What Size Neural Network Gives Optimal Generalization? Convergence Properties of Backpropagation

Steve Lawrence[1,2]*, C. Lee Giles[1]†‡, Ah Chung Tsoi[2]
{lawrence,act}@elec.uq.edu.au, giles@research.nj.nec.com

[1] NEC Research Institute, 4 Independence Way, Princeton, NJ 08540

[2] Department of Electrical and Computer Engineering
University of Queensland, St. Lucia 4072, Australia

Technical Report
UMIACS-TR-96-22 and CS-TR-3617
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742

June 1996 (Revised August 1996)

Abstract

One of the most important aspects of any machine learning paradigm is how it scales according to problem size and complexity. Using a task with known optimal training error, and a pre-specified maximum number of training updates, we investigate the convergence of the backpropagation algorithm with respect to a) the complexity of the required function approximation, b) the size of the network in relation to the size required for an optimal solution, and c) the degree of noise in the training data. In general, for a) the solution found is worse when the function to be approximated is more complex, for b) oversized networks can result in lower training and generalization error in certain cases, and for c) the use of committee or ensemble techniques can be more beneficial as the level of noise in the training data is increased. For the experiments we performed, we do not obtain the optimal solution in any case. We further support the observation that larger networks can produce better training and generalization error using a face recognition example where a network with many more parameters than training points generalizes better than smaller networks.

Keywords: Local Minima, Generalization, Committees, Ensembles, Convergence, Backpropagation, Smoothness, Network Size, Problem Complexity, Function Approximation, Curse of Dimensionality.

* http://www.neci.nj.nec.com/homepages/lawrence
† Also with the Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742.
‡ http://www.neci.nj.nec.com/homepages/giles.html

1 Introduction

Statements regarding the training and generalization error of MLPs similar to the following occur often in the neural network literature and community:

1. *"BP is actually a gradient method, and therefore, there is no guarantee at all that the absolute minimum can be reached. In spite of this theoretical remark, researchers involved with BP applications know that this is not a very serious problem. BP often leads to a global minimum, or at least makes it possible to meet practical stopping criteria."*
2. *"We have found local minima to be very rare and that the system learns in a reasonable period of time."*
3. *"Backpropagation works well by avoiding non-optimal solutions."*
4. *"We should not use a network with more parameters than the number of data points available."*

Statements 1 to 3 say that while local minima are expected, they nevertheless either do not affect the quality of the solution greatly, or they occur so infrequently that the effect can be ignored in practice (Breiman (1994) makes the following comment about local minima: *"Almost none of the neural net people seem to worry about landing in local minima"*). Statement 4 expresses the intuition that the degrees of freedom in the model should be less than the total number of data points available for training.

In this paper, we show that a solution near the optimal solution is often not obtained. The relative quality of the solution obtained will be investigated as a function of the following variables: a) the complexity of the required function approximation, b) the size of the network in relation to the size required for an optimal solution, and c) the degree of noise present in the data. The results indicate that a) the solution found is worse when the function to be approximated is more complex, b) oversized networks can result in lower training and generalization error in certain cases, and c) the use of committee or ensemble techniques can be more beneficial as the amount of noise in the training data is increased. Further support for the observation that larger networks can, in certain cases, produce better training and generalization error is provided with a face recognition example where a network with 364 times more parameters than training points generalizes better than smaller networks. Techniques to control generalization are not used in order to illustrate this case.

2 Local Minima

It has been shown that the error surface of a backpropagation network with one hidden layer and $t-1$ hidden units has no local minima, if the network is trained with an arbitrary set containing t different inputs[1] (Yu, 1992).

In practice, however, other features of the error surface such as "ravines" and "plateaus" (Baldi and Hornik, 1988) can present difficulty for optimisation. For example, the two error functions shown in figure 1 (from (Gori, 1996)) do not have local minima. However, the function on the left is expected to be more difficult to optimise with gradient descent. For the purposes of this paper, the criterion of interest considered is "the best solution found in a given practical time limit."

[1]For large t, it may be impractical to use a network large enough in order to ensure that there are no local minima.

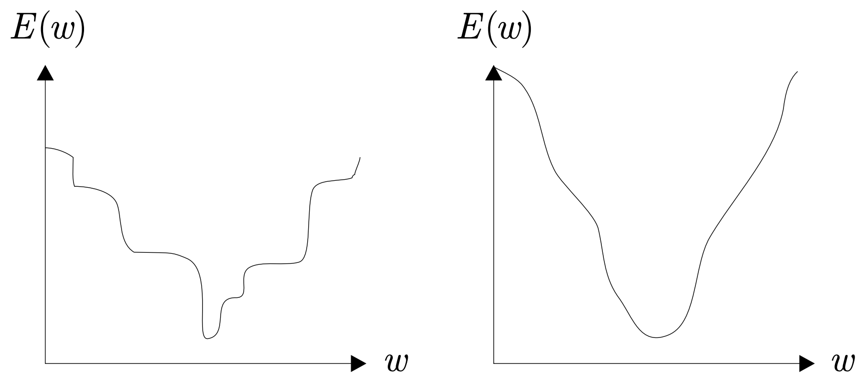

Figure 1. Examples of two possible error functions of one dimension (from (Gori, 1996)). The abscissa corresponds to the value of the single parameter and the ordinate corresponds to the error function. Although neither of these functions contains local minima, the function on the left is expected to be less suitable for gradient descent optimisation due to the "flat" regions.

3 Prior Work

The error surface of very small networks has been characterized previously, e.g. for an XOR network (Hamey, 1995). However, practical networks often contain hundreds or thousands of weights[2] and, in general, theoretical and empirical results on small networks do not scale up to large networks. One reason may be attributed to the interference effect in the training process. Consider the backpropagation training algorithm, if the hidden layer neurons are not all in saturation, the gradients evaluated at the hidden layer neurons are coupled (the update of each parameter generally affects many other parameters). For a network with more hidden layer neurons, this interference effect is expected to be more pronounced.

Caruana presented a tutorial at NIPS 93 (Caruana, 1993) with generalization results on a variety of problems as the size of the networks was varied from "too small" to "too large". "Too small" and "too large" are related to the number of parameters in the model (without consideration of the distribution of the data, the error surface, etc.). Caruana reported that large networks *rarely do worse* than small networks on the problems he investigated. The results in this paper partially correlate with that observation. Caruana suggested that "backprop ignores excess parameters".

Crane, Fefferman, Markel and Pearson (1995) used real-valued data generated by a random target network, and attempted training new networks on the data in order to approximate the number of minima on the error surface under varying conditions. The use of random target networks in this fashion has been referred to as the student teacher problem (Saad and Solla, 1995). Motivated by this work, a very similar technique is used in this paper in order to evaluate the quality of the local minima which are found using backpropagation as a function of various parameters.

Saad and Solla (1996) used the student teacher problem to analyze the effect of noise on on-line learning. For

[2]Networks with up to 1.5 million weights have been used for speech phoneme recognition (Bourlard and Morgan, 1994).

the case of training examples corrupted with additive output noise (they also analyze model noise), Saad and Solla have shown that small noise levels may shorten the symmetric phase of learning while larger values may lengthen the phase. Generalization error increases as the noise level is increased. For the asymptotic case, training with a fixed learning rate results in a non-vanishing asymptotic generalization error. They show that learning rate decay schemes can remove the effects of additive output noise asymptotically.

Müller, Finke, Schulten, Murata and Amari (1996) also used randomly generated "teacher" networks in order to create training examples for "student" networks. They perform a detailed study of generalization as a function of the number of training samples for classification tasks[3]. For networks with up to 256 weights, they demonstrate strong overfitting for a small number of training examples, a region where the generalization error scales according to $1/N_{tr}^2$ where N_{tr} is the number of training examples, and asymptotic scaling according to $N_w/2N_{tr}$ where N_w is the number of weights in the network.

There are several theories for determining the optimal network size e.g. the NIC (Network Information Criterion) (Amari, 1995) which is a generalization of the AIC (Akaike Information Criterion) (Akaike, 1973; Akaike, 1974) widely used in statistical inference, the generalized final prediction error[4] (GPE) as proposed by Moody (1992), and the Vapnik-Chervonenkis (VC) dimension (Maass, 1995; Abu-Mostafa, 1989; Bartlett, 1993) – which is a measure of the expressive power of a network[5]. NIC relies on a single well-defined minimum to the fitting function and can be unreliable when there are several local minima (Ripley, 1995). There is very little published computational experience of the NIC, or the GPE. Their evaluation is prohibitively expensive for large networks.

VC bounds have been calculated for various network types (Cohn and Tesauro, 1992). Early VC-dimension work handles only the case of discrete outputs. For the case of real valued outputs, a more general notion of a "dimension" is required. Such a "pseudo-dimension" can be defined by considering a loss function which measures the deviation of predictions from the target values (Maass, 1995). VC bounds are likely to be too conservative because they provide generalization guarantees simultaneously for any probability distribution and any training algorithm. The computation of VC bounds for practical networks is difficult. Apart from small examples, we are unaware of any systematic procedures for the evaluation of VC bounds for typical practical networks.

Other work addressing local minima or the number of samples required with respect to generalization include (Baum and Haussler, 1989; Sartori and Antsaklis, 1991; McInerny, Haines, Biafore and Hecht-Nielsen, 1989; Yu, 1992; Gori and Tesi, 1992). These approaches are limited due to the assumptions they make, e.g. typical limitations include applicability only to linearly separable problems, consideration only of true local minima as opposed to regions where gradient descent becomes "stuck" (such as "plateaus"), and no consideration of limits on training time.

[3]With respect to the results reported here, overfitting behaviour for classification tasks is expected to be different due to the use of training patterns with asymptotic targets.

[4]The final prediction error (FPE) is an alternative method for determining the order of a dynamical process, originally proposed by Akaike (1970), and generalized to the neural network setting by Moody (1992).

[5]Very briefly, this is the largest set of examples that can be shattered by the network, where a set of x examples is "shattered" by the network if for each of the 2^x possible ways of dividing the x samples into disjoint sets S_1 and S_2, there exists a function computable by the network such that the output is 1 for members of S_1 and the output is 0 for members of S_2 (for a binary classification problem).

4 Artificial Task

To investigate empirical performance we have chosen an artificial task so that we a) know the optimal solution, and b) can carefully control various parameters. The task is as follows and is very similar to the procedure used in (Crane et al., 1995):

1. An MLP with m_i input nodes, m_h hidden nodes, and m_o output nodes (denoted by $m_i : m_h : m_o$ and later referred to as the "data generating network") is initialized with random weights, uniformly selected within a specified range, i.e., w_i in the range $-K$ to K, where w_i are the weights of the network except the biases, and K is a constant. K is 1.0 for the results reported in this paper except when otherwise specified. The bias weights are initialized to small random values in the range $(-0.1, 0.1)$. As K is increased, the "complexity" of the function mapping is increased as will be discussed in more detail in section 6.2.
2. N_{tr} data points are created by selecting random inputs with zero mean and unit variance and propagating them through the network to find the corresponding outputs. This dataset $\mathcal{S}$ forms the training data for subsequent simulations. The procedure is repeated to create a test dataset with N_{te} points. N_{te} is 5000 for all simulations reported in this paper. The choice of zero mean and unit variance inputs is not too unrealistic because the inputs to an MLP are often normalised to have zero mean and unit variance (the distribution may not be normal however) (Le Cun, 1993).
3. The training data set $\mathcal{S}$ is used to train new MLPs, known subsequently as the "trained networks" with the following architecture: $m_i : m'_h : m_o$. For certain tests, m'_h is varied from m_h to M, where $M >> m_h$. The initial weights of these new networks are set using the procedure suggested in Haykin (1994) (i.e. they are not equal to the weights in the network used to create the dataset). They are initialized on a node by node basis as uniformly distributed random numbers in the range $(-2.4/F_i, 2.4/F_i)$ where F_i is the fan-in of neuron i. Theoretically, if $m'_h \geq m_h$, then the optimal training set error is zero (for the case where no noise is added to the data).

Figure 2 shows the process graphically.

5 Methodology for Exploring Convergence

The artificial task will be used to explore the convergence of the networks while varying certain parameters in a controlled manner. Both the training and the generalization performance will be investigated. The baseline network topology is 20:10:1, where 20, 10, and 1 were chosen to represent a typical network where the number of inputs is greater than the number of hidden nodes and the specific values were chosen such that the total training time of the simulations was reasonable. The following methodology is used:

1. The following parameters of the simulations are varied one at a time: a) the maximum value used for setting the weights in the generating network ($1 \leq K \leq 10$), b) the size of the trained networks

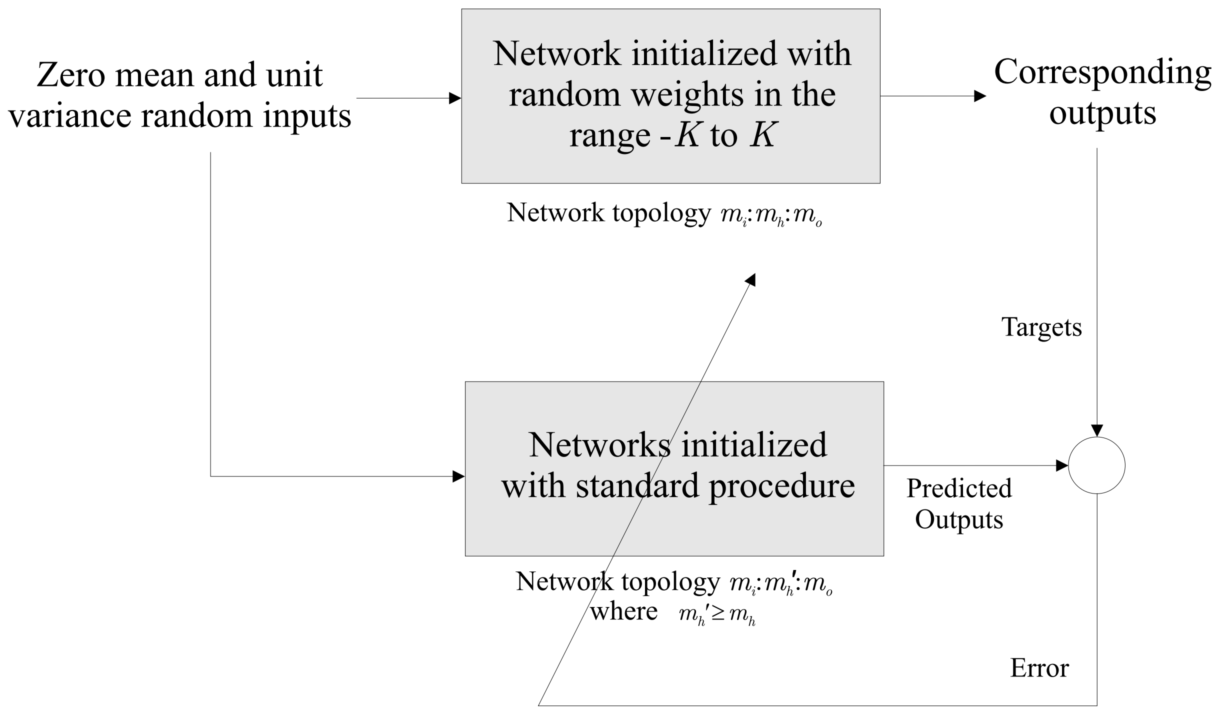
Zero mean and unit variance random inputs
Network initialized with random weights in the range -K to K
Corresponding outputs
Network topology m_i:m_h:m_o
Targets
Networks initialized with standard procedure
Predicted Outputs
Network topology m_i:m_h':m_o where $m_h' \geq m_h$
Error

Figure 2. The process of creating the data sets.

($10 \leq m_h' \leq 50$, c) the size of the training dataset ($200 \leq N_{tr} \leq 20,000$, and d) the amount of zero mean Gaussian noise added to the training data (from zero to a standard deviation of 2% of the standard deviation of the input data).

2. Each configuration of the MLP is tested with ten simulations, each with a different starting condition (random weights).

3. Stopping criterion. No stopping criterion, and no method of controlling generalization [6] is used (other than a maximum number of updates) in order to demonstrate this case. All networks are trained for an identical number of stochastic updates (5×10^5). It is expected that overfitting could occur.

We used the standard MLP: $y_k^l = f\left(\sum_{i=0}^{N_{l-1}} w_{ki}^l y_i^{l-1}\right)$ where y_k^l is the output of neuron k in layer l, N_l is the number of neurons in layer l, w_{ki}^l is the weight connecting neuron k in layer l to neuron i in layer $l-1$, $y_0^l = 1$ (bias), and f is the hyperbolic tangent function. The number of weights in each network is thus $(m_i+1)m_h + (m_h+1)m_o$.

Standard backpropagation was used with stochastic update (update after each training point). Batch update was also investigated – convergence was found to be very poor even when training times were extended by an order of magnitude. The quadratic cost function was used: $E = \frac{1}{2}\sum_{i=1}^{N} e_i^2 = \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{m_o}(d_{ki} - y_{ki})^2$, where d_{ki} is the desired value of the k th output neuron for the i th training sample from the training data set $\mathcal{S}$, and y_{ki} is the value of k th output neuron of the MLP, in response to the i th training sample. The learning rate was 0.05.

[6] There are many ways of controlling generalization, e.g. a) early stopping, b) weight decay or weight elimination, and c) pruning – e.g. OBD (optimal brain damage) (Le Cun, Denker and Solla, 1990) and OBS (optimal brain surgeon) (Hassibi and Stork, 1993).

6 Simulation Results

Results for varying the network size, the training set size, the function complexity, and the amount of noise added to the training data are presented in the following sections.

6.1 Network Size

This section investigates the training and generalization behavior of the networks with the generating network size fixed but the trained network size increasing. For all cases, the data was created with a generating network architecture $20 : 10 : 1$, and the random weight maximum value, K, was 1. The trained networks had the following architecture: $20 : m'_h : 1$, where m'_h was varied from 10 to 50. Theoretically, the optimal training set error for all networks tested is zero, as $m'_h \geq m_h$. However, none of the networks trained here obtained the optimal error (using backpropagation for 5×10^5 updates)[7].

Considering that networks with more than 10 hidden units contain more degrees of freedom than is necessary for zero error, a reasonable expectation would be for the performance to be worse, on average, as the number of hidden units is increased. Figure 3 shows the training and test set error as the number of hidden units in the trained network is varied from 10 to 50. The number of training points, N_{tr}, is 2000. On average, a better solution is found in the larger networks when compared with the 10 hidden units networks. The best mean training and generalisation error occurs for networks with 40 hidden units. This trend varies according to the generating network size (number of inputs, hidden nodes and outputs), the nature of the target function, etc. For example, the optimal size networks perform best for certain tasks, and in other cases the advantage of larger networks can be even greater.

Figure 4 shows the results for the case of 20,000 and the case of 200 training points. Similar results are obtained for 20,000 training points, i.e. on average, a better solution is found in the larger networks when compared with the 10 hidden units networks. The best mean training and generalisation error also occurs for networks with 40 hidden units in this case.

For 200 data points, the best mean training error occurs at 50 hidden units and the best mean generalisation error occurs at 30 hidden units. However, in this case the generalisation error is quite poor for all networks (the number of data points is probably too small to accurately characterise the target function, cf. the curse of dimensionality). The number of parameters in the networks is greater than 200, even for the case of 10 hidden units, as shown in table 1. This leads to the question: would networks smaller than the generating network generalise better? In this case the answer was no – networks with 5 to 9 hidden units resulted in worse performance.

[7]Alternative optimization techniques (e.g. conjugate gradient) can improve convergence in many cases. However, these techniques often lose their advantage with larger problems.

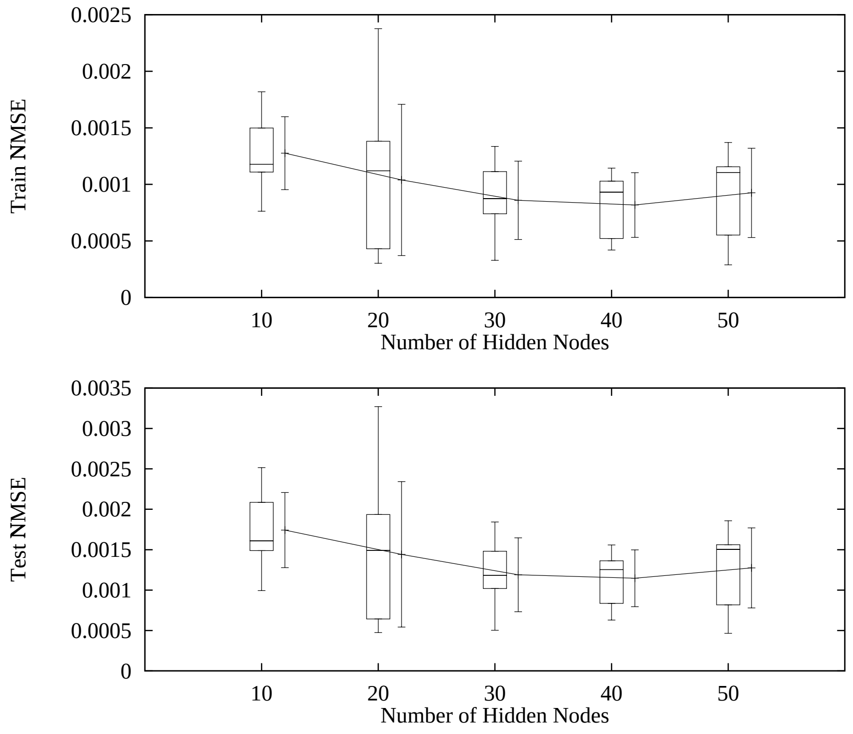

Figure 3. The error for networks with a topology 20:m'_h:1 using 2,000 training points. The graph on the top is the training error. The graph on the bottom is the test error. The abscissa corresponds to the number of hidden nodes. Each result is averaged over ten simulations. Box-whiskers plots are shown on the left in each case along with the mean plus or minus one standard deviation which is shown on the right in each case.

Number of hidden nodes	10	20	30	40	50
Number of parameters	221	441	661	881	1101

Table 1. The number of parameters in the networks as the number of hidden nodes is varied from 10 to 50.

It is of interest to observe the effect of noise on this problem. Figure 5 shows the results for the case of 200 training points when Gaussian noise is added to the input data with a standard deviation equal to 1% of the standard deviation of the input data. A similar trend to figure 4 is observed. The best generalization error, on average, is obtained for networks containing 40 hidden nodes in this case.

The results in this section should not be taken to indicate that oversized networks should always be used. However, the results do indicate that oversized networks may generalize well, and that if training is more successful in the larger networks then it is possible for the larger to also generalize better than the smaller networks. A few observations:

1. It remains desirable to find solutions with the smallest number of parameters.

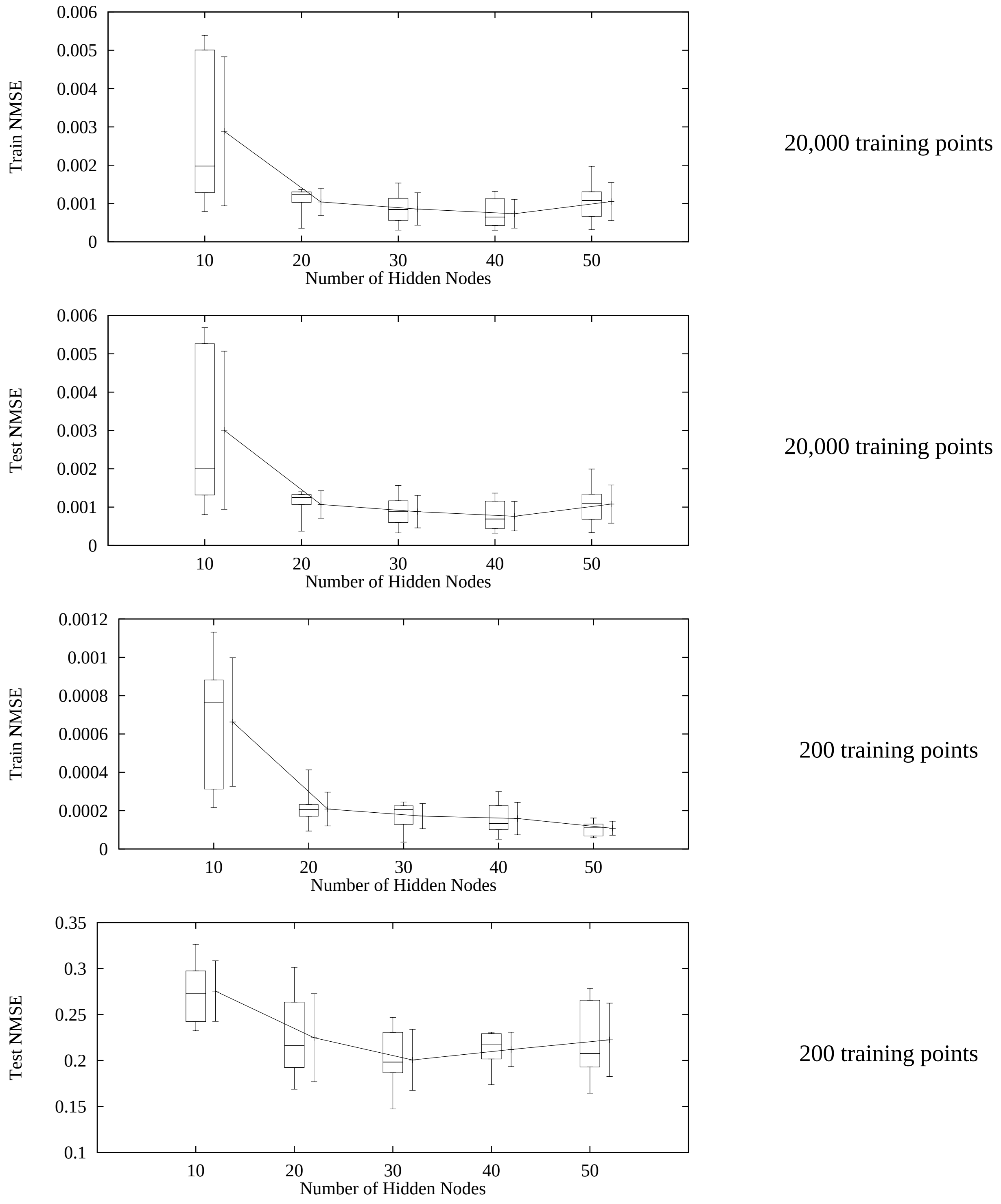

Figure 4. The error for the trained networks as shown in figure 3 when using 20,000 training points and when using 200 training points. From top to bottom: 20,000 training points – training error, 20,000 training points – test error, 200 training points – training error, 200 training points – test error.

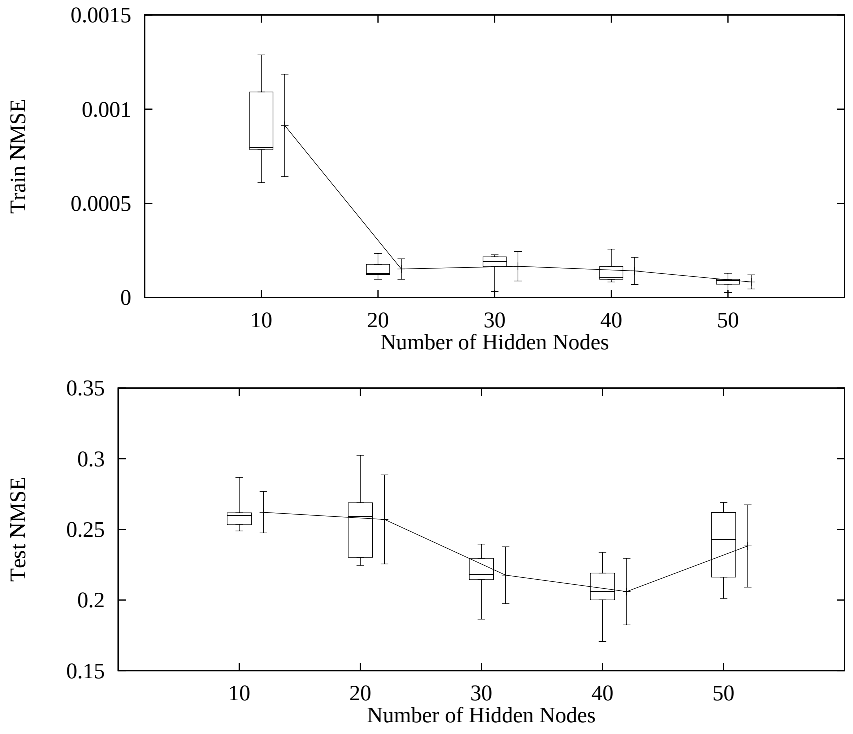

Figure 5. The error for the trained networks as shown in figure 4 for 200 training points and with noise added to the input data. The noise was Gaussian with a standard deviation equal to 1% of the standard deviation of the original data. Top to bottom: training and test errors.

2. A similar result would not be expected if a globally optimal solution was found in the small networks, i.e. if the 10 hidden unit networks were trained to zero error then it would be expected that any networks with extra degrees of freedom would result in worse performance.
3. The distribution of the results is important. For example, observe in figure 4 that the advantage of the larger networks for 20,000 training points is decreased when considering the minimum error rather than the mean error.
4. The number of trials is important. If sufficiently many trials are performed then it should be possible to find a near optimal solution in the optimal size networks (in the limit of an infinite number of random starting points, finding a global optimum is guaranteed). Any advantage from using larger size networks would be expected to disappear.
5. Note that there has deliberately been no control of the generalization capability of the networks (e.g. using a validation set or weight decay), other than a maximum number of updates. There are many solutions which fit the training data well that will not generalize well. Yet, contrary to what might be expected, the results indicate that it is possible for oversized networks to provide better generalization. Successive pruning and retraining of a larger network (Hassibi and Stork, 1993) may arrive at a network with similar size to the smaller networks here but with improved training and generalization error.
6. In terms of computation on serial machines, it may be desirable to investigate performance when the number of individual weight updates (the number of iterations times the number of weights) is equal rather than the number of training iterations (i.e. more training iterations could be done in the same time for the smaller networks). What these results show is that the local optimum found for larger networks

may be better when doing the same number of updates per weight and also that this may correspond to better generalization, i.e. this says something about the nature of the error surface as the network size is increased (the extra degrees of freedom may help avoid poor local minima). The experiments in the next section with a face recognition problem show that it is possible to observe the same phenomenon even when the smaller networks are trained for a greater number of iterations.

6.1.1 Degrees of Freedom

Rules based on the degrees of freedom in the model have been proposed for selecting the topology of an MLP, e.g. *"The number of parameters in the network should be (significantly) less than the number of examples"* or *"Each parameter in an MLP can comfortably store 1.5 bits of information. A network with more than this will tend to memorize the data."* (according to CMU folklore).

These rules aim to prevent overfitting, but they are unreliable as the optimal number of parameters is likely to depend on other factors, e.g. the quality of the solution found, the distribution of the data points, the amount of noise, and the nature of the function being approximated.

Specific rules, such as those mentioned above, are not commonly believed to be accurate (Sarle, 1996). However, the stipulation that the number of parameters must be less than the number of examples is typically believed to be true for common datasets. The results here indicate that this is not always the case.

Face Recognition Example This section presents results on real data. Figure 7 shows the results of training an MLP to classify 10 people from images of their faces[8]. The training set contains 5 images per person, for a total of 50 training patterns[9]. The test set contained a different set of 5 images per person. A small window was stepped over the images and the image samples at each point were quantized using a two dimensional self-organizing map (Kohonen, 1995). The outputs of the self-organizing map for each image sample were used as the inputs to the MLP . A subset of the images is shown in figure 6. In each case, the networks were trained for 25,000 updates. The networks used contain many more parameters than the number of training points, as shown in table 2, yet the best training error and the best generalization error corresponds to the largest model. Note that a) generalization has not been controlled using, for example, a validation set or weight decay, and b) overfitting would be expected with sufficiently large networks.

When simulated on serial machines, larger networks require longer training times for the same number of updates. Hence, it is of interest to compare what happens when the smaller networks are trained for longer

[8]This is not proposed as an intelligent face recognition technique.

[9]The database used is the ORL database which contains a set of faces taken between April 1992 and April 1994 at the Olivetti Research Laboratory in Cambridge and is available from `http://www.cam-orl.co.uk/facedatabase.html`. There are 10 different images of 40 distinct subjects in the database. There are variations in facial expression and facial details. All the images are taken against a dark homogeneous background with the subjects in an up-right, frontal position, with tolerance for some tilting and rotation of up to about 20 degrees. There is some variation in scale of up to about 10%. The images are greyscale (256 levels) with a resolution of 92x112.

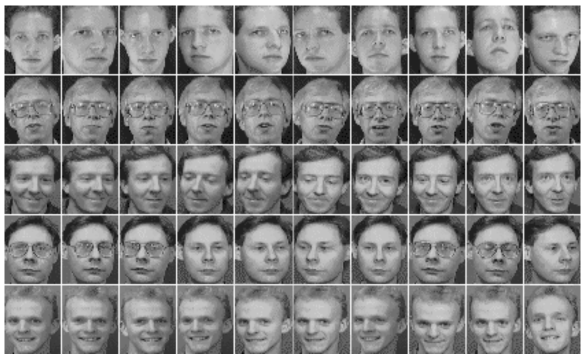

Figure 6. A subset of the face images used in the experiments.

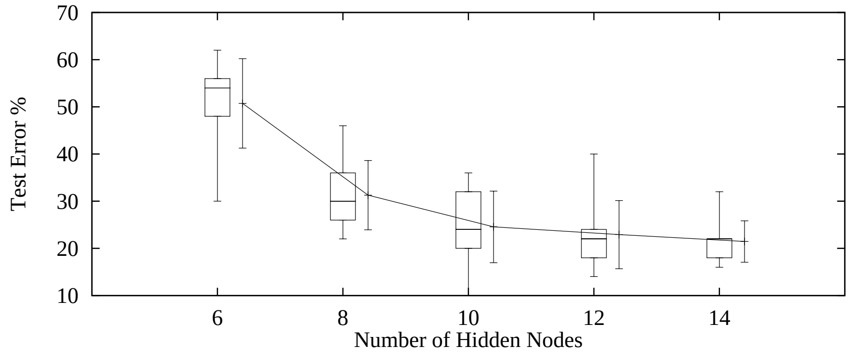

Figure 7. Face recognition experiments showing that the optimal number of weights in a network can be much larger than the number of data points. The smallest model, with six hidden nodes, has 156 times more parameters than training points (7810 parameters). The largest model, with 14 hidden nodes, has 364 times more parameters than training points (18210 parameters). The test error is given as the percentage of examples incorrectly classified.

Number of hidden nodes	6	8	10	12	14
Number of weights	7810	10410	13010	15610	18210

Table 2. The number of parameters in the face recognition network as the number of hidden nodes is increased.

than the larger networks. Figure 8 presents the results of training all of the networks twice as long. It can be observed that the results do not change significantly.

Experiments with MLP networks for speech phoneme recognition have also suggested that better performance can be obtained by overspecifying the number of parameters and using a cross-validation set to terminate training (Renals, Morgan, Cohen and Franco, 1992; Robinson, 1994).

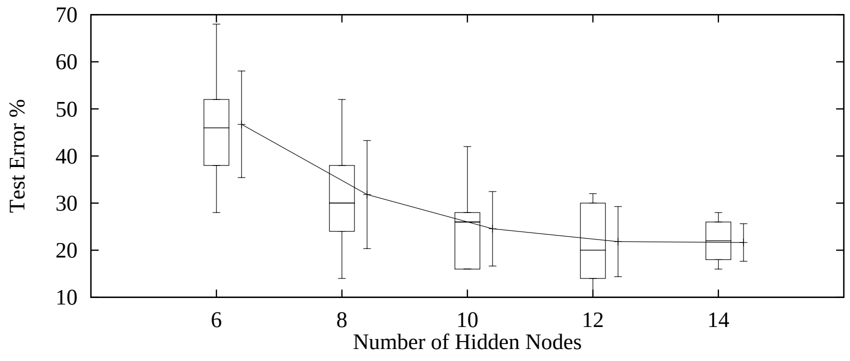

Figure 8. As per figure 7 with the networks trained for twice as long.

6.2 Function Complexity

This section investigates the quality of the solution found by backpropagation as the maximum random weight in the generating network, K, is increased. Figure 10 shows the results for target and generating networks with topology $20 : 10 : 1$. $N_{tr} = 2000$ and $N_{te} = 5000$. It is difficult to visualize the target function as K is increased. A simple method which provides an indication of the complexity is plotted in figure 9 and is created as follows:

```
for each output $o$
    for each input $i$
        set all inputs $\neq i$ equal to $0$
        plot output $o$ as input $i$ is varied from -2 to 2
        repeat $10$ times
            set all inputs $\neq i$ equal to Gaussian random values ($\mu = 0$, $\sigma^2 = 1$)
            plot output $o$ as input $i$ is varied from -2 to 2
```

From figure 9, it can be observed that the function “complexity” increases when K is increased from 1 to 5. Hence, K may be considered as a parameter controlling the function “complexity” although a precise definition of “complexity”[10] is not being used.

Again, the optimal training set error for all networks is zero because $m'_h \geq m_h$. As K increases, corresponding to the target function becoming more “complex”, it can be observed that the solution found becomes significantly worse. Worse generalization may be expected as K increases, however the focus here is on the training error when compared to the optimal error of zero.

This behaviour can be explained by considering the error surface as K is increased. MLP error surfaces can have many areas with shallow slopes in multiple dimensions (Hecht-Nielsen, 1990). This is typically a

[10]Large weights do not always correspond to target functions which are not “smooth”, for example this is not the case when fitting the function sech(x) using two $tanh$ sigmoids (Cardell, Joerding and Li, 1994) (because sech$(x) = \lim_{d\to 0}(tanh(x+d) - tanh(x))/d$, i.e. the weights become indefinitely large as the approximation improves).

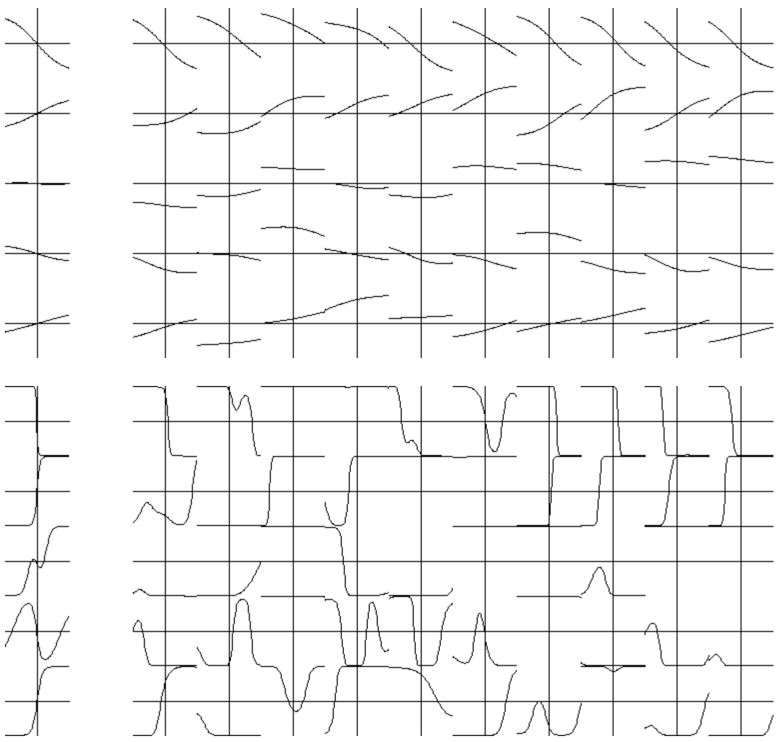

Figure 9. Plots indicating the complexity of the target function for varying K. Above $K = 1$, below $K = 5$. The networks used for creating these plots had the topology 20:10:1. The rows correspond to the first five inputs of these networks. The first column corresponds to the case where all other inputs are set to zero, and the remaining columns correspond to the cases where the other inputs are set to random values, as per the pseudo-code. The abscissa of each individual plot corresponds to the value of the input for that row and ranges from -2 to 2. The ordinate of each individual plot corresponds to the network output.

result of the weighted sum at one or more of the hidden nodes becoming large, which causes the output of these hidden nodes to become insensitive to small changes in the respective weights (the unit is operating in a tail of the sigmoid function where the slope is small). As K increases, the optimal solution requires the nodes to operate in the region where the slope is small more often.

This result highlights a point regarding the application of MLP models: the nature of the target function is often not considered in detail. Perhaps consideration of the implicit bias towards “smooth” models can be of help and preprocessing efforts could be directed towards formulating the required approximation to better suit the MLP. Additionally, it is expected that if the required approximation is “smoother” then the weights of the network are less likely to be driven towards large values, nodes are less likely to become saturated, and generalization performance may be expected to improve.

6.3 Ensembles of Networks and Noisy Training Data

Committees, or an ensemble of networks, are known to be able to improve generalization performance (Jacobs, 1995; Drucker, Cortes, Jackel, Le Cun and Vapnik, 1994; Krogh and Vedelsby, 1995; Perrone and

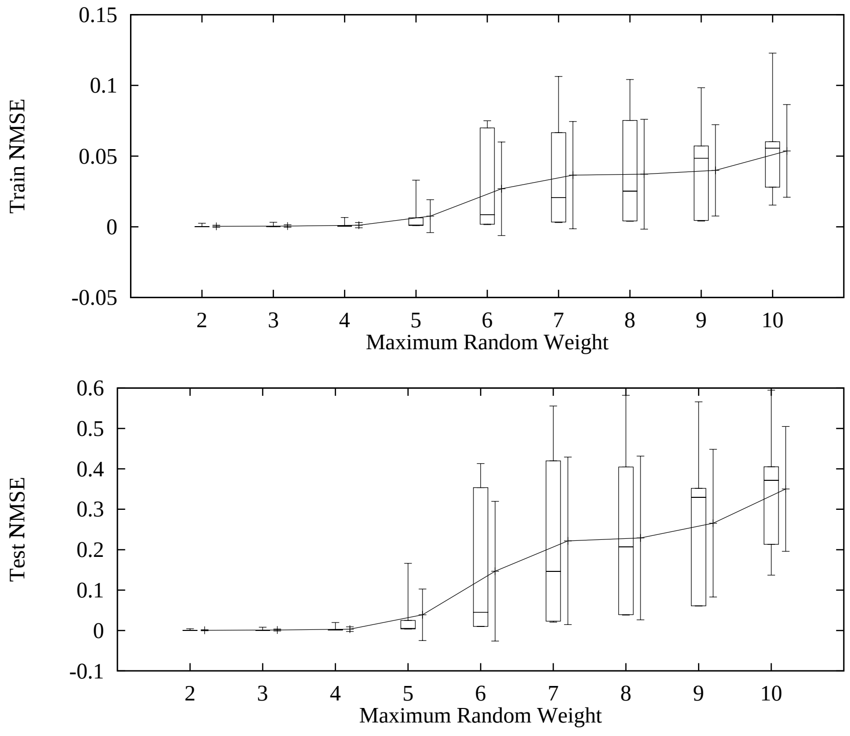

Figure 10. Error for networks with the topology 20:10:1 as the maximum random weight for the generating network is increased. The graph on the top is the training error. The graph on the bottom is the test error. The abscissa corresponds to the value of K. Each result is averaged over ten simulations – plus and minus one standard deviation error bars are plotted at each point.

Cooper, 1993). This section investigates the effect of using committees as the amount of noise added to the training data increases. A simple weighted ensemble of networks is used. Consider the bias/variance dilemma as in (Geman, Bienenstock and Doursat, 1992) where the MSE may be decomposed into bias and variance components:

$$\text{MSE}_{bias} = (E_D[f(x)] - E[y|x])^2 \tag{1}$$

$$\text{MSE}_{variance} = E_D\left[(f(x) - E_D[f(x)])^2\right] \tag{2}$$

where E_D represents the expectation with respect to a training set, D, and $f(x)$ is the approximating function. With finite training data, reducing the bias generally increases the variance and vice versa. For a multilayer perceptron, there is another variance term due to convergence to local minima which can be reduced using ensembles, and the effect of this reduction is greater if the individual networks have larger variance (see (Naftaly, Intrator and Horn, 1995)). Increasing noise levels, and the resulting poorer convergence, may induce this condition. Therefore, it is expected that the ensemble technique may be more beneficial as the noise level is increased.

Figure 11 shows the results of using 1 to 4 committee networks as the standard deviation of zero mean Gaussian noise added to the training data is varied from 0 to 2% of the standard deviation of the input data. It can be observed that the use of more networks in the ensemble does appear to be more successful as the noise is increased. The networks had topology 20:10:1 and were trained for 5×10^5 updates. N_{tr} was 2,000 and each result was averaged over 10 different starting conditions.

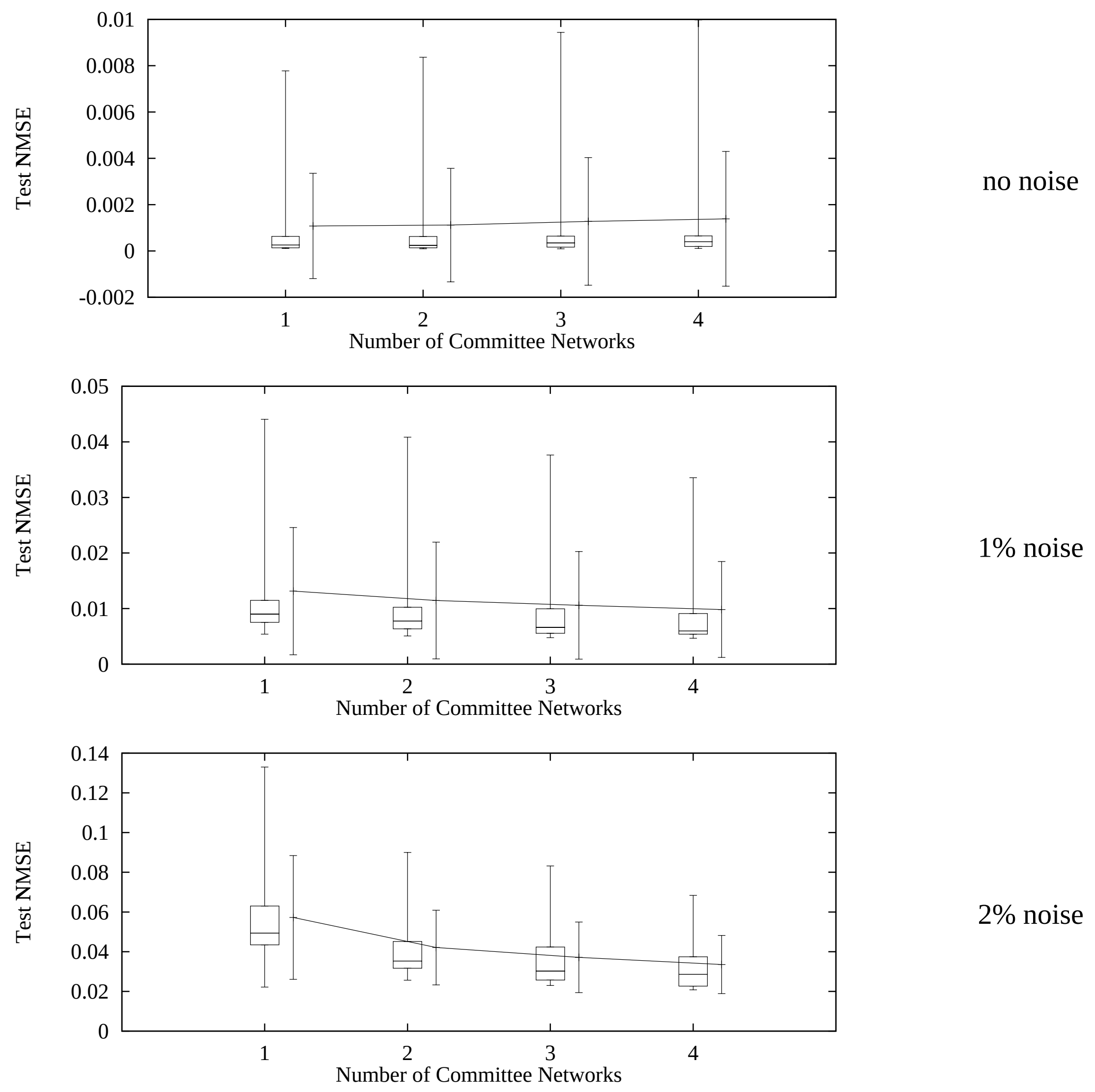

Figure 11. Test error as the number of committee networks is increased from 1 to 4 in each plot. From top to bottom: no noise, and Gaussian noise with a standard deviation of 1%, and 2% of the standard deviation of the input data respectively.

7 Discussion

Using an artificial task where the optimal training error was known and a sample real world problem along with a pre-specified maximum number of training updates, it was shown that:

1. The solution found by gradient descent with a practical training time is often significantly worse than a globally optimal solution.

2. In certain cases, the best generalization can occur when the number of parameters in the network is greater than the number of data points. This can occur when the larger networks are easier to train.

   This result should not be taken to indicate that oversized networks should always be used. However, the result does indicate that, in certain cases, training can be more successful in larger networks, and consequently, it is possible for larger networks to result in improved generalization. It remains desirable to find solutions with the smallest number of parameters.

3. The solution found by gradient descent can be significantly worse as the function to be approximated becomes more complex.

4. The use of ensemble techniques can be increasingly beneficial as the level of noise in the training data increases.

5. Given the set of functions that a particular MLP can approximate and a pre-specified maximum number of training updates, certain functions are "harder" to approximate using backpropagation than others.

7.1 Network Size and Degrees of Freedom

A simple explanation for why larger networks can sometimes provide improved training and generalisation error is that the extra degrees of freedom can aid convergence, i.e. the addition of extra parameters can decrease the chance of becoming stuck in local minima or on "plateaus", etc. (Kröse and van der Smagt, 1993).

This section provides plots of the function approximated by the trained networks and the network weights which indicate the following: a) the function approximated by the oversized networks remains relatively "smooth", and b) after training, the extra degrees of freedom in the larger networks contribute to the function approximated in only a minor way. The plots in this section were created using a smaller task in order to aid visualisation: the generating networks had topology 5:5:1 and the trained networks had 5, 15, and 25 hidden units. 1,000 training data points were used and the random weight maximum value was 2.

Figures 12 to 14 provide an indication of the function approximated by the networks as the network size (5, 15, and 25 hidden units) and amount of noise (Gaussian noise with standard deviation 0, 5%, and 10% of the input standard deviation) in the data are varied. The plots are generated as described in section 4. The dotted lines show the target function (from the generating network) and the solid line shows the function approximated by the trained network. Observations: a) the training network approximation tends to be less accurate for the optimal size network, b) the approximation appears relatively "smooth" in all cases.

Figures 15 to 17 show the weights in the trained networks as the network size (5, 15, and 25 hidden units) and amount of noise (Gaussian noise with standard deviation 0, 5%, and 10% of the input standard deviation)

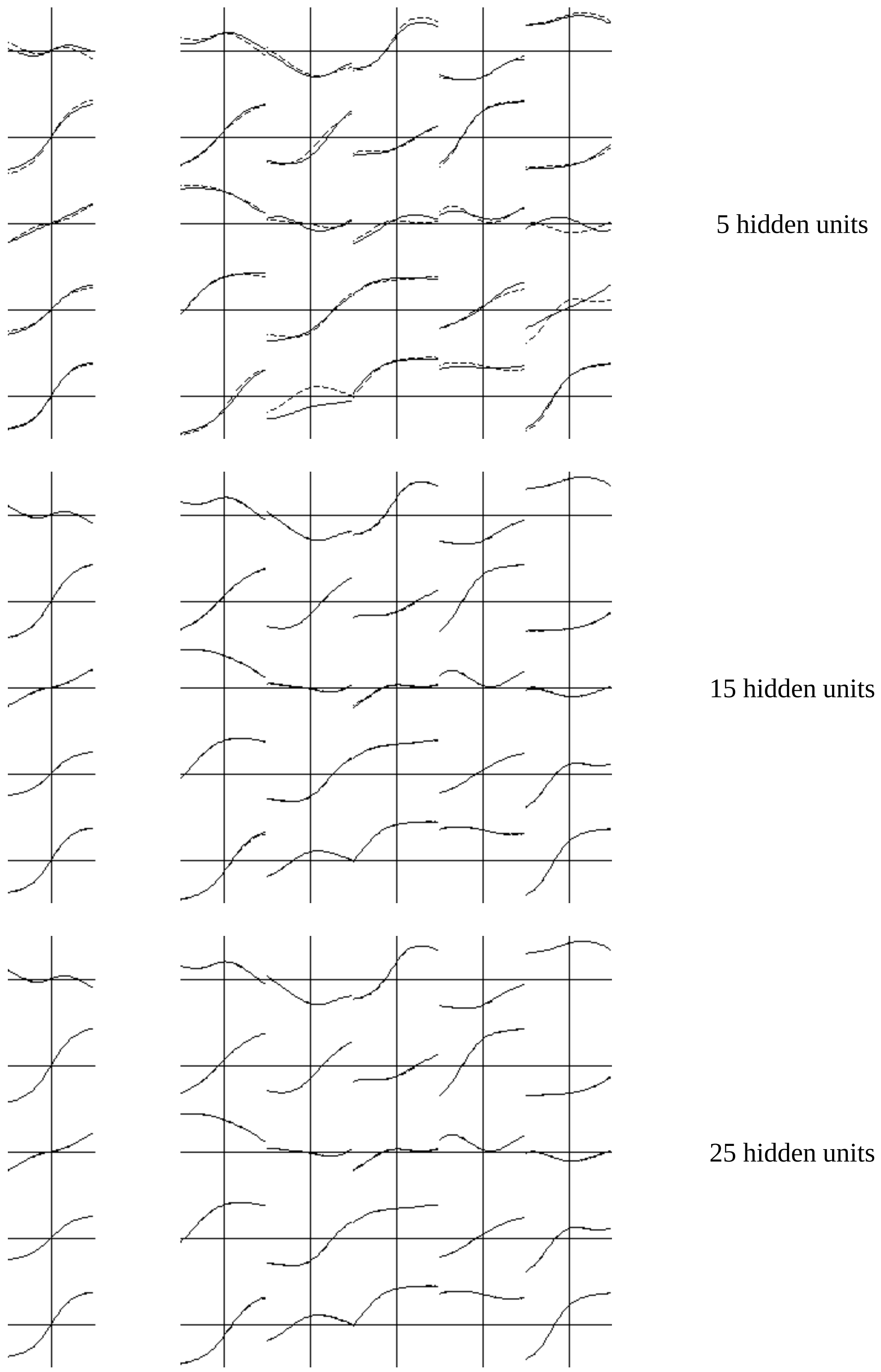

Figure 12. The function approximated by networks with 5, 15, and 25 hidden units for the case of no noise. The plots are generated as described in section 4. The dotted lines show the target function (from the generating network) and the solid line shows the function approximated by the trained network.

in the data are varied. Each diagram is plotted as follows: The columns (1 to 6) correspond to the weights

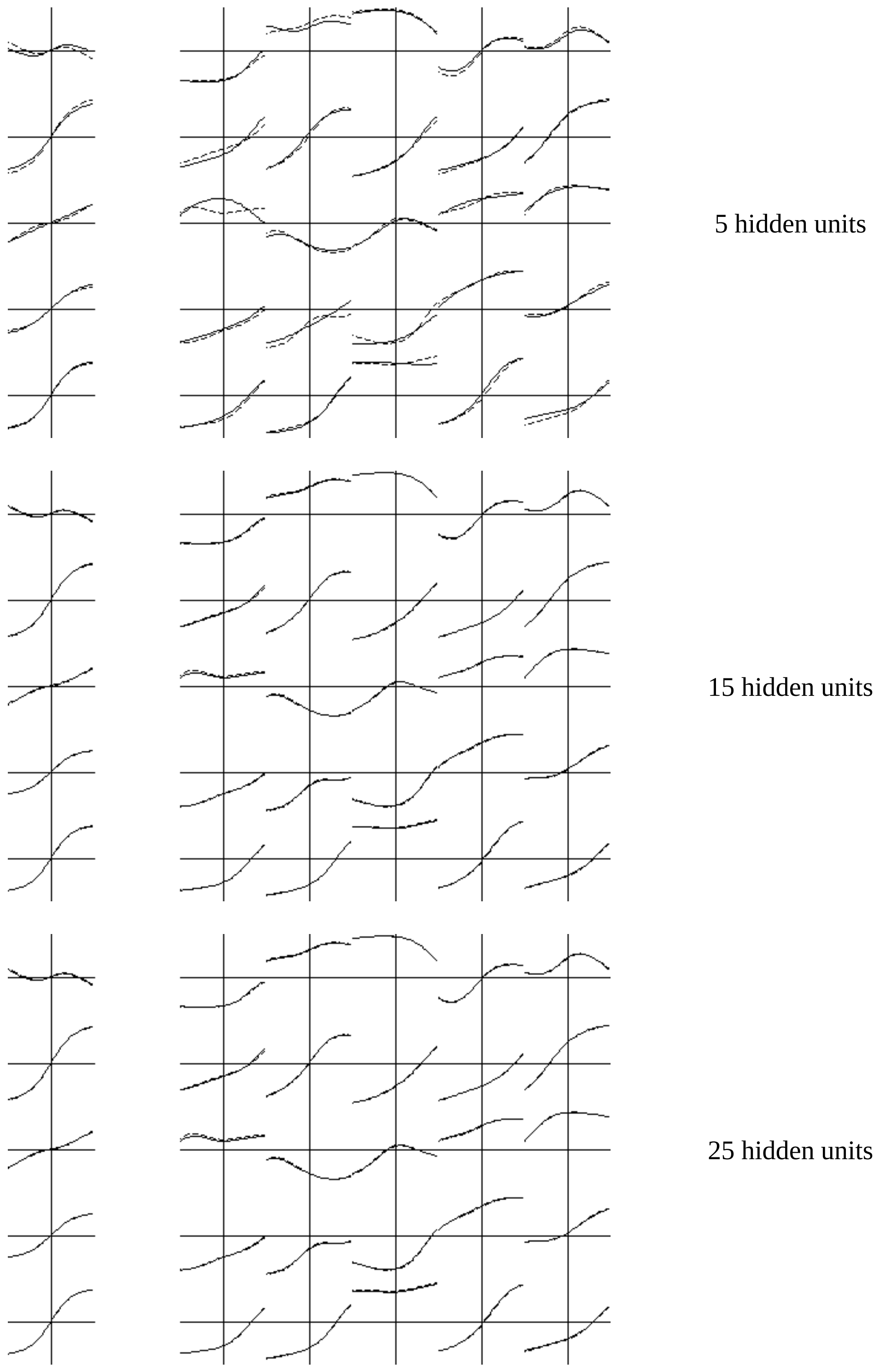

Figure 13. The function approximated by networks with 5, 15, and 25 hidden units for the case of Gaussian noise with standard deviation 5% of the standard deviation of the inputs. The plots are generated as described in section 4. The dotted lines show the target function (from the generating network) and the solid line shows the function approximated by the trained network.

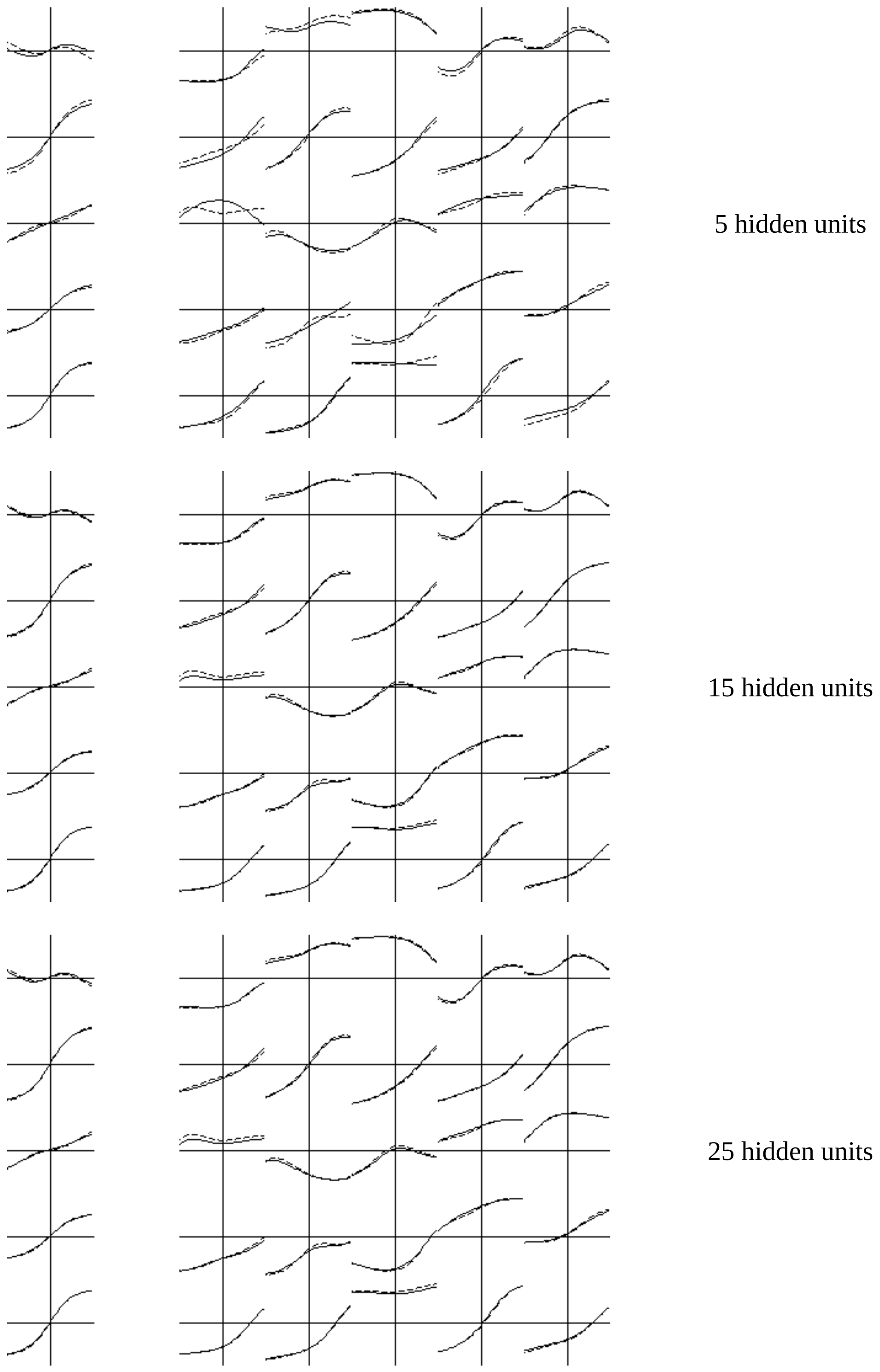

Figure 14. The function approximated by networks with 5, 15, and 25 hidden units for the case of Gaussian noise with standard deviation 10% of the standard deviation of the inputs. The plots are generated as described in section 4. The dotted lines show the target function (from the generating network) and the solid line shows the function approximated by the trained network.

from the hidden nodes to the bias and the 5 input nodes. The rows are organised into groups of two with a space between each group. The number of groups is equal to the number of hidden nodes in the trained network. For the two rows in each group, the top row corresponds to the generating network and the bottom row corresponds to the trained network. The idea is to compare the weights in the generating and trained networks. There are a couple of difficulties which arise in this comparison which are resolved as follows. Firstly, there is no reason for hidden node 1 in the generating network to correspond to hidden node 1 in the trained network, etc. This problem is resolved by finding the best matching set of weights in the trained network for each hidden unit in the generating network (using the Euclidean distance between the weight vectors), and matching the hidden nodes of the trained and generating networks accordingly. Additionally, these best matches are ordered according to the respective distances between the weight vectors, i.e. the top two rows shows the generating network hidden node which was best approximated by a hidden node in the trained network. Likewise, the worst match is at the bottom. A second problem is that trying to match the weights from the hidden nodes to the input nodes does not take into account the output layer weights, e.g. exactly the same hidden node function could be computed with different weights if the hidden nodes weights are scaled and the output layer weights are scaled accordingly. For the case of only one output which is considered here, the solution is simple: the hidden layer weights are scaled according to the respective output layer weight. Each individual weight (scaled by the appropriate output weight) is plotted as follows: the square is shaded in proportion to the magnitude of the weight, where white equals 0 and black equals the maximum value for all weights in the networks. Negative weights are indicated by a white square inside the outer black square which surrounds each weight.

Observations: a) the generating network weights are often matched more closely by the larger networks (consider the fourth and fifth best matching groups of two rows), b) the extra weights in the larger networks contribute to the final approximation in only a minor way, and c) the results indicate that pruning (and optionally retraining) the larger networks may perform well.

A conclusion is that backpropagation can result in the underutilisation of network resources in certain cases (i.e. some parameters may be ineffective or only partially effective due to sub-optimal convergence).

Since reporting the work contained in this paper, S. Hanson has stated (Hassoun, Cherkassky, Hanson, Oja, Sarle and Sudjianto, 1996): *"Whether in the language of approximation theory (overfitting etc.) or statistical estimation (bias vs. variance) it is clear that too many parameters in some nonparametric models can be grievous, however with many Neural Networks, more parameters can actually improve things."* and *"Such [phenomena] which arise uniquely in Neural Network applications should be more of a focus for statisticians rather than an anomaly to be ignored"*. This section has investigated the phenomenon under controlled conditions and discussed how the phenomenon may arise.

The phenomenon has also been observed by others, e.g. Dadson (1996) states *"I find that in practice networks with a very parsimonious number of neurons are hard to train"*, Slomka (1996) states that slightly larger than optimal size networks have often improved performance, and Back (1992) states that *"overparameterized synapses give lower MSE and variance than exact order synapses"* in the context of modelling nonlinear systems with FIR and IIR MLP networks.

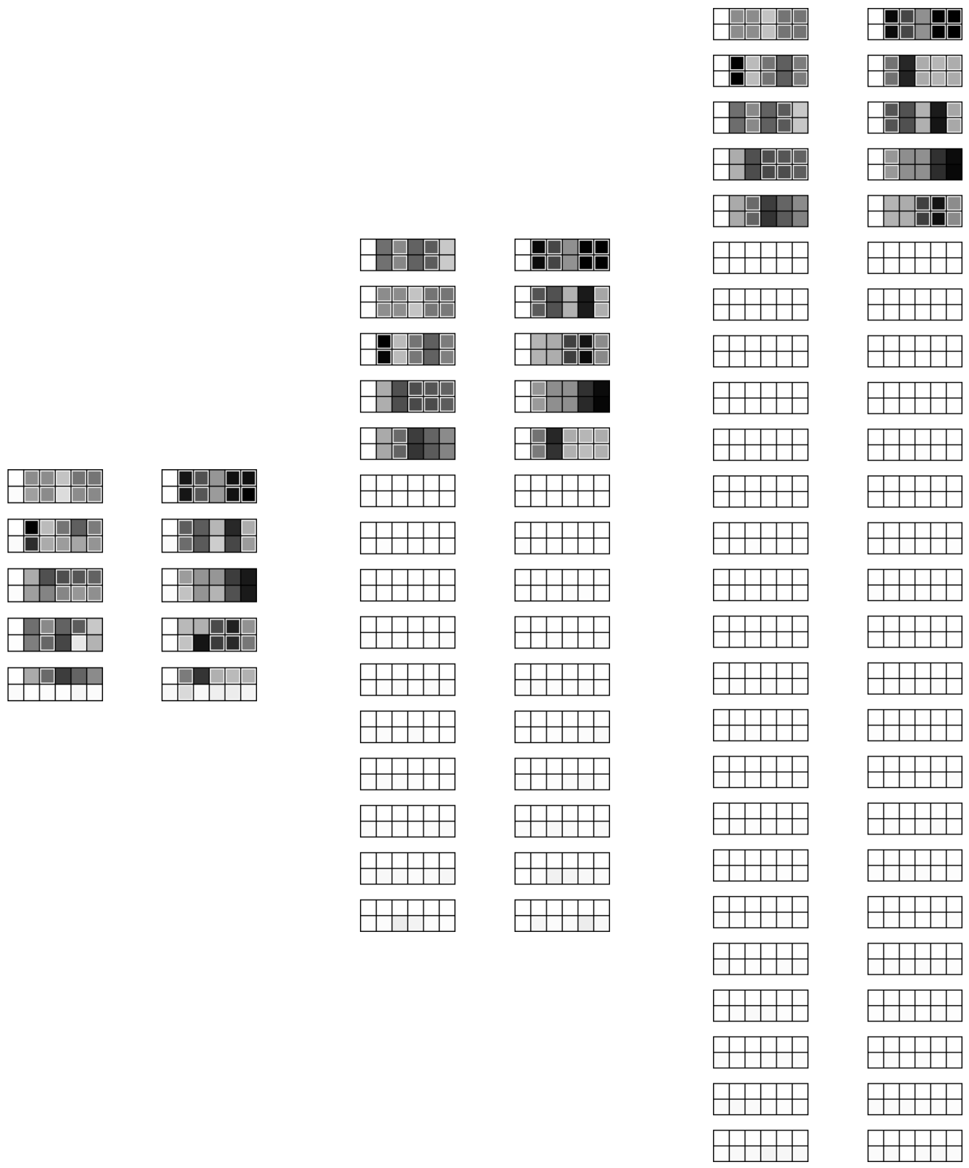

Figure 15. The weights after training in networks with 5, 15, and 25 hidden units for the case of no noise. In each case, the results are shown for two networks with different random starting weights. The plotting method is described in the text. For each pair of rows, the top row corresponds to the generating network and the bottom row corresponds to the trained network. Observe that a) the generating network weights are often matched more closely by the larger networks (compare the fourth and fifth set of two rows), and b) the extra weights in the larger networks contribute to the final approximation in only a minor way.

7.2 Occam's Razor

The results showing that larger than optimal size networks can generalize better, in certain cases, are not in contradiction with Occam's razor. Occam's razor, which advocates the simpler out of a number of possible solutions, is not applicable to the situation where each solution is of a different quality, i.e. while larger

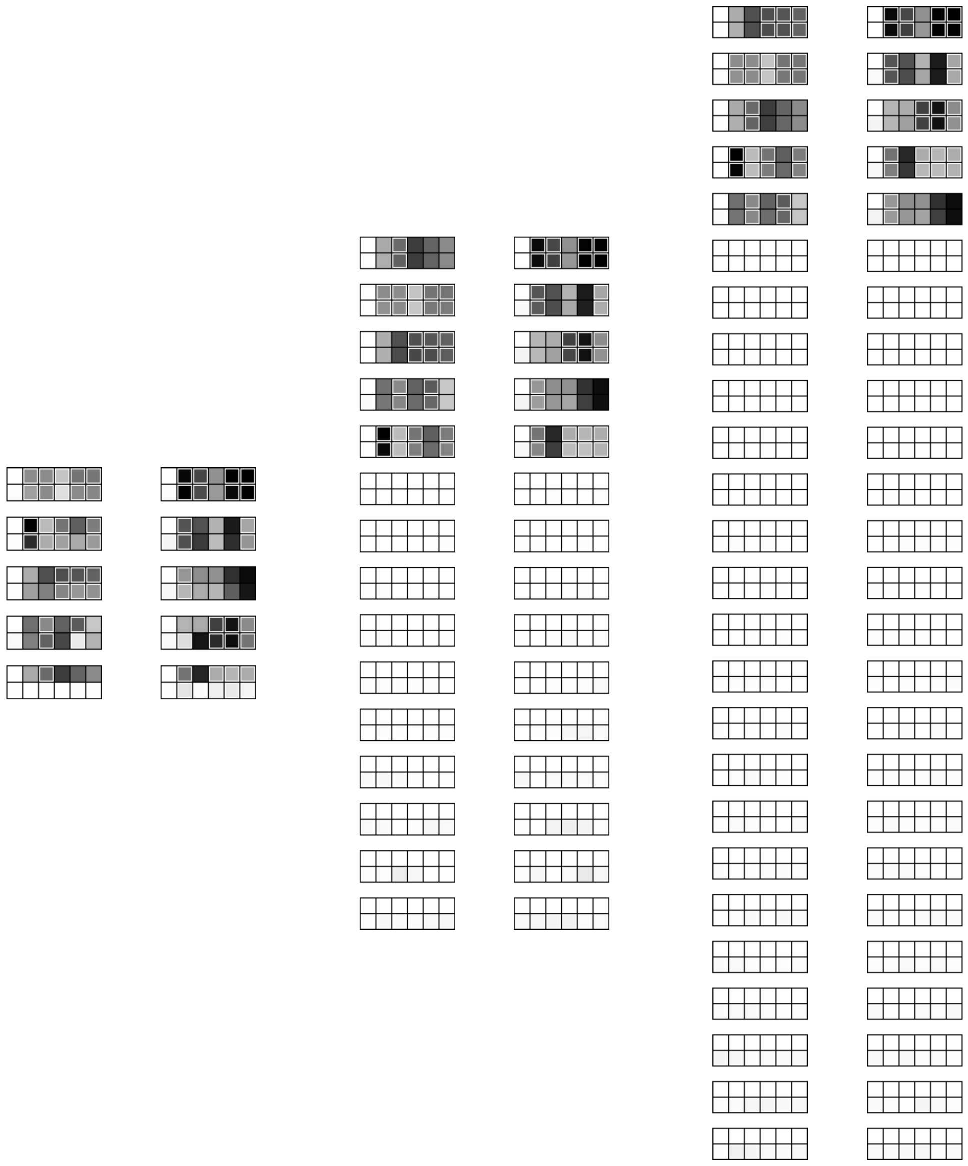

Figure 16. The weights after training in networks with 5, 15, and 25 hidden units for the case of Gaussian noise with standard deviation 5% of the standard deviation of the inputs. In each case, the results are shown for two networks with different random starting weights. The plotting method is described in the text. For each pair of rows, the top row corresponds to the generating network and the bottom row corresponds to the trained network. Observe that a) the generating network weights are often matched more closely by the larger networks (compare the fourth and fifth set of two rows), and b) the extra weights in the larger networks contribute to the final approximation in only a minor way.

networks can provide improved generalization performance, this typically only happens when the larger networks are better models of the training data.

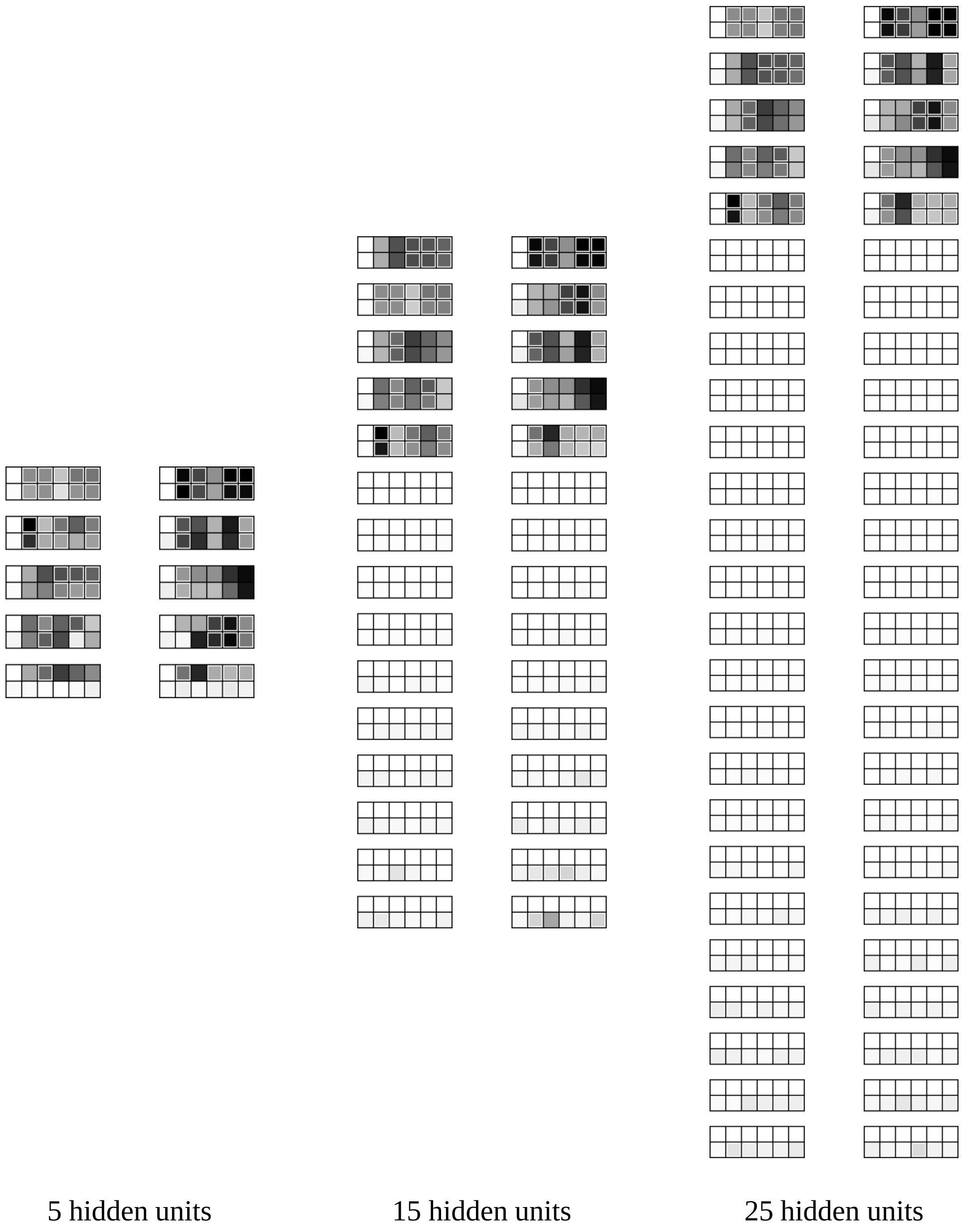

Figure 17. The weights after training in networks with 5, 15, and 25 hidden units for the case of Gaussian noise with standard deviation 10% of the standard deviation of the inputs. In each case, the results are shown for two networks with different random starting weights. The plotting method is described in the text. For each pair of rows, the top row corresponds to the generating network and the bottom row corresponds to the trained network. Observe that a) the generating network weights are often matched more closely by the larger networks (compare the fourth and fifth set of two rows), and b) the extra weights in the larger networks contribute to the final approximation in only a minor way.

7.3 Learning Theory

The results are also not in contradiction with statistical learning theory. Vapnik (1995) states that machines with a small VC dimension are required to avoid overfitting. However, he also states that *"it is difficult to* *approximate the training data"*, i.e. for a given problem in MLP approximation, the goal is to find the appropriate network size in order to minimize the tradeoff between overfitting and poor approximation. Vapnik suggests that the use of *a priori* knowledge is required for small training error and small generalisation error.

For the case of linear output neurons, Barron (1991; 1992) has derived the following bound on the total risk for an MLP estimator:

$$O\left(\frac{C_f}{m_h}\right) + O\left(\frac{m_h m_i}{N_{tr}} \log N_{tr}\right) \tag{3}$$

where C_f is the first absolute moment of the Fourier magnitude distribution of the target function f and is a measure of the "complexity" of f. Again, a tradeoff can be observed between the accuracy of the best approximation (which requires larger m_h), and the avoidance of overfitting (which requires a smaller m_h/N_{tr} ratio). However, this does not take into account limited training time and different rates of convergence for different f. The left-hand term (the approximation error) corresponds to the error between the target function and the closest function which the MLP can implement. For the artificial task, the approximation error is zero for $m'_h \geq 10$. Based on this equation, it is likely that $m'_h = 10$ would be selected as the optimal network size (note that the results reported here use sigmoidal rather than linear output neurons).

Recent work by Bartlett (1996) correlates with the results reported here. Bartlett comments: *"the VC-bounds seem loose; neural networks often perform successfully with training sets that are considerably smaller than the number of network parameters"*. Bartlett shows (for classification) that the number of training samples only needs to grow according to A^{2l} (ignoring log factors) to avoid overfitting, where A is a bound on the total weight magnitude for a neuron and l is the number of layers in the network. This result and either an explicit (weight decay etc.) or implicit bias towards smaller weights leads to the phenomenon observed here, i.e. larger networks may generalize well and therefore better generalization is possible from larger networks if they can be trained more successfully than the smaller networks.

7.4 The Curse of Dimensionality

Consider $x_i \in \mathcal{R}^n$. The regression, $f(x)$ is a hypersurface in $\mathcal{R}^n$. If $f(x)$ is arbitrarily complex and unknown then dense samples are required to approximate the function accurately. However, it is hard to obtain dense samples in high dimensions. This is the "curse of dimensionality" [11] (Friedman, 1995). The relationship between the sampling density and the number of points required is $\approx N^{\frac{1}{n}}$ (Friedman, 1995) where n is the dimensionality of the input space and N is the number of points. Thus, if N_1 is the number of points for a given sampling density in 1 dimension, then in order to keep the same density as the dimensionality is increased, the number of points must increase according to N_1^n.

Kolmogorov's theorem shows that any continuous function of n dimensions can be completely characterized by a 1-dimensional continuous function. Specifically, Kolmogorov's theorem (Friedman, 1995; Kolmogorov, 1957; Kůrková, 1991; Kůrková, 1995) states that any continuous

$$f(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{2n+1} g_f\left(\sum_{i=1}^{n} \lambda_i Q_j(x_i)\right) \tag{4}$$

[11] Other definitions of the "curse of dimensionality" exist, however we use the definition of Friedman (1995).

where $\{\lambda_i\}_1^n$ are universal constants that do not depend on f, $\{Q_j(z)\}_1^{2n+1}$ are universal transformations which do not depend on f, and $g_f(u)$ is a continuous, one-dimensional function which totally characterises $f(x_1, x_2, \ldots, x_n)$ (g_f is typically highly nonsmooth), i.e. there is a one dimensional continuous function that characterises any continuous function of n arguments. As such, we can see that the problem is not so much the dimensionality, but the complexity of the function (high dimensional functions typically have the potential to be more complex) (Friedman, 1995), i.e. the curse of dimensionality essentially says that in high dimensions, the less data points we have, the simpler the function has to be in order to represent it accurately. The No Free Lunch theorem (Wolpert and Macready, 1995) shows that, if we do not make any assumptions regarding the target function, no algorithm performs better than any other on average. In other words, we need to make assumptions. A convenient and useful assumption (which corresponds to common sensory data in many instances) is that of smoothness. As demonstrated, smoother functions correspond to faster convergence. Intuitively this is reasonable – more complex functions correspond to a greater degree of saturation in the nodes, and the backpropagated error approaches zero in saturated regions.

7.5 Weight Distributions

In certain cases, standard backpropagation can lead to an implicit bias towards smaller weights as the following experiment shows. Networks were trained as before using data generated from a network initialised using $K = 20$, $N_{tr} = 20,000$ (cf. the curse of dimensionality and more points required for more complex target functions), no generalization control, and 5×10^5 iterations. Figure 18 shows box-whiskers plots of the distribution of weights after training for networks with 10 to 50 hidden nodes. Observe that the weights in the trained networks are, on average, smaller than those in the generating network.

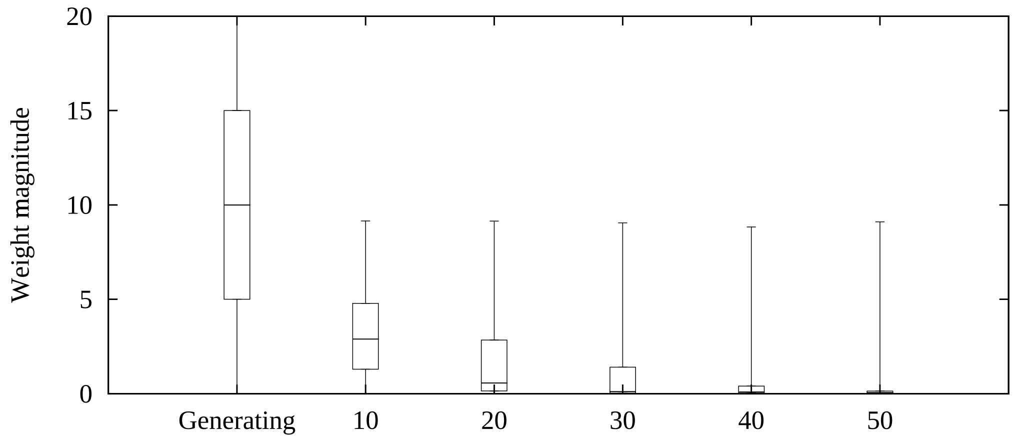

Figure 18. Box-whiskers plots showing the weight magnitude distribution for the generating network (10 hidden nodes, on the left) and the trained networks with 10 to 50 hidden nodes. Averaged over 10 trials in each case.

7.6 Universal Approximation

The results are negative in terms of the possibility of training large homogeneous MLPs to parsimoniously represent arbitrary functions. Even for the case of relatively small maximum weights in the network, it can be seen that convergence may be difficult for the most parsimonious solutions. While large MLPs have been successful in pattern recognition (e.g. speech phoneme recognition (Bourlard and Morgan, 1994)), we suggest that it can be difficult to find parsimonious solutions which employ appropriate internal representations.

With reference to biological neural networks, the reason that we can learn the things we do is, perhaps, critically linked to the pre-wiring of the human brain. For example, we know that we have a lot of difficulty training a chimpanzee to learn a human language, let alone MLPs. This conclusion applies to the homogeneous MLP type of universal approximation approach to learning by example. There is a whole class of algorithms for learning by example which operate on the basis of looking for regularities and then incorporating these regularities into the model (e.g. grammar induction algorithms). In contrast, the computational capacity of MLPs is static.

Finally, we quote Minsky and Papert's epilogue to "Perceptrons" (1988):

"In the early years of cybernetics, everyone understood that hill-climbing was always available for working easy problems, but that it almost always became impractical for problems of larger sizes and complexities. We were very pleased to discover that [perceptron convergence] could be represented as hill-climbing; however that very fact led us to wonder whether such procedures could dependably be generalized, even to the limited class of multi-layer machines we have named Gamba perceptrons. The situation seems not to have changed much – we have seen no contemporary connectionist publication that casts much new theoretical light on the situation. Then why has [backpropagation, gradient descent] become so popular in recent years? In part this is because it is so widely applicable and because it does indeed yield new results (at least on problems of rather small scale). Its reputation also gains, we think, from it being presented in forms that shares, albeit to a lesser degree, the biological plausibility of [the perceptron]. But we fear that its reputation also stems from unfamiliarity with the manner in which hill-climbing methods deteriorate when confronted with larger scale problems.

Minsky and Papert's assertion that local minima and related difficulties are a problem appears to be valid. We agree – it does not appear that standard MLP networks trained with backpropagation can be scaled up to arbitrarily large problems. However, while there are certain fundamental limitations to the performance of this class of learning algorithms, MLPs have produced better results than some notable alternatives, e.g. perceptrons with threshold units (Werbos, 1974). The imposition of the sigmoid non-linearities in MLPs allows the use of gradient descent optimisation and empirical results suggests that the error surface can be (relatively speaking) quite suitable for a gradient descent based optimization process.

8 Appendix A: Generalization and Overfitting

This section provides a brief overview of generalization and overfitting.

Generalization refers to how well a model performs on unseen data, i.e. the model is trained using a given training set and generalization corresponds to the expected performance of the model on new patterns.

Mathematically, the goal of MLP training can be formulated as minimization of a cost function (Bengio, 1996):

$$E_{true} = \int_{\mathbf{x},\mathbf{d}} e(f(\mathbf{x},\mathbf{w}),\mathbf{d})p(\mathbf{x},\mathbf{d}) \ d\mathbf{x}d\mathbf{d} \tag{5}$$

where e is a local cost function, f is the function implemented by the MLP, $\mathbf{x}$ is the input to the model, $\mathbf{d}$ is the desired output of the model, $\mathbf{w}$ corresponds to the weights in the network, and p represents the probability distribution. The objective of training is to optimise the parameters $\mathbf{w}$ such that E_{true} is minimised:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \int_{\mathbf{x},\mathbf{d}} e(f(\mathbf{x},\mathbf{w}),\mathbf{d})p(\mathbf{x},\mathbf{d}) \ d\mathbf{x}d\mathbf{d} \tag{6}$$

E_{true} is the generalization error (Bengio, 1996), i.e. the expected performance of the MLP on new patterns randomly chosen from $p(\mathbf{x},\mathbf{d})$. In practice $p(\mathbf{x},\mathbf{d})$ is not known. Instead, a training set $\mathcal{T} = \{\mathbf{x}_p, \mathbf{d}_p\}_1^{N_p}$ is given, where N_p is the number of patterns, and an approximation of E_{true} is minimised which is called the *empirical error* (Vapnik, 1982) or training error (Bengio, 1996):

$$E = \sum_{p=1}^{N_p} e(\mathbf{x}_p, \mathbf{d}_p) \tag{7}$$

The quadratic and relative entropy cost functions are examples of such an error function.

A very important question is how well a model trained to minimise E generalises (i.e. how low E_{true} is). This is important because low E (performance on the training set) does not necessarily mean low E_{true} (expected performance on new patterns).

An MLP provides a function mapping from input values to desired output values. This mapping is generally “smooth” (in a sense defined by the nature of the activation function, the topology of the network, and the training algorithm) and allows interpolation between the training points. Consider the simple case of an input with only one dimension as shown in figure 19. The training patterns, marked with a cross, contain noise. The true, underlying function mapping may be that shown in the middle graph. However, without any controlling scheme, the MLP may seriously underfit (the left-hand graph in figure 19) or overfit (the right-hand graph in figure 19) the data. Observe that the average error on the training samples is highest for the underfitting graph in figure 19 and lowest for the overfitting graph. For the case of overfitting, the error on the training samples may be very low, but error on test samples may be high (consider test points in between the training points on the overfitting graph), i.e. for a given MLP, as training is continued past the “correct fit” point, generalization performance may decrease. This is the well known bias/variance tradeoff (Geman et al., 1992) – in the underfitting case, the MLP estimator produces estimates which have high bias but low variance (an estimator is said to be biased if, on average, the estimated value is different to the expected value). In the overfitting case, bias of the estimator is low but variance is high. There exists an optimum between the two extremes.

The degree to which overfitting is possible is related to the number of training patterns and the number of parameters in the model. In general, with a fixed number of training patterns, overfitting can occur when the model has too many parameters (too many degrees of freedom). Figure 20 illustrates the idea using polynomial approximation. A training dataset was created which contained 21 points according to the equation $y = \sin(x/3) + \nu$ where ν is a uniformly distributed random variable between -0.25 and 0.25. The equation was evaluated at $0, 1, 2, \ldots, 20$. This dataset was then used to fit polynomial models (Rivlin, 1969; Press, Teukolsky, Vetterling and Flannery, 1992) with orders between 2 and 20. For order 2, the approximation is poor as shown in figure 20. For order 10, the approximation is reasonably good. However, as the order (and number of parameters) increases, significant overfitting is evident. At order 20, the approximated function fits the training data very well, however the interpolation between training points is very poor.

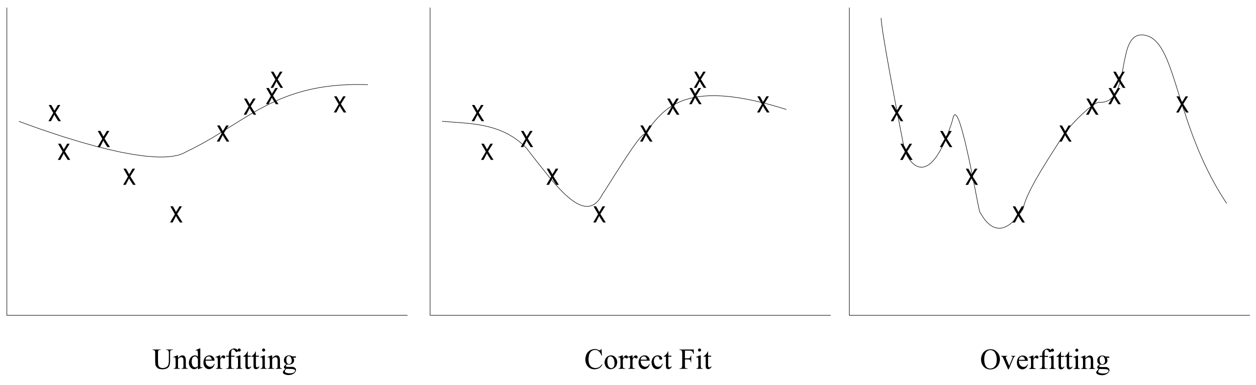

Figure 19. Underfitting and overfitting.

Figure 21 shows the results of using an MLP to approximate the same training set[12]. As for the polynomial case, the smallest network with one hidden unit (4 weights including bias weights), did not approximate the data well. With two hidden units (7 weights), the approximation is reasonably good. In contrast to the polynomial case however, networks with 10 hidden units (31 weights) and 50 hidden units (151 weights) also resulted in reasonably good approximations. Hence, for this particular (very simple) example, MLP networks trained with backpropagation do not lead to a large degree of overfitting, even with more than 7 times as many parameters as data points. It is certainly true that overfitting can be a serious problem with MLPs. However, this example highlights the possibility that MLPs trained with backpropagation may be biased towards smoother approximations.

Figure 22 shows a different example where significant overfitting can be seen in larger MLP models. The same equation was used as for the previous example except the equation was only evaluated at $0, 1, 2, \ldots, 5$, creating 6 data points. The figure shows the results of using MLP models with 1 to 4 hidden nodes. For this example, the 3 and 4 hidden node cases produce an approximation which is expected to result in worse generalization. A test dataset was created by evaluating the equation without noise ($y = \sin(x/3)$) at intervals of 0.1. Tables 3 and 4 show the results on the test set for the models trained on the first and second example respectively. For the first example, the largest network provided the best generalization. However, for the second example, the network with 2 hidden nodes provided the best generalization – larger networks resulted in worse generalization due to overfitting.

Hidden Nodes	1	2	10	50
Training MSE	0.373	0.0358	0.0204	0.0204
Test MSE	0.257	0.0343	0.0222	**0.0201**

Table 3. Results for MLP interpolation of the function $y = \sin(x/3)$ in the range 0 to 20. The best generalization corresponded to the largest network tested which had 50 hidden nodes.

[12]Training details were as follows. A single hidden layer MLP, backpropagation, 100,000 stochastic training updates, and a learning rate schedule with an initial learning rate of 0.5 were used.

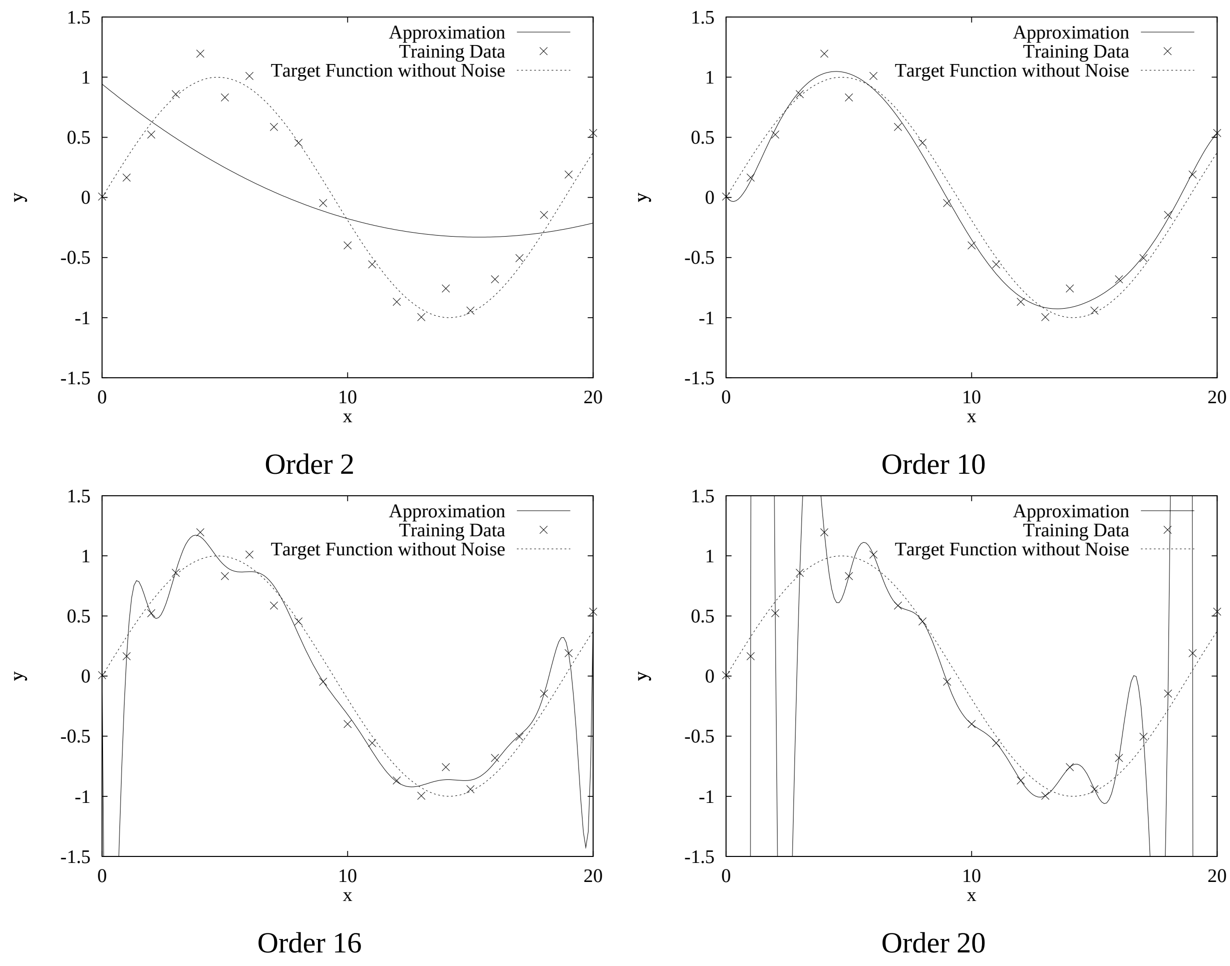

Figure 20. Polynomial interpolation of the function $y = \sin(x/3) + \nu$ in the range 0 to 20 as the order of the model is increased from 2 to 20. ν is a uniformly distributed random variable between -0.25 and 0.25. Significant overfitting can be seen for orders 16 and 20.

Hidden Nodes	1	2	10	50
Training MSE	0.0876	0.0347	4.08e-5	7.29e-5
Test MSE	0.0943	**0.0761**	0.103	0.103

Table 4. Results for MLP interpolation of the function $y = \sin(x/3)$ in the range 0 to 5. The best generalization corresponded to 2 hidden nodes – larger networks resulted in higher error due to overfitting.

Acknowledgments

We would like to acknowledge useful discussions with and comments from R. Caruana, S. Hanson, G. Heath, B.G. Horne, T. Lin, K.-R. Müller, T. Rognvaldsson, and W. Sarle. The opinions expressed herein are those of the authors. This work has been partially supported by the Australian Research Council (ACT) and the Australian Telecommunications and Electronics Research Board (SL). Thanks to the Olivetti Research Laboratory and F. Samaria for compiling and maintaining the ORL database.

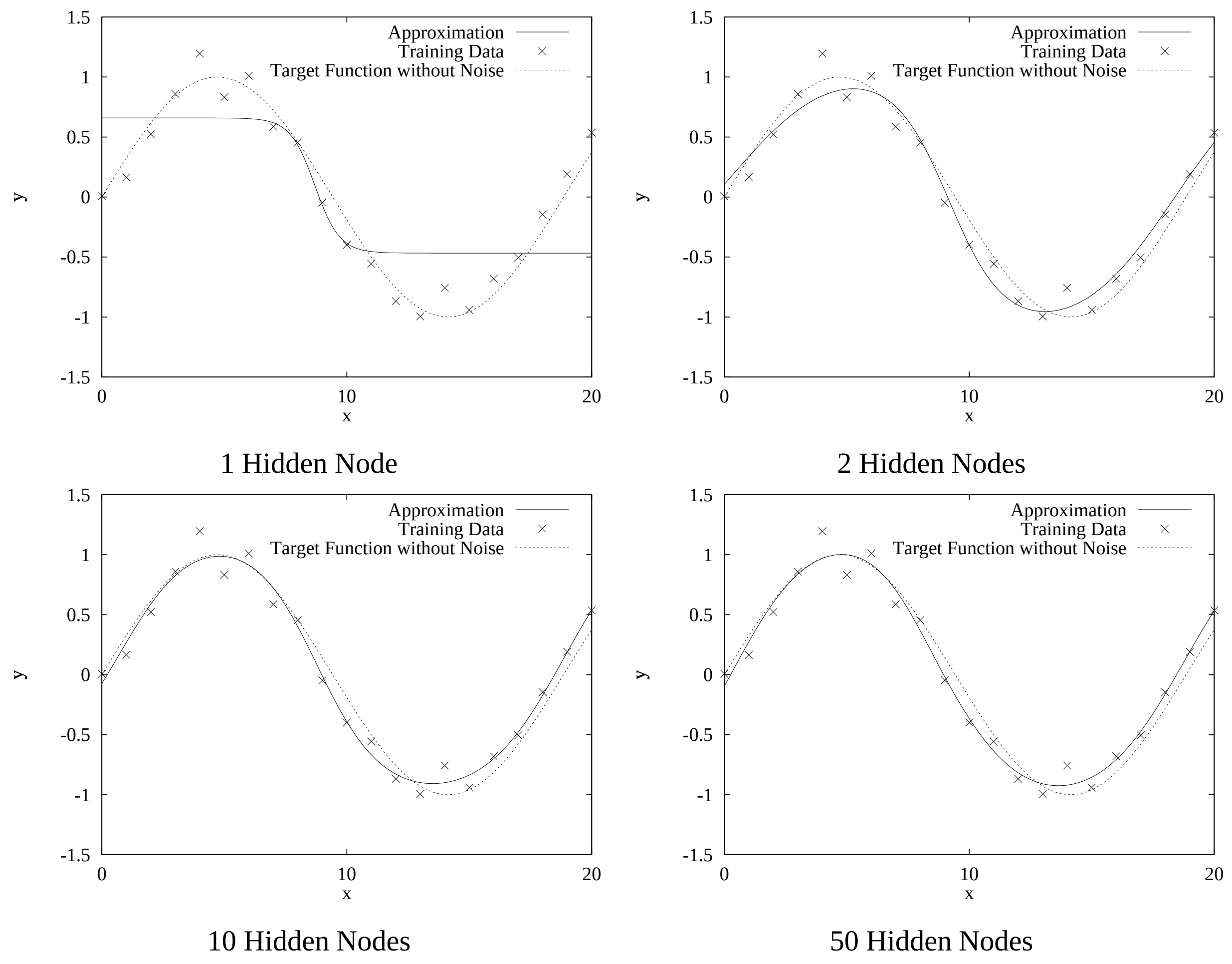

Figure 21. MLP interpolation of the function $y = \sin(x/3) + \nu$ in the range 0 to 20 as the number of hidden nodes is increased from 1 to 50. ν is a uniformly distributed random variable between -0.25 and 0.25. A large degree of overfitting can not be observed.

References

Abu-Mostafa, Y. (1989), 'The Vapnik-Chervonenkis dimension: Information versus complexity in learning', *Neural Computation* **1**(3), 312–317.

Akaike, H. (1970), 'Statistical predictor identification', *Annals of the Institute for Statistical Mathematics* **22**, 203–217.

Akaike, H. (1973), Information theory and an extension of the maximum likelihood principle, *in* B. N. Petrov and F. Csaki, eds, 'Proceeding 2nd International Symposium on Information Theory', Akademia Kiado, Budapest, pp. 267–281.

Akaike, H. (1974), 'A new look at the statistical model identification', *IEEE Transactions on Automatic Control* **19**, 716–723.

Amari, S. (1995), Learning and statistical inference, *in* M. A. Arbib, ed., 'The Handbook of Brain Theory and Neural Networks', MIT Press, Cambridge, Massachusetts, pp. 522–526.

Back, A. (1992), New Techniques for Nonlinear System Identification: A Rapprochement Between Neural Networks and Linear Systems, PhD thesis, Department of Electrical Engineering, University of Queensland.

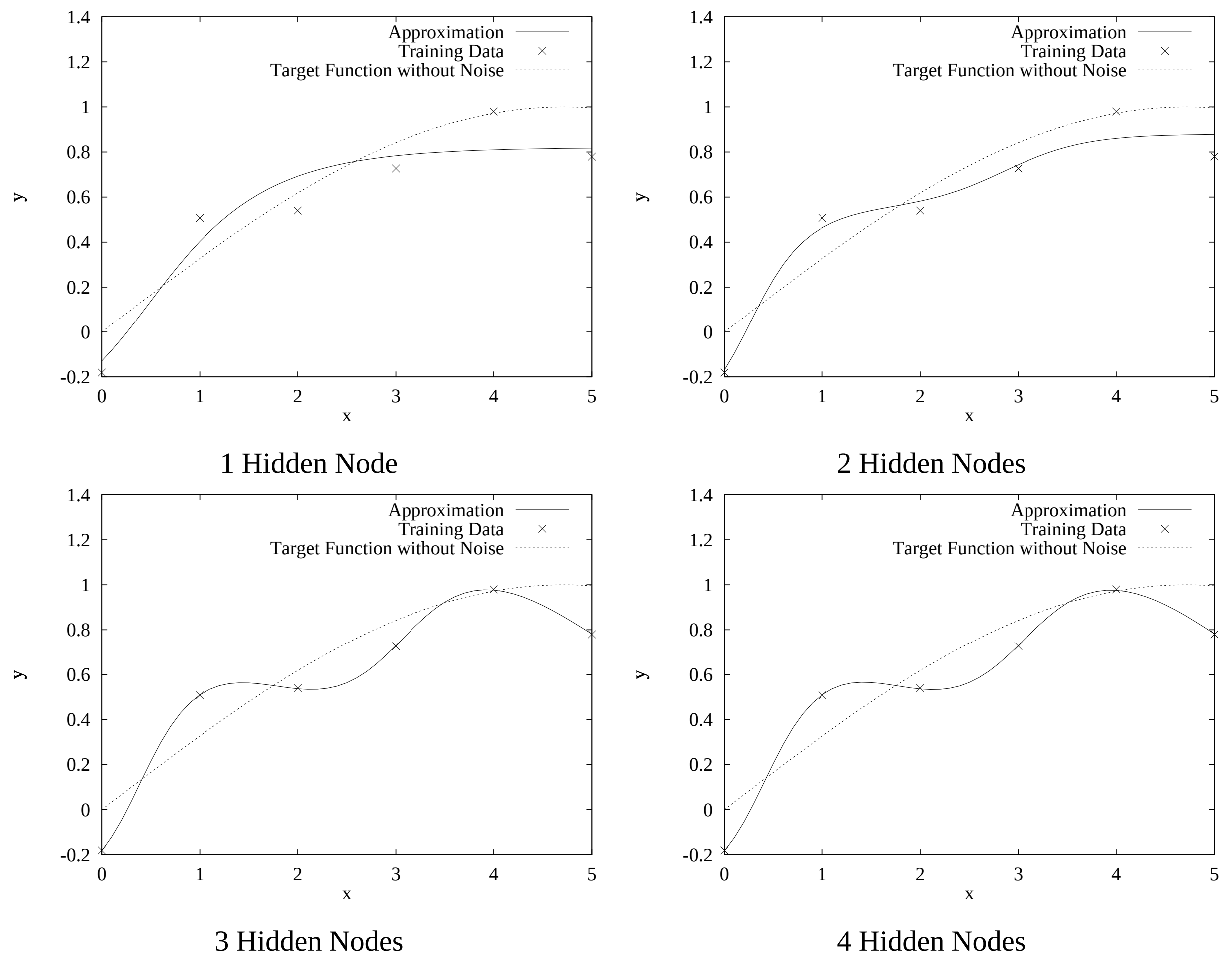

Figure 22. MLP interpolation of the function $y = \sin(x/3) + \nu$ in the range 0 to 5 as the number of hidden nodes is increased from 1 to 4. ν is a uniformly distributed random variable between -0.25 and 0.25. The network with two hidden nodes provides the best generalization performance – larger networks resulted in worse generalization due to overfitting.

Baldi, P. and Hornik, K. (1988), 'Neural networks and principal component analysis: Learning from examples without local minima', *Neural Networks* **2**(1), 53–58.

Barron, A. (1991), Complexity regularization with application to artificial neural networks, *in* G. Roussas, ed., 'Non-parametric Functional Estimation and Related Topics', Kluwer Academic Publishers, Dordrecht, The Netherlands, pp. 561–576.

Barron, A. (1992), Neural net approximation, *in* 'Proceedings of the Seventh Yale Workshop on Adaptive and Learning Systems', Yale University, New Haven, CT, pp. 69–72.

Bartlett, P. (1993), 'Vapnik-Chervonenkis dimension bounds for two-and three-layer networks', *Neural Computation* **5**(3), 371–373.

Bartlett, P. (1996), The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network, Technical report, Australian National University.

Baum, E. and Haussler, D. (1989), 'What size net gives valid generalization?', *Neural Computation* **1**(1), 151–160.

Bengio, Y., ed. (1996), *Neural Networks for Speech and Sequence Recognition*, Thomson.

Bourlard, H. and Morgan, N. (1994), *Connnectionist Speech Recognition: A Hybrid Approach*, Kluwer Academic Publishers, Boston, MA.

Breiman, L. (1994), 'Discussion of neural networks and related methods for classification', *Journal of the Royal Statistical Society B* **56**(3), 409–456.

Cardell, N. S., Joerding, W. and Li, Y. (1994), 'Why some feedforward networks cannot learn some polynomials', *Neural Computation* **6**(4), 761–766.

Caruana, R. (1993), *Generalization vs. Net Size*, Neural Information Processing Systems, Tutorial, Denver, CO.

Cohn, D. and Tesauro, G. (1992), 'How tight are the Vapnik-Chervonenkis bounds?', *Neural Computation* **4**(2), 249–269.

Crane, R., Fefferman, C., Markel, S. and Pearson, J. (1995), Characterizing neural network error surfaces with a sequential quadratic programming algorithm, *in* 'Machines That Learn', Snowbird.

Dadson, J. (1996), 'Article in comp.ai.neural-nets, message id: 4t19h0$ko8@sjx-ixn5.ix.netcom.com, 23 July'.

Drucker, H., Cortes, C., Jackel, L., Le Cun, Y. and Vapnik, V. (1994), 'Boosting and other ensemble methods', *Neural Computation* **6**, 1289–1301.

Friedman, J. (1995), 'Introduction to computational learning and statistical prediction', Tutorial Presented at Neural Information Processing Systems, Denver, CO.

Geman, S., Bienenstock, E. and Doursat, R. (1992), 'Neural networks and the bias/variance dilemma', *Neural Computation* **4**(1), 1–58.

Gori, M. (1996), 'An introduction to computational suspiciousness', Seminar presented at the University of Queensland, Brisbane, Australia.

Gori, M. and Tesi, A. (1992), 'On the problem of local minima in backpropagation', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14**(1), 76–86.

Hamey, L. (1995), Analysis of the error surface of the XOR network with two hidden nodes, Technical Report 95/167C, Macquarie University, Sydney, Australia.

Hassibi, B. and Stork, D. G. (1993), Second order derivatives for network pruning: Optimal Brain Surgeon, *in* C. L. Giles, S. J. Hanson and J. D. Cowan, eds, 'Advances in Neural Information Processing Systems', Vol. 5, Morgan Kaufmann, San Mateo, CA.

Hassoun, M., Cherkassky, V., Hanson, S., Oja, E., Sarle, W. and Sudjianto, A. (1996), Panel on neural networks and statistical models, *in* 'International Conference on Neural Networks, ICNN 96', IEEE, Piscataway, NJ, pp. 18–22.

Haykin, S. (1994), *Neural Networks, A Comprehensive Foundation*, Macmillan, New York, NY.

Hecht-Nielsen, R. (1990), *Neurocomputing*, Addison Wesley, New York.

Jacobs, R. (1995), 'Methods for combining experts' probability assessments', *Neural Computation* **7**, 867–888.

Kohonen, T. (1995), *Self-Organizing Maps*, Springer-Verlag, Berlin, Germany.

Kolmogorov, A. (1957), 'On the representation of continuous functions of several variables by superpositions of continuous functions of one variable and addition', *Dokl* **114**, 679–681.

Krogh, A. and Vedelsby, J. (1995), Neural network ensembles, cross validation, and active learning, *in* G. Tesauro, D. Touretzky and T. Leen, eds, 'Advances in Neural Information Processing Systems', Vol. 7, The MIT Press, pp. 231–238.

Kröse, B. and van der Smagt, P., eds (1993), *An Introduction to Neural Networks*, fifth edn, University of Amsterdam.

Kŭrková, V. (1991), 'Kolmogorov's theorem is relevant', *Neural Computation* **3**(4), 617–622.

Kŭrková, V. (1995), Kolmogorov's theorem, *in* M. A. Arbib, ed., 'The Handbook of Brain Theory and Neural Networks', MIT Press, Cambridge, Massachusetts, pp. 501–502.

Le Cun, Y. (1993), 'Efficient learning and second order methods', Tutorial presented at Neural Information Processing Systems 5.

Le Cun, Y., Denker, J. and Solla, S. (1990), Optimal Brain Damage, *in* D. Touretzky, ed., 'Advances in Neural Information Processing Systems', Vol. 2, (Denver 1989), Morgan Kaufmann, San Mateo, pp. 598–605.

Maass, W. (1995), Vapnik-Chervonenkis dimension of neural networks, *in* M. A. Arbib, ed., 'The Handbook of Brain Theory and Neural Networks', MIT Press, Cambridge, Massachusetts, pp. 522–526.

McInerny, J., Haines, K., Biafore, S. and Hecht-Nielsen, R. (1989), Back propagation error surfaces can have local minima, *in* 'International Joint Conference on Neural Networks', Vol. 2, (Washington 1989), IEEE, New York, p. 627.

Minsky, M. and Papert, S. (1988), *Perceptrons*, expanded version of the original 1969 edn, MIT Press, Cambridge, MA.

Moody, J. (1992), The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems, *in* J. Moody, S. J. Hanson and R. P. Lippmann, eds, 'Advances in Neural Information Processing Systems', Vol. 4, Morgan Kaufmann, San Mateo, CA, pp. 847–854.

Müller, K., Finke, M., Schulten, K., Murata, N. and Amari, S. (1996), 'A numerical study on learning curves in stochastic multi-layer feed-forward networks', *Neural Computation* p. in press.

Naftaly, U., Intrator, N. and Horn, D. (1995), Training single networks for optimal ensemble performance, *in* T. L. Gerald Tesauro, David Touretzky, ed., 'Advances in Neural Information Processing Systems 7', MIT Press, Cambridge, Massachusetts.

Perrone, M. P. and Cooper, L. N. (1993), When networks disagree: Ensemble method for neural networks, *in* R. J. Mammone, ed., 'Neural Networks for Speech and Image processing', Chapman-Hall.

Press, W., Teukolsky, S., Vetterling, W. and Flannery, B. (1992), *Numerical Recipes*, second edn, Cambridge University Press, Cambridge.

Renals, S., Morgan, N., Cohen, M. and Franco, H. (1992), Connectionist probability estimation is the Decipher speech recognition system, *in* 'Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing', pp. 601–604.

Ripley, B. (1995), 'Statistical ideas for selecting network architectures', Invited Presentation, Neural Information Processing Systems 8.

Rivlin, T. (1969), *An Introduction to the Approximation of Functions*, Blaisdell Publishing Company, Waltham, Massachusetts.

Robinson, A. (1994), 'An application of recurrent nets to phone probability estimation', *IEEE Transactions on Neural Networks* **5**(2), 298–305.

Saad, D. and Solla, S. (1995), 'Exact solution for on-line learning in multilayer neural networks', *Physical Review Letters* **74**, 4337–4340.

Saad, D. and Solla, S. (1996), Learning from corrupted examples in multilayer networks, Technical Report NCRG/96/019, Aston University, UK.

Sarle, W. (1996), 'Comp.ai.neural-nets frequently asked questions', `ftp://rtfm.mit.edu/pub/usenet/news.-answers/ai-faq/neural-nets`.

Sartori, M. A. and Antsaklis, P. (1991), 'A simple method to derive bounds on the size and to train multilayer neural networks', *IEEE Transactions on Neural Networks* **2**, 467–471.

Slomka, S. (1996), 'Personal communication'.

Vapnik, V. (1982), *Estimation of Dependencies Based on Empirical Data*, Springer-Verlag, Berlin.

Vapnik, V. (1995), *The Nature of Statistical Learning Theory*, Springer.

Werbos, P. (1974), Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, PhD thesis, Harvard University.

Wolpert, D. H. and Macready, W. G. (1995), No free lunch theorems for search, Technical Report SFI-TR-95-02-010, The Santa Fe Institute.

Yu, X.-H. (1992), ‘Can backpropagation error surface not have local minima’, *IEEE Transactions on Neural Networks* **3**, 1019–1021.