Neural Methodology for Prediction and Identification of Non-Linear Dynamic Systems

Cesare Alippi
CNR C.S.I.S.E.I.
c/o Politecnico di Milano
Piazza L. da Vinci 32, 20133 Milano, Italy
Email: alippi@elet.polimi.it

Vincenzo Piuri
Department of Electronics and Information
Politecnico di Milano
Piazza L. da Vinci 32, 20133 Milano, Italy
Email: piuri@elet.polimi.it

Abstract

A structured design methodology is introduced to support and guide the designer in using neural techniques to identify and predict complex dynamic non-linear systems.

1: Introduction

Prediction and identification of a dynamic system with a black-box approach is necessary whenever the equations ruling the system are unknown or they are computationally untractable. In fact, in many real cases, the computational complexity may prevent an effective use for several applications (e.g., real time monitoring and adaptive/predictive control).

Traditional black-box techniques (such as MA, AR, ARMA, ARX, ARMAX [1]) can be applied whenever the system is linear or can be linearized with an acceptable accuracy around a working point. Highly non-linear systems, as well as systems characterized by a smooth nonlinearity but operating in a wide range, may not be easily represented by a single linear model, thus requiring decomposition in several working points and a model for each of them.

Neural techniques are an alternative approach due to their intrinsic non-linearity, adaptability, noise immunity, generalization ability and robustness. Several successful experiments and general ideas on prediction and identification have been reported in the literature (e.g., [2, 4]), but a general framework is not yet available to guide the designer towards the construction of at least a nearly optimal solution with a reasonable confidence. This paper presents a comprehensive methodology for design neural networks in identification and prediction applications of non-linear dynamic systems.

2: A Methodology for Creating Neural Predictors and Identifiers

A general methodology for neural modeling must be able to deal with several aspects of a complex dynamic non-linear system, ranging from the internal structure to the interactions among components, from external constraints to safety issues, from stability to intrinsic control loops. As a consequence, it must effectively deal with the following basic problems which often occur in real applications:

- modeling a large complex system. This task may be difficult if the system is considered as a whole, due to the computational complexity and the interfering objectives of the optimization functions. The system needs therefore to be decomposed into simpler subsystems.
- large non-linearities and several working points. When several working points (and, as consequence, models) need to be taken into account, the neural approach should limit their number with respect to the traditional techniques by exploiting the intrinsic non-linearities and generalization abilities. Local modeling around the working points or global modeling in the whole operating range must be captured in the neural descriptions: the first case allows for an accurate modeling but it requires separate neural models, while the second approach generates a single model having a possible lower accuracy.
- local control loops cannot sometimes be opened. In fact, they may be safety loops, or opening them may increase the training complexity, or may require a detailed knowledge of physical phenomena. A basic behavior can be learnt by training the network in presence of the control loops, while the model can be then refined by opening (at least virtually) the control loops.

By taking into account such problems, the overall design methodology is summarized in the following phases:

- *system partitioning*,
- *selection of working ranges and points*,
- *modeling in the small with closed control loops* for each operating range and point,
- *modeling in the small with virtually-open control loops* for each range and point,
- *modeling in the large with closed control loops* to merge the local behaviors,
- *modeling in the large with virtually-open control loops* to merge the local behaviors.

Obviously, some of the above steps may be avoided when the application requirements and the system characteristics do not need them or when the model achieves satisfactory performances with respect to the application.

Partitioning of the complex system into separate subsystems communicating through a limited number of system variables allows to limit the complexity of the modeling procedure. Training a large network capable of capturing the behavior of the whole system requires a longer and more accurate procedure to avoid behavior. Concurrent design by groups working in parallel thus becomes feasible.

As a result, the system is decomposed in SISO or MISO subsystems (with possible reconvergent paths) since they are easily described by observing and measuring input or internal system variables, while either one system output or one internal system measurable variable is generated. The partitioning phase is guided by: subsystem simplicity, functional decoupling, interconnection clusters, observability and measurability of variables and controllability of subsystems inputs. A good approach starts from physical and topological system analyses to identify the components naturally composing the system.

Modeling in the small captures the non-linearity of the system behavior around a specific working point. Selection of these points and their neighborhoods is performed by analyzing the physical features and the typical operating conditions. Localization of the relevant non-linearities and partitioning into operating ranges is strictly related to the specific application case, but can be performed easily from a rough physical system description or a preliminary input-output experimental data analysis.
The neural model associated to a working point is obtained by applying the following steps:

- generating the input stimuli and the expected outputs in the given working range,
- selecting the sampling period,
- selecting the samples for learning, testing, and validation,
- normalizing the samples,
- selecting the neural model,
- selecting the optimization figure of merit,
- selecting the training procedure,
- configuring the neural model,
- possible model optimization,
- validating the neural model.

A detailed description of the above phases is given in the subsequent sections. The resulting neural model is valid only in the given range around the considered working point. As a consequence, several neural models must be created to cover the whole operating range of the system, as in the traditional approach; however, in neural solutions, it is possible to exploit their intrinsic non-linearities and their generalization ability so as to reduce the number of working points necessary to achieve a good approximation of the system behavior. Overlapping the local behaviors at the boundaries of adjacent ranges is implicitly achieved even if merging is not complete and homogeneous. It should be noted that such neural models are structurally similar differing basically on the number of neurons and the value assumed by interconnection weights.
Modeling in the small is particularly suited to achieve a high accuracy around each working point; this solution may be not acceptable when a single model is strictly required by the application.

Modeling with closed control loops considers the feed-back loops included in the system under modeling. The control loops limit the process behavior by correlating its inputs. As a consequence, the training procedure usually converges quickly to a good solution. Such a model may be acceptable in some applications, e.g., when the control loops cannot be opened or when the information loss is not critical for model accuracy.

Modeling with open control loops allows to capture a more detailed knowledge about the process. In some applications the control loops can be opened, the inputs can be directly accessed and the associated outputs observed; preliminary modeling with closed loops can thus be avoided, while the additional knowledge may be exploited for possible further accuracy. In other cases, the control loops cannot be opened but a more detailed knowledge about the subsystem behavior is desirable for accuracy and generalization ability. This can be obtained by uncorrelating (at least partially) inputs and fed-back signals by adding small perturbations to the subsystem feedback signals so that the subsystem is *modeled with virtually-open control loops.* A priori, modeling with closed loops should be applied; if results are not satisfactory the virtually-open loops modeling is considered.

Modeling in the large is directed to create a single model covering the whole operating range of the system, even if a loss in accuracy can be achieved with respect to the set of local models. This approach captures the overall behavior by spanning through all individual working local ranges.

The configuration procedure consists of the same steps performed for modeling in the small but sample selection is performed on the whole inputs' range. Also in modeling in the large, the closed loop phase as well as the open (or virtually-open) phase must be considered by adopting the same approach discussed for modeling in the small.

3: Creating the Data Sets for Training, Testing and Validation

In the data extraction phase the inputs of the system to be modeled are excited and the associated outputs measured. Data must be sufficiently informative to characterize the system behavior and are subdivided in three sets for modeling purposes:

- the *learning set* is used to training the neural model,
- the *testing set* is necessary to monitor the evolution over training time of the model performance and provides a criterion for the termination of learning,
- the *validation set,* at the end of learning, estimates the performances of the model.

Testing and validation sets must be independent from the training one to guarantee a significant evaluation of the model thus avoiding biasedness over the training set.

The first step to create the above data sets consists of creating adequate *excitation* signals for each system input. These signals must persistently exciting (i.e. they must be able to fully excite all the system dynamics). Ideally, a white random noise signal should be consider to cover the whole significant input range. Unfortunately, this may not be a feasible signal in many real cases due to operating, physical, safety and economical constraints; a good alternative is a random signal to be composed of a combination of trains of steps and ramps with random amplitudes, periods and stepnesses (e.g., a Pseudo Random Binary Signal), covering the whole range of the input values. In MISO systems, this signal must be generated independently for each input to avoid possible correlations.

In modeling with closed loops, the fed-back inputs are not considered as external inputs and therefore no excitations need to be generated. In modeling with virtually-open loops, perturbations (white noise or random signals) are added either to the set points or to the controlled inputs.

The sampled data are obtained by *sampling* inputs and output signals according to a suited sampling period *T*. Identification of an optimal *T* is relevant for the model quality to minimize the variance of the weights estimation. The complexity of the equation-based system description does not allow in general any direct determination of *T*. As an empirical rule, an estimation can be obtained from spectral analysis of the input and the output signals. Each input signal is considered to be band limited: the sampling frequency is assumed to be the double of the maximal cut frequency among all inputs and outputs. A correction factor (typically, 5) is usually adopted for safety due to the difficulties in selecting the cut frequencies.

In some cases, the maximum absolute value of an input or output is one or more orders of magnitude greater than the others so that the learning task could be ill-conditioned. Data *normalization* is thus necessary to improve efficiency of the training algorithm and to limit the possible dominance of some inputs or outputs with respect to the others. Data normalization avoids this problem by suitably scaling the data to be given to the network. Good results are usually achieved by scaling inputs and outputs into the [-1,1] range.

4: Selecting the Neural Paradigm

Selection of the neural paradigm means definition both of the network topology and of the neurons' operation. Static systems are modeled by feed-forward multi-layered networks, but dynamics need more complex paradigms: memory elements must be included in the neural model to hold information about the past behavior of inputs and internal states. Several models have been proposed in the literature: in practice, the best approach is to consider simple model first and increasing their complexity only when the modeling accuracy is not satisfactory.

The dynamic system to modeled can be viewed as a black box in order to abstract from its internal characteristics. No knowledge is a priori required about its structure, even if any information about the system can be exploited to reduce the model complexity or simplify the parameter estimation. A generic system can be described by one of the following prediction model families:

$$Y_p(t) = \phi(U(t), U(t-1), U(t-2), \ldots, U(t-h), Y(t-1), Y(t-2), \ldots, Y(t-k))$$

$$Y_p(t) = \phi(U(t), U(t-1), U(t-2), \ldots, U(t-h), Y(t-1), Y(t-2), \ldots, Y(t-k), \\ Y_p(t-1), Y_p(t-2), \ldots, Y_p(t-k))$$

and by the fully recurrent family of models for the pure identification case:

$$Y_p(t) = \phi(U(t), U(t-1), U(t-2), \ldots, U(t-h), Y_p(t-1), Y_p(t-2), \ldots, Y_p(t-k))$$

being $U(t)$ the input vector at time t, $Y(t)$ the output vector, $Y_p(t)$ the predicted output vector, and ϕ a non linear function. The identification models are intrinsically recurrent for dynamic systems since the system state must be taken into account; prediction models are basically non recurrent, even if recurrent structures have shown better approximation ability in some applications [2].

The basic structure for *prediction* can be derived from the nature of the operation itself: foreseeing the system behavior after a given time interval from current and past inputs and from past actual outputs. This leads to a feed-forward multi-layered network, as in Fig. 1a; one or two hidden layers are enough for a good approximation in the NARX error model (i.e., the predicted output is a function of current and past inputs, actual past outputs and a white-noise additive signal) [2]. For SISO systems (generalization to MISO is straightforward), the network inputs are the primary inputs of the real dynamic system

and as many previous system outputs as required by the dynamics. The number of feedback inputs is derived from the real system, e.g., by using a priori information coming from the physical model. The network's output is the system forecast output. Hidden neurons have usually sigmoid transfer functions, while the output neuron is linear; a variation is the radial-basis network having a gaussian activation function in all neurons.
In some cases (e.g., NARMAX models), accuracy of the predicted output must be increased by including the past prediction errors, so that the neural structure is recurrent, as in Fig. 1b [2]. In some systems, the actual outputs are not easily observable so that past predicted outputs are fed back instead of the actual ones (Fig. 1c) [3]. Stability and accuracy problems occur since learning operates on data possibly affected by noise. This is suited when predicted outputs depend on current and past system states and inputs.
Feed-back loops at the level of the individual hidden layer have been considered, but no specific enhancement has been observed with such structures.

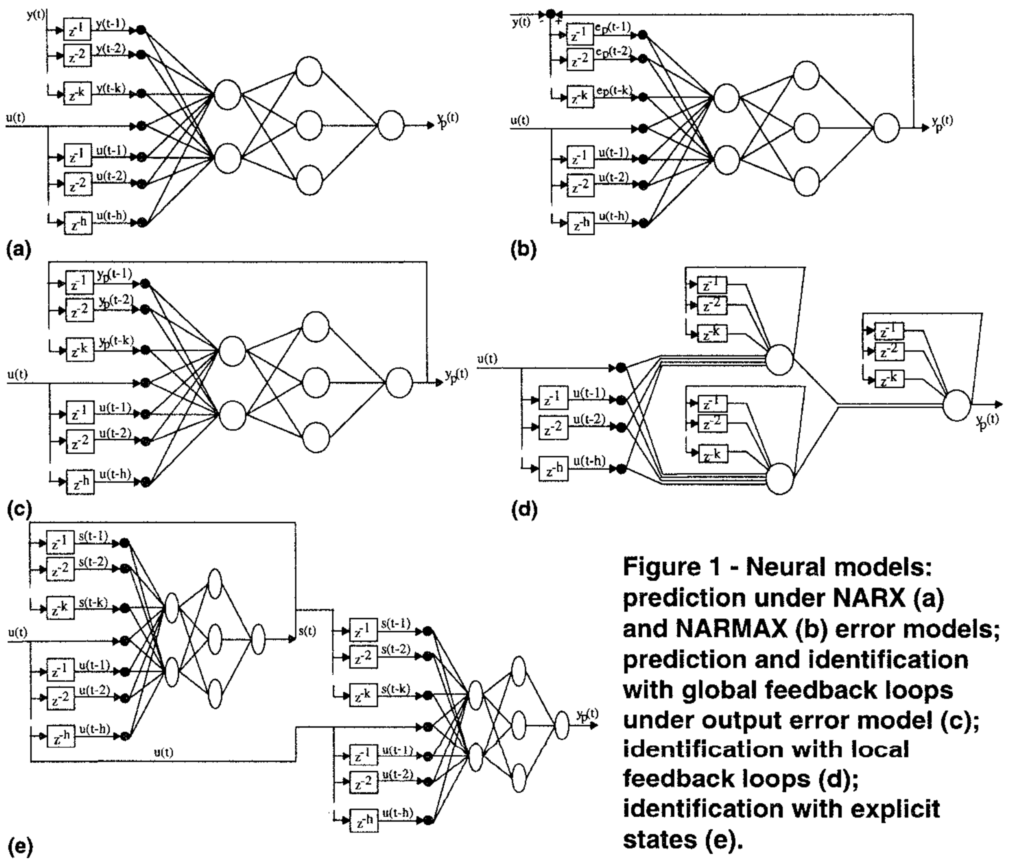

Figure 1 - Neural models: prediction under NARX (a) and NARMAX (b) error models; prediction and identification with global feedback loops under output error model (c); identification with local feedback loops (d); identification with explicit states (e).

A rough idea of the output error model best matching the system black-box behavior is a priori required to achieve the optimum approximation; generally it can be derived from a rough physical model. Otherwise, selection can be performed a posteriori by evaluating the prediction error and testing if it is a white noise; in this case, the error model is correct, conversely, another model should be considered.

Models for *identification* have to capture the whole system behavior without any connection to the actual system outputs, but only to inputs, since the neural model must be able to replace the actual system.
Neural networks with *global feed-back loops* (Fig. 1c) gave interesting results [3]. Inputs are constituted by current and past primary inputs and by current and past predicted outputs, while the neural network is defined as the one presented in the prediction case.
Another structure adopts feed-back loops *locally* within each neuron (Fig. 1d) [2]. These networks are quite difficult to be configured since often they have too many degrees of freedom and local states. This leads to a long training procedure.
Alternative topologies may use *distributed* memory elements at the level of each layer; additional interconnections may be considered among neurons belonging to the same layer, as well as unstructured network topologies that do not limits the computation propagation to a single direction. From preliminary experiments, these complex structures seem provide only additional models with limited influence in the practice.
System theory suggests a structure based on the *state-output* system model (Fig. 1e). The state variables are made explicit and the system model is partitioned in two blocks. The first block uses the past inputs and states to compute the current state; the second block approximates the output from the current state and inputs. Two feed-forward multi-layered networks are required: the network for the states is recurrent, while the one for the output is simply regressive. Even if suited to represent any dynamic system, this structure may be difficult to be created since the state may be not or not easily measurable.

The specific neural topology must be specialized by defining the number of layers, the numbers of neurons in each layer, and the interconnection weights. In general, no direct relationship is available to guide the first two choices. From personal experience or from the literature, the designer usually identifies a structure large enough to deal with the specific application and verifies the correctness of this assumption by experiments. Underdimensioning may not allow to capture the complete system behavior due to lack of degrees of freedom. Overdimensioning induces usually overfitting but the presence of the test mechanism keeps this effect under control. Once chosen significant training and testing sets, if the testing error presents a U-shape over the time, the model is overdimensioned with respect to the training set. Conversely, if the testing error is decreasing in the average over time, the model has been properly dimensioned.

5: Training and Validating the Neural Model

The *learning* procedure tailors the neural model to the specific application: the *learning* set is presented to the inputs, outputs are computed accordingly and compared to the expected values, while weights are adjusted to minimize the optimization figure.

The training scheme used to configure a neural *predictor* is shown in Fig. 2a for the model of Fig. 1a. Current and past inputs are given to the network with the past system outputs. Extension to the case of Fig. 1b can be obtained by feeding the past prediction errors into the network as additional inputs; the case of Fig. 1c can be dealt with by replacing the past system outputs with the past predicted outputs as in Fig. 2b. Stability is straightforward in the first case since it is a feed-forward structure, while the recurrent nature of the other two cases and the use of possibly erroneous outputs may induce problems, in particular for MISO and MIMO systems. Stability, whenever possible, is learnt during training, but it is usually paid as a prolonged application of the learning procedure.

The learning scheme for the neural *identifier* with global feed-back loops (Fig. 1a) can be derived from the corresponding one-step predictor, when the training procedure has an absolute error with the same magnitude of signal measurement. The expected output is quite identical to the actual output of the modeled system at every time and, as a consequence, it can replace the actual system output at any time (Fig. 1c). During training, the network topology is the prediction one (Fig. 1a) with the configuration scheme of Fig. 2a; after training, the network topology is shown in Fig. 1c. In general, this approach is quite fast due to the absence of system feed-backs, but it is effective for SISO systems and presents problems in the MISO case.

An alternative approach trains the network of Fig. 1a directly. Stability, whenever possible, is learnt during training, even if a prolonged learning may be necessary.

In the presence of local or distributed feed-back loops, the training procedure must deal with the recurrent model structure, directly. For the state-output model of Fig. 1e, the learning scheme separates the state network (which uses past states and inputs) from the expected output network (which operates on current inputs and state); each of these training is performed independently.

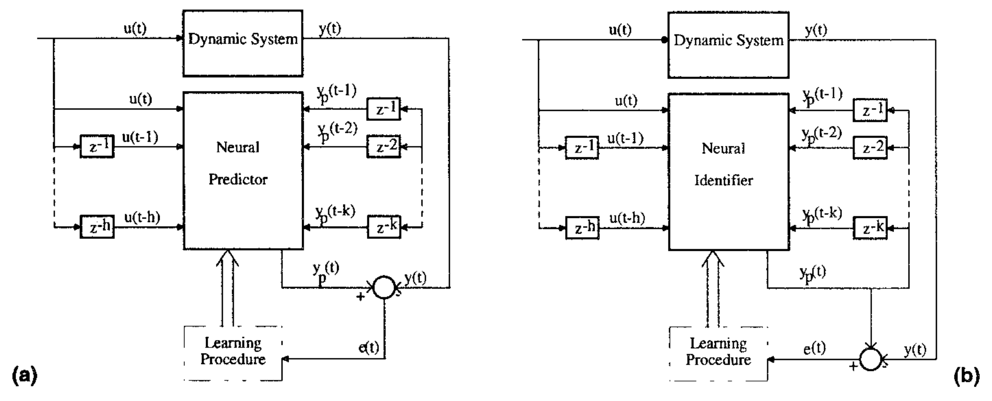

Figure 2 - Learning schemes for prediction (a) and identification (b).

Since dynamics have to be captured, data sets presented at each iteration of the learning procedure are not composed by pairs of input-output values, but by continuous sequences. Data in the learning set are therefore organized in subsets and batches. Each subset is related to a specific transient event in the system, while each batch is a group containing as many data as the network primary and state inputs. The number of batches in a subset is related to the meaningful length of the transient event, i.e., to the characteristic time constants of the system. Subsets must be cascaded without introducing discontinuities that do not exist in the real world. Subsets are presented several times in a random order to avoid possible network polarization on the last subsets and memory decay for the initial subsets. Within each subset, batches are sequentially presented several times to enforce a good understanding of the event by exciting the neuron polarization.

During learning, an optimization figure of merit is evaluated: the most used is the mean square error, evaluated on the whole outputs' trajectory or on an outputs' subset. Weight adjustment is performed by an external supervisor by using the descendent-gradient algorithm, the conjugate-gradient algorithm, the quasi-Newton approach, or other minimization techniques. Weights modification can be performed either at each sampled data or at the end of each batch of input-output. No specific rule is available to define a priori the number of repetitions of each subset presentation.
Learning termination can be identified by using the test data set: as soon as the test error continues to increase, the generalization ability begins to decrease due to overfitting and learning should stop. However, in the practice, the test procedure can be applied only periodically to avoid an unnecessary computational overhead; a compromise between quality and performance is achieved by applying the test phase at the end of groups of subsets containing a limited number of transient events (typically, few tens of times the number of events in the test set).

If the obtained network is overdimensioned with respect to the problem, optimization techniques can be envisaged. They reduce the topological complexity by removing unnecessary neurons or interconnections either by pruning at the end of the training phase or by modifying the figure of merit used during learning so that to penalize large topologies.

To evaluate the model quality, i.e., the generalization ability, validation is performed by applying the validation data set at the end of learning. The measure generated by validation is the same figure of merit used for learning.

6: Conclusions

A general methodology for practical application of neural prediction and identification of non-linear dynamic systems has been discussed; guidelines have been given as a general framework to overcome uncertainty and drawbacks of the traditional approaches to the use of neural networks. Due to the overall complexity and interrelationships among design choices, effectiveness of each choice need to be checked on the specific application: the method limits spectrum of alternatives and clarifies the design operations.

This methodology may be useful in design and implementation of advanced adaptable noise-tolerant control, monitoring and personnel training systems as well as in embedded systems with heterogeneous (neural and algorithmic) components: two of the most exciting challenges for industry and academia in the near future.

References

[1] L. Ljung, *System identification: theory for the user*, Prentice Hall, New Jersey, 1987

[2] *IEEE Trans. Neural Networks, Special Issue on Dynamic Recurrent Neural Networks*, vol.5, no.2, 1994

[3] K.S. Narendra, P. Parthasarathy, "Identification and control of dynamic systems using neural networks", *IEEE Trans. Neural Networks*, vol. 1, no. 1, 1990

[4] C. Alippi, V. Piuri, "Experimental Neural Networks for Prediction and Identification", *IEEE Trans. Instrumentation and Measurement*, vol. 45, April 1996