# Real-Time Construction of Neural Networks

Kang Li[1], Jian Xun Peng[1], and Minrui Fei[2]

[1] School of Electronics, Electronic Engineering & Computer Science
Queen's University Belfast, Ashby Building, Stranmillis Road, Belfast BT9 5AH, UK
{K.Li, J.Peng}@qub.ac.uk
[2] Shanghai Key Laboratory of Power Station Automation Technology,
School of Mechatronics and Automation, Shanghai University, Shanghai 200072, China

**Abstract.** A stepwise two-stage algorithm is proposed for real-time construction of generalized single-layer networks (GSLNs). The first stage of this algorithm generates a network using a forward selection procedure, which is then reviewed at the second stage to replace insignificant neural nodes. The main contribution of this paper is that these two stages are performed within one regression context using Cholesky decomposition, leading to significantly neural network performance and concise real-time network construction procedures.

## 1 Introduction

The generalized single-layer networks (GSLN) represent a large class of flexible and efficient structures due to their excellent approximating capabilities [1][4]. A GSLN is a linear combination of some basis functions that are arbitrary (usually nonlinear) functions of the inputs. Examples are RBF networks and Volterra networks.

A critical issue in the application for GSLNs is that a large number of candidate basis functions may have to be considered initially, from which only small subset is selected to represent the data by minimizing a cost function. This however becomes extremely difficult if one has to enumerate all combinations to find the best subset, part of which is often referred to as the curse of dimensionality problem in the literature. To overcome this difficulty, forward subset selection methods seem to be one of very few feasible approaches [2][3][5][6]. Forward subset selection algorithms select the best basis function each time by minimizing the cost function, and this procedure is repeated until the desired number of, say $n$, basis functions have been selected. If $n$ is unknown a 'prior', some selection criteria could be applied to stop the network construction, such as the Akaike's information criteion (AIC) [7].

However, the major problem with forward approaches is that the resultant network is not necessarily efficient [6][8]. In this paper, a stepwise algorithm is proposed for real time construction of GLSNs based on the Cholesky decomposition. In the proposed method, the network is generated using a forward subset selection procedure, which is then reviewed, and insignificant neurons (basis functions) are replaced, resulting in a network of significantly improved performance.

S. Kollias et al. (Eds.): ICANN 2006, Part I, LNCS 4131, pp. 140 – 149, 2006.

## 2 Problem Formulation

Suppose $M$ candidate basis functions $\{\phi_i(t), i = 1,2,\cdots,M\}$, and $N$ samples are used for network construction and training, producing to the following matrices

$$\mathbf{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \cdots, \boldsymbol{\phi}_{\mathrm{M}}],\ \boldsymbol{\phi}_{\mathrm{i}} = [\phi_{\mathrm{i}}(1), \phi_{\mathrm{i}}(2), \cdots, \phi_{\mathrm{i}}(\mathrm{N})]^{\mathrm{T}}, \mathrm{i} = 1,2,\cdots,\mathrm{M} \tag{1}$$

If one needs to select $n$ significant basis functions, denoted as $\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n$, which form a selected *regression matrix*

$$\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n] \tag{2}$$

and produce the network output of

$$\mathbf{y} = \mathbf{P}\boldsymbol{\theta} + \mathbf{e} \tag{3}$$

which best fits the data in the sense that

$$J = \|\mathbf{\Lambda e}\| = \|\mathbf{\Lambda}(\mathbf{y} - \mathbf{P}\boldsymbol{\theta})\| \to \min \tag{4}$$

where $\mathbf{y} = [y(1), y(2), \cdots, y(N)]^{\mathrm{T}}$ is the vector of the target outputs; $\|\bullet\|$ denotes the 2-norm of a vector; $\boldsymbol{\theta}$ is the output weights; $\mathbf{\Lambda}$ is the diagonal *weighting matrix*, $\mathbf{\Lambda} = diag(\lambda(1), \cdots, \lambda(N))$; $J$ is the *cost function* which is the sum of the squared weighted errors (SSWE). If $\mathbf{P}$ is fully column ranked, optimum estimation of the output weights is given by

$$\boldsymbol{\theta} = [(\mathbf{\Lambda P})^T(\mathbf{\Lambda P})]^{-1}(\mathbf{\Lambda P})^T(\mathbf{\Lambda y}) \tag{5}$$

There are $M!/(n!/(M-n)!)$ possible combinations. Practically forward subset selection is the most popular approach.

## 3 Forward Selection

The forward approach selects a basis function each time, and this procedure repeats for $n$ times to produce the network of $n$ basis functions. Obviously a series of intermediate neural networks are generated along the process.

Denote the regression matrices of the intermediate network of $k$ basis functions as

$$\mathbf{P}_k = [\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_k], k = 1,2,\cdots,n \tag{6}$$

The corresponding cost function becomes

$$J(\mathbf{P}_k) = \mathbf{y}^T\mathbf{\Lambda}^2\mathbf{y} - \mathbf{y}^T\mathbf{\Lambda}^2\mathbf{P}_k(\mathbf{P}_k^T\mathbf{\Lambda}^2\mathbf{P}_k)^{-1}\mathbf{P}_k^T\mathbf{\Lambda}^2\mathbf{y} \tag{7}$$

where $\mathbf{\Lambda}^2 = \mathbf{\Lambda}^T\mathbf{\Lambda}$. If $\mathbf{\Lambda P}_k$ is of full column rank, then $\mathbf{W} = \mathbf{P}_k^T\mathbf{\Lambda}^2\mathbf{P}_k = [w_{i,j}]_{k\times k}$ is symmetric and positive definite, and can be decomposed as

$$\mathbf{W} = \mathbf{P}_k^T \mathbf{\Lambda}^2 \mathbf{P}_k = \tilde{\mathbf{A}}^T \mathbf{D} \tilde{\mathbf{A}} \tag{8}$$

where $\mathbf{D} = diag(d_1, \cdots, d_k)$ is a diagonal matrix and $\tilde{\mathbf{A}} = [\tilde{a}_{i,j}]_{k\times k}$ is a unity upper triangular matrix. Define

$$\mathbf{A} \triangleq \mathbf{D}\tilde{\mathbf{A}} = [a_{i,j}]_{k\times k}, a_{i,j} = \begin{cases} 0, & j < i \\ d_i \tilde{a}_{i,j} & j \geq i \end{cases} \tag{9}$$

where $\tilde{a}_{i,i} = 1$ and $d_i = a_{i,i}$ for $i = 1, \cdots, k$.

From (8), it gives

$$a_{i,j} = w_{i,j} - \sum_{s=1}^{i-1} a_{s,i} a_{s,j} / a_{s,s}, i = 1, \cdots, k, \; j = i, \cdots, k \tag{10}$$

Define

$$\mathbf{a}_y \triangleq \mathbf{A}\boldsymbol{\theta} = \mathbf{D}\tilde{\mathbf{A}}\boldsymbol{\theta} = [a_{1,y}, \cdots, a_{k,y}]^{\mathrm{T}} \tag{11}$$

$$\mathbf{w}_y \triangleq \mathbf{P}_k^{\mathrm{T}} \mathbf{\Lambda}^2 \mathbf{y} = [w_{1,y}, \cdots, w_{k,y}]^{\mathrm{T}} \tag{12}$$

Then left-multiply $\mathbf{W} = \mathbf{P}_k^{\mathrm{T}} \mathbf{\Lambda}^2 \mathbf{P}_k$ on both sides of (5) for $\mathbf{P} = \mathbf{P}_k$, substituting (8) gives

$$\tilde{\mathbf{A}}^{\mathrm{T}} \mathbf{a}_y = \mathbf{w}_y \tag{13}$$

As $\tilde{\mathbf{A}}$ is a unity upper triangular matrix, using (9), $\mathbf{a}_y$ could be computed as

$$a_{i,y} = w_{i,y} - \sum_{i=1}^{k-1} a_{s,i} a_{s,y} / a_{s,s}, i = 1, \cdots, k \tag{14}$$

Substituting (8) into (7) and noting (13) gives

$$J(\mathbf{P}_k) = \mathbf{y}^{\mathrm{T}} \mathbf{\Lambda}^2 \mathbf{y} - \mathbf{a}_y^{\mathrm{T}} \mathbf{D}^{-1} \mathbf{a}_y = \mathbf{y}^{\mathrm{T}} \mathbf{\Lambda}^2 \mathbf{y} - \sum_{i=1}^{k} a_{i,y}^2 / a_{i,i} \tag{15}$$

Suppose that one more basis function, denoted as $\mathbf{p}_{k+1}$, is selected, then

$$J(\mathbf{P}_{k+1}) = \mathbf{y}^{\mathrm{T}} \mathbf{\Lambda}^2 \mathbf{y} - \sum_{i=1}^{k+1} a_{i,y}^2 / a_{i,i} \tag{16}$$

The reduction in the cost function due to the new basis function $\mathbf{p}_{k+1}$ is

$$\Delta J_{k+1}(\mathbf{p}_{k+1}) = J(\mathbf{P}_k) - J(\mathbf{P}_{k+1}) = a_{k+1,y}^2 / a_{k+1,k+1} \tag{17}$$

where $\mathbf{P}_{k+1} = [\mathbf{P}_k, \mathbf{p}_{k+1}]$.

According to (15), the elements $a_{i,j}$ will not change as the new basis function $\mathbf{p}_{k+1}$ is introduced. To minimize the cost function given previously selected basis functions $\mathbf{p}_1, \cdots, \mathbf{p}_k$ are fixed, one needs to select a basis function from the rest candidates which maximizes $\Delta J_{k+1}$

$$\min\{J([\mathbf{P}_k, \boldsymbol{\phi}])\} = J(\mathbf{P}_k) - \max\{\Delta J_{k+1}(\boldsymbol{\phi})\}, \quad s.t. \quad \boldsymbol{\phi} \in \{\boldsymbol{\phi}_{k+1}, \cdots, \boldsymbol{\phi}_M\} \tag{18}$$

where $\{\boldsymbol{\phi}_{k+1},\cdots,\boldsymbol{\phi}_M\}$ denotes the candidate basis function pool from the full regression matrix $\boldsymbol{\Phi}$, i.e. $\boldsymbol{\Phi}=[\mathbf{p}_1,\mathbf{p}_2,\cdots,\mathbf{p}_k,\boldsymbol{\phi}_{k+1},\cdots,\boldsymbol{\phi}_M]$.

To minimize (18), $\tilde{\mathbf{A}}$, $\mathbf{A}$, and $\mathbf{a}_y$ defined above are augmented as follows to store corresponding information of the selected basis functions and all the candidates.

For a model of $k$ basis functions, augment $\tilde{\mathbf{A}}$ and $\mathbf{A}$ from $k$-by-$k$ to $k$-by-$M$, and $\mathbf{a}_y$ from $k$-by-1 to $M$-by-1, respectively. For $\mathbf{A}$ defined in (9), it becomes

$$\mathbf{A}=[a_{i,j}]_{k\times M},\ a_{i,j}=\begin{cases}0, & j<i\\ w_{i,j}-\sum_{s=1}^{i-1}a_{s,i}a_{s,j}/a_{s,s}, & i\le j\le M\end{cases} \tag{19}$$

$$w_{i,j}=\begin{cases}\mathbf{p}_i^T\boldsymbol{\Lambda}^2\mathbf{p}_j, & j\le k\\ \mathbf{p}_i^T\boldsymbol{\Lambda}^2\boldsymbol{\phi}_j, & j>k\end{cases}, \tag{20}$$

Matrix $\tilde{\mathbf{A}}$ defined in (9) becomes

$$\tilde{\mathbf{A}}=[\tilde{a}_{i,j}]_{k\times M},\tilde{a}_{i,j}=a_{i,j}/a_{i,i}. \tag{21}$$

which is unity upper triangular, but not a square matrix. Vector $\mathbf{a}_y$ in (14) becomes

$$\mathbf{a}_y=[a_{i,y}]_{M\times 1},\ a_{i,y}=w_{i,y}-\sum_{s=1}^{i-1}a_{s,i}a_{s,y}/a_{s,s} \tag{22}$$

where $w_{i,y}=\begin{cases}\mathbf{y}^{\mathrm{T}}\boldsymbol{\Lambda}^2\mathbf{p}_i, & i\le k\\ \mathbf{y}^{\mathrm{T}}\boldsymbol{\Lambda}^2\boldsymbol{\phi}_i, & i>k\end{cases}$.

In addition, another $M$-by-1 vector $\mathbf{b}$ can be defined as

$$\mathbf{b}=[b_i]_{M\times 1},\ b_i=w_{i,i}-\sum_{s=1}^{i-1}a_{s,i}a_{s,i}/a_{s,s} \tag{23}$$

where $w_{i,i}=\begin{cases}\mathbf{p}_i^{\mathrm{T}}\boldsymbol{\Lambda}^2\mathbf{p}_i, & i\le k\\ \boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{\Lambda}^2\boldsymbol{\phi}_i, & i>k\end{cases}$. Obviously for $w_{i,i}$ $i=1,\cdots,k$, it holds that

$$b_i=a_{i,i},\ i=1,\cdots,k \tag{24}$$

Based on (17), the contribution of each of the candidate basis functions is given by

$$\Delta J_{k+1}(\boldsymbol{\phi}_i)=a_{i,y}^2/b_i,\quad i=k+1,\cdots,M \tag{25}$$

The one from $\{\boldsymbol{\phi}_{k+1},\cdots,\boldsymbol{\phi}_M\}$ which gives the maximum contribution is then selected as the ($k$+1)'th basis function.

Assume $\Delta J_{k+1}(\boldsymbol{\phi}_j)=\arg\max\{\Delta J_{k+1}(\boldsymbol{\phi}_i),i=k+1,\cdots,M\}$, i.e. candidate $\boldsymbol{\phi}_j$ is selected and denote $\mathbf{p}_{k+1}=\boldsymbol{\phi}_j$. Other candidates are reordered and $\{\boldsymbol{\phi}_{k+2},\cdots,\boldsymbol{\phi}_M\}$ becomes the new candidate pool. According to (17), the ($k$+1)'th basis function leads to a further reduction of the cost function by $\Delta J_{k+1}(\mathbf{p}_{k+1})$.

Since $\mathbf{p}_{k+1} = \boldsymbol{\phi}_j$, both $\boldsymbol{\phi}_{k+1}$ and $\boldsymbol{\phi}_j$ in the original full regression matrix $\boldsymbol{\Phi}$ have to be interchanged, leading to the change of various intermediate matrices and vectors. For $\mathbf{A}$ and $\tilde{\mathbf{A}}$, columns $k$+1 and $j$ should also be interchanged as

$$\hat{a}_{i,k+1} = a_{i,j}, \hat{a}_{i,j} = a_{i,k+1}, \hat{\tilde{a}}_{i,k+1} = \tilde{a}_{i,j}, \hat{\tilde{a}}_{i,j} = \tilde{a}_{i,k+1}, \ i = 1, \cdots, k \tag{26}$$

where $\hat{a}_{i,k+1}$ denotes the updated $a_{i,k+1}$. To denote an updated element, a cap is applied to the element here after.

Similarly, elements $k$+1 and $j$ for $\mathbf{a}_y$ and $\mathbf{b}$ are interchanged as follows

$$\hat{a}_{k+1,y} = a_{j,y}, \hat{a}_{j,y} = a_{k+1,y}, \hat{b}_{k+1} = b_j, \hat{b}_j = b_{k+1}. \tag{27}$$

In addition, as the ($k$+1)'th basis function is selected, a new row should be appended to $\mathbf{A}$ and $\tilde{\mathbf{A}}$

$$\left.\begin{aligned} a_{k+1,i} &= w_{k+1,i} - \textstyle\sum_{s=1}^{k} \tilde{a}_{s,k+1} a_{s,i} \\ w_{k+1,i} &= \mathbf{p}_{k+1}^T \boldsymbol{\Lambda}^2 \boldsymbol{\phi}_i, \tilde{a}_{k+1,i} = a_{k+1,i} / a_{k+1,k+1} \end{aligned}\right\}, i = k+1, \cdots, M \tag{28}$$

Furthermore, for $\mathbf{a}_y$ and $\mathbf{b}$, elements from the ($k$+2)'th to the last one should be updated as follows according to the definitions (22) and (23)

$$\hat{a}_{i,y} = a_{i,y} - \tilde{a}_{k+1,i} a_{k+1,y}, \hat{b}_i = b_i - \tilde{a}_{k+1,i} a_{k+1,i}, \ i = k+1, \cdots, M \tag{29}$$

Given above details, a forward selection procedure similar to [5] can be proposed.

## 4 A Two-Stage Algorithm

To overcome the drawbacks of the forward selection algorithm, one solution is to review all the selected basis functions once the forward selection procedure terminates. For each selected basis function (of a of size $n$), say $\mathbf{p}_i$, $1 \le i \le n$, its contribution to the cost function reduction $\Delta J_n(\mathbf{p}_i)$ is compared with that of the candidate which has the maximum contribution among all the candidates. Denote the maximum candidate contribution as $\Delta J_n(\boldsymbol{\phi}_j)$, and if $\Delta J_n(\mathbf{p}_i) < \Delta J_n(\boldsymbol{\phi}_j)$ then $\mathbf{p}_i$ is said to be insignificant, and will be replaced by $\boldsymbol{\phi}_j$, in the meantime $\mathbf{p}_i$ is put back into the candidate pool. Thus, the cost function can be further reduced by $\Delta J_n(\boldsymbol{\phi}_j) - \Delta J_n(\mathbf{p}_i)$. The remaining problem is, to compute the contribution of a previously selected basis function, say $\mathbf{p}_i$ to the cost function and compare its contribution with that of the candidates $\boldsymbol{\phi}_{n+1}, \cdots, \boldsymbol{\phi}_M$, an appropriate regression context should be re-constructed. That is, $\mathbf{A}$, $\tilde{\mathbf{A}}$, $\mathbf{a}_y$ and $\mathbf{b}$ have to updated.

In order to assess the significance of the basis function $\mathbf{p}_i$, the first step is to move $\mathbf{p}_i$ to the $n$'th position of the full regression matrix $\boldsymbol{\Phi}$ as if it were the last selected basis function in the forward selection procedure. This is done by a series of

interchanges between two adjacent basis functions $\mathbf{p}_x$ and $\mathbf{p}_{x+1}$ for $x = i, \cdots, n-1$ and the regression context is updated correspondingly.

Suppose $\mathbf{p}_1, \cdots, \mathbf{p}_n$ are the selected basis functions in the order of selection. If two adjacent basis functions $\mathbf{p}_x$ and $\mathbf{p}_{x+1}$ is to be changed, $\mathbf{A}$, $\tilde{\mathbf{A}}$, $\mathbf{a}_y$ and $\mathbf{b}$ have to be updated. Denote the $n$ basis functions in the new selected order as $\mathbf{p}_1, \cdots, \mathbf{p}_{x-1}, \hat{\mathbf{p}}_x, \hat{\mathbf{p}}_{x+1}, \mathbf{p}_{x+2} \cdots, \mathbf{p}_n$, where $\hat{\mathbf{p}}_x = \mathbf{p}_{x+1}$ and $\hat{\mathbf{p}}_{x+1} = \mathbf{p}_x$. Then based on (10), for columns 1 to *x-1* in $\mathbf{A}$, since

$$\left.\begin{aligned} \hat{w}_{i,x} &= \mathbf{p}_i^{\mathrm{T}} \boldsymbol{\Lambda}^2 \hat{\mathbf{p}}_x = \mathbf{p}_i^{\mathrm{T}} \boldsymbol{\Lambda}^2 \mathbf{p}_{x+1} = w_{i,s+1} \\ \hat{w}_{i,x+1} &= \mathbf{p}_i^{\mathrm{T}} \boldsymbol{\Lambda}^2 \hat{\mathbf{p}}_{x+1} = \mathbf{p}_i^{\mathrm{T}} \boldsymbol{\Lambda}^2 \mathbf{p}_x = w_{i,x} \\ \hat{w}_{i,k} &= w_{i,k}, k = i, \cdots, x-1, x+2, \cdots, M \end{aligned}\right\}, i = 1, \cdots, x-1 \tag{30}$$

therefore

$$\left.\begin{aligned} \hat{a}_{i,x} &= a_{i,x+1},\ \hat{a}_{i,x+1} = a_{i,x} \\ \hat{a}_{i,k} &= a_{i,k}, k = i, \cdots, x-1, x+2, \cdots, M \end{aligned}\right\}, i = 1, \cdots, x-1 \tag{30}$$

Noting $\hat{w}_{x,x+1} = \hat{\mathbf{p}}_x^{\mathrm{T}} \boldsymbol{\Lambda}^2 \hat{\mathbf{p}}_{x+1} = \mathbf{p}_{x+1}^{\mathrm{T}} \boldsymbol{\Lambda}^2 \mathbf{p}_x = w_{x+1,x}$, for the *x*'th row of **A**, gives

$$\hat{a}_{x,x+1} = a_{x,x+1} \tag{32}$$

Noting $\hat{w}_{x,x} = \hat{\mathbf{p}}_x^{\mathrm{T}} \boldsymbol{\Lambda}^2 \hat{\mathbf{p}}_x = \mathbf{p}_{x+1}^{\mathrm{T}} \boldsymbol{\Lambda}^2 \mathbf{p}_{x+1} = w_{x+1,x+1}$, it holds that

$$\hat{a}_{x,x} = a_{x+1,x+1} + a_{x,x+1} a_{x,x+1} / a_{x,x} \tag{31}$$

while for the rest elements in the *x*'th row, it gives

$$\hat{a}_{x,j} = a_{x,j} + a_{x,x+1} a_{x,j} / a_{x,x},\ j = x+2, \cdots, n \tag{32}$$

Noting $\hat{w}_{x+1,x+1} = \hat{\mathbf{p}}_{x+1}^{\mathrm{T}} \boldsymbol{\Lambda}^2 \hat{\mathbf{p}}_{x+1} = \mathbf{p}_x^{\mathrm{T}} \boldsymbol{\Lambda}^2 \mathbf{p}_x = w_{x,x}$, for the (*x*+1)'th row of **A**, yields

$$\hat{a}_{x+1,x+1} = a_{x,x} - (a_{x,x+1})^2 / \hat{a}_{x,x} \tag{35}$$

$$\hat{a}_{x+1,j} = a_{x,j} - a_{x,x+1} \hat{a}_{x,j} / \hat{a}_{x,x},\ j = x+2, \cdots, M \tag{36}$$

Noting $\hat{w}_{i,j} = \hat{\mathbf{p}}_i^{\mathrm{T}} \boldsymbol{\Lambda}^2 \hat{\mathbf{p}}_j = \mathbf{p}_i^{\mathrm{T}} \boldsymbol{\Lambda}^2 \mathbf{p}_j = w_{i,j}$, $i, j > x+1$, for elements of **A** in row *x*+2,

$$\begin{aligned} \hat{a}_{x+2,j} &= \hat{w}_{x+2,j} - \textstyle\sum_{s=1}^{x+1} \hat{a}_{s,x+2} \hat{a}_{s,j} / \hat{a}_{s,s} = w_{x+2,j} - \\ &- \textstyle\sum_{s=1}^{x-1} \dfrac{a_{s,x+2} a_{s,j}}{a_{s,s}} - \dfrac{\hat{a}_{x,x+2} \hat{a}_{x,j}}{\hat{a}_{x,x}} - \dfrac{\hat{a}_{x+1,x+2} \hat{a}_{x+1,j}}{\hat{a}_{x+1,x+1}} \end{aligned} \tag{37}$$

Furthermore, it could be derived that

$$\frac{\hat{a}_{x,x+2} \hat{a}_{x,j}}{\hat{a}_{x,x}} - \frac{\hat{a}_{x+1,x+2} \hat{a}_{x+1,j}}{\hat{a}_{x+1,x+1}} = \frac{a_{x,x+2} a_{x,j}}{a_{x,x}} - \frac{a_{x+1,x+2} a_{x+1,j}}{a_{x+1,x+1}} \tag{38}$$

which means that the ($x$+2)'th row of $\mathbf{A}$ has no change, and row ($x$+2) to row $n$ of $\mathbf{A}$ have no change as well, i.e.

$$\hat{a}_{i,j} = a_{i,j}\,,\ i = x+2,\cdots,n\,,\ j = x+2,\cdots,n \tag{39}$$

Similarly,

$$\left.\begin{aligned}\hat{a}_{x,y} &= a_{x+1,y} + a_{x,x+1}a_{x,y} / a_{x,x}\\ \hat{a}_{x+1,y} &= a_{x,y} - a_{x,x+1}\hat{a}_{x,j} / \hat{a}_{x,x}\end{aligned}\right\} \tag{40}$$

For vector $\mathbf{b}$, both the $x$'th and the ($x$+1) 'th element are changed also as follows

$$\hat{b}_x = \hat{a}_{x,x},\ \hat{b}_{x+1} = \hat{a}_{x+1,x+1} \tag{41}$$

Now eventually $\mathbf{p}_i$ is moved to the $n$ position in the selected basis functions, its contribution can then be computed as follows based on (17)

$$\Delta J_n(\mathbf{p}_i) = \Delta J_n(\hat{\mathbf{p}}_n) = a_{n,y}^2 / a_{n,n} \tag{42}$$

where $\mathbf{p}_i = \hat{\mathbf{p}}_n$ indicates that $\mathbf{p}_n$ has been moved to the $n$'th position. To compute the contribution of the candidate basis functions, define:

$$\left.\begin{aligned}a_{s,y}^{(-i)} &= a_{s,y} + a_{n,s}a_{n,y} / a_{n,n}\\ b_s^{(-i)} &= b_s + (a_{n,n+1})^2 / a_{n,n}\end{aligned}\right\},\ s = n+1,\cdots,M \tag{43}$$

which are the corresponding elements in vector $\mathbf{a}_y$ and $\mathbf{b}$ if the basis function $\mathbf{p}_i$ is pruned from the network. The contribution of all the candidates is given by

$$\Delta J_n(\boldsymbol{\phi}_s) = (a_{s,y}^{(-i)})^2 / b_s^{(-i)}, s = n+1,\cdots,M \tag{44}$$

Now the significance of $\mathbf{p}_i$ can then be checked given the above derivations. First, identify the candidate basis function that gives the maximum contribution

$$\Delta J_n(\boldsymbol{\phi}_j) = \max\{\Delta J_n(\boldsymbol{\phi}_s), s = n+1,\cdots,M\}\,. \tag{45}$$

If $\Delta J_n(\boldsymbol{\phi}_j) > \Delta J_n(\mathbf{p}_i)$, $\mathbf{p}_i$ becomes insignificant and should be replaced by $\boldsymbol{\phi}_j$ as the new basis function in the network, while $\mathbf{p}_i$ should be put back into the candidate pool, taking the position of $\boldsymbol{\phi}_j$. In this case, the regression context needs to be updated again due to the interchange of $\mathbf{p}_i$ and $\boldsymbol{\phi}_j$. For matrix $\mathbf{A}$, interchange columns $n$ and $j$ from row 1 to $n$-1 due to the interchange of $\mathbf{p}_i$ and $\boldsymbol{\phi}_j$, and re-calculate the $n$'th row of $\mathbf{A}$ as

$$\left.\begin{aligned}&\hat{a}_{n,n} = b_j^{(-i)},\quad \hat{a}_{n,j} = a_{n,j}, w_{n,k} = \hat{\hat{\mathbf{p}}}_n^{\mathrm{T}}\boldsymbol{\Lambda}^2\boldsymbol{\phi}_k\\ &\hat{a}_{n,k} = w_{n,k} - \sum_{s=1}^{n-1}\frac{a_{s,n}a_{s,k}}{a_{s,s}}, k = n+1,\cdots,M, k \neq j\end{aligned}\right\} \tag{46}$$

In addition, elements $n$ to $M$ of vector $\mathbf{a}_y$ and $\mathbf{b}$ is updated respectively as

$$\left.\begin{aligned} &\hat{a}_{n,y} = a_{n,y}^{(-i)}, a_{j,y} = a_{n,y} - \hat{a}_{n,j}\hat{a}_{n,y} / \hat{a}_{n,n}, \\ &a_{s,y} = a_{s,y}^{(-i)} - \frac{\hat{a}_{n,s}\hat{a}_{n,y}}{\hat{a}_{n,n}},\ s = n+1, \cdots, M, s \neq j \end{aligned}\right\} \tag{47}$$

$$\left.\begin{aligned} &\hat{b}_n = b_j^{(-i)}, \hat{b}_j = a_{n,n} - (\hat{a}_{n,n+1})^2 / \hat{a}_{n,n} \\ &\hat{b}_s = b_s^{(-i)} - (\hat{a}_{n,n+1})^2 / \hat{a}_{n,n},\ s = n+1, \cdots, M, s \neq j \end{aligned}\right\} \tag{48}$$

Similarly the corresponding element of $\tilde{\mathbf{A}}$ can be recalculated based on (21).

The computational complexity analysis for the above proposed algorithm can be performed by referring to [6].

## 5 Real Time Implementation

To facilitate real time neural network construction using the above algorithm, a weighting scheme for the data samples is applied to reduce the data storage. In this paper, the weights for samples at current time instance $t$ are always set at 1 and $\lambda^{\tau/2}$ for the $t-\tau$, where $0 < \lambda < 1$ and normally close to 1. In this way, the weights for the past data samples decrease exponentially as time goes on, leading to the gradual removal of past data samples in the real-time neural network construction.

In detail, the weight matrix $\mathbf{W}$ in (8) and $\mathbf{w}_y$ in (12) is applied to all basis functions, i.e.

$$\mathbf{W} = [w_{i,j}]_{M\times M}, w_{i,j} = \boldsymbol{\phi}_i^{\mathrm{T}} \boldsymbol{\Lambda}^2 \boldsymbol{\phi}_j \tag{49}$$

$$\mathbf{w}_y = [w_{i,y}]_{M\times 1}, w_{i,y} = \mathbf{y}^{\mathrm{T}} \boldsymbol{\Lambda}^2 \boldsymbol{\phi}_i \tag{50}$$

Note that $\mathbf{W}$ is symmetric and only the upper triangle, including the diagonal elements, needs to be stored. Applying the exponential weighting scheme, gives

$$\left.\begin{aligned} &w_{i,j}(t) = \textstyle\sum_{\tau=1}^{t} \lambda^{t-\tau} \phi_i(t)\phi_j(t) \\ &w_{i,y}(t) = \textstyle\sum_{\tau=1}^{t} \lambda^{t-\tau} y(t)\phi_j(t) \end{aligned}\right\} \tag{51}$$

In real time implementation, $\mathbf{W}$ and $\mathbf{w}_y$ are dynamically updated as

$$w_{i,j}(t) = \lambda w_{i,j}(t-1) + \phi_i(t)\phi_j(t), w_{i,j}(0) = 0 \tag{52}$$

$$w_{i,y}(t) = \lambda w_{i,y}(t-1) + y(t)\phi_j(t),\ w_{i,y}(0) = 0 \tag{53}$$

In this weighting scheme, for $w_{i,j}$, if $|\phi_i(t)\phi_j(t)| \to C$ as $t \to \infty$, then $|w_{i,j}(t)| \to C/(1-\lambda)$. The algorithm proposed in this paper is a two-stage method, therefore the real time implementation is also divided into two stage.

To initialize the algorithm, let $k = 0$ and $w_{i,y} = 0, w_{i,j} = 0,\ j > i$ for $i = 1, \cdots, M$, as each sample of data is recorded, following procedure is implemented.

A) With the recorded new sample of data, update $\mathbf{W}$ and $\mathbf{w}_y$ respectively.

B) Update $\mathbf{A}$, $\tilde{\mathbf{A}}$, $\mathbf{a}_y$ and $\mathbf{b}$ in (19), (21), (22) and (23), respectively

C) If $k < n$, implement step b), c) and d) of the forward selection procedure to select the $k$'th basis function. Otherwise implement the reviewing procedure to check all the $n$ selected basis functions.

D) Identify the output weights based on (5). The neural network (of $k$ basis functions) can then be used for real time application.

Note that the neural network size $n$ is given previously. This procedure is implemented recursively.

## 6 Simulations

Consider the following non-linear dynamic system

$$y(t) = 0.5 + 0.5\,y(t-1) - 0.05\,y^2(t-2) + 0.8u(t-2) + u^2(t-1) + \xi(t) \tag{54}$$

where $t$, y and $u$ represent the time series, the system output and input, respectively; $\xi(t)$ is the system noise given as $\xi \sim N(0,0.05)$. By simulating (54) with $u(t)$ uniformly distributed within the range [-1.0, 1.0], two data sets of 500 samples for each were generated, the first one for neural network construction and training and the other for validation. Full polynomial of orders 0 to 3 of $y(t\text{-}l)$ and $u(t\text{-}l)$ for $l$ = 1, 2, and 3 are used. The network size is set to be 5. Both the proposed two-staged algorithm and the forward-only algorithm were then used to select 5 basis functions. The indices of the selected terms and the corresponding SSWE are listed in Table 1 for both the algorithms. Table 2 compares the normalized one-step-ahead and long-term predictions errors over both the training and validation data sets, respectively, of the two produced neural networks. From the simulation example, it is obvious that the two stage algorithm achieves a neural network of far better performance than forward subset selection methods.

**Table 1.** Selected nodes and performance

| Algorithm | Index of the selected terms | SSWE |
|---|---|---|
| Forward | 1, 2, 6, 23, 44 | 5.4425 |
| Two-staged | 1, 2, 6, 14, 23 | 1.3728 |

**Table 2.** Normalized prediction error (%)

| Algorithm | Training data | | Validation data | |
|---|---|---|---|---|
| | One step | Long term | One step | Long term |
| Forward | 6.78 | 7.51 | 6.53 | 7.28 |
| Two-stage | 3.41 | 3.40 | 3.23 | 3.21 |

## 7 Conclusion

In this paper, a stepwise two-stage algorithm has been proposed for real-time construction of generalized single-layer networks (GSLNs), which enables both network growing and network modification. The main contribution of this paper is that these two directions of network construction are performed within one regression context using Cholesky decomposition, leading to both significantly neural network performance and concise network construction procedures. Simulation results have demonstrated the effectiveness of this method.

## Acknowledgement

Dr K. Li wishes to acknowledge the financial support of the UK Engineering and Physical Sciences Research Council (EPSRC Grant GR/S85191/01). Dr M. Fei wishes to acknowledge the financial support of Shanghai Leading Academic Disciplines (T0103).

## References

1. K. M. Adeney, M. J. Korenberg: Iterative fast orthogonal search algorithm for MDL-based training of generalized single-layer networks. Neural Networks, Vol 13 (2000) 787-799.
2. S. Chen, S. A. Billings, W. Luo: Orthogonal Least Squares Methods and Their Application to Non-linear System Identification. International Journal of Control, Vol. 50, No.5 (1989) 1873-1896.
3. S. Chen, J. Wigger: Fast Orthogonal Least Squares Algorithm for Efficient Subset Model Selection. IEEE Transactions on Signal Processing, Vol. 43, No.7, (1995) 1713-1715.
4. B. Igelnik and Y. H. Pao, “Additional Perspectives of feedforward neural-nets and the functional-link”, IJCNN '93, Nagoya, Japan, 1993.
5. K. Li, J. Peng, G. Irwin: A fast nonlinear model identification method. IEEE Transactions on Automatic Control. Vol. 50, No. 8 (2005) 1211-1216.
6. K. Li, J. Peng, Er-Wei Bai: A two-stage algorithm for identification of nonlinear dynamic systems. Automatica, Vol. 42, No 7 (2006) (in press).
7. H. Akaike: A New Look at the Statistical Model Identification. J. R. Statist. Soc. Ser. B., Vol.36 (1974) 117-147.
8. Sherstinsky, R. W. Picard: On the Efficiency of the Orthogonal Least Squares Training Method for Radial Basis Function Networks. IEEE Transactions on Neural Networks, Vol. 7, No. 1 (1996) 195-200.