Quantitative Finance

ISSN: 1469-7688 (Print) 1469-7696 (Online) Journal homepage: https://www.tandfonline.com/loi/rquf20

Machine learning for quantitative finance: fast derivative pricing, hedging and fitting

Jan De Spiegeleer, Dilip B. Madan, Sofie Reyners & Wim Schoutens

To cite this article: Jan De Spiegeleer, Dilip B. Madan, Sofie Reyners & Wim Schoutens (2018) Machine learning for quantitative finance: fast derivative pricing, hedging and fitting, Quantitative Finance, 18:10, 1635-1643, DOI: 10.1080/14697688.2018.1495335

To link to this article: https://doi.org/10.1080/14697688.2018.1495335

Quantitative Finance, 2018
Vol. 18, No. 10, 1635–1643, https://doi.org/10.1080/14697688.2018.1495335

Machine learning for quantitative finance: fast derivative pricing, hedging and fitting

JAN DE SPIEGELEER†, DILIP B. MADAN ‡, SOFIE REYNERS§ and WIM SCHOUTENS*§

†Risk Concile, B-3001 Leuven, Kapeldreef 60, Belgium
‡Robert H. Smith School of Business University of Maryland College Park, MD. 20742 College Park, MD, USA
§Department of Mathematics, University of Leuven, Celestijnenlaan 200B, B-3001 Leuven, Belgium

(*Received 8 June 2018; accepted 18 June 2018; published online 27 July 2018*)

In this paper, we show how we can deploy machine learning techniques in the context of traditional quant problems. We illustrate that for many classical problems, we can arrive at speed-ups of several orders of magnitude by deploying machine learning techniques based on Gaussian process regression. The price we have to pay for this extra speed is some loss of accuracy. However, we show that this reduced accuracy is often well within reasonable limits and hence very acceptable from a practical point of view. The concrete examples concern fitting and estimation. In the fitting context, we fit sophisticated Greek profiles and summarize implied volatility surfaces. In the estimation context, we reduce computation times for the calculation of vanilla option values under advanced models, the pricing of American options and the pricing of exotic options under models beyond the Black–Scholes setting.

Keywords: Machine learning; Gaussian processes; Derivative pricing; Hedging; Volatility surface

1. Introduction

In the derivatives world, daily zillions of computations need to be done. Models need to be calibrated. Derivative instruments need to be priced. Hedge positions need to be calculated. Risk management indicators need to be determined.

Many of these calculations deliver information which is moreover only useful for a limited period of time. If time ticks or markets move, many of the calculated values are outdated and need to be updated. In fact, real-time updates and information can be crucial in successfully running a derivatives business. Hence it is critical that the computation time needed for all these calculations be as small as possible. Since models and instruments have become more and more complex, this is not always trivial and one often has to rely on time-consuming techniques, like Monte Carlo simulation. Although one has in the last decades invested lots of research in speeding up the calculations, many institutions still face the limitations. Even if a calculation is possible in a fraction of a second, say the de-Americanization of an American option value, it can still be problematic if these calculations need to be repeated almost continuously on thousands of option prices and thousands of underliers. Real-time values are hence not always feasible.

On the other hand, many of the actual calculations are often very similar. The instruments to be priced can be summarized in a few parameters, like a strike, a barrier, a maturity, etc. Further, the market is often modelled with also a limited set of parameters (interest rate, dividend yield, some model parameters, for example, the five parameters of the Heston model). This context is actually ideal for machine learning techniques. Since the input parameters are of a limited dimension, we hence can summarize all this information in a training set, to allow the machine to learn the actual result function. This output is moreover often just a single real number, a price, a hedge parameter or any other output variable.

In this paper, we show how we can deploy machine learning techniques in the context of traditional quant problems. We illustrate that for many classical problems, we can arrive at speed-ups of several orders of magnitude by deploying machine learning techniques based on Gaussian process regression (GPR). The price we have to pay for this extra speed is some loss of accuracy. However, we show that this reduced accuracy is often well within reasonable limits and hence very acceptable from a practical point of view.

To be concrete, we first employ the GPR techniques for learning machines for fitting purposes. We start with showing the strengths of the method with fitting a non-trivial Gamma profile. Next, we illustrate the fitting ability by letting the machine learn the implied volatility surface of a given

*Corresponding author. Email: wim.schoutens@kuleuven.be

underlier on a given day. As a second line of applications, we apply the techniques in the setting of the pricing of exotic derivatives under advanced models. We illustrate how one can speed up time-consuming calculations with factors of several magnitudes. In particular we show how American option prices, exotic option prices, hedge parameters can be learned accurately. The technique can also be used to speed up calibration, to audit mark-to-market structured products, or to de-Americanize American option values, to name only a few other applications.

2. GPR explained

A short review of GPR is presented, for an extended treatment we refer to Rasmussen and Williams (2006). Consider a (training) set of observations $(X,\boldsymbol{y}) = \{(\boldsymbol{x_i}, y_i) \mid i = 1,\ldots,n\}$ where each $\boldsymbol{x_i}$ is an input vector of dimension d and y_i is the corresponding output. The goal is to find the relation between inputs and outputs. In GPR this is done by assuming

$$y_i = f(\boldsymbol{x_i}) + \varepsilon_i, \tag{1}$$

where $f(\boldsymbol{x})$ is a Gaussian process and $\varepsilon_i \sim \mathcal{N}(0,\sigma_n^2)$ are i.i.d. random variables representing the noise in the data.

A Gaussian process $f(\boldsymbol{x})$ is a, possibly infinite, collection of random variables, any finite subset of it having a joint Gaussian distribution. The process $f(\boldsymbol{x})$ is completely defined by its mean function $m(\boldsymbol{x})$ and a covariance or kernel function $k(\boldsymbol{x},\boldsymbol{x}')$. One often assumes a zero mean function. For a sample $(X,\boldsymbol{f}) = \{(\boldsymbol{x_i}, f_i) \mid i = 1,\ldots,n\}$ generated from $f(\boldsymbol{x})$, we then have

$$\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{0}, K(X,X)), \tag{2}$$

where $K(X,X)$ is the covariance matrix, defined as

$$K(X,X) = \begin{bmatrix} k(\boldsymbol{x_1},\boldsymbol{x_1}) & \cdots & k(\boldsymbol{x_1},\boldsymbol{x_n}) \\ k(\boldsymbol{x_2},\boldsymbol{x_1}) & \cdots & k(\boldsymbol{x_2},\boldsymbol{x_n}) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x_n},\boldsymbol{x_1}) & \cdots & k(\boldsymbol{x_n},\boldsymbol{x_n}) \end{bmatrix}. \tag{3}$$

GPR is a Bayesian method, as it starts from a prior Gaussian process to impose initial knowledge about the function f (e.g. prefer smooth functions) and then combines this with the observed data points $(X,\boldsymbol{y})$, leading to a posterior distribution of functions. Mathematically, this boils down to conditioning a joint distribution on the observations.

Consider a matrix X_* consisting of n_* test inputs, each of dimension d, and denote the corresponding (unknown) vector of function values by $\boldsymbol{f_*}$. Then the joint distribution of training outputs $\boldsymbol{y}$ and function values $\boldsymbol{f_*}$ is Gaussian,

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f_*} \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} K(X,X)+\sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right). \tag{4}$$

Note that $\boldsymbol{y}$ can be replaced by $\boldsymbol{f}$ when the noise $\sigma_n^2 = 0$. This happens when we have 'clean' observations of the underlying process f. To evaluate the GP posterior in the test inputs, the joint distribution (4) is conditioned on the observations $\boldsymbol{y}$,

$$\begin{aligned} \boldsymbol{f_*} \mid X_*, X, \boldsymbol{y} \sim \mathcal{N}\big(&K(X_*,X)\left[K(X,X)+\sigma_n^2 I\right]^{-1}\boldsymbol{y}, \\ &K(X_*,X_*) - K(X_*,X)\left[K(X,X)+\sigma_n^2 I\right]^{-1}K(X,X_*)\big) \end{aligned} \tag{5}$$

and point predictions correspond to the mean of this distribution.

A commonly used covariance function is the squared exponential (SE) kernel,

$$\begin{aligned} k(\boldsymbol{x},\boldsymbol{x}') &= \sigma_f^2 \exp\left(\frac{-|\boldsymbol{x}-\boldsymbol{x}'|^2}{2l^2}\right) \\ &= \sigma_f^2 \exp\left(\frac{-\sum_{k=1}^{d}(x_k - x_k')^2}{2l^2}\right) \end{aligned} \tag{6}$$

with σ_f^2 the highest possible covariance and l a length-scale parameter that determines the smoothness of the fit. The above kernel is a special case of the more general radial basis function

$$k(\boldsymbol{x},\boldsymbol{x}') = \sigma_f^2 \exp\left(\frac{-\sum_{k=1}^{d}|x_k - x_k'|^p}{2l_k^2}\right). \tag{7}$$

The parameters, σ_f^2 and l, in the former and σ_f^2, p and l_k in the latter kernel function are called hyperparameters, and need to be estimated from the training data, e.g. by a maximum likelihood estimation. A common approach is to obtain them by maximizing the marginal (log-)likelihood:

$$-\tfrac{1}{2}\log(\det(K(X,X))) - \tfrac{1}{2}\boldsymbol{y}^{\mathrm{T}}K(X,X)^{-1}\boldsymbol{y} + \text{constant}. \tag{8}$$

The mean function is often taken equal to zero, but can be modelled using a set of basis functions. Consider

$$g(\boldsymbol{x}) = f(\boldsymbol{x}) + h(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\beta}, \tag{9}$$

where $f(\boldsymbol{x})$ is a Gaussian process with zero mean function and kernel function $k(\boldsymbol{x},\boldsymbol{x}')$, $h(\boldsymbol{x})$ is a set of fixed basis functions and $\boldsymbol{\beta}$ are the corresponding coefficients that are estimated from the data. In this way we consider the model as a linear model, where the residuals are modelled via a Gaussian process. Frequently used sets of basis functions are $h(\boldsymbol{x}) = (\boldsymbol{1},\ \boldsymbol{x})$ and $h(\boldsymbol{x}) = (\boldsymbol{1},\ \boldsymbol{x},\ \boldsymbol{x}^2)$.

In this paper we will always use an SE kernel function and quadratic basis functions, unless specified otherwise.

3. Applications of GPR in quantitative finance

In a nutshell the whole procedure works as follows. For a large grid of input parameters (reflecting the derivative and the market situation), option prices and sensitivities such as Gamma, Vega, ... are stored in a repository. These values can just come from market data or can be estimated/calculated under the given model using existing techniques. From this data structure a dedicated covariance matrix K is inferred.

Figure 1. Classical approaches for fitting Γ. (a) Analytical model, (b) grid solution and (c) regression model.

Once this matrix K is calculated and inverted†, the training of the model is accomplished. After applying some GPR-related algebra to this matrix, any new calculation regarding the derivative structure is executed extremely fast. For most of the applications below, we just applied the standard GPR fitting algorithms, as available in the Matlab toolbox.

3.1. Fitting

3.1.1. An illustrative example. Let us start with an illustrative simple example. Consider a long position in a cliquet call option or a position in a call spread, for which we want to study the Gamma (Γ) as a function of the share price S. As illustrated in figure 1, the Gamma of these types of derivatives is double-signed. When the share price increases, Gamma flips from positive to negative.

Cliquet options are often priced with a Heston model, emphasizing the need of a stochastic volatility model to deal with some intrinsic (forward starting) features. Hence the smooth representation of Γ for a range of stock price values S (in figure 1(a)) requires a large computational workload. As a quick fix, one often proposes two classical alternative approaches. The most simple approach is a grid solution. For a selection of share price values $S \in \{a, b, c, d\}$, the value of $\Gamma(S)$ is calculated. These calculations take place overnight and the results are stored in a repository. Intraday, the value of $\hat{\Gamma}(S)$ is obtained via linear interpolation in the grid. Figure 1(b) illustrates the bias between the estimate and the analytical solution. Using a grid with a higher density can of course improve results, but this comes with an additional computational cost. Alternatively, a regression model can be deployed. Suppose we are using polynomials up to order 3 to replicate the analytical solution with a traditional linear regression model. Estimated outputs are obtained by the regression model as:

$$\hat{\Gamma}(S_i) = w_0 + w_1 S_i + w_2 S_i^2 + w_3 S_i^3. \tag{10}$$

The bias of using this approach is illustrated in figure 1(c).

Next, we show what a machine learning technique such as GPR could achieve using the same inputs. Figure 2 shows a fit on the same 4 points as before. A further increase to 6 data points leads to an even better fit.

So far the share price S was the only variable input parameter, we will now extend it to two dimensions. However, the general idea works for more input parameters as we will illustrate in other applications below. We first construct a training set by taking different combinations of the share price S and the maturity T, visualized by the grid in figure 3(a). For each crossing point in the grid, we calculate the analytical value of Γ. The GPR model now fits the relation between the set of inputs (S_i, T_i) and the corresponding outputs Γ_i. Note that the training is the only time-consuming component of GPR, but it has to be performed only once for each application. Once trained, the GPR model can be used to calculate Gamma values for unobserved combinations of S and T. By way of an example, consider the point $(S, T) = (110, 0.3)$. The GPR model predicts a Gamma of -64.0186814394502, while the analytical value is equal to -64.01868145719564. The model hence makes an absolute error of $1.7745\text{e}-08$.

To evaluate the model's performance on a larger scale, a test set is constructed by taking a collection of new combinations as presented by the grid in figure 3(c). Note that we have slightly reduced the input ranges, since GPR performs worse on the boundaries of the training set. It is hence recommended to make the ranges of the training set slightly larger than the ranges of the actual estimation application. In figure 4 the absolute errors corresponding to the predictions on the test set are visualized.

† Matrix inversion is often implemented via a Cholesky decomposition (Benoît 1924, Rasmussen and Williams 2006), which is more stable than actually inverting the matrix. For small matrices, i.e. small values of n, ordinary matrix inversion can be performed. For the results in this paper we used the Matlab functions fitrgp and predict. However if the dimension increases, special techniques need to be deployed. We mention LU-factorization and blockwise Cholesky decomposition, which aim at solving traditional memory problems that one encounters when inverting large matrices. For future work we will employ Cholesky and blockwise Cholesky routines to handle problems with many more data points.

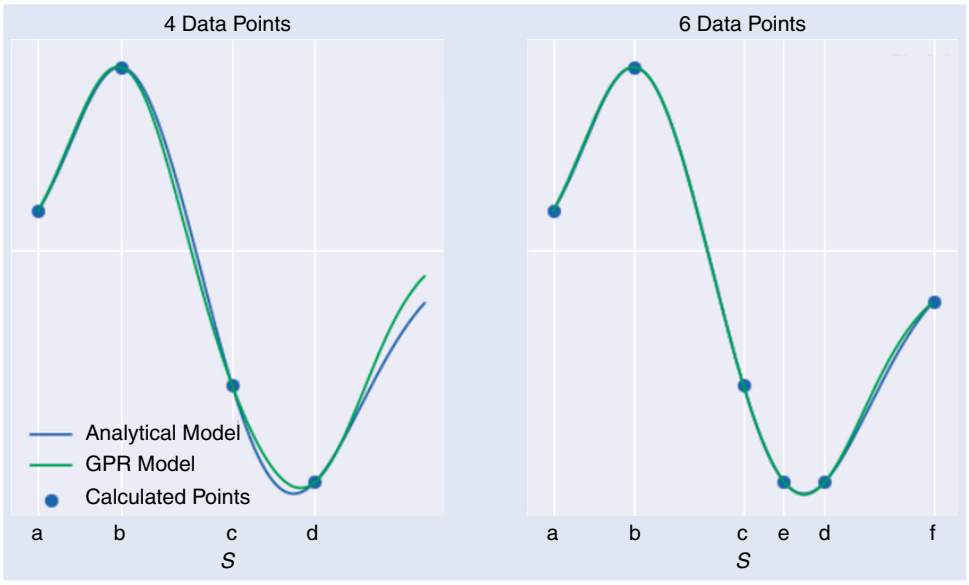

Figure 2. GPR fit for Γ.

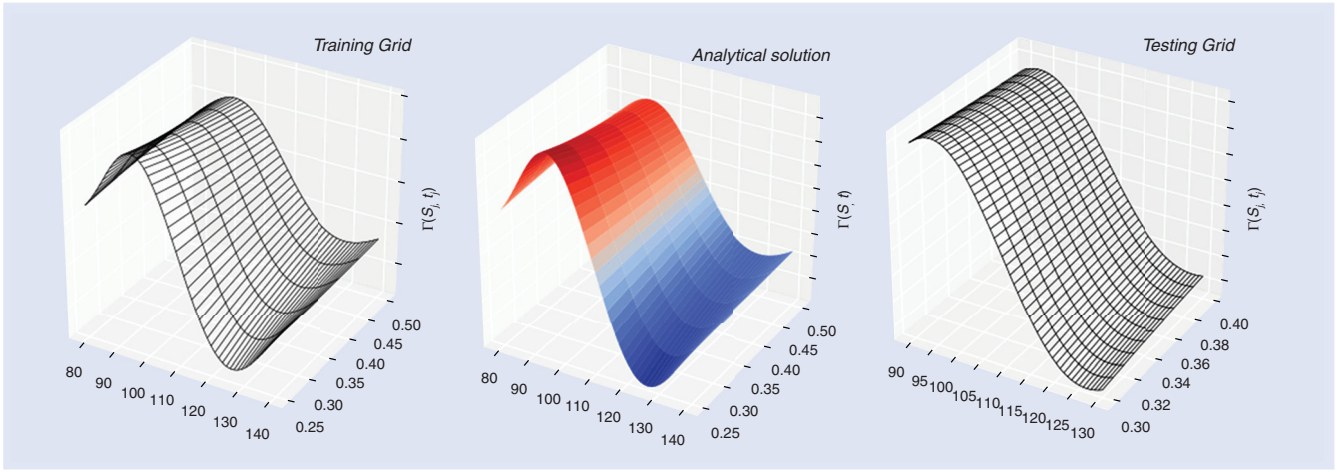

Figure 3. GPR training and validation for Γ in two dimensions. (a) Training set, (b) analytical values and (c) test set.

3.1.2. Summarizing volatility surfaces. Next, we illustrate the fitting capabilities of GPR in the context of summarizing implied volatility surfaces. These obtained fits are particularly interesting in the backtesting of option trading strategies as they require the marking to market of all open positions on a regular basis. Hence, it is of interest to develop time series of implied volatilities on a fixed grid of moneyness and maturity as a dynamic summary of the option surface. We therefore illustrate how GPR techniques can build functions for each underlying asset and each date. We note that furthermore, such functions could also be used to identify derivatives of option prices with respect to strike and maturity that are needed in building local volatility models.

We recognize that unlike the other applications illustrated here, one is not now summarizing a mathematical function that was expensive to evaluate with a view towards enhancing the computation speed, but the focus is on fitting observed values and estimating unobserved value. Hence we have to first ask whether one expects the existence of such a function that one is asking the machine to learn. For a perspective on this question we note that in the absence of static arbitrage it is known that the observed option prices must be consistent with the existence of a one dimensional Markov martingale with option prices being discounted expected payouts under the martingale model (Carr and Madan 2005, Davis and Hobson 2007). Numerous option pricing models are parametrically parsimonious examples of such martingales. To the extent that such an underlying martingale model exists that ties together all the option prices at a specific point of calendar time, there is then an underlying mathematical function describing the option prices and by implication also the implied volatilities. It may be a complicated function requiring many parameters, but a function to be learned, nonetheless. In fact one could be tempted to argue that if GPR fails to adequately summarize the surface then perhaps a static arbitrage is present.

For many underlying assets and most days we now have up to a thousand options quoted with a wide range of strikes and maturities. The maturities range from a few days up to three years with over 10 maturities in many cases. Apart from summarizing the implied volatility surface, it is also useful to summarize the forward curve and the rate curve to

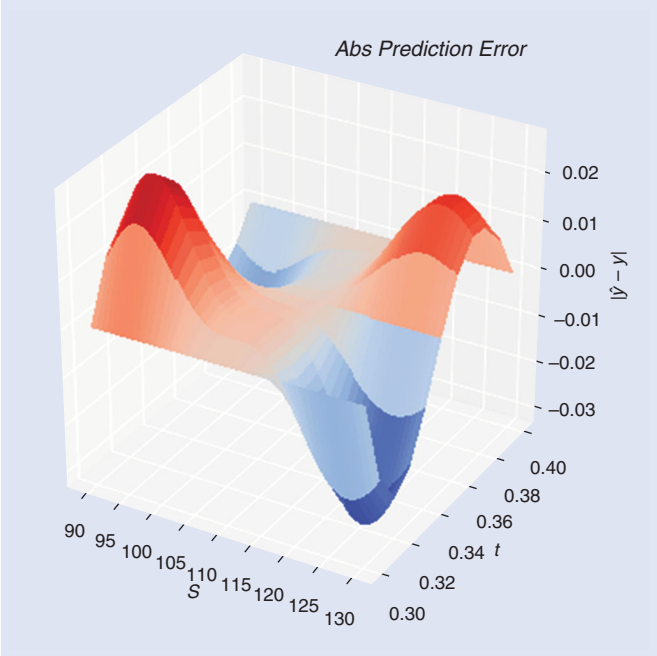

Figure 4. Absolute errors of GPR prediction.

extract forward prices and discount rates for arbitrary maturities. Hence, GPR was used to build three functions each day for some 2000 days and 200 underlying assets. We report on the results for S&P 500 index. The three functions deliver (i) the implied volatility as a function of the moneyness measured by the logarithm of the strike to the forward and the option's maturity, ii) the forward price as a function of maturity and (iii) the discount rate as a function of maturity. The GPR was trained each day for each underlier on all exchange traded quotes. The three functions are stored for each name and day.

By way of an illustration of the results we present the case of options on the S&P 500 index on 21 December 2017. There were 982 options in the training set covering 12 maturities. The kernel function (7) was used. The maximum absolute error between market and GPR predicted implied volatilities (in-sample) was 77 basis points. Figure 5 presents the market and GPR predicted implied volatilities at 6 maturities of 29, 85, 176, 365, 449, and 729 days.

Data without prior cleaning was submitted to the algorithm and resulted in general in very small errors. We calculated the maximum absolute error and the standard deviation of the residuals (residual standard error) across time from 4 January 2010 to 17 April 2018. Figure 6 presents these maximum and residual standard errors respectively, where the latter are expressed in basis points.

Note that the maximum error is representing the worst prediction in the complete dataset on the given date and is often just an outlier arising from far out of or in the money options where the market data input itself is rather uncertain or biased due for example to a large bid and ask spread. As one says, the proof of the pudding is in eating, therefore we put the obtained GPR fits to work in option strategy backtesting.

More precisely, the GPR based marking surfaces were employed to implement the daily sale of three-month 20, 30, 40, and 50 delta strangles on the S&P 500 index. The derivatives sold were of three-month maturities with the prescribed deltas and so they were not exchange traded, but the GPR marking function was used to infer their values and evaluate their deltas. Indeed, if time ticks or markets move, the issued options have no longer a 20, 30, 40 or 50% delta, but mark-to-market values and corresponding deltas need to be recalculated. It is here that GPR comes into its own. At maturity the payouts were made as they occurred on each option in the strangle. All open positions were marked to market using the GPR marking functions. Each day the portfolio of open positions was hedged to be delta neutral with the delta determined at the implied volatility for the option as it is predicted by the GPR marking function. Figure 7 shows the cumulated unhedged and unmarked cash flow, the cumulated unhedged

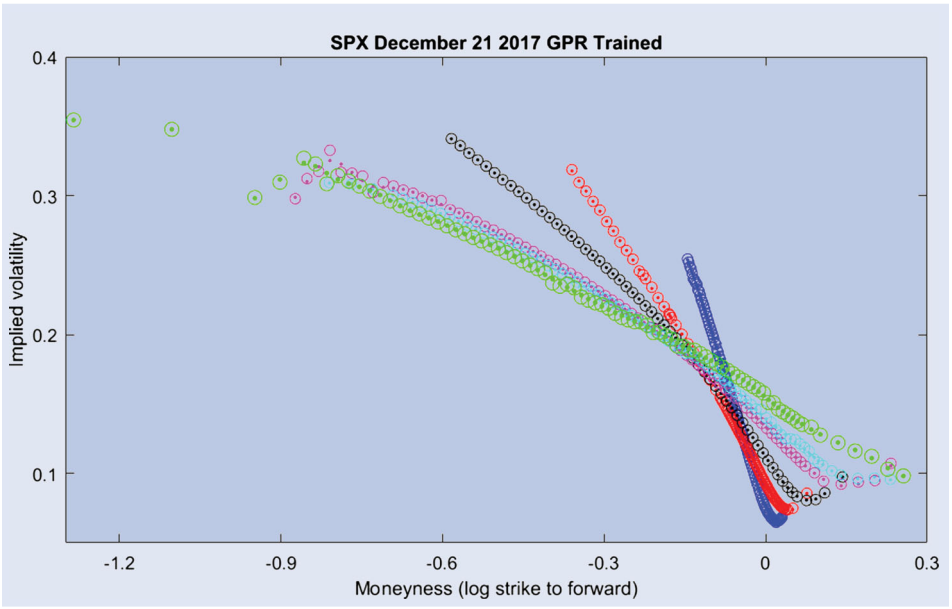

Figure 5. Market implied volatilities are displayed by circles while the GPR predictions are displayed as dots. The maturities of 29, 85, 176, 365, 449, and 729 days are displayed in the colours blue, red, black, magenta, cyan and green.

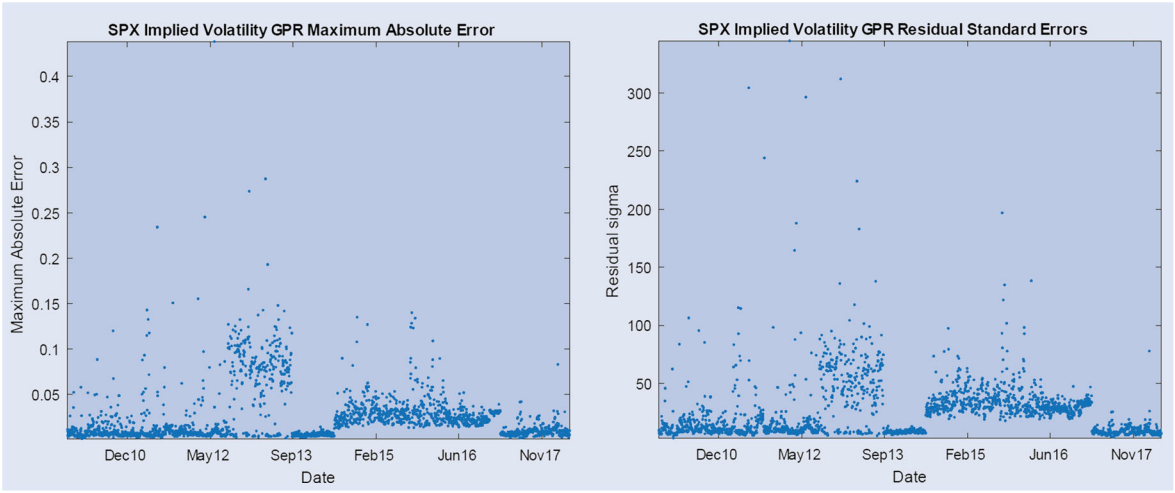

Figure 6. Maximum absolute error (left) and residual standard error in basis points (right) in GPR prediction of SPX implied volatilities across time from 4 January 2010 to 17 April 2018.

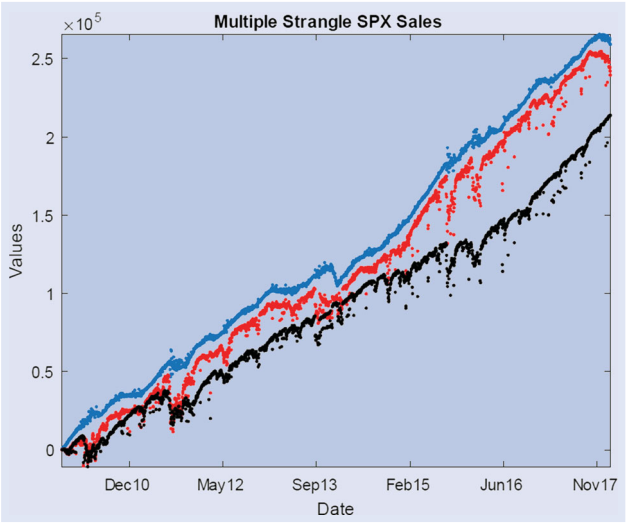

Figure 7. The cumulated unhedged and unmarked cash flow is displayed in blue. The red curve includes the marked to market value of open short positions. The black curve includes the hedged cash flows, delta hedged at the Black Merton Scholes implied volatilities as predicted by the GPR marking function.

cash flow plus mark-to-market value of open positions, and finally the cumulated hedged cash flow plus the mark-to-market value of open positions. The strategy was executed from 4 January 2010 to 11 January 2018.

3.2. Derivative pricing

3.2.1. European options. Plain vanilla options are used in multiple applications: as a building block of structured products, for calculating Greeks, for calibrating an advanced pricing model... A frequently used method for pricing them under advanced settings like the Heston model (Heston 1993) or Lévy models (Schoutens 2003) is the Fast Fourier Transform (FFT) algorithm (Carr and Madan 1999). Despite the 'fast' in its name, it still takes some time to price multiple options all at once. In this section we build a GPR model for the FFT-based European call price that predicts prices even faster. All experiments are performed for both the Variance Gamma model (Madan and Seneta 1990, Madan and Milne 1991, Madan *et al.* 1998) and the Heston model.

The modelling procedure starts by constructing a training set $(X, \mathbf{y})$. The input matrix X consists of n combinations of product, market and model parameters: strike price (K), maturity (T), interest rate (r), dividend yield (q) and the model-specific parameters, being σ, ν and θ for the Variance Gamma model and $\kappa, \rho, \theta, \eta$ and v_0 for the Heston model†. With a bit of effort, we can actually rewrite the problem with a limited set of parameters. Indeed the risk-neutral expected payoff of the call option is only dependent on the forward moneyness and the distribution at maturity. For the Variance Gamma model, we hence only need as inputs $(F, \tilde{\sigma}, \tilde{\nu}, \tilde{\theta})$, where F is the forward moneyness and $\tilde{\sigma} = \sigma\sqrt{T}$, $\tilde{\nu} = \nu/T$ and $\tilde{\theta} = \theta T$ are the rescaled VG parameters, corresponding to the distribution at maturity.

Random parameter combinations are sampled uniformly over the ranges given in table 1. Only for σ and v_0, we follow a slightly different procedure. These parameter values are partially sampled from a bounded exponential distribution, to incorporate more small volatilities in the training set, leading to better performance.

For each parameter combination, the price of the European call option (EC) is calculated by the FFT algorithm. The training set is then fed to the algorithm that fits the GPR model. Roughly speaking, this comes down to estimating hyperparameters and inverting an $n \times n$ matrix. Although the training set consists of clean outputs f_i, it is often recommended to let the algorithm assume noisy data and incorporate a strictly positive noise σ_n, leading to less situations where the matrix $K(X,X) + \sigma_n^2 I$ is (nearly) singular.

Once the learning procedure is finished, the model is ready for prediction. The pricing performance is measured in terms of maximum and average absolute errors,

$$\text{MAE} = \max\{|EC_{\text{FFT}}(i) - EC_{\text{GPR}}(i)|,\ i = 1, \ldots, n\} \quad (11)$$

† κ = rate of mean reversion, ρ = correlation stock – vol, θ = vol of vol, η = long run variance, v_0 = initial variance.

Table 1. Parameter ranges used for training and validation.

	Product/market	VG	Heston
Training set			
	$K : 40\% \to 160\%$	$\sigma : 0.05 \to 0.45$	$\kappa : 1.4 \to 2.6$
	$T : 11M \to 1Y$	$\nu : 0.55 \to 0.95$	$\rho : -0.85 \to -0.55$
	$r : 1.5\% \to 2.5\%$	$\theta : -0.35 \to -0.05$	$\theta : 0.45 \to 0.75$
	$q : 0\% \to 5\%$		$\eta : 0.01 \to 0.1$
			$v_0 : 0.01 \to 0.1$
Test set			
	$K : 50\% \to 150\%$	$\sigma : 0.1 \to 0.4$	$\kappa : 1.5 \to 2.5$
	$T : 11M \to 1Y$	$\nu : 0.6 \to 0.9$	$\rho : -0.8 \to -0.6$
	$r : 1.5\% \to 2.5\%$	$\theta : -0.3 \to -0.1$	$\theta : 0.5 \to 0.7$
	$q : 0\% \to 5\%$		$\eta : 0.02 \to 0.1$
			$v_0 : 0.02 \to 0.1$

$$\text{AAE} = \frac{1}{n}\sum_{i=1}^{n} |EC_{\text{FFT}}(i) - EC_{\text{GPR}}(i)| \tag{12}$$

for both the training set (in-sample prediction) and a test set (out-of-sample prediction). The latter is constructed by sampling uniformly new inputs over slightly smaller ranges, as specified in table 1.

A summary of the empirical results, both in terms of computation time and accuracy, is given in table 2. Note that the out-of-sample performance seems to be better than the in-sample performance. This is due to the slightly smaller parameter ranges in the test set. As expected, the larger the training set, the stronger the model and hence the more accurate the predictions. The main drawback is that the model becomes slower, leading to a trade-off between the computation time and the desired accuracy. Fortunately, there are more factors impacting the accuracy.

(1) The number of parameters (i.e. the dimension d). The lower this number, the better the results. This is why modelling VG-based prices is easier than modelling Heston-based prices.

(2) The width of the parameter ranges. Again, the smaller the ranges, the more dense the training set, the better the results. To achieve a higher accuracy rate, one can for instance work with different submodels based on maturity or moneyness.

Next, we scatter plot the prices obtained under the semi-analytical technique (FFT) versus the GPR obtained estimates. In a perfect estimation we would observe a straight line. In figure 8, we see that our scatter plot of (100 000) out-of-sample predictions of European option prices with a GPR model trained on 10 000 points hardly deviates from this straight line and this under both models.

3.2.2. American options. Pricing American options is often performed using slightly time-consuming tree methods, which order of complexity is quadratic in the number of time steps. We construct a GPR model for the price of an American put option (AP), according to the binomial tree model with time steps of one business day (1/250).

The model is fitted on a training set with input matrix $X = [\boldsymbol{K},\ \boldsymbol{T},\ \boldsymbol{r},\ \boldsymbol{q},\ \boldsymbol{\sigma}]$ and output vector $\boldsymbol{y} = \mathbf{AP}$, being the corresponding American option prices obtained via the binomial tree model. The training and test set are constructed by sampling the input parameters uniformly over the ranges in table 3. For the training set however, we included relatively more small values of σ, as explained previously.

The model's predictive performance is summarized in table 4, the corresponding scatter plot is provided in figure 9. The time needed to compute prices with the binomial tree model depends on the maturity of the considered option. For the GPR model however, maturity has no impact on the computation time. Speed-up will hence become even more spectacular when pricing long-term options.

Table 2. GPR performance with respect to the FFT-method, for differently sized training sets.

	VG			Heston		
Size of training set	5000	10 000	20 000	5000	10 000	20 000
In-sample prediction						
MAE	0.0065	0.0051	0.0046	0.0078	0.0086	0.0088
AAE	3.2771e − 04	2.5847e − 04	2.0492e − 04	5.3348e − 04	4.3611e − 04	3.6575e − 04
Out-of-sample prediction						
MAE	0.0028	0.0022	0.0016	0.0060	0.0054	0.0048
AAE	2.2508e − 04	1.6942e − 04	1.2828e − 04	5.8991e − 04	4.4112e − 04	3.6623e − 04
Speed-up	× 30	× 15	× 7	× 40	× 20	× 10

Notes: Out-of-sample predictions correspond to a test set of size 100 000. Speed-ups are obtained by comparing the CPU time for both methods.

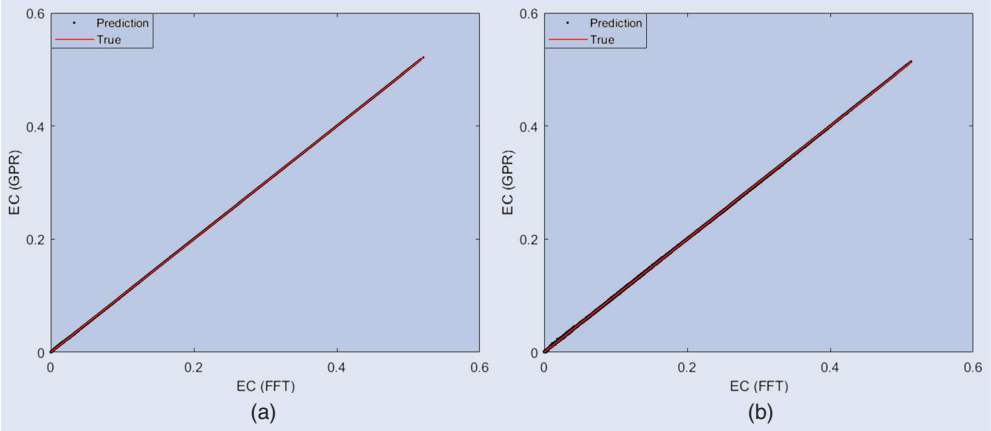

Figure 8. Out-of-sample prediction of European option prices (100 000 points) with a GPR model trained on 10 000 points. (a) Variance Gamma and (b) Heston.

Table 3. Parameter ranges used for training and validation.

Training set				
$K : 40\% \to 160\%$	$T : 11M \to 1Y$	$r : 1.5\% \to 2.5\%$	$q : 0\% \to 5\%$	$\sigma : 0.05 \to 0.55$
Test set				
$K : 50\% \to 150\%$	$T : 11M \to 1Y$	$r : 1.5\% \to 2.5\%$	$q : 0\% \to 5\%$	$\sigma : 0.1 \to 0.5$

Table 4. GPR's predictive performance for American put options, based on differently sized training sets.

Size of training set	5000	10 000	20 000
In-sample prediction			
MAE	0.0061	0.0040	0.0042
AAE	3.7840e − 04	3.3777e − 04	3.0124e − 04
Out-of-sample prediction			
MAE	0.0074	0.0086	0.0077
AAE	0.0012	9.1684e − 04	9.1573e − 04
Speed-up	× 137	× 70	× 33

Note: Out-of-sample predictions correspond to 100 000 test points.

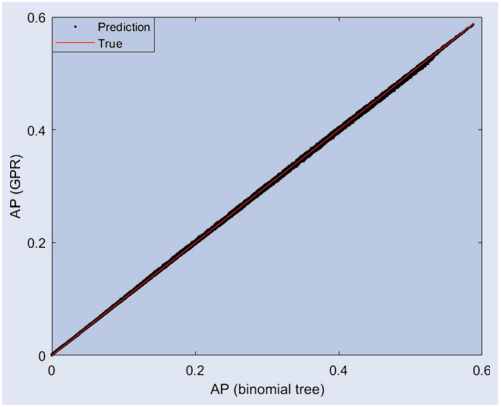

Figure 9. Out-of-sample prediction of American put option prices (100 000 points) with a GPR model trained on 10 000 points.

3.2.3. Barrier options. Under models beyond the Black–Scholes setting, barrier options are typically priced with a Monte Carlo procedure, since their payoff depends on whether or not the price of the underlying asset has crossed a predetermined barrier level H and no closed-form solutions exist. In this section, we focus on the price of a down-and-out barrier put option (DOBP), where the Heston model is used to describe the market. The Monte Carlo procedure is based on 100 000 price paths of the underlying, constructed by taking daily steps (1/250) up to maturity date.

We build a training set with inputs $X = [\boldsymbol{K}, \boldsymbol{H}, \boldsymbol{T}, \boldsymbol{r}, \boldsymbol{q}, \boldsymbol{\kappa}, \boldsymbol{\rho}, \boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{v_0}]$, by sampling uniformly (except for v_0) according to the ranges in table 5. The DOBP option prices are calculated† for each row in X, leading to the output vector $\boldsymbol{y}$.

We have reserved the best wine for the end of the party: while the pricing errors are relatively small, the speed-ups achieved are tremendous and are in the ranges of a factor of a couple of 1000 s. Predictive performance of the resulting GPR model is summarized in table 6, while the corresponding scatter plot is provided in figure 10.

4. Conclusion

In this paper, we have presented new machine learning applications for solving traditional quant problems, like curve fitting, derivative pricing and hedging. More precisely, we have employed a Bayesian non-parametric technique, called GPR. We focused on the fitting of sophisticated Greek curves and implied volatility surfaces. Also, we illustrated the use for pricing vanilla and exotic options under advanced models. Key in the procedure is the selection of an appropriate kernel function and a fast matrix inversion of the corresponding

† For each parameter combination, the same random numbers are used to construct the 100 000 Heston-based price paths of the underlying.

Table 5. Parameter ranges used for training and validation.

Training set				
$K : 40\% \to 160\%$	$H : 0.55 \to 0.99$	$T : 11M \to 1Y$	$r : 1.5\% \to 2.5\%$	$q : 0\% \to 5\%$
$\kappa : 1.4 \to 2.6$	$\rho : -0.85 \to -0.45$	$\theta : 0.35 \to 0.75$	$\eta : 0.01 \to 0.16$	$v_0 : 0.01 \to 0.16$
Test set				
$K : 50\% \to 150\%$	$H : 0.6 \to 0.95$	$T : 11M \to 1Y$	$r : 1.5\% \to 2.5\%$	$q : 0\% \to 5\%$
$\kappa : 1.5 \to 2.5$	$\rho : -0.8 \to -0.5$	$\theta : 0.4 \to 0.7$	$\eta : 0.02 \to 0.16$	$v_0 : 0.02 \to 0.16$

Table 6. GPR's predictive performance for down-and-out barrier put options, based on differently sized training sets.

Size of training set	5000	10 000	20 000
In-sample prediction			
MAE	0.0104	0.0148	0.0127
AAE	6.4718e – 04	4.8786e – 04	4.5981e – 04
Out-of-sample prediction			
MAE	0.0159	0.0086	0.0078
AAE	9.9330e – 04	6.7386e – 04	5.3261e – 04
Speed-up	× 11074	× 5850	× 2172

Note: Out-of-sample predictions correspond to 10 000 test points.

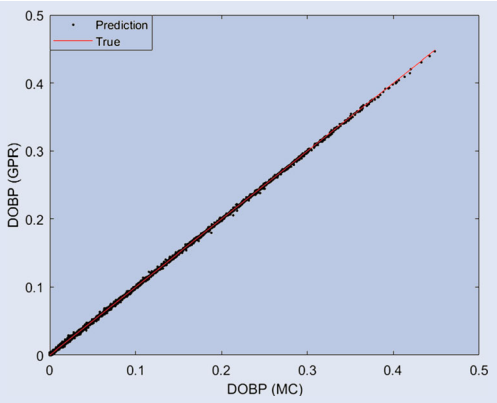

Figure 10. Out-of-sample prediction of down-and-out barrier put option prices (10 000 points) with a GPR model trained on 10 000 points.

covariance matrix. Using training sets of sizes around 10 000 and hence corresponding kernel matrices of 10 000 by 10 000, leads to a training procedure that can be executed in a reasonable amount of time. This training has to be performed just once. The resulting model can then be used for prediction (if parameters are still in the initial ranges of the training set), leading to speed-ups of several orders of magnitude. We have presented the techniques for a series of curves, products and models. We note that the same methodology can actually be applied for a much broader range of applications and that the above mentioned sizes can with the appropriate computer power and techniques (matrix inversion) be extended in order make the accuracy even better.

Disclosure statement

No potential conflict of interest was reported by the authors.

ORCID

Dilip B. Madan http://orcid.org/0000-0002-0033-9077

References

Benoît, Cdt., Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés à un système d'équations linéaires en nombre inférieur à celui des inconnues, (procédé du commandant cholesky). *Bull. Géodésique*, 1924, **2**, 67–77.

Carr, P. and Madan, D.B., Option valuation using the fast Fourier transform. *J. Comput. Finance*, 1999, **2**, 61–73.

Carr, P. and Madan, D.B., A note on sufficient conditions for no arbitrage. *Finance Res. Lett.*, 2005, **2**, 125–130.

Davis, M. and Hobson, D., The range of traded option prices. *Math. Finance*, 2007, **17**, 1–14.

Heston, S.L., A closed form solution for options with stochastic volatility with applications to bonds and currency options. *Rev. Finan. Stud.*, 1993, **6**, 327–343.

Madan, D.B. and Milne, F., Option pricing with V.G. martingale components. *Math. Finance*, 1991, **1**, 39–55.

Madan, D.B. and Seneta, E., The V.G. model for share market returns. *J. Bus.*, 1990, **63**, 511–524.

Madan, D.B., Carr, P. and Chang, E.C., The variance Gamma process and option pricing. *Eur. Finan. Rev.*, 1998, **2**, 79–105.

Rasmussen, C.E. and Williams, C.K.I., *Gaussian Processes for Machine Learning*, 2006 (MIT Press: Cambridge, MA).

Schoutens, W., *Lévy Processes in Finance: Pricing Financial Derivatives*, 2003 (Wiley: New York).