15th Triennial World Congress, Barcelona, Spain

# MLP BASED NONLINEAR DYNAMIC SYSTEM MODELING THROUGH IMPROVED TRAINING ALGORITHM

**Kang Li, Steve Thompson**

*School of Mechanical & Manufacturing Engineering*
*Queen's University Belfast*
*Ashby Building, Stranmillis Rd., Belfast BT9 5AH, UK*

Abstract: Multi-Layer Perceptron network modeling for nonlinear dynamic systems is studied. The situations that only a relatively small number of training data is available and that the training data does not cover all system dynamics are mainly concerned. An improved method is proposed by training with two sets of data, which is shown to give better generalization performance in the above-mentioned circumstances. *Copyright @2002 IFAC*

Keyword: neural networks, training, generalization, nonlinear systems, dynamic modeling.

## 1. INTRODUCTION

The main objective in nonlinear dynamic system modeling using artificial neural networks is to obtain a network model with good generalization performance (Hertz, and Krogh, 1991; Narenda and Parthasarathy, 1990; Qin et al, 1992).Generalization refers to the ability to predict unseen data. It is understood that in neural network training, if the training stops when the training error reaches the minimum, the generalization performance can be poor (Prechelt, 1998). Various solutions to this problem are available, including cross-validated early stopping technique and regularization (Poggio and Girosi, 1990; Prechelt, 1998). In cross-validated early stopping technique, two data sets are involved, one is used for training, and the other is used for validation. The cross-validated early stopping technique is implemented as follows: the behavior of the trained network is evaluated by the validation data and the training will stop at the point that the error on the validation data set starts to rise instead of decreasing. Regularization technique is in fact a constrained training process, where a penalty term ( $DE$ ) is added to the cost function $E$ which restricts the variance of the model. The modified cost function $\tilde{E}$ becomes:

$$\tilde{E} = E + lDE \qquad (1)$$

where $l$ is a scalar that determines the influence of $DE$ .

It has been proved that early stopping is equivalent to zero-order regularization (Poggio and Girosi, 1990); there is an analogy between the number of iterations and the regularization parameter $l$ in (1). The difficulty with regularization is in choosing the regularization parameter. In order to determine a possible regularization parameter, additional computation and inference based on other criteria are required. In general the early stopping technique is widely used because it is simple to understand and implement and has been reported to be superior to regularization methods in many cases.

In this paper, an improved method of training neural networks is proposed based on the previous result

made by the authors (Li and Thompson 2000). To achieve better generalization performance in case that the available training data set is of relatively small size and does not cover whole system dynamics, two sets of data are used. The proposed method adopts the core concept in the cross-validated early stopping technique. The main difference between the new method and the early stopping technique is that in the new method both data sets are explicitly involved in training to update the weights. In order to do so, two training loops, namely an inner loop and an outer loop, are introduced into the new method. The inner loop performs cross-validated early stopping training using a standard gradient-descent type algorithm. The outer loop exchanges the training data set and the validation data set and initiates inner loop training. The whole training process comprises a number of outer loop training cycles, and the results of previous outer loop training cycles are used to initialize the following outer loop cycle. Since in each outer loop cycle only one data set is directly used in training to update the weights, the final training error of the new method will be distributed within a close neighborhood of the minimal error. The motivations behind this method are summarized as follows:

1). Although all available training algorithms are designed to minimize the cost function of one set of data, in practice, two or more disjoint data sets will have been collected.

2). The training data set can be of relatively small size and not able to cover all system dynamics. The reasons are that:

- In industrial process, most useful training data are acquired by filed tests and experiments. As a result the collected data is unlikely to cover whole system dynamics;
- In many industrial processes, operating conditions vary from time to time;
- Industrial processes are always corrupted with white/colored noise and unexpected disturbances.

In the above situations an early stopping technique is quite useful. In our method, training is performed using two data sets and the training process is able to automatically ensure that the final training error is distributed within a close neighborhood of the minimal error over the whole training data.

3). Although the cross-validated early stopping technique is useful when the training data is insufficient to reflect the whole system dynamics. It suffers several problems,

- The time to stop the training process largely depends on the validation data set.
- When there is only one data set is available for training, there is no well-recognized guideline on how to split the supervised data into training data and validation data sets.
- Training may get stuck in local minima.
- The training may stop too early.

Since the validation data is not explicitly used in training, there is no guarantee that an appropriate tradeoff is made between training data and validation data. In the new method, training is performed over the two data sets separately, and the results acquired in previous trainings are used as initial conditions for subsequent training.

4). It is understood that there are multi-local minimums in the error space and that the training process can stick in a local minimum before reaching the global minimum. To avoid this situation, a conventional means is to train the network repeatedly, which is time consuming and there is no guarantee that the final result will improve significantly. In the proposed method, the solution is to add perturbation to the initial values of the weights in each outer loop training cycle. In order to avoid an excessive computational burden, the stop criteria for the inner and outer loops are designed so that the overall cost function will decrease along some predefined trajectory.

The paper is organized as follows. In section 2, key features in this new method are introduced, the training algorithm is proposed. Further discussions are made on how to choose training parameters. In section 3, simulations are made using the improved algorithm. Section 4 is the conclusion.

## 2. THE TRAINING ALGORITHM

### *2.1 The training algorithm*

In this paper, the multi-layer perceptron (MLP) neural network is used. Two typical activation functions $f$ are the sigmoid function and the hyperbolic tangent function. Most MLP training algorithms are recursive learning algorithms based on Newton-type gradient-descent techniques. Recent advances in training used second-order optimization techniques. Typically, they involve the calculation of at least an approximate Hessian Matrix associated with the function to be optimized. Some popular update algorithms are LM (Levenberg-Marquart) method, Broyden-Fletcher-Golfarb-Shanno (BFGS) algorithm, the scaled conjugate gradient algorithm, etc. (Hertz, and Krogh, 1991; Hagen and Menhaj, 1994). In this paper, the LM method will be used.

Suppose we have a network model with $p$ input nodes and $q$ output nodes. Two data sets are collected for training, i.e. $Z_1$ and $Z_2$ which contain $N_1$ and $N_2$ data points respectively,

$$\begin{cases} Z_1 = \{z_{1i}, i = 1,2,...,N_1\} \\ z_{1i} = \{[u_{11}(i)u_{12}(i)...u_{1p}(i)]^T ; [t_{11}(i)t_{12}(i)...t_{1q}(i)]^T\} \end{cases} \quad (2)$$

$$\begin{cases} Z_2 = \{z_{2i}, i = 1,2,...,N_2\} \\ z_{2i} = \{[u_{21}(i)u_{22}(i)...u_{2p}(i)]^T ; [t_{21}(i)t_{22}(i)...t_{2q}(i)]^T\} \end{cases} \quad (3)$$

where $z_{1i}, i = 1,2,...,N_1$ and $z_{2i}, i = 1,2,...,N_2$ are data points in the two data sets; $u_{1i}(j), i = 1,2,...,p, j = 1,2,...,N_1$ and $u_{2i}(j), i = 1,2,...,p, j = 1,2,...,N_2$ are input values in the two data sets; $t_{1i}(j), i = 1,2,...,q, j = 1,2,...,N_1$ and $t_{2i}(j), i = 1,2,...,q, j = 1,2,...,N_2$ are the output targets.

Then the cost functions for these two data sets are defined as:

$$E_1(Z_1;w) = \sum_{j=1}^{N_1}\sum_{i=1}^{q}(y_{1i}(j) - t_{1i}(j))^2 = \|e_1\|^2 \quad (4)$$

$$E_2(Z_2;w) = \sum_{j=1}^{N_2}\sum_{i=1}^{q}(y_{2i}(j) - t_{2i}(j))^2 = \|e_2\|^2 \quad (5)$$

where $y_{1i}(j), i = 1,2,...,q, j = 1,2,...,N_1$ and $y_{2i}(j), i = 1,2,...,q, j = 1,2,...,N_2$ are the outputs of the network model given inputs from the two training data sets, $q$ is the number of output nodes, $w$ is the adjustable weight vector, $e_1$ and $e_2$ are error vectors.

The cost function of these two data sets is defined as

$$E_0(Z_0;w) = E_1(Z_1;w) + E_2(Z_2;w) = \|e_0\|^2 \quad (6)$$

where $Z_0 = Z_1 + Z_2$, $e_0$ is the error vector. A recursive training algorithm to update the weights with respect to the cost function defined in form of (4), (5) or (6) may take the following form,

$$\begin{cases} w^{(i+1)} = w^{(i)} - m^{(i)}H^{(i)}E_j'(w^{(i)}), \quad j = 0,1,2 \\ w^{(0)} = w_0 \end{cases} \quad (7)$$

where $m$ is the step size which is determined by some search along the indicated line. $H$ is some positive definite matrix, $E_j', j = 0,1,2$ is the first derivative of the cost function of either (4), (5) or (6) with respect to the weights. The initial value for $\omega$ could be some prior guess. In practical, $w_0$ is generated randomly.

In particular, for the LM approach, (7) will take the form of

$$w^{(i+1)} = w^{(i)} - m^{(i)}(R_j^T P_j^T + lI)^{-1} P_j^T e_j(w^{(i)}) \quad j = 0,1,2 \quad (8)$$

where $P_j = (\frac{\partial e_j}{\partial w})$, $I$ is an identity matrix, $e_j, j = 0,1,2$ is defined in (4), (5) and (6).

In order to help better understanding of the algorithm to be followed, the key features in this method are introduced in the following as a start:

- The training algorithm comprises two training loops, namely inner loop and outer loop. The inner loop performs network training over one of the two data sets using LM method. For example, if data set $Z_1$ is used in training, then network weights are updated in the inner loop as follows:

$$w^{(i+1)} = w^{(i)} - m^{(i)}(R_1^T P_1 + lI)^{-1} P_1^T e_1(w^{(i)}) \quad (9)$$

- The outer loop performs training repetitively over the two data sets separately. For example, if data set $Z_1$ is used for training in the $i^{th}$ outer loop cycle, then $Z_2$ will be used for training in the $(i+1)^{th}$ outer loop cycle.
- Each outer loop cycle performs a complete cross-validated early stopping type training over one of the two data sets, and training within an outer loop cycle is implemented in the inner loop.
- The initial values of weights for an outer loop training cycle comprises two parts, one part uses the previous training results, and the other part is a disturbance, which is designed to help the training avoid getting stuck in local minima. Suppose we want to initialize the $k^{th}$ outer loop training cycle, the initial values for weights $w_k^{(0)}$ are calculated as:

$$w_k^{(0)} = [a(w_{k-1}^* + w_{best}^*) + b\aleph_k]/g \quad (10)$$

where $w_{best}^*$ is the best solution in previous $(k-1)$ outer loop training cycles, $w_{k-1}^*$ is the solution of the $(k-1)^{th}$ outer loop training cycle, $\aleph_k$ is a uniformly distributed random matrix with the same dimension of $w$, and the upper and lower bound of values of entries in $\aleph_k$ are the maximal and minimal values of entries in matrix $(w_{k-1}^* + w_{best}^*)$. For the first outer loop cycle, values for the entries in these matrices are generated randomly. $a, b \in [0,1], g > 1$ are training parameters. $a$ and $b$ represent the percentages of previous training results and the disturbance item in the initial value of weights.

- In the new method the training error $E_0(Z_0;w)$ decreases as training proceeds. An ideal situation is that this training error can decrease along a pre-defined trajectory as outer loop training proceeds. In order to do so, the target for an outer loop training cycle, e.g. $k^{th}$ outer loop cycle, is set to be:

$$E_0(Z_0;w_k^{(i)}) < hE_0(Z_0;w_{k-1}^*) \quad (11)$$

where $w_k^{(i)}$ are the weights derived from the current $i^{th}$ iteration within the current $k^{th}$ outer loop training cycle, $w_{k-1}^*$ is the solution of the $(i-1)^{th}$ outer loop training cycle, $h \in (0,1)$ is a training parameter and has been chosen to be 0.1 in practice. However, the inner loop training within an outer loop cycle can not continue infinitely if (11) is not achieved. In this case, the cross-validated early stop criterion for the inner loop training within an outer loop cycle is applied:

$$E_2(Z_2;w_k^{(i)}) > E_2(Z_2;w_k^{(i-1)}) \quad (12)$$

where $w_k^{(i)}, w_k^{(i-1)}$ are the weights derived from the current $i^{th}$ iteration and the last $(i-1)^{th}$ iteration within the current $k^{th}$ outer loop cycle, $Z_2$ is the validation set that is not directly involved in training in current outer loop training cycle to update the weights. In conclusion, the inner loop training within an outer loop cycle will stop if either (11) or (12) is satisfied.

- The whole training process comprises a number of outer loop cycles, the criteria to stop the whole training process are defined as:

$$k > N_{out}^{max} \quad (13)$$

and

$$E_0(Z_0;w_k^*) \le E_{min} \quad (14)$$

where $k$ is the number of outer loop training cycles; $N_{out}^{max}$ is the maximal outer loop cycles which can be a number between 10 to 20; $w_k^*$ is the result of the current outer loop training cycle, $E_{min}$ is the user defined desirable training error. In conclusion, the whole process will stop if (13) or (14) is satisfied.

*Algorithm: MLP training with two sets of data*

*Step 1. Initialization phase.*

1) Select two training data sets.
2) Let $Z_1$ always denote the training data set in an outer loop cycle, $Z_2$ denotes the validation data set used in (12) to stop the training process in an outer loop cycle.
3) Select a gradient-descent type training algorithm for the inner loop training, in this paper the LM method is used.
4) Determine the parameters $a, b, g$ in (10). They are always chosen to be $a = 0.67$, $b = 0.33$, $g = 4$.
5) Determine maximal outer loop training cycles $N_{out}^{max}$, it has been chosen to be 10 in all our examples. Set the desired training error $E_{min}$.
6) $w_{k-1}^*$ for $k=1$ is generated randomly, let $w_{best}^* = w_{k-1}^*$ for $k=1$, and calculate $E(Z_0;w_{best}^*)$.
7) Calculate the initial values of weights as (10) for the first outer loop training cycle, and set the number of outer loop cycle $k=1$.

*Step 2. An outer loop cycle training phase.*

Training is implemented within the inner loop:

a) The network weights are updated using (9).
b) After each training epoch in the inner loop training, check whether or not the criterions (11) or (12) are met. If either (11) or (12) is satisfied, stop inner loop training and the current outer loop cycle is terminated, go to step 3. Otherwise, go to a) and inner loop training continues.

*Step 3. Update phase.*

1) If (11) is satisfied, or (12) is satisfied together with $E(Z_0;w_k^*) < E(Z_0;w_{best}^*)$, then update intermediate variables $w_{k-1}^*$, $E(Z_0;w_{k-1}^*)$, $w_{best}^*$, $E(Z_0;w_{best}^*)$ as follows:

- $w_{k-1}^* = w_k^*$, and $w_{best}^* = w_k^*$;
- $E(Z_0;w_{k-1}^*) = E(Z_0;w_k^*)$,

and

- $E(Z_0;w_{best}^*) = E(Z_0;w_k^*)$.

2). If (12) is satisfied but $E(Z_0;w^{(best)}) > E(Z_0;w_{k+1}^*)$, then $w_{k-1}^* = w_k^*$, and $E(Z_0;w_{k-1}^*) = E(Z_0;w_k^*)$.

*Step 4. Check phase.* If either (13) or (14) is satisfied, the whole training process stops and $w_{best}^*$ is the solution. Otherwise, go to step 5.

*Step 5. Preparation phase*. Exchange the training data set and the validation data set, i.e. if one data set is used as the training data set in last outer loop training cycle, then it will be used as the validation data set for the following cycle. Compute the initial values of weights for the following outer loop cycle according to (10). $k = k + 1$, and go to *Step 2*.

*2.2 Selection of training parameters*

In this training algorithm, key training parameters have to be determined.

- $a$, $b$, $g$ in (10). $a$ and $b$ represent the percentages of previous training results and the disturbance item in the initial values, therefore $a, b \in [0,1]$ and $a + b = 1$. If $a = 1$ and $b = 0$, then no disturbance is added, and training in an outer loop cycle is solely performed on the base of previous outer cycle results, the computation

complexity is reduced, but the training can stick at local minima. However, if $a = 0$ and $b = 1$, previous outer loop training information is not used and the training process approximates to a number of independent cross-validated early stopping training processes and the computation complexity can increase significantly. Therefore, $\alpha$ and $b$ are chosen so that the disturbance is large enough to help training escape a local minina, but does not fully mask the values of previous training results. It has been found that $a = 0.67$, $b = 0.33$ is the best choice, Coincidently it uses the golden segmentation principle to produce the initial value. $g$ is used to ensure that the searching for weights always starts from small values. In practice, $g = 4$.

- The maximal outer loop cycles, $N_{out}^{max}$ determines how many cross-validated early stopping type training are performed to get a global optimal solution. The larger $N_{out}^{max}$ is, the more complex the computation becomes. According to (11), if $h = 0.1$, after 10 outer loop cycle, the training error is expected to reduce to $10^{-10}$ times the initial training error. Although in practice, this can be hardly achieved, it has been found that $N_{out}^{max} = 10$ is a good choice, and further increase of $N_{out}^{max}$ may not improve the training solution significantly, but will increase the computational burden.

## 3. SIMULATION EXAMPLES

The proposed method has been tested on various artificial nonlinear dynamic systems, where the training data set is of relatively small size and does not cover all system dynamics. All simulations are performed using MATLAB, MathWorks, Inc., on a Pentium III PC with machine frequency of 700MHz.

For each of the following examples, three data sets are used. The first two data sets can be used for training, and their sizes are relatively small. The third independent data set is used to test the generalization performance, and is not involved in training explicitly or implicitly, and generally 4 to 8 times larger than the other two data sets. Consequently, the following examples test the generalization performance of produced networks. This is typical in modeling/simulating nonlinear dynamic systems, where once the model is built up, it will be tested or used for a rather long period of time under varying conditions. For the reason of comparison, in each modeling/simulation example, three training methods are used:

- *Method 1*: The new method proposed in this paper. In all examples, the maximal outer loop cycle number is chosen to be 10.
- *Method 2:* The cross-validated early stopping technique, i.e. the training will stop once the error on the validation data starts to rise instead of decrease;
- *Method 3:* The first two data sets are used for training, i.e. the two data sets are combined as one training data set and the training process will stop only when training error cannot make further decrease.

*Example 1*

A two-input/one-output MLP is used to model/simulate the dynamics of the following nonlinear system:

$$y(t) = \frac{u(t-1) - 0.9y(t-1)}{1 + y(t-1)^2} \qquad (15)$$

where $y(t)$ and $y(t-1)$ are system outputs at time instant $t$ and $t$-$1$ respectively, $u(t-1)$ is the system input at time instant $t$-$1$. The system input $u(t)$ is uniformly distributed within the range of [0, 1].

The MLP model is a (2,10,1) network, which has 10 hidden nodes; 2 inputs: $y(t-1)$ and $u(t-1)$; one output: $y(t)$. Each of the two training data sets has 50 data points respectively. The generalization data set has 400 data points. These data sets are all generated by simulating (15) with test signal $u(t)$ of range [0, 1].

Table 1 Training results for example 1

| | Generalization Error | Iterations | Flops |
|---|---|---|---|
| Method 1 | $5.89\times10^{-5}$ | 246 | $9.36\times10^{7}$ |
| Method 2 | $1.48\times10^{-1}$ | 6 | $2.27\times10^{6}$ |
| Method 3 | $2.28\times10^{-3}$ | 200 | $1.13\times10^{8}$ |

Table 1 lists the training results by the three training methods. In table 1, 'Iterations' is the total number of iterations using (9) to update the network weights; 'Flops' is used to account the total number of float point operations in a computation process (In Matlab additions and subtractions are one flop if real and two if complex. Multiplications and divisions count one flop each if the result is real and six flops if it is not). Obviously, Method 1 is able to produce far better generalization performance, and using early stopping technique only once will not produce a network with better performance. The result of method 3 shows that when training data sets are small, even use all training data sets, the network still can still produce bias.

*Example 2*

A four-input/one-output MLP is used to model/simulate the dynamics of the following nonlinear system:

$$y(t) = 0.2\frac{u(t-1)u(t-2)}{1+y(t-1)^2+y(t-2)^2} + e(t) \quad (16)$$

where $y(t)$, $y(t-1)$ and $y(t-2)$ are system outputs in time *t, t-1, t-2* respectively; $u(t-1)$, $u(t-2)$ are system inputs in time *t-1, t-2*; $e(t)$ is white noise of range of [0, 0.01].

The MLP model is (4,10,1), which has 10 hidden nodes, 4 inputs: $u(t-1)$, $u(t-2)$, $y(t-1)$, $y(t-2)$, and one output: $y(t)$. Each of the two training data sets has 50 data points respectively. These two training data sets are generated by simulating (16) with input signal u(t) of range [-0.5, 0.5]. The generalization data set has 400 data points that are generated by simulating (16) with input signal $u(t)$ of range [-0.7, 0.7]. Apparently, the two training data sets do not cover the whole range of values in the generalization data. Data in these three sets are corrupted with white noise.

The training process using method 1 is depicted in Fig.1 and results using the three training methods are summarized in table 2. According to Fig. 1, both training error and generalization error can approach different local minima in different outer loop cycles.

Table 2 Training result for example 2

| | Generalization Error | Iterations | Flops |
|---|---|---|---|
| Method 1 | $6.38\times10^{-2}$ | 59 | $4.70\times10^{7}$ |
| Method 2 | $4.12\times10^{-1}$ | 2 | $2.21\times10^{6}$ |
| Method 3 | 1.40 | 30 | $3.45\times10^{7}$ |

## 4. CONCLUSION

In this paper, MLP based nonlinear dynamic system modeling has been studied, an improved method has been proposed for MLP training with two sets of data, which is able to give better generalization performance in the situation that the available training data set is of small size and does not cover all system dynamics. Simulation examples have shown the merit of this method.


## ACKNOWLEDGEMENTS

Acknowledgement is made to the British Coal Utilisation Research Association and the UK Department of Trade and Industry for a grant in aid of this research but the views expressed are those of the authors, and not necessarily those of BCURA or the Department of Trade and Industry.


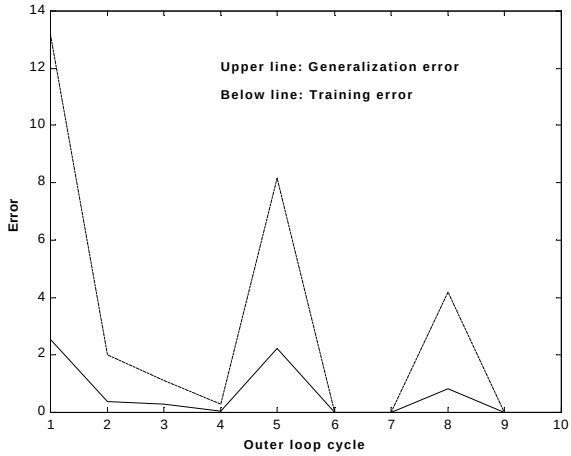


Fig. 1 Training and generalization error with outer loop cycles

## REFERENCES


Chng, E.S., S. Chen, and B. Mulgrew (1996). Gradient radial basis function networks for nonlinear and stationary time series prediction. *IEEE Trans. Neural Networks*, **17**, 190-194.

Hagen, M. T. and M.B. Menhaj (1994). Training feedforward networks with the marquardt algorithm. *IEEE Trans. on Neural Networks*, **5**, 989-993.

Hertz, and A. Krogh (1991). *Introduction to the theory of neural computation*, Reading, MA: Addison-Wesley.

Li, K., S. Thompson (2000). Developing NOx Emission Model for a Coal-fired Power Generation Plant Using Artificial Neural Networks. *Proceedings of UKACC International Conference on CONTROL 2000*, Cambridge.

Narenda, K.S., K. Parthasarathy (1990). Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Networks*,**1**, 4-27.

Qin, S., H. Su, and T.J. McAvoy (1992). Comparison of four neural-net learning methods for dynamic system identification. *IEEE Trans. Neural Networks*, **3**, 122-130

Poggio, T.and F. Girosi (1990). Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, **247**, 978-982.

Prechelt, L. (1998). Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, **11**, 761-767.