FINM 32000: Homework 7

Due Friday March 11, 2022 at 11:59pm

The code in the `ipynb` file should do Problem 1 if you set `hw7MC.algorithm = 'value'`. It should do Problem 2 if you set `hw7MC.algorithm = 'policy'`

Problem 1

Complete the coding of the provided `ipynb` file which prices the Bermudan put option under GBM, with the same parameters as in the Excel worksheet from class (which has been posted on Canvas), using the Longstaff-Schwartz method.

Report an estimated price, based on 10000 paths.

At each exercise date, do the regression using only the paths that are in-the-money (at that specific date – so there may be different subsamples on different dates), not all of the paths.

Problem 2

The Longstaff-Schwartz method can be regarded as an example of a *Reinforcement Learning* (RL) algorithm. It selects actions ("exercise" vs. "continue") to try to maximize an expected reward (option payoff) that depends on the transitions of a state variable (the underlying X).

In particular, Longstaff-Schwartz takes a *Value*-function approach to solving the dynamic programming formulation of the Reinforcement Learning problem. It finds an estimate $\hat{f}_n$ (same notation as L7) of the *value function* for the continuation action, by using OLS regression, of simulated continuation payoffs on the state variable. This estimated continuation value $\hat{f}_n$ is compared against the value function for the exercise action, which is just the payoff function (for example $\text{Payoff}(X) = K - X$ in the case of a put):

$$\text{If } \hat{f}_n(X_{t_n}) > \text{Payoff}(X_{t_n}) \text{ then continue to hold at time } t_n$$
$$\text{If } \hat{f}_n(X_{t_n}) \leq \text{Payoff}(X_{t_n}) \text{ then exercise at time } t_n$$

Here we will consider a different approach to RL.

In contrast to Value-function RL, another approach to Reinforcement Learning is the *Policy*-based approach. Rather than trying to estimate the value function (for the continuation action), it tries to more directly optimize the time-t_n policy function, let's denote it Φ, which maps each X to one of two outputs: $\{0, 1\}$, where 0 denotes continuing to hold, while 1 denotes stopping (exercising).

$$\text{If } \Phi(X_{t_n}) = 0 \text{ then continue to hold at time } t_n$$
$$\text{If } \Phi(X_{t_n}) = 1 \text{ then exercise at time } t_n$$

In the particular one-dimensional example of put pricing that we have been studying, we know what form the stopping policy function should take. In theory it should be an indicator function

$$\Phi_{c_n}(X) = \mathbf{1}_{X \le c_n}$$

with a parameter c_n is a specific "critical" or "threshold" level of the stock price X. Below c_n you should exercise, and above c_n you should continue to hold the put. So, in principle, we could try to estimate the optimal threshold c_n by choosing it to maximize the average, across all simulated paths, of the simulated payout resulting from the policy Φ_{c_n} at time t_n.

However, this optimization has some numerical difficulties, due to the discontinuity of this "hard stopping" decision function Φ which only has two outputs $\{0,1\}$. So suppose that we optimize a smoother function, a "*soft* stopping" decision function φ which produces outputs in the interval *between* 0 and 1. Let φ have two parameters a, b (which may depend on the time slice n) and specifically let φ be[1] a *sigmoid* or *logistic* function of $b(X-a)$:

$$\varphi_{a,b}(X) = \frac{1}{1+\exp(-b(X-a))}. \qquad (*)$$

For large negative b, the $\varphi_{a,b}$ will behave similarly to Φ_a, in that it's near 1 for $X < a$ and near 0 for $X > a$. But unlike the hard stopping decision function, the soft decision function φ is more optimizer-friendly, because it varies continuously between 0 and 1. It can be interpreted as making the exercise decision randomly, with probability $\varphi_{a,b}(X)$ of exercising, and probability $1-\varphi_{a,b}(X)$ of continuing to hold, conditional on X. At time t_n the optimizer should optimize

$$\max_{a,b}\left(\frac{1}{M}\sum_{m=1}^{M}\Big(\varphi_{a,b}(X^m_{t_n})\times(K-X^m_{t_n})+(1-\varphi_{a,b}(X^m_{t_n}))\times(\text{Continuation payout on the } m\text{th path})\Big)\right)$$

where X^m denotes the mth simulated path. Then calculate payouts by converting this optimized soft stopping decision into a hard stopping decision by

$$\Phi(X_{t_n}) = \mathbf{1}_{\varphi_{\hat{a},\hat{b}}(X_{t_n})\ge 0.5} \times \mathbf{1}_{\text{Payoff}(X_{t_n})>0}$$

where $\hat{a}$ and $\hat{b}$ denote the optimized parameter values. Multiplying by $\mathbf{1}_{\text{Payoff}(X_{t_n})>0}$ makes sure that you are not exercising OTM options. It should not be needed if your φ has been trained correctly, but we include it as a precaution.

Implement this policy optimization approach, by completing the code in the `ipynb` file. Most of the coding is already provided.

[1] On this problem, which is simple in the sense that the exercise region in X-space is just a one-dimensional interval, a single sigmoid function (*) is sufficient to approximate the optimal stopping policy.

On harder problems, where the exercise region may be a complicated subset of a multidimensional X-space, the function (*) can be upgraded to a *deep neural network*.

For instance see `http://jmlr.org/papers/volume20/18-232/18-232.pdf`