Overfitting and neural networks: Conjugate gradient and backpropagation

Conference Paper · February 2000

DOI: 10.1109/IJCNN.2000.857823 · Source: IEEE Xplore

2 authors, including:

Steve Lawrence
Google Inc.
154 PUBLICATIONS **18,013** CITATIONS

Steve Lawrence and C. Lee Giles. *Overfitting and Neural Networks: Conjugate Gradient and Backpropagation*, International Joint Conference on Neural Networks, Como, Italy, July 24–27, IEEE Computer Society, Los Alamitos, CA, pp. 114–119, 2000.

Overfitting and Neural Networks: Conjugate Gradient and Backpropagation

Steve Lawrence and C. Lee Giles

NEC Research Institute, 4 Independence Way, Princeton, NJ 08540

{lawrence,giles}@research.nj.nec.com

Abstract

Methods for controlling the bias/variance tradeoff typically assume that overfitting or overtraining is a global phenomenon. For multi-layer perceptron (MLP) neural networks, global parameters such as the training time (e.g. based on validation tests), network size, or the amount of weight decay are commonly used to control the bias/variance tradeoff. However, the degree of overfitting can vary significantly throughout the input space of the model. We show that overselection of the degrees of freedom for an MLP trained with backpropagation can improve the approximation in regions of underfitting, while not significantly overfitting in other regions. This can be a significant advantage over other models. Furthermore, we show that "better" learning algorithms such as conjugate gradient can in fact lead to worse generalization, because they can be more prone to creating varying degrees of overfitting in different regions of the input space. While experimental results cannot cover all practical situations, our results do help to explain common behavior that does not agree with theoretical expectations. Our results suggest one important reason for the relative success of MLPs, bring into question common beliefs about neural network training regarding training algorithms, overfitting, and optimal network size, suggest alternate guidelines for practical use (in terms of the training algorithm and network size selection), and help to direct future work (e.g. regarding the importance of the MLP/BP training bias, the possibility of worse performance for "better" training algorithms, local "smoothness" criteria, and further investigation of localized overfitting).

1 Introduction

One of the most important aspects of machine learning models is how well the model generalizes to unseen data. Multi-layer perceptron (MLP) neural networks have been very successful in many problems, often providing improved generalization over competing machine learning methods. Much has been written about controlling the bias/variance tradeoff (which affects generalization) in neural network and other machine learning models [2, 10, 5, 8, 6, 11]. This paper focuses on the bias/variance tradeoff and overfitting with respect to the training algorithm and the degrees of freedom in the model.

2 Overfitting

This section provides a brief overview of generalization and overfitting. The goal of MLP training can be formulated as minimization of a cost function [9, 3]: $E_{true} = \int_{\mathbf{x},\mathbf{d}} e(f(\mathbf{x},\mathbf{w}),\mathbf{d})p(\mathbf{x},\mathbf{d})\ d\mathbf{x}d\mathbf{d}$ 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 optimize the parameters $\mathbf{w}$ such that E_{true} is minimized: $\hat{\mathbf{w}} = \text{argmin}_{\mathbf{w}} \int_{\mathbf{x},\mathbf{d}} e(f(\mathbf{x},\mathbf{w}),\mathbf{d})p(\mathbf{x},\mathbf{d})\ d\mathbf{x}d\mathbf{d}$. E_{true} is the generalization error, 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 minimized which is called the *empirical error* or training error [3]: $E = \sum_{p=1}^{N_p} e(\mathbf{x}_p,\mathbf{d}_p)$. A very important question is how well a model trained to minimize E generalizes (i.e. how low E_{true} is). This is important because low E (performance on the training set) does not necessarily result in low E_{true} (expected performance on new patterns).

MLPs and other machine learning models are susceptible to "overfitting". Figure 1 illustrates the concept using polynomial approximation. A training dataset was created by evaluating $y = \sin(x/3) + \nu$ at $0, 1, 2, \ldots, 20$ where ν is a uniformly distributed random variable between -0.25 and 0.25. This dataset was then used to fit polynomial models with orders between 2 and 20. For order 2, the approximation is poor. For order 10, the approximation is reasonably good. However, as the order (and number of parameters) increases, significant overfitting and increasingly poor generalization is evident. At order 20, the approximated function fits the training data very well, however the

interpolation between training points is very poor. Overfitting can also be a very important problem in MLPs, and much work has been devoted to preventing overfitting with techniques such as model selection, early stopping, weight decay, and pruning [10, 5, 6, 7].

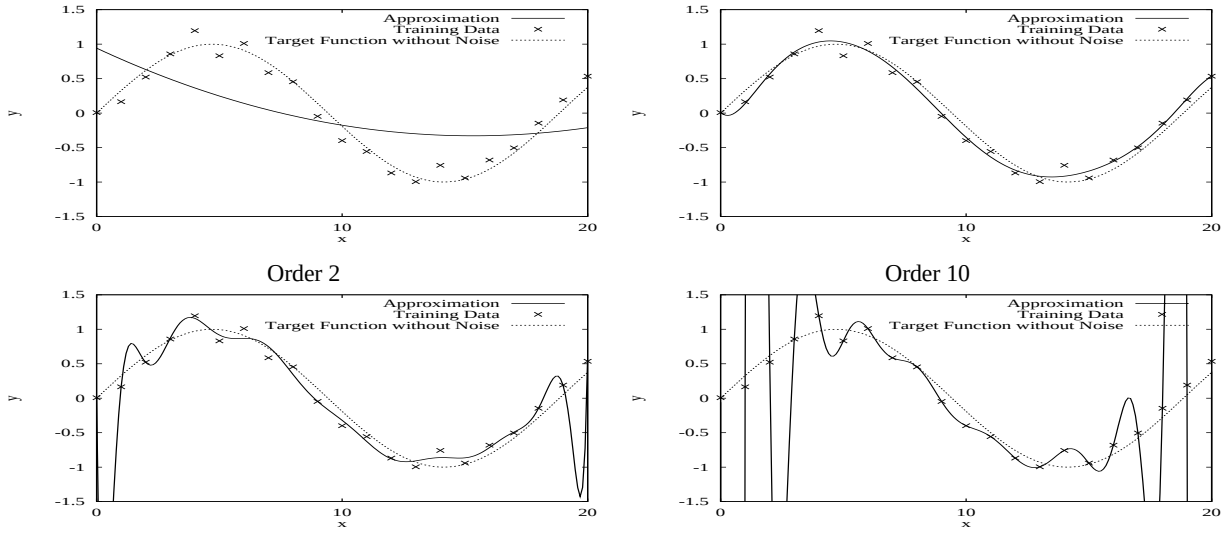


Figure 1. Polynomial interpolation of data from $y = \sin(x/3) + \nu$. Significant overfitting can be seen for orders 16 and 20.

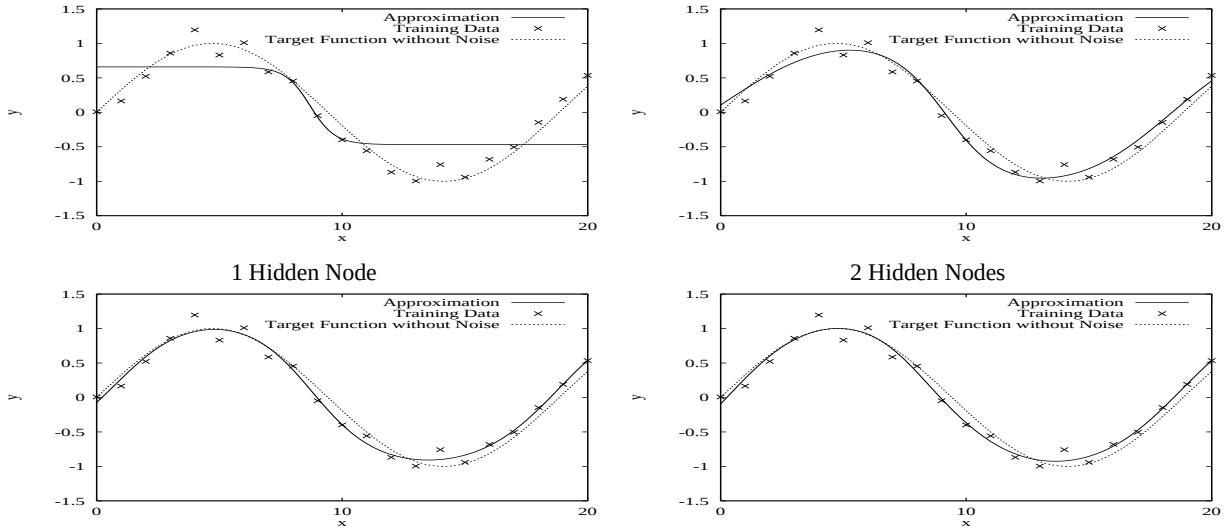
Figure 2 shows the results of using an MLP to approximate the same training set as used in the polynomial approximation example. The training details were as follows: a single hidden layer MLP, backpropagation, and 100,000 stochastic training updates were used. The learning rate was reduced linearly to zero from an initial learning rate of 0.5 (reducing the learning rate to zero is required for convergence and we have found that the linear reduction performs similarly to other learning rate schedules [4]). As for the polynomial case, the smallest network with one hidden unit (4 weights including the 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. Other experiments that we have performed indicate that this possibility is not limited to very simple examples, as used here for ease of visualization. Figure 3 shows a different example where significant overfitting can be seen in larger MLP models. The same equation was used but 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. For the first example, the largest network produced the best generalization error (hidden layer sizes (1, 2, 10, 50) corresponded to test errors (0.257, 0.0343, 0.0222, **0.0201**). For the second example, the 2 hidden node networks were best (hidden layer sizes (1, 2, 3, 4) corresponded to test errors (0.0943, **0.0761**, 0.103, 0.103). Results are averaged over ten trials.

3 Local Overfitting

Techniques such as model selection and early stopping typically assume that overfitting is a global phenomenon. However, overfitting can vary significantly throughout the input space of the model. Figure 4 shows the results of polynomial interpolation for data generated from the following equation:

$$y = \begin{cases} -\cos(x) + \nu & 0 \leq x < \pi \\ \cos(3(x - \pi)) + \nu & \pi \leq x \leq 2\pi \end{cases} \quad (1)$$

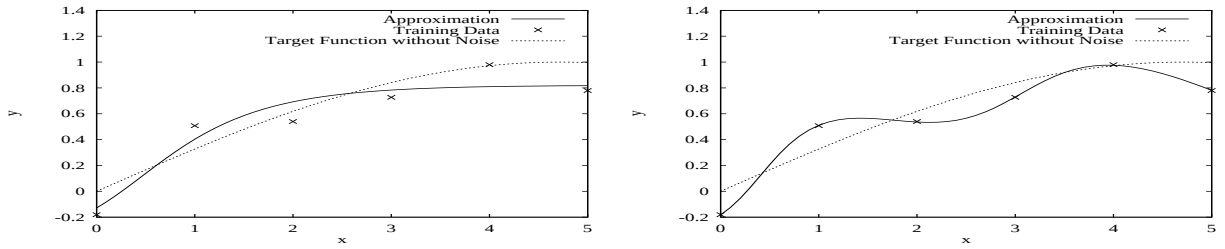
ν is a uniformly distributed random variable between -0.25 and 0.25. Five equally spaced points were generated in the first region and 15 equally spaced points were generated in the second region, i.e. there are two regions which have different underlying functions and different data densities. Observe that the overfitting behavior is different in the two regions of the data. The order 6 model provides a reasonable approximation of the left hand region and



1 Hidden Node | 2 Hidden Nodes

10 Hidden Nodes | 50 Hidden Nodes

Figure 2. MLP interpolation of data from $y = \sin(x/3) + \nu, 0 \leq x \leq 20$. A large degree of overfitting can not be observed.



1 Hidden Node | 4 Hidden Nodes

Figure 3. MLP interpolation of data from $y = \sin(x/3) + \nu, 0 \leq x \leq 5$. Only one and four hidden node networks are shown. Two hidden nodes provided the best generalization performance – larger networks resulted in worse generalization due to overfitting.

larger models show significant overfitting in this region. However, for the right hand region the order 6 model shows significant underfitting and the order 10 model is much better. No model performs well on both regions with respect to overfitting. Figure 5 shows the results of MLP interpolation (20,000 batch updates, learning rate linearly reduced to zero starting at 0.5) for the same data. We observe that small networks result in underfitting. However, larger networks fit the entire function well *without resulting in significant overfitting in the left hand region.*

The behavior seen in the MLP networks is due to the training algorithm resulting in sub-optimal solutions and it is therefore of interest to investigate the behavior with other training algorithms. Of the (pseudo-)second order training algorithms, conjugate gradient (CG) appears to have been the most successful and popular. Figure 6 shows the results of the same problem when using CG training instead of BP. CG typically results in lower training error for this problem. However, we observe that significant overfitting can occur. While this may be expected for the networks with excess degrees of freedom, we note that even networks with only 4 hidden nodes can result in significant overfitting. Also note that the degree of overfitting often varies significantly throughout the input space.

The behavior shown in figures 5 and 6 varies from trial to trial and only typical results are depicted. Figure 7 shows the results of 10 simulations for each case. The BP trained networks would be preferred for this problem, because even if a CG trained network of the appropriate size is used, the network is still prone to overfitting on a percentage of trials as shown in figure 6. Note that cross-validation could be used (and is recommended) to select the optimal size of the networks, however individual trials with CG typically still result in more overfitting (the computational expense of cross-validation can also be significant, especially with sparse data).

4 Discussion

While experimental results cannot cover all practical situations, our results do help to explain common behavior [1] which does not agree with theoretical expectations. Our results[1] indicate that the solution found by BP is often

[1] Including the simple examples in this paper and a visualization technique (which is not shown here) for viewing more complex approximations.

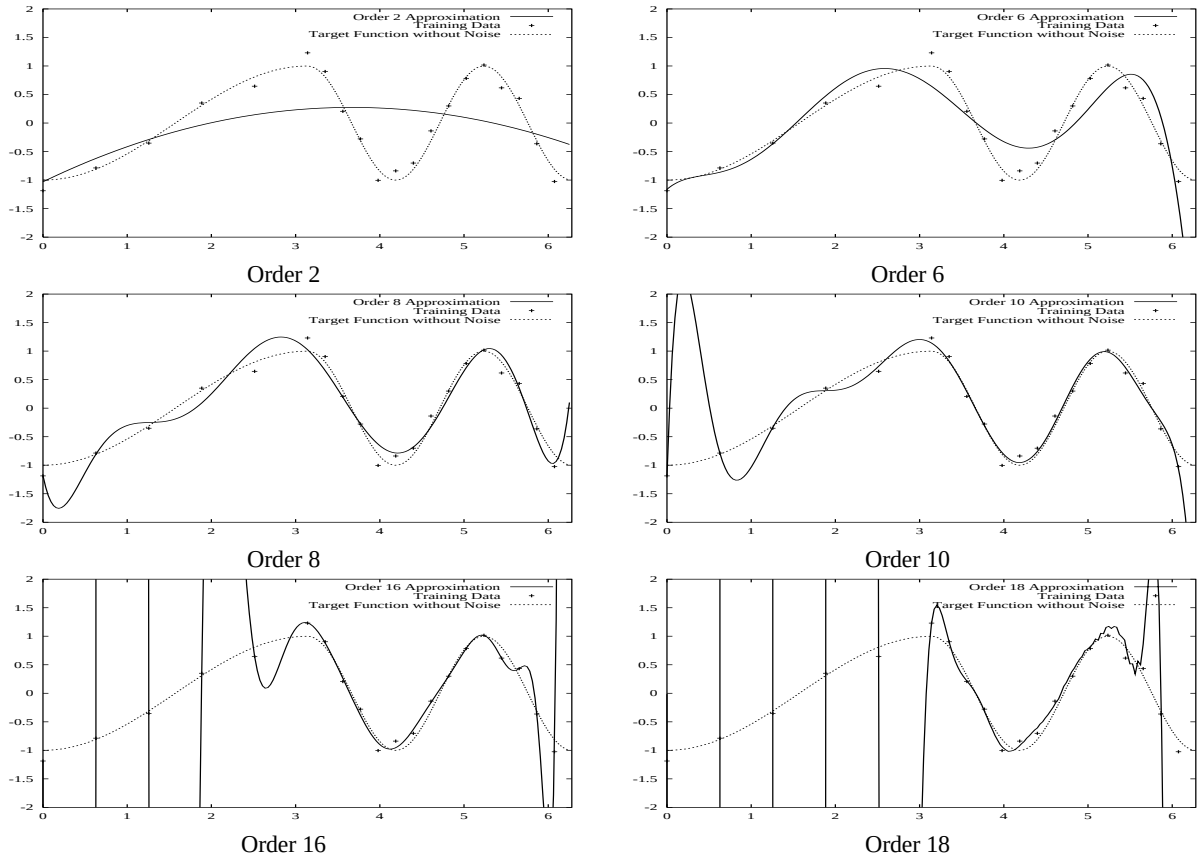


Figure 4. Polynomial interpolation of data from equation 1 as the order of the model is increased from 2 to 18. Overfitting behavior differs in the left and right hand regions.

biased towards a "smoother" solution. Considering the BP training algorithm, this is perhaps not surprising – a more "complex" approximation requires larger weights (in general), and BP can have difficulty finding networks with many large weights due to shallow gradients and local minima. The results also suggest that backpropagation can result in the underutilization of network resources in certain cases (i.e. some parameters may be ineffective or only partially effective due to sub-optimal convergence).

Theoretical work by [1] supports these results. Bartlett comments: *"the VC-bounds seem loose; neural networks often perform successfully with training sets that are considerably smaller than the number of weights"*. 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 suggests that networks with smaller weights will generalize better than similar networks with large weights. Investigation of the weights from BP and CG trained networks shows that BP training tends to result in smaller weights.

The fact that backpropagation can be biased towards smoother approximations and "fail" to find better minima corresponding to more complex approximations can thus be seen to have two implications:

1. Overselection of the degrees of freedom in an MLP trained with BP may be significantly less damaging than overselection for an MLP trained with CG (or other models such as polynomial approximation) [2].
2. Considering the case where the function being approximated has multiple regions with qualitatively different underlying functions and/or different data densities: the relative insensitivity of BP-trained MLPs to overselection of the degrees of freedom may allow BP to fit different regions more accurately when CG-trained networks (or other models such as polynomial approximation) exhibit significantly different degrees of overfitting throughout the input space. We note that global regularization methods such as cross validation may not help algorithms like CG here. For example, networks may start to overfit in one region while still underfitting in another.

[2] This may help to explain why neural network textbooks typically use radial basis function networks or polynomial approximation, rather than a BP trained MLP, to illustrate overfitting.

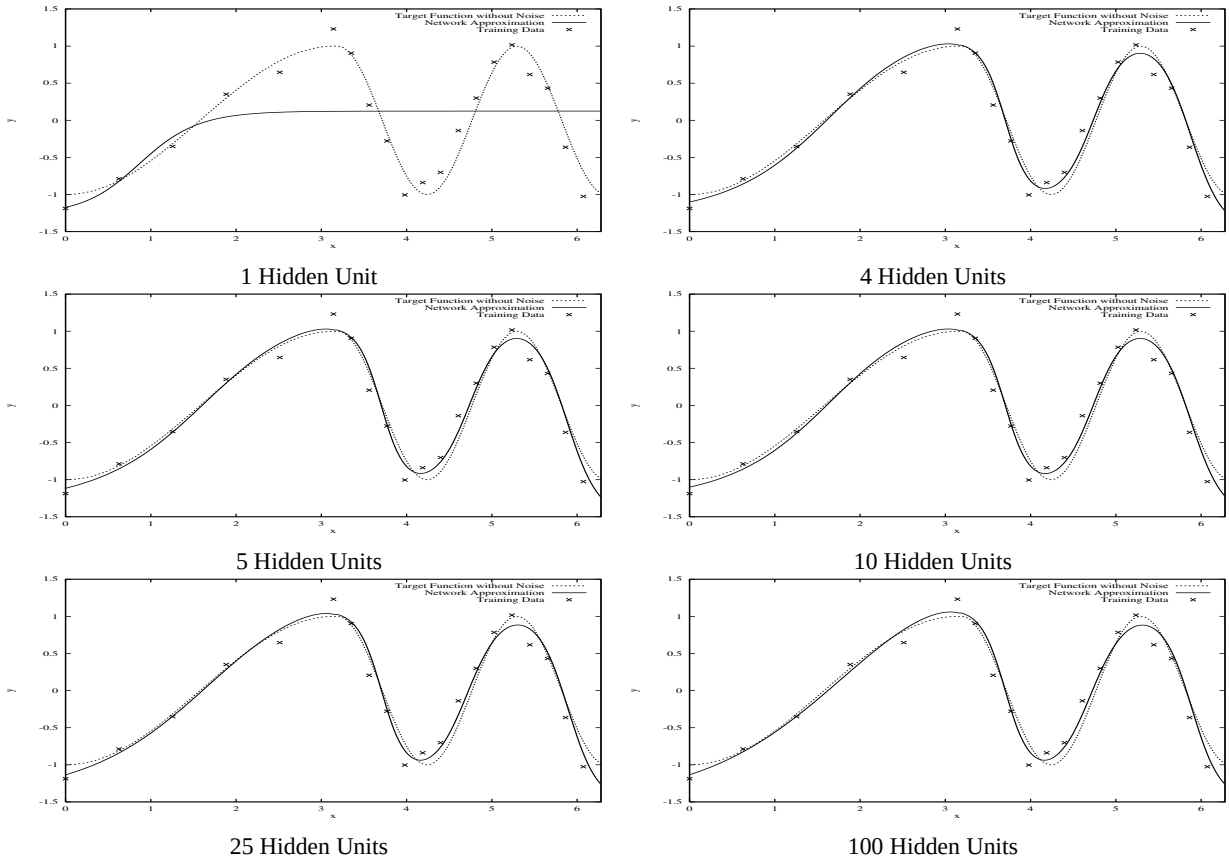


Figure 5. MLP interpolation using BP training of data from equation 1 as the number of hidden units is increased. No significant overfitting is evident.

5 Future Work

Using a smaller problem that aids visualization, our results help explain a phenomenon that is difficult to analyze with larger problems. These results help to direct future work, e.g. a) regarding the importance of the possibility of MLP/BP training being biased towards "smoother" approximations, and how this affects the theory and practice of neural network training in practical situations, b) regarding the possibility of "better" training algorithms resulting in worse performance, especially in the case where the underlying function or data density varies throughout the input space, c) local "smoothness" criteria for the determination of varying function smoothness in different regions, and d) further investigation of localized overfitting and possibly methods for controlling the bias/variance tradeoff appropriately in different regions of the input space.

6 Summary

Overfitting is typically not a global phenomenon (although methods for controlling the bias/variance tradeoff often assume that it is), and can vary significantly throughout the input space of the model. In this case, overselection of the degrees of freedom for an MLP can, in certain cases, improve the approximation in regions of underfitting, while not significantly overfitting in other regions. This can be a significant advantage over other models. Furthermore, "better" learning algorithms such as conjugate gradient can, in certain cases, lead to worse generalization because they can be more prone to creating varying degrees of overfitting in different regions of the input space. In contrast, the mode of "failure" exhibited by backpropagation can in fact be beneficial and result in improved generalization over the "better" algorithms, in so much as the implicit bias created by the network structure and training algorithm matches the desired target function. This bias may account for part of the success that MLPs have encountered over competing methods.

References

[1] Peter L. Bartlett. For valid generalization the size of the weights is more important than the size of the network. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 134. The MIT Press, 1997.

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

[3] Y. Bengio. *Neural Networks for Speech and Sequence Recognition.* Thomson, 1996.

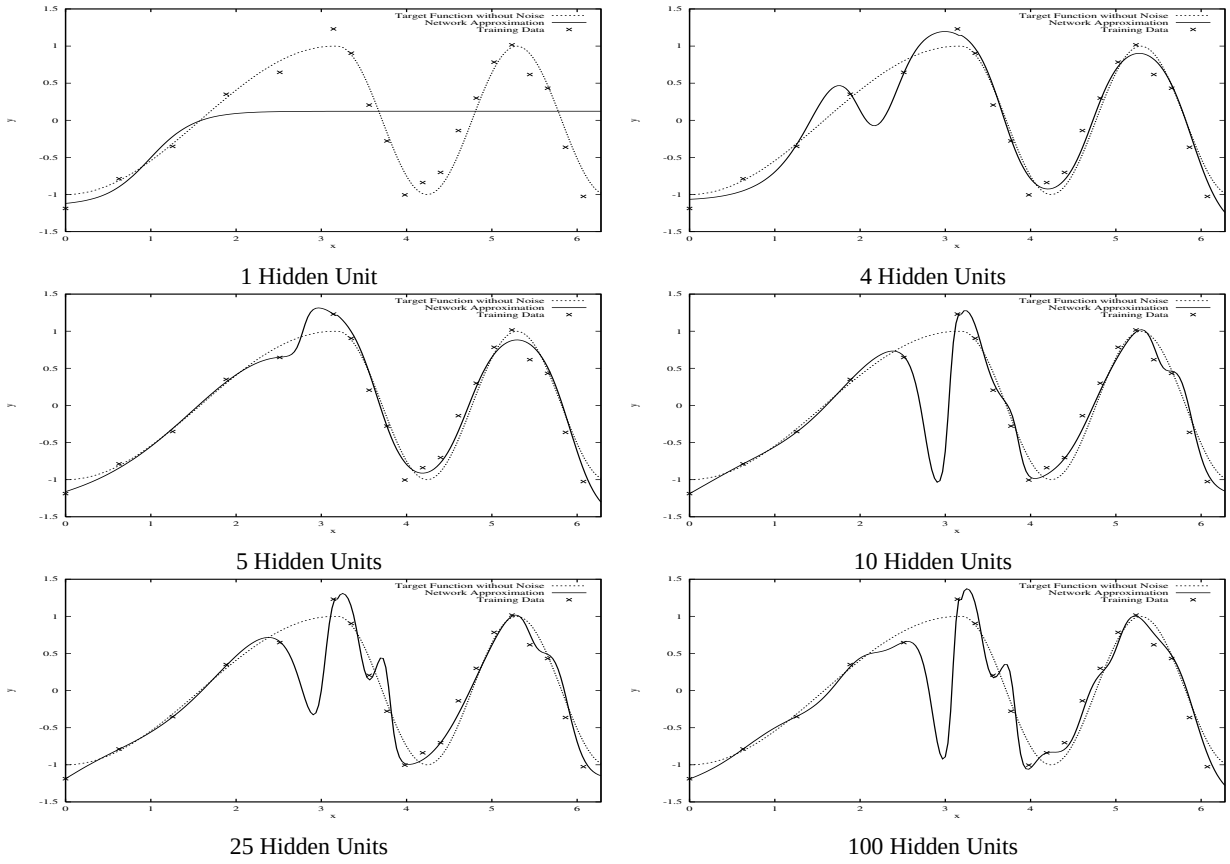


Figure 6. MLP interpolation using CG training of data from equation 1 as the number of hidden units is increased. Significant overfitting is evident, even for many models with only 4 hidden units.

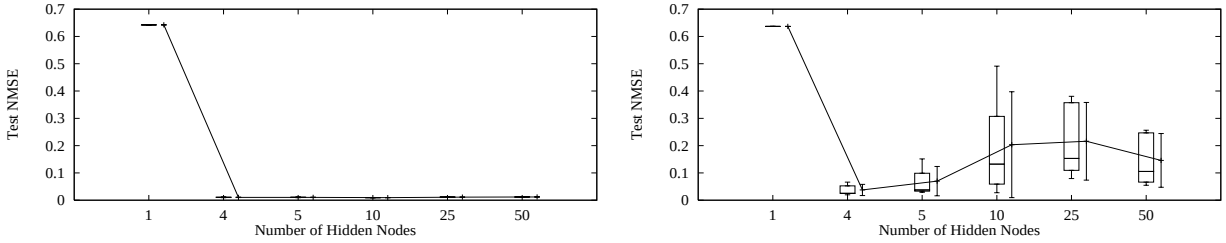


Figure 7. Test NMSE (normalized mean squared error) results for MLP interpolation of data from equation 1 using BP (left) and CG (right). Each set of simulations is shown with box-whiskers plots (on the left in each case) and the mean plus and minus one standard deviation (on the right in each case).

[4] C. Darken and J.E. Moody. Note on learning rate schedules for stochastic optimization. In R.P. Lippmann, J.E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 3, pages 832–838. Morgan Kaufmann, San Mateo, CA, 1991.

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

[6] A. Krogh and J.A. Hertz. A simple weight decay can improve generalization. In J.E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 950–957. Morgan Kaufmann, San Mateo, CA, 1992.

[7] Y. Le Cun, J.S. Denker, and S.A. Solla. Optimal Brain Damage. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 598–605, San Mateo, 1990. (Denver 1989), Morgan Kaufmann.

[8] J.E. Moody. The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. In J.E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 847–854. Morgan Kaufmann, San Mateo, CA, 1992.

[9] V.N. Vapnik. *Estimation of Dependencies Based on Empirical Data*. Springer-Verlag, Berlin, 1982.

[10] A.S. Weigend, D.E. Rumelhart, and B.A. Huberman. Generalization by weight-elimination with application to forecasting. In R. P. Lippmann, J.E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 3, pages 875–882. Morgan Kaufmann, San Mateo, CA, 1991.

[11] D. Wolpert. On bias plus variance. *Neural Computation*, 9(6):1211–1243, 1997.