LAZY LEARNING: A LOCAL METHOD FOR SUPERVISED LEARNING

Gianluca Bontempi, Mauro Birattari, Hugues Bersini
Iridia
Université Libre de Bruxelles
Brussels, Belgium
{gbonte,mbiro,bersini}@ulb.ac.be
http://iridia.ulb.ac.be/∼lazy

Abstract

The traditional approach to supervised learning is *global* modeling which describes the relationship between the input and the output with an analytical function over the whole input domain. What makes global modeling appealing is the nice property that even for huge datasets, a parametric model can be stored in a small memory. Also, the evaluation of the parametric model requires a short program that can be executed in a reduced amount of time. Nevertheless, modeling complex input/output relations often requires the adoption of global nonlinear models, whose learning procedures are typically slow and analytically intractable. In particular, *validation* methods, which address the problem of assessing a global model on the basis of a finite amount of noisy samples, are computationally prohibitive.

For these reasons, in recent years, interest has grown in pursuing alternatives to global modeling techniques. A demonstration is the popularity which approaches based on the *divide-and-conquer* strategy have gained in the research community. The divide-and-conquer strategy consists in attacking a complex problem by dividing it into simpler problems whose solutions can be combined to yield a solution to the original problem.

Instances of the divide-and-conquer approach are *local modeling* techniques. These techniques do not fit the whole dataset but perform the prediction of the output for specific test input values, also called *queries*. For that purpose, the database of observed input/output data is always kept in memory and the output prediction is obtained by interpolating the samples in the neighborhood of the query point.

This chapter presents a set of novel local modeling techniques, called *Lazy Learning* techniques, and their application to a number of experimental problems.

We show that, once a Lazy Learning perspective is adopted, the use of conventional linear techniques in nonlinear settings is easy and effective. A local Lazy Learning technique makes possible the adoption of linear methods to the widest range, not simply in the parametric identification procedure but also in the model validation step.

In terms of applications, we show that promising areas of application for these algorithms are regression modeling and data mining. Experimental results on simulated and real-world tasks, as well as the successful participation to two international competitions for learning techniques, demonstrate that these new techniques are competitive with existing methods and can be considered as a potential alternative to state-of-the-art approaches in a number of practical domains.

1 Introduction

The subject of this chapter is the automatic design of models from data. In particular, we focus on *supervised learning* problems, where the goal is to model the relation between a set of *input* variables, and one or more *output* variables, which are considered somewhat dependent on the inputs.

The idea of extracting useful knowledge from volumes of data is common to many disciplines, from statistics to machine learning, from econometrics to system identification and adaptive control. The procedure for finding useful patterns in data is called with different names by different communities; examples are knowledge extraction, pattern analysis, data processing. More recently, the set of computational techniques and tools to support the modeling of large amount of data has been grouped under the more general label of *data mining* (Fayyad *et al.*, 1996).

Modeling from data is often viewed as an art, mixing the insight of the expert with the information contained in the observations. Typically, a modeling process is not a sequential process but is better represented as a sort of loop with a lot of feedbacks and a lot of interactions with

the model designer. Different steps are repeated several times aiming to reach, through continuous refinements, a good description of the phenomenon underlying the data.

The process of modeling is made of a *preliminary* phase which brings the data from their original form to a structured configuration, and a *learning* phase which aims to select the *model*, or *hypothesis*, that best approximates the data (Fig. 1).

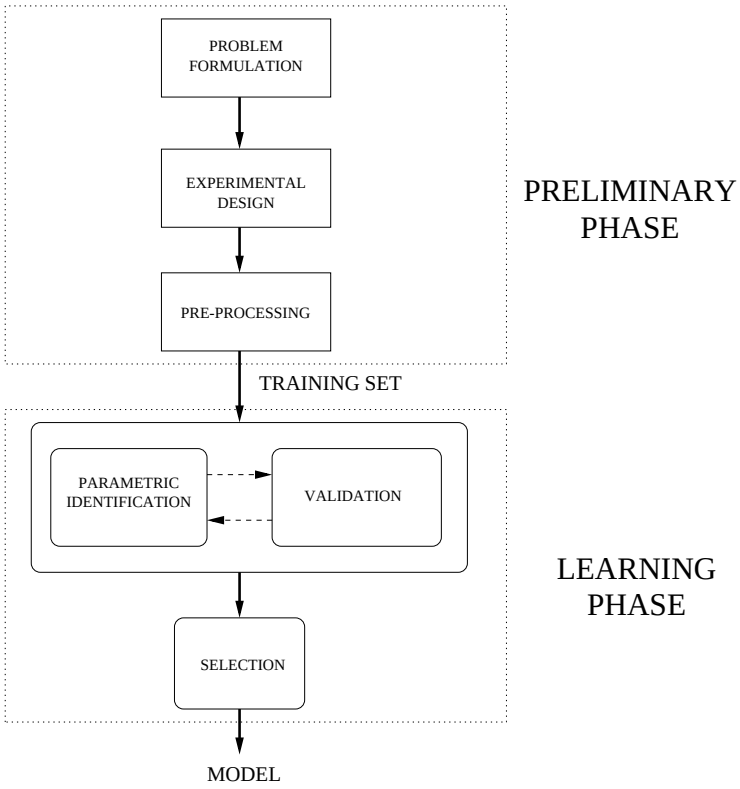

Figure 1. The modeling process and its decomposition in preliminary phase and learning phase.

The preliminary phase can be decomposed in the following steps:

Problem formulation. This is the first and somewhat the most important step of a learning procedure. Here, the model designer chooses a particular application domain, a phenomenon to be studied, and

hypothesizes the existence of an unknown dependency which is to be estimated from experimental data.

Experimental design. This procedure returns a dataset which is expected to be a representative sample of the phenomenon and to maximize the performance of the modeling process.

Pre-processing. In this step, raw data are cleaned to make learning easier. Pre-processing includes a large set of actions on the observed data, as noise filtering, outlier removal, missing data treatment, feature selection, and so on.

Once the preliminary phase has returned the dataset in a structured input/output form, called *training set* (Fig. 2), the learning phase begins. This chapter will focus exclusively on this second phase assuming that the preliminary steps have already been performed by the model designer.

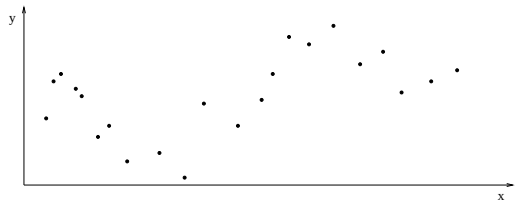

Figure 2. A training set for a learning problem with one input variable and one output variable. The dots represent the observed samples.

The learning procedure is essentially a search, in a space of possible model configurations, of the model which best represents the phenomenon underlying the data. As any other search task, the learning procedure requires both a *search space*, where the solution has to be found, and some *assessment criterion* which measures the quality of the solutions in order to select the best one.

The search space is defined by the designer using a set of nested classes with increasing complexity. For our introductory purposes, it is sufficient to consider here a *class* as a set of input/output models (e.g. the set of polynomial models) with the same *model structure* (e.g. second order

degree) and the *complexity* of the class as a measure of the set of input/output mappings which can approximated by the models belonging to the class.

As far as the assessment of a model is concerned, in this chapter we consider only quantitative criteria. We will assume that the goal of modeling is to attain a good *generalization*. This means that the selected model is expected to return an accurate prediction of the dependent variable when new values (i.e. not present in the training set) of the independent vector are presented.

Once the classes of models and the assessment criteria are fixed, the search for the best model starts. The search algorithm is made of two nested loops: *structural* identification and *parametric* identification.

Structural identification is the outer loop which seeks the model structure, i.e. the class, which is expected to have the best performance. It is composed of a *validation* phase, which assesses each model structure on the basis of the chosen assessment criterion, and a *selection* phase which returns the best model structure on the basis of the validation output. Parametric identification is the inner loop which returns the best model for a fixed model structure. The two procedures are intertwined since the structural identification requires the outcome of the parametric step in order to assess the goodness of a class. A well-known technique which iterates parametric identifications and validations is *cross-validation* (Stone, 1974).

In these terms, the learning process could appear as a standard problem of optimization. Unfortunately, reality is far more complex. In fact, since the amount of data is finite, there exists a strong correlation between the parametric and the structural steps, which makes non-trivial the problem of assessing and finally choosing the predictor.

In statistical literature, a learning procedure is commonly interpreted in terms of the bias/variance dilemma (Geman *et al.*, 1992). Results in statistical learning theory show that the generalization error of a learning machine is made of two components: a *bias* term, which measures the lack of representational power of the class of models and a *variance* term, which accounts for the sensitivity of the approximator to the noise in the

data. It follows that a complex model (e.g. a neural network with many layers) guarantees low bias at the cost of a large variance, while a simple model (e.g. a linear model) reduces the variance at the cost of an increased bias. The goal of structural identification is to find the optimal trade-off between nonlinearity and noise on the basis of a finite amount of data.

The considerations made so far are somewhat independent of the family of models taken into consideration. It is indeed evident that a learning procedure is quite sensitive to the class of models taken into consideration for fitting the data. For that reason, along the years statisticians and machine learning researchers have proposed a number of model architectures, with the aim of finding approximators able to combine high generalization with effective learning procedures. An outline of the learning architectures we will take into consideration in the next section is presented in Figure 3.

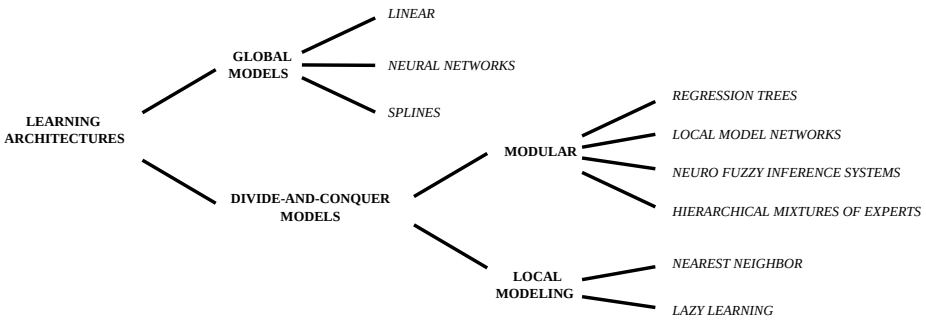

Figure 3. A panorama of learning architectures. For introductory purposes, a number of learning approaches (in italics) existing in literature have been grouped into more explicative families of approximators (in boldface).

2 From global to local modeling

A family of models traditionally used in supervised learning is the family of *global models* which describes the relationship between the input and the output values as a single analytical function over the whole input domain (Fig. 4). In general, this makes sense when it is reasonable to believe that a physical-like law describe the data over the whole set of operating conditions. Examples of well-known global parametric models

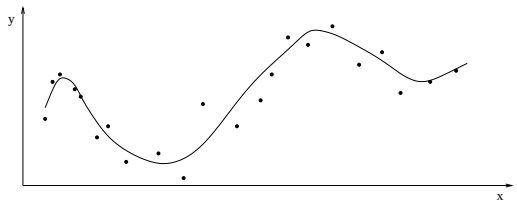

Figure 4. A global model (solid line) which fits the training set (dotted points) for a learning problem with one input variable (x-axis) and one output variable (y-axis).

in literature are linear models (Draper & Smith, 1981), nonlinear statistical regressions (Seber & Wild, 1989), Splines (Boor, 1978) and Neural Networks (Rumelhart *et al.*, 1986).

A nice property of global modeling is that, even for huge datasets, a parametric model can be stored in a small memory. Moreover, the evaluation of the model requires a short program that can be executed in a reduced amount of time. These features have undoubtedly contributed to the success of the global approach in years when most computing systems imposed severe limitations on users.

The problem of learning a parametric global model from a set of observed input/output data can be seen as a problem of function estimation, which consists in choosing from a given domain of parametric functions, the one which best approximates the unknown data distribution. Unfortunately, when only a finite amount of data is available the task of selecting the approximator which guarantees the best generalization is not trivial. For a generic global model, the parametric identification consists in a nonlinear optimization problem which is not analytically tractable due to the numerous local minima and for which only a suboptimal solution can be found through a slow iterative procedure. Similarly, the problem of selecting the best model structure in a generic nonlinear case cannot be handled in analytical form and requires time consuming validation procedures.

For these reasons, in recent years, alternatives to global modeling techniques, as the *divide-and-conquer* approach, gained popularity in the modeling community. The *divide-and-conquer* principle consists in attacking a complex problem by dividing it into simpler problems whose

solutions can be combined to yield a solution to the original problem. This principle presents two main advantages. The first is that simpler problems can be solved with simpler estimation techniques; in statistics this means to adopt linear techniques, well studied and developed over the years. The second is that the learning method can better adjust to the properties of the available dataset. Training data are rarely distributed uniformly in the input space. Whenever the distribution of patterns in the input space is uneven, a proper local adjustment of the learning algorithm can significantly improve the overall performance.

Two are the main instances of the divide-and-conquer principle: the modular approach, which originated in the field of system identification, and the local modeling approach, which was first proposed in the statistical nonparametric literature.

Modular architectures are input/output approximators composed of a number of modules which cover different regions of the input space. This is the idea of *operating regimes* which proposes a partitioning of the operating range of the system as a more effective way to solve modeling and control problems (Johansen & Foss, 1993). Fuzzy Inference Systems (Takagi & Sugeno, 1985), Local Model Networks (Murray-Smith, 1994), Radial Basis Functions (Moody & Darken, 1989), Classification and Regression Trees (Breiman *et al.*, 1984), and Hierarchical Mixture of Experts (Jordan & Jacobs, 1994) are well-known examples of this approach.

Although these architectures are a modular combination of local models, their learning procedure is still performed on the basis of the whole dataset. Hence, learning in modular architectures remains a functional estimation problem, with the advantage that the parametric identification can be made simpler by the adoption of local linear modules. However, in terms of structural identification the problem is still nonlinear and requires the same procedures used for generic global models.

A second example of divide-and-conquer methods are *local modeling* techniques (Cleveland & Loader, 1995), which turn the problem of function estimation in a problem of value estimation. The goal is not to model the whole statistical phenomenon but to return the best output for

a given test input, hereafter called the *query*. The motivation is simple: why should the problem of estimating the values of an unknown function at given points of interest be solved in two stages? Global modeling techniques first estimate the function (*induction*) and second estimate the values of the function using the estimated function (*deduction*). In this two-stage scheme one actually tries to solve a relatively simple problem (estimating the values of a function at given points of interest) by first solving, as an intermediate problem, a much more difficult one (estimating the function).

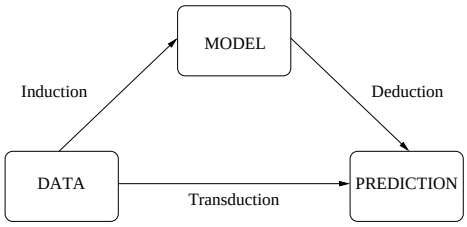

Figure 5. Function estimation (model induction + model evaluation) vs. value estimation (direct prediction from data).

Local modeling techniques take an alternative approach, defined as *transduction* by Vapnik (1995) (Fig. 5). They focus on approximating the function only in the neighborhood of the point to be predicted. So doing the approach requires to keep in memory the dataset for each prediction, instead of discarding it as in the global modeling case. At the same time, local modeling requires only simple approximators, e.g. constant and/or linear, to model the dataset in a neighborhood of the query point. An example of local linear modeling in the case of a single-input single-output mapping is presented in Fig. 6.

Many names have been used in the past to label variations of the local modeling approach: memory-based reasoning (Stanfill & Waltz, 1987), case-based reasoning (Kolodner, 1993), local weighted regression (Cleveland, 1979), nearest neighbor (Cover & Hart, 1967), just-in-time (Cybenko, 1996), lazy learning (Aha, 1997), exemplar-based, instance based (Aha, 1990),... These approaches are also called *nonparametric* in the literature (Hastie & Tibshirani, 1990; Scott, 1992), since they relax the assumptions on the form of a regression function, and let the data search for a suitable local description of the available data. Lo-

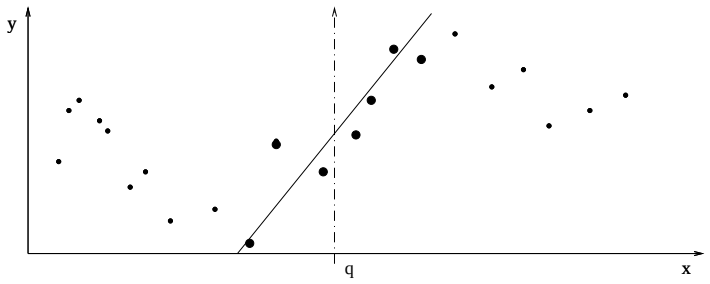

Figure 6. Local modeling of the input/output relationship between the input variable x and the output variable y, on the basis of a finite set of observations (dots). The value of the variable y for $x = q$ is returned by a linear model (solid line) which fits the samples in a neighborhood of the query point (bigger dots).

cal modeling techniques have also a long history of applications. They have been used for prediction (Farmer & Sidorowich, 1987), classification (Cover & Hart, 1967), regression (Cleveland, 1979), and control (Atkeson, 1989; Moore, 1991).

The following section will focus on the application of local modeling to the regression problem.

3 Local modeling for regression

The idea of local regression arose independently at different times and in different countries in the 19^{th} century. The early literature on smoothing by local fitting focused on one independent variable with equally spaced values. For an historical review of early work on local regression see (Cleveland & Loader, 1995). The modern view of smoothing by local regression has origins in the 1950's and 1960's in the kernel methods introduced in the density estimation setting. As far as regression is concerned, the first modern works on local regression were proposed by Nadaraya (1964) and Watson (1969).

3.1 Nadaraya-Watson estimators

Consider an independent variable $x \in \mathcal{X} \subset \Re^n$ and a dependent output $y \in \mathcal{Y} \subset \Re$.

Suppose that a *training set*

$$D_N = \{\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \ldots, \langle x_N, y_N\rangle\} \tag{1}$$

made of N pairs $\langle x_i, y_i\rangle \in \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ independent and identically distributed (i.i.d), has been observed.

Let K be a real-valued function

$$K : \Re^n \times \Re^n \times \Re^+ \to \Re^+ \tag{2}$$

where the first argument is a n-dimensional input, the second argument is typically called the *center* and the third argument, called *bandwidth* or *smoothing parameter*, controls the size of the local neighborhood. Assume that K satisfies

$$0 \leq K(x, q, B) \leq 1 \tag{3}$$

$$K(q, q, B) = 1 \tag{4}$$

The Nadaraya-Watson kernel regression estimator is given by

$$\hat{y}_q = \frac{\sum_{i=1}^{N} K(x_i, q, B) y_i}{\sum_{i=1}^{N} K(x_i, q, B)} \tag{5}$$

where N is the number of samples in the training set. The idea of kernel estimation is simple. Consider the case of a rectangular kernel in one dimension ($n = 1$). In this case the estimator (5) is a simple moving average with equal weights: the estimate at point q is the average of observations y_i corresponding to the x_i's belonging to the window $[q-B, q+B]$.

If $B \to \infty$ then the estimator tends to the average $h = \frac{\sum_{i=1}^{N} y_i}{N}$ and thus for mappings $f(\cdot)$ which are far from constant the bias become large. If B is smaller than the pairwise distance between the sample points x_i then the estimator reproduces the observations $h(x_i) = y_i$. In this extremal case the bias tends to zero at the cost of high variance. In general terms, by increasing B we increase the bias of the estimator, while by reducing B we obtain a larger variance. The optimal choice for B corresponds to an equal balance between bias and variance.

The Nadaraya-Watson estimator suffers from a series of shortcomings: it has large bias particularly in regions where the derivative of the regression function $f(x)$ is large or the input density distribution is nonuniform. A more severe problem is the large bias which occurs when estimating at a boundary region (Hastie & Loader, 1993).

Once the weakness of the local constant approximation was recognized, a more general local linear regression appeared in the late 1970's (Cleveland, 1979; Stone, 1977; Katkovnik, 1979). Work on local regression continued throughout the 1980's and 1990's, focusing on the application of smoothing to multidimensional problems (Cleveland & Devlin, 1988). Local linear regression is an attractive method both from the theoretical and the practical point of view. In practice, it adapts easily to various kinds of input distributions (e.g. random, fixed, highly clustered or nearly uniform) and it is not affected by boundary effects.

3.2 Parameter identification in local linear regression

Local linear regression uses a weighted least-squares regression (Myers, 1994), where weights are assigned to observed data according to Eq. (3). Local regression estimates can be expressed as

$$\hat{y}_q = q^T \hat{\beta} = q^T (X^T W^T W X)^{-1} X^T W^T W y \tag{6}$$

where implicitly the assumption that the inverse of $X^T W^T W X$ exists is made.

3.3 Structural identification in local regression

While the parametric identification in a local regression problem is quite simple and reduces to a weighted least-squares, there are several choices to be made in terms of model structure. Structural identification in local modeling involves, among other things, the selection of a family of local approximators (e.g. constant or linear), the selection of a metric to evaluate which examples are more relevant, and the selection of the *bandwidth* which indicates the size of the region in which the data are correctly modeled by members of the chosen family of approximators.

In formal terms, we say that structural identification in local modeling

addresses the local bias/variance dilemma where the bias accounts for the portion of error due to the nonlinearity of the mapping and the *variance* term accounts for the sensitivity of the approximator to the noise in the data.

These are the most relevant parameters in local structure identification:

- the kernel function K,
- the order of the local polynomial,
- the bandwidth parameter,
- the distance function,

In the following sections, we will present in detail the importance of these structural parameters and finally we will discuss the existing methods for tuning and selecting them.

3.3.1 The kernel function

Under the assumption that the data to be analyzed are generated by a continuous mapping $f(\cdot)$, we want to consider positive kernel functions $K(\cdot, \cdot, B)$ that are peaked at $x = q$ and that decay smoothly to 0 as the distance between x and q increases.

Some considerations can be made on how relevant is the kernel shape for the final accuracy of the prediction. First, it is evident that a smooth weight function results in a smoother estimate. On the other side, for hard-threshold kernels, as q changes, available observations abruptly switch in and out of the smoothing window. Second, it is relevant to have kernel functions with nonzero values on a compact bounded support rather than simply approaching zero for $|x - q| \to \infty$. This allows faster implementations, since points further from the query than the bandwidth can be ignored with no error.

Finally, most authors agree on attributing a low degree of sensitivity of the final prediction to the shape of the kernel function. Asymptotic results for nonparametric regression show that the overall form of the weight

function does not have an appreciable effect with respect to the mean squared error (Priestley & Chao, 1972). In particular, the kernel selection is less critical, the more the dimension of the input space increases and the more accurately the selection is done on the other parameters.

3.3.2 The local polynomial order

In Equation (6) we considered the case of a first-order local polynomial. However, the method can be easily extended to higher degrees. The choice of the local polynomial degree is a bias/variance trade-off. Generally speaking, a higher degree will generally produce a less biased, but a more variable estimate than a lower degree one.

Some asymptotic results in literature assert that a good practice in local polynomial regression is to adopt a polynomial order which differs of an odd degree from the order of the terms to be estimated (Fan & Gijbels, 1996). In practice, this means that if the goal of local polynomial regression is to estimate the value of the function in the query point, it is advisable to use orders of odd degree; otherwise, if the purpose is to estimate the derivatives in the query point it is better to fit with even degrees. However, others suggest in practical applications not to rule out any type of degree (Cleveland & Loader, 1995).

In the previous sections, we already introduced some consideration on degree zero fitting. This choice very rarely appears to be the best choice in terms of prediction, even if it presents a strong advantage in computational terms. By using a polynomial degree greater than zero we can typically increase the bandwidth by a large amount without introducing intolerable bias. Despite the increased number of parameters the final result is smoother thanks to an increased neighborhood size.

A degree having an integer value is generally assumed to be the only possible choice for the local order. However, the accuracy of the prediction results to be highly sensitive to discrete changes of the degree.

A possible alternative is *polynomial mixing*, proposed in global parametric fitting by Mallows (1974) and in local regression by Cleveland and Loader (Cleveland & Loader, 1995). Polynomial mixings are polynomials of fractional degree $p = m + c$ where m is an integer and $0 < c < 1$.

The mixed fit is a weighted average of the local polynomial fits of degree m and $m+1$ with weight $1-c$ for the former and weight c for the latter

$$f_p(\cdot) = (1-c)f_m(\cdot) + cf_{m+1}(\cdot) \tag{7}$$

We can choose a single mixing degree for all x or we can use an adaptive method by letting p vary with x.

3.3.3 The bandwidth

A natural question is how wide the local neighborhood should be so that a local approximation holds. This is equivalent to asking how large the bandwidth parameter should be in the function $K(\cdot)$. If we take a small bandwidth B, we are able to cope with the eventual nonlinearity of the mapping, that is, in other terms, we keep the modeling bias small. However, since the number of data points falling in this local neighborhood is also small, we cannot average out the noise from the samples and the variance of our prediction will be consequently large (Fig. 7). On the

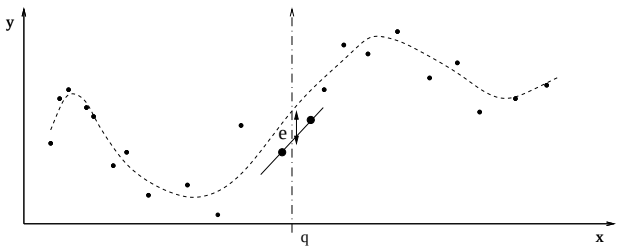

Figure 7. Local modeling with a too narrow bandwidth. The mapping (dashed line) is approximated by a linear model (solid line) which fits a too small number of noisy neighbors (bigger dots). The prediction in q is consequently poor.

other hand, if the bandwidth is too large, we could smooth excessively the data, then introducing a large modeling bias (Fig. 8).

In the limit case of an infinite bandwidth, for example, a local linear model turns to be a global linear fitting which, by definition, cannot take into account any type of nonlinearity.

A vast amount of literature has been devoted to the bandwidth selection problem. Various techniques for selecting smoothing parameters have

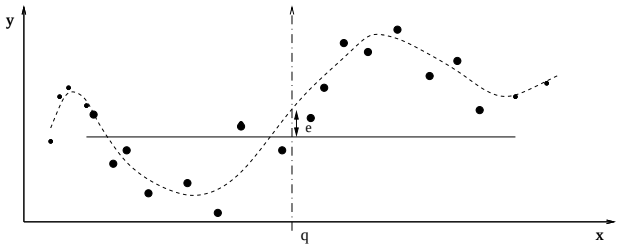

Figure 8. Local modeling with a too large bandwidth. The mapping (dashed line) is approximated by a linear model (solid line) which fi ts an excessively large number of neighbors (bigger dots). The prediction in q is consequently poor.

been proposed during the last decades in different setups, mainly in kernel density estimation (Jones *et al.*, 1995) and kernel regression.

Two are the main strategies for the bandwidth selection:

Constant bandwidth selection. The bandwidth B is independent of the training set D_N and the query point q.

Variable bandwidth selection. The bandwidth is a function $B(D_N)$ of the dataset D_N. For a variable bandwidth a further distinction should be made between the local and global approach.

1. A *local variable* bandwidth $B(D_N, q)$ is not only function of the training data D_N but also changes with the query point q. An example is the nearest neighbor bandwidth selection where the bandwidth is set to be the distance between the query point and the k^{th} nearest sample (Stone, 1977).
2. A *global variable bandwidth* is a function $B(D_N)$ of the data set but is the same for all the queries. However, a further degree of distinction should be made between a *point-based* case where the bandwidth $B(x_i)$ changes with the samples in the training set and an *uniform* case where B is the same for all the samples contained in D_N.

A constant bandwidth is easy to interpret and can be suffi cient if the unknown curve is not too wiggly, i.e. has an high smoothness. Such a

bandwidth, however, fails to do a good job when the unknown curve has a rather complicate structure. To capture the complexity of such a curve a variable bandwidth is needed. A variable bandwidth allows for different degrees of smoothing, resulting in a possible reduction of the bias at peaked regions and of the variance at flat regions. Further a variable local bandwidth can adapt to the data distribution, to different level of noise and to changes in the smoothness of the function. Fan and Gijbels (1992) argue for point-based in favor of query-based local bandwidth selection mainly for computational effi ciency reasons.

3.3.4 The distance function

The performance of any local method depends critically on the choice of the distance function $d : \Re^n \times \Re^n \to \Re$. In the following we will defi ne some distance functions for ordered inputs:

Unweighted Euclidean distance

$$d(x,q) = \sqrt{\sum_{j=1}^{n}(x_j - q_j)^2} = \sqrt{(x-q)^T(x-q)} \tag{8}$$

Weighted Euclidean distance

$$d(x,q) = \sqrt{(x-q)^T M^T M(x-q)} \tag{9}$$

The unweighted distance is a particular case of the weighted case for M diagonal with $m_{jj} = 1$.

Unweighted L_p norm (Minkowski metric)

$$d(x,q) = \left(\sum_{j=1}^{N}|x_j - q_j|\right)^{\frac{1}{p}} \tag{10}$$

Weighted L_p norm It is computed through the unweighted norm $d(Mx, Mq)$.

It is important to remark that when an entire column of M is zero, all points along the corresponding direction get the same relevance in the distance computation. Also, notice that once the bandwidth is selected, some terms in the matrix M can be redundant parameters of the local learning procedure. The redundancy can be eliminated by requiring the determinant of M to be one or fixing some element of M.

Atkeson *et al.* (1997) distinguish between three ways of using distance functions:

Global distance function. The same distance is used at all parts of the input space.

Query-based distance function. The distance measure is a function of the current query point. Examples are in (Stanfill & Waltz, 1987; Hastie & Tibshirani, 1996; Friedman, 1994).

Point-based local distance functions. Each sample in the training set has an associated distance metric (Stanfill & Waltz, 1987). This is typical of classification problems where each class has an associated distance metric (Aha, 1989; Aha, 1990).

3.3.5 The selection of local parameters

As seen in the previous sections, there are several parameters that affect the accuracy of the local prediction. Generally, they cannot be selected and/or optimized in isolation as the accuracy depends on the whole set of structural choices. At the same time, they do not all play the same role in the determination of the final estimation. It is common belief in local learning literature that the bandwidth and the distance function are the most important parameters. The shape of the weighting function, instead, plays a secondary role.

In the following we will mainly focus on the methods existing for bandwidth selection. They can be classified in

Rule of thumb methods. They provide a crude bandwidth selection which in some situations may result sufficient. Examples of rule

of thumb is provided in (Fan & Gijbels, 1995) and in (Hardle & Marron, 1995).

Plug-in techniques. The exact expression of optimal bandwidth can be obtained from the asymptotic expressions of bias and variance (Ruppert & Wand, 1994; Fan & Gijbels, 1996), which unfortunately depends on unknown terms. The idea of the direct plug-in method is to replace these terms with estimates. This method was first introduced by (Woodrofe, 1970) in density estimation. Examples of plug-in methods for non parametric regression are reported in (Ruppert *et al.*, 1995).

Data-driven estimation. It is a selection procedure which estimates the generalization error directly from data. Unlike the previous approach, this method does not rely on the asymptotic expression but it estimates the values directly from the finite data set. To this group belong methods like cross-validation, Mallow's C_p, Akaike's AIC and other extensions of methods used in classical parametric modeling.

The debate on the superiority of plug-in methods over data-driven methods is still open and the experimental evidences are contrasting. Results on behalf of plug-in methods come from (Woodrofe, 1970; Ruppert *et al.*, 1995; Park & Marron, 1990). On the other side Loader (1987) showed how the supposed superior performance of plug-in approaches is a complete myth. The use of cross-validation for bandwidth selection has been investigated in several papers, mainly in the case of density estimation (Jones *et al.*, 1995). In regression, an adaptation of Mallow's C_p was introduced by Rice (1984) for constant fitting and by Cleveland and Devlin (1988) in local polynomial regression. Cleveland and Loader (1995) suggested local C_p and local PRESS for choosing both the degree of local polynomial mixing and the bandwidth.

Plug-in methods are built on a series of assumptions on the statistical process underlying the data set and on theoretical results which are more reliable the larger is the number of points. We believe that in a common black-box configuration where no a priori information is available, the adoption of data driven techniques can be a promising approach to the problem.

In the following we will focus on the automatic choice of the bandwidth from data, neglecting any a priori information. There are several ways in which data-driven methods can be used for structural identification. Atkeson *et al.* (1997) distinguish between

Global tuning. The structural parameters are tuned by optimizing a data driven assessment criterion on the whole data set. An example is the General Memory Based Learning (GMBL) described in (Moore *et al.*, 1992).

Query-based local tuning. The structural parameters are tuned by optimizing a data driven assessment criterion query-by-query.

Point-based local tuning. A different set of structural parameters is associated to each point of the training set.

4 The lazy idea

A particular class of local modeling algorithms are the so-called lazy techniques (Aha, 1997). They are *query-based* local learning algorithms, i.e. they defer the whole learning process until a specific query needs to be answered. Once the prediction is returned, they discard both the answer and the constructed model.

All the local modeling approaches perform a certain amount of operations when a prediction for a specific query is required. However, we wish to distinguish between simple computational processing, such as finding the set of neighbors in the dataset and more complex learning procedures, such as performing locally the structural identification. While the former is in some sense analogous to the kind of operations we find in other parametric models (e.g. the propagation of the values in a Neural Networks or the activation of the kernel functions in a Basis Function Network), the latter is in our view more distinctive of lazy methods. It is possible to encounter different degrees of "laziness" in local learning algorithms. For instance, a k Nearest Neighbor (k-NN) algorithm is hardly a lazy approach since, after the query is presented, it requires only a reduced amount of learning procedure, namely the computation of an

average. On the contrary, a local method, which waits for the query to select the number of neighbors and/or other structural parameters, presents in our view a higher degree of laziness.

5 The Lazy Learning algorithm

This section will present our Lazy Learning (LL) algorithm for local learning in regression problems (Bontempi *et al.*, 1999e; Birattari *et al.*, 1999). In particular, the data analysis problems we want to address with our approach have these characteristics

- Supervised learning problems with multi dimensional input.
- Finite amount of training data.
- Unequally spaced observations in the input domain.
- Condition of realistic noise on data. In particular we assume the data to be generated as follows:

$$y_i = f(x_i) + w_i, \tag{11}$$

  where $\forall i$, w_i is a random variable such that $E[w_i] = 0$ and $E[w_i w_j] = 0$, $\forall j \neq i$, and such that $E[w_i^m] = \mu_m(x_i)$, $\forall m \geq 2$, where $\mu_m(\cdot)$ is the unknown m^{th} moment of the distribution of w_i and is defined as a function of x_i. In particular for $m = 2$, the last of the above mentioned properties implies that no assumption of global homoscedasticity is made.

- No further knowledge on f and/or w is available a priori.

In this real, even pessimistic, situation the analyst cannot rely on his insight to perform the set of choices that characterize a local learning procedure. As a consequence we will focus only on a data driven scenario where parametric and structural identification are performed on the basis of the available training set.

The original Lazy Learning method we introduce is:

1. a local modeling approach which locally combines constant (Fig.9) and linear models (Fig.10),
2. a lazy method where the structural identification is repeated each time a query is presented.

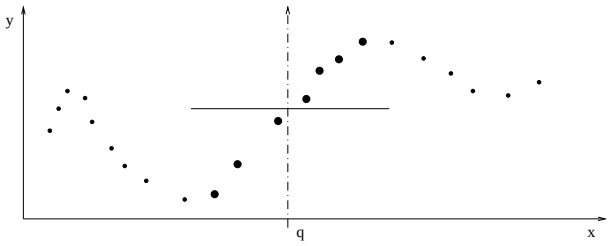

Figure 9. Local constant model. The constant model estimates the value of the function in q by fitting the 7 nearest neighbors (bigger dots).

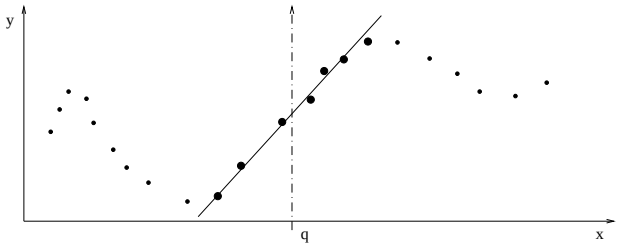

Figure 10. Local linear model. The linear model estimates the value of the function in q by fitting the 7 nearest neighbors (bigger dots).

In terms of the parametric identification procedure, the Lazy Learning method adopts standard techniques of linear regression, analogously to other local modeling techniques.

The novelty of the LL approach stays mainly in its structural identification procedure. It is well-known that the performance of a local approximator is quite sensitive to the structural identification choices performed by the designer. We saw in Section 3.3 that the structural parameters do not all play the same role in the determination of the final accuracy and that a common belief is that the bandwidth and the distance metric be the most relevant parameters.

Our novel Lazy Learning method proposes a lazy structural identification which selects the bandwidth parameter in order to minimize the generalization error of the resulting approximator.

The *query-by-query bandwidth selection* method we propose consists of three steps:

1. the *generation* of local candidates,
2. the local *validation* through linear cross-validation, and
3. the final prediction obtained either by *selection* or *combination*.

Each time a prediction is required for a specific query point, a set of local models, each including a different number of neighbors, is generated and identified. The model identification is based on the *recursive least-squares* algorithm. This is an appealing and efficient solution to the intrinsically incremental problem of identifying a sequence of local models, centered in the query point, each including a growing number of neighbors. So doing, we reduce the bandwidth selection problem to the selection of the number k of neighboring examples which are given a non-zero weight in the local modeling procedure.

Once the candidate models have been generated, the generalization ability of each of them is assessed through a local cross-validation procedure. Here we use the leave-one-out PRESS statistic (Allen, 1974). It is worth noticing that this leave-one-out procedure does not involve any significant computational overload, since the PRESS statistic uses partial results returned by the recursive least-squares algorithm.

Finally two possible variants in local model selection are proposed: a *competitive* approach and a *cooperative* approach.

The competitive approach is based on the *winner-takes-all* strategy, which selects among the local candidates the best model in terms of cross-validation performance.

The cooperative approach is based on the application of the *combination of estimators* technique to local modeling. The rationale is to combine the

outputs of the generated local models in order to increase the accuracy of the final prediction, by reducing the variance of the estimators.

6 Local model generation

Goal of the model generation procedure is to generate a set of candidate model structures among which the best one is to be selected. The more this procedure is effective, the easier will be the selection of a powerful structure at the end of the whole identification. Traditionally there have been a number of popular ways to search through a large collection of model structures. Maron and Moore (1997) distinguish between two main methods of model generation: (i) *brute force* methods, which require a heavy computational effort to perform an exhaustive search in the space of model structures and (ii) *search* methods which generate a number of possible candidates in a space defined with respect to some structural parameter (e.g. the number of neurons in a Feed Forward Neural Network or the number of basis functions in a BFN).

Here, we will adopt a search approach where the structural parameter is the bandwidth B of the kernel. Let us assume that

1. a metric on the input space $\Re^n$ is given (Section 3.3.4),
2. the quantity $d(x, q)$ denotes the distance from the query point to the x point,
3. the pair $\langle x(k), y(k) \rangle$ is the k^{th} nearest neighbor of the query point in the training set, i.e. that
$$d(x(i), q) \leq d(x(j), q) \qquad \forall i \leq j \tag{12}$$
4. the bandwidth B is defined as a function of the neighbors location, e.g. B is equal to the distance of the query point to the k^{th} neighbor
$$B(k) = d(x(k), q) \tag{13}$$
5. the bandwidth $B(k)$ is allowed to range over a domain, whose lower bound is the minimum number k_m of neighbors and upper bound is the maximum number k_M of neighbors.

Hence, the problem of local structural identification can be seen as a problem of bandwidth selection, and the problem of bandwidth selection as a search problem in the space of $B(k)$, where the number of neighbors k takes value between a minimum k_m and a maximum k_M.

By ranging the quantity k over the interval $[k_m, k_M]$, the local model generation procedure returns a set of local models, whose parameters are fitted on the set of neighbors selected by the bandwidth $B(k)$.

We will consider both a set of constant fitting models $\mathcal{C}_k$, $k = k_m^c, \ldots, k_M^c$, (Fig. 11) and a set of linear fitting models $\mathcal{L}_k$, $k = k_m^l, \ldots, k_M^l$ (Fig. 12)[1]. The parametric estimation requires the computa-

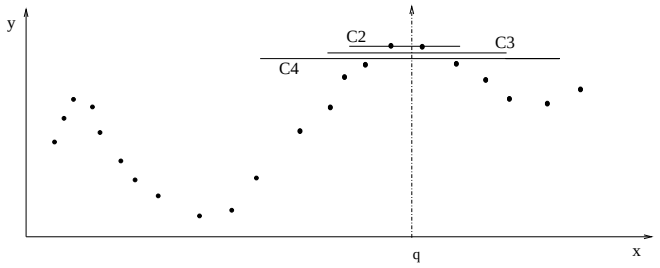

Figure 11. Generation of local constant models. The figure represents three constant models $\mathcal{C}_k$, having different number of neighbors ($k = 2, 3, 4$), which estimate the value of the function in q.

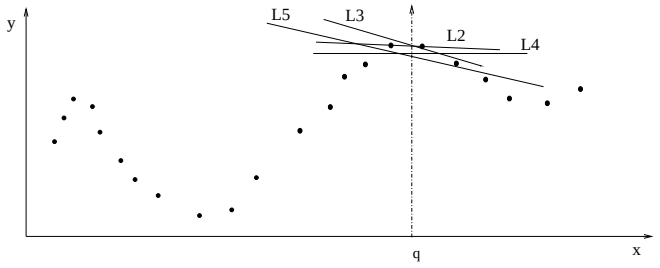

Figure 12. Generation of local linear models. The figure represents four linear models $\mathcal{L}_k$, having different number of neighbors ($k = 2, 3, 4, 5$), which estimate the value of the function in q.

tion of Eq. (5) for each constant model $\mathcal{C}_k$ and of Eq. (6) for each linear

[1]For the sake of simplicity, we will not distinguish between the constant and the linear indices k^c and k^l in the rest of the chapter. However, the distinction will result useful in the description of the experiments.

model $\mathcal{L}_k$. This is an heavy computational task which can be speeded up thanks to some nice properties of linear least-squares methods. An example is the adoption of the recursive least-squares algorithm to perform an incremental parametric identification of the set of local models $\mathcal{C}_k$ and $\mathcal{L}_k$. The recursive least-squares technique for local model generation (Bontempi *et al.*, 1998; Bontempi *et al.*, 1999b; Birattari *et al.*, 1999) will be discussed in detail in the following section.

6.1 Recursive least-squares for model generation

Recursive least-squares (RLS) algorithms have been developed in model identification and adaptive control literature (Goodwin & Sin, 1984). Typically, RLS are used to identify a linear model when data are not available as a batch but are observed sequentially in time.

Here, we have not a temporal sequence, nevertheless query neighbors can be ordered according to the distance $d(x_i, q)$ so to provide a spatial sequence (12). Once the neighbors have been ordered, a standard RLS can be used, not to update a model from time t to time $t+1$, but to obtain the parameters of the model fitted on the $k+1$ nearest neighbors by updating the parameters of the model with k examples. In this context, we will then speak of recursive least-squares *in space* as opposed to the more traditional recursive least-squares *in time*.

The main assumption to be made for using a recursive approach in the local model generation is the adoption of an uniform weight kernel

$$K(x_i, q, B) = \begin{cases} 1 & \text{if } d(x_i, q) \leq B, \\ 0 & \text{otherwise;} \end{cases} \tag{14}$$

The main advantage deriving from the adoption of the weight function defined in Eq. (14), is that, simply by updating the parameters of the model identified using the k nearest neighbors, it is straightforward and inexpensive to obtain the parameters for the model with the $k+1$ nearest neighbors.

Let us see the details for the constant and the linear case.

6.1.1 Recursive constant model generation

Consider a set of local constant models $\mathcal{C}_k$, an uniform kernel (14) and a bandwidth (13) which ranges over $k = k_m, \ldots, k_M$. Fixed a value of k, the parametric identifi cation of the constant model $\mathcal{C}_k$ is performed by computing the mean of the outputs of the k nearest neighbors. Hence, the prediction returned by $\mathcal{C}_k$ is

$$\hat{y}_q^c(k) = \frac{\sum_{i=1}^{k} y(i)}{k} \tag{15}$$

where $y(i)$ is the output value of the i^{th} nearest neighbor of q in the training set.

If we aim to identify the parameters of the model $\mathcal{C}_{k+1}$ on the basis of the parameters of $\mathcal{C}_k$, a recursive formulation is straightforward

$$\hat{y}_q^c(k+1) = \frac{k\hat{y}_q^c(k) + y(k+1)}{k+1} \tag{16}$$

where $y(k+1)$ is the output of the $(k+1)^{\text{th}}$ nearest neighbor.

6.1.2 Recursive linear model generation

In the linear case, given a query point q, and under the hypothesis of a local homoscedasticity of w_i, the parameter $\hat{\beta}$ of a local linear approximation of $f(\cdot)$ in a neighborhood of q can be obtained by solving the local polynomial regression:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\beta} \sum_{i=1}^{k} \left\{ \left(y(i) - x(i)^T\beta\right)^2 K\left(x(i), q, B\right) \right\}, \tag{17}$$

where $K(\cdot)$ is the kernel function, B is the bandwidth, and where a constant value 1 has been appended to each vector x_i in order to consider a constant term in the regression.

In matrix notation, the solution of the above stated weighted least-squares problem is given by

$$\hat{\beta} = (X^T W^T W X)^{-1} X^T W^T W y = (Z^T Z)^{-1} Z^T v = V Z^T v, \tag{18}$$

where X is a matrix whose i^{th} row is $x(i)^T$, y is a vector whose i^{th} element is $y(i)$, W is a diagonal matrix whose i^{th} diagonal element is $w_{ii} = \sqrt{K\left(x(i), q, B\right)}$, $Z = WX$, $v = Wy$, and the matrix $X^T W^T W X = Z^T Z$ is assumed to be non-singular so that its inverse $V = (Z^T Z)^{-1}$ is defined.

Given the local least-squares parameter $\hat{\beta}$, the prediction in q is finally given by:

$$\hat{y}_q^l = q^T \hat{\beta}. \tag{19}$$

Consider now a set of local linear models $\mathcal{L}_k$, an uniform kernel (14) and a bandwidth (13) which ranges over $k = k_m, \ldots, k_M$. Fixed a value of k, only the k nearest neighbors of q are considered in Eq. (17) to identify the vector of parameters $\hat{\boldsymbol{\beta}}$. In matrix notation this means that the k diagonal terms of the matrix W which correspond to the k nearest neighbors take a unit value while all the other diagonal terms are null.

Let us denote by $\hat{\boldsymbol{\beta}}(k)$ the least-squares vector of parameters $\hat{\boldsymbol{\beta}}$ identified with k neighbors. The recursive least-squares algorithm provides a fast way to identify the vector $\hat{\boldsymbol{\beta}}(k+1)$ using the $k+1$ nearest neighbors on the basis of the vector $\hat{\beta}(k)$ identified using the k nearest neighbors. By performing a step of the standard recursive least-squares algorithm we have:

$$\begin{cases} V(k+1) = V(k) - \dfrac{V(k)x(k+1)x^T(k+1)V(k)}{1 + x^T(k+1)V(k)x(k+1)} \\ \gamma(k+1) = V(k+1)x(k+1) \\ e(k+1) = y(k+1) - x^T(k+1)\hat{\beta}(k) \\ \hat{\beta}(k+1) = \hat{\beta}(k) + \gamma(k+1)e(k+1) \\ \hat{y}_q^l(k+1) = q^T \hat{\beta}(k+1) \end{cases} \tag{20}$$

where $V(k) = (Z^T Z)^{-1}$, $x(k+1)$ is the $(k+1)^{\text{th}}$ nearest neighbor of the query point, and $\hat{y}_q^l(k)$ denotes the prediction in the query point returned by a linear model estimated on the basis of the k nearest neighbors.

Once an initialization $\hat{\beta}(0) = \tilde{\beta}$ and $V(0) = \tilde{V}$ is given, Eq. (20) recursively evaluates, for different values of k, (i) a first order approximation

$\hat{\boldsymbol{\beta}}(k)$ of the regression function $f(\cdot)$ in q and (ii) a prediction $\hat{y}_q^l(k)$ of the value of the regression function in the query point.

Notice that $\tilde{\beta}$ is an *a priori* estimate of the parameter and $\tilde{V}$ is the covariance matrix that reflects the reliability of $\tilde{\beta}$ (Bierman, 1977). If a priori information is not available, the following initializations are usually adopted: $\tilde{\beta} = 0$ and $\tilde{V} = \lambda I$, with λ large and where I is the identity matrix.

7 Local model validation

In the previous section we introduced recursive least-squares as an effective method for generating local model candidates. These models have now to be validated in order to proceed to the final model selection. We will consider in the following sections some alternative methods for validating the set of candidates.

7.1 Leave-one-out for local models

This section focuses on the adoption of leave-one-out for cross-validated assessment of local constant and linear models in the case of the uniform kernel (14) and the bandwidth (13).

The formula of leave-one-out for constant models can be easily derived. Since the constant fit with the j^{th} point set aside is

$$\hat{y}^{-j} = \frac{\sum_{i \neq j} y(i)}{k-1} \tag{21}$$

the j^{th} element of the leave-one-out vector error is

$$e_j^{\text{loo}}(k) = y(j) - \frac{\sum_{i \neq j} y(i)}{k-1} = k \frac{y(j) - \hat{y}_q^c(k)}{k-1} \tag{22}$$

where $\hat{y}_q^c(k)$ is given in (15) and k is the number of neighbors used to estimate the constant model. A recursive formulation of the $e_j^{\text{loo}}(k)$ quantity, which directly obtains $e_j^{\text{loo}}(k)$ from $e_j^{\text{loo}}(k-1)$ with no intermediate step (15), is given in (Birattari & Bontempi, 1999).

As far as linear models are concerned, an efficient computation of the leave-one-out error is given by the PRESS statistic (Allen, 1974) , which is a fast way to compute the leave-one-out assessment of a linear approximator. The main strength of this formulation is that the assessment of a local linear model can be obtained as a by-product of the model identification, at a reduced computational cost.

Using the PRESS statistic, it is possible to calculate the l-o-o error without explicitly identifying the parameters $\hat{\beta}^{-j}(k)$

$$e_j^{\text{loo}}(k) = y_j - x_j^T \hat{\beta}^{-j}(k) = \frac{y(j) - x(j)^T \hat{\beta}(k)}{1 - H_{jj}} \tag{23}$$

where H_{jj} is the j^{th} diagonal element of the *Hat matrix* $H = ZVZ^T = Z(Z^TZ)^{-1}Z^T$.

By (23) and (20) we have a formulation of the PRESS statistic which takes advantage of the recursive formulation of the model generation procedure:

$$e_j^{\text{loo}}(k) = y(j) - x(j)^T \hat{\beta}^{-j}(k) = \frac{y(j) - x(j)^T \hat{\beta}(k)}{1 - x(j)^T V(k) x(j)} \qquad \forall j : d(x_j, q) \leq B(k) \tag{24}$$

It is easy to show that the model generation procedure described in the previous section returns as a by-product all the elements necessary for the leave-one-out computation. Equation (16) contains all that is required in (22) for the constant leave-one-out computation. At the same time Equation (20) returns the matrix $V(k)$ and the vector $\hat{\boldsymbol{\beta}}(k)$ which make possible, by Equation (24), the direct calculation of the linear leave-one-out cross-validation errors without the need of any further model identification.

To make easier the following analysis it is useful to define, for each value of k, the following quantities:

1. the $[k \times 1]$ vector

$$e^{\text{loo}}(k) = \left\{e_j^{\text{loo}}(k)\right\} \qquad j = 1, \ldots, k \tag{25}$$

that contains all the leave-one-out errors (24) associated to the model identified with the k nearest neighbors,

2. the cross-validated mean squared error

$$\mathrm{MSE}_{\mathrm{loo}}(k) = \frac{\sum_{j=1}^{k} \omega_j \left(e_j^{\mathrm{loo}}(k)\right)^2}{\sum_{j=1}^{k} \omega_j} \tag{26}$$

which estimates the generalization error of the local model fitted on the k nearest neighbors.The weights ω_j are used to discount the contribution of the j^{th} error $e_j^{\mathrm{loo}}(k)$ according to the distance of the j^{th} neighbor from the query point (Atkeson *et al.*, 1997).

8 Local model selection

The model generation procedure, described in the previous section, returns, for the number k of neighbors ranging between k_m and k_M, a set of constant model predictions $\hat{y}_q^c(k)$ and a set of linear model predictions $\hat{y}_q^l(k)$, each associated with the respective leave-one-out error vectors $e^{\mathrm{loo}}(k)$, $k = k_m, \ldots, k_M$.

The goal of local model selection is to use all this information in order to return a final prediction $\hat{y}_q$ of the value of the regression function in the query point. Two main paradigms deserve to be considered: the first is based on the selection of the *best* approximator according to a predefined criterion, the second returns the prediction as a combination of different local models.

8.1 The winner-takes-all approach

The *winner-takes-all* selection paradigm consists in comparing the whole set of models $\mathcal{C}_k$ and $\mathcal{L}_k$, $k = k_m, \ldots, K_M$ and in selecting the local model which presents the minimum estimated generalization error (26). The final prediction $\hat{y}_q$ will be the one returned by the selected local model.

A classical error criterion to assess the performance of a local model is the cross-validated estimate of the *mean squared error* criterion. Then

the predicted output $\hat{y}_q$ is the output of the model

$$\mathcal{M}_k \in \{\mathcal{C}_k, \mathcal{L}_k : k = k_m, \ldots, k_M\}$$

where

$$\mathcal{M}_k = \arg\min_{\mathcal{C}_k, \mathcal{L}_k} \mathrm{MSE}_{\mathrm{loo}}(k) \tag{27}$$

where $\mathrm{MSE}_{\mathrm{loo}}(k)$ is defined in (26). In this approach the only information extracted from the vector of leave-one-out errors is the sample mean, intended as the most relevant statistic describing the error distribution. In alternative, to select the bandwidth of the neighborhood region we could use the information contained in the e^{loo} vectors to a greater extent, e.g. by using an empowered statistical procedure. This alternative winner-takes-all method is discussed in (Bontempi *et al.*, 1998; Bontempi, 1999)

8.2 The local model combination

An alternative to the winner-takes-all paradigm is provided by the local *combinations of estimates* (Wolpert, 1992). The idea of combination is common to a large amount of literature in neural networks and machine learning (Breiman, 1996). What is original here is the application of this idea to a local modeling setting. The set of estimators is not made of a number of global models but of a subset of the local models $\mathcal{C}_k$ and $\mathcal{L}_k$ generated as in Section 6.

By adopting the *mean squared error* criterion, the final prediction $\hat{y}_q$ is obtained as a weighted average of the best b models, where b is a parameter of the algorithm[2]. Suppose the predictions $\hat{y}_q(k)$ and the error vectors $e^{\mathrm{loo}}(k)$ have been ordered creating a sequence of integers $\{k_i\}$ so that $\mathrm{MSE}_{\mathrm{loo}}(k_i) \leq \mathrm{MSE}_{\mathrm{loo}}(k_j)$, $\forall i < j$. Assume that the local model predictions are unbiased and uncorrelated. Hence, the combined prediction in the query point q is given by

$$\hat{y}_q = \frac{\sum_{i=1}^{b} \zeta_i \hat{y}_q(k_i)}{\sum_{i=1}^{b} \zeta_i}, \tag{28}$$

[2] As an alternative we can decide to combine the best b^c constant models and the best b^l linear models.

where the weights are the inverse of the mean squared errors:

$$\zeta_i = \frac{1}{\mathrm{MSE}_{\mathrm{loo}}(k_i)}$$

This is an example of the *generalized ensemble method* (GEM) (Perrone & Cooper, 1993).

9 Experiments

A typical issue in machine learning is the assessment of new learning techniques with respect to more conventional methods.

In this section we present an experimental session, made of 23 supervised learning benchmarks, which aims to compare under the same operating conditions the Lazy Learning approach with a number of state-of-the-art algorithms.

In some cases, as for Regression Trees and Feed Forward Neural Networks, we employed commercial softwares; in other cases, as for Basis Function Networks and Mixtures of Experts, we implemented the learning algorithms.

For many state-of-the-art approaches the implementation of the parameter identification procedure is relatively standard, nevertheless it is hard to choose one structural identification procedure among the large amount of existing approaches. For that reason and the high sensitivity of the final accuracy to the structural identification strategy, we decide to adopt for non-lazy methods the *a posteriori best-case* approach. This approach consists in generating a set of alternative structures and then selecting the one which a posteriori behaves the best on the test set. Note that this procedure is strongly biased in favor of these techniques, since the portion of generalization error due to a wrong model selection is not taken into account. Therefore, the experimental results obtained for these state-of-the-art approaches can be considered as a sort of optimistic bound for their real performance.

9.1 Datasets

The experimental session contains 23 datasets. The first fifteen datasets are from the UCI[3] Repository of machine learning databases (Merz & Murphy, 1998), with the exception of the *Ozone* database provided by Leo Breiman. These 15 datasets are made of input/output samples collected experimentally in real modeling problems.

The 8 last datasets, forming the so-called Kin family, are from the Delve (Data for Evaluating Learning in Valid Experiments) repository[4]. They were synthetically generated as variations of the same model: a simulation of the forward dynamics of an 8 link all-revolute robot arm. The task in all the 8 datasets is to predict the distance of the end-effector from a target. The inputs are variables like joint positions, twist angles, etc.

The Kin family has been specifically generated for a supervised learning problem and so the individual datasets span the corners of a cube whose dimensions represent: (i) the number of inputs (8 or 32), (ii) the degree of non-linearity (fairly linear or non-linear), and (iii) the amount of synthetic noise in the output (moderate or high). All datasets in this family have "Kin" as the base of their name (Kinematics). An underscore (_) is appended to this name, followed by:

1. An integer value signifying the number of input attributes in each case, for example '32'.
2. One of the characters 'f' or 'n' signifying 'fairly linear' or 'non-linear', respectively.
3. One of the characters 'm' or 'h' signifying 'medium unpredictability/noise' or 'high unpredictability/noise', respectively.

A summary of the characteristics of each dataset, in terms of number of inputs and the number of samples, is presented in Table 1.

[3] http://www.ics.uci.edu/~mlearn/MLRepository.html
[4] http://www.cs.toronto.edu/~delve/

Dataset	Number of examples	Number of regressors
Housing	330	8
Cpu	506	13
Prices	209	6
Mpg	159	16
Servo	392	7
Ozone	167	8
Bodyfat	252	13
Pool	253	3
Energy	2444	5
Breast	699	9
Abalone	4177	10
Sonar	208	60
Bupa	345	6
Iono	351	34
Pima	768	8
Kin_8fh	8192	8
Kin_8nh	8192	8
Kin_8fm	8192	8
Kin_8nm	8192	8
Kin_32fh	8192	32
Kin_32nh	8192	32
Kin_32fm	8192	32
Kin_32nm	8192	32

Table 1. A summary of the characteristics of the datasets considered.

9.2 Methods

This section presents the set of learning methods which are evaluated in the experimental session.

9.2.1 The Lazy Learning method

This method, described in Section 5, adopts the recursive identification and the local PRESS validation algorithm. We compare alternative strategies for model selection, and alternative types of local models:

Lb1: Local bandwidth selection for linear local models. The number of neighbors is selected on a query-by-query basis and the prediction returned is the one of the best linear model according to the mean square error criterion MSE_{loo} (winner-takes-all).

Lb0: Local bandwidth selection for constant local models. The number of neighbors is selected on a query-by-query basis and the prediction returned is the one of the best constant model according to the mean square error criterion MSE_{loo} (winner-takes-all).

LbC: Local combination of estimators. This is an example of the method described in Section 8.2. For each query it combines the outputs of the best 2 linear local models and the best 2 constant models.

The number of neighbors for constant models is allowed to range between $k_m^c = 3$ and $k_M^c = 10p$, while for linear models between $k_m^l = p$ and $k_M^l = 10p$ where $p = n + 1$ and n is the number of inputs.

The distance function is a weighted Euclidean distance (see Eq. (9)). The metric is global and weighted by the relative linear influence (*relevance*) of the regressors (Friedman, 1994). This means that the metric M in (9) is a diagonal matrix with

$$M_{jj} = \sqrt{\frac{\hat{\beta}_j^2}{\sum_{j=1}^{n} \hat{\beta}_j^2}} \tag{29}$$

where $\hat{\beta}_j$ is the j^{th} term of the least-squares vector estimated on the whole training set.

9.2.2 Local modeling methods

They are two local modeling approaches where the number of neighbors is selected not on a query-by-query basis but through a procedure of cross-validation on the whole training set (see the global tuning approach described in Section 3.3.5). We consider two approximators:

Gb1: Local modeling technique with linear local models and global bandwidth selection. The number of neighbors k is fixed for all

the queries and is obtained by minimizing the assessment error in a 20-fold cross-validation on the training set.

Gb0: Local modeling technique with constant local models and global bandwidth selection. As in **Gb1**, the value of k is optimized globally and kept constant for all the queries.

The bounds on the number of neighbors, the distance function and the metric are the same we defined in the previous section.

9.2.3 The Regression Tree method: Cubist

Cubist[5] is a commercial software, distributed by Prof. Quinlan, which implements the Regression Tree architecture.

It is a rule-based tool for generating piecewise-linear models on the basis of input/output data. The software has a number of different options and features. The results we report are obtained by combining the regression tree with a nearest-neighbor model (Quinlan, 1993).

9.2.4 Feed Forward Neural Networks

Feed Forward Neural Networks (**FNN**) are the most common example of neural networks for supervised learning (Bishop, 1994).

We choose a two-layer architecture with a first sigmoid layer and a second linear layer, trained by the Levenberg-Marquardt algorithm. As stated in the introduction, a fair comparison with Feed Forward Neural Networks should require a state-of-the-art neural selection procedure. In order to avoid possible criticism on this subject, we decide to perform no structural identification but to compute the predictions for several different structures and to return the best a posteriori result on the test set. This allow us to remain independent of the large amount of literature on neural structure selection and to return an optimistic result for the neural approximator. In particular we choose as structural parameter the number of neurons in the first layer and we made it vary over a range between 2 and 12 neurons.

[5] `http://www.rulequest.com`

The software used to perform the experiments is the set of routines for Feed-Forward Neural Networks provided in the Matlab© Neural Network toolbox.

Note that in Table 2 and 3 we report only the results obtained by the best a posteriori neural structures which are not necessarily the same among the different datasets.

9.2.5 Mixtures of Experts

We consider a one-level Mixture of Experts (**ME**) architecture trained according to the parametric identification method described in (Xu *et al.*, 1995). The experiments are performed using our own Matlab© implementation of the ME learning method.

As in the case of Feed Forward Neural Networks, we perform no structural identification but we generate several different structures and we return the best a posteriori result on the test set. The structural parameter is chosen to be the number of experts at the bottom level, ranging over $[3, 12]$ for the first 19 datasets and over $[3, 8]$ for the last four.

9.2.6 Local Model Networks

We consider two local model networks (Johansen & Foss, 1993) with different membership functions and different initializations:

LMNf It is a local model network with triangular memberships which is initialized by the fuzzy hyperplane clustering procedure (Babuska, 1996).

LMNk It is a local model network with Gaussian memberships which is initialized by the k-means clustering procedure (Moody & Darken, 1989).

For both the models, the number of rules is the structural parameter which is allowed to range over the interval $[2, 12]$ for the first 19 datasets and over $[3, 8]$ for the last four. We choose the *a posteriori best-case* strategy for choosing the number of local models.

The experiments are performed using the Matlab© toolbox for Neuro-Fuzzy data analysis[6] developed at Iridia (Bontempi & Birattari, 1999).

9.3 The experimental methodology

Each approach is tested on each dataset using the same 10-fold cross-validation strategy.

Each dataset is first normalized through a Z-score scaling (Masters, 1995) and then divided randomly into 10 groups of nearly equal size. In turn, each of these groups is used as a testing set while the remaining ones provide the examples. Thus all the methods perform a prediction on the same unseen cases, using for each of them the same set of examples.

This guarantees the same experimental conditions for all the approaches.

9.4 Results

We report the results using two synthetic indices of performances and an exhaustive set of paired comparisons between the learning methods.

In Table 2 we present, for each learning method, the absolute prediction error averaged over the 10 cross-validation groups.

The second index of performance we report is the *relative error*, defined as the mean square prediction error on unseen cases, normalized by the variance of the test set. The relative errors are presented in Table 3 and show a similar picture to Table 2, although the mean square errors considered here penalize larger absolute errors.

Since the methods are tested on the same examples under the same conditions, we use the sensitive one-tailed paired test of significance to perform an exhaustive paired comparison of all the methods for all the benchmarks. In what follows, by "significantly better" we mean better at least at a 5% significance level. The whole amount of paired comparisons is reported in (Bontempi, 1999).

[6] http://iridia.ulb.ac.be/~gbonte/software/Local/FIS.html

Dataset	Lb1	Lb0	LbC	Gb1	Gb0	Cub	FNN	ME	LMNf	LMNk
Housing	2.10	2.56	2.06	2.03	2.47	2.14	2.44	2.61	3.01	3.29
Cpu	26.41	34.86	25.97	26.82	29.17	28.33	32.19	28.98	42.95	44.83
Prices	1511.5	1577.4	1525.4	1389.3	1510.1	1377.7	2209	2236	2173.1	2859.1
Mpg	1.87	1.95	1.85	1.91	1.98	1.91	2.06	2.05	2.09	2.57
Servo	0.31	0.29	0.31	0.33	0.33	0.36	0.41	0.68	1.11	1.07
Ozone	3.39	3.42	3.16	3.26	3.18	3.06	3.34	3.17	3.18	3.40
Bodyfat	93e-4	95e-4	87e-4	87e-4	88e-4	87e-4	86e-4	87e-4	85e-4	96e-4
Pool	0.65	0.75	0.63	0.60	0.79	0.63	0.59	0.63	0.60	0.68
Energy	7.83	15.68	8.04	7.91	16.14	10.26	11.38	20.14	18.64	17.88
Breast	0.049	0.042	0.042	0.055	0.048	0.071	0.058	0.048	0.085	0.091
Abalone	1.58	1.55	1.49	1.55	1.49	1.51	1.46	1.54	1.52	1.55
Sonar	0.38	0.18	0.17	0.37	0.22	0.20	0.32	0.68	0.36	4.87
Bupa	0.39	0.38	0.37	0.38	0.43	0.37	0.38	0.40	0.46	0.41
Iono	0.21	0.12	0.11	0.21	0.11	0.14	0.18	0.17	0.42	1.58
Pima	0.31	0.28	0.28	0.31	0.31	0.27	0.32	0.31	0.33	0.33
Kin_8fh	36e-3	39e-3	35e-3	34e-3	36e-3	35e-3	33e-3	35e-3	34.2e-3	34.1e-3
Kin_8nh	0.15	0.15	0.14	0.14	0.15	0.14	0.13	0.15	0.17	0.16
Kin_8fm	11e-3	17e-3	11e-3	11e-3	15e-3	12e-3	9.6e-3	13e-3	14e-3	12e-3
Kin_8nm	86e-3	105e-3	89e-3	87e-3	100e-3	91e-3	70e-3	118e-3	136e-3	129e-3
Kin_32fh	0.22	0.25	0.21	0.23	0.23	0.21	0.20	0.21	0.36	0.36
Kin_32nh	0.38	0.39	0.36	0.39	0.36	0.36	0.32	0.33	0.35	0.37
Kin_32fm	97e-3	152e-3	95e-3	99e-3	0.14	94e-3	79e-3	94e-3	0.32	0.32
Kin_32nm	0.34	0.35	0.32	0.35	0.33	0.32	0.27	0.29	0.33	0.34

Table 2. Mean absolute error obtained on unseen cases.

A concise summary of the comparisons is reported in Table 4, which counts the number of times that a model was signifi cantly worse than another during the experimental session. It is evident that the less is the fi gure in the table, the better is the global performance on the totality of datasets.

9.5 Discussion

The fi rst consideration about the experimental session can be done by taking a look at the summary in Table 4. In this particular ranking, where the lowest the best, the LL approach **LbC** is the winner, followed by **Cubist** and by **FNN**. Note that this summary has for only purpose to give a qualitative indication of the performance of an approach on the whole amount of benchmarks and that further considerations should be done by looking in detail at the single tables.

Considering the performances on the single datasets, we remark a differ-

Dataset	Lb1	Lb0	LbC	Gb1	Gb0	Cub	FNN	ME	LMNf	LMNk
Housing	11.72	20.29	11.60	10.68	17.02	14.72	13.78	19.95	26.49	34.18
Cpu	9.98	35.65	9.78	9.82	14.12	12.61	13.68	30.98	42.61	47.54
Prices	14.06	21.78	16.62	11.88	20.08	11.83	40.31	41.40	39.66	68.86
Mpg	12.64	13.02	11.90	12.69	13.23	12.64	13.26	14.12	14.67	23.05
Servo	16.92	20.67	16.73	17.73	24.85	19.72	27.17	55.42	97.46	90.27
Ozone	31.96	33.44	27.62	29.00	27.55	25.11	29.80	27.95	27.43	33.00
Bodyfat	38.92	38.22	30.84	35.13	33.49	33.57	31.20	33.68	31.80	43.24
Pool	12.29	16.14	12.07	10.31	16.33	11.54	9.85	10.90	10.19	12.97
Energy	0.32	0.71	0.33	0.32	0.75	0.54	0.41	1.17	0.83	1.07
Breast	10.94	14.53	12.83	10.59	11.89	20.56	14.93	14.49	13.39	15.17
Abalone	48.06	48.97	43.77	45.07	44.39	43.90	41.62	44.23	43.65	45.03
Sonar	92.87	46.29	42.80	88.79	52.21	58.36	98.69	>100	83.35	>100
Bupa	95.72	98.14	94.00	80.83	85.24	104	88.66	95.69	372	89.04
Iono	50.31	43.31	37.95	47.79	36.66	44.74	50.40	45.92	83.06	>100
Pima	83.56	82.61	81.07	70.37	70.10	82.96	98.21	86.56	74.08	73.22
Kin_8fh	29.00	33.22	26.71	26.08	28.79	26.48	24.55	26.31	25.90	25.81
Kin_8nh	47.11	52.31	43.42	43.76	48.35	46.00	36.24	51.37	60.42	53.36
Kin_8fm	3.45	8.82	3.32	3.28	6.99	3.91	2.63	5.05	5.99	4.02
Kin_8nm	17.29	26.87	17.87	17.69	25.52	20.71	11.90	34.80	44.99	39.59
Kin_32fh	37.99	47.13	33.91	40.63	40.36	31.89	30.09	32.56	99.41	99.34
Kin_32nh	96.35	97.69	83.12	100.15	83.47	84.29	68.37	73.05	81.64	90.02
Kin_32fm	9.50	23.18	9.00	9.90	19.76	8.94	6.19	8.91	99.11	98.70
Kin_32nm	88.95	90.73	76.46	91.49	78.39	74.92	56.51	63.22	78.91	86.99

Table 3. Relative error (%) obtained on unseen cases.

Method	Number of times the method was significantly worse than another
Lb1	74
Lb0	96
LbC	23
Gb1	58
Gb0	81
Cub	40
FNN	53
ME	80
LMNf	132
LMNk	145

Table 4. A comparative summary of the performances of the used methods.

ent behavior between real datasets (the first 15) and synthetically generated datasets (the last 8). While the **LbC** approach and **Cubist** do better on the first class of data, the **FNN** approach outperforms the other ones on the second class.

A possible explanation is the following: methods like Lazy Learning and Regression Trees are tailored to configurations with non uniform noise and with no analytical model underlying the data distribution. On the contrary, data obtained by simulating a mathematical model, as the kinematic model of a manipulator in the Kin family, and distorted with artificial and uniformly distributed noise are the ideal setting for a global model, like the **FNN**. A global model can easily deal with an uniform disturbance since the totality of data gives information about the noise. On the contrary, divide-and-conquer approaches exploit only a portion of data with the consequence of a less reliable tradeoff between signal detection and noise reduction.

As far as the LL approach is concerned, it is important to compare the performance of the local combination with respect to the winner-takes-all approach in the model selection step (Section 8). According to the comparative tables, the method **LbC** results 14 times significantly better than the *winner-takes-all* linear **Lb1** and 2 times significantly worse. Also, it results 18 times significantly better than the *winner-takes-all* constant **Lb0** and 0 times worse.

As far as local modeling is concerned, an interesting comparison concerns the performance of the **LbC** method based on a query-by-query bandwidth selection with respect to local methods based on a global tuning of the number of neighbors (Section 9.2.2). The method **LbC** is 10 times significantly better than **Gb1** and only 2 times significantly worse. Also, **LbC** is 16 times significantly better than **Gb0** and is never significantly worse.

As far as the comparison of the best LL method with **Cubist** is concerned, the **LbC** technique performs significantly better than the regression tree **Cubist** on 8 datasets and significantly worse on 2 datasets.

Similar considerations hold for the comparison of **LbC** with the **ME** approach (12 times better and 4 worse) and the Local Model Network approaches.

Some words should be spent also about the bad performance of the Local Model Network approaches. We suspect that the LMN approach suffered of the high dimensionality of the regression problems taken into consid-

eration, which made the final performance of the algorithm excessively dependent on the initialization step.

It is our belief that no experimental result can produce a definitive sentence on the superiority of a learning method over another one. Furthermore, new techniques, like the Lazy Learning method, should be tested in many real tasks before being claimed as a consolidate method. However, we think that a rigorous experimental setup, as that presented in this chapter, can give a qualitative indication whether an algorithm is somewhat a promising tool for model design. In these terms, the examples given throughout this section highlight that the Lazy Learning approach, and in particular its version **LbC** based on the combination of estimators, can be an effective technique for supervised learning, featuring a generalization accuracy comparable and sometimes significantly better than those of state-of-the-art algorithms.

Examples of applications of the LL algorithm to real problems are given in the following section.

10 Lazy Learners at work

The Lazy Learning method is currently applied with success by the Iridia laboratory to a number of academic and industrial problems. Here we list some of them:

Financial prediction of stock markets. This is a joint project of Iridia and the research center of Masterfood, which adopts the Lazy Learning techniques for the prediction of some market indices.

Prediction of chaotic time series. The Lazy Learning method for iterated time series prediction (Bontempi *et al.*, 1999c) ranked second among 17 participants to the International Competition on Time Series organized by the *International Workshop on Advanced Black-box techniques for nonlinear modeling* held in Leuven, Belgium (Suykens & Vandewalle, 1998).

Data analysis. The Lazy Learning method took part to the *Third International Competition of Data Analysis by Intelligent Techniques*

organized by the European Network *Erudit*, where it was awarded as a runner-up among 21 participants (Bontempi *et al.*, 1999a).

Non linear control and identification task. The Lazy Learning method for nonlinear control (Bontempi *et al.*, 1999b; Bontempi *et al.*, 1999d) is currently tested on a set of benchmarks proposed in the LTR project FAMIMO (Fuzzy Algorithm for Multi Input Multi Output processes).

Modeling of industrial processes. The Lazy Learning technique is employed to model the rolling steel mill process of the FaFer Usinor steel company in Charleroi, Belgium. It is also the subject of an active collaboration of Iridia with the Honeywell Technology Center in Minneapolis, USA.

Electrical power load forecasting. A joint project of Iridia and Tractebel Belgium is adopting local techniques for the forecasting of the electrical power load.

Prediction of economic variables. A joint project of Iridia and Dieteren, the first Belgian car dealer, is studying the adoption of Lazy Learning techniques to predict the annual amount of sales on the basis of historical data.

11 The importance of being Lazy

Goal of this chapter was to introduce local modeling, and in particular its Lazy Learning version, as an appealing alternative to existing techniques. Theoretical and algorithmic considerations, joined with a number of experiments, aimed to raise Lazy Learning techniques not to a privileged status but rather to a parity condition with respect to state-of-the-art methods.

This final section summarizes the main advantages that derive from adopting a Lazy Learning technique in a modeling task.

Few assumptions. Lazy Learning assumes no a priori knowledge on the process underlying the data. The only available information is represented by a finite set of input/output observations.

Query-by-query design. Lazy Learning solves the problem of modeling an input/output mapping in a local perspective, making no assumption on the existence of a global function describing the data and making no assumption on the properties of the noise. This feature is particularly relevant in real datasets where problems of missing features, non stationarity and measurement errors make appealing a data-driven and assumption-free approach like Lazy Learning.

Lazy Learning gives linearity a chance. A lot of powerful techniques and well-founded theoretical results in linear statistical theory, linear regression and linear control analysis are reused effectively by Lazy Learning in a nonlinear setting. This property makes easier the implementation and the theoretical analysis of a modeling and/or control method based on Lazy Learning.

Fast design. The learning procedure of Lazy Learning is based on a fast method to generate a set of candidate models based on a recursive technique, a fast method to identify the parameters of a local model based on linear least-squares and a fast technique to validate the local models - the PRESS - which is obtained at no additional cost as a by product of the parametric identification.

Statistically informative. By adopting a local modeling technique and a powerful validation technique, the Lazy Learning returns at the same cost of the prediction also a statistical description of the uncertainty affecting the prediction itself. This a relevant property which is used in the method to combine different local predictors or that can be effectively employed to combine the Lazy Learning prediction with other kind of approximators.

On-line learning. The Lazy Learning method can easily deal with online learning tasks where the number of training samples increases with time. In this case, the adaptiveness of the method is obtained by simply adding the points to the stored dataset.

Non stationary tasks. The Lazy Learning method can deal with time-varying configurations where the stochastic process underlying the data is non stationary. In this case, it is sufficient to mean the notion of neighborhood not in a spatial way but in a spatio-temporal

sense. For each query point, the neighbors will not be the samples which have simply similar inputs but the ones that both have similar inputs and have been collected recently in time. Therefore, the time variable becomes a further precious feature to consider for accurate prediction.

Successful applications. Lazy Learning has been successful in addressing a number of practical problems, from regression modeling to time series prediction and nonlinear control.

References

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

D. W. Aha. 1990. *A Study of Instance-Based Algorithms for Supervised Learning Tasks: Mathematical, Empirical and Psychological Observations*. Ph.D. thesis, University of California, Irvine, Department of Information and Computer Science.

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

D. M. Allen. 1974. The relationship between variable and data augmentation and a method of prediction. *Technometrics*, **16**, 125–127.

C. G. Atkeson. 1989. Using local models to control movement. *Pages 79–86 of:* D. Touretzky (ed), *Advances in neural information processing systems, 1*. San Mateo, CA: Morgan Kaufmann.

C. G. Atkeson, A. W. Moore, & S. Schaal. 1997. Locally weighted learning. *Artificial Intelligence Review*, **11**(1–5), 11–73.

R. Babuska. 1996. *Fuzzy Modeling and Identification*. Ph.D. thesis, Technische Universiteit Delft.

G. J. Bierman. 1977. *Factorization Methods for Discrete Sequential Estimation*. New York, NY: Academic Press.

M. Birattari, & G. Bontempi. 1999. *Lazy Learning Vs. Speedy Gonzales : A fast algorithm for recursive identification and recursive validation of local constant models*. Tech. rept. TR/IRIDIA/99-6. IRIDIA-ULB, Brussels, Belgium.

M. Birattari, G. Bontempi, & H. Bersini. 1999. Lazy learning meets the recursive least-squares algorithm. *Pages 375–381 of:* M. S. Kearns, S. A. Solla, & D. A. Cohn (eds), *Advances in Neural Information Processing Systems 11*. Cambridge: MIT Press.

C. M. Bishop. 1994. *Neural Networks for Statistical Pattern Recognition*. Oxford, UK: Oxford University Press.

G. Bontempi. 1999. *Local Learning Techniques for Modeling, Prediction and Control*. Ph.D. thesis, IRIDIA- Université Libre de Bruxelles.

G. Bontempi, & M. Birattari. 1999. *Toolbox for Neuro-Fuzzy Identification and Data Analysis, For use with Matlab*. Tech. rept. 99-9. IRIDIA-ULB, Bruxelles, Belgium.

G. Bontempi, M. Birattari, & H. Bersini. 1998. Recursive lazy learning for modeling and control. *Pages 292–303 of: Machine Learning: ECML-98 (10th European Conference on Machine Learning)*.

G. Bontempi, M. Birattari, & H. Bersini. 1999a. Lazy Learners at work: the Lazy Learning Toolbox. *In: Proceeding of the 7th European Congress on Inteligent Techniques and Soft Computing EUFIT '99*.

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

G. Bontempi, M. Birattari, & H. Bersini. 1999c. Local learning for iterated time-series prediction. *Pages 32–38 of:* I. Bratko, & S. Dzeroski (eds), *Machine Learning: Proceedings of the Sixteenth International Conference*. San Francisco, CA: Morgan Kaufmann Publishers.

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

G. Bontempi, M. Birattari, & H. Bersini. 1999e. A model selection approach for local learning. *Artificial Intelligence Communications*. in press.

C. De Boor. 1978. *A Practical guide to splines*. New York: Springer.

L. Breiman. 1996. Stacked regressions. *Machine Learning*, **24**(1), 49–64.

L. Breiman, J. H. Friedman, R. A. Olshen, & C. J. Stone. 1984. *Classification and Regression Trees*. Belmont, CA: Wadsworth International Group.

W. S. Cleveland. 1979. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, **74**, 829–836.

W. S. Cleveland, & S. J. Devlin. 1988. Locally weighted regression: an approach to regression analysis by local fi tting. *Journal of American Statistical Association*, **83**, 596–610.

W. S. Cleveland, & C. Loader. 1995. Smoothing by Local Regression: Principles and Methods. *Computational Statistics*, **11**.

T. Cover, & P. Hart. 1967. Nearest neighbor pattern classifi cation. *Proc. IEEE Trans. Inform. Theory*, 21–27.

G. Cybenko. 1996. Just-in-Time Learning and Estimation. *Pages 423–434 of:* S. Bittanti, & G. Picci (eds), *Identification, Adaptation, Learning. The Science of Learning Models from data*. NATO ASI Series. Springer.

N. R. Draper, & H. Smith. 1981. *Applied Regression Analysis*. New York: John Wiley and Sons.

J. Fan, & I. Gijbels. 1992. Variable bandwidth and local linear regression smoothers. *The Annals of Statistics*, **20**(4), 2008–2036.

J. Fan, & I. Gijbels. 1995. Adaptive order polynomial fi tting: bandwidth robustifi cation and bias reduction. *J. Comp. Graph. Statist.*, **4**, 213–227.

J. Fan, & I. Gijbels. 1996. *Local Polynomial Modelling and Its Applications*. Chapman and Hall.

J. D. Farmer, & J. J. Sidorowich. 1987. Predicting chaotic time series. *Physical Review Letters*, **8**(59), 845–848.

U. Fayyad, G. Piatetsky-Shapiro, & P. Smyth. 1996. The KDD Process for Extracting Useful Knowledge from Volumes of Data. *Communications of the ACM*, **39**(11), 27–34.

J. H. Friedman. 1994. *Flexible metric nearest neighbor classification*. Tech. rept. Stanford University.

S. Geman, E. Bienenstock, & R. Doursat. 1992. Neural networks and the bias/variance dilemma. *Neural Computation*, **4**(1), 1–58.

G. C. Goodwin, & K. S. Sin. 1984. *Adaptive Filtering Prediction and Control*. Prentice-Hall.

W. Hardle, & J. S. Marron. 1995. Fast and simple scatterplot smoothing. *Comp. Statist. Data Anal.*, **20**, 1–17.

T. Hastie, & C. Loader. 1993. Local regression: automatic kernel carpentry. *Statistical Science*, **8**, 120–143.

T. Hastie, & R. Tibshirani. 1990. *Generalized Additive Models*. London, UK: Chapman and Hall.

T. Hastie, & R. Tibshirani. 1996. Discriminant Adaptive Nearest Neighbor Classifi cation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **18**(6), 607–615.

T. A. Johansen, & B. A. Foss. 1993. Constructing NARMAX models using ARMAX models. *International Journal of Control*, **58**, 1125–1153.

M. C. Jones, J. S. Marron, & S. J. Sheather. 1995. A brief survey of bandwidth selection for density estimation. *Journal of American Statistical Association*, **90**.

M. J. Jordan, & R. A. Jacobs. 1994. Hierarchical Mixtures of Experts and the EM Algorithm. *Neural Computation*, **6**, 181–214.

V. Y. Katkovnik. 1979. Linear and nonlinear methods of nonparametric regression analysis. *Soviet Automatic Control*, **5**, 25–34.

J. Kolodner. 1993. *Case-Based Reasoning*. Morgan Kaufmann.

C. R. Loader. 1987. *Old Faithful Erupts: Bandwidth Selection Reviewed*. Tech. rept. Bell-Labs.

C. Mallows. 1974. Discussion of a paper of Beaton and Tukey. *Technometrics*, **16**, 187–188.

O. Maron, & A. Moore. 1997. The Racing Algorithm: Model Selection for Lazy Learners. *Artificial Intelligence Review*, **11**(1–5), 193–225.

T. Masters. 1995. *Practical Neural Network Recipes in C++*. New York, NY: Academic Press.

C. J. Merz, & P. M. Murphy. 1998. UCI Repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

J. Moody, & C. J. Darken. 1989. Fast learning in networks of locally-tuned processing units. *Neural Computation*, **1**(2), 281–294.

A. Moore. 1991. Fast, robust adaptive control by learning only forward models. *In:* J. E. Moody, S. J. Hanson, & R. P. Lippman (eds), *Advances in neural information processing systems, NIPS 4*. San Mateo, CA: Morgan Kaufmann.

A. W. Moore, D. J. Hill, & M. P. Johnson. 1992. An empirical investigation of brute force to choose features, smoothers and function approximators. *In:* S. Janson, S. Judd, & T. Petsche (eds), *Computational Learning Theory and Natural Learning Systems*, vol. 3. Cambridge, MA: MIT Press.

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

R. H. Myers. 1994. *Classical and Modern Regression with Applications*. second edn. Boston, MA: PWS-KENT Publishing Company.

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

B. U. Park, & J. S. Marron. 1990. Comparison of data-driven bandwidth selectors. *Journal of American Statistical Association*, **85**, 66–72.

M. P. Perrone, & L. N. Cooper. 1993. When networks disagree: Ensemble methods for hybrid neural networks. *Pages 126–142 of:* R. J. Mammone (ed), *Artificial Neural Networks for Speech and Vision*. Chapman and Hall.

M. B. Priestley, & M. T. Chao. 1972. Non-parametric Function Fitting. *Journal of Royal Statistical Society, Series B*, **34**, 385–392.

J. R. Quinlan. 1993. Combining instance-based and model-based learning. *Pages 236–243 of: Machine Learning. Proceedings of the Tenth International Conference*. Morgan Kaufmann.

J. Rice. 1984. Bandwidth choice for nonparametric regression. *The Annals of Statistics*, **12**, 1215–1230.

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

D. Ruppert, & M. P. Wand. 1994. Multivariate locally weighted least squares regression. *The Annals of Statistics*, **22**(3), 1346–1370.

D. Ruppert, S. J. Sheather, & M. P. Wand. 1995. An effective bandwidth selector for local least squares regression. *Journal of American Statistical Association*, **90**, 1257–1270.

D. W. Scott. 1992. *Multivariate density estimation*. New York: Wiley.

G. A. F. Seber, & C. J. Wild. 1989. *Nonlinear regression*. New York: Wiley.

C. Stanfi ll, & D. Waltz. 1987. Toward memory-based reasoning. *Communications of the ACM*, **29**(12), 1213–1228.

C. Stone. 1977. Consistent nonparametric regression. *The Annals of Statistics*, **5**, 595–645.

M. Stone. 1974. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B*, **36**(1), 111–147.

J. A. K. Suykens, & J. Vandewalle (eds). 1998. *Nonlinear Modeling: Advanced Black-Box Techniques*. Kluwer Academic Publishers. Chap. The K. U. Leuven Time Series Prediction Competition, pages 241–251.

T. Takagi, & M. Sugeno. 1985. Fuzzy identifi cation of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, **15**(1), 116–132.

V. N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. New York, NY: Springer.

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

D. Wolpert. 1992. Stacked Generalization. *Neural Networks*, **5**, 241–259.

M. Woodrofe. 1970. On choosing a delta-sequence. *Ann. Math. Statist.*, **41**, 1665–1671.

L. Xu, M. I. Jordan, & G. E. Hinton. 1995. An Alternative Model for Mxtures of Experts. *Pages 633–640 of:* G. Tesauro, D. Touretzky, & T. Leen (eds), *Advances in Neural Information Processing Systems*, vol. 7. The MIT Press.