Deep Learning application to the solution of High-dimensional Partial Differential Equations and Backward Stochastic Differential Equations

Sadia Noor
Supervised by Dr. Javed Hussain Brohi

November 14, 2022

Table of Contents

Curse of Dimensionality

- It is well known that numerical algorithms for high-dimensional PDEs have long suffered the so called **"curse of dimensionality"**, namely, the complexity of the algorithm grows exponentially as the dimension grows.
- A limited number of cases where practical high-dimensional algorithms have been developed in the literature:
  - High dimensional linear parabolic PDEs: Feynman-Kac + Monte Carlo
  - Invscid Hamilton-Jabobi Equation: Algorithms based on Hopf formula (see Darbon and Osher 2016; [21])
  - Semi-parabolic PDEs with polynomial nonlinearity: Branching Duffusion method (see Henry-Labordere 2012; [21])
  - General semi-parabolic PDEs: Multi-level Picard Method (see Weinan, Hutzenthaler, et al. 2019; [21])

Curse of dimensionality

- However, **Deep Learning** has emerged in machine learning in recent years and has proven to be very effective in dealing with large class of high-dimensional problems in computer vision, natural language processing, time series analysis, etc. Deep learning might hold the key to tackle the curse of dimensionality.
- The bridge between high dimensional parabolic PDEs and Deep Learning is **Backward Stochastic Differential Equation**.

Semilinear Parabolic PDEs

We consider a general class of semilinear parabolic PDEs, which is of the form:

$$\frac{\partial u}{\partial t}(t,x) + \frac{1}{2}\operatorname{Tr}\left(\sigma\sigma^{\mathrm{T}}(t,x)\left(\operatorname{Hess}_x u\right)(t,x)\right) + \nabla u(t,x)\cdot\mu(t,x) + f\left(t,x,u(t,x),\sigma^{\mathrm{T}}(t,x)\nabla u(t,x)\right) = 0 \tag{1}$$

Note that:

- PDE is defined on $[0, T] \times \mathbb{R}^d$
- $\mu(t,x) : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ is a known vector-valued function,
- $\sigma(t,x) : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^{d\times d}$ is a known matrix-valued function,
- Tr denotes the trace of a matrix,
- f is a known nonlinear function.

The goal is to find a function $u(t,x)$ satisfying the PDE given a terminal condition $u(T,x) = g(x)$ for some specified function g.

Nonlinear Feynman-Kac formula

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, $W : [0, T] \times \Omega \to \mathbb{R}^d$ be a d-dimensional standard Brownian motion, and $\{\mathcal{F}_t\}_{t\in[0,T]}$ be the filtration generated by $\{W_t\}_{t\in[0,T]}$. Consider $\{\mathcal{F}_t\}_{t\in[0,T]}$-adapted process $\{(X_t, Y_t, Z_t)\}_{t\in[0,T]}$ s.t.

$$X_t = \xi + \int_0^t \mu(s, X_s)\, ds + \int_0^t \sigma(s, X_s)\, dW_s \tag{2}$$

$$Y_t = g(X_T) + \int_t^T f(s, X_s, Y_s, Z_s)\, ds - \int_t^T Z_s^T dW_s \tag{3}$$

Under suitable regularity assumption on μ, σ and f, one can prove existence and uniqueness of the solution process and $\forall t \in [0, T]$, and it holds $\mathbb{P}$-a.s. (see Pardoux and Peng 1992; [22]) that

$$Y_t = u(t, X_t), \quad Z_t = \sigma^T(x, X_t)\nabla u(t, X_t) \tag{4}$$

Conversion to BSDE

Now the solution u of the PDE(1) satisfies the following BSDE

$$u(t, X_t) - u(0, X_0) = -\int_0^t f\left(s, X_s, u(s, X_s), \sigma^T \nabla u(s, X_s)\right) ds + \int_0^t [\nabla u(s, X_s)]^T \sigma(s, X_s)\, dW_s \tag{5}$$

Discretization and Euler scheme

In order to numerically solve this BSDE, we discretize the equation (5). Time is discretized via a partition:

$$[0, T] : 0 = t_0 < t_1 < t_2 < \ldots < t_N = T \tag{6}$$

We consider the Euler-Maruyama scheme for $n \in \{0, 1, 2, \ldots, N-1\}$, let $\Delta t_n = t_{n+1} - t_n$ and $\Delta W_n = W_{t_{n+1}} - W_{t_n}$. We can get an approximation for X_t:

$$X_{t_{n+1}} - X_{t_n} \approx \mu\left(t_n, X_{t_n}\right)\Delta t_n + \sigma\left(t_n, X_{t_n}\right)\Delta W_n \tag{7}$$

We also approximate $u\left(t, X_t\right)$:

$$u\left(t_{n+1}, X_{t_{n+1}}\right) - u\left(t_n, X_{t_n}\right) = -f\left(t_n, X_{t_n}, u\left(t_n, X_{t_n}\right), \sigma^T \nabla u\left(t_n, X_{t_n}\right)\right) \Delta t_n + \left[\nabla u\left(t_n, X_{t_n}\right)\right]^T \sigma\left(t_n, X_{t_n}\right)\Delta W_n$$

Unknown gradient

The difference equations are:

$$\begin{aligned} X_{t_{n+1}} - X_{t_n} \approx& \mu\left(t_n, X_{t_n}\right)\Delta t_n + \sigma\left(t_n, X_{t_n}\right)\Delta W_n \\ u\left(t_{n+1}, X_{t_{n+1}}\right) - u\left(t_n, X_{t_n}\right) =& - f\left(t_n, X_{t_n}, u\left(t_n, X_{t_n}\right), \sigma^T \nabla u\left(t_n, X_{t_n}\right)\right) \\ & \Delta t_n + \left[\nabla u\left(t_n, X_{t_n}\right)\right]^T \sigma\left(t_n, X_{t_n}\right)\Delta W_n \end{aligned}$$

However, the critical issue here is that at any given time step, an estimate of $\nabla u\left(t_n, X_{t_n}\right)$ is not known and hard to estimate. A deep learning approach can be used to obtain an estimate of $\sigma^\top\left(t_n, X_{t_n}\right)\nabla u\left(t_n, X_{t_n}\right))$ at each time step. This approach is so called **"Deep BSDE method"**.

Deep BSDE method

A feed-forward neural network with parameters θ is used to estimate the gradients at each time. We take several Monte Carlo samples of $\{\Delta W_{t_n}\}_{0\leq n\leq N}$ and use the network to estimate the end value $\hat{u}\left(\{X_{t_n}\},\{W_{t_n}\}\right)$. We use the following loss function to improve the neural network.

$$\ell(\theta) = \mathbb{E}\left[\left|g\left(X_{t_N}\right) - \hat{u}\left(\{X_{t_n}\},\{W_{t_n}\}\right)\right|^2\right]$$

Stochastic Gradient Descent is used to improve the network.Once a number of iterations have occurred, a final estimate of the initial function value is reached.

Example 1: Recursive Pricing with Default Risk

We apply the Deep BSDE Solver to recursive pricing model with default risk. (see Weinan E, (22) where the equivalent form of this PDE was given as BSDE.) The nonlinear Black-Scholes equation in $[0, T] \times \mathbb{R}^{100}$ associated to recursive pricing with default risk becomes:

$$\frac{\partial u}{\partial t}(t,x) + \bar{\mu} x \cdot \nabla u(t,x) + \frac{\bar{\sigma}^2}{2} \sum_{i=1}^{d} |x_i|^2 \frac{\partial^2 u}{\partial x_i^2}(t,x)$$
$$-(1-\delta) Q(u(t,x)) u(t,x) - R u(t,x) = 0.$$

with the terminal condition $g(x) = \min\{x_1, \ldots, x_{100}\}$ for $x = (x_1, \ldots, x_{100}) \in \mathbb{R}^{100}$.

Example 1: Recursive Pricing with Default Risk

No. of iteration steps m	Mean of loss function	Mean of U^{θ_m}	Relative L^1-appr. error	Relative L^1-Abs. error	Runtime in Sec. for one realization of U^{θ_m}
0	123.582	46.0682	0.196017	11.2318	0
1000	40.172	52.3008	0.087246	4.9992	93
2000	26.0527	55.7642	0.026803	1.5358	174
3000	25.7447	56.9638	0.005867	0.3662	236
4000	25.528	57.0434	0.004478	0.2566	297
5000	25.4771	57.0478	0.004401	0.2522	361
6000	25.1507	57.0886	0.003689	0.2114	461

Table - The Table shows the Numerical simulations for the deep BSDE solver in the case of the PDE (11).

Example 1: Recursive Pricing with Default Risk

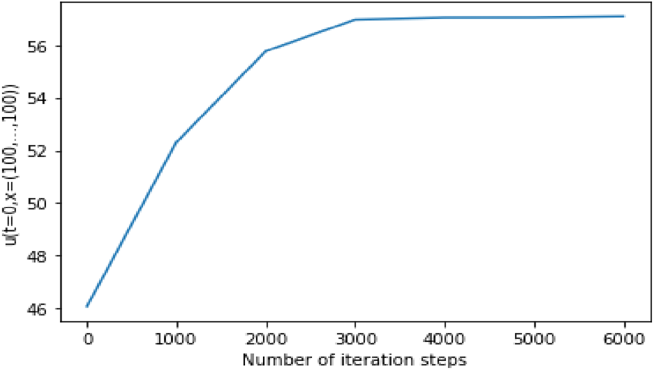

Figure - Plot of $\mathcal{U}^{\Theta_m}$ as an approximation of $u(t = 0, x = (100, \ldots, 100))$ against the number of iteration steps in the case of the 100-dimensional recursive pricing model.

Example 1: Recursive Pricing with Default Risk

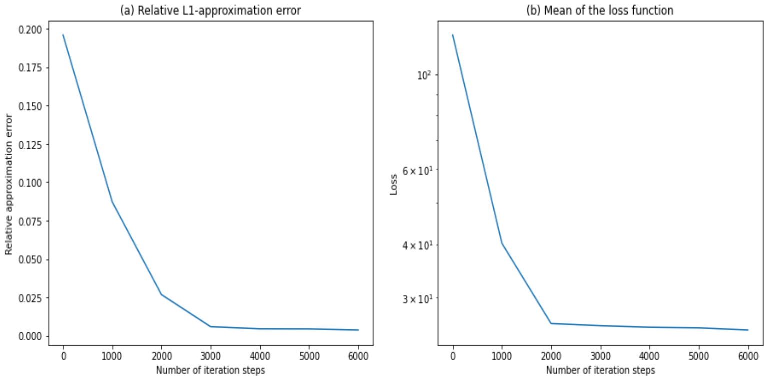

Figure - Relative L^1-approximation error of $\mathcal{U}^{\Theta_m}$ and mean of the loss function against $m \in \{1, 2, 3, \ldots, 6000\}$.

Example 2: Pricing with different interest rates for borrowing and lending

We apply the deep BSDE solver to a pricing problem of an European financial derivative in a financial market where the risk free bank account used for the hedging of the financial derivative has different interest rates for borrowing and lending. (see Weinan E, 22)

Example 2: Pricing with different interest rates for borrowing and lending

The PDE in this case shows that for all $t \in [0, T), x = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$ it holds that $u(T, x) = g(x)$ and

$$\frac{\partial u}{\partial t}(t,x) + \frac{\bar{\sigma}^2}{2} \sum_{i=1}^{d} |x_i|^2 \frac{\partial^2 u}{\partial x_i^2}(t,x)$$

$$- \min \left\{ R^b \left(u(t,x) - \sum_{i=1}^{d} x_i \frac{\partial u}{\partial x_i}(t,x) \right), R^l \left(u(t,x) \right.\right.$$

$$- \sum_{i=1}^{d} x_i \frac{\partial u}{\partial x_i}(t,x) = 0.$$

Example 2: Pricing with different interest rates for borrowing and lending

No. of iteration steps m	Mean of loss function	Mean of U^{θ_m}	Relative L^1-appr. error	Relative L^1-Abs. error	Runtime in Sec. for one realization of U^{θ_m}
0	61.0845	16.6819	0.2167754	4.6171	0
1000	41.3837	20.2499	0.0492558	1.0491	38
2000	40.57	21.2022	0.0045448	0.0968	66
3000	40.5826	21.2876	0.0005352	0.0114	93
4000	40.7411	21.2991	0.0000047	0.0001	129

Table - The Table shows the Numerical simulations for the deep BSDE solver in the case of the PDE (16).

Example 2: Pricing with different interest rates for borrowing and lending

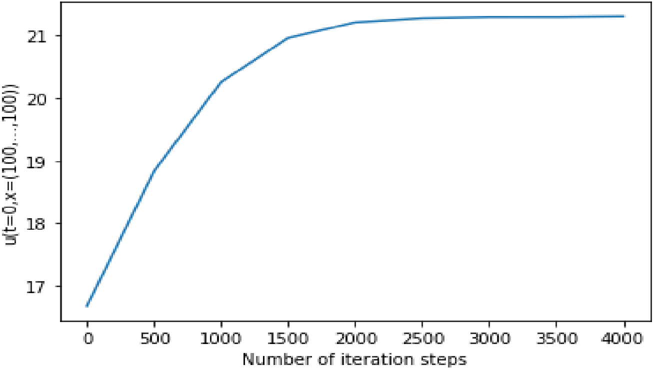

Figure - Plot of $\mathcal{U}^{\Theta_m}$ as an approximation of $u(t,x)$ in the case of the 100-dimensional pricing model with diff interest rates.

Example 2: Pricing with different interest rates for borrowing and lending

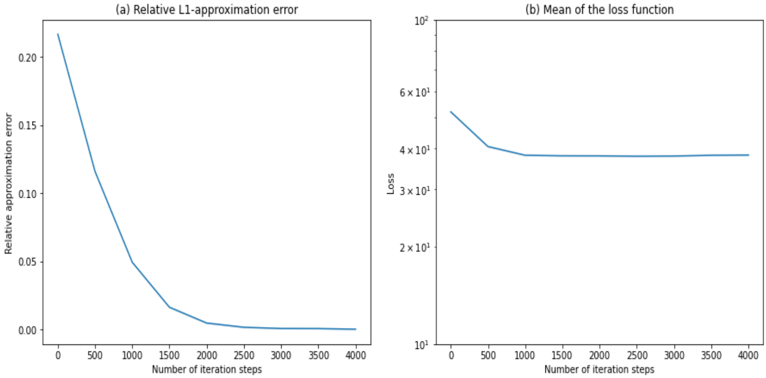

Figure - Relative L^1-approximation error of $\mathcal{U}^{\Theta_m}$ and mean of the loss function against $m \in \{1, 2, 3, \ldots, 4000\}$.

Conclusion

- Reviewed an algorithm for solving parabolic partial differential equations (PDEs) and backward stochastic differential equations (BSDEs) in high dimension,
- by making an analogy between the BSDE and reinforcement learning with the gradient of the solution playing the role of the policy function,
- and the loss function given by the error between the prescribed terminal condition and the solution of the BSDE.
- The policy function is then approximated by using deep neural network.
- Efficiency and accuracy of the algorithms was checked by implementing it on some 100-dimensional nonlinear PDEs used in finance.

References I

- Darbon, J., Osher, S. (2016). Algorithms for overcoming the curse of dimensionality for certain Hamilton Jacobi equations arising in control theory and elsewhere. Research in the Mathematical Sciences, 3(1), 1-26.
- Henry-Labordere, P., Oudjane, N., Tan, X., Touzi, N., Warin, X. (2019, February). Branching diffusion representation of semilinear PDEs and Monte Carlo approximation. In Annales de l'Institut Henri Poincar , Probabilit s et Statistiques (Vol. 55, No. 1, pp. 184-210). Institut Henri Poincar .
- Weinan, E, Martin Hutzenthaler, et al. (2019). On multilevel Picard numerical approximations for high-dimensional nonlinear parabolic partial differential equations and high-dimensional nonlinear backward stochastic differential equations . In: Journal of Scientific Computing 79.3, pp. 1534 1571.

References II

- Pardoux, Etienne and Shige Peng (1992). Backward stochastic differential equations and quasilinear parabolic partial differential equations . In: Stochastic partial differential equations and their applications. Springer, pp. 200 217.
- Hutzenthaler, M., Jentzen, A., Kruse, T. (2019). On multilevel Picard numerical approximations for high-dimensional nonlinear parabolic partial differential equations and high-dimensional nonlinear backward stochastic differential equations. Journal of Scientific Computing, 79(3), 1534-1571.
- Han, J., Jentzen, A. (2017). Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. Communications in mathematics and statistics, 5(4), 349-380.