Solving the Heat Equation with Neural Networks

Alexander Havrilla & Alden Pritchard
Carnegie Mellon University
Pittsburgh, PA 15213
alumhavr@andrew.cmu.edu; atpritch@andrew.cmu.edu

December 9, 2020

1 Introduction

The Deep Galerkin Method was first proposed in 2018 by Justin Sirignano and Konstantinos Spiliopoulos, who used the method to numerically solve the Black-Scholes equation for options pricing. This main advantage of the DGM is that it is mesh-free, and as a result does not suffer from the curse of dimensionality, allowing us to compute values in feasible time for higher-dimensional setups. Sirignano and Spiliopoulos proved rigorously that using a deep neural network, as the number of layers goes to infinity, the network approximation converges point wise to the true solution. This is done by choosing a loss function which consists of distance from some initial condition, distance from some boundary condition, and distance from the size of the differential operator applied to the function. By minimizing the sum of these three terms, the method aims to generate a function which closely approximates the initial condition, boundary condition, and a solution on the interior of the domain, thereby giving a numerical solution to the PDE.

2 Background

General commentary

2.1 PDEs and Heat Equation

2.2 Traditional Approximation Methods

3 Related Work

Black Scholes Paper and Toronto Review(These both come from University of Toronto, fix this name)

3.1 DGM Architecture

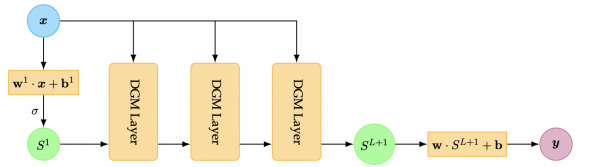

Figure 5.2: *Bird's-eye perspective of overall DGM architecture.*

The overall DGM network architecture consists of a fully connected layer, followed by some number of stacked DGM layers that take as input the original input x and the output of the previous DGM layer, with another fully connected layer at the end. We experimented with different depth networks in anticipation of a trade-off between training time and network accuracy as implied by the results from Sirignano and Spiliopoulos.

3.2 Individual DGM layer

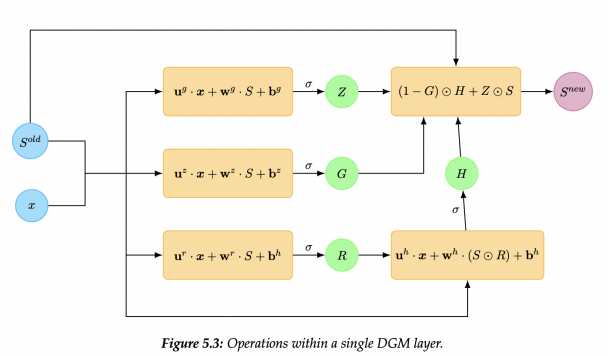

Figure 5.3: *Operations within a single DGM layer.*

Each individual DGM layer consists of four sums of linear maps and an output function. In total, each layer contains eight weight matrices, four bias vectors, and four activation functions, which are eventually combined in the layer's output function.

4 Methods

General Commentary

4.1 Architecture

4.2 Sampling Methodology

4.3 Initial/Boundary Conditions

5 Results

General Commentary

5.1 Depth of Network

5.2 Training Time

5.3 Number of Samples

5.4 Dependence on Dimension

6 Analysis

6.1 Depth of Network

6.2 Training Time

6.3 Number of Samples

6.4 Dependence on Dimension

7 Future Work

General Commentary

7.1 Different Architectures

7.2 Dealing Higher Dimensions

7.3 Other PDEs

References

References follow the acknowledgments. Use unnumbered first-level heading for the references. Any choice of citation style is acceptable as long as you are consistent. It is permissible to reduce the font size to `small` (9 point) when listing the references. **Remember that you can use a ninth page as long as it contains *only* cited references.**

[1] Al-Aradi A. Correia A. Naiff D. Jardim G. Solving Nonlinear High-Dimensional Partial Differential Equations via Deep Learning.

[2] Han. J. Solving High-dimensional PDEs Using Deep Learning. https://www.pnas.org/content/pnas/115/34/8505.fu

[3] Penko V. https://github.com/vitkarpenko/FEM-with-backward-Euler-for-the-heat-equation

[4] Sirignano. J, spiliopoulos K. DGM: A deep learning algorithm for solving partial differential equations. https://arxiv.org/abs/1708.07469