Computers chem. Engng, Vol. 16, No. 4, pp. 379–391, 1992
Printed in Great Britain.

0098-1354/92 $5.00 + 0.00

NONLINEAR PLS MODELING USING NEURAL NETWORKS

S. J. QIN and T. J. MCAVOY†
Chemical Process Systems Laboratory, Department of Chemical Engineering, University of Maryland, College Park, MD 20742, U.S.A.

(Received 20 March 1991; final revision received 27 September 1991; received for publication 10 December 1991)

Abstract—This paper discusses the embedding of neural networks into the framework of the PLS (partial least squares) modeling method resulting in a neural net PLS modeling approach. By using the universal approximation property of neural networks, the PLS modeling method is genealized to a nonlinear framework. The resulting model uses neural networks to capture the nonlinearity and keeps the PLS projection to attain robust generalization property. In this paper, the standard PLS modeling method is briefly reviewed. Then a neural net PLS (NNPLS) modeling approach is proposed which incorporates feedforward networks into the PLS modeling. A multi-input–multi-output nonlinear modeling task is decomposed into linear outer relations and simple nonlinear inner relations which are performed by a number of single-input–single-output networks. Since only a small size network is trained at one time, the over-parametrized problem of the direct neural network approach is circumvented even when the training data are very sparse. A conjugate gradient learning method is employed to train the network. It is shown that, by analyzing the NNPLS algorithm, the global NNPLS model is equivalent to a multilayer feedforward network. Finally, applications of the proposed NNPLS method are presented with comparison to the standard linear PLS method and the direct neural network approach. The proposed neural net PLS method gives better prediction results than the PLS modeling method and the direct neural network approach.

1. INTRODUCTION

Statistical data analysis finds wide application in establishing a model from experimental or historical data. A typical application of this technique is to relate quality indices of a product to all possible variables which can affect the quality. Suppose that $y_i (i = 1, 2, \ldots, p)$ are p different quality indices, $x_j (j = 1, 2, \ldots, m)$ are m causal variables, and there are n samples of data being observed. Two data matrices can be formulated as follows:

$$\mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1p} \\ y_{21} & y_{22} & \cdots & y_{2p} \\ \cdots & \cdots & \cdots & \cdots \\ y_{n1} & y_{n2} & \cdots & y_{np} \end{bmatrix} \in \mathscr{R}^{n \times p},$$

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix} \in \mathscr{R}^{n \times m},$$

in which each row is composed of one observation. The data analysis problem is to relate the matrix **Y** as some function of the matrix **X** so as to predict **Y** (e.g. quality) using the data of **X**.

A simple approach is to assume a linear relation between **Y** and **X**:

$$\mathbf{Y} = \mathbf{XC} + \mathbf{E}, \tag{1}$$

then find the coefficient matrix **C** by the ordinary least-squares method as follows:

$$\mathbf{C} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}. \tag{2}$$

However, there are two cases in practice where the matrix $\mathbf{X}^T\mathbf{X}$ is not invertible: (i) the input variables (columns of **X**) are correlated; and (ii) the number of samples n is *smaller* than the number of input variables m. In the case when n is larger but not *much* larger than m, the regression model could also be distorted by measurement noise and therefore give poor generalization on new data. Hence, in the case of correlated inputs and/or limited observations, the ordinary least-squares method fails to give a model which is robust to noise.

In order to have a robust model which generalizes, a partial least-squares (PLS) method was proposed by Wold (1966). The PLS method projects the data down to a number of principal factors and then models the factors by 1-D linear regressions. Since the dimension of each factor is one, the problems of correlation and limited observations are circumvented. Application of the PLS method includes

†To whom all correspondence should be addressed.

many practical problems such as interpreting fluorescence spectra (Lindberg *et al.*, 1983; McAvoy *et al.*, 1989), analyzing detergent quality (Fuller *et al.*, 1988), modeling of quality indices for cosmetic data (Wold *et al.*, 1989), and analyzing glass spectral data (Haaland and Thomas, 1988). A detailed and easy-to-understand description of the PLS algorithm is given by Geladi and Kowalski (1986a,b), while theoretical insight into PLS is revealed in Manne (1987) and Höskuldsson (1988).

Although the PLS regression method provides a good remedy for the problems of correlated inputs and limited observations, its major restriction is that only linear information can be extracted from the data. Since many practical data are inherently nonlinear in nature, it is desirable to have a robust method which can model any nonlinear relation. As a successful step towards nonlinear PLS modeling, a quadratic PLS modeling method was proposed by Wold *et al.* (1989). In their approach a quadratic inner model is used and the model is derived iteratively. However, the nonlinearity of the quadratic PLS method is still very limited. A more general form of nonlinearity was suggested by the authors, but no details are given in the paper. Another shortcoming of the method is that it is complicated, especially when more complex nonlinearity is employed. Therefore, more generic and relatively simple approaches are highly desirable.

Multilayer neural networks (Werbos, 1974; Rumelhart *et al.*, 1986), as a fast developing research area, have been proven by many researchers (Hornik *et al.*, 1989; Cybenko, 1989) as universal approximators of any continuous function with arbitrarily desired accuracy. Using the universal approximation property, a direct neural network approach (e.g. McAvoy *et al.*, 1989; Piovoso and Owens, 1991) is proposed to relate matrices **Y** and **X** by a neural network as:

$$\mathbf{Y} = \mathcal{N}(\mathbf{X}) + \mathbf{E}, \tag{3}$$

where **E** is the residual matrix after regression. $\mathcal{N}(\cdot)$ stands for the nonlinear map performed by the network. Unfortunately, although the direct network approach performs a lot better than linear techniques in some cases, it has similar problems to the ordinary least-squares method in the case of correlated inputs and/or limited data. Particularly in the case of limited data, the number of weights in a multilayer network of m inputs and p outputs could be larger than the number of observations. Therefore, some of the weights cannot be uniquely determined from the observed data. In this case the direct network approach leads to overfitting.

From the previous description, it is very desirable to integrate the PLS regression and neural networks to formulate an approach which can handle nonlinearity, correlated inputs and limited observations. As the scope of this paper, we propose a neural net PLS (NNPLS) method which has the potential of modeling any continuous nonlinear relation and also attaining the robust properties of PLS regression. First of all, the standard PLS modeling algorithm is briefly reviewed. Then a generic framework of the NNPLS method is proposed, which incorporates a feedforward network into the PLS modeling. Using the NNPLS method, a multi-input–multi-output nonlinear modeling task can be decomposed into linear outer relations and univariate nonlinear inner relations, which are approximated by a number of single-input–single-output (SISO) neural networks. Thirdly, the network training algorithms are discussed. In principle, the network can be trained by either patternwise or batchwise backpropagation learning rules, both of which are shown to give rise to the same model if a small learning rate is specified (Qin *et al.*, 1991). Here, in the NNPLS method, we find it convenient to use a conjugate gradient method (Leonard and Kramer, 1990) because: (i) the learning is much faster; and (ii) the learning rate is calculated automatically and it need not to be specified in advance. The number of hidden units of each inner network is determined by a simplified cross-validation scheme. Since only a SISO network is trained at one time, the over-parametrized problem of the direct network approach is circumvented in the case of limited data. It is shown that the global NNPLS model is equivalent to a multilayer feedforward network, except the weights are determined by the NNPLS algorithm. The weight calculation consists of eigenvector calculation and a conjugate gradient training. Finally, applications of the proposed NNPLS method are presented for a set of cosmetic data and for a fluorescence data interpretation. The NNPLS results are compared to the results of the standard linear PLS method and the direct neural net approach.

2. LINEAR PLS ALGORITHM

The linear PLS method is basically a particular multilinear regression algorithm which can handle correlated inputs and limited data. This algorithm is directly related to principal component analysis (Jolliffe, 1986). Here we briefly discuss the idea of the linear PLS method so as to establish a background for presenting the NNPLS method. Those readers who are interested in the detailed PLS algorithm are referred to the literature (e.g Geladi and Kowalski, 1986a; Martens and Næs, 1989; Höskuldsson, 1988).

The linear PLS decomposes matrices **X** and **Y** into bilinear products plus residual matrices:

$$\mathbf{X} = \mathbf{t}_l \mathbf{p}_l^T + \mathbf{E}_l, \quad (4)$$

$$\mathbf{Y} = \mathbf{u}_l \mathbf{q}_l^T + \mathbf{F}_l, \quad (5)$$

where $\mathbf{t}_l, \mathbf{u}_l \in \mathscr{R}^n$ are score vectors of the first principal factor, and $\mathbf{p}_l \in \mathscr{R}^m$, $\mathbf{q}_l \in \mathscr{R}^p$ are loading vectors corresponding to this factor. All the four vectors are determined such that the residual matrices $\mathbf{E}_l$ and $\mathbf{F}_l$ are minimized. One can see that the above decomposition is very similar to calculating the first principal components for **X** and **Y**, but PLS does the calculation in a particular way (Geladi and Kowalski, 1986a). The above two equations formulate a PLS outer model. After the outer calculation, the score vectors are related by a linear inner model:

$$\mathbf{u}_l = b_l \mathbf{t}_l + \mathbf{r}_l, \quad (6)$$

where b_l is a coefficient which is determined by minimizing the residual $\mathbf{r}_l$. After going through the above calculation, the residual matrices are calculated as:

$$\text{for matrix } \mathbf{X}, \quad \mathbf{E}_l = \mathbf{X} - \mathbf{t}_l \mathbf{p}_l^T, \quad (7)$$

$$\text{for matrix } \mathbf{Y}, \quad \mathbf{F}_l = \mathbf{Y} - b_l \mathbf{t}_l \mathbf{q}_l^T. \quad (8)$$

Then the second factor is calculated based on the residuals $\mathbf{E}_l$ and $\mathbf{F}_l$ by going through the same procedure as for the first factor. The same procedure is repeated until the last factor a is calculated, which leaves almost no information in the residual matrices $\mathbf{E}_h$ and $\mathbf{F}_h$. The overall calculation is schematically depicted in Fig. 1.

The number of factors a is usually determined by cross-validation. Theoretically, if more and more factors are used, one eventually gets the least-squares solution for the training data. However, the goal of building a PLS model is to give predictions on new observations. If too many factors are used, the model would fit the training samples too well, and therefore it fits the noise in the training data as well. The cross-validation method is used to avoid over-fitting on training data. A typical way of doing cross-validation is to leave one or several samples out at a time, and then train the model with the remaining data. After training, the model is tested on the samples which are not used in training. This procedure is repeated until every sample has been left out once. Summing up all the test errors over each factor, the optimal number of factors is chosen as the location of the minimum of the sum of test errors. One can see that the cross-validation method is quite laborious, but it is useful in determining the number of factors. More details of the method can be found in Stone (1978) and Geladi and Kowalski (1986b).

It can be shown that the PLS algorithm is a particular way of doing multiple linear regression which has robust prediction properties. As a matter of fact, the PLS technique is widely applied, whether or not the data are limited or correlated. Recent work of Stone and Brooks (1990) shows that ordinary least squares, partial least squares and principal component regression are simply three particular cases of what is called *continuum regression*. The advantages of using the PLS algorithm is that the outer model transforms the original data into latent factors, by which the orthogonality of the score vectors is achieved (Höskuldsson, 1988).

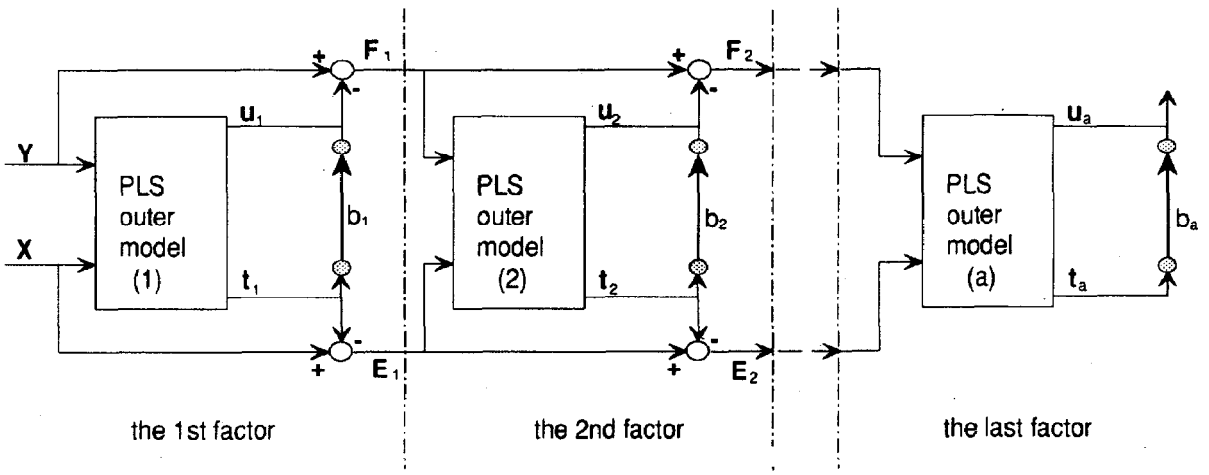

Fig. 1. A schematic illustration of the PLS modeling method: each inner model (b_i relationship) is performed by a linear regressor.

3. NNPLS MODELING METHOD

Since data in practice are usually nonlinear in nature, it is more desirable to have a nonlinear PLS modeling approach which can represent any nonlinear relationship and still attain the robust generalization property of the PLS approach. Keeping the outer relation in linear PLS so as to have the robust prediction property, we propose an NNPLS modeling approach which uses neural networks as the inner regressors:

$$\mathbf{u}_h = \mathcal{N}(\mathbf{t}_h) + \mathbf{r}_h, \tag{9}$$

where $\mathcal{N}(\cdot)$ stands for the nonlinear relation represented by a neural network. The NNPLS method is schematically shown in Fig. 2. The PLS outer transform is kept to generate score variables from the data. Then the scores ($\mathbf{u}_h$ and $\mathbf{t}_h$) are used to train the inner network models. In general, any type of neural network which performs nonlinear maps from the input to the output can be incorporated into the NNPLS framework. These networks can be multilayer feedforward networks, radial basis functions (Poggio and Girosi, 1990) or recurrent networks (Werbos, 1988), etc.

The NNPLS method differs from the direct network approach in that the data are not directly used to train the neural networks but are preprocessed by the PLS outer transform. It is this transform that decomposes a multivariate regression problem into a number of univariate regressors. Each regressor is implemented by a neural network in this method. A direct benefit of doing so is that only a SISO network is trained at a time. The number of weights to be determined is much smaller than that in an m-input–p-output problem when the direct network approach is used. By reducing the number of weights down to a smaller number, the over-parametrized problem is circumvented. Also, the number of local minima is expected to be fewer owing to the use of a smaller size network.

An advantage of using neural networks as the inner regressors is due to their nonlinear approximation property. It has been proven that the mapping performed by a neural network, $\mathcal{N}(\cdot)$, can approximate any continuous function with arbitrarily desired accuracy. Hornik *et al.* (1989) prove that a network with only one hidden layer of sigmoidal units is enough to have universal approximation properties. Later on, Hornik *et al.* (1990) show that not only the function itself, but also its derivatives can be approximated by a one-hidden-layer network. Recent work of Huang and Huang (1991) show that the number of hidden units of such a one-hidden-layer network is bounded. Owing to the simplicity and the universal approximation property of such a network, it will be used to build a nonlinear inner model to implement a nonlinear PLS approach. This type of network has one *sigmoidal* hidden layer and one *linear* output layer.

The NNPLS algorithm can be constructed based on the NNPLS framework shown in Fig. 2. It is basically composed of the PLS outer transform and inner network training schemes. Keeping the PLS outer transform the same as in the linear PLS method, the NNPLS algorithm can be formulated as follows:

1. Scale $\mathbf{X}$ and $\mathbf{Y}$ to zero-mean and one-variance. Let $\mathbf{E}_0 = \mathbf{X}$, $\mathbf{F}_0 = \mathbf{Y}$ and $h = 1$.
2. For each factor h, take $\mathbf{u}_h =$ some $\mathbf{y}_j$.

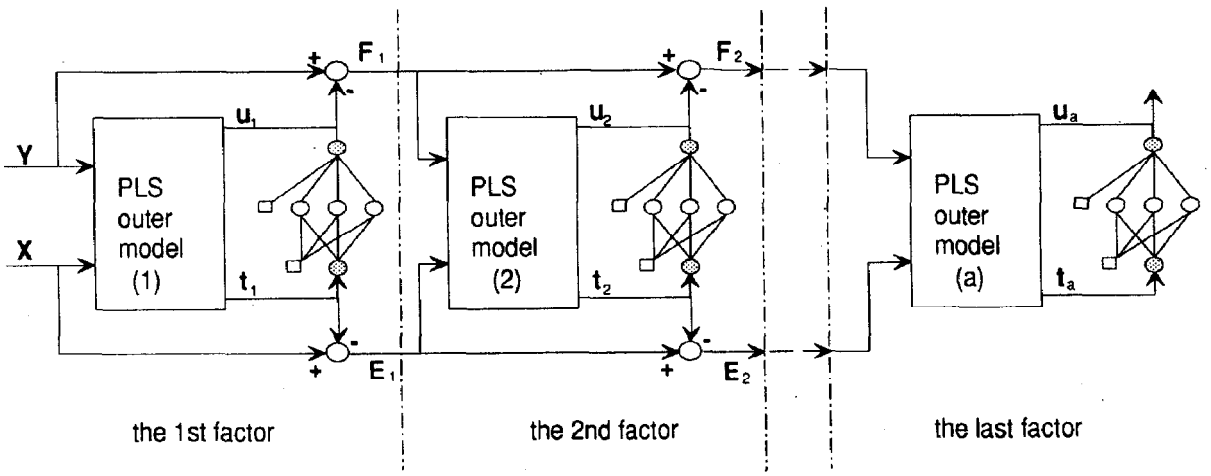

Fig. 2. A schematic illustration of the NNPLS method: the data are transformed to latent scores, then neural networks are used to learn the scores.

3. PLS outer transform:

   • in matrix **X**:

$$\mathbf{w}_h^T = \arg_{\mathbf{w}_h} \min \|\mathbf{E}_{h-1} - \mathbf{u}_h \mathbf{w}_h^T\|^2 = \mathbf{u}_h^T \mathbf{E}_{h-1} / \mathbf{u}_h^T \mathbf{u}_h,$$

normalize $\mathbf{w}_h$ to norm 1. (10)

$$\mathbf{t}_h = \arg_{\mathbf{t}_h} \min \|\mathbf{E}_{h-1} - \mathbf{t}_h \mathbf{w}_h^T\|^2 = \mathbf{E}_{h-1} \mathbf{w}_h; \quad (11)$$

   • in matrix **Y**:

$$\mathbf{q}_h^T = \arg_{\mathbf{q}_h} \min \|\mathbf{F}_{h-1} - \mathbf{t}_h \mathbf{q}_h^T\|^2 = \mathbf{t}_h^T \mathbf{F}_{h-1} / \mathbf{t}_h^T \mathbf{t}_h,$$

normalize $\mathbf{q}_h$ to norm 1. (12)

$$\mathbf{u}_h = \arg_{\mathbf{u}_h} \min \|\mathbf{F}_{h-1} - \mathbf{u}_h \mathbf{q}_h^T\|^2 = \mathbf{F}_{h-1} \mathbf{q}_h.$$

Iterate this step until it converges. (13)

4. Calculate the **X** loadings and rescale the variables:

$$\mathbf{p}_h^T = \arg_{\mathbf{p}_h} \min \|\mathbf{E}_{h-1} - \mathbf{t}_h \mathbf{p}_h^T\|^2 = \mathbf{t}_h^T \mathbf{E}_{h-1} / \mathbf{t}_h^T \mathbf{t}_h, \quad (14)$$

$$\text{normalize } \mathbf{p}_h: \quad \mathbf{p}_h := \mathbf{p}_h / \|\mathbf{p}_h\|, \quad (15)$$

$$\mathbf{t}_h := \mathbf{t}_h \|\mathbf{p}_h\|, \quad (16)$$

$$\mathbf{w}_h := \mathbf{w}_h \|\mathbf{p}_h\|. \quad (17)$$

5. Find the inner network model: train the inner network such that the following error function is minimized.

$$J_h = \|\mathbf{u}_h - \mathcal{N}(\mathbf{t}_h)\|^2. \quad (18)$$

   A conjugate gradient training method is used in the algorithm and details of the training are described in the next section.

6. Calculate the residuals for factor h:

$$\text{for matrix } \mathbf{X}, \quad \mathbf{E}_h = \mathbf{E}_{h-1} - \mathbf{t}_h \mathbf{p}_h^T, \quad (19)$$

$$\text{for matrix } \mathbf{Y}, \quad \mathbf{F}_h = \mathbf{F}_{h-1} - \hat{\mathbf{u}}_h \mathbf{q}_h^T, \quad (20)$$

   where

$$\hat{\mathbf{u}}_h = \mathcal{N}(\mathbf{t}_h). \quad (21)$$

7. Let $h := h + 1$, return to Step 2 until all principal factors are calculated.

The number of factors h could be determined by cross-validation. However, since the regular cross-validation method is too laborious in this case which involves training neural networks, a simplified cross-validation method will be used which splits the data into two data sets: one set is for training, the other set is for testing. The number of factors is chosen such that the model gives the minimum prediction error for the test set. It should be noted that, similar to the linear PLS method, the number of factors in the NNPLS modeling is not always well-defined. One can end up with different number of factors when different training and test sets are used. In the case of the NNPLS method, the number of factors is also related to the complexity of the inner networks being used. Too few hidden units would require more factors to extract all useful information from the data. On the other hand, too many hidden units would overfit the data. Therefore, it is important to choose the optimal number of the hidden units. We will discuss this issue in the next section.

4. INNER NETWORK TRAINING

4.1. *Choosing a sigmoidal function*

It has been shown by Stinchcombe and White (1989) that many types of nonlinear functions in the hidden layer can allow the network to be a universal approximator. Therefore, one can choose different types of nonlinear activation functions. Usually, a sigmoidal function valued from zero to one is used. In this particular application, the network is used to model the relation between $\mathbf{u}_h$ and $\mathbf{t}_h$ as shown in equation (9). It is known that both $\mathbf{u}_h$ and $\mathbf{t}_h$ have the following property (Geladi and Kowalski, 1986a):

$$\sum_{i=1}^{n} u_{hi} = 0, \quad (22)$$

$$\sum_{i=1}^{n} t_{hi} = 0, \quad (23)$$

where u_{hi} and t_{hi} are the ith elements of $\mathbf{u}_h$ and $\mathbf{t}_h$, respectively. Therefore, we choose the following centered sigmoid:

$$\sigma(z) = \frac{1 - e^{-z}}{1 + e^{-z}}, \quad (24)$$

to model the inner relation because zero input to the sigmoid gives zero output. Note that the derivative of the centered sigmoid is:

$$\sigma'(z) = \tfrac{1}{2}[1 - \sigma^2(z)]. \quad (25)$$

These relations are useful in deriving the learning rules for networks with centered sigmoidal functions.

4.2. *Choosing a learning algorithm*

The neural networks used in the NNPLS method can be trained by the generalized delta learning rule (Rumelhart *et al.*, 1986). However, one has to specify the learning rate for each network if the generalized delta rule is used. Here, we use the conjugate gradient learning method (Leonard and Kramer, 1990; Fletcher and Powell, 1963; Lasdon *et al.*, 1967) to train the network for the following reasons: (i) the learning speed of the conjugate gradient method is much faster than backpropagation; and (ii) the learning rate constants are calculated automatically and adaptively so that they do not need to be specified before training. It is the second point that makes the conjugate gradient method most convenient for the NNPLS method.

The conjugate gradient method is presented in Fletcher and Powell (1963). This method considers not only the descent direction, but also the curvature of the error surface. Hence, it is faster than the gradient descent method. The use of the conjugate gradient method to train a feedforward network is given in Leonard and Kramer (1990), in which the relationship between the conjugate gradient method and the generalized delta rule is also revealed. Denoting all the weights in a network as one vector, the conjugate gradient algorithm in Lasdon *et al.* (1967) is followed to train the inner networks in the NNPLS method.

4.3. *Determining the number of hidden units*

A remaining issue in designing the inner networks is to determine how many hidden units are required for each nonlinear factor. In general, the number of hidden units is associated with the complexity of the functional mapping from the input to the output. A more complex input–output relationship would normally require more hidden units. However, it is not true that the more hidden units being used, the better the generalization is. Too many hidden units would result in an over-parametrized model, while too few hidden units would under-parametrize the problem. Therefore, it is important to properly choose the number of hidden units. In the NNPLS method, we use the simplified cross-validation scheme to determine the number of hidden units. Since the data have been divided into a training set and a test set, what one needs to do is to generate scores for the test set, $\mathbf{u}_h^{test}$ and $\mathbf{t}_h^{test}$, as shown in Fig. 3:

$$\mathbf{t}_h^{test} = \mathbf{E}_{h-1}^{test}\mathbf{w}_h, \tag{26}$$

$$\mathbf{u}_h^{test} = \mathbf{F}_{h-1}^{test}\mathbf{q}_h, \tag{27}$$

where

$$\mathbf{E}_h^{test} = \mathbf{E}_{h-1}^{test} - \mathbf{t}_h^{test}\mathbf{p}_h^T; \quad \mathbf{E}_0^{test} = \mathbf{X}^{test}, \tag{28}$$

$$\mathbf{F}_h^{test} = \mathbf{F}_{h-1}^{test} - \mathcal{N}(\mathbf{t}_h^{test})\mathbf{q}_h^T; \quad \mathbf{F}_0^{test} = \mathbf{Y}^{test}, \tag{29}$$

where $\mathbf{w}_h$, $\mathbf{p}_h$ and $\mathbf{q}_h$ have been calculated in the NNPLS algorithm. Note that the scores for the training set, $\mathbf{u}_h$ and $\mathbf{t}_h$, have already been generated in the NNPLS algorithm. Therefore, we propose to start an inner network with one hidden unit and train it on the training pairs $\mathbf{u}_h$ and $\mathbf{t}_h$, then test if more hidden units are required based on the prediction error on the test pairs, $\mathbf{u}_h^{test}$ and $\mathbf{t}_h^{test}$. By eventually adding more hidden units to the network, one can find the optimal number of hidden units which gives the best prediction error for the test pairs.

4.4. *Initializing the inner networks*

By using the conjugate gradient training method, the network will go the nearest local minimum from the initial point. Therefore, it is possible for the network to go to different local minima by varying the initial weights. A usual approach is to choose the initial weights randomly, by which one can use different initial weights to find out which local minimum is better. However, one would have to keep trying to find a better minimum and save the initial weights to make the result repeatable. In the NNPLS approach, we propose a linear scheme which initializes the weights by the linear PLS result. We use the best linear model between $\mathbf{u}_h$ and $\mathbf{t}_h$ to initialize the first hidden unit of the network. Additional hidden units will be initialized with small random numbers. As learning proceeds, the neural network will find a minimum which is better than the initial linear model. Although the linear initialization technique could not guarantee a global minimum, it gives a repeatable and reasonable solution.

5. ANALYSIS OF THE NNPLS METHOD

It is known that the linear PLS method is a specific and robust approach to multiple linear regression. In this section, we will show that the NNPLS model is equivalent to a multilayer feedforward network but is trained specifically by the NNPLS algorithm. In order to make the presentation easier, the following notation is defined.

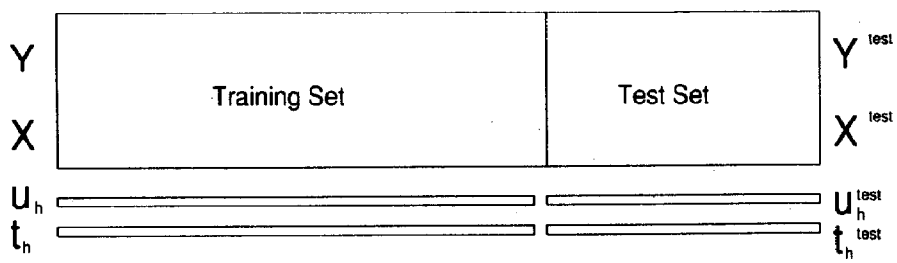

Fig. 3. Generate scores for the training and test sets in order to determine the number of hidden units using cross-validation.

Definition 1

For any matrix $\mathbf{A} \in \mathscr{R}^{m \times n}$, a nonlinear operator $\sigma(\cdot)$: $\mathscr{R}^{m \times n} \rightarrow \mathscr{R}^{m \times n}$ is defined as:

$$\text{element}_{ij} \text{ of } \sigma(\mathbf{A}) = \sigma(\text{element}_{ij} \text{ of } \mathbf{A}), \quad (30)$$

where $\sigma(\cdot)$ is a univariate nonlinear function, which could be a sigmoidal function.

Note that the above definition is also valid when $\mathbf{A}$ is a vector or a scalar. The property of this operator is given in the following lemma.

Lemma 1

Let $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_a] \in \mathscr{R}^{n \times a}$ and $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_a]^T \in \mathscr{R}^{a \times m}$ then:

$$\sigma(\mathbf{A})\mathbf{B} = \sum_{h=1}^{a} \sigma(\mathbf{a}_h)\mathbf{b}_h^T. \quad (31)$$

Proof

Using Definition 1, we have:

$$\sigma(\mathbf{A})\mathbf{B} = [\sigma(\mathbf{a}_1), \sigma(\mathbf{a}_2), \ldots, \sigma(\mathbf{a}_h)][\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_a]^T = \sum_{h=1}^{a} \sigma(\mathbf{a}_h)\mathbf{b}_h^T. \quad (32)$$

Q.E.D.

Using the above notation, the inner network model used in the NNPLS method [equation (21)] can be represented as:

$$\hat{\mathbf{u}}_h = \mathscr{N}(\mathbf{t}_h) = \sigma(\mathbf{t}_h\boldsymbol{\omega}_{1h}^T + \mathbf{e}\boldsymbol{\beta}_{1h}^T)\boldsymbol{\omega}_{2h} + \mathbf{e}\beta_{2h}, \quad (33)$$

where $\boldsymbol{\omega}_{1h} \in \mathscr{R}^{n_h}$ and $\boldsymbol{\omega}_{2h} \in \mathscr{R}^{n_h}$ are the weight vectors for the input and output layers, and $\boldsymbol{\beta}_{1h} \in \mathscr{R}^{n_h}$ and β_{2h} are the bias weights for the input and output layers, respectively. n_h is the number of hidden units for the hth inner network model. $\mathbf{e} \in \mathscr{R}^n$ is a vector in which all the elements are 1. The notation is also depicted in Fig. 4. By analyzing the NNPLS algorithm, the relationship between the matrix $\mathbf{X}$ and its score vector $\mathbf{t}_h$ is revealed in the following lemma.

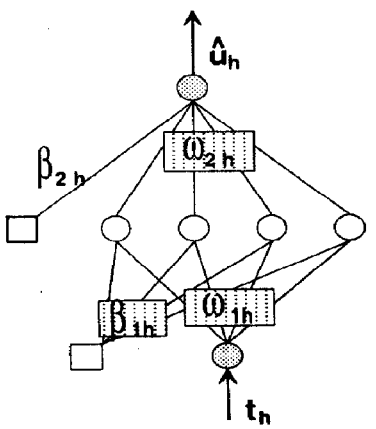

Fig. 4. Notation for the inner network used in the NNPLS method.

Lemma 2

The score vector $\mathbf{t}_h$ is related to $\mathbf{X}$ through the following relation:

$$\mathbf{t}_h = \mathbf{X}\tilde{\mathbf{w}}_h, \quad (34)$$

where

$$\tilde{\mathbf{w}}_h = \prod_{j=1}^{h-1} (\mathbf{I} - \mathbf{w}_j\mathbf{p}_j^T)\mathbf{w}_h \quad (35)$$

and $\mathbf{I} \in \mathscr{R}^{m \times m}$ is the identity matrix.

Proof

The proof of this lemma is omitted here. An equivalent proof can be found in Höskuldsson (1988).

The global NNPLS model is derived in the following theorem.

Theorem 1

The global NNPLS model relates the input matrix $\mathbf{X}$ and the output matrix $\mathbf{Y}$ through the following relation:

$$\mathbf{Y} = \sigma(\mathbf{X}\boldsymbol{\Omega}_1 + \mathbf{e}\boldsymbol{\beta}_1^T)\boldsymbol{\Omega}_2 + \mathbf{e}\boldsymbol{\beta}_2^T + \mathbf{F}_a, \quad (36)$$

where

$$\boldsymbol{\Omega}_1 \triangleq [\tilde{\mathbf{w}}_1\boldsymbol{\omega}_{11}^T, \tilde{\mathbf{w}}_2\boldsymbol{\omega}_{12}^T, \ldots, \tilde{\mathbf{w}}_a\boldsymbol{\omega}_{1a}^T] \in \mathscr{R}^{m \times N_{\text{hid}}}, \quad (37)$$

$$\boldsymbol{\Omega}_2 \triangleq \begin{bmatrix} \omega_{21}\mathbf{q}_1^T \\ \omega_{22}\mathbf{q}_2^T \\ \vdots \\ \omega_{2a}\mathbf{q}_a^T \end{bmatrix} \in \mathscr{R}^{N_{\text{hid}} \times p}, \quad (38)$$

$$\boldsymbol{\beta}_1^T \triangleq [\boldsymbol{\beta}_{11}^T, \boldsymbol{\beta}_{12}^T, \ldots, \boldsymbol{\beta}_{1a}^T] \in \mathscr{R}^{N_{\text{hid}}}, \quad (39)$$

$$\boldsymbol{\beta}_2 \triangleq \sum_{h=1}^{a} \beta_{2h}\mathbf{q}_h \in \mathscr{R}^p \quad (40)$$

and

$$N_{\text{hid}} \triangleq \sum_{h=1}^{a} n_h. \quad (41)$$

In words, the global NNPLS model of equation (36) is equivalent to an m-input–p-output network with N_{hid} hidden units. The network weights are given in equations (37–40).

Proof

Equation (20) in the NNPLS algorithm can be rewritten as below by iteration:

$$\mathbf{Y} = \sum_{h=1}^{a} \hat{\mathbf{u}}_h\mathbf{q}_h^T + \mathbf{F}_a. \quad (42)$$

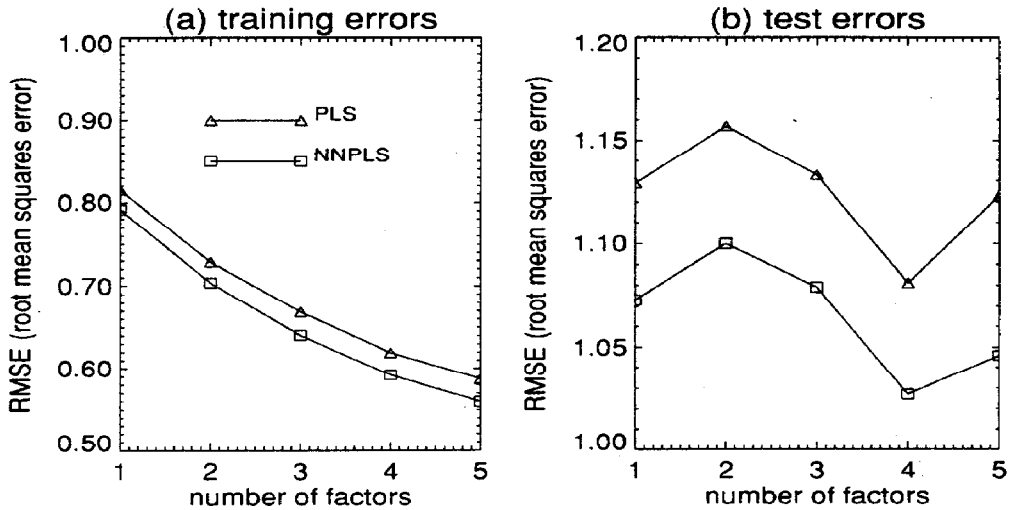

Fig. 5. Cross-validation result for the cosmetic data: (a) training errors by the NNPLS and PLS methods; and (b) prediction errors by the NNPLS and PLS methods.

By using equation (33) and Lemma 2, the above equation becomes:

$$\mathbf{Y} = \sum_{h=1}^{a} [\sigma(\mathbf{X}\tilde{\mathbf{w}}_h\omega_{1h}^{\mathrm{T}} + \mathbf{e}\beta_{1h}^{\mathrm{T}})\omega_{2_h}\mathbf{q}_h^{\mathrm{T}} + \mathbf{e}\beta_{2h}\mathbf{q}_h^{\mathrm{T}}] + \mathbf{F}_a. \tag{43}$$

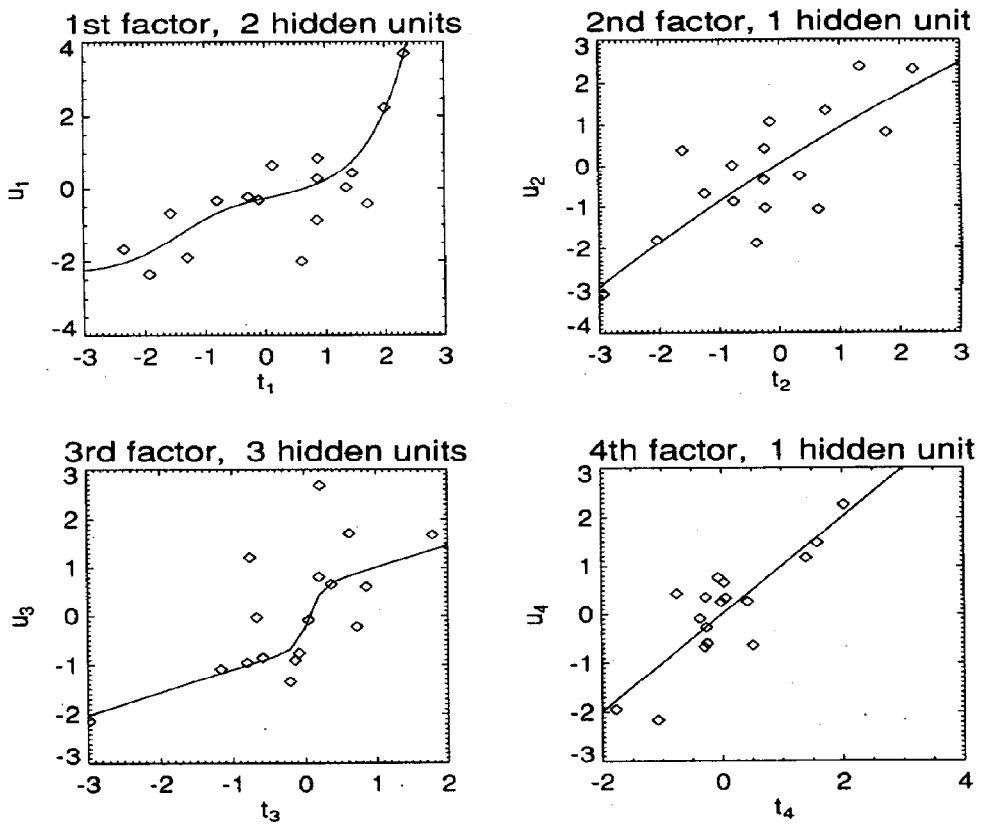

Fig. 6. Principal inner models for the cosmetic data analysis using the NNPLS method. Scores (◇); inner models (——).

Applying Lemma 1 and using the notation in equations (37–40), equation (43) can be further simplified as:

$$\mathbf{Y} = \sigma(\mathbf{X}[\tilde{\mathbf{w}}_1\omega_{11}^{\mathrm{T}}, \tilde{\mathbf{w}}_2\omega_{12}^{\mathrm{T}}, \ldots, \tilde{\mathbf{w}}_a\omega_{1a}^{\mathrm{T}}] + \mathbf{e}[\boldsymbol{\beta}_{11}^{\mathrm{T}}, \boldsymbol{\beta}_{12}^{\mathrm{T}}, \ldots, \boldsymbol{\beta}_{1a}^{\mathrm{T}})] \begin{bmatrix} \omega_{21}\mathbf{q}_1^{\mathrm{T}} \\ \omega_{22}\mathbf{q}_2^{\mathrm{T}} \\ \vdots \\ \omega_{2a}\mathbf{q}_a^{\mathrm{T}} \end{bmatrix} + \mathbf{e}\sum_{h=1}^{a} \beta_{2h}\mathbf{q}_h^{\mathrm{T}} + \mathbf{F}_a, \quad (44)$$

$$= \sigma(\mathbf{X}\boldsymbol{\Omega}_1 + \mathbf{e}\boldsymbol{\beta}_1^{\mathrm{T}})\boldsymbol{\Omega}_2 + \mathbf{e}\boldsymbol{\beta}_2^{\mathrm{T}} + \mathbf{F}_a, \quad (45)$$

which is the final NNPLS model. One can easily see that the global NNPLS model is equivalent to a network with input layer weights $\boldsymbol{\Omega}_1$, output layer weights $\boldsymbol{\Omega}_2$, input bias $\boldsymbol{\beta}_1$ and output bias $\boldsymbol{\beta}_2$.

Q.E.D.

This theorem is of both theoretical and practical interest. On one hand, it reveals that the NNPLS modeling method is a specific type of network modeling approach which has good generalization properties. It should be noted that, like the linear PLS method, the NNPLS method does specify a particular structure for the model, which prevents us from using more than necessary parameters. On the other hand, the theorem gives explicitly all the weights of the global NNPLS model in equations (37–40). One could assemble the inner network models into a global network. The calculation of the network weights consists of eigenvector calculations and regular network training. The global network obtained will perform an identical mapping as the NNPLS model and the global network is a multilayer feed-forward network.

6. APPLICATION EXAMPLES

6.1. *Analysis of cosmetic data*

A set of cosmetics qualimetrics data given in Wold *et al.* (1989) in their presentation of their nonlinear PLS algorithm is used here to test the effectiveness of the NNPLS method. The data are generated from a test that 17 different formulations of face creams applied to 10 women models. Overall 11 quality indicators are evaluated by specialized evaluators during the test. The data analysis task in this project is to establish a model between the cream compositions and the quality indicators, so that the cream

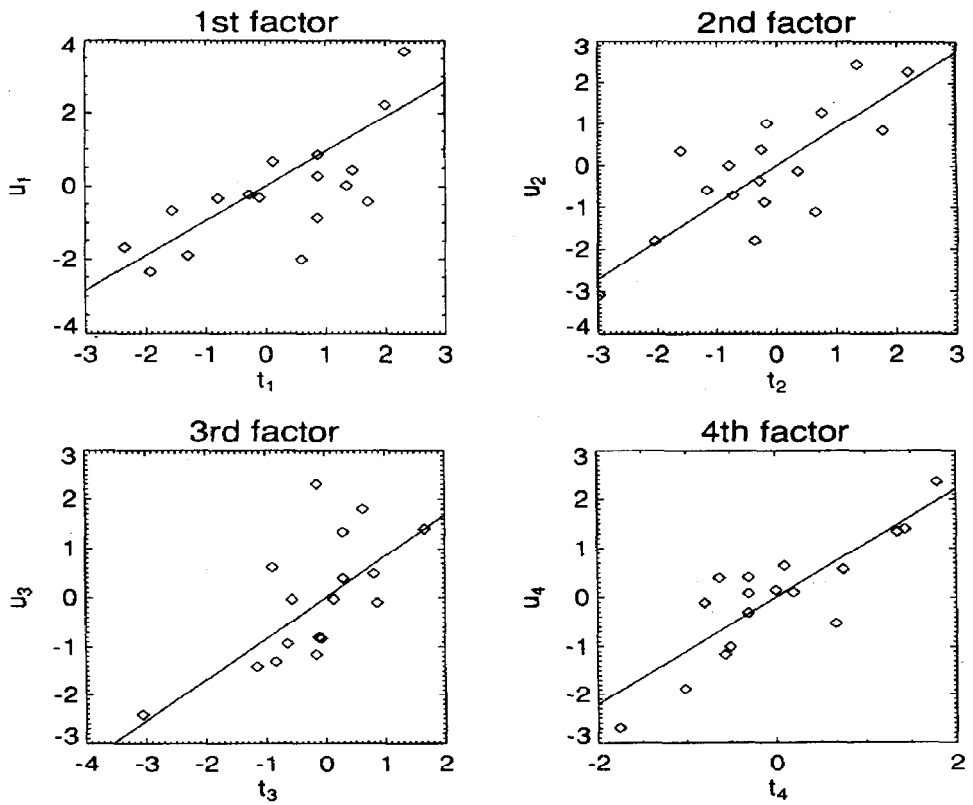

Fig. 7. Principal inner models for the cosmetic data analysis using the linear PLS method. Scores (◇); inner models (——).

Table 1. Training and test RSMEs (root mean square errors) for the fluorescence data using neural net approach

No. of hidden units	Training RMSE	Test RMSE
1	0.9185	0.7495
2	0.2480	0.6478
3	0.4631×10^{-3}	0.2626
4	0.2875×10^{-11}	0.2657

Table 2. Comparison of predicted relative RMSEs for the test fluorescence data set by three different methods

	Neural net (%)	Linear PLS (%)	NNPLS (%)
Output 1	49.95	28.51	15.26
Output 2	20.23	18.77	13.25

quality can be predicted from the cream composition. In this data set, the number of samples $n = 17$, the dimension of the input vector (compositions) $m = 8$ and the dimension of the output vector (quality indicators) $p = 11$. This is a typical example in which the number of samples is not significantly larger than the number of the variables. As a consequence, even if a regular network with only two hidden units is used, it would be over-parametrized and the weights of the network could not be uniquely determined.

In the application of the NNPLS modeling to this example, the data are divided into two groups: the first 11 samples are used as the training set and the remaining 6 samples are used for testing. The simplified cross-validation is used to determine the optimal number of factors. Each factor is modeled using a SISO network, of which the number of hidden units is also determined automatically by simplified cross-validation. Figure 5a shows the root mean square errors during the training phase, while Fig. 5b depicts the test errors vs the number of factors. It turns out that both the NNPLS model and the PLS model give the best predictions when four factors are used. It can also be seen that the prediction error of the NNPLS method is smaller than that of the PLS method corresponding to the optimal number of factors, $h = 4$.

Figure 6 shows the four principal inner relations by the NNPLS model. The latent scores for the first and third factors are nonlinear. Therefore, the NNPLS algorithm requires two hidden units for the first factor and three hidden units for the third factor, respectively, while the second and the fourth factors are almost linear and each requires only one hidden unit. It is seen that the NNPLS approach is capable of representing linear and nonlinear relationships.

For the sake of comparison, the principal inner relations using the PLS method are plotted in Fig. 7. One can see easily that, although the PLS regression gives a best linear least-squares model, it is not able to pick up the nonlinearity underlying the data. This fact explains why the NNPLS method is superior to the linear PLS method.

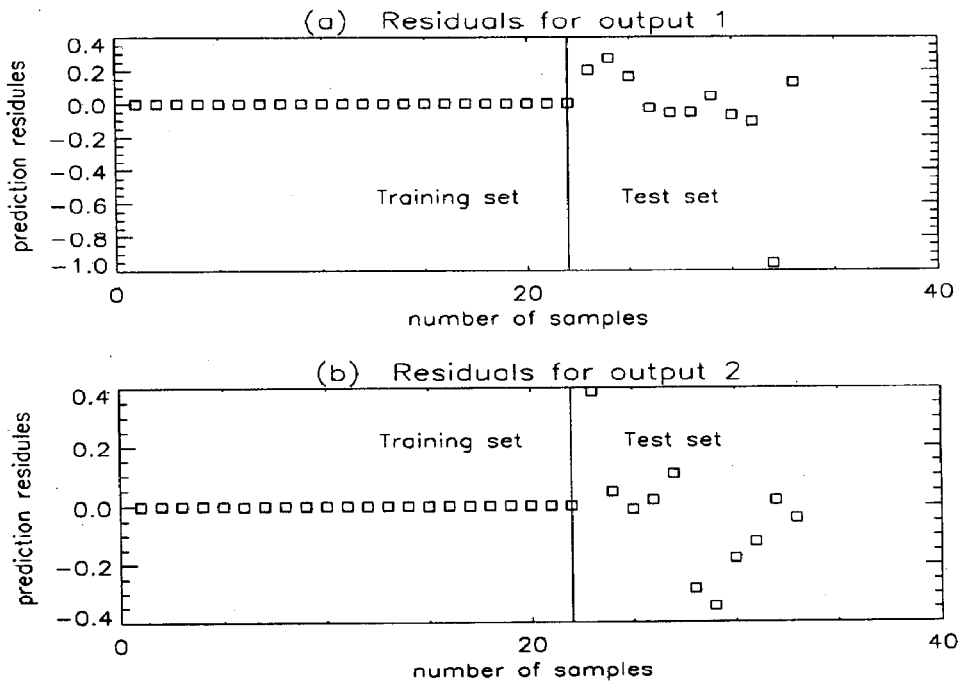

Fig. 8. Training and prediction residuals for the fluorescence data using the direct network approach. Overfitting of the training data is observed for both outputs.

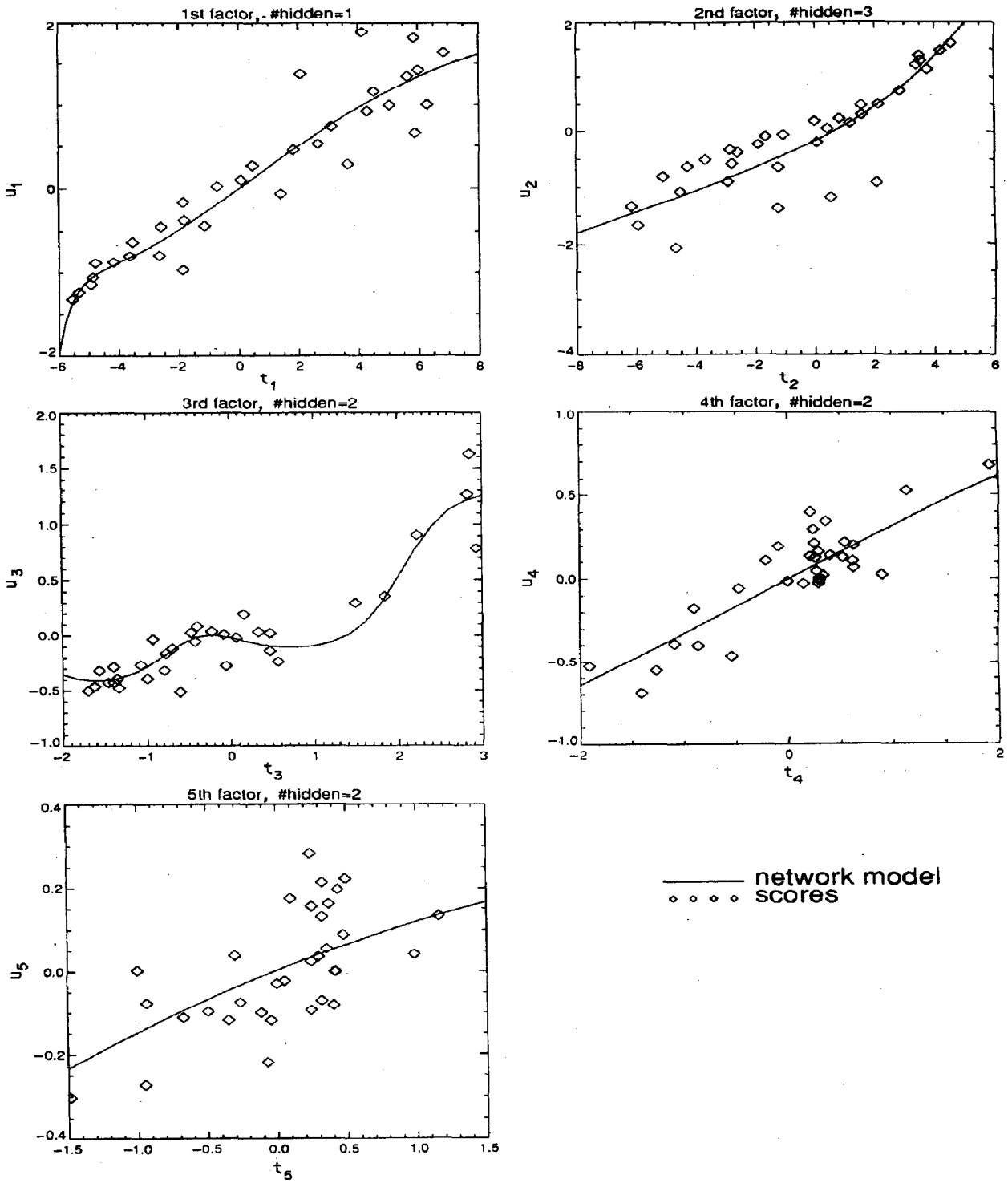

Fig. 9. Principal inner models in the fluorescence data analysis using the NNPLS method.

6.2. *Deconvolution of fluorescence spectra*

The objective of deconvoluting fluorescence spectra here is to estimate bioconcentrations via fluorescence spectra obtained from solutions of amino acid fluorophores. The data consist of 30 inputs which are the measured fluorescence intensity at various wavelengths, and 2 outputs which are the mole fraction and molarity of the solution. Since the concentration ranges from 10^{-7} to 10^{-4} molar, it is transformed into a logarithmic scale in order to handle very small numbers. Overall 33 samples were collected. This fluorescence data set is another real example where the number of samples is very limited. We will show in this application how the direct neural network approach overfits the data and thus results in poor generalization. Also, we will show how the NNPLS method outperforms the direct network approach and the linear PLS method.

A two-layer network with one sigmoidal hidden layer is used to interpret the fluorescence data in the direct network approach. In order to choose the optimal number of hidden units, the simplified cross-validation method is employed. The first 22 samples are used as the training set and the last 11 samples as the test set. The training and test errors vs the number of hidden units are given in Table 1. Cross-validation shows that three hidden units give the best prediction error. However, the test error is much worse than the training error. The reason is that the over-parametrized network fits the training data too well and results in poor generalization on the test data. A clear picture of the overfitting can also be seen in Fig. 8, where the residuals of the two outputs are depicted. The training residual for each training sample is almost zero, but the residuals for the test set are significantly far from zero. The physical significance is that the network overfits the training data and passes exactly through the training samples. As a result, it cannot generalize on new data.

For the purpose of comparison, the PLS and NNPLS methods are applied to the same data under the same conditions. It is found by cross-validation that the PLS model requires 7 factors, while the NNPLS model requires 5 factors. Comparison of the relative root-mean-square errors (RRMSEs) of the test set for each individual output are listed in Table 2. The relative RMSE is defined as:

$$\text{RRMSE}_i = \left[\frac{1}{n}\sum_{j=1}^{n}\left(\frac{y_{ij}-\hat{y}_{ij}}{y_{ij}}\right)^2\right]^{1/2} \times 100\%.$$

It can be seen from Table 2 that the NNPLS method gives the best prediction errors for all outputs. The inner relation performed by the NNPLS method is depicted in Fig. 9, which shows how the NNPLS method captures the nonlinearity and thus outperforms the linear PLS method. Another good feature of the NNPLS method over the direct network approach is that the nonlinear relationship can be visualized by plotting the inner relations, which help us to inspect the quality of the fit.

7. CONCLUSION

Although the linear PLS modeling method can give a robust linear model from sparse data, it is not capable of modeling nonlinearity. By using the universal approximation property of a feedforward neural network, a neural net PLS (NNPLS) approach is proposed which is capable of modeling nonlinear relations and also attaining the robust generalization properties of PLS regression. By decomposing a multivariable modeling task into a number of small modeling problems, the NNPLS approach is able to circumvent the over-parametrized problem of the direct network approach. Since only a small size network is used to learn the inner relations, the network training task becomes smaller and easier. Furthermore, the quality of modeling can be made visible by plotting the inner relations, which can be used for further analysis such as outlier detection. The analysis of the NNPLS approach shows that the global NNPLS model is equivalent to a feedforward network. Analysis of two real data sets shows that the NNPLS method is superior to the linear PLS method and the direct network approach in achieving a higher fidelity model from the data used.

NOMENCLATURE

a = Optimal number of factors
b_h = Regression coefficient for an inner model
$\mathbf{e}$ = Vector of which all elements are one
$\mathbf{f}(\cdot)$ = General matrix-valued nonlinear function
h = Dummy index for factors
$\mathbf{r}_h$ = Residual vector of an inner model
i = Dummy index
j = Dummy index
m = Dimension of the input vector
n = Number of samples
n_h = Number of hidden units in the hth inner network model
p = Dimension of the output vector
$\mathbf{p}_h$ = ($\in \mathcal{R}^m$) Loading vector for matrix $\mathbf{X}$
$\mathbf{q}_h$ = ($\in \mathcal{R}^p$) Loading vector for matrix $\mathbf{Y}$
$\mathbf{t}_h$ = ($\in \mathcal{R}^n$) Score vector of matrix $\mathbf{X}$
t_{hi} = ith Element for vector $\mathbf{t}_h$
$\mathbf{u}_h$ = ($\in \mathcal{R}^n$) Score vector of matrix $\mathbf{Y}$
u_{hi} = ith Element for vector $\mathbf{u}_h$
$\hat{\mathbf{u}}_h$ = ($\in \mathcal{R}^n$) Network prediction for $\mathbf{u}_h$
$\mathbf{w}_h$ = ($\in \mathcal{R}^m$) Weight vector for matrix $\mathbf{X}$
$\tilde{\mathbf{w}}_h$ = ($\in \mathcal{R}^m$) Defined in equation (35)
$\mathbf{y}_j$ = ($\in \mathcal{R}^n$) Column of matrix $\mathbf{Y}$
$\mathbf{C}$ = ($\in \mathcal{R}^{m\times p}$) Regression matrix
$\mathbf{E}$ = ($\in \mathcal{R}^{n\times m}$) Residual matrix in regression
$\mathbf{E}_h$ = ($\in \mathcal{R}^{n\times m}$) Residual matrix for matrix $\mathbf{X}$
$\mathbf{F}_h$ = ($\in \mathcal{R}^{n\times p}$) Residual matrix for matrix $\mathbf{Y}$

J_h = Error function for the inner networks
N_{hid} = Number of hidden units in the global NNPLS model
$\mathbf{X}$ = ($\in \mathcal{R}^{n \times m}$) Data matrix for input variables
$\mathbf{Y}$ = ($\in \mathcal{R}^{n \times p}$) Data matrix for output variables
$\boldsymbol{\beta}_{1h}$, $\boldsymbol{\beta}_{2h}$ = Bias weight vectors for hth inner network model
$\boldsymbol{\beta}_1$, $\boldsymbol{\beta}_2$ = Bias weight vectors for the global NNPLS model
$\sigma(\cdot)$ = Centered sigmoidal function
$\boldsymbol{\sigma}(\cdot)$ = Matrix-valued nonlinear operator (Definition 1)
ω_{1h}, ω_{2h} = Weight vectors for hth inner network model
$\boldsymbol{\Omega}_1$, $\boldsymbol{\Omega}_2$ = Weight matrices for the global NNPLS model
$(\cdot)^T$ = Transpose of a vector or a matrix
$\mathcal{N}(\cdot)$ = Mapping performed by a neural network
$\| \ \|$ = Euclidean norm
All boldface small letters are column vectors
All boldface capital letters are matrices

REFERENCES

Cybenko G., Approximation by superpositions of a sigmoidal function. *Maths Controls, Signals Syst.* **2,** 303–314 (1989).

Fletcher R. and M. J. D. Powell, A rapid descent method for minimization. *Comput. J.* **3,** 163–367 (1963).

Fuller M. P., G. L. Ritter and C. S. Draper, Partial least-squares quantitative analysis of infrared spectroscopic data. Part I: algorithm implementation; Part II: application to detergent analysis. *Appl. Spectrosc.* **42,** 217–236 (1988).

Geladi P. and B. R. Kowalski, Partial least-squares regression: a tutorial. *Analyt. Chim. Acta* **185,** 1–17 (1986a).

Geladi P. and B. R. Kowalski, An example of 2-block predictive partial least-squares regression with simulated data. *Analyt. Chim. Acta* **185,** 19–32 (1986b).

Haaland D. M. and E. V. Thomas, Partial least-squares methods for spectral analysis: 1. Relation to other quantitative calibration methods and the extraction of qualitative information. 2. Application to simulated and glass spectral data. *Analyt. Chem.* **60,** 1193–1208 (1988).

Hornik K., M. Stinchcombe and H. White, Multilayer feedforward neural networks are universal approximators. *Neural Networks* **2,** 359–366 (1989).

Hornik K., M. Stinchcombe M. and H. White, Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks* **3,** 551–560 (1990).

Höskuldsson A., PLS regression methods. *J. Chemomet.* **2,** 211–228 (1988).

Huang S.-C. and Y.-F. Huang, Bounds on the number of hidden neurons in multilayer perceptrons. *IEEE Trans. Neural Networks* **2,** 47–55 (1991).

Kowalski B., R. Gerlach and H. Wold, Chemical systems under indirect observation. In *Systems Under Indirect Observation* (K. Jöreskog and H. Wold, Eds), pp. 191–209. North-Holland, Amsterdam (1982).

Jolliffe I. T., *Principal Component Analysis.* Springer-Verlag, New York (1986).

Lasdon L. S., S. K. MItter and A. D. Warren, The conjugate gradient method for optimal control problems. *IEEE Trans. Automat. Control* **AC-12,** 132–138 (1967).

Leonard J. and M. K. Kramer, Improvement of the backpropagation algorithm for training neural networks. *Computers chem. Engng* **14,** 337–341 (1990).

Lindberg W., J. Persson and S. Wold, Partial least-squares method for spectrofluorimetric analysis of mixtures of humic and ligninsulfonate. *Analyt. Chem.* **55,** 643–648 (1983).

Manne R., Analysis of two partial least squares algorithms for multivariate calibration. *Chemomet. Intell. Lab. Syst.* **2,** 187–197 (1987).

Martens H. and T. Næs, *Multivariate Calibration.* Wiley, New York (1989).

McAvoy T. J., N. S. Wang, N. Naidu, N. V. Bhat, J. Gunter and M. Simmons, Interpreting biosensor data via backpropagation. *Int. Joint Conf. Neural Networks,* Vol. I, pp. 227–233, Washington, DC (1989).

Piovoso M. J. and A. J. Owens, Sensor data analysis using artificial neural networks. *Int. Conf. Chem. Process Control,* Texas (1991).

Poggio T. and F. Girosi, Networks for approximation and learning. *Proc. IEEE* **78,** 1481–1497 (1990).

Qin S., H.-T. Su and T. J. McAvoy, Comparison of four neural net learning methods for dynamic system identification. *IEEE Trans. Neural Networks* (1991).

Stinchcombe M. and H. White, Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions. *Int. Joint Conf. on Neural Networks,* Vol. I, pp. 613–617. Washington, DC (1989).

Rumelhart D., G. Hinton and R. Williams, *Parallel Distributed Processing,* Chap. 8. MIT Cambridge, MA (1986).

Stone M., Cross-validation: a review. *Math. Operationsforsch. Statist., Ser. Statist.* **9**(1) (1978).

Stone M. and R. J. Brooks, Continuum regression: cross-validated sequentially constructed prediction embracing ordinary least squares, partial least squares and principal component regression. *J.R. Statist. Soc. Ser. B* **52,** 237–269 (1990).

Werbos P. J., Beyond regression: new tools for prediction and analysis in the behavioral science. Ph.D. Dissertation, Harvard University, Cambridge, MA (1974).

Werbos P. J., Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks* **1,** 339–356 (1988).

Wold H., Nonlinear estimation by iterative least squares procedures. In *Research Papers in Statistics* (F. David, Ed.). Wiley, New York (1966).

Wold S., N. Kettaneh-Wold and B. Skagerberg, Nonlinear PLS modeling. *Chemomet. Intell. Lab. Syst.* **7,** 53–65 (1989).