A Radial Basis Function Approach to Financial Time Series Analysis

by

James M. Hutchinson

Master of Science in EECS, Massachusetts Institute of Technology (1986)

Submitted to the

Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

at the

Massachusetts Institute of Technology

February, 1994

Signature of Author____

James M. Hutchinson
Department of Electrical Engineering and Computer Science
Email: hutch@ai.mit.edu

Certified by____

Professor Tomaso Poggio
Artificial Intelligence Laboratory
Thesis Co-Supervisor

Certified by____

Professor Andrew Lo
Sloan School of Management
Thesis Co-Supervisor

Accepted by____

Professor Frederic R. Morgenthaler
Chair, Department Committee on Graduate Students

A Radial Basis Function Approach to Financial Time Series Analysis

by

James M. Hutchinson

Submitted to the Department of Electrical Engineering and Computer Science on December 1, 1993 in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Abstract

Billions of dollars flow through the world's financial markets every day, and market participants are understandably eager to accurately price financial instruments and understand relationships involving them. Nonlinear multivariate statistical modeling on fast computers offers the potential to capture more of the underlying dynamics of these high dimensional, noisy systems than traditional models while at the same time making fewer restrictive assumptions about them. For this style of exploratory, nonparametric modeling to be useful, however, care must be taken in fundamental estimation and confidence issues, especially concerns deriving from limited sample sizes. This thesis presents a collection of practical techniques to address these issues for a modeling methodology, Radial Basis Function networks. These techniques include efficient methods for parameter estimation and pruning, including a heuristic for setting good initial parameter values, a pointwise prediction error estimator for kernel type RBF networks, and a methodology for controlling the "data mining" problem. Novel applications in the finance area are described, including the derivation of customized, adaptive option pricing formulas that can distill information about the associated time varying systems that may not be readily captured by theoretical models. A second application area is stock price prediction, where models are found with lower out-of-sample error and better "paper trading" profitability than that of simpler linear and/or univariate models, although their true economic significance for real life trading is questionable. Finally, a case is made for fast computer implementations of these ideas to facilitate the necessary model searching and confidence testing, and related implementation issues are discussed.

Thesis Committee: Prof. Tomaso Poggio, Prof. Andrew Lo, Prof. Patrick Winston, Prof. Tomas Lozano-Perez

Acknowledgements

First and foremost I would like to thank Tomaso Poggio, who has been a continual source of ideas and support throughout my program. I would also like to thank Andrew Lo, who helped me to believe that finance was an interesting area in which to apply these ideas, and the rest of my thesis committee, Tomas Lozano-Perez and Patrick Winston, for their helpful comments. My work benefited greatly from a friendly and productive relationship with Strategic Products Group of Citicorp in London, especially due to the insightful comments and thoughtful feedback of Andrew Dyson. I am also grateful to Federico Girosi for clarifying my thinking on many technical issues and patiently guiding me through the math in Appendix A; to Xiru Zhang for keeping me in touch with practical considerations and for writing the first version of the Connection Machine RBF software in Chapter 6; to Anand Bodapati for alerting me to the strategic issues in statistical modeling; to my officemates David Beymer and Brian Subirana for countless discussions and diversions; and to the members of the AI lab and Thinking Machines Corporation for the many open and intelligent discussions that research depends upon. Finally I would like to thank Anne Fricker for keeping me happy and sane throughout the process, my father for teaching me early on to love learning, and my mother, whose enthusiasm and support helped me to reach this point.

*

This thesis describes research done at the Massachusetts Institute of Technology Artificial Intelligence Laboratory, where the author was supported by an ARPA AASERT grant administered under the Office of Naval Research contract N00014-92-J-1879. Additional support was provided by the Office of Naval Research under contract N00014-93-1-0385, by a grant from the National Science Foundation under contract ASC-9217041 (this award includes funds from ARPA provided under the HPCC program), and by Siemens AG. Support for the A.I. Laboratory's artificial intelligence research was provided by ARPA contract N00014-91-J-4038. The author would also like to acknowledge the extracurricular support of Thinking Machines Corporation and the Strategic Products Group of Citicorp London.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Financial markets are incredible systems. Thousands of instruments are traded by millions of participants every day, around the world, in a never ending battle to make money. Is it possible to capture the workings of the the markets in a mathematical model? Is it possible to find neglected areas of the markets where a careful application of statistics might reveal persistent systematic price discrepancies? Is it possible to predict the stock market?

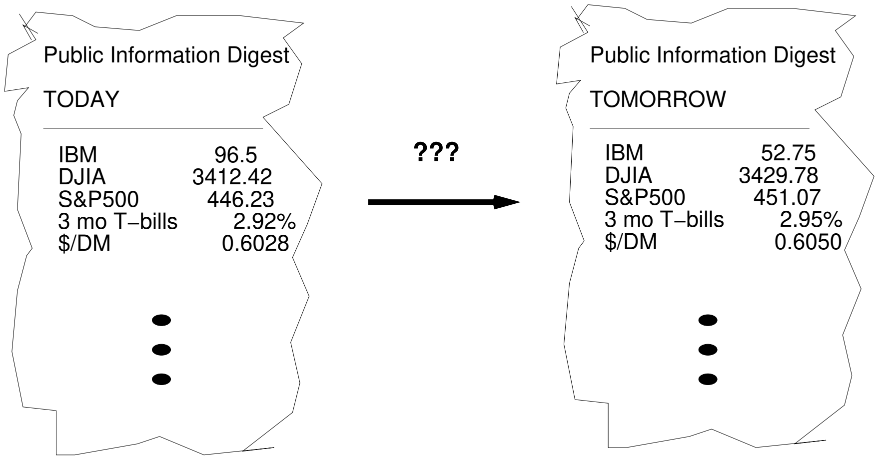


Figure 1-1: Can historical market data be used to predict tomorrow's stock market prices?

This thesis is an attempt to apply a new nonlinear statistical modeling technique, Radial Basis Functions, to the rather bold task of stock market prediction and analysis. The success of our approach will ultimately depend on whether or not there are significant nonlinear relationships in the markets that can be discovered empirically. However, before we even get to that stage, it will be imperative for us to develop new algorithms for selecting, estimating, and diagnosing these models, and for harnessing powerful computers to assist in our search. This necessity is partly due to the complex, noisy nature of stock market data we are modeling, and partly due to our own ignorance about the type of relationships we are looking for. Regardless of our success on the stock market prediction problem, these new algorithms promise to be valuable tools for modeling many real world systems.

Before jumping in and exploring new technology for its own sake, though, lets take a moment to see why our particular choice of technology may be a good one for financial modeling problems in general, and stock market prediction problems specifically.

1.1 Financial Modeling

Finance is the science of the relationship between business and money:

```
fi.nance \f*-'nan(t)s, 'fi--., fi--'\ n [ME, payment, ransom, fr. MF,
  fr. finer to end, pay, fr. fin end m more at FINE pl  1: money or
  other liquid resources esp. of a government or business  2: the
  system that includes the circulation of money, the granting of
  credit, the making of investments, and the provision of banking
  facilities  3: the obtaining of funds or capital : FINANCING
```

Financial modeling, then, is the attempt to capture mechanisms and relationships in finance. What tools do we need to capture and understand the workings of the financial markets, the playing field for this game of money? A distinction frequently made in financial analysis is between *quantitative* and fundamental or *qualitative* methods.

Certainly numbers and quantitative analysis are invaluable for the accounting side of the financial markets, to "keep score" of how participants are doing. But in the last 30 years the financial world has embraced a decidedly quantitative orientation for many parts of the decision making processes as well, widely adopting quantitative theories such as modern portfolio theory, the Capital Asset Pricing Model, and option pricing theory. Why have these techniques been so widely adopted?

Modern Portfolio Theory (MPT) developed from the work of Markowitz (1959) on selecting a good mix of stocks to hold in a portfolio. Markowitz emphasized that investors ought to maximize the expected returns of their investments *for a given level of risk*. He proposed the variance of returns of a portfolio as the measure of its risk, and covariance of a stock's return with respect to the portfolio as a measure of how diversifying that stock would be for the given portfolio. His formulation led to a solution of the portfolio selection problem in terms of a quadratic optimization problem, yielding one of the first systematic approaches to the problem available to investors, and consequently this approach is at the core of a large fraction of the portfolio management systems today.

The Capital Asset Pricing Model (CAPM), due primarily to the work of William Sharpe (1964), John Lintner (1965), and Jan Mossin (1966), is one of a number of models that grew out of Modern Portfolio Theory. It further quantifies the relationship between risk and expected return of an asset by modeling the return of an asset as a linear function of the return of the "market" as a whole. The strength of this linear relationship, beta, is now a standard statistic reported for assets.

Seminal work in option pricing theory by Fischer Black and Myron Scholes (1973) quantified the relationship between a now standard financial instrument called an "option" and the stock or other asset underlying it, based on assumptions about the statistical properties of the underlying asset. The understanding promoted by the strong theoretical derivation and explicit assumptions of the Black-Scholes model and its variants has fueled its wide acceptance and use in this rapidly growing segment of the financial markets, and the theory has also found wide use in other areas of finance.

Although these theories have become widely accepted, it would be a mistake to think that even the investment arena of finance is driven entirely by quantitative analysis. On the contrary, many of investing's most successful players have adhered to a "fundamental" approach involving careful hand-tailored qualitative assessments and decisions (take for instance Peter Lynch (1989)). However, I believe that much more financial analysis and decision making would be done quantitatively if it were possible to accurately and comprehensively quantify the relevant factors and interactions between factors. Ever increasing amounts of raw data about the financial markets and world economies are becoming available to facilitate this. The real limitation here is our ability to create quantitative models that work and that we can believe.

1.2 Statistics / Time Series

Statistics is at the core of much of financial modeling. It is used to circumvent the unfortunate fact that we don't know how to capture the complexity of the markets. Financial markets involve thousands or millions of participants, whose rules and reasons for doing things are hidden from us. Furthermore, the businesses implicit in the securities traded in the markets are themselves another level of complexity we can't hope to fully capture. Thus we must content ourselves with gross characterizations of market forces, and attribute a large part of the events we see to non-deterministic random "noise". In modern portfolio theory, for instance, we abandon hope of knowing the exact future course of security prices, and instead characterize the behavior of their returns as following a normal distribution, where the interaction between securities is summarized by covariance. CAPM simplifies this even further by asserting that the interaction between securities can be captured by their relationship to one variable, the "market". In general, many quantitative financial models involve characterizing market forces as random variables with certain statistical distributions, and many of the interactions between these variables are often assumed to be non-existent or of a certain rigid form so that the analysis is tractable.

Time series are ordered sequences of numbers relating to some observed or measured entity, where the ordering of the sequence is typically done based on the time of the observation or measurement. Typical examples in business and economics are daily closing stock prices, weekly interest rates, and yearly earnings (see Figure 1-2). Contrary to the ease with which we can come up with examples of time series in finance, however, there is a remarkable *lack* of the use of systematic time series *analysis* techniques in finance. For instance, very simple models (e.g. constant or first order linear models) are sometimes used to predict future values of important company attributes such as sales, but little use is made of higher order models. Why is this?

A commonly held belief about financial markets is that they are *efficient*, which is often taken to mean that predictability cannot be profitably exploited in a trading rule on the basis of publicly available information once the proper discounting for risk is done (for instance see Jensen (1978)). A narrower statement of this basic belief is the Random Walk Hypothesis, which proposes that the best prediction for future values of a series is the last observed value. If these hypotheses are even *close* to being true, it is easy to see that statistical modeling will not easily yield useful results.

Following the ideas of many researchers in the area, my hypothesis is that the dynamical systems comprising the financial markets require more complex models than have been tried previously. For instance, virtually all statistical modeling and hypothesis testing in the financial markets has traditionally been done with *linear* models (e.g. CAPM, APT). Partly this has been done for practicality; linear models

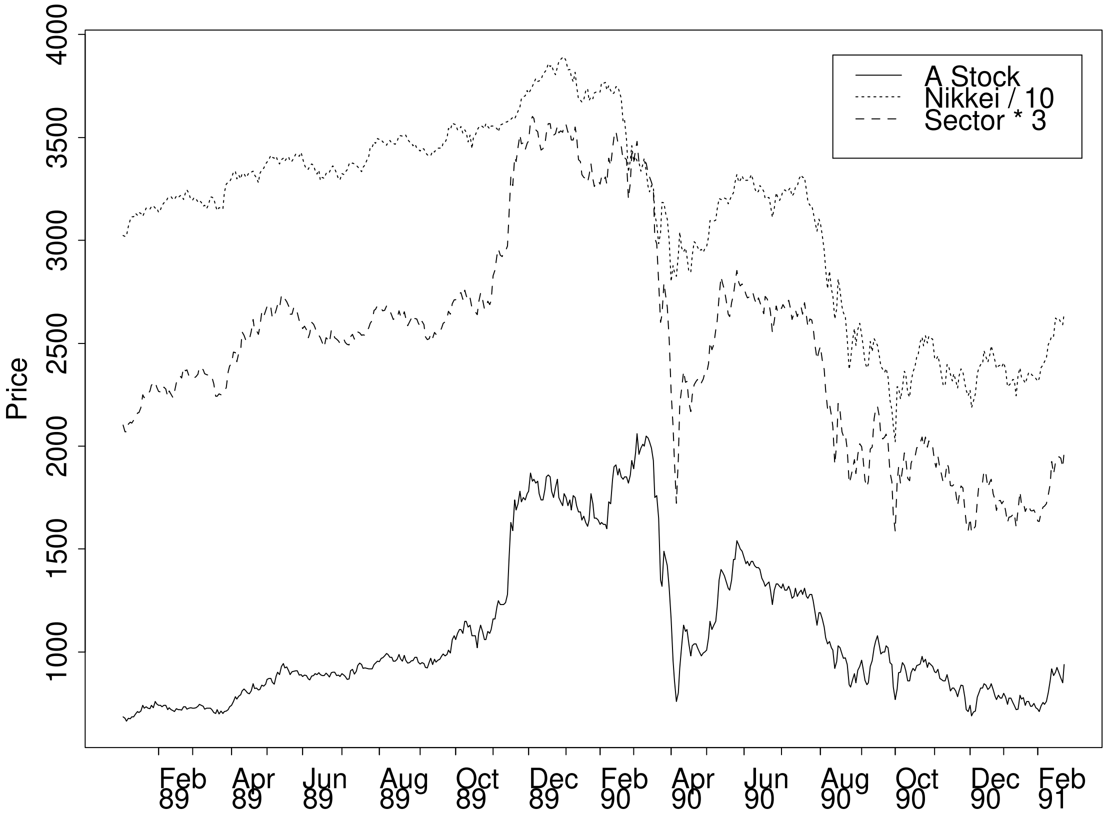


Figure 1-2: Typical data used in this thesis: Japanese daily closing prices for a stock, its industry sector, and a broad market index.

have the best developed and understood techniques for specification, estimation, and testing. However, it is possible (or even likely) that many important relationships in finance are nonlinear, and that no simple transformation can be made to make them linear over a large enough range to be interesting (Tong (1990) gives some interesting arguments in this direction). Indeed, recent results in Tsay (1988) and LeBaron (1990) indicate that simple nonlinear or "regime switching" models can be effective for prediction.

Another possibility for capturing complexity may lie in estimating larger models. In the context of univariate models, this means using more lagged values of a given time series in the model. If the underlying system we are looking for involves data from multiple sources (or if there are multiple state variables for the system), then multivariate models are a natural candidate for capturing the complexity of the system. The problem with larger models, however, is that the potential for capturing extra complexity does not come for free. Larger models mean more parameters, which means either that we need more data to estimate the parameters, or we are less certain in our estimates (and thus in the overall usefulness of the model). Since the amount of data we can (or are willing to) use is often fixed, good statistical modeling methodology often then amounts to a careful consideration of the interplay between model complexity and reliability. These will be recurrent themes throughout this work.

1.3 Learning Networks

Techniques for nonparametric nonlinear statistical modeling have proliferated over the last 15 years. Projection pursuit regression, multilayer perceptrons (sometimes called "backpropagation networks"[1]), and radial basis functions are three popular

[1] More accurately, the term "backpropagation" is now typically used to refer to the particular gradient descent method of estimating parameters, while the term "multilayer perceptron" is used to refer to the specific functional form described below.

examples of these techniques. Although originally developed in different contexts for seemingly different purposes, these techniques can all be viewed as nonparametric approaches to the problem of nonlinear regression. Following Barron and Barron (1988) we call this general class of methods *learning networks*, to emphasize this unifying view and acknowledge their common history. We now review the definitions and relevant properties of these networks.

1.3.1 Standard Formulations

In this section we show the standard formulations for the learning networks used in this thesis. For ease of presentation we will assume the "multiple regression" situation of mapping multiple input variables into a univariate output, although the true multivariate situation is a straightforward extension in all cases. Given the well known relation in statistical estimation theory between the size of models, number of data points, and approximation error, we also focus on the number of parameters implied by each model so that we can later make comparisons between them on a roughly equal footing. Note however that the notion of counting free parameters is a simplistic measure of the complexity of nonlinear models, and it may be possible to be more accurate with other measures (e.g. the nonlinear generalizations of the influence matrix in Wahba (1990)).

Radial Basis Functions

Radial Basis Functions (RBFs) were first used to solve *interpolation* problems - fitting a curve exactly through a set of points (see Powell (1987) for a review). More recently the RBF formulation has been extended by a variety of researchers to perform the more general task of approximation (see Broomhead and Lowe (1988), Moody and Darken (1989) and Poggio and Girosi (1990)). In particular, Poggio and Girosi (1990) show how RBFs can be derived from classical regularization techniques for handling ill-posed problems. A general formulation for Radial Basis Functions can be written as follows:

$$f(\vec{x}) = \sum_{i=1}^{k} c_i * h_i(\|\vec{x} - \vec{z_i}\|) + p(\vec{x}) \tag{1.1}$$

where $\vec{x}$ is a vector of the d inputs x^1 thru x^d, the $\vec{z_i}$'s are d-dimensional vector prototypes or "centers", $\|\cdot\|$ is some vector norm, the c_i's are coefficients, the h_i's are scalar functions, and $p()$ is a polynomial function. Note that this formulation is more general than that used in many studies in that the $\vec{z_i}$'s can vary, the vector norm need not be Euclidean, k is typically less than the number of points in the data set, and the basis functions h_i can vary for each center[2]. In this work we take the vector norm to be a weighted Euclidean norm defined by a d by d matrix W, and the polynomial term will be taken to be just the linear and constant terms, thus resulting in the following formulation:

$$f(\vec{x}) = \sum_{i=1}^{k} c_i * h_i((\vec{x} - \vec{z_i})^T \cdot W^T \cdot W \cdot (\vec{x} - \vec{z_i})) + \sum_{i=1}^{d} c_{i+k} * x^i + c_{k+d+1} \tag{1.2}$$

Intuitively, an RBF network "operates" in the following manner. First, weighted distances are computed between the input $\vec{x}$ and a set of "prototypes" $\vec{z}$. These scalar distances are then transformed thru a set of nonlinear basis functions h, and these outputs are summed up in a linear combination with the original inputs and a constant. Common choices for the basis functions $h(x)$ are gaussians (i.e. e^{-x/σ^2}) and multiquadrics (i.e $\sqrt{x + \sigma^2}$), although Micchelli (1986) showed that a large class of functions are appropriate.

Note that networks of this type can generate any real valued output. In applications were we have a priori knowledge of the range of the desired outputs, it can be advantageous to apply some nonlinear transfer function to the outputs to reflect that knowledge. This will be the case in this paper, for instance, and thus some of

[2]This formulation has been called "hyper basis functions" by Poggio and Girosi (1990). In this thesis we use the term "radial basis functions" to encompass both the interpolation scheme used by Powell and the subsequent generalizations of that basic method.

the RBF networks used here will be augmented with an "output sigmoid". Thus the new network will be of the form $g(f(\vec{x}))$ for $f(\vec{x})$ as above and $g(z) = 1/(1 + e^{-z})$.

Given a particular modeling problem (i.e. a set of inputs and desired outputs), model fitting amounts to estimating the parameters of the RBF approximation: the $d*(d+1)/2$ unique entries of the symmetric weight matrix $W^T W$, the $d*k$ elements of the centers $\vec{z}$, and the $d+k+1$ coefficients c. Thus the total number of parameters that need to be estimated for d-dimensional inputs and k centers is then $d*k+d^2/2+3*d/2+k+1$.

Multilayer Perceptrons

Multilayer perceptrons (MLPs) are arguably the most popular type of "neural network", the general category of methods that derive their original inspiration from simple models of biological nervous systems. They were developed independently by Parker (1985) and Rumelhart et.al. (1986) and popularized by the latter. Following the notation of Section 1.3.1, a general formulation for MLPs with one output can be written as follows:

$$f(\vec{x}) = h\left(\sum_{i=1}^{k} c'_i * h(\sum_{j=1}^{d} c_{i,j} * x_j + c_{i,d+1}) + c'_{k+1}\right) \tag{1.3}$$

where h is typically taken to be a smooth, monotonically increasing function such as the "sigmoid" function $1/(1+e^{-x})$, the c and c''s are coefficients, and k is the number of "hidden units". This is typically referred to as an MLP with "one hidden layer" because the basic "sigmoid of a dot product" equation is nested once, but the nesting be repeated more times. Note that unlike the RBF formulation, the nonlinear function h in the MLP formulation is typically fixed for the entire network. Fitting MLP models given the inputs $\vec{x}$ and desired univariate outputs $f(\vec{x})$ then amounts to solving for $(d+1)*k$ parameters c and $(k+1)$ parameters c', for a total of $(d+2)*k+1$ parameters.

Projection Pursuit Regression

Projection pursuit methods are a class of methods that emerged from the statistics community for analyzing high dimensional data sets by looking at low dimensional projections of it. Friedman and Stuetzle (1991) developed a version particularly for the nonlinear regression problem called projection pursuit regression (PPR). Similar to MLPs, PPR solutions are composed of projections of the data (i.e. dot products of the data with estimated coefficients), but unlike MLPs they also estimate the nonlinear combining functions from the data. The formulation for PPR then can be written exactly as in Equation (1.3) if we note that the inner h are different for each i and are computed from the data (typically with a smoother), and the outer h is taken to be the identity function. In terms of the number of parameters PPR estimates, note that the use of a smoother for estimating the inner h's complicates matters. Our simplistic approach is to count each use of the smoother as estimating one parameter (i.e. the bandwidth of the smoother), although as mentioned in Section 1.3.1 it may be possible to be more exact. Thus the total number of parameters is $d * k$ projection indices, k linear coefficients and k smoothing bandwidths, for a total of $(d+2)*k$ parameters.

1.3.2 Network Properties

Although the various learning network techniques originated from a variety of backgrounds and generally are not completely understood, some common properties are worth noting.

Approximation

All of the above learning networks have been shown to possess some form of a *universal approximation* property. For instance, Huber (1985) and Jones (1987) have shown that with sufficiently many terms, any square integrable function can be approximated arbitrarily well by PPR. Cybenko (1988) and Hornik (1989) have shown that one hidden layer MLPs can represent to arbitrary precision most classes of linear and nonlinear continuous functions with bounded inputs and outputs. Finally, Poggio and Girosi (1990) show that RBFs can approximate arbitrarily well any continuous function on a compact domain. In a related vein, Poggio and Girosi also show that RBFs have the best approximation property - i.e. there is always a choice for the parameters that is better than any other possible choice - a property that is not shared by multilayer perceptrons.

Error Convergence

The universal approximation results, however, say nothing about how easy it is to find those good approximations, or how efficient they are. In particular, does the number of data points we will need to estimate the parameters of a network grow exponentially with its size (the so-called "curse of dimensionality")? Recent results show that this is not necessarily true *if we are willing to restrict the complexity of the function we want to model.* For instance Barron (1991) recently derived bounds on the rate of convergence of the approximation error in MLPs based on the number of examples given assumptions about the smoothness of the function being approximated. Chen (1991) has shown similar results for PPR. Girosi and Anzellotti (1992) derived bounds on convergence in RBFs using somewhat more natural assumptions about the smoothness of the function being approximated. Niyogi and Girosi (1993) subsequently extended this result for the estimation problem and derived a bound on the "generalization error" of RBFs, i.e. the error an RBF network will make on unseen data.

The importance and centrality of generalization error bounds to the process of data driven modeling is worth noting. In particular, these bounds show that for a fixed number of data points, the generalization error that we can expect from a network first decreases as the network complexity (i.e. number of parameters) increases, then after a certain point the error *increases* (see Figure 1-3). For many of the financial modeling problems considered in this thesis the data set size is to some extent fixed, and thus these results indicate that there will be an optimal number of parameters to use for that size of data set.

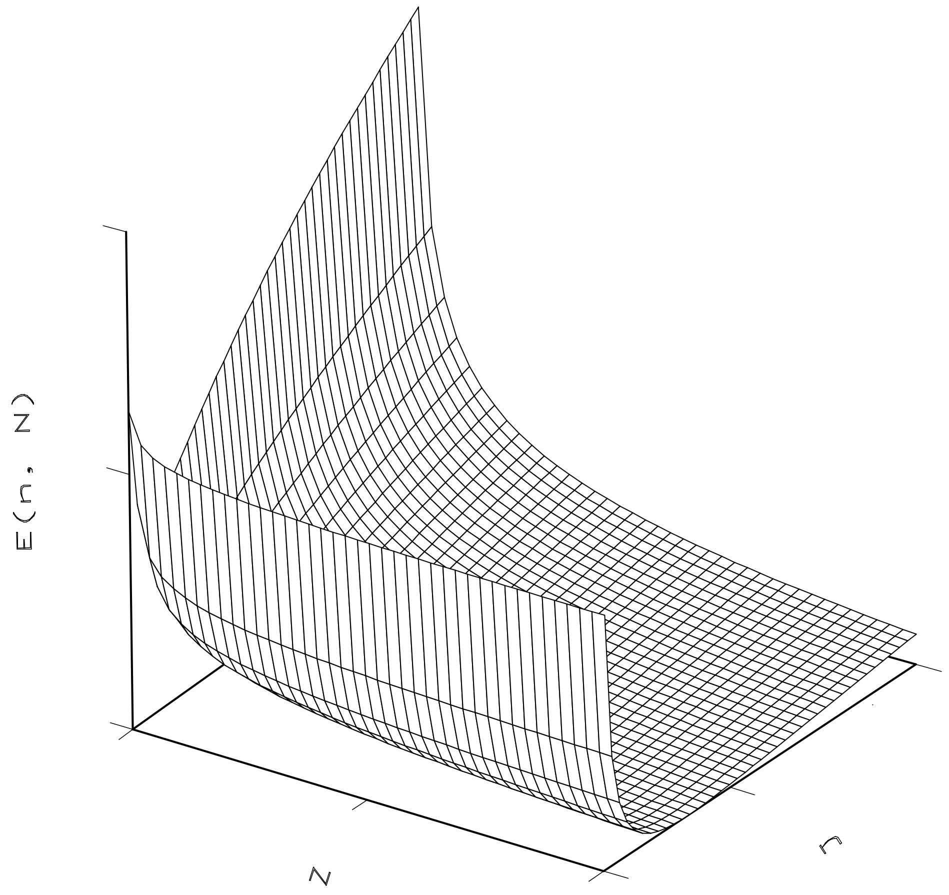


Figure 1-3: Generalization error $E(N, n)$ for a gaussian RBF network as a function of the number of data points N and the number of network parameters n (reprinted with permission from Niyogi and Girosi (1993)).

Other interesting estimation properties have been investigated for PPR in particular. Diaconis and Shahshahani (1984) studied necessary and sufficient conditions for functions to be represented *exactly* using PPR. Donoho and Johnstone (1989) demonstrated the duality between PPR and kernel regression in two dimensions, and showed that PPR is more parsimonious for modeling functions with angular smoothness.

Parameter Estimation Methods

In our treatment above we have focused on the representation used by each method, but of course a critical concern is how to actually estimate the parameters of the models. To some extent these issues can be divorced from one another, and in fact there is a large body of literature concerned with applying various estimation schemes to various networks. Generally this work shows that the speed and accuracy of the estimation process depends on what kind of derivative information is used, whether all parameters are estimated simultaneously or incrementally, and whether all the data is used at once in a "batch" mode or more incrementally in an "online" mode. In Chapter 2 we will more fully explore estimation techniques for RBF networks, the central method in this thesis. However, a rigorous comparison of methods is not the primary goal of this work; rather it is to see if *any* method can yield useful results. As such we have adopted the most common estimation schemes for our use of the other types of learning networks (esp. in Chapter 3). In particular we adopt Levenberg-Marquardt for batch mode estimation of the RBF networks, gradient descent (with momentum) for online mode estimation of the MLP networks, and the Friedman and Stuetzle algorithm for PPR (which uses a Newton method to compute the projection directions and the "supersmoother" for finding the nonlinear functions h).

Equivalence of Different Learning Networks

There is another reason why we choose not to delve too deeply into the merits of particular learning networks over others; recent theoretical developments suggest that there are significant connections between many of these networks. Maruyama, Girosi and Poggio (1991), for instance, showed an equivalence between MLP networks with normalized inputs and RBF networks. Subsequently, Girosi, Jones and Poggio (1993) showed that a wide class of approximation schemes could be derived from regularization theory, including RBF networks and some forms of PPR and MLP networks. Nonetheless we expect each formulation to be more efficient at approximating some functions than others, and as argued by Ng and Lippman (1991), we should be mindful that the practical differences in using each method (e.g. in running time or memory used) may be a more differentiating factor than model accuracy.

1.4 A State Space / RBF Approach

In this section we outline our strategy for mapping time series prediction problems investigated in this thesis onto the learning networks introduced in the previous section. We can classify the general techniques for time series analysis and prediction into two categories. (1) In the case that a lot of information about the underlying model of a time series is known (such as whether it is linear, quadratic, periodic, etc.), the main task left is then to estimate a few parameters of the model to fit the observation data. Sufficient observations can make this kind of model quite accurate and powerful. Unfortunately, for many problems in finance and economics the underlying models are often unknown or ill-specified. (2) At the other extreme, the only thing available is a set of observations. For such problems, people often assume that the underlying model has some "state variables," which determine what the values of the time series should be.

Formally, we define a general state space model as follows:

Let $......x_{t-j}, ..., x_{t-1}, x_t, x_{t+1}, ...$ be a time series, we assume:

$$x_{t+1} = f(y^1_{t+1}, ..., y^i_{t+1}, ..., y^d_{t+1}) + N_{t+1}$$

where N_{t+1} represents random noise at time $t+1$ and $y^1_{t+1}, ..., y^d_{t+1}$ are *state variables*, and

$$y^i_{t+1} = g_i(y^1_t, ..., y^k_t, ..., y^d_t, x_t, x_{t-1}...), \qquad i = 1, 2, ..., d$$

f and g_i's are some functions. Note that x and y can be scalar quantities for univariate models, but they could also be vector valued for the more general multivariate setting. This formulation is a generalization of that given in Chatfield (1989) to approximate nonlinear models. The motivation of using state variables is that they often correspond to certain features or properties of the time series and can help us understand and characterize the series. They can also help to simplify the computations for analysis and prediction (see Figure 1-4). In our work, f is some general tool for function approximation (*model fitting*), and g_i's transform the original "raw data" to a new representation (*model building*).

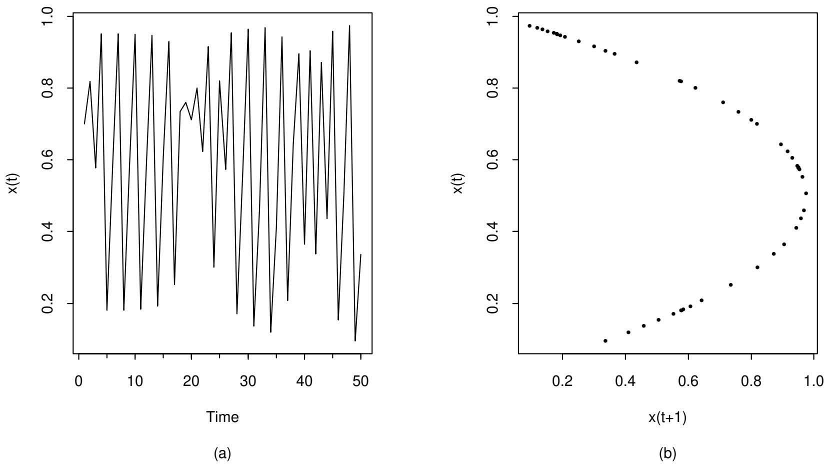


Figure 1-4: A time plot (a) of the logistic function $x_{t+1} = 3.9 * x_t * (1 - x_t)$ is difficult to decipher, but a state space plot (b) clearly shows the underlying model. Following the notation of the text, this simple example uses one state variable $y^1_{t+1} = x_t$ and can then be written as $x_t = 3.9 * y^1_t * (1 - y^1_t)$.

1.4.1 Model Fitting

Given this formulation for our models, it is now up to us to specify algorithms for how to choose the functions f and the g_i's given a particular data set of observations. The purpose of our formulation was to pick a *nonlinear* representation at least for f, since otherwise the formulation would degenerate into classical linear ARMA time series models. In this thesis we will primarily choose RBF networks for model fitting because of our experience with these networks, although all of the learning networks from Section 1.3 are reasonable choices.

A central problem that arises from the large size of the RBF networks we propose to use is that of *overfitting* the data, that is, having so many degrees of freedom in the model that some of them capture meaningless features of the noise in the data. One possible way of reducing the number of parameters is to only use a diagonal matrix for W, which could suffice if the different inputs are in roughly commensurate units. Another possibility is to keep the centers $\vec{z}$ fixed at a subset of the inputs, which can work nicely when the problem is regular or there are many examples. Nonetheless, as in all statistical modeling, the chances of our model fitting *unseen* data nicely depends critically on maintaining a balance between the number of data points used for estimation and the number of parameters estimated. Thus two major questions addressed in Chapter 2 of this thesis are how to estimate these parameters efficiently and how to avoid unnecessary parameters.

Fast supercomputer implementations of these techniques will also prove useful in this thesis. Partly this is because of our tendency to want to use as much data as possible in fitting these relatively complex models, but it also will facilitate fitting the multitude of models we will evaluate in both searching for the appropriate function form for the models, and in testing their robustness via sample reuse strategies. For these reasons, we have developed a general purpose RBF time series modeling program on the Connection Machine supercomputer, which will be described in Chapter 6.

1.4.2 Model Building

The use of "black box" techniques to discover structure in data sets is compelling, since they relieve us of the need to hypothesize the exact functional form of the structure. Indeed, it would be nice if we could simply give the computer our data set and have it report to us the "best" formula for describing the data. However, this is naive for at least three reasons.

First, with a data set of any appreciable size (esp. with many different variables) the combinatorial number of models to consider would preclude any sort of exhaustive search, even on today's fastest supercomputers. Unfortunately, in the general case exhaustive search is required, since we cannot infer anything about the performance of an untried model from the performance of related models without knowing the relationship between the variables. For example, "all subsets" methods are often advocated for modeling linear systems since incremental or "stepwise" methods are not guaranteed to find the best model (see Cryer and Miller (1991)).

Second, finding good *representations* for the data is a crucial and labor intensive ingredient in finding good fitting models. This corresponds to choosing the g_i functions in our state space formulation above. By representation we refer to all of the possible things we might do to massage the raw data into the form that is ultimately used in model fitting, and includes issues such as which variables to use, how to sample the data, what (if any) simple transformations to perform on the data values, and how to encode the resulting values. Inattention to these issues can deteriorate our ability to find good models just as easily as a poor choice of model fitting technique. For instance, scalar transformations of the input data are often necessary to express inputs in commensurate units, satisfy distributional assumptions (e.g. normality), achieve independence (e.g. first differencing serially dependent time series), or limit the influence of "outliers". In general, a major lesson from Artificial Intelligence research is that *representation* plays an important role in problem solving (for instance see Winston (1984) or Brachman and Levesque (1985)). A good representation embodies our prior information about how to solve problems in a domain by making useful information explicit and by stripping away obscuring clutter. Two different representations can be equivalent in terms of expressive power (e.g. the class of functions expressible in both representations is the same), but differ dramatically in the efficiency or ease with which to solve problems.

Finally, the tools for creating nonlinear statistical models are much too immature to consider completely automating them. Identification and diagnostic tools are too piecemeal to believe that they would catch all conceivable situations. Most fitting methods are too sensitive to permit naive use. The often heard advice of statisticians is relevant here: look at your data, check your assumptions, and know your methods.

So what tools do we have available for determining the representation we should use? Certainly we should take advantage of the corresponding tools for linear modeling, since our nonlinear models may in fact be no more than linear, or contain a significant linear component. Thus for instance we can use linear correlation measures for identifying related variables. Plotting data will also be a significant tool - the power of the human eye to detect subtle regularities should not be underestimated. However, our main tool in finding good representations will be the nonlinear fitting method itself. The idea here is to compare two representations by trying the fitting method on each, and making an "apples to apples" comparison of the resulting performance.

1.4.3 Model Reliability

Our division of the overall modeling problem into model "building" and "fitting" is admittedly somewhat arbitrary, and there are certainly dependencies between the two activities. Ultimately the important thing is not exactly how we label the pieces of the final result (nor even how we found them!), but rather *how accurate the complete model is at describing unseen data.*

In this regard we will make heavy use of sample re-use techniques from statistics such as cross-validation (see Efron and Gong (1983)), which give better estimates of the error we are likely to see on unseen data. But there is a problem with the way we intend to use these techniques - they only provide unbiased error estimates on the *first attempt* at modeling the data. For a variety of reasons we are *repeatedly* attempting to fit the same set of data, and thus we run the risk of "data mining"; that is, finding a spurious relationship in the data simply from looking too hard. Because of the general importance of this concern for anyone adopting an data driven approach to modeling, we will spend some time reviewing it and offer some suggestions for minimizing its impact.

In addition to the question of our overall confidence in a model, we will also address the question of *where* our models are accurate. We do this by deriving a pointwise variance estimate for RBF networks, that is, an expression for what the variance is of each output of the model. In predictive systems this will allow us to quote a range for the outputs, or potentially say "I don't know" because of the high uncertainty of the model at that point. This could lead to higher fidelity models in regions of high data density by dropping constraints of obtaining a spatially uniform fit. Finally, it may be a useful notion for systems that have time varying statistics by providing an operational definition of a "dead model" - one that no longer makes sufficiently focused predictions.

1.4.4 Tasks Besides Prediction

As hinted at above, this approach to financial modeling is broader than simply trying to predict tomorrow's stock market average. We would like estimates of what modeling error we are likely to see on future data, how certain we are of predictions, and when are our models outliving their usefulness. In this spirit, what are the other tasks from the finance perspective that we might profitably explore with this approach?

First, given a system that predicts the price of some security, there are numerous ways that those predictions can be used to implement a trading system. From a finance perspective we don't care about the R^2 or RMS error or some other statistical modeling measure of a model's performance: we care about how much money the system can make, or how reliably it can make money, thus we should measure our success in those terms. Furthermore, optimizing some performance measure of the trading system may imply the appropriateness of a different error measure for the modeling process (e.g. least absolute value instead of least squares).

Another issue we would like to explore concerns the non-stationary, time varying nature of financial data. In a heuristic attempt to limit the effects of non-stationarity we will be tempted to avoid use of data that is too old. To get enough data for estimation, then, we will naturally try to exploit another dimension, such as looking at cross-sectional models (i.e. models that relate different securities to each other, rather than just to themselves).

Finally, we propose the usefulness of applying this modeling technology to other areas besides prediction. One example is deriving a monetary value of a source of data based on the best available trading system which uses it. Another example is finding persistent discrepancies between market prices and accepted pricing theories (e.g. for options).

1.5 Other Approaches

Nonlinear time series analysis is a relatively new area of investigation, and it has been approached from a variety of backgrounds. The statistics community pioneered it in the 1980's by proposing extensions to existing linear models (esp. the ARIMA models of Box & Jenkins (1976)), for instance combining two or more linear models in a simple nonlinear way (e.g. threshold autoregressive or TAR models of Tong and Lim (1980)). For reviews of the numerous possibilities here see Priestley (1988), Tong (1990), or Granger (1991). These approaches are pleasing because of the scrutiny given in their development for the standard statistical considerations of model specification, estimation, and diagnosis, but their generally parametric nature tends to require significant a priori knowledge of the form of relationship being modeled.

Independently in the late 1980's, the physics and dynamical systems community has constructed nonlinear state space models, motivated by the phenomena of chaos. Crutchfield and MacNamara (1987) introduced a general method for estimating the "equations of motion" (i.e. model of the time behavior) of a data set cast into the state space formulation, which included a novel measure of the usefulness of the model based on entropy. Farmer and Sidorowich (1989) make a case for breaking up the input domain into neighborhoods and approximating the function locally using simple techniques (e.g. linear or quadratic fitting). Note that the work by this community addresses many interesting problems besides prediction, such as optimal sampling strategies, identifying the dimensionality of the system, identifying characteristics of the system (i.e. Lyapunov exponents) that determine how feasible it is to do long term prediction, and testing if a data set is nonlinear.

Many of the attempts from the dynamical systems area used RBFs as the function approximation method, although we note that these previous approaches restricted the RBF formulation given here in some way. Broomhead and Lowe (1988) applied RBFs to predicting the logistic map, and showed the usefulness of using fewer centers than data points. Casdagli (1989) investigated how error scaled with number of examples for strict interpolation RBFs (i.e. data used as fixed centers). Jones et.al. (1990) show how normalizing the basis functions and adapting the gradient uses the data more efficiently for predicting the logistic map and Mackey-Glass equation. Kadirkamanathan et.al. (1991) give a method similar to RBFs where they incrementally add basis functions to the approximation as dictated by the distribution of error.

Simultaneously related attempts were being made in the "neural network" community in the late 1980's, focusing more on practical applications and on the issue of prediction accuracy relative to other methods. Lapedes and Farber (1987) applied multilayer perceptrons (MLP) to some of the same prediction problems popular in the chaos community, the Logistic map and the Mackey-Glass equation, and found them to be superior to the Linear Predictive method and the Gabor, Wiener & Volterra polynomial method, and comparable to the local linear maps of Farmer and Sidorowich. White (1988) used MLP to model univariate IBM stock returns, but found no significant out of sample performance. The problem of overfitting on the training data was noted by White and many other authors, and this inspired the wider use of sample reuse techniques from statistics, such as cross validation methods. Utans and Moody (1991) clearly state the advantages of doing so, and also develop a new estimator of out of sample prediction error which penalizes for "effective" number of parameters of general nonlinear models. They also apply this measure using MLP networks to the problem of predicting corporate bond ratings and show superior performance over linear regression. Weigend (1991) developed a technique for penalizing extra parameters in an MLP network and show how the resulting parsimonious networks outperform the corresponding TAR model. De Groot and Würtz (1991) also find evidence for the usefulness of MLP networks by comparing them with traditional statistical models such as linear, TAR, and bilinear models on univariate prediction problems. They also note the superiority of smarter parameter optimization methods than gradient descent, and note that the Levenberg-Marquardt method worked best for their problems. Refenes (1992) proposed a method for incrementally adding units to the MLP paradigm and showed how his method outperformed linear ARMA models on predicting foreign exchange rates. Finally, a number of researchers have tried other more or less vanilla applications of MLP networks to financial market prediction problems, but often the writeups of this work are plagued by insufficient detail concerning critical aspects of their models (e.g. variable selection and preprocessing) and/or the performance measures quoted are not sufficiently explained or benchmarked (for example Kimoto et.al. (1990) or Wong and Tan (1992)).

1.6 Major Contributions

This section briefly summarizes the unique aspects and important contributions in this thesis, which fall in three main areas. First, I have developed specific algorithms and methodologies for the efficient use of RBF networks for modeling noisy and high dimensional systems. Novel aspects of this work include:

- Insights into how RBF's operate most efficiently drawn from analogies to related systems.
- Integration of state-of-the-art nonlinear estimation procedures and a general parameter pruning method not widely known in the machine learning community.
- An elliptical clustering heuristic for setting initial RBF parameter values.
- Derivation of a pointwise variance estimate for RBF predictions.
- Suggestions for managing the "data mining" problem, and recognition that it is not completely solved by sample reuse techniques.

Second, I have demonstrated that the data driven, learning network approach is useful for financial modeling. I have:

- Demonstrated superior performance of multivariate, nonlinear models of the Japanese stock market.
- Developed customized, adaptive option pricing formulas that may be superior to theoretically derived textbook formula.
- Showed novel applications of modeling technology besides prediction, by computing a monetary value for market data based on its arrival time.

Finally, I have offered some reasons for the usefulness of a fast computer implementation of these techniques to facilitate the use of large, real world data sets and repeated modeling attempts, and I have implemented a general purpose RBF code on the Connection Machine parallel computer.

1.7 Outline

The rest of this thesis is organized as follows. Chapter 2 discusses how to efficiently estimate RBF networks and raises some parallels between RBFs and other systems. These techniques and heuristics are put to use in Chapter 3, where we develop a non-parametric method for estimating the pricing formula of derivative financial assets and demonstrate its usefulness on synthetic and real examples. Chapter 4 addresses the critical question for data driven statistical modeling; how confident can we be that the individual estimates and the overall model are valuable for unseen data? Chapter 5 provides more examples of applying this style of modeling to financial time series prediction, and finds that although interesting models can be found, their economic relevance is questionable. Chapter 6 presents some of the implementation issues encountered in this work, and outlines a parallel implementation on the Connection Machine system. Finally, Chapter 7 briefly discusses the results of this thesis and lists some ideas for future work in this area.

Chapter 2

Radial Basis Function Parameter Estimation

In Chapter 1 we introduced the equations defining Radial Basis Functions, but said little about how to find good values for the unknown parameters in those equations. Results about universal approximation and error convergence tell us about the existence and efficiency of our particular representation, but they tell us nothing about *how* to set the coefficients c, centers $\vec{z}$, and norm weights W from Equation 1.2. Before we can successfully apply these networks to nontrivial problems, therefore, we must find efficient ways of estimating these parameters, and that will be the subject of this chapter.

General methods for estimating nonlinear parameters have been around for many years - the relatively sophisticated Levenberg-Marquardt method outlined in Section 2.1, for instance, dates from 1963. We begin this chapter with a brief look at applying these methods to estimating Radial Basis Function parameters. But in our quest to push the limits of these systems to complex, high dimensional problems, we will find that used naively, general methods quickly fall prey to the ubiquitous problem in non-convex optimization problems - local minima (see Figure 2-1). Stochastic methods can help in terms of offering the system a chance of getting out of local minima, but they are often too slow to be palatable.

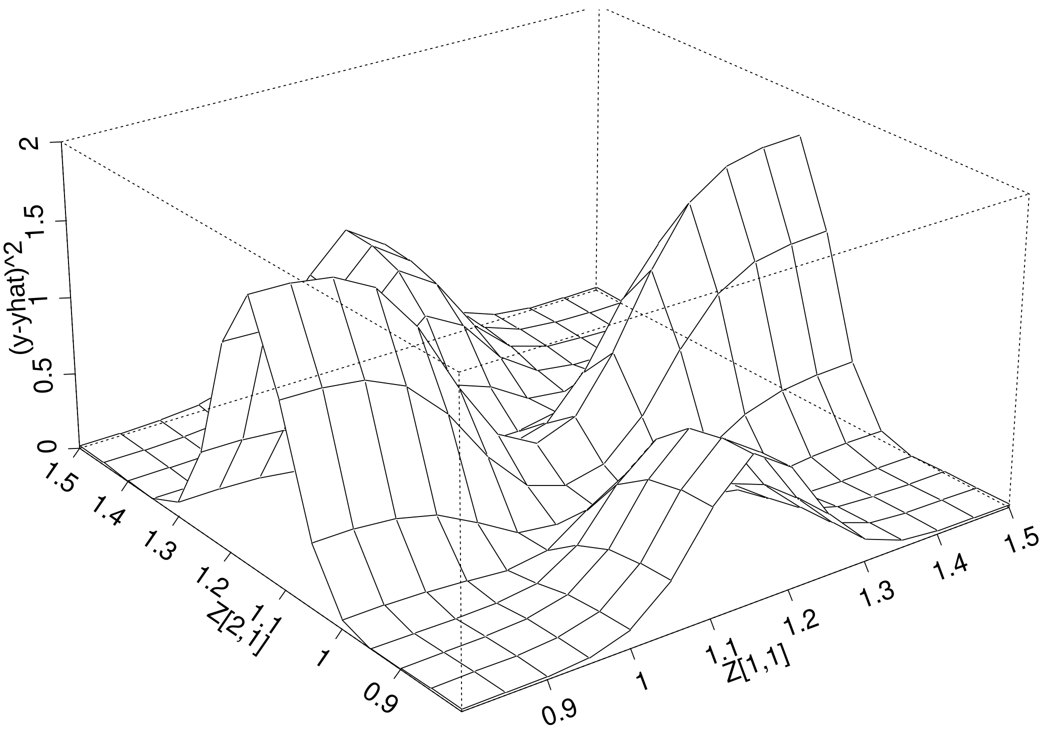


Figure 2-1: Local minima in radial basis function parameter estimation. This plot shows the sum squared error of an option pricing model from Chapter 3 as two center parameters are varied.

Two approaches to solving this problem are investigated in this chapter. The first approach is to take advantage of the specific form of RBF approximations to either help constrain the estimation process, or to provide good initial values for the parameters, so that subsequent use of a general estimation method is more likely to produce acceptable results. The second approach is to first estimate a simple model, and then use those estimates as initial values for successively more complex models. Together these approaches will greatly increase the speed and quality of our overall estimation process.

2.1 General Methods

We begin by applying some standard methods to our problem. Given a function $f(x, \mathbf{a})$ which depends nonlinearly on an p vector of parameters $\mathbf{a}$ as well as the inputs x, how can we solve for $\mathbf{a}$? The standard approach is to define a "cost" or "merit" function χ^2 which when evaluated over the available data takes on small values for good choices of the parameters $\mathbf{a}$. A pervasive choice for χ^2 is

$$\chi^2(\mathbf{a}) = \sum_{i=1}^{n}(y_i - f(x_i, \mathbf{a}))^2 \tag{2.1}$$

where $\{(x_i, y_i) : i = 1, n\}$ are the example input/output pairs[1]. Since we are assuming that f depends nonlinearly on $\mathbf{a}$, minimizing χ^2 in general cannot be done directly, and iterative methods must be used. In this thesis we discuss two such methods: a second order method, commonly referred to as the Levenberg-Marquardt method, which attempts to follow the contours of the χ^2 surface in parameter space; and a simple stochastic method which offers the prospect of jumping out of (or through!) local minima by taking random steps in parameter space.

2.1.1 Levenberg-Marquardt

A simple approach to minimizing the merit function χ^2 is to use information about the gradient of the function to step along the surface of the function "downhill" towards a minimum, i.e. update the parameters $\mathbf{a}$ at each iteration using the rule

$$\delta\mathbf{a} = -\epsilon\nabla\chi^2(\mathbf{a}) \tag{2.2}$$

[1]This choice of χ^2 is often motivated by assuming independent and normally distributed measurement errors with constant variance, in which case minimizing χ^2 is a maximum likelihood estimator, but in fact it is a reasonable choice even if those assumptions are not true.

where the gradient $\nabla\chi^2(\mathbf{a})$ is composed of the p first derivatives $\delta\chi^2/\delta\mathbf{a}_k$ of χ^2 and ϵ is a small positive constant. This for instance is one of the estimation strategies proposed in Poggio and Girosi (1990) for radial basis functions.

The problem with the gradient descent approach is in choosing ϵ: we'd like it to be small, so that we stay on the χ^2 surface and thus ensure we make progress moving downhill, but we'd also like it to be big so that we converge to the solution quickly. Solutions to this dilemma include varying ϵ in response to how well previous steps worked, or iteratively finding the minimum in the direction of the gradient (i.e. "line minimization").

The Levenberg-Marquardt method (see Marquardt (1963)) takes a different approach, by recognizing that the *curvature* of the function gives us some information about how far to move along the *slope* of the function. It approximates the χ^2 function with a second order Taylor series expansion around the current point $\mathbf{a}_0$:

$$\chi^2(\mathbf{a}) \approx \chi^2(\mathbf{a}_0) + \nabla\chi^2(\mathbf{a}_0)^T \cdot \mathbf{a} + \frac{1}{2}\mathbf{a}^T \cdot \mathbf{H} \cdot \mathbf{a} \tag{2.3}$$

where $\mathbf{H}$ is the Hessian matrix evaluated at $\mathbf{a}_0$, i.e.

$$[\mathbf{H}]_{kl} \equiv \left.\frac{\delta^2\chi^2}{\delta\mathbf{a}_k\delta\mathbf{a}_l}\right|_{\mathbf{a}_0} \tag{2.4}$$

Since the approximating function is quadratic its minimum can easily be moved to using step size

$$\delta\mathbf{a} = -H^{-1} \cdot \nabla\chi^2(\mathbf{a}_0) \tag{2.5}$$

However, this approximation will not always be a good one (especially early in the estimation process), and thus the Levenberg-Marquardt method allows the user to adopt any combination of the simple gradient descent rule and the inverse Hessian rule by multiplying the diagonal of $\mathbf{H}$ with a constant λ (i.e. $\mathbf{H}'_{kk} \equiv \mathbf{H}_{kk}(1+\lambda)$). Thus a typical iterative strategy is to use more of a gradient descent step by increasing λ

when the previous $\delta\mathbf{a}$ doesn't work (i.e. increases χ^2), and use more of an inverse Hessian step by decreasing λ when the previous $\delta\mathbf{a}$ does work (i.e. decreases χ^2).

Some comments about using the Hessian for RBF estimation are in order. First, note that since the function f inside of χ^2 is an RBF model calculating the Hessian can be done analytically, although following Press et.al. (1988) we drop the second derivatives of f in evaluating $\mathbf{H}$ to minimize the effects of outliers on the quadratic approximation. Second, the large number of parameters in RBF models is problematic; the added cost of assembling and inverting the Hessian may not be worth the speedup in convergence gained for large models. In fact even gradient descent may be unsatisfactory for very non-smooth cost functions; we will pursue this thought in the next section.

Perhaps a more immediate problem is that the Hessian matrix is likely to be ill-conditioned for large models, causing numerical problems for the inversion procedure despite the tendency of the Levenberg-Marquardt method to avoid these problems by increasing λ and making the Hessian diagonally dominant. This problem is especially severe early in the estimation process when we are far from a minimum and there are many conflicting ways the parameters could be improved. We avoid this problem by using a Singular Value Decomposition for inverting $\mathbf{H}'$, and zeroing singular values less than 10^{-6} times the largest value.

On the other hand, the inverse Hessian provides valuable information about the accuracy of parameter estimates, and can be useful for pruning unnecessary parameters from our model. If the quadratic approximation made above is reasonably accurate and the residuals from our model are normally distributed, $2\mathbf{H}^{-1}$ is an estimate of the covariance matrix of the standard errors in the fitted parameters $\mathbf{a}$ (see Press et.al. (1988)), thus a t-test on $\mathbf{a}_k/(2\sqrt{[\mathbf{H}]_{kk}})$ can be used to determine if the k-th parameter should be removed from the model. Even if the residuals are not normally distributed, the above test may be a reasonable way to identify unnecessary parameters. Hassibi and Stork (1992) use this approach for multilayer perceptron networks, and give some evidence that it is superior to schemes that assume diagonal dominance of the Hessian.

However, we would like to point out the advantage of looking at eliminating multiple parameters simultaneously. In RBF networks, for instance, it makes little sense to talk about eliminating just the scale parameter σ from one basis function. Similarly, why should we eliminate the coefficient c connecting a basis function to the output without eliminating all of the parameters of that basis function? Instead it makes sense to consider the change in our cost function that would occur if we eliminated an entire basis function at once. If the above quadratic approximation holds and we are at the minimum, the increase in χ^2 for setting q parameters to zero is

$$\Delta\chi^2 = \frac{1}{2}\mathbf{a}^T \cdot \mathbf{P}^T \cdot \left[\mathbf{P} \cdot \mathbf{H}^{-1} \cdot \mathbf{P}^T\right]^{-1} \cdot \mathbf{P} \cdot \mathbf{a} \tag{2.6}$$

where $\mathbf{P}$ is a $q \times p$ projection matrix which selects the q dimensions of interest from $\delta\mathbf{a}$ and $\mathbf{H}^{-1}$. If our model residuals are normally distributed, this quantity follows a χ^2 distribution with q degrees of freedom and we have an exact confidence test. Note that commonly we will be uncertain both about the normality assumption and the accuracy of the quadratic approximation, and thus we typically must take this statistic with a grain of salt, especially if we attempt to test a confidence region that is large with respect to the nonlinear structure in the χ^2 surface.

As a simple example of the above pruning methods, we can model a linear equation with a RBF network and see if the above diagnostics tell us to drop the nonlinear unit. To check this we fit the equation $y = 2x_1 - 3x_2 + 5 + \epsilon$, where x_1 and x_2 are evenly spaced along [-50,50] and $\epsilon \sim N(0, 0.1)$ is independent gaussian noise, and we used an RBF model with one gaussian basis function[2]. The results in Table 2.1 show that the above tests correctly reject the single nonlinear basis function parameters,

[2] Note that because of the degeneracy of this simple example, the input weights W and the center $\vec{z}$ were held fixed to prevent complete dependence between the parameters.

Parameter	Value	χ^2_1	p-value
c_1	0.0004276	-1.836e-07	0.9997
c_2	2	-1.298e+04	0.0000
c_3	-3	-12.5	0.0004
c_4	5	6.467e+08	0.0000
σ	0.6408	-0.4107	0.5216

Table 2.1: Example of pruning RBF parameters using the inverse Hessian (see text for details). P-value given is probability that the confidence region includes zero, which is significant for the scale parameter σ and the linear coefficient c_1 of the gaussian unit. The joint test for these parameters yields a χ^2_2 statistic of 0.4106 with a p-value of 0.8144. Thus both tests correctly indicate that we should drop the nonlinear unit.

both individually and jointly.

2.1.2 Random Step

Unfortunately if we are modeling complex nonlinear functions, our cost function χ^2 tends to be quite rough and plagued by local minima, and following the gradient will not necessarily lead to good solutions. Stochastic optimization methods attempt to get around this problem by using randomness to allow the evolving parameter estimates to move out of (or through!) local minima. Caprile and Girosi (1990) propose a very simple stochastic method that they found useful for estimating RBF parameters. The basic loop of their algorithm looks like this:

Randomly draw $\delta\mathbf{a}$ uniformly from $-\omega$ to ω

if $(\chi^2(\mathbf{a}+\delta\mathbf{a}) < \chi^2(\mathbf{a}))$ then

$\mathbf{a} \leftarrow \mathbf{a}+\delta\mathbf{a}$

$\omega \leftarrow \alpha \cdot \omega$ (for constant α, $\alpha > 1$)

else

$\omega \leftarrow \beta \cdot \omega$ (for constant β, $0 < \beta < 1$)

if $(\omega < \omega_{min})$ then $\omega \leftarrow \omega_0$

The basic idea is that random changes to a subset of the parameters in $\mathbf{a}$ are tried out and accepted if they lower the value of the cost function χ^2. This means that no attempt is made to stay on the surface of the χ^2 function, and in fact even from a local minima a lucky choice of $\delta\mathbf{a}$ could send the estimation process "through the mountains" into a lower cost "valley". The idea behind this adaptive step size ω is presumably to accelerate the estimation process in regions where good steps are abundant. Because a sequence of unlucky changes can make the range ω arbitrarily small, a threshold ω_{min} is typically used to reset the range to its starting value ω_0.

In using this "random step" algorithm for estimating real problems, however, we find that the probability of accepting $\delta\mathbf{a}$ drops off quickly as the estimation process proceeds. Given that the primary attraction of the random step method is its simplicity, we offer a further simplification. If the probability of accepting $\delta\mathbf{a}$ is low, we are effectively drawing a random number from a sum of uniform distributions. Since parameters are typically chosen so that the maximum number of allowed consecutive failures (i.e. $\log_\beta(\omega_0/\omega_{min})$) is large, this sum approximates a normal distribution. Since the variance of each uniform distribution is $\omega^2/3$ and they are all independent, it is easy to prove that the variance of the normal distribution approaches $\omega_0^2/(3(1-\beta^2))$. Thus a reasonable modification to the random step algorithm is simply draw from a fixed normal distribution, requiring the choice of only one scale parameter for the noise.

As a simple test of the effectiveness of drawing from a normal distribution, we minimized the function

$$f(x) = \begin{cases} sin(x) - 0.10101x & \text{if } x < 99 \\ 0 & \text{if } x >= 99 \end{cases} \tag{2.7}$$

using both the variable uniform distribution and the fixed normal distribution. This function has a global minimum of $f(99) \approx -11$, although a series of local minima separate it from our starting condition of $x_0 = 0$. For this test we somewhat arbitrarily chose $\alpha = 1.5, \beta = 0.75, \omega_0 = 4$, and $\omega_{min} = 0.0001$, and ran each trial for 1000 iterations. In 8192 independent trials of each method, both found the minimum of -11 on at least one trial, but interestingly the average minimum across all trials was considerably better when using the fixed normal distribution (-10.9 vs -1.4).

2.2 RBF Specific Methods

In Section 2.1 we investigated general methods for estimating nonlinear parameters, without attempting to take advantage of the particular structure of RBF models. In this section we would like to see to what extent we can use the specific structure of RBF models to help ease the complexity of the general problem. We have found useful ideas in this vein from two different sources of inspiration: first, by making analogies between RBFs and other traditional statistical systems; and second, by adopting approaches that avoid numerical instabilities. The common goal of these ideas is to come up with extra constraints or better initial parameter values so that the methods from Section 2.1 converge quickly to good minima.

2.2.1 Relation to General Linear Least Squares

If considered in isolation, estimation of the coefficients c_i from Equation (1.1) is exactly analogous to the statistical method commonly known as general linear least squares, and the usual least squares estimator for the $(k+d+1)$ vector of coefficients

$$\hat{c} = (X^T X)^{-1} X^T y \tag{2.8}$$

can be used on the $[n \times (k+d+1)]$ data matrix X of transformed inputs where

$$X = \begin{bmatrix} h_1(\|x_1 - z_1\|) & \ldots & h_k(\|x_1 - z_k\|) & x_1^1 & \ldots & x_1^d & 1 \\ h_1(\|x_2 - z_1\|) & \ldots & h_k(\|x_2 - z_k\|) & x_2^1 & \ldots & x_2^d & 1 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots \\ h_1(\|x_n - z_1\|) & \ldots & h_k(\|x_n - z_k\|) & x_n^1 & \ldots & x_n^d & 1 \end{bmatrix}$$

and x_i^j denotes the j-th component of observation i. Inverting $X^T X$, however, is unusually subject to numerical problems because of the likelihood in our high dimensional nonlinear setting of the matrix being ill-conditioned. Dyn and Levin (1983) proposed a solution to this problem in the context of vanilla interpolation RBFs that preconditioned the X matrix thru the use of an iterated Laplacian operator, which tends to make the matrix diagonally dominant and thus more easily inverted. We instead adopt the solution of using a robust inversion routine; in particular, following Press et.al. (1988) we use the Singular Value Decomposition method for solving (2.8), zeroing singular values that are six orders of magnitude smaller than the largest singular value. As noted in further discussion of this issue in Chapter 6, however, numerical stability alone does not guarantee good, meaningful solutions to the overall problem. We also note that depending on the cost function χ^2 of interest, other linear solvers (e.g. least median or least L1 regression) may be more appropriate than least squares.

2.2.2 Relation to Kernel Regression

There is an obvious similarity between RBF models and the class of models derived from a standard statistical method known as *kernel regression*. The goal in kernel regression is to come up with nonparametric models of μ in the system

$$y_j = \mu(x_j) + \epsilon_j, \qquad j = 1, ..., n \tag{2.9}$$

where we are given n observations $\{(x_j, y_j) : j = 1, ..., n\}$, and the ϵ_j are zero mean, uncorrelated random variables. For ease of exposition we will restrict ourselves to scalar valued x_j here, although all of the results discussed have straightforward extensions to the vector case. The general form of a kernel estimator is a weighted average of the example outputs y_j near the input x, i.e.

$$\mu_\lambda(x) = \sum_{j=1}^{n} y_j K_\lambda(x - x_j) \tag{2.10}$$

where $K_\lambda(u)$ is the kernel function parameterized by a bandwidth or smoothing parameter λ. Typically assumptions are made about the function K, for instance that its integral is 1 and it is symmetric about the origin.

We note that Equation (2.10) can be seen as a restrictive form of the general RBF Equation (1.1). This is obviously the case for a carefully chosen set of RBF parameters; in particular, Equation (1.1) reduces to Equation (2.10) if we use the observed outputs y as the coefficients c, all n observations as centers, the kernel function K for each basis function h_i, the smoothing parameter λ for each basis function scale parameter σ, the simple Euclidean norm for the distance measure (i.e. $W = I$), and if we drop the polynomial term $p(\vec{Y})$.

However, there is a broader connection between RBFs and kernel regression in the context of estimation than the above correspondence of variables implies. To see this, consider the solving the linear portion of the RBF equations once we have fixed the parameters "inside" the basis functions. At this point Equation 1.1 is just

$$y = \vec{c} \cdot \vec{h}(\vec{x}; W, \vec{z}, \vec{\sigma}) \tag{2.11}$$

which must be satisfied at every data point $(\vec{x}, y)$. If we chose the typical least squares solution to this linear problem this becomes

$$y = \vec{y}(H^T H)^{-1} H^T \cdot \vec{h}(\vec{x}; W, \vec{z}, \vec{\sigma}) \tag{2.12}$$

where $\vec{y}$ is the vector of all outputs in the data set, and H is the matrix of all basis function outputs for the data set. Thus we can think of RBF networks as being linear combinations of the outputs $\vec{y}$, and note that the kernel regression of Equation 2.10 is a dual form of the RBF representation if we choose the kernel function K such that

$$K_\lambda = (H^T H)^{-1} H^T \cdot \vec{h}(\vec{x}; W, \vec{z}, \vec{\sigma}) \tag{2.13}$$

This duality encourages us to apply results from the kernel regression literature, at least in this restrictive sense. We point out a few such results here; interested readers are encouraged to pursue others in the excellent review in Eubank (1988).

First, a basic observation is that the most important problem associated with the use of a kernel estimator is the selection of a good value for the smoothing parameter λ (and thus by our above duality principle, the RBF scale parameters σ and W). Although asymptotic results concerning the optimal choice of λ have been obtained (e.g. see Theorem 4.2 in Eubank (1988)), their practical usefulness is dubious since they rely on knowledge of the unknown function. Similarly in deriving RBF networks from regularization theory and adopting a Bayesian interpretation, Poggio and Girosi (1990) note that σ and W are in principle specified by the prior probability distribution of the data, although in practice this can be quite difficult to estimate. Thus from either perspective, a reasonable approach for these parameters is to set them from the data using some robust technique such as cross-validation.

Other kernel regression results have implications for how we choose the basis function in RBF models. In particular, the Nadaraya-Watson kernel estimator

$$\mu_\lambda(x) = \frac{\sum_{j=1}^{n} y_j K_\lambda(x - x_j)}{\sum_{j=1}^{n} K_\lambda(x - x_j)} \tag{2.14}$$

was originally derived for use when the (x_j, y_j) are independent and identically distributed as a continuous bivariate random variable (X, Y) (i.e. the x_j are *not* fixed experimental design points). Note that in this case $\mu(x)$ estimates the conditional mean $E[Y|X = x]$, and the ϵ_j will be independent but *not* identically distributed. Under certain restrictions this estimator is known to have a variety of consistency properties (see Eubank (1988)). For RBF modeling problems that fit the above assumptions, then, a natural constraint to place on the basis functions h is to normalize their outputs in the above fashion.

If our inputs (i.e. x_j's) are nonstochastic, we can be more specific and suggest a particular choice for the basis functions. It is well known that if we want to minimize the sum squared error cost function χ^2 from Equation (2.1), the asymptotically optimal choice of unparameterized kernels is the *quadratic* or *Epanechnikov* kernel (see Epanechnikov (1969))

$$K(u) = \begin{cases} 0.75(1 - u^2) & |u| \leq 1 \\ 0 & |u| > 1 \end{cases} \tag{2.15}$$

Note that to obtain this result we must make some assumption about the second moment of the kernel such as

$$\int_{-1}^{1} u^2 K(u) du = \alpha \neq 0 \tag{2.16}$$

so that we cannot shrink the kernel functions (and thus χ^2) arbitrarily small.

A final suggestion from the kernel estimation literature concerns estimation bias (i.e. $E[\mu_\lambda(x) - \mu(x)]$). It can be shown that kernels with support on the entire line have difficulties with bias *globally*, since it is typically impossible to choose one smoothing parameter λ to minimize bias everywhere. If the inputs x are unequally spaced, the situation is even worse, since the estimator will effectively be computed from different numbers of observations in different input regions. Kernels with finite support (such as the Epanechnikov kernel above), however, localize the bias problems to the boundary of the data, where specialized *boundary kernels* can often be defined to lessen the bias (see Gasser and Müller (1979) for examples). *Variable bandwidth* *estimators*, which vary λ depending on the density of x, can also help here, although it is not known how to do this optimally in general. For RBF networks this would correspond to using different input weights W in different regions of the input space, an idea we will pursue further next.

2.2.3 Relation to Normal Densities

Our analogy to kernel regression ignored details of what exactly forms the *inputs* to our basis functions, and in this section we focus on that aspect of the problem. If we assume that our input data x is now drawn from k independent multivariate normal variables X_i of dimension d and common variance Σ, i.e.

$$x \in X_i \qquad \text{for } \; X_i \sim N_d(\mu_i, \Sigma), \qquad i = 1, ..., k \tag{2.17}$$

then the joint probability densities for each population are given by

$$f_i(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_i)^T\Sigma^{-1}(x-\mu_i)\right\} \tag{2.18}$$

The correspondence to RBF Equation (1.2) is clear if we choose the basis functions h_i to be gaussians, the centers $\vec{z}_i$ to be the means μ_i, and we set the weight matrix W such that $W^T \cdot W = \Sigma^{-1}$. We could use these RBF models, for instance, to approximate any linear combination of the densities, such as standard classification rules. This analogy also suggests the use of multiple weight matrices W to handle the more general case of populations with unequal covariances.

However in general we don't know the means μ_i or the common covariance Σ of the populations, and must estimate them from the data. We can estimate Σ using the usual sample covariance matrix, although we note that since this matrix has a potentially large number of parameters (i.e. $d(d+1)/2$), it may be practical to assume that the d dimensions of the X_i are independent and thus we only need to estimate the d diagonal elements of Σ. We also caution practitioners that basis functions parameterized with a scale parameter σ may be overspecified by letting all elements of the W matrix vary as well.

Similarly, we can estimate the means μ_i using the usual sample mean statistic, if we know the population that each data point was drawn from. Otherwise, the use of a *clustering* technique suggests itself, an idea which we will explore further in the next section.

2.2.4 Heuristic Algorithm for Initial Parameter Values

In the spirit of combining the suggestions and considerations from the previous sections, we offer the following heuristic algorithm for finding reasonable initial values for RBF parameters. In crude outline the algorithm is as follows:

1. Initialize centers to randomly selected observations
2. Initialize W using either I, $\hat{\Sigma}^{-1}$, or diag($\hat{\Sigma}^{-1}$),
   depending on amount and type of data
3. Use elliptical k-means clustering (possibly on inputs *and* outputs)
   to improve centers and partition data among the centers
4. For each center, estimate local covariance matrices
   using partitions of data found in step 3.
5. Pool similar local covariance estimates.
6. If centers/partitions have changed, go to step 3.

This algorithm is a generalization of the one used in Moody and Darken (1989). In particular, it is much better suited to handling inhomogeneous multimodal data, or data with input dimensions of incommensurate scales. Some comments about the algorithm are in order:

- The clustering in step 3 may be done with the conjunction of input and output data if we suspect that density of samples in the input space does not necessarily correspond to complexity of the function.

- The definition of "similar" in step 5 typically must be rather loose unless the number of centers k is quite small relative to the amount of data we have, otherwise we can't afford to estimate separate W matrices for each basis function.

- Typically the above algorithm obviates the need for a separate scale parameter σ for each basis function. However, if σ is necessary, a reasonable rule of thumb to avoid numerical problems is to choose σ based on the local density of input samples around each center, similar to variable bandwidth kernel estimators.

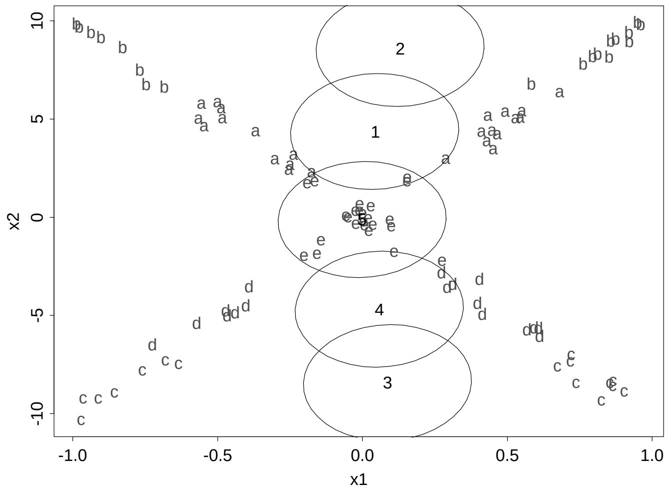
An example of this algorithm is shown in Figure 2-2. Note how a simple k-means clustering step (shown in part (a)) is dominated by the global variance structure of variable x_2, whereas one iteration of the above algorithm (shown in part (b)) serves to capture the local structure of the input data nicely, which undoubtedly would improve any subsequent fitting process.

As a final note, we caution the reader against using the above algorithm blindly. In fact we rarely use the above algorithm automatically Instead, we prefer to look at the clustered and transformed data at each stage, following the fundamental adage of statistics to know your data and methods.

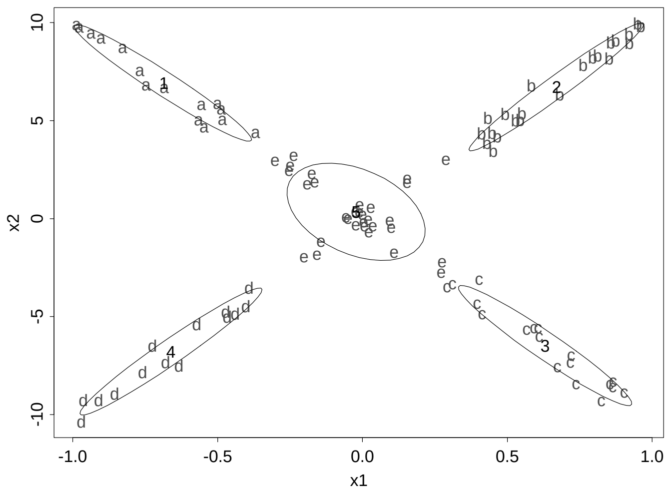
2.2.5 Starting with Simple Models

Despite our best efforts, sometimes the initial values we use for our estimation process will be quite poor, either because our data doesn't satisfy the usual assumptions, or our function is fiendishly complex. In these cases we find that an approach similar to *interior point methods* in optimization can help the speed and quality of the estimation process. The general approach is as follows:

1. Increase the smoothness of the cost function χ^2 by holding a subset of the parameters at fixed values.



(a) Isotropic clusters



(b) Iterative elliptical clusters

Figure 2-2: "X" data example of iterative elliptical k-means clustering. (a) Result of single k-means clustering step, using simple Euclidean distance. (b) Result of iterative clustering technique described in text. In both graphs, the numbers 1-5 indicate center positions, the lower case letters a-e indicate which center (cluster) each data point "belongs" to, and ellipses indicate equipotential lines for the inputs to the basis functions.

2. Solve network.

3. Use those estimates as initial values for larger/full model.

For instance we know from Section 2.2.1 that we can solve for the linear parameters c directly, thus a reasonable initial strategy is the "traditional RBF" approach of using the Euclidean norm, all the observations (or a large subset) as centers, and just solving for the c's. A second step might be to let the centers move, and then letting the scale parameters σ or the weights W move. This in fact is the strategy used for some of the larger models estimated in Chapter 3.

Chapter 3

Application One: Option Pricing

In this chapter we propose a nonparametric method for estimating the pricing formula of a derivative asset using the learning networks introduced in Chapter 1. Although not a substitute for the more traditional arbitrage-based pricing formulas, network pricing formulas may be more accurate and computationally more efficient alternatives when the underlying asset's price dynamics are unknown, or when the pricing equation associated with no-arbitrage condition cannot be solved analytically. To assess the potential value of network pricing formulas, we simulate Black-Scholes option prices and show that learning networks can recover the Black-Scholes formula from a two year training set of daily options prices, and that the resulting network formula can be used successfully to both price and delta-hedge options out-of-sample. For purposes of comparison, we estimate models using four popular methods: OLS, radial basis functions, projection pursuit regression, and multilayer perceptrons. To illustrate the practical relevance of our network pricing approach, we apply it to the pricing and delta-hedging of S&P500 futures options from 1987 to 1991, where we find some evidence that the network pricing formulas outperform the Black-Scholes formula.

3.1 Background

Much of the success and growth of the market for options and other derivative securities may be traced to the seminal papers by Black and Scholes (1973) and Merton (1973), in which closed-form option pricing formulas were obtained through a dynamic hedging argument and a no-arbitrage condition. The celebrated Black-Scholes and Merton pricing formulas have now been generalized, extended, and applied to such a vast array of securities and contexts that it is virtually impossible to provide an exhaustive catalog. Moreover, while closed-form expressions are not available in many of these generalizations and extensions, pricing formulas may still be obtainable numerically.

In each case, the derivation of the pricing formula via the hedging/no-arbitrage approach, either analytically or numerically, depends intimately on the particular parametric form of the underlying asset's price dynamics $S(t)$. A misspecification of the stochastic process for $S(t)$ will lead to systematic pricing and hedging errors for derivative securities linked to $S(t)$. Therefore, the success or failure of the traditional approach to pricing and hedging derivative securities, which we call the *parametric pricing method*, is closely tied to the ability to capture the dynamics of the underlying asset's price process.

In this chapter, we propose an alternative data-driven method for pricing and hedging derivative securities, a *nonparametric pricing method*, in which the data is allowed to determine both the dynamics of $S(t)$ and its relation to the prices of derivative securities with minimal assumptions on $S(t)$ and the derivative pricing model. To do this we will use learning networks introduced in Chapter 1 which will take as inputs the observable economic variables that influence the derivative's price, e.g. underlying asset price, strike price, and time to maturity, and whose outputs will the derivative prices. When properly trained, the network "becomes" the derivative pricing formula and can be used in same way that formulas obtained from the parametric pricing method are used: for pricing, delta-hedging, and simulation exercises.

These network-based models have several important advantages over the more traditional parametric models. First, since they do not rely on restrictive parametric assumptions such as lognormality or sample-path continuity, they may be robust to the specification errors that plague parametric models. Second, they are adaptive, and respond to structural changes in the data generating process in ways that parametric models cannot. Finally, they are flexible enough to encompass a wide range of derivative securities and fundamental asset price dynamics, yet relatively simple to implement.

Of course, all of these advantages do not come without cost - the nonparametric pricing method is highly data intensive, requiring large quantities of historical prices to obtained a sufficiently well trained network. Therefore, such an approach would be inappropriate for thinly traded derivatives, or newly created derivatives that have no similar counterparts among existing securities[1]. Also, if the fundamental asset's price dynamics are well understood and an analytical expression for the derivative's price is available under these dynamics, then the parametric formula will almost always dominate the network formula in pricing and hedging accuracy. Nevertheless, these conditions occur infrequently enough that there may still be great practical value in constructing derivative pricing formulas by learning networks.

3.2 Learning the Black-Scholes Formula

3.2.1 Motivation

Given the supposed power of learning networks, the first question we must answer is if they can approximate the unknown nonlinear functions which define derivative

[1] However, since newly created derivatives can often be replicated by a combination of existing derivatives, this is not as much of a limitation as it may seem at first.

asset prices in real world financial markets. The simplest formula for these functions that we might hope to find is of the form

$$c = f(S, X, T) \tag{3.1}$$

where c is the call option price, S is the underlying asset price, X is the exercise price of the option, and T is the time to expiration of the option. Note that this formula makes no mention of some of the usual factors for option pricing, in particular measures of the underlying asset price distribution and risk free rates of return, and thus the function f will typically be specific to a particular underlying asset and interest rate environment to implicitly capture these factors.

However, what if the true f is actually a function of time, for instance due to these implicit factors being time varying? This could easily be argued to be the case for real financial markets, and thus we will have to address these concerns when we look at real data in Section 3.3. For now, though, let's restrict our attention to an ideal world where this is not a problem, so that we can address the more basic question of how easily learning networks can approximate a *reasonable* option pricing formula. For this purpose we will use the well known option pricing formula based on the work of Black and Scholes (1973).

The standard Black-Scholes theory makes a number of assumptions about the real world. Of particular interest for our discussion is the assumption of a time *invariant* distribution for the underlying asset returns, the lognormal distribution (i.e. $\ln(S_t/S_{t-1}) \sim N(\mu, \sigma^2)$ for constants μ and σ). Also, the risk free rate of return is assumed to be constant as well. With these and other assumptions, the following formula can be derived for the price of a European call option:

$$c = S\Phi(d_1) - Xe^{-rT}\Phi(d_2) \tag{3.2}$$

where

$$d_1 = \frac{\ln(S/X) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$
$$d_2 = \frac{\ln(S/X) + (r - \sigma^2/2)T}{\sigma\sqrt{T}}$$

$\Phi(x)$ is the cumulative probability distribution function for a standardized variable, r is a constant risk free rate of return, and σ is the constant volatility of the underlying asset's returns.

In a world where the Black-Scholes assumptions hold, we should be able to find our function f in the form of Equation (3.1), although it will be specific to a particular underlying asset (i.e. σ) and interest rate environment (i.e. r). In fact, we can simply the problem even further by normalizing prices by the strike price and looking for a function of two inputs[2]:

$$c/X = f'(S/X, T) \tag{3.3}$$

Thus our strategy is to simulate underlying asset and option prices that follow the Black-Scholes assumptions, and then apply learning networks to determine if they converge satisfactorily to Black-Scholes formula when viewed in the form of Equation (3.3).

3.2.2 Calibrating the Simulations

For concreteness let us assume that the underlying assets for our simulations are replications of a "typical" American stock, with an initial price S_0 of \$50.00, an average annualized continuously compounded rate of return μ of 10%, and an an-

[2]More generally, we can appeal to Theorem 8.9 of Merton (1990) to claim that this normalization is valid at least when the stock returns are independently distributed.

nualized volatility σ of 20%. Following the lognormal distribution assumption and taking the number of days per year to be 253, we draw 506 random numbers X_t from $N(\mu/253, \sigma^2/253)$ to get 2 years of daily continuously compounded returns, which are converted to prices with the usual relation $S_t = S_0 * e^{\sum_{i=1}^{t} X_i}$ for $t > 0$.

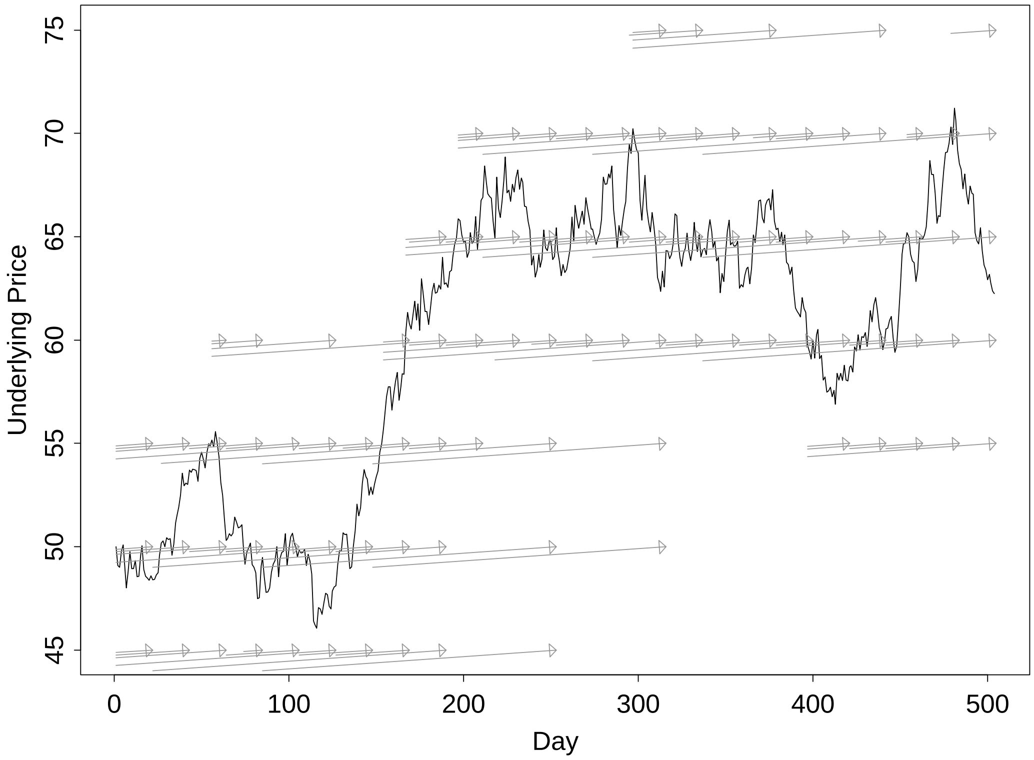


Figure 3-1: Typical simulated sample path (see text for parameters). Dashed line represents stock price, while the arrows represent the options on the stock. The y-coordinate of the tip of the arrow indicates the strike price (arrows are slanted to make different introduction and expiration dates visible).

Now we would like to bracket the stock price path with options. To obtain a representative sampling of options, we follow the Chicago Board Options Exchange (CBOE) rules for introducing options on stocks. Since a thorough description of these rules is unnecessarily detailed for our purposes, we summarize only the most salient features here. At any one time, CBOE stock options outstanding on a particular stock have 4 unique expiration dates: the current month, the next month, and the following two expirations from a quarterly schedule. Strike prices for the options are multiples of \$5[3]. When options expire and a new expiration date is introduced, the two strike prices closest to the current stock price are used. If the current price is very close to one of those strike prices[4], a third strike price is used to better bracket the current price. If the stock price moves outside of the current strike price range, another strike price is generally added for all expiration dates to bracket that price[5]. Note that we assume that all of the options generated in the above way are traded every day, although in the real world far from the money and/or long dated options are often not actively traded.

We refer to a stock price path and the associated options as one "sample path". A typical sample path is shown in Figure 3-1. We can also plot the sample path as a 3D surface if we divide stock and option prices by the appropriate strike price and consider the option price as a function of the form of Equation (3.3) - see Figure 3-2.

Our experimental setup for the simulations was as follows. A training set of 10 sample paths was generated, each using the above mentioned parameters. All options in a sample path were concatenated to produce the data matrix and response vector needed for fitting a function of the form given in Equation (3.3). Note that because the options generated for a particular sample path are a function of the random stock price path, the size of this data matrix (in terms of number of options and total number of data points) varies between sample paths. For our training set, the number of options per sample path ranged between 71 and 91, with an average of 81. The total number of data points ranged between 5227 and 6847, with an average of 6001. For each type of learning network, one network was then fit on each sample path in the training set, then tested on out-of-sample data from a test set of 5 different sample paths, again

[3]Since all of the simulated stock prices used here fall in the range of \$25 to \$200.
[4]Within \$1 in our simulations.
[5]In our simulations, this was not done for options with less than a week till expiry.

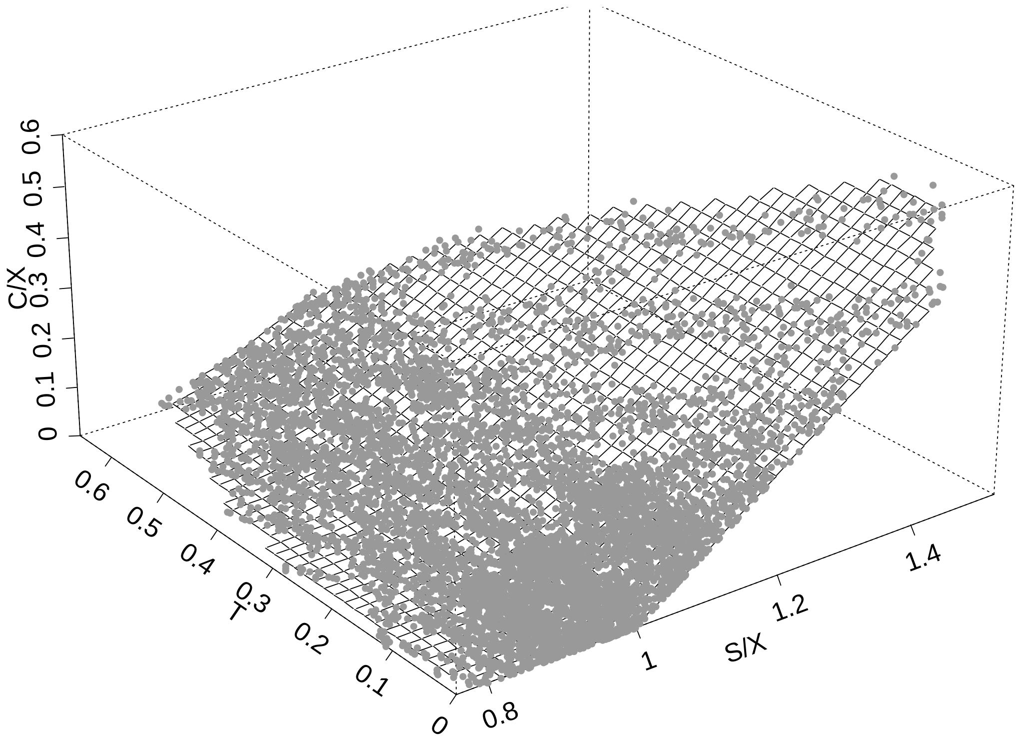


Figure 3-2: Simulated call option prices normalized by strike price and plotted versus stock price and time to expiration. Points represent daily observations. Note the denser sampling of points close to expiry is due to the CBOE strategy of always having options which expire in the current and next month.

all generated with the same parameters. This generated a matrix of results for each type of learning network, with one result for every pair of training sample path (and associated network), and test sample path. In this way it was possible for us to assess the consistency of a single network across multiple test sample paths, as well as the consistency of multiple networks (of the same type and architecture) on a single test sample path.

3.2.3 Performance Measures

How shall we measure the performance of the learning networks on the option pricing problem? First of course we are interested in how well the network's option prices match the "true" option prices. This can be measured using standard regression statistics, since the (normalized) option price is the response variable in our models. We chose to look at the R^2 and residual standard error statistics, although any "sum squared error" based statistic would give equivalent results.

A second set of measures comes from a very practical desire to see how cheaply we can *dynamically hedge* the options. By computing the networks' estimate of an option's delta (i.e. $\frac{\partial c}{\partial S}$) at each point, we can use the standard delta-adjusted hedging strategy to replicate the option through time, and compare the discounted cost of doing so with the initial price of the option. By a simple arbitrage argument, these quantities should be equal if we adjust the hedge continuously and there are no transaction costs. Thus if we define the "hedging error" H to be

$$H = c_0 - \text{hedging cost}, \tag{3.4}$$

we would like H to be as close to zero and with as small a variance as possible. A measure which meets both of these requirements is the "prediction error", which is defined as

$$\mathrm{PE} = \sqrt{\mathrm{E}(H)^2 + \mathrm{Var}(H)} \tag{3.5}$$

where E and Var are the expected value and variance operators respectively.

Note that for the RBF and MLP learning networks, delta can be computed analytically from the network by taking the derivative. For PPR, however, the use of a smoother for estimating the nonlinear functions h forces a numerical approximation of delta, which we accomplish with a first order finite difference with a increment ∂S of size 1/1000 of the range of S.

How should we combine the test results from multiple options and test sets? For instance, a weighted sum of the hedging errors would be the wealth generated from holding some delta hedged portfolio. However, we are interested in the simpler question of how this strategy works for each particular type of option, and thus we will focus primarily on summary statistics applied across options of (roughly!) the same term and degree of being in or out-of-the-money.

3.2.4 Linear and Network Pricing Formulas

Now we are set to estimate pricing formulas of the form of Equation (3.3) on the synthetic data test sets. For comparison, we start with two simple types of linear models estimated using ordinary least squares (OLS). The first type is one linear model for the entire input space, and the second type is two linear models, one for options currently in-the-money, and one for options currently out-of-the-money. Typical estimates of these models are shown in Table 3.1. Note that in terms of the implied dynamic hedging strategy, the first type of models say to buy a fixed number of shares of stock in the beginning (0.6886 in the example in Table 3.1) and hold them till expiration, regardless of the actual stock price changes. The second type of model improves on this by flipping between hedging with a small number of shares (0.1882 in the example) and a large number (0.9415 in the example) depending on whether the current stock price is greater than or less than the strike price.

The nonlinear models obtained with learning networks, on the other hand, yield estimates of option prices and deltas that are difficult to distinguish visually from

Residual Standard Error = 0.027, Multiple R-Square = 0.9098
N = 6782, F-statistic = 34184.97 on 2 and 6779 df, p-value = 0

	coef	std.err	t.stat	p.value
Intercept	-0.6417	0.0028	-231.4133	0
S/X	0.6886	0.0027	259.4616	0
T	0.0688	0.0018	38.5834	0

(a) Single linear model.

Residual Standard Error = 0.0062, Multiple R-Square = 0.9955
N = 3489, F-statistic = 385583.4 on 2 and 3486 df, p-value = 0

	coef	std.err	t.stat	p.value
Intercept	-0.9333	0.0012	-763.6280	0
S/X	0.9415	0.0011	875.0123	0
T	0.0858	0.0006	150.6208	0

(b) "In-the-money" linear model.

Residual Standard Error = 0.007, Multiple R-Square = 0.8557
N = 3293, F-statistic = 9753.782 on 2 and 3290 df, p-value = 0

	coef	std.err	t.stat	p.value
Intercept	-0.1733	0.0022	-80.3638	0
S/X	0.1882	0.0023	80.6965	0
T	0.0728	0.0007	108.2335	0

(c) "Out-of-the-money" linear model.

Table 3.1: Regression summaries for typical linear models.

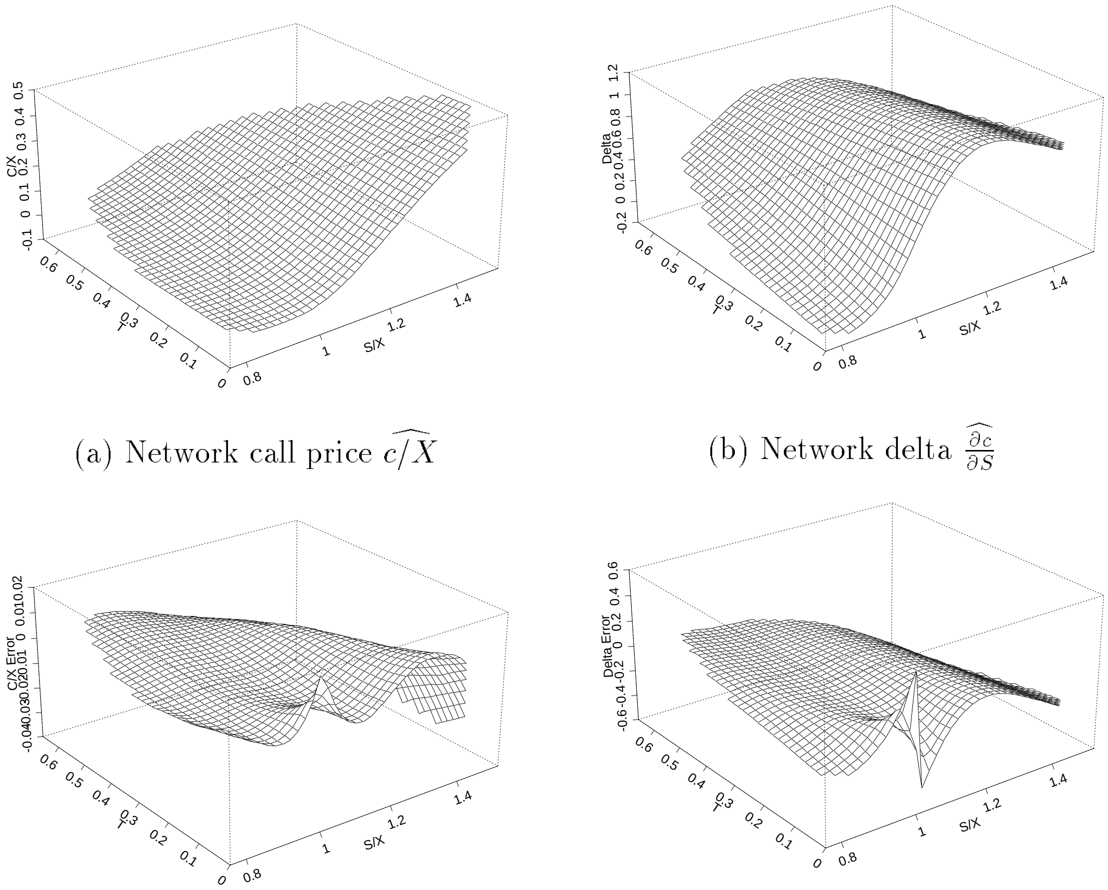
the true Black-Scholes values. An example of the estimates and errors for an RBF network is shown in Figure 3-3, which was estimated from the same data as the linear models from Table 3.1. The largest errors in these networks tend to be right at the discontinuity for options at the money at expiration, and also along the boundary of the sample points. The equation for this RBF network is

$$\begin{aligned}
\widehat{c/X} = & -0.06\sqrt{\begin{bmatrix} S/X-1.35 \\ T-0.45 \end{bmatrix}^T \begin{bmatrix} 59.79 & -0.03 \\ -0.03 & 10.24 \end{bmatrix} \begin{bmatrix} S/X-1.35 \\ T-0.45 \end{bmatrix} + 2.55} \\
& -0.03\sqrt{\begin{bmatrix} S/X-1.18 \\ T-0.24 \end{bmatrix}^T \begin{bmatrix} 59.79 & -0.03 \\ -0.03 & 10.24 \end{bmatrix} \begin{bmatrix} S/X-1.18 \\ T-0.24 \end{bmatrix} + 1.97} \\
& +0.03\sqrt{\begin{bmatrix} S/X-0.98 \\ T+0.20 \end{bmatrix}^T \begin{bmatrix} 59.79 & -0.03 \\ -0.03 & 10.24 \end{bmatrix} \begin{bmatrix} S/X-0.98 \\ T+0.20 \end{bmatrix} + 0.00} \\
& +0.10\sqrt{\begin{bmatrix} S/X-1.05 \\ T+0.10 \end{bmatrix}^T \begin{bmatrix} 59.79 & -0.03 \\ -0.03 & 10.24 \end{bmatrix} \begin{bmatrix} S/X-1.05 \\ T+0.10 \end{bmatrix} + 1.62} \\
& +0.14S/X - 0.24T - 0.01
\end{aligned} \tag{3.6}$$

Note that the centers in the RBF model are not constrained to lie within the range of the inputs, and in fact do not in the third and fourth centers in our example. PPR and MLP networks of similar complexity generate similar response surfaces, although as we shall see in the next section, each method has its own area of the input space that it models slightly more accurately than the others.

3.2.5 Out-of-Sample Pricing and Hedging

In this section we discuss the out-of-sample results of fitting the various learning networks to the simulated option data. Although the learning networks we consider are nonparametric methods, they all have one basic parameter that needs to be chosen - the number of nonlinear terms (i.e. "hidden units", basis functions, projections) to



(a) Network call price $\widehat{c/X}$

(b) Network delta $\widehat{\frac{\partial c}{\partial S}}$

(c) Call price error $\widehat{c/X} - c/X$

(d) Delta error $\widehat{\frac{\partial c}{\partial S}} - \frac{\partial c}{\partial S}$

Figure 3-3: Typical behavior of 4 nonlinear term RBF model.

use in the approximation. When we model noisy real data, we expect that there will be an "optimal" number of parameters where the complexity of the model is balanced against its out-of-sample error. For the noise free simulated data in this section, however, finding this optimal value is somewhat meaningless, and we are more interested in how fast our error measures drop with increased model complexity.

The graph of average prediction error of hedging costs versus number of parameters for the variety of models we tried is shown in Figure 3-4. The "knee" in the curve for each method occurs roughly at 4 nonlinear terms ($\approx$ 20 total parameters), and the minimum prediction error among the networks tried was for an RBF network with 40 multiquadric nonlinear terms (126 total parameters[6]). Note that the prediction error of the "true" Black-Scholes model is not zero because hedges are dynamically adjusted once daily, not continuously.

Given that learning networks with 4 nonlinear terms are sufficient to explain most of the variance in our simplified option pricing problem, it is interesting to look closer at the performance of these size networks to see which types of options (i.e. which region of input space) each network performs the best on. To do this, we divide each dimension of the input space into 3 regimes: long, medium, and short term for the time to expiration (T) axis, and in, near, and out-of-the-money for the normalized stock price (S/X) axis. Pairwise conjunctions of these regimes then form 9 groups that we can inspect. Breakpoints between the regimes are chosen to give each of the 9 groups roughly equal numbers of data points: for our simulations we choose these breakpoints to be 2 and 5 months for T, and 0.97 and 1.03 for S/X.

Average prediction error for these "duration/richness" groups for the 4 nonlinear term learning networks can be seen in Table 3.2. Inspection of the table reveals that each learning network type has its own region of the input space that its nonlinear units are particularly efficient at approximating. RBF networks, for instance, seem

[6]The centers and the parameter of the multiquadric were kept fixed in this network to reduce the number of free parameters.

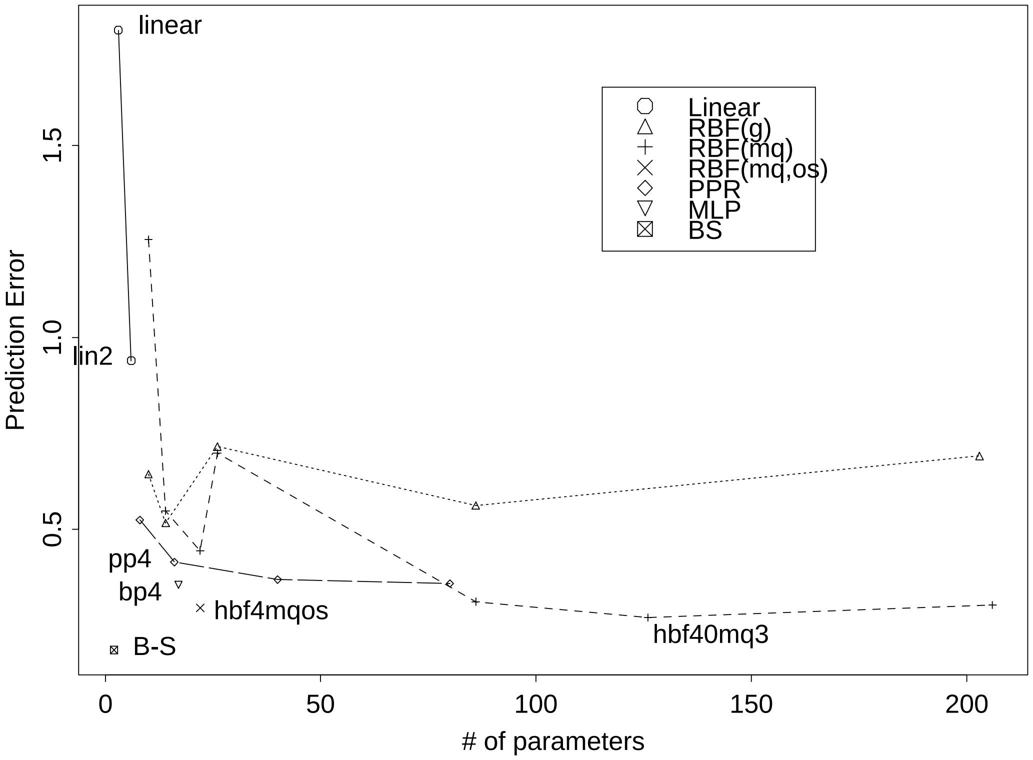


Figure 3-4: Prediction error of learning networks averaged across all training and test set pairs of simulated option data. Interesting points on the curve include "linear" for one global linear model of the data; "lin2" for two linear models, one for in-the-money options and one for out-of-the-money options; "hbf4mqos" for RBF models with 4 multiquadric centers and an output sigmoid; "pp4" for PPR models with 4 projections; "bp4" for MLP models with 4 hidden units; "hbf40mq3" for RBF models with 40 *fixed* multiquadric centers; and "B-S" for the exact Black-Scholes model using the true system parameters.

Short term	linear	hbf4mqos	pp4	bp4	B-S	$\bar{c_0}$
In-the-money	1.42	0.20	0.23	0.36	0.11	4.63
Near-the-money	1.44	0.47	0.40	0.51	0.29	1.78
Out-of-the-money	2.23	0.28	0.24	0.35	0.18	0.39

Medium term	linear	hbf4mqos	pp4	bp4	B-S	$\bar{c_0}$
In-the-money	1.70	0.17	0.84	0.35	0.22	5.80
Near-the-money	2.04	0.40	0.35	0.60	0.15	2.29
Out-of-the-money	3.19	0.40	0.41	0.37	0.29	1.02

Long term	linear	hbf4mqos	pp4	bp4	B-S	$\bar{c_0}$
In-the-money	2.36	0.39	0.88	0.61	0.26	6.39
Near-the-money	3.00	0.43	0.49	0.61	0.31	4.05
Out-of-the-money	3.67	0.59	0.58	0.40	0.29	2.30

Table 3.2: Out-of-sample average prediction error for 4 nonlinear term learning networks and the "true" Black-Scholes model. See text for definitions of groups, and Figure 3-4 for definitions of model acronyms. Units of prediction error are in dollars, and can be compared with $\bar{c_0}$, the average initial price of the options in each group.

to have substantially less error for in-the-money options, regardless of duration. PPR networks seem to outperform for short term options, and MLP networks do best on medium and long term out-of-the-money options.

3.3 An Application to S&P500 Futures Options

3.3.1 Motivation

In Section 3.2 we showed that learning networks can efficiently approximate the Black-Scholes pricing formula if the data follows the necessary assumptions, and thus we gained some confidence that learning networks can handle a reasonable option pricing world. However, the critical question for these networks is to ascertain whether or not they can capture *real* market prices better than theoretically based models when there is uncertainty about what assumptions hold, and thus what theoretical model to use.

Thus as one test of the practical relevance of our empirical modeling approach, we now apply it to S&P500 futures options prices, and compare it to the Black-Scholes model applied to the same data.

3.3.2 The Data and Experimental Setup

The data used for these experiments are daily closing prices of S&P500 futures and futures options for the 5 year period from January 1987 to December 1991. Futures prices over this period are shown in Figure 3-5. There were 24 different futures contracts and 998 futures call options active during this period[7]. The futures contracts have quarterly expirations, and on any given day 40-50 call options based on 4 different futures contracts were typically traded.

Our experimental setup for using the S&P500 data is similar to that given in Section 3.2.2 for the simulated data. We divided the S&P500 data into 10 six month subperiods for the purpose of training and testing the learning networks. Six month subperiods were chosen so the number of data points in each training set was roughly comparable to the number used in Section 3.2. Data for the second half of 1989 is shown in Figures 3-6 and 3-7. Notable differences between this data and the simulated data of Section 3.2 are the presence of "noise" in the real data and the irregular inactivity of the options (esp. near term out of the money options). For the S&P500 data, the number of futures call options per subperiod ranged from 70 to 179, with an average of 137. The total number of data points per subperiod ranged from 4454 to 8301, with an average of 6246. To limit the effects of non-stationarity and avoid data snooping, we trained a separate learning network on each of the first 9 subperiods, and tested those networks only on the data from the immediately following subperiod, thus yielding 9 test sets for each network. We also separately considered the last 7 test sets (i.e. data from July 88 to December 91) to assess the possibility of our results being strongly influenced by the October 87 crash.

[7] To simplify matters we did not use the available put options.

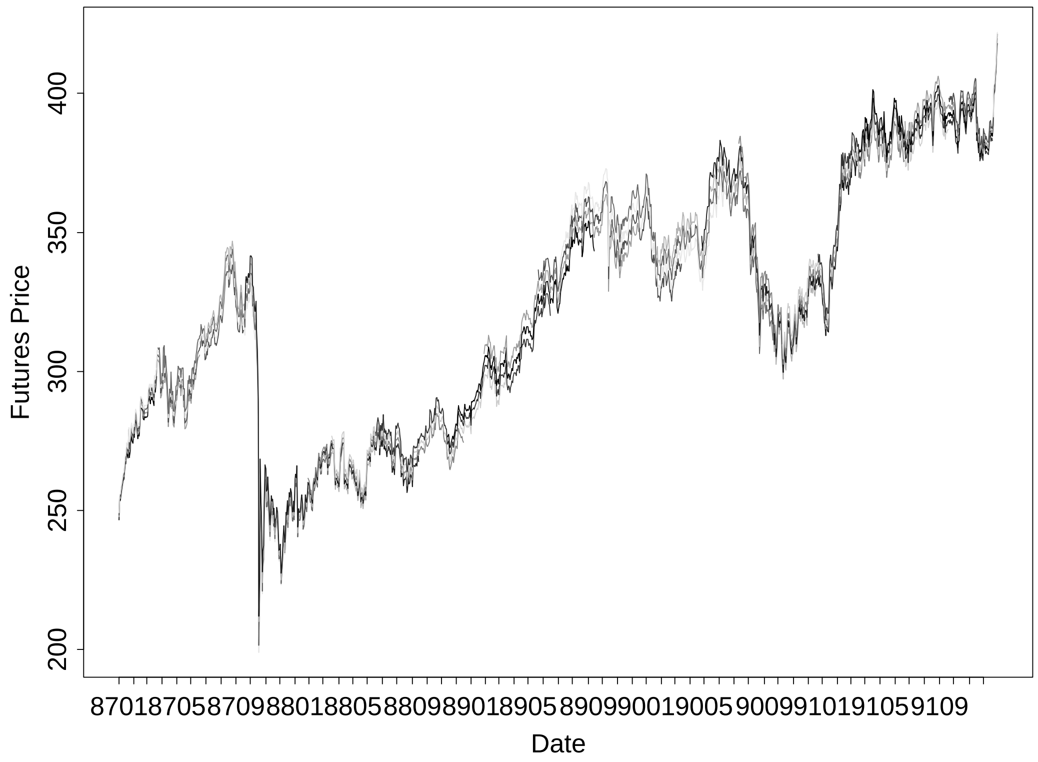


Figure 3-5: Overlay of S&P500 futures prices for all contracts active from January 1987 to December 1991.

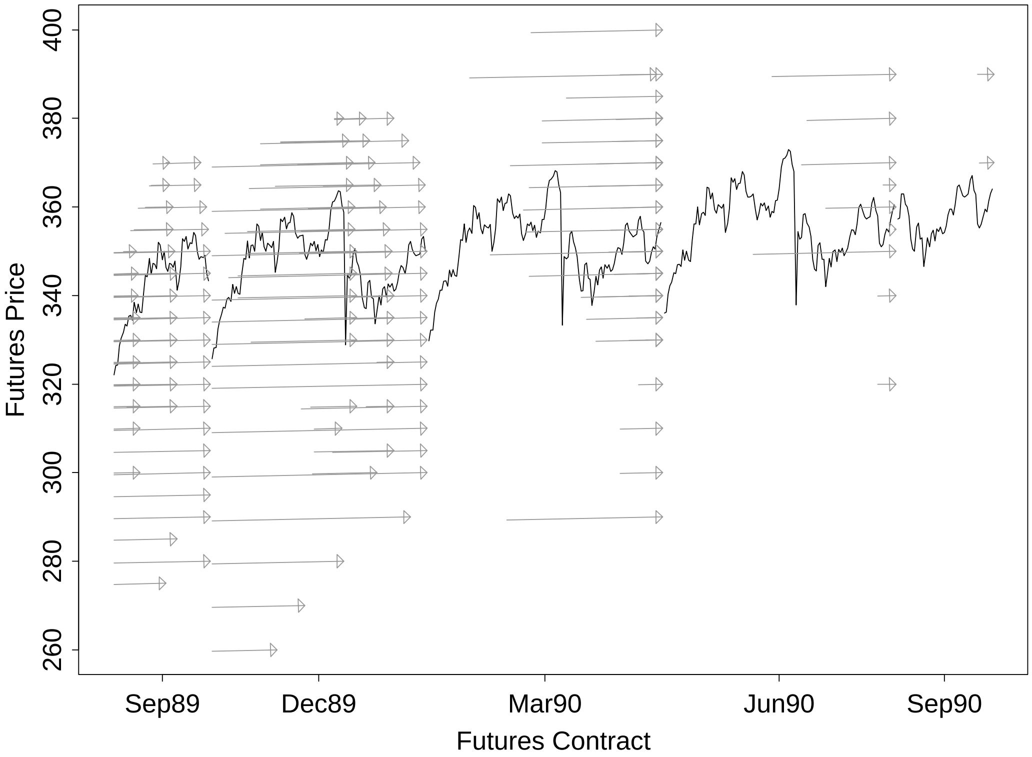


Figure 3-6: S&P500 futures and futures options active from July through December 1989. Dashed line represents futures price, while the arrows represent the options on the future. The y-coordinate of the tip of the arrow indicates the strike price (arrows are slanted to make different introduction and expiration dates visible).

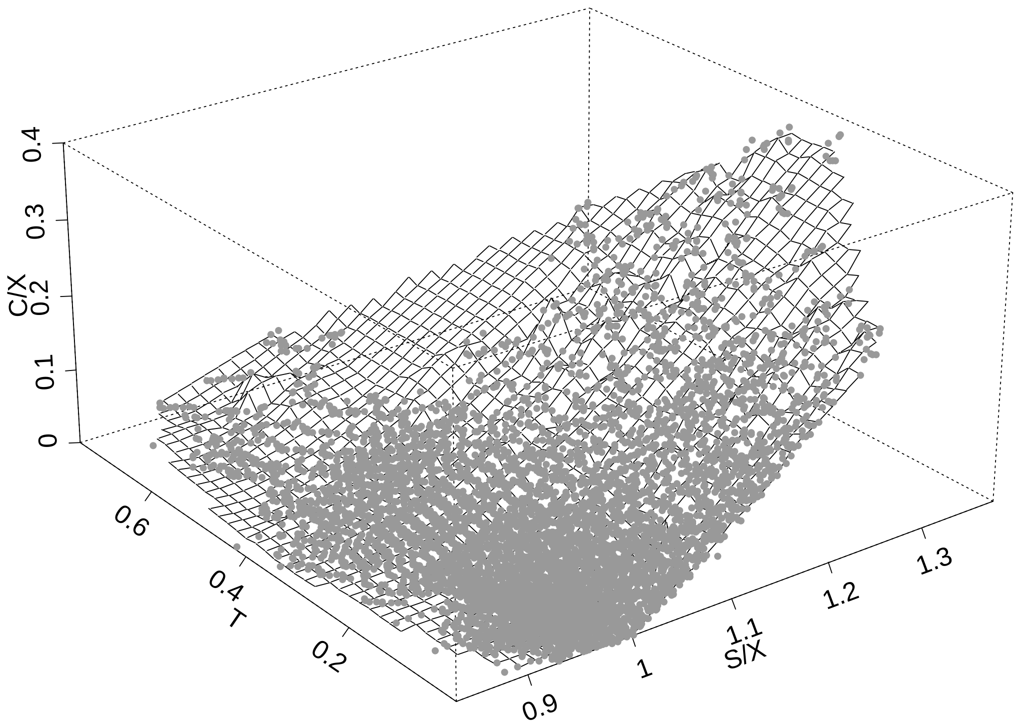


Figure 3-7: July through December 1989 S&P500 futures call option prices, normalized by strike price and plotted versus stock price and time to expiration. Points represent daily observations. Note the bumpiness of the surface, and the irregular sampling away from the money.

3.3.3 Estimating Black-Scholes

Estimating and comparing models on the S&P500 data will proceed much as it did in Section 3.2 for the linear and learning network models. Unlike our simulated world, however, the Black-Scholes model parameters r and σ must be estimated when using real data. From a strict theoretical viewpoint Black-Scholes model assumes that both of these parameters are constant over time, and thus we might be tempted to estimate them using all available past data. Few practitioners adopt this approach, however, due to substantial empirical evidence of nonstationarity in interest rate and asset price distributions. A common compromise is to estimate the parameters using only a window of the the most recent data. We follow this latter approach for the S&P500 data. Specifically, we estimate the Black-Scholes volatility σ for a given S&P500 futures contract using

$$\hat{\sigma} = s/\sqrt{60} \tag{3.7}$$

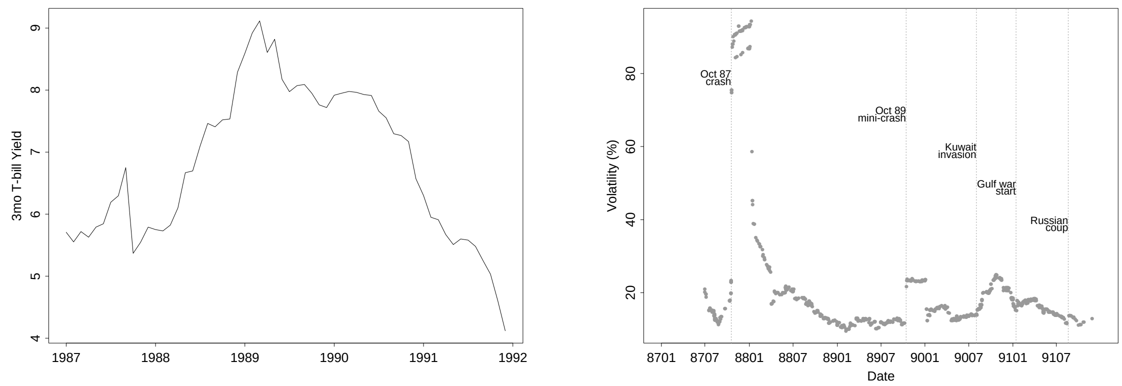
where s is the standard deviation of the 60 most recent continuously compounded daily returns of the contract. We approximate the risk free rate r to use for each futures option as the yield of the 3 month Treasury bill on the close of the month before the initial activity in that option (see Figure 3-8).

3.3.4 Out-of-Sample Pricing and Hedging

In this section we present the out-of-sample results of fitting the various models to the S&P500 data. Based on our experience with simulated data, we chose learning networks with 4 nonlinear terms as a good compromise between accuracy and complexity, although we note that re-examining this tradeoff would be interesting on the noisy S&P500 data[8].

The out-of-sample tests show some evidence that the learning networks outperform

[8] A sample re-use technique such as cross-validation would be appropriate in this context for selecting the best number of nonlinear terms.



(a) Risk free rate $\hat{r}$

(b) Volatility $\hat{\sigma}$

Figure 3-8: Black-Scholes parameters estimated from S&P500 data (see text for details). Values for $\hat{\sigma}$ fall between 9.63% and 94.39%, with a median of 16.49%.

the Black-Scholes model on this data. The delta hedging prediction error measurements broken down by duration/richness groups are shown in Tables 3.3 and 3.4. Similar to our results on the simulated data, each learning network has some portion of the input space on which it performs the best, although it is less clear here how generally those regions can be summarized. Interestingly, results from the October 1987 crash influenced subperiods still show the learning networks with lower prediction error than the Black-Scholes model, except for near term in-the-money options.

Rigorous hypothesis testing concerning relative sizes of hedging error is difficult, primarily because of the dependence of the options price paths. Selecting a single non-overlapping sequence of options for testing would solve the dependence problem, but would throw out 98% of the available options. Instead we present a less rigorous test on all of the data, but caution the reader to not to give it undue weight. Since we have hedging errors for each option and learning network, we can use a paired t-test to compare the Black-Scholes absolute hedging error on each option with the network's absolute hedging error on the same option. The null hypothesis is that the average difference of the two hedging errors is zero, and the alternate (one-sided)

Short term	linear	lin2	hbf4mqos	pp4	bp4	BS60	$\bar{c_0}$
In the money	6.70	4.92	5.04	4.52	4.94	4.42	24.26
Near the money	8.70	4.12	3.49	3.37	3.42	2.76	8.04
Out of the money	8.38	2.71	2.17	2.31	1.63	1.59	1.00

Medium term	linear	lin2	hbf4mqos	pp4	bp4	BS60	$\bar{c_0}$
In the money	9.48	6.41	6.70	6.53	5.62	5.93	35.88
Near the money	8.82	6.93	4.18	5.02	4.54	5.31	10.62
Out of the money	11.27	4.69	2.53	2.73	2.32	2.55	2.74

Long term	linear	lin2	hbf4mqos	pp4	bp4	BS60	$\bar{c_0}$
In the money	8.23	6.14	7.24	11.40	5.60	7.58	39.27
Near the money	8.55	8.58	6.37	5.55	5.17	6.18	16.14
Out of the money	12.13	7.35	3.54	5.39	4.36	5.02	6.86

Table 3.3: Delta hedging prediction error for the out-of-sample S&P500 data from July 1988 to December 1991 (i.e. excluding October 1987 crash influenced subperiods).

Short term	linear	lin2	hbf4mqos	pp4	bp4	BS60	$\bar{c_0}$
In the money	10.61	8.80	7.27	9.23	9.12	3.94	20.18
Near the money	16.30	12.73	7.77	7.48	8.08	9.09	10.76
Out of the money	23.76	8.48	7.43	5.51	5.34	10.53	5.44

Medium term	linear	lin2	hbf4mqos	pp4	bp4	BS60	$\bar{c_0}$
In the money	9.18	11.17	7.13	12.57	13.90	16.00	36.05
Near the money	24.48	13.36	7.59	5.65	5.11	6.12	12.98
Out of the money	34.31	14.80	12.30	9.44	9.64	13.46	7.45

L Long term	linear	lin2	hbf4mqos	pp4	bp4	BS60	$\bar{c_0}$
In the money	24.97	22.37	13.84	23.75	27.13	30.36	28.08
Near the money	35.06	12.93	10.78	10.11	12.27	16.03	16.98
Out of the money	29.07	14.05	9.50	8.59	8.10	10.86	10.26

Table 3.4: Delta hedging prediction error for the out-of-sample S&P500 data from July 1987 to July 1988 (i.e. October 1987 crash influenced subperiods).

Pair	t-statistic	p-value
linear vs BS60	-15.1265	1.0000
lin2 vs BS60	-5.7662	1.0000
hbf4mqos vs BS60	2.1098	0.0175
pp4 vs BS60	2.0564	0.02
bp4 vs BS60	3.7818	0.0001

Table 3.5: Paired t-test comparing relative magnitudes of absolute hedging error, using results from all S&P500 test sets (i.e. data from July 1987 to December 1991). The degrees of freedom for each test were 1299, although see comments in the text concerning dependence.

hypothesis is that the difference is positive (i.e. the learning network hedging error is smaller). Results of this (rough) test show evidence that all three learning networks outperform the Black-Scholes model, while the linear models do not (see Table 3.5).

It is also interesting to look at the computer time needed to estimate these models, although note that the code used was not particularly optimized, and we made no attempt to ascertain the best estimation method for each type of learning network. Nonetheless, second order methods seem advantageous on this problem. For instance, the MLP network gradient descent equations were updated for 10000 iterations, requiring roughly 300 minutes per network on a multiuser SUN SPARCstation II, while the Levenberg-Marquardt method for the RBF networks used from 10 to 80 iterations and took roughly 7 minutes per network. Similarly, the PPR networks (with a Newton method at the core) took roughly 120 minutes per network.

3.4 Discussion

Some caveats should be made concerning these results. Although results from our first attempt are encouraging, we cannot yet claim the general usefulness of this type of approach based only on results for a specific instrument and time period (i.e. S&P500 futures options for 1987 to 1991). Also, we are aware that there are many different theoretical derivative pricing models and practical tricks for these models that may improve the performance of our benchmark theoretical model on any given data set, and we intend to investigate some of these alternatives in the future.

That being said, we believe there is reason to be cautiously optimistic about our general approach, and feel there are a number of promising directions for future work in this area. Perhaps highest on this list is the necessity to look at other input factors for these models, for instance measures that traders commonly look at such as implied volatility. A related idea would be to fold in a time series approach to the current model in an attempt to capture temporal dependence. This could involve adding past values of inputs and outputs, or could mean modeling the squared residuals using a GARCH or related type of approach. For all of these ideas, the particular choice of market is undoubtedly important, and it may be productive to look at other markets that are not as well understood or as accurately described with current theoretical models.

Other ideas for these networks are motivated by concerns from the statistical side. First, it probably is important to fine tune the network architecture (e.g. number of nonlinear units, type of basis functions) on the specific data for each problem and/or market, although we have not done this. It would also be interesting to determine the minimal amount (e.g. number of days) of data necessary to achieve reasonable approximations. In terms of applying the learning networks more appropriately, one idea might be to use model prediction error estimates to decide when an instrument can be safely hedged using that model. Similarly, it may be advantageous to combine different models to get lower variance estimates across the input range.

Finally, we note that there are other applications for these empirical models besides their direct use as pricing formulas. In particular, a useful methodology may be to track the discrepancies between these empirical models and a favored theoretical model as a diagnostic of when and where reality is diverging from theory.

Chapter 4

Prediction and Model Confidence

What guarantee do we have that our carefully crafted empirical models will work on *unseen* data? It is well known that a model can be constructed to fit a fixed data set (i.e. the "in sample" or "training set" data) arbitrarily well, but that does not necessarily imply that the model will describe new data (i.e. the "out of sample" or "testing set" data) from that domain equally well. In this chapter we address some of the central questions about how to measure and maximize the *confidence* we have in our models and model outputs. We begin by reviewing the notion of pointwise prediction confidence intervals, and we present such an estimator for a subset of RBF networks that is asymptotically correct. However, bootstrap estimators may be more useful in practice, although they are computationally expensive and little is known about their theoretical properties. For overall model fitness we describe and evaluate the use of standard methods (e.g. cross-validation) on RBF networks. Finally we discuss the ubiquitous problem of *data mining* in this context - the tendency for us to find spurious relationships in data simply from looking at the same data too long - and note a few techniques to help in this regard, although ultimately we believe the only fully satisfactory solution is to keep some data aside untouched until the *very last* inference test.

4.1 Prediction Confidence

Consider the two example data sets and fitted linear models in Figure 4-1. In which case are we more confident that new, previously unseen points will fall close to the fitted lines? Judging simply from the relative dispersion of the training set points around the fitted lines clearly the answer in this case is data set (a), regardless of where in the input domain the new points are drawn from. Formalizing this intuitive notion will provide an objective answer to the question of how useful a model is for prediction tasks.

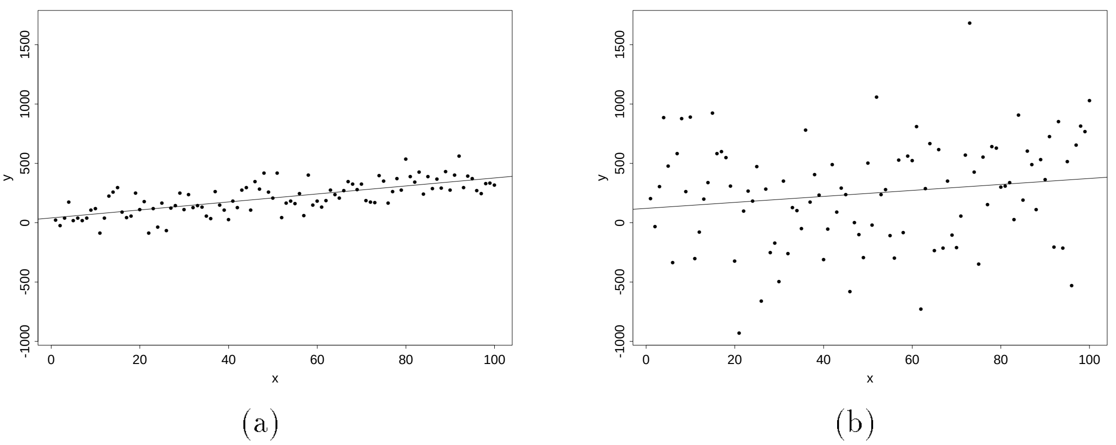


Figure 4-1: Which example linear fit is "better"?

4.1.1 Formulation

A general description of our modeling goal is to find equations in the form of the regression relationship

$$y_j = \mu(\vec{x}_j) + \epsilon_j, \qquad j = 1, ..., n \tag{4.1}$$

where once again we will restrict our discussion to the case of $(\vec{x}_j, y_j) \in \{\Re^d, \Re\}$. One way to think of the problem is that of trying to find the mean value $\mu(\vec{x})$ appropriate for each input $\vec{x}$. From this viewpoint, our informal notion of confidence can be stated precisely: at each input $\vec{x}$, we would like to know the upper and lower bounds $\hat{\mu}_u$ and $\hat{\mu}_l$ such that

$$\Pr\left\{\hat{\mu}_l(\vec{x}) \leq \mu(\vec{x}) \leq \hat{\mu}_u(\vec{x}) \ \text{ for any } \ \vec{x}\right\} = 1 - \alpha \tag{4.2}$$

for some predetermined fraction α. The pair $\hat{\mu}_u$ and $\hat{\mu}_l$ are often said to define a $100(1-\alpha)$ percent pointwise confidence interval for the mean $\mu(\vec{x})$, and we denote them with hats to emphasize that they are typically estimated from the data.

4.1.2 Linear Case

If our estimate $\hat{\mu}(\vec{x})$ of the true $\mu(\vec{x})$ is a linear function, and we assume the errors ϵ are independent and identically distributed normal random variables, it is well known that the quantity

$$\frac{\hat{\mu}(\vec{x}) - \mu(\vec{x})}{s\sqrt{\vec{x}_*^T(\mathbf{X}^T\mathbf{X})^{-1}\vec{x}_*}} \tag{4.3}$$

follows a t distribution with $n-d-1$ degrees of freedom, where $\vec{x}_*$ is the new input, s is the usual sample standard deviation of the estimated error term $\hat{\epsilon}$, and $\mathbf{X}$ is the data matrix obtained from concatenating the n column vectors $\vec{x}_j$ (see Johnson and Wichern (1988) for a nice review of the linear case). Thus our desired confidence limits are

$$\hat{\mu}_l, \hat{\mu}_u = \hat{\mu}(\vec{x}) \pm t(\alpha/2; n-d-1)s\sqrt{\vec{x}_*^T(\mathbf{X}^T\mathbf{X})^{-1}\vec{x}_*} \tag{4.4}$$

Note however that this derivation is valid only if the distribution of ϵ is not a function of the input $\vec{x}$, a point we will return to momentarily.

4.1.3 RBF Case

Can we derive similar confidence intervals for RBF network outputs? In a restricted sense we can; although our method is shown to be asymptotically correct only for RBF networks of the kernel type (see Chapter 2), in fact it may be nonetheless useful for more general cases. Before we present our result, however, two notable differences from the linear case above deserve some discussion.

A first difference with the linear case concerns the uniformity of the confidence interval across the input space. In the linear case the width of the confidence intervals is relatively uniform, partly because of our assumption of i.i.d. error ϵ, and partly because our variance estimate is global function of all of the $\vec{x}_j$'s. However, as the function complexity and input dimensionality of the problems we consider increase, it is desirable to find more *local* estimates of confidence, in the sense that they depend strongly on local input data density and error estimates. In fact this is possible for the kernel-type RBF networks, due to their local representation.

The second difference with the linear case concerns *bias*. In the linear case it is easy to show that $\hat{\mu}(\vec{x})$ is an unbiased estimator of $\mu(\vec{x})$ (i.e. $E[\hat{\mu}(\vec{x}) - \mu(\vec{x})] = 0$), so that on the average the only contribution to the error term is from the variance of the estimator. Unfortunately, this proof is not easy for the RBF case, and in fact the best results we know of in this regard are various asymptotic proofs of unbiasedness for kernel regression (e.g. Krzyżak (1986), Härdle (1990)). For this reason, from a practical viewpoint a careful assessment of the bias situation is in order for for real problems with finite data sets, despite any claims we might make about unbiasedness in showing other asymptotic results.

We now present a pointwise confidence interval result for kernel-type RBF networks based on the heuristic method of Leonard, Kramer and Ungar (1991). Following the notation of RBF Equation (1.1), let us assume we use all of the data as centers (i.e. $k = n$ and $\vec{z}_i = \vec{x}_i$ for $i = 1..n$), identical basis functions which are decreasing functions with maxima at the centers, and identical scale parameters (e.g. the σ of the gaussian) which are decreasing functions of the number of data points n. Then our conjecture is that for a new input $\vec{x}_*$ the random varaiable

$$\frac{\hat{f}(\vec{x}_*) - f(\vec{x}_*)}{Q(\vec{x}_*)} \tag{4.5}$$

where

$$\begin{aligned} Q(\vec{x}_*) &= \sqrt{A} \cdot \frac{\sum_{i=1}^{k} h(\|\vec{x}_* - \vec{z}_i\|)s_i}{\sum_{i=1}^{k} h(\|\vec{x}_* - \vec{z}_i\|)} \\ s_i &= \sqrt{\frac{\sum_{j=1}^{n} h(\|\vec{z}_i - \vec{z}_j\|)(y_j - \hat{f}(\vec{z}_j))^2}{\sum_{j=1}^{n} h(\|\vec{z}_i - \vec{z}_j\|)}} \\ \text{and} & \\ A &= \int h^2(u)du \end{aligned}$$

converges in distribution to $N(B, 1)$ where the bias term B is a function of the derivatives of f and the marginal density of the data $\vec{x}$ (denote this by $g(\vec{x})$). In cases were this bias term is judged to be small relative to the variance term above, we can then write approximate confidence intervals for the RBF case as

$$\hat{f}_l(\vec{x}), \hat{f}_u(\vec{x}) = \hat{f}(\vec{x}) \pm z(\alpha/2)Q(\vec{x}) \tag{4.6}$$

where $z(p)$ is the $100p$ quantile of the standard normal distribution.

Proof of this conjecture relies upon a similar result for kernel regression due to Härdle (1990). In particular we can see that our above conjecture is equivalent to a smoothed version of Härdle's Theorem 4.2.1, if we can show two things. First, we note that Härdle's assumptions must hold, notably that our scale parameter σ must decrease at a rate proportional to $n^{1/5}$, $f(\vec{x})$ and $g(\vec{x})$ must both be twice differentiable, the conditional variance of ϵ must be continuous around $\vec{x}_*$, and the basis functions $h(u)$ must have a bounded moment higher than the second[1]. Second,

[1] Note that the bounded moment assumption excludes increasing basis functions such as the

we must show that the spatial averaging of our conditional variance estimate s_i in $Q(\vec{x}_*)$ is correct at the data, i.e. that $Q(\vec{x}_i) \rightarrow \sqrt{A} \cdot s_i$ as $n \rightarrow \infty$ for all $i = 1..n$, but this is clearly the case if $\sigma \sim n^{-1/5}$ and the moments of $h(u)$ are bounded as above. Thus to complete the analogy we note that the marginal density $g(\vec{x})$ is approximated in the limit by $1/n \sum_{i=1}^{n} h(\|\vec{x} - \vec{x}_i\|)$ given the above conditions.

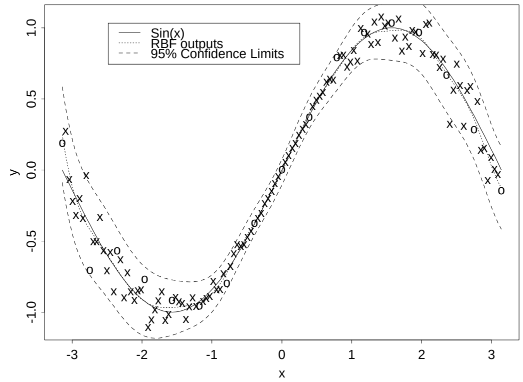
4.1.4 Examples

A few examples will serve to illustrate some properties of the RBF pointwise confidence limits. First, a 1D example of these confidence limits is shown in Figure 4-2 for data chosen on a evenly spaced grid. Here the data were generated according to the relation $y = \sin x + x \cdot N(0, (0.05)^2)$. Selection of the basis function scale parameter σ is a careful balance of the bias in $\hat{f}(\vec{x})$ from oversmoothing and the variance in our confidence limit from undersmoothing. For a suitable choice, though, the Q statistic is able to estimate the size of the heteroskedastic error term of this example quite well, something the global linear method has no chance of doing.

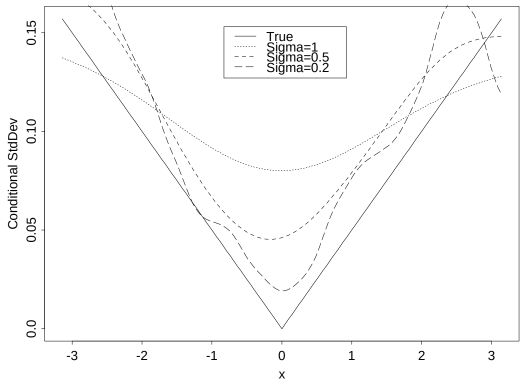
In general variance arises in our estimates not just from the inherent noise of the problem, but also from not having enough data points in some regions of the input space. In Figure 4-3 we show an example where the location of the input data are chosen randomly in such a way that the marginal density of x does *not* exactly coincide with the "interesting" features of the function. In particular, the input data x is drawn from a $N(1, 5^2)$ distribution, but the function is $y = \sin x e^{-0.1*(x+1)^2} + (x - 10)N(0, (0.01)^2)$, which has interesting features well away from $x = 1$, the point of highest marginal density. However, the increased size of the confidence intervals for lower marginal densities is quickly dominated by the variance from the noise term.

---

multiquadric.



(a)



(b)

Figure 4-2: 1D example of RBF pointwise confidence limits. Graph (a) shows the data, true function, and approximate 95% confidence limits computed by Equation (4.6) for $\sigma = 0.5$. 17 evenly spaced training points (shown with `o`'s) were used to fit the curve, and 83 new points (shown with `x`'s) were used for testing. Graph (b) shows the true versus estimated conditional standard deviation of the error term for various values of the scale parameter σ.

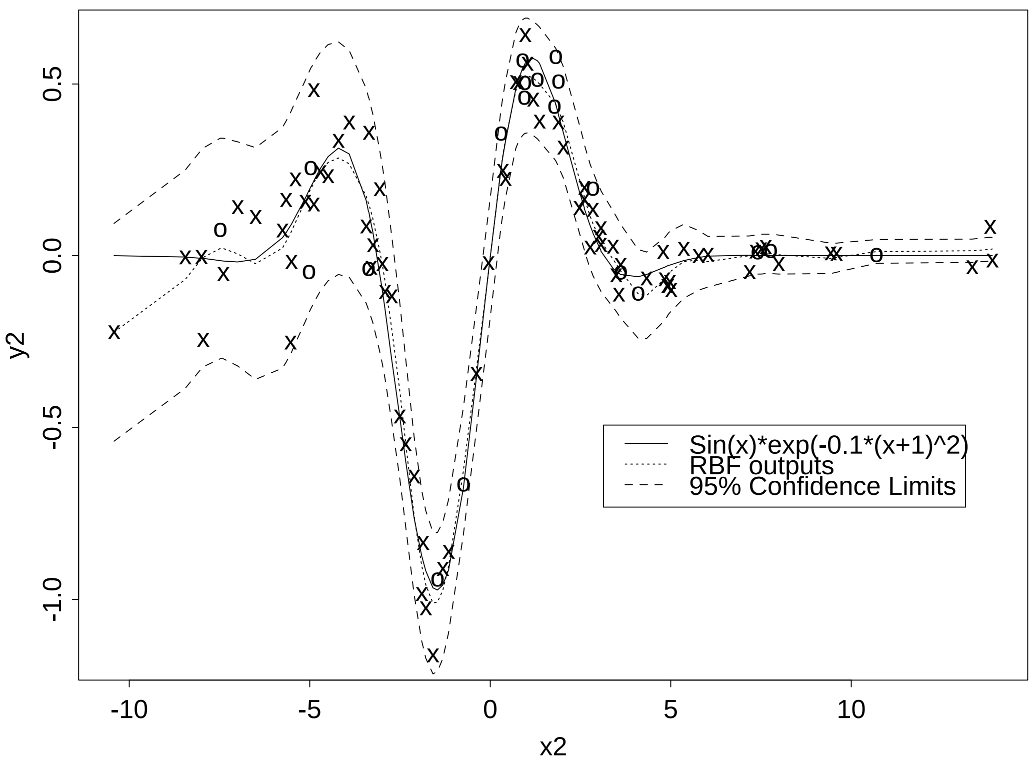


Figure 4-3: Example of RBF pointwise confidence limits for uneven data distribution. 20 randomly sampled training points (shown with `o`'s) were used to fit the curve, and 80 new points (shown with `x`'s) were used to test the confidence limits. The value of 1.0 is used here for the scale parameter σ.

4.1.5 Limitations for General RBFs

Up to this point our discussion about confidence intervals for RBF networks has been confined primarily to kernel type networks, where we are doing "local" modeling with a high density of centers/data. For many applications this is a fruitful approach, but it has significant limitations. For instance, what reason in general do we have for believing that the smoothness prior (and therefore the particular choice of basis function) that is correct for our function is also appropriate for doing the spatial smoothing of the Q statistic? In particular, as we decrease the number of centers used (for instance to control the number of free parameters per data point) this smoothness prior on the conditional variance of the error becomes increasingly important. In addition, fewer centers will generate increasingly inaccurate estimates of the conditional density $g(\vec{x})$ of the data.

Thus as we push our use of RBF networks toward the more parametric approach using only a few carefully chosen parameters, better pointwise confidence limits may be available in other ways. For instance, Härdle (1990) gives some compelling arguments for the use of various *bootstrap* estimates in the case of kernel regression, although little in the way of theoretical properties are known about them. The general idea behind bootstrap methods is to fit numerous models to resampled sets of the data, and compute confidence intervals from the quantiles of fitted values obtained at each point. Since these methods do not in general depend upon the particular form of the fitting method they could be used in the general RBF case.

4.1.6 Related Work

Methods for estimating the conditional error variance have been explored for other nonparametric techniques. From the statistics literature, Gu and Wahba (1992) derive Bayesian confidence intervals for a quite general class of smoothing splines. For multilayer perceptrons, Buntine and Weigend (1991) and Neal (1992) adopt Bayesian formulations and derive estimates of the posterior probability distributions for the network outputs based on somewhat arbitrary priors for both the free parameters and the conditional error variance. In the context of characterizing implementation considerations Lovell, Bartlett and Downs (1992) bound the variance of multilayer perceptron outputs due to errors in the free parameters and inputs, assuming that those errors are independent and small.

4.2 Model Confidence

In some sense we have "jumped the gun" by asking questions about the detailed local performance in the input space of our models. Pointwise confidence results may indeed allow the user the ability to fine tune model usage, for instance in cases where uniform convergence is either not possible or not desirable, however model diagnosis typically begins with consideration of the *overall* merit of the model and an assessment of the truth of the model assumptions. Verifying model assumptions is an important diagnostic step, for instance involving checks of the independence and normality of model residuals and the lack of obvious uncaptured "structure" in the data (see Cryer and Miller (1991) for a nice introduction), but we merely recommend it here and will not delve into it explicitly. On the other hand, as do many other authors we believe that determining the overall merit of a model is a critical step for empirically guided modeling efforts, and thus we cannot help but add some of our own comments and insights to the large literature on the subject.

4.2.1 Training Set Measures

The simplest strategy for determining the overall merit of a model amounts to calculating some global measure of model fitness, such as the χ^2 measure of Chapter 2, on the training set data. If we do this for the two examples from Figure 4-1, for instance, the χ^2 scores of 916,985 for (a) and 19,230,192 for (b) confirm our graphical impression that (a) is a much better fitting model[2].

If we fit a linear model to the data and our inputs are independent and normally distributed, we can be more precise about this inference using the F-test. This standard regression diagnostic tests the null hypothesis that all of the linear coefficients are actually zero (i.e. not needed). For a data set with n points and a linear model with p coefficients, the test is performed by computing the F-statistic

$$F = \frac{(\chi^2(y,\bar{y}) - \chi^2(\hat{y},y))/p}{\chi^2(\hat{y},y)/(n-p-1)} \tag{4.7}$$

where $\chi^2(a,b) = \sum_{i=1}^{n}(a-b)^2$, $\hat{y}$ denotes our model outputs, and $\bar{y}$ denotes the sample mean of the true outputs y, and comparing it against the F distribution with p and $n-p-1$ degrees of freedom. Performing this test on the examples from Figure 4-1, for example, reveals that we can reject the null hypothesis with near 100% confidence for model (a), but with only 90% confidence for model (b).

Can we use this F-test on the linear portion of our RBF networks (or for that matter, any of the usual linear regression diagnostic tests)? One could for instance think of the nonlinear basis functions as an elaborate preprocessing stage for a standard OLS linear model. However, satisfying the normality and independence assumptions of the test would put significant restrictions on the choices we can make for the basis functions. First, the basis function scaling parameters would have to be kept small to limit the induced dependence detween nearby basis function outputs. Second, the form of the basis functions would have to be chosen carefully to transform the input data sample density into a gaussian probability density function. Exact solutions to this problem are explored in Appendix A, but qualitatively it is clear that as the input dimensionality of the problem rises, the basis function chosen typically must decay to zero quickly to counterbalance the fact that the density of points in a d-dimensional hypersphere scales as $O(r^{d-1})$ for a distance r from the center. Nonetheless, even if

[2]Note that since the two examples have the same number of data points there is no need to worry about normalizing χ^2 to make the comparison fair.

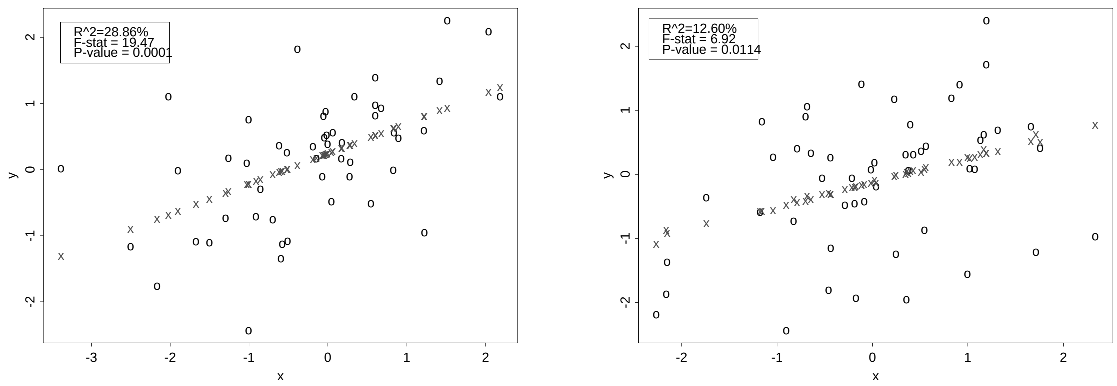
the independence and normality assumptions are questionable the F-test can be used as an approximate test of a model's overall significance, and we will in fact do so in Chapter 5.

4.2.2 Sample Reuse Techniques and Data Mining

The use of single sample methods may be adequate if our model is correctly specified, but typically we would like the data to guide our model specification, and thus we must take care to avoid *overfitting* the data, i.e. fitting an over-parameterized model. This central problem in data driven modeling is often referred to as the "generalization" problem by the neural network community, and has been extensively studied by statisticians. Sample reuse techniques such as the jacknife, the bootstrap, and cross-validation are a general class of techniques that help with this problem by repeatedly dividing the training data up into two pieces and holding one piece out to use only for testing, thus in effect testing versions of the model on "out of sample" data (see Efron and Gong (1983) for a nice treatment of these methods). Empirical tests of these different techniques often indicate mild superiority of the cross-validation style versions; for instance Härdle (1990) provides evidence that generalized cross validation may be a good choice for kernel regression methods. For this reason we make heavy use of cross-validation in our RBF and other model building.

However, cross-validation does not guarantee good out of sample test performance, even if we disregard possible bias in the cross-validation estimates due to data dependence and other issues. In fact, we are sure to find spurious relationships in data by chance simply from looking too hard at the same data. In the econometrics community this phenomena is often called "data mining", with the analogy that if you will find nuggets of "gold" if you dig enough for them (see Lovell (1983)).

To illustrate this, consider the example shown in Figure 4-4. The simple linear models shown were obtained by searching for relations between pairs of variables from a database of 1000 variables of 50 observations each by randomly selecting 1000 pairs and keeping the model with the best fit. The first search was done with a single model fit over the entire sample for each chosen pair of variables, while the second search was done similarly but using cross-validation to "ensure generalization". Unfortunately, the F-statistics for these models tell us that they are both significant (at the 99.99% and 98.86% level respectively), despite the fact that our "database" is actually a large collection of independent normally distributed random samples!



(a) (b)

Figure 4-4: Example of "data mining" by searching for good models from a fixed set of data. See text for details. Model (a) was obtained by fitting one model over the entire sample for each trial, while (b) was the best obtained from using 5-fold cross-validation. In each case, `o`'s represent the training data, and `x`'s represent the model outputs.

In statistical inference terms what has happened is that we have made Type I errors; after all, a 98.86% confidence level *means* that 1.14% of the time random variables will happen to have at least that large of an F-statistic. In actuality what we should have been doing is testing the hypothesis that our F-statistic is significant *for the number of modeling attempts we made.* In other words, if we tried fitting m models we really want there to be only a 1% chance (say) that we erroneously accept *any* of those m models. The well known Bonferroni method for simultaneous confidence intervals addresses this issue by recognizing that the least confident we should be in making m statements is if the correctness of each statement is mutually exclusive and independent of the correctness of the other statements. Thus we should have at least $100(1 - m \cdot \alpha)$ percent confidence in m simultaneous statements that we are individually confident about at the $100(1 - \alpha)$ percent level. On our example from Figure 4-4, for instance, the Bonferroni method would require an F-statistic of 19.85 or greater for even a 95% confidence level on 1000 trials, which neither of the models have.

However, since the Bonferroni method ignores any possible dependence between the constituent confidence statements it produces rather loose lower bounds. In some simple cases it is possible to quantify the effect of such dependences, but generally we will have to back off in our desire to make rigorous confidence statements simply because of the complexity of the typical development process that leads to most "final" models. Even if one is lucky enough to obtain a good results with a model on the first attempt at testing against a completely new set of data, how many of us would have the restraint to simply accept that model, without trying to fine tune it by tweaking input variables or including others, or comparing it with other modeling techniques? Our tendency to do just that is quite justifiable in terms of finding the "best model", but unfortunately it wreaks havoc with our ability to make valid inferences about the results because it is impossible to accurately reflect the effects of typical ad hoc model exploration on our inferences. Furthermore, as pointed out quite nicely in Granger and Newbold (1974), this tendency must be considered on an organizational level as well as a personal level, since for instance previous results from other researchers have just as much potential to bias our efforts and conclusions as our own past work. Ultimately these considerations point to the limited amount of data available in many domains as the fundamental restriction in what we can learn about the domain.

Chapter 5

Application Two: Price Prediction

We now have all the necessary tools to try our hand at predicting financial time series. Chapter 1 introduced our basic approach to time series forecasting, and Chapters 2 and 4 detailed the RBF algorithms and ideas that we need to build models. In this chapter we will give an answer to the question of whether or not multivariate nonlinear models offer any improvement over traditional linear and/or univariate approaches to financial time series forecasting. In order to arrive at our answer, small sample sizes and suspicions of nonstationarity of the underlying systems will encourage us to look at cross-sectional analysis in order to use as much data as possible in specifying models and making valid inferences. Ultimately, though, our answer will be a qualified "no"; although multivariate and/or nonlinear models can in some cases explain financial data significantly better than simpler models, the economic significance of the examples presented is suspect due to practical issues such as transaction costs, liquidity, and data availability. However, regardless of whether direct use of these models is profitable, they can be useful in other ways, such as testing hypotheses about the markets and placing a value on timely data.

5.1 Industry Sector Models

The search for good statistical models typically begins with some *theory* about how the underlying system of interest works, and some idea about what sort of variables might be relevant for the model. The complexity of the financial markets and our relatively inexact understanding of them may tend to make our theory rather ad hoc and ill-specified, but it is important nonetheless in terms of reducing the amount of searching we need to do, and thus minimize the data mining considerations of Chapter 4. The theory may come from a variety of sources; extending or refining other known theories, gleaning ideas from market participants, or arguing from economic first principles, to list just a few.

Take for example a commonly held theory about industry sectors; it is often claimed that some stocks in a sector tend to *lead* the price movements in that sector, while other stocks tend to *lag*. Presumably this is because prices of the smaller, less closely followed companies in the sector are slower to reflect new information about the market, although there may be other reasons. Regardless of the "true" explanation, we should be able to quantify and test this theory by modeling stock prices as a function of various sector and market indexes.

5.1.1 The Data and Notation

The data for our sector models, and the other models presented in this chapter, is daily closing prices from the Tokyo stock exchange for the period from January 4, 1989, to February 25, 1991. An example of the specific data used for the sector models is shown in Figure 5-1 for Kyokuyo Company, a medium sized Japanese fishing company, as well as the corresponding values for a number of broad market indicators, including the Nikkei 225, small, medium and large capitalization indexes, and the industry sector index which contains Kyokuyo. In the following work, the first 430 points from each time series (i.e. up to September 13, 1990) were used to build models, and the following 100 values were held back for out of sample testing.

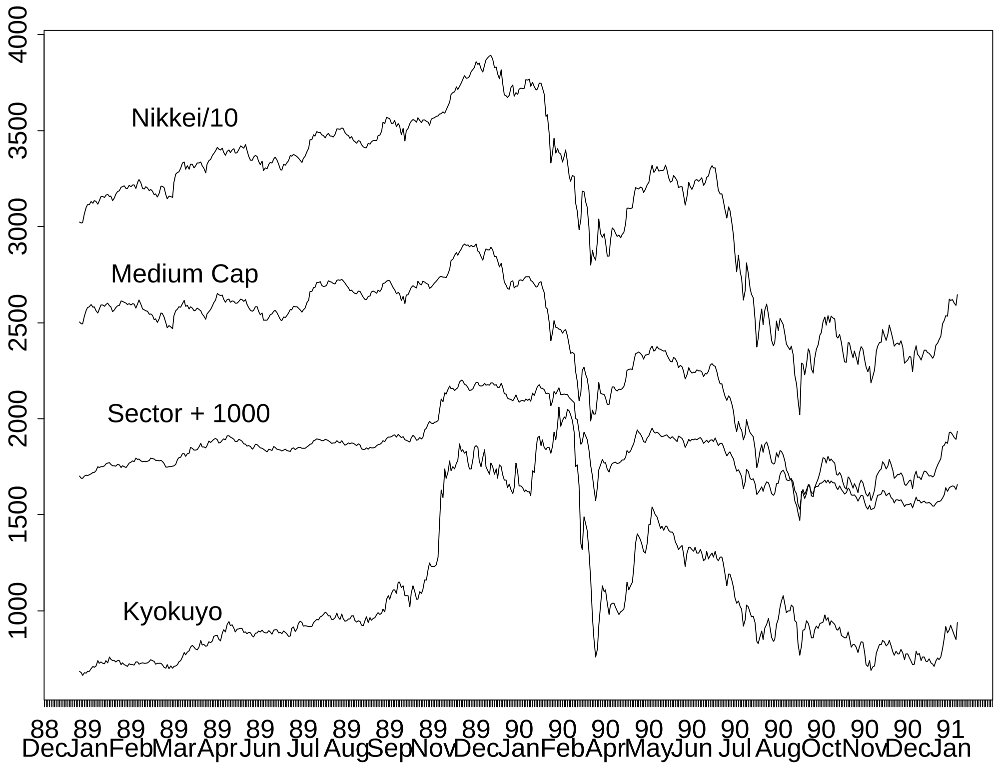


Figure 5-1: Daily closing prices of the Kyokuyo Company, its industry sector, capitalization sector, and the broad Nikkei 225 index. Vertical line separates training and testing data for sector models.

A bit of notation will be needed to refer to these series in our models. Let us denote the price of the target stock (i.e. Kyokuyo) as $P = \{p_1, p_2, \ldots, p_n\}$. Similarly, we can denote the Nikkei index with $N = \{n_t\}$, the small cap index with $S = \{s_t\}$, the medium cap index with $M = \{m_t\}$, the large cap index with $L = \{l_t\}$, and finally the sector index with $T = \{t_t\}$. Also, we denote noise or residual series with a_t or ϵ_t, depending on whether we are assuming a normal distribution or not.

Linear models generated from this raw data tend to be misspecified, however, because of the temporal dependence in these series. This misspecification typically shows up in the form of dependent residuals, a violation of standard regression assumptions. Following common practice for financial time series, we transform the price series above by taking the first difference of the logarithm, i.e.

$$r_t^p = log(p_t) - log(p_{t-1}) = log(p_t/p_{t-1}). \tag{5.1}$$

If the stock in question does not pay dividends then r^p is the stock's *continuously compounded return*. In this chapter we will attempt to find models of the target stock returns, and thus we will assume the data follows an equation of the form

$$r_t^p = f(r^n, r^s, r^m, r^l, r^t) + \epsilon_t \tag{5.2}$$

where f may use any past values of the input series. Our model then will be an estimate $\hat{f}$ of f, and we will assess the performance of the model by looking at the estimated residuals

$$\hat{\epsilon}_t = r_t^p - \hat{r}_t^p \tag{5.3}$$

where $\hat{r}_t^p$ are the predictions of our model $\hat{f}$ on the given data set.

5.1.2 Performance Measures

In measuring the performance of our predictive financial models, we like to distinguish between the performance of the *statistical model*, and the performance of any *trading system* we devise using the model. In many situations we create models that do not *directly* tell us what to do to make money, even though this may be our primary goal. The model may predict, for instance, that some stock's price will rise by 1% tomorrow[1], but it doesn't directly tell us how much (if any) of the stock to buy. Thus we adopt two different sets of performance measures: one set to assess the model fitness, and the second set to assess the profitability of simple trading systems based

[1] Or perhaps even that the price will rise between 0.5% and 1.5% with probability 95%.

on the model's predictions.

From the statistical side, most of the models will be fit using some sort of sum squared error cost function, and thus it makes sense to first look at a related error measure. We choose the usual multiple coefficient of determination, defined by

$$R^2 = 100\left[1 - \left(\sum_{i=1}^{n} \hat{\epsilon}_i^2\right) / \left(\sum_{i=1}^{n}(r_i^p - E[r^p])^2\right)\right] \tag{5.4}$$

However, when we test out of sample predictions we will be interested in both the mean and variance of the errors, and thus we also report the root mean squared prediction error defined by

$$RMSPE = \sqrt{E[\hat{\epsilon}]^2 + \mathrm{Var}[\hat{\epsilon}]} \tag{5.5}$$

where $E[]$ and Var[] are the usual sample mean and variance operators.

From the finance side, we are interested in whether or not we can make money by using the statistical model to trade. A reasonable first order trading strategy based on the above type of predictions simply buys a fixed number of shares of the stock if the prediction is positive, and sells that same number of shares short if the prediction is negative. More complex trading strategies which vary the quantity and suggest "no change" are certainly possible (especially with pointwise prediction variance estimates), but our simple buy/sell strategy will serve for our evaluation purposes. One measure of the success of this simple strategy is to calculate the percentage of times that the model predicts the correct sign of the return[2]. We also would like to know how much money we will make on the trading strategy, and for this we use the *annual rate of return*, which is defined as follows. Let r_i denote the continuously compounded rate of return of our trading strategy in period i. Then

[2]Note that the "special case" of the return being zero is relatively common for short period returns, and we do not count such points as correct unless the model explicitly predicts zero.

our annual rate of return is

$$ARR = \frac{k}{n} \sum_{i=1}^{n} r_i \tag{5.6}$$

where n is the number of points in the entire test period, and k is the number of trading time units per year (e.g. 52 for weekly trading, and roughly 253 for daily trading). However, these returns may be surprisingly large (esp. for high frequency trading) due to the fact that we are not including any transaction costs in our evaluation. Rather than attempting to find some "reasonable" transaction costs to use in the above calculations, we instead report the *break even transaction cost* (BETC), which we define as the percentage of the value of each transaction that would have to be charged for ARR to be zero for the entire test period. Note that we assume the transaction cost is paid only when the position is changed, not necessarily at every time period.

The final measure of trading performance we use is the so-called *Sharpe ratio*, which attempts to normalize returns according to the risk of the trading strategy. This measure is defined as

$$\text{SHARPE} = (E[R] - E[R^f])/\sqrt{\text{Var}[R]} \tag{5.7}$$

where R is the simple return of our trading strategy ($R = e^r - 1$ for our continuously compounded return from above), and R^f is the simple rate of return for a risk free investment over the same period[3]. The Sharpe ratio can be shown to be the best measure to maximize for an isolated investment decision if CAPM style assumptions hold (see Bodie et.al 1989), and nonetheless is widely used even when the CAPM assumptions seem tenuous.

[3]In this chapter we use 3 month deposit rates converted to the appropriate holding period for R^f, which should be negligibly different from the more textbook use of government insured bond rates for the exact same holding period as our investment.

5.1.3 Benchmark Trading Strategies

A popular model for stock price series is the random walk model, which in our notation can be formulated as

$$r_t^p = \epsilon_t \tag{5.8}$$

where ϵ is random noise with $E[\epsilon] = 0$ and $E[\epsilon_i \epsilon_j] = 0$ for $i, j \neq 0$. One implication of the model is that the best prediction of the tomorrow's price is simply today's price, which captures the notion that if any traders had true information about tomorrow's price they would act on it until the price reflected that information. Thus our random walk trading strategy is simply a prediction of zero returns for each period, and will be a useful benchmark for the R^2 and RMSPE performance measures. Note that since the SIGN statistic counts the number of points for which the predictions and actual data have the same sign (either -1, 0, or 1), its value for the unbiased random walk strategy shows the percentage of days with no change in price. Another interesting benchmark strategy for the betting measures is simply to buy the stock on the first day and hold it to the end of the period, which tells us the SIGN, ARR, and SHARPE measures for the stock itself. Performance results for both of these strategies applied to our target stock, Kyokuyo Company, are shown in Table 5.1.

5.1.4 Univariate ARMA Models

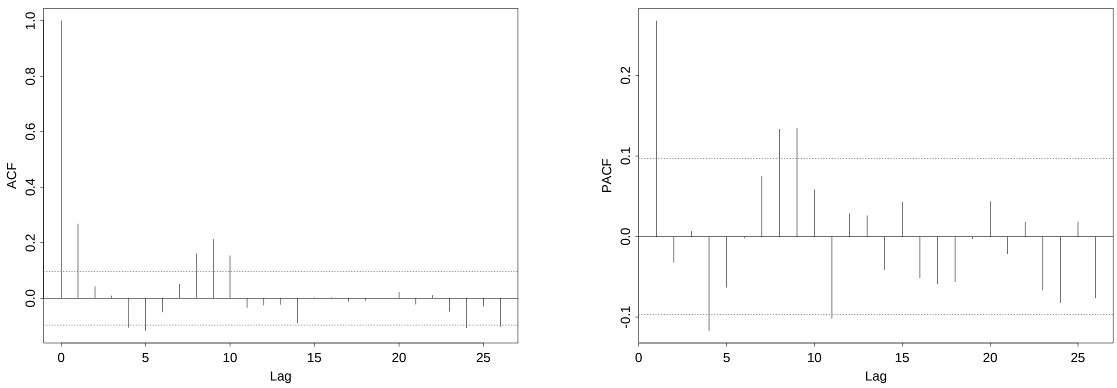
Let us begin our modeling attempts by seeing what we can do by building a linear model using only the stock returns time series itself. Box and Jenkins (1976) proposed autoregressive moving average (ARMA) models to capture the linear correlation between any specified lags of a univariate time series and the error term of the model from previous time points. Following our notation from above, we can write an ARMA(p, q) model as

$$r_t = \mu + \phi_1 r_{t-1} + \phi_2 r_{t-2} + \ldots + \phi_p r_{t-p} + \epsilon_t - \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2} - \ldots - \theta_q \epsilon_{t-q} \tag{5.9}$$

Model	R^2	RMSPE	SIGN	ARR	BETC	SHARPE
Random walk	-0.08	3.49	7.24	NA	NA	NA
	0.00	3.89	5.10	NA	NA	NA
Buy and Hold	NA	NA	45.56	24.81	52.14	0.02
	NA	NA	44.90	-2.73	-1.05	-0.01
ARMA(0,1)	7.14	3.36	46.26	109.18	0.73	0.12
	0.29	3.88	51.02	74.18	0.51	0.07
ARMA(1,0)	7.13	3.36	43.22	91.11	0.64	0.10
	-0.38	3.90	48.98	134.48	1.05	0.13
OLS NMT	17.73	3.16	53.27	196.35	1.98	0.22
	-7.03	4.02	47.96	94.01	0.85	0.09
M-estimates	8.97	3.33	52.57	176.61	1.76	0.20
	4.73	3.79	48.98	104.24	0.99	0.10
RBF 2mq	19.78	3.12	52.57	182.33	1.86	0.20
	-4.53	3.97	50.00	118.15	1.07	0.11

Table 5.1: Summary of the prediction performance for models of Kyokuyo Company daily stock prices. All measures except SHARPE are percentages. See text for discussion of models and performance measures. First line for a model indicates performance in sample; second line indicates performance out of sample.

where μ is the mean of the returns process and the ϕ's and θ's are constant coefficients. No optimal way for deciding on the order p and q of the model in general is known, but useful tools in their specification are the sample autocorrelation (ACF) and partial autocorrelation (PACF) functions. ACF measures the correlation between different lags of a time series, while PACF measures the residual correlation after the correlation implied from earlier lags is subtracted out[4]. Figure 5-2 show the ACF and PACF for the in-sample Kyokuyo returns, and indicate that first order ARMA models should suffice to capture what little linear structure is present in the data.



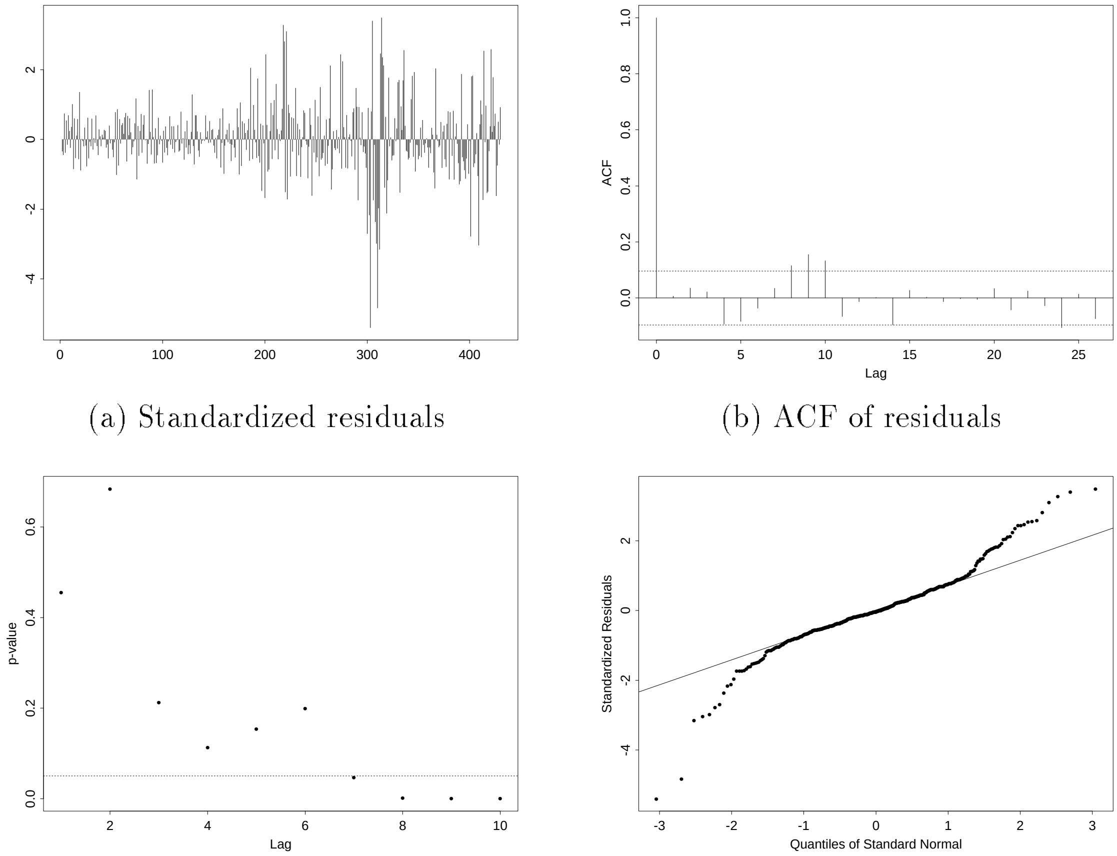
(a) (b)

Figure 5-2: Sample autocorrelation (a) and partial autocorrelation (b) functions for Kyokuyo returns. Dashed lines indicate approximate 95% significance levels.

ARMA(0,1) and ARMA(1,0) models fit to the Kyokuyo in-sample data using a conditional maximum likelihood fitting procedure yielded parameter estimates of θ_1 = -0.2724 (stderr = 0.0465) and ϕ_1 = 0.2685 (stderr = 0.0466). The usual diagnostics for these models indicate a good overall fit, and the residuals do not display much correlation structure left in them although our normality and constant variance assumptions are suspect (see Figure 5-3). Note that the constant trend term μ was not

[4]Interested readers are encouraged to consult a standard time series textbook such as Wei (1990) for more detail on the definition and use of ACF and PACF.

included in these models, based on a standard t-test of the mean of the returns (mean = 0.09%, stdev = 3.48%).



(a) Standardized residuals

(b) ACF of residuals

(c) Box-Ljung goodness of fit statistic

(d) QQ plot of residuals

Figure 5-3: Diagnostics for the ARMA(0,1) model $r_t^p = \epsilon_t + 0.2724\epsilon_{t-1}$.

One step ahead predictions for the ARMA(0,1) model are shown in Figure 5-4. The prediction performance of both models is roughly the same, which is an expected result for first order ARMA models with small coefficients (see Table 5.1). We also fit seasonal ARMA models in an attempt to capture the weaker correlation structure in the data at lags 5 and 9, but these did not yield significantly better fits than the simple first order models.

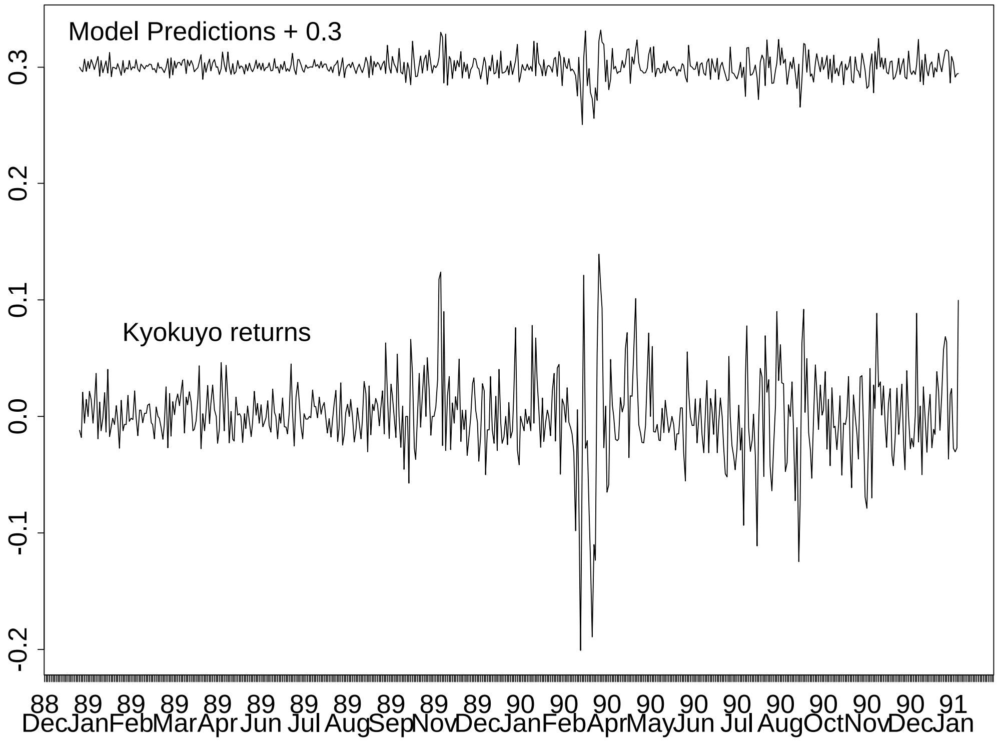


Figure 5-4: One step ahead predictions for the ARMA(0,1) model. Note that the small magnitude of the predictions is indicative of the poor explanatory power typical of stock return models.

5.1.5 Multiple Regression Models

In an attempt to improve on our univariate results, we now turn to multiple regression linear models. The simplest model using all of our time series as regressors is of the form

$$r_t^p = \mu + \beta_p r_{t-1}^p + \beta_n r_{t-1}^n + \beta_s r_{t-1}^s + \beta_m r_{t-1}^m + \beta_l r_{t-1}^l + \beta_t r_{t-1}^t + a_t \tag{5.10}$$

where we are modeling the returns at time t as a linear combination of all the time series at $t-1$. However, we cannot expect all of the coefficients β to be statistically significant, and we would like to eliminate unneeded terms. Rather than fitting the full model and one by one eliminating insignificant coefficients, we performed an all subsets regression test on the all of the available predictors, using Mallow's Cp statistic (see Figure 5-5) and the adjusted R^2 statistic to select appropriate reduced models. This technique suggests using the NMT or PNMT terms as regressors. The t-statistic of the P term in the PNMT model is too small to be significant, however, and thus we reject that model. OLS regression results for the NMT model are given in Table 5.2. Diagnostics for the NMT model are similar to those for our univariate models and show little autocorrelation in the residuals, although normality and constant variance assumptions are suspect. Unfortunately prediction performance results show significant improvement over the univariate models *only* for the in-sample period (see Table 5.1).

5.1.6 Robust Methods

A striking feature of the Kyokuyo price series is the large rise around November 1989 and the breathtaking fall around March 1990. A natural concern is that points such as these are contaminating our models, and thus we attempted to fit a robust multivariate model to the data to account for this. This model was obtained by performing M-estimation, which uses iteratively reweighted least squares to approximate the ro-

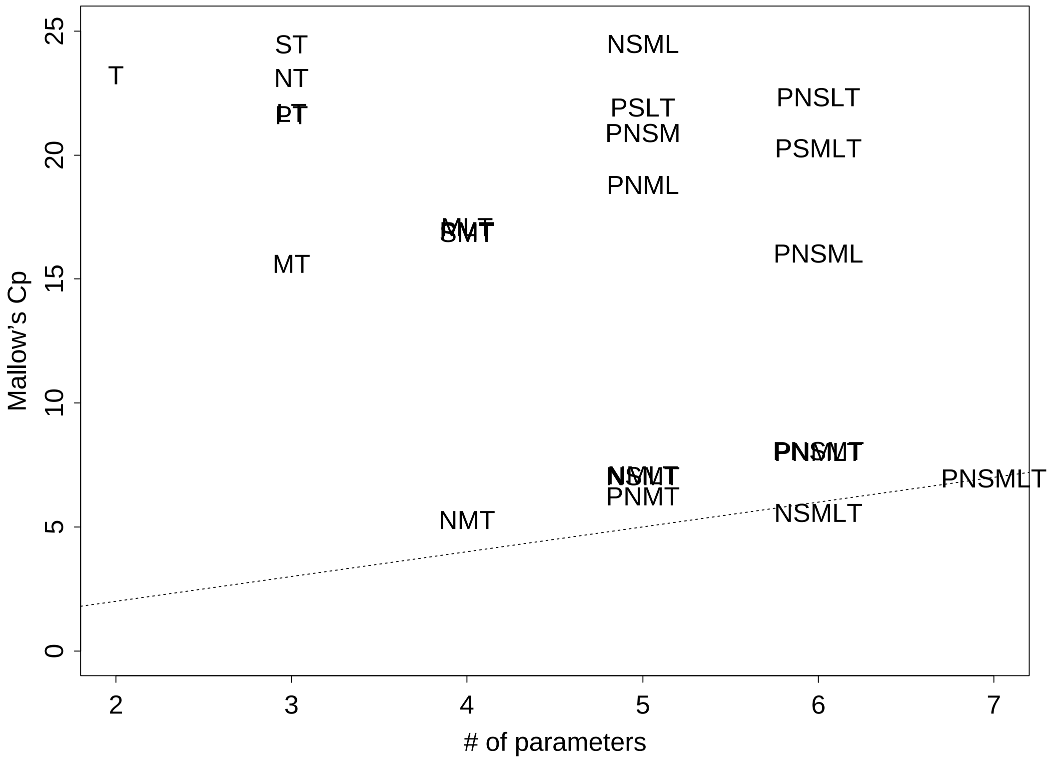


Figure 5-5: All possible subsets regression for Kyokuyo. Good models have values of Cp close to the line shown.

Residual Standard Error = 0.0317, Multiple R-Square = 0.1773
N = 428, F-statistic = 30.4658 on 3 and 424 df, p-value = 0

	coef	std.err	t.stat	p.value
Intercept	0.0019	0.0015	1.2316	0.2188
Nikkei	-1.4241	0.4061	-3.5068	0.0005
MedCap	1.8525	0.4166	4.4470	0.0000
Sector	0.4806	0.0875	5.4899	0.0000

Table 5.2: OLS regression results for NMT model of Kyokuyo returns.

bust fit, with residuals from the current fit passed through a weighting function to give the weights for the next iteration. The fitted coefficients for our NMT model are $\beta_n = -0.1194$, $\beta_m = 0.3682$, and $\beta_t = 0.1031$, and the estimated weights are shown in Figure 5-6. Indeed, the procedure gives low weight to the highly volatile stretches of the data, and some of the prediction performance measures show improvement (see Table 5.1). As a side note, if we apply the M-estimation procedure to the price series alone (i.e. a univariate ARMA(1,0) model), the coefficient obtained is very close to zero, a result consistent with the random walk model.

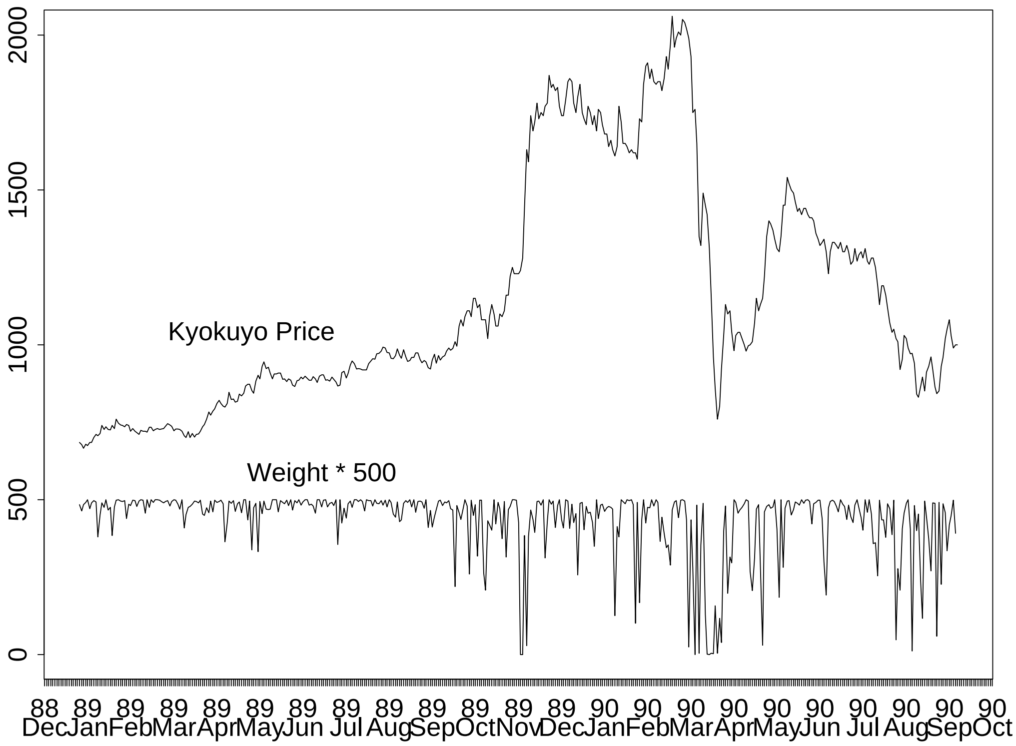


Figure 5-6: Weights estimated in robust multiple regression.

5.1.7 RBF Models

Many other global linear models could be tried on our Kyokuyo prediction problem, but it seems likely that the above methods yield results that are representative of what we would obtain from other linear methods. There is also the pervasive belief that many people have spent years in unsuccessful searches for predictive linear models. Our hypothesis is that nonlinear models of multivariate inputs is the key to making headway in this area, and thus we now put our hypothesis to the test by fitting an RBF model to the Kyokuyo data.

For comparison purposes, we use the same input data as Sections 5.1.5 and 5.1.6 (i.e. NMT inputs at time $t-1$), and fit an RBF model with 2 multiquadric centers. To lessen the number of parameters we need to estimate, we fix the input weight matrix W at square root of the inverse of the input covariance matrix. These parameter choices were made using cross-validation style model specification, as discussed in Chapter 4. The model obtained was

$$\begin{aligned}
\widehat{r_t^p} &= -0.012\sqrt{\left\|\begin{bmatrix} r_{t-1}^n \\ r_{t-1}^m \\ r_{t-1}^t \end{bmatrix} - \begin{bmatrix} -1.75\% \\ -2.52\% \\ -3.46\% \end{bmatrix}\right\|_W^2 + 0.09} \\
&\quad +0.014\sqrt{\left\|\begin{bmatrix} r_{t-1}^n \\ r_{t-1}^m \\ r_{t-1}^t \end{bmatrix} - \begin{bmatrix} 1.95\% \\ -1.75\% \\ 1.12\% \end{bmatrix}\right\|_W^2 + 0.64} \\
&\quad +0.274 r_{t-1}^n + 0.359 r_{t-1}^m + 0.513 r_{t-1}^t - 0.091
\end{aligned} \tag{5.11}$$

for

$$M = \begin{bmatrix} 217.56 & -148.63 & -21.94 \\ -148.63 & 226.66 & -9.45 \\ -21.94 & -9.45 & 51.75 \end{bmatrix}$$

Note that the model diagnostics were similar to the linear models (i.e. no apparent serial correlation in the residuals, but questionable normality). The prediction performance of this model, however, seems slightly better than the linear models, especially in the out of sample period (see Table 5.1).

5.1.8 Significance of Results

Based on the results in Table 5.1 can we conclude that our nonlinear RBF model is superior to the linear models? Can we claim that any of the models significantly outperform the random walk model? Unfortunately we cannot make any such claims, because the consistency and magnitude of outperformance is too small in each case. For instance, if we assume the excess returns of our trading strategies are normally distributed and we interpret the SHARPE measure as a t-test, we cannot reject the null hypothesis that the average excess return is zero. Similarly, if assume that the signs of our predictions are temporally independent and we interpret the SIGN measure as a one sided proportions test, Pearson's χ^2 statistic for 98 observations (the length of our out of sample test set) indicates that we would need roughly 57% correct to be 95% confident in rejecting the null hypothesis that we are merely achieving the random guessing performance of 47.45%[5], but that is not the case for any of our models.

Aside from simply finding models with much larger magnitude outperformance, the only way we can strengthen our inferences is to *use more data*[6]. There are a number of ways we can do this, although practical considerations often restrict the usefulness of each approach. For instance, we could increase the size of the testing period and hope that the performance measures did not decrease over the longer period; however, data availability and fears of nonstationarity may limit our ability to do this. Alternately we could use higher frequency data and hope the same type of input factors are relevant at the faster time scale, although typically model

[5] Since we never count returns of zero as "correct", random guessing on the out of sample period would on be $(100\% - 5.10\%)/2 = 47.45\%$ correct on average.

[6] Indeed, using more data also helps limit the data mining problem in our model specification search.

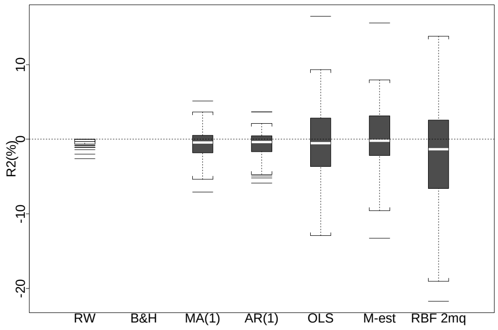
performance is quite sensitive to this. In the stock market domain, however, we have a perhaps less problematic approach available to us; we can build structurally identical models for many different stocks, and make inferences based on the aggregate performance of the collection of these models.

5.1.9 Using Other Stocks

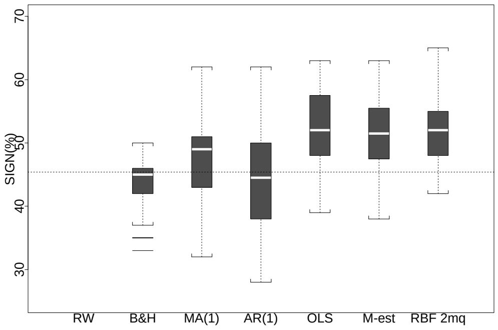
Rather than looking in the time dimension for more data to ease our inference difficulties, we suggest looking in the "space" dimension. In the sector model case this means building structurally identical models for many different stocks, which allows us to look at our performance measures across more data. To this end we estimated the models of Sections 5.1.3 thru 5.1.7 on a sample of 40 stocks randomly chosen from the Tokyo Stock Exchange First Section[7], using the same training and testing periods as for Kyokuyo. The distribution of performance measures for the different stocks do in fact bolster our confidence that the multivariate models are significantly better than both the benchmark and the linear models (see Figure 5-7).

The outperformance of the multivariate models can be shown somewhat more rigorously by extending our proportions test and t-test from Section 5.1.8 to use our expanded results. First we would like to compute a Pearson's χ^2 proportions test on our SIGN measures averaged across the 40 stocks, which are BUY&HOLD = 43.83%, ARMA(0,1) = 47.73%, ARMA(1,0) = 44.45%, OLS NMT = 52.20%, M-estimates = 51.175%, and RBF 2mq = 52.10%. A somewhat liberal test would be to pool the counts of getting the correct sign together for all 40 stocks, and test it against $40 \cdot 98 = 3920$ trials and the average random guessing SIGN measure of 45.41%. A 95% confidence interval for this one-sided test is 46.73%, which encourages us to reject the null hypothesis in all but the BUY&HOLD and ARMA(1,0) cases. However, the assumption of independence between the SIGN measures for each stock

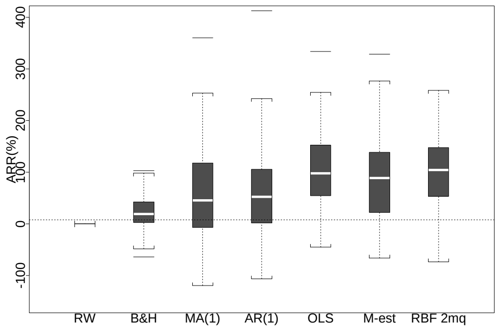
[7]The stock codes used were 1301, 1802, 1923, 2501, 3101, 3405, 3862, 4004, 4092, 4503, 4613, 4902, 5401, 5563, 5701, 5711, 5802, 6302, 6383, 6474, 6501, 6701, 6758, 6857, 7011, 7203, 7701, 7731, 8051, 8053, 8232, 8302, 8311, 8402, 8585, 8604, 8755, 8801, 9001, and 9501.



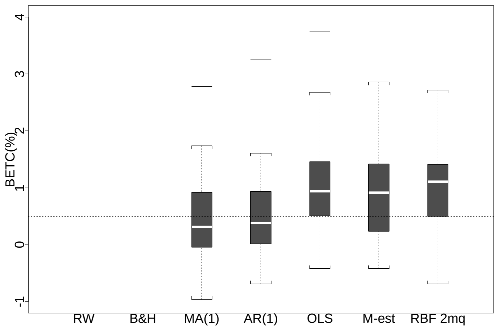
(a) R^2



(b) SIGN



(c) ARR



(d) BETC

Figure 5-7: Boxplots of out of sample prediction performance across 40 randomly chosen TSE1 stocks. For each box, the midline indicates the median of the distribution, the top and bottom of the box indicate the first and third quartiles, the whiskers extend to 1.5 times the interquartile range, and the detached horizontal lines indicate "outliers" beyond this range. Dashed horizontal reference lines are drawn at 0% for R^2, the random guessing score of 45.41% for SIGN, the risk free rate of 7.78% for ARR, and the one way break even transaction cost of 0.5% for BETC.

is problematic, because we know that even randomly selected stocks will be correlated to some extent. To gain perspective on how much this dependence is affecting us, we can also test against only 98 trials, which is tantamount to assuming the SIGN measures from different stocks are perfectly correlated. Using this more conservative test the 95% confidence level is roughly 54%, which we do not reach for any model type[8]. A final note about these tests is that the "random guessing" SIGN measure is an estimate from the stock data, and may be biased (for instance if the market follows a random walk with a nonzero mean).

In a related vein we note that the distribution of the annual rate of return (ARR) measure across the 40 stocks is roughly gaussian for each of the model types, and thus it makes sense to apply a t-test on the 40 ARR numbers for each model type to test if ARR is higher than the risk free rate[9]. We can take this approach one step further and compute a paired t-test for each pair of model types, in order to see which model type ARR means are significantly higher than other model types. To avoid the escalated chance of incorrectly rejecting the null hypothesis in these tests due simply to the large number of them, we adopt Bonferonni's method for simultaneous t-tests. Note that these tests should be used only as rough approximations because of our concern about the independence between model performance on different stocks. However, results of these approximate tests once again indicate that the multivariate models (especially OLS and RBF) are significantly better than the other models, although the difference between the multivariate models is not quite significant (see Table 5.3). A similar test can be performed on the BETC measure to show that the one way break even transaction cost is greater than 0.5% for the OLS and RBF models, but not the ARMA or M-estimate models.

What about our original hypothesis that some stocks should lag sector prices

[8]The model type with the best SIGN measure, OLS NMT, has an 89% confidence level for this test.

[9]This test corresponds to taking the Sharpe measure of an equally weighted portfolio of the 40 trading strategies for each model type.

	B&H	ARMA(0,1)	ARMA(1,0)	OLS	M-est	RBF
BUY&HOLD	0.93					
ARMA(0,1)	1.65	2.72				
ARMA(1,0)	1.99	0.91	**3.20**			
OLS NMT	**4.53**	**3.44**	**3.02**	**7.46**		
M-estimates	**3.43**	1.95	1.68	-2.38	**5.55**	
RBF 2mq	**5.33**	**3.26**	**3.14**	-0.06	1.86	**8.14**

Table 5.3: Approximate t-statistics for ARR measures across 40 stock models of each type. Diagonal elements are one-sided tests against the risk free ARR rate of 7.78% for this period. Off diagonal elements are paired t-tests that the average ARR of the "row" model type is higher than the average ARR of the "column" model type. Bold entries exceed the overall 95% confidence level of 2.994 for 21 simultaneous t-tests each with 39 degrees of freedom.

and some should lead? If this hypothesis is true, we should see that our models for some stocks should be relatively accurate and profitable, while others should not. In particular, we suspect that the laggards are small capitalization stocks. Evidence in favor of this hypothesis can be shown by computing the rank correlation between stocks' in-sample R^2 measures and their market capitalization, which is significantly negative for the OLS and M-estimate models (-0.43 and -0.46 respectively). Indeed, if only trade the 20 stocks with the highest in-sample R^2 measures, in both cases the average out-of-sample ARR measures increase (104% to 124% for OLS models, and 84% to 90% for M-estimate models). In general, though, it appears that we have delivered on only half of our promise; the use of multivariate input data has improved our predictions, but the benefit of using a nonlinear RBF model isn't clear thus far.

5.2 Margin Models

In this section we will apply the insights and methodology we developed for sector models to a second example, margin trading. One way that investors take larger bets than they otherwise might be able to do is to buy or sell stocks on margin.

Buying stock on margin amounts to paying only a percentage of the total cost of the purchase, and implicitly borrowing the remainder. Similarly selling stock on margin means that the investor need only leave a percentage of the proceeds of a short sale in their account as protection against an increase in the cost of closing out the position. In this way margin trading provides the investor with greater leverage than ordinary trading, and thus is often adopted by speculative investors.

The Japanese equity market is well known, historically at least, for the speculative fever that regularly attacks the favored "stock of the month", where that stock's price undergoes a breathtaking rise and eventual fall which cannot typically be justified from any fundamental considerations of the underlying company. To some extent these speculative price "bubbles" are thought to be caused by the (again, at least historical) tendency of the largest four brokerage houses in Japan to use their large numbers of salespeople to push particular stocks on investors. Can we use margin trading information as a surrogate for this speculative behavior and predict these disproportionate price movements? This is the goal of our modeling efforts in this section.

5.2.1 The Data

The data used for these experiments is the total balance of margined shares held (either long or short) at the end of each week for the 40 stocks introduced in Section 5.1.9, along with the corresponding stock returns, for the 111 week period from January 8, 1989 to February 17, 1991. Let us denote the margin buying balance as $B = \{b_t\}$, the margin selling balance as $S = \{s_t\}$, and the target stock price again as $P = \{p_t\}$. Exploratory analysis of this data indicate some support for our theory of a relation between margin data and price bubbles. For instance, quick jumps in margin buying and selling often occur around large jumps in price, although the exact timing of these variables is not clear (see Figure 5-8).

To model this data we again use the stock returns rather than the raw prices. Sim-

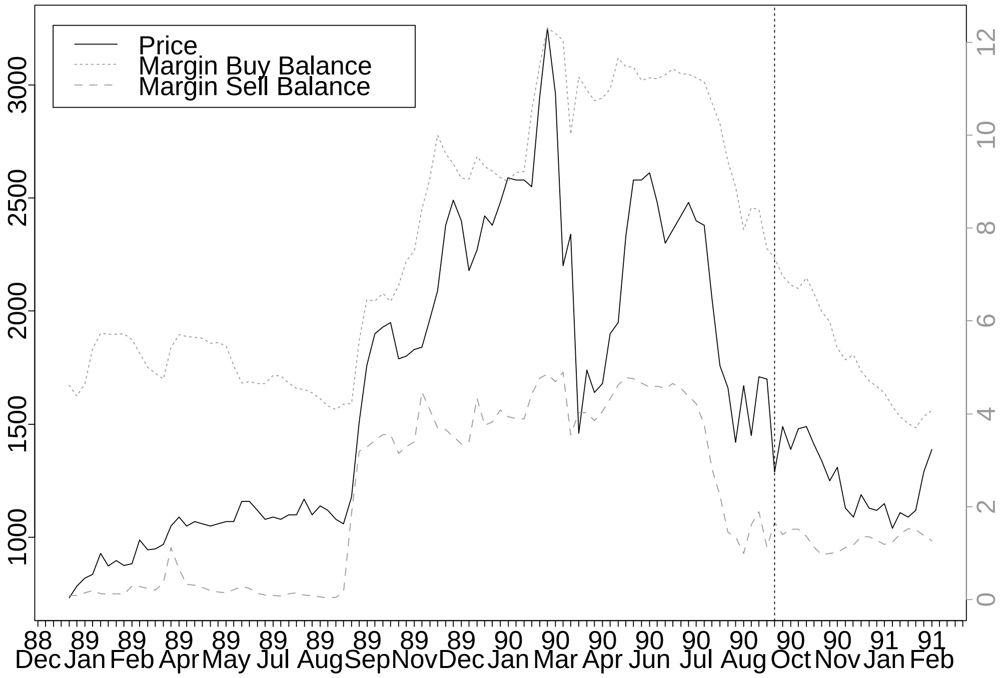


Figure 5-8: Weekly margin and price data for Nippon Chemical. Left axis shows price per share in yen, and the right axis shows margin balance expressed as a percentage of shares outstanding. Dashed vertical line shows split of data into training and out of sample testing sets.

ilarly after some experimentation with various transformations of the margin balance data, we use the "return" of that as well, although this has a less obvious meaning. As before we denote these returns as $r_t^b \equiv log(b_t/b_{t-1})$, $r_t^s \equiv log(s_t/s_{t-1})$, and $r_t^p \equiv log(p_t/p_{t-1})$. For the purposes of out of sample testing, we divide the data set at the same date as our daily sector models, yielding a training set of the first 90 weeks of returns data and a test set of the last 20 weeks.

5.2.2 Models

Following similar model identification steps as in Section 5.1, a reasonable multivariate model of the margin data was found to be

$$r_t^p = \mu + f(r_{t-1}^p, r_{t-2}^p, r_{t-1}^b, r_{t-2}^b, r_{t-1}^s, r_{t-2}^s) + \epsilon_t \tag{5.12}$$

i.e. using the values of the previous two weeks of each series to predict the price for this week. As usual, the unfortunately small size of the database encouraged us to focus attention on models with small numbers of parameters. The linear OLS versions of this model, for instance, have 7 parameters. For RBF models we chose to use 4 gaussian nonlinear units with fixed centers and fixed input weights, thus yielding 11 free parameters (4 gaussian scale parameters and 7 output coefficients).

Perhaps not surprisingly, we were not able to find good models for all of the 40 stocks in our sample. For instance, looking at the OLS models described by Equation (5.12), the F-test of the model fitness as a whole could not reject the null hypothesis at the 95% significance level for 28 out of 40 of the models. We believe our inability to find good models for some of these stocks is due simply to the fact that not all of the stocks encountered substantial margin trading activity during our training period. To support this belief, we note a rank correlation of -0.27 between the F-statistic p-value for our OLS models and the median percentage of weekly total trading volume that are margin buy trades. Thus we continue our analysis with only

	B&H	ARMA(0,1)	OLS	RBF
BUY&HOLD	**4.04**			
ARMA(0,1)	-0.99	0.46		
OLS	-2.01	-1.66	-1.39	
RBF 4g	1.00	1.67	2.16	**3.15**

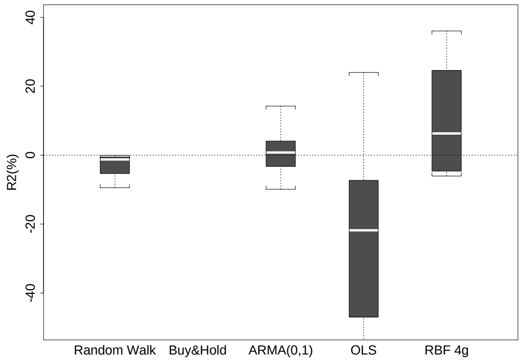
Table 5.4: Approximate t-statistics for ARR measures across 12 margin models of each type. Diagonal elements are one-sided tests against the risk free ARR rate of 7.78% for this period. Off diagonal elements are paired t-tests that the average ARR of the "row" model type is higher than the average ARR of the "column" model type. Bold entries exceed the overall 95% confidence level of 3.11 for 10 simultaneous t-tests each with 11 degrees of freedom.

the 12 stocks that from our F-test seem to have reasonable margin models. Out of sample results for all of our model types across these 12 stocks are shown in Figure 5-9. RBF models perform the best for this period, and accumulate an average annual rate of return of 293% with an average one way break even transaction cost of 2.14%. These results indicate that the nonlinear RBF models may be more appropriate for this problem than the others, although we note that the overall bull market during this period makes the buy and hold strategy a stronger than usual choice. The matrix of t-test results on the rate of return for each strategy across an equal weighted portfolio of the 12 stocks are shown in Table 5.4.

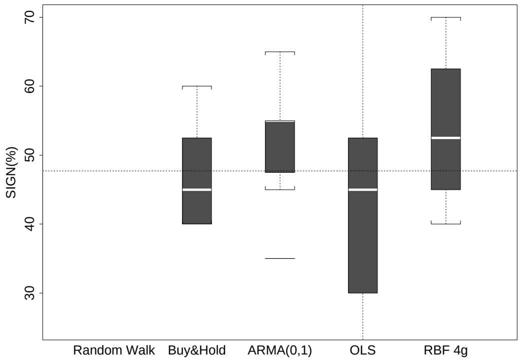
5.3 Economic Significance

Although we seem to have discovered some useful stock return models in the previous sections, can we claim any economic significance about them? Should we put them forth as refutation of some form of the efficient markets hypothesis? Unfortunately the answer to these questions is "no".

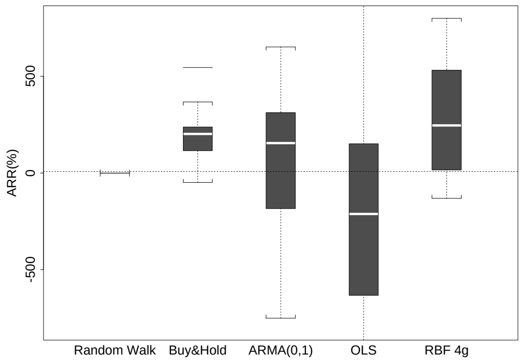
Despite our attempt to be realistic about measuring the *trading* performance of our models (including consideration of transaction costs), we have made a number of assumptions in our testing that are or may be unwarranted. For instance, in our sector



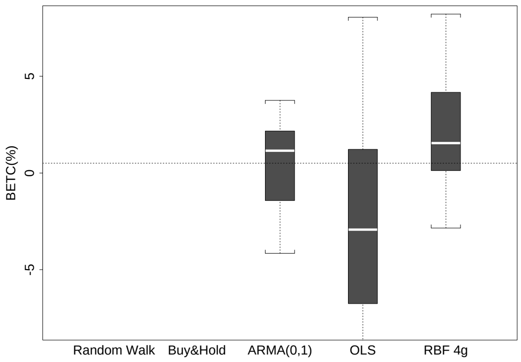
(a) R^2



(b) SIGN



(c) ARR



(d) BETC

Figure 5-9: Boxplots of out of sample prediction performance for margin models across 12 TSE1 stocks. For each box, the midline indicates the median of the distribution, the top and bottom of the box indicate the first and third quartiles, the whiskers extend to 1.5 times the interquartile range, and the detached horizontal lines indicate "outliers" beyond this range. Dashed horizontal reference lines are drawn at 0% for R^2, the random guessing score of 47.08% for SIGN, the risk free rate of 7.78% for ARR, and the one way break even transaction cost of 0.5% for BETC.

models we made the tacit assumption that when we wanted to buy a particular stock that we predicted would rise in price tomorrow, we could purchase an acceptable number of shares of it at today's closing price. Unfortunately the stocks that we identified as most likely to be profitable are probably in fact the very stocks for which this assumption is least valid - small cap stocks. Even if we could get the prices we wanted for these trades, the volume we could manage might not be enough to keep our transaction costs at a reasonable level. Thus before claiming economic significance for our sector (or any other) models, we would first have to provide evidence that trades at the required size and price could be executed, although in this case we do not have the data to do this.

Our margin models suffer from a equally disarming problem - the margin balance data is in fact not available in time to make our predictions. Although we were careful to use postdated information in our predictions - i.e. the prediction for change in price from the close of one Friday to the close of the following Friday depended only on data describing margin balances up to the first Friday - in fact the data we used is not published until Thursday of the predicted week, thus eliminating the value of our models for trading purposes.

We've touched on a few of the most common practical considerations (liquidity, asynchronous trading, data availability) that can cause problems when we actually go to implement our strategies in the real markets, but there are a multitude more. For instance related to the liquidity question, it might be crucial to factor in some estimate of how the price we obtain is influenced by the size of our transaction, especially if we plan to do relatively large trades. Another complication is that historical data is often corrected or revised after the fact, and it may be difficult to obtain the original version. Finally, some strategies may simply require the execution of more trades than existing infrastructure can handle. In general, deciding if "paper" trading strategies will work in the real world takes a surprising amount of effort and attention.

Regardless of the economic significance of our models, however, they can be quite useful in other ways. For instance in the case of our margin models, if we hypothesize different delivery days for the margin balance information we find that our hypothetical trading profits decrease monotonically as the delivery time approaches the actual time, thus in essence placing a monetary *time value* on this source of data (see Figure 5-10). In general we believe that these types of models can provide a quantitative methodology for making money if their trading characteristics are engineered to careful match an investor's inherent advantages in data availability, trading execution, or market presence.

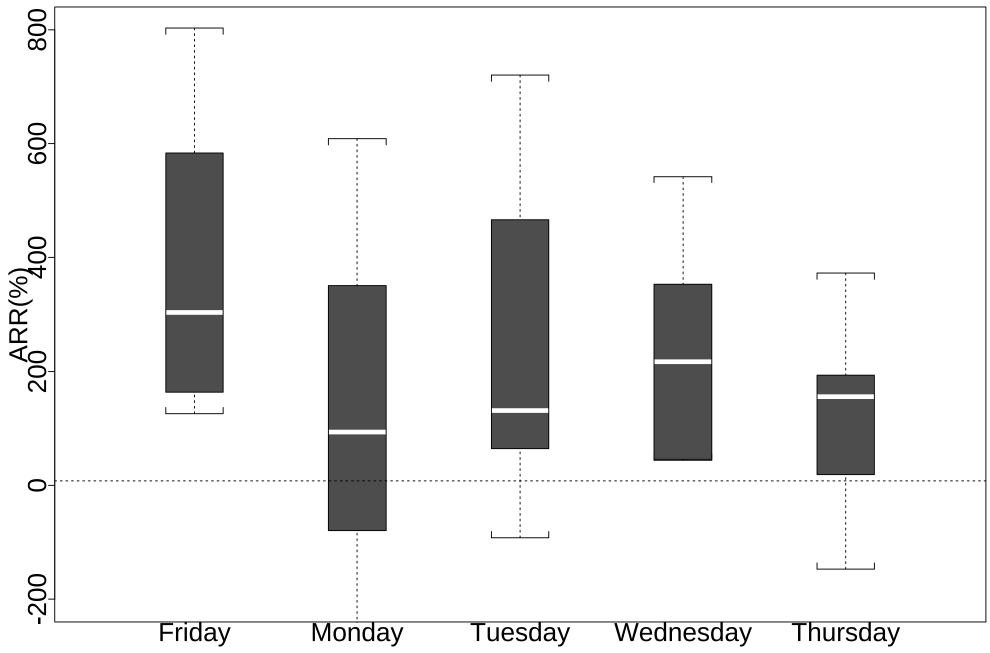


Figure 5-10: Distribution of ARR performance for the 12 best RBF margin models for different hypothetical delivery days of the margin balance data.

Chapter 6

Implementation Issues

In Chapter 2 we touched upon how we might implement the ideas presented in this thesis when we discussed estimation algorithms. In this chapter we return to this topic and further explore the issues and possibilities that arise when writing the computer programs that implement the style of statistical modeling we would like to do. Generally speaking, the advent of fast, massively parallel computer systems encourage us to pursue this nonlinear/multivariate exploratory style of modeling. The increased size of these computer systems allow us to work on much larger and more realistic data sets, and their increased speed enables us to use computationally intense algorithms that in some cases ameliorate our lack of analytic or theoretical results. However, we must take care to use the machines in sensible and efficient ways if we are to profitably pursue this style of modeling. In addition to the data mining worries discussed in Chapter 4, the nonlinearity and large size of some problem domains we might consider will cause extra numerical problems that we will have to guard against.

6.1 Parallel Computer Implementations

Much of the computation necessary for exploratory statistical modeling can be found directly in the equations describing the various estimators used. These equations typically involve the repeated evaluation of various functions, some fairly complex, across large sets of data. This style of repetitive computation has a natural analog in the *data parallel* computing paradigm of the parallel computing world, a paradigm commonly supported on massively parallel computers such as the Connection Machine system that was used to generate some of the results in this thesis. The data parallel paradigm logically assigns one data element (e.g. of an array) to each processor in the parallel machine, and evaluates functions of that data simultaneously across selected subsets of those processors. This contrasts with process level parallel computing, where different stages in an algorithm are allocated to different processors and run simultaneously, passing data from one stage to the next.

Although on a superficial level the data parallel computing paradigm is a nice match for the kinds of computations we would like to do, it is nontrivial to come up with applications that use this style in an efficient and sensible way. The novelty of the paradigm and the machines that support it mean that much of the software necessary must be written from a low level. On one hand writing software "from the ground up" on these powerful machines provides an opportunity to rethink algorithms, but on the other hand we will also be forced to reconsider issues such as data layout, floating point precision, and numerical stability of algorithms. Before undertaking such an effort it makes sense to understand where the motivations and potential benefits lie in doing so.

6.1.1 Large Data Sets

Many studies that have applied learning networks to time series analysis have used quite small data sets, such as the annual record of sunspot activity. Why do we need fast and/or parallel computers to study a data set with only a few hundred points? Indeed, a fast workstation may be sufficient in this particular case even for the most demanding learning algorithms. In general, however, we argue that for real world problems our tendency will be to try to use as much data as possible in our modeling efforts, both to improve our situation from an estimation standpoint, and to increase the complexity of functions which we can learn. Extensions to the option pricing application from Chapter 3, for instance, could involve a sampling across the operating range of *all* the feasible inputs (e.g. stock price, strike price, time to expiration, risk free rates, volatility) which could easily lead one to use data sets with millions of examples.

Thus despite initial impressions, we will often want to use much larger sets of data than we might imagine initially. Convergence results for learning networks such as those found in Barron (1991), and Girosi and Anzellotti (1993), for instance, show that the size of errors a statistical model makes is a function of the complexity of the underlying relationship being learned and the number of data points available for training. For a fixed underlying relationship, then, the way we get better models is to use more data points. However if for whatever reason we must be mostly passive in our collection of data, we must go about finding this extra data in smarter ways than simply collecting more data.

One way of doing this in our modeling context is to use observations from similar relationships in some way, either to help set some prior for the model (for example by setting the general functional form of the cross-sectional models of Chapter 5), or directly to solve a joint estimation problem (for example formulating one model for all stocks in the market). Another possible solution to the sample size problem is to create "new" data, which can be done in a number of ways. If we know the noise model of the data, we can simply resample the existing data and add the appropriate noise to obtain more data[1]. If our estimate of the underlying relationship is good, it may be possible to do even better than this by using the function estimate to interpolate between examples. Techniques such as the latter have been used to analyze chaotic time series, for instance see Farmer and Sidorowich (1989).

From an implementation viewpoint, then, we will often generate data sets large

[1] This is the idea behind naive bootstrap methods.

enough to use a few hundred processors efficiently if we are careful to lay out the data so that the data expanding style of resampling and interpolation will not cause unnecessary data motion. Another overriding efficiency consideration related to this is how much time we will have to spend reading the data into the machine and write out the results - if the I/O performance of our parallel machine is not fast enough, for instance, it may not be worth the effort of writing highly efficient computational routines.

6.1.2 Computationally Intense Algorithms

The ability to handle large data sets is nice for solving real world problems, but there is another perhaps more interesting way that parallel computers can help with exploratory statistical modeling - they provide the speed to use algorithms too computationally intense to consider otherwise. These algorithms fall into two broad classes: first, learning network specific ways to parallelize the networks' iterative, time consuming estimation algorithms; and second, general methods for identifying good models and estimating our confidence in them.

Large Matrix Algebra

Estimating RBF networks is a good case in point. RBF networks are often touted as being more computationally efficient than other learning networks (such as multilayer perceptrons) that require iterative techniques because an important stage in the estimation of RBF networks, the output coefficients, can be done directly using matrix inversion (for instance see Moody and Darken (1989)). This claim may be true if we fix the other parameters or if we can use fast heuristics for their estimation (such as the clustering approach of Moody and Darken for selecting centers), but in general we will have to adopt an iterative approach for RBF networks as well if we want to estimate all of the parameters with a general minimization method such as Levenberg-Marquardt. Although we cannot make any claims about increased efficiency in the general case, it is clear that practically we will get a lot of mileage out of the use of efficient matrix algebra routines such as matrix inversion, multiplication, and transposition. Note that this will be true even in the direct solution case if we need to select a large number of centers, and therefore have to invert a large matrix to solve for the output coefficients.

However, even if our parallel machine has a nice library of matrix algebra routines, it is not necessarily clear what is the best way to organize the overall structure of the computations necessary. Various efforts have been made to implement a multilayer perceptron estimation program on the Connection Machine system, for instance, and the best implementation to use depends on the size of the data set and the complexity of the function being learned (see Singer (1990)). If we have enough data it is likely that spreading the data over the entire machine and estimating one network at a time will be the best solution - this is the strategy used below on our implementation in Section 6.1.3.

Parallel Search

Another interesting strategy is to estimate multiple networks simultaneously. The goal of this strategy could simply be the "embarrassingly parallel" application of satisfying many model estimation requests, either from multiple users or from a model identification search for one user. Note that the judicious use of the latter can to some extent offset our lack of prior knowledge in the problem domain. Estimating multiple networks could also be pursued in the context of a parallel search for good minima on a single problem. This search could be as simple as simultaneously checking various samples of the parameter space, or it could use a more advanced algorithm to coordinate the search. One natural candidate for this kind of search is "genetic algorithms", which update a pool of candidate solutions using operators analogous to those found in biological evolution. Hillis (1990) shows how this kind of search algorithm can be parallelized for the Connection Machine by dividing the candidate solutions amongst the processors and updating them using the communication capabilities of the machine, although his system is not specifically targeted towards statistical modeling.

Sample Reuse

A final way that we can leverage the power of fast computers by estimating multiple networks is in implementing the sample reuse methods of Chapter 4. Cross-validation, for example, multiplies the number of models that must be estimated by a constant but significant factor (e.g. 10), while each model is still estimated on the bulk of the data (e.g. 90% of the points). Not coincidentally, the use of sample reuse techniques such as cross-validation is most important in a data driven, exploratory setting to minimize the dangers of data mining, and thus any system that we propose which might "mine" the data (for instance based on the parallel search strategies outlined above) should also incorporate the added expense of a sample reuse technique. Furthermore, bootstrap techniques for generating confidence intervals require estimation of 10's to 100's of identical models to collect statistics on the distributional properties of the system. Regardless of the exact mix of methods we choose, the best implementation strategy for such a joint sample reuse / parallel search system would again depend on the size of the data set and the complexity of the estimation process, although computationally there doesn't need to be any distinction between different cross-validation model instances and bootstrap model instances and parallel search model instances, as long as we can conceptually group models when computing some global values such as the cross-validation error. This type of operation is well supported by the segmented parallel prefix class of operations on the Connection Machine system, for example.

6.1.3 A CM-2 Implementation

An effort was made early in the research for this thesis to build a general purpose Radial Basis Function estimation program for the Connection Machine CM-2 system. This program was written entirely in CM Fortran, and made heavy use of the matrix algebra routines in the prepackaged CMSSL Scientific Software Library on the CM-2. A general file interface was provided to the program so that it could be used for any problem of learning from real numbered examples. The program supported the estimation techniques described in Chapter 2, including random step with gaussian noise, batch style gradient descent, and Levenberg-Marquardt.

Some comments are in order about the naive random step algorithm. As noted in Caprile and Girosi (1990), this algorithm is more efficient if different noise distributions are used for qualitatively different groups of parameters. In our implementation for RBF networks, we group the input weights, centers, scale parameters, and output coefficients separately, and allow a separate mean and standard deviation for the noise distribution for each group. In addition, we assign each group a *selection probability* which determines how what percentage of the time a parameter from that group will be chosen for modification. We view this as a possible precursor to more sophisticated algorithms which temporally alter these selection probabilities, for instance to first concentrate on center placement, then scale. We also note that significant computational savings can be made when implementing the random step algorithm by not recomputing the entire network for all parameter changes. Indeed, for a single scale parameter or output coefficient we compute the incremental change in overall error due only to that term of the equation. However, we have found experimentally that trials for the centers and input weights are virtually never accepted unless the output coefficients are also adjusted to compensate, and thus we follow all such trials by solving for the least squares optimal coefficients using the matrix inversion procedure.

Thus a basic random step iteration for RBF networks is to run the network forward for all inputs, while computing a matrix inverse to solve for the output coefficients.

Each iteration of the Levenberg-Marquardt algorithm is of similar complexity, although it requires additional calculation of the network derivatives. Timings on the Connection Machine for random step iterations are shown in Figure 6-1, although we note that this code was not particularly optimized for speed. Also note that because of the matrix inversion, the storage requirement for this operation is $\mathcal{O}(n \cdot k)$ for k centers and n data points, whereas the simple gradient descent algorithm is $\mathcal{O}(n+k)$ and thus can handle significantly more data on a fixed size machine. However, on informal comparisons of these algorithms on examples from the research in this thesis, the Levenberg-Marquardt (LM) algorithm consistently converged faster than random step or gradient descent (both in terms of number of iterations and in terms of wall clock time) despite the fact that we found it was better to adjust the variable parameter of LM (i.e. controlling the degree of use of the inverse Hessian versus the gradient) quite slowly to avoid getting too optimistic or pessimistic about the validity of LM's quadratic approximation.

For the reasons presented above, we believe that it makes sense to implement such an exploratory nonlinear statistical modeling software package on a parallel computer. It should be noted, however, that in some applications the need for interprocessor communication will be very light, and thus a network of workstations may be more appropriate from a cost-performance point of view. This is especially true for the mostly independent searches and simple models that tend to dominate early stages of the exploratory process in a noisy domain, although of course in other domains large models may be appropriate from the very beginning. Regardless, from a practical point of view a parallel computer implementation's ability to greatly speed up the turnaround time for experiments, ease the handling large data sets, and permit the use of computationally intense algorithms give it a definite advantage when compared to existing serial implementations. Zhang and Hutchinson (1993), for example, argue that this advantage was a reason for their success in the 1992 Santa Fe Institute Time Series Prediction Competition. We also note that in terms of gaining broad acceptance

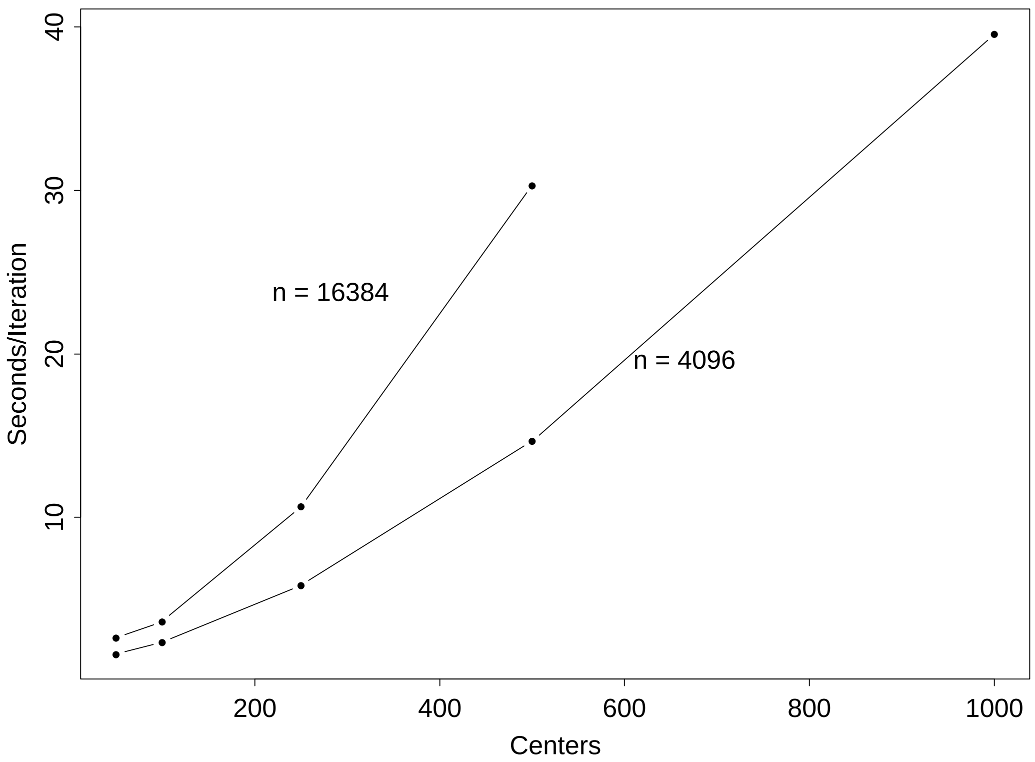


Figure 6-1: Timings for random step algorithm on a 16K processor Connection Machine CM-2 for two different numbers of data points n.

among users, the floating point performance of a statistical software package may be a necessary condition, but other considerations such as functionality, I/O performance, graphics, and documentation often prevail.

6.2 Numerical Considerations

In this section we review some of the numerical difficulties that we encountered while implementing various RBF estimation programs. Although these difficulties are certainly present for the CM-2 implementation discussed above, they stem from the complexity and large size of the networks being estimated, and thus we discuss them in a separate section to highlight the fact that they will be encountered in the estimation of complex, large RBF networks implemented on *any* hardware platform. In addition, we feel that understanding these numerical difficulties is important not only to get good solutions; but also that exploring the difficulties in a bottom-up style can often lead to new insights about the estimation process itself.

6.2.1 Matrix Inversion

In Chapter 2 we noted two places where a required matrix inversion operation was possibly ill-conditioned: in the direct solution of the output coefficients (see Equation 2.8), and at the heart of the Levenberg-Marquardt method when inverting the Hessian matrix (see Equation 2.5). Although blind adoption of a stable matrix inversion routine such as Singular Value Decomposition (SVD) will for the most part remedy the numerical difficulties here, it won't necessarily yield the most meaningful solutions. It is worthwhile, therefore, to stop for a moment and consider the causes of this numerical instability and see if in some cases there are better ways around the problem.

In general the difficulties with matrix inversion come about from nearly collinear columns in the matrix. In the specific case of RBF networks, this collinearity can come from about from the linear terms in the formulation, which is exactly analogous to the well understood problems with collinearity in linear regression. The typical strategy for dealing with the problem in linear regression is to choose a better set of regressors via some technique such as stepwise regression (see Cryer and Miller (1991)). The other source of collinearity we might encounter is from the nonlinear basis functions, and by implication our choices for free parameters of those functions: the centers $\vec{z}$, the basis function scale parameters σ, and the input weights W. One way this can happen is if the rotation and stretching of the data caused by W does not produce a good distribution of data points and basis function centers, although typically this is not a concern if W is chosen as some form of the inverse covariance matrix of the data. A second way that we can get collinear basis functions is if the sampling of their centers is either too dense or much too sparse relative to the scale parameters, which causes either nearby or all basis function outputs to be virtually identical. This can be rectified by individually changing the scale of the basis functions, shifting their centers, or eliminating some of them. Note that automatic methods can be devised to implement these corrections; the techniques outlined in Chapter 2, for example, are designed to avoid this kind of collinearity. However, just as in the case of linear regression collinearity, manual inspection by someone knowledgeable in the problem domain is preferred whenever possible to take full advantage of any domain or case specific considerations.

6.2.2 Precision

Another implementation issue in estimating RBF networks concerns numerical precision. Because of their large size, inhomogeneity, and possibly local structure, estimation of RBF networks tends to deal with many numbers of vastly different orders of magnitude. On one hand, local structure can imply the insignificance of many of these numbers, and may enable us to greatly reduce the quantity of floating point operations required to achieve answers virtually identical to the "true" full precision answers. On the other hand, inhomogeneity and large size of the networks can generate function values and derivatives that are difficult to combine accurately without using high precision. Unlike the collinearity questions, however, our concerns about precision are relatively easy to analyze and handle and only require that we take the time to do so.

The basic idea behind our observation about locality is that if we are willing to use kernel-type basis functions that have a maximum at zero and decrease fairly rapidly away from zero, then the network's output at each point will largely be a function only of the nearby basis functions, and thus we do not need to waste computer cycles calculating the other basis functions. Put another way, a reasonable approximation to Equation (1.1) is if we drop the terms in the summation where $h_i(\|\vec{x} - \vec{z}_i\|) < \epsilon$ for a particular input $\vec{x}$ and some small constant ϵ which we can choose relative to the overall maximum and minimum of h_i (e.g. if h_i is a gaussian with outputs ranging from 0 to 1, perhaps we don't need values less than $\epsilon = 10^{-4}$). Note that we may be tempted to determine significance versus the particular minimum and maximum outputs for a given input, but this requires computing all of the outputs and thus defeats our goal of saving computation.

How much computation would this type of approximation save? If we assume that multiplication, addition, and subtraction take the same amount of time, it takes H of these generic operations to compute each basis function, and the polynomial term $p(\vec{x})$ is linear, then computing the full RBF equation for a single input takes

$$2k \cdot (2d(d+1) + H) + 2d$$

operations for k basis functions and d-dimensional inputs. If on a particular application it makes sense to choose ϵ such that we only need to compute $a \cdot k$ of the basis functions, then, we would save

$$2k(1-a) \cdot (2d(d+1) + H)$$

operations. Incidentally, we note that the decision to use a diagonal or constant d x d weight matrix is important from a computational complexity point of view only if the basis functions are simple relative to the input dimensionality (i.e. $H \not\gg 2d(d+1)$).

On the negative side, RBF networks can place unusually stringent requirements on numerical precision, especially when using increasing basis functions which carefully balance one another, or when we are early in the estimation process and our parameter estimates are far from optimal. For contrast, consider the multilayer perceptron representation in Equation (1.3); if the free parameters c are chosen to be small zero mean random numbers with bias terms set to the mean of the inputs, then the sigmoidal nonlinear units will operate in their linear range and generate well conditioned, low variance outputs right from the very first few iterations of a typical estimation process. If we chose random values for RBF network free parameters, on the other hand, intermediate calculations in the network can take on values of arbitrary and widely varying magnitudes, which increase the need for stable algorithms and high precision arithmetic. The large size of some of the systems considered only worsens this situation by increasing the number and nesting of finite precision calculations that are made. This in fact is a major motivation for the initial parameter estimate heuristics of Chapter 2. Choosing an input weight matrix W that adequately handles disproportionate scales of different input dimensions is an important and common example of this. However, the general importance of this type of numerical problem is lessened somewhat by the fact that typically these conditions will not hold near the minima of the network cost function, and thus may not greatly affect the final solutions obtained except for a possible bias towards "stable" parameter sets.

Chapter 7

Conclusion

In Chapter 1 we began with the rather ambitious goal of trying to predict the stock market using a data driven, nonlinear and multivariate statistical modeling approach. Did any of the results obtained in this thesis constitute "success"? Strictly speaking, we would have to reply "no"; after all, the commonly understood definition of predicting the stock market implies developing a system which can make money more reliably than other investment strategies, and unfortunately we have not done that. However often the ideas and techniques developed in the pursuit of a lofty goal are useful in their own right, and we hope that is the case here. In this concluding chapter we discuss some of the contributions and limitations of this thesis from a few broader perspectives, while pointing out interesting areas for further investigation.

7.1 Financial Markets

Although the general mechanisms underlying the evolution of stock market prices elude us, we believe there is still room for cautious optimism about the use of exploratory statistical modeling in the financial markets. At the very least, the results of Chapter 5 indicate that statistical models may provide a systematic way to leverage an investor's natural advantages in terms of access to data, market presence, or market expertise. It seems that the best way to do this is to think locally about what narrow markets, trading environments, and data that one has a competitive advantage with, rather than globally about broad forces and highly efficient financial instruments. Perhaps successful systems will not be predictive in nature (e.g. the options models from Chapter 3), and perhaps they won't output prices directly (e.g. determining the time value of data in Chapter 5) , but approached in this way, it may be possible to develop models that some of the time tell us something useful about the markets.

This line of thinking points to a few ideas for future work. First is the general goal of finding *hedging* relationships, which model the spread between specified financial instruments by explicitly going long (i.e. buying) some of them, and going short (i.e. selling) the others. Because the modeler is looking for the difference of two (or more) instruments, it may be possible to avoid the need to capture some aspects of the price dynamics, and instead focus on some more localized mechanisms. A second idea is to think more explicitly about "regime variables" which might split the problem domain into simpler parts, rather than depending upon the flexibility of nonlinear maps to segment the domain. These regime variables could be based on the kinds of information that traders often look at, or they might be chosen strictly on statistical grounds.

Our conclusion that learning network technology is not enough by itself to "break open the stock market problem" should not be surprising to most. The scientific method suggests that we use experimental data to confirm or refute preconceived hypotheses and to suggest new ones, and it is clear why that time honored methodology is appropriate in this case; combinatorics, statistical inference considerations, and limited data set sizes are all against us if we attempt to derive everything from the data. Thus we prefer to start from a theoretical model and use the data to look for important discrepancies from that theory, where perhaps the steps we can take away from theory are slightly larger with the more powerful nonlinear tools. For instance in our option pricing application, first matching the Black-Scholes theoretical results provides a sound base for future work to expand from exploring more advanced alternatives, such as explicitly including volatility measures as inputs.

7.2 Statistical Modeling

The central question in exploratory statistical modeling is the question of confidence; if we don't know how much to believe our models and predictions, we may be better off not using them or developing them in the first place. Our preoccupation with the topic of confidence in Chapter 4 underlines our belief that the choice of methodologies and technologies used in these problems should revolve around considerations of confidence and data mining. Although such considerations will for the near future necessitate a good dose of expert judgment, more rigorously derived statistical estimators and diagnostic tests would help.

Our pointwise variance estimator in Chapter 4 was a step in this direction, but it was not as general as one would like. In particular, that result was asymptotic and relied upon large numbers of nonlinear units, a requirement that contradicts our tendency to prefer simpler models. A more useful estimator may be found in extensions of the distributional results in Appendix A. Those results may also form the basis for better diagnostic tests of RBF networks, and thus deserve further study.

Another idea derived primarily from statistical concerns is that of combining different models and pieces of models. At one end of this spectrum of ideas is the simple concept of combining model forecasts to reduce variance. This idea is a long standing one in the statistics community (for a review see Clemen (1989)), and has recently been rediscovered by a number of researchers in the connectionist and machine learning community (for instance see Perrone (1993) or Buntine and Weigend (1991)), where they have been found to be particularly useful in part because of the variance arising from random initial conditions and relatively ill-conditioned estimation procedures. On the other end of the spectrum the combination idea can be thought of as motivation for creating complex models from simpler pieces that each in some (perhaps minor) way produce interesting transformations or "add value" to the inputs, where in this case the combining functions subsume a substantial portion of the "work" (for instance see Jordan and Jacobs (1993)). In the case of RBF networks, our algorithm in Chapter 2 for sometimes including multiple input weight matrices to handle multimodal data falls into the later category, and perhaps could be augmented with other systematic techniques for piecing together better performing networks.

However, a fundamental conclusion of this thesis is that it does not pay to focus solely on *what* techniques are used - it is also vitally important to carefully understand *how* to use them, and what data to use them on. Partly this is due to the fact that the tools are not robust and must be used carefully, and partly this reflects the fact that the best models are typically obtained by incorporating as much domain specific expertise as possible. As the interaction between this expertise and the statistical methodology becomes well defined it can be captured in an algorithm (such as the combination ideas above), but until then it must be embodied in the expert intensive style of manually checking and plotting data, assumptions and diagnostics. In particular, good financial time series modeling methodology relies heavily upon checking the source, composition, and quality of the data used.

7.3 High Performance Computing

Is has often been argued that the advent of fast, ubiquitous computing power has enabled the increasingly popular method of using exploratory modeling as a means to quantitatively understand many problem domains, including the financial markets. Indeed in Chapter 6 we have argued that the need for large databases, sample reuse techniques, and limited model identification searches, along with the high dimensionality and complexity of the models presently considered all necessitate the large storage, networks, and fast execution rates of today's computer systems. Although the use of megabytes of data and megaflops of number crunching does not by itself guarantee that better models and more useful insights will be obtained, careful attention to statistical methodology should increase the leverage that this computing power provides to the exploratory process.

From a computer science perspective, we are interested in developing systems that learn the most useful information possible from a problem domain for a given cost. However, this high level tradeoff between amount and quality of learning versus cost is not well understood currently. For instance, if we are to use parallel computers to minimize complex cost functions (e.g. for estimation), what are the best algorithms for coordinating that search? "Single thread" search algorithms such as Levenberg-Marquardt may be the most efficient in terms of least amount of "wasted" computation, but "multiple thread" algorithms such as parallel stochastic search may be best in terms of solution quality per unit of processing time. A better understanding of these tradeoffs will allow us to determine the relative importance we give to communication bandwidth, memory, and computation, and thus engineer the most cost effective learning systems.

Appendix A

RBF Distributions

In considering the statistical properties of RBF networks it is useful to know how the networks transform the probability distributions of the inputs. Clearly this transformation depends on the type of basis function used and the dimensionality of the problem considered. However, we note that a fundamental attribute of this transformation comes about from the use of a distance metric.

To see this, assume we have a random variable $x \in \Re^d$ with uniform probability distribution on the d-dimensional sphere of radius R, i.e.

$$P(\mathbf{x}) = C\theta(R - \|\mathbf{x}\|) \tag{A.1}$$

where θ is the Heaviside (step) function and C is a normalization constant. Let us compute C, imposing:

$$1 = \int_{R^d} d\mathbf{x}\ P(\mathbf{x}) = C \int_{R^d} d\mathbf{x}\ \theta(R - \|\mathbf{x}\|) = C \int_0^R dr\ r^{d-1} \int d\Omega_d = C\frac{R^d}{d}\Gamma_d$$

where $d\Omega_d$ is the element of solid angle in d dimensions, and Γ_d is its integral over the entire sphere[1]. Therefore the uniform probability distribution on the sphere is:

$$P(\mathbf{x}) = \frac{d}{R^d \Gamma_d} \theta(R - \|\mathbf{x}\|) \ . \tag{A.2}$$

Denoting $r \equiv \|\mathbf{x}\|$, we can now compute the probability distribution of the norm of the vector $\mathbf{x}$ as:

$$P(r) = \int_{R^d} d\mathbf{x} \ P(\mathbf{x})\delta(\|\mathbf{x}\| - r) = \frac{d}{R^d \Gamma_d} \int_{R^d} d\mathbf{x} \ \theta(R - \|\mathbf{x}\|)\delta(\|\mathbf{x}\| - r) =$$

$$= \frac{d}{R^d \Gamma_d} \int_0^\infty dh \ h^{d-1} \theta(R - h)\delta(h - r) \int d\Omega_d = \frac{d}{R^d} r^{d-1} \theta(R - r)$$

This result is simply a statement of the fact that as the dimensionality of a problem increases, the bulk of the volume of a hypersphere is concentrated more and more in its outermost "layers". In fact, the probability that a point $\mathbf{x}$ has norm r between R_1 and R_2 is:

$$P(R_1 \leq \|\mathbf{x}\| \leq R_2) = \int_{R_1 \leq \|\mathbf{x}\| \leq R_2} d\mathbf{x} \ P(\mathbf{x}) = \frac{R_2^d - R_1^d}{R^d} \ . \tag{A.3}$$

Taking $R_2 = R$ we have that the probability that the distance of a point from the surface is greater than R_1 is:

$$P(R_1 \leq \|\mathbf{x}\| \leq R) = 1 - (\frac{R_1}{R})^d \ .$$

It is also instructive to compute the average value of r:

$$< r > = \int_0^\infty dr \ rP(r) = \frac{d}{d+1} R$$

Notice how the average value of r tends to the radius of the sphere R as the dimension increases.

[1]The function $\Gamma_d = \int d\Omega_d$ can be computed in terms of the Euler Γ-function.

The implications of this phenomena on the basis function outputs of an RBF network can be seen in the following related result. Let $h(\|\mathbf{x}\|)$ be the output of a Radial Basis Function unit. We want to derive the probability distribution P of the random variable $y = h(\|\mathbf{x}\|)$ given the probability distribution of $\mathbf{x}$, which we assume to be the radial function $P(\|\mathbf{x}\|)$. We have:

$$P(y) = \int_{R^d} d\mathbf{x}\ P(\|\mathbf{x}\|)\delta(y - h(\|\mathbf{x}\|) = \Gamma_d \int_0^\infty dr\ r^{d-1} P(r)\delta(y - h(r))\ .$$

We also assume that the basis function $h(r)$ is invertible on the real positive axis. Performing the change of variable $s = h(r)$ we obtain:

$$P(y) = \Gamma_d \int_0^\infty \frac{ds}{h'(h^{-1}(s))}[h^{-1}(s)]^{d-1} P(h^{-1}(s))\delta(y - s)$$

and consequently:

$$P(y) = \Gamma_d \frac{1}{h'(h^{-1}(y))}[h^{-1}(y)]^{d-1} P(h^{-1}(y)) \tag{A.4}$$

If we now set $g(y) = h^{-1}(y)$ this expression can be rewritten as

$$P(y) = \Gamma_d g'(y)[g(y)]^{d-1} P(g(y))\ . \tag{A.5}$$

where we used the rule of differentiation of the inverse function:

$$g'(y) = \frac{1}{h'(h^{-1}(y))}\ .$$

If we are given the probability distribution P, and we want to find out what kind of basis function h transform P in P, the previous equation can be seen as a differential equation for g.

Although the differential equation (A.5) is very difficult to solve in general cases, in the case in which the probability distribution P is uniform over a sphere the function

h can be expressed in terms of the inverse of the density of the probability distribution P. In fact, taking the sphere of radius 1 for simplicity, it can be rewritten as :

$$P(y) = dg'(y)[g(y)]^{d-1}\theta(1 - g(y)) \ . \tag{A.6}$$

We now make the following choice for g:

$$g(y) = \left[\int_{-\infty}^{y} ds \ P(s)\right]^{\frac{1}{d}} \ . \tag{A.7}$$

The first derivative of g is therefore:

$$g'(y) = \frac{1}{d}\left[\int_{-\infty}^{y} ds \ P(s)\right]^{\frac{1}{d}-1} P(y)$$

and substituting it in equation (A.6) we obtain

$$P(y) = \left[\int_{-\infty}^{y} ds \ P(s)\right]^{\frac{1}{d}-1} P(y) \left[\int_{-\infty}^{y} ds \ P(s)\right]^{1-\frac{1}{d}} \theta(1 - g(y)) = P(y)\theta(1 - g(y)) \ .$$

Since by definition $g(y) < 1$ for all the values of y, the step function has always value equal to one, and the previous equation is an identity.

Consequently we conclude that, if we want the output of a Radial Basis Functions unit to be distributed according to the distribution P whenever the input variables are distributed uniformly inside a sphere of radius 1, the basis function h should be such that:

$$h^{-1}(y) = \left[\int_{-\infty}^{y} ds \ P(s)\right]^{\frac{1}{d}} \ . \tag{A.8}$$

If, for example, we want the output of a radial basis function unit to be have a Gaussian distribution, the basis function should be:

$$h(\|\mathbf{x}\|) = erf^{-1}(\|\mathbf{x}\|^{d})$$

where $erf(y)$ is the error function, that is the density of the gaussian distribution.

Bibliography

[1] A.R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. Technical Report 58, Department of Statistics, University of Illinois at Urbana-Champaign, Champaign, IL, March 1991.

[2] A.R. Barron and R.L. Barron. Statistical learning networks: a unifying view. In *20th Symposium on the Interface: Computing Science and Statistics*, pages 192–203, 1988.

[3] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81, May-June 1973.

[4] Zvi Bodie, Alex Kane, and Alan J. Marcus. *Investments*. Richard D. Irwin, Inc., 1989.

[5] G.E.P. Box and G.M. Jenkins. *Time Series Analysis, Forecasting, and Control*. Holden-Day, San Fransisco, 2nd edition, 1976.

[6] Ronald J. Brachman and Hector J. Levesque, editors. *Readings in Knowledge Representaiton*. Morgan Kaufmann Publishers, Inc., 1985.

[7] D.S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[8] W.L. Buntine and A.S. Weigend. Bayesian back-propagation. *Complex Systems*, 5:603–643, 1991.

[9] B. Caprile and F. Girosi. A nondeterministic minimization algorithm. Artificial Intelligence Memo 1254, Massachusetts Institute of Technology, 1990.

[10] M. Casdagli. Nonlinear prediction of chaotic time series. *Physica D*, 35:335–356, 1989.

[11] C. Chatfield. *The Analysis of Time Series*. Chapman and Hall, 1989.

[12] H. Chen. Estimation of a projection-pursuit type regression model. *Ann. Statistics*, 19:142–157, 1991.

[13] R.T. Clemen. Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting*, 5:559–583, 1989.

[14] James P. Crutchfield and Bruce S. McNamara. Equations of motion from a data series. *Complex Systems*, 1:417–452, 1987.

[15] Jonathan Cryer and Robert Miller. *Statistics for Business: Data Analysis and Modeling*. PWS-KENT Publishing, Boston, 1991.

[16] G. Cybenko. Approximation by superpositions of a sigmoidal function. Technical Report 856, University of Illinois, Dept. of Electrical and Computer Engineering, 1988.

[17] C. de Groot and D. Würtz. Analysis of univariate time series with connectionist nets: A case study of two classical examples. In F. Murtagh, editor, *Workshop on Neural Networks for Statistical and Economic Data*, pages 95–112. Munotec Systems, 1991.

[18] P. Diaconis and M. Shahshahani. On nonlinear functins of linear combinations. *SIAM J. Sci. Stat. Comput.*, 5(1):175–191, 1984.

[19] D.L. Donoho and I. Johnstone. Projection-based approximation and a duality with kernel methods. *Ann. Stat.*, 17:58–106, 1989.

[20] Nira Dyn and David Levin. Iterative solution of systems originating from integral equations and surface interpolation. *Siam J. Numer. Anal.*, 20(2), April 1983.

[21] Bradley Efron and Gail Gong. A leisurely look at the bootstrap, the jackknife, and cross-validation. *The American Statistican*, 37(1):36 – 48, February 1983.

[22] V.A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theor. Probab. Appl.*, 14:153–158, 1969.

[23] Randall L. Eubank. *Spline Smoothing and Nonparametric Regression.* Marcel Dekker, New York, 1988.

[24] J.H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association, Theory and Methods Section*, 76(376), December 1981.

[25] Th. Gasser and H.G. Müller. Kernel estimation of regression functions. In Th. Gasser and M. Rosenblatt, editors, *Smoothing Techniques for Curve Estimation.* Heidelberg: Springer, 1979.

[26] F. Girosi and G. Anzellotti. Rates of convergence of approximation by translates. A.I. Memo 1288, Massachusetts Institute of Technology Artificial Intelligence Laboratory, 1992.

[27] F. Girosi, M. Jones, and T. Poggio. Priors, stabilizers and basis functions: from regularization to radial, tensor and additive splines. Artificial Intelligence Memo 1430, Massachusetts Institute of Technology, 1993.

[28] F. Girosi and T. Poggio. Networks and the best approximation property. *Biological Cybernetics*, 63:169–176, 1990.

[29] C.W.J. Granger and P. Newbold. Spurious regressions in econometrics. *Journal of Econometrics*, 2:111–120, 1974.

[30] C.W.J. Granger and T. Teräsvirta. Modelling dynamic nonlinear relationships. Unpublished manuscript, December 1991.

[31] Chong Gu and Grace Wahba. Smoothing spline anova with component-wise bayesian confidence intervals. *To appear in J. Computational and Graphical Statistics*, 1992.

[32] Wolfgang Härdle. *Applied Nonparametric Regression*. Cambridge University Press, Cambridge, 1990.

[33] B. Hassibi and D.G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Proceedings of Neural Information Processing Systems 92*, 1992.

[34] W. Daniel Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*, 42:228–234, 1990.

[35] K. Hornik. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[36] P.J. Huber. Projection pursuit. *Ann. Stat.*, 13(2):435–525, 1985.

[37] Farmer J.D. and Sidorowich J.J. Exploiting chaos to predict the future and reduce noise. In *Evolution, Learning and Cognition*. World Scientific, Singapore, 1989.

[38] M.C. Jensen. Some anomalous evidence regarding market efficiency. *Journal of Financial Economics*, 6:95–101, 1978.

[39] Richard Johnson and Dean Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, New Jersey, 1988.

[40] L.K. Jones. On a conjecture of Huber concerning the convergence of projection pursuit regression. *Ann. Stat.*, 15(2):880–882, 1987.

[41] R.D. Jones, Y.C. Lee, C.W. Barnes, G.W. Flake, K. Lee, P.S. Lewis, and S. Qian. Function approximation and time series prediction with neural networks. Technical Report LA-UR-90-21, Los Alamos National Laboratory, 1990.

[42] M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the em algorithm. Technical Report 9301, Massachusetts Institute of Technology Department of Brain and Cognitive Sciences, April 1993.

[43] V. Kadirkamanathan, M. Niranjan, and F. Fallside. Sequential adaptation of radial basis function neural networks and its application to time-series prediction. In R.P. Lippmann, J. Moody, and D.S. Touretzky, editors, *Advances in Neural Information Processing (NIPS-90)*, San Mateo, CA, April 1991. Morgan Kaufmann.

[44] Takashi Kimoto, Kazuo Asakawa, Morio Yoda, and Masakazu Takeoka. Stock market prediction system with modular neural networks. In *Proceedings of IJCNN-90*, San Diego, 1990.

[45] Adam Krzyżak. Learning, radial basis function nets and nonparametric regression. preprint.

[46] Adam Krzyżak. The rates of convergence of kernel regression estimates and classification rules. *IEEE Transactions on Information Theory*, IT-32(5), September 1986.

[47] Alan Lapedes and Robert Farber. Nonlinear signal processing using neural networks: Prediction and system modeling. Technical Report LA-UR-87-2662, Los Alamos National Laboratory, July 1987.

[48] Blake LeBaron. Some relations between volatility and serial correlations in stock market returns. Technical Report 9002, Social Systems Research Institute, University of Wisconsin, Madison, February 1990.

[49] J.A. Leonard, M.A. Kramer, and L.H. Ungar. A neural network architecture that computes its own reliability. *Computers and Chemical Engineering*, Submitted to, August 1991.

[50] John Lintner. The valuation of risk assets and the selection of risky investments in stock portfolios and capital budgets. *Review of Economics and Statistics*, February 1965.

[51] D. Lovell, P. Bartlett, and T. Downs. Error and variance bounds on sigmoidal neurons with weight and input errors. Neuroprose preprint, University of Queensland, 1992.

[52] M. C. Lovell. Data mining. *Review of Economics and Statistics*, 65(1):1–12, February 1983.

[53] Peter S. Lynch. *One up on Wall Street: how to use what you already know to make money in the market.* Simon and Schuster, New York, 1989.

[54] Harry M. Markowitz. *Portfolio Selection: Efficient Diversification of Investments.* John Wiley & Sons, New York, 1959.

[55] D. W. Marquardt. *J. Soc. Ind. Appl. Math*, 11:431–441, 1963.

[56] M. Maruyama, F. Girosi, and T. Poggio. A connection between GRBF and MLP. Artificial Intelligence Memo 1291, Massachusetts Institute of Technology, 1991.

[57] R.C. Merton. Theory of rational option pricing. *Bell Journal of Economics and Management Science*, 4:141–183, Spring 1973.

[58] Robert C. Merton. *Continuous-Time Finance.* Blackwell Publishers, Cambridge, 1990.

[59] Charles A. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22, 1986.

[60] J. Moody and C. Darken. Fast learning in networks of locally tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

[61] Jan Mossin. Equilibrium in a capital asset market. *Econometrica*, October 1966.

[62] Radford M. Neal. Bayesian training of backpropagation networks by the hybrid monte carlo method. Technical Report CRG-TR-92-1, University of Toronto Department of Computer Science, April 1992. email radford@cs.toronto.edu.

[63] K. Ng and R. Lippman. A comparative study of the practical characteristics of neural network and conventional pattern classifiers. In R. Lippman, J. Moody, and D. Touretsky, editors, *Advances in Neural Information Processing Systems 3*. Morgan-Kaufman, 1991.

[64] P. Niyogi and F. Girosi. On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions. Artificial Intelligence Memo (in preparation), Massachusetts Institute of Technology, 1993.

[65] D.B. Parker. Learning logic. Technical Report 47, Center for Computational Research in Economics and Management Science, MIT, April 1985.

[66] Michael P. Perrone. *Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Measure Optimization.* PhD thesis, Brown University, Institute for Brain and Neural Systems; Dr. Leon N Cooper, Thesis Supervisor, May 1993.

[67] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of IEEE*, 78(9):1481–1497, 1990.

[68] M. J. D. Powell. Radial basis functions for multivariable interpolation: A review. In J. C. Mason and M. G. Cox, editors, *Algorithms for Approximation*, pages 143–167. Clarendon Press, Oxford, 1987.

[69] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, Cambridge, 1988.

[70] M.B. Priestley. *Non-linear and Non-stationary Time Series*. Academic Press, London, 1988.

[71] A.N. Refenes. Constructive learning and its application to currency exchange rate prediction. In E. Turban and R. Trippi, editors, *Neural Network Applications in Investment and Finance Services*, chapter 27. Probus Publishing, 1992.

[72] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representation by error propagation. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, chapter 8. MIT Press, Cambridge, MA, 1986.

[73] William Sharpe. Capital asset prices: A theory of market equilibrium. *Journal of Finance*, September 1964.

[74] Alexander Singer. Implementations of Artificial Neural Networks on the Connection Machine. Technical Report RL90-2, Thinking Machines Corporation, 245 First Street, Cambridge, MA 02142, January 1990.

[75] H. Tong. *Non-linear time series: a dynamical system approach*. Clarendon Press, Oxford, 1990.

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

[77] Ruey S. Tsay. Nonlinear time series analysis: Diagnostics and modelling. Department of statistics working paper, Carnegie Mellon University, December 1988.

[78] Joachim Utans and John Moody. Selecting neural network architecture via the prediction risk: Application to corporate bond rating prediction. In *First International Conference on Artificial Intelligence Applications on Wall Street*, Los Alamitos, CA, 1991. IEEE Computer Society Press.

[79] Grace Wahba. *Spline Models for Observational Data*, volume 59 of *Regional Conference Series in Applied Mathematics*. SIAM Press, Philadelphia, 1990.

[80] William W. S. Wei. *Time Series Analysis*. Addison-Wesley, 1990.

[81] Andreas S. Weigend, David E. Rumelhart, and Bernardo A. Huberman. Generalization by weight-elimination with application to forecasting. In R.P. Lippmann, J. Moody, and D.S. Touretzky, editors, *Advances in Neural Information Processing (NIPS-90)*, San Mateo, CA, April 1991. Morgan Kaufmann.

[82] Halbert White. Economic prediction using neural networks: The case of ibm daily stock returns. Department of economics, University of California at San Diego, 1988.

[83] Patrick H. Winston. *Artificial Intelligence*. Addison-Wesley, Reading, Mass, 2nd edition, 1984.

[84] Francis Wong and Pan Yong Tan. Neural network and genetic algorithm for economic forecasting. Technical report, National University of Singapore Institute of Systems Science, 1992. Available from thgoh@iss.nus.sg.

[85] Xiru Zhang and James M. Hutchinson. Simple algorithms on fast machines: Practical issues in nonlinear time series prediction. In A. Weigend and N. Gershenfeld, editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*, pages 219–241, Reading, MA, 1993. Addison-Wesley.