Application of the CMAC input encoding scheme in the *N*-tuple approximation network

A. Kolcz
N.M. Allinson

Indexing terms: Networks, Coding, CMAC, N-tuple networks

Abstract: The *N*-tuple approximation network offers many advantages over conventional neural networks in terms of speed of operation and its ability to realise arbitrary nonlinear mappings. However, its generalisation/selectivity properties depend strongly on the form of input encoding being used in the system. The paper analyses the suitability of use of the CMAC code for the *N*-tuple networks, and compares its properties with existing schemes. It is argued that the application of this type of encoding can provide desirable monotonic mapping between input and pattern space distances without the penalty of very long binary patterns as is the case for bar-chart encoding. Additionally, similarities between the classic *N*-tuple and CMAC networks are highlighted.

1 Introduction

Two major applications of *N*-tuple neural networks [5] are pattern recognition [3, 4] and arbitrary function approximation [7]. In both cases the input to the system is transformed into a binary pattern which is entered onto the system retina (traditionally envisioned as a rectangular array of binary pixels, see Fig. 1). Groups of *N* pixel locations of the retina are subsequently chosen in a way characteristic of the sampling algorithm being used. In most cases the sampling process is random. Although complete sampling is most common (i.e. each pixel is taken only once) under- and oversampling are possible. Each *N*-tuple of the pattern input addresses a separate *N*-tuple memory whose contents are interpreted differently for each architecture.

For pattern recognition systems, the input to the network is usually a pattern representing a discrete function over the discrete domain of the network's retina. The contents of each *N*-tuple memory contain set bits in locations corresponding to addresses which were picked by the training patterns. The width of the memory words depends on the number of classes to be distinguished. The class response of the network to a random trial pattern is equal to the number of *N*-tuple memories which produce a 1. For proper discrimination, the response of the network should be substantial only for patterns close to the trained class. Allinson and Johnson [4] showed that the response decreases (approximately) exponentially with the Hamming distance between the test pattern and the pattern cluster contained in the *N*-tuple memories.

The function approximation network accepts the input of a *D*-dimensional numeric vector, whose coordinates are converted into a binary form and the resulting pattern is then presented to the network's retina. The sampled *N*-tuple subpatterns serve as addresses to a set of tuple memories, each containing numeric weights in an arbitrary format. The addressed

Paper 1004E (C2, C3), first received 9th March and in revised form 8th September 1993

The authors are with the Image Engineering Laboratory, Department of Electronics, University of York, Heslington, York YO1 5DD, United Kingdom

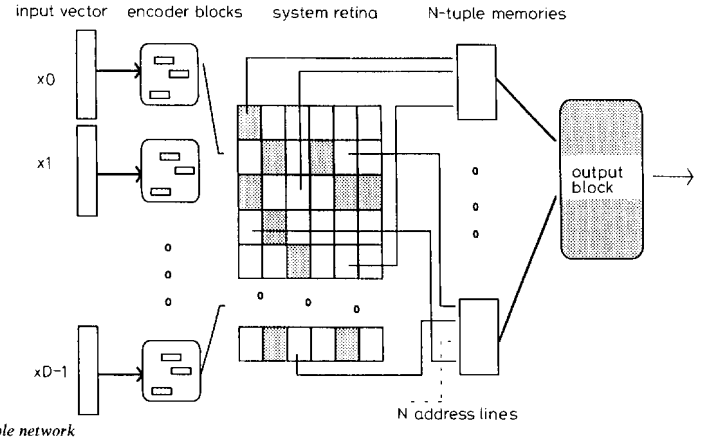

Fig. 1 *A general N-tuple network*

words are summed to produce an output response, similar to the classic perceptron scheme, but with no final thresholding. The least mean squares scheme is usually used for training. The output is generated by a form of interpolation between stored responses, which requires that the number of N-tuple addresses shared by any two different input vectors decreases with their distance.

It is clear that the points lying close together in the input space should be assigned patterns close in the binary space using the Hamming metric. The function of the input encoding is to provide this property while keeping the pattern size as small as possible to minimise the memory requirements. Some codes (e.g. the barchart code) fail to satisfy the requirement, which makes it necessary to look for other more effective solutions.

In this paper the approximation N-tuple network is considered. The CMAC-type coding scheme is analysed from the point of its validity for input encoding of a general N-tuple system. It is shown that the CMAC is, in fact, a special case of the N-tuple approximation network.

1.1 CMAC network

The CMAC is a supervised neural network, similar to the perceptron in its operation, but one which employs a special kind of nonlinear mapping in the input variable encoding [1, 2]. The mapping results in the property that only points located very close to each other in the input space share any weights. The code is generated by transforming a D-dimensional input vector space into a K-dimensional address space, where K is an arbitrarily chosen association coefficient, a major parameter of the CMAC architecture, responsible for its generalisation properties. The result produced by the CMAC is computed as a sum of selected weights. The transformation is a stepwise process which proceeds as follows (Fig. 2):

(i) Each variable, x_i, of the input vector, $\boldsymbol{X}[x_0, x_1, \ldots, x_{D-1}]$, is assigned a K-dimensional field vector $\boldsymbol{M}_i[m_{i0}, m_{i1}, \ldots, m_{iK-1}]$, where the field positions (codewords) are calculated according to [6]:

$$m_{ij} = \left\lfloor \frac{x + K - 1 - j}{K} \right\rfloor$$

$$i \in [0, D-1], \quad j \in [0, K-1], \quad 0 \leqslant m_{ij} \leqslant \left\lfloor \frac{L-1}{K} \right\rfloor$$

This is equivalent to passing each variable over a set of K overlapping quantisers, each having the quantisation step of K, and each shifted by one input resolution element with respect to its neighbours.

(ii) The field vectors $\boldsymbol{M}_0, \boldsymbol{M}_1, \ldots, \boldsymbol{M}_{D-1}$ are concatenated positionwise to produce the address vector $\boldsymbol{V}$, whose co-ordinates are used to address their respective weight memories during the response computation. The same address vector is used for every output dimension of the network.

(iii) When the system does not possess adequate memory resources, i.e. the length of the address vector is too high, hashing procedures are used to reduce the effective address length. The resulting noise is known as hashing collisions [6].

Two of the most important features of the CMAC architecture are the speed of operation (i.e. few weight additions, since K is usually small) and very high convergence rate.

2 Space transformations

The input to the system consists of a D-dimensional vector $\boldsymbol{X}$ of integer co-ordinates, each within the range of L discrete values, i.e.

$$\boldsymbol{X} = [x_0, x_1, \ldots, x_{D-1}]$$

where $x_i = 0, 1, \ldots, L-1$ for $i = 0, 1, \ldots, D-1$. Each of the co-ordinates is transformed independently into a binary pattern by means of a code C,

$$C: \boldsymbol{X} \rightarrow C\boldsymbol{X}, \quad C\boldsymbol{X} = [Cx_0, Cx_1, \ldots, Cx_{D-1}]$$

In the simplest case C is a natural binary code. If W designates the number of bits necessary to represent each co-ordinate of this vector then the total length of the pattern is $D \cdot W$. The code C is generally an 'into' mapping, as there can be patterns with no corresponding input vectors. Such redundancy may be necessary to enforce the desired properties of generalisation.

The final binary pattern is subjected to the Hamming vector distance metric given by

$$h(C\boldsymbol{X}, C\boldsymbol{Y}) = \sum_{i=0}^{D-1} h(Cx_i, Cy_i), \quad h(C\boldsymbol{X}, C\boldsymbol{Y}) \in [0, D \cdot W]$$

and

$$h(Cx_i, Cy_i) \in [0, W]$$

As distances between subpatterns provide additive contributions to the total Hamming distance (owing to the independent encoding of the vector co-ordinates), a similar relationship in the input space makes it easier to analyse the distance mapping properties of the encoding transformation. Therefore, the city-block distance is used as the metric function of the input space, i.e.

$$d(\boldsymbol{X}, \boldsymbol{Y}) = \sum_{i=0}^{D-1} |x_i - y_i|, \quad d(\boldsymbol{X}, \boldsymbol{Y}) \in [0, D \cdot (L-1)]$$

3 Desired code properties

The major requirement of a satisfactory input code is that it maintains the distance relationships between points in the input and pattern spaces, i.e.

$$\forall \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}: d(\boldsymbol{X}, \boldsymbol{Y}) \leqslant d(\boldsymbol{X}, \boldsymbol{Z}) \Rightarrow h(C\boldsymbol{X}, C\boldsymbol{Y}) \leqslant h(C\boldsymbol{X}, C\boldsymbol{Z})$$

If we define a distance mapping function g_x as

$$g_x(r) = \min \{d(C\boldsymbol{X}, C\boldsymbol{Y}): d(\boldsymbol{X}, \boldsymbol{Y}) = r\}$$

$$\text{for integer} \quad 0 \leqslant r \leqslant D \cdot (L-1)$$

then in the desired mapping g_x should be monotonically increasing at each point $\boldsymbol{X}$ of the input domain. In other words, if vectors belong to a certain spherical neighbourhood in the input space, then their encodings should belong to a corresponding neighbourhood in the pattern space. Because the dynamic range of the Hamming distance is usually lower than that of the input-space distance and depends on the amount of redundancy employed by the specific code, the strictly monotonic distance mapping is impossible over the whole input domain. The requirement of the monotonic distance mapping is quite strict but it can be relaxed to some extent if we note that, owing to the properties of N-tuple sampling, the response of the network is not a linear function of the Hamming distance between patterns, $C\boldsymbol{X}$ and $C\boldsymbol{Y}$, but an exponential one. Consequently, if the Hamming distance increases beyond some threshold value, the response will be small enough to render the

distance relationship irregularities above this value insignificant. Therefore, if g_x is locally increasing within some neighbourhood of every point in the input domain, and

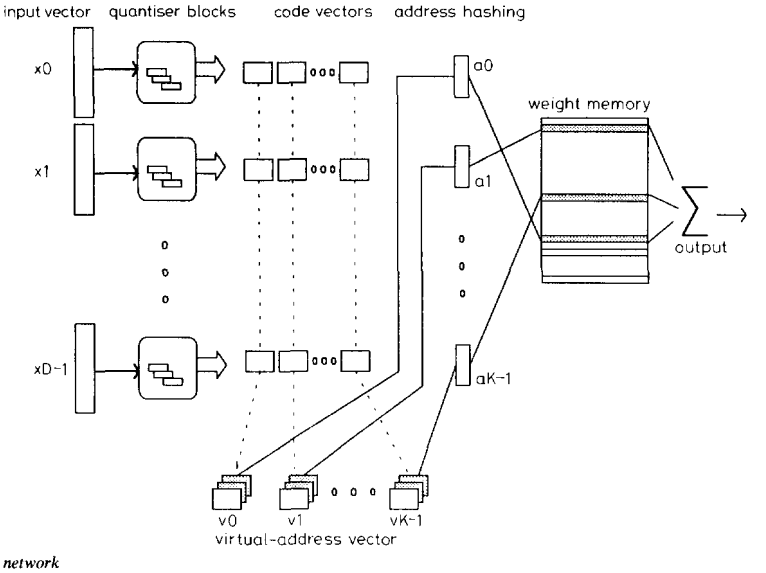

Fig. 2 *The CMAC network*

remains sufficiently bounded from below outside this neighbourhood, the mapping provides satisfactory discrimination properties as far as the N-tuple system is concerned. The locally monotonic mapping can be formulated as follows,

$$\exists R, T_{out}\ \forall \boldsymbol{X}\ g_x(r) \quad \text{is monotonically increasing}$$
$$\text{for integer } r \quad 0 \leqslant r \leqslant R$$

and

$$g_x(r) \geqslant T_{out} \qquad \text{for integer} \quad r > R$$

The radius R determines the size of the monotonicity neighbourhood (a closed ball in metric space terminology) of the distance mapping at each point of the input space; whereas T_{out} designates the lower bound on the pattern space distance outside this neighbourhood. Additionally, if $T_{in} = \max_x \{g_x(r): 0 \leqslant r \leqslant R\}$, it is desirable that $T_{out} \geqslant T_{in}$, so that the transformation does not distort distance relationship in regions of monotonicity. R, T_{in} and T_{out} constitute the distance mapping parameters.

Since it is sometimes important to analyse properties of the distance mapping only for points belonging to a certain subset Δ of the input space, the distance mapping over a set is defined as

$$g_x(r, \Delta) = \min \{d(C\boldsymbol{X}, C\boldsymbol{Y}): d(\boldsymbol{X}, \boldsymbol{Y}) = r \wedge Y \in \Delta\}$$
$$\text{for integer } r \quad 0 \leqslant r \leqslant D \cdot (L - 1)$$

4 Comparison of different codes

4.1 Binary code

The binary code provides a natural representation of an integer variable, but has very nonlinear and non-monotonic properties as far as the pattern space is concerned. Specifically, integer values differing just by one level can have their Hamming distance anywhere between one and the maximum. The N-tuple sampling process assumes that all bits in the input pattern have the same relative importance. This is certainly not the case with the natural binary notation, where the bits are ordered in the increasing level of significance. Another deficiency of the natural binary encoding is the relatively small dynamic range of the pattern space distance $(W + 1)$, when compared with the input range (2^w).

4.2 Gray code

The Gray code does not improve the dynamic range of the pattern space distance but results in a more regular distance mapping. Because of its properties, integer variables differing by one level have exactly one bit position changed in their Gray code representation. Therefore, in very small neighbourhoods the distance mapping is monotonically increasing.

4.3 Concatenated Gray code

The concatenated Gray code [8] provides some remedy to the small dynamic range of the Gray code. The encoding process assigns each scalar integer variable a K-dimensional code vector, where each co-ordinate corresponds to a Gray code of the input offset by a shift value,

$$Cx = [G(x), G(x + s_1), \ldots, G(x + s_{K-1})]$$
$$\text{for integers} \quad 0 \leqslant s_i \leqslant L \quad \text{and} \quad 1 \leqslant i < K$$

There are different arrangements of the shift values s_i and for a particular K and range of input variables it is possible to find some optimal configuration, in which the irregularities of the Gray code are averaged. The distance of one in the input space is transformed to the distance of K in the pattern space. Although this code is more regular, the distance mapping is monotonic only for neighbourhoods with a radius of one, as for the Gray code. Additionally, large values of concatenated fields result in large distances between neighbouring patterns.

4.4 Barchart code

The barchart, or thermometer, code has long been recognised as that having ideal properties since it results in

identical distances in both input and pattern spaces. The major disadvantage of using this code is its length, which for a multidimensional input might simply become impractical. Tattersall [7] has proved that for this code the number of common weights shared by distinct points decreases exponentially with their input/pattern distance.

4.5 CMAC code

To begin, let us limit the number of input dimensions to just one. In this case the field vector M generated for the input X is identical to the address vector V, since no concatenation takes place. A discrete metric over the vector-field space is chosen, because this distance directly relates to the number of fields shared by two different inputs. Owing to the quantisation encoding of the CMAC, the distance in the field vector space is a clipped version of the input distance, which can be expressed as

$$d(Cx, Cy) = |x - y| \bmod (K + 1), \quad 0 \leqslant d(Cx, Cy) \leqslant K$$

Thus the distance mapping is monotonically increasing (though not strictly) over the whole input domain. Namely,

$$\forall x \; g_x(r) = \begin{cases} r & \text{for } 0 \leqslant r \leqslant K \\ K & \text{for } K < r < L \end{cases}$$

so the function has clear linear and saturated regions. We define the CMAC neighbourhood $\Omega(X)$ of a point X as a set of input points whose distance from the centre is not saturated. It follows directly that each neighbourhood contains (excluding boundary conditions) $2 \cdot K + 1$ points. For a multidimensional input, the CMAC system considers any two addresses as different if not all the corresponding fields are identical. However, for the N-tuple encoding application, each field is equally relevant and the metric in the field vector space is defined as a direct extension of the 1-dimensional model. Instead of thinking of the global pattern as a K-dimensional vector of concatenated field subvectors, we consider it as a D-dimensional vector of K-dimensional field subvectors. Using the previously derived expression for the distance in the field vector space, the new distance is naturally extended to

$$d(CX, CY) = \sum_{i=0}^{D-1} d(Cx_i, Cy_i)$$

$$= \sum_{i=0}^{D-1} |x_i - y_i| \bmod (K + 1)$$

$$0 \leqslant d(CX, CY) \leqslant D \cdot K$$

with each $\Omega(X)$ containing $D \cdot (2 \cdot K + 1)$ points. In contrast with the 1-dimensional case, the distance mapping here is monotonic only when no saturation occurs for any dimension, i.e. only for points in $\Omega(X)$. Therefore g_x is locally (and strictly) monotonically increasing in the largest spherical neighbourhood which fits in $\Omega(X)$, and has the parameters

$$R = K, \quad T_n = K, \quad T_{out} = K$$

Fig. 3 shows the plot of the function $d(CX, C[8, 8])$ for the CMAC ($K = 4$) encoding of a 2-dimensional ($L = 16$) input. The most important difference between the input and field vector spaces is that large distances between integer vectors owing to significant changes in few variables will only still produce small distances in the transformed space. This is caused by the clipping mechanism in the distance mapping.

To arrive at the final binary pattern each discrete field position is substituted with its actual form, i.e. an integer

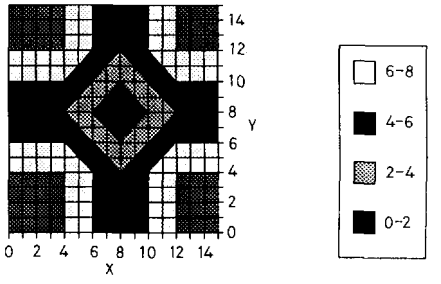

Fig. 3 *CMAC vector field space neighbourhood for 2-dimensional input space*

in the range: $0, \ldots, \lceil (L - 1)/K \rceil$, which can be represented by the natural binary or an alternative code of length W. Increments by one in the field space distance correspond to one field position changing its value, and hence changing its binary representation. Furthermore, note that within the CMAC neighbourhood each field position differs numerically by at most one from that of the centre, i.e.

$$|m_{ij} - m_{ij}^{centre}| \leqslant 1$$

$$\text{for } j = 0, 1, \ldots, K - 1 \quad \text{and} \quad i = 0, 1, \ldots, D - 1$$

Each change of a field position results in a nonzero Hamming distance increase in the range of $1, \ldots, W$. Consequently, in regions where the field space distance mapping is not saturated, the global distance mapping g_x is monotonically increasing (with a varying but nonzero increment), with the parameters

$$R = K, \quad T_{in} = K \cdot W, \quad T_{out} = K$$

As situations where $T_{out} < T_{in}$ are possible, points close to each other in the input space can be mapped into distant patterns, and distant points into similar patterns. Fortunately this problem can be solved by the additional application of Gray coding as shown in Section 4.7. Fig. 4 shows the three levels of abstraction in the CMAC input encoding that have been distinguished.

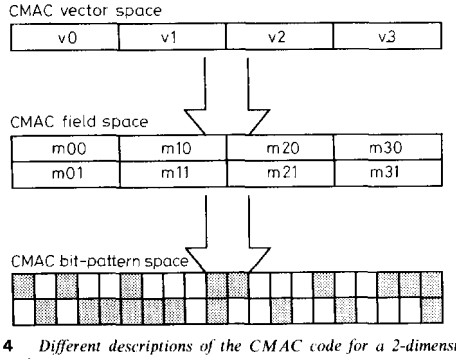

Fig. 4 *Different descriptions of the CMAC code for a 2-dimensional input and K = 4*

4.7 Use of Gray code for field position representation in the CMAC coding

It has been noted that inside the neighbourhood each field position of the pattern vector changes only by one (increment or decrement) in the algebraic sense. If the field positions are represented in Gray code rather than

Table 1: Distance mapping properties of the CMAC code

Code		Monotonicity neighbourhood		CMAC neighbourhood	
CMAC	Inside	$r \leqslant g_x(r) \leqslant r \cdot W$	$0 \leqslant r \leqslant K$	$r \leqslant g_x\{r, \Omega(\boldsymbol{X})\} \leqslant r \cdot W$	$0 \leqslant r \leqslant D \cdot K$
	Outside	$K \leqslant g_x(r) \leqslant D \cdot K \cdot W$	$K \leqslant r \leqslant D \cdot (L-1)$	$D \cdot K \leqslant g_x\{r, \Omega(\boldsymbol{X})\} \leqslant D \cdot K \cdot W$	$D \cdot K \leqslant r \leqslant D \cdot (L-1)$
CMAC/Gray	Inside	$g_x(r) = r$	$0 \leqslant r \leqslant K$	$g_x\{r, \Omega(\boldsymbol{X})\} = r$	$0 \leqslant r \leqslant D \cdot K$
	Outside	$K \leqslant g_x(r) \leqslant D \cdot K \cdot W$	$K \leqslant r \leqslant D \cdot (L-1)$	$D \cdot K \leqslant g_x\{r, \Omega(\boldsymbol{X})\} \leqslant D \cdot K \cdot W$	$D \cdot K \leqslant r \leqslant D \cdot (L-1)$

the natural binary one, these changes will result in an increment by one in the Hamming distance. Therefore, the constraints on the distances inside and outside the monotonicity neighbourhood are reformulated to

$$R = K, \quad T_{in} = K, \quad T_{out} = K$$

which correctly preserves the distance relationships. Additionally, the use of Gray code does not influence the field word length W, which has the same value as in the natural binary notation. Table 1 outlines the properties of g_x for different neighbourhoods with and without additional Gray coding.

5 Code length considerations

The above considerations show that the relationship between distances in the input and pattern spaces remains monotonic in the relaxed sense. Another issue, however, is the number of bits required by this code. As each field is encoded by W bits, the total number is $D \cdot K \cdot W$, which compares favourably with the barchart code [which requires $D \cdot (L-1)$ bits] if $K \cdot W \ll L - 1$. Fig. 5 provides the length of the CMAC code for input dimension for $L = 256$ and different values of the association coefficient K.

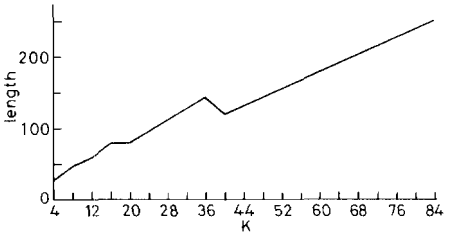

Fig. 5 *CMAC code length in bits per input vector dimension as a function of K for L = 256*

6 Comparison between the CMAC network and the SLLUP

The approximation N-tuple network is sometimes called a single layer lookup-table perceptron (SLLUP) [7]. It appears that this network is very similar to the CMAC architecture and, indeed, the CMAC network is a special case of the N-tuple approximation system. The CMAC architecture can be considered as a N-tuple system where N is equal to the length of the address co-ordinate of the K-dimensional address vector produced by the CMAC encoding, and the number of samples taken is K (and so is the number of N-tuple memories), i.e.

$$N_c = D \cdot W = D \cdot \log \left\lceil \frac{L-1}{K} \right\rceil$$

which results in the total memory requirement of $P_c = K \cdot 2^{N_c}$. Additionally, the samples are taken in an ordered fashion, so that a single sample/tuple comprises exactly a single address vector co-ordinate. By introducing random sampling, the CMAC system would be transformed into a classic N-tuple system but, of course, the neighbourhood properties in the new mapping would be different. By using a smaller value of N the SLLUP requires less weight storage than the original CMAC, i.e.

$$P = \frac{D \cdot K \cdot W}{N} 2^N$$

with the reduction with respect to CMAC of

$$\frac{P_c}{P} = \frac{N}{D \cdot W} \cdot 2^{Nc-N} = \frac{N}{N_c} 2^{Nc-N}$$

Therefore, the need for address hashing may be simplified or even omitted. On the other hand, the number of weights contributing to each output computation is increased, thus making the network operation more complex.

For the N-tuple approximation network, using the CMAC algorithm for input co-ordinate encoding will benefit the network's discriminatory properties for an appropriate value of the association coefficient K. The particular choice of this value will be affected by the memory capabilities of the system.

7 Examples

Figs. 6–10 shows the neighbourhoods in the pattern space for a 2-dimensional input and different code types.

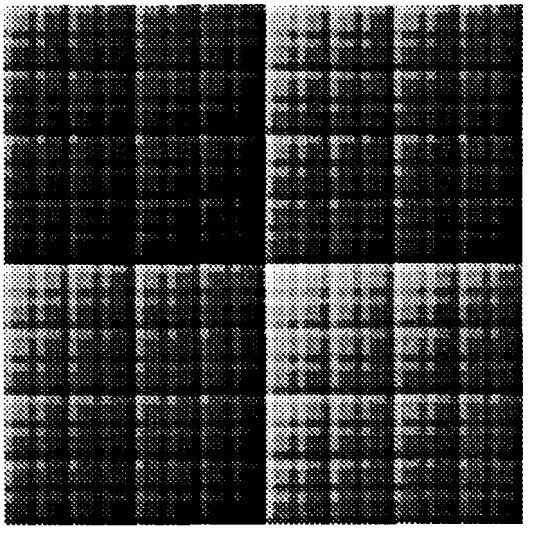

Fig. 6 *Binary encoding*

The input vector variables are integers with $L = 64$ levels each, and the neighbourhood centre has been chosen at the point (32, 32). Darker regions correspond to larger distances. The increased regularity in the shape of the neighbourhood for the CMAC case can clearly be seen.

A number of quality measures have been proposed [8], which express how well a function follows the

monotonicity constraint. One such method relies on accumulating the deviations from a monotonic increase,

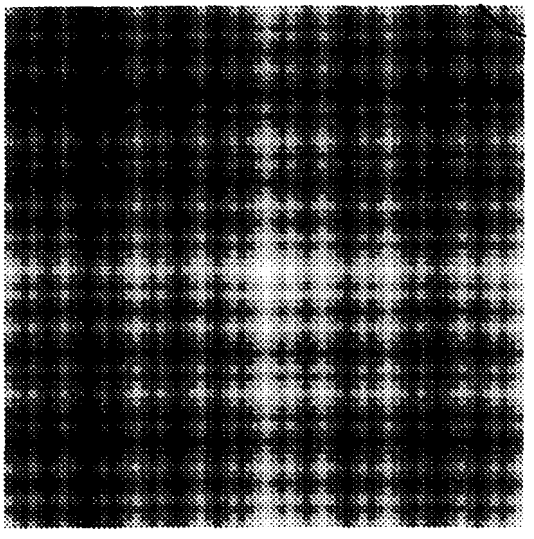

Fig. 7 *Gray code encoding*

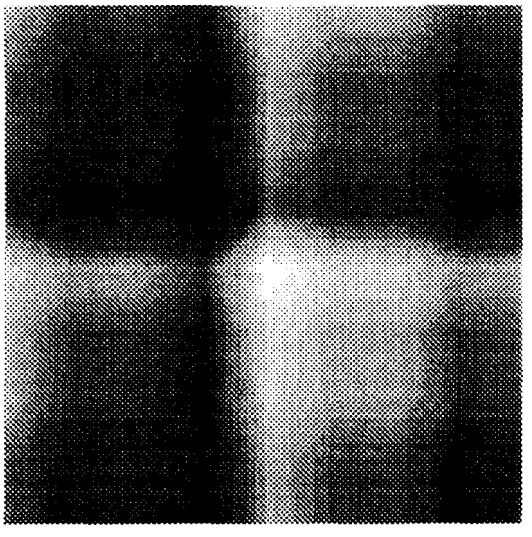

Fig. 8 *CMAC encoding with K = 4*

on a point-to-point basis, for an arbitrary choice of the neighbourhood centre, and then producing an averaged result. This measure can be expressed as

$$\frac{1}{L \cdot W} \sum_{x=0}^{L-1} \left\{ \sum_{y=0}^{x-1} \langle h(C(y-1), Cx) - h(Cy, Cx) \rangle + \sum_{y=x-1}^{0} \langle h(C(y+1), Cx) - h(Cy, Cx) \rangle \right\}$$

$$\langle x \rangle = \begin{cases} x & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases}$$

The higher the value of the measure the lower the quality of the code. Table 2 summarises the measured quality of different codes for the 1-dimensional input ($L = 64$).

Table 2: Code quality comparison

Code	Measure
Binary	5.47
Gray	4.75
Concatenated Gray	2.13
CMAC ($K = 4$)	1.32
CMAC/Gray ($K = 4$)	1.1

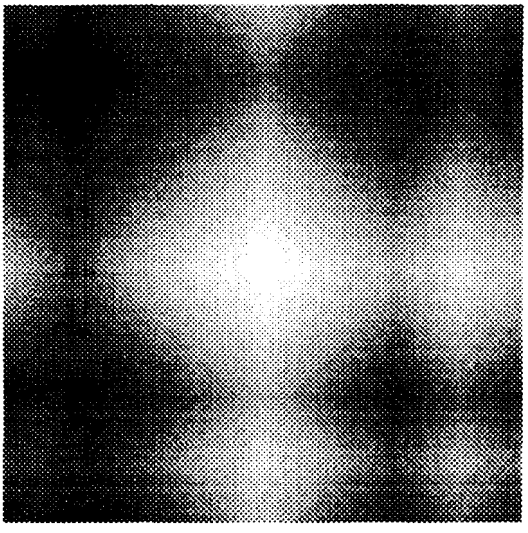

Fig. 9 *CMAC/Gray encoding with K = 4*

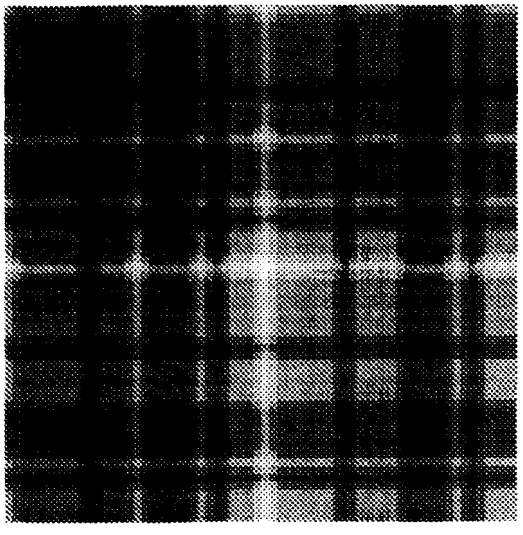

Fig. 10 *Concatenated Gray encoding with K = 2 and shifts of 0 and 1*

8 Conclusions

The CMAC code provides the local monotonic-increasing property in the distance mapping from the input vector space to the pattern space. The radius of the monotonicity neighbourhood around each input space point X is given by the association coefficient K. The use of a Gray code for field co-ordinate encoding is necessary to avoid pattern distance ambiguity between points

inside and outside the neighbourhood. Adjustment of the CMAC association coefficient allows a tradeoff between the size of the monotonicity neighbourhood and the length of the code. Further, the CMAC system is a special case of an N-tuple approximation network. By using random N-tuple sampling it can be transformed into a SLLUP and the hashing part of the CMAC system can be omitted, since the requirement on the memory size is relaxed for small values of N.

9 References

1 ALBUS, J.S.: 'A new approach to manipulator control: the cerebellar model articulation controller (CMAC)', *J. Dyn. Sys. Meas. Control*, 1975, **97**, (3), pp. 220–227

2 ALBUS, J.S.: 'Data storage in the cerebellar model articulation controller (CMAC)', *J. Dyn. Sys. Meas. Control*, 1975, **97**, (3), pp. 228–233

3 ALEXANDER, I., THOMAS, W., and BOWDEN, P.: 'WISARD, a radical new step forward in image recognition', *Sensor Rev.*, 1984, pp. 120–124

4 ALLINSON, N.M., and JOHNSON, J.M.: 'Realisation of self-organising neural map in $\{0, 1\}^n$ space', *in* TAYLOR, J.G., and MANNION, C.L.T. (Eds.): 'New developments in neural computing' (Adam Hilger, Bristol, 1992), pp. 79–86

5 BLEDSOE, W.W., and BROWNING, I.: 'Pattern recognition and reading by machine'. Proceedings of IRE Joint Computer Conference, 1959, pp. 225–232

6 KOLCZ, A., and ALLINSON, N.M.: 'Reconfigurable logic implementation of memory-based neural networks: a case study of the CMAC network'. Proceedings of the 3rd international workshop on VLSI for Neural Networks and Artificial Intelligence, 1992

7 TATTERSAL, G.D., FOSTER, S., and JOHNSTON, R.D.: 'Single-layer lookup perceptrons', *IEE Proc. F*, 1991, **138**, pp. 46–54

8 TATTERSAL, G.D., FOSTER, S., and JOHNSTON, R.D.: 'Single-layer lookup perceptrons'. Patent Application 8906558.5, March 1989