Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 70 (2007) 2735–2743
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Heuristics for the selection of weights in sequential feed-forward neural networks: An experimental study

Enrique Romero*, René Alquézar

Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain

Available online 22 May 2007

Abstract

The selection of weights of the new hidden units for sequential feed-forward neural networks (FNNs) usually involves a non-linear optimization problem that cannot be solved analytically in the general case. A suboptimal solution is searched heuristically. Most models found in the literature choose the weights in the first layer that correspond to each hidden unit so that its associated output vector matches the previous residue as best as possible. The weights in the second layer can be either optimized (in a least-squares sense) or not. Several exceptions to the idea of matching the residue perform an (implicit or explicit) orthogonalization of the output vectors of the hidden units. In this case, the weights in the second layer are always optimized. An experimental study of the aforementioned approaches to select the weights for sequential FNNs is presented. Our results indicate that the orthogonalization of the output vectors of the hidden units outperforms the strategy of matching the residue, both for approximation and generalization purposes.

Keywords: Bias/variance decomposition; Feed-forward neural networks; Sequential approximations

1. Introduction

The well-known architecture of a fully connected feed-forward neural network (FNN) with one hidden layer of units and one output linear unit computes a function $f:\mathbb{R}^I \to \mathbb{R}$ defined as

$$f(x) = b_0 + \sum_{k=1}^{N} \lambda_k \varphi_k(\omega_k, x, b_k), \quad \omega_k \in \mathbb{R}^I \ \lambda_k, b_0, b_k \in \mathbb{R}, \tag{1}$$

where N is the number of units in the hidden layer and φ_k is the activation function of the kth hidden unit. For convenience, we refer to the weights in the first layer ($\{\omega_k\}_{k=1}^N$) as *hidden-layer weights* and to the weights in the second layer ($\{\lambda_k\}_{k=1}^N$) as *output-layer weights*. The *biases* $\{b_k\}_{k=1}^N$ can be seen as part of the hidden-layer weights and b_0 as an output-layer weight.

The selection of a proper number of hidden units for FNNs is a very important issue in practical applications, and it has been widely discussed in the literature. In terms of the bias/variance decomposition, as the number of hidden units of an FNN grows, bias decreases and variance increases [9]. This happens because the flexibility of the model also grows with the number of hidden units [3].

Sequential algorithms (also known as constructive or incremental) for FNNs allow to dynamically construct the network, without setting *a priori* the architecture [18]. These methods start with a small network (usually with no hidden units), and sequentially add hidden units until a satisfactory solution is found. Although it cannot be guaranteed that the solution obtained is minimal, it is expected to have a few number of terms.

Pruning methods are an alternative to sequential ones [28]. Pruning algorithms work roughly as follows. A larger than needed network is trained until an acceptable solution is found. Subsequently, some hidden units or weights are removed (pruned) if they are considered useless. Then, the network is retrained, and the process starts again.

The sequential approach, however, presents a number of advantages over the pruning one [18]. First, specifying the

*Corresponding author. Tel.: +34 934015613.
E-mail addresses: eromero@lsi.upc.edu (E. Romero), alquezar@lsi.upc.edu (R. Alquézar).

0925-2312/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2006.05.022

initial network is straightforward. For pruning algorithms it is not clear how large the initial network should be. Second, pruning algorithms spend most of the time training networks larger than necessary. Sequential algorithms always search for small solutions first. Therefore, sequential methods are more likely to obtain smaller networks with less computational cost than pruning ones.

In practice, sequential approximations have been found to be very competitive in different tasks, such as speech recognition [24], voltage and current fault detection [20], image compression [22], or handwritten digit recognition [31]. Experiments on well-known benchmark problems for classification and regression tasks have also shown the good properties of sequential methods.

However, the selection of weights of the new hidden units for sequential FNNs minimizing the sum-of-squares error function usually involves a non-linear optimization problem that cannot be solved analytically in the general case. A suboptimal solution is searched heuristically. In this work, an experimental study of some of these heuristics is presented, both for classification and regression tasks.

We will focus on sequential approximations that obtain the new hidden-layer weights while keeping the previously selected ones fixed. That is, the previously obtained hidden-layer weights are frozen in order to obtain the new ones [19]. Most models found in the literature choose the hidden-layer weights of the new unit so that its output vector matches the previous residue as best as possible [8,11,15,24]. After the selection of the hidden-layer weights of the new unit, the output-layer weights can be either optimized (in a least-squares sense) or not. Several alternative approaches to the idea of matching the residue perform an (implicit or explicit) orthogonalization of the output vectors of the hidden units, so that the new hidden-layer weights are selected taking into account the interactions with the previously selected ones in order to minimize the global error [5,29,31]. In this case, the output-layer weights are always optimized. Theoretical results show that these two approaches may construct a sequence of output functions convergent to the target one. However, although the non-linear optimization problem posed at every step is, in theory, easier to solve than the optimization problem for a non-sequential procedure, it cannot be solved analytically in the general case [14], and a suboptimal solution is searched heuristically. The difficulty lies on the selection of the hidden-layer weights, since the optimal output-layer weights can be computed analytically (they can be obtained by solving a linear equations system, since output units are linear) [1].

An experimental study of the aforementioned approaches to select the weights for sequential FNNs is presented. To the best of our knowledge, this is the first time that this assessment is carried out so exhaustively. Other related works are more focused on the comparison with alternative models [31] or are less complete than the current study [19,29].

The results of our experiments indicate that the orthogonalization of the output vectors of the hidden units (taking into account the interactions with the previously selected hidden-layer weights) outperforms the strategy of matching the residue, both for approximation and generalization purposes. For noise-free data sets, the orthogonalization of the output vectors of the hidden units allows to obtain better approximations than the strategy of matching the residue with the same number of hidden units and a moderate increase in the execution times. For noisy data sets, the models obtained with the orthogonalization of the output vectors have less hidden units than those obtained matching the residue, yielding to better performance with lower execution times. These results are in agreement with the bias/variance decomposition, since the same level of approximation is achieved with less hidden units (that is, simpler models with the same bias are obtained), yielding better generalization performance.

The rest of the paper is organized as follows. Several approaches to obtain the weights in sequential FNNs are discussed in Section 2. Section 3 is devoted to describe the sequential models tested. The experiments can be found in Section 4. Finally, Section 5 concludes and outlines some directions for further research.

2. Heuristics for the selection of weights in sequential FNNs

The construction of a sequential FNN can be formulated as a *state space search problem* [18], where the *state space* corresponds to the collection of functions that can be obtained by the model. The *initial state* is usually a network with no hidden units. The *evaluation criterion* is, most of the times, an estimation of the network performance and the *termination of the search* happens either when the performance of the model is satisfactory or it begins to deteriorate. The *search strategy* determines how the model evolves during the construction. This is the most interesting part of the algorithm, and includes the number of hidden units added at every step (usually set to one), the choice of the activation function for the new hidden units (usually predefined *a priori*) and the selection of the hidden-layer weights for the new hidden unit, together with the adjustment of the weights of the whole network. The rest of the section is devoted to a detailed description of the most common approaches found in the literature for the selection of weights in sequential FNNs.

Most of the sequential algorithms for FNNs keep the previously selected hidden-layer weights fixed and search only for the hidden-layer weights of the new hidden unit and for the output-layer weights of the whole network. Some exceptions to this approach are the dynamic node creation [2] and resource-allocating network [26] schemes, where a new hidden unit is added during the training procedure if the current network does not show a certain desired behavior. The disadvantage of weight freezing is that each optimization step is not optimal, and this can result in larger networks than those in which all the weights

are optimized. In contrast, the difficulty and the computational cost of solving the underlying optimization problem is reduced (see [16,18,19]).

The general structure of sequential schemes that keep the previously selected hidden-layer weights fixed is shown in Algorithm 1. Given a data set $D = \{x_1, \ldots, x_L\}$ with L patterns, the objective is to minimize the sum-of-squares error function $\|f - X_N\|_2^2$, where $f = (f_1, \ldots, f_L)$ is the target vector and $\|g\|$ stands for the 2-norm. The output X_N is a vector in $\mathbb{R}^L$ which can be defined as (output units are linear)

$$X_N = \sum_{k=1}^{N-1} \lambda_k^N h_{\omega_k} + \lambda_N^N h_{\omega_N}, \tag{2}$$

where N is the number of hidden units, $h_{\omega_k} = (\varphi_k(\omega_k, x_1, b_k), \ldots, \varphi_k(\omega_k, x_L, b_k))$ is the output vector of the kth hidden unit (with hidden-layer weights ω_k), λ_k^N are the output-layer weights of that hidden unit at step N, and $X_0 = 0$. The hidden-layer weights ω_N of the new hidden unit are obtained by optimizing a certain objective function, which somehow depends on the previously obtained hidden-layer weights. In some cases, the output-layer weights are independent of the selection of the hidden-layer weights, and they can be computed after the hidden-layer weights have been already selected. In other ones, the output-layer weights are involved in the process of the selection of the hidden-layer weights, as explained in the rest of the section.

Algorithm 1. A general algorithm for sequential FNNs that keeps the previously selected hidden-layer weights fixed and searches only for the hidden-layer weights of the new unit and the output-layer weights of the whole network.

Algorithm
Initialize $X_0 = 0$
repeat
Increase by 1 the number of hidden units N
Pick an activation function φ_N for the new hidden unit
Obtain the hidden-layer weights ω_N of the new hidden unit
Fix the hidden-layer weights in the network
Compute/Adjust the output-layer weights $\{\lambda_k^N\}_{k=1}^N$ to obtain X_N
until a certain stopping criterion is satisfied
end Algorithm

2.1. Matching the previous residue

Many sequential schemes choose the new hidden-layer weights ω_N so that h_{ω_N} matches the previous residue as best as possible [19], that is

$$(\omega_N, \lambda_N^N) = \arg\min_{(\omega,\lambda)} \|(f - X_{N-1}) - \lambda h_\omega\|_2^2. \tag{3}$$

By inner product properties, this is equivalent to say that

$$\omega_N = \arg\max_{\omega} |\langle f - X_{N-1}, h_\omega\rangle|^2 / \|h_\omega\|_2^2, \tag{4}$$

where $\langle\cdot,\cdot\rangle$ is the dot product in $\mathbb{R}^L$ and h_ω is the output vector of a hidden unit with hidden-layer weights ω. Equivalently, the desired hidden-layer weights maximize the Fourier transform of the residue at every step. After the selection of the hidden-layer weights of every new hidden unit, either its (optimal) output-layer weight is computed or the output-layer weights of the whole network are optimized. Both computations can be made using inner product properties:

(1) When only the (optimal) output-layer weight of the new hidden unit (with hidden-layer weights ω_N) is computed, we have

$$\lambda_N^N = \langle f - X_{N-1}, h_{\omega_N}\rangle / \|h_{\omega_N}\|_2^2. \tag{5}$$

The output-layer weights of the rest of the units remain unchanged, that is, $\lambda_1^N = \lambda_1^{N-1}, \ldots, \lambda_{N-1}^N = \lambda_{N-1}^{N-1}$. In this case, it can be proved that (see, for example, [29])

$$\|f - X_N\|_2^2 = \|f - X_{N-1}\|_2^2 - \frac{|\langle f - X_{N-1}, h_{\omega_N}\rangle|^2}{\|h_{\omega_N}\|_2^2}. \tag{6}$$

(2) When the output-layer weights of the whole network are optimized, they can be obtained by solving a linear equations system (output units are linear) [1]:

$$A_N \cdot (\lambda_1^N, \ldots, \lambda_{N-1}^N, \lambda_N^N)^t = (\langle f, h_{\omega_1}\rangle, \ldots, \langle f, h_{\omega_{N-1}}\rangle, \langle f, h_{\omega_N}\rangle)^t, \tag{7}$$

where $A_N[i,j] = \langle h_{\omega_i}, h_{\omega_j}\rangle$ for $1 \leqslant i,j \leqslant N$. In this case, it can be proved that

$$\|f - X_N\|_2^2 = \|f\|_2^2 - \|X_N\|_2^2. \tag{8}$$

Some models sharing these underlying ideas are projection pursuit regression [8] originally described in the statistics field, matching pursuit [24] in the context of signal processing, or projection pursuit learning network [11] in the neural networks framework. Similarly, and also depending on the previous residue, the hidden-layer weights obtained by the cascade-correlation algorithm ([6,23,30], see also [19]) maximize the correlation between the new hidden unit and the residual error. Sometimes, convex approximations are constructed, as in the incremental linear quasi-parallel algorithm [15] (see also [10,32]).

2.2. Interacting hidden-layer weights

Some exceptions to the idea of matching the residue are the orthogonal least squares learning algorithm [5], kernel matching pursuit with *pre-fitting* [31] and the sequential approximation with optimal coefficients and interacting frequencies [29], where an (implicit or explicit)

orthogonalization of the output vectors of the hidden units is performed. The hidden-layer weights ω_N are selected taking into account the interactions of h_{ω_N} with $h_{\omega_1},\ldots,h_{\omega_{N-1}}$ in order to minimize $\|f-X_N\|_2^2$. The interactions are discovered by means of the optimal output-layer weights $\{\lambda_k^N\}_{k=1}^N$, which can be computed with (7). In other words, ω_N' is considered better than ω_N' if h_{ω_N} allows, together with the previously selected hidden-layer weights (and after computing the optimal output-layer weights of the whole network), a better approximation of f than $h_{\omega_N'}$. There is no explicit intention to match the residue.

3. Sequential models tested

In the following, we will refer to *MPR*, *OCMPR* and *SAOCIF* as:

- Matching the previous residue (*MPR*): The new hidden-layer weights ω_N are selected according to (4). The output-layer weights of the whole network are not optimized. That is, $X_N = X_{N-1} + \lambda_N^N h_{\omega_N}$, with λ_N^N computed according to (5).
- Optimal coefficients after matching the previous residue (*OCMPR*): The new hidden-layer weights ω_N are selected according to (4). Subsequently, the output-layer weights of the whole network are optimized by solving their associated linear equations system (7).
- Sequential approximation with optimal coefficients and interacting frequencies (*SAOCIF*): The new hidden-layer weights ω_N are selected taking into account the interactions of h_{ω_N} with $h_{\omega_1},\ldots,h_{\omega_{N-1}}$ in order to minimize $\|f-X_N\|_2^2$. The interactions are discovered by means of the optimal output-layer weights, obtained with (7).

As it is well-known, an approximation scheme is said to have the convergence property if it is able to produce a sequence of output functions convergent to the target. *MPR*, *OCMPR* and *SAOCIF* have been proved to have the convergence property (see [13,24] for *MPR*, [15,19] for *OCMPR* and [29] for *SAOCIF*, for example). Unfortunately, these theoretical results cannot be directly applied in practice. The optimization problem posed in the general case cannot be solved analytically, and a suboptimal solution is searched heuristically. Note that the difficulty lies on the selection of the hidden-layer weights, since the output-layer weights can always be computed analytically (they can be obtained by solving a linear equations system, since output units are linear) [1].

An experimental study of *MPR*, *OCMPR* and *SAOCIF* is presented. To the best of our knowledge, this is the first time that this type of study is carried out so exhaustively. Other related works are more focused on the comparison with alternative models [31] or are less complete than the current study [19,29].

In order to compare their respective heuristics, algorithms for *MPR*, *OCMPR* and *SAOCIF* were designed (see Algorithm 2) as particular cases of that in Algorithm 1. Both the activation function and the weights are selected in Algorithm 2. Regarding the activation function, it can be chosen, for example, from a finite list of functions fixed *a priori* (see Section 4). Regarding the weights, a number of hidden-layer candidate weights are assigned to the new hidden unit at every step. Every hidden-layer candidate weights are tested according to the objective function of every model.

Algorithm 2. Algorithms for *MPR*, *OCMPR* and *SAOCIF*.

Algorithm
$X_0 = 0$
repeat
Increase by 1 the number of hidden units N
for every activation function φ in a finite list
Assign φ to the new hidden unit
for every hidden-layer candidate weights ω
Assign ω to the new hidden unit
MPR/*OCMPR*: Following (4), set $\omega_N := \omega$ if $|\langle f - X_{N-1}, h_\omega\rangle|^2/\|h_\omega\|_2^2$ is maximized
SAOCIF: Compute the optimal output-layer weights of the whole network $\{\lambda_k^N\}_{k=1}^N$ with (7), and set $\omega_N := \omega$ if $\|f-X_N\|_2^2$ is minimized
end for
Set $\varphi_N := \varphi$ if ω_N has changed
end for
Fix the hidden-layer weights ω_N in the network
MPR: Following (5), compute its optimal output-layer weights
$\lambda_N^N = \langle f - X_{N-1}, h_{\omega_N}\rangle/\|_2^2$ for h_{ω_N}
OCMFT/*SAOCIF*: Compute the optimal output-layer weights of the whole network $\{\lambda_k^N\}_{k=1}^N$ with (7)
until a certain stopping criterion is satisfied
end Algorithm

At first sight, it could be thought that the computational cost of *SAOCIF* is very high when compared to that of *MPR* and *OCMFT*. However, *SAOCIF* satisfies a number of interesting properties that allow to implement it in an efficient fashion (see [29]).

4. Experiments

The experiments were performed with Algorithm 2. The main objective of these experiments was to compare *MPR*, *OCMPR* and *SAOCIF* (see Section 3). In particular, we studied the effect of their respective criteria of selection of weights both for approximation and generalization purposes. Execution times were also compared.

The overfitting problem has not been treated explicitly in the experiments. The main reason was that no additional distorting element was wanted to include in the comparisons. In practice, for noisy data sets, one should define a stopping criterion to stop the addition of new hidden units to the network. It could be done based on a validation set, for example. We have assumed the hypothesis that any such stopping criterion would behave similarly in the three strategies tested. Therefore, in this work we have only wanted to show the power of each strategy in a relative manner, not in absolute terms.

4.1. HEA data sets

We used the same artificial data sets described in [11], where five non-linear functions $g_n : [0,1] \times [0,1] \to \mathbb{R}$, $n \in \{1,2,3,4,5\}$, were defined as follows:

- Simple interaction function:

$$g_1(x_1, x_2) = 10.391((x_1 - 0.4)(x_2 - 0.6) + 0.36).$$

- Radial function:

$$g_2(x_1, x_2) = 24.234(r^2(0.75 - r^2)),$$

where $r^2 = (x_1 - 0.5)^2 + (x_2 - 0.5)^2$.

- Harmonic function:

$$g_3(x_1, x_2) = 42.659((2 + x_1)/20 + Re(z^5)),$$

where $z = x_1 + ix_2 - 0.5(1 + i)$.

- Additive function:

$$g_4(x_1, x_2) = 1.3356 \times (1.5(1 - x_1) + e^{2x_1 - 1}\sin(3\pi(x_1 - 0.6)^2) + e^{3(x_2 - 0.5)}\sin(4\pi(x_2 - 0.9)^2)).$$

- Complicated interaction function:

$$g_5(x_1, x_2) = 1.9(1.35 + e^{x_1}\sin(13(x_1 - 0.6)^2)e^{-x_2}\sin(7x_2)).$$

In [11], 225 pairs (x_1, x_2) of values were generated from a uniform distribution in $[0,1] \times [0,1]$. These data were used for all five functions in order to generate five noise-free training sets:

$$\text{HEA}n\text{-NF} = \{(x_1^i, x_2^i, g_n(x_1^i, x_2^i))\}_{i=1,\ldots,255},$$

where $n \in \{1,2,3,4,5\}$. In addition, another five training data sets were generated by adding independent and identically distributed Gaussian noise to the previous ones:

$$\text{HEA}n\text{-WN} = \{(x_1^i, x_2^i, g_n(x_1^i, x_2^i) + 0.25\varepsilon^i)\}_{i=1,\ldots,255},$$

where $\varepsilon^i \sim \mathcal{N}(0,1)$. The test set was built by sampling every function on a regularly spaced grid on $[0,1] \times [0,1]$ consisting of 10,000 points. In summary, 10 training sets (5 noise-free and 5 noisy versions) and 5 test sets were generated in [11] for the 5 aforementioned functions. These data sets have been widely used in the literature (see [12,17,19,23,30], for example).

4.1.1. Experimental setting

In our experiments, we constructed 10 training sets for every function, each containing 225 points, changing the initial seed of the random function for the uniform distribution (the noise-free data sets). Similar to [11], 10 noisy training data sets were generated in the same way. For every function, the test set in [11] was used as a validation set for the adjustment of the parameters in the preprocessing procedure. For the final results, a new test set was constructed, with an offset of 0.0025 with respect to the input points in the original test set. In summary, 100 training sets (50 noise-free and 50 noisy versions), 5 validation sets and 5 test sets were generated for the five aforementioned functions. For every function and version (noise-free and noisy), every model was trained with every one of the 10 different training sets, and tested on the test set.

A maximum of 50 hidden units were added to the initial architecture. The activation function was selected from a list including the hyperbolic tangent, sine, cosine and Gaussian functions, with multi-layer perceptrons units. Ten-thousand hidden-layer candidate weights in $[-0.5, +0.5] \times [-0.5, +0.5]$ were generated at random for every model. The biases were considered as part of the hidden-layer weights. Other parameters (the gain factor, for example) were chosen in the preprocessing procedure, independently for every model.

4.1.2. Results

Results are shown in Table 1 as the average of the minimum squared test set errors. Figures in boldface indicate the best results. Numbers in brackets are $\hat{\sigma}_n/\sqrt{n}$, the standard errors estimated from the sample standard deviation $\hat{\sigma}_n$.[1] The average number of hidden units where these minima were achieved is also shown. For the HEA1-NF data set, *OCMPR* and *SAOCIF* achieved a training error less than 0.000001 with less than 50 hidden units, so that it made no sense adding new hidden units.

Regarding the overall behavior, *SAOCIF* obtained better results than *OCMFT*, which in turn compared favorably with *MPR*. This fact can be understood by looking at the number of hidden units of the obtained results:

(1) For noise-free data sets the number of hidden units was close to 50, the maximum number of allowed hidden units. This is due to the fact that overfitting was not observed during the learning process with these data sets. Therefore, the best results were obtained by those models that were able to fit more accurately the data.

[1] Under normality assumptions, the confidence interval can be computed from this value. For example, the deviation of the true value from the observed mean $\overline{x_n}$ is, with probability 0.95, less than $1.96\hat{\sigma}_n/\sqrt{n}$ [7].

Table 1
Results for the HEA data sets

Data set	Test set error			Num. hidden units		
	MPR	*OCMPR*	*SAOCIF*	*MPR*	*OCMPR*	*SAOCIF*
HEA1-NF	0.06 (0.1)	**0.00 (0.0)**	**0.00 (0.0)**	49.3	20.3	9.6
HEA2-NF	29.81 (2.6)	1.18 (0.5)	**0.18 (0.1)**	49.1	47.1	44.1
HEA3-NF	694.11 (27.2)	31.41 (6.9)	**5.57 (2.2)**	49.2	45.0	49.1
HEA4-NF	69.29 (6.9)	24.32 (1.8)	**9.80 (3.7)**	49.4	45.8	47.3
HEA5-NF	49.22 (7.8)	14.79 (3.4)	**7.76 (1.0)**	49.8	49.4	46.2
HEA1-WN	14.37 (3.1)	**10.70 (1.6)**	15.37 (2.9)	39.1	4.5	4.4
HEA2-WN	110.85 (6.8)	99.13 (7.1)	**89.47 (6.3)**	30.9	19.1	15.4
HEA3-WN	704.35 (50.5)	303.65 (18.9)	**256.01 (32.6)**	49.9	33.6	31.6
HEA4-WN	213.67 (8.0)	202.91 (14.7)	**120.58 (11.5)**	47.0	27.5	17.4
HEA5-WN	174.63 (9.0)	211.82 (18.0)	**173.35 (5.7)**	35.0	23.7	19.4

Numbers in brackets are $\hat{\sigma}_n/\sqrt{n}$, the standard errors estimated from the sample standard deviation $\hat{\sigma}_n$.

The scheme of selection of hidden-layer weights of *SAOCIF* allows to find better approximations with the same number of hidden units as *OCMPR* or *MPR*. The same happens when *OCMPR* is compared with *MPR*.

(2) For noisy data sets, there was a high correlation between the number of hidden units and the goodness of the model: those models that attain their minima with less hidden units usually obtain better results. *SAOCIF* obtained simpler models (in terms of the number of hidden units), with the same training error, than *OCMFT*. According to the bias/variance trade-off, the minimum test set error should be smaller for *SAOCIF* than for *OCMFT*, as observed. The same happens with respect to *MPR*.

As an example, Fig. 1 shows the evolution of the average training and test errors of *OCMPR* and *SAOCIF* with respect to the number of hidden units for the HEA4 data sets, where the aforementioned behaviors can be observed. The interacting hidden-layer weights selected by *SAOCIF* show a better behavior than hidden-layer weights selected so as to match the previous residue, both for approximation and generalization purposes. Note that the differences among methods are greater for noise-free data sets than for noisy ones.

The execution times needed to obtain the final solutions were, as expected, highly correlated with the number of hidden units of the resulting models. For noise-free data sets, *MPR* was faster than *OCMFT*, which was in turn faster than *SAOCIF*: taking the mean execution time for *MPR* as 1, the relative execution times for *OCMPR* and *SAOCIF* were 1.02 and 1.57, respectively. Therefore, a better performance was obtained with a moderate increase in the execution times. For noisy data sets, in contrast, *SAOCIF* was the fastest method: taking the mean execution time for *SAOCIF* as 1, the relative execution times for *OCMPR* and *MPR* were 1.10 and 2.01, respectively. In this case, the larger computational cost of every step for *SAOCIF* was compensated with the lower number of steps (number of hidden units) in the obtained solutions.

4.2. The two spirals data set

The well-known *Two spirals* problem consists in identifying the points of two interlocking spirals. Both training and test set comprise 194 points with balanced classes. The training set is symmetric with respect to the target (i.e., if (x, y) belongs to a class, the training set also contains the point $(-x, -y)$, which belongs to the other class). The test set is not symmetric, and it is obtained adding an offset to the points in the training set.

4.2.1. Experimental setting

This problem was tested with *MPR*, *OCMPR* and *SAOCIF* for the logistic activation function. Two-hundred hidden-layer candidate weights were generated at random for every model, within a certain range of weights. The maximum number of hidden units added was 500, and no more hidden units were added once the whole training set was learned. The sign of the output value indicates the class assigned to the input pattern.

4.2.2. Results

Results are shown in Table 2 as the average of 10 runs of the algorithms. Column 'Train' indicates the percentage of the training set which has been learned. Column 'Test' indicates the generalization performance obtained by an average-output committee of the resulting networks. As already known, this is a very hard problem for the logistic function because of its intrinsic high non-linearity and radial symmetry, but it could be learned by *OCMPR* and *SAOCIF* and an adequate (and very large) range of weights. *MPR* could not solve the problem with the same ranges, at least with 500 hidden units. Generalizations in the obtained solutions was very good.

In these experiments, *OCMPR* took less execution time than *SAOCIF*, which in turn was faster than *MPR*. The

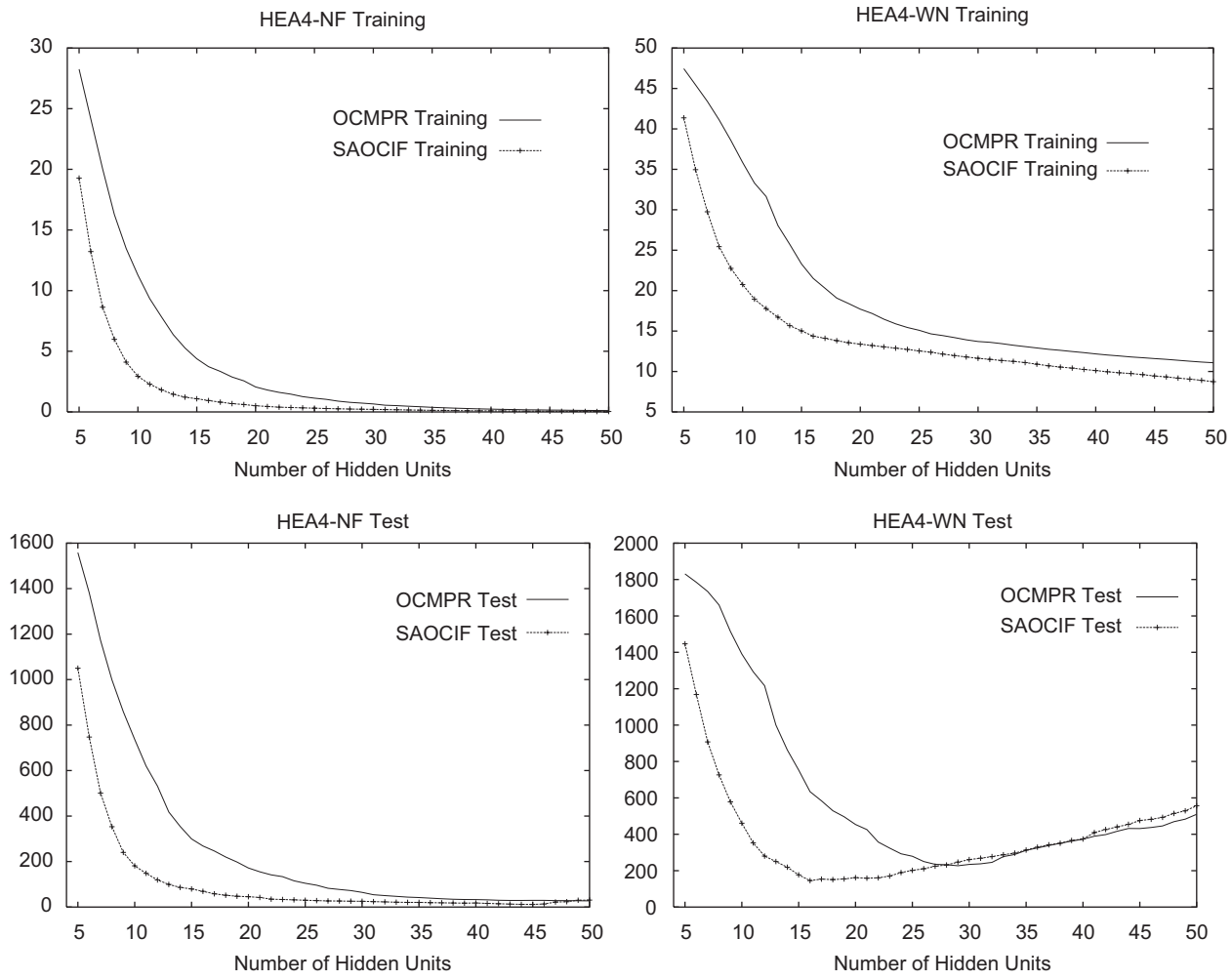

Fig. 1. Evolution of the average training and test error of *OCMPR* and *SAOCIF* with respect to the number of hidden units for the HEA4-NF (left) and HEA4-WN (right) data sets.

Table 2
Results for the *Two spirals* data set

Method	WRange	Training %	Test %	NHid
MPR	$[-16,+16]$	98.97	99.48	500 (0.00)
MPR	$[-8,+8]$	—	—	NP
OCMFT	$[-8,+8]$	100	100	105.2 (3.52)
SAOCIF	$[-8,+8]$	100	100	91.7 (2.33)

Columns 'WRange' and 'NHid' indicate the range within which hidden-layer candidate weights are selected and the average number of hidden units in the resulting networks, respectively. Numbers in brackets are $\hat{\sigma}_n/\sqrt{n}$, the standard errors estimated from the sample standard deviation $\hat{\sigma}_n$.

execution time for *MPR* was very high because of the large number of hidden units required. Taking the mean execution time for *OCMPR* as 1, the relative execution times for *SAOCIF* and *MPR* in these experiments were 1.52 and 10.98, respectively. However, the number of hidden units in the obtained solutions was smaller for *SAOCIF* than for *OCMPR* and *MPR*, and it was obtained with a moderate increase in the execution times.

4.3. Real world data sets

Real world benchmark data sets from the UCI repository [4] were also used to compare *MPR*, *OCMPR* and *SAOCIF*. These data sets, namely *Hepatitis* (19 input variables, 155 examples) and *Ionosphere* (33 input variables, 351 examples), have a high input dimension with respect to the number of examples, and they have been widely used in the literature.

4.3.1. Experimental setting

A maximum of 100 hidden units were added to the initial architecture. In these experiments, the networks constructed by *MPR*, *OCMPR* and *SAOCIF* computed the hyperbolic tangent activation function in their hidden units (all the hidden units had the same activation function). Hidden-layer candidate weights were selected from the data set, as in [5,29,31] in a deterministic manner: for every hidden unit to be added, every point in the training set was tested as hidden-layer candidate weights. The biases were computed in a heuristic manner as follows: the bias of the

Table 3
Results for the real world data sets

Data set	Test %			Num. hidden units		
	MPR	*OCMFT*	*SAOCIF*	*MPR*	*OCMPR*	*SAOCIF*
Hepatitis	85.42 (0.70)	88.90 (1.12)	**89.55 (1.07)**	76.8	19.2	12.2
Ionosphere	89.77 (0.58)	91.54 (0.74)	**92.23 (0.62)**	82.5	20.2	18.9

Numbers in brackets are $\hat{\sigma}_n/\sqrt{n}$, the standard errors estimated from the sample standard deviation $\hat{\sigma}_n$.

new hidden unit is such that, if the rest of the hidden-layer weights are 0, the output of the hidden unit is the mean value of the error obtained with the previously selected hidden units. It can be easily obtained by computing the residue and then inverting the obtained value with respect to the activation function of the hidden unit. Every bias was computed and fixed previous to the selection of the rest of the hidden-layer weights, so that all the hidden-layer candidate weights were tested with the same bias (although different for every hidden unit).

For every model, an exhaustive search process was performed in order to select its parameters (the gain factor, for example). For every parameters configuration, five repetitions of a 5-fold cross-validation were performed. Previous to every cross-validation, the examples in the data set were randomly shuffled.

4.3.2. Results

Results are shown in Table 3 as the average of the maximum test set accuracies. Figures in boldface indicate the best results. Similar to the HEA data sets, a high correlation between the number of hidden units and the goodness of the model is observed: those models that attain their optimal values with less hidden units also obtain better generalization results. In this case, however, differences are not as large as in the HEA data sets.

Similar to the *Two spirals* data set, the execution time for *MPR* was higher because of the large number of hidden units in the final solutions, and similar to the HEA noisy data sets, *SAOCIF* was the fastest method: taking the mean execution time for *SAOCIF* as 1, the relative execution times for *OCMPR* and *MPR* in these experiments were 1.06 and 5.17, respectively.

5. Conclusions and future work

This work empirically shows that the orthogonalization of the hidden vectors (interacting hidden-layer weights) outperforms the strategy of matching the residue, both for approximation and generalization purposes. The importance of the interacting hidden-layer weights lies in the hypothesis that they allow to find better partial approximations, with the same number of hidden units, than hidden-layer weights selected just to match the residue as best as possible. Likewise, the same level of approximation may be achieved with less hidden units. Therefore, in terms of the bias/variance decomposition, it is possible to obtain simpler models with the same bias.

New experiments could be designed by changing the heuristic to select the hidden-layer weights of the new unit in the tested models. It would be interesting to test whether the observed differences remain or not. The field of evolutionary algorithms offers a number of search algorithms to that end. For example, a population of hidden-layer weights could evolve driven by a breeder genetic algorithm (BGA) [25] with the squared error as the fitness function.

A different point of view could be introduced in the construction of FNNs if we reconsidered the goodness of the previously selected hidden-layer weights. Comparing the addition of hidden units in a sequential FNN with the selection of features in the sequential forward selection procedure for feature selection [21], we can see that they share the same general principles, although applied to different objects. Whereas sequential FNNs applies the forward selection to the hidden units, sequential forward selection applies it to the features. Therefore, other feature selection search procedures can be applied to the construction of FNNs. In particular, floating methods [27] may help to obtain more compact networks. Note that the criterion to remove a previously selected hidden unit, needed in floating algorithms, cannot be directly based on the approximation of a previous residue, and it should be redefined. In contrast, the same idea used by *SAOCIF* to add a new term can be used to remove an old one: the hidden unit such that, when removed allows to obtain (after computing the optimal output-layer weights of the resulting network) the smallest error.

Acknowledgments

This work has been supported by the Consejo Interministerial de Ciencia y Tecnología (CICYT), under project CGL2004-04702-C02-02.

References

[1] N.I. Achieser, Theory of Approximation, Frederick Ungar Pub. Co., New York, 1956.
[2] T. Ash, Dynamic node creation in backpropagation networks, Connection Sci. 1 (4) (1989) 365–375.
[3] A.R. Barron, Approximation and estimation bounds for artificial neural networks, Mach. Learn. 14 (1) (1994) 115–133.

[4] C.L. Blake, C.J. Merz, UCI repository of machine learning databases, University of California, Irvine, Department of Information and Computer Science, ⟨http://www.ics.uci.edu/~mlearn/MLRepository.html⟩, 1998.
[5] S. Chen, C.F.N. Cowan, P.M. Grant, Orthogonal least squares learning algorithm for radial basis function networks, IEEE Trans. Neural Networks 2 (2) (1991) 302–309.
[6] S.E. Fahlman, C. Lebiere, The cascade-correlation learning architecture, in: Advances in Neural Information Processing Systems, vol. 2, Morgan Kaufmann, 1990, pp. 524–532.
[7] A. Flexer, Statistical evaluation of neural network experiments: minimum requirements and current practice, in: 13th European Meeting on Cybernetics and System Research, vol. 2, 1996, pp. 1005–1008.
[8] J.H. Friedman, W. Stuetzle, Projection pursuit regression, J. Am. Stat. Assoc. 76 (1981) 817–823.
[9] S. Geman, E. Bienenstock, R. Doursat, Neural networks and the bias/variance dilemma, Neural Comput. 4 (1) (1992) 1–58.
[10] K. Hlaváčková, M.M. Fischer, An incremental algorithm for parallel training of the size and the weights in a feedforward neural network, Neural Process. Lett. 11 (2) (2000) 131–138.
[11] J.N. Hwang, S.R. Ray, M. Maechler, D. Martin, J. Schimert, Regression modelling in back-propagation and projection pursuit learning, IEEE Trans. Neural Networks 5 (3) (1994) 342–353.
[12] J.N. Hwang, S.S. You, S.R. Lay, I.C. Jou, The cascade-correlation learning: a projection pursuit learning perspective, IEEE Trans. Neural Networks 7 (2) (1996) 278–289.
[13] L.K. Jones, On a conjecture of huber concerning the convergence of projection pursuit regression, Ann. Stat. 15 (2) (1987) 880–882.
[14] V. Kůrková, Incremental approximation by neural networks, in: M. Karny, K. Warwick, V. Kůrková (Eds.), Dealing with Complexity: A Neural Network Approach, Springer, London, 1998, pp. 177–188.
[15] V. Kůrková, B. Beliczyński, Incremental approximation by one-hidden-layer neural networks, in: International Conference on Artificial Neural Networks, vol. 1, 1995, pp. 505–510.
[16] T.Y. Kwok, D.Y. Yeung, Experimental analysis of input weight freezing in constructive neural networks, in: IEEE International Conference on Neural Networks, 1993, pp. 511–516.
[17] T.Y. Kwok, D.Y. Yeung, Use of bias term in projection pursuit learning improves approximation and convergence properties, IEEE Trans. Neural Networks 7 (5) (1996) 1168–1183.
[18] T.Y. Kwok, D.Y. Yeung, Constructive algorithms for structure learning in feedforward neural networks for regression problems, IEEE Trans. Neural Networks 8 (3) (1997) 630–645.
[19] T.Y. Kwok, D.Y. Yeung, Objective functions for training new hidden units in constructive neural networks, IEEE Trans. Neural Networks 8 (5) (1997) 1131–1148.
[20] W.M. Lin, C.D. Yang, J.H. Lin, M.T. Tsay, A fault classification method by RBF neural network with OLS learning procedure, IEEE Trans. Power Delivery 16 (4) (2001) 473–477.
[21] H. Liu, H. Motoda, Feature Selection for Knowledge Discovery and Data Mining, Kluwer Academic Publishers, Dordrecht, 1998.
[22] L. Ma, K. Khorasani, Application of adaptive constructive neural networks to image compression, IEEE Trans. Neural Networks 13 (5) (2002) 1112–1126.
[23] L. Ma, K. Khorasani, New training strategies for constructive neural networks with application to regression problems, Neural Networks 17 (4) (2004) 589–609.
[24] S.G. Mallat, Z. Zhang, Matching pursuits with time-frequency dictionaries, IEEE Trans. Signal Process. 41 (12) (1993) 3397–3415.
[25] H. Mühlenbein, D. Schlierkamp-Voosen, Predictive models for the breeder genetic algorithm I. Continuous parameter optimization, Evol. Comput. 1 (1) (1993) 25–49.
[26] J. Platt, A resource-allocating network for function interpolation, Neural Comput. 3 (2) (1991) 213–225.
[27] P. Pudil, J. Novovičová, J. Kittler, Floating search methods in feature selection, Pattern Recognition Lett. 15 (11) (1994) 1119–1125.
[28] R. Reed, Pruning algorithms—A survey, IEEE Trans. Neural Networks 4 (5) (1993) 740–747.
[29] E. Romero, R. Alquézar, A sequential algorithm for feed-forward neural networks with optimal coefficients and interacting frequencies, Neurocomputing 69 (13–15) (2006) 1540–1552.
[30] N.K. Treadgold, T.D. Gedeon, Exploring constructive cascade networks, IEEE Trans. Neural Networks 10 (6) (1999) 1335–1350.
[31] P. Vincent, Y. Bengio, Kernel matching pursuit, Machine Learning 48 (1–3) (2002) 165–187 (Special issue on New Methods for Model Combination and Model Selection).
[32] T. Zhang, A general greedy approximation algorithm with applications, in: Advances in Neural Information Processing Systems, vol. 14, MIT Press, 2002, pp. 1065–1072.

Enrique Romero was born in Barcelona, Spain, in 1966. He received the licenciate degree in Mathematics in 1989 from the Universitat Autònoma de Barcelona. In 1994, he received the licenciate degree in Computer Science from the Universitat Politècnica de Catalunya (UPC). In 1996, he joined the Department of Llenguatges i Sistemes Informàtics, UPC, as an assistant professor. He received the M.Sc. degree in Artificial Intelligence and the Ph.D. degree in Computer Science from the UPC in 2000 and 2004, respectively. His research interests include Neural Networks, Support Vector Machines and Feature Selection.

René Alquézar was born in Barcelona, Spain, in 1962. He received the licentiate and Ph.D. degrees in Computer Science from the Universitat Politècnica de Catalunya (UPC), Barcelona, in 1986 and 1997, respectively. From 1987 to 1991 he was with the Spanish company NTE as technical manager of R&D projects on image processing applications for the European Space Agency (ESA). From 1991 to 1994 he was with the Institut de Cibernètica, Barcelona, holding a predoctoral research grant from the Government of Catalonia and completing the doctoral courses of the UPC on Artificial Intelligence. He joined the Department of Llenguatges i Sistemes Informàtics, UPC, in 1994 as an assistant professor, and since March 2001 he has been an associate professor in the same department. His current research interests include Neural Networks, Structural and Syntactic Pattern Recognition and Computer vision.