ELSEVIER

Fuzzy Sets and Systems 121 (2001) 59–72

FUZZY sets and systems

www.elsevier.com/locate/fss

The local paradigm for modeling and control: from neuro-fuzzy to lazy learning

Gianluca Bontempi*,[1], Hugues Bersini, Mauro Birattari[2]

Iridia – CP 194/6, Université Libre de Bruxelles, 50, av. Franklin Roosevelt, 1050 Bruxelles, Belgium

Abstract

The composition of simple local models for approximating complex nonlinear mappings is a common practice in recent modeling and control literature. This paper presents a comparative analysis of two different local approaches: the *neuro-fuzzy* inference system and the *lazy learning* approach. *Neuro-fuzzy* is a hybrid representation which combines the linguistic description typical of fuzzy inference systems, with learning procedures inspired by neural networks. *Lazy learning* is a memory-based technique that uses a query-based approach to select the best local model configuration by assessing and comparing different alternatives in cross-validation. In this paper, the two approaches are compared both as learning algorithms, and as identification modules of an adaptive control system. We show that *lazy learning* is able to provide better modeling accuracy and higher control performance at the cost of a reduced readability of the resulting approximator. Illustrative examples of identification and control of a nonlinear system starting from simulated data are given.

Keywords: Local modeling; Identification; Control design

1. Introduction

The problem of modeling a process from observed data has been the object of several disciplines from nonlinear regression to machine learning and system identification. In the literature dealing with this problem, two main opposing paradigms have emerged: the global versus the local one.

Global models have two main properties. First, they cover the whole set of operating conditions of the system underlying the available data. In general, this makes sense when it is reasonable to believe that the system is driven by a physical-like law. However, in many engineering problems, dealing in a global way with real processes requires complicated model representations. Second, global models solve the problem of learning an input–output mapping as a problem of function estimation: from a given set of parametric functions $f(x, \alpha)$ with $\alpha \in \Lambda$, the one which best approximates the unknown data distribution is selected. Examples of functional estimators are linear models, nonlinear statistical regressions, splines [21] and neural networks [42].

* Corresponding author. Tel.: +32-2-6503168; fax: +32-2-6502715.

E-mail addresses: gbonte@ulb.ac.be (G. Bontempi), bersini@ulb.ac.be (H. Bersini), mbiro@ulb.ac.be (M. Birattari).

[1] Supported by the European Union TMR Grant FMBICT960692.

[2] Supported by the F.I.R.S.T. program of the Région Wallonne, Belgium.

0165-0114/01/$ - see front matter
PII: S0165-0114(99)00172-4

The local paradigm originates from the idea of relaxing one or both of the global modeling features.

As a first step, the global description may be replaced by a modular architecture where the modules are simple models which focus on different part of the input space. This is the idea of *operating regimes* which assumes a partitioning of the operating range of the system in order to solve modeling and control problems [32]. Fuzzy inference systems [48], radial basis functions [36], CART [23] and hierarchical mixtures of experts [34], are well-known examples of this approach. It is important to remark that, although these architectures are characterized by an augmented readability, they still are a particular type of functional approximators.

Memory-based methods [6] aim to solve the learning problem relaxing also the second feature of global modeling. Given that the problem of functional estimation is hard to solve in a generic setting, they focus on approximating the function only in the neighborhood of the point to be predicted. In global modeling, a relatively simple problem (estimation of the function value) is solved by first solving a much more difficult intermediate problem (a function estimation). Memory-based learning, on the other hand, turns out to be a single-step approach where the learning problem is seen as a value estimation rather than a function estimation problem. To this end, memory-based methods require the storage of the dataset in opposition to functional methods which discard the data after training. Memory-based techniques are an old idea in classification [26], regression [25], and time-series prediction [29]. The idea of memory-based approximators as alternative to global models originated in non-parametric statistics [28,11] to be later rediscovered and developed in the machine learning fields [1,22].

This paper will focus on *neuro-fuzzy inference systems* and *lazy learning* as prototypes of these different ideas of local modeling. The aim is to provide the reader with a comparison between the two approaches in modeling and control tasks. As far as modeling is concerned, we will describe the respective learning algorithms by giving a particular attention to the model selection procedures and to the validation methods.

Neuro-fuzzy systems [24,31,13] are examples of hybrid modeling. The basic idea underlying these models is to reconcile a dichotomy emerged in literature between different approaches to the implementation of intelligent systems. On the one hand, approaches like neural networks renounce readability for performance. On the other, knowledge-based systems, like fuzzy systems, have the aim of harmonizing the continuous nature of reality with the symbolic nature of human reasoning. Hybrid approaches provide a third way: here the knowledge of the human expert is used to improve not only the readability of the models but also the performance of data-driven learning methods. As an example, we will propose our neuro-fuzzy technique which integrates a model structure based on fuzzy production rules and a tuning procedure inspired by neural networks.

The term *lazy learning* [2] designates the whole set of memory-based techniques that defer processing of the dataset until they receive an explicit request for information (e.g. prediction or local modeling). There has been recently a new impetus to the adoption of these techniques for modeling [7,45] and control problem [43,8]. In classical memory-based modeling, an amount of options is tuned by the data analyst according to heuristic criteria and a priori assumptions. Here, we propose a *lazy learning* technique whose main feature is the adoption of an automatic statistical procedure to select the local approximator. In particular, we use the PRESS statistic [39] which is a simple, well-founded and economical way to perform *leave-one-out* cross-validation and to assess the performance in generalization of local linear models. The algorithm that we propose is an automatic selection procedure which searches for the optimal model configuration, by returning for each candidate model its parameters and a statistical description of its generalization properties.

The contribution of the paper in the control domain is a comparison of the two approaches as alternative methods to extend linear control techniques to nonlinear discrete-time control problems. The idea of employing linear techniques in a nonlinear setting is not new in control literature but has recently gained a new popularity owing to methods for combining multiple estimators and controllers in different operating regimes of the system [44,38]. In particular, we will see a self-tuning regulator (STR) architecture [4] where discrete-time conventional control is combined with local model identification. This control system

can be thought of as composed of two loops. The inner one consists of the process and a feedback regulator. The parameters of the regulator are adjusted by the outer loop, represented by a neuro-fuzzy identifier or by a *lazy learning* estimator.

Our experimental results in identification and in control, show that the *lazy learning* approach outperforms both the neuro-fuzzy method and a conventional linear self-tuning regulator. Moreover, we show that *lazy learning* takes further advantage from its memory-based nature which makes it easily adaptable to changing external environments. In fact, when new data samples are available, no model re-training is required but a simple update of the existing dataset is sufficient.

The remainder of the paper is organized as follows. In Section 2 we will introduce the neuro-fuzzy architecture. In Section 3 we will introduce the *lazy* modeling technique based on a model selection procedure. Details on the control system implementation are given in Section 4. A comparison between the two algorithms as identification modules for local control can be found in Section 4.1. In Section 5 simulation examples of identification and of control are given. Finally, in Section 6 a comparison between the two approaches in terms of readability versus accuracy is provided.

2. Neuro-fuzzy as a multi model description

Takagi and Sugeno [48] introduced the fuzzy rule-based system for nonlinear modeling of a generic input–output mapping $f:\Re^m \to \Re$. A Takagi–Sugeno (TS) fuzzy inference system is a set of r rules

$$\begin{cases} \text{If } x_1 \text{ is } A_1^1 \text{ and } x_2 \text{ is } A_2^1 \ldots \text{and } x_m \text{ is } A_m^1 \\ \quad \text{then } y^1 = f^1(x_1, x_2, \ldots, x_m), \\ \ldots \\ \text{If } x_1 \text{ is } A_1^r \text{ and } x_2 \text{ is } A_2^r \ldots \text{and } x_m \text{ is } A_m^r \\ \quad \text{then } y^r = f^r(x_1, x_2, \ldots, x_m). \end{cases} \tag{1}$$

The first part (antecedent) of each rule is defined as a fuzzy AND proposition where A_j^i is a fuzzy set on the jth premise variable defined by the membership function $\mu_j^i:\Re^m \to [0,1]$. The second part (consequent) is a crisp function f^i of the input vector $[x_1, x_2, \ldots, x_m]$.

By means of the fuzzy sets A_j^i, the input domain of the function f is softly partitioned in regions where the mapping is locally approximated by the models f^i. The TS inference system uses the weighted mean criterion to recombine all the local representations in a global approximator:

$$\hat{y} = \frac{\sum_{i=1}^r \mu^i y^i}{\sum \mu^i}, \tag{2}$$

where μ^i is the degree of fulfillment of the ith rule.

An interesting special case is provided by the linear TS fuzzy inference system where the consequents are linear models $f^i = \sum_{j=1}^m a_j^i x_j + b^i$ [47]. In this case the TS system can be used to return a local linear approximation about a generic point of the input domain. Consider for example an input $\bar{\boldsymbol{x}} = [\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_m]$. The TS rule combination returns

$$f_{\text{lin}}(\boldsymbol{x}) = \frac{\sum_{i=1}^r \mu^i(\bar{\boldsymbol{x}}) \left(\sum_{j=1}^m a_j^i x_j + b^i\right)}{\sum \mu^i(\bar{\boldsymbol{x}})} \tag{3}$$

as the linear approximation about $\bar{\boldsymbol{x}}$ of the function $f(\boldsymbol{x})$.

In a conventional fuzzy approach, the membership functions and the consequent models are chosen by the model designer according to his a priori knowledge. If this knowledge is not available but a set of input–output data is observed from the process modeled by f, the elements of a fuzzy system can be put into a parametric form and the parameters can be tuned by a learning procedure. In this case the fuzzy system turns into a *neuro-fuzzy* approximator [13]. Neuro-fuzzy systems are a powerful trade off in terms of readability and efficiency between a human-like representation of the model and a learning method [16]. While learning methods adapt the parameters of the inference system to the observed data, the fuzzy architecture makes easier the task of integrating the available expert knowledge in the learning system. In the next section we will see in detail our neuro-fuzzy learning procedure.

2.1. Structural and parametric learning in neuro-fuzzy inference systems

In a neuro-fuzzy system, two types of tuning are required, namely *structural* and *parametric tuning*.

Structural tuning aims to find a suitable number of rules and a proper partition of the input space. Once

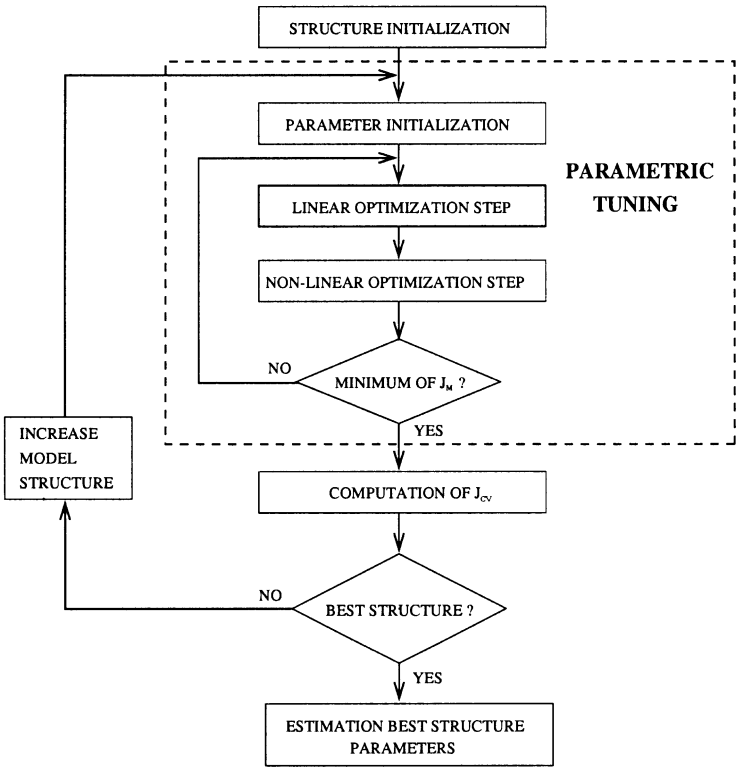

Fig. 1. Flow-chart of the neuro-fuzzy learning procedure.

available a satisfactory structure, the parametric tuning searches for the optimal membership functions together with the optimal parameters of the consequent models. There may be a lot of structure/parameter combinations which make the fuzzy model behave in a satisfactory way. The problem can be formulated as that of finding the structure complexity which will give the best performance in generalization [50]. In our approach, we choose the number of rules as the measure of complexity to be properly tuned on the basis of available data. We adopt an incremental approach where different architectures having different complexity (i.e. number of rules) are first assessed in cross-validation and then compared in order to select the best one. The whole learning procedure is represented in the flow chart in Fig. 1.

The initialization of the architecture is provided by a hyper-ellipsoidal fuzzy clustering procedure inspired by Babuska and Verbruggen [10]. This procedure clusters the data in the input–output domain obtaining a set of hyper-ellipsoids which are a preliminary rough representation of the input/output mapping. Methods for initializing the parameters of a fuzzy inference system from the outcome of the fuzzy clustering procedure are described in [9]. Here, we use the axes of the ellipsoids (eigenvectors of the scatter matrix) to initialize the parameters of the consequent functions f^i, we project the cluster centers on the input domain to initialize the centers of the antecedents and we adopt the scatter matrix to compute the width of the membership functions. An example of fuzzy clustering in the case of a single-input–single-output function modeled by a fuzzy inference system with gaussian antecedents is represented in Fig. 2.

Once the initialization is done, the learning procedure begins. Two optimization loops are nested: the parametric and the structural one. The parametric loop (the inner one) searches for the best set of parameters by minimizing a sum-of-squares cost function J_M which depends exclusively on the training set. In

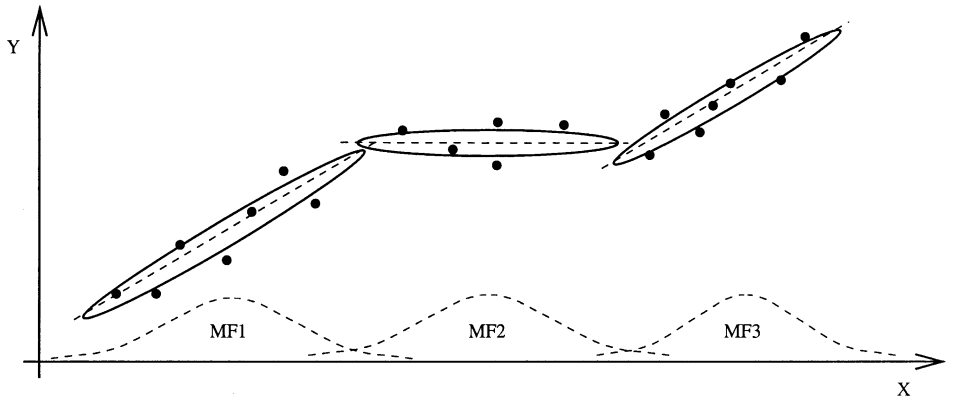

Fig. 2. The hyper-ellipsoidal clustering initialization procedure.

the case of linear TS models this minimization procedure can be decomposed into a least-squares problem to estimate the linear parameters of the consequent models f^i [31] and a nonlinear minimization (Levenberg–Marquardt) to find the parameters of the membership functions A_j^i [13].

The structural identification loop (the outer one) searches for the best structure, in terms of optimal number of rules, by increasing gradually the number of local models. The different structures are assessed and compared according to their performance J_{CV} in *K-fold* cross-validation [46]. This procedure uses a high proportion of the available data to train the current model structure and gives a reliable estimate of the performance in generalization. Unfortunately, the training process has to be repeated as many times as the number K of partitions of the training set, making the whole learning process computationally expensive.

The model with the best cross-validation performance is then selected to represent the input–output mapping and consequently trained on the whole dataset.

3. Lazy learning modeling

Lazy learning estimates the value of the unknown function by focusing on the region surrounding the point where the estimation itself is required.

Let us consider as unknown mapping $f : \Re^m \to \Re$ of which we are given a set of N samples $\{(\boldsymbol{\varphi}_1, y_1), (\boldsymbol{\varphi}_2, y_2), \ldots, (\boldsymbol{\varphi}_N, y_N)\}$. These examples can be collected in a matrix $\boldsymbol{\Phi}$ of dimensionality $[N \times m]$, and in a vector $\boldsymbol{y}$ of dimensionality $[N \times 1]$.

Given a specific query point $\boldsymbol{\varphi}_q$, the prediction of the value $y_q = f(\boldsymbol{\varphi}_q)$ is computed as follows. First, for each sample $(\boldsymbol{\varphi}_i, y_i)$ a weight w_i is computed as a function of the distance $d(\boldsymbol{\varphi}_i, \boldsymbol{\varphi}_q)$ from the query point $\boldsymbol{\varphi}_q$ to the point $\boldsymbol{\varphi}_i$. Each row of $\boldsymbol{\Phi}$ and $\boldsymbol{y}$ is then multiplied by the corresponding weight creating the variables $\boldsymbol{Z} = \boldsymbol{W}\boldsymbol{\Phi}$ and $\boldsymbol{v} = \boldsymbol{W}\boldsymbol{y}$, with $\boldsymbol{W}$ diagonal matrix having diagonal elements $\boldsymbol{W}_{ii} = w_i$. Finally, a locally weighted regression model (LWR) is fitted solving the equation $(\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z})\boldsymbol{\beta} = \boldsymbol{Z}^{\mathrm{T}}\boldsymbol{v}$ and the prediction of the value $f(\boldsymbol{\varphi}_q)$ is obtained evaluating such a model in the query point:

$$\hat{y}_q = \boldsymbol{\varphi}_q^{\mathrm{T}}(\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z})^{-1}\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{v}. \tag{4}$$

Typically, the data analyst who adopts a local regression approach, has to take a set of decisions related to the model (e.g. the number of neighbors, the weight function, the parametric family, the fitting criterion to estimate the parameters). Our *lazy learning* method extends the classical approach with a method that automatically selects the adequate configuration.

To this aim, we import tools and techniques from the field of linear statistical analysis. The most important of these tools is the PRESS statistic [39], which is a simple, well-founded and economical way to perform *leave-one-out* cross validation [27] and therefore to assess the performance in generalization of local linear models. This statistic returns the leave-one-out cross-validation error of a linear model at the same

computational cost of the linear regression. As a consequence, the performance of a local memory-based model can be easily assessed with no additional computational burden. It is worth noting that this also means that *lazy learning* can return, along with each predicted value, an estimation of its standard error. This property is more relevant once compared with the intrinsic difficulty of extracting the same information from a generic functional approximator.

In our approach, alternative configurations are assessed and compared in order to select the best one. The same selection strategy is used to select the local weighting function [15], as well as various structural aspects like the set of features and the degree of the local polynomial approximator [12]. The general ideas of the approach can be summarized as follows.

1. The task of learning an input–output mapping is decomposed into a series of local linear approximations, each tailored on a specific query point.
2. Each local approximation is treated as a model selection problem in the space of alternative model configurations.
3. The generalization ability of each candidate model is assessed through cross-validation which is efficiently computed using the PRESS statistic.

In previous works, we proposed two algorithms based on the adoption of recursive techniques for the linear parameter estimation [15] and on a paired permutation test for the comparison of the performances of different candidate models [17].

The method has been applied also to problems of multivariate data analysis [20] and time-series prediction [19].

4. Local control design

We will first define some notations for the single-input–single-output (SISO) case. Consider a class of discrete-time dynamic systems whose equations of motion can be expressed in the form

$$\begin{aligned} y(k) = {} & f(y(k-1),\ldots,y(k-ny),u(k-d),\ldots,\\ & u(k-d-nu),e(k-1),\ldots,e(k-ne))\\ & + e(k), \end{aligned} \tag{5}$$

where k denotes the time, $y(k)$ is the system output, $u(k)$ the input, $e(k)$ is a zero-mean disturbance term, $d>0$ is the relative degree and $f(\cdot)$ is some nonlinear function.

This model is known as the NARMAX model [35]. Let us assume we have no physical description of the function $f(\cdot)$ but a set of pairs $[u(k), y(k)]$ is available. Defining the information vector as

$$\begin{aligned} \boldsymbol{\varphi}(k-1) = [& y(k-1),\ldots,y(k-ny),u(k-d),\ldots,\\ & u(k-d-nu),e(k-1),\ldots,e(k-ne)], \end{aligned} \tag{6}$$

the system (5) can be written in the form

$$y(k) = f(\boldsymbol{\varphi}(k-1)) + e(k). \tag{7}$$

4.1. Neuro-fuzzy and lazy learning for control: a comparative analysis

Although nonlinearity characterizes most real control problems, methods for analysis and control design are considerably more powerful and theoretically better founded for linear systems than for nonlinear ones. In the following, we provide a comparison between the neuro-fuzzy and the *lazy* approach as two ways of extending linear techniques to nonlinear problems.

Neuro-fuzzy: A neuro-fuzzy architecture is a particular example of *local model network* [32,33]. The idea of local model network extends the concept of operating point by introducing the notion of *operating regime*. An operating regime is a set of operating points where the system behavior can be described approximately with a simple model. A validity region, and a local description of the system behavior are associated to each operating regime.

In the neuro-fuzzy formalism the validity region of a local model f^i is represented by the corresponding membership function (1).

Two are the most common approaches in neuro-fuzzy control systems [37]: (i) synthesize a controller for each local model f^i and then combine the control actions with a weighted sum, (ii) interpret the linear TS architecture as a time-varying linear model (Eq. (3)) which returns a linear approximation to the system. The linear approximation is then used to derive the linear controller.

In this paper we will consider only this second formulation, which allows a more direct comparison with the *lazy* approach.

Lazy learning: This approach shares with the neuro-fuzzy approach the idea of decomposing a difficult problem in simpler local problems. Furthermore, both the approaches can return a local linear description of the process (see Eqs. (4) and (3)).

The main difference concerns the model identification procedure. Neuro-fuzzy aims to estimate a global description which covers the whole system operating domain, whereas memory-based techniques focus simply on the current operating point. Neuro-fuzzy is more time-consuming in the identification phase but it is faster in prediction. However, when a new piece of data is observed, model update may require to perform the whole neuro-fuzzy modeling process from scratch. On this matter, *lazy learning* takes an advantage from the absence of a functional approximator: once a new input–output example is observed, it is enough to update the dataset which stores the input–output pairs. *Lazy learning* is, therefore, intrinsically adaptive while neuro-fuzzy requires proper online procedures to deal with sequential problems.

In the following section we will introduce an example of local indirect controller where both approaches can be employed to implement the identification module. This will allow an experimental comparison both on identification and on control simulations.

4.2. The local self-tuning controller

We propose a hybrid architecture for the indirect control of nonlinear discrete-time plants from their observed input–output behavior. An indirect control scheme [5,40] combines an identification module which computes an estimate $\hat{\boldsymbol{\vartheta}}$ of the unknown parameters $\boldsymbol{\vartheta}$, with a control law $u(k) = K(\boldsymbol{\varphi}(k), \boldsymbol{\vartheta})$ implemented as a function of the plant parameters. In the adaptive version, the estimator generates the estimate $\hat{\boldsymbol{\vartheta}}$ at each sampling period k by processing the observed input–output behavior. This estimate is then assumed to be a specification of the real plant and used to compute the control law $u(k) = K(\boldsymbol{\varphi}(k), \hat{\boldsymbol{\vartheta}})$ (certainty equivalence paradigm).

Our approach adopts the local learning identification procedures described in Sections 2 and 3 to apply conventional linear control techniques to nonlinear problems. We propose a local version of the conventional self-tuning regulator (STR) architecture [4]. In STR, identification is performed by a recursive least-squares estimator which updates a linear model each time a new input–output sample is observed. In our approach there is no global linear model description but at each time-step the system dynamics is linearized in the neighborhood of the operating regime. The linearization returns an input–output description of the system dynamics in the form

$$A(z)y(k) = z^{-d}B(z)u(k) + C(z)e(k) + b, \tag{8}$$

where A, B, and C are polynomials in the forward shift operator z and b is an offset term that takes into account configurations which are far from the equilibrium locus [18].

Once the approximation (8) is available, a standard technique like the minimum-variance (MV) method [3] can be used to design the controller. However, such a controller is stable only if the plant is minimum phase, which means that B has at least one of its roots outside the unit circle.

Pole placement (PP) design [5] is an alternative technique to deal with non-minimum-phase configurations. The procedure requires first to choose the desired closed-loop pole positions and then to calculate the appropriate controller.

The proposed control algorithm is described in detail in Fig. 3. Note that with the term *local model* we mean the identification module (neuro-fuzzy or *lazy*) which returns a local approximation to the system dynamics.

4.2.1. Local self-tuning control analysis

In this section we present a stability analysis of the local self-tuning control architecture. In the local self-tuning regulator, the nonlinear plant (7) is parameterized as a linear system where parameters are changing with the observable state. This means that the nonlinear model can be written as a linear model where parameters vary with the state of the system. This configuration reminds the linear parameter varying (LPV) configuration introduced by Shamma and Athans [44] in their analysis of gain scheduling controllers, or the state-dependent models presented by Priestley [41]. These models can be represented as follows:

$$A(\boldsymbol{\varphi}(k))y(k) = z^{-d}B(\boldsymbol{\varphi}(k))u(k) + C(\boldsymbol{\varphi}(k))e(k). \tag{9}$$

Repeat these steps at each sampling period.

1. Acquisition of the vector (6).
2. Linearization of the function $f(\cdot)$ about (6). The linearization is computed by the *local model* algorithm.
3. Derivation of the polynomials A, B, C and the offset b of (8) from the linearized model.
4. Design of a MV/PP controller for (8) which satisfies the required properties (stability, accuracy, speed ...) of the closed loop behavior.
5. Computation of the control signal.
6. Update of the dataset by storing the new input-output observation (only for adaptive *lazy learning*).

Fig. 3. The local self-tuning controller algorithm.

Let us now assume that there exists a LPV model which represents in a sufficiently accurate manner the nonlinear system (7). If we make the hypothesis of complete controllability and observability, the closed-loop system may be put into the state-space form

$$\boldsymbol{x}(k+1) = \boldsymbol{F}(\boldsymbol{\varphi}(k))\boldsymbol{x}(k) + \boldsymbol{v}(k) \tag{10}$$

This representation allows us to analyze the stability of our controller. If the regulator is designed such that the eigenvalues of $\boldsymbol{F}(\boldsymbol{\varphi}(k))$ are stable, then the system (10) will be asymptotically stable for any fixed value of $\boldsymbol{\varphi}$ (frozen time stability). However, this is not a sufficient condition for the uniform asymptotic stability of the system. If we consider the set of values assumed by the matrix $\boldsymbol{F}(\boldsymbol{\varphi}(k))$, a sufficient condition for uniform stability [49] is that it exists a common matrix $\boldsymbol{P} > 0$ such that:

$$\boldsymbol{F}(\boldsymbol{\varphi}(k))^{\mathrm{T}}\boldsymbol{P}\boldsymbol{F}(\boldsymbol{\varphi}(k)) - \boldsymbol{P} < 0 \quad \text{for all } k. \tag{11}$$

With the pole placement technique we can impose the same stable closed loop transfer function for all k. It follows that there exists a matrix $\boldsymbol{P}$ that satisfies Eq. (11). Then, under the following assumptions:

- the system is completely controllable and observable,
- the system (7) can be put in the form (9),
- the approximation error of the *local model* identifier is negligible,

the equilibrium of the nonlinear system (7), controlled by the *local model* PP self-tuning controller, is globally asymptotically stable.

5. Simulation studies

In this experimental study we will consider a nonlinear SISO system described by the difference equation

$$y(k+1) = \frac{y(k)y(k-1)y(k-2)(y(k-2)-1)u(k-1)+u(k)}{1+y^2(k-1)+y^2(k-2)}. \tag{12}$$

The system is represented in the input–output form

$$y(k+1) = f(y(k), y(k-1), y(k-2), u(k), u(k-1)). \tag{13}$$

We assume to have an initial dataset DB of 5000 points collected by exciting the system with a random uniform input (zero mean and unit variance).

In Section 5.1 we will compare the performance of the two local approaches on an identification task, while in Section 5.2 the two approaches will be compared with a conventional linear self-tuning architecture.

5.1. The local identification of a nonlinear discrete-time system

In this simulation we consider the task of predicting the output of system (12), once excited by a random test input $u(k)$ having the same statistical properties of the data in *DB*.

The prediction is done for 500 time steps assuming to have available at each instant k the regression vector $[y(k), y(k-1), y(k-2), u(k), u(k-1)]$. Let us see the performance of the two local approaches.

Neuro-fuzzy. We consider an architecture with triangular membership function and linear consequents. The training dataset *DB* is employed to select the neuro-fuzzy structure having the least generalization error in cross-validation. In Fig. 4 we report the diagram of the cross-validation performance versus the number of inference rules.

We choose $r = 6$ number of rules as the optimal complexity. The model with 6 rules is then estimated on the whole dataset. The plot is Fig. 5a shows the identification error. We obtain an RMSE $= 0.04$ (root mean square error).

Lazy learning. We use the same training dataset *DB*. We adopt the recursive identification method described in [17] to estimate the local model. In spite of the fact that a local model has to be estimated for each prediction, the whole learning process (training, validation and prediction) is largely shorter than in the neuro-fuzzy case (3 min versus 2 days of computation time on the same machine). Also, we are able to obtain a better performance (RMSE $= 0.03$). The plot in Fig. 5b shows the identification error.

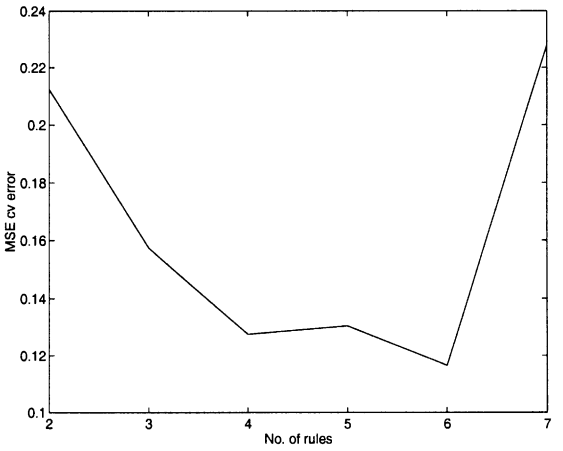

Fig. 4. Neuro-fuzzy structural tuning: number of rules versus MSE (mean square error) in cross-validation.

5.2. The local self-tuning control of a nonlinear discrete-time system

In this simulation, we control the nonlinear system described by Eq. (12). The reference output $y_{\mathrm{ref}}(k)$ is given by a periodic square wave.

We compare three self-tuning control systems where the controller is designed by the pole-placement method described in Section 4.2 and where the identification is performed by a recursive least-squares algorithm, a neuro-fuzzy system and a *lazy learning* algorithm, respectively.

Goal of the experiment is to show that the *lazy learning* approach is intrinsically adaptive and that it requires no particular change to deal with an online update of the input–output dataset.

Let us see the performance of the three approaches.

Linear STR: The linear identification module is first initialized with the parameters returned by the least-squares linearization of the system (13). The least-squares estimation is performed on the basis of the available dataset *DB*. During the simulation, the linear parameters are updated online by a standard recursive least-squares algorithm [30] with forgetting factor $\mu = 0.92$. Reference and the system output are reported in Fig. 6.

Neuro-fuzzy: In order to control the system we adopt the same structure which was identified in the previous section (6 inference rules). The plot in Fig. 7 shows the reference and the system output.

The control system behavior exhibits a steady-state error. A possible explanation could be related to the fact that to follow the reference value the control signal u has to reach values corresponding to regions of the input domain not enough represented in the training set *DB*. The control error is due to the extrapolation error of the neuro-fuzzy model. We remark that no other device is added to the STR controller to compensate this error. In these examples, the goal is simply to compare the two learning approaches in the same control conditions.

Lazy learning: We initialize the *lazy learning* dataset with *DB*. We will present two simulations. In

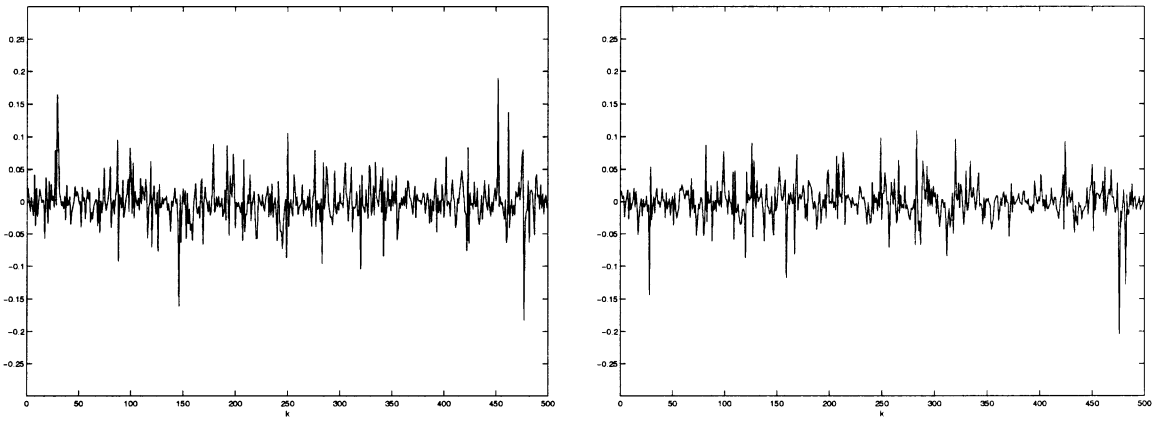

Fig. 5. System identification: (a) neuro-fuzzy: identification error (RMSE = 0.04), (b) lazy learning: identification error (RMSE = 0.03).

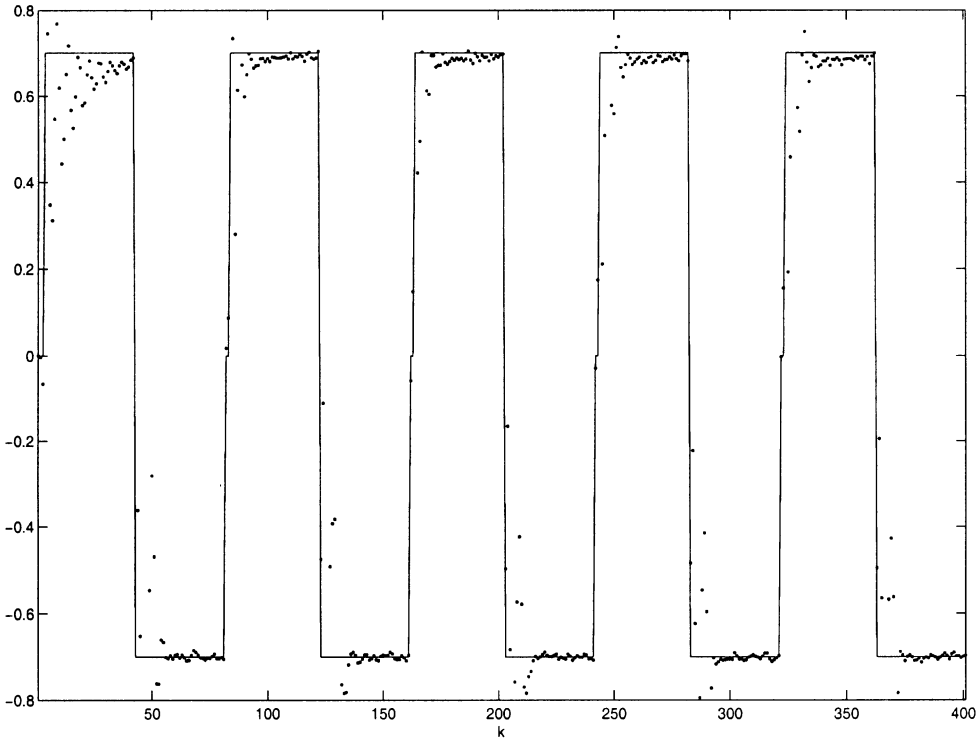

Fig. 6. Linear self-tuning control: reference (solid) and system (dotted) outputs (RMSE = 0.121).

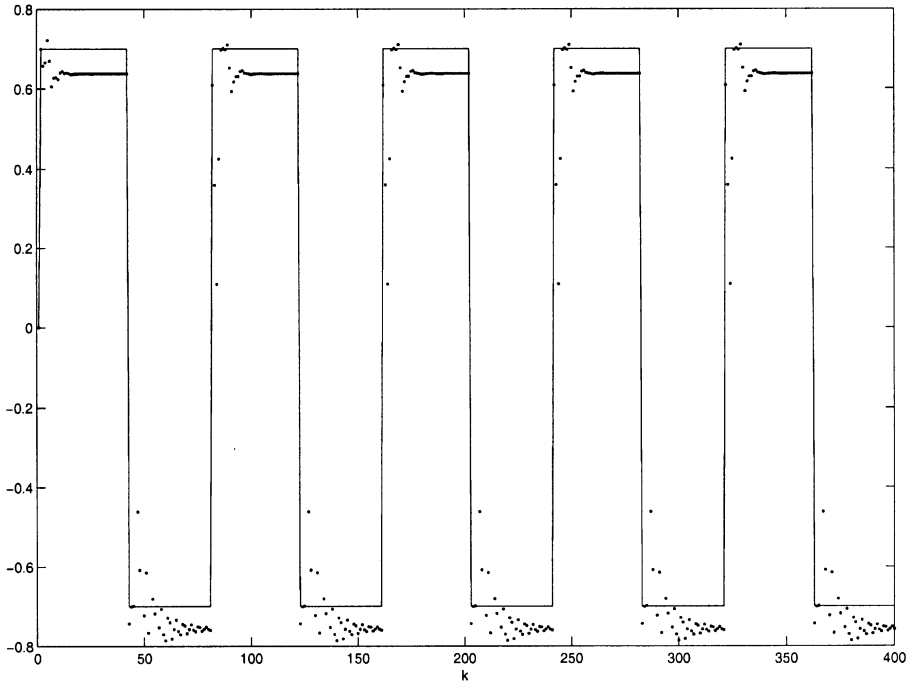

Fig. 7. Neuro-fuzzy self-tuning control: reference (solid) and system (dotted) outputs (RMSE = 0.098).

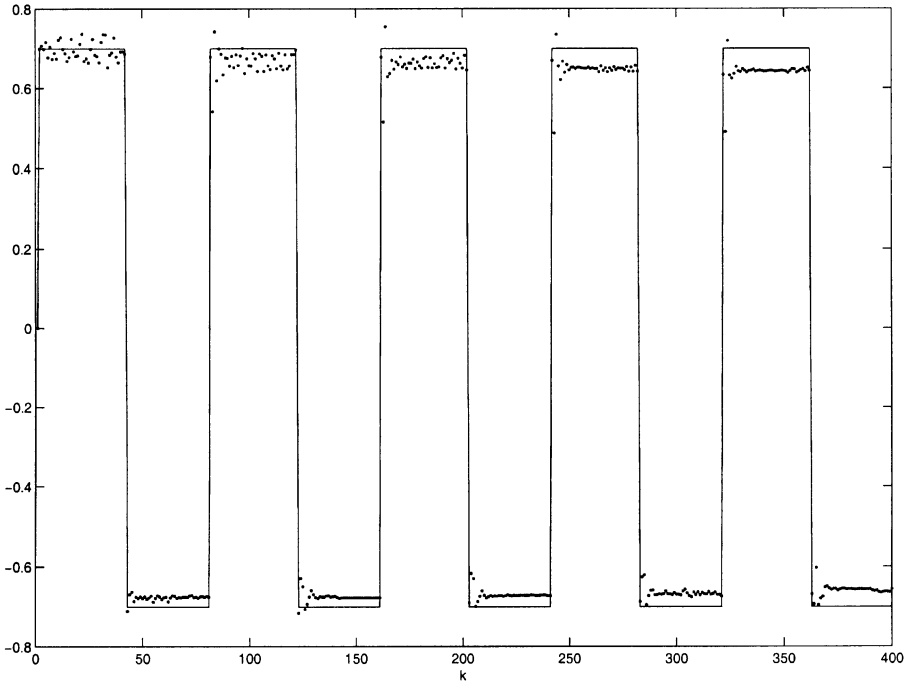

Fig. 8. Lazy learning self-tuning control: reference (solid) and system (dotted) outputs (RMSE = 0.042).

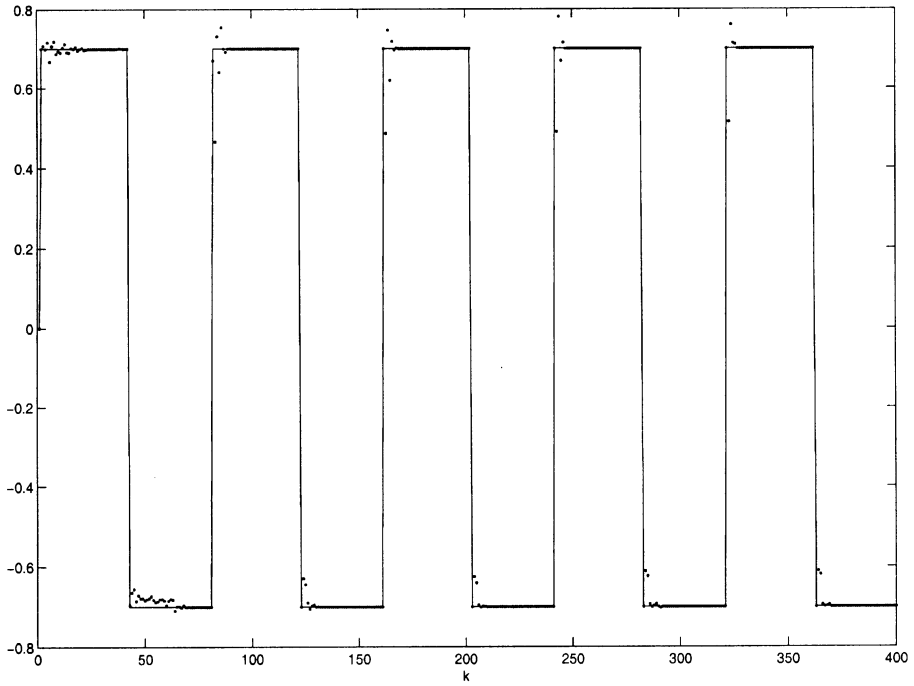

Fig. 9. Lazy learning self-tuning adaptive control: reference (solid) and system (dotted) outputs (RMSE = 0.026).

the first one, the training dataset is kept fixed during the whole simulation (non adaptive case), while in the second one the dataset is updated on-line each time a new input–output pair is returned by the simulated system (adaptive case).

The plot in Fig. 8 shows the reference and the system output in the non-adaptive case while Fig. 9 presents the adaptive case. The *lazy* non-adaptive controller has a better performance than the neuro-fuzzy and the linear one, but the steady-state error persists. This does not happen in the adaptive formulation. It is interesting to see that in this case the *lazy* controller is able to cancel the steady-state error after few simulation steps, compensating to the deficiency of the non-adaptive version.

The experimental results show that the *lazy learning* approach is able to outperform both the classical adaptive approach and the neuro-fuzzy implementation. It is important to remark that the adaptation of the *lazy learning* controller is obtained at the cost of a simple update of the dataset and that it requires no adaptive formulation as it should be the case in a generic nonlinear approach.

6. Is readability compatible with accuracy?

Neuro-fuzzy and *lazy learning* share the *divide-and-conquer* approach: accuracy in modeling is increased by decomposing complex global problems into simpler local sub-problems. At the same time, they are on opposite sides for what concerns the readability of the resulting model.

The idea underlying neuro-fuzzy models is that by hybridizing the adaptivity of neural networks together with the linguistic nature of fuzzy systems, it is possible to synthesize models which are not only accurate but also easy to interpret.

Lazy learning appears to be among the most efficient techniques for adaptive modeling and control, mainly when data are collected in a sequential fashion from a changing external environment. However, it is also one of the least readable techniques since it does not conduct to any form of explicit representation neither in a linguistic form nor in a mathematical one. It rather uses the raw data as the best model, never trying to explicitly capture the underlying analytical structure.

The paper does not aim to present any definitive result about the superiority of one approach over the other. We believe that neuro-fuzzy methods can be profitably applied to real problems where an a priori human knowledge is available. Anyway, a functional multi model approach should not be considered as the only way to exploit the available qualitative information. A future challenge will be to explore alternative ways to integrate *lazy* methods with linguistic knowledge, with the aim of preserving readability without losing efficiency.

References

[1] D.W. Aha, Incremental, instance-based learning of independent and graded concept descriptions, in: Sixth International Machine Learning Workshop, San Mateo, CA, Morgan Kaufmann, 1989, pp. 387–391.

[2] D.W. Aha, Editorial, Artif. Intell. Rev. 11 (1–5) (1997) 1–6.

[3] K.J. Åström, Computer control of a paper machine: an application of linear stochastic control theory, IBM J. Res. Dev. 11 (1967) 389–404.

[4] K.J. Åström, Theory and applications of adaptive control — a survey, Automatica 19 (5) (1983) 471–486.

[5] K.J. Åström, B. Wittenmark, Computer-Controlled Systems: Theory and Design, Prentice-Hall International Editions, Englewood Cliffs, NJ, 1990.

[6] C.G. Atkeson, Memory-based approaches to approximating continuous functions, in: M. Casdagli, S. Eubank (Eds.), Nonlinear Modeling and Forecasting, Addison-Wesley, Reading, MA, 1992, pp. 503–521.

[7] C.G. Atkeson, A.W. Moore, S. Schaal, Locally weighted learning, Artif. Intell. Rev. 11 (1–5) (1997) 11–73.

[8] C.G. Atkeson, A.W. Moore, S. Schaal, Locally weighted learning for control, Artif. Intell. Rev. 11 (1–5) (1997) 75–113.

[9] R. Babuska, Fuzzy modeling and identification, Ph.D. Thesis, Technische Universiteit Delft, 1996.

[10] R. Babuska, H.B. Verbruggen, Fuzzy set methods for local modelling and identification, in: R. Murray-Smith, T.A. Johansen (Eds.), Multiple Model Approaches to Modeling and Control, Taylor and Francis, London, 1997, pp. 75–100.

[11] J.K. Benedetti, On the non parametric estimation of regression functions, J. Roy. Statist. Soc. Ser. B 39 (1977) 248–253.

[12] H. Bersini, M. Birattari, G. Bontempi, Adaptive memory-based regression methods, in: Proceedings of the IEEE International Joint Conference on Neural Networks, 1998, pp. 2102–2106.

[13] H. Bersini, G. Bontempi, Now comes the time to defuzzify the neuro-fuzzy models, Fuzzy Sets and Systems 90 (2) (1997) 161–170.

[14] H. Bersini, G. Bontempi, C. Decaestecker, Comparing rbf and fuzzy inference systems on theoretical and practical basis, in: F. Fogelman-Soulie', P. Gallinari (Eds.), ICANN'95, International Conference on Artificial Neural Networks, 1995, pp. 169–174.

[15] M. Birattari, G. Bontempi, H. Bersini, Lazy learning meets the recursive least-squares algorithm, in: M.S. Kearns, S.A. Solla, D.A. Cohn (Eds.), Advances in Neural Information Processing Systems 11, MIT Press, Cambridge, MA, 1999, pp. 375–381.

[16] G. Bontempi, H. Bersini, Identification of a sensor model with hybrid neuro-fuzzy methods, in: A.B. Bulsari, S. Kallio (Eds.), Neural Networks in Engineering Systems (Proceedings of the 1997 International Conference on Engineering Applications of Neural Networks (EANN '97), Stockholm, Sweden), 1997, pp. 325–328.

[17] G. Bontempi, M. Birattari, H. Bersini, Recursive lazy learning for modeling and control, in: Machine Learning: ECML-98 (10th European Conference on Machine Learning), 1998, pp. 292–303.

[18] G. Bontempi, M. Birattari, H. Bersini, Lazy learning for modeling and control design, Internat. J. Control, 1999, in press.

[19] G. Bontempi, M. Birattari, H. Bersini, Local learning for iterated time series prediction, in: I. Bratko, S. Džeroski (Eds.), Machine Learning: Proceedings of the 16th International Conference, Morgan Kaufman, San Francisco, CA, 1999, pp. 32–38.

[20] G. Bontempi, M. Birattari, H. Bersini, A model selection approach for local learning, Artif. Intell. Comm., 1999, in press.

[21] C. De Boor, A Practical Guide to Splines, Springer, New York, 1978.

[22] L. Bottou, V.N. Vapnik, Local learning algorithms, Neural Computation 4 (6) (1992) 888–900.

[23] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth International Group, Belmont, CA, 1984.

[24] M. Brown, C.J. Harris, Neurofuzzy Adaptive Modelling and Control, Prentice-Hall, Hemel Hempstead, 1994.

[25] W.S. Cleveland, Robust locally weighted regression and smoothing scatterplots, J. Amer. Statist. Assoc. 74 (1979) 829–836.

[26] T. Cover, P. Hart, Nearest neighbor pattern classification, Proc. IEEE Trans. Inform. Theory, 1967, pp. 21–27.

[27] B. Efron, R.J. Tibshirani, An Introduction to the Bookstrap, Chapman & Hall, New York, NY, 1993.

[28] V.A. Epanechnikov, Non parametric estimation of a multivariate probability density, Theory Probab. Appl. 14 (1969) 153–158.

[29] J.D. Farmer, J.J. Sidorowich, Predicting chaotic time series, Phys. Rev. Lett. 8 (59) (1987) 845–848.

[30] G.C. Goodwin, K.S. Sin, Adaptive Filtering Prediction and Control, Prentice-Hall, Englewood Cliffs, NJ, 1984.

[31] J.S.R. Jang, C.T. Sun, E. Mizutani, Neuro-Fuzzy and Soft Computing, Matlab Curriculum Series, Prentice-Hall, Englewood Cliffs, NJ, 1997.

[32] T.A. Johansen, B.A. Foss, Constructing narmax models using armax models, Internat. J. Control 58 (1993) 1125–1153.

[33] T.A. Johansen, B.A. Foss, Semi-empirical modeling of nonlinear dynamic systems through identification of operating

regimes and local models, in: K.J. Hunt, G.R. Irwin, K. Warwick (Eds.), Neural Network Engineering in Dynamic Control Systems, Springer, Berlin, 1995, pp. 105–126.
[34] M.J. Jordan, R.A. Jacobs, Hierarchical mixtures of experts and the em algorithm, Neural Comput. 6 (1994) 181–214.
[35] I.J. Leontaritis, S.A. Billings, Input–output parametric models for non-linear systems, Internat. J. Control 41 (2) (1985) 303–344.
[36] J. Moody, C.J. Darken, Fast learning in networks of locally-tuned processing units, Neural Comput. 1 (2) (1989) 281–294.
[37] R. Murray-Smith, K. Hunt, Local model architectures for nonlinear modelling and control, in: K.J. Hunt, G.R. Irwin, K. Warwick (Eds.), Neural Network Engineering in Dynamic Control Systems, Springer, Berlin, 1995, pp. 61–82.
[38] R. Murray-Smith, T.A. Johansen, Multiple Model Approaches to Modelling and Control, Taylor and Francis, London, 1997.
[39] R.H. Myers, Classical and Modern Regression with Applications, 2nd Edition, PWS-KENT Publishing Company, Boston, MA, 1994.
[40] K.S. Narendra, A.M. Annaswamy, Stable Adaptive Systems, Prentice-Hall, Englewood Cliffs, NJ, 1989.
[41] M.B. Priestley, Non-linear and Non-Stationary Time Series Analysis, Academic Press, New York, 1988.
[42] D.E. Rumelhart, G.E. Hinton, R.K. Williams, Learning representations by backpropagating errors, Nature 323 (9) (1986) 533–536.
[43] S. Schaal, C.G. Atkeson, Robot juggling: An implementation of memory-based learning, IEEE Control Systems 14 (1) (1994) 57–71.
[44] J.S. Shamma, M. Athans, Gain scheduling: potential hazards and possible remedies, IEEE Control Systems Mag. (1992) 101–107.
[45] A. Stenman, F. Gustafsson, L. Ljung, Just in time models for dynamical systems, in: 35th IEEE Conference on Decision and Control, Kobe, Japan, 1996, pp. 1115–1120.
[46] M. Stone, Cross-validatory choice and assessment of statistical predictions, J. Roy. Statist. Soc. Ser. B 36 (1) (1974) 111–147.
[47] M. Sugeno, G.T. Kang, Structure identification of fuzzy model, Fuzzy Sets and Systems 28 (1988) 15–33.
[48] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. System, Man Cybernet. 15 (1) (1985) 116–132.
[49] K. Tanaka, M. Sugeno, Stability analysis and design of fuzzy control systems, Fuzzy Sets and Systems 45 (1992) 135–156.
[50] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.