On Global–Local Artificial Neural Networks for Function Approximation

David Wedge, David Ingram, David McLean, Clive Mingham, and Zuhair Bandar

***Abstract*—We present a hybrid radial basis function (RBF) sigmoid neural network with a three-step training algorithm that utilizes both global search and gradient descent training. The algorithm used is intended to identify global features of an input–output relationship before adding local detail to the approximating function. It aims to achieve efficient function approximation through the separate identification of aspects of a relationship that are expressed universally from those that vary only within particular regions of the input space. We test the effectiveness of our method using five regression tasks; four use synthetic datasets while the last problem uses real-world data on the wave overtopping of seawalls. It is shown that the hybrid architecture is often superior to architectures containing neurons of a single type in several ways: lower mean square errors are often achievable using fewer hidden neurons and with less need for regularization. Our global–local artificial neural network (GL-ANN) is also seen to compare favorably with both perceptron radial basis net and regression tree derived RBFs. A number of issues concerning the training of GL-ANNs are discussed: the use of regularization, the inclusion of a gradient descent optimization step, the choice of RBF spreads, model selection, and the development of appropriate stopping criteria.**

***Index Terms*—Global, hybrid, local, overtopping, regularization.**

I. Introduction

THIS paper describes and assesses methods for the creation and training of artificial neural networks (ANNs) that contain two different types of artificial neurons: those based on projection functions and those based on kernel-type functions. For the purposes of the research described in this paper, we have chosen two commonly used functions as representative of these types of function: the bipolar sigmoid function described by (1) and (2) and the Gaussian radial basis function (RBF) of (3) and (4). However, we believe that our results are applicable to a variety of related functions

$$v = \mathbf{i}^{\mathbf{T}}\mathbf{w} - w_0 \tag{1}$$

$$y = \frac{1-\exp(-v)}{1+\exp(-v)} \tag{2}$$

$$v = |\mathbf{i} - \mathbf{w}| \tag{3}$$

$$y = \exp\left(-w_0^2 v^2\right). \tag{4}$$

In (3), $|\cdot|$ indicates the Euclidean norm. In both cases the output y is dependent upon an input vector $\mathbf{i}$, a weight vector of the same order as the input vector $\mathbf{w}$, and a bias weight w_0. However, for RBF neurons [(3), (4)] the vector $\mathbf{w}$ is usually interpreted as a "center" and the scalar w_0 as "steepness," since the response of the neuron drops off with distance from $\mathbf{w}$ at a rate determined by w_0.

The practical aim in creating these hybrid networks is to give an accurate approximation of an unknown real-valued function using noisy data by separating out the global and local parts of the approximation. Support for the separation of global and local parts of a function comes from three main areas: mathematical analysis, cognitive psychology, and developments within machine intelligence.

Donoho and Johnstone [1] have shown that kernel-based and projection-based functions have complementary properties. In particular, they show that "ancillary smoothness" in the target function may be used to reduce the effective dimensionality of the data. They define an angularly smooth function as one that varies slowly with angle, while a function with radial smoothness shows small local variations in value. Projection-based functions are seen to respond well to angular smoothness while kernel-based functions respond well to radial smoothness. For complex, high-dimensional functions, we expect to find aspects of both types of smoothness. In order to achieve optimum results with the smallest possible network, it therefore seems advisable to use neurons with both projection-based and kernel-based functions.

As well as having a sound mathematical basis hybrid networks seem to have more biological validity than pure multilayer perceptron (MLP) networks [2], [3]. There is considerable evidence that the human brain processes information in a modular way [4], [5]. For example, global and local aspects of visual stimuli are processed by different parts of the brain, suggesting the specialization of neurons for these different purposes [6], [7]. Further, brain development often occurs in stages, with each stage dependent upon the completion of previous stages [8]. The architectural structure of our hybrid networks is similarly reflected in a stepwise training algorithm [4].

As computing power increases computer scientists are dealing with larger, higher dimensioned datasets and, presumably, more complex underlying functions. Hrycej [4] and Moussa [9] believe that there is a need to use more complex models such as modular ANNs in order to satisfactorily model these functions. Each module within a network may then be

Manuscript received June 30, 2005; revised December 8, 2005. The wave overtopping data used in this paper was provided by the CLASH project supported by the European Commission under EESD within the Fifth Framework Programme under Contract EVK3-CT-2001-00058.

D. Wedge is with Silent Talker, Manchester Metropolitan University, Manchester, Lancashire M1 5GD, U.K. (e-mail: d.wedge@mmu.ac.uk).

D. Ingram is with School of Engineering and Electronics, University of Edinburgh, Edinburgh EH9 3JL, U.K. (e-mail: David.Ingram@ed.ac.uk).

D. McLean, C. Mingham, and Z. Bandar are with the Department of Computing and Mathematics, Manchester Metropolitan University, Manchester, Lancashire M1 5GD, U.K. (e-mail: d.mclean@mmu.ac.uk; c.mingham@mmu.ac.uk; z.bandar@mmu.ac.uk).

Digital Object Identifier 10.1109/TNN.2006.875972

assigned a different task, or subtask, according to the particular architecture of that module or the training method applied to it.

Poggio and Girosi have suggested the use of networks containing both Gaussian and other functions in a single layer. These networks are extensions of traditional RBF networks called "HyperBFs" [10]. They contain a single hidden layer containing Gaussian functions of variable width and additional nonradial functions. Girosi *et al.* [11] have demonstrated mathematically the close relationship of HyperBFs to regularization theory. The hybrid networks described in this paper may be seen as an implementation of HyperBFs, with a particular emphasis on the separation of global and local variations in the regression function.

Moody and, more recently, Ferrari *et al.*, have highlighted the difficulty in identifying both the coarse structure and the fine detail of an input–output relationship [12], [13]. Their multiresolution techniques use RBF neurons of differing widths to solve this scaling problem. Our approach builds on this work, allowing extra flexibility in the choice of RBF widths and the addition of sigmoid functions to map features of the function that are more suited to this geometric form.

The research described in this paper originates from investigations into alternative neural network architectures for the prediction of overtopping of seawalls during storm events [14]. The data used in training have a high-dimensional input space and are mostly found in small clusters, each of which corresponds to a particular seawall geometry. Details are given in Section III-B. We have reported previously that RBF networks give lower errors than MLP networks on these data [15]. This is unsurprising, given the well-known ability of RBF networks to interpolate within clusters of data [2], [16], [17]. However, we are dissatisfied with the use of pure RBF networks with this dataset for three practical reasons.

- Due to the high dimensionality of the data, the RBF networks created are very large, containing approximately 200 RBF neurons. It is hoped that knowledge-extraction techniques will be applied to the networks in the future. However, in order to obtain meaningful knowledge, one must start with smaller networks. Friedman and Stuetze [18] have suggested that projection-based models overcome the "curse of dimensionality" better than kernel-based models, primarily because they take into account all data points during training.
- RBF networks are generally effective at local estimation but less effective at global approximation [17]. One of the aims of the research is to create a system that can predict overtopping for novel seawall geometries, so the ability to interpolate into sparsely populated areas of input space is a priority.
- It has been widely reported that RBF networks have a tendency to overfit data unless strong regularization is introduced [19], [20]. However, the use of regularization adds an additional parameter into model estimation, making the training process less automatic.

This paper presents a novel way to create and train networks that contain both sigmoid and RBF neurons in a single layer and examines several issues related to the implementation of such networks. In our method, the selection and training of sigmoidal and RBF neurons are treated separately and sequentially. The aim is to create a global approximation using sigmoid neurons and to then add local detail to the approximation function with RBF neurons. Although designed for a particular dataset, we believe that our training algorithm could have applicability to a range of datasets. We investigate the extent of its usefulness through the use of synthetic benchmark datasets as well as the real-world overtopping dataset.

Recent research has been carried out into the use of genetic algorithms for the selection of MLP-RBF architectures [21], [22]. This has focused on the process of architecture selection and uses Levenberg–Marquardt training to optimize weights. In contrast, we take a fairly crude approach to architecture selection, training all reasonable architectures, but have developed a novel training algorithm that takes advantage of the characteristics of a hybrid architecture.

Our work is related to that of Cohen and Intrator [23]. They have created hybrid networks which they call perceptron radial basis nets (PRBFNs). Their approach is to cluster the data and then to choose a neuron, either sigmoidal or radial-based, that approximates the local function within each cluster. Our approach also has some similarities to the regression tree derived RBFs (RT-RBF) of Orr [24]. Like them, we take a multiresolution approach, starting with coarse structure and moving to finer detail. However, Orr *et al.* do not use sigmoid neurons or their associated training algorithms. Finally, our method may be compared to the modular ANNs exemplified by the work of Jordan and Jacobs [25]. Their "hierarchical mixture of experts" approach explicitly partitions data and creates separate networks for each partition. The outputs of the networks are then combined using a "soft" gating function.

The three approaches described above all have a common feature: they all involve the partitioning of data as a first step. Rather than splitting up the data, our approach is to split up the underlying function into global and local features. We use all of the training data in all phases of training. The choice of neuron transfer function and synaptic weights is therefore made on the basis of variations in the overall input–output mapping rather than on features of a particular input cluster.

The structure of the rest of this paper is as follows. Section II describes the global–local artificial neural network (GL-ANN) training algorithm in detail. Sections III-A and -B describe, respectively, the salient features of the benchmark and overtopping datasets used as demonstration applications. Section III-C gives specific details of the training method used for each dataset. Section IV reports the results from all datasets and discusses issues concerning the use of GL-ANNs including the use of regularization and gradient descent optimization. Section V gives some concluding remarks and describes areas of future research.

II. Global–Local Artificial Neural Networks

A. Overview

MLP and RBF networks have complementary properties. While both are theoretically capable of approximating a function to arbitrary accuracy using a single hidden layer [26], [27], their operation is quite different [28]. MLP networks have a fixed architecture and are usually trained using a variant of gradient descent. They invariably incorporate neurons with sigmoid activation functions. Their response therefore varies across the whole input space and weight training is affected by all training points. RBF networks, on the other hand, are most commonly created using a constructive algorithm. Gradient descent training is usually replaced by deterministic, global methods such as forward selection of centers with orthogonal least squares (FS-OLS). This method is described in detail in Section II-B.

Whereas MLPs are effective at identifying global features of the underlying function, RBF networks have the capacity to identify local variation in the function [4], [29], [30]. MLPs are more distributive in their representation of the input–output relationship. For this reason they may be seen as more "emergent" and opaque [4], [31]. On the other hand, RBF centers are deliberately selected, often from the training set, as representatives of the training set, or a subset thereof. RBF networks are slightly more transparent and are easier to interpret symbolically [4]. The training of RBF networks is generally faster, since it involves the solving of linear rather than nonlinear equations [19], [32]. However, RBF networks often contain many more neurons than the corresponding MLP networks, partly offsetting the advantage in computational efficiency [30].

A hybrid ANN containing both sigmoidal and radial neurons may have the advantages of both RBF and MLP ANNs, i.e., computational efficiency, good generalization ability, and a compact network architecture. We approximate on a global level first using an MLP and then add RBF neurons using FS-OLS, in order to add local detail to the approximating function. For this reason, we call our network a global–local artificial neural network. Identifying coarse structure before fine detail makes sense from a computational point of view [12]. This sequential process may also mirror the operation of biological brains: there is considerable evidence from cognitive psychology that humans identify global features of visual stimuli before local features [33] and that the global features affect the interpretation of the local features [34]. The training process is completed with an optimization step that adjusts the weights of all neurons, including RBF centers and widths.

Our approach has a number of practical advantages.

- There is no need to cluster the data prior to training. This gives more flexibility to the FS-OLS process and avoids three possible problems. First, clustering may reflect the distribution of the available data rather than the underlying functionality. Secondly, clustering generally reflects the distribution of the input data but does not take into account the distribution of the output data [29]. This is a problem for highly nonlinear data such as the wave overtopping data, for which small changes in the inputs sometimes cause large changes in the output. Finally, unsupervised clustering can lead to very large, and therefore overfitted, networks [35].
- All phases of training take into account all of the training data, keeping the variance low [36]. This is not the case for PRBFN [23] or RT-RBF [24], which choose each neuron to be representative of a particular cluster of data points.
- Training is carried out in a stepwise fashion, as illustrated in Fig. 1. In most cases, each step improves upon the previous approximation. It is therefore possible to assess the effectiveness of each step individually, giving some insight into the training process [4].
- It is often possible to achieve lower test errors using GL-ANNs than with either MLP or RBF networks (Section IV-A).
- The GL-ANNs required to give a given test error are generally smaller than the corresponding RBF networks (Section IV-B).
- Our results suggest that, unlike pure RBF networks and PRBFNs, GL-ANNs do not require regularization. From our investigations, it appears that the MLP created in the first phase of training has a moderating effect on the selection and training of RBF neurons added subsequently (Section IV-E).

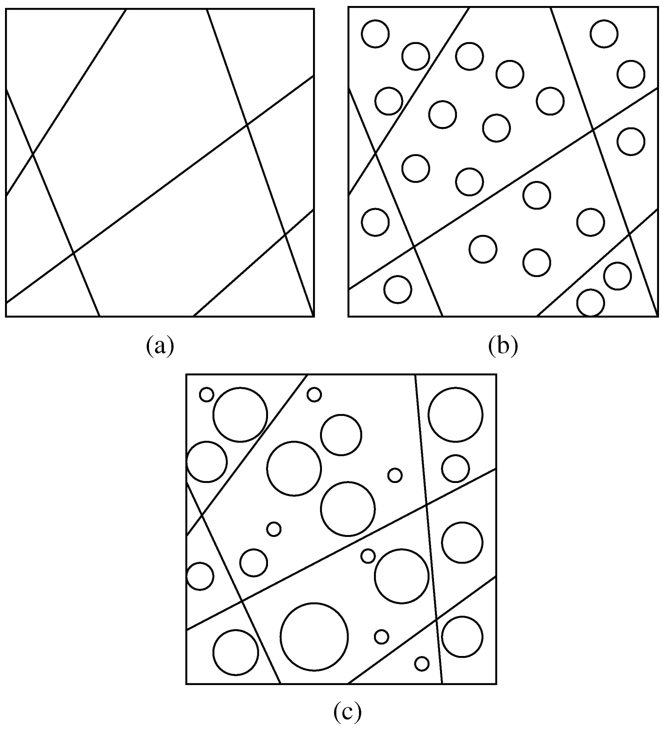


Fig. 1. Diagrammatic representation of the GL-ANN training process. (a) Sigmoid neurons only, (b) hybrid with fixed RBFs, and (c) hybrid with adjustable RBFs.

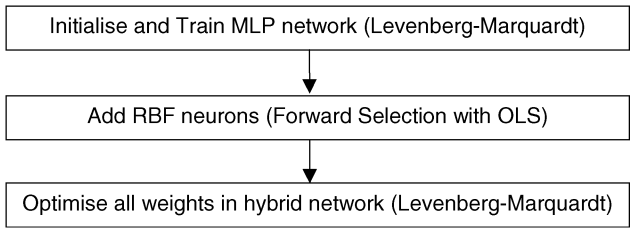

Initialise and Train MLP network (Levenberg-Marquardt)
Add RBF neurons (Forward Selection with OLS)
Optimise all weights in hybrid network (Levenberg-Marquardt)

Fig. 2. GL-ANN three-step training algorithm.

B. Training Method

The training of GL-ANNs is performed in three stages, as illustrated in Fig. 2. At each stage attempts have been made to select a training method that is efficient in terms of computational power, given the architecture of the network. In order to automate the process as much as possible, algorithms whose outcome is highly dependent upon user-determined training parameters (such as momentum coefficient or regularization parameter) have been avoided.

Before training is performed the data are scaled, so that each input attribute has a similar influence on training. All input and output attributes undergo a linear transformation to give them a range of $[-0.8, 0.8]$. The same training data are then used throughout the training process. Training starts with an untrained MLP, with weights initialized to random values in the range $[-1.0, 1.0]$. The input and weight initialization ranges have been chosen such that the hidden layer neurons have outputs with standard deviation of approximately one. LeCun *et al.* have shown that gradient descent training is most effective under these circumstances [37].

Fixed numbers of sigmoidal neurons are used in a single hidden layer. The output neuron has a linear activation function and fixed bias. In order to achieve efficient gradient descent, the Levenberg–Marquardt (LM) method [38], [39] is used to train the MLP networks. In order to use this procedure, local partial derivatives are first calculated for the input weights (including bias weight) using (5). Local inputs and weights are given by i_k and w_k, respectively, and y is the pertinent neuron's output

$$\frac{\partial y}{\partial w_k} = \frac{1-y^2}{2} i_k. \tag{5}$$

The local partial derivatives may be used to obtain estimates of the Hessian matrix using the standard LM procedure. Training is stopped reaching a minimum error gradient or after a maximum number of epochs.

In the second stage, RBF neurons are added to the existing MLP network. The RBF centers are chosen from the training data using the FS procedure. The RBF neurons employ symmetrical radial functions with fixed widths at this stage. After each addition, the output weights from both sigmoidal and RBF neurons are adjusted using OLS minimization. The algorithm used has been adapted from Chen *et al.*'s method [32], [40] to make it applicable to hybrid networks. If the training data contain m items, each is regarded as a potential RBF center and the full design matrix $\mathbf{F}$ is an m-by-m matrix containing the outputs of each RBF neuron given each input. The design matrix for a network containing p RBF centers $\mathbf{H}$ is an m-by-p matrix containing columns selected from $\mathbf{F}$. If the target outputs are given by $\mathbf{t}$, the optimal output weights $\hat{\mathbf{w}}$ may then be determined from (6), giving the minimum least square error

$$\hat{\mathbf{w}} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{t}. \tag{6}$$

Chen factorizes the design matrix into an orthogonal matrix $\tilde{\mathbf{F}}$ and an upper triangular matrix, so saving computational requirements. $\tilde{\mathbf{F}}$ is made up of m columns, each representing the outputs of a particular RBF neuron. These may be represented by column vectors $\tilde{\mathbf{f}}_J$, where $1 \leq J \leq m$. $\tilde{\mathbf{F}}$ is adjusted according to (7) each time a neuron is added in order to keep the columns orthogonal. This leads to a series of orthogonal matrices $\tilde{\mathbf{F}}_1, \tilde{\mathbf{F}}_2 \cdots \tilde{\mathbf{F}}_p$, in which the subscripts indicate the number of RBF neurons present in the network

$$\tilde{\mathbf{F}}_{\mathbf{n}+1} = \tilde{\mathbf{F}}_{\mathbf{n}} - \frac{\tilde{\mathbf{f}}_{\mathbf{J}}\tilde{\mathbf{f}}_{\mathbf{J}}^{\mathbf{T}}\tilde{\mathbf{F}}_{\mathbf{n}}}{\tilde{\mathbf{f}}_{\mathbf{J}}^{\mathbf{T}}\tilde{\mathbf{f}}_{\mathbf{J}}}. \tag{7}$$

The reduction in sum squared error (SSE) is then given by (8). By running through all possible values of J, one can identify the neuron that gives the greatest reduction in SSE. Each time a neuron is added, its index must be removed from the list of available values to ensure that the same neuron is not added twice, which would result in a singular design matrix

$$\tilde{S}_m - \tilde{S}_{m+1} = \frac{(\mathbf{t}^{\mathbf{T}}\tilde{\mathbf{f}}_{\mathbf{J}})^2}{\tilde{\mathbf{f}}_{\mathbf{J}}^{\mathbf{T}}\tilde{\mathbf{f}}_{\mathbf{J}}}. \tag{8}$$

Obtaining the error reduction using (8) is considerably quicker than obtaining the design matrix $\mathbf{H}$ for each J, solving (6), and then simulating all potential networks to obtain the possible SSEs. The reason for the efficiency of the orthogonal least squares method is that it recognizes that only the component of the new column in a direction orthogonal to the span of the existing design matrix $\tilde{\mathbf{f}}_{\mathbf{J}}$ can contribute to a reduction in SSE. It is therefore unnecessary to consider the whole of the design matrix $\mathbf{H}$. Full derivations of (7) and (8) are provided in [40] and [32].

In our method, only the components of the candidate centers orthogonal to the outputs of both the sigmoid and the RBF neurons are considered when calculating the error reduction. This requires the following modifications.

- The addition of extra columns to the design matrix to represent the outputs of the sigmoid neurons. $\mathbf{F}$ is therefore nonsquare, containing, for m training items and n sigmoid neurons, m rows and $m+n$ columns.
- Before any RBF neurons are added, the design matrix must be orthogonalized by carrying out the orthogonalization of (7) for each existing sigmoid neuron, so ensuring that only the components orthogonal to the existing neurons' outputs are considered.

Adding the RBF neurons to a preexisting MLP network results in the selection of different centers to those that would be chosen if adding to an empty network. This is because the outputs of the sigmoid neurons already span some of the input space. This affects the values within the matrix $\tilde{\mathbf{F}}$. As a consequence, the center that is chosen will be the one that results in hidden layer outputs with the largest component orthogonal to the outputs of the existing neurons. A nonformal way of viewing this phenomenon is that the RBF neurons "fill the gaps" left by the sigmoid functions.

During the second training step, only the connections between hidden and output neurons are set using the modified OLS procedure. In the final training stage all weights, including hidden layer weights and each RBF steepness, are optimized using LM training. The local partial derivatives for RBF weights (centers) and steepness are given, respectively, by (9) and (10). i_k, w_k, and y are used as in (5), while v is the Euclidean norm of (3).

For RBF input weights (centers)

$$\frac{\partial y}{\partial w_k} = 2yw_0^2(i_k - w_k). \tag{9}$$

For RBF bias weight (steepness)

$$\frac{\partial y}{\partial w_k} = -2w_0v^2y. \tag{10}$$

A restriction has to be introduced to prevent zero or negative steepnesses. Where a weight update would produce a nonpositive steepness, that particular update is not applied.

Applying the LM method to the partial derivatives obtained from (5), (9), and (10) optimizes the whole network. This has a number of effects.

- The steepnesses of RBF neurons are allowed to vary, so that RBF neurons do not all have the same spread.
- Errors are backpropagated, so the connections between the input and hidden layer neurons may be adjusted to accommodate the presence of the RBF neurons.
- RBF centers are allowed to move. The centers are no longer constrained to coincide with a training pattern, so the position of a radial function may be fine-tuned such that it represents the data cluster around it more effectively.

Using this algorithm, a series of networks with different architectures are created. Their performance is then assessed using unseen test data. In each case, the data are randomly sampled several times to determine the training—test split and averages are taken over all runs. Ten runs are used for the benchmark tests and 30 for the wave overtopping data.

III. Applications

A. Benchmark Datasets

Four benchmark tests are employed. They are all function approximation tasks using synthetic data. The tests are taken from Cohen and Intrator [23], where comparisons are made with a number of other approaches. For this reason the treatment varies between the different tests but is consistent with the approach of Cohen and Intrator. While this creates some inconsistency, it allows the consideration of a variety of datasets and permits comparison with a number of alternative methods.

The first function is

$$f(x) = \sin(12x) \tag{11}$$

with x randomly selected from [0,1] and $f(x)$ corrupted by Gaussian noise with standard deviation 0.1 and a mean of zero. The training and test sets both contain 50 samples [32]. Given the noisy data, the minimum mean square error (MSE) achievable with the test data is 0.01.

The second function is the two-dimensional sine wave

$$f(x) = 0.8\sin\left(\frac{x_1}{4}\right)\sin\left(\frac{x_2}{2}\right) \tag{12}$$

with $x_1 = [0, 10]$ and $x_2 = [-5, 5]$. The training data are made up of 200 randomly selected items, again corrupted with Gaussian noise of standard deviation 0.1 and mean zero. However, clean data are used for testing purposes, arranged in a 20 by 20 grid to cover the entire input space. The test set therefore contains 400 data items.

The third function is a simulated alternating current used by Friedman in the evaluation of multivariate adaptive regression splines [41]. It is given by

$$Z(R, \omega, L, C) = \sqrt{R^2 + (\omega L - 1/\omega C)^2} \tag{13}$$

where Z is the impedance, R the resistance, ω the angular frequency, L the inductance, and C the capacitance of the circuit. The input ranges are $R = [0, 100]$, $\omega = [40\pi, 560\pi]$, $L = [0, 1]$, and $C = [1 \times 10^{-6}, 11 \times 10^{-6}]$. Two hundred random samples, with Gaussian noise of standard deviation 175 and zero mean applied to Z, are used for training. Five thousand random clean samples are used for testing.

The fourth function is the Hermite polynomial

$$f(x) = 1 + (1 - x + 2x^2)e^{-x^2} \tag{14}$$

with x randomly selected from [−4,4]. One hundred random samples corrupted by Gaussian noise of standard deviation 0.1 and zero mean are used for training purposes. One hundred clean samples are used for testing. This function was first used by Mackay [42].

B. Wave Overtopping

Much research has been conducted into predicting overtopping at seawalls during storm events. One approach is to use scale models in laboratories [43], but this is expensive and time-consuming. An alternative is to numerically model a particular seawall configuration and sea state, e.g., [44]. However, accurate simulation requires a detailed knowledge of both the geometry of the seawall and sea conditions. Results may therefore be applied only to individual scenarios.

As part of the European CLASH project [14], [45], a large overtopping database has been compiled. We have used this database as a resource for testing our GL-ANNs. The data have been collected from both model and prototype sites and cover a wide range of defensive structures and incident wave conditions. They comprise a large number of independent variables and are, consequently, sparse. Furthermore, the data tend to cluster into groups reflecting different physical structures that may be used as seawalls. In between these clusters are large regions, so-called "white spots," which are virtually devoid of data. Many of these white spots represent structural configurations that are impossible, impractical, or ineffectual as a sea defense. The problem should therefore be viewed as a series of subtasks, each having the aim of predicting overtopping volumes for a particular subset of all possible seawall configurations. Due to the locally varying nature of the data, it might be expected that this problem would be solved more accurately by RBF than MLP networks [17], [46]. On the other hand, the curse of dimensionality associated with high-dimensional inputs might be better solved with an MLP network [18].

Ten input parameters are selected for training primarily on the basis of information content as described in an earlier paper

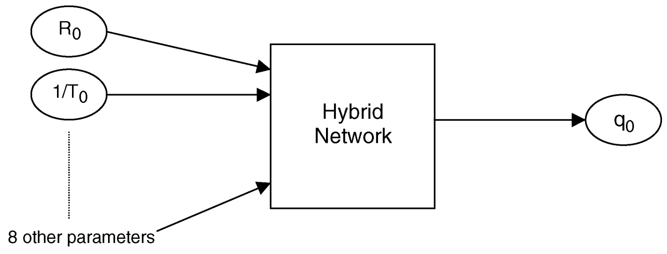


Fig. 3. CLASH input and output parameters.

TABLE I
MEAN SQUARE ERRORS WITH TEST DATA FOR EACH DATASET

	1D sine	2D sine	Friedman	Mackay	CLASH
MLP(L-M)	1.75e-2	1.28e-3	1.02e-1	2.14e-3	1.16e-2
FS-OLS	1.19e-2	0.95e-3	1.52e-1	1.41e-3	1.04e-2
GL-ANN	1.20e-2	1.11e-3	0.98e-1	1.30e-3	0.94e-2
RT-RBF	0.88e-2	2.28e-3	1.12e-1	1.52e-3	-
PRBFN	0.66e-2	1.28e-3	1.50e-1	1.50e-3	-
Minimum error	1.00e-2	-	-	-	-

[15] (see Fig. 3). Of these parameters, seven describe the structure of the seawall in question. The remaining parameters are the angle of wave attack, the water depth at the toe of the wall, and the mean wave period. The single output parameter is the logarithm of the mean overtopping rate per meter of wall per second q_0. The inverses or logarithms of some inputs are utilized in order to give near-normal distributions, and all inputs and the output are normalized to a range of [−0.8,0.8] using a linear transformation. The original dataset contains parameters indicating the reliability of each data point. These are used to select the more reliable elements, resulting in 3053 items used in neural network training. [15]

From previous research [43] it is known that the parameters R_0 (dimensionless freeboard) and T_0 have the most influence on q_0 and that the three quantities are related approximately by (15)

$$q_0 = AT_0 \exp\left(\frac{-BR_0}{T_0}\right). \tag{15}$$

In this equation, A and B are empirically determined parameters that depend on the structure of the seawall. A and B vary slowly with changes in the structure. Further, B is on the order of 3000 times the size of A. Treating A and B as constants therefore gives the approximately linear relationship in (16)

$$ln(q_0) \approx \frac{-BR_0}{T_0}. \tag{16}$$

A hybrid network could be well suited to these data, since the MLP network may represent the approximate relationship in (16) well, leaving the RBF neurons to identify local variations in the function [47], [48].

C. Method

For each dataset a number of MLP networks were created containing a single hidden layer and different numbers of sigmoid neurons. Weights were initialized to small random values and training performed using the LM algorithm. Training was stopped when a minimum gradient was achieved or a maximum number of training epochs had been performed.

RBF training was performed using the FS-OLS algorithm. This resulted in a series of networks for each dataset, each containing a different number of RBF neurons. Each architecture, corresponding to a unique number of RBF neurons, has been assessed separately.

GL-ANNs used the MLP networks described above as a starting point. RBF neurons were added using a modified FS-OLS algorithm (see Section II-B). Again each architecture, determined by both the number of sigmoid and RBF neurons, was considered separately.

For all three types of network, a series of networks were created. The number of sigmoid neurons ranged between one and ten initially. Further sigmoid neurons were added only if the minimum test error occurred with ten sigmoid neurons. The same procedure was followed when adding RBF neurons. The maximum number of RBF neurons was a quarter of the number of training patterns. This condition was rarely invoked as the minimum error was usually reached before this number of neurons had been added. A range of RBF steepnesses (starting steepnesses in the case of the GL-ANNs) were used for RBF and GL-ANN networks. All results given in the next section refer to the ANN architecture and RBF steepness that gave the lowest test MSE when averaged over all test datasets.

IV. RESULTS AND DISCUSSION

A. Mean Square Errors

Mean MSE results for the unseen test data, averaged over all runs for MLP, FS-OLS, GL-ANN, RT-RBF, and PRBFN networks, are given in Table I. RT-RBF results are taken from [23], [24], and [49]. PRBFN results are from [23]. In the case of the third dataset we follow Friedman [41] in dividing the MSE by the variance of the test data. The CLASH results are the MSEs of the normalized outputs.

In the case of the more complex datasets (Friedman, Mackay, and CLASH), the GL-ANN gives lower MSEs than both pure networks. In fact, for the CLASH dataset, the GL-ANN gives errors comparable to those from numerical simulation, even though the latter is specific to a particular structure and sea state [44], [50].

With the one-dimensional (1-D) sine data, GL-ANN and pure RBF networks achieve comparable results. After the first two steps of training GL-ANNs again give very similar results to pure RBF networks (Section IV-D). However, gradient descent leads to overtraining of the hybrid networks with this dataset. Overall the introduction of sigmoid neurons appears to be unhelpful for the sinewave datasets. Given the similar shape of the Gaussian function used in our RBF neurons to a sinewave, it is perhaps unsurprising that pure RBF networks can approximate sinewaves well. We would expect other classes of function to be well approximated by pure MLP networks. However, the latter scenario does not present a problem for GL-ANNs. Due to their stepwise training algorithm they start out as MLP networks. If the addition of RBF neurons did not reduce the test error it would be stopped immediately.

TABLE II
NUMBER OF HIDDEN LAYER NEURONS FOR SINE FUNCTION DATASETS

	1D sine			2D sine		
	S	R	T	S	R	T
MLP(L-M)	12	0	12	5	0	5
FS-OLS	0	6	6	0	13	13
GL-ANN	1	6	7	1	16	17

TABLE III
NUMBER OF HIDDEN LAYER NEURONS FOR FRIEDMAN, MACKAY, AND CLASH DATASETS

	Friedman			Mackay			CLASH		
	S	R	T	S	R	T	S	R	T
MLP(L-M)	5	0	5	14	0	14	14	0	14
FS-OLS	0	48	48	0	7	7	0	195	195
GL-ANN	3	3	6	1	2	3	6	74	80

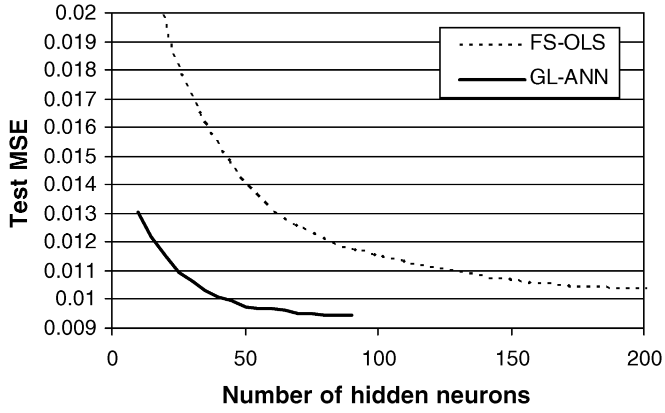


Fig. 4. Test MSE as a function of number of hidden neurons for the CLASH dataset.

For all but the 1-D sine data the errors achieved by both RT-RBF and PRBFN are higher than those produced by GL-ANN. Further, the results quoted for the 1-D sine dataset are below the minimum error achievable, given the inbuilt noise in the data [23]. We believe that this result cannot therefore be taken at face value.

B. Hidden Layer Sizes

Tables II and III give the number of neurons used in the most successful networks. In each case S, R, and T refer to the number of sigmoid, RBF, and total neurons in the hidden layer, respectively. For the first two datasets (Table II), the GL-ANN cannot improve on the RBF networks. However, it imitates the RBF networks by using the smallest possible number of sigmoid neurons, i.e., one.

For the more complex functions (Table III), the GL-ANNs perform better than the RBF networks and create significantly different networks. The GL-ANN uses just three hidden neurons to reproduce Mackay's function and six for Friedman's. With the CLASH dataset the errors of both networks decrease as neurons are added, before reaching a minimum. However, the GL-ANN networks require considerably fewer neurons to achieve a given test error (Fig. 4). These results show that the GL-ANN is parsimonious in its use of hidden neurons. This result is useful in itself since it means that GL-ANNs may be trained quickly. Occam's razor suggests that a more compact model is likely to be closer to a correct solution [51]. Further, if ANNs are to be used for knowledge extraction, it is important that they be small in order to give useful symbolic information [52].

TABLE IV
OPTIMUM RBF SPREADS FOR PURE RBF AND HYBRID NETWORKS

	1D sine	2D sine	Friedman	Mackay	CLASH
FS-OLS	0.45	0.8	2.4	0.1	0.6
GL-ANN	0.4	0.9	0.84	0.15	0.37

TABLE V
MEAN SQUARE ERRORS WITH AND WITHOUT GRADIENT DESCENT OPTIMIZATION

	1D Sine	2D Sine	Friedman	Mackay	CLASH
FS-OLS without L-M	1.19e-2	0.95e-3	1.55e-1	1.41e-3	1.04e-2
FS-OLS with L-M	1.19e-2	0.98e-3	1.25e-1	0.96e-3	1.03e-2
GL-ANN without L-M	1.25e-2	0.95e-3	1.00e-1	1.58e-3	0.99e-2
GL-ANN with L-M	1.20e-2	1.11e-3	0.98e-1	1.30e-3	0.94e-2

C. RBF Spreads

The spread of a radial basis function is defined as the distance from the function's center that will give a response of 0.5. It is inversely proportional to the neuron's steepness and gives an indication of the function's "size." Table IV gives the spreads of the RBF neurons used in the most successful RBF and GL-ANN networks. In the latter case, these are the average finishing spreads, after alteration by the third training step. These results suggest two trends.

- The spreads generally increase as the dimensionality of the input data increases. This is to be expected, since a greater spread is required in order to cover a higher dimensional space.
- The GL-ANNs usually have comparable or narrower spreads than the RBF networks. This confirms the idea that the presence of the sigmoidal neurons frees the RBF neurons to concentrate on local variation in the input–output function. A similar effect has been observed when an output bias is introduced into RBF networks [53].

D. Gradient Descent Optimization

It has been demonstrated previously that gradient descent optimization has a beneficial effect on RBF networks [54]. In this section, we attempt to identify the extent to which the success of GL-ANNs is due to the optimization step in the training algorithm and to what extent it is due to the hybrid architecture. To do this, we have applied gradient descent optimization to pure RBF networks and compared the results for GL-ANNs (see Table V). The final LM optimization step is found to reduce the MSEs of both the GL-ANNs and the pure RBF networks for most datasets, suggesting that this is an important factor in the success of GL-ANNs. Further evidence of the importance of gradient descent optimization is given by a comparison of the network sizes with and without this step. As Table VI shows, considerably

TABLE VI
OPTIMAL HIDDEN LAYER SIZES WITH AND WITHOUT GRADIENT DESCENT OPTIMIZATION

	1D Sine	2D Sine	Friedman	Mackay	CLASH
FS-OLS without L-M	6	13	48	7	195
FS-OLS with L-M	6	16	6	2	75
GL-ANN without L-M	8	10	10	9	186
GL-ANN with L-M	7	17	6	3	80

TABLE VII
TEST MSES WITH REGULARIZATION

	1D Sine	2D Sine	Friedman	Mackay	CLASH
RBF	1.19e-2	0.89e-3	1.46e-1	1.38e-3	0.92e-2
Hybrid	1.19e-2	0.90e-3	1.00e-1	1.53e-3	0.92e-2

TABLE VIII
NUMBER OF HIDDEN NEURONS WITH REGULARIZATION

	1D Sine		2D Sine		Friedman		Mackay		CLASH	
RBF	6	0	16	0	50	0	7	0	382	0
Hybrid	1	6	1	12	5	5	1	8	6	382

fewer neurons are required to achieve an optimum network, i.e., one with the lowest MSE, if the optimization step is performed.

When gradient descent optimization is applied to the pure RBF networks, the advantage of GL-ANNs still remains for the larger, high-dimensional datasets (Friedman and CLASH). This result shows that the hybrid structure of GL-ANNs plays some part in its effectiveness. However, the results for the one- and two-dimensional datasets suggest that a hybrid architecture may not be appropriate for simpler, low-dimensional datasets. Further investigation of the limitations of hybrid architectures is clearly required.

E. Regularization

Previous authors have noted that regularization is usually required when setting RBF output weights [20]. We have introduced regularization into the FS-OLS step for both hybrid and RBF networks, using a wide range of regularization parameters (powers of ten between 1.0 and 10^{-10}). The minimum test MSEs obtained are given in Table VII. All values refer to unoptimized networks, in order to isolate the effect of regularization. A comparison with the first and third rows of Table V shows that regularization has little effect on the errors of well-sized networks. Only those networks that have more than the optimum number of RBF neurons, and are therefore overfitting the data, benefit greatly from regularization.

The relative unimportance of regularization is also seen when considering the sizes of the networks created (Table VIII). The number of neurons within the best performing networks is generally not reduced with the introduction of regularization. Indeed, in many cases the optimum networks have identical architectures to those produced without regularization, with only slight modifications to the output weights. This situation may be contrasted with the effect of gradient descent optimization, which in many cases results in large reductions in network size.

Regularization does result in a lower MSE than gradient descent optimization for the CLASH dataset. However, the number of neurons required to obtain this MSE is very large. The GL-ANN attains its minimum MSE using 80 neurons. In order to improve upon this error, the regularized networks require 263 neurons in the case of the pure RBF network and 278 neurons in the case of the hybrid network. We feel that the smaller GL-ANN must be preferred on the grounds of network size: the smaller network is trained more quickly and, from a Bayesian point of view, is more likely to reflect the underlying function.

The small effect of regularization is seen to some extent for both the pure RBF networks and the hybrid networks and therefore challenges the assumption that regularization is essential in OLS training of RBF networks [20]. However, we believe that the sigmoid neurons in the initial MLP may have a specific regularizing effect on the weights of RBF neurons added subsequently and that further regularization is therefore unnecessary. Further support is given to this idea by a comparison of GL-ANNs and pure RBF networks. The average output weight of a neuron in the most successful RBF networks is 18.8, compared to 1.29 for the best GL-ANNs.

The regularizing effect of the sigmoid neurons may be explained in a qualitative manner as follows. OLS selection identifies the weights that minimize the least square error. However, the core MLP network has already achieved this end, to some extent. When relatively localized RBF neurons are added to the network, their optimum weights are likely to be small in order to minimize the risk of upsetting the generalizing ability of the existing ANN. The number of RBF neurons is likely to be small, for the same reason. Overfit is therefore avoided by using fewer neurons with lower weights. A similar effect has been observed by Orr when wide RBFs are added before narrow RBFs in a pure RBF network [53]. However, we believe that neurons with a bipolar response may have a particularly strong regularizing effect. Further investigation of this issue is required, possibly including a formal mathematical treatment.

F. Stopping Criteria and Model Selection

Related to the topic of regularization is the need for a stopping criterion. The first and third training stages involve LM training and are terminated upon reaching a minimum gradient or after a fixed number of epochs of training have been performed. However, it is not clear at what point the second stage should be stopped, i.e., at what point one should stop adding RBF neurons. The approach used in this study is to consider the second step in isolation by considering the test error obtained at this point. RBF neurons are added ten at a time, until no decrease is seen in this test error. The third training step is then performed on all of the interim networks created, up to the one with the minimum test error.

In order to make training more efficient, we would like to find a method of identifying the optimum network size during the second stage of training. This would save time during the third training phase, as well as the second phase, since it would be necessary to perform LM optimization only on the networks with optimum architecture. The use of a validation set after the FS-OLS phase is not useful for two reasons.

- Validation identifies very large optimum networks after FS-OLS, containing about 200 hidden neurons for the CLASH dataset. However, after LM optimization much more compact networks, containing about 80 neurons for the CLASH dataset, are found to give substantially lower MSEs.

- The use of a validation set before the termination of training introduces information that may bias training. Ideally we wish to identify an optimum model after the FS-OLS step without resorting to the validation set.

Bayesian methods are widely used for model selection [55]. They have a number of advantages over the use of cross-validation: they have a sound mathematical basis, they may incorporate regularization in a natural way, they may be used in the selection of input variables, and they do not require the use of information from a test set [56]. If applied before the final step they refer to unoptimized networks. However, when used as model selection criteria, Bayesian inference techniques assess the evidence of a range of networks being the most "plausible" [57]. Assuming that the final step makes small changes to the overall weights, Bayesian inference could give some indication which architectures are likely to provide good models, before performing this step.

Even if applied after all three phases of training are completed, Bayesian model selection, using a technique such as Bayesian random searching (BARS) [56], would introduce heuristic search methods for model selection that could be more efficient and more automated than a brute force approach. An alternative heuristic would be the use of genetic algorithms (GAs) [21]. Both of these methods could allow the search for number of sigmoid neurons, number of RBF neurons, and selection of input parameters to be assessed concurrently. While they may be fruitful areas of further study, they go beyond the scope of this paper.

V. Conclusion and Future Research

We have presented a novel training method for hybrid MLP-RBF networks. Our method identifies coarse features of an input–output function using sigmoid neurons and then adds detail through localized radial basis functions. This allows the training algorithm to focus on one task at a time, with the most effective technique used for each particular objective. Thus, gradient descent may be used to identify global features of the relationship before forward selection of centers adds local variation to the approximating function.

It has been demonstrated that GL-ANNs can give lower MSEs than pure RBF networks and MLPs on a range of function approximation tasks. GL-ANNs have also been shown to compare favorably with the more sophisticated approaches of PRBFN and RT-RBF. A key feature of GL-ANNs is their efficiency in representing an unknown function: the number of neurons required to approximate a function to a given accuracy is often much lower than with other methods.

For some of the datasets investigated, hybrid networks are unable to improve on pure RBF networks. In these cases, the GL-ANNs seem to imitate pure RBF networks. The best results are found to be comparable to those for pure RBF networks and occur with a single sigmoid neuron, which effectively acts as an additional bias.

We have tried to avoid the use of user-set parameters whenever possible. One of these parameters is the regularization parameter. It has been demonstrated that neither pure RBF networks nor GL-ANNs benefit greatly from regularization, provided networks of appropriate size are used. The belief in the need for regularization may have arisen from a focus on fully interpolated networks containing all training points as radial centers. More compact networks are likely to be less liable to overfit and therefore have less need of regularization. We have shown informally that the bipolar neurons used in our GL-ANNs are likely to have a particularly strong regularizing effect and that GL-ANNs therefore have even less need of additional regularization than do pure RBF networks.

While regularization appears to have little effect on improving the generalizing abilities of GL-ANNs, or well-sized RBF networks, gradient descent optimization appears to be very valuable. An important reason for the success of GL-ANNs is that they combine the closed, stepwise algorithm of forward selection with the more open, parallel algorithm of gradient descent. This results in more compact networks as well as greater prediction accuracy.

A difficulty with the training of any hybrid network is model selection. We have taken a fairly crude approach, but more heuristic methods such as BARS or GAs might prove more efficient. The GL-ANN training algorithm has a specific difficulty in identifying an end-point in the FS-OLS phase of training, which can lead to long training times due to unnecessary training at this stage. Some suggestions have been made including the use of Bayesian inference techniques. This is a possible area of future research.

The boundaries on the effective use of GL-ANNs need to be explored further. Our results suggest that GL-ANNs are better suited to more complex and high-dimensioned problems and it would be interesting to see how GL-ANNs perform on real-world classification problems.

Observations concerning the regularizing effect of the hybrid method and the ability to identify novel functions and to imitate pure networks are somewhat tentative at this time. A rigorous mathematical treatment of these features should be attempted in the future.

Further investigation into the similarities between hybrid artificial neural networks and biological brains might be fruitful. Such research could operate in both directions: biological systems may provide an inspiration for the development of hybrid ANNs; on the other hand, the consideration of hybrid ANNs might help us to understand how and why biological processing often takes place in a modular, stepwise fashion.

Acknowledgment

The authors would like to thank the CLASH project [14] for providing the wave overtopping data used in this paper.

References

[1] D. Donoho and I. Johnstone, "Projection-based approximation and a duality with kernel methods," *Ann. Statist.*, vol. 17, pp. 58–106, 1989.

[2] T. Poggio and F. Girosi, A theory of networks for approximation and learning Massachusetts Inst. Tech., Artificial Intelligence Lab., Cambridge, MA, Jul. 1989, AI Memo 1140.

[3] G. Auda and M. Kamel, "Modular neural networks: a survey," *Int. J. Neural Syst.*, vol. 9, pp. 129–151, 1999.

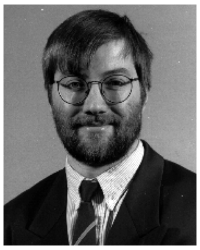
[4] T. Hrycej, *Modular Learning in Neural Networks: A Modularized Approach to Classification*. New York: Wiley, 1992.

[5] P. Poirazi, C. Neocleous, C. Pattichis, and C. Schizas, "Classification capacity of a modular neural network implementing neurally inspired architecture and training rules," *IEEE Trans. Neural Netw.*, vol. 15, no. 3, pp. 597–612, May 2004.

[6] S. Johannes, B. Wieringa, M. Matzke, and T. Munte, "Hierarchical visual stimuli: electrophysical evidence for separating left hemispheric global and local processing mechanisms in humans," *Neurosci. Lett.*, vol. 210, pp. 111–114, 1996.
[7] R. Hubner, "Hemispheric differences in global/local processing revealed by same-different judgements," *Visual Cogn.*, vol. 5, no. 4, pp. 457–478, 1998.
[8] J. Piaget, *Meine Theorie der geistigen Intelligenz*. Frankfurt, Germany: Fischer Taschenbuch Verlag, 1983.
[9] M. Moussa, "Combining expert neural networks using reinforcement feedback for learning primitive grasping behavior," *IEEE Trans. Neural Netw.*, vol. 15, no. 3, pp. 629–638, May 2004.
[10] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, vol. 78, no. 9, pp. 1481–1497, Sep. 1990.
[11] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Comput.*, vol. 7, pp. 219–269, 1995.
[12] J. Moody, "Fast learning in multi resolution hierarchic," in *Advances in Neural Information Processing Systems I*, D. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 29–39.
[13] S. Ferrari, M. Maggioni, and N. Borghese, "Multiscale approximation with hierarchical radial basis functions networks," *IEEE Trans. Neural Netw.*, vol. 15, no. 1, pp. 178–188, Jan. 2004.
[14] J. de Rouck, Crest level assessment of coastal structures by full scale monitoring, neural network prediction and hazard analysis on permissible wave overtopping [Online]. Available: http://www.clash-eu.org
[15] D. Wedge, D. Ingram, D. McLean, C. Mingham, and Z. Bandar, "Neural network architectures and wave overtopping," *Proc. Inst. Civil Engineering 2005: Maritime Engineering*, vol. 158, no. MA3, pp. 123–133, 2005.
[16] D. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Syst.*, vol. 2, pp. 321–355, 1988.
[17] S. Lawrence, A. Tsoi, and A. Back, "Function approximation with neural networks and local methods: bias, variance and smoothness," in *Proc. Australian Conf. Neural Networks (ACNN96)*, P. Bartlett, A. Burkitt, and R. Williamson, Eds., 1996, pp. 16–21.
[18] J. Friedman and W. Stuetzle, "Projection pursuit regression," *J. Amer. Statist. Assoc.*, vol. 76, pp. 817–823, 1981.
[19] C. Campbell, "Constructive learning techniques for designing neural network systems," in *Optimization Techniques*, ser. Neural Network Systemes Techniques and Applications, C. Leondes, Ed. San Diego, CA: Academic, 1998, vol. 2.
[20] M. Orr, "Regularisation in the selection of radial basis function centres," *Neural Comput.*, vol. 7, pp. 606–623, 1995.
[21] N. Jiang, Z. Zhao, and L. Ren, "Design of structural modular neural networks with genetic algorithm," *Adv. Software Eng.*, vol. 34, pp. 17–24, 2003.
[22] Y. Liu and X. Yao, "Evolutionary design of artificial neural networks with different nodes, evolutionary computation," in *Proc. 3rd IEEE Int. Conf. Evolutionary Computation*, 1996, pp. 670–675.
[23] S. Cohen and N. Intrator, "A hybrid projection-based and radial basis function architecture: Initial values and global optimisation," *Pattern Anal. Applicat.*, vol. 5, pp. 113–120, 2002.
[24] M. Orr, K. Takezawa, A. Murray, S. Ninomiya, M. Oide, and T. Leonard, "Combining regression trees and radial basis function networks," *Int. J. Neural Syst.*, vol. 10, no. 6, pp. 453–465, 2000.
[25] M. Jordan and R. Jacobs, "Hierarchical mixtures of experts and the em algorithm," *Neural Comput.*, vol. 6, pp. 181–214, 1994.
[26] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Contr. Signals Syst.*, vol. 2, pp. 303–314, 1989.
[27] J. Park and I. Sandberg, "Approximation and radial-bais-function networks," *Neural Comput.*, vol. 5, pp. 304–316, 1993.
[28] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1999.
[29] C. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Clarendon, 1995.
[30] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Comput.*, vol. 1, pp. 281–294, 1989.
[31] G. Hinton, J. McClelland, and D. Rumelhart, "Distributed representations," in *Parallel Distributed Processing*, ser. Explorations in the Microstructure of Cognition. Cambridge, MA: MIT Press, 1986, vol. 1.
[32] M. Orr, Introduction to radial basis function networks Inst. Adaptive Neural Computation, Univ. Edinburgh, Apr. 1996 [Online]. Available: http://www.anc.ed.ac.uk/ mjo/papers/intro.ps.gz, Tech. Rep.
[33] D. Navon, "Forest before trees: The precedence of global features in visual perception," *Cogn. Psychol.*, vol. 9, pp. 353–383, 1977.
[34] S. Christman, "Individual differences in stroop and local-global processing: A possible role of interhemispheric interaction," *Brain Cogn.*, vol. 45, pp. 97–118, 2001.
[35] S. Arisariyawong and S. Charoenseang, "Dynamic self-organised learning for optimizing the complexity growth of radial basis function neural networks," in *Proc. IEEE Int. Conf. Industrial Technology*, Dec. 11–14, 2002, pp. 655–660.
[36] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Comput.*, vol. 4, pp. 1–58, 1992.
[37] Y. LeCun, L. Bottou, G. Orr, and K.-R. Muller, "Efiicient backprop," in *Neural Networks: Tricks of the Trade*, ser. Lecture Notes in Computer Science, G. Orr and K. Muller, Eds. Berlin, Germany: Springer-Verlag, 1998, vol. 1524.
[38] M. Hagan and M. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Trans. Neural Netw.*, vol. 5, no. 6, pp. 989–993, Nov. 1994.
[39] T. Masters, *Advanced Algorithms for Neural Networks: C++ Source Book*. Frankfurt, Germany: Wiley, 1995.
[40] S. Chen, C. Cowan, and P. Grant, "Orthogonal least squares learning for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 302–309, Mar. 1991.
[41] J. Friedman, "Multivariate adaptive regression splines," *Ann. Statist.*, vol. 19, pp. 1–141, 1991.
[42] D. Mackay, "Bayesian interpolation," *Neural Comput.*, vol. 4, pp. 415–447, 1992.
[43] P. Besley, Overtopping of seawalls: Design and assessment manual HR Wallingford, Environment Agency, Tech. Rep. W178, 1999.
[44] J. Shiach, C. Mingham, D. Ingram, and T. Bruce, "The applicability of the shallow water equations for modelling violent wave overtopping," *Coastal Eng.*, vol. 51, pp. 1–15, 2004.
[45] J. de Rouck, Second detailed interim rep. CLASH, Ghent Univ., Tech. Rep. CLA127/296, Feb. 2004.
[46] M. Hassoun, *Fundamentals of Artificial Neural Networks*. Cambridge, MA: MIT Press, 1995.
[47] H. Mhaskar, "Neural networks for localized approximation of real functions," in *IEEE Workshop Neural Networks Signal Processing*, Sep. 6–9, 1993.
[48] C. Chui, X. Li, and H. Mhaskar, "Neural networks for localized approximation," *Math. Comput.*, vol. 63, pp. 607–623, 1994.
[49] M. Orr, J. Hallam, A. Murray, and T. Leonard, "Assessing RBF networks using delve," *Int. J. Neural Syst.*, vol. 10, no. 5, pp. 397–415, 2000.
[50] K. Hu, C. Mingham, and D. Causon, "Numerical simulation of wave overtopping of coastal structures using the non-linear shallow water equations," *Coastal Eng.*, vol. 41, pp. 433–465, 2000.
[51] D. Mackay, "Probable networks and plausible predictions—A review of practical Bayesian methods for supervised neural networks," *Network: Comput. Neural Syst.*, vol. 6, pp. 469–505, 1995.
[52] E. Kolman and M. Margaliot, "Are artificial neural networks white boxes?," *IEEE Trans. Neural Netw.*, vol. 16, no. 4, pp. 844–851, Jul. 2005.
[53] M. Orr, Matlab functions for radial basis function networks Inst. for Adaptive and Neural Computation (Univ. of Edinburgh), Jun. 1999 [Online]. Available: http://www.anc.ed.ac.uk/ mjo/software/rbf2.zip, Rep. Tech.
[54] F. Schwenker, H. Kestler, and G. Palm, "Three learning phases for radial-basis-function networks," *Neural Netw.*, vol. 14, pp. 439–458, 2001.
[55] G. Schwarz, "Estimating the dimension of a model," *Ann. Statist.*, vol. 6, pp. 461–464, Mar. 1978.
[56] H. Lee, "Model selection for neural network classification," *J. Class.*, vol. 18, pp. 227–243, 2001.
[57] D. Mackay, "A practical Bayesian framework for backprop networks," *Neural Comput.*, vol. 4, pp. 448–472, 1992.

David Wedge received the B.A. degree in chemistry from Oxford University, Oxford, U.K., in 1989, the MSc. degree in software engineering from Huddersfield University, Huddersfield, U.K., in 2001, and the Ph.D. degree in artificial intelligence from Manchester Metropolitan University (MMU), Manchester, U.K., in 2006.

He is currently with Silent Talker, a research and development company owned by MMU, developing the use of artificial neural networks for psychological profiling.

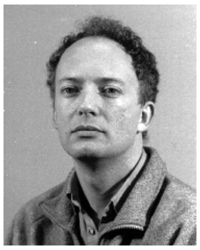


David Ingram received the B.Sc. degree in mathematics, statistics, and computing from the University of Greenwich, Greenwich, U.K., and the Ph.D. degree in computational fluid dynamics from Manchester Metropolitan University (MMU), Manchester, U.K.

He is a Reader in the Institute of Energy Systems, School of Engineering and Electronics, The University of Edinburgh, Edinburgh, U.K. Until recently, he was a Reader in Scientific Computation at MMU. His prime research interests lie in the simulation and computer modeling of wave interactions with coastal structures.

David McLean received the B.Sc. degree in computer science from the University of Leeds, Leeds, U.K., and the Ph.D. degree in neural networks from Manchester Metropolitan University (MMU), Manchester, U.K.

He is a Senior Lecturer in the Department of Computing and Mathematics at MMU. He has previous experience as a researcher and developer at DERA and Thomson Marconi Sonar. His prime academic interests lie in artificial intelligence, especially neural networks and conversational agents.



Clive Mingham received the B.Sc. degree in mathematics from University of Warwick, Coventry, U.K., and the M.A. degree in mathematics from University of California at Los Angeles, Los Angeles.

He is Reader in Hydroinformatics in the Department of Computing and Mathematics at Manchester Metropolitan University (MMU), Manchester, U.K. He has previous experience as a researcher and developer at British Telecom. His prime academic interests lie in computational hydraulics with emphasis on high-resolution methods on Cartesian cut cell meshes.

Zuhair Bandar received the B.Sc. (Eng.) degree in electrical engineering from Mosul University, in 1972, the M.Sc. degree in electronics from the University of Kent at Canterbury, Kent, U.K., in 1974, and the Ph.D. degree in artificial intelligence and neural networks from Brunel University, Uxbridge, Middlesex, U.K., in 1981.

He is a Reader in Intelligent Systems in the Department of Computing and Mathematics, Manchester Metropolitan University (MMU), Manchester, U.K. He is a founder member and the Managing director of Convagent Ltd., a company which undertakes fundamental research and development of Conversational Agents. His research interest also include the application of artificial intelligence techniques to psychological profiling from nonverbal behavior. Aspects of this work have been patented.