MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY
and
CENTER FOR BIOLOGICAL AND COMPUTATIONAL LEARNING

A.I. Memo No. 1471 April 1994
C.B.C.L. Paper No. 92

A Nonparametric Approach to Pricing and Hedging Derivative Securities Via Learning Networks

James M. Hutchinson, Andrew Lo and Tomaso Poggio

Abstract

We propose a nonparametric method for estimating the pricing formula of a derivative asset using learning networks. 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 comparison, we estimate models using four popular methods: ordinary least squares, radial basis function networks, multilayer perceptron networks, and projection pursuit. To illustrate the practical relevance of our network pricing approach, we apply it to the pricing and delta-hedging of S&P 500 futures options from 1987 to 1991.

This report describes research done within the Artificial Intelligence Laboratory and the Sloan School of Management's Research Program in Computational Finance at the Massachusetts Institute of Technology. This research is sponsored by grants from the Office of Naval Research under contracts N00014-92-J-1879 (AASERT) and N00014-93-1-0385. Support for the A.I. Laboratory's artificial intelligence research is provided by ARPA contract N00014-91-J-4038. Additional support was provided by the National Science Foundation under contract ASC-9217041, the Research Program in Computational Finance, Siemens AG, ATR Audio and Visual Perception Research Laboratories. J. Hutchinson is with PHZ Partners (One Cambridge Center, Cambridge, MA 02142). A portion of this research was conducted during A. Lo's tenure as an Alfred P. Sloan Research Fellow. T. Poggio is supported by the Uncas and Helen Whitaker Chair at MIT's Whitaker College.

1 Introduction

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 obtained 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 a *parametric* pricing method, is closely tied to the ability to capture the dynamics of the underlying asset's price process.

In this paper, 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. We take as inputs the primary economic variables that influence the derivative's price, e.g., current fundamental asset price, strike price, time-to-maturity, etc., and define the derivative price to be the output into which the learning network maps the inputs. When properly trained, the network "becomes" the derivative pricing formula which may be used in the same way that formulas obtained from the parametric pricing method are used: for pricing, delta-hedging, simulation exercises, etc.

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 are robust to the specification errors that plague parametric models. Second, they are adaptive, and respond to structural changes in the data-generating processes 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 these advantages do not come without some cost—the nonparametric pricing method is highly data-intensive, requiring large quantities of historical prices to obtain 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 rarely enough that there may still be great practical value in constructing derivative pricing formulas by learning networks.

In Section 2, we provide a brief review of learning networks and related statistical methods. To illustrate the promise of learning networks in derivative pricing applications, in Section 3 we report the results of several Monte Carlo simulation experiments in which radial basis function (RBF) networks "discover" the Black-Scholes formula when trained on Black-Scholes call option prices. Moreover, the RBF network pricing formula performs as well as the Black-Scholes formula in delta-hedging a hypothetical option, and in some cases performs even better [because of the discreteness-error in the Black-Scholes case arising from delta-hedging daily instead of continuously]. To gauge the practical relevance of our nonparametric pricing method, in Section 4 we apply the RBF pricing model to daily call option prices on S&P 500 futures from 1987 to 1991 and compare its pricing and delta-hedging performance to the naive Black-Scholes model. We find that in many cases, the network pricing formula outperforms the Black-Scholes model. We suggest several directions for future research and conclude in Section 5.

2 Learning Networks: A Brief Review

Over the past 15 years, a number of techniques have been developed for modeling nonlinear statistical relations nonparametrically. In particular, projection pursuit regression, multilayer perceptrons [often called "backpropagation networks"[2]], and radial basis functions are three popular examples of such techniques. Although originally developed in different contexts for seemingly different purposes, these techniques may all be viewed as nonparametric methods for performing nonlinear regressions. Following Barron and Barron (1988) we call this general class of methods *learning networks* to emphasize this unifying view and acknowledge their common his-

[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.

[2] 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.

tory. In the following sections, we shall provide a brief review of their specification and properties. Readers already familiar with these techniques may wish to proceed immediately to the Monte Carlo simulation experiments of Section 3.

2.1 Standard Formulations

In this section we describe the standard formulations of the learning networks to be used in this paper. For expositional simplicity, we shall focus our attention on the problem of mapping multiple input variables into a univariate output variable, much like regression analysis, although the multivariate-output case is a straightforward extension.

Given the well-known trade-offs between degrees of freedom and approximation error in general statistical inference, we shall also consider the number of parameters implied by each model so that we can make comparisons between them on a roughly equal footing. Note, however, that the number of free parameters is a crude measure of the complexity of nonlinear models, and more refined measures may be available, e.g., the nonlinear generalizations of the influence matrix in Wahba (1990).

A common way to visualize the structure of these networks is to draw them as a graph showing the connections between inputs, nonlinear "hidden" units, and outputs [see Figure 1].

2.1.1 Radial Basis Functions

Radial Basis Functions (RBFs) were first used to solve the *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 several 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 the classical regularization problem in which some unknown function $y = f(\vec{x})$ is to be approximated given a sparse dataset $(\vec{x}_t, y_t)$ and some smoothness constraints. In terms of our multiple-regression analogy, the d-dimensional vector $\vec{x}_t$ may be considered the "independent" or "explanatory" variables, y_t the "dependent" variable, and $f(\cdot)$ the [possibly] nonlinear function that is the conditional expectation of y_t given $\vec{x}_t$, hence:

$$y_t \;=\; f(\vec{x}_t) \;+\; \epsilon_t \qquad , \qquad \mathrm{E}[\epsilon_t|\vec{x}_t] \;=\; 0\;. \quad (1)$$

The regularization [or "nonparametric estimation"] problem may then be viewed as the minimization of the following objective functional:

$$H(\hat{f}) \;\equiv\; \sum_{t=1}^{T} \left(||\hat{y}_t - \hat{f}(\vec{x}_t)||^2 + \lambda ||Pf(\vec{x}_t)||^2 \right) \quad (2)$$

where $||\cdot||$ is some vector norm and P is a differential operator. The first term of the sum in (2) is simply the distance between the approximation $\hat{f}(\vec{x}_t)$ and the observation y_t, the second term is a penalty function that is a decreasing function of the smoothness of $\hat{f}(\cdot)$, and λ controls the trade-off between smoothness and fit.

In its most general form, and under certain conditions [see, for example, Poggio and Girosi (1990)], the solution to (2) is given by the following expression:

$$\hat{f}(\vec{x}) = \sum_{i=1}^{k} c_i h_i(||\vec{x} - \vec{z}_i||) + p(\vec{x}) \quad (3)$$

where $\{\vec{z}_i\}$ are d-dimensional vector prototypes or "centers", $\{c_i\}$ are scalar coefficients, $\{h_i\}$ are scalar functions, $p(\cdot)$ is a polynomial, and k is typically much less than the number of observations T in the sample. Such approximants have been termed "hyperbasis functions" by Poggio and Girosi (1990) and are closely related to splines, smoothers such as kernel estimators, and other nonparametric estimators.[3]

For our current purposes, we shall take the vector norm to be a weighted Euclidean norm defined by a $(d \times d)$ weighting matrix W, and the polynomial term shall be taken to be just the linear and constant terms, yielding the following specification for $\hat{f}(\cdot)$:

$$\begin{aligned} \hat{f}(\vec{x}) \quad &= \quad \sum_{i=1}^{k} c_i h_i \Big(\; (\vec{x} - \vec{z}_i)' W' W (\vec{x} - \vec{z}_i) \; \Big) \\ &\quad + \alpha_0 \;+\; \vec{\alpha}_1' \vec{x} \end{aligned} \quad (4)$$

where α_0 and $\vec{\alpha}_1$ are the coefficients of the polynomial $p(\cdot)$. Micchelli (1986) shows that a large class of basis functions $h_i(\cdot)$ are appropriate, but the most common choices for basis functions $h(x)$ are Gaussians e^{-x/σ^2} and multiquadrics $\sqrt{x + \sigma^2}$.

That networks of this type can generate any real-valued output, but in applications where we have some *a priori* knowledge of the range of the desired outputs, it is computationally more efficient to apply some nonlinear transfer function to the outputs to reflect that knowledge. This will be the case in our application to derivative pricing models, in which some of the RBF networks will be augmented with an "output sigmoid", which maps the range $(-\infty, \infty)$ into the fixed range $(0, 1)$. In particular, the augmented network will be of the form $g(\hat{f}(\vec{x}))$ where $g(u) = 1/(1 + e^{-u})$.

For a given set of inputs $\{\vec{x}_t\}$ and outputs $\{y_t\}$, RBF approximation amounts to estimating the parameters of the RBF network: the $d(d+1)/2$ unique entries of the matrix $W'W$, the dk elements of the centers $\{\vec{z}_i\}$, and the $d + k + 1$ coefficients α_0, $\vec{\alpha_1}$, and $\{c_i\}$. Thus the total number of parameters that must be estimated for d-dimensional inputs and k centers is dk+$(d^2/2)$+$(3d/2)$+k+1.

[3]To economize on terminology, in this paper we use the term "radial basis functions" to encompass both the interpolation techniques used by Powell and its subsequent generalizations.

2.1.2 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 2.1.1, a general formulation of MLPs with univariate outputs may be written as follows:

$$\hat{f}(\vec{x}) \quad = \quad h\left(\sum_{i=1}^{k} \delta_i h(\beta_{0i} + \vec{\beta}_{1i}'\vec{x}) + \delta_0\right) \qquad (5)$$

where $h(\cdot)$ is typically taken to be a smooth, monotonically increasing function such as the "sigmoid" function $1/(1+e^{-x})$, $\{\delta_i\}$ and $\vec{\beta}$ are coefficients, and k is the number of "hidden units". The specification (5) is generally termed an MLP with "one hidden layer" because the basic "sigmoid-of-a-dot-product" equation is nested once—the nesting may of course be repeated arbitrarily many times, hence the term "multilayer" perceptron. Unlike the RBF formulation, the nonlinear function h in the MLP formulation is usually fixed for the entire network.

For a given set of inputs $\{\vec{x}_t\}$ and outputs $\{y_t\}$, fitting an MLP model amounts to estimating the $(d+1)k$ parameters $\{\beta_{0i}\}$ and $\{\vec{\beta}_{1i}\}$, and the $k+1$ parameters $\{\delta_i\}$, for a total of $(d+2)k+1$ parameters.

2.1.3 Projection Pursuit Regression

Projection pursuit is a method that emerged from the statistics community for analyzing high-dimensional datasets by looking at their low-dimensional projections. Friedman and Stuetzle (1981) developed a version for the nonlinear regression problem called projection pursuit regression (PPR). Similar to MLPs, PPR models 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. Following the notation of Section 2.1.2, the formulation for PPR with univariate outputs can be written as

$$\hat{f}(\vec{x}) \quad = \quad \sum_{i=1}^{k} \delta_i h_i(\vec{\beta}_i'\vec{x}) + \delta_0 \qquad (6)$$

where the functions $h_i(\cdot)$ are estimated from the data [typically with a smoother], the $\{\delta_i\}$ and $\vec{\beta}$ are coefficients, and k is the number of projections. Note that δ_0 is commonly taken to be the sample mean of the outputs $f(\vec{x})$.

In counting the number of parameters that PPR models require, a difficulty arises in how to treat its use of smoothers in estimating the inner h functions. A naive approach is to count each smoothing estimator as a single parameter, its bandwidth. In this case, the total number of parameters is dk projection indices, k linear coefficients, and k smoothing bandwidths, for a total of $(d+2)k$ parameters. However, a more refined method of counting the degrees of freedom, e.g., Wahba (1990), may yield a slightly different count.

2.2 Network Properties

Although the various learning network techniques originated from a variety of backgrounds, with implications and characteristics that are not yet fully understood, some common and well-established properties are worth noting.

2.2.1 Approximation

All of the above learning networks have been shown to possess some form of a *universal approximation* property. For example, Huber (1985) and Jones (1987) prove that with sufficiently many terms, any square-integrable function can be approximated arbitrarily well by PPR. Cybenko (1988) and Hornik (1989) demonstrate 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—there is always a choice for the parameters that is better than any other possible choice—a property that is not shared by MLPs.

2.2.2 Error Convergence

The universal approximation results, however, say nothing about how easy it is to find those good approximations, or how computationally 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 wish to model.* For example, Barron (1991) derives 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) obtains similar results for PPR. Girosi and Anzellotti (1992) derive bounds on convergence in RBFs using somewhat more natural assumptions about the smoothness of the function being approximated. Niyogi and Girosi (1994) extend this result for the estimation problem, and derive a bound on the "generalization error" of RBFs, 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—number of parameters—increases, then after a certain point the error *increases* [see Figure 2]. For the financial modeling problems considered in this paper, 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.

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

2.2.3 Model Specification

A key question for most approximation techniques and in particular for neural network-like schemes concerns the type and the complexity of the model or the network to be used for a specific problem. Different approaches and different network architectures correspond to different choices of the space of approximating functions. A specific choice implies a specific assumption about the nature of the nonlinear relation to be approximated. For example, Girosi, Jones and Poggio (1993) have shown that different assumptions about smoothness of the function to be approximated lead to different approximation schemes, such as different types of Radial Basis Functions, as well as different kinds of splines and of ridge approximators. Certain classes of smoothness assumptions in the different variables even lead to multilayer perceptron architectures. The number of basis functions, and more in general of network parameters, is a related and difficult issue. Even if one type of architecture can be chosen based on prior knowledge about the smoothness to be expected in the specific problem, the question remains about the appropriate complexity of the architecture, that is the number of parameters. A general answer does not yet exist and is unlikely to be discovered any time soon. The standard approach to the problem relies on cross-validation techniques and variations of them [Wahba (1990)]. A related, more fundamental approach—called structural risk minimization—has been developed by Vapnik (1982).

2.2.4 Parameter Estimation Methods

In our discussion above, we have focused primarily on the specification of $\hat{f}(\cdot)$ for each method, but of course a critical concern is how each of the model's parameters are to be estimated. To some extent, the estimation issue may be divorced from the specification issue. Indeed, there is a large body of literature concerned solely with the estimation of network parameters. Much of this literature shows that the speed and accuracy of the estimation process depends on the kind of derivative information used, whether all parameters are estimated simultaneously or sequentially, and whether all the data is used at once in a "batch" mode or sequentially in an "on-line" mode. In Hutchinson (1993), estimation techniques for RBF networks are more fully explored.

However, a rigorous comparison of estimation methods is not the primary goal of our paper; rather, our objective 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. In particular we adopt Levenberg-Marquardt for batch mode estimation of the RBF networks, gradient descent [with momentum] for on-line 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].

Although not pursued here, readers interested in exploring the trade-offs between on-line and batch-mode estimation are encouraged to consult the "stochastic approximation" literature [see Robbins and Monro (19510, Ljung & Soderstrom (1986), and Widrow and Stearns (1985)]. In general, it is not known why on-line methods used with neural network techniques often seem to perform better than batch methods on large-scale, nonconvex problems. It seems difficult to extract any general conclusions from the diverse body of literature reporting the use of different on-line and batch techniques across many disparate applications.

2.2.5 Equivalence of Different Learning Networks

There is another reason that we do not focus on the merits of one type of learning network over another: recent theoretical developments suggest that there are significant connections between many of these networks. For example, Maruyama, Girosi, and Poggio (1991) show an equivalence between MLP networks with normalized inputs and RBF networks. Girosi, Jones and Poggio (1993) prove that a wide class of approximation schemes can be derived from regularization theory, including RBF networks and some forms of PPR and MLP networks. Nevertheless, we expect each formulation to be more efficient at approximating some functions than others, and as argued by Ng and Lippman (1991), the practical differences in using each method, e.g., in running time or memory used, may be more important than model accuracy.

3 Learning the Black-Scholes Formula

Given the power and flexibility of learning networks to approximate complex nonlinear relations, a natural application is to derivative securities whose pricing formulas are highly nonlinear even when they are available in closed form. In particular, we pose the following challenge: if option prices were truly determined by the Black-Scholes formula exactly, can learning networks "learn" the Black-Scholes formula? In more standard statistical jargon: can the Black-Scholes formula be estimated nonparametrically via learning networks with a sufficient degree of accuracy to be of practical use?

In this section, we face this challenge by performing Monte Carlo simulation experiments in which various learning networks are trained on artificially generated Black-Scholes option prices, and then compared to the Black-Scholes formula both analytically and in out-of-sample hedging experiments to see how close they come. Even with training sets of only six months of daily data, learning network pricing formulas can approximate the Black-Scholes formula with remarkable accuracy.

While the accuracy of the learning network *prices* is obviously of great interest, this alone is not sufficient to ensure the practical relevance of our nonparametric approach. In particular, the ability to *hedge* an option position is as important, since the very existence of an arbitrage-based pricing formula is predicated on the ability to replicate the option through a dynamic hedging strategy. This additional constraint motivates the regularization techniques and, in particular, the RBF networks used in this study. Specifically, delta-hedging strategies require an accurate approximation of the derivative of the underlying pricing formula, and the need for accurate approximations of derivatives leads directly to the smoothness constraint imposed by regularization techniques such as RBF networks.[4] Of course, whether or not the delta-hedging errors are sufficiently small in practice is an empirical matter, and we shall investigate these errors explicitly in our simulation experiments and empirical application described below.

However, the accuracy we desire cannot be achieved without placing some structure on the function to be approximated. For example, we begin by asserting that the option pricing formula $f(\cdot)$ is smooth in all its arguments, and that its arguments are: the stock price $S(t)$, the strike price X, and the time-to-maturity $T-t$. In fact, we know that the Black-Scholes formula also depends on the risk-free rate of interest r and the volatility σ of the underlying asset's continuously-compounded returns, e.g.,

$$C(t) = S(t)\Phi(d_1) - Xe^{-r(T-t)}\Phi(d_2) \tag{7}$$

where

$$\begin{aligned} d_1 &= \frac{\ln(S(t)/X) + (r + \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}}, \\ d_2 &= d_1 - \sigma\sqrt{T-t} \end{aligned}$$

and $\Phi(\cdot)$ is the standard normal cumulative distribution function. However, if r and σ are fixed throughout the network's training sample as we shall assume, then the dependence of the option's price on these two quantities cannot be identified by *any* nonparametric estimator of $f(\cdot)$ in the way that (7) does.[5] Of course, if interest rates and volatility vary through time as they do in practice, learning networks can readily capture their impact on option prices explicitly.

One further simplification we employ is to assume that the statistical distribution of the underlying asset's return is independent of the level of the stock price $S(t)$, hence by Theorem 8.9 of Merton (1990, Chapter 8), the option pricing formula $f(\cdot)$ is homogeneous of degree one in both $S(t)$ and X, so that we need only estimate $f(S(t)/X, 1, T-t)$. By requiring only two rather than three inputs to our learning networks we may be lessening the number of data points required for learning, but it should also be possible to relax these assumptions and use all three inputs.

We can now outline the components of our Monte Carlo simulation experiment, which consists of two phases: training and testing. The training phase entails generating sample paths of stock and option prices on which the learning networks are "trained", i.e., the network parameters are fitted to each sample path so as to minimize a quadratic loss function. This yields a network pricing formula which is then "tested" on newly-simulated sample paths of stock and option prices, i.e., various performance measures are calculated for the network pricing formula using the test path.

To obtain a measure of the success of the "average" network pricing formula, we repeat the training phase for many independent option/stock price sample paths, apply each network formula to the *same* test path, and average the performance measures across training paths. To obtain a measure of the "average success" of any given network pricing formula, we do the reverse: for a single training path, we apply the resulting network pricing formula on many independent option/stock price test paths, and average the performance measures across *test* paths.

Since we conduct multiple training-path and test-path simulations, our simulation design is best visualized as a matrix of results: each row corresponds to a separate and independent training path, each column corresponds to a separate and independent test path, and each cell contains the performance measures for a network trained

[4] In fact, it is well known that the problem of numerical differentiation is ill-posed. The classical approach [Rheinsch (1967)] is to regularize it by finding a sufficiently smooth function that solves the variational problem in (2). As we discussed earlier, RBF networks as well as splines and several forms of MLP networks follow directly from the regularization approach and are therefore expected to approximate not only the pricing formula but also its derivatives [provided the basis function corresponding to a smoothness prior is of a sufficient degree, see (Poggio and Girosi, 1991): in particular, the Gaussian is certainly sufficiently smooth for our problem]. A special case of this general argument is the result of Gallant and White (1992) and Hornik, Stinchcombe, and White (1990) who show that single-hidden-layer MLP networks can approximate the derivative of an arbitrary nonlinear mapping arbitrarily well as the number of hidden units increases.

[5] This is one sense in which analytical pricing formulas for derivative securities are preferred whenever available.

on a particular training path and applied to a particular test path. Therefore, the "average success" of a given network may be viewed as an average of the performance measures across the columns of a given row, and the performance of the "average network" on a given test path may be viewed as an average of the performance measures across the rows of a given column. Although these two averages obviously closely related, they do address different aspects of the performance of learning networks, and the results of each must be interpreted with the appropriate motivation in mind.

3.1 Calibrating the Simulations

In the first phase of our Monte Carlo simulation experiment—the training phase—we simulate a two-year sample of daily stock prices, and create a cross-section of options each day according to the rules used by the Chicago Board Options Exchange (CBOE) with prices given by the Black-Scholes formula. We refer to this two-year sample of stock and [multiple] option prices as a single "training path", since the network is trained on this sample.

We assume that the underlying asset for our simulation experiments is a "typical" NYSE stock, with an initial price $S(0)$ of \$50.00, an annual continuously-compounded expected rate of return μ of 10%, and an annual volatility σ of 20%. Under the Black-Scholes assumption of a geometric Brownian motion:

$$dS(t) \;=\; \mu S(t)dt \;+\; \sigma S(t)dW(t) \qquad (8)$$

and taking the number of days per year to be 253, we draw 506 pseudorandom variates Z_t from the distribution $N(\mu/253, \sigma^2/253)$ to obtain two years of daily continuously-compounded returns, which are converted to prices with the usual relation $S(t) = S(0)e^{\sum_{i=1}^{t} Z_i}$ for $t > 0$.

Given a simulated training path $\{S(t)\}$ of daily stock prices, we construct a corresponding path of option prices according to the rules of the Chicago Board Options Exchange (CBOE) for introducing options on stocks. Since a thorough description of these rules is unnecessary for our purposes, we summarize only the most salient features here.[6] At any one time, CBOE stock options outstanding on a particular stock have four unique expiration dates: the current month, the next month, and the following two expirations from a quarterly schedule. The CBOE sets strike prices at multiples of \$5 for stock prices in the \$25 to \$200 range, which all of our simulated prices fall into. 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—within \$1 in our simulations—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.[7] We assume that all of the options generated according to these rules are traded every day, although in practice, far-from-the-money and long-dated options are often very illiquid.

A typical training path is shown in Figure 3. We can also plot the training path as a 3-dimensional surface if we normalize stock and option prices by the appropriate strike price and consider the option price as a function of the form $f(S/X, 1, T-t)$ [see Figure 4]. 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 across sample paths. For our training set, the number of options per sample path range from 71 to 91, with an average of 81. The total number of data points range from 5,227 to 6,847, with an average of 6,001.

3.2 Training Network Pricing Formulas

Now we are set to estimate or train pricing formulas of the form of $f(S/X, 1, T-t)$ on the simulated training paths, using two "inputs": $S(t)/X$ and $T-t$. For comparison, we first estimate two simple linear models estimated using ordinary least squares (OLS). The first model is linear regression of the option price on $S(t)/X$ and $T-t$. The second is a pair of linear regressions, one for options currently in the money, and another for those currently out of the money. Typical estimates of these models are shown in Table 2.

Although these linear models seem to fit quite well, with R^2s well above 80%, they have particularly naive implications for delta-hedging strategies. In particular, delta-hedging with the first linear model would amount to purchasing a certain number of shares of stock in the beginning [0.6886 in the example in Table 2] and holding them until expiration, regardless of stock price movements during the option's life. The second linear model improves on this slightly by switching between hedging with a large number [0.9415 in Table 2b] and a small number of shares [0.1882 in Table 2c] depending on whether the current stock price is less than or greater than the strike price.

The nonlinear models obtained from learning networks, on the other hand, yield estimates of option prices and deltas that are difficult to distinguish visually from the true Black-Scholes values. An example of the estimates and errors for an RBF network is shown in Figure 5, which was estimated from the same data as the linear models from Table 2. The estimated equation for this particular RBF network is shown in Table 1. Observe from Table 1 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

[6] See Hull (1993) for more details.

[7] In our simulations, this was not done for options with less than one week to expiration.

our example. The largest errors in these networks tend to occur at the kink-point for options at the money at expiration, and also along the boundary of the sample points.

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.

Our choice of model-complexity is not arbitrary, and in fact is motivated by our desire to minimize error and maximize "fit" for out-of-sample data. In this regard, a critical issue in specifying learning networks is how many nonlinear terms—"hidden units", basis functions, projections, etc.—to use in the approximation. Following the discussion in Section 2.2.2, for actual market data, we might expect an optimal number of parameters that minimizes out-of-sample error. But in the simulations of this section, the data are noise-free [in the sense that there is a deterministic formula generating the outputs from the inputs], hence we are interested primarily in how quickly adding more parameters reduces the error. Preliminary out-of-sample tests with independent sample paths have indicated diminishing returns beyond 4 nonlinear terms [as measured by the percent of variance explained], thus we adopt this specification for all the learning networks considered in this paper.[8] In the next sections we will assess how well we have done in meeting our goal of minimizing out-of-sample error.

3.3 Performance Measures

Our learning networks estimate the option prices $\widehat{C/X}$, thus our first performance measure is simply the usual coefficient of determination, R^2, of those estimated values compared with the true option prices C/X, computed for the out-of-sample data.

However, the R^2 measure is not ideal for telling us the practical value of any improvement in pricing accuracy that the learning networks might give us. A more meaningful measure of performance for a given option pricing formula is the "tracking error" of various replicating portfolios designed to delta-hedge an option position, using the formula in question to calculate the hedge ratios or deltas. In particular, suppose at date 0 we sell one call option and undertake the usual dynamic trading strategy in stocks and bonds to hedge this call during its life. If we have correctly identified the option pricing model, and if we can costlessly and continuously hedge, then at expiration the combined value of our stock and bond positions should exactly offset the value of the call. The difference between the terminal value of the call and the terminal combined value of the stock and bond positions may then serve as a measure of the accuracy of our network approximation. Of course, since it is impossible to hedge continuously in practice, there will always be some tracking error due to discreteness, therefore we shall compare the RBF tracking error with the tracking error of discrete delta-hedging under the exact Black-Scholes formula.

More formally, denote by $V(t)$ the dollar value of our replicating portfolio at date t and let

$$V(t) \;=\; V_S(t) \;+\; V_B(t) \;+\; V_C(t) \tag{9}$$

where $V_S(t)$ is the dollar value of stocks, $V_B(t)$ is the dollar value of bonds, and $V_C(t)$ is the dollar value of call options held in the portfolio at date t. The initial composition of this portfolio at date 0 is assumed to be:

$$V_S(0) \;=\; S(0)\Delta_{\mathrm{RBF}}(0) \tag{10}$$

$$V_C(0) \;=\; -\,F_{\mathrm{BS}}(0) \tag{11}$$

$$V_B(0) \;=\; -\Big(V_S(0)+V_C(0)\Big) \tag{12}$$

where $F_{\mathrm{BS}}(\cdot)$ is the Black-Scholes call option pricing formula, $F_{\mathrm{RBF}}(\cdot)$ is its RBF approximation, and

$$\Delta_{\mathrm{RBF}}(t) \;\equiv\; \frac{\partial F_{\mathrm{RBF}}(t)}{\partial S}$$

The portfolio positions (10) – (12) represent the sale of one call option at date 0, priced according to the theoretical Black-Scholes formula $F_{\mathrm{BS}}(0)$, and the simultaneous purchase of $\Delta_{\mathrm{RBF}}(0)$ shares of stock at price $S(0)$, where $\Delta_{\mathrm{RBF}}(0)$ is the derivative of the RBF approximation $F_{\mathrm{RBF}}(0)$ with respect to the stock price.[9] Since the stock purchase is wholly financed by the combination of riskless borrowing and proceeds from the sale of the call option, the initial value of the replicating portfolio is identically zero, thus

$$V(0) \;=\; V_S(0) \;+\; V_B(0) \;+\; V_C(0) \;=\; 0\,.$$

Prior to expiration, and at discrete and regular intervals of length τ [which we take to be one day in our simulations], the stock and bond positions in the replicating portfolio will be rebalanced so as to satisfy the following relations:

$$V_S(t) \;=\; S(t)\Delta_{\mathrm{RBF}}(t), \tag{13}$$

$$V_B(t) \;=\; e^{r\tau}V_B(t-\tau) \;-\; S(t)\Big(\Delta_{\mathrm{RBF}}(t)-\Delta_{\mathrm{RBF}}(t-\tau)\Big) \tag{14}$$

where $t = k\tau \le T$ for some integer k. The tracking error of the replicating portfolio is then defined to be the value of the replicating portfolio $V(T)$ at expiration

[8] 4 nonlinear terms corresponds to approximately 20 total parameters.

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

date T. From this, we obtain the following performance measure:

$$\xi \equiv e^{-rT} \mathrm{E}[\, |V(T)|\,]\ . \tag{15}$$

The quantity ξ is simply the present value of the expected absolute tracking error of the replicating portfolio. Although for more complex option portfolios, ξ may not be the most relevant criterion, nevertheless ξ does provide some information about the accuracy of our option pricing formula.[10]

A third measure of performance may be defined by combining the information contained in the expected tracking error with the variance of the tracking error. In particular, we define the "prediction error" η as:

$$\eta \equiv e^{-rT} \sqrt{\mathrm{E}^2[V(T)] + \mathrm{Var}[V(T)]} \tag{16}$$

which is the present value of the square root of the sum of the squared expected tracking error and its variance. The inclusion of the variance of $V(T)$ is significant—the expected tracking error of a delta-hedging strategy might be zero, but the strategy is a poor one if the variance of the tracking error were large. We shall use all three measures R^2, ξ, and η in our performance analysis below.

3.4 Testing Network Pricing Formulas

To assess the quality of the RBF pricing formula obtained from each training path, we simulate an independent six-month sample of daily stock prices—a "test path"—and use the trained network to delta-hedge various options [individually, not as a portfolio] introduced at the *start* of the test path. By simulating many independent test paths, 500 in our case, and averaging the absolute tracking errors over these paths, we can obtain estimates $\hat{\xi}$ and $\hat{\eta}$ of the expected absolute tracking error ξ and the prediction error η for each of the ten network pricing formulas. The performance of the network delta-hedging strategy may then be compared to the performance of a delta-hedging strategy using the Black-Scholes formula.

3.4.1 Out-of-Sample R^2 Comparisons

As a preliminary check of out-of-sample performance, we observe that the pricing errors of the direct model outputs $\widehat{C/X}$ are typically quite small for all of the networks examined, with out-of-sample R^2's of 99% and above for the "average" network [except for the single linear model]. These results are presented in Table 3. From the minimum R^2 values, it is also evident that not all types of networks yield *consistently* good results, perhaps because of the stochastic nature of the respective estimation processes.

3.4.2 Tracking Error Comparisons

Table 4 reports selected raw simulation results for a call option with 3 months to expiration and a strike price X of \$50. In each row, the absolute tracking errors for delta-hedging this option are reported for the network pricing formula training on a single training path, the entries in each column corresponding to a different test path for which the absolute tracking error is calculated. For example, the $(1, 2)$-entry 0.2719 is the absolute tracking error for delta-hedging this 3-month \$50-strike option over test path #100, using the network pricing formula trained on training path #1.

For comparison, over the same test path the absolute tracking error for a delta-hedging strategy using the Black-Scholes formula is 0.3461, reported in the last row. The fact that the RBF network pricing formula can yield a smaller delta-hedging error than the Black-Scholes formula may seem counterintuitive. After all, the Black-Scholes formula is indeed the correct pricing formula in the context of our simulations. The source of this apparent paradox lies in the fact that we are delta-hedging *discretely* [once a day], whereas the Black-Scholes formula is based on a continuously-adjusted delta-hedging strategy. Therefore, even the Black-Scholes formula will exhibit some tracking error when applied to Black-Scholes prices at discrete time intervals. In such cases, an RBF pricing formula may well be more accurate since it is trained directly on the discretely-sampled data, and not based on a continuous-time approximation.

Of course, other columns in Table 4 show that Black-Scholes can perform significantly better than the RBF formula [for example, compare the $(1, 1)$-entry of 0.6968 with the Black-Scholes value of 0.0125]. Moreover, as the delta-hedging interval shrinks, the Black-Scholes formula will become increasingly more accurate and, in the limit, will have no tracking error whatsoever. However, since such a limit is empirically unattainable for a variety of institutional reasons, the benefits of network pricing formulas may be quite significant.

For a more complete comparison between RBF networks and the Black-Scholes formula across all 500 test paths, Table 5 reports the fraction of test paths for which each of the ten RBF networks exhibit *lower* absolute tracking error than the Black-Scholes formula. Similar comparisons are also performed for the single-regression model ["Linear-1"], the two-regression model ["Linear-2"], a projection pursuit regression ["PPR"] with four projections, and a multilayer perceptron ["MLP"] with one hidden layer containing four units.

The third column of entries in Table 5 show that in approximately 36 percent of the 500 test paths, RBF networks have lower tracking error than the Black-Scholes formula. For this particular option RBF networks and PPR networks have quite similar performance, and both are superior to the three other pricing models—the next closest competitor is the MLP, which outperforms the

[10] In particular, other statistics of the sample path $\{V(t)\}$ for the entire portfolio may be of more concern, such as its maximum and minimum, and the interaction between $\{V(t)\}$ and other asset returns.

Black-Scholes formula for approximately 26 percent of the test paths.

Of course, tracking errors tend to vary with the terms of the option such as its time-to-maturity and strike price. To gauge the accuracy of the RBF and other pricing models across these terms, we report in Tables 6 – 10 the fraction of test paths for which each of the four pricing models outperforms Black-Scholes for strike prices $X = 40$, 45, 50, 55, and 60, and times-to-maturity $T - t = 1$, 3, and 6 months.

Table 6 shows that the average RBF network—averaged over the ten training paths—performs reasonably well for near-the-money options at all three maturities, outperforming Black-Scholes between 12% and 36% of the time for options with strike prices between \$45 and \$55. As the maturity increases, the performance of the average RBF network improves for deep-out-of-the money options as well, outperforming Black-Scholes for 30% of the test paths for the call with a strike price of \$60.

Tables 7 and 8 provides similar comparisons for the average MLP and PPR networks, respectively—averaged over the same training paths as the RBF model—with similar results: good performance for near-the-money options at all maturities, and good performance for deep-out-of-the-money options at longer maturities.

Not surprisingly, Tables 9 and 10 show that the linear models exhibit considerably weaker performance than either of the network models, with fractions of outperforming test paths between 0.0% and 10.3% for the single-regression model, and between 0.0% and 14.6% for the two-regression model. However, these results do offer one important insight: even simple linear models can sometimes, albeit rarely, outperform the Black-Scholes model when delta-hedging is performed on a daily frequency.

Finally it is important to note that network pricing formulas should be monitored carefully for *extrapolation*. Because the networks are trained on a sampling of points covering a specific region of input space, it should not be surprising that they may not perform as well on points outside of this region. For example, Figure 6 illustrates that the worst tracking error for RBF networks in our simulations occurred for test data that was well outside of the range of the training data.

3.4.3 Prediction Error Comparisons

To complete our performance analysis of the networking option pricing formulas, we compare the estimated prediction errors $\hat{\eta}$ of the network delta-hedging strategies to those of the Black-Scholes formula. Recall from (16) that the prediction error combines the expectation and variance of the absolute tracking error, hence the estimated prediction error is calculated with the sample mean and sample variance of $|V(T)|$, taken over the 500 test paths. The benchmarks for comparison are the estimated prediction errors for the Black-Scholes delta-hedging strategy, given in Table 11.

Once again, we see from Table 11 that delta-hedging with the Black-Scholes at discrete intervals does not yield a perfect hedge. The estimated prediction errors are all strictly positive, and are larger for options near the money and with longer times-to-maturity.

However, under the prediction error performance measure the Black-Scholes formula is superior to all of the learning network approaches for this simulated data [see Tables 12 – 16]. For example, these tables show that the average RBF network has larger estimated prediction errors than Black-Scholes for all option types [although RBF networks have smaller errors than the other learning network types] and that the linear models are significantly worse than the others.[11] We also note that the pattern of errors is somewhat different for each learning network, indicating that each may have its own area of dominance.

Overall, we are encouraged by the ease with which the learning networks achieved error levels similar to those of the Black-Scholes formula, and on a problem posed in the latter's favor. We suspect that the learning network approach will be a promising alternative for pricing and hedging derivatives where there is uncertainty about the specification of the asset return process.

4 An Application to S&P 500 Futures Options

In Section 3 we have shown that learning networks can efficiently approximate the Black-Scholes pricing formula if the data were generated by it, and this provides some hope that our nonparametric approach may be useful in practice. After all, if there is some uncertainty about the parametric assumptions of a typical derivative pricing model, it should come as no surprise that a *nonparametric* model can improve pricing and hedging performance. To gauge the practical relevance of learning networks in at least one context, we apply it to the pricing and hedging of S&P 500 futures options, and compare it to the Black-Scholes model applied to the same data. Despite the fact that the Black-Scholes model is generally not used in its original form in practice, we focus on it here because it is still a widely-used benchmark model, and because it serves as an example of a parametric model whose assumptions are questionable in the context of this data.

[11] We caution the reader from drawing too strong a conclusion from the ordering of the RBF, MLP, and PPR results, however, due to the sensitivity of these nonparametric techniques to the "tuning" of their specifications, e.g., number of hidden nodes, network architecture, etc. In particular, the superiority of the RBF network results may be due to the fact that we have had more experience in tuning their specification.

4.1 The Data and Experimental Setup

The data for our empirical analysis are daily closing prices of S&P 500 futures and futures options for the 5-year period from January 1987 to December 1991. Futures prices over this period are shown in Figure 7. There were 24 different futures contracts and 998 futures call options active during this period.[12] The futures contracts have quarterly expirations, and on a typical day 40 to 50 call options based on 4 different futures contracts were traded.

Our specification is similar to that given in Section 3.1 for the simulated data. We divide the S&P 500 data into 10 non-overlapping six-month subperiods for training and testing the learning networks. Six-month subperiods were chosen to match approximately the number of data points in each training path with those of our simulations in Section 3. Data for the second half of 1989 is shown in Figures 8 and 9. Notable differences between this data and the simulated data of Section 3 are the presence of "noise" in the real data and the irregular trading activity of the options, especially for near-term out-of-the-money options.

For the S&P 500 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 4,454 to 8,301, with an average of 6,246. To limit the effects of nonstationarities and to 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 paths for each network. We also considered the last 7 test paths separately, i.e., data from July 1988 to December 1991, to assess the influence of the October 1987 crash on our results.

4.2 Estimating Black-Scholes Prices

Estimating and comparing models on the S&P 500 data will proceed much as it did in Section 3 for the linear and learning network models. However, the Black-Scholes parameters r and σ must be estimated when using actual market data. From a theoretical perspective, the 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 nonstationarities in interest rates and asset-return distributions. A common compromise is to estimate the parameters using only a window of the most recent data. We follow this latter approach for the S&P 500 data. Specifically, we estimate the Black-Scholes volatility σ for a given S&P 500 futures contract using

$$\hat{\sigma} = s/\sqrt{60} \tag{17}$$

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 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 10].

4.3 Out-of-Sample Pricing and Hedging

In this section we present the out-of-sample results of fitting the various models to the S&P 500 data. Based on our experience with the simulated data, we chose learning networks with 4 nonlinear terms as a good compromise between accuracy and complexity, although it may be worth re-examining this trade-off on actual S&P 500 data.[13]

The out-of-sample tests show some evidence that the learning networks outperform the naive Black-Scholes model on this data. This is hardly surprising, given the fact that many of the assumptions of the Black-Scholes formula are violated by the data, e.g., geometric Brownian motion, constant volatility, frictionless markets, etc.

As with the simulated-data-trained learning networks, the performance of each of actual-data-trained networks varied over the input space. To see how the performance varies in particular, we divide each dimension of the input space into three regimes: long-, medium-, and short-term for the time-to-expiration ($T-t$) input, and in-, near-, and out-of-the-money for the stock-price/strike-price (S/X) input. Specifically, breakpoints of 2 and 5 months for the $T-t$ input and 0.97 and 1.03 for the S/X input were chosen to yield approximately the same number of datapoints in each of the 9 paired categories. The delta-hedging prediction errors, broken down by these maturity/richness groups, are shown in Tables 17 and 18. Interestingly, results from the subperiods influenced by the October 1987 crash still yield lower prediction errors for the learning networks than for the Black-Scholes model, except for near-term in-the-money options.

For completeness we also show the out-of-sample R^2's [see Table 19] and the absolute hedging error comparison [see Table 20] as we did in Section 3.4 for the synthetic data. Table 19, for instance, shows that the average out-of-sample R^2 of roughly 85% for the estimated Black-Scholes model is somewhat worse than that of the other network models. Note however that unlike the case for our synthetic data, the options in the S&P 500 data set are *not* independent, and thus we must look at these results with caution. Furthermore, we only have one test set for each trained network, and thus for the hedging error comparison in Table 20 we show these results broken down by test period instead of the summary

[12] For simplicity, we focus only on call options in our analysis.

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

statistics shown in Section 3.4.2. Nonetheless, this table shows that the learning networks exhibit less hedging error than the estimated Black-Scholes formula in a substantial fraction of the options tested—up to 65% of the options tested against the MLP network for the July – December 1990 testing period.

From these results, it is difficult to infer which network type performs best in general. Hypothesis tests concerning the relative sizes of hedging error are difficult to formulate precisely because of the statistical dependence of the option-price paths. Focusing on a single non-overlapping sequence of options would solve the dependence problem, but would throw out 98% of the available options. Instead, we present a less formal test on all of the data, but caution the reader 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 [one-sided] alternative hypothesis is that the difference is positive, i.e., the learning network hedging error is smaller. Results of this simple test show evidence that all three learning networks outperform the Black-Scholes model, while the linear models do not [see Table 21].

It is also interesting to compare the computing time required to estimate these models, although no effort was made to optimize our code, nor did we attempt to optimize the estimation method for each type of learning network. With these qualifications in mind, we find that second order methods seem preferred for our application. For example, the MLP network gradient descent equations were updated for 10,000 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.

5 Conclusions

Although parametric derivative pricing formulas are preferred when they are available, our results show that nonparametric learning-network alternatives can be useful substitutes when parametric methods fail. While our findings are promising, we cannot yet claim that our approach will be successful in general—for simplicity, our simulations have focused only on the Black-Scholes model, and our application has focused only on a single instrument and time period, S&P 500 futures options for 1987 to 1991. In particular, there are a host of parametric derivative pricing models, as well as many practical extensions of these models that may improve their performance on any particular data set. We hope to provide a more comprehensive analysis of these alternatives in the near future.

However, we do believe there is reason to be cautiously optimistic about our general approach, with a number of promising directions for future research. Perhaps the most pressing item on this agenda is the specification of additional inputs, inputs that are not readily captured by parametric models such as the return on the market, general market volatility, and other measures of business conditions. A related issue is the incorporation of the predictability of the underlying asset's return, and cross-predictability among several correlated assets [see Lo and Wang (1993) for a parametric example]. This may involve the construction of a factor model of the underlying asset's return and volatility processes.

Other research directions are motivated by the need for proper statistical inference in the specification of learning networks. First, we require some method of matching the network architecture—number of nonlinear units, number of centers, type of basis functions, etc.—to the specific dataset at hand in some optimal [and, preferably, automatic] fashion.

Second, the relation between sample size and approximation error should be explored, either analytically or through additional Monte Carlo simulation experiments. Perhaps some data-dependent metric can be constructed, such as the model prediction error, that can provide real-time estimates of approximation errors in much the same way that standard errors may be obtained for typical statistical estimators.

And finally, the need for better performance measures is clear. While typical measures of goodness-of-fit such as R^2 do offer some guidance for model selection, they are only incomplete measures of performance. Moreover, the notion of degrees of freedom is no longer well-defined for nonlinear models, and this has implications for all statistical measures of fit.

Further Acknowledgements: We thank Harrison Hong and Terence Lim for excellent research assistance, and Petr Adamek, Federico Girosi, Chung-Ming Kuan, Barbara Jansen, Blake LeBaron, and seminar participants at the DAIS Conference, the Harvard Business School, and the American Finance Association for helpful comments and discussion.

References

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

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

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

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

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

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

[9] A. Gallant and H. White. On learning the derivatives of an unknown mapping with multilayer feedforward networks. *Neural Networks*, 5:128–138, 1992.

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

[11] 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.

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

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

[14] K. Hornik, M. Stinchcombe, and H. White. Universal approximation of an unknown mapping and its derivatives. *Neural Networks*, 3:551–560, 1990.

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

[16] J. C. Hull. *Options, Futures, and Other Derivative Securities.* Prentice-Hall, Englewood Cliffs, New Jersey, 2nd edition, 1993.

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

[18] L. Ljung and T. Söderström. *Theory and Practice of Recursive Identification.* MIT Press, Cambridge, 1986.

[19] A. Lo and J. Wang. Implementing option pricing models when asset returns are predictable. Research Program in Computational Finance Working Paper RPCF–1001–93, MIT Sloan School of Management, 1993.

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

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

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

[23] 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.

[24] P. Niyogi and F. Girosi. On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions. A.I. Memo 1467, MIT Artificial Intelligence Laboratory, 1994.

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

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

[27] 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.

[28] C. H. Reinsch. Smoothing by spline functions. *Numer Math*, 10:177–183, 1967.

[29] H. Robbins and S. Monro. A stochastic approximation model. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[30] 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.

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

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

[33] B. Widrow and S. D. Stearns. *Adaptive Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1985.

$$
\begin{aligned}
\widehat{C/X} \quad = \quad & - 0.06\sqrt{\begin{bmatrix} S/X - 1.35 \\ T-t-0.45 \end{bmatrix}' \begin{bmatrix} 59.79 & -0.03 \\ -0.03 & 10.24 \end{bmatrix} \begin{bmatrix} S/X - 1.35 \\ T-t-0.45 \end{bmatrix} + 2.55} \\
& - 0.03\sqrt{\begin{bmatrix} S/X - 1.18 \\ T-t-0.24 \end{bmatrix}' \begin{bmatrix} 59.79 & -0.03 \\ -0.03 & 10.24 \end{bmatrix} \begin{bmatrix} S/X - 1.18 \\ T-t-0.24 \end{bmatrix} + 1.97} \\
& + 0.03\sqrt{\begin{bmatrix} S/X - 0.98 \\ T-t+0.20 \end{bmatrix}' \begin{bmatrix} 59.79 & -0.03 \\ -0.03 & 10.24 \end{bmatrix} \begin{bmatrix} S/X - 0.98 \\ T-t+0.20 \end{bmatrix} + 0.00} \\
& + 0.10\sqrt{\begin{bmatrix} S/X - 1.05 \\ T-t+0.10 \end{bmatrix}' \begin{bmatrix} 59.79 & -0.03 \\ -0.03 & 10.24 \end{bmatrix} \begin{bmatrix} S/X - 1.05 \\ T-t+0.10 \end{bmatrix} + 1.62} \\
& + 0.14 S/X \; - \; 0.24(T-t) \; - \; 0.01 \, .
\end{aligned}
$$

Table 1: Example estimated RBF equation from Section 3.2.

Residual Standard Error = 0.027, $R^2 = 0.9098$, $N = 6782$
$F_{2,6779}$-statistic = 34184.97, 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
$T-t$	0.0688	0.0018	38.5834	0

(a) Single linear model.

Residual Standard Error = 0.0062, $R^2 = 0.9955$, $N = 3489$
$F_{2,3486}$-statistic = 385583.4, 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
$T-t$	0.0858	0.0006	150.6208	0

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

Residual Standard Error = 0.007, $R^2 = 0.8557$, $N = 3293$
$F_{2,3290}$-statistic = 9753.782, 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
$T-t$	0.0728	0.0007	108.2335	0

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

Table 2: Regression summaries for typical linear models.

	Linear-1	Linear-2	RBF	PPR	MLP	B-S
Min	14.72	94.34	98.58	55.23	76.60	100.00
Mean	83.40	99.27	99.95	99.08	99.48	100.00
Max	95.57	99.82	99.99	100.00	99.96	100.00

Table 3: Out-of-sample R^2 values [in percent] for the learning networks, summarized across all training and out-of-sample test sets. "Linear-1" refers to the single-regression model of the data; "Linear-2" refers to the two-regression model, one for in-the-money options and one for out-of-the-money options; "RBF" refers to a radial-basis-function network with 4 multiquadric centers and an output sigmoid; "PPR" refers to a projection pursuit regression with four projections; and "MLP" refers to a multilayer perceptron with a single hidden layer containing four units.

	Test #100	Test #200	Test #300	Test #400	Test #500
Train #1	0.6968	0.2719	0.1154	0.0018	0.5870
Train #2	0.6536	0.2667	0.0882	0.0903	0.5523
Train #3	0.6832	0.2622	0.0698	0.0370	0.5534
Train #4	0.7175	0.2682	0.0955	0.0155	0.5918
Train #5	0.6938	0.2767	0.1055	0.0229	0.5993
Train #6	0.6755	0.2692	0.1085	0.0083	0.5600
Train #7	0.6971	0.2690	0.1104	0.0054	0.5809
Train #8	0.7075	0.2717	0.1087	0.0022	0.5859
Train #9	0.6571	0.2652	0.1016	0.0013	0.5389
Train #10	0.7105	0.2706	0.1135	0.0038	0.5913
B-S	0.0125	0.3461	0.0059	0.0677	0.0492

Table 4: Simulations of absolute delta-hedging errors for RBF networks for an at-the-money call option with $X = 50$, $T-t = 3$ months, and Black-Scholes price \$2.2867. The current stock price $S(0)$ is assumed to be \$50. The last row displays the same errors for the Black-Scholes formula.

	Linear-1	Linear-2	RBF	PPR	MLP
Train #1	0.062	0.102	0.354	0.362	0.260
	(0.011)	(0.014)	(0.021)	(0.021)	(0.020)
Train #2	0.048	0.112	0.340	0.390	0.264
	(0.010)	(0.014)	(0.021)	(0.022)	(0.020)
Train #3	0.088	0.108	0.380	0.350	0.268
	(0.013)	(0.014)	(0.022)	(0.021)	(0.020)
Train #4	0.084	0.098	0.370	0.340	0.254
	(0.012)	(0.013)	(0.022)	(0.021)	(0.019)
Train #5	0.062	0.100	0.358	0.360	0.278
	(0.011)	(0.013)	(0.021)	(0.021)	(0.020)
Train #6	0.056	0.108	0.364	0.378	0.274
	(0.010)	(0.014)	(0.022)	(0.022)	(0.020)
Train #7	0.084	0.102	0.368	0.362	0.272
	(0.012)	(0.014)	(0.022)	(0.021)	(0.020)
Train #8	0.080	0.104	0.358	0.328	0.262
	(0.012)	(0.014)	(0.021)	(0.021)	(0.020)
Train #9	0.066	0.104	0.368	0.374	0.272
	(0.011)	(0.014)	(0.022)	(0.022)	(0.020)
Train #10	0.080	0.104	0.354	0.382	0.280
	(0.012)	(0.014)	(0.021)	(0.022)	(0.020)

Table 5: Fraction of 500 test sets in which the absolute delta-hedging error was lower than Black-Scholes for an at-the-money call option with $X = 50$, $T-t = 3$ months, and Black-Scholes price \$2.2867 [standard errors are given in parentheses]. The current stock price $S(0)$ is assumed to be \$50.

RBF		$X=40$	$X=45$	$X=50$	$X=55$	$X=60$
$T-t=1$	Mean	0.001	0.120	0.278	0.266	0.032
	(SE)	(0.000)	(0.005)	(0.006)	(0.006)	(0.003)
	Min	0.000	0.108	0.270	0.176	0.022
	Max	0.002	0.140	0.284	0.332	0.040
$T-t=3$	Mean	0.072	0.296	0.361	0.269	0.254
	(SE)	(0.004)	(0.006)	(0.007)	(0.006)	(0.006)
	Min	0.054	0.242	0.340	0.248	0.170
	Max	0.084	0.336	0.380	0.322	0.336
$T-t=6$	Mean	0.164	0.263	0.316	0.243	0.304
	(SE)	(0.005)	(0.006)	(0.007)	(0.006)	(0.007)
	Min	0.120	0.220	0.298	0.234	0.276
	Max	0.200	0.310	0.324	0.258	0.320

Table 6: Fraction of 500 test sets in which the absolute delta-hedging error using an RBF network with 4 multiquadric centers and an output sigmoid is lower than the Black-Scholes delta-hedging error, for call options with strike price X and time-to-maturity $T-t$ months on a non-dividend-paying stock currently priced at \$50. Within each panel, the top entry of each column is the average of this fraction across the 10 training paths, the second entry [in parentheses] is the standard error of that average, and the third and fourth entries are the minimum and maximum across the 10 training paths.

MLP		$X=40$	$X=45$	$X=50$	$X=55$	$X=60$
$T-t=1$	Mean	0.000	0.046	0.238	0.125	0.019
	(SE)	(0.000)	(0.003)	(0.006)	(0.005)	(0.002)
	Min	0.000	0.034	0.228	0.110	0.008
	Max	0.000	0.066	0.246	0.132	0.028
$T-t=3$	Mean	0.022	0.174	0.268	0.354	0.280
	(SE)	(0.002)	(0.005)	(0.006)	(0.007)	(0.006)
	Min	0.004	0.130	0.254	0.324	0.216
	Max	0.040	0.220	0.280	0.386	0.384
$T-t=6$	Mean	0.030	0.187	0.252	0.330	0.253
	(SE)	(0.002)	(0.006)	(0.006)	(0.007)	(0.006)
	Min	0.004	0.152	0.204	0.298	0.216
	Max	0.074	0.212	0.302	0.354	0.274

Table 7: Fraction of 500 test sets in which the absolute delta-hedging error using an MLP network with a single hidden layer containing four units is lower than the Black-Scholes delta-hedging error, for call options with strike price X and time-to-maturity $T-t$ months on a non-dividend-paying stock currently priced at \$50. See Table 6 for details.

PPR		$X=40$	$X=45$	$X=50$	$X=55$	$X=60$
$T-t=1$	Mean	0.000	0.165	0.316	0.303	0.024
	(SE)	(0.000)	(0.005)	(0.007)	(0.006)	(0.002)
	Min	0.000	0.118	0.272	0.208	0.006
	Max	0.002	0.198	0.394	0.364	0.052
$T-t=3$	Mean	0.060	0.282	0.363	0.325	0.177
	(SE)	(0.003)	(0.006)	(0.007)	(0.007)	(0.005)
	Min	0.006	0.202	0.328	0.244	0.076
	Max	0.126	0.344	0.390	0.420	0.286
$T-t=6$	Mean	0.125	0.287	0.315	0.293	0.197
	(SE)	(0.005)	(0.006)	(0.007)	(0.006)	(0.006)
	Min	0.020	0.190	0.290	0.234	0.116
	Max	0.202	0.346	0.352	0.358	0.286

Table 8: Fraction of 500 test sets in which the absolute delta-hedging error using a PPR network with four projections is lower than the Black-Scholes delta-hedging error, for call options with strike price X and time-to-maturity $T-t$ months on a non-dividend-paying stock currently priced at \$50. See Table 6 for details.

Linear-1		$X=40$	$X=45$	$X=50$	$X=55$	$X=60$
$T-t=1$	Mean	0.000	0.020	0.103	0.016	0.002
	(SE)	(0.000)	(0.002)	(0.004)	(0.002)	(0.001)
	Min	0.000	0.012	0.068	0.010	0.002
	Max	0.000	0.032	0.124	0.026	0.002
$T-t=3$	Mean	0.003	0.029	0.071	0.018	0.007
	(SE)	(0.001)	(0.002)	(0.004)	(0.002)	(0.001)
	Min	0.000	0.016	0.048	0.010	0.006
	Max	0.010	0.060	0.088	0.032	0.012
$T-t=6$	Mean	0.012	0.035	0.039	0.037	0.019
	(SE)	(0.002)	(0.003)	(0.003)	(0.003)	(0.002)
	Min	0.010	0.026	0.024	0.034	0.010
	Max	0.016	0.046	0.050	0.042	0.026

Table 9: Fraction of 500 test sets in which the absolute delta-hedging error using a single-regression model is lower than the Black-Scholes delta-hedging error, for call options with strike price X and time-to-maturity $T-t$ months on a non-dividend-paying stock currently priced at \$50. See Table 6 for details.

Linear-2		$X=40$	$X=45$	$X=50$	$X=55$	$X=60$
$T-t=1$	Mean	0.000	0.080	0.146	0.068	0.004
	(SE)	(0.000)	(0.004)	(0.005)	(0.004)	(0.001)
	Min	0.000	0.060	0.128	0.058	0.004
	Max	0.000	0.090	0.170	0.092	0.004
$T-t=3$	Mean	0.018	0.107	0.104	0.095	0.033
	(SE)	(0.002)	(0.004)	(0.004)	(0.004)	(0.003)
	Min	0.010	0.088	0.098	0.080	0.020
	Max	0.024	0.116	0.112	0.112	0.052
$T-t=6$	Mean	0.045	0.082	0.072	0.082	0.059
	(SE)	(0.003)	(0.004)	(0.004)	(0.004)	(0.003)
	Min	0.032	0.074	0.056	0.068	0.038
	Max	0.054	0.090	0.080	0.096	0.072

Table 10: Fraction of 500 test sets in which the absolute delta-hedging error using a two-regression model is lower than the Black-Scholes delta-hedging error, for call options with strike price X and time-to-maturity $T-t$ months on a non-dividend-paying stock currently priced at \$50. See Table 6 for details.

B-S	$X=40$	$X=45$	$X=50$	$X=55$	$X=60$
$T-t=1$	0.001	0.069	0.217	0.116	0.007
$T-t=3$	0.043	0.146	0.213	0.155	0.098
$T-t=6$	0.088	0.157	0.208	0.211	0.147

Table 11: Estimated prediction errors for the absolute tracking error of a delta-hedging strategy using the Black-Scholes formula, for call options with strike price X and time-to-maturity $T-t$ months on a non-dividend-paying stock currently priced at \$50, estimated across 500 independent test paths. Since the Black-Scholes parameters are assumed to be known, not estimated, these errors do not vary across training paths.

RBF		$X=40$	$X=45$	$X=50$	$X=55$	$X=60$
$T-t=1$	Mean	0.044	0.164	0.310	0.157	0.039
	(SE)	(0.003)	(0.002)	(0.002)	(0.001)	(0.001)
	Min	0.031	0.150	0.298	0.152	0.035
	Max	0.059	0.172	0.316	0.163	0.045
$T-t=3$	Mean	0.142	0.215	0.296	0.257	0.155
	(SE)	(0.008)	(0.002)	(0.001)	(0.001)	(0.001)
	Min	0.113	0.208	0.291	0.249	0.152
	Max	0.177	0.225	0.299	0.263	0.161
$T-t=6$	Mean	0.286	0.271	0.309	0.340	0.214
	(SE)	(0.011)	(0.006)	(0.002)	(0.002)	(0.001)
	Min	0.236	0.243	0.299	0.329	0.207
	Max	0.334	0.300	0.315	0.347	0.224

Table 12: Estimated prediction errors for the absolute tracking error of a delta-hedging strategy using an RBF network with 4 multiquadric centers and an output sigmoid, for call options with strike price X and time-to-maturity $T-t$ months on a non-dividend-paying stock currently priced at \$50, estimated across 500 independent test paths. Within each panel, the top entry of each column is the average of the estimated prediction error across the 10 training paths, the second entry [in parentheses] is the standard error of that average, and the third and fourth entries are the minimum and maximum across the 10 training paths.

MLP		$X=40$	$X=45$	$X=50$	$X=55$	$X=60$
$T-t=1$	Mean	0.214	0.264	0.389	0.209	0.060
	(SE)	(0.024)	(0.008)	(0.006)	(0.004)	(0.002)
	Min	0.124	0.228	0.365	0.194	0.050
	Max	0.386	0.314	0.429	0.234	0.075
$T-t=3$	Mean	0.690	0.323	0.366	0.285	0.178
	(SE)	(0.118)	(0.016)	(0.003)	(0.004)	(0.002)
	Min	0.271	0.261	0.356	0.270	0.171
	Max	1.477	0.417	0.388	0.308	0.194
$T-t=6$	Mean	1.187	0.733	0.400	0.356	0.264
	(SE)	(0.174)	(0.087)	(0.007)	(0.004)	(0.002)
	Min	0.538	0.425	0.373	0.344	0.255
	Max	2.377	1.352	0.448	0.377	0.274

Table 13: Estimated prediction errors for the absolute tracking error of a delta-hedging strategy using an MLP network with a single hidden layer containing four units, for call options with strike price X and time-to-maturity $T-t$ months on a non-dividend-paying stock currently priced at \$50, estimated across 500 independent test paths. See Table 12 for further details.

PPR		$X=40$	$X=45$	$X=50$	$X=55$	$X=60$
$T-t=1$	Mean	0.198	0.121	0.271	0.147	0.081
	(SE)	(0.094)	(0.005)	(0.006)	(0.004)	(0.024)
	Min	0.028	0.101	0.245	0.131	0.028
	Max	0.991	0.144	0.301	0.167	0.261
$T-t=3$	Mean	1.180	0.275	0.276	0.238	0.247
	(SE)	(0.299)	(0.056)	(0.006)	(0.011)	(0.046)
	Min	0.134	0.174	0.254	0.202	0.136
	Max	3.113	0.759	0.309	0.320	0.555
$T-t=6$	Mean	2.140	1.056	0.383	0.367	0.443
	(SE)	(0.383)	(0.201)	(0.045)	(0.029)	(0.074)
	Min	0.511	0.246	0.259	0.268	0.224
	Max	4.337	2.325	0.719	0.589	0.931

Table 14: Estimated prediction errors for the absolute tracking error of a delta-hedging strategy using a PPR network with four projections, for call options with strike price X and time-to-maturity T–t months on a non-dividend-paying stock currently priced at \$50, estimated across 500 independent test paths. See Table 12 for further details.

Linear-1		$X=40$	$X=45$	$X=50$	$X=55$	$X=60$
$T-t=1$	Mean	1.047	0.967	0.911	1.672	1.879
	(SE)	(0.096)	(0.091)	(0.036)	(0.091)	(0.098)
	Min	0.561	0.507	0.813	1.251	1.425
	Max	1.492	1.393	1.132	2.135	2.375
$T-t=3$	Mean	1.849	1.486	1.697	2.624	3.015
	(SE)	(0.172)	(0.117)	(0.049)	(0.153)	(0.163)
	Min	0.983	0.959	1.580	1.936	2.260
	Max	2.649	2.091	2.013	3.411	3.845
$T-t=6$	Mean	2.276	2.124	2.170	2.910	3.780
	(SE)	(0.213)	(0.149)	(0.073)	(0.173)	(0.214)
	Min	1.208	1.495	2.000	2.170	2.805
	Max	3.275	2.926	2.629	3.821	4.879

Table 15: Estimated prediction errors for the absolute tracking error of a delta-hedging strategy using a single-regression model, for call options with strike price X and time-to-maturity $T-t$ months on a non-dividend-paying stock currently priced at \$50, estimated across 500 independent test paths. See Table 12 for further details.

Linear-2		$X=40$	$X=45$	$X=50$	$X=55$	$X=60$
$T-t=1$	Mean	0.212	0.207	0.724	0.455	0.518
	(SE)	(0.018)	(0.013)	(0.011)	(0.034)	(0.045)
	Min	0.154	0.168	0.681	0.335	0.344
	Max	0.340	0.304	0.776	0.628	0.739
$T-t=3$	Mean	0.371	0.555	1.054	0.836	0.790
	(SE)	(0.029)	(0.003)	(0.013)	(0.024)	(0.067)
	Min	0.277	0.539	0.995	0.767	0.539
	Max	0.586	0.566	1.118	0.972	1.130
$T-t=6$	Mean	0.500	0.955	1.544	1.454	1.042
	(SE)	(0.027)	(0.008)	(0.022)	(0.019)	(0.055)
	Min	0.412	0.909	1.452	1.373	0.880
	Max	0.709	0.988	1.650	1.563	1.342

Table 16: Estimated prediction errors for the absolute tracking error of a delta-hedging strategy using a two-regression model, for call options with strike price X and time-to-maturity T–t months on a non-dividend-paying stock currently priced at \$50, estimated across 500 independent test paths. See Table 12 for further details.

Short term	Linear-1	Linear-2	RBF	PPR	MLP	B-S	$\bar{C}(0)$
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-1**	**Linear-2**	**RBF**	**PPR**	**MLP**	**B-S**	$\bar{C}(0)$
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-1**	**Linear-2**	**RBF**	**PPR**	**MLP**	**B-S**	$\bar{C}(0)$
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 17: Delta-hedging prediction error for the out-of-sample S&P 500 data from July 1988 to December 1991, i.e., excluding the subperiods directly influenced by the October 1987 crash, averaged across all training/test sets.

Short term	Linear-1	Linear-2	RBF	PPR	MLP	B-S	$\bar{C}(0)$
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-1**	**Linear-2**	**RBF**	**PPR**	**MLP**	**B-S**	$\bar{C}(0)$
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
Long term	**Linear-1**	**Linear-2**	**RBF**	**PPR**	**MLP**	**B-S**	$\bar{C}(0)$
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 18: Delta-hedging prediction error for the out-of-sample S&P 500 data from July 1987 to July 1988, i.e., the subperiods directly influenced by the October 1987 crash, averaged across all training/test sets.

	Linear-1	Linear-2	RBF	PPR	MLP	B-S
Min	7.85	82.63	81.33	92.26	92.28	37.41
Mean	75.57	95.54	93.26	96.56	95.53	84.76
Max	95.74	99.44	98.41	99.54	98.98	99.22

Table 19: Out-of-sample R^2 values [in percent] for the learning networks, summarized across the 9 out-of-sample S&P 500 futures options test sets.

	Linear-1	Linear-2	RBF	PPR	MLP
Jul 87 - Dec 87	0.160	0.377	0.506	0.593	0.580
Jan 88 - Jun 88	0.189	0.357	0.476	0.497	0.538
Jul 88 - Dec 88	0.122	0.341	0.382	0.358	0.301
Jan 89 - Jun 89	0.221	0.405	0.534	0.550	0.481
Jul 89 - Dec 89	0.355	0.428	0.529	0.609	0.543
Jan 90 - Jun 90	0.329	0.423	0.557	0.550	0.631
Jul 90 - Dec 90	0.230	0.425	0.540	0.569	0.649
Jan 91 - Jun 91	0.296	0.419	0.497	0.346	0.313
Jul 91 - Dec 91	0.248	0.337	0.218	0.327	0.317

Table 20: Fraction of out-of-sample test set S&P 500 futures options in which the absolute delta-hedging error for each learning network was lower than the Black-Scholes delta-hedging error, shown for each test period.

Pair	t-statistic	p-value
Linear-1 vs B-S	-15.1265	1.0000
Linear-2 vs B-S	-5.7662	1.0000
RBF vs B-S	2.1098	0.0175
PPR vs B-S	2.0564	0.02
MLP vs B-S	3.7818	0.0001

Table 21: Paired t-test comparing relative magnitudes of absolute hedging error, using results from all S&P 500 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.

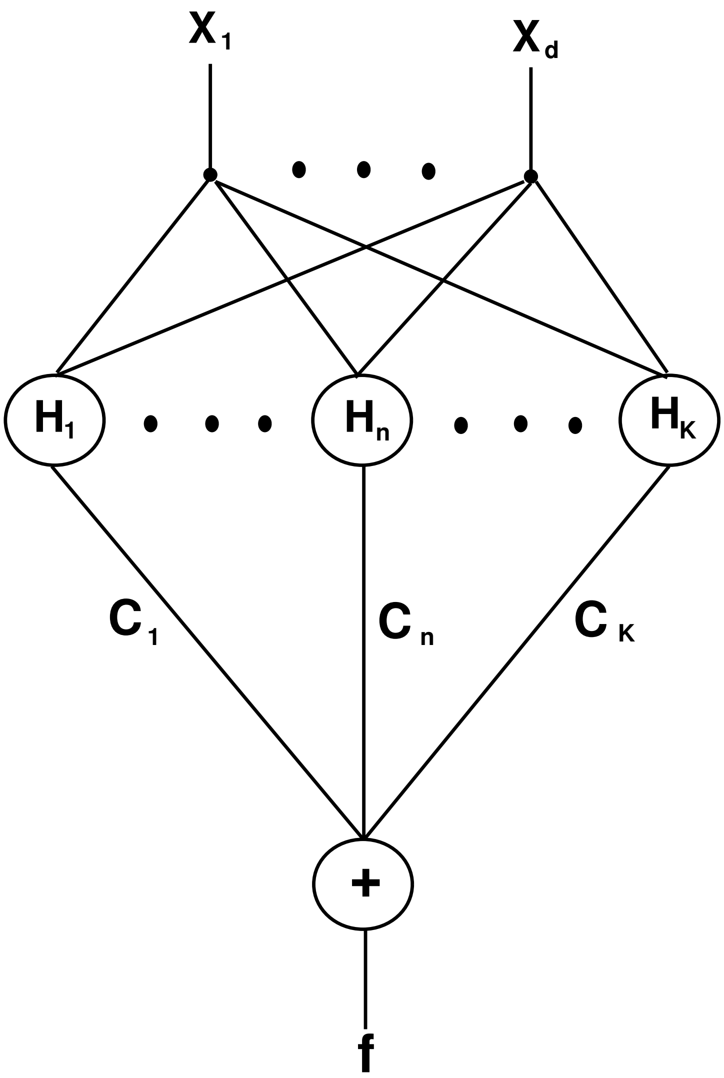


Figure 1: Structure of the learning networks used in this paper.

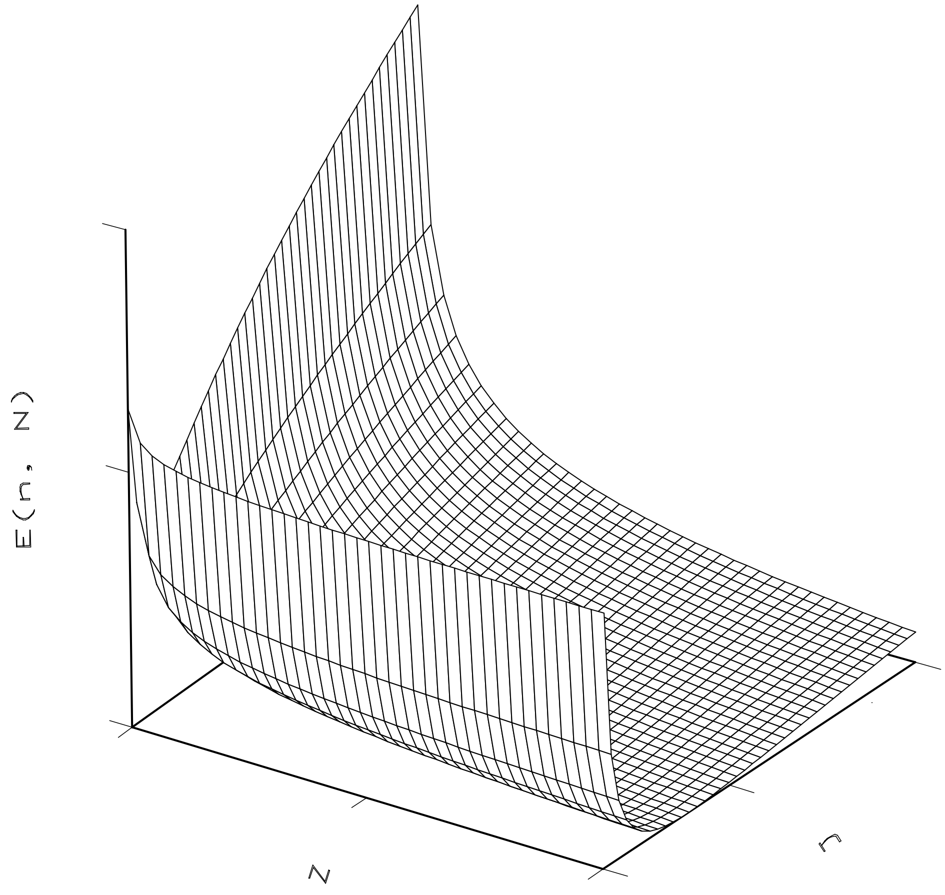


Figure 2: Generalization error $E(N, n)$ for a Gaussian RBF network as a function of the number of data points N and the number of network parameters n [reprinted with permission from Niyogi and Girosi (1994)].

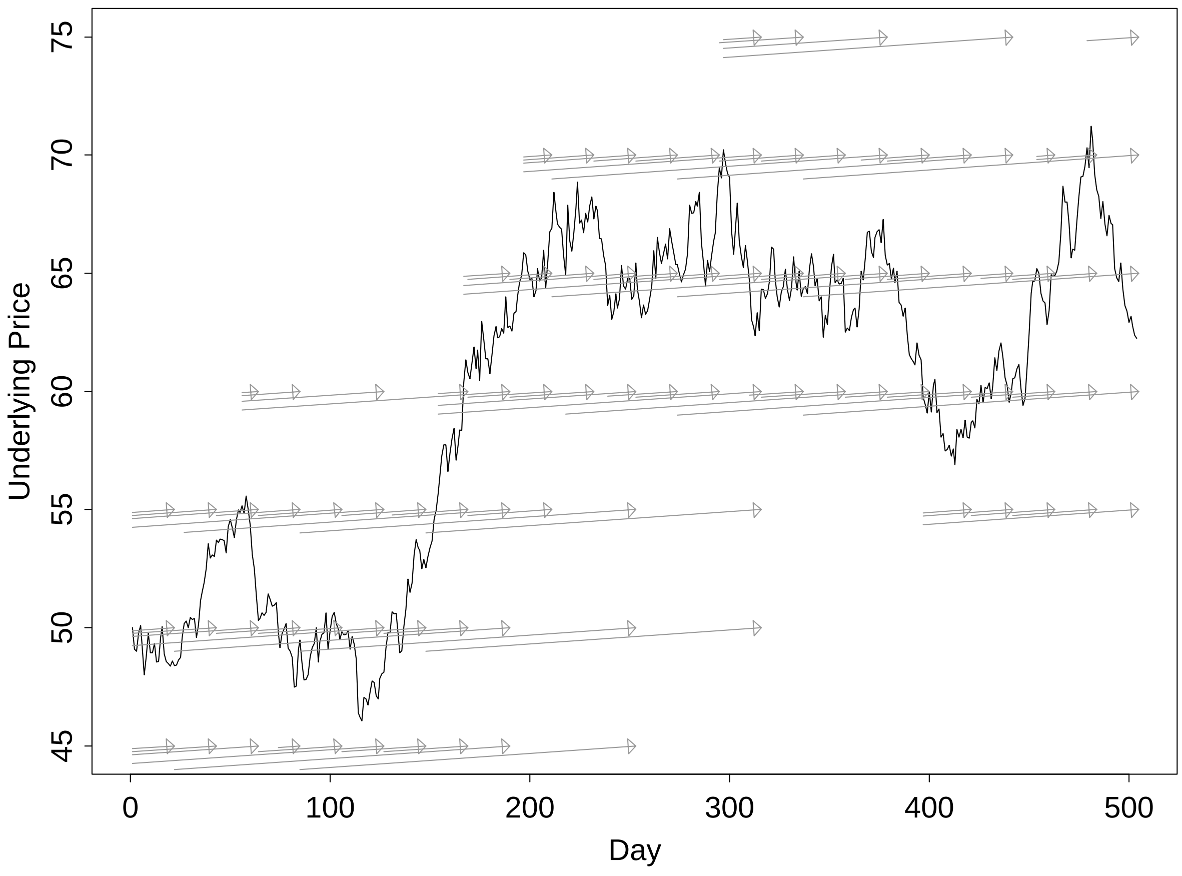


Figure 3: Typical simulated training path [see the text for parameters]. Dashed line represents stock price, while the arrows represent the options on the stock. The y-coordinate of the tip of the arrow indicates the strike price [arrows are slanted to make different introduction and expiration dates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

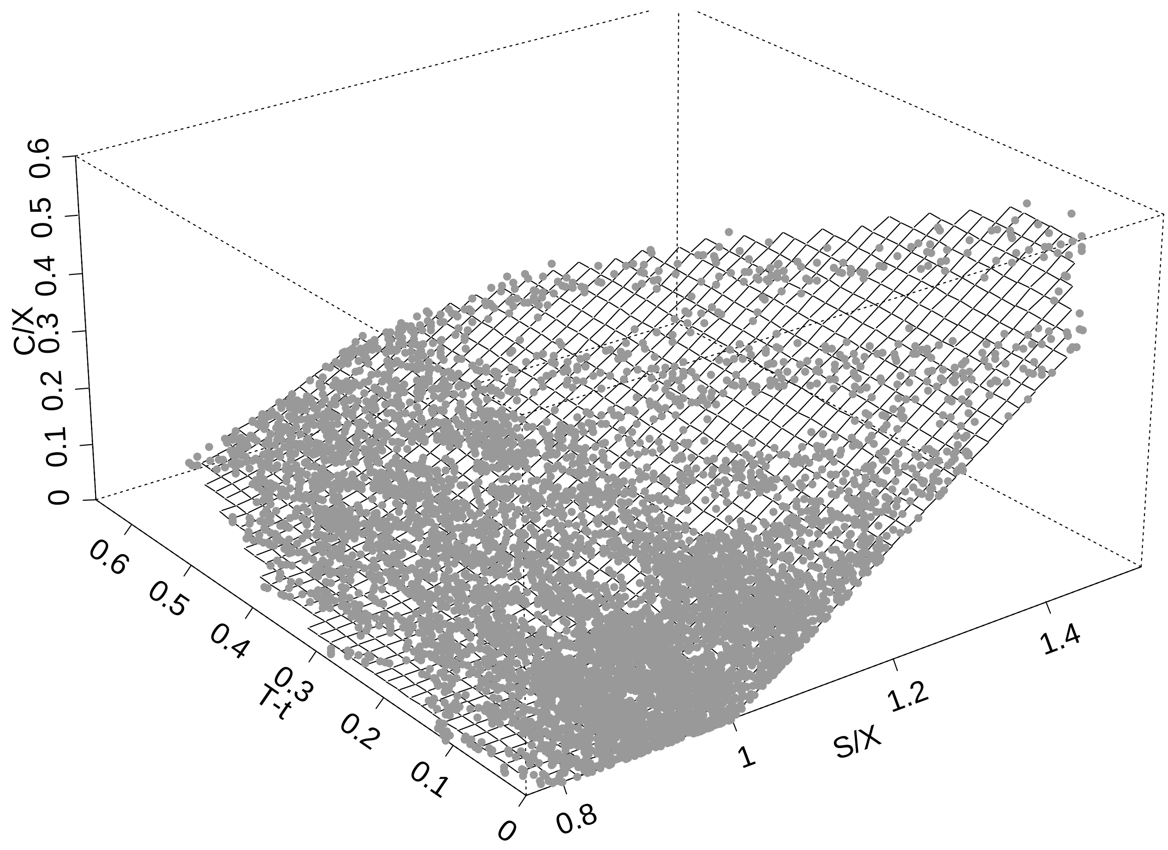
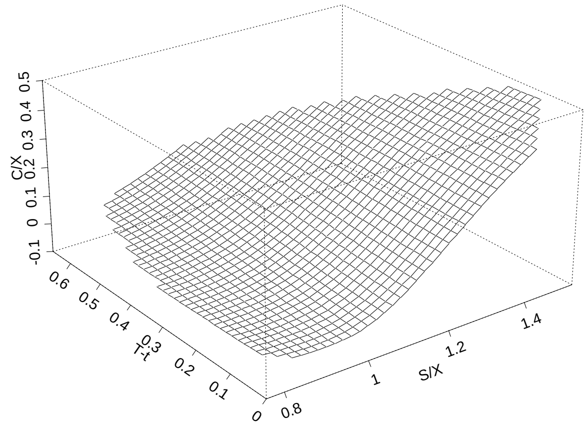
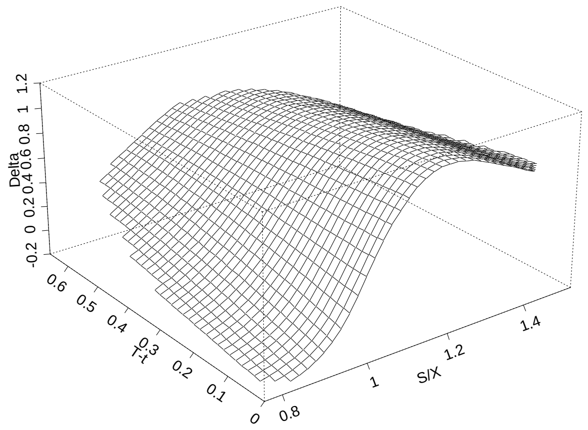


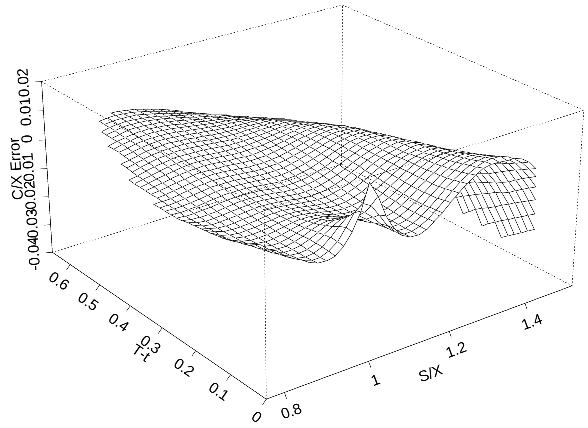
Figure 4: Simulated call option prices normalized by strike price and plotted versus stock price and time to expiration. Points represent daily observations. Note the denser sampling of points close to expiration is due to the CBOE strategy of always having options which expire in the current and next month.



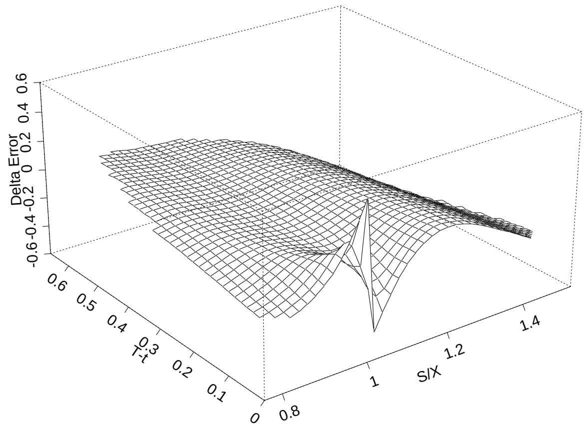
(a) Network call price $\widehat{C/X}$



(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 5: Typical behavior of 4 nonlinear term RBF model.

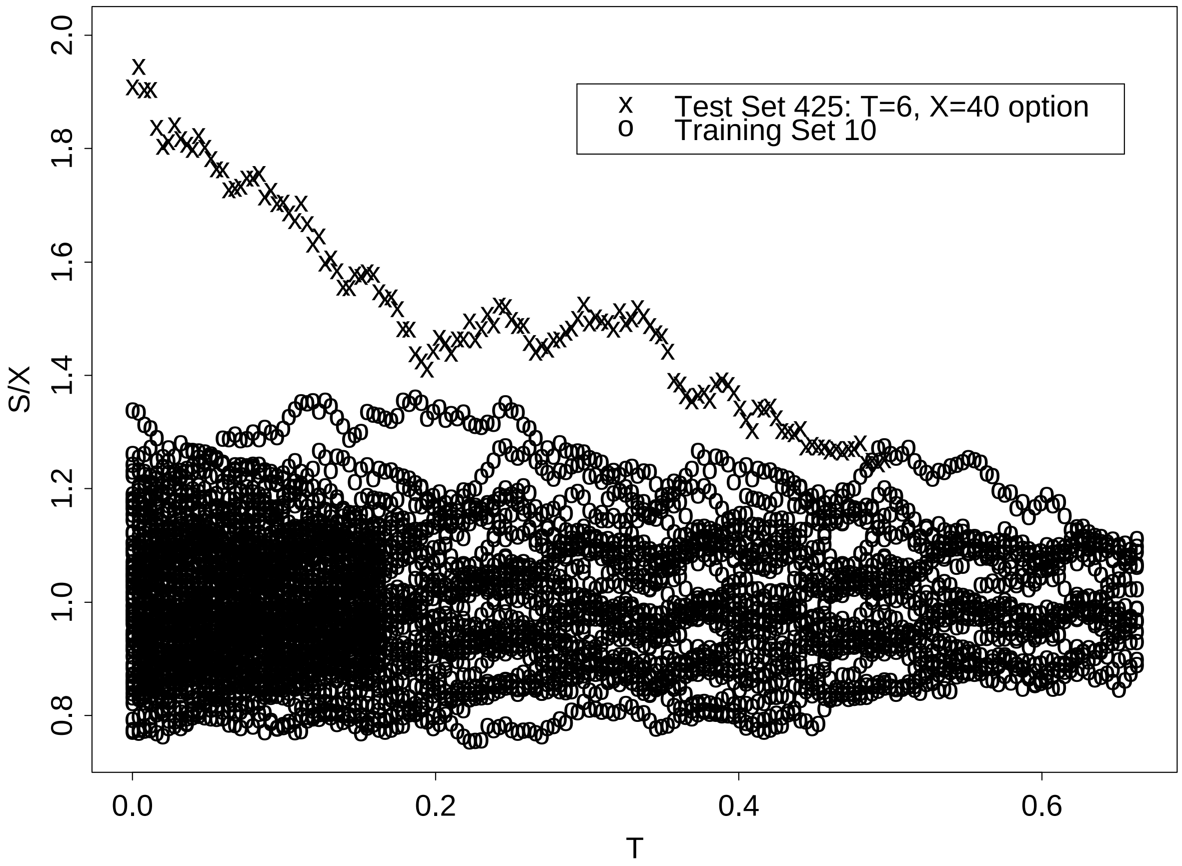


Figure 6: Input points in the training set and test set for the RBF network with the largest error measure $\hat{\xi}$.

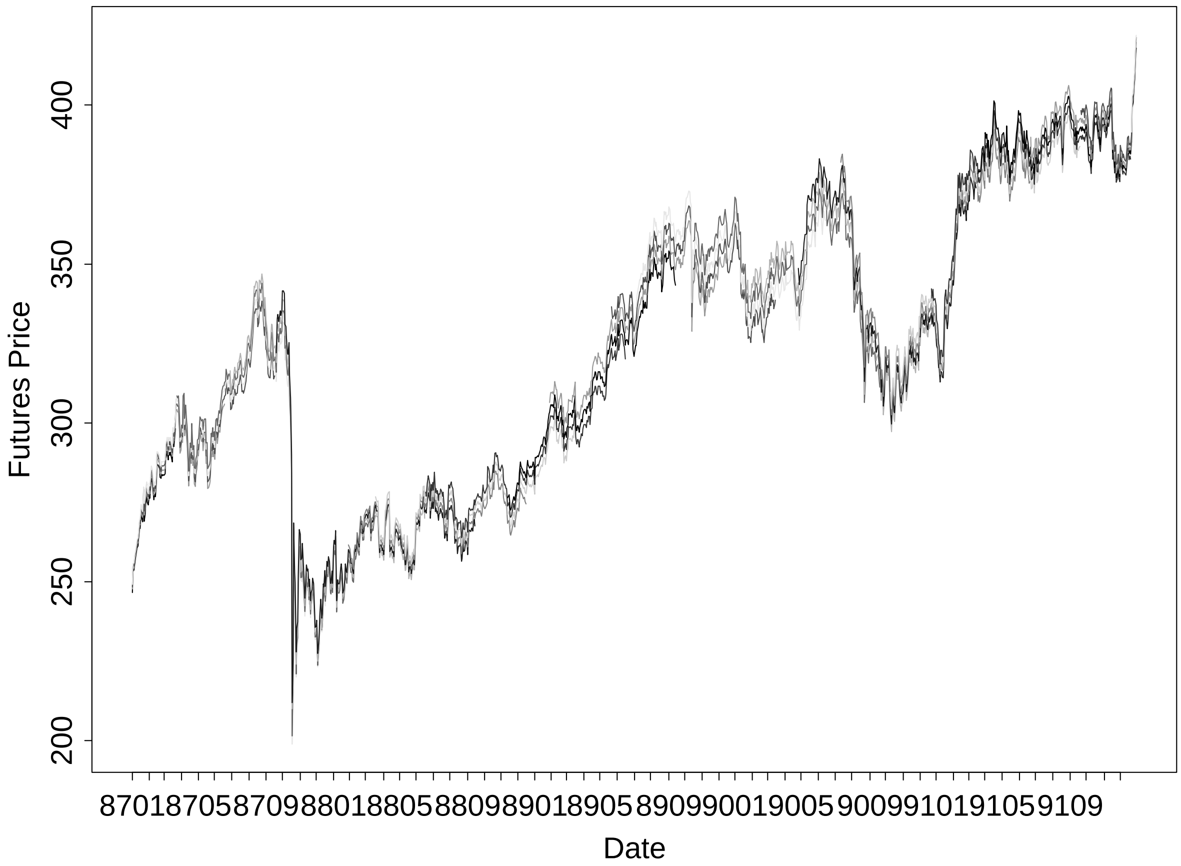


Figure 7: Overlay of S&P 500 futures prices for all contracts active from January 1987 to December 1991.

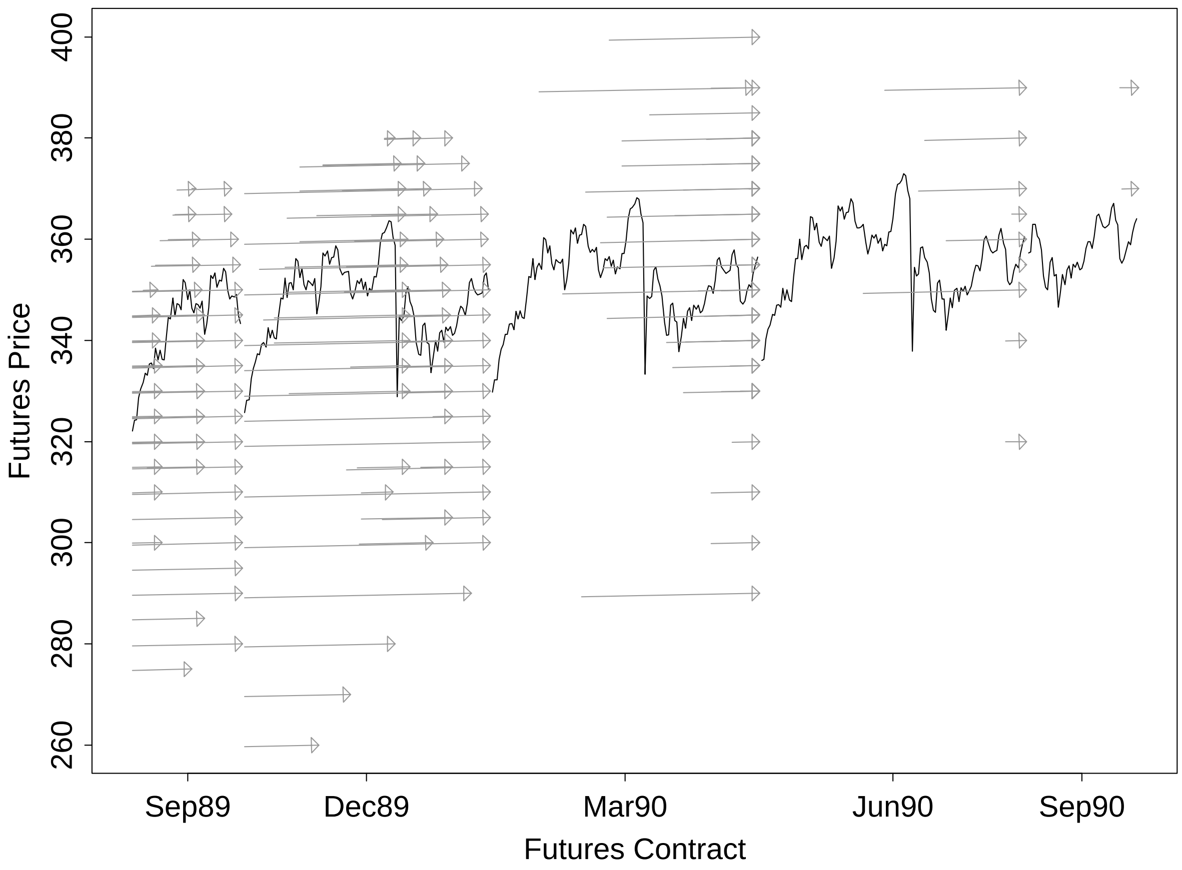


Figure 8: S&P 500 futures and futures options active from July thru December 1989. Dashed line represents futures price, while the arrows represent the options on the future. The y-coordinate of the tip of the arrow indicates the strike price [arrows are slanted to make different introduction and expiration dates visible].

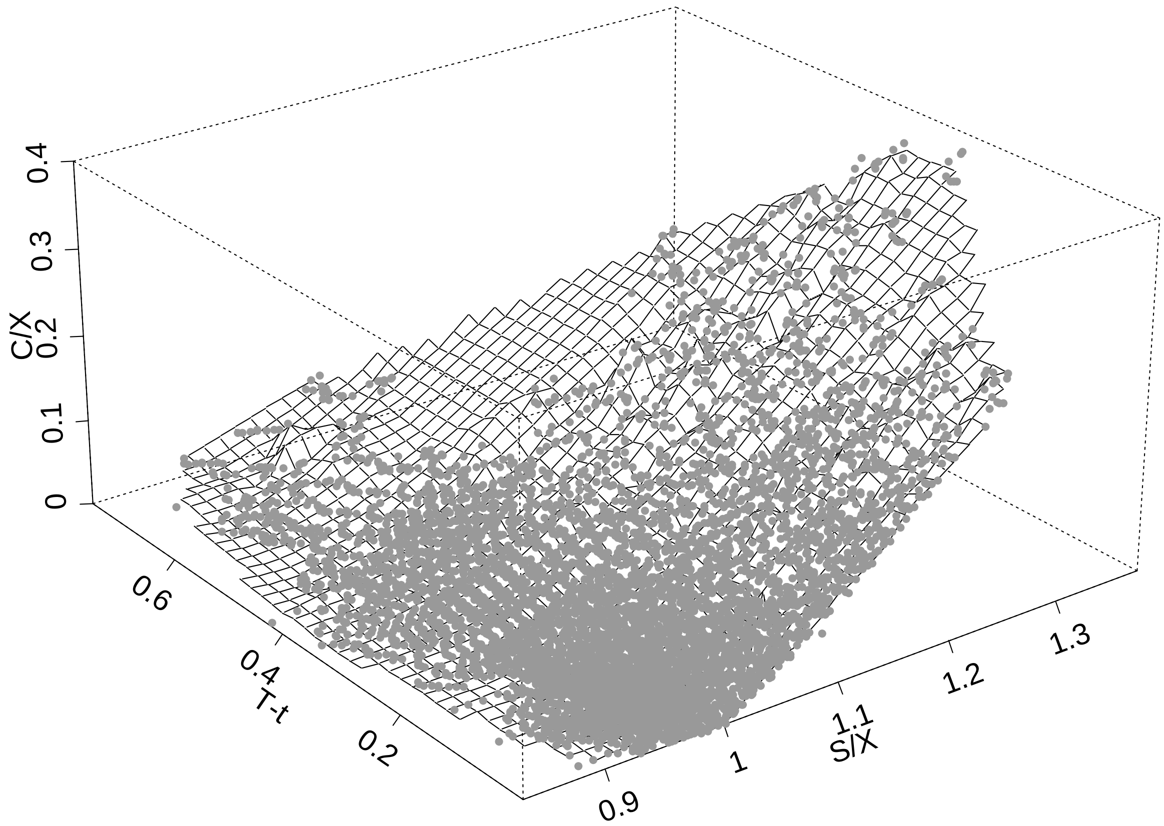
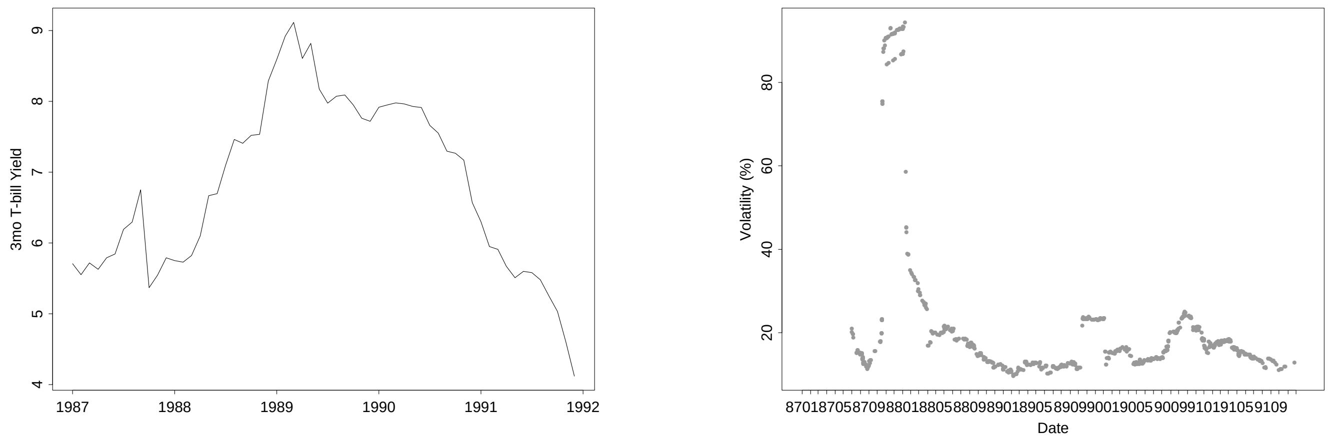


Figure 9: July thru December 1989 S&P 500 futures call option prices, normalized by strike price and plotted versus stock price and time to expiration. Points represent daily observations. Note the bumpiness of the surface, and the irregular sampling away from the money.



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

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

Figure 10: Black-Scholes parameters estimated from S&P 500 data (see text for details). Values for $\hat{\sigma}$ fall between 9.63% and 94.39%, with a median of 16.49%.