Pergamon

0097-8485(95)00085-2

Computers Chem. Vol. 20, No. 4, pp. 439–448, 1996

0097-8485/96 $15.00 + 0.00

PARALLEL PROCESSING OF CHEMICAL INFORMATION IN A LOCAL AREA NETWORK—II. A PARALLEL CROSS-VALIDATION PROCEDURE FOR ARTIFICIAL NEURAL NETWORKS

E. P. P. A DERKS,* M. L. M. BECKERS, W. J. MELSSEN and L. M. C. BUYDENS

Laboratory for Analytical Chemistry, Faculty of Science, Catholic University of Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands

(Received 19 June 1995; in revised form 31 August 1995)

Abstract—This paper describes a parallel cross-validation (PCV) procedure, for testing the predictive ability of multi-layer feed-forward (MLF) neural networks models, trained by the generalized delta learning rule. The PCV program has been parallelized to operate in a local area computer network. Development and execution of the parallel application was aided by the HYDRA programming environment, which is extensively described in Part I of this paper. A brief theoretical introduction on MLF networks is given and the problems, associated with the validation of predictive abilities, will be discussed. Furthermore, this paper comprises a general outline of the PCV program. Finally, the parallel PCV application is used to validate the predictive ability of an MLF network modeling a chemical non-linear function approximation problem which is described extensively in the literature. Copyright © 1996 Elsevier Science Ltd

1. INTRODUCTION

Multi-layer feed-forward (MLF) neural networks are often applied in chemistry as non-linear function approximators or as classifier systems. Their flexibility and powerful modeling abilities have proven to be appealing features for solving problems in various fields of academic as well as industrial research environments. Neural networks are generally easy to implement in software and little a priori knowledge concerning the modeling process is required. However, the "black-box" concept of neural networks does not allow any analytical model validation, as opposed to, e.g., standard least-square based regression methods. This makes the neural network approach less attractive for situations in which the reliability of the model needs to be quantified in a statistically sound way.

In chemometrics, cross-validation has become a commonly used technique for estimating the number of latent variables (pseudo-rank) of a calibration model, especially in situations where only a limited number of measurements is available. Since no separate test set is used, cross-validation can be considered as an internal validation method for creating a model with optimal predictive abilities. A model is considered optimal when the best compromise is found between the accuracy of fit (bias) and generalization ability (variance).

The estimation of the pseudo-rank for a calibration model can be compared to the estimation of the optimal number of hidden units† for a neural network model.

Since neural networks have proven to be very powerful non-linear function approximators, the risk of overfitting, yielding a poor generalization ability, becomes even more critical. Non-random components in observational noise and fluctuations due to experimental conditions are easily modeled by neural networks when a superfluous number of hidden units is used, leading to poor predictive abilities.

Applying conventional cross-validation on a neural network based model, obtained from sparse data, will inevitably yield unreliable results since neural networks suffer from the fact that the models obtained are, in general, unreproducible. This is a result of the fact that training involves a gradient descent search in an error hyperplane containing, apart from the global minimum, many local minima. Hence, considering the huge number of possible network configurations and ways to initialize the weight values of a network, it is highly unlikely that a set of weights (which determine together with the network architecture the actual model) obtained from a single training session will correspond to the best fitting model (i.e. the model associated with the global minimum in the error hyperplane).

In this paper, an approach is presented to allow cross-validation on neural networks in order to

*Author for correspondence.

†The modeling ability of a neural network depends as well on the choice of activation functions as on the number of hidden units. However, in this work only sigmoidal activation functions are assumed.

estimate the optimal number of hidden units. Our approach is based on the assumption that the probability of finding the global minimum, corresponding to a unique set of weights, increases when multiple starting positions on the error hyperplane are used for the gradient descent search. Although no guarantee can be given that the global minimum will be found, at least the probability of getting stuck in local minima will be reduced considerably.

Since the method described above, requires a lot of computing time and administration (for example, scheduling and monitoring the progress of the training sessions), a parallel extension of the conventional cross-validation procedure has been developed.

Since this work focuses on the implementation of the PCV procedure for MLF networks, a brief introduction about MLF neural networks and cross-validation is given. [For technical details of the parallel program environment HYDRA, the reader is referred to Melssen *et al.* (1996).] Additionally, an application of the parallel cross-validation procedure on a real-world dataset, taken from de Weijer *et al.* (1992), describing the relation between the physical structure and the properties of PET yarns, will be discussed.

2. THEORY

2.1. MLF networks

Artificial neural networks (ANN) are based on concepts of the behavior of the human brain. Although artificial neural networks are primitive compared to their biological counterparts, they exhibit some interesting properties which make them useful as multivariate tools in various fields of research. During the last decade, ANN have been successfully applied in non-linear modeling, classification, signal processing and process control (White & Sofge, 1992).

Various types of neural networks are known, based on different functionalities. The multi-layered structured networks are popular and widely applied in the field of chemistry. In Fig. 1 the architecture of such a feed-forward network is shown. The MLF networks generally consist of three layers. The number of neurons in the input and output layer is determined by the dimension (i.e. the number of variables) of the input and output objects, respectively.

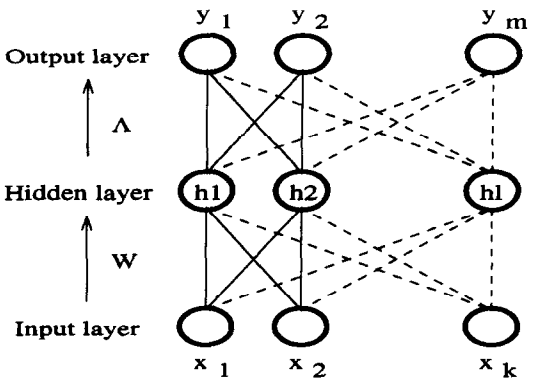

Fig. 1. The feed-forward weight connections of MLF networks.

Given a dataset $[X; Y]$, where the matrix $X(N \times K)$ contains N vectors consisting of K input variables and where the matrix $Y(N \times M)$ contains N vectors comprising M output variables, a general equation for output unit m of a three layered feed-forward network is given by

$$\hat{y}_{im} = \sum_{j=1}^{L} \lambda_j \Psi(x_i \cdot w_j + w_{0_j}) \tag{1}$$

where L represents the number of hidden units, x_i is the input vector, w_j is the vector of the jth hidden unit, w_{0_j} is the corresponding bias and λ_j is the associated weight vector connecting the jth hidden unit to the output unit. For notational convenience the bias term is omitted in the following equations by adding it as an extra column vector to the weight matrix W.

The sum of the weighted input is transformed by an activation function Ψ, usually the sigmoid function:

$$\Psi(x \cdot w) = \frac{1}{1 + \exp(-(x \cdot w))}. \tag{2}$$

When all data are used simultaneously, equation (1) can be written in matrix form

$$\hat{Y}_m = \Psi(X^T W) \cdot \Lambda. \tag{3}$$

Here, W represents the input weight matrix $(K \times L)$, Λ represents the $(L \times M)$ output weight matrix and Ψ is the matrix containing the activation operators.

The neural network can be trained by minimizing the differences between the predicted and actual outputs as specified by the Root Mean Squared Error (RMSE)

$$\epsilon_m = \sqrt{\sum_{i=1}^{N} \frac{(y_{im} - \hat{y}_{im})^2}{N}}. \tag{4}$$

The convergence process can be followed by monitoring the overall RMSE (ϵ_0), giving an average indication about the variances of residual errors on the output units, by normalizing the errors on the M output units

$$\epsilon_0 = \sqrt{\sum_{m=1}^{M} \frac{\epsilon_m^2}{M}}. \tag{5}$$

The global minimum can be searched by means of gradient descent methods like the generalized delta learning rule (McClelland & Rumelhart, 1988; White & Sofge, 1992) or optimization techniques like genetic algorithms (Schaffer *et al.*, 1992) or simulated annealing (Amato *et al.*, 1991; Bos, 1993). For the MLF networks, gradient descent techniques have become mandatory since these are theoretically well understood and easy to implement in software. However, a severe drawback is that local minima are encountered. In the conventional backpropagation

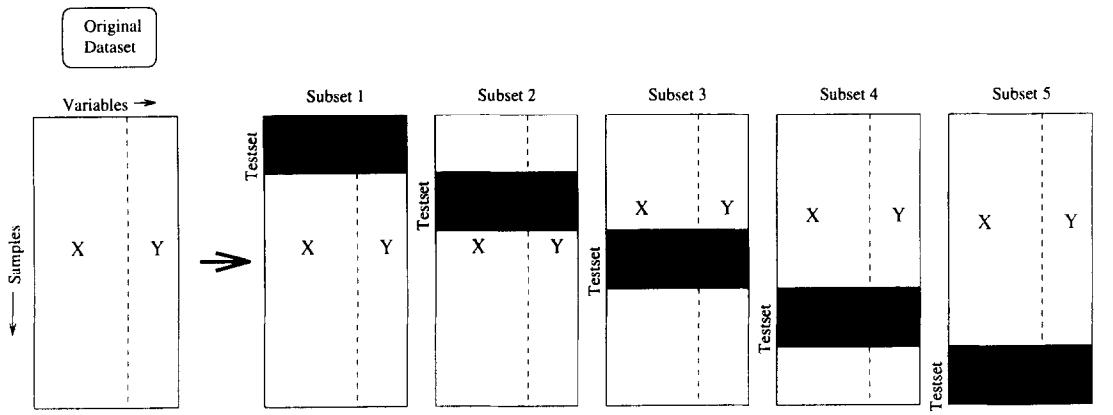

Fig. 2. Subdivision of the data into training and test sets for cross-validation.

algorithm, training is performed by random selection and propagation of patterns from a finite-sized training set yielding a finite number of pathways to escape from a local minimum. Hence, in practice, networks trained by gradient descent methods often fail to converge to the global minimum.

One solution to solve this problem is to incorporate a stochastic element in the search process. This approach introduces an additional random factor to the training process, in order to provide a means to escape from local minima. Commonly, noise to the inputs or weights is added. These techniques are referred to as noise injection (Holmström & Koistinen, 1992; Abunawass & Owen, 1993). An additional side effect of noise injection is the improved generalization capability (Abunawass & Owen, 1993).

Another way to avoid local minima is simply by choosing various random start positions (i.e. weights) for the gradient descent search.

2.2. *Cross-validation*

In spite of the fact that a neural network or any other non-linear modeling technique requires a large number of training samples, in practice, only a limited number of samples can be obtained. Consequently, the problem arises that insufficient samples are available to create a training and test set. The additional test set is required to estimate the optimal number of units in the hidden layer of the neural network. Cross-validation uses a number of subsets of the data for creating training and test sets. The principles of cross-validation have extensively been described (Allen, 1974; Stone, 1974; Geisser, 1975; Wold, 1978) so only the general concept will be described.

The cross-validation data are obtained by dividing the data (size N) into Q training and test sets (training set (size $N - A$), test set (size A), whereas ($A \approx N/Q$)).

Then, the network is being trained and the outputs of the A test samples are used to calculate the predicted residual sum of squares (*PRESS*) represented by equation (6)

$$PRESS = \sum_{i=1}^{A} (y_i - \hat{y}_i)^2. \quad (6)$$

The subdivision of the data into training and test sets is shown graphically in Fig. 2.

This procedure is repeated Q times. The prediction errors are accumulated for every test set yielding the total *PRESS*. Given the number of hidden units, the predicted residual sum of squares yields a statistic to test the predictive ability of the neural network. Cross-validation on an increasing number of hidden units generally shows that the *PRESS* of the training set continuously decreases whereas the *PRESS* of the test set increases with the number of hidden units, as is graphically presented in Fig. 3.

For conventional soft modeling methods (e.g. PLS, PCR), the number of latent variables or the pseudo-rank can be estimated by the ratio between two successive *PRESS* values, as a measure to test the significance of the successive dimension (Osten, 1988).*

Since *PRESS* values are not normalized for the number of objects, no direct comparisons between different sized problems are allowed. Other measures which do not depend on the dimension of the data have been reported in the literature (Cruciana, 1992), e.g. the predictive variance (*PV*) equation (7) and the Q^2 equation (8).

$$PV = PRESS/N \quad (7)$$

$$Q^2 = 1 - PRESS/SSY, \quad (8)$$

where *SSY* relates to the initial sum of squared prediction errors. In Martens & Naes (1989) the Root Mean Squared Error (RMSE) of prediction has been proposed as a measure for the total prediction error since it includes bias as well as variance.

In this paper, the RMSE equation (4) has been used as a measure for cross-validation to test the predictive ability of neural network models. Since the generalized delta rule does not produce reproducible

*It needs to be emphasized that using F-test based criteria for estimating the number of hidden units in a neural network makes little sense, since the neural models are not reproducible, due to local minima.

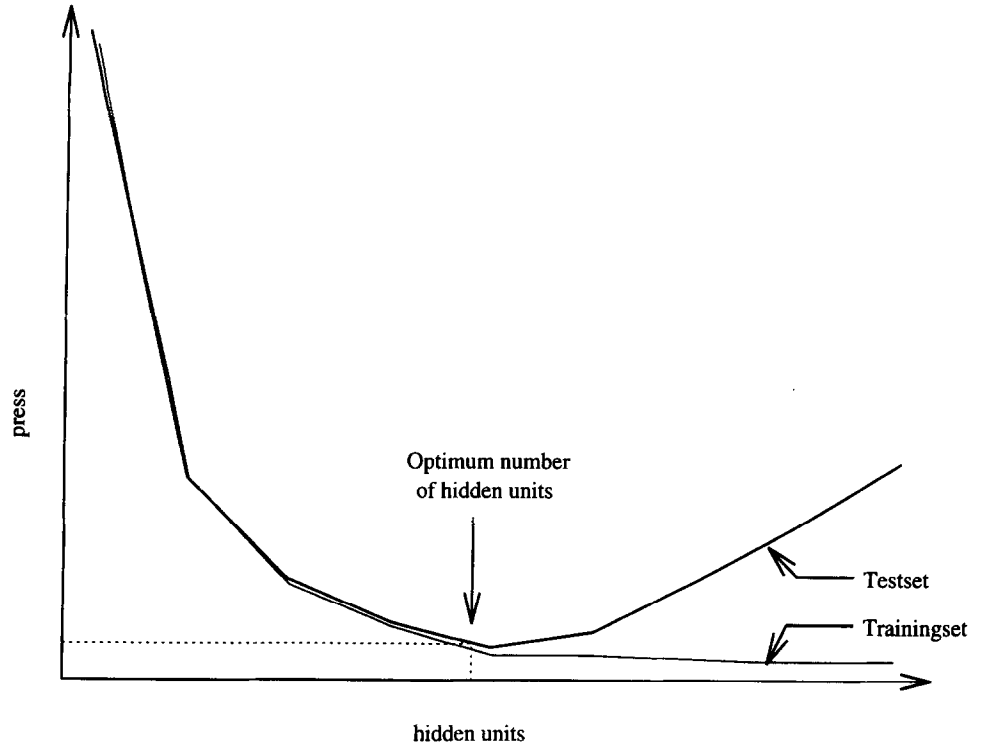

Fig. 3. Selecting the number of hidden units by finding a compromise between accuracy of fit and predictive ability.

weights, it can be imagined that faulty neural network models are obtained which are not fully trained, due to the presence of local minima in the error hyperspace, so special attention has to be paid during cross-validation that local minima are avoided.

Ending up in local minima can be avoided by reinitializing the gradient descent pathway from different locations in the error hyperplane, during the cross-validation. As can be expected, the computational effort becomes quite substantial. For example, when cross-validation is applied with Q subdivisions, on 1 up to L hidden units and R reinitializations, the neural network has to be trained $Q \cdot L \cdot R$ times. In general, this leads to long execution times accompanied by long-lasting computer overloads. For example, the cross-validation of a neural network for 1 up to 10 hidden units, on data subdivided into 10 cross-validation sets, applying 5 reinitializations ($L = 10$, $Q = 10$, $R = 5$), requires 500 independent training sessions.

3. HYDRA DRIVEN PARALLEL CROSS-VALIDATION

The computation time can be reduced, approximately by a factor Q by parallelizing the cross-validation procedure. This means that Q networks are trained simultaneously on the Q subdivisions of the data. One single parallel run yields identical results as a standard cross-validation procedure. As already mentioned in the previous section, it is highly recommendable to retrain the neural network R times, starting from different positions on the error hyperplane.

From the prediction errors equation (4) obtained from the R neural network models, the minimum, the mean and the standard deviation are calculated. The minimum prediction error is used for estimating the number of hidden units whereas the mean and standard deviation are used as statistics to give insight into the ruggedness of the error-hyperplane and the reproducibility of the various training processes.

Since the retraining cycles are completely independent for every subdivision of the data, parallel cross-validation can be performed by using the HYDRA parallel programming environment (Melssen *et al.*, 1996).

HYDRA driven cross-validation consists of the next three steps.

1. Assign the Q data subdivisions (training and test set) to Q computers and set the number of hidden units to 1.
2. Train and test the networks for L hidden units and record the prediction errors. Then, repeat the training and test procedures on each computer R times (for the calculation of the statistics (minimum, mean, standard deviation) of the R prediction errors).
3. Increase the number of hidden units by one and repeat step 2. When all networks for L hidden units are trained and tested, the procedure will finish, yielding a cross-validation table containing all prediction errors.

The cross-validation table can be used to plot the statistics (minimum, mean or standard deviation) of the prediction errors, as a function of the number of hidden units and the data subdivisions. This way, the information considering the effect of data selection (dividing the data in training and test sets) and the effect of the number of hidden units is preserved.

By means of averaging the statistics of the prediction errors for all the cross-validation subsets, a general measure for the predictive ability, based on the data used, is obtained. In the Experimental Section the procedure will be described by an example from chemical practice.

HYDRA exploits the computational power of the fastest available workstations in a local area computer network and takes care of the control of all the parallel tasks. In this paper, technical details about HYDRA are omitted, since they are extensively described in the first part of this series (Melssen *et al.*, 1996).

3.1. Configuration

In this section, the configuration of training parameters and the (parallel) cross-validation parameters are described.

3.1.1. Training parameters. The configuration of the neural network involves initializing the training parameters (McClelland & Rumelhart, 1988), e.g. the learning speed (η), the momentum factor (α), injection noise, seed values for the initial weights and termination criteria. Most of these parameters have been described extensively in McClelland & Rumelhart (1988). Hence, in this section, only the parameters affecting the probability of getting stuck in local minima will be briefly described.

The momentum factor and the injection of noise during learning, both especially help to avoid ending up in local minima. Due to the momentum factor, the training process has a more conservative character (i.e. the actual weight adaptions consist of weighted sums of current and previous weight adaptions ($w_t = w_{t-1} + \alpha \cdot \Delta w_{t-1} + \eta \cdot \Delta w_t$)), which is required to "climb" out of local minima.

Injection noise (Holmström & Koistinen, 1992; Abunawass & Owen, 1993) is introduced by adding random selections from a normal distribution to the input variables, multiplied by a magnitude factor specifying the percentage injection noise. For example, in the case of autoscaled data (corrected for mean and variance), the variances of the input variables are scaled to $\sigma = 1$. Consequently 99.7% of all input values are within 3σ limits. The addition of p% normal distributed noise on variable x_i can be established by means of the following equation

$$x_i = x_i + p \cdot \left[\frac{6 \cdot n_i}{100}\right], \tag{9}$$

where $p/100$ represents the fraction of the noise range, divided by the data range and n_i is randomly selected from a normal distribution.

The addition of p% normal distributed noise to range-scaled data can be performed in a similar way using

$$x_i = x_i + p \cdot \left[\frac{(UB - LB) \cdot n_i}{100}\right], \tag{10}$$

where the symbols LB and UB denote the lower and upper bound, respectively. Applying injection noise during training is of vital importance for the cross-validation results, since it helps the network to escape from local minima.

3.1.2. Cross-validation parameters. The configuration of the cross-validation (CV) procedure, involves specification of the number of hidden units (L), the number of subdivisions (Q) of the data set into training and test sets and the number of retraining procedures (R).

The choice of the number of data subsets depends on the number of examples contained in the data. Note that Q subsets (training and test sets) are directed to the Q selected processing computers. For example, when five computers are available, a dataset containing 100 samples can be split in five separate training and test sets, as is shown in Fig. 2. The gray boxes represent the selected test sets, whereas the remaining samples are used for training the network.

Since subset selection is a very important aspect which influences the results of cross-validation to a high extent, extra attention has to be paid to the selection of training and test samples. Erroneous cross-validation results will occur when the test or training sets are not representative for the relationship to be modeled. Especially, for test sets, which contain a limited number a examples, the chance of selecting non-representative samples is considerable. In order to limit the effect of selecting non-representative samples in the test set, a simple range check is performed on every data segmentation. The range check controls the data selection in such a way that only test samples are selected which are in the same domain as the training samples.

3.2. Implementation

The global concept of the PCV program is depicted in Fig. 4. The figure shows schematically that Q subsets are distributed on Q workstations. As the programs operate independently (asynchronous operation mode), attention will be focused on the processes on single hosts.

Since the training session on a single workstation might be too time consuming, the training process is segmented into computational blocks. Within one block the network tries to adapt the weights for a limited number of trials (epochs). Between the block, the load levels and status of the workstation are monitored in order to reschedule the process to another host, if necessary.

When the training procedure is started, the network weights will be initialized by selecting random values from a given distribution, as specified in a

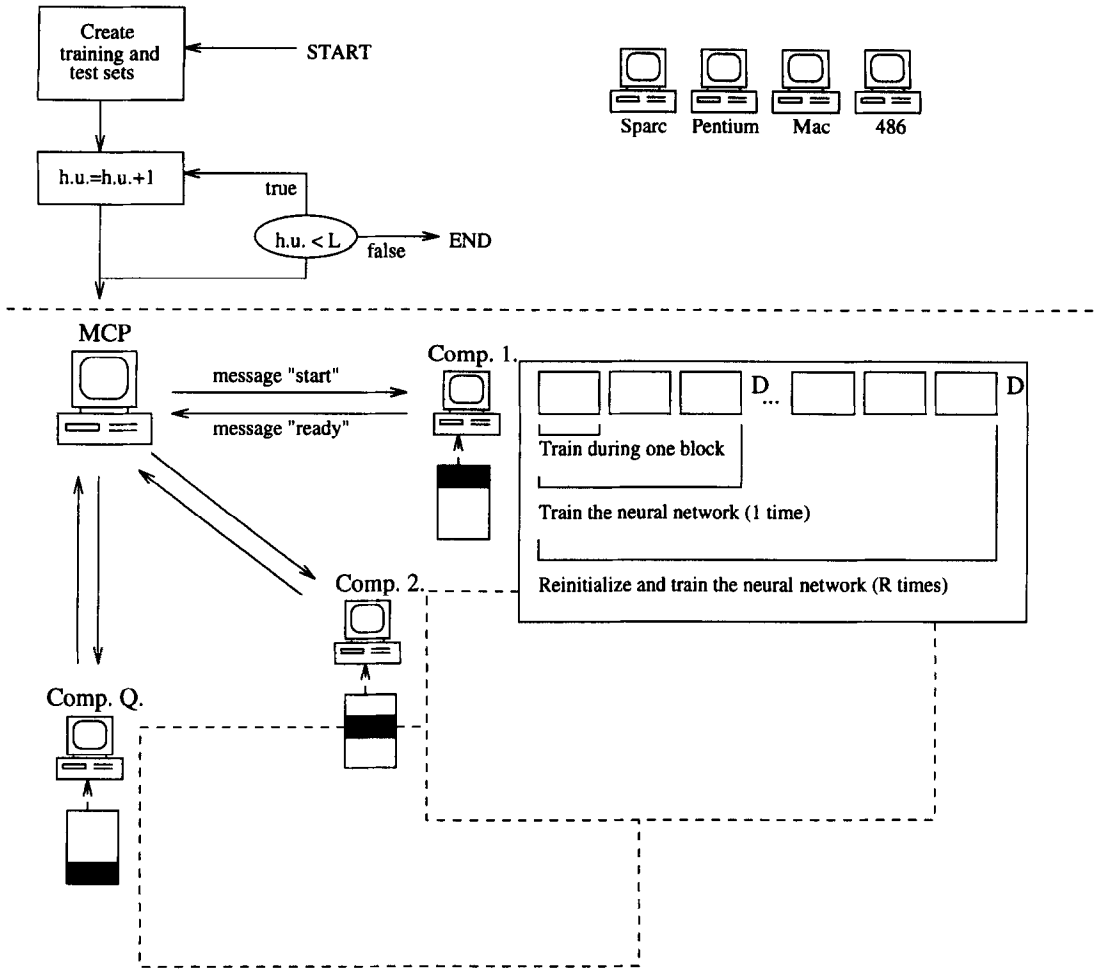

Fig. 4. A conceptual graph of Parallel Cross-validation. First the cross-validation sets are generated and directed to the fastest computers (containing arbitrary operating systems) available in the network. The MCP commands the various computers to read the configuration data, initialize the weights and start the training processes, for a given number of hidden units. Between the successive computing blocks, the computers are monitored for their performance and replaced if necessary. As soon as one training session has completed (D), the prediction errors of the training and test set are written to a cross-validation table. The reinitializations and retraining processes will be repeated $R \cdot L$ times (1 up to R reinitializations and 1 up to L hidden units).

network configuration file. Note that training parameters (e.g. η, α, epochs) are identical for every single training session. As soon as the maximum number of epochs has been exceeded, a message will be transmitted to the Master Control Program (MCP), indicating that the first block of training has finished. After sending this message, the weights are written to a file. Next, the weights from the previous blocks are read from disk and a successive block will start training the network. This process will repeat until the maximum number of blocks is encountered. The RMSE values for the training and test sets, specified by equation (4), are calculated and written to the cross-validation table.

Finally, the control will be given back to the MCP. At this time, the neural network has been trained and tested for its predictive ability, based on a single subdivision of the original dataset. As soon as the rest of the workstations have finished their calculations, the parallel cross-validation has completed for a single initialization. In the next run, the weights will be reinitialized by random sampling from a known distribution with a new seed value. The procedure described above will be repeated as is depicted in Fig. 4.

As soon as all retraining sessions have finished, the control is given back to the MCP. The number of hidden units is increased and the procedure described above will be repeated until the specified maximum number of hidden units is met.

4. EXPERIMENTAL

4.1. Modeling the physical structure and mechanical properties of yarns

The parallel cross-validation procedure has been applied to data provided by AKZO-Nobel, The

Table 1. The PCV table containing the minimum RMSE values

	Hidden units								
CV set	2	4	6	8	10	12	14	16	18
1	0.251	0.185	0.170	0.166	0.162	0.166	0.171	0.164	0.191
2	0.245	0.201	0.191	0.184	0.184	0.186	0.185	0.189	0.196
3	0.263	0.186	0.167	0.160	0.155	0.162	0.161	0.190	0.185
4	0.248	0.181	0.165	0.157	0.155	0.153	0.173	0.173	0.177
5	0.267	0.198	0.182	0.179	0.171	0.172	0.185	0.200	0.212
Mean	0.255	0.189	0.175	0.169	0.165	0.167	0.175	0.183	0.192

The last row presents the standard cross-validation results obtained by averaging the minimum RMSE values of the CV subsets.

Netherlands. The data are used for modeling the relationships between the physical structure and the mechanical properties of industrial poly(ethylene terephthalate) yarns. For a detailed description of the dataset, the reader is referred to de Weijer *et al.* (1992).

The most important characteristics of the physical structure of yarns are the crystallinity, size and orientation of the molecules. In total, 11 structure quantities have been measured. The mechanical yarn properties are described by parameters containing information about tensile strength, energy, absorbance, elongation and modulus.

4.2. Materials and methods

The cross-validation program has been developed in the C programming language and has been executed in a network consisting of a number of (SUN Sparc™) workstations. A windows based user-interface has been developed in Matlab 4.2™ which establishes an environment for high performance numeric computation and visualization.

The MLF network contained one hidden layer with tangens hyperbolicus activation functions on every hidden unit whereas the input and output layer contained linear activation functions. Both input and output data were range-scaled between -1 and 1 in order to accommodate to the tangens hyperbolicus activation function of the MLF network. The dataset consisting of approximately 300 samples has been rearranged in random order to eliminate the time effect in the data. The cross-validation sets were created by dividing the data into five test and training sets (Fig. 2). A simple outlier-detection procedure has been carried out in order to remove non-representative test objects outside the range of the training objects.

During the cross-validations, the network parameters ($\eta = 0.005$, $\alpha = 0.01$, noise = 0.01, epochs = 4000) remained constant for every training process. The number of hidden units ranged from 1 up to 18. These values have been chosen based on information obtained from prior experiments (de Weijer *et al.*, 1992; Derks *et al.*, 1995). A relatively large number of hidden units has been chosen to visualize the effect of overfitting.

4.3. Results

During the cross-validation, the neural networks were retrained five times with different seed values selected from the internal clock of the main host ($R = 5$). The RMSE values of the test set were constantly recorded for every neural model in the cross-validation procedure. The execution time for the parallel cross-validation (5 CV-runs, 1 up to 18 hidden units, 5 reinitializations, yielding $1.8 \cdot 10^6$ epochs in total) took approximately 2 h on the network, containing 10 workstations.

In Table 1 the minimum prediction errors, obtained from R reinitialized and retrained networks, are summarized for five subdivisions ($Q = 5$) and 2 up to 18 hidden units ($L = 18$). In the left-hand side of Fig. 5, the prediction errors are graphically displayed. It can easily be seen that the minimum prediction error, expressed as MIN(RMSE), mainly depends on the number of hidden units, whereas no significant dependency of the data subdivisions can be seen.

In Table 2 the standard deviations of the prediction errors are summarized. Additionally, the content of the table is displayed graphically on the right-hand side of Fig. 5. Again, the standard deviation of the prediction errors, expressed as STD(RMSE), mainly depends on the number of hidden units. The small variation between the CV-subdivisions demonstrates that the samples in the five training and test sets are representative for the relations to be modeled.

Neural networks with a small number of hidden units appear to yield more reproducible prediction errors than the "overtrained" networks. An explanation for this phenomenon is that small networks correspond to error hyperplanes which are smoother and easier to explore than the rugged error hyperplanes for large networks. Given a large number of hidden units, the situation might arise that the gradient descent search in the rigid search space ends in arbitrary local minima. Incidentally, the global minimum can also be encountered. It is evident that the optimal number of hidden units with respect to the predictive ability can be selected by tracing the minimum RMSE. The standard deviation of the prediction errors from reinitialization and retrained networks provides additional information about the reproducibility of the neural network models.

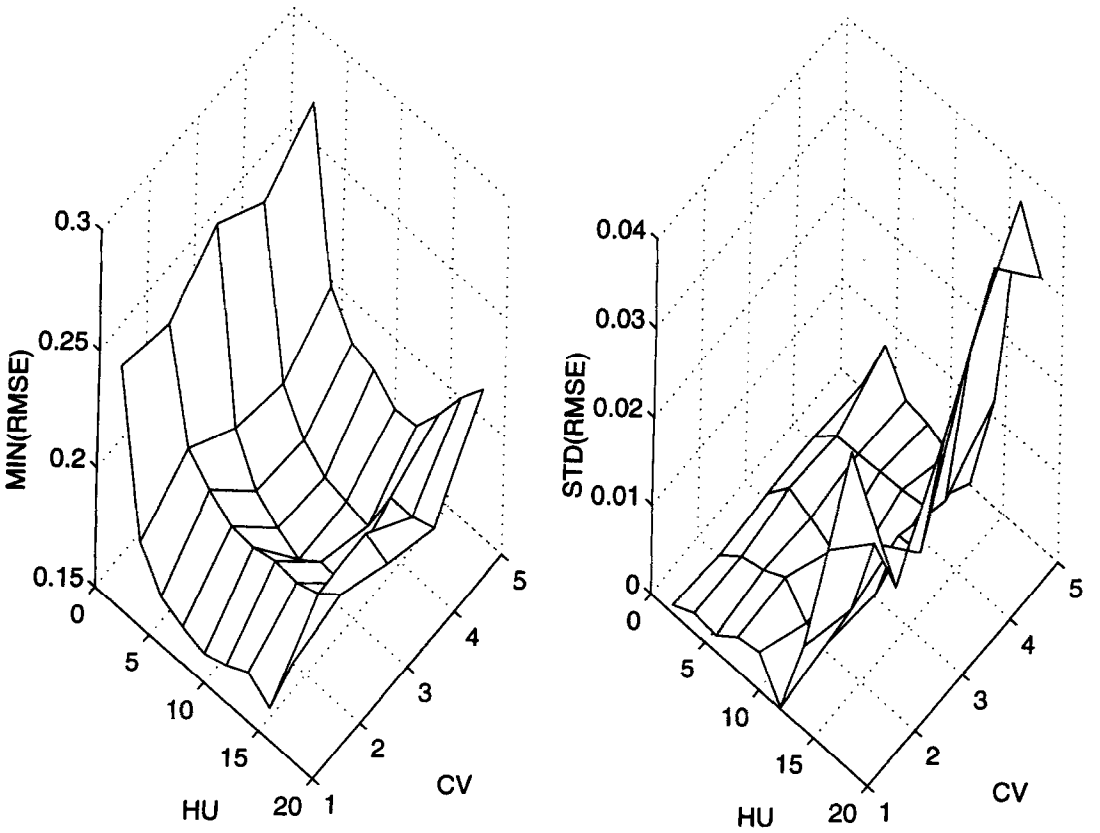

Fig. 5. Left: The minimum RMSE values obtained from the CV-subset i (x-axis) and hidden unit j (y-axis), based on R reinitializations. Right: The standard deviation of the RMSE values obtained from CV-subset i (x-axis) and hidden unit j (y-axis), for R reinitializations.

Averaging the predictions for the CV-subdivisions, Fig. 6 can be obtained. The MEAN(RMSE) as a function of the number of hidden units shows a clear minimum. The figure also reveals that the reproducibility decreases for large networks, as discussed above.

It can be concluded that 10 hidden units will suffice to create an optimal neural network model. Taking a larger number of hidden units results in a drastic deterioration of the predictive performance. In that case, the neural models become overtrained and lose their generalization ability for recognizing the test samples.

Since the overall RMSE equation (5) obscures the predictive ability per output unit, the RMSE of each of the 11 output units equation (4), are visualized in Fig. 7. It can be concluded that the output units 2, 9 and 11 show poor predictive ability whereas the units 1, 7 and 8 perform best, which is in agreement with our expectations based on prior information of experimental errors (Derks *et al.*, 1995). Note that these fairly good results could not be as well obtained by conventional cross-validation of the neural network models.

5. CONCLUSION

Conventional cross-validation involves the subdivision of data in Q training and test sets and calculating the predictive ability with the test sets with the models

Table 2. The PCV table containing the RMSE standard deviations obtained by reinitialization and retraining the networks per host

	Hidden units								
CV set	2	4	6	8	10	12	14	16	18
1	0.0009	0.0022	0.0018	0.0038	0.0043	0.0002	0.0093	0.0224	0.0356
2	0.0001	0.0022	0.0028	0.0042	0.0013	0.0013	0.0072	0.0168	0.0141
3	0.0008	0.0036	0.0025	0.0010	0.0011	0.0005	0.0092	0.0094	0.0249
4	0.0008	0.0031	0.0033	0.0035	0.0037	0.0041	0.0069	0.0266	0.0375
5	0.0005	0.0070	0.0031	0.0026	0.0013	0.0002	0.0119	0.0364	0.0301
Mean	0.0006	0.0036	0.0027	0.0030	0.0023	0.0013	0.0089	0.0223	0.0285

In the last row the values are averaged to give an indication of the reproducability of the RMSE as a function of the number of hidden units.

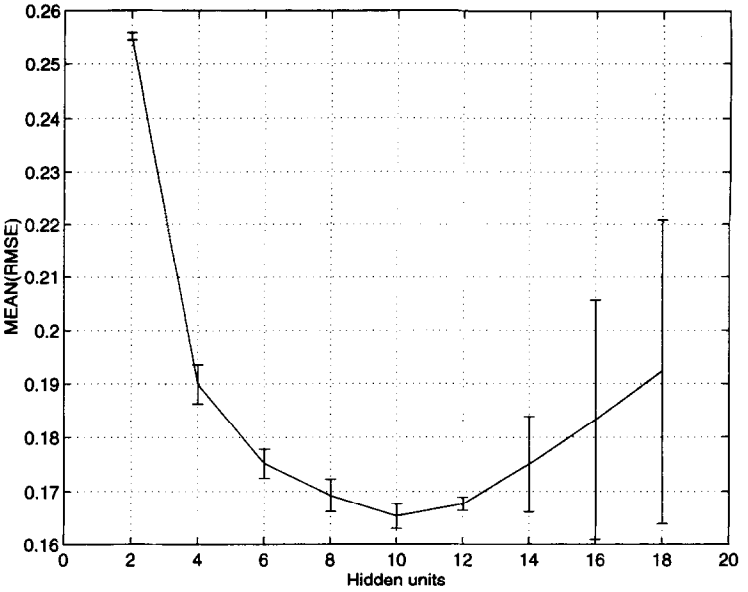

Fig. 6. The average RMSE as a function of the number of hidden units. The reproducability of RMSE is expressed by the 2σ confidence intervals.

obtained from the training sets, for an increasing number of 1 up to L hidden units. The RMSE value of the models is a general measure of the predictive ability of the modeling method used.

Problems arise when conventional cross-validation is applied on neural network models trained by the generalized delta learning rule, due to the fact that weight initializations yield unreproducible neural network models. Every new initialization can be regarded as a new start position for the gradient descent search for the global minimum. Although special learning parameters (e.g. noise injection, momentum factor) can help to avoid local minima, no guarantee of finding the global minimum can be given. The probability of finding the global minimum might be enhanced by selecting various random start positions for the gradient descent search. Consequently, there exists a bigger chance of "walking around" the local minima. Obviously, the chance of finding the global minimum directly depends on the smoothness of the error hyperplane and the number of local minima. Cross-validation by means of re-initializations and retraining the networks yields information about the smoothness of the error hyperplane and probably a better "neural" model can be established.

The HYDRA driven cross-validation of the neural network applied on the yarn-data, yields good results

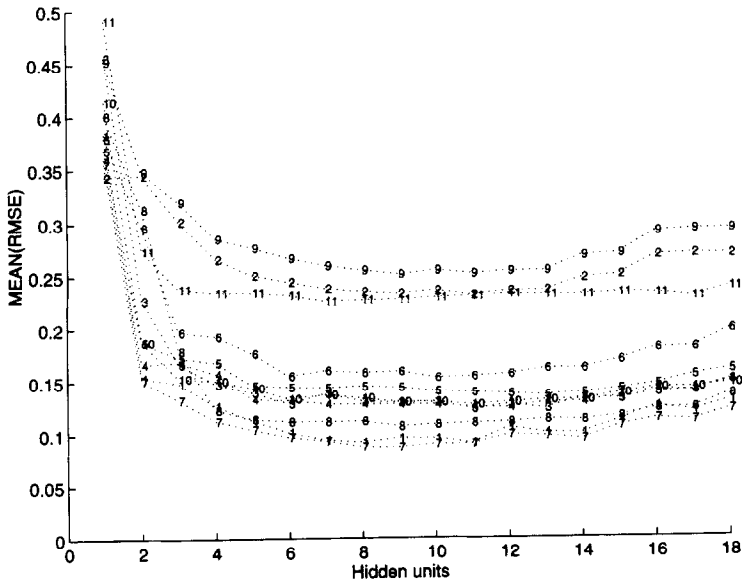

Fig. 7. The average RMSE per output unit, as a function of the number of hidden units.

which are in agreement with prior knowledge about the yarn properties. Additionally, information about the gradient descent pathways is contained, allowing some insight into the effect of network initialization and the ruggedness of the error hyperplane.

Finally, we conclude that the rather easy implementation of the cross-validation program in HYDRA, yields a considerable gain in execution time (2 h vs 10 h for a conventional cross-validation on a single computer). Moreover, a robust and reliable execution in a computer network can be guaranteed.

Acknowledgements—The authors wish to acknowledge Geert Rolf for the development of HYDRA and Ronald Pijpers for the implementation of the parallel cross-validation program. E. P. P. A. Derks is supported by the Dutch Foundation for Chemical Research (SON) on behalf of the Dutch Organization for Scientific Research (NWO) and partially by the EC project RENEGADE (ERBCHRXCT930419)).

REFERENCES

Abunawass A. M. & Owen C. B. (1993) *Sci. Artif. Neural Networks* 1966.

Allen D. M. (1974) *Technometrics* **16,** 125.

Amato S., Apolloni B., Caporali G., Madesani U. & Zanaboni A. (1991) *Neurocomputing* **3,** 207.

Bos A. (1993) PhD thesis, University of Twente, The Netherlands.

Cruciana G., Baroni M., Clementi S., Costantino G., Riganelli D. & Skagerberg B. (1992) *J. Chemometrics* **6,** 335.

Derks E. P. P. A., Sànchez Pastor M. S. & Buydens L. M. C. (1995) *Chemometrics Intell. Lab. Syst.* **28,** 49.

Geisser S. (1975) *J. Am. Statis. Assoc.* **70,** 320.

Holmström L. & Koistinen P. (1992) *IEEE Trans. Neural Networks* **3.**

Martens H. & Naes T. (1989) *Multivariate Calibration.* John Wiley & Sons, New York.

McClelland J. L. & Rumelhart D. E. (1988) *Parallel Distributed Processing*, Vol. 2. MIT Press, London.

Melssen W. J., Derks E. P. P. A., Beckers M. L. M. & Buydens, L. M. C. (1996) *Computers Chem.* **20,** 431.

Osten D. W. (1988) *J. Chemometrics* **2,** 39.

Schaffer J. D., Whitley D. & Eshelman L. J. (1992) *Proc. COGANN-92,* 1.

Stone M. (1974) *J. Roy. Statist. Soc.* **36,** 111.

Weijer de A. P., Buydens L., Kateman G. & Heuvel H. M. (1992) *Chemometrics Intell. Lab. Syst.* **16,** 77.

White D. A. & Sofge D. A. (1992) *Handbook of Intelligent Control.* Multiscience Press.

Wold S. (1978) *Technometrics* **20,** 397.