Appears in Australian Conference on Neural Networks, ACNN 96, Edited by Peter Bartlett, Anthony Burkitt, and Robert Williamson, Australian National University, pp. 16–21, 1996.

Function Approximation with Neural Networks and Local Methods: Bias, Variance and Smoothness

Steve Lawrence[*], Ah Chung Tsoi, Andrew D. Back[†]
{lawrence,act,back}@elec.uq.edu.au

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

Abstract

We review the use of global and local methods for estimating a function mapping $\mathcal{R}^m \Rightarrow \mathcal{R}^n$ from samples of the function containing noise. The relationship between the methods is examined and an empirical comparison is performed using the multi-layer perceptron (MLP) global neural network model, the single nearest-neighbour model, a linear local approximation (LA) model, and the following commonly used datasets: the Mackey-Glass chaotic time series, the Sunspot time series, British English Vowel data, TIMIT speech phonemes, building energy prediction data, and the sonar dataset. We find that the simple local approximation models often outperform the MLP. No criterion such as classification/prediction, size of the training set, dimensionality of the training set, etc. can be used to distinguish whether the MLP or the local approximation method will be superior. However, we find that if we consider histograms of the k-NN density estimates for the training datasets then we can choose the best performing method *a priori* by selecting local approximation when the spread of the density histogram is large and choosing the MLP otherwise. This result correlates with the hypothesis that the global MLP model is less appropriate when the characteristics of the function to be approximated varies throughout the input space. We discuss the results, the smoothness assumption often made in function approximation, and the bias/variance dilemma.

1 Introduction

The problem of learning by example can be considered equivalent to a multivariate function approximation problem in many cases [22], ie. find a mapping $\mathcal{R}^m \Rightarrow \mathcal{R}^n$ given a set of example points. It is common and convenient to decompose the problem into n mappings to $\mathcal{R}^1$. We are most interested in the case where the data is high-dimensional and corrupted with noise, and when the function to be approximated is believed to be smooth in some sense. The smoothness assumption is required because the problem of function approximation (especially from sparse data) is ill-posed and must be constrained.

Function approximation methods fall into two broad categories: global and local. Global approximations can be made with many different function representations, eg. polynomials, rational approximation, and multi-layer perceptrons [8]. Often a single global model is inappropriate because it does not apply to the entire state space. To approximate a function f, a model must be able to represent its many possible variations. If f is complicated, there is no guarantee that any given representation will approximate f well. The dependence on representation can be reduced using local approximation where the domain of f is broken into local neighbourhoods and a separate model is used for each neighbourhood [8]. Different function representations can be used in both local and global models as shown in table 1.

Global models	Local models
Linear	None
Polynomial	Weighted average
Splines	Linear
Neural networks	Polynomial
…	Splines
	Neural networks
	…

Table 1: Global and local function approximation methods.

2 Neural Networks

It has been shown that an MLP neural network, with a single hidden layer, can approximate any given continuous function

[*] http://www.neci.nj.nec.com/homepages/lawrence

[†] This work has been partially supported by the Australian Telecommunications and Electronics Research Board (SL) and the Australian Research Council (ACT and ADB).

on any compact subset to any degree of accuracy, providing that a sufficient number of hidden layer neurons is used [5, 17]. However, in practice, the number of hidden layer neurons required may be impractically large. In addition, the training algorithms are “plagued” by the possible existence of many local minima or “flat spots” on the error surface. The networks suffer from “the curse of dimensionality”.

3 Local Approximation

Local approximation is based on nearest-neighbour techniques. An early use of nearest-neighbours was in the field of pattern classification. Fix and Hodges [9] classified new patterns by searching for a similar pattern in a stored set and using the classification of the retrieved pattern as the classification of the new one. Many papers thereafter suggested new rules for the classification of a point based on its nearest-neighbours (weighted averages, etc.). For function approximation, the threshold autoregressive model of [27] is of some interest. The model effectively splits the state space in half and uses a separate linear model for each half. The LA techniques considered here can be seen as an extension to this concept where the space is split into many parts and separate (non-)linear models are used in each part. LA techniques have a number of advantages:

• Functions which may be too complex for a given neural network to approximate globally may be approximated.

• Rapid incremental learning is possible without degradation in performance on previous data (necessary for online applications and models of biological learning).

• Rapid cross-validation testing is possible by simply excluding points in the training data and using them as test points.

However, LA techniques can exhibit poor generalisation, slow performance, and increased memory requirements:

• Slow performance. The most straightforward approach to finding nearest-neighbours is to compute the distance to each point which is an $O(N)$ solution. This can be reduced to $O(logN)$ by using a decision tree. The K-D tree is a popular decision tree introduced by Bentley [2], which is a generalisation of a binary tree to the case of k keys. Discrimination is still performed on the basis of a single key at each level in the tree, however, the key used at each level cycles through all available keys as the decision process steps from level to level. Bentley gives an algorithm for finding m nearest neighbours in k-dimensional space requiring $log_2 n$ nodes to be visited and approximately $m2^k$ distance calculations [12]. The K-D tree is known to scale poorly in high dimensions - significant improvements can be found with approximate nearest neighbour techniques [1].

• Memory requirements. This problem can be partially addressed by removing unneccesary training data from regions with little uncertainty.

Determining the optimal number of neighbours to use is difficult because the answer usually depends on the location in the input space. Some possibilities include: a) using a fixed number of neighbours, b) using as many neighbours as can be found within a fixed radius, and c) clustering the data and using a number of neighbours equal to the number of points in each cluster. In general, the approaches vary from the simplest case which ignores the variation of the function throughout space, to more complex algorithms which attempt to select a number of neighbours appropriate to the interpolation scheme and the local properties of the function. This is not simple - using a small number of neighbours increases the variance of the results under the presence of noise. Increasing the number of neighbours can compromise the local validity of a model (eg. approximating a curved manifold with a linear plane) and increase the bias of results. This is the classic bias/variance dilemma [25] which a designer often faces.

Algorithms such as: classification and regression trees (CART) [3], multivariate adaptive regression splines (MARS) [11], ID3 [23], and the hierarchical mixtures of experts (HME) algorithm of Jordan and Jacobs [18], are local approximation models where the input space is divided, at training time, into a hierarchy of regions where simple surfaces are fit to the local data.

3.1 Interpolation

The type of local model used controls the method of interpolation between the neighbours. Typically, only simple models have been used. We list some possibilities, in order of complexity.

1. No interpolation - the test output is equal to the closest training point output.

2. Weighted average - the output for the test point is equal to the average of the output of the k nearest-neighbours. This average is usually weighted inversely by the distance from the test point [7]. Weighted averages have been used extensively (eg. [26]) and analysed extensively (eg. [15]).

3. Linear models - the group of nearest-neighbours is used to create a local linear model. This model is then used to find the desired value for a new point. One example, which we use later, is given by Casdagli [4]:

$$\mathbf{y} = \begin{bmatrix} \mathbf{1} & \mathbf{x_1} & \dots & \mathbf{x_k} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_k \end{bmatrix} \quad (1)$$

where α_i, $i = 0, 1, \ldots$ are constants, k is the number of neighbours, and n is the dimension of the input vector $\mathbf{x_i}$, $i = 1, 2, \ldots, k$. The parameters α_i, $i = 0, 1, \ldots, k$ are found by using ordinary least squares.

Weighted regressions to fit local polynomial models have been used by many people for classification (eg. [20]). Recently, they have also become popular for approximation. Farmer and Sidorowich [8] have used local polynomial models for forecasting but have only had success using low order models.

4. Non-linear models. One example is splines. Mhaskar [21] has used tensor product b-splines for local approximation. Standard results in spline approximation theory can be used. Any non-linear model is a candidate for local approximation models and hence we may even use multi-layer perceptrons.

4 Simulations

In order to investigate the relation between the methods in practice, we have performed simulations with the following datasets:

1. The Mackay-Glass equation. The Mackey-Glass equation is a time delay differential equation first proposed as a model of white blood cell production [19]: $\frac{dx}{dt} = \frac{ax(t-\tau)}{[1+x^c(t-\tau)]} - bx(t)$ where the constants are commonly chosen as $a = 0.2$, $b = 0.1$ and $c = 10$. The delay parameter τ determines the behaviour of the system. For $\tau > 16.8$ the system produces a chaotic attractor. We have used $\tau = 30$. Our dataset consisted of 3000 training patterns and 500 test patterns.

2. Vowel data. Speaker independent recognition of the eleven steady state vowels of British English using a specified training set of 10 LPC derived log area ratios [24]. There are 528 training patterns and 462 test patterns.

3. Sunspot data. Sunspots are dark blotches on the sun and yearly averages have been recorded since 1700 [10]. In this example, 220 samples are used for training, and the remaining 60 for testing.

4. Sonar data. Discrimination between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock, as used in [16]. 103 patterns have been used in both the training and test sets.

5. Speech data. This dataset consists of the phoneme "aa" extracted from the TIMIT database and arranged as a number of sequences: prior phoneme(s), current phoneme, next phoneme(s). Raw speech data was pre-processed into a sequence of RASTA-PLP frames[1]. The analysis window (frame) was 20ms, the window step size was 10ms, i.e., the analysis window overlaps the succeeding frame by half a frame, and the order of PLP used was 9 - creating 10 feature values per frame (the power of the segment of signal in the frame together with 9 RASTA PLP coefficients). The phonemes were extracted from speakers coming from the same demographic region. Multiple speakers were used and the speakers used in the test set were not contained in the training set. The training set contained 2000 frames, where each phoneme is roughly 10 frames. The test set contained 2000 frames, and an additional validation set containing 2000 frames was used to control generalisation.

6. Building energy data. This dataset was problem A of the "Great energy predictor shootout" contest[2]. There are 14 inputs (time, temperature, humidity, etc.), 3 outputs (energy predictions), 2104 training points, 1052 validation points, and 1052 test points.

Simulations have been performed using a single nearest-neighbour model, the local linear approximation model given earlier with a range of neighbourhood sizes, and an MLP[3] using a range of hidden node numbers. For the multi-layer perceptron we used the $tanh$ activation function, and a search then converge learning rate schedule [6]: $\eta = \frac{\eta_0}{\frac{n}{N/2} + \frac{c_1}{max\left(1, (c_1 - \frac{max(0, c_1(n - c_2 N))}{(1-c_2)N}\right)}}$ where η = learning rate, η_0 = initial learning rate = 0.1, N = total training epochs, n = current training epoch, $c_1 = 50$, $c_2 = 0.65$. Target outputs were scaled by 0.8. We used stochastic update (ie. the parameters are updated after every pattern presentation as opposed to batch update where the parameters are only updated once per complete pass through the training set). Weights were initialised from a set of random candidates based on training set performance. An input window of 6 time steps was used for these problems: Mackey-Glass, Sunspots, Speech. Four simulations using different random seeds were performed for each MLP trial.

5 Results

Table 2 and figure 1 show the results of our simulations. We have shown the classification error for the classification problems and the NMSE for the remainder. Comparing the LA methods to the MLP, we observe that each method performs best on three out of the six problems and there are significant differences between the results obtained with each method. The LA methods perform best on the Sonar, Mackey-Glass, and

[1]RASTA-PLP is a technique of feature extraction which incorporates perceptual characteristics.

[2]"The Great Energy Predictor Shootout" - The First Building Data Analysis And Prediction Competition; ASHRAE Meeting; Denver, Colorado; June, 1993.

[3]$y_k^l = f\left(\sum_{i=0}^{N_{l-1}} w_{ki}^l y_i^{l-1}\right)$ 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 commonly a sigmoid function.

Sunspots problems while the MLP performs best on the Vowel, Building, and Speech problems.

It appears to be difficult to make concise conclusions about the results. If we look at the size of the datasets, the dimensionality of the datasets, the degree of noise, etc. we do not find a common factor with which we could have *a priori* selected either the MLP or the LA methods and found the same best solution. However, it is known that MLPs do not respond well to isolated data points [28] (meaning points in the input space where the density of the training data is very low compared to other areas), hence we may expect the global MLP methods to be less appropriate if the training data has a lot of points where the density of the data is low. Furthermore, it is harder for a global MLP model to adapt to differing density throughout the input space than it is for an LA model. Figure 2 shows k-NN density estimates[4] [13] of the training datasets for $k = 3$. The estimates have been normalised so the median (a statistic insensitive to outliers) estimate is 1. The graphs for those datasets where the best performing model is based on the LA approach are characterised by a greater spread of the the individual density estimates (ie. the density of the data varies more as we move around the input space). The vowel dataset does not appear to follow this rule as clearly. However, if we rank the spread of the datasets according to the interquartile range[5] of the density estimates we obtain: Building 0.42, Speech 0.82, Vowel 1.1, Sonar 1.3, Sunspots 1.4, Mackey-Glass 1.8. The best performing method for the models in this order is: MLP, MLP, MLP, LA, LA, LA. Hence, for the datasets we have used here, the spread of the density estimates can be used to distinguish whether the MLP or LA method will be superior. We note that we expect a general criterion for selecting the best method *a priori* to also include characteristics such as: dimensionality and size of the training set, nature of the function which must be approximated, amount of noise, etc.

For the vowel dataset, we note that the difference between the MLP and LA methods is small compared to the other datasets. We also note that the size of the local neighorhood and the number of hidden nodes in the best models is correlated with the size of the training data sets.

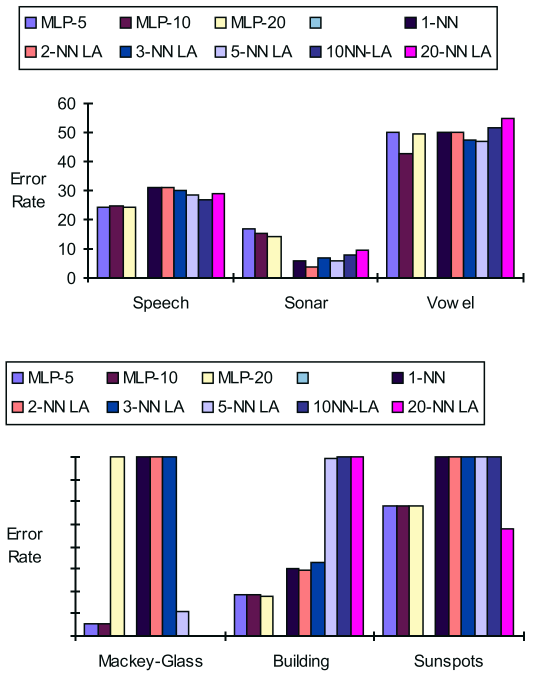

Figure 1: Results. The fourth column separates the MLP and LA models and is empty. The error rates for the 10-NN LA model and the 20-NN LA model on the Mackey-Glass dataset are too small to distinguish. The data has been scaled so that all results could be displayed on the same graph (see table 2 for the numerical values).

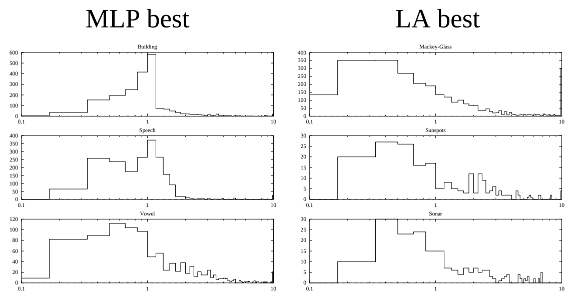

Figure 2: Density estimates. The x-axis represents the density estimates, has a log scale from 0.1 to 1 (center) to 10, and is divided into a number of sections. Density estimates are normalised so the median estimate is 1. The y-axis represents the number of points which fall into each density section. From top to bottom, the left column shows the estimates for the Building, Speech and Vowel datasets and the right column shows the Mackey-Glass, Sunspots and Sonar datasets. An MLP model is the best performing model for the datasets on the left and an LA model is the best performing model for the datasets on the right.

6 Discussion and Open Issues

The bias/variance dilemma, which has been well covered in the literature (eg. [14]), can help explain the varying results of the different methods. Dividing data into regions can decrease the bias of an estimator, but generally increases the variance (consider linear regression where the most peripheral points in the

[4]Computation is based on finding the smallest volume which contains k neighbours for each point.

[5]We use the interquartile range because the standard deviation is good for reasonably symmetrical distributions without outliers.

Test NMSE	1-NN	2-NN LA	3-NN LA	5-NN LA	10NN-LA	20NN-LA
Mackey-Glass	1.0×10^{-2}	6.7×10^{-3}	5.4×10^{-3}	5.3×10^{-4}	8.1×10^{-6}	1.9×10^{-6}
Building	1.52	1.46	1.64	3.98	5.1	14.1
Sunspots	0.54	0.49	0.42	0.81	0.42	0.24

Test NMSE	MLP-5		MLP-10		MLP-20	
Mackey-Glass	2.9×10^{-4}	5.9×10^{-5}	2.7×10^{-4}	4.7×10^{-5}	2.4×10^{-2}	3.9×10^{-5}
Building	0.91	0.022	0.91	0.012	0.90	0.025
Sunspots	0.29	0.0082	0.29	0.018	0.29	0.022

Test Error %	1-NN	2-NN LA	3-NN LA	5-NN LA	10NN-LA	20NN-LA
Speech	31.1	31	29.9	28.4	26.7	28.9
Sonar	5.83	3.88	6.8	5.83	7.77	9.71
Vowel	49.8	50	47.2	47	51.7	55

Test Error %	MLP-5		MLP-10		MLP-20	
Speech	24.1	1.2	24.5	0.68	24	1.9
Sonar	17.1	2.97	15.2	1.48	14.2	1.48
Vowel	49.9	3.3	42.5	3.4	49.4	1.51

Summary	Input dimension	Training points	Noise?	Best model
Sonar	60	103	yes	2-NN
Mackey-Glass	6 (input window)	3000	no	20-NN
Sunspots	6	220	yes	20-NN
Vowel	10	528	yes	10-MLP
Building	14	2104	yes	20-MLP
Speech	6x10 = 60	2000	yes	20-MLP

Table 2: Results of local approximation versus global MLP approximation for six commonly used datasets. 1-NN is single nearest neighbour approximation, k-NN LA is Casdagli's local approximation using k neighbours, MLP-n denotes the multi-layer perceptron with n hidden nodes. Each MLP result is followed by the standard deviation.

input space are the most important points for decreasing the variance). Local approximation algorithms generally tend to be variance increasing algorithms which is particularly problematic in high-dimensional spaces where data becomes exceedingly sparse [18]. The bias/variance tradeoff can be controlled in the multi-layer perceptron using various methods including controlling the number of hidden nodes (and hence the representational power of the network), and weight elimination. In the k-NN local approximation algorithm used here, bias increases and variance decreases as k increases, due to the simple local model used. We can use cross-validation to set these controls over the bias and variance, with different difficulties in each case. For the local approximation case, we can perform leave-one-out validation as discussed earlier, but better performance can often result from varying the appropriate control throughout the input space, and estimation becomes difficult as the sample size decreases. For the MLP, it is generally impractical to perform leave-one-out validation, leading to the common practice of using a separate generalisation set, which means withholding potentially useful information during model construction. Additionally, the effects of local minima may be important.

It is well known that appropriate pre-processing of data can produce a significant difference in performance. For both neural network and local approximation solutions, better accuracy will be obtained when the function that must be approximated is smooth in some sense. For neural networks, and all local models involving curve fitting, smoothness is an assumption usually made by researchers. Hence, it is important to know whether the function we are trying to approximate is smooth. A smoothness criterion can be formulated for a differentiable function, eg. $J(f) = \int_0^1 (f^{(m)}(t))^2 dt$ for one dimension, or $J_m^d(f) = \sum_{\alpha_1+\ldots+\alpha_d=m} \frac{m!}{\alpha_1!\ldots\alpha_d!} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \left(\frac{\partial^m f}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}} \right)^2 \prod_j dx_j$ for multiple dimensions. However, we generally do not have a differentiable function because this is what we are trying to estimate. Smoothness criteria for a dataset exist - eg. the roughness and outlier indexes of Watson [29]. However, these require the use of the natural neighbour order of the data. Natural neighbour order is based on calculation of the Voronoi tessellation which is impractical for dimensions greater than about 10. How, then, can we relate the dataset smoothness to the smoothness of the required function approximation? One possibility is that the smoothness can be quantified by considering the accuracy at which local models with a defined smoothness property fit the training data. Local linear models may perform better when we are trying to approximate a function which is too complex for a given neural network.

Major problems with global neural network solutions include a) local minima or "flat spots" - as the network size (and hence the number of parameters) is increased, accuracy often decreases due to the training algorithm becoming stuck in local minima or "flat spots", b) excessive training time, and c) computational expense or dataset size reduction in order to perform accurate cross-validation for controlling the bias/variance tradeoff. Major problems with local approximation include a) interpolation in sparse, high dimensional spaces, and b) appropriate selection of local neighbourhoods and appropriate interpolation at all times. These are open issues for further research.

7 Conclusions

We have reviewed the use of global and local methods for estimating a function mapping $\mathcal{R}^m \Rightarrow \mathcal{R}^n$. We have examined the relationship between the methods in light of the smoothness assumption, and the bias/variance dilemma. We have performed an empirical comparison using the multi-layer perceptron (MLP) global neural network model, and local approximation models on a number of commonly used datasets covering both classification and prediction problems.

In order to find an *a priori* method to determine whether the global non-linear MLP approach or the local-linear approximation approach would be superior we considered histograms of k-NN density estimates for the training datasets. For the datasets we considered, we found that local-linear approximation performs better when the spread of the histogram is greater (ie. the density of the function to be approximated varies more as we move around the input space).

References

[1] S. Arya and D.M. Mount. Algorithms for fast vector quantization. In J. A. Storer and M. Cohn, editors, *Proceedings of DCC 93: Data Compression Conference*, pages 381–390. IEEE Press, 1993.

[2] J.L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

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

[4] M. Casdagli. Chaos and deterministic versus stochastic non-linear modelling. *J.R. Statistical Society B*, 54(2):302–328, 1991.

[5] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[6] 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.

[7] S.A. Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-6, 4:325–327, 1976.

[8] D. Farmer and J. Sidorowich. Exploiting chaos to predict the future and reduce noise. In W.C. Lee, editor, *Evolution, Learning, and Cognition*, pages 277–330. World Scientific, Singapore, 1988.

[9] E. Fix and J.L. Hodges. Discriminatory analysis – nonparametric discrimination: Consistency properties. Technical Report Project 21-49-004, Report No. 4, 261-279, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

[10] Peter V. Foukal. The variable sun. *Scientific American*, 262:34, 1990.

[11] J.H. Friedman. Multivariate Adaptive Regression Splines. *Annals of Statistics*, 19(1):1–141, 1991.

[12] J.H. Friedman, J.L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic time. Technical Report 75-482, Stanford CS, 1975.

[13] K. Fukunaga. *Introduction to Statistical Pattern Recognition, Second Edition.* Academic Press, Boston, MA, 1990.

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

[15] W. J. Gordon and J.A. Wixom. Shepards method of metric interpolation to bivariate and multivariate interpolation. *Mathematics of Computation*, 32 (141):253–264, 1978.

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

[17] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[18] M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[19] M.C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197:287, 1977.

[20] D.H. McLain. Drawing contours from arbitrary data points. *The Computer Journal*, 17(4):318–324, 1974.

[21] H.N. Mhaskar. Neural networks for localized approximation of real functions. In C.A. Kamm et al., editor, *Neural Networks for Signal Processing III: Proceedings of the 1993 IEEE Workshop.* IEEE, 1993.

[22] T. Poggio, F. Girosi, and M. Jones. From regularization to radial, tensor and additive splines. In C.A. Kamm et al., editor, *Neural Networks for Signal Processing III: Proceedings of the 1993 IEEE Workshop*, pages 3–10. IEEE, 1993.

[23] Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, California, 1993.

[24] A.J. Robinson. *Dynamic Error Propagation Networks.* PhD thesis, Cambridge University Engineering Department, Cambridge, UK, February 1989.

[25] T. Sauer. Time series prediction using delay coordinate embedding. In A.S. Weigend and N.A. Gershenfeld, editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley, 1993.

[26] D. Shepard. A two-dimensional function for irregularly spaced data. In *Proceedings of the 23rd ACM National Conference*, pages 517–524, 1968.

[27] H. Tong and K.S. Lim. Threshold autoregression, limit cycles and cyclical data. *Journal of the Royal Statistical Society B*, 42(3):245–292, 1980.

[28] D. Waltz. Memory-based reasoning. In M. Arbib and J. Robinson, editors, *Natural and Artificial Parallel Computation*, pages 251–276. MIT Press, Cambridge, MA, 1990.

[29] D.F. Watson. *Contouring: A Guide to the Analysis and Display of Spatial Data.* Pergamon Press, 1992.