

Experimental Neural Networks for Prediction and Identification

Cesare Alippi and Vincenzo Piuri, *Member, IEEE*

Abstract—In this paper we prove the effectiveness of using simple NARX-type (Nonlinear Auto-Regressive model with eXogenous variables) recurrent neural networks to identify time series and nonlinear dynamical systems. Experimentally we show that, whenever the process generating the data is ruled by a linear model, the performances provided by the neural network are comparable with the ones given by the optimal predictor determined according to the Kolmogorov–Wiener theory. On the other hand, whenever the system to be modeled is intrinsically nonlinear, its performance approaches that obtainable with classical linear identification. The work extends that suggested by Narendra in [1] by considering a reduced set of training data and a black-box model for the system to be identified.

I. INTRODUCTION

THE identification problem is to infer relationships between past data and future ones of unknown time series and dynamical systems. In both cases the ultimate goal is to provide a time-dependent model approximating the behavior of the process generating the data.

In classical approaches the search for the optimal approximation model is carried out within a parameterized identification family (usually linear such as Moving Average “MA,” Auto-Regressive “AR,” and/or their combination “ARMA” for time series or with an eXogenous “X” variable e.g., ARX or ARMAX for dynamical processes) and it is chosen to optimize a given figure of merit [2] (e.g., a mean-squared error function “MSE”).

Because of its simplicity, a linear model does not always suffice to approximate adequately a generic nonlinear system in its whole working environment. Therefore, to improve approximation accuracy, various solutions have been envisaged which generally encompass process linearization around a set of working points [2], [3] and expansions in series of functionals (e.g., Volterra’s expansion) [3]: the bottleneck now becomes the computational load. Obviously, difficulties increase when the model ruling the process is unknown; in this case, the system needs to be treated as a black-box model.

For this purpose neural networks, because of their intrinsic nonlinearity and computational simplicity, are natural candidates to approximate a given model. In fact, the nonlinear parametric family obtainable with neural structures extends

the linear one by providing nonlinear models; among them are the NAR and NARX subfamilies.

In particular, neural networks of the regression type (input-hidden-output feedforward topologies with sigmoidal activation functions in hidden units and a single output neuron characterized by a linear activation mapper) have been proved to be universal approximators [4], [5], in the sense that they are able to approximate any real function under *mild* assumptions (and therefore the unknown behavior in a time series with an arbitrary precision). The problem of predicting a future element in time series is therefore well posed. Theoretical support for the identification problem are given, on the other hand, by the Stone and the Stone–Weierstrasse approximating theorems [1], while in [6] it is shown that simple NARX neural networks are equivalent to more complex fully connected recurrent networks [7]. Thus, using results about the universality and strong computational power of fully connected recurrent nets [8], we conclude that NARX networks are universal dynamical systems.

The structure of the paper is as follows. In Section II, we provide tools to optimally determine an NAR-type neural predictor. It is shown that, whenever the training set provides sufficient excitation (i.e., it contains enough information to infer the law ruling the time series) and in correspondence with MA, AR, and ARMA models, the neural network provides performance comparable with the optimal ones obtainable with the Kolmogorov–Wiener theory. In fact, we experimentally prove that the generalization error (namely the estimation of the variance of the residual predicting error) tends to the optimal one, thus showing the validity of prediction by neural learning. The analysis is then extended in Section III to cover the identification case where the envisaged systems are ruled by nonlinear dynamics. Limits posed by the Narendra approach, namely requirements for long sequences of data (an unfeasible requirement in most industrial processes) and a priori knowledge of the nonlinear placements within the model (which requires not always available knowledge of the process), have been solved by developing an appropriate procedure for constructing the training set and considering the system as a black box.

Finally, the proposed methodologies have been validated in Section IV to identify a real-world problem: the input/output relationship of a brushless motor used to control the joint of an industrial robot. The input is the current of the motor the output is its angular velocity (and, consequently, the joint position). We discover that neural networks provide better identification

Manuscript received April 24, 1995; revised October 30, 1995.

The authors are with the Dipartimento di Elettronica ed Informazione, 20133 Milano, Italy.

Publisher Item Identifier S 0018-9456(96)02497-7.

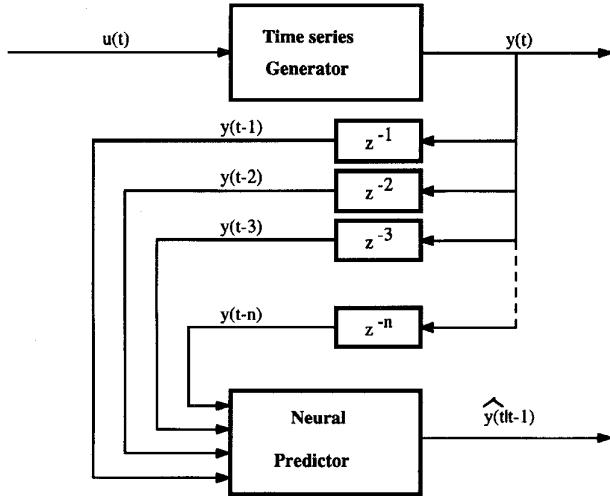


Fig. 1. A structure for the one-step neural predictor.

performance and a simpler parameter estimation procedure when compared to the optimal linear ARX model.

II. PREDICTION OF SYSTEMS GOVERNED BY LINEAR MODELS

The goal of this section is to verify that, under the assumption that the process generating the data is linear (and therefore, by knowing the model by which the Kolmogorov–Wiener theory provides the optimal predictor), a properly designed neural predictor provides performance comparable with the optimal. In Section II-A we will initially focus the attention on how to construct, from data, an optimal neural predictor, leaving to subsequent Sections II-B–D performance validation and comparison.

A. Constructing an Optimal Neural Predictor

Let

$$y_1, y_2, y_3, \dots, y_N \quad (1)$$

be the given time series, affected by noise and generated by an unknown process.

We will consider the case where the quantity of available data for (1) is large enough to allow partitioning the data set into a training set (the first set of N_D data is used to train the network) and a validation set (the model selection is based on cross-validation). If this is not the case, cross-validation techniques cannot be successfully taken into account, and different criteria should be envisaged (e.g., FPE, AIC, and RIS criteria [9]).

The structure for a one-step neural predictor is now given in Fig. 1. The neural predictor receives a sliding window of past y 's and learns how to infer the next one. The extension to multi step prediction is now straightforward. Say that we want to predict the k th step, then the neural predictor simply receives the subset of past y 's till time t and infers the $y(t+k)$ value; therefore the whole procedure coincides with that of the one-step prediction. It should be noted that the correct dimension $N_I = n$ of the sliding window (and therefore the

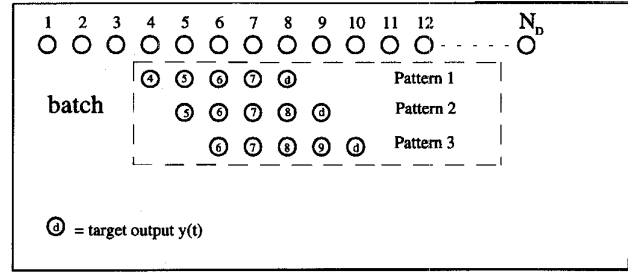


Fig. 2. Extraction of a generic training batch.

input complexity of the neural predictor) is unknown, as well as the number N_H of hidden units; such dimensions need to be determined to optimize performances.

To construct an optimal neural approximating model we have to determine an appropriate training set containing all the relevant information of the process and define a suitable topology matching complexity and performance. Let us tackle first the training-set construction issue which requires introduction of four entities:

- the number of training data N_D to be considered;
- the number of patterns N_P constituting a batch (the set of patterns constituting a full training sequence);
- the number of batches N_B to be extracted from the training set;
- the number of times N_L a generic batch is presented to the network during learning.

Let us clarify the procedure. During the training-set construction, an index k is uniformly selected from the $[1, N_D - N_I - N_P + 2]$ interval and defines the starting point for the batch. Patterns are now extracted to create N_P learning schemes of (*input*, *output*) type, where *input* addresses the past y 's window and *output* the target output [the next value provided by the time series in a one-step prediction or the $y(t+k)$ in the k -step prediction case].

Such learning schemes are now presented to the network one by one, and weights are updated as required by the chosen learning algorithm. The training procedure now operates N_L times on the given batch. An example of the batch extraction procedure is shown in Fig. 2 for the case $k = 4$, $N_I = 4$, $N_P = 3$ where the generic $y(i)$ th datum is represented with the i th circle and the target desired value with the d label.

It is interesting to note that by extracting learning schemes according to the proposed procedure, a reduced set of data is sufficient for training. The assumption on which the procedure is based is that the training set contains all the relevant information necessary to learn the application. The requirement coincides with the hypothesis that the training data have been generated by full excitation of input signals such as a white noise which, when considered as an input signal, is able to excite all the process dynamics (thus providing nontrivial outputs). In the time series case we will consider MA, AR, ARMA models as processes generating the data: the full excitation signal now becomes the white noise present in the model. In the identification case associated with (1) there is a

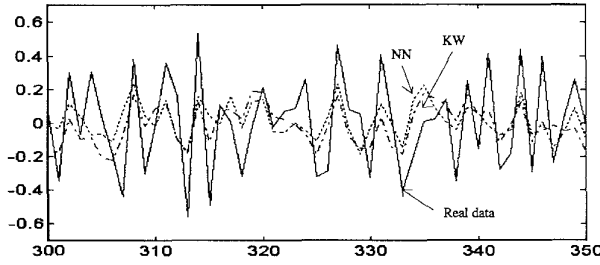


Fig. 3. The one-step predictors for the MA(2) process.

sequence of the input signal $u(t) = u_t$

$$u_1, u_2, u_3, \dots, u_N$$

which has generated (1) itself.

The generic training couple (*input, output*), now contains as *input* not only a past y 's window till time $t-1$ but also a past u 's window till time t while the desired output becomes y_t . Here, the full excitation signal to be used (just for generating training and validation data) would preferably be white noise $u(t) = WN(0, \lambda^2)$.

We should observe that whenever a white noise signal cannot be used as the input signal (let us think of an industrial plant), the signals to be taken into account should be impulses, steps and ramps. In addition, if a model for a plant is available (even if very simple), its inputs can be excited with a white noise signal while the models' output can be integrated with the plant's real data for training.

In subsequent experiments, we will consider a set of $N_D = 100$ data while at least 500 data will constitute the validation set (the variance λ^2 of the noise is set to be one). Training is based on the classical back-propagation algorithm choosing $N_B = 1000$, $N_P = 1$, $N_L = 1$.

A second relevant aspect is related to the determination of an optimal network which requires matching of the information content present in the application with the topological complexity of the neural structure. This can be done by applying an optimization procedure, for instance a spectral decomposition technique [10]. This method requires an over-dimensioned network to be successfully trained and locally optimizes each layer by first reducing the number of input neurons, and then the number of hidden units. The methodology relies on the information content flowing through the layer to be optimized. In particular, the most relevant singular values associated with the output of the layer coincide with the number of units to be considered for it.

Now compare the performance provided by a neural predictor for the case where the processes generating the data are MA, AR, and ARMA models, respectively.

B. The MA Case

The chosen process generating the data is ruled by the MA model

$$y(t) = e(t) - \frac{1}{2}e(t-1) - \frac{1}{4}e(t-2) \quad (2)$$

characterized by a white noise with zero mean and λ^2 variance [$e(t) = (0, \lambda^2)$].

TABLE I
PERFORMANCE COMPARISON FOR THE MA(2) PROCESS

MA(2)	Kolm-Wiener	NN 2-6-1	Theory
Validation error	0.99	1.07	1

TABLE II
PERFORMANCE COMPARISON FOR THE AR(1) PROCESS

AR(1)	Kolm-Wiener	NN 1-6-1	Theory
Validation error	1.03	1.06	1

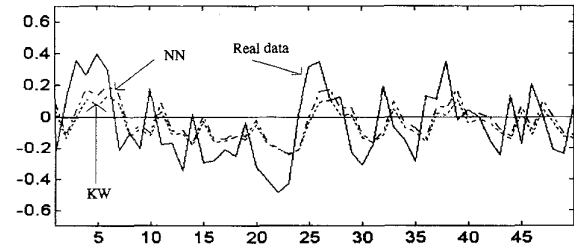


Fig. 4. The one-step predictors for the AR(1) process.

By applying the Kolmogorov-Wiener theory the optimal one-step predictor becomes

$$\hat{y}[(t+1)|t] = \frac{1}{2}\hat{y}[t|(t-1)] + \frac{1}{4}\hat{y}[(t-1)|(t-2)] - \frac{1}{2}y(t) - \frac{1}{4}y(t-1) \quad (3)$$

while the associated one-step prediction error coincides with the variance of the white noise:

$$\text{Var}\{y(t) - \hat{y}[t|(t-1)]\} = \lambda^2. \quad (4)$$

Let us now focus attention on the neural predictor. We chose a $2 \times 6 \times 1$ neural topology with the two inputs receiving the $y(t-1)$ and $y(t-2)$ past values.

The variance estimations of the prediction error for the Kolmogorov-Wiener (KW) and the neural (NN) predictors evaluated on the validation set are presented in Table I with the optimal prediction error set by the theory (1 since $\lambda^2 = 1$). The time series, the optimal KW predictor and the NN predictor performances are presented in Fig. 3 where the worst prediction interval is shown.

Performances provided in Table I and Fig. 3 show how the neural predictor was able to learn the behavior of the optimal predictor.

C. The AR Case

Let us now consider a system ruled by the AR(1) model

$$y(t) = \frac{1}{2}y(t-1) + e(t) \quad (5)$$

where the optimal predictor can be simply obtained by removing the noise contribution from (5):

$$\hat{y}[(t+1)|t] = \frac{1}{2}y(t-1). \quad (6)$$

As we did for the MA model, performances are given in Table II and depicted in Fig. 4 where the worst predicting interval is given.

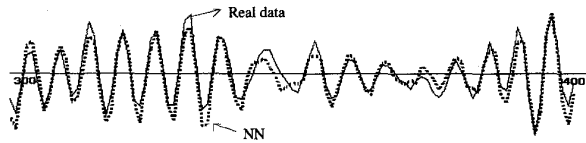


Fig. 5. Performance provided by the neural network in the worst interval.

Once more, the neural predictor, without knowing the model equations but simply exploiting data, is able to provide performance very close to the optimal.

D. The ARMA Case

Finally, let us consider a more complex case where the time series is ruled by the linear ARMA model (the model has been suggested in [11])

$$\begin{aligned} y(t) = & 0.95 \cdot y(t-1) - 0.99 \cdot y(t-2) \\ & + e(t) + 0.3 \cdot e(t-1) + 0.4 \\ & \cdot e(t-2) + 0.5 \cdot e(t-3) \end{aligned} \quad (7)$$

with $e(t)$ a white Gaussian noise with zero mean and λ^2 variance.

The optimal one-step predictor, according to the Kolmogorov-Wiener theory reduces to

$$\begin{aligned} \hat{y}[(t+1)|t] = & -0.3 \cdot \hat{y}[t|(t-1)] - 0.4 \cdot \hat{y}[(t-1)|(t-2)] \\ & - 0.5 \cdot \hat{y}[(t-2)|(t-3)] + 1.25 \cdot y(t) \\ & - 0.59 \cdot y(t-1) + 0.5 \cdot y(t-2). \end{aligned} \quad (8)$$

In this more complete example let us focus the attention on the whole procedure leading to the optimal neural predictor.

A set of $N_D = 100$ data was considered for the training phase, while the 700 consequent data created the validation set (the variance of the white noise was set to one: $\lambda^2 = 1$). Training was based on the classical back-propagation algorithm with the choice $N_B = 50$, $N_P = 5$, $N_L = 10$.

The spectral decomposition approach was then applied to an over-dimensioned $6 \times 3 \times 1$ network and suggested an optimal topology with two inputs (this is reasonable if we note that the AR component of the ARMA model has a time dependency of two). After optimization of the input layer the network was retrained and the topological minimization procedure reapplied, now to the hidden layer. The optimal number of hidden neurons was determined to be two. Cross-validation proved that such a topology was the best one (in the generalization ability sense).

Comparisons between the optimal KW predictor and the best neural one in the most critical interval are given in Figs. 5 and 6. In particular Fig. 5 presents the NN's performance while Fig. 6 the performance provided by the KW's predictor. It can be seen that the neural prediction almost coincides with the optimal one. To quantify the validity of the approach, we measured the mean-squared errors MSE_{tr} and MSE_{val} over the training and the validation sets. Results are given in Table III.

It can be seen that the neural network with structure 6-3-1 is not optimal since the validation error (i.e., the correct estimation of the prediction error variance) is far from the opti-

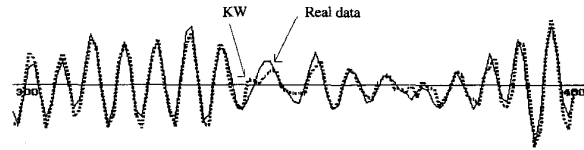


Fig. 6. Performance provided by the one-step Kolmogorov-Wiener predictor on the same critical interval.

TABLE III

COMPARISON BETWEEN DIFFERENT NEURAL PREDICTORS AND THE OPTIMAL ONE

ARMA	Training set (100 data)	Validation set (700 data)
Kolm.-Wiener	$MSE_{tr}=1.4$	$MSE_{val}=0.88$
NN 6-3-1	$MSE_{tr}=2.46$	$MSE_{val}=2.52$
NN 2-2-1	$MSE_{tr}=0.79$	$MSE_{val}=1.02$

mum (i.e., 1). On the other hand, the 2-2-1 topology provides performance comparable with the optimal one, thus proving the validity of the approach. Several experiments were carried out by considering different MA, AR, and ARMA models for the process generating the data. In all cases, whenever the network topology is optimized, the neural predictor provides performance comparable with the Kolmogorov-Wiener optimal predictor.

III. IDENTIFICATION OF DYNAMIC SYSTEMS

Identification of dynamic systems with neural networks has been suggested for the first time in [1] where, under the stationarity hypothesis for the process generating the data, it is shown how NARX neural networks are able to solve the problem.

The proposed methodology presents two fundamental limits which prevent usage of neural identification techniques for most real applications:

- a partial knowledge is required about the presence and placement of nonlinearities within the process (on inputs, on the AR component, etc.) and
- a large number of data (in [1] at least 10^5 data) are requested to accomplish training.

Both requirements are unfeasible in many real applications since the model of the process is generally unknown (and therefore the system must be treated as a black-box model) and a limited number of data are available (there is a cost to extract and measure process variables). Such limits can be solved by applying the training methodology we will propose in this section.

Let us consider first the extension in which the process is treated as a black-box model. A natural scheme for the neural identifier, which extends the linear ARX model family, is given in Fig. 7. The output of the identifier depends on past inputs and past outputs and can be expressed as

$$y(t) = f[u(t), \dots, u(t-m), y(t-1), \dots, y(t-n)].$$

Once more, as in the neural predictor, m and n are unknown and need to be finely tuned to optimize performances.

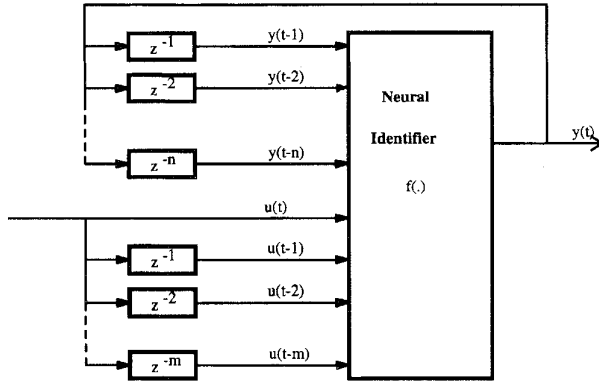


Fig. 7. The working scheme of a NARX-type neural identifier.

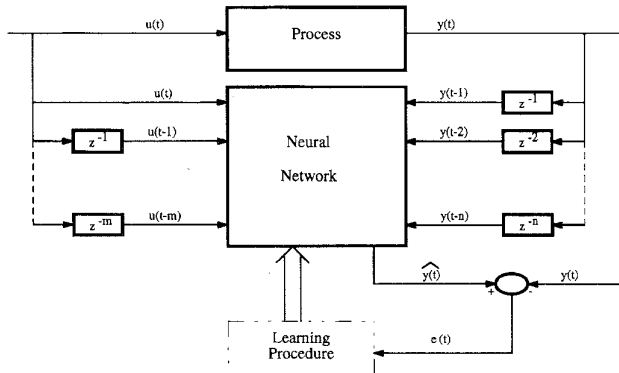


Fig. 8. The first training configuration.

There are two different training approaches to estimate the weights of the neural network. In the first approach an identifier is seen as a predictor able to predict infinite steps ahead or, equivalently, it is assumed that the performance of the one-step prediction is optimal with respect to all the possible system dynamics. During learning the links between the real process and the identifier are as depicted in Fig. 8, and the training modality coincides with training a one-step neural predictor (we assume in this case a classical training procedure such as back-propagation). During the recall phase the process and the identifier are completely disconnected; the one-step predictor is supposed to be optimal, and its output is fed back to create the configuration presented in Fig. 7.

Unfortunately, the first training procedure may suffer from unstableness. In fact, by considering the past predicted output as input, we introduce a discrepancy (caused by accurate but not perfect prediction, presence of noise, round-off errors, etc.) which may lead to poor performance (in some cases we also experienced unstable behavior). To overcome such a problem we considered a second training configuration as given in Fig. 9 (once training has been perfected the recall configuration remains that of Fig. 7).

We should note that the two training configurations require two different training algorithms. During training, in fact, the first neural model (configuration of Fig. 8) realizes a function of the type

$$\hat{y}(t) = f[u(t), \dots, u(t-m), y(t-1), \dots, y(t-n)] \quad (9)$$

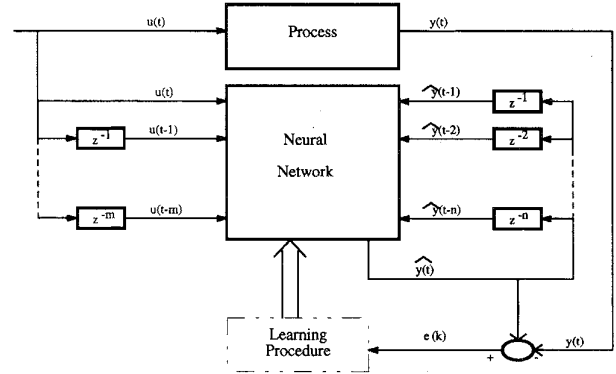


Fig. 9. The second training configuration.

which can be trained by applying a standard back-propagation algorithm, while in the second case the model is that of a recurrent network (configuration of Fig. 9)

$$\hat{y}(t) = f[u(t), \dots, u(t-m), \hat{y}(t-1), \dots, \hat{y}(t-n)] \quad (10)$$

and needs a recurrent training algorithm such as the one given in [7]. Despite the fact that the first approach may provide good performance, we prefer the second one to avoid unstable behavior during recall.

The second problem posed by the approach presented in [1], namely the requirement of a large number of data, can be solved by utilizing the same training scheme extracting procedure suggested in Section II-A for the neural predictor. In particular, experiments show that the number of patterns per batch N_B should be one as well as the number of training cycles per batch N_L . In this way it gives the same relevance to all training schemes.

A main result is that, by applying the first training procedure (to be consistent with results presented in [1]), a simple NARX network with a single hidden layer (and a smaller number of neurons) suffices to solve their benchmarks (conversely, a neural structure containing two hidden layers is required).

Here, we propose a new complex model to be identified (see also [12]) whose model can be described as

$$\begin{aligned} y(t) = & y(t-1) + [y^2(t-2) \cdot u(t-1)]^2 \\ & + [y^2(t-3) \cdot u(t-2)]^2 + u(t) / \\ & [1 + y^2(t-3) \cdot y(t-2) \\ & + y(t-3) \cdot y^2(t-2)]. \end{aligned} \quad (11)$$

The optimal network topology was determined by using the spectral decomposition approach and led to $m = 3$, $n = 2$ for a total of 6 inputs. This is in complete agreement with (11) where the temporal window dependency is the same. The final network structure was a 6-5-1 topology, while training was characterized by $u(t) = WN(0, 1)$, $N_D = 100$, and $N_B = 80\,000$.

We already tackled the white noise input issue by stating that its presence is fundamental for being able to influence all the process dynamics, thus generating significant and relevant training data.

In this context, a reduced number of training data (here just 100 data and not thousands as requested by Narendra's

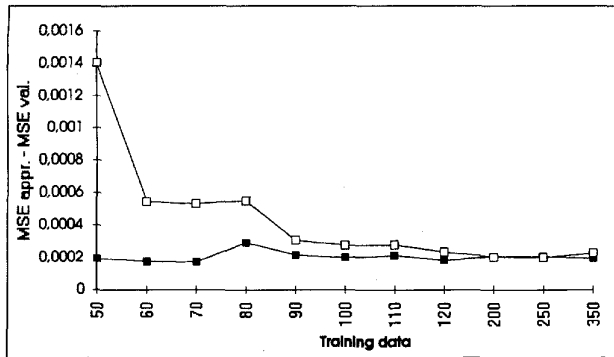


Fig. 10. The MSE training and the MSE validation as a function of the training set size.



Fig. 11. Validation performance in correspondence with $u(t) = \text{WN}(0, 1)$.



Fig. 12. Validation performance in correspondence to the input given in (12).

methodology) sufficed to successfully train the neural network. The number of training data was determined by measuring the validation error with respect to a different training set. The MSE during training and validation as a function of the number of training data is presented in Fig. 10. It can be seen that good performances are already obtainable by considering just 100 training data.

In Fig. 11 and 12 validation performances obtained by considering respectively the $u(t) = \text{WN}(0, 1)$ and

$$u(t) = 0.3 \cdot \sin\left(\frac{2 \cdot \pi \cdot k}{25}\right) + 0.4 \cdot \sin\left(\frac{2 \cdot \pi \cdot k}{32}\right) + 0.3 \cdot \sin\left(\frac{2 \cdot \pi \cdot k}{40}\right) \quad (12)$$

input signals are given.

In particular, the output of the real process is plotted with a continuous line and the output of the neural identifier with a dotted line. As we can see, a very simple network is able to identify the process behavior.

IV. IDENTIFICATION OF THE INPUT/OUTPUT RELATIONSHIP OF A BRUSHLESS MOTOR

The problem of identifying the model of a brushless motor represents an interesting application. The motor is used to control a joint of a six-joint robot arm (Comau-SMART 3S) able to handle an object at the end-effector. The control input for the motor is a current, while the output is its angular velocity which allows positioning of the joint.

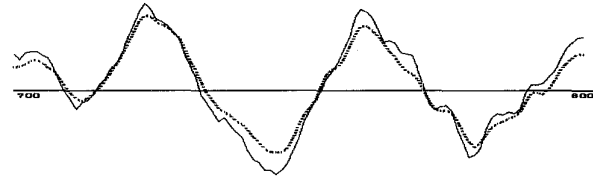


Fig. 13. The ARX(3, 3) model's performance on the validation set.

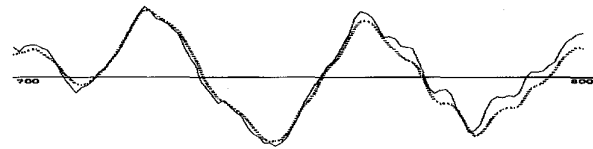


Fig. 14. The neural network performance on the validation set.

The process is intrinsically nonlinear, and nonlinearities are mainly concentrated on:

- Coulomb friction and
- hysteresis of speed reducers.

Nonlinearities need to be taken into account for a fine control; this would require an adequate and accurate model for the whole process which unfortunately is either unknown or too complex. The goal is then to identify the input-output relationship by considering the motor as a black-box model and applying the methodology presented in the previous paragraph. During data measurement all the joints are kept fixed apart from the one to be identified. A set of 1000 input/output schemes were measured by considering different loads for the gripper. We obtained good identification, and we will focus the attention on the 6th joint for which classical identification techniques have been applied.

In particular, such techniques led to an ARX(3, 3) model. Neural identification was also carried out by first considering an over-dimensioned recurrent network characterized by $N_D = 300$ and $N_B = 30000$ and then reducing the topological complexity by means of the spectral decomposition approach. The final optimal neural network structure was 18-10-1 (12 past outputs, 5 past inputs, and the actual input).

Performances on the validation set are now plotted in Fig. 13 for the ARX(3, 3) case and Fig. 14 for the neural model. Once more, to quantify the improvement in performance we computed the global mean-squared error in both cases which was 1.42 for the neural network optimal structure and 2.12 for the optimal ARX model.

Results prove how the neural model was able to better identify the model. The residual error was then computed, and it is plotted in Fig. 15 for the ARX model and in Fig. 16 for the neural identifier. It can be easily seen that the ARX identifier is biased (and therefore the model, being an optimal model, is not able to catch nonlinearities), while the residual error associated with the neural identifier is closer to a white noise.

V. CONCLUSIONS

In this paper, we have proved how neural networks of the NARX type can be successfully used to identify time

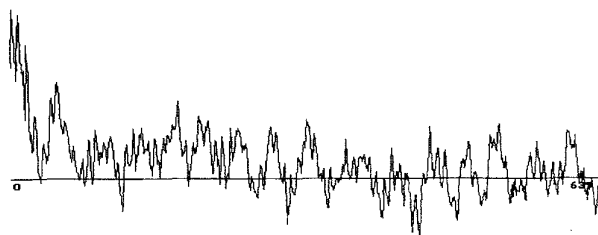


Fig. 15. Residual error for the ARX(3, 3) model.

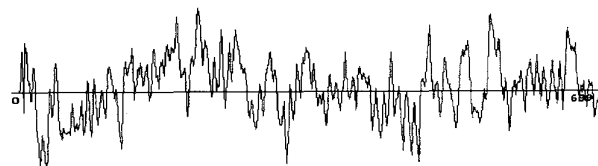


Fig. 16. Residual error for the neural identifier.

series and dynamic systems. In particular, whenever the time series is ruled by a linear approach, the neural network provides performances comparable with the ones obtainable with the optimal predictor set by the Kolmogorov-Wiener theory. Furthermore, whenever the world is nonlinear, linear models show both inefficiency and inaccuracy. Identification by neural networks therefore becomes an interesting and valuable tool to simply identify the process behavior. This has been shown in a real application: the identification of the input-output relationship of a brushless motor.

REFERENCES

- [1] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, no. 1, 1990.
- [2] L. Ljung, *System Identification: Theory for the User*. Englewood Cliffs, NJ: Prentice Hall, 1987.
- [3] S. A. Billings, "An overview of nonlinear systems identification," in *Proc. IFAC Ident. and Syst. Parameter Est.*, York, UK, 1985.
- [4] G. Cybenko, "Approximation by superposition of sigmoidal function," *Math Control System Signals*, 1989.
- [5] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, 1989.
- [6] H. T. Siegelmann, B. G. Horne, and C. L. Giles, "Computational capabilities of NARX networks," in *Proc. Machines that Learn*, Snowbird, UT, Apr. 4-7, 1995.
- [7] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," in *Neural Computation*, vol. 1, 1989.
- [8] H. T. Siegelmann and E. D. Sontag, "On the computational power of neural networks," *J. Computer and System Science*, to be published.
- [9] C. Alippi, "Extending the FPE and the effective number of parameters to biased estimators: How to select and validate optimal neural topologies," *Internal Report of the Department of Brain and Cognitive Sciences*, MIT, Cambridge, June 1995.
- [10] C. Alippi and V. Piuri, "Topological minimization of multi-layered feed-forward neural networks by spectral decomposition" in *Proc. Int. Joint Conf. Neural Networks IEEE-IJCNN92*, Beijing, China, Nov. 3-6, 1992.
- [11] M. Marseguerra, S. Minoggio, A. Rossi, and E. Zio, "Neural network predictions and fault diagnosis applied to stationary and non stationary ARMA modeled time series," *Progress on Nuclear Energy*, 1992.
- [12] F. Previdi and M. Serratori, "Identificazione e Predizione di sistemi dinamici usando reti neurali," M.Sc. thesis, Dip. Elettronica, Politecnico di Milano, 1993.



Cesare Alippi received the degree (summa cum laude) in electronic engineering in 1990 and the Ph.D. degree in 1995 from the Politecnico di Milano, Milano, Italy.

He was honorary researcher and research fellow at the department of Computer Science, University College, London, UK, from 1991 to 1993. His research interests include neural networks (mathematics, learning, and applications), genetic algorithms and fault tolerance.



Vincenzo Piuri (S'84-M'86) received the Dr. Ing. degree in electronic engineering in 1984 and the Ph.D. degree in information engineering in 1989, from Politecnico di Milano, Milano, Italy.

He is Associate Professor in Operating Systems at the Politecnico di Milano. His research interests include distributed and parallel computing systems, computer arithmetic, neural networks, and fault tolerance.

Dr. Piuri is a member of AEI, IMACS, and INNS, and is Secretary of the IEEE Instrumentation and Measurement Society, North Italy Chapter.