10

When networks disagree: Ensemble methods for hybrid neural networks

MICHAEL P. PERRONE and LEON N. COOPER
Physics Department, Neuroscience Department, Institute for Brain and Neural Systems, Box 1843, Brown University, Providence, RI 02912, USA

10.1 Introduction

Hybrid or multi-neural network systems have frequently been employed to improve results in classification and regression problems (Cooper, 1991; Reilly *et al.*, 1988, 1987; Scofield *et al.*, 1991; Baxt, 1992; Bridle and Cox, 1991; Buntine and Weigend, 1992; Hansen and Salamon, 1990; Intrator *et al.*, 1992; Jacobs *et al.*, 1991; Lincoln and Skrzypek, 1990; Neal, 1992a,b; Pearlmutter and Rosenfeld, 1991; Wolpert, 1990; Xu *et al.*, 1992, 1990). Among the key issues are how to design the architecture of the networks; how the results of the various networks should be combined to give the best estimate of the optimal result; and how to make best use of a limited data set. In what follows, we address the issues of optimal combination and efficient data usage in the framework of ensemble averaging.

In this chapter we are concerned with using the information contained in a set of regression estimates of a function to construct a better estimate. The statistical resampling techniques of jackknifing, bootstrapping and cross-validation have proven useful for generating improved regression estimates through bias reduction (Efron, 1982; Miller, 1974; Stone, 1974; Gray and Schucany, 1972; Härdle, 1990; Wahba, 1990, for review). We show that these ideas can be fruitfully extended to neural networks by using the ensemble methods presented. The basic idea behind these resampling techniques is to improve one's estimate of a given statistic θ, by combining multiple estimates

Artificial Neural Networks for Speech and Vision. Edited by Richard J. Mammone. Published in 1993 by Chapman & Hall, London. ISBN 0 412 54850 X

of θ generated by subsampling or resampling of a finite data set. The jackknife method involves removing a single data point from a data set, constructing an estimate of θ with the remaining data, testing the estimate on the removed data point and repeating for every data point in the set. One can then, for example, generate an estimate of θ's variance using the results from the estimate on all of the removed data points. This method has been generalized to include removing subsets of points. The bootstrap method involves generating new data sets from one original data set by sampling randomly with replacement. These new data sets can then be used to generate multiple estimates for θ. In cross-validation, the original data is divided into two sets: one which is used to generate the estimate of θ, and the other which is used to test this estimate. Cross-validation is widely used neural network training to avoid over-fitting. The jackknife and bootstrapping methods are not commonly used in neural network training due to the large computational overhead.

These resampling techniques can be used to generate multiple distinct networks from a single training set. For example, resampling in neural net training frequently takes the form of repeated on-line stochastic gradient descent of randomly initialized nets. However, unlike the combination process in parametric estimation, which usually takes the form of a simple average in parameter space, the parameters in a neural network take the form of neuronal weights which generally have many different local minima. Therefore, we cannot simply average the weights of a population of neural networks and expect to improve network performance. Because of this, one typically generates a large population of resampled nets, chooses the one with the best performance and discards the rest. This process is very inefficient. Here we present ensemble methods which avoid this inefficiency, and avoid the local minima problem by averaging in functional space not parameter space. In addition, we show that the ensemble methods actually benefit from the existence of local minima, and that with the ensemble framework, the statistical resampling techniques have very natural extensions. All of these aspects combined provide a general theoretical framework for network averaging, which in practice generates significant improvement on real-world problems.

10.2 Basic Ensemble Method

In this section we present the Basic Ensemble Method (BEM), which combines a population of regression estimates to estimate a function $f(x)$ defined by $f(x) = E[y|x]$.

Suppose that that we have two finite data sets whose elements are all independent and identically distributed random variables: a training data set $\mathscr{A} = \{(x_m, y_m)\}$ and a cross-validatory data set $\mathscr{CV} = \{(x_l, y_l)\}$. Further, suppose that we have used $\mathscr{A}$ to generate a set of functions, $\mathscr{F} = f_i(x)$, each

element of which approximates $f(x)$.[1] We would like to find the best approximation to $f(x)$ using $\mathscr{F}$.

One common choice is to use the **naive estimator**, $f_{\text{naive}}(x)$, which minimizes the mean square error relative to $f(x)$,[2]

$$\text{MSE}[f_i] = E_{\mathscr{CV}}[(y_m - f_i(x_m))^2],$$

thus

$$f_{\text{naive}}(x) = \arg\min_i \{\text{MSE}[f_i]\}.$$

This choice is unsatisfactory for two reasons: First, in selecting only one network from the population of networks represented by $\mathscr{F}$, we are discarding useful information that is stored in the discarded networks; second, since the $\mathscr{CV}$ data set is random, there is a certain probability that some other network from the population will perform better than the naive estimate on some other previously unseen data set sampled from the same distribution. A more reliable estimate of the performance on previously unseen data is the average of the performances over the population $\mathscr{F}$. Below, we will see how we can avoid both of these problems by using the BEM estimator, $f_{\text{BEM}}(x)$, and thereby generate an improved regression estimate.

Define the **misfit** of function $f_i(x)$, the deviation from the true solution, as $m_i(x) \equiv f(x) - f_i(x)$. The mean square error can now be written in terms of $m_i(x)$ as

$$\text{MSE}[f_i] = E[m_i^2].$$

The average mean square error is therefore

$$\overline{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{i=N} E[m_i^2].$$

Define the BEM regression function $f_{\text{BEM}}(x)$, as

$$f_{\text{BEM}}(x) \equiv \frac{1}{N}\sum_{i=1}^{i=N} f_i(x) = f(x) - \frac{1}{N}\sum_{i=1}^{i=N} m_i(x).$$

If we now assume that the $m_i(x)$ are mutually independent with zero mean,[3]

[1] For our purposes, it does not matter how $\mathscr{F}$ was generated. In practice we will use a set of backpropagation networks trained on the $\mathscr{A}$ data set, but started with different random weight configurations. This replication procedure is standard practice when trying to optimize neural networks.

[2] Here, and in all of what follows, the expected value is taken over the cross-validatory set $\mathscr{CV}$.

[3] We relax these assumption in section 10.4, where we present the Generalized Ensemble Method.

we can calculate the mean square error of $f_{BEM}(x)$ as

$$\begin{aligned}\mathrm{MSE}[f_{\mathrm{BEM}}] &= E\left[\left(\frac{1}{N}\sum_{i=1}^{i=N} m_i\right)^2\right] \\ &= \frac{1}{N^2}E\left[\sum_{i=1}^{i=N} m_i^2\right] + \frac{1}{N^2}E\left[\sum_{i\neq j} m_i m_j\right] \\ &= \frac{1}{N^2}E\left[\sum_{i=1}^{i=N} m_i^2\right] + \frac{1}{N^2}\sum_{i\neq j} E[m_i]E[m_j] \\ &= \frac{1}{N^2}E\left[\sum_{i=1}^{i=N} m_i^2\right], \end{aligned} \tag{1}$$

which implies that

$$\mathrm{MSE}[f_{\mathrm{BEM}}] = \frac{1}{N}\overline{\mathrm{MSE}}. \tag{2}$$

This is a powerful result because it tells us that by averaging regression estimates, we can reduce our mean square error by a factor of N when compared to the population performance. By increasing the population size, we can in principle make the estimation error arbitrarily small! In practice, however, as N gets large our assumptions on the misfits, $m_i(x)$, eventually break down (see section 10.5).

Consider the individual elements of the population $\mathscr{F}$. These estimators will more or less follow the true regression function. If we think of the misfits functions as random noise functions added to the true regression function and these noise functions are uncorrelated with zero mean, then the averaging of the individual estimates is like averaging over the noise. In this sense, the ensemble method is smoothing in functional space and can be thought of as a regularizer with a smoothness assumption on the true regression function.

An additional benefit of the ensemble method's ability to combine multiple regression estimates is that the regression estimates can come from many different sources. This fact allows for great flexibility in the application of the ensemble method. For example, the networks can have different architectures or be trained by different training algorithms or be trained on different data sets. This last option – training on different data sets – has important ramifications. One standard method for avoiding over-fitting during training is to use a cross-validatory hold-out set.[4] The problem is that since the network is never trained on the hold-out data the network may be missing valuable information about the distribution of the data particularly if the total data set is small. This will always be the case for a single network using a cross-

[4] The cross-validatory hold-out set is a subset of the total data available to us and is used to determine when to stop training. The hold-out data is not used to train.

validatory stopping rule. However, this is not a problem for the ensemble estimator. When constructing our population $\mathscr{F}$, we can train each network on the entire training set and let the smoothing property of the ensemble process remove any over-fitting or we can train each network in the population with a different split of training and hold-out data. In this way, the population as a whole will see the entire data set while each network has avoided over-fitting by using a cross-validatory stopping rule. Thus the ensemble estimator will see the entire data set while the naive estimator will not. In general, with this framework we can now easily extend the statistical jackknife, bootstrap and cross-validation techniques (Efron, 1982; Miller, 1974; Stone, 1974) to find better regression functions.

10.3 Intuitive illustrations

In this section, we present two toy examples which illustrate the averaging principle which is at the heart of the ensemble methods presented in this chapter.

For our first example, consider the classification problem depicted in Figure 10.1. Regions A and B represent the training data for two distinct classes which are Gaussianly distributed. If we train a perceptron on this data, we find that hyperplanes, 1, 2, and 3 all give perfect classification performance for the training data; however, only hyperplane 2 will give optimal generalization performance. Thus, if we had to choose a naive estimator from this population of three perceptrons, we would be more likely than not to choose a hyperplane with poor generalization performance. For this problem, it is clear that the BEM estimator (i.e. averaging over the three hyperplanes) is more reliable.

For our second example, suppose that we want to approximate the Gaussian distribution shown in Figure 10.2a and we are given two estimates shown in Figure 10.2b. If we must choose either one or the other of these estimates we will incur a certain mean square error; however, if we average these two functional estimates we can dramatically reduce the mean square error. In Figure10.2c, we represent the ensemble average of the two estimates

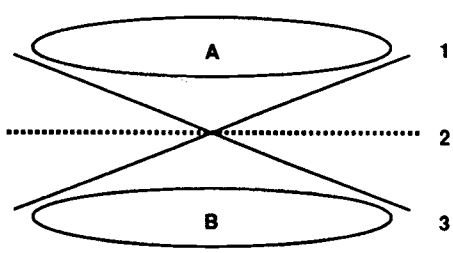

Figure 10.1 *Toy classification problem. Hyperplanes 1 and 3 solve the classification problem for the training data, but hyperplane 2 is the optimal solution. Hyperplane 2 is the average of hyperplanes 1 and 3.*

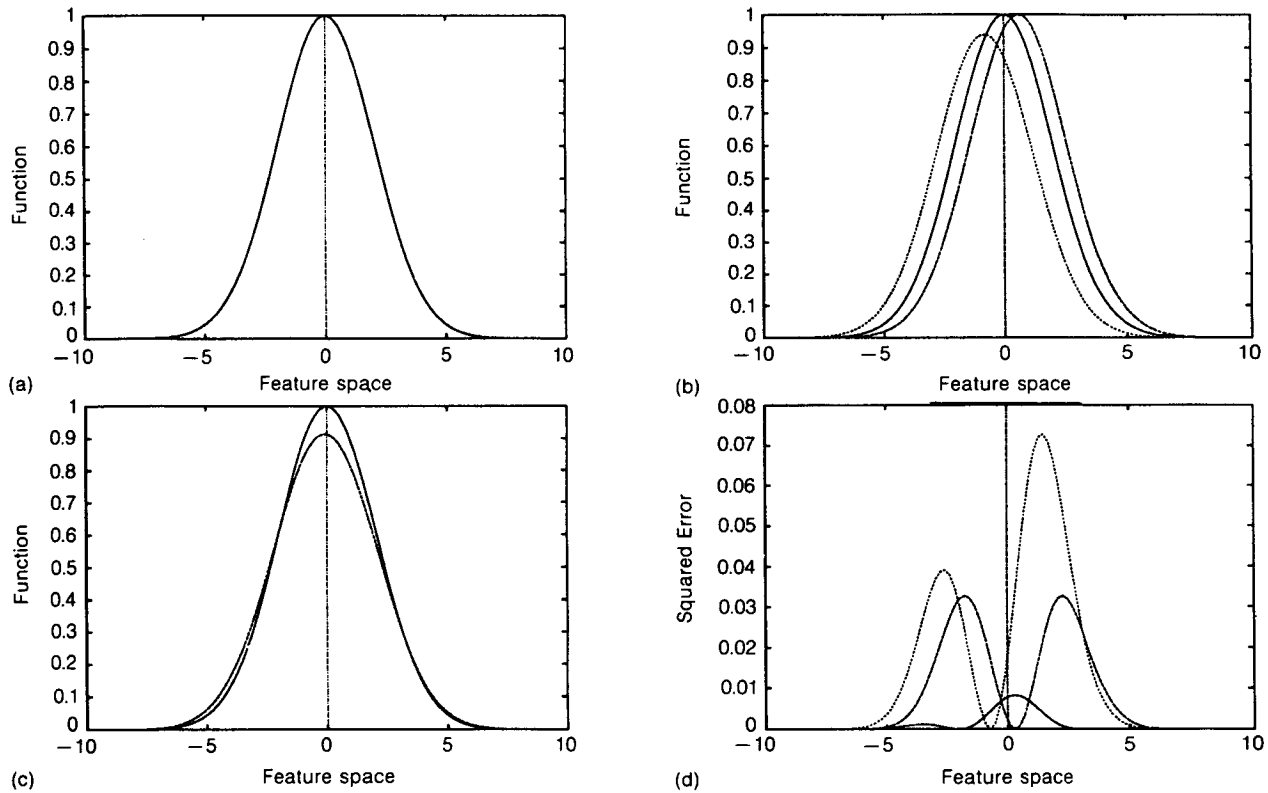

Figure 10.2 *(a) Gaussian to be estimated; (b) two randomly chosen Gaussian estimates compared to the true Gaussian; (c) ensemble average estimate compared to the true Gaussian; (d) square comparison of the three estimates. Notice that the ensemble estimate gives the smallest square error.* —— *: true function;* - - -*: estimate 1;* .... *: estimate 2.*

from Figure 10.2b. Comparing Figure 10.2b to Figure 10.2c, it is clear that the ensemble estimate is much better than either of the individual estimates. In Figure 10.2d, we compare the square error of each of the estimates.

We can push this simple example a little further to demonstrate a weakness of the Basic Ensemble Method and the need for a more general approach. Suppose that $x \sim \mathcal{N}(0, \sigma^2)$ and we are given $\mathcal{D} \equiv \{x_i\}_{i=1}^{i=N}$ and σ. We can estimate the true Gaussian by estimating its mean with

$$\mu \equiv \frac{1}{N} \sum_{j=1}^{j=N} x_j,$$

or we can use a modification of the jackknife method (Gray and Schucany, 1972) to construct a population of estimates from which we can construct an ensemble estimator. Define the ensemble estimate as

$$g_{\text{ensemble}}(x) \equiv \frac{1}{N} \sum_{j=1}^{j=N} g(x; \mu_{(-i)}),$$

where

$$\mu_{(-i)} \equiv \frac{1}{N-1} \sum_{j \neq i} x_j$$

and

$$g(x; \alpha) \equiv \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\alpha)^2/\sigma^2}.$$

We can now explicitly compare these two estimates using the mean integrated square error (MISE) of the estimates,

$$\text{MISE}[g(x; \alpha)] \equiv E_{\mathcal{D}}\left[\int_{-\infty}^{+\infty} (g(x; \alpha) - g(x; 0))^2 \, dx\right].$$

Calculating $\text{MISE}[g(x; \mu)]$ and $\text{MISE}[g_{\text{ensemble}}(x)]$ in this special case, it is easy to show that

$$\text{MISE}[g(x; \mu)] < \text{MISE}[g_{\text{ensemble}}(x)].$$

Comparing this result with equation (2), there seems to be a contradiction: the ensemble average is performing worse on average! The reason the ensemble performs worse on average is that two of the main assumptions from section 10.2 are wrong: The misfit functions from the population are not uncorrelated nor are they in general zero mean. Since these assumptions do not hold in general, we present an alternative formulation of the ensemble method in which these assumptions are not made (see section 10.4).

The above example helps to illustrate two important aspects of neural network regression which should be considered when performing ensemble averages: For neural networks, the existence of multiple local minima prohibits the simple parameter averaging we performed when we approximated

$g(x;\mu)$ in the example above. And in general, we do not know whether the function we are trying to estimate is representable by a given neural network as we assumed was true in the example above.

10.4 Generalized Ensemble Method

In this section we extend the results of section 10.2 to a generalized ensemble technique which always generates a regression estimate which is as low or lower than both the best individual regressor, $f_{\text{naive}}(x)$, and the basic ensemble regressor, $f_{\text{BEM}}(x)$, and which avoids overfitting the data. In fact, it is the best possible of any linear combination of the elements of the population $\mathscr{F}$.

Define the Generalized Ensemble Method estimator, $f_{\text{GEM}}(x)$, as

$$f_{\text{GEM}}(x) \equiv \sum_{i=1}^{i=N} \alpha_i f_i(x) = f(x) + \sum_{i=1}^{i=N} \alpha_i m_i(x),$$

where the α_i's are real and satisfy the constraint that $\sum \alpha_i = 1$. We want to choose the α_i's so as to minimize the MSE with respect to the target function $f(x)$. If again we define $m_i(x) \equiv f(x) - f_i(x)$ and in addition define the symmetric correlation matrix $C_{ij} \equiv E[m_i(x)m_j(x)]$ then we find that we must minimize

$$\text{MSE}[f_{\text{GEM}}] = \sum_{i,j} \alpha_i \alpha_j C_{ij}. \tag{3}$$

We now use the method of Lagrange multipliers to solve for α_k. We want α_k such that $\forall k$

$$\partial_{\alpha_k}\left[\sum_{i,j} \alpha_i \alpha_j C_{ij} - 2\lambda\left(\sum_i \alpha_i - 1\right)\right] = 0.$$

This equation simplifies to the condition that

$$\sum_k \alpha_k C_{kj} = \lambda.$$

If we impose the constraint, $\sum \alpha_i = 1$, we find that

$$\alpha_i = \frac{\sum_j C_{ij}^{-1}}{\sum_k \sum_j C_{kj}^{-1}}. \tag{4}$$

If the $m_i(x)$'s are uncorrelated and zero mean, $C_{ij} = 0 \; \forall i \neq j$ and the optimal α_i's have the simple form

$$\alpha_i = \frac{\sigma_i^{-2}}{\sum_j \sigma_j^{-2}},$$

where $\sigma_i^2 \equiv C_{ii}$, which corresponds to the intuitive choice of weighting the f_i's by the inverse of their respective variances and normalizing. Combining

equations (3) and (4), we find that the optimal MSE is given by

$$\mathrm{MSE}[f_{\mathrm{GEM}}] = \left[\sum_{ij} C_{ij}^{-1}\right]^{-1}. \tag{5}$$

The results in this section depend on two assumptions: the rows and columns of C are linearly independent and we have a reliable estimate of C. In certain cases where we have nearly duplicate networks in the population $\mathscr{F}$, we will have nearly linearly dependent rows and columns in C, which will make the inversion process very unstable and our estimate of C^{-1} will be unreliable. In these cases, we can use heuristic techniques to sub-sample the population $\mathscr{F}$ to assure that C has full rank (see section 10.6). In practice, the increased stability produced by removing near degeneracies outweighs any information lost by discarding nets. Since the C we calculate is the sample correlation matrix not the true correlation matrix, C is a random variable as are $\mathrm{MSE}[f_{\mathrm{GEM}}]$ and the optimal α_i's. Thus noise in the estimate of C can lead to bad estimates of the optimal α_i's. If needed, we can get a less biased estimate of C^{-1} by using a jackknife procedure (Gray and Schucany, 1972) on the data used to generate C.

Note also that the BEM estimator and the naive estimator are both special cases of the GEM estimator, and therefore $\mathrm{MSE}[f_{\mathrm{GEM}}]$ will always be less than or equal to $\mathrm{MSE}[f_{\mathrm{BEM}}]$ and $\mathrm{MSE}[f_{\mathrm{naive}}]$. An explicit demonstration of this fact can be seen by comparing the respective MSE's under the assumption that the $m_i(x)$'s are uncorrelated and zero mean. In that case, comparing equations (1) and (5), we have

$$\mathrm{MSE}[f_{\mathrm{BEM}}] = \frac{1}{N^2}\sum_i \sigma_i^2 \geqslant \left[\sum_i \sigma_i^{-2}\right]^{-1} = \mathrm{MSE}[f_{\mathrm{GEM}}],$$

with equality only when all of the σ_i are identical. This relation is easily proven using the fact that $a/b + b/a \geqslant 2\ \forall a, b > 0$. Similarly, we can write

$$\mathrm{MSE}[f_{\mathrm{naive}}] = \sigma_{\min}^2 \geqslant \left[\sum_i \sigma_i^{-2}\right]^{-1} = \mathrm{MSE}[f_{\mathrm{GEM}}].$$

Thus we see that the GEM estimator provides the best estimate of $f(x)$ in the mean square error sense.

10.5 Experimental results

In this section, we report on an application of the Generalized Ensemble Method to the NIST OCR database. The characters were hand-segmented, hand-labeled and preprocessed into 120 dimensional feature vectors by convolution with simple kernels. The database was divided into three types (numbers, uppercase characters and lowercase characters) and each of these types was divided into independent training, testing and cross-validatory sets with sizes listed in Table 10.1.

Table 10.1 *Database divisions*

Data set	Training set	CV set	Testing set	Classes
Numbers	13241	13241	4767	10
Uppercase	11912	11912	7078	26
Lowercase	12971	12970	6835	26

We trained a population of 10 single hidden unit layer backpropagation networks for a variety of different hidden unit layer sizes for each type of data. Each network was initialized with a different random configuration of weights. Training was stopped using a cross-validatory stopping criterion. For simplicity, we calculated the weights for the GEM estimator under the assumption that the misfits were uncorrelated and zero mean.

Straight classification results are shown in Figures 10.3, 10.5 and 10.7. In these plots, the classification performance of the GEM estimator (labeled 'Ensemble'), the naive estimator (labeled 'Best Individual') and the average estimator from the population (labeled 'Individual') are plotted versus the number of hidden units in each individual network. Error bars are included for the average estimator from the population. In all of these plots there is an increase in performance as the number of hidden units increases. Notice, however, that in all of these results the ensemble estimator was not only better than the population average but it was also as good as or better than the naive estimator in every case.

Typically for character recoginition problems, it is worse for the network to make an error than it is for the network to reject a pattern. This weighted cost

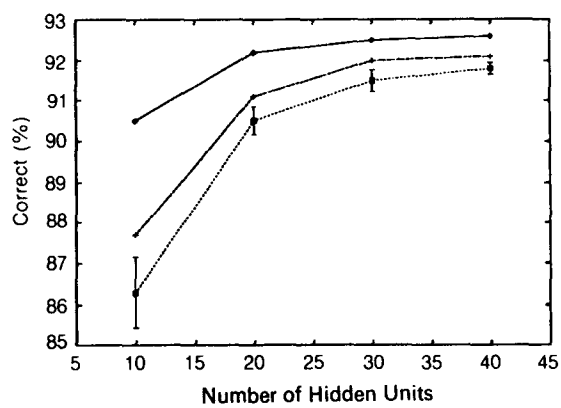

Figure 10.3 *Upper case percentages.* ◆*: ensemble;* +*: best individual;* ■*: individual.*

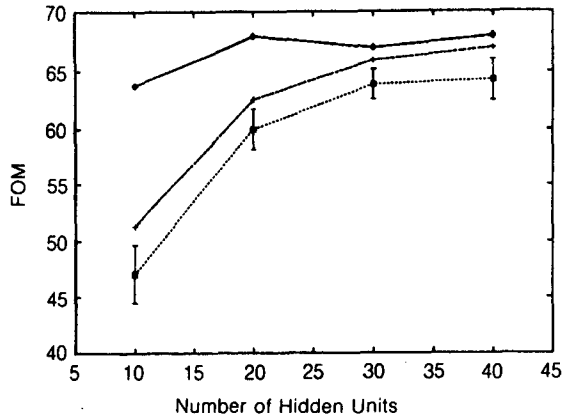

Figure 10.4 *Upper case FOM.* ◆*: ensemble;* +*: best individual;* ■*: individual.*

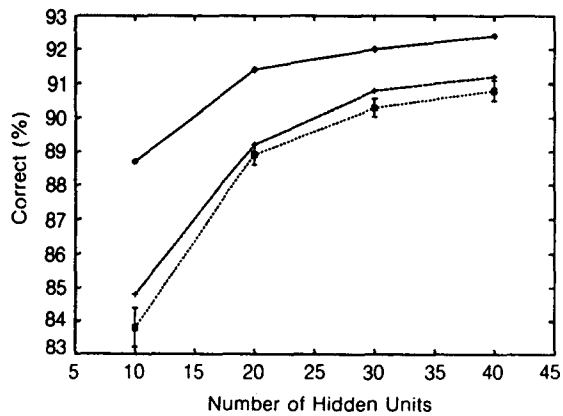

Figure 10.5 *Lower case percentage.* ◆*: ensemble;* +*: best individual;* ■*: individual.*

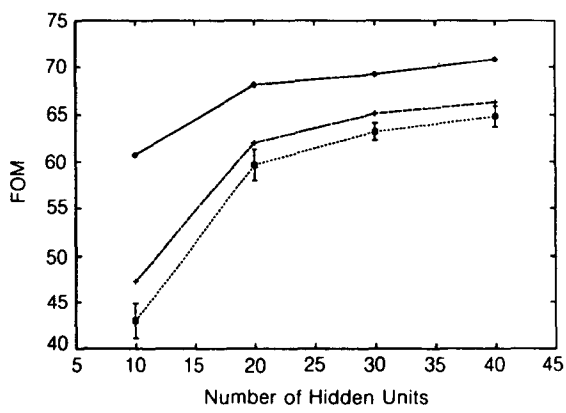

Figure 10.6 *Lower case FOM.* ◆*: ensemble;* +*: best individual;* ■*: individual.*

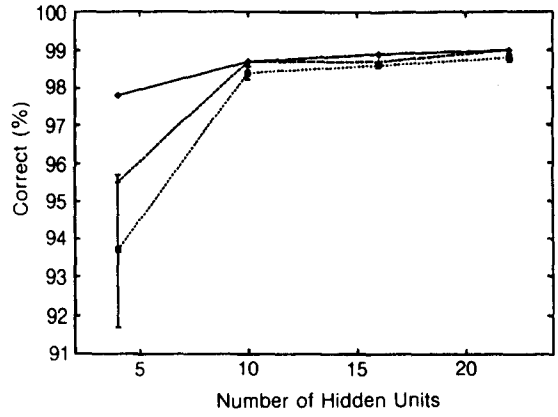

Figure 10.7 *Numbers percentage.* ◆*: ensemble;* +*: best individual;* ■*: individual.*

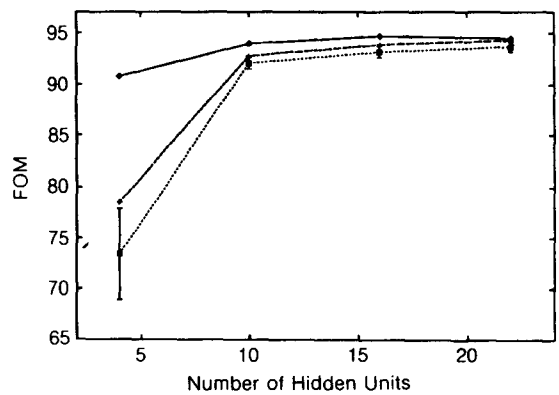

Figure 10.8 *Numbers FOM.* ◆*: ensemble;* +*: best individual;* ■*: individual.*

can be taken into account by calculating a Figure of Merit (FOM) instead of a straight performance measure. We define our FOM as follows:

$$\text{FOM} \equiv \%\text{Correct} - \%\text{Rejected} - 10(\%\text{Error}).$$

In our simulations we found an optimal rejection threshold for each network based on the cross-validatory set. FOM results are shown in Figures 10.4, 10.6 and 10.8. Again, notice that in all of these results, just as in the straight classification results, the ensemble estimator was not only better than the population average but it was also better than the naive estimator.

These results for a difficult, real-world problem show that the GEM estimator is significantly and dramatically better than standard techniques.

It is important to consider how many networks are necessary for the ensemble methods presented in this paper to be useful. If we take the BEM

result seriously (equation 2), we should expect that increasing the number of networks in the population can only improve the BEM estimator. However as stated in section 10.2, eventually our assumptions on the misfits breakdown and equation (2) is no longer valid. This fact is clearly demonstrated in Figure 10.9, where we show the FOM performance saturate as the number of nets in the population increases. In the figure, we see that saturation in this example occurs after only six or eight nets are in the ensemble population. This is a very interesting result because it gives us a measure of how many distinct[5] nets are in our population. This knowledge is very useful when sub-sampling a given population. This result also suggests a very important observation: Although the number of local minima in parameter space is extremely large, the number of distinct local minima in functional space is actually quite small!

We can make another important observation if we compare Figure 10.9 with Figure 10.4. Consider the value of the FOM on the test data for an ensemble of 4 networks (Figure 10.9). Compare this value to population average FOM for nets with 40 hidden units (Figure 10.4). These values are not

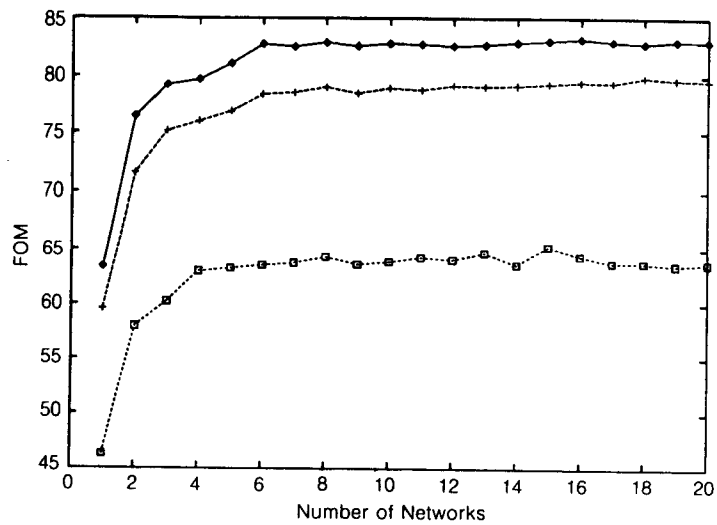

Figure 10.9 *Ensemble FOM versus the number of nets in the ensemble. Ensemble FOM graphs for the upper case training, cross-validatory and testing data are shown. Each net in the populations had 10 hidden units. The graphs are for a single randomly chosen ordering of 20 previously trained nets. No effort was made to optimally choose the order in which the nets were added to the ensemble. Improved ordering gives improved results.* ◆*: train FOM;* +*: CV FOM;* □*: test FOM.*

[5] By 'distinct', we mean that the misfits of two nets are weakly correlated. It is, of course, arguable what should be considered weakly correlated. For the purposes of this chapter, networks are distinct if the related correlation matrix, C, has a robust inverse.

significantly different; however, training a population of large nets to find the best estimator is computationally much more expensive than training and averaging a population of small nets. In addition, small networks are more desirable since they are less prone to over-fitting than large networks.[6]

It is also interesting to note that there is a striking improvement for an ensemble size of only two.

10.6 Improving BEM and GEM

One simple extension of the ensemble methods presented in this chapter is to consider the BEM and GEM estimators of all of the possible populations which are subsets of the original network population $\mathscr{F}$.[7] All the information we need to perform subset selection is contained in the correlation matrix, C, which only has to be calculated once.

In principal, the GEM estimator for $\mathscr{F}$ will be a better estimator than the GEM estimator for any subset of $\mathscr{F}$, however, in practice, we must be careful to assure that the correlation matrix is not ill-conditioned. If, for example, two networks in the population are very similar, two of the rows of C will be nearly collinear. This collinearity will make inverting the correlation matrix very error prone and will lead to very poor results. Thus in the case of the GEM estimator it is important to remove all duplicate (or nearly duplicate) networks from the population $\mathscr{F}$. Removing duplicates can be easily done by examining the correlation matrix. One can remove all networks for which the dot product of its row in the correlation matrix with any other row in the correlation matrix is above some threshold. This threshold can be chosen to allow a number of nets equal to the number of distinct networks in the population, as described in section 10.5.

An alternative approach (Wolpert, 1990), which avoids the potential singularities in C, is to allow a perceptron to learn the appropriate averaging weights. Of course, this approach will be prone to local minima and noise due to stochastic gradient descent, just as the original population $\mathscr{F}$ was; thus we can train a population of perceptrons to combine the networks from $\mathscr{F}$ and then average over this new population. A further extension is to use a nonlinear network (Jacobs *et al.*, 1991; Reilly *et al.*, 1987; Wolpert, 1990) to learn how to combine the networks with weights that vary over the feature space, and then to average an ensemble of such networks. This extension is reasonable since networks will, in general perform better in certain regions of the feature space than in others.

[6] Of course we cannot make the individual nets too small or they will not have sufficient complexity.

[7] This approach is essentially the naive estimator for the population of BEM and GEM estimators. Averaging over the population of BEM or GEM estimators will not improve performance.

In the case of the BEM estimator, we know that as the population size grows our assumptions on the misfits, $m_i(x)$, are not longer valid. When our assumptions break down, adding more nets to the population is a waste of resources, since it will not improve the performance, and if the nets we add have particularly poor performance, we can actually lower the performance of the BEM estimator. Thus, it would be ideal if we could find the optimal subset of the population $\mathcal{F}$ over which to average. We could try all the $2^N - 1$ possible non-empty subsets of $\mathcal{F}$, but for large N this search becomes unmanageable. Instead, we can order the elements of the population according to increasing mean square error[8], and generate a set of N BEM estimates by adding successively the ordered elements of $\mathcal{F}$. We can then choose the best estimate. The BEM estimator is then guaranteed to be at least as good as the naive estimator.

We can further refine this process by considering the difference between the mean square error for the BEM estimator for a population of N elements and the mean square error for the BEM estimator for the same population plus a new net. From this comparison, we find that we should add the new net to the population if the following inequality is satisfied:

$$(2N+1)\mathrm{MSE}[\hat{f}_N] > 2\sum_{i\neq \mathrm{new}} E[m_{\mathrm{new}}m_i] + \mathrm{E}[m_{\mathrm{new}}^2],$$

where $\mathrm{MSE}[\hat{f}_N]$ is the mean square error for the BEM estimator for the population of N and m_{new} is the misfit for the new function to be added to the population. The information to make this decision is readily available from the correlation matrix C. Now, if a network does not satisfy this criterion, we can swap it with the next untested network in the ordered sequence.

10.7 Conclusions

We have developed a general mathematical framework for improving regression estimates. In particular, we have shown that by averaging in functional space, we can construct neural networks which are guaranteed to have improved performance.

An important strength of the ensemble method is that it does not depend on the algorithm used to generate the set of regressors, and therefore can be used with any set of networks. This observation implies that we are not constrained in our choice of networks, and can use nets of arbitrary complexity and architecture. Thus the ensemble methods described in this chapter are completely general in that they are applicable to a wide class of problems, including neural networks and any other technique which attempts to minimize the mean square error.

[8] The first element in this sequence will be the naive estimator.

One striking aspect of network averaging is the manner in which it deals with local minima. Most neural network algorithms achieve sub-optimal peformance specifically due to the existence of an overwhelming number of sub-optimal local minima. If we take a set of neural networks which have converged to local minima and apply averaging we can construct an improved estimate. One way to understand this fact is to consider that, in general, networks which have fallen into different local minima will perform poorly in different regions of feature space, and thus their error terms will not be strongly correlated. It is this lack of correlation which drives the averaging method. Thus, the averaging method has the remarkable property that it can efficiently utilize the local minima that other techniques try to avoid.

It should also be noted that since the ensemble methods are performing averaging in functional space, they have the desirable property of inherently performing smoothing in functional space. This property will help avoid any potential over-fitting during training.

In addition, since the ensemble method relies on multiple functionally independent networks, it is ideally suited for parallel computation during both training and testing.

We are working to generalize this method to take into account confidence measures and various nonlinear combinations of estimators in a population.

Acknowledgements

We would like to thank the members of the Brown University Institute for Brain and Neural Systems, in particular Nathan Intrator for many useful discussions. We would also like to thank Nestor Inc. for making the NIST OCR database available to us.

Research was supported by the Office of Naval Research, the Army Research Office, and the National Science Foundation.

References

Baxt, W.G. (1992) Improving the accuracy of an artifical neural network using multiple differently trained networks. *Neural Computation*, **4**(5).

Bridle, J.S. and Cox, S.J. (1991) RecNorm: simultaneous normalization and classification applied to speech recognition, in *Advances in Neural Information Processing Systems 3*.

Buntine, W.L. and Weigend, A.S. (1992) Bayesian back-propagation. *Complex Systems*, **5**, 603–43.

Cooper, L.N. (1991) Hybrid neural network architectures: Equilibrium systems that pay attention, in *Neural Networks: Theory and Applications* (eds R.J. Mammone and Y. Zeevi), Academic Press, New York, pp. 81–96.

Efron, B. (1982) *The Jackknife, the Bootstrap and Other Resampling Plans*. SIAM, Philadelphia, PA.

Gray, H.L. and Schucany, W.R. (1972) *The Generalized Jackknife Statistic*. Dekker, New York, NY.

Hansen, L.K. and Salamon, P. (1990) Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**(10), 993–1000.

Härdle, W. (1990) *Applied Nonparametric Regression.* University of Cambridge Press, New York, NY.

Intrator, N., Reisfeld, D. and Yeshurun, Y. (1992) Face recognition using a hybrid supervised/unsupervised neural network. Preprint.

Jacobs, R.A., Jordan, M.I., Nowlan, S.J. and Hinton, G.E. (1991) Adaptive mixtures of local experts. *Neural Computation*, **3**(2).

Lincoln, W.P. and Skrzypek, J. (1990) Synergy of clustering multiple back propagation networks, in *Advances in Neural Information Processing Systems 2.*

Miller, R.G. (1974), The jackknife – a review. *Biometrika*, **61**(1), 1–16.

Neal, R.M. (1992a) Bayesian learning via stochastic dynamics, in *Advances in Neural Information Processing Systems*, (eds. J.E. Moody, S.J. Hanson and R.P. Lippmann) Morgan Kaufmann, San Mateo, CA.

Neal, R.M. (1992b) Bayesian mixture modeling by Monte Carlo simulation. *Technical report crg-tr-91-2*, University of Toronto.

Pearlmutter, B.A. and Rosenfeld, R. (1991) Chaitin-kolmogorov complexity and generalization in neural networks, in *Advances in Neural Information Processing Systems 3.*

Reilly, D.L., Scofield, C.L., Cooper, L.N. and Elbaum, C. (1988) Gensep: A multiple neural network learning system with modifiable network topology, in *Abstracts of the First Annual International Neural Network Society Meeting.*

Reilly, R.L., Scofield, C.L., Elbaum, C., and Cooper, L.N. (1987) Learning system architectures composed of multiple learning modules. *Proc. IEEE First Int. Conf. on Neural Networks*, Volume 2.

Scofield, C., Kenton, L. and Chang, J. (1991) Multiple neural net architectures for character recognition. *Proc. Compcon, San Francisco, CA*, 487–91.

Stone, M. (1974) Cross-validatory choice and assessment of statistical predictions (with discussion). *J. Roy. Stat. Soc. Ser. B*, **36**, 111–47.

Wahba, G. (1990) *Spline Models for Observational Data.* SIAM, Philadelphia. PN.

Wolpert, D.H. (1990) Stacked generalization. *Technical report LA-UR-90-3460*, Complex Systems Group, Los Alamos, NM.

Xu, L., Krzyzak, A. and Suen, C.Y. (1990) Associative switch for combining classifiers. *Technical report x9011*, Department of Computer Science, Concordia University, Montreal, Canada.

Xu, L., Krzyzak, A. and Suen, C.Y. (1992) Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, **22**(3).