Adaptive Computation Techniques for Time Series Analysis

by

Daniel Nikolaev Nikovski
Engineer of Computer Systems and Control
Technical University Sofia, Bulgaria

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Master of Science degree

Department of Computer Science
in the Graduate School
Southern Illinois University
at Carbondale
July 1995

ABSTRACT

of the thesis of DANIEL NIKOLAEV NIKOVSKI, for the Master of Science degree in COMPUTER SCIENCE, presented on July 1995, at Southern Illinois University at Carbondale.

TITLE: Adaptive Computation Techniques for Time Series Analysis

MAJOR PROFESSOR: Dr. Mehdi Zargham

We address the problem of predicting nonlinear time series by adaptive computation algorithms. Two connectionist learning models, Radial Basis Functions and Hierarchical Mixtures of Experts are used for state-space reconstruction of dynamical systems from training examples. This thesis is the first known application of Hierarchical Mixtures of Experts to the problem of time series prediction.

The two learning methods are tested on a number of experimental time series, including the logistic iterated map, output from a NH_3 far-infrared laser, and stock prices of five public-traded companies. Both methods predict accurately the output of the logistic iterated map and the laser. Furthermore, both of them outperform convincingly classical linear regression methods on these tasks. None of the methods, however, is able to predict the quickly changing dynamics of the stock market.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Mehdi Zargham, for his keen interest, assistance and guidance in my work on adaptive computation, as well as for giving me the opportunity to do research and teaching in this fascinating scientific field. Thanks are also due to my committee members, Dr. Kenneth Danhof and Dr. Norman Carver, for their time and suggestions. Special thanks to my family and friends for their care and support.

CONTENTS

ABSTRACT ii

ACKNOWLEDGEMENTS iii

LIST OF FIGURES vi

LIST OF TABLES vii

CHAPTERS

1. INTRODUCTION 1

    1.1 Statement of the Problem 1
    1.2 Theoretical Framework 1
    1.3 Purpose and Significance of the Study 4
    1.4 Scope and Delimitations of the Study 5
    1.5 Outline of the Rest of the Thesis 5

2. REVIEW OF RELATED LITERATURE 7

    2.1 Linear Time Series Prediction and Modeling 7
    2.2 Nonlinear Systems 8
    2.3 Adaptive Computation 10

3. RADIAL BASIS FUNCTIONS 13

    3.1 Computational Model 13
    3.2 Network Representation 16
    3.3 Approximation of Highly Nonlinear Functions 17

4. HIERARCHICAL MIXTURES OF EXPERTS 22

    4.1 Computational Model 22
    4.2 Learning Rules 25
    4.3 A Numerical Example 27
        4.3.1 First iteration 27
        4.3.2 Iteration 1001 30
    4.4 Approximation of Highly Nonlinear Functions 31

5. APPLICATIONS 34

    5.1 Programming Environment 34
    5.2 Learning the Logistic Map 35
    5.3 Prediction of $NH_3 - FIR$ Laser Output 37
    5.4 Stock Market Prediction 42

6. CONCLUSION 50

BIBLIOGRAPHY 52

VITA 55

LIST OF FIGURES

1 RBF network 17

2 Nonlinear test function. 18

3 RBF approximation, $\sigma = 0$, $S = 0.326$ 19

4 RBF approximation, $\sigma = 0.1$, $S = 0.330$ 20

5 RBF approximation, $\sigma = 0.2$, $S = 0.373$ 21

6 The tree-like architecture of HME 23

7 Training set for HME with one gate and two experts 28

8 HME approximation, $\sigma = 0$, $S = 0.326$ 32

9 HME approximation, $\sigma = 0.1$, $S = 0.330$ 32

10 HME approximation, $\sigma = 0.2$, $S = 0.373$ 33

11 Logistic map surface. 35

12 Realization of the logistic iterated map 36

13 Logistic map: prediction error vs. number of RBFs. 38

14 Logistic map: prediction error vs. number of experts in HME. 38

15 Output of a NH_3 far-infrared laser. 39

16 Monthly prices of IBM stock over 240 months. 45

17 Monthly returns of IBM stock over 240 months. 45

18 Prediction of IBM stock prices with 10 RBFs 47

19 Prediction of IBM stock prices with 50 RBFs 47

20 Prediction of IBM stock prices with 100 RBFs 48

LIST OF TABLES

5.1 NRMSE of RLS, RBF, and HME on the laser output time series. . . . 41

5.2 Comparison of RBF, HME, and RLS: t-statistic and attained significance level α. 42

5.3 Average NRMSE of RBF over 40 predictions, with retraining 46

5.4 Out-of-sample NRMSE of RBF and HME on daily stock prices. 49

CHAPTER 1

INTRODUCTION

1.1 Statement of the Problem

The analysis of time series presents an important problem in many areas of science and technology. Disciplines such as econometrics, systems science, communication technology, and control theory among others need efficient methods to analyze and predict the time course of observed variables. A time series consists of a set of observations of a variable, measured at equal time intervals. Typical examples of time series are the prices of stocks and other securities, or the measurements of the output of a physical system such as the position of an aircraft, ship, or robot manipulator.

In all approaches to time series analysis it is assumed that the time series is generated by an underlying dynamic process. The problem of time series analysis consists of creating a suitable mathematical model of this underlying dynamic process. Such a model can serve a number of purposes. First, it can be used to gain insights into the nature of the observed phenomenon. Second, it can be used to predict the future behavior of the process. Third, the knowledge of the way the dynamic system operates can be used to control this system.

1.2 Theoretical Framework

In the general case the system that corresponds to a given time series can be modeled by a set of coupled stochastic difference equations of the form

$$X^{(t+1)} = f(X^{(t)}, W^{(t)}, t) \tag{1.1}$$

$$y^{(t)} = g(X^{(t)}, v^{(t)}, t) \tag{1.2}$$

Here X is a vector variable, called the state of the system, y is the measurable output variable that forms the time series, and W and v are stochastic noise variables. Note that X and W are vector variables, while y and v are scalars. The discrete time is denoted by t.

The traditional approach to time series modeling is to make some major simplifications in the model above in order to make the analysis tractable. First, the system is assumed to be time-invariant, that is, the dependence of the functions f and g on the time t is dropped. Next, the functions f and g are assumed to be linear and are replaced by the two matrices F and H and the vector G:

$$X^{(t+1)} = FX^{(t)} + HW^{(t)} \tag{1.3}$$

$$y^{(t)} = GX^{(t)} + v^{(t)} \tag{1.4}$$

The set of equations 1.1 and 1.2 can be analyzed with a wide variety of techniques such as regression analysis, ARIMA modeling, and optimal filtering, which will be discussed in the next chapter. The use of these techniques is the standard practice in both econometrics and systems science.

However, the simplifications leading to linear systems are not always justified in real applications. There is no such thing as a perfectly linear system in the real world, and by necessity linear models are only approximations of the real processes. In certain cases such approximations are not good enough. Thus, many

prediction, system identification and control tasks call for nonlinear models, instead of the traditional linear ones.

It has been only recently that notable progress in nonlinear modeling has been possible. New developments in two areas — nonlinear dynamics and nonsymbolic artificial intelligence (AI) have made possible the creation of general nonlinear modeling techniques. These techniques convincingly outperform linear modeling for many complex applications [37].

In 1981 F.Takens proved an important theorem which justifies the use of lagged (delayed) values of the time series variable to reconstruct the state space of the underlying dynamic system [33]. According to the theorem, if the dimensionality of the system is d, a vector Y consisting of the most recent $2d+1$ values of the measured variable y can be used to identify unambiguously the true state of the system X at that moment of time. That is, there exists a diffeomorphism (one-to-one mapping) $\mathcal{M}$ between the vector Y and the system state X. The dimension $2d+1$ of the vector Y is called embedding dimension, and the vector Y itself is called embedding vector, or time-lagged vector. That is,

$$Y^{(t)} = [y^{(t)}, y^{(t-1)}, y^{(t-2)}, \ldots, y^{(t-2d)}]^T \tag{1.5}$$

However, this is only an existence theorem, and there are a number of issues to be solved before the theorem can be put to practical use. First, it is not easy to establish from the time series data what the dimensionality of the system is. This issue arises with linear modeling too, but there the autocorrelation function of the time-series can be used to this end [2]. Among the techniques used in nonlinear modeling are estimation of the correlation integral, the fractal dimension and various box-counting methods [34, 14].

The second major issue, which is also the topic of this thesis, is the discovery of the actual mapping between the lagged vector and the system state. In

practice, we are more interested in another mapping $\mathcal{N}$ which includes $\mathcal{M}$. We are interested in the function $\mathcal{N}$ that maps the current lagged vector Y_n to the value of the time series y_{n+1} at the next moment.

Unfortunately, the theorem does not tell us anything about what this mapping might be. The only thing that is sure is that the mapping exists, and that the mapping is a function indeed. This follows from the fact that $\mathcal{M}$ is a diffeomorphism. The mapping $\mathcal{N}$ is the composition of $\mathcal{M}$, the system function f, and the inverse $\mathcal{M}^{-1}$ of $\mathcal{M}$ (which exists under the consequences of the theorem):

$$\mathcal{N}(Y) = \mathcal{M}^{-1}(f(\mathcal{M}(Y))) \tag{1.6}$$

What we need in order to discover $\mathcal{N}$ is a universal function approximator (UFA) that can capture the mapping if it is given examples in the form of lagged vectors. We can use to this end techniques from the area of nonsymbolic artificial intelligence (AI), which are also called adaptive computation techniques. The computational models most often used in function approximation are multilayer perceptrons (MLP) [35] and radial basis functions (RBF) [4].

The utility of these techniques is that they can infer nonlinear dependencies between data variables that take on real values. Their predecessors are the traditional symbolic learning methods in AI that work with Boolean variables only. The computational and learning abilities of various adaptive computation models are discussed in the next chapter.

1.3 Purpose and Significance of the Study

The purpose of the present study is to apply several adaptive computation techniques to the prediction of complex time series. The specific research goal is to investigate which methods and under what conditions will outperform the traditional linear modeling techniques. Of special interest is if adaptive

computation techniques can be used in complex, noisy, and time-variant environments such as the stock market.

If successful, the techniques can be used to increase the precision of econometric forecasts and improve the monitoring and control of complex physical devices and natural phenomena.

1.4 Scope and Delimitations of the Study

The field of adaptive computation and its application to time series analysis has undergone an explosive growth in the recent years. It is obvious that a very comprehensive investigation, comprising all known adaptive computation methods is not feasible.

Instead, the two most promising techniques are put to test — radial basis functions (RBF) [28, 29] and hierarchical mixtures of experts [24]. RBFs are known to possess good approximating abilities and fast learning. On the other side, the results of the operation of HME are more suitable for subsequent analysis, and this model can possibly track time-variant systems.

1.5 Outline of the Rest of the Thesis

Chapter 2 surveys the literature on time series analysis. The popular ARIMA models are described, as well as the problem of recursive identification of linear time-invariant systems. The nonlinear system identification problem is further discussed and fractal dynamics are introduced. A brief review of the field of adaptive computation concludes the chapter.

Chapters 3 and 4 deal with the computational architecture and learning rules of RBF and HME, respectively. The function approximation abilities of the two models are explored.

Chapter 5 reports the experimental results from the application of RBF and HME on three types of time series: the logistic map (a computer generated highly

nonlinear time series), a $NH_3 - FIR$ laser that exhibits chaotic behavior, and stock prices of a number of companies.

Chapter 6 discusses the results, draws conclusions and suggests directions for further research.

CHAPTER 2

REVIEW OF RELATED LITERATURE

2.1 Linear Time Series Prediction and Modeling

Before the 1920s the modeling of time series was done by interpolating the value of the time series forward in the time domain. This approach did not consider the dynamic system underlying the time series, and instead treated it as a normal function of time.

The state-space approach to time series analysis was applied for the first time by Yule in 1927. He used autoregressive techniques to predict the annual number of sunspots. An autoregressive (AR) equation on M lagged values of the time series has the form

$$y^{(t)} = \sum_{m=1}^{M} a_m y^{(t-m)} + v^{(t)} \tag{2.1}$$

where $y^{(t)}$, $t = 1, T$ are the values of the time series, $v^{(t)}$ is Gaussian noise, and a_m, $m = 1, M$ are called regression coefficients.

The system can also be viewed as driven by more than one noise terms $v^{(t-p)}$, $p = 1, P$, each of which has it own contribution measured by a coefficient b_p:

$$y^{(t)} = \sum_{p=1}^{P} b_p v^{(t-p)} \tag{2.2}$$

For historical reasons the latter model is called a moving average (MA) model. Indeed, the signal $y^{(t)}$ is viewed here as a weighted avearge of the noise components $v_{(t-p)}$.

A natural approach is to combine the two models by allowing both an autoregressive and a moving average component:

$$y^{(t)} = \sum_{m=1}^{M} a_m y^{(t-m)} + \sum_{p=1}^{P} b_p v^{(t-p)} \tag{2.3}$$

This model is called accordingly an ARMA(M,P) model. Box and Jenkins [2] proposed an efficient algorithm for calculating the coefficients a_m and b_p from the spectrum of the signal.

The Box-Jenkins algorithm is a batch algorithm — it needs all samples of the time-series in order to estimate the spectrum and find the coefficients. The method works very well with systems that are close to linear and are time-invariant. For systems whose dynamics vary with time, however, it is not directly applicable. For such cases, the autoregressive method can be extended to an on-line method that can track these changes in dynamics. The algorithm takes the form of a learning rule that updates the model A of all autoregressive parameters a_m after each sample arrives:

$$A^{(t)} = A^{(t-1)} + \rho Y^{(t)}(y^{(t)} - (A^{(t-1)})^T Y^{(t)}) \tag{2.4}$$

Here $y^{(t)} - (A^{(t-1)})^T Y^{(t)}$ is the error of the model - the difference between the sampled and the predicted values of the time series. The correction of the parameter vector A is proportional to this error via a learning rate factor ρ. For this case, the optimal value for ρ can be determined analytically [26] but practically any small positive value will result in successful approximation of the model.

Similar, although not so simple learning rules can be derived for the case of general ARMA models [26]. The use of such on-line algorithms is known under the name recursive system identification.

2.2 Nonlinear Systems

For nearly 50 years linear models dominated entirely the field of time series prediction and modeling. Their use, however, is justified only when the dynamical

equations of the system are linear or close to linear. While a number of important systems in nature are indeed linear, many more are not. The linear approach relies on the premise that the power spectrum of a time series characterizes it well. There are, however, many non-linear systems of very low dimensionality that have frequency components in all parts of the spectrum and look like nondeterministic, noisy processes. For example, the logistic iterated map

$$y^{(t)} = \lambda y^{(t-1)}(1 - y^{(t-2)}) \tag{2.5}$$

where λ is a constant, has very broad spectrum, and because of this cannot be modeled as an ARMA process of reasonably small dimensionality.

This means that the state of such a system in state space does not stay on a hyperplane of low dimensionality. Rather, it wanders along a general nonplanar hypersurface (a nonlinear manifold), which is also called strange attractor. The attractor of a dynamical systems is the set of points to which the trajectory of the system in state space converges. A linear system has a simple attractor - a fixed point or a simple limit cycle, while the attractor of a nonlinear system might have complex form, hence the name strange. ARMA modeling is deceived by strange attractors and fails to discover that the system is actually low-dimensional, well determined, and in principle predictable.

Estimating the true nonlinear dimension of the attractor (and determining if the system is indeed nonlinear of low dimensionality) has been an intractable task until the 1980s. In 1985 Grassberger and Procaccia proposed an algorithm for discovering the dimensionality of a general manifold [11]. The more computationally efficient class of box counting algorithms was introduced by Theiler to the same end in 1987 [34]. Typically, these algorithms require an enormous number of samples of the time series in order to estimate reliably the dimension of the attractor of the system.

To make matters worse, nonlinear systems often exhibit chaotic behavior, which makes them unpredictable even if somehow the true dynamics of motion could be found. The term "chaos" refers to the sensitivity of the trajectory that the dynamic system follows in state space on the initial point from where it starts. In a regular, non-chaotic system two trajectories that start from nearby points stay close as time progresses. On the contrary, in a chaotic system two trajectories might diverge very quickly even when they start from almost identical points. Lorenz demonstrated that the evolution of many dynamical systems, including the weather, cannot be predicted reliably for long time in the future, because a small variation in the starting point may result in totally unpredictable outcome further ahead in time. The rate with which the system becomes unpredictable is determined by the Lyapunov exponents of the system — that is, the eigenvalues of the linearized matrix F of equation (1.1).

Estimating the true dimension of the attractor is of large practical importance in light of the theorem of Takens, discussed in Chapter 1. If the dimensionality of the attractor is reasonably small, we can expect that a mapping exists to the unknown value of the time series from a lagged vector consisting of only twice as many previous values of the time series. Discovering what this mapping is, of course, presents another problem.

2.3 Adaptive Computation

Discovering general nonlinear mappings from examples has not been a very well researched problem before the advent of connectionism in the mid-1980s. In the linear case, statistical regressive procedures have been in use for a long time, but they do not extend automatically to nonlinear mappings. At the same time, the machine learning methods of the classical symbolic AI that can discover Boolean functions from examples, do not extend to real-valued functions either.

The new field of connectionism was born in the mid-1980s from the merging of ideas from classical AI, neural science, very large scale integration (VLSI) circuits, and statistics [31]. The central tool in connectionism is the Universal Function Approximator (UFA) in the form of parallel and distributed parametrized computational architectures. The best known connectionist architectures are the Multi-Layer Perceptrons (MLP), popularly known as "neural networks". A type of MLP, called Time-Delay Neural Network (TDNN) has been very successfully used for time series analysis and prediction [35].

The problem with MLP is that the learning is very slow — on the order of hundreds of thousands of iterations through the training set, while linear algorithms need typically one to ten iterations to converge. Paradoxically, the reason for the slowness of MLP is exactly the property that gives its power — the fact that MLPs do the approximation with a generic nonlinear function, the sigmoid.

Other less known, but arguably more powerful and faster learning architectures are Radial Basis Functions (RBF), and Hierarchical Mixtures of Experts (HME). RBFs have been used for time series analysis by Casdagly [4] and Hutchinson [20]. An application of HME to this task has been done by Waterhouse and Robinson [36]. These two methods will be described in greater detail in Chapters 3 and 4 respectively.

Another successful extension of the classical AI methods are Fuzzy Logic Systems (FLS). These systems learn fast and their operation can be explained in terms of firing rules in a fuzzy production system.

Yet another AI-related method for function approximation comes from the field of Genetic Programming (GP). The respective machine learning procedure there is called symbolic regression [25]. Basic genetic algorithms have also been extensively used for financial modeling [1, 6].

All these methods are examples of adaptive computation, or soft computing — that is, techniques that rely on self-modifying learning rules to approximate complex dependencies. These methods are to a great extent complementary to each other.

CHAPTER 3

RADIAL BASIS FUNCTIONS

3.1 Computational Model

Radial Basis Functions (RBF) have been known in approximation theory as a powerful and computationally efficient method for interpolation and approximation of functions. With the development of the connectionist approach to computation, it was noticed that the computation performed by these functions can be carried out by parallel and distributed processing elements similar to MLP [28].

RBFs solve the real multivariate interpolation problem, defined as follows [28]:

Given T different points $\{X^{(t)} \in R^n | t = 1, T\}$ and T real numbers $\{y^{(t)} \in R | t = 1, T\}$ find a function F from R^n to R satisfying the interpolation conditions:

$$F(X^{(t)}) = y^{(t)}, \quad t = 1, T \tag{3.1}$$

We can see that this problem is exactly the same as the problem of learning the underlying dynamic process of a time series, as defined in Chapter 1.

Unlike MLPs, the RBF approach chooses F from a linear functional space of dimension T. The fact that this space is linear simplifies immensely the search for the function F and results in much faster learning than the one possible with MLP.

The basis of the functional space is chosen to be the set of functions

$$\{h(\| X - X^{(t)} \|) \quad t = 1, T\} \tag{3.2}$$

where the continuous function h is called radial basis function, and $\| \cdot \|$ is the L^2 norm on R^n. In other words, the RBF h takes as its argument the distance $r = \| X - X^{(t)} \|$ from a point X to an example $X^{(t)}$ from the training set. Note that even if h is a linear function with respect to its argument (the distance r), it is nonetheless nonlinear with respect to X itself. Thus the RBFs provide nonlinear processing of the raw inputs.

Possible and popular RBFs are the following [28]:

$$h(r) = e^{-(\frac{r}{c})^2} \quad \text{(gaussian)} \tag{3.3}$$

$$h(r) = \frac{1}{(c^2 + r^2)^\alpha} \quad \alpha > 0 \tag{3.4}$$

$$h(r) = (c^2 + r^2)^\beta \quad 0 < \beta < 1 \tag{3.5}$$

$$h(r) = \sqrt{r^2 + c^2} \quad \text{(multiquadric)} \tag{3.6}$$

$$h(r) = r \quad \text{(linear)} \tag{3.7}$$

where c, α, and β are suitable constants.

The solution of the interpolation problem has the following form [28]:

$$F(X) = \sum_{t=1}^{T} \omega_t h(\| X - X^{(t)} \|) \tag{3.8}$$

A polynomial term can be added to the RBF expansion too [28].

Since the expression above is linear in the coefficients ω_t, the RBF approach can be considered to be a generalized least squares algorithm, and solved as a

system of linear equations. Let the vectors Y, Ω, and the matrix H be defined as follows:

$$(Y)_j = y^{(j)}, \quad (\Omega)_i = \omega_i, \quad (H)_{ij} = h(\| X^{(j)} - X^{(i)} \|) \tag{3.9}$$

Then the coefficients ω_t are given by

$$\Omega = H^{-1}Y. \tag{3.10}$$

The above set of equations is solvable whenever the matrix H is not singular.

In the general case, however, we are not interested in precise interpolation of the training set, but in its approximation in an optimal, least squares sense. There is usually noise in the measured data in the training set, and precise interpolation is not very meaningful anyway. A reasonable approximation, possibly with few coefficients w_t, is preferred in practice.

A number of approaches is possible to this end. The obvious and easiest approach is to take RBFs on only a subset of the training examples, being careful that this subset span the whole area occupied by the original training set. The chosen examples are called knots. Another approach is to perform clustering on the training set, and take as knots for the RBFs the centers of the formed clusters. This is the approach taken by Hutchison [20] for time series analysis.

Whatever the approach is, let C_α are the chosen knots. The expansion will then have the form

$$F(X) = \sum_{\alpha=1}^{K} \omega_\alpha h(\| X - C_\alpha \|) \tag{3.11}$$

where K is the number of knots, $K < T$. The requirement $F(X^{(t)}) = y^{(t)}$ results in the system

$$y^{(t)} = \sum_{\alpha=1}^{K} \omega_\alpha h(\| X^{(t)} - C_\alpha \|) \quad t = 1, T \tag{3.12}$$

The system is overconstrained, because it has T equations for K unknowns. The most we can hope for is to solve it in a least-squares sense. This can be done in many ways, for example with Singular Value Decomposition ([30]) or by using the Moore-Pensrose pseudoinverse [28]. With the latter approach, the solution for the expansion coefficients ω_α is given by ([28])

$$\Omega = H^+ Y \tag{3.13}$$

where $(H)_{i\alpha} = h(\| X^{(i)} - C_\alpha \|)$ and the pseudoinverse of H is given by ([28])

$$H^+ = (H^T H)^{-1} H^T \tag{3.14}$$

This can be done, of course, only when $(H^T H)^{-1}$) exists.

In practice, however, an on-line learning rule for the coefficients ω_t might be preferable, especially for large training sets. The recursive least squares learning rule (given in chapter 2) can be used.

3.2 Network Representation

The computation described above can be performed very efficiently by a parallel two-layer network as shown in Figure 1.

The nodes compute the distance from a number of centers to the input vector X, aind apply an RBF to this distance. In the case when the centers are less than the number of training examples, the network is called a generalized RBF network.

It should be noted that this computing architecture bears much more resemblance to natural neural circuits than the MLP architecture [28]. Thus, if any parallel distributed architecture has to be called "neural network", RBF networks deserve to be called so much more than MLPs.

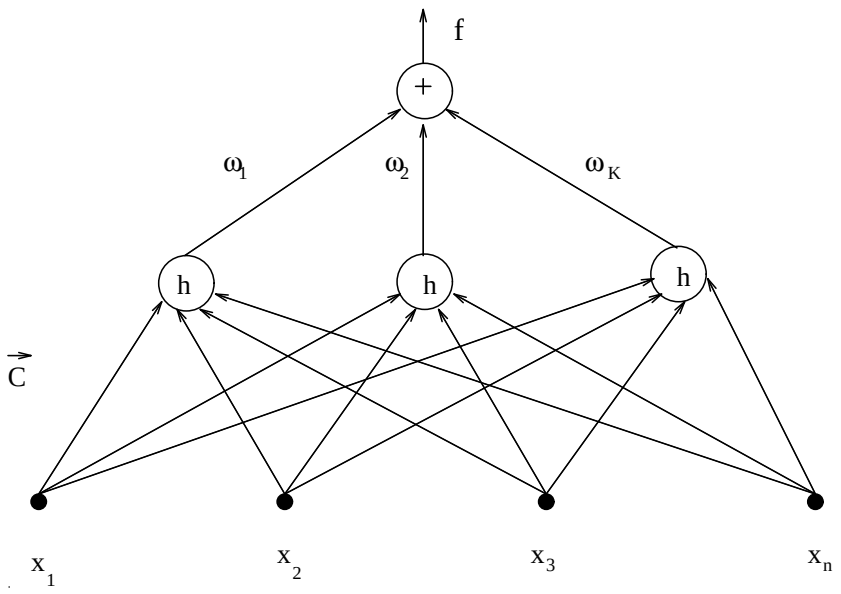

Figure 1. RBF network

RBF networks can be readily implemented in VLSI hardware. The NI1000 chip from Nestor/Intel and the ZISC chip from IBM are examples [21].

A further extension of the model, proposed by Poggio and Girosi ([28]) is to replace the L_2 norm by a weighted norm. This extension is called HyperBF functions.

3.3 Approximation of Highly Nonlinear Functions

To test the approximation abilities of an RBF network, a two dimensional highly nonlinear function was tested. The purpose of the test was to discover how many centers are typically needed for good approximation, how well the network will deal with noise, and what the possible limit to approximation is with this method.

The chosen function was (Figure 2)

$$y(x_1, x_2) = log(1 + \frac{sin(3\pi x_1)}{(3\pi x_1)})(1 - x_2) - 0.3 \tag{3.15}$$

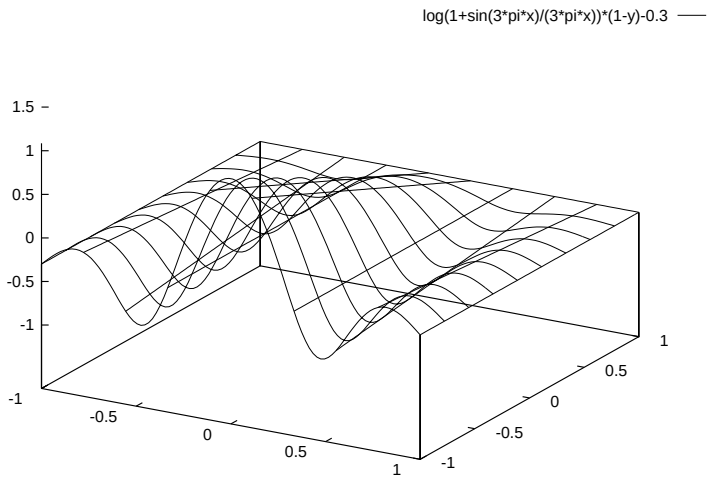

Figure 2. Nonlinear test function.

Three cases were tested, with different levels of Gaussian measurement noise, added to both the inputs and the output. For each of the cases both the training and testing sets had 1000 examples each. The in- and out-of-sample error were computed as the root of the mean squared error (RMSE) on the training and testing sets respectively:

$$RMSE = \sqrt{\frac{1}{1000}\sum_{t=1}^{1000}(y^{(t)} - \hat{y}^{(t)})^2} \tag{3.16}$$

where $\hat{y}^{(t)})^2$ is the output of the RBF network given input $X^{(t)}$.

For each of the cases the sample standard deviation S of the training set is also computed, in order to give idea of how well the network approximates the function. S is defined as

$$S = \sqrt{\frac{1}{1000}\sum_{t=1}^{1000}(y^{(t)} - \bar{y})^2} \tag{3.17}$$

where $\bar{y}$ is the average of all $y^{(t)}$.

The RMSE was divided by S to give the normalized root mean squared error (NRMSE), which is a practical estimate of the approximation power of the predictors.

The linear RBF $h(r) = r$ was used in all experiments, based on Casdagli's observation that various RBFs give comparable performance [4].

The only free parameter in a generalized RBF network is the number of knots. The in-sample and out-of-sample NRMSE were plotted against the number of knots. The results are shown in Figures 3, 4, and 5.

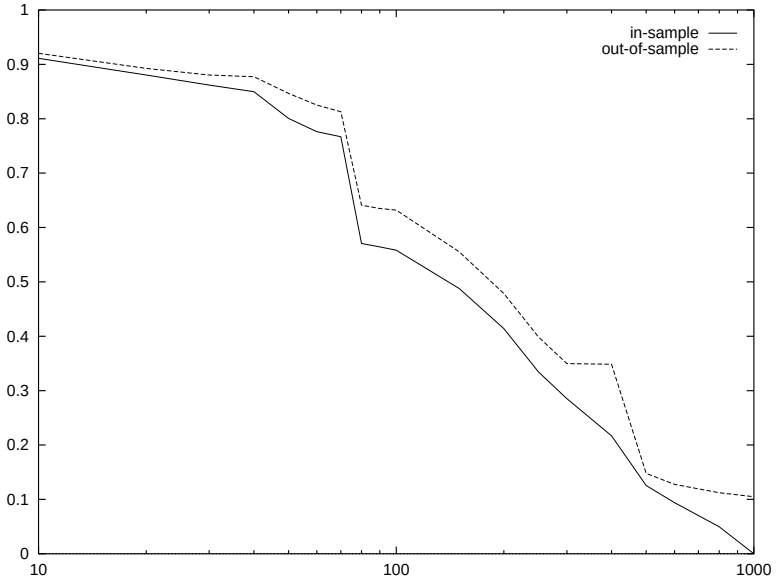

Figure 3. RBF approximation, $\sigma = 0$, $S = 0.326$

Figure 3 demonstrates how the out-of-sample error is successfully driven down by reducing the in-sample error, i.e., good generalization occurs indeed. The out-of-sample NRMSE asysmptotes to about 10%, which can be considered to be the practically attainable precision with this method, if no noise is present in the data.

If the data are noisy, as in Figure 4, generalization still occurs, yet the practically attainable precision of a prediction is much worse. The added noise

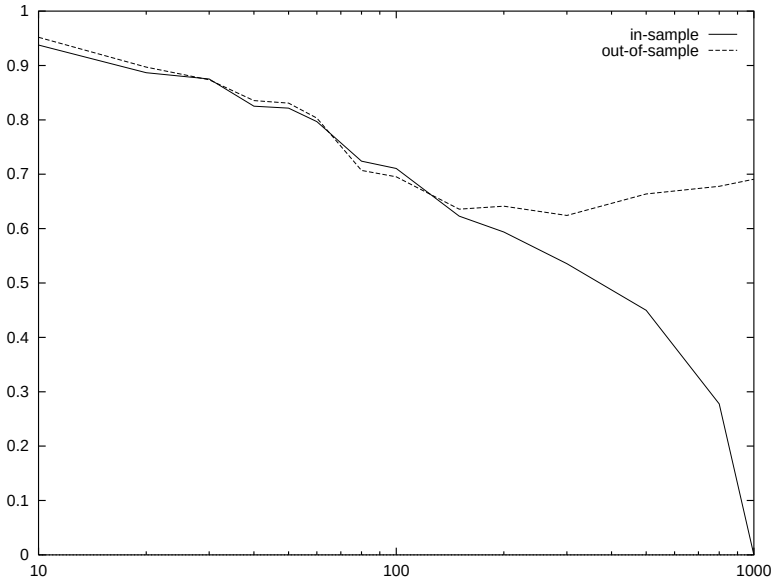

Figure 4. RBF approximation, $\sigma = 0.1$, $S = 0.330$

has relative amplitude of $\sigma/S = 30\%$, which corresponds to a signal-to-noise (S/N) ratio of about 3. Notice that the asymptotic value of the NRMSE (65%) is more than just the sum of the amplitude of the noise and the approximation error for noiseless data (10%, as discovered in Figure 3).

Figure 4 demonstrates also that overfitting occurs, that is, allowing more free parameters than necessary worsens the generalization, because the extra parameters fit the noise instead of rejecting it. As the figure shows, the best generalization is achieved when the number of RBFs is about one tenth to one third of the number of training examples. This agrees well with results from MLP learning [18].

Figure 5 demonstrates the maximum noise level that still permits at least some generalization. Here the noise amplitude σ is 54%, which corresponds to an S/N ratio of about 2. Note that the noise increases the in-sample error so much that the out-of-sample error is initially lower. (The out-of-sample error is calculated with respected to the true, noiseless value of the function.)

The conclusion from these experiments is that RBFs can reliably generalize

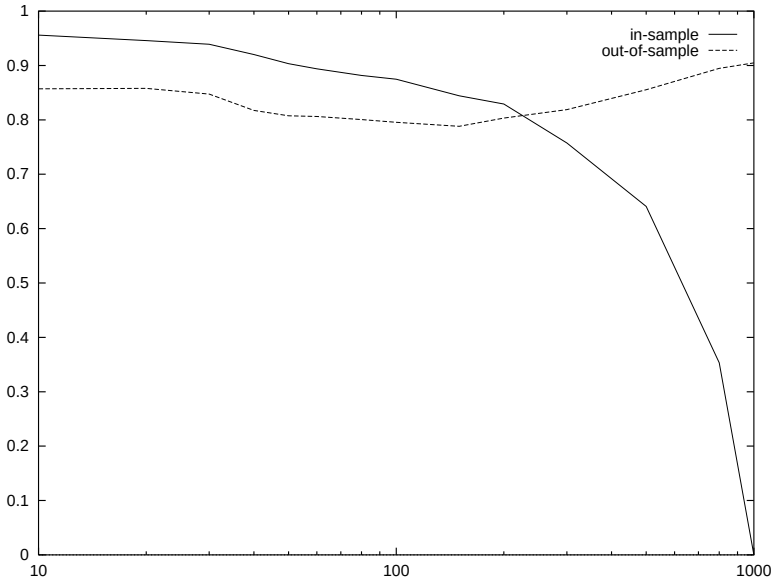

Figure 5. RBF approximation, $\sigma = 0.2$, $S = 0.373$

from examples if the S/N ratio is at least 2 and about 10% of the training examples are taken as centers of the RBF elements.

CHAPTER 4

HIERARCHICAL MIXTURES OF EXPERTS

4.1 Computational Model

In this chapter an alternative method for approximation of nonlinear functions is considered — the Hierarchical Mixture of Experts (HME) model [24]. In addition to being able to perform good nonlinear approximation, this method is able to learn time-varying functions too.

The HME model, like RBFs, also extends the linear learning algorithm, described in Chapter 2, but in a way different than the one taken by the RBF approach. The idea of the method is to view the nonlinear function as a piecewise linear function — that is, formed by separate linear hyperplanes in different regions of the input space.

If this approach is taken, a number of additional subproblems arise:

1. How are the optimal boundaries of these regions to be found?
2. How can the hyperplanes be locally tuned, i.e., for their own region only?
3. How can the individual hyperplanes be "blended", so that the approximated function is smooth?

Nowlan, Hinton, Jordan and Jacobs proposed a general architecture, called Mixture of Experts, that solves successfully these problems. Jordan and Jacobs subsequently refined the architecture, adding a hierarchy in the processing and called it accordingly Hierarchical Mixture of Experts (HME) [24].

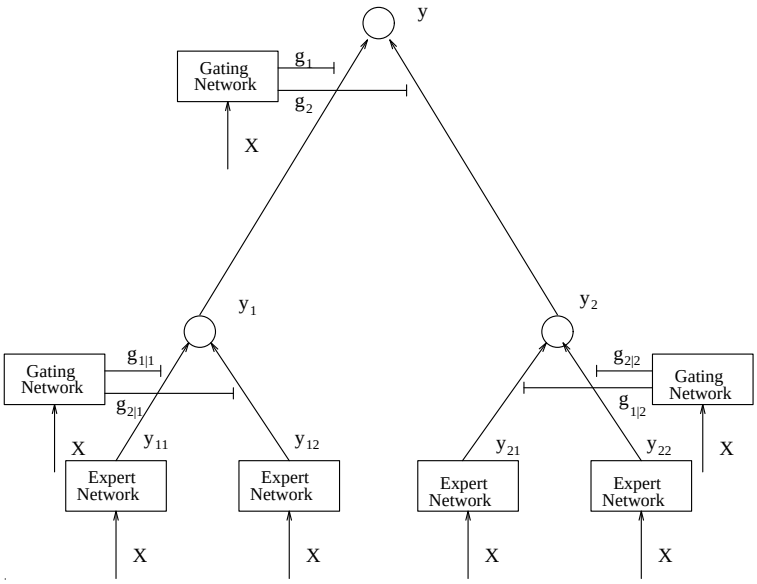

Figure 6. The tree-like architecture of HME

An HME has two types of processing elements — experts and gates (Figure 6). A hierarchy in the form of a tree is built from these nodes, with the gates being the nonterminal nodes of the tree and the experts forming the leaves of the tree. The purpose of the experts is to form locally linear models for a prespecified number of regions (equal to the number of leaves of the tree). Each of the experts forms a linear prediction $\hat{y}_{ij}$ of the true output of the network y for a given input vector X:

$$\hat{y}_{ij} = U_{ij}^T X \tag{4.1}$$

where U_{ij} is the vector of linear parameters.

Some of the experts' outputs are close to the true value of y, some are not; i.e., some experts are "more qualified" to produce output for a given vector X than others. The purpose of the gating nodes is to assign weights to the predictions $\hat{y}_{ij}$ of the individual experts — this weight should be proportional to the precision with which an expert approximates the function at the current

location. For the case of binary trees, shown in Figure 6, two values g_1 and g_2 are output by the gate at the root of the tree, with their sum equal to 1:

$$g_i = \frac{e^{\xi_i}}{\sum_k e^{\xi_k}},\ i = 1, 2,\ k = 1, 2 \tag{4.2}$$

$$\xi_i = V_i^T X \tag{4.3}$$

Here the vector V_i parametrizes the gating expert.

The predicted output $\hat{y}$ is then a weighted average of the predictions at the lower level $\hat{y}_i$:

$$\hat{y} = \sum_{i=1}^{2} g_i \hat{y}_i \tag{4.4}$$

The weights of the gating expert perform a partioning of the input space into two parts, in each of which one of the weights is close to 1, while the other weight is close to 0. For example, the gating weight g_1 is given by

$$g_1 = \frac{e^{\xi_1}}{e^{\xi_1} + e^{\xi_2}} = \frac{1}{1 + e^{-(V_1 - V_2)^T X}} \tag{4.5}$$

This is a logistic ridge function, whose steepness and orientation are determined by the vector $V_1 - V_2$. The larger the length of this vector, the steeper the split is.

The predictions at the lower level are formed in a similar way, this time using directly the predictions of the experts:

$$\hat{y}_i = \sum_{j=1}^{2} g_{j|i} \hat{y}_{ij} \tag{4.6}$$

$$g_{j|i} = \frac{e^{\xi_{ij}}}{\sum_k e^{\xi_{ik}}}, \; i = 1, 2, \quad j = 1, 2, \quad k = 1, 2 \tag{4.7}$$

$$\xi_{ij} = V_{ij}^T X \tag{4.8}$$

More intermediate levels can be added, increasing the number of experts in the network. The trees do not have to be binary — any branching factor can be used.

4.2 Learning Rules

Jordan and Jacobs [24] derived learning rules for the experts and the gates on the basis of a probability model and a maximum-likelihood approach. A probability model for each expert has to be defined in order to use maximum likelihood estimation. We are interested in the case when the noise is Gaussian with unit variance and the expert is linear. Then the probability distribution of the output of a particular expert is given by

$$P(y|X, U_{ij}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(y-\hat{y}_{ij})^2}{2}} \tag{4.9}$$

This is a measure of how well the predicted output $\hat{y}_{ij}$ matches the observed output y. Since the gating coefficients come from a multinomial distribution (a binomial one in the case of a binary tree), the probability distribution of the root gate is given by

$$P(y|X, \theta) = \frac{1}{\sqrt{2\pi}} \sum_i g_i \sum_j g_{j|i} e^{-\frac{(y-\hat{y}_{ij})^2}{2}} \tag{4.10}$$

Here the parameter vector θ consists of all possible parameters of the HME model (parameters of all gates and experts). The likelihood $\mathcal{L}(\theta, y|X)$ that the

observed output y is generated by the HME model with a particular set of parameters θ is precisely the probability defined above:

$$\mathcal{L}(\theta; y|X) = P(y|X, \theta) \tag{4.11}$$

Imrovement of this likelihood can be obtained by applying a suitable learning rule. It is easier to work with the log-likelihood instead of the original likelihood:

$$l(\theta, y|X) = ln\mathcal{L}(\theta; y|X) = ln\frac{1}{\sqrt{2\pi}}\sum_i g_i \sum_j g_{j|i} e^{-\frac{(y-\hat{y}_{ij})^2}{2}} \tag{4.12}$$

The log-likelihood can be maximized by the popular gradient ascent algorithm that updates the parameters in the direction of the gradient of the log-likelihood. The following on-line learning rules result:

$$\Delta U_{ij} = \rho h_i h_{j|i}(y - \hat{y}_{ij})X \tag{4.13}$$

$$\Delta V_i = \rho(h_i - g_i)X \tag{4.14}$$

$$\Delta V_{ij} = \rho h_i(h_{j|i} - g_{j|i})X \tag{4.15}$$

where the quantities h_i, $h_{j|i}$ and h_{ij} are called posterior probabilities and are defined as follows:

$$h_i = \frac{g_i \sum_j g_{j|i} P_{ij}(y|X, \theta)}{\sum_i g_i \sum_j g_{j|i} P_{ij}(y|X, \theta)} \tag{4.16}$$

$$h_{j|i} = \frac{g_{j|i} P_{ij}(y|X, \theta)}{\sum_j g_{j|i} P_{ij}(y|X, \theta)} \tag{4.17}$$

$$h_{ij} = \frac{g_i g_{j|i} P_{ij}(y|X, \theta)}{\sum_i g_i \sum_j g_{j|i} P_{ij}(y|X, \theta)} \tag{4.18}$$

Note that h_{ij} is the product of h_i and $h_{j|i}$.

Jordan and Jacobs have recently derived another set of learning rules, based on the Expectation-Maximization (EM) algorithm used in statistics [24]. These EM learning rules learn even faster than the ones based on the maximum-likelihood approach.

4.3 A Numerical Example

To illustrate the operation of the learning algorithm, a two-dimensional numerical example will be developed in this section. The network consists of two experts and one gating node.

The data were generated by two linear functions:

$$y = 2x - 1, \quad x > 0, \tag{4.19}$$

and

$$y = -x, \quad x \leq 0. \tag{4.20}$$

The choice of generating function was random, with equal probability that either line would be used. The training and testing set consisted of 100 examples each. The examples in the training set are shown in Figure 7.

The operation of the algorithm will be illustrated for one of the training examples, $x = -0.535187$, $y = 0.535187$. Clearly, this example was generated from the second straight line above. Two moments of the operation of the algorithm will be shown — the first one for the very first iteration of the algorithm, and the second one for the 1001-th iteration (that is, when the algorithm has almost converged.)

4.3.1 First iteration

The random number generator initializes the parameter vectors of the experts U_1 and U_2 and the parameter vectors of the gate V_1 and V_2 as follows:

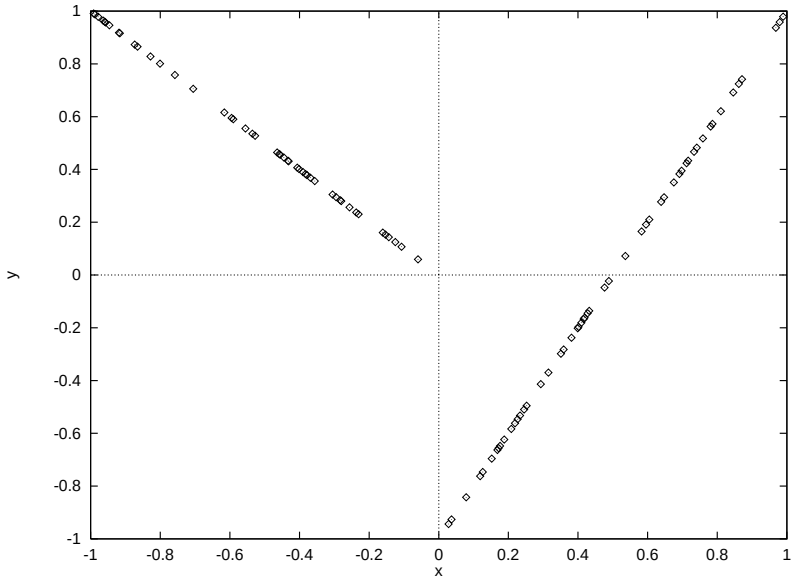

Figure 7. Training set for HME with one gate and two experts

$$U_1 = \begin{bmatrix} -0.269907 \\ -0.869579 \end{bmatrix} \qquad U_2 = \begin{bmatrix} -0.481137 \\ 0.581966 \end{bmatrix}$$

$$V_1 = \begin{bmatrix} -0.681001 \\ -0.664693 \end{bmatrix} \qquad V_2 = \begin{bmatrix} -0.086941 \\ 0.09521 \end{bmatrix}$$

The input x is augmented with a unit element to form the vector X:

$$X = \begin{bmatrix} -0.535787 \\ 1 \end{bmatrix}$$

The work of the algorithm starts by finding the weighting factors g_1 and g_2:

$$\xi_1 = 1.029156 \quad \xi_2 = 0.141739$$

$$g_1 = 0.708357 \quad g_2 = 0.291643$$

Note that the first expert is given more weight ($g_1 > g_2$), and is thus likely to represent this example and the others from the same line in the future iterations.

The next step is to produce the "opinions" $\hat{y}_1$ and $\hat{y}_2$ of the two experts, as well as the probabilities P_1 and P_2 that the real output has been produced by each of the experts:

$$\hat{y}_1 = -0.725129 \qquad \hat{y}_2 = 0.839464$$

$$P_1 = 0.204253 \qquad P_2 = 0.911572$$

Clearly, the second expert "explains" this example better, and thus tends to represent this example and the others from the same line in the future — just the contrary to the situation with the gate. Which of them will prevail will be decided by the other training examples.

The next step is to calculate the posterior probabilities h_1 and h_2:

$$h_1 = 0.352426 \qquad h_2 = 0.647574$$

Now the updates for the parameter vectors are done, with a fixed learning rate $\rho = 0.2$:

$$\Delta U_1 = \begin{bmatrix} -0.047543 \\ 0.088833 \end{bmatrix} \qquad \Delta U_2 = \begin{bmatrix} 0.021091 \\ -0.039408 \end{bmatrix}$$

$$\Delta V_1 = \begin{bmatrix} 0.038098 \\ -0.071186 \end{bmatrix} \qquad \Delta V_2 = \begin{bmatrix} -0.038098 \\ 0.071186 \end{bmatrix}$$

Note that the updates for the parameter vectors of the gate have the same magnitude but opposite signs. This completes the first example of the first iteration of the algorithm.

4.3.2 Iteration 1001

After 1000 iterations, the algorithm has nearly converged:

$$U_1 = \begin{bmatrix} -1.0283 \\ -0.0159237 \end{bmatrix} \qquad U_2 = \begin{bmatrix} 1.9906 \\ -0.992945 \end{bmatrix}$$

$$V_1 = \begin{bmatrix} -16.8574 \\ -0.17304 \end{bmatrix} \qquad V_2 = \begin{bmatrix} 16.0895 \\ 0.586863 \end{bmatrix}$$

Clearly the first expert gets to repressent the second line, and vice versa. Note that the large first elements of the parameter vectors of the gates lead to a split very close to $x = 0$, as it is supposed to be. The exact place of the split x_s (where $g_1 = g_2$) is

$$x_s = \frac{0.586863 - 0.17304}{16.0895 + 16.8574} = 0.01256$$

The split is also very sharp. The algorithm proceeds as previously, this time giving

$$\xi_1 = 9.194901 \quad \xi_2 = -8.024028$$

$$g_1 = 0.999999 \quad g_2 = 0.000001$$

As noted, the first expert gets to represent this example. This is also evident from the "opinions" $\hat{y}_1$ and $\hat{y}_2$ of the two experts, as well as from the probabilities P_1 and P_2 that the real output y has been produced by each of the experts:

$$\hat{y}_1 = 0.534409 \quad \hat{y}_2 = -2.058288$$

$$P_1 = 0.999999 \quad P_2 = 0.001199$$

The posterior probabilities h_1 and h_2 happen to be exactly the same as the priors g_1 and g_2:

$$h_1 = 0.999999 \qquad h_2 = 0.000001$$

Consequently, no learning takes place for the gate, and only minimal learning takes place for the first expert:

$$\Delta U_1 = \begin{bmatrix} -0.000083 \\ 0.000156 \end{bmatrix} \qquad \Delta U_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Delta V_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \Delta V_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

These updates and the attained values of the parameters demonstrate that the algorithm has practically converged.

4.4 Approximation of Highly Nonlinear Functions

For comparison reasons, the HME model was tested on the nonlinear function described in Chapter 3. Unlike RBFs, an HME model has two free parameters — the number L of levels of the tree, and the learning speed ρ. Since thorough scanning of these two parameters was not feasible, tests were performed for $L = 3$ and $\rho = 0.2$, over 8000 iterations. Figures 8, 9, and 10 show the learning curves for the three cases described in Chapter 3.

Figure 8 shows that in the absence of noise both in- and out-of-sample error decrease slowly as learning proceeds. Good generalization is evident.

In the presence of substantial noise, as is the case with Figure 9, some learning and generalization still occurs. If the noise is as high as the one in Figure 10, however, neither the in- nor out-of-sample errors can be diminished through training. This establishes some practical limits for noise tolerance of HME.

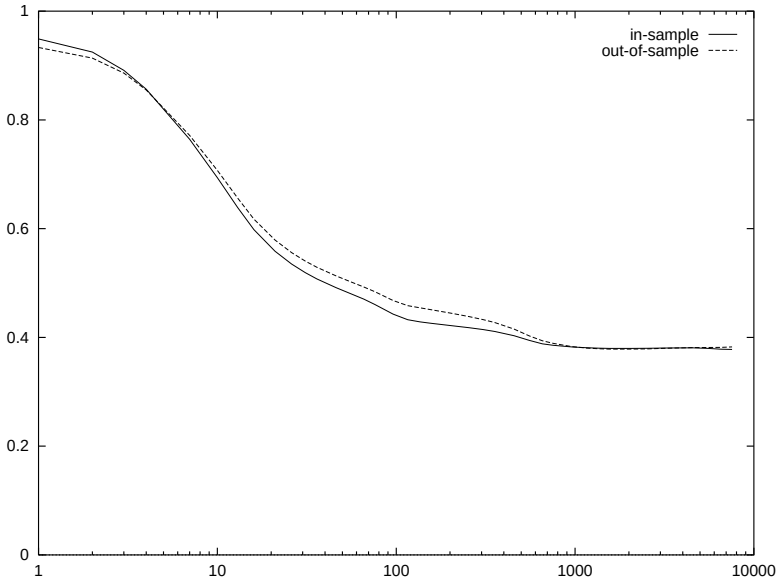

Figure 8. HME approximation, $\sigma = 0$, $S = 0.326$

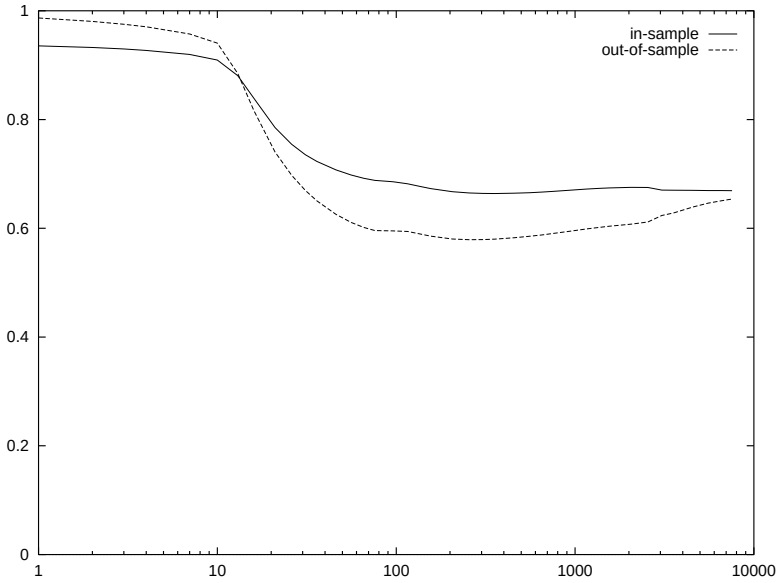

Figure 9. HME approximation, $\sigma = 0.1$, $S = 0.330$

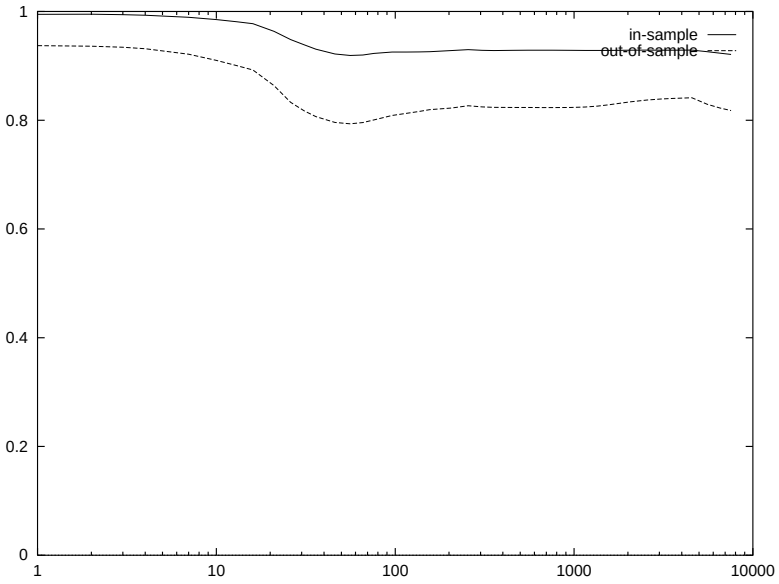

Figure 10. HME approximation, $\sigma = 0.2$, $S = 0.373$

CHAPTER 5

APPLICATIONS

The experiments in chapters 3 and 4 show that both RBF and HME are good nonlinear approximators that can learn complex multidimensional nonlinear mappings from examples. How they will perform on time series prediction is, however, another question. To test it, a number of experiments were performed with time series of various types. The results of these experiments are reported in this chapter.

Three types of time series were tested:

1. Computer generated chaotic data. The logistic map was used (equation 1.11), with $\lambda = 2.20539$.

2. Data from a physical experiment with $NH_3 - FIR$ (methyl far-infrared) laser.

3. Stock market data. Daily quotes of 4 companies over 370 days, as well as monthly quotes of one more company over 240 days were tested.

5.1 Programming Environment

The Ts/tools software package for time series analysis from the Santa Fe Institute for the Studies of Complexity was used as a basic tool. The package was modified and extended with utilities for recursive least squares estimation and hierarchical mixture of experts.

The package runs in UNIX environment. It consists of utilities that can be either included in shell scripts or used interactively through a character user interface.

A number of auxilliary utilities were developed too, whose purpose was to generate and preprocess test data, and compute various statistics of the time series.

5.2 Learning the Logistic Map

The logistic map is a good test case to compare the performance of RBF and HME. The predictors have two inputs, and the nonlinear mapping can be visualized easily (Figure 11):

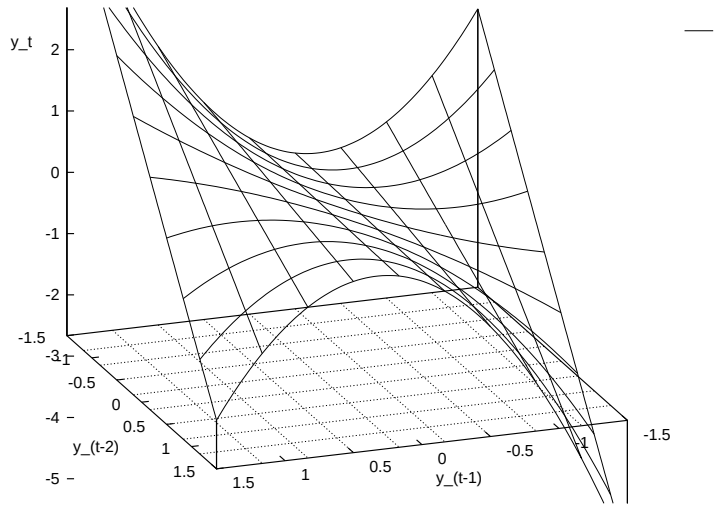

Figure 11. Logistic map surface.

It is clear from the figure that the surface is highly nonlinear and a single plane cannot approximate it well, no matter what the orientation of the plane is.

A particular realization of the iterated logistic map does not traverse all the surface, but only a trajectory on it. However, since the dynamic system is chaotic, each point of the surface has a nonzero probability to be visited sooner or later. The case for $\lambda = 2.20539$ was tested over 512 iterations, with initial conditions $y^{(0)} = 0.529$ and $y^{(1)} = 0.92$ (Figure 12).

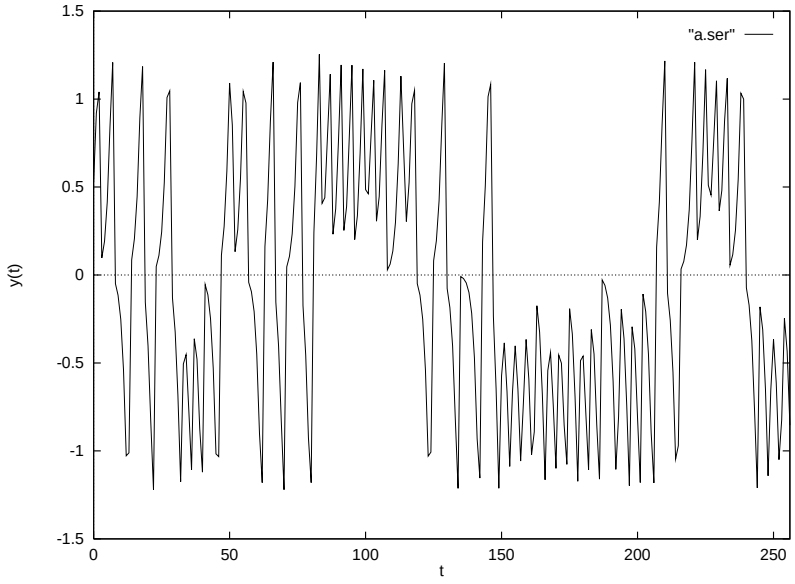

Figure 12. Realization of the logistic iterated map $y^{(t)} = 2.20539 y^{(t-1)}(1 - y^{(t-2)})$

The first 250 points were used for training, and the next 250 were used for testing. The embedding dimension was set to 4. The in- and out-of-sample error were measured for both RBF and HME approximation.

For the case of RBF, the only free parameter of the architecture is the number of knots. These were varied from 10 to 250, and the resulting in- and out-of-sample errors are plotted in Figure 13. The plot shows that even only 10 knots reduce the error in half, and further increasing of the number of knots results in gradual vanishing of the in-sample error (as is expected from an interpolation scheme such as RBF). What is very important, however, is that the out-of-sample error follows closely the in-sample error. This fact demonstrates excellent generalization. Of course, the out-of-sample error does not vanish — it stays at a value of about 6%, which can be considered the methodical error of RBF in this application.

For the case of HME approximation, three parameters exist — the learning rate ρ, the number of iterations, and the number of experts in the hierarchy. In

order to produce an illustrative plot, only one of these parameters, the number of experts was varied and the in- and out-of-sample errors were measured. The results are shown in Figure 14. In this figure all errors are measured after $10,000$ iterations of the gradient ascent learning rule with $\rho = 0.1$, with the exception of the case with 8 experts, which diverged after 2556 iterations. The reported errors for that case are at iteration number 2556. Note that this arrangement does not prevent overfitting from happening.

In fact, the figure shows that overfitting does indeed happen: while the in-sample error decreases slowly with increasing the number of experts, the out-of-sample error does not follow the same tendency. Rather, it tends to oscillate around 50%. Nevertheless, it is half the error that would have resulted from outputting the average of the time series.

It should be noted that Figure 14 should not be taken as indication that HME does not approximate well. The purpose of this figure is to demonstrate what level of approximation can be achieved for a fixed time (10000 iteratons in this case) with various number of experts in HME. It is quite conceivable that a much better approximation is attainable if the space of learning rates is scanned and the progress of the learning is monitored so that it can be stopped before overfitting occurs.

5.3 Prediction of $NH_3 - FIR$ Laser Output

The second test case was chosen to be a real physical system — a $NH_3 - FIR$ laser [19]. Part of the time series is shown in Figure 15. As it is visible, the output is a train of pulses with varying amplitude. Based on the provided time series data ([12]), one training and nine testing data sets were prepared. The training set had 992 examples, and the testing sets had 1000 examples each.

This problem is harder than the prediction of synthetic time series, because the output is susceptible to measurement error that obscures the internal

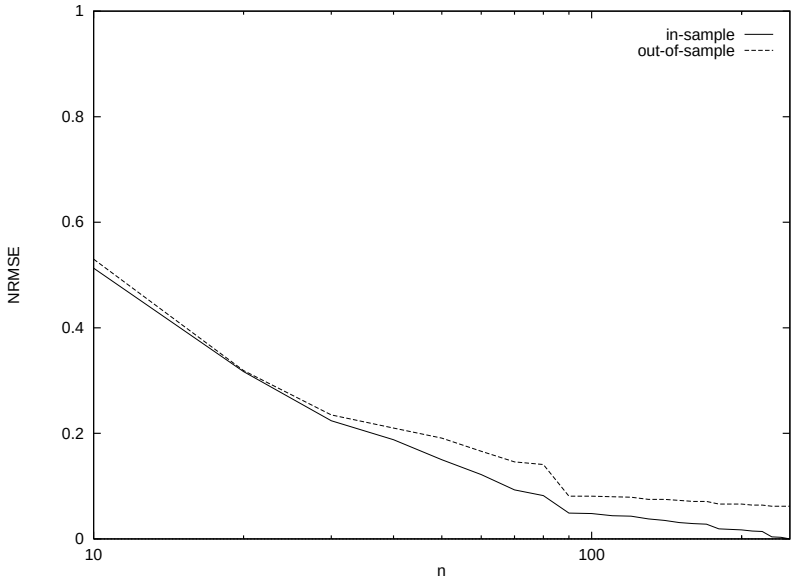

Figure 13. Logistic map: prediction error vs. number of RBFs.

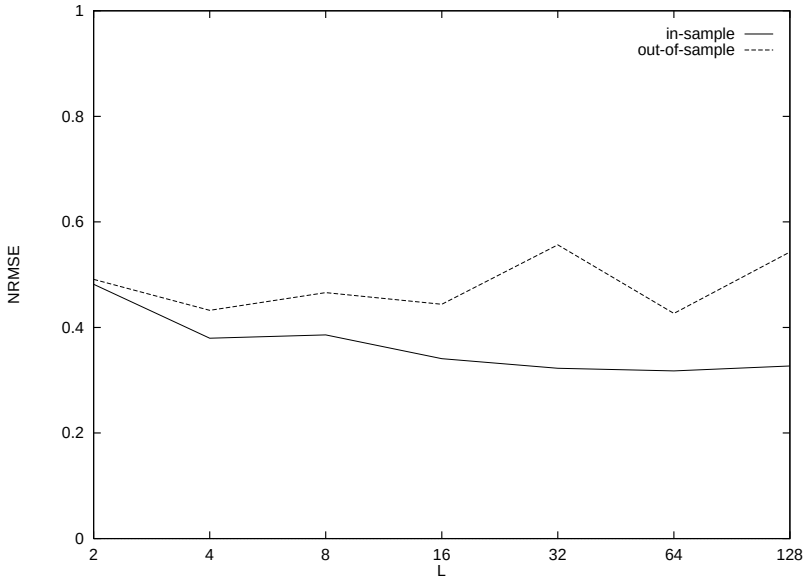

Figure 14. Logistic map: prediction error vs. number of experts in HME.

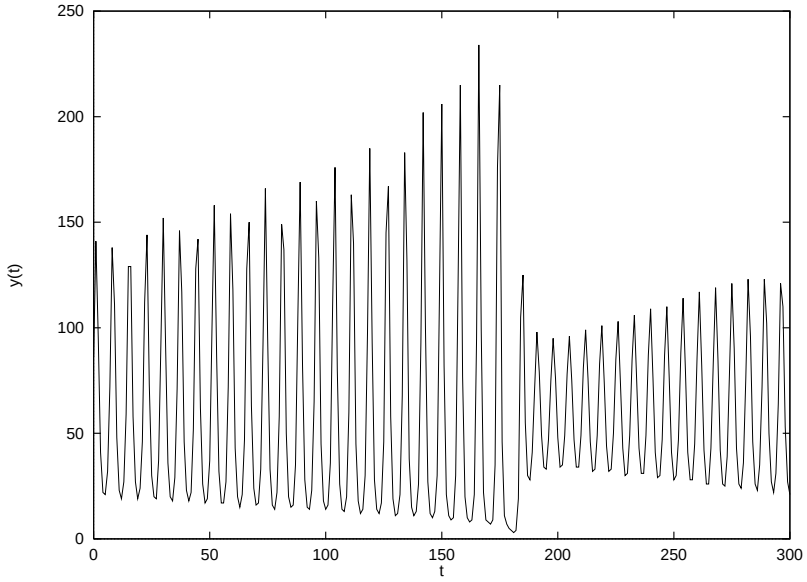

Figure 15. Output of a NH_3 far-infrared laser.

dynamics of the system. However, the dynamics equations stay invariant over time, and a good learning method should be able to extract them from sufficiently many examples.

Determining a suitable value for the embedding dimension m is not trivial. For the experiments reported below, embedding dimension equal to 8 was used, for a number of reasons. First, it has been demonstrated that the time series exibits chaotic behavior similar to that of a Lorenz system that has an attractor with a dimension less than 3 ([19]). Thus it follows from the theorem of Takens that an embedding dimension of 7 or more is sufficient. Second, Sauer was able to predict well this time series using a locally linear model with embedding dimension 8 [12].

As can be seen from Figure 16, the laser output is a form of oscillation, and it can be expected that a linear predictor might be at least partly successful in one-step predictions. Thus, the two nonlinear methods under consideration (RBF and HME) should be compared with linear prediction and not with simply the average of the time series.

Expressed as a statistical formalism, this means that we will have as our null

hypothesis the statement that the generalization power (the out-of-sample error) of each of the nonlinear methods is equal to the generalization power of linear prediction. If we are able to disprove this hypothesis, we will have proven the converse hypothesis, namely that the nonlinear methods are better (or worse, but anyway different in performance) than the linear predictor. To this end, we need a way to establish the variance of the generalization errors of the linear and nonlinear predictors. Such an estimate can be obtained by evaluating the methods on a number of testing sets instead of on a single one.

For the laser output, models for RLS, RBF, and HME were learned from the same sequence of 992 training examples. The linear model (RLS) is unique under this procedure. In order to guarantee that the algorithm has surely converged, 10 iterations through the training set were performed.

However, different models can be obtained with RBF (by varying the number of knots) and with HME (by varying the number of experts.) For these two methods, cross-validation was performed with the first of the nine test sets, as follows. The model was learned with the examples in the training set, and the out-of-sample error on testing set 1 was monitored. The model that performed best was selected for further estimation of the out-of-sample error on the remaining eight testing sets.

The best RBF model was attained with 990 RBFs (2 less the number of examples). The best HME model had 7 levels (128 experts) and was obtained after 10000 iterations with learning rate $\rho = 0.01$.

The average approximation error of RLS, RBF, and HME over the eight testing sets is shown in Table 5.1. Since the errors in Table 5.1 are normalized, they have the meaning of percentages of the standard deviation of the time series. Thus RLS diminishes by training the NRMS in-sample error to 43%, HME to 7%, and RBF to even less than 1%. It is obvious that the data is highly nonlinear and

the classical linear regression performs poorly on this time series, while the nonlinear function approximators are much better.

	RLS	RBF	HME
In-sample	0.43309	0.00247	0.07028
Out-of-sample ($\bar{E}$)	0.44586	0.10212	0.10414

Table 5.1. NRMSE of RLS, RBF, and HME on the laser output time series.

In order to check if the three methods have indeed different approximation abilities, statistical analysis has to be done. The comparisons were performed based on the difference in approximation error on the eight testing sets. Each of the three methods (RLS, RBF, and HME) were tested on each of the eight testing sets. This presents a paired statistical analysis, since the same sample was used for all methods. In order to compare any two methods, for example RLS and HME, the difference in performance d_i is computed for each of the eight testing sets ([27]):

$$d_i = E_i^{HME} - E_i^{RLS} \quad i = 1, 8 \tag{5.1}$$

where E_i^{HME} and E_i^{RLS} are the approximation errors on the i-th testing set of HME and RLS respectively.

Then the average difference $\bar{d}$ and the sample standard deviation $\hat{s}$ of the differences were computed as follows:

$$\bar{d} = \frac{\sum_{i=1}^{8} d_i}{8} \tag{5.2}$$

$$\hat{s} = \sqrt{\frac{\sum_{i=1}^{8} (d_i - \bar{d})^2}{7}} \tag{5.3}$$

The paired t statistic was computed as

$$t = \frac{\bar{d}}{\hat{s}/\sqrt{7}}. \tag{5.4}$$

The t statistic and the attained significance level α of the differences between the three methods for a two-tailed test is shown in Table 5.2:

	t	α
RBF vs. HME	0.355286	0.440666
RBF vs. RLS	42.07407	$< 10^{-11}$
HME vs. RLS	42.73772	$< 10^{-11}$

Table 5.2. Comparison of RBF, HME, and RLS: t-statistic and attained significance level α.

The attained significance levels are calculated as

$$\alpha = P(T > \frac{t_\alpha}{2}), \tag{5.5}$$

where T is a random variable following Student's t-distribution.

Table 5.2 shows that both RBF and HME are definitely better than RLS. As for the differences between RBF and HME, RBF seems to outperform HME, though at a level that is not statistically significant. Thus we assume that there is no statistical evidence that either RBF or HME is better than the other method.

It should be noted that all three methods generalize well — their out-of-sample error is not substantially higher than their in-sample error. What makes them different in performance is the extent to which the in-sample error can be reduced by means of training on the examples in the training set. The nonlinear approximation performed by HME and RBF seem to be the most important reason for their better performance.

5.4 Stock Market Prediction

The stock market is another area where the application of nonlinear prediction methods seems promising. Financial time series, as any other time

series, are generated by a corresponding stochastic nonlinear dynamic system, which can be modeled in principle with a set of stochastic nonlinear difference equations. However, while the dynamics of the laser, analyzed in the previous section, are time-independent, it is very plausible to expect that the dynamics of the stock market shift quickly as new financial players make their transactions on the market.

In addition to that, many random factors exert their influence on the market, making the discovering of the underlying dynamics even more difficult. These factors are represented as noise in the set of stochastic difference equations. In fact, there is no evidence that disproves the conjecture that the fluctuations in the stock market are simply driven by random noise and no meaningful dynamics are present.

This last hypothesis is known as the Efficient Market, or Random Walk Hypothesis [7]. This hypothesis formally means that the next value of a stock price is its current value plus a displacement that is a zero-mean normally distributed random variable, uncorrelated with the displacements at previous moments in time. If this hypothesis is true, the best prediction for the next value is simply the current value, i.e., no profit can be made by trying to predict the stock market.

If, however, the displacement is correlated with previous displacements, then it should be predictable. In order to put this idea in the time series analysis framework described in Chapter 1, a new series should be generated from the time course of a stock price. A common practice in financial modeling is to take the ratio of successive values of the stock price rather than their difference:

$$R_t = \frac{p_t}{p_{t-1}} \tag{5.6}$$

This ratio can be further normalized by either subtracting 1, or taking its logarithm. In the latter case, the obtained value is called the continuously compounded return of the stock. In the experiments reported in this section, the first approach was used instead:

$$r_t = R_t - 1 \tag{5.7}$$

For the purposes of this description, we will call the variable r_t the return of the stock. The rationale behind working with returns rather than the absolute values of the stock is the hypothesis that is contrary to the Efficient Market Hypothesis — namely, that the future displacements are correlated with previous displacements, and not with the absolute level of the stock. For example, if a decrease by 1% is always followed by an increase of 3%, the absolute level of the stock does not matter; if a learning method is able to discover this fact, every time in the future when the stock drops 1%, a trading strategy based on the findings of the learning method will manage to realize a profit of 3%, no matter at what level the stock stands. Of course, this example is oversimplified and it can be expected that if correlations between displacements exist, their nature will be much more involved.

As a first test series, 240 monthly prices of the stock of IBM were tested. The time series is shown in Figure 16, and the returns calculated by formula (5.2) are shown in Figure 17.

The embedding dimension was set to 8 following the results of Hall, who reported that neural networks having between 4 and 8 inputs were very successful at predicting monthly and weekly stock prices [16].

This choice of embedding dimension results in an embedded time series of 232 elements, 192 of which were used for training, and 40 for testing. However, in order to guarantee that the approximator uses all information available prior to

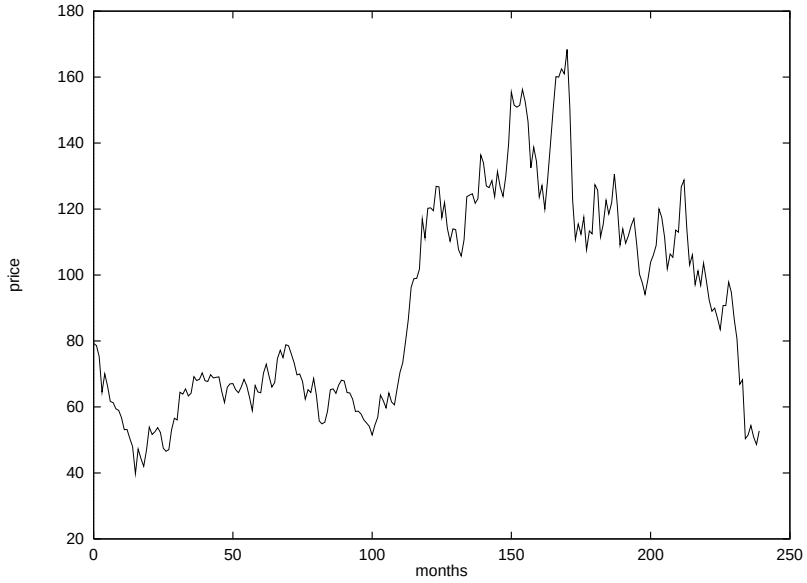

Figure 16. Monthly prices of IBM stock over 240 months.

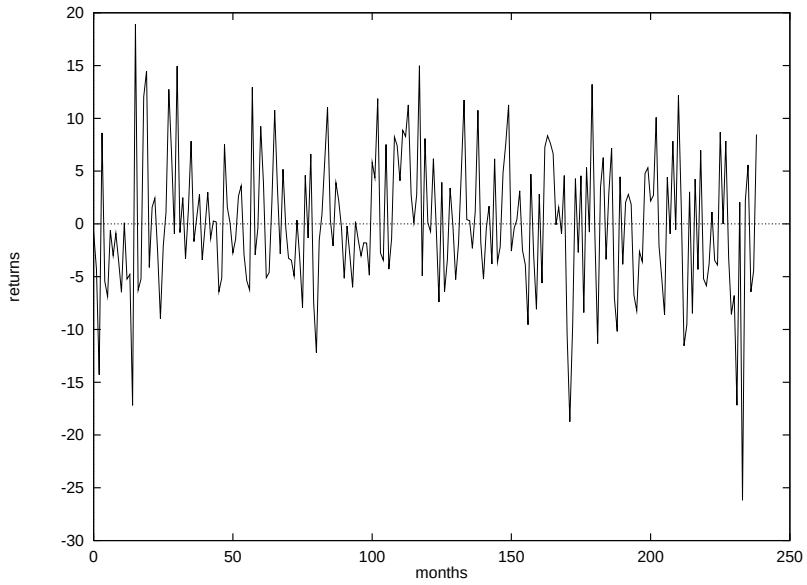

Figure 17. Monthly returns of IBM stock over 240 months.

the moment of prediction, the training set was expanded with new elements as the prediction process proceeded. For example, for the prediction of the 193-th monthly return, the approximator was trained with the 192 returns from the first to the 192-th. For the prediction of the next, 194-th return, the 193-th one was added to the training set and the earliest one (the first) was discarded from the training set. Analogously, for the prediction of the 195-th return, the 194-th was added and the second one discarded, and so on, keeping the number of training examples equal to 192.

Figures 18, 19, and 20 show the real and predicted stock returns with 10, 50, and 100 RBFs respectively. Seemingly the prediction improves with increasing the number of RBFs — at least the predicted curve starts to resemble the real one. However, a look at Table 5.3 can convince us that the chances to make money with an RBF predictor actually worsen with increasing the number of RBFs. None of the out-of-sample errors in Table 5.3 manages to go below 1.0, which is the prediction error of a random-walk strategy (that is, assuming that the price stays the same, or the return r_t is 0). The table suggests that even though the in-sample error can be brought to a value much below 1.0, the resulting model cannot generalize well over new data, and its prediction has performance worse than that of random walk.

	Average in-sample	Average out-of-sample
10 RBFs	0.902400	1.299235
50 RBFs	0.7768399	1.790849
100 RBFs	0.612064	1.722756

Table 5.3. Average NRMSE of RBF over 40 predictions, with retraining

It might be argued that the reason for RBF's failure to predict well the IBM stock data is due to a wrong time scale. Indeed, it could be expected that the dynamics of the underlying state-space system, if any, are very much likely to

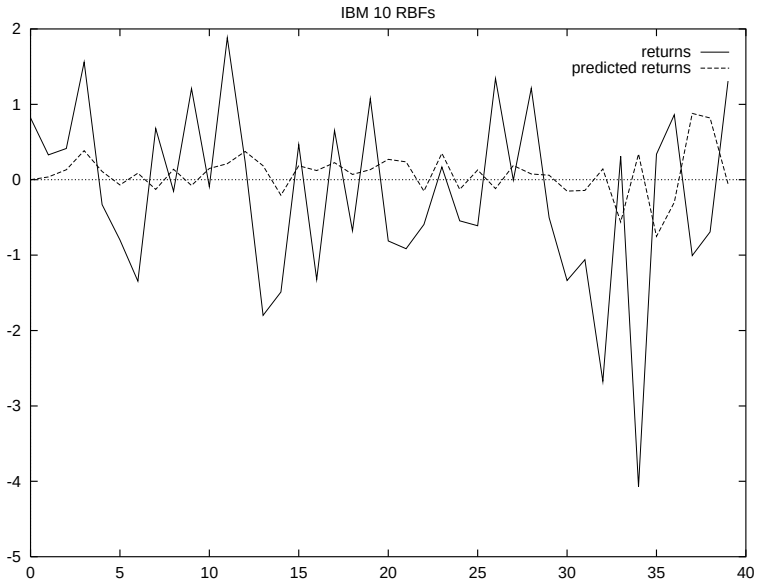

Figure 18. Actual and predicted values of IBM stock returns over 40 months. Approximation with 10 RBFs.

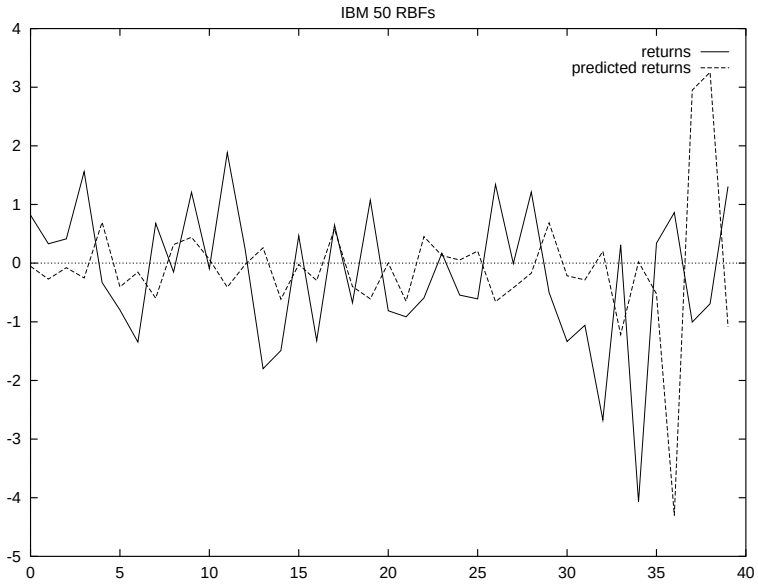

Figure 19. Actual and predicted values of IBM stock returns over 40 months. Approximation with 50 RBFs.

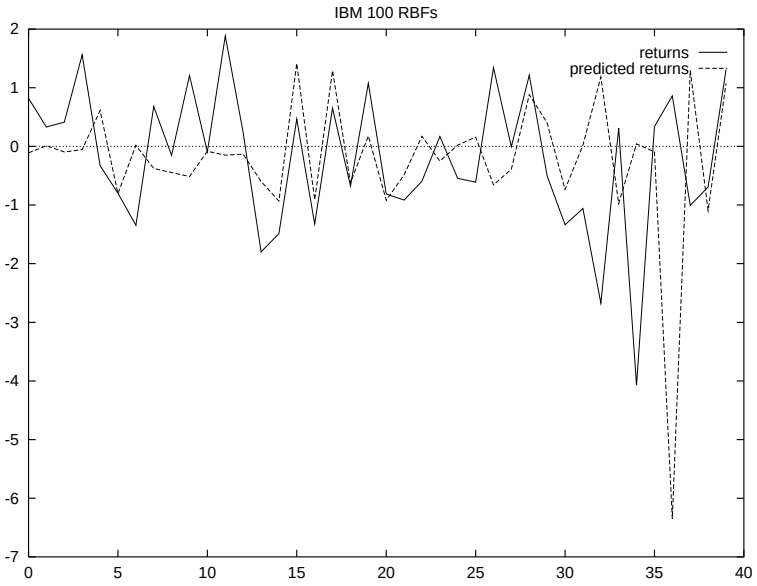

Figure 20. Actual and predicted values of IBM stock returns over 40 months. Approximation with 100 RBFs.

change over the months and years that the experimental time series of IBM stock comprises.

In an attempt to eliminate this obvious dependency on time scale, another set of experiments was performed with daily instead of monthly prices. Five companies were selected, with well known blue-chip companies as well as speculative stock forming the selection. The training and testing sets consisted of 183 points each, with embedding dimension 8. As in the experiment with IBM stock, the returns r_t were used rather than the absolute value of the stock. Table 5.4 shows the results from prediction with RBF and HME. For RBF, 10 values of the number of RBFs were tried, and Table 3 reports the best and worst out-of-sample results. For HME, 32 experts and learning speed $\rho = 0.1$ was only used, and hence a single out-of-sample error is reported, the one which resulted from the HME model after 7536 iterations through the training set.

As can be seen from Table 5.4, no predictor could beat the random walk strategy, and unfortunately we have to conclude that the daily stock prices seem

Company	Best RBF	Worst RBF	HME
Intergraph Corp.	1.035	1.115	1.152
Coca-Cola	1.034	1.187	1.103
Navistar Intl. Corp.	1.048	1.168	1.245
Motorola	1.007	1.185	1.037

Table 5.4. Out-of-sample NRMSE of RBF and HME on daily stock prices.

to be unpredictable by either learning methods.

CHAPTER 6

CONCLUSION

The application of adaptive computation learning methods to time series prediction has been explored, and a number of experiments on time series of practical significance have been performed. Two learning methods have been tested for state-space reconstruction of the dynamics of nonlinear systems — approximation via Radial Basis Functions and via Hierarchical Mixtures of Experts.

The experiments have shown that both methods can be successfully used for one-step prediction of chaotic nonlinear time series generated by time-invariant dynamic systems. Furthermore, both methods have had better performance on the task of prediction of laser output than traditional linear regression.

However, neither of the methods was successful at predicting very noisy and possibly time-variant systems such as the stock market. Although both monthly and daily time scales were used, the approximation methods could not disprove the random walk hypothesis. This certainly does not prove that the stock market is unpredictable — it only indicates that the two nonlinear learning methods under consideration cannot predict it. The possible reasons are the swiftly changing dynamics of the stock market, and the requirement for many training examples that the two learning methods (and all other presently known learning methods) have.

The obtained results suggest that the requirement for time-invariant systems is crucial to the success of their prediction, since it has been possible to reconstruct well the dynamics of time-invariant systems, but the learning methods

have failed to predict time-variant ones. However, the performed experiments demonstrate that at least one of the simplifications, traditionally made to render time series analysis tractable, need not restrain researchers anymore — nonlinear systems can be modeled as reliably as linear ones have been modeled so far. The other major simplification, that of time-invariance, cannot be relaxed yet. It can be conjectured that learning methods that can learn from much fewer examples than current learning methods might provide better tools for state-space reconstruction of nonlinear time-variant systems.

BIBLIOGRAPHY

[1] Bauer, R.J., Jr. (1994). Genetic Algorithms and Investment Strategies. New York: Wiley.

[2] Box, G.E.P., and G.M. Jenkins (1970). Time Series Analysis: Forecasting and Control. San Francisco: Holden-Day.

[3] Brock, W. A. (1988). Nonlinearity and complex dynamics in economics and finance, in Andersen, P. W., K.J. Arrow, and D. Pines, (Eds.) The Economy as an Evolving Complex System, Reading, MA: Addison-Wesley, 77-97.

[4] Casdagli, M. (1989). Nonlinear prediction of chaotic time series. Physica D, vol. 35, 335-356.

[5] Casdagli, M., and S. Eubank (Eds.) (1992). Nonlinear Modeling and Forecasting. Reading, MA: Addidson-Wesley.

[6] Colin, A.M. (1994). Genetic algorithms for financial modeling, In Deboeck, G.J. (Ed.) (1994). Trading on the Edge. New York: Wiley., 148-174.

[7] Deboeck, G.J. (Ed.) (1994). Trading on the Edge. New York: Wiley..

[8] Deboeck. G. J., and M. Cader (1994). Pre- and postprocessing of financial data, In Deboeck, G.J. (Ed.) (1994). Trading on the Edge. New York: Wiley., 27-44.

[9] Elbert, Theodore (1984). Estimation and Control of Systems. New York: Van Nostrand Reinhold Company.

[10] Embrechts, M. (1994). Basic Concepts of Nonlinear Dynamics and Chaos Theory, In Deboeck, G.J. (Ed.) (1994). Trading on the Edge. New York: Wiley., 265-279.

[11] Frison, T. (1994). Nonlinear data analysis techniques. In Deboeck, G.J. (Ed.) (1994). Trading on the Edge. New York: Wiley., 280-296.

[12] Gershenfeld, N.A., and A.S. Weigend (1994). The Future of Time Series: Learning and Understanding. In Weigend, A.S, and N.A. Gershenfeld (Eds.) (1994) Time Series Prediction. Reading, MA: Addison-Wesley, 1-70.

[13] Girosi, F., and T. Poggio (1989). Networks and the Best Approximation Property. AI Memo 1164, MIT AI Lab. Available at ftp://ftp.ai.mit.edu/ai-publications/1000-1499/AIM-1164.ps.Z

[14] Granger, C.W.J., and P. Newbold (1986). Forecasting Economic Time Series. Academic Press.

[15] Green, H., and M.Pearson (1994). Neural nets for foreign exchange trading. In Deboeck, G.J. (Ed.) (1994). Trading on the Edge. New York: Wiley., pp. 123-129.

[16] Hall, J. W. (1994). Adaptive selection of US stocks with neural nets, In Deboeck, G.J. (Ed.) (1994). Trading on the Edge. New York: Wiley., 45-65.

[17] Harvey, A.C. (1993). Time Series Models. Hemel Hempstead, UK: Harvester Wheatsheaf.

[18] Hertz, J.A., Anders S. Krogh, and Richard G. Palmer (1991). Introduction to the Theory of Neural Computation, Santa Fe Institute Studies in the Sciences of Complexity, Reading, MA: Addison-Wesley.

[19] Hübner, U., C.-O. Weiss, N.B. Abraham, and D. Tang (1994). Lorenz-like chaos in $NH_3 - FIR$ lasers, In Weigend, A.S, and N.A. Gershenfeld (Eds.) (1994) Time Series Prediction. Reading, MA: Addison-Wesley, 73-104.

[20] Hutchinson, J.M. (1994). A Radial Basis Function Approach to Financial Time Series Analysis. Ph.D. Dissertation, Department of Computer Science and Electrical Engineering, Massachusetts Institute of Technology.

[21] IBM (1994). IBM Microelectronics ZISC zero instruction set computer: preliminary information, Supplement to Proc. of World Congress on Neural Networks, INNS Press.

[22] Jordan, M.I., and Jacobs, R.A. (1993). Hierarchical Mixtures of Experts and the EM Algorithm. AI Memo 1440, MIT AI Lab. Available at ftp://ftp.ai.mit.edu/ai-publications/1000-1499/AIM-1440.ps.Z

[23] Jordan, M.I., and L. Xu (1993). Convergence results for the EM approach to mixtures of experts architectures. AI Memo 1458, MIT AI Lab. Available at ftp://ftp.ai.mit.edu/ai-publications/1000-1499/AIM-1458.ps.Z

[24] Jordan, M.I., and Jacobs, R.A. (1994). Hierarchical mixtures of experts and the EM algorithm. Neural Computation, vol. 6, 181-214.

[25] Koza, J. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA: MIT Press.

[26] Ljung, L., and T. Söderström (1983). Theory and Practice of Recursive Identification. Cambridge, MA: MIT Press.

[27] Mendenhall, W., D.D. Wackerly, and R.L. Scheaffer (1990). Mathematical Statistics with Applications. Boston: PWS-KENT.

[28] Poggio, T., and F. Girosi (1989). A Theory of Networks for Approximation and Learning. AI Memo 1140, MIT AI Lab. Available at ftp://ftp.ai.mit.edu/ai-publications/1000-1499/AIM-1140.ps.Z

[29] Poggio, T., and F.Girosi (1990). Regularization algorithms for learning that are equivalent to multilayer networks. Science, vol. 247, 978-982.

[30] Press, W.H et al. (1992). Numerical Recipes in C, Cambridge University Press.

[31] David Rumelhart, Jay MacClelland, (eds.) (1986), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1,2, Cambridge, MA: MIT Press.

[32] Sauer, T. (1994). Time series prediction by using delay coordinate embedding. In Weigend, A.S, and N.A. Gershenfeld (Eds.) (1994) Time Series Prediction. Reading, MA: Addison-Wesley, 175-193.

[33] Takens, F. (1981). Detecting strange attractors in turbulence. In D.A. Rand and L.-S. Young (eds.), Dynamical Systems and Turbulence, Lecture Notes in Mathematics, Vol. 898, 336-381. Berlin: Springer.

[34] Theiler, J. (1987). Efficient algorithm for estimating the correlation dimension from a set of discrete points. Physical Review A, vol. 36, 4456-4462.

[35] Wan, E.A. (1994). Time series prediction by using a connectionist network with internal delay lines. In Weigend, A.S, and N.A. Gershenfeld (Eds.) (1994) Time Series Prediction. Reading, MA: Addison-Wesley, 195-217.

[36] Waterhouse, S.R., and A.J. Robinson (1995). Non-linear prediction of acoustic vectors using hierarchical mixtures of experts, in G. Tesauro, D.S. Touretzky, and T.K. Leen, (Eds.) Neural Information Processing Systems 7, Cambridge, Massachusetts: MIT Press.

[37] Weigend, A.S, and N.A. Gershenfeld (Eds.) (1994) Time Series Prediction. Reading, MA: Addison-Wesley

[38] White, A. (1993). Economic prediction using neural networks: the case of IBM daily stock returns. In R.R. Trippi and E. Turban (Eds.), Neural Networks in Finance and Investing, Chicago: Phobus.

[39] Yoda, M. (1994). Predicting the Tokyo stock market. In Deboeck, G.J. (Ed.) (1994). Trading on the Edge. New York: Wiley., 66-79.

VITA

Graduate School

Southern Illinois University

Daniel Nikolaev Nikovski Date of Birth: April 13, 1969

606 West College Street, Apt.4, Rm. 6, Carbondale, Illinois 62901

150 Hristo Botev Boulevard, Apt. 54, 4004 Plovdiv, Bulgaria

Technical University - Sofia, Bulgaria

Engineer of Computer Systems and Control

Thesis Title:

Adaptive Computation Techniques for Time Series Analysis

Major Professor: Dr. Mehdi Zargham