ACTIVE TRAINING ON THE CMAC NONLINEAR ADAPTIVE SYSTEM

*Luis Weruaga, *Juan Morales and *Rafael Verdú*

Austrian Academy of Sciences, Donau-City Strasse 1, A-1220 Wien, Austria.
luis.weruaga@oeaw.ac.at
*Cartagena University of Technology, Campus Muralla del Mar s/n, E-30202 Cartagena, Spain.
rafael.verdu@upct.es

ABSTRACT

The CMAC neural network presents a rigid architecture for learning and generalizing simultaneously, a limitation stressed with sparse or non-dense training datasets, and hardly solved by the current training algorithms. This paper proposes a novel training algorithm that overcomes the mentioned tradeoff. The training mechanism is based on the minimization of the energy of curvature of the output, solution based on the active deformable model theory. This leads to a cell-interaction-based internal update that preserves the efficient hashed indexing and the original learning capabilities, and delivers a higher generalization degree than the a-priori embedded in the CMAC architecture. The theoretical analysis is supported with comparative results on the inverse kinematics of a robotic arm.

1. INTRODUCTION

The Cerebellar Model Articulation Controller (CMAC) is a non-linear adaptive system proposed by Albus in the mid-1970s [1] with in-built simple computation, local generalization and fast learning properties. In spite of its elegant features, because of its rigid structure and the fact that a CMAC cannot represent an arbitrary function, it has attracted modest interest. The theoretical attempts done for defining the identification capabilities [2] [3] have not provided conclusive results. Moreover, the straightforward and simple implementation of the gradient-based training (Albus rule, LMS) on the CMAC still represents the conductive line of the last works on training algorithmic [4].

In most of the feed-forward neural networks the ability to generalize, that is, to deliver similar outputs to close inputs, is provided by the structure and not by the algorithm: in the multi-layer-perceptron (MLP) that ability lies on the sigmoidal function and in the number of neurons, the gradient-based training or the Expectation Maximization algorithm for radial basis functions (RBF), when pursuing generalization, presents severe local minima, etc. In the case of the CMAC, the hypercube size determines the generalization degree. However, since the LMS on the CMAC architecture spreads the training error among the excited local functions, this algorithm does not guarantee smoothness or narrow bandwidth in the output, and the meaning of generalization is thus misunderstood. This statement is especially true for sparse or non-dense training sets, for which LMS training produces undesired overfitting. This fact leads to a rigid tradeoff between generalization and learning performance in the whole input domain.

This paper proposes a novel CMAC training algorithm that overcomes the mentioned generalization-learning tradeoff. By following closely the previous definition of generalization, a curvature-minimization-based algorithm is implemented on the CMAC architecture. This gives rise to an iterative mechanism that propagates the external knowledge to the whole input domain by means of the interaction between the CMAC cells, while preserving the efficient hashed indexing. Given that the training mechanism is inspired from the theory of active deformable models [5], the new learning structure has been named called Active CMAC.

2. PREMISES

Let $\mathscr{X} = \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots \tilde{\mathbf{x}}_L\}$ and $\mathscr{Z} = \{z_1, z_2, \ldots z_L\}$ be respectively the input and desired output training datasets, where the elements in $\mathscr{X}$ are d-dimensional real-valued, $\tilde{\mathbf{x}}_i \in \mathbb{R}^d$, and the outputs z_i are real-valued, $z_i \in \mathbb{R}$. Let $y = f(\tilde{\mathbf{x}}, \mathbf{w})$ represent the output of CMAC to input $\tilde{\mathbf{x}}$ (with $\mathbf{w}$ being the CMAC weights), able to learn the correspondence $\mathscr{X} \rightarrow \mathscr{Z}$. Apart from the usual objective of minimizing the Mean Square Error of the training dataset, $MSE = \sum_i |z_i - y_i|^2$, the system is to provide a "good" answer to any input $\tilde{\mathbf{x}}$ not present in that dataset, that is, to keep low $E = (z-y)^2$, where z is the hypothetical desired answer to input $\tilde{\mathbf{x}}$.

2.1 CMAC architecture

Let $\tilde{\mathbf{x}}$ be the d-dimensional analog input vector and $\mathbf{x} = [x_1 \ldots x_d]$ its digitized conversion, where $x_i \in [0, \ldots, P_i - 1]$ and P_i is the number of discrete intervals of the i-th dimension. The output of the CMAC can be expressed as

$$f(\mathbf{x}) = \sum_{m=0}^{p-1} \sum_{\mathbf{n}_m} w_m(\mathbf{n}_m)\, \phi(\mathbf{x} - \mathbf{c}(\mathbf{n}_m)), \tag{1}$$

where p is the integer generalization parameter, $w_m(\cdot)$ is the m-th look-up-table (LUT) ($m = 0, \ldots, p-1$), $\mathbf{n}_m = (n_1^{(m)}, \ldots, n_d^{(m)})$ is the d-component indexing vector for the m-th LUT, $\phi(\mathbf{x})$ is the basis function, defined as

$$\phi(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} \in [0,\, p-1]^d, \\ 0, & \text{elsewhere,} \end{cases} \tag{2}$$

and $\mathbf{c}(\mathbf{n}_m)$ is the d-dimensional vector that shifts the basis function to the position described by

$$\mathbf{c}(\mathbf{n}_m) = p\,\mathbf{n}_m - m[1 \overset{d}{\ldots} 1]. \tag{3}$$

Equation (1) describes the CMAC output as a linear combination of p-wide hypercubes (2) located at fixed places (3). The global structure is based on p independent hypercube grids, each associated to a certain LUT, offset along the main hyperdiagonal in the input space (3).

Given the option range in the CMAC design, such as different alternatives for the basis functions [6], parameter p dependent on the dimension [7], etc., for the sake of simplicity in the exposition, and without loss of generality, the following schema is used: generalization parameter p dimension-independent, the local functions are hypercubes (as described by (2)), the LUTs have equal size, and the discretization levels in each dimension is the same, P. [1]

2.2 Deformable models and active learning

The problem of minimizing the elasticity or rigidity of a one-dimensional curve, $v(s)$, can be accomplished with the minimization of the following energy functional

$$S = \int_0^L \alpha \left| \frac{\partial v(s)}{\partial s} \right|^2 + \beta \left| \frac{\partial v(s)^2}{\partial s^2} \right|^2 ds + S_G, \qquad (4)$$

where α and β are the elasticity and rigidity parameters [5], and S_G is the external force energy. This minimization represents the base of the theory of active deformable models.

The iterative procedure that achieves the minimization of S, its extension to higher dimensional regular structures and its frequency-based implementation is detailed in [8]. In the general case of a discrete d-dimensional regular structure $v(n_1,\ldots,n_d)$, the minimization of its curvature can be accomplished by the following frequency-domain recursive equation

$$V^{(\xi)}(\bar{\omega}) = \frac{\eta}{\eta + K(\bar{\omega})} V^{(\xi-1)}(\bar{\omega}) + G^{(\xi)}(\bar{\omega}), \qquad (5)$$

where ξ is discrete time, $\bar{\omega} = \{\omega_1,\ldots,\omega_d\}$ is the multidimensional frequency (associated to the discrete space $\mathbf{n} = \{n_1,\ldots,n_d\}$), $V^{(\xi)}(\bar{\omega})$ is the Fourier transform of the active structure $v^{(\xi)}(\mathbf{n})$, $G^{(\xi)}(\bar{\omega})$ is the spectrum of the external forces, η is the mass term, and $K(\bar{\omega})$ is the stiffness spectrum, defined by

$$K(\bar{\omega}) = \sum_{i=1}^{d} \alpha_i |1 - e^{j\omega_i}|^2 + \sum_{i=1}^{d} \beta_i |1 - e^{j\omega_i}|^4, \qquad (6)$$

where α_i and β_i are the elasticity and rigidity parameters for the i-th dimension.

Equation (5) represent a smart method for obtaining a smooth hypersurface constrained to external conditions. The degree of smoothness will depend on the values for α_i and β_i in (6). The double mechanism of this process (5), the internal curvature minimization-based criteria and the process of learning external data, represents a powerful training method for reaching both learning and generalization performance. Figure 1 shows graphically the iterative learning mechanism: the input is the external data or training data, and the output the internal parameters of the system at epoch ξ.

3. ACTIVE CMAC

In order to develop a CMAC with the ability to generalize more than what would correspond according to its generalization parameter p, and without loosing learning performance, a training based on the schema of Figure 1 is proposed here. The active training is achieved by implementing the frequency-based formulation of the active deformable model theory on the structure of the CMAC. The external update is accomplished by a passive training algorithm [1][4].

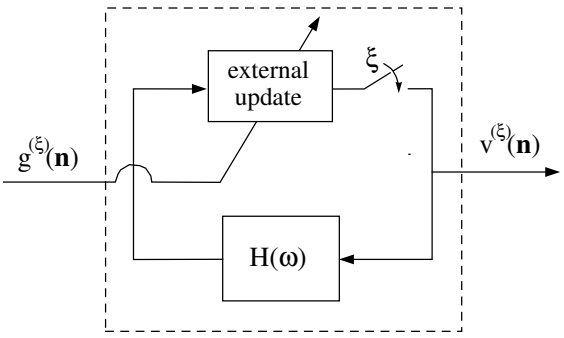

Figure 1: Active learning mechanism.

3.1 Active training

As proven in [3], and as can be deduced from the convolution in (1), the Fourier transform of the CMAC output is

$$F(\bar{\omega}) = \Phi(\bar{\omega}) \sum_{m=0}^{p-1} W_m(p\,\bar{\omega})\, e^{jm(\omega_1+\ldots+\omega_d)}, \qquad (7)$$

where $W_m(\omega_1,\ldots,\omega_d)$ is the Fourier transform of the m-th LUT, that is,

$$W_m(\bar{\omega}) = \sum_{n_1=0}^{M-1} \overset{d}{\ldots} \sum_{n_d=0}^{M-1} w_m(\mathbf{n})\, e^{-j(\omega_1 n_1+\ldots+\omega_d n_d)}, \qquad (8)$$

and $\Phi(\bar{\omega})$ is the Fourier transform of the basis function $\phi(\mathbf{x})$ (assumed to be a hypercube).

In order to implement the active mechanism (5) into the CMAC, $V(\bar{\omega})$ in (5) is to be replaced by the CMAC spectrum (7). This leads to the difference equation [2]

$$\sum_{m=0}^{p-1} W_m^{(\xi)}(p\,\bar{\omega})\, e^{jm\Sigma_i \omega_i} = H(\bar{\omega}) \sum_{m=0}^{p-1} W_m^{(\xi-1)}(p\,\bar{\omega})\, e^{jm\Sigma_i \omega_i}, \qquad (9)$$

where $H(\bar{\omega}) = \eta(\eta + K(\bar{\omega}))^{-1}$.

Equation (9) provides the LUT spectrum update in an implicit fashion. The explicit version is

$$\mathbf{W}^{(\xi)} = \mathbf{H}\, \mathbf{W}^{(\xi-1)}, \qquad (10)$$

where $\mathbf{W}^{(\xi)}$ is the LUT spectra vector

$$\mathbf{W}^{(\xi)} = \left(W_1^{(\xi)}(\bar{\omega})\; W_2^{(\xi)}(\bar{\omega})\; \ldots\; W_p^{(\xi)}(\bar{\omega}) \right)^T, \qquad (11)$$

[1] For this case, that brings simplicity in the notation, the input space is $L_T = P^d$ large, $P = pM+1$, the size of each LUT is M^d, and the total number of weights in CMAC is thus $N = pM^d$.

[2] The term of external forces is not included in this development. As shown in Figure 1, this external update and the new process for spreading the knowledge (9) are performed sequentially.

and matrix $\mathbf{H}$ contains the coupling among the LUTs,

$$\mathbf{H} = \begin{pmatrix} H_0(\bar{\omega}) & H_1(\bar{\omega}) & \dots & H_{p-1}(\bar{\omega}) \\ H_{-1}(\bar{\omega}) & H_0(\bar{\omega}) & \dots & H_{p-2}(\bar{\omega}) \\ \vdots & \vdots & \ddots & \vdots \\ H_{-p+1}(\bar{\omega}) & H_{-p+2}(\bar{\omega}) & \dots & H_0(\bar{\omega}) \end{pmatrix}. \quad (12)$$

From the previous equations it is simple to deduce that each element of matrix $\mathbf{H}$ is defined by

$$H_m(\bar{\omega}) = \sum_{k_1=0}^{p-1} \overset{d}{\cdots} \sum_{k_d=0}^{p-1} H\left(\frac{\bar{\omega} - 2\pi\mathbf{k}}{p}\right) e^{jm\sum_i \frac{w_i - 2\pi k_i}{p}}, \quad (13)$$

where $\mathbf{k} = \{k_1, \dots k_d\}$ is the index of the spectral replica.

The spatial counterpart of equation (10) is

$$w_m{}^{(\xi)}(\mathbf{n}) = \sum_{k=0}^{p-1} h_{k-m}(\mathbf{n}) * w_k{}^{(\xi-1)}(\mathbf{n}), \quad (14)$$

where $*$ denotes discrete convolution, and $h_{k-m}(\mathbf{n})$ is the convolution kernel that couples the k-th LUT to update the m-th one.

3.2 Neural interaction

Equation (14) represents the active part of the CMAC novel training proposed in this paper. Returning to Figure 1, this part of the training is carried out by the linear filter block. The update of the LUT weights in (14) is performed internally and under no external influence. After this active LUT adjustment, the external update of the LUTs is carried out by a passive training algorithm and with all the training data. The whole process is repeated several times in order to learn profusely the external data (reinforcement) and to propagate that knowledge to areas that are not externally explored.

A graphical interpretation of the CMAC active learning (14) is contained in Figure 2: the active update of a (dark shaded) cell in the m-th LUT is carried out with a specific linear combination of the very cell and the neighboring ones.

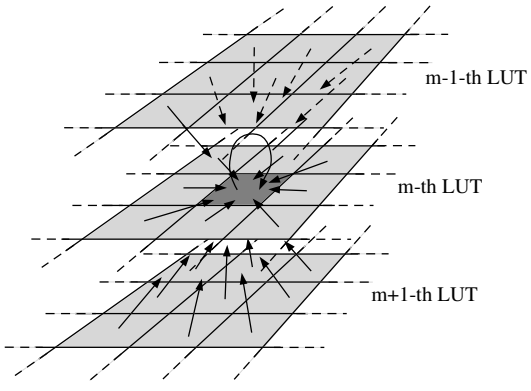

Figure 2: Cell interaction caused by the active training.

The weight of a k-th LUT cell in that linear combination is ruled by the respective value in kernel $h_{k-m}(\mathbf{n})$. Although from a theoretical point of view all cells of all LUTs are involved in that linear combination, from a practical approach only a very few number of cells are relevant. These are the closest neighbors to the cell. Finally, the set of p LUTs have a circular property, that is, the 0-th and $p-1$-th LUTs are neighbors. This can be stated by analyzing (13), which gives also rise to the following properties of the spatial kernels

$$h_{k\pm p}(\mathbf{n}) = h_k(\mathbf{n} \pm [1 \overset{d}{\cdots} 1]), \quad (15)$$

$$h_{-k}(\mathbf{n}) = h_k(-\mathbf{n}). \quad (16)$$

4. SIMULATION RESULTS

In order to evaluate the active training we chose one of the first scenarios that the CMAC was intended for: inverse kinematics of a (virtual) robotic arm. For the sake of simplicity, the arm has two L-long segments, and these are controlled respectively by angles θ and ϕ. For a certain angle value the position reached by the arm is given by

$$x = -L\sin(\theta) + L\sin(\theta + \phi) \quad (17)$$

$$y = \ L\cos(\theta) - L\cos(\theta + \phi) \quad (18)$$

The inverse of the previous equations is a smooth function of difficult analytical solution. In order to test generalization and learning performance simultaneously, an artificial disturbance (based on a narrow gaussian) was added to the inverse equations, this simulating an exceptional case in the robotic arm kinematics (for instance, to avoid a certain obstacle with the help of additional segments).

In this scenario, the training was carried out with two main methods: a) a passive training (the Albus learning rule, or LMS), and the active learning proposed in this paper ($\eta = 0.2$, $\alpha_i = 1$, $\beta_i = 0$, and the LMS is the passive part). The details of the simulations are: the input spatial coordinates x and y were discretized in $P = 37$ levels, a set of space locations $\mathscr{X} = \{(x_1, y_1) \cdots (x_L, y_L)\}$ together with the corresponding angles $\mathscr{Z} = \{(\theta_1, \phi_1) \cdots (\theta_L, \phi_L)\}$ were acquired randomly, and $L = 137$ (which represents a training set 10% representative of the total input data, $L_T = 37^2$).

In Figure 3 and Table 1 the graphical and numerical results of these experiments are shown respectively. The graphical results of Figure 3 show the excellent performance of the active training for learning and generalizing simultaneously. As expected, at higher values of p the CMAC is not able to learn properly the narrow gaussian shape, but generalization is achieved at any value of p. The Albus rule is not an efficient learning mechanism for disperse training datasets, and generalization and learning performance cannot be achieved simultaneously. The previous facts are supported by the numerical results of Table 1: the active training provides excellent performance for learning (E_t) and generalization (E_g), and both numbers are close to the actual CMAC learning performance (E column); on the contrary, the Albus rule gives rise to overfitting (E_t) and poor generalization (E_g).

Finally the active learning does not provide a proper solution at the boundaries, because of an inefficient strategy applied at those regions. A proper solution for generalizing at the boundaries, based on the ideas in [8], is currently under investigation. Also the understanding of the solutions achieved with the parameters of the active training (η, α, β), which have a direct influence on the spectrum of the output, may give another point of view to the analysis of the learning capabilities of the CMAC.

architecture			active training		passive training	
p	N/L_T	E	Et	Eg	Et	Eg
2	53%	-53	-41	-26	-54	-2
4	29%	-44	-33	-30	-54	-8
6	21%	-35	-32	-30	-54	-14
9	16%	-28	-27	-25	-50	-18
12	14%	-25	-23	-23	-36	-18

Table 1: Comparison between the performance of CMAC active and passive trainings. The column 'architecture' describes the CMAC: p generalization parameter, N/Lt ratio CMAC size/input space size, and E residual training error (in dB) using a training dataset that covers the whole input space . Columns 'active' and 'passive' contain the performance for the respective training methods on the (reduced) actual training dataset: Et and Eg are respectively the MSE (in dB) over the training dataset and on the input space that is not available for training.

5. CONCLUSIONS

This paper has proposed a novel training algorithm for the CMAC neural network. The theory of active deformable models on the CMAC architecture leads to a knowledge-propagation mechanism that takes place among the CMAC weights. The graphical and numerical results on the problem of robotic arm inverse kinematics prove the excellent learning and generalization performance of the new active CMAC, especially with sparse training datasets. The frequency-based perspective of the new algorithm may provide a better understanding of the CMAC learning capabilities. This suggestion along with a proper learning mechanism at the input space boundaries cope the further research work.

REFERENCES

[1] J.S. Albus, "A new approach to manipulator control: the Cerebellar Model Articulation Controller (CMAC)," *J. Dyn. Sys., Measur. and Control*, 97, pp. 220-227, 1975.

[2] M. Brown, C. J. Harris, and P. C. Parks, "The interpolation capabilities of the binary CMAC," Neural Networks, Vol. 6, pp. 429-440, 1993.

[3] F.J. González, A.R. Figueiras, and A. Artés, "Fourier analysis of the generalized CMAC neural network," *Neural networks*, 11, pp. 391-396, 1998.

[4] H. Chao, X. Lixin, and Z. Yuhe, "Learning Convergence of CMAC Algorithm," *Neural Processing Letters, Kluwer Acad. Pub.*, 14(1), pp. 61-74, 2001.

[5] J. Liang, T. McInerney, and D. Terzopoulos, "United Snakes," *Proc. 7th IEEE Intl. Conf. Comput. Vision*, pp. 933-940, 1999.

[6] C.T. Chiang and C.S. Lin, "CMAC with general basis functions," *Neural Networks*, 9, pp. 1199-1211, 1996.

[7] F.J. González, A. Artés, and A.R. Figueiras, "Generalizing CMAC Architecture and Training," *IEEE Trans. Neural Networks*, vol. 9, no. 6, pp. 1509-1514, 1998.

[8] L. Weruaga, R. Verdú, and J. Morales, "Frequency domain formulation of active deformable models," accepted at *IEEE Trans. Pattern Anal. and Machine Intell.*.

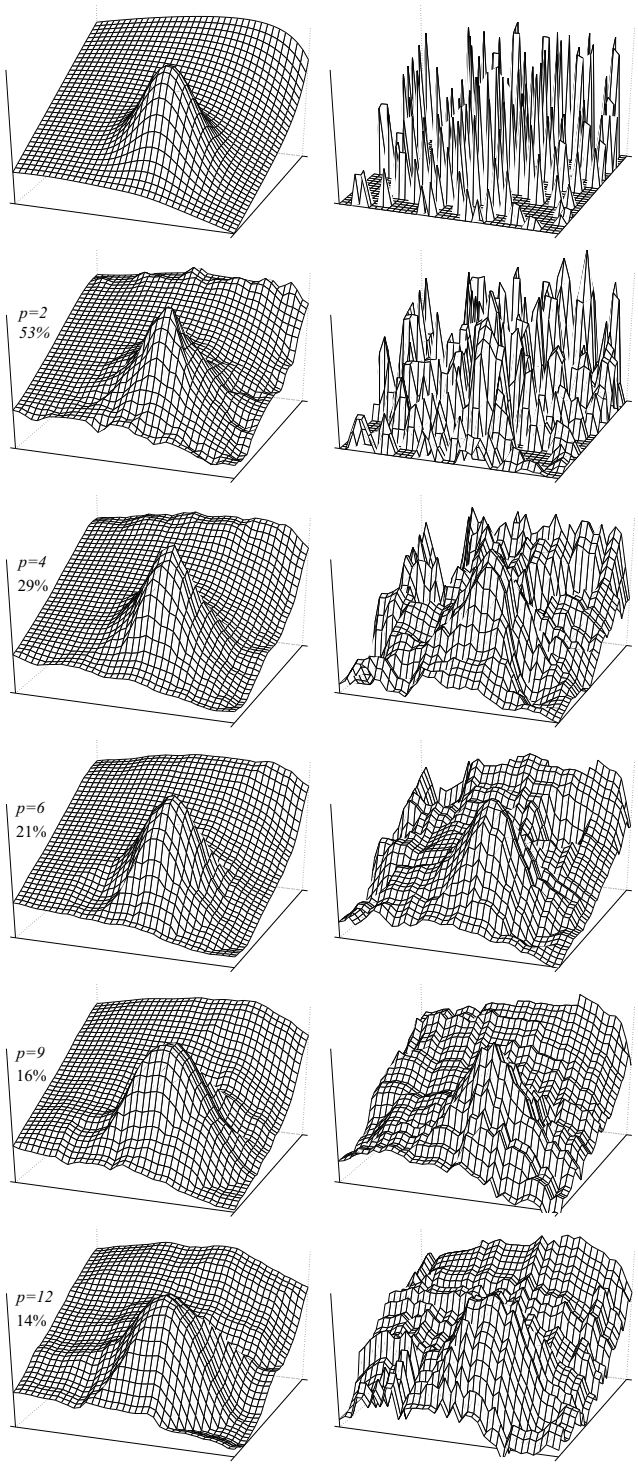

Figure 3: The upmost row contains the surface of the analytical solution for angle θ of the inverse robotic arm kinematics (left picture), and the training dataset (right one); the following rows contain the output of the CMAC after the training process with the active training (left picture) and the Albus rule (right one) for different generalization values. In each row, the generalization parameter p and the ratio N/L_T are shown at the left.