ANNUAL REVIEWS

Annual Review of Statistics and Its Application

Statistical Machine Learning for Quantitative Finance

M. Ludkovski

Department of Statistics and Applied Probability, University of California, Santa Barbara, California, USA; email: ludkovski@pstat.ucsb.edu

Annu. Rev. Stat. Appl. 2023. 10:271–95

First published as a Review in Advance on November 22, 2022

The *Annual Review of Statistics and Its Application* is online at statistics.annualreviews.org

https://doi.org/10.1146/annurev-statistics-032921-042409

Keywords

surrogates, machine learning in finance, parametric option pricing

Abstract

We survey the active interface of statistical learning methods and quantitative finance models. Our focus is on the use of statistical surrogates, also known as functional approximators, for learning input–output relationships relevant for financial tasks. Given the disparate terminology used among statisticians and financial mathematicians, we begin by reviewing the main ingredients of surrogate construction and the motivating financial tasks. We then summarize the major surrogate types, including (deep) neural networks, Gaussian processes, gradient boosting machines, smoothing splines, and Chebyshev polynomials. The second half of the article dives deeper into the major applications of statistical learning in finance, covering (*a*) parametric option pricing, (*b*) learning the implied/local volatility surface, (*c*) learning option sensitivities, (*d*) American option pricing, and (*e*) model calibration. We also briefly detail statistical learning for stochastic control and reinforcement learning, two areas of research exploding in popularity in quantitative finance.

1. INTRODUCTION

Quantitative finance (QF) is concerned with frameworks for managing trading activities in financial markets. The cornerstone of modern QF consists of stochastic models that aim to capture the random phenomena pervasive in markets. A typical QF model specifies stochastic dynamics for the system of interest (e.g., a particular stock share price) and then poses particular tasks for that system (e.g., hedging a financial contract). At an abstract level, the model defines a data-generating process, and the modeler manipulates quantities linked to that process. This perspective highlights the manifold links between QF and the design and analysis of computer experiments (DACE). Like in DACE, the fundamental modeling goal in QF is to capture observed market features and structural properties and achieve the maximum degree of realism. At the same time, the fundamental application of financial models is to analyze their output in an accurate and efficient manner. The above tension between model complexity and tractability has grown as the QF field has matured and has accelerated in the past decade as new computational paradigms have vastly expanded the envelope of what is feasible. Compared with even the early 2000s, when compact parametric models were de rigueur, practicing QF researchers, or quants, now routinely operate with large nonparametric frameworks that are coupled with advanced machine learning–flavored numerical techniques. As a result, the DACE lens is essential for decoupling the modeling complexity from computational demands. The glue connecting such expensive experiments with cheap numerical proxies is statistical machine learning.

This review surveys the current landscape of statistical learning in QF. For a broader overview, readers are referred to the recent monograph by Dixon et al. (2020), the upcoming edited volume by Capponi & Lehalle (2022), and the deep learning–oriented surveys by Ruf & Wang (2020), Charpentier et al. (2021), and Hambly et al. (2021).

The main object in statistical learning is the surrogate (Gramacy 2020), which is an empirically trained functional approximator. More precisely, a surrogate is a function $\widehat{f} : \mathcal{X} \to \mathbb{R}^p$ mapping inputs $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ into outputs, normally scalars ($p = 1$). The surrogate is constructed from a training set $\mathcal{D} = (\mathbf{x}^{1:N}, y^{1:N})$, lifting it into a model describing the relations between $\mathbf{x}$'s and y's. This is done by finding a function $\widehat{f}(\cdot)$ that minimizes a loss functional $\mathcal{R}(f; \mathcal{D})$. The loss is constructed from a metric $\mathcal{L}(\hat{y}, y)$ that defines the distance between surrogate predictions and the given data, describing the cost of incorrectly predicting outputs. In the idealized world, the loss is then the expected value based on the random variables (X, Y) that generate the independent and identically distributed (i.i.d.) data in $\mathcal{D}$,

$$\bar{\mathcal{R}}(f) := \mathbb{E}[\mathcal{L}(f(X), Y)].$$

Since the joint distribution of (X, Y) cannot be ascertained, we approximate $\bar{\mathcal{R}}$ with the empirical loss $\mathcal{R}(\widehat{f}; \mathcal{D}) := \frac{1}{N}\sum_i \mathcal{L}(\widehat{f}(\mathbf{x}^i), y^i)$.

1.1. Motivating Setups

In this article, a surrogate (also called an emulator or metamodel) refers to a broad class of statistical models that enable faster computation while not constrained to a fixed family of functions. Examples include neural networks (NNs), Gaussian processes (GPs), gradient boosting (GB) machines, random forests, support vector machines, and splines (Ruppert 2004, James et al. 2013). Such nonparametric techniques entail approximation classes that are data-driven and hence vary as training data sets change.

To motivate analysis of input–output relationships, consider a financial option written on an underlying asset with value S_t and price $\Pi(t, S_t)$ at time t. The modeler has a training set $\mathcal{D} = \{(t^i, S^i, y^i) : i = 1, \ldots, N\}$, where $y^i \simeq \Pi(t^i, S^i)$, and the goal is to infer the pricing functional $(t, S) \mapsto \Pi(t, S)$ for generic (t, S). This setting covers several subcases.

1. Computing $\Pi(t, S)$ exactly for a given (t, S) is possible, but is challenging and time-consuming. For example, it might necessitate solving a partial differential equation (PDE) or applying a fast Fourier transform. Then $\mathcal{D}$ is a collection of inputs where $y^i = \Pi(t^i, S^i)$ was evaluated exactly, and the goal is to obtain a cheaper representation of the pricing map by extrapolating these y^i values.
2. Option prices are evaluated through a Monte Carlo engine. For a given (t^i, S^i), the modeler has access to a noisy estimate of $\Pi(t^i, S^i)$, namely an empirical average Y^i of $\check{N}$ samples, with precision being $\mathcal{O}(\check{N}^{-1/2})$:

$$Y^i := \frac{1}{\check{N}} \sum_{j=1}^{\check{N}} Y^i_j, \qquad \mathbb{E}[Y^i_j] = \Pi(t^i, S^i), \quad \text{indep.} \qquad 1.$$

   Here we again aim to facilitate model-based computations, replacing the computationally intensive simulations in Equation 1 with a statistical approach that learns the pricing map from a few examples.
3. $\mathcal{D}$ are actual observations, harvested from available market data. The surrogate implements the concept of data-driven modeling by keeping explicitly parameterized frameworks in the background and focusing on efficient statistical representations that closely match observed inputs and outputs. For example, one wishes to leverage vast quantities of historical stock trajectories to minimize risk via data-driven hedging ratios.

From the probabilistic perspective, all the above setups reduce to the evaluation of (conditional) expectations, that is, functionally learning $\mathbf{x} \mapsto \mathbb{E}[g(X_t)|X_0 = \mathbf{x}]$, where X represents the state of the financial system of interest (S being one of its coordinates) and $g(\cdot)$ a payoff. Indeed, this is the canonical setting for statistical learning, interpreted as predicting the mean response from a data-generating process. Conditional expectations not only are central to option pricing as discussed in Section 3 but also underpin the stochastic control applications detailed in Sections 6 and 7.

1.2. Surrogate as a Tool

By construction, QF surrogates are not a goal unto themselves but a building block in a bigger setup. Those higher tasks are tool-agnostic; that is, there is no intrinsic reason to prefer one surrogate class to another. As such, the modeler should be willing to experiment with multiple tools and focus on the methodology, abstracting implementation details.

The simplest operation on a fitted surrogate $\widehat{f}$ is prediction, that is, evaluation of $\widehat{f}(\mathbf{x}_*)$ at some test location(s) $\mathbf{x}_*$. The test locations can be in sample ($\mathbf{x}_*$ matches one of the training $\mathbf{x}^i$) or out of sample. In the latter case, one often conceptually distinguishes between interpolation (where $\mathbf{x}_*$ is within the range of the training inputs) and extrapolation. Learning theory implies that extrapolation errors are generally much larger and less controllable than interpolation errors; therefore, extrapolation is to be avoided if at all possible. In some cases, prediction is needed over massive test sets, and there are synergies from vectorizing the prediction computation.

Beyond prediction, many other surrogate operations, some of them rather complex, arise in QF problems. In Section 4 we consider learning the gradients of $f(\cdot)$; in Section 5 we discuss optimizing over $f(\cdot)$ or inverting it. In tail risk measurement (Risk & Ludkovski 2018), the goal is to integrate (compute conditional value at risk) $f(\cdot)$ over an adaptive domain. In stochastic games, the goal is to find a fixed point of $f(\cdot)$.

By decoupling the training step from prediction, the surrogate acts as a storage container for the information obtained during fitting. This is an essential selling point for statistical learning since it offers the opportunity to invest in offline training separately from real-time prediction, matching the separation of front and back offices. It is also the critical distinction from the standard probabilistic task of approximating a given quantity. To wit, given an analytically intractable expectation, the classical Monte Carlo method constructs an approximation that is evaluated on the spot to any desired level of accuracy. In contrast, a surrogate model first builds a general purpose approximator; later, it is evaluated at yet-unknown inputs.

1.3. Statistical Learning for Quantitative Finance Experts

Let us review some of the terminology relevant for statistical learning. Statistical learning is concerned with inferring an input–output relationship. We distinguish between inputs (also known as designs or features) $\mathbf{x}$, true (latent) responses $f(\mathbf{x})$, and observations $y(\mathbf{x})$. The outputs y can either be exact samples of $f(\cdot)$ (the noise-free setup) or be stochastically sampled. In the latter case, noise is usually assumed to be additive, stationary (in $\mathbf{x}$), and Gaussian, $y(\mathbf{x}) = f(\mathbf{x}) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_Y^2)$ i.i.d., although most algorithms can generalize (modulo computational challenges) to nonadditive, non-Gaussian state-dependent noise. Thus, statistical learning subsumes both interpolation of exact samples and smoothing of noisy data and extrapolation.

The interpretation of the input space $\mathbf{x} \in \mathcal{X}$ is context-specific; an input $\mathbf{x}$ can include both stochastic quantities (i.e., values of a stochastic process) and model or market parameters, such as option properties (strike, maturity, etc.), volatility, or interest rates. Thus, $\mathbf{x}$ is almost always multidimensional. Picking precisely what $\mathbf{x}$ does and does not contain is frequently a key modeling choice. The overall training set $\mathcal{D}$ may be fixed and given (an external data set) or be generated by the modeler themselves. In the latter framework, constructing $\mathcal{D}$ is known as experimental design.

To fit a surrogate, one selects an approximation space $\mathcal{H}$ and a loss function $\mathcal{L} \equiv \mathcal{L}(\hat{y}^{1:N}, y^{1:N})$, where $\hat{y}^i$ is the surrogate prediction at $\mathbf{x}^i$. The most common choice is mean-squared error, $\mathcal{L}_{\mathrm{MSE}} := \frac{1}{N}\sum_{i=1}^{N}(\hat{y}^i - y^i)^2$, which matches the probabilistic structure of conditional expectations that are defined as L^2 minimizers. Further criteria are $\mathcal{L}_{\mathrm{MAE}} = \frac{1}{N}\sum_i |\hat{y}^i - y^i|$, $\mathcal{L}_{\mathrm{MAPE}} = \frac{1}{N}\sum_i \frac{|\hat{y}^i - y^i|}{y^i}$, $\mathcal{L}_{\mathrm{MAX}} = \max_{1\le i\le N} |\hat{y}^i - y^i|$, and $\mathcal{L}_{R^2} = 1 - \frac{\sum_i(\hat{y}^i - y^i)^2}{\sum_i(\bar{y} - y^i)^2}$. The surrogate $\widehat{f}$ is then taken to be the empirical minimizer of $\mathcal{L}(f(\mathbf{x}^{1:N}), y^{1:N}) = \mathcal{R}(f; \mathcal{D})$,

$$\widehat{f} = \arg\min_{f \in \mathcal{H}} \mathcal{L}\left(f(\mathbf{x}^{1:N}), y^{1:N}\right). \tag{2}$$

Practically speaking, the surrogate is defined through its (hyper)parameters ϑ so that the fitting step can be actualized as an optimization problem for finding $\hat{\vartheta}$ that minimizes $\mathcal{R}(f; \mathcal{D})$. This is generally a high-dimensional, nonconvex optimization task, and global solutions are not guaranteed. Indeed, the performance of various surrogate frameworks is often predicated on how well the latter optimization can be carried out. A common approach is to apply stochastic gradient descent, generating a sequence of $\vartheta^{(j)}$'s over training epochs $j = 0, 1, \ldots$ that involve minibatches (i.e., subsets) of $\mathcal{D}$.

The optimization in Equation 2 may be regularized by adding additional terms. Regularization is applied to mitigate overfitting; it can also include penalty terms to give preference to surrogates that satisfy additional soft constraints. Financial domain knowledge can be embedded through context-specific loss metrics $\mathcal{L}(\cdot)$ or shape-constrained surrogates (Dugas et al. 2000, Yang et al. 2017, Huh 2019, Chataigner et al. 2021, Zheng et al. 2021).

To assess the quality of $\widehat{f}$, one takes a test set $\mathcal{C}$ and a fitness metric (often different from the loss metric) $\mathcal{W}(\widehat{f}, \mathcal{C})$. For surrogate assessment, the relation between the test and the training sets $\mathcal{D}$ and $\mathcal{C}$ is essential. Generalization bias—overestimating model performance due to looking at in-sample results—is a well-known concern. Indeed, without testing on new, unseen inputs, one is not able to detect overfitting. Similarly, model performance might be misleading if there is any unfair advantage in preselecting the test set. For example, when working with real data sets, Ruf & Wang

(2020) point out the potential for data snooping that arises when randomly partitioning between training and testing sets. Instead, chronological partitioning should be considered; this is also necessary since there is a lot of time inhomogeneity in financial data, such as varying volatility regimes.

In many financial contexts data are collected continuously (e.g., by scraping market quotes every day), known as streaming or online learning. To make such learning efficient, one desires surrogates that can be updated rather than fully refit every time more training data arrive. Updating is possible through warm-starting the underlying optimizers [i.e., judiciously initializing $\vartheta^{(0)}$] and sometimes via explicit updating equations. The case where the streamed training data are picked by the modeler is called sequential design and brings forth the topic of the value of information from collecting additional samples.

1.3.1. Geometry of the training set. While training inputs are sometimes gridded, most surrogates operate with mesh-free experimental designs. The classical QF learning algorithms tend to generate inputs as samples along simulated paths of a stochastic process. This means that the experimental design has a sampling density $\mathbf{x} \sim p(\mathbf{x})$. Alternatives include sampling $\mathbf{x}$ using Latin hypercube designs or low-discrepancy quasi–Monte Carlo sequences (Lemieux 2009). Stratification or importance sampling (Glasserman 2004) may also come into view. It is important to remember that surrogate predictions can only be trusted to approximate well over the training domain. It is therefore prudent to monitor the use of the surrogate and retrain if the test domain $\mathcal{C}$ shifts relative to the range of $\mathcal{D}$, known as concept drift.

1.3.2. Convergence. Statistical learning paradigms afford extensive convergence theories regarding the approximation quality (Györfi et al. 2002). The first fundamental type of results concerns consistency, that is, the approximation error vanishing in a certain limit, typically as the sample size goes to infinity (Elie et al. 2020, Cheridito & Gersey 2021). This might be possible only for true responses $f(\cdot)$ belonging to some given class of functions or universally (e.g., for any twice-differentiable function). The second type of results is about asymptotic convergence rates (Belomestny 2011b, Glau & Mahlstedt 2019, Gonon & Schwab 2021). The third type is about convergence in terms of surrogate hyperparameters ϑ, keeping the data-generating process fixed. These results concern the question of using more basis functions (or more neurons in NNs), and often require taking joint limits in $|\mathcal{D}|$ and $|\vartheta|$ (Clément et al. 2002, Glasserman & Yu 2004a, Belomestny et al. 2010a). Finally, for some learning frameworks it is possible to obtain probabilistic guarantees on their quality, that is, to do a self-assessment that provides localized error bounds on the surrogate prediction. This uncertainty quantification is essential for adaptive learning and, moreover, allows the modeler to reject surrogate results if its self-reported quality is insufficient. Such guarantees are increasingly important for financial risk managers and for assessing model risk. A further common challenge is that many QF problems generate a sequence of interdependent surrogates, motivating analysis of error propagation from one surrogate to another.

1.4. Quantitative Finance Applications for Machine Learning Experts

In this section, we summarize the major domains of QF where statistical learning has been applied. These problems are then examined in more detail in Sections 3–6.

Let us first review one example of the basic option pricing problem. In the Black–Scholes model, a stock with value S_t at time t is modeled as a geometric Brownian motion, that is, a continuous-time stochastic process with dynamics given by the linear stochastic differential equation driven by a Brownian motion (W_t),

$$\mathrm{d}S_t = rS_t\,\mathrm{d}t + \sigma S_t\,\mathrm{d}W_t, \qquad 3.$$

parameterized by the constant interest rate r and constant volatility σ. This implies that $S_T|S_t$ is log normal. We wish to evaluate the price of a financial derivative on (S_t), that is, a contract that entitles its owner to a payoff that depends on S. Given a payoff $g(S_T)$, no-arbitrage pricing theory implies that a fair price of this contract at date t is the expected discounted payoff conditional on information available at time t,

$$\mathbb{E}\left[e^{-r(T-t)}g(S_T)\middle|\, S_t\right]. \tag{4.}$$

For a call payoff $g_{\text{Call}}(S) := (S-K)_+$, exact integration is possible, yielding the Black–Scholes formula for $\Pi(t,S) = \mathbb{E}\left[e^{-r(T-t)}(S_T-K)_+|S_t=S\right]$, which is a function of the five parameters $(S, K, r, \sigma, T-t)$. Note that the drift being r in Equation 3 reflects the fact that the dynamics of (S_t) are already stated under the risk-neutral $\mathbb{Q}$-measure that is the relevant one for contingent claim valuation, skipping the issue of risk premia common in econometrics.

In general, Equation 4 does not admit closed-form solutions and is best thought of as an input–output map, with inputs being the contract and model parameters θ and output being the option price. To set the mathematical framework, let (X_t) denote the stochastic process summarizing relevant financial quantities and taking values in $\mathcal{X} \subseteq \mathbb{R}^d$. We assume the standard probabilistic structure of $(\Omega, \mathcal{F}, (\mathcal{F}_t), \mathbb{P})$, where X_t is adapted to the filtration $(\mathcal{F}_t)$. In the most common situation, one considers European-style contracts, where the payoff at maturity date T is a functional, $g(X_T) \in \mathbb{R}$, $g \in L^2(\mathbb{P})$. The no-arbitrage price at date t is $\mathbb{E}^{\mathbb{Q}}\left[e^{-\int_t^T r_s ds}g(X_T)\middle|\,\mathcal{F}_t\right]$, where (r_t) is the risk-free interest rate process, $\mathbb{Q}$ is the pricing measure, and the σ-algebra $\mathcal{F}_t$ summarizes the available information up to $t \in [0, T]$. When (X_t) is Markov, the expectation is a function of X_t,

$$\Pi(t,\mathbf{x}) := \mathbb{E}^{\mathbb{Q}}\left[e^{-\int_t^T r_s ds}g(X_T)\middle|\, X_t = \mathbf{x}\right]. \tag{5.}$$

This is the case for the instance when (X_t) satisfies a stochastic differential equation,

$$\mathrm{d}X_t = \mu(X_t)\,\mathrm{d}t + \sigma(X_t)\,\mathrm{d}W_t^Q, \tag{6.}$$

where (W_t^Q) is a (multidimensional) Brownian motion.

Modern QF models feature either more sophisticated payoffs (for example, path-dependent ones that depend on $X_{[t,T]}$ and not just on X_T) or more sophisticated dynamics than in Equation 3, precluding analytic expressions for $\Pi(t,\mathbf{x})$. For example, the Heston stochastic volatility model has two-dimensional $X_t = (S_t, v_t)$ with five parameters $\theta = (r, \kappa, \eta, \sigma, \rho)$ and dynamics

$$\begin{cases} \mathrm{d}S_t = rS_t\,\mathrm{d}t + \sqrt{v_t}S_t\,\mathrm{d}W_t^1; \\ \mathrm{d}v_t = \kappa(\eta - v_t)\,\mathrm{d}t + \sigma\sqrt{v_t}\mathrm{d}W_t^2, \qquad \mathrm{d}\langle W_t^1, W_t^2\rangle = \rho\mathrm{d}t. \end{cases} \tag{7.}$$

Respective evaluation of a single call and put price requires solving either a PDE or a Monte Carlo simulation in order to integrate $g(\cdot)$ against the bivariate conditional distribution of (S_t, v_t) (Glasserman 2004); quants seek the broader surrogate $(t, S, v, \theta) \mapsto \widehat{\Pi}(t, S, v, \theta)$.

1.4.1. Option hedging. A hedging strategy is a sequence of functions $h(k,\cdot) : \mathbb{R}^d \to \mathbb{R}$ such that $h(k,\mathbf{x})$ specifies holdings of the asset S in period t_k given overall state $\mathbf{x}$. The strategy is assumed to be self-financing; that is, cash is dynamically moved in or out of a riskless savings account to account for intermediate profits and losses without additional cash flows. The resulting discounted wealth at T is the discrete stochastic integral

$$V_T := (h\cdot S)_T = V_0 + \sum_{k=0}^{K_T-1} h(k, X_{t_k})[S(X_{t_{k+1}}) - S(X_{t_k})] - c(h(k, X_{t_k})), \tag{8.}$$

where $c(\cdot)$ captures transaction costs. The goal of hedging is to minimize the expected hedging error $\mathbb{E}[\mathcal{L}(e^{-rT}g(X_T), V_T)]$ relative to the discounted option payoff $e^{-rT}g(X_T)$. As usual, this is evaluated using a batch of N samples of $X_{0:K_T}$, for example, mean trading error $\mathcal{W}_{\mathrm{MTE}}(g(\mathbf{x}_T^{1:N}), v_T^{1:N}) := \frac{1}{N}\sum_n\{e^{-rT}g(\mathbf{x}_T^n) - v_T^n\}$ and median absolute trading error $\mathcal{W}_{\mathrm{MATE}}(g(\mathbf{x}_T^{1:N}), v_T^{1:N}) = \frac{1}{N}\sum_n |e^{-rT}g(\mathbf{x}_T^n) - v_T^n|$. As in option pricing, one is interested in surrogates $\widehat{h}(k,\cdot)$ for the hedging ratios. Observe how the above fitness criteria are highly implicit with respect to $\widehat{h}(k,\cdot)$. Training can be model-free, using historical paths of (S_t).

Bermudan option: American-style contract that can be exercised at a discrete set of prespecified time instances

1.4.2. American options.

In American-style contracts, the buyer can collect her payoff at any time $\tau \le T$ prior to maturity. This choice is made dynamically as market conditions unfold. The pricing problem for an American option with payoff $g(t, \mathbf{x})$ (we notationally subsume discounting) is known to reduce to finding the decision rule τ, namely a stopping time to maximize expected reward g,

$$\mathbb{E}^{\mathbb{Q}}\left[g(\tau, X_\tau)\right] \to \max! \qquad 9.$$

Conditioning on $X_t = \mathbf{x}$, the maximization is over the collection $\mathfrak{S}_t$ of $(\mathcal{F}_t)$-stopping times bigger than t and less than or equal to T. For computational purposes, one restricts to a discrete-time paradigm, where decisions are made at K_T prespecified instances $t_0 = 0 < t_1 < \cdots < t_k < \cdots < t_{K_T} = T$, where $t_k = k\Delta t$ for a given discretization step Δt. Pricing of the resulting Bermudan options via statistical learning is discussed in Section 6.

2. COMMON SURROGATE TYPES

The simplest surrogate is a linear model that uses the approximation space $\mathcal{H} = \mathrm{span}(B_1, \ldots, B_M)$, that is, linear combinations of the basis functions $B_m(\cdot)$, translating Equation 2 into a finite-dimensional optimization problem in terms of the $M+1$ respective coefficients $\vec{\vartheta}$, with $\widehat{f}(\mathbf{x}) = \vartheta_0 + \sum_{m=1}^{M}\vartheta_m B_m(\mathbf{x})$. Targeted or adaptive selection of the B_m's is common in QF problems. An important case of the above is natural cubic splines—functions that are piecewise cubic polynomial and are twice continuously differentiable (i.e., satisfy zeroth-, first- and second-order smoothness constraints at the respective knot sites). Cubic splines with M knots have $4M+2$ hyperparameters $\vec{\vartheta}$; M is often selected adaptively.

2.1. Neural Networks

A feed-forward NN is a composition of linear and nonlinear functions, arranged in layers. Each layer contains an affine function that combines inputs and then a nonlinear activation function to pass these on to the next layer. A feed-forward NN surrogate with L layers is a function $\mathcal{N}(x) : \mathbb{R}^d \mapsto \mathbb{R}^p$ of the form

$$\mathcal{N}(\mathbf{x}) = H_L(\phi_L(H_{L-1}(\ldots(\phi_1(H_1(\mathbf{x}))\ldots)))),$$

where each $H_\ell : \mathbb{R}^{n_{\ell-1}} \mapsto \mathbb{R}^{n_\ell}$ is an affine map $H_\ell(\mathbf{x}) = W_\ell \cdot \mathbf{x} + w_\ell$, and ϕ_ℓ's are the nonlinear activation functions applied coordinate-wise. Common activation functions are the rectified linear unit (ReLU) $\phi(x) = \max(x, 0)$ and exponential linear unit (ELU) $\phi(x) = \max(e^x - 1, x)$. The parameter n_ℓ is the number of nodes in layer $\ell = 1, \ldots, L$. Training an NN means learning the matrices $\widehat{W}_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and the vectors $\widehat{w}_\ell \in \mathbb{R}^{n_\ell}$ such that the NN outputs $\widehat{\mathcal{N}}(\mathbf{x}^i)$ are close to the observed outputs y^i. The total number of the NN hyperparameters ϑ is $|\vartheta| = \sum_{\ell=1}^{L} n_\ell \times (n_{\ell-1}+1)$ and is often in the thousands. In practice, the nonconvex optimization problem to find $\widehat{W}_\ell$ and $\widehat{w}_\ell$ is achieved via gradient descent, gradually improving weights $\vartheta^{(j)}$ as training mini-batches indexed by j are considered. The weights are randomly initialized with samples $\vartheta^{(0)}$ drawn from either the uniform or Gaussian distributions. One common procedure with the ReLU activation function initializes the weights in layer ℓ according to $w_\ell^{(0)} \sim \mathcal{N}(0, 2/n_\ell)$ (Ferguson & Green 2018). To enable such generic initialization all inputs (and outputs) are prescaled into the $[0, 1]^d$ hypercube.

Several features of NNs are useful for QF contexts. First, the universal approximation property asserts that NNs are able to approximate continuous functions arbitrarily well; that is, it guarantees sufficient flexibility given NNs with enough parameters. Second, NNs support extensive offline training together with fast prediction, resolving many of the scalability limitations of other surrogates. Third, the NN minibatches work well with streamed data.

2.2. Gaussian Processes

GP regression is a flexible nonparametric regression method (Rasmussen & Williams 2006) that views the map $\mathbf{x} \to f(\mathbf{x})$ as a realization of a Gaussian random field so that [in the abstract metamodel probability space, independent of the probabilistic structure present in (X_t)] any finite collection of $\{f(\mathbf{x}), \mathbf{x} \in \mathcal{X}\}$ is multivariate Gaussian. For any $n \geq 1$ design sites $\{\mathbf{x}^i\}_{i=1}^n$, GP regression posits that $(f(\mathbf{x}^1), \dots, f(\mathbf{x}^n)) \sim \mathcal{N}(\mathbf{m}_n, \mathbf{K}_n)$, with mean vector $\mathbf{m}_n := [m(\mathbf{x}^1; \vartheta), \dots, m(\mathbf{x}^n; \vartheta)]$ and $n \times n$ covariance matrix $\mathbf{K}_n$ composed of $\kappa(\mathbf{x}^i, \mathbf{x}^{i'}; \vartheta)$ for $1 \leq i, i' \leq n$. The role of $m(\cdot)$ is to capture the known trends in the response, and the role of the positive semidefinite kernel function $\kappa(\cdot,\cdot)$ is to capture the spatial dependence structure in $\mathbf{x}$, generalizing the notion of a covariance matrix. Thus, GPs place a probabilistic prior directly on the space of functions, without imposing a parametric structure on $\widehat{f}$.

Given the training data set $\mathcal{D} = \{\mathbf{x}^i, y^i\}_{i=1}^N$, GP regression infers the posterior of $f(\cdot)$ by assuming an observation model $y(\mathbf{x}) = f(\mathbf{x}) + \epsilon$ with a Gaussian noise term $\epsilon(\mathbf{x}) \sim \mathcal{N}(0, \sigma_Y^2)$. Conditioning equations for multivariate normal vectors imply that the posterior $f(\mathbf{x}_*)|\mathcal{D}$ at any arbitrary input $\mathbf{x}_*$ is also Gaussian:

$$m_*(\mathbf{x}_*) := m(\mathbf{x}_*) + K^T(\mathbf{K} + \sigma_\epsilon^2 \mathbf{I})^{-1}(\mathbf{y} - \mathbf{m}) = \mathbb{E}\Big[f(\mathbf{x}_*)\big|\mathcal{D}\Big], \qquad 10.$$

$$\mathbf{y} = [y^1, \dots, y^N]^T, \qquad \mathbf{m} = [m(\mathbf{x}^1; \vartheta), \dots, m(\mathbf{x}^N; \vartheta)]^T,$$

$$K^T = [\kappa(\mathbf{x}_*, \mathbf{x}^1; \vartheta), \dots, \kappa(\mathbf{x}_*, \mathbf{x}^N; \vartheta)]$$

and $\mathbf{K}$ is an $N \times N$ covariance matrix described through the kernel function $\kappa(\cdot,\cdot\,; \vartheta)$.

GP fitting corresponds to selecting an appropriate function space $\mathcal{H} \equiv \mathcal{H}_\vartheta$ by optimizing the hyperparameters ϑ that drive $m(\cdot)$ and $\kappa(\cdot,\cdot)$. This is done in a hierarchical manner, first fixing a kernel family and then using maximum likelihood optimization to infer ϑ given $\mathcal{D}$. Once ϑ is chosen, the kriging equations (Equation 10) yield the surrogate output.

A popular choice for $\kappa(\cdot,\cdot)$ is the (anisotropic) squared exponential (SE) family, parameterized by the length scales $\{\ell_{\text{len},k}\}_{k=1}^d$ and the process variance σ_p^2:

$$\kappa_{\text{SE}}(\mathbf{x}, \mathbf{x}') := \sigma_p^2 \exp\Big(-\sum_{k=1}^d \frac{(x_k - x'_k)^2}{2\ell_{\text{len},k}^2}\Big). \qquad 11.$$

Other popular kernels are the Matérn-5/2 and Matérn-3/2 families (Roustant et al. 2012):

$$\kappa_{\text{M52}}(\mathbf{x}, \mathbf{x}') := \sigma_p^2 \prod_{k=1}^d \left(1 + \frac{\sqrt{5}}{\ell_{\text{len},k}}|x_k - x'_k| + \frac{5}{3\ell_{\text{len},k}^2}(x_k - x'_k)^2\right) e^{-\frac{\sqrt{5}}{\ell_{\text{len},k}}|x_k - x'_k|},$$

$$\kappa_{\text{M32}}(\mathbf{x}, \mathbf{x}') := \sigma_p^2 \prod_{k=1}^d \left(1 + \frac{\sqrt{3}}{\ell_{\text{len},k}}|x_k - x'_k|\right) e^{-\frac{\sqrt{3}}{\ell_{\text{len},k}}|x_k - x'_k|}.$$

The GP kernel $\kappa(\mathbf{x}, \mathbf{x}')$ controls the smoothness of $m_*(\cdot)$. The SE kernel (Equation 11) yields infinitely differentiable fits $m_*(\cdot)$, while a Matérn kernel of order $k + 1/2$ yields approximators that are in $\mathcal{C}^k$. Thus Matérn-3/2 surrogates are in $\mathcal{C}^1$ and Matérn-5/2 surrogates are in $\mathcal{C}^2$. Note that modulo rescaling, all the above belong to the class of radial basis functions, where $\kappa(x, x')$ is a function of $|x - x'|$ only; nonseparable kernels are also available.

Feature engineering: selecting the best subset of pricing features to feed a statistical surrogate

More advanced GP surrogates that have been considered in QF contexts include shape-constrained GPs (Cousin et al. 2016, Chataigner et al. 2021), multioutput GPs (Crépey & Dixon 2020) that can simultaneously learn multiple pricing functions (useful for portfolio valuation), and heteroskedastic GPs (Binois et al. 2018). GP kernels can also be composed via addition, multiplication, and convolution (Duvenaud 2014).

GPs intrinsically support updating via iterated conditioning and also excel in interpolating within sparse data environments where $\mathcal{D}$ is small. A further key feature of GP models is uncertainty quantification via the posterior covariance $\mathrm{Cov}(f(\mathbf{x}_*^1), f(\mathbf{x}_*^2)|\,\mathcal{D}) = \kappa(\mathbf{x}_*^1, \mathbf{x}_*^2) - K_1^T[\mathbf{K} + \sigma_\epsilon^2\mathbf{I}]^{-1}K_2$, where $K_i = [\kappa(\mathbf{x}_*^i, \mathbf{x}^1; \vartheta), \dots, \kappa(\mathbf{x}_*^i, \mathbf{x}^N; \vartheta)]$ for $i = 1, 2$. The interpretation is that $\mathbf{x}_* \mapsto m_*(\mathbf{x}_*)$ is the most likely input–output map that is consistent with the training data set $\mathcal{D}$, and $\mathrm{Var}(f(\mathbf{x}_*)|\mathcal{D})$ is the model uncertainty capturing the range of other potential input–output fits. The latter allows for internal model assessment. For example, high predictive uncertainty can allow the modeler to reject the existing surrogate prediction in favor of either retraining it or even using direct evaluation of the underlying data generator (Crépey & Dixon 2020).

2.3. Gradient Boosting

GB yields ensemble models of the form

$$f(\mathbf{x}) = \mathrm{const} + \sum_j \nu h_j(\mathbf{x}), \qquad 12.$$

where the base models $h_j(\cdot)$ are obtained sequentially and ν is the learning rate. The key concept of GB is that the base h_j's are obtained in a stagewise manner; that is, $f_j(\mathbf{x}) := h_0 + \sum_{k \le j} \nu h_k(\mathbf{x}) = f_{j-1}(\mathbf{x}) + \nu h_j(\mathbf{x})$ is learned by fitting $h_j(\cdot)$ conditional on f_{j-1} and the jth training batch of size N_j. The most common base model is a decision tree; the shrinkage parameter $\nu < 1$ is used for stability. Fitting h_j is done greedily by minimizing the loss function $\mathcal{L}(f_{j-1} + \nu h_j, \mathcal{D}) = \min_h \mathcal{L}(f_{j-1}(\mathbf{x}^{1:N_j}) + \nu h(x^{1:N_j}), y^{1:N_j})$ over base h's. In other words, the next base model is chosen to minimize the residual errors from the previous model over the new training batch analogously to a gradient descent step. Modern versions of GB, such as LightGBM (Ke et al. 2017), include multiple machine learning enhancements that stabilize the surrogate and mitigate overfitting (e.g., dropout, randomly removing some of the earlier h_k's when solving for h_j in order to lower the influence of the first few base models). Because the base models are trees, feature engineering is critical for GB in order to more easily find good leaf splits during training.

2.4. Chebyshev Interpolation

Building upon classical polynomial interpolators, Gaß et al. (2018) and Glau et al. (2020) develop tensorized Chebyshev polynomials of order $(n_1, \dots, n_d)$ that represent

$$\widehat{f}(\mathbf{x}) = \sum_{j_1=0}^{n_1} \dots \sum_{j_d=1}^{n_d} c_{j_1,\dots,j_d} T_{j_1,\dots,j_d}(\mathbf{x}), \qquad 13.$$

where $\boldsymbol{c}$ are the Fourier coefficients and $T_{j_1,\dots,j_d}(\mathbf{x}) = \prod_{i=1}^d \cos(j_i \arccos x_i)$. In order to tractably extend Equation 13 to high-dimensional problems $d \gg 5$, Glau et al. (2020) propose to first train on a small subset of the Chebyshev grid and then use a completion algorithm via a low-rank approximation. This is combined with gradually adding training sites and adaptively increasing the rank. The tensor of the coefficients $\boldsymbol{c}$ is obtained using tensor-matrix multiplication and the fast Fourier transform. Chebyshev interpolation offers extensive convergence theory.

Call option: the right but not the obligation to buy the underlying asset for $\$K$

Bermudan swaption: the right but not the obligation to enter into an interest rate swap on any of the predetermined dates t_k prior to T

3. LEARNING TO PRICE OPTIONS

Daily tasks in managing financial positions involve numerous calculations of contract prices, hedging ratios, risk measures, and value adjustments. As the underlying stochastic models have become more sophisticated, their implementation is more computationally intensive. As a result, computational gains afforded by cloud computing, parallel processing, faster processors, and better algorithms are canceled out by the ever higher number of computations to be made (e.g., more valuation adjustments to account for credit risk or funding, more accuracy, more risk sensitivities, new structured products) and by each computation taking longer. Nowadays, trading desks do millions of similar computational tasks each day.

To make these repeated calculations efficient and lightning fast, a surrogate is employed to learn the price of the option as a function of the underlying price and contract characteristics. Indeed, a derivatives valuation model is ultimately a function that maps inputs, consisting of market data and trade-specific terms, to an output representing the option value. The basic loss function is the squared distance $\mathcal{L}_{\mathrm{MSE}}$ between observed (simulated) option prices and surrogate predictions. While a simple contract such as the call option in a Black–Scholes model has a total of five inputs, more complex products such as Bermudan swaptions in Libor models have valuation functions with hundreds of inputs, involving all the properties of the underlying swap and option exercise schedule.

The most well-studied surrogates for this task are NNs. Following the early pioneering work by Hutchinson et al. (1994), the field rapidly expanded recently with Culkin & Das (2017) being the first to apply deep NNs. The recent survey by Ruf & Wang (2020) presents an exhaustive and monumental table that lists more than 150 papers that investigated NNs for option pricing. They compare extant proposals across six categories, including model setup, NN setup, and case studies considered. Complementary literature has considered GPs (De Spiegeleer et al. 2018, Crépey & Dixon 2020), Chebyshev polynomials (Olivares & Alvarez 2016, Gaß et al. 2018, Glau et al. 2020), GB (Davis et al. 2020), and cubic splines (Olivares & Alvarez 2016).

3.1. Inputs and Outputs

In order to apply statistical learning for option pricing, one must specify the respective inputs $\mathbf{x}$ and outputs y. For $\mathbf{x}$ the pricing features may include not just the relevant stochastic states such as underlying price, but also option parameters λ and model parameters θ. For calls and puts, the most common features are asset price S, time t, strike K (an example of λ), and volatility σ (an example of θ). For many models, the option pricing function is homogeneous of degree one with respect to S and K, making it common to reparameterize via their ratio, known as the moneyness S/K. This enables having a more stationary training set, reduces overfitting (Garcia & Gençay 2000), and lowers the dimension of the inputs, yielding computational efficiency. Similarly, rather than looking at time t and option maturity T separately, it is common to reparameterize the surrogate via time to maturity $\tau = T - t$. In the context of GB, Davis et al. (2020) suggest using features, such as discount factors e^{-rT}, and first-order interactions between the parameters of the stochastic model [e.g., $\kappa \cdot \eta$ in the Heston model (Equation 7)]. Given its prominence, the volatility σ is often used as a feature. In local or stochastic volatility models, one often includes the Black–Scholes implied volatility σ_{Imp}. The most common surrogate output y is the option price; if working with moneyness features, the output could also be the option price divided by its strike Π/K.

Among test cases, existing literature usually sticks to the most liquid contracts, such as on the S&P 500 and DAX indices. Noise, limited observations, and special geometry (e.g., only certain strikes being liquid) are all challenges in working with real data sets and imply that some methods may not translate well from synthetic to real-life applications.

Implied volatility: the value σ_{Imp} that inverts the Black–Scholes formula for a given call/put price $P^{Mkt}(t, S)$ and known r, K, T

3.2. Training

To approximate a derivatives valuation function, the modeler must select the domain of application $\bar{\mathcal{D}}$ and then the actual (discrete) training set $\mathcal{D} \in \bar{\mathcal{D}}$. Since the surrogate by construction minimizes the error between its estimates and that of the training data that it is presented, it is important to choose a representative domain. The trade-off between larger and smaller $\bar{\mathcal{D}}$ is one of generality versus model complexity and training time (Ferguson & Green 2018). For example, in the case of a Bermudan swaption we may choose to take as given the properties of a specific trade and then train the model against only a variety of input market data scenarios. Lower-dimensional $\bar{\mathcal{D}}$ will lead to smaller surrogates and hence lower training requirements (size of $\mathcal{D}$, amount of time spent training, etc.).

The sampling distribution underlying $\mathcal{D}$ is also crucial. For example, there is little point in generating a large volume of examples that yield zero option value. Similarly, regions of rapidly changing valuations need to be assigned more training data. Hence, the sampling distribution should often be nonuniform and capture financial dependence between its inputs. One may also complement with synthetic boundary inputs to enforce out-of-sample extrapolation shape (Ackerer et al. 2020, Ludkovski & Saporito 2022).

In terms of training the NN surrogate itself, Garcia & Gençay (2000) and Gençay & Qi (2001) look at enhancements such as early stopping and bagging. Gonon & Schwab (2021) establish convergence rates for deep NN surrogates of option pricing in exponential Lévy models, exploiting the relative smoothness of the latter functionals and pointing out the role of different activation functions. Specifically, they show that $\mathcal{O}(\epsilon^{-1})$ neurons are needed to achieve an error of at most ϵ.

3.3. No Arbitrage

Option pricers must satisfy certain model-free constraints imposed no arbitrage. For example, for call options, no arbitrage implies (we provide the financial names for the respective constraints on price sensitivities)

- a calendar spread constraint, where prices increase in time to maturity $T - t$;
- a bull spread constraint, where prices are increasing in moneyness S/K; and
- a butterfly spread constraint, where prices are convex in moneyness.

Dugas et al. (2009) achieve global arbitrage-free predicted prices using hard constraints through a special one-layer network architecture. Yang et al. (2017) use a gated network structure, constrained to have nonnegative weights and certain activation functions. Chataigner et al. (2020) built a related, sparsely connected architecture with a convex activation function $\text{softplus}(x) = \log(1 + e^x)$. An alternative is to penalize no-arbitrage violations, known as soft constraints (Ackerer et al. 2020, Chataigner et al. 2020). Itkin (2019) introduces penalty functions to enforce the positivity of the first and second derivatives of Π with respect to maturity T and strike K, respectively, in addition to the negativity of the first derivative with respect to K. Note that such penalization is done during training and hence does not necessarily apply to out-of-sample inputs.

3.4. Learning Implied Volatility

Delta hedging: stochastic calculus theory shows that holding (and continuously rebalancing) a portfolio with $\Delta(t, S_t)$ shares will replicate the option payoff, achieving perfect risk elimination

Theta: sensitivity of option value to time to maturity $T - t$

Instead of learning option prices, a strand of literature considers learning the implied volatility surface $\sigma_{\mathrm{Imp}}(\cdot)$. The respective volatility smile—the dependence of σ_{Imp} on option parameters—reflects deviations of the market from the Black–Scholes paradigm as the strike K and maturity T vary. This can be done by training on option prices $\Pi(T, K)$ or directly on implied volatility (i.e., transforming the inputs as well) (Liu et al. 2019). The respective surrogate enables transferring information obtained from vanilla calls and puts to a full-fledged asset model whereby one may price (with or without surrogates) arbitrary exotic options (i.e., options with nonstandard payoffs, unlike calls and puts). The translation is achieved through the Dupire formula that converts implied volatilities into a local volatility surface (Gatheral 2011),

$$\frac{\sigma^2(T,K)}{2} = \frac{\partial_T \Pi(T,K) + rK\partial_K \Pi(T,K)}{K^2 \partial^2_{KK} \Pi(T,K)}.$$

Observe that above, the right-hand side is required to be positive; that is, the surrogate for Π must be constrained to have nonnegative ∂_T and nonnegative ∂^2_{KK}.

Ackerer et al. (2020) train NN surrogates to observed implied volatilities using a mix of L^1 and L^2 norms and penalize violations of the above constraints (see also Zheng et al. 2021). They point out that inputting σ_{Imp} is more stable statistically than training to observed prices. Chataigner et al. (2020) use the implied volatility surface as an intermediate transformation—both inputs and desired outputs are option prices, and σ_{Imp} is solely an intermediate output of the surrogate. Fengler (2009) uses natural cubic splines with monotonicity and convexity constraints to build an arbitrage-free implied volatility surface. Glau et al. (2019a) learn implied volatility surfaces using Chebyshev interpolation.

4. LEARNING THE GREEKS

The task of computing contract sensitivities is a fundamental problem in QF. Known collectively as the Greeks, option sensitivities are essential for risk management. For example, Delta hedging manages risk by controlling for the sensitivity of the financial derivative to the underlying spot price, Theta manages risk by controlling for the sensitivity to the passing of time, and so on. Thus, successful hedging strategies depend on accurately learning such sensitivities.

Given a pricing functional $\Pi(\cdot)$, the Greek evaluation problem consists of evaluating $\partial\Pi$ with respect to the desired coordinate(s). Let $\Pi(\mathbf{x}; \lambda)$ be the option price, where $\mathbf{x}$ is the current stochastic market state (e.g., the underlying stock price S) and λ is the contract or model feature (e.g., the strike, the maturity, or the volatility). As the gradients of $\Pi(\cdot)$ with respect to different coordinates of either $\mathbf{x}$ or λ are rarely available analytically, several papers (Fu et al. 2012, Capriotti et al. 2017, Ruf & Wang 2022) have addressed Greek functional approximation.

As a canonical example of hedging, consider learning the Delta of a European option,

$$\Delta(t, S) := \partial\Pi(t, S)/\partial S,$$

for arbitrary time t and underlying price S. The functional $\Pi(\cdot, \cdot)$ is not directly known, and one is provided a training set $\mathcal{D} = \{(t^i, S^i, y^i) : i = 1, \ldots, N\}$, where $y^i \simeq \Pi(t^i, S^i)$. This is very similar to option pricing, except now the task is to learn $(t, S) \mapsto \Delta(t, S)$.

One approach is to fit a surrogate $(t, S) \mapsto \widehat{\Pi}(t, S)$ and then set $\widehat{\Delta}(t, S) := \partial\widehat{\Pi}(t, S)/\partial S$. The second step can be done either analytically for certain surrogates or via backpropagation, that is, adjoint algorithmic differentiation (Capriotti & Giles 2012, Capriotti et al. 2017). Surrogate construction for Greeks estimation must be approached with care, since for some classes, such as a spline-based $\widehat{\Pi}$, differentiation is known to lead to highly unstable or even nonsensical estimates for $\widehat{\Delta}$ and other gradients (Jain & Oosterlee 2015). This is because the typical L^2 criterion $\mathcal{L}_{\text{MSE}}$ that is driving the fitting of $\widehat{\Pi}(t, S)$ is completely unaware of the subsequent plan to compute gradients. Indeed, assessment of $\widehat{\Delta}$ is typically based on properties of the tracking error V_T in Equation 8.

GPs have been considered for this task by Crépey & Dixon (2020), Chataigner et al. (2021), and Ludkovski & Saporito (2022), relying on the availability of analytic derivatives $\frac{\partial m_*}{\partial x_j}(\mathbf{x}_*) = \frac{\partial m}{\partial x_j}(\mathbf{x}_*) + \frac{\partial \kappa}{\partial x_j}(\mathbf{x}_*, \mathbf{x}^{1:N})(\mathbf{K} + \sigma_\epsilon^2 \mathbf{I})^{-1}(\mathbf{y} - \mathbf{m})$, which admit the closed-form expression for the three kernel families provided in Section 2.2. On the one hand, this reduces the error in $\widehat{\Delta}$ since only a single approximation is needed, and the differentiation is exact. On the other hand, GPs offer an in-model assessment of the accuracy of $\widehat{\Delta}$ by rigorously propagating the underlying uncertainty about $\widehat{\Pi}$. The outputted credible bands around $\widehat{\Delta}$ guide the end user on how well the surrogate learned the desired Greek. This information is relevant for trading purposes, such as in the context of no-transaction regions under trading costs (Whalley & Wilmott 1997). Chataigner et al. (2021) consider shape-constrained GPs since arbitrage constraints heavily impact the shape of the Greeks (such as Δ being monotone).

NN surrogates for Greek estimation have been considered by Chataigner (2021), Chataigner et al. (2021), Jain & Oosterlee (2015), and Davis et al. (2020). NNs are attractive thanks to their uniform approximation, not just for smooth response functions but also for their derivatives (Hornik et al. 1990). Chebyshev interpolation for Greeks is proposed by Maran et al. (2021).

Figure 1 displays the estimated Delta of a call option in a Black–Scholes model (Equation 3). We employ two different surrogate types: GPs and NNs. The first method employs a Matérn-5/2 covariance kernel and a constant prior mean $m(\mathbf{x}) = \beta_0$; the NN employs the ELU activation function, with $L = 3$ layers and $n_\ell = 64$ neurons. The GP is trained via maximum likelihood and implemented in R; the NN is trained via the Adam algorithm and implemented in Python. For both cases the training set is two dimensional across time to maturity τ and stock price S, fixing the option strike $K = 40$, interest rate $r = 0.04$, and volatility $\sigma = 0.22$. Call price data are coming

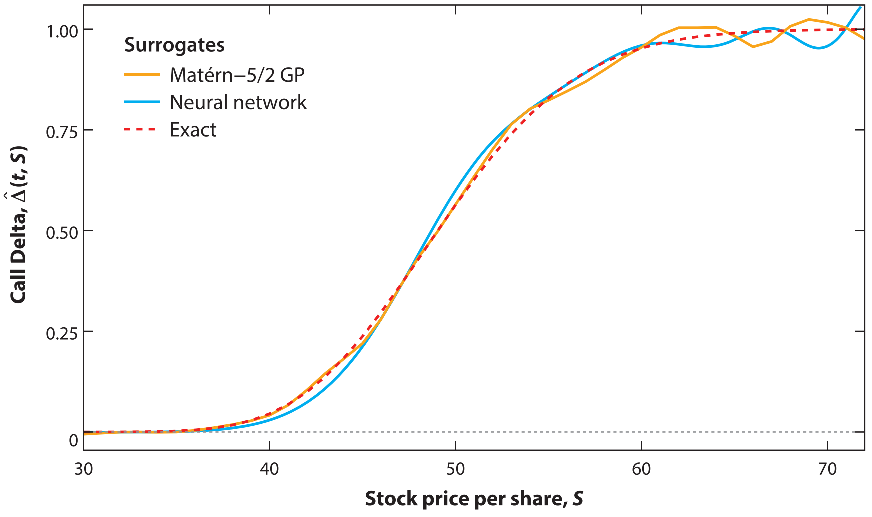

Figure 1

Estimated Delta $S \mapsto \widehat{\Delta}(t, S)$ of a Black–Scholes call with parameters $K = 40$, $r = 0.04$, and $\sigma = 0.22$, and time to maturity $T - t = 0.3$ as a function of S. Abbreviations: GP, Gaussian process; NN, neural network.

from noisy Monte Carlo samples, using $\check{N} = 400$ draws of S_T to approximate $\mathbb{E}^{\mathbb{Q}}[e^{-r(T-t)}(S_T - K)_+ | S_t = s]$ at $N = 400$ gridded input sites. We observe some errors around the edges with a nonmonotone $\widehat{\Delta}$ and violation of the no-arbitrage restriction on the far right, $\widehat{\Delta} > 1$. The fact that the surrogate prediction is better in the middle of the training domain $\bar{\mathcal{D}}$ is very typical. Ludkovski & Saporito (2022) provide more on the role of experimental design, demonstrating, for instance, that learning sensitivities on a grid is much faster than on an irregularly sampled domain.

An alternative way to learn the Delta specifically is to take the hedging perspective. Since delta hedging is risk-minimizing, one can learn hedging ratios by minimizing one-step portfolio risk. To this end, Ruf & Wang (2022) propose minimizing local portfolio variance, $\mathrm{Var}(V(\delta)) := \mathrm{Var}(\delta \cdot S_1 + (1 + r\Delta t)(\Pi_0 - \delta \cdot S_0) - \Pi_1)$, where S_0, S_1 are the underlying prices today and tomorrow and Π_0, Π_1 are the respective call prices. Modulo time discretization, the minimizer is $\delta^* = \partial \Pi(0, S_0)/\partial S$, that is, the Delta. Ruf & Wang (2022) then train an NN surrogate to real-life (S&P 500) observations of $(S_0^{1:N}, S_1^{1:N}, \Pi_0^{1:N}, \Pi_1^{1:N})$ plus additional features, such as σ_{Imp}. To match no-arbitrage constraints that imply that the hedging ratio must lie in the interval [0, 1], they employ a sigmoid activation function on the output (see also Halperin 2020).

5. MODEL CALIBRATION

The model calibration task (Bayer et al. 2019, Liu et al. 2019, Benth et al. 2021, Horvath et al. 2021) considers the inverse problem of finding the best set of model parameters θ that match observed empirical prices. To this end, one wishes to obtain the pricing functional $\theta \mapsto \Pi(t, \mathbf{x}, \lambda; \theta)$ and then to solve the calibration task,

$$\inf_\theta \frac{1}{M} \sum_{m=1}^{M} \left(\Pi(t, \mathbf{x}^m, \lambda^m; \theta) - P^{Mkt}(t, \mathbf{x}^m, \lambda^m)\right)^2, \qquad 14.$$

where $\mathcal{P} := \{P^{Mkt}(t, \mathbf{x}^m, \lambda^m) : m = 1, \dots, M\}$ is the collection of observed option prices with features $\lambda^{1:M}$ at date t.

With a statistical machine learning approach introduced by Horvath et al. (2021), one first trains a surrogate $\widehat{\Pi}(t, \mathbf{x}, \lambda; \theta)$ using a training set $\mathcal{D} := \{(t^j, \mathbf{x}^j, \lambda^j; \theta^j), j = 1, \dots, N\}$ (of size N, picked by the modeler) and then optimizes θ for the given observation set $\mathcal{P}$,

$$\hat{\theta}(\mathcal{P}) := \arg\inf_\theta \frac{1}{M} \sum_m \left(\widehat{\Pi}(t, \mathbf{x}^m, \lambda^m; \theta) - P^{Mkt}(t, \mathbf{x}^m, \lambda^m)\right)^2. \qquad 15.$$

Since solving Equation 14 requires repeatedly evaluating Π across different θ's, when the latter is expensive, there is a large gain from switching to a fast-to-evaluate surrogate $\widehat{\Pi}$. Indeed, Equation 15 can yield orders-of-magnitude performance gains, allowing on-the-fly calibration by traders in real time. At the same time, the modeler may invest essentially unlimited resources in training $\widehat{\Pi}$ (e.g., via deep NNs) as long as its ultimate evaluation is fast.

Huh (2019) calibrates exponential Lévy models using NNs. Itkin (2019) suggests calibrating models by first building a surrogate for the forward pricing functional $(\mathbf{x}, \lambda, \theta) \mapsto \Pi(\mathbf{x}, \lambda; \theta)$ and then inverting it to learn the map $(\mathbf{x}, \lambda, \Pi) \mapsto \theta$ via a second NN. As an example, he considers learning the implied volatility map $(\mathbf{x}, \lambda, \Pi) \mapsto \sigma_{\mathrm{Imp}}(\mathbf{x})$ and then using observed σ_{Imp} to calibrate a local volatility model.

6. LEARNING TO STOP

Valuing American-style contracts is a particular case of optimal stopping problems (OSPs), where the goal is to evaluate the value function $V : [0, T] \times \mathcal{X} \to \mathbb{R}$ representing expected reward,

$$V(t, \mathbf{x}) := \sup_{\tau \in \mathfrak{S}_t} \mathbb{E}\left[g(\tau, X_\tau) | X_t = \mathbf{x}\right]. \qquad 16.$$

For the remainder of the section we adopt the discrete-time paradigm of Bermudan options, indexing by k rather than t_k and taking $T = t_{K_T}$.

It is most intuitive to think of optimal stopping as dynamic decision making. At each exercise step k, the controller must decide whether to stop (0) or continue (1), which, within a Markovian structure, is encoded via the feedback strategy $\mathcal{A} = (A_{0:K_T}(\cdot))$ with each $A_k(\mathbf{x}) \in \{0, 1\}$. The action map A_k gives rise to the stopping region,

$$\mathcal{S}_k := \{\mathbf{x} \in \mathcal{X} : A_k(\mathbf{x}) = 0\} \subseteq \mathcal{X},$$

where the decision is to stop, and in parallel defines the corresponding first hitting time,

$$\tau_{A_{k:K_T}} := \inf\{\ell \geq k : A_\ell(X_\ell) = 0\} \wedge K_T = \sum_{m=k}^{K_T} m \cdot (1 - A_m(X_m)) \prod_{\ell=k}^{m-1} A_\ell(X_\ell), \qquad 17.$$

which is an optimal exercise time after k. Hence, solving an OSP is equivalent to classifying any $(\mathbf{x}, k)$ into $\mathcal{S}_k$ or its complement, the continuation set. For example, in the most well-known example of the Bermudan put, it is known that $\mathcal{S}_k = [0, \bar{s}_k]$; that is, one should stop as soon as the asset price drops below the exercise thresholds $\bar{s}_k$.

The action set A_k is characterized recursively as

$$A_k(\mathbf{x}) = 1 \quad \Longleftrightarrow \quad \mathbb{E}\Big[g(\tau_{A_{k+1:K_T}}, X_{\tau_{A_{k+1:K_T}}}) | X_k = \mathbf{x}\Big] > g(k, \mathbf{x}); \qquad 18.$$

that is, one should continue if the expected reward-to-go dominates the immediate payoff. Denoting the step-ahead conditional expectation of the value function by

$$q(k, \mathbf{x}) := \mathbb{E}\left[V(k+1, X_{k+1}) \,|X_k = \mathbf{x}\right] \qquad 19.$$

and using the dynamic programming principle that asserts that $V(k, \mathbf{x}) = \max\big(g(k, \mathbf{x}), \mathbb{E}[V(k+1, X_{k+1}) | X_k = \mathbf{x}]\big)$, we can equivalently write $A_k(\mathbf{x}) = 1 \;\Leftrightarrow\; q(k, \mathbf{x}) > g(k, \mathbf{x})$. The q-value $q(k, \cdot)$ is known as the continuation value.

The regression Monte Carlo (RMC) (also known as the Longstaff–Schwartz Algorithm or least squares Monte Carlo, the terminology being historical and potentially a misnomer for statisticians) framework (Longstaff & Schwartz 2001, Egloff 2005) recursively constructs approximate $\widehat{A}_k$'s through iterating on either Equation 18 or Equation 19. Thus, the RMC framework generates functional approximations of the continuation values $\hat{q}(k, \cdot)$ in order to build $\widehat{A}_k(\cdot)$. The RMC loop is initialized with $\widehat{V}(K_T, \mathbf{x}) = g(K_T, \mathbf{x})$, and for $k = K_T - 1, \ldots, 1$ it repeats as follows.

1. Learn the q-value $\hat{q}(k, \cdot)$.
2. Set $\widehat{A}_k(\mathbf{x}) := 1_{\{\hat{q}(k,\mathbf{x}) > g(k,\mathbf{x})\}}$.
3. Set $\widehat{V}(k, \mathbf{x}) := \max\big(\hat{q}(k, \mathbf{x}), g(k, \mathbf{x})\big)$.

The principal surrogate fitting task in step 1 can be implemented as learning $\mathbf{x} \mapsto \mathbb{E}[\widehat{V}(k + 1, X_{k+1}) | X_k = \mathbf{x}]$ (Tsitsiklis & van Roy 2001) or as approximating $\mathbf{x} \mapsto \mathbb{E}[g(\tau_{\widehat{A}_{k+1:K_T}}, X_{\tau_{\widehat{A}_{k+1:K_T}}}) \big| X_k = \mathbf{x}]$ (Longstaff & Schwartz 2001; see also a multistep, look-ahead version in Egloff et al. 2007). These choices are distinct because $\widehat{V}(k+1, \mathbf{x}) \neq \mathbb{E}[g(\tau_{\widehat{A}_{k+1:K_T}}, X_{\tau_{\widehat{A}_{k+1:K_T}}}) | X_{k+1} = \mathbf{x}]$ due to the approximation error.

Classical RMC employs an $\mathcal{L}_{\text{MSE}}$ loss function with user-specified basis functions, fitted based on a training set $\mathcal{D}_k$. The Monte Carlo reference is to the standard strategy of constructing $\mathcal{D}_k$ from M i.i.d. draws of X_k, obtained by a Monte Carlo simulation of M respective paths of (X_k) (thus $\mathcal{D}_k$ and $\mathcal{D}_{k+1}$ are not independent). Observe again a mismatch between $\mathcal{L}$ and the performance metric for evaluating $\widehat{A}_{0:K}$, which is based on the induced expected reward $\mathbb{E}\big[g(\tau_{\widehat{A}_{0:K_T}}, X_{\tau_{\widehat{A}_{0:K_T}}}) \big| X_0 = \mathbf{x}\big]$, as usual evaluated via a Monte Carlo average. Nevertheless RMC has turned out to be enormously successful; the seminal article by Longstaff & Schwartz (2001) has more than 4,000 citations and has spawned numerous enhancements [see reviews in Broadie & Cao 2008, Kohler 2010, Tompaidis & Yang 2013, and the monograph by Belomestny & Schoenmakers (2018)].

6.1. Surrogates for Optimal Stopping

Among major types of learning approaches, that is, different ways of executing the regression, we mention the following:

- piecewise regression with adaptive subgrids by Bouchard & Warin (2011), the idea being to avoid global fits that tend to be too rough and to come up with simple base fits (constant or linear) defined over a collection of subregions C_ℓ: $\widehat{f}(\mathbf{x}) = \sum_{\ell=1}^{L} \hat{f}_\ell(\mathbf{x}) 1_{\{\mathbf{x} \in C_\ell\}}$, with the latter chosen to be rectangular and equi-probable in $\mathcal{D}_k$ [see also dynamic trees proposed by Gramacy & Ludkovski (2015) who generate partitions via a probabilistic genetic ensemble approach resembling a random forest];
- regularized regression, such as LASSO (Kohler & Krzyżak 2012) and ridge regression (Hu & Zastawniak 2020), the idea being to start with a large number of potential bases and prevent overfitting by shrinking the irrelevant regression coefficients to zero;
- kernel regression by Belomestny (2011b), the idea being to provide a nonparametric surrogate based on a kernel function $\kappa(\mathbf{x}; h)$ that reduces to the choice of the kernel bandwidth h (see also nearest-neighbor regression in Agarwal & Juneja 2015);
- GP regression by Goudenège et al. (2019), Goudenège et al. (2020), and Ludkovski (2018);
- NN surrogates introduced by Haugh & Kogan (2004) and Kohler et al. (2010) within a single-layer architecture [Becker et al. (2020) recently considered deep NNs];
- Chebyshev polynomials by Glau et al. (2019b);
- smoothing splines by Kohler (2008); and
- adaptive regression bases by Belomestny et al. (2018).

Respective error analysis, focusing on error propagation from $\widehat{q}(k+1, \cdot)$ to $\widehat{q}(k, \cdot)$, is provided by Clément et al. (2002), Belomestny (2011a), Belomestny (2011b), Fromkorth & Kohler (2011), and Zanger (2018), primarily addressing linear models or classical kernel regressors.

Figure 2 visualizes three different surrogates when pricing a Bermudan put. The underlying dynamics $X_t \equiv S_t$ follow the one-dimensional Black–Scholes model; we consider 25 exercise rights over the horizon of $T = 1$ year and strike $K = 40$. The two panels show various approximations of the timing value $\widehat{T}(k, \mathbf{x}) := q(k, \mathbf{x}) - g(k, \mathbf{x})$ for two intermediate steps $k = 10$ and 20. The timing value summarizes the relative merit of stopping and continuing: The buyer should exercise when the timing value is negative, $\widehat{A}_k(\mathbf{x}) = 0 \Leftrightarrow \widehat{T}(k, \mathbf{x}) < 0$, and continue otherwise. The value function can be recovered via $\widehat{V}(k, \mathbf{x}) = g(k, \mathbf{x}) + \max(0, \widehat{T}(k, \mathbf{x})t)$. In **Figure 2** we observe that the spline surrogate yields the smoothest fit, while the GP surrogate appears to overfit in this example. The displayed 95% uncertainty band for the latter is rather wide and straddling the zero level, indicating that it is aware that the predicted exercise rule is not very accurate. We again observe some instabilities at the edges of the training domain. Additional comparisons are provided by Ludkovski (2020).

We refer readers to Ludkovski (2018), Gramacy & Ludkovski (2015), and Ludkovski (2020) for a discussion of experimental designs for RMC. Note that the surrogate is needed for only the in-the-money region $\mathcal{X}_{\text{in}} := \{\mathbf{x} : g(k, \mathbf{x}) > 0\}$; when $g(k, \mathbf{x}) = 0$ it is clear that continuing is better, $\widehat{A}_k(\mathbf{x}) = 1$. Thus, $\mathcal{D}_k$ is typically restricted to lie in $\mathcal{X}_{\text{in}}$, for example, $\mathcal{D}_k \subseteq [25, 40]$ in **Figure 2**.

Motivated by the possibility of exact integration for certain functionals in the Black–Scholes world, one may bypass the approximation of conditional expectations by picking the form of $\widehat{V}$

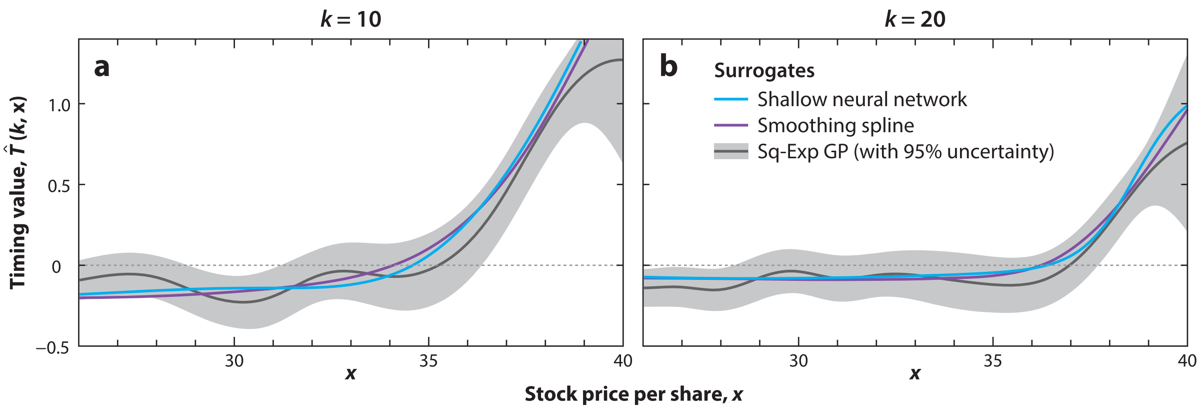

Figure 2

Surrogates for the timing value $\widehat{T}(k, \mathbf{x})$ in a one-dimensional Bermudan put problem, with $\mathbf{x}$ being the stock price. We show three surrogates fitted to the same problem: a GP utilizing the kernel in Equation 11 (*black*), a smoothing spline (*purple*), and a shallow one-layer NN (*blue*). The surrogates are from time step $k = 10$ ($t = 0.4$, *left*) and $k = 20$ ($t = 0.8$, *right*). We also display the uncertainty quantification regarding the GP fit of $\hat{T}(k, \mathbf{x})$ (*gray bands*, 95%). Abbreviations: GP, Gaussian process; NN, neural network.

such that $\mathbb{E}[\widehat{V}(k+1, X_{k+1})|\, X_k]$ can be done analytically. Glasserman & Yu (2004b), Balata & Palczewski (2017, 2018), and Glau et al. (2019b) all consider polynomial-type surrogates for American options and more general control problems.

A deep learning approach to American option pricing was proposed by Becker et al. (2019) via learning the action sets. Using Equation 17 one may rewrite

$$V(k, X_k) = g(k, X_k)(1 - A_k(X_k)) + g(\tau_{A_{k+1:K_T}}, X_{\tau_{A_{k+1:K_T}}})A_k(X_k).$$

Parameterizing A_k via an NN A_k^{ϑ}, one proceeds to learn $\widehat{\vartheta}$ by repeatedly maximizing the minibatch empirical average of the above right-hand side over the paths $\mathbf{x}_{0:K_T}^{1:N}$,

$$\frac{1}{N}\sum_{n=1}^{N}\left\{g(k, \mathbf{x}_k^n)(1 - A_k^{\vartheta}(\mathbf{x}_k^n)) + g(\tau_{\widehat{A}_{k+1:K_T}}, \mathbf{x}^n_{\tau_{\widehat{A}_{k+1:K_T}}})A_k^{\vartheta}(\mathbf{x}_k^n)\right\}. \qquad 20.$$

In order to implement stochastic gradient descent over Equation 20 in ϑ, A^{ϑ} must be smoothed during training by modifying its output activation function so as to map into (0, 1) rather than {0, 1}. Becker et al. (2019) propose to reuse the same NN weights $\widehat{\vartheta}$ for the mollified and the hard-thresholded A_k surrogates. In a very recent preprint, Reppen et al. (2022) consider learning the stopping set $\mathfrak{S}_t$ through its graph or boundary (assumed to be a smooth curve, e.g., as in American put setups).

The natural strategy of finding Greeks for Bermudan contracts based on the already computed surrogate for $V(t, \mathbf{x})$ has been treated by Belomestny et al. (2010b), Jain & Oosterlee (2015), Glau et al. (2019b), and Jain et al. (2019).

Swing option: a contract that provides flexibility as to when and how much of a commodity is taken; the buyer has multiple exercise opportunities to swing the quantity delivered

6.2. Extensions: Swinging, Switching, and Impulsing

Beyond American options, OSPs are highly relevant as building blocks for further dynamic decision-making problems. For example, in multiple optimal stopping formulations the decision maker must select a sequence of $\tau_1 < \tau_2 < \cdots < \tau_M$ distinct stopping times and maximize $\mathbb{E}[\sum_m e^{-r\tau_m} g(X_{\tau_m})]$. This corresponds to making a sequence of M decisions with the goal of maximizing aggregate discounted reward. Such swing options are common in commodity and electricity markets (Carmona & Touzi 2008). Extending RMC to multiple stopping requires constructing surrogates $\hat{V}^{(m)}(k, \mathbf{x})$ that enumerate the number m of remaining exercise rights, capturing the marginal value of each decision. Each $\hat{V}^{(m)}, m \geq 1$ is characterized as a solution of an OSP where the payoff is related to $\hat{V}^{(m-1)}$. RMC for multiple stopping was pioneered by Meinshausen & Hambly (2004) using linear models. Readers are referred to Kirkby & Deng (2019) for a version utilizing B-splines and Ludkovski (2021) for implementation with GPs. Deschatre & Mikael (2020) extend the policy parameterization of Becker et al. (2019) to multiexercise contracts.

Optimal switching formulations arise in the limit $M \to \infty$ above, whereby the controller is able to make an infinite series of discrete decisions, sequentially altering the state of the controlled system (X_t). This class of models also subsumes the situation where the decisions are not binary but are in some finite action space $\mathcal{U}$. Motivating case studies include on/off control of a power plant, management of natural gas storage facilities (which can be in the gas injection, gas withdrawal, and holding regimes) (Carmona & Ludkovski 2010, Mazières & Boogert 2013, Nadarajah et al. 2017, Ludkovski & Maheshwari 2020), and capacity expansion models (Aid et al. 2014). Simulation-based algorithms for optimal switching construct several surrogates $V^{(u)}(k, \mathbf{x})$ that are indexed by the current control regime $u \in \mathcal{U}$. Among recent works, Ludkovski & Maheshwari (2020) study GP surrogates for this purpose and Bachouch et al. (2022) consider deep NNs.

Impulse control is a further generalization that features a double sequence of stopping times and impulse amounts, $A := (\tau_m, z_m)$. The interpretation is that the state process (X_t^A) is subject to stochastic differential equation dynamics, as well as repeated lumpy interventions or shocks of size $z_m \in \Xi$ at chosen instances $\tau_m \in [0, T]$: $X_s^{t,x,A} = x + \int_t^s \mu(X_r^{t,x,A})\mathrm{d}r + \int_t^s \sigma(X_r^{t,x,A})\mathrm{d}W_r + \sum_{m:t<\tau_m \leq s} z_m$. The goal of the controller is to maximize rewards driven by X and her actions (τ_m, z_m). Impulse control can be reduced to repeated optimal stopping where the action is compound: After deciding to act, the controller evaluates the intervention operator $\mathcal{M}\widehat{V}(t, \mathbf{x}) := \sup_{z \in \Xi}\{\widehat{V}(t, \mathbf{x} + z) - \kappa(\mathbf{x}, z)\}$ to select the best impulse. Ludkovski (2022) and Deschatre & Mikael (2020) provide further details.

7. SURROGATES IN STOCHASTIC CONTROL ALGORITHMS

In Markov stochastic control the starting point for statistical learning is the Bellman equation, which provides a recursive characterization of the underlying value function. Let (X_k^u) be the discrete-time controlled state process, with controls (u_k) taking values in $u_k \in \mathcal{U} \subseteq \mathbb{R}^p$. Let $g(k, \mathbf{x}, u)$ be the stepwise cost that depends on the current state $\mathbf{x}$ and control u. Then the Bellman equation for minimizing costs on the horizon $k \in \{0, 1, \ldots, K_T - 1\}$ with terminal condition $\bar{V}(\mathbf{x})$ is

$$V(k, \mathbf{x}) = \inf_{u \in \mathcal{U}} \Big\{ g(k, \mathbf{x}, u) + \mathbb{E}\left[V(k+1, X_{k+1}^u) | X_k = \mathbf{x}\right] \Big\}, \qquad V(K_T, \mathbf{x}) = \bar{V}(\mathbf{x}), \qquad 21.$$

with the superscript emphasizing the influence of the control on the transition density of $X_{k+1}^u | X_k$. Like in the prior section, an increasingly popular approach is to construct functional approximators $\widehat{V}(k, \cdot)$ that are trained based on empirical samples of $(X_k, u_k, X_{k+1}^{u_k})$. This approach dates back to at least Chen et al. (1999), who proposed spline surrogates. Readers are referred to Deisenroth et al. (2009) for a GP implementation and Belomestny et al. (2010a) for linear models in the flavor of RMC. Recently, there has been an explosion of interest in applying deep learning (Han & E 2016, Fecamp et al. 2020, Germain et al. 2021, Bachouch et al. 2022).

A different approach is to parameterize the set of strategies $\widehat{u}^{\vartheta}(k, \cdot)$ and then to maximize expected reward over ϑ (Huré et al. 2021). Another alternative is to learn the q-value $\widehat{q}(k, \mathbf{x}, u)$ that summarizes costs to go jointly in terms of state-action pairs; optimal policy is then extracted as the minimizer of $\widehat{q}(k, \mathbf{x}, \cdot)$ (Chen & Ludkovski 2021). Note that in Equation 21, optimal feedback control $u^*(k, \mathbf{x})$ can be characterized in terms of the gradient of $V(k, \mathbf{x})$, connecting to the literature in Section 4. This intrinsic coupling between value function and feedback control is weakened in actor-critic approaches that build separate surrogates for $\widehat{V}$ and $\widehat{u}$ in the interest of computational efficiency (Guéant & Manziuk 2019, Cao et al. 2021). Experimental design for Equation 21 is addressed, for example, in the control randomization (Kharroubi et al. 2014, Zhang et al. 2019) approach.

Complementary to the above are machine learning techniques for nonlinear PDEs, which can be used to characterize continuous-time stochastic control problems using the Hamilton–Jacobi–Bellman representation. A respective Deep Galerkin Method was proposed by Sirignano & Spiliopoulos (2018) and has generated nearly 1,000 citations in less than 5 years.

There is by now a long list of specific financial control problems where statistical machine learning algorithms have been taken up. We mention portfolio optimization with transaction costs (Cong & Oosterlee 2016, Zhang et al. 2019), optimal execution in limit order books (Leal et al. 2020), and market making (Guéant & Manziuk 2019). Additional applications motivated by financial mathematics settings include (adaptive) robust control (Chen & Ludkovski 2021), principal agent problems (Baldacci et al. 2019), constrained control (Balata et al. 2021), ranking problems (Hu 2019), and McKean–Vlasov problems (Carmona & Laurière 2021).

A fruitful research area has been to extend the above methods to stochastic games, where equilibrium strategies are characterized through best-response conditions (Han & Hu 2020, Laurière 2021). One motivation is that finding equilibria requires repeated optimization over best responses, which is expensive to do directly and where fast surrogates are greatly beneficial. Fixed-point iteration concepts may be combined with the sequential training of the surrogate for best response (see, for example, the class of fictitious play algorithms).

Finally, we mention in passing reinforcement learning (RL) approaches that aim to solve for $V(t, \mathbf{x})$ all at once across space and time (Dixon et al. 2020, Charpentier et al. 2021, Hambly et al. 2021). RL has been investigated especially for learning data-driven hedging strategies (Buehler et al. 2019, Kolm & Ritter 2019, Cao et al. 2021, Giurca & Borovkova 2021, Ruf & Wang 2022) that aspire to be model-free.

8. OUTLOOK

Historically, statistical learning in QF has evolved semi-independently across several distinct applications such as American option pricing and learning the implied volatility surface. It remains the case that many papers propose new methodologies geared for a narrow or very specific context such that their broader relevance is hard to assess. Recent software suites such as those by Gevret et al. (2018) and Ludkovski (2021) aim to facilitate such meta-analysis and cross-comparison. Another gap is between the common test beds in academic circles and the practical concerns faced by practitioners such that the real-life applicability of new methods is often limited.

We are currently witnessing a surge of theoretical publications as well as a proliferation of industry start-ups that claim computational breakthroughs enabled by techniques such as deep learning. It will take some time to sort out what will be the long-standing advances that pass the test of time. What is clear is that QF applications have enough specialized features that customization and tailoring are critical, and hence no single tool is ever going to be the right one for all tasks. As such, it is worthwhile to adopt the higher perspective afforded by statistical learning theories and be simultaneously conversant in the language of stochastics, finance, statistics, and machine learning.

FUTURE ISSUES

1. Barriers due to different terminology used across research communities remain a challenge but also provide ongoing opportunities for knowledge transfer.
2. There is a need for better benchmarking test beds. Despite many recent works providing reproducible computational packages and notebooks, there is a lack of common benchmarks to enable meaningful comparison of tools and definition of state-of-the-art performance.
3. Principled selection of surrogate types and of training designs remains underexplored in quantitative finance applications.

DISCLOSURE STATEMENT

The author is not aware of any affiliations, memberships, funding, or financial holdings that might be perceived as affecting the objectivity of this review.

ACKNOWLEDGMENTS

M.L. is partially supported by National Science Foundation grant DMS-1821240. Comments from an anonymous reviewer are appreciated.

LITERATURE CITED

Ackerer D, Tagasovska N, Vatter T. 2020. Deep smoothing of the implied volatility surface. *Adv. Neural Inform. Proc. Syst.* 33:11552–63

Agarwal A, Juneja S. 2015. Nearest neighbor based estimation technique for pricing Bermudan options. *Int. Game Theory Rev.* 17(1):1540002

Aid R, Campi L, Langrené N, Pham H. 2014. A probabilistic numerical method for optimal multiple switching problems in high dimension. *SIAM J. Financ. Math.* 5(1):191–231

Bachouch A, Huré C, Langrené N, Pham H. 2022. Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications. *Methodol. Comput. Appl. Probab.* 24(1):143–78

Balata A, Ludkovski M, Maheshwari A, Palczewski J. 2021. Statistical learning for probability-constrained stochastic optimal control. *Eur. J. Oper. Res.* 290(2):640–56

Balata A, Palczewski J. 2017. Regress-later Monte Carlo for optimal inventory control with applications in energy. arXiv:1703.06461 [math.OC]

Balata A, Palczewski J. 2018. Regress-later Monte Carlo for optimal control of Markov processes. arXiv:1712.09705 [math.OC]

Baldacci B, Manziuk I, Mastrolia T, Rosenbaum M. 2019. Market making and incentives design in the presence of a dark pool: a deep reinforcement learning approach. arXiv:1912.01129 [q-fin.MF]

Bayer C, Horvath B, Muguruza A, Stemper B, Tomas M. 2019. On deep calibration of (rough) stochastic volatility models. arXiv:1908.08806 [q-fin.MF]

Becker S, Cheridito P, Jentzen A. 2019. Deep optimal stopping. *J. Mach. Learn. Res.* 20:2712–36

Becker S, Cheridito P, Jentzen A. 2020. Pricing and hedging American-style options with deep learning. *J. Risk Financ. Manag.* 13(7):158

Belomestny D. 2011a. On the rates of convergence of simulation-based optimization algorithms for optimal stopping problems. *Ann. Appl. Probab.* 21(1):215–39

Belomestny D. 2011b. Pricing Bermudan options by nonparametric regression: optimal rates of convergence for lower estimates. *Finance Stochast.* 15(4):655–83

Belomestny D, Kolodko A, Schoenmakers J. 2010a. Regression methods for stochastic control problems and their convergence analysis. *SIAM J. Control Optim.* 48(5):3562–88

Belomestny D, Milstein GN, Schoenmakers J. 2010b. Sensitivities for Bermudan options by regression methods. *Decis. Econ. Finance* 33(2):117–38

Belomestny D, Schoenmakers J. 2018. *Advanced Simulation-Based Methods for Optimal Stopping and Control: With Applications in Finance*. London: Palgrave Macmillan

Belomestny D, Schoenmakers J, Spokoiny V, Tavyrikov Y. 2018. Optimal stopping via deeply boosted backward regression. arXiv:1808.02341 [math.NA]

Benth FE, Detering N, Lavagnini S. 2021. Accuracy of deep learning in calibrating HJM forward curves. *Digit. Finance* 3(3):209–48

Binois M, Gramacy RB, Ludkovski M. 2018. Practical heteroskedastic Gaussian process modeling for large simulation experiments. *J. Comput. Graph. Stat.* 27(4):808–21

Bouchard B, Warin X. 2011. Monte-Carlo valorisation of American options: facts and new algorithms to improve existing methods. In *Numerical Methods in Finance*, ed. R Carmona, PD Moral, P Hu, N Oudjane, pp. 215–55. Heidelberg, Ger.: Springer

Broadie M, Cao M. 2008. Improved lower and upper bound algorithms for pricing American options by simulation. *Quant. Finance* 8(8):845–61

Buehler H, Gonon L, Teichmann J, Wood B. 2019. Deep hedging. *Quant. Finance* 19(8):1271–91

Cao J, Chen J, Hull J, Poulos Z. 2021. Deep hedging of derivatives using reinforcement learning. *J. Financ. Data Sci.* 3(1):10–27

Capponi A, Lehalle CA, eds. 2022. *Machine Learning in Financial Markets: A Guide to Contemporary Practice*. Cambridge, UK: Cambridge Univ. Press. In press

Capriotti L, Giles M. 2012. Adjoint Greeks made easy. *Risk* 25(9):92–98

Capriotti L, Jiang Y, Macrina A. 2017. AAD and least-square Monte Carlo: fast Bermudan-style options and XVA Greeks. *Algorithmic Finance* 6(1–2):35–49

Carmona R, Laurière M. 2021. Deep learning for mean field games and mean field control with applications to finance. arXiv:2107.04568 [math.OC]

Carmona R, Ludkovski M. 2010. Valuation of energy storage: an optimal switching approach. *Quant. Finance* 10(4):359–74

Carmona R, Touzi N. 2008. Optimal multiple stopping and valuation of swing options. *Math. Finance* 18(2):239–68

Charpentier A, Elie R, Remlinger C. 2021. Reinforcement learning in economics and finance. *Comput. Econ.* In press. **https://doi.org/10.1007/s10614-021-10119-4**

Chataigner M. 2021. *Some contributions of machine learning to quantitative finance: volatility, nowcasting, CVA compression.* PhD Thesis, Univ. Paris-Saclay, Paris

Chataigner M, Cousin A, Crépey S, Dixon M, Gueye D. 2021. Beyond surrogate modeling: learning the local volatility via shape constraints. *SIAM J. Financ. Math.* 12(3):SC58–69

Chataigner M, Crépey S, Dixon M. 2020. Deep local volatility. *Risks* 8(3):82

Chen T, Ludkovski M. 2021. A machine learning approach to adaptive robust utility maximization and hedging. *SIAM J. Financ. Math.* 12(3):1226–56

Chen V, Ruppert D, Shoemaker C. 1999. Applying experimental design and regression splines to high-dimensional continuous-state stochastic dynamic programming. *Oper. Res.* 47(1):38–53

Cheridito P, Gersey B. 2021. Computation of conditional expectations with guarantees. arXiv:2112.01804 [stat.CO]

Clément E, Lamberton D, Protter P. 2002. An analysis of a least squares regression algorithm for American option pricing. *Finance Stochast.* 6:449–71

Cong F, Oosterlee CW. 2016. Multi-period mean–variance portfolio optimization based on Monte-Carlo simulation. *J. Econ. Dyn. Control* 64:23–38

Cousin A, Maatouk H, Rullière D. 2016. Kriging of financial term-structures. *Eur. J. Oper. Res.* 255(2):631–48

Crépey S, Dixon MF. 2020. Gaussian process regression for derivative portfolio modeling and application to credit valuation adjustment computations. *J. Comput. Finance* 24(1):47–81

Culkin R, Das SR. 2017. Machine learning in finance: the case of deep learning for option pricing. *J. Invest. Manag.* 15(4):92–100

Davis J, Devos L, Reyners S, Schoutens W. 2020. Gradient boosting for quantitative finance. *J. Comput. Finance* 24(4):1–40

De Spiegeleer J, Madan DB, Reyners S, Schoutens W. 2018. Machine learning for quantitative finance: fast derivative pricing, hedging and fitting. *Quant. Finance* 18(10):1635–43

Deisenroth MP, Rasmussen CE, Peters J. 2009. Gaussian process dynamic programming. *Neurocomputing* 72(7):1508–24

Deschatre T, Mikael J. 2020. Deep combinatorial optimisation for optimal stopping time problems: application to swing options pricing. arXiv:2001.11247 [q-fin.CP]

Dixon MF, Halperin I, Bilokon P. 2020. *Machine Learning in Finance*. Cham, Switz.: Springer

Dugas C, Bengio Y, Bélisle F, Nadeau C, Garcia R. 2000. Incorporating second-order functional knowledge for better option pricing. *Adv. Neural Inform. Proc. Syst.* 13:472–78

Dugas C, Bengio Y, Bélisle F, Nadeau C, Garcia R. 2009. Incorporating functional knowledge in neural networks. *J. Mach. Learn. Res.* 10:1239–62

Duvenaud D. 2014. *Automatic model construction with Gaussian processes*. PhD Thesis, Univ. Cambridge, Cambridge, UK

Egloff D. 2005. Monte Carlo algorithms for optimal stopping and statistical learning. *Ann. Appl. Probab.* 15(2):1396–432

Egloff D, Kohler M, Todorovic N. 2007. A dynamic look-ahead Monte Carlo algorithm for pricing Bermudan options. *Ann. Appl. Probability* 17(4):1138–71

Elie R, Perolat J, Laurière M, Geist M, Pietquin O. 2020. On the convergence of model free learning in mean field games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34, pp. 7143–50. Palo Alto, CA: AAAI

Fecamp S, Mikael J, Warin X. 2020. Deep learning for discrete-time hedging in incomplete markets. *J. Comput. Finance* 25(2):51–85

Fengler MR. 2009. Arbitrage-free smoothing of the implied volatility surface. *Quant. Finance* 9(4):417–28

Ferguson R, Green AD. 2018. *Deeply learning derivatives*. SSRN Work. Pap. 3244821

Fromkorth A, Kohler M. 2011. Analysis of least squares regression estimates in case of additional errors in the variables. *J. Stat. Plan. Inference* 141(1):172–88

Fu H, Jin X, Pan G, Yang Y. 2012. Estimating multiple option Greeks simultaneously using random parameter regression. *J. Comput. Finance* 16(2):85–118

Garcia R, Gençay R. 2000. Pricing and hedging derivative securities with neural networks and a homogeneity hint. *J. Econom.* 94(1–2):93–115

Gaß M, Glau K, Mahlstedt M, Mair M. 2018. Chebyshev interpolation for parametric option pricing. *Finance Stochast.* 22(3):701–31

Gatheral J. 2011. *The Volatility Surface: A Practitioner's Guide*. Hoboken, NJ: John Wiley & Sons

Gençay R, Qi M. 2001. Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging. *IEEE Trans. Neural Netw.* 12(4):726–34

Germain M, Pham H, Warin X. 2021. Neural networks-based algorithms for stochastic control and PDEs in finance. arXiv:2101. [math.OC]

Gevret H, Langrené N, Lelong J, Warin X, Maheshwari A. 2018. *STochastic OPTimization library in C++*. Res. Rep., EDF Lab., Paris

Giurca A, Borovkova S. 2021. *Delta hedging of derivatives using deep reinforcement learning*. SSRN Work. Pap. 3847272

Glasserman P. 2004. *Monte Carlo Methods in Financial Engineering*. New York: Springer

Glasserman P, Yu B. 2004a. Number of paths versus number of basis functions in American option pricing. *Ann. Appl. Probab.* 14(4):2090–119

Glasserman P, Yu B. 2004b. Simulation for American options: regression now or regression later? In *Monte Carlo and Quasi-Monte Carlo Methods 2002*, ed. H Niederreiter, pp. 213–26. Berlin: Springer

Glau K, Herold P, Madan DB, Pötz C. 2019a. The Chebyshev method for the implied volatility. *J. Comput. Finance* 23(3):1–31

Glau K, Kressner D, Statti F. 2020. Low-rank tensor approximation for Chebyshev interpolation in parametric option pricing. *SIAM J. Financ. Math.* 11(3):897–927

Glau K, Mahlstedt M. 2019. Improved error bound for multivariate Chebyshev polynomial interpolation. *Int. J. Comput. Math.* 96(11):2302–14

Glau K, Mahlstedt M, Pötz C. 2019b. A new approach for American option pricing: the dynamic Chebyshev method. *SIAM J. Sci. Comput.* 41(1):B153–80

Gonon L, Schwab C. 2021. Deep ReLU network expression rates for option prices in high-dimensional, exponential Lévy models. *Finance Stochast.* 25(4):615–57

Goudenège L, Molent A, Zanette A. 2019. Variance reduction applied to machine learning for pricing Bermudan/American options in high dimension. arXiv:1903.11275 [q-fin.CP]

Goudenège L, Molent A, Zanette A. 2020. Machine learning for pricing American options in high-dimensional Markovian and non-Markovian models. *Quant. Finance* 20(4):573–91

Gramacy RB. 2020. *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences*. Boca Raton, FL: CRC

Gramacy RB, Ludkovski M. 2015. Sequential design for optimal stopping problems. *SIAM J. Financ. Math.* 6(1):748–75

Guéant O, Manziuk I. 2019. Deep reinforcement learning for market making in corporate bonds: beating the curse of dimensionality. *Appl. Math. Finance* 26(5):387–452

Györfi L, Kohler M, Krzyzak A, Walk H. 2002. *A Distribution-Free Theory of Nonparametric Regression*, Vol. 1. New York: Springer

Halperin I. 2020. QLBS: Q-learner in the Black-Scholes (-Merton) worlds. *J. Deriv.* 28(1):99–122

Hambly B, Xu R, Yang H. 2021. Recent advances in reinforcement learning in finance. arXiv:2112.04553 [q-fin.MF]

Han J, E W. 2016. Deep learning approximation for stochastic control problems. arXiv:1611.07422 [cs.LG]

Han J, Hu R. 2020. Deep fictitious play for finding Markovian Nash equilibrium in multi-agent games. *Proc. Mach. Learn. Res.* 107:221–45

Haugh M, Kogan L. 2004. Pricing American options: a duality approach. *Oper. Res.* 52(2):258–70

Hornik K, Stinchcombe M, White H. 1990. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Netw.* 3(5):551–60

Horvath B, Muguruza A, Tomas M. 2021. Deep learning volatility: a deep neural network perspective on pricing and calibration in (rough) volatility models. *Quant. Finance* 21(1):11–27

Hu R. 2019. Deep fictitious play for stochastic differential games. arXiv:1903.09376 [math.OC]

Hu W, Zastawniak T. 2020. Pricing high-dimensional American options by kernel ridge regression. *Quant. Finance* 20(5):851–65

Huh J. 2019. Pricing options with exponential Lévy neural network. *Expert Syst. Appl.* 127:128–40

Huré C, Pham H, Bachouch A, Langrené N. 2021. Deep neural networks algorithms for stochastic control problems on finite horizon: convergence analysis. *SIAM J. Numer. Anal.* 59(1):525–57

Hutchinson JM, Lo AW, Poggio T. 1994. A nonparametric approach to pricing and hedging derivative securities via learning networks. *J. Finance* 49(3):851–89

Itkin A. 2019. Deep learning calibration of option pricing models: some pitfalls and solutions. arXiv:1906.03507 [q-fin.CP]

Jain S, Leitao Á, Oosterlee CW. 2019. Rolling adjoints: fast Greeks along Monte Carlo scenarios for early-exercise options. *J. Comput. Sci.* 33:95–112

Jain S, Oosterlee CW. 2015. The stochastic grid bundling method: efficient pricing of Bermudan options and their Greeks. *Appl. Math. Comput.* 269:412–31

James G, Witten D, Hastie T, Tibshirani R. 2013. *An Introduction to Statistical Learning*. New York: Springer

Ke G, Meng Q, Finley T, Wang T, Chen W, et al. 2017. LightGBM: A highly efficient gradient boosting decision tree. *Adv. Neural Inform. Proc. Syst.* 30:3146–54

Kharroubi I, Langrené N, Pham H. 2014. A numerical algorithm for fully nonlinear HJB equations: an approach by control randomization. *Monte Carlo Methods Appl.* 20(2):145–65

Kirkby J, Deng S. 2019. Swing option pricing by dynamic programming with B-spline density projection. *Int. J. Theor. Appl. Finance* 22(08):1950038

Kohler M. 2008. A regression-based smoothing spline Monte Carlo algorithm for pricing American options in discrete time. *Adv. Stat. Anal.* 92(2):153–78

Kohler M. 2010. A review on regression-based Monte Carlo methods for pricing American options. In *Recent Developments in Applied Probability and Statistics*, ed. L Devroye, B Karasözen, M Kohler, R Kornpp, pp. 37–58. Heidelberg, Ger.: Springer

Kohler M, Krzyżak A. 2012. Pricing of American options in discrete time using least squares estimates with complexity penalties. *J. Stat. Plan. Inference* 142(8):2289–307

Kohler M, Krzyżak A, Todorovic N. 2010. Pricing of high-dimensional American options by neural networks. *Math. Finance* 20(3):383–410

Kolm PN, Ritter G. 2019. Dynamic replication and hedging: a reinforcement learning approach. *J. Financ. Data Sci.* 1(1):159–71

Laurière M. 2021. Numerical methods for mean field games and mean field type control. In *Mean Field Games*, ed. F Delarue, pp. 221–82. Providence, RI: Am. Math. Soc.

Leal L, Laurière M, Lehalle CA. 2020. Learning a functional control for high-frequency finance. arXiv:2006.09611 [math.OC]

Lemieux C. 2009. *Monte Carlo and Quasi-Monte Carlo Sampling*. New York: Springer

Liu S, Borovykh A, Grzelak LA, Oosterlee CW. 2019. A neural network-based framework for financial model calibration. *J. Math. Ind.* 9(1):9

Longstaff FA, Schwartz ES. 2001. Valuing American options by simulation: a simple least-squares approach. *Rev. Financ. Stud.* 14(1):113–47

Ludkovski M. 2018. Kriging metamodels and experimental design for Bermudan option pricing. *J. Comput. Finance* 22(1):37–77

Ludkovski M. 2020. mlOSP: Towards a unified implementation of regression Monte Carlo algorithms. arXiv:2012.00729 [q-fin.CP]

Ludkovski M. 2021. `mlOSP`: Regression Monte Carlo algorithms for optimal stopping. *R package*, version 1.0. **https://github.com/mludkov/mlOSP**

Ludkovski M. 2022. Regression Monte Carlo for impulse control. *Math. Action* 11:73–90

Ludkovski M, Maheshwari A. 2020. Simulation methods for stochastic storage problems: a statistical learning perspective. *Energy Syst.* 11(2):377–415

Ludkovski M, Saporito Y. 2022. KrigHedge: Gaussian process surrogates for Delta hedging. *Appl. Math. Finance* 28(4):330–60

Maran A, Pallavicini A, Scoleri S. 2021. *Chebyshev Greeks: Smoothing Gamma without bias*. SSRN Work. Pap. 3872744

Mazières D, Boogert A. 2013. A radial basis function approach to gas storage valuation. *J. Energy Mark.* 6(2):19–50

Meinshausen N, Hambly BM. 2004. Monte Carlo methods for the valuation of multiple-exercise options. *Math. Finance* 14(4):557–83

Nadarajah S, Margot F, Secomandi N. 2017. Comparison of least squares Monte Carlo methods with applications to energy real options. *Eur. J. Oper. Res.* 256(1):196–204

Olivares P, Alvarez A. 2016. Pricing basket options by polynomial approximations. *J. Appl. Math.* 2016:9747394

Rasmussen CE, Williams CKI. 2006. *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press

Reppen AM, Soner HM, Tissot-Daguette V. 2022. Neural optimal stopping boundary. arXiv:2205.04595 [q-fin.PR]

Risk J, Ludkovski M. 2018. Sequential design and spatial modeling for portfolio tail risk measurement. *SIAM J. Financ. Math.* 9(4):1137–74

Roustant O, Ginsbourger D, Deville Y. 2012. DiceKriging, DiceOptim: two R packages for the analysis of computer experiments by Kriging-based metamodeling and optimization. *J. Stat. Softw.* 51(1):1–55

Ruf J, Wang W. 2020. Neural networks for option pricing and hedging: a literature review. *J. Comput. Finance* 24(1):1–46

Ruf J, Wang W. 2022. Hedging with linear regressions and neural networks. *J. Bus. Econ. Stat.* 40(4):1442–54

Ruppert D. 2004. *Statistics and Finance: An Introduction*. New York: Springer

Sirignano J, Spiliopoulos K. 2018. DGM: a deep learning algorithm for solving partial differential equations. *J. Comput. Phys.* 375:1339–64

Tompaidis S, Yang C. 2013. Pricing American-style options by Monte Carlo simulation: alternatives to ordinary least squares. *J. Comput. Finance* 18(1):121–43

Tsitsiklis JN, van Roy B. 2001. Regression methods for pricing complex American-style options. *IEEE Trans. Neural Netw.* 12(4):694–703

Whalley AE, Wilmott P. 1997. An asymptotic analysis of an optimal hedging model for option pricing with transaction costs. *Math. Finance* 7(3):307–24

Yang Y, Zheng Y, Hospedales T. 2017. Gated neural networks for option pricing: Rationality by design. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ed. S Singh, S Markovitch, pp. 52–58. Palo Alto: AAAI Press

Zanger DZ. 2018. Convergence of a least-squares Monte Carlo algorithm for American option pricing with dependent sample data. *Math. Finance* 28(1):447–79

Zhang R, Langrené N, Tian Y, Zhu Z, Klebaner F, Hamza K. 2019. Dynamic portfolio optimization with liquidity cost and market impact: a simulation-and-regression approach. *Quant. Finance* 19(3):519–32

Zheng Y, Yang Y, Chen B. 2021. Incorporating prior financial domain knowledge into neural networks for implied volatility surface prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 3968–75. New York: Assoc. Comput. Mach.