Machine Learning for Pricing American Options in High Dimension

Ludovic Goudenège[*†]

Andrea Molent[‡§]

Antonino Zanette[¶]

Abstract

In this paper we propose an efficient method to compute the price of American basket options, based on Machine Learning and Monte Carlo simulations. Specifically, the options we consider are written on a basket of assets, each of them following a Black-Scholes dynamics. The method we propose is a backward dynamic programming algorithm which considers a finite number of uniformly distributed exercise dates. On these dates, the value of the option is computed as the maximum between the exercise value and the continuation value, which is approximated via Gaussian Process Regression. Specifically, we consider a finite number of points, each of them representing the values reached by the underlying at a certain time. First of all, we compute the continuation value only for these points by means of Monte Carlo simulations and then we employ Gaussian Process Regression to approximate the whole continuation value function. Numerical tests show that the algorithm is fast and reliable and it can handle also American options on very large baskets of assets, overcoming the problem of the curse of dimensionality.

Keywords: Machine Learning, Gaussian Process Regression, American options, Monte Carlo methods

1 Introduction

In this paper we consider one of the most compelling problems among the still open issues in the field of computational finance: pricing and hedging American options in high dimension. From a practical point of view, the efficient numerical evaluation of American options which consider as underlying a baskets of d assets is a very challenge because of the so-called "curse of dimensionality" which avoids the direct application of standard numerical schemes such as finite difference or tree methods. Specifically, this curse of dimensionality

[*]This work was supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH.

[†]Fédération de Mathématiques de CentraleSupélec - CNRS FR3487, France - ludovic.goudenege@math.cnrs.fr

[‡]This work was supported by the Chair of Quantitative Finance of FiQuant at CentraleSupélec - MICS (EA 4037), France.

[§]Dipartimento di Scienze Economiche e Statistiche, Università degli Studi di Udine, Italy - andrea.molent@uniud.it

[¶]Dipartimento di Scienze Economiche e Statistiche, Università degli Studi di Udine, Italy - antonino.zanette@uniud.it

means that the computational cost and the memory requirement increase exponentially with the dimension of the problem.
Given these conditions, Monte Carlo methods appears as the only feasible approach. Several new ideas have appeared in this research area, which can be divided into four groups. The first type of approach consists in employing a recombinant tree in order to obtain a discretization of the underlying diffusion. An example of this approach is given by the stochastic mesh method of Broadie and Glasserman [6], the quantization algorithms of Bally, Pagès and Printems [3], the stochastic grid method of Jain and Oosterlee [11]. The second idea makes use of regression on a truncated basis of L^2 in order to compute the conditional expectations. This is done in Longstaff and Schwartz [15] and in Tsisiklis and Van Roy [17]. The third idea consists in exploiting the representation formulas for the conditional expectation using Malliavin calculus. This has been done by Lions and Reigner [14], Bouchard and Touzi [4], Bally, Caramellino and Zanette [2] and Abbas-Turki and Lapeyre[1]. Finally, the last group consists of duality-based approaches for Bermudan option pricing, which are proposed by Rogers [16], Haugh and Kogan [10], and Lelong [13], which can be used to construct bounds on the option value.

In this paper, we propose a new method that couples Machine Learning techniques and Monte Carlo methods. Machine Learning has recently been applied in the field of derivative pricing. For example, it has been proposed by De Spiegeleer et al. [8] to predict the price of the derivatives from a training set made of observed prices for particular combinations of model parameters. Specifically, they consider Gaussian Process Regression (GPR) which is a Bayesian non-parametric technique that allows one to make accurate predictions very quickly. Afterwards, the same technique has been applied within the Variable Annuities to the scope of pricing and Greeks calculation by Goudenège et al. [9]. In this case, the number of parameters that affect the price function, and thus the dimension of the interpolation domain, is high and can exceed ten.

Since GPR can successfully be employed to approximate the price function over a high dimensional domain from sparse observed points, we propose to employ it in a time step algorithm to solve the stochastic control problem associated with the American options. The algorithm we propose proceeds backward over time and it computes the price function on a set of predetermined points given by possible values of the underlying. In particular, at each time step, we use a set of Monte Carlo simulations together with GPR to approximate the continuation value at these points. The option price is then obtained as the maximum between the continuation value and the intrinsic value of the option. For the sake of simplicity, we name this new approach Gaussian Process Regression - Monte Carlo method (GPR-MC).

The paper is organized as follows. In Section 2 we present American options for the Black-Scholes d-dimensional model. In Section 3 we briefly review the GPR method and we explain how to apply it for the American options evaluation scope. In Section 4 we report the numerical results. Finally, Section 5 draws the conclusions.

2 American options in the multi-dimensional Black-Scholes model

An American option with maturity T is a derivative instrument whose holder can exercise the intrinsic optionality at any moment, from inception up to maturity. Let $\mathbf{S} = (\mathbf{S}_t)_{t\in[0,T]}$ denote the d-dimensional underlying process. Such a stochastic process is assumed to randomly evolve according to the multidimensional Black-Scholes model: under the risk neutral measure, such a model is given by the following equation

$$dS_t^i = r\, S_t^i\, dt + \sigma_i\, S_t^i\, dW_t^i, \qquad i = 1, \dots, d, \tag{2.1}$$

with $\mathbf{S}_0 = (s_{0,1}, \ldots, s_{0,d}) \in \mathbb{R}_+^d$ the spot price, r the (constant) spot rate, $\sigma = (\sigma_1, \ldots, \sigma_d)$ the vector of volatilities, $\mathbf{W}$ a d-dimensional correlated Brownian motion and ρ_{ij} the instantaneous correlation coefficient between W_t^i and W_t^j. Moreover, let $\Psi(\mathbf{S}_T)$ denote the cash-flow associated with the option. Thus, the price at time t of an American option having maturity T and payoff function $\Psi : \mathbb{R}_+^d \to \mathbb{R}$ is then

$$v(t, \mathbf{x}) = \sup_{\tau \in \mathcal{T}_{t,T}} \mathbb{E}_{t,\mathbf{x}} \left[e^{-r(\tau - t)} \Psi(\mathbf{S}_\tau) \right], \tag{2.2}$$

where $\mathcal{T}_{t,T}$ stands for the set of all the stopping times taking values on $[t, T]$ and $\mathbb{E}_{t,\mathbf{x}}[\cdot]$ is the expectation given all the information at time t and assuming $\mathbf{S}_t = \mathbf{x}$.

For simulation purposes, the $d-$dimensional Black-Scholes model can be written alternatively using the Cholesky decomposition. Specifically, for $i \in \{1, \ldots, d\}$ we can write

$$dS_t^i = S_t^i (r dt + \sigma_i \Sigma_i d\mathbf{B}_t), \tag{2.3}$$

where $\mathbf{B}$ is a d-dimensional Brownian motion and Σ_i is the i-th row of the matrix Σ defined as a square root of the correlation matrix Γ, given by

$$\Gamma = \begin{pmatrix} 1 & \rho_{12} & \ldots & \rho_{1d} \\ \rho_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \rho_{d1} & \ldots & \ldots & 1 \end{pmatrix}$$

3 Machine Learning for American options in the multi-dimensional Black-Scholes model

3.1 Gaussian Process Regression

In this Section, we present a brief review of Gaussian Process Regression and for a comprehensive treatment we refer to Rasmussen and Williams [18].

Gaussian Process Regression (also known as GPR) is a class of non-parametric kernel-based probabilistic models which represents the input data as the random observations of a Gaussian stochastic process. The most important advantage of this approach in relation to other parametric regression techniques is that it is possible to effectively exploit a complex dataset which may consist of points sampled randomly in a multidimensional space.

In general, a Gaussian process $\mathcal{G}$ is a collection of random variables defined on a common probability space $(\Omega, \mathcal{F}, P)$, any finite number of which have consistent joint Gaussian distributions. We are interested in Gaussian processes for which the random variables in $\mathcal{G}$ are indexed by a point $\mathbf{x} \in \mathbb{R}^d$, $d \in \mathbb{N}$. Therefore, for all $\mathbf{x} \in \mathbb{R}^d$, $\mathcal{G}(\mathbf{x}) : \Omega \to \mathbb{R}$ is a Gaussian random variable and if $X = \{\mathbf{x}_p, p = 1, \ldots, P\} \subset \mathbb{R}^d$ then $(\mathcal{G}(\mathbf{x}_1), \ldots, \mathcal{G}(\mathbf{x}_P))^\top$ is a random Gaussian vector. Moreover, a Gaussian process is fully specified by its mean function $\mu(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$ and by its covariance function $k(\mathbf{x}, \mathbf{x}') : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$.

Now, let us consider a training set $\mathcal{D}$ of P observations (the input data), $\mathcal{D} = \{(\mathbf{x}_p, y_p), p = 1, \ldots, P\}$ where $X = \{\mathbf{x}_p, p = 1, \ldots, P\} \subset \mathbb{R}^d$ denotes the set of input vectors and $Y = \{y_p, p = 1, \ldots, P\} \subset \mathbb{R}$ denotes the set of scalar outputs. These observations are modeled as the realization of the sum of a Gaussian process and a noise source. Specifically,

$$y_p = f_p + \varepsilon_p \tag{3.1}$$

where $\{f_p = \mathcal{G}(\mathbf{x}_p), p = 1, \ldots, P\}$ is a Gaussian process and $\{\varepsilon_p, p = 1, \ldots, P\}$ are i.i.d. random variables such that $\varepsilon_p \sim \mathcal{N}\left(0, \sigma_P^2\right)$. Moreover, the distribution of $\mathbf{f} = (f_1 \ldots f_P)$ is assumed to be given by

$$\mathbf{f} \sim \mathcal{N}(\mathbf{0}, K(X, X)), \tag{3.2}$$

where $K(X, X)$ is a $P \times P$ matrix with $K(X, X)_{p_1, p_2} = k(\mathbf{x}_{p_1}, \mathbf{x}_{p_2})$ for $p_1, p_2 = 1, \ldots, P$. Thus

$$\mathbf{y} \sim \mathcal{N}\left(\mathbf{0}, K(X, X) + \sigma_P^2 I_P\right), \tag{3.3}$$

where I_P is the $P \times P$ identity matrix.

Now, in addition, let us consider a test set $\tilde{X}$ of M points $\{\tilde{\mathbf{x}}_m, m = 1, \ldots, M\}$. The realizations $\tilde{f}_m = \mathcal{G}(\tilde{\mathbf{x}}_m)$ are not known but rather we want to estimate them considering the observed realizations of $\mathcal{G}$ in $\mathcal{D}$. The *a priori* joint distribution of $\mathbf{y}$ and $\tilde{\mathbf{f}} = \left(\tilde{f}_1, \ldots, \tilde{f}_M\right)$ is given by

$$\begin{bmatrix} \mathbf{y} \\ \tilde{\mathbf{f}} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0}_P \\ \mathbf{0}_M \end{bmatrix}, \begin{bmatrix} K(X, X) + \sigma_P^2 I_P & K\left(X, \tilde{X}\right) \\ K\left(\tilde{X}, X\right) & K\left(\tilde{X}, \tilde{X}\right) \end{bmatrix}\right) \tag{3.4}$$

where $K\left(\tilde{X}, \tilde{X}\right)$ is a $M \times M$ matrix given by $K\left(\tilde{X}, \tilde{X}\right)_{m_1, m_2} = k(\tilde{\mathbf{x}}_{m_1}, \tilde{\mathbf{x}}_{m_2})$ for $m_1, m_2 = 1, \ldots, M$, $K\left(X, \tilde{X}\right)$ is a $P \times M$ matrix given by $K\left(X, \tilde{X}\right)_{p,m} = k(\mathbf{x}_p, \tilde{\mathbf{x}}_m)$ for $p = 1, \ldots, P$, $m = 1, \ldots, M$ and $K\left(\tilde{X}, X\right)$ is a $M \times P$ matrix given by $K\left(\tilde{X}, X\right)_{m,p} = k(\tilde{\mathbf{x}}_m, \mathbf{x}_p)$ for $m = 1, \ldots, M$, $p = 1, \ldots, P$.

Since we know the values for the training set, we can consider the conditional distribution of $\tilde{\mathbf{f}}$ given $\mathbf{y}$. It is possible to prove that $\tilde{\mathbf{f}}|\tilde{X}, \mathbf{y}, X$ follows a Gaussian distribution given by

$$\tilde{\mathbf{f}}|\tilde{X}, \mathbf{y}, X \sim \mathcal{N}\left(\mathbb{E}\left[\tilde{\mathbf{f}}|\tilde{X}, \mathbf{y}, X\right], Cov\left[\tilde{\mathbf{f}}|\tilde{X}, \mathbf{y}, X\right]\right), \tag{3.5}$$

where

$$\mathbb{E}\left[\tilde{\mathbf{f}}|\tilde{X}, \mathbf{y}, X\right] = K\left(\tilde{X}, X\right)\left[K(X, X) + \sigma_P^2 I_P\right]^{-1}\mathbf{y} \tag{3.6}$$

and

$$Cov\left[\tilde{\mathbf{f}}|\tilde{X}, \mathbf{y}, X\right] = K\left(\tilde{X}, \tilde{X}\right) - K\left(\tilde{X}, X\right)\left[K(X, X) + \sigma_P^2 I_P\right]^{-1} K\left(X, \tilde{X}\right). \tag{3.7}$$

Therefore, a natural choice consists in predicting the values $\tilde{\mathbf{f}}$ through $\mathbb{E}\left[\tilde{\mathbf{f}}|\tilde{X}, \mathbf{y}, X\right]$. Moreover, also confidence intervals for such a prediction can be computed.

The computation in (3.6) requires the knowledge of the covariance function K and of the noise variance σ_P^2. A typical used covariance function is the Matern 3/2 kernel, which is given by

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2\left(1 + \frac{\sqrt{3}\,\|\mathbf{x} - \mathbf{x}'\|_2}{\sigma_l}\right)\exp\left(-\frac{\sqrt{3}\,\|\mathbf{x} - \mathbf{x}'\|_2}{\sigma_l}\right) \tag{3.8}$$

where σ_f^2 is called the signal variance and σ_l is called the length-scale. The parameters σ_f^2, σ_l of the kernel function and σ_P^2 of the noise are called hyperparameters and need to be estimated. A common approach is to consider the maximum likelihood estimates which can be obtained by maximizing the log-likelihood function of the training data, that is by maximizing the following function:

$$-\frac{1}{2}\log\left(\det\left(K(X, X) + \sigma_P^2 I_P\right)\right) - \frac{1}{2}\mathbf{y}^\top\left[K(X, X) + \sigma_P^2 I_P\right]^{-1}\mathbf{y}. \tag{3.9}$$

The development of the GPR model can be divided in the training step and the evaluation step (also called testing step). The training step only requires the knowledge of the training set $\mathcal{D}$ and it consists

in estimating the hyperparameters and computing the matrix product $A = \left[K(X,X) + \sigma_P^2 I_P\right]^{-1} \mathbf{y}$. The evaluation step can be computed only after the training step has been accomplished and it consists in obtaining the predictions via the computation of $K\left(\tilde{X}, X\right) A$. We stress out that the training step is independent of the test set $\tilde{X}$. Thus one can store the values computed during the training step and perform the evaluation step many times with a small computational cost, which is $\mathcal{O}(P \cdot M)$.

Remark 1. *We observe that the computation time depends only marginally on the size d of the space where the points lie, as the value of d only impacts in the time taken to calculate distances between the points which appears in the covariance matrix K, that is* $\|\mathbf{x} - \mathbf{x}'\|_2$.

3.2 Machine Learning Monte Carlo algorithm for American options

We approximate the price of an American option with the price of a Bermudan option on the same basket. Specifically, let N be the number of time steps and $\Delta t = T/N$ the time increment. The discrete exercise dates are $t_n = n\,\Delta t$, as $n = 0, 1, \ldots, N$. If $\mathbf{x}$ represents the vector of the underlying prices at the exercise date t_n, then the value of the option is given by

$$v(t_n, \mathbf{x}) = \max\left(\Psi(\mathbf{x}), E_{t_n,\mathbf{x}}\left[e^{-r\Delta t} v\left(t_{n+1}, \mathbf{S}_{t_{n+1}}\right)\right]\right). \tag{3.10}$$

First of all, knowing the function $v(t_{n+1}, \cdot)$, we can compute $v(t_n, \cdot)$ by approximation of the expectation. In order to compute the expectation in formula (3.10), we consider a set X of P points whose coordinates represent certain possible values for the underlyings:

$$X = \{\mathbf{x}^p = (x_1^p, \ldots, x_d^p), p = 1, \ldots, P\} \subset \mathbb{R}^d. \tag{3.11}$$

Suppose now we want to compute $v(t_n, \mathbf{x}^p)$ but only for $\mathbf{x}^p \in X$. This is done by a one step Monte Carlo simulation. In particular, for each $\mathbf{x}^p \in X$, we simulate a set $\tilde{X}_p$

$$\tilde{X}^p = \{\tilde{\mathbf{x}}^{p,m} = (\tilde{x}_1^{p,m}, \ldots, \tilde{x}_d^{p,m}), m = 1, \ldots, M\} \subset \mathbb{R}^d \tag{3.12}$$

of M possible values for $\mathbf{S}_{t_{n+1}}$ according to $\mathbf{S}_{t_n} = \mathbf{x}^p$ and to equation (2.3). Then, the price function can be approximated for each $\mathbf{x}^p \in X$ by

$$\hat{v}(t_n, \mathbf{x}^p) = \max\left(\Psi(\mathbf{x}^p), \frac{e^{-r\Delta t}}{M} \sum_{m=1}^{M} v(t_{n+1}, \tilde{\mathbf{x}}^{p,m})\right), \tag{3.13}$$

if the quantities $v(t_{n+1}, \tilde{\mathbf{x}}^{p,m})$ are known for all these simulated points $\tilde{\mathbf{x}}^{p,m}$. If we proceed backward, the function $v(t, \cdot)$ is known for $t = T$ since it is equal to the payoff function $\Psi(\cdot)$ and thanks to (3.13) it is known, through an approximation, also for $t = t_{N-1}$ and $\mathbf{x}^p \in X$. In order to assess $v(t_{N-2}, \mathbf{x}^p)$ for all $\mathbf{x}^p \in X$, and thus going on up to $t = 0$, it is necessary to evaluate the function $v(t_{N-1}, \cdot)$ for all the points in $\tilde{X} = \bigcup_{p=1}^{P} \tilde{X}^p$. This cannot be done directly since we know $\hat{v}(t_{N-1}, \cdot)$ only for the points in X and not for all those in $\tilde{X}$. To overcome this issue, we perform the interpolation of the function $\hat{v}(t_{N-1}, \cdot)$ by means of the GPR method over X. More generally, let $\tilde{v}(t_n, \cdot)$ be the GPR prediction of $\hat{v}(t_n, \cdot)$. Then, we can proceed backward and for any $t_n < t_{N-1}$ we can compute

$$\hat{v}(t_n, \mathbf{x}^p) = \max\left(\Psi(\mathbf{x}^p), \frac{e^{-r\Delta t}}{M} \sum_{m=1}^{M} \tilde{v}(t_{n+1}, \tilde{\mathbf{x}}^{p,m})\right). \tag{3.14}$$

The choice of the set X is a sensitive question. Let H^p be the p-th point of the Halton quasi-random sequence in $\mathbb{R}^d$ and Φ^{-1} the inverse cumulative distribution of a standard normal distribution. Let us define $h(i,p,t)$ as follows:

$$h(i,p,t) = e^{\left(r-\frac{1}{2}\sigma_i^2\right)t+\sigma_i\sqrt{t}\sum_{j=1}^{d}\Sigma_{d,j}\Phi^{-1}\left(H_j^p\right)}, \tag{3.15}$$

for $i = 1,\dots,d$, $p = 1,\dots,P$ and $t \in \mathbb{R}_+$. In order to define X, for $i = 1,\dots,d$ and $p = 2,\dots,P$ we set

$$x_i^1 = s_{0,i}, \tag{3.16}$$

and

$$x_i^p = s_{0,i} \cdot h(i,p-1,T). \tag{3.17}$$

Similarly, we use the Halton sequence to compute the (quasi-) Monte Carlo simulations required by the algorithm, that is

$$\tilde{x}_i^{p,m} = x_i^p \cdot h(i,m,\Delta t). \tag{3.18}$$

In particular, the blue dots in Figure 3.1 are the points of the sets X and $\tilde{X}^1$ (for a given set of parameters) while the orange point is the initial spot.

The sketch of the backward algorithm is presented here.

Preprocessing: compute $\mathbf{x}^p$ and $\tilde{\mathbf{x}}^{p,m}$ by using equations (3.17) and (3.18)

Step $N-1$: shaping of $\tilde{v}(t_{N-1},\cdot)$:

- $\hookrightarrow$ For $p = 1,\dots,P$ compute $\hat{v}(t_{N-1},\mathbf{x}^p) = \max\left(\Psi(\mathbf{x}^p), \frac{e^{-r\Delta t}}{M}\sum_{m=1}^{M}\Psi(\tilde{\mathbf{x}}^{p,m})\right)$
- $\hookrightarrow$ Define the training set $\mathcal{D} = \{(\mathbf{x}^p, \hat{v}(t_{N-1},\mathbf{x}^p)), p = 1,\dots,P\}$
- $\hookrightarrow$ Apply GPR on $\mathcal{D}$ to obtain $\tilde{v}(t_{N-1},\cdot)$

Step $N-2$: shaping of $\tilde{v}(t_{N-2},\cdot)$:

- $\hookrightarrow$ For $p = 1,\dots,P$ compute $\hat{v}(t_{N-2},\mathbf{x}^p) = \max\left(\Psi(\mathbf{x}^p), \frac{e^{-r\Delta t}}{M}\sum_{m=1}^{M}\tilde{v}(t_{N-1},\tilde{\mathbf{x}}^{p,m})\right)$
- $\hookrightarrow$ Define the training set $\mathcal{D} = \{(\mathbf{x}^p, \hat{v}(t_{N-2},\mathbf{x}^p)), p = 1,\dots,P\}$
- $\hookrightarrow$ Apply GPR on $\mathcal{D}$ to obtain $\tilde{v}(t_{N-2},\cdot)$

$\vdots$ $\leftarrow$ Steps $n = N-3,\dots,1$ [replace $N-2$ with n and $N-1$ with $n+1$;]

Step 0: computation of the price:

$$\hat{v}(0,\mathbf{s}_0) = \max\left(\Psi\left(\mathbf{x}^1\right), \frac{e^{-r\Delta t}}{M}\sum_{m=1}^{M}\tilde{v}\left(t_{N-1},\tilde{\mathbf{x}}^{1,m}\right)\right)$$

Remark 2. *We remark that when using a quasi-Monte Carlo sequence it is important to consider leaping. This technique consists in considering only some uniformly subsampled points of the original sequence, which improves convergence. However, the leap values, must be chosen with care. In fact, many values lead to sequences that do not touch on large sub-hyper-rectangles of the unit hypercube, failing to be a uniform quasi-random point set (see Kocis and Whiten [12]). A common rule for choosing the leap values for the Halton sequence consists in setting the value to $q-1$, where q is a prime number that has not been used to generate the sequence.*

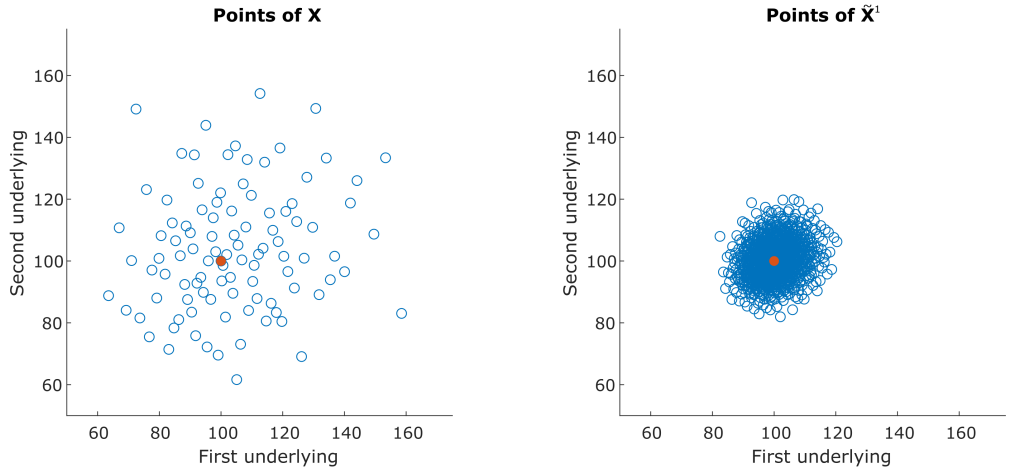

Figure 3.1: The sets X and $\tilde{X}^1$ for $d = 2$, $P = 100$, $M = 1000$ and $N = 10$.

Remark 3. *We observe that the GPR-MC method can be easily parallelized since the conditional expectations that appear in each Monte Carlo step of the algorithm are independent of each other and can be calculated separately. Thus, this feature allows one to significantly reduce the computational time.*

4 Numerical Results

In this Section we report some numerical results in order to investigate the effectiveness of the proposed Machine Learning algorithm for pricing American options in the multi-dimensional Black-Scholes model.

In particular, we consider the following payoff examples:

- Arithmetic basket Put
$$\Psi(\mathbf{S}_T) = \left(K - \frac{1}{d} \sum_{i=1}^{d} S_T^i \right)_+ ,$$
- Geometric basket Put
$$\Psi(\mathbf{S}_T) = \left(K - \left(\prod_{i=1}^{d} S_T^i \right)^{\frac{1}{d}} \right)_+ ,$$
- Call on the Maximum on d-assets
$$\Psi(\mathbf{S}_T) = \left(\max_{i=1\dots d} S_T^i - K \right)_+ .$$

We consider the following parameters $T = 1$, $S_i = 100$, $K = 100$, $r = 0.05$, constant volatilities $\sigma_i = 0.2$, constant correlations $\rho_{ij} = 0.2$ and $N = 10$ exercise dates. Moreover, we consider $P = 250, 500$ or 1000 points and $M = 10^3, 10^4$ or 10^5 Monte Carlo simulations. As opposed to the other input parameters, we vary the dimension d, considering $d = 2$, 5, 10, 20, 40 and 100. The algorithm has been implemented in MATLAB and computations have been preformed on a server which employs a 2.40 GHz Intel® Xenon® processor (Gold 6148, Skylake) and 20 GB of RAM. Figure 4.1 shows the GPR surface which approximates the price of an Arithmetic basket Put option at different times to maturity.

We present now the numerical results obtained with the GPR-MC for the three payoff examples.

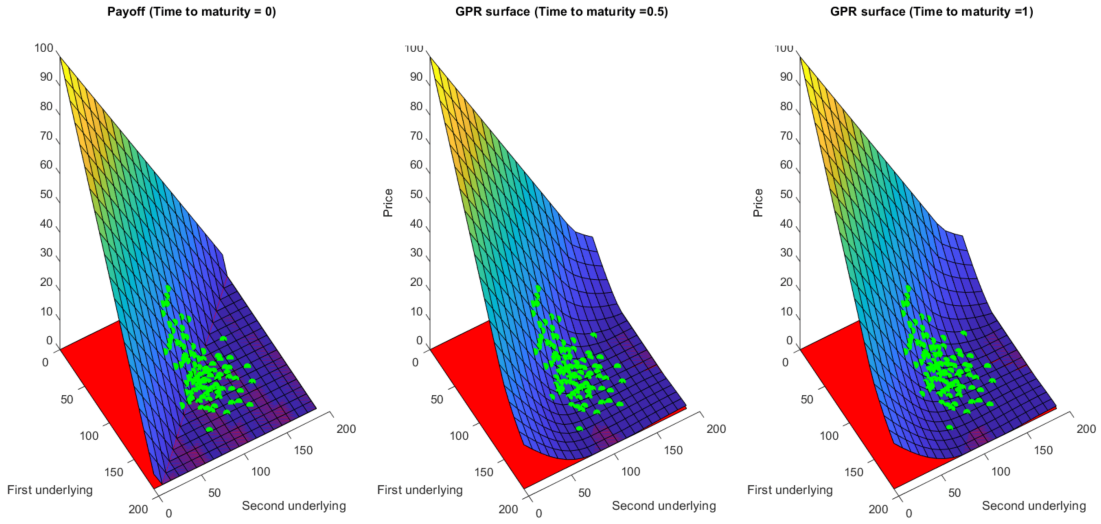

Figure 4.1: The plot of the price surfaces for Arithmetic basket Put with $d = 2$, $P = 100$, $M = 1000$ and $N = 10$. The green points are those employed to build the GPR model, that is $(\mathbf{x}^p, \hat{v}\,(t_n, \mathbf{x}^p))$.

4.1 Geometric basket Put option

As far as the Geometric basket Put is considered, it is possible to reduce the problem of pricing in the d-dimensional model to a one dimensional American Put option in the Black-Scholes model with opportune parameters that we can be computed in a easy way, for example using the CRR algorithm with 1000 steps (see Cox et al. [7]). Therefore in this case we have a reliable benchmark to test the algorithm. Moreover, when d is smaller than 10 we can also compute the price by means of a multi-dimensional binomial tree (see Boyle et al. [5]). In particular, the number of steps employed for the binomial tree is equal to 200 when $d = 2$ and to 50 when $d = 5$. For values of d larger than 5, prices cannot be approximated via such a tree, because the memory required for the calculations would be too large. Results are reported in Table 1. We observe that the proposed method is accurate and stable and the computational time can be very small. We also stress out that the computer processing time increase little with the size of the problem and this makes the method particularly effective when the dimension of the problem is high. This is due to the fact that the dimension affects significantly only the computational time of the Monte Carlo step while the GPR step is only minimally distressed (see Remark 1).

4.2 Arithmetic basket Put option

As opposed to the Geometric basket Put option, in this case we have no method to obtain a fully reliable benchmark. However, for small values of d, a reference price can be obtained by means of a multidimensional tree method (see Boyle et al. [5]), just as shown for the Geometric case. Results are reported in Table 2. As in the case of the Geometric basket Put, the values do not change much with respect to the number P of points and to the number M of Monte Carlo simulations. Moreover, computational times are reasonable and do not depend too much on the dimension d of the problem.

	P	250			500			1000				
d	M	10^3	10^4	10^5	10^3	10^4	10^5	10^3	10^4	10^5	Tree	Bm
2		4.59 (8)	4.58 (27)	4.57 (236)	4.59 (24)	4.57 (115)	4.57 (808)	4.59 (53)	4.57 (515)	4.57 (3199)	4.62	4.62
5		3.46 (8)	3.43 (27)	3.43 (244)	3.45 (21)	3.42 (116)	3.42 (903)	3.44 (61)	3.41 (380)	3.41 (3025)	3.44	3.45
10		2.96 (8)	2.93 (28)	2.93 (256)	2.94 (17)	2.92 (103)	2.92 (1121)	2.93 (65)	2.90 (384)	2.90 (2982)		2.97
20		2.74 (9)	2.69 (52)	2.68 (276)	2.74 (24)	2.69 (104)	2.69 (870)	2.76 (59)	2.70 (388)	2.70 (3577)		2.70
40		2.60 (12)	2.50 (53)	2.50 (336)	2.63 (25)	2.53 (123)	2.53 (873)	2.67 (61)	2.57 (409)	2.57 (4681)		2.56
100		2.49 (14)	2.33 (80)	2.32 (507)	2.52 (25)	2.35 (151)	2.34 (1246)	2.58 (68)	2.40 (432)	2.40 (4288)		2.47

Table 1: American price results for a Geometric basket Put option using the GPR-MC method. In the last two columns the prices obtained by using a multidimensional tree and the exact benchmark. The values in brackets are the computational times (in seconds).

	P	250			500			1000			
d	M	10^3	10^4	10^5	10^3	10^4	10^5	10^3	10^4	10^5	Tree
2		4.39 (8)	4.37 (27)	4.37 (235)	4.39 (16)	4.38 (100)	4.37 (881)	4.39 (46)	4.37 (446)	4.37 (3263)	4.42
5		3.15 (8)	3.13 (27)	3.12 (245)	3.12 (23)	3.10 (101)	3.10 (823)	3.12 (54)	3.09 (377)	3.09 (4491)	3.15
10		2.64 (9)	2.62 (28)	2.62 (260)	2.62 (16)	2.59 (105)	2.59 (921)	2.61 (47)	2.58 (385)	2.58 (3486)	
20		2.34 (9)	2.28 (52)	2.28 (281)	2.37 (23)	2.32 (105)	2.32 (975)	2.44 (67)	2.38 (525)	2.38 (3548)	
40		2.09 (10)	1.99 (54)	1.99 (341)	2.20 (29)	2.09 (126)	2.09 (966)	2.28 (80)	2.18 (410)	2.17 (3774)	
100		1.96 (14)	1.80 (79)	1.80 (504)	2.03 (20)	1.85 (134)	1.84 (1457)	2.12 (59)	1.93 (439)	1.92 (4113)	

Table 2: American price results for an Arithmetic basket Put option using the GPR-MC method. In the last column the prices obtained by using a multidimensional tree. The values in brackets are the computational times (in seconds).

	P	250			500			1000			
d	M	10^3	10^4	10^5	10^3	10^4	10^5	10^3	10^4	10^5	Tree
2		16.82 (8)	16.86 (28)	16.87 (242)	16.81 (23)	16.85 (139)	16.86 (882)	16.81 (47)	16.85 (377)	16.86 (3443)	16.86
5		27.05 (8)	27.12 (28)	27.13 (242)	27.09 (21)	27.16 (141)	27.17 (906)	27.12 (45)	27.19 (385)	27.20 (3426)	27.20
10		35.73 (8)	35.82 (29)	35.82 (260)	35.12 (18)	35.19 (105)	35.19 (917)	35.07 (63)	35.16 (402)	35.17 (3201)	
20		43.13 (9)	43.22 (51)	43.22 (276)	42.83 (18)	42.93 (105)	42.93 (939)	42.65 (64)	42.75 (398)	42.76 (3618)	
40		54.92 (10)	55.00 (54)	55.01 (340)	51.04 (8)	51.13 (126)	51.15 (978)	50.66 (63)	50.68 (419)	50.70 (3691)	
100		57.70 (14)	57.71 (78)	57.71 (503)	54.38 (28)	54.41 (157)	54.41 (1293)	59.60 (82)	59.68 (516)	59.69 (4838)	

Table 3: American price results for a Call on the Maximum option using the GPR-MC method. In the last column the prices obtained by using a multidimensional tree. The values in brackets are the computational times (in seconds).

4.3 Call on the Maximum option

Similarly to the case of the Arithmetic basket Put, in this case we have no method to obtain a fully reliable benchmark. However, for small values of d, we can approximate the price obtained by means of a multidimensional tree method. Results, which are reported in Table 3, are similar to those for the Put on arithmetic mean.

5 Conclusions

In this paper we have proposed a new approach called Gaussian Process Regression - Monte Carlo to price American options on baskets of assets, each of them following a Black-Scholes dynamics. The method exploits both Gaussian Process Regression and Monte Carlo simulations. Numerical results show that the method is accurate and fast for baskets including up to 100 assets. Furthermore, the computation time is shortly growing with respect to the dimension of the basket. Moreover, we stress out that the algorithm is completely parallelizable and therefore the computing time can be significantly reduced. Machine Learning seems to be a very promising tool for American option pricing in high dimension, overcoming the problem of the curse of dimensionality.

References

[1] L. Abbas-Turki and B. Lapeyre. American options by Malliavin calculus and nonparametric variance and bias reduction methods. *SIAM Journal on Financial Mathematics*, 3(1):479–510, 2012.

[2] V. Bally, L. Caramellino, and A. Zanette. Pricing and hedging American options by Monte Carlo methods using a Malliavin calculus approach. *Monte Carlo Methods and Applications mcma*, 11(2):97–133, 2005.

[3] V. Bally, G. Pagès, and J. Printems. First-order schemes in the numerical quantization method. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics*, 13(1):1–16, 2003.

[4] B. Bouchard and N. Touzi. Discrete-time approximation and Monte-Carlo simulation of backward stochastic differential equations. *Stochastic Processes and their applications*, 111(2):175–206, 2004.

[5] P. P. Boyle, J. Evnine, and S. Gibbs. Numerical evaluation of multivariate contingent claims. *The Review of Financial Studies*, 2(2):241–250, 1989.

[6] M. Broadie and P. Glasserman. Pricing American-style securities using simulation. *Journal of economic dynamics and control*, 21(8-9):1323–1352, 1997.

[7] J. C. Cox, S. A. Ross, and M. Rubinstein. Option pricing: A simplified approach. *Journal of financial Economics*, 7(3):229–263, 1979.

[8] J. De Spiegeleer, D. B. Madan, S. Reyners, and W. Schoutens. Machine Learning for quantitative finance: fast derivative pricing, hedging and fitting. *Quantitative Finance*, 18(10):1635–1643, 2018.

[9] L. Goudenège, A. Molent, and A. Zanette. Gaussian process regression for pricing variable annuities with stochastic volatility and interest rate. *preprint arXiv:1903.00369*, 2019.

[10] M. B. Haugh and L. Kogan. Pricing American options: a duality approach. *Operations Research*, 52(2):258–270, 2004.

[11] S. Jain and C. W. Oosterlee. Pricing high-dimensional Bermudan options using the stochastic grid method. *International Journal of Computer Mathematics*, 89(9):1186–1211, 2012.

[12] L. Kocis and W. J. Whiten. Computational investigations of low-discrepancy sequences. *ACM Transactions on Mathematical Software (TOMS)*, 23(2):266–294, 1997.

[13] J. Lelong. Dual pricing of American options by Wiener chaos expansion. *SIAM Journal on Financial Mathematics*, 9(2):493–519, 2018.

[14] P. Lions and H. Regnier. Calcul du prix et des sensibilités d'une option américaine par une méthode de Monte Carlo. *preprint*, 2, 2001.

[15] F. A. Longstaff and E. S. Schwartz. Valuing American options by simulation: a simple least-squares approach. *The review of financial studies*, 14(1):113–147, 2001.

[16] L. C. Rogers. Monte Carlo valuation of American options. *Mathematical Finance*, 12(3):271–286, 2002.

[17] J. N. Tsitsiklis and B. Van Roy. Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives. *IEEE Transactions on Automatic Control*, 44(10):1840–1851, 1999.

[18] C. K. Williams and C. E. Rasmussen. *Gaussian Processes for Machine Learning*, volume 2. MIT Press Cambridge, MA, 2006.