Information Processing and Management of Uncertainty in Knowledge-based Systems, IPMU'1998, pp. 878-885, Paris, June 1998

The Hinges model: A one-dimensional continuous piecewise polynomial model

Eugenio Fco. SÁNCHEZ-ÚBEDA
Universidad Pontificia Comillas
Instituto de Investigación Tecnológica
Alberto Aguilera 23, 28015 Madrid - SPAIN
Eugenio.Sanchez@iit.upco.es

Louis WEHENKEL
Research Associate, F.N.R.S.
University of Liège - Institut Montefiore
Sart-Tilman B28 - B-4000 Liège BELGIUM
lwh@montefiore.ulg.ac.be

Abstract

In this article we propose an efficient approach to flexible and robust one-dimensional curve fitting under stringent high noise conditions. This is an important subproblem arising in many automatic learning tasks. The proposed algorithm combines the noise filtering feature of an existing scatterplot smoothing algorithm (the Supersmoother) with the flexibility and computational efficiency of piecewise linear hinges models. The former is used in order to provide a first approximation of the noise in the data, in a preprocessing step. Then, the latter are used in order to provide a closed form approximation of the underlying curve and further to reduce bias of the Supersmoother thanks to an efficient refitting algorithm, using updating formulas. The proposed technique is assessed on a synthetic test problem and one closer to real world data.

1 Introduction

In this paper we focus on one-dimensional regression, i.e. *curve fitting* from scatterplot data. Given a set of N points[1]

$$(x_i, y_i = f(x_i) + \epsilon_i)_{i=1,N} \tag{1}$$

the objective here is to find a simple enough function $\hat{f}(\cdot)$ such that the following equation holds

$$y_i = \hat{f}(x_i) + \hat{\epsilon}_i, \tag{2}$$

with small enough error estimates $\hat{\epsilon}_i$, as measured empirically by the overall mean square error (MSE) :

$$MSE = N^{-1} \sum_{i=1,N} \hat{\epsilon}_i^2. \tag{3}$$

This problem, although simple with respect to multidimensional regression problems, is highly relevant in practice. In particular, in many automatic learning algorithms it appears as a subproblem solved repeatedly in the attempt of building multidimensional models from combinations of elementary one-dimensional bricks.

For example, in projection pursuit regression the aim is to build a linear combination of such one-dimensional models along selected projections of the multidimensional data [1, 2, 3]. Similarly, classification or regression trees and also fuzzy trees approximate multidimensional functions as sums of products of one-dimensional ones [4, 5, 6, 7]. Other examples of algorithms which could take advantage of such a feature are multilayer perceptrons and radial basis functions networks [8].

In such multidimensional learning problems, it is not unusual that several thousand to several hundred thousand candidate one-dimensional curve fitting subproblems need to be solved when building up a multidimensional model. Furthermore, typical sample sizes may range from several hundred to several hundred thousand. Thus, the most paramount feature of an algorithm is computational efficiency in terms of both model learning and model use.

In addition, given the fact that in automatic learning it is generally desirable to interpret the obtained result, we seek functions in closed form approximations of maximal simplicity. Also, in solving the curve fitting problem we try to avoid as much as possible strong (e.g. parametric) assumptions about the underlying function. For the same reason we make only minimal assumptions on the statistical nature of the process (see foonote 1).

Since the above curve fitting problem is obviously not well defined, its solution requires some ad hoc assumptions, e.g. concerning regularity, complexity or a priori probability, which may be either specified explicitly or incorporated implicitly in an algorithm solving the problem. Thus, loosely speaking we will consider that a function which realizes a good compromise between these ad hoc assumptions and data fit will be a good solution to our problem.

In order to reach the above objectives, we propose to solve the curve fitting problem in three successive steps :

1. scatterplot smoothing, i.e. computation of the random noise estimates $\hat{\epsilon}_i$;
2. representation, i.e. approximating the smoothed scatterplot $s_i = y_i - \hat{\epsilon}_i$ by a simple function of x_i in closed

[1] The x_i are supposed to be independent and identically distributed samples from an unknown probability density function, and the true errors are supposed to have zero mean : $f(x) = E\{y|x\}$. In the sequel we suppose, without loss of generality, that scatterplots are sorted along the x values

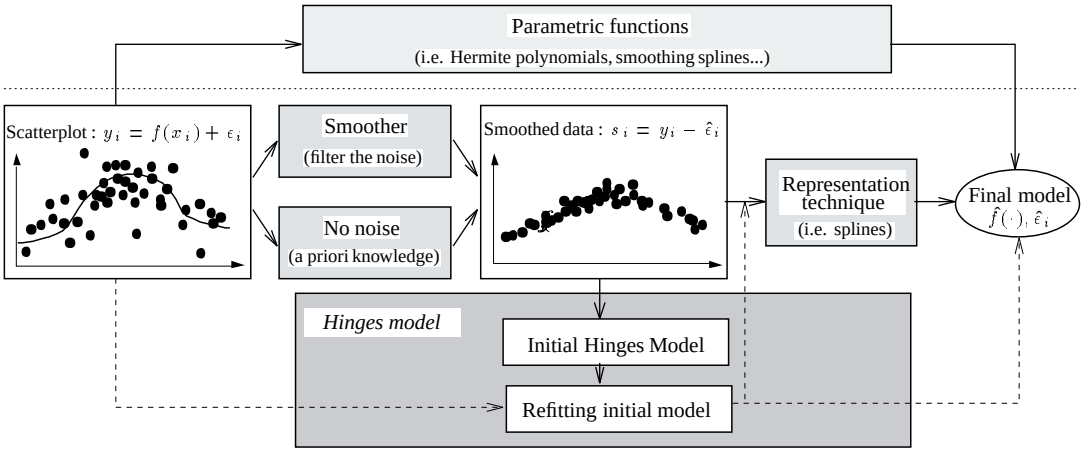

Figure 1: Parametric vs. non-parametric curve fitting.

form, with an appropriate number of parameters;

3. refitting, i.e. tuning the parameters of the closed form approximation to further minimize the overall MSE.

Figure 1 shows schematically the proposed approach combining scatterplot smoothing and the Hinges model, suggesting also in the upper part how parametric approaches, e.g. based on Hermite polynomial functions, would try to solve the overall problem in a single step [9, 10].

For the scatterplot smoothing we use the so-called Supersmoother proposed by Friedman in the context of projection pursuit regression. This technique has indeed shown to be effective in extracting the structure from very diverse types of scatterplot data. Then, for representation, we consider both piecewise linear and piecewise cubic Hinges (see [11] for cubic ones). They are indeed at the same time simple and flexible enough to represent a very large variety of functions, smooth or not. Step 2 allows us to select the appropriate number of pieces in the Hinges model (i.e. control the variance of the model), while refitting allows reducing bias with respect to the Supersmoother. We provide algorithms in order to solve these tasks in a computationally very efficient way. In particular, step 2 uses a greedy divide and conquer strategy together with a computationally efficient pruning approach, while step 3 exploits updating formulas. Additionally, the proposed technique allows the fast and accurate computation of the derivatives, useful in many multidimensional automatic learning algorithms.

The rest of the paper is organized as follows. Section 2 introduces the linear hinges model and Section 3 describes the various steps required to obtain this model. Section 4 provides simulation results, both in terms of accuracy and computational efficiency, using two test problems. Section 5 draws some conclusions and provides some directions for further work. Mathematical details concerning the updating schemes are collected in the Appendix.

2 The linear hinges model

In the context of one-dimensional regression or curve fitting, continuous piecewise polynomial functions are not

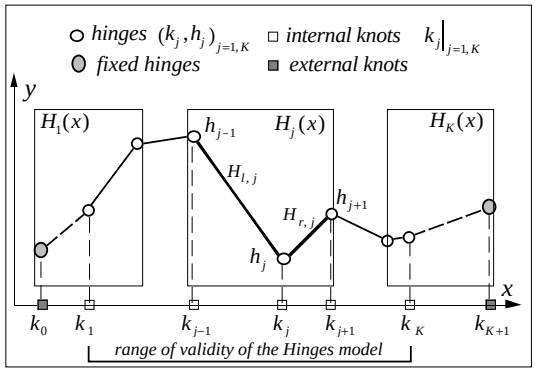

Figure 2: Hinges model definition.

new. The success of these methods is based on the idea of partitioning the interval $[x_1, x_N]$ appropriately into small intervals, on each of which the true underlying function $f(\cdot)$ (2) is then approximated by a suitable polynomial. These piecewise functions are completely defined by continuity conditions and a set of parameters : typically, the order of the polynomials used (a user defined integer parameter) and several other real-valued parameters (the number depends on the order used) associated with the points defining the pieces, the so-called knots.

While the choice of the order of the polynomials is (usually) not very important, the choice of the knots (number and location) tends to be crucial because they control the flexibility of the model.

Figure 2 shows the proposed model. There are K hinges, one for each internal knot. The two 'external' knots specify the boundary conditions for unique solution. They must be out of the interval of validity $[x_1, x_N]$ of the Hinges model.

This piecewise linear model is thus defined by the two external knots (Fig. 2) and by $2K$ parameters, the (x, y) coordinates of the internal knots[2] :

$$(k_0, h_0), (k_j, h_j)_{j=1,K}, (k_{K+1}, h_{K+1}) \qquad (4)$$

The complete Hinges model is formed by a set of hinges,

[2] In this paper we use the term 'knot' to denote both x and y coordinates of the points defining the hinges.

expressed by

$$\boldsymbol{H}(x) \equiv \{H_j(x) \ : k_{j-1} \leq x \leq k_{j+1}\}_{1 \leq j \leq K}. \quad (5)$$

One hinge is defined (Fig. 2) by the current (k_j, h_j) knot plus two straight-line segments $H_{l,j}$ and $H_{r,j}$ (connected to the knot j)[3]. Note that the hinge is also determined completely by three knots: previous, current and following knot. The previous and the following knot define the interval $[k_{j-1}, k_{j+1}]$ of validity of the current hinge (sketched by the rectangle of Fig. 2). Then, the model of one hinge can be expressed mathematically as :

$$H_j(x) \equiv 1[k_{j-1}, k_j] H_{l,j}(x) + 1(k_j, k_{j+1}] H_{r,j}(x) \quad (6)$$

where

$$\begin{array}{l} H_{l,j}(x) = h_{j-1} + (\Delta h_{l,j}/\Delta k_{l,j})(x - k_{j-1}) \\ H_{r,j}(x) = h_{j+1} + (\Delta h_{r,j}/\Delta k_{r,j})(x - k_{j+1}) \end{array} \quad (7)$$

$$\begin{array}{l} \Delta h_{l,j} = h_{j-1} - h_j; \Delta k_{l,j} = k_{j-1} - k_j \\ \Delta h_{r,j} = h_j - h_{j+1}; \Delta k_{r,j} = k_j - k_{j+1} \end{array} \quad (8)$$

and $1[k_{j-1}, k_j]$ and $1(k_j, k_{j+1}]$ are the indicator functions of the left and right intervals defining the hinge.

Note that the continuity of this Hinges model is clearly expressed by the relation :

$$H_{j-1}(k_j) = H_j(k_j) = H_{j+1}(k_j). \quad (9)$$

3 How the linear hinges model is obtained

Our strategy is simple (Fig. 1) : first use the Supersmoother for smoothing the scatterplot; then exploit its result to build an Initial Hinges Model (IHM); finally refit this latter model by minimizing the MSE, which yields the Linear Hinges Model (LHM) of reduced bias. Observe that this strategy combines the advantages of nonparametric models (the Supersmoother) with the qualities of the parametric ones.

Note that other strategies have been proposed in the literature to solve this problem like TURBO [12] and its successor MARS [13, 14] or the hinges and ramps of [15].

Next sections elaborate further on smoothing the scatterplot, deriving the IHM from the smoothed data and refitting the latter to yield the LHM.

3.1 Supersmoother

The supersmooother [2, 16] is based on local averaging of the scatterplot (1). It consists in computing the values s_i defined by :

$$s_i = ave\{y_j \ : i - b \leq j \leq i + b\} \quad (10)$$

where ave denotes some way of averaging like the mean or the median operators. The parameter b is the bandwidth or span of the smoother; it controls the tradeoff between continuity and flexibility. The Supersmoother uses a variable-span, determined by local linear fits and cross-validation as a function of the abscissa value.

[3]The subscripts l and r denote the left- and right-hand side of the hinge.

Among the many local averaging smoothers, we select this nonparametric method because it combines computational efficiency with a good capacity of getting the main structure of the scatterplot. In other words, this 'automatic' method provides good adaptation to varying curvature in the scatterplot and to varying noise levels. It is computationally efficient but suffers from slight under fitting, in many circumstances.

Note that the drawbacks associated with the Supersmoother have been reported in [9, 12]: (a) poor performance in very high-noise environments, (b) deficient estimation of its derivatives (using first-order differences of the estimates) and, (c) the 'untreatable' nature of the tabulated result (specific values associated with each observation). Problems (b) and (c) may be solved by using (parametric) Hermite polynomial functions [9]. In [17] authors propose also a parametric model (B-splines of order three, with equally spaced knots, their number being user defined). To avoid problem (c), in [18] authors propose approximating the tabulated result by a cubic spline, but the parameters of the spline are determined by a least squares fit to the Supersmoother results. Notice that the first two approaches depart from the non-parametric feature of the Supersmoother, while the latter is merely a representation technique.

3.2 The initial hinges model (IHM)

3.2.1 Overall strategy

The overall strategy we use to obtain the IHM consists in two complementary steps : (i) top-down growing procedure using a greedy recursive partitioning algorithm; (ii) pruning together with cross-validation to select the appropriate number of hinges. This strategy is commonly used in contexts such as decision trees [4, 19]. The available data set is partitioned into two subsets : the growing set used in the growing stage and the pruning set used after that for pruning.

3.2.2 Growing procedure

The objective of growing is to replace the smoothed scatterplot by a piecewise linear model, without loss of accuracy. In the context of describing in a natural manner lines and shapes of pictures, authors in [20] describe the so-called 'iterative end-point fits'. Because we know that our problem is well suited (i.e. fit something that looks like a 'smooth function'), we can use this efficient technique here. Note that because this strategy is intended for the very generic problem of representing a function by using a piecewise linear model, the approximation of the function can be as accurate as desired.

After filtering the noise with the smoother, we have a new set of smoothed data s_i (10), sorted along the x values. The growing procedure is initialized with a straight line defined by the two end points (see Fig. 3). Thus the first and last smoothed data become multiple knots [21], corresponding both to the first and last internal knots of the model, as well as to the external knots $(k_0 = k_1 = x_1; h_0 = h_1 = s_1)$, $(k_{K+1} = k_K = x_N; h_{K+1} = h_K = s_N)$. Then, this model is progressively refined using a one-dimensional

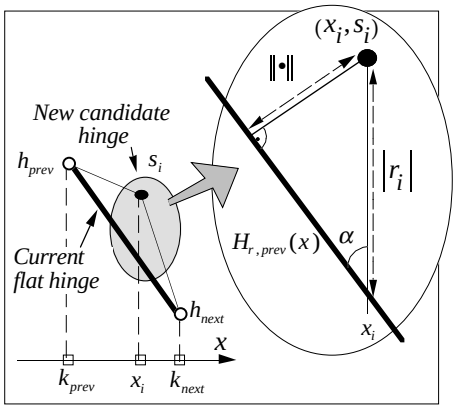

Figure 3: New and 'flat' hinge absolute residual.

recursive partitioning procedure. It starts considering the complete range of values of x_i and looks for the 'best' position of the (first) new knot (k_{new}, h_{new}) to split the interval $[k_1, k_N]$ in two new (non overlapping) sub-intervals $[k_1, k_{new}]$ and $[k_{new}, k_K]$. This splitting process is repeated recursively for each of the two new intervals thus created, until all points of the smoothed scatterplot are sufficiently well approximated by the resulting hinges model, which we denote by IHM_{max}.

The IHM_{max} obtained by using this top-down growing procedure can be represented by a binary tree (see example in §4). As in all recursive partitioning algorithms, (e.g. [22, 23]), three important issues need to be stated: (i) definition of candidate splits ; (ii) the selection of the 'best' split ; (iii) the termination criteria used.

Because we know that a Hinges model with one hinge at each point can correctly represent them, a natural strategy to obtain a good split is to use as candidate knots the smoothed data points falling in the active interval $[k_{prev}, k_{next}]$. Thus, for a given active interval the heuristic selects as best split k^*_{new} the abscissa value $x_* \in [k_{prev}, k_{next}]$ such that (x_*, s_*) 'differs more' of the current model (a flat hinge) in the active interval (see Fig. 3). We use the absolute residual as distance because it produces the same results than the Euclidean distance, but requires less computations. Thus, we need only to scan the smoothed data in $[k_{prev}, k_{next}]$ and select the one which is at the far end of the straight line.

The stopping criterion is as follows : if the MSE of the flat hinge in the current active interval (Fig. 3) is lower than a threshold then the interval is not split further. Note that this implies obviously that the distance of all the points of the smoothed scatterplot in this interval are well enough approximated by the flat hinge. In the worst case, the number of knots will be equal to the number of samples in the growing set, but in general it will be much smaller.

3.2.3 Pruning : selection of the IHM complexity

Our objective is to obtain the best possible representation (in terms of fidelity to the data) with the smallest number of parameters (in terms of complexity). The growing procedure yields fidelity but generally at the cost of high complexity, i.e. a high number of knots. The pruning procedure aims at selecting a subset of these latter so as to reduce complexity while preserving accuracy with respect to the original data.

Conceptually, pruning consists in two steps.

1. Generation of a sequence of hinges models of decreasing complexity (the pruning sequence):
$$IHM_{max}, IHM_{max-1}, \ldots IHM_0 \qquad (11)$$
removing knots one by one.
2. Selection of the best IHM from this sequence (11). This means that each pruned model is evaluated on an independent sample (the pruning set) and the one which realizes the best compromise between fidelity and data fit is selected.

Like in regression tree pruning, even for a moderate number of hinges in IHM_{max}, there is an extremely large number of distinct ways of pruning up to the simplest Hinges model. A selection of a reasonable number of different models, each of them with a different number of hinges, is necessary. Basically, the idea is to select, for a fixed number of hinges, the 'best' IHM.

However, unlike in regression trees, our heuristic growing procedure does not guarantee that for each new split the MSE of the model decreases. Moreover, it is possible that the first selected splits (associated to nodes in the top of the tree) are the clear candidates to be removed during the pruning phase. This implies that bottom-up pruning (used in decision trees) would not be a good idea in the hinges model. This, together with the underlying idea of getting an effective implementation, makes us opt for a different approach.

It would be preferable to select a knot to prune in such a way that the hinges model changes as less as possible. Further simplifying this consists of pruning the knot which is closest to the straight line connecting the preceding and the following knots. Thus, for a given IHM the *pruning distance* is defined as[4] :

$$d_j = |h_j - (h_{j-1} + (h_{j-1} - h_{j+1})\frac{(k_j - k_{j-1})}{(k_{j-1} - k_{j+1})}|. \qquad (12)$$

To generate the pruning sequence, we first compute the pruning distances (12) for all the internal knots. After that, we remove the knot of minimum distance and update (using (12)) the distances for the previous and following knots to the removed one, the others being unchanged. This process is repeated until we reach the minimum number of knots (i.e. no internal knots). Note that this method implies very simple computations, of computational complexity linear in the number of initial knots, i.e. at worst linear in the number of samples in the learning set.

After generating the sequence of pruned IHMs, we select one of them as the optimum-sized model. To select the best pruned IHM we estimate the MSE of each model of the pruning sequence using a different, independent set of samples (the *pruning set*, $\boldsymbol{PS}$). Notice that these samples are not smoothed by the Supersmoother.

[4]Note that this distance is equivalent to the residual and it is not invariant to rotations, opposite to the Euclidean distance.

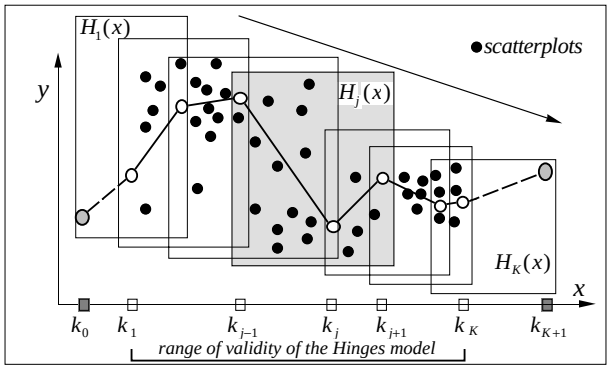

Figure 4: One refitting pass over the hinges.

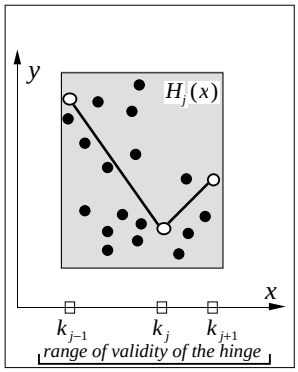

Figure 5: Tuning one hinge.

Then, the best pruned model will be the one minimizing the number of hinges (its complexity) and the estimated MSE. We use the so-called '1 standard error Rule' [4] to deal with errors that can appear in the estimation of the MSE using a finite $\boldsymbol{PS}$. Basically, this rule allows choosing the simplest model whose accuracy is comparable to the minimal MSE, taken into account the uncertainties of our estimate.

3.3 Refitting the initial hinges model

3.3.1 Motivation

In the Introduction we already mentioned the fact that the Supersmoother, while compromising between fidelity to the data (to avoid overfitting) and noise reduction (to avoid oversmoothing), often suffers from high bias. Thus, given the fact that we derived the IHM in order to fit the smoothed scatterplot, this latter will also suffer from bias. However, since the IHM is a parametric model flexible enough to represent the main structure in the data, it should be possible to improve it in terms of accuracy by refitting it to the original scatterplot. Notice that this refitting is possible only for parametric models, whose parameters may be tuned in order to remove the possible errors due to oversmoothing.

However, if the model is too flexible then we would expect refitting to increase variance more strongly than it would reduce bias, with an end result of possible degradation. But the IHM, thanks to the combination of scatterplot smoothing and pruning, catches the correct flexibility of the Hinges model for the problem (scatterplot) at hand. Thus its refitting should not lead to high variance. Furthermore, its parameters being derived from the Supersmoother should provide a good starting point for refitting.

3.3.2 Overall cyclic procedure

The proposed strategy consists in several passes over the Hinges model until all free parameters of (4) are stabilized. Each of these passes (Fig. 4) considers the hinges one by one, moving the central knot to minimize the MSE of the scatterplot data in its range (see Fig. 5).

During the stage described here we (re)fit this (partially frozen) initial model using the complete initial scatterplot to obtain the final result. We call the model obtained using this strategy the *Linear Hinges Model*.

3.3.3 Tuning one hinge

Thus, the core of the proposed (cyclic) strategy is the adjustment of one particular hinge $H_j(x)$ by using the scatterplot that enters into its range of validity $[k_{j-1}, k_{j+1}]$. In this tune we have only two parameters, the coordinates (k_j, h_j) of the central knot. The criterion to carry out this fit is the minimization of the MSE of the hinge in its range of validity, estimated by using the scatterplot points :

$$\begin{aligned} MSE(k_j, h_j) &= n^{-1}(SE_l + SE_r) \\ SE_l &= \textstyle\sum_{k_{j-1} \le x_i \le k_j} [y_i - H_j(x_i)]^2 \\ SE_r &= \textstyle\sum_{k_j < x_i \le k_{j+1}} [y_i - H_j(x_i)]^2 \end{aligned} \tag{13}$$

where n is the total number of scatterplot points in the interval.

Now the problem consists in obtaining the values (k_j^*, h_j^*) minimizing (13). Equations (7) show that (13) is quadratic in h_j. This means that for a given value of k_j, setting the derivative with respect to h_j of the MSE of eqn (13) to zero yields an analytical expression for h_j^*

$$h_j^* = h_j^*(k_j), \tag{14}$$

which can be substituted in (13) (see appendix).

Hence, knot optimization reduces to a one-dimensional search, which could be solved using a bracketing/bisection strategy. Instead of using this approach, we use an enumerative approach evaluating (14) at each distinct observation abscissa value x_i in the support set of the current hinge. We start by evaluating (14) at the first candidate position in the support set and for the following evaluations take advantage of the computations carried out before by using the updating formulas both for the computation of $h_j^*(k_j)$ and $MSE(k_j, h_j^*(k_j))$ given in the appendix.

3.3.4 Bounding the search space

The IHM has the property of being smooth because we have used a set of smoothed data for building it. When we refit this model using the scatterplot this property is not longer insured. It is possible to take care of this effect by restricting the space of search to a region close to the initial position of the hinge. This has the additional advantage of accelerating the refitting process.

Here we use a very simple approach : the user decides the size of the search space $[(k_j + \eta \Delta k_{l,j}), (k_j - \eta \Delta k_{r,j})]$ by

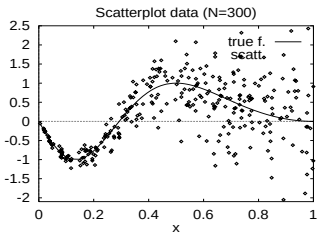

Figure 6: Scatterplot.

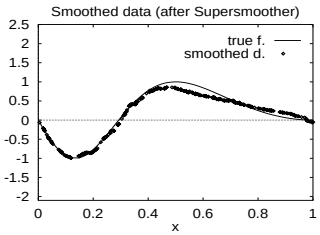

Figure 7: Smoothed data.

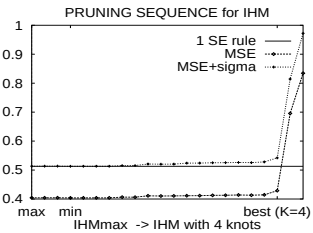

Figure 9: Pruning sequence.

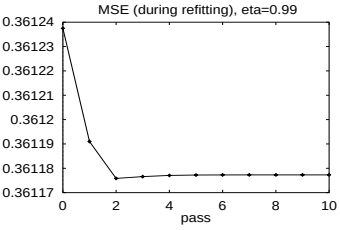

Fig. 10: MSE of LHM (refit).

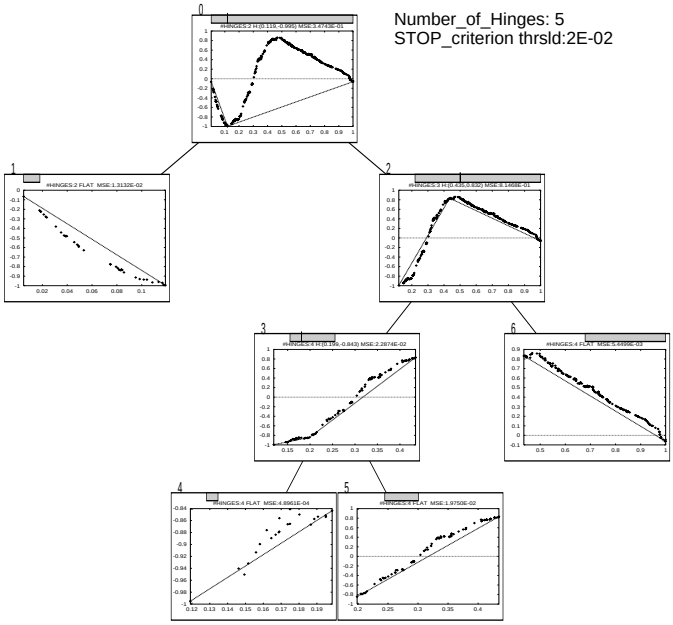

Figure 8: IHM growing tree (first 3 steps).

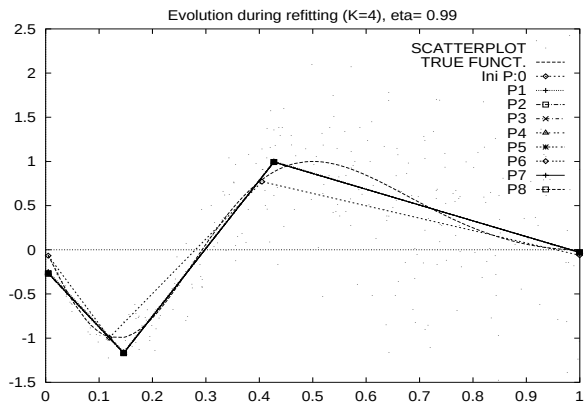

Figure 11: Sequence of LHM models during refitting.

using the parameter η.[5] The parameter η can take values in $[0, 1]$. The first value consists in freezing the abscissa of the hinge whereas the second one corresponds to searching the full range of validity of the hinge.

4 Simulation experiments

This section provides some simulation results using two test problems : a synthetic one and one closer to real world data. See [11] for further comparisions with the parametric Hermite polynomials.

4.1 An illustrative example

To show the proposed approach we use the test problem given in [12]. The scatterplot data consists of N pairs, obtained from $y_i = \sin[2\pi(1 - x_i)^2] + x_i\epsilon_i$, where ϵ_i is i.i.d. $N(0, 1)$. The values of x_i are drawn randomly from the uniform distribution in the interval $[0, 1]$.

Figure 6 shows the scatterplot for $N = 300$ and the underlying true function. We use the Supersmoother to (partially) remove the noise. Figure 7 shows these smoothed data (10). The Supersmoother has been used in its simplest form, i.e. without any bass enhancement (see [16]). Note the over-smoothed result around the two "peaks" of the true function ($x \approx 0.12$ and $x \approx 0.5$). Figure 8 shows successive steps of the IHM_{max} growing method (§3.2). Figure 9 shows the pruning sequence for arriving to the final IHM. The LHM after each refitting pass is shown in Fig. 11, with $\eta = 0.99$. The way this refitting process stabilizes is summarized in Fig. 10. Note that after the third pass the model is stable, and the resulting approximation is significantly better than the Supersmoother results.

If more scatterplots are available, a better result can be obtained. Table 1 shows the MSE of the Supersmoother, the IHM and the LHM when $N = 300$ and $N = 1000$. This error has been evaluated using three different test data: The Learning Set ($\boldsymbol{LS}$) used for building the model, the Scatterplot Test Set ($\boldsymbol{STS}$), an independent set of 1000 data and, the True Test Set ($\boldsymbol{TTS}$) consisting of the values of the true underlying function, evaluated at the 1000 abscissa values of the $\boldsymbol{STS}$. This latter set allows us to measure whether the model has indeed separated the noise from the underlying function. The $\boldsymbol{STS}$ is used for showing the generalization capability. Note that the Supersmoother is better (in terms of MSE) than the IHM, but the LHM is even better. Note also that the errors are very close to the existing ones in the data sets (first and second columns).

4.2 Semi-real application: the OMIB problem

In this section we use a semi-real data set related to the OMIB (one-machine-infinite-bus) power system transient stability test problem used in [19]. The input variable is a linear combination of six attributes representing relevant parameters of the power system. The output is a non-linear transformation of the degree of stability determined by numerical simulations. The "noise" is due to approximation errors stemming from non-linear interactions among the variables defining the linear combination. These data can

[5] The increments are given by (8) and k_j is the position of the hinge before refitting it.

Table 1: MSE of Supersmoother, IHM and LHM

	True function		Supersmoother		IHM		LHM			
	$N = 300$	$N = 1000$	$N = 300$	$N = 1000$	$N = 300$	$N = 1000$	$N = 300$		$N = 1000$	
					$(K = 4)$	$(K = 4)$	$\eta = 0.01$	$\eta = 0.99$	$\eta = 0.01$	$\eta = 0.99$
LS	0.36680	0.34202	0.35906	0.34123	0.38481	0.37727	0.36199	0.36117	0.34229	0.34207
STS	0.36863		0.37458	0.37641	0.40295	0.41538	0.37590	0.37322	0.37682	0.37639
TTS	0		0.01077	0.00768	0.03876	0.04801	0.01117	0.00805	0.00598	0.00558

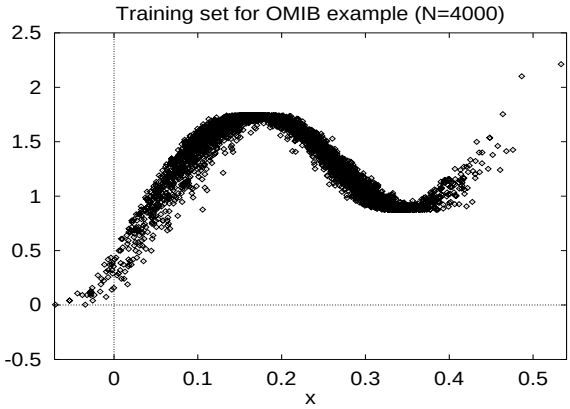

Figure 12: Learning set used in the OMIB example.

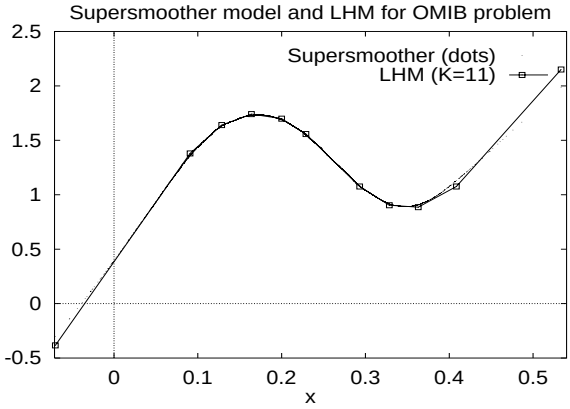

Figure 13: Supersmoother and obtained LHM.

be viewed as result of projecting values of a hypersurface along the direction defined by the linear combination. There are 20,000 sample points, we use 4,000 points for training (2,000 of them as pruning set), (see Fig 12), and the rest 16,000 for testing.

Figure 13 shows the LHM obtained using the pruning approach and $\eta = 0.01$. Note that in this example the noise is small and we have enough learning points to reach good results with the existing methods. Table 2 shows the error for these models, estimated using the learning data ($\boldsymbol{LS}$) and the test data ($\boldsymbol{TS}$). Rough CPU-times to build the models are also shown, measured on a SUN UltraSparc 1 (167 MHz). The CPU time for the LHM is the total one, i.e., the time to obtain the Supersmoother plus the time to get the best IHM plus the time to refit the model. To obtain the LHM ten refitting passes have been made. IHM_{max} corresponds to $K = 54$: during the pruning phase 54 candidate models have been considered and the optimally pruned one IHM_{best} contains only $K = 11$ hinges. Note that the overall CPU time required for growing and pruning the IHM is negligible (16ms) with respect to the time required for the

Table 2: MSE and CPU times : OMIB example

	Supersm.	IHM_{best}	LHM refitting	
		$K = 11$	$\eta = 0.01$	$\eta = 0.99$
LS	0.005376	0.005524	0.005364	0.005369
TS	0.005660	0.005802	0.005655	0.005673
CPU times	100 ms	16ms (total)	181 ms	386 ms

scatterplot smoothing and refitting. Note also that the final LHM has a small number of parameters ($K = 11$) and it is slightly better than the Supersmoother in terms of MSE.

5 Conclusions and future work

We have presented an efficient approach to flexible and robust one-dimensional curve fitting under stringent high noise conditions. It allows combining the properties of nonparametric models (the Supersmoother) with the qualities of the parametric ones (piecewise polynomials). Its effectiveness is corroborated by two examples.

The proposed model can be useful not only for approximating a one-dimensional function, but in complex algorithms where it can be viewed as a small piece that is called repeatedly. For example, it can be used in the context of projection pursuit methods, for discretizing continuous attributes in the context of machine learning, or for summarising a temporal curve, since it is possible to obtain simple descriptors based on the presented models. In the context of fuzzy learning techniques, this method could be used to define membership functions to discretise continuous attributes.

Concerning further work, we should study the possibility of extending the ideas of refitting the model by using a different measure, such as the average sum of residuals, to obtain a (possibly) more robust method. The proposed refitting technique is based on several passes, where each pass always goes from the first hinge to the last one. Other strategies could be tested.

Finally, further investigations should be carried out in order to exploit the linear hinges model in the context of multidimensional regression techniques such as projection pursuit for example.

References

[1] J. H. Friedman. Classification and multiple regression through projection pursuit. Technical Report 12, Stanford University - Department of Statistics, 1985.

[2] J. H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association, Theory and Methods Section*, 76(376), 1981.

[3] P. J. Huber. Projection pursuit. *The Annals of Statistics*, 13(2):435–475, 1985.

[4] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Belmont Wadsworth International, 1984.

[5] J.R. Quinlan. Bagging, boosting, and c4.5. In *Proceedings of AAAI'96 National Conference on Artificial Intelligence*, volume 1, pages 725–730, 1996.

[6] M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.

[7] X. Boyen and L. Wehenkel. Automatic induction of continuous decision trees. In *Proc. of IPMU'96, Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 419–424, Granada, July 1996.

[8] S. Haykin. *Neural networks. A comprehensive foundation*. IEEE Press, 1994.

[9] J. N. Hwang and S. S. You. The cascaded correlation learning: a projection pursuit learning perspective. *IEEE Transactions on Neural Networks*, 7(2):278–289, 1996.

[10] T. Y. Kwok and D. Y. Yeung. Use of bias term in projection pursuit learning improves approximation and convergence properties. *IEEE Transactions on Neural Networks*, 7(5):1168–1183, 1996.

[11] E.F. Sánchez-Úbeda and L. Wehenkel. Smoothing the hinges model: A one-dimensional continuous piecewise cubic polynomial model. To be submitted.

[12] J. H. Friedman and B. W. Silverman. Flexible parsimonious smoothing and additive modeling. *Technometrics*, 31(1), 1989.

[13] J. H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–141, 1991.

[14] J. H. Friedman. Adaptive spline networks. In R.P. Lippman, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 3, pages 675–683. Morgan Kaufman, 1991.

[15] L. Breiman. Current research. In D. H. Wolpert, editor, *The Mathematics of Generalization, Proceedings of the SFI/CNLS Workshop on Formal Approaches to Supervised Learning*. Addison-Wesley, 1995.

[16] J. H. Friedman. A variable span smoother. Technical Report 5, Stanford University - Department of Statistics, 1984.

[17] Y. Zhao and C. G. Atkeson. Implementing projection pursuit learning. *IEEE Transactions on Neural Networks*, 7(2):362–373, 1996.

[18] J. H. Friedman, W. Stuetzle, and A. Schroeder. Projection pursuit density estimation. *Journal of the American Statistical Association, Theory and Methods Section*, 79(387), 1984.

[19] L. Wehenkel. Discretization of continuous attributes for supervised learning. Variance evaluation and variance reduction. In *Proc. of Int. Fuzzy Systems Assoc. World Congress*, volume 1, pages 381–388, Prague, 1997.

[20] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[21] C. de Boor. *A Practical Guide to Splines*, volume 27 of *Applied mathematical sciences*. Springer-Verlag, 1978.

[22] L. Wehenkel. Machine learning for power system security assessment. *IEEE Expert, Intelligent Systems and their Applications*, 12(5):60–72, 1997.

[23] L. Wehenkel. *Automatic learning techniques in power systems*. Kluwer Academic, Boston, 1997.

A Refitting using updating scheme

A.1 Analytical formula for h_j^*

Setting the derivative $\frac{\partial MSE}{\partial h_j}$ of equation (13) to zero we obtain

$$\begin{array}{l} h_j^*(k_j) = \frac{A+B}{C} \\ A = \frac{h_{j-1}}{\Delta k_{l,j}^2}(S_{2,l} + \Delta k_{l,j} S_{1,l}) - \frac{SY_{1,l}}{\Delta k_{l,j}} \\ B = \frac{h_{j+1}}{\Delta k_{r,j}^2}(S_{2,r} - \Delta k_{r,j} S_{1,r}) + \frac{SY_{1,r}}{\Delta k_{r,j}} \\ C = \frac{S_{2,l}}{\Delta k_{l,j}^2} + \frac{S_{2,r}}{\Delta k_{r,j}^2} \\ SY_{1,l} = \sum_{k_{j-1} \le x_i \le k_j} y_i (x_i - k_{j-1}) \\ SY_{1,r} = \sum_{k_j < x_i \le k_{j+1}} y_i (x_i - k_{j+1}) \\ S_{u,l} = \sum_{k_{j-1} \le x_i \le k_j} (x_i - k_{j-1})^u \\ S_{u,r} = \sum_{k_j < x_i \le k_{j+1}} (x_i - k_{j+1})^u \\ u = 1, 2 \end{array} \tag{15}$$

A.2 Updating scheme

A.2.1 Updating scheme of eqn. (15)

We start by evaluating first eqn. (15) setting $k_j = x_1$, where x_1 denotes the first data point in the current interval. For the subsequent candidate positions we use the following updating scheme :

$$\begin{array}{l} SY_{1,l}^{new} = SY_{1,l}^{prev} + y_{new}(x_{new} - k_{j-1}) \\ SY_{1,r}^{new} = SY_{1,r}^{prev} - y_{new}(x_{new} - k_{j+1}) \\ S_{u,l}^{new} = S_{u,l}^{prev} + (x_{new} - k_{j-1})^u \\ S_{u,r}^{new} = S_{u,r}^{prev} - (x_{new} - k_{j+1})^u \\ u = 1, 2 \end{array} \tag{16}$$

where (x_{new}, y_{new}) are the coordinates of the scatterplot point which goes from right to left.

A.2.2 Updating the value of MSE

Having computed the optimal value of h_{new}^* we can in principle recompute the MSE (13) for the present knot position (x_{new}, h_{new}^*). The updating scheme is obtained as follows :

$$\begin{array}{l} SE_l = SYY_{0,l} - 2h_{j-1} SY_{0,l} - 2\frac{\Delta h_{l,j}}{\Delta k_{l,j}} SY_{1,l} \\ \qquad +2h_{j-1}\frac{\Delta h_{l,j}}{\Delta k_{l,j}} S_{1,l} + \frac{\Delta h_{l,j}^2}{\Delta k_{l,j}^2} S_{2,l} + n_l h_{j-1}^2 \\ SE_r = SYY_{0,r} - 2h_{j+1} SY_{0,r} - 2\frac{\Delta h_{r,j}}{\Delta k_{r,j}} SY_{1,r} \\ \qquad +2h_{j+1}\frac{\Delta h_{r,j}}{\Delta k_{r,j}} S_{1,r} + \frac{\Delta h_{r,j}^2}{\Delta k_{r,j}^2} S_{2,r} + n_r h_{j+1}^2 \\ SYY_{0,l} = \sum_{k_{j-1} \le x_i \le k_j} y_i^2 \\ SY_{0,l} = \sum_{k_{j-1} \le x_i \le k_j} y_i \\ SYY_{0,r} = \sum_{k_j < x_i \le k_{j+1}} y_i^2 \\ SY_{0,r} = \sum_{k_j < x_i \le k_{j+1}} y_i \end{array} \tag{17}$$

where n_l and n_r denote the number of scatterplot points in the left and right subintervals.

These formulas lend themselves to the following updating scheme :

$$\begin{array}{l} SYY_{0,l}^{new} = SYY_{0,l}^{prev} + y_{new}^2 \\ SY_{0,l}^{new} = SY_{0,l}^{prev} + y_{new} \\ SYY_{0,r}^{new} = SYY_{0,r}^{prev} - y_{new}^2 \\ SY_{0,r}^{new} = SY_{0,r}^{prev} - y_{new} \\ n_l^{new} = n_l^{prev} + 1 \\ n_r^{new} = n_r^{prev} - 1 \end{array} \tag{18}$$