Solving Nonlinear Partial Differential Equations via Deep Galerkin Method

Solving Nonlinear Partial Differential Equations via Deep Galerkin Method

Xinjun Ji[1]

[1]School of the Hong Kong Polytechnic University, Hong Kong, China

*Corresponding author e-mail: xinjun_ji@163.com

Abstract. This paper introduces a one-dimensional nonlinear Black-Scholes equation. Because nonlinear models are difficult to obtain analytical solutions, finite difference schemes are usually used to solve them. Numerical solutions of partial differential equations have always been a hot research topic in the frontier; the finite element method, finite body method, and other methods are commonly used in research. In general, these methods require the use of grids to approximate the definition space of PDEs. When we have a finer grid, the more accurate solution will be. However, the smaller the grid, the higher the computing cost and the larger the storage space. Therefore, the meshless method is adopted in this paper.

1. Introduction

This paper introduces a one-dimensional nonlinear Black-Scholes equation. The Black-Scholes option pricing model, a model established by Robert and Myron, is designed to accurately price financial derivatives that fluctuate due to price fluctuations in the market. Because nonlinear models are difficult to obtain analytical solutions, finite difference schemes are usually used to solve them. Numerical solutions of partial differential equations have always been a hot research topic in the frontier. In general, traditional methods require the use of grids to approximate the definition space of PDEs. When we have a finer grid, the solution will be more accurate. However, the smaller the grid, the higher the computing cost and the larger the storage space. Therefore, the meshless method is adopted in this paper. This paper briefly introduces the deep learning method in partial differential equations and then introduces the Deep Galerkin Method proposed by Sirignano and Spiliopoulos[1]. The Deep Galerkin Method is used to solve nonlinear partial differential Black-Scholes equations. Finally, a numerical example of the Black-Scholes equation is given. The Deep Galerkin Method (DGM) establishes the loss function by parameterization and penalizes the deviation of the fitting function from the expected differential operator and boundary conditions. This method uses a computational graph and backpropagation algorithm to compute differential operators efficiently and directly compute boundary conditions. For the training data, the sample points used by the network are obtained from a random sampling of the function-defined region and are optimized by random gradient descent.

2. The Black-Scholes Partial Differential Equation

While solving the Black-Scholes equation, we use the Deep Galerkin Method. Some basic assumptions of the Black-Scholes model are as follows:

（1）There are no short-selling restrictions, which do not allow traders to use all of their profits to short the market. The frictionless nature of the market is that all securities and assets are highly divisible, and, at the same time, the efficiency of the market will be free of taxes and exchange fees.

（2）The risk-free rate is constant, at t= (0, T), the borrowing rate is the same, and within this interval, there is no arbitrage opportunity, and the interest rate is calculated as the risk-free continuous compound r.

（3）There is no dividend on the stock at t=(0, T).

（4）Stock trading is continuous, and stock price is a random process that follows a lognormal distribution. This random process includes the following conditions: continuous change of stock price; The return expectation and price variance of stock are stable in the validity period. Stocks have independent returns.

Symbol Description:

S: Stock price

K: Strike price

V: Post-expiration value of the option

σ:Volatility

r: Risk-free rate

T: Maturity time

t: Time variable

$$\frac{\partial V}{\partial t}+\frac{1}{2}\sigma^2(t,S,\frac{\partial V}{\partial S},\frac{\partial^2 V}{\partial S^2})S^2\frac{\partial^2 V}{\partial S^2}+rS\frac{\partial V}{\partial S}-rV=0$$

The boundary conditions for solving this equation are 1, 2, 3

(1) The value after expiration of the option can be expressed as $V(S,T)=(S-K)^+$

(2) $\lim_{S\to\infty}\frac{V(S,t)}{S}=1$; as the stock price increases, the option price gets closer to the stock price.

(3) $\lim_{S\to 0}V(S,t)=0$; once a stock price reaches zero, it generally does not return to its original state.

3. An introduction to deep learning

3.1. Artificial neural network

Deep learning is a technique that uses an artificial neural network (ANN). In the process of using it, sample data needs to be trained first. The neural network can then be trained to perform relevant tasks. The process of using a trained artificial neural network is called "reasoning." During reasoning, the neural network evaluates the supplied data according to the learned rules.

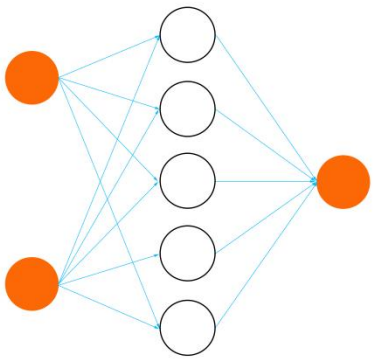

Figure 1. Artificial neural network.

3.2. Neurons, layers, and connections

Artificial neural networks have many interconnected "neurons." In the simplest case, these layers refer specifically to the input and output layers. The value of the connection input layer contains the weight of the corresponding connection, through which the corresponding result can be calculated at the output layer.

3.3. Deep artificial neural network

Compared with ordinary ANN, deep learning also has a "hidden layer." After data input from the input layer is entered, the "hidden layer" is used for calculation. The resulting matrix of each hidden layer is the input matrix of the next layer. Therefore, only the output matrix of the last layer provides the results.

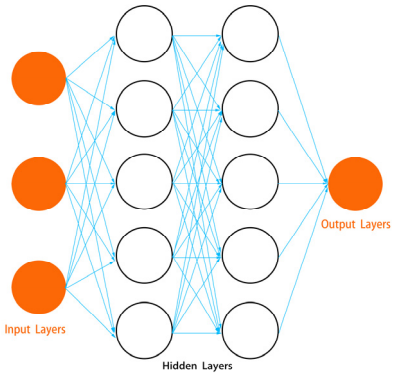

Figure 2. Deep artificial neural network.

3.4. Training

When training the neural network, the initial focus is set randomly, and then sample data is gradually added. Use training rules to adjust relationship weights based on input data and expected results. Therefore, in the process of neural network training, we need to use non-repetitive, variable sample data. If you train with a large number of very similar or duplicate data, ANN will not be able to estimate the fields when it encounters data that is different from the sample data. This is the overfitting of artificial neural networks.

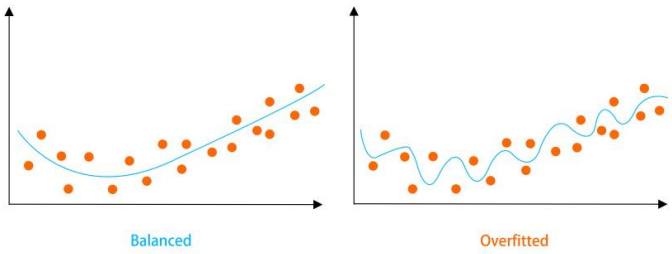

Figure 3. Estimation of different sample data.

4. The Deep Galerkin Method

4.1. Introduction

Partial differential equations are widely used to simulate phenomena in nature. Firstly, they are used in physical science. With the development of society, they are increasingly involved in chemistry, life science, and even humanities and social sciences. Therefore, partial differential equations are actually the most important bridge between the whole analytical mathematics and applied science. Black-Scholes equation models option prices. It can also be used to perfect the model when dividends are paid. From this model, we can derive the Black-Schels formula and estimate the theoretical price of the European option. The formula brought about a boom in the options market. For the traditional calculation method of partial differential equations, the more grids, the higher the accuracy, which leads to an increase in the calculation amount, so this paper adopts the meshless calculation method.

4.2. Method introduction

Sirignano and Spiliopoulos(2018)[1] proposed the Deep Galerkin Method, which is a grid-free Method for solving partial differential equations and also uses neural networks for solving. This paper adopts the meshless method of the paper 'DGM: A Deep Learning Algorithm for Solving partial Differential equations' [1] published in 2018.

Essentially, neural networks can be a type of function, u_θ. The input layer and output layers are the income layer and output layer of u_θ, respectively. So, we use this function u_θ as the approximation function in the partial differential equation, and we solve for this function, and we solve for the partial differential equation. Therefore, we solve the following partial differential equation

$$\begin{cases} (\partial_t + \mathcal{L})u(x,t) = 0, (x,t) \in \Omega \times [0,T] \\ u(x,0) = u_0(x), x \in \Omega \\ u(x,t) = g(x,t), x \in \partial\Omega \times [0,T] \end{cases}$$

Because u depends on variable x and variable t, so u is an unknown function, and $\partial\Omega$ is the boundary of space Ω. So, we need u_θ to satisfy the system as far as possible. So the cost function is

$$J(u_\theta) = \left|\left|\partial_t u_\theta(x,t;\theta) + \mathcal{L}u_\theta(x,t;\theta)\right|\right|^2_{\Omega,\nu_1\times[0,T]} + \left|\left|u_\theta(x,t;\theta) - g(x,t)\right|\right|^2_{\partial\Omega,\nu_2\times[0,T]} + \left|\left|u_\theta(x,0;\theta) - u_0(x)\right|\right|^2_{\Omega,\nu_3}$$

In this function, $\left|\left|f\right|\right|^2_{\Omega,\nu} = \int_\Omega f^2 \nu dx$, where v is the probability density function of Ω. $J(u_\theta)$ is the approximate error of the function u_θ. If the value of $J(u_\theta)$ is very small, which means that u_θ is roughly satisfied partial differential equations, initial conditions, and boundary conditions in the above equations. Therefore, we need to minimize the value of $J(u_\theta)$

$$\min_\theta J(u_\theta)$$

4.3. Algorithm

1. According to probability densities ν_1 and ν_2, we generate random points (x_n, t_n) and (z_n, τ_n) from $\Omega \times [0,T]$ and $\partial\Omega \times [0,T]$ respectively. And according to the probability density ν_3, we generate a random point ω_n from Ω.

2. Calculate the squared error $G(\theta_n, s_n)$, where $s_n = \{(t_n, x_n), (\tau_n, z_n), \omega_n\}$:

$$G(\theta_n, s_n) = (\partial_t f(x_n, t_n; \theta_n) + \mathcal{L}f(x_n, t_n; \theta_n))^2 + (f(z_n, \tau_n; \theta_n) - g(z_n, \tau_n))^2 + (f(\omega_n, 0; \theta_n) - u_0(\omega_n))^2$$

3. The implementation of random gradient descent:

$$\theta_{n+1} = \theta_n - \alpha_n \nabla_\theta G(\theta_n, s_n)$$

4. Repeat the above steps until convergence is achieved.

5. Numerical Examples

Reference [6] introduced a numerical example. Compared to that paper, the re-sample set is expanded, and the number of times to re-sample new time-space domain points =100 is expanded to the number of times to re-sample new time-space domain points =500.

In this numerical example of the DGM approach, the following parameter values for neural network training can be used：

Interest rate r = 5%
Volatility σ = 25%
Initial stock price S_0 = 50
Maturity time T = 1
Strike price K = 50

Layer number=3
Nodes per layer=50
Learning rate=0.001
Sample stage=50, 100, 500 (Number of times to re-sample new time-space domain points)
Steps per sample=10 (Number of SGD steps to take before re-sampling)

In the figure below, we compare the real option value and estimated option value of different maturities based on the DGM method at the sample stage=50, 100, and 500, respectively. In the same sample stage, the fitting effect of the model was better with the growth of maturity time, and the model was better with the growth of the sample stage.

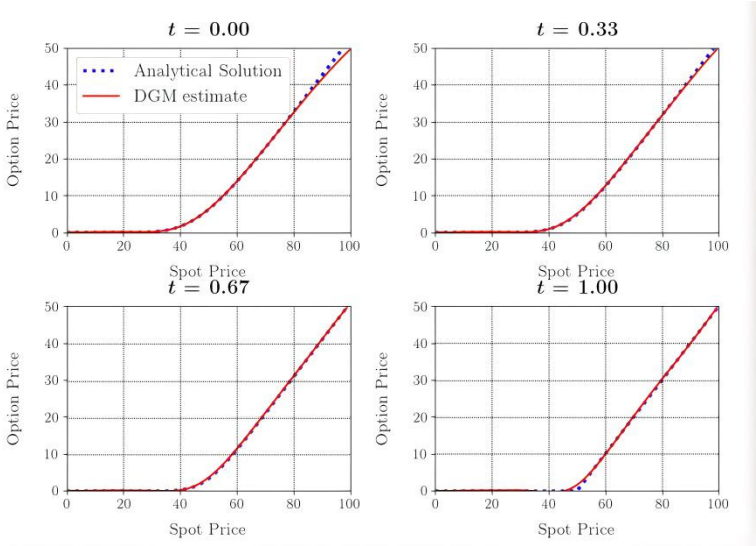

Figure 4. Sample stage=50.

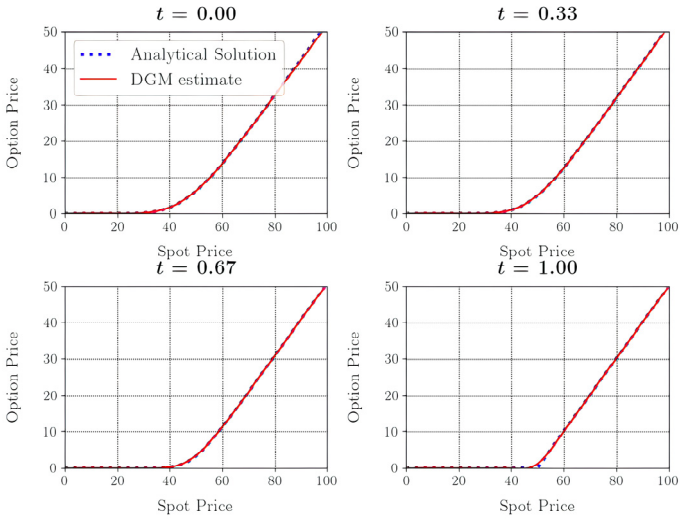

Figure 5. Sample stage=100.

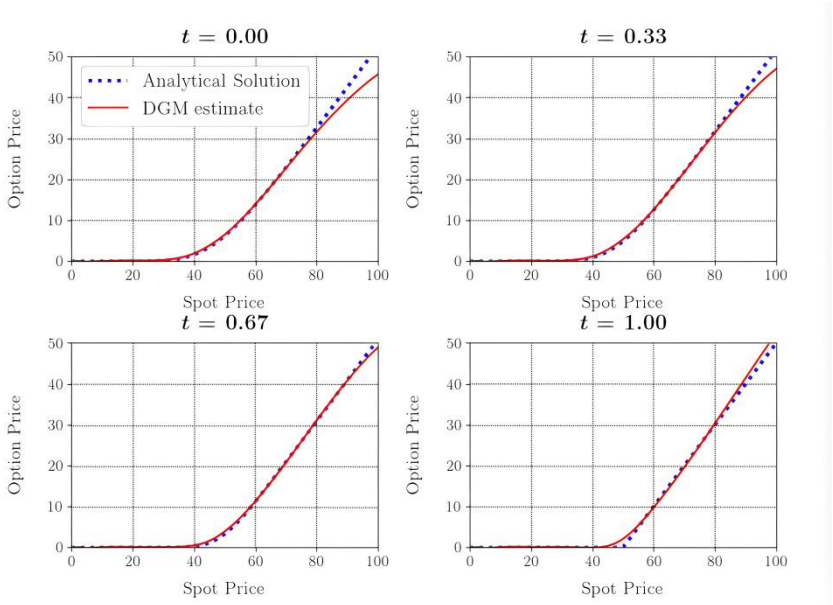

Figure 6. Sample stage=500.

6. Conclusion

In this paper, the Deep Galerkin Method is used to solve the one-dimensional nonlinear Black-Scholes equation. Firstly, the form, properties, and difference scheme of the equation are introduced. Then the calculation process of the Deep Galerkin Method is introduced. Finally, the Deep Galerkin Method is applied to the numerical solution of the Black-Scholes equation. The Deep Galerkin Method is used to solve the nonlinear Black-Scholes equation and linear equation.

According to the analysis, the one-dimensional nonlinear Black-Scholes equation is calculated by the Deep Galerkin Method. When the sample set is large enough, the model fitting effect is good, which indicates that the Deep Galerkin Method is feasible to solve nonlinear partial differential equations. Different sample stages were also compared. In the same sample stage, the fitting effect of the model was better with the growth of maturity time, and the model was better with the growth of the sample stage. In this paper, the combination of a one-dimensional nonlinear problem and the Deep Galerkin Method is discussed.

References

[1] Sirignano J, Spiliopoulos K . DGM: A deep learning algorithm for solving partial differential equations[J]. Journal of Computational Physics, 2017, 375.
[2] Al-Aradi A, Correia A, Naiff D, et al. Applications of the Deep Galerkin Method to Solving Partial Integro-Differential and Hamilton-Jacobi-Bellman Equations[J]. 2019.
[3] Al-Aradi A, Correia A, Naiff D, et al. Extensions of the Deep Galerkin Method[J]. 2019.
[4] Zang Y, Bao G, Ye X, et al. Weak Adversarial Networks for High-dimensional Partial Differential Equations[J]. 2019.
[5] Hasan A, Pereira J M, Ravier R, et al. Learning Partial Differential Equations from Data Using Neural Networks[J]. 2019.
[6] Al-Aradi A, Correia A, Naiff D, et al. Solving Nonlinear and High-Dimensional Partial Differential Equations via Deep Learning. arXiv.org, 2018.