A strong regularization on a hybrid MLP/RBF architecture achieves small bias and small variance error[⋆]

Shimon Cohen and Nathan Intrator

School of Computer Science, Tel Aviv University
www.math.tau.ac.il/∼nin

Abstract. We introduce a Forward Backward and model selection algorithm for constructing a hybrid network of radial and perceptron hidden units for regression. The algorithm determines if a radial or a perceptron unit is required at a given region of input space. Given an error target, the algorithm also determines the number of hidden units. Then the algorithm uses model selection criteria and prunes unnecessary weights. This results in a final architecture which is often much smaller than an RBF network or a MLP.

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

1 Introduction

The construction of a network architecture which contains units of different types at 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 [23, 11]. 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 [5, 4]. This architecture produced far better classification and regression results when compared with advanced RBF methods or with MLP architectures. In this work, we further introduce a novel training methodology, which:

1. evaluates the need for additional hidden units,
2. chooses optimally their nature – MLP or RBF – and
3. determines their optimal initial weight values.

The determination of additional hidden units is based on an incremental strategy which searches for regions in input space for which the input/output function approximation leads to highest residual error. The last step is to prune unnecessary parameters and to select the best model from the sequence of nested models using Bayesian model selection or Likelihood Ratio Test (LRT). This approach, coupled with optimal determination of initial weight values for the additional hidden units, constructs a computationally efficient training algorithm which appears to scale up with the complexity of the data, which is better than regular MLP or RBF methods.

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

⋆ 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.

approximator, namely, it can approximate any L^2 function, it is also known that the approximation may be very greedy, with the number of hidden units growing very large as a function of the desired approximation error. When the number of train patterns in the train sample set 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 donut) with projection units or vice versa. 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 [2] where the goal is to reach a good division of the input space and use a very 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 [28, 19], where a gating network performs the division of the input space and small neural networks perform the function approximation at each region separately. Nowlan's [28] 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 neural networks as terminating nodes was proposed by [20]. The boosting algorithm [8] 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 accounts the geometric structure of the input data indirectly. A remote family of architectures, where the function approximation is constructed incrementally, is projection pursuit [13] and additive models [15, 16].

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 [7], 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) [15, 16]. 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 [32], 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 [23] 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 [11] 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 [11]. 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 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

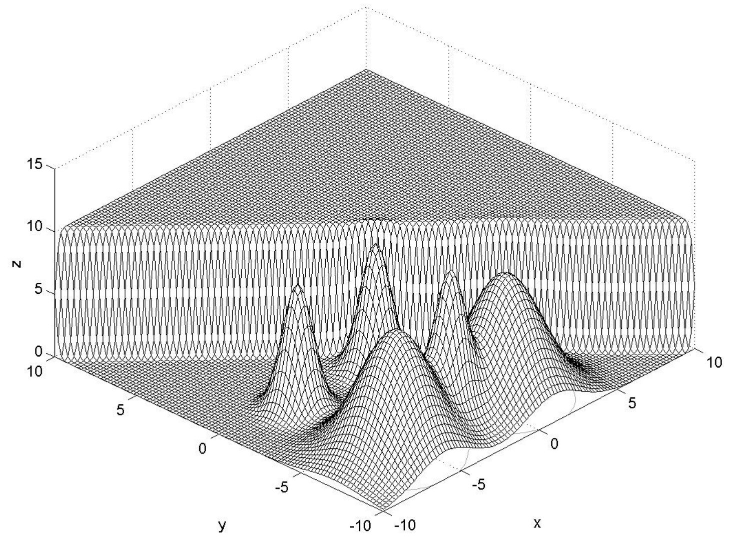

Fig. 1. Data that is composed of five clusters and a sigmoidal surface.

hill-plateau and other regions that contain Gaussian bumps, as demonstrated in Figure 1. The hybrid architecture [4, 5], 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. The first training step in the previous approach [4, 5] was to cluster the data. In the next step, we tested two hypotheses for each cluster. If a cluster was far from radial Gaussian, we rejected this hypothesis and accepted the null hypothesis. However, we had to use a threshold for rejecting the normal distribution hypothesis. When it was decided that a data cluster was likely to be normal, an RBF unit was used and otherwise a Perceptron (projection) unit was used. The last step was to train the hybrid network with full gradient descent on the full parameters.

However, the selection based on a simple hypothesis test could be improved and suffered from an unclear way of estimating the hypothesis rejection threshold. Another problem with the old approach is that the number of hidden units has to be given to the algorithm in advance. In this paper, we introduce a Forward Backward Model Selection (FBMS) algorithm that automatically selects the type of hidden units as well as the number of units. In addition the algorithm removes unnecessary parameters using statistical model selection criteria like the Likelihood Ratio Test (LRT) or the Bayesian Information Criterion (BIC) [31].

There are several approaches to set the structure of a Neural Network. The first one is forward selection. This approach starts with a small [26, 10] network and add units until an error goal is reached. Another approach is to start with a large network and [9] prune unnecessary [21] units, until a given criteria is met.

In this paper we use a combination of these approaches. We start with a small network and expand it until a given error goal is met. Thus, the algorithm determines the number of hidden units automatically. As noted above, a very difficult task in training a hybrid neural network is to find the radial and projection parts automatically. This problem is amplified for high dimension data, where the data cannot be visualized very well. We propose a novel way to select the type of hidden unit automatically. After training the network, we propose to prune unnecessary weights using Bayesian techniques or LRT. The FBMS algorithm leads to smaller networks while maintaining good generalization of the resulting network.

3 Parameter estimation and model selection

An incremental architecture with more than one type of building component, requires four decisions at each step; (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; (iv) prune unnecessary weights.

The SMLP [11] network uses both RBF and Perceptron units at each cluster. In higher order networks [23], 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 region 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 train 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 cluster.
- Full gradient descent.
- Pruning of unnecessary weights.

In subsequent sections, we describe each step of the algorithm in more details.

3.1 Data clustering

The objectives of regression and function approximation are different than classification. The main implications on the training algorithm are summarized below:

1. In classification, the output variable takes a finite set of values – the class labels. In regression the output value, the dependent variable is continuous. Thus, training is more difficult as it is not enough to find only the class boundaries but also to correctly approximate the function in the whole space.
2. In regression problems, the output variable may be different from their target. It is acceptable to have similar values, in the sense that the difference is not as important for classification. In other words the differences between the regressor and the true values may be worthless approximation error. In classification problems the output labels, the class tags, may not be related to any distance operator. The output response different from the correct one may be unacceptable, that is, the class with maximum posterior probability is not the correct one.
3. In regression, a more appropriate objective for the clustering algorithm is a reduction in the sum squared error (SSE) while in classification, the entropy measure is more appropriate [4].
4. In classification problems the a-priori class membership may be available some times (it exists partially in the TSS). This however, is not an interesting information in regression problems.

Next we describe a clusterization algorithm for regression and function approximation.

We start by clustering the data based on the function values. This is done by minimizing an objective function, the Sum of Square Errors (SSE). The SSE is equivalent to the maximum likelihood under Gaussian assumption about the noise. Thus, reducing the SSE is equivalent to the maximization of the data likelihood.

Consider a data set D, we attempt to decompose it into a set $\{C_i\}_{i=1}^{k}$, such that $\bigcup C_i = D$ and $C_i \bigcap C_j = \phi$ for $i \neq j$.

Let the objective function be the Sum of Square Residual about the mean (SSR):

$$SSR(C_0) = \sum_{y_i \in C_0} (y_i - \bar{yo})^2,$$

where $\bar{yo}$ is the mean of $y_i \in C_0$. Given a training data set D, we attempt to decompose it into a set $\{C_i\}_{i=1}^{k}$, such that $\bigcup C_i = D$ and $C_i \bigcap C_j = \phi$ for $i \neq j$. The following algorithm, which is similar to the one proposed in CART [2], splits the cluster with the largest objective function reduction into two clusters. Consider a split for a cluster C_0 into two clusters C_1 and C_2. Let the SSR reduction be defined as follows:

$$\Delta SSR(C_0) = SSR(C_0) - (SSR(C_1) + SSR(C_2)). \tag{2}$$

Finding a possible set of candidate for splitting Since a cluster C with n members has $2^n - 1$ possible number of meaningful splits, an exhaustive search is not feasible. CART [2] solves this problem by considering each axis $1 \leq l \leq d$ of the input space separately and searches for the best split of the form $x_i \geq \lambda_i$ for axis x_i. The sorting of the data (by considering each axis separately) reduces the number of possibilities to dn. We seek a similar approach which is not as restricted as CART to projections that are parallel to the axes.

Splitting rule We assume that the underlying function to be estimated is continuous. Let y_1 be the minimum value of the function in C_0, let x_1 be the respected pattern. Let y_2 be the maximum value of the function in C_0 and let x_2 be the respected pattern.

The splitting procedure is defined as follows:

- For each pattern find the Euclidean distance to x_1 and x_2.
- If the distance to x_1 is smaller associate it to C_1, otherwise to C_2.
- Choose the split of a cluster with the largest reduction in the objective function.

The splitting is continued until an error goal of the hybrid classifier is reached or a predefined maximum number of clusters is achieved.

Additional distance measures such as Mahalanobis or Manhattan can be considered.

The above splitting rules are simple to implement and can perform data splits that are not parallel to the feature axes.

3.2 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 train set are independent.

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

$$L = \frac{1}{(2\pi)^{\frac{N}{2}} \sigma^N} \exp(-\frac{\sum_{n=1}^{N} (y_n - t_n)^2}{2\sigma^2}), \tag{3}$$

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:

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

Thus, minimization the SSE is equivalent to the maximization of the likelihood of the data. To obtain the maximum likelihood value of σ, we derive Eq. 4 with respect to sigma.

$$\frac{\partial LL}{\partial \sigma} = -\frac{N}{\sigma} + \frac{\sum_{n=1}^{N} (y_n - t_n)^2}{\sigma^3}. \tag{5}$$

Thus, we obtain the maximum likelihood value for σ:

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{n=1}^{N} (y_n - t_n)^2. \tag{6}$$

We will use Eq. 4 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) to select the type of unit. We use either the BIC or LRT for the pruning process. Next we describe the two approaches.

We start by describing the Bayesian approach. Mackay [25] used the evidence of the model for model selection. Kass and Raftery [22] 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:

$$p(D|M) = \int_w p(D, w|M)dw = \int_w p(D|w, M)p(w|M)dw. \tag{7}$$

The Bayes Factors are then defined as:

$$\frac{p(M_1|D)}{p(M_2|D)} = \frac{p(D|M_1)p(M_1)}{p(D|M_2)p(M_2)}. \tag{8}$$

The integration of Eq. 7 can be performed by using Laplace integral [22, 30] which approximates the integrand by a quadratic function (Taylor approximation to the second order). Thus, the value of the integral becomes:

$$p(D|M) \cong (2\pi)^d |H|^{-1|2} p(D|W_{m_0}, M)p(W_{m_0}|M), \tag{9}$$

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. 7 is by using Monte Carlo Markov Chain (MCMC) techniques [27], 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. 9 by using Gaussian distribution to the a-priori parameters density [22] to arrive at:

$$BIC \equiv \log(p(D|M)) = \log(p(D, W_{m_0}|M)) - \frac{d}{2}\log(N), \tag{10}$$

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

Substitute Eq. 6 and Eq. 4 into Eq. 10 we obtain:

$$BIC_i = -\frac{1}{N}\log(2\pi) - N\log(\hat{\sigma}) - \frac{N}{2} - \frac{d_i}{2}\log(N). \tag{11}$$

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 train data set D, we note that:

$$p(y|D) = \sum_{i=1}^{m} p(y, M_i|D) = \sum_{i=1}^{m} p(y|M_i, D)p(M_i|D). \tag{12}$$

Equation 12 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.

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 [29]:

$$-2\log(\frac{p(D, W_{m_0}|M1)}{p(D, W_{m_0}|M2)}) = \chi^2(d_2 - d_1). \tag{13}$$

This approach uses $P - Values$ to reject the null hypothesis. That is, the simple model is equivalent to the complicated one. Please 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. 6, we arrive at the following test:

$$\chi^2(d_2 - d_1) = 2N \log(\sigma_1^2) - 2N \log(\sigma_2^2). \tag{14}$$

3.3 Unit selection

Now, that we have constructed a decomposition of the input space into more homogeneous subsets, it is time to choose for each such subset the appropriate hidden unit, namely a projection or a radial unit. Since in this case the models are not nested, we apply only the Bayesian approach to select between the different units.

First, we set the parameters of hidden units. Consider two $1 - D$ data fitting problems;

Figure 2 depicts the possible $1 - D$ prototypical projections, an RBF projection (left side) and Ridge projection (right side). The lower part was obtained by multiplying the functions values by -1, although, this is the case when the forward weight is negative.

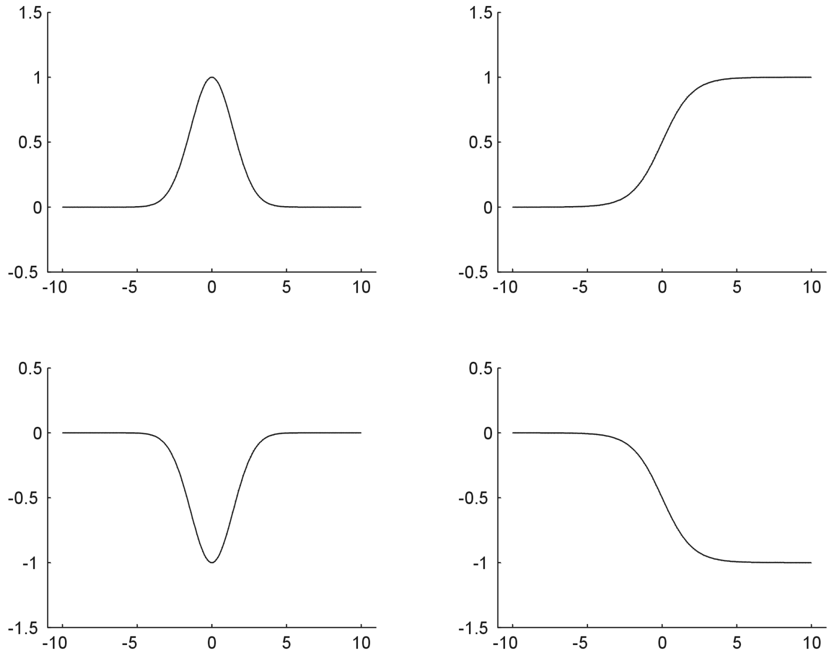

Fig. 2. Left side: upper part positive RBF, lower part negative RBF. Right side: upper part positive ridge, lower part negative ridge

The ridge projection is monotonically increasing with the correlation between its weight vector and the data points. It achieves its maximum value when the correlation is maximized (for a unit projection vector).

Therefore, the ridge weight vector W_{m_0} should be proportional to the pattern where the function acquires its maximum value.

The RBF function is monotonically decreasing with the distance from the maximum point. Thus, the center of the RBF is located at the function maximum point. In this case selection of the value W_{m_0} that maximizes the likelihood is trivial.

After the parameters of the hidden unit have been approximated, we can treat a given unit as a Generalized Linear Model (GLM) and compute the full evidence of the models. We assume that the prior of the forward weight is Gaussian with an unknown variance β. Thus, we have the following probability model for the likelihood:

$$p(D|W_{m_0}, M) = \frac{1}{(2\pi)^{N/2}\alpha^N} exp(\frac{-\sum_{i=1}^{N}(y_i - t_i)^2}{2\alpha^2}). \tag{15}$$

The probability model for the prior:

$$p(W_{m_0}|M) = \frac{1}{(2\pi)^{1/2}\beta} exp(-\frac{w^2}{2\beta^2}). \tag{16}$$

Let y_i be the ridge or RBF values for each $x_i \in C_i$, and let t_i be the targets. Thus, the integrand of the evidence in Eq. 7 can be viewed as:

$$L = \frac{1}{(2\pi)^{N/2}\alpha^N} exp(\frac{-\sum_{i=1}^{N}(y_i - t_i)^2}{2\alpha^2}) \frac{1}{(2\pi)^{1/2}\beta} exp(-\frac{w^2}{2\beta^2}). \tag{17}$$

The maximization of the above function is equivalent to the minimization of the negative log:

$$LL = N\log(\alpha) + \frac{\sum_{i=1}^{N}(y_i - t_i)^2}{2\alpha^2} + \log(\beta) + \frac{w^2}{2\beta^2}, \tag{18}$$

where we have discarded the constants factors. Deriving Eq. 18 with respect to α and equating to zero, we obtain:

$$\alpha^2 = \frac{\sum_{i=1}^{N}(y_i - t_i)^2}{N}. \tag{19}$$

Deriving Eq. 18 with respect to β we obtain:

$$\beta^2 = w^2. \tag{20}$$

Deriving Eq. 18 with respect to w_0, we obtain:

$$w_0 = \frac{1}{N}(\sum_{i=1}^{N}(yi - w\sum_{i=1}^{N}\phi_i). \tag{21}$$

Deriving Eq. 18 with respect to w and setting to zero, we obtain:

$$w = \frac{\beta^2\sum_{i=1}^{N} t_i\phi_i - \frac{\beta^2}{N}\sum_{i=1}^{N} t_i \sum_{i=1}^{N}\phi_i}{\beta^2\sum_{i=1}^{N}\phi_i^2 - \frac{\beta^2}{N}\sum_{i=1}^{N}\phi_i\sum_{i=1}^{N}\phi_i + \alpha^2}. \tag{22}$$

The Hessian of the negative log-likelihood can be computed to arrive at:

$$\mathbf{H} = \begin{pmatrix} \frac{\sum_{i=1}^{N}\phi_i^2}{\alpha^2} + \frac{1}{\beta^2} & \frac{\sum_{i=1}^{N}\phi_i}{\alpha^2} \\ \frac{\sum_{i=1}^{N}\phi_i}{\alpha^2} & \frac{N}{\alpha^2} \end{pmatrix}.$$

Thus, the evidence of the model from Eq. 9 is:

$$LL = -\frac{1}{2}N\log(\alpha) - \frac{1}{2}log(\beta) - \frac{1}{2}log(|H|). \tag{23}$$

Thus, we purpose the following algorithm for unit selection:

– Initialize α and β.
– Loop: compute w, w_o using Eq. 21 and Eq. 22
– Compute α, β using Eq. 19 and Eq. 20.
– Until the difference in LL (Eq. 23) is small.

We note that for large samples $\hat{w} \approx w$ where $\hat{w}$ is the MLE, and $H \approx Ni$, where i is the expected Fisher information matrix for one observation. This is a $(d \times d)$ matrix whose (i,j) element is $-E[\frac{\partial^2 logp(y_1|w)}{\partial w_i \partial w_j}]$, the expectation being taken over the values of y_1, with w held fixed. Thus, $|H| \approx N^d |i|$. This approximation introduces an $O(N^{-\frac{1}{2}})$ into equation 23. If we neglect this term, we arrive at:

$$Evid \approx -N \log(\alpha) - log(\beta). \tag{24}$$

The intuition behind Eq. 24 is that the evidence is large for small MSE errors, but as the weight w becomes larger, the evidence becomes smaller. Thus, if the function does not truly fit the data a larger w is needed, and the evidence has a natural cost complexity factor in this case. This is a practical explanation of the evidence in this case.

The above algorithm converges very fast and the evidence of each model can be computed from Eq. 23. The model with the highest evidence is selected.

Note that our approach of divide and conquer applies only to unit selection. In the resulting architecture, each unit can see the whole data and contribute to the output for every pattern. This is different than the divided and conquer approach of decision tree [2], or to the soft partition of the input space in Hierarchical Mixture of Experts [24].

3.4 Pruning

The last step of our method is to prune unnecessary weights. The prune 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:

$$\phi(x) = \exp(-\sum_{i=1}^{N} \frac{(x_i - c_i)^2}{\sigma_i^2}). \tag{25}$$

If we make the substitution:

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

we obtain:

$$\phi(x) = \exp(-\sum_{i=1}^{d} \frac{(x_i - c_i)^2 r_i^2}{2}). \tag{26}$$

Consider setting r_j to zero:

$$\begin{aligned} \phi(x) = \exp(-\sum_{i=1, i\neq j}^{d} \frac{(x_i - c_i)^2 r_i^2}{2}) \exp(-\frac{(x_j - c_j)^2 r_j^2}{2}) \\ = \exp(-\sum_{i=1, i\neq j}^{d} \frac{(x_i - c_i)^2 r_i^2}{2}) \exp(0) \\ = \exp(-\sum_{i=1, i\neq j}^{d} \frac{(x_i - c_i)^2 r_i^2}{2}). \end{aligned} \tag{27}$$

The partial derivatives with respect to r_i are:

$$\frac{\partial \phi}{\partial r_i} = -\phi(x)(x_i - c_i)^2 r_i. \tag{28}$$

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. 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 [21, 14]. We start with the full model. At each step, we prune the least significant weight. That is, the weight that contributes least to the objective function. When the BIC criterion is 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

In this section we compare our method to RBFN that uses Clustering for Function Approximation (CFA) [18]. We also compare our clustering stage of the FBMS algorithm versus the one in [18]. 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 [18]. 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.

For the approximation problems we followed [18] and measured the normalized root mean square error (NRMSE):

$$NRMSE = \sqrt{\frac{\sum_{i=1}^{n}(f(x^i) - t^i)^2}{\sum_{i=1}^{n}(f(x^i) - \bar{t})^2}}, \tag{29}$$

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:

$$f_1(x) = \frac{sin(2\pi x)}{exp(x)}, x \in [0, 10]. \tag{30}$$

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

The second function, also taken from [18], to consider is:

$$f_2(x) = 0.2 + 0.8(x + 0.7 \sin(2\pi x)), x \in [0, 1] \tag{31}$$

from 21 equidistant input-output training examples belonging to the interval $[0, 1]$.

The third function approximate also used in [18] is two-input data as follows:

$$f_3(x1, x2) = \frac{(x_1 - 2)(2x1 + 1)}{1 + x_1^2} \frac{(x_2 - 2)(2x_2 + 1)}{1 + x_2^2}, x_1, x_2 \in [-5, 5] \tag{32}$$

where a complete set of 441 examples obtained from a grid of $21x21$ 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 [18].

Figure 3 shows f_1 in continuous line, the output of PRBFN is displayed in dashed line and the prototypes are shown as rectangles. The prototypes are located at the extremum points of the function where the variance is large. This is also a very 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 [18] the, prototypes are allocated to the constant parts of the function as well.

Function	f1	f2	f3
RBFN-CFA	0.952±0.001	0.380±0.035	0.926±0.008
PRBFN2	0.103±0.000	0.082±0.000	0.663±0.000

Table 1. Comparison of normalized mean squared error results on three data sets (see [18] for details). Results are given just after the initialization procedure using four prototypes.

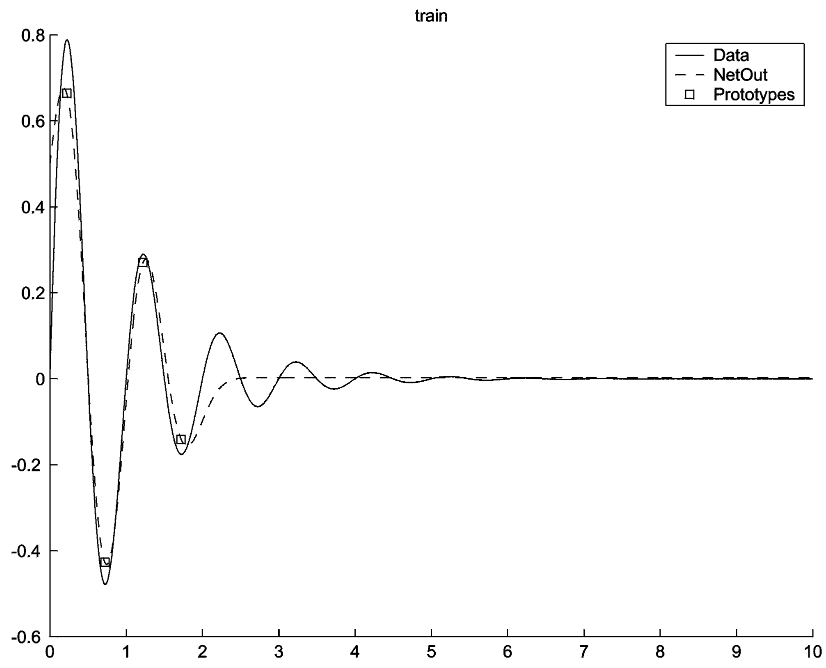

Fig. 3. f_1 taken from [18]. The net output is in dash and the prototypes are the rectangles.

4.2 Regression

We start with three variants of RBF. Our first extension to a hybrid of projection and RBF units (PRBFN) [4, 5] presents hybrid architecture, model selection and parameter estimation (PRBFN2).

Orr's RBF [9] 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 [33], 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 [1] is based on the Expectation Maximization algorithm [6] 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 [17] 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 1:

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

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 train 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 2D sine wave,

$$y = 0. \sin(x_1/4)\sin(x_2/2),$$

with 200 training patterns sampled at random from an input range $x_1 \in [0,\ 10]$ and $x_2 \in [-5,\ 5]$. 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 [9, for further details]. 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 [12]. 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 [9] outperforms the other methods on this data set. However, the PRBFN neural network achieves similar results to Orr's method.

	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
PRBFN2	0.01±0.01	0.46±0.19	0.12±0.03

Table 2. Comparison of Mean squared error results on three data sets (see [9] for details). Results on the test set are given for several variants of RBF networks which were used also by Orr to asses RBFs. MSE Results of an average over 100 runs including standard deviation are presented.

Next, we describe regression results for variant of well known regressors on the Pumadyn data. The Pumadyn data from the DELEVE archive [3] is generated from a simulation of the dynamic 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 amount of noise and non-linearity. The data set is divided into another two groups. Each of which is non-linear with high noise. The first group has 8 inputs ($Pumadyn8$) and the second group has 32 inputs. There are 5 sizes of the train sample set: 64, 128,256,512,1024. Testing the methods on all of these show how the method scale with the train sample size. In other words, one can see how the variance of a method behaves to different train sample size.

For these data sets, we obtain the results of several methods from the DELVE archive and compare our methods to those results. These methods are:

- **Lin-1** Linear least squares regression.
- **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 [27]. a maximum a-posteriori via conjugate gradient.

– **MARS3.6-bag-1** Multivariate adaptive regression splines (MARS) [12], version 3.6 with bagging.
– **PRBFN-AS-RBF** RBF with pruning as described in section 3.4.
– **PRBFN-AS-MLP** MLP with pruning as described in section 3.4.
– **PRBFN-LRT** Full PRBFN method LRT for pruning.
– **PRBFN2** PRBFN - Evidence model selection for unit type and BIC pruning as described in section 3.3.

Further more details can be obtained from the DELVE web site [3].

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-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
PRBFN2	**0.75**±0.11	**0.43**±0.02	**0.38**±0.01	**0.37**±0.02	**0.34**±0.01

Table 3. Regression on Pumadyn with 32 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-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
PRBFN2	**0.48**±0.03	**0.38**±0.01	**0.34**±0.01	**0.33**±0.01	**0.32**±0.01

Table 4. Regression on Pumadyn with 8 input non-linear with high noise

5 Discussion

The work presented in this paper represent a major step in constructing an incremental hybrid architecture for regression. It was motivated by the success of the original hybrid architecture which was introduced in [4, 5]. Several assumptions were made in various parts of the architecture construction. Our aim was to show that even under these assumptions, an architecture that is smaller in size and better in generalization performance can already be achieved. Furthermore, while this architecture is particularly useful when the data contain ridge and Gaussian parts, its performance was not below the performance of the best known MLP or RBF networks when data that contains only one type of structure was used.

In previous work [4, 5], we used hard threshold for unit type selection. The previous algorithm also accepted the number of hidden units in advance. This paper introduces an algorithm that automatically reveals the relevant parts of the data and maps these parts onto RBF or Ridge functions respectively. The algorithm also finds the number of hidden units for the network given only an error target. The automatic unit type detection uses the Bayesian evidence principle for regression. The pruning method suggested in this paper is applied to individual weights of each hidden unit. It produces a smaller net with better generalization. Two cirterions have been used: BIC and LRT. The BIC criterion has 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. In other words when the change in the likelihood is significant.

We have tested the new architecture construction on seven regression problems and three 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 this cases, the first step of the training that allocates many protopyes to the dynamic part successfuly modeled the function and has shown low error rates. In the LogGaus data set, which is composed of Ridge and Gaussian parts – an excellent example for our hybrid – results were again improved with our proposed architecture construction. The tests on the Pumadyn family have raised interesting observations. The Bayesian Information Criterion (BIC) is superior to the LRT. The pruning algorithm that has been described in this paper improved the generalization of regressors. However, regressors with one type of hidden units, like RBF or MLP, can be improved by using the unit type selection method we have purposed here. On Pumadyn the pruning has removed 95% of the weights. For instance for the small data sets on average only 10 weights are needed. The proposed method has always been one of the best regressors and it has very good performance on pumadyn32-nh. The proposed method appears to have the ability to better model extensively studied, nonlinear data and, in particular, demonstrate increased generalization, while keeping the number of the estimated parameters smaller.

References

1. C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.
2. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* The Wadsworth Statistics/Probability Series, Belmont, CA, 1984.
3. C.E.Rasmussen, R.M. Neal, G.E. Hinton, D. Van Camp, Z. Ghahrman M. Revow, R. Kustra, and R. Tibshirani. The delve manual. 1996.
4. S. Cohen and N. Intrator. Automatic model selection in a hybrid perceptron/radial network, 2002. The name is the same as in the proceedings.
5. 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.
6. 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.
7. D. L. Donoho and I. M. Johnstone. Projection-based approximation and a duality with kernel methods. *Annals of Statistics*, 17:58–106, 1989.
8. 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.
9. M.J.L Orr et al. Combining regression trees and radial basis functions. *Int. J. of Neural Systems*, 10(6):453–466, 2000.
10. S. E. Fahlman and C. Lebiere. The cascade–correlation learning architecture. CMU-CS-90-100, Carnegie Mellon University, 1990.
11. 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.
12. J. H. Friedman. Mutltivariate adaptive regression splines. *The Annals of Statistics*, 19:1–141, 1991.
13. J. H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76:817–823, 1981.
14. 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.
15. T. Hastie and R. Tibshirani. Generalized additive models. *Statistical Science*, 1:297–318, 1986.

16. T. Hastie and R. Tibshirani. *Generalized Additive Models.* Chapman and Hall, London, 1990.
17. David W. Hosmer and Stanley Lemeshow. *Applied Logistic Regression.* Wiley Series in Probability and Mathematical Statistics, 1989.
18. H. Pomares J. Ortega J. Gonzalez, I. Rojas and A. Prieto. A new clustering techniques for function approximation,. *IEEE Transaction on Neural Networks*, 13:132–142, 2002.
19. 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.
20. 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.
21. N. Sugie K. Suzuki, I. Horiba. A simple neural network algorithm with application to filter synthesis. *Neural Processing Letters, Kluwer Academic Publishers, Netherlands*, 13:43–53, 2001.
22. R. E. Kass and A. E. Raftery. Bayes factors. *Journal of The American Statistical Association*, 90:773–795, 1995.
23. 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.
24. R. A. Jacobs M. I. Jordan. Hierarchical mixture of experts and the EM algorithm. *Nueral Computation*, 6:181–214, 1994.
25. D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.
26. 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.
27. R. M. Neal. *Bayesian Learning for Neural Networks.* Springer, New York, 1996.
28. S. J. Nowlan. Soft competitive adaptation: Neural network learning algorithms basd on fitting statistical mixtures. Ph.D. dissertation, Carnegie Mellon University, 1991.
29. A. Papoulis. *Probbaility, Random Variables, and Stochastic Process*, volume 1. McGRAW-HILL, New York, third edition, 1991.
30. D. G. Stork R. O. Duda, P. E. Hart. *Pattern Classification.* John Wiley Sons, INC., New York, 2001.
31. G. Schwarz. Estimationg the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.
32. C. J. Stone. The dimensionality reduction principle for generalized additive models. *The Annals of Statistics*, 14:590–606, 1986.
33. P.D. Wasserman. *Advanced Methods in Neural Computing.* Van Nostrand Reinhold, New York, 1993.

A Approximation power

In this section we show that the algorithm described in this paper converges to a usefull solution. We prove that an algorithm that does not make the optimal steps converges, and hence this algorithm converges. We first assume that the algorithm uses the MLE estimation for the unit type, that is, the best forward weights are selected for w, w_0 in the unit selection procedure. We also assume that only RBF units were choosen. Otherwise, if a ridge were choosen then the network had smaller error in each step and the convergence would be assured again.

ROC A1 *Let $V = \{x_i, y_i\}_{i=1}^n$ a data set where $x_i \in R^d$. Let f be a continous function in V and assume that f is bounded on V. Let $\epsilon > 0$, and assume that all the units chosen are Gaussians. Assume that, $|f(x)| < b \forall x \in V$, and let $V_x = \{x_i\}_{i=1}^n$. The algorithm converges with rate $O(1/n)$.*

We set the approximator at each step as follows:

$$f(\hat{x}) = \sum_{i=1}^{K} y_i \exp(-\frac{||x - x_i||^2}{r_i^2}). \tag{33}$$

We set r_i as follows:

$$r_i^2 < \min_{x_j \in V_x} \frac{||x_j - x_i||^2}{|log(b) - log(\frac{\epsilon}{n})|}. \tag{34}$$

For the first Kx_i chosen by the algorithm there exist:

$$\begin{aligned} |f(x_i) - f(\hat{x_i})| &< |f(x_i) - y_i| + |\sum_{j \neq i} y_j \exp(-\frac{||x_j - x_i||^2}{r_i^2})| \\ &< 0 + \frac{K-1}{n}\frac{\epsilon}{n} < \frac{\epsilon}{n} \end{aligned} \tag{35}$$

For all the other points x_k that have not yet been selected the following hold:

$$\begin{aligned}|f(x_k) - f(\hat{x_i})| &< |y_k| + |\sum_j y_j \exp(-\frac{||x_j - x_i||^2}{r_i^2})| \\ &< b + \frac{n-K}{n}\frac{\epsilon}{n} < b + \frac{\epsilon}{n}. \end{aligned} \tag{36}$$

Thus, the sum of Squares error is:

$$\begin{aligned}\sum_{i=1}^{n}(f(x_i) - f(\hat{x_i}))^2 = \sum_{x_i, i\leq k}(f(x_i) - f(\hat{x_i}))^2 + \sum_{x_i, i>k}(f(x_i) - f(\hat{x_i}))^2 \\ < \frac{\epsilon k}{n} + \frac{b\epsilon(n-k)}{n}, \end{aligned} \tag{37}$$

and hence we see that the residual error decreases and the decrease is: $\epsilon\frac{b-1}{n}$. It follows that if we choose $b > 1$ that at each step the decrease of the error is $O(1/n)$.