To appear: Special issue of Information Fusion on multiple experts

Automatic model selection in a hybrid perceptron/radial network*

Shimon Cohen **Nathan Intrator**†
School of Computer Science
Tel-Aviv University
Ramat Aviv 69978, Israel
www.math.tau.ac.il/∼nin

Revised, April 2002

Abstract

We provide several enhancements to our previously introduced algorithm for a sequential construction of a hybrid network of radial and perceptron hidden units [6]. At each stage, the algorithm sub-divides the input space in order to reduce the entropy of the data conditioned on the clusters. The algorithm determines if a radial or a perceptron unit is required at a given region of input space, by using the local likelihood of the model under each unit type. Given an error target, the algorithm also determines the number of hidden units. This results in a final architecture which is often much smaller than an RBF network or an MLP. A benchmark on six classification problems is given. The most striking performance improvement is achieved on the vowel data set [8].

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 the network size. A more common approach to achieving higher architecture flexibility is via the use of more flexible units [27, 16]. 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 [6]. This architecture, produced better classification and regression results when compared with advanced Radial Basis Functions (RBF) methods or with Multi Layer Perceptron (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 in case of classification). 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, better than regular MLP or RBF methods. The algorithm proposed here can be used for model selection in different architectures as well, e.g., thin spline units for RBFs. The convergence properties of the hybrid architecture are assured by the convergence properties of each of its components and the fact that there is a

*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.

†Corresponding author, current address: Box 1843, Brown University, Providence, RI 02912, USA, Phone 401-863-3857, Fax 401-863-3494; email Nathan_Intrator@brown.edu

global gradient descent used during the training procedure. The efficiency of the architecture is demonstrated empirically by the small size of the resulting network and its superior generalization performance.

2 Motivation for incremental methods and hybrid networks

There are different 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.

Analyzing cases where convergence of the architecture (as a function of number of hidden units) is slow [2], 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 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, we shall point out some approaches which indicate some of the highlights that had motivated our work. Work on trees is reviewed in [3] where the goal is to reach a good division of the input space and use a 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 [30, 24], where a gating network performs the division of the input space and small neural networks perform the function approximation at each region separately. Nowlans' experiments with such architecture led him to the conclusion that it is better to have different types of architectures for the gating network and for the networks that perform the function approximation at the different regions [30]. 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 [25]. The boosting algorithm [12] is another variant of the space division approach, where the division is done based on the error performance of the given architecture. In contrast to previous work, this approach takes into accounts the geometric structure of the input data only indirectly. A remote family of architectures where the function approximation is constructed incrementally is the projection pursuit [19] and the additive models [20, 21].

If one accepts the idea of constructing a local simple architecture for 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 have studied the duality between projection-based approximation and radial kernel methods [11], we have decided to use RBF or perceptron units. Donoho's work [11] 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) [20, 21]. 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 [36], the additional degrees of freedom, require strong regularization to avoid over-fitting. Higher order networks have at least quadratic terms in addition to the linear term of the projections [27] 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 [16] 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 [16]. 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. Due to the structure of the hidden units, 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.

The hybrid architecture [6], which we call Perceptron Radial Basis Net (PRBFN), automatically finds the relevant functional parts from the data concurrently, thus avoiding possible local minima that results from sequential methods. The first training step in the previous approach [6] 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 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.

In this work, we adapt a CART-like method for input space separation [3]. This is used to find regions in input space where error is high and an additional unit, either an RBF or an MLP can help reduce the error. We then introduce a local objective function and local likelihood ratio to improve the selection algorithm between RBF or perceptron units for a given region. Next, we introduce a stopping rule based on the data for addition of hidden units, eliminating the need to pre-specify the number of hidden units. This results in a smaller architecture with an automatic model selection, which performs as well and often better than the previous hybrid architecture which we had optimized manually [6].

There are several approaches to set the structure of a Neural Network. The first one is forward selection. This approach starts with a small [29, 14] network and add units until an error goal is reached. Another approach is to start with a large network and [13] 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 or a confidence level is achieved. 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 is amplified for high dimensional data, where the data cannot be visualized very well. We propose a novel way to select the type of hidden unit automatically for classification. A similar algorithm can be proposed for regression. However, we will concentrate on classification in this work. The proposed 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 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) apply global optimization on all of the parameters.

The SMLP [16] network uses both RBF and Perceptron units at each cluster. In higher order networks [27] quadratic and linear terms always exist and strong regularization must be used to avoid over-fitting. We prefer to try to select the proper unit for each region in input space. The splitting process is stopped when the number of patterns in a given region is lower than a predefined number. 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 sub-regions and select the appropriate type of hidden unit for each region. During the division of the space into sub-regions, we can estimate the overall error and stop the splitting process when an error goal is reached or a confidence level is achieved. We therefore achieve network size control that is data driven. 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 based on the class labels. This is done by minimizing a sub additive objective function Er, Entropy for example. The entropy aims to purify the class label structure of clusters and is defined as follows:

$$H = -\sum_{i=1}^{n} p_i log(p_i) \tag{2}$$

where p_i is the probability of class i in a given cluster.

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 $i \neq j$. The following algorithm, is similar to the one proposed in CART [3], it splits the cluster with the largest objective function reduction into two clusters. Let $Er(C_0)$ be the value of the objective function on the cluster C_0. Consider a split for a cluster C_0 into two clusters C_1 and C_2. Let the objective function reduction defined as follows:

$$\Delta Er(C_0) = Er(C_0) - (Er(C_1) + Er(C_2)). \tag{3}$$

Note, that the definition of $Er(C_0)$ takes into account the empirical probability of C_0 which is given by:

$$\hat{P}_{C_0} = |C_0|/|D|.$$

Since a cluster C with n members has $2^n - 1$ possible number of meaningful splits, an exhaustive search is not feasible. CART [3] 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 and the time complexity to $O(dnlog(n))$. We seek a similar approach which is not as restricted as CART to projections that are parallel to the axes. To demonstrate our approach, we consider specific objective function, for data splitting and classification.

3.2 Objective function and splitting rule for classification

In this section we want to expand the splitting rule to include directly the class labels in the data. We basically attempt to split along the discriminant directions in the data. Following an application of the entropy splitting rule described above or in its simpler form the Gini rule [3], we study the local cluster properties using discriminant analysis. Fisher linear discriminant analysis [15] gives a discriminating hyperplane between two classes. For completeness, we provide here the equations for computing the hyperplane w and its shift from the origin w_0, as the latter is slightly different than Fisher's original calculation. Consider two subsets V_1, V_2 with their mean give by:

$$m_i = \frac{1}{N_i} \sum_{x \in V_i} x.$$

Let $y_i \in \{-1, 1\}$ be the class labels corresponding to these subsets respectively. The scatter matrices are defined as follows:

$$\begin{aligned} S_i &= \sum_{x \in V_i} (x - m_i)(x - m_i)^T, \\ S_w &= S_1 + S_2. \end{aligned} \tag{4}$$

The projection vector w which defines the discriminating hyperplane is given by:

$$w = S_w^{-1}(m1 - m2). \tag{5}$$

Full details are given e.g., in [33]. The shift of the hyperplane from the origin is computed using the following relation:

$$w^T x + w_0 = 0.$$

We define the following objective function:

$$E = \sum_{i=1}^{n} (w^T x_i + w_0 - y_i)^2.$$

Given w we can now minimize E to obtain w_0:

$$\frac{\partial E}{\partial w_0} = \sum_{i=1}^{n} (w^T x_i + w_0 - y_i) = 0. \tag{6}$$

To arrive at:

$$w_0 = \frac{\sum_{i=1}^{n} y_i - w^T \sum_{i=1}^{n} x_i}{n}. \tag{7}$$

The above calculation requires the inversion of the matrix S_w^{-1}. This inversion can be avoided using the following calculation; We find a projection such that the SSE is minimized and the distance between the centers is maximized. A projected cluster center is given by:

$$\frac{1}{n} \sum_{i=1}^{n} w^T x_i = w^T \frac{1}{n} \sum_{i=1}^{n} x_i.$$

Thus, we wish to maximize the following criterion:

$$S = \frac{1}{2} \sum_{i=1} (w^T x_i + w_0 - y_i)^2 + \frac{1}{2} (w^T m_1 - w^T m_2)^2, \tag{8}$$

where m_1 and m_2 are the mean vectors of the patterns from the first and the second class respectively. Deriving equation 8 and equating to zero we arrive at:

$$w_0 = \sum_{i=1}^{n} y_i - w^T \sum_{i=1}^{n} x_i, \tag{9}$$

$$w = \frac{\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} yi/n}{\sum_{i=1}^{n} x_i^T x_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i/n - \parallel m1 - m2 \parallel^2}. \tag{10}$$

Thus, we propose the following splitting procedure for a given cluster:

- Select the two classes with the maximum number of patterns.
- Use equation 10 and equation 9 to compute w and w_0.
- For each pattern x_i compute: $w^T x_i + w_0$ if the result is positive associate it with the first cluster and otherwise with the second one.

The above splitting rules are simple to implement and can perform data splits that are not parallel to the feature axes. The time complexity of this procedure is only $O(n \times d)$, with d being the data dimensionality and n the number of patterns.

3.2.1 Stopping criterion for the splitting

The stopping rule for the splitting process described above, seeks to determine if a candidate split is meaningful - that is, whether it differs significantly from a random split. Since we split among the two prominent classes, we consider only patterns from these two classes. Consider a split that sent N_l patterns to the left node. Let N_l^i be the number of patterns sent to the left node of class i by the splitting rule. Let $E[N_l^i]$

be the expected number of patterns from a random split. We define the deviation of the results due to our candidate split by means of $chi - squared$ statistic:

$$\chi^2 = \sum_{i=1}^{2} \frac{(N_l^i - E[N_l^i])^2}{E[N_l^i]} \tag{11}$$

When the value of χ^2 vanishes the splitting rule is random and not informative. If the candidate split does not yield a χ^2 exceeding the chosen confidence level, splitting is stopped. A typical confidence level would be 0.05 (and this is the default level for our method). Alternatively, the splitting can be continued until the sum of squared error of all the nodes reaches a predefined value or a predefined maximum number of clusters is achieved.

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. We formulate this problem as a classical model selection task and thus apply a commonly used formulation to do this; Mackay [28] used the evidence for model selection. Kass and Raftery [26] utilize Bayes Factors for model selection.

For this problem we have found that the maximum likelihood model selection suffice. We wish to choose the model that best explains the training data. This approach is described below.

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 $p(M)$. The likelihood factors are then defined as:

$$\frac{p(M_1|D)}{p(M_2|D)} = \frac{p(D|M_1)p(M_1)}{p(D|M_2)p(M_2)}. \tag{12}$$

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 likelihood ratio:

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

In our case the maximum likelihood is computed for each cluster and unit type. The unit type with the higher likelihood is selected.

For simplicity, we assume that the clustering which has been performed in the previous step, has partly purified the clusters, namely, each cluster mostly contains patterns of a single class. Under this assumption, it is reasonable to use the likelihood of the data points of single class within the cluster as an indication of the fit of different unit types. When the assumption does not hold, more refined clustering is needed.

3.4 Determining the initial perceptron-unit weights

For completeness, we present here the estimation of the initial weight values for perceptron units, which were introduced in [6]. This is done by considering a linear approximation for the weights of the projection. 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:

$$L(w, \alpha) = \sum_{i=1}^{N} w^T x_i + \alpha(w^T w - 1), \tag{13}$$

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

$$\frac{\partial L}{\partial w} = \sum_{i=1}^{N} x_i + 2\alpha w, \tag{14}$$

and the partial derivative with respect to α is

$$\frac{\partial L}{\partial \alpha} = \sum_{i=1}^{d} w_i^2 - 1. \tag{15}$$

For convenience, let $Z = \sum_{i=1}^{N} x_i$. Setting Equation 14 to zero gives:

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

Squaring both sides and using (15) gives:

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

Thus, we obtain:

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

or,

$$w = \pm \frac{Z}{\| Z \|}. \tag{16}$$

The Hessian, which is derived from Equation 14, provides the correct sign of w_j and ensures the maximization procedure:

$$\frac{\partial^2 J}{\partial w^2} = 2\alpha I. \tag{17}$$

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

$$w = \frac{Z}{\| Z \|}. \tag{18}$$

3.5 Determining the likelihood of a cluster under a perceptron or RBF model

While a cluster may not be radially symmetric or even Gaussian, it may still be possible that an RBF unit will do a better job than an MLP unit at the vicinity of the cluster. We therefore replace the criterion we introduced in [6] which tested for a Gaussian distribution, with the following criterion which compares likelihoods under the two models.

Under independence assumption between the observations, the likelihood is given by

$$p(D|M) = \prod_{i=1}^{N} p(x_i|C), \tag{19}$$

where

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

for ridge function, and

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

with m given by $1/N \sum_{i=1}^{N} x_i$, for an RBF function. However, since the ridge function is an improper probability density function (PDF), that is:

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

We thus, 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 results for three variants of RBF, as well as, our first extension to a hybrid architecture (PRBFN) [6] and the present hybrid architecture, which includes model selection, clustering and initial parameter estimation (PRBFN2).

Orr's RBF [13] method ($RBF - Reg - Tree$) is based on regression tree for clustering. This methods builds a large tree and then prunes it using model selection criteria. Matlab's RBF package ($RBF - OLS$) implements an incremental algorithm [37], 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 [7] 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 [23] algorithm ($LogistReg$). $LogistReg$ and $RBF - Reg - Tree$ work only on two class classification problems. The following results are given on the test portion of each data set and represent an average over 100 runs with different initial conditions and in some cases with different splits for train and tests. The standard errors indicate variability over those runs. Table 1 summarizes the classification results (in percentage) on the different data sets for the different RBF and MLP classifiers and the proposed hybrid architecture.

Algorithm	Sonar	Vowel	Waveform	Hepatitis	Iris	Letters
BP-Lev-Marq	90.4±0.5	51±3.0	70.3±6	75.4 ±7	88.8±9.4	–
LogistReg	74.0±0	–	–	80.8 ±0	–	–
RBF-Reg-Tree	71.7±0.5	–	–	79.8±5	–	–
RBF-OLS	82.3±2.4	51.6±2.9	83.8±0.2	82.7±3	96±2.4	–
RBF-EM	–	48.4±2.4	83.5±0.2	77.3±3	95.5±1.7	85.49±2.0
PRBFN	91.3±2.1	67.0±2.1	85.8±0.2	82.1±4	95.8±2.7	85.5 ±1.9
PRBFN2	92.3±1.9	68.4±1.9	85.8±0.3	84.8±4	96.8±1.1	94.02 ±0.0

Table 1: Percent classification results of different classifiers variants on six data sets.

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 [31] 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 Sejnowski report on results with feed-forward architectures [35] 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 PRBFN2 network, that used only 10 hidden units. The previous version PRBFN used 12 hidden units on this problem.

The Deterding vowel recognition data [8, 16] 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% [16] and was achieved with 44 hidden units. Our new algorithm (PRBFN2) achieved 68.4% correct classification with only 22 hidden units. For this problem PRBFN needed 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 [3]. 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 instances, and 300 are used for training. Recent reports on this data-set can be found in [22, 4]. Each of these reports used a different size training set. We used the smaller training set size as in [22] who report 80.9% correct classification. The Optimal Bayes classification rate is 86% correct classification, the CART decision tree algorithm achieved 72%, and Nearest Neighbor Algorithm achieved 38%. PRBFN2 achieved 85.8% correct classification on this data set.

The Hepatitis data set is a two class classification problem and the input vector space has 19 attributes. The task is to predict if a patient will survive. Diaconis and Efron [9] achieved 80% correct classification on this dataset and Cestink et al. [5] achieved 83%. The Iris data set [15] contains three classes, each with 50 instances. The classes refer to a type of iris plant. Each pattern is composed of four attributes. We divided the data set into two halves one for train and one for test. This procedure is repeated 100 times. The Letters data set [18] contains 20,000 patterns. The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15. We trained on the first 16000 patterns and tested the model on the remaining 4000. Frey et al. result [18] was a little over 80%, PRBFN2 achieved 93% correct classification. Other results over 90% are reported in [10, 17, 32]. Since this data set is large some of the classifiers were able run.

5 Discussion

The work presented in this paper extends our earlier work on hybrid architectures [6]. 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 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 e.g., the Sonar or Iris data-sets.

In previous work [6] we used fixed threshold for unit type selection and a predefined number of hidden units. This paper introduced an algorithm that finds automatically the relevant parts of the data (via tree-like clustering) and maps these parts onto RBF or Ridge functions using a likelihood test. The algorithm also finds the number of hidden units for the network given only an error target. The automatic unit type selection uses the maximum likelihood principle in different manner for classification.

The proposed network construction and training method, has several means to avoid overfitting: (i) The number of patterns in each region can not be below a certain predefined number. (ii) A confidence level is used to determine whether a new split is performed, thus, the flexibility and size of network can be controlled by this parameter. (iii) The fact that either a radial basis function or a projection unit is fit locally to the data reduces overfitting as a better fit to the data is more likely. (iv) RBF optimization, as well as full gradient descent optimization on the hybrid network, are done via a constrained minimization. (v) An error goal is pre set for the mean square error, in order to enable early stopping. (vi) Unit type selection relies on likelihood ratio and not absolute numbers, ensuring that the better unit will be chosen to locally fit the data.

We have tested the new architecture and training algorithm on six classification problems. The resulting network does appear to be smaller than competing architectures, validating the usefulness of the overfitting avoidance methods. There are two cases where significant improvement was obtained. In the extensively studied vowel data set, the proposed hybrid architecture achieved average results which are superior to the best known results [34] while using a smaller number of hidden units. On the waveform classification problem [3], our results are close to the Bayes limit for the data and are better than the current known results.

In summary, the proposed method appears to have the ability to better model nonlinear data, in particular, demonstrate increased generalization, while keeping the number of the estimated parameters smaller. The potential flexibility is greater than in an MLP or RBF networks as the architecture includes both these models, however the strong regularization keeps the model smaller than other competing models.

Acknowledgments

We would like to thank the anonymous revierwers for making very useful comments which improved the exposition of this work.

References

[1] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[2] M. Brady, R. Raghavan, and J. Slawny. Back propagation fails to separate where perceptrons succeed. *IEEE Trans. Circuits and Systems*, 36(5):665–674, May 1989.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* The Wadsworth Statistics/Probability Series, Belmont, CA, 1984.

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

[5] G. Cestnik, I. Konenenko, and I. Baratko. Assistant-86: A knowledge-elicitation tool for sophisticated users. In I. Bartko and N. Lavarac, editors, *Progress in Machine Learning.* Sigma Press, 1987.

[6] 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.

[7] 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.

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

[9] P. Diaconis and B. Efron. Computer-intensive methods in statistics. *Scientific American*, 248, 1983.

[10] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error correcting code. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[11] D. L. Donoho and I. M. Johnstone. Projection-based approximation and a duality with kernel methods. *Annals of Statistics*, 17:58–106, 1989.

[12] H. Drucker, R. Schapire, and P. Simard. Improving performance in neural networks using a boosting algorithm. In Steven J. Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 42–49. Morgan Kaufmann, 1993.

[13] M.J.L Orr et al. Combining regression trees and radial basis functions. *Int. J. of Neural Systems*, 10(6):453–466, 2000.

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

[15] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[16] 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.

[17] T. C. Fogarty. First nearsest neighbor classification on frey and slate's letter recognition problem. *Machine Leraning*, 9:387, 1994.

[18] P. W. Frey and D. J. Slate. Letter recognition using holland-style adaptive classifiers. *Machine Learning*, 6, 1991.

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

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

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

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

[23] David W. Hosmer and Stanley Lemeshow. *Applied Logistic Regression*. Wiley Series in Probability and Mathematical Statistics, 1989.

[24] 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.

[25] 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.

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

[27] 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.

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

[29] 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.

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

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

[32] D. Partridge and W. B. Yates. Data-defined problems and multiversion neural-net systems. *Journal of Intelligence Systems*, 7(1-2):19–32, 1997.

[33] D. G. Stork R. O. Duda, P. E. Hart. *Pattern Classification*. John Wiley Sons, INC., New York, 2001.

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

[35] 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.

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

[37] P.D. Wasserman. *Advanced Methods in Neural Computing*. Van Nostrand Reinhold, New York, 1993.