ESANN'1998 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 22-23-24 April 1998, D-Facto public., ISBN 2-9600049-8-1, pp. 1-6

Extending the CMAC Model: Adaptive Input Quantization

Grzegorz Piotr Klebus

Institute of Electronics Fundamentals, Warsaw University of Technology
Nowowiejska 15/19, 00–665 Warsaw, POLAND
E-mail: klebus@ipe.pw.edu.pl

Abstract. This paper presents an extension to the Cerebellar Model Articulation Controller, a fast and efficient approximation tool. The input transformation in the standard model is fixed, thus the internal representation does not adapt to the distribution of the input data. The proposed approach adds to the input transformation adaptiveness of the controllable range. The new algorithm does not change the rest of the standard CMAC model, so it can be easily integrated into existing applications.

1. Introduction

The Cerebellar Model Articulation Controller, proposed by Albus [1] as a model of cerebellum, is a learning architecture with local generalization property. It is a useful method for function approximation and was applied by many researchers in numerous tasks (see [2]). The major advantage of the model is its very fast training and ability to approximate small- to medium-dimensional nonlinear mappings.

The CMAC model is a kind of sparse coarse-coded look-up table, although it can be considered a neural network [4]. It quantizes the input space that is further transformed into a highly-dimensional basis functions space. Basis functions are linearly combined to produce the model output. In the standard model the transformation is fixed and does not depend on the input data. In this paper a novel approach is presented which lets the CMAC model self-tune to the distribution of the training data, thus constructing better approximation than in the case of the fixed input transformation. The network with the adaptive input quantization still possesses all the advantages of the standard CMAC.

2. Standard CMAC Model

The CMAC model is a lattice-based associative memory with local generalization capability [2, 4]. The given input vector $\mathbf{x}$ from the input space $X \subset \mathbb{R}^n$ is first quantized:

This work was partially supported by the Polish Committee for Scientific Research under Grant No. 8T11C 046 11.

for each input dimension the activated quant number is computed. The vector of activated quants numbers is then used to determine the activation values of the basis functions a_i. The p basis functions form a vector in a highly dimensional association space. The input transformation is constructed in such a way that only a constant number (called a *generalization parameter*, ρ) of basis functions is activated. The output transformation is a simple linear combiner. The vector of the basis functions values forms a kind of extended representation of the original input vector. This representation is build in such a manner that even complex nonlinear problems can be solved using the linear output transformation. The input transformation is topology conserving, in the sense that input vectors near in the input space activate the association space sets with a number of common elements, and distant vectors activate completely unrelated basis functions. This feature provides a local generalization ability with range controlled by the generalization parameter.

The model output is $y(\mathbf{x}) = \sum_{i=1}^{p} a_i(\mathbf{x}) w_i$ for the given input vector $\mathbf{x}$. Since only a subset of ρ basis functions is activated (with nonzero values), and $\rho \ll p$ usually holds, it suffices to sum only the basis functions from the subset.

Implementations of the CMAC model usually use hash coding techniques to compress the large association space into a manageable weights space; see, e.g., [2]. A model with more than one output can be easily constructed using one quantization module and many output weights vectors. After the activated basis functions responses are determined, they are used in weighted sums for each output.

Input Quantization. The input quantization transforms the input space X into some *normalized input space* X' by means of quantization. Input dimensions must be bounded, that is two values should be specified for each of them, $x_i^{\min}$ and $x_i^{\max}$, denoting the minimum and maximum value of the i-th input.

Each input dimension is divided into a number of non-overlapping intervals, or quants. Quantization is determined by specifying *knots*, which divide each dimension into intervals. Knots are denoted $\lambda_{i,j}$, where $i = 1,\ldots,n$ is the input dimension, and $j = 1,\ldots,r_i$ is the number of knots for that dimension. The knots must satisfy the following inequality: $x_i^{\min} < \lambda_{i,1} \leq \cdots \leq \lambda_{i,r_i} < x_i^{\min}$. These knots are termed *internal*. It is convenient to introduce *external knots* that lay outside or on the input boundaries; namely, let $\lambda_{i,0} = x_i^{\min}$ and $\lambda_{i,r_i+1} = x_i^{\max}$.

The j-th quant of the i-th dimension is defined as

$$I_{i,j} = \begin{cases} [\lambda_{i,j-1}, \lambda_{i,j}) & \text{for } i = 1,\ldots,r_i, \\ [\lambda_{i,j-1}, \lambda_{i,j}] & \text{for } i = r_i + 1. \end{cases} \tag{1}$$

There are $r_i + 1$ intervals on dimension i. The input value x_i activates the quant inside which it lies.

There are different knot placement strategies. Quants can be of the same width (the uniform quantization), or of variable widths. The latter helps the model construct finer approximation with respect to the probability density of input data. Later in this paper an algorithm for automatic adaptation of the input quantization is presented, which eliminates the need to hand-tune the CMAC input transformation.

ESANN'1998 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 22-23-24 April 1998, D-Facto public., ISBN 2-9600049-8-1, pp. 1-6

Basis functions. Basis functions are defined on compact supports (receptive fields), that is they have nonzero output on bounded regions of the input space. They are organized in ρ lattices, or planes, in such a manner that supports do not overlap and cover all the input space. The support of each basis function is a hypercube of edge ρ in the normalized input space. Lattices are displaced relative to each other and any input vector activates only one basis function from each lattice. Several displacement strategies exist; see, e.g., [2]. In the Albus CMAC, the basis functions were binary, thus implementing a piecewise constant approximation. It is also possible to use higher order basis functions to obtain smooth approximation [2].

Training. The CMAC model is trained using a simple Least Mean Squares (LMS) rule $\Delta w_i = \eta(d-y)$, thus avoiding the problem of local minima; η is the training rate, d is the desired output.

3. Adaptive Input Quantization

The properly constructed input quantization can improve the model performance by using greater number of fine quants in the regions of the input space in which the input data are dense. The approximation is then more accurate where the problem is better represented by the training set. The standard CMAC model, however, does not have the ability to adapt the input quantization to the input data distribution. This section presents an algorithm for adaptive input quantization in the CMAC model.

The rough idea behind this algorithm is to try to move knots near the presented input value slightly towards it. After a sufficient number of training steps, the input quantization is reorganized: quants become finer in the regions with high probability of data occurrence. This way the quantization self-adapts to the underlying training data distribution . The formulation of the algorithm is similar to competitive learning in unsupervised neural networks [3].

Let us consider an input dimension i; the i-th input value is thus x_i. For each internal knot ($j = 1, \ldots, r_i$) of the input i the update rule

$$\Delta\lambda_{i,j}(t) = \alpha(t)h(t)\left[x_i - \lambda_{i,j}\right] \tag{2}$$

is applied, where t denotes the current training step. Consequently, knots move slightly towards the current input value x_i. In the above equation $\alpha(t)$ denotes the time-varying learning rate, $h(t)$ is a neighborhood (or kernel) function that decreases to 0 as the knot value moves away from the input value. The widths of certain quants can become very small after a large number of training steps; this can be avoided by applying (2) only if after that the quant width is greater than the minimum assumed width. The learning rate in (2) controls the amount of correction applied to the knot values, and should decrease during learning to avoid instability.

After the learning rule is applied, the standard CMAC algorithm is employed to determine the absolute output error and weight vector update.

ESANN'1998 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 22-23-24 April 1998, D-Facto public., ISBN 2-9600049-8-1, pp. 1-6

3.1. Neighborhood Functions.

The neighborhood function h controls the range of the input value in which the knots positions are adapted during training. The radius of neighborhood should decrease as training proceeds. The function h depends upon the distance between the input value x_i and the considered knot value $\lambda_{i,j}$. There are two classes of neighborhood functions: *input space functions* $h(x_i, \lambda_{i,j}, t)$, measuring the closeness of the input value and the knot value, and *normalized space functions* $h(k, j, t)$, measuring the adjacency of the activated quant k and the considered knot number j.

Input Space Neighborhood Functions. Generally, input space functions can be easily formed as decreasing univariate functions of the distance between the input and knot values. They may also have an adjustable parameter σ determining the width of the neighborhood. The value decrease in time, e.g., $\sigma(t) = \sigma_{\text{init}}\sigma_{\text{dec}}^t$, to obtain a closer neighborhood in the final phase of training. The value σ_{init} in the above equation is the initial neighborhood radius, and $0 < \sigma_{\text{dec}} \leq 1$ is the decay rate. See examples in Appendix A.

Normalized Space Neighborhood Functions. The normalized space neighborhood functions measure the distance between points on the quantized input axes and also use some non-increasing univariate function of the distance as their value. See examples in Appendix A.

Features of the extended CMAC model The proposed extension to the standard CMAC model provides a simple and effective method of tuning the input quantization to the distribution of the input data in unsupervised way. The algorithm resembles the known competitive neural networks learning techniques, although, by contrast, it operates on each of the input dimensions, not on the whole input space. It may be applied without any modification to the rest of the standard CMAC algorithm, either the classical Albus formulation or improvements thereof, and it works in an on-line manner[1].

4. Experimental Results

Preliminary experimental evaluations have been carried out using the `building` dataset from the PROBEN1 collection. They aimed at showing that the proposed approach may be used in certain situations instead of the standard CMAC model to obtain better approximation.

The experiments were conducted on a slightly modified `building` dataset: the day of week was coded as a real number instead of 1-of-7 binary values. There was no validation set, it was used as a part of the training set. Other settings were standard. Each experiment consisted of 500 runs. The experiments were carried out for three different generalization ratios for four CMAC models: the standard one ('std') and

[1] adaptation occurs after presentation of each training vector independently upon other presentations

ESANN'1998 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 22-23-24 April 1998, D-Facto public., ISBN 2-9600049-8-1, pp. 1-6

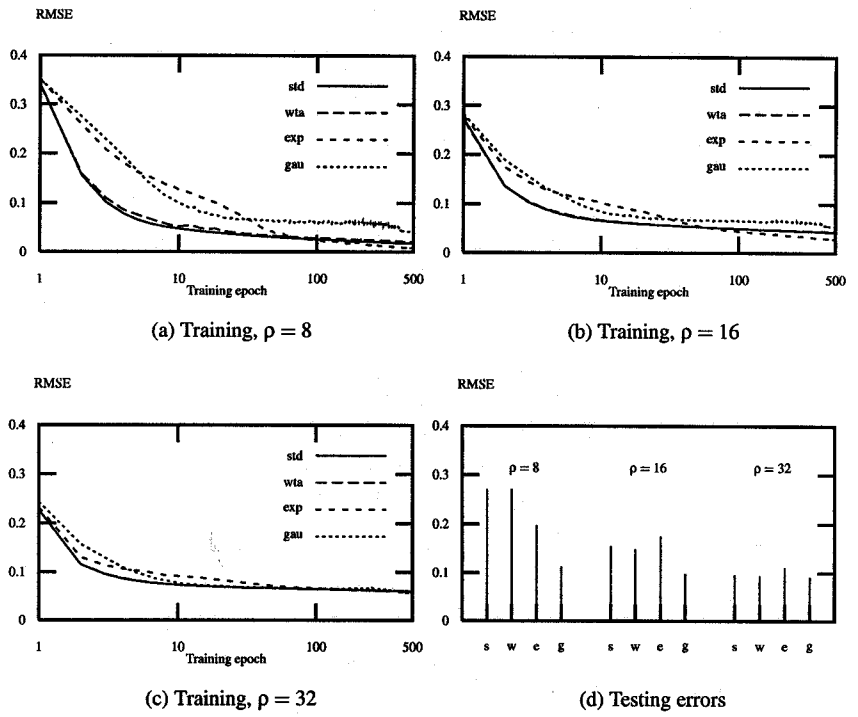

Figure 1: Learning curves for different ρ values (x axis in logarithmic scale)

the adaptive ones with different neighborhood functions – WTA ('wta'), exponential ('exp'), and Gaussian ('gau'). The results are presented in Fig. 1, where the Root Mean Square (RMS) Error is plotted versus the training epoch number. Due to the space constraints of the paper, the full specification of the experiments and the detailed results can be obtained from the author via e-mail.

The results of the preliminary experiments show that the convergence speed during training is similar for all the algorithms, not counting a few first presentations where the standard CMAC performs better. The generalization capability is, however, reflected by the test set errors. It follows from Fig. 1(d) that the adaptive quantization extensions can substantially improve the generalization in CMAC especially when the generalization parameter ρ is low. For higher ρ values the performance is approximately the same, since the generalization mechanism of the standard CMAC itself is satisfactory.

The best properties has the algorithm with the Gaussian neighborhood function. Worse performance of the remaining two functions is probably caused by somewhat ad hoc choice of neighborhood radii widths for each case. It is worth noting, that the larger neighborhood yields better results; compare the poor performance of the WTA rule, in which only the two nearest knots are updated.

More experiments are needed to examine the properties of the proposed methods

ESANN'1998 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 22-23-24 April 1998, D-Facto public., ISBN 2-9600049-8-1, pp. 1-6

to a greater extent. It is particularly interesting how the neighborhood initial radius influences the approximation quality of the extended CMAC model.

5. Conclusion

This paper presents a new extension to the CMAC model. The adaptive input quantization lets the model approximate correctly even when the generalization ratio ρ is relatively small. This approach seems promising and worth continuing. Particularly important are experimental evaluations investigating the influence of the neighborhood radius on the training capability.

The ability to add and remove quants when appropriate is another enhancement to the standard model, allowing still better adaptation to the training data. Along with the adaptive quantization, these two additions to the standard CMAC would substantially improve the self-organization capabilities of the network.

References

[1] J.S. Albus. *Theoretical and Experimental Aspects of a Cerebellar Model.* PhD thesis, University of Maryland, 1972.

[2] M. Brown and C. Harris. *Neurofuzzy Adaptive Modelling and Control.* Prentice Hall International, 1994.

[3] T. Kohonen. The self-organizing map. In *Proc. IEEE*, volume 78, pages 1464–1480, 1990.

[4] W.T. Miller, F.H. Glanz, and L.G. Kraft. CMAC: An associative neural network alternative to backpropagation. In *Proceedings of the IEEE*, volume 78, pages 1561–1567, 1990.

A. Examples of Functions

Input Space Neighborhood Functions

Square
$$h(x_i,\lambda_{i,j},t) = \begin{cases} 1 & \text{if } |x_i-\lambda_{i,j}| \le \sigma(t), \\ 0 & \text{otherwise;} \end{cases} \tag{3a}$$

Gaussian
$$h(x_i,\lambda_{i,j},t) = \exp\left(-(x_i-\lambda_{i,j})^2/\sigma(t)\right) . \tag{3b}$$

Normalized Space Neighborhood Functions

WTA
$$h(k,j,t) = \begin{cases} 1 & \text{for } j=k,k+1, \\ 0 & \text{otherwise;} \end{cases} \tag{4a}$$

Exponential
$$h(k,j,t) = \begin{cases} \exp(-|k-j|/\sigma(t)) & \text{if } j \le k, \\ \exp(-(|k-j|-1)/\sigma(t)) & \text{if } j > k. \end{cases} \tag{4b}$$