A DEEP LEARNING ALGORITHM FOR OPTIMAL INVESTMENT STRATEGIES UNDER MERTON'S FRAMEWORK

DAEYUNG GIM AND HYUNGBIN PARK

Abstract. This paper treats Merton’s classical portfolio optimization problem for a market participant who invests in safe assets and risky assets to maximize the expected utility. When the state process is a $d$-dimensional Markov diffusion, this problem is transformed into a problem of solving a Hamilton–Jacobi–Bellman (HJB) equation. The main purpose of this paper is to solve this HJB equation by a deep learning algorithm: the deep Galerkin method, first suggested by J. Sirignano and K. Spiliopoulos. We then apply the algorithm to get the solution to the HJB equation and compare with the result from the finite difference method.

1. Introduction

Consider the following expected utility maximization problem:

$$\max_{\pi} \frac{1}{p} \mathbb{E} \left[ (X_T^\pi)^p \mid X_t = x, Y_t = y \right],$$

where $\pi$ is a portfolio, $X^\pi$ a wealth process and $Y$ a state variable with the utility function $(1/p)x^p := U(x)$. This kind of problem is first suggested by R. C. Merton [29], which is the most fundamental and pioneering in economics. The Merton problem has played as a key for an investor’s wealth allocation in several assets under some market circumstances. Since then there have been lots of studies about Merton problem under various conditions. In [4], F. E. Benth et al. studied Merton problem under the Black-Scholes setting by using the OU type stochastic volatility model. C. Kühn and M. Stroh studied optimizing portfolio of Merton problem under a limit-ordered market in view of a shadow price in [21]. The research on the optimal investment based on inside information and drift parameter uncertainty was conducted by A. Danilova et al. in [11]. M. Nutz [30] studied the utility maximization in a...
semimartingale market setting with the opportunity process. S. L. Hansen [19] suggested an optimal investment strategy with investors’ partial and private information. J. Pedersen and G. Peskir applied the Lagrange multiplier to solve nonlinear mean-variance optimal portfolio selection problem in [31]. Also there was research on the optimal portfolio strategies using over-reaction and under-reaction in [9]. In [26], Z. Liang and M. Ma researched a robust Merton problem using the constant relative/absolute risk aversion utility functions under the time-dependent sets of confidence.

In this paper we follow the overall market setting in [17] and induce the so-called Hamilton–Jacobi–Bellman equation under time variable $t$, variable $x$ representing wealth process and variable $y = (y_1, \ldots, y_d)$ from the $d$-dimensional state variable. We can optimize the portfolio by means of finding a solution to the HJB equation. By using some properties including homotheticity and concaveness, we eliminate the optimizing term to change the HJB equation into a nonlinear partial differential equation.

Under this circumstance we face with the problem of solving nonlinear PDEs. Because in general most PDEs do not have analytic solutions, there exist several well-known numerical tools. These classical approaches can be found in [1] and [8].

At the same time there has been some studies about solving PDEs with a deep neural network. I. S. Kang et al. [24] and I. E. Lagaris [23] suggested the neural network algorithm on a fixed mesh. Also in [28], R. S. Beidokhti and A. Malek suggested the numerical hybrid DNN optimizing method. However in case of the higher dimension of PDEs, these grid-based methods would be computationally inefficient: a curse of dimensionality.

Recently there have been several researches to get rid of the curse of dimensionality using machine learning techniques. J. Han et al. and M. Hutzenthaler et al. suggested a deep backward stochastic differential equation method with the Feynman–Kac formula ([14,18]).

The deep learning algorithm mainly used in this paper is the deep Galerkin method introduced in [35]. It is computationally efficient since there does not need to make any mesh or grid. We define a loss functional to minimize $L^2$-norm about the desired differential operator and other conditions from the PDE. To make the loss small enough as we want, we sample random points from the domain and optimize by means of stochastic gradient descent. After deriving surfaces, we also apply the finite difference method (FDM) in order to compare surfaces from both algorithms: DGM and FDM. For further research on the deep Galerkin method, see [2] and [3].

This paper is organized as follows. In Section 2, we start by describing the general setting of this paper, and induce the partial differential equation with optimizing term: the HJB equation. The deep Galerkin method algorithm and neural network approximation theorem from [35] are presented in Section 3, with some part of code for each step of DGM algorithm. Numerical test of the algorithm is presented in Section 4. Specifically, we model 2 dimensional state
process by the OU process and the CIR process, return process by the Heston model. Then we use the calibrated parameters from [10] and [16]. We display the solution surface at each fixed time in some pre-determined domain of the state variable. We finally analyze surfaces from the deep Galerkin method and those from the finite difference method. Conclusions can be found in Section 5, and proofs of neural network approximation theorem are in Section 6.

2. Optimal investment problem

In the case that an economic agent is in time interval $[0, T]$, the problem is that he or she has to decide how to invest in several risky assets or safe assets as time goes by, starting with the initial wealth. This problem was first suggested by Merton in the 1960s: Merton problem, known as a utility maximization problem. The aim of the agent is to establish a portfolio strategy in such a way of maximizing utility under some conditions. In this section we describe the general setting of this paper, and induce the HJB equation. We finally reach to a nonlinear PDE by using some properties. The above problem is equivalent to a matter of finding a solution of the equation.

2.1. Market with the Merton problem

We first start by describing market with the following framework. Assume that the market has $n + 1$ assets $S^{(0)}, S^{(1)}, \ldots, S^{(n)}$, where $S^{(0)}$ is safe and $S^{(1)}, \ldots, S^{(n)}$ are risky. One can make a decision to the investment by a $d$-dimensional state variable $Y = (Y^{(1)}, \ldots, Y^{(d)})$ satisfying:

\[
dY_t = b(Y_t) \, dt + a(Y_t) \, dW_t,
\]

where $W = (W^{(1)}, \ldots, W^{(d)})$ denotes a standard Brownian motion.

Let $r$ be the interest rate, $\mu$ be the excess returns, and $\sigma$ be the volatility matrix. We also assume that the prices of the assets satisfy:

\[
dS^{(0)}_t = rS^{(0)}_t \, dt,
\]

\[
\frac{dS^{(i)}_t}{S^{(i)}_t} = r \, dt + dR^{(i)}_t, \quad 1 \leq i \leq n,
\]

where $R = (R^{(1)}, \ldots, R^{(n)})$ denotes the cumulative excess return satisfying:

\[
dR^{(i)}_t = \mu_i(Y_t) \, dt + \sum_{j=1}^n \sigma_{ij}(Y_t) \, dZ^{(j)}_t, \quad 1 \leq i \leq n.
\]

$\rho = (\rho_{ij}) = d \langle Z, W \rangle_t / dt$ denotes the cross correlations between the Brownian motions $Z$ and $W$. $\Sigma = \sigma \sigma^T = d \langle R, R \rangle_t / dt$ is the matrix of quadratic covariance of returns, and $\Upsilon = \sigma \rho a^T = d \langle R, Y \rangle_t / dt$ denotes the correlation between the return and the state process.
In the market, an investor buys the risky assets by a portfolio \( \pi = (\pi_t(i))_{t \geq 0} \) for \( i = 1, \ldots, n \). The wealth process \( X^\pi = (X^\pi_t)_{t \geq 0} \) corresponding to the portfolio satisfies

\[
\frac{dX^\pi_t}{X^\pi_t} = r\,dt + \pi^T_t\,dR_t, \quad X^\pi_0 \geq 0.
\]

Observe first that the portfolio process \((\pi_t)_{t \geq 0}\) is \( \mathcal{F}_t \)-measurable, where the filtration \( \mathcal{F} = (\mathcal{F}_t)_{t \geq 0} \) is generated by the return \( R \) and state variable \( Y \). It might be clear in light of the investor’s eyes: he or she has all informations about state and asset return from time \( t = 0 \) to the current time. Note also the portfolio process is integrable with respect to the return process \( R \). By the Merton problem, we assume the investors’ utility function is defined by the following:

\[
U(x) = \frac{1}{p} x^p, \quad 0 < p < 1.
\]

For fixed wealth \( x \) and state \( y = (y_1, \ldots, y_d) \) satisfying (2.1) and (2.5), our aim is to maximize the conditional expectation of terminal wealth utility given wealth and state at time \( t \), that is

\[
\max_{(\pi_t)_{t \geq 0}} \frac{1}{p} E[(X^\pi_T)^p \mid X_t = x, Y_t = y].
\]

### 2.2. The Hamilton–Jacobi–Bellman equation

Now we substitute the problem of utility maximization to that of solving the PDE, namely the Hamilton–Jacobi–Bellman equation. There needs to be some definitions before approaching to the HJB equation.

**Definition.** A portfolio process \( \pi = (\pi_t)_{t \geq 0} \) is called an admissible portfolio if

- For every \( t \in [0, T] \) and \((x, y) \in D \subset \mathbb{R} \times \mathbb{R}^d \), \( \pi(t, x, y) \in U \), where \( U \subset \mathbb{R} \) is a fixed subset.
- For any given initial points \((t, x)\) and \( y = (y_1, \ldots, y_d) \), the following SDE has a unique solution:

\[
\begin{align*}
    dX^\pi_s &= rX^\pi_s \, ds + \pi^T_s \, dR_s, \\
    X^\pi_t &= x.
\end{align*}
\]

- For any given initial point \((t, y) = (t, y_1, \ldots, y_d) \), the following SDE has a unique solution:

\[
\begin{align*}
    dY_s &= b(Y_s) \, ds + a(Y_s) \, dW_s, \\
    Y_t &= y.
\end{align*}
\]

By now we assume the portfolio \( \pi \) is admissible.

**Definition.** Let \( U \) be an investor’s utility function.
For each $\pi$, we define the expected value function $V^\pi$ as
\[ V^\pi(t, x, y) = \mathbb{E}[U(X_T^\pi) | X_t = x, Y_t = y], \]
given (2.6) and (2.7).

We define the optimal value function $V$ as
\[ V(t, x, y) = \sup_\pi V^\pi(t, x, y). \]

The following theorem justifies a conversion from the way of finding optimal portfolio to that of solving PDEs having optimizing term. Heuristic process for deriving the HJB equation is in [5, Chapter 19], in the way of limiting procedures in dynamic programming.

**Theorem 2.1.** Assume the following.

- The market has a safe asset $S^{(0)}$ whose dynamics is expressed in (2.2).
- The market has $n$ risky assets satisfying (2.3), with the return process $R$ following the diffusion (2.4).
- There exists an optimal portfolio $\hat{\pi} = (\hat{\pi}_t^{(1)}, \ldots, \hat{\pi}_t^{(n)})_{t \geq 0}$.
- The optimal value function $V$ is regular, that is, $V \in C^{1,2,2}$ with respect to $(t,x,y)$, $y = (y_1, \ldots, y_d)$.

Then the following hold:

1. $V$ satisfies the Hamilton–Jacobi–Bellman equation
\[
V_t + b^T(\nabla_y V) + \frac{1}{2} \text{tr}[a^T(\nabla^2_y V) a] + rxV_x
+ \sup_\pi \left[ \pi^T (\mu V_x + \Upsilon(\nabla_y V_x)) x + \frac{1}{2} x^2 V_{xx} \pi^T \Sigma \pi \right] = 0,
\]
   
   $$(t,x,y) \in [0,T] \times D, \quad V(0,x,y) = U(x), \quad (x,y) \in D.$$  

2. An optimizing term in the above equation can be achieved by $\pi = \hat{\pi}$:
\[
\hat{\pi} = \arg \sup_\pi \left[ \pi^T (\mu V_x + \Upsilon(\nabla_y V_x)) x + \frac{1}{2} x^2 V_{xx} \pi^T \Sigma \pi \right].
\]

If we define the optimal value function as
\[ V(t, x_1, \ldots, x_d) = \sup_{(x_1, \ldots, x_d) \geq 0} \mathbb{E} \left[ \frac{1}{P} (X_T^X)^p \mid X_t^X = x, Y_t^{(1)} = y_1, \ldots, Y_t^{(d)} = y_d \right], \]
by Theorem 2.1 with the Itô formula, one can derive the Hamilton–Jacobi–Bellman equation:
\[
V_t + b^T(\nabla_y V) + \frac{1}{2} \text{tr}[a^T(\nabla^2_y V) a] + rxV_x
+ \sup_\pi \left[ \pi^T (\mu V_x + \Upsilon(\nabla_y V_x)) x + \frac{1}{2} x^2 V_{xx} \pi^T \Sigma \pi \right] = 0,
\]
where the terminal condition of (2.8) is $V(T, x, y) = (1/p) x^p$. Note that $\nabla_y V = (V_{y_1}, \ldots, V_{y_d})$ and $\nabla^2_y V = (V_{y_iy_j})_{1 \leq i,j \leq d}$ stand for the gradient and
the Hessian of \( V \) with respect to \( y = (y_1, \ldots, y_d) \), respectively. Because of the concaveness of \( V \) in \( x \) and \( \sup_\pi (\pi^T b + \frac{1}{2} \pi^T A \pi) = -\frac{1}{2} b^T A^{-1} b \) for negative definite matrix \( A \), (2.8) becomes

\[
V_t + b^T (\nabla_y V) + \frac{1}{2} \text{tr}[a^T (\nabla_y^2 V) a] + r x V_x \\
- (\mu V_x + \Upsilon (\nabla_y V_x)) \frac{\Sigma^{-1}}{2 V_{xx}} (\mu V_x + \Upsilon (\nabla_y V_x)) = 0,
\]

with the corresponding optimal portfolio

\[
\pi = \pi(t, x, y_1, \ldots, y_d) = -\frac{1}{x V_{xx}} \Sigma^{-1} (\mu V_x + \Upsilon (\nabla_y V_x)).
\]

Since the utility function is homothetic, we define the reduced value function \( u \) as

\[
V(t, x, y_1, \ldots, y_d) = \frac{1}{p} x^p u(t, y_1, \ldots, y_d).
\]

If we put (2.10) into (2.9) and divide each component by \( x^p \), (2.9) becomes

\[
u_t + (b^T - q \mu^T \Sigma^{-1} \Upsilon) \nabla_y u + \frac{1}{2} \text{tr}[a^T (\nabla_y u) a] \\
+ (pr - q \mu^T \Sigma^{-1} \mu) u - \frac{q}{2 u} (\nabla_y u)^T \Upsilon \Sigma^{-1} \Upsilon (\nabla_y u) = 0,
\]

where the terminal condition of (2.11) is \( u(T, y_1, \ldots, y_d) = 1 \). In (2.11), we set \( q = p/(p-1) \) for simplicity. Also the following is the reduced optimal portfolio:

\[
\pi(t, y_1, \ldots, y_d) = \frac{1}{1 - p} \left( \Sigma^{-1} \mu + \Sigma^{-1} \Upsilon (\nabla_y u) \frac{1}{u} \right).
\]

3. Deep Galerkin method

Now we investigate how to solve the PDEs such as (2.11). Since only few PDEs have analytic solutions, there are well-known numerical tools including the Monte Carlo method exemplified by the Feynman–Kac theorem and the finite difference method. However one of the most difficult facts is a curse of dimensionality. In particular in grid-based numerical methods, the number of mesh points grows explosively as the dimension goes higher, so in [35] J. Sirignano et al. suggest a DNN-based algorithm for approximating solution of PDEs: the deep Galerkin method (DGM), such that there is no need to make any mesh.

With the parametrized deep neural network, say \( f \), a loss functional \( f \mapsto J(f) \) is defined to minimize \( L^2 \)-norm about the desired differential operator and terminal condition. To make the loss small enough as we want, the network samples random points from the pre-determined domain and is optimized by means of the stochastic gradient descent. In this section we first introduce the DGM algorithm. We then state the approximation theorem in order to justify this new algorithm.
3.1. Algorithm

Let \( u = u(t, y) \) be an unknown function which satisfies the PDE:

\[
\partial_t u(t, y) + \mathcal{L} u(t, y) = 0, \quad (t, y) \in [0, T] \times D,
\]

\[
u(t, y) \in D,
\]

where \( D \subseteq \mathbb{R}^d \). Our aim is to express the solution of (3.1) as a neural network function \( f = f(t, y; \theta) \) in place of \( u \). \( \theta = (\theta^{(1)}, \ldots, \theta^{(K)}) \) denotes a vector of network parameters.

Define a loss functional \( J := J_1 + J_2 \) with

\[
J_1(f) := \| \partial_t f(t, y; \theta) + \mathcal{L} f(t, y; \theta) \|_{[0,T] \times D, \nu_1}^2,
\]

\[
J_2(f) := \| f(T, y; \theta) - u_T(y) \|_{D, \nu_2}^2.
\]

Note that above terms are expressed in terms of \( L^2 \)-norm, that is, \( \| h(y) \|_{Y, \nu}^2 = \int_Y |h(y)|^2 \nu(dy) \). Each functionals \( J_1 \) and \( J_2 \) determine how well the approximation has conducted in view of the PDE differential operator and terminal condition. The aim is to find a parameter \( \hat{\theta} \) in such a way of minimizing \( J(f) \), equivalently,

\[
\hat{\theta} = \arg\min_{\theta} J(f(t, y; \theta)).
\]

As the error \( J(f) \) goes smaller, the approximated function \( f \) would get closer to the solution \( u \). Hence \( f(t, y; \hat{\theta}) \) might be the best approximation of \( u(t, y) \).

The algorithm of DGM is as follows:

1. Set initial values of \( \theta_0 = (\theta^{(1)}_0, \ldots, \theta^{(K)}_0) \) and determine the learning rate \( \beta_n \).
2. Sample random points \( (t_n, y_n) \) in \([0, T] \times D\) according to probability density \( \nu_1 \). Likewise, pick random points \( w_n \) from \( D \) with density \( \nu_2 \).
3. Calculate the error for the randomly sampled points \( s_n = \{(t_n, y_n), w_n\} \):

\[
L(\theta_n, s_n) = \left( \partial_t + \mathcal{L} f(t_n, y_n; \theta_n) \right)^2 + \left( f(T, w_n; \theta_n) - u_T(w_n) \right)^2.
\]
4. Use the stochastic gradient descent at \( s_n \):

\[
\theta_{n+1} = \theta_n - \beta_n \nabla_{\theta} L(\theta_n, s_n).
\]
5. Repeat until \( \| \theta_{n+1} - \theta_n \| \) is small enough.

The following is some part of code for each step of DGM algorithm:

```python
# 1-1. Initializing the neural network parameter
oper_init = tf.global_variables_initializer()

# 1-2. Initializing the learning rate
lrn_rate = tf.train.exponential_decay(init_lrn_rate, glob_step, dec_step, dec_rate, staircase=True)
```
optimizer = tf.train.AdamOptimizer(1n_rate).minimize(loss_tnsr)

# 2-1. Generating random samples: interior of the domain
  t_int = np.random.uniform(low=0, high=T, size=[nSim_int,1])
  y1_int = np.random.uniform(low=y1_low, high=y1_high, size=[nSim_int,1])
  y2_int = np.random.uniform(low=y2_low, high=y2_high, size=[nSim_int,1])

# 2-2. Generating random samples: terminal condition
  t_ter = T * np.ones(nSim_ter,1)
  y1_ter = np.random.uniform(low=y1_low, high=y1_high, size=[nSim_ter,1])
  y2_ter = np.random.uniform(low=y2_low, high=y2_high, size=[nSim_ter,1])

# 3. Calculating L^2-error of differential operator / terminal condition
# differential operator
  J1 = tf.reduce_mean(tf.square(diff_u))
# terminal condition
  J2 = tf.reduce_mean(tf.square(fitted_ter - target_ter))
  J = J1 + J2

# 4. Stochastic gradient descent step
for k in range(steps_per_sample):
  loss, J1, J2, k = sess.run([loss_tnsr, J1_tnsr, J2_tnsr, optimizer],
   feed_dict={t_int_tnsr:t_int, y1_int_tnsr:y1_int, y2_int_tnsr:y2_int, t_ter_tnsr:t_ter, y1_ter_tnsr:y1_ter, y2_ter_tnsr:y2_ter})

3.2. Neural network approximation

The following neural network approximation theorem is stated in [35]. In other words, there exists a collection of approximated neural network functions that converge to a solution of quasilinear parabolic PDEs.

Theorem 3.1. Define $C^n$ as a collection of DNN functions with $n$ hidden neurons in a single hidden layer. Assume that $u = u(t, y)$ is an unknown solution for (3.1). Under certain conditions in [35], there exists a neural network function $f^n$ with $n$ hidden neurons such that the following hold:
(1) $J(f^n) \to 0$ as $n \to \infty$,
(2) $f^n \xrightarrow{\text{strongly}} u$ in $L^\rho([0,T] \times D)$ as $n \to \infty$, where $\rho < 2$.

Some part of proofs for our formulation in this paper is in Section 6. Further details including conditions and proofs are in Section 7 and Appendix A in [35].

4. Numerical test

The key purpose of this section is to solve (2.11) with the deep Galerkin method and compare the numerical solution with the one derived by the well-known finite difference method.

4.1. Model settings

We first set some specific settings of the market model. For our experiment we assume that there are two ways of decision for trading, i.e., 2 dimensional state variable $Y = (Y^{(1)}, Y^{(2)})$. Let $Y^{(1)}$ be the Ornstein-Uhlenbeck (OU) process and $Y^{(2)}$ be the Cox-Ingersoll-Ross (CIR) process. This state variable is expressed by the following matrix form:

$$
\begin{pmatrix}
dY^{(1)}_t \\
dY^{(2)}_t
\end{pmatrix} =
\begin{pmatrix}
\theta^{(1)}(k^{(1)} - Y^{(1)}_t) \\
\theta^{(2)}(k^{(2)} - Y^{(2)}_t)
\end{pmatrix} dt +
\begin{pmatrix}
a^{(1,1)}Y^{(2)}_t \\
a^{(2,1)}Y^{(2)}_t \\
a^{(1,2)} \sqrt{Y^{(2)}_t} \\
a^{(2,2)} \sqrt{Y^{(2)}_t}
\end{pmatrix} dW_t.
$$

We also assume that there is a risky asset $S^{(1)}$ in the market, that is:

$$dS^{(1)}_t = rS^{(1)}_t dt + S^{(1)}_t dR_t,$$

where the cumulative excess return $R$ follows the diffusion:

$$dR_t = Y^{(1)}_t dt + \sigma \sqrt{Y^{(2)}_t} dZ_t, \quad (\sigma \in \mathbb{R})$$

which is known as the Heston model. In this case the correlation matrix between $Z$ and $W$ is of the form $\rho = (\rho_1, \rho_2)$ satisfying:

$$\langle Z, W^{(i)} \rangle = \rho_i dt, \quad 1 \leq i \leq 2.$$

4.2. Calibration

Now for the next step we need to set the value of parameters. Let $\mathcal{P}$ be a vector of parameters to be determined given by

$$\mathcal{P} = \left\{ \theta^{(1)}, \theta^{(2)}, k^{(1)}, k^{(2)}, a^{(1,1)}, a^{(1,2)}, a^{(2,1)}, a^{(2,2)}, \sigma, \rho_1, \rho_2 \right\}.$$

We shortly introduce the calibrating process using the nonlinear least squares optimization from the market data. For more detail, see [10] and [16].

Define the Percentage Mean Squared Error (PMSE) between the price $C_{\text{market}}$ from the market and the model price $C_{\text{model}}$ of the European call option derived from the double Heston model in [16] and [25]:

$$\text{PMSE} := \sum_{j=1}^{n} w_j \left( \frac{C_{\text{market}}(S^{(0)}, K_j, T_j, r) - C_{\text{model}}(S^{(0)}, K_j, T_j, r, \mathcal{P})}{C_{\text{market}}(S^{(0)}, K_j, T_j, r)} \right)^2,$$
where the weights $w_j$ satisfies:

$$ w_j = \frac{1}{\sqrt{|C^{(j)}_{ask} - C^{(j)}_{bid}|}}. $$

The optimal parameter vector $\mathcal{P}^*$ is determined by the following nonlinear least squares problem

$$ \mathcal{P}^* = \arg \inf \mathcal{P} \text{MSE}. $$

Table 1 shows the optimal parameters on the observed market data from the S&P500 index at the close of the market in September 2010.

|          | $(1)$          | $(2)$          | $(1,1)$        | $(2,1)$        |
|----------|----------------|----------------|----------------|----------------|
| $\theta$ | 0.1646         | 0.2333         | -0.6594        | -0.6692        |
| $k$      | 0.1301         | 0.0958         | 0.7518         | 0.7431         |
| $a$      | -0.6594        | -0.6692        | 0.7518         | 0.7431         |
| $\rho_1$ | -0.2949        | -0.2919        | 0.7072         | 0.7266         |
| $\rho_2$ |                |                |                |                |

4.3. Implementation

Now let us solve (2.11) by the DGM algorithm under conditions from the above setting. For the numerical test, we set the interest rate $r = 1\%$, the maturity time $T = 1$ and the power utility preference parameters $p = 0.0005$ and $p = 0.5$. We sampled 1000 time-space points $(t, y_1, y_2)$ in the interior of the domain $[0, T] \times [-10, 10] \times [0, 10]$ and 100 space points at terminal time $T$. We set 100 steps to resample new time-space domain points. Before resampling, each stochastic gradient descent step is repeated 10 times. We set 50 hidden neurons in a hidden layer. From starting 0.001, learning rate decreased with decay rate 0.96 as the step goes by.

After solving (2.11) by the DGM algorithm, investors can choose their states $(y_1, y_2) \in [-10, 10] \times [0, 10]$ for fixed $t \in [0, T]$. The optimal portfolio can be constructed using (2.12) as:

$$ \pi_{DGM}(t, y_1, y_2) = \frac{1}{1 - p} \left( \Sigma^{-1} \mu + \Sigma^{-1} \nabla_y \Phi \left( u_{DGM} \right) \frac{1}{u_{DGM}} \right). $$

To sum up, one can get the value of $u$ and the portfolio value $\pi$ at every time or state. The investor could buy or sell a risky asset $S^{(1)}$ based on the value of the portfolio to maximize utility from terminal wealth.

Figure 1 shows surfaces of the solution $u_{DGM}$ of (2.11) using DGM algorithm in different times, with the power utility preference parameter $p = 0.0005$. We chose some part of domain $[0, 1] \times [0, 1]$ as a plot range for convenience. Figure 2 shows surfaces of the solution of (2.11) in $p = 0.5$, with the restricted plot range $[0, 5] \times [0, 5]$. In both figures, for different values of utility parameter $p$, we can easily notice the fact that the surface tends to the plane $u = 1$ as time goes to the terminal time $T$: the terminal condition of (2.11). Note however
Figure 1 is more regular than Figure 2 in the sense that the value of $L^2$-loss in $p = 0.0005$ was remarkably smaller than that in $p = 0.5$. Hence we may infer the value of market preference parameter $p$ has played a significant role for using the deep Galerkin method algorithm.

Figure 1. Surface of solution by the deep Galerkin method. ($p = 0.0005$)
Figure 2. Surface of solution by the deep Galerkin method. 
\((p = 0.5)\)

4.4. Comparing with the finite difference method

Now we solve (2.11) using the finite difference method (FDM). The domain has equally divided 40 grids satisfying:

\[
0 = t^0 < t^1 < \cdots < t^{40} = T, \\
-10 = y^0_1 < y^1_1 < \cdots < y^{40}_1 = 10, \\
0 = y^0_2 < y^1_2 < \cdots < y^{40}_2 = 10.
\]

First of all, we discretize the solution \(u\) as

\[
u^n_{i,j} := u(t^n, y^i_1, y^j_2), \quad 0 \leq i, j, n \leq 40.
\]
With this notation, we can substitute the equation (2.11) using the following central difference formula:

\[
\begin{align*}
    u_t &= \frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t}, \\
    u_{y_1} &= \frac{u_{i+1,j}^n - u_{i-1,j}^n}{2(\Delta y_1)}, \\
    u_{y_2} &= \frac{u_{i,j+1}^n - u_{i,j-1}^n}{2(\Delta y_2)}.
\end{align*}
\]

Note that we used the forward difference for discretizing \(u_t\) in order to get the values of \((u_{i,j}^n)_{1 \leq i,j \leq 40}\) by using the values of \((u_{i,j}^{n+1})_{1 \leq i,j \leq 40}\) for \(n = 0, \ldots, 39\).

Also the central difference approximations of the second derivative of \(u\) are given by:

\[
\begin{align*}
    u_{y_1 y_1} &= \frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{(\Delta y_1)^2}, \\
    u_{y_2 y_2} &= \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{(\Delta y_2)^2}, \\
    u_{y_1 y_2} &= \frac{u_{i+1,j+1}^n - u_{i+1,j-1}^n - u_{i-1,j+1}^n + u_{i-1,j-1}^n}{4(\Delta y_1)(\Delta y_2)}.
\end{align*}
\]

Then the PDE (2.11) becomes a nonlinear equation with 1521 (=39×39) unknowns \((u_{i,j}^n)_{1 \leq i,j \leq 39}\) for each \(n = 0, \ldots, 39\). The equation is of the form:

\[
\begin{align*}
    &u_{i,j}^{n+1} - u_{i,j}^n + C_1 \frac{u_{i+1,j}^n - u_{i-1,j}^n}{2(\Delta y_1)} + C_2 \frac{u_{i,j+1}^n - u_{i,j-1}^n}{2(\Delta y_2)} \\
    &+ C_3 \frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{(\Delta y_1)^2} + C_4 \frac{u_{i+1,j+1}^n - u_{i-1,j+1}^n - u_{i-1,j-1}^n + u_{i+1,j-1}^n}{4(\Delta y_1)(\Delta y_2)} \\
    &+ C_5 \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{(\Delta y_2)^2} + C_6 u_{i,j}^n - q \frac{u_{i,j}^n}{2u_{i,j}^n} \left[ C_7 \frac{(u_{i+1,j}^n - u_{i-1,j}^n)}{(2\Delta y_1)} \right]^2 \\
    &- \frac{q}{2u_{i,j}^n} \left[ C_8 \frac{u_{i+1,j}^n - u_{i-1,j}^n - u_{i-1,j+1}^n - u_{i+1,j-1}^n}{2(\Delta y_1)(\Delta y_2)} + C_9 \frac{(u_{i,j+1}^n - u_{i,j-1}^n)^2}{2(\Delta y_2)} \right] = 0,
\end{align*}
\]

where \(C_1, \ldots, C_9\) are constants. Note that the terminal condition \(u(T,y) = u(T, y_1, y_2) = 1\) also becomes

\[
u_{i,j}^0 = 1 \quad \text{for all} \quad 0 \leq i, j \leq 40.
\]

Since (2.11) has no boundary condition, we used the boundary data from the DGM algorithm. Figure 3 shows surfaces of the solution using the finite difference method in different times with \(p = 0.0005\). We used the Newton's method since the equation is nonlinear. For more detail, see [33].

With the same value of \(p\), the absolute errors between the solution from the deep Galerkin method and the one from the finite difference method are displayed in Figure 5. Notice that the error between these algorithms is getting slightly larger as the time \(t\) goes to zero. This may be due to the time-reversely performed finite difference method algorithm, from \(t = T\) to \(t = 0\). In other words, the stability on the solution from the terminal condition was gradually weakened as the time goes to zero.

In a different point of view, combining Figure 5 with Figure 1 and Figure 3, we conclude the solution is well-estimated by the deep neural network. It usually takes about 5 minutes to train the network. On the other hand, it only
takes less than 30 seconds to find the surface of solution by the FDM. One can deduce this traditional algorithm would be more efficient for time-saving. However, it is not always true. Figure 4 shows surfaces derived from the finite difference method algorithm with $p = 0.5$, same domain with Figure 2. In Figure 4, the solution has extremely large or small values. This singularity may have arisen since the system of equations is nonlinear. In other words, the matter of finding inverse matrix in the Newton’s method at each step $n = 39, \ldots, 0$ would make the value of solutions undesirable.

Figure 3. Surface of solution using the finite difference method
5. Conclusion

In this paper we first modeled the market with a safe asset and some risky assets whose dynamics satisfy the diffusion process with returns. We then induced the HJB equation to maximize the expectation of an investor’s utility, given by investment opportunities modeled by a \(d\)-dimensional state process. Using some properties including homotheticity and concaveness, we finally derived a nonlinear partial differential equation and approximated the solution with a deep learning algorithm.

For comparison with the deep Galerkin method, we applied the finite difference method to find an approximated solution. In case of the utility parameter being quite small, \(p = 0.0005\), we found that the solution has estimated well by...
the neural network. However in the case of $p = 0.5$, there were several singular points in solution surfaces approximated by the finite difference method. Hence unlike the deep Galerkin method, this mesh-based algorithm showed some defects such as a singularity by a nonlinearity of discretized version of partial differential equations. This concludes that the DGM algorithm is relatively stable and has less difficulties to approximate the solution for PDEs.

Furthermore, all above procedures in Section 4 were performed only with the 2-dimensional state process. If the dimension $d$ of state process increases, since there would exist millions of grids, it would be more computationally efficient to apply the DGM algorithm than the FDM algorithm. Finally with the approximated solution from the relatively stable DGM algorithm, the investor
A deep learning algorithm for optimal investment strategies can decide how to allocate one’s wealth in several risky assets by the optimal portfolio formula.

Also there has some further studies to be researched. The stability or regularity of the solution is to be researched as the following are changed: model or dimension of a state variable $Y$, value of calibrated parameters, market preference parameter $p$ and sampling domain. Also in the optimal portfolio formula, the stability on a gradient term needs to be considered. Meanwhile, J. Sirignano et al. proved the convergence of the DGM algorithm only in a class of quasilinear parabolic PDEs. Although they referred that the algorithm can be applied to other types of PDEs, there needs to be some researches for the stability of hyperbolic, elliptic or fully nonlinear PDEs.

6. Proof of Theorem 3.1

Here we now justify Theorem 3.1 by proving the following two theorems in special cases. The main steps of proofs are from [20] and [35] based on universal approximation arguments. Note that the problem formulation in this paper is different from the above papers. Thus, for completeness of the paper, we provide the rigorous proof as follows. The first theorem shows the convergence of $J(f)$: there exists a deep neural network $f$ such that the loss functional $J(f)$ tends to the arbitrary small. The latter one stands for the convergence of the DNN function to the solution of PDEs.

6.1. Convergence of the loss functional

Assume $D \subset \mathbb{R}^d$ is bounded with a smooth boundary $\partial D$. Denote $D_T = [0, T) \times D$. Consider the following form of quasilinear parabolic PDE:

\[ \partial_t u(t, y) - \text{div}(\alpha(t, y, u, \nabla u)) + \gamma(t, y, u, \nabla u) = 0, \quad (t, y) \in D_T, \]

\[ u(T, y) = u_T(y), \quad y \in D, \]

with

\[ \mathcal{G}[u](t, y) := \partial_t u(t, y) - \text{div}(\alpha(t, y, u, \nabla u)) + \gamma(t, y, u, \nabla u). \]

Then the above differential operator $\mathcal{G}$ can be expressed as

\[ \mathcal{G}[u](t, y) = \partial_t u(t, y) - \sum_{i,j=1}^{d} \frac{\partial \alpha_{ij}}{\partial u_j} \frac{\partial u_i}{\partial y_j} - \sum_{i=1}^{d} \frac{\partial \alpha_i}{\partial y_i} u - \sum_{i=1}^{d} \frac{\partial \alpha_i}{\partial y_i} + \gamma(t, y, u, \nabla u) \]

\[ =: \partial_t u(t, y) - \sum_{i,j=1}^{d} \frac{\partial \alpha_{ij}}{\partial u_j} \frac{\partial u_i}{\partial y_j} + \hat{\gamma}(t, y, u, \nabla u). \]

**Theorem 6.1.** Let $\mathcal{C}^n(\psi)$ be a collection of DNN functions with $n$ hidden neurons in a single hidden layer:

\[ \mathcal{C}^n(\psi) = \left\{ \zeta : \mathbb{R}^{1+d} \to \mathbb{R} : \zeta(t, y) = \sum_{i=1}^{n} \beta_i \psi \left( \sum_{j=1}^{d} \alpha_{ji} y_j + c_i \right) \right\}, \]
where $\psi$ is an activation function and $\theta = (\beta_1, \ldots, \beta_n, \alpha_{11}, \ldots, \alpha_{dn}, c_1, \ldots, c_n)$ in $\mathbb{R}^{2n+n(1+d)}$ is a vector of the neural network parameters. Assume the following:

- $\psi$ is in $C^2(\mathbb{R}^d)$, bounded and non-constant.
- $[0, T] \times D$ is compact.
- $\text{supp} \nu_1 \subset D_T$ and $\text{supp} \nu_2 \subset D$.
- The above PDE (6.1) has a unique solution, where this solution belongs to both $C(D_T)$ and $C^{1+\frac{\eta}{2}, 2+\eta}(D_T)$ for $0 \leq \eta \leq 1$, and
  $$\sup_{D_T} (|\nabla y u(t, y)| + |\nabla^2_y u(t, y)|) < \infty.$$

- $\tilde{\gamma}(t, y, u, p)$ and $\partial u_j \frac{\partial \alpha_i(t, y, u, p)}{\partial p_j}$ for $1 \leq i, j \leq d$ are locally Lipschitz continuous, where Lipschitz constant has a polynomial growth in $u$ and $p$.
- $\frac{\partial \alpha_i(t, y, u, p)}{\partial u_j}$ is bounded for $1 \leq i, j \leq d$.

Then there is a constant

$$K = K \left( \sup_{D_T} |u|, \sup_{D_T} |\nabla y u|, \sup_{D_T} |\nabla^2_y u| \right) > 0$$

such that for arbitrary positive $\epsilon > 0$, there is a DNN function $f$ in $\mathcal{C}(\psi) = \bigcup_{n=1}^{\infty} \mathcal{C}^n(\psi)$ satisfying $J(f) \leq K\epsilon$.

Proof. By Theorem 3 in [20], for every $\epsilon > 0$ and $u \in C^{1,2}([0, T] \times \mathbb{R}^d)$, there is a DNN function $f = f(t, y; \theta)$ in $\mathcal{C}(\psi)$ such that

$$\sup_{D_T} |\partial_t u - \partial_t f| + \sup_{D_T, \partial t \leq 2} |\partial(t)^j u - \partial(t)^j f| < \epsilon.$$  

Also we may assume for $C > 0$, nonnegative constants $c_1, c_2, c_3$ and $c_4$,

$$\tilde{\gamma}(t, y, u, p) - \gamma(t, y, v, q) \leq C \left( |u|^2 + |v|^2 + |p|^2 + |q|^2 + 1 \right) (|u - v| + |p - q|),$$

by the local Lipschitz continuity of $\gamma(t, y, u, p)$ in $u$ and $p$. We abbreviate $u(t, y)$ and $f(t, y; \theta)$ for convenience. From the Hölder inequality with exponents $r_1$ and $r_2$,

$$\int_{D_T} |\tilde{\gamma}(t, y, f, \nabla y f) - \hat{\gamma}(t, y, u, \nabla y u)|^2 \, dv_1$$

$$\leq C \left( \int_{D_T} (|f|^{r_1} + |u|^{r_2} + |\nabla y f|^{r_3} + |\nabla y u|^{r_4} + 1)(|f - u|^2 + |f| - |\nabla y u|^2) \, dv_1 \right)^{\frac{1}{r_1}}$$

$$\times \left( \int_{D_T} (|f - u|^2 + |\nabla y f - \nabla y u|^2)^{r_2} \, dv_1 \right)^{\frac{1}{r_2}}.$$
\[
\leq C \left( \int_{D_T} (|f - u|^2 + |\nabla_y f - \nabla_y u|^2 + |u|^{c_1} + |\nabla_y u|^{c_2} + 1)^{r_1} \, dv_1 \right)^{\frac{1}{r_1}} \\
\times \left( \int_{D_T} (|f - u|^2 + |\nabla_y f - \nabla_y u|^2)^{r_2} \, dv_1 \right)^{\frac{1}{r_2}} \\
\leq C \left( \epsilon^{c_1} + \epsilon^{c_2} + \sup_{D_T} |u|^{c_1 \vee c_2} + \sup_{D_T} |\nabla_y u|^{c_2 \vee c_4} \right) \epsilon^2.
\]

Each constant \(C\) from the above inequalities may differ from each other. The last inequality holds because of (6.2).

Also we may assume

\[
\left| \frac{\partial \alpha_i(t, y, u, p)}{\partial p_j} - \frac{\partial \alpha_i(t, y, v, q)}{\partial q_j} \right| \leq C \left( |u|^{\frac{c_1}{2}} + |v|^{\frac{c_2}{2}} + |p|^{\frac{c_3}{2}} + |q|^{\frac{c_4}{2}} + 1 \right) \times (|u - v| + |p - q|),
\]

by the local Lipschitz continuity of \(\frac{\partial \alpha_i(t, y, u, p)}{\partial p_j}\) in \(u\) and \(p\). For convenience, we denote

\[
\xi(t, y, h, \nabla h, \nabla^2 h) = \sum_{i,j=1}^{d} \frac{\partial \alpha_i(t, y, h, \nabla h)}{\partial y_{ij}} \partial y_{ij} h(t, y).
\]

In spirit to the above procedure we used the Hölder inequality with exponents \(p\) and \(q\):

\[
\int_{D_T} \left| \xi(t, y, u, \nabla_y u, \nabla^2 u) - \xi(t, y, f, \nabla_y f, \nabla^2 f) \right|^2 \, dv_1 \\
= \int_{D_T} \left[ \sum_{i,j=1}^{d} \left( \frac{\partial \alpha_i(t, y, f, \nabla f)}{\partial f_{ij}} - \frac{\partial \alpha_i(t, y, u, \nabla u)}{\partial u_{ij}} \right) \partial y_{ij} u(t, y) \right]^2 \, dv_1 \\
+ \int_{D_T} \left[ \sum_{i,j=1}^{d} \frac{\partial \alpha_i(t, y, f, \nabla f)}{\partial f_{ij}} \left( \partial y_{ij} f(t, y; \theta) - \partial y_{ij} u(t, y) \right) \right]^2 \, dv_1 \\
\leq C \sum_{i,j=1}^{d} \left( \int_{D_T} |\partial y_{ij} u(t, y)|^{2p} \, dv_1 \right)^{\frac{1}{2}} \left( \int_{D_T} \left| \frac{\partial \alpha_i}{\partial f_{ij}} \right|^{2q} \, dv_1 \right)^{\frac{1}{2}} \\
+ C \sum_{i,j=1}^{d} \left( \int_{D_T} \left| \frac{\partial \alpha_i}{\partial y_{ij}} \right|^{2p} \, dv_1 \right)^{\frac{1}{2}} \left( \int_{D_T} \left| \partial y_{ij} f(t, y; \theta) - \partial y_{ij} u(t, y) \right|^{2q} \, dv_1 \right)^{\frac{1}{2}} \\
\leq C \sum_{i,j=1}^{d} \left( \int_{D_T} |\partial y_{ij} u(t, y)|^{2p} \, dv_1 \right)^{\frac{1}{2}} \\
\times \left( \int_{D_T} (|f - u|^2 + |\nabla_y f - \nabla_y u|^2)^{\frac{r_2}{2}} \, dv_1 \right)^{\frac{1}{r_2}}.
\]
\[
\times \left( \int_{D_T} (|f - u|^3 + |\nabla_y f - \nabla_y u|^3 + |u|^{\kappa_1 \kappa_2} + |\nabla_y u|^{\kappa_3 \kappa_4} + 1)^{\theta_1} \, dt \right)^{\frac{1}{\theta_1}} + C \sum_{i,j=1}^d \left( \int_{D_T} \left| \frac{\partial \alpha}{\partial f_{ij}} \right|^{2p} \, dt \right)^{\frac{1}{p}} \left( \int_{D_T} \left| \partial_{y_i y_j} f(t, y; \theta) - \partial_{y_i y_j} u(t, y) \right|^2 \, dt \right)^{\frac{1}{2}} \leq C \epsilon^2.
\]

We also abbreviate \(\partial_{\alpha_i (t, y, f, \nabla f)}\) and \(\partial_{\alpha_i (t, y, u, \nabla u)}\) as \(\partial_{\alpha_i}\) and \(\partial_{u_i}\), respectively, from the above inequalities.

To sum up, we finally obtain the following inequality:
\[
J(f) = \|G[f]\|_{D_T, v_1}^2 + \|f(T, y; \theta) - u_T(y)\|_{D, \nu_2}^2
= \|G[f] - G[y]\|_{D_T, v_1}^2 + \|f(T, y; \theta) - u_T(y)\|_{D, \nu_2}^2
\leq \int_{D_T} (|\partial_t u - \partial_1 f|^2 + |\xi(t, y, u, \nabla u, \nabla^2 u) - \xi(t, y, f, \nabla f, \nabla^2 f)|^2) \, dt
+ \int_{D_T} |\gamma(t, y, f, \nabla_y f) - \gamma(t, y, u, \nabla_y u)|^2 \, dt
+ \int_{D} |f(T, y; \theta) - u_T(y)|^2 \, dt
\leq K \epsilon^2
\]
for some constant \(K > 0\).

### 6.2. Convergence of the DNN function to the solution of PDEs

As we done in Section 6.1, consider the quasilinear parabolic PDE (6.1) and the following loss functional

\[
J(f) = \|G[f]\|_{D_T, v_1}^2 + \|f(T, y; \theta) - u_T(y)\|_{D, \nu_2}^2
\]

By Theorem 6.1, there is a neural network \(f^n\) such that \(J(f^n)\) tends to 0. Each \(f^n\) satisfies the following:

\[
G[f^n](t, y) = h^n(t, y), \quad (t, y) \in D_T,
\]

\[
f^n(T, y) = u^n_T(y), \quad y \in D,
\]
and

\[
\|h^n\|_{D_T, v_1}^2 + \|u^n_T - u_T\|_{D, \nu_2}^2 \to 0 \text{ as } n \to \infty.
\]

**Theorem 6.2.** Assume the following:

- \(\|\alpha(t, y, u, p)\| \leq \mu(\|p\| + \kappa(t, y))\) for all \((t, y) \in D_T\), with \(\mu > 0\) and \(\kappa \in L^2(D_T)\) being positive.
- \(\alpha\) is continuously differentiable in \((y, u, p)\).
- Both \(\alpha\) and \(\gamma\) are Lipschitz continuous, uniformly on the following form of compact sets:

\[
\{(t, y, u, p) : t \in [0, T], y \in \bar{D}, 0 \leq |u| \leq C, 0 \leq \|p\| \leq C\}.
\]
\[ \langle p, \alpha(t, y, u, p) \rangle \geq \nu \|p\|^2 \text{ for some } \nu > 0. \]
\[ \langle p_1 - p_2, \alpha(t, y, u, p_1) - \alpha(t, y, u, p_2) \rangle > 0 \text{ for some } \nu > 0, \text{ for every } p_1, p_2 \in \mathbb{R}^d \text{ with } p_1 \neq p_2. \]
\[ |\gamma(t, y, u, p)| \leq \|p\|\lambda(t, y) \text{ for all } (t, y) \in D_T, \text{ with } \lambda \in L^{d+2}(D_T) \text{ being positive}. \]

\[ u_T(y) \in C^{0,2+\xi}_{\partial \Omega}(\bar{D}) \text{ for some } \xi > 0. \]

\[ \|u(y)\|_{C^{0,\alpha}(\bar{D})} = \sup_{y \in D} |u(y)|^{\alpha} + \sup_{y_1, y_2 \in D, y_1 \neq y_2} \frac{|u(y_1) - u(y_2)|}{|y_1 - y_2|^{\beta-\alpha}}. \]

Then

1. The PDE (6.1) has a unique bounded solution
\[ u \in C^{0,\delta}(\bar{D}_T) \cap W^{1,2}_0(D_T) \cap L^2(0, T; W^{1,2}_0(D)), \quad \delta > 0, \]
for any interior subdomain \( D_T \subset D_T \).

2. \( f^n \rightarrow u \) strongly in \( L^p(D_T) \) for every \( p < 2 \).

Note that in case of the class of quasilinear parabolic PDEs with boundary conditions, we should also consider the limiting process in the weak formulation of PDEs and use the Vitali’s theorem. For more detail, see Appendix A in [35]. See also [7], [12], [13] and [27].

Proof of Theorem 6.2. Existence, regularity and uniqueness for (6.1) follows from Theorem 2.1 in [32] and Theorem 6.3 to 6.5 of Chapter V.6 in [22]. Boundness holds by Theorem 2.1 in [32]. See also Chapter V 2 from [22].

Let \( f^n \) be the solution of (6.3). By Lemma 4.1 of [32], \( \{f^n\}_{n \in \mathbb{N}} \) is uniformly bounded in both \( L^\infty(0, T; L^2(D)) \) and \( L^2(0, T; W^{1,2}_0(D)) \). Then we can pick a subsequence from the sequence of neural networks \( \{f^n\}_{n \in \mathbb{N}} \), where we denote also by \( \{f^n\}_{n \in \mathbb{N}} \) for convenience, satisfying

- \( f^n \xrightarrow{w} u \) in \( L^\infty(0, T; L^2(D)) \),
- \( f^n \rightharpoonup u \) weakly in \( L^2(0, T; W^{1,2}_0(D)) \),
- \( f^n(\cdot, t) \rightharpoonup v(\cdot, t) \), weakly in \( L^2(D) \), for every fixed \( t \in [0, T) \),

for some functions \( u, v \). Since the norm of \( f \) in a space \( L^2(0, T; W^{1,2}_0(D)) \) is defined as
\[ \|f\|_{L^2(0, T; W^{1,2}_0(D))} = \left( \int_0^T \|f\|_{W^{1,2}_0(D)}^2 \, dt \right)^{\frac{1}{2}}, \]
where
\[ \|f\|_{W^{1,2}_0(D)}^2 = \sum_{|\alpha| \leq 2} \|D^\alpha f\|_{L^2(D)}^2 = \|f\|^2_{L^2(D)} + \|Df\|^2_{L^2(D)} + \|D^2f\|^2_{L^2(D)}; \]
\{\nabla_y f^n\}_{n \in \mathbb{N}} \text{ is uniformly bounded in } L^2(0,T;W^{1,2}_0(D)).

Let \( q = 1 + \frac{d}{2q+q} \in (1,2) \). By the Hölder inequality with exponents \( r_1 > 1 \) and \( r_2 > 1 \),

\[
\int_{D_T} |\gamma(t,y,f^n,\nabla_y f^n)|^q \, dt \, dy \leq \left( \int_{D_T} |\lambda(t,y)|^{r_1q} \, dt \, dy \right)^{\frac{1}{r_1}} \times \left( \int_{D_T} |\nabla_y f^n(t,y)|^{r_2q} \, dt \, dy \right)^{\frac{1}{r_2}}.
\]

Choose \( r_2 = \frac{2}{q} \). Then we get \( r_1 = \frac{2}{2-q} \) and hence \( r_1q = d + 2 \). Since \( \lambda \in L^{d+2}(D_T) \) and \( \{\nabla_y f^n\}_{n \in \mathbb{N}} \) is uniformly bounded,

\[
\int_{D_T} |\gamma(t,y,f^n,\nabla_y f^n)|^q \, dt \, dx \leq C
\]

for some \( C > 0 \).

The growth assumption on \( \alpha \) and the above argument imply that \( \{\partial_t f^n\}_{n \in \mathbb{N}} \) is uniformly bounded in \( L^1(0,T;W^{-1,2}(D)) \). Let \( \delta_1, \delta_2 \) be the conjugate exponents satisfying \( \delta_2 > \max \{2, d\} \). By the Gagliardo–Nirenberg–Sobolev inequality and the Rellich–Kondrachov compactness theorem (for further details, see Chapter 5 in [15]), the following embeddings hold:

\[
W^{-1,2}(D) \subset W^{-1,\delta_1}(D), \quad L^2(D) \subset W^{-1,\delta_1}(D) \quad \text{and} \quad L^2(D) \subset W^{-1,\delta_1}(D),
\]

and hence \( \{\partial_t f^n\}_{n \in \mathbb{N}} \) is uniformly bounded in \( L^1(0,T;W^{-1,\delta_1}(D)) \).

By Corollary 4 in [34] and the following embedding

\[
W^{1,2}_0(D) \subset L^2(D) \subset W^{-1,\delta_1}(D),
\]

\( \{f^n\}_{n \in \mathbb{N}} \) is relatively compact in \( L^2(D_T) \), in other words,

\[
f^n \rightarrow u \text{ strongly in } L^2(D_T) \text{ as } n \rightarrow \infty.
\]

Thus

(6.4) \( f^n \rightarrow u \) almost everywhere in \( D_T \) up to subsequences.

Note that from Theorem 3.3 of [6], we get

(6.5) \( \nabla f^n \rightarrow \nabla u \) almost everywhere in \( D_T \).

Hence \( f^n \rightarrow u \) strongly in \( L^\rho(0,T;W^{1,\rho}_0(D)) \) and so in \( L^\rho(D_T) \) for every \( \rho < 2 \), by (6.4) and (6.5). \( \Box \)

**Acknowledgement.** Hyungbin Park was supported by Research Resettlement Fund for the new faculty of Seoul National University, South Korea. In addition, Hyungbin Park was supported by the National Research Foundation of Korea (NRF) grants funded by the Ministry of Science and ICT (No.
A DEEP LEARNING ALGORITHM FOR OPTIMAL INVESTMENT STRATEGIES

2017R1A5A1015626, No. 2018R1C1B5085491 and No. 2021R1C1C1011675) and the Ministry of Education (No. 2019R1A6A1A10073437) through the Basic Science Research Program. Financial support from the Institute for Research in Finance and Economics of Seoul National University is gratefully acknowledged.

References

[1] Y. Achdou and O. Pironneau, *Computational methods for option pricing*, Frontiers in Applied Mathematics, 30, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2005. https://doi.org/10.1137/1.9780898717495

[2] A. Al-Alradi, A. Correia, G. Jardim, D. Naïff, and Y. Saporito, *Solving nonlinear and high-dimensional partial differential equations via deep learning*, arXiv preprint arXiv:1811.08782.

[3] A. Al-Alradi, A. Correia, G. Jardim, D. Naïff, and Y. Saporito, *Applications of the deep Galerkin method to solving partial integro-differential and Hamilton-Jacobi-Bellman equations*, arXiv preprint arXiv:1912.01455.

[4] F. E. Benth, K. H. Karlsen, and K. Reikvam, *Merton’s portfolio optimization problem in a Black and Scholes market with non-Gaussian stochastic volatility of Ornstein-Uhlenbeck type*, Math. Finance 13 (2003), no. 2, 215–244. https://doi.org/10.1111/1467-9965.00015

[5] T. Björk, *Arbitrage Theory in Continuous Time*, Oxford University Press, Oxford, 2009.

[6] L. Boccardo, A. Dall’Aglio, T. Gallouët, and L. Orsina, *Nonlinear parabolic equations with measure data*, J. Funct. Anal. 147 (1997), no. 1, 237–258. https://doi.org/10.1006/jfan.1996.3040

[7] L. Boccardo, M. M. Porzio, and A. Primo, *Summability and existence results for nonlinear parabolic equations*, Nonlinear Anal. 71 (2009), no. 3-4, 978–990. https://doi.org/10.1016/j.na.2008.11.066

[8] R. L. Burden, J. D. Faires, and A. C. Reynolds, *Numerical Analysis*, Prindle, Weber & Schmidt, Boston, MA, 1978.

[9] G. Callegaro, M. Gągi, S. Scotti, and C. Sgarra, *Optimal investment in markets with over and under-reaction to information*, Math. Financ. Econ. 11 (2017), no. 3, 299–322. https://doi.org/10.1007/s11579-016-0182-8

[10] R. Crisóstomo, *An analysis of the Heston stochastic volatility model: Implementation and calibration using Matlab*, CNMV Working Paper (2014), no. 58.

[11] A. Danilova, M. Monoyios, and A. Ng, *Optimal investment with inside information and parameter uncertainty*, Math. Financ. Econ. 3 (2010), no. 1, 13–38. https://doi.org/10.1007/s11579-010-0025-y

[12] L. Debmath, *Nonlinear Partial Differential Equations for Scientists and Engineers*, third edition, Birkhäuser/Springer, New York, 2012. https://doi.org/10.1007/978-0-8176-8265-1

[13] R. Di Nardo, F. Feo, and O. Guibé, *Existence result for nonlinear parabolic equations with lower order terms*, Anal. Appl. (Singap.) 9 (2011), no. 2, 161–186. https://doi.org/10.1142/S021953051001790

[14] W. E, M. Hutzenthaler, A. Jentzen, and T. Kruse, *On multilevel Picard numerical approximations for high-dimensional nonlinear parabolic partial differential equations and high-dimensional nonlinear backward stochastic differential equations*, J. Sci. Comput. 79 (2019), no. 3, 1534–1571. https://doi.org/10.1007/s10915-018-09093-0

[15] L. C. Evans, *Partial Differential Equations*, Amer. Math. Soc., 2002.

[16] S. Fallah and F. Mehdoust, *On the calibration of fractional two-factor stochastic volatility model with non-Lipschitz diffusions*, Commun. Stat. Simul. Comput. (2020), 1–20.
[17] P. Guasoni and S. Robertson, Static fund separation of long-term investments, Math. Finance 25 (2015), no. 4, 789–826. https://doi.org/10.1111/mafi.12017

[18] J. Han, A. Jentzen, and W. E, Solving high-dimensional partial differential equations using deep learning, Proc. Natl. Acad. Sci. USA 115 (2018), no. 34, 8505–8510. https://doi.org/10.1073/pnas.1718942115

[19] S. L. Hansen, Optimal consumption and investment strategies with partial and private information in a multi-asset setting, Math. Financ. Econ. 7 (2013), no. 3, 305–340. https://doi.org/10.1007/s11579-012-0086-1

[20] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (1991), no. 2, 251–257.

[21] C. Kühn and M. Stroh, Optimal portfolios of a small investor in a limit order market: a shadow price approach, Math. Financ. Econ. 3 (2010), no. 2, 45–72. https://doi.org/10.1007/s11579-010-0027-9

[22] O. A. Ladyzhenskaia, V. A. Solonnikov and N. N. Ural’tseva, Linear and quasi-linear equations of parabolic type, Amer. Math. Soc., 1988.

[23] I. E. Lagaris, A. C. Likas and D. G. Papageorgiou, Neural-network methods for boundary value problems with irregular boundaries, IEEE Trans. Neural. Netw. 11 (2000), no. 5, 1041–1049.

[24] H. Lee and I. S. Kang, Neural algorithms for solving differential equations, J. Comput. Phys. 91 (1990), no. 1, 110–131. https://doi.org/10.1016/0021-9991(90)90007-N

[25] V. Lemaire and T. Montes, Stationary Heston model: Calibration and pricing of exotics using product recursive quantization, arXiv preprint arXiv:2001.03101.

[26] Z. Liang and M. Ma, Robust consumption-investment problem under CRRA and CARA utilities with time-varying confidence sets, Math. Finance 30 (2020), no. 3, 1055–1072. https://doi.org/10.1111/mafi.12217

[27] M. Magliocca, Existence results for a Cauchy-Dirichlet parabolic problem with a repulsive gradient term, Nonlinear Anal. 166 (2018), 102–143. https://doi.org/10.1016/j.na.2017.09.012

[28] A. Malek and R. Shekari Beidokhti, Numerical solution for high order differential equations using a hybrid neural network—optimization method, Appl. Math. Comput. 183 (2006), no. 1, 260–271. https://doi.org/10.1016/j.amc.2006.05.068

[29] R. C. Merton, Lifetime portfolio selection under uncertainty: The continuous-time case, Rev. Econ. Stat. (1969), 247–257.

[30] M. Nutz, The opportunity process for optimal consumption and investment with power utility, Math. Financ. Econ. 3 (2010), no. 3-4, 139–159. https://doi.org/10.1007/s11579-010-0031-0

[31] J. L. Pedersen and G. Peskir, Optimal mean-variance portfolio selection, Math. Financ. Econ. 11 (2017), no. 2, 137–160. https://doi.org/10.1007/s11579-016-0174-8

[32] M. M. Porzio, Existence of solutions for some “noncoercive” parabolic equations, Discrete Contin. Dynam. Systems 5 (1999), no. 3, 553–568. https://doi.org/10.3934/dcds.1999.5.553

[33] C. Remani, Numerical Methods for Solving Systems of Nonlinear Equations, Lakehead University Thunder Bay, Ontario, Canada, 2013.

[34] J. Simon, Compact sets in the space $L^p(0,T;B)$, Ann. di Mat. Pura ed Appl. 146 (1986), no. 1, 65–96.

[35] J. Sirignano and K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018), 1339–1364. https://doi.org/10.1016/j.jcp.2018.08.029
A DEEP LEARNING ALGORITHM FOR OPTIMAL INVESTMENT STRATEGIES

Daeyung Gim
Division of Derivative Pricing
Korea Asset Pricing
Seoul 03131, Korea
Email address: dygim@koreaap.com, gbe375@snu.ac.kr

Hyungbin Park
Department of Mathematical Sciences and Research Institute of Mathematics
Seoul National University
Seoul 08826, Korea
Email address: hyungbin@snu.ac.kr, hyungbin2015@gmail.com