On the Hinge Finding Algorithm for Hinging Hyperplanes – Revised Version

P. Pucar
Department of Electrical Engineering
Linköping University
S-581 83 Sweden
Email: predrag@isy.liu.se

J. Sjöberg
Department of Electrical Engineering
Linköping University
S-581 83 Sweden
Email: sjoberg@isy.liu.se

Revised version, submitted to IEEE Trans. on Info. Theory.

Abstract

This paper concerns the estimation algorithm for hinging hyperplane (HH) models, a non-linear black box model structure suggested in [3]. The estimation algorithm is analysed and it is shown that it is a special case of a Newton algorithm applied on a quadratic criterion. This insight is then used to suggest possible improvements of the algorithm so that convergence can be guaranteed.

In addition the way of updating the parameters in the HH model, is discussed. In [3] a stepwise updating procedure is proposed. In this paper we stress that simultaneous updating of the model parameters can be preferable in some cases.

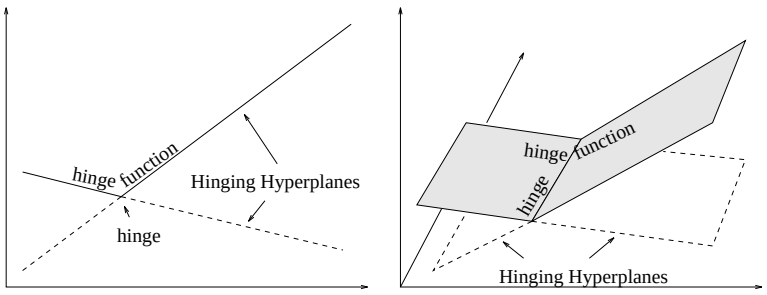
Key words: Nonlinear function approximation, hyperplanes, numerical methods.

1 Introduction

There has been a large activity during the past years in the field of non-linear function approximation. Many interesting results have been reported in connection with, for example the *projection pursuit regression* in [5], neural network approach, see [7] and references therein, and the recent wavelets approach, see [2]. The first two methods are closely related to the *hinging hyperplane* (HH) model investigated here. All the different approaches can be described as basis function expansions

$$f(\mathbf{x}) = \sum_{i=1}^{K} h_i(\mathbf{x}) \tag{1}$$

and they differ only in the choice of basis $h_i(\mathbf{x})$. One important difference between the basis functions used in HH models, projection pursuit and NN models as opposed to the basis function used in the wavelet approach, is that the first three mentioned basis functions have their non-linearity positioned across certain directions. In other directions the function is constant. A name for this kind of functions is *ridge functions*.



(a) 1-Dimensional (b) 2-Dimensional

Figure 1: Hinge, hinging hyperplane and hinge function.

The wavelet basis is a localized one. If data is clustered along subspaces it can be preferable to use one of the ridge basis functions. In the NN approach the basis function is the *sigmoidal function*.

Recently a new interesting approach to non-linear function approximation named *hinging hyperplanes*, was reported [3]. The HH approach uses *hinge functions* as basis functions in the expansion (1). A hinge function is maybe most easily illustrated by a figure, see Figure 1.

Assume that the two hyperplanes are given by

$$h^+ = \mathbf{x}^T\theta^+, \quad h^- = \mathbf{x}^T\theta^- \tag{2}$$

where $\mathbf{x} = [1, x_1, x_2, \cdots, x_m]^T$, is the regressor vector and θ^+ and θ^- are the parameter vectors defining the hyperplanes. These two hyperplanes are joined together at $\{\mathbf{x} : \mathbf{x}^T(\theta^+ - \theta^-) = 0\}$. The joint, $\Delta = \theta^+ - \theta^-$, or multiples of Δ, are defined as the *hinge* for the two hyperplanes h^+ and h^-. The solid/shaded part of the two hyperplanes as in Figure 1, is explicitly given by

$$h = \max(h^+, h^-) \quad \text{or} \quad h = \min(h^+, h^-)$$

and are defined as the hinge function. Which combination of hyperplanes that is chosen, *i.e.*, whether the min or max function is used, is given when the parameters θ^+ and θ^- are estimated. In Section 2, a more detailed review of the estimation algorithm for the HH model presented in [3], is given.

In this contribution two issues will be penetrated. One is the *hinge finding algorithm* (HFA) as it is presented in [3]. It will here be shown that the HFA actually is a Newton algorithm for function minimization applied on a quadratic loss function, and suggestions on how to improve the HFA will be given so that convergence can be guaranteed. The original HFA, depending on the function approximated, can behave in three ways: 1) the algorithm converges and a hinge location is found, 2) the algorithm is stuck in a limit cycle altering the hinge location between a series of different values, and 3) the HFA will not converge at all and the hinge is located outside the data support. The improvement is straightforward when realizing what family of numerical algorithms the HFA actually belongs to. The improvement will guarantee global convergence of the algorithm which means that the algorithm converges to a *local* minimizer of a non-linear functional regardless of the initial parameter guess. This is in the spirit of [4].

The second issue is the way additional basis functions, *i.e.*, hinge functions are introduced into the HH model. In [3] the hinges are introduced one after the other and the parameters of the already introduced hinges are fitted before the next one is introduced. The fitting of the parameters after a new hinge function has been incorporated is also performed in an iterative way. One step is taken with the HFA for each hinge function. This approach will be discussed and compared to other possible estimation algorithms.

In the original presentation [3], it is advocated for HH models as a superior alternative to NN models. One of the main argument is the efficient estimation algorithm for HH models. It will be shown that the same algorithms are applicable for both model structures. It follows from this that the choice of model structure, *i.e.*, HH model or NN model should not be made based on algorithmic reasons but rather on assumptions on the unknown relationship which is to be modeled.

The paper is organized in the following way. In Section 2 the HFA and the strategy for updating and adding hinge functions is reviewed. In Section 3 the novel insights and improvements based on these are presented. The estimation algorithm when the HH model consists of several hinge functions is discussed in Section 4. Finally in Section 5 some comparisons of performance of the different algorithms are given.

2 Hinging Hyperplanes Function Approximation

The general goal is to find a model $f(\cdot)$ which approximates an unknown function $g(\mathbf{x})$ as good as possible. To fit the parameters we have data available $\{y_i, \mathbf{x}_i\}_{i=1}^N$, where y_i are (noisy) measurments of $g(\mathbf{x}_i)$.

The choice of non-linear black box model $f(\cdot)$ for a particular problem is an important issue. A model where the basis functions manage to describe the data in an efficient way can be expected to have good properties and, hence, is to be prefered. This is, however, the kind of prior knowledge which is rather exceptional. Instead the choice of a specific black box model structure is usually guided by other arguments.

The main advantages of the new HH approach are:

- An upper bound on the approximation error, is available.
- The estimation algorithm used in the HH algorithm is a number of least-squares algorithms which can be executed fast and in a computationally efficient way.
- It *may* be an useful model structure since what is obtained by HH approximation is piecewise linear models, and linear models have proven to be useful in a large number of problems.

In [3] the upper bound on the approximation error for HH models is stated. Assume that a sufficiently smooth function $g(\mathbf{x})$ is given, where sufficiently smooth means that the following integral is finite

$$\int ||\omega||^2 \hat{g}(\omega) d\omega = c < \infty,$$

then there are hinge functions $h_1, \ldots, h_K$ such that

$$||g - \sum_{i=1}^{K} h_i||_2 \leq \frac{(2R)^2 c}{K^{\frac{1}{2}}},$$

where R is the radius of the sphere within which we want to approximate the function, c is defined above and the $\hat{g}(\omega)$ is the Fourier transform of $g(\mathbf{x})$. The proof of the theorem is an extension of Barron's result for sigmoidal neural networks given in [1]. This means that the HH model is as efficient as neural networks for the L_2-norm. This should be compared to the best achievable convergence rate for any linear estimator for functions in the class

$$\int_{\mathcal{R}^m} ||\omega|| \hat{g}(\omega) d\omega < \infty.$$

The lower rate for linear estimators is approximately $K^{-1/m}$. This indicates that the largest gain using NN or HH models is obtained when the dimension of the input space is high.

2.1 Hinge Finding Algorithm

In the estimation algorithm, proposed in [3], used for estimating HH models there is one "subroutine" that is often called, namely the hinge finding algorithm. Here the HFA is reviewed. As stated above the hinge is the subspace of the input space that satisfies the equation $\mathbf{x}^T \Delta = 0$, where $\Delta = \theta^+ - \theta^-$. Given a data set $\{y, \mathbf{x}_i\}$, the HFA consists of the following steps:

1. Choose an initial split of the data, or in other words, choose the initial hinge. Name the two sets of data S_+ and S_-.

2. Calculate the least-squares coefficients of a hyperplane fitted to the values of $\{y, \mathbf{x}\}$ in S_+ and denote the parameters by θ^+. Do the analogous with the $\{y, \mathbf{x}\}$'s in S_- to obtain θ^-.

3. Update S_+ and S_- by finding the new data sets according to the expressions $S_+ = \{\mathbf{x} : \mathbf{x}^T \cdot (\theta^+ - \theta^-) > 0\}$ and $S_- = \{\mathbf{x} : \mathbf{x}^T \cdot (\theta^+ - \theta^-) \leq 0\}$.

4. Go to 2 until the hinge function has converged.

The HFA is illustrated by Figure 2. The function we want to approximate is $g(x) = x^2$, and in Figure 2 we use samples from that function paired with the x-values as the input to the HFA algorithm. For later use we state the second least-square step of the algorithm

$$\begin{aligned} \theta^+ &= (\sum_{\mathbf{x}_i \in S_+} \mathbf{x}_i \mathbf{x}_i^T)^{-1} \sum_{\mathbf{x}_i \in S_+} \mathbf{x}_i y_i \\ \theta^- &= (\sum_{\mathbf{x}_i \in S_-} \mathbf{x}_i \mathbf{x}_i^T)^{-1} \sum_{\mathbf{x}_i \in S_-} \mathbf{x}_i y_i. \end{aligned} \tag{3}$$

As mentioned above, the HH models are preferably used when the dimension of the input space is high. Some examples in this paper are, however, of low dimension purely for the sake

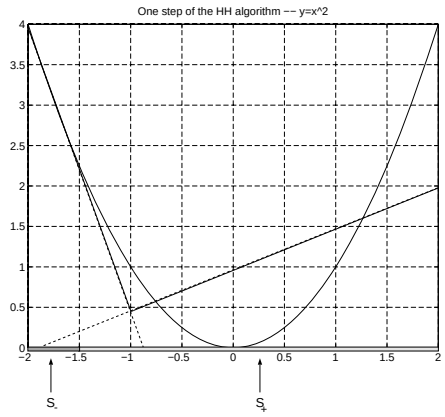


Figure 2: The initial split of data was at $x = -1.5$. The least-squares estimates in S_- and S_+ are the two lines. Their intersection $x = -1$ is the new hinge position which gives the new split of the data for the next step of HFA.

of clarity of the presentation. From Figure 2 it is obvious how the hinge function should be chosen (recall the min vs. max discussion). Choosing the minimum of the two hyperplanes as a hinge function would have the consequence of using the data in S_+ to calculate the approximating hyperplane in S_-.

If the unknown function $g(\mathbf{x})$ itself is a hinge function then it can be shown that the HFA will converge towards the true hinge location. If $g(\mathbf{x})$ is an arbitrary function, there are three different ways the HFA can take, as mentioned in Section 1. In practical applications with real data involved this unpredictable behavior of the HFA causes problems. Let us look at the following example for some further insights into the problems associated with hinge search. Consider the function given in Figure 3. If Breiman's HFA is applied to this data set, the resulting hinge position will vary dramatically for different initial values. The evolution of the hinge position with different initial conditions is depicted in Figure 4, where the y-axis denotes the initial hinge position, and the x-axis represents the number of iterations of the HFA. The empty parts of the y-axis, where it seems that no initial hinge positions have been tested, are the initial values that will cause the hinge to go outside the border of the support. In this case one of the sets S_+ and S_- contains all data and the other one is empty. If this happens the algorithm stops since step 2 cannot be performed and the obtained function is linear in the domain of the data support. From Figure 4 it can be concluded that for this particular function as shown in Figure 3, if Breiman's HFA is used, there would be two convergence points, and one limit cycle. There is also an interval from 0.25 to 0.325 which, if taken as the initial hinge positions, will lead to no converged hinge at all. That is, if the HFA is initialized in that region, the hinge would end up at a position outside the support area.

As a summary, Breiman's algorithm cannot guarantee convergence, and depending on the problem and the initial parameter value it might even diverge. Usually covergence can be assured by modifying the parameter steps so that a criterion is decreaseed in each step.

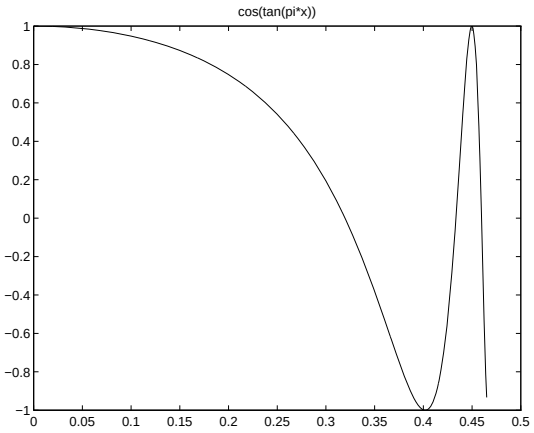


Figure 3: Function $\cos(\tan(\pi \cdot x))$ with $x = [0, 0.46]$

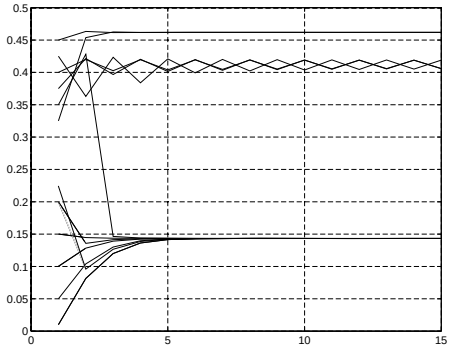


Figure 4: Convergence of the HFA for different initial values. On the x-axis the number of iterations is shown, and the y-axis is the initial hinge position. The empty intervals on the y-axis correspond to the initial values that do not converge.

Breiman's algorithm, however, does not use any criterion so this modification is not straightforward. In Section 3 we will show how the algorithm should be modified.

2.2 HH Algorithm

Essentially the HH algorithm is a strategy to stepwise increase the number of hinge functions in the model by using the HFA. The procedure is as follows. Given $\{y_i, \mathbf{x}_i\}$, run the HFA on the available data, estimating the first hinge function. To introduce an additional hinge function calculate the difference between the given data and the estimated hinge (the residuals) $\tilde{y}_{[1]} = y - h_1$ and run the HFA on $\tilde{y}_{[1]}$ obtaining h_2.

Now, run the HFA on $\tilde{y}_{[2]} = y - h_2$ and reestimate h_1. Iterate between the reestimation of h_1 and h_2 until the procedure has converged. If a third hinge function is added the procedure is analogous, first calculate $\tilde{y}_{[1,2]} = y - h_1 - h_2$ and run the HFA to obtain h_3, and then reiterate with the HFA on $\tilde{y}_{[2,3]}$, $\tilde{y}_{[1,3]}$ and $\tilde{y}_{[1,2]}$.

In [3] the advice is to just run one step of the HFA in each iteration after introducing hinge function number two. It is not clear whether this sequential updating of the hinge function parameters is the best one and a number of variants are immediately apparent, *e.g.*, could all the hinge function parameters be updated simultaneously, or could a more efficient way to update the parameters after introducing an additional hinge function be just to simply start all over again re-initializing all the parameters? Also, as the HFA may not converge at all, it is clear that the HH algorithm in its original shape, is not a reliable algorithm. This will be further discussed in Section 3 and 4.

3 Globally Convergent HFA

In this section the course taken will be quite different from the one taken when deriving the original HFA. However, the resulting scheme is the same, and the alternative derivation will place the algorithm in a broader context of numerical algorithms and will give some hints on how the algorithm can be improved.

This approach uses a gradient based search for the minimum of a quadratic criterion. To differentiate a function which consists of a sum of highly non-linear `max` and `min` elements might rise some worries. This is, however, not a problem. With respect to the *parameters* the *criterion* is smooth and the gradient and the Hessian both exist for all values of the parameters.

Assume that a data set $\{y_i, \mathbf{x}_i\}_{i=1}^N$ is given and the objective is to fit a hinge function using the given set of data. This is always the problem definition when the HFA is considered and in the general HH algorithm y is iteratively replaced by $\tilde{y}_{[\cdot]}$ in sequences. The HFA remains the same regardless of the present choice of y, when it is "called" from the HH algorithm. The input to HFA is always a data set of the form as above.

Let us formulate the objective in the following way. Given the criterion of fit

$$V_N(\theta) = \frac{1}{2} \sum_{i=1}^{N} (y_i - h(\mathbf{x}_i, \theta))^2, \tag{4}$$

calculate the parameter θ that minimizes it. Formally it can be expressed as

$$\hat{\theta} = \arg\min_{\theta} V_N(\theta)$$

where θ is a vector that can be written as

$$\theta = \begin{pmatrix} \theta^+ \\ \theta^- \end{pmatrix}.$$

Recall that the function h is defined as

$$h(\mathbf{x}, \theta) = \max\,(\text{or } \min)\{h^+,\, h^-\} \tag{5}$$

and S_+ and S_- are defined as those half-spaces where the first, respectively the second argument of (5) holds.

As we will use gradient based methods we need the derivative of the hinge function with respect to the parameters. The derivative with respect to θ^+ becomes (with the analogue expression for the derivative with respect to θ^-)

$$\frac{dh(\mathbf{x}, \theta)}{d\theta^+} = \begin{cases} \mathbf{x} & \text{if } \mathbf{x} \in S_+ \\ 0 & \text{if } \mathbf{x} \in S_- \end{cases} \tag{6}$$

So the derivative is just $\mathbf{x}$, as in the linear regression case, if $\mathbf{x} \in S_+$ and zero otherwise. Possible data points on the hinges are not any problem in the algorithm, since the hinges have measure zero in the space $\mathcal{R}^m$ (recall that $x \in \mathcal{R}^m$), there will be no points at the hinge in the generic case. To have a totally well defined problem one can let the hinge belong to one of the two sets, which is the solution adopted in Breiman's paper. Another possibility is to define the derivative as, *e.g.*, zero at the hinge, which means that any data at the hinge is excluded from the fit.

To compute the minimum of $V_N(\theta)$ with a standard Newton procedure, the gradient and the Hessian of the criterion V_N, is needed. As for the derivative of the hinge function we separate the parameter vector into θ^+ and θ^-.

$$\nabla V_N = \begin{pmatrix} \frac{\partial V_N}{\partial \theta^+} \\ \frac{\partial V_N}{\partial \theta^-} \end{pmatrix} = \begin{pmatrix} -\sum_{i=1}^N \frac{dh(\mathbf{x}_i,\theta)}{d\theta^+}(y_i - h(\mathbf{x}_i, \theta)) \\ -\sum_{i=1}^N \frac{dh(\mathbf{x}_i,\theta)}{d\theta^+}(y_i - h(\mathbf{x}_i, \theta)) \end{pmatrix}$$

$$= \begin{pmatrix} -\sum_{\mathbf{x}_i \in S_+} \mathbf{x}_i(y_i - \mathbf{x}_i^T\theta^+)) \\ -\sum_{\mathbf{x}_i \in S_-} \mathbf{x}_i(y_i - \mathbf{x}_i^T\theta^-)) \end{pmatrix} = \begin{pmatrix} -\sum_{\mathbf{x}_i \in S_+} \mathbf{x}_i(y_i - \mathbf{x}_i^T\theta^+)) \\ -\sum_{\mathbf{x}_i \in S_-} \mathbf{x}_i(y_i - \mathbf{x}_i^T\theta^-)) \end{pmatrix}.$$

The expression of the derivative is the same as in the linear regression case with the modification that only the data in the correct half-plane are included.

The Hessian is obtained by differentiating V_N once again

$$\nabla^2 V_N = \begin{pmatrix} \frac{\partial(-\sum_{\mathbf{x}_i \in S_+} \mathbf{x}_i(y_i - \mathbf{x}_i^T\theta^+)))}{\partial\theta^+} & \frac{\partial(-\sum_{\mathbf{x}_i \in S_+} \mathbf{x}_i(y_i - \mathbf{x}_i^T\theta^+)))}{\partial\theta^-} \\ \frac{\partial(-\sum_{\mathbf{x}_i \in S_-} \mathbf{x}_i(y_i - \mathbf{x}_i^T\theta^-)))}{\partial\theta^+} & \frac{\partial(-\sum_{\mathbf{x}_i \in S_-} \mathbf{x}_i(y_i - \mathbf{x}_i^T\theta^-)))}{\partial\theta^-} \end{pmatrix}.$$

The off diagonal elements are equal to zero since the intersection of S^+ and S^- is zero. The derivative of the expressions in the diagonal is straightforward since the hinge function is linear in the region over which the summation is performed. The result is thus

$$\nabla^2 V_N = \begin{pmatrix} \sum_{\mathbf{x}_i \in S_+} \mathbf{x}_i\mathbf{x}_i^T & \mathbf{0} \\ \mathbf{0} & \sum_{\mathbf{x}_i \in S_-} \mathbf{x}_i\mathbf{x}_i^T \end{pmatrix}. \tag{7}$$

We can now apply the Newton algorithm to find the minimum of (4), see [4]. This means that we have the following iterative search algorithm.

$$\begin{aligned} \theta_{k+1} &= \theta_k - (\nabla^2 V_N)^{-1}\nabla V_N \\ &= \theta_k + \begin{pmatrix} \sum_{\mathbf{x}_i \in S_+} \mathbf{x}_i\mathbf{x}_i^T & \mathbf{0} \\ \mathbf{0} & \sum_{\mathbf{x}_i \in S_-} \mathbf{x}_i\mathbf{x}_i^T \end{pmatrix}^{-1} \begin{pmatrix} \sum_{\mathbf{x}_i \in S_+} \mathbf{x}_i(y_i - \mathbf{x}_i^T\theta_k^+) \\ \sum_{\mathbf{x}_i \in S_-} \mathbf{x}_i(y_i - \mathbf{x}_i^T\theta_k^-) \end{pmatrix} \\ &= \theta_k + \begin{pmatrix} (\sum_{\mathbf{x}_i \in S_+} \mathbf{x}_i\mathbf{x}_i^T)^{-1} \sum_{\mathbf{x}_i \in S_+} \mathbf{x}_i(y_i - \mathbf{x}_i^T\theta_k^+) \\ (\sum_{\mathbf{x}_i \in S_-} \mathbf{x}_i\mathbf{x}_i^T)^{-1} \sum_{\mathbf{x}_i \in S_-} \mathbf{x}_i(y_i - \mathbf{x}_i^T\theta_k^-) \end{pmatrix} \\ &= \theta_k + \begin{pmatrix} (\sum_{\mathbf{x}_i \in S_+} \mathbf{x}_i\mathbf{x}_i^T)^{-1} \sum_{\mathbf{x}_i \in S_+} \mathbf{x}_i y_i - \theta_k^+ \\ (\sum_{\mathbf{x}_i \in S_-} \mathbf{x}_i\mathbf{x}_i^T)^{-1} \sum_{\mathbf{x}_i \in S_-} \mathbf{x}_i y_i - \theta_k^- \end{pmatrix}. \end{aligned} \tag{8}$$

In the last expression for the Newton step the rule for calculation of the next θ from step 2 of the HH algorithm (3) is recognized. If it is rewritten we obtain the expression

$$\theta_{k+1} = \theta_k + (\theta_{k+1}^{\mathrm{Br}} - \theta_k).$$

where θ^{Br} is the parameter which would have been obtained if the HH algorithm was used. The conclusion is that using a Newton algorithm for minimization of (4) is equivalent to using the HFA. Generally, Newton's method is not globally convergent, since no precaution is taken regarding the decrease of the loss function. One of the conventional solutions to the convergence problem of Newton's method is to include a line search. The modified algorithm is the damped Newton algorithm. The damped Newton algorithm will in our case give the following parameter update recursion

$$\theta_{k+1} = \theta_k + \mu(\theta_{k+1}^{\mathrm{Br}} - \theta_k).$$

The strategy for choosing μ is to first try a full Newton step, *i.e.*, $\mu = 1$, and if that fails to decrease the loss function, a sequence of decreasing μ's, *e.g.* $\mu = \{\frac{1}{2}, \frac{1}{4}, \ldots\}$ will be tried.

In [4] other strategies for the decrease of μ are suggested where the function evaluations that are performed when new μ's are tested, are used for building local higher order models of the cost function. These higher order models are used as base for calculation of μ's to test. However, for clarity in the examples given in this paper, the simplest possible strategy is used. It is straightforward to include more sophisticated algorithms.

Let us end this section by stating some insights:

- To assure convergence, the HFA suggested in [3] should be modified with a step length. This necessity is exemplified in Section 5.
- One single parameter update, (3) or (8), means that we solve a least-squares problem. There is a non-linear effect due to that the subspaces S_+ and S_- change together with the parameters. Caused by this change $\mathbf{x}_i^T\theta^+$ will not apply to exactly the same data as θ^+ was estimated on. The step length is introduced to limit this non-linear effect and to prevent a too large change of the subspaces S_+ and S_- in one single iteration.

4 Simultaneous Estimation of Hinge Function Parameters

In the previous section it was concluded that the HFA is equivalent to Newton's algorithm for minimization of a quadratic criterion. However, only parameters associated to one hinge function are changed, and even when the model consists of many hinge functions the HFA algorithm considers only one of them at the time. An alternative is to apply a damped Newton method to all parameters at the same time, which would give a simultaneous parameter update. In this section we discuss possible advantageous with this approach.

First we calculate the gradient and the Hessian of the criterion. Consider a HH model with K hinge functions

$$f(\mathbf{x}) = \sum_{i=1}^{K} h_i(\mathbf{x}) \tag{9}$$

where $h_i(\mathbf{x})$ are the hinge functions of the form (2). Let the parameters be organized in one column

$$\theta = \begin{bmatrix} \theta_+^1 \\ \theta_-^1 \\ \vdots \\ \theta_+^K \\ \theta_-^K \end{bmatrix}$$

where the index shows to which hinge function the parameter vector belongs.

Using (6) the gradient of the criterion (4) becomes

$$\nabla V = \begin{bmatrix} -\sum_{\mathbf{x}_i \in S_+^1} \mathbf{x}_i (y_i - f(\mathbf{x}_i)) \\ -\sum_{\mathbf{x}_i \in S_-^1} \mathbf{x}_i (y_i - f(\mathbf{x}_i)) \\ \vdots \\ -\sum_{\mathbf{x}_i \in S_+^K} \mathbf{x}_i (y_i - f(\mathbf{x}_i)) \\ -\sum_{\mathbf{x}_i \in S_-^K} \mathbf{x}_i (y_i - f(\mathbf{x}_i)) \end{bmatrix}. \tag{10}$$

where we skipped the index N indicating the number of data. Notice that the blocks only differ from each other by the terms in the sums. Each sum includes the data of a half-space.

Differentiating the gradient once more gives the Hessian

$$\nabla^2 V = \begin{bmatrix} \nabla^2 V_{11} & \nabla^2 V_{12} & \dots & \nabla^2 V_{1K} \\ \nabla^2 V_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \nabla^2 V_{K1} & \dots & \dots & \nabla^2 V_{KK} \end{bmatrix}, \tag{11}$$

$$\nabla^2 V_{ij} = \begin{pmatrix} \sum_{\mathbf{x}_k \in S_+^i \cap S_+^j} \mathbf{x}_k \mathbf{x}_k^T & \sum_{\mathbf{x}_k \in S_+^i \cap S_-^j} \mathbf{x}_k \mathbf{x}_k^T \\ \sum_{\mathbf{x}_k \in S_-^i \cap S_+^j} \mathbf{x}_k \mathbf{x}_k^T & \sum_{\mathbf{x}_k \in S_-^i \cap S_-^j} \mathbf{x}_k \mathbf{x}_k^T \end{pmatrix}.$$

Each component looks exactly as in the linear regression case with that modification that only those data which belong to the intersection of two half-spaces are included.

The diagonal blocks look like (7), *i.e.*,

$$\nabla^2 V_{ii} = \begin{pmatrix} \sum_{\mathbf{x}_i \in S_+^i} \mathbf{x}_i \mathbf{x}_i^T & \mathbf{0} \\ \mathbf{0} & \sum_{\mathbf{x}_i \in S_-^i} \mathbf{x}_i \mathbf{x}_i^T \end{pmatrix}.$$

and have zero off-diagonal terms since the half-spaces S_+^i and S_-^i have no intersection by definition.

An example will be used to illustrate the calculations. Assume that the hinging hyperplane model consists of two hinge functions. From (10) the gradient can be expressed as

$$\nabla V = \begin{bmatrix} -\sum_{\mathbf{x}_i \in S_+^1} \mathbf{x}_i (y_i - \mathbf{x}_i^T \theta_+^1) - \sum_{\mathbf{x}_i \in S_+^1 \cap S_+^2} \mathbf{x}_i (y_i - \mathbf{x}_i^T \theta_+^2) - \sum_{\mathbf{x}_i \in S_+^1 \cap S_-^2} \mathbf{x}_i (y_i - \mathbf{x}_i^T \theta_-^2) \\ -\sum_{\mathbf{x}_i \in S_-^1} \mathbf{x}_i (y_i - \mathbf{x}_i^T \theta_-^1) - \sum_{\mathbf{x}_i \in S_-^1 \cap S_+^2} \mathbf{x}_i (y_i - \mathbf{x}_i^T \theta_+^2) - \sum_{\mathbf{x}_i \in S_-^1 \cap S_-^2} \mathbf{x}_i (y_i - \mathbf{x}_i^T \theta_-^2) \\ -\sum_{\mathbf{x}_i \in S_+^2} \mathbf{x}_i (y_i - \mathbf{x}_i^T \theta_+^2) - \sum_{\mathbf{x}_i \in S_+^2 \cap S_+^1} \mathbf{x}_i (y_i - \mathbf{x}_i^T \theta_+^1) - \sum_{\mathbf{x}_i \in S_+^2 \cap S_-^1} \mathbf{x}_i (y_i - \mathbf{x}_i^T \theta_-^1) \\ -\sum_{\mathbf{x}_i \in S_-^2} \mathbf{x}_i (y_i - \mathbf{x}_i^T \theta_-^2) - \sum_{\mathbf{x}_i \in S_-^2 \cap S_+^1} \mathbf{x}_i (y_i - \mathbf{x}_i^T \theta_+^1) - \sum_{\mathbf{x}_i \in S_-^2 \cap S_-^1} \mathbf{x}_i (y_i - \mathbf{x}_i^T \theta_-^1) \end{bmatrix}.$$

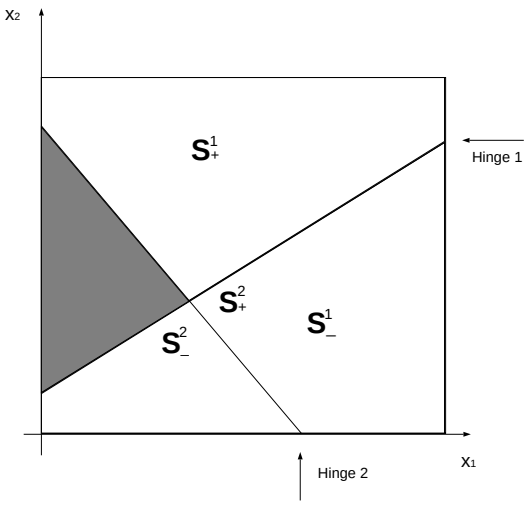


Figure 5: Example of summation areas for two hinge functions. The shaded area $S_+^1 \cap S_-^2$ is the part of hinging hyperplane model that is influenced both by θ_+^1 and θ_-^2.

The case above is illustrated in Figure 5, where a two-dimensional example is given, and the lines represent the partition of the space into the half-spaces S_+ and S_-. When the gradient is differentiated, to obtain the second derivative, the off diagonal blocks will contain terms of the type $\sum \mathbf{x}_i \mathbf{x}_i^T$, where the summation index goes over intersections of two hyperplanes belonging to different hinge functions. At first sight the calculation of the second derivative might look messy. However, using a software package that utilizes vector and matrix multiplication, this kind of operation is performed in one step.

Having obtained both the gradient and the second derivative all components for applying a Newton type algorithm are available, *e.g.*, the parameters can be updated according to

$$\theta_{k+1} = \theta_k - \mu(\nabla^2 V)^{-1} \nabla V. \tag{12}$$

Remark 1: When the damped Newton method is implemented one avoids the computational demanding computation of the inverse of the Hessian. Instead one solves a system of linear equations

$$\nabla^2 V \Delta\theta_{k+1} = -\nabla V$$

where the parameter update $\Delta\theta_{k+1} = \theta_{k+1} - \theta_k$ is the unknown. This can be done very fast in, *e.g.*, MATLAB.

Remark 2: It can be shown that the original description of the HH model with the max/min basis functions is over-parameterized. This means that one has to use the pseudo-inverse in (12). Alternatively, by changing the parameterization a more sparse description with less parameters can be obtained. See [6].

When can we expect to obtain a better performance with a simultaneous update like (12) than with the HH algorithm? There is no clear answer to this question. The Newton

algorithm corresponds to a second order Taylor expansion of the criterion. If this is a good approximation of the criterion then we also can expect the Newton step to be good.

Using the HFA algorithm implies that the off-diagonal elements in the Hessian (11) are not considered. This makes each iteration faster but must typically be compensated by some additional iterations. If the criterion is close to quadratic and if the off-diagonal elements are of importance, then this will be a disadvantage. Typically the quadratic expansion is a good approximation close to the minimum and if the criterion has a narrow valley in the parameter space then we can expect that neglecting the off-diagonal elements slows down the process considerably. See [4]

Far away from the minimum, *e.g.*, at the beginning of the search, the quadratic expansion might not be applicable and then it might be advantageous to neglect the off-diagonal elements.

In the introduction it was mentioned that the HH model can be viewed as a basis expansion. A function expansion where the basis functions are orthonormal all parameters can be estimated independent from each other, *i.e.*, all off-diagonal elements of the Hessian are zero. For the HH model, however, the basis functions overlap and the importance of this overlap depends problem. It depends not only on the data but also on the current parameters θ_k.

The simultaneous update becomes more computational expansive when the number of parameters increase, *i.e.*, when more hinge functions are included in the HH model. Then it might be interesting to use the *conjugate gradient* method which builds up a Newton step by a series of gradient steps avoiding to compute the Hessian. This algorithm has been found successful in many neural network applications. See [8] and further references there.

In Section 5 the simultaneous update is compared to the HH algorithm in some simulation examples.

5 Examples

This section is divided into two parts, where the first part treats the improvement of the HFA by introducing a step length parameter to assure convergence. The second part deals with the simultaneous updating of all parameters instead of only a subset of them, and the performance is compared to the HH algorithm from [3].

5.1 Performance of the Modified HFA

Let us use the same data as in Figure 3 for which the original HFA did not converge for all initial parameter values. The HFA is now modified with a step length and started at different initial parameter values which correspond to different initial splits of the data. The result is that the modified algorithm always converges to one of the two local minima. The evolution of the hinge position is depicted in Figure 6. Compare this to the behavior of the original algorithm, depicted in Figure 4. For one of the initial values the algorithm jumps from one local minimum to the atractor of the second minimum. Such jumps can be prevented by implementing a more advanced step length rule, *e.g.*, the Armijo-Goldstein rule, see [4].

Figure 6 should be compared to Figure 4 in Section 2. When the damped Newton algorithm is used the HFA converges for all initial values, while using the unmodified HFA will

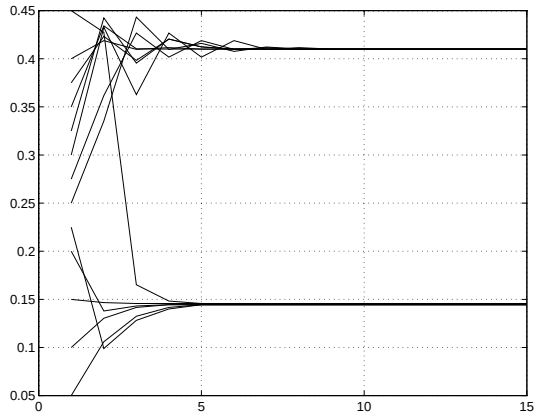


Figure 6: Evolution of the hinge position for a number of initial data splits in the interval $x = [0, \ldots, 0.46]$. There are two local minima.

result in no convergence or limit cycle behavior for some initial hinge intervals.

5.2 Simultaneous Parameter Updating

Two examples will be presented illuminating the practical differences between the stepwise updating of the parameters in the HFA and the simultaneous updating described in Section 4.

One iteration with the simultaneous updating means one step with the Newton update (12). One iteration with the HFA algorithm means one cycle of Newton updates where each hinge function is updated once. The HFA iteration will always be faster than the simultaneous one. However, in general, not as efficient (in the sense that the criterion decreases less).

In the examples to follow we will see that HFA is a short cut to gain speed which can turn out not to be the fastest way.

5.2.1 Simultaneous vs. Stepwise Updating

In this example we will compare the performance of the HFA and the Newton algorithm when applied to data generated by an HH model in two dimensions. The HH model contains two hinge functions and is depicted in Figure 7.

The position of the hinges of the two hinge functions are easier seen in Figure 8.

The input data is uniformly distributed on the square $[0, 1]^2$. The number of samples is $101 \times 101 = 10201$. The equations of the two hinges are:

$$\begin{aligned} x_1 &= 0.4 \\ x_1 - 0.1x_2 &= 0.45. \end{aligned}$$

The true parameter vectors, giving the hinges above, are perturbed and is then used as the initial vector. The equations of the two initial hinges given to the algorithms are:

$$1.1747x_1 + 0.0940x_2 = 0.3887$$

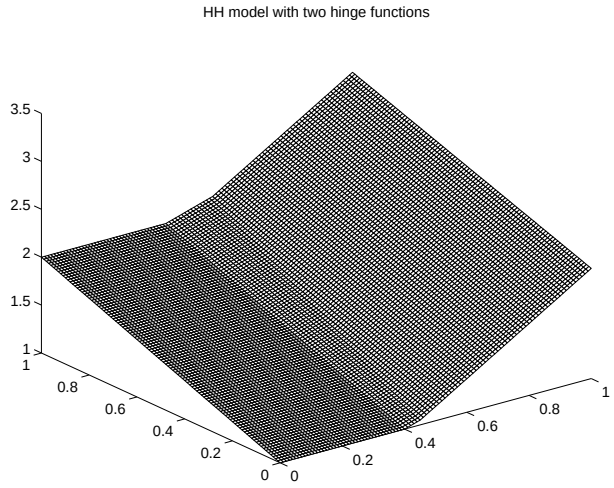


Figure 7: Two dimensional HH model consisting of two hinge functions.

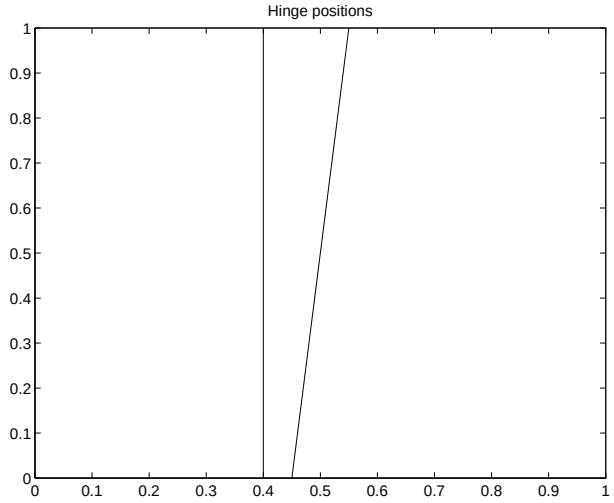


Figure 8: Hinges of the two hinge functions in Figure 7.

$$1.0527x_1 - 0.2045x_2 = 0.1956$$

Using the same initial parameter vectors the two algorithms are executed. The result is presented in Figures 9 and 10. To avoid a too messy plot only the result after every two iterations is shown in the plot.

The initial hinge positions are marked by "initial hinge #1" and "initial hinge #2" in Figure 9. Using the HFA algorithm both hinge positions jump to the right of the true positions and then very slowly converge towards the true hinges. In Figure 9 the hinge position for hinge number 1 almost does not move for iterations 4 to 8. Using the simultaneous updating Newton algorithm, on the contrary, the hinges converged to their true positions after 8 iterations. For the HFA over 225 iterations are necessary for convergence (not shown in the figure). When the HFA had iterated 225 iterations, the hinge positions were:

$$\begin{aligned} 1.0297x_1 - 0.0015x_2 &= 0.4126 \\ 0.9706x_1 - 0.0985x_2 &= 0.4374, \end{aligned}$$

which still deviates from the true position.

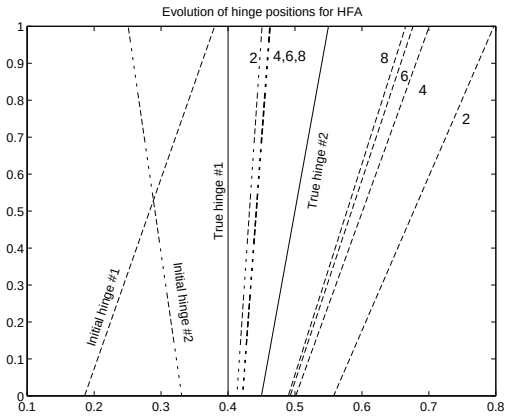


Figure 9: Hinge position evolution for 8 iterations of the HFA.

Remark 3: Following the recommended procedure in [3] we should have run the HFA using one hinge function until it converged, and then introduced the second hinge function. After the second hinge function is introduced, the HFA should be re-iterated with one iteration at a time. This procedure results in a worse result than the one-iteration procedure used in this example.

It is important to calculate the total time needed for the minimization of the criterion, and not to only consider how computationally complex parts of the algorithm are. For this two-hinge example an HFA iteration takes about 4.2 seconds and with the 225 iterations needed to reach close to minimum, the HFA needs about 15 minutes. For the simultaneous Newton algorithm each iteration demands approximately 5 seconds, multiplied with the number of iterations (8) this algorithm needs only 40 seconds for the minimization.

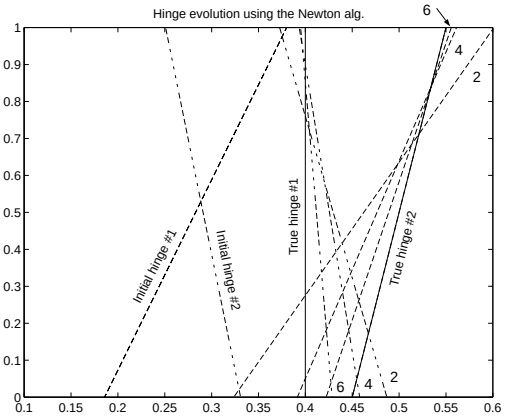


Figure 10: Hinge position evolution for 8 iterations of the Newton algorithm. Note that the last iteration ends up on the true hinge position.

Since the simultaneous updating Newton algorithm performed significantly better than the HFA, the off-diagonal elements in the Hessian are important in this example. This is well in line with the discussion in Section 4. The HFA is a short cut to gain some speed but as we have just seen is can easily turn out to be slower than the simultaneous Newton update.

5.2.2 A High Interaction Example

As the last example of this paper we have chosen to compare the algorithms in this paper using the same simulated data as in [3]. There the following "high interaction example" is designed. The dimension of the input space is 10 and the distribution of $\mathbf{x}$ is uniform on $[0,1]^{10}$. The following three functions are then formed:

$$\begin{aligned} l_1 &= 10x_1 + 9x_2 + 3x_3 + 7x_4 - 6x_5 - 5x_6 - 9x_7 - 3x_8 - 2x_9 - x_{10} \\ l_2 &= -x_1 - 2x_2 - 3x_3 - 4x_4 - 5x_5 - 6x_6 + 7x_7 + 8x_8 + 9x_9 + 10x_{10} \\ l_3 &= -x_1 - 2x_2 - 3x_3 + 4x_4 + 5x_5 + 4x_6 - 3x_7 - 2x_8 - x_9. \end{aligned}$$

The sample size was chosen to 400 and the functions l_1, l_2 and l_3 were normalized in such a way that they have the upper 97.5% sample equals 2.0. For example, if we consider the function l_1 we have obtained 400 samples of the linear function. Assume the samples of l_1 are ordered in a vector from the lowest, labeling the lowest sample by 1, to the largest, labeling the largest sample by 400. Then the element $l_{\text{ord}}(390)$ at sample number 390, which is 97.5 % of 400, in that ordered vector is read off. Finally, the function l_1 is multiplied with the same factor that would give the result 2 if it multiplied the read off value $l_{\text{ord}}(390)$. A second triple is formed by $v_i = 2(l_i - 2)$. Finally, the non-linear function is

$$g(\mathbf{x}) = \frac{e^{v_1}}{1+e^{v_1}} + \frac{e^{v_2}}{1+e^{v_2}} + \frac{e^{v_3}}{1+e^{v_3}}.$$

The function g is normalized by the factor a, so that the standard deviation of the function was 4.0, and then white noise with standard deviation 1.0 was added. The expression for the output data is hence

$$y_i = a \cdot g(\mathbf{x}_i) + e_i.$$

A general comment that can be made regarding the function above is that it belongs to the family named ridge functions, which are well suited for approximations with HH models. The shape of this particular function $g(\mathbf{x})$ is not possible to match exactly by HH models. In other words, an example chosen not to be too difficult, but not too easy either.

In [3] it is concluded that the best fit was obtained by a three hinge HH model. The conclusion comes from inspection of a, so called $\mathrm{PE}_{\mathrm{GCV}}$ criterion, which is a combination of the mean residual sum-of-squares (MRSS) and a penalty on the number of used hinge functions.

Our experience when running the algorithm proposed in this paper and the HFA is that there is a variation of the cpu-time depending on the chosen initial value of the parameter vector. Of course, also the final MRSS depends on the initial value due to local minima. The experiments were performed on a SPARCstation LX (running MATLAB). The cpu-time used varies from 5.35 seconds to 28.8 seconds for the algorithm with simultaneous updating of the parameters, and from 7.55 seconds to 35.73 seconds for the HFA. The MRSS goes from 1.12 to 1.16 for both algorithms depending on the initial value. The MRSS is the same for both algorithms, because the same minimum was reached using both the algorithms. We have summarized the results in Table 1.

HFA			Newton		
# iter	time	MRSS	# iter	time	MRSS
47	35.733	1.157	23	28.800	1.157
25	16.734	1.154	7	9.300	1.156
40	33.105	1.130	19	24.530	1.131
10	7.550	1.123	4	5.350	1.123
40	29.960	1.164	15	19.583	1.162

Table 1: Results of a comparison between HH model estimation using HFA and the method proposed in this paper. The first and fourth columns shows the number of iterations used by the HFA and Newton algorithm, respectively. The second and fifth columns contain the cpu-time, in seconds, used by the algorithms. The third and sixth column display the achieved mean residual sum-of-squares.

The initial values of the parameter vectors were picked at random. We used the same initial vectors for both algorithms. The table above shows the cpu-time consumed by the algorithms, the number of iterations needed, and the MRSS accomplished. One iteration

in the HFA case is one complete update of the HH model, *i.e.*, one iteration of each hinge function in the HH model.

The conclusion is that the performance of the simultaneous updating algorithm proposed here is better than the HFA algorithm. The average time needed per iteration is lower for the HFA, but the Newton algorithm needs less than half the number of iterations compared to the HFA.

6 Conclusions

In this paper the estimation algorithm for the recently introduced hinging hyperplane model has been considered. The main building block of the estimation algorithm is the so called hinge finding algorithm. It has been shown that it is a Newton algorithm for minimization of a quadratic loss function. From this insight we can make the following conclusions.

- Difficulties with the convergence with the original hinge finding algorithm can be circumvented by using standard techniques in numerical minimization theory. By introducing a step length the method is converted into a damped Newton algorithm which assures convergence.
- A damped Newton search can also be applied to all parameters simultaneously. This is an alternative to the iterative HH algorithm suggested in [3]. The difference between these two approaches and advantages and disadvantages of the proposed algorithm have been discussed. Which method is best, is problem dependent, but in general it can be expected that a simultaneous update gives faster convergence close to the minimum.
- From the two conclusions above, it is clear that there are no evident algorithmic advantageous with the HH model in comparison with for example neural nets. The same numerical search methods are applicable for both of these model structures.

The conclusions have been illustrated in the examples.

References

[1] A.R. Barron. "Universal Approximation Bounds for Superpositions of a Sigmoidal Function". *IEEE Trans. on Information Theory*, 39:930–945, May 1993.

[2] A. Benveniste, A. Juditsky, B. Delyon, Q. Zhang, and P-Y. Glorennec. Wavelets in identification. In Mogens Blanke and Torsten Söderström, editors, *Preprint SYSID '94 – 10th IFAC Symposium on System Identification*, volume 2, pages 27–48, July 1994.

[3] L. Breiman. "Hinging Hyperplanes for Regression, Classification, and Function Approximation". *IEEE Trans. on Information Theory*, 39(3), May 1993.

[4] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[5] J.H. Friedman and W. Stuetzle. "Projection pursuit regression". *Journal of the American Statistics Association*, 19:817–823, 1981.

[6] P. Pucar and J. Sjöberg. Parameterization and conditioning of hinging hyperplane models. Lith-isy-r-1727, available at: `http://rt-gw.isy.liu.se/cgi-bin/reports?year 1995`, Department of Electrical Engineering, Linköping University, Sweden, 1995.

[7] J. Sjöberg, H. Hjalmarsson, and L. Ljung. Neural networks in system identification. In Mogens Blanke and Torsten Söderström, editors, *Preprint SYSID '94 – 10th IFAC Symposium on System Identification*, volume 2, pages 49–72, July 1994.

[8] P.P. van der Smagt. "Minimisation Methods for Training Feedforward Neural Networks". *Neural Networks*, 7(1):1–11, 1994.