ELSEVIER

Neurocomputing 29 (1999) 57–84

NEUROCOMPUTING

www.elsevier.com/locate/neucom

The general memory neural network and its relationship with basis function architectures

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

[a]*Department of Electrical and Computer Engineering, University of Colorado at Colorado Springs, 1420 Austin Bluffs Parkway, Colorado Springs, CO 80918, USA*

[b]*Department of Electrical Engineering and Electronics, UMIST, P.O. Box 88, Manchester, M60 1QD, UK*

Received 15 January 1998; accepted 26 August 1998

Abstract

A generalization of a class of neural network architectures based on a multiple quantization of input space combined with memory lookup operations is presented under the name of a general memory neural network (GMNN). Within this common framework it is shown that networks of this type are – for a variety of learning schemes – response-equivalent to basis function networks (i.e., radial basis function and kernel regression networks). In particular, this equivalence holds even if a GMNN does not employ explicit basis functions, which makes the architecture attractive from an implementational point of view and allows fast operation, both in the learning and response modes. Variants of the GMNN are discussed and examples of existing architectures conforming to this common framework are given. © 1999 Published by Elsevier Science B.V. All rights reserved.

Keywords: Memory-based networks; Basis functions; N-tuple sampling; RAM neurons; RAMnet; RBF; GRNN; CMAC; SLLP

1. Introduction

Let us consider the common problem of estimating (approximating) a multivariate function

$$f: \Re^D \supset \Omega \rightarrow \Re$$

defined on a compact subset Ω of a D-dimensional real space, where the goal is to construct a function g such that a suitably defined distance $\pi(g,f)$ between the estimate

*Corresponding author.
E-mail address: ark@eas.uccs.edu (A. Kołcz)

and the unknown function is minimized [21,35]. It is assumed that the unknown mapping is both continuous and smooth, and that the information about f is limited to a finite set of input–output pairs

$$\mathscr{T} = \{\boldsymbol{x}^t, y^t\}_{t=1}^{T} \quad \text{where } y^t = f(\boldsymbol{x}^t) + \varepsilon^t,$$

where ε^t represents the noise (error) component representing our uncertainty about the data-generation process. In the general (statistical) case, $\boldsymbol{x}^t$ and y^t are instances of random variables X and Y, respectively, whereas ε^t represents an instance of a random noise variable $\mathscr{E}$.

Recent advances in the field of artificial neural networks have shown that a large class of neural network architectures (including multilayer perceptrons (MLPs) [21] and radial basis functions (RBFs) [29,34] possess the universal approximation property [16,18,42] and is therefore able to approximate any continuous smooth mapping arbitrarily well (given sufficient amounts of training data and implementational resources). These results provide a good theoretical foundation for applying such networks to practical problems. On the other hand, the theoretically analysed forms of networks such as RBF (e.g., with one basis function per training point) are often computationally too expensive in practice and various heuristic approaches (e.g., basis-set reduction via subset selection [12] or clustering [29]) are used to enable processing of large data sets with acceptable speed. Such modified networks tend to perform well in practice [28,29,32], although their formal analysis is more difficult.

There also exist neural network architectures that are known to perform well in practical applications and are computationally efficient, but whose capabilities are significantly less understood than those of more mainstream architectures. In this work we concentrate on one particular class of such networks and show that there are, in fact, close links between this class and the well-known basis function architectures (i.e., the RBF and kernel regression (KR) [19,20,37] networks). Networks that belong to this class consist essentially of a single layer of memory cells (containing adjustable weights) and an address generating unit, which selects a *fixed* number of memory weights during every response computation. If the address generating part of such a network is fixed throughout its operation, and if the number of participating weights is small, the network response (usually obtained by summing the selected weights) can be very fast, which makes it suitable for real-time applications [39]. Additionally, since the memory weights are the only adjustable parameters, the network output is linear in terms of the adjustable parameters and reliable learning schemes (e.g., least mean square [43]) can be employed. At the same time, the nonlinearity inherently involved in the generation of memory addresses ensures that a large class of nonlinear mappings can be realized by these networks. Particular examples include the cerebellar model articulation controller (CMAC) [2,13] and the N-tuple network [9,10,5,38], which have tended to be analysed on an individual level. Here we concentrate on the similarities between different variants of such networks and show how a general framework (which we name a general memory neural network (GMNN)) may be adequate to characterize their properties.

The paper is organized as follows. In Section 2 we present an example of a simple network (based on a histogram regression estimator) that illustrates the main idea behind the multiple input-space quantization, characteristic of the class of networks discussed in this paper. Section 3 introduces a formal definition of the GMNN architecture and addresses certain issues involved in the implementation of such networks. In Section 4 we discuss aspects of the input-space quantization present in a GMNN mapping, distinguishing between network variants performing multiple quantization of Ω in an implicit and explicit way. Section 5 introduces certain important parameters used to describe the GMNN mapping. A further use of them is made in Section 6, which establishes a relation between the GMNN and basis function networks of the RBF and KR type. In Section 7 three examples of existing forms of the GMNN architecture are presented and the paper is concluded in Section 8.

2. The (multiple) histogram approach to regression estimation

One simple way of obtaining an estimate of f is to apply a form of the histogram method, where the input domain is divided into a number of disjoint cells (i.e., it is quantized) and the value of f for points inside a single cell is obtained by averaging the values of the training-set points falling within this cell [19].

Applicability of the histogram method is somewhat limited, however, as in order to obtain a "reasonable" resolution of the estimate it is necessary to divide the input domain into a large number of small cells – but, to ensure that each cell contains a statistically valid estimate of f, its size should be large enough to contain at least one training point. One method to overcome this problem is to vary the histogram-cell size according to the local density of data [11]. Such adaptive methods of network design can become computationally expensive for large training sets, especially in high-dimensional settings, where excessively large training-set sizes would have to be used to provide even a moderate resolution of the estimate over Ω, as the number of histogram cells tends to grow exponentially with D. These difficulties associated with the approximation of multivariate functions are common to many methods and are often known as "the curse of dimensionality" [35].

Now, consider an approach where instead of just one, several (say K) separate histogram estimates of f are obtained in such a way that the cell arrangements associated with the individual histogram designs are different from one another and no two cells belonging to distinct histogram designs are identical. It follows that every point in Ω lies in a region resulting from an intersection of K overlapping cells belonging to the individual histogram quantizers. As a result, even though the individual quantization cells can be relatively large (thus ensuring a significant content of the training-set points falling into them), their intersections can result in a sufficiently fine partitioning of Ω, so adequate overall network resolution is obtained. Such a network can be visualized as a set of parallel quantization layers, which, when superimposed on each other, yield the final cell arrangement (see Fig. 1), with the network response being constant within each of the resulting cells.

Let us consider the overall network response resulting from this scheme. For any particular network input, $\boldsymbol{x}$, the response is defined as an average of the individual

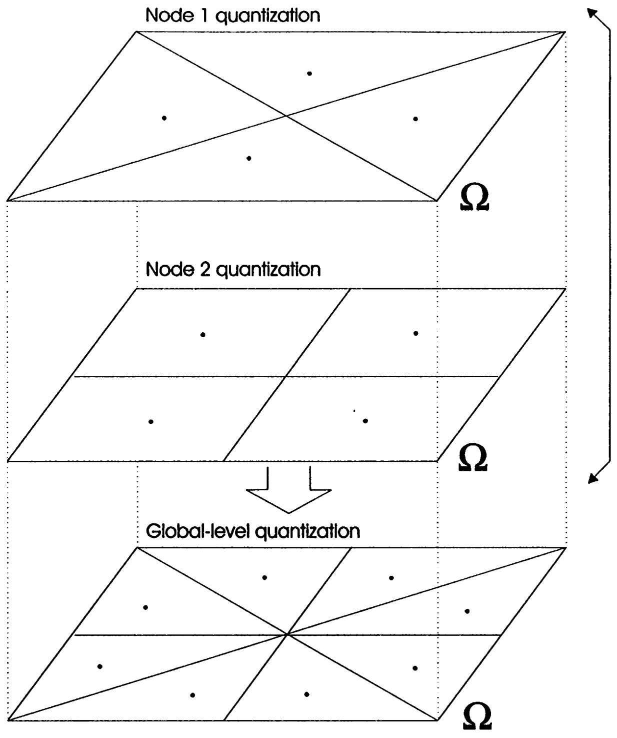


Fig. 1. A superposition of distinct low-resolution quantizations leads to a global VQ of a higher resolution.

histogram estimates, i.e.,

$$g(\boldsymbol{x}) = \hat{f}(\boldsymbol{x}) = \frac{1}{K}\sum_{k=1}^{K} \hat{y}_k(\boldsymbol{x}), \tag{1}$$

where $\hat{y}_k(\boldsymbol{x})$ denotes the estimate of $f(\boldsymbol{x})$ obtained by the kth histogram and is given by

$$\hat{y}_k(\boldsymbol{x}) = \frac{\sum_{t=1}^{T} y^t[\boldsymbol{x}^t \in \mathscr{R}_k(\boldsymbol{x})]}{\sum_{t=1}^{T}[\boldsymbol{x}^t \in \mathscr{R}_k(\boldsymbol{x})]}, \tag{2}$$

where $\mathscr{R}_k(\boldsymbol{x})$ is the cell selected by $\boldsymbol{x}$ in the kth histogram network. We follow here Iverson's notation [22], by which any logic (i.e., true-or-false) statement enclosed in square brackets evaluates to 1 if it is true; otherwise it evaluates to 0. Consequently, the denominator of Eq. (2) is equal to the number of training points falling into $\mathscr{R}_k(\boldsymbol{x})$ and will be denoted by $c_k(\boldsymbol{x})$, i.e., $c_k(\boldsymbol{x}) = \sum_{t=1}^{T}[\boldsymbol{x}^t \in \mathscr{R}_k(\boldsymbol{x})]$.

An obvious tradeoff involved in a design of a multi-histogram network concerns the effective resolution of response versus the number of quantizers, K. It might seem that with increasing K it is possible to obtain arbitrarily small intersections of quantization cells. Note, however, that the network response is created as an average of the K histogram estimates. It will thus be "oversmoothed" when K becomes too large (each histogram estimate may be potentially influenced by training-set points at a considerable distance from the current evaluation point). Therefore, a suitable value of K should be determined on the basis of information provided by the training data – for example, using cross-validation techniques [19,20].

The network response (1) might be further modified by assigning more weight to those currently selected cells that contain more training-set points, e.g.,

$$g(\boldsymbol{x}) = \frac{\sum_{k=1}^{K} \hat{y}_k(\boldsymbol{x}) c_k(\boldsymbol{x})}{\sum_{k=1}^{K} c_k(\boldsymbol{x})}. \tag{3}$$

As discussed in [24] and later in this paper, this form of output response is in fact equivalent to a KR network [20], with piecewise-constant kernels whose support is determined by the network quantization cells.

The histogram example represents a class of networks that perform a K-fold quantization of Ω and create their responses as a sum of weights associated with the K cells selected by the network input. Networks of this type can be considered as consisting of K individual subnetworks, each providing a constant-within-cell response determined by a numeric weight associated with a quantization cell.

It can be seen that a quantization subnetwork is in fact equivalent to a weight memory store, with the cell-selection process corresponding to choosing the appropriate memory address. The whole network consists of K weight memories (and their associated address generating networks) and an output unit, which combines the selected weights. Consequently, networks of this type can be thought of as extensions of look-up memories, and terms such as memory (RAM) based networks are often applied [5].

Obviously, particular memory-based networks will differ with respect to the input-space quantization (i.e., memory addressing techniques) employed, the structure of the weight memories, as well as the way in which the overall network response is produced. However, many important properties of these architectures can be analysed at a general level. To facilitate such an analysis we propose a common framework for memory-based networks referred to by the generic term general memory neural network (GMNN).

3. The GMNN architecture

3.1. Network definition

Following the histogram example, we consider a class of networks performing a mapping $\Re^D \supset \Omega \to \Re$, employing K separate and distinct quantizations of the input space. A common architecture of a general memory neural network encompassing this class can be formally defined to consist of [24]:

1. A set of K memory nodes, with the kth node having $|\mathscr{A}_k|$ addressable locations. Typically, a location will comprise one or more numeric values in a real or integer format.

2. An address generating network, assigning an address vector $A(\boldsymbol{x}) = [A_1(\boldsymbol{x}), \dots, A_K(\boldsymbol{x})]$ to every point $\boldsymbol{x}$ from the input domain; $A_k(\boldsymbol{x})$ denotes the address generated for the kth memory node.
3. An output functional

$$\Xi\colon [W_1(A_1(\boldsymbol{x})), \dots, W_K(A_K(\boldsymbol{x}))] \to \Re$$

combining the contents of memory locations selected in the respective memory nodes. $W_k(A_k(\boldsymbol{x}))$ denotes the memory contents selected by $(A_k(\boldsymbol{x}))$ and will be further designated as $W_k(\boldsymbol{x})$. The vector of selected memory locations will be denoted by

$$W(\boldsymbol{x}) = [W_1(\boldsymbol{x}), \dots, W_K(\boldsymbol{x})]$$

and the combined mapping performed by the network by g

$$g : \Omega \ni \boldsymbol{x} \to A(\boldsymbol{x}) \to W(\boldsymbol{x}) \to g(\boldsymbol{x}) \in \Re.$$

4. A learning procedure that allows the network to adjust the values of nodal memory contents, based on the training set, so that some error criterion $\pi(f,g)$ is minimized.

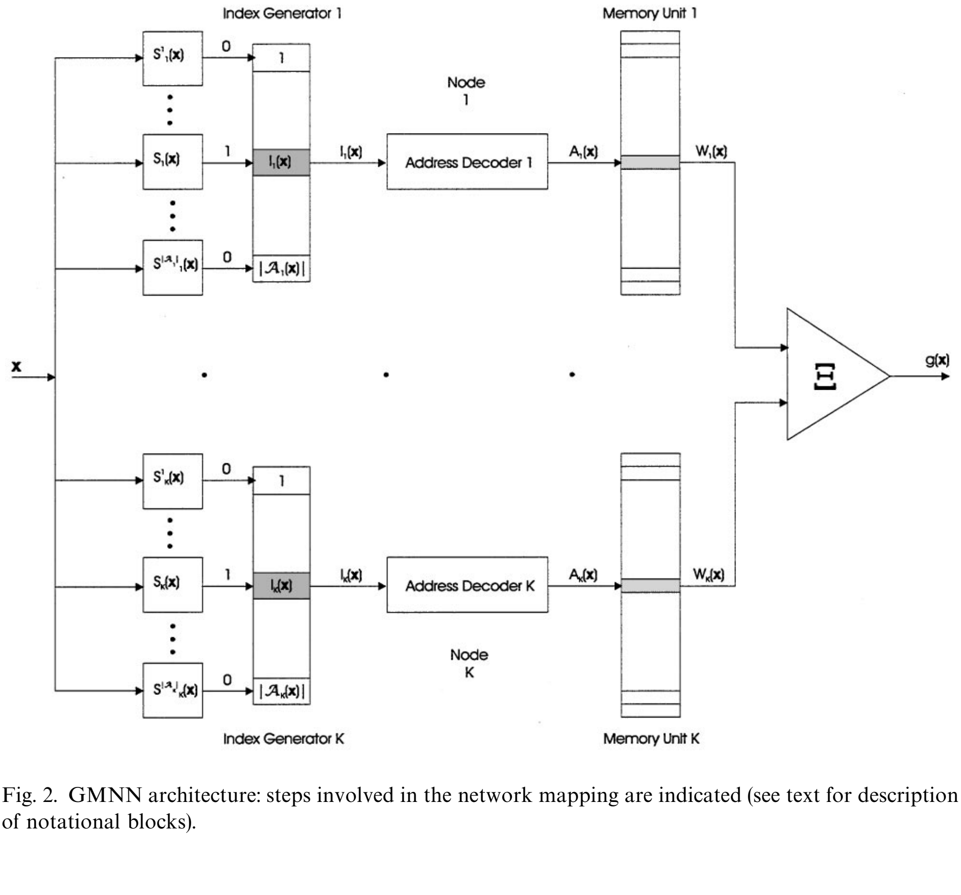


Fig. 2. GMNN architecture: steps involved in the network mapping are indicated (see text for description of notational blocks).

Fig. 2 illustrates the general form of the GMNN architecture. Some elements present in this figure (e.g., selector functions) will be explained further while discussing the quantization performed by the network. Variants of the network with respect to the format of memory contents and the form of the output response will also be discussed in subsequent sections. Here, we define the basic form of the network as one where every memory location contains a numeric weight (i.e., $W_k(\boldsymbol{x}) \equiv w_k(\boldsymbol{x}) \in \Re$) and the network response is given by the sum (or average) of the selected weights, i.e.,

$$g(\boldsymbol{x}) = \sum_{k=1}^{K} w_k(\boldsymbol{x}) \quad \text{or} \quad g(\boldsymbol{x}) = \frac{1}{K} \sum_{k=1}^{K} w_k(\boldsymbol{x}). \tag{4}$$

According to Eq. (4), provided that the address generation part of GMNN mapping is fixed during network operation (which will be assumed throughout this paper), the network response is linear with respect to its adjustable parameters.

4. Quantization aspects of the GMNN mapping

4.1. Implicit quantization schemes

The definition of the GMNN as well as the histogram example explicitly indicate K separate quantization layers. This approach is indeed taken in a number of existing variants of the GMNN architecture, such as the CMAC [2], N-tuple [10] and B-spline [28] networks.

However, a similar functionality can be obtained when certain computational shortcuts are applied to "traditional" basis function networks (especially when the employed basis functions have localised but unbounded support). For example, in the case of RBF and KR networks it is in many cases ineffective to evaluate all of the network's basis functions at a given input point, as the vast majority of them will often be approximately zero. Instead, only those functions that significantly contribute to the network's response are chosen. Alternatively, one might choose only those basis functions whose centres lie sufficiently close to the current evaluation point (this approach is closely related to choosing suitable basis functions of *finite* support at the outset, so that the set of non-zero bases at a given point can be determined using appropriate distance measures [20]).

In either of these approaches, each of the network's basis functions is "active" only inside a certain region of the input space, representing its effective support and the set of active functions is determined by the network input. If the number of participating bases is fixed to, say K, the structure of the GMNN emerges, where in principle it is possible to identify the K separate quantization layers.

To see why this is the case, let us consider the set of K basis functions (and their associated support regions) selected for a particular input point, $\boldsymbol{x}$. By choosing one of the K active basis functions (say b), and by considering points in a progressively increasing neighbourhood of $\boldsymbol{x}$, it is possible to identify the set of all basis functions that are not in the original set and become active (i.e., replace b in the set of K active functions) as the support boundary of b is crossed. As a basis function is always active within its support and K functions must be active at any point in Ω, the support regions of the identified functions are adjacent to (and non-overlapping with) the support of b.

By recursively repeating this process for each of the adjacent bases it is possible to identify a layer of basis functions whose support regions partition Ω into a set of adjacent, non-overlapping cells. Since this process can be performed in turn for all of the K original non-overlapping active bases (selected for $\boldsymbol{x}$), it is thus possible to distinguish the K distinct quantization layers of the network. This variant of the GMNN architecture can be considered as a combination of a basis function network with a K-nearest-neighbour selection rule to identify the set of active bases for any input point, $\boldsymbol{x}$.

Fig. 3 presents a GMNN as performing nodal quantization implicitly. Here, for each network input a K-element address *vector* is selected, with the vector being constant for points in a global-level quantization region. It is easily seen that this process is in fact equivalent to selecting the address-vector components individually by separate nodal quantizers (as in Fig. 2). The meaning of some symbols presented in Fig. 3 will be explained later in this section, when global-level input-space quantization performed by a GMNN is discussed.

4.2. Quantization at a single-node level

We have seen that whatever method of vector quantization is performed (i.e., implicit or explicit) it is possible to define K separate partitions of Ω. Considering the compact nature of the input space, each of the K quantizers will have a finite number of cells, with every cell being closed and bounded [17]. The set of cells corresponding to the kth quantizer will be denoted by $\mathscr{A}_k$ and their number by $|\mathscr{A}_k|$.

Let us consider the kth network node. For any input $\boldsymbol{x}$ a unique quantization cell, $\mathscr{R}_k(\boldsymbol{x})$, is associated with $\boldsymbol{x}$ at that node, which can also be described as assigning to $\boldsymbol{x}$ an index value, $I_k(\boldsymbol{x})$, where every cell is associated with a unique index. Thus if $I_k(\boldsymbol{x}) = i \in \{1, \ldots, |\mathscr{A}_k|\}$, the corresponding quantization region is defined by

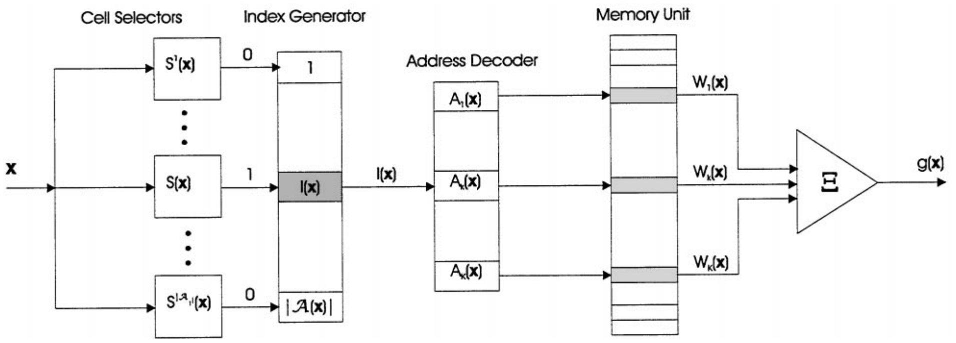


Fig. 3. GMNN architecture with input-space quantization performed at a global level. This interpretation is equivalent to the nodal view of the network, presented in Fig. 2.

$\mathscr{R}_k^i = \mathscr{R}_k(\boldsymbol{x}) = \{\boldsymbol{x} \in \Omega : I_k(\boldsymbol{x}) = i\}$. We can formally define selector functions associated with individual quantization cells, such that the selector function $S_k^i(\boldsymbol{x})$ is equal to unity inside the corresponding region, $\mathscr{R}_k^i$, and is zero outside, i.e., $S_k^i(\boldsymbol{x}) = [\boldsymbol{x} \in \mathscr{R}_k^i]$. The (one) selector function that is non-zero for $\boldsymbol{x}$ at the kth network node will be denoted by $S_k(\boldsymbol{x})$. Considering the properties of cell-selector functions, the basic forms of the GMNN output response (4) are equivalent to

$$g(\boldsymbol{x}) = \sum_{k=1}^{K} w_k(\boldsymbol{x}) S_k(\boldsymbol{x}) \quad \text{and} \quad g(\boldsymbol{x}) = \frac{\sum_{k=1}^{K} w_k(\boldsymbol{x}) S_k(\boldsymbol{x})}{\sum_{k=1}^{K} S_k(\boldsymbol{x})}. \tag{5}$$

The role of the selector and index functions is also illustrated in Fig. 2.

4.3. Quantization at the global level

When the cell arrangements corresponding to the individual network nodes are superimposed on each other, a global quantization structure emerges, where every quantization cell is a product of an intersection of K nodal quantization cells. The set of quantization regions at the global network level will be denoted by $\mathscr{A}$, and their number by $|\mathscr{A}|$. It follows that $|\mathscr{A}|$ must be at least as large as the highest resolution among the nodal quantizers, with the upper limit given by the number of all possible arrangements of nodal indices. Some index combinations are not allowed due to the particular structure of the quantizers. Consequently, $|\mathscr{A}|$ satisfies

$$|\mathscr{A}|_{\min} \le |\mathscr{A}| \le |\mathscr{A}|_{\max},$$

$$|\mathscr{A}|_{\min} = \max_k |\mathscr{A}_k|,$$

$$|\mathscr{A}|_{\max} = \sum_{i_1=1}^{|\mathscr{A}_1|} \cdots \sum_{i_K=1}^{|\mathscr{A}_K|} \prod_{k=1}^{K} [\{\boldsymbol{x} \in \Omega : I_k(\boldsymbol{x}) = i_k\} \ne \emptyset].$$

The global-level quantization cells and their selector functions are defined analogously to the single-node case, and are given by

$$\mathscr{R}^i = \mathscr{R}(\boldsymbol{x}) = \{\boldsymbol{x} \in \Omega : I(\boldsymbol{x}) = i\}, \quad i = 1, \ldots, |\mathscr{A}|,$$

$$S^i(\boldsymbol{x}) = [\boldsymbol{x} \in \mathscr{R}^i].$$

Here, $I(\boldsymbol{x})$ denotes the global-level index function. The significance of the index and selector functions is also shown in Fig. 3.

4.4. Cell-activation functions

In the simplest GMNN form (Eqs. (4) and (5)) a piecewise-constant output is produced, which suitability depends on the particular application, and in many cases it may be adequate. However, when a "smooth" solution is desired, the GMNN should provide gradual transition between the values corresponding to neighbouring quantization regions, as well as a suitable variability of response within a single quantization region [15].

One way of achieving this effect is by modulating the "flat" cell selectors by additional functions, which achieve a maximum at a cell's centre and gradually approach zero at the cell's boundaries. It is assumed that activation functions are positive within their associated cells and zero outside – that is, their support is defined by the associated quantization regions. Consequently, if μ_k^i denotes the activation function associated with $\mathscr{R}_k^i$ then $\mu_k^i = \mu_k^i \cdot S_k^i$. Conversely, the selector function, S_k^i, can be obtained by performing a threshold operation on the corresponding activation function, μ_k^i, which is positive inside the cell, $\mathscr{R}_k^i$, and zero outside, i.e., $S_k^i = h(\mu_k^i)$, where $h(x) = [x > 0]$. It is clear that activation functions provide an extension of cell selectors and in further discussion we will consider GMNNs with arbitrary activation functions.

The continuity of the overall network response is not guaranteed however, as the activation functions may still be truncated to zero at the cell boundaries. For some

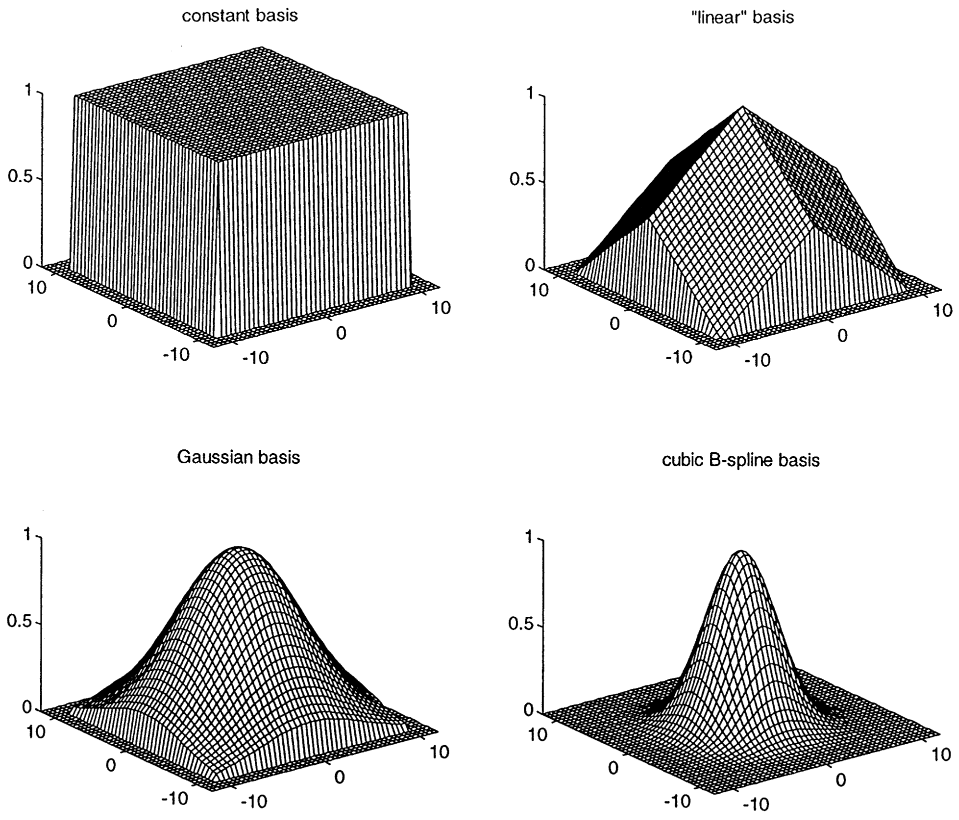


Fig. 4. Illustration of the truncation discontinuities occurring in basis functions with finite support (here the [− 10, 10]-by-[− 10,10] region); in the examples shown, only the cubic B-spline basis (the bottom-right surface) provides a smooth fall-off to zero at the cell boundaries.

quantization schemes, and notably those associated with B-spline networks [28], it is possible to define activation functions in such a way that they continuously approach zero at cell boundaries, but this is rather difficult in the general case [8,28]. Fig. 4 shows an example of four different basis functions and their boundary behaviour for the case of a square two-dimensional cell. In these examples, only the cubic B-spline basis guarantees mapping continuity.

When activation functions are taken into account, each of the addressed weights will be modulated according to the value of the associated activation function for the current network input. Thus, the output response will be given by

$$g(\boldsymbol{x}) = \sum_{k=1}^{K} w_k(\boldsymbol{x})\mu_k(\boldsymbol{x}) \quad \text{or} \quad g(\boldsymbol{x}) = \frac{\sum_{k=1}^{K} w_k(\boldsymbol{x})\mu_k(\boldsymbol{x})}{\sum_{k=1}^{K} \mu_k(\boldsymbol{x})} \tag{6}$$

in the standard and normalized variants, respectively, with $\mu_k(\boldsymbol{x})$ denoting the activation function corresponding to $\mathscr{R}_k(\boldsymbol{x})$. The right-hand-side formula of (6) represents a normalization of the network output with respect to the activation functions.

4.5. *Reproduction vectors*

So far our description of nodal quantizers has been associated with selecting a set of K memory addresses for the network input. In the strict sense, a vector quantizer should assign a unique reproduction vector to each quantization cell [17]. If “flat” activation functions are employed, the use of reproduction vectors is not immediately obvious, as the network response is generated solely on the basis of the addressed memory contents. On the other hand, the introduction of “graded” (i.e., localized within their support) activation functions makes it natural to use reproduction vectors as basis-function centres.

5. Address distance and proximity functions

The introduction of “graded” activation functions creates a direct analogy between the GMNN and basis function architectures, such as the RBF and KR networks. However, the very nature of the GMNN mapping leads to an implicit concept of a basis function associated with this architecture, even when the GMNN has a piecewise-constant response over Ω. In this section we introduce several important auxiliary functions, which facilitate such an interpretation of the GMNN mapping.

Let us assume for now that the activation functions are equivalent to the cell selectors. The response of the GMNN for two distinct points $\boldsymbol{x}$ and $\boldsymbol{y}$ will be related as long as they share at least one nodal-level quantization cell. The similarity between the responses, $g(\boldsymbol{x})$ and $g(\boldsymbol{y})$, can be expected to grow with the number of shared quantization cells, with identical responses obtained when the number of shared regions is equal to K – that is, when the two points fall into the same global quantization cell. Therefore, the similarity between the GMNN responses generated for two distinct points can be quantified (to a certain extent) by the number of shared nodal quantization cells, and conversely, the dissimilarity of responses can be expressed by the number of different nodal quantization cells generated for $\mathbf{x}$ and $\mathbf{y}$.

Since each quantization cell is uniquely associated with a memory address, the similarity (or dissimilarity) measures, expressed in terms of the number of shared (or distinct) quantization cells, can also be defined in terms of the number of shared (distinct) memory addresses. Similarity and distance measures will be defined as address proximity and distance functions, respectively. It should be noted, however, that the address proximity can express only a *potential* similarity between network responses, as it takes into account neither the contents of memory locations (i.e., their values) nor their format. Below we present formal definitions of the address distance and proximity functions and their properties.

The address distance, $\rho(\mathbf{x},\mathbf{y})$, between any two input points $\mathbf{x},\mathbf{y} \in \Omega$ is given by the number of distinct memory addresses generated for $\mathbf{x}$ and $\mathbf{y}$ by the GMNN, i.e.,

$$\rho : \Omega^2 \to \{0,1,\dots,K\} \quad \rho(\mathbf{x},\mathbf{y}) = \sum_{k=1}^{K} [A_k(\mathbf{x}) \neq A_k(\mathbf{y})],$$

whereas the address proximity, $\kappa(\mathbf{x},\mathbf{y})$, between $\mathbf{x}$ and $\mathbf{y}$ is given by the number of nodal addresses they share

$$\kappa : \Omega^2 \to \{0,1,\dots,K\}, \quad \kappa(\mathbf{x},\mathbf{y}) = \sum_{k=1}^{K} [A_k(\mathbf{x}) = A_k(\mathbf{y})].$$

The function $\rho(\mathbf{x},\mathbf{y})$ represents a generalized Hamming distance between the address vectors $A(\mathbf{x})$ and $A(\mathbf{y})$, whereas $\kappa(\mathbf{x},\mathbf{y})$ represents its complement. It follows that

$$\forall(\mathbf{x},\mathbf{y} \in \Omega)\ \rho(\mathbf{x},\mathbf{x}) = 0 \quad \text{and} \quad \rho(\mathbf{x},\mathbf{y}) = \rho(\mathbf{y},\mathbf{x}),$$

$$\forall(\mathbf{x},\mathbf{y} \in \Omega)\ \kappa(\mathbf{x},\mathbf{x}) = K \quad \text{and} \quad \kappa(\mathbf{x},\mathbf{y}) = \kappa(\mathbf{y},\mathbf{x}),$$

$$\forall(\mathbf{x},\mathbf{y} \in \Omega)\ \kappa(\mathbf{x},\mathbf{y}) = K - \rho(\mathbf{x},\mathbf{y}).$$

The address distance and proximity functions quantify the topological properties of the GMNN mapping. In particular, it is desirable that points lying close to each other (according to some input metric, d) in the input space should have a high address overlap (i.e., high address proximity), whereas points that are distant in Ω should share few (if any) nodal addresses (i.e., high address distance). Therefore, one possible requirement on the quantization part of a GMNN is concerned with ensuring that, on average, the address distance between input-space points is monotonic with their input-space distance, i.e.,

$$d(\mathbf{x},\mathbf{y}) \geq d(\mathbf{x},\mathbf{z}) \Rightarrow E[\rho(\mathbf{x},\mathbf{y})] \geq E[\rho(\mathbf{x},\mathbf{z})] \quad \text{for any } \mathbf{x},\mathbf{y},\mathbf{z} \in \Omega.$$

For any input point, $\mathbf{x}$, we define its neighbourhood $\mathcal{N}(\mathbf{x})$ (according to the GMNN mapping) as the set of points in Ω that share at least one nodal address with $\mathbf{x}$:

$$\mathcal{N}(\mathbf{x}) = \{\mathbf{y} \in \Omega \colon \exists(k \in \{1,\dots,K\}) A_k(\mathbf{x}) = A_k(\mathbf{y})\}. \tag{7}$$

Alternative ways of specifying $\mathcal{N}(\mathbf{x})$ include

$$\mathcal{N}(\mathbf{x}) = \{\mathbf{y} \in \Omega \colon \rho(\mathbf{x},\mathbf{y}) < K\} = \{\mathbf{y} \in \Omega \colon \kappa(\mathbf{x},\mathbf{y}) > 0\}.$$

Thus only the training-set points falling into $\mathcal{N}(\boldsymbol{x})$ can influence the network response at $\boldsymbol{x}$.

It is convenient to analyse the properties of $\rho(\boldsymbol{x},\boldsymbol{y})$ and $\kappa(\boldsymbol{x},\boldsymbol{y})$ by introducing a nodal incidence function $M_k(\boldsymbol{x},\boldsymbol{y})$, which is equal to 1 if the points $\boldsymbol{x}$ and $\boldsymbol{y}$ share the same address at node k, and is equal to 0 otherwise, i.e.,

$$M_k(\boldsymbol{x},\boldsymbol{y}) = [A_k(\boldsymbol{x}) = A_k(\boldsymbol{y})], \quad k = 1,\dots,K.$$

Using this definition, the address distance and proximity functions can be expressed as

$$\sum_{k=1}^{K} M_k(\boldsymbol{x},\boldsymbol{y}) = \kappa(\boldsymbol{x},\boldsymbol{y}) = K - \rho(\boldsymbol{x},\boldsymbol{y}). \tag{8}$$

The incidence function can also be expressed in terms of the cell selectors as

$$M_k(\boldsymbol{x},\boldsymbol{y}) = S_k(\boldsymbol{x})S_k(\boldsymbol{y}). \tag{9}$$

6. Alternative basis-function interpretations of the GMNN mapping

6.1. Basis function networks and their analogies with GMNN

Two related groups of neural network architectures can be described to have a basis-function form. These are radial basis function (RBF) and kernel regression (KR) networks. RBF networks have long been applied to (multivariate) function approximation (and interpolation) problems [33,34] and have been widely adopted by the neural network community [12,21]. An RBF network builds its response as superposition of basis functions

$$g(\boldsymbol{x}) = \sum_{i=1}^{N} w_i \varphi_i(\boldsymbol{x} - \boldsymbol{c}^i), \tag{10}$$

where usually $\varphi_i(\boldsymbol{x} - \boldsymbol{c}^i) = \varphi_i(\|\boldsymbol{x} - \boldsymbol{c}^i\|)$, $\{\boldsymbol{c}^i\}_1^N$ represent basis-function centres and $\{w_i\}_1^N$ are the adjustable weights. In some applications a normalized version of an RBF network is employed [29], where the response has a form of

$$g(\boldsymbol{x}) = \frac{\sum_{i=1}^{N} w_i \varphi_i(\boldsymbol{x} - \boldsymbol{c}^i)}{\sum_{i=1}^{N} \varphi_i(\boldsymbol{x} - \boldsymbol{c}^i)}. \tag{11}$$

A kernel regression network [20] (also known as Nadaraya–Watson kernel regression estimator [30,41] and a general regression neural network [37]) has its roots in the kernel method of nonparametric probability density estimation [19,35,36]. Given a set of kernel functions $\{\varphi_t\}_1^T$ (and their associated bandwidth parameters) the response of a KR network is given by

$$g(\boldsymbol{x}) = \frac{\sum_{t=1}^{T} y^t \varphi_t(\boldsymbol{x} - \boldsymbol{x}^t)}{\sum_{t=1}^{T} \varphi_t(\boldsymbol{x} - \boldsymbol{x}^t)} \tag{12}$$

that is, it is fully defined by the training set $\{\boldsymbol{x}^t, y^t\}_1^T$, unlike in the case of an RBF network where the weights $\{w_i\}_1^N$ have to be determined. This apparent simplicity of network "training" is one of the main advantages of a KR network in practical applications.

Given the above, and by taking the discussed forms of a GMNN response into account, it is clear that the GMNN can be considered to have a basis-function form, with the basis functions given by the activation functions (6). Under this interpretation, the total number of basis functions is equal to the combined number of nodal quantization cells $\sum_{k=1}^{K}|\mathscr{A}_k|$, with only a portion of them having their "heights" defined by the training data (it can be expected that the training points will fall only into a fraction of the available quantization cells). In the special case of "flat" activation functions, the bases have a box-window form, analogous to the histogram regression estimator, which results in a piecewise-constant response of the network (Eqs. (4) and (5)).

It is, however, possible to obtain another interpretation of a GMNN response, where the basis functions are determined by the training data (i.e., there is a one-to-one correspondence between the training points and the basis functions). As shown in [24], for the special case of "flat" activation functions, the response of a GMNN trained with a general supervised error-correcting rule can be expressed as $g(\boldsymbol{x}) = \sum_{t=1}^{T} \Delta^t \kappa(\boldsymbol{x}, \boldsymbol{x}^t)$, with the address proximity functions $\{\kappa(\boldsymbol{x}, \boldsymbol{x}^t)\}_{t=1}^T$ playing the role of basis functions and the parameters $\{\Delta^t\}_{t=1}^T$ determined by the learning procedure. A modification to the structure of memory locations leads to responses of the forms $g(\boldsymbol{x}) = \sum_{t=1}^{T} \Delta^t \kappa(\boldsymbol{x}, \boldsymbol{x}^t) / \sum_{t=1}^{T} \kappa(\boldsymbol{x}, \boldsymbol{x}^t)$ and $g(\boldsymbol{x}) = \sum_{t=1}^{T} y^t \kappa(\boldsymbol{x}, \boldsymbol{x}^t) / \sum_{t=1}^{T} \kappa(\boldsymbol{x}, \boldsymbol{x}^t)$, with the latter being analogous to that of a KR network. In the remainder of this section we extend our results presented in [24], and in particular, we consider the general case of arbitrary cell-activation functions.

6.2. The GMNN as a kernel regression network

To introduce the alternative basis-function interpretation of a GMNN mapping, we first concentrate on the KR-like form of the GMNN output, given by

$$g(\boldsymbol{x}) = \frac{\sum_{t=1}^{T} y^t \kappa(\boldsymbol{x}, \boldsymbol{x}^t)}{\sum_{t=1}^{T} \kappa(\boldsymbol{x}, \boldsymbol{x}^t)}.$$

This form of response can be realized by a GMNN with "flat" activation functions when the format of network memory locations is modified such that every numeric weight is augmented by an integer "counter" (i.e., every memory location consists now of two fields: a weight and a counter) that is incremented by 1 every time the location is accessed by a *new* training pair during the learning process, thus recording the number of distinct training points that addressed the corresponding memory location.

The network response is redefined as the sum of the addressed weights over the sum of the addressed counters

$$g(\boldsymbol{x}) = \frac{\sum_{k=1}^{K} w_k(\boldsymbol{x})}{\sum_{k=1}^{K} c_k(\boldsymbol{x})}$$

and the following learning algorithm is used to set the counter and weight memory fields:

Each training pair $(\boldsymbol{x}^t, y^t) \in \mathscr{T}$ is presented to the network exactly once, where the weights addressed by $\boldsymbol{x}^t$ are incremented by y^t, while the addressed counters are incremented by 1, i.e.,

$$w_k(\boldsymbol{x}^t) \leftarrow w_k(\boldsymbol{x}^t) + y^t, \quad k = 1, \dots, K,$$

$$c_k(\boldsymbol{x}^t) \leftarrow c_k(\boldsymbol{x}^t) + 1, \quad t = 1, \dots, T$$

and, initially, all network memory locations are set to zero. After the training is complete, every network weight is equal to the sum of training-set values of points that fell into its corresponding cell, whereas the associated counter is equal to their number. The output response can be expressed as (see (8))

$$g(\boldsymbol{x}) = \frac{\sum_{k=1}^{K} w_k(\boldsymbol{x})}{\sum_{k=1}^{K} c_k(\boldsymbol{x})} = \frac{\sum_{k=1}^{K} \sum_{t=1}^{T} y^t M_k(\boldsymbol{x}, \boldsymbol{x}^t)}{\sum_{k=1}^{K} \sum_{t=1}^{T} M_k(\boldsymbol{x}, \boldsymbol{x}^t)}$$
$$= \frac{\sum_{t=1}^{T} y^t \kappa(\boldsymbol{x}, \boldsymbol{x}^t)}{\sum_{t=1}^{T} \kappa(\boldsymbol{x}, \boldsymbol{x}^t)} \tag{13}$$

which corresponds to the output of a KR network, with the kernel functions given by the proximity functions. The kernel-regression form of a GMNN response is in fact equivalent to a variant of the histogram-regression estimate of f given by Eq. (3). Note that formula (13) can be expressed as

$$g(\boldsymbol{x}) = \frac{\sum_{k=1}^{K} (w_k(\boldsymbol{x})/c_k(\boldsymbol{x})) c_k(\boldsymbol{x})}{\sum_{k=1}^{K} c_k(\boldsymbol{x})},$$

where

$$\frac{w_k(\boldsymbol{x})}{c_k(\boldsymbol{x})} = \hat{y}(\mathscr{R}_k(\boldsymbol{x})) = \frac{\sum_{t=1}^{T} y^t [\boldsymbol{x}^t \in \mathscr{R}_k(\boldsymbol{x})]}{\sum_{t=1}^{T} [\boldsymbol{x}^t \in \mathscr{R}_k(\boldsymbol{x})]}.$$

Hence, $\hat{y}(\mathscr{R}_k(\boldsymbol{x}))$ represents the average value of training points falling into $\mathscr{R}_k(\boldsymbol{x})$ and thus a (single) histogram regression estimate of f for points in that region. The global network response, on the other hand, weighs the contributions of individual estimates according to the coverage of the corresponding quantization regions by the training-set points, where regions covered more densely have higher priority, i.e.,

$$g(\boldsymbol{x}) = \frac{\sum_{k=1}^{K} \hat{y}(\mathscr{R}_k(\boldsymbol{x})) c_k(\boldsymbol{x})}{\sum_{k=1}^{K} c_k(\boldsymbol{x})}.$$

In contrast, a simple average of histogram estimates (1) assigns equal ranks to all participating regions.

In the case where activation functions are "graded", the above learning scheme is modified whereby at each presentation of a new training point the increment values of the addressed weights and counters are additionally modulated by the activation functions, i.e.,

$$\begin{aligned} w_k(\boldsymbol{x}^t) &\leftarrow w_k(\boldsymbol{x}^t) + y^t \mu_k(\boldsymbol{x}^t), & k &= 1,\dots,K,\\ c_k(\boldsymbol{x}^t) &\leftarrow c_k(\boldsymbol{x}^t) + \mu_k(\boldsymbol{x}^t), & t &= 1,\dots,T. \end{aligned}$$

The output response of the network is now defined as

$$g(\boldsymbol{x}) = \frac{\sum_{k=1}^{K} w_k(\boldsymbol{x})\mu_k(\boldsymbol{x}^t)}{\sum_{k=1}^{K} c_k(\boldsymbol{x})\mu_k(\boldsymbol{x}^t)}.$$

Consequently, the network response can be expressed as

$$\begin{aligned} g(\boldsymbol{x}) &= \frac{\sum_{k=1}^{K} w_k(\boldsymbol{x})\mu_k(\boldsymbol{x}^t)}{\sum_{k=1}^{K} c_k(\boldsymbol{x})\mu_k(\boldsymbol{x}^t)} = \frac{\sum_{t=1}^{T} y^t \sum_{k=1}^{K} \mu_k(\boldsymbol{x}^t)\mu_k(\boldsymbol{x})}{\sum_{t=1}^{T}\sum_{k=1}^{K} \mu_k(\boldsymbol{x}^t)\mu_k(\boldsymbol{x})}\\ &= \frac{\sum_{t=1}^{T} y^t \kappa_\mu(\boldsymbol{x},\boldsymbol{x}^t)}{\sum_{t=1}^{T} \kappa_\mu(\boldsymbol{x},\boldsymbol{x}^t)}, \end{aligned}$$

where

$$\kappa_\mu(\boldsymbol{x},\boldsymbol{x}^t) = \sum_{k=1}^{K} \mu_k(\boldsymbol{x})\mu_k(\boldsymbol{x}^t).$$

The "extended" kernel function κ_μ has the same support as κ, which is apparent when we consider that (Eqs. (8) and (9))

$$\kappa(\boldsymbol{x},\boldsymbol{y}) = \sum_{k=1}^{K} M_k(\boldsymbol{x},\boldsymbol{y}) = \sum_{k=1}^{K} S_k(\boldsymbol{x})S_k(\boldsymbol{y}), \tag{14}$$

and

$$\kappa_\mu(\boldsymbol{x},\boldsymbol{y}) = \sum_{k=1}^{K} \mu_k(\boldsymbol{x})\mu_k(\boldsymbol{y}) \tag{15}$$

and that $S_k(\boldsymbol{x})$ and $\mu_k(\boldsymbol{x})$ have identical supports. However, the values of κ_μ depend both on the number of common addresses generated for its two arguments, as well as on the values of the activation functions at these two points. Hence, if the activation functions are continuous (but non-constant) within their regions of support, the resulting form of κ_μ will be piecewise continuous, with step discontinuities occurring at the quantization-cell boundaries.

6.3. A simplified version of the kernel regression network

The variant of the GMNN allowing normalization of the network response with respect to the kernel functions, κ_μ and κ, requires the presence of counter fields in the network memory locations, thus leading to increased memory requirements in practical applications. Following the histogram example outlined in the introduction, it is possible to form a simplified variant of the KR network (as realized by the GMNN) where the counter fields are no longer necessary.

Let us consider a network response where the output is normalized with respect to the activation functions (6). Using the cell activators as kernel functions, it is possible to set each memory weight to the KR estimate of f for points inside the corresponding cell (i.e., its "centre"),

$$w_k^i = \hat{y}(\mathscr{R}_k^i) = \frac{\sum_{t=1}^{T} y^t \mu_k^i(\boldsymbol{x}^t)}{\sum_{t=1}^{T} \mu_k^i(\boldsymbol{x}^t)}.$$

Therefore, the response of the network (according to Eq. (6)) to a new input, $\boldsymbol{x}$, is now given by

$$g(\boldsymbol{x}) = \frac{\sum_{k=1}^{K} w_k(\boldsymbol{x})\mu_k(\boldsymbol{x})}{\sum_{k=1}^{K} \mu_k(\boldsymbol{x})} = \frac{\sum_{k=1}^{K} \hat{y}(\mathscr{R}_k(\boldsymbol{x}))\mu_k(\boldsymbol{x})}{\sum_{k=1}^{K} \mu_k(\boldsymbol{x})}. \tag{16}$$

It can be seen that Eq. (16) directly corresponds to the histogram estimate of Eq. (1), and the formulas become identical when $\mu_k(\boldsymbol{x}) \equiv S_k(\boldsymbol{x})$ $(k = 1, \ldots, K)$. Of course, a natural choice of an activation function is that of one attaining its maximum at the centre of the associated quantization cell, and monotonically decreasing for points farther away from the centre (e.g., a Gaussian). In this case $\hat{y}(\mathscr{R}_k^i)$ can be interpreted as a KR estimate of f at the centre, $\boldsymbol{c}_k^i$, of $\mathscr{R}_k^i$ (i.e., $\hat{y}(\mathscr{R}_k^i) = \hat{y}(\boldsymbol{c}_k^i)$), and the network output given by Eq. (16) is equivalent to

$$g(\boldsymbol{x}) = \frac{\sum_{k=1}^{K} \hat{y}(\boldsymbol{c}_k(\boldsymbol{x}))\mu_k(\boldsymbol{x})}{\sum_{k=1}^{K} \mu_k(\boldsymbol{x})}.$$

6.4. The GMNN as a radial basis function network

Let us consider the LMS learning scheme [43], in which the training-set points are considered iteratively, and at each step of the algorithm the instantaneous squared error between the network output, $g(\boldsymbol{x}^t)$, and the desired value, y^t, is minimized. A procedure of the LMS type can be applied, since the only adaptable network parameters are the memory weights, which contribute linearly to network output. During every step of the iterative training procedure, each of the participating K weights is adjusted along the direction of the gradient of the error function. For clarity of presentation, let us assume that the network response is given by $g(\boldsymbol{x}) = \sum_{k=1}^{K} w_k(\boldsymbol{x})\mu_k(\boldsymbol{x})$, but the following discussion is equally valid for the normalized case (see Eq. (6)). The instantaneous squared error and its gradient with respect to the kth participating weight are given by

$$\varepsilon^2 = (y - g(\boldsymbol{x}))^2 = \left(y - \sum_{k=1}^{K} w_k(\boldsymbol{x})\mu_k(\boldsymbol{x}) \right)^2,$$

$$\frac{\partial \varepsilon^2}{\partial w_k} = -\, 2\varepsilon \mu_k(\boldsymbol{x}),$$

respectively. Hence, each of the participating weights is updated by a common value $(\sim \varepsilon)$ modulated by the corresponding activation function. If the update operation occurs for the ith presentation of the tth training pair, the weight modifications have the form

$$w_k(\boldsymbol{x}^t) \leftarrow w_k(\boldsymbol{x}^t) + \Delta_i^t \mu_k(\boldsymbol{x}^t), \quad k = 1, \dots, K,$$

where Δ_i^t represents the common term for all weights. For the standard version of the LMS algorithm $\Delta_i^t = \eta(y^t - g(\boldsymbol{x}))$, where η is the learning rate [43]. The training proceeds in this manner until a stopping criterion is reached, at which point the tth training pair has been seen by the network T_t times (we assume that $T_t > 0$ for $t = 1, \dots, T$).

After the training is complete, the value of a weight w_j^k (corresponding to the cell $\mathscr{R}_j^k$ at the kth quantization layer) will depend only on the training-set points that fell into $\mathscr{R}_j^k$, i.e.,

$$w_j^k = \sum_{t=1}^{T} \sum_{i=1}^{T_t} \Delta_i^t \mu_j^k(\boldsymbol{x}^t) = \sum_{t=1}^{T} \Delta^t \mu_j^k(\boldsymbol{x}^t),$$

where Δ^t denotes the accumulated common weight update corresponding to the tth training sample. Consequently, the response of the network to a new excitation is given by

$$\begin{aligned} g(\boldsymbol{x}) &= \sum_{k=1}^{K} w_k(\boldsymbol{x})\mu_k(\boldsymbol{x}) = \sum_{k=1}^{K} \mu_k(\boldsymbol{x}) \sum_{t=1}^{T} \Delta^t \mu_k(\boldsymbol{x}^t) = \sum_{t=1}^{T} \Delta^t \sum_{k=1}^{K} \mu_k(\boldsymbol{x})\mu_k(\boldsymbol{x}^t) \\ &= \sum_{t=1}^{T} \Delta^t \kappa_\mu(\boldsymbol{x}, \boldsymbol{x}^t), \end{aligned} \tag{17}$$

where the kernel function κ_μ, is defined by Eq. (15). So, analogous to the kernel-regression case, the response of a trained GMNN can be considered equivalent to the response of an RBF network, with the bases given by the extended address proximity functions, $\{\kappa_\mu(\boldsymbol{x}, \boldsymbol{x}^t)\}_{t=1}^{T}$. Of course, unlike most RBF cases, the basis functions are position dependent and may have step discontinuities.

To achieve a normalization of the network response with respect to the basis set $\{\kappa_\mu(\boldsymbol{x}, \boldsymbol{x}^t)\}_{t=1}^{T}$, we augment each of the memory locations by a counter field, as discussed in the context of the KR form of the GMNN response (in the previous section). To set the counter fields just one pass through the training set is required, whereas the subsequent passes are used to adjust memory weights only. In the case of a normalized response, the output of a trained network to an excitation, $\boldsymbol{x}$, is given by

$$\begin{aligned} g(\boldsymbol{x}) &= \frac{\sum_{k=1}^{K} w_k(\boldsymbol{x})\mu_k(\boldsymbol{x})}{\sum_{k=1}^{K} c_k(\boldsymbol{x})\mu_k(\boldsymbol{x})} = \frac{\sum_{t=1}^{T} \Delta^t \sum_{k=1}^{K} \mu_k(\boldsymbol{x})\mu_k(\boldsymbol{x}^t)}{\sum_{t=1}^{T} \sum_{k=1}^{K} \mu_k(\boldsymbol{x})\mu_k(\boldsymbol{x}^t)} \\ &= \frac{\sum_{t=1}^{T} \Delta^t \kappa_\mu(\boldsymbol{x}, \boldsymbol{x}^t)}{\sum_{t=1}^{T} \kappa_\mu(\boldsymbol{x}, \boldsymbol{x}^t)}. \end{aligned}$$

6.5. Discussion

We have shown that it is possible to associate an *implicit* form of a basis function with the GMNN architecture, which leads to close analogies between the response of a trained GMNN and the RBF and KR networks. The identified basis functions (given by $\{\kappa_\mu(\boldsymbol{x},\boldsymbol{x}^t)\}_1^T$) of the GMNN are generally position dependent with possible step discontinuities within their (finite) support. As such, they differ from the usual choice of smooth, position independent (and often "radially" symmetric) functions used typically in applications of RBF and KR networks. However, as shown in [31], functions of this form satisfy the very mild constraints placed on the types of nonlinearities suitable for a basis in RBF networks. After normalization, they can also represent valid pdf functions and satisfy the (also mild) constraints placed on kernels in KR networks [19].

The position dependence of basis functions in a GMNN can be exploited to vary their bandwidth according to local properties of the training data. This is accomplished indirectly by locally controlling the resolution of GMNN quantizers. On the other hand, we suggest that if the regularity of quantization employed by a GMNN is high, one might attempt to derive an expected (average) form of $\kappa_\mu(\boldsymbol{x},\boldsymbol{y})$, which could further be used to analyse (assess or predict) network performance based on the performance of its RBF and KR models constructed using the expected form of $\kappa_\mu(\boldsymbol{x},\boldsymbol{y})$ as the basis. This approach has indeed been successfully applied in the case of the N-tuple network [25].

One advantage of the implicit realization of the basis set, $\{\kappa_\mu(\boldsymbol{x},\boldsymbol{x}^t)\}_1^T$, is that the functions do not have to be evaluated explicitly during network operation, which leads to the network-response time being independent of the training-set size used. As a result, when a moderately small number of memory nodes, K, is used, the response time of a GMNN can be very fast, as observed in practical applications [39]. On the other hand, since the basis functions are not accessible, it is not possible to apply direct linear algebra techniques to solve the approximation problem, as is the case in standard applications of RBF networks.

Two basis-function interpretations of a GMNN mapping have been presented. According to the first one, the basis set is equivalent to the cell-activation functions of nodal quantization cells, with exactly K bases being active for any $\boldsymbol{x} \in \Omega$. The total number of basis functions is here directly determined by the combined number of cells in the nodal quantization layers, although only those bases are actually defined (i.e., are not identically equal to zero) whose corresponding cells are visited by the training points. This interpretation stresses the locality of a GMNN response, which is always dependent only on those basis functions whose support regions overlap with a certain local neighbourhood, $\mathcal{N}(\boldsymbol{x})$ (7), of the current evaluation point.

On the other hand, the alternative interpretation of the GMNN mapping assigns one basis function to every training-set point, and thus emphasises the analogy between the GMNN architecture and the RBF and KR networks. Furthermore, the identified basis functions have a "graded" form, even in the case when the GMNN exhibits a piecewise-constant response over Ω. This latter property provides a justification for using "flat" activation functions in a GMNN and provides a means for analysis of the network's properties in this important case.

7. Examples of the GMNN architecture

Here we present a short description of three representative GMNN variants, focusing on the quantization schemes they employ where, interestingly, all three networks use a form of scalar-product vector quantization, which results in cells of a hyper-rectangular shape. Consequently, we begin with outlining the basic principles of creating the quantization layers of a GMNN when scalar-product vector quantization is involved.

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

This is an important case, since in scalar-product quantization every component of an input vector is quantized individually. Let m denote the number of scalar quantizers employed (as will be explained later, K does not always have to be equal to m when $D > 1$). Each of them divides the range of the input variable into a sequence of consecutive intervals delimited by uniformly (for clarity) placed “threshold” points.

According to the GMNN definition, there must be K (m in this case) nodal quantization cells overlapping at any input point, with no two of the overlapping cells being identical however. One simple way to satisfy this requirement is to impose

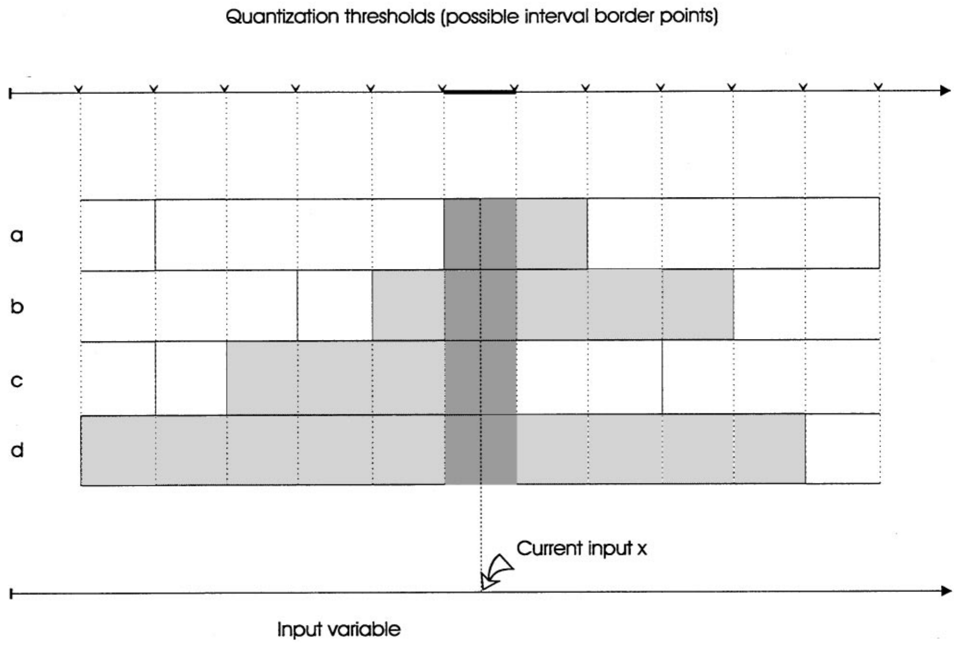


Fig. 5. Example of GMNN quantization of a scalar integer variable for the case of $m = 4$ (individual quantization layers are labelled as: a, b, c, d). A border between quantization intervals at a given threshold position is allowed for one layer only (see text for details). When the value of the input variable is incremented or decremented by one unit, *at most one* quantizer will change its output.

a condition by which any threshold point defines a border between quantization intervals only at one of the m quantization layers. If no further restrictions are placed, the cells may vary according to their length, both within one quantization layer and when any two layers are considered (see Fig. 5). This type of scalar quantization is typically employed by the N-tuple network.

A more regular quantization structure results if we further demand that every threshold becomes a border point for one of the m quantization layers (in this way the maximum possible effective quantization resolution is achieved). A particular form of this scheme results when every nodal quantization interval extends for m units, so every mth threshold belongs to the same quantization layer. Visually, the quantization-cell arrangements are staggered (shifted by one unit with respect to each other), as shown in Fig. 6. This form of scalar quantization is typically employed by CMAC and B-spline networks.

7.2. *The multidimensional case* ($D > 1$)

According to the principles of the GMNN mapping no two nodal quantization cells can be identical. By taking the scalar-product nature of quantization into account [17] it is sufficient to say that for any two cells, at least one of their D projections on the coordinate axes should differ. A D-dimensional hyper-rectangular cell is here

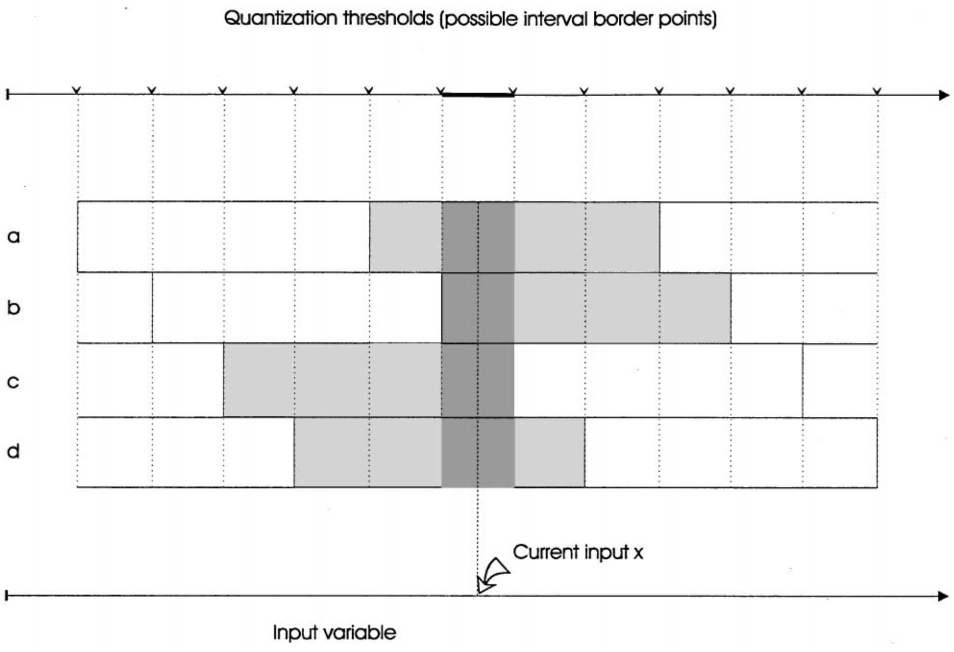


Fig. 6. Example of GMNN quantization of a scalar integer variable for the case of $m = 4$. The overlapping (staggered) arrangement of the K quantizers (labelled: a, b, c, d) is indicated. When the value of the input variable is incremented or decremented by one unit, *only one* quantizer will change its output.

directly determined by specifying D intervals corresponding to its projections on the coordinate axes, where each projection corresponds to a scalar quantization cell. Let m be the number of scalar quantizers in each dimension. It follows that there are m distinct scalar intervals selected for an input point at each vector dimension, and consequently there can be at most m^D distinct D-dimensional cells, any two of which differ at least along one scalar projection. Hence, a GMNN network with $K = m^D$ nodes can be created by using m scalar quantizers along each input dimension. This approach is taken by the B-spline network.

Alternatively, one might require that the projections of the D-dimensional cells should differ along every input dimension. In this approach (typically taken by the CMAC and N-tuple networks), if K scalar quantizers are employed along each input dimension then the total number of network nodes is also equal to K.

7.3. The cerebellar model articulation controller (CMAC) network

CMAC was introduced by Albus as a functional model of mammalian cerebellum [1–4]. Since then it has found many applications, mainly in the fields of robotics and control [27], where the high speed of operation (and learning) and ease of implementation offered by this network are especially desirable. In most cases the network is implemented with a piecewise-constant response, although the use of B-splines as cell-activation functions has also been suggested [26].

The quantization scheme presented above (i.e., with the staggered arrangement of K scalar quantizers along each dimension, and with K D-dimensional cells selected for any input) corresponds roughly to the one proposed originally by Albus [2]. Many variations of this scheme exist, mainly corresponding to the ordering of the staggered scalar quantizers for different input dimensions. Fig. 7 shows an example of the quantization-cell arrangement of a CMAC network for the case of $D = 2$ and $K = 4$. Quantization lattices and individual cells corresponding to the respective network nodes are shown in different shades of grey.

7.4. The N-tuple network

The N-tuple network was introduced by Bledsoe and Browning [9,10] and initially applied to pattern recognition applications [44]. Further investigations of this architecture led to many extensions [5,7] (e.g., approximation of continuous mappings [38]) and also to commercial applications [6].

In its traditional form, the network consists of an R-bit binary “retina” array, onto which an input vector is transformed. Each of the K network memory nodes uses an N-bit long address word, which is determined by connecting individual N address lines to retina bits.

In the case where network input is provided by a D-dimensional real vector, the quantization (at a single-node level) can also be performed by distributing N scalar quantization thresholds within the ranges of the D scalar variables. An nth bit of an address word is then set to ‘1’ or ‘0’, depending on whether the current value of an input variable crosses the threshold or not (see Fig. 8). Usually, the placement of these

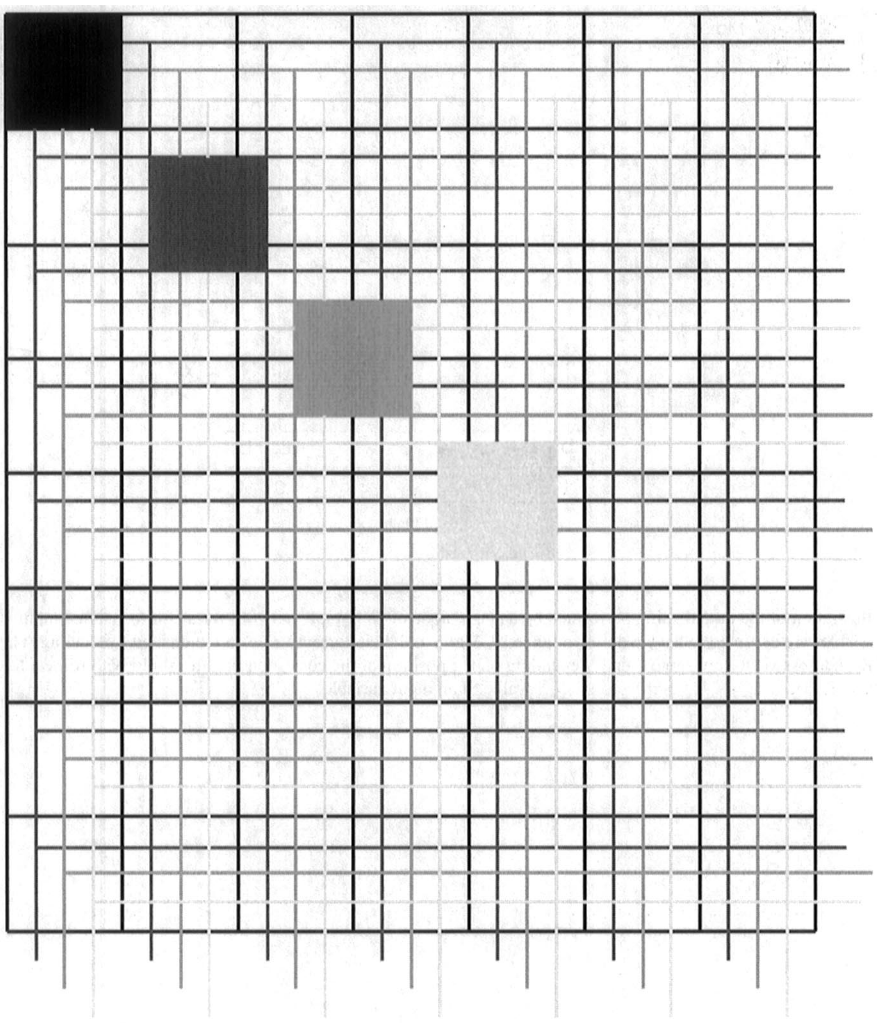


Fig. 7. Illustration of the CMAC quantization scheme for the two-dimensional case, with $K = 4$. The overlapping square quantization lattices, corresponding to the individual network nodes, are drawn in different shades of grey for clarity.

thresholds is performed at random for each network node (placement is fixed for network's life). As a result, the scalar quantization resolution varies between the individual input dimensions.

7.5. The B-spline network

B-splines have been widely used in computer graphics with applications to curve and surface fitting [8]. They possess many desirable properties and have been

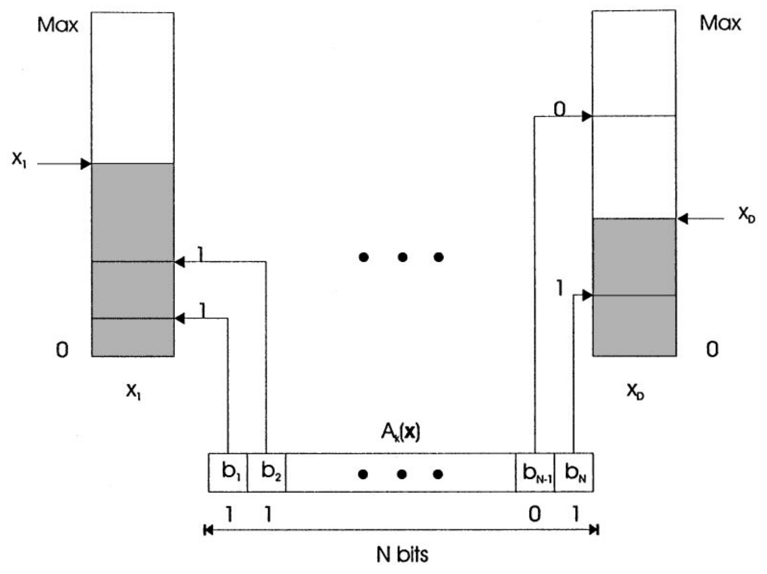


Fig. 8. Illustration of the quantization performed by a single node of an N-tuple network for a D-dimensional input vector, $\boldsymbol{x}$. All input variables are assumed to have a uniform range [0,Max], and their current values are indicated by shading. The individual bits of the address vector are set to 0 or 1, depending on the position of the corresponding threshold with respect to the current value of an input variable.

extensively investigated both from practical and theoretical points of view [14,40].

A scalar B-spline function of order m (usually in the range 2,3,4) represents a piecewise-polynomial C^{m-1} continuous function supported by a finite interval. A scalar B-spline network is defined by a number of knots, which determine the support regions of basis functions, where the control points are often chosen to be equally spaced. Exactly m functions are non-zero at any point, and every B-spline function is supported by m inter-knot intervals. Consequently, the input domain is quantized by m staggered overlapping quantizers, as illustrated in Fig. 9.

In a D-dimensional case, B-spline functions have a tensor-product form and there are m^D non-zero basis functions at every point in Ω. In an extension of the scalar case, Ω is thus quantized by m^D hyper-rectangular regular grids. It was noted in [26] that the quantization performed by the CMAC network can be considered to be a subset of the "full" quantization present in a corresponding B-spline network. The similarity between tensor-product B-spline networks and networks of the CMAC type has been observed by Moody [28].

A GMNN utilizing B-spline functions has $K = m^D$, where each node performs a regular quantization of Ω in the form of a hyper-rectangular grid. Unlike CMAC and N-tuple networks, the activation functions are defined here explicitly.

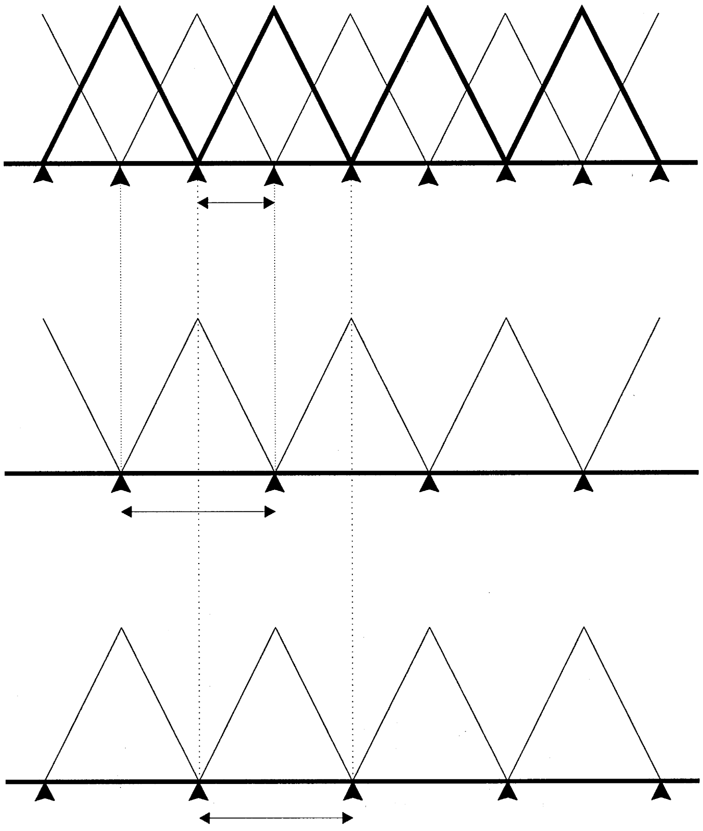


Fig. 9. Illustration of the overlapping quantizer structure induced by B-spline functions for the linear (order 2) case with uniform knot placement. As it can be seen, the regular arrangement of overlapping triangular kernels can be split into two basis-function sequences, where the individual kernels are adjacent to each other, in each case defining a uniform quantization of the input range. By combining the quantizers into a single B-spline network, the quantization resolution is doubled.

8. Conclusions

In this paper we considered the problem of approximation of a multivariate real function by a certain class of neural networks whose operation inherently depends on the quantization of the input space, $\Omega \subset \mathrm{R}^D$. It was shown that for this class of networks it is possible to reduce the detrimental effects associated with high-dimensional problems by performing multiple, nonidentical quantizations of Ω at low resolutions, rather than employing a single high-resolution quantization. Networks of this type can be visualized as consisting of several (K) parallel quantization layers, each associated with a separate memory unit. For any input point, $\boldsymbol{x} \in \Omega$, each network quantizer selects a single memory cell to which $\boldsymbol{x}$ belongs. The selected quantization cells provide, in turn, memory addresses within their respective memory nodes. The contents of these selected memory locations are combined to provide the overall network output.

As a result of the multiple-quantization approach, every point in Ω lies inside a region given by an intersection of K quantization cells corresponding to the K network nodes. However, since these cells are overlapping, but not identical, their intersection can be substantially smaller than any of the participating cells. In this way the effective resolution of the network can be relatively high, with the size of quantization cells in the individual (quantization) layers being large enough to ensure their adequate coverage by the training set.

Since networks of this type essentially consist of a single layer of memory cells (K of which are selected for any output computation) and an address-generating unit, a common architecture under the name of general memory neural network (GMNN) is suggested. There are several existing variants of the GMNN architecture, with the distinguishing features being the different input-space quantization schemes employed. The GMNN abstraction allows the analysis of properties common to these architectures without being bound by the particular structure of the address-generation units, or the organization of the memory nodes.

It has been shown that GMNN networks are capable of producing responses analogous to RBF and KR networks, where the basis (or kernel) functions are given by the GMNN address proximity functions. In particular, they can be realized implicitly, so the network performance is not impaired even when large training sets are used.

A variant of the GMNN was proposed that allows normalization of the response with respect to the basis functions. As a result, networks of the GMNN type can be made analogous to the normalized RBF and KR networks. To achieve this effect, every memory location has to consist of two numeric fields, instead of just one (i.e., the weight). However, a simplified version of the normalized network was also suggested that allows the implementation of a KR network without increasing memory requirements. This GMNN variant is closely related to the histogram regression estimate. Finally, several memory-based networks were presented, mainly from the perspective of their input-space quantization schemes, providing examples of the GMNN architecture. We hope that the concept of a GMNN presented in this paper will help to increase our understanding of networks conforming to this architecture and perhaps will also lead to a development of new GMNN variants.

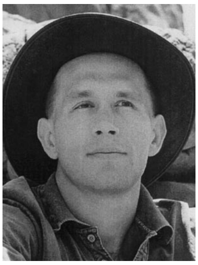
Certain aspects of the material presented in this paper are also discussed in [23].

References

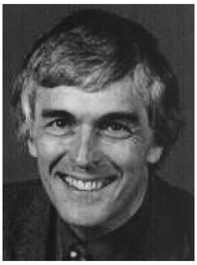
[1] J.S. Albus, A theory of cerebellar function, Math. Biosci. 10 (1971) 25–61.
[2] J.S. Albus, A new approach to manipulator control: the cerebellar model articulation controller (CMAC), J. Dynamic Systems Meas. Control 97 (3) (1975) 220–227.
[3] J.S. Albus, Data storage in the cerebellar model articulation controller (CMAC), J. Dynamic Systems Meas. Control 97 (3) (1975) 228–233.

[4] J.S. Albus, Mechanisms of planning and problem solving in the brain, Math. Biosci. 45 (1979) 247–293.
[5] I. Aleksander, T.J. Stonham, A guide to pattern recognition using random-access memories, IEE Proc. E Comput. Digital Techniq. 2 (1) (1979) 29–40.
[6] I. Aleksander, W. Thomas, P. Bowden, WISARD, a radical new step forward in image recognition, Sensor Rev. 4 (3) (1984) 120–124.
[7] I. Aleksander, M.J. Wilson, Adaptive windows for image processing, IEE Proc. E Comput. Digital Techniq. 132 (5) (1985) 233–245.
[8] R.H. Bartels, J.C. Beatty, B.A. Barsky, An Introduction to Splines for use in Computer Graphics and Geometric Modeling, Morgan Kaufmann, Los Altos, CA, 1987.
[9] W.W. Bledsoe, C.L. Bisson, Improved memory matrices for the N-tuple pattern recognition method, IRE Trans. Electron. Comput. EC-11 (1962) 414–415.
[10] W. Bledsoe, I. Browning, Pattern recognition and reading by machine, in IRE Joint Computer Conference, 1959, pp. 225–232.
[11] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth and Brooks/Cole, Monterey, CA, 1984.
[12] D.S. Broomhead, D. Lowe, Multivariable functional interpolation and adaptive networks, Complex Systems 2 (1988) 321–355.
[13] M. Brown, C.J. Harris, P.C. Parks, The interpolation capabilities of the binary CMAC, Neural Networks 6 (3) (1993) 429–440.
[14] C. de Boor, A Practical Guide to Splines, Springer, New York, 1978.
[15] J.H. Friedman, Multivariate adaptive regression splines, Ann. Statist. 19 (1) (1991) 1–141.
[16] K. Funahashi, On the approximate realization of continuous mappings by neural networks, Neural Networks 2 (1989) 183–192.
[17] A. Gersho, R.M. Gray, Vector Quantization and Signal Compression, Kluwer Academic, Boston, 1992.
[18] F. Girosi, T. Poggio, Networks and the best approximation property, Biol. Cybernet. 63 (1990) 169–176.
[19] D.J. Hand, Kernel Discriminant Analysis, Research Studies Press (Wiley), Chichester, 1982.
[20] W. Härdle, Applied Nonparametric Regression, Cambridge University Press, Cambridge, 1990.
[21] S. Haykin, Neural Networks: A Comprehensive Foundation, IEEE Press, Macmillan, New York, 1994.
[22] K.E. Iverson, A Programming Language, Wiley, New York, 1962.
[23] A. Kołcz, Approximation properties of memory-based artificial neural networks, Ph.D. Thesis, University of Manchester Institute of Science and Technology (UMIST), Department of Electrical Engineering and Electronics, 1996.
[24] A. Kołcz, N.M. Allinson, General memory neural network – extending the properties of basis networks to RAM-based architectures, Proceedings of the 1995 IEEE International Conference on Neural Networks (ICNN '95), Perth, Western Australia, 1995, pp. 1638–1643.
[25] A. Kołcz, N.M. Allinson, N-tuple regression network, Neural Networks 9 (1996) 855–869.
[26] S.H. Lane, D.A. Handelman, J.J. Gelfand, Theory and development of higher-order CMAC neural networks, IEEE Control Systems Mag. (1992) 23–30.
[27] W.T. Miller, F.H. Glanz, L.G. Kraft, CMAC: An associative neural network alternative to back-propagation, Proc. IEEE 78 (10) (1990) 1561–1567.
[28] J. Moody, Fast learning in multi-resolution hierarchies, in: D.S. Touretzky (Ed.), Advances in Neural Information Processing Systems 1, Morgan Kaufmann, Los Altos, CA, 1989, pp. 29–39.
[29] J. Moody, C.J. Darken, Fast learning in networks of locally-tuned processing units, Neural Comput. 1 (1989) 282–294.
[30] E. Nadaraya, On estimating regression, Theory Probab. Appl. 15 (1964) 134–137.
[31] J. Park, I.W. Sandberg, Universal approximation using radial basis function networks, Neural Comput. 3 (1991) 246–257.
[32] J. Platt, A resource-allocating network for function interpolation, Neural Comput. 3 (2) (1991) 213–225.

[33] M.J.D. Powell, Radial basis functions for multivariable interpolation: a review, in: J.C. Mason, M.G. Cox (Eds.), Algorithms for Approximation, Clarendon Press, Oxford, 1987.
[34] M.J.D. Powell, The theory of radial basis functions in 1990, in: W.A. Light (Ed.), Advances in Numerical Analysis Volume II: Wavelets, Subdivision Algorithms and Radial Basis Functions, Oxford University Press, Oxford, 1992, pp. 105–210.
[35] D.W. Scott, Multivariate Density Estimation, Wiley-Interscience, New York, 1992.
[36] D.F. Specht, Probabilistic neural networks, Neural Networks 3 (1) (1990) 109–118.
[37] D.F. Specht, A general regression neural network, IEEE Trans. Neural Networks 2 (6) (1991) 568–576.
[38] G.D. Tattersall, S. Foster, R.D. Johnston, Single-layer lookup perceptrons, IEE Proc. F 138 (1991) 46–54.
[39] H. Tolle, E. Ersü, Neurocontrol – Learning Control Systems Inspired by Neuronal Architectures and Human Problem Solving Strategies, Springer, Berlin, 1992.
[40] G. Wahba, Spline Models for Observational Data, SIAM, Philadelpia, PA, 1990.
[41] G.S. Watson, Smooth regression analysis, Sankhya A 26 (1964) 359–372.
[42] H. White, Learning in artificial neural networks, Neural Comput. 1 (4) (1989) 425–464.
[43] B. Widrow, S.D. Stearns, Adaptive Signal Processing, Prentice-Hall, Englewood Cliffs, NJ, 1985.
[44] R. Rohwer, M. Morciniec, A theoretical and experimental account of n-tuple classifier performance, Neural Comput. 8 (3) (1996) 629–642.



Aleksander Kolcz received the magister inżynier degree from Politechnika Łódzka, Poland, in 1991, the M.Sc. by Research degree from University of York, UK, in 1993, and the Ph.D. degree from UMIST, UK, in 1996, all in electronics. Since 1996 he has been with the University of Colorado at Colorado Springs as a post-doctoral Research Associate in the Department of Electrical and Computer Engineering. His research interests are in the area of intelligent/adaptive systems and neural networks.



Nigel M. Allinson is Professor of Image Engineering and Head of the Image Engineering and Neural Computing Group at UMIST. His research interests include development of imaging devices and systems, especially for scientific instruments, pattern recognition and neural networks – with a particular interest in memory-based networks and self-organising systems. He has published some 120 journal and conference papers.