ELSEVIER

Neurocomputing 14 (1997) 85–99

NEUROCOMPUTING

Accommodation to outliers in identification of non linear SISO systems with neural networks

Gérard Bloch *, Philippe Thomas, Didier Theilliol

Centre de Recherche en Automatique de Nancy, CNRS URA D821 ESSTIN, Rue Jean Lamour, 54500 Vandoeuvre, France

Received 2 January 1995; accepted 2 November 1995

Abstract

The problem of non-linear Single Input Single Output system identification in the presence of large errors in data is considered. Combining the capabilities of neural networks to solve non-linear problems by learning and a robust recursive prediction error learning rule based on the modelling of the errors, a new algorithm is drawn up. Its potentialities are illustrated through simulation studies.

Keywords: Identification; Neural networks; Robustness; Non-linear system; Outliers

1. Introduction

Artificial neural networks have been the focus of a great deal of attention during the last decade, due to their capabilities to solve non-linear problems by learning. Such networks provide a parallel structure with very simple processing elements. Although a broad range of neural networks (NN) architectures and learning rules are available [9,12,14,23], the backpropagation algorithm for multilayer feedforward networks [21] is the most popular approach for engineering applications. Backpropagation or derived algorithms have been successfully applied for classification and pattern recognition [20,24], fault detection [10,13], non-linear control [17] and process modelling and identification [2,3,18].

On the other hand, an extensive literature on system identification can be found. Among general textbooks on the subject, Box and Jenkins [4], Söderström and Stoica

* Corresponding author. Email: bloch@cran.esstin.u-nancy.fr

[22] and Ljung [15] can be mentioned. Particularly, much effort has been devoted to tackle the presence of outliers in experimental input and output data used for identification. Large errors or outliers in data can be for instance caused by offset of sensors, failure of transducers, analog to digital conversion errors or even by malfunctioning of transmission devices. The related works are mainly based on modelling such outliers to produce so called robust identification algorithms. But these works are limited to linear systems.

In this paper, the main feature of neural networks, the ability to identify non-linear systems and a robust recursive prediction error algorithm, based on the modelling of errors due to Huber [11], are combined. This modelling has been used by Puthenpura and Sinha [19] for a robust linear recursive least squares type identification algorithm. The convergence of this kind of algorithms particularly for non-linear systems is very slow. So, a robust feature is introduced in a recursive Gauss-Newton type of algorithm, first employed by Chen et al. [5] for neural networks. The goal is to accommodate to outliers in order to eliminate their effects in the identification of non linear SISO systems by an appropriate choice of the criterion to be minimised.

2. Multilayer feedforward neural networks for identification

In this part, the structure of the multilayer feedforward neural network, used for identification of dynamic single input single output (SISO) systems, is presented. The network shown in Fig. 1 is composed of interconnected processing units in three successive layers.

The first or input layer is composed of 'transparent' units which do not perform any computation but simply distribute theirs inputs to all neurons in the next layer called hidden layer ($x_k^0 = x_k(t), \forall k$). For identification purposes, several authors such as Cybenko [7] or Funahashi [8] have established that multilayer feedforward neural networks with a single hidden layer are able to approximate continuous functions. The last layer is the output layer, composed of a single neuron and its output gives the estimated output of the SISO system.

Neurons in the hidden and output layers are identical and can be represented as illustrated in Fig. 2.

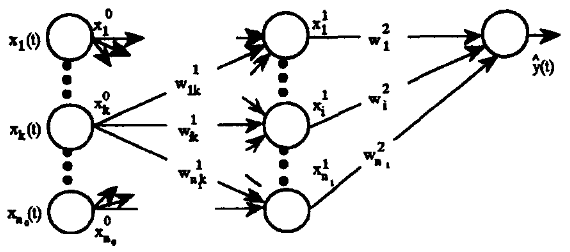

Fig. 1. A three layers feedforward neural architecture.

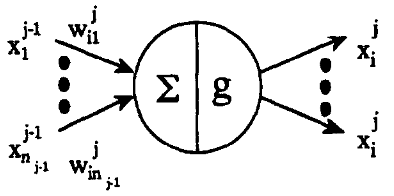

Fig. 2. Neuron i in layer j.

The ith neutron in the layer j receives n_{j-1} inputs $\{x_1^{j-1},\cdots,x_{n_{j-1}}^{j-1}\}$ from layer $j-1$ with associated weights $\{w_{i1}^{j},\cdots,w_{in_{j-1}}^{j}\}$. This neuron first computes the weighted sum of the n_{j-1} inputs:

$$z_i^j=\sum_{k=1}^{n_{j-1}} w_{ik}^j x_k^{j-1}+b_i^j \tag{1}$$

where b_i^j is a bias or threshold term. The output of the neuron is a non-linear function of the sum in (1):

$$x_i^j=g(z_i^j) \tag{2}$$

where g denotes the activation function, chosen as often to be a sigmoidal function, here:

$$g(x)=1/(1+e^{-x}) \tag{3}$$

with $\lim\limits_{x\to-\infty} g(x)=0$ and $\lim\limits_{x\to+\infty} g(x)=1$.

So, for the structure and the notations of Fig. 1, a network with a single hidden layer can be defined by the following model:

$$\hat{y}(t)=g(z) \tag{4a}$$

$$z=\sum_{k=1}^{n_1} w_k^2 x_k^1+b^2 \tag{4b}$$

$$x_i^1=g(z_i^1) \tag{4c}$$

$$z_i^1=\sum_{k=1}^{n_0} w_{ik}^1 x_k(t)+b_i^1 \tag{4d}$$

$x(t)=[x_1(t)\ldots x_{n_0}(t)]^{\mathrm{T}}$ is the $n_0.1$ input vector.

In the following, n_d, n_u and n_y refer respectively the input time delay, the numbers of lagged system inputs and outputs to be applied to input layer of the network. In the training phase of the neural network, i.e. the identification step, input $u(t-n_d)$, $u(t-n_d-1)\ldots$ and output $y(t-1)$, $y(t-2)\ldots$ values of the process are successively applied on the input layer of the network in order to produce estimated value of the system output:

$$\hat{y}(t,\Theta)=NN\big(y(t-1),\ldots,y(t-n_y),\ u(t-n_d),\ldots,u(t-n_d-n_u+1)\big) \tag{5}$$

where $\Theta = [\theta_1 \dots \theta_{n_\theta}]^T$ comprises all the unknown weights and biases of the network. The dimension n_θ of the parameter vector Θ is defined as:

$$n_\theta = (n_0 + 1).n_1 + (n_1 + 1)$$

where $n_0 = n_y + n_u$ is the number of the input layer neurons and n_1 has the number of the hidden layer neurons. So the predictor can be noted $NN(n_y, n_u, n_d, n_1)$.

To avoid the saturation of the activation function (3), particularly for the output neuron, contained between 0 and 1, observed input and output data of the system must undergo a transformation reducing them between 0 and 1. However, the same notation for original and normalised data is used in the following.

3. Recursive prediction error method

The general framework of the learning rule used in the following is now presented. The backpropagation algorithm [21] is the first training method to estimate parameters of multilayer neural network and is a gradient algorithm designed to minimise the mean square error between the output of the network and the desired output.

The recursive prediction error or RPE algorithm, first introduced by Chen et al. [5,6] for training neural networks, is a general recursive parameter estimation method which minimises the prediction error using an approximation of the Gauss-Newton search direction. Only the version of the RPE algorithm introduced in [5] has been considered here. Billings and Jamaluddin [3] have shown that the RPE algorithm provides an effective method of learning neural networks. Backpropagation can be viewed as a simplified version of the RPE algorithm. Compared with backpropagation, RPE algorithm involves increased computational load at each iteration, but presents faster convergence, yielding to shorter global computational time. Furthermore, RPE removes the dependence of the estimation algorithm on the user selectable parameters such as learning rate and momentum. Indeed, with an inappropriate combination of these parameters, backpropagation performs badly. In any case, neither backpropagation nor RPE algorithm ensure to reach a final estimation corresponding to a global minimum of the criterion.

The RPE algorithm starts from the general criterion:

$$J(t, \Theta) = \gamma(t) \sum_{k=1}^{t} \beta(t, k) 1(\epsilon(k, \Theta), k) \tag{6}$$

where $\gamma(t)$ is the adaptation gain a time t with $\Sigma_{k=1}^{t} \gamma(t)\beta(t, k) = 1$, $\epsilon(k, \Theta)$ is the scalar prediction error and $1(\epsilon(k, \Theta), k)$ can be chosen as a quadratic function weighted by the innovation variance $\Lambda(k)$:

$$1(\epsilon(k, \Theta), k) = \tfrac{1}{2}\Lambda^{-1}(k)\epsilon^2(k, \Theta) \tag{7}$$

The minimisation of the criterion (6) can be performed according to:

$$\hat{\Theta}(t) = \hat{\Theta}(t-1) + f_t(\hat{\Theta}(t-1)) \tag{8}$$

where $\hat{\Theta}(t)$ is the estimate of Θ at time t and $f_t(\hat{\Theta}(t-1))$ is a search direction based on information about $J(t, \Theta)$.

The parameter vector Θ is estimated for each $t = 1, \dots, N$, where N is the number of available observations. For off-line estimation, the data set is presented several times and each presentation is called iteration. The Gauss-Newton search direction is used here and is defined by:

$$f_t(\Theta) = -[R(t)]^{-1} \nabla J(t, \Theta) \tag{9}$$

where $R(t)$ and $\nabla J(t, \Theta)$ are respectively the $n_\theta . n_\theta$ approximate hessian matrix and the $n_\theta .1$ gradient of $J(t, \Theta)$. The derivation is given by Ljung [15] and yields the general recursive prediction error algorithm:

$$\epsilon(t) = y(t) - \hat{y}(t/\hat{\Theta}(t-1)) \tag{10a}$$

$$\hat{\Theta}(t) = \hat{\Theta}(t-1) + \gamma(t) R^{-1}(t) \psi(t) \Lambda^{-1}(t) \epsilon(t) \tag{10b}$$

$$R(t) = R(t-1) + \gamma(t)[\psi(t) \Lambda^{-1}(t) \psi^{\mathrm{T}}(t) - R(t-1)] \tag{10c}$$

where $\psi(t)$ is the $n_\theta .1$ gradient of $\hat{y}$ with respect to Θ:

$$\psi(t) = \left[\frac{\partial \hat{y}(t/\Theta)}{\partial \Theta} \right].$$

The elements of $\psi(t)$ must be written depending on the location of the parameters in the network, in spirit of backpropagation. It can be first shown from Eq. (3) that:

$$\frac{\partial g(x)}{\partial x} = g(x)(1 - g(x)) \tag{11}$$

For the parameters of the output layer, Eqs. (11), (4a) and (4b) then yield:

$$\frac{\partial \hat{y}}{\partial w_k^2} = \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w_k^2} = \hat{y}(1 - \hat{y}) x_k^1 \tag{12a}$$

$$\frac{\partial \hat{y}}{\partial b^2} = \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial b^2} = \hat{y}(1 - \hat{y}) \tag{12b}$$

For the parameters of the hidden layer, Eqs. (11) and (4) yield:

$$\frac{\partial \hat{y}}{\partial w_{ik}^1} = \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial x_i^1} \frac{\partial x_i^1}{\partial z_i^1} \frac{\partial z_i^1}{\partial w_{ik}^1} = \hat{y}(1 - \hat{y}) w_i^2 x_i^1 (1 - x_i^1) x_k \tag{13a}$$

$$\frac{\partial \hat{y}}{\partial b_i^1} = \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial x_i^1} \frac{\partial x_i^1}{\partial z_i^1} \frac{\partial z_i^1}{\partial b_i^1} = \hat{y}(1 - \hat{y}) w_i^2 x_i^1 (1 - x_i^1) \tag{13b}$$

So, the differentiation of $\hat{y}$ with respect to $\theta_j (j = 1, \dots, n_\theta)$ can be summarised as follows:

$$\psi_j = \begin{cases} \hat{y}(1 - \hat{y}) x_k^1 & \text{if } \theta_j = w_k^2, \quad 1 \le k \le n_1 \\ \hat{y}(1 - \hat{y}) & \text{if } \theta_j = b^2 \\ \hat{y}(1 - \hat{y}) w_i^2 x_i^1 (1 - x_i^1) x_k & \text{if } \theta_j = w_{ik}^1, \quad 1 \le k \le n_0 \\ \hat{y}(1 - \hat{y}) w_i^2 x_i^1 (1 - x_i^1) & \text{if } \theta_j = b_i^1 \end{cases} \tag{14}$$

As developed in the appendix, the practical implementation of the algorithm (10) avoids to invert $R(t)$ at each step:

$$\hat{\Theta}(t) = \hat{\Theta}(t-1) + L(t)\epsilon(t) \tag{15a}$$

$$L(t) = \frac{P(t-1)\psi(t)}{\lambda(t)\Lambda(t) + \psi^{\mathrm{T}}(t)P(t-1)\psi(t)} \tag{15b}$$

$$P(t) = \frac{1}{\lambda(t)}\left[P(t-1) - L(t)\psi^{\mathrm{T}}(t)P(t-1)\right] \tag{15c}$$

where

$$P(t) = \gamma(t)R^{-1}(t) \tag{16}$$

and

$$\lambda(t) = \frac{\gamma(t-1)}{\gamma(t)}(1-\gamma(t)) \tag{17}$$

The so called forgetting factor is calculated practically by:

$$\lambda(t) = \lambda_0(1-\lambda(t-1)) + (1-\lambda_0)$$

4. Robustification of the algorithm

Large errors or outliers are quite difficult to be detected and picked out before identification and can cause the parameters to be highly biased. In order to tackle this problem, Puthenpura and Sinha [19] have developed a robust recursive identification method for linear dynamic systems. This scheme is weighted least squares algorithm with particular weights and is very similar to the robust Kalman filter obtained by Masreliez and Martin [16]. Based on the modelling of outliers due to Huber [11], it considers that the measurement noise $e(t)$ which contamines the noise free output is from the family D_μ, defined by:

$$D_\mu = \{D \mid D = (1-\mu)G + \mu H, 0 \le \mu \le 1\} \tag{18}$$

where G is the usual normal distribution, H an arbitrary symmetric long-tailed distribution and μ the probability of occurring large errors. In fact H is assumed to be also normal, but with a larger variance compared to G:

$$e(t) \sim (1-\mu)N(0, \sigma_1^2) + \mu N(0, \sigma_2^2) \tag{19}$$

where N represents a normal distribution, with $\sigma_2^2 > \sigma_1^2$.

The probability μ of occurring large errors being unknown, the preceding model is replaced by:

$$e(t) \sim [1-\delta(t)]N(0, \sigma_1^2) + \delta(t)N(0, \sigma_2^2) \tag{20}$$

where $\delta(t)=0$ for $|\epsilon(t)| \le M$ and $\delta(t)=1$ for $|\epsilon(t)| > M$, $\epsilon(t)$ is the prediction error and M a preassigned bound which can be taken as $3\sigma_1$ [1]. So the weighting factor appearing in Eq. (15) will be chosen as:

$$\Lambda(t) = [1 - \delta(t)]\sigma_1^2 + \delta(t)\sigma_2^2 \tag{21}$$

to reduce the influence of large innovations. Moreover, variances σ_1^2 and σ_2^2 can be updated as:

$$\begin{aligned} &\sigma_1^2(t) = \sigma_1^2(t-1) + \frac{1}{(t-\tau)}\left(\epsilon^2(t) - \sigma_1^2(t-1)\right) && \text{for } |\epsilon(t)| \le 3\sigma_1(t-1) \\ &\sigma_1^2(t) = \sigma_1^2(t-1) && \text{otherwise} \end{aligned} \tag{22a}$$

and

$$\begin{aligned} &\sigma_2^2(t) = \sigma_2^2(t-1) + \frac{1}{\tau}\left(\epsilon^2(t) - \sigma_2^2(t-1)\right) && \text{for } |\epsilon(t)| > 3\sigma_1(t-1) \\ &\sigma_2^2(t) = \sigma_2^2(t-1) && \text{otherwise} \end{aligned} \tag{22b}$$

with $\tau = 0$, for $t = 1$ and $\tau = \tau + 1$ whenever $|\epsilon(t)| > 3\sigma_1(t-1)$. As pointed out by Puthenpura and Sinha, τ is the estimated number of outliers.

$\sigma_2^2(0)$ can be chosen as $\sigma_2^2(0) = 3\sigma_1^2(0)$. $\sigma_1^2(0)$ should be chosen much greater than the real value of the noise variance so that in the beginning of the identification no residual ϵ(t) appears like outliers. With this choice, σ_1^2 converges to the true value of the noise variance. When σ_1^2 is close to the noise variance, outliers are detected and their influence on identification becomes insignificant. If $\sigma_1^2(0)$ is chosen very small (or null), all residuals have an absolute value $|\epsilon(t)|$ greater than $3\sigma_1(t-1)$, only σ_2^2 converges to the noise variance (but with the influence of outliers) and the parameter estimation is biased when outliers are present. If $\sigma_1^2(0)$ is chosen very large, the accommodation to outliers is just delayed.

In the next part, the three following algorithms are applied from Eq. (15) to a simulated example:

– NN (Neural Network) with $\Lambda(t) = 1$ and $\lambda(t) = 1$,

– FNN (Neural Network with Forgetting factor) where $\Lambda(t) = 1$ and $\lambda(t)$ is calculated by the recursive relation $\lambda(t) = \lambda_0(1 - \lambda(t-1)) + (1 - \lambda_0)$, with $\lambda_0 = 0.99$ and $\lambda(0) = 0.95$,

– RNN (Robust Neural Network) where $\lambda(k) = 1$ and $\Lambda(k)$ is computed by Eqs. (21) and (22).

5. Simulation results

A non-linear Hammerstein system example, introduced by Billings [3], is considered:

$$\begin{aligned} &y(k) = 0.8\,y(k-1) + 0.4\,\mathrm{NL}(u(k-1)) + e(k) \\ &\mathrm{NL}(k-1) = u(k-1) + u(k-1)^2 + u(k-1)^3 \end{aligned} \tag{23}$$

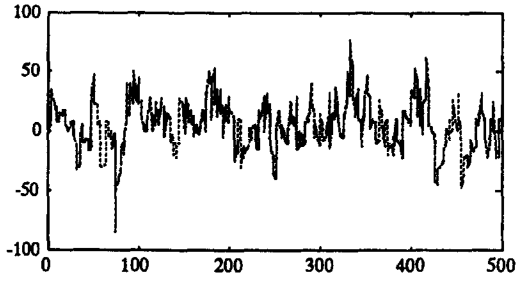

Fig. 3. Output signal contamined by outliers.

where $u(k)$ is the system input at time k, chosen as a sequence uniformly distributed between -4 and 4, in order to study the system on the whole non-linearity, $y(k)$ is the system output, $e(k)$ is a gaussian noise such as $e \sim N(0, \sigma_1^2)$ when no outliers are present. In order to show the influence of the outliers on the output, Fig. 3 presents the 500 output values, where the noise variance σ_1^2 is equal to 2.56 ($\sigma_1 = 1.6$), contamined by 25 randomly located outliers. These large errors are simply simulated by multiplying the original values of the noise by a factor f equal to 20.

Fig. 4 shows the difference between the preceding series and the corresponding outliers free and noise free simulation. The impact of the outliers filtered by the process dynamics can be noticed. Fig. 5 presents the difference between the noisy and outliers free series and the noise free series, which represents the noise filtered by the process, in order to show its variation range, significantly smaller than for the preceding figure.

For the different following trials, the neural predictor has the same structure $NN(n_y = 2, n_u = 2, n_d = 1, n_1 = 3)$ and the initial weights, randomly chosen between 0 and 1, are kept. The initial value of the covariance matrices P is chosen equal to 100 I and $\sigma_1^2(0)$ equal to 5 times the variance of the noise.

For the off-line identification considered in this part, the data set is presented 20 times (iterations). Moreover, a second data set is used for the (cross-)validation of the neural models. This fresh data set is called validation set and has characteristics similar to the original identification set, concerning the input shape, the distribution of the noise

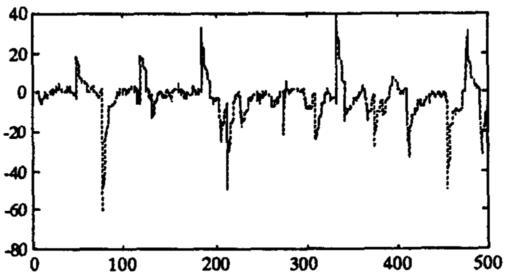

Fig. 4. Impact of outliers on the output data.

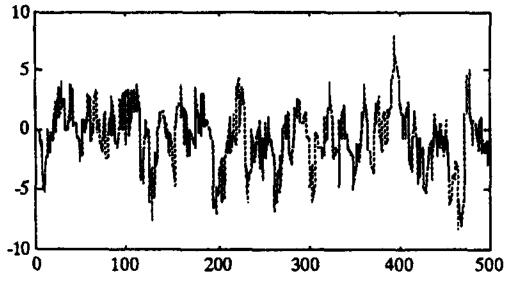

Fig. 5. Noise filtered by the process.

and its variance, but is outlier free. The following examples give the values of the residual criterion:

$$V=\frac{1}{N}\sum_{k=1}^{N}(\hat{y}(k)-y(k))^2$$

where $y(k)$ is the system output at time k, $\hat{y}(k)$ is the output estimated by the neural model obtained after 20 identification iterations and where $N=500$.

Fig. 6 presents, for $\sigma_1^2=0.64$, the variation of the residual criterion when the number of outliers (with multiplicative factor f equal to 25) is varied from 0 (no outliers) to 50.

Figure 6(a) concerns the data set used for identification, Fig. 6(b) the validation set. NN and RNN algorithms yield very stable results for the validation set contrary to those for FNN. But the residual criterion is reduced by 5 from NN to RNN.

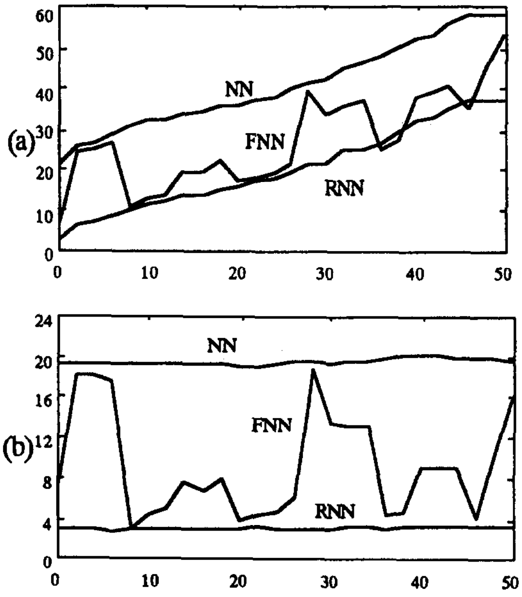

Fig. 6. Residual criterion w.r.t. the outliers number. (a) identification set, (b) validation set.

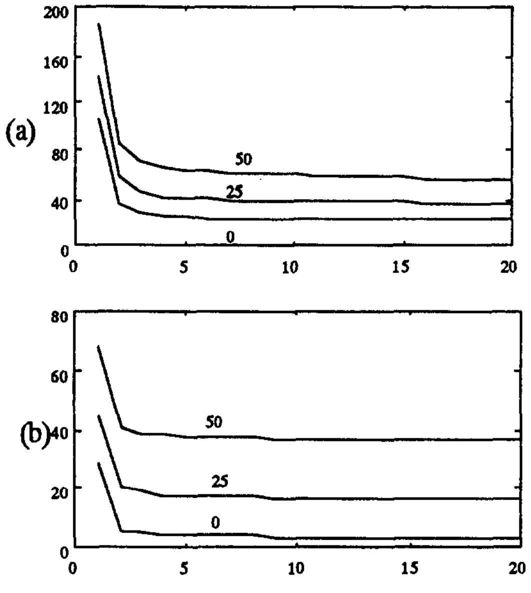

Fig. 7. Residual criterion with respect to iterations. (a) NN, (b) RNN.

Figs. 7(a) and 7(b) show the evolution during learning of the residual criterion for algorithms NN and RNN, respectively, for 0, 25 and 50 outliers. The convergence of the RNN algorithm appears clearly faster. However, as shown in Fig. 8, the variation of the criterion computed from the validation set is slower.

The second trail deals with a variation of the multiplicative factor f from 1 (no outliers) to 50, for 50 outliers and for $\sigma_1^2 = 0.64$. The results are similar as those of the preceding trial, as shown in Fig. 9.

In the third trial, for 50 outliers with multiplicative factor equal to 25, the noise variance σ_1^2 is varied from 0 (no noise) to 16. Results are given in Fig. 10. As for the preceding trial, the behaviour of the FNN algorithm appears disconcerting, such an algorithm being misadapted for contamined data. Note also on Fig. 10(b) that the RNN algorithm, superior to the simple NN, produces a good estimation of the noise variance from 2.5.

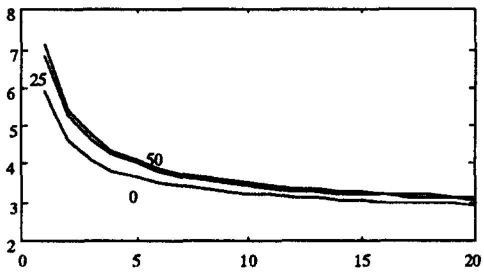

Fig. 8. Residual criterion for validation set (RNN).

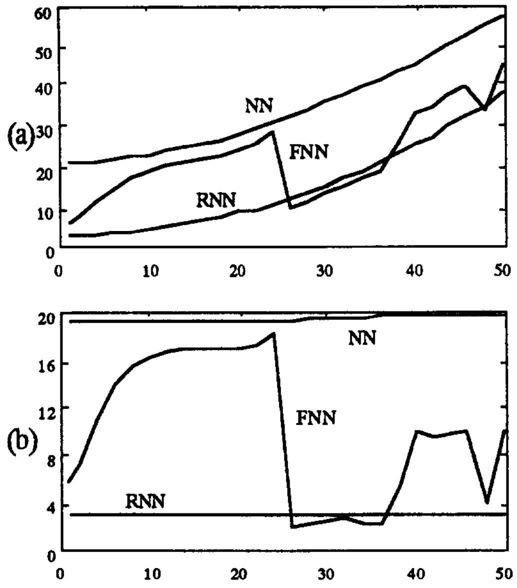

Fig. 9. Residual criterion with respect to the factor f. (a) identification set, (b) validation set.

In the last trail, σ_l^2 is fixed to 0.64 and a bias varying from 0 (no bias) to 20 is added to the last 50 values of the original outliers free noise. The results given in Fig. 11 are very significant. While increasing with RNN for the identification set as the bias

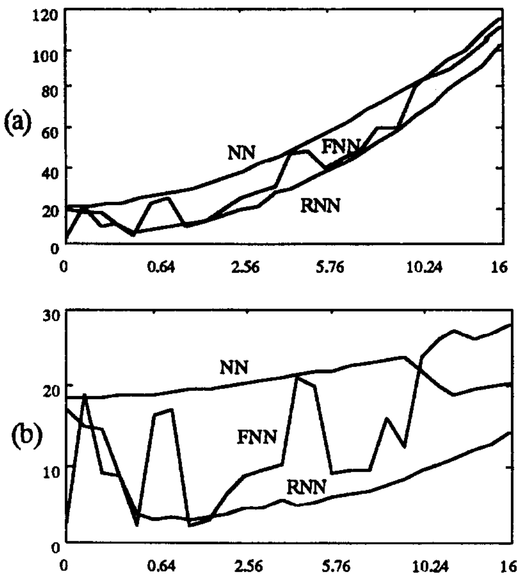

Fig. 10. Residual criterion with respect to σ_l^2. (a) identification set, (b) validation set.

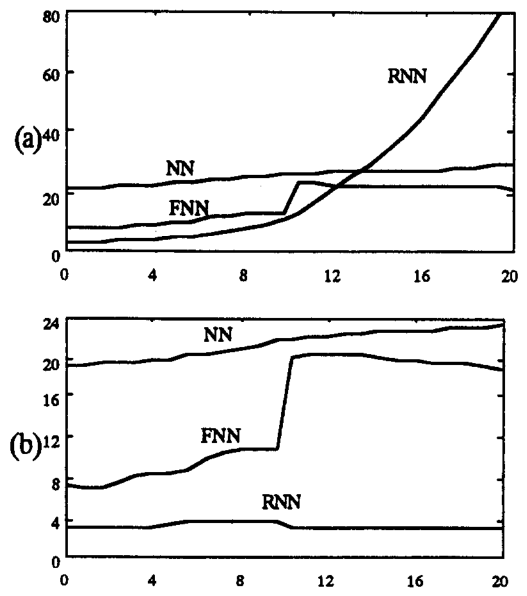

Fig. 11. Residual criterion w.r.t. bias magnitude. (a) identification set, (b) validation set.

increases, the residual criterion is remarkably stable for the validation set, contrary to the other algorithms.

These last results confirm the interest of the presented robust non linear predictor for detection of process changes.

6. Conclusion

The neural nets have a structure which allows, with the adaptation of the backpropagation, the use of the various and sometimes classical parameter estimation algorithms. The problem of non-linear Single Input Single Output system identification in the presence of outliers in data has been considered. Combining the capabilities of neural networks to solve non-linear problems by learning and a robust recursive prediction error learning rule based on the modelling of the errors, a new algorithm has been drawn up. The results obtained suggest that this algorithm can be employed for identification from contamined data, but also for failure detection and for robust control. The proposed method can be easily extended to MIMO systems.

Acknowledgement

The authors gratefully acknowledge the reviewers for their instructive comments.

Appendix

Let us consider the general prediction error algorithm introduced in (10):

$$\varepsilon(t) = y(t) - \hat{y}(t/\hat{\Theta}(t-1)) \tag{A1a}$$

$$R(t) = [1-\gamma(t)]R(t-1) + \gamma(t)\psi(t)\Lambda^{-1}(t)\psi^{\mathrm{T}}(t) \tag{A1b}$$

$$\hat{\Theta}(t) = \hat{\Theta}(t-1) + \gamma(t)R^{-1}(t)\psi(t)\Lambda^{-1}(t)\varepsilon(t) \tag{A1c}$$

To avoid inverting $R(t)$ at each step, it is convenient to introduce:

$$P(t) = \gamma(t)R^{-1}(t) \tag{A2}$$

and apply to (A1b) the matrix inversion lemma:

$$(A+BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1} \tag{A3}$$

Taking $A = [1-\gamma(t)]R(t-1)$, $B = D^{\mathrm{T}} = \psi(t)$, and $C = \gamma(t)\Lambda^{-1}(t)$ gives:

$$P(t) = \frac{\gamma(t)}{1-\gamma(t)}\left[R^{-1}(t-1) - \frac{R^{-1}(t-1)\psi(t)\psi^{\mathrm{T}}(t)\dfrac{R^{-1}(t-1)}{1-\gamma(t)}}{\dfrac{\Lambda(t)}{\gamma(t)} + \psi^{\mathrm{T}}(t)\dfrac{R^{-1}(t-1)}{1-\gamma(t)}\psi(t)}\right] \tag{A4}$$

Let us recall that the forgetting factor $\lambda(t)$ is linked to $\gamma(t)$ by (17):

$$\lambda(t) = \frac{\gamma(t-1)}{\gamma(t)}[1-\gamma(t)] \tag{A5}$$

Eq. (A2) can be rewritten as:

$$P(t-1) = \gamma(t-1)R^{-1}(t-1) \tag{A6}$$

Combining (A5) and (A6) gives:

$$\frac{P(t-1)}{\lambda(t)} = \frac{\gamma(t)}{1-\gamma(t)}R^{-1}(t-1) \tag{A7}$$

and introducing (A7) in (A4) gives quite directly:

$$P(t) = \frac{1}{\lambda(t)}\left[P(t-1) - \frac{P(t-1)\psi(t)\psi^{\mathrm{T}}(t)P(t-1)}{\lambda(t)\Lambda(t) + \psi^{\mathrm{T}}(t)P(t-1)\psi(t)}\right] \tag{A8}$$

Taking:

$$L(t) = \gamma(t)R^{-1}(t)\psi(t)\Lambda^{-1}(t) \tag{A9}$$

in (A1c) gives:

$$\hat{\Theta}(t) = \hat{\Theta}(t-1) + L(t)\varepsilon(t) \tag{A10}$$

Introducing (A2) and (A8) in (A9) yields after some calculations:

$$L(t)=\frac{P(t-1)\psi(t)}{\lambda(t)\Lambda(t)+\psi^{\mathrm{T}}(t)P(t-1)\psi(t)} \tag{A11}$$

and substituting (A11) in (A8) gives with (A1a), (A11) and (A10) the final algorithm:

$$\varepsilon(t)=y(t)-\hat{y}\big(t/\hat{\Theta}(t-1)\big) \tag{A12a}$$

$$L(t)=\frac{P(t-1)\psi(t)}{\lambda(t)\Lambda(t)+\psi^{\mathrm{T}}(t)P(t-1)\psi(t)} \tag{A12b}$$

$$P(t)=\frac{1}{\lambda(t)}\big[P(t-1)-L(t)\psi^{\mathrm{T}}(t)P(t-1)\big] \tag{A12c}$$

$$\hat{\Theta}(t)=\hat{\Theta}(t-1)+L(t)\varepsilon(t) \tag{A12d}$$

References

[1] K.J. Aström, Maximum likelihood and prediction error methods, *Automatica* 16 (1980) 551–574.

[2] N. Bhat and T.J. McAvoy, Use of neural nets for dynamic modelling and control of chemical process systems, *Comput. Chem. Engng.* 14 (1990) 573–582.

[3] S.A. Billings, H.B. Jamaluddin and S. Chen, A comparison of the backpropagation and recursive prediction error algorithms for training neural networks, *Mechanical Systems and Signal Processing* 5 (1991) 233–255.

[4] G.E.P. Box and G.M. Jenkins, *Time Series Analysis, Forecasting and Control* (Holden Day, San Francisco, 1970).

[5] S. Chen, S.A. Billings and P.M. Grant, Non-linear system identification using neural networks, *Int. J. Control* 51 (1990) 1191–1214.

[6] S. Chen, C.F.N. Cowan, S.A. Billings and P.M. Grant, Parallel recursive prediction error algorithm for training layered neural networks, *Int. J. Control* 51 (1990) 1215–1228.

[7] G. Cybenko, Approximation by superpositions of a sigmoidal function, *Math. Control Signals Systems* 2 (1989) 303–314.

[8] K. Funahashi, On the approximate realisation of continuous mappings by neural networks, *Neural Networks* 2 (1989) 183–192.

[9] S. Grossberg, Non-linear neural networks: principles, mechanisms, and architectures, *Neural Networks* 1 (1988) 17–61.

[10] J.C. Hoskins and D.M. Himmelblau, Artificial neural network models of knowledge representation in chemical, *Comput. Chem. Engng.* 12 (1988) 881–890.

[11] P.J. Huber, Robust estimation of a location parameter, *Ann. Math. Stat.* 35 (1964) 73–101.

[12] T. Kohonen, *Self-Organisation and Associative Memory* (Springer-Verlag, New York, 1988).

[13] M.A. Kramer and J.A. Leonard, Diagnosis using backpropagation neural networks – Analysis and criticism, *Comput. Chem. Engng.* 14 (1990) 1323–1338.

[14] R.P. Lippman, An introduction to computing with neural nets, *IEEE ASSP Mag.* 4 (1987) 4–22.

[15] L. Ljung, *System Identification: Theory for the User* (Prentice-Hall, Englewood Cliffs, NJ, 1987).

[16] C.J. Masreliez and R.D. Martin, Robust bayesian estimation for the linear model and robustifying the Kalman filter, *IEEE Trans. A.C.* 22 (1977) 361–371.

[17] E.P. Nahas, M.A. Henson and D.E. Seborg, Non-linear internal model control strategy for neural network models, *Comput. Chem. Engng.* 16 (1992) 1039–1057.

[18] K.S. Narendra and K. Parthasarathy, Identification and control of dynamical systems using neural networks, *IEEE Trans. on Neural Networks* 1 (1990) 4–27.

[19] S. Puthempura and N.K. Sinha, A robust recursive identification method, *Control-Theory and Advanced Technology* 6 (1990) 683–695.
[20] A. Rajavelu, M.T. Musavi and M.V. Shirvarkar, A neural network approach to character recognition, *Neural Networks* 2 (1989) 387–393.
[21] D.E. Rumelhart and J.L. McClelland, *Parallel Distributed Processing* (MIT Press, Cambridge, 1986).
[22] T. Söderström and P. Stoica, *System Identification* (Prentice-Hall, Englewood Cliffs, N.J, 1987).
[23] B. Widrow and M.A. Lehr, 30 years of adaptive neural networks: perceptron, madaline, and backpropagation, *Proc. IEEE* 78 (1990) 1415–1442.
[24] Q. Xue, Y.H. Hu and W.J. Tompkins, Neural-network-based adaptive matched filtering for QRS detection, *IEEE Trans. on Biomedical Engineering* 39 (1992) 317–329.

Gérard Bloch received the Ph. D. in Electrical Engineering in 1988 from University Henri Poincaré (Nancy, France). He is an associate professor at the Ecole Supérieure des Sciences et Technologies de l'Ingénieur de Nancy (University Henri Poincaré, Nancy, France) where he is in charge of the last year leading to M. Sc. in Automatic Control. He is at the Centre de Recherche en Automatique de Nancy (CNRS URA 821). His research interests include process diagnosis, system identification and application of neural networks to automatic control.

Philippe Thomas received his M. Sc. in Automatic Control in 1993 from University Henri Poincaré (Nancy, France) and is currently a Ph. D. student in the field of neural networks at the Centre de Recherche en Automatique de Nancy (CNRS URA 821). His research interests include non linear system identification with neural networks or parametric methods.

Didier Theilliol received the Ph. D. in Neural Networks domain from the University Henri Poincaré (Nancy, France) in 1993. He is an associate professor of control engineering at the Centre de Recherche en Automatique de Nancy (CNRS URA 821). His research interests is the application of process diagnosis.