Lazy Learning vs. Fuzzy Systems: Two multiple-model approaches for adaptive control

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

IRIDIA, Université Libre de Bruxelles
50, av. Franklin Roosevelt
1050 Brussels, Belgium
Tel: +32 2 650 27 29, Fax: 32 2 650 27 15
eberto@iridia.ulb.ac.be, {mbiro,gbonte,aduchate,bersini}@ulb.ac.be

Abstract

This paper presents an approach for modeling and controlling discrete-time non-linear dynamical system. Our approach is well suited in the cases when a model of the system is not given a priori *and has to be learned from a limited amount of data, or when the system possesses a time varying dynamics. The controller consists of a multiple step ahead direct adaptive controller. At each time step a forward simulation of the system composed by the controller and the plant model is performed. This dynamic information is then used to adapt the parameters of the controller. In order to obtain good results it is necessary to have a good model of the process to control. Takagi-Sugeno fuzzy systems and* lazy learning, *are two approaches which can be successfully used to simulate the process to control. Simulation examples of the control of the manifold pressure of a car engine are given.*

1. Introduction

The idea of employing linear techniques in a nonlinear setting is not new in control literature but had recently gained popularity thanks to methods for combining multiple estimators and controllers in different operating regimes of the system [12, 8].

The goal of this paper is to compare a Takagi-Sugeno [14] fuzzy systems with *lazy learning* [1] on the task of modeling an unknown dynamic system in the framework of the implementation of a direct adaptive controller (DAC). In this type of controllers, at each time step a forward simulation of the system composed by the controller plus the plant is performed. The results of this computation are then used to tune the controller by adapting its parameters. In order to perform the forward simulation a model of the plant is required. Its quality is directly related with the performance of the controller, since errors in the model could lead to incorrect updates of its parameters.

The problem of modeling a process from a limited amount of 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 paradigms have emerged: global versus *divide and conquer*.

Global modeling builds a single functional model on the basis of the dataset. This has traditionally been the approach taken in neural network modeling and other form of nonlinear statistical regression. The available dataset is used by a learning algorithm to produce a model of the mapping and then the dataset is discarded and only the model is kept.

Multiple model approaches [12], also known as *divide and conquer* techniques, partition a complex problem into simpler ones whose solutions can be combined to provide a solution of the original problem. An example of these approaches are the *modular* architectures where different modules are composed in order to cover the input space. Even if these approaches use the combination of local models, the learning procedure remains a functional estimation problem, and requires the same procedures used for generic global models. Takagi-Sugeno fuzzy systems represent a well known example of this approach.

Another example of multiple model methods are the so called *local modeling* techniques, 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 to 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 not calculated, it is necessary to keep into 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* is one of these local modeling techniques.

Takagi-Sugeno fuzzy systems and *lazy learning* can be successfully used within DAC for the modeling of the controlled plant. In particular the advantages of the latter become evident if it is necessary to perform a prediction when an initially limited amount of process data is available, but further examples can be collected on line.

Another comparison between a fuzzy system and lazy learning as identification modules in a different nonlinear control framework has been proposed in Bontempi *et al.* [7].

2. Local modeling as an optimization problem

In Takagi-Sugeno fuzzy systems the computation of the value of the unknown function is performed by the fuzzy interpolation of a set of linear systems which locally approximate the desired function. These systems define a set of fuzzy rules which partition the input space:

$$\mathcal{R}^{(l)}: \text{IF } \varphi_1 \text{ IS } A_1^{(l)} \text{ AND } \ldots \text{ AND } \varphi_n \text{ IS } A_n^{(l)} \text{ THEN } y = a_0^{(l)} + a_1^{(l)}\varphi_1 + \cdots + a_n^{(l)}\varphi_n$$

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

$$y = \sum_{l=1}^{M} \frac{\mu_{A^{(l)}}(\varphi)\left(a_0^{(l)} + a_1^{(l)}\varphi_1 + \cdots + a_n^{(l)}\varphi_n\right)}{\sum_{k=1}^{M}\mu_{A^{(k)}}(\varphi)}$$

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

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

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

In the *lazy learning* 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.

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

Given a specific query point φ_q, the prediction of the value $y_q = f(\varphi_q)$ is computed as follows. First, for each sample $(\varphi_{[i]}, y_{[i]})$ a weight w_i is computed as a function of the distance $d(\varphi_{[i]}, \varphi_q)$ from the query point φ_q to the point $\varphi_{[i]}$. Each row of $\mathbf{\Phi}$ and $\mathbf{y}$ is then multiplied by the corresponding weight creating the variables $\mathbf{Z} = \mathbf{W}\mathbf{\Phi}$ 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 $(\mathbf{Z}^T\mathbf{Z})\beta = \mathbf{Z}^T\mathbf{v}$ and the prediction of the value $f(\varphi_q)$ is obtained evaluating such a model in the query point:

$$\hat{y}_q = \varphi_q^T(\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{v}. \quad (1)$$

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.* [6], 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 [13], which is a simple, well-founded and economical way to perform *leave-one-out* cross validation [10] 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 [6]. One of the most important parameters to be tuned in a local model configuration is the size of the region surrounding φ_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 φ_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

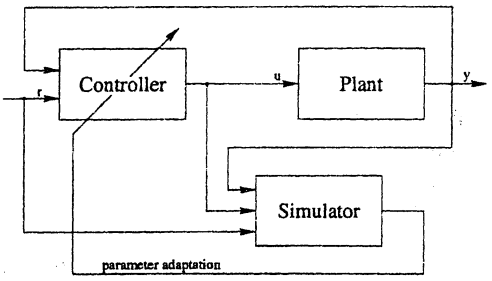

Figure 1. Schematic representation of the system

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 [6].

3. System description

Figure 1 shows the structure of a closed loop control system which is composed by: 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:

$$y_i(k+1) = F_i\big(\mathbf{y}(k),\ldots,\mathbf{y}(k-p_y+1), \mathbf{u}(k),\ldots,\mathbf{u}(k-p_u+1)\big), \quad i=1,\ldots,m \quad (2)$$

where the scalar $y_i(k)$ is the i^{th} output of the plant at time k, $\mathbf{u}(k)$ is the input vector $[u_1(k),u_2(k),\ldots,u_n(k)]^T$ and $\mathbf{y}(k)$ is the output vector $[y_1(k),y_2(k),\ldots,y_m(k)]^T$, 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:

$$\phi(k) = [\mathbf{y}(k),\ldots,\mathbf{y}(k-p_y+1), \mathbf{u}(k-1),\ldots,\mathbf{u}(k-p_u+1)] \quad (3)$$

equation (2) can be rewritten as:

$$y_i(k+1) = F_i\big(\phi(k),\mathbf{u}(k)\big), \qquad i=1,\ldots,m \quad (4)$$

which in vector notation becomes:

$$\mathbf{y}(k) = \mathbf{F}\big(\phi(k),\mathbf{u}(k)\big) \quad (5)$$

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

$$\mathbf{u}(k) = \mathcal{F}\big(\mathbf{r}(k),\psi(k);\mathbf{w}\big) \quad (6)$$

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 $[r_1(k),r_2(k),\ldots,r_m(k)]^T$. 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 is equal to:

$$\hat{y}_i(k+1) = \hat{F}_i\big(\phi(k),\mathbf{u}(k)\big), \qquad i=1,\ldots,m \quad (7)$$

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 figure 1 it is possible to design a learning algorithm, based on the principles defined in generalized predictive control theory [9], which performs an adaptation of the weights of the controller using information about the future behaviour of the system. The predictor will provide the controller with information regarding the futures values of the $\hat{\mathbf{y}}$ 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. It exists a unique series of inputs $[\mathbf{u}(k),\mathbf{u}(k+1),\ldots,\mathbf{u}(k+H_c)]$ which leads the output signals $\mathbf{y}$ towards $\mathbf{r}$ at time $k+H_p$. 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 ($H_p \geq H_c$). Control actions are considered constant once the control horizon is reached: $\mathbf{u}(t) = \mathbf{u}(k+H_c)$ for $t > k+H_c$;

3. Every $\frac{\partial F(\cdots)}{\partial \mathbf{u}(\mathbf{t})}$, for $t < k$ is equal to 0 since past actions are considered constant;

4. The controller $\mathcal{F}\big(\mathbf{r}(k),\psi(k);\mathbf{w}\big)$ can approximate the series of perfect control actions $[\mathbf{u}(k),\mathbf{u}(k+1),\ldots,\mathbf{u}(k+H_c)]$ 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 [11].

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

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \eta \frac{\partial J}{\partial \mathbf{w}} \tag{8}$$

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

$$\begin{aligned} J = &\frac{1}{2} \sum_{t=k}^{k+H_p} \big(\mathbf{r}(t) - \hat{\mathbf{y}}(t)\big)^T Q \big(\mathbf{r}(t) - \hat{\mathbf{y}}(t)\big) \\ &+ \frac{1}{2} \sum_{t=k-1}^{k+H_c} \big(\Delta \mathbf{u}(t)\big)^T R \big(\Delta \mathbf{u}(t)\big) \end{aligned} \tag{9}$$

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 $\big(\mathbf{r}(t) - \hat{\mathbf{y}}(t)\big)$ while the matrix $R \in \Re^{m \times m}$ has the effect to penalize the large variations of $\Delta \mathbf{u}(t)$ which could destabilize the system. Substituting this expression in equation 8 and recalling equation 5 it is possible to obtain:

$$\begin{aligned} \mathbf{w}(k+1) = \mathbf{w}(k) - \eta \Bigg(&\sum_{t=k}^{k+H_p} \big(\mathbf{r}(t) - \hat{\mathbf{y}}(t)\big)^T Q \frac{\partial \hat{\mathbf{y}}(t)}{\partial \mathbf{w}} \\ &+ \sum_{t=k-1}^{k+H_c} \big(\Delta \mathbf{u}(t)\big)^T R \frac{\partial \Delta \mathbf{u}(t)}{\partial \mathbf{w}} \Bigg) \end{aligned} \tag{10}$$

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

$$\begin{aligned} \frac{\partial \mathbf{u}(t-1)}{\partial \mathbf{w}} &= \frac{\partial \mathcal{F}\big(\mathbf{r}(t-1), \psi(t-1); \mathbf{w}\big)}{\partial \mathbf{w}} \\ &= \frac{\partial \mathcal{F}(\cdots)}{\partial \mathbf{w}} + \frac{\partial \mathcal{F}(\cdots)}{\partial \psi(t)} \frac{\partial \psi(t)}{\partial \mathbf{w}} \end{aligned} \tag{11}$$

and

$$\begin{aligned} \frac{\partial \hat{\mathbf{y}}(t)}{\partial \mathbf{w}} &= \frac{\partial \hat{\mathbf{F}}\big(\mathbf{u}(t-1), \phi(t-1)\big)}{\partial \mathbf{w}} \\ &= \frac{\partial \hat{\mathbf{F}}(\cdots)}{\partial \phi(t-1)} \frac{\partial \phi(t-1)}{\partial \mathbf{w}} \\ &+ \frac{\partial \hat{\mathbf{F}}(\cdots)}{\partial \mathbf{u}(t-1)} \left(\frac{\partial \mathcal{F}(\cdots)}{\partial \mathbf{w}} + \frac{\partial \mathcal{F}(\cdots)}{\partial \psi(t-1)} \frac{\partial \psi(t-1)}{\partial \mathbf{w}} \right) \end{aligned} \tag{12}$$

The full description of this algorithm can be found in Bertolissi *et al.* [4].

The multiple step algorithm is a quite complex algorithm. It allows to control system 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. 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 assumption 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 behaviour 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 errors accumulation.

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 *lazy learning* represents an ideal solution in all the cases when the dynamics of the system cannot be completely defined from the available training data, or the dynamics of the system is time variant. In the case of neural networks, or fuzzy systems, when new data becomes available a new identification procedure is usually necessary to incorporate this new knowledge in the model. In the case of *lazy learning* it is just necessary to add the new data to the database which will be used to calculate the subsequent predictions. This means that the lazy learning simulator can be updated on line, while the controller is running. The implementation of forgetting strategies is also straightforward since it is only necessary to delete the old points from the database.

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 this is the fresh air quantity introduced into the cylinders through an electrical throttle which control the manifold pressure.

In this paper we propose, as an example, a subsection of

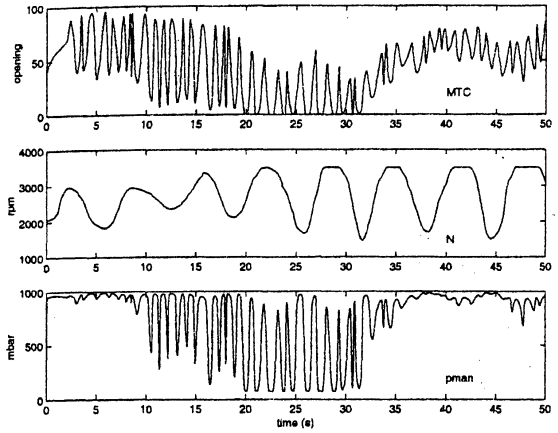

Figure 2. Training sequence used to identify the model of the manifold pressure

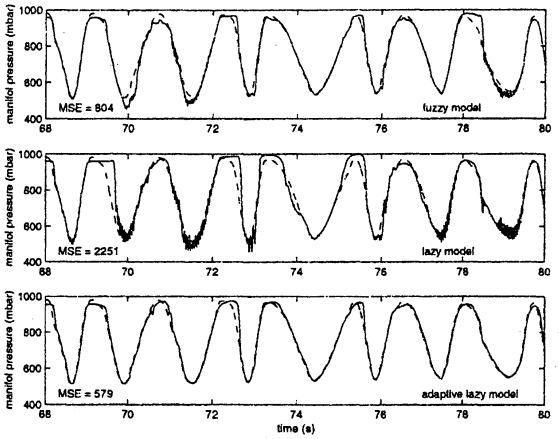

Figure 3. Performance of the adaptive controller based on a fuzzy model (top), a lazy model (center) and a lazy adaptive model (bottom). The solid line is the actual output of the plant, while the reference is represented by the dashed line.

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 which determines the the manifold pressure $pman \in [200\ 1024]$. The process in a second order highly nonlinear system. The control action is evaluated on the basis of the fresh air throttle control $MTC \in [0\ 100]$, which indicates the position of the throttle, and the engine speed $N \in [800\ 3400]$, which indicates the number of revolutions per minute (rpm) of the engine.

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 is available for the modeling of the process, the plant has been identified using only the first 10000 points of a 50000 points sequence collected by exciting the system with a pseudo random input which had been successfully used to identify the system. In this way the data provides information only on part of the system dynamics. The training data used for the identification procedures is displayed in figure 2. Two different simulators have been implemented, one based on a lazy learning approximator, and one based on a Takagi-Sugeno fuzzy system. As far as the lazy learning simulator is concerned, linear local models had been considered, and the number of neighbors varied in the range between 30 and 60. In addition the prediction for each query had been obtained as a combination of the 4 best local models, according to the PRESS estimate, out of those considered in the above mentioned range [6]. As far as the Takagi-Sugeno fuzzy system is concerned, an incremental identification procedure had been used [2]. This procedure automatically generates a fuzzy partition of the process input space by letting grow the number of fuzzy rules until an optimum is achieved with respect to the given performance criteria. In this particular case 13 rules were defined for describing the dynamical system.

In this particular experience the controller itself had 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 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 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 mhillisecconds. The controller is described by 108 Takagi-Sugeno fuzzy rules which cover the input space. At 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 $\eta = 10^{-8}$.

Figure 3 shows the performance of the adaptive controllers (solid line) after 68 seconds of training. The top part of the figure displays the results obtained when a fuzzy model was used for modeling the controlled plant. It is possible to see that performance is sometime quite poor and the controller is not able to drive the output of the plant to the desired values. The graph in the middle shows the performance of the adaptive controller using the lazy learning approximator. It is apparent that the performance is quite poor. The problems arise in the same places where the fuzzy controller displays deficiencies, but the resulting output is less smooth. This means that the prediction and the values of the derivatives are less reliable, leading the system to exhibit high frequency components. It is possible

to conclude that the fuzzy system in this case offered better generalization capabilities in comparison to the *lazy learning* approach. This 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. It is possible to see that this type of problems can be overcome by means of an adaptive lazy learning approximator whose performance is reported in the graph at the bottom of figure 3. The adaptation procedure is performed every time the prediction differ more than 0.1% from the output of the system. In this case the input-output regressor is added to the database of the examples, thus adding information in areas of the space which do not host enough points. In his way a larger number of points is used for the computations of the queries which increases the stability of the derivatives. At the beginning of the displayed sequence the dimensions of the database of the adaptive lazy learning approximator were increased of about 8%.

All the experiments have been performed taking advantage of the NLMIMO Toolbox [3], which integrates the Lazy Learning Toolbox[1] [5] as one of its components.

6. Conclusions and future developments

Divide and conquer techniques are powerful techniques for learning from a limited amount of data. We illustrated and analyzed the performance of an adaptive controller used in conjunction with three different approaches for simulating the plant to be controlled. When the model of the plant is not adaptive it has been seen that the use of a Takagi-Sugeno model leads the controller to better results. However when adaptation capabilities are added to the *lazy learning* simulator, the performance of the controller improves. This is due to that fact that the adaptation overcomes the problems related with the unmodeled dynamics of the system. The main advantage of this adaptation process consists in the fact that it is easy to implement and does not require further computations.

Future developments will mainly focus on the implementation of more sophisticated forward simulators, and on the introduction of more refined adaptation techniques for what concerns the lazy learning simulator.

References

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

[2] H. Bersini, A. Duchateau, and N. Bradshaw. Using incremenmtal learning algorithms in the search for minimal and effective fuzzy models. In *Proceedings of the FUZZ-IEEE '97*, pages 1417–1422, Barcelona, Spain, 1997.

[3] E. Bertolissi, A. Duchateau, and H. Bersini. Nlmimo, nonlinear multi-input multi-output toolbox. In *Proceedings of the MATHMOD 2000 Conference*, Vienna, Austria, 2-4 February 2000.

[4] E. Bertolissi, A. Duchateau, H. Bersini, and F. V. Berghen. Direct fuzzy control for MIMO processes. In *Proceedings FUZZ-IEEE 2000*, San Antonio, Texas, 7-10 May 2000.

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

[6] M. Birattari, G. Bontempi, and H. Bersini. Lazy learning meets the recursive least-squares algorithm. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 375–381, Cambridge, 1999. MIT Press.

[7] G. Bontempi, H. Bersini, and M. Birattari. The local paradigm for modeling and control: From neuro-fuzzy to lazy learning. *Fuzzy Sets and Systems*, 1999. in press.

[8] G. Bontempi, M. Birattari, and H. Bersini. Lazy learning for modeling and control design. *International Journal of Control*, 72(7/8):643–658, 1999.

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

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

[11] Y. Landau. *Adaptive Control. The Model Reference Approach*. Marcel Dekker, 1979.

[12] R. Murray-Smith and T. A. Johansen, editors. *Multiple Model Approaches to Modeling and Control*. Taylor and Francis, 1997.

[13] R. H. Myers. *Classical and Modern Regression with Applications*. PWS-KENT Publishing Company, Boston, MA, second edition edition, 1994.

[14] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15(1):116–132, 1985.

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