Model structure determination in neural network models

Humberto M. Henriquea,*, Enrique L. Limab, Dale E. Seborgc

aDepartamento de Engenharia Química, Universidade Federal de Uberlândia, Av. João Naves D’Ávila, no 2160, Uberlândia, MG 38400-100, Brazil
bPrograma de Engenharia Química/COPPE, Universidade Federal do Rio de Janeiro, Cidade Universitária, Centro de Tecnologia, Rio de Janeiro, RJ 21945-970, Brazil
cDepartment of Chemical Engineering, University of California, Santa Barbara, CA 93106, USA

Received 15 October 1999; accepted 20 March 2000

Abstract

Feedforward neural networks (FNN) have been used intensively for the identification and control of chemical engineering processes. However, there is no efficient model structure determination methodology for a particular mapping application. This has resulted in a tendency to use networks that are much larger than required. In this paper a new procedure for model structure determination in feedforward neural networks is proposed. This procedure is based on network pruning using the orthogonal least-squares technique to determine insignificant or redundant synaptic weights, biases, hidden nodes and network inputs. The advantages of this approach are discussed and illustrated using simulated and experimental data. The results show that the orthogonal least-squares technique is quite efficient in determining the significant elements on the neural network models. The results also show the importance of pruning procedures to identify parsimonious FNN models. © 2000 Elsevier Science Ltd. All rights reserved.

Keywords: Neural networks; Pruning of neural networks; Model structure determination; System identification; Dynamic systems; Nonlinear modeling

1. Introduction

Feedforward neural networks (FNNs) have been used intensively for chemical engineering applications. In this computational paradigm knowledge is captured by the strength of the network interconnections (synaptic weights), which can be calculated in an iterative way using a steepest descent-based optimization routine in order to minimize a given objective function (Rumelhart, Hinton & Williams, 1988). The main advantage of the steepest descent algorithm is parallelizability. However, more efficient routines for serial computers are available, as the one based on the Marquardt–Levenberg algorithm used in this work. The convergence of this algorithm is quadratic and, as a consequence, it is computationally more faster than the linear converging steepest descent approaches.

Cybenko (1989) showed that any mapping from \( \mathbb{R}^p \) to \( \mathbb{R}^q \) could be achieved with two layers of hidden nodes. Hornik, Stinchcombe and White (1990) showed that any mapping could also be achieved, to an arbitrary degree of accuracy, using a neural network with one hidden layer and a sufficient large number of nodes. However, to increase the mapping accuracy, users often increase the network complexity using two or even three hidden layers with a large number of nodes. The main problem with this approach is that a neural network with a large number of weights has a strong tendency to overfit the data and to perform poorly on data not used in the training phase. In addition, the weight parameter estimation procedure becomes extremely difficult because of numerous local minima.

It is well known that large neural networks often have large number of redundancies that increase the network complexity without significantly increasing the mapping accuracy. In this context, the purpose of this paper is to develop a methodology to identify and eliminate redundant and insignificant network parameters.

This paper is organized as follows. Section 2 presents a brief review about the classical pruning methodology in
FNNs. In Section 3 is given the necessary background on orthogonal least-squares estimation and the methodology of the FNNs pruning based on this technique. Section 4 presents and discuss results obtained by using the proposed method in simulated and experimental cases, including a nonlinear chemical engineering problem. Finally, Section 5 presents the main conclusions.

2. Classical pruning techniques

Parsimonious structures for FNNs can be obtained by the process of pruning, that consists in deleting unnecessary weights or/and nodes, starting with a large network and going on reducing it until a parsimonious model is obtained. Once the network is simplified, problems related to overfitting disappear.

Hagiwara (1990) suggested that the number of hidden nodes could be reduced by identifying "bad" nodes. A "badness" factor associated with each node depends on the error being backpropagated through that node. Nodes with high "badness" factors are deleted.

Mozer and Smolensky (1989) have suggested that hidden nodes can be deleted by a process of "relevance" assessment (known as "skeletonization"). The "relevance" of each node “i” is represented by the difference between the mean square error of the network output (E), calculated from training data, with and without the node “i”. The network is trimmed by removing irrelevant nodes.

There is another method that does not delete nodes but individual weights. This technique is based on the minimization of a dual objective function (Ew) during the training phase of the network:

\[ E_w = E + \lambda E_{\text{com}}, \quad \lambda > 0, \]

where \( \lambda \) is a weighting parameter. The "complexity" function (Ecom) depends on network "complexity". To overcome some problems presented this method, Bhat and McAvoy (1992) suggested a different "complexity" function. They defined Ecom as follows:

\[ E_{\text{com}} = \sum_k N_k \sum_j w_{jk}^2, \]

where \( N_k \) is the number of active weights (\( w_{jk} \)) arriving at the \( k \)th node. They obtained good results for an illustrative linear example and for the dynamic behavior of pH in a continuous-stirred-tank reactor. These methods are computationally heavy because they require a line search in \( \lambda \).

Le Cun, Denker and Solla (1990) proposed the optimal brain damage (OBD) method that utilizes a measure of the "saliency" of a weight by estimating the second derivative of the network output error \( E \) with respect to that weight. The complexity of the technique is also reduced by a large factor by constraining certain weights to be equal. When the weight vector \( w \) is perturbed, the change in the network output error is approximately determined by

\[
\delta E = \sum_i j_i \delta w_i + \frac{1}{2} \sum_i h_{ii} \delta w_i^2 \\
+ \frac{1}{2} \sum_{i \neq j} h_{ij} \delta w_i \delta w_j + g(||\delta w||^3),
\]

where \( \delta w_i \) represent the components of the perturbed parameter vector \( \delta w, j_i \) the components of the gradient of \( E \) with respect to \( w \), and the \( h_{ij} \) the elements of the Hessian matrix \( H \):

\[ j_i = \frac{\partial E}{\partial w_i}, \]

\[ h_{ij} = \frac{\partial^2 E}{\partial w_i \partial w_j} \]

Since pruning is carried out on a well-trained network, the first term on the right-hand side of Eq. (3) will be zero because, in this case, \( E \) is at a minimum. Also, if the perturbations are small, the last term will be negligible. Since \( H \) is a very large matrix, the OBD method reduce computational problems removing those weights that at least affect the training error, based on a diagonal approximation. This leaves

\[ \delta E = \frac{1}{2} \sum_i h_{ii} \delta w_i^2. \]

It turns out that the second derivative \( h_{kk} \) can be calculated by a modified backpropagation rule. The "saliency" \( s_k \) of the weight \( w_k \) is then defined as

\[ s_k = h_{kk} w_k^2 \frac{1}{2}. \]

Pruning is carried out iteratively, that is, train to a reasonable error level, compute "saliencies", delete low "saliency" weights and resume training.

When the diagonal assumption is inaccurate, it can lead to the removal of the wrong weights. To overcome this possibility, Hassibi, Stork and Wolff (1992) proposed the Optimal Brain Surgeon (OBS) method, that follows the same idea used in the OBD method, but removing the diagonal assumption. However, it is impractical for large networks. An early stopping procedure monitors the error on a validation set and halts learning when this error starts to increase. There is no guarantee that the learning curve passes through the optimal point, and the final weights are sensitive to the learning dynamics. Weight decay (ridge regression) adds a term to the objective function that penalizes large weights. The proper coefficient for this term is not known a priori, so one must perform several optimizations with different values, leading to a cumbersome process. In this paper, the OBS
and OBS methods, as implemented by Nørgaard (1996),
are used to compare with the proposed method.

3. Orthogonal least-squares-based pruning

The pruning method here proposed is based on the
orthogonal least-squares algorithm as described by
Billings, Chen and Korenberg (1989). Consider the linear
regression function

\[ z(t) = \sum_{i=1}^{m} p_i(t) \theta_i + \zeta(t) \quad (t = 1, \ldots, n), \]  

(7)

where \( z(t) \in \mathcal{R} \) is a dependent variable, \( p_i(t) \) is the \( i \)th regressor or predictor, \( \zeta(t) \in \mathcal{R} \) is the modeling error and \( \theta_i \) is the \( i \)th unknown parameter to be estimated. Eq. (7)
can be written in a compact form

\[ z = P\theta + \zeta, \]  

(8)

where \( z = [z(1) \ldots z(n)]^T \in \mathcal{R}^n \), \( P = [p_1 \ldots p_m] \in \mathcal{R}^n \times \mathcal{R}^m \) with \( p_i = [p_i(1) \ldots p_i(n)]^T \), \( \theta = [\theta_1 \ldots \theta_m]^T \in \mathcal{R}^m \) and
\( \zeta = [\zeta(1) \ldots \zeta(n)]^T \in \mathcal{R}^n \). For the model given by Eq. (8),
the least-squares estimated parameter vector is given by
the classical “normal equation”

\[ \hat{\theta} = (P^TP)^{-1}P^Tz. \]  

(9)

In many cases Eq. (9) presents computational
problems. A numerically stable method for solving these
problems is based on an orthogonalization procedure
that will be briefly discussed. Consider the Choleski
factorization of the regressor matrix \( P \):

\[ P^TP = A^TDA, \]  

(10)

where \( A \) is an upper triangular matrix with unity
diagonal elements and \( D \) is a diagonal matrix with positive
diagonal elements. Now, using matrix \( A \), Eq. (8) can be
modiﬁed to give

\[ z = Bg + \xi, \]  

(11)

where \( B = PA^{-1} \), \( g = A\theta \) and \( B = [b_1 \ldots b_m] \) with
\( b_i = [b_i(1) \ldots b_i(n)]^T \). The auxiliary regressor matrix
\( B \) is orthogonal because \( B^TB = (PA^{-1})^T(PA^{-1}) = D \). It
can be computed recursively from:

\[ b_1(t) = p_1(t) \]

\[ b_k(t) = p_k(t) - \sum_{i=1}^{k-1} x_{ik} b_i(t), \quad k = 2, \ldots, m, \]  

(12)

where

\[ x_{ik} = \frac{\sum_{l=1}^{n} b_i(l)p_k(l)}{\sum_{l=1}^{n} b_i(l)^2}. \]  

(13)

The auxiliary parameter vector \( g = [g_1 \ldots g_m]^T \)
satisfies

\[ g = D^{-1}B^Tz - D^{-1}B^T\xi \]  

(14)

so that, neglecting errors, the estimated \( g \) is given by:

\[ \hat{g} = D^{-1}B^Tz \]  

(15)

or

\[ \hat{\theta}_i = \frac{\sum_{l=1}^{n} b_i(l)z(l)}{\sum_{l=1}^{n} b_i(l)^2}, \quad i = 1, \ldots, m. \]  

(16)

Estimates of the original parameters can be computed
from:

\[ \tilde{\theta} = \hat{\theta} - (A - I)\hat{\theta}. \]  

(17)

That is,

\[ \tilde{\theta}_m = \hat{g}_m \]  

(18)

\[ \tilde{\theta}_i = \hat{\theta}_i - \sum_{k=i+1}^{m} x_{ik} \tilde{\theta}_k, \quad i = m - 1, \ldots, 1. \]

From Eq. (11)

\[ z^Tz = g^TB^T Bg + \xi^T \xi + g^TB^T \xi + \xi^T Bg. \]  

(19)

Considering that \( \zeta(t) \) is a mean white sequence
uncorrelated with \( p_i(t) \), it can be shown that

\[ z^Tz = \sum_{i=1}^{n} |g_i^2b_i^2| + \xi^T \xi \]

or

\[ \frac{1}{n} - z^Tz = \sum_{i=1}^{m} \left[ g_i^2 \frac{1}{n} - b_i^2b_i \right] + \sigma_z^2, \]  

(20)

where \( \sigma_z^2 \) is the modeling error variance. Therefore, the
contribution of the dependent variable to the variance
that is explained by the auxiliary regressor \( b_i(t) \) is
\( (g_i^2/n)b_i^2b_i \). This result can be used as a criterion for model
structure selection: only the auxiliary regressors \( b_i \), or
the regressors \( p_i \), that contribute more for the dependent
variable are considered in Eq. (11) or Eq. (8). It
serves as our “saliency” measure for the regressors \( b_i \) or
\( p_i \). So, regressors with low “saliency” are removed and
this result is an effective reduction of the number of
network parameters. Therefore, the procedure captures
all significant model terms and still eliminates any
numerical ill-conditioning of the \( P^TP \) matrix. The method
has been tested in Korenberg, Billings, Liu and McIlroy
(1988) and Billings et al. (1989) for polynomial NAR-
MAX models and it has shown to be very efficient for
model structure selection. The multivariable case is easily
obtained as a straightforward extension of the single
variable case. It should be note that a similar approach
can be obtained through singular-value decomposition
also known as PLS or Karhunen–Loeve transform. In
this case the magnitude of singular values are propor-
tional of importance of the regressors in the matrix
\( P \) and, therefore, they could be used as “saliency”
measure for these regressors. However, computational
The effort involved in singular-value decomposition approach is much more intensive than Choleski factorization.

Now, consider the FNN shown in Fig. 1 that can be mathematically expressed as follows:

\[
\begin{align*}
\mathbf{v}'_1(t) & = \begin{bmatrix}
W_{1,1} & \cdots & W_{1,p} \\
\vdots & \ddots & \vdots \\
W_{N_h,1} & \cdots & W_{N_h,p}
\end{bmatrix} \mathbf{u}(t) + \mathbf{d}_1 \\
\vdots & = f[\mathbf{v}'_1(t)] \\
\mathbf{v}_N(t) & = \begin{bmatrix}
W_{2,1} & \cdots & W_{2,1,N_h} \\
\vdots & \ddots & \vdots \\
W_{2,q,1} & \cdots & W_{2,q,N_h}
\end{bmatrix} \mathbf{v}_1(t) + \mathbf{d}_2 \\
\vdots & = f[\mathbf{v}_N(t)] \\
\mathbf{y}'_1(t) & = \begin{bmatrix}
W_{1,1} & \cdots & W_{1,q} \\
\vdots & \ddots & \vdots \\
W_{N_h,1} & \cdots & W_{N_h,q}
\end{bmatrix} \mathbf{u}(t) + \mathbf{d}_1 \\
\vdots & = f[\mathbf{y}'_1(t)] \\
\mathbf{y}_N(t) & = \begin{bmatrix}
W_{2,1} & \cdots & W_{2,1,q} \\
\vdots & \ddots & \vdots \\
W_{2,q,1} & \cdots & W_{2,q,q}
\end{bmatrix} \mathbf{v}_1(t) + \mathbf{d}_2 \\
\vdots & = f[\mathbf{y}_N(t)] \\
\mathbf{y}(t) & = \mathbf{W}_2 \mathbf{v}(t) + \mathbf{d}_2 \\
\mathbf{y}(t) & = f[\mathbf{y}(t)]
\end{align*}
\]

or concisely in the form

\[
\mathbf{v}'(t) = \mathbf{W}_1 \mathbf{u}(t) + \mathbf{d}_1, \\
\mathbf{v}(t) = f[\mathbf{v}'(t)] \\
\mathbf{y}(t) = \mathbf{W}_2 \mathbf{v}(t) + \mathbf{d}_2, \\
\mathbf{y}(t) = f[\mathbf{y}(t)]
\]

where \( f(\cdot) \) represents any activation function, \( N_h \) is the number of hidden nodes, \( p \) is the number of system inputs and \( q \) is the number of system outputs. In order to transform Eqs. (21) and (23) into the form of Eq. (8), both can be rewritten as concatenated relations:

\[
\begin{align*}
\mathbf{v}'(t) & = [1 \ \mathbf{u}(t)] \begin{bmatrix}
\mathbf{d}_1^T \\
\mathbf{W}_1^T
\end{bmatrix} \\
\mathbf{y}'(t) & = [1 \ \mathbf{v}(t)] \begin{bmatrix}
\mathbf{d}_1^T \\
\mathbf{W}_2^T
\end{bmatrix}
\end{align*}
\]

Once the neural network has already been trained and since \( \mathbf{u}(t) \) and \( \mathbf{y}(t) \) are known, the intermediate signals \( \mathbf{y}'(t) \) and \( \mathbf{v}(t) \) can be computed as in Eqs. (24) and (22), respectively. Now, Eqs. (26) and (27) are in the form of Eq. (8). Therefore, the OLS algorithm can be applied and the significant parameters of \( \mathbf{W}_2, \mathbf{W}_1, \mathbf{d}_2 \) and \( \mathbf{d}_1 \) (weights and bias) can be selected. If the output of a hidden node is not significant for any output node, this node is deleted from the neural network. The same occurs to an input node that is not significant to any hidden node. Then, the pruned neural network is retrained and the pruning method is applied again. This procedure goes on until convergence. Results have shown that only a few iterations are needed for convergence. The algorithm is general (any type of activation function) and can be described briefly as follows:

1. Start with a FNN model and train until convergence.
2. Apply the orthogonal least-squares algorithm to the neural network.
3. Select all significant synaptic weights and biases.
4. If all synaptic weights and biases are significant go to step 6.
5. Train the pruned network and go to step 2.
6. End.

4. Examples

In this section the proposed algorithm is tested on four cases; the first two cases were also used in comparison with the StripNet algorithm presented by Bhat and McAvoy (1992). For each case, a large FNN was initially trained until convergence was obtained using hyperbolic tangent functions as activation. Then the network was pruned using the proposed OLS pruning algorithm and the requirement that the pruned networks had to be explained at least 99.99% of the variance of the intermediate signals \( \mathbf{y}'(t) \) and \( \mathbf{v}(t) \). The first two examples are representive of steady-state modeling, while the others are representative of dynamic modeling. The OBD and OBS methods, implemented by Nørgaard (1996), were compared with the proposed method. For these methods, retraining was carried out every time that the pruning procedure reached 5% of elimination of the remaining weights. The value of 5% was chosen after extensive computational investigation.

4.1. Steady-state examples

These examples are simple but illustrate very well how redundant and insignificant inputs can be identified and
deleted from the neural model. The system to be modeled is described as follows:

\[ y(u_1, u_2) = u_1 + u_2. \]  

(27)

In the first example, the initial FNN used for this mapping has five hidden nodes, three inputs being two deterministic numerical sequences \((u_1, u_2)\) and a random sequence \((u_3)\) and one output. Therefore, the full network has 26 parameters. Obviously, the output \(y(u_1, u_2)\) can be determined by using only \(u_1\) and \(u_2\) because \(u_3\) is not relevant. Two data sets with 500 points each were used for the training and validation phases. The neural network performance was measured by a multiple correlation coefficient \((r^2)\), which is defined as follows (Milton & Arnold, 1990):

\[
r^2 = \left(1 - \frac{\sum_{t=1}^{n} [y_d(t) - \hat{y}(t)]^2}{\sum_{t=1}^{n} [y_d(t) - \bar{y}]^2}\right) \times 100\%,
\]

(28)

where \(y_d(t)\) is the target value of \(y(t)\), \(\bar{y}\) is the mean value of the \(y(t)\) and \(n\) is the data number. The redundant inputs and hidden nodes were deleted after the proposed OLS algorithm was run on the network. Table 1 and Fig. 2 present the results for the OLS, the OBD and the OBS pruned methods. The OLS pruned FNN has two input nodes, two hidden nodes, one output node and two biases. Note that the chosen inputs are \(u_1\) and \(u_2\), the number of weights was reduced from 26 to 8 and there are only minor changes in the \(r^2\) coefficient among full

<table>
<thead>
<tr>
<th></th>
<th>No. of parameter</th>
<th>(r^2) (training phase)</th>
<th>(r^2) (validation phase)</th>
<th>CPU time ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full FNN</td>
<td>26</td>
<td>100</td>
<td>100</td>
<td>—</td>
</tr>
<tr>
<td>OLS pruned FNN</td>
<td>8</td>
<td>99.99</td>
<td>99.98</td>
<td>1</td>
</tr>
<tr>
<td>OBS pruned FNN</td>
<td>8</td>
<td>99.99</td>
<td>99.98</td>
<td>9.82</td>
</tr>
<tr>
<td>OBD pruned FNN</td>
<td>8</td>
<td>99.99</td>
<td>99.98</td>
<td>10.01</td>
</tr>
</tbody>
</table>

Table 1 Results of the pruning procedure for the static example with an insignificant input

*Ratio between cpu time spent for pruning in a particular method and the fastest one.

Fig. 2. Pruned networks for the static example with an insignificant input.
Table 2
Results of the pruning procedure for the static example with a redundant input

<table>
<thead>
<tr>
<th></th>
<th>No. of parameter</th>
<th>$r^2$ (training phase)</th>
<th>$r^2$ (validation phase)</th>
<th>CPU time ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full FNN</td>
<td>26</td>
<td>100</td>
<td>100</td>
<td>—</td>
</tr>
<tr>
<td>OLS pruned FNN</td>
<td>8</td>
<td>99.54</td>
<td>99.62</td>
<td>1</td>
</tr>
<tr>
<td>OBS pruned FNN</td>
<td>7</td>
<td>99.48</td>
<td>99.46</td>
<td>10.08</td>
</tr>
<tr>
<td>OBD pruned FNN</td>
<td>21</td>
<td>100</td>
<td>99.98</td>
<td>10.64</td>
</tr>
</tbody>
</table>

Fig. 3. Pruned networks for the static example with a redundant input.

FNN and OLS pruned FNN for training and validation phases. Similar results were obtained for the OBD and OBS methods. These results show that all three methods were able to identify and eliminate the insignificant input to the network. However, the proposed OLS method was computationally much more efficient than both OBD and OBS methods.

In the second example, the input $u_3$ was changed to be $u_3 = au_1 + bu_2$. Thus, the third FNN input was linearly dependent on $u_1$ and $u_2$ and, therefore, redundant. The FNN output $y(u_1, u_2)$ can be determined by using only two of the three available inputs. The same full network was used as starting point and the same procedure was repeated. Results are shown in Table 2 and Fig. 3. Note that the inputs $u_1$ and $u_3$ were chosen for the OLS and the OBS pruned methods and the numbers of weights were reduced from 26 to 8 and from 26 to 7, respectively. The OBD pruned method was not able to identify the redundant input. There was very little variation in the $r^2$ coefficient between the full and pruned networks, for both training and validation phases. The proposed OLS method was again the most efficient.

These results show that if an input set is available for a particular mapping, then the proposed algorithm is able to identify and eliminate those inputs which are linearly dependent. This capability is particularly useful for discrete-time modeling where large amount of data are available and co-linearity of data often occurs.

4.2. Dynamic systems

Before introducing the dynamic cases, it is convenient to discuss some practical aspects of the representation of these type of problems. Consider the general dynamic system

$$\frac{dy(t)}{dt} = f[y(t), u(t)].$$

(29)
The natural way to represent this system using neural networks is to consider neurons with dynamic characteristics (You & Nikolaou, 1993). This approach has the advantage of producing models of small dimension. For a single-input single-output system (SISO), the resulting neural network will have only one input. The main problem with these types of neural networks is related to the difficulties involved in the training phase. A popular alternative is to consider a neural network with static neurons representing a discrete approximation of the dynamic system in the form of a NARMAX model (Su, McAvoy & Werbos, 1992):

\[
y(t + 1) = f[y(t), y(t - 1), \ldots, y(t - n_y), u(t), u(t - 1), \ldots, u(t - n_u)].
\]

(30)

In this case, the main advantage is associated with the simplicity of the training phase. The problem is that the number of required network inputs increases with \(n_y\) and \(n_u\), so that the network structural dimension also increases. Another problem is that the determination of the \(n_y\) and \(n_u\) values normally requires a tedious iterative process. For these reasons, in this work we consider a different alternative that consists of the direct representation of Eq. (30) with a static neural network followed by a numerical integration to recover \(y(t + 1)\). If the value of \(y(t)\) from plant is used as initial condition for obtaining \(y(t + 1)\) by integration then an one step ahead prediction is obtained, but if the value of \(y(t)\) is obtained from previous integration step then a multiple step ahead prediction is obtained.

The proposed algorithm will be used to determine the size and the topology of FNN models of a pH neutralization process. The same strategy and conditions used for the static case will be used again. The first pH example, a simulation study, is used to test the capability of the algorithm to produce a parsimonious nonlinear dynamic model. Then an experimental application is used to test the capability to cope with data corrupted by measurement noise. In these two examples, the significance of each neural model was evaluated using various statistical criteria (Pottmann & Seborg, 1992):

- Corrected multiple correlation coefficient \((r^2_a)\) according to Amemiya.
- Final prediction error (FPE).
- Akaike's information criterion (AIC(\(\phi\))).
- Kinchin's law of iterated logarithm criterion (LILC).
- The Bayesian information criteria (BIC).

The last four criteria evaluate the trade-off between model size, determined by the number of parameters, and model accuracy, determined by a loss function.

4.2.1. Simulation study: pH neutralization

Consider the continuous pH neutralization process, shown in Fig. 4. It has three inlet streams — an acid stream (HNO₃), a base stream (NaOH) and a buffer stream (NaHCO₃) — and one effluent stream. The liquid level in the tank is kept constant. The FNN model predicts the time derivatives of pH as a function of the base flow rate. The physical model of the UCSB pH neutralization process was developed by Hall and Seborg (1989) based on the assumptions of perfect mixing, constant density, complete solubility of the involved ions and no substance entering or leaving the system, except through the flow streams. The chemical reactions for the neutralization process are

\[
\begin{align*}
\text{H}_2\text{O} & \rightleftharpoons \text{OH}^- + \text{H}^+, \\
\text{H}_2\text{CO}_3 & \rightleftharpoons \text{HCO}_3^- + \text{H}^+, \\
\text{HCO}_3^- & \rightleftharpoons \text{CO}_3^{2-} + \text{H}^+. 
\end{align*}
\]

Following the approach of Gustafsson and Waller (1983), two reaction invariants are defined for the \(i\)th stream:

\[
W_a = [\text{H}^+] - [\text{OH}^-] - 2[\text{HCO}_3^-],
\]

(31)

\[
W_b = [\text{H}_2\text{CO}_3] + [\text{HCO}_3^-] + [\text{CO}_3^{2-}].
\]

(32)

The quantities \(W_a\) and \(W_b\) are called reaction invariants because they are not affected by the extent of the reactions. It is assumed that the reactions are fast enough for the system to be at equilibrium. Then the equilibrium relations can be used to determine the hydrogen ion concentration from the reaction invariants. The equilibrium constants are given by

\[
K_{a_i} = \frac{[\text{HCO}_3^-][\text{H}^+]}{[\text{H}_2\text{CO}_3]},
\]

(33)

\[
K_{a_2} = \frac{[\text{CO}_3^{2-}][\text{H}^+]}{[\text{HCO}_3^-]},
\]

(34)

\[
K_w = [\text{H}^+][\text{OH}^-].
\]

(35)

By combining Eqs. (31)–(35), the following implicit algebraic relation between \([\text{H}^+]\), \(W_a\) and \(W_b\) can be derived:

\[
W_a = [\text{H}^+] \frac{K_w}{[\text{H}^+]} - W_b \frac{(K_{a_1}/[\text{H}^+]) + 2(K_{a_2} K_{a_3}/[\text{H}^+])^2}{1 + (K_{a_1}/[\text{H}^+]) + (K_{a_2} K_{a_3}/[\text{H}^+])^2}.
\]

(36)
Since the liquid level is constant $Q_4$ is equal to $(Q_1 + Q_2 + Q_3)$, the dynamic model of the neutralization process consists of the component balances on the reaction invariants only:

$$V_R \frac{d(W_a)}{dt} = Q_1(W_{a_1} - W_a) + Q_2(W_{a_2} - W_a)$$

$$+ Q_3(W_{a_3} - W_a),$$

$$V_R \frac{d(W_b)}{dt} = Q_1(W_{b_1} - W_b) + Q_2(W_{b_2} - W_b)$$

$$+ Q_3(W_{b_3} - W_b).$$  \hspace{0.5cm} (37)

These equations allow the reaction invariants in the tank to be determined as a function of time, assuming that the initial conditions and flow rates and concentrations of the inlet streams are known. The pH value of the solution in the tank can be determined from Eq. (44) and invariants $W_a$ and $W_b$. The nominal conditions for the system are shown in Table 3.

The system was excited using a random uniform step sequence for base flow rate $Q_3$, with a step probability (probability of a step change occurring at any given sampling instant) equal to 0.9 (Bomberger & Seborg, 1998). Fig. 6a shows this excitation. A sampling period of 15 s was used, as per previous studies. Data from the time interval (0–2.1 h) were used for the FNN training phase and from the interval (2.1–4.2 h) for the validation phase. The network inputs were $Q_3(t)$, $\text{pH}(t)$ and the network output was $\frac{d[\text{pH}(t)]}{dt}$. The derivatives of pH were calculated numerically by finite-difference schemes. An initial neural network with 15 hidden nodes was trained until convergence, using the Levenberg–Marquardt method once a serial computer was used for simulation. Then the OLS, OBD and OBS pruning algorithms were applied. By inspection it is easy to see that as more negative the indexes FPE, LILC, BIC are better the performance of a pruning algorithm is for training and validation phases. For $r^2$, good values are close to one. For this example the results are shown in Fig. 5 and

---

**Table 3**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Nominal value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume of reactor</td>
<td>3289 cm$^3$</td>
</tr>
<tr>
<td>Acid flow rate ($Q_1$)</td>
<td>15.5 ml/s</td>
</tr>
<tr>
<td>Base flow rate ($Q_1$)</td>
<td>14.5 ml/s</td>
</tr>
<tr>
<td>Buffer flow rate ($Q_2$)</td>
<td>0.55 ml/s</td>
</tr>
<tr>
<td>pH value</td>
<td>7.0</td>
</tr>
<tr>
<td>Acid conc. (in $Q_1$)</td>
<td>0.003 M HNO$_3$</td>
</tr>
<tr>
<td>Base conc. (in $Q_3$)</td>
<td>0.003 M NaOH</td>
</tr>
<tr>
<td>Buffer conc. (in $Q_3$)</td>
<td>0.00005 M NaHCO$_3$</td>
</tr>
<tr>
<td>Buffer conc. (in $Q_2$)</td>
<td>0.03 M NaHCO$_3$</td>
</tr>
</tbody>
</table>

---

Fig. 5. Full and pruned networks for the simulation study.
Table 4. They reveal a significant reduction in the complexity of the network for all cases. The removal of a large number of synaptic weights increased the prediction accuracy of the model, as can be seen in Table 4. All indexes have similar values for OLS, OBD and OBS pruning algorithms for training and validation phases but better values than full network. This indicates that the eliminated synaptic weights were indeed redundant. However, the performance of the OLS method can be considered better than the performance of OBD and OBS methods, because it was just as accurate but more parsimonious and computationally efficient. Results in Table 4 show that the proposed method was almost eight times faster than the OBS method and more than nine times faster than the OBD method.

Fig. 6 shows also the performance of the full FNN and the OLS pruned FNN outputs for training and validation data. The LSODAR solver (Hindmarsh, 1980) was used for integrating of the initial value problem represented by Eq. (29) to predict the outputs. The accuracy of these predictions is remarkable because they are multiple-ahead-predictions; only initial conditions from the physical model and the input sequence were used. In addition, the capability of the neural model to predict the steady-state behavior was tested on the titration curve. The steady-state conditions of the physical and neural models were calculated using the continuation method.

<table>
<thead>
<tr>
<th>No. of parameters</th>
<th>Training phase</th>
<th>Validation phase</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No. of parameters</td>
<td>r²</td>
</tr>
<tr>
<td>Full</td>
<td>61</td>
<td>0.980</td>
</tr>
<tr>
<td>OLS</td>
<td>28</td>
<td>0.994</td>
</tr>
<tr>
<td>OBD</td>
<td>27</td>
<td>0.990</td>
</tr>
<tr>
<td>OBS</td>
<td>31</td>
<td>0.993</td>
</tr>
</tbody>
</table>

The dynamic response of the simulation study is shown in Fig. 6. (a) Random uniform step sequence for Q; (b) Model response; (c) Full and OLS pruned NN pH residuals.
(Kubicek & Marek, 1983). The titration curves in Fig. 7 shows that good results can be obtained with the neural models and better results were obtained with the pruned model instead of the full one, due to better generalization capability. The accuracy of the predicted titration curve is remarkable, considering that there were no steady-state data in the training and the validation databases. The prediction of titration curve is very important for control purposes since information about changes in steady-state gain can be obtained from this curve.

4.2.2. Experimental applications

In this case a multiple-input single-output pH process is considered. The process studied in this section is a part of a complex experimental pH neutralization system, which was designed by Hall and Seborg (1989) to serve as a demonstration unit for the experimental evaluation of advanced control and identification techniques. The stirred-tank system shown in Fig. 4 has three inlet streams — an acid stream (HNO$_3$), a base stream (NaOH with a trace amount of NaHCO$_3$) and a buffer stream (NaHCO$_3$) — and one effluent stream. The liquid level in the tank is allowed to vary freely, thus introducing time-varying flow dynamics. Additional time delay is present due to transportation delay of approximately 30 s associated with the pH measurement device. Nominal operating conditions are given in Table 3. A detailed description of the experimental system can be found in Johnson (1995). The experimental system was excited using simultaneously random uniform step sequences for $Q_1$ and $Q_3$, with a step probability equal to 0.9 and...
Table 5

<table>
<thead>
<tr>
<th>No. of parameters</th>
<th>Training phase CPU time ratio</th>
<th>Validation phase CPU time ratio</th>
<th>r²</th>
<th>FPE</th>
<th>LILC</th>
<th>AIC</th>
<th>BIC</th>
<th>OLS</th>
<th>OBS</th>
<th>OBD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>0.910</td>
<td>1</td>
<td>76</td>
<td>0.800</td>
<td>911</td>
<td>1073</td>
<td>-591</td>
<td>-786</td>
<td>-1442</td>
<td>-1372</td>
</tr>
<tr>
<td>OLS</td>
<td>0.932</td>
<td>2</td>
<td>27</td>
<td>0.942</td>
<td>1565</td>
<td>1520</td>
<td>-1511</td>
<td>-1587</td>
<td>-1697</td>
<td>-1750</td>
</tr>
<tr>
<td>OBD</td>
<td>0.962</td>
<td>3</td>
<td>24</td>
<td>0.952</td>
<td>1798</td>
<td>1738</td>
<td>-1750</td>
<td>-1877</td>
<td>-1951</td>
<td>-1877</td>
</tr>
<tr>
<td>OBS</td>
<td>0.975</td>
<td>4</td>
<td>29</td>
<td>0.975</td>
<td>1999</td>
<td>1951</td>
<td>-1951</td>
<td>-2037</td>
<td>-2117</td>
<td>-2037</td>
</tr>
</tbody>
</table>

Fig. 9. Dynamic response of the UCSB neutralization process. (a) Random uniform step sequence for $Q_1$ and $Q_3$; (b) Experimental response; (c) Full and OLS pruned NN pH residuals.

Due to the 30 s time delay in pH measurement, the network inputs are $Q_1(t-2), Q_3(t-2)$ and pH($t$), and the network output is $d[pH(t)]/dt$. The experimental pH signal was filtered using a first-order digital filter and derivatives of experimental pH signal were calculated numerically by finite-difference schemes. An initial neural network with 15 hidden nodes was trained until convergence using the Levenberg–Marquardt method; then it was subjected to the proposed OLS, OBD and OBS pruning algorithms. The results shown in Fig. 8 and Table 5 reveal a significant reduction in the complexity of the network for all these methods. The removal of a large number of synaptic weights increased the models prediction accuracy, as can seen from Table 5. This indicates that the eliminated synaptic weights were indeed redundant. Performance of all methods were similar to previous example and the performance of the proposed OLS method can be considered better than the performance of the OBD and OBS methods because it was more accurate for the validation phase; in addition, it was computationally more efficient. Results in Table 5 show that the OLS method was approximately 16 times faster than the OBS and
OBD methods. It is possible to conclude that the computational effort of the OLS method is much less sensitive to the dimension of the neural network than the other methods. This feature becomes more attractive when considering high dimension neural network applications.

Fig. 9 also shows the performance of the full and OLS pruned FNNs outputs for training and validation data. Similar to the simulated example, these outputs were generated by integrating the initial value problem represented by Eq. (29). Therefore, these outputs are multiple-ahead-predictions because only initial conditions from experimental data were used. Even for the experimental application, the accuracy of multiple-ahead-prediction is remarkable. In addition, the capability of the neural model to predict the steady-state behavior was tested, in comparison with experimental data, by calculating the titration curve of the system using the continuation method (Kubicek & Marek, 1983). These results can be seen in Fig. 10. Again, good neural network predictions were obtained with the pruned neural network model, probably due to better generalization capability. The accuracy of the predicted titration curve is remarkable, once there is no steady state information in the training and validation databases.

5. Conclusions

A new pruning algorithm to determine the model structure in neural networks has been proposed. The algorithm can be applied to FNNs with any activation function. Results indicated that the algorithm is very efficient and accurately determines redundant and insignificant network parameters, thus allowing parsimonious FNN models. Statistical criteria confirmed that pruned NN models are more accurate than full models for predicting data not used in the training phase. The main advantages of this algorithm are:

- A large decrease in the number of weights, that implies in a drastic reduction network training time. This training time is often a major concern in many applications in chemical engineering;
- A large decrease in computational time for pruning procedure compared with traditional algorithms such as OBD and OBS.

The proposed pruning algorithm has performed well for both static and dynamic models and both simulation studies and experimental applications. These results demonstrate that the proposed OLS pruning algorithm is a very promising method for determining the optimal size and topology of neural networks, for use in applications such as process identification and control.

### Notation

- A: upper triangular matrix
- AIC: Akaike’s information criterion
- B: auxiliary regressor matrix
- BIC: Bayesian information criteria
- \(d_1\): bias vector of the hidden layer
- \(d_2\): bias vector of the output layer
- D: diagonal matrix
- E: mean square error of the network
- \(E_{com}\): complexity function
- \(E_w\): dual objective function
- f: neural activation function
- FPE: final prediction error
- g: auxiliary parameter vector
- \(h_{ij}\): elements of Hessian matrix
- \(j_i\): components of the gradient of \(E\) with respect to weights of neural network
- \(K_{a1}\): first carbonic acid equilibrium constant
- \(K_{a2}\): second carbonic acid equilibrium constant
- \(K_w\): water equilibrium constant
- LILC: Kinchin’s law of iterated logarithm criterion
- M: number of regressors of the linear regression function
- n: number of data used for OLS estimation
- \(N_h\): number of hidden nodes
- \(N_k\): number of active weights arriving at “k” node
- P: regressor matrix
- p: number of input system
- q: number of output system
- \(Q_1\): acid flow rate
- \(Q_2\): buffer solution flow rate
- \(Q_3\): base flow rate
- \(Q_4\): effluent flow rate
- \(r^2\): performance index
$r_A^2$ corrected multiple correlation coefficient
$s_y$ saliency of weight $w_k$
$t$ time
$u$ system input vector
$v$ intermediate signal of neural network after passing through the activation function
$v'$ intermediate signal of neural network before passing through the activation function
$V_R$ volume of the reactor
$W_1$ synaptic weight matrix of the first layer
$W_2$ synaptic weight matrix of the second layer
$W_a$ invariant of reaction in $i$th stream defined by Eq. (31)
$W_b$ invariant of reaction in $i$th stream defined by Eq. (32)
$y$ system or neural network output vector
$y'$ output signal of neural network before passing through the activation function
$z_i$ dependent variable or a term to regress upon

Greek letters
$A_{ik}$ elements of upper triangular matrix $A$
$\delta$ relevance of a node of a neural network
$\epsilon_j$ deviation between actual system output and estimated system output
$\theta_j$ unknown parameters to be estimated
$\lambda$ modeling error parameter
$\xi_1^2$ modeling error variance

References


