Available online at www.sciencedirect.com
SCIENCE DIRECT®
Neurocomputing 69 (2005) 142–157
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Computationally efficient sequential learning algorithms for direct link resource-allocating networks

Vijanth S. Asirvadam[a], Seán F. McLoone[b,*], George W. Irwin[c]

[a]*Faculty of Engineering and Computer Technology, Asian Institute of Medicine Science and Technology, 08000 AmanJaya, Sungai Petani, Kedah Darul Aman, Malaysia*
[b]*Department of Electronic Engineering, National University of Ireland, Maynooth, Maynooth, Co. Kildare, Republic of Ireland*
[c]*School of Electrical and Electronic Engineering, Queen's University Belfast, Belfast BT9 5AH, N. Ireland, UK*

Received 22 April 2004; received in revised form 3 December 2004; accepted 9 February 2005
Available online 24 August 2005

Abstract

Computationally efficient sequential learning algorithms are developed for direct-link resource-allocating networks (DRANs). These are achieved by decomposing existing recursive training algorithms on a layer by layer and neuron by neuron basis. This allows network weights to be updated in an efficient parallel manner and facilitates the implementation of minimal update extensions that yield a significant reduction in computation load per iteration compared to existing sequential learning methods employed in resource-allocation network (RAN) and minimal RAN (MRAN) approaches. The new algorithms, which also incorporate a pruning strategy to control network growth, are evaluated on three different system identification benchmark problems and shown to outperform existing methods both in terms of training error convergence and computational efficiency.

Keywords: System identification; Radial basis functions; Extended Kalman Filter; Resource allocating-network.

*Corresponding author.
E-mail addresses: vijanth@ieee.org (V.S. Asirvadam), sean.mcloone@eeng.may.ie (S.F. McLoone), g.irwin@qub.ac.uk (G.W. Irwin).

0925-2312/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2005.02.017

1. Introduction

While radial basis functions (RBF) have been used for many years for high dimensional interpolation, it was not until 1988 that they were introduced by Broomhead and Lowe [6] as the basis for a neural network architecture. They operate by generating a weighted linear combination of a set of basis functions (normally Gaussian) defined on the input space to give an overall input-output mapping:

$$y_k = f(\boldsymbol{x}_k) = \sum_{i=1}^{m} h_i \phi_i(\boldsymbol{x}_k; \boldsymbol{c}_i, \sigma_i) + b, \tag{1}$$

$$\phi_i(\boldsymbol{x}_k; \boldsymbol{c}_i, \sigma_i) = \exp\left(-\frac{\|\boldsymbol{x}_k - \boldsymbol{c}_i\|^2}{\sigma_i^2}\right). \tag{2}$$

Here h_i are the linear output layer weights, $\boldsymbol{c}_i$ and σ_i are the centres and widths of the Gaussian basis functions, b is an optional bias weight and m is the number of neurons in the network. Practically RBF networks have a number of advantages over the well-known multilayer perceptron (MLP) with regard to training, locality of approximation and transparency. In particular, RBF networks with localised basis functions learn information at one operating point of a nonlinear process without degrading information assimilated at other operating regimes [16].

The main concern when using RBF networks is how to select the centres for the basis functions so that they provide adequate support for the mapping. The traditional approach of placing them on a uniform grid or at the location of each input vector from a subset of the input-data is generally not practical due to the curse of dimensionality. Prior selection of RBF centres with the aid of clustering techniques is possible if representative training data is available a priori. However, even if such data is available there is no guarantee that number and placement of centres will be appropriate if the plant operating point and/or dynamics change significantly with time.

Resource-allocating networks (RAN) were introduced by Platt [17] as a solution to these problems. RANs are simply RBFs in which recursive training of parameters is combined with a procedure for recursively adding centres. In [17] RAN weights were updated using stochastic gradient descent. Kadirkamanathan and Niranjan [10] modified RAN by developing a second order recursive method, known as RAN-EKF (extended Kalman filter) and described it as sequential learning. Yingwei et al. [19] extended the work on RAN-EKF by including a pruning strategy to obtain a more parsimonious RBF network and called it minimal RAN (MRAN).

In this paper two approaches to reduce the computational and memory requirements of sequential learning algorithms are considered. The first involves decomposing training on a layer by layer and neuron by neuron basis and is achieved by neglecting inter neuron weight interactions. This leads to a significant reduction in memory requirements for algorithms that employ cost function curvature information in their weight updates as the approximation of the Hessian matrix (or its

inverse) used to represent the curvature information is consequently much sparser. Network decomposition was first introduced by Jankowski and Kadirkamanathan [8] in connection with the weight covariance matrix in the RAN-EKF algorithm and more recently by Asirvadam et al., [1,3] in connection with a recursive Levenberg–Marquardt algorithm proposed by Ngia and Sjöberg [14].

The second approach to improving efficiency is the adoption of a minimal weight update policy. Here the basic idea is that, rather than updating the entire set of weights at each iteration, only the weights associated with the neuron (or neurons) closest to the current input in the network, known as the *winner neuron(s)* are updated. Junge and Unbehauen [9] did pioneering work on minimal update approaches and a similar methodology has also previously been applied by Asirvadam et al. [2]. Li et al. [11] introduced a minimal update policy for MRANs and showed it to be superior to Junge and Unbehauen's [9] minimal update approach.

The main contribution of this paper is to combine these two approaches to reducing complexity in order to maximize the computational efficiency and performance of sequential learning of radial basis function models, thus increasing their viability for real-time identification of nonlinear processes. Extended-Kalman filter (EKF) and recursive Levenberg-Marquardt (RLM) implementations are developed for a general class of RAN referred to as direct-link RANs (DRAN) or direct-link RBFs (DRBF). DRANs are simply conventional RBFs, as defined by eq. (1), augmented by a linear input–output mapping, that is

$$y_k = \sum_{i=1}^{m} h_i \phi(\boldsymbol{x}_k; \boldsymbol{c}_i, \sigma_i) + \boldsymbol{b}^{\mathrm{T}} \bar{\boldsymbol{x}}_k. \tag{3}$$

The bias term is included as weight b_0 in vector $\boldsymbol{b}$,

$$\boldsymbol{b} = \begin{bmatrix} b_0 \ \ldots \ b_{n_i} \end{bmatrix}^{\mathrm{T}} \text{ with } \bar{\boldsymbol{x}}_k = \begin{bmatrix} 1 & \boldsymbol{x}_k^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}. \tag{4}$$

Parameter n_i is the number of inputs fed to the network. DRBFs provide a natural extension from linear-to-nonlinear modelling and facilitate the inclusion of a priori linear modelling experience. They are particularly beneficial in sequential learning since the direct link (linear) term has global support and thus provides a degree of generalisation even when the network is in its infancy.

The remainder of the paper is organised as follows. Section 2 describes sequential learning for RANs and the pruning strategy developed by Yingwei et al. [19]. Section 3 introduces decomposed EKF and RLM for DRANs while Section 4 describes the minimal update alternatives. Simulation results evaluating the new training schemes are presented in Section 5 and finally Section 6 concludes the paper.

2. Sequential learning algorithms

Sequential learning algorithms differ from classical RBF training rules in that they combine new centre allocation with weight updating in one routine. The basic

algorithm structure is as follows:

$$y_k = \sum_{i=1}^{m} h_i \exp\left(-\frac{\|\boldsymbol{x}_k - \boldsymbol{c}_i\|^2}{\sigma_i^2}\right). \tag{5}$$

Let $(\boldsymbol{x}_k,\ y_k)$ be a new data point to be fitted by the RBF network in (5). The sequential learning growth criteria (which is based on [16]) for the network are

$$d_{nc} < 2\alpha\sigma_{nc} \tag{6}$$

and

$$\sum_{j=k-(M-1)}^{k} \frac{|e_j|}{M} > \bar{\varepsilon}_{\min}, \qquad e_j = d_j - y_j. \tag{7}$$

In (6) d_{nc}, defined as,

$$d_{nc} = \min_i \|\boldsymbol{x}_k - \boldsymbol{c}_i\|, \qquad i = 1, 2, \ldots m, \tag{8}$$

is the distance between input vector $\boldsymbol{x}_k$ and the centre of the nearest hidden neuron, ϕ_{nc} while σ_{nc} is the width of the nearest neuron. The desired output corresponding to input $\boldsymbol{x}_j$ is given by d_j while the scalar α (determines the Gaussian locality range) is usually 1.0. The error threshold parameter, $\bar{\varepsilon}_{\min}$, in (7) is computed as

$$\bar{\varepsilon}_{\min} = \max\left(\varepsilon_{\min}, \gamma^k \varepsilon_{\max}\right). \tag{9}$$

An average moving-window instantaneous error is used as a novel criterion instead of the instantaneous error e_k as the latter is vulnerable to outliers [12]. Scalars $\varepsilon_{\min}$, $\varepsilon_{\max}$ and $\gamma < 1$ are the user defined parameters.

If either (or both) criteria in (6) and (7) are not satisfied then all the network parameters are adapted to fit the new point using a recursive Gauss–Newton method, also known as the EKF algorithm [10]. This can be described as follows:

$$\boldsymbol{w}_k = \left[c_1^{\mathrm{T}}, \sigma_1, h_1, \ldots, c_m^{\mathrm{T}}, \sigma_m, h_m\right]^{\mathrm{T}}_{N_w \times 1}, \tag{10}$$

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + P_k \boldsymbol{\psi}_k e_k, \tag{11}$$

$$e_k = d_k - y_k, \tag{12}$$

$$\boldsymbol{\psi}_k = \frac{\partial}{\partial \boldsymbol{w}_k} y(\boldsymbol{x}_k, \boldsymbol{w}_k), \tag{13}$$

$$P_k = P_{k-1} - \frac{P_{k-1}\boldsymbol{\psi}_k \boldsymbol{\psi}_k^{\mathrm{T}} P_{k-1}}{1 + \boldsymbol{\psi}_k^{\mathrm{T}} P_k \boldsymbol{\psi}_k}. \tag{14}$$

Here N_w is the total number of parameters in the network. On the other hand if the growth criteria in (6) and (7) are satisfied then a new Gaussian basis function is assigned as follows:

$$\boldsymbol{c}_{m+1} = \boldsymbol{x}_k, \quad h_{m+1} = |e_k|, \quad \sigma_{m+1} = \beta\frac{d_{nc}}{2}, \tag{15}$$

$$P_k = \begin{bmatrix} P_{k-1} & 0 \\ 0 & I_{n_w \times n_w} \end{bmatrix}. \tag{16}$$

The scalar β is a user defined parameter (usually set to 1.0) which determines the degree of overlap between neurons and the vector dimension n_w is equal to the number of new parameters arising from the inclusion of the new Gaussian basis function. This form of sequential learning is known as RAN-EKF [10]. In Platt's [17] RAN algorithm the weight vector update in (11) reduces to

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} + \lambda_k \boldsymbol{\psi}_k e_k, \tag{17}$$

where λ_k is the learning rate which can be a constant or time varying. Meanwhile the new parameter assignments for the new basis function are simplified to (15) without the need for the second order weight covariance matrix P_k in (16).

2.1. Pruning strategy for sequential learning algorithms

The pruning strategy is based on Yingwei et al. [19] in which the Gaussian kernels which show the least contribution to the model output for the past M sample instants are eliminated. The pruning procedure can be summarised as follows.

- Compute the output of all the Gaussian kernel functions, $i = 1, 2, \ldots, m$.

$$r_{k(i)} = h_i \exp\left(-\frac{\|\boldsymbol{x}_k - \boldsymbol{c}_k\|^2}{\sigma_i^2}\right). \tag{18}$$

- Find the largest absolute Gaussian basis function output value

$$r_{\max} = \max\left(\left|r_{k(i)}\right|\right), \quad i = 1, 2, \ldots, m. \tag{19}$$

- Determine the normalised contribution factor for each basis function

$$\kappa_i = \left|\frac{r_{k(i)}}{r_{\max}}\right|, \quad i = 1, 2, \ldots, m. \tag{20}$$

- If $\kappa_j < \delta$ for M consecutive sample instances then prune the jth hidden neuron and reduce the dimensionality of $\boldsymbol{w}_k$, $\boldsymbol{\psi}_k$ and P_k accordingly.

Yingwei et al. [19] called the combination of this pruning strategy with RAN-EKF a minimal RAN (MRAN).

3. Decomposed training

The EKF covariance matrix, P_k involves $O(N_w^2)$ storage requirements and computational complexity, where $N_w = \dim(\boldsymbol{w}_k)$. As training proceeds, P_k grows

with the number of network parameters and quickly becomes a limiting factor in the use of EKF for on-line applications. However, by discarding the inter-neuron weight correlations, as proposed by Jankowski and Kadirkamanathan, P_k can be decomposed on a neuron by neuron basis leading to a much sparser matrix. Decomposition of the covariance matrix can be further extended and applied to DRANs where $\bar{P}_k$ (covariance matrix of DRAN) is also partitioned with respect to the direct-link and RBF parameters. This corresponds to an additional layer level decomposition and leads to the following reduced memory implementation.

$$\bar{P}_k \cong \begin{bmatrix} \left[P_K^{DL}\right] & 0 \\ 0 & \left[\tilde{P}_k\right]_{\tilde{N}_w \times \tilde{N}_w} \end{bmatrix}_{\bar{N}_w \times \bar{N}_w}, \tag{21}$$

$$\tilde{P}_k \cong \begin{bmatrix} \left[\tilde{P}_{k(1)}\right]_{n_w \times n_w} & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & \left[\tilde{P}_{k(m-1)}\right]_{n_w \times n_w} & \cdots \\ 0 & 0 & \cdots & \left[\tilde{P}_{k(m-1)}\right]_{n_w \times n_w} \end{bmatrix}_{\tilde{N}_w \times \tilde{N}_w}. \tag{22}$$

Here $\tilde{P}_{k(i)}$ represents the correlation matrix (or covariance matrix) for the ith neuron weights and P_k^{DL} is represents the covariance matrix of the direct link weights. The numerical complexity and memory requirements for the decomposed covariance matrix are $O\left(m \times n_w^2\right)$ compared to $O\left(m^2 \times n_w^2\right)$ for P_k in conventional full matrix second-order sequential learning. A graphical interpretation of the sparseness achieved is shown in Fig. 1.

The gradient vector can be equivalently decomposed as

$$\boldsymbol{\psi}_k = \left[\boldsymbol{\psi}_{k(1)}^{\mathrm{T}} \quad \boldsymbol{\psi}_{k(2)}^{\mathrm{T}} \quad \cdots \quad \boldsymbol{\psi}_{k(m)}^{\mathrm{T}}\right]^{\mathrm{T}}. \tag{23}$$

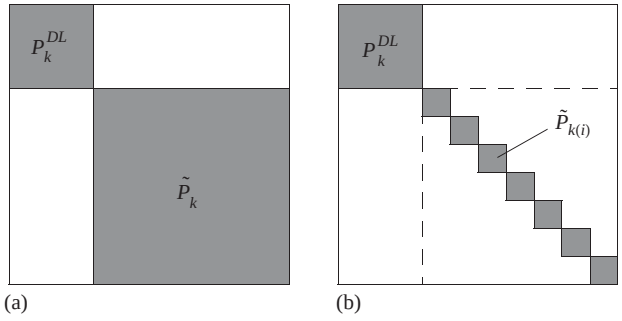

Fig. 1. Graphical interpretation of layer-level and neuron-level decomposition: (a) Layer-level decomposition; (b) Neuron-level decomposition.

There is also the potential for parallel implementation [15] as the weight vector update is decomposed into m sub-algorithms:

$$\boldsymbol{w}_{k+1(i)} = \boldsymbol{w}_{k(i)} + \tilde{P}_{k(i)}\boldsymbol{\psi}_{k(i)}e_k, \tag{24}$$

$$\boldsymbol{w}_{k(i)} = \left[\boldsymbol{c}_i^{\mathrm{T}}\sigma_i h_i\right]^{\mathrm{T}}, \qquad \boldsymbol{\psi}_{k(i)} = \left[\frac{\partial y_k}{\partial \boldsymbol{w}_{k(i)}}\right]_{n_w x 1}, \tag{25}$$

$$\tilde{P}_{k(i)} = \tilde{P}_{k-1(i)} - \frac{\tilde{P}_{k-1(i)}\boldsymbol{\psi}_{k(i)}\boldsymbol{\psi}_{k(i)}^{\mathrm{T}}\tilde{P}_{k-1(i)}}{1+\boldsymbol{\psi}_{k(i)}^{\mathrm{T}}\tilde{P}_{k(i)}\boldsymbol{\psi}_{k(i)}}, \qquad i = 1, 2, \ldots, m \tag{26}$$

The direct link weights are updated based on the following procedure

$$\boldsymbol{b}_k = \boldsymbol{b}_{k-1} + P_k^{\mathrm{DL}}\boldsymbol{x}_k e_k, \tag{27}$$

$$P_k^{\mathrm{DL}} = P_{k-1}^{\mathrm{DL}} - \frac{P_{k-1}^{\mathrm{DL}}\boldsymbol{x}_k\boldsymbol{x}_k^{\mathrm{T}}P_{k-1}^{\mathrm{DL}}}{1+\boldsymbol{x}_k^{\mathrm{T}}P_{k-1}^{\mathrm{DL}}\boldsymbol{x}_k}. \tag{28}$$

The resulting topology–algorithm combination, a direct-link resource-allocating network with decomposed EKF weight adaptation, will be referred to as DRAN-DEKF.

3.1. Decomposed RLM (DRAN-DRLM)

A major advantage of decomposing $\bar{P}_k$ (for RANs or DRANs) is the size of each of the covariance sub-matrices $\tilde{P}_{k(i)}$ is invariant during training in comparison to P_k (for RAN-EKF) which changes dimension from one iteration to the next. This facilitates the application of the powerful decomposed recursive Levenberg–Marquardt training algorithm [3] to sequential learning of DRBFs (DRAN-DRLM). The main difference between DRLM and DEKF is that in DRLM a regularisation term ρ is added to one of the diagonal element of $\tilde{P}_{k(i)}$ (of DEKF) at each sample instant. The routine for updating each $\tilde{P}_{k(i)}$ using DRLM is given as follows:

$$S_{k(i)} = \Lambda + \Omega_{k(i)}^{\mathrm{T}}\tilde{P}_{k-1(i)}\Omega_{k(i)}, \tag{29}$$

$$\tilde{P}_{k(i)} = \tilde{P}_{k-1(i)} - \tilde{P}_{k-1(i)}\Omega_{k(i)}S_{k(i)}^{-1}\Omega_{k(i)}^{\mathrm{T}}\tilde{P}_{k-1(i)}, \tag{30}$$

$$\Omega_{k(i)}^{\mathrm{T}} = \begin{bmatrix} & & \boldsymbol{\psi}_{k(i)}^{\mathrm{T}} & & \\ 0 & \ldots & 1 & \ldots & 0 \end{bmatrix}, \quad \Lambda = \begin{bmatrix} 1 & 0 \\ 0 & \rho \end{bmatrix}^{-1} \quad \text{and} \quad i = 1, \ldots, m, \tag{31}$$

where $\Omega_{k(i)}$ is a $n_w \times 2$ matrix whose second column has only one non zero element located at $k_{r(i)} \bmod n_w + 1$. The scalar $k_{r(i)}$ is a counter that keeps track of the age of the decomposed kernel function ϕ_i and is used to determine the location of the non-zero element in $\Omega_{k(i)}$ at each iteration. It is initialised to 1 when the ith basis function is created and incremented every weight update thereafter. If the ith basis function is

being removed, during pruning, the scalar $k_{r(i)}$ will be discarded automatically. A global optimisation of the linear weights of the direct-link network is implemented as in and (27) and (28).

4. Minimal update algorithms

Minimal weight update algorithms are motivated by memory and time constraints, particularly in real-time modelling and control applications. In early work on RANs with minimal update by Bomberger and Seborg [5] the weight update rule was restricted to the linear connections of RBF neurons which had significant influence over the model output at a given time. Junge and Unbehauen [9] and Li et al. [11] proposed limiting weight updates to the parameters of a single *winner neuron* to further improve computational efficiency. Here the minimal update *winner neuron* concept is extended to DRAN-DEKF and DRAN-DRLM training as follows.

4.1. Minimal decomposed algorithm

Rather than adapting all the decomposed network parameters as in (25)–(26), the minimal update DRAN-DEKF algorithm (DRAN-mDEKF) updates only the weights of the neuron whose centre is closest to the current input vector $\boldsymbol{x}_k$. Thus:

$$c = \arg\left\{\min_i \|\boldsymbol{x}_k - \boldsymbol{c}_i\|\right\}, \qquad i = 1, 2, \ldots, m \tag{32}$$

$$\boldsymbol{w}_{k(c)} = \mathbf{w}_{k-1(c)} + \tilde{P}_{k(c)}\boldsymbol{\psi}_{k(c)}e_k, \tag{33}$$

$$\boldsymbol{w}_{k(c)} = \begin{bmatrix}\boldsymbol{c}_c^{\mathrm{T}} & \sigma_c & h_c\end{bmatrix}^{\mathrm{T}}, \qquad \boldsymbol{\psi}_{k(c)} = \left[\frac{\partial y_k}{\partial \boldsymbol{w}_{k(c)}}\right]_{n_w \times 1}, \tag{34}$$

$$\tilde{P}_{k(c)} = \tilde{P}_{k-1(c)} - \frac{\tilde{P}_{k-1(c)}\boldsymbol{\psi}_{k(c)}\boldsymbol{\psi}_{k(c)}^{\mathrm{T}}\tilde{P}_{k-1(c)}}{1 + \boldsymbol{\psi}_{k(c)}^{\mathrm{T}}\tilde{P}_{k(c)}\boldsymbol{\psi}_{k(c)}}. \tag{35}$$

Alternatively, covariance matrix $\tilde{P}_{k(c)}$ can be estimated using DRLM and updated as follows:

$$S_{k(c)} = \Lambda + \Omega_{k(c)}^{\mathrm{T}}\tilde{P}_{k-1(c)}\Omega_{k(c)}, \tag{36}$$

$$\tilde{P}_{k(c)} = \tilde{P}_{k-1(c)} - \tilde{P}_{k-1(c)}\Omega_{k(c)}S_{k(c)}^{-1}\Omega_{k(c)}^{\mathrm{T}}\tilde{P}_{k-1(c)}, \tag{37}$$

$$\Omega_{k(c)}^{\mathrm{T}} = \begin{bmatrix} & \boldsymbol{\psi}_{k(c)}^{\mathrm{T}} & \\ 0 \quad \ldots & 1 & \ldots \quad 0 \end{bmatrix} \qquad \text{and} \quad \Lambda = \begin{bmatrix} 1 & 0 \\ 0 & \rho \end{bmatrix}^{-1}, \tag{38}$$

where $\Omega_{k(c)}$ is a $n_w \times 2$ matrix whose second column has only one non-zero element located at $k_{r(c)} \bmod n_w + 1$. Counter $k_{r(c)}$ is incremented every time $\boldsymbol{w}_{k(c)}$ is updated. The direct link network weights are unaffected by this modification and are updated

using (27) and (28) as before. The resulting algorithm is referred to as minimal DRAN-DRLM (DRAN-mDRLM).

5. Simulation results

5.1. Benchmark problems

Three benchmark modelling problems are used in the evaluation of the proposed DRBF sequential learning algorithms.

Time series prediction benchmark: the objective here is to model a simulated nonlinear time series described by the nonlinear difference equation

$$y_k = \left[0.8 - 0.5\exp\left(-y_{k-1}^2\right)\right]y_{k-1} - \left[0.3 - 0.9\exp\left(-y_{k-1}^2\right)\right]y_{k-1}^2 + 0.1\sin\left(\pi y_{k-1}\right). \tag{39}$$

The neural model is based on the nonlinear autoregressive (NAR) model structure, that is

$$y_k = f\left(y_{k-1}, y_{k-2}\right) + \eta_k, \tag{40}$$

where the noise η_k is a Gaussian white noise sequence with zero mean and variance 0.01.

Coupled-tank system benchmark: modelling the nonlinear dynamics of a two tank system [7] (shown in Fig. 2) is used as the second benchmark. The two tanks are connected by a short pipe and the amount of liquid flowing into the tanks is regulated by a control valve. The identification problem considered here is the prediction of liquid level (h_2) of the second tank (Tank 2) given the input flow (q_1) to the first tank (Tank 1). The network input vector for the DRBF model is

$$\boldsymbol{x}_k = \begin{bmatrix} h_{2(k-1)} & q_{1(k-1)} \end{bmatrix} \tag{41}$$

and the output is $h_{2(k)}$. Complete details of the benchmark can be found in [7].

Continuous stirred tank reactor benchmark: Continuous stirred tank reactors (CSTR) are highly nonlinear systems and as such useful benchmarks for neural network modelling techniques. The CSTR considered here is a single-input–single-output (SISO) one, where the output is the product concentration, C and the input is the coolant flow rate, q. The chemical reaction which produces the compound inside the reactor is an exothermic process, which raises the temperature, and reduces the reaction rate. The objective is to control the measured concentration of the product, C, by manipulating the coolant flow rate, q. The CSTR equations are given by

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{q_{if}}{V}\left(T_{if} - T\right) + K_1 C\exp\left(\frac{-E}{R\cdot T}\right) + K_2 q\cdot\left[1 - \exp\left(-\frac{K_3}{q}\right)\right]\cdot\left(T_{\mathrm{cf}} - T\right), \tag{42}$$

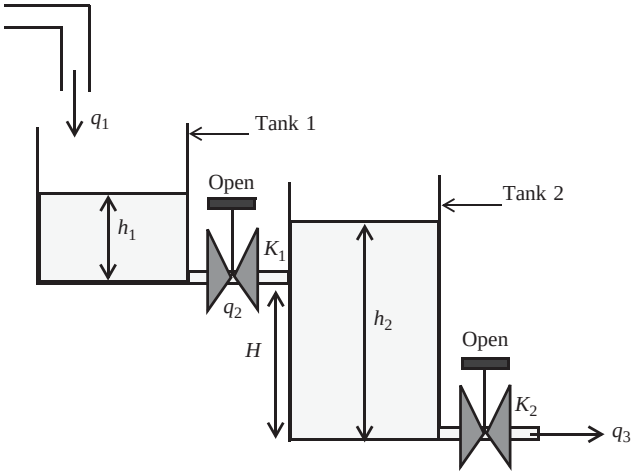

Fig. 2. Split level two-tank system.

$$\frac{\mathrm{d}C}{\mathrm{d}t} = \frac{q_{if}}{V}(C_{if} - C) - K_o C \exp\left(\frac{-E}{R \cdot T}\right). \tag{43}$$

Details of the equation constants, nominal setting and operating conditions for the CSTR benchmark can be found in [13]. Here the process is identified using a neural model assuming a second order nonlinear autoregressive with external input (NARX) model structure. The network input vector is thus of the form

$$\boldsymbol{x}_k = \begin{bmatrix} C_{k-1} & C_{k-2} & q_{k-1} & q_{k-2} \end{bmatrix} \tag{44}$$

and the output is the product concentration at the kth sample instant, C_k.

5.2. Performance evaluation

To get an indication of the global modelling quality of a DRAN at a particular iteration of the sequential learning process model performance is measured in terms of the percentage normalised mean squared error (%NMSE),

$$J(\boldsymbol{w}_k) = \frac{100}{p} \sum_{n=1}^{N_v} (y(\boldsymbol{x}_n, \boldsymbol{w}_k) - d_n)^2 \Bigg/ \sum_{n=1}^{N_v} d_n^2, \tag{45}$$

computed over an N_v pattern representative test data set, $(\boldsymbol{x}_n, d_n)$. Figs. 3(a) and 4(a) show the typical evolution of this performance index for the coupled-tank and CSTR benchmark problems with each of the four sequential learning algorithms under consideration. The learning curves for RAN-EKF are also included for comparison. Fig. 3(b) also shows the typical growth of a DRAN during training while Fig. 4(b) demonstrates the parallel modelling performance achievable with a minimal update-DRLM trained DRAN for the CSTR benchmark.

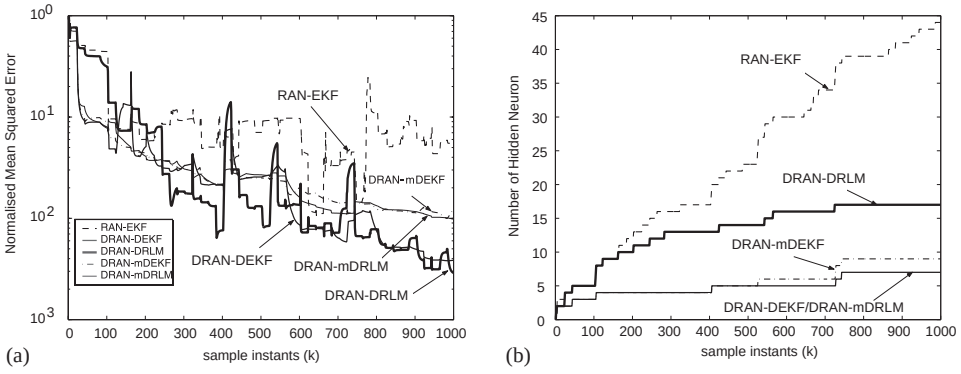

Fig. 3. Sequential learning performance on coupled tank system (without pruning). (a) %NMSE learning curve for two tank system; (b) Number of hidden neurons.

Tables 1–3 summarise the performance of each of the sequential learning algorithms discussed in the paper (with and without pruning) on the three benchmarks following 1000 iterations of training. For comparison purposes results are also included for RAN-EKF and DRAN-EKF. The recorded performance indices are the mean NMSE (E [$J(\boldsymbol{w}_k)$]) computed over the last 500 iterations, the final number of hidden neurons (m) and the corresponding number of network parameters (N_w). The user defined algorithm parameters were the same for each problem and are as given in Table 4. Finally, Table 5 summarises the approximate memory and computational requirements of each algorithm.

These results show that, in addition to providing significant savings in memory and computational requirements, the decomposed EKF and RLM algorithms consistently yield better NMSE performance than DRAN-EKF. The results also show that the minimal update implementations are promising alternatives to DRLM and DEKF as they have an even lower computational cost yet achieve comparable performance. When used in conjunction with pruning the proposed algorithms maintain their superiority while at the same time producing parsimonious networks.

The decomposed algorithms significantly outperform conventional RAN-EKF and MRAN, but a comparison with the results for DRAN-EKF reveals that this is principally due to the benefits offered by the DRAN architecture.

An examination of the NMSE performance of the different EKF and RLM implementations shows that RLM is marginally superior to EKF. This is consistent with other comparisons of the algorithms reported in the literature [3,14].

The superior NMSE achieved by the decomposed training algorithms may seem remarkable given that they discard a significant proportion of the curvature information in computing the weight updates. However, this is in fact the reason for their success. The stochastic nature of the covariance matrix as it is computed recursively over several iterations and the fact that it varies as a function of RAN

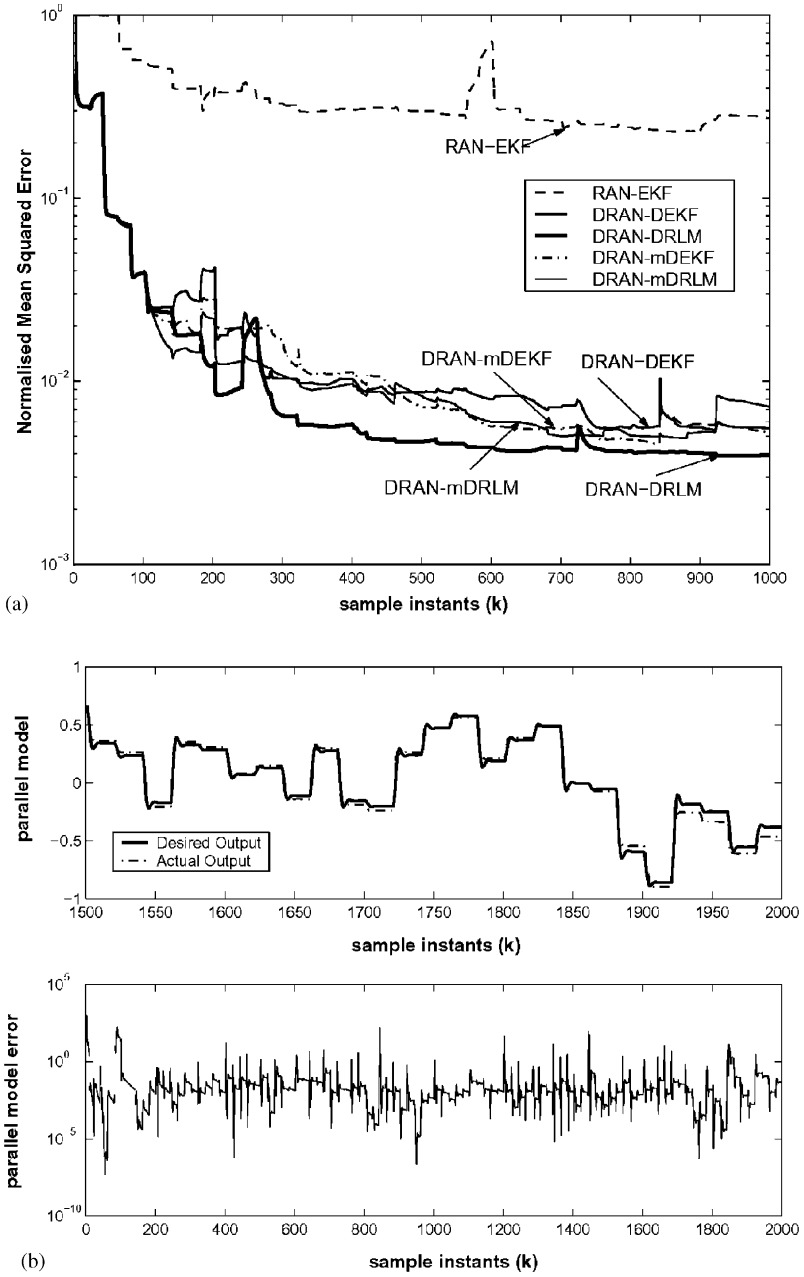

Fig. 4. Sequential learning performance on CSTR benchmark (without pruning). (a) %NMSE learning curve for CSTR benchmark; (b) Parallel model obtain using DRAN-mDRLM.

Table 1
Performance of sequential learning algorithms on the time series benchmark

	Without pruning			With pruning		
	E(NMSE)	*m*	N_w	*E(NMSE)*	*m*	N_w
RAN-EKF[a]	0.0300	52	313	0.0289	15	91
DRAN-DEKF	0.0320	6	27	0.0320	6	27
DRAN-DRLM	0.0191	7	31	0.166	7	31
DRAN-mDEKF	0.0223	6	27	0.0223	6	27
DRAN-mDRLM	0.0102	7	31	0.0102	7	31
DRAN-EKF	0.0293	13	81	0.0293	13	81

[a]RAN-EKF with pruning is usually reffered to as MRAN.

Table 2
Performance of sequential learning algorithms on the two tank benchmark problem

	Without pruning			With pruning		
	E(NMSE)	*m*	N_w	*E(NMSE)*	*m*	N_w
RAN-EKF[a]	0.0628	44	265	0.1189	41	247
DRAN-DEKF	0.0099	7	45	0.0239	5	33
DRAN-DRLM	0.0096	17	105	0.0082	6	39
DRAN-mDEKF	0.0151	9	57	0.0282	4	27
DRAN-mDRLM	0.0146	7	45	0.0309	4	27
DRAN-EKF	0.0450	7	45	0.0450	7	45

[a]RAN-EKF with pruning is usually reffered to as MRAN.

Table 3
Performance of sequential learning algorithms on the CSTR benchmark

	Without pruning			With pruning		
	E(NMSE)	*m*	N_w	*E(NMSE)*	*m*	N_w
RAN-EKF[a]	0.2854	30	301	0.2910	21	211
DRAN-DEKF	0.0069	9	95	0.0020	2	25
DRAN-DRLM	0.0042	9	95	0.0027	3	35
DRAN-mDEKF	0.0057	9	95	0.0020	2	25
DRAN-mDRLM	0.0062	9	95	0.0026	3	35
DRAN-EKF	0.0090	19	195	0.0095	12	125

[a]RAN-EKF with pruning is usually reffered to as MRAN.

Table 4
User defined parameters

Growth Criteria parameters	$\varepsilon_{\min} = 0.3 \quad \varepsilon_{\max} = 0.5 \quad \gamma = 0.98$
Pruning method parameters	$M = 30 \quad \delta = 0.0001$
Others	$\rho = 1 \quad \lambda = 1$

Table 5
Comparison of the memory requirements and computation for various DRBF sequential learning algorithms

Algorithms	Memory requirement	Computation load
RAN-EKF (MRAN)	$O(N_w^2)$	$O(3N_w^2)$
DRAN-EKF	$O\left((N_w+\bar{N}_i)^2\right)$	$O\left(3(N_w+\bar{N}_i)^2\right)$
DRAN-DEKF/DRLM	$O\left(mn_w^2+\bar{N}_i^2\right)$	$O\left(3mn_w^2+3\bar{N}_i^2\right)$
DRAN-mDKEF/mDRLM	$O\left(mn_w^2+\bar{N}_i^2\right)$	$O\left(3n_w^2+3\bar{N}_i^2\right)$

Note: N_w is the total number of weights, n_w is the number of weights for one hidden neuron, $\bar{N}_i$ is the number of inputs including the bias, m is the number of hidden neurons.

parameters makes it ill-conditioned and unreliable. Focusing on the near diagonal curvature information is thus beneficial in this respect. Furthermore, decomposed training can be viewed as an approximation to separable recursive least squares which in turn is an approximation to off-line separable nonlinear least squares training. One of the properties of separable nonlinear least squares is that the resulting covariance matrix is better conditioned than the corresponding matrix in conventional nonlinear least squares [18]. Empirical studies have shown that this also holds true for separable recursive least squares [4]. Thus, it is likely that the decomposed and minimal update algorithms also benefit from being better conditioned.

6. Conclusions

Two computationally efficient learning algorithms have been developed for DRANs, a generalisation of RANs where the RBF topology is modified to include direct links between the inputs and the output node. The first is obtained by partitioning the network in terms of the direct-link and RBF parameters and decomposing the RBF parameter covariance matrix on a neuron by neuron basis. The second is a variant of the first in which only the parameters of a winner neuron are updated at each iteration to reduce the computational burden. Results presented for implementations, with and without pruning, show that the decomposed sequential learning approaches consistently outperform DRAN-EKF in terms of the final percentage NMSE achieved. In addition, the algorithms use less memory, require less computation time per iteration and produce more parsimonious networks.

The minimal update algorithms (DRAN-mDEKF and DRAN-mDRLM) are especially noteworthy as they achieve a substantial reduction in computational load compared to DRAN-mDEKF and DRAN-mDRLM without sacrificing NSME performance.

While originally motivated by the search for computationally efficient algorithms, the superior learning capabilities of DRLM and mDRLM make them the first choice algorithms for training DRANs even when memory requirements and real-time constraints are not an issue.

References

[1] V.S. Asirvadam, S.F. McLoone, G.W. Irwin, Fast and efficient sequential learning algorithms using direct-link RBF networks, IEEE Workshop on Neural Networks for Signal Processing, vol. XIII, Toulouse, France, September 2003, pp. 849–858.
[2] V.S. Asirvadam, S.F. McLoone, G.W. Irwin, Minimal update sequential learning algorithms for RBF networks, United Kingdom Automatic Control Conference (UKACC)-Control, Sheffield, 10–12 September 2002, pp. 71–76.
[3] V.S. Asirvadam, S.F. McLoone, G.W. Irwin, Parallel and separable recursive levenberg–marquardt training algorithm', IEEE Workshop on Neural Networks for Signal Processing XII, Martigny, Switzerland, 4–6 September 2002, pp. 129–138.
[4] V.S. Asirvadam, S.F. McLoone, G.W. Irwin, Separable recursive training algorithms for feedforward neural networks, IEEE World Congress on Computational Intelligence, Honolulu, Hawaii, 12–17 May 2002, pp. 1212–1217.
[5] J.D. Bomberger, D.E. Seborg, On-line updating of radial basis function model, Proceeding IFAC Symposium on Nonlinear Control System Design, 1995, pp. 175–179.
[6] D.S. Broomhead, D. Lowe, Multivariable functional interpolation and adaptive networks, Complex System 2 (1988) 321–355.
[7] T. Heckenthaler, S. Engell, Robust approx imately time-optimal fuzzy control of a two-tank system, IEEE Control Systems Mag 14 (1994) 24–30.
[8] N. Jankowski, V. Kadirkamanathan, Statistical control of growing and pruning in RBF-like neural networks, Third Conference on Neural Network and Their Applications, Kule, Poland, 1997, pp. 663–670.
[9] T.F. Junge, H. Unbehauen, On-line identification of a continuous stirred tank reactor using structurally adaptive radial basis function networks, SYSID 97, Proceedings of the IFAC Symposium Fukuaka, 1997, pp. 723–728.
[10] V. Kadirkamanathan, M. Niranjan, A function estimation approach to sequential learning with neural network, Neural Comput. 5 (1993) 723–728.
[11] Y. Li, N. Sundarajan, P. Saratchandran, Analysis of minimal radial basis function network algorithm for real-time identification of nonlinear dynamic systems, IEE Proc.—Control Theory Appl. 147 (4) (2000) 476–484.
[12] A. McLachlan, D. Lowe, Tracking of non-stationary time-series using resource allocation RBF networks, Proceedings of the 13th European Meeting on Cybernetics and System Research, 1996, pp. 1066–1071.
[13] J.D. Morningred, B.E. Paden, D.E. Seborg, D.A. Mellichamp, An adaptive nonlinear predictive controller, Chem. Eng. Sci. 47 (4) (1992) 755–762.
[14] S.H. Ngia, J. Sjöberg, Efficient training of neural nets for nonlinear adaptive filtering using a recursive Levenberg–Marquardt algorithm, IEEE Trans. Signal Process. 48 (7) (2000) 1915–1927.
[15] S.F. McLoone, G.W. Irwin, Fast parallel off-line training of multilayer perceptrons, IEEE Trans. Neural Networks, 8 (3) (1997) 646–653.
[16] S.F. McLoone, Neural network identification of avr loop dynamics: a comparison between MLPS and RBFS, 35th Universities Power Engineering Conference (UPEC), September 2000.
[17] J. Platt, A resource-allocating network for function interpolation, Neural Comput. 3 (1991) 213–225.
[18] J. Sjoberg, M. Viberg, Separable non-linear least-squares minimization - possible improvements for neural net fitting, Proceedings of the IEEE Workshop in Neural Networks for Signal Processing, Amelia Island Plantation, FL, 24–26 September 1997.

[19] L. Yingwei, N. Sundarajan, P. Saratchandran, A sequential learning scheme for function approximation using minimal radial basis function neural networks, Neural Comput. 9 (1997) 461–478.

Dr. Vijanth Sagayan Asirvadam is from an old mining city of Malaysia called Ipoh. He studied at the University of Putra, Malaysia for the Bachelor Science (Hon) majoring in Statistics and graduated in April 1997 before leaving for Queen's University Belfast to do his Masters where he received his Master's Science degree in Engineering Computation with a Distinction. He has worked briefly as a Lecturer in a private higher institution and as a System Engineer in Multimedia University, Malaysia. He later joined the Intelligent Systems and Control Research Group at Queen's University Belfast in November 1999 where he completed his Doctorate (Ph.D.) research in Online and Constructive Neural Learning methods. He is currently a Lecturer in the Faculty of Information Science and Technology at Multimedia University, Malaysia. His research interests include neural network and statistical learning for black-box modelling, model validation and data mining.

Dr. Seán McLoone, a native of Glenties, Co. Donegal, Ireland, received his third-level education at the Queen's University of Belfast, where he was awarded a Master of Engineering Degree with Distinction in Electrical and Electronic Engineering in 1992 followed by a Ph.D. in Control Engineering in 1996. Following a period as a postdoctoral research fellow he was appointed as a Lecturer in Control Engineering at the University in January 1998. In September 2002, he joined the Department of Electronic Engineering at NUI Maynooth, Co. Kildare, Ireland.

At a professional level, Dr. McLoone is a Chartered Engineer, a member of the Institute of Electrical Engineers (IEE), the Institute of Electrical and Electronic Engineers (IEEE) and the Institution of Engineers of Ireland (IEI). He is the current secretary of the UK and Republic of Ireland (UKRI) Section of the IEEE, chairman of the IEEE UKRI Control & Communications (Ireland) joint chapter and a member of the IFAC Technical Committee on Cognition and Control.

His research interests include optimisation and parallel computing for neural, wavelet and local model network training, linear and non-linear system identification techniques applied to dynamical system modelling, time series prediction and signal processing, multiple model approaches to modelling and control, fault diagnosis and statistical process control.

George Irwin is a Professor of Control and Director of the Virtual Engineering Centre at Queen's University, Belfast, UK. His current research interests include identification and control using neural networks, fuzzy neural systems, local model networks, and multivariate statistics. Much of his work involves industrial collaboration and he was a Director of Anex6 Ltd., a spin out company from his group specialising in process monitoring. He has published over 300 research papers and 6 edited books. Prizes include four IEE Premiums, a Best Paper award from the Czech Academy of Sciences and the 2002 Honeywell International Medal from the UK Institute of Measurement and Control. He is a former Editor-in-Chief of the IFAC journal Control Engineering Practice and current chair of the UK Automatic Control Council. Prof. Irwin has been elected Fellow of the Royal Academy of Engineering and Member of the Royal Irish Academy. He is a chartered engineer, an IEEE fellow, a fellow of the IEE and a fellow of the Institute of Measurement and Control.