IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, January 2024

Solving High-Dimensional Partial Differential Equations using Deep Learning

N. Srinivas[1] and Dr. Manjeet Jakhar[2]
Research Scholar, Department of Mathematics[1]
Research Guide, Department of Mathematics[2]
NIILM University, Kaithal, India

Abstract: *Solving high-dimensional partial differential equations (PDEs) poses a significant challenge due to the computational complexity and memory requirements involved. Traditional numerical methods encounter limitations when dealing with large-scale problems, motivating the exploration of alternative techniques. Deep learning has emerged as a promising approach to address these challenges by leveraging the representational power of neural networks. In the context of solving high-dimensional PDEs, deep learning techniques offer several advantages, including scalability, flexibility, and the ability to learn complex mappings between input and output spaces. By utilizing architectures such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), researchers have developed innovative methods to approximate solutions to high-dimensional PDEs. These approaches often involve training neural networks on simulated or experimental data to learn the underlying dynamics of the system, enabling efficient and accurate predictions. Additionally, techniques such as physics-informed neural networks (PINNs) integrate domain knowledge into the learning process, enhancing the robustness and interpretability of the models. Despite the progress achieved, challenges remain in optimizing network architectures, handling large datasets, and ensuring generalization to diverse problem domains. Nevertheless, the intersection of deep learning and high-dimensional PDEs holds great promise for advancing computational science and engineering applications, paving the way for more efficient and scalable solutions to complex physical phenomena*

Keywords: Deep Learning, Solving, Single-line Approach, Efficiency

I. INTRODUCTION

Solving high-dimensional partial differential equations (PDEs) has been a longstanding challenge in various scientific and engineering fields due to the computational complexity associated with traditional numerical methods. However, recent advancements in deep learning techniques have shown promise in tackling this issue effectively. By leveraging deep learning models such as neural networks, researchers have been able to develop innovative approaches for solving PDEs in high-dimensional spaces efficiently. These methods typically involve training neural networks to approximate the solution of the PDE by learning from a set of training data or by directly solving the PDE using techniques like deep Galerkin methods or physics-informed neural networks. The advantage of using deep learning in this context lies in its ability to handle complex, nonlinear relationships within the PDE while effectively capturing high-dimensional features of the problem domain. Moreover, deep learning-based approaches offer scalability and adaptability, allowing for the solution of PDEs in diverse high-dimensional scenarios encountered in fields such as computational physics, finance, materials science, and beyond. As researchers continue to explore and refine these techniques, the integration of deep learning into the realm of high-dimensional PDE solving holds great promise for advancing our understanding and application of complex physical phenomena across various disciplines.

II. METHODOLOGY

To tackle high-dimensional partial differential equations (PDEs) using deep learning, we employ neural networks as function approximators to learn the complex mappings between input variables and the PDE solution. The methodology involves several key steps:

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, January 2024

1. Problem Formulation: Define the high-dimensional PDE problem, including the governing equations, boundary conditions, and initial conditions.

2. Data Generation: Generate a dataset of input-output pairs by sampling the high-dimensional input space and solving the PDE numerically using traditional methods (e.g., finite element methods or finite difference methods).

3. Network Architecture Design: Design deep neural network architecture suitable for approximating the PDE solution. This architecture typically includes multiple layers of neurons, possibly incorporating convolutional layers for spatial dimensions and recurrent layers for temporal dimensions in time-dependent PDEs.

4. Loss Function Definition: Define a loss function that quantifies the discrepancy between the neural network predictions and the true PDE solution. Common choices include mean squared error or other physically motivated loss functions.

5. Training: Train the neural network by minimizing the defined loss function using gradient-based optimization techniques such as stochastic gradient descent (SGD) or Adam.

6. Validation and Testing: Validate the trained model on a separate validation dataset and assess its performance. Additionally, test the model on unseen data to evaluate its generalization capability.

The methodology can be summarized mathematically as:

$$\min_{\theta} \sum_{i=1}^{N} L\left(y_i, f_\theta(x_i)\right)$$

where θ represents the parameters of the neural network, L denotes the chosen loss function, f_θ is the neural network model, and (x_i, y_i) are the input-output pairs from the dataset.

Nonlinear Black–Scholes Equation with Default Risk

The Black-Scholes equation is a cornerstone in financial mathematics, used to model the price dynamics of financial derivatives. However, when considering default risk, the equation becomes nonlinear due to the presence of a default boundary. This nonlinear Black-Scholes equation with default risk is challenging to solve, particularly in high-dimensional settings, where traditional numerical methods struggle due to the curse of dimensionality.

To address this challenge, deep learning techniques have emerged as a promising approach for solving high-dimensional partial differential equations (PDEs) efficiently. Deep neural networks (DNNs) have shown remarkable capabilities in learning complex mappings from input to output spaces, making them suitable candidates for approximating the solutions of PDEs.

One approach involves formulating the nonlinear Black-Scholes equation with default risk as an initial or boundary value problem and training a deep neural network to approximate its solution. The neural network takes the relevant inputs such as asset price, time, volatility, and default boundary information and outputs an approximation of the option price.

Mathematically, the solution $V(S,t)$ of the nonlinear Black-Scholes equation with default risk can be represented as:

$$\frac{\partial V}{\partial t} + \mathcal{L}_{BS}(V) = 0$$

where $\mathcal{L}_{BS}(V)$ represents the Black-Scholes operator with default risk. By training the neural network on a dataset of known solutions or using reinforcement learning techniques, the network learns to approximate the solution of the nonlinear Black-Scholes equation accurately, even in high-dimensional spaces, offering a promising avenue for efficient and accurate pricing of financial derivatives under default risk.

Hamilton–Jacobi–Bellman Equation

The Hamilton-Jacobi-Bellman (HJB) equation is a fundamental concept in optimal control theory, often used to solve high-dimensional partial differential equations (PDEs) arising in various fields such as finance, robotics, and aerospace engineering. It represents a dynamic programming equation that characterizes the optimal value function of a stochastic control problem. In many practical scenarios, solving the HJB equation analytically is intractable due to the curse of dimensionality, especially in high-dimensional state spaces.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, January 2024

To address this challenge, deep learning techniques have been increasingly employed to approximate solutions to the HJB equation in high-dimensional settings. Deep neural networks (DNNs) are particularly well-suited for this task due to their ability to learn complex mappings from input to output spaces. By parameter zing the value function in the HJB equation with a deep neural network, one can leverage techniques such as stochastic gradient descent to iteratively update the network parameters until convergence to an approximate solution is chieved.

The HJB equation in its general form is given by:

$$\frac{\partial V}{\partial t} + \min_{u \in U}\{-H(x, u, \nabla V)\} = 0$$

where V is the value function, t is time, x is the state, u is the control input, and H is the Hamiltonian of the system. Deep learning methods seek to approximate the value function V directly from data, bypassing the need for explicit analytical solutions in high-dimensional spaces. By training the neural network on historical data or through reinforcement learning techniques, the approximate solution to the HJB equation can be obtained, enabling efficient and scalable solutions to complex optimal control problems.

Allen–Cahn Equation

The Allen-Cahn equation, a classical partial differential equation (PDE) in the field of materials science and phase transitions, describes the evolution of a scalar order parameter in time. It is given by:

$$\frac{\partial u}{\partial t} = \epsilon \nabla^2 u - f(u) + \eta(\mathbf{x}, t)$$

where $u(\mathbf{x}, t)$ represents the order parameter, ∇^2 is a small positive constant representing the interface thickness, \(\nabla^2\) is the Laplacian operator, $f(u)$ is a double-well potential, and $\eta(\mathbf{x}, t)$is a noise term.

Solving high-dimensional versions of the Allen-Cahn equation poses significant challenges due to the curse of dimensionality and the computational complexity involved. Recently, deep learning techniques have emerged as powerful tools for approximating solutions to high-dimensional PDEs. Specifically, neural networks can learn complex mappings from input space to solution space without explicitly solving the PDE.

In the context of the Allen-Cahn equation, deep learning approaches involve training neural networks to approximate the solution manifold. By feeding input data consisting of spatial and temporal coordinates into the neural network, it learns to predict the evolution of the order parameter over time. This approach bypasses the need for traditional grid-based numerical methods, offering advantages in terms of computational efficiency and scalability to high-dimensional systems. Moreover, deep learning frameworks can handle noisy data and capture intricate spatiotemporal patterns, making them particularly well-suited for modeling complex phase transitions described by the Allen-Cahn equation in high-dimensional systems.

III. MATERIALS AND METHODS

To solve high-dimensional partial differential equations (PDEs) using deep learning, we adopt a neural network-based approach that leverages the expressive power of deep learning architectures to approximate the solutions. The key idea is to train a deep neural network to directly learn the mapping from the input space, typically the domain of the PDE, to the solution space, where the solution of the PDE resides. Let's denote the input space as Ω and the solution space as U. The PDE we aim to solve is of the form:

$$F(u(x), \nabla u(x), \nabla^2 u(x), \ldots, \nabla^n u(x), x) = 0$$

where $u(x)$ is the solution function defined on Ω, and ∇ represents the gradient operator. Our goal is to find an approximation $u_\theta(x)$ such that $F(u_\theta(x), \nabla u_\theta(x), \ldots, \nabla^n u_\theta(x), x) \approx 0$, where θ denotes the parameters of the neural network.

We employ a deep neural network architecture, such as a convolution neural network (CNN) or a recurrent neural network (RNN), to learn this mapping. The network takes x as input and outputs $u_\theta(x)$. To train the network, we minimize a suitable loss function that quantifies the error between the predicted solution $u_\theta(x)$ and the true solution of the PDE. This is achieved through techniques like stochastic gradient descent (SGD) or more advanced optimization algorithms. Additionally, we may augment the training data using techniques like data augmentation or domain decomposition to improve generalization and handle high-dimensional input spaces efficiently.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, January 2024

BSDE Reformulation

In recent years, the quest to tackle high-dimensional partial differential equations (PDEs) has led to a surge of interest in leveraging deep learning techniques for their solution. One promising avenue in this pursuit is the reformulation of these PDEs as backward stochastic differential equations (BSDEs), a framework that offers a unique perspective on the underlying dynamics. By expressing the PDEs in terms of BSDEs, we can tap into the rich reservoir of methodologies developed for solving stochastic processes, harnessing their flexibility and efficiency.

At the heart of this reformulation lies the fundamental relationship between PDEs and BSDEs, encapsulated by the Feynman-Kac formula:

$$u(t,x) = \mathbb{E}\left[\int_t^T f(s,X_s)ds + g(X_T) \mid X_t = x\right]$$

Here, $u(t,x)$ represents the solution of the PDE at time t and position x, while $f(s,X_s)$ and $g(X_T)$ encode the drift and terminal conditions, respectively, of the corresponding BSDE. By manipulating this relationship and exploiting the expressive power of deep learning architectures such as neural networks, we can effectively approximate the solution of the original PDE. This approach not only circumvents the curse of dimensionality that plagues traditional numerical methods but also opens up avenues for capturing complex nonlinearities and high-dimensional dependencies with remarkable accuracy.

In essence, the reformulation of high-dimensional PDEs as BSDEs, coupled with the prowess of deep learning, represents a paradigm shift in the realm of computational mathematics, offering a potent tool for tackling some of the most challenging problems in science and engineering.

IV. CONCLUSION

In conclusion, employing deep learning techniques for solving high-dimensional partial differential equations (PDEs) presents a promising avenue with both challenges and opportunities. By leveraging neural networks, particularly deep architectures like convolutional neural networks (CNNs) or recurrent neural networks (RNNs), researchers have demonstrated remarkable success in approximating solutions to complex PDEs in high-dimensional spaces. These methods offer advantages such as scalability to large datasets, flexibility in handling non-linearities, and potential for parallelization across computing resources.

Mathematically, the deep learning framework for solving PDEs involves parameterizing the solution space using neural networks. For instance, in the context of solving the heat equation in a high-dimensional domain Ω, the temperature distribution $u(\mathbf{x},t)$ can be approximated by a neural network $u_\theta(\mathbf{x},t)$, where θ represents the network parameters. This parameterization transforms the PDE into a problem of learning the parameters θ that minimize the discrepancy between the predicted and actual solutions, typically through techniques like stochastic gradient descent.

However, challenges persist, including the need for large amounts of labeled data, issues with generalization to unseen regions of the domain, and computational inefficiency for extremely high-dimensional systems. Additionally, the interpretability of deep learning models in the context of PDE solutions remains an ongoing research area.

Despite these challenges, the marriage of deep learning and high-dimensional PDEs holds great promise for advancing scientific computing, enabling more efficient and accurate solutions to a wide range of physical phenomena across various disciplines. Continued research efforts aimed at addressing the aforementioned challenges will likely lead to further advancements in this field.

REFERENCES

[1]. Bellman RE (1957) Dynamic Programming (Princeton Univ Press, Princeton).
[2]. Goodfellow I, Bengio Y, Courville A (2016) Deep Learning (MIT Press, Cambridge, MA).
[3]. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521:436–444.
[4]. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, eds Bartlett P, Pereira F, Burges CJC, Bottou L, Weinberger KQ (Curran Associates, Inc., Red Hook, NY), Vol 25, pp 1097–1105.
[5]. Hinton G, et al. (2012) Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Process Mag 29:82–97.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, January 2024

[6]. Silver D, Huang A, Maddison CJ, Guez A, et al. (2016) Mastering the game of Go with deep neural networks and tree search. Nature 529:484–489.
[7]. Pinkus A (1999) Approximation theory of the MLP model in neural networks. Acta Numerica 8:143–195.
[8]. Pardoux ´ E, Peng S (1992) Backward stochastic differential equations and quasilinear parabolic partial differential equations. Stochastic Partial Differential Equations and Their Applications (Charlotte, NC, 1991), Lecture Notes in Control and Information Sciences, eds Rozovskii BL, Sowers RB (Springer, Berlin), Vol 176, pp 200–217.
[9]. Pardoux E´ , Tang S (1999) Forward-backward stochastic differential equations and quasilinear parabolic PDEs. Probab Theor Relat Fields 114:123–150.
[10]. Darbon J, Osher S (2016) Algorithms for overcoming the curse of dimensionality for certain Hamilton–Jacobi equations arising in control theory and elsewhere. Res Math Sci 3:19.
[11]. E W, Hutzenthaler M, Jentzen A, Kruse T (2016) On multilevel Picard numerical approximations for high-dimensional nonlinear parabolic partial differential equations and high-dimensional nonlinear backward stochastic differential equations.arXiv:1607.03295.
[12]. Henry-Laborde` re P (2012) Counterparty risk valuation: A marked branching diffusion approach. arXiv:1203.2369.
[13]. Henry-Laborde` re P, Tan X, Touzi N (2014) A numerical algorithm for a class of BSDEs via the branching process. Stoch Proc Appl 124:1112–1140.
[14]. Kingma D, Ba J (2015) Adam: A method for stochastic optimization. arXiv:1412.6980. Preprint, posted December 22, 2014.
[15]. Black F, Scholes M (1973) The pricing of options and corporate liabilities. J Polit Econ 81:637–654; reprinted in Black F, Scholes M (2012) Financial Risk Measurement and Management, International Library of Critical Writings in Economics (Edward Elgar, Cheltenham, UK), Vol 267, pp 100–117.
[16]. Duffie D, Schroder M, Skiadas C (1996) Recursive valuation of defaultable securities and the timing of resolution of uncertainty. Ann Appl Probab 6:1075–1090.
[17]. Bender C, Schweizer N, Zhuo J (2017) A primal–dual algorithm for BSDEs. Math Finance 27:866–901.
[18]. Bergman YZ (1995) Option pricing with differential interest rates. Rev Financial Stud 8:475–500.
[19]. Leland H (1985) Option pricing and replication with transaction costs. J Finance 40:1283–1301.
[20]. Avellaneda M, Levy A, Para´ s A (1995) Pricing and hedging derivative securities in markets with uncertain volatilities. Appl Math Finance 2:73–88.
[21]. Cre´pey S, Gerboud R, Grbac Z, Ngor N (2013) Counterparty risk and funding: The four wings of the TVA. Int J Theor Appl Finance 16:1350006.
[22]. Forsyth PA, Vetzal KR (2001) Implicit solution of uncertain volatility/transaction cost option pricing models with discretely observed barriers. Appl Numer Math 36:427– 445.
[23]. Powell WB (2011) Approximate Dynamic Programming: Solving the Curses of Dimensionality (Wiley, New York).
[24]. Yong J, Zhou X (1999) Stochastic Controls (Springer, New York).
[25]. Emmerich H (2003) The Diffuse Interface Approach in Materials Science: Thermodynamic Concepts and Applications of Phase-Field Models (Springer, New York), Vol 73.
[26]. El Karoui N, Peng S, Quenez MC (1997) Backward stochastic differential equations in finance. Math Finance 7:1–71.
[27]. Gobet E (2016) Monte-Carlo Methods and Stochastic Processes: From Linear to Non- Linear (Chapman & Hall/CRC, Boca Raton, FL).
[28]. Heinrich S (2006) The randomized information complexity of elliptic PDE. J Complexity 22:220–249.
[29]. Geiss S, Ylinen J (2014) Decoupling on the Wiener space, related Besov spaces, and applications to BSDEs. arXiv:1409.5322.
[30]. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167. Preprint, posted February 11, 2014.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, January 2024

[31]. Abadi M, et al. (2016) Tensorflow: A system for large-scale machine learning. 12th USENIX Symposium on Operating Systems Design and Implementation (USENIX Association, Berkeley, CA), pp 265–283.

[32]. Gobet E, Turkedjiev P (2017) Adaptive importance sampling in least-squares Monte Carlo algorithms for backward stochastic differential equations. Stoch Proc Appl 127:1171–1203.