International Journal of Pattern Recognition and Artificial Intelligence
© World Scientific Publishing Company

On different model selection criteria in a Forward and Backward regression hybrid network*

Shimon Cohen Nathan Intrator
School of Computer Science
Tel Aviv University
Tel Aviv 69978, ISRAEL
shimon-c@orbotech.com, nin@cs.tau.ac.il
www.cs.tau.ac.il/∼nin

An assessment of the performance hybrid network with different model selection criteria is considered. These criteria are used in an automatic model selection algorithm for constructing a hybrid network of radial and Perceptron hidden units for regression. A forward step builds the full hybrid network; A model selection criterion is used for controlling the network-size and another criterion is used for choosing the appropriate hidden unit for different regions of input space. This is followed by a conservative pruning step using Likelihood Ratio Test, Bayesian or Minimum Description Length, which leads to robust estimators with low variance. The result is a small architecture that performs well on difficult, realistic, benchmark data-sets of high dimensionality and small training size. Best results are obtained by using the Bayesian approach for the model selection.

Keywords: Projection units; RBF Units; Hybrid Network Architecture; SMLP; Clustering; Regularization.

1. Introduction

The construction of a network architecture that contains units of different types in the same hidden layer is not commonly done. One reason is that such construction makes model selection more challenging, as it requires the determination of each unit type in addition to the determination of network size. A more common approach to achieving higher architecture flexibility is via the use of more flexible units [30,17]. The potential problem of such a construction is over flexibility which leads to over-fitting.

We have introduced a training methodology for a hybrid MLP/RBF network [6,4,7]. This architecture produced better classification and regression results when compared with advanced RBF methods or with MLP architectures. In this paper, we introduce a novel model selection approach using the Minimum Description

*This work was partially supported by the Israeli Ministry of Science and by the Israel Academy of Sciences and Humanities – Center of Excellence Program. Part of this work was done while N. I. was affiliated with the Institute for Brain and Neural Systems at Brown University and supported in part by ONR grants N00014-98-1-0663 and N00014-99-1-0009.

Length (MDL) principle. We compare this approach with previous approaches: Likelihood Ratio Test and the Bayesian approach [7].

2. Motivation for incremental methods and the use of a hybrid MLP/RBF networks

There are many ways to decompose a function into a set of basis functions. The challenging task is to use a complete set which converges fast to the desired function (chosen from a sufficiently wide family of functions). For example, while it is well known that MLP with as little as a single hidden layer is a universal approximator, namely, it can approximate any L^2 function, it is also known that the approximation may be greedy, with the number of hidden units growing large as a function of the desired approximation error [1]. When the number of training patterns in the Training Sample Set (TSS) is low, this approximation introduces a large variance and the prediction of the regressor becomes unreliable.

Analyzing cases where convergence of the architecture (as a function of number of hidden units) is slow, reveals that often there is at least one region of input space where an attempt is being made to approximate a function that is radially symmetric (such as a doughnut) with projection units or vice versa [11,17]. This suggests that an incremental architecture which chooses the appropriate hidden unit for different regions of input space can lead to a far smaller architecture. Earlier approaches, attempted to construct a small network approximation to the desired function at different regions of input space. This approach, which was called “divide and conquer”, has been studied since the eighties in the machine learning and connectionists community. Rather than reviewing the vast literature on that approach, we shall point out some approaches which indicate some of the highlights that have motivated our work. Work on trees is reviewed in [3] where the goal is to reach a good division of the input space and use a simple architecture at the terminating nodes. That work suggested some criteria for splitting the input space and provided a cost complexity method for comparing the performance of architectures with different size. An approach which constructs more sophisticated architectures at the terminating nodes was proposed in [35,26], where a gating network performs the division of the input space and small networks perform the function approximation at each region separately. Nowlan’s [35] many experiments with such architecture led him to the conclusion that it is better to have different type of architectures for the gating network and for the networks that perform the function approximation at the different regions. He suggested using RBF for the gating network and MLP for the function approximation, and thus constructed the first hybrid architecture between MLP and RBF. A tree approach with artificial neural networks as terminating nodes was proposed by [27]. The boosting algorithm [12] is another variant of the space division approach, where the division is done based on the error performance of the given architecture. In contrast to previous work, this approach takes into account the geometric structure of the input data indirectly. A remote family of architectures, where the function approximation is constructed incrementally, is projection pursuit [19] and additive models [22,23]. Another approach is taken by the cascade correlation algorithm, which incrementally adds hidden units to meet a given error goal [15].

If one accepts the idea of constructing a local simple architecture to different regions in input space, then the question becomes, which architecture family should be used. The local architecture should be as simple as possible in order to avoid over-fitting to the smaller portion of regional training data. Motivated by theoretical work that has studied the duality between projection-based approximation and radial kernel methods [11], we have decided to use RBF or Perceptron units. Donoho's work has shown that a function can be decomposed into two parts, the radial part and the ridge (projection based) part, and that the two parts are mutually exclusive. It is difficult however, to separate the radial portion of a function from its projection-based portion before they are estimated, but a sequential approach which decides on the fly, which unit to use for different regions in input space, has a potential to find a useful subdivision.

The most relevant statistical framework to our proposal is Generalized Additive Models (GAM) [22,23]. In that framework, the hidden units (the components of the additive model) have some parametric form, usually polynomial, which is estimated from the data. While this model has nice statistical properties [46], the additional degrees of freedom, require strong regularization to avoid over-fitting. Higher order networks have at least one quadratic term in addition to the linear term of the projections [30] as a special case of GAM:

$$y = \sum_i w_i g(\sum_j w_{ij} x_j + \sum_k \sum_l w_{ikl} x_k x_l + a_i) + w_0. \tag{1}$$

While they present a powerful extension of MLPs, and can form local or global features, they do so at the cost of squaring the number of input weights to the hidden nodes. Flake [17] has suggested an architecture similar to GAM where each hidden unit has a parametric activation function which can change from a projection based to a radial function in a continuous way [17]. This architecture uses a squared activation function, thus called Squared MLP (SMLP), and only doubles the input dimension of the input patterns.

Our proposed hybrid architecture extends both MLP and RBF networks, by combining RBF and Perceptron units in the same hidden layer. Unlike the previously described methods, this does not increase the number of parameters in the model, at the cost of predetermining the number of RBF and Perceptron units in the network. The hybrid network is useful especially in cases where the data includes some regions that contain hill-plateau and other regions that contain Gaussian bumps, this is further discussed in [6]. The hybrid architecture [4,6,7], which we call Perceptron Radial Basis Net (PRBFN), automatically finds the relevant functional parts from the data concurrently, thus avoiding possible local minima that result from sequential methods. There are several approaches to set the structure of an

Artificial Neural Network. The forward selection approach starts with a small [33,16] network and adds units until an error goal is reached. Another approach is to start with a large network and prune unnecessary units, until a given criteria is met [14,47]. We introduced a combination of these approaches in [7]. This algorithm starts with a small network and expands the network until a given error goal is met, namely, it determines automatically the size of the network. A difficult task in training a hybrid artificial neural network is to find the radial and projection parts automatically [7]. This problem is amplified for high dimensional data, where the data cannot be visualized very well. We proposed a novel way to select the type of hidden unit automatically. After training the network, we proposed to prune unnecessary weights using Bayesian techniques or LRT. The Forward Backward Model Selection (FBMS) algorithm leads to smaller networks while maintaining good generalization of the resulting network. The importance of a small architecture with strong regularization is revealed when there are few samples in the Training Sample Set (TSS), and the given function is corrupted with a large amount of noise [7]. Weight decay can be used to reduce the overfitting in these cases [36,29]. Another approach, which helps to reduce overfitting, is weight elimination or pruning of parameters [50,21,7]. In this work, we evaluate different model selection criteria for selecting the number of hidden units and for pruning. Using the Minimum Description Length (MDL) principle [40,41,43], a novel model selection criterion is derived. We use several data sets to assess the performance of the hybrid network and the MDL criterion. In addition, we consider eight new data sets for function approximation, and compare our method to the Generalized Radial Basis Function (GRBF) algorithm that is trained using the soft-EM algorithm [31].

3. Parameter estimation and model selection

For completeness, we give here a short description of the FBMS algorithm [7]. Here are the main steps of this algorithm: (i) find the next region in input space where a hidden unit might be needed, (ii) decide which unit to add, a RBF or a Perceptron, (iii) train the network, and (iv) prune unnecessary weights.

The SMLP [17] network uses both RBF and Perceptron units at each cluster. In higher order networks [30], quadratic and linear terms always exists and strong regularization must be used to avoid over-fitting. We prefer to attempt to select the proper unit for each region in input space. Thus, the number of hidden units is minimal and over-fitting is reduced. In order to select the type of units in a high dimensional space, one has to divide the space into regions and then select the type of hidden unit for each region. During the division of the space into small regions, we can estimate the overall error and stop the splitting process when an error goal is reached. Thus, monitoring the size of the network as well. After the initial parameter estimation, there may be many unnecessary parameters and when the training data-set is small an over-fitting may occur. Thus, a backward elimination of weights can improve the prediction of the algorithm and reduces the variance of the estimator.

For these reasons there are several steps in estimating the parameters and structure of the hybrid network. We outline the algorithm's steps to achieve these goals as follows:

- Data clustering and splitting to reduce an error objective function.
- Automatic selection of unit type for each sub-space.
- Full gradient descent.
- Pruning of unnecessary weights.

Full details of the above steps are given in [5,7].

3.1. *Model selection*

The method that is described in this paper uses a statistical approach to select models. Thus, we define a probability model for regression. We assume that the target function values are corrupted by Gaussian noise with zero mean and equal variance σ^2. In addition, we assume that the noise is independent. We also assume that the patterns in the training set are independent.

Thus, the likelihood of the data given the model is:

$$L = \frac{1}{(2\pi)^{\frac{N}{2}} \sigma^N} \exp\left(-\frac{\sum_{n=1}^{N} (y_n - t_n)^2}{2\sigma^2}\right), \tag{2}$$

where N is the number of examples, t is the desired response and y is the output of the regressor. The maximization of the above function is equivalent to the maximization of its log value:

$$LL = -\frac{N}{2}\log(2\pi) - N\log(\sigma) - \frac{\sum_{n=1}^{N} (y_n - t_n)^2}{2\sigma^2}. \tag{3}$$

Thus, minimization the SSE is equivalent to the maximization of the likelihood of the data. To obtain the maximum likelihood value of σ, we differentiate Eq. (3) with respect to σ:

$$\frac{\partial LL}{\partial \sigma} = -\frac{N}{\sigma} + \frac{\sum_{n=1}^{N} (y_n - t_n)^2}{\sigma^3}. \tag{4}$$

Thus, we obtain the maximum likelihood value for σ:

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{n=1}^{N} (y_n - t_n)^2. \tag{5}$$

We will use Eq. (3) when we compute the model selection criterion as described below.

Model selection is applied twice in our method. The first time is when the type of hidden unit has to be selected. The second time is when the weights are pruned and, thus, there are two nested models. The task in this case is to select the best model from a sequence of models. We use the Bayesian Information Criterion (BIC) or

MDL to select the type of unit. We use BIC, MDL or LRT for the pruning process. Next we describe these approaches.

3.1.1. *The Bayesian approach*

We start by describing the Bayesian approach. Mackay [32] used the evidence of the model for model selection. Kass and Raftery [28] utilize Bayes Factors for model selection.

Given a data-set D, the task is to choose between two (or more) models M_1, M_2. Each model has a parametric family of weights attached to it, with its prior probability $p(w|M)$. The probability of the data under each model is given by:

$$p(D|M) = \int_w p(D, w|M)dw = \int_w p(D|w, M)p(w|M)dw. \tag{6}$$

The Bayes Factors are then defined as:

$$\frac{p(M_1|D)}{p(M_2|D)} = \frac{p(D|M_1)p(M_1)}{p(D|M_2)p(M_2)}. \tag{7}$$

The integration of Eq. (6) can be performed by using Laplace integral [28,13] which approximates the integrand by a quadratic function (Taylor approximation to the second order). Thus, the value of the integral becomes:

$$p(D|M) \cong (2\pi)^d |H|^{-1|2} p(D|W_{m_0}, M)p(W_{m_0}|M), \tag{8}$$

where H is the Hessian matrix of the approximation and W_{m_0} is the most probable value of the likelihood $p(D|M)$. Another way to compute the integration of Eq. (6) is by using Monte Carlo Markov Chain (MCMC) techniques [34], which for this purpose is more computationally intensive. Note that this calculation takes into account the performance of the model in the vicinity of the parameter vector m_0 and is, thus, much more informative than a simple likelihood at m_0. With the lack of a-priori knowledge, we assume that a model with an RBF or a Perceptron as a hidden unit is equally likely, thus:

$$p(M_1) = p(M_2).$$

This leads to the integrated likelihood ratio:

$$\frac{p(D|M_1)}{p(D|M_2)}.$$

The BIC approximation can be derived from Eq. (8) by using Gaussian distribution to the a-priori parameters density [28] to arrive at:

$$BIC \equiv \log(p(D|M)) = \log(p(D, W_{m_0}|M)) - \frac{d}{2}\log(N), \tag{9}$$

where $\log(p(D, W_{m_0}|M))$ is the maximum likelihood estimation of the parameters and d are the number of parameters.

Substituting Eq. (5) and Eq. (3) into Eq. (9), we obtain:

$$BIC_i = -\frac{1}{N}\log(2\pi) - N\log(\hat{\sigma}) - \frac{N}{2} - \frac{d_i}{2}\log(N). \tag{10}$$

The first and third terms on the right are constant and can be eliminated when selecting models on the same data-set.

The truly Bayesian approach, then, uses the evidence as the weights of the models in order to get a weighted prediction. That is, when a prediction of a new value y is to be made given the training data-set D, we note that:

$$p(y|D) = \sum_{i=1}^{m} p(y, M_i|D) = \sum_{i=1}^{m} p(y|M_i, D)p(M_i|D). \tag{11}$$

Equation (11) shows that the evidence of a model can be used as weight when averaging the predictions of all models. When the best model gives good prediction the averaging process can be skipped and the best model can be used for prediction. Thus, the FBMS algorithm uses the best model for prediction.

3.1.2. *Minimum Description Length*

In the Minimum Description Length (MDL) formulation, one searches for a model that allows the shortest data encoding, together with a description of the model itself [39,42]. MDL can be formulated on an imaginary communication game, in which a sender observes the data D and communicates it to the receiver. Having observed the data, the sender discovers that the data has some regularity that can be captured by a model M. This encourages the sender to send the data using a probability that is dictated by the model, instead of sending the data with a uniform prior probability. Due to noise, there are possible errors. Both, the errors and the model, are sent so that the receiver is able to recover the data. Complex models can achieve high error correction tolerance, thus, high data accuracy, but their description is expensive. In contrast, too simple models are more sensitive to errors, thus are less accurate. Thus, there is a tradeoff between accuracy and the compactness of the code.

To transmit the data, the sender composes a message which consists of the message with the length $\ell(M)$ that specifies the model, and $\ell(D|M)$ that specifies the length of the data given the model. The goal of the sender is to find the model that minimizes the length of this encoded message $\ell(M, D)$, called the description length:

$$\ell(M, D) = \ell(D|M) + \ell(M). \tag{12}$$

According to Shannon's theory to encode a random variable X with a known distribution by the minimum number of bits, a realization of x has to be encoded by $-\log(p(x))$ bits [45,9]. Thus the description length is:

$$\ell(M, D) = -\log(p(D|M)) - \log(p(M)), \tag{13}$$

where $p(M|D)$ is the probability of the output data given the model, and $p(M)$ is $a\!-\!priori$ model probability. Typically the MDL principle is used to select the model with the shorter description length. To make larger margin between the "support vectors", one seeks to minimize the second layer of weights of the neural network [8]. A similar idea is weight decay, which helps to improve generalization [29]. Hinton and Camp (1993) used zero-mean Gaussian distribution for the neural network weights. This motivates us to define $a - priori$ small values for the network parameters. We assume normal probability model for the prior:

$$p(M) = \frac{1}{(2\pi)^{1/2}\beta^d}\exp\left(-\frac{\sum_{i=1}^d w_i^2}{2\beta^2}\right), \tag{14}$$

where d is the number of parameters in the hybrid network, β is the standard deviation and w_i are the parameters of the estimator. In addition, we assume that the errors that the model make are i.i.d with normal distribution.

Thus, the likelihood of the data given the model is:

$$p(D|M) = \frac{1}{(2\pi)^{\frac{N}{2}}\alpha^N}\exp\left(-\frac{\sum_{n=1}^N (y_n - t_n)^2}{2\alpha^2}\right), \tag{15}$$

where t_n is the target value for the nth pattern, y_n is the respected output of the expert and α is the standard deviation. Under these assumptions the description length of the model is:

$$\begin{aligned}\ell(M,D) = {} & \frac{N}{2}\log(2\pi) + N\log(\alpha) + \frac{\sum_{n=1}^N (y_n - t_n)^2}{2\alpha^2} + \\ & \frac{d}{2}\log(2\pi) + d\log(\beta) + \sum_{i=1}^d \frac{w_i^2}{2\alpha^2}.\end{aligned} \tag{16}$$

Differentiating Eq. (16) with respect to α and equating to zero, we obtain:

$$\alpha^2 = \frac{1}{N}\sum_{n=1}^N (y_n - t_n)^2. \tag{17}$$

Differentiating Eq. (16) with respect to β and equating to zero, we obtain:

$$\beta^2 = \frac{1}{d}\sum_{i=1}^d w_i^2. \tag{18}$$

Substituting Eq. (17) and Eq. (18) into Eq. (16) and discarding the constant terms, we arrive at:

$$\ell(M,D) = N\log(\alpha) + d\log(\beta) + \frac{d}{2}(1 + \log(2\pi)). \tag{19}$$

Equation (19) shows that the description length of the model is a tradeoff between the errors (α) and the number of parameters d and their average value (β).

3.1.3. *Nested models and the Likelihood Ratio Test*

The LRT can be used to select between two nested models. Given two models such that $M1 \subset M2$, the LRT test is defined as follows [37]:

$$-2\log(\frac{p(D, W_{m_0}|M1)}{p(D, W_{m_0}|M2)}) = \chi^2(d_2 - d_1). \tag{20}$$

This approach uses $P - Values$ to reject the null hypothesis. That is, the simple model is equivalent to the complicated one. Note that this approach only defines confidence intervals to the selection (that is threshold). It does not select the best possible model. The LRT criterion is applicable only when the models are nested. Thus, this process is applicable only for the pruning process.

Using the maximum likelihood estimator for σ Eq. (5), we arrive at the following test:

$$\chi^2(d_2 - d_1) = 2N\log(\sigma_1) - 2N\log(\sigma_2), \tag{21}$$

where d_1 and d_2 are the number of parameters of the first and second models.

3.2. ***Forward step***

In the forward step, the space is divided recursively into sub-spaces. While the Mean Squared Error (MSE) is typically smaller with each division, the description length of the model, or its Bayesian Information Criterion, can obtain larger or smaller values. Thus, there are several models, with different model selection criterion value. When the division process stops [7], the best model is selected, using either the BIC or the MDL criterion. The space division stops when a predefined maximum number of hidden units is achieved.

When the MDL is used, the model with the smallest description length from Eq. (19) is selected. When the Bayesian Information Criterion is used, Eq. (10), the model with the largest BIC value is selected. Thus, the number of hidden units in the network is completely defined from the model selection criteria.

3.3. ***Pruning***

The last step of the FBMS algorithm is to prune unnecessary weights. The pruning process can prune inputs weight to a ridge unit, a feature of a RBF unit or a hidden unit. We use a diagonal approximation to the covariance matrix. Thus, the activation of a RBF function is given by:

$$\phi(x) = \exp(-\sum_{i=1}^{N}\frac{(x_i - c_i)^2}{\sigma_i^2}). \tag{22}$$

If we make the substitution:

$$\frac{1}{\sigma} = r,$$

we obtain:

$$\phi(x) = \exp(-\sum_{i=1}^{d} \frac{(x_i - c_i)^2 r_i^2}{2}). \tag{23}$$

Consider setting r_j to zero:

$$\begin{aligned}\phi(x) &= \exp(-\sum_{i=1, i\neq j}^{d} \frac{(x_i - c_i)^2 r_i^2}{2}) \exp(-\frac{(x_j - c_j)^2 r_j^2}{2}) \\ &= \exp(-\sum_{i=1, i\neq j}^{d} \frac{(x_i - c_i)^2 r_i^2}{2}) \exp(0) \\ &= \exp(-\sum_{i=1, i\neq j}^{d} \frac{(x_i - c_i)^2 r_i^2}{2}). \end{aligned} \tag{24}$$

The partial derivatives with respect to r_i are:

$$\frac{\partial \phi}{\partial r_i} = -\phi(x)(x_i - c_i)^2 r_i. \tag{25}$$

Thus, if we wish to prune a feature i, we set the respected r_i to zero for the proper radial function. Note, that this eliminates two parameters. For the ridge input weights we add a matrix w_{ij} were $w_{ij} \in 0, 1$. These weights signal a pruned parameter. That is, if $w_{ij} = 0$, the weight j to projection unit i is pruned. Since the models are nested, it is possible to select the model using LRT. On the other hand, the best model can be selected by the BIC approximation to its evidence, or by the MDL principle. Our method uses both criteria when the BIC is its default criterion for model selection. The pruning process that we implement here is similar to [47,21]. We start with the full model and, at each step, we prune the least significant weight. That is, the weight that contributes least to the objective function. When the BIC criterion or the MDL principle are used, the search is exhaustive for the best model from a range of possible nested models. When the LRT criterion is used, the pruning is stopped when a given threshold is reached. The threshold is derived from the given significance value ($P - value$) of the χ^2 distribution with one or two degrees of freedom. Typically, a default value is 95% that matches to 3.841 of the corresponding distribution for one degree of freedom.

4. Experimental results

This section describes function approximations and regressions results on several approximation and regression problems.

4.1. *Function approximation*

We start by comparing our method to RBFN that uses Clustering for Function Approximation (CFA) [20]. We also compare our clustering stage of the FBMS algorithm versus the one in [20]. The $RBFN - CFA$ implements RBFN algorithm where the first stage is clusterization and is done by using the function values in order to achieve a better clustering for function approximation as described in [20]. After the clusterization $RBFN-CFA$ is trained to adjust the parameters, centers and widths of the radial as well as the output weights by Levenberg-Marquardt algorithm. We denote the previous algorithm as PRBFN and PRBFN2, the new algorithm, with model selection.

For the approximation problems we followed [20] and measured the normalized root mean square error (NRMSE):

$$NRMSE = \sqrt{\frac{\sum_{i=1}^{n}(f(x^i) - t^i)^2}{\sum_{i=1}^{n}(f(x^i) - \bar{t})^2}}, \tag{26}$$

where $f(x^i)$ is the function approximator output of the input vector x^i and $\bar{t}$ is the mean output of all input vectors.

The first target to approximate is:

$$f_1(x) = \frac{\sin(2\pi x)}{\exp(x)}, x \in [0, 10]. \tag{27}$$

We use four prototypes and 1000 samples of f_1 generated by evaluating inputs taken uniformly from the interval $[0, 10]$. The second function from [20] is:

$$f_2(x) = 0.2 + 0.8(x + 0.7\sin(2\pi x)), x \in [0, 1], \tag{28}$$

using 21 equidistant input-output training examples belonging to the interval $[0, 1]$. The third function approximate also used in [20] is two-input data as follows:

$$f_3(x1, x2) = \frac{(x_1 - 2)(2x1 + 1)}{1 + x_1^2}\frac{(x_2 - 2)(2x_2 + 1)}{1 + x_2^2}, x_1, x_2 \in [-5, 5], \tag{29}$$

where a complete set of 441 examples obtained from a grid of 21×21 points uniformly distributed in the input interval defined for f_3. Table 1 shows the results of these three data-sets. The results for $RBFN - CFA$ are quoted from [20].

Table 1. Comparison of normalized mean squared error results on three data-sets.

Function	f1	f2	f3
RBFN-CFA	0.952±0.001	0.380±0.035	0.926±0.008
PRBFN	0.788±0.008	0.177±0.001	0.752±0.061
PRBFN2	0.103±0.000	0.082±0.000	0.663±0.000

Note: Results are given just after the initialization procedure using four prototypes.

Figure 1 depicts the regression function f_1, the output of the regression network and the location of prototype points used by the algorithm to construct the architecture. The prototypes are located at the extrema points of the function where the variance is large. This is also a good initial condition and, thus, this algorithm has superior results to CFA. Please note that no prototype is allocated to the linear part, where in [20] the prototypes are allocated to the constant parts of the function as well.

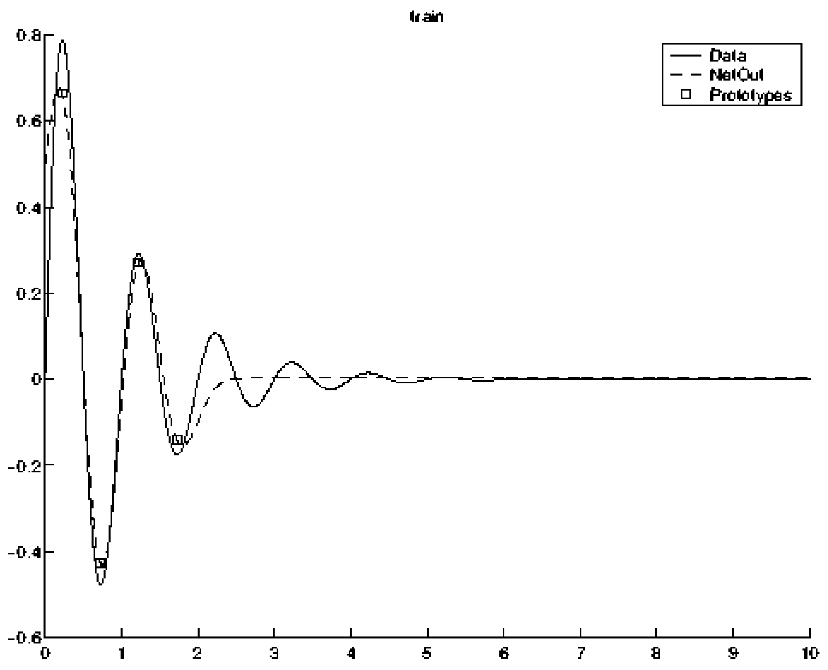

Fig. 1. The regression function f_1 (continuous line), the output of the PRBFN regression network (dashed line) and the location of prototype points (rectangles).

In addition, we consider the set of eight two dimensional functions used in [48,31] to compare the performance of several adaptive methods. These functions, which form a suitable test set, are described in Table 2. For the above data sets, we use the results from [31]. They used a Generalized Radial Basis Function (GRBF) allowing a different variance along each input dimension. The GRBF algorithm updates the centers, variances and forward weights [31]. In addition, they used the Expectation-Maximization (EM) algorithms to obtain better and faster convergence. We follow the tests in [31] and report the Signal to Error Ratio (SER) in dB defined as follows:

$$SER = 20\log(\frac{1}{N}\sum_{i=1}^{N}|\frac{y_i}{e_i}|).$$

Table 2. Functions used to generate two-dimensional data sets.

Name	Function	Domain
Func 1	$y = \sin(x_1 x_2)$	$[-2, 2]$
Func 2	$y = \exp(x_1 \sin(\pi x_2))$	$[-1, 1]$
Func 3	$y = \frac{40\exp(8((x_1-0.5)^2+(x_2-0.5)^2))}{\exp(8((x_1-0.2)^2+(x_2-0.7)^2))+\exp(8((x_1-0.7)^2+(x_2-0.2)^2))}$	$[0, 1]$
Func 4	$y = (1 + \sin(2x_1 + 3x_2)/(3.5 + \sin(x_1 - x_2))$	$[-2, 2]$
Func 5	$y = 42.659(0.1 + x_1((0.05 + x_1^4 - 10x_1^2 x_2^2 + 5x_2^4))$	$[-0.5, 05]$
Func 6	$y = 1.3356[1.5(1 - x_1 + \exp(2x_1 - 1)\sin(3\pi(x_1 - 0.6)^2) + \exp(3(x_2 - .05))\sin(4\pi(x_2 - 0.9)^2)]$	$[0, 1]$
Func 7	$y = 1.9[1.35 + \exp(x_1)\sin(13(x_1 - 0.6)^2] + \exp(3(x_2 - 0.5))\sin(4\pi(x_2 - 0.9)^2)]$	$[0, 1]$
Func 8	$y = \sin(2\pi\sqrt{x_1^2 + x_2^2})$	$[-1, 1]$

Note: The first column denotes the function name, the second the function and the third is the domain for both coordinates.

Table 3. Results of function approximation on eight data sets.

	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8
Gradient	20.4	40.5	32.2	19.0	26.1	28.7	26.3	16.2
EM-1	17.9	30.6	34.7	14.1	27.4	29.4	30.5	14.7
EM-2	18.6	29.8	34.8	16.3	27.8	28.9	30.5	15.0
Soft-EM	20.0	39.3	42.9	19.5	31.8	33.6	34.3	16.3
PRBFN2	54.2	62.7	49.7	34.3	94.9	101.9	66.4	32.3

Note: The first four rows are taken from [31]. The last row contains the results of the PRBFN algorithm that is described in this work. Each result is obtained with 15 hidden units. However, a PRBFN with 15 hidden units has less parameters since a Perceptron has less parameters than a RBF unit with diagonal covariance matrix, and the unit type selection algorithm can select either a Perceptron or RBF unit for each cluster.

4.2. *Regression*

We start with three variants of hybrid network. Our first extension to a hybrid of projection and RBF units (PRBFN) hybrid architecture [4,6], model selection and parameter estimation (PRBFN2) [7], FBMS using MDL as described in this paper (PRBFN-MDL).

Orr's RBF [14] method ($RBF - Reg - Tree$) is based on regression tree for clusterization. This methods builds a large tree and then prunes it using model selection criteria to achieve a smaller tree. Matlab's RBF package ($RBF - OLS$) implements an incremental algorithm [49], a new unit is added with a center that corresponds to the pattern with the largest contribution to the current objective function. Bishop's algorithm [2] is based on the Expectation Maximization algorithm [10] for clustering ($RBF - EM$). In addition, for some data-sets, we have also used a backpropogation algorithm using Levenberg-Marquardt optimization technique ($BP - Lev - Marq$), and a Logistic Regression [25] algorithm ($LogistReg$). The following results are given on the test portion of each data-set and represent an average over 100 runs and include standard error.

The LogGaus data-set is a composition of one ridge function and three Gaussians as follows:

$$f(x) = \frac{1}{1+\exp(-w^T x)} + \sum_{i=1}^{3} \exp\left(-\frac{\| x - m_i \|^2}{2\sigma^2}\right),$$

where $w = (1,1)$, the centers of the Gaussian functions are at $(1,1),(1,-5),(-4,-2)$ and $\sigma = 1$. A random normally distributed noise with zero mean and 0.1 variance is added to the function. The whole data-set is composed of 441 points and it is divided randomly into two sets of 221 and 220 points each. The first set serves as the training set and the second one is the test set. All the regressors, that we have tested did not reveal the true structure of the data, only PRBFN2 revealed the three Gaussians and the ridge function. This fact is amplified from the results on this data-set. Thus, we make the observation that PRBFN has high performance when the data is composed from ridge and Gaussian-s. If the data is composed either from Gaussians or ridge function, it can reach the performance of other regressors.

The second data-set is a two dimensional sine wave,

$$y = 0.8\sin(x_1/4)\sin(x_2/2),$$

with 200 training patterns sampled at random from an input range $x_1 \in [0,\ 10]$ and $x_2 \in [-5,\ 5]$ (see [14] for details). The clean data was corrupted by additive Gaussian noise with $\sigma = 0.1$. The test set contains 400 noiseless samples arranged in a 20 by 20 grid pattern, covering the same input ranges. Orr measured the error as the total squared error over the 400 samples. We follow Orr and report the error as the SSE on the test set.

The third data-set is a simulated alternating current circuit with four input dimensions (resistance R, frequency ω, inductance L and capacitance C and one output impedance $Z = \sqrt{R^2 + (\omega L - 1/\omega C)^2}$. Each training set contained 200 points sampled at random from a certain region [14]. Again, additive noise was added to the outputs. The experimental design is the same as the one used by Friedman in the evaluation of MARS [18]. Friedman's results include a division by the variance of the test set targets. We follow Friedman and report the normalized MSE on the test set. Orr's regression trees method [14] outperforms the other methods on this data-set. However, the PRBFN network achieves results similar to Orr's method.

The next experiments were done on the Pumadyn regression data, taken from the DELEVE archive [38]. This data was generated from a simulation of the dynamics of a Puma robot arm. The target is the angular acceleration of one of the links and the inputs are various joint angles, velocities and torques. The Pumadyn data-set has several groups, and we have used the data group with the largest noise and non-linearity. This group has either 8 or 32 inputs, each of which is non-linear with high noise. The training set size is: 64,128,256,512,1024. This is done to test how well each method scales with training sample size.

The methods we use to compare our results are:

- **Lin-1** Linear least squares regression.

Table 4. Comparison of test-set Mean squared error results on three data-sets.

Algorithm	LogGauss	2D Sine	Friedman
RBF-Reg-Tree	0.02±0.14	0.91±0.19	0.12±0.03
RBF-OLS	-	0.74±0.41	0.20±0.03
RBF-EM	0.02±0.02	0.53±0.19	0.18±0.02
PRBFN	0.02±0.02	0.53±0.21	0.15±0.03
PRBFN-MDL	0.02±0.02	0.50±0.21	0.15±0.03
PRBFN2	0.01±0.01	0.46±0.19	0.12±0.03

Note: MSE Results are an averaged of 100 runs and include standard deviation. They are given for several variants of RBF networks which were used by Orr to asses RBFs.

- **kNN-cv-1** KNN for regression. K is selected by using leave one out cross validation.
- **MLP-ens-1** MLP ensembles with early stopping and conjugate gradient.
- **ME-ese-1** Mixtures of experts using early stopping.
- **HME-ens-1** Hierarchical mixtures of experts using early stopping.
- **GP-map-1** Gaussian processes for regression, trained using a maximum a-posteriori approach implemented by conjugate gradient optimization.
- **MLP-MC-1** Multi-layer Perceptron (ensembles) networks trained by MCMC methods [34]. a maximum a-posteriori via conjugate gradient.
- **MARS3.6-bag-1** Multivariate adaptive regression splines (MARS)[18], version 3.6 with bagging.
- **PRBFN** The first PRBFN algorithm as described in [6].
- **PRBFN-AS-RBF** RBF with pruning as described in Section 3.3.
- **PRBFN-AS-MLP** MLP with pruning as described in Section 3.3.
- **PRBFN-LRT** Full PRBFN method LRT for pruning.
- **PRBFN-MDL** Full PRBFN method with the MDL principle for pruning.
- **PRBFN2** PRBFN - Evidence model selection for unit type and BIC pruning as described in [7].

Additional details can be obtained from the DELVE web site [38].

5. Discussion

The work presented in this paper provides a comparison between several statistical methods for model selection. The FBMS is a constructive algorithm, which can be used for different types of neural networks. It was motivated by the success of the original hybrid architecture which was introduced in [4,6,7]. Several assumptions were made in various parts of the architecture construction. We have shown that that under these assumptions, an architecture that is smaller in size and better in generalization performance can be obtained. Furthermore, while this architecture is particularly useful when the data contains ridge and Gaussian parts, its performance

Table 5. Regression on Pumadyn with 32 input non-linear with high noise.

Training size	64	128	256	512	1024
Lin-1	1.98±0.25	1.20±0.05	0.96±0.02	0.89±0.02	0.86±0.02
kNN-cv-1	1.00±0.02	1.01±0.03	0.94±0.02	0.92±0.02	0.90±0.02
MLP-ens-1	1.25±0.04	1.13±0.09	0.96±0.02	0.89±0.02	0.86±0.02
HME-ens-1	1.22±0.02	1.12±0.04	0.96±0.02	0.89±0.02	0.87±0.02
GP-map-1	1.01±0.06	0.70±0.12	0.38±0.01	0.36±0.01	0.35±0.01
MLP-mc-1	0.88±0.06	0.58±0.06	0.50±0.09	0.59±0.06	0.35±0.01
MARS3.6-bag-1	0.93±0.06	0.53±0.03	0.38±0.01	0.35±0.01	0.34±0.01
PRBFN	1.15±0.13	1.12±0.08	1.03±0.06	0.87±0.13	0.78±0.13
PRBFN-AS-RBF	1.14±0.2	0.57±0.09	0.40±0.02	0.39±0.02	0.38±0.03
PRBFN-AS-MLP	1.11±0.08	0.84±0.06	0.69±0.07	0.54±0.06	0.40±0.02
PRBFN-LRT	1.45±0.2	1.14±0.09	0.79±0.07	0.55±0.05	0.44±0.03
PRBFN-MDL	0.85±0.2	0.49±0.09	0.43±0.07	0.37±0.04	0.34±0.02
PRBFN2	0.75±0.10	0.43±0.02	0.38±0.01	0.37±0.02	0.34±0.01

Note: Regressors predictions are improved as the data-set get larger. PRBFN2 outperforms all the regressors on the small data-sets.

Table 6. Regression on Pumadyn with 8 input non-linear with high noise.

Training size	64	128	256	512	1024
Lin-1	0.73±0.019	0.68±0.02	0.65±0.01	0.63±0.014	0.63±0.02
kNN-cv-1	0.79±0.02	0.71±0.02	0.64±0.01	0.58±0.019	0.53±0.02
MLP-ens-1	0.72±0.02	0.67±0.02	0.61±0.01	0.49±0.01	0.41±0.01
HME-ens-1	0.72±0.02	0.67±0.02	0.61±0.01	0.54±0.02	0.44±0.02
GP-map-1	0.44±0.03	0.38±0.01	0.35±0.01	0.33±0.01	0.32±0.01
MLP-mc-1	0.45±0.01	0.39±0.02	0.35±0.01	0.32±0.01	0.32±0.01
MARS3.6-bag-1	0.51±0.02	0.38±0.01	0.36±0.01	0.34 ±0.01	0.34±0.01
PRBFN	0.63±0.07	0.58±0.11	0.48±0.09	0.38±0.05	0.35±0.02
PRBFN-AS-RBF	0.51±0.03	0.38±0.02	0.36±0.01	0.33 ±0.01	0.32±0.01
PRBFN-AS-MLP	0.57±0.05	0.59±0.14	0.37±0.02	0.33 ±0.08	0.32±0.01
PRBFN-LRT	0.72±0.11	0.60±0.05	0.43±0.02	0.41 ±0.01	0.35±0.02
PRBFN-MDL	0.6±0.2	0.52±0.09	0.43±0.07	0.42±0.05	0.32±0.02
PRBFN2	0.48±0.03	0.38±0.01	0.34±0.01	0.33±0.01	0.32±0.01

was not below the performance of the best known MLP or RBF networks when data which contains one type of structure was used.

The parameter pruning method suggested in this paper, is applied to individual weights of each hidden unit. Its role in producing a smaller net with better generalization is critical. Three criteria have been used: BIC,MDL and LRT. The BIC and MDL have the ability to choose the best model out of many models. The LRT works only on nested models and the process is terminated when the null hypothesis is rejected, i.e., when the change in the likelihood is significant. The results in this work show that the LRT method does not perform as well as the MDL or the Bayesian method for pruning of the hybrid estimator. It appears that the Bayesian approach, which was used here, is superior to the MDL approach. This approach makes the assumption that the prior distribution of the model parameters is normal with zero mean, which is not necessarily correct for the centers of the RBF. This assumption has not been made in the Bayesian case [44,28]. However, the MDL approach can be very useful for Perceptrons, when small values of their parameters simplifies the network [24].

We have tested the new architecture construction on seven regression problems and eleven approximation problems. There are four cases where better results were obtained. These results occur in the approximation problems where the function is composed of parts that vary considerably and parts that are almost linear, e.g. f_1. In these cases, the first step of the training that allocates many prototypes to the dynamic part successfully modeled the function and has shown low error rates. In the LogGaus data-set, which is composed of Ridge and Gaussian parts – most suitable for our hybrid – results were again improved with our proposed architecture construction.

The tests on the Pumadyn family have raised interesting observations; Our proposed pruning algorithm dramatically improved the generalization of regressors, while the pruning process has removed 95% of the weights. For instance for the small data-sets, only 10 weights were needed on average. The proposed method has always been one of the best regressors and it has good performance on pumadyn32-nh. In summary, based on data that has been extensively studied by others, it appears that the proposed method is able to better model nonlinear data and, in particular, it can improve generalization, while keeping the number of the estimated parameters smaller compared to other regression methods.

References

1. A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.
2. C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.
3. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* The Wadsworth Statistics/Probability Series, Belmont, CA, 1984.
4. S. Cohen and N. Intrator. Automatic model selection in a hybrid perceptron/radial network. *Information Fusion*, 3(4):259–266, 2002.
5. S. Cohen and N. Intrator. Forward and backward selection in regression hybrid network. In *Third International workshop on Multiple Classifier Systems*, 2002.
6. S. Cohen and N. Intrator. A hybrid projection based and radial basis function architecture: Initial values and global optimization. *Pattern Anal. Appl. (Special issue on Fusion of Multiple Classifiers)*, 5(2):113–120, 2002.
7. S. Cohen and N. Intrator. On the evidence in a forward and backward regression hybrid network, 2003. Submitted.
8. Cortes and Vapnik. Support-vector networks. *Machine Learning*, 20, 1995.
9. T. Cover and J. Thomas. *Elements of Information Theory.* Wiley, 1991.
10. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Proceedings of the Royal Statistical Society*, B-39:1–38, 1977.
11. D. L. Donoho and I. M. Johnstone. Projection-based approximation and a duality with kernel methods. *Annals of Statistics*, 17:58–106, 1989.

12. H. Drucker, R. Schapire, and P. Simard. Improving performance in neural networks using a boosting algorithm. In Steven J. Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 42–49. Morgan Kaufmann, 1993.
13. R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley Sons, INC., New York, 2001.
14. M.J.L Orr et al. Combining regression trees and radial basis functions. *Int. J. of Neural Systems*, 10(6):453–466, 2000.
15. S. E. Fahlman and C. Lebiere. The cascade–correlation learning architecture. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 524–532. Morgan Kaufmann, San Mateo, CA, 1990.
16. S. E. Fahlman and C. Lebiere. The cascade–correlation learning architecture. CMU-CS-90-100, Carnegie Mellon University, 1990.
17. G.W. Flake. Square unit augmented, radially extended, multilayer percpetrons. In G. B. Orr and K. Müller, editors, *Neural Networks: Tricks of the Trade*, pages 145–163. Springer, 1998.
18. J. H. Friedman. Mutltivariate adaptive regression splines. *The Annals of Statistics*, 19:1–141, 1991.
19. J. H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76:817–823, 1981.
20. J. Gonzalez, I. Rojas, H. Pomares, J. Ortega, and A. Prieto. A new clustering techniques for function approximation,. *IEEE Transaction on Neural Networks*, 13:132–142, 2002.
21. B. Hassibi and D. G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In C. L. Giles, S. J. Hanson, and J. D. Cowan, editors, *Advances in Neural Information Processing Systems*, volume 5. Morgan Kaufmann, San Mateo, CA, 1993.
22. T. Hastie and R. Tibshirani. Generalized additive models. *Statistical Science*, 1:297–318, 1986.
23. T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman and Hall, London, 1990.
24. G. E. Hinton and D. van Camp. Keeping neural networks simple by minimizing the description length of the weights. In *Sixth ACM conference on Computational Learning Theory*, pages 5–13, July 1993.
25. David W. Hosmer and Stanley Lemeshow. *Applied Logistic Regression*. Wiley Series in Probability and Mathematical Statistics, 1989.
26. R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.
27. M. I. Jordan and R. A. Jacobs. Hierarchies of adaptive experts. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 985–992. Morgan Kaufmann, San Mateo, CA, 1992.
28. R. E. Kass and A. E. Raftery. Bayes factors. *Journal of The American Statistical Association*, 90:773–795, 1995.
29. A. Krogh and J. A. Hertz. A simple weight decay can improve generalization. In J.E. Moody, S.J Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 950–957. Morgan Kaufmann, San Mateo, CA, 1992.
30. Y.C. Lee, G. Doolen, H.H. Chen, G.Z.Sun, T. Maxwell, H.Y. Lee, and C.L. Giles. Machine learning using higher order correlation networks. *Physica D*, pages 22–D:276–306, 1986.

31. I. Santamaria M. Lazaro and C. Pantaleon. A new EM-based training algorithm for RBF networks. *Neural Networks*, 16:69–77, 2003.
32. D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.
33. John Moody. Prediction risk and architecture selection for neural networks. In V. Cherkassky, J. H. Friedman, and H. Wechsler, editors, *From Statistics to Neural Networks: Theory and Pattern Recognition Applications.* Springer, NATO ASI Series F, 1994.
34. R. M. Neal. *Bayesian Learning for Neural Networks.* Springer, New York, 1996.
35. S. J. Nowlan. Soft competitive adaptation: Neural network learning algorithms basd on fitting statistical mixtures. Ph.D. dissertation, Carnegie Mellon University, 1991.
36. S. J. Nowlan and G. E. Hinton. Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4:473–493, 1992.
37. A. Papoulis. *Probbaility, Random Variables, and Stochastic Process*, volume 1. McGRAW-HILL, New York, third edition, 1991.
38. C.E. Rasmussen, R.M. Neal, G.E. Hinton, D. Van Camp, Z. Ghahrman M. Revow, R. Kustra, and R. Tibshirani. The delve manual. *Graduate Department of Computer Science, University of Toronto*, 1996.
39. J. Rissanen. Modeling by shortest data description. *Automat.*, 14:465–471, 1978.
40. J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11:416–431, 1983.
41. J. Rissanen. Universal coding, information, prediction, and estimation. *IEEE Transactions on Information Theory*, 30:629–636, 1984.
42. J. Rissanen. Minimum description length principal. *Encyclopedia of Statistical Science*, pages 523–527, 1985.
43. J. Rissanen. Stochastic complexity and modeling. *Annals of Statistics*, 14:1080–1100, 1986.
44. G. Schwarz. Estimationg the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.
45. C. E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27:379–423 and 623–656, 1948.
46. C. J. Stone. The dimensionality reduction principle for generalized additive models. *The Annals of Statistics*, 14:590–606, 1986.
47. K. Suzuki, I. Horiba, and N. Sugie. A simple neural network algorithm with application to filter synthesis. *Neural Processing Letters, Kluwer Academic Publishers, Netherlands*, 13:43–53, 2001.
48. F. Mulier V. Cherkassky, D. Gehring. Comparison of adaptive methods for function estimation from samples. *IEEE Transactions of Neural Neural*, 7(4):969–984, 1996.
49. P.D. Wasserman. *Advanced Methods in Neural Computing.* Van Nostrand Reinhold, New York, 1993.
50. A. S. Weigend, D. E. Rumelhart, and B. A. Huberman. Generalization by weight-elimination with application to forecasting. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 3, pages 875–882. Morgan Kaufmann, San Mateo, CA, 1991.