ELSEVIER

Fuzzy Sets and Systems 128 (2002) 3–14

FUZZY sets and systems

www.elsevier.com/locate/fss

Data-driven techniques for direct adaptive control: the lazy and the fuzzy approaches

Edy Bertolissi, Mauro Birattari, Gianluca Bontempi *, Antoine Duchâteau, Hugues Bersini

IRIDIA, Université Libre de Bruxelles 50, av. Franklin Roosevelt, 1050 Brussels, Belgium

Abstract

This paper presents an approach to modeling and controlling discrete-time non-linear dynamical system on the basis of a finite amount of input/output observations. The controller consists of a multiple-step-ahead direct adaptive controller which, at each time step, first performs a forward simulation of the closed-loop system and then makes an adaptation of the parameters of the controller. This procedure requires a sufficiently accurate model of the process in order to meet the control requirements. Takagi–Sugeno fuzzy systems and *Lazy Learning* are two approaches which have been proposed in control literature as effective ways of identifying a plant. This paper compares these two approaches in two main configurations: (i) when the number of observations is fixed and (ii) when new observations are collected on-line after each control action. Simulation examples of the control of the manifold pressure of a car engine are given. © 2002 Published by Elsevier Science B.V.

Keywords: Direct adaptive control; Lazy learning; Takagi–Sugeno fuzzy systems

1. Introduction

The challenging task of modeling real phenomena on the basis of a limited set of observations has been addressed by several disciplines, ranging from nonlinear regression to machine learning and system identification. In the data-driven modeling community, two main paradigms have emerged: global versus *divide and conquer*.

Global modeling builds a single functional model on the basis of the dataset. This is the traditional approach used in linear method [14], neural networks [23], and other forms of nonlinear statistical regression [21]. The available dataset is used by a learning algorithm to produce a model of the mapping, then the dataset is discarded and only the functional description is kept.

Divide and conquer techniques, also known as multiple model approaches [19], partition a complex problem into simpler ones whose solutions can be combined to provide a solution of the original problem. Examples of these approaches are *modular* architectures, like Radial Basis Functions [17], Local Model Networks [18] or Takagi–Sugeno fuzzy systems [24], where different modules are composed in order to cover the input space. Although these approaches propose a combination of local models, the final outcome is again a functional description of the

* Corresponding author.
E-mail addresses: eberto@iridia.ulb.ac.be (E. Bertolissi), mbiro@ulb.ac.be (M. Birattari), gbonte@ulb.ac.be (G. Bontempi), aduchate@iridia.ulb.ac.be (A. Duchâteau), bersini@ulb.ac.be (H. Bersini).

process over the whole input domain. For this reason, global procedures are still required in the structural identification step, e.g. for the assessment and selection of the optimal complexity of the approximator.

Another example of multiple model method is the so-called *local modeling* technique [12], where the problem of function estimation is transformed into one of value estimation. In these approaches the goal is not to find a model which explains the whole process, but to find the best output for a specific given input (called *query*). Local techniques renounce a complete description of the input–output relation, and aim at approximating the function only in the neighborhood of the point to be predicted. The objective of these techniques is to improve the prediction accuracy at the cost of a reduced readability of the resulting model. Since an approximation function is never explicitly estimated, it is necessary to keep in memory the whole dataset for each prediction, and therefore the quantity of memory required for these approaches is much larger than the one necessary in the other cases. *Lazy learning* [1] and nearest neighbor are examples of local modeling techniques.

The goal of this paper is to compare different divide and conquer techniques in the task of modeling an unknown dynamic system in the framework of direct adaptive control (DAC) [2]. We consider the case in which only a limited amount of process data is initially available, and further examples may be collected on-line. In the DAC architecture, at each time step a forward simulation of the system composed by the controller and plant pair is performed. The results of this computation are then used to tune the controller by adapting its parameters. This control technique requires an accurate identification of the plant since the quality of the model affects the final performance of the closed-loop system.

In particular here we compare two instances of DAC architecture. The first one adopts a Takagi–Sugeno fuzzy system as a model of the plant, while the second is based on the *lazy learning* technique.

Another comparison between fuzzy systems and lazy learning as identification modules in a nonlinear control architecture has been discussed in Bontempi et al. [10] and in Bontempi [9]. In that work the authors focused on a self-tuning regulator (STR) [3], and compared the two approaches on an artificial benchmark. Here, we consider a DAC architecture and we simulate the control of a real process, namely, an engine model implemented by Siemens Automotive and proposed within the FAMIMO project [1] (ESPRIT LTR Project 21911). It is interesting to notice that the experimental results obtained here appear to be consistent with the ones presented in the previous work.

2. Multiple modeling with Takagi–Sugeno fuzzy systems and lazy learning

In Takagi–Sugeno (TS) fuzzy systems, an input/output relation $F:\Re^n \to \Re$ is modeled as a fuzzy interpolation of a number of linear systems which approximate the desired function in local regions described by membership functions. These systems are typically represented by a set of fuzzy rules which partition the input space:

$$\begin{aligned}\mathscr{R}^{(j)}\colon\ &\text{IF } \varphi_1 \text{ IS } A_1^{(j)} \text{ AND} \ldots \text{AND } \varphi_n \text{ IS } A_n^{(j)}\\ &\text{THEN } y = a_0^{(j)} + a_1^{(j)}\varphi_1 + \cdots + a_n^{(j)}\varphi_n.\end{aligned}$$

The system output is a weighted average of the individual rule outputs:

$$y = \sum_{j=1}^{M} \frac{\mu_{A^{(j)}}(\boldsymbol{\varphi})\,(a_0^{(j)} + a_1^{(j)}\varphi_1 + \cdots + a_n^{(j)}\varphi_n)}{\sum_{k=1}^{M} \mu_{A^{(k)}}(\boldsymbol{\varphi})}, \tag{1}$$

where the weights $\mu_{A^{(j)}}(\boldsymbol{\varphi})$ are computed according to

$$\mu_{A^{(j)}}(\boldsymbol{\varphi}) = \prod_{i=1}^{n} \mu_{A_i^{(j)}}(\varphi_i).$$

This approach allows us to model a system by means of the decomposition of a nonlinear mapping into a collection of local linear models. Since this approach proposes to structure the problem in a series of local models, Takagi–Sugeno models can be easily constructed from numerical data, which are used to identify the number and the attributes of the fuzzy rules.

In the *Lazy Learning* (LL) approach, the estimation of the value of the unknown function is performed giving the whole attention to the region surrounding the point where the estimation is required.

[1] `http://iridia.ulb.ac.be/~famimo/`.

Let us consider an unknown mapping $f : \Re^n \rightarrow \Re$ of which we are given a set of N samples $\{(\boldsymbol{\varphi}_{[i]}, y_{[i]})\}_{i=1}^{N}$. These examples can be collected in a matrix $\boldsymbol{\Phi}$ of dimensionality $[N \times m]$, and in a vector $\mathbf{y}$ of dimensionality $[N \times 1]$.

Given a specific query point $\boldsymbol{\varphi}_{\mathrm{q}}$, the prediction of the value $y_{\mathrm{q}} = f(\boldsymbol{\varphi}_{\mathrm{q}})$ 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 $d(\boldsymbol{\varphi}_{[i]}, \boldsymbol{\varphi}_{\mathrm{q}})$ from the query point $\boldsymbol{\varphi}_{\mathrm{q}}$ to the point $\boldsymbol{\varphi}_{[i]}$. Each row of $\boldsymbol{\Phi}$ and $\mathbf{y}$ is then multiplied by the corresponding weight creating the variables $\mathbf{Z} = \mathbf{W}\boldsymbol{\Phi}$ and $\mathbf{v} = \mathbf{W}\mathbf{y}$, with $\mathbf{W}$ diagonal matrix having diagonal elements $\mathbf{W}_{ii} = w_i$. Finally, a locally weighted regression model (LWR) is fitted solving the equation $(\mathbf{Z}^{\mathrm{T}}\mathbf{Z})\boldsymbol{\beta} = \mathbf{Z}^{\mathrm{T}}\mathbf{v}$ and the prediction of the value $f(\boldsymbol{\varphi}_{\mathrm{q}})$ is obtained evaluating such a model in the query point:

$$\hat{y}_{\mathrm{q}} = \boldsymbol{\varphi}_{\mathrm{q}}^{\mathrm{T}}(\mathbf{Z}^{\mathrm{T}}\mathbf{Z})^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{v}. \tag{2}$$

Here, we will focus mainly on the procedural aspects of the modeling technique. 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). In this paper we take advantage of the method described in Birattari et al. [8], which automatically selects, for each query point, the adequate configuration. This is done by importing tools and techniques from the field of linear statistical analysis. The most important of these tools is the PRESS statistic [20], which is a simple, well-founded and economical way to perform *leave-one-out* cross-validation [15] and therefore to assess the performance in generalization of local linear models. Due to its short computation time which allows its intensive use, it is the key element of the *lazy learning* approach to modeling data. In this modeling procedure the performance of a model in cross-validation is the criterion adopted to choose the best local model configuration [8]. One of the most important parameters to be tuned in a local model configuration is the size of the region surrounding $\boldsymbol{\varphi}_{\mathrm{q}}$, in which the function $f(\cdot)$ can be conveniently approximated by a linear local model. Such a parameter can be related to the number of training examples which fall into the region of linearity. The task of identifying the region of linearity is therefore akin to the task of finding, among the examples available, the number k of neighbors of $\boldsymbol{\varphi}_{\mathrm{q}}$ to be used in the local regression fit. Thus, different models are considered, each fitted on a different number of examples, and the *leave-one-out* cross-validation is used to compare them and to select the one for which the predicted error is smaller. To make the procedure faster and to avoid repeating for each model the parameter and the PRESS computation, an incremental approach based on recursive linear techniques is adopted [8].

The reader interested in an analysis of local methods in terms of approximation and convergence properties should refer to Cybenko [13].

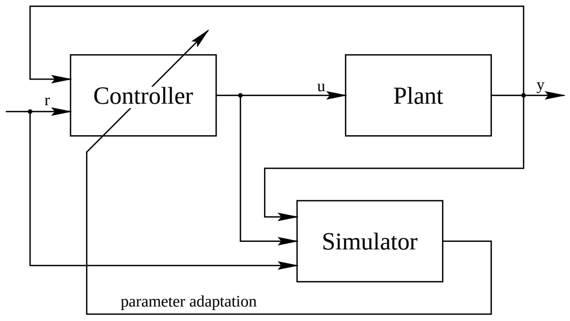

Fig. 1. Schematic representation of the system.

3. System description

Fig. 1 shows the structure of a closed-loop control system which is composed of the plant to be controlled, the simulator, and the controller.

Assume that the n-input–m-output plant is expressed in terms of its input–output representation:

$$\begin{aligned} y_i(k+1) = F_i(&\mathbf{y}(k), \ldots, \mathbf{y}(k-p_y+1), \\ &\mathbf{u}(k), \ldots, \mathbf{u}(k-p_u+1)), \\ &i = 1, \ldots, m, \end{aligned} \tag{3}$$

where the scalar $y_i(k)$ is the ith output of the plant at time k, $\mathbf{u}(k)$ is the input vector $[u_1(k), \ldots, u_n(k)]^{\mathrm{T}}$ and $\mathbf{y}(k)$ is the output vector $[y_1(k), \ldots, y_m(k)]^{\mathrm{T}}$, F_i is an unknown nonlinear function, and p_y and p_u are the known structure orders of the system for the output i.

If the regressor vector $\phi(k)$ is defined as follows:

$$\phi(k) = [\mathbf{y}(k), \dots, \mathbf{y}(k - p_y + 1),$$
$$\mathbf{u}(k-1), \dots, \mathbf{u}(k - p_u + 1)], \tag{4}$$

Eq. (3) can be rewritten as

$$y_i(k+1) = F_i(\phi(k), \mathbf{u}(k)), \quad i = 1, \dots, m \tag{5}$$

which in vector notation becomes

$$\mathbf{y}(k+1) = \mathbf{F}(\phi(k), \mathbf{u}(k)). \tag{6}$$

The control action $\mathbf{u}(k)$ is fed into the plant:

$$\mathbf{u}(k) = \mathscr{F}(\mathbf{r}(k), \psi(k); \mathbf{w}), \tag{7}$$

where $\psi(k)$ is a regressor vector obtained from time delayed values of the inputs and of the outputs, $\mathbf{w}$ is the set of parameters describing the controller and $\mathbf{r}(k)$ is the vector of the reference signals $[r_1(k), r_2(k), \dots, r_m(k)]^{\mathrm{T}}$. The controller will produce a control action $\mathbf{u}(k)$ which will drive the plant outputs $\mathbf{y}(k+1)$ at the values specified by the vector $\mathbf{r}(k)$. Since the dynamics of the controller is much faster than the one of the plant to be controlled it is supposed that the control action at time k is influenced by the output of the plant at time k, without any delay.

The predictor performs a forward simulation of the system composed by the plant and the controller. Each predicted output of the plant

$$\hat{y}_i(k+1) = \hat{F}_i(\phi(k), \mathbf{u}(k)), \quad i = 1, \dots, m, \tag{8}$$

where $\hat{F}_i$ is the estimate of F_i, learned on the basis of the available dataset. The results of the forward simulation are then used to perform the parametric adaptation of the controller.

4. Multiple-step-ahead adaptive control

Using the structure of the system depicted in Fig. 1 it is possible to design a learning algorithm, based on the principles defined in generalized predictive control theory [11], which performs an adaptation of the weights of the controller using information about the future behavior of the system. The predictor will provide the controller with information regarding the future values of $\hat{\mathbf{y}}$ and $\mathbf{u}$ up to the prediction horizon. It is necessary to make the following assumptions:

(1) The state of the process at any time can be reconstructed using the information inside the regressor $\phi(k)$;
(2) A unique series of inputs $[\mathbf{u}(k), \mathbf{u}(k+1), \dots, \mathbf{u}(k+H_{\mathrm{c}})]$ exists, which leads the output signals $\mathbf{y}$ to $\mathbf{r}$ at time $k + H_{\mathrm{p}}$. H_{c} is the control horizon (the length of the time horizon where a control signal is applied) H_{p} is the prediction horizon (the length of the time horizon where the future states of system are simulated). The prediction horizon must be bigger or equal to the control horizon ($H_{\mathrm{p}} \geqslant H_{\mathrm{c}}$). Control actions are considered constant once the control horizon is reached: $\mathbf{u}(t) = \mathbf{u}(k + H_{\mathrm{c}})$ for $t > k + H_{\mathrm{c}}$. Note that this assumption is required in order to guarantee the convergence of the gradient-based method illustrated in the following;
(3) Every $\partial F(\cdots)/\partial \mathbf{u}(t)$ for $t < k$ is equal to 0 since past actions are considered constant;
(4) The controller $\mathscr{F}(\mathbf{r}(k), \psi(k); \mathbf{w})$ can approximate the series of perfect control actions $[\mathbf{u}(k), \mathbf{u}(k+1), \dots, \mathbf{u}(k + H_{\mathrm{c}})]$ to any degree of accuracy in the region of interest for some "perfectly tuned" weights $\mathbf{w} = \mathbf{w}^{\star}$;
(5) The speed of adaptation of the weights is low in order to be able to separate, in the measurement of the error, the effects of the parameters adjustment from the input signal variations [16].

These assumptions allow the design of an adaptation algorithm based on the gradient descent:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \eta \frac{\partial J}{\partial \mathbf{w}}. \tag{9}$$

In order to train the parameters $\mathbf{w}$ the following cost function is selected:

$$J = \frac{1}{2} \sum_{t=k}^{k+H_{\mathrm{p}}} (\mathbf{r}(t) - \hat{\mathbf{y}}(t))^{\mathrm{T}} Q (\mathbf{r}(t) - \hat{\mathbf{y}}(t))$$
$$+ \frac{1}{2} \sum_{t=k-1}^{k+H_{\mathrm{c}}} \Delta\mathbf{u}(t)^{\mathrm{T}} R \Delta\mathbf{u}(t), \tag{10}$$

where $\Delta\mathbf{u}(t) = \mathbf{u}(t) - \mathbf{u}(t-1)$ and $\mathbf{u}(k-2) = 0$. The matrix $Q \in \Re^{n \times n}$ weights the errors $(\mathbf{r}(t) - \hat{\mathbf{y}}(t))$ while the matrix $R \in \Re^{m\times m}$ has the effect of penalizing the large variations of $\Delta\mathbf{u}(t)$ which could destabilize the system.

Substituting this expression into Eq. (9) and recalling Eq. (6) it is possible to obtain

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \eta\left(\sum_{t=k}^{k+H_p}(\mathbf{r}(t)-\hat{\mathbf{y}}(t))^{\mathrm{T}}Q\frac{\partial\hat{\mathbf{y}}(t)}{\partial\mathbf{w}} + \sum_{t=k-1}^{k+H_c}\Delta\mathbf{u}(t)^{\mathrm{T}}R\frac{\partial\Delta\mathbf{u}(t)}{\partial\mathbf{w}}\right). \tag{11}$$

This expression can be calculated by recursively computing the following two expressions:

$$\frac{\partial\mathbf{u}(t-1)}{\partial\mathbf{w}} = \frac{\partial\mathscr{F}(\mathbf{r}(t-1),\psi(t-1);\mathbf{w})}{\partial\mathbf{w}} = \frac{\partial\mathscr{F}(\cdots)}{\partial\mathbf{w}} + \frac{\partial\mathscr{F}(\cdots)}{\partial\psi(t)}\frac{\partial\psi(t)}{\partial\mathbf{w}} \tag{12}$$

and

$$\frac{\partial\hat{\mathbf{y}}(t)}{\partial\mathbf{w}} = \frac{\partial\hat{\mathbf{F}}(\mathbf{u}(t-1),\phi(t-1))}{\partial\mathbf{w}} = \frac{\partial\hat{\mathbf{F}}(\cdots)}{\partial\phi(t-1)}\frac{\partial\phi(t-1)}{\partial\mathbf{w}} + \frac{\partial\hat{\mathbf{F}}(\cdots)}{\partial\mathbf{u}(t-1)} \times\left(\frac{\partial\mathscr{F}(\cdots)}{\partial\mathbf{w}} + \frac{\partial\mathscr{F}(\cdots)}{\partial\psi(t-1)}\frac{\partial\psi(t-1)}{\partial\mathbf{w}}\right). \tag{13}$$

The full description of this algorithm can be found in Bertolissi et al. [6].

The multiple step algorithm is a quite complex algorithm. It allows the control of systems which are not minimum phase and any desired output does not need to be reachable in one step, as happens in the case of the single-step-ahead versions of the same approach [22]. This comes from the fact that instead of looking one step in the future in order to select a control policy, we consider a longer time horizon. This approach also implies that the learning algorithm can now learn on the basis of a series of control actions instead of only one control action in order to reach the desired system output.

The relaxation of these assumptions has two costs: a higher computational load and the need for a more precise model. The first cost is easily understandable and is due to the simulation at each step of the closed-loop behavior over a long time horizon. The second cost comes from the fact that the long term predictions are more difficult to achieve and need better precision, at each step, in order to avoid accumulation of the errors.

The quality of the model is therefore fundamental for the implementation of a good controller. Takagi–Sugeno fuzzy systems and *lazy learning* can both be used to produce a model of the plant to be controlled.

However, in all the cases when the dynamics of the controller plant cannot be completely defined from the available training data, or its dynamics is time variant, it is useful to add adaptation capabilities to the model of the plant in order to improve the performance of the system.

In the case of fuzzy systems, when new data become available a new identification procedure is usually necessary to incorporate this new knowledge in the model. An approach which allows the implementation of on-line learning consists in adapting only the linear consequents of the fuzzy rules, without modifying the position of the centers and the shape of the fuzzy rules. Given

$$y^j(\boldsymbol{\varphi}) = a_0^{(j)} + a_1^{(j)}\varphi_1 + \cdots + a_n^{(j)}\varphi_n = l^{(j)}\boldsymbol{\varphi}$$

and

$$A^{(j)} = \frac{\mu_{A^{(j)}}(\boldsymbol{\varphi})}{\sum_{k=1}^{M}\mu_{A^{(k)}}(\boldsymbol{\varphi})}$$

Eq. (1) can be rewritten as

$$y = \sum_{j=1}^{M}A^{(j)}y^j(\boldsymbol{\varphi}) = \sum_{j=1}^{M}A^{(j)}l^{(j)}\boldsymbol{\varphi} = \boldsymbol{L}^{\mathrm{T}}\boldsymbol{\Phi},$$

where $\boldsymbol{L}^{\mathrm{T}} = [l^{(1)}\dots l^{(M)}]$ and $\boldsymbol{\Phi} = [A^{(1)}\boldsymbol{\varphi}\dots A^{(M)}\boldsymbol{\varphi}]^{\mathrm{T}}$. This is a linear system whose parameters $\boldsymbol{L}$ can be adapted using a a recursive least mean square algorithm.

This adaptation procedure is not equivalent to a new identification procedure, since it affects only some of the parameters which are used to describe the fuzzy model. This means that if new examples bring information on a dynamics which cannot be characterized by the known fuzzy rules, the fuzzy model will change the parameters of the linear consequents trying to model the new behavior. A new identification, on the other hand, could modify also the position and the shape of some fuzzy rules in addition to the linear consequents, and it could even lead to the definition of new rules in order to capture the richer dynamics. A new identification would be too expensive from a computational point of view. It is worth noticing here that the adaptation of the consequents is a linear procedure while the adaptation of the positions and of the shapes of the rules would involve a nonlinear procedure.

In the case of *lazy learning* adaptation capabilities can be implemented simply by adding the new data, as they becomes available, to the database of examples which will be used to calculate the subsequent predictions. This means that the *lazy learning* simulator can be easily updated on-line, while the controller is running. Moreover, the addition of new points associated with unmodeled dynamics does not change the prediction accuracy of the model in other areas.

5. Simulation studies

Due to the constraints concerning pollutant emissions, consumption and efficiency, car manufacturers are currently studying new concepts for engines. At present several efforts are focused on direct injection engines (GDI). This novel kind of engine allows two operation modes, the usual one called homogeneous mode, and a new one called stratified mode, with lean air fuel mixtures. In this way a significant decrease of consumption and pollutant emission is expected. Engineers should tune and control several parameters in order to control the engine to achieve the maximum performance. One of these is the fresh air quantity introduced into the cylinders through an electrical throttle which controls the manifold pressure.

In this paper we propose, as an example of a real process, the control of the manifold pressure of a GDI engine. The full engine model has been implemented by Siemens Automotive and it is of one of the benchmarks defined within the FAMIMO project (ESPRIT LTR Project 21911). The aim of the controller is to set the position of the electrical throttle, $MTC \in [0\ 100]$, which determines the manifold pressure $pman \in [100\ 1024]$. The pressure also depends on the engine speed $N \in [900\ 3500]$, which is measured in revolutions per minute (rpm) and is a noncontrollable input of the system. The process is a second-order nonlinear system.

Two sets of experiments have been performed. The first set of experiments assessed the control capabilities of the simulator based on the fuzzy and lazy models without adaptation capabilities. These experiments have been carried out in noisy and noise-free scenarios, as defined in the engine benchmark. The second set of experiences was targeted to the evaluation of the performance of the controller in the same scenarios, when adaptation capabilities were added to the models of the plant. In all the experiments the direct adaptive controller had to follow a pseudo-sinusoidal signal which pushes the plant in areas where its dynamics are not well described by the training data.

Since the aim of the paper is to show that it is possible to use a *lazy learning* approximator when an insufficient number of data are available for the modeling of the process, the plant has been identified using 5000 points collected by exciting the system with a pseudo-random input. In this way the data provides information only on part of the system dynamics and therefore cannot be used to obtain a reliable model. The training data used for the identification procedures is displayed in Fig. 2.

As far as the noise is concerned, the benchmark introduced an additive disturbance at the level of the sensor of the manifold pressure. The readings of the pressure were disturbed by two different noise components: a first proportional component which consisted of a uniform distributed noise bounded within 2% of the level of the signal, and a second uniform distributed component which was defined within ± 30 mbar. As far as the lazy learning simulator is concerned, linear local models have been considered where the number of neighbors vary in the range between 30 and 60. In addition, the prediction for each query has been obtained as a combination of the 6 best local models, according to the PRESS estimate, out of those considered in the above mentioned range [8]. As far as the Takagi–Sugeno fuzzy system is concerned, the identification procedure that has been used [4] automatically generates a fuzzy partition of the process input space by letting grow the number of fuzzy rules until an optimum is achieved

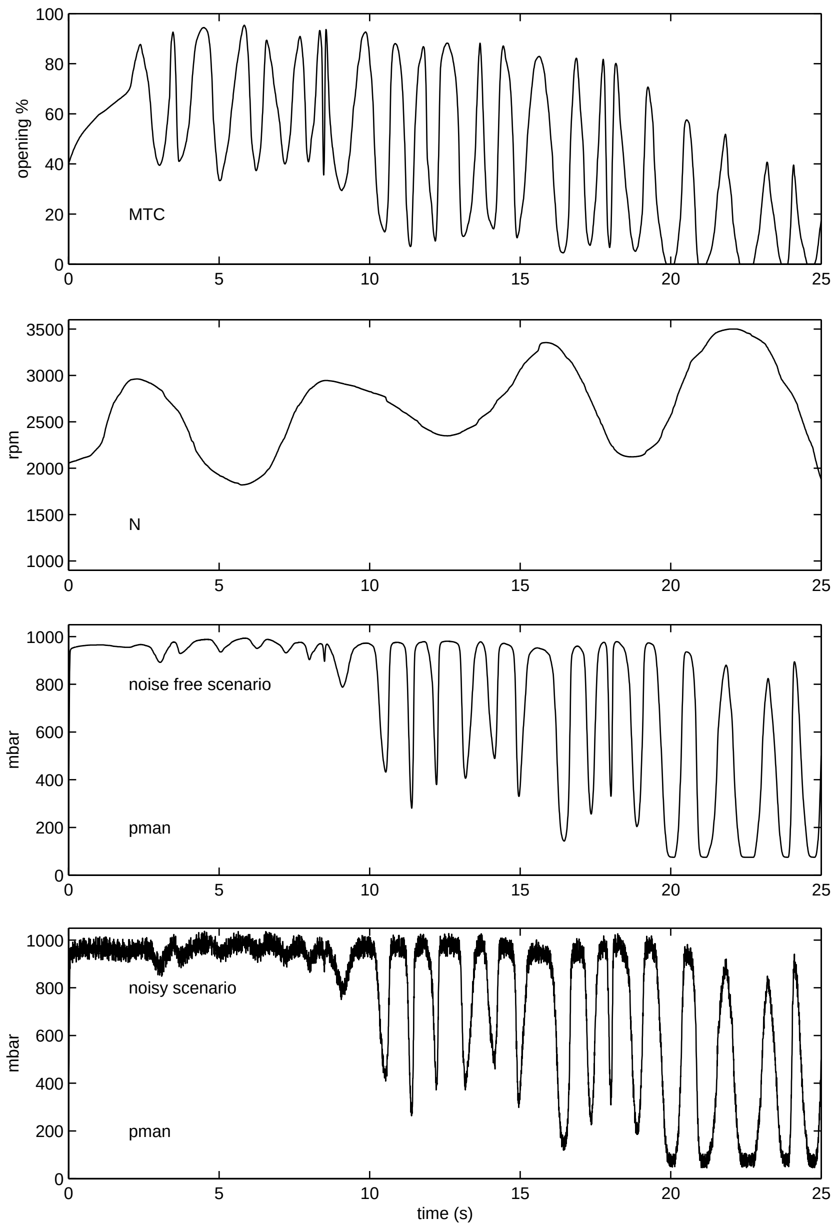

Fig. 2. Training sequences used to identify the model of the manifold pressure. *MTC* represents the control action, *N* is a noncontrollable input of the plant, while *pman* represents the output of the plant. The noise-free and noisy outputs of the plant are shown in the bottom graphs. The signals have been sampled for 25 s with a period of 0.005, thus collecting 5000 samples.

with respect to the given performance criteria. In this particular case 14 rules were defined for describing the dynamical system in the noise-free scenario, while only 8 in the noisy scenario.

In this particular experience the controller itself has been implemented as a Takagi–Sugeno fuzzy system. It is worth noting that any other (differentiable) parametric controller could have been used as well and

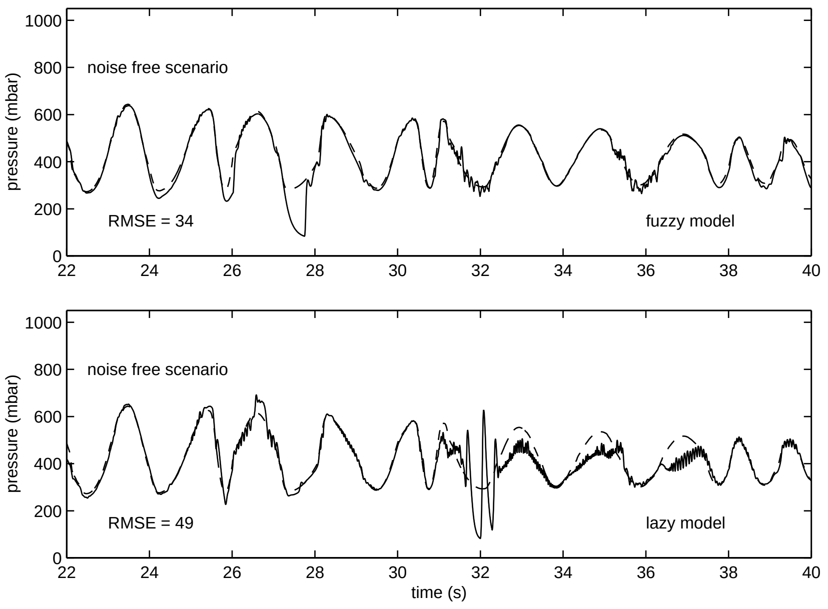

Fig. 3. Performance of the controller based on a fuzzy model, and a lazy model in a noise-free environment. The solid line is the actual output of the plant, while the reference is represented by the dashed line. The performance is measured in terms of the root mean squared error (RMSE).

that there is no correlation between the choice of a particular controller and the choice of the type of simulator. The control action is the position of the throttle (MTC), and it is calculated on the basis of the current value of the manifold pressure (pman), the desired value of the manifold pressure (pman_d), and the number of revolutions per minute of the engine (N). The sampling time of the system is equal to 5 ms. The controller is described by 108 Takagi–Sugeno fuzzy rules, obtained by evenly distributing the centers of the rules on the four-dimensional input space ($108 = 3 \times 3 \times 3 \times 4$). In the beginning the consequences of all the rules have been initialized to 0. The adaptation algorithm uses a 5-step-ahead prediction horizon and a learning rate $\eta = 10^{-8}$.

The graphs of Fig. 3 show the performance of the controller (solid line) in the temporal window between 22 and 40 s in the noise-free scenario. In the previous temporal window between 0 and 22 s, the controller has been following the reference signal in order to initialize the rules of the controller.

The graphs show the performance of the fuzzy and lazy models built starting from the initial 5000 input–output pairs. It is possible to notice that in both cases the controller shows a poor performance in terms of the observed root mean squared error (RMSE). The output of controller based on the fuzzy model of the plant presents a smoother response, and both the controllers show deviations from the setpoint on some occasions. In particular it is possible to notice that both controllers tend to show deficiencies in the same areas, but the output of the controller based on the fuzzy model is smoother. This means that the prediction and the values of the derivatives in the lazy model are less reliable, leading the system to exhibit high frequency components. The smoother performance of the controller based on the fuzzy model may be caused by the fact that when the estimation of the model is done in an area where limited information is available, the fuzzy system offers better extrapolation abilities. In the case of the lazy simulator the use of few points for calculating the same quantity leads to poor values for the derivatives.

The graphs of Fig. 4 show the performance of the controller in the noisy scenario.

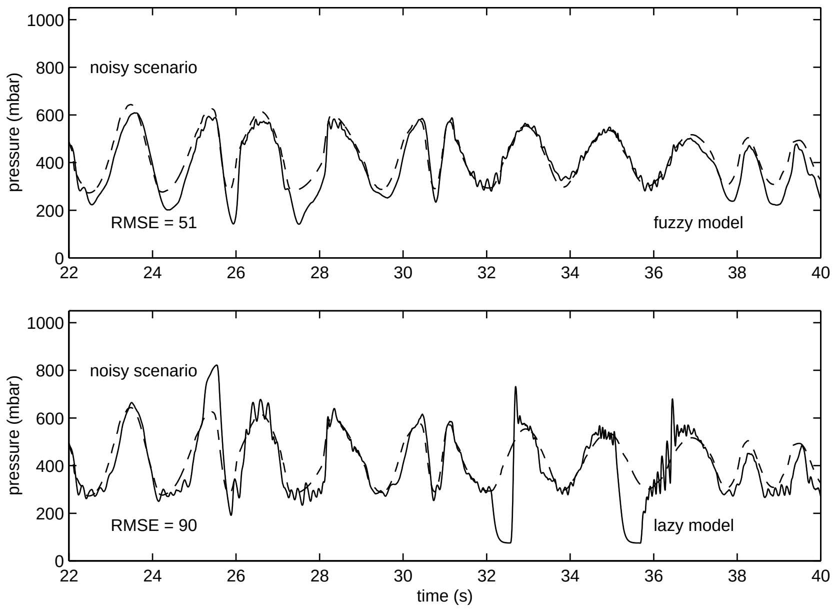

Fig. 4. Performance of the controller based on a fuzzy model, and a lazy model in a noisy environment. The solid line is the actual output of the plant, while the reference is represented by the dashed line. The output of the controller is measured before addition of the sensor noise, and gives the real value of the pressure at the manifold level.

The graphs show the readings of the pressure before the addition of the sensor noise, and therefore they give its real value at the manifold level. It is possible to see that the controller shows a poorer performance in comparison with the noise-free scenario. In this case, the controller based on the fuzzy model manages to follow the set point, the one based on the lazy learning shows deficiencies in several situations. Even in the noisy scenario it is possible to notice how the controller based on the fuzzy model manages to achieve a better performance, due to the generalization properties of the fuzzy model.

An important point of this first set of experiments was to highlight the fact that the given initial data is insufficient to build a model of the plant required to perform a good forward simulation. Therefore, the second set of experiments focused on the evaluation of the performance of the controller in the same scenarios when adaptation capabilities were added to the model of the plant.

The graphs of Fig. 5 show the performance of the controller when used in connection with the adaptive versions of the fuzzy and the lazy model of the plant in the noise-free scenario.

In both cases it is possible to see an improvement of the performance of the system; however, it is clear from the image that the controller based on the adaptive lazy system outperforms the one based on the fuzzy model. In the case of the fuzzy model the recursive least mean square algorithm is used to on-line update the parameters of the consequences of the rules of the fuzzy model. In the case of the lazy model the adaptation procedure consists in adding a new input–output pair to the database of the examples at each time step. In the case of the controller based on the lazy system it is possible to see that the ringing signal which characterized the nonadaptive version of Fig. 3 disappeared. This is because as new points are added to the database of examples the lazy model can find neighbors that are closer to the query point and therefore more informative: this increases the stability and the accuracy of the derivatives.

The graphs of Fig. 6 show the performance of the controller based on the fuzzy and lazy models of the plant in the noisy scenario.

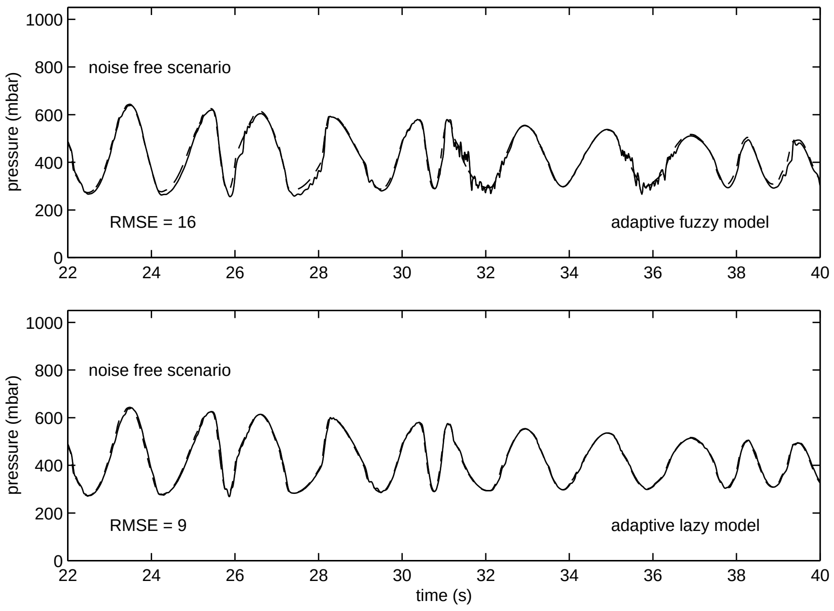

Fig. 5. Performance of the controller based on an adaptive fuzzy model, and an adaptive lazy model in a noise-free environment. The solid line is the actual output of the plant, while the reference is represented by the dashed line.

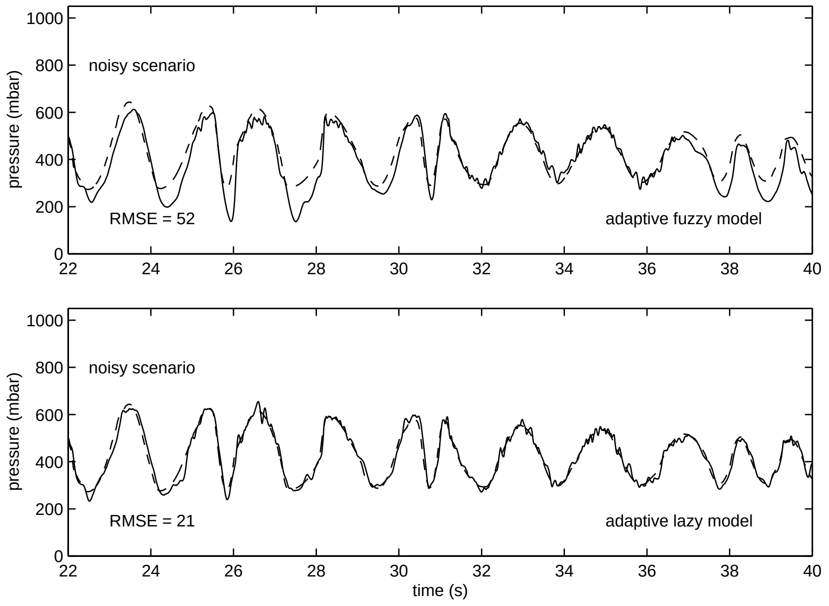

Fig. 6. Performance of the controller based on an adaptive fuzzy model, and an adaptive lazy model in a noisy environment. The solid line is the actual output of the plant, while the reference is represented by the dashed line. The output of the controller is measured before addition of the sensor noise, and gives the real value of the pressure at the manifold level.

It is possible to see that there is no improvement of the performance of the controller when the adaptive version of the fuzzy model of the plant was used instead of static version. However, the controller based on the lazy adaptive version of the model of the plant shows a great improvement in comparison with the one based on the static version, and it performs better than the one based on the fuzzy model. Therefore, even in the noisy case it is possible to appreciate the benefit of using a controller based on a lazy adaptive version of the model of the plant.

All the experiments proposed in this section have been performed with the NLMIMO Toolbox [5], and with the Lazy Learning Toolbox[2] [7].

5.1. *Some considerations on memory requirements for lazy learning*

One of the shortcomings of the adaptive lazy learning approach is that as time passes the database of examples grows in size. This obviously leads to an increase in memory requirements, and also to an increase in computational cost since more points need to be scanned in order to find the neighbors of each query. Forgetting strategies could be implemented, but particular care is necessary in this case since this approach could lead to the elimination of information about the dynamics of the systems in some areas of the state space.

A different approach consists in adding a new point to the database only when the prediction differs more than a predefined threshold from the real output of the plant. This approach is based on the principle that if the prediction is sufficiently accurate for a particular region, there is no need to enrich the database by adding new examples in that region. We propose a detailed analysis of the application of this principle to the case of a noise-free scenario. In this particular case, it is straightforward to evaluate the accuracy of a prediction: it is sufficient to compare the prediction with the reading of the output of the plant. In a more general noisy scenario, since the reading is affected by sensor noise, such a comparison is meaningless and must be replaced by some other criterion based for instance on a statistical test.

[2] `http://iridia.ulb.ac.be/ ∼lazy/`.

Table 1
Performance of the system in the noise-free scenario using different values of the adaptation threshold. Points are added to the database of the example used by the lazy system only if the difference between the predicted value and the measured value is greater than the threshold

Threshold (mbar)	Points added	RMSE
0	8000	9.43
0.01	6756	9.43
0.05	4735	9.16
0.1	3562	9.72
0.5	1153	9.87
1	548	10.10
5	41	16.81
10	28	27.98
50	6	74.35
100	4	62.72
200	1	54.07
∞	0	49.43

As far as the noise-free scenario is concerned, Table 1 reports the performance of the system using different values of the threshold. The second column reports the number of points added to the database during the experiment. The table shows, as a general trend, that the performance decreases as the value of the threshold is increased. In particular it can be noticed that the reduction, up to a certain level, of the number of points added to the database does not significantly alter the performance of the system. On the other hand, for larger values of the threshold, when fewer points are added to the database, the performance of the system decreases more dramatically.

6. Conclusions

Divide and conquer techniques are powerful techniques for learning from a limited amount of data. We illustrated and analyzed the performance of a controller used in conjunction with two different approaches for modeling the plant to be controlled, starting from a limited amount of input–output data. When the model of the plant does not take advantage of examples gathered on-line, it has been seen that the use of a Takagi–Sugeno model leads the system to a smoother response. However, when new examples are gathered on-line the performance of the system improves, and the controller based on the lazy system outperforms the one based on the fuzzy model.

Acknowledgements

This research was partially supported by the European FAMIMO project (LTR Project 21911 Reactive Scheme, Task 4.2). The work of Edy Bertolissi was supported by a Marie Curie Fellowship (CEC-TMR Contract No. ERBFM-BICT972495). The work of Mauro Birattari was supported by the FIRST program of the Région Wallonne, Belgium. The work of Gianluca Bontempi was supported by a Marie Curie Fellowship (CEC-TMR Contract No. FM-BICT960692).

References

[1] D.W. Aha, Editorial, Artificial Intelligence Rev. 11 (1–5) (1997) 1–6.

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

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

[4] H. Bersini, A. Duchateau, N. Bradshaw, Using incremental learning algorithms in the search for minimal and effective fuzzy models, in: Proc. FUZZ-IEEE'97, Barcelona, Spain, 1997, pp. 1417–1422.

[5] E. Bertolissi, A. Duchateau, H. Bersini, Nlmimo, non-linear multi-input multi-output toolbox, in: Proc. MATHMOD 2000 Conf., Vienna, Austria, 2–4 February 2000.

[6] E. Bertolissi, A. Duchateau, H. Bersini, F. Vanden Berghen, Direct fuzzy control for MIMO processes, in: Proc. FUZZ-IEEE 2000, San Antonio, TX, 7–10 May 2000.

[7] M. Birattari, G. Bontempi, The lazy learning toolbox, for use with matlab, Technical Report TR/IRIDIA/99-7, IRIDIA-ULB, Brussels, Belgium, 1999.

[8] 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.

[9] G. Bontempi, Local Learning Techniques for Modeling, Prediction and Control, Ph.D. Thesis, IRIDIA-Université Libre de Bruxelles, 1999.

[10] G. Bontempi, H. Bersini, M. Birattari, The local paradigm for modeling and control: From neuro-fuzzy to lazy learning, Fuzzy Sets and Systems 121 (1) (2001) 59–72.

[11] D.W. Clarke, C. Mohtadi, P.S. Tuffs, General predictive control — part 1. The basic algorithm, Automatica 23 (2) (1987) 137–148.

[12] W.S. Cleveland, C. Loader, Smoothing by local regression: principles and methods, Comput. Statist. 11 (1995).

[13] G. Cybenko, Just-in-time learning and estimation, in: S. Bittanti, G. Picci (Eds.), Identification, Adaptation, Learning. The Science of Learning Models from Data, NATO ASI Series, Springer, Berlin, 1996, pp. 423–434.

[14] N.R. Draper, H. Smith, Applied Regression Analysis, Wiley, New York, 1981.

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

[16] Y. Landau, Adaptive Control. The Model Reference Approach, Marcel Dekker, New York, 1979.

[17] J. Moody, C.J. Darken, Fast learning in networks of locally-tuned processing units, Neural Comput. 1 (2) (1989) 281–294.

[18] R. Murray-Smith, A local model network approach to nonlinear modelling, Ph.D. Thesis, Department of Computer Science, University of Strathclyde, Strathclyde, UK, 1994.

[19] R. Murray-Smith, T.A. Johansen (Eds.), Multiple Model Approaches to Modeling and Control, Taylor & Francis, London, 1997.

[20] R.H. Myers, Classical and Modern Regression with Applications, 2nd Edition, PWS-KENT Publishing Company, Boston, MA, 1994.

[21] M.J.D. Powell, Radial Basis Functions for multivariable interpolation: a review, Algorithms for Approximation, Clarendon Press, Oxford, 1987, pp. 143–167.

[22] J.-M. Renders, M. Saerens, H. Bersini, Fuzzy adaptive control of a certain class of siso discrete-time processes, Fuzzy Sets and Systems 85 (1) (1997) 50–61.

[23] D.E. Rumelhart, G.E. Hinton, R.K. Williams, Learning representations by backpropagating errors, Nature 323 (9) (1986) 533–536.

[24] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. Systems Man Cybernet. 15 (1) (1985) 116–132.