Neural Computation, vol. 14, pp. 1481-1506

LOCAL OVERFITTING CONTROL VIA LEVERAGES

Gaétan MONARI*,**, Gérard DREYFUS*

*École Supérieure de Physique et de Chimie Industrielles de la Ville de Paris

Laboratoire d'Électronique
10, rue Vauquelin - F 75005 PARIS - FRANCE

**USINOR

DSI/DISA SOLLAC FOS bat. LB1
F 13776 FOS-sur-Mer Cedex - FRANCE

ABSTRACT

We present a novel approach to dealing with overfitting in black-box models. It is based on the leverages of the samples, i.e. on the influence that each observation has on the parameters of the model. Since overfitting is the consequence of the model specializing on specific data points during training, we present a selection method for nonlinear models, which is based on the estimation of leverages and confidence intervals. It allows both the selection among various models of equivalent complexities corresponding to different minima of the cost function (e.g. neural nets with the same number of hidden units), and the selection among models having different complexities (e.g. neural nets with different numbers of hidden units). A complete model selection methodology is derived.

1. INTRODUCTION

The traditional view of overfitting refers mostly to the bias / variance tradeoff, introduced in (Geman & al., 1992): a family of parameterized functions with too few parameters, with respect to the complexity of a problem, is said to have too large a bias, because it cannot fit the deterministic model underlying the data. Conversely, when the model is over-parameterized, the dependence of the resulting functions on the particular training set is too large, and so is the variance of the corresponding family of parameterized functions. Therefore, overfitting is usually detected by the fact that the modeling error on a test set is much larger than the modeling error on the training data.

In practice, there are two major ways of preventing overfitting:

- *a priori*, by limiting the variance of the considered family of parameterized functions. These regularization methods include weight decay (see (MacKay, 1992) for a bayesian

approach to weight decay), and similar cost function penalizations, as well as early stopping (Sjöberg & al., 1992). None of them exempts the model designer from an additional estimation of the generalization performance of the selected model;

- *a posteriori*, by estimating the generalization performance on data that have not been used to fit the model. This approach relies on data re-sampling and has given rise to the cross-validation methods (Stone, 1974), including leave-one-out. Statistical tests can be further used, after selecting candidate models, to test whether differences in the estimated performances are significant (see for instance (Anders & al., 1999)).

The limitations of the above methods are well known: weight decay and similar methods require the estimation of meta-parameters, while resampling methods tend to be computationally intensive.

In the present paper, we consider overfitting as a local phenomenon that occurs when the influence of one (or more) particular example(s) on the model becomes too large, because of the excessive flexibility of the model. Therefore, we suggest a new approach to model selection that takes overfitting into account on the basis of the *leverages* of the available samples, i.e. on the influence of each sample on the parameters of the model.

In the next section, we recall briefly the mathematical framework of this approach. In sections 3 and 4, we show that this perspective on overfitting suggests a model selection method, which takes into account both an estimation of the generalization error and the distribution of the influences of the training examples: section 3 is devoted to model selection within a given family of parameterized functions, while section 4 focuses on the choice of the appropriate family among several candidate families. Section 5 illustrates the method with several examples.

2. MATHEMATICAL FRAMEWORK

The present paper discusses static single-output processes with a non-random n-input vector $\boldsymbol{x} = [x_1, ..., x_n]^T$ and an output y_p which is considered as a measurement of a random variable Y_p. We assume that an appropriate model can be written under the form:

$$Y_p = r(\boldsymbol{x}) + W \tag{1}$$

where W is a zero-mean random variable and $r(\boldsymbol{x})$ is the unknown regression function. A data set of N input-output pairs $\{\boldsymbol{x}^k, y_p{}^k\}_{k=1, ..., N}$ is assumed to be available for estimating the parameters of the model $f(\boldsymbol{x}, \boldsymbol{\theta})$. In the following, all vectors are column vectors, denoted by bold type letters, e.g. the n-vectors $\boldsymbol{x}$ and $\{\boldsymbol{x}^k\}$.

In (Monari & al. 2000), the effect of withdrawing an example from the training set was investigated. Assume that a model with parameters $\boldsymbol{\theta}_{LS}$ has been found, by minimizing the least-squares cost function computed on the training set. Under the sole assumption that the

removal of an example from the training set has a small effect on $\boldsymbol{\theta}_{LS}$ (in contrast to standard leave-one-out, no stability assumption is required in the present approach, as discussed in section 3.2 and in Appendix 2), a second-order Taylor development of the model output with respect to the parameters was derived. It was shown that this generates an approximate model that is locally linear with respect to the parameters, and whose variables are the components of the gradient of the model output with respect to the parameters. Introducing the jacobian matrix $Z = [\boldsymbol{z}^1, ..., \boldsymbol{z}^N]^T$, where $\boldsymbol{z}^i = \left.\frac{\partial f\left(x^i, \boldsymbol{\theta}\right)}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{LS}}$, the *solution subspace* can be defined (in analogy with linear models) as the subspace determined by the columns of matrix Z (assuming that the latter has full rank). Then the quantity

$$h_{ii} = \boldsymbol{z}^{iT} (Z^T Z)^{-1} \boldsymbol{z}^i \tag{2}$$

is the i^{th} component of the projection, on the solution subspace, of the unit vector along axis i.

The quantities $\{h_{ii}\}_{i=1,...,N}$, termed *leverages* of the training examples, satisfy the following relations:

$$\forall i \in \left[1, ..., N\right] \quad 0 \le h_{ii} \le 1 \tag{3}$$

$$\sum_{i=1}^{N} h_{ii} = q \tag{4}$$

where q is the number of adjustable parameters of the model (e.g. the number of weights of a neural network, the number of monomials of a polynomial approximation, etc.). Equations (3) and (4) stem from the fact that the quantities $\{h_{ii}\}_{i=1,...,N}$ are the diagonal terms of the orthogonal projection matrix $Z (Z^T Z)^{-1} Z^T$.

If axis i is orthogonal to the solution subspace, all columns have their i^{th} component equal to zero; hence $\boldsymbol{z}^i = 0$ and $h_{ii} = 0$. Since the output of the model is the orthogonal projection of the process output onto the solution subspace, example i has essentially no influence on the model (it has none in the case of a linear model). If axis i lies within the solution subspace, then $h_{ii} = 1$ and example i is learnt almost perfectly (it is learnt perfectly in the case of a linear model). Thus, h_{ii} is an indication of the influence of example i on the model: the closer h_{ii} to 1, the larger the influence of example i on the model. This will be further shown below.

In the case where all examples have the same influence on the model, all leverages $\{h_{ii}\}_{i=1,...,N}$ equal $\frac{q}{N}$. In other words, each example uses up a fraction $1/N$ of the q available degrees of freedom (adjustable parameters) of the model. If one considers that overfitting results from an excessive influence of one (or more) examples on a model, then model selection aims at obtaining the model that has the best performances and in which the influences of all examples are roughly equal.

The influence of an example on the model should be reflected in the effect of its withdrawal from the training set: if an example has a large influence on the model ($h_{ii} \approx 1$), it should be very accurately learnt when it is present in the training set, but it should be badly predicted otherwise; conversely, if an example has no influence on the model ($h_{ii} \approx 0$), it should be predicted with equal accuracy irrespective of its presence or absence in the training set. Indeed, if we denote by R_i the residual of example i (i.e. the modeling error on example i when it is present in the training set: $R_i = y_p^{\,i} - f(\boldsymbol{x}, \boldsymbol{\theta}_{LS})$), an approximate expression of the prediction error $R_i^{(-i)}$ that would occur if this example had been removed from the training set, is given by:

$$R_i^{(-i)} \cong \frac{R_i}{1 - h_{ii}} \tag{5}$$

Details of the derivation of this relation are given in (Monari & al. 2000). This approximation is founded insofar as the 2nd order Taylor development of the output is valid, i.e. 3rd order terms are negligible.

Hence, the difference between the predictions of example i, depending on whether it belongs or not to the training set, is a function of its leverage h_{ii}[1]. This property will be taken advantage of in the selection method presented in the next sections.

If the jacobian matrix Z is not of rank q, i.e. if the manifold of Σ^N defined by the columns of Z is - at $\boldsymbol{\theta}_{LS}$ - not of full rank, the corresponding models must be discarded. For such models, the number of available parameters is locally too large in view of the number of training examples, which leads to an under-determination of these parameters. This is an extreme case of overfitting that can be detected either by computing directly the rank of Z (a difficult numerical task), or, indirectly, by checking the validity of relations (3) and (4). The latter solution supposes that we are able to compute the $\{h_{ii}\}_{i=1,\ldots,N}$, irrespective of the fact that $Z^{\mathrm{T}} Z$ is invertible or not. To address this problem, a Singular Value Decomposition (SVD, see for instance (Press & al., 1988)) of matrix Z can be performed, as shown in Appendix 1. SVD is very accurate and can always be performed, even if Z is singular. In the latter case however, the leverages are not computed accurately, hence should not be used (e.g. for computing the

---

[1] Actually, when h_{ii} approaches one, the residual R_i vanishes less rapidly than (1 - h_{ii}). This can be understood as follows: if an example has a strong influence on the model, its withdrawal from the training set causes its residual to be significantly different. Therefore, from relation (5), the quantity $R_i - \frac{R_i}{1 - h_{ii}} = \frac{h_{ii}}{1 - h_{ii}} R_i$ is not vanishingly small; hence, the estimate of the prediction error for this example $R_i^{(-i)} \cong \frac{R_i}{1 - h_{ii}}$ does not vanish either.

quantity E_p defined in the next section) because they do not comply with their basic properties (3) and (4).

Moreover, some estimates of the confidence intervals on the model output (Seber & al., 1989) rely on the assumption that Z has full rank. Under this assumption, a confidence interval on the model output for input $\boldsymbol{x}$ can be easily computed as:

$$E\left(Y_p \middle| \boldsymbol{x} \right) \in f\left(\boldsymbol{x}, \boldsymbol{\theta}_{LS} \right) \pm t_\alpha^{N-q} s \sqrt{ \boldsymbol{z}^{\mathrm{T}} \left(Z^{\mathrm{T}} Z \right)^{-1} \boldsymbol{z} } \tag{6}$$

where $\boldsymbol{z} = \left. \frac{\partial f\left(\boldsymbol{x}, \boldsymbol{\theta} \right)}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta} = \boldsymbol{\theta}_{LS}}$, t_α^{N-q} is the realization of a random variable with a Student's t-distribution with $N - q$ degrees of freedom and a level of significance $1 - \alpha$, and s is an estimate of the residual standard deviation of the model: $s = \sqrt{ \frac{1}{N-q} \sum_{i=1}^{N} R_i^2 }$. This is an additional motive for rejecting models with rank deficiency.

All theoretical material presented in sections 3 and 4 will be illustrated graphically on a small problem with one input and one output. The training data were generated using the function $y = \frac{\sin x}{x}$ and an additive gaussian noise of variance $\sigma^2 = 10^{-2}$. Fifty examples were drawn, with a uniform distribution of x within the interval [0; 15]. Throughout this paper, this data set will be referred to as the $\frac{\sin x}{x}$ problem. The application of our method to the selection of more complex, multivariable models will be demonstrated in section 5.

3. SELECTION OF A MODEL AMONG MODELS HAVING THE SAME ARCHITECTURE

For clarity, in the following, we consider that, once the number of inputs and outputs is chosen, model selection is performed in two steps:

- an architecture is chosen, i.e. a family of functions having the same complexity, within which the model is sought (e.g. the family $\boldsymbol{NN_3}$ of neural networks with five inputs, three hidden neurons and a linear output neuron); if the model is nonlinear with respect to the parameters, several trainings with different parameter initializations are performed, thereby generating various models of the same architecture (e.g. for neural nets a set of K models with 3 hidden neurons $\{NN_3^k, k = 1 \text{ to } K\}$). For this family of functions, the most appropriate model (e.g. model NN_3^{opt} in family $\boldsymbol{NN_3}$) is selected as described below in the present section;
- the previous step is performed for different architectures, i.e. for different families of parameterized functions, having different complexities (e.g. families $\boldsymbol{NN_i}$ of neural networks having the same number of inputs and outputs but different numbers i of hidden

neurons); this results in a set of models (e.g. models $\{NN_i^{opt}$, i = 0 to $I\}$). The most appropriate model among these (e.g. model NN^{opt}) is selected as described in section 4.

In this section, we first propose a criterion for the selection of a model among models having the same complexity, and we subsequently compare our approach with the traditional use of the leave-one-out technique. The choice among models of different complexities is addressed in section 4.

3.1. *A selection criterion*

A preliminary screening of the models corresponding to the different minima must be performed by checking the rank of the corresponding jacobian matrices: as stated in the previous section, models with rank deficiency are overfitted and should therefore be discarded[2]. However, the fact that, for a given architecture, the *global* minimum has rank deficiency does not mean that the considered architecture is too large: *local* minima may provide very good models[3], provided they are not rank deficient. Therefore, an additional selection criterion must be found.

To this end, we use the results presented in the previous section. In the spirit of leave-one-out cross-validation, we define the quantity E_p as:

$$E_p = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{R_i}{1-h_{ii}}\right)^2} \tag{7}$$

which is identical to the leave-one-out score except for the fact that the summation is performed on the *estimated* modeling errors given by relation (5), instead of being performed on the *actual* prediction error on each left-out example. This quantity can be compared to the traditional Training Mean Squared Error:

$$TMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} R_i^2} \tag{8}$$

E_p is larger than $TMSE$; it penalizes models that are strongly influenced by some examples. Therefore, this quantity is an appropriate choice criterion between models having the same

---

[2] Moreover, a rank deficiency of Z has an adverse effect on the efficiency of some second-order training algorithm (for instance the Levenberg-Marquardt algorithm). To overcome this, (Zhou & al., 1998) suggested a training algorithm that guarantees that the jacobian matrix is of full rank throughout training.

[3] It is well known that a suboptimal model, i.e. a model whose parameters do not minimize the cost function, may have a smaller generalization error than the global minimum. This is the basis of the "early stopping" regularization method, whereby training is stopped before a minimum is reached.

architecture but corresponding to various local minima of the cost function. Note that if all examples have the same leverage $\frac{q}{N}$, then $E_p = \frac{N}{N-q} TMSE$, hence E_p and $TMSE$ are equal in the large-sample limit for a model without overfitting.

To illustrate this, consider the case of a neural network with 4 sigmoidal hidden neurons and a linear output neuron, trained on the data set indicated in section 2. Starting from 500 different weight initializations and using the Levenberg-Marquardt algorithm leads to various minima, represented on Figure 1 as follows:

- the solutions without rank deficiency are plotted in the ($TMSE$, E_p) plane, using a logarithmic scale for the ordinates,
- the solutions with rank deficiency ($rank(Z) < q$), for which E_p cannot be computed reliably (see section 2), are plotted outside the frame of the graph.

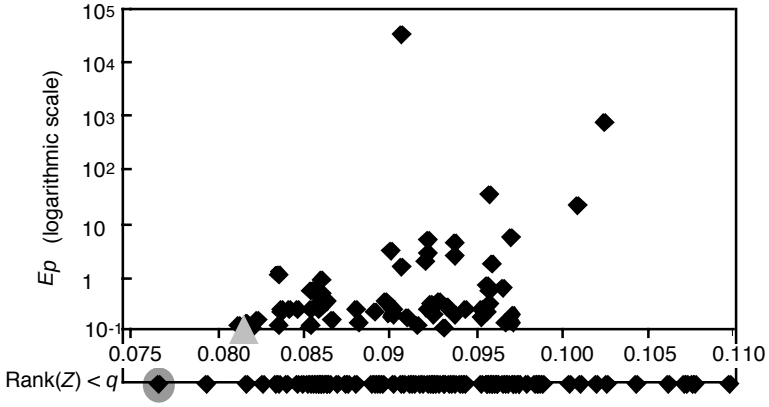

Figure 1: Distribution of the solutions obtained with networks having 4 hidden neurons. Models lying outside the frame of the graph are models with rank deficiency, for which only $TMSE$ can be computed. Note that such is the case for the model with the smallest value of the training cost function.

The analysis of this plot leads to several comments:

- even for this simple architecture, a very large number of different minima of the cost function are found,
- for this particular example, about 70 % of the solutions have rank deficiency, so that they may be discarded without further consideration. This includes the deepest minimum found - shown as a gray dot - which is likely to be a global minimum,
- for solutions without rank deficiency, the ratio of E_p to $TMSE$ varies from almost 1 to very high values, hence the choice of a logarithmic y-scale. This shows that some minima correspond to solutions that are over-influenced by a few training examples. As expected,

this overfitting is not apparent on *TMSE*, but it is on E_p. The solution with the smallest E_p is shown as a gray triangle.

The model outputs corresponding to the minima having respectively the smallest *TMSE* and the smallest E_p are shown on Figure 2.

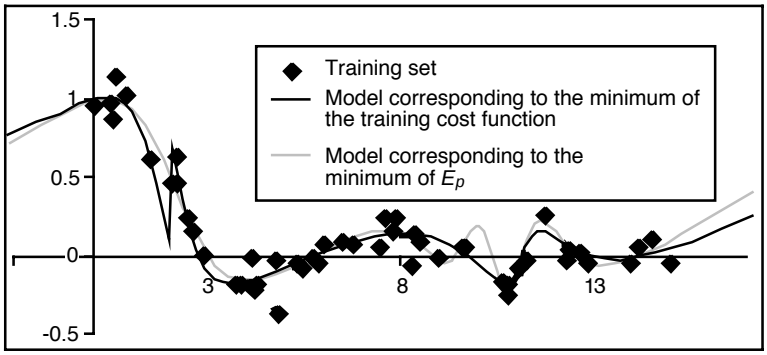

Figure 2: Outputs of models having 4 hidden neurons, selected on the basis of *TMSE* and of E_p.

Note that it is not claimed that the model with the smallest E_p is not overfitted. It is claimed that, among the various minima found with the weight initializations used, it is the model that, for the considered architecture, provides the best tradeoff between accuracy and uniform influence of the training examples. This point will be further addressed in the next section.

Starting from a linear model and increasing gradually the number of hidden neurons, one obtains the results shown on Figure 3. It appears that E_p is essentially constant between 2 and 4 neurons. Therefore, an additional step is required to perform the final model selection, i.e. to choose the appropriate number of hidden neurons; this is addressed in Section 4. In this particular case where the noise level is known, it may be inferred that models with more than 3 hidden neurons are probably overfitted since their *TMSE* is smaller than the standard deviation of the noise.

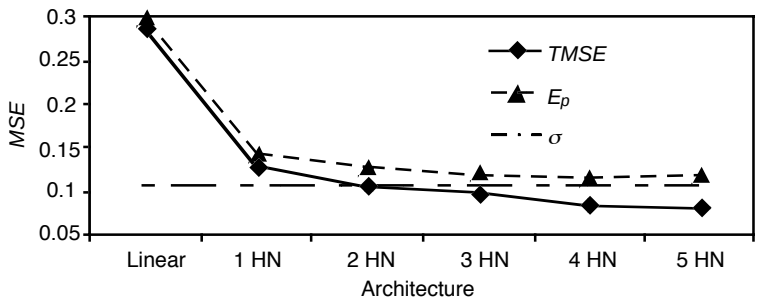

Figure 3: Evolution of the optimal E_p and of the corresponding training mean square error as a function of the number of hidden neurons. σ is the standard deviation of the noise.

The question that arises naturally is whether E_p (or the standard leave-one-out score E_t as defined below) is a good estimate of the true generalization error of the model. Under the hypothesis of "error-stability" introduced in his paper, sanity-check bounds for the leave-one-out error have been derived by (Kearns & al., 1997). Obtaining narrower bounds would require some additional stability properties of the training algorithm and / or cost function.

3.2. *Comparison with the standard leave-one-out procedure*

Since our approach relies on the first principles of the leave-one-out procedure, a comparison to the original procedure is of interest.

For models that are nonlinear with respect to their parameters, the most popular version of the leave-one-out, called "generalized leave-one-out" (Moody, 1994), consists in first finding a minimum of the cost function, with weights *W,* through training with the whole training set. The *N* subsequent trainings with one left-out example start with the set of initial weights *W*. Then, using the resulting *N* prediction errors on left-out examples, an estimation of the generalization error of the model is computed as:

$$E_t = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(R_i^{(-i)}\right)^2} \qquad (9)$$

This method assumes that the withdrawal of one example from the training set does not result in a jump from one minimum of the cost function to another one. We formalize this as follows (see appendix 2 for more details): consider a training procedure using a deterministic minimization algorithm of the cost function[4]; assume that a minimum of the cost function has been found, with a vector of parameters $\boldsymbol{\theta}_{LS}$. Assume that example *i* is withdrawn from the training set, and that a training is performed with $\boldsymbol{\theta}_{LS}$ as initial values of the parameters, leading to a new parameter vector $\boldsymbol{\theta}_{LS}^{(-i)}$; further assume that example *i* is subsequently reintroduced into the training set, and that a new training is performed with initial values of the parameters $\boldsymbol{\theta}_{LS}^{(-i)}$: in the following, the minimum of the cost function corresponding to a parameter vector $\boldsymbol{\theta}_{LS}$ will be said to be *stable for the usual leave-one-out procedure* if and only if, for each $i \in [1, \ldots, N]$, starting from $\boldsymbol{\theta}_{LS}$, the procedure described above retrieves $\boldsymbol{\theta}_{LS}$.

It has been known since (Breiman, 1996) that this is a major stability problem. In practice, it turns out that, if an overparameterized model is used, most minima of the cost function are unstable with respect to the leave-one-out procedure. Therefore, for all such minima, the computation of the leave-one-out score E_t and its comparison to E_p are meaningless.

---

[4] This excludes such algorithms as simulated annealing, probabilistic hill climbing, etc., which may overcome local minima.

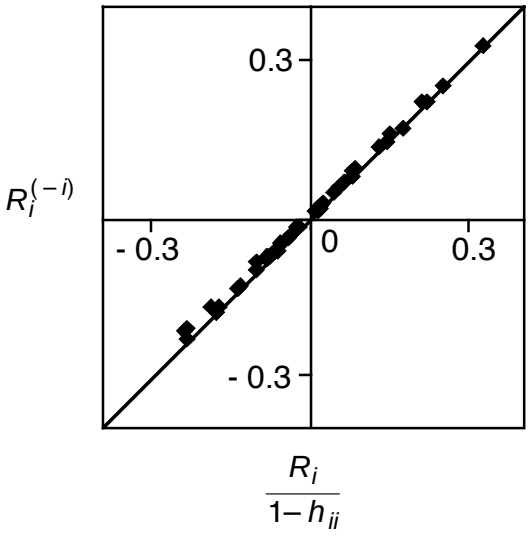

Figure 4: Application of relation (5) to a neural model of sin(x)/x with two hidden neurons, whose output is shown on Figure 7.

For minima that are stable with respect to the leave-one-out procedure, the validity of approximating E_t by E_p depends on the validity of the Taylor expansions used to derive relation (5). Checking this validity can be performed by estimating the curvature of the cost function (see (Antoniadis, 1992)); alternatively, one can actually perform the leave-one-out procedure for a model selected on the basis of E_p, and compare E_t and E_p. This is illustrated on Figure 4, for an approximation of $\frac{\sin x}{x}$ with two hidden neurons, for the minima that are stable and with full rank.

The time required for the computation of the quantities $\{h_{ii}\}$ from relation (2) and of the leave-one-out score E_p at the end of each training is negligibly small as compared to the time required for training; therefore, our procedure divides the computation time by N, as compared to the standard leave-one-out, where N is the number of examples.

3.3. *Conclusion*

We have shown in this section that E_p is an efficient basis for selecting a model, within a family of parameterized functions; furthermore, it eliminates automatically the solutions that are rank-deficient. Once a model has been selected on this basis for several families of parameterized functions, one has to select the appropriate architecture. This is addressed in the next section.

4. SELECTION OF THE OPTIMAL ARCHITECTURE

Having selected, for each architecture, e.g. for each number of hidden neurons, the minimum with the smallest value of E_p, we have to define a way of choosing the best architecture. It is

known that, among models with approximately the same performances, one should favor the model with the smallest architecture. In practice, however, the tradeoff between parsimony and performance may be difficult to perform; referring to Figure 3, the choice between 2, 3 or 4 hidden neurons is not easy in the absence of further information.

In this section, we show that the leverage may be used as an additional element in the process of choosing a model among candidate models having different architectures.

4.1. *Leverage distribution*

In a purely black-box modeling context, all data points of the training set are equally important; therefore, we suggest that, among models with approximately the same generalization error estimates (E_p), one should select the model for which the influence of the training examples is most evenly shared[5], i.e. for which the distribution of the leverages h_{ii} is narrowest around $\frac{q}{N}$. For example, the models with 2 and 4 hidden neurons of Figure 3 have significantly different leverage distributions as shown on Figure 5. As expected from relation (3), the leverages $\{h_{ii}\}_{i=1,\ldots,N}$ are centered on the corresponding values of $\frac{q}{N}$. However, the distribution is broader for the model with 4 hidden neurons. The width of the distribution is due to the fact that, for $x \in [8; 13]$, the number of examples is too small given the number of parameters of the model; this results in confidence intervals on the model output that are significantly larger in the corresponding region of input space than elsewhere.

---

[5] unless it is explicitly desired, for some reason arising from prior knowledge on the modeled process, that one or more example be of special importance.

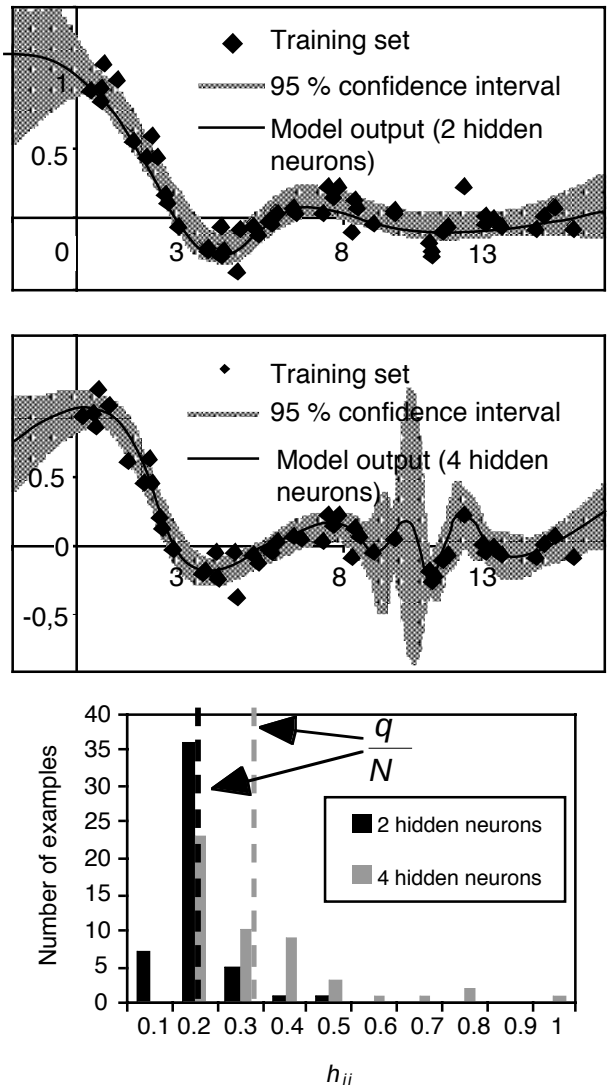

Figure 5: Model outputs (left) and leverage distributions for the models with 2 and 4 hidden neurons selected on the basis of E_p.

In order to characterize the leverage distribution of a given model, it is appropriate to use the mean value of the quantities $\left\{\sqrt{h_{ii}}\right\}_{i=1,\ldots,N}$, since the latter are involved in the computation of the confidence intervals. Indeed, the application of relation (6) to example *i* of the training set yields:

$$E\left(Y_p \middle| \boldsymbol{x}^{\boldsymbol{i}}\right) \in f\left(\boldsymbol{x}^{\boldsymbol{i}}, \boldsymbol{\theta}_{LS}\right) \pm t_{\alpha}^{N-q} s \sqrt{h_{ii}} \tag{10}$$

Therefore, we define the quantity:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} \sqrt{\frac{N}{q} h_{ii}} \tag{11}$$

Using Cauchy's inequality and relations (3) and (4), one can derive the following properties for μ, irrespective of N and q:

$$\begin{cases} \mu \leq 1 \\ \mu = 1 \Leftrightarrow \left(h_{ii} = \dfrac{q}{N} \right) \forall i \in [1, ..., N]. \end{cases} \tag{12}$$

Therefore, μ is a normalized parameter that characterizes the distribution of the $\{h_{ii}\}_{i=1,...,N}$ around their mean value. Irrespective of N and q, the closer μ to 1, the more evenly shared the influences among the training examples. Hence, μ can be used as an additional indication of the overfitting of a model.

To illustrate this, Figure 6 shows, in addition to the curves of Figure 3, the values of μ, and an estimate of the generalization error, obtained on a separate representative test set (100 examples). To this end, we define the Generalization Mean Square Error (*GMSE*) as:

$$GMSE = \sqrt{\frac{1}{\text{card}(Test\ set)} \sum_{x \in Test\ set} \left(y_p - f(\boldsymbol{x}, \boldsymbol{\theta}_{LS}) \right)^2} \tag{13}$$

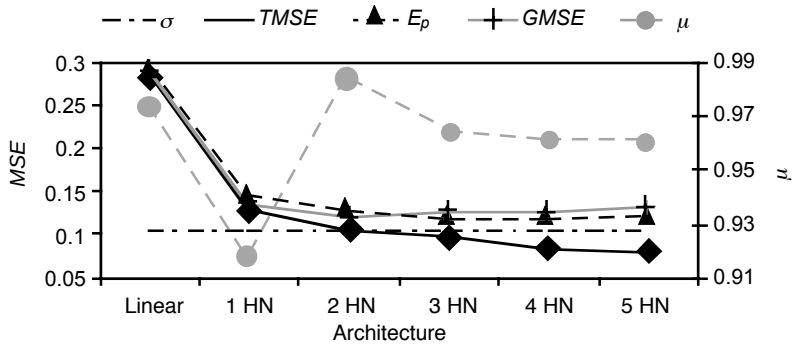

Figure 6: Performance indices of the solutions selected on the basis of E_p, including μ (right scale), and tested on a separate data set.

The analysis of Figure 6 leads to the following comments:

- the distinction between leverage distribution of the two models depicted on Figure 5 can actually be performed by μ (0.98 *vs* 0.96),
- the model with the highest value of μ is actually the model with the smallest value of *GMSE*. For larger architectures, both μ and *GMSE* get worse,
- indeed, models with more than 2 hidden neurons do not reach the performance of that with 2 hidden neurons. However, the increase of the *GMSE* turns out to be limited, which

demonstrates that selecting the minima on the basis of E_p is an effective means of limiting overfitting.

As a conclusion, considering several architectures with approximately the same values of E_p, the most desirable solution is the solution with the highest value of μ in the context of black-box modeling, i.e. if there is no reason, arising from prior knowledge, to devote a large fraction of the available degrees of freedom to one or to a few specific training samples.

4.2. *Model selection and extension of the training set*

In the previous section, we considered the situation where one wishes to select a model built from a data set that is available once and for all. It is often the case, however, that additional samples may be gathered, in order to improve the model. Then the following question must be addressed: in what region(s) of input space would it be useful to gather new data? We show in the following that the approach to model selection described in the previous section can be helpful in this respect.

To this end, one has to define a maximal acceptable value for the confidence interval. Using the expression of that interval (relation (6)), one can choose, for instance, the following threshold:

$$t_{\alpha}^{N-q}\, s \sqrt{\boldsymbol{z}^{\mathrm{T}} \left(Z^{\mathrm{T}} Z\right)^{-1} \boldsymbol{z}} < t_{\alpha}^{N-q}\, s \qquad (14)$$

This guarantees that the confidence on the model output $f\left(\boldsymbol{x}, \boldsymbol{\theta}_{LS}\right)$ is not poorer than that on the measured output $Y_p \,|\, \boldsymbol{x}$.

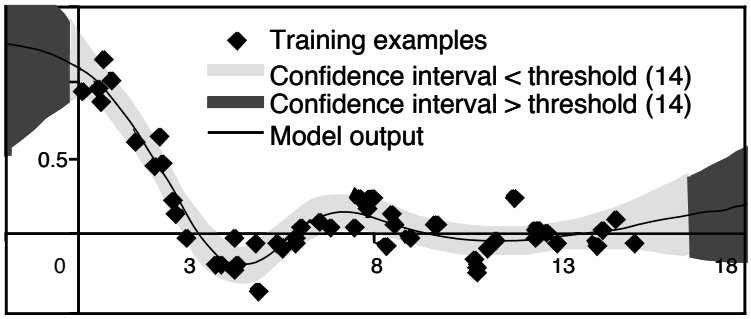

Figure 7: Application of threshold (14) to the confidence interval on the model output (2 hidden neurons).

To start with, consider the model with 2 hidden neurons, which has been selected, using μ, as the best possible model given the available training set (Figure 6): according to the threshold (14), this model should not be used outside the interval $x \in$ [-0.3, 16.5] (see Figure 7). In order to use the model outside this input space area, additional data should be gathered. This is not surprising since the interval [-0.3, 16.5] is roughly the interval from which the training set was drawn.

However, if it is desirable to improve the model *within* this interval, the confidence interval on the model with 2 hidden neurons does not provide any information for detecting where additional examples would be required. To that effect, we have to choose slightly more overfitted models, such as the model with 4 hidden neurons shown on Figure 5, and consider their confidence intervals. Then, the application of threshold (14) shows areas of input space, both within and outside the interval $x \in [-0.3, 16.5]$, where additional examples would be helpful (see Figure 8): the large confidence interval between 9 and 11 shows that the corresponding h_{ii} are large, i.e. that a large fraction of the degrees of freedom of the 4-hidden-neuron network was used to fit the training examples within this interval. Therefore, gathering new data in this area would indicate whether the wiggle of the output in this area is significant, or whether it is an artifact due to the small number of samples.

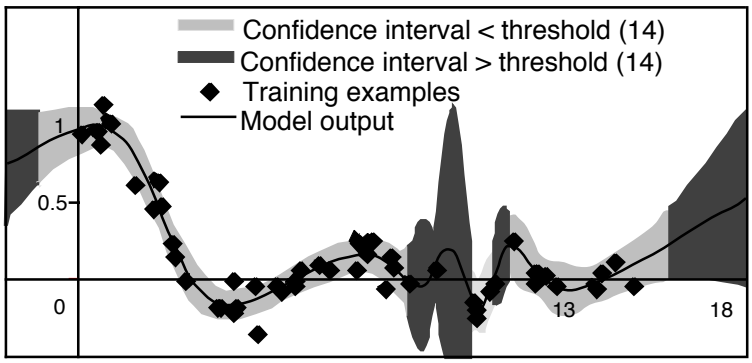

Figure 8: Application of threshold (14) to the confidence interval on the model output (4 hidden neurons).

Hence, considering slightly oversize models for selection is a means for detecting areas of input space where gathering additional data would be appropriate. Once additional examples are gathered, the selection procedure is performed again.

4.3. *Discussion*

For didactical reasons, the selection method discussed in sections 3 and 4 was split into two phases; for a given training set, architectures of increasing sizes were considered, as follows:

- For each architecture, perform several trainings, starting with various random weights,
  - For each model, compute the leverages $\{h_{ii}\}_{i=1,\ldots,N}$ of the training examples and check the validity of relations (2) and (3):
    - Discard the model in case of rank deficiency,
    - Otherwise, compute E_p,
  - Keep the model with the smallest value of E_p, and compute its parameter μ,
- Among the candidate models exhibiting a satisfactory E_p, select the model with the highest value of μ.

Thus, the method relies essentially on two quantities that can be compared, for a given problem, irrespective of the model size: E_p, which is an estimate of the generalization error, and μ, an index of the degree of overfitting of the model.

Hence, one should consider, for various architectures, all candidate models as a pair (E_p, μ), and perform model selection directly on the basis of the position of each model in the E_p - μ plane.

For instance, on the $\frac{\sin x}{x}$ problem, the selection should be performed by considering Figure 9, which represents about 1,000 candidate models for 5 different architectures. According to the required tradeoff between E_p and μ, the final model should be selected within the indicated area: models outside this area should not be chosen, for one can always select a better one (i.e. with a smaller E_p *and* a higher μ). Therefore, architectures with 1 and 5 hidden neurons can definitely be discarded.

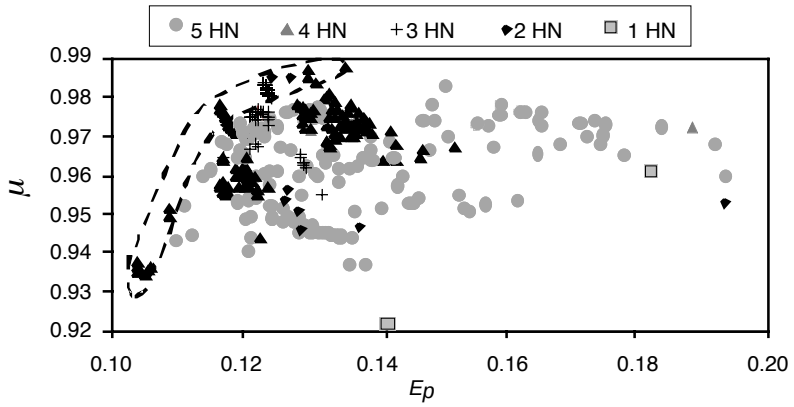

Figure 9: Performances of the candidate models with 1 to 5 hidden neurons.

As explained in section 4.2, the model designer should perform his final choice depending on his objectives. If he wants the best possible model given this particular training set, he should favor, within this dotted area, the solutions with the highest possible value of μ. Conversely, if he has the opportunity of gathering additional training examples, and wants to select the most relevant ones, he should favor slightly overfitted models: within this dotted area, he should favor E_p. The confidence intervals for the prediction of such a model indicate the areas, in input space, where gathering data would be desirable. With this new data, the selection procedure may be performed again, which may lead to less complex models.

5. VALIDATION ON NUMERICAL EXAMPLES

In the present section, we illustrate the use of our method, on two academic examples:

- we demonstrate, on a teacher-student problem, that the method we propose gives very good results, in contrast to the usual leave-one-out approach,

- for comparison purposes, we model a simulated process investigated previously in (Anders & al., 1999),

and we outline briefly an industrial application.

5.1. *Comparison to the standard leave-one-out approach on a teacher-student problem*

We consider the following problem: a data base of 800 examples is generated by a neural network with 5 inputs, one hidden layer of five neurons with sigmoidal (tanh) nonlinearity, and a linear output neuron:

$$E\left(Y_p \middle| \boldsymbol{x} \right) = \alpha_0 + \sum_{i=1}^{5} \alpha_i \tanh\left(\beta_i^0 + \sum_{j=1}^{5} \beta_i^j x_j \right) \tag{15}$$

Thus, we guarantee that the family of parameterized functions, in which we search for the model, actually contains the regression function. The weights and the inputs of this teacher network are uniformly distributed in [-1, +1] and [-3, +3] respectively, in order to use the sigmoids in their nonlinear zone. A Gaussian noise is added to the output, with a standard deviation equal to 20 % of the unconditional standard deviation of the model output: $\sigma^2 = 0.05$. Student networks with five inputs are investigated. We show in the following how all the above results can be successfully used to perform model selection. To this end, 300 examples are used to perform both training and model selection as described in the previous sections, and 500 examples are devoted to the tests.

All results reported below were obtained by estimating the gradient of the cost functions by backpropagation, and minimizing the cost functions by the BFGS algorithm (Press & al., 1988). The values of $\{h_{ii}\}_{i=1,\ldots,N}$ are computed by singular value decomposition of matrix Z, as presented in Appendix 1.

For increasing architecture sizes (from 1 to 9 hidden neurons), the minimum of the cost function with the smallest value of E_p was selected, and the following values were computed: *TMSE* and μ on the training set, and *GMSE* on the separate test set. These results are summarized on Figure 10.

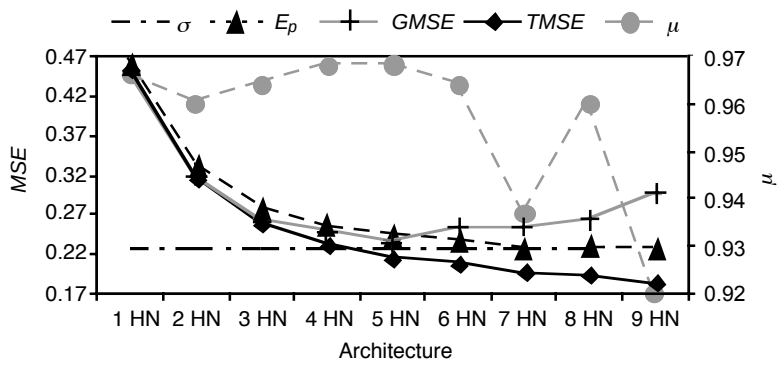

Figure 10: Performances of the solutions selected on the basis of E_p, including μ (right scale), and tested on a separate data set (teacher-student problem).

As on the $\frac{\sin x}{x}$ problem, E_p decreases monotonically and reaches approximately the value of the standard deviation of the noise. Based on the values of μ, the optimal solution appears to be the model having 5 hidden neurons. Indeed, this proves to be the model for which the generalization error *GMSE* is smallest. Remarkably, the weights of this network are almost identical to those of the teacher network (they should not be expected to be strictly identical, since noise is added to the output of the teacher network during generation of the data); actually, the weights of this network are identical to those obtained when training is performed *with initial weights equal to the teacher's weights*: this demonstrates that the selection method really finds the best possible model.

By contrast, the generalized leave-one-out approach appears to give very poor results:

- if leave-one-out is performed on the minima with the smallest value of *TMSE*, without checking Z's rank nor that the stability of the minima for the usual leave-one-out, the error E_t decreases as hidden neurons are added, and becomes significantly smaller than the standard deviation of the noise. Indeed, the global minima of the cost functions corresponding to architectures having 5 (i.e. the teacher network architecture) to 9 hidden neurons have a rank deficiency and are therefore overfitted solutions with poor *GMSE*,
- if leave-one-out is performed on the minima with the smallest value of *TMSE* among the minima without rank deficiency, without checking that the minima are stable for the usual leave-one-out, the error E_t reaches a plateau close to the standard deviation of the noise. However, both E_p and *GMSE* prove that the performances of these models are worse than expected from E_t. In fact, the corresponding minima appear to be unstable for the usual leave-one-out, which makes E_t invalid,
- if the procedure is restricted to the minima with the smallest value of *TMSE* among the minima without rank deficiency *and* stable for the usual leave-one-out, almost all minima, from 5 to 9 hidden neurons, are rejected. Furthermore, the procedure is extremely lengthy.

This exemplifies the limitations of the conventional generalized leave-one-out. When the number of training examples is small given the complexity of the family of parameterized functions, the withdrawal of some training examples makes the minima of the cost function unstable for the usual leave-one-out. By contrast, performing leave-one-out on the basis of relation (5) prevents from these stability problems. Furthermore, the computational overhead of this selection method is negligible, since we only have to compute the $\{h_{ii}\}_{i=1,\ldots,N}$ for each minimum of the cost function.

5.2. *Comparison to other selection methods on a benchmark problem*

In (Anders & al., 1999), the authors introduce a two-input process, simulated with the following regression function:

$$E\left(Y_p \,\middle|\, \boldsymbol{x} \right) = -0.5 + 0.2\, x_1^2 - 0.1 \exp(x_2) \qquad (16)$$

The inputs x_1 and x_2 are drawn from a normal distribution. Gaussian noise is added to the output, with a standard deviation equal to 20 % of the unconditional standard deviation of the model output, i.e. $\sigma^2 = 5\ 10^{-3}$.

To perform statistically significant comparisons, 1000 training sets of 500 examples each were generated with different realizations of the noise (the inputs remaining unchanged); a separate set of 500 samples was used for computing the Generalization Mean Square Error (*GMSE*). The following performance index was used:

$$\rho = \frac{GMSE^2 - \sigma^2}{\sigma^2} \,.\, 100\ \% \qquad (17)$$

Using several model selection techniques (hypothesis testing, information criteria and 10-fold cross-validation, each of them being followed - if necessary - by network pruning), the best performance (averaged over the 1000 training sets) reported in (Anders & al. 1999) was $EMBE\Delta\rho$ = 28 % (standard deviation not indicated). The authors state that these (relatively) poor performances arise from the complexity of the true regression function (16).

Under the same conditions, we performed model selection according to: (i) the present method, i.e. on the basis of E_p and μ as described in sections 3 and 4; (ii) the bayesian approach to weight decay[6] described in (MacKay, 1992). In both cases, the performance indices were very satisfactory. The results reported in this section are summarized on Table 1, where the present method is referred to as LOCL (Local Overfitting Control via Leverages).

---

[6] Due to the difficulties encountered while implementing this approach, we made the assumption that the noise level (meta-parameter β in the cost function) was known and set it to its true value. Without this strong assumption, the performances of the models selected on the basis of the "evidence", would certainly have been worse.

		(Anders & al. 1999)	LOCL method		Weight decay
			Outliers not removed	Outliers removed	
500 training samples	Avg. ρ	28 %	3 %	3 % *	2 %
	Std. dev.	Not indicated	2 %	2 %	2 %
100 training samples	Avg. ρ		126 %	27 % **	54 %
	Std. dev.		632 %	28 %	36 %
* No outlier detected			** 3 % outliers detected		

Table 1: Summary of numerical results on a benchmark problem.

Hence, this academic problem can be accurately and consistently solved by our method as well as by weight decay.

As proposed in (Rivals & al., 2000), a reduction of the size of the training set from 500 to 100 samples is of interest. Unlike (Rivals & al., 2000), we simulated the 1000 new training sets with different values of the noise *and* of the inputs: because of the decrease of the training set size, this was necessary to obtain statistically significant results[7]. Our method gave a value of ρ = 126 %, whereas, using weight decay[8], one obtains ρ = 54 %. This leads to the following comments:

1. in a problem with such a small number of training examples, knowing the true noise level[6] is a tremendous advantage which explains the superiority of our implementation of weight decay,
2. the standard deviation associated to our ρ shows the large scattering of the results.

However, advantage can be taken from the fact that our method is *local*, whereas alternative methods rely on *global* scores: we can show that the poor results reported above are due to only a few points of the test set. Just as in section 4.2, one can define a threshold that the confidence interval on the model output should not exceed. Choosing threshold (14), one obtains the following results: an average 3 % (std. dev. 2 %) of the test examples were discarded, resulting, on the remaining 97 % of the test samples, in an average model performance ρ = 27 %. This is comparable to the performances obtained by (Anders & al. 1999) using a training set that was 5 times as large.

---

[7] Otherwise, the results depend very strongly on the location of the training examples in input space.

[8] Keeping the inputs unchanged and using only E_p followed - if necessary - by network pruning, [Rivals & al., 2000] obtained $\rho EMBED$ = 140 % (std. dev. not given).

5.3. *An industrial application*

This method was used in a large-scale, industrial application: the modeling of the spot welding process. The quantity to be predicted was the diameter of the weld as a function of physical parameters measured during the process. Because of the relatively small number of examples at the beginning of the project, and of the economic and safety issues involved, model selection was a critical point.

Spot welding is the most widely used welding process in the car industry: millions of such welds are made every day. Two steel sheets are welded together by passing a very large current (tens of kiloamperes) between two electrodes pressed against the metal surfaces, typically for a hundred milliseconds (Figure 11). The heating thus produced melts a roughly cylindrical region of the metal sheets. After cooling, the diameter of the melted zone - typically 5 mm - characterizes the effectiveness of the process; a weld spot whose diameter is smaller than 4 mm is considered mechanically unreliable; therefore, the spot diameter is a crucial element in the safety of a vehicle. At present, no fast, non-destructive method exists for measuring the spot diameter, so that there is no way of assessing the quality of the weld immediately after welding. Therefore, a typical industrial strategy consists (i) in using an intensity that is much larger than actually necessary, which results in excessive heating, which in turn leads to the ejection of steel droplets from the welded zone (hence the “sparks” that can be observed during each welding by robots on automobile assembly chains), and (ii) in making a much larger number of welds than necessary, just to be sure that a sufficient number of valid spots are produced. Both the excessive current and the excessive number of spots result in a fast degradation of the electrodes, which must be changed or redressed frequently.

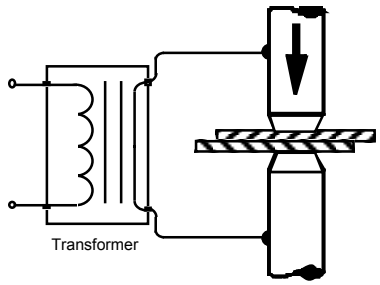

Figure 11: The welding process: two electrodes are pressed against the metal sheets and a strong electrical current is flown through the assembly.

For all the above reasons, the modeling of the process, leading to a reliable on-line prediction of the weld diameter, is an important industrial challenge. Modeling the dynamics of the welding process from first principles is a very difficult task, because (i) the computation time necessary for the integration of the partial differential equations of the knowledge-based model is many orders of magnitude larger than the duration of the process, which precludes real-time prediction of the spot diameter, and because (ii) many physical parameters

appearing in the equations are not known reliably. These considerations led to considering black-box modeling as an alternative. Since the process is non-linear and has several input variables, neural networks were natural candidates. The main goal was to predict the spot diameter from measurements performed during the process, immediately after weld formation, for on-line quality control.

The main concerns for the modeling task were the choice of the model inputs, and the limited amount of examples available initially in the database.

The quantities that are candidates for input selection are mechanical and electrical signals that can be measured during the welding process. Input selection was performed by forward stepwise selection of polynomial models, whereby the significance of adding a new input or removing a previously selected input is tested by statistical tests. The variables thus selected were subsequently used as inputs to neural networks. The experts involved in the knowledge-based modeling of the process validated this set.

Because no simple nondestructive weld diameter measurement exists, the database is built by performing a number of welds in well-defined condition, and subsequently tearing them off; the melted zone, remaining on one of the two parts, is measured. This is a lengthy and costly process, so that the initial training set was made of 250 examples. Using the confidence interval estimates (6), and the training set extension strategy discussed in section 4.2, further experiments were performed, so that, finally, a much larger database became available, half of which was used for training and half for testing (since the present selection method does not require any validation set). Model selection was performed using the full procedure discussed in section 4.

A full presentation of the results is beyond the scope of the present paper. Typical results are shown on Figure 12, which shows the scatter plots for spot diameter prediction on the training set and on the completely independent test set (for a certain type of steel sheet), together with the confidence intervals. The leave-one-out score E_p is equal to 0.27, while the *EQMT* is 0.23.

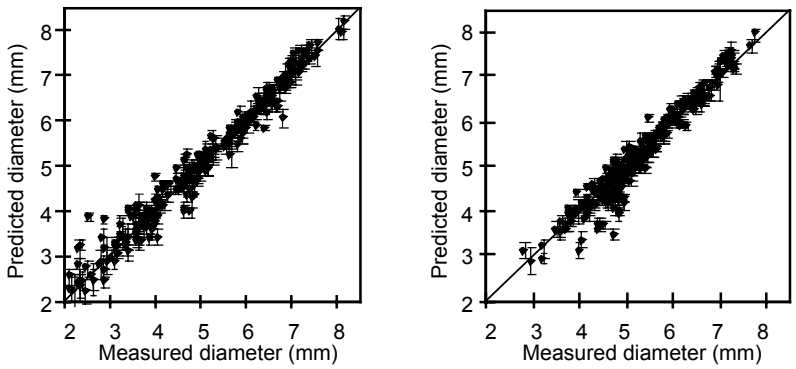

Figure 12: Scatter plots with confidence intervals for diameter prediction; left: training set; right: test set

The full description of this application is beyond the scope of this paper; non-confidential data can be found in (Monari, 1999).

6. CONCLUSION

A model selection method relying on local overfitting control via leverages (termed LOCL method) has been presented; it is based on the computation of the leverage of each example on the candidate models. The leverage of a given example can be regarded as an indication of the percentage of the degrees of freedom of the model that is used to fit the example. This allows a precise monitoring of overfitting, since the method relies on *local* indicators in sample space (the leverages), instead of relying on *global* indicators such as a cross-validation score or the value of a penalty function. Although it is similar in spirit to the generalized leave-one-out procedure, it is very economical in computation time, and it avoids the stability problems inherent to leave-one-out. Furthermore, the values of the leverages (or of the confidence interval computed therefrom) give an indication of areas, in input space, where new examples are needed. This method has been validated in a large-scale industrial application.

APPENDIX 1

COMPUTATION OF THE LEVERAGES

The present appendix shows how the values of the all-important quantities $\{h_{ii}\}_{i=1,\ldots,N}$ can be computed without matrix inversion by making use of the Singular Value Decomposition (see for instance (Press & al, 1988)) of matrix Z, which can always be performed, even if matrix Z is singular. Z can be written as:

$$Z = U\,W\,V^{\mathrm{T}}, \tag{A.1}$$

where:

- U is a (N, q) column-orthogonal matrix (i.e. $U^{\mathrm{T}}\,U = I$),
- W is a (q, q) diagonal matrix with positive or zero elements (the singular values of Z),
- V is a (q, q) orthogonal matrix (i.e. $V^{\mathrm{T}}\,V = V\,V^{\mathrm{T}} = I$).

We thus obtain:

$$(Z^{\mathrm{T}}\,Z)^{-1} = (V\,W\,U^{\mathrm{T}}\,U\,W\,V^{\mathrm{T}})^{-1} = (V\,W^{2}\,V^{\mathrm{T}})^{-1} = V\,W^{-2}\,V^{\mathrm{T}}. \tag{A.2}$$

Therefore, the elements of this (q, q) matrix can easily be computed as:

$$\left(Z^{\mathrm{T}}\,Z\right)^{-1}_{lj} = \sum_{k=1}^{q} \frac{V_{lk}\,V_{jk}}{W_{kk}^{2}} \tag{A.3}$$

From (A.3) and $h_{ii} = \boldsymbol{z}^{i\mathrm{T}}\,(Z^{\mathrm{T}}\,Z)^{-1}\,\boldsymbol{z}^{i}$, one gets:

$$h_{ii} = \sum_{k=1}^{q} \left(\frac{1}{W_{kk}} \sum_{j=1}^{q} Z_{ij}\,V_{jk} \right)^{2} \tag{A.4}$$

APPENDIX 2

STABILITY OF A MODEL FOR THE USUAL LEAVE-ONE-OUT PRCEDURE

Consider a training procedure using a deterministic minimization algorithm of the cost function[9]; assume that a minimum of the cost function has been found, with a vector of parameters $\boldsymbol{\theta}_{LS}$. Assume that example i is withdrawn from the training set, and that a training is performed with $\boldsymbol{\theta}_{LS}$ as initial values of the parameters, leading to a new parameter vector $\boldsymbol{\theta}_{LS}^{(-i)}$; further assume that example i is subsequently reintroduced into the training set, and that a new training is performed with initial values of the parameters $\boldsymbol{\theta}_{LS}^{(-i)}$:

> The minimum of the cost function corresponding to a parameter vector $\boldsymbol{\theta}_{LS}$ is said to be "**stable for the usual leave-one-out procedure**" if and only if, for each $i \in [1, \ldots, N]$, starting from $\boldsymbol{\theta}_{LS}$, the procedure described above retrieves $\boldsymbol{\theta}_{LS}$.

This definition is illustrated graphically on Figure 13.

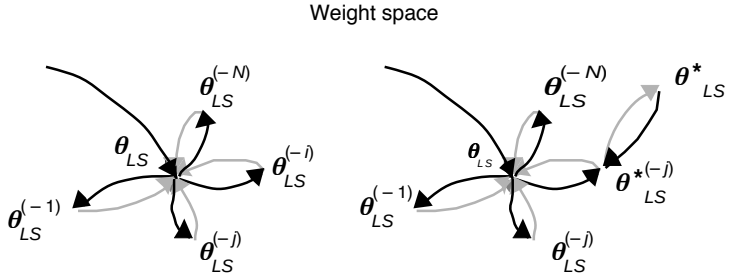

Figure 13: Minima that are stable (left) and unstable (right) for the usual leave-one-out procedure.

On this Figure, in the unstable case, after the removal of example i (and convergence to $\boldsymbol{\theta}_{LS}^{(-i)}$), the replacement of the same example into the training set makes the learning procedure converge to another solution with parameters $\boldsymbol{\theta}_{LS}^{*}$. In such a case, E_t is not an estimate of the generalization performance of the model with parameters $\boldsymbol{\theta}_{LS}$, since the prediction error $R_i^{(-i)}$ actually corresponds to the model with parameters $\boldsymbol{\theta}_{LS}^{*}$.

This definition is different from the stability considerations that were introduced by other authors in order to derive bounds on the estimate of the generalization performance (see

---

[9] This excludes such algorithms as simulated annealing, probabilistic hill climbing, etc., which may overcome local minima.

(Vapnik, 1982)). For instance, bounding the difference between the true error and the leave-one-out estimate thereof (with an arbitrary accuracy at a given level of significance) requires some stability of the training algorithm irrespective of the available data set (Kearns & al., 1997). Therefore, this stability does not depend on the considered minimum of the cost function: our definition of stability is less stringent than the alternative one.

Anyway, it should be remembered that, in contrast to the usual leave-one-out, the validity of our approach does not depend on any stability condition.

LITERATURE REFERENCES

Anders, U. & Korn, O. (1999). Model Selection in Neural Networks. *Neural Networks 12*, 309-323.

Antoniadis, A., Berruyer, J. & Carmona, R. (1992). *Régression non linéaire et applications*. Paris: Economica.

Breiman, L. (1996). Heuristics of Instability and Stabilization in Model Selection. *Annals of Statistics 24*, 2350-2383.

Geman, S., Bienenstock, E. & Doursat, R. (1992). Neural Networks and the Bias / Variance Dilemma. *Neural Computation* 4, 1-58.

Kearns, M. & Ron D. (1997). Algorithmic Stability and Sanity-Check Bounds for Leave-One-Out Cross Validation. *Tenth Annual Conference on Computational Learning Theory*, 152-162 (Association for Computing Machinery Press).

MacKay, D. J. C. (1992). A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation* 4, 448-472.

Monari, G. (1999). Sélection de modèles non linéaires par leave-one-out: étude théorique et application des réseaux de neurones au procédé de soudage par points. *Thèse de l'Université Paris 6*. Available from: http://www.neurones.espci.fr/Francais.Docs/publications.html.

Monari, G. & Dreyfus, G. (2000). Withdrawing an Example from the Training Set: an Analytic Estimation of its Effect on a Nonlinear Parameterized Model. *Neurocomputing Letters* 35, 195-201.

Moody, J. (1994). Prediction Risk and Neural Network Architecture Selection. *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*. Cherkassky, V., Friedman, J.H. & Wechsler, H. (eds). Springer Verlag.

Press, W. H., Teukolsky, S. A., Flannery, B. P. & Vetterling, W. T. (1988). *Numerical recipes in C: the art of scientific computing.* Cambridge University Press.

Rivals, I. & Personnaz, L. (2000). *A Statistical Procedure for Determining the Optimal Number of Hidden Neurons of a Neural Model,* ICSC Symposium on Neural Computation, Berlin.

Seber, G.A.F., & Wild, C.J. (1989). *Nonlinear regression*. Wiley Series in Probability and Mathematical Statistics. New York, New York: John Wiley & Sons.

Sjöberg, J. & Ljung, L. (1992). Overtraining, Regularization and Searching for Minimum in Neural Networks. *Technical Report LiTH-ISY-I-1297*, Department of Electrical Engineering. Linkoping University, S-591 93 Linkoping, http://www.control.isy.liu.se.

Stone, M. (1974). Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society B* 36, 111-147.

Vapnik V.N. (1982). Estimation of Dependances Based on Empirical Data. *Springer Verlag*. New-York.

Zhou, G., Si, J., (1998). A Systematic and Effective Supervised Learning Mechanism Based on Jacobian Rank Deficiency. *Neural Computation* 10, 1031-1045.