AN ON-LINE LEARNING APPROACH: A METHODOLOGY FOR TIME VARYING APPLICATIONS

N.M. ROEHL[*]
Catholic University, PUC-Rio
C.P. 38063, 22452-970, R.J., Brazil

and

C.E. PEDREIRA
Catholic University, PUC-Rio
C.P. 38063, 22452-970, R.J., Brazil
E-mail: pedreira@ele.puc-rio.br

Abstract: In this paper a new procedure to continuously adjust weights in a multi layered neural network is proposed. The network is initially trained by using traditional Backpropagation algorithm. After this first step, non-linear programming technique is used in order to properly on line calculate the new weights sets. This methodology is tailored to be used in time varying (non-stationary) models, eliminating necessity of retraining. Numerical results for a chaotic time series and an electricity load forecasting applications are presented.

Key Words: non-stationary data, neural networks, on-line training, forecasting.

1. INTRODUCTION

The main challenge when dealing with non-stationary data is to balance the information related to recent data with the information previously incorporated by the model. Multi layered neural networks [1] have been successfully used in a variety of relevant problems when invariance assumptions can be properly made. An attractive feature of neural networks is its natural ability to deal with non-linear data. The procedure proposed in this paper is directed to neural non-stationary models. Few papers can be found in the literature concerning time varying connectionist models [2][3][4].

[*] Now at CALTECH, Pasadena, U.S.A.

The proposed algorithm [5] could be divided in two phases. The first one consists in training a neural network by using a standard Backpropagation scheme. The size of the data set is crucial in this phase, it must be representative of the system and at the same time not very large, so that stationary hypotheses can be assumed in this set. In the second phase the system is put in use and at the same time an on-line training algorithm is turned on updating the weights in accordance to new incoming data. The weights are automatically adjusted when necessary only. In section 2, it is proved that if the series remain stationary for a period of time, the weight set is maintained.

The main objective is to keep the error related to the latest incoming data within a pre-established tolerance, while maximising the information incorporated up to that point. By choosing a balance parameter the designer is able to decide the relevance, in relation to incorporated information, that should be attributed to the new data. This is an interesting feature of the algorithm. For instance, in a situation of temporary large volatility, there is the possibility of trying not to damage the old model. The pre-established tolerance for the error of the recent incoming data generates an optimisation problem that is solved by a non-linear programming technique. The solution of this problem, as will be seen in the continuation, produces the new updated weight set. Off course an alternative to the proposed method would be to retrain the neural network, by using Backpropagation, every time a change in the model is detected, but this would not be an acceptable solution for most on line systems to which one needs a fast and automatic reaction.

2. THE PROBLEM ANALYSIS

Let **W** be the weight set produced by training a neural network with the data pairs $\{(x_i,d_i), i = 1,...,n\}$. Let (x_{n+1},d_{n+1}) be a new data pair that is presented to the system. The aim is to determine a new weight set **W'= W +** D**W** such that the following associated energy function is minimised, subject to the new data constraint.

$$\begin{aligned} &MIN\ E = \frac{1}{2}\sum_{i=1}^{n} \left\| d_i - y_i \right\|^2 \\ &\text{subject to } -e^{\ell} \le \left(d_i^{\ell} - y_i^{\ell} \right) \le e^{\ell},\ \ell = 1 \ldots o \end{aligned} \qquad (P_\varepsilon)$$

where $y \in \Re^o$ is the network output vector and ε^{ℓ}, $\ell = 1,...,o$ is the associated error tolerance. This objective function reflects the desire of minimising the error concerning the original training data while the constraint will keep the error associated with the last data within a specified tolerance. Note that this formulation establishes the trade off between the old model information and the new data. The bound parameters e^{ℓ}, $\ell = 1 \ldots o$ are chosen by the designer, allowing a tolerance control. This flexibility may constitute in a very useful tool, for

instance when a temporary large amount of noise affects the system, or a transient increase of the volatility level occurs. This adaptation is done independently from a moving window that discards a parcel of the old information that is not explaining the system anymore.

To improve analytic and implementation tractability, the constraint of problem P_ε is linearized by applying a first order Taylor series expansion. The optimisation problem can then be rewritten as a function of the weight changes ΔW. As a result, the feasible region is expressed as the intersection of the hyperspace defined by the linearized constraint and the region where the linearity assumption is valid. Algebraic details concerning these procedures can be found in Appendix 1.

The energy function of problem P_ε can also be rewritten in a compact form as:

$$J(z)=\frac{1}{2}z^t Kz$$

where $K=S^t S$, S is the matrix of sensitivity of y over a weight change and $z\equiv\left[\Delta W_{vec}^t:\left(\Delta V^t\right)_{vec}^t\right]^t \qquad z\in\Re^{p+q}$.

The algebraic calculation of matrix K is presented in Appendix 2.
Problem P_ε can now restated as:

$$\begin{aligned} &\text{MIN } J = \frac{1}{2}\, z^t Kz \\ &\text{subject to } c_1(\varepsilon) \le Az \le c_2(\varepsilon) \\ &\qquad z \in \beta \end{aligned} \qquad (P_\beta)$$

Let's label this problem as P_β. Here we define the linearity region

$$\beta \equiv \{\, z \in \Re^q : \|z_i\| \le \delta,\ i=1\ldots q \,\} \text{ for some } \delta > 0. \qquad (D_\beta)$$

Note that z represents the weight changes and c_1, c_2 are bounds on the activation error of the output units for the new data.
Problem P_β can be solved by a variety of non-linear programming algorithms (e.g. projected gradient, convex simplex). A gradient projection type algorithm (c.f. [1]) was used in the present case to solve this quadratic programming problem. In general, these methods lead to local optimality of the cost function. However, in this case, global convergence is guaranteed since one is minimising a convex functional over a convex feasible region (c.f. [2]). The solution of problem P_β is the wanted weight changes z.

A valuable characteristic of this procedure is the absence of *drift*. This important feature means that, if the new incoming data does not incorporate a variation in the original model, then the network weights will not be modified. This property can be proved by letting

$\{W_N, V_N\}$ be the weight set resulting from the training phase with the set of input-output pairs $D=\{(x_i, d_i); 1 \le i \le N\}$. Let (x_{N+1}, d_{N+1}) be a new input-output pair not in D that satisfies the network relationships $y = f_1[V_N^t u]$ *and* $u = f_2[W_N^t x]$, i.e., $d_{N+1} = f_1[V_N^t u]$. Therefore, the feasible region of problem P_β includes the origin. As global convergence is guaranteed, the algorithm will converge z = 0, resulting in $\Delta W = 0$ *and* $\Delta V = 0$.

3. NUMERICAL RESULTS

3.1 A Synthetic Experiment: Forecasting a Chaotic Time Series

In order to illustrate the forecasting potentiality of the proposed on-line learning scheme, a controllable experiment is presented. Chaotic time series data were generated based on the iterative equation $x_{t+1} = 4x_t(1 - x_t)$. A training set composed of 100 input-output pairs (x_t, x_{t+1}) was applied at the Backpropagation training phase. Non stationary behaviour was introduced by modifying the underlying equation to $x_{t+1} = 3.8x_t(0.95 - x_t)$. A data set with 50 patterns was produced by this new generator function.
The following three performance measures were used:

$$\text{MAPE} = \frac{1}{\text{M}} \sum_{\text{i}=1}^{\text{M}} \frac{|\text{d}_\text{i} - \text{y}_\text{i}|}{\text{d}_\text{i}} * 100;$$

$$\text{nRMSE} = \frac{\sqrt{\frac{1}{M} \sum_{i=1}^{M} (d_i - y_i)^2}}{STD(d)}$$

$$\text{U-Theil} = \frac{\sqrt{\frac{1}{M} \sum_{i=1}^{M} (d_i - y_i)^2}}{\sqrt{\frac{1}{M} \sum_{i=1}^{M} (d_i - d_{i-1})^2}}$$

where M is the number of points in the data set, y and d are the network output and the series correct values respectively.
The first experiment consists in setting the linearity boundary parameter δ (see definition D_β) equal 2 and evaluating the outcome for different values of error tolerance ε for out of sample data (Table 1). Other small values for δ would produce similar results.

TEST DATA	MAPE (%)	nRMSE	U-Theil
BACKPROP.	26.9592	0.1473	0.3678
$\varepsilon = 0$ %	4.7791	0.0333	0.0831
$\varepsilon = 1$ %	4.0393	0.0310	0.0774
$\varepsilon = 2$ %	3.7147	0.0288	0.0720
$\varepsilon = 3$ %	3.9036	0.0289	0.0722
$\varepsilon = 5$ %	4.6423	0.0339	0.0846
$\varepsilon = 10$ %	7.4311	0.0566	0.1413

TABLE 1 Out of sample data performance comparison

It could be noticed a significant improvement in the forecasting performance when using the proposed method in comparison to classical Backpropagation. The on-line scheme performed better for all the three measures and for all the choices of ε parameter. Note that, for larger values of ε the system behaviour tends to the standard Backpropagation, and for $\varepsilon = 0$ the results approaches the obtained in [3]. In this example the optimal choice of ε is 2 %, leading to the interpretation that some error related to the new incoming data should be allowed in order not to strongly damage the previous information incorporated by the system. Parameter $\varepsilon = 0$ would force the error concerned with the recent past data to be equal to zero since the constraint in problem P_ε would have been reduced to an equality.

3.2 A Real Data Experiment: Electricity Load Forecasting

Our second numerical experiment consists in applying the on-line algorithm to a long-term electricity load series. This series was monthly (average) collected in Brazil. Please note that we approached these data with the intention of exemplifying the proposed algorithm. It is out of the scope of this paper to produce a solution for the quite difficult problem of load forecasting. Off course we believe that this can be achieved by using the proposed method but in this case one should dedicate attention to the pre treatment of the data and maybe include some climatic variables e.g. temperature. From a total of 160 points available, the first 138 were used for the in sample training phase while the remaining 22 months were used for an out of sample validation phase. We chose, after the usual experimentation, a six units one hidden layer architecture. The input layer is composed by six input units corresponding time delayed observations. The output unit corresponds to the one step ahead forecasting. A different partition of the data set, e.g. more points for validation, would lead to an overfitting situation because of the number of parameters (weights) that must be estimated. In this case the on-line algorithm would be favoured in the comparison to Backpropagation because of its ability to adapt to new data. The resulting out of sample performance is shown in Table 2 for different values of parameter ε.

TEST DATA	MAPE (%)	nRMSE	U-Theil
BACKPROP.	3.6352	0.7805	1.0491
e = **0 %**	4.9361	0.8996	1.2092
e = **3 %**	4.3872	0.8297	1.1152
e = **10 %**	4.1133	0.7905	1.0625
e = **12 %**	3.6869	0.7886	1.0600
e = **15 %**	3.4458	0.7468	1.0038
e = **20 %**	3.5635	0.7670	1.0309

TABLE 2 - Out of sample errors

Figure 2 illustrates a behaviour comparison between proposed (ε = 15%) (o) and Backpropagation (*) approaches. The data is represented by the solid line.

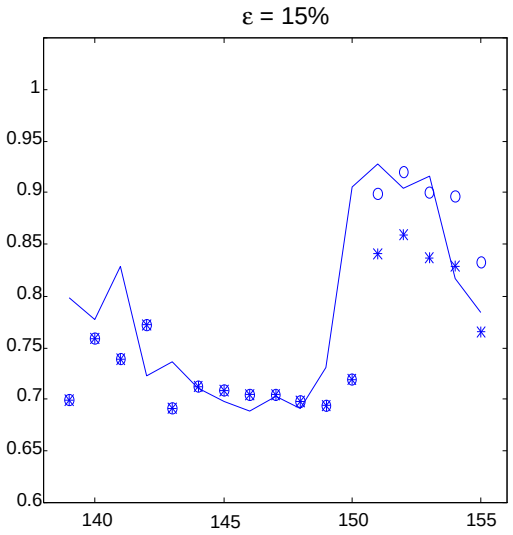

FIGURE 2

(o) --> (on-line forecasting); * --> (pure Backpropagation); solid line --> (data)

4. FINAL REMARKS

In this paper a new on-line training approach to the classical Backpropagation procedure was introduced. The algorithm has shown very encouraging results for prediction applications with non-stationary data. The method allows continuous weights adjustments as the underlying system dynamics change with time. One of the main advantages of our approach is the flexibility of controlling the trade off problem between fitting new incoming data and causing minimum damage to the original learned information. The proposed approach allows the designer to specify the new data matching accuracy. The smaller the chosen parameter ε,

the better the last data will be fitted but more damage will be imposed to the previously trained network.

Numerical Results for a synthetic chaotic series and for a real electricity load time series were presented as examples. Both experiments produced interesting behaviour in accordance to expected from the theoretical point of view.

We believe that the design of adaptive schemes for setting the tolerance parameter ε may bring further improvements, we leave this as a suggestion for future work.

REFERENCES:

[1] Haykin S. , 'Neural Networks - A Comprehensive Foundation' Macmillan Publ. 1999.

[2] Pedreira C.E. and Roehl N.M., 'An Adaptively Trained Neural Network', Proceedings of IEEE International Joint Conference on Neural Networks, vol.1, Nagoya, 1993.

[3] Park D.C., El-Sharkawi M.A. and MarksII R.J., 'An Adaptively Trained Neural Network', IEEE Trans on Neural Networks, Vol 2, N.3, 1991.

[4] Roehl N.M. and Pedreira C.E., 'On-Line Learning for Multi Layered Neural Network: Application to Time Series Prediction', in Neural Networks in Capital Markets, World Scientific Publ., 1995.

[5] Roehl N.M., 'Neural Networks On-line Training: A methodology for time varying Environments', PhD theses (in Portuguese), Catholic University of Rio de Janeiro, 1998.

[6] Gill P.E., Murray W. and Wright M.H, Practical Optimisation, Academic Press, 1981.

APPENDIX 1 - Feasible Region Determination

Let a three layer network be described by

$$y = f_1[V^t u] \qquad u = f_2[W^t x] \tag{1}$$

$y \in \Re^o$ is the output of the multi layered network, $u \in \Re^h$ represents the activation of the hidden neurones and $x \in \Re^I$ is the input data vector. Matrices W and V contain the weights linking input to hidden layer and hidden to output layer, respectively. The vector functions $f_1 \in \Re^o$ and $f_2 \in \Re^h$ are such that $f_1(.) = [f_1^1(.) \ldots f_1^o(.)]^t$ and $f_2(.) = [f_2^1(.) \ldots f_2^h(.)]^t$, where $f_1^i(.)$ and $f_2^i(.)$ are non-decreasing differentiable functions.

Let us assume that the new weight set (W',V') can be written as V'= V + ΔV and W'= W + ΔW, where the increments ΔV and ΔW are in $\Re^{hxo}$ and $\Re^{Ixh}$, respectively. In order to simplify notation we will omit the element denoting superscript. Now, for (x_{N+1}, d_{N+1}) , substituting (1) in problem P_ε constraint yields

$$-e \le d_{N+1} - f_1[(V + \Delta V)^t u] \le e$$

and so,

$$d_{N+1} - e \le f_1[(V + \Delta V)^t u] \le d_{N+1} + e \ .$$

Finally, because $(f_1^l)^{-1}$ is a monotonic increasing function $\forall l = 1, \ldots, o$, it follows:

$$f_1^{-1}(d_{N+1} - e) \le [(V + \Delta V)^t u] \le f_1^{-1}(d_{N+1} + e) \ . \tag{2}$$

On the other hand, applying a first order Taylor series expansion one gets

$$u \approx f_2(W^t x_{N+1}) + J f_2(W^t x_{N+1}) \Delta W^t x_{N+1} \ . \tag{3}$$

By substituting the approximation (3) into inequality (2), one obtains:

$$f_1^{-1}(d_{N+1} - e) - V^t f_2(W^t x_{N+1}) \le V^t J f_2(W^t x_{N+1}) \Delta W^t x_{N+1} + \Delta V^t f_2(W^t x_{N+1}) \le f_1^{-1}(d_{N+1} + e) - V^t f_2(W^t x_{N+1}). \tag{4}$$

Note that this approximation is reasonable if $\|r(\Delta b)\| \le g$, where $r(\Delta b)$ is the truncation error associated with the Taylor series expansion and γ is an upper bound such that $g > M\|x\|\left(\sum_{i=1}^{h}\|\Delta W_i\|^2\right)^{1/2}, \forall i = 1, \ldots, h$ and $M = \sup_{0<t<1}\|\nabla_b f(b + t\Delta b) - \nabla_b f(b)\|$. We can shortly demonstrate this fact. By definition $b = W^t x$ and $\Delta b = \Delta W^t x$ and so, $\|\Delta b\| \le \|x\|\left(\sum_{i=1}^{h}\|\Delta W_i\|^2\right)^{1/2}$ resulting in $g > M\|\Delta b\|$. However, by the mean value inequality theorem $\|r(\Delta b)\| \le M\|\Delta b\|$

and so, γ is an upper bound to the first order Taylor series expansion truncation error. This result implies in limiting perturbations on W, i.e., by developing γ's inequality one obtains $\left|\Delta w_{ij}\right|_{\max} < g \big/ M\|x\|\sqrt{hI}$. Another necessary assumption to validate inequality (4) is that $\Delta V_{ij} << V_{ij}$, $\forall\, i=1,\ldots,h \;\; e \;\; j=1,\ldots,o.$ For this condition to be valid we may choose β > 0 small such that $\Delta V_{ij} \le b\; V_{ij}$.

By limiting the perturbations on W and V a region where a linearization assumption is acceptable can be defined. Now, by rearranging ΔW and ΔV^t into vector form, inequality (4) can be written in a compact form as:

$$c_1^i(e) \le (Az)^i \le c_2^i(e) \;,\; i=1,\ldots,o \tag{5}$$

where $A=\left[A_p : A_q\right], A \in \Re^{o(p+q)} \qquad p=Ixh \;\; e \;\; q=hxo$

$z=\left[\Delta W_{vec}^t : \left(\Delta V^t\right)_{vec}^t\right]^t \qquad z \in \Re^{p+q}$

$A_p \in \Re^{oxp}$ is a solution of $A_p \Delta W_{vec} = V^t J f_2\left(W^t x_{N+1}\right)\Delta W^t x_{N+1}$

$A_q = \left[A_i\right] \in \Re^{oxq}$ where $A_i = \left[\underbrace{0 \cdots 0}_{h(i-1)}\; f_2\left[W^t x\right]^t \; 0 \cdots 0\right]$, $i=1,\ldots,o$

$c_1(e) \equiv f_1^{-1}\left(d_{N+1}-e\right) - V^t f_2\left(W^t x_{N+1}\right)$, $c_1(e) \in \Re^o$

$c_2(e) \equiv f_1^{-1}\left(d_{N+1}+e\right) - V^t f_2\left(W^t x_{N+1}\right)$, $c_2(e) \in \Re^o$

The feasible region can now be defined as the intersection between the linearity region and the hyperspace defined by constraint (5).

APPENDIX 2 - Matrix K Determination

The energy function of problem P_ε can also be rewritten in a compact form as follows:

$$J(z) = \frac{1}{2} z^t K z$$

where $K = S^t S$ and S is the matrix of sensitivity of y over a weight change. Next, the algebraic procedure that leads to matrix K determination is presented.

The main idea consists in finding the sensitivity of y_i over a weight change. Let us rewrite the energy function as follows:

$$E = \sum_{i=1}^{N} E_i$$

where $E_i = \frac{1}{2}\left\| d_i - y_i \right\|^2 = \frac{1}{2}\sum_{l=1}^{o}\left(d_i^l - y_i^l\right)^2 = \sum_{l=1}^{o} E_i^l$.

The sensitivity associated to input and output weights are

$$\frac{\partial E_i}{\partial w_{jk}} = \sum_{l=1}^{o}\frac{\partial E_i^l}{\partial w_{jk}} = \sum_{l=1}^{o} -\left(d_i^l - y_i^l\right)\frac{\partial y_i^l}{\partial w_{jk}} \tag{5}$$

$$\frac{\partial E_i}{\partial v_{kl}} = \sum_{l=1}^{o}\frac{\partial E_i^l}{\partial v_{kl}} = \frac{\partial E_i^l}{\partial v_{kl}} = -\left(d_i^l - y_i^l\right)\frac{\partial y_i^l}{\partial v_{kl}} \tag{6}$$

respectively. The weight interconnection between input neuron j and hidden neuron k, w_{jk} , is the jk^{th} element of W. The weight interconnection between hidden neuron k and output neuron l, v_{kl} , is the kl^{th} element of V.

Let us define $y_i^l = f_1^l\left[sum_y^l\right]$ *onde* $sum_y^l = \sum_{k=1}^{h} v_{kl}u_k$, $l = 1,\ldots,o$ and

$u_k = f_2^l\left[sum_{u_k}\right]$ *onde* $sum_{u_k} = \sum_{j=1}^{I} w_{jk}x_j$, $k = 1,\ldots,h$.

One can write

$$\begin{aligned}\frac{\partial y_i^l}{\partial w_{jk}} &= \left(\frac{\partial y_i^l}{\partial sum_{y^l}}\right)\left(\frac{\partial sum_{y^l}}{\partial u_k}\right)\left(\frac{\partial u_k}{\partial sum_{u_k}}\right)\left(\frac{\partial sum_{u_k}}{\partial w_{jk}}\right) \\ &= \frac{\partial f_1^l(x)}{\partial x}\bigg|_{x=y_i^l} v_{kl}\,\frac{\partial f_2^k(x)}{\partial x}\bigg|_{x=u_k} x_j \\ &= y_i^l\left(1-y_i^l\right)v_{kl}u_k\left(1-u_k\right)x_j \equiv SW_{i,jk}^l\end{aligned} \tag{7}$$

$$\begin{aligned}\frac{\partial y_i^l}{\partial v_{kl}} &= \left(\frac{\partial y_i^l}{\partial sum_{y^l}}\right)\left(\frac{\partial sum_{y^l}}{\partial v_{kl}}\right) = \frac{\partial f_1^l(x)}{\partial x}\bigg|_{x=y_i^l} u_k \\ &= y_i^l\left(1-y_i^l\right)u_k \equiv SV_{i,kl}^l\end{aligned} \tag{8}$$

where $SW_{i,jk}^l$ and $SV_{i,kl}^l$ are the sensitivity of y_i^l due to small changes in w_{jk} and v_{kl} , respectively. The energy variation due to weight changes associated to the i^{th} data can be calculated from equations (5), (6), (7), and (8):

$$\Delta E_i = \sum_{j,k}\left(\frac{\partial E_i}{\partial w_{jk}}\right)\Delta w_{jk} + \sum_{k,l}\left(\frac{\partial E_i}{\partial v_{kl}}\right)\Delta v_{kl}$$

$$= \sum_{j,k} SW_{i,jk}\Delta w_{jk} + \sum_{k,l} SV_{i,kl}\Delta v_{kl}$$

$$\therefore \quad \Delta E_i = SW_i^t \Delta W_{vec} + SV_i^t \left(\Delta V^t\right)_{vec}$$

where $SW_i^t \equiv \left[SW_{i,1} \ldots SW_{i,p}\right]$ and $SV_i^t \equiv \left[SV_{i,1} \ldots SV_{i,q}\right]$, $i = 1, \ldots, N$. and, by using matricial notation one can write:

$$\Delta E = \left[\Delta E_1 \; \Delta E_2 \cdots \Delta E_N \;\right]^t = S\, z \tag{9}$$

where $S = \begin{bmatrix} SW_1^t : SV_1^t \\ SW_2^t : SV_2^t \\ \vdots \quad\; \vdots \\ SW_N^t : SV_N^t \end{bmatrix} \in \Re^{Nx(p+q)}$. (10)

The objective function can also be rewritten in terms of energy variation as

$$J = \frac{1}{2}\sum_{i=1}^{N}\left(E_i^{'} - E_i\right)^2$$

where E_i and $E_i^{'}$ are the errors of the ith datum considering the weight sets $\{W, V\}$ and $\{W', V'\}$, respectively. And so, from (9) one gets:

$$J = \frac{1}{2}\sum_{i=1}^{N}\Delta E_i^2 = \frac{1}{2}\Delta E^t \Delta E = \frac{1}{2} z^t\, K\, z \quad \text{where } K = S^t S.$$

$$(P_\beta) \qquad \begin{array}{l} \textit{Minimise } J(z) = \dfrac{1}{2} z^t\, K\, z \\ \textit{subject to} \quad c_1(e) \le Az \le c_2(e) \\ \qquad\qquad z \in b \end{array}$$

where $b = \left\{ z \in \Re^q : |z_i| \le d_i,\ d_i > 0 \quad i = 1, \ldots, q \right\}$ is the region where the linearization assumption is valid.