ISSN: 2641-2039

Global Journal of Engineering Sciences

DOI: 10.33552/GJES.2021.07.000657

Iris Publishers

Mini Review

The Acceleration of Least Squares Monte Carlo in Risk Management

Lu Xiong*

Department of Mathematical Sciences, Middle Tennessee State University, Faculty of Actuarial Science, Faculty of Computational Science, USA

***Corresponding author:** Lu Xiong, Department of Mathematical Sciences, Middle Tennessee State University, Faculty of Actuarial Science, Faculty of Computational Science, USA.

Received Date: January 29, 2021

Published Date: February 09, 2021

Introduction

The Least Squares Monte Carlo (LSMC) method was first proposed by Longstaff and Schwartz [1] to price the American option, since then it has been applied in different industries from banking [2] to energy sector [3]. In the last decade, there is an increasing demand for sophisticated risk modeling [4]. To overcome the computational complexity of those models, the proxy techniques have gain popularity in both risk management practice and research over the last decade [5]. The idea of proxy is to approximate the original model with less features to reduce the computational complexity while keeping sufficient accuracy. Among the various proxy techniques, LSMC is a state-of-the-art approach. However, the polynomial of LSMC is still too complicated in multi-dimensional problems. There are several works that discussed how to further improve the computational speed of LSMC. AS.Chen and PF Shen [6] studied the computational complexity of LSMC. A.R. Choudhury [7] parallelized the LSMC algorithm for American option pricing. Another method to speed up LSMC is focusing on Monte Carlo simulation itself, using techniques such as Quasi-Monte Carlo to make LSMC more efficient [8].

The Application of LSMC in Risk Management

Solvency Capital Requirement (SCR) of Solvency II requires the computation of the economic capital, the minimum capital giving the insurance company a 99.5% survival probability over a one-year horizon via a full probability distribution forecast [9,10].

The SCR at level α=99.5% can be computed as

$$SCR = \arg\min_x \left\{ \mathrm{P}\left[AC_0 - \frac{AC_1}{1+r} > X \right] \le 1-\alpha \right\}$$

AC_t is the available capital at time t:

$$AC_t = ANAV_t + X_t + E_{\mathbb{Q}}\left[\sum_{i=t+1}^{T} \frac{X_i}{(1+r)^i} \middle| Y_s, s \in [0,t] \right]$$

$ANAV_t$ is the adjusted net asset value, X_t is the profit due to in-force business and Y_t is the market conditions at time t.

There is generally no closed-form solution, due to the very complex interactions between X and Y. Nested Monte Carlo (MC) Simulation [11] is usually need for SCR computing. To estimate AC_1, we simulate multiple paths of Y_s (outer scenarios), for each path of Y_s we simulate multiple paths of X_i (inner scenarios). Then the average of present value of X_i is the estimation of E_Q. However, Nested Monte Carlo is extremely time and computer memory consuming. Imagine we have 100,000 policies, for each policy simulate 10,000 outer scenario and 1,000 inner scenarios, each scenario takes 0.001 second, then it would take 32 years to finish the simulation! (Figure 1).

A popular approach in actuarial practice to overcome the time consuming of nested Monte Carlo is using LSMC. Since the $E_{\mathbb{Q}}\left[\sum_{i=t+1}^{T} \frac{X_i}{(1+r)^i} \middle| Y_s, s \in [0,t] \right]$ is a function of Y, it can be approximated by a polynomial:

$$E\left(f(x) \mid Y\right) \approx \sum_{i=0}^{D} c_i g^i$$

The idea is instead of using thousands of inner scenarios to find its expected present value $PV(X_1)$ at time t=1, we train a least square polynomial regressor to approximate it with much fewer inner scenarios and for each outer scenario. The errors tend to offset one another with enough data points. This ends up with an approximator highly accurate and significantly reduced the computing time.

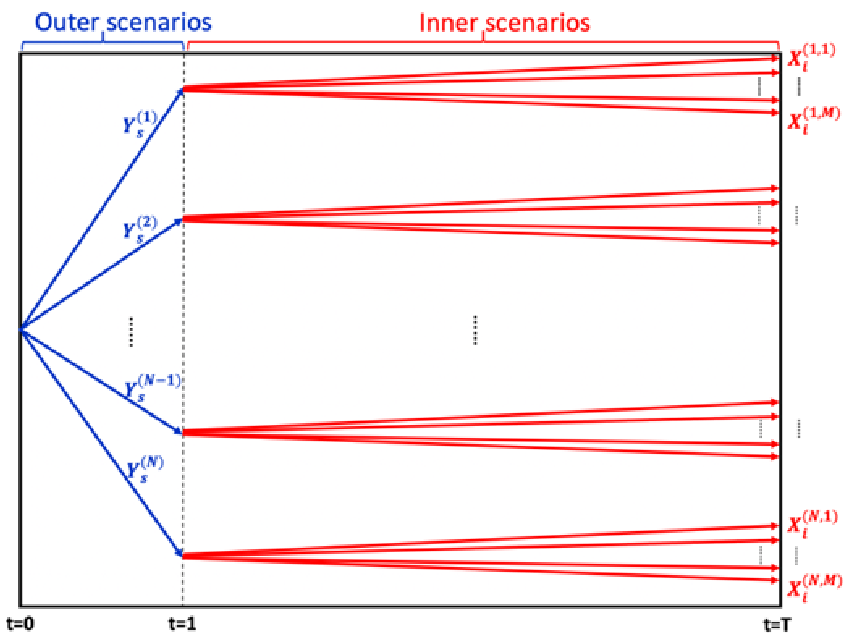

Figure 1: Split-up of scenarios in the nested Monte Carlo simulation.

Distributed Regression for LSMC Speedup

When it comes to multi-factor risks modeling approximation, the multi-dimensional polynomial would be extremely complicated. This would make the regression slow or not possible to finish within reasonable time.

To over the computational complexity of multi-risk factor LSMC, we propose distributed regression for LSMC. The idea of distributed regression is fairly simple: instead of running the regression on one computer, we distribute the regression task to multiple computers (usually using cloud computers), then average the regressed coefficients to get the final regression equation. In this way the computing time can be significantly reduced. We can mathematically prove this simple idea can actually obtain the optimal regression results [12].

There are several advantages of distributed regression: First, the computing time for the traditional least square regression is O(n^3), where n is number of observations in data. While for distributed regression, it's O(n^3/m^2), where m is the number of distributed computers. If we distributed the regression task to 10 computers, we could reduce to computing time to 1% of the original regression, 50 computers to 0.04%. Second, distributed regression can protect the data privacy, because very little or no communication is required when computing from distributed computers. Therefore, almost no data exchanged happened between different data platforms. If we have policy data stored in different platforms and we don't want to share the data across, we can use distributed regression to obtain the regression coefficients from each platform then average the coefficients to get the total regression equation.

We propose the following distribute regression algorithm for LSMC:

- **Step 1:** Suppose the conventional LSMC requires n·K outer scenarios. We have K worker computers and 1 master computer in our distributed system. Each worker computer generates *n* outer scenarios $Y_s^{(i)}$ and for each outer scenario simulate 1 inner scenario X_i. In this way, we obtain n pairs of local training data ($Y_s^{(i)}$, X_i).
- **Step 2:** Each worker computer run the least squares algorithm on the local data ($Y_s^{(i)}$, X_i) to get the coefficients c_k of the polynomial to fit the function $E_{\mathbb{Q}}\left[\sum_{i=t+1}^{T}\frac{X_i}{(1+r)^i}\middle|Y_s, s\in[0,t]\right]$.
- **Step 3:** Each worker computer sends its fitted coefficients c_k to the master computer. The master computer averages the coefficients $C=\frac{\sum_{k=1}^{K}C_k}{K}$ and output this as the final coefficients of the fitting polynomials. There is no data communication between the computers, only the trained coefficients are sent. This protects the data privacy.
- **Step 4:** Scale this algorithm with more worker computers to find the optimal number of computers in terms of computing speed and cost.

There are several advantages using distributed regression to accelerate the LSMC. 1) The current parallel algorithms for LSMC require the parallel computing of the big matrix inverse, while using distributed regression we only need compute the small matrix inversion for each chunk of data. 2) When comes to multi-risk modeling, the amount of the outer scenarios would be huge that no single computer can handle it. For a N risk-factor problem, it will require 10000^N outer scenarios if we simulate 10,000 outer scenarios for each risk-factor. If we use distributed regression, each computer only needs processes a smaller chunk of data assigned. 3) This divide-and-conquer type distributed learning method can also be applied to speed up other algorithms like clustering, tree-based method, deep learning etc. 4) Easy to be scaled on distributed framework like Map-reduce, or Spark [13].

Acknowledgement

None.

Conflict of Interest

No conflict of interest.

References

1. F Longstaff, E Schwartz (2001) Valuing American options by simulation: a simple least-squares approach. The review of financial studies 14(1): 113-147.
2. D Bauer, H Ha (2015) A least-squares Monte Carlo approach to the calculation of capital requirements. World Risk and Insurance Economics Congress, Munich, Germany.
3. S Nadarajah, F Margot, N Secomandi (2017) Comparison of least squares Monte Carlo methods with applications to energy real options. European Journal of Operational Research 256(1): 196-204.
4. AO Donnell (2010) Demand for Sophisticated Risk Management Capabilities Increasing. Insurance & Technology.
5. D Kopczyk (2018) Proxy modeling in life insurance companies with the use of machine learning algorithms. Available at SSRN 3396481.
6. AS Chen, PF Shen (2003) Computational Complexity Analysis of Least-squares Monte Carlo (LSM) for Pricing US Derivatives. Applied Ecomomics Letters 10: 223-229.
7. AR Choudhury, A King, S Kumar, Y Sabharwal (2008) Optimizations in Financial Engineering The Least Squares Monte Method of Longstaff and Schwartz. Parallel and Distributed Processing, IPDPS 2008. IEEE International.
8. L Yun (2010) Monte Carlo Simulation in Option Pricing. National University of Singapore. CORE.
9. D Bauer, A Reuss, D Singer (2012) On the calculation of the solvency capital requirement based on nested simulations." ASTIN Bulletin-Actuarial Studies in Non-Life Insurance 42(2): 453.
10. P Cadoni (2014) Internal models and Solvency II. Risk Books, London.
11. D Bauer, D Bergmann, A Reuss (2010) Solvency II and nested simulations–a least-squares Monte Carlo approach. Proceedings of the 2010 ICA congress.
12. Z Guo, L Shi, Q Wu (2017) Learning theory of distributed regression with bias corrected regularization kernel network. The Journal of Machine Learning Research 18(1): 4237-4261.
13. X Zhu, F Li, H Wang (2019) Least Squares Approximation for a Distributed System. arXiv preprint arXiv:1908.04904.