Nonlinear Black-Box Modeling in System Identification: a Unified Overview

Jonas Sjöberg, Qinghua Zhang,
Lennart Ljung, Albert Benveniste,
Bernard Deylon, Pierre-Yves Glorennec,
Håkan Hjalmarsson, and Anatoli Juditsky *

June 21, 1995

Abstract

A nonlinear black box structure for a dynamical system is a model structure that is prepared to describe virtually any nonlinear dynamics. There has been considerable recent interest in this area with structures based on neural networks, radial basis networks, wavelet networks, hinging hyperplanes, as well as wavelet transform based methods and models based on fuzzy sets and fuzzy rules. This paper describes all these approaches in a common framework, from a user's perspective. It focuses on what are the common features in the different approaches, the choices that have to be made and what considerations are relevant for a successful system identification application of these techniques.

It is pointed out that the nonlinear structures can be seen as a concatenation of a mapping from observed data to a regression vector and a nonlinear mapping from the regressor space to the output space. These mappings are discussed separately. The latter mapping is usually formed as a basis function expansion. The basis functions are typically formed from one simple scalar function which is modified in terms of scale and location. The expansion from the scalar argument to the regressor space is achieved by a radial or a ridge type approach.

Basic techniques for estimating the parameters in the structures are criterion minimization, as well as two step procedures, where first the relevant basis functions are determined, using data, and then a linear least squares step to determine the coordinates of the function approximation. A particular problem is to deal with the large number of potentially necessary parameters. This is handled by making the number of "used" parameters considerably less than the number of "offered" parameters, by regularization, shrinking, pruning or regressor selection.

A more mathematically comprehensive treatment is given in a companion paper (Juditsky et al., 1995).

Keywords : Nonlinear System, Model Structures, Parameter estimation, Wavelets, Neural Networks, Fuzzy Modeling.

1 Introduction

The key problem in system identification is to find a suitable model structure, within which a good model is to be found. Fitting a model within a given structure (parameter estimation) is

*LL, HH, and JS are with Department of Electrical Engineering, Linköping University, S-581 83 Linköping, Sweden, AB, BD, AJ, QZ are with IRISA/INRIA, Campus de Beaulieu, 35042 Rennes cedex, FRANCE, PYG is with INSA, 20 avenue des Buttes de Coësmes, 35045 Rennes cedex, France ; e-mail: `name@isy.liu.se` and `name@irisa.fr`

in most cases a lesser problem. A basic rule in estimation is *not to estimate what you already know.* In other words, one should utilize prior knowledge and physical insight about the system when selecting the model structure. It is customary to distinguish between three levels of prior knowledge, which have been color-coded as follows.

- White Box models: This is the case when a model is perfectly known; it has been possible to construct it entirely from prior knowledge and physical insight.
- Grey Box models: This is the case when some physical insight is available, but several parameters remain to be determined from observed data. It is useful to consider two sub-cases:
  - Physical Modeling: A model structure can be built on physical grounds, which has a certain number of parameters to be estimated from data. This could, e.g., be a state space model of given order and structure.
  - Semi-physical modeling: Physical insight is used to suggest certain nonlinear combinations of measured data signal. These new signals are then subjected to model structures of black box character.
- Black Box models: No physical insight is available or used, but the chosen model structure belongs to families that are known to have good flexibility and have been "successful in the past".

Black Box Models

For black-box *linear* models, the task is really to describe/approximate the system's frequency response (or impulse response) which is just a mapping from $\mathbf{R}$ to $\mathbf{R}^{pm}$ (where p is the number of outputs and m is the number of inputs). With the typically "nice" such functions that dominate applications, this is a rather modest approximation problem, which has been extensively and successfully handled within some well known linear black-box structures. Some typical such structures will be reviewed in Section 3.1.

The nonlinear black-box situation is much more difficult. The main reason for that is that nothing is excluded, and a very rich spectrum of possible model descriptions must be handled. We shall in this paper discuss the possibilities and limitations with such nonlinear black-box identification. The area is quite diverse and covers topics from mathematical approximation theory, via estimation theory and non-parametric regression, to algorithms and currently much-discussed concepts like *neural networks, wavelets* and *fuzzy models.* There are important links to classical statistical approaches in non-parametric regression and density estimation, with kernel methods and nearest neighbor-techniques. There is also a rich literature on the subject. Among many general treatments we may refer to books on neural networks, such as (Kung, 1993), (Haykin, 1994), to books on Fuzzy models, like (Brown and Harris, 1994; Wang, 1994) to books and surveys on non-parametric regression and density estimation, like (Stone, 1982), (Silverman, 1986), and (Devroye and Gyorfi, 1985), and to background material on wavelets and multi-resolution techniques, like (Daubechies, 1992; Chui, 1992; Ruskai et al., 1992; Meyer, 1990)

Organization of this paper

This paper will take the position of a practical user of nonlinear black-box models, describe what are the essential features of the available approaches, and discuss the issues he or she must deal with to successfully arrive at a good model from given observed data. The paper has a companion paper (Juditsky et al., 1995) that complements the material with more theoretical aspects. Each of the two papers can however be read independently.

The present paper is organized as follows. We shall first look into the modeling question and find that the general non-linear black box model can be seen as a concatenation of a mapping from past observed data to a regressor space, and from there by a non-linear, function expansion type, mapping to the space of the system's outputs. This is done in Section 2. The two mappings are then dealt with separately in Sections 3 and 4, respectively.

After an intermission to check the bearings, we then discuss basic model properties in Section 6, giving important insights in how to deal with the potentially large number of parameters required to handle arbitrary non-linear dynamical systems. Estimation techniques based on criterion optimization and direct methods are dealt with in Sections 7 and 8, respectively. How Fuzzy modeling fits into our general framework is then discussed in Section 9. Several numerical examples with real data are given in Section 10, and the user choices and attitudes are discussed in Section 11.

Glossary

We take in this paper a rather classical, statistical approach to the problem. Many earlier treatments, in particular on Neural Networks and Fuzzy models have had other perspectives, and developed special terms for traditional statistical concepts. We provide therefore a glossary for commonly used terms:

```
estimate = train, learn
validate = generalize
model structure = network
estimation data = training set
validation data = generalization set
overfit = overtraining
```

2 Nonlinear Black-Box Structures

The system identification problem is as follows: We have observed inputs, $u(t)$, and outputs, $y(t)$, from a dynamical system:

$$u^t = [u(1), u(2), \ldots, u(t)] \tag{1}$$

$$y^t = [y(1), y(2), \ldots, y(t)] \tag{2}$$

We are looking for a relationship between past observations $[u^{t-1}, y^{t-1}]$ and future outputs, $y(t)$:

$$y(t) = g(u^{t-1}, y^{t-1}) + v(t) \tag{3}$$

The additive term $v(t)$ accounts for the fact that the next output $y(t)$ will not be an exact function of past data. However, a goal must be that $v(t)$ is small, so that we may think of $g(u^{t-1}, y^{t-1})$ as a good prediction of $y(t)$ given past data.

Equation (3) models general discrete time dynamic systems. Since static systems can be viewed as a particular case of dynamic systems, we mainly focus on dynamic systems in this paper.

Now, how do we find the function g in (3)? In some way or another we have to search for it within a family of functions. Let us parameterize this function family with a finite-dimensional parameter vector θ:

$$g(u^{t-1}, y^{t-1}, \theta) \ . \tag{4}$$

Parameterizing the function g with a finite dimensional vector θ is usually an approximation. Indeed the main topic of this paper is how to find a good such parameterization and how to deal with it. Once we have decided upon such a structure and have collected a data set $[u^N, y^N]$ the quality of θ can naturally be assessed by means of the fit between the model and the data record:

$$\sum_{t=1}^{N} \|y(t) - g(u^{t-1}, y^{t-1}, \theta)\|^2 \ . \tag{5}$$

The norm and the actual way of achieving or trying to achieve the minimum in θ may differ, but most system identification schemes follow this concept.

Now, the model structure family (4) is really too general, and it turns out to be useful to write g as a concatenation of two mappings: one that takes the increasing number of past observations u^t, y^t and maps them into a finite dimensional vector $\varphi(t)$ of fixed dimension and one that takes this vector to the space of the outputs:

$$g(u^{t-1}, y^{t-1}, \theta) = g(\varphi(t), \theta) \tag{6}$$

where

$$\varphi(t) = \varphi(u^{t-1}, y^{t-1}) \tag{7}$$

We shall call this vector *regression vector* and its components will be referred to as *regressors.* We also allow the more general case that the formation of the regressors is itself parameterized:

$$\varphi(t) = \varphi(u^{t-1}, y^{t-1}, \eta) \tag{8}$$

which we for short write $\varphi(t, \eta)$. Sometimes $\eta = \theta$, i.e., the regression vector depends on all the model parameters. For simplicity, the extra argument η will however be used explicitly only when essential for the discussion.

The choice of the nonlinear mapping in (4) has thus been decomposed into two partial problems for dynamical systems:

1. How to choose the regression vector $\varphi(t)$ from past inputs and outputs.

2. How to choose the nonlinear mapping $g(\varphi)$ from the regressor space to the output space.

We shall address the possibilities for these two choices in the following two sections.

3 Regressors: Possibilities

To get some guidance about the choice of regressors, let us first review the linear case.

3.1 A Review of Linear Black-Box Models

The simplest dynamical model is the Finite Impulse Response model (FIR):

$$\begin{aligned} y(t) &= B(q)u(t) + e(t) \\ &= b_1 u(t-1) + \ldots + b_n u(t-n) + e(t) \end{aligned} \tag{9}$$

Here we have used q to denote the shift operator, so $B(q)$ is a polynomial in q^{-1}. The corresponding predictor $\widehat{y}(t|\theta) = B(q)u(t)$ is thus based on the regression vector

$$\varphi(t) = [u(t-1), u(t-2), \ldots, u(t-n)] \ .$$

As n tends to infinity we may describe the dynamics of all ("nice") linear systems. However, the character of the noise term $e(t)$ will not be modeled in this way.

The linear black-box structures used in practice are all variants of (9), using different ways of picking up "poles" of the system and different ways of describing the noise characteristics. The common models used can all, as in (Ljung, 1987), be summarized by the general family

$$A(q)y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t) \tag{10}$$

The special cases of (10) are known as the *Box-Jenkins* (BJ) model ($A = 1$), the *ARMAX* model ($F = D = 1$), the *Output-Error* (OE) model ($A = C = D = 1$) and the *ARX* model ($F = C = D = 1$). The predictor associated with (10) can be given in "pseudo-linear" regression form as (see eq (3.114) in (Ljung and Söderström, 1983))

$$\widehat{y}(t|\theta) = \theta^T \varphi(t, \theta) \tag{11}$$

The regressors, i.e., the components of $\varphi(t, \theta)$ are in this general case given by

1. $u(t-k)$ (associated with the B-polynomial)
2. $y(t-k)$ (associated with the A-polynomial)
3. $\widehat{y}_u(t-k|\theta)$ Simulated outputs from past u only (associated with the F-polynomial)
4. $\varepsilon(t-k) = y(t-k) - \widehat{y}(t-k|\theta)$ Prediction errors (associated with the C-polynomial)
5. $\varepsilon_u(t-k) = y(t-k) - \widehat{y}_u(t-k|\theta)$ Simulation errors (associated with the D-polynomial)

It should be remarked that in case $A \neq 1$, "simulated output" refers to the quantity $A(q)y(t)$.

A linear state-space model in predictor form

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) + K(y(t) - Cx(t)) \\ y(t) &= Cx(t) + e(t) \end{aligned} \tag{12}$$

can also be described as a pseudo-linear regression (11), with the predictor $\hat{y}(t|\theta) = Cx(t)$, and the states x being the regressors. Note that each component in $x(t)$ is obtained by linear filtering of past inputs and outputs, through filters that depend on θ (i.e., the matrices A, B, C and K.

$$x_i(t) = F_i^u(q, \theta)u(t) + F_i^y(q, \theta)y(t) \tag{13}$$

If $K = 0$, then $F_i^y(q, \theta) \equiv 0$, and we have a model of output error type.

The essential difference between the state-space regressors and the input-output regressors described earlier is that the latter contain blocks of the same regressor, time shifted a number of steps. This is also characteristic of state-space models of echelon type.

State-space regressors are thus less restricted in their internal structure. This implies that it might be possible to obtain a more efficient model with a smaller number of regressors by using a state-space model. State-space models in connection with neural nets are discussed in, e.g., (Rivals, 1995; Nerrand et al., 1993; Matthews, 1992).

3.2 Regressors for Nonlinear Black-Box Dynamical Models

The described regressors give all the necessary freedom for the linear black-box case, and it is natural to use these also in the nonlinear case. We thus work with structures of the kind

$$\widehat{y}(t|\theta) = g(\varphi(t), \theta) \tag{14}$$

where g is some nonlinear function parameterized by θ and the components of $\varphi(t)$ are similar to the just described regressors. For the input-output case, the two first ones, $u(t-k)$ and $y(t-k)$, are measured variables and cause no problems to include. The remaining three are all based on previous outputs from the black-box model $\widehat{y}(t-k|\theta)$, so we should write $\varphi(t,\theta)$ instead of $\varphi(t)$ in (14). The question then also arises how the simulated output $\widehat{y}_u(t-k|\theta)$ is computed if the network produces predicted outputs $\widehat{y}(t-k|\theta)$. The answer is that the output from the model (14) is equal to $\widehat{y}_u(t|\theta)$ if all measured outputs $y(t-k)$ in the regressors are replaced by the last computed $\widehat{y}_u(t-k|\theta)$.

Following the nomenclature for linear models it is natural to coin similar names for nonlinear models. This is well in line with, e.g. (Chen et al., 1990; Chen and Billings, 1992). We could thus distinguish between

- NFIR-models, which use only $u(t-k)$ as regressors
- NARX-models, which use $u(t-k)$ and $y(t-k)$ as regressors
- NOE-models, which use $u(t-k)$ and $\widehat{y}_u(t-k|\theta)$ as regressors. In this case the output of the model is also $\widehat{y}(t|\theta)$.
- NARMAX-models, which use $u(t-k)$, $y(t-k)$, and $\varepsilon(t-k|\theta)$ as regressors
- NBJ-models, which use $u(t-k)$, $\widehat{y}(t-k|\theta)$, $\varepsilon(t-k|\theta)$, and $\varepsilon_u(t-k|\theta)$ as regressors. In this case the simulated output $\widehat{y}_u$ is obtained as the output from (14), by using the same structure, replacing ε and ε_u by zeros in the regression vector $\varphi(t,\theta)$
- NON-LINEAR STATE-SPACE models, which use past components of virtual outputs, i.e., signal values at internal nodes of the network (see e.g. Figure 3 below) that do not correspond to the output variable.

In (Narendra and Parthasarathy, 1990) another notation is used for the same models when used in conjunction with neural networks. The NARX model is called Series-Parallel model and the NOE is called Parallel model.

The model structures NOE, NBJ, NARMAX and the non-linear state-space model correspond to recurrent structures (see Subsection 4.3) because parts of the regression vector consist of past outputs from the model. It is in general harder to work with recurrent structures. Among other things it becomes difficult to check under what conditions the obtained predictor model is stable, and it takes an extra effort to calculate gradients for model parameter estimation.

3.3 Other Choices of Regressors

So far we have discussed regressors that are just linear functions of measured inputs, measured outputs and model outputs. With physical insight about the system at hand, one should utilize that information to form new variables by transformations of the raw measurements. From a practical point of view, it is sufficient to regard what we have called *input*, u, and *output*, y, here, as suitable transformations of the raw measurements, formed in view of what is known about the system. Such *"Semi-physical Regressors"* could for example be a power signal formed by voltage and current measurements, if we believe that to be the essential stimulus for the system. Even if nonlinear structures are to be applied, there is no reason to waste parameters to estimate facts that are already known.

Another type of preprocessing of raw data in the light of prior knowledge is to use filtered input as regressors like

$$L_k(q)u(t), \quad k = 1, \ldots, d$$

rather than $u(t-k)$, where the filters L_k are tailored to the application. Laguerre and Kautz filters have been extensively discussed in these applications, e.g. (Wahlberg, 1991) and (Wahlberg, 1994). In (van den Hof et al., 1994) interesting generalizations of such regressor choices are described.

3.4 Some Other Structural Questions

The actual way that the regressors are combined clearly reflects structural assumptions about the system. Let us, for example, consider the assumption that the system disturbances are additive, but not necessarily white noise:

$$y(t) = g(u^t) + v(t) \tag{15}$$

Here u^t denotes all past inputs, and $v(t)$ is a disturbance, for which we only need a spectral description. It can thus be described by

$$v(t) = H(q)e(t)$$

for some white sequence $\{e(t)\}$. The predictor for (15) then is

$$\widehat{y}(t) = (1 - H^{-1}(q))y(t) + H^{-1}(q)g(u^t) \tag{16}$$

In the last term, the filter H^{-1} can equally well be subsumed in the general mapping $g(u^t)$. The structure (15) thus leads to a NFIR or NOE structure, complemented by a *linear* term containing past y.

In (Narendra and Parthasarathy, 1990) a related Neural Network based model is suggested. It can be described by

$$\widehat{y}(t) = f(\theta_1, \varphi_1(t)) + g(\theta_2, \varphi_2(t)) \tag{17}$$

where $\varphi_1(t)$ consists of delayed outputs and $\varphi_2(t)$ of delayed inputs. The parameterized functions f and g can be chosen to be linear or nonlinear by a neural net. A further motivation for this model is that it becomes easier to develop controllers from (17) than from the models discussed earlier.

In (McAvoy, 1992), it is suggested first to build a linear model for the system. The residuals from this model will then contain all unmodeled nonlinear effects. The Neural Net model could then be applied to the residuals (treating inputs and residuals as input and output), to pick up the nonlinearities. This is attractive, since the first step to obtain a linear model is robust and often leads to reasonable models. By the second Neural Net step, we are then assured to obtain at least as good a model as the linear one.

4 Nonlinear Mappings: Possibilities

4.1 Function Expansions and Basis Functions

The Basic Features

Now let us turn to the nonlinear mapping

$$g(\varphi, \theta) \tag{18}$$

which for any given θ goes from $\mathbf{R}^d$ to $\mathbf{R}^p$. At this point it does not matter how the regression vector $\varphi = (\varphi_1, \ldots, \varphi_d)^T$ was constructed. It is just a vector that lives in $\mathbf{R}^d$.

It is natural to think of the parameterized function family as function expansions:

$$g(\varphi, \theta) = \sum \alpha_k g_k(\varphi) \ . \tag{19}$$

We refer to g_k as *basis functions*, since the role they play in (19) is similar to that of a functional space basis. In some particular situations, they do constitute a functional basis. Typical examples are wavelet bases (see subsection 8.1).

We are going to show that expansion (19) with different basis functions, together with all the possible choice of regression vector φ presented in the previous section, plays the role of a unified framework for investigating most known nonlinear black-box model structures.

Now, the key question is: How to choose the basis functions g_k? Most well know nonlinear black-box model structures are composed of g_k obtained by parameterizing a single "mother basis function" that we generically denote by $\kappa(x)$. In such situations we generally write

$$g_k(\varphi) = \kappa(\varphi, \beta_k, \gamma_k) \text{``} = \kappa(\beta_k(\varphi - \gamma_k))'' \ . \tag{20}$$

The last equation is to be interpreted symbolically, and will be specified more precisely below. It stresses that β_k and γ_k denote parameters of different nature. Typically, β_k is related to the scale or to some directional property of $g_k(\varphi)$, and γ_k is some position or translation parameter.

A Scalar Example: Fourier Series Take $\kappa(x) = \cos(x)$. Then (19),(20) will be the Fourier series expansion, with β_k as the frequencies and γ_k as the phases.

Another Scalar Example: Piecewise Constant Functions Take κ as the unit interval indicator function:

$$\kappa(x) = \begin{cases} 1 & \text{for } 0 \leq x < 1 \\ 0 & \text{else} \end{cases} \tag{21}$$

and take, for example, $\gamma_k = k$, $\beta_k = 1/\Delta$ and $\alpha_k = f(k\Delta)$. Then (19), (20) gives a piecewise constant approximation of any function f. Clearly we would have obtained a quite similar result by a smooth version of the indicator function, e.g., the Gaussian bell:

$$\kappa(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \tag{22}$$

A Variant of the Piece-wise constant case Take κ to be the unit step function

$$\kappa(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases} \tag{23}$$

We then just have a variant of (21), since the indicator function can be obtained as the difference of two steps. A smooth version of the step, like the *sigmoid* function

$$\kappa(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \tag{24}$$

will of course give quite similar results.

Classification of single-variable basis functions

Two classes of single-variable basis functions can be distinguished depending on their nature:

- *Local Basis Functions* are functions having their gradient with bounded support, or at least vanishing rapidly at infinity. Loosely speaking, their variations are concentrated to some interval.

- *Global Basis Functions* are functions having infinitely spreading (bounded or not) gradient.

Clearly the Fourier series is an example of a global basis function, while (21), (22), (23) and (24) are all local functions.

Construction of multi-variable basis functions

In the multi-dimensional case ($d > 1$), g_k are multi-variable functions. In practice they are often constructed from the single-variable function κ in some simple manner. Let us recall the three most often used methods for constructing multi-variable basis functions from single-variable basis functions.

1. **Tensor product.** Given d single-variable functions $g_1(\varphi_1), \ldots, g_d(\varphi_d)$ (identical or not), the tensor product construction of multi-variable basis function is given by their product $g_1(\varphi_1) \cdots g_d(\varphi_d)$.

2. **Radial construction.** For any single-variable function κ the radial construction of multi-variable basis function of $\varphi \in \mathbf{R}^d$, has the form

$$g_k(\varphi) = g_k(\varphi, \beta_k, \gamma_k) = \kappa(\|\varphi - \gamma_k\|_{\beta_k}) \tag{25}$$

where $\|\cdot\|_{\beta_k}$ denotes any chosen norm on the space of the regression vector φ. The norm could typically be a quadratic norm

$$\|\varphi\|^2_{\beta_k} = \varphi^T \beta_k \varphi \tag{26}$$

with β_k as a possibly k-dependent positive definite matrix of dilation (scale) parameters. In simple cases β_k may be just scaled versions of the identity matrix.

3. **Ridge construction.** Let κ be any single-variable function. Then for all $\beta_k \in \mathbf{R}^d$, $\gamma_k \in \mathbf{R}$, a *ridge* function is given by

$$g_k(\varphi) = g_k(\varphi, \beta_k, \gamma_k) = \kappa(\beta_k^T \varphi + \gamma_k),\ \varphi \in \mathbf{R}^d \tag{27}$$

The ridge function is thus constant for all φ in the sub-space $\{\varphi \in \mathbf{R}^d\ :\ \beta_k^T \varphi = \text{ constant}\}$. As a consequence, even if the mother basis function κ has local support, the basis functions g_k will have unbounded support in this subspace. The resulting basis could be said to be *semi-global*, but the term *ridge function* is more precise.

Let us comment on the different possibilities. For evaluating a function constructed by *tensor product*, its factor functions must be evaluated separately, thus the computational cost is roughly proportional to the dimension d. For a function constructed by the other two methods, the dimension-dependent computational cost stays only in the evaluation of the norm of $\varphi - \gamma_k$ or the inner product $\beta_k^T \varphi$, consequently the dimension dependence is much weaker. For this reason, the tensor product is rarely used in large dimensional case. On the other hand, these methods yield very different forms of multi-variable functions. By using factors of different natures, the *tensor product construction* allows to build functions that behave very differently in different directions. The *radial construction* ensures some directional homogeneity. The *ridge construction* also offers some direction selective feature, even if these basis functions are necessarily constant in some directions. This, however, turns out to be a quite useful property in many practical cases. Note also that in some particular situations, two methods may lead to the same result, e.g. a multi-variable Gaussian function can be obtained by both tensor product construction and radial construction.

4.2 Connection to "Named Structures"

Here we briefly review some popular model structures. Other structures related to interpolation techniques are discussed in (Juditsky et al., 1995). They all have the general form of function expansions (19), and most of them are composed of basis functions g_k obtained by parameterizing some particular "*mother basis function*" κ as described in the previous section.

Wavelets. Wavelet decomposition is a typical example for the use of local basis functions. Loosely speaking, the "mother basis function" (usually referred to as *mother wavelet* in the wavelet literature, and there denoted by ψ rather than κ) is dilated and translated to form a wavelet basis[1]. In this context it is common to let the expansion (19) be doubly indexed according to scale and location, and use the specific choices (for one dimensional case) $\beta_j = 2^j$ and $\gamma_k = k$. This gives, in our notation,

$$g_{j,k}(\varphi) = 2^{j/2} \kappa(2^j \varphi - k)\ j, k \in \mathbf{Z}\ . \tag{28}$$

The multi-variable wavelet functions can be constructed by tensor products of scalar wavelet functions, but this is not the preferred method. See Section 8.

Compared to the simple example of a piece-wise constant function approximation in Section 4.1, we have here *multi-resolution* capabilities, i.e. several different scale parameters are used simultaneously and overlappingly. With suitably chosen mother wavelet and appropriate translation and dilation parameters, the wavelet basis can be made orthonormal, which makes it easy to compute the coordinates $\alpha_{j,k}$ in (19). We shall discuss this in some detail in Subsection 8.1 and extensively in (Juditsky et al., 1995).

[1] Strictly speaking, sometimes the dilated and translated wavelets may be a *frame* instead of a basis. See (Daubechies, 1992).

Wavelet and Radial Basis Networks. The choice of local basis functions in combination with the radial construction for multi-variable case (25), without any orthogonalization is found in both wavelet networks (Zhang and Benveniste, 1992) and radial basis neural networks (Poggio and Girosi, 1990).

Kernel estimators. Another well known example for use of local basis functions is *Kernel estimators* (Nadaraya, 1964; Watson, 1969). A kernel function[2] $\kappa(\cdot)$ is typically a bell-shaped function, and the kernel estimator has the form

$$g(\varphi) \quad = \quad \sum_{k=1}^{n} \alpha_k \, \kappa \left(\frac{\varphi - \gamma_k}{h} \right) \tag{29}$$

where h is a small positive number, γ_k are given points in the space of regression vector φ. This clearly is a special case of (19), (20).

Nearest Neighbors or Interpolation. Models which produce outputs depending on the closest estimation data points and interpolation models can also be described as expansions in basis functions. Assume that the data are drawn such that their φ-values form a uniform lattice in $\mathbf{R}^d$, Take κ as the indicator function (21), expanded to a hypercube by the radial approach (25) (using the max norm). Then choose the location and scale parameters in (25) such that the cubes $\kappa(\|\varphi - \gamma_k\|_{\beta_k})$ are tightly laid and that exactly each data point falls at the center of one cube. The corresponding expansions (19) will then be equivalent to the nearest neighbor model which consists in, for any value $\widehat{\varphi}$, taking as the output estimate the y-value of the data point whose φ-value is the closest to $\widehat{\varphi}$.

B-splines. B-splines are local basis functions which are piecewise polynomials. The connections of the pieces of polynomials have continuous derivatives up to a certain order, depending on the degree of the polynomials (De Boor, 1978; Schumaker, 1981). Splines are very nice functions, since they are computationally very simple and can be made as smooth as desired. For these reasons, they have been widely used in classic interpolation problems.

Sigmoid Neural Networks. The combination of the model expansion (19), with a ridge basis function (27) and the sigmoid choice (24) for mother function, gives the celebrated *one hidden layer feed-forward sigmoid neural net.*

Hinging Hyperplanes. The *hinging hyperplanes* model (Breiman, 1993) is closely related to the neural network, and corresponds to the choice of the *hinge function* rather than the sigmoid, for the mother basis function κ. The hinge function has the form of an "open book" (see Figure 1)) and is defined (Breiman, 1993) as

$$h(\varphi) = \pm \max \{\beta^+ \varphi + \gamma^+ \ , \ \beta^- \varphi + \gamma^- \}$$

where β^+, β^- are row vectors and γ^+, γ^- are scalars. In (Pucar and Sjöberg, 1995b) it is shown that the hinging hyperplane model is over-parameterized in its original form. By introducing the basis functions

$$\kappa(x) = \begin{cases} 0 & \text{for } x < 0 \\ \pm x & \text{for } x > 0 \end{cases}$$

[2]Usually the kernel function is noted as $K(\cdot)$.

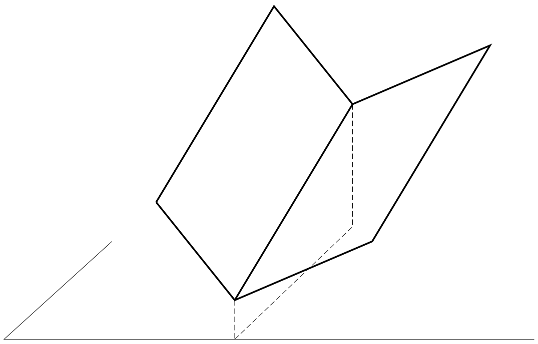

Figure 1: *A hinge function; the building block for hinging hyperplane models*

the hinging hyperplanes model can be expressed as

$$\sum \kappa(\beta^T \varphi + \gamma) + \mu^T \varphi + \gamma_0$$

where μ is a parameter vector with the same dimension as φ. Hence, the hinging hyperplane model is a ridge constructions with an additional linear term. Using hinge functions as basic functions yields the kind of piecewise linear model, proposed by (Sontag, 1981).

Projection pursuit regression. Another example of ridge type basis functions is the *projection pursuit regression* (Huber, 1985; Friedman and Stuetzle, 1981) having the form

$$g(\varphi, \theta) \quad = \quad \sum_k \alpha_k g_k \left(\beta_k \varphi + \gamma_k \right) \ , \tag{30}$$

where β_k are q by d matrices, $\varphi \in \mathbf{R}^d$, $d > q$, and $g_k \ : \ \mathbf{R}^q \to \mathbf{R}$ are some smooth fitted functions. The connection to our framework is obvious. The term "projection pursuit" derives from the fact that the q selected dimensions represent the projections in the regressor space which show the most significant patterns. In other words, there is not much that happens across these subspaces.

Partial Least Squares. The ridge basis function approaches have a connection, at least conceptually, to the Partial Least Squares (PLS) techniques, much used in Chemometrics, (Wold et al., 1984; Helland, 1990). PLS also employs techniques to select the most significant subspaces of a larger regressor space, so as to reduce the number of parameters to estimate.

Fuzzy Models. Also the so-called *fuzzy models* belong to the model structures of the class (19). In this case, the basis functions g_k are constructed from the fuzzy set membership functions and inference rules. How this works is further discussed in Section 9.

4.3 Network questions

So far we have viewed the model structures as *basis function expansions*, albeit with adjustable basis functions. Such structures are often referred to as *networks*, primarily since typically one "mother basic function" κ is repeated a large number of times in the expansion. Graphical illustrations of the structure therefore look like networks.

Multi-layer Networks

The network aspect of the function expansion is even more pronounced if the basic mappings are convolved with each other in the following manner:

Let the outputs of the basis functions be denoted by

$$\varphi_k^{(2)}(t) = g_k(\varphi(t)) = \kappa(\varphi(t), \beta_k, \gamma_k)$$

and collect them into a vector

$$\varphi^{(2)} = \left[\varphi_1^{(2)}(t), \ldots, \varphi_n^{(2)}(t)\right] .$$

Now, instead of taking a linear combination of these $\varphi_k^{(2)}(t)$ as the output of the model (as in (19)), we could treat them as new regressors and insert then into another "layer" of basis functions forming a second expansion:

$$g(\varphi, \theta) \quad = \quad \sum_l \alpha_l^{(2)} \kappa(\varphi^{(2)}, \beta_l^{(2)}, \gamma_l^{(2)}) \tag{31}$$

were θ denotes the whole collection of involved parameters: $\alpha_k, \beta_k, \gamma_k, \alpha_l^{(2)}, \beta_l^{(2)}, \gamma_l^{(2)}$. Within neural network terminology, (31) is called a *two-hidden layer* network. The basis functions $\kappa(\varphi(t), \beta_k, \gamma_k)$ then constitute the first hidden layer, while $\kappa(\varphi^{(2)}, \beta_l^{(2)}, \gamma_l^{(2)})$ give the second one. The layers are "hidden" because they do not show up explicitly in the output $g(\varphi, \theta)$ in (31), but they are of course available to the user. See Figure 2 for an illustration. Clearly we can repeat the procedure an arbitrary number of times to produce *multi-layer* networks. This term is primarily used for sigmoid neural networks, but applies as such to any basis function expansion (19).

The question of how many layers to use is however not easy. In principle, with many basis functions, one hidden layer is sufficient for modeling most practically reasonable systems. See, for example, (Cybenko, 1989; Barron, 1993). (Sontag, 1993) contains many useful and interesting insights into the importance of second hidden layers in the nonlinear structure.

Recurrent Networks

Another very important concept for applications to dynamical systems is that of *recurrent networks*. This refers to the situation that some of the regressors used at time t are outputs from the model structure at previous time instants:

$$\varphi_k(t) = g(\varphi(t-k), \theta)$$

See the illustration in Figure 3. It can also be the case that some component $\varphi_j(t)$ of the regressor at time t is obtained as a value from some interior node (not just at the output layer) at a previous time instant. Such model dependent regressors make the structure considerably more complex, but offer at the same time quite useful flexibility.

One might distinguish between input/output based networks and state-space based networks, although the difference is less distinct in the non-linear case. The former would be using only past outputs from the network as recurrent regressors, while the latter may feed back any interior point in the network to the input layer as a recurrent regressor. Experience with state-space based networks is quite favorable, e.g., (Rivals, 1995; Nerrand et al., 1993; Matthews, 1992).

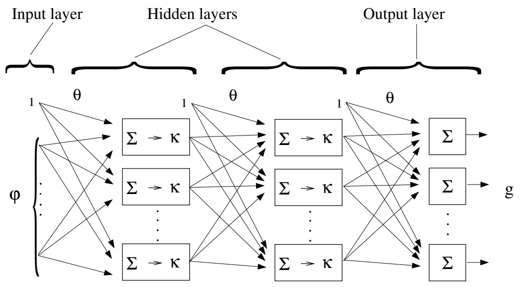

Figure 2: Feedforward network with two hidden layers.

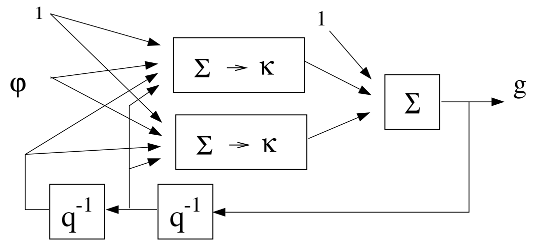

Figure 3: Example of a recurrent network. q^{-1} delays the signal one time sample.

5 Intermission

The rather formidable task to finding a black-box, non-linear model description has now been reduced to the following subproblems;

1. Select the regressors φ.

2. Select a scalar mother basis function κ

3. Let the expansion of this mother function into the regressor space be either of radial (25) or ridge (27) type, or possibly be a specific multidimensional function.

4. Determine the number of basis functions to be used in (19), as well as the number of hidden layers, according to (31).

5. Determine the values of the dilation and location parameters β_k and γ_k.

6. Determine the coordinate parameters α_k in (19).

The remainder of this article will deal with these steps. We shall discuss the user aspects of issues 1 and 3 in Section 11.

The combined effects of the choices in 2, 3, 4 and 5 will affect the approximating power of the model structure. The companion paper (Juditsky et al., 1995) is specifically devoted to this question.

The issues we now turn to are steps 5 and 6, which are the estimation questions. Basically, there are two possibilities for the dilation and location parameters in step 5:

- Let β and γ be continuous variables and estimate them at the same time as the α-parameters.

- Treat β and γ separately, for example by offering predetermined values for them, as in the wavelet approach. Then the estimation of coordinates α is a linear regression problem for one layer networks.

We shall deal with these approaches in Sections 7 and 8, respectively. First, however, in the next section we shall review some general aspects on model estimation.

6 Model Estimation and Model Properties

Several different techniques have been developed for estimating models. We will discuss such methods in more detail in the following two sections, but we shall first point to some basic and general features that affect the model properties. These turn out to have important implications both for the choice of basis functions and for the actual estimation process.

6.1 Models and Model Estimation

Consider our general black-box model

$$y(t) \sim g(\varphi(t), \theta) = \sum_{k=1}^{n} \alpha_k g_k(\varphi(t), \beta_k) \tag{32}$$

in which the parameters γ_k previously used have been included into β_k for brevity in the following discussion. For chosen basis functions g_k, a main goal with the model estimation is to choose the parameters so that the model fit becomes good. Assume that we are given a (finite) set Z_e^N of (measured) regressor-output pairs:

$$Z_e^N = \{(y(t), \varphi(t)) \ : t = 1, \ldots, N\} \ . \tag{33}$$

We refer to Z_e^N as the *estimation data set*, since the model parameter estimation will rely on it. Note that all or part of the parameters θ (i.e., α_k, β_k) need to be estimated from the data Z_e^N, depending on the choice of the basis functions and the estimation method. In the following, let us denote – with some abuse of notation – the *estimated part* of the parameters by the vector θ. Other, non-estimated parameters, will be subsumed in the basis functions g_k. Note that the dimension of θ is proportional to n, the number of basis functions used in (32). The actual number of estimated parameters, $\dim \theta$, will be denoted by m.

Now, a leading guideline for estimating θ will be to minimize the error between the output of the model and the measured output using Z_e^N, like in (5):

$$\min_\theta V_N(\theta, Z_e^N) = \frac{1}{N} \sum_{t=1}^{N} \|y(t) - g(\varphi(t), \theta)\|^2 \tag{34}$$

The actual method may be to perform this minimization explicitly (as detailed with in Section 7) or to use some constructive methods (as discussed in Section 8).

6.2 Model Quality

Suppose that the actual data can be described by

$$y(t) = g_0(\varphi(t)) + e(t) \tag{35}$$

where g_0 is some unknown "true model" and $e(t)$ is white noise with variance λ.

Let the estimate of θ based on Z_e^N be denoted by $\hat{\theta}_N$. We then want $g_0(\varphi)$ and $g(\varphi(t), \hat{\theta}_N)$ to "be close".

Measures of Model Quality

How to measure the quality of the model? There are of course many possible measures, suitable for different applications. We shall here focus on one that allows some important analytical results. We measure the fit between any given model θ and the true system by

$$\bar{V}(\theta) = E\|y(t) - g(\varphi(t), \theta)\|^2 = \lambda + E\|g_0(\varphi(t)) - g(\varphi(t), \theta)\|^2 \tag{36}$$

where we recall that λ is the variance of the noise $e(t)$. In this expression the regressors $\varphi(t)$ are assumed to be a stationary process. For most practical purposes, and under quite general conditions it can also be interpreted as the sample mean:

$$\bar{V}(\theta) = \lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} \|y(t) - g(\varphi(t), \theta)\|^2 \tag{37}$$

Here, no other conditions have to be imposed, other than that the indicated limit exists.

It is important to realize that the measure $\bar{V}$ *depends on the properties of the regressors.* What is "a good model" thus depends on what regressor sequence it will be applied to. In what follows we shall assume that the regressors in the measure (36) *have the same properties (distribution) as those used in* Z_e^N. This is a very important restriction. Within a given model structure, parameterized by θ of dimension m, we can define the best model according to the chosen quality measure:

$$\theta_*(m) = \arg\min_{\theta} \bar{V}(\theta) \tag{38}$$

where we showed explicitly the dependence on m. Note, again, that $\theta_*(m)$ will depend on the properties of φ.

To measure the quality of a given model $\hat{\theta}_N$ we shall use

$$E\bar{V}(\hat{\theta}_N) = V_*(m) \tag{39}$$

Here expectation E is with respect to the model $\hat{\theta}_N$. The measure (39) thus describes *the model's expected fit to the true system, when applied to a new data set, with the same properties (distribution) of the regressors* φ. In the notation $V_*(m)$ we stress that this measure, for given regressor properties, and a given model structure family, only depends on the model size m.

Basic Facts: Bias and Variance

We shall now quote some quite general results on model quality that can be found, e.g. in Chapter 16 of (Ljung, 1987). They are entirely independent of the model structure used, and are valid under quite general conditions.

Assume that the estimate $\hat{\theta}_N$ is obtained by minimization of (34). Assume also the model $\theta_*(m)$ is "quite good" in the sense that the model residuals should be white noise. Then the model quality criterion $V_*(m)$ as defined in (39) can be expressed as

$$\begin{aligned} V_*(m) &= E\bar{V}(\hat{\theta}_N) = \lambda + E\|g_0(\varphi(t)) - g(\varphi(t), \hat{\theta}_N)\|^2 \\ &\approx \underbrace{\lambda}_{\text{noise}} + \underbrace{E\|g_0(\varphi) - g(\varphi, \theta_*(m))\|^2}_{\text{bias}} + \underbrace{E\|g(\varphi, \theta_*(m)) - g(\varphi, \hat{\theta}_N)\|^2}_{\text{variance}} \end{aligned} \tag{40}$$

As indicated, $V_*(m)$ can be approximately decomposed into two parts: one due to the bias, the other due to the variance of the estimation. They are further examined in the following.

Bias As N tends to infinity we have

$$\hat{\theta}_N \to \theta_*(m) \tag{41}$$

then $V_*(m)$ will only involve the bias part. The estimate will thus converge to the best possible approximation of the true system, for the given model structure and model size (as measured by its prediction performance under the regressor properties used in the estimation data set).

Variance The estimated parameter vector $\hat{\theta}_N$ will have a certain covariance matrix, that describes its deviation from $\theta_*(m)$. This matrix is mostly not of direct interest, since the parameters do not have physical significance. Let us instead translate the variation in θ to the resulting variation in prediction performance. This gives

$$E\|g(\varphi(t),\hat{\theta}_N) - g(\varphi(t),\theta_*(m))\|^2 \approx \lambda\frac{m}{N}\ . \tag{42}$$

Here, as before, $\lambda = Ee^2(t)$, the variance of the true prediction errors defined in (35). The approximate equality is asymptotic in N. Also the expression is given for the case of scalar y. (In the multivariate case the quadratic norm used in (42) should be taken as the inverse of the covariance matrix of $e(t)$. The factor λ should then be omitted: it is subsumed in the used norm.)

Combining (40) and (42) gives

$$V_*(m) = E\bar{V}(\hat{\theta}_N) = \lambda + \lambda\frac{m}{N} + E\|g_0(\varphi) - g(\varphi,\theta_*(m))\|^2 = \bar{V}(\theta_*(m)) + \lambda\frac{m}{N}\ . \tag{43}$$

A useful interpretation of (43) is that it displays the expected loss when the model is applied to a new data set. It is important to realize that the expected value of the minimized loss function (i.e. the model's performance when applied to the estimation data) is quite different. With V_N defined by (34) we have

$$EV_N(\hat{\theta}_N) \approx \bar{V}(\theta_*(m)) - \lambda\frac{m}{N}\ . \tag{44}$$

Basic Consequences: Spurious Parameters

Within a given model structure family $\bar{V}(\theta_*(m))$ is a non-increasing function of m: The potential approximation degree increases with the number of basis functions used. The approximation capabilities of different structures in this respect will be commented upon shortly in Section 6.3. However, when the model is estimated, there is a direct penalty in using many parameters, as manifested by the variance contribution. An added parameter (m increased by 1) could very well be useful in that it decreases $V_*(\theta(m))$. However, as long as this decrease is less than λ/N the addition of this parameter is harmful for the overall model quality $V_*(m)$, and the parameter should not be included. We call such a parameter *spurious*. The term *overfit* is often used to describe what happens when spurious parameters are employed.

6.3 Model Structure Flexibility

Having the bias small for a given parameter dimension is a matter of having an efficient function basis: small bias achieved with few basis functions. Thus a great deal of attention is paid in the quality of basis function in terms of function approximation, regardless of statistical issues.

The black-box model structures reviewed earlier are all flexible enough to identify most reasonable systems in practice. On what concerns the nonlinear mapping from the regression vector to the output, (Juditsky et al., 1995) contains extensive discussions. Here we just mention some examples. It is well known that orthonormal wavelets form orthonormal basis of $L^2(\mathbf{R}^d)$ (Mallat, 1989; Daubechies, 1992). Several authors have shown that one-hidden-layer sigmoid network can approximate any continuous functions with an arbitrary accuracy, provided the number of basis functions used in the net is sufficiently large; and some error bounds are known (see, e.g. (Cybenko, 1989; Barron, 1993; Juditsky et al., 1995)). Similar results can be obtained for other one-hidden-layer networks, by using similar techniques.

6.4 Parameters Offered and Parameters Used

There is a natural way to approach the problem of minimizing (40) with respect to m: Try a sequence of models, with increasing m and estimate $V_*(m)$ either by testing the model on a validation data set, or by modifying the obtained loss for the estimation data in view of (43), (44). (The latter is the essence of the Akaike criteria AIC and FPE.)

In some simple model structures, there is a natural "ordering" of parameters. This is true for example for black-box models of single-input single-output dynamical systems: The model order serves as the ordering entity. For the non-linear black box models under discussion here, this is not the case. It is therefore not easy to carry out the mentioned program, without testing an astronomical amount of cases. This leads to the idea of "offering" the model structure a whole lot of parameters, and then try to decide which are the important – non spurious – ones, and then "use" only those. The number m in (40) should then correspond only to the number of actually used parameters. In this subsection we shall review some possibilities to achieve that feature.

Regularization: Pull towards the origin

One common and useful technique to distinguish between more and less "important" parameters is to add a penalty term to the criterion (34):

$$W_N(\theta, Z_e^N) = V_N(\theta_N, Z_e^N) + \delta\|\theta\|^2 \tag{45}$$

where δ is a small number. Intuitively, the idea is that a parameter that does not influence the first term very much will be kept close to zero by the second term. A parameter that is important for the model fit will however not be very much affected by the second term. Suppose we minimize (45) instead of (34), Then it can be shown, see e.g. (Sjöberg and Ljung, 1992) or (Moody, 1992), that (42) will still hold, with the important change that the number m is reduced to

$$r(m, \delta) = \sum_{k=1}^{m} \frac{\sigma_i^2}{(\sigma_i + \delta)^2} \tag{46}$$

where σ_i are the eigenvalues (singular values) of $\bar{V}''(\theta,)$, the second derivative matrix (the Hessian) of the criterion (36).

How to interpret (46)? A redundant parameter will lead to a zero eigenvalue of the Hessian. A small eigenvalue of V'' can thus be interpreted as corresponding to a parameter (combination) that is not so essential: "A spurious parameter". The regularization parameter δ is thus a threshold for spurious parameters. Since the eigenvalues σ_i often are widely spread (see (Saarinen et al., 1993) for the neural network case) we have

$$\begin{aligned} r(m, \delta) \;\; &\simeq \;\; m^{\#} = \# \text{ of eigenvalues of } V'' \\ &\qquad\quad \text{that are larger than } \; \delta \end{aligned}$$

We can think of $m^{\#}$ as "the efficient number of parameters in the parameterization". Regularization thus decreases the variance, but typically increases the bias contribution to the total error.

The parameter δ in (45) acts like a knob that affects the "efficient number of parameters used". It thus plays a similar role as the model size:

- Large δ: small model structure, small variance, large bias
- Small δ: large model structure, large variance, small bias

All this means that we can "offer" a large number of parameters for the fit, and then use δ in (45) to tune in the actual number $m^{\#}$ of "used" parameters. The tuning can be done by checking the model's prediction performance when applying it to a validation data set.

The added regularization term $\delta\|\theta\|^2$ in (45) can be changed to

$$W_N(\theta, Z_e^N) = V_N(\theta, Z_e^N) + \delta\|\theta - \theta^{\#}\|^2 \tag{47}$$

without changing the beneficial effects on the variance error. This penalty term corresponds to a prior Gaussian distribution for the parameters, viz that they have mean $\theta^{\#}$ and covariance matrix $2/\delta I$. In (MacKay, 1992) a Bayesian approach is introduced where the parameters may belong to different Gaussian distributions. This means that the spurious parameters can be excluded from the fit by associating them with a large prior at the same time as the important parameters, connected to a small prior, receive only a small bias. The additional Gaussian distributions describing the parameters can be estimated together with all other parameters. This is also described in (MacKay, 1991).

Regularization can also be used to include prior knowledge in the black-box model. Instead of penalizing the size of the parameters as in (47) one can add a complexity term which penalizes the distance to some nominal model. In (Suykens et al., 1994) an example of this approach is given.

Omitting basis functions

An alternative way to find the important parameters is to select the regressors to be used carefully, guided by the data. This is a classical topic in statistical regression and we shall review such techniques in Section 8.

A variant of this is *shrinking*. This means that components of $\hat{\theta}_N$ that are below a certain "noise level" are set to zero or pulled towards zero using a soft threshold. (The relationship to regularization is obvious.) This reduces variances without significantly changing the bias. The difficulty is to know or estimate this "noise level". This is also discussed in subsection 8.1 and extensively in (Juditsky et al., 1995) for the case of wavelets, where most spectacular results are obtained.

The equivalent of shrinking in connection with neural nets is called *pruning* and it has attracted much interest lately. See e.g., (Reed, 1993) for an overview and further references therein. In pruning, in difference from shrinking, also the dilation parameters are considered and possibly deleted.

7 Estimation Algorithms: Optimization Methods

In this section and the next one, we review methods for parameter estimation, i.e., for a given number n of chosen basis functions g_k, we deal with issues on how to estimate unknown parameters in the model.

If all the components of θ are "unknown", a basic approach is to minimize $V_N(\theta)$ as defined in (34) with respect to all the parameters. First a short review of algorithms for minimizing $V_N(\theta)$ are given and then some topics connected to this minimization will be discussed.

7.1 Methods of minimization

The Criterion

Given a scalar valued criterion like (34) then the parameter estimate is defined as the minimizing argument:

$$\widehat{\theta}_N = \arg\min V_N(\theta) \tag{48}$$

The estimate of the unknown function will then be

$$\widehat{g}_N(\varphi) = g(\varphi, \widehat{\theta}_N) \tag{49}$$

Sometimes a general, non-quadratic, norm is used instead of (34)

$$V_N(\theta) = \frac{1}{N}\sum_{t=1}^{N} \ell(\varepsilon(t,\theta)) \tag{50}$$

$$\varepsilon(t,\theta) = y(t) - g(\varphi(t),\theta)$$

The estimate $\widehat{\theta}_N$ is the maximum likelihood estimate for a specific noise assumption which depends on the choice of norm. The quadratic norm, *e.g.*, corresponds to the assumption of white Gaussian noise.

Entropy Interpretation

When probabilities are being estimated, *e.g.*, in classification problems, then it is common to choose a criterion based on the *relative entropy*. See, *e.g.*, (Cover and Thomas, 1991). This gives the maximum likelihood estimate of the probability (Baum and Wilczek, 1988). The relative entropy is defined

$$\text{Entropy} = p(\varphi)\cdot\log\left(\frac{p(\varphi)}{\widehat{p}(\varphi)}\right) \tag{51}$$

where $\widehat{p}(\varphi) = \widehat{p}(\varphi,\theta)$ and $p(\varphi)$ are the estimated and true probability for φ belonging to class $\mathcal{C}$. The entropy is non-negative and it is zero only if $\widehat{p}(\varphi) = p(\varphi)$.

Removing the parameter independent terms from (51) gives $-p(\varphi)\cdot\log(\widehat{p}(\varphi))$ which is the expectation value of $-\log(\widehat{p}(\varphi))$. If this expectation value is replaced by the sample mean one obtain the criterion

$$V_N(\theta) = -\sum_{\varphi\in\mathcal{C}} \log(\widehat{p}(\varphi)) \tag{52}$$

If several probabilities are considered at the same time, *e.g.*, in a two-class problem, then the criterion becomes a sum of terms like (52).

Nonlinear Optimization Methods

In general, the minimum of $V_N(\theta)$ cannot be computed analytically, so the minimization has to be done by some numerical search procedure. This is called *nonlinear optimization* and a classical treatment of the problem of how to minimize sum of squares is given in (Dennis and

Schnabel, 1983). A survey of methods for the NN application is given in (Kung, 1993) and in (van der Smagt, 1994).

Generally speaking, the numerical minimization of criteria of fit for identification purposes is a well established topic, and treated for general model structures, e.g., in (Ljung, 1987) and (Ljung and Glad, 1994). The general consensus is that one should use a damped Gauss-Newton algorithm with regularization features for ill conditioned Hessians – all of this to be defined shortly – in an off-line manner, unless the application demands on-line (recursive) algorithms. Given this, a surprising amount of applications in the Neural Network area have used gradient search in an on-line fashion. This has contributed to the popular opinion that Neural Networks require large amounts of time for their "training" (i.e. parameter estimation).

The Basic Search Algorithm

The discussion which follows is based on the quadratic norm (34), for other choices only minor modifications have to be done. Most efficient search routines are based on iterative local search in a "downhill" direction from the current point. We then have an iterative scheme of the following kind

$$\widehat{\theta}^{(i+1)} = \widehat{\theta}^{(i)} - \mu_i R_i^{-1} \nabla \widehat{f}_i \tag{53}$$

Here $\widehat{\theta}^{(i)}$ is the parameter estimate after iteration number i. The search scheme is thus made up from the three entities

- μ_i step size
- $\nabla \widehat{f}_i$ an estimate of the gradient $V_N'(\widehat{\theta}^{(i)})$
- R_i a matrix that modifies the search direction

It is useful to distinguish between two different minimization situations

(i) *Off-line* or *batch*: The update $\mu_i R_i^{-1} \nabla \widehat{f}_i$ is based on the whole available data record Z^N.

(ii) *On-line* or *recursive*: The update is based only on data up to sample i (Z^i), (typically done so that the gradient estimate $\nabla \widehat{f}_i$ is based only on data just before sample i.)

We shall concentrate on the off-line case below. For some general aspects on recursive techniques, we refer to (Sjöberg et al., 1994).

7.2 Search directions: Gradient and Newton

The basis for the local search is the gradient

$$V_N'(\theta) = -\frac{1}{N} \sum_{t=1}^{N} (y(t) - g(\varphi(t), \theta)) h(\varphi(t), \theta) \tag{54}$$

where

$$h(\varphi(t), \theta) = \frac{\partial}{\partial \theta} g(\varphi(t), \theta) \quad (m \times 1 - \text{vector}) \tag{55}$$

where $m = \dim \theta$. (Here we assume that y is scalar.) It is well known that gradient search for the minimum is inefficient, especially for ill-conditioned problems close to the minimum. Then it is optimal to use the *Newton search direction*

$$R^{-1}(\theta) V_N'(\theta) \tag{56}$$

where

$$R(\theta) = V_N''(\theta) = \frac{1}{N}\sum_{t=1}^{N} h(\varphi(t),\theta)h^T(\varphi(t),\theta) +$$

$$\frac{1}{N}\sum_{t=1}^{N}(y(t) - g(\varphi(t),\theta))\frac{\partial^2}{\partial\theta^2}g(\varphi(t),\theta) \tag{57}$$

The true Newton direction will thus require that the second derivative

$$\frac{\partial^2}{\partial\theta^2}g(\varphi(t),\theta)$$

is computed. Also, far from the minimum, $R(\theta)$ need not be positive semidefinite. Therefore alternative search directions are more common in practice:

- *Gradient direction.* Simply take

$$R_i = I \tag{58}$$

- *Gauss-Newton direction.* Use

$$R_i = H_i = \frac{1}{N}\sum_{t=1}^{N} h(\varphi(t),\hat{\theta}^{(i)})h^T(\varphi(t),\hat{\theta}^{(i)}) \tag{59}$$

- *Levenberg-Marquardt direction.* Use

$$R_i = H_i + \delta I \tag{60}$$

  where H_i is defined by (59) and δ may be used instead of a step size. A large δ gives a small step in the gradient direction and a small (zero) δ gives a Gauss-Newton step.

- *Conjugate gradient direction.* Construct the Newton direction from a sequence of gradient estimates. Loosely, think of V_N'' as constructed by difference approximation of d gradients. The direction (56) is however constructed directly, without explicitly forming and inverting V''.

It is generally considered, (Dennis and Schnabel, 1983), that the Gauss-Newton search direction is to be preferred. For ill-conditioned problems the Levenberg-Marquardt modification is recommended. The ideal step size μ in (53), would be $\mu = 1$, if the underlying criterion really was quadratic. What is typically done, is that several values of μ are tested (from 1 and down) until a new parameter value is found that gives a lower value of the criterion. This is what is referred to as the *damped Gauss-Newton method.*

However, good results with conjugate gradient methods have also been reported in NN applications (van der Smagt, 1994). Such methods where an approximation is used instead of the true Hessian are referred to as *Quasi-Newton* methods.

Equation (53) describes how the parameter update is done and this is the basic numerical method to find the minimum. The straight-forward approach is to estimate all parameter in each iteration. There also exist *two-stage* and *multi-stage* algorithms where only some of the parameters are updated in each iteration. By only considering a subset of the parameters the computational burden of each iteration becomes lower. This must, however, usually be compensated by a larger number of iterations. The advantage of this approach depends on the nature of the specific problem considered. For example, parameters connected to non-overlapping basis functions can be updated independent of each other.

A recent example of multi-stage methods is Breiman's algorithm for the parameter estimation in a hinging hyperplanes model (Breiman, 1993). Breiman suggests a scheme where only the parameters in connection with one hinge function should be updated in each iteration. At first, it is not at all obvious that the algorithm fall under the general description covered by (53) but it can be shown the algorithm is equivalent to a multi-stage Newton algorithm applied to a quadratic criterion. See (Pucar and Sjöberg, 1995a).

7.3 Back-Propagation: Calculation of the gradient

The only model-structure dependent quantity in the general scheme (53) is the gradient of the model structure (55). In connection with neural networks the celebrated *Back-Propagation Error algorithm*[3] (BP) is used to compute this gradient. Backpropagation has been described in several contexts, see *e.g.*, (Werbos, 1974; Rumelhart et al., 1986). For a one-hidden-layer sigmoid neural network, ((27)) it is straightforward to compute the gradient, since (omitting the subscript k)

$$\begin{aligned}
\frac{d}{d\alpha}\alpha g(\beta\varphi+\gamma) &= g(\beta\varphi+\gamma) \\
\frac{d}{d\gamma}\alpha g(\beta\varphi+\gamma) &= \alpha g'(\beta\varphi+\gamma) \\
\frac{d}{d\beta}\alpha g(\beta\varphi+\gamma) &= \alpha g'(\beta\varphi+\gamma)\varphi
\end{aligned}$$

where α and γ are scalars and β is a row vector. The BP algorithm in this case means that the factor $\alpha g'(\beta\varphi+\gamma)$ from the derivative with respect γ is re-used in the calculation of the derivative with respect to β.

The Back-Propagation algorithm is however very general and not limited to one-hidden-layer sigmoid neural network models. Instead, it applies to all network models and it can be described as the chain rule for differentiation applied to the expression (19) with a smart re-use of intermediate results which are needed at several places in the algorithm. For ridge construction models (27) where β_i is a parameter vector, *e.g.*, neural nets, the only complicated thing with the algorithm is actually to keep track of all indexes. When β_i is a parameter matrix, like in the wavelet model, then the calculation becomes somewhat more complicated, but the basic procedure remains the same.

When shifting to multi-layer network models the possibilities of re-using intermediate result increase and so does the importance of the BP algorithm. This can be described in an illustrative way, see Figure 4.

For recurrent models the calculation of the gradient becomes more complicated. The gradient $h(t)$ at one time instant does not only depend on the regressor $\varphi(t,\theta)$ but also on the gradient at the previous time instant $h(t-1)$. See (Nerrand et al., 1994) for a discussion on this topic. The additional problem to calculate the gradient does, however, not change anything essential in the minimization algorithm. In the neural network literature this is often referred to as *Back-propagation through time*.

7.4 Implicit Regularization, Stopped Iterations, and "Overtraining"

We have stated that the estimation of θ is performed by minimizing criterion (34). Then the iterations in the basic scheme (53) could be run until no further improvement in the fit can

[3] Sometimes in the NN literature the entire search algorithm is called Back-Propagation. It is, however, more consistent to keep this notation just for the algorithm used to calculate the gradient.

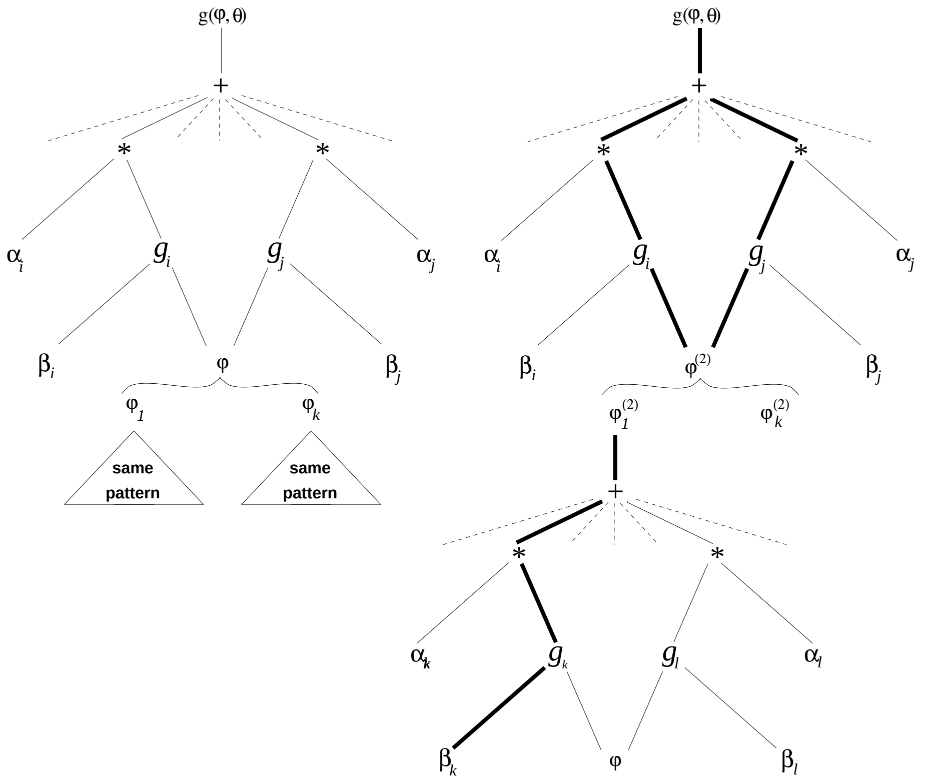

Figure 4: *Illustrating the backpropagation.* The figure shows two graphs. The graph on the left encodes a formula in the way expressions are encoded into abstract graphs in syntactic analyzers in computer science : for instance the graph above encodes $g(\varphi, \theta) = \sum_i \alpha_i * g_i(\varphi, \beta_i)$ (where $*$ denotes multiplication), *i.e.*, the general one-layer network formula (19). Nodes of this graph can be interpreted both as operators ($e.g.$, $+, *, g_i$) and as the evaluation of the expression encoded by the subtree located below this node. The triangle "same pattern" indicates that each component φ_l of φ could be the result of evaluating again the same function encoded by the same graph. This would immediately encode a two– and then multi–layer network. On the right hand side, we have expanded once the "same pattern", showing the case of two layers. We have shown in thick the paths linking the root $g(\varphi, \theta)$ to one particular parameter, say β_k. The semantic of the thickening is the following : each node on a thick path which is an operator ($e.g.$, $+, *, g_i$) is replaced by its partial derivative with respect to the node just below it on the thick path (here, we regard such node as the evaluation of the expression encoded by the subtree located below it). For instance, in the figure, g_i is replaced by $\partial g_i / \partial \varphi^{(2)}$ and g_k by $\partial g_k / \partial \beta_k$. By the chain rule for differentiation, it turns out that the graph on the right hand side encodes the partial derivative $\partial g(\varphi, \theta) / \partial \beta_k$! This graphical representation also explains why intermediate calculations can be shared for different partial derivatives, and this is indeed the nice feature of "backpropagating" the gradient. This presentation is due to G. Cybenko, (Saarinen et al., 1993).

be found, i.e. until a (local) minimum of V_N has been reached. However, it was noted early on in the neural network literature that if the model was evaluated on validation data, it first improved with the number of iterations, but then started to deteriorate with increasing number of iterations (despite the fact that the value of V_N, based on the estimation data of course continued to improve). This phenomenon was termed *overtraining*.

The effect can be explained as follows in words (see (Wahba, 1987) and (Sjöberg and Ljung,

1992) for more formal treatments): Suppose that the iterative search in (53) is started at $\hat{\theta}^{(0)} = \theta^{\#}$. The iterations will then pull the parameters towards the minimizing values. The parameters that have a substantial influence on the fit will feel a stronger pull, and will be adjusted quicker than the less important parameters. If the iterations are aborted before V_N has been minimized, the less important parameters will thus hang on around the initial value $\theta^{\#}$. This is pretty much the same result as if we had minimized the regularized criterion (47). Increasing the number of iterations i thus corresponds to decreasing the regularization parameter δ.

More precisely, the link is as follows (when quadratic approximations are applicable)

$$(I - \mu R^{-1} V'')^i \sim \delta(\delta I + V'')^{-1}$$

so, as the iteration number increases, this corresponds to a regularization parameter that decreases to zero as

$$\log \delta \sim -i \tag{61}$$

An increasing number of iterations is therefore equivalent to *a larger model structure – more "used" parameters*. The concept of overtraining is consequently just a reflection of the well known concept of *overfit*, defined in Section 6.4.

How to know when to stop the iterations? As $i \to \infty$ the value of the criterion V_N will of course continue to decrease, but as a certain point the corresponding regularization parameter becomes so small that increased variance starts to dominate over decreased bias. This should be visible when the model is tested on a fresh set – the *Validation data or Generalization data*. We thus evaluate the criterion function on this fresh data set, and plot the fit as a function of the iteration number. A typical such plot is shown in Figure 8 (Section 10.1). This method of terminating the iterations when the model fit (evaluated for the validation data) starts to increase will be called *stopped search*.

Regularization implemented as stopped search is called *implicit regularization* in contrast to the *explicit regularization* which is obtained by minimizing the modified criterion (45).

7.5 Local Minima

A fundamental problem with minimization tasks like (34) is that $V_N(\theta)$ may have several or many local (non-global) minima, where local search algorithms may get caught. There is no easy solution to this problem. It is usually well used effort to spend some time to come up with a good initial value $\hat{\theta}^{(0)}$ where to start the iterations. This is however not a realistic option in most nonlinear black-box problems where little *prior* knowledge is available. The best thing in such cases is usually to choose $\hat{\theta}^{(0)}$ by random in such a way that the support of the basis functions covers the interesting domain of the input space. Model structures using constructive estimation methods give some more options which are described in Section 8.

Other than that, only various global search strategies are left, such as random search, random restarts, simulated annealing and genetic algorithms.

8 Estimation Algorithms: Constructive Methods

Recall that in our general model structure (32), the total parameter vector θ is composed of two different parts, the coordinate parameters α_k on the one hand, and the dilation and location parameters (β_k, γ_k) on the other. For fixed parameters (β_k, γ_k), minimizing (34) (θ collects all the α_k in this case) is a linear least squares problem. Such problems are very "nice" in that efficient algorithms exist, no search has to be performed, and there is no problem with local minima. This assumes that the values of (β_k, γ_k) (thus the parameterized basis functions) have been "chosen" in some efficient manner. This approach is feasible only with some particular basis functions which come with natural choice of the values of (β_k, γ_k). For instance, wavelets are very well suitable for applying such approach. In fact, even in such situations, the choice of (β_k, γ_k) is often partially influenced by the observed data. For the algorithms considered in this section, data are used for selecting the values of these parameters from some practically finite set. This finite set of the values depends on the chosen basis functions and possibly on prior knowledge on the application. We refer to such approaches for model estimation as *constructive methods*.

Wavelets play an important role for constructive methods, so this section is mainly concentrated to wavelet based models.

8.1 Orthonormal wavelet decomposition and shrinking algorithms

Wavelets are a very interesting class of functions due to their special properties. In this subsection we first introduce some basic concepts about orthonormal wavelet basis, then we describe the wavelet shrinking techniques which make wavelets powerful nonlinear estimators.

Orthonormal bases of wavelets

Multiresolution analysis introduced by Yves Meyer and Stephane Mallat and further developed by Ingrid Daubechies provides orthonormal bases of $L_2(\mathbf{R})$ of the form[4] $\psi_{j,k}(\varphi) = \{2^{j/2}\psi(2^j\varphi - k) \;:\; j,k \in \mathbf{Z}\}$, i.e., each element of the basis is a translated and dilated version of a single *mother wavelet* ψ. *For the time being, let us consider only scalar* $\varphi \in \mathbf{R}$. For a function $f \in L_2(\mathbf{R})$, the inner product $\langle f, \psi_{j,k}\rangle$ performs zooming on f over a $O(2^{-j})$ width interval centered at point $2^{-j}k$. Thus, *large j corresponds to checking function f at fine scales.* This implies that a local singularity of a function f will affect only a small part of its coefficients in this wavelet basis. This is the main difference with the Fourier basis: a local singularity of f would affect the whole Fourier representation. Thus, using this basis, each $f \in L_2(\mathbf{R})$ is expanded into[5]

$$f \;=\; \sum_{j,k\in\mathbf{Z}} \alpha^*_{jk}\; \psi_{jk} \;,\; \text{ where } \alpha^*_{jk} = \langle f, \psi_{jk}\rangle \;,$$

i.e., $L_2(\mathbf{R})$ is decomposed into the doubly infinite orthogonal sum $L_2(\mathbf{R}) = \bigoplus_{j\in\mathbf{Z}} W_j$, where $W_j = \operatorname{span}\{\psi_{j,k}, k \in \mathbf{Z}\}$. In this expansion, j is the scale index, which ranges from infinitely coarse up to infinitely fine, and k is the translation index. Now, it is often useful in practice not to consider this double-sided expansion, but to use instead a one-sided expansion where all scales $j < 0$ are collapsed into a single basic "low resolution" subspace of L^2, i.e., we set $V_0 = \bigoplus_{j<0} W_j$. This can be achieved by associating to the mother wavelet ψ a so-called "father wavelet" (also termed "scale function") ϕ, whose translated versions suffice to span all scales $j < 0$. Thus the expansion we shall actually work with has the form

$$f \;=\; \underbrace{\sum_{k\in\mathbf{Z}} \alpha_{0k}\; \phi_{0k}}_{\text{“zero scale”}\, V_0} + \underbrace{\sum_{j\geq 0,k\in\mathbf{Z}} \alpha^*_{jk}\; \psi_{jk}}_{\text{“finer scales”}\, \bigoplus_{j\geq 0} W_j} \quad,\; \text{where}$$

[4]In the wavelet literature, wavelets are usually considered as functions of x and noted as $\psi(x)$. Here we consider wavelets as particular basis functions and use $\psi(\varphi)$ to denote wavelets, functions of the regression vector φ.

[5]The wavelet coefficient $\langle f, \psi_{j,k}\rangle$ is usually denoted by β_{jk}. Here we use α^*_{jk} in order to keep consistent our notations α and β. Note that $\langle .,.\rangle$ denotes the inner product in L_2.

$$
\begin{aligned}
\psi_{j,k}(\varphi) &= 2^{j/2}\psi(2^j\varphi - k)\ ,\ \phi_{0,k}(\varphi) \ = \ \phi(\varphi - k) \\
\alpha^*_{jk} &= \langle f, \psi_{j,k}\rangle\ ,\ \alpha_{0k} \ = \ \langle f, \phi_{0,k}\rangle\ ,
\end{aligned} \tag{62}
$$

and (62) is an orthonormal expansion. We refer the reader to (Juditsky et al., 1995) for a more formal introduction of this matter, as well as the discussion of smoothness properties of so constructed wavelets. Remark that the dilation parameter 2^j and translation parameter k of a wavelet correspond to the β and γ parameters of our generic "mother basis function" as first introduced in (20).

The very strong point of such orthonormal wavelet expansions is that *coarse scale coefficients can be recursively computed from fine scale ones, and vice-versa.* Let us explain this. If $g(\varphi) \in \bigoplus_{j<j_o} W_j$, then clearly $g(2\varphi) \in \bigoplus_{j<j_o+1} W_j$. Hence, since the family $\{\phi_{0k}\}$ spans V_0, then $\{\phi(2\varphi - k)\}$ spans the *next* finer scale $V_0 \oplus W_0$, i.e., we have

$$
\begin{aligned}
\phi(\varphi) &= \sqrt{2}\sum_k h_k\ \phi(2\varphi - k) \\
\psi(\varphi) &= \sqrt{2}\sum_k g_k\ \phi(2\varphi - k)
\end{aligned} \tag{63}
$$

for suitable (h_k) and (g_k). Equations (63) imply that, for $f \in L_2(\mathbf{R})$,

$$
\alpha_{jk} \ = \ \langle f, \phi_{jk}\rangle, \quad \alpha^*_{jk} = \langle f, \psi_{jk}\rangle \tag{64}
$$

obey the following *fine-to-coarse recursions*

$$
\alpha_{jk} \ = \ \sum_l h_{l-2k}\ \alpha_{j+1,l} \tag{65}
$$

$$
\alpha^*_{jk} \ = \ \sum_l g_{l-2k}\ \alpha_{j+1,l}. \tag{66}
$$

Recursions (65) and (66) are used to compute recursively from fine scales to coarse scales the orthonormal wavelet decomposition, with α_{j_0k} as initial condition (index "j_0" denotes the finest scale in these recursions). Assume that, in addition, the scale function ϕ is selected so that the computation of inner product $\langle f, \phi_{j_0k}\rangle$ in (64) can be efficiently performed. Then, *formulas (64), (65), and (66) together build a highly efficient procedure for computing the wavelet decomposition of f,* see (Juditsky et al., 1994) for efficient computation of inner product $\langle f, \phi_{jk}\rangle$. In addition, equations (65) and (66) can be inverted to yield the *coarse-to-fine recursion*

$$
\alpha_{jk} \ = \ \sum_l h_{k-2l}\ \alpha_{j-1,l} + g_{k-2l}\ \alpha^*_{j-1,l} \tag{67}
$$

For $f \in V_{j_0}$, we have, by definition of this space,

$$
f = \sum_k \alpha_{j_0k}\ \phi_{j_0k}\ , \tag{68}
$$

and, for $j_0 > 0$, since $V_{j_0} = V_0 \bigoplus W_0 \bigoplus W_1 .... \bigoplus W_{j_0-1}$,

$$
f = \sum_k \alpha_{0k}\ \phi_{0k} + \sum_{0\le j<j_0,k} \alpha^*_{jk}\ \psi_{jk}\ . \tag{69}
$$

Formulas (65) and (66) allow us to switch from representation (68) to representation (69). The latter one is generally much more compact since, when f is smooth, most α^*_{jk} are negligible.

We now move on discussing the multidimensional case. There exist two main types of constructions of the wavelet basis with dilation factor 2 in $\mathbf{R}^d$ ((Daubechies, 1992), 10.1). A first guess simply consists in taking tensor product functions generated by d one-dimensional bases:

$$\Psi_{j_1,k_1,\ldots,j_d,k_d}(\varphi) = \psi_{j_1,k_1}(\varphi_1) \times \ldots \times \psi_{j_d,k_d}(\varphi_d), \quad \varphi = (\varphi_1, \ldots, \varphi_d)^T \in \mathbf{R}^d \ . \tag{70}$$

This construction has the drawback of mixing different resolution levels j_i. Alternatively, if such a mixing is not desired, we proceed as follows. Introduce the scale function

$$\Phi(\varphi) \quad = \quad \phi(\varphi_1) \times \ldots \times \phi(\varphi_d) \tag{71}$$

and the $2^d - 1$ mother wavelets $\Psi^{(i)}(\varphi)\,, i = 1, \ldots, 2^d - 1$ obtained by substituting in (71) some $\phi(\varphi_j)$'s by $\psi(\varphi_j)$'s. Then the following family is an orthonormal basis of $L_2(\mathbf{R}^d)$:

$$\begin{gathered}\left\{\Phi_{0k}(\varphi), \Psi^{(1)}_{jk}(\varphi), \ldots, \Psi^{(2^d-1)}_{jk}(\varphi)\right\} \\ j \in \mathbf{N}_0, \ k = (k_1, \ldots, k_d) \in \mathbf{Z}^d\end{gathered} \tag{72}$$

where $\mathbf{N}_0 = \mathbf{N} \cup 0$, and

$$\begin{aligned}\Phi_{jk}(\varphi) \quad &= \quad 2^{jd/2} \ \Phi(2^j\varphi_1 - k_1, \ldots, 2^j\varphi_d - k_d) \\ \Psi^{(i)}_{jk}(\varphi) \quad &= \quad 2^{jd/2} \ \Psi^{(i)}(2^j\varphi_1 - k_1, \ldots, 2^j\varphi_d - k_d) \ .\end{aligned}$$

NOTE: As formula (72) shows, constructing and storing orthonormal wavelet bases become of prohibitive cost for large dimension d. This is the main limitation for using the otherwise very efficient techniques which rely on orthonormal wavelet bases (and their generalizations).

Wavelet shrinking algorithm

Assume that a N-sample of estimation data is available:

$$\{(y(t), \varphi(t)) \ : \ y(t) = g_0(\varphi(t)) + e(t), \ t = 1, \ldots, N\}$$

where g_0 is some unknown "true model", $\varphi(t)$ and $e(t)$, $t = 1, \ldots, N$, are i.i.d. sequences of random variables, and $Ee(t) = 0$, $Ee^2(t) = \lambda$. We assume for the time being that $\varphi(t)$ is *uniformly distributed* on $[0,1]^d$. For $g_0 \in L_2$, recall the (multidimensional) wavelet expansion

$$g_0(\varphi) \quad = \quad \sum_{k\in\mathbf{Z}} \alpha_{0k} \ \Phi_{0k}(\varphi) + \sum_{j=0}^{\infty} \sum_{k\in\mathbf{Z}^d} \sum_{l=1}^{2^d-1} \alpha^{*(l)}_{jk} \ \Psi^{(l)}_{jk}(\varphi) \ , \tag{73}$$

where

$$\alpha_{0k} = \int g_0(\varphi)\Phi_{0k}(\varphi)d\varphi \quad \text{and} \quad \alpha^{*(l)}_{jk} = \int g_0(\varphi)\Psi^{(l)}_{jk}(\varphi)d\varphi \ . \tag{74}$$

To construct an estimate of g_0, a first idea consists in using the law of large numbers and replacing, in expansion (73), the coefficients α_k and $\alpha^{*(l)}_{jk}$ by their empirical estimates

$$\widehat{\alpha}_{0k}(N) = \frac{1}{N}\sum_{t=1}^{N} y(t)\Phi_{0k}(\varphi(t)) \quad \text{and} \quad \widehat{\alpha^*}^{(l)}_{jk}(N) = \frac{1}{N}\sum_{t=1}^{N} y(t)\Psi^{(l)}_{jk}(\varphi(t)) \ . \tag{75}$$

Note that the assumption that $\varphi(t)$ is uniformly distributed has been used at this point. This brute-force estimate is impractical in many points: $\varphi(t)$ is not gently uniformly distributed in real life, most of the $\Psi_{jk}^{(l)}$'s would have just no $\varphi(t)$ sample in their support so that empirical averages $\widehat{\alpha^*}_{jk}^{(l)}(N)$ are not defined, and, finally, most of the remaining $\widehat{\alpha^*}_{jk}^{(l)}(N)$ would not be significantly different from "noise level" and should probably be discarded. All these issues are addressed in the following procedure (see (Juditsky et al., 1995) for its mathematical justification):

1. **Select relevant scales.** Obviously, in order to compute the empirical coefficient $\widehat{\alpha^*}_{jk}^{(l)}$, we need that at least several observations $\varphi(t)$ hit the support of $\Psi_{jk}^{(l)}(\varphi)$. Statistical laws of *loglog* type guarantee that this would generically hold for scales that are not too fine, more specifically, for $j \leq j_{\max}$, where

$$\frac{N}{\ln N} \leq 2^{d\, j_{\max}} \leq \frac{2N}{\ln N}$$

   Thus we brute-force set $\widehat{\alpha^*}_{jk}^{(l)} \equiv 0$ for $j > j_{\max}$. Note that we did not use the assumption that $\varphi(t)$ is uniformly distributed at this point.

2. **Collapse data into a synthetic regularly sampled record.** Assuming that $\varphi(t)$ has a smooth enough density, we shall approximate it by a constant over each bin

$$\Delta_k = \left[2^{-j_{\max}} k_1, 2^{-j_{\max}}(k_1+1)\right] \times \ldots \times \left[2^{-j_{\max}} k_d, 2^{-j_{\max}}(k_d+1)\right]$$

   of length $2^{-d\, j_{\max}}$. Then, since we (statistically) know that each such bin has enough data, we can (very coarsely) collapse the data within each bin into a single representative data point by taking simple averages, namely[6]

$$\widetilde{g}_0^{(N,k)} = \frac{\sum_{t=1}^{N} y(t)\ 1_{\{\varphi(t)\in\Delta_k\}}}{\sum_{t=1}^{N} 1_{\{\varphi(t)\in\Delta_k\}}}$$

   will be the synthetic output associated with the k-th bin Δ_k. Then we set

$$\widehat{\alpha}_{j_{\max},k} = 2^{d\, j_{\max}/2}\ \widetilde{g}_0^{(N,k)}$$

   i.e., we identify (up to the scaling factor) $\widetilde{g}_0^{(N,k)}$ with the father wavelet coefficients of our unknown function to be estimated, taken at the finest scale $j_{\max}$.

3. **Use fine-to-coarse recursions to get the wavelet expansion.** At this point we have constructed synthetic input-output pairs, where the input is the considered bin and output is the associated $\widehat{\alpha}_{j_{\max},k}$ estimate. Getting the full wavelet expansion is then performed by applying to these synthetic data the recursion formulae (65), (66). Use the multi-dimensional version of filters (65), (66) to compute $\widehat{\alpha}_{jk}$, $\widehat{\alpha^*}_{jk}^{(l)}$, $j = 0,\ldots,j_{\max}-1$, $l = 1,\ldots,2^d-1$:

$$\begin{aligned}
\widehat{\alpha}_{jk} &= \sum_i h_{i-2k}\ \widehat{\alpha}_{j+1,i} \\
\widehat{\alpha^*}_{jk}^{(l)} &= \sum_i g_{i-2k}^{l}\ \widehat{\alpha}_{j+1,i}\ .
\end{aligned}$$

[6] 1_A is the indicator function, i.e., 1_A equals 1 if A is true, 0 other wise.

4. **Shrink junk below noise level.** Now what we have at this point is an estimate of the full wavelet expansion of our unknown function, up to scale $j_{\max}$. Due to the local nature of wavelets, the expansion coefficients are significantly different from zero only for wavelets having significant variations of g_0 in their supports. Thus most of the coefficients in this expansion would basically contain only noise, with no relevant information. Since our wavelet basis is orthonormal, it can be shown that significance of wavelet coefficients can be tested *separately for each coefficient,* by comparing them to suitably selected thresholds, cf. (Juditsky et al., 1995). Thus we shrink the estimates $\widehat{\alpha^*}_{jk}^{(l)}$ according to

$$\widetilde{\alpha^*}_{jk}^{(l)} = \widehat{\alpha^*}_{jk}^{(l)} \, 1_{\{|\widehat{\alpha^*}_{jk}^{(l)}| \geq \lambda_j\}} ,$$

where λ_j is a properly selected threshold.

5. **Use coarse-to-fine recursions to reconstruct the finest scale.** We are now ready to use the "inverse" filter (67) to obtain $\widetilde{\alpha}_{j_{\max},k}$:

$$\widetilde{\alpha}_{j_{\max}k} = \sum_{i,l} h_{k-2i} \, \widetilde{\alpha}_{j-1,i} + g^l_{k-2i} \, \widetilde{\alpha^*}_{j-1,i}^{(l)} . \tag{76}$$

and we finally set

$$\widehat{g}_0^{(N)}(2^{-j_{\max}}k) \triangleq \widetilde{g}_0^{(N,k)} = 2^{-dj_{\max}/2} \, \widetilde{\alpha}_{j_{\max}k} .$$

Steps 1—5 above constitute our algorithm. Note that most of its computational burden is concentrated into the fine-to-coarse and coarse-to-fine recursions, for which packages are available (see, e.g. (Taswell, 1993)). Altogether, this is an extremely efficient algorithm for small dimensions (typically $d \leq 3$).

8.2 Techniques of basis functions selection

Orthonormal wavelet bases are really a very nice and restricted class of basis functions related to very fast estimation algorithms and efficient shrinking algorithms. However, they are practically applicable only to problems with a small number of regressors and reasonably distributed data. In this subsection, we introduce the techniques of basis function selection, applicable to a less restricted class of basis functions including non orthogonal wavelets. These techniques can handle applications with a moderately large number of regressors and sparse data.

With these techniques we shall be able to, based on observed data, select the values of β_k and γ_k (typically they are dilation and location parameters) from a finite set $\mathcal{B}$, or equivalently, to select basis functions $g_k(\varphi, \beta_k, \gamma_k)$ from a finite set of basis functions:

$$\mathcal{G} = \{g_k(\varphi, \beta_k, \gamma_k) \, : \, (\beta_k, \gamma_k) \in \mathcal{B}\} . \tag{77}$$

We shall refer to $\mathcal{G}$ as the *basis function set* in the following.

We first discuss how to construct the basis function set $\mathcal{G}$ before introducing the techniques of basis function selection.

Construction of the basis function set

For simplicity, we consider the case where the basis functions are parameterized versions of a single "mother basis function" κ, i.e., $g_k(\varphi, \beta_k, \gamma_k) = \kappa(\varphi, \beta_k, \gamma_k)$. The construction of $\mathcal{G}$ depends on the form of κ. Typically β_k and γ_k correspond to the dilation and translation parameters respectively; and the model is only to be estimated in some finite domain of the regression vector φ, up to some resolution level. This can suggest the choice of $\mathcal{B}$. Below we discuss three typical examples.

One hidden layer sigmoid networks, for which $g_k(\varphi) = \sigma(\beta_k \varphi + \gamma_k)$. Parameters β_k, γ_k should be chosen so that the non-flat part of $\sigma(\beta_k \varphi + \gamma_k)$ stays inside the domain of interest, and the values of (β_k, γ_k) are well "distributed". However, there is no clear idea for this "distribution". For this reason, it seems that the basis function selection techniques are *not* well suited for sigmoid networks.

Radial Basis Function (RBF) networks, for which $g_k(\varphi) = r(\beta_k(\varphi - \gamma_k))$ where r is a radial function. There are two possibilities for choosing the values of γ_k (the centers of the RBFs): take the values of γ_k on a uniform lattice in the regression vector space, or let the values of γ_k equal to the "observed" values of the regression vector. It is more difficult to choose the values of β_k. Adaptive clustering or vector quantization techniques can be used for this purpose (Poggio and Girosi, 1990).

Wavelet networks, for which $g(\varphi) = \psi(\beta_k(\varphi - \gamma_k))$, ψ is a wavelet function. The choice of β_k and γ_k (the dilation and translation parameters) is very well suggested by the wavelet transform. Typically, the values of β_k and γ_k form a regular lattice, as in wavelet bases and frames.

After $\mathcal{B}$ is chosen, the corresponding basis function set $\mathcal{G}$ is given by (77).

The construction of $\mathcal{G}$ may have some practical limitations when the dimension d of the regression vector φ is large, since typically the size of $\mathcal{G}$ increases exponentially with d. However, for applications of large regressor dimension, the estimation data are often sparse in the space of regression vector. This particularity of the data should be taken into account for the construction of $\mathcal{G}$. For instance, if the basis functions are generated from a local "mother basis function" $\kappa(\cdot)$, many basis functions in $\mathcal{G}$ constructed in some regular way do not contain any (or contain few) estimation data in their effective support[7]. Such basis functions can be immediately rejected. This will limit the size of $\mathcal{G}$.

Basis function selection algorithms

Assume that the basis function set $\mathcal{G}$ is chosen. Now the problem is, given a set of estimation data as defined in (33), how to select n basis functions from $\mathcal{G}$. This is a classical problem in regression analysis (Draper and Smith, 1981). For a given value of n, selecting n optimal basis functions could be in principle performed via exhaustive search that would consist in examining all the possible combinations of n basis functions from $\mathcal{G}$. The number of all the possible combinations is usually very large.

[7] The term "effective support" is used instead of "support" to deal with the case of non compactly supported basis functions.

Some special constructions of $\mathcal{G}$ result in *orthogonal* basis functions. In such situations, the basis function selection problem can be solved in a very efficient way. The wavelet shrinking algorithm described in subsection 8.1 is a very spectacular example. Even when the basis functions are not strictly orthogonal, but close to orthogonal, applying the shrinking technique can also give reasonable results. The near-tight wavelet frames (Daubechies, 1990) are typical examples of such almost orthogonal basis functions.

In the general case where the basis functions in $\mathcal{G}$ are not orthogonal, in order to overcome the combinatorial complexity of the exhaustive search, three different heuristics are reviewed in the following, details of these algorithms can be found in (Zhang, 1994).

The residual based selection (RBS). The idea of this method is to select, for the first stage, the basis function in $\mathcal{G}$ that best fits the estimation data, then repeatedly select the basis function from the remainder of $\mathcal{G}$ that best fits the residual of the previous fitting. In the literature of classical regression analysis, this method is referred to as *stagewise regression procedure.* See, for example, (Draper and Smith, 1981). Recently it has been used in the matching pursuit algorithm of (Mallat and Zhang, 1993) and the adaptive signal representation of (Qian and Chen, 1994).

Stepwise selection by orthogonalization (SSO). The RBS method does not explicitly consider the non-orthogonality of the basis functions in $\mathcal{G}$. The idea of this alternative method is to select, for the first stage, the basis function in $\mathcal{G}$ that best fits the estimation data, then repeatedly select the basis function from the remainder of $\mathcal{G}$ that best fits the estimation data *while combining with the previously selected basis functions.* For computational efficiency, later selected basis functions are orthogonalized to earlier selected ones. It has been used in radial basis function (RBF) networks and other nonlinear modeling problems in (Chen et al., 1989; Chen et al., 1991).

Backward elimination (BE). In contrast to the previous two methods, the backward elimination method starts by building the model using all the basis functions in $\mathcal{G}$, then eliminates one basis function per stage, while trying to deteriorate the model fit as little as possible. A recursive scheme between the elimination stages can be used to reduce the computational cost. This method is computationally expensive when $\mathcal{G}$ is large.

Continuous wavelet transform in combination with basis function selection

Applying the above mentioned techniques of basis function selection to non orthogonal wavelets yields an interesting family of models called *wavelet networks* (Zhang and Benveniste, 1992; Zhang, 1994). Though they are computationally less efficient than the wavelet shrinking algorithms in small dimensional case, they allow to handle problems of moderately large dimensions. The software package of wavelet networks in Matlab language is available via anonymous FTP (Zhang, 1993).

We need to recall some basic concepts of continuous wavelet transform at this point. We only consider radial wavelets here. The continuous wavelet transform and its inverse transform of a function f are respectively given by equations (79)–(80) below. These transforms use two functions $\psi(\varphi)$ and $\phi(\varphi) \in L_2(\mathbf{R}^d)$, both radial (i.e., depending only on $\|\varphi\|$, where $\|\cdot\|$ denotes the Euclidean norm in $\mathbf{R}^d$), known as the *synthesis and analysis wavelets.* More specifically, let

ψ and ϕ be radial functions satisfying

$$\forall \omega \in \mathbf{R}^d \ : \ \int_0^\infty a^{-1} \widehat{\psi}(a\omega)\widehat{\phi}(a\omega) da = 1 \ , \tag{78}$$

where $\widehat{\psi}(\omega)$ and $\widehat{\phi}(\omega)$ denote the Fourier transform of $\psi(\varphi)$ and $\phi(\varphi)$, respectively. Then for any function $f \in L_2(\mathbf{R}^d)$, the following formulae define an isometry between $L_2(\mathbf{R}^d)$ and a subspace of $L_2(\mathbf{R}^d \times \mathbf{R}_+)$ (Daubechies, 1992) :

$$u(a,t) = a^{d-1/2} \int f(\varphi) \ \phi(a(\varphi - t)) \ d\varphi \tag{79}$$

$$f(\varphi) = \int u(a,t) \ \psi(a(\varphi - t)) \ a^{d-1/2} \ da \ dt \tag{80}$$

where $a \in \mathbf{R}^+$ and $t \in \mathbf{R}^d$ are respectively the dilation and translation parameters.

As discussed in detail in (Delyon et al., 1995; Juditsky et al., 1995), the reconstruction formula (80) immediately explains why dilated/translated versions of the synthesis wavelet ψ are good candidate basis functions. Rewrite this formula as

$$\begin{aligned} f(\varphi) &= \int u(a,t) \ \psi(a(\varphi - t)) \ a^{d-1/2} \ da \ dt \\ &= \int a^{d/2} \ \psi(a(\varphi - t)) \ \mathrm{sign}(u(a,t)) \ a^{(d-1)/2} \ |u(a,t)| \ da \ dt \\ &= \frac{1}{C} \int a^{d/2} \ \psi(a(\varphi - t)) \ \mathrm{sign}(u(a,t)) \ w(a,t) \ da \ dt \end{aligned}$$

where we have renormalized $u(a,t)$ by a constant factor C so that the function $w(a,t) = Ca^{(d-1)/2} \ |u(a,t)|$ can be considered as a probability density function. Draw n independent random samples $(a_i, t_i)_{i=1,\ldots,n}$ with the density $w(a,t)$. Then we build

$$f_n(\varphi) = \frac{1}{n} \sum_{i=1}^{n} \ a_i^{d/2} \ \psi(a_i(\varphi - t_i)) \ \mathrm{sign}(u(a_i, t_i)) \ , \tag{81}$$

which, thanks to the law of large numbers, converges in L_2 to the true function f when $n \to \infty$. This justifies using dilated/translated versions of the synthesis wavelet ψ to build the basis function set $\mathcal{G}$. However, implementing the above procedure would require the estimation of the density function $w(a,t)$, which is computationally expensive.

Results reported in (Daubechies, 1990; Kugarajah and Zhang, 1995) about the so-called wavelet *frames* justify using wavelet families of the form $\{\psi(\alpha_0^j \varphi - k\beta_0) \ : \ j \in \mathbf{Z}, \ k \in \mathbf{Z}^d\}$. Therefore, in practice, wavelet *basis function sets* $\mathcal{G}$ (as introduced in (77)) are constructed from wavelet frames. Then, applying the algorithms of basis function selection yields wavelet networks (Zhang, 1994).

9 Encoding prior information via syntactic fuzzy models

We have claimed in Section 4 that fuzzy modeling can be seen as a particular choice of basis functions. We shall make this point more clear in this section. In addition, we discuss in detail what is the add-on provided by fuzzy modeling. We first introduce fuzzy models such as typically used in fuzzy control (Lee, 1990). Several presentations are possible, see for instance (Zadeh, 1994) (Takagi and Sugeno, 1985) (Sugeno and Yasukawa, 1993). The presentation we give here is slightly heterodox, but is simple and consistent.

9.1 Introduction to fuzzy logic

Fuzzy sets

Consider scalar input variables generically written as φ. A fuzzy set on $\mathbf{R}$ is defined by a linguistic label A, and its membership function $\mu_{\mathtt{A}} : \varphi \in \mathbf{R} \mapsto \mu_{\mathtt{A}}(\varphi) \in [0,1]$. The membership function $\mu_{\mathtt{A}}$ is the mathematical meaning of "fuzzy set A". Thus, for each actual value of φ, the statement "φ is A " has a value equal to $\mu_{\mathtt{A}}(\varphi)$, such statements are premises of so-called "fuzzy rules". A typical form of such statements is "φ is large". Be careful that this statement does not convey any information unless the membership function $\mu_{\mathtt{A}}$ of the fuzzy set large is specified. A frequently used form for membership functions is just a symmetric triangle, with parameterized width ("dilation") and location, as illustrated by figure 5.

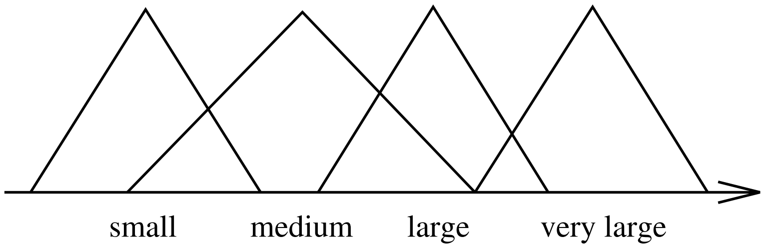

Figure 5: *Fuzzy sets.* This picture shows fuzzy membership functions corresponding to " small ", etc. The corresponding membership function is in fact of the form $\mu(\theta, \varphi)$, where θ is a parameter specifying the exact location, and width, within some parameterized family of basis functions.

Fuzzy operators

Fuzzy sets can be combined using the " and, or, not " operators of first order predicate logic. This allows to describe the combination of membership functions using *syntax*. For instance,

$$(\varphi_1 \texttt{ is } \mathtt{A}_1) \texttt{ and } (\varphi_2 \texttt{ is } \mathtt{A}_2\ ) \ \ldots \ \texttt{ and } (\varphi_{\mathtt{d}} \texttt{ is } \mathtt{A}_{\mathtt{d}})$$

is a fuzzy set involving the vector $(\varphi_1, \ldots, \varphi_{\mathtt{d}})$. Keyword " and " is a combinator of fuzzy sets which must be defined formally in terms of combination of membership functions. Similarly the operators " or " and " not " should be accordingly defined. Several choices have been proposed by various authors (Dubois and Prade, 1992), the most widely used ones are

$$\begin{aligned} \texttt{and}(u,v) = \min(u,v) \quad &, \quad \texttt{or}(u,v) = \max(u,v) \\ \texttt{and}(u,v) = uv \quad &, \quad \texttt{or}(u,v) = u+v-uv \\ \texttt{and}(u,v) = \max(0, u+v-1) \quad &, \quad \texttt{or}(u,v) = \min(1, u+v) \end{aligned} \tag{82}$$

(corresponding definitions for " and " and " or " are written on the same line) and

$$\texttt{not}(u) = 1 - u\ .$$

We try, now, to generalize the Boolean implication operator in continuous-valued logic[8]. As usual in logic, implication

[8]This is the point where we deviate from the usual presentation: in the fuzzy literature, implication is often encoded as a " and ", and the modus-ponens mechanism is modified accordingly. We preferred this presentation, since it is fully consistent and in accordance with the usual predicate calculus.

`(`φ` is A)  implies  (y is B)`

also written

`if `φ` is A   then   y is B`

is a macro which expands into

`(y is B)  or   not(`φ` is A)`

In the sequel, we shall encode the " `and` " as the product: $\mathtt{and}(u, v) = uv$, with corresponding codings for the " `not`, `or` ". Finally the implication is expanded as follows.

Denote by μ_A the membership function associated with fuzzy set `A` , and by $\mu_{A\Rightarrow B}$ the membership function of "`if` φ `is A then y is B` ". Using the formulas

$$(u \Rightarrow v) = (v \texttt{ or not } u) = v + (1-u) - v(1-u) = 1 - u + uv$$

we obtain an expression of implication (Reichenbach implication, in the literature):

$$\begin{aligned} \mu_{\mathtt{A}\Rightarrow\mathtt{B}}(\varphi, y) &= 1 - \mu_{\mathtt{A}}(\varphi) + \mu_{\mathtt{A}}(\varphi)\mu_{\mathtt{B}}(y) \\ &= 1 - \mu_{\mathtt{A}}(\varphi)\,(1 - \mu_{\mathtt{B}}(y)) \end{aligned} \tag{83}$$

This implication models a certainty rule of the form *the more* " φ `is A` " *the more certain* " `y is B` ", see (Dubois and Prade, 1992).

Fuzzy reasoning: modelling "fuzzy maps" via fuzzy rules

Fuzzy rules are statements of the form

`if `φ` is A  then  y is B`

Note that more complex premises can be used, using " `and`, `or`, `not` ". The *modus-ponens* is a mechanism in logic which maps predicates into predicates. Its counterpart in fuzzy logic allows an approximate application, so that conclusions to be drawn even the fact does not exactly agree with the the first part of the rule. It can be written

rule	:	`if` φ `is A   then`	`y is B`
fact	:	φ `is A'`	
conclusion	:		`y is B'`

Here, the fact "φ `is A'` " can be seen as the "input", the conclusion " `y is B'` " as the "output", and the fuzzy rule "`if` φ `is A then y is B` " is the "map". Thus, modus-ponens is a mechanism which combines membership functions and yields a membership function. A typical example could be that " φ `is A` " is " `The temperature is high` ", and that " φ `is A'` " corresponds to " `The temperature is very_high` ".

In the general case the mathematical translation of the fuzzy modus-ponens rule (Dubois and Prade, 1992) is defined as:

$$\mu_{\mathtt{B}'}(y) = \max_{\varphi} \{\mu_{\mathtt{A}'}(\varphi) \texttt{ and } \mu_{\mathtt{A}\Rightarrow\mathtt{B}}(\varphi, y)\} \tag{84}$$

where elimination of φ has been performed via maximization. Note that, in general, facts and conclusions are not ordinary numbers, but are rather fuzzy sets.

Inference with crisp inputs

Now we discuss the particular case of fuzzy reasoning with *crisp* (input) fact statement which is directly related to our general nonlinear black-box model formulation.

The fact `A'` in a statement "φ `is A'` " is crisp if the membership function of `A'` is such that $\mu_{\mathtt{A}'}(\varphi) = 1$ if $\varphi = \varphi_0$, and $\mu_{\mathtt{A}'}(\varphi) = 0$ otherwise, where φ_0 is an ordinary value. In this case, the modus-ponens mechanism (84) reduces to

$$\begin{aligned} \mu_{\mathtt{B}'}(y) &= \mu_{\mathtt{A}\Rightarrow\mathtt{B}}(\varphi_0, y) \\ \text{(by (83))} &= 1 - \mu_{\mathtt{A}}(\varphi_0)\,(1 - \mu_{\mathtt{B}}(y)) \ . \end{aligned} \tag{85}$$

Note that even in the case of crisp input fact, the conclusion `B'` is in general a fuzzy set.

9.2 Fuzzy Rule Bases as Models

Now, what does all this mean in a modeling/identification context? We shall first describe how sets of rules can be used to state the behavior of a system. The goal is to show the connection with more conventional models, like (3), and then also see how such fuzzy models can be parameterized like (14), (19).

Fuzzy Rule Bases

A "fuzzy rule basis" is a collection of fuzzy rules of the form, say,

$$\begin{array}{c} \mathtt{if}\ (\varphi_1\ \mathtt{is}\ \mathtt{A}_{1,1})\ \ldots\ \mathtt{and}\ (\varphi_\mathtt{d}\ \mathtt{is}\ \mathtt{A}_{1,\mathtt{d}})\ \ \mathtt{then}\ \ (\mathtt{y}\ \mathtt{is}\ \mathtt{B}_1) \\ \ldots\ldots\ldots\ldots \\ \mathtt{if}\ (\varphi_1\ \mathtt{is}\mathtt{A}_{\mathtt{p},1})\ \ldots\ \mathtt{and}\ (\varphi_\mathtt{d}\ \mathtt{is}\ \mathtt{A}_{\mathtt{p},\mathtt{d}})\ \ \mathtt{then}\ \ (\mathtt{y}\ \mathtt{is}\ \mathtt{B}_\mathtt{p}) \end{array} \tag{86}$$

where the fuzzy sets $\mathtt{A}_{\mathtt{j},\mathtt{i}}$ are doubly indexed: `i` is the index of the input coordinate, and `j` is the index of the rule. We denote the membership functions by $\mu_{\mathtt{A}_{\mathtt{j},\mathtt{i}}}(\varphi_i)$ and $\mu_{\mathtt{B}_\mathtt{j}}(y)$, respectively.

A Simple Example: A DC-motor Consider an electric motor with input voltage u and output angular velocity y. We would like to explain how the angular velocity at time t, i.e. $y(t)$, depends on the applied voltage and the velocity at the previous time sample. That is, we are using the regressors $\varphi(t) = [\varphi_1(t), \varphi_2(t)]^T$, where $\varphi_1(t) = u(t-1)$ and $\varphi_2(t) = y(t-1)$. Let us now device a rule base of the kind (86), where we choose $\mathtt{A}_{1,1}$ and $\mathtt{A}_{2,1}$ to be "low voltage", $\mathtt{A}_{3,1}$ and $\mathtt{A}_{4,1}$ to be "high voltage". We choose $\mathtt{A}_{1,2}$ and $\mathtt{A}_{3,2}$ to be "slow speed", while $\mathtt{A}_{2,2}$ and $\mathtt{A}_{4,2}$ are "fast speed". The membership function for "low voltage" is taken as $\mu_{\mathtt{A}_{1,1}}(\varphi_1) = \rho(\varphi_1, 3, 4)$, where

$$\rho(x,a,b) = \begin{cases} 1 & \text{for } x < a \\ 1 - (x-a)/(b-a) & \text{for } a \le x < b \\ 0 & \text{for } b \le x \end{cases} \tag{87}$$

The membership function for "high voltage" is taken as $\mu_{\mathtt{A}_{3,1}} = 1 - \mu_{\mathtt{A}_{1,1}}$. The membership functions for slow and fast speed are chosen analogously, with breaking points 8 and 15 rad/sec. The statements $\mathtt{B}_\mathtt{i}$ about the outputs are chosen to be triangles with vertices respectively located at 5, 10 and 20 rad/sec. We thus obtain a rule base:

```
If φ1(t) is low and φ2(t) is slow then y(t) is low,
If φ1(t) is low and φ2(t) is fast then y(t) is medium,
If φ1(t) is high and φ2(t) is slow then y(t) is medium,
If φ1(t) is high and φ2(t) is fast then y(t) is high.
```

Combining Rules

The rule base (86) is at first sight quite different from the models we have discussed in the other sections of this article. To see the connections we shall now give the mathematical translation of it.

Combining fuzzy rules within our fuzzy rule basis is interpreted as taking the " **and** " of their conclusions[9].

Then, the fuzzy rule basis (86) means

$$\texttt{y is B}'_1 \texttt{ and } \ldots \texttt{ and y is B}'_\texttt{p}$$

where the fuzzy sets $\texttt{B}'_\texttt{j}$ are defined according to (85).

Expressing the " **and** " combinator as the product of membership functions, we get

$$\begin{aligned} \mu_{\texttt{B}'}(y) &= \prod_{j=1}^{p} \mu_{\texttt{B}'_\texttt{j}}(y) \\ \text{(by (85))} &= \prod_{j=1}^{p} \left(1 - \prod_{i=1}^{d} \mu_{\texttt{A}_{\texttt{j,i}}}(\varphi_i) \left(1 - \mu_{\texttt{B}_\texttt{j}}(y)\right)\right) \\ &\approx 1 - \sum_{j=1}^{p} \left(1 - \mu_{\texttt{B}_\texttt{j}}(y)\right) \prod_{i=1}^{d} \mu_{\texttt{A}_{\texttt{j,i}}}(\varphi_i) \end{aligned}$$

where we have used the approximation $\prod_{j=1}^{p}(1-u_j) \approx 1 - \sum_{j=1}^{p} u_j$, which is valid for u_j small and p large.

Now, assume that the membership functions in the rule basis are subject to the following identity

$$\sum_{j=1}^{p} \prod_{i=1}^{d} \mu_{\texttt{A}_{\texttt{j,i}}}(\varphi_i) \equiv 1 \,. \tag{88}$$

This will be true if the membership functions defined in each input domain form a *strong fuzzy partition*, i.e., $\sum_j \mu_{\texttt{A}_{\texttt{j,i}}}(\varphi_i) = 1$ holds for all φ_i, and if the rule basis is "complete", i.e., it covers all the cases in terms of the fuzzy sets defined in the input domains (it is easy to verify that the DC-motor example above obeys this requirement). In this case we have

$$\mu_{\texttt{B}'}(y) = \sum_{j=1}^{p} \mu_{\texttt{B}_\texttt{j}}(y) \prod_{i=1}^{d} \mu_{\texttt{A}_{\texttt{j,i}}}(\varphi_i) \,. \tag{89}$$

Defuzzification

At this point, setting $\varphi = (\varphi_1, \ldots, \varphi_\texttt{d})$, formula (89) defines a function mapping points $\varphi \in \mathbf{R}^d$ into fuzzy sets. To get a function in the usual setting $\mathbf{R}^d \mapsto \mathbf{R}$, we perform the *defuzzification* of $\mu_{\texttt{B}'}(y)$, using the so-called "Height Method", see (Lee, 1990; Dubois and Prade, 1992). Using again property (88), we finally get the ordinary function

$$y = \sum_{j=1}^{p} y_j \left(\prod_{i=1}^{d} \mu_{\texttt{A}_{\texttt{j,i}}}(\varphi_i)\right) \triangleq \sum_{j=1}^{p} y_j \, w_j(\varphi) = g(\varphi) \tag{90}$$

[9]From our choice for implication, whereas if implication is encoded as a " **and** ", intermediate results $\mu_{B_j}(y)$ are aggregated by a " **or** ".

where $\varphi = (\varphi_1, \ldots, \varphi_d)$, y_j is the point at which $\mu_{\mathtt{B_j}}$ reaches its maximum value, and the definition of the weight functions $w_j(\varphi)$ is obvious. If property (88) does not hold, then the above defuzzification formula is modified accordingly (Wang, 1992) :

$$y = g(\varphi) = \frac{\sum_{j=1}^{p} y_j \; w_j(\varphi)}{\sum_{j=1}^{p} w_j(\varphi)} \; . \tag{91}$$

A rule basis may be directly built with *crisp* conclusions, i.e., $\mathtt{B_j}$ are ordinary values in (86). In this case no defuzzification is needed.

9.3 Back to the general black-box formulation

With (91) or (90) we are now back to the predictor model form we discussed in Section 2: A mapping from the regression vector φ to the (predicted) output.

Now, if some or all of the rules in the rule base need "tuning", we may introduce parameters to be tuned. These parameters could be all or some of the numbers y_i in (90) or in (91). For example, if y_i is unknown, it could be replaced by an adjustable parameter α_i.

Parameters can also be introduced in the membership functions. Usually, fuzzy set membership functions are parameterized functions of the form

$$\mu_{\mathtt{A}}(\varphi, \beta, \gamma) = \mu(\beta(\varphi - \gamma)) \tag{92}$$

where $\mu(\varphi)$ is a given function with values in $[0,1]$, β is a dilation factor, and γ is a translation factor, and the pair (β, γ) encodes the fuzzy set $\mathtt{A}$. Mostly used is the piecewise linear function μ such that $\mu(1) = 1$ and $\mu(\varphi) = 0$ for φ outside the interval $[0,2]$.

When parameters are introduced in this way, the model (90) takes the form

$$y = g(\varphi, \theta) = \sum_{j=1}^{p} \alpha_j g_j(\varphi, \beta, \gamma) \tag{93}$$

where the "basis functions" g_j are obtained from the parameterized membership functions as

$$g_j(\varphi, \beta, \gamma) = \prod_{i=1}^{d} \mu_{\mathtt{A_{j,i}}}(\beta_{ji}(\varphi_i - \gamma_{ji})) \; . \tag{94}$$

We are thus back to the basic situation of (19) and (20), the only difference being that the basis functions g_j are created by dilation and translation of a basic function $\mu(\varphi)$ in a more complex way than in (20). The estimation of the free parameters θ in (93) still follows the general theory.

If the fuzzy partition is fixed and not adjustable (i.e β and γ fixed), then we get a particular case of the Kernel estimate (29). Identified fuzzy models are often referred to as "neuro-fuzzy models" in the A.I. literature (Glorennec, 1993), since back-propagation procedure can be used for their training, as for neural networks. It is also proved that fuzzy models are universal approximators (Wang, 1992), which is not surprising.

To summarize, fuzzy models are described by fuzzy rule bases, plus some additional parameters which make vague statements such as "large", "small", etc., to be precise in terms of membership functions. The fuzzy rule basis exhibits the structure of the model, plus some coarse features related to the location of the elementary functions in the decomposition (90) or (91). Thus *fuzzy models are just particular instances of the general model structure (19), with the advantage of providing the fuzzy rules as a way to describe some possibly available*

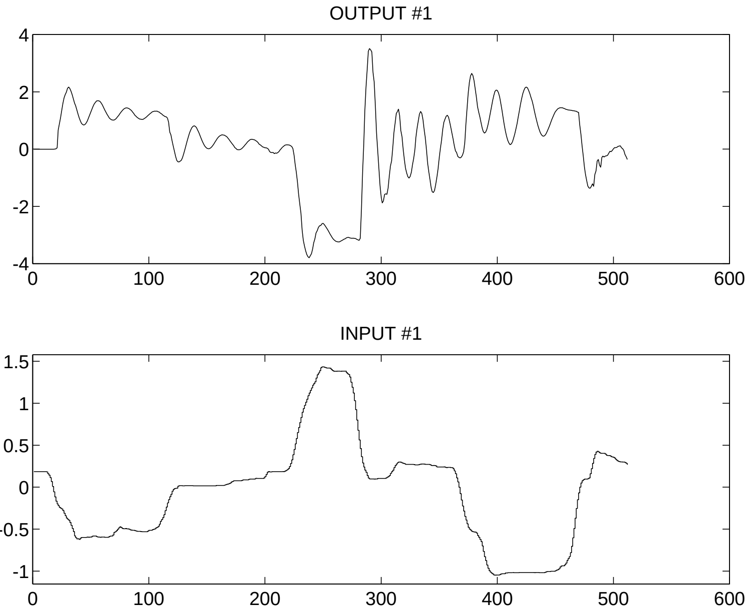

Figure 6: Measured values of oil pressure (top) and valve position (bottom).

prior knowledge. In the experiments reported in section 10, neuro-fuzzy modelling is used in the above sense. Also, in (Juditsky et al., 1994), an extension of the classical fuzzy modelling syntax is proposed to encompass *multiresolution* model structures, such as wavelet decompositions or networks.

10 Some Experiments

In this section we present some application examples of nonlinear black-box modeling, in order to give the reader some practical insights. They cover dynamic system modeling, static system modeling and fuzzy system modeling.

10.1 Modeling a Hydraulic Robot Actuator

In this section we shall study identification of a hydraulic actuator. We shall consider both linear models and nonlinear black box ones, based on neural networks and wavelet networks.

The Data. The position of a robot arm is controlled by a hydraulic actuator. The oil pressure in the actuator is controlled by the size of 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. In (Gunnarsson and Krus, 1990) a thorough description of this particular hydraulic system is given. Figure 6 shows measured values of the valve size u and the oil pressure y, which are input and output signals, respectively. As seen in the oil pressure, we have a very oscillative settling period after a step change of the valve size. These oscillations are caused by mechanical resonances in the robot arm.

A Linear Model. Following the principle "try simple things first" gives an ARX model which predicts the output by the three most recent past outputs and the two most recent past inputs,

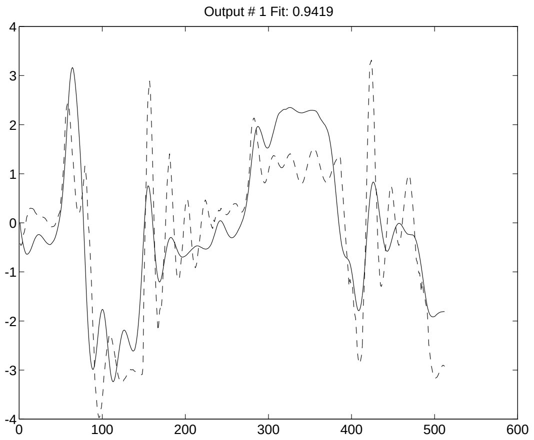

Figure 7: Simulation of the linear ARX model on validation data. Solid line: simulated signal. Dashed line: true oil pressure.

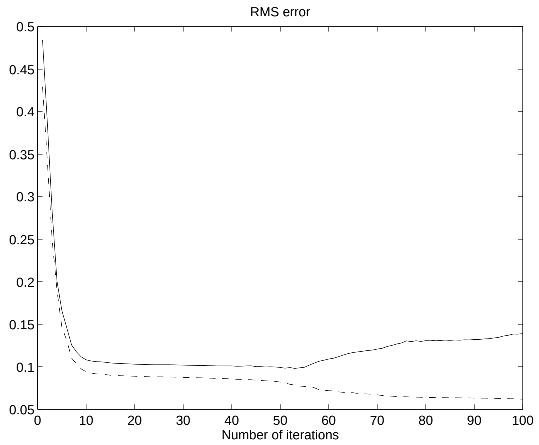

Figure 8: Sum of squared error during the training of the NARX model. Solid line: Validation data. Dashed line: estimation data.

i.e. the regression vector $\varphi = [y(t-1), y(t-2), y(t-3), u(t-1), u(t-2)]^T$ where y and u are the output and the input of the system, respectively. In Figure 7 the result of a simulation with the obtained linear model on validation data is shown. The result is not very impressive.

A Neural Network Model. Next, a NARX model based on an one-hidden-layer sigmoid neural network with ten hidden units is considered, as described in Section 4. The same regressor as for the linear model is used and this gives a model with 71 parameters. In Figure 8 it is shown how the quadratic criterion develops during the estimation for estimation and validation data, respectively. For the validation data the criterion first decreases and then it starts to increase again. This is the overtraining which was described in Section 7.4. The best model is obtained at the minimum and this means that not all parameters in the nonlinear model have converged and, hence, the "efficient number of parameters" is smaller than $\dim\theta = 71$.

The parameters which give the minimum are then used in the nonlinear model. When this model is simulated on the validation data it gives a root mean square error (RMS) of 0.467 which is considerably smaller than the 0.942 which is obtained with the linear model.

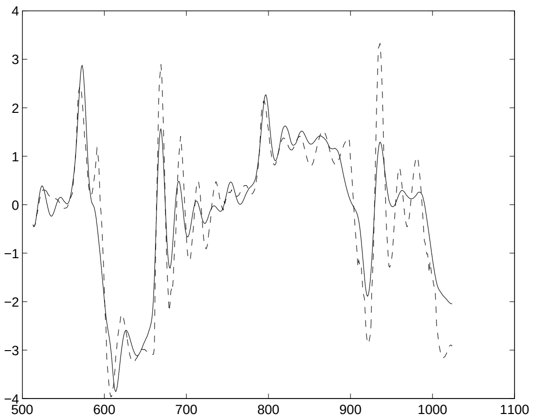

Figure 9: Simulation of the nonlinear wavelet network NARX model on validation data. Solid line: simulated signal. Dashed line: true oil pressure.

A Wavelet Network Model. Now, another NARX model based on a wavelet network is considered to model the hydraulic actuator in a similar way, with the same regressors. The used wavelet function is $\psi(\varphi) = (d - \varphi^T\varphi)e^{-\frac{1}{2}\varphi^T\varphi}$ with $d = \dim(\varphi)$. The dilation matrices β_k (in (26), were chosen as multiples of the identity matrix. First, we apply the SSO procedure (see Subsection 8.2) that iteratively selects wavelets into the model. By Akaike's final prediction error criterion the number of wavelets n_h is chosen to be 3, corresponding to 27 model parameters. Then, the NARX model issue of this iterative construction is used to simulate the output of the robot arm on the validation data. The corresponding RMS on the validation data is 0.647. Finally, the NARX model is refined by 10 iterations of the Levenberg-Marquardt procedure and is used for simulation in the same way as above. The result is depicted in Figure 9 and the RMS becomes 0.579. We can observe that the Levenberg-Marquardt procedure only slightly improved the result. This suggests that the iterative construction method found a model parameter close to a local minima searched by the Levenberg-Marquardt procedure.

Other Non-linear Structures. The two non-linear models considered so far have been obtained by just plugging in the regressor in the non-linear structure. We can also try some of the structures which were suggested in Section 3.4.

The structure (16) based on the assumption of additive noise gives us a NARX model which is linear in past $y(t)$. The parameter estimation becomes much easier with this model structure. Less parameters speeds up the numerical search and a model which is linear in some of the regressors also has less problem with local minima.

It turns out that with this structure it becomes advantageous to include two more regressors to those we had before and the predictor model becomes

$$\hat{y}(t) = g(u(t-1), \ldots, u(t-3)) + a_1 y(t-1) + \cdots + a_4 y(t-4) \tag{95}$$

where g is modeled by a neural net with 4 hidden units. This gives a model with 25 parameters which is about a third compared to the number of parameter of the first neural net model. This time there are no problems with overlearning during the estimation of the parameters. Simulating this model in the same manner as with the other models gave RMS 0.400.

If the linear part of the model (95) is replaced by a neural net then we obtain a NARX model consisting of two neural nets as in (17). This gives us more flexibilities than if the model is kept linear in past $y(t)$, but not quite so much flexibilities as in the first model where all regressors where fed into one large network. With 3 units in each neural net one obtain a model with 35 parameters. The RMS error for simulation on validation data became 0.328 and the simulation is depicted in Figure 10.

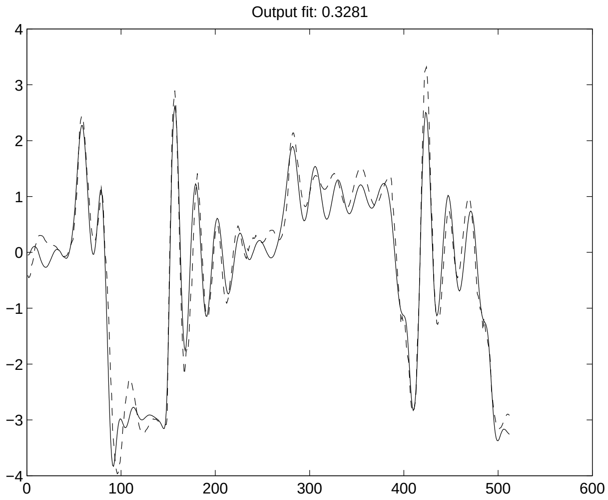

Figure 10: Simulation of the nonlinear neural network NARX model on validation data. Solid line: simulated signal. Dashed line: true oil pressure.

10.2 Modeling a Gas Turbine

Gas turbines are power motors, typically used in electrical power generators and aircrafts. Usually a gas turbine system is mainly composed of a compressor, one or several combustion chambers and an expansion turbine, as illustrated in Figure 11.

One of the purposes of our joint study with European Gas Turbine SA, Belfort, and Alcatel-Alsthom-Recherche, Marcoussis, was to develop a monitoring and diagnostics system for the joint system {combustion chambers, expansion turbine}. For this purpose a semi-physical model has been developed that predicts the temperature profile of the exhaust gas. Due to the phase shift of the gas in the turbine, this semi-physical model is strongly nonlinear (Zhang, 1991; Zhang et al., 1994).

18 thermocouples $t_1, \ldots, t_{18}$ are installed at the exhaust of the turbine to measure the output temperature profile. The compression rate ξ of the compressor and the rotation velocity ω of the turbine are also measured. As suggested by the semi-physical model, we have chosen the average of the measurements of the 18 thermocouples T_s, ξ and ω as regressors, i.e. $\varphi = (T_s, \xi, \omega)^T$; and the deviations from the average T_s of the thermocouples $y_i = t_i - T_s$, $i = 1, \ldots, 18$ as outputs of the black-box model.

We have experimented this approach on the data taken from a gas turbine of European Gas Turbine SA. The training data were collected during about 48 hours. We have re-sampled the data and kept only 1000 measurement points for model estimation. For the sake of brevity, we

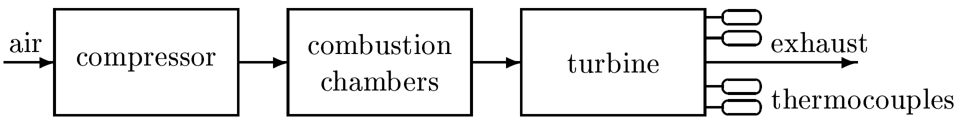

Figure 11: A gas turbine system

models	RBS-net	SSO-net	BE-net	semi-physical	linear
Init. RMS	0.0743	0.0708	0.0719		
Opt. RMS	0.0729	0.0743	0.0698	0.1136	0.0922
Init. flops	2.0718×10^7	4.3714×10^8	7.5143×10^7		
Opt. flops	1.5365×10^9	1.5365×10^9	1.5365×10^9	9.8041×10^8	9.6272×10^4
Init. time (sec.)	41.6	251.2	87.2		
Opt. time (sec.)	2461.8	2383.8	2456.5	2265.0	0.1921

Table 1: Performance evaluation of the turbine models. RBS-net, SSO-net and BE-net mean the wavelet network models initialized by RBS, SSO and BE procedures, respectively. RMS means square Root of Mean Square error. The RMSs are evaluated on the validation data Z_v. For the network models, Init. RMS corresponds to the initialized model, and Opt. RMS corresponds to the model optimized by 10 iterations of the Levenberg-Marquardt procedure. Flops is a Matlab measure of computational burden. The computation time is based on programs in Matlab 4.2 language executed on a Sun Sparc-2 workstation.

shall show only the results concerning the first thermocouple. The obtained models are tested on another set of measured data, which we refer to as the *validation data* Z_v.

We have tested the semi-physical model, a linear regression model and wavelet network models. For the wavelet network model, we have chosen the radial wavelet function $\psi(\varphi) = (d - \varphi^T\varphi)e^{-\frac{1}{2}\varphi^T\varphi}$ with $d = \dim(\varphi)$. The number of wavelets used in the networks is 40, corresponding to 204 model parameters. We initialize the wavelet networks with each of constructive procedures (RBS, SSO and BE, as described in Subsection 8.2) and optimize them with the Levenberg-Marquardt procedure.

The results are summarized in Table 1. The outputs corresponding to the validation data Z_v predicted by the linear regression model and by a wavelet network model are plotted in Figure 12. Though by the values of square root of mean square errors (RMS) in Table 1, the linear regression model is not too bad compared to other models; the plots in Figure 12 show that the wavelet model does significantly improve the prediction accuracy.

Sigmoid neural networks have also been tested on this example, the results are similar to those obtained with wavelet networks.

The nonlinear black-box models perform better than the semi-physical model in terms of output prediction. It is at the price of much higher computational complexity. On the other hand, the semi-physical model, though less accurate, allows to perform physical diagnosis of the system faults (Mathis, 1994); in contrast, the black-box models give the possibility to implement a finer global alarm of the monitoring system (see also (Mathis, 1994)), but the model parameters do not provide any physical information for fault diagnosis.

10.3 Modeling Glycæmic Variations

This is a medical example illustrating the use of fuzzy models.

Describing the problem. Glycæmic variations depend on several factors which are not easily quantifiable and, moreover, may vary with time. Food diet, physical activity, stress and emotions, proximity of meal, have effects that the doctors know how to *qualitatively* assess. For a healthy person, glycæmic regulation is ensured via the secretion of insulin by the pancreas. In case of organic deficiency, for diabetic persons, insulin must be artificially injected. Deciding the amount for injection is very difficult, because morphology, future physical activity, time of

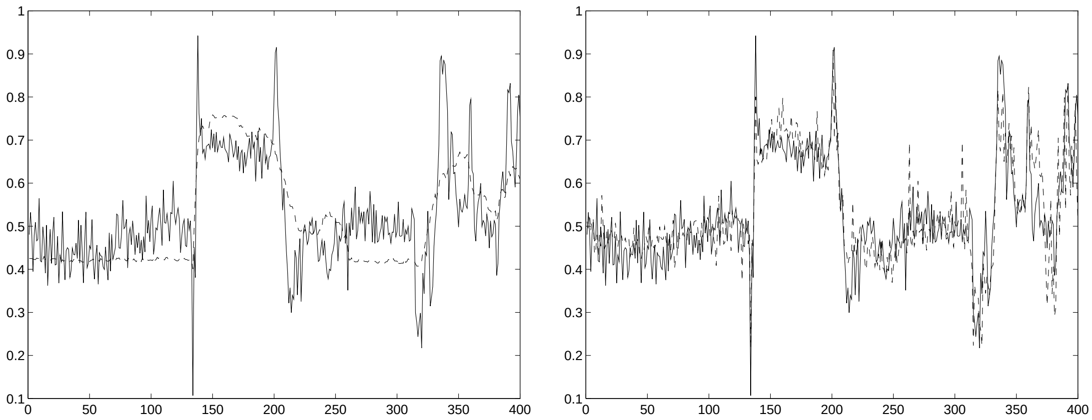

Figure 12: Comparison on the validation data Z_v of the predictions by the linear regression model (left) and by the wavelet network initialized by BE procedure (right). The solid lines represent the true measurements and the dashed lines represent the outputs of the models.

meal, glucide richness of meal, present glucose concentration, and results of the previous day, have to be taken into account. Moreover, injected insulin acts with delay, and its efficiency reduces as glucose concentration gets higher. Lastly, hypoglycæmia is almost always followed by hyperglycæmia. For an optimum glycæmic control, it would be better to anticipate before the glucose level rises, as it occurs for endogenic insulin secretion in healthy persons. To summarize, we have to deal with a nonlinear, unstable system, with time delay.

Doctors have devised empirical rules allowing the diabetic persons to approximatively compute themselves the insulin level for injection. For diabetic persons using a pump, insulin injection rate has two parts: the basic flow rate, denoted $B_a(t)$, and providing about 50% of daily insulin needs, and a variable part, the bolus, denoted $B_o(t)$, which is a flash injection to assimilate a recent meal.

Nevertheless, despite doctor's experience, it is very difficult to manually obtain a more or less constant glycæmic level, in part because a good control should take into account up to six input variables, which is far beyond human control capability. This motivated us to propose a predictive glycæmic model, as a basis for automatic injection control. This model uses as a basis the empirical rules of doctors, and takes into account the qualitative nature of available data. For this proposal, we have several "self-supervision note-books", i.e., daily support to control the context and the treatment of insulin-dependent diabetic patients under pump operation. Thus, each day the diabetic writes on his note-book 1/ time and actual glycemia, 2/ time, importance and quality of his meal, 3/ activity, 4/ insulin injection. Experimental results on this case study are reported now.

The variables of interest and their qualitative labels. Diabetologists' knowledge is expressed under the form of "rule of thumb" advice. We have used this knowledge to build a two hour ahead predictive model of glycæmic variations. This predictive model will be subsequently used in a control system. We have restricted our model to six inputs (current instant t is omitted for simplicity) as described in table 2. The output is the predicted variation of glycemia at time $t + 2$ hours, `DG(t+2)` $\in$ {`PVB, PB, PM, PS, Z, NS, NM, NB, NVB`}, where `P` means "Positive", `N` "Negative", `S` "Small", `B` "Big", etc. Figure 13 shows membership functions of glycemia, where the $(g_i)_{i=0}^{4}$ parameters must be determined by learning since their optimal value depends on the patient. Membership functions have been represented by simple first order

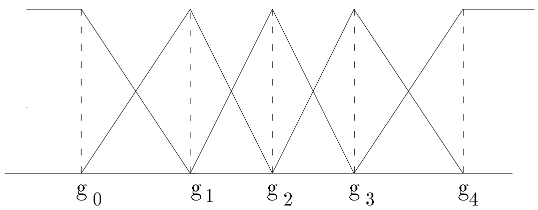

Figure 13: Fuzzy partition for glycemia

item	symbol	fuzzy values
glycæmia	`Gl`	`Very Low (VL)` `Low (L)` `Normal (N)` `High (H)` `Very High (VH)`
basis insulin injection rate	`Ba`	`Low`, `Normal`, `High`
flash insulin injection rate	`Bo`	`Low`, `Normal`, `High`
elapsed time since previous meal	`Dr`	`Far Before`, `Near`, `Just After`, `Far`
diet	`Nr`	`Fiber` , `Normal`, `Glucidic`
expected future activity	`Ac`	`Low`, `Normal`, `High`

Table 2: Fuzzy input variables

splines with free knots. Our method follows the following two steps:

1. start with an initial guess of the model, based on available (qualitative) prior knowledge;
2. tune this model to the particular patient in consideration, by performing learning or optimization from available data.

Expressing prior knowledge. Combining all possible qualitative values for the different inputs yields 1620 different cases, corresponding to the same amount of candidate fuzzy rules. In fact, only 64 rules were considered for our prior model, thus reflecting the actual domain for the input variables where meaningful knowledge exists. Example of such rules are

```
if  (GL(t) is VL)  and  (Nr(t) is N)  then  DG(t+2) is PB
if  (GL(t) is  L)  and  (Ba(t) is L)  then  DG(t+2) is NS
```

Figure 14 shows predicted glycemia at $t + \delta$ from glycemia at time t, with $\delta = 2$ hours, *before learning,* i.e., with only use of the prior model. The solid line shows the actual glycæmia and the dashed line the predicted one. The doctor's rules are quite efficient in predicting the effect of insulin injections. Still some spikes occur in the prediction error. Prediction error has mean $\mu = -0.20$ and standard deviation $\sigma = 0.38$.

Tuning the model for each patient. Using data from patient's note-book, we divided data file into two parts, one for learning and the other for validation (i.e., testing). Figure 15 shows

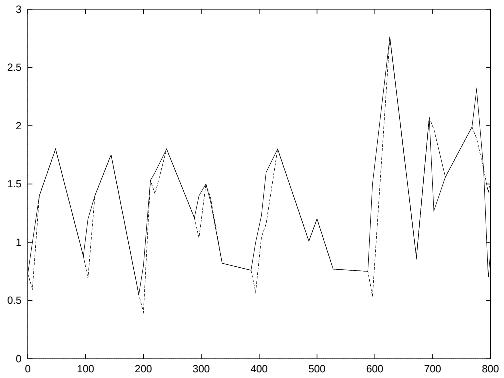

Figure 14: *Prior model:* two hour ahead prediction (dashed line) vs. actual (solid line) glycæmia

predicted glycemia at $t + 2$ from glycemia at time t, *after learning,* i.e., subsequent learning of the g_i parameters on data. The prediction error has mean $\mu = -0.0003$ and the standard deviation $\sigma = 0.29$. Some improvement is seen, note that such an improvement is likely to be patient dependent. The errors around time steps 700 and 800 are due to catheter changes (as marked in the note-book) which usually lead to inject more insulin than expected.

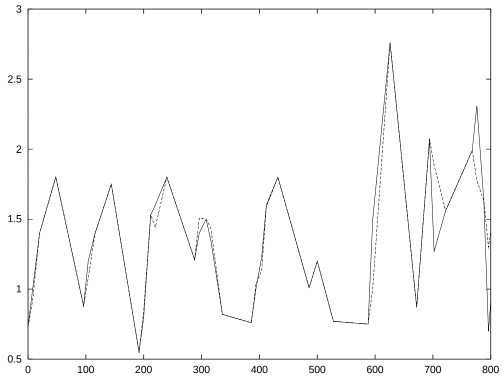

Figure 15: *Model after learning:* two hour ahead prediction (dashed line) vs. actual (solid line) glycæmia

Comments and conclusions about this example. The following conclusions can be drawn from this example:

- Fuzzy rules turned out to be a convenient way to express prior knowledge from doctors, in part because this prior knowledge is mainly qualitative. It is important to notice that this fuzzy rule basis was far from being equivalent to an exhaustive table describing the input-output map, since only a few percent (64/1620) of this table was described by the rules. This restriction is by itself a useful prior information about the range of validity of the modelling.

- Subsequent tuning of the prior model was performed while preserving the structure of the model, i.e., the fuzzy rules were not modified, only the g_i parameters hidden in the splines were adjusted. It would also be possible to use our prior model as initial guess but allow other "rules" to be introduced via learning; corresponding experiments are under progress.

- Another advantage of describing the model via fuzzy rules is the possibility to "decompile" the model after learning, again in the form of fuzzy rules, for return to the user (doctor or patient). Returning a mathematical model would be of little use for the average user, having no training in mathematics.

11 Summary and recommendations

System identification cannot be fully formalized and automated. A user must always blend his or her experience and common sense with established theory and methodology. In this section we take the position of a user with relevant software support available. Assume that we have collected input-output data from a system and shall estimate a model based on them. What are the things to consider for a successful result?

11.1 Some General Concerns

Look At the Data. This is the first and obvious step. It is often very revealing. Nonlinear effects can often be detected by ocular inspection: Are responses similar at different levels and in different directions? What time constants can be seen etc.

Try Simple Things First A good engineering principle is to *try simple things first.* In the identification context, "simple" may mean both the size and the computational complexity of models. In practice it certainly means that one should try linear models first, to see if they can solve the problem, and if not, get some insight into their shortcomings. From a theoretical estimation point of view, simplicity primarily refers to the number of estimated parameters. By searching from simpler to more complex models until a valid one is found, typically a good trade-off between bias and variance can be achieved.

Look Into the Physics Physical insights may suggest to (linearly or nonlinearly) transform raw measurements into new regressors. Try to use such semi-physical regressors (cf Section 3.3) first in linear black box structures. Only if this gives unsatisfactory results, or if physical insight is completely lacking it is time to move to the nonlinear black box structures described in this paper. Even for these models it makes sense, though, to use semi-physical regressors.

The Bias-Variance Trade-off The *bias-variance trade-off* (43) tells us that one should not excessively increase the number of estimated parameters (i.e., the number of basis functions in the model). The ultimate improvement of the model quality could be obtained by suitably choosing basis functions that would require physical knowledge and lead to physical models.

Validation and Estimation data The best way for evaluating a identified model is to test it on *fresh data* (which were not used for model estimation). We have pointed to this use of validation (or "generalization") data for determining the model complexity – the Bias-Variance – trade-off both in terms of model structure complexity, size of regularization parameter and when to stop the iterations. Checking out a potential model on validation data has a clear pragmatic appeal: Can it reproduce previously unseen data in a satisfactory manner, then it must be of some use.

The Notion of Efficient Number of Parameters The variance contribution to the model output error in (40) is, principally, proportional to the number of parameters used in the model structure, if they have been estimated by minimization of (34). This means that a parameter that is not so important for the model fit, will still contribute as much as an important parameter to the variance error. The intuitive explanation is that the un-important parameter will be estimated very inaccurately, so even though its influence is small on the fit, its large errors will still be damaging.

It is thus tempting to have estimation schemes that reward "the important parameters." There are a number of possibilities. *Regularization*, which was reviewed in subsection 6.4, is the classical method. A variant of regularization is *to stop the iterations* in the minimization of (34) before the true minimum has been found, as described in Section 7.4. A third way of focusing on important parameters is to first estimate "many" and then discard those that are "small", and then possibly reestimate the values of the remaining ones. This is what we called *shrinking* and basis function selection in Section 8. The remaining number of parameters will essentially determine the model variance error.

11.2 Structural Issues to Consider

Regressor selection

A rational question to ask would be: Given that I am prepared to use d regressors, how should I distribute these over the five possible regressor choices listed in Section 3? There is no easy and quantitative answer to this question, but we may point to the following general aspects:

- A first choice to consider consists in trying static models, i.e., taking only $u(t)$ as the regressor.
- Including $u(t-k)$ only, $k = 1, 2, \ldots$, requires that the whole dynamic response time is covered by past inputs. That is, if the maximum response time to any change in the input is Υ, and the sampling time is T, then the number of regressors should be Υ/T. This could be a large number. On the other hand, models based on a finite number of past inputs cannot be unstable in simulation, which often is an advantage.

  A variant of this approach is to form other regressors from u^t, *e.g.* by Laguerre filtering, (e.g (Wahlberg, 1991)). This retains the advantages of the FIR-approach, at the same time as making it possible to use fewer regressors. It does not seem to have been discussed in the context of nonlinear black boxes yet.
- Adding $y(t-k)$ to the list of regressors makes it possible to cover slow responses with fewer regressors. This is quite important for nonlinear models since trying to achieve the same objective with more delayed inputs is much more prohibitive than for linear models. A disadvantage is that past outputs bring in past disturbances into the model. The model is thus given an additional task to also sort out noise properties. A model based on past outputs may also be unstable in simulation from input only. This is caused by the fact that the past measured outputs are then replaced by past model outputs.
- Bringing in past predicted or simulated outputs $\hat{y}(t-k|\theta)$ or past values from other nodes in the network, that may be interpreted as state variables may be quite useful. It typically increases the model flexibility, but also leads to non-trivial difficulties, related to the *recurrent* nature of the resulting network. See subsection 4.3. Two problems must be handled:

– It may lead to instability of the network, and since it is a nonlinear model, this problem is not easy to monitor.
– The regressors that are fed back depend on θ. In order to do the minimization iterations in the true gradient direction, this dependence must be taken into account, which is not straightforward. If the dependence is neglected, convergence to local minima of the criterion function cannot be guaranteed.

The balance of this discussion is probably that the NARX-regressors $(y(t-k), u(t-k))$ should be the first ones to test.

Choice of Basis Functions

Now that the regression vector φ has been decided upon, the question is which function expansion (19) to use. We thus return to the choices listed in Section 4.2. This is a more difficult decision, and the collected experience on this is not yet substantial. All of the described model structures are capable of approximating any reasonable function. The question is to pick one that "suits the application," in the sense that only few terms will be needed.

Curse of Dimensionality. The dimension of the regression vector is denoted by d, so the function to be approximated by (19) has $\mathbf{R}^d$ as its domain. Even for moderate size of d the observations φ are by necessity very sparse in any bounded region of $\mathbf{R}^d$ of practical interest. For example, it takes N =ten billion observations to fill up the unit cube in $\mathbf{R}^{10}$ even with a coarse component-wise grid of granularity 0.1. This consideration is important for the choice between basis functions obtained by radial constructions and ridge constructions. See below.

Radial constructions. In view of the curse of dimensionality, local basis functions are a prime choice when the dimension of the regression vector is rather small. For $d \leq 3$, the wavelet basis function expansion would be an excellent choice, since the wavelet coefficients can be estimated very efficiently. For somewhat larger values of d it is natural to try out wavelet networks and radial basis networks. For large values of d, model structures based on local basis functions will simply not support any model statements outside the areas where observations have been made (which is not unreasonable).

Multi-resolution aspects A very useful feature of the wavelet models is that the scale parameters can be chosen very differently. Certain areas in the data space can be covered by basis functions with large support, while others can be covered with much finer granuality. Also one and the same region may be covered by both types, to pick up both fine details and more course trends. This could be a useful way to deal with the lack of data in certain regions. One may note, though, that the curse of dimensionality does not only relate to the possible lack of supporting data. Any prior seed of basis functions to be screened with the help of data will also be huge in high dimensions, and that may be a major obstacle. A solution has been proposed in Section 8.2: scan the available data and pick only those basis functions that contain enough data points in their supports.

Ridge constructions. Ridge constructions, like the ones used in sigmoidal neural networks and the hinging hyperplanes networks, deal with the curse of dimensionality by extrapolation. This means that the functions identify certain directions in the $[y, \varphi]$-space where "not much happens." In other words, these are projection directions which would show clear data patters in the projected picture. These directions are chosen as the global ones. The approach thus has clear connections with *projection pursuit*, (Friedman and Stuetzel, 1981). The advantage is that higher regression-vector dimensions can be handled, by extrapolation into unsupported data regions. Whether this is reasonable or not, depends of course on the application. Experience indicates that the approach is often successful.

Basis functions by prior verbal information. Building up the basis functions from fuzzy logic and fuzzy rules is another way of dealing with the curse of dimensionality. The extrapolation into unsupported data regions is then done based on the prior knowledge (right or wrong) about the system's behavior. In the regions where the model is supported by data, it is modified according to the information in the observations. A perhaps even more important aspect of the choice of basis functions via fuzzy sets, is to specify the domain of interest, i.e., the areas where input data are expected. This seems to be a quite appealing way to deal with partial data information.

12 Conclusions

In the toolbox for system identification techniques one should have black-box models for non-linear dynamical systems available. It is true that it is to be preferred to use physical insight to build up the nonlinear effects in a model, since this typically can be done using fewer parameters. However, such insight is not always available, and if linear approximative models are not good enough, there is no other choice than to turn to black box structures.

This topic is not at all new. The "classical" literature on the subject seems to have concentrated on global basis function expansions, such as Volterra expansions. These have apparently had limited success. The topic was really revived by the onslaught of neural network applications.

In this paper we have treated most of the possibilities for black-box nonlinear dynamical models in a common framework. We have pointed to the similarities in the different approaches and we have tried to pinpoint what the real choices are. The bottom line is that there is a choice of basis functions. Each of the basis functions also carry some parameters to let them adjust to the observed data. These parameters typically correspond to scale and location of the function support. Scale, location as well as function coordinates can either be estimated by one joint minimization process, or by a first, separate step to fix location and scale.

The perspective of this paper has been the user's. We have not given details about approximation theory or properties of the function expansions. We have focused on the choices that the user has to make for a successful application. More mathematical investigations can be found in the companion paper (Juditsky et al., 1995).

References

Barron, A. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. on Information Theory*, 39(3).

Baum, E. and Wilczek, F. (1988). Supervised learning of probability distributions by neural networks. In Andersson, D., editor, *Neural Information Processing Systems*, pages 52–61. Amer. Inst. of Physics, New York.

Breiman, L. (1993). Hinging hyperplanes for regression, classification and function approximation. *IEEE Trans. Information Theory*, 39(3):999–1013.

Brown, M. and Harris, C. (1994). *Neurofuzzy Adaptive Modelling and Control.* Prentice-Hall, New York.

Chen, S. and Billings, S. (1992). Neural networks for nonlinear dynamic system modelling and identification. *Int. J. Control*, 56(2):319–346.

Chen, S., Billings, S., and Grant, P. (1990). Non-linear system identification using neural networks. *Int. J. Control*, 51(6):1191–1214.

Chen, S., Billings, S., and Luo, W. (1989). Orthogonal least squares methods and their application to non-linear system identification. *Int. J. Control*, 50(5):1873–1896.

Chen, S., Cowan, C., and Grant, P. (1991). Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Trans. on Neural Networks*, 2(2):302–309.

Chui, C. (1992). *Wavelets : a tutorial in theory and applications.* Academic Press, Inc. Boston, San Diego.

Cover, T. and Thomas, J. (1991). *Information Theory.* Wiley, New York.

Cybenko, G. (1989). Approximation by superposition of a sigmoidal function. *Mathematics of control, signals and systems*, 2:303–314.

Daubechies, I. (1990). The wavelet transform, time-frequency. *IEEE Trans. on Information Theory*, 36(5):961–1005.

Daubechies, I. (1992). *Ten lectures on wavelets.* CBMS-NSF regional series in applied mathematics.

De Boor, C. (1978). *Practical guide to splines.* Applied mathematical sciences. Heidelberg, Springer, New York, Berlin.

Delyon, B., Juditsky, A., and Benveniste, A. (1995). Accuracy analysis for wavelet approximations. *IEEE Transactions on neural networks*, 6(1).

Dennis, J. and Schnabel, R. (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall, Englewood Cliffs, New Jersey.

Devroye, L. and Gyorfi, L. (1985). *Nonparametric Density Estimation.* John Wiley & Sons, New York.

Draper, N. and Smith, H. (1981). *Applied regression analysis.* Series in Probability and Mathematical Statistics. Wiley. Second edition.

Dubois, D. and Prade, H. (1992). Fuzzy sets in approximate reasoning, part 1. *Fuzzy Sets and Systems*, 40 n^{o}1.

Friedman, J. and Stuetzel, W. (1981). Projection pursuit regression. *J. Amer. Statist. ASsoc.*, 76:817–823.

Friedman, J. and Stuetzle, W. (1981). Projection pursuit regression. *J. Amer. Stat. Assoc.*, 76:817–823.

Glorennec, P. (1993). A general class of fuzzy inference systems. In *Proc. of CES2 Conf.*, Prague.

Gunnarsson, S. and Krus, P. (1990). Modelling of a flexible mechanical system containing hydralic actuators. Technical report, Dep. of Electrical Engineering, Linköping University, S-581 83 Linköping, Sweden.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation.* Macmillan College Publishing Company, 866 Third Avenue, NY.

Helland, I. (1990). Partial least squares regression and statistical models. *Scand. J. Statist.*, 17:97–114.

Huber, P. (1985). Projection pursuit (with discussion). *Ann. Statist.*, 13:435–475.

Juditsky, A., Hjalmarsson, H., Benveniste, A., Deylon, B., Ljung, L., Sjöberg, J., and Zhang, Q. (1995). Nonlinear black-box models in system identification: Mathematical foundations. *Automatica.* Available by anonymous ftp `130.236.24.1`.

Juditsky, A., Zhang, Q., Delyon, B., Glorennec, P.-Y., and Benveniste, A. (May 1994). Wavelets in identification. Technical report, IRISA.

Kugarajah, T. and Zhang, Q. (1995). Multi-dimensional wavelet frames. *IEEE Trans. on Neural Networks.* Accepted for publication in January 1995.

Kung, S. (1993). *Digital Neural Networks.* Prentice-Hall, Englewood Cliffs, New Jersey.

Lee, C. (1990). Fuzzy logic in control systems, parts i and ii. *IEEE Trans. on Systems, Man, and Cybernetics*, 20 n^{o}2.

Ljung, L. (1987). *System Identification: Theory for the User.* Prentice-Hall, Englewood Cliffs, NJ.

Ljung, L. and Glad, T. (1994). *Modeling of Dynamic Systems.* Prentice Hall, Englewood Cliffs, N.J.

Ljung, L. and Söderström, T. (1983). *Theory and Practice of Recursive Identification.* MIT Press, Cambridge, Massachusetts.

MacKay, D. (1991). *Bayesian Methods for Adaptive Models.* PhD thesis, Caltech, Pasadena, CA 91125.

MacKay, D. (1992). Baysian interpolation. *Neural Computation*, 4(3):415–447.

Mallat, S. (1989). Multiresolution approximation and wavelets orthonormal bases of $l^2(r)$. *Trans. Amer. Math. Soc.*, 315(1):69–8.

Mallat, S. and Zhang, Z. (1993). Matching pursuit with time-frequency dictionaries. Technical Report 619, New-York University, Computer Science Department.

Mathis, G. (1994). *Surveillance de turbine à gaz.* PhD thesis, Université de Rennes 1. In French.

Matthews, M. (1992). *On the Uniform Approximation of Nonlinear Discrete-Time Fading-Memory Systems using Neural Network Models.* PhD thesis, ETH, Zürich, Switzerland.

McAvoy, T. (1992). Personal communication.

Meyer, Y. (1990). *Ondelettes et Opérateurs*. Hermann.

Moody, J. (1992). The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. In Moody, J., Hanson, S., and Lippmann, R., editors, *Advances in Neural Information Processing Systems 4*. Morgan Kaufmann Publishers, San Mateo, CA.

Nadaraya, E. (1964). On estimating regression. *Theory of Prob. and Appl.*, 9:141–142.

Narendra, K. and Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Networks*, 1:4–27.

Nerrand, O., Roussel-Ragot, P., Personnaz, L., and Drefys, G. (1993). Neural networks and nonlinear adaptive filtering: Unifying concepts and new algorithms. *Neural Computation*, 5:165–199.

Nerrand, O., Roussel-Ragot, P., Urbani, D., Personnaz, L., and Drefys, G. (1994). Training recurrent neural networks: Why and how? an illustration in dynamical process modeling. *IEEE Trans. Neural Networks*, 5(2):178–184.

Poggio, T. and Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497.

Poggio, T. and Girosi, F. (1990). Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247:978–982.

Pucar, P. and Sjöberg, J. (1995a). On the hinge finding algorithm for hinging hyperplanes. Technical report, Report LiTH-ISY-R-1720, Dep. of Electrical Engineering, Linköping University, S-581 83 Linköping, Sweden. Available by anonymous ftp `130.236.24.1`.

Pucar, P. and Sjöberg, J. (1995b). On the parameterization of hinging hyperplane models. Technical report, Report LiTH-ISY-R-1717, Dep. of Electrical Engineering, Linköping University, S-581 83 Linköping, Sweden. Available by anonymous ftp `130.236.24.1`.

Qian, S. and Chen, D. (1994). Signal representation using adaptive normalized gaussian functions. *Signal Processing*, 36(1).

Reed, R. (1993). Pruning algorithms–a survey. *IEEE Trans. on Neural Networks*, 4(5):740–747.

Rivals, I. (1995). *Modélisation et Commande de Processus par Réseaux de Neurones; Application au Pilotage d'un Véhicule Autonome*. PhD thesis, Ecole Supérieure de Physique et Chimie Industrielles de la Ville de Paris, 10, rue Vauquelin, 75005 Paris, France.

Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, 323(9):533–536.

Ruskai, M., Beylkin, G., Coifman, R., Daubechies, I., Mallat, S., Meyer, Y., and Raphael, L., editors (1992). *Wavelets and their applications*. Jones and Bartlett books in mathematics, Boston.

Saarinen, S., Bramley, R., and Cybenko, G. (1993). Ill-conditioning in neural network training problems. *SIAM Journal on Scientific Computing*, 14(3):693–714.

Schumaker, L. L. (1981). *Spline functions : basic theory.* Pure and applied mathematics. Chichester, J. Wiley and sons, New York, Brisbane.

Silverman, B. (1986). *Density estimation for statistics and data analysis.* Chapman and Hall, London.

Sjöberg, J., Hjalmarsson, H., and Ljung, L. (1994). Neural networks in system identification. In *Preprint, 10th IFAC Symposium on System Identification, Copenhagen*, volume 2, pages 49–72. Available by anonymous ftp `130.236.24.1`.

Sjöberg, J. and Ljung, L. (1992). Overtraining, regularization, and searching for minimum in neural networks. In *Preprint 4th IFAC Symposium on Adaptive Systems in Control and Signal Processing*, pages 669–674, Grenoble, France.

Sontag, E. (1981). Nonlinear regulation: the piecewise linear approach. *IEEE Trans. on Automatic Control*, 26:346–358.

Sontag, E. (1993). Neural networks for control. In Trentelman, H. and Willems, J., editors, *Essays on Control: Perspectives in the Theory and its Applications*, volume 14 of *Progress in Systems and Control Theory*, pages 339–380. Birkhäuser.

Stone, C. (1982). Optimal global rates of convergence for nonparametric regression. *The Annals of Statistics*, 10:1040–1053.

Sugeno, M. and Yasukawa, T. (1993). A fuzzy logic based approach to qualitative modelling. *IEEE Trans. on Fuzzy Systems*, 1:7–31.

Suykens, J., Moor, B. D., and Vandewalle, J. (1994). Static and dynamic stabilizing neural controllers, applicable to transition between equilibrium points. *Neural Networks*, 7(5):819–831.

Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its application to modelling and control. *IEEE Trans. on Syst. Man and Cybernetics*, 15:116–132.

Taswell, C. (1993). Wavbox. Public domain MATLAB toolbox Anonymous FTP: simplicity.stanford.edu : /pub/taswell.

van den Hof, P., Heuberger, P., and Bokor, J. (1994). Identification with generelized orthonomal basis functions – statistical analysis and error bounds. In Blanke, M. and Söderström, T., editors, *Preprint 10th IFAC Symposium on System Identification*, volume 3, pages 3.207–3.212.

van der Smagt, P. (1994). Minimisation methods for training feedforward neural networks. *Neural Networks*, 7(1):1–11.

Wahba, G. (1987). Three topics in ill-posed problems. In Engl, H. and Groetsch, C., editors, *Inverse and Ill-posed Problems.* Academic Press.

Wahlberg, B. (1991). System identification using laguerre models. *IEEE Trans. on Automatic Control*, 36(5):551–562.

Wahlberg, B. (1994). System identification using kautz models. *IEEE Trans. on Automatic Control*, 39(6):1276–1282.

Wang, L. (1992). Fuzzy systems are universal approximators. In *Proc. First IEEE Conf. on Fuzzy Systems, 1163-1169*, San Diego.

Wang, L. (1994). *Adaptive Fuzzy Systems and Control: Design and Stability Analysis.* Prentice-Hall, Englewood Cliffs, NJ.

Watson, G. (1969). Smooth regression analysis. *Sankhya, Series*, A(26):359–372.

Werbos, P. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Science.* PhD thesis, Harvard University.

Wold, S., Ruhe, A., Wold, H., and Dunn, W. (1984). The collinearity problem in linear regression: The partial least squares approach to generalized inverses. *SIAM J. Sci. Stat. Comput.*, 5(3):743–753.

Zadeh, L. (March 1994). Fuzzy logic, neural networks, and soft computing. *Communications of the ACM*, 37 n^{o}3:77–86.

Zhang, Q. (1991). *Contribution à la surveillance de procédés industriels.* Thesis, Université de Rennes I.

Zhang, Q. (1993). Wavenet. Public domain MATLAB toolbox, Anonymous FTP: ftp.irisa.fr : /local/wavenet.

Zhang, Q. (1994). Using wavelet network in nonparametric estimation. Technical Report 833, IRISA.

Zhang, Q., Basseville, M., and Benveniste, A. (1994). Early warning of slight changes in systems and plants with application to condition based maintenance. *Automatica*, 30(1):95–113. Special Issue on Statistical Methods in Signal Processing and Control.

Zhang, Q. and Benveniste, A. (1992). Wavelet networks. *IEEE Trans. on Neural Networks*, 3(6):889–898.