Paper

Incrementally Solving Nonlinear Regression Tasks Using IBHM Algorithm

Paweł Zawistowski and Jarosław Arabas

Institute of Electronic Systems, Warsaw University of Technology, Warsaw, Poland

Abstract—This paper considers the black-box approximation problem where the goal is to create a regression model using only empirical data without incorporating knowledge about the character of nonlinearity of the approximated function. This paper reports on ongoing work on a nonlinear regression methodology called IBHM which builds a model being a combination of weighted nonlinear components. The construction process is iterative and is based on correlation analysis. Due to its iterative nature, the methodology does not require a priori assumptions about the final model structure which greatly simplifies its usage. Correlation based learning becomes ineffective when the dynamics of the approximated function is too high. In this paper we introduce weighted correlation coefficients into the learning process. These coefficients work as a kind of a local filter and help overcome the problem. Proof of concept experiments are discussed to show how the method solves approximation tasks. A brief discussion about complexity is also conducted.

Keywords—black-box modeling, neural networks, nonlinear approximation, nonlinear regression, support vector regression, weighted correlation.

1. Introduction

In this paper the problem of solving nonlinear regression tasks is considered. The task consists in finding a function $\hat{f} : \mathbb{R}^n \to \mathbb{R}$ such that for the approximated function $f : \mathbb{R}^n \to \mathbb{R}$ the error between f and $\hat{f}$ function values is minimal. The f function is unknown, however sample input data $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$ and function values $Y = \{y_1, \ldots, y_t : y_i = f(\mathbf{x}_i)\}$ are given.

As regression tasks occur in many areas of research and industry, various methods of solving such tasks have been developed. These range from simple linear regression, through generalized regression to black-box modeling algorithms such as neural networks.

This paper reports on development of a black-box approximation method called IBHM, which is a shorthand for Incrementally Built Heterogenous Model, introduced in [1] and [2]. The method iteratively creates models similar in structure to MLP or RBF neural networks [3]. During this process, IBHM determines both parameters and the model structure, so no a priori assumptions are required, which makes the method very convenient to use. Although this is a relatively new approach, it has already achieved very good results in comparison to other methods [4]. This paper focuses on presenting the proper background and ideas connected with IBHM and also formulates the latest version of the algorithm.

There are various other black-box approximation methods, such as e.g., the already mentioned neural networks, that process models similar to IBHM. Ideas similar to the concept of iterative correlation based learning can also be found in other areas of research. In this paper we briefly refer to these approaches in the context of the presented method.

The paper is organized as follows. Section 2 introduces the correlation based learning used by IBHM and describes the ideas behind it. A detailed algorithm formulation along with computational complexity discussion and comparison to other methods is given in Section 3. Section 4 reports obtained experimental results. Finally a summary and discussion of future work is given in Section 5.

2. Correlation Based Learning

2.1. Genesis

The basic concepts behind IBHM originate from the well known method of linear regression. This technique creates models $\hat{f} : \mathbb{R}^n \to \mathbb{R}$ having the form

$$\hat{f}(\mathbf{x}) = \mathbf{w}^T \mathbf{x} \; + \; w_0, \tag{1}$$

where $\mathbf{w}, \mathbf{x} \in \mathbb{R}^n$ and $w_0 \in \mathbb{R}$. Linear regression fails to lead to good results if $f(\mathbf{x})$ is nonlinear.

A possible way to overcome such problems is to use a mapping function $\Phi : \mathbb{R}^n \to \mathbb{R}^m$ to transform the original variable $\mathbf{x}$ before applying linear regression. If the dependency between $\Phi(\mathbf{x})$ and $f(\mathbf{x})$ is linear then the linear regression can be applied to construct the model which effectively is nonlinear

$$\hat{f}(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}) \; + \; w_0. \tag{2}$$

This approach would be a perfect solution to complicated modeling problems, but finding a proper Φ transformation is virtually impossible without detailed knowledge about the approximated function.

2.2. Single Nonlinear Component

Here we attempt to formulate a methodology that tries to guess the most appropriate form of the Φ mapping by selecting one of many possible candidates.
Let $h : \mathbb{R}^n \to \mathbb{R}$ be a scalarization function and $g : \mathbb{R} \to \mathbb{R}$ be a monotonous activation function. Now assume that $m = 1$, that is, a mapping $\Phi : \mathbb{R}^n \to \mathbb{R}$ is sufficient to build a linear model and has the form

$$\Phi(\mathbf{x}) = g\left(a \cdot h(\mathbf{x}, \mathbf{d}) + b\right) , \tag{3}$$

where $a, b \in \mathbb{R}$ are scalar parameters and $\mathbf{d} \in \mathbb{R}^n$ is a parameter vector.
The regression model has the form

$$\hat{f}(\mathbf{x}) = w_1 \cdot \Phi(\mathbf{x}) + w_0 = w_1 \cdot g\left(a \cdot h(\mathbf{x}, \mathbf{d}) + b\right) + w_0 . \tag{4}$$

Observe that the regression model parameters can be estimated by a two-step procedure:

- estimation of parameters a, b and $\mathbf{d}$,
- computing of weights w_0, w_1 via linear regression.

Prior to defining a method to estimate values of a, b and $\mathbf{d}$ a couple of observations has to be made. First notice that, as $\hat{f}(\mathbf{x})$ is to approximate $f(\mathbf{x})$, they must be linearly correlated. Inspection of Eq. (4) reveals that a high level of linear correlation is expected between $f(\mathbf{x})$ and $g(a \cdot h(\mathbf{x}, \mathbf{d}) + b)$. As the output values of the g function depend on parameters a and b, their proper values can be found by maximizing the correlation between f and g with respect to a and b.
A similar approach, which utilizes a rank correlation, can be used to find the $\mathbf{d}$ vector value. Observe that the output of the scalarization function h also has to be correlated with the output of f. In this case, however, linear correlation cannot be used, because the output values of h are transformed by the activation function which is nonlinear. In consequence, linear correlation may be improperly indicating dependencies between h and f, however, as the g function is monotonous, rank correlation will perform this task instead.Value of $\mathbf{d}$ can therefore be found by maximizing the rank correlation between f and $h(\mathbf{x}, \mathbf{d})$. The remaining weights w_0, w_1 can finally be estimated using linear regression.

2.3. Multiple Nonlinear Components

Up to this point the case of $m = 1$ has been considered. Now we consider a general case where $m > 1$ and assume that the Φ mapping has the following form

$$\Phi(\mathbf{x}) = \begin{bmatrix} g_1\left(a_1 \cdot h_1(\mathbf{x}, \mathbf{d}_1) + b_1\right) \\ \dots \\ g_m\left(a_m \cdot h_m(\mathbf{x}, \mathbf{d}_m) + b_m\right) \end{bmatrix} , \tag{5}$$

where $h_i : \mathbb{R}^n \to \mathbb{R}$ are scalarization functions, $g_i : \mathbb{R} \to \mathbb{R}$ are monotonous activation functions and $a_i, b_i, \mathbf{d}_i$ are parameters, $i = 1, \dots, m$.

Thus we consider the following regression model

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{m} w_i \cdot g_i(a_i \cdot h_i(\mathbf{x}, \mathbf{d}_i) + b_i) + w_0 . \tag{6}$$

All parameters in Eq. (6) can be estimated using the previously described procedure based on the correlation analysis iteratively, as presented in Algorithm 1. There the iteration loop starts in line 2. In iteration k first rank correlation between the current approximation residual ε_{k-1} and h output values is maximized to estimate $\mathbf{d}_k$ – line 4. Then a_k and b_k are similarly found via linear correlation maximization – line 5. Finally the residual ε_k is calculated – line 7 and linear regression is performed – line 8.

Algorithm 1:

Input: $X = \{\mathbf{x}_1, \dots, \mathbf{x}_m : \mathbf{x}_i \in R^n\}$ - training sample set
Result: $\hat{f}$ - approximation function

1 $\varepsilon_0(\mathbf{x}) \leftarrow f(\mathbf{x})$, $k \leftarrow 0$
2 **while** *the stop criterion is not satisfied* **do**
3 $\quad k \leftarrow k + 1$
4 $\quad \mathbf{d}_k \leftarrow \arg\max_j |r(h_k(X, \mathbf{d}), \varepsilon_{k-1}(X))|$
5 $\quad (a_k, b_k) \leftarrow \arg\max_{(a,b)} |r(g_k(a \cdot h_k(X, \mathbf{d}_k) + b), \varepsilon_{k-1}(X))|$
6 $\quad$ assume $\hat{f}_k(\mathbf{x}) = \sum_{i=1,\dots,k} w_i \cdot g_i(a_i \cdot h_i(\mathbf{x}, \mathbf{d}_i) + b_i)$
7 $\quad$ assume $\varepsilon_k(\mathbf{x}) = \hat{f}_k(\mathbf{x}) - f(\mathbf{x})$
8 $\quad [w_0, \dots, w_k] \leftarrow \arg\min_{[w_0,\dots,w_k]} \sum_{\mathbf{x} \in X} \left(\varepsilon_k(\mathbf{x})\right)^2$
9 **end**
10 $\hat{f}(\mathbf{x}) = \hat{f}_k(\mathbf{x})$

Finding parameters in the described fashion is in some sense a greedy approach. This is because during each iteration the method tries to fit the current scalarization and activation functions to the entire approximation residual even if this leads to a more complicated situation in the future. To get a better insight into this problem, consider

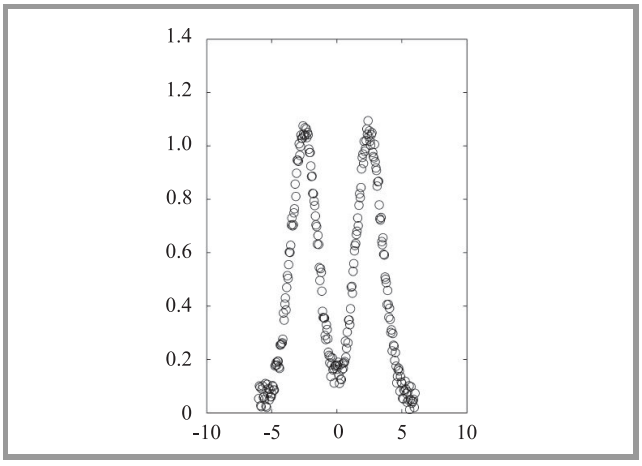

Fig. 1. An example approximated function.

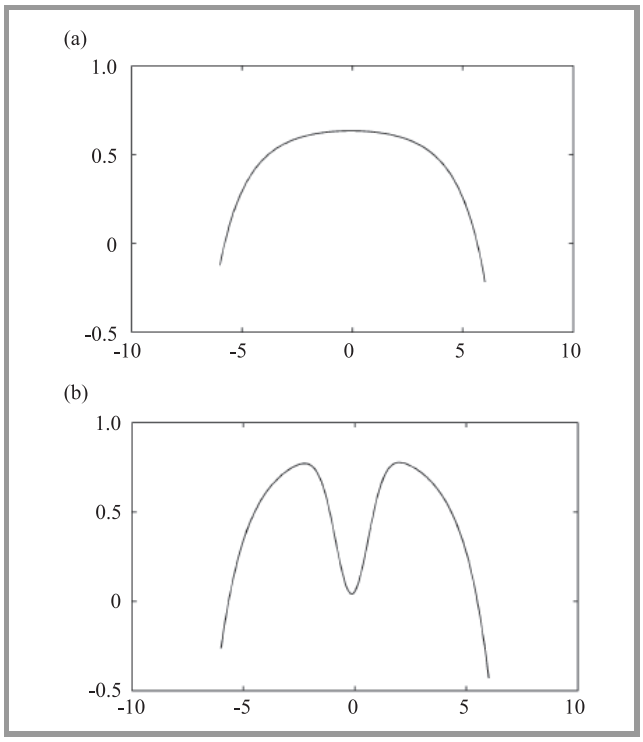

Fig. 2. Approximating function from Fig. 1 using Algorithm 1: (a) after the first iteration, (b) after the second iteration.

approximating the function given in Fig. 1. Application of Algorithm 1 would lead in this case to results similar to Fig. 2 which do not fully reflect the approximated function. This is because in the first iteration this approach tries to approximate the whole function instead on focusing on one of the two clearly distinct components. As a result it is unable to build a perfect model in two iterations.

2.4. Weighted Correlation

An improvement to the described situation can be made by using weighted correlation coefficients. Let us define the weighted linear correlation as

$$r_\omega(X,Y) = \frac{E_\omega(XY) - E_\omega(X)E_\omega(Y)}{\sqrt{(E_\omega(X^2) - E_\omega^2(X))\,(E_\omega(Y^2) - E_\omega^2(Y))}}\ , \quad (7)$$

and the weighted rank correlation as

$$\rho_\omega(X,Y) = r_\omega(rank(X), rank(Y))\ , \quad (8)$$

where

$$E_\omega(X) = \frac{\sum_{x\in X}\omega(x)x}{\sum_{x\in X}\omega(x)}\ . \quad (9)$$

Furthermore we consider a Gaussian weighting function of the form

$$\omega(x) = \frac{1}{\sqrt{2\pi}v}\, e^{\frac{-x^2}{2v^2}}\ . \quad (10)$$

When the algorithm uses the defined weighted coefficients instead of trying to decrease the approximation residual as much as possible in each iteration, the method focuses on identifying and approximating specific components of the approximated function. This means that IBHM tries to decompose the approximated function.

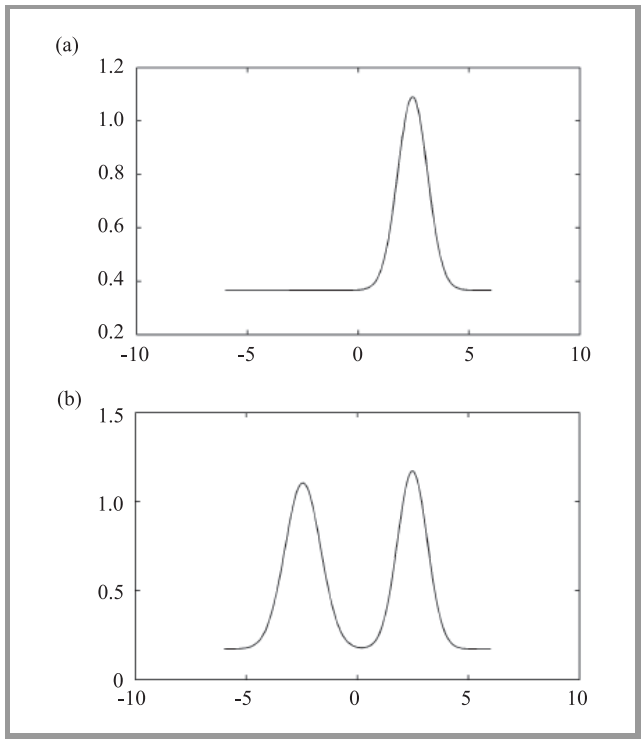

Fig. 3. Approximating function from Fig. 1 using Algorithm 1 with weighted correlation coefficients: (a) after the first iteration, (b) after the second iteration.

Coming back to the example, if we change lines 4 and 5 in Algorithm 1 to use the weighted correlation coefficients, the obtained approximation results are different as presented in Fig. 3. A close inspection of these results reveals that the weights work as a kind of a filter which puts focus on local features of the approximated function. Therefore it is possible to approximate a specific component of the approximated function, different in each iteration and get a more accurate model in the end.

3. IBHM Algorithm

3.1. Method Definition

Extending the conventions used in Subsection 2.1, let $H = \{\bar{h}_1, \ldots, \bar{h}_v : \bar{h}_i : \mathbb{R}^n \to \mathbb{R}\}$ denote the set of candidate scalarization functions and let $G = \{\bar{g}_1, \ldots, \bar{g}_u : \bar{g}_i : \mathbb{R} \to \mathbb{R}\}$ denote the set of candidate monotonous activation functions. The functions present in the final model are to be chosen from these two sets.

Algorithm 2 presents the proposed method, which builds the model given by Eq. (6), where $\forall_i\ g_i \in G \wedge h_i \in H$. The algorithm has three distinct parts which follow the procedure already described in Subsection 2.1. In the first part, for each candidate scalarization function, a proper parameter vector is found via the weighted rank correlation maxi-

mization – line 5. Then the best candidate is selected and set as the k-th scalarization function – lines 7–9. In the second part of the iteration, the parameters for candidate activation functions are found via the weighted linear correlation maximization – line 11. Then the best candidate is chosen as the k-th activation function – lines 13–15. In the third part of the iteration linear regression is used to estimate the model's weights – line 18.

Algorithm 2: IBHM

Input: $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m : \mathbf{x}_i \in R^n\}$ - training sample set
Result: $\hat{f}$ - approximation function

1 $\varepsilon_0(\mathbf{x}) \leftarrow f(\mathbf{x})$, $k \leftarrow 0$
2 **while** *the stop criterion is not satisfied* **do**
3 $\quad k \leftarrow k+1$
$\quad$ `/* Part 1 - finding scalarization function and parameter */`
4 $\quad$ **for** $i = 1, \ldots, |H|$ **do**
5 $\quad\quad \hat{\mathbf{d}}_i \leftarrow \arg\max_{\mathbf{d}} \left|\rho_\omega(\bar{h}_j(X, \mathbf{d}), \varepsilon_{k-1}(X))\right|$
6 $\quad$ **end**
7 $\quad i_k \leftarrow \arg\max_i \left|\rho_\omega(\bar{h}_i(X, \hat{\mathbf{d}}_i), \varepsilon_{k-1}(X))\right|$
8 $\quad \mathbf{d}_k \leftarrow \hat{\mathbf{d}}_{i_k}$
9 $\quad h_k \leftarrow \bar{h}_{i_k}$
$\quad$ `/* Part 2 - finding activation function and parameters */`
10 $\quad$ **for** $j = 1, \ldots, |G|$ **do**
11 $\quad\quad (\hat{a}_j, \hat{b}_j) \leftarrow \arg\max_{(a,b)} \left|r_\omega(\bar{g}_j(a \cdot h_k(X, \mathbf{d}_k) + b), \varepsilon_{k-1}(X))\right|$
12 $\quad$ **end**
13 $\quad j_k \leftarrow \arg\max_j \left|r_\omega(\bar{g}_j(\hat{a}_j \cdot h_k(X, \mathbf{d}_k) + \hat{b}_j), \varepsilon_{k-1}(X))\right|$
14 $\quad (a_k, b_k) \leftarrow (\hat{a}_{j_k}, \hat{b}_{j_k})$
15 $\quad g_k \leftarrow \bar{g}_{j_k}$
$\quad$ `/* Part 3 - extending the model */`
16 $\quad$ assume $\hat{f}_k(\mathbf{x}) = \sum_{i=1,\ldots,k} w_i \cdot g_i(a_i \cdot h_i(\mathbf{x}, \mathbf{d}_i) + b_i)$
17 $\quad$ assume $\varepsilon_k(\mathbf{x}) = \hat{f}_k(\mathbf{x}) - f(\mathbf{x})$
18 $\quad [w_0, \ldots, w_k] \leftarrow \arg\min_{[w_0,\ldots,w_k]} \sum_{\mathbf{x} \in X} (\varepsilon_k(\mathbf{x}))^2$
19 **end**
20 $\hat{f}(\mathbf{x}) = \hat{f}_k(\mathbf{x})$

Main loop of the algorithm is controlled by a stop criterion. This criterion should indicate if increase of the models complexity improves the overall results. A possible candidate method for that criterion is to stop when an increase in the Akaike Information Criterion (AIC) [5] is observed, where AIC is defined as

$$\mathrm{AIC}(X) = 2 \cdot p + |X| \cdot \ln\Big(\sum_{\mathbf{x} \in X} (\varepsilon_k(\mathbf{x}))^2\Big) \ , \qquad (11)$$

and p is the number of parameters estimated for the model[1].

[1] In case of m iterations of IBHM $p = m \cdot (n+3) + 1$.

Another possibility is to use a separate validation set to estimate the current model error and to stop when it increases.

3.2. Computational Complexity

Dominant operations which influence IBHM's complexity are connected with optimization tasks performed in each iteration. For that reason, a thorough complexity analysis of IBHM in the general case is impossible, as it would require precisely stating the complexities of solving unknown global optimization tasks. What follows however, is a rough discussion giving some insight into how costly is the algorithm in comparison with other methods.

In each IBHM iteration, a number of optimization tasks are solved. For each scalarization function from the H set, a global optimization of the $\mathbf{d} \in \mathbb{R}^n$ vector is solved. Estimation of parameters a, b is performed for each activation function from the set G. Each iteration is concluded with linear regression which is a quadratic problem of a size dependent on the iteration number.

When compared to methods which assume a fixed model structure, e.g., MLP neural networks, where only a single optimization task is solved, IBHM may seem to be overwhelmingly expensive. This is until dimensionality, a key factor connected with optimization tasks, is considered. Because of its iterative nature, the increase in the number of nonlinear components does not influence the number of parameters that undergo global optimization in each IBHM iteration. This means that in the optimization tasks solved by IBHM are simpler than in case of MLP.

Consider an example in which an approximated function $f : \mathbb{R}^n \to \mathbb{R}$ is is given. The approximation function is assumed to have the following structure

$$\hat{f}(\mathbf{x}) \ = \ \sum_{i=1}^{m} w_i \cdot tanh(a_i \cdot \mathbf{d}^T \mathbf{x} + b_i) \ + \ w_0 \ . \qquad (12)$$

This model corresponds to a MLP neural network with a single hidden layer of m-neurons and a hyperbolic tangent activation function. For MLP it is sufficient to assume that $a_i = 1$, therefore estimation of the parameters requires solving a single $m \cdot (n+2) + 1$ dimensional global optimization task.

The same model can also be constructed within m iterations of IBHM with $H = \{\mathbf{d}^t \mathbf{x}\}$ and $G = \{tanh(x)\}$. In this case we have to solve m global optimization tasks in $\mathbb{R}^n$ and $\mathbb{R}^2$ and m quadratic optimization tasks.

Table 1
Time units required to prepare the model from Eq. (12) (for $m = 10,\ n = 5$) in case of IBHM and MLP using optimization methods of various expected costs.

Alg. exp. cost	IBHM	MLP
$\Theta(N)$	70	71
$\Theta(N^2)$	2 900	5 041
$\Theta(N^3)$	133 000	357 911

If we have a global optimization method with the expected cost $\Theta(N^\alpha)$ where N is the optimization task dimensionality, we may estimate the computational effort required to prepare an MLP network as requiring $\Theta(mn+2m+1)^\alpha$ units. In the IBHM case, the cost is $\Theta(mn^\alpha+3m^\alpha)$ units plus the negligible cost of solving m quadratic optimization tasks. For various values of α the computational effort required by IBHM turns out to be far smaller then in case of MLP, as shown in Table 1.

3.3. Similar Methods

Neural networks. Multiple Layer Perceptron (MLP) or Radial Basis Function (RBF) type neural networks [3] can be used to create models very similar to those created by IBHM. The main difference is that IBHM does not require a priori assumptions about the final model structure, while constructing MLP or RBF networks requires setting the number of neurons to be used. Also the correlation based learning used by IBHM is a completely different from the techniques utilized by neural networks.
One important fact to notice is that IBHM may be utilized alongside neural networks as a preprocessing step estimating the required number of networks. This has been shown [4] to lead to good results in case of MLP models.

SVR. Support Vector machines for Regression (SVR, [6]) construct models with a similar structure to IBHM models. Another similarity is that, this method determines the model structure using the training data. The learning algorithm utilized determines most of the parameters directly using training data points, which are called support vectors. This is a different approach from the one IBHM uses.
The main drawback of SVR, not shared by IBHM, is that it tends to create very large models, which may lead to generalization problems.

GMDH. The Group Method of Data Handling [7] is a heuristic method which creates approximators using high order polynomials. The method works iteratively and puts focus on pairs of input variables in each iteration. Combination of these pairs form polynomials of higher orders which are added as new variables. The best from the new variables is treated as the current model. If the stop criterion is satisfied the algorithm terminates, otherwise the next iteration works on pairs of the variables introduced in the previous iteration and so on.
The iterative process uses a validation stop criterion, so the algorithm terminates when an error increase on the validation set is observer.
The GMDH learning process shares some similarities with the approach utilized by IBHM, however this method does not use correlation analysis during parameter estimation. Furthermore the estimation is done using a one step process, while IBHM splits this into three steps. Also GMDH works only on polynomials, while IBHM works on many different scalarization and activation functions.

CLEAN algorithms for signal processing. The incremental model building realized by IBHM can be viewed from a different perspective. In each iteration the algorithm tries to identify parts of the approximated function – its components. When such a component is identified, it is subtracted from the remaining residual, so that further components can be identified in future iterations. The idea to try and identify certain components present in a complex pattern, remove them and find other, previously not visible, components can be found in other areas of research. CLEAN radar algorithms ([8], [9]) are one such area. These algorithms address the problem of identifying multiple targets by iteratively filtering them out. Although the problem domain and methods are different from those utilized by IBHM, the core idea is quite similar.

4. Experiments

The focus of this paper is on reporting progress made on IBHM development therefore the goal of the presented experimental results is to illustrate the way the method works. For this purpose some toy problems were prepared and solved using IBHM.

4.1. Approximated Functions

The experiments were conducted using the following three functions:

$$f_1(x) = e^{-\frac{(x-8)^2}{2}} + e^{-\frac{x^2}{2}} + e^{-\frac{(x+8)^2}{2}} + \varepsilon_{N(0,0.01)} \ , \qquad (13)$$

$$f_2(x) = \frac{1}{5}\sin(x) + \frac{1}{10}x + \varepsilon_{N(0,0.01)} \ , \qquad (14)$$

$$f_3(x) = 1 - \tanh(x+6) + \tanh(x-6) + 2\cdot e^{-\frac{x^2}{2}} + \varepsilon_{N(0,0.01)} \ , \qquad (15)$$

where $\varepsilon_{N(0,0.01)}$ is a normally distributed random variable with mean 0 and variance 0.01. These functions were chosen as they represent various types of nonlinearities occurring in approximation tasks. The random component reflects the unknown factors influencing the approximated functions, therefore it compensates for the lack of knowledge.
For each of these functions two data sets were prepared: a training set of 160 examples and a test set of 80 examples. Test sets were used only to calculate the results reported in Subsection 4.3.

4.2. Algorithm Setup

IBHM was set up with $G = \{\tanh(x), x, \log\mathrm{sig}(x)\}$ and $H = \{d\cdot x, (x-d)^2\}$, where

$$\log\mathrm{sig}(x) = \frac{1}{1+\exp(-x)} \ . \qquad (16)$$

Both global optimization tasks were solved using the CMA-ES optimizer [10], while the Nelder-Mead Simplex [11] was used to perform the linear regression task. A validation criterion was used to determine when to stop the algorithm.

This criterion divided the training set into two parts: $\frac{2}{3}$ of data used to estimate the parameters, while the remaining part was used to estimate the error. The algorithm was stopped when an error increase was observed.

4.3. Results

For each approximated function, the algorithm was ran 20 times using the training data sets. Then using the test sets for each created model the mean squared error defined as

$$MSE = \frac{1}{n} \sum_{i=1,\ldots,n} \left(f(\mathbf{x}_i) - \hat{f}(\mathbf{x}_i) \right)^2 \qquad (17)$$

was calculated. Table 2 contains the aggregated results of the experiments. In this table the second column contains the standard deviations σ of the random components $\varepsilon_{N(0,\sigma)}$ present in the training data. For each problem the mean values and standard deviations of MSE values and model sizes are reported. The model size is the number of nonlinear components – activation functions used.

Table 2
The aggregated results of the experiments

Problem	Mean test MSE	Mean model size
f_1	0.0317 ± 0.0021	3.1 ± 0.3
f_2	0.0135 ± 0.0045	4.0 ± 0.3
f_3	0.0265 ± 0.0121	4.7 ± 0.8

When compared with the variance of the random components present in the training data the obtained error levels suggest that IBHM was able to capture the general approximated function structure without overfitting to the noise. This can be also noticed by inspection of the best models for each approximated function which are shown in Figs. 4, 5 and 6. The structures of these models are given in Appendix A.

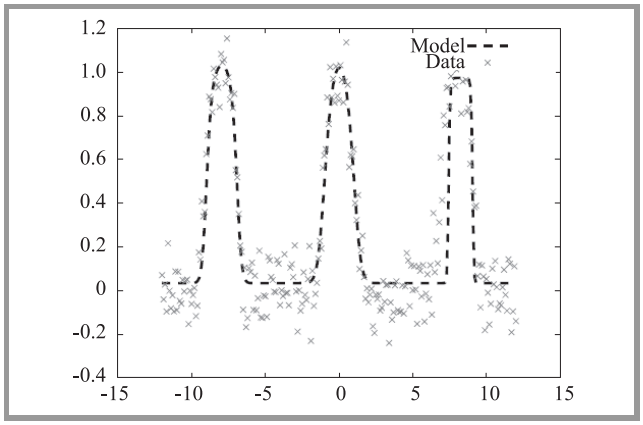

Fig. 4. The best ($MSE = 0.0320$) model created for $f_1(x)$ given in Eq. (19).

These results suggest that the noise present in the data does not degrade the correlation based learning method utilized by IBHM. Furthermore, it may be even argued that in some situations the presence of random components may be beneficial. This is because such components push the learning method away from focusing on small, insignificant parts of the approximated function and thus from overparametrization.

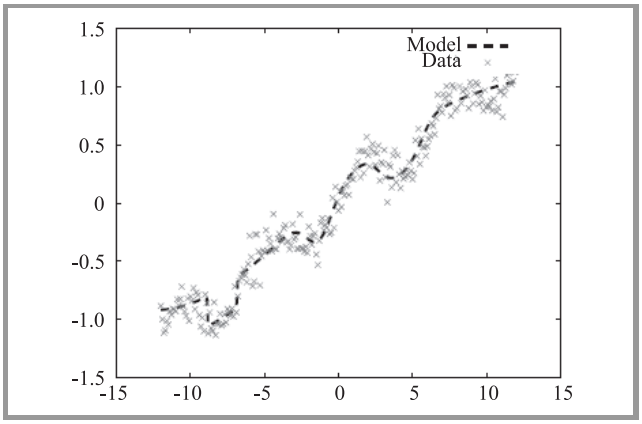

Fig. 5. The best ($MSE = 0.0109$) model created for $f_2(x)$ given in Eq. (20).

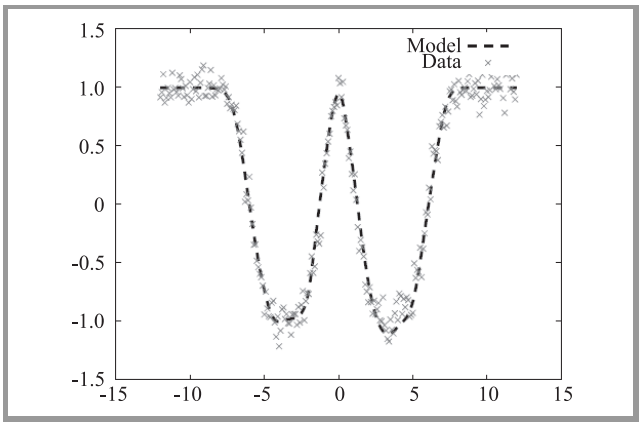

Fig. 6. The best ($MSE = 0.0165$) model created for $f_3(x)$ given in Eq. (21).

Another important aspect of the models built using IBHM was shown in [4]. This recent paper compared IBHM with MLP and SVR methods in the time series forecasting domain using 111 benchmark data sets from NN3 forecasting competition [12]. The comparison was based on results from experiments which put focus on evaluating the three different learning algorithms utilized by IBHM, MLP and SVR. The experimental procedure consisted in creating nonlinear autocorrelation models which defined the forecasts as

$$\hat{x}(t) = x(t-\tau_1) + \hat{f}\left(x(t-\tau_1), \ldots, x(t-\tau_{18})\right), \qquad (18)$$

where $\tau_1, \ldots, \tau_n$ were lags and $\hat{f}$ were nonlinear functions approximated using the three compared methods. The structure of the models where estimated using Akaike's Information Criterion. The built models' errors were estimated on test data sets and the whole training and

testing cycle was performed 25 times for each time series. The gathered data was used to estimate average MSE values and to perform pairwise comparison of models. In the comparison it was assumed that model A is better than B only when the median MSE for A was smaller than for B and the difference was statistically significant. Rank of a model for a time series was equal to one plus the number of models whose results were better for that time series. The results obtained are presented in Table 3.

Table 3
Summary of results aggregated over 111 time series from NN3 benchmark

Method	MSE mean	Model size*	Rank
IBHM	0.091 ± 0.212	1.09	1.76
MLP	0.093 ± 0.219	1.11	2.07
SVR	0.093 ± 0.209	4.80	2.46

* Number of nonlinear components averaged over all the models built by the given method.

These results show that IBHM performs well in comparison to MLP and SVR methods and that the models it creates tend to be rather small. This is an important virtue, as constructing large models may easily lead to overparameterization and generalization problems.

5. Summary and Future Work

IBHM is a promising approximation algorithm and the experimental results prove that the concepts behind it work. The method can be used in various fields ranging from time series forecasting to modeling complex physical processes. The models it constructs are similar in structure to those created using other well known and respected methods as MLP or SVR. Due to its iterative nature and decompositional properties, it does not require a priori assumptions about the final model structure, which makes it a convenient tool.

The future work on IBHM will focus on a couple of aspects. Theoretical analysis of the correlation based learning is to be conducted to provide a strong background for the algorithm. Furthermore, questions regarding optimal stop criteria and weighting functions need to be answered. There are also possibilities to extend the algorithm and to enable it to build more complex models with a structure similar to cascade correlation neural networks [13].

Another important aspect of further development is preparing an efficient implementation of the algorithm. The major amount of computation in IBHM is devoted to solving multiple optimization tasks. Fortunately these can be parallelized which opens up the possibility of utilizing General Purpose Graphical Processing Units (GPGPUs) to create a highly efficient implementation.

Appendix A

Sample Models

$$\begin{aligned}\hat{f}_1(x) \;=\; & 0.47\tanh\left(-6.98\cdot(x-8.25)^2+4.57\right)\\ & +1.06\,\mathrm{logsig}\left(-2.70\cdot(x+7.95)^2+2.58\right)\\ & -1.33\,\mathrm{logsig}\left(1.62\cdot(x-0.02)^2-1.01\right)+1.85\end{aligned} \tag{19}$$

$$\begin{aligned}\hat{f}_2(x) \;=\; & 43.06\,\mathrm{logsig}\left(0.01\cdot(x+12.42)^2+2.98\right)\\ & +0.54\,\mathrm{logsig}\left(0.64\cdot(x-4.15)^2-0.68\right)\\ & -0.27\,\mathrm{logsig}\left(-389.18\cdot(x+7.85)^2+390.57\right)\\ & -51.14\tanh\left(-0.34\cdot(x+1.30)^2-2.88\right)-93.57\end{aligned} \tag{20}$$

$$\begin{aligned}\hat{f}_3(x) \;=\; & 2.93\,\mathrm{logsig}\left(0.40\cdot(x-3.30)^2-0.93\right)\\ & -3.03\,\mathrm{logsig}\left(-0.38\cdot(x+2.70)^2+0.53\right)\\ & +0.46\,\mathrm{logsig}\left(-2.25\cdot(x+0.67)^2+2.11\right)\\ & -1.35\,\mathrm{logsig}\left(-1.02\cdot(x-5.74)^2+0.12\right)\\ & +1.90\tanh\left(0.31\cdot(x+5.25)^2+0.40\right)-3.84\end{aligned} \tag{21}$$

References

[1] J. Arabas and A. Dydyński, "An algorithm of incremental construction of nonlinear parametric approximators", in *Evolutionary Computation and Global Optimization 2006*, J. Arabas, Ed. Warsaw: WUT Press, Poland 2006, pp. 31–38.

[2] J. Arabas and A. Dydyński, "Nonlinear time-series modeling and prediction using correlation analysis", in Proc. Appl. Math. Mech. PAMM 2007, vol. 7, pp. 2030013–2030014, 2007.

[3] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[4] P. Zawistowski and J. Arabas, "Benchmarking IBHM method using NN3 competition dataset", in *Proc. Hybrid Artif. Intel. Syst. Conf. HAIS 2011*, Wrocław, Poland, 2011. LNCS, Springer, vol. 6678, pp. 263–270, 2011.

[5] H. Akaike, "A new look at the statistical model identification", *IEEE Trans. Autom. Contr.*, vol. 19, no.6, pp 716–723, 1974.

[6] B. Schölkopf and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[7] J. Farlow Stanley, "The GMDH algorithm of Ivakhnenko", *The American Statistician*, vol. 35, no. 4, pp. 210–215, 1981.

[8] D. Hai, "Effective CLEAN algorithms for performance-enhanced detection of binary coding radar signals", *IEEE Trans. Signal Process.*, vol. 50, no. 1, pp. 72–78, 2004.

[9] T. L. Foreman, "Reinterpreting the CLEAN algorithm as an optimum detector", in Proc. IEEE Radar Conf., Verona, NY, USA, 2006, pp. 24–27.

[10] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size", *IEEE Congr. Evol. Comput.*, pp. 1769–1776. 2005.

[11] J. A. Nelder and R. Mead, "A simplex method for function minimization", *Comput. J.*, vol. 7, pp. 308–313, 1965.

[12] "Artificial Neural Network and Computational Intelligence Forecasting Competition" [Online]. Available: http://www.neural-forecasting-competition.com/NN3/index.htm

[13] S. Fahlman and C. Lebiere, "The cascade-correlation learning architecture", *Adv. Neural Inform. Process. Sys.*, no. 2, pp. 524–532, 1990.

Paweł Zawistowski was born in Poland in 1984. He received the M.Sc. degree from Warsaw University of Technology (WUT), Poland, in 2008. Since 2008 he has been a Ph.D. student at the Faculty of Electronics and Computer Engineering, WUT. His scientific interests include neural modeling, black-box learning methods and meta-heuristics. He has taken part in various projects in which he constructed black-box models of industrial systems and natural phenomena.

E-mail: p.zawistowski.2@elka.pw.edu.pl
Institute of Electronic Systems
Warsaw University of Technology
Nowowiejska st 15/19
00-665 Warsaw, Poland

Jarosław Arabas – for biography, see this issue, p. 10.