PERGAMON

Neural Networks 12 (1999) 107–126

Neural Networks

Contributed Article

Basis function models of the CMAC network

Aleksander Kołcz[a,*], Nigel M. Allinson[b]

[a]*ECE Department, University of Colorado at Colorado Springs, 1420 Austin Bluffs Pkwy., Colorado Springs, CO 80918, USA*
[b]*Dept. of Electrical Engineering and Electronics, University of Manchester Institute of Science and Technology, Manchester, UK*

Received 9 July 1996; accepted 7 July 1998

Abstract

An interpretation of the Cerebellar Model Articulation Controller (CMAC) network as a member of the General Memory Neural Network (GMNN) architecture is presented. The usefulness of this approach stems from the fact that, within the GMNN formalism, CMAC can be treated as a particular form of a basis function network, where the basis function is inherently dependent on the type of input quantization present in the network mapping. Furthermore, considering the relative regularity of input-space quantization performed by CMAC, we are able to derive an expected (or average) form of the basis function characteristic of this network. Using this basis form, it is possible to create basis-functions models of CMAC mapping, as well as to gain more insight into its performance. The developments are supported by numerical simulations.

Keywords: Memory-based networks; Basic functions; GMNN; RBF; GRNN; CMAC

Nomenclature

$\hat{}$	estimate operator
$\|\ \|$	norm
$\|\ \|_2$	Euclidean norm
$\|\ \|$	absolute value, city-block norm
$\lceil\ \rceil$	ceiling operator: $\lceil x \rceil = \min_n n \geq x$ for integer n
$\lfloor\ \rfloor$	floor operation: $\lfloor x \rfloor = \max_n n \leq x$ for integer n
A_k	memory address (function) for the kth GMNN/CMAC node
d_A	fractal dimension of a chaotic attractor
D	dimension of the input space
E	expectation operator
$f(\mathbf{x})$	function to be approximated, unknown mapping
$g(\mathbf{x})$	CMAC mapping
$\kappa(\mathbf{u},\mathbf{v})$	basis (kernel) function, GMNN/CMAC proximity kernel
K	the number of nodes of a GMNN/CMAC, CMAC quantization constant
L	range of a (scalar) quantizer
$\mu(\mathbf{x})$	activation function
o	length of a scalar (global-level) quantization cell
Ω	input space
$\pi(f,g)$	error measure/criterion
q_k	scalar quantizer associated with the kth network node
$\boldsymbol{Q}$	quantization matrix
$\rho(\mathbf{u},\mathbf{v})$	GMNN/CMAC address distance
$\Re$	real space
$\Re^D$	D-dimensional real space
T	number of training-set samples
U	length of a scalar (nodal-level) quantization cell
$w_k(\mathbf{x})$	weight selected by $\mathbf{x}$ at node k
y	scalar point in the output space
x, $\mathbf{x}$	point in the input space

1. Introduction

The Cerebellar Model Articulation Controller (CMAC) was introduced by Albus (1971, 1975a, 1975b, 1979) who, concurrently with Marr (1969), developed a functional model of the mammalian cerebellum. The model takes advantage of the high degree of regularity present in the organization of the cerebellar cortex and offers numerous advantages from the implementational point of view. Furthermore, the network is inherently dependent on its adjustable parameters in a linear way (which makes it attractive where the training is concerned), and so well-understood linear algorithms (such as least mean squares (LMS)) are applicable. The CMAC network has become especially popular in the areas of robotics and control where the real-time capabilities of the network are of particular importance (Miller et al., 1990; Tolle and Ersü, 1992). Although a large portion of the reported results concerning

* Corresponding author. Tel.: +1-719-262-3187; Fax: +1-719-262-3589; e-mail: ark@eas.uccs.edu

0893-6080/99/$ - see front matter
PII: S0893-6080(98)00113-0

CMAC focuses on employing the network in practical applications, several more rigorous theoretical analyses of the CMAC mapping have also been considered (Parks and Militzer, 1989; Cotter and Guillerm, 1992), and some constraints on the classes of nonlinear mappings realizable by this network have been identified (Brown et al., 1993).

It has also been pointed out (Kołcz, 1996) that CMAC belongs to a wider class of neural network architectures, which has been recently introduced as the General Memory Neural Network (GMNN) (Kołcz and Allinson, 1995). In particular, networks of this class can be interpreted as particular variants of basis function architectures (e.g., radial basis function (RBF) (Broomhead and Lowe, 1988; Powell, 1992) and kernel regression (KR) (Härdle, 1990; Specht, 1991) networks), which provides additional insight into the properties of their mapping. In (Kołcz and Allinson 1995), we suggested that networks of the GMNN type whose structure is particularly regular could be modelled by basis function networks, with the basis function being an estimated (or average) version of the basis characteristic of the particular GMNN variant. In this paper, we propose such a representation of the CMAC network and demonstrate its usefulness in predicting and analysing the network behaviour. The paper is organized as follows: In Section 3 we introduce the general concept of CMAC mapping, particularly in the context of its equivalence with the GMNN architecture. Section 4 discusses the input-space quantization performed by CMAC, with emphasis being placed on the case of uniform quantization; standard and modified versions of CMAC encoding are considered. In Section 5 the expected form of the CMAC address distance is derived (for the uniform quantization case) and compared with experimental data. Section 6 provides performance comparison between the CMAC network and its basis-function model on a case problem of chaotic time-series prediction. The paper is concluded in Section 7.

2. CMAC and its generalizations

2.1. General structure of CMAC mapping

The function of CMAC has its roots in the operation of a biological cerebellar cortex and has an appeal due to its intuitiveness and simplicity. Essentially, CMAC covers the input space with a number of overlapping ‘sensors’ or ‘association cells’, such that each sensor is active for points within a certain small region of the input space and a fixed number of K sensors is activated by any given input (see Fig. 1). Any particular input to the network generates a response in the form of the combination of the sensors activated by this input. In fact, the CMAC architecture incorporates a linear array of memory cells (i.e., weights), with the sensors playing the role of address decoders. Thus, any input point activates exactly K memory cells, whose contents are then combined (e.g., by summation) into the output response of the network. The set of activated sensors is unique for points within a small region of the input space and, due to the overlapping arrangement of the sensors, points lying close to each other will lead to similar responses of the CMAC network.

More formally, the CMAC network represents a multivariate nonlinear mapping

$$g:\ \Re^D \supset \Omega \rightarrow \Re, \tag{1}$$

which domain is given by a compact (usually hyper-rectangular) region, Ω, of the D-dimensional real space, $\Re^D$, whereas the output range is given by $\Re$. Typically, the mapping produced by CMAC is piecewise constant. Vector-valued variants of the network can be considered as a direct extension of the scalar-valued case and will not be treated here.

The objective of CMAC mapping is to approximate a given continuous and smooth function, f, so that the distance, $\pi(g, f)$, between f and its estimate, g, is minimized. Of course, there exist numerous neural network architectures (and other nonparametric regression estimators) that can be used as function approximators. CMAC belongs to a distinctive class of architectures that base their operation on multiple quantization of the input space, with the distinguishing feature of the CMAC network within this class being the particular way in which the quantization is performed. CMAC can be considered to consist of a single layer of memory locations, an address-generating unit and a summing unit, where for any $\mathbf{x} \in \Omega$, K distinct memory addresses are generated (with K being a fixed number, usually much smaller than the total number of available memory locations). The contents of the addressed memory locations (i.e., weights) are subsequently summed to provide the overall network response to $\mathbf{x}$. The particular form of memory addressing (which will be described later in the context of CMAC quantization) employed by CMAC ensures that input points close to each other in Ω will share some memory locations, whereas points that are distant in the input space result in the selection of unrelated sets of memory weights. Thus defined, CMAC is characterized by a piecewise-constant response, although modifications to the basic architecture have been proposed (Lane et al., 1992), where the addressed weights are additionally modulated by external basis functions (e.g., B-splines), which increases the smoothness of CMAC mapping.

It can be seen that the discrete nature of address generation present in CMAC mapping imposes a form of quantization on the input space, whereby Ω is partitioned into adjacent disjoint regions, with points in a single region addressing the same set of K weights in CMAC memory. In fact, the principle of implicit input-space quantization present in CMAC mapping, combined with the constant number of weights selected for any response computation, can be extended to a larger class of architectures (i.e., the GMNN), which may differ with respect to the particular

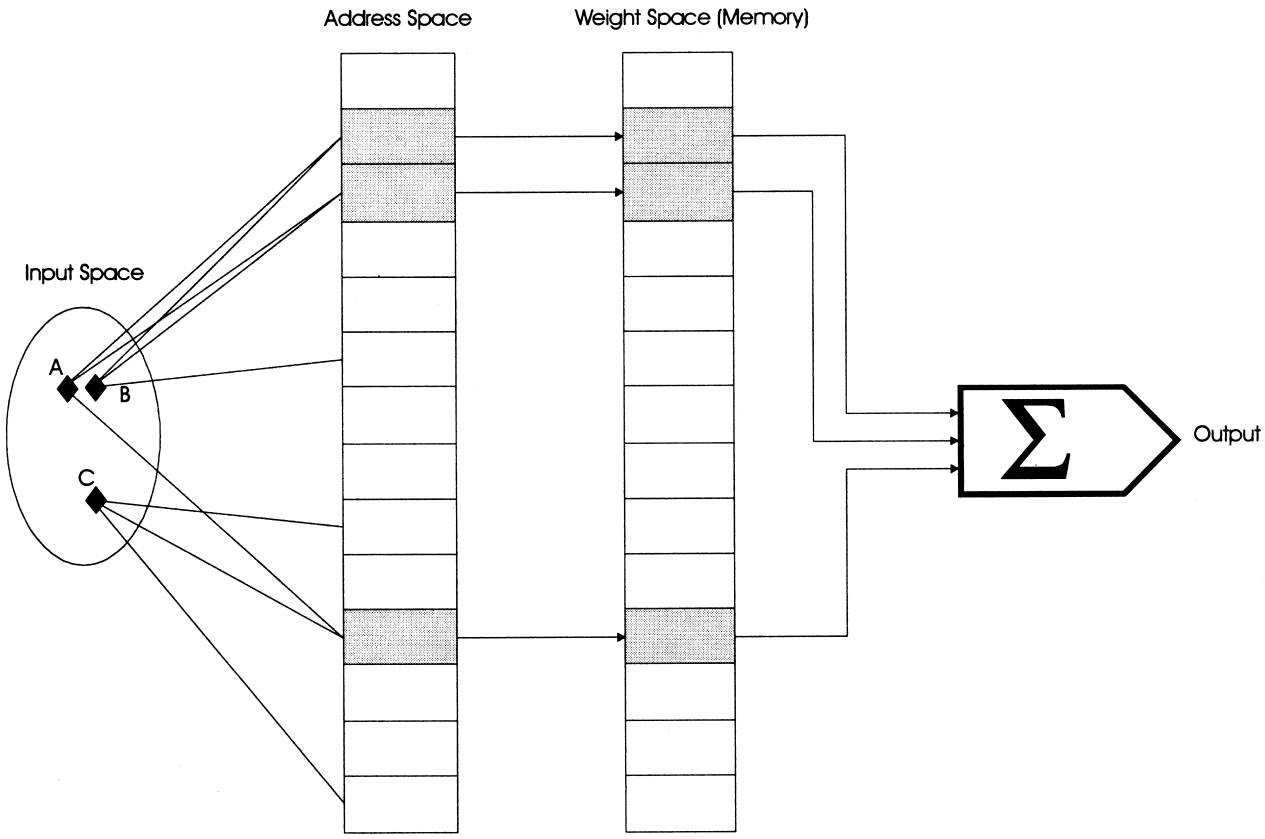

Fig. 1. Outline of the CMAC mapping. Every point in the input space 'excites' a fixed number (i.e. K) of association cells ($K = 3$ in this case), which in turn specify the physical memory locations containing adjustable weights. The weights can be simply added to produce an output. As illustrated here, points close to each other in the input space (i.e. points A and B) will share more association cells than points distant in the input space (i.e. points A and C, or points B and C). Grey boxes in the figure indicate the association cells and memory weights selected for point A. In principle, the space of association cells (memory addresses) can be much larger than the actual size of the physical memory (weights).

form of quantization (Kołcz and Allinson, 1995). The general framework of the GMNN provides insight into the operational principles of networks of the CMAC type and allows the establishment of links between networks of this class and basis-function architectures.

To take advantage of the GMNN interpretation of CMAC mapping, we briefly outline the concepts of the GMNN architecture before the structure of CMAC quantization is discussed in more detail.

2.2. The GMNN framework

In many practical realizations of basis-function networks (e.g., RBFs), the size of the basis set may pose serious computational challenges (in terms of achieving 'reasonable' training and response times) when the number of basis functions is large (e.g., when there is one basis function per training point and the training set contains many elements). One of the reasons for such situations is the infinite (or unbounded) support of popularly used basis functions (e.g., Gaussian). Considering that the 'mass' of basis functions is usually well localized in Ω, many of them will produce near-zero values for any given evaluation point, with only a small portion affecting network output. Hence, the process of evaluating all the network's basis functions is superfluous and hinders network performance. Several approaches to deal with this problem have been proposed. In particular, the number of basis functions participating in any computation can be reduced to those which are significantly different from zero at the current evaluation point (Moody and Darken, 1989), or alternatively, only the basis functions that are nearest to the current evaluation point are taken into account. Other methods of dealing with large training sets involve data clustering (Specht, 1991) and subset selection (Broomhead and Lowe 1988) (where only a small portion of the possible set of basis functions is used to define the network).

The GMNN represents a formalization of an approach where only (and exactly) K basis functions participate in any network-output computation. This requirement makes the support (or receptive field) of every basis function finite, and inherently imposes a K-fold quantization on the input space. The latter property stems from the fact that (by definition) exactly K bases overlap (or are 'active') at any particular point in Ω. For each of the K 'active' functions, it is possible to identify a set of neighbours (i.e., basis functions

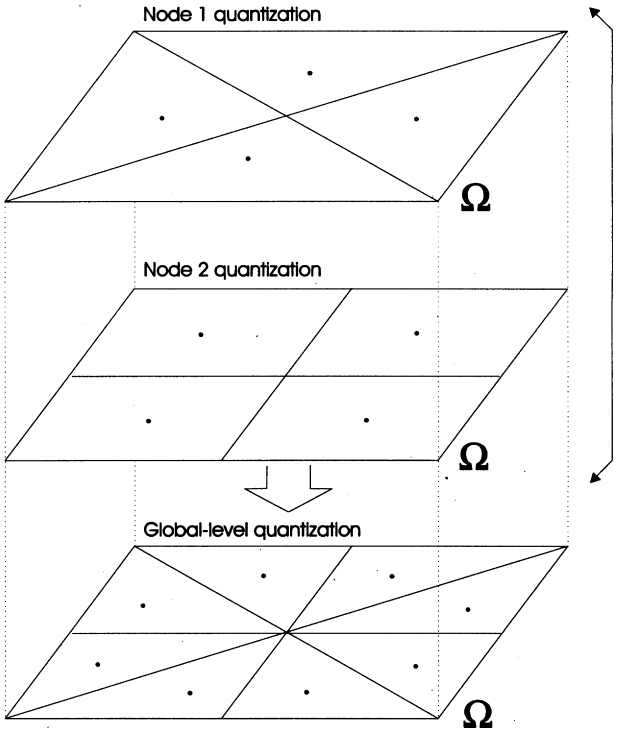

Fig. 2. A superposition of distinct low-resolution quantizations leads to a global VQ of a higher resolution.

that become 'active' when the support boundary of the current basis is crossed), which are adjacent but nonoverlapping with this particular basis. By repeating this argument recursively, it can be seen that each member of the original K 'active' basis functions has associated with it a set of functions that partition the input space into an arrangement of adjacent but nonoverlapping cells (where a cell is equivalent to the support region of one basis function), thus defining a quantization over Ω. Consequently, the whole network defines K individual quantization layers, where it is assumed that no two cells belonging to different layers can be identical. Note that when the individual quantization-cell arrangements are superimposed, a new (global-level) quantization of Ω results, with resolution higher than any of the K individual layers (see Fig. 2). In this way, by varying the form of the individual quantizers and their number, K, the effective resolution of input quantization can be controlled.

Thus the GMNN can be defined to consist essentially of a memory layer (divided into K units), an address-generating unit, which generates K addresses for any $\mathbf{x} \in \Omega$, and an output unit combining the contents of the selected locations into the final network response (Fig. 3 illustrates the GMNN architecture). The selected memory weights are normally modulated by the network's basis functions, although the basis functions of a GMNN do not have to be necessarily 'graded' (i.e., variable within their support), but can also be 'flat' (i.e., constant—for example, equal to one—within their region of support). Direct correspondence to CMAC mapping can be clearly seen.

One of the main properties of the GMNN architecture is that, even in the simple case of 'flat' basis functions (which is also characteristic of the CMAC network), it is possible to identify a set of (alternative) basis functions associated with the network, where the basis functions are 'graded' (although they may have a piecewise-constant form) and are inherently dependent on the type of quantization performed by the network.

To understand the nature of this alternative basis-function description, let us first examine the operation of a GMNN when the basis functions—according to their original definition—are 'flat'. In such a case, any input to the network results in a selection of K 'active' quantization regions (one for each network layer, or node), which in turn point at K distinct locations in the network's memory layer. The contents of the selected memory locations are subsequently summed (or averaged) to yield the overall network output

$$g(\mathbf{x}) = \sum_{k=1}^{K} w_k(\mathbf{x}) \qquad \text{or} \qquad g(\mathbf{x}) = \frac{1}{K}\sum_{k=1}^{K} w_k(\mathbf{x}) \tag{2}$$

where $w_k(\mathbf{x})$ denotes the weight selected in the kth memory node by $\mathbf{x}$. With certain modifications to the format of the memory contents, other forms of network response are also possible—see (Kołcz and Allinson, 1995) for details. It can be seen that the K network quantization nodes represent in fact K address generating units, which select appropriate memory addresses for the given network input. In this form, the GMNN can be interpreted as a generalization of a look-up table, where K locations are accessed at a time, instead of just one. More flexibility is introduced when the basis functions are 'graded'. In such a case the memory weights are additionally modulated by the basis functions, which leads to network responses of the form

$$g(\mathbf{x}) = \sum_{k=1}^{K} w_k(\mathbf{x})\mu_k(\mathbf{x})$$

or

$$g(\mathbf{x}) = \frac{\sum_{k=1}^{K} w_k(\mathbf{x})\mu_k(\mathbf{x})}{\sum_{k=1}^{K} \mu_k(\mathbf{x})} \tag{3}$$

where $\mu_k(\mathbf{x})$ denotes the basis function associated with the cell selected by $\mathbf{x}$ at node k. Note that each cell can have a distinct basis, so not only the value but also the functional form of $\mu_k(\mathbf{x})$ is dependent on $\mathbf{x}$. To avoid confusion between various interpretations of basis functions discussed in this paper, the basis functions associated with the quantization cells will be termed activation functions. Note that as long the address-generation part of the network mapping is fixed during its operation, the network response (as defined by Eqs. (2) and (3)) is linear with respect to the adjustable parameters (weights), with clear advantages as far as network training (i.e., minimization of the error function $\pi(f, g)$) is concerned.

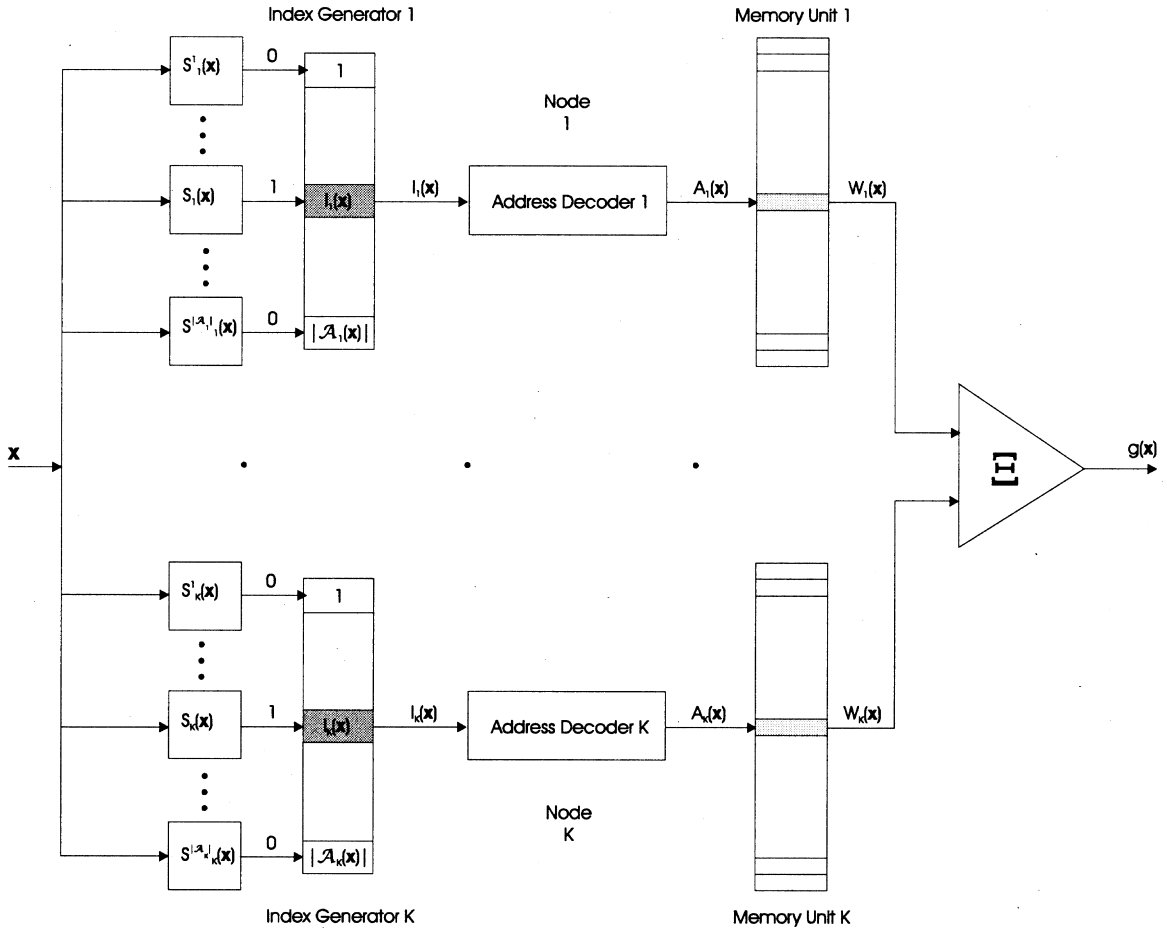

Fig. 3. GMNN architecture: steps involved in the network mapping are indicated. The networks consists of K quantization nodes, each having a number of cell-selectors (depicted as the $S(\mathbf{x})$-blocks). In each case, the quanitzation cell selected by the input is assigned a unique identifier (depicted as $I(\mathbf{x})$), which is then converted to a memory address (depicted as $A(\mathbf{x})$) used to access the contents (depicted as $W(\mathbf{x})$) of the module's memory. An equivalent process is carried out for each quantization module of the architecture and the contents of the addressed memory locations are combined to produce the network output.

The response of a GMNN is inherently local, as it involves only the K activation functions that are non-zero for the current evaluation point. The region of the input space given by the union of the support regions of the K activation functions effectively provides a neighbourhood of influence, $\mathcal{N}(\mathbf{x})$, for a given input point, $\mathbf{x}$—that is, the network response of $\mathbf{x}$ can be affected only by the points in Ω that lie within $\mathcal{N}(\mathbf{x})$. Alternatively, it can be said that only those input points that share at least one quantization cell with $\mathbf{x}$ can have their response related to $g(\mathbf{x})$. This property can be quantified by the address distance $\rho(\mathbf{x}, \mathbf{y})$ and proximity $\kappa(\mathbf{x}, \mathbf{y})$ functions, which are equal to the number of distinct and identical addresses, respectively, generated for the arguments $\mathbf{x}$ and $\mathbf{y}$.

Let $A_k(\mathbf{x})$ denote the address selected for $\mathbf{x}$ in the kth memory node (corresponding to the kth quantization layer). The address distance $\rho(\mathbf{x}, \mathbf{y})$ is defined as (ρ: $\Omega^2 \rightarrow \{0,...,K\}$)

$$\rho(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{K} [A_k(\mathbf{x}) \neq A_k(\mathbf{y})]. \tag{4}$$

The address proximity, $\kappa(\mathbf{x}, \mathbf{y})$, between $\mathbf{x}$ and $\mathbf{y}$ is given by the number of nodal addresses they share (κ:$\Omega^2 \rightarrow \{0,...,K\}$)

$$\kappa(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{K} [A_k(\mathbf{x}) = A_k(\mathbf{y})] = K - \rho(\mathbf{x}, \mathbf{y}). \tag{5}$$

Note that the address proximity function (proximity function, for short) quantifies only the *potential* similarity between the network response at $\mathbf{x}$ and $\mathbf{y}$, as it does not take the memory weights into account. It was shown (Kołcz and Allinson, 1995) that under a general class of error-correcting learning rules (including the popular LMS algorithm (Widrow and Stearns, 1985)) the response of a trained GMNN can be expressed as

$$g(\mathbf{x}) = \sum_{t=1}^{T} \Delta^t \kappa(\mathbf{x}, \mathbf{x}^t) \tag{6}$$

where the parameter set $\{\Delta^t\}_1^T$ is determined during network training. In particular, the analysis presented in (Kołcz and Allinson, 1995) considers an iterative update rule where at each (e.g., jth) iteration step one (say tth) training point is

presented to the network, and all (K) weights contributing to the network output for this input are subsequently updated by the same amount:

$$w_k \leftarrow w_k + \Delta_j^t \tag{7}$$

where Δ_j^t is proportional to the instantaneous error produced by the network for the jth presentation of the tth training point. As shown in (Kołcz and Allinson, 1995), once the training process is complete, the basis function parameters $\{\Delta^t\}_1^T$ of Eq. (6) are given by

$$\Delta^t = \sum_{j=1}^{T_j} \Delta_j^t \tag{8}$$

where T_j is the number of times the tth training point has been presented to the network during the learning process.

Eq. (6) is equivalent to the response of an RBF network, where the basis functions are associated with the training-set points and their 'heights' are given by $\{\Delta^t\}_1^T$. Unlike most implementations of an RBF network, however, the basis functions here are position dependent and piecewise constant.

2.2.1. Equivalent representations of the GMNN

In the discussion so far, the formulae describing the GMNN response have been expressed with the emphasis on the fact that exactly K memory locations are active during each such computation. However, an alternative (and fully equivalent) description is achieved when all memory locations are accounted for explicitly. In particular, let $a_i(\mathbf{x})$ denote the activation function associated with the ith ($i = 1,...,M$) memory location, such that $a_i(\mathbf{x}) = 1$ if the ith location is selected by the input $\mathbf{x}$. Otherwise $a_i(\mathbf{x})$ is equal to 0. M represents the total number of addressable locations of a GMNN. Note that the activation functions $\mu_k(\mathbf{x})$ of Eq. (3) represent a generalization over the functions $a_i(\mathbf{x})$, as they are allowed to vary within their activation regions.

Using this notation, the two network output formulas of Eq. (2) can be equivalently expressed as

$$g(\mathbf{x}) = \sum_{i=1}^{M} w_i \cdot a_i(\mathbf{x}) \qquad \text{and} \qquad g(\mathbf{x}) = \frac{\sum_{i=1}^{M} w_i \cdot a_i(\mathbf{x})}{\sum_{i=1}^{M} a_i(\mathbf{x})} \tag{9}$$

where w_i is the weight associated with the ith memory location. Thus the network output is given as a linear (in the parameters) expansion in terms of the activation functions associated with the memory. In the simplest case given above, the activation functions are binary, but they can have other fucntional forms (and become equivalent with the $\mu_k(\mathbf{x})$ functions introduced by Eq. (3)). Of course, only K of the M activation functions in Eq. (9) are nonzero for any particular input. Note that under this presentation, the address proximity $\kappa(\mathbf{x}, \mathbf{y})$, between $\mathbf{x}$ and $\mathbf{y}$ can be expressed as a scalar product of the activation vectors $[a_1(\mathbf{x}),...,a_M(\mathbf{x})]$ and $[a_1(\mathbf{y}),...,a_M(\mathbf{y})]$, i.e.,

$$\kappa(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{M} a_i(\mathbf{x}) \cdot a_i(\mathbf{y}). \tag{10}$$

For the special case of the N-tuple network (Bledsoe and Browning, 1959; Tattersall et al., 1991) it has been demonstrated how a certain modification of the network memory architecture (and modifications to the training regime) enable a normalization of the GMNN response with respect to the basic set, thus making it possible to achieve responses analogous to the normalized RBF and KR networks (Kołcz and Allinson, 1996). Here, we use the alternative GMNN representation to prove this result for the general case. Let us assume that the ith addressable location contains two fields w_i and c_i. During a one-pass training procedure (where each training point is presented to the network in turn), the training set $\{\mathbf{x}^t, y^t\}_1^T$ is used to update the addressed locations in the following way

$$w_i = w_i + y^t \cdot a_i(\mathbf{x}^t) \tag{11}$$

$$c_i = c_i + a_i(\mathbf{x}^t)$$

for $i = 1,...,M$. Thus after the training is finished, the w_i and c_i fields of the ith memory cell have their values equal to:

$$w_i = \sum_{t=1}^{T} y^t \cdot a_i(\mathbf{x}^t) \tag{12}$$

$$c_i = \sum_{t=1}^{T} a_i(\mathbf{x}^t)$$

respectively. The output of this modified network architecture is then defined as

$$g(\mathbf{x}) = \frac{\sum_{i=1}^{M} w_i \cdot a_i(\mathbf{x})}{\sum_{i=1}^{M} c_i \cdot a_i(\mathbf{x})} \tag{13}$$

which, using Eq. (12) and Eq. (10), can be expanded to

$$\begin{aligned} g(\mathbf{x}) &= \frac{\sum_{i=1}^{M} \left(\sum_{t=1}^{T} y^t \cdot a_i(\mathbf{x}^t) \right) \cdot a_i(\mathbf{x})}{\sum_{i=1}^{M} \left(\sum_{t=1}^{T} a_i(\mathbf{x}^t) \right) \cdot a_i(\mathbf{x})} \\ &= \frac{\sum_{t=1}^{T} y^t \sum_{i=1}^{M} a_i(\mathbf{x}^t) \cdot a_i(\mathbf{x})}{\sum_{t=1}^{T} \sum_{i=1}^{M} a_i(\mathbf{x}^t) \cdot a_i(\mathbf{x})} \\ &= \frac{\sum_{t=1}^{T} y^t \kappa(\mathbf{x}, \mathbf{x}^t)}{\sum_{t=1}^{T} \kappa(\mathbf{x}, \mathbf{x}^t)}. \end{aligned} \tag{14}$$

Hence, the GMNN is in this case output-equivalent with a kernel regression network (Härdle, 1990), with the proximity kernel, κ, as the basis function. In a way analogous to the kernel-regression network, the introduction of counter (i.e., c_i) fields to the GMNN memory cells (as explained above) allows us to normalize the response of a GMNN network trained with the NLMS algorithm from the form given by Eq. (6) to

$$g(\mathbf{x}) = \frac{\sum_{t=1}^{T} \Delta^t \kappa(\mathbf{x}, \mathbf{x}^t)}{\sum_{t=1}^{T} \kappa(\mathbf{x}, \mathbf{x}^t)} \tag{15}$$

when the network response is modified to the one given by Eq. (13) and the counter fields are set according to Eq. (11) (note that only one pass through the training set is necessary to set the counter fields). Extensions of these results to the case of GMNN architectures with arbitrary (i.e., 'graded') activation functions are presented in (Kołcz, 1996).

The advantage of the fully expanded representation is that we can treat the GMNN as a basis function network (although the centres of the basis functions $\{a_i\}_1^M$ do not correspond to the training set points as is often the case for the RBFs) and apply its standard formalism, for example to solve the least squares problem (Girosi et al., 1995). On the other hand, in practical computations, direct manipulation of the memory vector may lead to very large (and very sparse) systems of equations due to the large address space of a GMNN. An implementation of a GMNN (e.g., the CMAC network) may not physically realize all its addressable memory cells, and hashing techniques are commonly applied to manage the memory efficiently.

It can be seen that it is possible to build and analyse basis-function models of GMNN architectures, with the basis set given by $\{\kappa(\mathbf{x}, \mathbf{x}^t)\}_1^T$. This approach would be quite difficult, however, as the proximity functions do not have a clear functional form and are realized in a GMNN inherently, as a result of the *K*-fold quantization performed by the network. We suggest, however, that in certain cases, where the quantization performed by a GMNN is particularly regular, it makes sense to attempt to derive an expected form of the proximity function, which could be further used to gain insight into the operation of the network and perhaps to create an approximate basis-function model of it. In the remainder of this paper, we derive such an approximate basis for the CMAC network and assess its utility in predicting CMAC performance.

3. The structure of CMAC quantization

Each of the *K* nodes of a CMAC network performs a variant of scalar-product (vector) quantization of the input space—that is, each of the *D* components of an input vector is quantized individually, which results in hyper-rectangular quantization cells oriented along the coordinate axes in $\Re^D$. Each of the *K* vector quantizers has *D* scalar components (one per input-vector dimension), and conversely, each coordinate of an input vector is quantized by *K* scalar quantizers (one per network node). The interdependencies between the *K* scalar quantizers of the CMAC network are common across all *D* input coordinates, hence the scalar-input case ($D = 1$) will be treated first.

3.1. The scalar case ($D = 1$)

According to the constraints of the GMNN architecture, no two nodal quantization cells of the network should be identical. Limited to the scalar case, none of the quantization intervals produced by the *K* scalar quantizers should be the same—that is, at any input point, the quantization intervals to which *x* belongs in the individual quantization layers overlap, but no two cells belonging to different layers can be exactly the same. Let [0, *R*] denote the input domain (i.e., Ω). In order to provide a clear explanation of CMAC quantization we will define a (finite) sequence of threshold points within Ω

$$0 = t_1, t_2, ..., t_N = R \tag{16}$$

which specify the maximum desirable resolution of the overall network quantization. A further constraint requires that, at any layer, border points between quantization cells can occur only at the positions determined by the threshold points $\{t_i\}_1^N$. These constraints still leave much flexibility as to how the individual *K* quantizers of the network are designed (e.g., the cells of a single quantization layer can vary in length, and not all thresholds have to be utilized as inter-cell border points). CMAC quantization results if we additionally require that each quantizer partitions Ω into cells that extend for exactly *K* inter-threshold intervals (with the possible exception of boundary cells, which may be truncated). As a result, all threshold points are utilized (thus leading to the maximum desired quantization resolution) and the quantization-cell arrangements produce a characteristic staggered pattern, which is particularly clear if all quantizers are uniform (which is the case when $t_i - t_{i-1} = \text{const}$). Fig. 4 illustrates scalar CMAC quantization for the case of $K = 4$ and non-uniform placement of thresholds. It should be mentioned that an identical type of scalar quantization is used by B-spline networks (where *K* corresponds to the spline order) (Lane et al., 1992), although the quantization methods of CMAC and tensor-product B-splines (Bartels et al., 1987) differ for $D > 1$.

In the following discussion we will consider the case of uniform CMAC quantization, where $\{t_i\}_1^N$ define a uniform partition of Ω into $N - 1$ equal length unit intervals, i.e.,

$$t_2 - t_1 = t_3 - t_2 = ... = t_N - t_{N-1} = o = \text{const} \tag{17}$$

whereas the individual CMAC quantizers have cells of length $U = Ko$. As the input space is given by an interval of finite length, each quantizer will have a finite range,

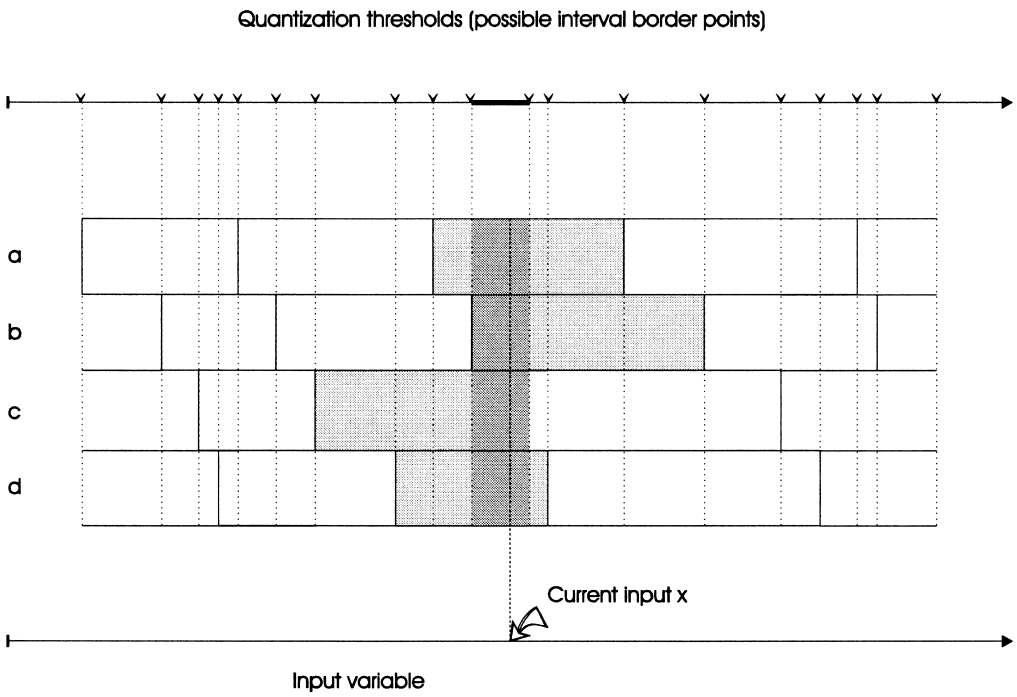

Fig. 4. Illustration on non-uniform scalar CMAC quantization for the case of $K = 4$ (individual quantization layers are labelled as: a, b, c, d). A border between quantization intervals at a given threshold position is allowed for one layer only (see text for details).

which (without loss of generality) will be assumed to be $0,...,L-1$ for all scalar quantizers. In the uniform case, the staggered arrangement of CMAC quantizers is particularly clear. In fact, given one of the quantizers, the remaining K-1 quantizers can be obtained by translating the original quantization-cell arrangement by o, $2o,...,(K-1)o$. Furthermore, starting with any point $x \in \Omega$, increments (or decrements) of Δx by o at a time will cause exactly one of the quantizers (at each step) to change its output for $x + \Delta x$ with respect to the values produced for x. Hence, for $\Delta x \geq Ko = U$, the points x and $x + \Delta x$ will share no quantization cells. The following table provides an example of the outputs produced by CMAC quantizers ($K = 4$) for consecutive, equally spaced points, with the interpoint distance equal to o.

	q_1	q_2	q_3	q_4
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	2	1	1	1
6	2	2	1	1
7	2	2	2	1
8	2	2	2	2
9	3	2	2	2
10	3	3	2	2

(18)

The first column indexes the displacements (in multiples of o) from the starting point, assumed here to be 0.

3.2. The multidimensional case ($D > 1$)

When $D > 1$, the quantization described for the one-dimensional case is performed in a similar manner for each component of the input vector. Each of the K vector quantizers of the network now consists of D individual scalar-quantizer components, one per each input dimension. As can be seen, this design guarantees that no two D-dimensional quantization cells are the same and, at the same time, that any two cells belonging to different quantization layers will differ additionally in the sense that none of their D projections on the coordinate axes will be the same (irrespective of whether the cells overlap or not). Fig. 5 illustrates CMAC quantization in a 2-dimensional case for $K = 4$.

Depending on the particular case, the overall resolution obtained for individual dimensions of Ω may vary. Also the relative order of staggered quantizers can be different for different dimensions. It has been proposed (Parks and Militzer, 1991) that the latter flexibility is used in order to optimize certain topological properties of CMAC mapping, and this will be discussed later.

A natural way to describe CMAC vector quantization of Ω is by representing the K network quantizers with a D-by-K matrix, $\mathbf{Q}$, where the dth row of $\mathbf{Q}$ represents the outputs of the K scalar quantizers corresponding to the dth dimension. Conversely, the kth column of $\mathbf{Q}$ represents the kth network quantizer and its D scalar components. Thus, the kth column of $\mathbf{Q}$ will be identical for points in the same quantization cell at the kth

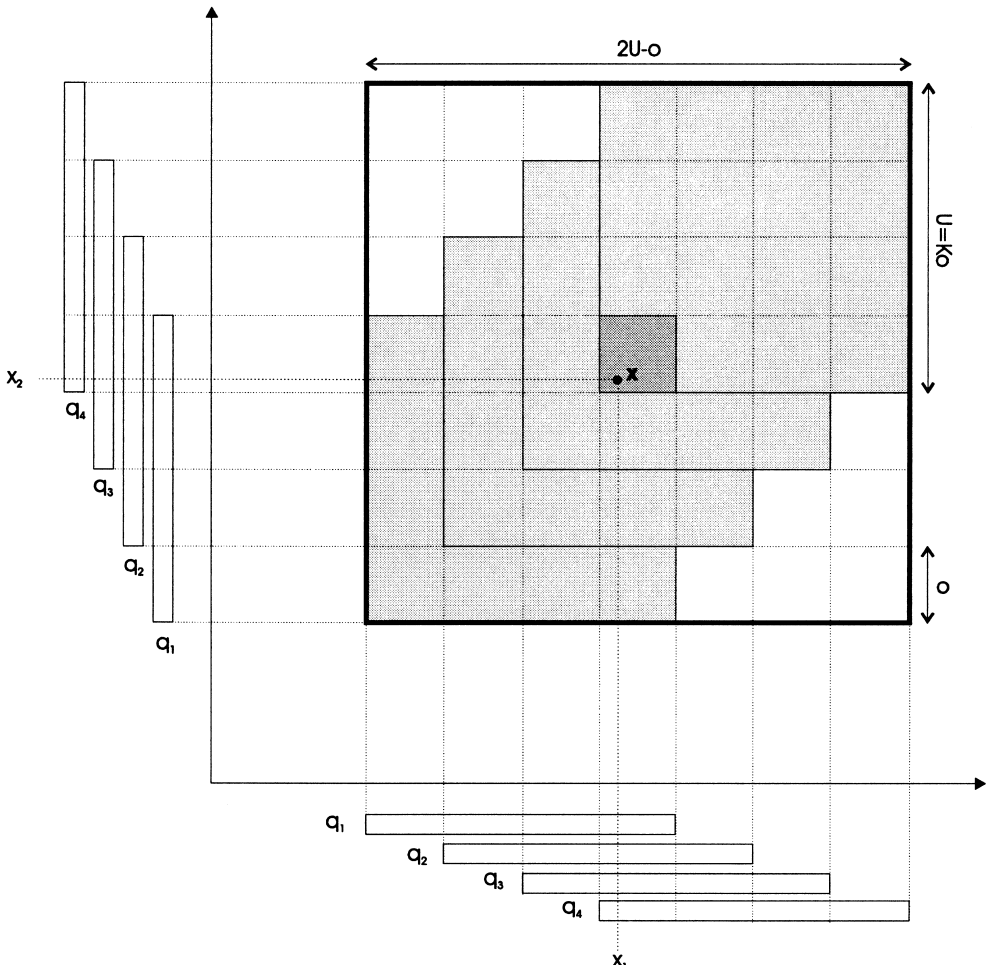

Fig. 5. Illustration of the spatial arrangement of overlapping neigbourhoods for the VQ performed by CMAC in a 2-dimensional case. The neighbourhood of influence for x is here given by a square of side $2U - o$.

network node (layer). The dth row, on the other hand, will be constant for points whose projections on the dth coordinate axis lie in the same (scalar) global-level quantization interval. For any input point, $\mathbf{x} = [x_1,...,x_D]$, the quantization matrix is given by

$$[x_1, ...x_D] \rightarrow \mathbf{Q}(\mathbf{x}) = \begin{bmatrix} q_1(x_1) & ... & q_K(x_1) \\ q_1(x_2) & ... & q_K(x_2) \\ . & ... & . \\ q_1(x_D) & ... & q_K(x_D) \end{bmatrix} \quad (19)$$

One of our main objectives is to analyse how the address distance of the CMAC network changes with respect to the input space distance. Of course, such dependence is highly local, especially in the D-dimensional case. However, we will attempt to estimate the expected dependence of ρ on the input-space distance. Such an approach is viable considering the rather regular quantization performed by CMAC at each node and makes sense especially in the case where all CMAC quantizers are uniform (which is favoured in practical applications due to ease of implementation).

As a consequence of scalar uniform quantization, each of the individual CMAC quantizers partitions the input space into hypercubical cells (of side U), whereas superimposing the K quantizations layers leads to a uniform quantization of Ω into hypercubical cells of side o. Therefore, both the individual quantizers and the global-level quantization can be considered as (truncated) lattice quantizers (Gersho and Gray, 1992), with the lattice points given by the quantization-cell vertices. Figure 6 provides an illustration of the lattice structure of CMAC quantization. The square grids corresponding to individual quantizers are given in different shades of grey, for better visualization. The overlapping structure of cells is clearly indicated. The global-level cells corresponding to top-left corners of cells belonging to individual layers are also shown. The characteristic placement of these points along diagonals in the input space can be seen. The particular pattern is typical for the original form of CMAC encoding, as introduced in Albus (1975b), and is the result of preserving the relative ordering of staggered scalar quantizers across all input coordinates.

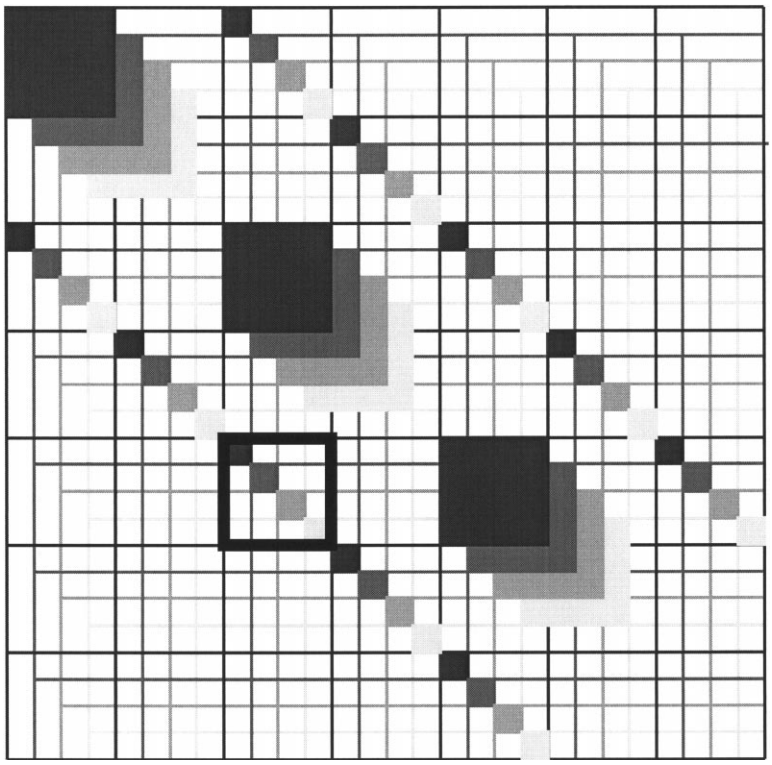

Fig. 6. Illustration of the uniform CMAC quantization scheme (in the standard version) for the 2-dimensional case, with $K = 4$. The overlapping square quantization lattices, corresponding to the individual network nodes, are drawn in different shades of grey for clarity. Characteristically for standard CMAC encoding, the top-left corners of nodal quantization cells are positioned diagonally in the input space, as indicated. The outlined square corresponds to the generating hypercube of the quantization lattice.

As suggested in Parks and Militzer (1991), a more uniform coverage of the input space takes place when the relative ordering of scalar CMAC quantizers is different for different dimensions. We consider these modifications here as they become significant later in this paper. To be able to see the available options of CMAC encoding, note that the global-level CMAC quantization lattice can be obtained by tiling an elementary (or generating) hypercube of side U (indicated by the outlined square in Figure 6). The generating hypercube itself contains K^D global-level quantization cells, K of which correspond to the chosen reference vertices (top-left corners in this particular 2-dimensional example) of the nodal quantization cells. The positions of the reference vertices inside the generating hypercube can be expressed as integer coordinates in the range $0,...,K-1$, and because of the staggered nature of CMAC scalar quantization, all D coordinates of any two vertices must be different. Using such notation, the vertex coordinates corresponding to the original CMAC code are given by the following matrix ($K = 4, D = 4$), where rows correspond to the vertices:

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \end{bmatrix} \tag{20}$$

Any other relative positioning of scalar quantizers can be obtained by reordering (i.e., performing a permutation) the elements of columns in Eq. (20) (where each column can be treated independently), with the total number of options equal to $(K!)^D$. For example, an alternative to Eq. (20) might have the form of

$$\begin{bmatrix} 0 & 3 & 0 & 3 \\ 1 & 1 & 2 & 2 \\ 2 & 0 & 1 & 1 \\ 3 & 2 & 3 & 3 \end{bmatrix} \tag{21}$$

Generally, given a particular optimality criterion and the value of D and K, all such matrices could be analysed by exhaustive search, where, due to the natural symmetry of this problem, many distinct permutations lead to identical matrices, thus reducing the actual number of options. A variant of this approach was considered in Parks and Militzer (1991), where the number of possibilities was additionally reduced by allowing vertices to be positioned only at points whose coordinates are generated by

$$\mathbf{c}^k = [(k-1)\delta_1 \bmod K, ..., (k-1)\delta_D \bmod K] \tag{22}$$

(for $k = 1,...,K$), where $\{\delta_1,...,\delta_D: \delta_d \in \{1,...,K\}\}$ represent a set of integers relatively prime to K and $\mathbf{c}^k$ denotes

the vertex corresponding to the kth quantization layer. Let P denote the number of positive integers belonging to $\{1,...,K\}$ and relatively prime to K. Then, the number of possible choices of $\{\delta_1,...,\delta_D\}$, considering that the order of elements inside this set is irrelevant, is $P^D/D!$, which is considerably less than the number of all possible generating matrices—that is, $(K!)^D$. Results obtained using this method for a range of D and K parameters have been published in Parks and Militzer (1991).

4. Derivation of CMAC address distance and proximity functions

It can be seen that addresses generated by CMAC for two input points, **x** and **y**, will be determined by the quantization matrices **Q(x)** and **Q(y)** produced for these two points. In particular, the kth components of the address vector generated for **x** and **y** will be identical as long the kth columns of **Q(x)** and **Q(y)** are the same. On the other hand, if the kth network quantizer produces different outputs for **x** and **y** for at least one dimension, then **Q(x)** and **Q(y)**—and hence $A(\mathbf{x})$ and $A(\mathbf{y})$—will be different. To facilitate further analysis we will introduce a 'change' matrix, **C(x, y)**, which quantifies the relationship between **Q(x)** and **Q(y)**. Namely, the c_{dk} component of **C** is equal to 1 if the corresponding components of **Q(x)** and **Q(y)** are different, and is 0 otherwise, i.e.

$$c_{dk} = [q_k(x_d) \neq q_k(x_d + \Delta x_d)]. \tag{23}$$

The number of '1's in **C** determines the number of different components in **Q(x)** and **Q(y)**, and can be considered as a generalized Hamming distance between these two matrices. It can be seen that the address proximity between **x** and **y** is equal to the number of columns of **C** that contain no '1's.

4.1. The CMAC address distance

Let us consider the behaviour of $\rho(\mathbf{x}, \mathbf{y} = \mathbf{x} + \Delta\mathbf{x})$ in the case where the offset vector, $\Delta\mathbf{x}$, contains just one non-zero component

$$\Delta\mathbf{x} = [..., 0, \Delta x_d, 0, ...] \tag{24}$$

that is, when the vectors **x** and **y** differ at one coordinate only. Considering the nature of CMAC quantization, all K scalar quantizers of the dth dimension will produce the same output as long as both x_d and $y_d = x_d + \Delta x_d$ lie in the same global-level quantization interval of length o. Depending on the position of x_d in the global-level quantization interval and on the sign of x_d, the point $x_d + \Delta x_d$ will lie in a different global-level quantization interval for some $|\Delta x_d| < o$ (with one of the nodal quantizers changing its output for y_d). However, a change of the global quantization interval will occur for $|\Delta x_d| = o$, irrespective of the initial position of x_d. Therefore, increments of $|\Delta x_d|$ in steps of o (starting with $|\Delta x_d| = 0$) will increment ρ in units of 1, until the saturation value, K, is reached, which corresponds to all K quantizers producing different outputs for x_d and y_d. This behaviour can be quantified by

$$\rho(x_d, x_d + \Delta x_d) = \rho(x_d) = \lfloor |\Delta x_d|/o \rfloor_K$$
$$= \begin{cases} \lfloor |\Delta x_d|/o \rfloor \text{ if } \lfloor |\Delta x_d|/o \rfloor < K \\ K \text{ otherwise} \end{cases} \tag{25}$$

where $\lfloor |x_d|/o \rfloor$ represents a quantized version of the offset Δx_d, and $\lfloor \cdot \rfloor$ is the 'floor' operator. For any starting position, x_d, the address distance becomes a function of the offset Δx_d (or more exactly its absolute value), and generally (for $D > 1$), it is a function of the vector displacement $\Delta\mathbf{x}$. In order to simplify the notation, but also to emphasise the dependence of $\rho(\mathbf{x}, \mathbf{x} + \Delta\mathbf{x})$ on $\Delta\mathbf{x}$, we will denote $\rho(\mathbf{x}, \mathbf{x} + \Delta\mathbf{x})$ by $\rho(\Delta\mathbf{x})$.

Let us now consider a more complex case when $\Delta\mathbf{x}$ has exactly two non-zero components (say corresponding to x_1 and x_2). In this case only two rows of **C** can be affected. To facilitate the analysis, let $Q(\Delta x_d)$ designate the set of quantizers for which $q_k(x_d + \Delta x_d) \neq q_k(x_d)$. The relationship between the elements of **C** as a function of the spatial displacement, $\Delta\mathbf{x}$, can be described by three individual cases (the zero entries in **C** are omitted for clarity):

1. $\rho(x_1) = \rho(x_2) = \alpha$ and $Q(\Delta x_1) = Q(\Delta x_2)$. Consequently, address changes occur at precisely the same positions and

$$\alpha = \rho(\Delta\mathbf{x}) < \rho(\Delta x_1) + \rho(\Delta x_2) = 2\alpha. \tag{26}$$

The table below illustrates this case when $K = 6$ and $\alpha = 4$. The changes due to Δx_1 and Δx_2 are indicated by '1's in the columns corresponding to the affected quantizers.

$$\begin{array}{c|cccccc} & Q_1 & Q_2 & Q_3 & Q_4 & Q_5 & Q_6 \\ \Delta x_1 \rightarrow & 1 & 1 & & 1 & & 1 \\ \Delta x_2 \rightarrow & 1 & 1 & & 1 & & 1 \end{array}. \tag{27}$$

2. $\rho(\Delta x_1) = \alpha$ and $\rho(\Delta x_2) = \beta$ and $Q(\Delta x_1) \cap Q(\Delta x_2) = \varnothing$. The address changes fully complement each other leading to

$$\rho(\Delta\mathbf{x}) = \rho(\Delta x_1) + \rho(x_2) = \alpha + \beta. \tag{28}$$

The table below provides an example of this case for $\alpha = 2$, $\beta = 4$ and $K = 6$.

$$\begin{array}{c|cccccc} & Q_1 & Q_2 & Q_3 & Q_4 & Q_5 & Q_6 \\ \Delta x_1 \rightarrow & 1 & 1 & & & & \\ \Delta x_2 \rightarrow & & & 1 & 1 & 1 & 1 \end{array}. \tag{29}$$

3. A combination of cases 1 and 2, where there is a partial overlap between the sets $Q(\Delta x_1)$ and $Q(\Delta x_2)$. It follows that

$$\max(\alpha, \beta) < \rho(\Delta\mathbf{x}) < \alpha + \beta$$

In the illustration below $\alpha = 4$, $\beta = 3$ and $K = 6$.

$$\begin{array}{c|cccccc} & Q_1 & Q_2 & Q_3 & Q_4 & Q_5 & Q_6 \\ \Delta x_1 \rightarrow & 1 & 1 & & 1 & 1 & \\ \Delta x_2 \rightarrow & & 1 & & 1 & 1 & 1 \end{array}. \tag{30}$$

It is clear that this analysis extends naturally to the case of

an arbitrary $\Delta\mathbf{x}$, and generally

$$\max_d(\rho(\Delta x_d)) \leq \rho(\Delta\mathbf{x}) \leq \sum_d \rho(\Delta x_d). \tag{31}$$

Therefore, $\rho(\mathbf{x}, \mathbf{x} + \Delta\mathbf{x})$ has its values located between the quantized versions of the $\|\Delta\mathbf{x}\|_\infty$ and $\|\Delta\mathbf{x}\|_1$ norms (with the saturation effects taken into account)

$$\max_d(\lfloor|\Delta\mathbf{x}_d|/o\rfloor_K) \leq \rho(\Delta\mathbf{x}) \leq \sum_d \lfloor|\Delta\mathbf{x}_d|/o\rfloor_K. \tag{32}$$

From the above analysis it is obvious that if for any coordinate $|\Delta x_d| \geq Ko$ then $\rho(\mathbf{x}, \mathbf{x} + \Delta\mathbf{x})$ becomes saturated to K, irrespective of the other components of the displacement vector. Therefore, we will concentrate on the case where none of the scalar displacements leads to a saturation of ρ, i.e.,

$$\rho(\Delta x_d) = \alpha_d < K \qquad d = 1, ..., D. \tag{33}$$

Let us also initially make an assumption that the combined effect of all components of $\Delta\mathbf{x}$ does not lead to a saturation of ρ, i.e.,

$$\sigma = \sum_{d=1}^{D} \alpha_d < K \tag{34}$$

where δ denotes the quantized norm $|\Delta\mathbf{x}|$. In this case some columns of $\mathbf{C}$ will consist of '0' entries only, and for certain offset vectors it is possible that there will be exactly one '1' per occupied column of $\mathbf{C}$. (An occupied column of $\mathbf{C}$ contains at least one '1'.) Let $\alpha_{\max}$ denote the maximum scalar distance, i.e.,

$$\alpha_{\max} = \max_d \alpha_d \tag{35}$$

with the corresponding dimension denoted by $d_{\max}$. Consequently, the possible range of ρ is limited to

$$\alpha_{\max} \leq \rho(\Delta\mathbf{x}) \leq \sigma. \tag{36}$$

In the following derivation we will assume that rows of $\mathbf{C}$ are mutually uncorrelated (on average), and that, for each dimension, the '1' components can be distributed in the corresponding row of $\mathbf{C}$ with equal probability. Let us consider the effects of the scalar components of $\Delta\mathbf{x}$ on $\mathbf{C}$ in a sequence, with $\Delta x_{d\max}$ taking effect first. Thus, after the first step, exactly $\alpha_{\max}$ columns of $\mathbf{C}$ are occupied and the remaining set of 'empty' columns will be denoted by $C = \{C_1, ..., C_{K-d_{\max}}\}$. The next (say dth) considered scalar offset, Δx_d, sets α_d '1's among the elements of the dth row, with the probability of changing the 'empty' (i.e., containing no '1's) status of any particular element of C given by

$$1 - \frac{\binom{K-1}{\alpha_d}}{\binom{K}{\alpha_d}} = \frac{\alpha_d}{K}. \tag{37}$$

Hence, when all elements of $\Delta\mathbf{x}$ are considered, the probability that a particular element of C will become occupied (by a '1') is given by

$$\sum_{\left(\substack{d=1\\ d\neq d_{\max}}\right)}^{D} \frac{\alpha_d}{K} = \frac{\sigma - \alpha_{\max}}{K} < 1. \tag{38}$$

Eq. (38) above can, in fact, be considered as an approximation to the formula of the probability that the column is going to be affected by at least one component of $\Delta\mathbf{x}$, which is given by

$$1 - \prod_{\left(\substack{d=1\\ d\neq d_{\max}}\right)}^{D} \left(1 - \frac{\alpha_d}{K}\right) = \sum_{\left(\substack{d=1\\ d\neq d_{\max}}\right)}^{D} \frac{\alpha_d}{K} \tag{39}$$

$$-\left(\sum_{d_1,d_2} \frac{\alpha_{d_1}\alpha_{d_2}}{K^2} - \sum_{d_1,d_2,d_3} \frac{\alpha_{d_1}\alpha_{d_2}\alpha_{d_3}}{K^3} ... \pm \prod_{\left(\substack{d=1\\ d\neq d_{\max}}\right)}^{D} \frac{\alpha_d}{K}\right).$$

As can be seen, when $\alpha_d/K \ll 1$, the higher terms of Eq. (39) can be ignored, thus leading to formula (38).

Considering that all elements of C have the same probability of becoming occupied (and can be treated independently from one another), and by using Eq. (38), the expected number of occupied columns is given by

$$(K - \alpha_{\max})\frac{\sigma - \alpha_{\max}}{K} \tag{40}$$

which leads to the expected value of ρ (given $\alpha_{\max}$ and σ) to have the form of

$$\begin{aligned} E[\rho(\sigma, \alpha_{\max})] &= \alpha_{\max} + (K - \alpha_{\max})\frac{\sigma - \alpha_{\max}}{K} \\ &= \sigma - \frac{\alpha_{\max}(\sigma - \alpha_{\max})}{K}. \end{aligned} \tag{41}$$

Given the value of σ, the maximum scalar distance, $\alpha_{\max}$, can take values in the range $s = \lceil\sigma/D\rceil, ..., \sigma$ with equal probability. Hence, the expected value of ρ, dependent on the value of σ only, can be obtained as

$$\begin{aligned} E[\rho(\sigma)] &= \frac{1}{\sigma - s + 1} \sum_{\alpha_{\max}=s}^{\sigma} E[\rho(\sigma, \alpha_{\max})] \\ &= \frac{1}{\sigma - s + 1} \sum_{\alpha_{\max}=s}^{\sigma} \left(\sigma - \frac{\alpha_{\max}(\sigma - \alpha_{\max})}{K}\right). \end{aligned} \tag{42}$$

The sum (Eq. (42)) can be split into three components, which evaluate to

$$\frac{\sigma}{\sigma - s + 1} \sum_{\alpha_{\max}=s}^{\sigma} 1 = \sigma$$

$$\frac{\sigma}{(\sigma - s + 1)K} \sum_{\alpha_{\max}=s}^{\sigma} \alpha_{\max} \frac{\sigma(\sigma + s)}{2K}$$

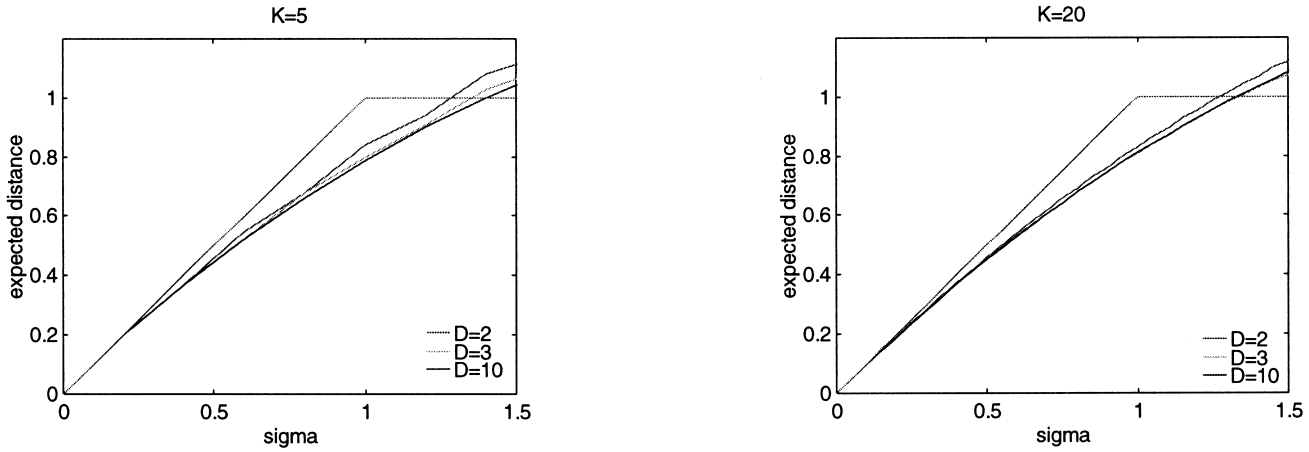

Fig. 7. The estimated CMAC address-distance function for a selection of K and D parameters. The saturated (to K) city-block distance is provided as a reference. The input and address distances are normalized with respect to K. It can be seen that the distance function converges for values of the parameter D greater than 2, and the convergence is faster for larger K (e.g. the right side of the figure).

$$\frac{\sigma}{(\sigma - s + 1)K} \sum_{\alpha_{\max}=s}^{\sigma} \alpha_{\max}^2 = \frac{\sigma(\sigma+1)(2\sigma+1) - s(s-1)(2s-1)}{6K(\sigma - s + 1)} = \frac{1}{K}\left(s\sigma + \frac{(\sigma - s)(2(\sigma - s) + 1)}{6}\right). \quad (43)$$

Hence, Eq. (42) can be expressed as

$$E[\rho(\sigma,s)] = \sigma - \frac{\sigma(\sigma+s)}{2K} + \frac{1}{K}\left(s\sigma + \frac{(\sigma - s)(2(\sigma - s) + 1)}{6}\right) = \sigma - \frac{(\sigma - s)(\sigma + 2s - 1)}{6K}. \quad (44)$$

As expected (see Eq. (32)), $E[\rho(\sigma,s)]$ depends on the quantized norms, $\|\Delta\mathbf{x}\|_\infty$ and $\|\Delta\mathbf{x}\|_1$, of the displacement vector. It can be seen that $E[\rho(\sigma,s)]$ represents a quadratic function in σ, which is especially clear in the special case of $s = 1$, when Eq. (44) reduces to

$$E[\rho(\sigma, s|s=1)] = \sigma - \frac{\sigma^2 - 1}{6K}. \quad (45)$$

By substituting $s = \lceil \sigma/D \rceil$ (where $\lceil\cdot\rceil$ represents the 'ceiling' operator) into Eq. (44) and approximating it by $\lceil \sigma/D \rceil \simeq \sigma/D + 0.5$, the following quadratic relationship is obtained

$$E[\rho(\sigma)] \simeq \frac{\sigma(6K + 0.5 + \frac{1}{D} - \sigma(1 - \frac{1}{D})(1 + \frac{2}{D}))}{6K} \quad (46)$$

where for fixed parameters K and D, $E[\rho]$ is a function (representing a parabola) of σ only. It can be seen that this parabola is positive for values of between 0 and

$$\sigma_{\max} = \frac{6K + 0.5 + \frac{1}{D}}{(1 - \frac{1}{D})(1 + \frac{2}{D})} \quad (47)$$

where $\sigma_{\max}$ is approximately equal to $6K$ when $6K \gg 1$ and $D \gg 1$. Of course, ρ can take values in the range 0,...,K only, whereas the parabola (Eq. (46)) peaks at $\sigma_{\text{peak}} \simeq 3K$ with the (approximate) maximum of $1.5K$. Elementary calculations determine the relevant range of the parameter as $0,...,\sigma_{\text{sat}} \simeq 1.3K$, in which ρ increases monotonically with σ and has its values in the range 0,...,K. If the distance-function formula (Eq. (46)) is to be used outside the range constrained by the assumptions stated in its derivation, its value should be clipped to the saturation level for $\sigma \geq \sigma_{\text{sat}}$. Figure 7 shows examples of $E[\rho(\sigma,s)]$ for $K = 5$ and 20, and $D = 2,3$, and 10. In the graphs, the saturation level (K) and the city-block distance (i.e., the $y = x$ line) are also indicated. It can be seen that, for larger values of D (i.e. $1/D \simeq 0$), the corresponding curves lie very close to each other, especially in the relevant region of $\sigma < \sigma_{\text{sat}}$, where they follow approximately the relationship

$$E[\rho(\sigma)] \simeq \frac{\sigma(6K - \sigma)}{6K} = \sigma\left(1 - \frac{\sigma}{6K}\right). \quad (48)$$

4.2. Estimation accuracy

In the derivations presented above, certain simplifying assumptions have been made about the properties of the 'change' matrix **C**. Although the assumption about the independence of element changes (from 0 to 1) occurring along the individual input variables (i.e., rows of **C**) is certainly valid, the assumption about the uniformity of distribution of the changes within each row of **C** is less accurate. Note that a row of **C** corresponds to one input variable and that there is a deterministic offset relationship

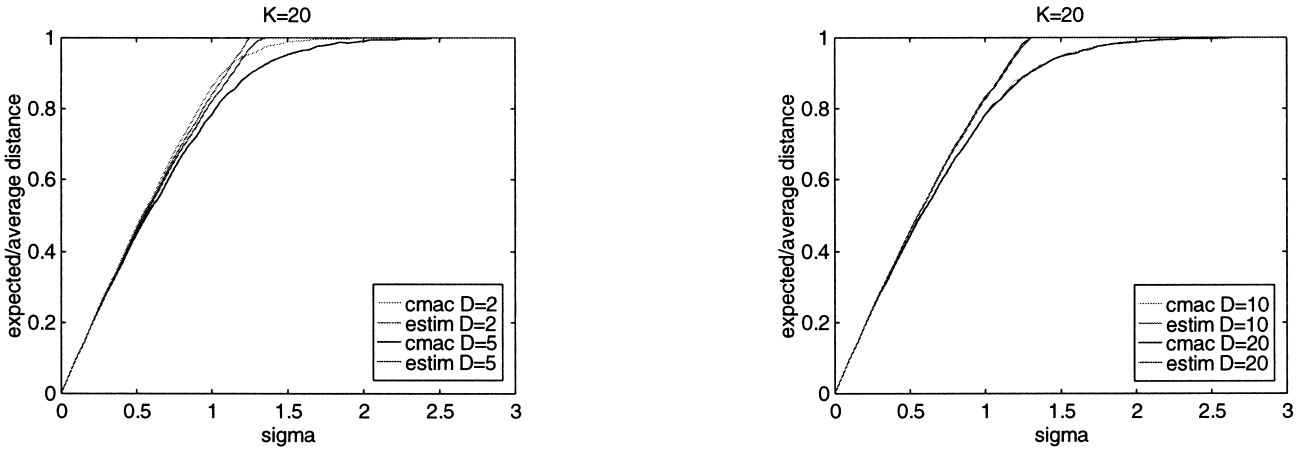

Fig. 8. The estimated and average CMAC address-distance functions are compared for $K = 20$ and a selection of the D parameter. The input and address distances are normalized with respect to K. Close correspondence between the estimates and actual (averaged) distances exists for $\sigma < K$ (i.e., normalized less than 1). For greater values of D (right side of the figure) both the expected and average distances are less sensitive to the particular values of D and tend to converge to distinct paths for $\sigma > K$.

between the scalar quantizers for each variable. Therefore, as the value of, the say dth, variable increases (decreases), elements of $\mathbf{C}$ will change from 0 to 1 in some predictable order, which depends on the initial value of x_d, the direction of change (i.e., an increase or a decrease) and the offset relationship between the scalar CMAC quantizers for the dth dimension. In fact, due to the high regularity of the original CMAC encoding, the changes occurring in this case tend to occur in the neighbouring elements of each row of $\mathbf{C}$. On the other hand, for some of the modified CMAC encodings (Parks and Militzer, 1991), the changes occurring within each row of $\mathbf{C}$ will be distributed more evenly (in a deterministic sense). Thus, it can be expected that for the modified CMAC encoding the agreement between the estimated and actual CMAC distance could be slightly better.

To compare the estimate (Eq. (46)) with the actual CMAC address distance, we performed a set of simulations where for every value of σ2000 random offset vectors (with $|\Delta\mathbf{x}| = \sigma$) were generated and the resulting address distances were averaged. As shown in Fig. 8 (for the example cases), the average CMAC address distance and its estimate, provided by Eq. (46), are in fairly good agreement, although the actual CMAC address distance saturates to K at a slower rate than the estimate. However, it should be pointed out that in the region of disagreement—that is, at the values of σ for which the saturation takes place (i.e., $\sigma \geq K$)—the assumptions taken during the derivation of Eq. (46) are no longer valid. Results obtained for other combinations of the D and K parameters were analogous to the ones presented in Fig. 8.

4.2.1. Dependency on the actual CMAC quantization

Although certain idealizing assumptions have been used to arrive at our estimate of ρ, they are not related to the particular variant of the CMAC quantization used (as long as uniformity is preserved for each scalar quantizer). In particular, our estimate does not depend on the permutations of columns in matrix $\mathbf{Q}$ (Eq. (19)), so Eq. (44) applies both to the standard CMAC encoding as well as to the modified scheme proposed by Parks and Militzer (1991), where different relative positioning of the quantization lattices is allowed. In fact, since in the latter case the individual scalar quantizers are more 'de-coupled', one could expect that the assumption about the independence of changes occurring along individual rows of $\mathbf{C}$ should be more accurate.

In order to assess if the alternative CMAC quantization schemes could lead to a better agreement with our estimate, the evaluation of the averaged CMAC distance was also performed for the case of the modified (optimized) quantization, as proposed by Parks and Militzer (1991). Although the average values of ρ obtained in this case were almost identical with the ones obtained for standard CMAC encoding, the deviations from the average tended to be smaller, which agrees with our predictions. Fig. 9 illustrated this point for $K = 40$ and $D = 20$ (analogous effects were observed for other values of D and K).

The formulas obtained for the expected CMAC address distance can be used directly to create the expected form of the (address proximity) basis function associated with the CMAC network, by means of Eq. (5), i.e.,

$$E[\kappa(\sigma)] = K - E[\rho(\sigma)]. \tag{49}$$

We compared the estimated form κ of with the actual CMAC proximity kernels for the 2-dimensional case for both the standard and modified CMAC encodings. The results are given in Figs. 10 and 11, respectively, where it can be seen that the optimized CMAC leads to a higher shape uniformity of proximity kernels, and also to their better agreement with the estimated form of κ.

Note that the estimate of the CMAC proximity kernel,

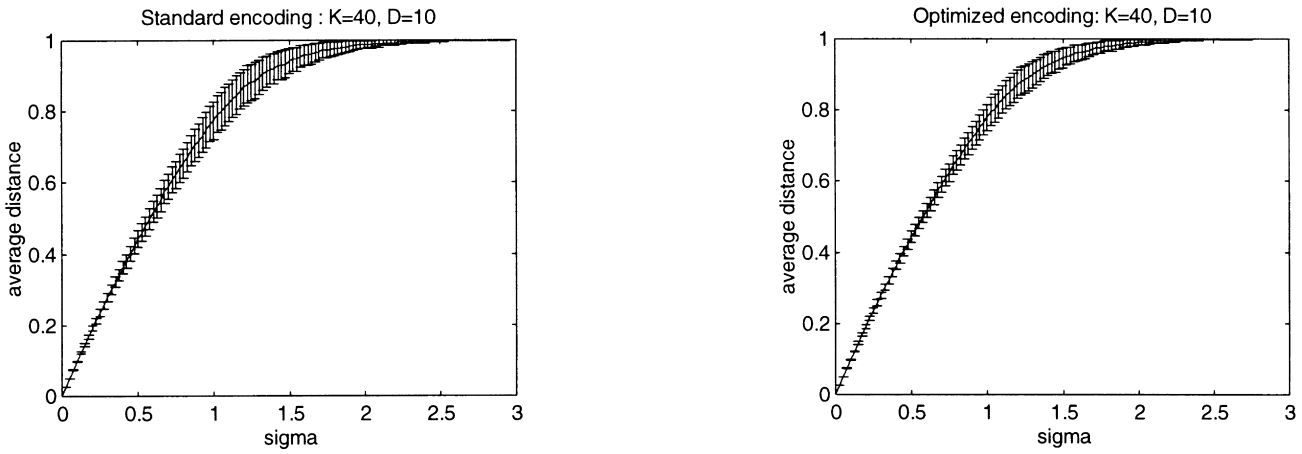

Fig. 9. Examples of the average behaviour of CMAC address distance for the standard and optimized versions of quantization (see text for details). It can be seen that the average behaviour of the distance function is almost identical in both cases, although the modified CMAC encoding is characterized by smaller deviations. The input and address distances are normalized with respect to K.

however useful, has certain limitations in practice, as the CMAC proximity function is piecewise constant, whereas the estimate derived in this study represents a continuous function. However, as the examples presented below illustrate, the estimated CMAC kernel function can be used successfully to predict the performance of the actual CMAC network and to choose its important parameters, such as the number of quantizers, K.

5. Numerical example

5.1. Problem setting

To compare the performance of CMAC and its model (with expected address proximity as the basis function) we considered the problem of predicting the chaotic Mackey-Glass series, which has received much attention in the neural network community (Lapedes and Farber, 1987; Moody and Darken, 1989; Moody, 1989). The series arises as a solution to the following difference-delay equation

$$x(t+1) - x(t) = -bx(t) + a\frac{x(t-\tau)}{1 + x(t-\tau)^{10}} \tag{50}$$

where both t and τ are integers. When the parameters a and b are set to $a = 0.2$ and $b = 0.1$, respectively, the series becomes chaotic for $\tau = 17$ and has a characteristic time of $t_{\text{char}} \approx 50$ (i.e., it almost repeats itself with that period). For this case the fractal dimension of its strange attractor is equal to $d_A \approx 2.1$. When the delay parameter, τ, is increased, higher-dimensional chaos results.

The prediction problem consists of estimating the future value of the series, based on a number of its past instances—that is, the following function has to be estimated (provided that it exists)

$$x(t + \Delta_F) = f(x(t), x(t-\Delta), ..., x(t-(M-1)\Delta)) \tag{51}$$

where Δ represents the delay and Δ_F the prediction offset. According to Takens' theorem (Takens, 1981), the number of past samples should be at least as large as the embedding dimension of the series so that the prediction of a chaotic time series is feasible. Analysis of the embedding dimension for this case presented in the literature gave positive results for $M = 4$ (Farmer and Sidorowich 1987; Aleksic, 1991). Following Farmer and Sidorowich (1987), the value of the time-delay parameter was chosen as $\Delta = 6$, whereas the future prediction offset was selected as $\Delta_F = 85 > t_{\text{char}}$.

Following the experiments presented in the literature, we chose training-set sizes ranging from 100 to 5000 samples. The estimation (generalization) accuracy was quantified by the normalized prediction squared error given by

$$\pi_n = \sqrt{\frac{\sum_{\text{test set}} (x(t) - \hat{x}(t))^2}{\sum_{\text{test set}} (x(t) - E[x(t)])^2}} \tag{52}$$

which represents the average Euclidean distance between the test vector and its estimate, normalized with respect to the standard deviation of the test set. This error measure is insensitive to the absolute values of the time-series data, which facilitates comparing the results of different methods for different data sets. When $\pi_n = 1$ the prediction is no better than providing the average value of the past series samples, whereas in the ideal case $\pi_n = 0$. No noise is assumed to be present in the time-series data. Consequently, the problem is equivalent to multivariate interpolation over a 4-dimensional space.

5.2. Network configurations

The CMAC network and its model were considered in the variants of a normalized RBF (NRBF) (Eq. (15)) and a KR network (Eq. (14)), where in both cases the network

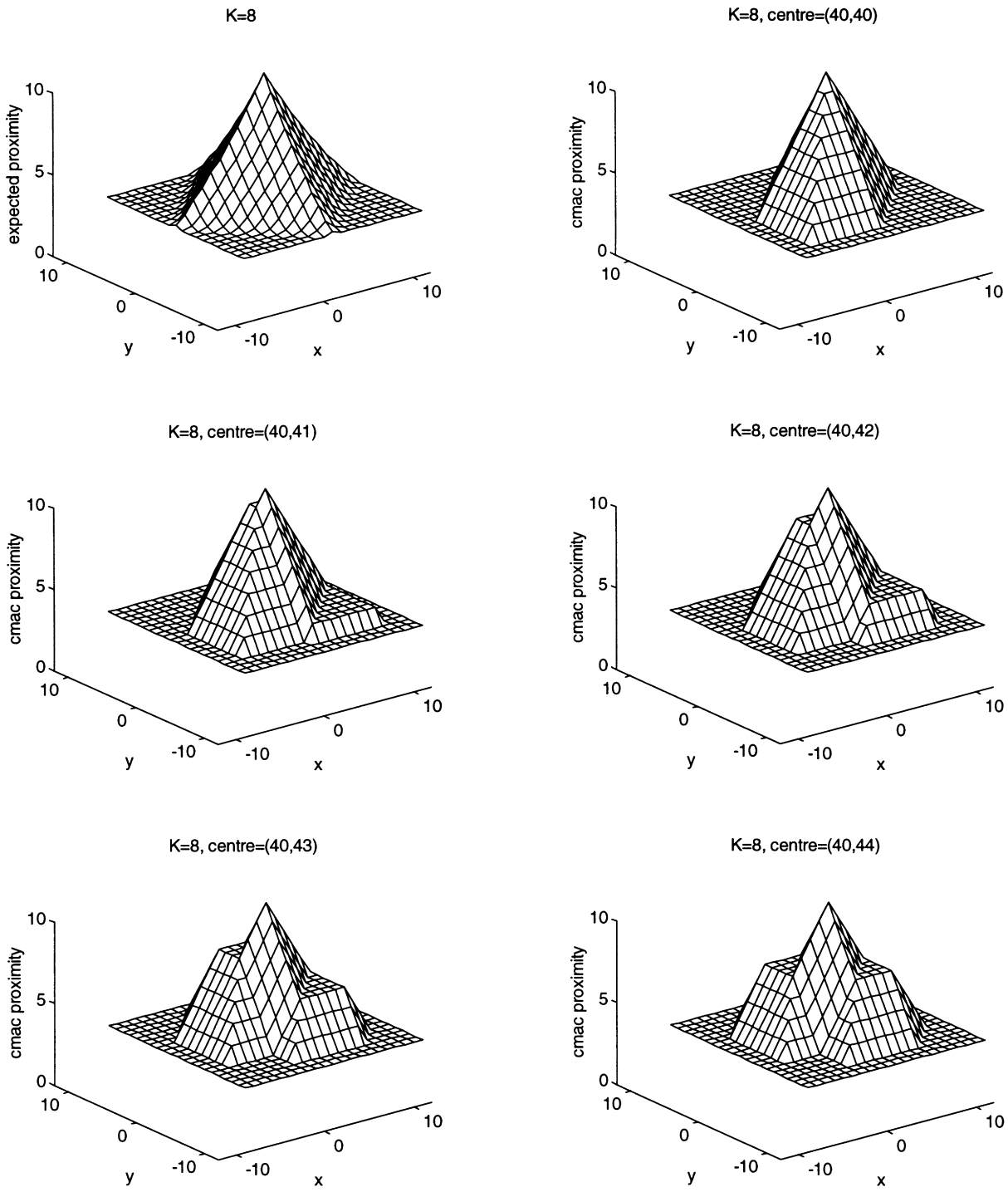

Fig. 10. Comparison between the expected (estimated) and actual CMAC address-proximity functions for a 2-dimensional case and a selection of centre (reference) points.

response is normalized with respect to the basis set. All variants of the network were implemented with standard CMAC encoding and hashing collisions were not resolved.

5.3. Training regime

Most practical implementations of the CMAC network rely on iterative procedures to find the network weights by solving the associated set of linear equations (assuming that the quantization part of the network is fixed). Although application of direct methods is also possible, it requires solving very large and very sparse linear systems, so the computation times tend to be very long (Parks and Militzer, 1992). Moreover, only a small percentage of the potentially

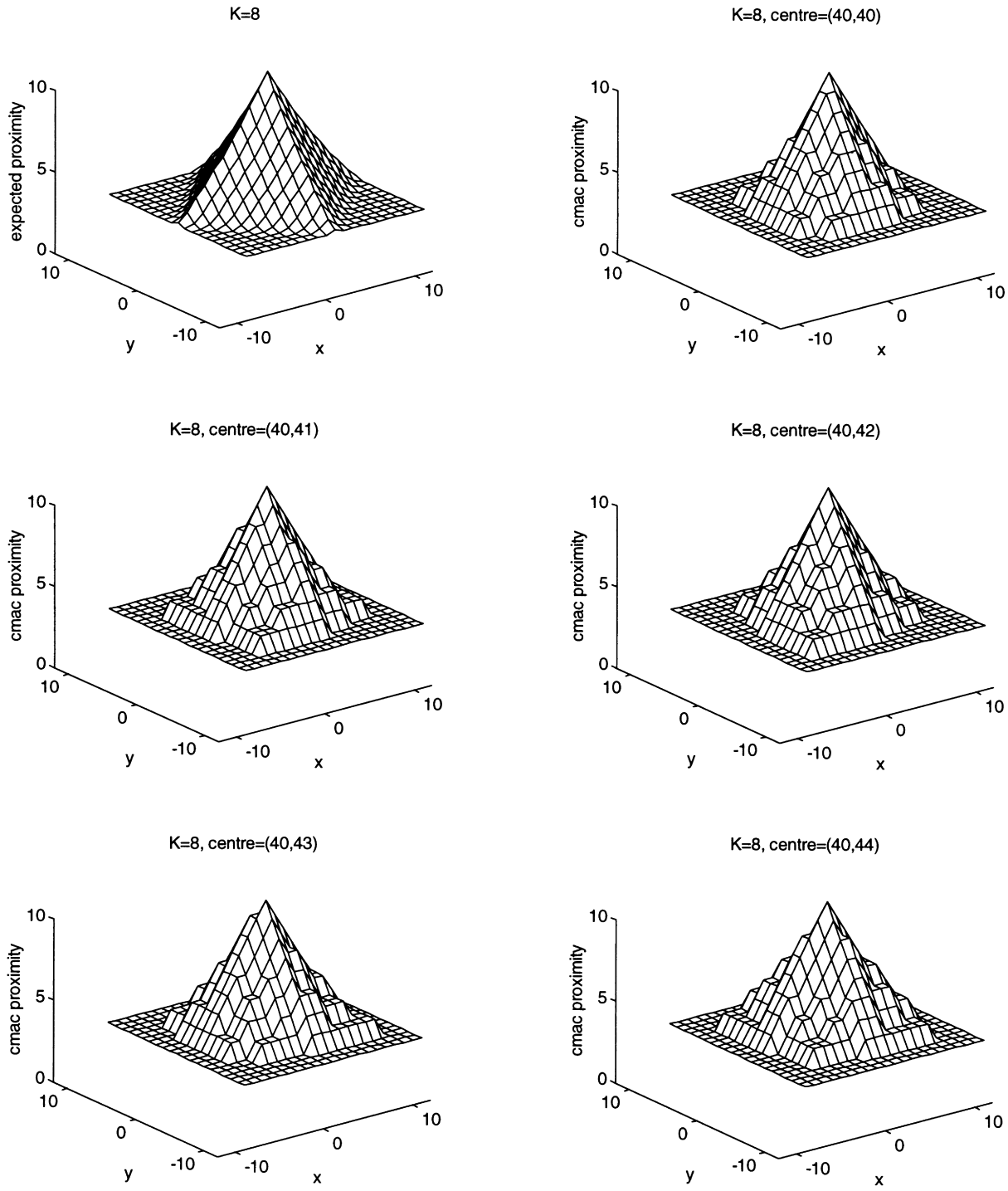

Fig. 11. Comparison between the expected (estimated) and actual CMAC address-proximity functions for a 2-dimensional case and a selection of centre (reference) points; optimized CMAC quantization used.

addressable CMAC locations will be affected by any particular training set, and practical implementations of the network often realise the CMAC memory (weights) using hashing techniques, so that only the portion of the CMAC address space visited by the training-set points is actually implemented. In such situations, application of iterative techniques greatly simplifies the learning process (which can be particularly important in hardware implementations of the network, e.g., Kołcz and Allinson, 1994).

The original training procedure proposed by Albus 1975a and 1975b) represents a special case of the well-known normalized LMS algorithm (NLMS) (Widrow and Stearns, 1985), where the learning adaptation rate is fixed (to the value of 1) such that the instantaneous CMAC error is

brought to zero at each iteration step. In fact, the normalized (rather than standard) LMS weight-update scheme provides an easy and secure way of selecting the adaptation rate, which can take any value in the (0, 2) interval (Tarrab and Feuer, 1988). The properties of the Albus' procedure and its various modifications have been analysed extensively by Parks and Militzer (1992) who also traced the origins of this adaptation scheme to the work of Karczmarz (1937). Certain improvements over the original scheme can be obtained by applying the relaxation principles to choose the order in which the training points are presented to the network (e.g., by always choosing the point for which the adaptation of weights will lead to the largest decrease of the instantaneous error produced by the network—Ellison, 1988). Despite potential problems affecting the convergence properties of the NLMS learning scheme applied to CMAC (which might occur if the underlying system of linear equations is unstable), the NLMS appears to work satisfactorily in practice and has been widely used in various applications of CMAC (Miller et al., 1990; Tolle and Ersü, 1992; Lane et al., 1992).

In the simulations considered in this work the NLMS learning scheme was used to train both the CMAC network and its basis-function models incorporating the estimated proximity-kernel function (Eq. (44), Eq. (49)). The use of iterative learning algorithms facilitates the comparison between the basis-function and the CMAC architectures because, although the former can be solved using standard matrix techniques (e.g., QR or singular value decomposition (Watkins (1991)), the latter (as explained above) evaluates its basis functions implicitly, and for any particular excitation only a small fraction of the network-memory locations is available for adaptation.

In the off-line learning scheme, four cases were considered, corresponding to the training-set sizes of $T = 100$, 500, 1000 and 5000. For each value of T, the NRBF network was trained using the normalized LMS algorithm (NLMS) with the adaptation rate equal to 1 (i.e., at each iteration step the network weights were adapted so as to nullify the instantaneous error at the current training point), for a fixed number of 100 iterations through the whole training set. During each training epoch, the training-set points were presented to the network sequentially. In all cases the test set consisted of 500 points of the series immediately following the training set.

5.4. Selection of the optimal smoothing parameters

Both the NRBF and KR networks can have, in principle, different smoothing parameters at each basis. However, since in the estimated-kernel models of CMAC the smoothing remains constant throughout the network (and is controlled by the parameter K), only this case was considered. Even in this simplified case, however, the optimum bandwidth of the basis has to be chosen, based on the information provided by the training set. Generally, the bandwidth parameters can be adjusted simultaneously with the rest of the network's parameters (e.g., weights) to minimize some global error criterion. However, such an approach is equivalent to nonlinear optimization because of the nonlinear dependence of the network response on the bandwidth; consequently the optimization process is difficult and liable to be trapped in local minima of the error function (Tarassenko and Roberts, 1994). Therefore, we adopted a hybrid scheme where the bandwidth is chosen first, and then adaptation of the weights follows. Heuristic schemes of bandwidth selection are often based on the nearest-neighbour properties of the training set. For networks with Gaussian units, for instance, Moody and Darken (1989) suggest setting the σ parameter of the basis $\kappa(\mathbf{x}) = \exp(-\|\mathbf{x}\|_2^2/\sigma)$ to the average squared Euclidean distance between nearest neighbours in the training set. Similar heuristics can also be designed if the kernel function depends on a non-Euclidean norm of its vector argument.

We chose to combine the heuristic method with the leave-one-out cross-validation scheme (Härdle, 1990). Cross validation is often used in KR networks and, as pointed out by Specht (1991), it usually exhibits a fairly wide region of 'good' bandwidth choices, preceded and followed by regions of worse performance. For the case of the KR network, a heuristic (i.e., nearest neighbour) choice of bandwidth served as the starting point in a local optimization procedure to identify the region of the best bandwidth. This optimum bandwidth choice was then extended to NRBF networks, which have essentially the same architecture (apart from training) as KR networks.

Fig. 12 shows the cross-validation results for the selection of K obtained for CMAC implementing a KR network and for a KR network utilizing the estimated CMAC kernel, (Eq. (46)). It can be seen that, although the absolute values of the cross-validation errors are somewhat different, both networks identify the optimal values of K as lying in the same range. Note that the range of 'best' K is fairly independent of the training-set size.

The number of quantizers K was in each case used to determine the size of the uniform scalar-quantization cell, which was obtained by dividing the effective range of each input variable (estimated from the training data) by K. Uniform quantization was then combined with the standard (i.e., not optimized) arrangement of CMAC quantization lattices for the individual input dimensions.

5.5. Test-set performance

Fig. 13 shows the test-set results achieved by CMAC and its NRBF and KR equivalent models. The model networks, employing smooth basis functions, obtained better approximation accuracy, although it can be seen that the LMS training brought the performance of CMAC to that of the KR model network. The differences in performance can be attributed to the inherent irregularity and piecewise-constant nature of CMAC basis functions, as well as the

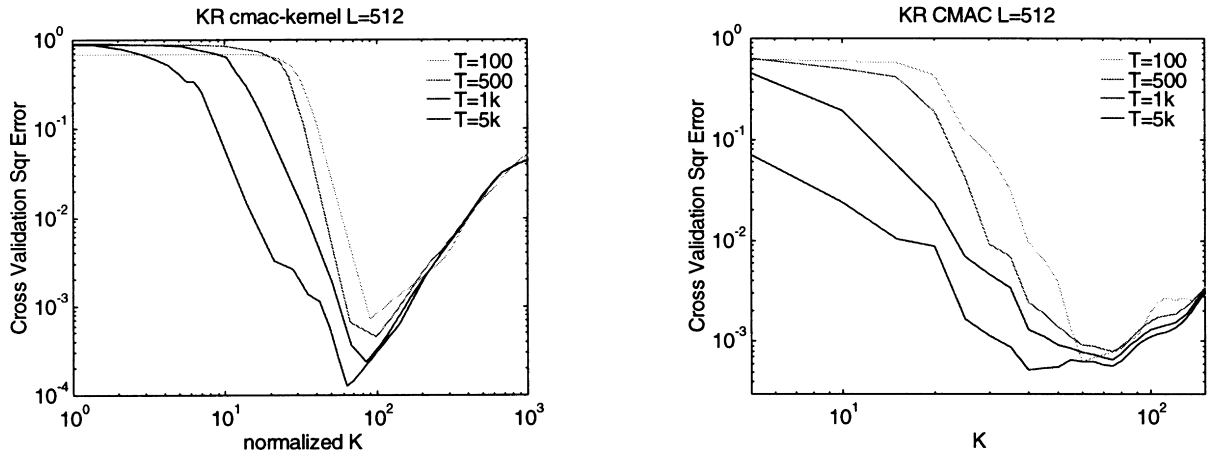

Fig. 12. Optimum selection of K via cross-validation for the KR network implemented by CMAC and its model. It can be seen that both the original network and its model lead to a similar choice of optimal parameters. Interestingly, the region of 'best' K seems to be fairly independent of the training-set size.

(unresolved) hashing collisions present in its mapping. In fact, Brown et al. (1993) have shown that for the class of piecewise-constant functions defined over the CMAC quantization lattice, the network can realize any additive function (i.e., a D-variate function that can be represented as a sum of D univariate components), but is inherently incapable of implementing an arbitrary function from this class. In this respect, a basis-function model of the CMAC network is more flexible, as it can potentially approximate any smooth mapping (CMAC may offer certain advantages if the original function has a piecewise-constant form as well). Further larger-scale experiments should be carried out to better assess the performance of basis-function models with respect to the original CMAC network.

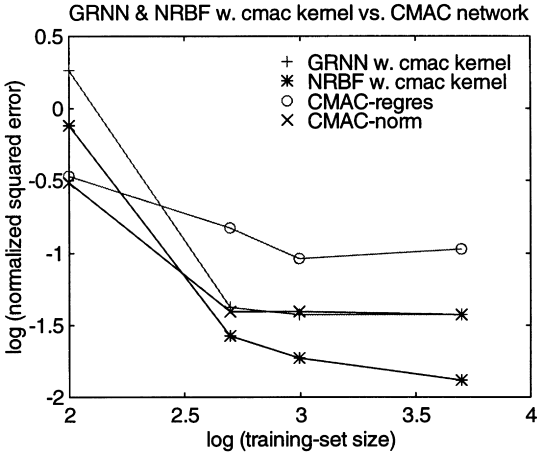

Fig. 13. Comparison between the performance of the CMAC network and its basis-function models in the considered configurations. The NRBF and KR networks utilizing the estimated CMAC kernel can be seen to perform better than the actual CMAC network in these configurations (see text for discussion). CMAC-regres and CMAC-norm correspond to CMAC implementations of KR and NRBF networks, respectively.

6. Conclusions

The quantization performed by the CMAC network has been described in the context of equivalence between CMAC and the more general GMNN architecture. Particular emphasis was placed on the case of uniform quantization, which is particularly regular and amenable to formal analysis.

In particular, we have derived the formula for the expected address distance function of the CMAC network and shown how it can be used to create an approximate basis-function model of this architecture. The closed-form approximate basis derived in this work allows us to create RBF- and KR-like models of the CMAC network where each basis function is associated with one training point, which differentiates them from the more straightforward interpretation of CMAC, where the basis functions are equivalent to the memory activation functions (i.e., they take only 0 or 1 values). The result obtained here allows us to compare CMAC with other basis function networks (e.g., RBFs), even in the case where the response of the CMAC network has inherently a piecewise-constant form. In the latter case, the utility of the continuous-basis estimate of the CMAC proximity kernel lies mainly in the area of providing a means of estimating optimal values of CMAC parameters (e.g., K) for a particular training set and thus facilitating network design. Of course, the actual response of the CMAC network remains piecewise-constant, unless additional smoothing operators (e.g., B-splines) are applied. The obtained estimated CMAC distance function (used to define the basis) was compared with the averaged results of actual simulations of the CMAC network. Close agreement between the actual and the predicted behaviour of the CMAC distance function was demonstrated (especially in the case of optimized CMAC encoding).

The performance of CMAC and its basis-function models was also compared on a benchmark problem of

Mackey-Glass chaotic time-series prediction. Although the approximation errors produced by both networks were comparable, we found the actual CMAC to be characterized by poorer performance than its model, which may be attributed to the piecewise constant nature of its response and the detrimental effect of unresolved hashing collisions. However, the model produced very good agreements as far as the selection of optimum network parameters was concerned.

We believe that the interpretation of CMAC mapping presented in this paper offers increased understanding of CMAC operation (which can be also extended to networks of similar structure). The basis-function model of the network and its utility in practical applications will be a matter of further research.

Some additional aspects of this work are also presented in (Kołcz, 1996).

Acknowledgements

One of the authors (AK) would like to thank the Overseas Research Foundation, York University and UMIST for supporting this research.

References

Albus, J.S. (1971). A theory of cerebellar function. *Mathematical Biosciences, 10*, 25–61.

Albus, J.S. (1975a). Data storage in the cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems, Measurement and Control, 97* (*3*), 228–233.

Albus, J.S. (1975b). A new approach to manipulator control: the Cerebellar Model Articulation Controller (CMAC). *Journal of Dynamic Systems, Measurement and Control, 97* (*3*), 220–227.

Albus, J.S. (1979). Mechanisms of planning and problem solving in the brain. *Mathematical Biosciences, 45*, 247–293.

Aleksic, Z. (1991). Estimating the embedding dimension. *Physica D, 52*, 362–368.

Bartels, R.H., Beatty, J.C., & Barsky, B.A. (1987). *An introduction to splines for use in computer graphics and geometric modeling*. Los Altos, CA: Morgan Kaufmann.

Bledsoe, W., & Browning, I. (1959). *Pattern recognition and reading by machine*. IRE Joint Computer Conference, pp. 225–232.

Broomhead, D.S., & Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems, 2*, 321–355.

Brown, M., Harris, C.J., & Parks, P.C. (1993). The interpolation capabilities of the binary CMAC. *Neural Networks, 6* (*3*), 429–440.

Cotter, N.E., & Guillerm, T.J. (1992). The CMAC and a Theorem of Kolmogorov. *Neural Networks, 5*, 221–228.

Ellison, D. (1988). On the convergence of the Albus perceptron. *IMA Journal of Mathematics Control and Information, 5*, 315–331.

Farmer, J.D., & Sidorowich, J.J. (1987). Predicting chaotic series. *Physical Review Letters, 59*, 845–848.

Girosi, F., Jones, M., & Poggio, T. (1995). Regularization theory and neural network architectures. *Neural Computation, 7*, 219–269.

Gersho, A., & Gray, R.M. (1992). *Vector quantization and signal compression*. Boston: Kluwer Academic.

Härdle, W. (1990). Applied nonparametric regression. Cambridge: Cambridge University Press.

Karczmarz, S. (1937). Angenäherte auflösung von systemen linearer gleichungen. *Bull. Int. Acad. Pol. Sic. Let., Cl. Sci. Math. Nat.*, 355–357.

Kołcz, A. (1996). *Approximation properties of memory-based artificial neural networks*. PhD thesis, University of Manchester Institute of Science and Technology (UMIST): Department of Electrical Engineering and Electronics.

Kołcz, A., & Allinson, N.M. (1994). Reconfigurable logic implementation of memory based neural networks: a case study of the CMAC network. In Delgano-Frias, J.G. and Moore, W.R. (Eds.), *VLSI for neural networks and artificial intelligence* (pp. 177–186). New York: Plenum.

Kołcz, A., & Allinson, N.M. (1995) General Memory Neural Network – extending the properties of basis networks to RAM-based architectures. In Proc. 1995 *IEEE Int. Conf. on Neural Networks (ICNN '95)* (pp. 1638–1643), Perth, Western Australia.

Kołcz, A., & Allinson, N. M. (1996). N-tuple regression network. *Neutral Networks, 9* (*5*), 855–869.

Lane, S.H., Handelman, D.A., & Gelfand, J.J. (1992). Theory and development of higher-order CMAC neural networks. *IEEE Control Systems Magazine*, 23–30.

Lapedes, A.S., & Farber, R. (1987). *Nonlinear signal processing using neural networks: prediction and system modelling*. Technical Report, Los Alamos Laboratory, Los Alamos, New Mexico.

Marr, D. (1969). A theory of cerebellar cortex. *J. Physiol., 202*, 437–470.

Miller, W.T., Glanz, F.H., & Kraft, L.G. (1990). CMAC: An associative neural network alternative to backpropagation. *Proc. IEEE, 78* (*10*), 1561–1567.

Moody, J. (1989). Fast learning in multi-resolution hierarchies. In Touretzky, D.S. (Ed.), *Advances in neural information processing systems* 1 (pp. 29–39). Morgan Kaufmann.

Moody, J., & Darken, C.J. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation, 1*, 281–294.

Parks, P.C., & Militzer, J. (1989). Convergence properties of associative memory storage for learning control systems. *Avtomatica i Telemekhanika, 50* (*2*), 158–184.

Parks, P.C., & Militzer, J. (1991). Improved allocation of weights for associative memory storage in learning control systems. In 1st *IFAC symposium on design methods of control systems*, pp. 507–512.

Parks, P.C., & Militzer, J. (1992). A comparison of five algorithms for the training of CMAC memories for learning control systems. *Automatica, 28* (*5*), 1027–1035.

Powell, M.J.D. (1992). The theory of radial basis functions in 1990. In Light, W.A. (Ed.), *Advances in numerical analysis Vol. II: wavelets, subdivision algorithms and radial basis functions* (pp. 105–210). Oxford: Oxford University Press.

Specht, D.F. (1991). A general regression neural network. *IEEE Transactions on Neural Networks, 2* (*6*), 568–576.

Takens, F. (1981). Detecting strange attractors in turbulence. In Rand, D.A., & Young, L.S. (Eds.), *Dynamical systems and turbulence* (pp. 366–381). Berlin: Springer-Verlag.

Tarassenko, L., & Roberts, S. (1994). Supervised and unsupervised learning in radial basis function classifiers. *IEE Proc.-Image Signal Process, 141* (*4*), 210–216.

Tarrab, M., & Feuer, A. (1988). Convergence and performance analysis of the normalized LMS algorithm with uncorrelated gaussian data. *IEEE Transactions on Information Theory, 34* (*4*), 680–691.

Tattersall, G.D., Foster, S., & Johnston, R.D. (1991). Single-layer lookup perceptrons. *IEEE Proceedings-F, 138* (*1*), 46–54.

Tolle, H., & Ersü, E. (1992). *Neurocontrol—learning control systems inspired by neuronal architectures and human problem solving strategies*. Berlin: Springer-Verlag.

Watkins, D.S. (1991). *Fundamentals of matrix computations*. New York: John Wiley and Sons.

Widrow, B., & Stearns, S.D. (1985). *Adaptive signal processing*. Englewood Cliffs, NJ: Prentice-Hall.