Automatic model selection in a hybrid perceptron/radial network*

Shimon Cohen **Nathan Intrator**
Computer Science Department
Tel-Aviv University
www.math.tau.ac.il/∼nin

Abstract

We introduce an algorithm for incrementaly constructing a hybrid network fo radial and perceptron hidden units. The algorithm determins if a radial or a perceptron unit is required at a given region of input space. Given an error target, the algorithm also determins the number of hidden units. This results in a final architecture which is often much smaller than an RBF network or a MLP. A benchmark on four classification problems and three regression problems is given. The most striking performance improvement is achieved on the vowel data set [4].

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 architexture flexibility is via the use of more flexible units [17, 8]. The potential problem of such a construction is over flexibility which leads to overfitting.

We have introduced a training methodology for a hybrid MLP/RBF network [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

*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.

methodology, which evaluates the need for additional hidden units, chooses optimaly 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 in case of classification). This approach, coupled with optimal determination of initial weight values for the additional hidden units, constructs a computationaly efficient training algorithm which appears to scale up with the complexity of the data, 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 approximator, namely, it can approximate any L^2 function, it is also known that the approximation may be very greedy, namely the number of hidden units may grow very large as a function of the desired approximation error.

Analysing cases where convergence of the architecture (as a function of number of hidden units) is slow, reveals that oten 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 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 concur", has been studied since the eighties in the machine learning and connectionists community. Rather than reviwing 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 size. An approach which constructs more sophisticated architectures at the terminating nodes was proposed in [20, 14], 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 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 to use 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 [15]. The boosting algorithm [6] 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 only indirectly. A remote family of architectures where the function approximation is constructed incrementally is projection pursuit [10] and additive models [11, 12].

If one accepts the idea of constructing a local simple architecture to different regions in input space, then the questions becomes, which architcture family should be used. The local architecture should be as simple as possible in order to avoid overfitting to the smaller portion of regional training data. Motivated by theoretical work that have studied the duality between projection-based approximation and radial kernel methods [5], 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) [11, 12]. 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 [25], the additional degrees of freedom, require strong regularization to avoid over-fitting. Higher order networks have at least a quadratic terms in addition to the linear term of the projections [17] 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 the hidden nodes. Flake [8] 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 [8]. 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 hill-plateau and other regions that contain Gaussian bumps, as demonstrated in figure 1. The hybrid architecture [3], 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 [3] was to clusterized 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

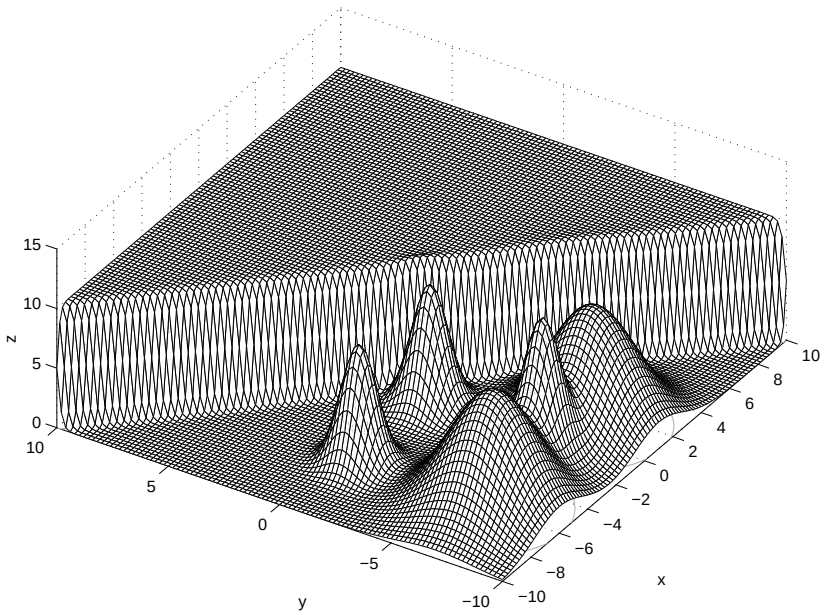

Figure 1: Data that is composed of five clusters and a sigmoidal surface.

that a data cluster is 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 new algorithm that automatically selects the type of hidden units as well as the number of units.

There are several approaches to set the structure of a Neural Network. The first one is forward selection. This approach starts with a small [19, 7] network and add units until an error goal is reached. Another approach is to start with a large network and [21] and prune un necessary units, until a given criteria is met.

In this paper we use the first approach. 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 an hybrid neural network is to find the radial and projection parts automatically. This problem in amplified for high dimension data, where the data cannot be visualized very well. We purpose a novel way to select the type of hidden unit automatically for regression and classification. This algorithm leads to smaller networks while maintaining good generalization of the resulting network.

3 Paramter estimation and model selection

An incremental architecture with more than one type of building components, requires three 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, an RBF or a perceptron; (iii) test whether the new unit is actually useful.

The SMLP [8] network uses both RBF and Perceptron units at each cluster. In higher order networks [17] 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 to regions and then to 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. 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.

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

3.1 Data clustering

We start by clustering the data and reducing an objective error function on the data. The objective function can be the Entropy, for classification, or the Sum of Square Errors (SSE) for regression problems. The entropy serves as a measure of information or surprise. The less entropy a cluster has the more uniform in class tags are the its patterns. 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.

The algorithm, that is described here, splits the cluster with the largest objective function (error value) into two clusters as follows. We feel that more work is needed in the theoretical justification of the splitting rule. This is subject for future work.

- Start with the whole data as one cluster.
- Find the cluster with the largest error.
- For regression problems use regression split and otherwise classification split.
- The splitting is continued until an error goal is reached or maximum number of clusters are achieved.

The splitting for regression divides each cluster into two regions. One region is where the target function is approximated rather good, the other one where there is still a large error. This is done as follows:

- Find the pattern with the largest error in the above cluster.
- Sort the patterns by distance to the pattern with the largest error value.
- Split the current cluster by considering $(n-1)$ ways to divide the cluster by forming a division on the sorted patterns.
- Choose the split with the largest reduction in the SSE.

The splitting procedure for classification problems minimizes the Entropy criterion. Thus, splitting the cluster with the largest Entropy split the cluster with maximum impurity. Breiman [1] has used the Gini criterion for splitting nodes in the CART algorithm. He has found that the two criterions are equivalents. We purpose the following splitting procedure for a given cluster:

- Select the two classes with the maximum number of patterns.
- Compute their mean and form two new clusters of these means.
- For each pattern associate it with the nearest mean and its cluster.

The above splitting procedures are simple and there is no need to work through every coordinate as done in the CART [1] algorithm.

3.2 Unit selection

Several authors have used the Bayesian approach for model selection. Mackay[18] uses the evidence to select a model out of a set of models. Kass and Raftery [16] consider the Bayes Factors, for given models $M1, M2$ and data set D. We follow this approach here by starting with the Bayesian formulation which computes the most probable model by integrating over the model parameters:

$$p(D|M) = \int_w p(D, w|M)dw = \int_w p(D|w, M)p(w|M)dw. \tag{2}$$

The Bayes Factors are then defined as:

$$\frac{p(M1|D)}{p(M2|D)} = \frac{p(D|M1)p(M1)}{p(D|M2)p(M2)}. \tag{3}$$

The integration of 2 can be performed by using Laplace integral. That is, approximating the integrand by a quadratic function. Thus, the value of the integral becomes:

$$p(m|D) \cong (2\pi)^d |H|^{-1|2} p(D|W_{m_0}, M)p(W_{m_0}|H), \tag{4}$$

Where H is the Hessian matrix of the approximation and W_{m_0} is the most probable value of the likelihood $p(D|M)$. Note that this calculation takes into account the performance of the model in the vicinicty 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:

$$p(M1) = p(M2).$$

This leads to the likelihood ratio:

$$\frac{p(D|M1)}{p(D|M2)}.$$

The purposed algorithm selects the type of hidden unit automatically by using likelihood ratio. The maximum likelihood is computed for each cluster and unit type. The unit type with the higher likelihood is selected. The maximum likelihood is defined differently for classification and regression problems and it is described in the next sub-sections.

3.2.1 Regression unit type selection

In the regression context the maximum likelihood is defined as follows:

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

Where t_n are the target values and y_n is the output the neural network. Maximization of the likelihood is equivalent to the minimization of the sum of squares, if one assumes that the function values are corrupted with a noise that is normally distributed with given variance σ^2. This assumption is plausible when the noise is a sum of signals from independents source according to the Central Limit Theorem.

As noted above, the selection of the unit type is done for each cluster and the above computation is repeated for each cluster.

To decide between the two possible units, we project the data eigher using an RBF or a ridge, thus consider two 1-D data fiting problems; The ridge projection is a monotonic 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 weight vector should be proportional to the average over all patterns in the cluster (there is a strong assumption here about the lack of effect of the sigmoidal term on the optimization, but remember that this is only to choos optimal initial conditions, and perform model selection.)

The RBF function, on the other hand, is monotonic decreasing with the distance from the maximum point. Thus, the center of the RBF is located at the function maximum point. Selection of the value that maximizes the likelihood in this case is trivial. Finally, the unit type with the higher likelihood is selected for the current cluster.

The above calculation did not take into account the sign of the forward parameter that connects the hidden unit to the ouput layer. In order to take it into account, we need to calculate the values that maximze and minimize the likelihood for each unit, and then calculate the sign of the forward connection (using the simple inverse procedure of the hidden unit activity matrix) and choose the values which are consistent with the sign of the forward connection. In other words, if the forward connection turns out to be positive, the value which maximized the likelihood should be chosen and vice versa.

Thus, the above procedure is repeated with the above transformation on each of the target function values. Finally, the most probable unit type of both cases is chosen.

3.2.2 Classification unit type selection

In classification, the target function has multiple values for each pattern. Thus, the previous technique can not be directly applied. For simplicity, we assume that the clustering which has been performed in the previous step, has purified the clusters, namely, each cluster mostly contains patterns from a single class. Under this assumption, it is reasonable to use the likelihood of the data points within the cluster as an indication of the fit of different unit types. Note, that this assumption may be too strong, especially in the early stages of incrementaly constructing an architecture. Relaxing this assumption is a subject for future work.

To derive the most probable value of the weights, a linear approximation for the weights of the ridge function is used. Thus, the ridge function is approximated by $w^T x$. Hence, we wish to maximize the scalar product of w and the sum of the patterns in the current cluster. Since this is an unconstrained optimization, a Lagrange multiplier is introduced to enforce the following constrain:

$$w^T w = 1,$$

arriving at the following objective function:

$$L(w, \alpha) = \sum_{i=1}^{N} w^T x_i + \alpha(w^T w - 1), \tag{6}$$

where N is the number of patterns in the current cluster. The partial derivative with respect to the weight vector is:

$$\frac{\partial L}{\partial w} = \sum_{i=1}^{N} x_i + 2\alpha w, \tag{7}$$

and the partial derivative with respect to α is

$$\frac{\partial L}{\partial \alpha} = \sum_{i=1}^{d} w_i^2 - 1. \tag{8}$$

For convenience, let $Z = \sum_{i=1}^{N} x_i$. Setting Equation 7 to zero gives:

$$Z = -2\alpha w.$$

Squaring both sides and using (8) gives:

$$\| Z \|^2 = 4\alpha^2.$$

Thus, we obtain:

$$2\alpha = \pm \| Z \|,$$

or,

$$w = \pm \frac{Z}{\| Z \|}. \tag{9}$$

The Hessian, which is derived from Equation 7, provides the correct sign of w_j and ensures the maximization procedure:

$$\frac{\partial^2 J}{\partial w^2} = 2\alpha I. \tag{10}$$

Thus, the Hessian is a diagonal matrix, and it is negative when α is negative, leading to setting w as follows:

$$w = \frac{Z}{\| Z \|}. \tag{11}$$

The response of an RBF unit is proportional to the distance of patterns from its center. Thus, we seek to minimize the the sum of distances of the patterns from an unknown vector. Let us define the following objective function:

$$L(m) = \sum_{i=n}^{N} (x_n - m)^2, \tag{12}$$

where x_n is a pattern vector in the current cluster and m is an unknown vector. The partial derivative with respect to m is

$$\frac{\partial L}{\partial m} = -2 \sum_{i=1}^{N} (x_i - m). \tag{13}$$

Equating to zero we arrive at:

$$m = \frac{1}{N} \sum_{i=1}^{N} x_i. \tag{14}$$

Thus, we have the most probable values of the center of the RBF and weight of the Ridge function. We, now compute the likelihood of the cluster data for the two models and select the most likely model. The likelihood is defined as

$$p(D/M) = \prod_{i=1}^{N} p(xi|C), \tag{15}$$

where

$$p(xi|C) = \frac{1}{1+\exp(-w^T x)}$$

for ridge function, and

$$p(xi|C) = \exp(\frac{- \parallel x - m \parallel^2}{2\sigma^2})$$

for RBF function. However, since the ridge function is an improper probability density function, that is:

$$\int_{-\infty}^{\infty} p(x|C)dx \neq 1$$

We normalize it by the factor:

$$\sum_{i=1}^{N} p(x_i|C),$$

Where N is the number of the patterns in the data set.

4 Experimental results

This section describes regression and classification results of several variants of RBF and the proposed PRBFN architecture on several data sets. Since this paper extends our first extension into hybrid MLP/RBF network [3], we shall denote the architecture resulting from the previous work as PRBFN and results from the construction presented in this work as PRBFN2. The following results are given on the test portion of each data set (full details are in [3]). They represent are an average over 100 runs and include standard error.

4.1 Regression

The LogGaus data set is a composition of one ridge function and three Gaussian-s as follows 1:

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

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 is composed of 441 points and it is divided randomly into two sets of 221 and 220 points each. The first set is served 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 Gaussian-s 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 Gaussian-s of 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 as 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 [21, 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 [9]. 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 [21] 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-Orr	0.02 ±0.14	0.91 ±0.19	0.12±0.03
Rbf-Matlab	-	0.74±0.4	0.2 ±0.03
Rbf-Bishop	0.02 ±0.02	0.53 ±0.19	0.18 ±0.02
PRBFN	0.02 ±0.02	0.53 ±0.19	0.15 ±0.03
PRBFN2	0.01±0.01	0.49 ±0.23	0.118±0.03

Table 1: Comparison of Mean squared error results on three data sets (see [21] 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.

4.2 Classification

We have used several data sets to compare the classification performance of the proposed methods to other RBF networks. The sonar data set attempts to distinguish between a mine and a rock. It was used by Gorman and Sejnowski [22] in their study of the classification of sonar signals using neural networks. The data has 60 continuous inputs and one binary output for the two classes. It is divided into 104 training patterns and 104 test patterns. The task is to train a network to discriminate between sonar signals that are reflected from a metal cylinder and those that are reflected from a similar shaped rock. There are no results for Bishop's algorithm as we were not able to get it to reduce the output error. Gorman and

Algorithm	Sonar	Vowel	waveform
RBF-Orr	71.7±0.5	–	–
RBF-Matlab	82.3±2.4	51.6±2.9	83.8±0.2
RBF-Bishop	–	48.4±2.4	83.5±0.2
PRBFN	91±2	67±2	85.8±0.2
PRBFN2	91.3±2	68±2	85±0.3

Table 2: Percent classification results of different classifiers variants on three data sets.

Sejnowski report on results with feed-forward architectures [24] using 12 hidden units. They achieved 90.4% correct classification on the test data with the angle dependent task. This result outperforms the results obtained by the different RBF methods, and is only surpassed by the proposed hybrid RBF—FF network.

The Deterding vowel recognition data [4, 8] is a widely studied benchmark. This problem may be more indicative of the type of problems that a real neural network could be faced with. The data consists of auditory features of steady state vowels spoken by British English speakers. There are 528 training patterns and 462 test patterns. Each pattern consists of 10 features and it belongs to one of 11 classes that correspond to the spoken vowel. The speakers are of both genders. The best score so far was reported by Flake using his SMLP units. His average best score was 60.6% [8] and was achieved with 44 hidden units. Our algorithm achieved 68% correct classification with only 27 hidden units. As far as we know, it is the best result that was achieved on this data set. The waveform data set is a three class problem which was constructed by Brieman to demonstrate the performance of the Classification and Regression Trees method [1]. Each class consists of a random convex combination of two out of three waveforms sampled discretely with added Gaussian noise. The data set contains 5000 instance, and 300 are used for training. Recent reports on this data-set can be found in [13, 2]. Each used a different size training set. We used the smaller training set size as in [13] who report best result of 19.1% error. The Optimal Bayes classification rate is 86% accuracy, the CART decision tree algorithm achieved 72% accuracy, and Nearest Neighbor Algorithm achieved 38% accuracy. PRBFN has achieved 85.8% accuracy on this data set. There is not much room for improvement over the PRBFN classifier, in this example.

Table 2 summarizes the percent correct classification results on the different data sets for the different RBF classifiers and the proposed hybrid architecture. As in the regression case, the STD is also given however, on the seismic data, due to the use of a single test set (as we wanted to see the performance on this particular data set only) the STD is often zero as only a single classification of the data was obtained in all 100 runs.

The Protein data set imposes a difficult classification problem since originally the number of instance of each class diverse significantly. The input dimension of the patterns is 20 and there are 2255 patterns and two classes. The data set is divided to 1579 patterns in the train set and 676 patterns in the test set. The first class in the train set has only 340 instance and the second one has 1239 instances. Thus, the a-priori probability of the first class is 0.2153

Algorithm	OKProb class1	OKProb class2	Total OKProb
RBF-Orr	– ± – –	–	–
RBF-Matlab	– ± – –	– ± – –	– ± – –
RBF-Bishop	75.18±3	79.49±1.5	78.6±2
PRBFN2	77.4±5	80.11±2	79.56±3

Table 3: Percent classification results of different classifiers variants on the Protein data sets.

while the a-priori probability of the second class is 0.7847. To overcome this problem we re-sample the first class patterns with normally distributed noise with mean zero and 0.01 variance. This data was not used in [3].

5 Discussion

The work presented in this paper represent a major step in constructing an incremental hybrid architecture. It was motivated by the success of the original hybrid architecture which was introduced in [3]. Several assumptions were made in various parts of the architecture construction. Some of them are more justified and some require further refinement which is the subject of future work. 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 architexture is particularly useful when the data contain 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 strucutre was used.

In previous work [3] 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 reveals 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 given only an error target. The automatic unit type detection uses the maximum likelihood principle in different manner for regression and classification. In regression the connection between the likelihood to the SSE is long known and used. In the classification case the output target function is not continuous and not scalar. The ridge function is an improper probability density function (PDF) and a normalization is made to transfer it into a PDF like function.

We have tested the new architecture construction on three regression problems and four classification problems. There are three cases where better results were obtained. In the extensively studied vowel data set, the proposed hybrid architecture achieved average results which are superior to the best known results [23] and uses a smaller number of hidden units. On the waveform classification problem [1], our results are close to the Bayes limit for the data and are better than the current known results. 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. We are excited about the ability to better model extensively studied, nonlinear data, in particular, demonstrate increased generalization, while keeping the number of the estimated parameters smaller.

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] J. Buckheit and D. L. Donoho. Improved linear discrimination using time-frequency dictionaries. Technical Report, Stanford University, 1995.

[3] S. Cohen and N. Intrator. A hybrid projection based and radial basis function architecture. In J. Kittler and F. Roli, editors, *Proc. Int. Workshop on Multiple Classifier Systems (LNCS1857)*, pages 147–156, Sardingia, June 2000. Springer.

[4] D. H. Deterding. *Speaker Normalisation for Automatic Speech Recognition*. PhD thesis, University of Cambridge, 1989.

[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] 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.

[7] S. E. Fahlman and C. Lebiere. The cascade–correlation learning architecture. CMU-CS-90-100, Carnegie Mellon University, 1990.

[8] 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.

[9] J. H. Friedman. Mutltivariate adaptive regression splines. *The Annals of Statistics*, 19:1–141, 1991.

[10] J. H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76:817–823, 1981.

[11] T. Hastie and R. Tibshirani. Generalized additive models. *Statistical Science*, 1:297–318, 1986.

[12] T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman and Hall, London, 1990.

[13] T. Hastie, R. Tibshirani, and A. Buja. Flexible discriminant analysis by optimal scoring. *Journal of the American Statistical Association*, 89:1255–1270, 1994.

[14] 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.

[15] 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.

[16] R. E. Kass and A. E. Raftery. Bayes factors. *Journal of The American Statistical Association*, 90:773–795, 1995.

[17] 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,*, 22:276–306, 1986.

[18] D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

[19] 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.

[20] S. J. Nowlan. Soft competitive adaptation: Neural network learning algorithms basd on fitting statistical mixtures. Ph.D. dissertation, Carnegie Mellon University, 1991.

[21] M. J. Orr, J. Hallman, K. Takezawa, A. Murray, S. Ninomiya, M. Oide, and T. Leonard. Combining regression trees and radial basis functions. Division of informatics, Edinburgh University, 1999. Submitted to IJNS.

[22] Gorman R. P. and Sejnowski T. J. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Network*, pages 75–89, 1988. Vol. 1.

[23] A. J. Robinson. *Dynamic Error Propogation Networks.* PhD thesis, University of Cambridge, 1989.

[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1, pages 318–362. MIT Press, Cambridge, MA, 1986.

[25] C. J. Stone. The dimensionality reduction principle for generalized additive models. *The Annals of Statistics*, 14:590–606, 1986.