Fall 1991

An improved multi-dimensional CMAC neural network: Receptive field function and placement

Pak-Cheung Edgar An
University of New Hampshire, Durham

Order Number 9200570

An improved multi-dimensional CMAC neutral network: Receptive field function and placement

An, Pak-Cheung Edgar, Ph.D.

University of New Hampshire, 1991

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

An Improved Multi-Dimensional CMAC Neural Network: Receptive Field Function and Placement

BY

Pak-Cheung Edgar An

B.S.E.E. University of Mississippi, May 1985
M.S.E.E. University of New Hampshire, May 1988

DISSERTATION

Submitted to the University of New Hampshire
in partial fulfillment of
the Requirements for the Degree of

Doctor of Philosophy

in

Engineering

September, 1991

This dissertation has been examined and approved.

Dissertation director, W. T. Miller III
Professor of Electrical and Computer Engineering

Filson H. Glanz, Professor of Electrical and Computer Engineering

Michael Carter, Assistant Professor of Electrical and Computer Engineering

Lee Zia, Associate Professor of Mathematics

Scott Mcintire, Assistant Professor of Mathematics

7/14/91

Date

DEDICATION

To Amy and Ethan

ACKNOWLEDGEMENTS

I would like to thank Dr. T. Miller, Dr. F. Glanz, Dr. M. Carter for their critical suggestions. I also would like to thank my wife, Amy, for her mental support and help in proofreading my thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

$\lvert\underline{W}\rvert$	Total Available Weight Storage
β	Learning Rate
C	Input Generalization
$CMAC_e$	CMAC Grouped from Even Layers of Receptive Fields
$CMAC_o$	CMAC Grouped from Odd Layers of Receptive Fields
C^N	N-dimensional Reference Hypercube
d_i	Displacement Offset in i^{th} Input Dimension
$\underline{D}$	Displacement Offset Vector $<d_1, \cdot\cdot d_N>$
$f_i(X)$	i^{th} receptive Field Strength for input X
HR1-3	Heuristic Center Placement Rules
I	Integer-Value Domain
N	Input Dimension
P1	2D Uniform Center Placement (figure 4.4)
P2	Center Placement with Non-Uniform Field Widths and Offsets (figure 4.6)
R	Real-Value Domain
σ_i	i^{th} Receptive Field Width
S	Number of Input Quantization Intervals
S^N	N-Dimensional Quantized Input Space
W_i	Weight Value of ith Receptive Field
X	Network Input ε R^N
Y(X)	Network Output ε R
$Y_d(X)$	Desired Network Output ε R

ABSTRACT

AN IMPROVED MULTI-DIMENSIONAL CMAC NEURAL NETWORK:
RECEPTIVE FIELD FUNCTION AND PLACEMENT

by

Pak-Cheung Edgar An

University of New Hampshire, September, 1991

The standard CMAC has been shown to have fast learning computation as a result of modular receptive field placement, rectangular receptive field shape and a simple weight adaptation algorithm. The standard CMAC, however, suffers from slow convergence at some critical frequency due to the rectangular receptive field shape. A linearly-tapered field, which requires a uniform placement, was used in this research. The receptive field placement of the standard CMAC becomes less uniform locally for a larger receptive field width. This dissertation suggests a new field placement which is more uniform without extra computation. Results show that the slow convergence at the critical frequency is eliminated, and the interaction of the linearly-tapered field with the new placement achieves more accurate function approximation. A theoretical bound on the receptive field width as a function

of the input dimension is proposed if a uniform placement is to be achieved. Also, a procedure for adapting receptive field density to minimize the weight usage for a given approximation accuracy is suggested.

CHAPTER I

INTRODUCTION

Artificial neural networks have been widely studied and applied in the areas of robotics control, signal processing and pattern recognition as these networks are capable of approximating complex system dynamics better than most traditional parameter estimation techniques [15]. In general, the study of artificial neural networks is different from the study of biological neural networks in that the former places its interest in modelling system behavior while the latter places its interest in modelling biological behavior [7]. This dissertation is concerned with the issues of localized receptive field function and placement for an artificial neural network called the Cerebellar Model Arithmetic Computer (CMAC). Results indicate that both the uniformity of the receptive field function and placement play an important role in approximating a desired output in a multi-dimensional input space (R^N).

Figure 1.1 shows a general artificial neural network architecture for a scalar input (X) and a scalar output (Y(X)). X and Y(X) are connected through multiple layers of processing elements. Each broken line indicates a possible connection between processing elements, and the connectivity strength is determined by an associated weight value (W). The

output of each processing element is a non-linear function of its weights and inputs. The common non-linear input-output relationships are the sigmoidal function, the binary threshold function and the linear threshold function (figure 1.2).

All existing neural networks can be categorized as either globally generalizing or locally generalizing. The network generalization is considered global if one or more of the adaptable parameters in the network, such as weights and biases, can potentially affect the network output at every point in the input space. On the other hand, the network generalization is considered local if only a small subset of the adaptable parameters can potentially affect the network output in a local region of the input space. In other words, the locally generalizing network can be viewed geometrically as a set of localized receptive fields (with finite region of support) distributed in R^N in which each localized receptive field will be excited when the input falls in its region.

While the network shown in figure 1.1 is commonly described as globally generalizing (assuming all processing elements are fully connected), the multi-layered network can behave as if it were made of localized receptive fields given a particular set of weights and biases (θ, which provides function shifting). Figure 1.3 shows how processing elements can be transformed into a set of localized receptive fields for a scalar input. Assume that the output of each processing element is a linear threshold function of its input, $\Sigma W_i X_i$ -

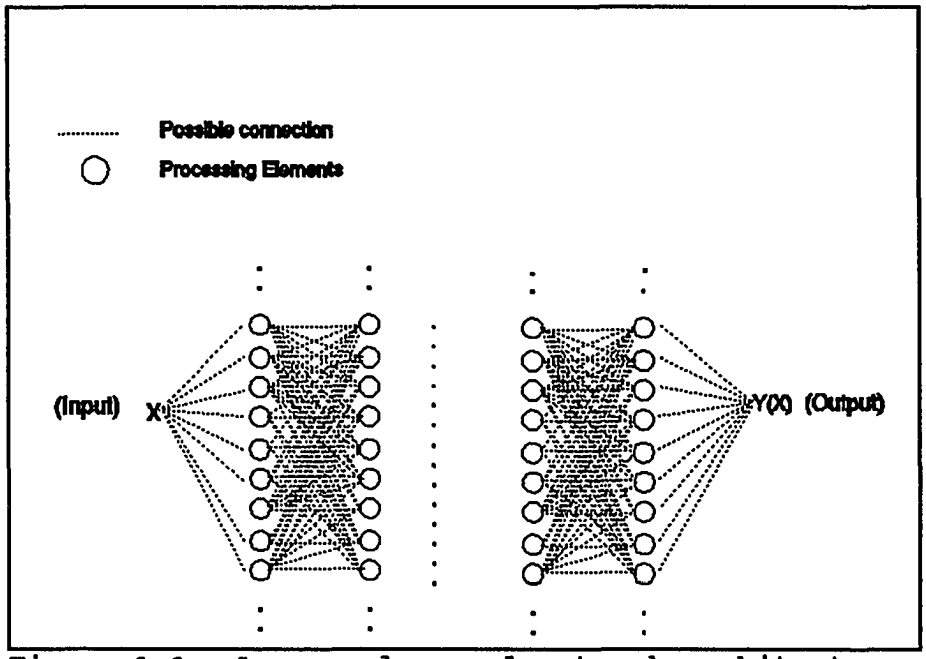

Figure 1.1 A general neural network architecture. X and Y(X) are the network input and output respectively.

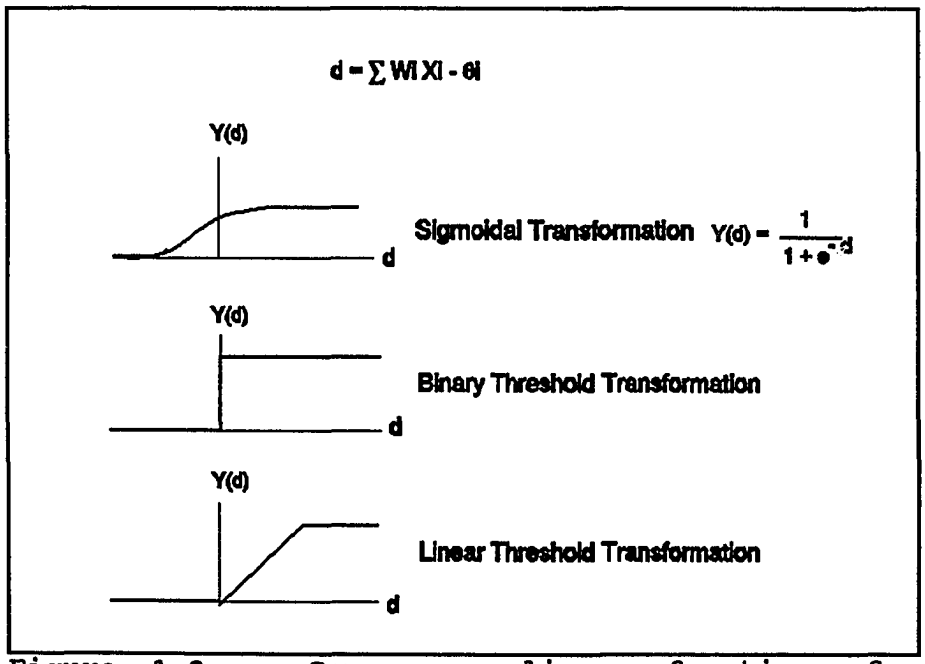

Figure 1.2 Common non-linear functions for processing elements.

θ_i. By setting W_i and θ_i appropriately as shown in figure 1.3, the processing elements in the second layer respond as localized receptive fields in which only one processing element in the second layer will have non-zero output for any input. A similar concept can be applied to construct localized receptive fields in a higher dimensional input space.

Compared to locally generalizing networks, globally generalizing networks have three major undesirable features in terms of learning computation and convergence. First, the time for adjusting weights and computing a network output is much longer because all the adaptable parameters must be adjusted for each training pair. Second, when the training is incremental (only one training sample is used at a time), there is a global learning interference between network outputs for any distant network inputs because of the non-locality of basis functions synthesized by the network. In order to reduce the learning interference, each weight must be adjusted in a small step, which causes the learning convergence to be sluggish. Last, there is a lack of a geometrical understanding of how to adjust the weights in those intermediate layers. The most commonly used technique for implementing gradient descent weight adjustment in a layered network is called back-propagation, in which an error signal propagates back through layers of processing elements during learning [24]. Since the relationship between the approximation error and the weights in the network can be

highly complicated, the weights often converge to yield a local minimum of the approximation error in which only a sub-optimal performance is achieved.

On the other hand, locally generalizing networks require fewer numerical computations per training cycle to generate network outputs because only a small portion of weights need to be adjusted in each training cycle. Also, because of the localized receptive field arrangement, the network outputs associated with distant inputs are independent. This eliminates any learning interference between network outputs associated with distant inputs. Each weight can thus be adjusted in a larger step, which results in faster learning convergence. Further, as the fixed weights in the first two layers (figure 1.3) are used to construct localized receptive fields, only the weights in the output layer need to be adjusted. Thus, the relationship between the approximation error and the weights is well behaved, and an optimal learning performance can be achieved. Based on these considerations, this dissertation studied only the locally generalizing networks. In other words, the receptive fields in these networks are strictly localized (with finite region of support).

In general, learning performance is best measured by the training error (how well a network can approximate the given training data), the generalization error (how well the network can generalize given data not in the training set) and the

speed of convergence. Clearly, the generalization error depends on the data distribution [25]. This dissertation focuses on the training error, the generalization error and the speed of convergence.

Chapter II reviews basic characteristics of four major locally generalizing networks: the Cerebellar Model Arithmetic Computer (CMAC), the Kohonen Map (KOH), the Sparse-Distributed-Memory (SDM) and the Radial Basis Functions (RBF). For these locally generalizing networks, the learning performance is highly dependent upon the receptive field function and placement in R^N. Although each network has its own unique features, this dissertation is specifically focused on the issues of receptive field function and placement in CMAC. The CMAC network output computation is inherently faster than other networks (refer to chapter II), which is highly desirable in a real-time control environment.

Chapter III discusses some undesirable features of the standard receptive field function and placement of CMAC in learning [5]. Chapter IV suggests an alternative CMAC architecture which preserves the desirable features but minimizes the undesirable features. Chapter V discusses some learning issues using multiple CMACs and a single CMAC with adaptive receptive field density (defined later). Chapter VI summarizes the effectiveness of the improved CMAC architecture in learning and offers suggestions for future research.

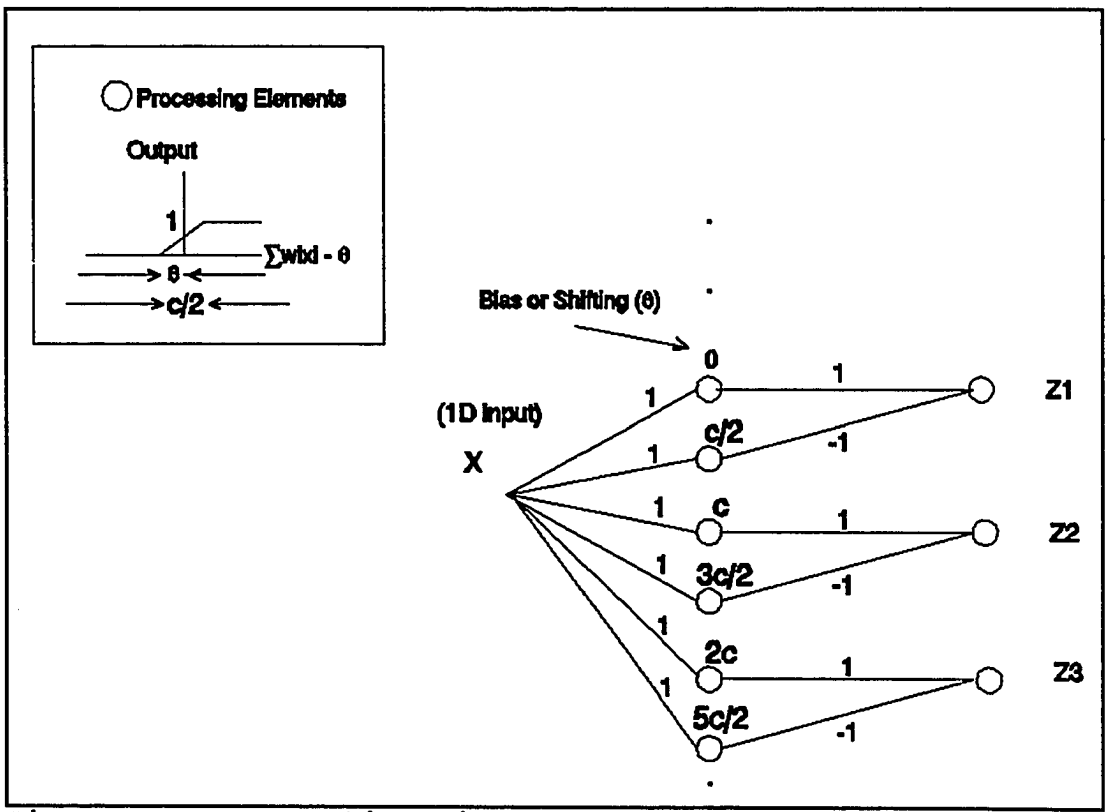

Figure 1.3 Receptive fields can be constructed by using a multi-layered network if weights and biases are chosen properly.

CHAPTER II

LOCALLY GENERALIZING NETWORKS

This chapter presents the physical architecture, the method of weight adaptation and the network function of four locally generalizing networks. Section II.1 presents the Cerebellar Model Arithmetic Computer (CMAC). Section II.2 presents the Kohonen Map (KOH). Section II.3 presents the Sparse-Distributed-Memory (SDM). Section II.4 presents the Radial Basis Functions (RBF). Finally, Section II.5 provides a summary comparison of these networks in terms of the network architecture and learning. As mentioned in Chapter I, the receptive fields in these networks are strictly localized. In other words, any receptive field with infinite region of support is avoided.

II.1 Cerebellar Model Arithmetic Computer (CMAC)

Albus proposed the Cerebellar Model Arithmetic Computer (CMAC) architecture to model the localized activities in the cerebellum [1][2][3]. Miller [17][18] later applied CMAC to a real-time closed-loop control problem. Functionally, CMAC is capable of approximating any arbitrary function within the desired input region.

Figure 2.1 shows the physical CMAC structure made of an

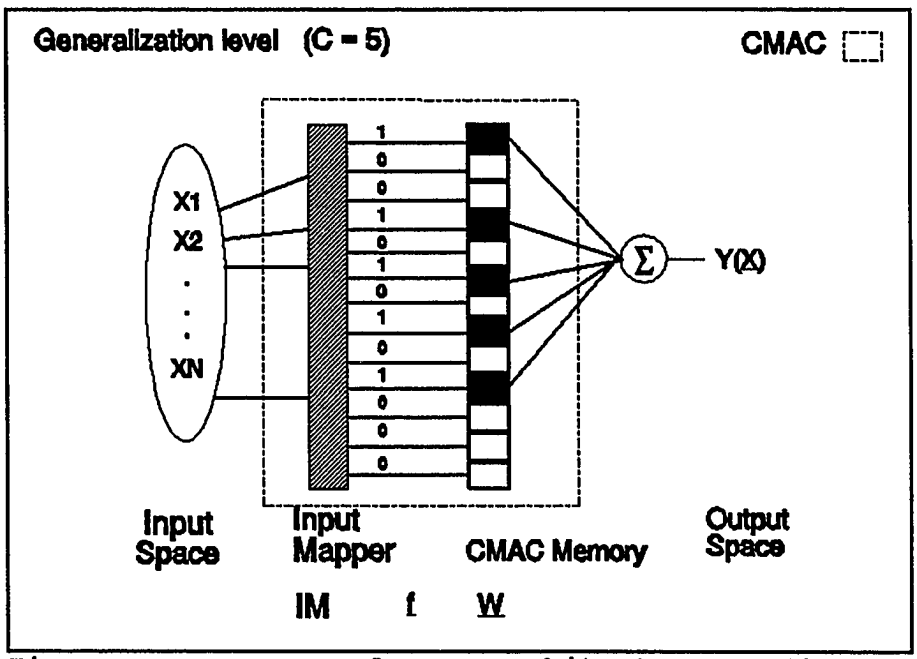

Figure 2.1 A general CMAC architecture. X_1 through X_N are input variables. $Y(\underline{X})$ is a scalar network output as indicated by a single $\underline{W}$. (CMAC_BLK.WPG)

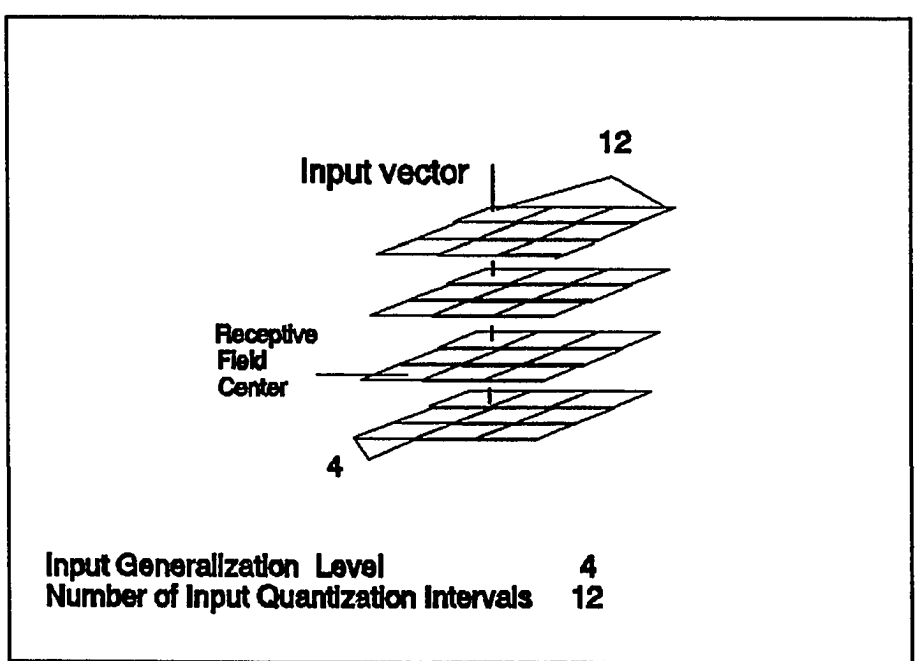

Figure 2.2 A CMAC Input Mapper.

input mapper (IM) and a weight vector ($\underline{W}$). These weights are used to reconstruct the desired network output after observing enough input-output data. IM is fixed *a priori*, and only $\underline{W}$ can be modified during learning. Given an input, $\underline{X}$, of dimension N, the output of IM, $\underline{f}$, has exactly C components of 1 and the remaining components of zero. The value of C defines the input generalization, and is fixed *a priori*. A network output, Y, is then defined as an inner product of $\underline{f}$ and $\underline{W}$ ($\Sigma W_i \cdot f_i$). In the case in which Y is a vector, each individual output (Y_i) will have its corresponding vector $\underline{W}_i$. This guarantees that the network outputs are totally independent.

Although the size of the vector $\underline{W}$ ($|\underline{W}|$) can be large (order of thousands), learning computation can still be fast because only C weights are adjusted for each training sample, and C is usually much smaller than $|\underline{W}|$.

IM computes the number of weights shared between any two inputs ($\underline{X}_1$, $\underline{X}_2$) approximately as C minus their Manhattan distance ($\Sigma\ |X_{1i} - X_{2i}|$). If the Manhattan distance is greater than C, no weights will be shared. In doing so, similar inputs produce similar outputs and dissimilar inputs produce independent outputs.

II.1.A Input Mapper

Figure 2.2 shows an example of a two-dimensional input mapper (IM). IM is made up of layers of receptive fields. Each receptive field is represented by a square in each layer, and

its associated weight is stored in $\underline{W}$. S^N is defined as a quantized input space in which each axis is quantized into S intervals (12 in this example). For the example in figure 2.2, the input generalization is 4 input quantization intervals wide, and each receptive field occupies 4X4 input quantization intervals.

In the standard CMAC, the input generalization is equal to the number of layers of receptive fields. Thus, there are four layers of receptive fields in figure 2.2. For any input, IM excites only one receptive field in each layer. This guarantees that exactly C receptive fields will respond to each training sample. The larger the value of S, the higher the input resolution, and vice versa.

While each receptive field responds only to a coarse input sub-region, the overall network resolution can be increased by offsetting each consecutive layer by one input quantization interval in each axis. Figure 2.3 shows all the centers of receptive fields superimposed onto a single layer. With the layer offset, 57 receptive fields (represented by circles) are required to cover the entire quantized input space (S^2). One can easily extend the mapping technique to a multi-dimensional input space (S^N) by treating each layer as a hyper-layer, and each receptive field as a hypercube of side C input quantization intervals. <u>C^N is defined as a reference hypercube of side C in which C individual receptive fields from each hyper-layer are superimposed</u>. These C individual

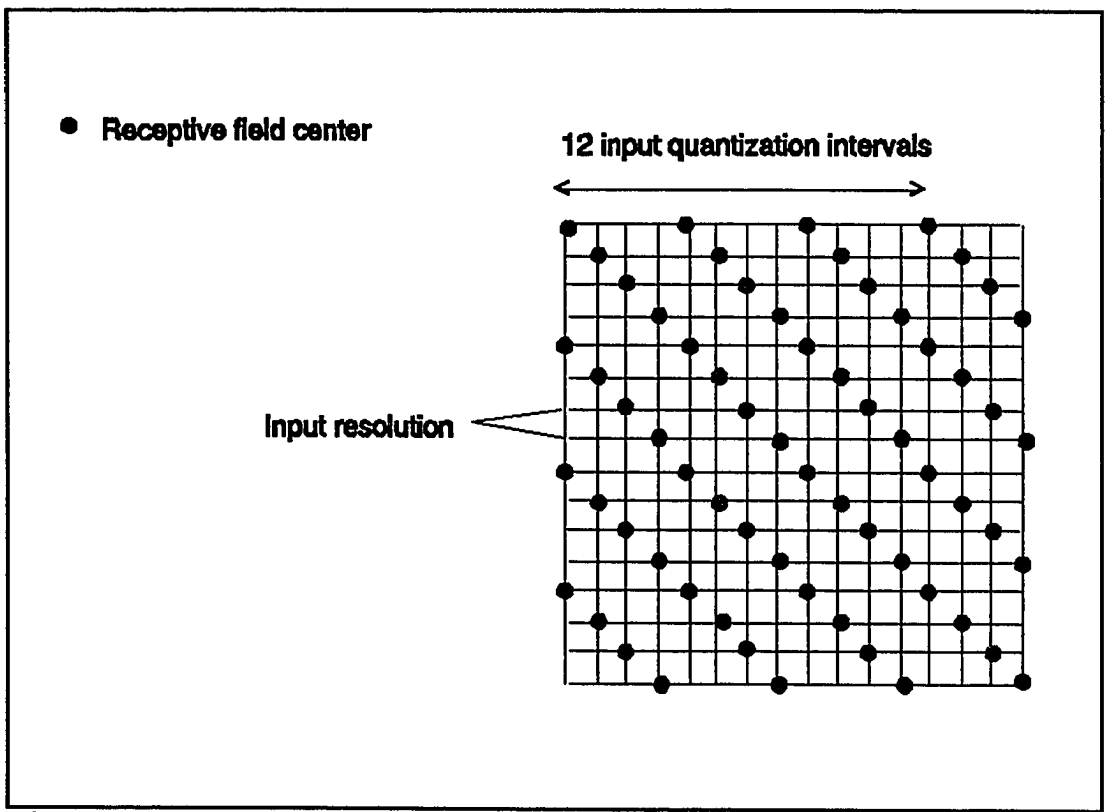

Figure 2.3 Superimposed receptive field center placement for a two-dimensional input.

receptive fields are each located in the same physical space as the reference hypercube (C^N).

The total number of receptive fields required (NRF) can be computed by using equation 2.1. If S+C-1 is divisible by C, the second term on the right hand side will become zero ($\alpha=\gamma$). NRF is then reduced to $(S+C-1)^N/C^{N-1}$ (the first term on the right hand side). If S+C-1 is not divisible by C, S+C-1 must lie somewhere between α and γ. Thus, extra receptive fields can be accounted for by using linear proportionality (the second term on the right hand side).

$$NRF = \alpha^N/C^{N-1} + \{\gamma^N/C^{N-1} - \alpha^N/C^{N-1}\}\cdot P \qquad \text{Eq. 2.1}$$

where $\alpha = \lfloor (S+C-1)/C \rfloor \cdot C$,

$\gamma = \lceil (S+C-1)/C \rceil \cdot C$,

$P = (S+C-1-\alpha)/C$.

If only the receptive fields of finest resolution (centers at each intersection of the input quantization intervals) were used, there would have been C^N number of receptive fields inside the reference hypercube of side C. By using hyperlayers of coarse receptive fields, the receptive field density (defined as the number of receptive fields inside the reference hypercube) is reduced by a factor of $1/C^{N-1}$. As a result, the receptive field coverage becomes more sparse for a larger N or C (except when N is 1). By superimposing the centers as in figure 2.3, centers of

receptive fields are distributed on the hyperdiagonal in C^N. Although the overall field coverage appears uniform in S^N, increasing C causes the field coverage to be less uniform inside C^N.

It is worth noticing that CMAC can only learn to approximate a network function in a finite input region because of the finite number of available weights, and more importantly, because of the finite support of the receptive field function used. If S and C are chosen to be 100 and 32 respectively in a 10-dimensional space (a reasonable size for robotics control problems), 59,705,579 receptive field centers will be required for learning in S^N. Fortunately in most control problems, the desired network output tends not to span the entire input space. Although the possible input space can be very large, the input space actually traversed is often much smaller. Therefore, a hashing technique can be applied to $\underline{W}$ to further reduce the number of weights. The hashing might deteriorate learning as it could cause undesirable collisions. A collision occurs when two distant inputs share the same weight. While there has not been any extensive research done on learning interference due to collisions, cumulative results have suggested that the effect of hashing on learning is minimal unless the frequency of collisions is high [17][18].

II.1.B Adaptation

The vector $\underline{W}$ is initialized to zero prior to learning.

When given a training sample $\{\underline{X}, Y_d(\underline{X})\}$, C receptive fields are excited, and their weights are summed to form a network output, $Y(\underline{X},k)$, where k is the number of training cycles iterated. These C weights are then adjusted by means of the gradient descent method shown in equation 2.2.

$$Y(\underline{X},k) = \sum_i W_i(k)$$

$$W_i(k+1) = \sum_i W_i(k) + (\beta(Y_d(\underline{X}) - Y(\underline{X},k)) / C \qquad \text{eq. 2.2}$$

where β is the learning rate, and $W_i(k)$ is one of C weights associated with the input $\underline{X}$. Usually, β is set less than 1 for smoother convergence.

II.2 Self-Organizing Kohonen Network

Kohonen proposed a self-organizing feature map to model the human visual and auditory cortex. Functionally, this map is capable of approximating continuous input statistics, and its primary applications are found in pattern clustering and classification [11][12][13][14].

In general, the self-organizing map contains a set of receptive fields arranged in a two-dimensional layer (figure 2.4). Each receptive field i has a weight vector ($\underline{W}_i$) of N components (W_{i1}, W_{i2},..W_{iN}), where N is the input dimension. As the mapping from the input ($\underline{X}$) to the receptive fields varies during learning, a search process is required to locate the "winner" receptive field (chosen to be a representative for the input).

The search is done by computing the Euclidean distance between $\underline{X}$ and $\underline{W}_i$ for each receptive field center. The "winner" receptive field is then chosen to be the one with the smallest Euclidean distance. After locating the "winner" receptive field, a fixed number of neighboring receptive fields (C) in the two-dimensional layer are also located. It is important to notice that these neighboring centers are only the "winner" center's neighbors in the two-dimensional layer, but not in the N-dimensional input space (R^N) . While a standard neighborhood region has not been defined, it is often chosen

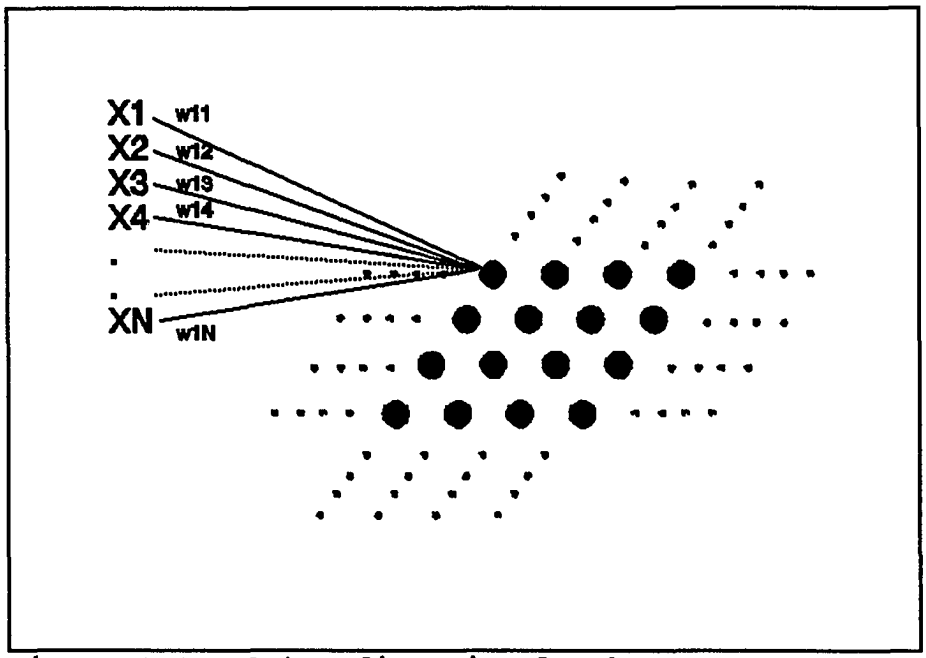

Figure 2.4 A two-dimensional Kohonen Map.

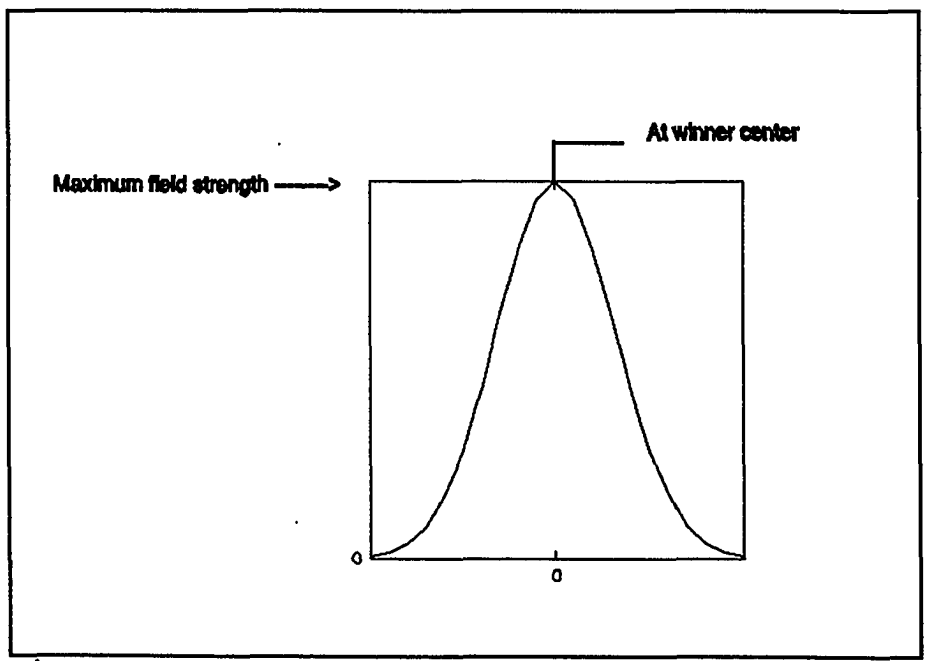

Figure 2.5 A Gaussian Pseudo receptive field function.

to be either hexagonal or rectangular in shape.

Prior to learning, the weight components of each receptive field are initialized with small random values. When the input ($\underline{X}$) is presented, the "winner" receptive field and its neighboring receptive fields are located. These weight vectors are then adjusted according to equation 2.3.

$$\underline{W}_i(k+1) = \underline{W}_i(k) + \beta\ f(i)\ (\underline{X} - \underline{W}_i(k)) \qquad \text{eq. 2.3}$$

where β is the learning rate and f(i) is the pseudo-receptive field function (figure 2.5). $\underline{W}_i(k)$ is the weight vector of receptive field i. The index i refers to the distance (in an integer metric) from the "winner" center in the two-dimensional layer. $\underline{W}_0(k)$ is the weight vector of the "winner" receptive field.

The name "pseudo" is used because the receptive field strength does not depend on the Euclidean distance between $\underline{X}$ and $\underline{W}_i$ in R^N, but rather it depends on the Euclidean distance in the two-dimensional layer. The pseudo receptive field function is tapered so that receptive fields closer to the "winner" receptive field receive larger weight updates than those farther away. The "winner" receptive field receives the maximum weight update. In practice, proper training requires β be very small (on the order of 0.01), which in turn requires a large number of training samples for proper convergence.

This type of learning is commonly categorized as

unsupervised or competitive because there is no explicit desired network output available for learning. The weights are adjusted only to form the mapping from input to receptive fields. An important observation from this adaptation rule is that as the training inputs are presented randomly, the weights will gradually self-organize in either an ascending or descending order so that no two distant centers will respond to the same input [14]. During the retrieving cycle with a testing input, only the output of the winner receptive field is 1, while the outputs of the others are zero.

II.3 Sparse-Distributed-Memory (SDM)

Kanerva proposed a model of human memory called Sparse-Distributed-Memory (SDM) [10]. SDM is particularly well suited to learning a set of iterated sequences for which the current network output becomes the next network input. SDM can also be constructed to perform function approximation. Both the input ($\underline{X}$) and the output ($\underline{X}'$) in SDM are binary vectors of dimension N ($\{0,1\}^N$). Usually, N is set as high as 1000 for proper learning.

In the case where N is 1000, the total number of possible inputs is 2^{1000} (M). In practice, the number of receptive fields available (M') is much smaller (10^6). In addition, the field placement is usually not deterministic. As a result, all of the receptive fields are sparsely and randomly distributed in $\{0,1\}^N$.

Each input excites a number of receptive fields (C) in its neighborhood. The neighborhood is a hypercube with its two farthest corners R apart (Hamming distance). R is a parameter which determines the input generalization. Because of the random placement of receptive fields, a search is required to locate these neighboring receptive fields. This can be done by computing the Hamming distance between $\underline{X}$ and each receptive field center. Any receptive fields with centers less than distance R from $\underline{X}$ are excited. A smaller R results in smaller

generalization, and vice versa. It is worth mentioning that the value of C might differ from one input to another because of the random field placement and the Hamming distance constraint.

The neighborhood distance threshold (R) and the centers' locations are fixed prior to learning. Only the receptive field contents can be modified during learning.

Given a desired input-output pair in each training cycle, SDM searches for receptive field centers close to $\underline{X}$ by measuring their Hamming distance. The desired output is then stored in the excited receptive fields, assuming that these receptive fields provide enough storage for learning. For example, if there are 10,000 inputs and 10^6 receptive fields, each receptive field will store an average of 10 inputs, given that the average value of C is 1000.

After enough training, each receptive field contains copies of all the previous training inputs within its receptive field. Given a testing input, the Hamming distance is again computed to locate these neighboring centers. $\underline{X}'$ is then generated according to the majority rule for which each individual output bit is the thresholded bit-sum of all the data stored in all C excited receptive fields. $\underline{X}'$ is the current network output, and becomes the network input in the next testing cycle. The threshold level for the bit-sum is another parameter to be chosen prior to retrieval. The ability to retrieve a correct output depends on the overlap between

the hypercube centered at the desired input and one at the testing input.

An important observation from this adaptation rule is that if the testing input ($\underline{X}$) is close to the desired input ($\underline{X}_d$), then $\underline{X}'$ is even closer to the desired output of $\underline{X}_d$, $\underline{X}_d'$, than $\underline{X}$ to $\underline{X}_d$ (figure 2.6). In cases in which the network has been trained to reproduce sequences (the current X' is the next X), the sequence formed by each network output will converge to the desired sequence. Kanerva introduced a critical distance (D_{cr}) at which the sequence will diverge from the desired one if the testing input is more than D_{cr} away from the desired input. D_{cr} is a function of the number of words stored and the neighborhood size [10].

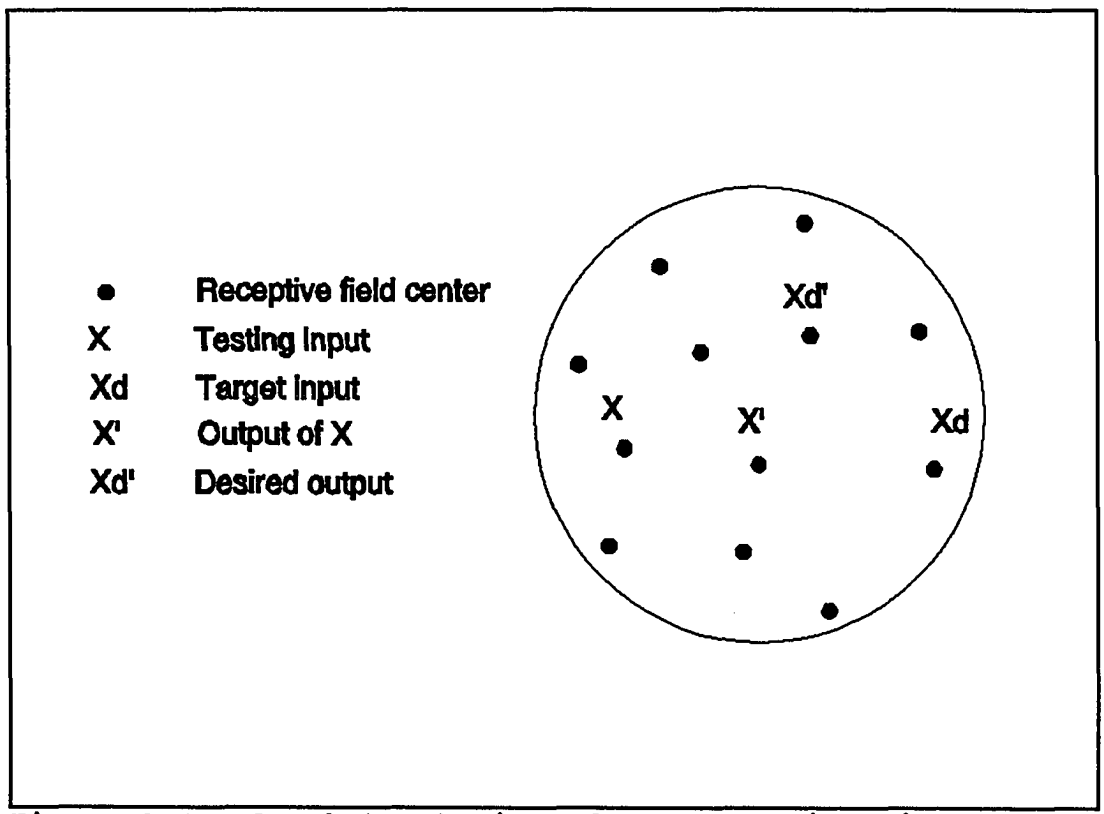

Figure 2.6 An abstract view of SDM receptive field center placement for an N-dimensional binary input.

II.4 Radial Basis Functions (RBF)

Radial Basis Functions are derived from a well known regularization technique, and is capable of performing function approximation given a set of training data and some prior knowledge about the surface smoothness of the desired network output [22][23].

Figure 2.7 shows an example of a one-dimensional RBF (one-input-one-output) network composed of linearly tapered receptive fields with non-uniform field widths and field locations. Thus, different inputs might excite different receptive fields. As the field locations and field widths adapt during learning, a search is required to locate these neighboring receptive fields because each receptive field is assumed to be strictly localized (with finite region of support).

L_i and σ_i stand for the ith field location and the ith field width respectively. When the input (X) is presented, each receptive field computes its Euclidean distance (d_i) between X and L_i. Centers of receptive fields with Euclidean distances less than σ_i are excited. The strength of receptive field i is $f_i(d_i(X),\sigma_i)$. The network output (Y) is then defined as a weighted sum of these individual non-linear receptive field functions.

$$Y = \sum_i W_i \cdot f_i(d_i(X), \sigma_i)$$

where W_i is a weight value associated with receptive field i.

Given that an input falls in receptive field i during learning, W_i, L_i and σ_i are adjusted by means of the gradient descent method. Other rules of adjusting these parameters are also available. Each individual variable is adjusted while keeping the other variables fixed. In batch learning (adjust variables based on all the given training pairs), the network cost function is defined as a sum of all the squared errors for all the training samples (equation 2.4). Some smoothness constraint on the function surface can also be incorporated in the cost function [22]. W_i is then updated according to equation 2.5. σ_i and L_i are updated according to equation 2.6 and 2.7 respectively.

$$e_j = Y_d(X_j) - \sum_i W_i \cdot f_i(d_i, \sigma_i) \qquad \text{eq. 2.4}$$

$$J = (1/2) \cdot \sum_j e_j^2$$

$$\delta W_i(k) = \sum_j \beta \cdot e_j \cdot f_i(d_i(X), \sigma_i) \qquad \text{eq. 2.5}$$

$$\delta \sigma_i(k) = \sum_j \beta \cdot e_j \cdot W_i \cdot (\partial f_i(d_i, \sigma_i) \ / \ \partial \sigma_i) \qquad \text{eq. 2.6}$$

$$\delta L_i = \sum_j \beta \cdot e_j \cdot W_i \cdot (\partial f_i(d_i, \sigma_i) \ / \ \partial L_i) \qquad \text{eq. 2.7}$$

where β is the learning rate [22][23].

In incremental learning (where only one training pair is used at a time), these equations can still be used by dropping the first summation sign on the right hand side.

The derivative of the radial basis function is only required when the field width or the field location needs to be adjusted. In practice, the initial field coverage is uniform (assume no prior knowledge about the input data distribution) and the learning rate is small. The choice of $f_i(d_i, \sigma_i)$ is crucial to learning. A well-known radial basis function is of Gaussian shape [22].

The learning performance of the network is often not optimal because the gradient surface with respect to these parameters usually contains multiple local minima.

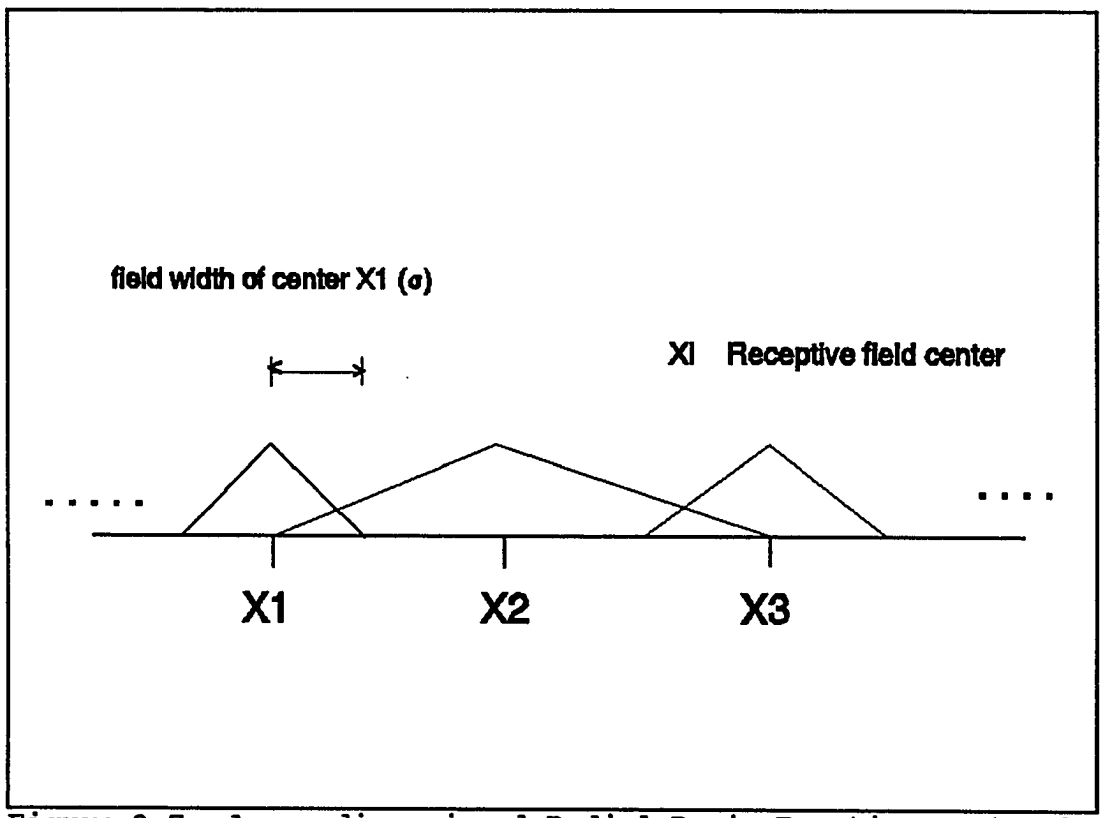

Figure 2.7 A one-dimensional Radial Basis Functions network.

II.5 Discussion

Six fundamental characteristics are summarized for the CMAC, KOH, SDM and the RBF networks as follows.

Supervised vs. Unsupervised - The RBF, CMAC and SDM networks are classified as supervised networks because of the availability of the desired network outputs. This allows the network to approximate a function. On the other hand, the KOH network is classified as an unsupervised network because a desired network output is not available. Its main use is to perform pattern clustering or classification.

Localized Receptive Field - All of these networks use localized receptive fields, which have properties of local generalization in the formation of network responses. That is, similar inputs produce similar outputs, while distant inputs produce independent outputs.

Receptive Field Width - The receptive field widths in the standard CMAC and SDM are fixed during learning. On the other hand, the field widths in RBF and KOH are adaptable during learning. RBF adapts the field widths based on the function approximation error while KOH adapts the field widths (in R^N) based on the input distribution in R^N.

Receptive Field Placement - The receptive field placement in both CMAC and SDM are fixed during learning. The field placements in CMAC are modular in S^N but the field placements in SDM are random in $\{0,1\}^N$. On the other hand, the receptive field placements in KOH and RBF are adaptable during learning. RBF adapts the field locations based on function approximation error, while KOH adapts the field locations based on the input distribution in R^N. Thus, CMAC is similar to SDM in terms of the fixed receptive field placement and field width (except the placement in SDM is irregular). However, hashing (without any collisions) in CMAC can be considered as an adaptive placement of available receptive fields based on the input distribution, like KOH.

Search For Neighboring Fields - In general, the number of receptive fields required for a network to approximate to a given accuracy grows exponentially with the dimensionality N. Thus, the search time for neighboring receptive fields in SDM and RBF increases with the input dimension. In contrast, the modular receptive field placement in CMAC eliminates any search for neighboring receptive fields. While the irregular (SDM) or adaptable (KOH, RBF) receptive field placement might be better models for biological systems, the search process is highly undesirable in real-time control applications.

Weight Adjustment - The weight adjustment during learning in

all of these networks utilizes the gradient descent method or a variation. As only a small number of weights are adjusted at a time (localized receptive field concept) using a simple weight update law and a relatively large learning rate (except KOH), the numerical computation is potentially fast (without considering the searching process). However, in RBF, not only the weights but also the field locations and the field widths need to be adjusted in each training cycle. Very often, the error surface with respect to these variables contains local minima, which can lead to a sub-optimal performance.

CHAPTER III

RECEPTIVE FIELD FEATURES IN THE STANDARD CMAC

While the standard CMAC is capable, in general, of approximating any arbitrary function, it is useful in practice to determine which training functions are more difficult to approximate. The level of difficulty is related to issues of the speed of convergence and the final approximation error.

This chapter presents sinusoid and impulse learning experiments and results for the standard CMAC. The corresponding desired outputs were space-spanning in nature. Because of the local generalization feature, the network output does not relate linearly to the network input. On the other hand, the network output is linear relative to the network weights. Suppose $\underline{W}_i$ generates $Y_i(X)$ and $\underline{W}_1+\underline{W}_2+\cdots\underline{W}_i$ generates $Y_1(X)+Y_2(X)+\cdots Y_i(X)$. If each $\underline{W}_i$ contains proper weight values approximating the ith Fourier component of the desired output, any arbitrary periodic function can thus be approximated by summing all the individual weight vectors (W_i). Any difficulty in approximating a particular Fourier component of the desired output will lend useful insights in approximating the desired output as a whole, which justifies the use of sinusoids of different frequencies in learning.

Section 1 of this chapter presents the experiments for a one-dimensional input. Section 2 presents the experiments for

a two-dimensional input. Each analysis in the following sections was performed under the following conditions. There were 50 input quantization intervals (S) in each axis. The input generalization was 10 input quantization intervals wide, and the learning rate (β) was set to 0.5. All the training pairs were obtained by sampling at each input quantization interval (and their intersection) in S^N. All the training samples then formed a training set. Thus, there were S^N possible independent training samples in the training set. Each sample input, X, could take any value from 0 to 1.

III.1 One-Dimensional Input

III.1.A Sinusoid Learning Experiment

In this experiment, the desired output was $\sin(2\pi MX)$, where M is the harmonic number. Ten sinusoidal functions of different spatial frequencies (the first harmonic to the tenth harmonic) were trained separately. The spatial wavelength of the first harmonic function was 50 input quantization intervals wide. In each training cycle, a sample was randomly chosen from the training set. The training ended when the standard deviation of the training error (the difference between the desired output and the network output at the training samples) was less than 0.1. Figure 3.1 shows the relationship between the number of training cycles needed to reach 0.1 RMS error and the function harmonic.

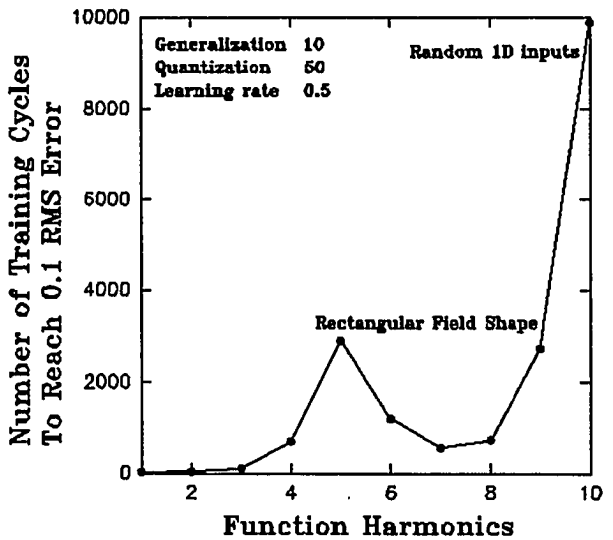

Figure 3.1 A relationship between the number of training cycles and the function harmonic for a scalar input.

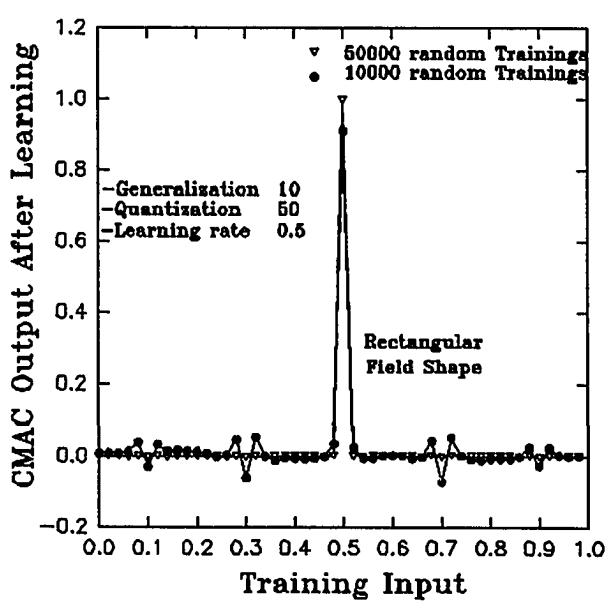

Figure 3.2 An CMAC input-output relationship for a desired impulse output.

The learning convergence was sluggish when functions reached integer multiples of a critical spatial frequency (f_{cr}) [5]. f_{cr} is defined as S/C (the fifth harmonic in this case). Also, the time to converge, in general, increased with the harmonic number, except near f_{cr}.

The slow convergence can be explained by examining the weight update at f_{cr} (the fifth harmonic) in the following example. All the training samples are used simultaneously (batch training). At f_{cr}, the spatial wavelength of the desired output is equal to the receptive field width (10 in this case). Assume that all of the weights are initialized to zero, and the learning rate is 1. Using the rectangular field function, the weight update (δW_i) is shown in equation 3.1.

$$\begin{aligned} \delta W_i = {} & [Y_i - (W_i + \cdots + W_{i+9})] / 10 + \\ & [Y_{i+1} - (W_{i+1} + \cdots + W_{i+10})] / 10 + \cdots \\ & [Y_{i+9} - (W_{i+9} + \cdots + Wi+_{18})] / 10 \\ = {} & (Y_i + Y_{i+1} + \cdots Y_{i+9}) / 10 \end{aligned} \qquad \text{eq. 3.1}$$

After the batch training, each W_i becomes δW_i. Since the desired output is sinusoidal with wavelength equal to the receptive field width, the sum of all the desired outputs within any complete cycle is exactly zero. Thus, each δW_i is zero, and the network output will never converge. When each sample is trained sequentially, δW_i depends on the specific sequence of the training samples, and will not be exactly

zero. However, the slow convergence can still be expected.

III.1.B Impulse Learning Experiment

In this experiment, the desired network output was a discrete impulse function of magnitude 1. The impulse was located at 0.5. Figure 3.2 shows the relationship between the network output and the input after 10,000 and 50,000 random trainings.

After 10,000 random trainings, the network output approximated the impulse fairly well except at places where inputs were taken from odd multiples of 0.1 (these glitches were C input quantization intervals apart). After 50,000 trainings, the glitches disappeared. The network output and the desired impulse were almost identical.

These periodic glitches can be explained by examining the following equations. Assume that the impulse is located at X_i with which W_i through W_{i+c-1} are associated. Also, assume that each W_i has its appropriate value to reconstruct the desired impulse output. Equations 3.2 through 3.5 show the weights associated with inputs near the impulse. Equation 3.6 indicates a strong coupling of weights of C input quantization intervals apart.

$$W_{i-1} + W_i + \cdots + W_{i+8} = 0 \qquad \text{eq. 3.2}$$

$$W_i + W_{i+1} + \cdots + W_{i+9} = 1 \qquad \text{eq. 3.3}$$

$$W_{i+1} + W_{i+2} + \cdots + W_{i+10} = 0 \qquad \text{eq. 3.4}$$

$$W_{i+2} + W_{i+3} + \cdots + W_{i+11} = 0 \qquad \text{eq. 3.5}$$

Subtracting eq.3.2 from eq.3.3, $W_{i+9} - W_{i-1} = 1$,

Subtracting eq.3.4 from eq.3.3, $W_i - W_{i+10} = 1$,

Subtracting eq.3.5 from eq.3.4, $W_{i+1} = W_{i+11}$,

By deduction,

Eq. 3.6 $\quad W_{i+j} = W_{i+j+c}, \qquad \forall\ j \neq 0,-1$

III.2 Two Dimensional Input

III.2.A. Sinusoid Learning Experiment

In this experiment, the desired output was $\sin(2\pi M(X+Y))$, where M is the harmonic number. Ten sinusoidal functions of different spatial frequencies (the first harmonic to the tenth harmonic) were trained separately. The spatial wavelength of the first harmonic function in each axis was 50 input quantization intervals wide. In each training cycle, a training sample was randomly chosen from the training set. Figure 3.3 shows the relationship between the standard deviation of the training error (difference between the network output and the desired output at the training samples) and the function harmonic after 10,000 and 50,000 random trainings.

The RMS error remained unchanged after 40,000 additional random trainings, implying that the approximation can not be improved further regardless of the training time. The network approximated better at lower spatial frequencies. At frequencies above f_{cr}, the network produced non-diminishing RMS values of about 0.8.

As mentioned in chapter 2, the number of available receptive fields is completely determined by S, C and N (equation 2.1). For a one-dimensional input, there are always more receptive fields (S+C-1) available than the number of

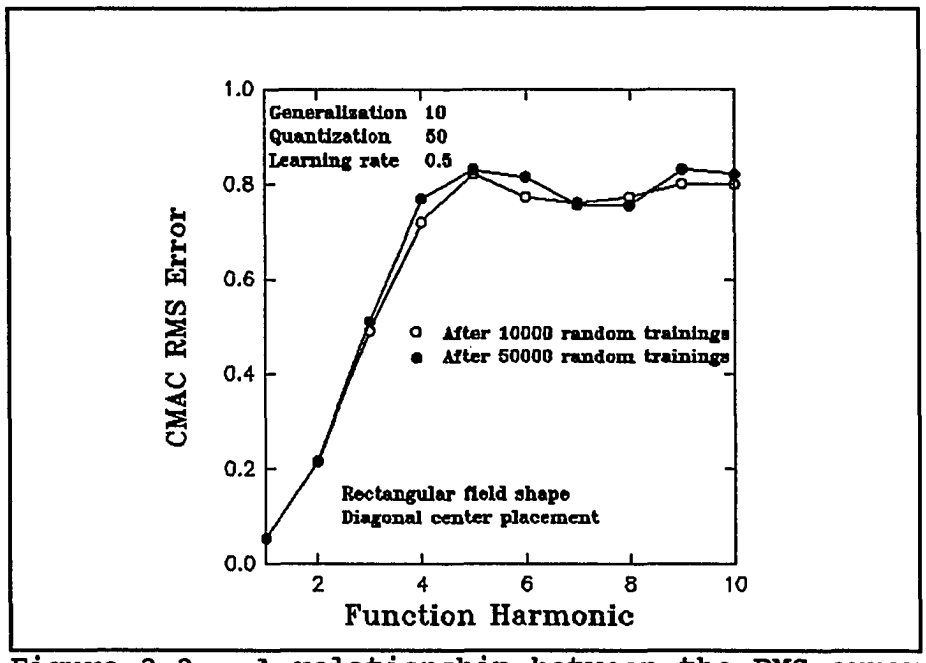

Figure 3.3 A relationship between the RMS error and the function harmonic for a 2D input.

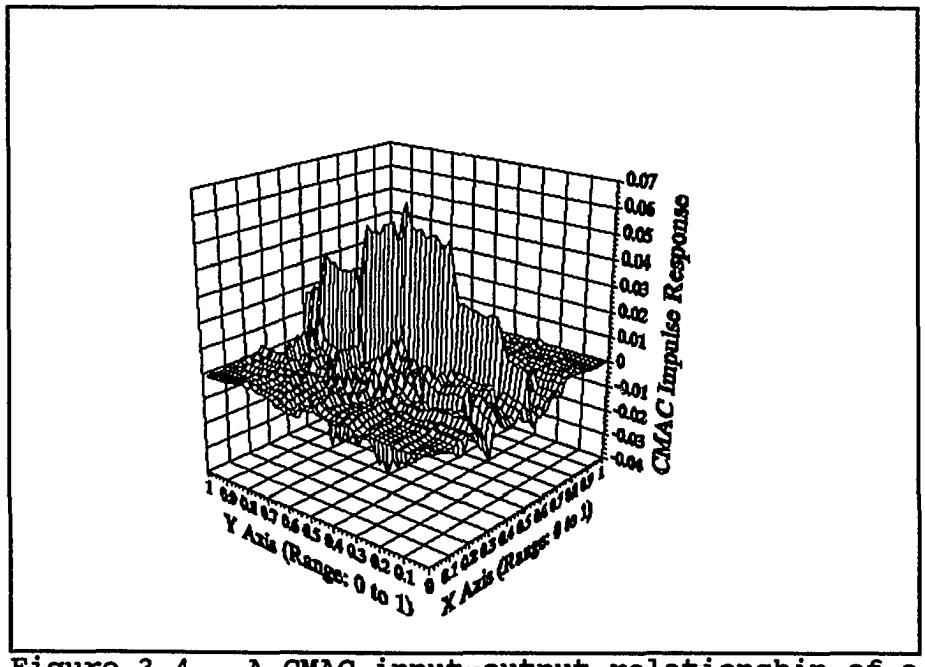

Figure 3.4 A CMAC input-output relationship of a two-dimensional desired impulse output after 10,000 random trainings with C set to 10.

possible independent training samples (S). This results in an under-determined condition in which there are infinite number of weight combinations which can reconstruct the desired output exactly at the training samples. However, for a two-dimensional input, there are fewer receptive fields available than the number of possible independent training samples (S^2). This results in an over-determined condition in which there is no set of weights which can reconstruct an arbitrary desired output exactly at the training samples.

III.2.B Impulse Learning Experiment

The desired network output was a discrete impulse function of magnitude 1. The impulse was located at 0.5 in each axis. Figure 3.4 shows the relationship between the network output and the input after 10,000 random trainings. Figure 3.5 shows the same relationship after 50,000 random trainings. Figure 3.6 shows the relationship between the network output and the input after 10,000 random trainings in which C was changed from 10 to 5.

In figure 3.4, the convergence was not complete because each sample was only trained an average of four times. In both figure 3.4 and 3.5, it is clearly seen that the network output did not converge to the desired impulse after 50,000 trainings. This is caused by an inadequate number of available receptive fields for learning (the peculiar network responses favored in the X axis has not been understood). Figure 3.5 and

3.6 show that the network output better resembled the desired impulse after changing C from 10 to 5 (which increased the number of receptive fields).

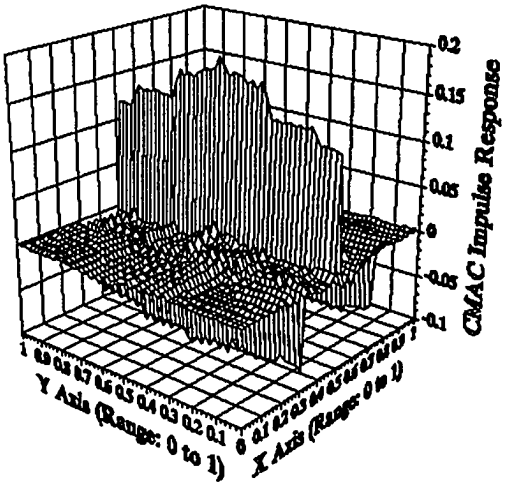

Figure 3.5 A CMAC input-output relationship of a two-dimensional desired impulse output after 50,000 random trainings with C set to 10.

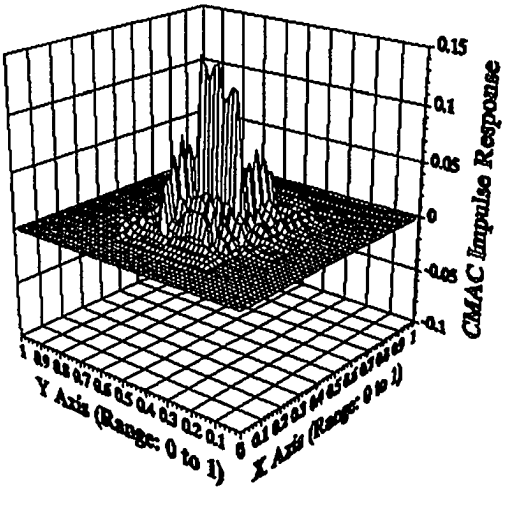

Figure 3.6 A CMAC input-output relationship of a two-dimensional desired impulse output after 10,000 random trainings with C set to 5.

III.3 Discussion

Four considerations should be taken into account when using CMAC to approximate a function. First, the convergence is slow at the critical spatial frequency (f_{cr}) when the sum of all the desired outputs within the receptive field is exactly zero and the rectangular receptive field function is used. Since f_{cr} was found to be an important spatial frequency limit for both a one-dimensional input and a two-dimensional input, it is very likely that it also is important for multi-dimensional inputs.

Second, the value of C has a different impact on the allocation of receptive fields in a one-dimensional CMAC and a multi-dimensional CMAC. Because the coarse receptive fields are arranged in a hyperlayer structure, the receptive field density (or the number of receptive fields inside the reference hypercube) is inversely proportional to C^{N-1}. For a one-dimensional input, a larger C allocates more receptive fields (C-1 more than S). However, for a multi-dimensional CMAC, a larger C allocates fewer receptive fields (always fewer than S^N) because of the reduction factor (C^{N-1}).

Third, the receptive field centers are placed on the hyper-diagonals in the input space. Increasing C not only reduces the number of receptive fields in S^N, but also forces the center placement to be less uniform in C^N. It is

speculated that this non-uniform center placement will become more pronounced for a larger N. In addition, the number of receptive fields required grows exponentially with N. This makes the experiment impossible without the use of hashing, which further complicates the analysis. This justifies the use of a two-dimensional CMAC in this chapter for learning.

Last, while learning a space-spanning desired output in S^N requires an excessive number of receptive fields, learning a trajectory in S^N, however, requires fewer receptive fields. For example, a trajectory on the hyperdiagonal in S^N is equivalent to a function in S^1, assuming that the hashing does not generate any collisions.

To summarize, the hyper-diagonal field placement becomes less uniform in C^N for a larger C. On the other hand, too small a C does not provide sufficient generalization in S^N. Also, the learning will be very sluggish at f_{cr} if the rectangular field function is used. Based on these undesirable features in the standard CMAC, Chapter IV presents an alternative receptive field function and placement, while still requiring the field placement to be modular and the field width to be uniform, which are desirable features in the standard CMAC.

CHAPTER IV

RECEPTIVE FIELD FUNCTION AND PLACEMENT

The use of a rectangular field function in a standard CMAC is undesirable for two reasons. First, the network output is insensitive to any input within a input quantization interval. Second, learning is sluggish at the critical spatial frequency (f_{cr}) [5]. A better receptive field should be tapered from the field center to its boundary so that the network output is continuous rather than piecewise constant. On the other hand, using a tapered receptive field function also means the network output becomes sensitive to the center placement within C^N. That is, a non-uniform center placement is likely to deteriorate the learning performance. This chapter studies both the effect of linearly-tapered field function and center placement on learning.

The region of support for the linearly-tapered receptive field used in this chapter is either hypercubic or hyperspherical. In order to define the receptive field strength over these regions of support, let the field strength of the ith receptive field projected onto the kth dimension be f_{ik}. Also, imagine that a hypersphere of radius R is bounded by the hypercube with all faces touching the hypersphere. If the region of support is hypercubic, the field strength of the ith receptive field for a given input is defined as the

smallest f_{ik}. For a two-dimensional input, the receptive field function is a pyramid. If the region of support is hyperspherical, the field strength of the ith receptive field is a function of the Euclidean distance between the input and the center of the ith receptive field. The field strength is zero for any input outside the hypersphere. For a two-dimensional input, the receptive field function is a cone. Although the linearly-tapered field with the hypercubic support introduces discontinuity in slope on the hyper-diagonal, the magnitude of the field strength is independent of the input dimension (N).

A general weight update equation for any arbitrary field function is given in equation 4.1. The network output Y for input $\underline{X}$ is $\Sigma f_i(\underline{X})W_i$, where $f_i(\underline{X})$ is the receptive field strength associated with $\underline{X}$. The weight adjustment δW_i is derived by means of the gradient descent method.

$$\delta W_i = \beta \cdot (Y_d(\underline{X}) - \sum_i f_i(\underline{X})W_i) \cdot f_i(\underline{X}) \;/\; \sum_i f^2_i(\underline{X}) \qquad \text{eq. 4.1}$$

where W_i is one of C weights associated with $\underline{X}$. $\Sigma\, f^2_i(\underline{X})$, a normalizing factor, is a sum of all C squared receptive field strengths associated with $\underline{X}$.

In the case of the rectangular receptive field, $\Sigma f^2_j(\underline{X})$ is equal to C because each $f_i(\underline{X})$ is 1. Thus eq.4.1 reduces to eq.2.2. Equation 4.1 is used in this chapter to account for

the linearly-tapered receptive field function by setting each f_i appropriately.

Section IV.1 of this chapter presents sinusoid learning experiments on CMAC for one-dimensional and two-dimensional inputs. A linearly-tapered field function with a diagonal center placement was used. Section IV.2 presents a sinusoid learning experiment for only a two-dimensional input. A linearly-tapered field function with a different center placement (P1) was used. Section IV.3 presents a sinusoid learning experiment for only a one-dimensional input. A CMAC with non-uniform field widths and offsets (P2) was studied for both a linearly-tapered and a rectangular field function. Section IV.4 presents three heuristic rules which provide a more uniform center placement. In particular for a two-dimensional input, a perfectly uniform center placement (defined shortly) can be achieved if the value of C satisfies a given constraint. Learning with a linearly-tapered receptive field with hyperspherical support was also studied. Section IV.5 presents experimental evaluation of the uniformity of placement for the diagonal placement and the heuristic placement.

Each analysis in Section IV.1-3 was performed under the following conditions, unless otherwise stated. There were 50 input quantization intervals (S) in each axis. The input generalization was 10 input quantization intervals wide, and the learning rate was 0.5. All of the training pairs were

obtained by sampling at the input quantization intervals (and their intersections) in S^N. All the training samples formed a training set. Thus, there were S^N possible independent training samples in the training set. Each sample input could take any value from 0 to 1. Also, the region of support for the linearly-tapered field function was hypercubic.

IV.1.A 1D Sinusoid Learning Experiment

In this experiment, the desired output was $\sin(2\pi MX)$, where M is the harmonic number. Ten sinusoidal functions of different spatial frequencies (the first harmonic to the tenth harmonic) were trained separately. The spatial wavelength of the first harmonic was 50 input quantization intervals wide. A linearly-tapered field function (with a hypercubic support) and a diagonal center placement was used.

In each training cycle, a sample was randomly chosen from the training set. The training ended when the standard deviation of the training error at the training samples was less than 0.1. Figure 4.1 shows the relationship between the number of training cycles required and the function harmonic.

The slow convergence at the critical spatial frequency (f_{cr}) was eliminated, indicating relatively wider learning bandwidth. However, the time to converge increased exponentially beyond f_{cr}. The poor learning convergence beyond f_{cr} suggests that any generalization larger than the spatial period of the sinusoid is undesirable, regardless of the receptive field function.

IV.1.B 2D Sinusoid Learning Experiment

In this experiment, the desired output was $\sin(2\pi M(X+Y))$, where M is the harmonic number. Ten sinusoidal functions of

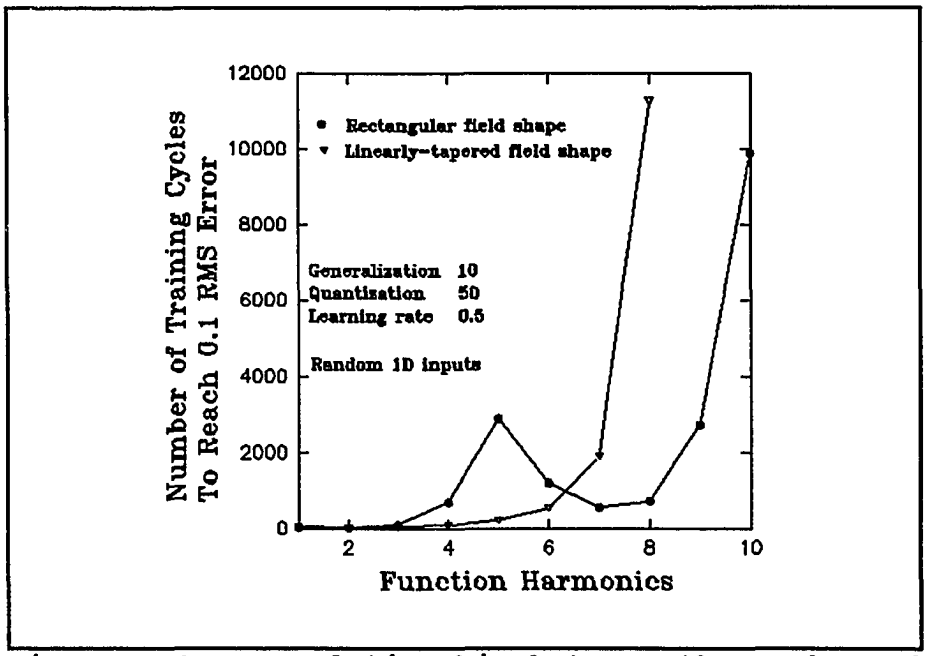

Figure 4.1 A relationship between the number of training cycles and the function harmonic.

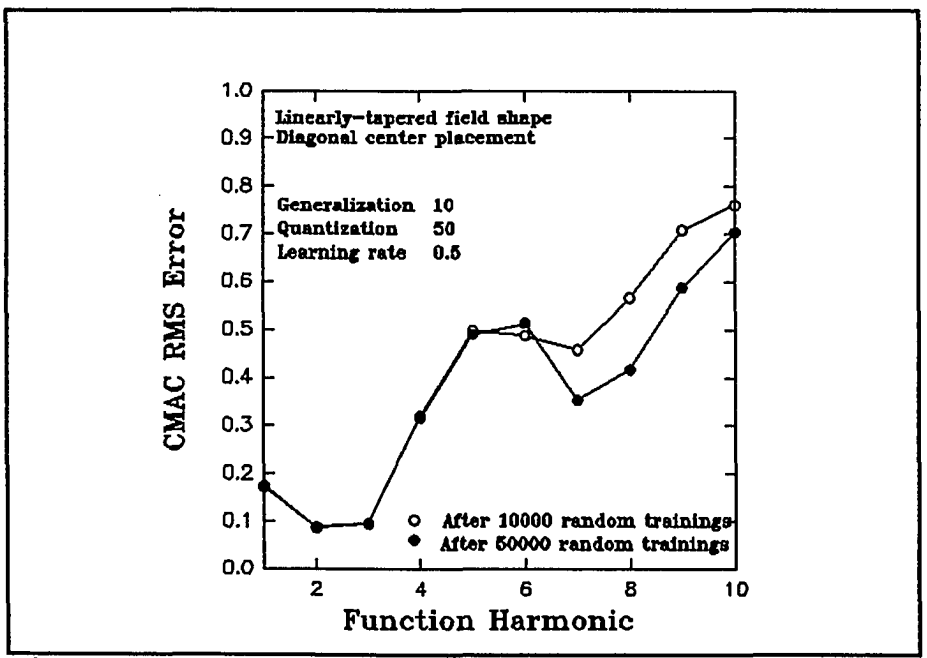

Figure 4.2 A relationship between the RMS error and the function harmonic.

different spatial frequencies (the first harmonic to the tenth harmonic) were trained separately. The spatial wavelength of the first harmonic in each axis was 50 input quantization intervals wide. A linearly-tapered field function with a diagonal center placement was used. Figure 4.2 shows the relationship between the standard deviation of the training error and the function harmonic after 10,000 and 50,000 random trainings.

While the error remained unchanged below f_{cr} after 40,000 extra trainings, a noticeable error reduction was observed beyond f_{cr}. Also, the error was unusually large at low spatial harmonics, compared to the rectangular field function (figure 3.3). The large error at low spatial harmonics can be explained by considering the following example using a one-dimensional CMAC with C set to 1 (only one layer of receptive fields).

In figure 4.3, suppose that X_A is near the field edge of f^* (maximum field strength W^* at the center of f^*), and X_B is near the center of f^*. Also, suppose a desired output for both X_A and X_B is Y_d, the weight adjustment δW^* due to X_A will be large because both $f^*(X_A)$ and $f^*(X_A) \cdot W^*$ are small (refer to eq. 4.1). On the other hand, δW^* due to input X_B will be relatively small because both $f^*(X_B) \cdot W_*$ and $f^*(X_B)$ are large (near 1). As both X_A and X_B fall within the same receptive field (f^*), W^* will thus oscillate which may result in a larger final approximation error. Based on this argument, approximating a

constant desired output (or any output with low spatial frequencies) with a non-uniform center placement and a linearly-tapered field function is undesirable.

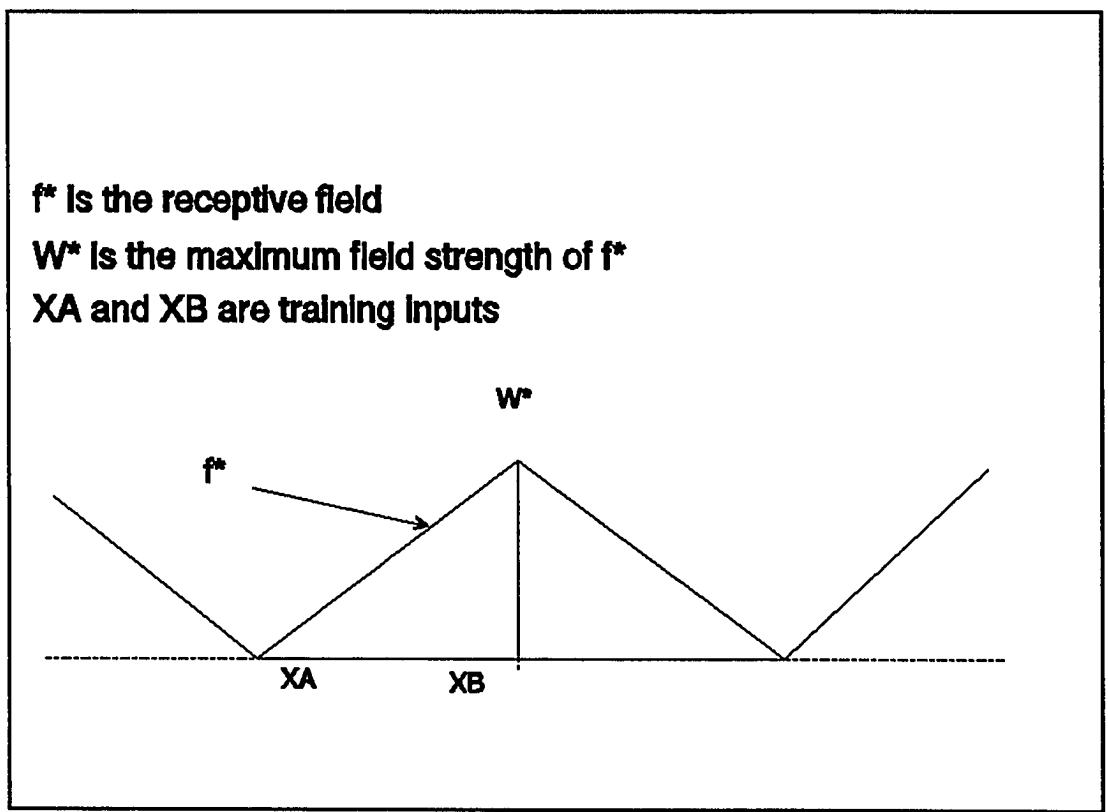

Figure 4.3 A single layer of linearly-tapered receptive fields for a scalar input.

IV.2 2D Sinusoid Learning Experiment with P1 Placement

The results in Section IV.1 show that, for a two-dimensional input, learning was worse at low spatial harmonics because of the interaction of the non-uniform center placement and the linearly-tapered receptive field function. Learning might be improved if a uniform center placement is available. Unfortunately, searching for a uniform center placement in S^N is inherently difficult. For this reason, the following experiment was carried out only for a two-dimensional input where a uniform center placement (P1) was easily arranged (figure 4.4). P1 is a perfectly uniform placement because the distribution of centers not only is uniform in a two-dimensional space (same Euclidean distance between any two nearest neighbors), but also the projected distribution of centers onto each axis is uniform. The uniform projection of centers onto each axis is especially desirable because the number of projected receptive fields is the same for each axis, allowing a uniform input resolution in each axis.

The experimental conditions were identical to those in Section IV.1.B, except that the P1 placement was used. Figure 4.5 shows the relationship between the RMS training error and the function harmonic after 10,000 and 50,000 random trainings. Comparing to figure 4.2, the undesirably large RMS errors found previously at low spatial harmonics were reduced

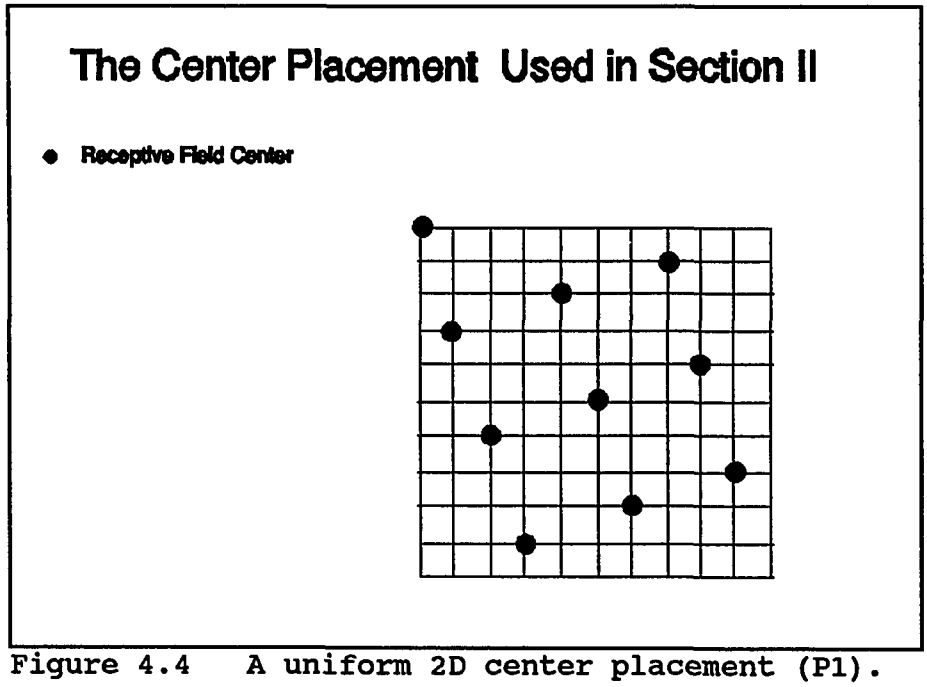

Figure 4.4 A uniform 2D center placement (P1).

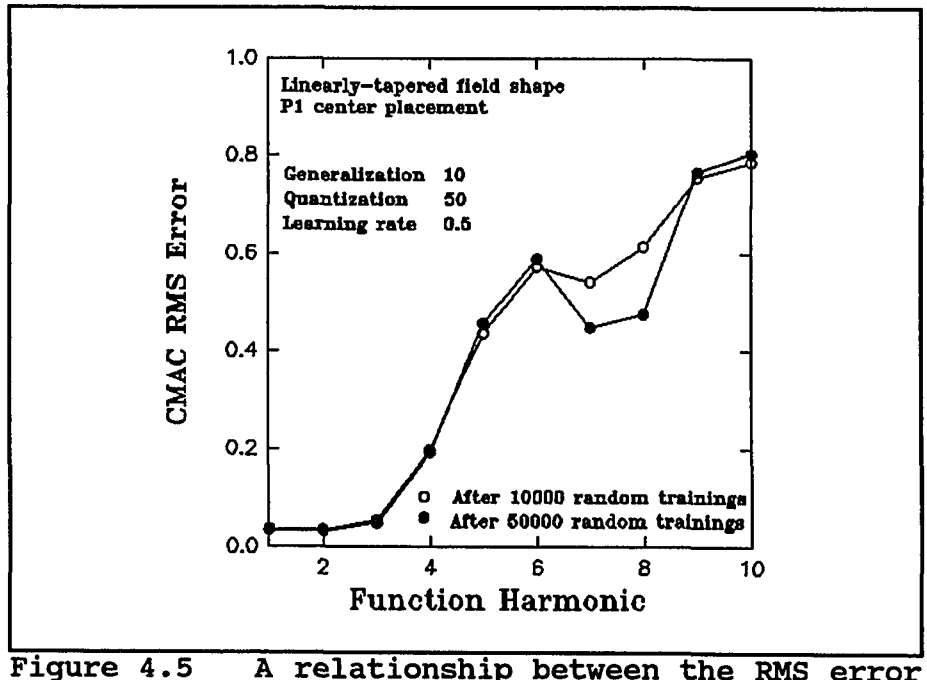

Figure 4.5 A relationship between the RMS error and the function harmonic for a 2D input.

significantly. However, no significant changes in RMS error were observed at high spatial harmonics between the diagonal placement and the P1 placement. While the uniformity of the center placement leads to a smaller final approximation error at low spatial harmonics, the effect of the uniform center placement becomes less significant at higher spatial harmonics given the receptive field width is fixed.

IV.3 1D CMAC With Non-Uniform Field Widths And Offsets

In both Section IV.1 and IV.2, the center placement assumed uniform field widths in each layer of the CMAC. In general, the placement can also be constructed by using different field widths for different layers. This section presents a one-dimensional sinusoid learning experiment based on a center placement (P2) with non-uniform field widths and non-uniform offsets. Both linearly-tapered field and rectangular field were used.

In P2, seven layers were used. The field widths in the first three layers were each set to 10, and their respective offsets were 0, 4 and 7. The field widths in the fourth and fifth layers were each set to 7, and their respective offsets were 0 and 4. The field widths in the last two layers were each set to 4, and their respective offsets were 0 and 2. The overall center placement is shown in figure 4.6. The sizes of these field widths were chosen in such a way that both the P2 placement and the diagonal placement had the same total number of receptive fields. Unlike the standard CMAC, the number of layers was not equal to the width of the receptive field (C). Also, it should be noticed that although the distribution of centers is uniform in each individual layer, the overall distribution in P2 is no longer uniform.

Figures 4.7 and 4.8 show the relationship between the RMS

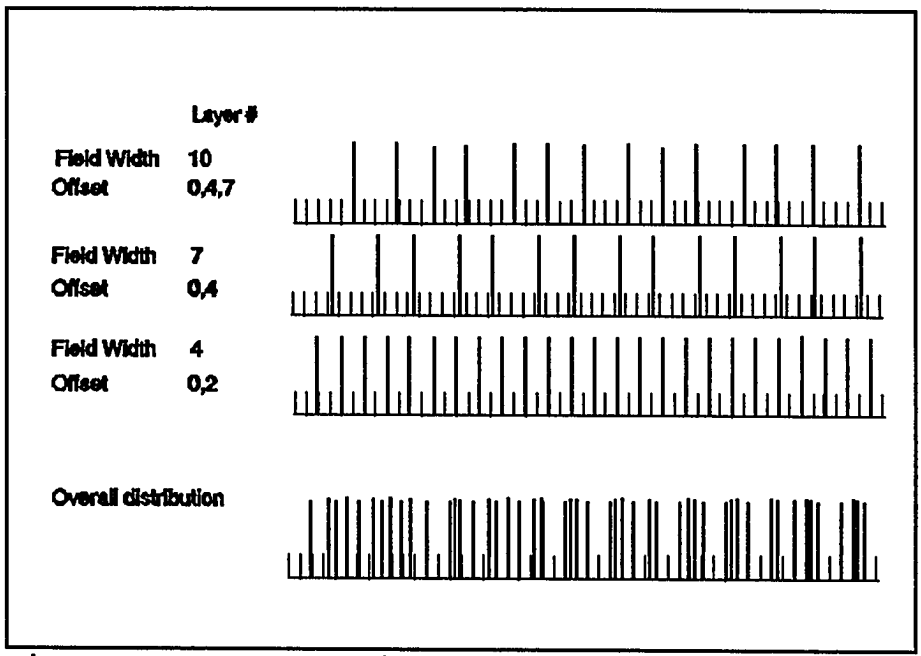

Figure 4.6 A non-uniform center distribution (P2) for a scalar input.

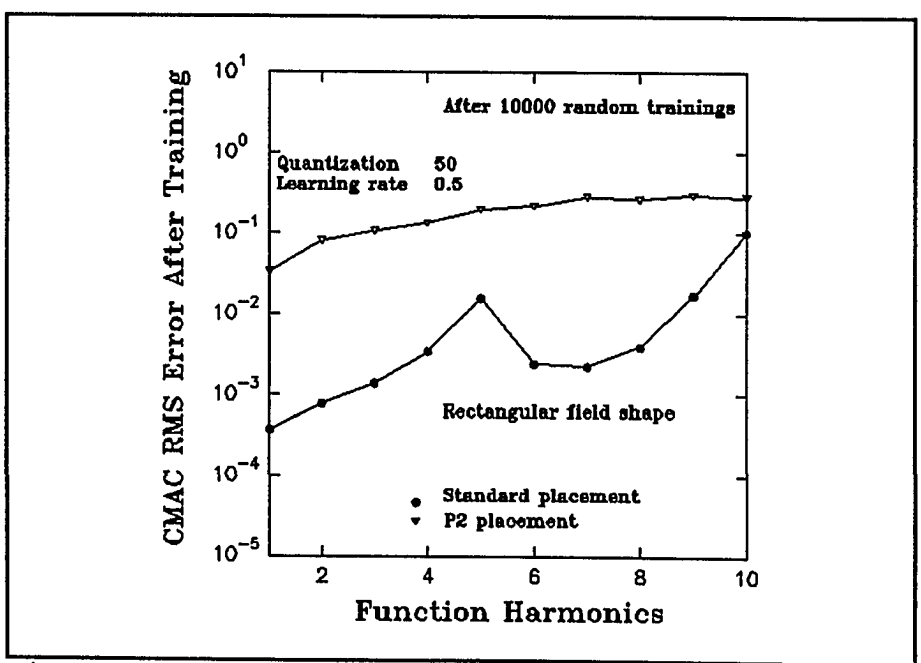

Figure 4.7 A relationship between the RMS error and the function harmonic for a 1D input.

error and the function harmonic using the rectangular field function and linearly-tapered field function respectively after 10,000 random trainings. In the case of the rectangular field function, the CMAC (parameter values, such as C, S and β, were given earlier in this chapter) with uniform field widths and offsets achieved substantially smaller RMS error than the P2 placement. The non-uniform field widths and offsets caused the field coverage to be non-uniform, which adversely affected the approximation accuracy. However, in contrast to the CMAC with uniform receptive field widths, the error magnitude for the P2 placement did not peak at f_{cr} because of the averaging effect of different field widths and offsets.

In the case of the linearly-tapered field function, similar results were found at low spatial harmonics. However, the P2 placement performed better at high spatial harmonics. At these harmonics, the desired output started to resemble a linearly-tapered field with a small field width (σ). The desired output can easily be approximated by using only a single layer of linearly-tapered receptive fields with the field width set to σ. This mere coincidence of matching the receptive field to the desired output should not be used to conclude that the non-uniform center placement is useful at high spatial harmonics.

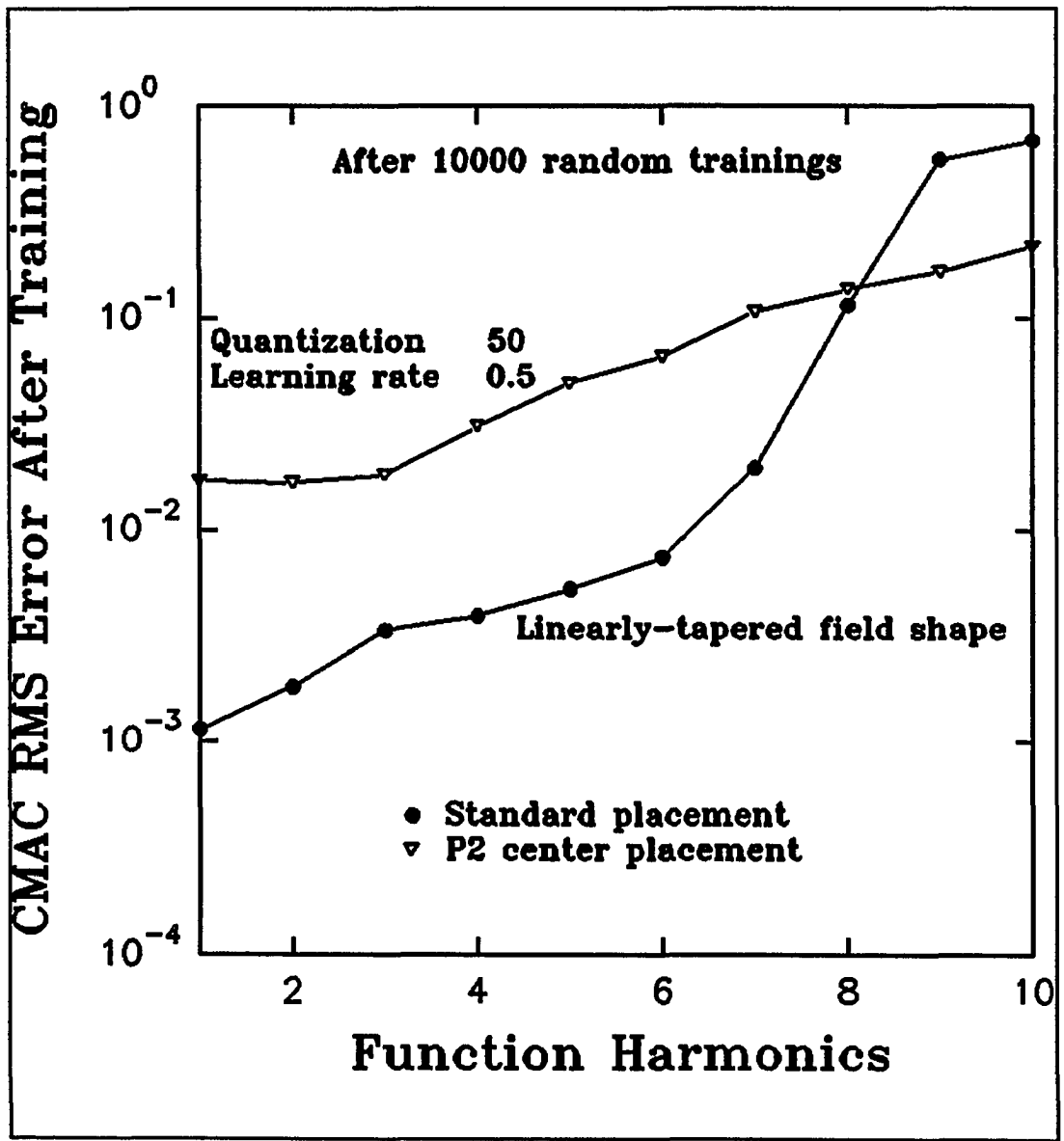

Figure 4.8 A relationhip between the RMS error and the function harmonic for a 1D input.

IV.4 Receptive Field Center Placement

Although the diagonal receptive field placement of the standard CMAC is undesirable for a large C in S^N because of its non-uniform center coverage, the arrangement has three useful features which are important to preserve. First, the receptive field width should be uniform or the coverage will not be uniform. Second, the receptive field placement should be modular in S^N. This eliminates unnecessary searching for neighboring fields in order to generate a network output. Last, the projection of receptive field centers onto each individual axis should be uniform in order to obtain a uniform input resolution.

Three heuristic rules were developed to provide a more uniform center placement in C^N while preserving these desirable features. A modular arithmetic expression is used to define each center's co-ordinates. The co-ordinates of the mth center inside C^N is defined as

$$[m \cdot d_1 \,\%\, C,\ m \cdot d_2 \,\%\, C, \cdots m \cdot d_N \,\%\, C], \qquad m \in (0, \cdots, C-1),$$
$$d_i,\ m \in I$$

where % is a modulo operator and d_i is the displacement offset in the ith dimension. Due to the modular feature constraint, d_i must be an integer.

By using this co-ordinate definition, the first center is always located at the origin $[0,0\cdots,0]$. A displacement vector $\underline{D}$ is defined as $<d_1,d_2,\cdots d_N>$, where d_1 is always set to 1. Thus, the center placement is completely described by the vector $\underline{D}$. $\underline{D}$ becomes a unity vector for the standard CMAC. The three heuristic rules (HR1, HR2 and HR3) which provides a more uniform displacement offset for the center placement in CMAC are given as follows.

HR1: $\alpha \cdot d_i \neq \gamma \cdot C$,

where $2 \leq i \leq N$, $1 \leq d_i < C/2$, $d_i,\alpha,\gamma \in I$

HR2: $d_i \neq d_j$, $2 \leq i,j \leq N$, $i \neq j$

HR3: Max $SD(\underline{D})$, $\underline{D} \in \{HR1\} \cap \{HR2\}$

where $SD(\underline{D})$ stands for a standard deviation of all components in $\underline{D}$.

HR1 prevents d_i's from sharing common factor(s) with C. As a result, d_i's satisfying HR1 ensure that the projection of centers on each axis will be uniform. This means that the number of independent projected receptive fields along each axis is always C. Each point in the space falls in exactly C receptive fields, and a unit step in position parallel to any axis crosses one and only one receptive field boundary. d_i should be less than C/2 due to the mirror image symmetry.

By avoiding any two d_i's which are the same, HR2 ensures

that the projected center placement onto any subspaces can not be diagonal. All the possible candidates to be used in HR2 must also satisfy HR1.

HR3 ensures that the standard deviation of all d_i's in $\underline{D}$ must be maximized. Otherwise, centers will be clustered in some local region in C^N. Again, these d_i's must also satisfy HR1 and HR2.

To illustrate how to apply these rules, let C be 10 and N be 2, the only choice for d_i after applying HR1 is 3. This choice (3) also satisfies HR2 and HR3 (because d_1 is 1 and d_2 is 3). If a vector $\underline{D}$ <1,3> is used, the centers' co-ordinates are [0,0], [1,3], [2,6], and so forth. The resulting center arrangement is shown in figure 4.4. Note that if C is prime, then each d_i has less than C/2 choices, which indicates that C should be greater than $2 \cdot N$. <u>This is a lower bound on C for any N if a uniform placement is to be achieved</u>. Also, if C is 2^α ($\alpha \varepsilon I$), then each d_i has less than C/4 choices (because only the odd numbers are valid), which indicates that such C should be greater than $4 \cdot N$. In general, a larger N requires a larger C in order to maintain uniform projected field coverage in each individual axis.

It is important to notice that these heuristic rules do not guarantee a uniform projection of centers onto higher dimensional subspaces (C^M, $M < N$). However, these heuristic rules allow the center placement to be more uniform than the diagonal placement. For a two-dimensional input, a perfectly

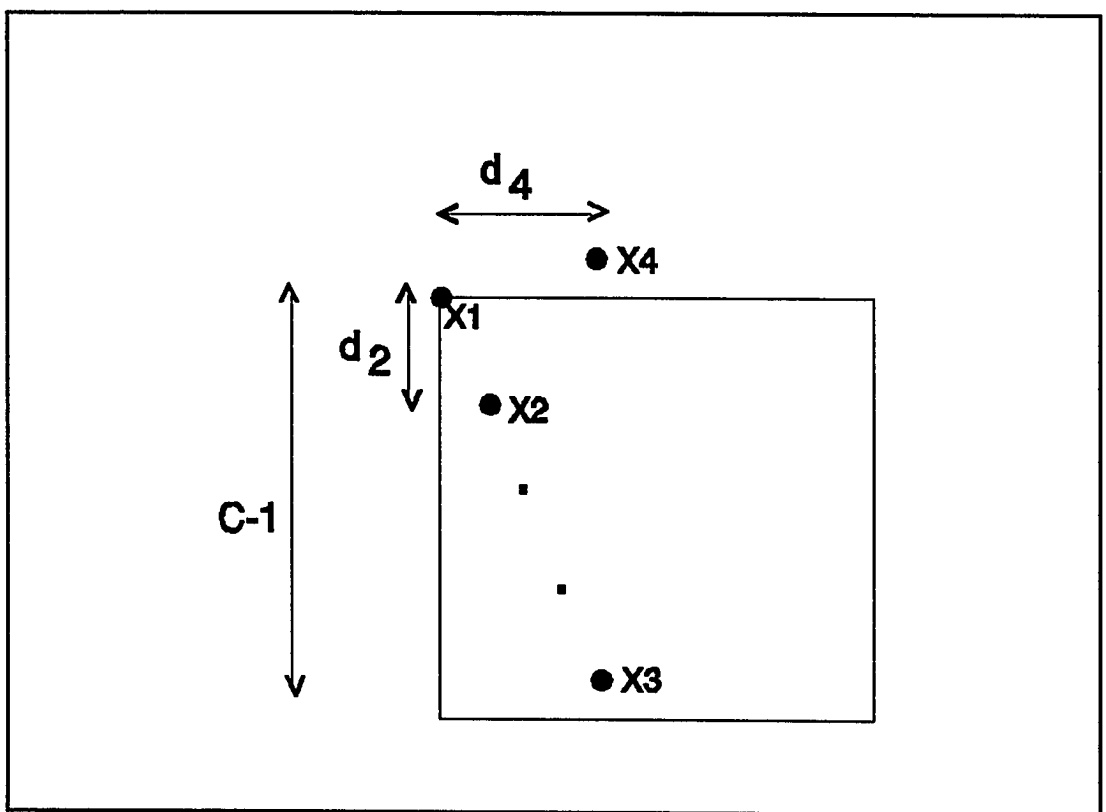

Figure 4.9 A configuration required for the proof of a uniform center placement for a 2D input.

uniform center placement can be achieved if d_2 is set to $\sqrt{(C-1)}$. In this case, $\underline{D}$ is $<1,\sqrt{(C-1)}>$. This condition can be derived by referring to figure 4.9. Let X_2 ($[1,d_2]$) be the immediate neighbor of X_1. By using symmetry of axis, there must be another center located at X_4 ($[d_4,-1]$). Also, there must be another center located at X_3 ($[d_4,C-1]$) because of the modular feature. In order to require the nearest neighbor of X_1 in each axis to be the same distance away, d_2 should be set to d_4 and $d_2 \cdot d_4 = C-1$, $d_2^2 = C-1$. For example, C can be either 10, 17 or 26.

Although there is no analytical proof of any better configuration for $\underline{D}$, numerical searches for other possible arrangements in C^N have not yet provided any significantly better solutions. Also, an optimal range of C has emerged from these heuristic rules. It has been mentioned that too large a C causes the field coverage to be sparse (refer to chapter two). On the other hand, too small a C causes the field coverage to be less uniform. To summarize, the lower bound on C for any N is $2 \cdot N$.

IV.5 Experimental Evaluation

Using a modulo arithmetic method to arrange receptive field centers, Parks and Militzer [20] performed an exhaustive search to select good displacement vectors $\underline{D}$ for various combinations of C and N. Their exhaustively searched placement is referred to in this dissertation as the Parks placement. Their search criterion was based on maximizing the minimum distance between two nearest neighbors (no learning was involved). Another way to evaluate the uniformity of the center placement is to set all weights equal to 1 (no learning), and to evaluate the network output at many random points in C^N. In this case, the network output is simply the sum of C field strengths. If the sum is not a constant over the region, the network is more responsive to some inputs than others. Since the network is designed without consideration of the exact function to be learned, such spatial variation in sensitivity is undesirable. It is assumed that the most uniform arrangement of receptive fields will have the least spatial variation in sensitivity.

Experiments were done using CMACs with C in the range from 2 to 50 in both S^3 and S^{10}. A linearly tapered receptive field with either hyperspherical or hypercubic support was used in all three placements (the diagonal placement, the heuristic placement and the Parks placement). In the heuristic

placement, only HR1 and HR3 were used to determine $\underline{D}$ (otherwise there might not exist any valid vector $\underline{D}$ for some values of C if HR2 is imposed). The network output was evaluated at 1000 random inputs inside C^N. A uniformity index was defined as a ratio of the standard deviation to the mean of these responses. As any input excites exactly C receptive fields, the index using a rectangular field will always be zero. Thus, the use of the rectangular field with the diagonal placement does not lend any useful insights in determining the uniformity of field coverage. In other words, the index is only meaningful for tapered receptive field functions.

Figure 4.10 shows the relationship between the uniformity indexes of the hypercubic support and the generalization (C) in S^3, while figure 4.11 shows the relationship between the indexes of the hyperspherical support and the generalization (C) in S^3. In both figure 4.10 and 4.11, all three center placements (diagonal, heuristic, Parks) were used. First, the indexes for the diagonal placement with the linearly-tapered receptive field were independent of C (except for a small C) for both regions of support. While increasing C makes the coverage less uniform, increasing C also increases the mean field strength. Second, the indexes for the heuristic placement were much smaller than the indexes for the diagonal placement for both regions of support, indicating that the heuristic placement was more uniform. However, the indexes for the heuristic placement were very sensitive to C (especially

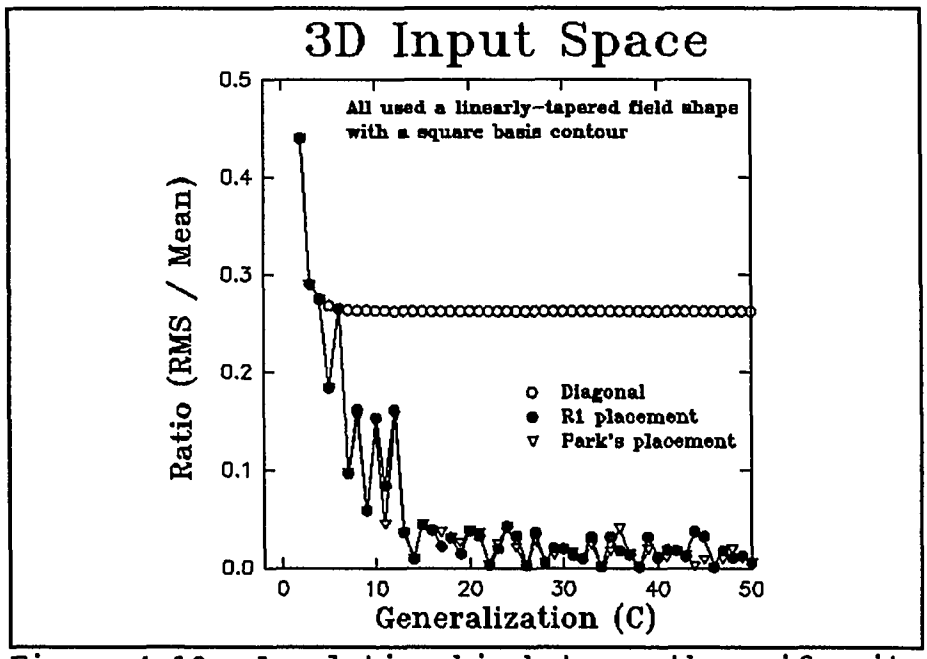

Figure 4.10 A relationship between the uniformity indexes and the generalization with the hypercubic contour for a 3D input.

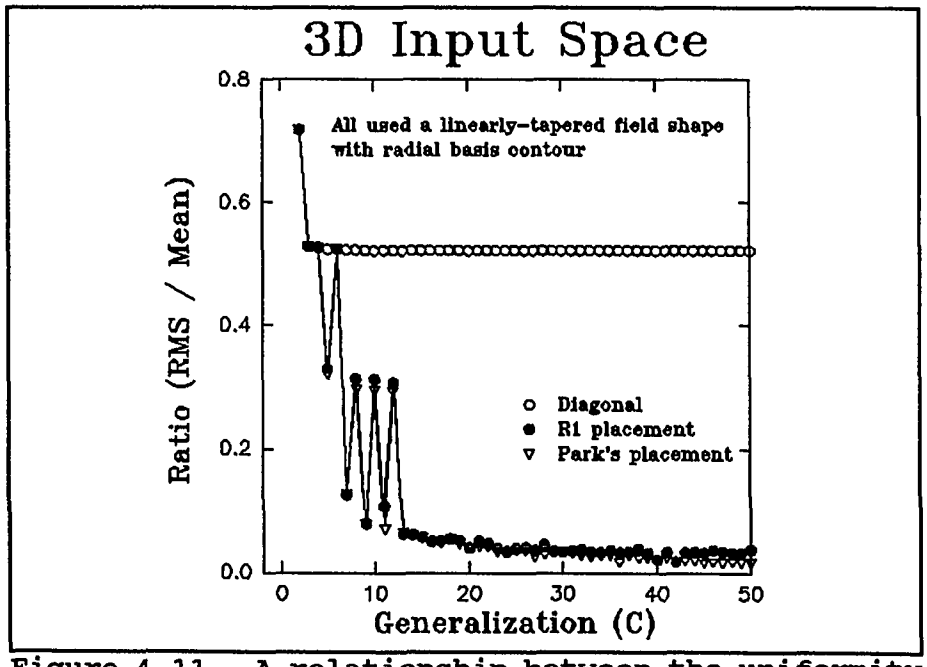

Figure 4.11 A relationship between the uniformity indexes and the generalization with the hyperspherical contour for a 3D input.

a small C). This suggests that field coverage for some values of C were better than others even though the same heuristic rules were applied. Last, the indexes for the Parks placement and the heuristic placement were very similar for both regions of support, suggesting that the heuristic placement is as good as the exhaustive search placement.

Figure 4.12 shows the relationship between the uniformity indexes of the hypercubic support and the generalization (C) in S^{10}, while figure 4.13 shows the relationship between the indexes of the hyperspherical support and the generalization (C) in S^{10}. As in S^3, the indexes for the diagonal placement with the linearly-tapered receptive field were independent of C (except for a small C) for the same reason. Also, the indexes for the Parks placement and the heuristic placement were very similar for both regions of support. These results are consistent with those obtained in S^3, suggesting that the heuristic rules are a good set of rules. However, the indexes with the hyperspherical support were much larger than those with the hypercubic support. Since the Euclidean distance from one center to its far corner inside C^N is $(\sqrt{N})\cdot(C/2)$, centers at far corners with respect to the input contributed insignificant field strengths to the network output in S^N. Thus, the overall mean strength was much smaller. The indexes for the heuristic placement were less sensitive to C than those in S^3.

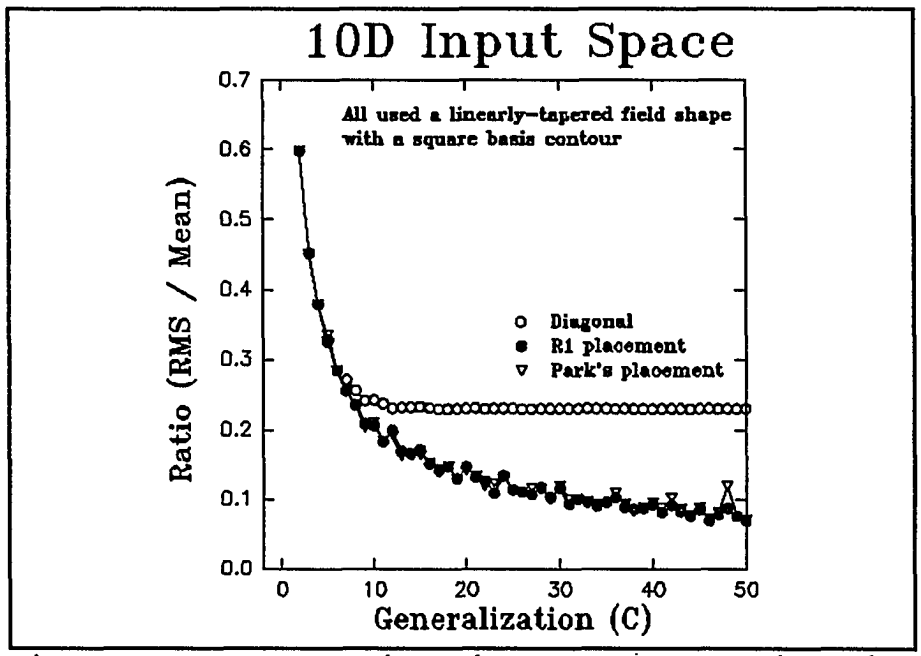

Figure 4.12 A relationship between the uniformity indexes and the generalization with the hypercubic contour for a 10D input.

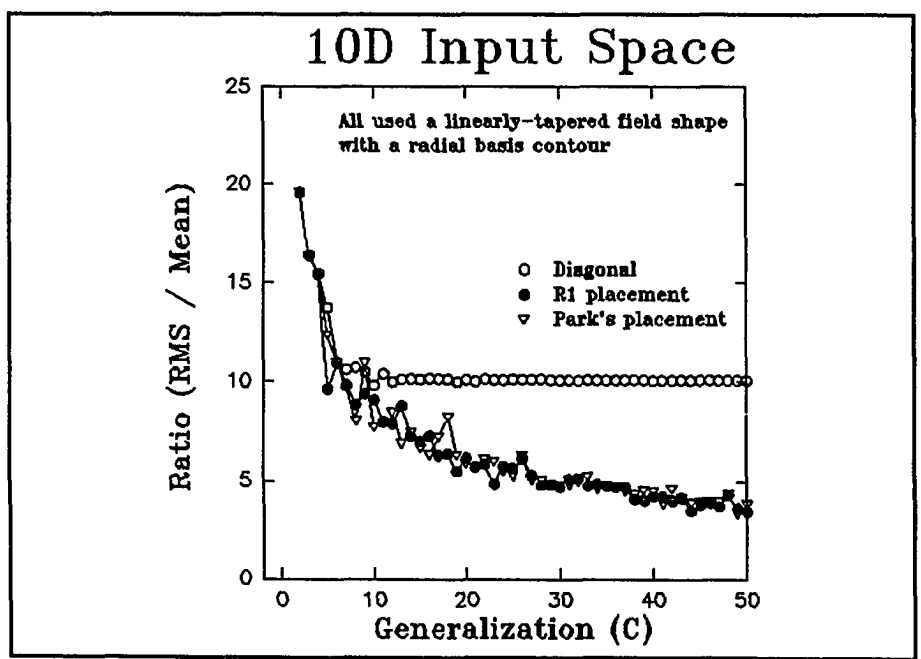

Figure 4.13 A relationship between the uniformity indexes and the generalization with the hyperspherical contour for a 10D input.

IV.6 Discussion

For the diagonal placement, C must be larger than or equal to N so that the hyper-diagonal inside C^N is completely filled. Otherwise, the coverage in C^N is sparse, and results in a larger uniformity index.

The heuristic placements used in Section IV.5 were generated by utilizing both HR1 and HR3. In figure 4.10 through 4.13, those center placements which violated HR2 had significantly larger indexes. This suggests that HR2 is an important constraint.

It is interesting to notice that the lower bound on C defined in Section IV.4 predicted the knee of each curve well (refer to figure 4.10-4.13). This suggests that this lower bound can be useful in assigning C for any N. In other words, while a smaller C ($< 2 \cdot N$) causes the center coverage to be less uniform inside C^N, a larger C results in fewer available receptive fields in S^N (the sparse center coverage).

The differences between the indexes for the diagonal placement and the heuristic placement were smaller in S^{10} than in S^3. However, in S^{10}, increasing C leads to a smaller index for the heuristic placement, but increasing C does not change the index for the diagonal placement ($C \geq 10$). Thus, the difference in indexes between the heuristic placement and the diagonal placement increases for a larger C.

The heuristic placement produces a better fit without requiring any additional numerical computation. The hyperspherical support was found to be undesirable in CMAC. A linearly-tapered receptive field with a hypercubic support was found to be better suited to the hypercubic structure of CMAC.

CHAPTER V

ADAPTIVE RECEPTIVE FIELD DENSITY CMAC

In the sinusoid learning experiments described in the previous chapters, the desired output was a sinusoid of a single spatial frequency in S^2. In practice, the desired output contains regions of high spatial frequencies and of low spatial frequencies. Thus, a different desired output (Y_d) in S^2 was studied in this chapter (figure 5.1). The desired output (Y_d) was $\sin(2\pi d)+\sin(10\pi d)$ inside a radius of 0.25 centered at <0.5,0.5>, where d^2 was $(x-0.5)^2+(y-0.5)^2$. Outside of this radius, the desired output was zero. A multi-dimensional CMAC and a two-dimensional CMAC have the shared properties of sparse and possibly non-uniform receptive field coverage and non-zero final approximation error. However, learning in S^N requires a huge number of virtual receptive fields which must be hashed into the physical weight storage. Based on these considerations, only a two-dimensional input was used in the following experiments.

Section V.1 provides a comparison between a single CMAC and multiple CMACs (several complete CMACs were used) after training to approximate Y_d. Section V.2 presents results of approximating Y_d with an adaptive receptive field density CMAC in which the receptive field density (or the number of receptive fields in the reference hypercube) is allowed to

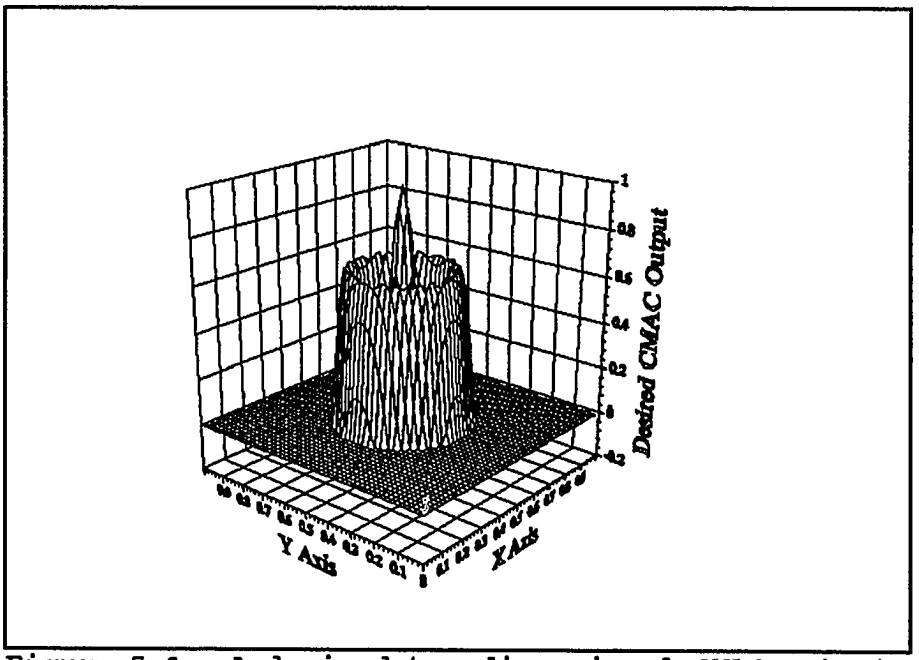

Figure 5.1 A desired two-dimensional CMAC output.

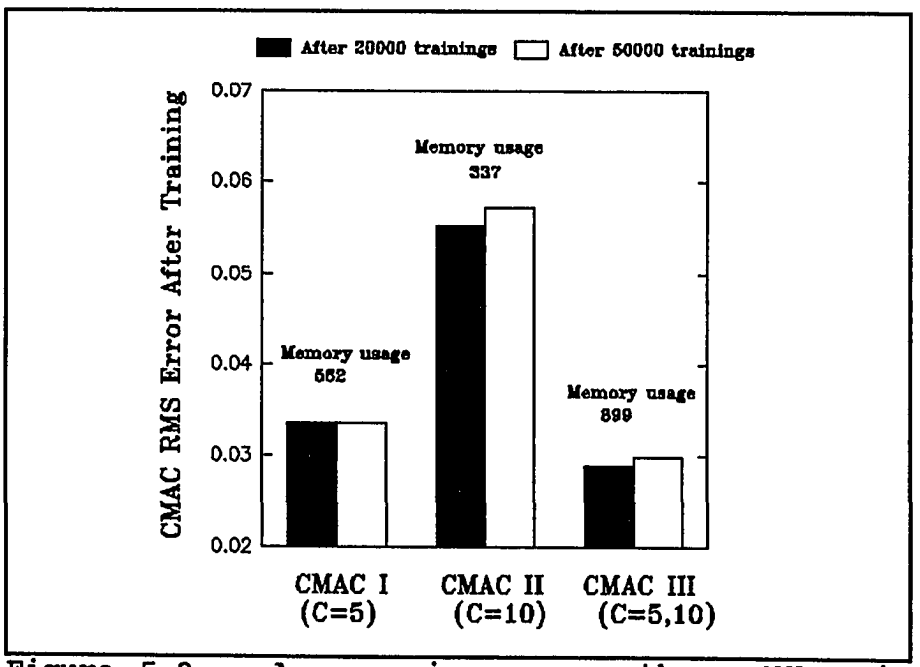

Figure 5.2 A comparison among three CMACs in terms of the training error after 20,000 and 50,000 random trainings.

vary during training. As fewer receptive fields are needed to represent a smoother function surface, one might expect fewer receptive fields would be needed in reconstructing the constant surface outside the circle centered at <0.5,0.5> (refer to figure 5.1) while more receptive fields would be needed in reconstructing the inner sinusoidal surface. This justifies the use of the adaptive receptive field density.

Experiments in this chapter were performed under the following conditions unless otherwise stated. There were 50 input quantization intervals (S) in each axis. All the training pairs were obtained by sampling at the input quantization intervals (and their intersections) in S^2. All the samples formed a training set. Each sample could take any value from 0 to 1. The receptive field was chosen to be linearly tapered with hypercubic support, and the center placement was set according to the heuristic rules (refer to chapter IV).

V.1 Multiple CMACs

While increasing the number of receptive fields in general reduces the approximation error (refer to figure 4.10-4.13), it is not clear whether to choose a CMAC with a smaller C (with uniform field widths) or to choose multiple CMACs with different field widths. Thus, three different CMAC structures were studied for learning.

The input generalization (C) for CMAC I was 5 input

quantization intervals wide, and that for CMAC II was 10 input quantization intervals wide. The learning rates (β) for both CMAC I and CMAC II were 0.5. The displacement vectors D for CMAC I and CMAC II were <1,2> and <1,3> respectively. Unlike CMAC I and CMAC II, CMAC III was made up of two individual CMACs with C set to 5 and 10 input quantization intervals wide, and the learning rate for each individual CMAC in CMAC III was 0.5. The absolute offset of the center placement in each individual CMAC in CMAC III was zero. The weight update δW_i for CMAC III was defined as

$$\delta W_i = \beta \cdot (Y_d(\underline{X}) - Y_5 - Y_{10}) \cdot f_i(\underline{X}) \;/\; \sum_i f^2_i(\underline{X}) \qquad i \in \text{CMAC III}$$

where Y_5 and Y_{10} are individual CMAC outputs with C set to 5 and 10 input quantization intervals respectively.

50,000 random trainings were performed for all three CMAC structures, and figure 5.2 shows the relationship between the RMS approximation errors after training and the weight usage after 20,000 and 50,000 random trainings.

The RMS errors did not change significantly in any of these CMACs after 30,000 additional trainings. This indicates enough trainings had been done to eliminate any transient behavior. As expected, CMAC I (C=5) produced much smaller RMS errors than CMAC II because more receptive fields (weights)

were available in CMAC I. In contrast, although there was a huge increase in weights available, CMAC III was not able to further reduce the error by any significant amount, as compared to CMAC I. Since the overall center coverage in CMAC III is no longer uniform (although each of the individual CMACs has uniform center coverage), this suggests that multiple CMACs with different individual field widths for approximating a function are undesirable in terms of the final approximation error for a fixed number of available weights (consistent with the result in Section IV.3).

V.2 Adaptive Receptive Field Density

Since fewer receptive fields are needed to approximate a smoother function surface and the field width should be kept uniform, a single CMAC with an adjustable weight density was used to approximate Y_d in this section. The input generalization was 10 input quantization intervals wide. Unlike a regular CMAC, even layers (the first layer and every other layer) of receptive fields were grouped together to form $CMAC_e$, while odd layers (the remaining layers) of receptive fields were grouped together to form $CMAC_o$. The learning rates for $CMAC_e$ and $CMAC_o$ were β_e and β_o respectively. If $CMAC_e$ and $CMAC_o$ were put together, a complete CMAC would be recovered. Although $CMAC_e$ and $CMAC_o$ each possessed half as much input resolution as their standard counterpart ($C = 10$), $CMAC_e$ and $CMAC_o$ each had a uniform field coverage (because the

displacement offsets of these sub-networks satisfy the heuristic rules).

Two phases ($\Phi 1$, $\Phi 2$) of training were involved. In $\Phi 1$, the learning rate was 0.5, and only the weights (W_{ei}) in $CMAC_e$ were adjusted as follows.

$$\delta W_{e,i} = 0.5 \cdot (Y_d(\underline{X}) - Y_e) \cdot f_{e,i}(\underline{X}) \;/\; \sum_i f^2_{e,i} \qquad i \in [\text{even}]$$

where Y_e is the output of $CMAC_e$.

$\Phi 2$ began after S_W training cycles in $\Phi 1$. In $\Phi 2$, the weights ($W_{e,i}$) in $CMAC_e$ were adjusted as follows.

$$\delta W_{e,i} = \beta_e \cdot (Y_d(\underline{X}) - Y_e - Y_o) \cdot f_{e,i} \;/\; \sum_i f^2_{e,i}(\underline{X}) \qquad i \in [\text{even}]$$

where Y_o is the output of $CMAC_o$.

The weights ($W_{o,i}$) in $CMAC_o$ were adjusted if the error residue at the beginning of each training cycle was greater than T_h, which is an error threshold level parameter.

$$\delta W_{o,i} = \beta_o \cdot (Y_d(\underline{X}) - Y_e - Y_o) \cdot f_{o,i}(\underline{X}) \;/\; \sum_i f^2_{o,i} \qquad i \in [\text{odd}]$$

With this adaptive receptive field density configuration, four new parameters are introduced: S_w, T_h, β_e and β_o.

Figure 5.3 shows the relationship between the standard deviation of the approximation error and eleven different parameter arrangements after 20,000 and 50,000 random trainings. The weight usage was also shown on the right side of each bar in the figure. In $\Phi 2$, only one or two parameters were varied in each case while the rest of the parameters were kept at their nominal values. Under the nominal condition (shown on the right side of figure 5.3), β_e and β_o were each set to 0.1, S_w was set to 5000, and T_h was set to 0.05.

Several special cases are worth noting. In case 1, only Y_e was used for training (S_w was very large) with β_e set to 0.5. In case 2, only Y_e was used for training (T_h was very large) with β_e set to 0.1 during $\Phi 2$. In case 3, both Y_e and Y_o were always used in each training cycle (because T_h was set to 0) with β_e set to 0.5 and β_o set to 0.1. The rest of the cases can be interpreted in a similar manner.

Results from cases 4, 8 and 9 show that the time to switch to $\Phi 2$ is not critical, which indicates that the general surface of the desired output was well approximated in $\Phi 1$.

Results from case 1 and 2 show that if only Y_e was used throughout the training, a smaller learning rate in $\Phi 2$ produced a smaller RMS error. This suggests that a relatively good approximated surface was developed in $\Phi 1$. In $\Phi 2$, any extra training would appear as a random perturbation on the

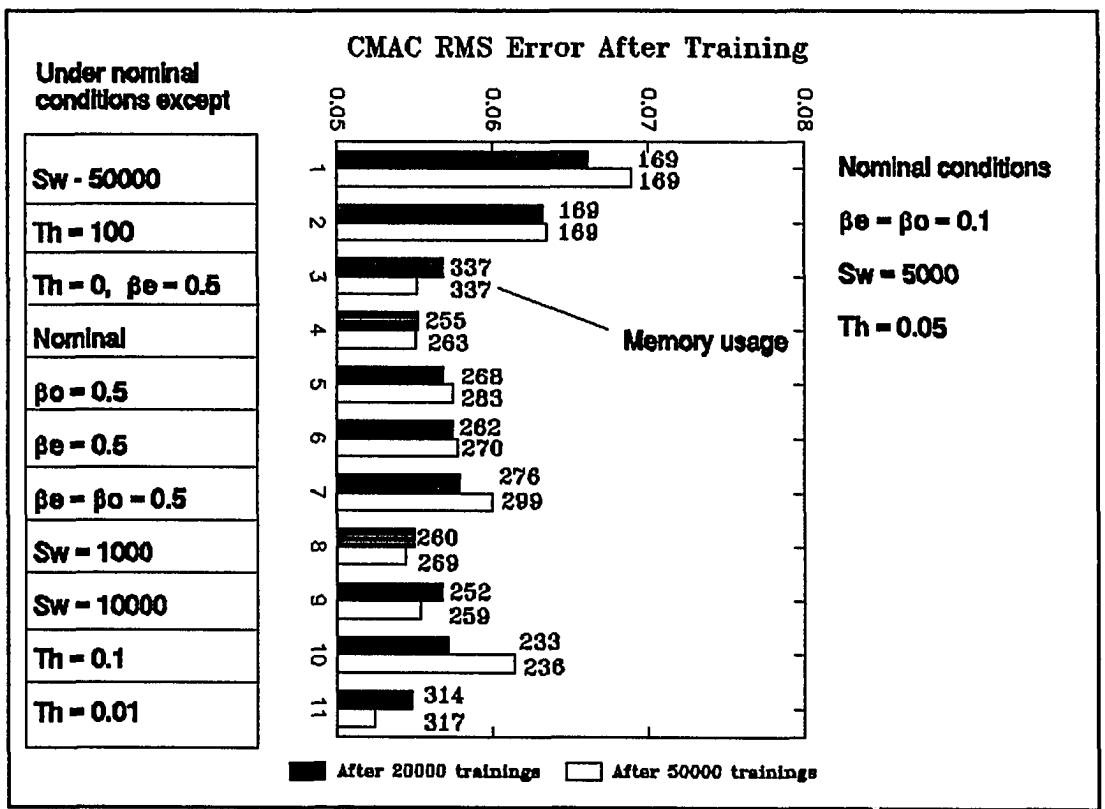

Figure 5.3 The effect of parameters in the adaptive receptive field density CMAC.

network output. Similar observations were found in CMAC II and CMAC III (figure 5.2). Also, results from case 4, 5, 6 and 7 show that smaller learning rates for both Y_e and Y_o produced smaller RMS errors and lower weight usage.

Results from case 4, 10 and 11 show that a smaller T_h produced a smaller RMS error at the expense of a larger number of weights used.

An optimal set of parameters has yet been formulated for a given desired output. Results from cases 3 and 4 clearly indicate that the number of weights used can be reduced significantly (23% weight reduction) by dividing the full size CMAC into two sub-halves. In general, the network is not only restricted to form two sub-halves. The CMAC network can be divided into any relative ratios of many sub-networks as long as the individual displacement offsets satisfy the heuristic rules. However, it is worth noticing that the size of the weight reduction depends on the desired network output.

CHAPTER VI

SUMMARY

This chapter summarizes observations made in previous chapters on the effect of the receptive field function and placement in CMAC on learning a space-spanning desired network output in S^N. A bound on the receptive field width or generalization in CMAC as a function of the input dimension is proposed. This chapter also suggests several issues on the CMAC architecture which deserve further research.

Modular Receptive Field Placement - The center placement in most receptive field networks is either random or adaptive. A time-consuming search process is required to locate neighboring receptive fields to generate a network output, which increases the computation time for a large N. In contrast to most receptive field networks, CMAC has a unique modular center placement in S^N, which completely avoids the search process.

Receptive Field Function and Critical Spatial Frequency - In the CMAC neural network, modular center placement is not the only feature which affects proper learning or approximation. Receptive field function is also a critical feature in reconstructing the desired output in S^N. The use of

a rectangular field function in the standard CMAC is undesirable for two reasons. First, the network output is not sensitive to variation of input within a single input quantization interval. Second, convergence is slow at the critical spatial frequency (f_{cr}) when the sum of all the desired outputs within a single receptive field is exactly zero. The use of a linearly tapered field function allows the network output to be continuous and also provides a wider learning bandwidth. However, the network with the linearly tapered field function becomes sensitive to the center placement inside C^N.

Diagonal Center Placement - In the standard CMAC, centers of receptive fields are placed on the hyper-diagonal inside C^N. Increasing C not only leads to fewer available receptive fields in S^N, but also forces the center placement to be less uniform in C^N. Thus, learning is particularly hard for a larger C because the coverage is more sparse as well as less uniform. In addition, a large C gives rise to a small f_{cr}. This inhibits the network from learning the high spatial harmonics embedded in the desired output.

Heuristic Center Placement - A more uniform center placement can be generated by utilizing the modulo arithmetic method. Three heuristic rules can be used to determine the displacement offsets in each input dimension. HR1 ensures that

the projection of centers onto each axis is uniform which allows the input resolution to be uniform in each axis. HR2 ensures that the projection of centers onto any subspace can not be diagonal. HR3 ensures that the projection of centers will not be clustered in some local region ($\subset C^N$). It is important to notice that these rules do not guarantee a uniform projection of centers onto any higher dimensional subspace. In other words, the center placement (by the heuristic rules) is not necessarily perfectly uniform. However, a perfectly uniform center placement in S^2 can be established if the displacement offset satisfies the constraint shown in Section 4 of chapter IV. Also, a lower bound on C $(2 \cdot N)$ for any N emerged from these heuristic rules, resulting in a more reliable determination of C for a given problem.

Receptive Field Region of Support - A hypercubic region of support for the linearly-tapered field function was shown to be better than a hyperspherical region of support (see chapter IV). However, the hypercubic support creates discontinuities in slope on the hyper-diagonals of the receptive field. While this might not have any direct adverse effect in approximating a function, the discontinuous region could be undesirable for some closed-loop control problems that require function derivatives [9]. Parks [21] suggests a field contour based on a super-ellipse function. In the super-

ellipse function, the field contours near the center of the receptive field appear radially symmetric, while those near the edge of the receptive field appear hypercubic. Although this receptive field function is inherently complex for learning computations, the discontinuity in slope along the hyperdiagonal is eliminated. Lane et al. [16] have proposed the use of multi-dimensional B-splines to form receptive field functions for CMAC neural networks. The generality of this approach is limited however by the complexity of B-splines functions in spaces of more than three dimensions.

Non-Uniform Field Width - A CMAC network with non-uniform field widths always results in a non-uniform receptive field placement, which is highly undesirable in terms of the approximation error for a fixed number of available weights. This also suggests that learning with multiple CMACs with different individual field widths is undesirable in terms of the final approximation error for a fixed number of available weights. Moody [19] has suggested a multi-resolution CMAC in which the receptive field widths are different in different layers. However, in [19], no direct comparison between the standard CMAC and his multi-resolution CMAC was made in terms of the speed of convergence or the final approximation error. It is speculated that the multi-resolution scheme for the receptive field width might be useful in terms of the speed of convergence, but not in terms of the final approximation error

given a fixed number of weights. This speculation deserves further investigation.

Adaptive Receptive Field Density - A uniform center placement is a reasonable placement given that there is no prior knowledge available about the desired network output. The uniform center placement avoids unnecessary network biases to the desired output. However, in the presence of training data which may not have uniform spatial frequency content throughout the space, the receptive field density in the network should be allowed to adapt in order to approximate the desired output more efficiently. Although the center placement with the adjustable receptive field density might be less uniform in S^N, the center placement remains uniform as seen by each training input. With this adjustable receptive field density feature, the network would require fewer receptive fields to approximate the desired output than the network with a fixed receptive field density.

Bound on the Receptive Field Width (C) - Results in this dissertation have led to a better understanding of the effect of uniformity of the receptive field placement on C. The proposed bound on C is shown in equation 6.1.

$$2 \cdot N \leq C \leq \min(P_i) \qquad \text{eq. 6.1}$$

where P_i stands for the spatial period of the Fourier

component i of the desired output, and is measured by the number of input quantization intervals.

It is very interesting to notice that the lower bound on C grows only linearly with the input dimension to achieve uniform input resolution in each axis, and is fixed once the input dimension is chosen. On the other hand, the upper bound depends on the input scaling (S). A smaller S results in a smaller $\min(P_i)$ because P_i is measured by the number of the input quantization intervals. Also, a smaller S implies there are fewer available receptive fields in S^N. Too small a S could have sub-optimal approximation performance because there might not be any C which satisfies equation 6.1. In general, C is determined by N, S and P_i.

FUTURE RESEARCH

Better Receptive Field Placement - The heuristic placement provides a uniform projection of centers onto each axis, and also guarantees the projection of centers onto subspaces are not diagonal. However, there has not been any theoretical proof of the existence (or non-existence) of a better placement. A proof of the optimality of the heuristic rules for the displacement vector selection would be most worthwhile.

Dimensionality Reduction - Although the desired output is defined in R^N, the desired output might be very dense only in some subspaces (R^M, $M < N$). In this case, it might be more efficient to allocate the receptive fields only in R^M when the issue of the approximation error and the limited weight storage are critical.

Partitioning of the input space - In the standard CMAC, the entire input space is partitioned into hypercubes. Results in Chapter IV.5 indicate that the hypercubic receptive field support is desirable because of the hypercubic partitioning in CMAC. However, other partitioning in the input space should not be ruled out. Given that the goal is to achieve fast network computation, the partitioning should be done so that all of the individual building blocks are modular. A search for neighboring receptive fields is thus avoided.

LIST OF REFERENCES

[1] J.S. Albus, "Brain, Behavior and Robotics", Byte Books, 1981.

[2] J.S. Albus, "A New Approach to Manipulator Control: the Cerebellar Model Articulation Controller (CMAC)", Transactions of the ASME, p220-227, September 1975.

[3] J.S. Albus, "Theoretical and Experimental Aspects of a Cerebellar Model", PhD dissertation, University of Maryland, December, 1972.

[4] P.C. E. An, W.T. Miller, P.C. Parks, "Design Improvements in Associative Memories for Cerebellar Model Articulation Controllers (CMAC)", 1st International Conference on Artificial Neural Networks, June 1991, Helsinki University of Technology.

[5] M.J. Carter, A. Nucci, W.T. Miller, P.C. E. An, F. Rudolph, "Slow Learning in CMAC Networks and Implications for Fault Tolerance", 24th Annual Conference on Information Sciences and Systems, Princeton University, March 1990.

[6] M. Gardner, "Mathematical Carnival", Chapter 18, Vintage Books, Random House, New York, 1977.

[7] S. Grossberg, The Adaptive Brain, Vol. II: Vision, Speech, Language, and Motor Control. Amsterdam, North-Holland.

[8] J.J Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", Proceeding of National Academy of Sciences, Vol.79, p2554-2558, April, 1982.

[9] M. Jordan, "Supervised Learning and Systems with Excess Degrees of Freedom", COINS Tech. Rep. 88-27, Amherst, MA, University of Massachusetts, Computer and Information Sciences, 1988.

[10] P. Kanerva, "Sparse Distributed Memory", MIT Press, Cambridge, MA, 1988.

[11] T. Kohonen, "Clustering, Taxonomy, and Topological Maps of Patterns", Proc. Sixth Int. Conf. on Pattern Recognition, p114-128, Munich, Germany, 1982.

[12] T. Kohonen, "Self-Organization and Associative Memory", Springer-Verlag, Heidelberg, 1984.

[13] T. Kohonen, "Adaptive, Associative, and Self-Organizing Functions in Neural Computing", Vol.26, No.23, Applied Optics, December 1987.

[14] T. Kohonen, "The Self-Organizing Map", Proceedings of the IEEE, Vol.78, No.9, p1464-1480, September 1990.

[15] L.G. Kraft, D.P. Campagna, "A Comparison of Neural Network and Traditional Adaptive Controllers", Automatic Controls Conference, Pittsburg, PA, June 1989.

[16] S. Lane, D. Handelman, J. Gelfand, "Higher-Order CMAC Neural Networks - Theory and Practice", Proceedings of the 1991 American Control Conference, Boston MA, June 26-28, 1991.

[17] W.T. Miller, F.H. Glanz, L.G. Kraft, "Application of a General Learning Algorithm to the Control of Robotic Manipulators", International Journal of Robotics Research, Vol.6, No.2, p84-98, 1987.

[18] W.T. Miller, "Sensor Based Control of Robotic Manipulators Using a General Learning Algorithm", IEEE Journal of Robotics and Automation, Vol.RA-3, No.2, p157-165, 1987.

[19] J. Moody, C. Darken, "Fast Learning in Networks of Locally-Tuned Processing Units", Neural Computation, Vol.1, No.2, p281-294, 1989.

[20] P.C. Parks, J. Militzer, "Improved Allocation of Weights for Associative Memory Storage in Learning Control Systems", Proc. IFAC Symposium on Design Methods of Control Systems, Zurich, July 1991.

[21] P.C. Parks, personal communication.

[22] T. Poggio, F. Girosi, "Networks for Approximation and Learning", Proceedings of the IEEE, Vol.78, No.9, p1481-1497, September 1990.

[23] T. Poggio, F. Girosi, "A Theory of Networks for Approximation and Learning", AI Memo No. 1140, Artificial Intelligence Laboratory, MIT, 1989.

[24] D.E. Rumelhart, G.E. Hinton, R.J. Williams, "Learning Internal Representation by Error Propagation", in Parallel Distributed Processing, Ch.8, p318-362, MIT Press, Cambridge, MA 1986.

[25] H.Seung, H. Sompolinsky, N. Tishby, "Learning Curves in Large Neural Networks", to appear in COLT 91.

[26] B. Widrow, M. Lehr, "30 Years of Adaptive Neural

Networks: Perceptron, Madaline, and Backpropagation", Proceedings of the IEEE, Vol.78, No.9, p1415-1442, September 1990.