Amauri Holanda de Souza Júnior

Regional Models and Minimal Learning Machines for Nonlinear Dynamic System Identification

Fortaleza

2014

Amauri Holanda de Souza Júnior

Regional Models and Minimal Learning Machines for Nonlinear Dynamic System Identification

A thesis presented to the Graduate Program of Teleinformatics Engineering at Federal University of Ceará in fulfillment of the thesis requirement for the degree of Doctor of Philosophy in Teleinformatics Engineering.

Federal University of Ceará

Center of Technology

Graduate Program of Teleinformatics Engineering

Supervisor: Guilherme Alencar Barreto

Fortaleza

2014

Amauri Holanda de Souza Júnior
Regional Models and Minimal Learning Machines for Nonlinear Dynamic System Identification/ Amauri Holanda de Souza Júnior. – Fortaleza, 2014-
114 p. : il. (algumas color.) ; 30 cm.

Supervisor: Guilherme Alencar Barreto

Thesis (Doctorate) – Federal University of Ceará
Center of Technology
Graduate Program of Teleinformatics Engineering, 2014.

1. Palavra-chave1. 2. Palavra-chave2. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

CDU 02:141:005.7

Amauri Holanda de Souza Júnior

Regional Models and Minimal Learning Machines for Nonlinear Dynamic System Identification

A thesis presented to the Graduate Program of Teleinformatics Engineering at Federal University of Ceará in fulfillment of the thesis requirement for the degree of Doctor of Philosophy in Teleinformatics Engineering.

Approved. 31st of October, 2014:

Guilherme Alencar Barreto
Federal University of Ceará (UFC)

Fernando José Von Zuben
State University of Campinas (UNICAMP)

Roberto Kawakami Harrop Galvão
Technological Institute of Aeronautics (ITA)

Francesco Corona
Federal University of Ceará (UFC) / Aalto University

João Paulo Pordeus Gomes
Federal University of Ceará (UFC)

Fortaleza
2014

For my grandmother Raimunda Oliveira, in memoriam.

Abstract

This thesis addresses the problem of identifying nonlinear dynamic systems from a *machine learning* perspective. In this context, very little is assumed to be known about the system under investigation, and the only source of information comes from input/output measurements on the system. It corresponds to the *black-box modeling* approach. Numerous strategies and models have been proposed over the last decades in the machine learning field and applied to modeling tasks in a straightforward way. Despite of this variety, the methods can be roughly categorized into global and local modeling approaches. Global modeling consists in fitting a single regression model to the available data, using the whole set of input and output observations. On the other side of the spectrum stands the local modeling approach, in which the input space is segmented into several small partitions and a specialized regression model is fit to each partition.

The first contribution of the thesis is a novel supervised global learning model, the *Minimal Learning Machine* (MLM). Learning in MLM consists in building a linear mapping between input and output distance matrices and then estimating the nonlinear response from the geometrical configuration of the output points. Given its general formulation, the Minimal Learning Machine is inherently capable to operate on nonlinear regression problems as well as on multidimensional response spaces. Naturally, its characteristics make the MLM able to tackle the system modeling problem.

The second significant contribution of the thesis represents a different modeling paradigm, called *Regional Modeling* (RM), and it is motivated by the *parsimonious principle*. Regional models stand between the global and local modeling approaches. The proposal consists of a two-level clustering approach in which we first partition the input space using the *Self-Organizing Map* (SOM), and then perform clustering over the prototypes of the trained SOM. After that, regression models are built over the clusters of SOM prototypes, or regions in the input space.

Even though the proposals of the thesis can be thought as quite general regression or supervised learning models, the performance assessment is carried out in the context of system identification. Comprehensive performance evaluation of the proposed models on synthetic and real-world datasets is carried out and the results compared to those achieved by standard global and local models. The experiments illustrate that the proposed methods achieve accuracies that are comparable to, and even better than, more traditional machine learning methods thus offering a valid alternative to such approaches.

Key-words: Nonlinear System Identification, Learning Machines, Clustering, Supervised Learning.

List of Figures

List of Tables

List of abbreviations and acronyms

AMSE	Averages on Mean Squared Error
ANFIS	Artificial Neuro-Fuzzy Inference System
APRBS	Amplitude Pseudo Random Binary Signal
ARX	AutoRegressive with eXogenous input
DB	Davies-Bouldin
ekNN	Enclosing k Nearest Neighbors
ELM	Extreme Learning Machine
GP	Gaussian Processes
ITL	Information Theoretic Learning
KDE	Kernel Dependency Estimation
kNN	k Nearest Neighbors
KS	Kolmogorov-Smirnov test
LL	Lazy Learning
LLM	Local Linear Map
LLR	Local Linear Regression
LMS	Least Mean Squares
LOO	Leave-One-Out
LS	Ordinary Least Squares
LWL	Locally Weighted Learning

MIMO	Multiple-Input Multiple-Output
MLM	Minimal Learning Machine
MLP	Multi-layer Perceptrons
MSE	Mean Squared Error
NARX	Nonlinear AutoRegressive with eXogenous input
NN	Natural Neighbors
NNi	Natural Neighbors Inclusive
OPELM	Optimally Pruned Extreme Learning Machine
RBF	Radial Basis Functions
RLM	Regional Linear Models
RLS	Recursive Least Squares
RM	Regional Modeling
RMSE	Root Mean Squared Error
SLFN	Single-hidden Layer Feedforward network
SVM	Support Vector Machines
SVR	Support Vector Regression
SISO	Single-Input Single-Output
SOM	Self-Organizing Map
TS	Takagi-Sugeno
VQ	Vector Quantization
VQTAM	Vector-Quantized Temporal Associative Memory

List of symbols

$\mathcal{X}$	Set of input data points used for estimation.
$\mathcal{Y}$	Set of output data points used for estimation.
N	Number of data points used for estimation.
$\mathbf{x}(n)$	Input vector at sampling time n.
$y(n)$	Output observation at sampling time n.
$\mathbb{X}$	Input space.
$\mathbb{Y}$	Output space.
$\boldsymbol{\beta}$	Parameters of linear models.
$\mathbf{w}_i$	i-th weight vector in Self-Organizing Maps.
$\mathbf{r}_m$	m-th reference input point (MLM).
$\mathbf{t}_m$	m-th reference output point (MLM).
$\mathbf{m}_l$	l-th hidden unit (SLFN).
$\mathbf{p}_i$	i-th k-means prototype
M	Number of reference points (MLM).
L	Number of hidden units (SLFN).
Q	Number of SOM prototypes.
K	Used for k means, kSOM, kNN.

Contents

Chapter 1

Introduction

Modeling takes a central role in engineering and science in general. Almost all systems that someone can think of or imagine can be described by a mathematical model (BILLINGS, 2013). Knowing a model that describes the diversity of behaviors that a dynamic system can reveal — particularly the nonlinear ones — is essential, not only for theoretic or applied research fields, but also for the process or control engineer who is interested in understanding better the dynamics of the system under investigation. The resulting model must approximate the actual system as faithfully as possible so that it can be used for several purposes, such as simulation, predictive control or fault detection.

This is an introductory chapter where we state the system identification problem to be addressed in this thesis in Section 1.1. In Section 1.2, we describe the objectives and motivations of the thesis. The structure of the thesis and contributions are reported in Section 1.3. Finally, in Section 1.4, the main scientific production during the Ph.D. course is presented.

1.1 The system identification problem

Despite the variety of definitions for the word "system", in this work a system is considered to be an object in which different variables interact at various time and space scales and that produces observable signals (KEESMAN, 2011). Many engineering applications need a compact and accurate description of the dynamic of such systems. We can accomplish that using the first principles of physics, mathematics, chemistry, biology, etc. However, deriving those models is complex because highly specialized knowledge about the system is needed, which may be lacking. The modeling approach based on pure physical aspects of a system is called white-box modeling (SJÖBERG et al., 1995).

An alternative way of describing dynamic systems is by finding a mathematical

description of a dynamic system from empirical data or measurements in the system. This is usually referred to as the *black-box modeling* approach (SJÖBERG et al., 1995), and it is the topic of this thesis. In cases where some partial a prior knowledge about the system is available while modeling it, such an approach stands between white-box and black-box modeling and it is said to be gray-box modeling (SOHLBERG, 2003).

A system is dynamic when its output at any time depends on its history, and not just on the current inputs. Thus, a dynamic system has memory and is usually described in terms of difference or differential equations. A single-input single-output dynamic system can be denoted by a set $\mathcal{O} = \{u(n), y(n)\}_{n=1}^{N}$ of inputs $u(n) \in \mathbb{R}$ and outputs $y(n) \in \mathbb{R}$ collected at sampling time n, and its input-output discrete time representation[1] is given by

$$y(n) = f(\mathbf{x}(n)) + \epsilon(n), \tag{1.1}$$

where $\mathbf{x}(n)$ represents the *regression vector* at sampling time n and its components are given as a function of the relevant variables of the system at previous time. Usually, $\mathbf{x}(n)$ encompasses past inputs and/or outputs and/or measurable disturbance observations. The additive term $\epsilon(n)$ accounts for the fact that the output observations $y(n)$ may not be an exact function of past data.

The function $f : \mathbb{X} \to \mathbb{Y}$ is the unknown, usually nonlinear, target function, where $\mathbb{X}$ denotes the input space, and $\mathbb{Y}$ is the output space. The ultimate goal in black-box system identification consists in finding a regression model $h \in \mathcal{H} : \mathbb{X} \to \mathbb{Y}$ that approximates f in its entire domain, based on a finite set of measurements in the system $\mathcal{O}$. Set $\mathcal{H}$ denotes the candidate models or hypotheses. Over the last decades, a number of learning models have been proposed to approximate or learn f, such as piecewise affine models (PAPADAKIS; KABURLASOS, 2010), neural networks (NARENDRA; PARTHASARATHY, 1990), fuzzy local models (TAKAGI; SUGENO, 1985; REZAEE; FAZEL ZARANDI, 2010), kernel-based methods (GREGORCIC; LIGHTBODY, 2008; ROJO-ALVAREZ et al., 2004), hybrid models (HABER; UNBEHAUEN, 1990; LIMA; COELHO; VON ZUBEN, 2007) and polynomial models (BILLINGS; CHEN; KORENBERG, 1989). Some of them will be discussed later in this thesis.

A common pipeline for identifying a dynamic system is illustrated in Figure 1. Basically, it includes: *i*) data acquisition; *ii*) model structure selection; *iii*) parameter estimation, and *iv*) model validation. The going backwards in Figure 1 occurs when the model is not appropriate, and then re-execution of previous stages is sometimes necessary to make a model adequate. In what follows, a brief description of each task within the system identification procedure is reported.

The algorithms used for black-box modeling of dynamic systems have mainly evolved in both automation/control and machine learning areas. Due to that, different

[1] A more general formulation consists of the space-state representation.

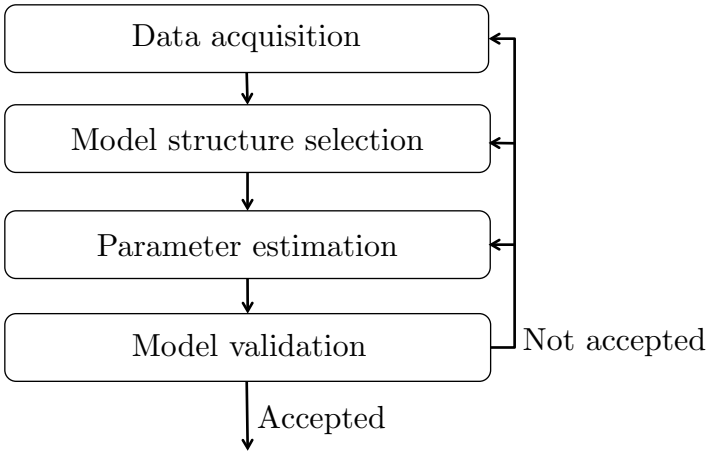


Figure 1: The system identification pipeline.

nomenclatures are used as synonyms in many cases. Therefore, in order to avoid confusion with the nomenclature used in this thesis, we provide a glossary of equivalence between terms:

- estimate = train, learn;
- validate = test, generalize;
- estimation data = training data/set;
- validation data = generalization set, test set;
- in-sample error = training error;
- out-of-sample error = test error, generalization error.

1.1.1 Data acquisition

The first step consists of collecting data from the system with the objective of describing how the system behaves over its entire range of operation. It includes decisions on the sampling time and the input signal. The data acquisition procedure is usually done by applying an input control signal, $u(n) \in \mathbb{R}$, to the system, and then observing the corresponding generated output, $y(n) \in \mathbb{R}$. As a result of this process, we have a set of corresponding input and output observations $\mathcal{O} = \{u(n), y(n)\}_{n=1}^{N}$, where n denotes a discrete time index. The signal $u(n)$ is also called *exogenous input* and $y(n)$ is the response.

The choice of the sampling time is crucial. A slow sampling rate may lead to an uncorrelated sequences, and then create difficulties in the identification procedure. A fast sampling rate may lead to extremely correlated sequences and then affect the parameter estimation step, since ill-conditioning regression matrices may be involved. Guidelines about the choice of the sampling rate of nonlinear systems are given by Aguirre (2007).

The quality of the model obtained is intimately related to the choice of the input signal used to excite the system. Ideally, the input-output time series should contain as much information as possible on the dynamics of the system. If nonlinearities are not present in the collected samples, it is impossible to learn the dynamics without any prior information about the system. The more complete and diverse the data is, the more feasible, from a learning point of view, the identification of the underlying system becomes. With respect to that, not only the frequency bands are important, but also amplitude information. The identification experiment which yields the data for the training set must be designed so that the input sequence excites the process dynamics as completely and uniformly as possible (BITTANTI; PIRODDI, 1996). To achieve this, white noise and *amplitude pseudo random binary signal* (APRBS) are commonly used.

There are many issues related to preliminary identification steps, such as pre-processing, normalization, nonlinearity detection, the choice of relevant variables in the system, etc. For a comprehensive discussion about such issues, we recommend the following texts: Nelles (2001) and Aguirre (2007).

1.1.2 Model structure selection

Selecting the model structure is concerned with defining a class of models that may be considered appropriate for describing the system. For this task, two steps are necessary:

- Select the dynamic structure of the model, which includes the definition of the regressors $\mathbf{x}(n)$.
- Define a model class $\mathcal{H}$ such that $h \in \mathcal{H} : \mathbb{X} \to \mathbb{Y}$ approximates the unknown static nonlinear mapping f.

Regarding the selection of the dynamic structure, a number of different models have been proposed, such as ARX (Auto-Regressive with eXogenous inputs), ARMAX (Auto-Regressive Moving Average with eXogenous inputs), NARX (Nonlinear ARX), NARMAX (Nonlinear ARMAX), NOE (Nonlinear Output Error), and NFIR (Nonlinear Finite Impulse Response). For a review about linear and nonlinear models for dynamic systems, we also recommend the reference text-books by Nelles (2001), Norgaard et al. (2000), and Aguirre (2007).

A wide class of nonlinear dynamic systems can be described in discrete time by the NARX model (NORGAARD et al., 2000), that is

$$\begin{aligned} y(n) &= f(y(n-1), \ldots, y(n-n_y), u(n-1), u(n-2), \ldots, u(n-n_u)) + \epsilon(n), \quad (1.2) \\ &= f(\mathbf{x}(n)) + \epsilon(n), \quad (1.3) \end{aligned}$$

where $\mathbf{x}(n) = [y(n-1), \ldots, y(n-n_y); u(n-1), u(n-2), \ldots, u(n-n_u)]^T$, $n_u \geq 1$ and $n_y \geq 1$ are the input-memory and output-memory orders, respectively. The functional form f denotes the unknown target function. The term $\epsilon(n)$ is often assumed to be white noise. The predictor associated with the NARX model, which we are interested in, is given by

$$\hat{y}(n) = h(y(n-1), \ldots, y(n-\hat{n}_y), u(n-1), u(n-2), \ldots, u(n-\hat{n}_u)), \quad (1.4)$$

$$= h(\mathbf{x}(n)), \quad (1.5)$$

where $h \in \mathcal{H}$ is a static nonlinear mapping that aims at approximating the "true" mapping f that generated the observations $u(n)$ and $y(n)$. The terms $\hat{n}_y$ and $\hat{n}_y$ are estimates of the system memory. Clearly, the regressors $\mathbf{x}(n)$, and consequently the model $h(\cdot)$, depend on $\hat{n}_y$ and $\hat{n}_u$, but we omit these dependencies in order to simplify the notation. As seen in Eq. (1.2), the choice of the NARX model only defines the relationship between variables in the dynamic system. In that sense, we need to refine our model class $\mathcal{H}$ by specifying the structure of the mapping $h(\cdot)$, which is the main topic of this thesis.

Model selection consists of picking a final hypothesis $g(\cdot)$ from the set of candidate models $\mathcal{H}$, using a finite set of observations $\mathcal{O}$. We are interested in models which produce good performance on data that have not been used for fitting/training the models, i.e., we are interested in the *generalization* or *out-of-sample* performance of a model. In general, we can assess generalization performance using only the available training examples. We can do this either analytically by means of optimization criteria — which combine model complexity and performance on estimation/training data, like the *Akaike information criterion* (AIC) (AKAIKE, 1974) and the *Bayesian information criterion* (BIC) (KASHYAP, 1977) — or by re-sampling methods, such as *cross-validation* and *bootstrap* (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

Re-sampling methods, such as *leave-one-out* (LOO) cross-validation, are of great importance for model selection. The idea of these approaches consists in using a separated part of the available samples in order to estimate the expected *out-of-sample* performance directly. The one which achieves the best performance on the fresh out-of-sample points is selected as the final model. Additionally, criteria like AIC and BIC can be used in combination with re-sampling methods (HONG et al., 2008).

1.1.3 Parameter estimation

The parameter estimation step provides the choice for the parameters in the model structure. The method for parameter estimation strongly depends on the model used for representing the dynamic system. Despite the large number of different models, in order to provide an estimate $\hat{\boldsymbol{\theta}}$ of the parameters $\boldsymbol{\theta}$ of the model $h(\mathbf{x}, \boldsymbol{\theta})$, we may consider a criterion such that the optimal estimate arises from an optimization procedure by minimizing or maximizing the chosen criterion. Many of the models approached in this thesis use the

residual sum of squares (RSS) between the observations $y(n)$ and $h(\mathbf{x}(n), \boldsymbol{\theta})$ as a criterion, that is

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{n=1}^{N} \Big(y(n) - h(\mathbf{x}(n), \boldsymbol{\theta}) \Big)^2. \tag{1.6}$$

If $\mathcal{H}$ is the class of linear mappings[2], then Eq. (1.6) leads to the well-known *ordinary least square* (LS) estimator (RAO; TOUTENBURG, 1999). We denote the final model choice by the mapping $g(.)$. The final model is the one whose structure is defined and its parameters estimated. In general, $g(\mathbf{x}(n)) = h(\mathbf{x}(n), \hat{\boldsymbol{\theta}})$, or

$$\begin{aligned} \hat{y}(n) &= h(\mathbf{x}(n), \hat{\boldsymbol{\theta}}) \\ &= g(\mathbf{x}(n)). \end{aligned} \tag{1.7}$$

Commonly, the variable $\hat{y}(n)$ represents the one-step-ahead predictions (or simply predictions) of the final model.

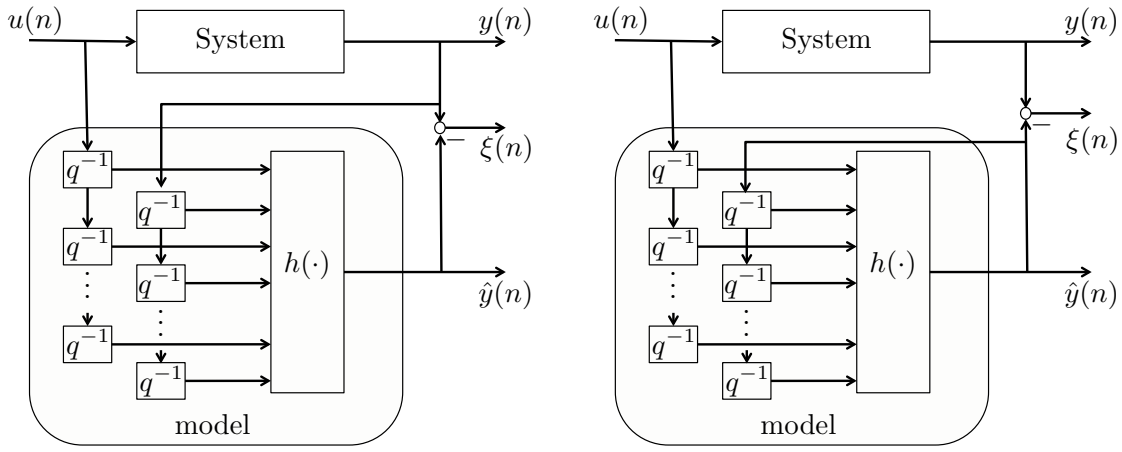
1.1.4 Model validation

There are two main approaches to validating the model $g(\cdot)$ in terms of its prediction capability: *i*) one-step-ahead predictions and *ii*) free-run simulation. On the one hand, one-step-ahead predictions use input and output observations at previous time as regressors in order to predict the current output. On the other hand, free-run simulation uses previous predictions rather than observations as regressors to predict the output. The free-run simulation case corresponds to the use of the model in *parallel mode*, whereas the one-step-ahead predictions correspond to a *series-parallel mode* of operation, as suggested by Narendra and Parthasarathy (1990). Figure 2 illustrates the series-parallel and parallel modes of operation. The term q^{-1} represents the unit-delay operator, in which $u(n)q^{-1} = u(n-1)$. The term $\xi(n)$ is called residual, and it is given by $\xi(n) = y(n) - \hat{y}(n)$. It is well-known in the system identification literature that model evaluation based on one-step-ahead predictions solely can result in wrong conclusions about the system dynamics (AGUIRRE, 2007; BILLINGS, 2013). Essentially, the one-step-ahead predictions will be close to the observed outputs when the sampling rate is high relative to the system dynamics (NORGAARD et al., 2000).

The *root mean squared error* (RMSE) between predictions and actual observations is a common choice to evaluate the goodness of fit of the model, i.e. it summarizes the discrepancy between the observed values and the predicted ones under the model in question:

$$RMSE(g) = \sqrt{\frac{1}{N_t} \sum_{n=1}^{N_t} (y(n) - g(\mathbf{x}(n)))^2}, \tag{1.8}$$

[2] By linear mappings we also mean nonlinear ones which can be written as a linear-in-the-parameters model.



(a) Series-parallel mode (b) Parallel mode

Figure 2: The two configurations for model's operation.

where N_t represents the number of test samples used for model validation. In principle, the smaller the RMSE, the better the reconstruction of the system dynamics. In this thesis, we use the term *out-of-sample error* to refer to the RMSE calculated over test/validation samples, while *in-sample error* is the RMSE for training/estimation samples.

In cases where it is not possible to take some samples out for validation, we can still validate our models by statistical residual analysis. The rationale for applying residual analysis is to verify if the residuals achieved by the final model on the estimation set are unpredictable. One of the most well-known approaches consists of the correlation tests proposed by Billings and Voon (1986) for validating nonlinear models:

$$\begin{aligned}
\rho_{\xi\xi} &= E\{\xi(n-\tau)\xi(n)\} = \delta(\tau), \\
\rho_{u\xi} &= E\{u(n-\tau)\xi(n)\} = 0, \forall\tau, \\
\rho_{u^2\xi} &= E\{(u^2(n-\tau) - \overline{u^2(n)})\xi(n)\} = 0, \forall\tau, \\
\rho_{u^2\xi^2} &= E\{(u^2(n-\tau) - \overline{u^2(n)})\xi^2(n)\} = 0, \forall\tau, \\
\rho_{\xi(\xi u)} &= E\{\xi(n)\xi(n-1-\tau)u(n-1-\tau)\} = 0, \forall\tau > 0,
\end{aligned}$$

where $\delta(0)$ is the Dirac delta function, $\overline{x}$ denotes the mean of x and $\xi(n)$ corresponds to the residual sequence $y(n) - \hat{y}(n)$ in the estimation set.

It is worth mentioning that model validation uniquely based on statistical residual analysis can be misleading. This may be because overparametrized models may be able to fit the estimation data completely, such that no residual is produced. However, overparametrized models have low generalization capability and can present spurious dynamics (AGUIRRE; BILLINGS, 1995). This is especially the case for polynomials and neural networks, where a poor design can easily lead to a high number of degrees of freedom. Equally, model validation only based on visual inspection between time series

of the free-run simulation and the actual system has to be carried out carefully. This is especially true when a low signal-to-noise ratio is present, since the amount of noise tends to corrupt the free-run simulation over time and predictions can look quite different from measurements on the system.

In this thesis, we occasionally evaluate models in terms of their capability in reconstructing *static nonlinearities*. The static behavior of a dynamic system in this thesis is represented by *stable fixed points*. In order to determine the stable fixed points, we adopt a simulation procedure. It consists in keeping the input constant (with outputs set to zero) and running the system until the variation on the outputs is less than a predefined threshold. For the interested reader, a review of different validation approaches can be found in Aguirre and Letellier (2009).

1.1.5 A synthetic example

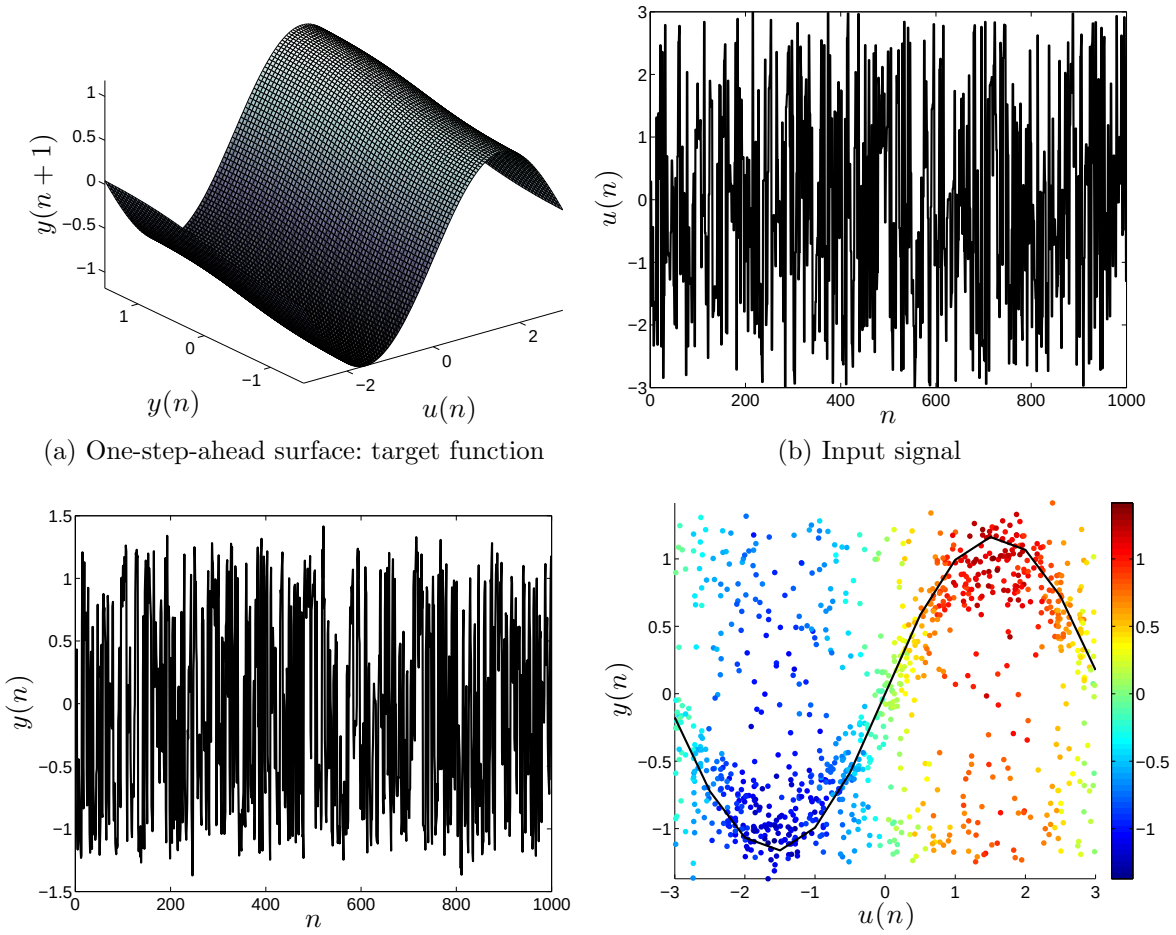
In order to illustrate the advantages and drawbacks of the methods proposed in this thesis, an artificial plant will be used repeatedly as a running example. This artificial plant has been evaluated by Gregorcic and Lightbody (2008). It consists of a discrete-time SISO system corrupted by additive noise given by

$$y(n+1) = 0.2\tanh(y(n)) + \sin(u(n)) + \epsilon(n). \tag{1.9}$$

The artificial plant is a nonlinear dynamic system whose outputs depend on previous input and output. It corresponds to a NARX system with $n_u = n_y = 1$ and Figure 3a shows the target function, i.e. the deterministic part of Eq. (1.9). For experimental evaluation, we generated a dataset which contains 1,000 samples with a uniformly distributed random input signal, $u(n) \sim \mathcal{U}[-3, 3]$, and $\epsilon(n) \sim \mathcal{N}(0, 0.01)$. The input and output signals are illustrated in Figures 3b and 3c respectively. For all experiments, we use 70% of the samples for training (model selection) and the remaining samples for testing (model validation) purposes. We have not carried out any normalization or pre-processing steps over the data samples.

Figure 3d depicts the data where colors denote the response ($y(n+1)$) along with the static equilibrium curve of the system, which represents its static behavior. To estimate the static behavior of the system, the input signal has been kept constant until the system to converge. A set of such collections with $u(n)$ varying from -3 to 3 were performed and the static behavior is shown by the solid black line in Figure 3d.

Later (in Chapter 5), a number of real-world datasets will be used for comparison and performance assessments of the models studied in this thesis.



(a) One-step-ahead surface: target function

(b) Input signal

(c) Output signal

(d) Data and static behavior (solid black line)

Figure 3: Artificial data created from Eq. (1.9).

1.2 Objectives of the thesis

The overall objective of this thesis is to propose novel approaches for nonlinear system identification using machine learning methods, which aims to produce accurate models at a low computational cost. As a consequence, we expect the novel proposals to represent feasible alternatives to classic reference methods in black-box modeling. In order to accomplish that, we may specifically follow the next steps:

- evaluate the use of learning models based on neural networks to SISO nonlinear system identification;
- propose a novel distance-based regression approach for supervised learning and evaluate its application on system identification tasks;
- propose a new approach for system identification and a novel method based on self-organizing neural networks;

- compare the different methods and paradigms on synthetic and real-world benchmarking system identification problems.

1.3 Chapter organization and contributions

In Chapter 2, global and local learning approaches that have been applied to system identification are described. It includes linear and nonlinear methods, such as a global linear method that uses *ordinary least square* (LS) for parameter estimation, and neural network architectures: *Multilayer Perceptrons*, *Extreme Learning Machines* and *Radial Basis Functions* networks. Also, a number of local approaches for system identification are discussed, such as *Local Linear Regression* approaches, and methods based on the *Self-Organizing Map* (SOM).

In Chapter 3, a novel distance-based regression method for supervised learning, called *Minimal Learning Machine* (MLM), is described and its application for system identification is illustrated. Original contributions of this chapter include:

- formulating a completely novel method called the Minimal Learning Machine. It includes the basic formulation and a comprehensive discussion on computational complexity and hyper-parameters.
- discussion on the links between MLM and classic methods, such as radial basis function networks;
- the proposal of supervised alternatives to the random selection of reference points in MLM.

In Chapter 4, we introduce a new modeling paradigm that stands between global and local approaches and it is called *regional modeling*. We describe the main characteristics of regional models and how we can use learning methods in the regional framework. Original contributions of this chapter include the following ones:

- the proposal of a new method for system identification via the clustering of the Self-Organizing Map.
- a robust extension of the proposed regional models through M-estimation (HUBER, 2004).

In Chapter 5, a set of experiments involving all the approaches studied throughout the thesis is reported. The experiments include tests on real-world system identification problems. A comparison performance analysis is also carried out.

In Chapter 6, we provide concluding remarks and discuss directions for future research on topics related to this thesis.

In Appendix A, we briefly describe the clustering algorithms used in this thesis.

In Appendix B, we extend the Minimal Learning Machine to classification problems, while evaluating its performance against standard classification methods. A binary classification example and four real-world problems are used for performance assessment.

In Appendix C, we evaluate the performance of the Minimal Learning Machine on eight regression problems and compare it against the state-of-the-art methods in machine learning.

1.4 Publications

During the development of the thesis, a number of articles have been published by the author. It includes articles which relate directly to thesis topics, as well as articles related to cooperation on projects between different research groups. Despite of that, all the articles are contributions on either Machine Learning/ Computational Intelligence or System Identification areas. The published articles related to the thesis are:

1. A. H. de Souza Junior, F. Corona, G. Barreto, Y. Miche and A. Lendasse, **Minimal Learning Machine: A novel supervised distance-based method for regression and classification**. *Neurocomputing*, to appear, 2014.

2. A. H. de Souza Junior, F. Corona and G. Barreto, **Regional models: A new approach for nonlinear dynamical system identification with the Self-Organizing Map**. *Neurocomputing*, vol. 147, n. 5, 36-45, 2015.

3. A. H. de Souza Junior, F. Corona and G. Barreto, **Minimal Learning Machine and Local Linear Regression for Nonlinear System Identification**. In *Proc. 20th Congresso Brasileiro de Automática*, 2066-2073, 2014.

4. A. H. de Souza Júnior, F. Corona, Y. Miche, A. Lendasse and G. A. Barreto, **Extending the Minimal Learning Machine for Pattern Classification**. In *Proc. 1st BRICS Countries Congress on Computational Intelligence - BRICS-CCI 2013*, 236-241, 2013.

5. A. H. de Souza Júnior, F. Corona, Y. Miché, A. Lendasse, G. Barreto and O. Simula, **Minimal learning machine: A new distance-based method for supervised learning**. In *Lecture Notes in Computer Science: Advances in Computational Intelligence*, F. Sandoval, A. Prieto, J. Cabestany and M. Graña Eds., vol. 7903, 408-415, 2013.

6. F. Corona, Z. Zhu, A. H. de Souza Júnior, M. Mulas, G. A. Barreto, and R. Baratti, **Monitoring diesel fuels with Supervised Distance Preserving Projections and Local Linear Regression**. In *Proc. 1st BRICS Countries Congress on Computational Intelligence - BRICS-CCI 2013*, 422-427, 2013.

7. A. H. de Souza Júnior, F. Corona and G. Barreto, **Robust regional modelling for nonlinear system identification using self-organising maps**. In *Advances in Intelligent Systems and Computing: Advances in Self-Organizing Maps*, P. A. Estévez, J. P. Príncipe and P. Zegers Eds., vol. 198, 215-224, 2013.

8. A. H. de Souza Júnior and G. Barreto, **Regional Models for Nonlinear System Identification Using the Self-Organizing Map**. In *Lecture Notes in Computer Science: Intelligent Data Engineering and Automated Learning - IDEAL 2012*, H. Yin, J. A. F. Costa and G. A. Barreto Eds., vol. 7435, 717-724, 2012.

Also, the following papers were produced during the thesis period as a result of research collaborations:

1. F. Corona, Z. Zhu, A. H. Souza Júnior, M. Mulas, E. Muru, L. Sassu, G. Barreto, R. Baratti. **Supervised Distance Preserving Projections: Applications in the quantitative analysis of diesel fuels and light cycle oils from spectra**. *Journal of Process Control*, under review, 2014.

2. R. L. Costalima, A. H. de Souza Junior, C. T. Souza and G. A. L. Campos, **MInD: don't use agents as objects**. In *Proc. 7th Conference on Artificial General Intelligence (Lectures Notes in Computer Science)*, vol. 8598, 234-237, 2014.

3. F. Corona, Z. Zhu, A. H. de Souza Júnior, M. Mulas and R. Baratti, **Spectroscopic monitoring of diesel fuels using Supervised Distance Preserving Projections**. In *Proc. IFAC 10th International Symposium on Dynamics and Control of Process Systems - DYCOPS 2013*, 63-68, 2013.

4. César L. C. Mattos, Amauri H. Souza Júnior, Ajalmar R. Neto, Guilherme Barreto, Ronaldo Ramos, Hélio Mazza and Márcio Mota, **A Novel Approach for Labelling Health Insurance Data for Cardiovascular Disease Risk Classification**. In: *11th Brazilian Congress on Computational Intelligence*, 1-6, 2013.

5. César L. C. Mattos, Amauri H. Souza Júnior, Ajalmar R. Neto, Guilherme Barreto, Ronaldo Ramos, Hélio Mazza and Márcio Mota, **Cardiovascular Disease Risk Classification From Health Insurance: A Big Data Experimental Comparison**. In: *Proc. 2nd Brazilian Conference on Intelligent Systems*, 1-12, 2013.

Chapter 2

Global and Local Learning Models for System Identification

Although several techniques for nonlinear dynamic system identification have been proposed, they can be categorized into one of the two following approaches: global and local modeling. On the one hand, global modeling involves the utilization of a single model structure that approximates the whole mapping between the input and the output of the system being identified. On the other hand, local modeling utilizes multiple models to represent the input-output dynamics of the system of interest (LAWRENCE; TSOI; BACK, 1996). In this chapter we briefly overview global and local strategies for nonlinear system identification. The emphasis of this chapter is on a basic understanding of the approaches that will be further used for comparison. In Section 2.1 we discuss global models and their application to the identification of nonlinear dynamic systems. Section 2.2 introduces local modeling techniques. Section 2.3 gives the closing remarks of the chapter.

2.1 Global modeling

In global modeling, the assumption is that the relationship between the input and output values can be described by a single analytical function over the whole input domain. In the literature related to NARX models, a number of global models have been proposed, such as *polynomials* (BILLINGS; CHEN; KORENBERG, 1989), *rational models* (BILLINGS; ZHU, 1991), *neural networks* (NARENDRA; PARTHASARATHY, 1990) and *wavelet networks* (CAO et al., 1995). A different line of action is that based on *block-oriented systems* in which a linear transfer function model is applied either before or after a nonlinear static mapping. These are called Hammerstein and Wiener models (BILLINGS; FAKHOURI, 1982). Also, Volterra models (CAMPELLO; FAVIER;

AMARAL, 2004) comprise classical global modeling attempts. In fact, global models constitute the mainstream in nonlinear system identification and control (YU, 2004; LI; YU, 2002; NARENDRA, 1996).

Due to the large number of global models, the focus of this thesis is on approaches from machine learning and computational intelligence fields, particularly, neural network based models. The field of *machine learning* abounds in efficient supervised methods and algorithms that can be equally applied to regression and classification tasks (BISHOP, 1995), and whose application in system identification is straightforward. An exhaustive and fair list of such methods would be hard to present here, but one can certainly mention as state-of-the-art methods the *multilayer perceptron* (MLP) (RUMELHART; HINTON; WILLIAMS, 1986), *radial basis functions networks* (RBF) (POGGIO; GIROSI, 1990; WU et al., 2012) *support vector regression* (SVR) (VAPNIK, 1998; SMOLA; SCHOLKOPF, 2004), as well as more recent approaches, such as those based on *extreme learning machine* (ELM) (HUANG; ZHU; ZIEW, 2006), *Gaussian processes* (RASMUSSEN; WILLIAMS, 2006) and *information-theoretic learning* (ITL) (PRINCIPE, 2010).

This section describes global approaches for nonlinear system identification, including the well-known *ordinary least square* (LS) estimation method, which is commonly used for estimating part of the nonlinear model's parameters. In Section 2.1.2, we briefly report neural network architectures and their application to system identification.

2.1.1 Least Squares and ARX models

As mentioned, the *ordinary least squares* method (RAO; TOUTENBURG, 1999) is an estimation method applied to linear-in-the-parameter models. Despite its early proposal, it is still one of the most used methods in machine learning and related areas.

We are interested in approximating a target function $f : \mathbb{X} \to \mathbb{Y}$ from empirical input and output observations $\{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$, with $\mathbf{x}_n \in \mathbb{R}^D$ and $y_n \in \mathbb{R}$, following the model in Eq. (1.1). For the task, one can use the linear model/hypothesis

$$h(\mathbf{x}) = \boldsymbol{\beta}^T \mathbf{x}, \tag{2.1}$$

where $h : \mathbb{R}^D \to \mathbb{R}$, $\boldsymbol{\beta} = \{\beta_j\}_{j=1}^{D}$ is the vector of parameters and $\mathbf{x} = \{x_j\}_{j=1}^{D}$ is the regression vector (or regressors).

As usual, we need to quantify how well h approximates f. A feasible choice to ensure approximation quality is the squared error $e(f(\mathbf{x}), h(\mathbf{x})) = (h(\mathbf{x}) - f(\mathbf{x}))^2$, which can be averaged over the whole set of observations: $\sum_{n=1}^{N}(h(\mathbf{x}_n) - f(\mathbf{x}_n))^2$. Unfortunately, we do not have access to f, so we may use the output measurements y to ensure good approximation to f. In doing so, we define the least square loss function J_{LS} by

$$J_{LS}(\boldsymbol{\beta}) = \sum_{n=1}^{N}(h(\mathbf{x}_n) - y_n)^2, \tag{2.2}$$

By minimizing 2.2 using the model 2.1 we have that the least square estimator is

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}. \tag{2.3}$$

where the matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]^T$ is a $N \times D$ matrix whose n-th row corresponds to the n-th regression vector $\mathbf{x}_n$. Similarly, the column vector $\mathbf{y}$ encompasses the N output observations in its rows.

We may assume that the dynamic SISO system we are working with can be described mathematically by the ARX (AutoRegressive with eXogenous input) model (LJUNG, 1999):

$$\begin{aligned} y(n) &= a_1 y(n-1) + \cdots + a_{n_y} y(n-n_y) + b_1 u(n-n_0-1) + \cdots + b_{n_u} u(n-n_0-n_u) + \epsilon(n), \\ &= \sum_{j=1}^{n_y} a_j y(n-j) + \sum_{l=1}^{n_u} b_l u(n-n_0-l) + \epsilon(n), \end{aligned} \tag{2.4}$$

where $u(n) \in \mathbb{R}$ and $y(n) \in \mathbb{R}$ denote, respectively, the input and output of the model at time step n, while $n_u \geq 1$ and $n_y \geq 1$ are the input-memory and output-memory orders, respectively. The error term $\epsilon(n)$ is assumed to follow a white noise process. The parameter n_0 ($n_0 \geq 0$) is a delay term, known as the process dead-time. Without lack of generality, we assume $n_0 = 0$ in this thesis, thus obtaining the following ARX model:

$$y(n) = \sum_{j=1}^{n_y} a_j y(n-j) + \sum_{l=1}^{n_u} b_l u(n-l) + \epsilon(n). \tag{2.5}$$

By defining the input vector $\mathbf{x}(n) \in \mathbb{R}^{n_u+n_y}$ at time step n and the vector of parameters $\boldsymbol{\beta} \in \mathbb{R}^{n_u+n_y}$ as

$$\mathbf{x}(n) = [y(n-1) \ \cdots \ y(n-n_y) \mid u(n-1) \ \cdots \ u(n-n_u)]^T, \tag{2.6}$$

$$\boldsymbol{\beta} = [a_1 \ \cdots \ a_{n_y} \mid b_1 \ \cdots \ b_{n_u}]^T, \tag{2.7}$$

we can write the output of the ARX model in Eq. (2.5) simply as

$$y(n) = \boldsymbol{\beta}^T\mathbf{x}(n) + \epsilon(n) \tag{2.8}$$

By comparing Eqs. (2.1) and (2.8), one may observe that the ARX model can use the LS estimation method to find the optimal vector of parameters in the least squares sense. For that, all that is required is to create the regressors following Eq. (2.6). Thus, the predictor (or final model) associated with the ARX model is $\hat{y}(n) = g(\mathbf{x}(n)) = \hat{\boldsymbol{\beta}}^T\mathbf{x}(n)$, with $\hat{\boldsymbol{\beta}}$ given by Eq. (2.3). Under the assumption that $\epsilon(n)$ is white noise, the LS estimator is unbiased and equivalent to the maximum likelihood estimator (AGUIRRE, 2007).

Since the LS estimation solution will be used extensively throughout the thesis, important remarks are necessary. An important issue is that the regression matrix $\mathbf{X}$ must

be full-rank. Otherwise, $\mathbf{X}^T\mathbf{X}$ does not have an inverse. Also, the number of rows N of $\mathbf{X}$ must exceed (or equal) the number of columns D in Eq. (2.3). If not, $\mathbf{X}^T\mathbf{X}$ is not full-rank and it may not correspond to the LS estimator. In those cases where $N < D$, a least-norm solution is usually preferred and it represents a regularized solution. Clearly, situations in which $N < D$ are undesired for estimation and data fitting purposes since an infinity number of solutions is possible. From now on, in the context of system identification, $D = n_y + n_u$.

2.1.2 Single-hidden layer feedforward networks

In this section, we briefly discuss *artificial neural networks* (ANN), specifically a special type of ANN called *single-hidden layer feedforward networks* (SLFN). The term "artificial neural networks" was originally motivated by the way that biological structures process information in the brains of humans and animals. However, eventually, most artificial neural networks used in engineering are at least as closely related to mathematics, statistics and optimization as to the biological model (NELLES, 2001). ANNs are based on processing units called *neurons* or nodes which are combined to form a network. An SLFN has one hidden layer of neurons that work in parallel. These neurons are connected to other neurons in the output layer. The layers may be organized in such a way that the input vector enters the hidden layer and then reaches the output layer. The mathematical representation of SLFN is given by

$$h(\mathbf{x}) = \sum_{l=1}^{L} \beta_l \phi(\mathbf{x}, \mathbf{m}_l) + \beta_0 = \boldsymbol{\beta}^T \mathbf{z}, \tag{2.9}$$

where L denotes the number of neurons in the hidden layer, $\boldsymbol{\beta} \in \mathbb{R}^{L+1}$ is the weight vector of the linear output unit, $\phi(\cdot) : \mathbb{R} \to \mathbb{R}$ is a nonlinear mapping, also called activation function, which depends on the input vector $\mathbf{x} \in \mathbb{R}^{n_y+n_u}$ and the vector of parameters $\mathbf{m}_l$ of the l-th hidden unit. The notation $\phi(\mathbf{x}, \mathbf{m}_l)$ stands for the fact that the fixed univariate mapping $\phi(\cdot)$ depends on the input vector and the parameters of the hidden units, and we can simplify the notation by expressing $\phi(\mathbf{x}, \mathbf{m}_l) = \phi_l(\mathbf{x})$. An alternative interpretation for Eq. (2.9) is that the vector $\mathbf{z}$ results of a nonlinear transform $\Phi : \mathbb{X} \to \mathbb{Z}$ on $\mathbf{x}$ such that $\mathbf{z} = [1, \phi_1(\mathbf{x}), \ldots, \phi_L(\mathbf{x})]^T$. Thus, we can write $\mathbf{z} = \Phi(\mathbf{x})$. In other words, the SLFN approaches correspond to linear models in a transformed space $\mathbb{Z}$. Figure 4 shows the structure of SLFNs, with the regression vector $\mathbf{x}$ comprised of input and output samples of a SISO system.

The SLFN approaches differ in the way they define $\phi(\cdot)$ and the parameters $\mathbf{m}_l$, how they combine $\mathbf{x}$ and $\mathbf{m}_l$ to provide a single variable as input for ϕ, and how they compute $\boldsymbol{\beta}$. In the following, we present the particularities of the *Radial Basis Functions Networks*, *Multilayer Perceptrons* and *Extreme Learning Machines*. For an overview of most neural network architectures, we recommend Haykin (2009). Bishop (2006) discusses

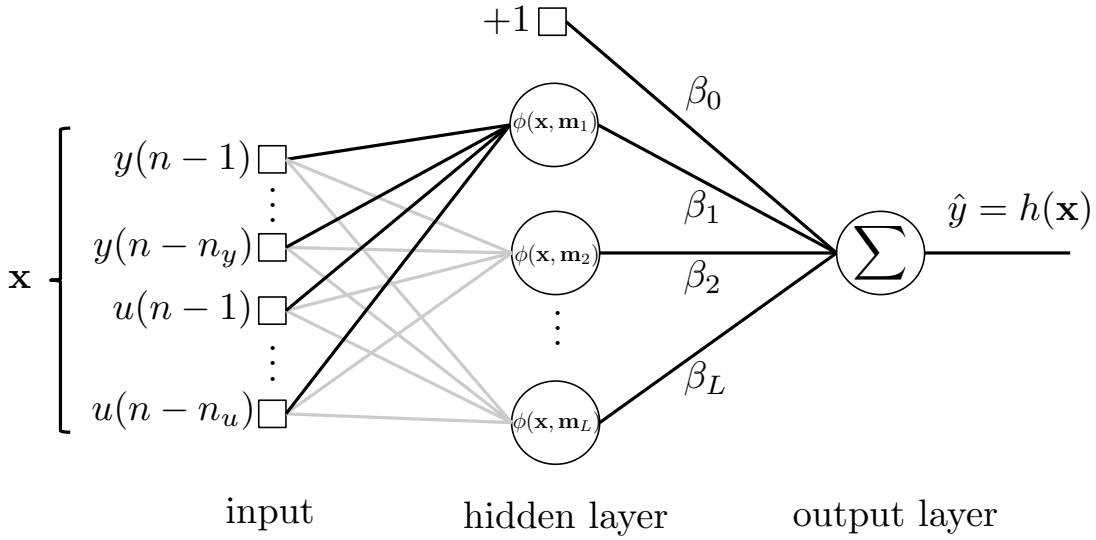


Figure 4: General structure of single-hidden layer feedforward networks.

the use of neural networks for pattern classification, including Bayesian approaches. For a review of neural networks for system identification, we suggest the books by Nelles (2001) and Norgaard et al. (2000), and the seminal paper by Narendra and Parthasarathy (1990).

2.1.2.1 MultiLayer Perceptrons

The *multilayer perceptrons* (MLP) (HAYKIN, 2009) is the most widely known and used neural network architecture. In the literature, the MLP is even used as a synonym for neural networks. The operation of MLPs is based on units or neurons called *perceptrons*. The operation of these neurons can be split into two parts. The first part consists of projecting the input vector $\mathbf{x}$ on the weights $\mathbf{m}_l$ through the inner product $\mathbf{m}_l^T\mathbf{x}$. Second, the nonlinear activation function ϕ transforms the projection result. If several perceptron neurons are used in parallel and connected to an output neuron, the MLP network with one hidden layer is obtained. The MLP can be mathematically formulated as

$$h(\mathbf{x}) = \sum_{l=1}^{L} \beta_l \phi(\mathbf{m}_l^T\mathbf{x}) + \beta_0 = \boldsymbol{\beta}^T\mathbf{z}, \tag{2.10}$$

where L is the number of hidden neurons, $\mathbf{x} \in \mathbb{R}^{n_y+n_u+1}$ is the input vector added a bias term $x_0 = 1$, $\mathbf{m}_l \in \mathbb{R}^{n_y+n_u+1}$ is the weight vector of the l-th hidden neuron, $\boldsymbol{\beta} \in \mathbb{R}^{L+1}$ is the weight vector of the output unit and, $\mathbf{z} = [1, \phi(\mathbf{m}_1^T\mathbf{x}), \ldots, \phi(\mathbf{m}_L^T\mathbf{x})]^T$ represents the projection of $\mathbf{x}$ in the space of the hidden layer. Common choices for the activation function $\phi(\cdot) : \mathbb{R} \to \mathbb{R}$ are the logistic function $\phi(\mathbf{m}_l^T\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{m}_l^T\mathbf{x})}$ and the hyperbolic tangent $\phi(\mathbf{m}_l^T\mathbf{x}) = \tanh(\mathbf{m}_l^T\mathbf{x})$. The two functions share the interesting property that their derivative can be expressed in terms of the function itself (NELLES, 2001).

The estimation of the parameters of an MLP, e.g. $\boldsymbol{\beta}$ and $\{\mathbf{m}_l\}_{l=1}^L$, is usually achieved by minimizing the difference between the network output $h(\mathbf{x}(n))$ and the measured output

$y(n)$ from the estimation samples $\{(\mathbf{x}(n), y(n))\}_{n=1}^{N}$. The typical approach to MLP training is the error back-propagation algorithm (RUMELHART; HINTONT; WILLIAMS, 1986). Essentially, this procedure applies chain rule for derivative calculations with respect to the parameters, which corresponds to a steepest descent algorithm. In fact, many nonlinear optimization techniques can be used to estimate the MLP parameters. A description of strategies for MLP training is beyond the scope of this thesis, and can be found in Haykin (2009), Nelles (2001) and Bishop (1995).

The MLP network, as described by Eq. (2.10), represents only the most commonly applied MLP type. Different variants exist. Sometimes the output neuron is not of the pure linear combination type but is chosen as a complete perceptron. This means that an additional activation function at the output is used. Another possible extension is the use of more than one hidden layer. In fact, it can be used as many hidden layer as necessary. SLFNs with several hidden layers has gained special attention from the machine learning community recently, under the general name of *deep networks*, since efficient learning algorithms for training such networks have been derived (BENGIO, 2009).

2.1.2.2 Extreme Learning Machines

The *Extreme Learning Machine* (ELM) is a class of SLFNs, recently proposed by Huang, Zhu and Ziew (2006), for which the weights from the inputs to the hidden neurons are randomly chosen, while only the weights from the hidden neurons to the output are analytically estimated. The hidden layer does not need to be tuned, and its parameters are independent of training data. According to Huang, Wang and Lan (2011), ELM offers significant advantages, such as a fast learning speed, ease of implementation, and less human intervention than more traditional SLFNs, such as the MLP network.

The structure of ELM is exactly the same as MLP, including the choice for the activation function ϕ. The difference between MLP and ELM relies on the learning algorithm, i.e., how the model parameters are estimated. Whereas the MLP is usually trained using nonlinear optimization techniques, the ELM training consists of two steps:

1. Random initialization of the weights in the hidden layer $\{\mathbf{m}_l\}_{l=1}^{L}$. Usually, the weight vectors $\mathbf{m}_l$ are randomly sampled from a uniform (or normal) distribution.

2. Estimation of the output weights by the LS algorithm.

For the second step in the ELM training, let $\mathbf{Z} = [\mathbf{z}(1)\ \mathbf{z}(2)\ \cdots\ \mathbf{z}(N)]^T$ be a $N \times (L+1)$ matrix whose N rows are the hidden-layer output vectors $\mathbf{z}(n) \in \mathbb{R}^{L+1}$, $n = 1, ..., N$, where N is the number of available training input patterns, and $\mathbf{z}(n) = [1, \phi(\mathbf{m}_1^T\mathbf{x}(n)), \ldots, \phi(\mathbf{m}_L^T\mathbf{x}(n))]^T$. Similarly, let $\mathbf{y} = [y(1)\ y(2)\ \cdots\ y(N)]^T$ be a $N \times 1$ vector whose the n-th row is the output observation $y(n)$ associated with the input pattern

$\mathbf{x}(n)$. Since the ELM is linear in the parameters $\boldsymbol{\beta}$, the weight vector $\boldsymbol{\beta}$ can be easily computed by means of the LS method as follows

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{Z}^T\mathbf{Z}\right)^{-1}\mathbf{Z}^T\mathbf{y}. \tag{2.11}$$

2.1.2.3 Radial Basis Function Networks

The *Radial Basis Function* network (RBF) (BUHMANN, 2003) is also a SLFN, with the difference being that the activation function ϕ, also called *radial basis function*, depends on a distance metric. A radial basis function approximation of a function $f : \mathbb{R}^{n_y+n_u} \rightarrow \mathbb{R}$ takes the form

$$h(\mathbf{x}) = \sum_{l=1}^{L} \beta_l \phi(\|\mathbf{x} - \mathbf{m}_l\|_2) + \beta_0, \tag{2.12}$$

where $\phi : [0, \infty) \rightarrow \mathbb{R}$ is a fixed univariate function, $\boldsymbol{\beta} = [\beta_0, \beta_1, \beta_2, \ldots, \beta_L]^T$ denotes a vector of coefficients usually estimated by the LS method and $\mathbf{m}_l \in \mathbb{R}^{n_y+n_u}$ corresponds to the centers of the radial basis functions $\phi(.)$. In other words, the RBF approximation is a linear combination of translates of a fixed function $\phi(\cdot)$ of a single real variable. In system identification applications, it is common to add an autoregressive and moving average linear terms to Eq. (2.12) (ALVES; CORREA; AGUIRRE, 2007).

Historically, radial basis functions were introduced in the context of exact function interpolation. Thus, usually the number of centers equals the number of learning points, $L = N$, and the centers are the learning points, $\mathbf{m}_n = \mathbf{x}(n)$, for $n = 1, \ldots, N$. In this case, the RBF approximation (Eq. 2.12) may fit the training examples perfectly. However, in machine learning applications, the output observations are noisy, and exact interpolation is undesirable because it may lead to overfitting.

The design of RBF models encompasses four basic decisions: *i*) The choice of radial basis functions $\phi(\cdot)$; *ii*) How to compute the coefficients $\{\beta_l\}_{l=0}^{L}$; *iii*) How to determine the number of radial basis function L; and *iv*) How to locate the centers $\mathbf{m}_l$.

Among a variety of radial basis functions, some well-known functions are listed below:

- Linear: $\phi(\|\mathbf{x} - \mathbf{m}_l\|) = \|\mathbf{x} - \mathbf{m}_l\|$.
- Cubic: $\phi(\|\mathbf{x} - \mathbf{m}_l\|) = \|\mathbf{x} - \mathbf{m}_l\|^3$.
- Multiquadric: $\phi(\|\mathbf{x} - \mathbf{m}_l\|) = \sqrt{\|\mathbf{x} - \mathbf{m}_l\|^2 + \sigma^2}$, where $\sigma \in \mathbb{R}$ is a constant.
- Gaussian: $\phi(\|\mathbf{x} - \mathbf{m}_l\|) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{m}_l\|^2}{\sigma_l^2}\right)$, where $\sigma_l \in \mathbb{R}$ is a constant.

The coefficients of the linear output unit $\boldsymbol{\beta}$ are generally computed using the LS algorithm. In doing so, let us define the matrix $\mathbf{Z} = \{z_{nl}\}$, with $z_{n0} = 1$ for all $n = 1, \ldots, N$,

and

$$z_{nl} = \phi(\|\mathbf{x}(n) - \mathbf{m}_l\|), \quad n = 1, \ldots, N; l = 1, \ldots, L.$$

where N again represents the number of training points. In addition, let us collect the output observations $y(n)$, $n = 1, \ldots, N$ in a column vector $\mathbf{y}$. Thus, the LS estimation of $\boldsymbol{\beta} \in \mathbb{R}^{L+1}$ is

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{Z}^T\mathbf{Z}\right)^{-1}\mathbf{Z}^T\mathbf{y}. \tag{2.13}$$

The number of radial basis function L, as well as the hyper-parameters σ_l, are usually optimized using sampling methods, like k-fold cross-validation. Regarding the choice of the centers of the radial basis functions $\mathbf{m}_l$, many approaches are feasible. The simplest strategy would be to select the center randomly from the training points $\{\mathbf{x}(n)\}$. Another possibility is through clustering on the input space, which constitutes the standard approach. Normally, the *k-means algorithm* (MACQUEEN, 1967) is used to provide prototypes as centers of the radial functions. The drawback of these methods is that the center location depends only on training data distribution in the input space. Thus, the complexity of the underlying function that has to be approximated is not taken into account. Supervised techniques that use output information to locate the centers where the underlying function to be approximated is more complex are usually desired. An efficient supervised learning approach for choosing centers is proposed by Chen, Cowan and Grant (1991). For an overview of different strategies for center placement in RBF networks, we recommend Sarra and Kansa (2009) and Nelles (2001).

2.2 Local modeling

Local modeling consists in a divide-and-conquer approach where the function approximation problem is solved by dividing it into simpler problems whose solutions can be combined to yield a solution to the original problem. These approaches have been a source of much interest because they have the ability to locally fit to the shape of an arbitrary surface. This feature is particularly important when the dynamic system characteristics vary considerably throughout the input space. Regarding learning models, there are two main different local modeling approaches: *i*) modular architectures, and *ii*) local approximating methods. Modular architectures consists of a number of modules (models) covering different parts of the input space. This is the idea of *operating regimes* which assume a partitioning of the operating range of the system in order to solve the modeling problem. Examples of modular architectures include the Local Model Network (LMN) (MURRAY-SMITH, 1994; HAMETNER; JAKUBEK, 2013), the Local Linear Mapping (LLM) (WALTER; RITTER; SCHULTEN, 1990), the Adaptive-Network-Based Fuzzy Inference System (ANFIS) (JANG, 1993), Takagi-Sugeno (TS) models (TAKAGI;

SUGENO, 1985), and Mixture of Experts (MEs) (JACOBS et al., 1991; LIMA; COELHO; VON ZUBEN, 2007).

Local approximating approaches turn the problem of function estimation into a problem of value estimation. In other words, they do not aim to return a complete description of the input/output mapping but rather to approximate the function in a neighborhood of the point to be predicted. Examples of local approximating methods are Lazy Learning (LL) (BONTEMPI; BIRATTARI; BERSINI, 1999), Locally Weighted Learning (LWL) (ATKESON; MOORE; SCHAAL, 1997), Local Linear Regression (LLR) (GUPTA; GARCIA; CHIN, 2008), and the kSOM model (SOUZA; BARRETO, 2010).

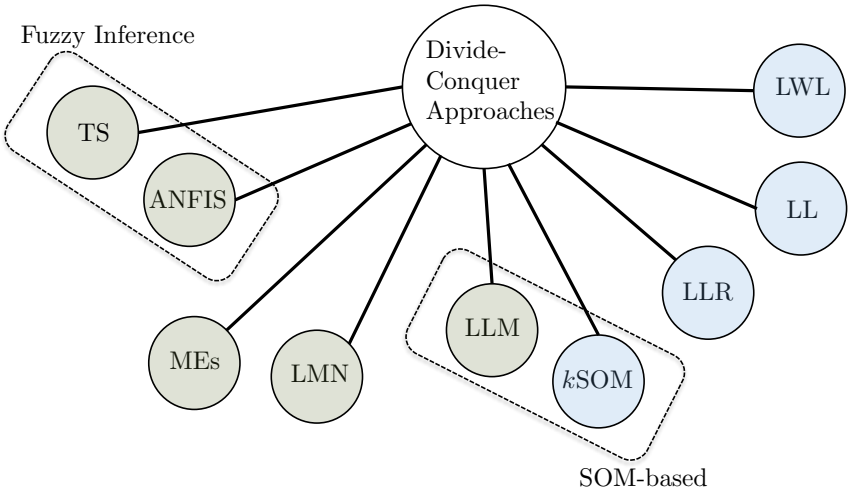


Figure 5: Taxonomy of divide-and-conquer approaches. Blue circles denote local approximating methods whereas green circles represent modular architectures.

Figure 5 summarizes the taxonomy of local models and it emphasizes methods based on the Self-Organizing Map (SOM) and Fuzzy Inference ones. Clearly, it does not aim to represent a comprehensive list of methods. In what follows, we discuss local modeling approaches, including both modular architectures and local approximating strategies.

2.2.1 Modular architectures

In this section, we describe classic modular approaches to the system identification problem. We focus on methods based on the *Self-Organizing Map* (SOM) (KOHONEN, 2013). First, we briefly introduce the Self-Organizing Map algorithm. Second, we explore an alternative for using the Self-Organizing Map to build regression models for system identification, the Local Linear Map (LLM) model.

2.2.1.1 The Self-Organizing Map

The *Self-Organizing Map* is a well-known competitive learning algorithm. The SOM learns from examples a mapping (projection) from a high-dimensional continuous input

space $\mathbb{X}$ onto a low-dimensional discrete space (lattice) $\mathbb{A}$ of Q neurons, which are arranged in fixed topological forms, e.g., as a rectangular 2-dimensional array. The map $i^*(\mathbf{x}) : \mathbb{X} \rightarrow \mathbb{A}$, defined by the set of prototypes or weight vectors $\mathcal{W} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_Q\}, \mathbf{w}_i \in \mathbb{R}^{n_u+n_y}$, assigns to an input vector $\mathbf{x} \in \mathbb{R}^{n_u+n_y}$ a *winning neuron* $i^* \in \mathbb{A}$, determined by

$$i^* = \arg\min_{\forall i} \|\mathbf{x} - \mathbf{w}_i\|, \tag{2.14}$$

where $\|\cdot\|$ denotes the Euclidean distance.

During SOM iterative training, at each training step j, one sample vector $\mathbf{x}(n(j))$ is randomly chosen from the input data set $\mathcal{X}$. The winning neuron and its topological neighbors are moved closer to the input vector by the following weight update rule:

$$\mathbf{w}_i(j+1) = \mathbf{w}_i(j) + \alpha(j)\eta(i^*, i; j)[\mathbf{x}(n(j)) - \mathbf{w}_i(j)] \tag{2.15}$$

where $0 < \alpha(j) < 1$ is the learning rate and $\eta(i^*, i; j)$ is a weighting function which limits the neighborhood of the winning neuron. A usual choice for $\eta(i^*, i; j)$ is given by the Gaussian function:

$$\eta(i^*, i; j) = \exp\left(-\frac{\|\mathbf{i}(j) - \mathbf{i}^*(j)\|^2}{2\sigma^2(j)}\right) \tag{2.16}$$

where $\mathbf{i}(j)$ and $\mathbf{i}^*(j)$ are respectively, the coordinates of the neurons i and i^* in the output array at time j, and $\sigma(j) > 0$ defines the radius of the neighborhood function at time j. The variables $\alpha(j)$ and $\sigma(j)$ should both decay with time to guarantee convergence of the weight vectors to stable steady states. In this thesis, we adopt an exponential decay for both variables $\alpha(j)$ and $\sigma(j)$.

Weight adjustment is performed until a steady state of global ordering of the weight vectors has been achieved. In this case, we say that the map has converged. The resulting map also preserves the topology of the input samples in the sense that adjacent patterns are mapped into adjacent regions on the map. Due to this topology-preserving property, the SOM is able to cluster input information and spatial relationships of the data on the map. Despite its simplicity, the SOM algorithm has been applied to a variety of complex problems (VAN HULLE, 2010; YIN, 2008; KOHONEN et al., 1996) and has become one of the most popular artificial neural networks.

2.2.1.2 The Local Linear Map

The first local modeling approach to be described has been called *Local Linear Map* (LLM) (WALTER; RITTER; SCHULTEN, 1990) by its proponents, and was originally applied to nonlinear time series prediction. The basic idea of the LLM is to associate each neuron in the SOM with a linear *finite impulse response* (FIR) filter trained with the *least mean squares* (LMS) adaptation algorithm (WIDROW, 2005). The SOM array is used to quantize the input space in a reduced number of prototype vectors (hence,

Voronoi regions), while the filter associated with the winning neuron provides a local linear estimate of the output of the mapping being approximated.

For modeling purposes, the input vector $\mathbf{x}(n)$ is built at each time step n by sliding through the input-output time series. Vector quantization of the input space $\mathbb{X}$ is performed by the LLM as in the usual SOM algorithm, with each neuron i owning a prototype vector $\mathbf{w}_i$, $i = 1, \ldots, Q$. In addition, associated with each weight vector $\mathbf{w}_i$, there is a coefficient vector $\boldsymbol{\beta}_i \in \mathbb{R}^{n_u+n_y}$ of the i-th local model.

The output value of the LLM-based local model is computed as

$$\hat{y} = \boldsymbol{\beta}_{i^*}^T \mathbf{x}, \tag{2.17}$$

where $\boldsymbol{\beta}_{i^*}$ is the coefficient vector associated with the winning neuron i^* for $\mathbf{x}$. From Eq. (2.17), one can easily note that the coefficient vector $\boldsymbol{\beta}_{i^*}$ is used to build a local linear approximation of the unknown global input-output mapping or target function.

Since the adjustable parameters of the LLM algorithm are the set of prototype vectors $\mathbf{w}_i$ and their associated coefficient vectors $\boldsymbol{\beta}_i, i = 1, 2, \ldots, Q$, we need two learning rules. The rule for updating the prototype vectors $\mathbf{w}_i$ follows exactly the one given in Eq. (2.15). The learning rule of the coefficient vectors $\boldsymbol{\beta}_i(j)$ is an extension of the normalized LMS algorithm, that also takes into account the influence of the neighborhood function $\eta(i^*, i; j)$:

$$\boldsymbol{\beta}_i(j+1) = \boldsymbol{\beta}_i(j) + \alpha'(j)\eta(i^*, i; j)\Delta\boldsymbol{\beta}_i(j), \tag{2.18}$$

where $0 < \alpha'(j) < 1$ denotes the learning rate of the coefficient vector, and $\Delta\boldsymbol{\beta}_i(j)$ is the error correction rule of Widrow-Hoff, given by

$$\Delta\boldsymbol{\beta}_i(j) = \left[y(n(j)) - \boldsymbol{\beta}_i^T(j)\mathbf{x}(n(j))\right] \frac{\mathbf{x}(n(j))}{\|\mathbf{x}(n(j))\|^2}, \tag{2.19}$$

where $\mathbf{x}(n(j))$ and $y(n(j))$ are randomly chosen input sample and corresponding output from the training sets $\mathcal{X}$ and $\mathcal{Y}$. After trained for a large number of iterations, the weight vectors $\mathbf{w}_i(j)$ of the SOM and the associated coefficient vectors $\boldsymbol{\beta}_i(j)$ are "frozen".

The basic idea behind the SOM-based local approaches is the partitioning of the input space into non-overlapping regions, called Voronoi cells, whose centroids correspond to the weight vectors of the SOM. Then an interpolating hyperplane is associated with each Voronoi cell or a small subset of them, in order to estimate the output value of the target function.

2.2.2 Local approximating models

Local approximating strategies have been proposed in the literature of system identification and machine learning under many different names, such as just-in-time models, prototype-based or memory-based methods, lazy learning, and look-up tables. In

this section, we focus on the SOM-based approaches and we discuss the recently proposed local linear regression methods with enclosing neighborhood.

2.2.2.1 The VQTAM Approach

The *Vector-Quantized Temporal Associative Memory* (VQTAM) (BARRETO; ARAÚJO, 2004) approach aims to build an input-output associative mapping using the *Self-Organizing Map*. It is a generalization to the temporal domain of a SOM-based associative memory technique that has been used by many authors to learn static (memoryless) input-output mappings, especially within the domain of robotics.

According to the VQTAM framework for system identification, during the training process, the training set $\mathcal{X}$ has its elements $\mathbf{x}(n)$ augmented by incorporating output information $y(n)$. In doing so, the augmented input vector $\mathbf{x}^{aug}(n) \in \mathbb{R}^{n_u+n_y+1}$ is given by

$$\mathbf{x}^{aug}(n) = [\mathbf{x}(n), y(n)] = [y(n-1), \ldots, y(n-n_y), u(n-1), \ldots, u(n-n_u), y(n)].$$

As expected, the augmented prototypes of the SOM network $\mathbf{w}_i^{aug} \in \mathbb{R}^{n_u+n_y+1}$ can be decomposed into two parts as well. The first part, denoted by $\mathbf{w}_i \in \mathbb{R}^{n_y+n_u}$, carries information about the original input space. The second part, denoted $w_i^{out} \in \mathbb{R}$, represents the coordinate associated with output information, such that $\mathbf{w}_i^{aug} = [\mathbf{w}_i, w_i^{out}]$.

The VQTAM learning algorithm differs slightly from the basic SOM algorithm. The winning neuron is determined based only on the original input space, such that

$$i^* = \underset{\forall i \in \mathbb{A}}{\operatorname{argmin}} \{\|\mathbf{x} - \mathbf{w}_i\|\}. \tag{2.20}$$

In the iterative weight updating step, the augmented vectors are used:

$$\mathbf{w}_i^{aug}(j+1) = \mathbf{w}_i^{aug}(j) + \alpha(j)\eta(i^*, i; j)[\mathbf{x}^{aug}(n(j)) - \mathbf{w}_i^{aug}(j)], \tag{2.21}$$

where $\mathbf{x}^{aug}(n(j))$ is the augmented vector randomly selected at iteration j.

Once VQTAM has been trained, the estimated output $\hat{y}$ for an input test point $\mathbf{x}$ is simply computed as $\hat{y} = w_{i^*}^{out}$, where i^* refers to the winning neuron to $\mathbf{x}$.

2.2.2.2 The kSOM Model

The kSOM was introduced by Souza and Barreto (2010) and originally evaluated on inverse system identification tasks. The kSOM model uses a single local linear model whose vector of coefficients is time-varying, i.e. it is computed at each iteration based on the prototype vectors of the neuron nearest to the current input vectors and of its K nearest neighbors.

The idea behind the kSOM is to train firstly the VQTAM as described in Section 2.2.2.1, in order to have a reduced representation of the input-output mapping encoded

in the weight vectors of the VQTAM. Then, for an out-of-sample input vector $\mathbf{x}$, the coefficients of a linear local model are estimated using the weight vectors of the K first winning neurons $\{i_1^*, i_2^*, \ldots, i_K^*\}$. These neurons are selected as follows:

$$\begin{aligned} i_1^* &= \underset{\forall i}{\operatorname{argmin}} \{\|\mathbf{x} - \mathbf{w}_i\|\} \\ i_2^* &= \underset{\forall i \neq i_1^*}{\operatorname{argmin}} \{\|\mathbf{x} - \mathbf{w}_i\|\} \\ &\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ i_K^* &= \underset{\forall i \neq \{i_1^*, \ldots, i_{K-1}^*\}}{\operatorname{argmin}} \{\|\mathbf{x} - \mathbf{w}_i\|\} \end{aligned} \tag{2.22}$$

Let the set of K winning augmented weight vectors be denoted by $\{\mathbf{w}_{i_1^*}^{aug}, \mathbf{w}_{i_2^*}^{aug}, \ldots, \mathbf{w}_{i_K^*}^{aug}\}$. Recall that, due to the VQTAM training style, each augmented weight vector $\mathbf{w}_i^{aug}$ has a portion associated with elements of the original input space $\mathbf{x}(n)$ and another associated with $y(n)$. So, the kSOM uses the corresponding K pairs of prototype vectors $\{\mathbf{w}_{i_k^*}, w_{i_k^*}^{out}\}_{k=1}^K$, with the aim of building a local linear mapping:

$$w_{i_k^*}^{out} = \boldsymbol{\beta}^T \mathbf{w}_{i_k^*}, \qquad k = 1, \ldots, K \tag{2.23}$$

where $\boldsymbol{\beta} \in \mathbb{R}^{n_y + n_u}$ is a coefficient vector. Eq. (2.23) can be written in a matrix form as

$$\mathbf{w}^{out} = \mathbf{W}\boldsymbol{\beta}, \tag{2.24}$$

where the output vector $\mathbf{w}^{out} \in \mathbb{R}^K$ and the regression matrix $\mathbf{W} \in \mathbb{R}^{K \times (n_u + n_y)}$ are defined as follows

$$\mathbf{w}^{out} = \begin{bmatrix} w_{i_1^*}^{out} & w_{i_2^*}^{out} & \cdots & w_{i_K^*}^{out} \end{bmatrix}^T \tag{2.25}$$

and

$$\mathbf{W} = \begin{pmatrix} w_{i_1^*,1} & w_{i_1^*,2} & \cdots & w_{i_1^*,n_u+n_y} \\ w_{i_2^*,1} & w_{i_2^*,2} & \cdots & w_{i_2^*,n_u+n_y} \\ \vdots & \vdots & \vdots & \vdots \\ w_{i_K^*,1} & w_{i_K^*,2} & \cdots & w_{i_K^*,n_u+n_y} \end{pmatrix}. \tag{2.26}$$

In practice, since we usually have $n_u + n_y < K$, the matrix $\mathbf{W}$ is non-square. In this case, we estimate the coefficient vector $\boldsymbol{\beta}$ by means of the regularized least-squares method:

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{W}^T\mathbf{W} + \lambda\mathbf{I}\right)^{-1} \mathbf{W}^T \mathbf{w}^{out}, \tag{2.27}$$

where $\mathbf{I}$ is the identity matrix of order $n_u + n_y$, and $\lambda > 0$ (e.g. $\lambda = 0.001$) is a small constant added to the diagonal of $\mathbf{W}^T\mathbf{W}$ to make sure that this matrix is full rank. Once $\boldsymbol{\beta}$ is estimated, we can locally approximate the output of $\mathbf{x}$ as:

$$\hat{y} \;=\; \hat{\boldsymbol{\beta}}^T \mathbf{x}.$$

An approach quite similar to kSOM was introduced by Principe, Wang and Motter (1998) and used for inverse local modeling by Cho et al. (2007), Cho et al. (2006). In this architecture, the required prototype vectors are not selected as the K nearest prototypes to the current input vector, but rather automatically selected as the winning prototype at time j and its $K-1$ topological neighbors. If a perfect topology preservation is achieved during SOM training, the neurons in the topological neighborhood of the winning prototype are also the nearest ones to the current input vector. However, if topological defects are present, as usually occurs for multidimensional data, this property cannot be guaranteed. Thus, the use of this architecture is limited to topology-preserving VQ algorithms. The kSOM model, however, is general enough to be used with different types of VQ algorithms.

2.2.2.3 Local Linear Regression

Local Linear Regression (LLR) (GUPTA; GARCIA; CHIN, 2008) is a nonlinear estimation approach. The spirit of LLR is that, over a small subset of the input domain, a simple linear model can approximate sufficiently well the true mapping to the output. LLR retains the simplicity of a global linear model and can overcome its low accuracy (ZHU et al., 2011).

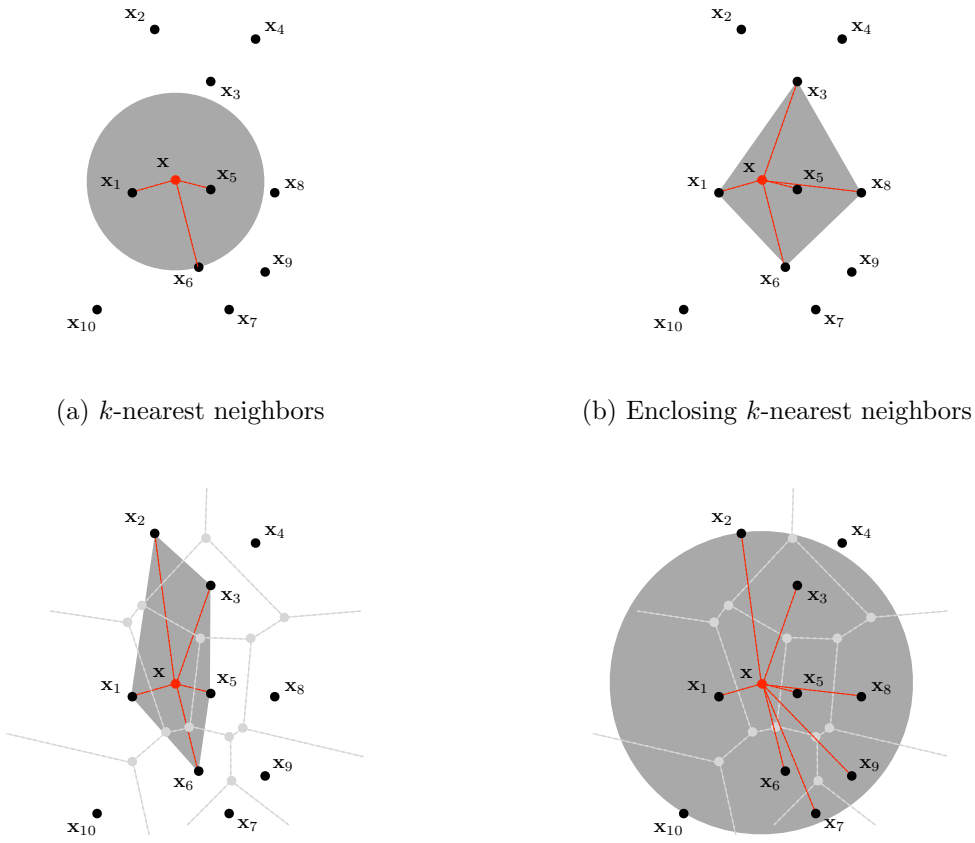
Again, we are given N training points $\mathbb{X} \rightarrow \mathbb{Y} = \{(\mathbf{x}(1), y(1)), \ldots, (\mathbf{x}(N), y(N))\}$, where $\mathbf{x}(n) \in \mathbb{R}^{n_y+n_u}$ and $y(n) \in \mathbb{R}$. For an arbitrary input test point $\mathbf{x} \in \mathbb{R}^{n_u+n_y}$, LLR estimates its output as $\hat{y} = \hat{\boldsymbol{\beta}}^T\mathbf{x} + \hat{\beta}_0$, the least-squares hyperplane over the neighborhood $\mathcal{J}_\mathbf{x}$ of $\mathbf{x}$:

$$(\hat{\boldsymbol{\beta}}, \hat{\beta}_0) = \underset{\boldsymbol{\beta},\beta_0}{\operatorname{argmin}} \sum_{\mathbf{x}(n)\in\mathcal{J}_\mathbf{x}} (y(n) - \boldsymbol{\beta}^T\mathbf{x} - \beta_0)^2.$$

The definition of the neighborhood and the number of neighbors are crucial for local linear regression. In this section, we briefly define four major neighborhood definition strategies for LLR from a geometrical point of view.

Classic *k-nearest neighbors* (kNN) define a neighborhood $\mathcal{J}_\mathbf{x}^{kNN}$ of $\mathbf{x}$ using k of its neighbors, according to a specified distance metric. Usually, the Euclidean metric is used and the number of neighbors k is fixed or cross-validated. Despite its simplicity, one major problem in kNN is the selection of the neighborhood size: *i*) too few neighbors may lead to a neighborhood that does not enclose the test point, which might give a large estimation variance and, *ii*) too many neighbors to impose enclosure may cause the model to over-smooth. How to select adaptively k is an open issue.

If $\mathcal{J}_\mathbf{x}$ encloses $\mathbf{x}$, we call it an enclosing neighborhood, i.e., $\mathbf{x} \in \texttt{conv}(\mathcal{J}_\mathbf{x})$, where the convex hull of a point set $\mathcal{S}=\{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ is defined as $\texttt{conv}(\mathcal{S})=\{\sum_{i=1}^n \omega_i\mathbf{s}_i \mid \sum_{i=1}^n \omega_i = 1, \omega_i \geq 0\}$. Figure 6a shows a kNN neighborhood of size $k=3$ for a test point $\mathbf{x}$. Recently, Gupta, Garcia and Chin (2008) proved that if a test point is in the convex hull enclosing its neighborhood, then the variance of the local linear regression estimate is bounded by the



(a) k-nearest neighbors

(b) Enclosing k-nearest neighbors

(c) Natural neighbors

(d) Natural neighbors inclusive

Figure 6: Neighborhoods: a) $\mathcal{J}_{\mathbf{x}}^{kNN}=\{\mathbf{x}_1,\mathbf{x}_5,\mathbf{x}_6\}$, b) $\mathcal{J}_{\mathbf{x}}^{ekNN}=\{\mathbf{x}_1,\mathbf{x}_3,\mathbf{x}_5,\mathbf{x}_6,\mathbf{x}_8\}$, c) $\mathcal{J}_{\mathbf{x}}^{NN}=\{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_5,\mathbf{x}_6\}$, and d) $\mathcal{J}_{\mathbf{x}}^{NNi}=\{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_5,\mathbf{x}_6,\mathbf{x}_7,\mathbf{x}_8,\mathbf{x}_9\}$.

variance of the measurement noise. Such a property is fundamental to limit erratic results. In the following, three enclosing neighborhood definition strategies are briefly overviewed.

Enclosing k-nearest neighbors (ekNN) It is based on the kNNs of $\mathbf{x}$ and extends them to define a neighborhood that encloses it. ekNN is the neighborhood of the kNNs with the smallest k such that $\mathbf{x} \in \texttt{conv}(\mathcal{J}_{\mathbf{x}}(k))$, where $\mathcal{J}_{\mathbf{x}}(k)$ is the set of kNNs of $\mathbf{x}$ (GUPTA; GARCIA; CHIN, 2008). If $\mathbf{x}$ is outside of the convex hull of the set $\mathcal{X}$, no such k exists. Define *distance to enclosure* as $d(\mathbf{x},\mathcal{J}_{\mathbf{x}}) = \min_{\mathbf{z}\in\texttt{conv}(\mathcal{J}_{\mathbf{x}})} \|\mathbf{x}-\mathbf{z}\|_2$, where $\mathbf{z}$ is any point in the convex hull around the neighborhood of $\mathbf{x}$, as demonstrated in Figure 6b. Note that $d(\mathbf{x},\mathcal{J}_{\mathbf{x}}) = 0$ only if $\mathbf{x} \in \texttt{conv}(\mathcal{J}_{\mathbf{x}})$. Then, the e$k$NN neighborhood is $\mathcal{J}_{\mathbf{x}}(k^*)$ with $k^* = \min_k\{k \mid d(\mathbf{x},\mathcal{J}_{\mathbf{x}}(k)) = d(\mathbf{x},\mathcal{X})\}$. The complexity for building a convex hull using k neighbors is $O(k^{\lfloor (n_y+n_u)/2 \rfloor})$, where $\lfloor\cdot\rfloor$ is the floor function.

Natural neighbors (NN) Natural neighbors are based on the Voronoi tessellation of the training samples and the test point. The natural neighbors of $\mathbf{x}$ are defined as those

points whose Voronoi cells are adjacent to the cell including $\mathbf{x}$. Natural neighbors have the so-called *local coordinates property*, which is used to prove that the natural neighbors form an enclosing neighborhood if $\mathbf{x} \in \texttt{conv}(\mathcal{X})$. Figure 6c shows an example of natural neighbors for the point $\mathbf{x}$.

Natural neighbors inclusive (NNi) In some cases of non-uniformly distributed local areas, a training point which is far from the test point can be one of its natural neighbors, but a nearer point is excluded for its neighborhood. To overcome this situation, natural neighbors inclusive has been proposed to include both the natural neighbors and those training points within the distance to the furthest natural neighbor. That is, $\mathcal{J}_{\mathbf{x}}^{NNi} = \{\mathbf{x}(j) \in \mathcal{X} \mid \|\mathbf{x} - \mathbf{x}(j)\| \leq \max_{\mathbf{x}(n) \in \mathcal{J}_{\mathbf{x}}^{NN}} \|\mathbf{x} - \mathbf{x}(n)\|\}$. Figure 6d is an example of natural neighbors inclusive.

2.3 Conclusions

This chapter overviewed global and local learning methods with direct application to system identification tasks. In the realm of global models, the focus was on neural networks architectures. Thus, we discussed MLP, ELM and RBF networks. With respect to local models, we introduced the Local Linear Regression approaches, and SOM-based architectures, which mainly included LLM and KSOM algorithm. The architectures studied in this chapter are going to be further used for comparison purposes.

Chapter 3

The Minimal Learning Machine

Supervised machine learning methods for regression and classification have been designed mostly for data types that lie in vector spaces, i.e. response (i.e. output) and/or predictor (i.e. input) variables are often arranged into vectors of predefined dimensionality. There are other types of data, however, such as graphs, sequences, shapes, images, trees and covariance matrices, which are less amenable to being treated within standard regression/classification frameworks. These data types usually do not lie in a natural vector space, but rather in a metric space.

For this type of data, usually referred to as *structured data* (HAGENBUCHNER; SPERDUTI; TSOI, 2003; HAMMER et al., 2004), a more general approach to the characterization of the data items is to define a distance (or dissimilarity) measure between data items and to provide a learning algorithm that works with the resulting distance matrix. Since pairwise distance measures can be defined on structured objects (e.g. graphs, trees or strings), this procedure provides a bridge between the classical and the structural/syntactic approaches to pattern recognition (BUNKE, 1993; SCHALKOFF, 1992).

Pairwise distance data occur frequently in empirical sciences, such as psychology, economics, ecology and biochemistry, with most of the algorithms developed to handle this kind of data falling into the realm of unsupervised learning, predominantly as clustering (HAMMER; HASENFUSS, 2010; SOMERVUO; KOHONEN, 1999; GRAEPEL; OBERMAYER, 1999; HOFMANN; BUHMANN, 1997) or multidimensional scaling algorithms (WANG, 2011).

For regression tasks, there are prior works in which the response and/or the predictor variables are expressed as distance (i.e. dissimilarity) matrices. Cuadras and Arenas (1990) proposed an approach to regression where only the predictors are expressed as a distance and classical multidimensional scaling (a.k.a principal coordinates analysis) (WANG, 2011) is used to generate scores. The response variable is then regressed on these scores. McArdle

and Anderson (2001) performed MANOVA[1] on ecological data with only knowledge of the distance matrix of the response variable. Finally, Lichstein (2006) proposed a modeling approach where both the response and predictor variables are represented as distance matrices. However, since their method converts the distance matrices to vectors in a column-wise fashion, useful information provided by the geometry of the problem is lost.

For classification tasks, we refer the reader to the works of Hammer et al. (2014), Zhu, Schleif and Hammer (2014), and Graepel et al. (1999). Roughly speaking, these works introduce extended versions of classification algorithms for data characterization by means of a matrix of pairwise similarities or more general dissimilarity measures, rather than explicit feature vectors. In Hammer et al. (2014) the authors propose a general learning framework that unifies previous attempts of making LVQ algorithms capable of handling non-vectorial data, such as kernel GLVQ (SCHLEIF et al., 2012; QIN; SUGANTHAN, 2004) and relational GLVQ (GISBRECHT et al., 2012). This is possible by means of a pseudo-Euclidean embedding[2] of similarity (or dissimilarity) data, i.e., every finite data set which is characterized by pairwise similarities or dissimilarities can be embedded in a so-called pseudo-Euclidean vector space. Zhu, Schleif and Hammer (2014) proposes an LVQ-based classifier that, in addition to its ability to directly deal with arbitrary symmetric dissimilarity matrices, provides confidence/reliability measures for the classification results. Finally, in the pioneering work by Graepel et al. (1999), they suggested classification algorithms based on linear models which operate on distance data from both Euclidean and pseudo-Euclidean spaces.

As mentioned in the previous paragraphs, data characterization by means of pairwise dissimilarity measures have been associated with the processing of structured data, either for regression or classification purposes. However, we argue that the use of dissimilarity measures for data characterization may also be beneficial for the processing of unstructured data types, by allowing, for example, a nonlinear learning problem to be tackled by linear models. Bearing this in mind, we introduce a new supervised nonparametric method, called the Minimal Learning Machine (MLM), aiming at the efficient design of distance-based regression models or pattern classifiers for unstructured data types.

Learning in MLM consists in building a linear mapping between input and output distance matrices. In the generalization phase, the learned distance map is used to provide an estimate of the distance from M output reference points to the target output value. Then, the output point estimation is formulated as a multilateration problem based on the predicted output distance and the locations of the reference points. Given its general

[1] Acronym for multivariate analysis of variance.

[2] Non-Euclidean dissimilarities arise naturally when we want to build a measure that incorporates important knowledge about, e.g. the relation between objects to be classified. Pseudo-Euclidean embedding allows one to embed such dissimilarities in a vector space in order to use standard (Euclidean) classification tools.

formulation, the Minimal Learning Machine is inherently capable of operating on nonlinear regression problems as well as on multidimensional response spaces. Because of that, the MLM can be extended for classification in a straightforward manner (Appendix B) (SOUZA JUNIOR et al., 2013b). One of the main advantages of the MLM is that it requires tuning of a single parameter, which is the number of reference points used to obtain an estimate of the response variable.

Based on experiments on regression (Appendix C) and classification (Appendix B) problems, the proposed distance-based method, when applied to standard vectorial (i.e. unstructured) data types, achieves accuracies that are comparable to those achieved by standard supervised nonlinear machine learning methods, thus offering a simpler alternative to these nonlinear approaches. Results on system identification problems, further reported in Chapter 5, also highlight that the MLM does not suffer from numerical problems commonly present in reference nonlinear global models, outperforming them.

The remainder of the chapter is organized as follows. In Section 3.1, the Minimal Learning Machine is presented. Section 3.2 discusses the computational complexity of MLM and its hyperparameter. In Section 3.3, we describe the links between MLM and two other methods: RBF network and Kernel Dependency Estimation. Section 3.4 extends the MLM to system identification tasks, which is the focus of the thesis. Section 3.5 describes some strategies to the selection of reference points in MLM. In Section 3.6, we illustrate the performance of the MLM on two synthetic examples. Finally, Section 3.7 presents concluding remarks.

3.1 Basic formulation

In this section, we formulate the *Minimal Learning Machine* (MLM) (SOUZA JUNIOR et al., 2013a) and we discuss its computational complexity and its hyperparameter.

We are given a set of N input points $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}$, with $\mathbf{x}_i \in \mathbb{R}^D$, and the set of corresponding outputs $\mathcal{Y} = \{\mathbf{y}_i\}_{i=1}^{N}$, with $\mathbf{y}_i \in \mathbb{R}^S$. Assuming the existence of a continuous mapping $f : \mathbb{X} \to \mathbb{Y}$ between the input and the output space, we want to estimate f from data with the multiresponse model

$$\mathbf{Y} = f(\mathbf{X}) + \mathbf{R}.$$

The columns of the matrices $\mathbf{X}$ and $\mathbf{Y}$ correspond to the D inputs and S outputs respectively, and the rows to the N observations. The columns of the $N \times S$ matrix $\mathbf{R}$ correspond to the residuals.

The MLM is a two-step learning method designed to

1. reconstructing the mapping existing between input and output distances;

2. estimating the response from the configuration of the output points.

In the following, the two steps are discussed.

For a selection of reference input points $R = \{\mathbf{r}_m\}_{m=1}^{M}$ with $\mathcal{R} \subseteq \mathcal{X}$ and corresponding outputs $\mathcal{T} = \{\mathbf{t}_m\}_{m=1}^{M}$ with $\mathcal{T} \subseteq \mathcal{Y}$, define $\mathbf{D}_x \in \mathbb{R}^{N \times M}$ in such a way that its m-th column contains the distances $d(\mathbf{x}_i, \mathbf{r}_m)$ between the N input points $\mathbf{x}_i$ and the m-th reference point $\mathbf{r}_m$. Analogously, define $\mathbf{\Delta}_y \in \mathbb{R}^{N \times M}$ in such a way that its m-th column contains the distances $\delta(\mathbf{y}_i, \mathbf{t}_m)$ between the N output points $\mathbf{y}_i$ and the output $\mathbf{t}_m$ of the m-th reference point. Assuming that exists a mapping κ between the input distance matrix $\mathbf{D}_x$ and the corresponding output distance matrix $\mathbf{\Delta}_y$, it can be reconstructed using the following multiresponse regression model

$$\mathbf{\Delta}_y = \kappa(\mathbf{D}_x) + \mathbf{E}.$$

The columns of the matrix $\mathbf{D}_x = [d(\mathbf{x}, \mathbf{r}_1), \ldots, d(\mathbf{x}, \mathbf{r}_M)]$ correspond to the M input vectors; the columns of the matrix $\mathbf{\Delta}_y = [\delta(\mathbf{y}, \mathbf{t}_1), \ldots, \delta(\mathbf{y}, \mathbf{t}_M)]$ correspond to the M response vectors, and the N rows correspond to the observations. The columns of the $N \times M$ matrix $\mathbf{E}$ correspond to the M residuals.

Assuming further that the mapping κ between input and output distance matrices has a linear structure for each response, the regression model has the form

$$\mathbf{\Delta}_y = \mathbf{D}_x\mathbf{B} + \mathbf{E}. \tag{3.1}$$

The columns of the $M \times M$ regression matrix $\mathbf{B}$ correspond to the coefficients for the M responses. The matrix $\mathbf{B}$ can be estimated from data through a minimization of the multivariate residual sum of squares as loss function, i.e.,

$$\begin{aligned} \mathrm{RSS}(\mathbf{B}) &= \sum_{m=1}^{M}\sum_{i=1}^{N}\Big(\delta(\mathbf{y}_i, \mathbf{t}_m) - \kappa_m(d(\mathbf{x}_i, \mathbf{r}_m))\Big)^2, && \text{(3.2a)} \\ &= \mathrm{tr}\Big((\mathbf{\Delta}_y - \mathbf{D}_x\mathbf{B})^T(\mathbf{\Delta}_y - \mathbf{D}_x\mathbf{B})\Big). && \text{(3.2b)} \end{aligned}$$

Under the conditions where the number of equations in Eq. (3.1) is larger than the number of unknowns, the problem is overdetermined and, usually, with no solution. This corresponds to the case where the number of selected reference points is smaller than the number of available points (i.e. $M < N$). In this case, we have to rely on the approximate solution provided by the ordinary least squares estimate of $\mathbf{B}$,

$$\hat{\mathbf{B}} = (\mathbf{D}_x^T\mathbf{D}_x)^{-1}\mathbf{D}_x'\mathbf{\Delta}_y. \tag{3.3}$$

If in Eq. (3.1) the number of equations equals the number of unknowns (i.e. $M = N$ because all the learning points are also reference points), then the problem is uniquely determined and has a single solution if the matrix $\mathbf{D}_x$ is full-rank. In this case,

$$\hat{\mathbf{B}} = (\mathbf{D}_x)^{-1}\mathbf{\Delta}_y. \tag{3.4}$$

Clearly less interesting is the case where in Eq. (3.1) the number of equations is smaller than the number of unknowns (i.e. for $M > N$, corresponding to the situation where, after selecting the reference points, only a smaller number of learning points is used). This case leads to an underdetermined problem with, usually, infinitely many solutions.

It is worth mentioning that, since the MLM assumes a linear mapping between the distance matrices, other learning (or estimation) technique can be chosen instead of using the ordinary least squares method, such as the least mean squares (LMS) (WIDROW, 2005) or even the recursive least squares (RLS) (HAYKIN, 2001) algorithms.

Given the possibility for $\mathbf{B}$ to be either uniquely solvable (Eq. (3.4)) or estimated (Eq. (3.3)), for an input test point $\mathbf{x} \in \mathbb{R}^D$ whose distances to the M reference input points $\{\mathbf{r}_m\}_{m=1}^M$ are collected in the vector $\mathbf{d}(\mathbf{x}, \mathcal{R}) = [d(\mathbf{x}, \mathbf{r}_1) \dots d(\mathbf{x}, \mathbf{r}_M)]$, the corresponding distances between the unknown output $\mathbf{y}$ and the known outputs $\{\mathbf{t}_m\}_{m=1}^M$ of the reference points are estimated as

$$\hat{\boldsymbol{\delta}}(\mathbf{y}, \mathcal{T}) = \mathbf{d}(\mathbf{x}, \mathcal{R})\hat{\mathbf{B}}. \tag{3.5}$$

The vector $\hat{\boldsymbol{\delta}}(\mathbf{y}, \mathcal{T}) = [\hat{\delta}(\mathbf{y}, \mathbf{t}_1) \dots \hat{\delta}(\mathbf{y}, \mathbf{t}_M)]$ provides an estimate of the geometrical configuration of $\mathbf{y}$ and the reference set $\mathcal{T}$, in the $\mathbb{Y}$-space.

The problem of estimating the output $\mathbf{y}$, given the outputs $\{\mathbf{t}_m\}_{m=1}^M$ of all the reference points and estimates $\hat{\boldsymbol{\delta}}(\mathbf{y}, \mathcal{T})$ of their mutual distances, can be understood as a *multilateration* problem (NIEWIADOMSKA-SZYNKIEWICZ; MARKS, 2009) to estimate its location in $\mathbb{Y}$. The problem of locating $\mathbf{y} \in \mathbb{R}^S$ is equivalent to solving the overdetermined set of M nonlinear equations corresponding to $(M+1)$-dimensional hyperspheres centered in $\mathbf{t}_m$ and passing through $\mathbf{y}$. Figure 7 graphically depicts the problem for $S = 2$.

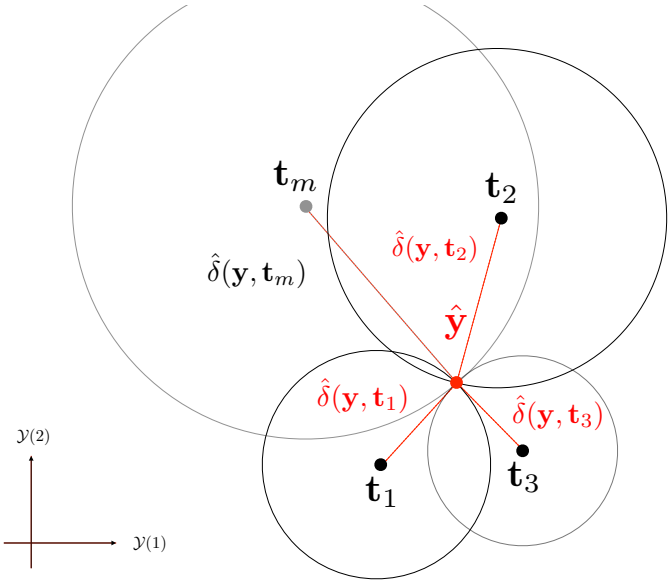


Figure 7: Output estimation.

Given the set of $m = 1, \ldots, M$ spheres each with radius equal to $\hat{\delta}(\mathbf{y}, \mathbf{t}_m)$

$$(\mathbf{y} - \mathbf{t}_m)^T(\mathbf{y} - \mathbf{t}_m) = \hat{\delta}^2(\mathbf{y}, \mathbf{t}_m), \tag{3.6}$$

the location of $\mathbf{y}$ can be estimated from the minimization of the objective function

$$J(\mathbf{y}) = \sum_{m=1}^{M} \Big((\mathbf{y} - \mathbf{t}_m)^T(\mathbf{y} - \mathbf{t}_m) - \hat{\delta}^2(\mathbf{y}, \mathbf{t}_m) \Big)^2. \tag{3.7}$$

The objective function has a minimum equal to 0 that can be achieved if and only if $\mathbf{y}$ is the solution of Eq. (3.6). If it exists, such a solution is thus global and unique. Due to the uncertainty introduced by the estimates $\hat{\delta}(\mathbf{y}, \mathbf{t}_m)$, an optimal solution to Eq. (3.7) can be achieved by any minimizer $\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmin}}\, J(\mathbf{y})$ like the nonlinear least square estimates from standard gradient descent methods. In the following, the Levenberg-Marquardt (LM) method (MARQUARDT, 1963) is preferred.

3.2 Parameters and computational complexity

From the exposed, the number of reference points M is virtually the only hyperparameter that the user needs to select in order to fine-tune an MLM model. As always, a selection based on conventional validation or on standard resampling methods for cross-validation could be adopted for the task (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

Two figures of merit can be considered for selecting M, the Average Mean Squared Error for the output distances ($AMSE(\boldsymbol{\delta})$) and the Average Mean Squared Error for the responses ($AMSE(\mathbf{y})$), which are given by

$$AMSE(\boldsymbol{\delta}) = \frac{1}{M} \sum_{m=1}^{M} \frac{1}{N_v} \sum_{i=1}^{N_v} (\delta(\mathbf{y}_i, \mathbf{t}_m) - \hat{\delta}(\mathbf{y}_i, \mathbf{t}_m))^2, \tag{3.8}$$

and

$$AMSE(\mathbf{y}) = \frac{1}{S} \sum_{s=1}^{S} \frac{1}{N_v} \sum_{i=1}^{N_v} (y_i^{(s)} - \hat{y}_i^{(s)})^2. \tag{3.9}$$

For a set of N_v validation pairs $(\mathbf{x}_i, \mathbf{y}_i)$, the $AMSE(\boldsymbol{\delta})$ quantifies how well the distances $\delta(\mathbf{y}_i, \mathbf{t}_k)$ between the N_v output responses $\mathbf{y}_i$ and the outputs of the M selected reference points $\mathbf{t}_m$ are estimated $\hat{\delta}(\mathbf{y}_i, \mathbf{t}_m)$. The $AMSE(\mathbf{y})$, in turn, is only performed after both the distance regression and the estimation steps of the MLM are completed, thus, it quantifies how well the S-dimensional outputs $y_i^{(s)}$ are estimated $\hat{y}_i^{(s)}$. In the case of univariate responses ($S = 1$) the $AMSE(\mathbf{y})$ reduces to the conventional Mean Square Error for the outputs ($MSE(y)$).

Ideally, if we had enough data, we would set aside a validation set and use it to assess the performance of our model trained with a varying number M of reference points.

The value of M that optimizes the chosen figure of merit is then used to learn the final model with all the data. However, since the data are often scarce, this is usually impossible and we need to resort to cross-validation. We split the data into a number N_f of roughly equal-sized parts and for the i-th part we train the model with a varying number M of reference points on data from the other $N_f - 1$ parts and we calculate the corresponding figure of merit. This is repeated for $i = 1, 2, \ldots, N_f$ and the resulting figure of merit for the same value of M averaged over all the N_f folds. Again, the value of M that minimizes the chosen figure of merit is then used to learn the final model with all the data. The case $N_f = N$ is known as leave-one-out cross-validation.

The training procedure of the Minimal Learning Machine is sketched in Algorithm 1. The training computation can be roughly divided into two parts: *i*) calculation of the pairwise distance matrices in the output and input space; and *ii*) calculation of the least-square solution for the multiresponse linear regression problem on distance matrices. The first part takes $\Theta(MN)$ time (see Cormen et al. (2009) for a review of algorithmic asymptotic analysis). The computational cost of the second part is driven by the calculation of the Moore-Penrose pseudoinverse matrix.

Algorithm 1 MLM training procedure

Input: Training data sets $\mathcal{X}$ and $\mathcal{Y}$, and M.
Output: $\hat{\mathbf{B}}$, $\mathcal{R}$ and $\mathcal{T}$.

1. Randomly select M reference points, $\mathcal{R}$, from $\mathcal{X}$ and their corresponding outputs, $\mathcal{T}$, from $\mathcal{Y}$;
2. Compute $\mathbf{D}_x$: The distance matrix between $\mathcal{X}$ and $\mathcal{R}$;
3. Compute $\mathbf{\Delta}_y$: The distance matrix between $\mathcal{Y}$ and $\mathcal{T}$;
4. Calculate $\hat{\mathbf{B}} = (\mathbf{D}_x^T \mathbf{D}_x)^{-1} \mathbf{D}_x^T \mathbf{\Delta}_y$.

One of the most used methods for the calculation of Moore-Penrose pseudoinverses is the SVD (GOLUB; LOAN, 1996), which runs in $\Theta(M^2 N)$ time. This method is very accurate but its drawback lies in the computational time constants that makes it time-intensive. In order to speed up the computation, several methods have been proposed (for example, see Katsikis, Pappas and Petralias (2011), and Courrieu (2005)). In Katsikis, Pappas and Petralias (2011), the computation is optimized by using a special type of tensor product and QR factorization, whereas the method proposed by Courrieu (2005) is based on a full-rank Cholesky decomposition. Even if such approaches significantly improve the computational time of computing the Moore-Penrose inverse matrix, the time complexity is still equal to that provided by the SVD method. Even though, one might consider them for large datasets or real-time applications.

The time complexity of the MLM training phase is driven by the computation of the Moore-Penrose matrix and then it is given by $\Theta(M^2 N)$. In order to establish a comparison, the MLM training computational cost is similar to what presented by an ELM network when the number of hidden neurons is equal to the number of reference points.

It is worth noting that the ELM is considered one of the fastest methods for nonlinear regression and classification tasks (MICHE et al., 2010).

Algorithm 2 MLM test procedure

Input: $\hat{\mathbf{B}}$, $\mathcal{R}$, $\mathcal{T}$ and $\mathbf{x}$.
Output: $\hat{\mathbf{y}}$.
1. Compute $\mathbf{d}(\mathbf{x}, \mathcal{R})$;
2. Compute $\hat{\boldsymbol{\delta}}(\mathbf{y}, \mathcal{T}) = \mathbf{d}(\mathbf{x}, \mathcal{R})\hat{\mathbf{B}}$;
3. Use $\mathcal{T}$ and $\hat{\boldsymbol{\delta}}(\mathbf{y}, \mathcal{T})$ to find an estimate for $\mathbf{y}$. This can be accomplished by any gradient descent algorithm over the cost function in Eq. (3.7).

Concerning the computational analysis of the generalization step in MLM (Algorithm 2), we consider the Levenberg-Marquardt method due to its fast and stable convergence, even though any gradient descent method can be used on the minimization step in Eq. (3.7). For each iteration, the LM method involves the computation of the Jacobian matrix $\mathbf{J} \in \mathbb{R}^{M \times S}$ and the inverse of $\mathbf{J}^T\mathbf{J}$. In this regard, the computational complexity of the LM algorithm is about $\Theta(I(MS^2 + S^3))$, where S is the dimensionality of $\mathbf{y}$ and I denotes the number of iterations. In most of the regression and classification problems, S is a small number and then the complexity turns to be proportional to the number of reference points and number of iterations. This is slightly worse than what is presented, for instance, by SVM methods with M support vectors, which is linear in M with small constant factor. Also, ELM testing phase runs in $\Theta(MS)$ time, with M corresponding to the number of hidden units.

3.3 Links with Multiquadric Radial Basis Functions and Kernel Dependency Estimation

An alternative interpretation for the MLM lies in the theory of RBFs. A radial basis function approximation of a function $f : \mathbb{R}^D \to \mathbb{R}$ takes the form

$$h(\mathbf{x}) = \sum_{l=1}^{L} \beta_l \phi(\|\mathbf{x} - \mathbf{m}_l\|), \tag{3.10}$$

where $\phi : [0, \infty) \to \mathbb{R}$ is a fixed univariate function, $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_L]$ denotes a vector of coefficients usually estimated by the ordinary least squares method and $\mathbf{m}_l$ correspond to centers of the radial basis functions $\phi(.)$. In other words, the RBF approximation $h(\mathbf{x})$ is a linear combination of translates of a fixed function $\phi(.)$ of a single real variable.

Similarly to the RBF approximation, the MLM equations for the distance regression step (Eq. (3.5)) are given by

$$\hat{\delta}(\mathbf{y}, \mathbf{t}_m) = \sum_{j=1}^{M} b_{jm} d(\mathbf{x}, \mathbf{r}_j), \qquad m = 1, \ldots, M. \tag{3.11}$$

Among a variety of radial basis functions, a well-known function is the multiquadric function (HARDY, 1971), where $\phi(\|\mathbf{x} - \mathbf{m}_l\|) = (\|\mathbf{x} - \mathbf{m}_l\|^2 + \sigma^2)^{1/2}$, with $\sigma \in \mathbb{R}$. By simply setting the constant $\sigma = 0$, the multiquadric function becomes the identity $\phi(||\mathbf{x} - \mathbf{m}_l||) = \|\mathbf{x} - \mathbf{m}_l\| = d(\mathbf{x}, \mathbf{r}_m)$, with the centers $\mathbf{m}_l$ corresponding to the reference points $\mathbf{r}_m$, and $L = M$.

Analogously to the RBF network, the MLM approximates a function by a linear combination of translates of the multiquadric function, with the difference that the target function is a distance function in the output space. Moreover, the MLM uses a randomly chosen subset of the learning points as centers, which is also a feasible choice for the design of RBF learning models (BISHOP, 2006). The number of reference points, or centers, corresponds to a hyperparameter to be optimized based on a specific dataset.

Since estimating distances is not the final goal in learning the target function f, the MLM requires an optimization step to estimate outputs based on the estimated distances. It is a deterministic problem given that the distances are perfectly reconstructed. Indeed, because the distances are usually not perfectly estimated, the output estimation as an optimization procedure allows the use of many different techniques, including methods for robust estimation (HUBER, 2004).

Weston et al. (2002) introduced the kernel dependency estimation (KDE) approach, which is a kernel based method for learning a dependency between two classes of objects — one class being the input and the other class the corresponding output. The objects can be defined in terms of vectors, images, strings, trees or graphs. In a like manner to MLM, KDE learning employes similarity measures in both input and output spaces, and it also requires an output estimation, which is in this case called pre-image problem. There are, however, four notable differences between the MLM and KDE. Firstly, the KDE is a kernel method and, as such, it relies on kernel functions in order to embed the objects into vector spaces, whereas any dissimilarity or proximity measure can be used in MLM. Secondly, the KDE requires a PCA step to be applied to the kernel output similarity matrix. In MLM, one uses the distance matrix directly in order to learn the input-output mapping. Thirdly, the pre-image problem (i.e. output estimation) in KDE is solved using a very simplified algorithm, where the closest output sample is chosen from the available training set. In MLM, an optimization procedure is carried out as described in Algorithm 2. Finally, the KDE was evaluated mainly on classification problems, while the MLM is evaluated comprehensively in both, regression and classification tasks.

3.4 Application to system identification

The extension of the Minimal Learning Machine to system identification is straightforward, since we have adopted the *external dynamics* approach in this thesis. The

term "external dynamics" stems from the fact that the nonlinear dynamic model can be separated into two parts: a nonlinear static approximator and an external dynamic filter bank (NELLES, 2001). The MLM assumes the role of the nonlinear static mapping. Thus, all that is required is to design the regression vectors accordingly to the system dynamics (memory orders n_u and n_y), as illustrated in Figure 8. In the context of SISO systems, the input and output learning sets are given by $\mathcal{X} = \{\mathbf{x}(n)\}_{n=1}^{N}$, with $\mathbf{x}(n) = [y(n-1), \ldots, y(n-n_y); u(n-1), \ldots, u(n-n_u)] \in \mathbb{R}^{n_u+n_y}$, and a set of corresponding outputs $\mathcal{Y} = \{y(n)\}_{n=1}^{N}$.

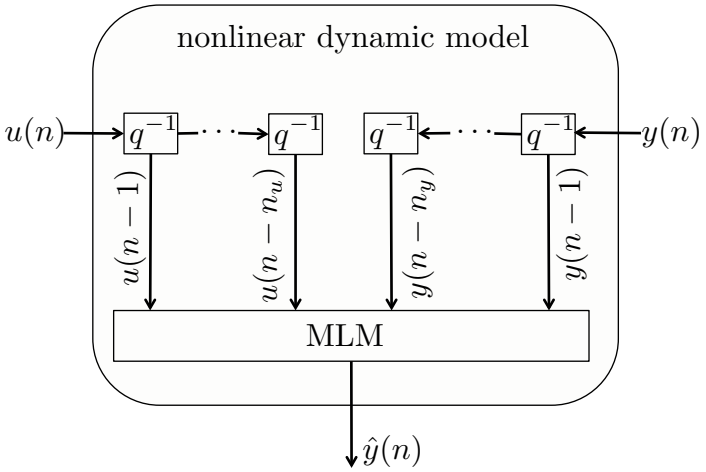


Figure 8: Use of the MLM for nonlinear system identification: external dynamics approach.

A remarkable characteristic of MLM is its inherently capability of operating on multidimensional responses. Therefore, the MLM can be used in the identification of MIMO (Multiple-Inputs Multiple-Outputs) systems without any additional step or adaptation.

3.5 On the selection of reference points

A crucial aspect related to MLM training is the selection of reference points. In this section, we discuss two simple strategies for selecting reference points in MLM.

Based on the RBF theory for center placement, the first natural choice is to carry out centroid-based selection. In doing so, the *k-medoids* algorithm (KAUFMAN; ROUSSEEUW, 1990) is a feasible choice. A description of the k-medoids method can be found in Appendix A. The rationale of k-medoids is to select the most representative k points from a set, in the sense of minimizing an intra-cluster error measure (dissimilarity), while maximizing an inter-cluster error measure. Figure 9 shows the selection of 60 reference points (input space) using the k-medoids clustering method on the synthetic example presented in chapter 1.

A different strategy would be to carry out center placement from the most complex (high curvature) portion of the target function in a supervised manner. Among such

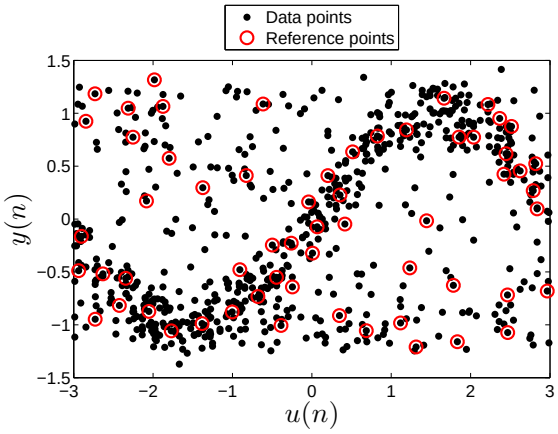


Figure 9: Selection of reference points through k-medoids.

strategies, a well-known method was proposed by Chen, Cowan and Grant (1991) and it is based on the *orthogonal least-squares* method. The advantage in performing supervised center/reference placement is that we can select centers in those regimes where they are most effective in terms of modeling error reduction (NELLES, 2001).

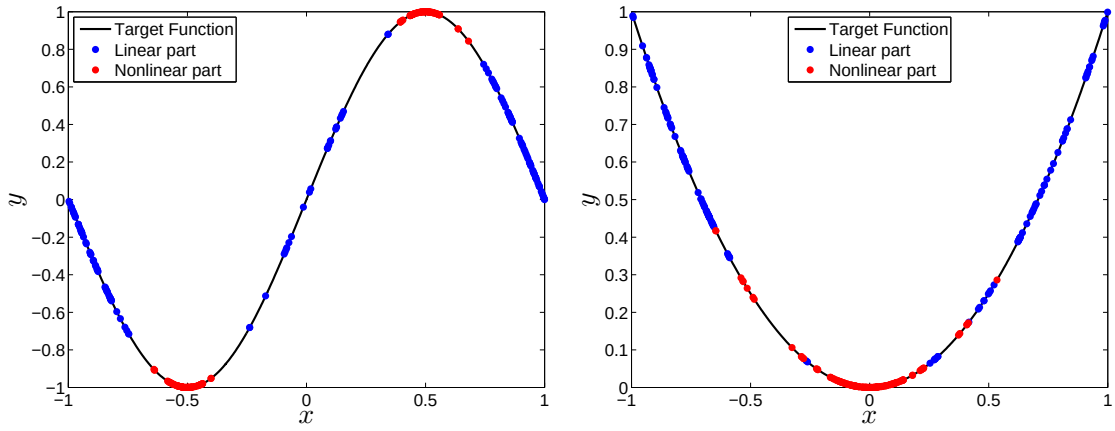
We propose a method based on distance computations and nonparametric hypothesis testing. The idea is to locally match distance measurements on the input and output spaces. Thus, for all training pairs $\{\mathbf{x}(n), y(n)\}$, we compute distances to the k nearest points in the training sets $\mathcal{X}$ and $\mathcal{Y}$, respectively. Let us assume that column vector $\mathbf{d}_k(n)$ encompasses the distances from $\mathbf{x}(n)$ to its k neighbors (nearest points) in $\mathcal{X}$. Likewise, $\boldsymbol{\delta}_k(n)$ denotes the distances between $y(n)$ and its k neighbors in $\mathcal{Y}$. Now, think of $\mathbf{d}_k(n)$ and $\boldsymbol{\delta}_k(n)$ as local distance distributions on the input and output spaces. Assuming that the target function $f : \mathbb{X} \to \mathbb{Y}$ satisfies the Weierstrass continuity definition[3], then the distributions $\mathbf{d}_k(n)$ and $\boldsymbol{\delta}_k(n)$ are equal (matches) up to a normalization factor if we consider a neighborhood of $f(\cdot)$ such that f is linear.

To ensure the matching between the input and output local distance measurements, we use the resulting p-value of the nonparametric Kolmogorov-Smirnov (KS) hypothesis test (MASSEY, 1951). The null hypothesis is that $\mathbf{d}_k(n)$ and $\boldsymbol{\delta}_k(n)$ are samples coming from the same distribution. Therefore, the idea is to apply the test and use the p-values from the two-sample KS test as a ranking criterion, indicating which points correspond to the most linear part of the target function. Clearly, the measurements $\mathbf{d}_k(n)$ and $\boldsymbol{\delta}_k(n)$ are normalized to have zero-mean and unit variance before applying the KS test.

Additionally, heuristics are necessary to select the references on the basis of the resulting ranking. For instance, a simple strategy would be to randomly select 20% of the reference points from those where the null hypothesis is accepted (high p-value) and 80% from those 40% with smallest p-values. Figure 10 illustrates the procedure for two toy

[3] $f(x)$ is continuous at $x = x_0$ if $\forall\ \varepsilon > 0\ \exists\ \delta > 0$ such that for every x in the domain of f, $|x - x_0| < \delta \Rightarrow |f(x) - f(x_0)| < \varepsilon$.

functions. For the experiment, we used neighborhood size (k) equal to 100. The red points are the 200 points with the smallest p-values and the blue ones are the 200 points with highest p-values.



(a) $y = \sin(x)$ (b) $y = x^2$

Figure 10: Selection of reference points through KS test for two different functions.

3.6 Illustrative examples

In this section, we illustrate the Minimal Learning Machine using two synthetic problems, one for regression and one for system identification. The regression problem consists in the estimation of a smoothed and nonlinear version of the parity function. The system identification problem consists in the synthetic plant described in Chapter 1.

3.6.1 The smoothed parity

To illustrate the behavior of the Minimal Learning Machine for regression problems, we generated 2^{13} bi-dimensional input points uniformly distributed in the unit-square, $\mathbf{x} \in [0, 1]^2$, and built the response using the smoothed parity function model $y = f(\mathbf{x}) + \varepsilon$, where $f = \sin(2\pi x_1)\sin(2\pi x_2)$ and $\varepsilon \sim \mathcal{N}(0, 0.1^2)$, Figure 11.

We analyzed the performance of the MLM for N learning points ranging from 2^1 to 2^{12} and M randomly selected reference points such that always $M \leq N$. An independent set of $N_v = 2^{12}$ points is used for validating the MLM in terms of its hyperparameter M. Two figures of merit are used, the $AMSE(\boldsymbol{\delta})$ for the output distances and the $MSE(y)$ for the single response. In our experiments, the Minimal Learning Machines are trained with all the different N-sized learning sets and for all possible number of reference points. For each size N of the learning set and for a varying number M of reference points, the figures of merit of the distance regression and the output estimation steps are then evaluated on the validation set (Figure 12).

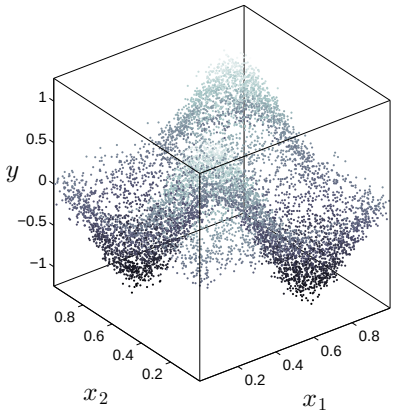
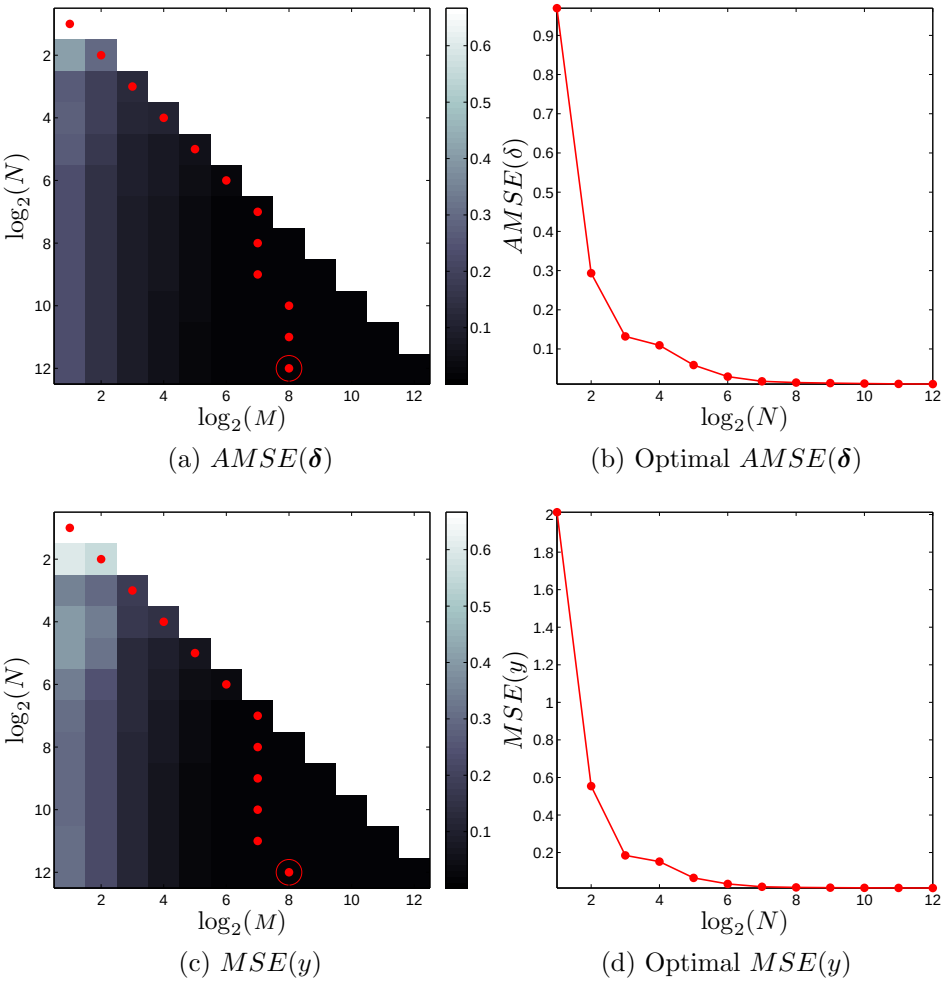


Figure 11: The *smoothed* parity function: Data



(a) $AMSE(\boldsymbol{\delta})$ (b) Optimal $AMSE(\boldsymbol{\delta})$

(c) $MSE(y)$ (d) Optimal $MSE(y)$

Figure 12: The *smoothed* parity function: Figures of merit.

Figure 12a shows the $AMSE(\boldsymbol{\delta})$ performance of the Minimal Learning Machine in the distance estimation step, for different combinations of N learning points and M reference points. For a given number N of learning points, the number of reference points that leads to the best performances is denoted by a red dot and, then, a red circle is used to denote the MLM with the overall smallest $AMSE(\boldsymbol{\delta})$. Related to that, Figure 12b

illustrates the $AMSE(\boldsymbol{\delta})$ achieved by the best performing MLMs for different sizes of the learning set. Analogously for the output estimation step, Figure 12c shows the $MSE(\mathbf{y})$ performances, where again the best performing MLMs are denoted by red dots, and a red circle denotes the best MLM overall. By the same token, Figure 12d shows the performance of the best MLMs for different sizes of the learning set. Based on the results depicted in Figure 12b and 12d, it is possible to observe that the MLM performance improves as the number N of learning points increases; an expected result. On the other hand, the optimal number M of reference points does not necessarily grow at the same rate of the number of learning points. After some point, including more reference points decreases the generalization performances of the MLM, both in terms of distance regression and output estimation, a clear indication that the MLM starts overfitting. Considering the nature of the sets of equations that characterize the two steps of the MLM learning, this is again an expected result further confirmed by experimental evidence.

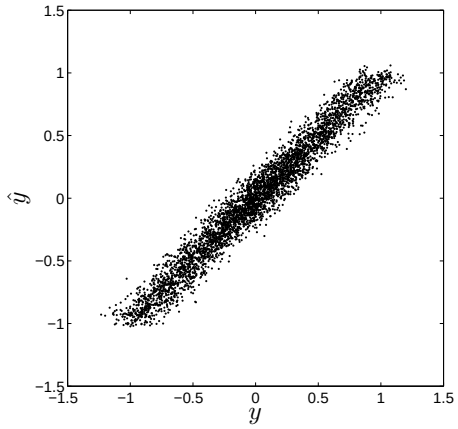


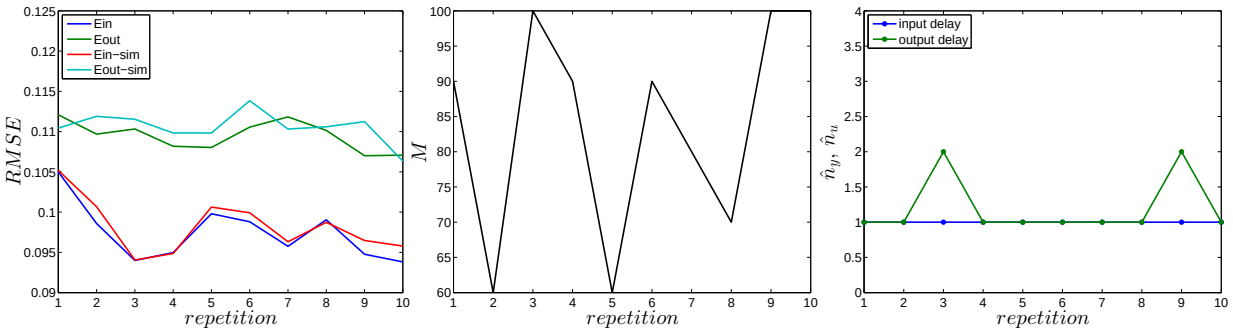
Figure 13: The *smoothed* parity function: Output estimation with $N = 2^{12}$ and $M = 2^8$.

The best Minimal Learning Machine overall was found to be the one trained using $N = 2^{12}$ learning points and $M = 2^8$ reference points. It is worth noticing that for both the distance regression and the output estimation step, the optimal number of reference points is found to be the same. Figure 13 illustrates the validation results when estimating the response with the best performing MLM. Interestingly, the $MSE(y)$ achieved by this MLM is equal 0.011, which tends to the variance of the additive noise in the response ($\text{Var}(\varepsilon) = 0.010$) and thus also to the smallest Mean Square Error that any regression model can achieve without overfitting.

3.6.2 Synthetic example

In order to illustrate the performance of the MLM on nonlinear modeling, we have run experiments with the synthetic example. Since we assume the memory order is unknown, we combine grid-search and 10-fold cross-validation to select the optimal memory orders, with $n_y, n_u = \{1, 2, 3\}$, and the number of reference points M varying from

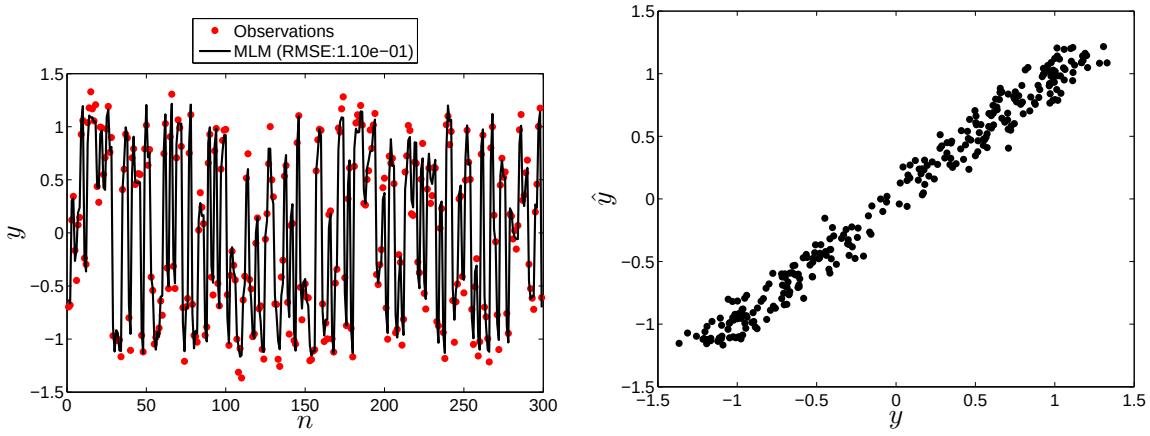
10 to 100 with step size of 10. Figure 14 illustrates the results for ten independent runs. Figure 14a illustrates the in-sample and out-of-sample RMSE values for the one-step-ahead predictions (E_{in} and E_{out}) and free-run simulations (E_{in-sim} and $E_{out-sim}$). With respect to the RMSE performance, we observe that the simulation and prediction results are quite similar (Figure 14a). Still on the RMSE performance we may verify that the out-of-sample performance is more stable (smaller variation) than the in-sample performance. By contrasting Figures 14a and 14b, the in-sample error in general decreases as more reference points are selected, as expected. The optimal number of reference points varied from 60 to 100 (Figure 14b). The memory order terms are depicted in Figure 14c, where a wrong order detection led to a selection of 100 reference points (repetitions 3 and 9).



(a) MLM accuracies (b) Number of reference points (c) Delays

Figure 14: Synthetic example: MLM results.

Figure 15 illustrates the MLM performance in terms of its simulation capability on the validation/testing set. From Figure 15b the MLM is able to recover the system dynamics accurately. Also, the resulting RMSE value approximates the standard deviation of the noise signal.



(a) Simulation on validation set (b) Estimates: simulation

Figure 15: Synthetic example: MLM estimates.

The approximation of the MLM for the static behavior of the system is illustrated in Figure 16. The MLM is able to accurately approximate the static curve of the system.

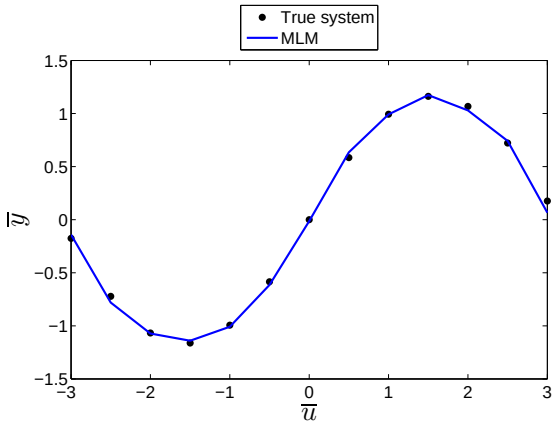


Figure 16: Synthetic example: MLM static behavior.

3.7 Concluding remarks

This chapter introduced a new learning method called the Minimal Learning Machine, or MLM for short. Learning in MLM consists in reconstructing the mapping between input and output distance matrices and then exploiting the geometrical arrangement of the output points for estimating the response. Based on our experiments, a multiresponse linear regression model is capable of reconstructing the mapping existing between the aforementioned distance matrices. Also, the MLM is inherently capable of operating on multidimensional responses. We have reported the MLM performances on classification and regression problems in Appendices B and C, respectively. On a large number of real-world problems, the Minimal Learning Machine has achieved accuracies that are comparable to what is obtained using state-of-the-art classification and regression methods. In addition, we illustrated through the synthetic example given in Chapter 1 the application of the MLM on system identification tasks.

The computational complexity of the MLM training procedure is low, competing with the fastest machine learning approaches, such as the ELM network. Regarding the test procedure, we reported that combining random selection of reference points and the Levenberg-Marquardt optimization method lead to good performance.

With respect to related works, the relationship between the MLM and RBF network was discussed, and we have shown that the MLM training can be interpreted as the use of a multiquadric RBF network in the estimation of distances in the output space. Additionally, we described the similarities and differences between the MLM and KDE method.

Regarding the selection of reference points, we discussed two alternatives to the random selection described in the MLM basic formulation. The number of strategies for that is not limited to what is presented in this chapter. In fact, most of the methodologies already proposed in the literature can be applied to the MLM. For example, ranking criterion based on subset selection methods, as proposed by Miche et al. (2010) and Chen,

Cowan and Grant (1991), can be equally applied. A comprehensive evaluation of the impact of the different methods for reference point placement is out of the scope of the thesis, and will not be further discussed.

Finally, a significant advantage of the MLM over other supervised learning methods is that the MLM has only one hyperparameter to be optimized using standard resampling methods, like leave-one-out (LOO) cross validation. Because the MLM is linear in the parameters on the distance regression step, once the reference points are determined, the LOO error can be analytically found using the PRESS (or PREdiction Sum of Squares) statistics (BONTEMPI; BIRATTARI; BERSINI, 1998).

Chapter 4

Regional Modeling

As introduced in Chapter 2, modeling techniques can be categorized into two general paradigms: global and local modeling. Global modeling consists in fitting a single regression model to the available data, using the whole set of input and output observations. On the other side of the spectrum stands the local modeling approach, in which the input space is segmented into several small partitions and a specialized regression model is fit to each partition. The rationale of the local modeling approach relies on the fact that complex (e.g. nonlinear) dynamics of the input-output mapping can be represented by multiple simpler mappings. It usually brings the advantage of interpretability.

However, the alleged flexibility of using multiple local models comes with some costs. One of the main problems is that it is not straightforward to select the appropriate number of local models beforehand, without any prior information. An inappropriate selection may cause the over- or under-identification of the original system dynamics (WANG; SYRMOS, 2007). Regarding SOM-based models, another shortcoming is related to the occurrence of dead (or interpolating) neurons[1] after SOM training. In this case, it is impossible to associate a local model with this type of neuron, since there are no data points to estimate the parameters of the corresponding local model.

To handle these shortcomings, we propose a novel approach to system identification, called *Regional Modeling* (RM), that stands in between the global and local modeling approaches. RM is originally motivated by the two-level clustering approach introduced by Vesanto and Alhoniemi (2000) and can be thought of as an extension of their work to regression problems, particularly to nonlinear dynamic system identification. The RM principle is to combine a smaller number of models in order to cover a larger area of the input space than purely local models.

In this chapter, we propose regional models based on clustering of the SOM. For this

[1] Neurons whose prototypes are located at positions without data points.

purpose, we first partition the input space using a single SOM network with Q prototypes, and then perform clustering using the k-means algorithm (MACQUEEN, 1967) over the prototypes of the trained SOM in order to find an optimal number K_{opt} ($K_{opt} \ll Q$) of clusters of SOM prototypes. The optimal number of clusters can be found by using any cluster validation technique, such as the Davies-Bouldin index (DAVIES; BOULDIN, 1979; HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2001). A given cluster of SOM prototypes defines a region in the input space formed by merging the Voronoi cells of the prototypes belonging to that cluster. Finally, for each individual cluster of SOM prototypes, we build a regional regression model using only the data vectors mapped to that specific cluster.

It is worth mentioning that, by using K_{opt} regional models instead of Q local models, the RM approach is much more parsimonious than the local modeling approach, in the sense that few models are required to describe the dynamics of the system of interest. A second advantage is that the user does not need to worry a great deal about the specification of the number Q of SOM prototypes, since the subsequent application of k-means clustering to the SOM prototypes makes the number of regional models relatively decoupled from large variations in the value of Q. In other words, large variations in Q do not produce large variations in K_{opt}, a property that confer considerable robustness to the RM approach. A third advantage of the RM approach over local modeling is that regional models can be constructed even if dead/interpolating neurons occur after SOM training. This is possible because any of the existing dead/interpolating neurons will belong eventually to one of the K_{opt} regions available. Finally, Vesanto and Alhoniemi (2000) concluded that a two-step clustering approach performs well when compared with direct clustering of the data while reduce its computational time.

The remainder of the chapter is organized as follows. In Section 4.1, a regional modeling framework based on the SOM is described and its application to a synthetic plant is illustrated (Section 4.1.1). In Section 4.2 we present the fundamentals of M-estimation and its use in the context of regional modeling. In Section 4.3, we discuss related works. Closing remarks are given in Section 4.4.

4.1 Regional modeling by clustering of the SOM

Regional Modeling (RM) (SOUZA JUNIOR; BARRETO; CORONA, 2015) is an alternative approach to the more traditional local and global modeling techniques. A single global model may not be able to capture the dynamics of regions with high curvatures, which might result in smoothing of the details. On the one hand, local models are good for capturing such details of the system dynamics. On the other hand, it is relatively difficult to specify an adequate number of local models. A number that is too small may cause under-identification, with too few submodels to represent important regions of the

system dynamics. A large number causes over-identification, with submodels attached to unimportant regions of the system dynamics. The RM approach then comes out as an alternative to finding a tradeoff between the standard global and local approaches.

In order to build regional models through clustering of the SOM, we follow the procedure introduced by Vesanto and Alhoniemi (2000). Thus, the first step requires training the SOM as usual, with Q prototypes, using the available training input examples $\mathcal{X}$.

Once trained, for each SOM prototype $\mathbf{w}_i$, $i = 1, \ldots, Q$, we collect the input samples associated to the ith prototype as:

$$\mathcal{V}_i = \{\mathbf{x}(n) \in \mathcal{X} \mid \|\mathbf{x}(n) - \mathbf{w}_i\| \leq \|\mathbf{x}(n) - \mathbf{w}_j\|, \forall j = 1, \ldots, Q\}. \tag{4.1}$$

The subset $\mathcal{V}_i$ denotes the input training samples whose closest SOM prototype is $\mathbf{w}_i$. In traditional local modeling, the elements of $\mathcal{V}_i$ and their corresponding outputs are used for building linear models. In regional modeling, an additional step is needed, and it corresponds here to the clustering of the SOM prototypes.

Clustering approaches can be divided into hierarchical and partitional methods. Although one may use any clustering algorithm, in this thesis, we chose partitional methods since they do not depend on previously found clusters. More specifically, we use the k-means algorithm as the clustering method in combination with the Davies-Bouldin (DB) index in order to select the optimal number of clusters (see Appendix A for details on the DB index). By doing this, we follow previous experiments by Vesanto and Alhoniemi (2000).

As usual for clustering validity indices, the Davies-Bouldin criterion minimizes the within-cluster distance and maximizes the inter-cluster distance, additionally indicating low values for spherical clusters (VESANTO; ALHONIEMI, 2000). Since the k-means also tries to find spherical clusters, the DB index is a suitable choice for evaluating k-means partitioning. In such an approach, we compute $K = 1, 2, \ldots, K_{max}$ partitioning of the SOM prototypes and the corresponding DB index values as well. The optimal partitioning, represented by K_{opt} partitions, is then chosen by the following search procedure:

$$K_{opt} = \underset{K=1,\ldots,K_{max}}{\operatorname{argmin}} \; DB(\mathcal{W}, \mathcal{P}^K), \tag{4.2}$$

where $\mathcal{P}^K = \{\mathbf{p}_j\}_{j=1}^{K}$, $\mathbf{p}_j \in \mathbb{R}^{n_u+n_y}$, denotes the set of K prototypes of the k-means algorithm, while $\mathcal{W} = \{\mathbf{w}_i\}_{i=1}^{Q}$ is the set of SOM prototypes.

After K_{opt} is selected, the r-th cluster of SOM prototypes $\mathcal{A}_r$ is comprised of all SOM prototypes $\mathbf{w}_i$ that are mapped onto the prototype $\mathbf{p}_r$ of the k-means, which is

$$\mathcal{A}_r = \{\mathbf{w}_i \mid \|\mathbf{w}_i - \mathbf{p}_r\| \leq \|\mathbf{w}_i - \mathbf{p}_j\|, \forall j = 1, \ldots, K_{opt}\}. \tag{4.3}$$

We are interested in partitioning the input samples $\mathcal{X}$, since we may use them to fit regression models. For this, we collect the data samples associated with the SOM

prototypes of the r-th cluster in order to create the r-th data partitioning $\mathcal{X}_r$ given by

$$\mathcal{X}_r = \bigcup_{\mathbf{w}_i \in \mathcal{A}_r} \mathcal{V}_i, \qquad r = 1, \ldots, K_{opt}. \tag{4.4}$$

In words, each element of $\mathcal{X}$ is mapped into a region $\mathcal{X}_r$, comprised of those input vectors whose closest SOM prototype belongs to $\mathcal{A}_r$. Finally, once the original dataset has been divided into K_{opt} subsets (one per region), the last step consists in building K_{opt} regional regression models using $\mathcal{X}_r$, $r = 1, \ldots, K_{opt}$.

Assume that we arrange the N_r elements of $\mathcal{X}_r$ along the rows of a matrix $\mathbf{X}_r$. Thus, for regional linear models, one can simply estimate the coefficient vector $\boldsymbol{\beta}_r$ of the r-th regional model using the ordinary least-squares method as follows:

$$\hat{\boldsymbol{\beta}}_r = (\mathbf{X}_r^T \mathbf{X}_r)^{-1} \mathbf{X}_r \mathbf{y}_r, \qquad r = 1, ..., K_{opt}, \tag{4.5}$$

where $\mathbf{y}_r \in \mathbb{R}^{N_r}$ is a vector of N_r observed output values associated with the N_r row-vectors in $\mathbf{X}_r$. Henceforth, we denote by *Regional Linear Model* (RLM) a regional model built using K_{opt} linear models.

For an out-of-sample input vector $\mathbf{x} \in \mathbb{R}^{n_u+n_y}$, the test procedure consists in computing the predicted output value using the suitable regional model. For RLMs, the output value is computed as

$$\hat{y} = \hat{\boldsymbol{\beta}}_{r^*}^T \mathbf{x}, \tag{4.6}$$

where r^* is the index of the winning regional model.

Assuming we are using the k-means algorithm as the clustering method, the index of the winning regional model, r^*, can be selected as

$$r^* = \underset{r=1,\ldots,K_{opt}}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{p}_r\|, \tag{4.7}$$

where $\mathbf{p}_r$ is the r-th prototype of the k-means algorithm. An alternative way of determining the winning region and consequently the wining model is:

1. Find the closest SOM prototype to $\mathbf{x}$:

$$i^* = \underset{i=1,\ldots,Q}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{w}_i\|. \tag{4.8}$$

2. Find the closest k-means prototype to $\mathbf{w}_{i^*}$:

$$r^* = \underset{r=1,\ldots,K_{opt}}{\operatorname{argmin}} \|\mathbf{w}_{i^*} - \mathbf{p}_r.\| \tag{4.9}$$

Selecting the winning region using Eq. (4.7) or Eqs. (4.8) and (4.9) produces results that are only slightly different. Based on that, we used Eq. (4.7) in the experiments of this thesis since it is faster and eliminates the need to keep the SOM prototypes.

It is worth emphasizing that regional modeling does not define which learning/regression model one may use over the input space regions. We have shown here how to fit linear models, but regional modeling is not limited to them. If we choose to build nonlinear regional models, all that is required is to fit the chosen learning model using the input vectors in $\mathcal{X}_r$ and the corresponding target values in $\mathcal{Y}_r$, for $r = 1, \ldots, K_{opt}$. Fast models are preferred, and precautions may be taken in order to not produce over-parametrized nonlinear models, since the amount of data in each region can be small. Good candidate models are the ELM and the MLM models, where overfitting can be controlled by constraining the number of hidden units and reference points, respectively, or in the light of regularization theory. In addition, ELM and MLM have fast training and have not been comprehensively evaluated for system identification tasks, where, for example, oversampled systems may cause matrix ill-conditioning problems. We use the term *Regional Extreme Learning Machines* (RELM) and *Regional Minimal Learning Machines* (RMLM) to denote regional models comprised of ELM networks and MLMs respectively. The regional models are able to deal with *multiple-input multiple-output* (MIMO) systems since the regression models handle multiple outputs.

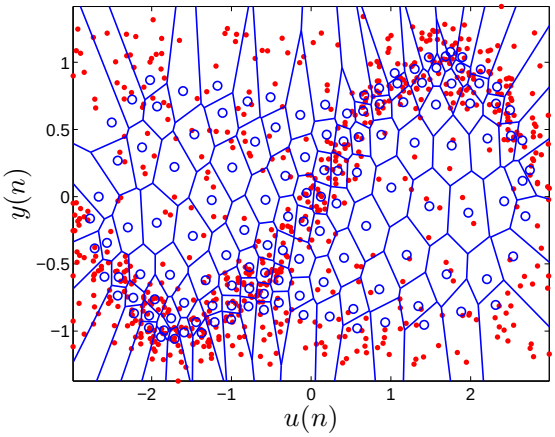
The rationale of using a nonlinear model over regions is because each region encompasses a larger portion of the input space than a purely local model. Thus, nonlinearities may be present. If this is the case, a slightly nonlinear model may be a better choice.

4.1.1 Illustrative example

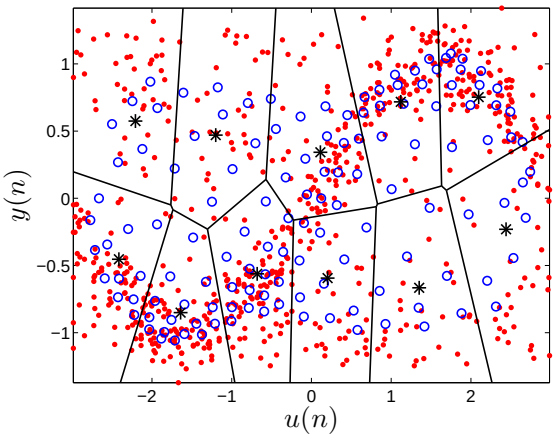
In this section we illustrate the performance of regional linear models on the synthetic plant used throughout the thesis.

Figure 17a shows the SOM prototypes and the corresponding Voronoi cells over the data in the input space. Data samples are depicted with red dots, and SOM prototypes are denoted by blue circles. The rule of thumb of $Q = 5\sqrt{N}$ (VESANTO et al., 2000) was used in the experiment, i.e., the number of SOM prototypes equals five times the square root of the number of training points. In this example, $N = 699$ and $Q = 133$. Figure 17a corresponds to the first step for building regional models. After that, we must apply vector quantization of the SOM prototypes using the k-means algorithm. This is illustrated in Figure 17b, where the optimal number of clusters was found by the DB index ($K_{opt} = 11$). In addition, Figure 17c shows the regions in the input space, i.e., the data points used to build 11 independent linear regression models. As one may observe, in the conventional local approach, Q models would be built, while the RM method delivers 11 models.

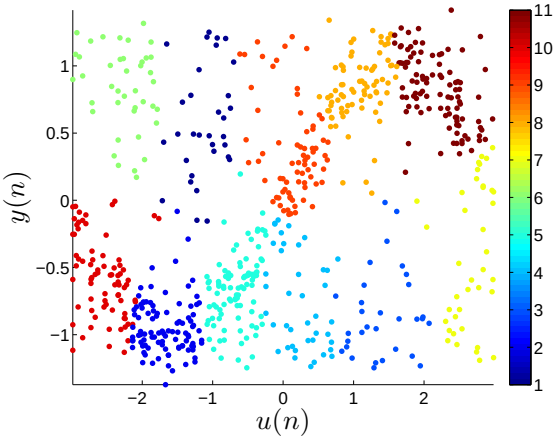
In order to evaluate sensitivity to different initialization settings of regional linear models, we show, in Figure 18, performance results obtained over 10 independent runs. Figure 18a reports the RMSE performance on the training and test stages for the one-step-ahead prediction and simulation cases. One may observe that the simulation performances



(a) Data (in red) and SOM prototypes (in blue)



(b) Clusters centers found by k-means (in black)

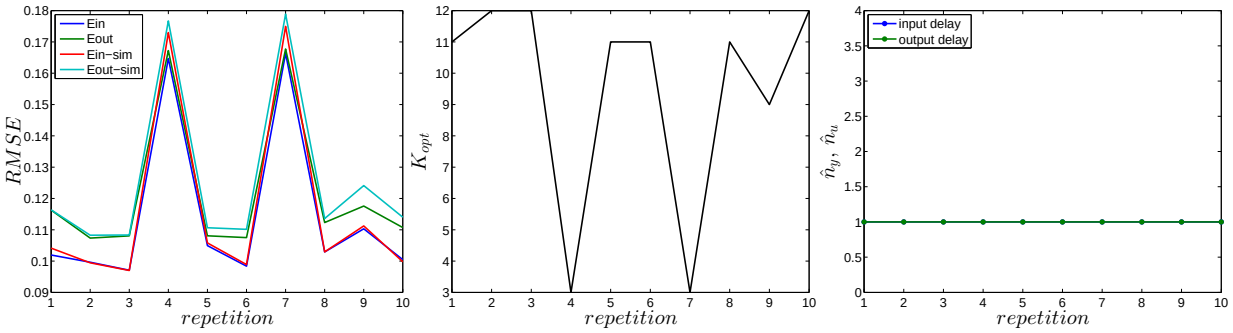


(c) Eleven regions

Figure 17: Synthetic example: steps for the partitioning of the input space via regional modeling.

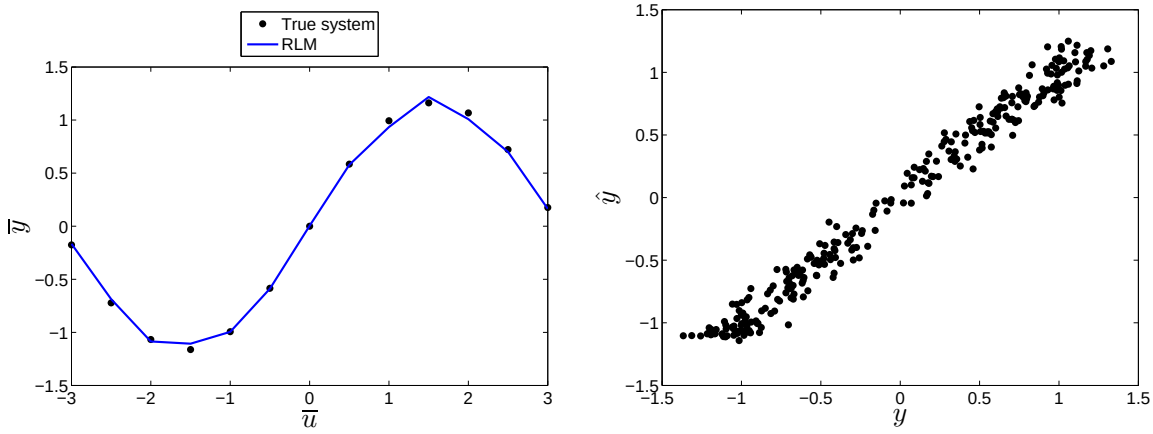
are virtually identical to the one-step-ahead prediction ones. Moreover, the out-of-sample performances track the in-sample ones, with out-of-sample error values slightly higher than the corresponding in-sample ones. By contrasting Figures 18a and 18b, the worst performances are achieved when the clustering process renders a small number of regions, $K_{opt} = 3$. In these cases (repetitions 4 and 7), three linear models can not accurately approximate the highly nonlinear dynamics presented by the synthetic example. With regard to the memory orders, the terms n_u and n_y are correctly reconstructed for all repetitions (Figure 18c), i.e., $\hat{n}_u = 1$ and $\hat{n}_y = 1$.

Estimates for the static and dynamic behaviors of the synthetic plant are reported in Figures 19a and 19b, respectively. This correspond to the best case scenario, i.e., the performances obtained by the model which reported the smallest RMSE value for the out-of-sample simulation case ($E_{out-sim} = 0.11$). From Figure 19a, one may observe that the static behavior is well reconstructed with 11 linear models. Regarding the dynamic case, the estimates $\hat{y}$ were obtained via free-run simulation of the RLM, and Figure 19b shows that the system dynamics was well approximated.



(a) RLM accuracies (b) Number of regions, K_{opt} (c) Delays

Figure 18: Synthetic example: RLM results.



(a) Static behavior (b) Dynamic behavior ($E_{out-sim} = 0.11$)

Figure 19: Synthetic example: RLM estimates.

4.2 Outlier Robust Regional Models

An important feature of the ordinary least-squares approximate solution is that it assigns the same importance to all error samples, i.e., all errors contribute the same way to the final solution. A common approach to handle this problem consists in removing outliers from data and then trying the usual least-squares (LS) fit. A more principled approach, known as *robust regression*, uses estimation methods that are not sensitive to outliers as the LS method.

Huber (1964) introduced the concept of M-estimation, where M stands for "maximum likelihood" type, and where robustness is achieved by minimizing a function other than the sum of the squared errors. Based on Huber's theory, a general M-estimator applied to the r-th RLM model minimizes the following objective function:

$$J(\boldsymbol{\beta}_r) = \sum_{\mathbf{x}(n)\in\mathcal{X}_r} \rho(y(n) - \boldsymbol{\beta}_r^T \mathbf{x}(n)) = \sum_{\mathbf{x}(n)\in\mathcal{X}_r} \rho(e_r(n)), \tag{4.10}$$

where the function $\rho(\cdot)$ computes the contribution of each error $e_r(n) = y(n) - \boldsymbol{\beta}_r^T \mathbf{x}(n)$ to the objective function and $\boldsymbol{\beta}_r$ is the parameter vector associated with the r-th regional

linear model. Reasonable choices for the function $\rho(\cdot)$ should possess the following properties (FOX; WEISBERG, 2010): (i) $\rho(e_r(n)) \geq 0$; (ii) $\rho(0) = 0$; (iii) $\rho(e_r(n)) = \rho(-e_r(n))$; and (iv) $\rho(e_r(n)) \geq \rho(e_r(n'))$, for $|e_r(n)| > |e_r(n')|$. The LS estimator is a particular M-estimator, achieved when $\rho(e_r(n)) = e_r^2(n)$.

Parameter estimation is defined by the estimating equation which is a weighted function of the objective function derivative. Let ψ to be the derivative of ρ. Differentiating ρ with respect to the parameter vector $\boldsymbol{\beta}_r$, we have

$$\sum_{\mathbf{x}(n)\in\mathcal{X}_r} \psi(y(n) - \boldsymbol{\beta}_r^T \mathbf{x}(n))\mathbf{x}(n)^T = \mathbf{0}, \tag{4.11}$$

where $\mathbf{0}$ is a $(n_u + n_y)$-dimensional row vector of zeros. Then, defining the weight function $\gamma(e_r(n)) = \psi(e_r(n))/e_r(n)$, and let $\gamma_r(n) = \gamma(e_r(n))$, the estimating equations are given by

$$\sum_{\mathbf{x}(n)\in\mathcal{X}_r} \gamma_r(n) e_r(n) \mathbf{x}(n)^T = \mathbf{0}. \tag{4.12}$$

Solving the estimating equations corresponds to solving a weighted least-squares problem, thus minimizing $\sum \gamma_r^2(n) e_r^2(n)$. It is worth noting, however, that the weights depend on the residuals, which in turn depend upon the estimated coefficients, and the estimated coefficients depend upon the weights. As a consequence, a closed-form solution for the vector $\boldsymbol{\beta}_r$ is not possible. Thus, an iterative estimation method called *iteratively re-weighted least-squares* (IRLS) (FOX; WEISBERG, 2010) is commonly used. The steps of the IRLS algorithm in the context of training the RLM model are described next.

IRLS Algorithm for RLM Training

▷ **Step 1** - Provide an initial estimate $\hat{\boldsymbol{\beta}}_r^{(0)}$ using the ordinary LS approximate solution as in Eq. (4.6).

▷ **Step 2** - At each iteration j of the IRLS algorithm, collect the residuals from the previous iteration $e_r(n)^{(j-1)}$ for all input vectors $\mathbf{x}(n) \in \mathcal{X}_r$ and then compute their corresponding weights $\gamma_r(n)^{(j-1)} = \gamma(e_r(n)^{(j-1)})$.

▷ **Step 3** - Solve for the new weighted-least-squares estimate of $\boldsymbol{\beta}_r^{(j)}$:

$$\hat{\boldsymbol{\beta}}_r^{(j)} = \left[\mathbf{X}_r^T \mathbf{B}^{(j-1)} \mathbf{X}_r\right]^{-1} \mathbf{X}_r^T \mathbf{B}^{(j-1)} \mathbf{y}_r, \tag{4.13}$$

where $\mathbf{B}^{(j-1)} = \text{diag}\{\gamma_r(1)^{(j-1)}, \gamma_r(2)^{(j-1)}, \ldots, \gamma_r(N_r)^{(j-1)}\}$ is an $N_r \times N_r$ weight matrix, and N_r is the number of elements of $\mathcal{X}_r$.

Repeat Steps 2 and 3 until either the convergence of the estimated coefficient vector $\hat{\boldsymbol{\beta}}_r^{(j)}$ or j reaches its maximum allowed value j_{max}.

Several weighting functions for the M-estimators can be chosen, such as the Huber's function (HUBER, 1964):

$$\gamma(e_r(n)) = \begin{cases} \frac{k_e}{|e_r(n)|}, & \text{if } |e_r(n)| > k_e \\ 1, & \text{otherwise.} \end{cases} \tag{4.14}$$

where the parameter k_e is a threshold value, above which the residual is penalized inversely proportional to the absolute value of its magnitude. According to Huber's function in Eq. (4.14), once the condition $|e_r(n)| > k_e$ is met, the higher the absolute value of the residual $|e_r(n)|$, the smaller the values of the weight $\gamma(e_r(n))$.

Smaller values of k_e produce more resistance to outliers, but at the expense of lower efficiency when the errors are normally distributed. In particular, $k_e = 1.345s$ for the Huber function, where s is a robust estimate of the standard deviation of the residuals. A usual approach is to take $s = \text{MAR}/0.6745$, where MAR is the median absolute residual. For details on M-estimation and its practical aspects, we recommend the text by Barros (2013).

Despite the fact that the theoretic development just presented has been focused on the RLM model, it is straightforward to apply it to any learning method which includes linear models in its formulation, such as the MLM and the ELM models.

4.3 Related works

The key aspect of regional modeling is to fit a learning model over partitions of the input space larger than those provided by purely local models. This can be accomplished by combining/merging models (KALHOR; ARAABI; LUCAS, 2011), or clustering (FERRARI-TRECATE et al., 2003; WANG; SYRMOS, 2007; ABONYI et al., 2003). Abonyi et al. (2003) proposed a local approach based on the Delaunay Tessellation of the codebook of Self-Organizing Maps to obtain non-overlapping operating regimes of dynamic systems. Applying Delaunay Tessellation is completely different from clustering SOM prototypes. In fact, even more input space partitions than SOM prototypes are delivered after performing Delaunay Tessellation. Wang and Syrmos (2007) and Ferrari-Trecate et al. (2003) discussed the application of clustering to nonlinear systems, in an attempt to model different system regimes. Their works share the characteristic of applying the clustering directly to the data samples. In Wang and Syrmos (2007), a hierarchical clustering approach is used, whereas Ferrari-Trecate et al. (2003) proposes a modified k-means algorithm as clustering method. Thus, Regional Modeling distinguishes from previous attempts by the clustering algorithm, and the two-stages clustering procedure, i.e., application of clustering over SOM prototypes instead of over data samples directly.

Some local modeling techniques allow the control of the granularity of the input space partitioning. For instance, the parameter k in k-NN models somehow relates to

partitioning granularity, such that a large number of neighbors k may cover a large input space area. However, a fixed k does not take into account the fact that data points can be non-uniformly spaced. Thus, a large k may not be adequate in the sense that the local models can be corrupted by including far away data points. In opposite, methods based on clustering of local models try to combine models which are similar, and keep separated models that are not.

4.4 Closing remarks

In this chapter we have introduced the regional modeling framework, a novel approach for nonlinear dynamic system identification based on the SOM. The proposed methodology led to the design of regional models, which stands in between the global and local models. Regional modeling extends the two-level clustering approach by Vesanto and Alhoniemi (2000) to regression problem, particularly, to system identification. The first step in regional modeling requires the partition of the input space using the SOM, followed by clustering on the prototypes of the trained SOM. In the second step, regional regression models are built over the clusters (i.e. over the regions) of SOM prototypes. Here, the optimal number of clusters of SOM prototypes was searched for using the Davies-Bouldin validity index.

Using the proposed framework, it is possible to build regional linear and nonlinear regression models. For the linear case, we used ARX models whose parameters were estimated using the ordinary least-squares method. Regional NARX models can be built using, for example, the ELM network. Additionally, we extended the regional modeling framework to handle data with outliers by developing a robust variant of the proposed regional models through the use of M-estimation. It is worth noting that the regional paradigm is also inherently able to operate on multidimensional responses (multiple input multiple output systems), since it just depends on the regression model to be used in the regional approach.

Chapter 5

Experiments

In this chapter, we carry out a performance comparison with the models studied in this thesis. The analysis is conducted over four dynamic systems, two of which correspond to synthetic systems, while the others consist of datasets of real-world plants. The experiments follow a general methodology, which we use throughout the chapter. The analysis is applied to *i*) global models (MLM, ELM, RBF, ARX/LS); *ii*) local models (kNN, NNi, ekNN, LLM, kSOM); and *iii*) regional model (RLM) for direct system identification. For the task, we use four benchmark plants, including simulated and real-world systems. Section 5.1 gives a detailed description of the experimental setup and methodology. Sections 5.2, 5.3, 5.4 describe the dynamic systems used in the experiments and report the corresponding performance results.

5.1 Methodology

As usual, the data are split into estimation and validation sets. Using the estimation samples, a *10-fold cross-validation* step is performed in order to select the best models' hyperparameters, i.e., automatic model selection. This step aims to optimize the models' hyperparameters as well as the memory order terms n_u and n_y.

The only hyperparameter of the Minimal Learning Machine, the number of reference points M, is selected through grid-search from a range of $10\% - 90\%$ of the number of learning/estimation points, with a step size of 5%. The centers of the radial basis functions in RBF networks are selected through k-means clustering, and the number of radial basis functions is optimized from the range $\{10, 20, \ldots, 100\}$. For RBF networks, we use Gaussian basis functions, whose spread values are also optimized via exhaustive search in {0.001, 0.01, 0.1, 1}. The optimal number of hidden units in ELM networks is selected from the range $\{10\%, 20\%, \ldots, 90\%\}$ of the number of estimation samples.

The regional approach has been trained using the default settings from the SOM Matlab toolbox (VESANTO et al., 2000), with the number of prototypes Q is in the order of $5\sqrt{N}$, where N corresponds to the number of training samples. By using the default settings, we expect that if the regional modeling paradigm provides an acceptable performance using its basic configuration, then this property is a good indicator for use of the method. In addition, the regional approach uses $K_{max} \approx \sqrt{Q}$.

The local linear regression approaches do not have hyperparameters to optimize, except for the kNN whose value k is selected from $\{1\%, 5\%, 10\%\}$ of the estimation points. As originally proposed, for the SOM-based local models (kSOM and LLM), we use the sequential training of a one-dimensional SOM with the same amount of prototypes used for the regional approaches, i.e. $Q \approx 5\sqrt{N}$. The initial and final learning rates are set to $\alpha_0 = 0.1$ and $\alpha_T = 0.01$. The initial and final values of the neighborhood function radius are $\sigma_0 = 10$ and $\sigma_T = 0.01$. The learning rate α' is set to 0.01 and the number of epochs is 100. For the kSOM model, we also perform an exhaustive search on the parameter k ranging from 5 to 25 (step size of 5).

Regarding the memory order terms, for the synthetic plants, we carry out grid-search over the range $\{1, 2, 3\}$, and then we select the combination of n_u, n_y that minimizes the mean squared error over the 10-fold cross-validation step. For the real-world plants, we select the memory orders from $\{1, 2, 3, 4\}$.

For model validation, we use the root mean square error (RMSE) between output estimates and observations from a validation set in both one-step-ahead predictions and free-run simulation scenarios, and residual analysis. The residual analysis consists of the correlation tests proposed by Billings and Voon (1986), which were described in Chapter 1.

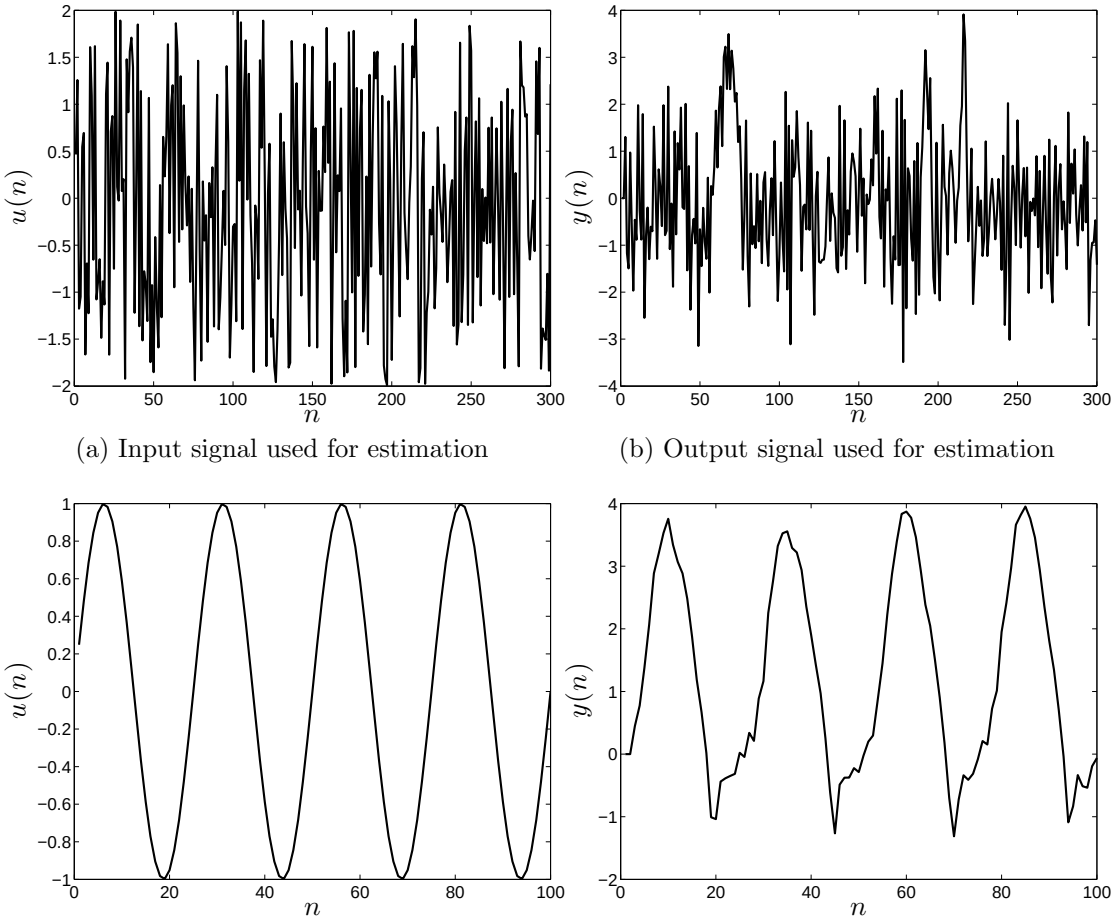
5.2 Example: Narendra's plant

In this section we describe a simulated system first introduced by Narendra and Parthasarathy (1990). The example was originally proposed to demonstrate the use of neural networks to model dynamic systems, and its equation is given by

$$y(n+1) = \frac{y(n)y(n-1)[y(n)+2.5]}{1+y^2(n)+y^2(n-1)} + u(n) + \epsilon(n), \tag{5.1}$$

where $u(n)$ and $y(n)$ denote the input and output respectively, and $\epsilon(n) \sim \mathcal{N}(0, 0.01)$. In this example, the memory order terms are $n_u = 1$ and $n_y = 2$. For the identification process, we used a random input signal uniformly distributed in the interval $[-2, 2]$, i.e., the estimation dataset was comprised of samples using $u(n) \in \mathcal{U}[-2, 2]$. A dataset of 300 samples was generated to train the models. The model validation step was carried out over a time-series of 100 samples, with $u(n) = \sin(2\pi n/25)$. Figures 20a and 20b show the input and output time-series used for estimation, while Figures 20c and 20d report the

input and output validation series. A similar experimental setup was previously designed by Verdult (2002).



(a) Input signal used for estimation

(b) Output signal used for estimation

(c) Input signal used for validation

(d) Output signal used for validation

Figure 20: Narendra's plant: time-series.

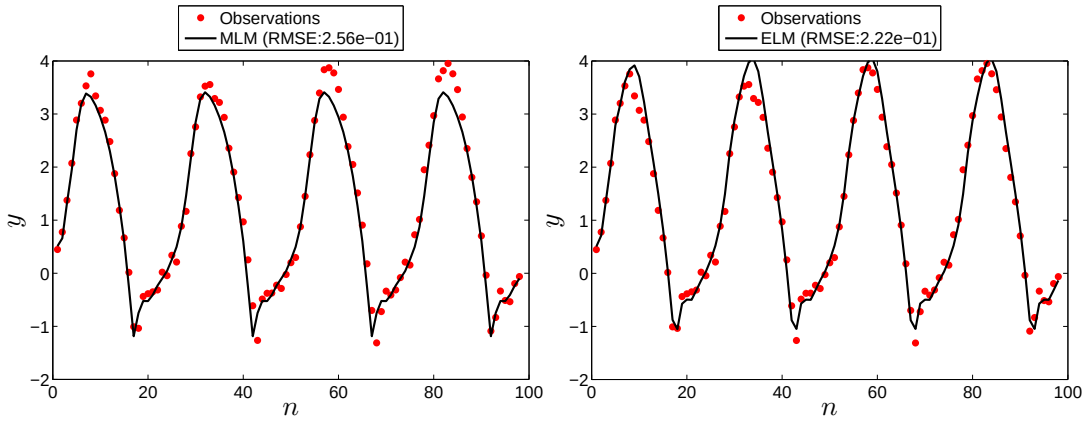
We report in Table 1 the performances on the validation set. It includes the average RMSE values in the one-step-ahead predictions and free-run simulations over 10 independent runs. The smallest RMSE averages are in boldface.

From Table 1, the best performing model is the MLM for both prediction and simulation cases. Also, the MLM presented the smallest variance (of those that vary). Contrasting ELM and RBF models, the ELM achieved better results on simulation. Among the local approaches, the NN method provides the best results for both simulation and prediction. For comparison, RLM presents similar results to what is presented by the best performing local approach. Based on the RMSE for the simulation case, the ARX/LS does not capture the system dynamics accurately. In fact, ARX/LS represents the worst model for both prediction and simulation.

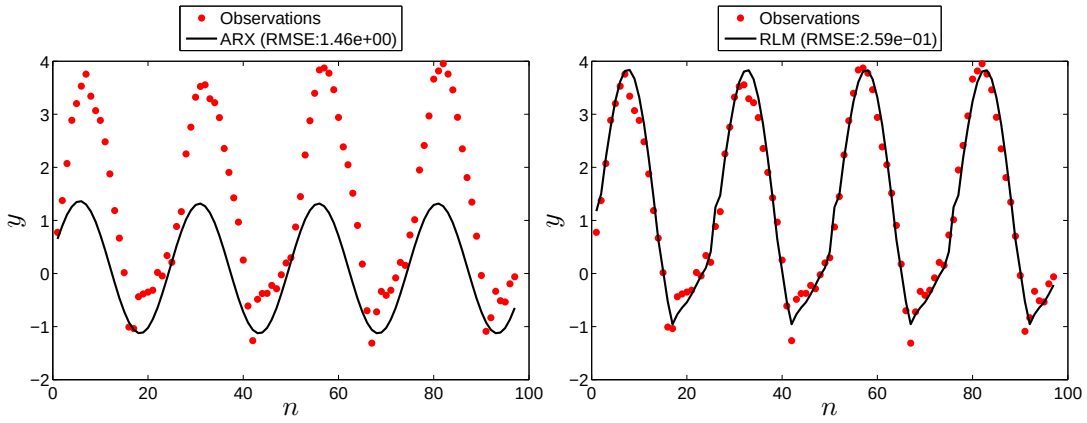
For a selection of models (MLM, ELM, ARX and RLM), we illustrate in Figure 21 the models simulations on the validation set. The red dots represent the target outputs

Table 1: Averages on validation performance (RMSE): Narendra's plant.

Models	Prediction		Simulation	
	mean	*std*	*mean*	*std*
MLM	**1.84e-01**	4.18e-03	**2.71e-01**	8.01e-03
ELM	2.86e-01	1.10e-01	3.43e-01	8.05e-02
RBF	2.58e-01	8.83e-02	5.50e-01	5.21e-01
ARX	6.31e-01	-	1.43e+00	-
RLM	2.34e-01	3.83e-02	4.20e-01	1.34e-01
LLM	3.64e-01	5.72e-02	6.19e-01	1.31e-01
kSOM	2.25e-01	1.66e-02	4.41e-01	7.82e-02
kNN	3.19e-01	-	6.25e-01	-
NN	2.13e-01	-	4.22e-01	-
NNi	2.92e-01	-	5.20e-01	-
ekNN	2.02e-01	-	4.27e-01	-



(a) MLM ($\hat{n}_u = 1$, $\hat{n}_y = 2$) (b) ELM ($\hat{n}_u = 1$, $\hat{n}_y = 2$)



(c) ARX ($\hat{n}_u = 3$, $\hat{n}_y = 3$) (d) RLM ($\hat{n}_u = 1$, $\hat{n}_y = 2$)

Figure 21: Narendra's plant: simulations on the validation set.

(observations), and the black lines denote the approximation of the models. The depicted simulations correspond to the best case (smallest RMSE) performance of each model. The MLM does not to accurately approximate the observations with high values, while

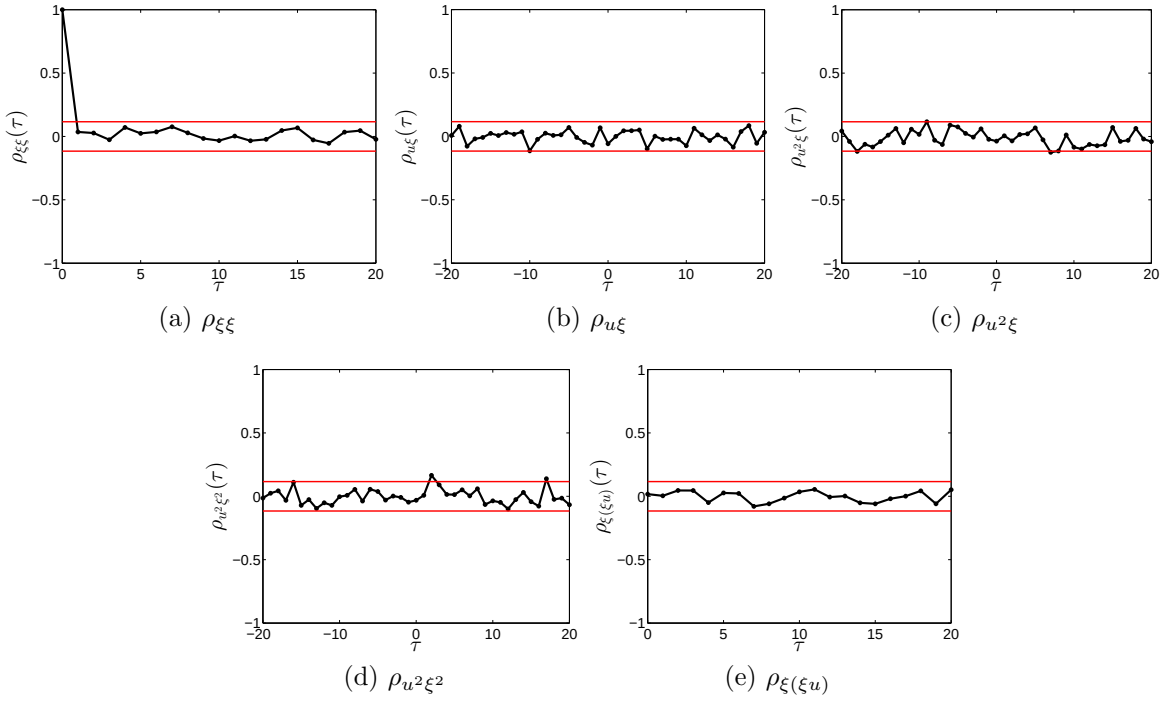


Figure 22: Narendra's plant: residual analysis for the MLM.

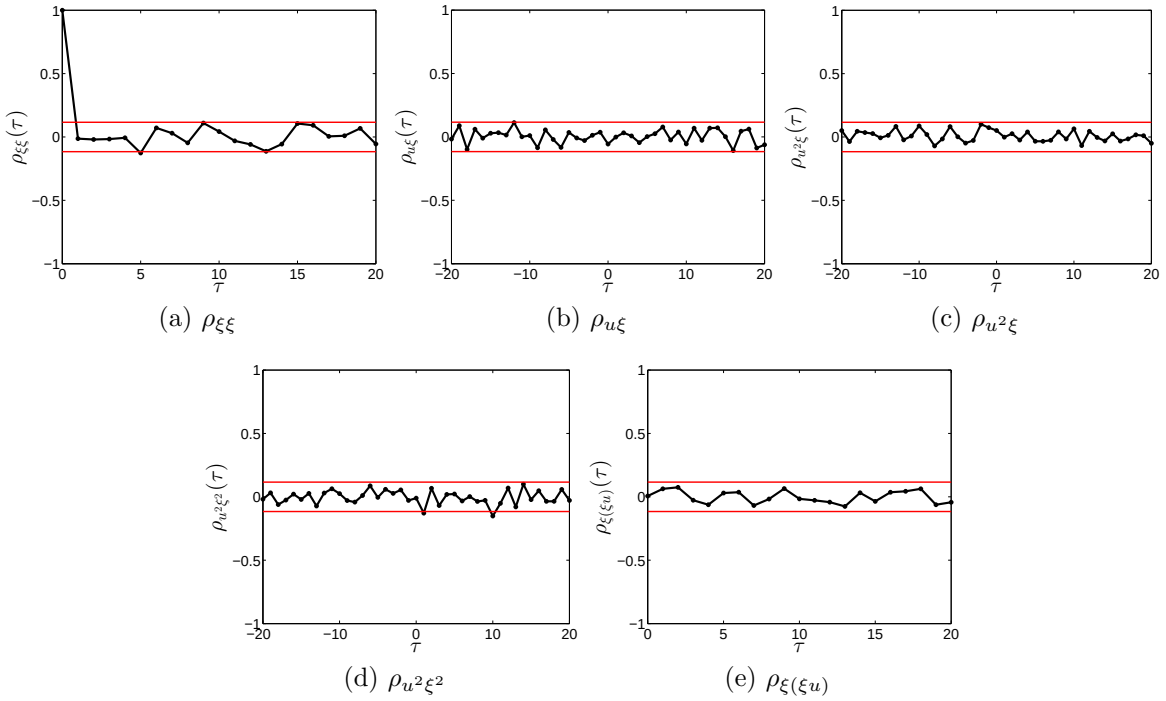
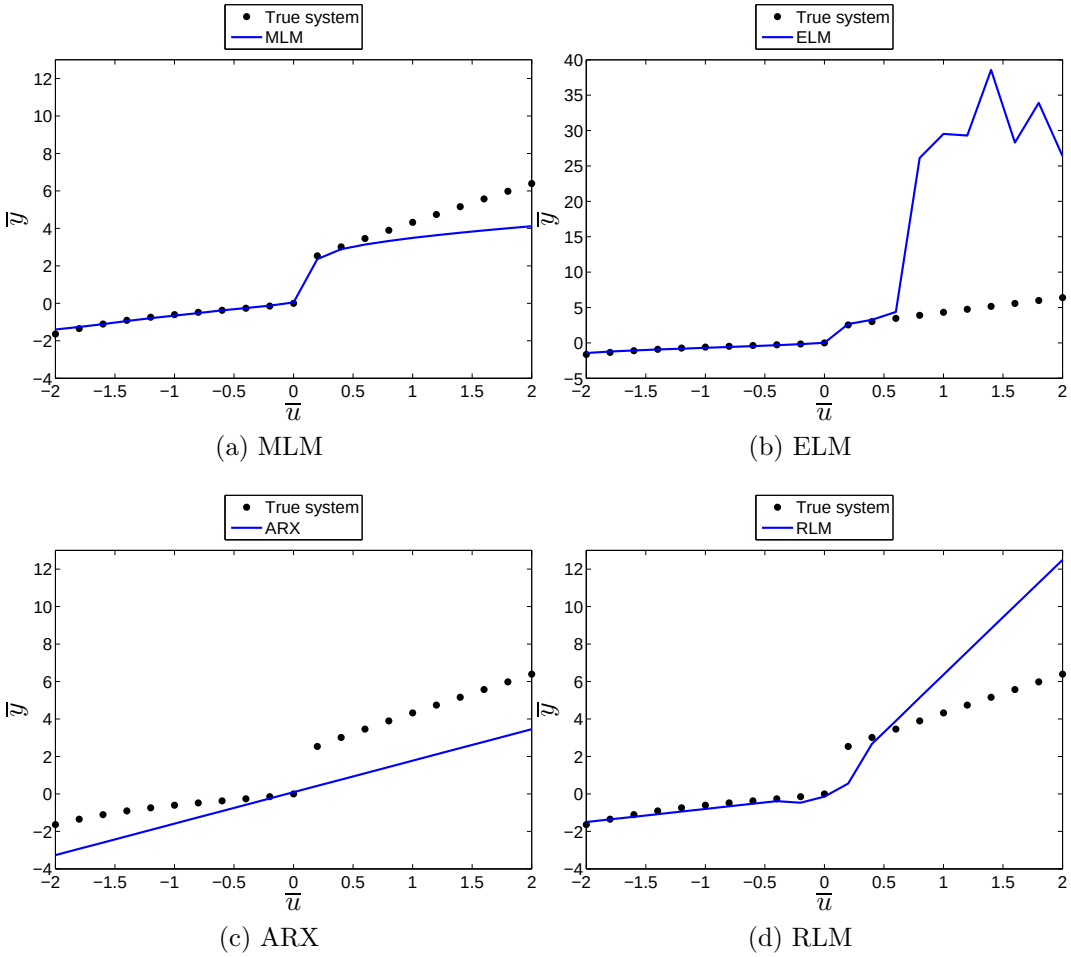


Figure 23: Narendra's plant: residual analysis for the ELM.

the RLM and ELM seem to do it. With regard to the best case, the ELM achieved the smallest RMSE value ($2.22e-01$). Clearly, the ARX/LS model does not capture the system dynamics. It just reproduces the sinusoidal behavior of the input signal. Considering the

correct memory order, the MLM, RLM and ELM models recovered the correct values of n_u and n_y (i.e., $n_u = 1$ and $n_y = 2$), while the ARX/LS presented a tendency towards high order terms.



(a) MLM (b) ELM

(c) ARX (d) RLM

Figure 24: Narendra's plant: static behavior.

Figures 22 and 23 show the correlation tests with a confidence interval of 95% for the MLM and ELM, respectively. The MLM and ELM were the two best performing models. The correlation tests were applied to the best results (smallest RMSE) achieved with the MLM and ELM over the 10 independent repetitions. As one may observe, overall, the MLM and ELM passed all the statistical tests.

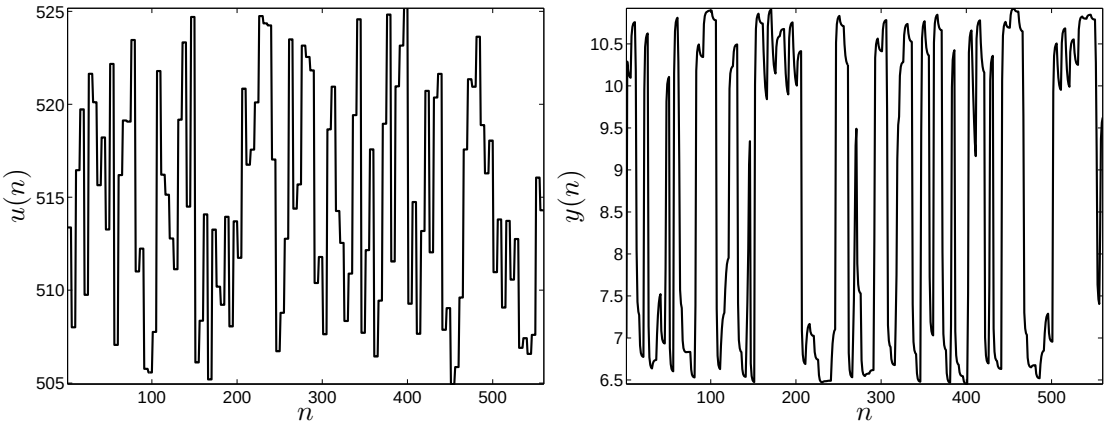
Figure 24 illustrates the static behavior of the plant and the approximation provided by the models — MLM, ELM, ARX and RLM — for a constant input signal $\bar{u}$ in the interval $[-2, 2]$. The MLM, ELM and RLM models accurately approximate the static behavior in the range $[-2, 0.5]$. For values of $\bar{u}$ higher than 0.5, the approximation becomes rather poor. Interestingly, the MLM is the model that best approximates $\bar{y}$ for high values of $\bar{u}$, while the ELM network achieves extremely poor performance in such a range. As well, the ARX model does not approximate the static behavior, even for small values of $\bar{u}$.

5.3 Example: pH dynamics

This section presents performance evaluation over a chemical process system. The system under study corresponds to the dynamic response of pH in a continuous stirred tank reactor (CSTR). The CSTR has two input streams, one containing sodium hydroxide and the other acetic acid. A dynamic model for the pH in the tank was proposed by McAvoy, Hsu and Lowenthal (1972). From material balances on Na^+ and total acetate $HAC + AC^-$ and assuming that acid-base equilibria and electroneutrality relationships hold, the acetate balance, sodium ion balance, HAC equilibrium, water equilibrium and electroneutrality can be written respectively as:

$$\begin{aligned} V\dot{\zeta} &= F_1C_1 - (F_1 + F_2)\zeta \\ V\dot{\eta} &= F_2C_2 - (F_1 + F_2)\eta \\ \frac{[AC^-][H^+]}{[HAC]} &= K_a \\ [H^+][OH^-] &= K_w \\ \eta + [H^+] &= [OH^-] + [AC^-] \end{aligned}$$

The parameters for the CSTR used in this thesis are given by Bhat and McAvoy (1990). A training dataset was developed by forcing the F_2 stream with a 2% PRBS signal superimposed upon its steady state value ($F_2 = 515$). The F_2 and pH response used for model estimation are shown in Figure 25. A full set of 800 samples was generated and 70% of the samples were used for estimation and the rest for validation/test.



(a) Rate of $NaOH$ (F_2) (b) pH values

Figure 25: pH dynamics: estimation/training data.

Table 2 reports the average performance (RMSE) in terms of prediction (one-step-ahead) and simulation over 10 independent runs. The best performing model is MLM, followed by the NNi model. Also, the MLM presents the most stable performance,

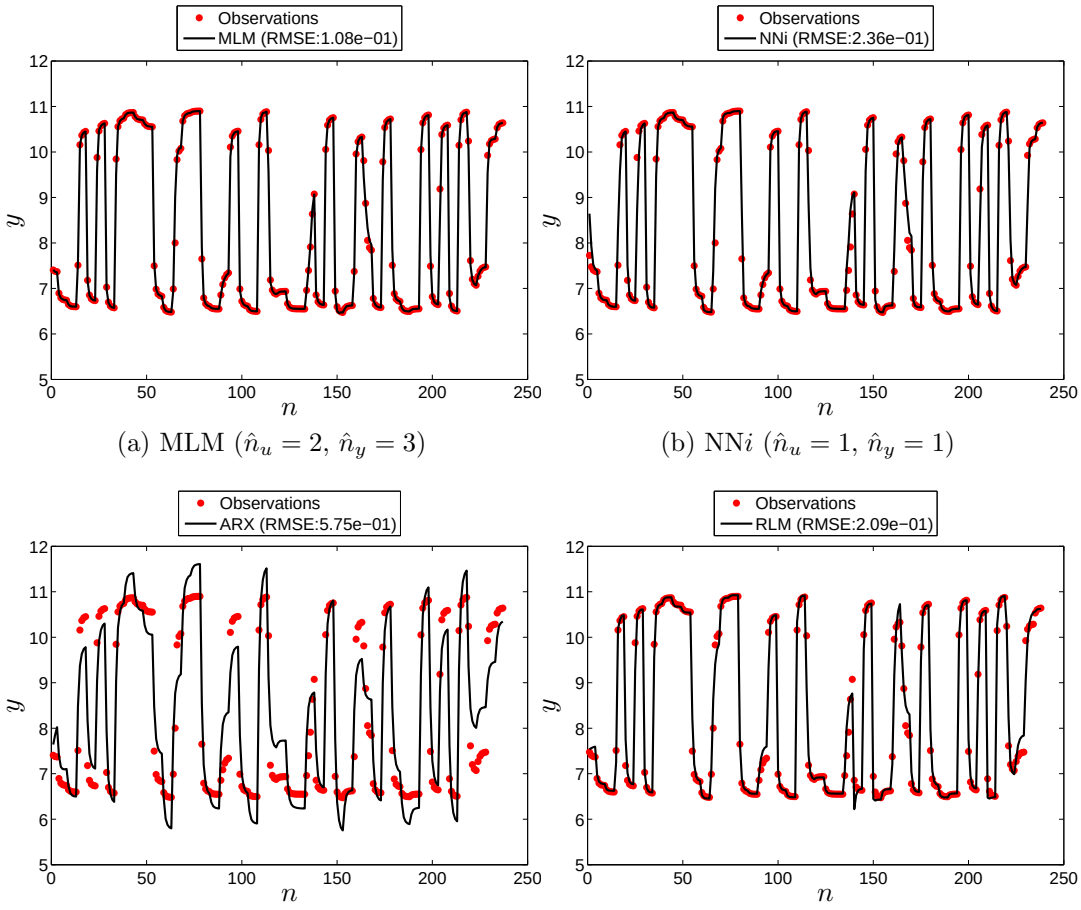
represented by the smallest variance. This is an interesting result, since MLM is based on a random selection of reference points. It can be explained by a selection of a large number of reference points M during the cross-validation process. In fact, for all repetitions, M was selected equals either $0.8N$ or $0.9N$, where N denotes the number of samples used for estimation. The LLM and kSOM models achieve results only slightly better than what is obtained with an linear global model (ARX/LS). It means that the use of multiple linear models have not been beneficial. The regional linear model approach achieved results which are comparable to what is provided by the LLR methods (except ekNN) and RBF network. In opposite, the ELM network "explodes" when used in the simulation mode. The reason for why the ELM "explodes" relies on two issues: *normalization* and *regularization*. As a solution, normalizing the input regression vectors makes them lie in a more suitable range of values, considering the domain of the nonlinear activation function in the hidden layer. However, after normalization, the matrix comprised of the activation values of the hidden layers can still be low-rank. It poses numerical problems for computing the pseudo-inverse of such a matrix. One alternative to address the problem is through *ridge regression* (regularization). For instance, if we normalize the data comprised of the regression vectors $\{\mathbf{x}(n)\}$ to have zero-mean and unit-variance, and apply ridge regression with regularization coefficient equal to $\lambda = 10^{-4}$, the RMSE value provided by the ELM network for the simulation case becomes 0.30 (much smaller than 1.63e+09). Neumann and Steil (2013) and Horata, Chiewchanwattana and Sunat (2013) have already warned for such numerical problems in ELMs.

Table 2: Averages on validation performance (RMSE): pH dynamics.

Models	Prediction		Simulation	
	mean	*std*	*mean*	*std*
MLM	**1.17e-01**	4.64e-03	**1.16e-01**	5.13e-03
ELM	5.83e-01	1.69e-01	1.63e+09	2.83e+09
RBF	1.79e-01	2.36e-02	2.95e-01	1.75e-01
kSOM	3.45e-01	6.10e-02	4.03e-01	8.71e-02
LLM	3.84e-01	1.99e-02	4.08e-01	2.68e-02
RLM	2.74e-01	5.34e-02	2.91e-01	7.01e-02
kNN	2.30e-01	-	2.42e-01	-
ARX	4.68e-01	-	5.75e-01	-
NN	2.44e-01	-	2.72e-01	-
NNi	2.17e-01	-	2.36e-01	-
ekNN	2.00e+09	-	-	-

In order to visually assess performance in terms of the approximation provided by the models, we show in Figure 26 the best-case performances achieved by MLM, NNi, ARX/LS and RLM models on the validation set. We observe that the MLM approximates the system dynamics quite accurately. The NNi model also reconstructs mostly the system dynamics, with approximation errors mainly around samples 1 and 140. In comparison to

MLM and NNi, the ARX/LS model performs poorly. Even though the NN model achieved smaller orders than the RLM, the best-case RLM error is slightly smaller than what is achieved by the NN method.



(a) MLM ($\hat{n}_u = 2$, $\hat{n}_y = 3$)

(b) NNi ($\hat{n}_u = 1$, $\hat{n}_y = 1$)

(c) ARX ($\hat{n}_u = 3$, $\hat{n}_y = 3$)

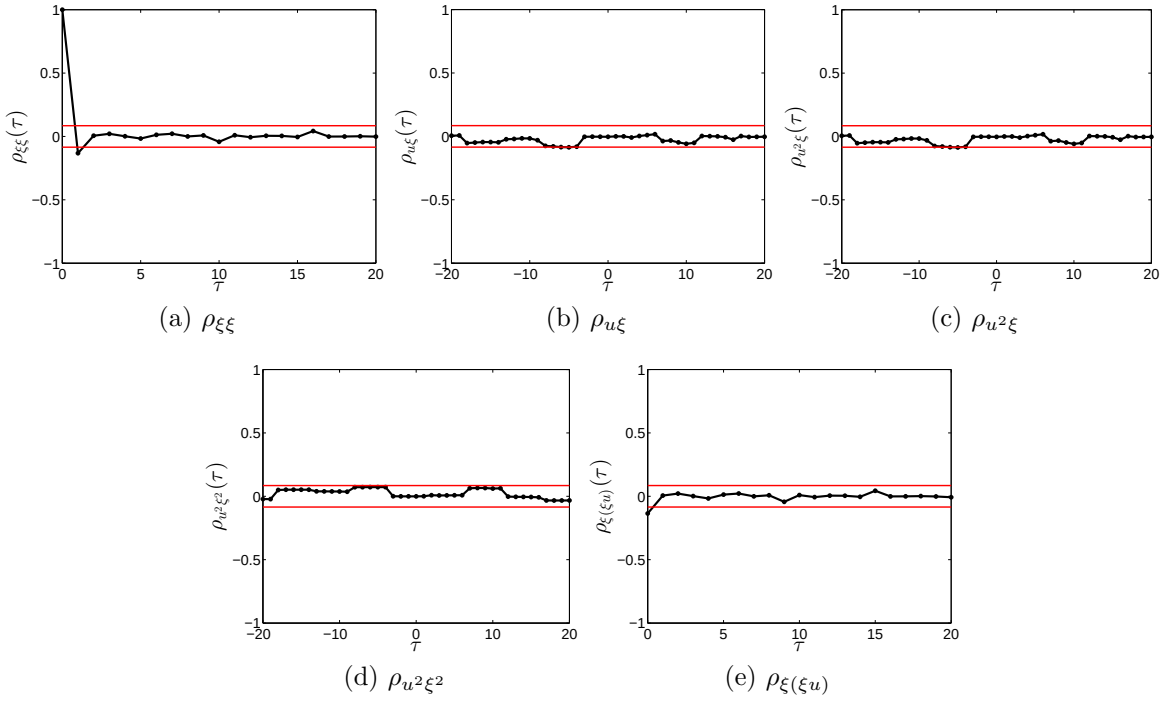
(d) RLM ($\hat{n}_u = 2$, $\hat{n}_y = 2$)

Figure 26: pH dynamics: simulations on the validation set.

Figure 27 shows the residual analysis for the MLM model with a confidence interval of 95% (represented by the red lines). Overall, the MLM passed the correlation tests. Likewise, Figure 28 illustrates the residual analysis for the RLM method. Again, the correlation tests were mostly satisfied. The tests were applied to the best performing case of each model, i.e., from the repetition that reported the smallest RMSE on simulation.

5.4 Identification of a hydraulic actuator

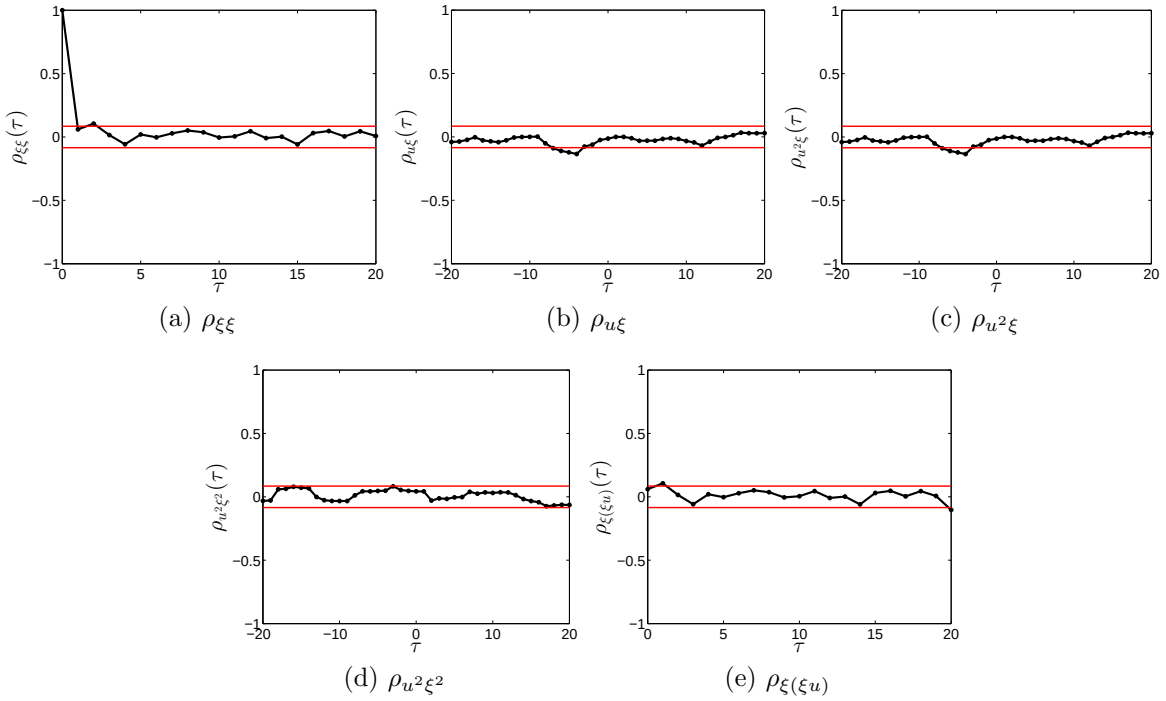
This plant consists of input and output time series of a hydraulic actuator of a mechanical crane. The structure has four actuators: one for the rotation of the whole structure, one to move the arm, one to move the forearm and one to move a telescopic extension of the forearm. This plant was chosen because it has a long arm and a long



(a) $\rho_{\xi\xi}$ (b) $\rho_{u\xi}$ (c) $\rho_{u^2\xi}$

(d) $\rho_{u^2\xi^2}$ (e) $\rho_{\xi(\xi u)}$

Figure 27: *pH* dynamics: residual analysis for the MLM.



(a) $\rho_{\xi\xi}$ (b) $\rho_{u\xi}$ (c) $\rho_{u^2\xi}$

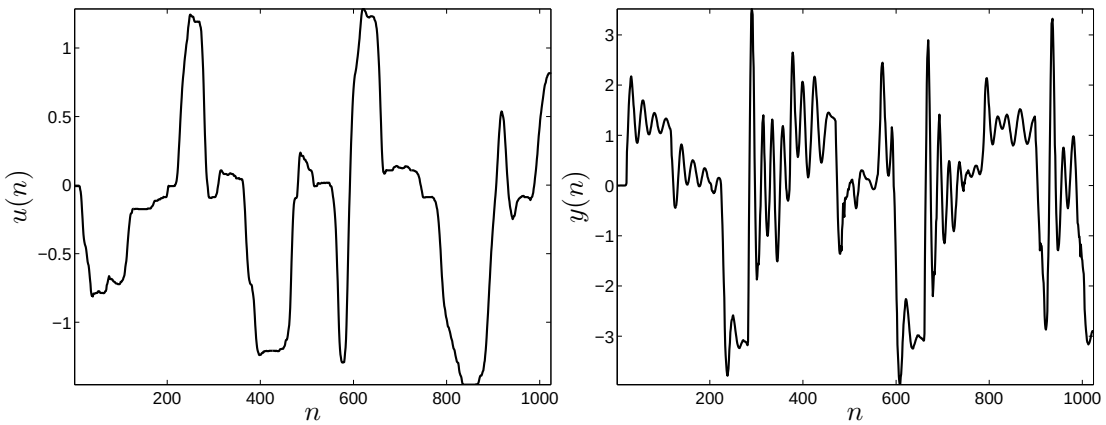
(d) $\rho_{u^2\xi^2}$ (e) $\rho_{\xi(\xi u)}$

Figure 28: *pH* dynamics: residual analysis for the RLM.

forearm with considerable flexibility on mechanic structure, making the movement of the whole crane oscillative and hard to control.

The position of the arm is controlled by a hydraulic actuator. The oil pressure

in the actuator is controlled by the valve opening through which the oil flows into the actuator. The position of the robot arm is then a function of the oil pressure. Figure 29 shows measured values of the valve position (input time series, $\{u(t)\}$) and the oil pressure (output time series, $\{y(t)\}$), which are input and output signals, respectively. As seen in the oil pressure, it contains a very oscillative settling period after a step change of the valve opening. These oscillations are caused by mechanical resonances in the robot arm (SJÖBERG et al., 1995).



(a) Measured values of valve position. (b) Measured values of the oil pressure.

Figure 29: Hydraulic actuator plant.

The dataset related to the hydraulic actuator plant is comprised of 1024 input-output samples and it is available at `http://www.esat.kuleuven.be/sista/daisy/`. Half of the time series was used for estimation and the rest for validation. Following the previous methodology for model selection and with the lag terms n_y and $n_u \in \{1, 2, 3, 4\}$, Table 3 shows the averages and standard deviations of RMSE values drawn from 10 independent runs of one-step-ahead predictions and free simulation on the validation set.

On the basis of Table 3, we observe that the prediction results are quite similar for all the methods, except for ekNN and LLM. It corresponds to the classical situation in which even wrong (not accurate) models look very accurate based on one-step predictions. This model inadequacy is represented by a high RMS error in simulation. Among the evaluated methods, MLM, NN and NNi are the best performing ones. ELM and RBF networks performs quite poorly, with results that are much worse than what achieved by a global linear model (ARX/LS) — on the simulation case. Also, the MLM presents a stable performance, represented by a small variance in comparison to ELM and RBF models. This is an interesting result, since MLM is based on a random selection of reference points. It can be explained by a large number of reference points M, selected during the cross-validation process. For all the runs, M was selected equals $0.9N$. Figure 30 shows that, in fact, the cross-validation error decreases with the number of reference points. The multiple lines in Figure 30 denote the 10 independent runs.

Table 3: Averages on validation performance (RMSE): Actuator.

Models	Prediction		Simulation	
	mean	*std*	*mean*	*std*
MLM	**1.01e-01**	1.26e-03	**5.36e-01**	9.72e-02
ELM	1.55e-01	2.29e-02	1.40e+01	2.35e+01
RBF	1.05e-01	1.65e-03	1.14e+02	1.41e+02
kSOM	1.40e-01	2.39e-02	3.54e+07	1.12e+08
LLM	2.45e-01	1.37e-02	1.57e+00	8.57e-02
RLM	1.14e-01	1.96e-03	8.16e-01	8.56e-02
kNN	1.14e-01	-	9.66e-01	-
ARX	1.10e-01	-	9.93e-01	-
NN	1.10e-01	-	7.17e-01	-
NNi	1.06e-01	-	7.29e-01	-
ekNN	2.37e-01	-	1.41e+00	-

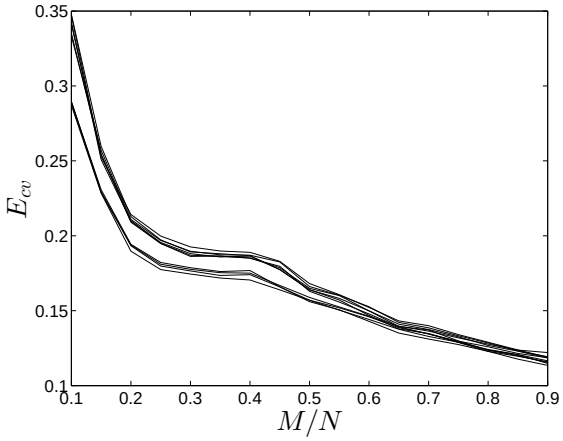
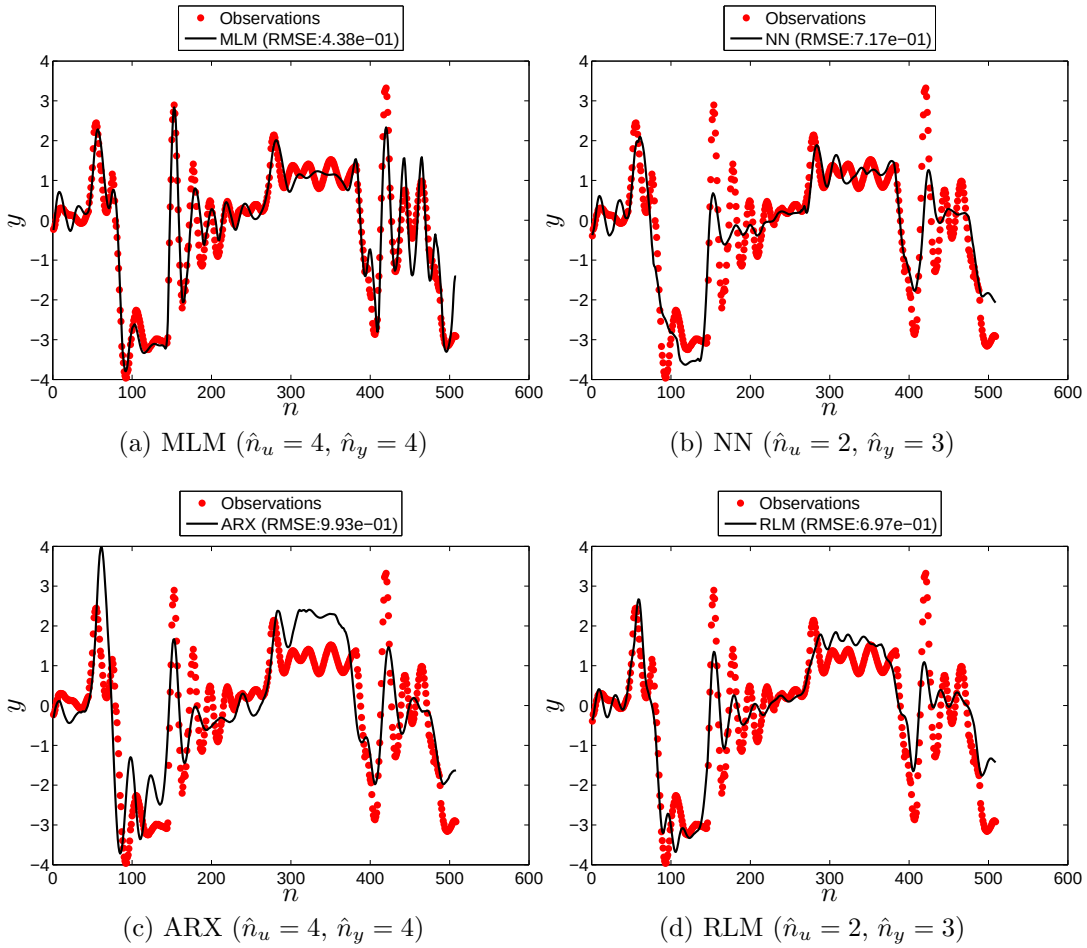


Figure 30: Actuator plant: RMSE values on cross-validation per number of reference points.

It is important to understand which factors affected ELM and RBF models, and led to an inaccurate performance. A first look may point to ill-conditioned matrices. For that, we tried a Tikhonov regularized version of ELM (DENG; ZHENG; CHEN, 2009), which consists in adding a small term λ to the diagonal of the matrix $\mathbf{Z}^T\mathbf{Z}$ in the LS solution for the weights of the output layer. The regularized ELM achieves RMS error equal to 0.76, with a regularization term $\lambda = 10^{-4}$. Similar improvement through regularization can be achieved for RBF networks as well.

Figure 31 illustrates the best-case simulation performances of the MLM, NN, ARX and RLM models. As can be seen, the MLM model best reproduces the system dynamics whereas the ARX/LS is rather inaccurate. The RLM performance stands in between MLM and ARX ones, with many parts of the output time-series being poorly reconstructed. With regard to the memory order, the MLM and ARX methods provided higher order models than the NN and RLM.

Figures 32a-32e show the residual analysis for the MLM. The MLM passes the



(a) MLM ($\hat{n}_u = 4$, $\hat{n}_y = 4$)

(b) NN ($\hat{n}_u = 2$, $\hat{n}_y = 3$)

(c) ARX ($\hat{n}_u = 4$, $\hat{n}_y = 4$)

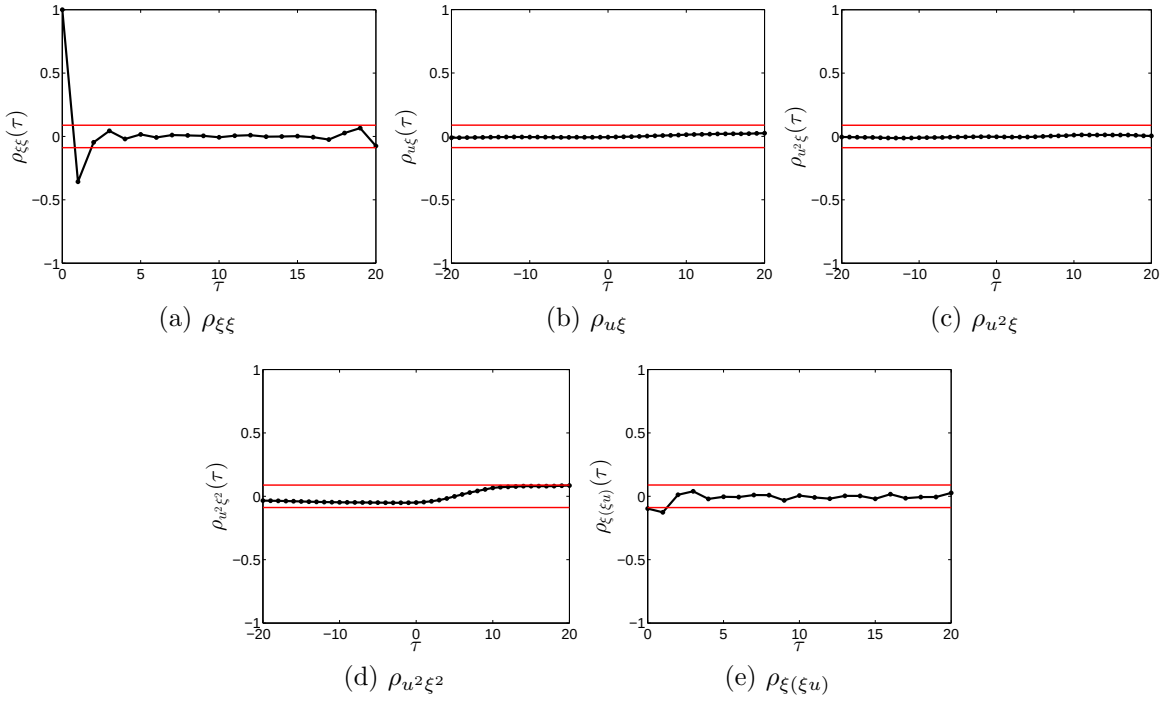
(d) RLM ($\hat{n}_u = 2$, $\hat{n}_y = 3$)

Figure 31: Actuator plant: simulations on the validation set.

correlation tests mostly, except to autocorrelation (Figure 32a) with $\tau = 1$. Figure 33 illustrates the residual analysis for the RLM model — the second best performing model, considering the best case in simulation—, which, overall, passes the correlation tests as well. Although, one may observe inadequacy in the autocorrelation function ($\rho_{\xi\xi}$) and the cross-correlation between u^2 and ξ^2 ($\rho_{u^2\xi^2}$), where the confidence interval of 95% is not satisfied.

5.5 Identification of a heater with variable dissipation

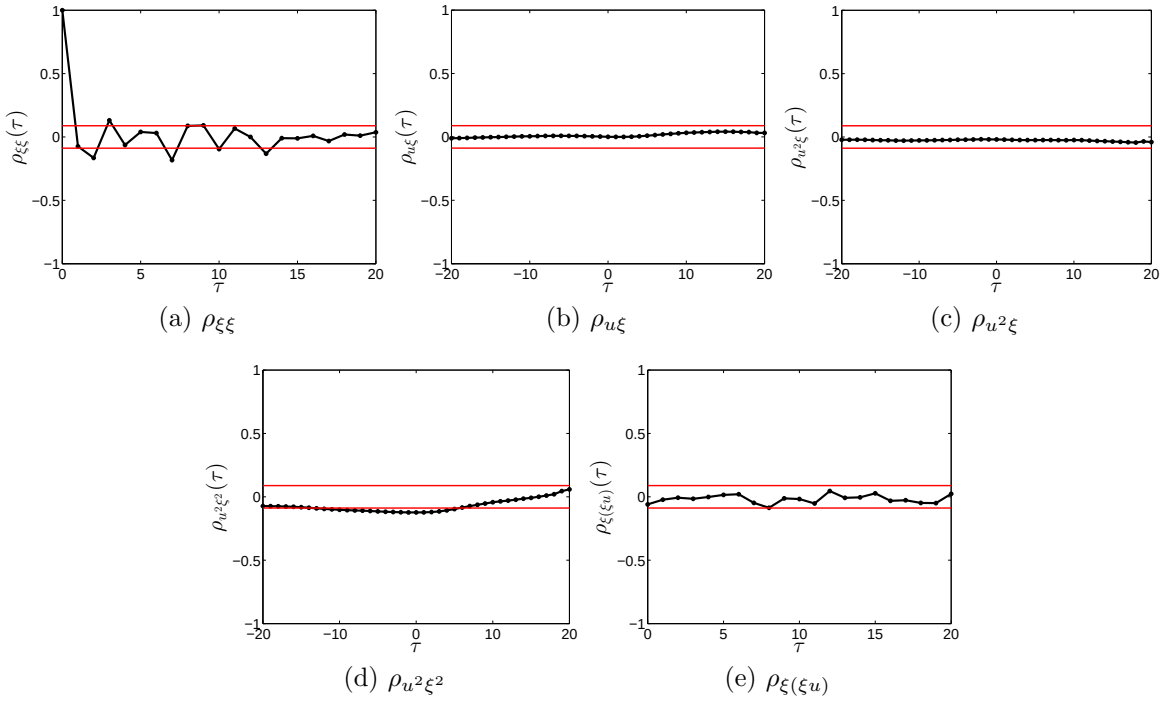
The system to be identified in this section consists of a electric heater with variable dissipation, proposed by Aguirre (2007). The variation is produced by connecting a fan to the system. The input and output signals are the voltage applied to the heater and the amplified output of a thermocouple, respectively. We are interested in modeling the dynamics of the heater. The input consists a random signal, in which each random level was kept constant for 10 seconds. The chosen sampling time was 1 seconds, but the signals were downsampled with factor equal to 12, resulting in a sampling time of 12 seconds. The



(a) $\rho_{\xi\xi}$ (b) $\rho_{u\xi}$ (c) $\rho_{u^2\xi}$

(d) $\rho_{u^2\xi^2}$ (e) $\rho_{\xi(\xi u)}$

Figure 32: Hydraulic actuator: residual analysis for the MLM.



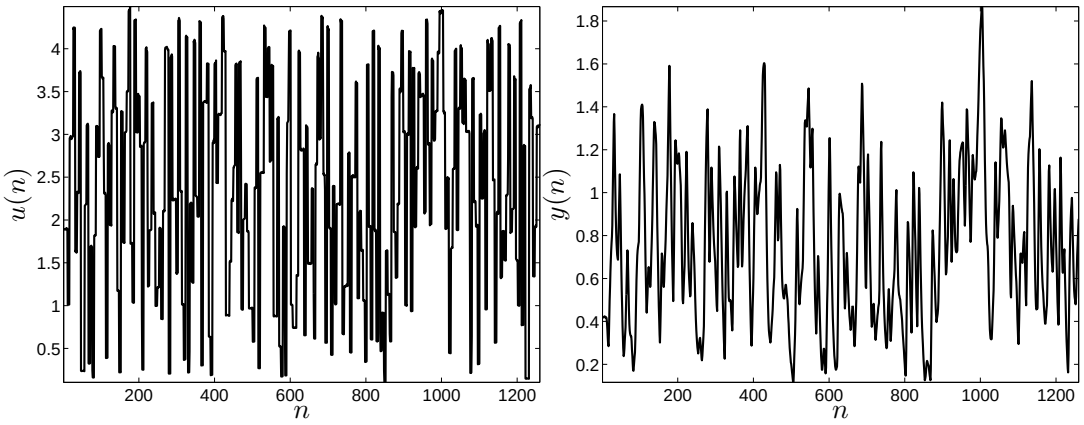
(a) $\rho_{\xi\xi}$ (b) $\rho_{u\xi}$ (c) $\rho_{u^2\xi}$

(d) $\rho_{u^2\xi^2}$ (e) $\rho_{\xi(\xi u)}$

Figure 33: Hydraulic actuator: residual analysis for the RLM.

signals used in the first 7500 seconds of the experiments were used for identification, while the 7500 seconds subsequent were used for validation. The input and output time-series collected from the whole experiment is illustrated in Figure 34. The dataset is available by

its proponent at `http://www.cpdee.ufmg.br/~MACSIN/`.



(a) Input signal (b) Output signal

Figure 34: Heater system: input and output time-series.

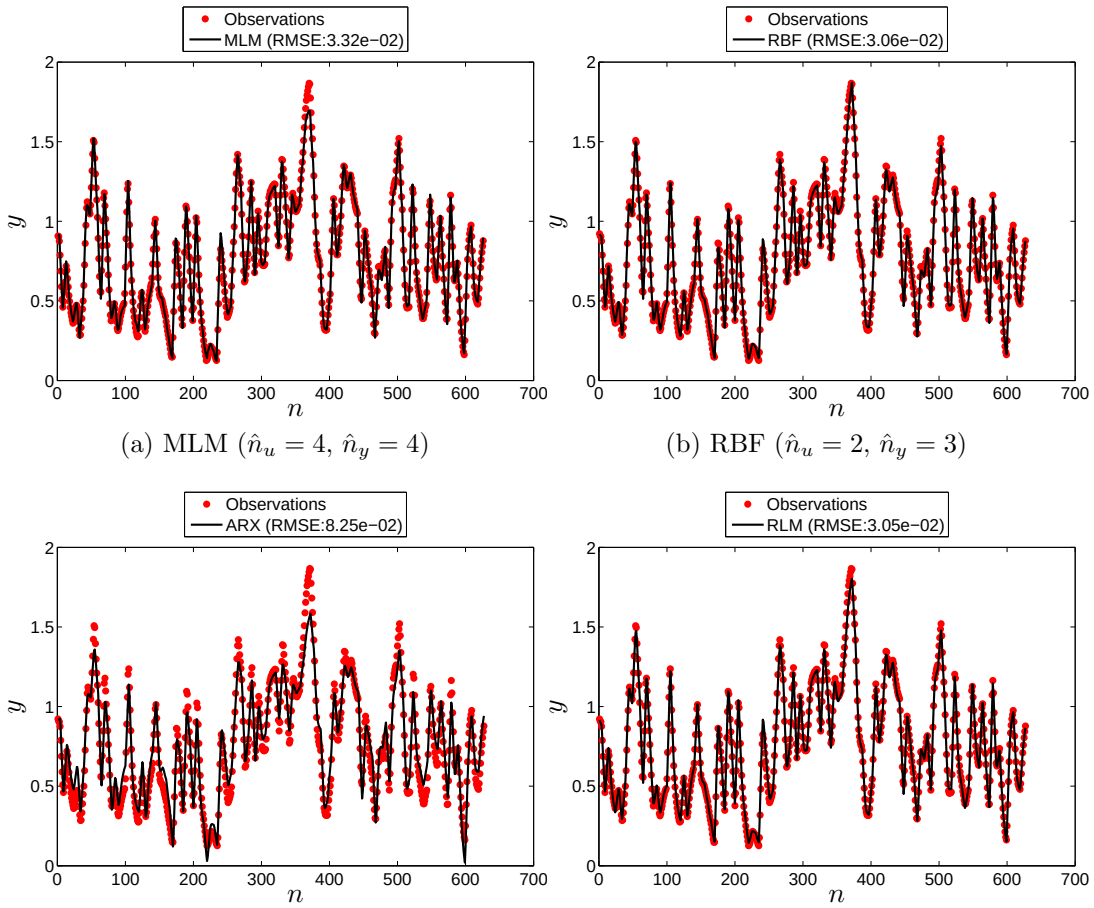
Table 4 reports the averages and standard deviations of RMSE values on the validation set over 10 repetitions for prediction and simulation matters. The RBF network achieved the smallest RMSE value on simulation. In fact, the results achieved by MLM, ELM, RBF, RLM, NN, NNi and ekNN are similar. For the first time, the MLM was not the best performing model neither in prediction nor in simulation. However, the MLM performance did not degrade as much as kNN did in the simulation case, where it returned the highest RMSE average. On prediction, both methods performed comparably.

Table 4: Averages on validation performance (RMSE): Heater.

Models	Prediction		Simulation	
	mean	*std*	*mean*	*std*
MLM	2.32e-02	1.35e-03	3.50e-02	1.19e-03
ELM	1.14e-02	1.67e-04	3.16e-02	1.57e-03
RBF	**1.14e-02**	1.25e-04	**3.11e-02**	3.82e-04
kSOM	1.78e-02	3.69e-03	1.43e-01	1.05e-01
LLM	7.07e-02	3.18e-03	1.24e-01	1.48e-02
RLM	1.15e-02	1.53e-04	3.22e-02	1.90e-03
kNN	2.60e-02	-	2.03e-01	-
ARX	1.43e-02	-	8.23e-02	-
NN	1.19e-02	-	3.31e-02	-
NNi	1.20e-02	-	3.68e-02	-
ekNN	1.17e-02	-	3.54e-02	-

Figure 35 depicts the approximation of selected models (MLM, RBF, ARX and RLM) in the simulation of the dynamic behavior of the heater system when excited using the validation input signal. The plots correspond to the best cases achieved over ten independent runs. As one may observe, the RBF and RLM presents indistinguishable

approximations. The MLM can not approximate the peak around sample $n = 380$. This is the reason for MLM has achieved higher RMSE value than the RBF and RLM models. The ARX/LS model does not capture the dynamics at the peaks and valleys of the output time-series.



(a) MLM ($\hat{n}_u = 4$, $\hat{n}_y = 4$)

(b) RBF ($\hat{n}_u = 2$, $\hat{n}_y = 3$)

(c) ARX ($\hat{n}_u = 3$, $\hat{n}_y = 2$)

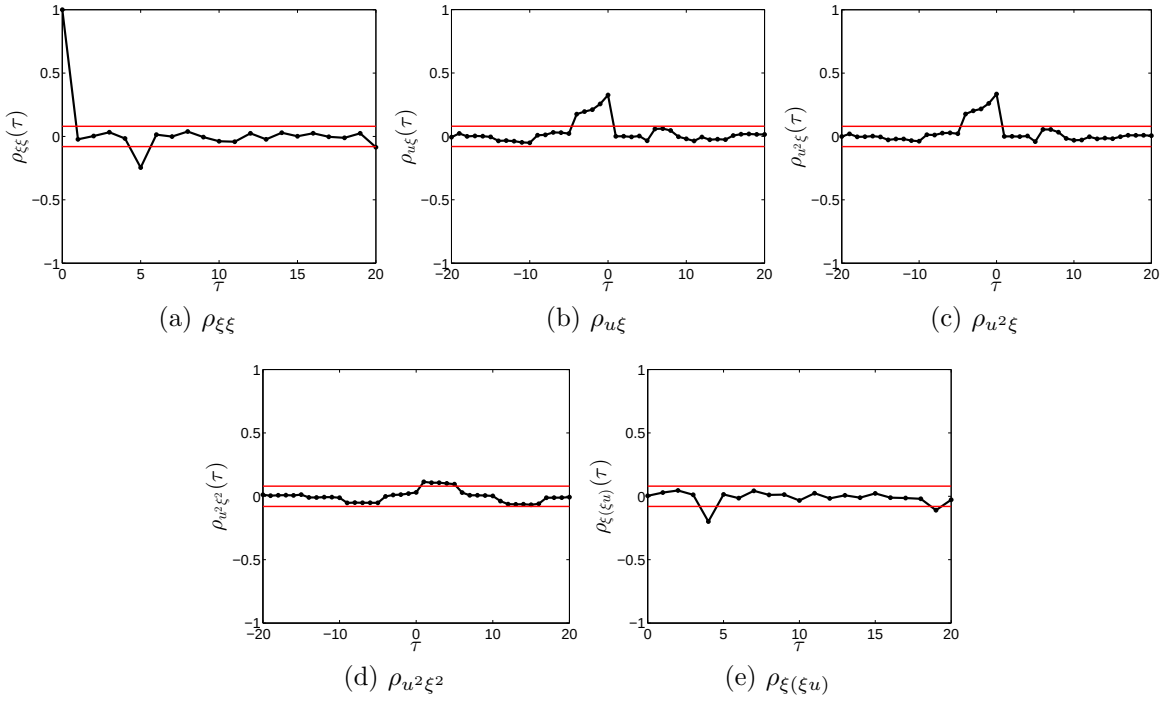
(d) RLM ($\hat{n}_u = 1$, $\hat{n}_y = 3$)

Figure 35: Heater: simulations on the validation set.

Figures 36 and 37 show the correlation tests on the residuals provided by the models with smallest RMSE — the RBF and RLM models. As one may observe, the RBF model does not pass the correlation tests in almost all tests. It represents that the RBF contains unmodeled dynamics. The correlation tests results for the RLM are indistinguishable from those reported for the RBF.

5.6 Closing remarks

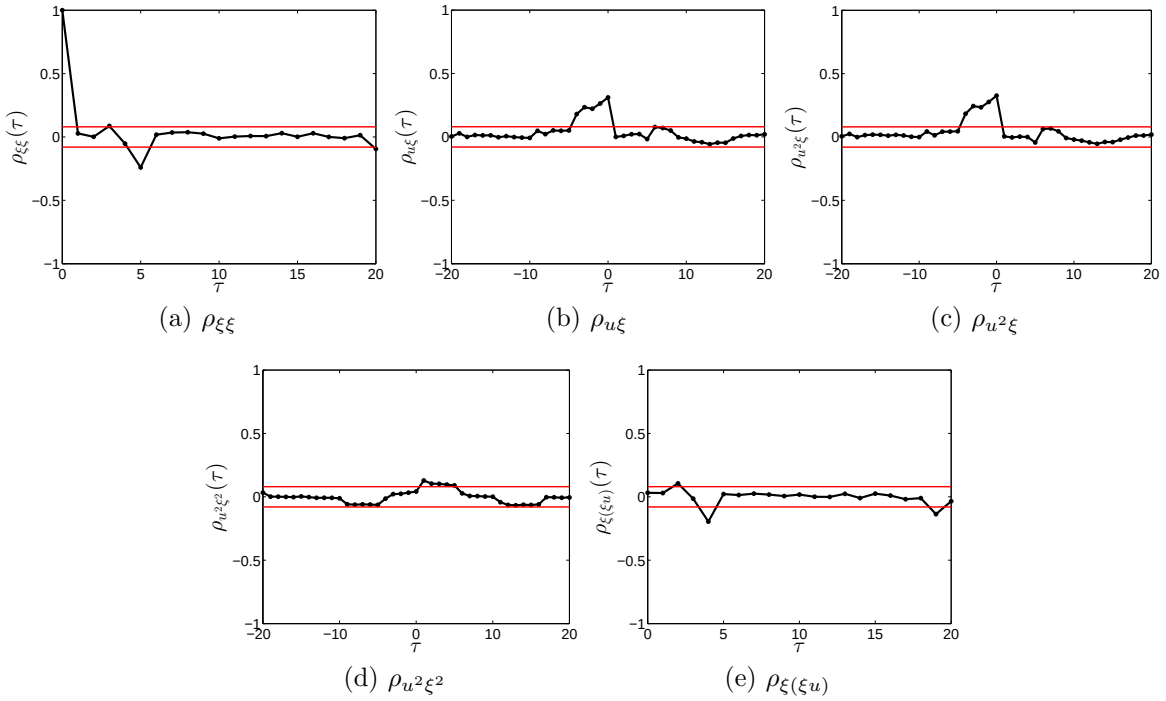
This chapter reported the performance results of the two main proposals of this thesis (MLM and RLM) and compared them against global and local standard methods in the literature of system identification. The performance assessment was carried out over four benchmarking datasets, corresponding to SISO systems. We were interested in the dynamic



(a) $\rho_{\xi\xi}$ (b) $\rho_{u\xi}$ (c) $\rho_{u^2\xi}$

(d) $\rho_{u^2\xi^2}$ (e) $\rho_{\xi(\xi u)}$

Figure 36: Heater: residual analysis for the RBF.



(a) $\rho_{\xi\xi}$ (b) $\rho_{u\xi}$ (c) $\rho_{u^2\xi}$

(d) $\rho_{u^2\xi^2}$ (e) $\rho_{\xi(\xi u)}$

Figure 37: Heater: residual analysis for the RLM.

modeling of such systems, thus performing evaluation in terms of predictability using error measure over one-step-ahead predictions and free-run simulations of the underlying models.

For the first three plants, the MLM reported the best performance results. With regard to global models, the RBF and ELM reported poor results, which can just be improved via regularization and/or normalization steps. This is an advantage for the MLM in comparison to RBF and ELM, since applying regularization brings the need for optimizing an extra hyperparameter. The ARX/LS model also presented poor results, which could be expected from modeling nonlinear systems with linear structures. But, still, precautions may be taken when the analysis is made based on one-step-ahead predictions. Among the local models, the kSOM and LLM methods achieved the worst results, overall. Such a poor performance likely comes from the use of many local models in comparison to RLM, for instance. The best local approximation was achieved by the LLR methods, particularly, by the NN and NNi methods, which for system identification tasks reported rather similar performances. For the last evaluated plant (heater system), the performances obtained by the LLR methods, RLM, RBF, MLM, and ELM models were similar. The RLM slightly outperformed the MLM model.

Chapter 6

Conclusions and Future Directions

This thesis deals with the problem of nonlinear dynamic systems identification from a *machine learning* point of view. The machine learning approach to system identification is mainly based on the design of regression models that are virtually capable of describing any nonlinear dynamics. Such methods can be roughly categorized into global and local approaches. Global modeling consists of fitting a single regression model to the available data, whereas in local modeling the data are segmented into small partitions and a specialized regression model is fit to each partition. The objective of this thesis is to propose novel machine learning techniques to approach the nonlinear system identification problem from global and local perspectives.

The first contribution of the thesis is a novel global method, the *Minimal Learning Machine* (MLM). Learning in MLM consists in building a linear map between distances among points in the input and output spaces, and then estimating the response from the geometry of the outputs. The second contribution is a method that stands between global and local modeling, the *Regional Modeling* (RM) approach. In RM, we partition the input space using the *Self-Organizing Map* (SOM), and then we cluster the prototypes of the SOM. Regression models are then built over the SOM clusters, the regions.

A comprehensive performance evaluation of the MLM and RM has been carried out on synthetic and real-world datasets. The performances were compared to the results achieved by reference regression models. For the purpose the following methods have been selected: *Auto-Regressive with Exogenous Inputs* models (ARX); *Extreme Learning Machines* (ELM); *Redial Basis Function* networks (RBF); *Local Linear Maps* (LLM); *k*SOM; *Local Linear Regression* models (*k*NN, e*k*NN, NN and NN*i*). Based on the experimental results, both the MLM and RM are capable to achieve accuracies that are comparable to, and even better than, traditional machine learning methods.

In more details, in this thesis, we have:

- presented the nonlinear system identification problem in a concise way, while approaching the main issues related to the problem.
- overviewed global and local models commonly used for supervised learning and with direct application to system identification. In this regard, we mainly focused on the methods proposed in the machine learning field. This is in consonance with the proposal of the thesis of addressing the system identification problem from a learning perspective, in which all the information about the underlying systems comes from the available data.
- presented a supervised learning algorithm, Minimal Learning Machine (MLM), for solving regression, classification, and system identification problems. Learning in MLM consists in building a linear mapping between input and output distance matrices. In the generalization phase, the learned distance map is used to provide an estimate of the distance from M output reference points to the target output value. Then, the output point estimation is formulated as a multilateration problem based on the predicted output distance and the locations of the reference points. Given its general formulation, the Minimal Learning Machine is inherently capable of operating on nonlinear regression problems as well as on multidimensional response spaces.
- discussed issues related to the MLM algorithm, such as computational complexity, selection of hyperparameter, links with other reference methods, and selection of reference points. In addition, the MLM properties were illustrated for a regression and system identification synthetic example.
- evaluated the performance of the MLM for system identification through the modeling of four benchmarking problems. The performance of the MLM was compared against global, local and regional models. For three of the problems, the MLM was the best performing model in terms of its prediction capability. Moreover, model validation via residual analysis was carried out, in which the MLM passed the correlation tests, overall.
- carried out a comprehensive comparison between MLM and the state-of-the-art methods in machine learning (GP, SVM, ELM, RBF and MLP) in both classification and regression problems. The MLM achieved competitive performance, outperforming the reference models in many cases, and, thus, representing a valid and fast alternative to the traditional methods.
- evaluated, for the first time in the literature, enclosing neighborhood LLR methods on system identification tasks. We have shown that Natural Neighbors and Natural Neighbors Inclusive present rather similar performances, and the enclosing ekNN method is the most erratic method.

- introduced a new approach for system identification, called Regional Models (RM). The method proposal was motivated by the principle of parsimony. The proposed methodology led to the design of regional models, which stands in between the global and local models. Regional modeling extends the two-level clustering approach by Vesanto and Alhoniemi (2000) to regression problems, particularly, to system identification. The first step in regional modeling requires the partition of the input space using the SOM, and then clustering over the prototypes of the trained SOM. In the second step, regional regression models are built over the clusters (i.e. over the regions) of SOM prototypes. It turns out that regional modeling provides less complex and fast alternatives to traditional local modeling methods.

- discussed an outlier robust extension of Regional Models through the use of M-estimation theory in the estimation of the parameters of (regional) linear models.

- compared the performance of the Regional Linear Models (RLM) in the task of dynamic system identification against global and local models. In general, the regional models were outperformed by the MLM, but in comparison to the other methods, the performance is rather similar to the LLR methods (NN and NNi), and better than the LLM and kSOM models. An important issue is that we have tested the RLM without hyperparameter optimization. Rather, we provided results with the standard configuration from the SOM toolbox.

The major contributions of this thesis are two novel methods: the Minimal Learning Machine and Regional Models. We have focused on the description of a general framework and its basic formulation. As a consequence, a number of extensions and variants can be proposed for both methods. In what follows, we discuss the main directions on each proposal.

6.1 Future Directions on the Minimal Learning Machine

The first aspect that can be explored in the MLM algorithm is the *selection of reference points*. We have shown (in Appendices B and C) that the optimal number of reference points does not grow at the same rate as the number of learning points. In those cases, a random selection may provide a large variance in the results. Even though we have already discussed some options in Section 3.5, a comprehensive evaluation of selection approaches is still necessary. In this regard, we highlight the use of supervised pruning strategies for selecting the reference points.

Outlier robust extension for the MLM is another important aspect of the proposed method that should be further investigated. Since the MLM training step encompasses a linear model between distances, the application of M-estimation would be straightforward.

Also, robust estimation can be applied in the multilateration step. Still on the multilateration step, fast procedures to provide outputs estimates are desired. This includes the search for fast optimization methods or different strategies to locate the output estimate.

An important asset of the MLM is that it has only one hyperparameter to be optimized, the number of reference points. However, we can assume a different *distance metric* than the Euclidean one used throughout this thesis. How the MLM behaves when we change its metric is an interesting topic that has not be answered yet.

Since the Minimal Learning Machine is a general supervised learning method, one may consider to apply most of the variants already proposed for standard machine learning methods. For example, regularization strategies can be adopted for the MLM, even though we have not found serious problems in what concerns ill-conditioned matrices.

With regard to the application of the MLM to system identification, two main issues arise: *i*) *online system identification* (parameter estimation); *ii*) performance *evaluation of MIMO systems*. The first can be achieved by combining recursive (sequential) and/or growing selection of reference points, and sequential estimation of the linear model between distances. For the second issue, the MLM is able to handle multiresponse cases by definition. Thus, its use in the modeling of MIMO system is direct, but still has not been evaluated.

6.2 Future Directions on Regional Modeling

The proposed RM models allow a number of different algorithmic options into its steps. For instance, we have applied the k-means algorithm as the clustering method. However, the k-means method suffers from a high variance estimate, i.e., the method is unstable. Thus, the impact of clustering techniques into regional models is an open issue. In this regard, we are particularly curious about the use of the *correlation-based clustering* method (FUJIWARA; KANO; HASEBE, 2010), since the regression vectors in system identification tasks present a correlated nature. Still, in the clustering procedure, different cluster validity indices can be explored.

Another topic is online system identification with regional models, which has not been explored yet. With respect to that, growing and hierarchical self-organizing maps can be used in combination with the fitting of adaptive (sequential) linear models, such as the LMS algorithm. As well, the evaluation of regional models on the modeling of MIMO systems is important, even though the application would be straightforward.

While the Regional Models have been evaluated in the presence of outliers (SOUZA JUNIOR; BARRETO; CORONA, 2015), our formulation handles only outliers in the output space. However, robust strategies can also be employed during the SOM training. To achieve this, robust self-organizing maps have been proposed (MORENO et al., 2004),

and thus can be used in the regional modeling framework.

Finally, an extension of the regional models to classification problems is a prominent research topic. In principle, there is no impediment for that, since the models used in each region can be as general as any learning machine. Similarly, classification using local linear models has already been proposed in the literature (ALPAYDIN; JORDAN, 1996).

Bibliography

ABONYI, J. et al. Process analysis and product quality estimation by self-organizing maps with an application to polyethylene production. *Computers in Industry*, v. 52, n. 3, p. 221–234, 2003. Cited on page 68.

AGUIRRE, L. A. *Introdução à Identificação de Sistemas – Técnicas Lineares e Não-Lineares Aplicadas a Sistemas Reais*. 3rd. ed. [S.l.]: Editora UFMG, 2007. Cited 5 times on pages 17, 18, 20, 29, and 82.

AGUIRRE, L. A.; BILLINGS, S. Dynamical effects of overparametrization in nonlinear models. *Physics Letters D: Nonlinear Phenomena*, v. 80, p. 26–49, 1995. Cited on page 21.

AGUIRRE, L. A.; LETELLIER, C. Modeling nonlinear dynamics and chaos: A review. *Mathematical Problems in Engineering*, v. 2009, n. 1, 2009. Cited on page 22.

AKAIKE, H. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, v. 19, n. 6, p. 716–723, 1974. Cited on page 19.

ALPAYDIN, E.; JORDAN, M. I. Local linear perceptrons for classification. *IEEE Transactions on Neural Networks*, v. 7, p. 788–792, 1996. Cited on page 92.

ALVES, G. B.; CORREA, M. V.; AGUIRRE, L. A. Steady-state performance constraints for dynamical models based on radial basis function networks. *Engineering Applications of Artificial Intelligence*, v. 7, n. 20, p. 924–935, 2007. Cited on page 33.

ATKESON, C. G.; MOORE, A. W.; SCHAAL, S. Locally weighted learning. *Artificial Intelligence Review*, v. 3, n. 1-5, p. 11–73, 1997. Cited on page 34.

BARRETO, G. A.; ARAÚJO, A. F. R. Identification and control of dynamical systems using the self-organizing map. *IEEE Transactions on Neural Networks*, v. 15, n. 5, p. 1244–1259, 2004. Cited on page 37.

BARROS, A. L. B. P. *Revisitando o problema de classificação de padrões na presença de outliers usando técnicas de regressão robusta*. Tese (Doutorado) — Universidade Federal do Ceará, 2013. Cited on page 68.

BENGIO, Y. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, v. 2, n. 1, p. 1–127, 2009. Cited on page 32.

BEZDEK, J. C. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics Part B*, v. 28, n. 3, p. 301–315, 1998. Cited on page 104.

BHAT, N.; MCAVOY, T. J. Use of neural nets for dynamic modeling and control of chemical process systems. *Computers and Chemical Engineering*, v. 14, n. 4, p. 573–583, 1990. Cited on page 76.

BILLINGS, S. A. *Nonlinear System Identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. [S.l.]: John Wiley & Sons, 2013. Cited 2 times on pages 15 and 20.

BILLINGS, S. A.; CHEN, S.; KORENBERG, M. J. Identification of MIMO nonlinear systems using a forward-regression orthogonal estimator. *International Journal of Control*, v. 49, n. 6, p. 2157–2189, 1989. Cited 2 times on pages 16 and 27.

BILLINGS, S. A.; FAKHOURI, S. Y. Identification of systems containing linear dynamic and static nonlinear elements. *Automatica*, v. 1, n. 18, p. 15–26, 1982. Cited on page 27.

BILLINGS, S. A.; VOON, W. S. F. Correlation based model validity tests for nonlinear models. *International Journal of Control*, v. 44, n. 1, p. 235–244, 1986. Cited 2 times on pages 21 and 71.

BILLINGS, S. A.; ZHU, Q. M. Rational model identification using an extended least-squares algorithm. *International Journal of Control*, v. 54, n. 3, p. 529–546, 1991. Cited on page 27.

BISHOP, C. M. *Neural Networks for Pattern Recognition*. New York, NY: Oxford University Press, Inc., 1995. Cited 4 times on pages 28, 32, 108, and 111.

BISHOP, C. M. *Pattern Recognition and Machine Learning*. [S.l.]: Springer, 2006. Cited 2 times on pages 30 and 51.

BITTANTI, S.; PIRODDI, L. Nonlinear identification and control of a heat exchanger: A neural network approach. *Journal of Franklin Institute*, v. 334B, n. 1, p. 135–153, 1996. Cited on page 18.

BONTEMPI, G.; BIRATTARI, M.; BERSINI, H. Recursive lazy learning for modeling and control. In: *Proceedings of the 10th European Conference on Machine Learning*. [S.l.: s.n.], 1998. v. 1398, p. 292–303. Cited on page 59.

BONTEMPI, G.; BIRATTARI, M.; BERSINI, H. Lazy learning for modeling and control design. *International Journal of Control*, v. 72, n. 7/8, p. 643–658, 1999. Cited on page 34.

BUHMANN, M. D. *Radial Basis Functions*. [S.l.]: Cambridge University Press, 2003. Cited 3 times on pages 33, 108, and 111.

BUNKE, H. Structural and syntactic pattern recognition. In: *Handbook of Pattern Recognition & Computer Vision*. River Edge, NJ, USA: World Scientific, 1993. p. 163–209. Cited on page 43.

CAMPELLO, R. J. G. B.; FAVIER, G.; AMARAL, W. C. Optimal expansions of discrete-time Volterra models using Laguerre functions. *Automatica*, v. 40, n. 5, p. 815–822, 2004. Cited on page 27.

CAO, L. et al. Predicting chaotic time series with wavelet networks. *Physica D*, v. 1, n. 1, p. 225–238, 1995. Cited on page 27.

CHANG, C.-C.; LIN, C.-J. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, v. 2, n. 27, p. 1–27, 2011. Cited 2 times on pages 108 and 112.

CHEN, S.; COWAN, C. F.; GRANT, P. M. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, v. 2, n. 2, p. 302–309, 1991. Cited 3 times on pages 34, 53, and 58.

CHO, J. et al. Modeling and inverse controller design for an unmanned aerial vehicle based on the self-organizing map. *IEEE Transactions on Neural Networks*, v. 17, n. 2, p. 445–460, 2006. Cited on page 39.

CHO, J. et al. Quasi-sliding mode control strategy based on multiple linear models. *Neurocomputing*, v. 70, n. 4-6, p. 962–974, 2007. Cited on page 39.

CORMEN, T. H. et al. *Introduction to Algorithms*. 3rd. ed. [S.l.]: The MIT Press, 2009. Cited on page 49.

COURRIEU, P. Fast computation of Moore-Penrose inverse matrices. *Neural Information Processing Letters and Reviews*, v. 8, p. 25–29, 2005. Cited on page 49.

CUADRAS, C.; ARENAS, C. A distance based regression model for prediction with mixed data. *Communication in Statistics-Theory and Methods*, v. 19, n. 6, p. 2261–2279, 1990. Cited on page 43.

DAVIES, D. L.; BOULDIN, D. W. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Ingelligence*, v. 1, n. 2, p. 224–227, 1979. Cited 2 times on pages 61 and 104.

DENG, W.; ZHENG, Q.; CHEN, L. Regularized extreme learning machine. In: *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining*. [S.l.: s.n.], 2009. p. 389–395. Cited on page 81.

FERRARI-TRECATE, G. et al. A clustering technique for the identification of piecewise affine systems. *Automatica*, v. 39, n. 2, p. 205–217, 2003. Cited on page 68.

FOX, J.; WEISBERG, S. *An R Companion to Applied Regression*. 2nd. ed. [S.l.]: Sage Publications, 2010. Cited on page 67.

FUJIWARA, K.; KANO, M.; HASEBE, S. Development of correlation-based clustering method and its application to software sensing. *Chemometrics and Intelligent Laboratory Systems*, v. 101, n. 2, p. 130 – 138, 2010. Cited on page 91.

GISBRECHT, A. et al. Linear time relational prototype based learning. *International Journal of Neural Systems*, v. 22, n. 5, p. 1–11, 2012. Cited on page 44.

GOLUB, G. H.; LOAN, C. F. V. *Matrix Computations*. 3rd. ed. [S.l.]: Johns Hopkins University Press, 1996. Cited on page 49.

GRAEPEL, T. et al. Classification on pairwise proximity data. In: JORDAN, M. I.; KEARNS, M. J.; SOLLA, S. A. (Ed.). *NIPS Proceedings*. Cambridge, MA: MIT Press, 1999. v. 11, p. 438–444. Cited on page 44.

GRAEPEL, T.; OBERMAYER, K. A stochastic self-organizing map for proximity data. *Neural Computation*, v. 11, n. 1, p. 139–155, 1999. Cited on page 43.

GREGORCIC, G.; LIGHTBODY, G. Nonlinear system identification: from multiple-model networks to gaussian processes. *Engineering Applications of Artificial Intelligence*, v. 21, n. 7, p. 1035–1055, 2008. Cited 2 times on pages 16 and 22.

GUPTA, M. R.; GARCIA, E. K.; CHIN, E. Adaptive local linear regression with application to printer color management. *IEEE Transactions on Image Processing*, v. 17, p. 936–945, 2008. Cited 3 times on pages 34, 40, and 41.

HABER, R.; UNBEHAUEN, H. Structure identification of nonlinear dynamic systems: a survey on input/output approaches. *Automatica*, v. 26, n. 4, p. 651–677, 1990. Cited on page 16.

HAGENBUCHNER, M.; SPERDUTI, A.; TSOI, A. C. A self-organizing map for adaptive processing of structured data. *IEEE Transactions on Neural Networks*, v. 14, n. 3, p. 491–505, 2003. Cited on page 43.

HALKIDI, M.; BATISTAKIS, Y.; VAZIRGIANNIS, M. On clustering validation techniques. *Journal of Intelligent Information Systems*, v. 17, n. 2-3, p. 107–145, 2001. Cited on page 61.

HAMETNER, C.; JAKUBEK, S. Local model network identification for online engine modelling. *Information Sciences*, v. 220, p. 210–225, 2013. Cited on page 34.

HAMMER, B.; HASENFUSS, A. Topographic mapping of large dissimilarity data sets. *Neural Computation*, v. 22, n. 9, p. 2229–2284, 2010. Cited on page 43.

HAMMER, B. et al. Learning vector quantization for (dis-)similarities. *Neurocomputing*, v. 131, p. 43–51, 2014. Cited on page 44.

HAMMER, B. et al. A general framework for unsupervised processing of structured data. *Neurocomputing*, v. 57, p. 3–35, 2004. Cited on page 43.

HARDY, R. Multiquadric equations of topography and other irregular surfaces. *Journal of Geophysical Research*, v. 76, n. 8, 1971. Cited on page 50.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd. ed. [S.l.]: Springer, 2009. Cited 2 times on pages 19 and 48.

HAYKIN, S. *Adaptive filter theory*. 4th. ed. New Jersey, NJ: Prentice-Hall, 2001. Cited on page 47.

HAYKIN, S. *Neural Networks and Learning Machines*. 3rd. ed. [S.l.]: Prentice Hall, 2009. Cited 3 times on pages 30, 31, and 32.

HOFMANN, T.; BUHMANN, J. Pairwise data clustering by deterministic annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 19, n. 1, p. 1–14, 1997. Cited on page 43.

HONG, X. et al. Model selection approaches for non-linear system identification: a review. *International Journal of Systems Science*, v. 39, n. 10, p. 925–946, 2008. Cited on page 19.

HORATA, P.; CHIEWCHANWATTANA, S.; SUNAT, K. Robust extreme learning machine. *Neurocomputing*, v. 102, n. 1, p. 31–44, 2013. Cited on page 77.

HUANG, G.-B.; WANG, D. H.; LAN, Y. Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics*, v. 2, p. 107–122, 2011. Cited on page 32.

HUANG, G. B.; ZHU, Q. Y.; ZIEW, C. K. Extreme Learning Machine: Theory and applications. *Neurocomputing*, v. 70, n. 1–3, p. 489–501, 2006. Cited 4 times on pages 28, 32, 108, and 111.

HUBER, P. J. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, v. 35, n. 1, p. 73–101, 1964. Cited 2 times on pages 66 and 68.

HUBER, P. J. *Robust Statistics*. [S.l.]: John Wiley & Sons, 2004. Cited 2 times on pages 24 and 51.

JACOBS, R. A. et al. Adaptive mixtures of local experts. *Neural Computation*, v. 3, n. 1, p. 79–87, 1991. Cited on page 34.

JANG, J.-S. R. ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 23, p. 665–685, 1993. Cited on page 34.

KALHOR, A.; ARAABI, B. N.; LUCAS, C. Reducing the number of local linear models in neuro-fuzzy modeling: A split-and-merge clustering approach. *Applied Soft Computing*, v. 11, n. 1, p. 5582–5589, 2011. Cited on page 68.

KASHYAP, R. L. Bayesian comparison of different classes of dynamic models using empirical data. *IEEE Transactions on Automatic Control*, v. 22, n. 5, p. 715–727, 1977. Cited on page 19.

KATSIKIS, V. N.; PAPPAS, D.; PETRALIAS, A. An improved method for the computation of the Moore-Penrose inverse matrix. *Applied Mathematics and Computation*, v. 217, p. 9828–9834, 2011. Cited on page 49.

KAUFMAN, L.; ROUSSEEUW, P. J. *Finding groups in data: An introduction to cluster analysis*. [S.l.]: Wiley, 1990. Cited 2 times on pages 52 and 103.

KEESMAN, K. J. *System Identification: An Introduction*. [S.l.]: Springer, 2011. (Advanced textbooks in control and signal processing). Cited on page 15.

KOHONEN, T. Essentials of the self-organizing map. *Neural Networks*, v. 37, p. 52–65, 2013. Cited on page 35.

KOHONEN, T. K. et al. Engineering applications of the self-organizing map. *Proceedings of the IEEE*, v. 84, n. 10, p. 1358–1384, 1996. Cited on page 36.

LAWRENCE, S.; TSOI, A. C.; BACK, A. D. Function approximation with neural networks and local methods: Bias, variance and smoothness. In: *Australian Conference on Neural Networks (ACNN)*. [S.l.: s.n.], 1996. p. 16–21. Cited on page 27.

LI, X.; YU, W. Dynamic system identification via recurrent multilayer perceptrons. *Information Sciences*, v. 147, n. 1–4, p. 45–63, 2002. Cited on page 27.

LICHSTEIN, J. W. Multiple regression on distance matrices: a multivariate spatial analysis tool. *Plant Ecology*, v. 188, n. 2, p. 117–131, 2006. Cited on page 43.

LIMA, C. A. M.; COELHO, A. L. V.; VON ZUBEN, F. J. Hybridizing mixtures of experts with support vector machines: Investigation into nonlinear dynamic systems identification. *Information Sciences*, v. 177, n. 10, p. 2049–2074, 2007. Cited 2 times on pages 16 and 34.

LJUNG, L. *System Identification: Theory for the user*. 2nd. ed. Englewood Cliffs, NJ: Prentice-Hall, 1999. Cited on page 29.

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In: Le Cam, L. M.; NEYMAN, J. (Ed.). *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. Berkeley, Califonia. University of California Press: [s.n.], 1967. v. 1, p. 281–297. Cited 3 times on pages 34, 61, and 103.

MARQUARDT, D. W. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, v. 11, n. 2, p. 431–441, 1963. Cited on page 48.

MASSEY, F. J. The kolmogorov-smirnov test for goodness of fit. *Neural Computation*, v. 46, n. 253, p. 68–78, 1951. Cited on page 53.

McArdle, B.; ANDERSON, M. Fitting multivariate models to community data: a comment on distance-based redundancy analysis. *Ecology*, v. 82, n. 1, p. 290–297, 2001. Cited on page 43.

MCAVOY, T. J.; HSU, E.; LOWENTHAL, S. Dynamics of pH in controlled stirred tank reactor. *Industrial & Engineering Chemistry Process Design and Development*, v. 11, n. 1, p. 68–70, 1972. Cited on page 76.

MICHE, Y. et al. OP-ELM: Optimally Pruned Extreme Learning Machine. *IEEE Transactions on Neural Networks*, v. 21, n. 1, p. 158–162, 2010. Cited 2 times on pages 49 and 58.

MORENO, S. et al. Robust self organizing maps. In: *CIARP*. [S.l.]: Springer, 2004. (Lecture Notes in Computer Science, v. 3287), p. 179–186. Cited on page 91.

MURRAY-SMITH, R. *A local model network approach to nonlinear modelling*. Tese (Doutorado) — University of Strathclyde, 1994. Cited on page 34.

NARENDRA, K. S. Neural networks for control theory and practice. *Proceedings of the IEEE*, v. 84, n. 10, p. 1385–1406, 1996. Cited on page 27.

NARENDRA, K. S.; PARTHASARATHY, K. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, v. 1, n. 1, p. 4–27, 1990. Cited 5 times on pages 16, 20, 27, 30, and 71.

NELLES, O. *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. [S.l.]: Springer, 2001. Cited 7 times on pages 18, 30, 31, 32, 34, 51, and 53.

NEUMANN, K.; STEIL, J. J. Optimizing extreme learning machines via ridge regression and batch intrinsic plasticity. *Neurocomputing*, v. 102, n. 15, p. 23 – 30, 2013. Cited on page 77.

NIEWIADOMSKA-SZYNKIEWICZ, E.; MARKS, M. Optimization schemes for wireless sensor network localization. *International Journal of Applied Mathematics and Computer Science*, v. 19, n. 2, p. 291–302, 2009. Cited on page 47.

NORGAARD, M. et al. *Neural Networks for Modelling and Control of Dynamic Systems*. [S.l.]: Springer-Verlag, 2000. Cited 3 times on pages 18, 20, and 30.

PAPADAKIS, S. E.; KABURLASOS, V. G. Piecewise-linear approximation of non-linear models based on probabilistically/possibilistically interpreted intervals' numbers (INs). *Information Sciences*, v. 180, n. 24, p. 5060–5076, 2010. Cited on page 16.

POGGIO, T.; GIROSI, F. Networks for approximation and learning. *Proceedings of the IEEE*, v. 78, n. 9, p. 1481–1497, 1990. Cited on page 28.

PRINCIPE, J. C. *Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives*. [S.l.]: Springer, 2010. Cited on page 28.

PRINCIPE, J. C.; WANG, L.; MOTTER, M. A. Local dynamic modeling with self-organizing maps and applications to nonlinear system identification and control. *Proceedings of the IEEE*, v. 86, n. 11, p. 2240–2258, 1998. Cited on page 39.

QIN, A. K.; SUGANTHAN, P. N. A novel kernel prototype-based learning algorithm. In: *Proceedings of the 17th International Conference on Pattern Recognition (ICPR'2004)*. [S.l.: s.n.], 2004. v. 4, p. 621–624. Cited on page 44.

RAO, C. R.; TOUTENBURG, H. *Linear Models: Least Squares and Alternatives*. 2nd. ed. [S.l.]: Springer, 1999. Cited 2 times on pages 20 and 28.

RASMUSSEN, C. E.; WILLIAMS, C. K. I. *Gaussian Processes for Machine Learning*. Cambridge, MA: The MIT Press, 2006. Cited 4 times on pages 28, 108, 111, and 112.

REZAEE, B.; FAZEL ZARANDI, M. H. Data-driven fuzzy modeling for Takagi–Sugeno–Kang fuzzy system. *Information Sciences*, v. 180, n. 2, p. 241–255, 2010. Cited on page 16.

ROJO-ALVAREZ, J. L. et al. Support vector method for robust ARMA system identification. *IEEE Transactions on Signal Processing*, v. 52, n. 1, p. 155–164, 2004. Cited on page 16.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by backpropagating errors. *Nature*, v. 323, n. 9, p. 533–536, 1986. Cited on page 28.

RUMELHART, D. E.; HINTONT, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, v. 323, n. 6088, p. 533–536, 1986. Cited on page 31.

SARRA, S. A.; KANSA, E. J. Multiquadric radial basis function approximation methods for the numerical solution of partial differential equations. *Advances in Computational Mechanics*, v. 2, 2009. Cited on page 34.

SCHALKOFF, R. J. *Pattern Recognition: Statistical, Structural and Neural Approaches*. [S.l.]: John Wiley & Sons, 1992. Cited on page 43.

SCHLEIF, F.-M. et al. Efficient kernelized prototype based classification. *International Journal of Neural Systems*, v. 21, n. 6, p. 443–457, 2012. Cited on page 44.

SJÖBERG, J. et al. Non-linear black-box modeling in system identification: a unified overview. *Automatica*, v. 31, p. 1691–1724, 1995. Cited 3 times on pages 15, 16, and 80.

SMOLA, A. J.; SCHOLKOPF, B. A tutorial on support vector regression. *Statistics and Computing*, v. 14, n. 3, p. 199–222, 2004. Cited 3 times on pages 28, 108, and 111.

SOHLBERG, B. Grey box modelling for model predictive control of a heating process. *Journal of Process Control*, v. 13, n. 3, p. 225 – 238, 2003. Cited on page 16.

SOMERVUO, P.; KOHONEN, T. Self-organizing maps and learning vector quantization for feature sequences. *Neural Processing Letters*, v. 10, n. 2, p. 151–159, 1999. Cited on page 43.

SOUZA JUNIOR, A. H.; BARRETO, G. A.; CORONA, F. Regional models: A new approach for nonlinear system identification via clustering of the self-organizing map. *Neurocomputing*, v. 147, n. 5, p. 31 – 46, 2015. Cited 2 times on pages 61 and 91.

SOUZA JUNIOR, A. H. et al. Minimal learning machine: A new distance-based method for supervised learning. In: *Proceedings of the 12th International Work Conference on Artificial Neural Networks (IWANN'2013)*. [S.l.]: Springer, 2013. (Lecture Notes in Computer Science, v. 7902), p. 408–416. Cited on page 45.

SOUZA JUNIOR, A. H. et al. Extending the minimal learning machine for pattern classification. In: *Proceedings of the 1st BRICS countries conference on computational intelligence*. [S.l.: s.n.], 2013. v. 1, p. 1–8. Cited on page 44.

SOUZA, L. G. M.; BARRETO, G. A. On building local models for inverse system identification with vector quantization algorithms. *Neurocomputing*, v. 73, n. 10-12, p. 1993–2005, 2010. Cited 2 times on pages 34 and 38.

TAKAGI, T.; SUGENO, M. Fuzzy identification of systems and its application to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, v. 15, n. 1, p. 116–132, 1985. Cited 2 times on pages 16 and 34.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. *Introduction to Data Mining*. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 2005. Cited 2 times on pages 102 and 104.

VAN HULLE, M. Self-organizing maps. In: ROZENBERG, G.; BAECK, T.; KOK, J. (Ed.). *Handbook of Natural Computing: Theory, Experiments, and Applications*. [S.l.]: Springer-Verlag, 2010. p. 1–45. Cited on page 36.

VAPNIK, V. N. *Statistical learning Theory*. [S.l.]: Wiley-Interscience, 1998. Cited on page 28.

VERDULT, V. *Nonlinear system identification: A state-space approach*. Tese (Doutorado) — University of Twente, 2002. Cited on page 72.

VESANTO, J.; ALHONIEMI, E. Clustering of the self-organizing map. *IEEE Transactions on Neural Networks*, v. 11, n. 3, p. 586–600, 2000. Cited 5 times on pages 60, 61, 62, 69, and 90.

VESANTO, J. et al. Self-organizing map in Matlab: the SOM toolbox. In: *Proceedings of the Matlab DSP Conference*. [S.l.: s.n.], 2000. p. 35–40. Cited 2 times on pages 64 and 71.

WALTER, J.; RITTER, H.; SCHULTEN, K. Non-linear prediction with self-organizing map. In: *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'90)*. [S.l.: s.n.], 1990. v. 1, p. 587–592. Cited 2 times on pages 34 and 36.

WANG, J. Classical multidimensional scaling. In: *Geometric Structure of High-Dimensional Data and Dimensionality Reduction*. [S.l.]: Springer, 2011. p. 115–129. Cited on page 43.

WANG, X.; SYRMOS, V. L. Nonlinear system identification and fault detection using hierarchical clustering analysis and local linear models. In: *Proceedings of the 15th Mediterranean Conference on Control & Automation (MED)'07*. [S.l.: s.n.], 2007. p. T23–015. Cited 2 times on pages 60 and 68.

WESTON, J. et al. Kernel dependency estimation. In: BECKER, S.; THRUN, S.; OBERMAYER, K. (Ed.). *NIPS Proceedings*. Cambridge, MA: MIT Press, 2002. p. 873–880. Cited on page 51.

WIDROW, B. Thinking about thinking: The discovery of the LMS algorithm. *IEEE Signal Processing Magazine*, v. 22, n. 1, p. 100–106, 2005. Cited 2 times on pages 36 and 47.

WILCOXON, F. Individual comparisons by ranking methods. *Biometrics*, v. 1, n. 6, p. 80–83, 1945. Cited 2 times on pages 109 and 112.

WU, Y. et al. Using radial basis function networks for function approximation and classification. *ISRN Applied Mathematics*, v. 2012, n. ID324194, p. 1–34, 2012. Cited 3 times on pages 28, 108, and 111.

YIN, H. The self-organizing maps: Background, theories, extensions and applications. In: FULCHER, J.; JAIN, L. C. (Ed.). *Computational Intelligence: A Compendium*. [S.l.]: Springer-Verlag, 2008, (Studies in Computational Intelligence, v. 115). p. 715–762. Cited on page 36.

YU, W. Nonlinear system identification using discrete-time recurrent neural networks with stable learning algorithms. *Information Sciences*, v. 158, p. 131–157, 2004. Cited on page 27.

ZHU, X.; SCHLEIF, F.-M.; HAMMER, B. Adaptive conformal semi-supervised vector quantization for dissimilarity data. *Pattern Recognition Letters*, v. 49, p. 138–145, 2014. Cited on page 44.

ZHU, Z. et al. Local linear regression for soft-sensor design with application to an industrial deethanizer. In: *Proceedings of the 18th World Congress of the International Federation of Automatic Control*. [S.l.: s.n.], 2011. p. 2839–2844. Cited on page 40.

APPENDIX A

Clustering algorithms

Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups (clusters). In this regard, the similarity criterion takes a important role. Another important issue related to clustering method is the number of clusters. For selecting the optimal number of clusters, usually, we adopt a cluster validity index. Clustering techniques have been subject of interest and has long played an important role in a wide variety of fields: psychology and other social sciences, biology, statistics, pattern recognition, information retrieval, machine learning, and data mining.

Different taxonomies for clustering strategies exist. For instance, we cite the one proposed by Tan, Steinbach and Kumar (2005) that categorizes the clustering algorithms into: *i*) partitional versus hierarchical, *ii*) exclusive versus overlapping versus fuzzy, *iii*) complete versus partial. In this thesis, we are particularly interested in partitional algorithms due to their wide usage and simplicity. Partitional clustering algorithms divide a data set into a number of, usually non-overlapping, clusters.

In what follows, we describe two clustering techniques used in the thesis: k-means algorithms (Section A.1) and k-medoids (Section A.2). In addition, we introduce the Davies-Bouldies (DB) index for selecting the optimal number of clusters in Section A.3.

A.1 k-means algorithm

One type of partitional methods is centroid-based clustering, in which clusters are represented by a central vector, which may not necessarily be a member of the data set. When the number of clusters is fixed to K, k-means clustering gives a formal definition as an optimization problem: find the K cluster centers and assign the objects to the nearest cluster center, such that the squared distances from the cluster are minimized.

The optimization problem itself is known to be NP-hard, and thus the common approach is to search only for approximate solutions. A particularly well known approximative method is the k-means algorithm (MACQUEEN, 1967). It does however only find a local optimum, and is commonly run multiple times with different random initializations. Most k-means-type algorithms require the number of clusters - K - to be specified in advance, which is considered to be one of the biggest drawbacks of these algorithms. Furthermore, the algorithms prefer clusters of approximately similar size, as they will always assign an object to the nearest centroid. Still, the k-means algorithm is one of the most common and simple clustering methods.

The k-means algorithm consists of:

1. Choose initial values for the K cluster centers $\mathbf{p}_k$, $k = 1, \ldots, K$. This can be accomplished by picking randomly K data points from the available training samples.
2. Assign all data points to their nearest cluster prototype.
3. Compute the centroid prototype of each cluster. Set each cluster prototype to the centroid of its cluster, that is,

$$\mathbf{p}_k = \frac{\sum_{k \in \mathcal{S}_k} \mathbf{x}(k)}{N_k} \tag{A.1}$$

   where $\mathcal{S}_k$ denote the set the data points indices whose the nearest prototype is $\mathbf{p}_k$, N_k is the number of elements in the set $\mathcal{S}_k$.
4. If any cluster prototype has been moved in the previous step, go to Step 2; otherwise the clustering process is done.

A.2 k-medoids algorithm

The k-medoids algorithm is a partitional clustering technique that attempt to minimize squared error, the distance between points labeled to be in a cluster and a point designated as the center of that cluster. In contrast to the k-means algorithm, k-medoids chooses data points as centers (medoids) and is usually referred as more robust to noise and outliers in comparison to k-means. A medoid is the most centrally located object in a cluster.

The most common implementation of k-medoid clustering is the *partitioning around medoids* (PAM) algorithm (KAUFMAN; ROUSSEEUW, 1990):

1. **Initialization**: randomly select K of the N data points as the medoids.
2. **Assignment step**: Associate each data point to the closest medoid.

3. **Update step**: For each medoid k and each data point n associated to k, swap k and n and compute the total cost of the configuration (that is, the average dissimilarity of n to all the data points associated to k). Select the medoid n with the lowest cost of the configuration.

4. Repeat alternating steps 2 and 3 until there is no change in the assignments.

A.3 Cluster validity indexes

There are two main types of clustering validation: internal and external ones. Internal validation occurs when a clustering result is evaluated based on the data that was clustered itself. These methods usually assign the best score to the algorithm that produces clusters with high similarity within a cluster and low similarity between clusters. In external validation, clustering results are evaluated based on data that was not used for clustering, such as known class labels and external benchmarks (a set of pre-classified items). The interested reader may see the text-book by Tan, Steinbach and Kumar (2005) and Bezdek (1998) for a review of cluster validity indexes. In the experiments of this thesis, we have adopted the Davies-Bouldies (DB) validity index to select the optimal number of clusters.

A.3.1 Davies-Bouldin index

The Davies–Bouldin index (DAVIES; BOULDIN, 1979) can be calculated by the following formula:

$$DB = \frac{1}{K} \sum_{k=1}^{K} \max_{k \neq j} \left(\frac{\sigma_k + \sigma_j}{d(\mathbf{p}_k, \mathbf{p}_j)} \right)$$

where K is the number of clusters, $\mathbf{p}_x$ is the centroid of cluster x, σ_x is the average distance of all elements in cluster x to centroid $\mathbf{p}_x$, and $d(\mathbf{p}_k, \mathbf{p}_j)$ is the distance between centroids $\mathbf{p}_k$ and $\mathbf{p}_j$. Since algorithms that produce clusters with low intra-cluster distances (high intra-cluster similarity) and high inter-cluster distances (low inter-cluster similarity) will have a low DB index, the clustering algorithm that produces a collection of clusters with the smallest DB index is considered the best algorithm based on this criterion.

APPENDIX B

Minimal Learning Machine for Classification

In this appendix, we formulate the Minimal Learning Machine for classification problems (Section B.1); we illustrate its characteristics on a synthetic classification example (Section B.2); and we report the performance results achieved on four benchmarking classification problems (Section B.3).

B.1 Formulation

An important class of problems is classification, where we are concerned with predicting categories usually denoted by qualitative outputs, also called class labels. For the task, we are still given a set of N input points $X = \{\mathbf{x}_i\}_{i=1}^{N}$, with $\mathbf{x}_i \in \mathbb{R}^D$, and the set of their corresponding class labels $L = \{l_i\}_{i=1}^{N}$, with $l_i \in \{C_1, \ldots, C_S\}$, where C_j denotes the j-the class; for $S = 2$, the problem is referred to as binary classification, whereas for $S > 2$ we have multi-class applications.

The Minimal Learning Machine can be extended to classification problems in a straightforward manner by representing the S class labels in a vectorial fashion through an 1-of-S encoding scheme. In such approach, a S-level qualitative variable is represented by a vector of S binary variables or bits, only one of which is *on* at a time. Mathematically, the set of outputs $Y = \{\mathbf{y}_i\}_{i=1}^{N}$, with $\mathbf{y}_i \in \mathbb{R}^S$, that corresponds to the input points X is then defined in such a way that the jth-component of $\mathbf{y}_i$ is set to α if $l_i = C_j$ and β otherwise, where α and β are integer scalars such as $\alpha > \beta$. An usual choice is $\alpha = 1$ and $\beta = -1$.

In classification of a test observation $\mathbf{x}$ with unknown class label $l \in \{C_1, \ldots, C_S\}$, the estimated class $\hat{l}$ associated to the output estimate $\hat{\mathbf{y}}$ is given by $\hat{l} = C_{s^*}$, where

$$s^* = \underset{s=1,\ldots,S.}{\operatorname{argmax}} \{\hat{y}^{(s)}\}. \tag{B.1}$$

As one can easily notice, for binary classification problems, we may simplify the approach by using a binary single output scheme where the outputs are represented by scalars $y_i \in \{\alpha, \beta\}$ in correspondence to the two classes.

Given this formulation, the Minimal Learning Machine provides unified implementation for regression, binary and multi-class applications.

B.2 Illustrative example

We illustrate the Minimal Learning Machine on a binary classification problem, the Tai Chi symbol, where the Yin and Yang regions are the two non-convex and nonlinearly separable classes. For the task, we generated 2^{13} bidimensional input points uniformly distributed in the Tai Chi symbol, and after assigning the class labels to the Yin an Yang regions we purposely mislabeled 10% of the observations, Figure 38. Half of the dataset is used for training the Minimal Learning Machines with a varying number N of learning points and a number M of randomly selected reference points, again with $M \leq N$. The 2^{12} remaining samples are used for validation.

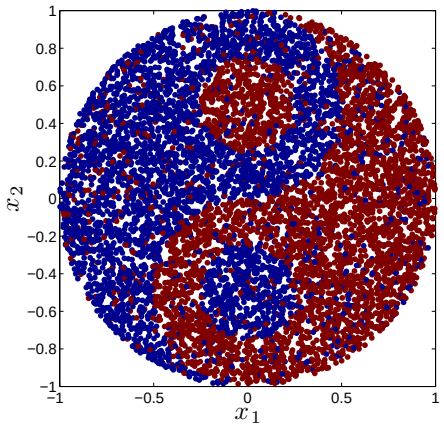
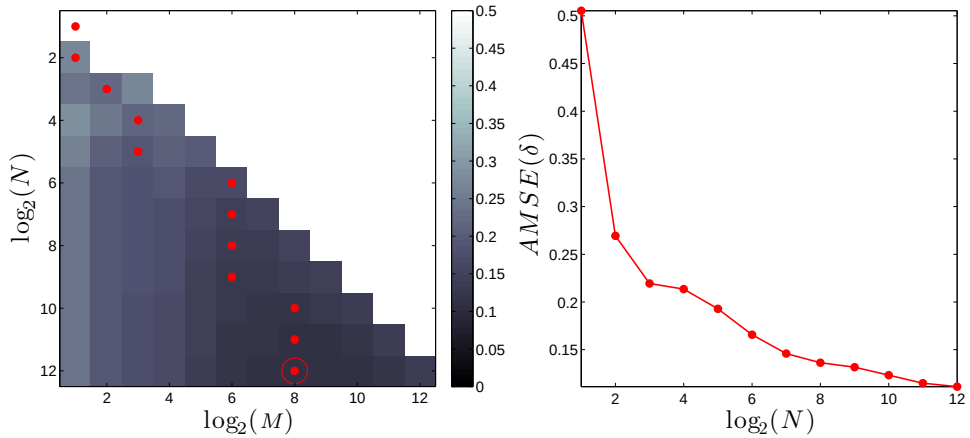


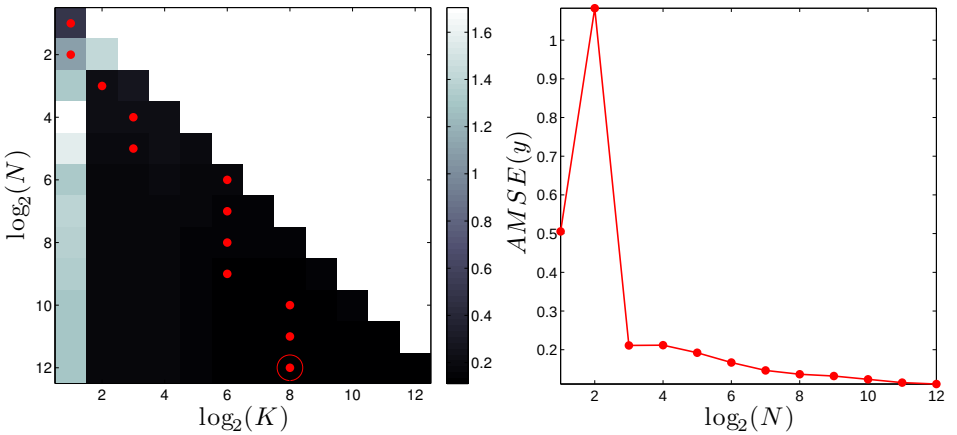
Figure 38: The Tai Chi symbol: Data.

We evaluated the performance of the Minimal Learning Machine on the validation set using the $AMSE(\boldsymbol{\delta})$ and the $AMSE(\mathbf{y})$, Figure 39. Figure 39a and 39c depict with red dots the configuration of the best MLMs for given sizes of the learning set and the circle is used to depict the best model overall. As expected, for each size N of the learning set it is again possible to select an optimal number M of reference points that minimizes the validation error, from the point of view of both the distance regression and the output estimation step. The overall best configuration of the MLM is found to be the one that is based on the largest number of learning point ($N = 2^{12}$) and a number of reference points equal to $M = 2^8$.

With respect to the estimation step, Figure 40 shows the estimated classes in validation using the overall best MLM configuration. The classification accuracy achieved



(a) $AMSE(\boldsymbol{\delta})$ (b) Optimal $AMSE(\boldsymbol{\delta})$



(c) $AMSE(\boldsymbol{y})$ (d) Optimal $AMSE(\boldsymbol{y})$

Figure 39: The Tai Chi symbol: Figures of merit.

by this MLM is equal to 88%, which tends to the percentage of purposely mislabeled data.

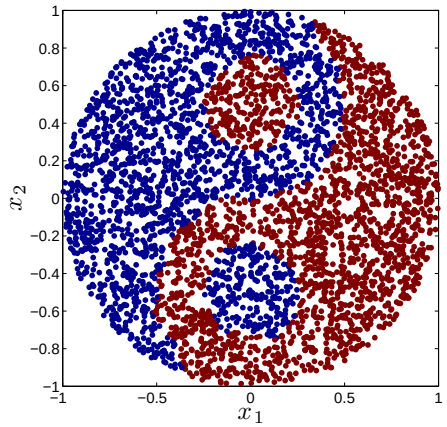


Figure 40: The Tai Chi symbol: Output estimation, with $N = 2^{12}$ and $M = 2^8$.

B.3 Experiments

In this section, we present the results achieved by the Minimal Learning Machine on four real-world datasets commonly used for benchmarking purposes in classification. The performance of the MLM is then compared to what is achieved by five other reference methods: The Extreme Learning Machine (ELM) (HUANG; ZHU; ZIEW, 2006), the Radial Basis Function Network (RBF) (BUHMANN, 2003; WU et al., 2012), the Support Vector Machine (SVM) (SMOLA; SCHOLKOPF, 2004), Gaussian Processes (GP) (RASMUSSEN; WILLIAMS, 2006), and the MultiLayer Perceptron (MLP) (BISHOP, 1995).

Table 5: Description of the datasets: input/output dimensionality and number of training/test samples.

Dataset	Dim		# Samples	
	In	Out	Train	Test
Wisconsin Breast Cancer	30	2	379	190
Pima Indians Diabetes	8	2	512	256
Iris	4	3	100	50
Wine	13	3	118	60

The datasets are available from the University of California at Irvine (UCI) Repository (`www.ics.uci.edu/~mlearn/`) and a description of the datasets is summarized in Table 5. The datasets have been chosen to object heterogeneity in the number of samples and inputs. All the datasets have been preprocessed in the same way. Categorical variables have been removed as well as samples containing missing values. Ten different random permutations of the whole datasets are taken, and two thirds are used to create the training set and the remaining for the test set. Then, the training set is normalized to zero mean and unit variance, and the test set is normalized using the same mean and variance from the training set. It may also be noticed that the proportions of the classes, for classification cases, have been kept balanced: each class is represented in an equal proportion, in both training and test sets.

The hyper-parameters for the SVM and the MLP are selected using 10-fold cross-validation. The SVM is learned using the SVM toolbox (CHANG; LIN, 2011) with default settings for the hyper-parameters and grid search: the grid is logarithmic between 2^{-2} and 2^{10} for each hyper-parameter; nu-SVC has been used for classification and epsilon-SVR for regression, with radial basis function kernel. The MLP is trained using Levenberg-Marquardt optimization and a range of hidden units from 1 to 20. The learning of GP is based on the default settings in the Matlab Toolbox (RASMUSSEN; WILLIAMS, 2006). The ELM network uses sigmoid kernel and it has been validated with the number of hidden units ranging from 10 to 100 with increments of 10. For the experiments with the RBF network, the centers of the gaussian basis functions are selected by the k-means algorithm, with the number of centers varying from 5% to 50% (step size of 5%) of the number of

learning points. We also applied 10-fold cross-validation to select the optimal number of centers. The only hyper-parameter of the Minimal Learning Machine, the number of reference points M, has been optimized using 10-fold cross-validation with reference points randomly selected in a range of 5% to 100% (with a step size of 5%) of the available training samples.

B.3.1 Results

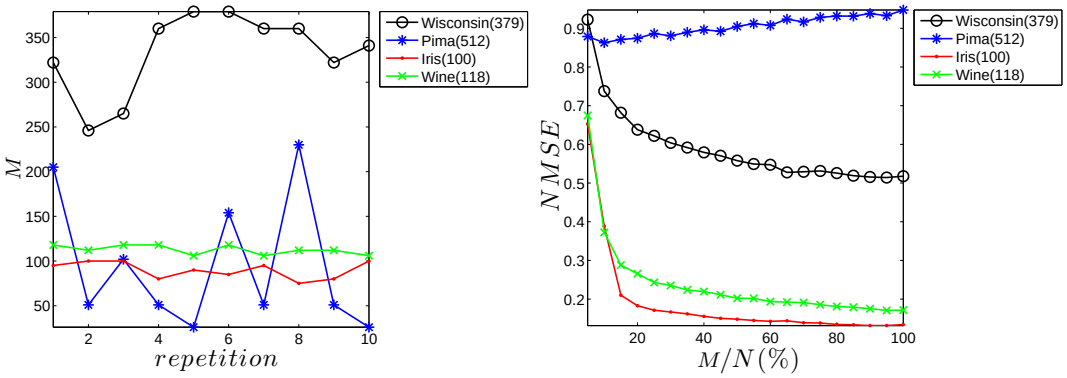
In order to evaluate the MLM performance for classification problems, we use the mean success classification rate and the corresponding standard deviations. To assess statistical significance for the results achieved by the MLM in comparison to the other methods, we carried out the nonparametric Wilcoxon signed-ranks hypothesis test (WILCOXON, 1945) with significance level equal to 5%. The null hypothesis is that the difference between MSE values comes from a distribution with zero median. The results are reported in Table 6.

Table 6: Test performance: accuracies (%), the corresponding standard deviations and Wilcoxon signed-ranks test results (✓: fail to reject, ×: reject). For each dataset, the best performing models are in boldface.

Datasets	**Models**					
	MLM	ELM	RBF	SVM	GP	MLP
Wisconsin B. C.	**97.7** **0.6** 	95.7 1.2 ×	95.6 1.4 ×	91.6 1.7 ×	97.3 0.9 ✓	96.6 1.9 ×
Pima I. D.	74.2 1.7 	74.6 2.1 ✓	74.5 2.3 ✓	72.7 1.5 ✓	**76.3** **1.8** ×	75.2 1.9 ✓
Iris	95.0 1.4 	96.0 2.3 ✓	**96.4** **1.8** ×	95.4 1.9 ✓	95.6 2.3 ✓	94.8 3.8 ✓
Wine	**99.0** **1.2** 	97.2 3.5 ✓	98.0 2.0 ×	95.8 2.9 ×	96.2 2.1 ×	96.0 2.4 ×

From Table 6 we can observe that the MLM exhibits an equivalent or even better generalization performance in comparison to the other models. Moreover, the MLM has shown a stable performance since its standard deviations are smaller than those of the other methods, particularly on the Wisconsin B. C. and Wine datasets. In opposite, the SVM model presented the worst performances overall. Based on the hypothesis tests, the MLM achieves performances that are statistically distinct from the other methods, specially for the Wine and Wisconsin B. C. datasets. With regard to Iris and Pima I. D. sets, the MLM provides results mostly equivalent to the state-of-the-art methods.

We report in Figure 41 the MLM performance during the cross-validation procedure. From Figure 41, one may notice that it is not needed as many reference points as learning points, particularly for large datasets, e.g., the Pima I. D. dataset. Although the optimal



(a) Optimal M value over different runs (b) Error per number of reference points

Figure 41: MLM cross-validation performance on classification. Legends also contain the total number of training samples.

number of reference points is about 100% of the learning points ($M = N$) for Wine and Iris datasets (Figure 41a), we observe that 20-40% of the number of learning points has provided a good threshold for selecting M, where the error measures (Normalized Mean Squared Error, NMSE[1]) stabilize (Figure 41b), and then increasing M does not reduce the error considerably.

[1] The mean squared errors have been normalized to be between 0-1.

APPENDIX C

Minimal Learning Machine for Regression

In this appendix, we present the results achieved by the Minimal Learning Machine on eight real-world datasets commonly used for benchmarking purposes in regression.

C.1 Experiments

The performance of the MLM is then compared to what is achieved by five other reference methods: The Extreme Learning Machine (ELM) (HUANG; ZHU; ZIEW, 2006), the Radial Basis Function network (RBF) (BUHMANN, 2003; WU et al., 2012), the Support Vector Machine (SVM) (SMOLA; SCHOLKOPF, 2004), Gaussian Processes (GP) (RASMUSSEN; WILLIAMS, 2006), and the MultiLayer Perceptron (MLP) (BISHOP, 1995).

Table 7: Description of the datasets: input/output dimensionality and number of training/test samples.

Dataset	Dim		# Samples	
	In	Out	Train	Test
Ailerons	5	1	4752	2377
Elevators	6	1	6344	3173
Breast Cancer	32	1	129	65
Boston Housing	13	1	337	169
Servo	4	1	111	56
Abalone	8	1	2784	1393
Stocks	9	1	633	317
Auto Price	15	1	106	53

The datasets are available from the University of California at Irvine (UCI) Repository (`www.ics.uci.edu/~mlearn/`) and a description of the datasets is summarized in

Table 7. The datasets have been chosen to object heterogeneity in the number of samples and inputs. All the datasets have been preprocessed in the same way. Categorical variables have been removed as well as samples containing missing values. Ten different random permutations of the whole datasets are taken, and two thirds are used to create the training set and the remaining for the test set. Then, the training set is normalized to zero mean and unit variance, and the test set is normalized using the same mean and variance from the training set. It may also be noticed that the proportions of the classes, for classification cases, have been kept balanced: each class is represented in an equal proportion, in both training and test sets.

The hyperparameters for the SVM and the MLP are selected using 10-fold cross-validation. The SVM is learned using the SVM toolbox (CHANG; LIN, 2011) with default settings for the hyper-parameters and grid search: the grid is logarithmic between 2^{-2} and 2^{10} for each hyper-parameter; nu-SVC has been used for classification and epsilon-SVR for regression, with radial basis function kernel. The MLP is trained using Levenberg-Marquardt optimization and a range of hidden units from 1 to 20. The learning of GP is based on the default settings in the Matlab Toolbox (RASMUSSEN; WILLIAMS, 2006). The ELM network uses sigmoid kernel and it has been validated with the number of hidden units ranging from 10 to 100 with increments of 10. For the experiments with the RBF network, the centers of the gaussian basis functions are selected by the k-means algorithm, with the number of centers varying from 5% to 50% (step size of 5%) of the number of learning points. We also applied 10-fold cross-validation to select the optimal number of centers. The only hyper-parameter of the Minimal Learning Machine, the number of reference points M, has been optimized using 10-fold cross-validation with reference points randomly selected in a range of 5% to 100% (with a step size of 5%) of the available training samples.

C.1.1 Results

All the models are evaluated using the mean and standard deviation of the resulting MSE over 10 independently drawn test sets. We also carried out a statistical evaluation of the MLM performance against those achieved by the other models using the Wilcoxon signed-ranks test (WILCOXON, 1945) with significance level equal to 5%. The null hypothesis is that the difference between MSE values comes from a distribution with zero median. In our case, we compare the MLM performance against each other method for each dataset.

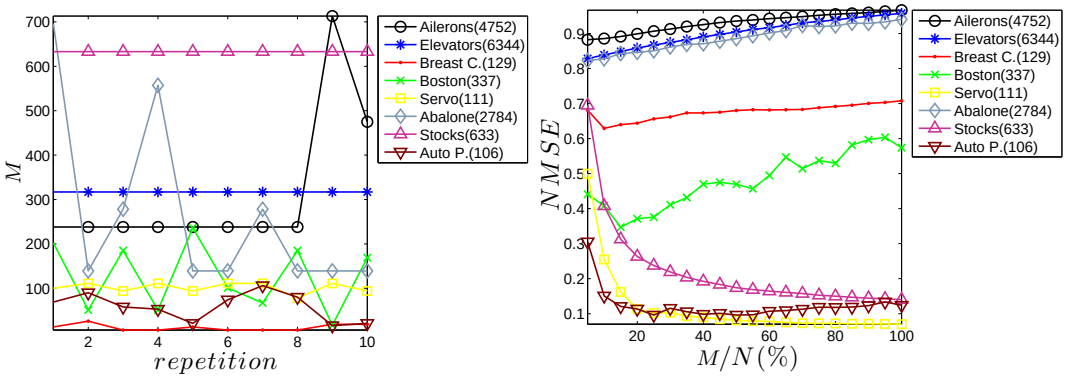
On the basis of the experimental results reported in Table 8, we can observe that the state-of-the-art models seem to be able to achieve similar accuracies. In this regard, the MLM also achieves performances that are comparable to such methods. The MLM presents the smallest MSE (average) for five out of eight regression problems. Also, even though we

Table 8: Test results: MSE, standard deviations (below the MSE) and Wilcoxon signed-ranks test results (✓: fail to reject, and ×: reject). The best performing models are in boldface.

Datasets	**Models**					
	MLM	ELM	RBF	SVM	GP	MLP
Ailerons	**2.7**$e-8$	2.9$e-8$	3.0$e-8$	1.3$e-7$	**2.7**$e-8$	2.7$e-7$
	1.6$e-9$	1.5$e-9$	1.8$e-9$	2.6$e-8$	1.9$e-9$	4.4$e-9$
		×	×	×	✓	×
Elevators	**2.0**$e-6$	2.1$e-6$	2.1$e-6$	6.2$e-6$	**2.0**$e-6$	2.6$e-6$
	6.1$e-8$	5.5$e-8$	6.8$e-8$	6.8$e-7$	5.0$e-8$	9.0$e-8$
		✓	×	×	×	✓
Breast Cancer	**1.1**$e+3$	1.2$e+3$	1.2$e+3$	1.2$e+3$	1.3$e+3$	1.5$e+3$
	1.8$e+2$	1.4$e+2$	1.8$e+2$	7.2$e+1$	1.9$e+2$	4.4$e+2$
		✓	✓	✓	×	×
Boston	1.9$e+1$	2.2$e+1$	2.0$e+1$	3.4$e+1$	**1.1**$e+1$	2.2$e+1$
	9.0	7.1	6.6	3.1$e+1$	3.5	8.8
		✓	✓	✓	×	✓
Servo	**4.6**$e-1$	7.0 $e-1$	6.1$e-1$	6.9$e-1$	4.8$e-1$	6.0$e-1$
	3.0$e-1$	2.5 $e-1$	3.4$e-1$	3.2$e-1$	3.5$e-1$	3.2$e-1$
		×	×	×	✓	✓
Abalone	4.7	4.6	4.7	**4.5**	**4.5**	4.6
	3.3$e-1$	2.4$e-1$	2.3$e-1$	2.7$e-1$	2.4$e-1$	5.0$e-1$
		✓	✓	✓	✓	✓
Stocks	**4.1**$e-1$	9.0$e-1$	7.1$e-1$	5.1$e-1$	4.4$e-1$	8.8$e-1$
	5.8$e-2$	7.3$e-2$	2.0$e-1$	9.8$e-2$	5.0$e-2$	2.1$e-1$
		×	×	×	✓	×
Auto Price	2.6$e+7$	1.3$e+7$	1.1$e+7$	9.8$e+7$	2.0$e+7$	**1.0**$e+7$
	2.7$e+7$	4.1e+6	5.4$e+6$	8.4$e+6$	1.0$e+7$	3.9$e+6$
		✓	✓	✓	✓	✓

have applied random selection of reference points, the standard deviation of MSEs reported by MLM is rather similar to the smallest values achieved by the state-of-the-art methods. The hypothesis tests have shown that the MLM presents statistical difference to the other methods. For the Stocks and Ailerons datasets, the MLM was the best performing model and the null hypothesis can not be rejected only for the GP model. In contrast, the MLM results are not statistically distinct from the other methods for the Auto Price and Abalone datasets — cases in which the MLM is not the best performing model.

As noticed, the MLM performance is quite similar to the state-of-the-art methods. Thus, the computational complexity takes an important role in the decision making process of selecting the most appropriate method. In this regard, an essential aspect for fast MLM training is the number of reference points, or more specifically, the property that the optimal number of reference points does not grow at the same rate of the number of learning points (dataset size). In order to illustrate such a property, we report in Figure 42 the results of the cross-validation phase in terms of selecting M for all the 10 independent runs. Figure 42a shows that the rate between the optimal number of reference points



(a) Optimal M value over different runs (b) Error per number of reference points

Figure 42: MLM cross-validation performance on regression. Legends also contain the total number of training samples.

per number of learning points is almost equal to one for small datasets (up to about 600 samples), and it is stable number. Concerning large datasets (Abalone dataset), our experiments have shown that it is not needed as many reference points as learning points. Figure 42b illustrates how the validation errors (Normalized Mean Squared Error, NMSE[1]) change as a function of the proportion of reference points. Based on Figure 42b, using 20% of the learning points as reference points seems to be a good choice for most datasets.

[1] The mean squared errors have been normalized to be between 0-1.