Forward and Backward selection in regression hybrid network

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 (FBMS) for constructing a hybrid regression network of radial and perceptron hidden units. The algorithm determines whether 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 a RBF network or a MLP. Results for various data sizes on the Pumadyn data indicate that the resulting architecture competes and often outperform best known results for this data set.

Keywords: Hybrid Network Architecture, SMLP, Clustering, Regularization, Nested Models, Model Selection.

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 [12, 6]. The potential problem of such a construction is over flexibility which leads to over-fitting.

Analyzing cases where convergence of MLP or RBF networks (as a function of number of hidden units) is slow, reveals that often there is at least one region in 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 in input space can lead to a far smaller and more effective architecture.

Earlier approaches, attempted to construct a small network approximation to the desired function at different regions of input space via a "divide and conquer" approach. Rather than reviewing the vast literature on that, we shall point out some approaches which indicate some of the highlights that had motivated our work. Work on trees is reviewed in [1] 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 sizes. An approach which constructs more sophisticated architectures at the terminating nodes was proposed in [14, 9], where a gating network performs the division of the input space and small neural networks perform the function approximation at each region separately. Donoho's work [5] 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.

We have introduced a training methodology for a hybrid MLP/RBF network [4, 3]. This architecture, produced far better classification and regression results compared with advanced RBF methods or with MLP architectures. In this work, we further introduce a novel training methodology, which evaluates the need for additional hidden units, chooses optimally their nature – MLP or RBF – and 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, Forward and Backward Model Selection (FBMS), 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, better than regular MLP or RBF methods.

2 Parameter estimation and model selection

An incremental architecture with more than one type of building components, 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, RBF or a perceptron; (iii) train the network; (iv) prune unnecessary weights. The SMLP [6] network uses both RBF and Perceptron units at each cluster. In higher order networks [12] quadratic and linear terms always exist and strong regularization must be used to avoid over-fitting. Our proposed FBMS selects the proper unit for each region in input space. Thus, the number of hidden units remains minimal and over-fitting is reduced. In order to select the type of units, FBMS divides the space into homogeneous regions and selects the type of hidden unit for each region. During the division of the space into small regions the overall error is estimated and the splitting is stopped when an error goal is reached. Thus, the network size is monitored using the error criterion. When the training data set is small, an over-fitting may occur. Thus, a backward elimination of weights can improve the prediction and reduce the variance of the estimator. The steps in the algorithm include:

– Data clustering and splitting to reduce an error objective function.
– Automatic selection of unit type for each cluster.
– Full gradient descent on the resulting hybrid architecture.
– Pruning of unnecessary weights.

Below, we describe in detail each step.

2.1 Data clustering

We start by clustering the data based and minimizing 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, is similar to the one proposed in CART [1], it 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:

$$\Delta SSR(C_0) = SSR(C_0) - (SSR(C_1) + SSR(C_2)). \tag{1}$$

The splitting is continued until an error goal of the hybrid classifier is reached or a predefined maximum number of clusters is achieved. Since a cluster C with n members has $2^n - 1$ possible number of meaningful splits, an exhaustive search is not feasible. CART [1] solves this problem by considering each axis $1 \leq l \leq d$ of the input space separately and search 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 restricted to projections that are parallel to the axes.

Splitting rule We assume that the underlying function to be estimated is continuous. Consider a subset C_0 of the data. Let y_1 be the minimum value of the function in C_0 with a corresponding input x_1. Let y_2 be the maximum value of the function in C_0 with input x_2.

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 the pattern to C_1, otherwise to C_2.
– Choose to split the cluster with the largest reduction in the objective function.

Other distance measures such as Mahalanobis or Manhattan can be considered depending on prior knowledge about the problem.

2.2 Model selection

FBMS uses a classical statistical approach to select models. 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. Thus, the likelihood of the data given the model is:

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

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 of the SSE is equivalent to the maximization of the likelihood of the data. To obtain the maximum likelihood value of σ we differentiate 3 with respect to sigma.

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

This leads to the maximum likelihood estimate for σ:

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

We use 3 to perform model selection when the type of a hidden unit has to be selected, or 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. The Bayesian information Criterion (BIC) is used to select the type of unit and either the BIC or LRT is used for the pruning process.

We start by describing the Bayesian approach. 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 [11] which approximates the integrand by a quadratic function. Thus, the value of the integral becomes [11, 16]:

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

Where H is the Hessian matrix of the approximation and W_{m_0} is the most probable value of the likelihood $p(D|M)$. With the lack of a-priori knowledge we assume that a model with a 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: $p(D|M_1)/p(D|M_2)$.

The BIC approximation can be derived from 8 by using Gaussian distribution to the a-priori parameters density [11] 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 5 and 3 into 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 predication for a new value y as follows:

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

Equation 11 shows that the evidence of a model can be used as weight when averaging the predications of all models. When the best model gives good predications, the averaging process can be skipped. Then, the FBMS algorithm uses the single best model for predication.

The LRT can be used to select between two nested models. Given two models $M1 \subset M2$ the LRT test is defined as follows [15]:

$$-2\log(\frac{p(D, W_{m_0}|M1)}{p(D, W_{m_0}|M2)}) = \chi^2(d_2 - d_1). \tag{12}$$

This approach uses $P - Values$ to reject the null hypothesis, that is, the simple model is equivalent to the complicated one. The LRT criteria 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 5 we arrive at the following test:

$$\chi^2(d2 - d1) = 2N\log(\sigma_1^2) - 2N\log(\sigma_2^2). \tag{13}$$

2.3 Unit selection

After the decomposition of the input space into more homogeneous subsets, a unit type is selected for each such subset. Since the models are not nested the BIC criterion is applied. The maximum likelihood is computed for each cluster

and unit type and the unit type with the higher BIC value is selected. First we set the parameters of hidden units.

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. To set the forward weights the log-likelihood is computed as follows. Let C_i be the set of points of the current cluster. Let $\phi(x_i)$ be the ridge or RBF values for each $x_i \in C_i$, and let t_i be the targets. We define the objective function:

$$E(w, w_0) = \frac{1}{2} \sum_{i=1}^{N} (w^T \phi(x_i) + w_0 - t_i)^2. \tag{14}$$

The partial derivatives with respect to w and w_0 are:

$$\begin{aligned} \frac{\partial E}{\partial w} &= \sum_{x_i \in C_i} (w^T \phi(x_i) + w_0 - t_i)\phi(x_i) = 0. \\ \frac{\partial E}{\partial w_0} &= \sum_{x_i \in C_i} (w^T \phi(x_i) + w_0 - t_i) = 0. \end{aligned} \tag{15}$$

Thus, we obtain the locally forward weights of the specific unit:

$$w = \frac{n \sum_{x_i \in C_i} \phi(x_i) t_i - \sum_{x_i \in C_i} t_i \sum_{x_i \in C_i} \phi(x_i)}{n \sum_{x_i \in C_i} \phi(x_i)^2 - (\sum_{x_i \in C_i} \phi(x_i))^2}, \tag{16}$$

and

$$w_0 = \frac{\sum_{x_i \in C_i} t_i - w \sum_{xi \in C_i} \phi(x_i)}{n}. \tag{17}$$

Substituting w and w_0 into Eq. 14 gives the error value of the fit for each unit type. The error is inversely proportional to the likelihood.

The above procedure is repeated for each cluster and using 10, the most probable unit type is selected.

2.4 Pruning

The last step of the algorithm prunes unnecessary weights. This can remove inputs weight to a ridge unit, a feature of a RBF unit or a hidden unit. Using a diagonal approximation to the covariance matrix, the activation of a RBF function is given by:

$$\phi(x) = \exp(-\sum_{i=1}^{N} \frac{(x_i - c_i)^2}{\sigma_i^2}). \tag{18}$$

If we make the substitution: $\frac{1}{\sigma} = r$, we obtain:

$$\phi(x) = \exp(-\sum_{i=1}^{d} \frac{(x_i - c_i)^2 r_i^2}{2}). \tag{19}$$

Consider setting r_j to zero:

$$\phi(x) = \exp(-\sum_{i=1, i\neq j}^{d} \frac{(x_i - c_i)^2 r_i^2}{2}) \exp(0) = \exp(-\sum_{i=1, i\neq j}^{d} \frac{(x_i - c_i)^2 r_i^2}{2}).$$

The partial derivatives with respect to r_i is:

$$\frac{\partial \phi}{\partial r_i} = -\phi(x)(x_i - c_i)^2 r_i. \tag{20}$$

Thus, to prune a feature i it suffices to set r_i to zero. This eliminates two parameters. For the ridge input weights we add a matrix w_{ij} were $w_{ij} \in \{0, 1\}$.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. FMBS uses both criteria when BIC is the default one. Our pruning process is similar to [10, 8]; We start with the full model and at each step we prune the least significant weight. That is, the weight that least contributes 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, we use a default value of 95% that matches to $P = 3.841$ of the corresponding distribution for one degree of freedom.

3 Experimental results

This section describes regression results for the Pumadyn data from the DELEVE archive [2]. The 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 contains several sets with different noise levels. We have used the data set with the largest noise. This data set is divided into two groups. The first has 8 inputs ($Pumadyn8$) and the second has 32 inputs. There are 5 sizes of the train sample set: 64, 128,256,512,1024. Studying a regressor on all subsets indicates how the method scales with the train sample size.

For these data sets, we compare our results to several other methods that have been used in the past. We thus provide results for the following methods:

– **lin-1** Linear least squares regression.

– **knn-cv-1** K-nearest neighbors for regression. K is selected by using leave-one-out cross-validation.
– **mars3.6-bag-1** Multivariate adaptive regression splines (MARS) [7], version 3.6 with bagging.
– **mlp-ese-1** Multilayer perceptron ensembles, trained with early stopping implemented by conjugate gradient optimization.
– **mlp-mc-1** Multilayer perceptron networks trained by MCMC methods [13].
– **gp-map-1** Gaussian processes for regression, trained using a maximum a-posteriori approach implemented by conjugate gradient optimization.
– **PRBFN-AS-RBF** Using the a regular RBF network with Gaussians units with the pruning algorithm.
– **PRBFN-AS-MLP** Using MLP network with the pruning algorithm.
– **PRBFN-LRT** The PRBFN method with the LRT for the pruning algorithm.
– **PRBFN-FBMS** The hybrid network with BIC for the model selection in the process. Each unit type is selected as described in section 2.3.

Further details can be obtained from the DELVE web site [2].

Pumadyn32nh	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-ese-1	1.25±0.04	1.13±0.09	0.96±0.02	0.89±0.02	0.86±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
PRBFN-FBMS	0.75±0.11	0.43±0.02	0.38±0.01	0.37±0.02	0.34±0.01

Table 1. Regression on Pumadyn with 32 input non-linear with high noise

4 Discussion

The work presented in this paper represents a practical step in constructing an incremental hybrid architecture for regression. It was motivated by the success of the original hybrid architecture which was introduced in [3, 4]. 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 contains

Pumadyn8nh	64	128	256	512	1024
lin-1	0.73±0.02	0.68±0.02	0.65±0.01	0.63±0.01	0.63±0.02
knn-cv-1	0.79±0.02	0.71±0.02	0.64±0.01	0.58±0.02	0.53±0.02
mlp-ese-1	0.72±0.02	0.67±0.02	0.61±0.01	0.49±0.01	0.41±0.01
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
PRBFN-FBMS	0.48±0.03	0.38±0.01	0.34±0.01	0.33±0.01	0.32±0.01

Table 2. Regression on Pumadyn with 8 input non-linear with high noise

ridge and Gaussian parts, its performance were not below the performance of the best known MLP or RBF networks when data that contains only one type of structure was used[1].

In our previous work [4], we have used hard threshold for unit type selection. That algorithm also accepted the number of hidden units in advance. This paper introduces an algorithm that finds automatically 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 for a given error target. The automatic unit type detection uses the maximum likelihood principle for regression, and the proposed pruning method is applied to individual weights of each hidden unit. Two criteria have been used: BIC and LRT. The BIC criterion can be used for choosing the best model out of a general family of models. The LRT works only with nested models, where the process is terminated when the null hypothesis is rejected.

The tests on the Pumadyn family of data sets suggests that the Bayesian Information Criterion (BIC) is superior to the LRT. Most importantly, the proposed method can achieve better performance for smaller data set sizes. This property is very useful for problems where large training data set is difficult to obtain, e.g., medical data or, most recently, protein and gene expression data.

References

1. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* The Wadsworth Statistics/Probability Series, Belmont, CA, 1984.
2. 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.
3. S. Cohen and N. Intrator. Automatic model selection of ridge and radial functions. In *Second International workshop on Multiple Classifier Systems*, 2001.

[1] This finding is based on other data sets which are not included in this paper due to lack of space.

4. S. Cohen and N. Intrator. A hybrid projection based and radial basis function architecture: Initial values and global optimization. *To appear in Special issue of PAA on Fusion of Multiple Classifiers*, 2001.
5. D. L. Donoho and I. M. Johnstone. Projection-based approximation and a duality with kernel methods. *Annals of Statistics*, 17:58–106, 1989.
6. 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.
7. J. H. Friedman. Mutltivariate adaptive regression splines. *The Annals of Statistics*, 19:1–141, 1991.
8. 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.
9. 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.
10. 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.
11. R. E. Kass and A. E. Raftery. Bayes factors. *Journal of The American Statistical Association*, 90:773–795, 1995.
12. 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.
13. R. M. Neal. *Bayesian Learning for Neural Networks*. Springer, New York, 1996.
14. S. J. Nowlan. Soft competitive adaptation: Neural network learning algorithms basd on fitting statistical mixtures. Ph.D. dissertation, Carnegie Mellon University, 1991.
15. A. Papoulis. *Probbaility, Random Variables, and Stochastic Process*, volume 1. McGRAW-HILL, New York, third edition, 1991.
16. D. G. Stork R. O. Duda, P. E. Hart. *Pattern Classification*. John Wiley Sons, INC., New York, 2001.