Deep Optimal Stopping

Ludovic Goudenège

March 3, 2020

Premia 22

1 Models of options

We consider the following models for the stock price S_t and the variance V of the volatility:

- Black-Scholes
$$\left\{ \begin{array}{rcl} dS_t & = & (r-\delta)S_t + \sigma S_t dW_t^S, \end{array} \right.$$
- Heston
$$\left\{ \begin{array}{rcl} dS_t & = & (r-\delta)S_t + \sqrt{V_t} S_t dW_t^S, \\ dV_t & = & \alpha_v(\beta_v - V_t)dt + \omega\sqrt{V_t} dW_t^V, \end{array} \right.$$

where r is the risk-free rate, δ is the dividend, σ is the volatility of the asset, ω is the volatility of the volatility, (α_v, β_v) are respectively the reverting speed and the long-run variance. The two Brownian motions W^S and W^V are correlated such that $\langle W^S, W^V \rangle = \gamma$.

We will consider the pricing of an American option (or precisely a Bermudan option) which could be a call or a put (the payoff will be denoted h) with given maturity T and strike K. The initial values of S and V are given by S_0 and V_0. It consists in computing the value of

$$u(t,x) = \sup_{\tau \in \mathcal{T}_{t,T}} \mathbb{E}[e^{-r(\tau - t)} h(S_\tau) | S_t = x]$$

at time $t = 0$, given $u(T, x) = h(x)$, and where $\mathcal{T}_{t,T}$ are stopping times.

Regarding the payoff function h, we will consider multiple types of multi-dimensional options i.e. defined for asset S_i with $i \in \{1, ..., d\}$ with possibly a negative weight/coefficient w (usually $\frac{1}{d}$ for the arithmetical basket option) or negative strike K to take into account put and call options.

- The arithmetical basket option such that $((w \sum_{i=1}^d S_i) - K)^+$.
- The best of basket option such that $((\max_{i=1}^d wS_i) - K)^+$.
- The geometrical basket put option such that $(K - (\prod_{i=1}^d S_i)^{\frac{1}{d}})^+$.
- The geometrical basket call option such that $((\prod_{i=1}^d S_i)^{\frac{1}{d}} - K)^+$.

In all cases, we have to develop a method that can efficiently learn an optimal policy for stopping problems of the form

$$\sup_{\tau \in \mathcal{T}} \mathbb{E}[g(\tau, S_\tau)]$$

2 Deep Optimal Stopping

In the article [4], they have considered a deep learning method to reproduce an optimal policy, using the fact that it could be modeled by a sequence of stopping decision $(f_n)_{1\leq n\leq N} : \mathbb{R}^d \to \{0,1\}$

Consider the auxiliary stopping problems

$$V_n := \sup_{\tau\in\mathcal{T}_n} \mathbb{E}[g(\tau, S_\tau)]$$

for $n = 0, 1, \ldots, N$, where $\mathcal{T}_n$ is the set of all S-stopping times satisfying $n \leq \tau \leq N$ with an obvious time discretization of the pricing problems described in section 1.

Since $\mathcal{T}_N$ consists of the unique element $\tau_N = N$, one can write $\tau_N = N f_N(S_N)$ for the constant function $f_N = 1$. Moreover, for given $n \in \{0, 1, \ldots, N\}$ and a sequence of measurable function $f_n, f_{n+1}, \ldots, f_N : \mathbb{R}^d \to \{0,1\}$ (with $f_N = 1$) we can define

$$\tau_n = \sum_{m=n}^{N} f_m(S_m) \prod_{j=n}^{m-1} (1 - f_j(S_j))$$

which is a stopping time in $\mathcal{T}_n$ (and this form is sufficient to find an approximate solution of the problem, see [4]).

The neural network approximation consists on finding parameters $\theta \in \mathbb{R}^q$ to implement functions $f^\theta : \mathbb{R}^d \to \{0,1\}$. More precisely, let $n \in \{0, 1, \ldots, N-1\}$, and assume parameter values $\theta_{n+1}, \theta_{n+2}, \ldots, \theta_N \in \mathbb{R}^q$ have been found (such that $f^{\theta_N} = 1$) and the stopping time

$$\tau_n = \sum_{m=n}^{N} f_m^{\theta_m}(S_m) \prod_{j=n}^{m-1} (1 - f_j^{\theta_j}(S_j))$$

produces an expected value $\mathbb{E}[g(\tau_{n+1}, S_{\tau_{n+1}})]$ close to the optimum V_{n+1}.

The aim is to determine $\theta_n \in \mathbb{R}^q$ such that

$$\mathbb{E}[g(n, S_n)F^{\theta_n}(S_n) + g(\tau_{n+1}, S_{\tau_{n+1}})(1 - F^{\theta_n}(S_n))]$$

is close to the supremum $\sup_{\theta\in\mathbb{R}^q} \mathbb{E}[g(n, S_n)F^{\theta}(S_n) + g(\tau_{n+1}, S_{\tau_{n+1}})(1 - F^{\theta}(S_n))]$ for a feed forward network

$$F^\theta := \psi \circ a_I^\theta \circ \phi_{I-1} \circ a_{I-1}^\theta \circ \phi_{I-2} \circ \cdots \circ \phi_1 \circ a_1^\theta$$

with ψ is the standard logistic function, $(\phi_i)_{1\leq i\leq I}$ are standard ReLU activation function and $(a_i)_{1\leq i\leq I}$ are affine function. Once this has been done, we set

$$f_n := 1_{[0,\infty[} \circ a_I^{\theta_n} \circ \phi_{I-1} \circ a_{I-1}^\theta \circ \phi_{I-2} \circ \cdots \circ \phi_1 \circ a_1^{\theta_n}.$$

3 Implementation

3.1 How to call the program ?

This program has been implemented in a common framework with other methods using neural network to solve pricing problems described in section 1. These methods are using the Longstaff-Schwartz algorithm with classical polynomial regression, neural network regression, and pre-trained neural network regression (see [1, 2, 3]). These methods can be called by the same commands using specific options.

Precisely, you call the program with the command `python montecarlo.py` followed by

* At least one of the two following options

  1. `-infile` Specify the file describing the payoff, the dynamic of asset and numerical parameters (see 3.2)
  2. `-indir` Run all the problems in the directory.

* Optional arguments could be added

  `-outfile` Path to the output file.

  `-verbose` Be a verbose program.

  `-seed` Specify the seed of the random generator.

  `-loops` Specify the number of launches of the algorithm.

  `-processes` Specify the number of processes to run in parallel when using `-loops`.

  `-euro` Compute the European price.

* At least one of the two following options

  1. `-pol` Use the classical polynomial regressions.
  2. `-dnn` Use the deep neural networks for regression.
  3. `-lsnn_train` Use the deep neural networks for regression and record it on the disk.
  4. `-lsnn_read` Use the deep neural networks written on the disk for regression.
  5. `-dos` Use the deep optimal stopping algorithm.

3.2 How to write a file of parameters ?

On the file of parameters (the name of file should be entered after `-infile` option (see 3.1), you have to write the following values.

For a Black-Scholes model with arithmetical basket of options, with obvious signification of all parameters, here is an example of the file.

```
#This is a comment
model type                          <string> bs
model size                          <int> 5
strike                              <float> -100
spot                                <vector>  100
maturity                            <float> 3
volatility                          <vector> 0.2
interest rate                       <float>  0.05
correlation                         <float> 0.2
dividend rate                       <vector> 0.0

option type                         <string> basket
payoff coefficients                 <vector> -0.2

dates                               <int> 10
#Sub grid for path generation
sub ticks                           <int> 1
MC iterations                       <int> 100000
degree for polynomial regression    <int> 6
neural network file                 <string> ./models/nn_file
number of hidden layers             <int> 0
number of neurons per layer         <int> 128
epochs                              <int> 10
```

For a Heston model with arithmetical basket of options, with obvious signification of all parameters, here is an example of the file.

```
#This is a comment
model type                          <string> heston
model size                          <int> 1
strike                              <float> -100
```

```
spot                                <vector>  100
maturity                            <float> 1
initial volatility                  <vector> 0.01
volatility of volatility            <vector> 0.2
long run variance                   <vector> 0.01
reverting rate                      <vector> 2
interest rate                       <float>  0.0953
correlation                         <float>  0.5
asset vol correlation               <float>  -0.3
dividend rate                       <vector> 0.0

option type                         <string> basket
payoff coefficients                 <vector> -1.

dates                               <int> 10
#Sub grid for path generation
sub ticks                           <int> 3
MC iterations                       <int> 100000
degree for polynomial regression    <int> 6
neural network file                 <string> ./models/nn_file
number of hidden layers             <int> 0
number of neurons per layer         <int> 128
epochs                              <int> 5
```

References

[1] Goudenège, L. and Sainrat, T. (2020). Longstaff-Schwartz algorithm with Neural Network for Bermudan Option Pricing. 2

[2] Lapeyre, B. and Lelong, J. (2019). Neural Network Regression for Bermudan Option Pricing. arXiv preprint arXiv:1907.06474. 2

[3] Kohler, M., Krzyzak, A. and Todorovic, N. (2010). Pricing Of High-Dimensional American Options By Neural Networks. Mathematical Finance, Vol. 20. 2

[4] Becker, S., Cheridito, P. and Jentzen, A. Deep optimal stopping Arxiv:1804.05394 2