

or

$$\hat{\theta}(k)=\hat{\theta}(k-1)+\gamma(k)\rho'\left(e(k)\right)\frac{\partial e(k)}{\partial\theta_j} \quad (14)$$

where $\gamma>0$ – parameter that affects the training speed and which can be selected differently for different network parameters.

Training of traditional CMAC that uses $\rho\left(e(i)\right)=0,5e^2(i)$ and rectangular basis functions, occurs on each step after the presentation of training pairs $\{\boldsymbol{x}(k),y(k)\}$, where $y(k)$ – function's value, that corresponds $\boldsymbol{x}(k)$, and consists in the correction of only those of its ρ weights that correspond to the single components of the association vector for a given vector $\boldsymbol{x}(k)$.

In this case, the training algorithm for all i, j, for which $a_i(k)=a_j(k)=1$, is the following:

$$w_j(k+1)=w_j(k)+\gamma\left(y(k)-\frac{1}{\rho}\sum_{i=1}^{n}w_i(k)\right), \quad (15)$$

where $\gamma\in(0,1]$ – parameter that affects the speed of training.

When membership functions with a form other than rectangular are used, this algorithm can be written as follows:

$$w(k+1)=w(k)+\gamma(k)\left(\frac{y(k)-a^T(k)\Phi(x)w(k)}{\left\|\Phi(x)a(k)\right\|^2}\Phi(x)a(k)\right), \quad (16)$$

where $\gamma(k)$ - in general case a variable parameter.

Multistep network training algorithms were considered in [7,8].

6 Evolving ANN CMAC

During switching from ANN to EANN for all types of networks the common evolutionary procedure (initialization population, an estimation of the population, selection, cross-breeding, mutations) is used. Differences are only in the method of encoding the structure and parameters of a particular form of ANN in the chromosome.

At the beginning of EA functioning, a population P_0 that consisting of N individuals (ANN): $P_0=\{H_1,H_2,...,H_N\}$ is randomly initialized. The proper choice of the N's value is very important as this parameter significantly affects the speed of the algorithm and its selection is critical for real-time systems.

Each individual in the population at the same time gets its own unique description, encoded in the chromosome $H_j=\{h_{1j},h_{2j},...h_{Lj}\}$, which consists of L gene, wherein $h_{ij}\in[w_{\min}\,w_{\max}]$ - *i*-th value of *j*-gene chromosome ($w_{\min}$ - the minimum and $w_{\max}$ - maximum allowable values, respectively).

Figure 1 shows an example ECMAC chromosome's format and the correspondence between genes and network parameters stored in the chromosome. It should be noted that chromosome length depends on the dimensionality of the problem and the maximum amount of memory.

As seen from the drawing, it consists of a chromosome gene in which information about corresponding network parameters is stored. At the beginning of the chromosome, there are genes that contain information about the parameters of the noise and they are active only in case of the noisy measurements. Next gene's block encodes the number of levels and the quantization steps, the shape of the receptive field of neurons and type of algorithm that is used for hashing information.

Due to the large amount of the BF that can be used in CMAC, there is a special gene in its chromosome BF, that is responsible for coding the type of the used functions. There is also a gene H in the chromosome, that encodes a type of the hashing algorithm (If its value is set to 0 then hashing is not used).

Then, in the chromosome, there is a group of genes encoding weighting parameters directly relevant to the associative neurons. During the initialization phase, initial values are assigned to all these parameters by using a random number generator.

Since during evolution mutation may occur in the parameters affecting the amount of used associative neurons, the length of the chromosome can vary. The use of variable length chromosomes occur individuals with specific genetic code segments (introns) which are not used for coding characteristics [28-29].

Typically, introns are used in the EA:

- as noncoding bits that are uniformly added to the genetic code (in this case, introns only fill the space between the active genes of the chromosome);

- as the nonfunctional parts of the genetic code, i.e., parts of the decision which do not actually do anything, thus not affect the fitness of the chromosome (this usually occurs in the genetic programming and in the chromosomes, which are subject to the cycle of development after birth);

-as posterior useless part of the chromosome, which do not participate in the calculation of its fitness (usually it manifests itself in some types of competitive-trained neural networks, in which only neurons-winners in contrast to other neurons that are a posteriori useless affect network performance results).

Introns appear in other types of neural networks, where they are called potentially useful waste.

The control of introns amount in the population carried out by:

- the use of special operators, that alter the length of the chromosome and add or remove introns (experimental results show that some number of introns improves overall properties EA);

- the use of the selection operator: depending on the number of introns, the value of individual fitness function will increase or decrease.

Once the initial population is formed, the fitness of each individual part in it evaluates by some defined fitness function.

Conventionally, as such a function the quadratic one is used:

$$F(x) = \frac{1}{M}\sum_{i=1}^{M}\left(y^{*}(x_i) - \hat{y}(x_i)\right)^2, \tag{17}$$

where $y^{*}(k)$ - the desired network response; $\hat{y}(k)$ - real output signal; M – sample size.

The next step is the selection of individuals, the chromosomes of which are involved in the formation of the new generation, and subsequent hybridization.

The task of crossing operator (crossover) is the transfer of genetic information from the parent individuals to their offspring.

After completion of the operator’s work, any gene of any individual in the new population may mutate, i.e. change its value.

Since chromosome uses hybrid coding, during the mutations various operations must be performed for different encoding methods.

For example, in the case of the gene that is responsible for neuron’s activation and uses binary encoding, inverse mutation should be used.

For coding the BF and weighting parameters, that uses real values, different types of mutations may be used.

Thus ECMAC algorithm can be represented as follows:

- create an initial population (initialization of each individual chromosome, estimation of the initial population);

- the stages of evolution - the construction of a new generation (selection of candidates for mating /breeding, hybridization, i.e. causing by each pair of selected candidates some new individuals, mutation, evaluation of the new population);

- check the completion criterion, if not satisfied - go back to the stages of evolution.

7 First Modeling Experiment

It is considered the problem of nonlinear dynamic object identification that is described by the equation:

$$y(k) = \max\left[e^{-10u^2(k)};\ e^{-50y^2(k-1)};\right.$$

$$\left.;\ 1.25e^{-5\left(u^2(k)+y^2(k-1)\right)}\right] + \xi(k), \tag{18}$$

where $u(k)$ - the input signal, that is a stationary random sequence with the random uniform distribution in the interval [-1, 1].

During the study of this object, the population of CMAC networks consisting of 100 individuals was used.

The population evolved during 500 epochs. All configurable network parameters (including R and ρ) were determined by EA.

It should be noted that the values of R and ρ determine the amount of memory that is used for storing network parameters and significantly affect the accuracy of the approximation.

Graphs of the network-winner fitness function and the required amount of memory to store its parameters are shown in Figures 2 and 3.

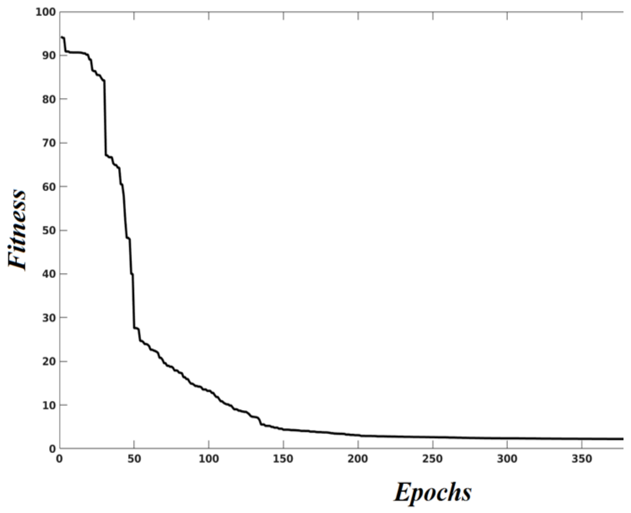

Figure 2: The network-winner fitness function.

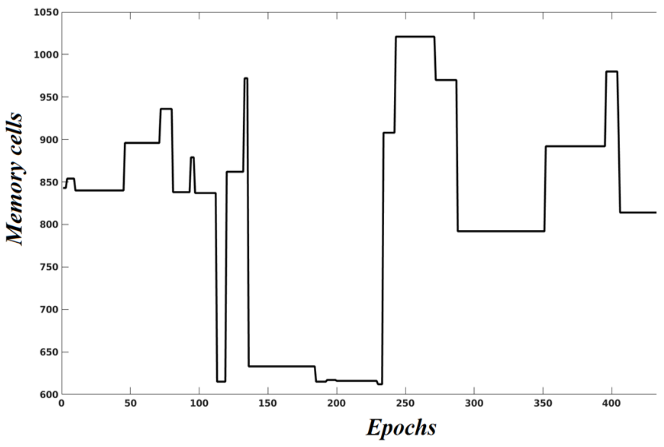

Figure 3: Required amount of memory.

Results of the stationary object (18) identification are shown in Figures 4 and 5.

So, Figure 4 shows the surface itself, according to the equation described, and Figure 5 - the surface recovered by ECMAC with ξ (k) = 0.

The winning network comprises 814 weighting parameters with R = 186 and ρ = 95, and rectangular activation function was chosen.

Figure 6 shows the results of the object (18) identification in the presence of the random noise ξ (k) that is normally distributed in the interval [-0.3, 0.3].

In this case, the winning network used cosine activation functions (10).

$y(k)$

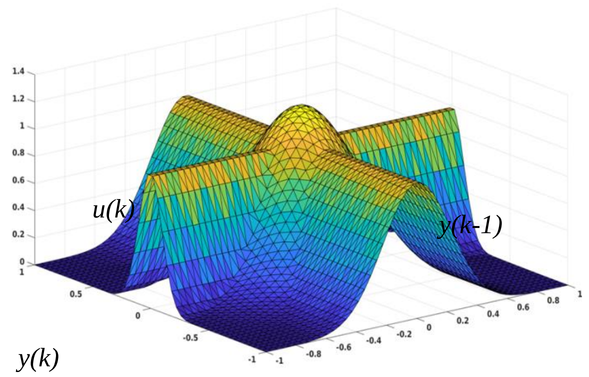

Figure 4: The surface itself, according to the equation described.

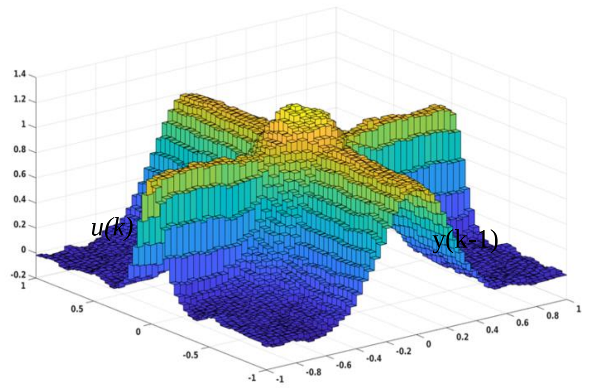

Figure 5: The surface recovered by ECMAC.

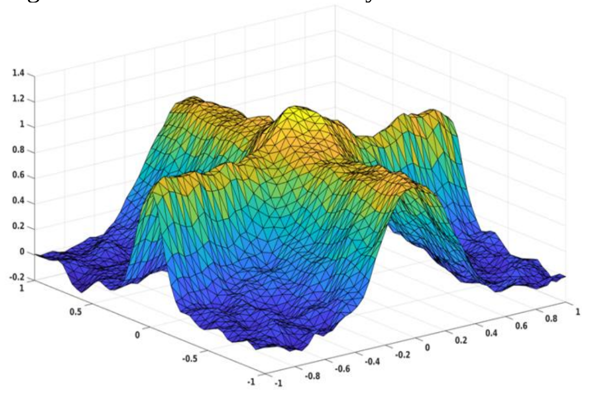

Figure 6: The results of the object (13) identification.

8 Second Modeling Experiment

We solved the problem of the identification of a dynamic object described by the equation:

$$y(k)=\frac{y(k-1)y(k-2)y(k-3)y(k-4)(y(k-3)-1)}{1+u(k)^2+y(k-2)^2}+$$

$$+\frac{u(k)}{1+u(k)^2+y(k-2)^2} \qquad (19)$$

To solve this problem, we used a population of evolving CMAC networks comprising 250 individuals.

After reaching the required accuracy of identification, to assess the quality of the resulting model, to the object and the winner network the same control actions were given:

a) $u(k)=\sin(\pi k/200)+\cos(\pi k/400)$;

b) $u(k)=-1.5+0.001k$.

The experimental results for the cases a) and b) are shown in Figure 7 and 8.

The solid line shows the output signal of the object and the dashed - neural network model output.

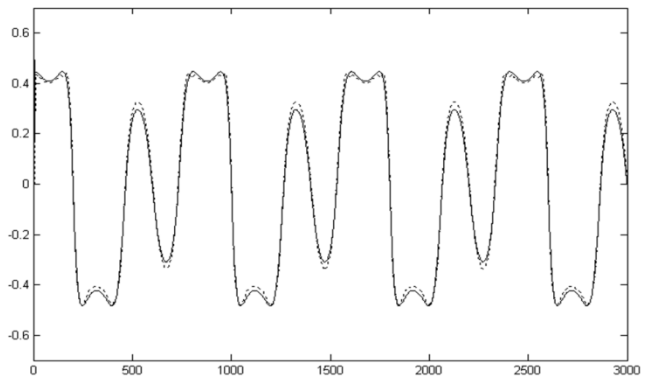

Figure 7: The experimental results for the a)-case.

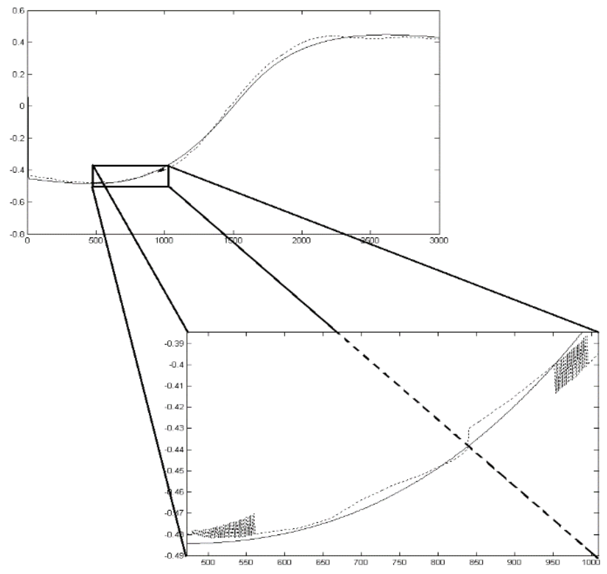

Figure 8: The experimental results for the b)-case.

As can be seen from the simulation results, the accuracy of identification of a multidimensional object (19) via evolving networks CMAC is sufficiently high and error – is inessential.

In Figure 8 there are some minor oscillations that appeared due to rounding off calculations.

9 Third Modeling Experiment

It is considered the problem of controlling a multidimensional nonlinear dynamic object described by the following equations:

$$y_1(k) = \frac{y_1(k-1)}{1+y_2(k-1)^2} + u_1(k-1);$$
$$y_2(k) = \frac{y_1(k-1)y_2(k-1)}{1+y_2(k-1)^2} + u_2(k-1). \quad (20)$$

Uncorrelated random sequences with a uniform distribution law in the interval [–1, 1] were used as inputs of the network during the training.

Training was carried out on the basis of the presentation of a network of 10000 training pairs and the following parameters of the CMAC neural network: activation functions - trigonometric; *R*=100; *ρ*=40.

The required memory capacity for such parameters is 5818 memory cells. The size of the population was 300 individuals.

The required values of the output signals were set in the following way:

$$y_1^*(k) = \sin(\pi k/100);$$
$$y_2^*(k) = \begin{cases} 0.5, k = \overline{1,500}; \\ \sin(\pi k/300) + \sin(\pi k/500), k = \overline{501,1000}. \end{cases} \quad (21)$$

The results of the neural control are shown in Figures 9 and 10.

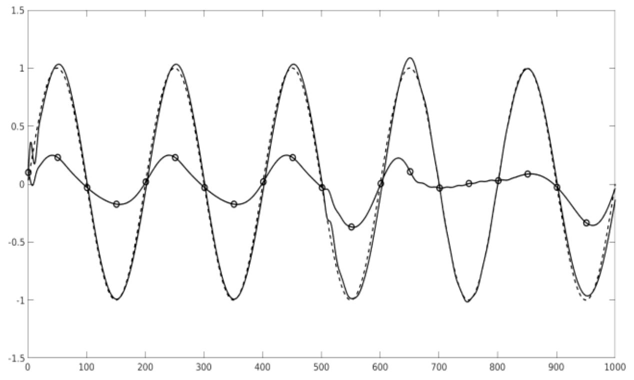

Figure 9: The experimental results for the $y_1(k)$

In all these figures, the dotted line shows the required output signal $y_i^*(k)$, solid line – real $\hat{y}_i(k)$, and the line with the circles - the corresponding change of the control signal $u_i(k)$ $(i=1,2)$.

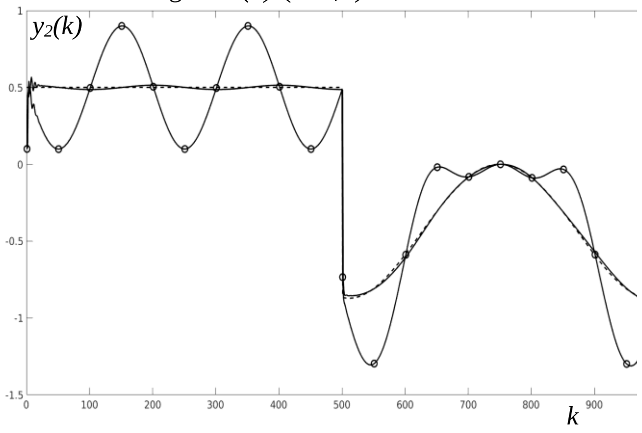

Figure 10: The experimental results for the $y_2(k)$.

10 Conclusions

The results showed that the evolving neural network CMAC is quite effective and convenient in solving practical problems (identification of nonlinear objects, control, etc).

Substantial savings of required memory in combination with evolutionary training algorithms make it particularly attractive for implementation in real complex dynamic object control systems in the presence of noisy measurements.

An additional advantage of the evolutionary approach to CMAC network training is the solution of the problem of choice the receptive field form of the associative neurons that is affecting the method and the accuracy of the approximation of the studied functions.

In the case ECMAC this problem is solved automatically.

References

[1] Marr, D. (1969). Theory of Cerebellar Cortex. *Journal Physiology*, Vol. 202, 437-470.

[2] Albus, J. (1975). A new approach to manipulator control: the cerebellar model articulation controller (CMAC). *J. Dynamic Systems, Measurement, and Control*, Vol. 97, №3, 220-227. https://doi.org/10.1115/1.3426922

[3] Albus, J. (1975). Data storage in cerebellar model articulation controller (CMAC). *J. Dynamic Systems, Measurement and Control*, Vol. 97, №3, 228-233. https://doi.org/10.1115/1.3426923

[4] Miller, W., Glanz, F., Kraft, L. (1990). CMAC: An associative neural network alternative to backpropagation. *Proc. of the IEEE*, Vol. 78, №10, 1561–1567. https://doi.org/10.1109/5.58338

[5] Miller, T., Hewes, R., Glanz, F., Kraft, L. (1990). Real-time dynamic control of industrial manipulator using a neural-network-based learning controller. *IEEE Trans. Robot. Automat.*, Vol. 6, 1-9. https://doi.org/10.1109/70.88112

[6] Iigumi, Y. (1996). Hierarchical image coding via cerebral model arithmetic computers. I*EEE Trans. Image Processing*, Vol. 5, 1393-1401. https://doi.org/10.1109/83.536888

[7] Avdeyan, E., Hormel, M. (1991). The increase of the rate of convergence of the learning process in a special system of associative memory. *Automation and telemechanics*, Vol. 6, 1-9.

[8] Rudenko, O., Bessonov, A. (2005). CMAC Neural Network and Its Use in Problems of Identification and Control of Nonlinear Dynamic Objects. *Cybernetics and Systems Analysis,* Vol.41, Issue 5, 647–658. https://doi.org/10.1007/s10559-006-0002-x

[9] Li, H.-Y., Yeh, R.-G., Lin, Y.-C., Lin, L.-Y., Zhao, J., Rudas, I. (2016). Medical Sample Classifier Design Using Fuzzy Cerebellar Model Neural Networks. *Acta polytechnica Hungarica*, Vol. 13, №6, 7-24.

https://doi.org/10.12700/aph.13.6.2016.6.1

[10] Shafik, A., Abdelhameed, M., Kassem ,A. (2014). CMAC Based Hybrid Control System for Solving Electrohydraulic System Nonlinearities. *Int. Journal of Manufacturing, Materials, and Mechanical Engineering,* Vol.4(2), 20-26. https://doi.org/10.4018/ijmmme.2014040104

[11] Lee, C.-H., Chang, F.-Y. (2014). An Efficient Interval Type-2 Fuzzy CMAC for Chaos Time-Series Prediction and Synchronization. *IEEE Transactions on Cybernetics*, Vol. 44, №3, 329-341. https://doi.org/10.1109/tcyb.2013.2254113

[12] Chung, C.-C., Lin, C.-C. (2015). Fuzzy Brain Emotional Cerebellar Model Articulation Control System Design for Multi-Input Multi-Output Nonlinear. *Acta Polytechnica Hungarica*, Vol. 12, № 4. 39-58. https://doi.org/10.12700/aph.12.4.2015.4.3

[13] Dorokhov, O., Chernov, V., Dorokhova, L., Streimkis, J. (2018). Multi-criteria choice of alternatives under fuzzy information, *Transformations in Business and Economics*, Vol. 2, 95-106.

[14] Malyarets L., Dorokhov, O., Dorokhova L. (2018). Method of constructing the fuzzy regression model of bank competitiveness. Journal of Central Banking Theory and Practice, Vol. 7, №2, 139–164. https://doi.org/10.2478/jcbtp-2018-0016

[15] Xu, S., Jing, Y. (2016). Research and Application of the Pellet Grate Thickness Control System Base on Improved CMAC Neural Network Algorithm. *Journal of Residuals Science & Technology,* Vol. 13, № 6, 1501-1509.

[16] Floreano, D., Mattiussi, C. (2008). *Bio-Inspired Artificial Intelligence Theories, Methods, and Technologies.* The MIT Press Cambridge, Massachusetts-London, England.

[17] Andries, P. (1997). *Engelbrecht Computational Intelligence An Introduction.* John Wiley & Sons.

[18] Yao, X. (1993). A Review of Evolutionary Artificial Neural Networks. *Int. J. Intell. Syst.*, №8 (4), 539-567.

[19] Yao, X. (1999). Evolving Artificial Neural Networks. *Proc. of the IEEE*, Vol. 87, №9,1423-1447. https://doi.org/10.1109/5.784219

[20] Holland, J. (1975). *Adaptation in Natural and Artificial Systems. An Introductory Analysis With Application to Biology, Control and Artificial Intelligence.* University of Michigan.

[21] Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, MA.

[22] Knuth, D. (1973). *Sorting and Searching, in the Art of Computer Programming*. Menlo Park, Calif., Addison Wesley.

[23] Wang, Z.-Q., Schiano, J., Ginsberg, M. (1996). Hash-Coding in CMAC Neural Networks. *IEEE Int. Conf. on Neural Networks*, Vol. 3, 1698-1703. https://doi.org/10.1109/icnn.1996.549156

[24] Rudenko, O., Bessonov, O.(2004). Hashing information in a neural network CMAC. *Control Systems and Machines,* №5, 67-73.

[25] Chiang, C.-T., Lin, C.-S. (1996). CMAC with General Basis Functions. *Neural Networks*, Vol. 7, №7, 1199-1211.

[26] Lane, S., Handelman, D., Gelfand, J. (1992). Theory and Development of Higher-Order CMAC Neural Networks. *IEEE Control Systems*, Vol. 12, № 2, 23-30. https://doi.org/10.1109/37.126849

[27] Rudenko, O., Bessonov, O. (2004). On the Choice of Basis Functions in a Neural Network CMAC. *Problems of Control and Informatics*, № 2, 143–154.

[28] Wu, A.(1995). Empirical Studies of the Genetic Algorithm with Non-Coding Segments. *Evolutionary Computation*, Vol. 3(2), 121-147.

[29] Castellano, J. (2001). *Scrapping or Recycling: the Role of Chromosome Length-Altering Operators in Genetic Algorithms.* GeNeura Group, Department of Architecture and Computer Technology, University of Granada. (2001).