Some neural network schemes for quadratic BSDEs

D. Bussell-Arnold, C.Garcia-Trillos

UCL

June 27, 2022

Motivation

- We consider an investor who wishes to optimise their expected exponential utility from terminal wealth.
- Let H_t be the price process of a stock whose stochastic log-spot price N_t follows the following Heston model:

$$dN_t = -\frac{1}{2}\nu_t dt + \sqrt{\nu_t} dB_t$$

$$d\nu_t = \lambda(\theta - \nu_t)dt + \sigma\sqrt{\nu_t} dW_t$$

- The investor's initial wealth is x and their trading strategies are deterministic functions of time π where $\pi(t)$ denoted the amount of money invested in stock H at time t.
- The wealth process of the investor is therefore described by

$$X_t^{x,\pi} = x + \int_0^t \frac{\pi(s)}{H_s} dH_s = x + \int_0^t \pi(s) dN_s$$

Motivation

- We can then solve the utility maximisation problem

$$V(x) = \sup_{\pi} \mathbb{E}[-\exp(-\gamma X_T^{x,\pi})], \quad x \in \mathbb{R}, \quad \gamma > 0$$

by finding the generator f of the $BSDE$

$$Y_t = 0 - \int_t^T Z_s dW_s + \int_t^T f(\nu_s, Z_s) ds$$

such that the process $L_t = \exp(-\gamma X_t^{x,\pi} + Y_t)$ is a supermartingale for all strategies π and is a martingale for a certain strategy π^{opt}.

- The generator is found to be $f(r,z) = \frac{\gamma}{2}(\rho^2 - 1)z^2 + \frac{1}{8\gamma^3} r + \frac{1}{2\gamma}\rho z\sqrt{r}$.
- The martingale property of $L_t^{\pi^{\text{opt}}}$ then allows us to find our value function as follows: $V(x) = -\exp[-\gamma(x + Y_0)]$

Motivation

- Quadratic BSDEs arise in finance from utility optimization problems with exponential utility functions.
- Consider the following Markovian quadratic BSDE on the filtered probability space $(\Omega, \mathcal{F}^W, \mathcal{F}_t^W, \mathbb{P})$:

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

$$X_t = x_0 + \int_0^t \mu(X_s)ds + \int_0^t \sigma(X_s)dW_s$$

- Richou and Chassagneux introduced a theoretical scheme for the resolution of the quadratic BSDE.
- Due to their well known success in higher dimensional problems we decide to use neural networks in order to approximate the expectations in the theroretical scheme and provide a full numerical implementation

Theoretical Scheme

$$Y_i^\pi = \mathbb{E}_{t_i}[Y_{i+1}^\pi + \Delta t_i f_N(X_i^\pi, Y_i^\pi, Z_i^\pi) - \Delta t_i Z_i^\pi H_i^R] \tag{1}$$

$$Z_i^\pi = \mathbb{E}_{t_i}[Y_{i+1}^\pi H_i^R] \tag{2}$$

where N is such that the projection ϕ of Z onto the ball κN such that $f_N(x, y, z) = f(x, y, \phi(z))$ is N-Lipschitz.

- We note that $(H_i^R)_{0<n}$ satisfies the following properties:

$$\mathbb{E}_i[H_i^R] = 0 \tag{3}$$

$$\Delta t_i \mathbb{E}_i[(H_i^R)^T H_i^R] = \Delta t_i \mathbb{E}_i[H_i^R (H_i^R)^T] = c_i I_{d\times d} \text{ and } \frac{\lambda}{d} \leq c_i \leq \frac{\Lambda}{d}, \tag{4}$$

where λ, Λ are positive constants that do not depend on R, for sufficiently large R.

qNeural Scheme 1

We use two neural networks $\mathcal{U}_i, \mathcal{Z}_i \in \mathcal{NN}^{\rho}_{d,1,L,m}$ with two square error loss functions $\mathcal{L}_1, \mathcal{L}_2$. We use stochastic gradient descent in order to obtain the optimized solutions $\hat{\mathcal{U}}, \hat{\mathcal{Z}}$.

- Initialise from estimation $(\hat{\mathcal{U}}, \hat{\mathcal{Z}})$ of (Y_{t_n}, Z_{t_n}) with $(\hat{\mathcal{U}}_n, \hat{\mathcal{Z}}_n) = (g(X_{t_n}), 0)$
- For $i = N-1, \ldots, 0$, given $\hat{\mathcal{U}}_{i+1}$ and $\hat{\mathcal{Z}}_{i+1}$ use a pair of neural networks $(\mathcal{U}_i, \mathcal{Z}_i) \in \mathcal{NN}^{\rho}_{d,1,L,m}(\mathbb{R}^{N_m}) \times \mathcal{NN}^{\rho}_{d,1,L,m}(\mathbb{R}^{N_m})$ and compute by stochastic gradient descent the minimizers $\hat{\mathcal{U}}_i$ and $\hat{\mathcal{Z}}_i$ of the expected quadratic loss functions $\mathcal{L}^1_i$ and $\mathcal{L}^2_i$ respectively:

$$\mathcal{L}^1_i(\mathcal{U}_i) := \mathbb{E}|\hat{\mathcal{U}}_{i+1}(X_{t_{i+1}}) - F(t_i, X_{t_i}, \mathcal{U}_i(X_{t_i}), \mathcal{Z}_i(X_{t_i}), \Delta t_i)|^2$$

$$\mathcal{L}^2_i(\mathcal{Z}_i) = \mathbb{E}|\hat{\mathcal{U}}_{i+1} H^R_i - \mathcal{Z}_i|^2$$

where

$$F(x, y, z, \Delta_{t_i}) = y - f_N(x, y, z)\Delta t_i + z \Delta t_i H_i$$

Over Parameterisation Theorem

Theorem

Suppose $\rho = ReLu$ $m \geq O\left(\frac{(ML/\delta)^{30} d \log^2(\varepsilon^{-1})}{b}\right)$, $b \in [M]$ *and for every pair* $i, j \in [M]$ *we have* $||x_i - x_j||_2 > \delta$ *where* δ *is the minimum distance between sample points. Starting from random initialization, with probability at least* $1 - e^{-O(log^2(m))}$, *stochastic gradient descent with learning rate* $\omega = O(\frac{b\delta d}{M^5 L^2 m log^2(m)})$ *and mini batch size* b *ensures that the squared error loss function is less than* ε *in* $T = O(\frac{n^7 L^2 log^2 m}{b\delta^2} log(\frac{1}{\varepsilon}))$ *iterations.*

Allen-Zhu,18

Theorem and Methods

Theorem

For $\alpha \in (0, 1/2)$ *with* $N = n^{\alpha}, R = \log(n)$, *we have for all* $\eta > 0$

$$\mathbb{E}\left[\sup_{0 \le i \le n} |Y_i^{\pi} - \hat{\mathcal{U}}_i|^2\right] \le C_{\alpha,\eta} h^{1-\eta}$$

where $m \ge O\left(\frac{(ML/\delta)^{30} d \log^2(\varepsilon^{-1})}{b}\right)$, $b \in [M]$

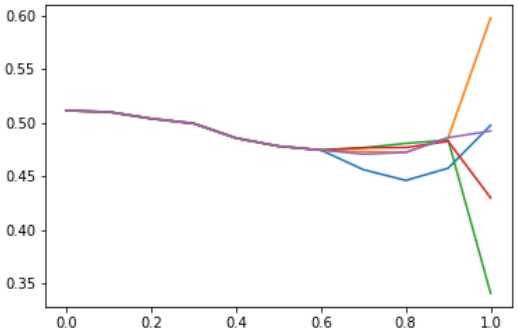
- Monte Carlo Sampling error- using classical variance bounding methods
- Neural network estimation errors- using an Over Parameterisation Theorem
- Accumulating error terms contributed by Z terms- reduced by making a change of measure
- Martingale Representation Theorem used to bound terms resulting from our capping H

Results qNeural 1

$$X_t = X_0 + \int_0^t \nu X_s dW_s$$

$$Y_t = g(X_1) + \int_t^1 \frac{a}{2}|Z_s|^2 ds - \int_t^1 Z_s dW_s$$

- $a = 1, \nu = 1, g(x) = \frac{\sum_{i=1}^d x_i}{(\sum_{i=1}^d x_i^2)+0.01}$
- $M = 2000, n = 10, m = 1000, d = 10$
- $Y_0^\pi = 0.511$
- Relative error= 2.7%



qNeural Scheme 2

We directly approximate the Markovian representative, $u(t,x)$, using a single deep neural network in the following way:

- Use forward Euler-Maruyama schemes:

$$X^{\pi}_{i+1} = X^{\pi}_i + b(X^{\pi}_i)\Delta t_i + \sigma(X^{\pi}_i)\Delta W_i$$

$$Y^{\pi}_{i+1} = Y^{\pi}_i + f_N(X^{\pi}_i, Y^{\pi}_i, Z^{\pi}_i)\Delta t_i + (Z^{\pi}_i)^T \Delta t_i H_i$$

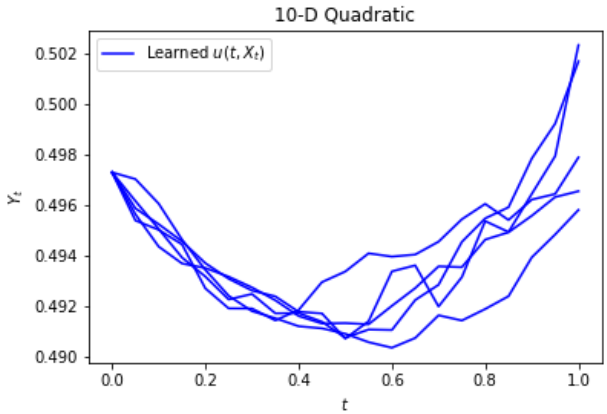
- We minimise the loss function given by:

$$\sum_{m=1}^{M}\sum_{i=1}^{n-1} |Y^{\pi}_{m,i+1} - Y^{\pi}_{m,i} - f^m_i \Delta t_i - (Z^{\pi}_{m,i})^T \Delta t_i H_i|^2 + \sum_{m=1}^{M} |Y^{\pi}_{m,n} - g(X^{\pi}_{m,n})|^2$$

- Z is obtained via automatic differentiation

Results qNeural 2

- $a = 1, \nu = 1, g(x) = \frac{\sum_{i=1}^{d} x_i}{(\sum_{i=1}^{d} x_i^2)+0.01}$
- $M = 1000, n = 20, d = 10$
- $Y_0^{true} = 0.525$
- $Y_0^{\pi} = 0.498$
- Relative error= 5.0%



Comparison

qNeural 1

- Advantages:
- by decomposing the global problem into smaller ones, we may expect to help the gradient descent method to provide estimations closer to the real solution.
- at each time step, we initialize the weights and bias of the neural network to the weights and bias of the previous time step treated. This allows us to start with a value close to the solution, hence avoiding local minima which are too far away from the true solution.
- Disadvantgaes:
- Parameters must be calculated for a total of $2(N-1)$ neural networks. This can possibly result in overfitting and greatly increase computation time.

Comparison

qNeural 2

- Advantages:
- Using a single global neural network greatly reduces the amount of parameters that need to be learned and can reduce the potential of overfitting.
- Number of parameters is independent of the number of time steps in the discretization.
- Disadvantages
- Accuracy suffers due to learing the equation using a forward scheme. We may not be able to match the terminal condition with great accuracy as we do not initialise our algorithm at $g(X_T)$.

Future work

- Investigate convergence rate numerically. How do we train our neural network in order to realise the convergence rate we have proved?
- Provide a theoretical result (proof) for the convergence of qNeural 2.
- Can we generalise this to systems of qBSDEs?
- Is there a procedure to determine our choice of α?