ISA TRANSACTIONS®

ISA Transactions 42 (2003) 1–0

Neural networks for process control and optimization: Two industrial applications

Gérard Bloch,[a],* Thierry Denoeux[b],†

[a]*Centre de Recherche en Automatique de Nancy (CRAN), UMR CNRS 7039, France and Ecole Supérieure des Sciences et Technologies de l'Ingénieur de Nancy (ESSTIN), Université Henri Poincaré, Nancy 1, France*

[b]*Heudiasyc, UMR CNRS 6599, France and Université de Technologie de Compiègne, France*

(Received 17 July 2001; accepted 16 April 2002)

Abstract

The two most widely used neural models, multilayer perceptron (MLP) and radial basis function network (RBFN), are presented in the framework of system identification and control. The main steps for building such nonlinear black box models are regressor choice, selection of internal architecture, and parameter estimation. The advantages of neural network models are summarized: universal approximation capabilities, flexibility, and parsimony. Two applications are described in steel industry and water treatment, respectively the control of alloying process in a hot dipped galvanizing line and the control of a coagulation process in a drinking water treatment plant. These examples highlight the interest of neural techniques, when complex nonlinear phenomena are involved, but the empirical knowledge of control operators can be learned. © 2003 ISA—The Instrumentation, Systems, and Automation Society.

Keywords: Neural networks; Computer modeling and simulation; Control; Optimization; Steel industry; Drinking water treatment

1. INTRODUCITON

Artificial neural networks have been the focus of a great deal of attention during the last two decades, due to their capabilities to solve nonlinear problems by learning from data. Although a broad range of neural network architectures can be found, multilayer perceptrons (MLP's) and radial basis function networks (RBFN's) are the most popular neural models, particularly for system modeling and identification [1,2], control [3,4], and time series forecasting.

In Section 2, these two neural models are presented and related to the general task of system identification from experimental data. The different methods for choosing the input variables (regressors), selecting the internal architecture, and learning the weights (i.e., estimating the parameters) are reviewed. The advantages of these models are then summarized, as compared to other nonlinear structures. The third part briefly introduces the application of neural networks to process control. Finally, two case studies are described in the last section: the intelligent control of a hot dipped galvanizing line, and the control of a coagulation process in a water treatment plant. These two applications show the interest of neural learning in an industrial production context when complex physical phenomena are involved, particularly at the upper level of set-point determination.

**E-mail address*: gerard.bloch@esstin.uhp-nancy.fr
†*E-mail address*: Thierry.Denoeux@utc.fr

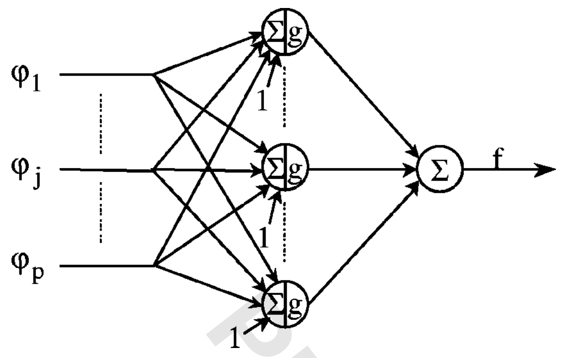

Fig. 1. One hidden layer perceptron, with linear output node.

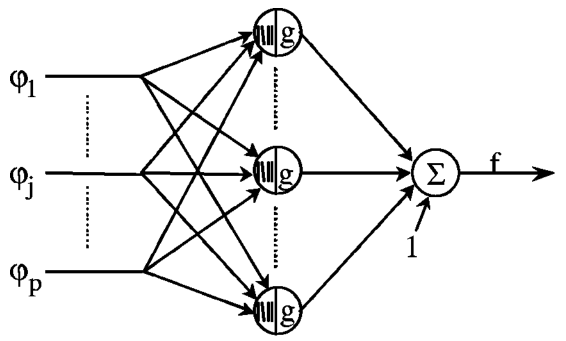

Fig. 3. Radial basis function network.

2. NONLINEAR SYSTEM MODELING WITH NEURAL NETWORKS

2.1 Two neural models

Only a reduced form of multilayer perceptron (MLP) (or feedforward sigmoid neural network) will first be presented here: the one hidden layer perceptron with linear output unit. Although particular, this model will be called MLP in the following. Its form is given, for single output f, by

$$f=\sum_{k=1}^{n} w_k^2 g\left(\sum_{j=1}^{p} w_{kj}^1 \varphi_j + b_k^1\right) + b^2, \qquad (1)$$

where φ_j, $j=1,\ldots,p$, are the inputs of the network, w_{kj}^1 and b_k^1, $k=1,\ldots,n$, $j=1,\ldots,p$, are the weights and biases of the hidden layer, the activation function g is a sigmoid function, often chosen as the hyperbolic tangent $g(x)=2/(1+e^{-2x})-1$, w_k^2, $k=1,\ldots,n$, and b^2 are the weights and bias of the output neuron or node (see Figs. 1 and 2).

The restriction to only one hidden layer and to a linear activation function at the output brings the general perceptron closer to other nonlinear models, neural or not. Indeed, the one hidden layer perceptron corresponds to a unique particular choice, the sigmoid function, for the basis function g_k, and to a "ridge" construction for the inputs [2] in a function expansion:

$$f(\varphi,\theta)=\sum_{k=1}^{n} \alpha_k g_k(\varphi,\beta_k), \qquad (2)$$

where $\varphi=[\varphi_1 \cdots \varphi_p]^T$ is the regression vector and the parameter vector θ is the concatenation of all the weights w and biases b.

Choosing a Gaussian function $g(x)=e^{-x^2/\sigma^2}$ as basis function and a radial construction for the inputs leads to the radial basis function network (RBFN) [5]:

$$\begin{aligned} f(\varphi,\theta) &= \sum_{k=1}^{n} \alpha_k g_k(\varphi)+\alpha_0 \\ &= \sum_{k=1}^{n} \alpha_k g(\|\varphi-\gamma_k\|_{\beta_k})+\alpha_0 \\ &= \sum_{k=1}^{n} \alpha_k \exp\left(-\frac{1}{2}\sum_{j=1}^{p}\frac{(\varphi_j-\gamma_{kj})^2}{\beta_{kj}^2}\right)+\alpha_0 \end{aligned} \qquad (3)$$

where $\gamma_k=[\gamma_{k1}\cdots\gamma_{kp}]^T$ is the "center" or "position" of the kth Gaussian and $\beta_k=[\beta_{k1}\cdots\beta_{kp}]^T$ its "scale" or "width" (see Figs. 3 and 4).

The process of approximating nonlinear relationship from data can be decomposed in several steps:

- determining the structure of the regression vector φ or selecting the inputs of the network;

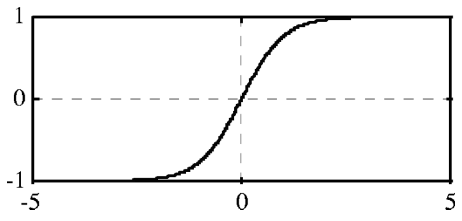

Fig. 2. Hyperbolic tangent function.

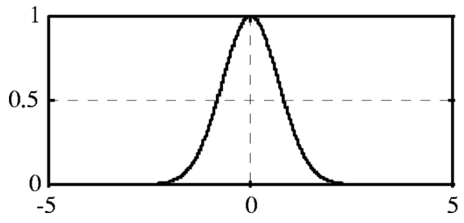

Fig. 4. Gaussian bell.

- choosing the nonlinear mapping f or, in the neural network terminology, selecting an internal network architecture;
- estimating the parameter vector θ, i.e., (weight) ‘‘learning’’ or ‘‘training.’’

As recalled in Fig. 5, this approach is similar to the classical one for linear system identification [6], the selection of the model structure being, nevertheless, more involved. Several general comments concerning these three points will be done in the following.

2.2 The regressors

For dynamic systems in discrete time t, a natural approach [7,3] is to reuse the input structure of linear models, particularly the general input-output model family [6]

$$A(q^{-1})y(t)=\frac{B(q^{-1})}{F(q^{-1})}u(t)+\frac{C(q^{-1})}{D(q^{-1})}e(t), \tag{4}$$

where $u(t)$ and $y(t)$ are, respectively, the system input and output, $e(t)$ is a white noise independent

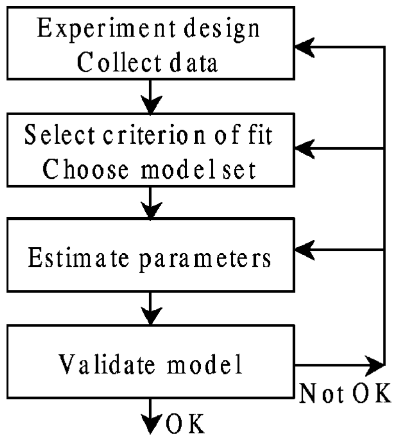

Fig. 5. Identification procedure.

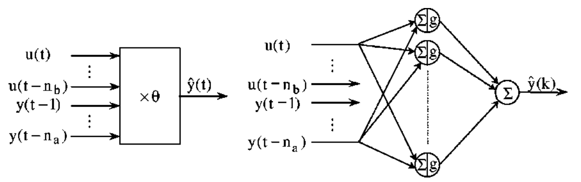

Fig. 6. ARX (left) and NARX (right) models.

from past inputs, and where A, B, C, D and F are polynomials in the backward shift operator q^{-1}.

This approach has several attractive advantages, pointed out by Nørgaard [8], namely:

- It is a natural extension of the well-known linear models. The internal architecture can be increased gradually as a higher flexibility is needed to model more complex nonlinear relationships.
- The structural decisions required by the user are reduced to a level that is reasonable to handle.
- The approach is suitable for the design of control systems.

The predictor associated with model (4) can be expressed in ‘‘pseudolinear’’ form as $\hat{y}(t|\theta) = \varphi(t,\theta)^T\theta$, where φ is the regression vector and θ is the parameter vector. It can be extended to nonlinear models as $\hat{y}(t|\theta)=f(\varphi(t,\theta),\theta)$. Depending on the choice of the regressors in $\varphi(t)$, different models, with *N* (for nonlinear) or *NN* (for neural network) added, can be derived [9]:

- NFIR, with delayed measured inputs $u(t-k)$ as regressors;
- NARX, with delayed measured inputs $u(t-k)$ and outputs $y(t-k)$ as regressors;
- NOE, with $u(t-k)$ and outputs simulated from past inputs u only $\hat{y}_u(t-k|\theta)$ as regressors,
- NARMAX, with $u(t$-$k)$, $y(t-k)$, and prediction errors $\varepsilon(t-k)=y(t-k)-\hat{y}(t-k|\theta)$ as regressors,
- NBJ, with $u(t$-$k)$, $\varepsilon(t-k)=y(t-k)$-$\hat{y}(t-k|\theta)$, $\varepsilon(t-k)$, and $\varepsilon_u(k-t)=y(k-t)-\hat{y}_u(k-t|\theta)$ as regressors.

As an example, Fig. 6 illustrates the parallel between (linear) ARX and NARX models.

Several methods have been proposed for the selection of the regressors prior to parameter estimation. Battiti [10] used entropy measures for selecting ''features.'' For dynamic systems, He and Assada [11] described a very computationally extensive method to determine the lag space, i.e., the number of delayed signal used as regressors, for deterministic systems. But most of the time, the selection of the network inputs is done after or during learning, see e.g., Ref. [12], and is part of the network architecture determination process.

2.3 Selection of the network architecture

Most of the methods for finding the optimal network architecture in view of a particular estimation problem are iterative techniques and are more or less derived from linear regression algorithms, where the architecture selection is embedded in parameter estimation. They are applied to the selection of inputs, hidden nodes, or individual weights. They can be classified into three groups:

- Forward selection adds the ''best'' neuron (and the corresponding parameters) to an existing model, see for instance Refs. [13] and [14].
- Backward selection removes the ''least relevant'' parameters, including pruning methods, see Refs. [15] and [16], for reviews. The optimal brain damage (OBD) [17] and the optimal brain surgeon (OBS) [18] algorithms are the most widely used pruning methods. In these methods, an initial network, ''large enough'' to describe the system, is determined and then reduced iteratively, by removing useless ''spurious'' parameters. As pruning leads to a simpler model, it alleviates the overfitting problem, i.e., the learning of noise and unknown underlying model of the system at the same time; as a result, it generally leads to an improvement of the model generalization abilities.
- Finally, stepwise regression combines both approaches; see Ref. [12] for regressor selection, or Ref. [19].

2.4 Parameter estimation

Learning (i.e., parameter estimation) methods for MLP's are very numerous and can be presented in three classes. In the first one, methods exploiting the particular architecture of these networks as a succession of layers can be found; see Refs. [20] and [21], for instance. The second class comprises various first- or second-order local, gradient-based procedures; see Ref. [22] for a review. Global, or stochastic, optimization methods constitute the third class, including particularly evolutionary algorithms. As reviewed by Yao [23], such algorithms are used not only for parameter estimation, but also for architecture determination and learning rule adaptation. Hybrid methods, combining gradient descent and evolutionary algorithms, have also been proposed.

The performances of learning algorithms for MLP's are sensitive to a large range of factors, including:

- the choice of the error function, which can be simply quadratic, robust to outliers [24] or regularized;
- the weight initialization scheme, which can considerably influence the number of iterations and may have an impact on generalization [25,26];
- the stopping criterion;
- parameters specific to the different methods, as well as, i.e.,...the user's skills in using a particular one.

The best method is thus problem dependent. The batch Levenberg-Marquardt algorithm, although giving no guarantee to reach a global minimum, is often recommended.

For RBF networks, there are different approaches to estimate the parameters α_k, $k = 0,\ldots,n$, which appear linearly in model (3), the centers $\gamma_k = [\gamma_{k1} \ldots \gamma_{kp}]^T$ and widths $\beta_k = [\beta_{k1} \ldots \beta_{kp}]^T$, $k = 1,\ldots,n$. A commonly used method separates the estimation of the $\alpha_k$parameters, on one hand, from that of centers and scales, on the other hand. The centers and the scales are determined in an unsupervised manner, i.e., without using the outputs y of the system, by one of the various clustering methods, such as the hard or fuzzy C-means, for example. Such methods aim at determining compact clusters in a set of multidimensional points, in our case the different system input observations. The centers of gravity of the clusters are used as centers for the RBF's. The scales β_k can then be computed from the clusters or fixed heuristically by the user (sometimes the same β is used for each RBF and each input dimension). The radial basis functions $g_k(\varphi)$ being fixed, the α_k parameters are simply determined by least-squares estimation. Simplicity is often claimed as the main advantage for the RBF networks. Nevertheless, the determination of the centers by clustering is not always obvious, and its suffers from the drawbacks as other methods: numerous parameters to tune (type of algorithm, number of clusters, initial centers, metric used,$\cdots$), and dependence of the results to initialization. Moreover, learning the centers without supervision is obviously suboptimal with respect to the approximation task.

Another approach starts from a small enough number of centers and estimates simultaneously the ‘‘linear’’ parameters and the centers and widths, by an iterative method, such as one of the previously described for MLP’s. See, for example, Ref. [27].

Finally, there are efficient but memory demanding methods, more or less derived from the orthogonal least-squares (OLS) algorithm [28], which allow us to obtain simultaneously the ‘‘linear’’ parameters, the centers, and their number [29]. All the input observations are considered as candidate centers and, after orthogonalization, are incorporated one by one in a forward manner until a specified error threshold is reached.

2.5 The advantages of the one hidden layer perceptron and RBFN

Among the numerous nonlinear models, neural or not, which can be used to estimate a nonlinear relationship, the one hidden layer perceptron (OHLP), as well as the radial basis function network (RBFN), present interesting features, which can be summarized in few words: they are *flexible and parsimonious nonlinear black box models*, with *universal approximation capabilities*.

Several researchers have proved that OHLP [30] as well as RBFN [5] are universal approximators, i.e., they can approximate any nonlinear function, from a space of finite dimension to another, with any degree of accuracy. Other models share this property, such as polynomial models, trigonometric series, splines, and orthogonal function expansions. However, roughly speaking, OHLP’s and RBFN’s are expansions of parametrized functions involving adjustable parameters; consequently, it can be shown that they require fewer parameters than expansions of fixed functions to reach a specified error goal [31]. In that sense, they are parsimonious. The price to pay for using parametrized functions in the expansion is the existence of numerous local minima in the error surface.

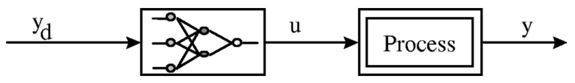

Fig. 7. Direct inverse control (open loop).

Moreover, OHLP’s and RBFN’s are flexible: the more complex (nonlinear) the relationship is to model, the more numerous will be the nodes or the parameters of the corresponding neural network. That means that their internal complexity can be easily increased, without changing the global form of the model. They belong to the general class of nonparametric models that do not make any assumption about the parametric form of the function to be approximated. In that sense, they constitute flexible regression tools.

For these various reasons, only these two neural models, the one hidden layer perceptron and the radial basis function network, are employed in the applications described further. This choice avoids having to determine the number of hidden layers.

3. NEURAL NETWORKS FOR CONTROL

Neural networks can be included in various control schemes [4,32,33]. Agarwal [34] proposed a systematic classification, with two main classes. In the first category, neural networks are only used as aids for system modeling, control-law implementation, or supervisory action. In the second one, they are used as controllers, with different training approaches. Before presenting examples of systems in each category, we first discuss some general issues regarding the design of neural network based controllers.

3.1 Controller learning

One of the first control strategies which has been proposed is to ‘‘train’’ a neural network to behave like the inverse of the process, and then use it as a controller (see Fig. 7). For a nonlinear SISO process to be controlled, with input u and output y, it is assumed that the model can be expressed by

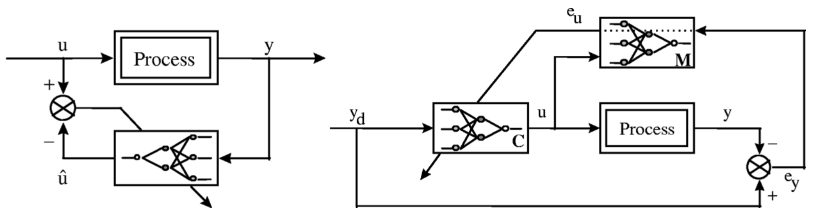

Fig. 8. (a) Direct learning. (b) Specialized learning through direct neural model.

$$y(t+1)=f(y(t),\ldots,y(t-n+1),$$
$$u(t),\ldots,u(t-m)).$$

So the inverse model of the system can be built by using a neural network:

$$\hat{u}(t)=\hat{f}^{-1}(y(t+1),y(t),\ldots,y(t-n+1),$$
$$u(t-1),\ldots,u(t-m)).$$

This latter model can be used for control by replacing the actual system output at time $t+1$, $y(t+1)$, by the reference $y_d(t+1)$ [35].

In the case where the direct model is one-to-one and stable, learning of the inverse model can be done directly from the system only [cf. Fig. 8(a)]. The inverse model can also be taught to be configured as a process controller C by a recursive gradient-based algorithm. This ‘‘specialized’’ learning requires the process Jacobian (gradient), which can be obtained from physical knowledge of the process, if available, or approximated by

- applying small variations (Δu) to the process input;
- observing the output (Δy), and
- calculating an approximate gradient ($\Delta y/\Delta u$).

Alternatively, the process can be approximated by a (direct) linear model M or by a nonlinear neural one, from which the gradient can be derived [see Fig. 8(b)] [27]. Note that the latter approach is quite close to conventional adaptive control.

3.2 Several neural control schemes

Control schemes using neural networks can also be divided according to the use of a direct model of the process. If such a model is not required, the methods are called direct control. They include the copy of an existing controller, in case of complicated or costly devices used as controller, or of human, nonexplicit, control laws; direct control with inverse model (Fig. 7); adaptive direct control; feedforward direct control, etc. In feedforward direct control, the inverse neural model is used in parallel with a conventional controller (e.g., PID) (see Fig. 9). The role of the PID is to ensure regulation and stabilization of the controlled process, while the neural network compensates for nonlinearities of the process. Supplementary variables can be applied at the input of the inverse neural model, static or dynamic, to take into account changes in operating points. This scheme has been implemented in one of the applications described in the next section.

Indirect-type control methods require a direct model of the process to be controlled. They include optimal control, indirect adaptive control, internal model control, predictive control, control by feedback linearization, etc. As an example, a scheme of neural internal model control is given, for stable processes, in Fig. 10.

Most of the classical control schemes can be extended by using neural networks. The major difficulties in their application to control lie in the associated computation costs, for fast systems, and

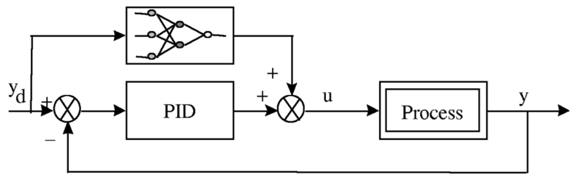

Fig. 9. Neural feedforward control.

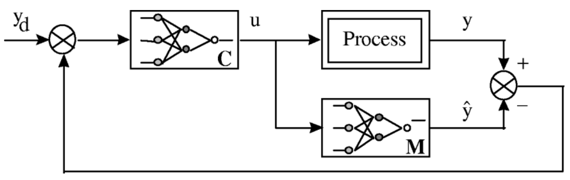

Fig. 10. Neural internal model control.

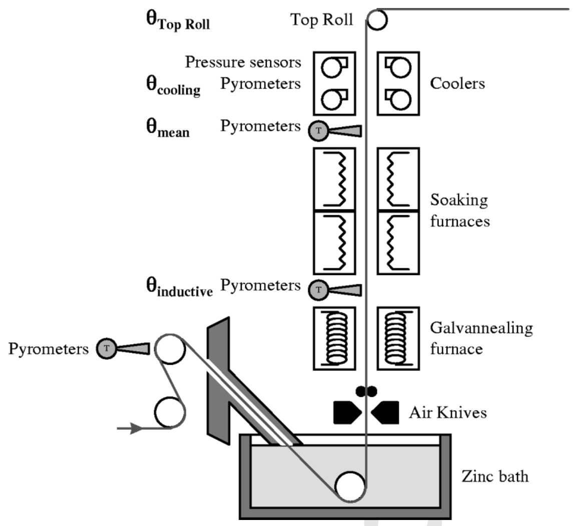

Fig. 11. The galvannealing section.

in the difficulties to establish the stability of the resulting schemes. It must be said that developing neural dynamic control often remains an excessively heavy task when classical linear methods can be applied with acceptable results, even if computer aided tools are now available [36]. As shown in the following, neural techniques appear, nevertheless, as very useful alternatives when physical knowledge of process is insufficient, particularly at control levels higher than classical dynamic control.

4. TWO APPLICATIONS

4.1 Intelligent control of alloying process in a hot dipped galvanizing line

The first presented study was conducted as part of a collaboration between CRAN and the Sollac company, concerning the hot dip galvanizing line of Florange (France) [37]. The line was designed for the production of galvanized steel sheets of outside car panels with optimum surface quality. The line is 500 m long, and is equipped with about 5000 sensors. It produces 300 000 tons/year of galvanized steel sheet, at a speed of 80–120 m/min. A layout of the galvannealing section of this plant, including galvannealing furnace, soaking furnace, and air cooling section, is shown in Fig. 11. At the exit of the zinc bath, the coating strip is annealed to allow the diffusion of strip iron to the coating. The strip is reheated with an inductive furnace up to a set point, named inductive temperature ($\theta_{inductive}$), and goes through a soaking furnace, the inner temperature of which is called the mean temperature (θ_{mean}). It then passes through the cooling part to stop the galvannealing reaction. The quality of the product is related to the percentage of iron at the surface. An underalloyed product (lack of iron in the coat) is caused by an insufficient alloying temperature, while an overalloyed product is obtained when the thermal cycle is too high. The problem is to determine and control the optimal inductive temperature, knowing the operating conditions which are the speed, width, and thickness of the strip and the heating power applied to the furnace.

This brief description highlights some general features of plants in the steel industry: numerous and interconnected describing variables, complex physical phenomena, only partially known in an industrial production context, nonlinear relationships, importance of the skill of the operators. As pointed out by Harris [33], many processes, being too complex for direct modeling based on physical laws, are manually regulated by human operators before automatic controls are installed. The plant operator is able to cope with plant nonlinearities and slowly varying parameters, to respond to complex sets of noisy observations and poorly specified constraints, and to satisfy multiple subjective performance criteria. Thus one of the basic ideas of the presented ‘‘intelligent’’ control application is to incorporate the flexible and creative attributes of human controllers, while avoiding their associated characteristics of unreliability.

The overall equipment effectiveness (OEE) of the line can be increased by analyzing the sources of losses: stopping (for failures, tools changes, preparations, and setting), slowing (light running, microfailures, production rate lowering), and product quality faults, and classifying them with respect to their economical impact. The means for eliminating or reducing these loss causes are then related to the functional hierarchical decomposition of the plant (computer integrated production): sensors, control, optimization. The approach must improve the performances over a wide range of operating conditions and the fault tolerance and reconfigurability degree of the plant. An important aspect is to reduce the design cost of control procedures. The low level validation of measurements (temperatures, pressures), the supervision of sen-

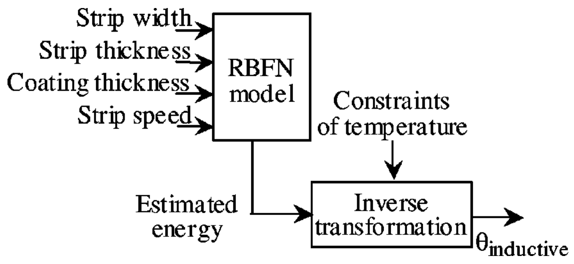

Fig. 12. Estimation of the inductive temperature.

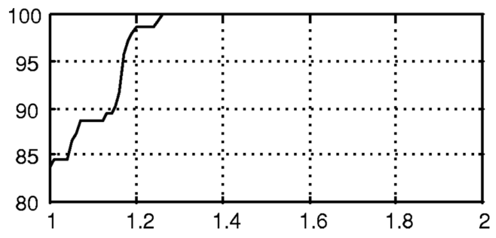

Fig. 14. Precision of predicted temperatures (% of points with respect to error in %).

sors and operating conditions, will be not described here. Only the optimization and control aspects are presented.

4.1.1. Optimization of the alloying thermal cycle

The improvement of the product quality requires the determination of the optimal set points of the thermal cycle, particularly $\theta_{inductive}$, for different line speeds and product types. The metallurgy knowledge is not sufficient to explain and to know the optimal temperature of the alloying reaction. The complexity of the reaction and the numerous nonlinear relationships between all variables lead to the use of learning algorithms from the operating points fixed by the control operators during several months.

Figure 12 summarizes the scheme to estimate the $\theta_{inductive}$ temperature. The idea is to model the energy supplied to the strip, with respect to the features of the strip and the operating conditions, and then use the constraints of the thermal cycle to calculate $\theta_{inductive}$. For the energy model building (cf. Fig. 13), the considered variables are the line speed, the features of the strip, and the ‘‘measured’’ energy, calculated from all the temperatures of the tower and the line speed, and used as target for learning. From dynamic data, operating points are extracted to generate static databases. Each point is validated as good or bad using several criteria fixed by the operators, with respect to the quality of the product, and only good points are kept. The remaining steady states are then separated in two sets, a modeling one of 260 points and a test one of 130 points. The process typically operates around a finite number of operating points corresponding to the different strip formats and speeds: consequently, the most suitable neural model was found to be a radial basis function network. The neural model building process is described in Ref. [37], where *K*-means or fuzzy *C*-means clustering methods are compared with orthogonal least-squares (OLS) [28] squares algorithm for the determination of the hidden node number (see paragraph 2.4). Final results are considered as very satisfactory: the prediction errors of inductive temperature calculated from the validation database are never greater than the absolute precision of the corresponding sensor. As shown in Fig. 14, 98% of the points are estimated with an error lower than 1.2%.

4.1.2. Control of the induction furnace

This part is focused on the alloying cycle control, and particularly the strip temperature at the exit of the induction furnace. A power preset (*Pgal*) to apply to the furnace is determined using a steady-state inverse model to obtain a strip temperature close to the optimal temperature estimated previously (cf. Fig. 15). The behavior of the

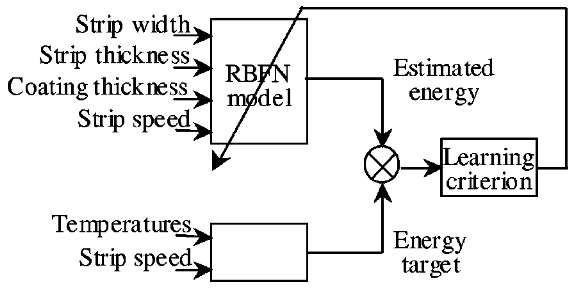

Fig. 13. Model learning.

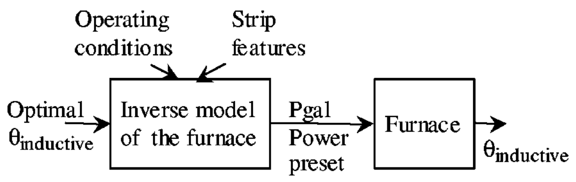

Fig. 15. Open loop control of the inductive temperature.

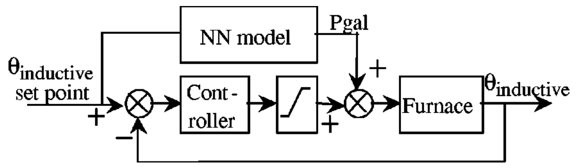

Fig. 16. Control architecture for the inductive temperature.

furnace being nonlinear, a perceptron, with one hidden layer of sigmoidal units and linear output node, is used to build the inverse model. Quadratic or robust criteria are employed to estimate the weights. The results are compared, for the different criteria and for various hidden node numbers, by considering the minimal values of the root mean square error and the maximal absolute error on learning and test data sets. The best model, with only four hidden nodes, is obtained from robust learning. Because of the small modeling errors, and to take into account the weak fluctuations of the unknown strip temperature at the entrance of the furnace, a control loop is implemented on the process (see Fig. 16). Note that the chosen strategy includes the possibility to disconnect the control loop in case of measurement unavailability of $\theta_{inductive}$. That allows for the use of only the neural inverse model in open loop in order to maintain a sufficient degree of fault tolerance.

The neural learning approach allows the system to incorporate the skill of the control operators in automatic control and optimization systems, with a moderate design cost. While guaranteeing a required degree of fault tolerance, the implemented control architecture leads to a decrease of the occurrence of the underalloyed products and permits a progressive reduction of operator intervention in furnace control.

4.2 Control of a coagulation process in water treatment

4.2.1. Context of the application

Water treatment involves complex physical, chemical, and biological processes that transform raw water into drinking water. In spite of important fluctuations in raw water characteristics, due to natural perturbation or occasional pollution, the quality of the drinking water produced has to be maintained at a level compatible with official standards, while minimizing operating costs. The objective of this second study, which is the result of a collaboration between Heudiasyc and Ondeo, was to build a model of the coagulation process, so as to determine the optimum quantity of chemical reagents, as a function of input water quality [38]. As no model of this process is available, a neural network based system was designed for that purpose.

Figure 17 depicts the main processes in a typical plant for surface water treatment. Raw water is abstracted from the resource (a river in this case) and pumped to the treatment works. A typical plant consists in two main process units: clarification and filtration. The coagulation process, which takes place in the clarification unit, is brought about by adding a highly ionic salt (aluminum sulfate) to the water. A bulky precipitate is formed and removed as sludge. The coagulation process accounts for the elimination of most of the undesirable substances from the raw water and hence tight monitoring and control of this process is essential. The main difficulty is to determine the optimum quantity of chemical reagent related to raw water characteristics. Poor control leads to waste of expensive chemicals, failure to meet the water

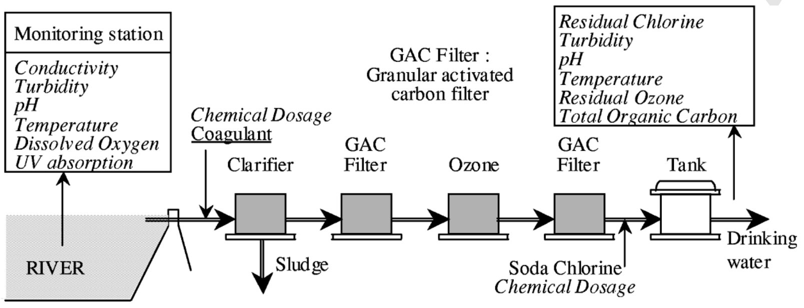

Fig. 17. Simplified synopsis of a water treatment plant.

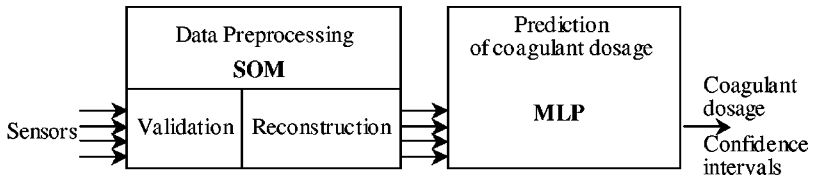

Fig. 18. Structure of the system for automatic coagulation control.

quality targets, and reduced efficiency of sedimentation and filtration processes.

4.2.2. Specific requirements

Given the high variability of the inputs and the low reliability of available sensors, an important requirement in this application is *robustness* against erroneous sensor measurements or unusual water characteristics, due to accidental pollution. In our system, such a robustness is achieved using a modular architecture composed of two levels: a preprocessing level responsible for outlier rejection and missing data reconstruction, and a prediction level involving the determination of the optimal coagulant amount from raw water characteristics (Fig. 18). Neural network models are involved at both levels: data validation and reconstruction is carried out by a self-organizing feature map (SOM) which compares input vectors to reference patterns learned in an unsupervised manner, and prediction of coagulant amount is performed by a MLP.

A second important requirement from the considered application is the possibility to install the system at low cost in various sites, which necessitates a methodology for designing and training the neural networks automatically from new data, including the phases of data validation and model choice. Our system uses pruning and resampling techniques for automatic determination of the network architecture and computation of confidence bounds for the predictions.

4.2.3. Data preprocessing

Applications in the environmental domain such as the one considered in this section generally rely on complex sensors located at remote sites. The processing of the corresponding measurements for generating higher level information (such as predictions of optimal coagulant dosage) must therefore account for possible sensor failures and incoherent input data. This can be achieved by computing distances between input vectors and reference patterns, or *prototypes*. The determination of prototypes from data in an unsupervised way may be performed using the self-organizing map (SOM) algorithm introduced by Kohonen [39]. The SOM model can be used at the same time to visualize the clusters in a data set, and to represent the data on a two-dimensional map in a manner that preserves the nonlinear relations of the data items, nearby items being mapped to neighboring positions on the map. A previous application of SOM's to water quality monitoring was described in Ref. [40].

Self-organizing maps allow the system to detect atypical data or outliers by monitoring the distance between each input vector x and its closest reference vector. If this distance is greater than a specified threshold, the current sample is considered invalid. The contributions of each of the components of vector x to the distance are then examined to determine more precisely which sensors should be declared faulty. These sensor measurements are then disconnected to compute a new winning prototype with only valid parameters.

For reconstruction, each missing value of a given input variable is estimated by the value of the corresponding component of the winning prototype. In order to improve the reconstruction accuracy, a combination of the k nearest nodes is used. Each missing or invalid value is estimated by a combination of the corresponding component in the k nearest prototypes. More details about this procedure can be found in Ref. [38].

4.2.4. Prediction of coagulant dosage

The prediction of optimal coagulant dosage from water characteristics was addressed as a nonlinear regression problem. Six variables measured continuously on the raw water (turbidity, conductivity, pH, temperature, dissolved oxygen, and UV absorption) were used as input variables. Target values for the coagulant dosing rate were provided by results of laboratory analyses (‘‘jar tests’’) per-

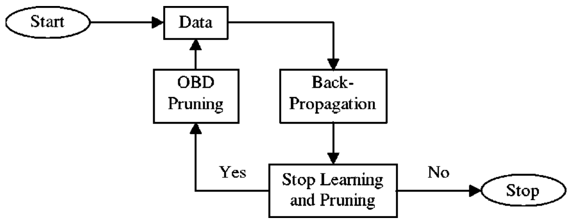

Fig. 19. Learning and pruning algorithm.

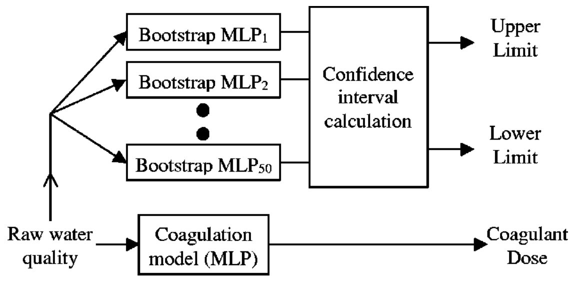

Fig. 20. Bootstrap sampling for the generation of prediction intervals.

formed once a day by the plant operators. One year of raw data was available, from which a set of 1600 complete learning samples was constructed by removing erroneous and incomplete measurements, and averaging the data over 1-h time intervals. A total of 1120 samples (about 70%) was exploited to build the model, the rest being used as an independent test set. Among the training data, approximately 30% was left out as a validation set for optimizing the architecture.

We used a conventional MLP architecture with one hidden layer of sigmoidal units, and training was performed by minimization of the mean-squared error function. For the determination of the architecture, the optimal brain damage (OBD) pruning algorithm [17] was used. This method is summarized in Fig. 19. A large initial network is first trained using the back-propagation algorithm applied to the sum of squares error function. The ‘‘saliency’’ of each weight is then computed as a function of the second derivatives of the error function, and the weights with the lowest saliency values are deleted. The process is iterated until the cross- validation error, as estimated using an independent data set, starts to increase. In this approach, the initial number of hidden units is only required to excess the optimal hidden layer size. It was fixed arbitrarily to 20 in our application, corresponding to 161 initial connection weights. The network complexity was then automatically reduced by the pruning procedure to a final number of 16 hidden units and 66 weights.

To better assess the reliability of the system in on-line operation, it was demanded by the operators that the system provide not only point estimates of the coagulant dosing rate, but also confidence intervals. Bootstrap sampling [41] was used to generate confidence intervals for the system outputs, following an approach first proposed in Ref. [42]. This technique is illustrated in Fig. 20. In this approach, b bootstrap subsets of the initial training set are used to train b MLP models using the architecture and training procedure described previously. When a vector is fed into these networks, the b outputs provide an estimate of the distribution of the target variable for the current input. Lower and upper confidence limits for the prediction related to any given input vector are then obtained by sorting these outputs and selecting, e.g., the 10% and 90% cumulative levels. The prediction accuracy and confidence bounds computed on a validation set are shown in Fig. 21.

Extensive field testing on a pilot site has demonstrated the efficiency of the approach, and widespread dissemination to other sites is currently planned. Expected benefits are treated water of a more consistently high quality, together with improved security of service, as the system will respond reliably and effectively over long periods. Significant savings in coagulant usage have al-

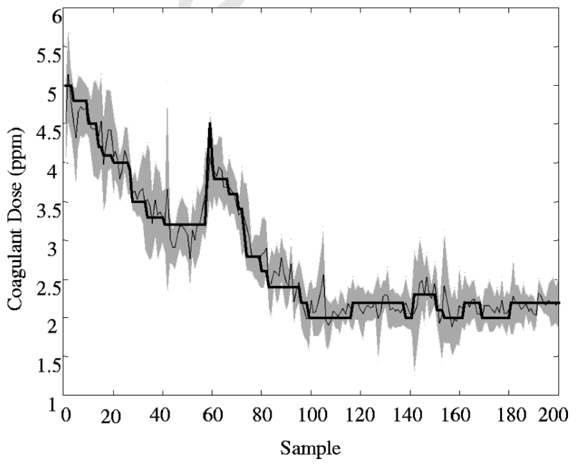

Fig. 21. Actual (thick line) versus predicted (thin line) coagulant dosage with neural network model on test data and confidence interval (shaded region).

ready been observed. The performance of the network is obviously dependent on the quality and completeness of the data available for training the system. Consequently, continuous updating of training data during operational use is expected to further improve the performance of the system.

5. CONCLUSION

The use of artificial neural networks for control is motivated by their universal approximation capabilities. The feedforward one hidden layer perceptron, with linear activation at the output, and the radial basis function network provide simple and flexible structures for nonlinear modeling. As illustrated through the two applications previously described, neural learning is particularly useful when complex nonlinear phenomena, only partially known in an industrial production context, are involved. In such a case, the empirical knowledge of control operators can be learned, particularly for determining optimal set points. So, if most of the classical dynamic control schemes can be extended by using neural networks, neural techniques seem particularly adapted to control levels higher than classical dynamic control, and yield control applications with interesting properties: operation over a wide range of conditions, improved fault tolerance degree, and moderated design cost.

References

[1] Chen, S. and Billings, S., Int. J. Control **56**, 319–346 (1992).
[2] Sjöberg, J. *et al.*, Automatica **31**, 1691–1724 (1995).
[3] Narendra, K. S. and Parthasarathy, K., IEEE Trans. Neural Netw. **1**, 4–27 (1990).
[4] Hunt, K., Sbarbaro, D., Sbikowski, R., and Gawthrop, P. J., Automatica **28**, 1083–1112 (1992).
[5] Poggio, T. and Girosi, F., Proc. IEEE **78**, 1481–1497 (1990).
[6] Ljung, L., System Identification: Theory for the User, 2nd edition (Prentice-Hall, Englewood Cliffs, NJ, 1999).
[7] Sjöberg, J. *et al.*, Automatica **31**, 1691–1724 (1995).
[8] M. Nørgaard, System identification and control with neural networks, Ph.D. thesis, Dept. of Automation, Technical University of Denmark, 1996.
[9] Sjøberg; and Ngia, L. S. H., in Nonlinear Modeling–Advanced Black-Box Techniques, edited by J. Suykens. Kluwer Academic Publishers, Boston, 1998, Chap. 1, pp. 1–28.
[10] Battiti, R., IEEE Trans. Neural Netw. **5**, 537–550 (1994).
[11] He, X. and Assada, H., American Control Conference, San Francisco, San Francisco, CA, 1993.
[12] Cibas, T., Fogelman Soulie, F., Gallinari, P., and Raudys, S., Neurocomputing **12**, 223–248 (1996).
[13] Fahlman, S. E., and Lebiere, C., in Advances in Neural Information Processing Systems, edited by D. S. Touretsky. Morgan Kaufmann, San Mateo, CA, 1990, Vol. 2, pp. 524–532.
[14] Lengellé, R. and Denoeux, T., Neural Networks **9**, 83–97 (1996).
[15] Reed, R., IEEE Trans. Neural Netw. **4**, 740–747 (1994).
[16] Jutten. C., and Fambon, O., 3rd European Symposium on Artificial Neural Networks ESANN'95, Brussels, Belgium, 129–140, 1995.
[17] Le Cun, Y., Denker, J. S., and Solla, S., in Advances in Neural Information Processing Systems (Ref. [13]), p. 598– 605.
[18] Hassibi, B., and Stork, D. G., in Advances in Neural Information Processing Systems, edited by S. H. Hanson. Morgan Kaufmann, San Mateo, CA, 1993, Vol. 5, pp. 164–171.
[19] Ji, C. and Psaltis, D., Neural Networks **10**, 1133–1141 (1997).
[20] Parisi, R., DiClaudio, E., Orlandi, G., and Rao, B. D., IEEE Trans. Neural Netw. **7**, 1450–1460 (1996).
[21] Thomas, P., Bloch, G., and Humbert, C., 6th European Symposium on Artificial Neural Networks ESANN'98, Bruges, Belgium, 279–284, 1998.
[22] Sheperd, A. J., Second-Order Methods for Neural Networks, Springer, London, 1997.
[23] Yao, X., Proc. IEEE **87**, 1423–1447 (1999).
[24] Bloch, G., Thomas, P., and Theilliol, D., Neurocomputing **14**, 85–99 (1997).
[25] Denoeux, T. and Lengellé, R., Neural Networks **6**, 351–363 (1993).
[26] Thomas. P., and Bloch, G., 15th IMACS World Congress on Scientific Computation, Modeling and Applied Mathematics, edited by A. Sydow and T. Verlag. Berlin, 1997, Vol. 4, pp. 295–300.
[27] Renders, J. M., Algorithmes Génétiques et Réseaux de Neurones. Hermés, Paris, 1995.
[28] Chen, S., Cowan, F. N., and Grant, P. N., IEEE Trans. Neural Netw. **2**, 302–309 (1991).
[29] Orr, M. J. L., Recent advances in radial basis function networks, Technical Report www.ed.ac.uk/~mjo/papers/recad.ps, Institute for Adaptive and Neural Computation, Edinburgh University, UK, 1999.
[30] Cybenko, G., Math. Control, Signals, Syst. **2**, 303–314 (1989).
[31] Barron, A. R., IEEE Trans. Inf. Theory **39**, 930–945 (1993).
[32] Narendra, K. S. and Parthasarathy, K., IEEE Trans. Neural Netw. **1**, 4–27 (1990).
[33] Harris, C. J., Advances in Intelligent Control. Taylor & Francis, London, 1994.
[34] Agarwal, M., IEEE Control Syst. Mag. **17**, 78–84 (1997).
[35] Hunt, K., Sbarbaro, D., Sbikowski, R., and Gawthrop, P. J., Automatica **28**, 1083–1112 (1192).
[36] Nørgaard, M., Neural network based system identification toolbox. Technical Report 95-E-773, Institute of

Automation, Technical University of Denmark, 1995.
[37] Bloch, G., Sirou, F., Eustache, V., and Fatrez, P., IEEE Trans. Neural Netw. **8**, 910–918 (1997).
[38] Valentin, N. and Denoeux, T., Intell. Data Anal. **5**, 23–39 (2001).
[39] Kohonen, T., Self-Organizing Maps. Springer-Verlag, Heidelberg, 1995.
[40] Trautmann, T., and Denoeux, T., ICNN'95, Perth, Australia, 73–78 (1995).
[41] Efron, B. and Tibshirani, R. J., An Introduction to the Bootstrap. Chapmann & Hall, New York, 1993.
[42] Lippmann, R. P., Kukolich, L., and Shahian, D., in Advances in Neural Information Processing Systems, edited by G. Tesauro. MIT Press, Menlo Park, CA, 1995, Vol. 7, p. 1055–1062.

Gérard Bloch received the Ph.D. degree in automatic control in 1988 from University Henri Poincaré, Nancy, France. He is currently a professor with the Ecole Supérieure des Sciences et Technologies de l'Ingénieur de Nancy at the same University. He is with the Centre de Recherche en Automatique de Nancy and his research interests include nonlinear system identification and observation, process diagnosis, and application of neural networks to automatic control.

Thierry Denoeux graduated in 1985 as an engineer from the Ecole Nationale des Ponts et Chaussees in Paris, and received a doctorate from the same institution in 1989. He is currently a full professor with the Department of Information Processing Engineering at the Univeristy of Compiègne, France. His current research interests concern fuzzy data analysis, belief functions theory and, more generally, the management of imprecision and uncertainty in data analysis, pattern recognition, and information fusion.