Chin. Ann. Math. Ser. B
42(2), 2021, 199–216
DOI: 10.1007/s11401-021-0253-x

Chinese Annals of Mathematics, Series B

Gradient Convergence of Deep Learning-Based Numerical Methods for BSDEs*

Zixuan WANG[1] Shanjian TANG[2]

Abstract The authors prove the gradient convergence of the deep learning-based numerical method for high dimensional parabolic partial differential equations and backward stochastic differential equations, which is based on time discretization of stochastic differential equations (SDEs for short) and the stochastic approximation method for nonconvex stochastic programming problem. They take the stochastic gradient decent method, quadratic loss function, and sigmoid activation function in the setting of the neural network. Combining classical techniques of randomized stochastic gradients, Euler scheme for SDEs, and convergence of neural networks, they obtain the $O(K^{-\frac{1}{4}})$ rate of gradient convergence with K being the total number of iterative steps.

Keywords PDEs, BSDEs, Deep learning, Nonconvex stochastic programming, Convergence result
2020 MR Subject Classification 62L20,90C26

1 Introduction

Deep learning has recently sparked academic interest due to its great success in many application fields like image identification, voice recognition, natural language processing, and instant machine translation. The success of deep learning also leads to the deep learning-based algorithm in ordinary differential equations (ODEs for short), partial differential equations (PDEs for short) and stochastic control problems. Malek and Beidokhti [18] reported a novel hybrid method based on optimization techniques and neural networks for the solution of high ODEs. Sirignano et al. [24] proposed a deep Galerkin method, since it is similar in spirit to Galerkin methods, with the solution approximated by a neural network instead of a linear combination of basis functions. Beck et al. [1] delivered a numerical approximation of the Kolmogorov PDE on an entire region without suffering from the curse of dimensionality by means of deep learning. Rudd [21] presented a method for solving PDEs using neural networks, which uses a constrained-backpropagation approach for preserving prior knowledge during incremental training for solving nonlinear elliptic and parabolic PDEs adaptively in non-stationary environments. Han and E [13] developed a deep learning approach that directly solves high-dimensional stochastic control problems based on Monte-Carlo sampling, and the objective function for the control problem plays the role of the loss function for the deep neural network. E, Han and

Manuscript received January 7, 2020. Revised May 14, 2020.
[1]Corresponding author. Department of Finance and Control Sciences, Shanghai Center for Mathematical Science, Fudan University, Shanghai 200433, China. E-mail: 15110840014@fudan.edu.cn
[2]Department of Finance and Control Sciences, School of Mathematical Sciences, Fudan University, Shanghai 200433, China. E-mail: sjtang@fudan.edu.cn
*This work was supported by the National Key R&D Program of China (No. 2018YFA0703900) and the National Natural Science Foundation of China (No. 11631004).

Jentzen [8] presented a deep learning-based numerical method for solving parabolic PDEs and backward stochastic differential equations (BSDEs for short) in high dimension and demonstrated its success, and we shall prove its gradient convergence result here.

The reason why deep networks work well in the above fields has been remaining to be a mystery, as it generally underlies a highly nonconvex optimization problem. Recently, there is a growing interest in the mathematical properties of these algorithms. E [10] used continuous dynamical systems to model nonlinear functions for machine learning in high-dimensional case. Furthermore, the continuous dynamical system approach to deep learning was explored by Li et al. [16], who proposed a framework for training algorithms. The convergence on the deep networks can be traced back to the study of nonconvex stochastic programming by Ghadimi and Lan [11], who focused on the theoretical development of stochastic approximation type methods. The methods can solve nonconvex stochastic programming problems which can be used to establish a theoretical framework on neural networks. Ithapu et al. [15] analyzed mini-batch stochastic gradients on multi-layer deep networks, and proved a gradient convergence on neural networks. The convergence of a new back-propagation algorithm with adaptive momentum (instead of stochastic gradient decent) was also studied (see [23, 28]). There are also many other ways in studying this problem. Carreira and Wang [4] proposed the method of auxiliary coordinates, which replaces original deeply nested problem with a constrained problem involving a different function in an augmented space without nesting, then the constrained problem can be solved with penalty-based methods using alternating optimization over the parameters and the auxiliary coordinates. E et al. [9] gave "a posteriori" error estimates for two-layer neural networks. The convergence of block coordinate descent type algorithms to a critical point of objective functions under natural conditions of neural network was considered (see [26, 27]). Zou et al. [29] studied the binary classification problem and showed that with a proper random weight initialization, the stochastic gradient descent method can find the global minima of the training loss for an over-parameterized deep ReLU (Rectified Linear Unit) network, under mild assumption on the training data.

The study of numerical methods for forward backward stochastic differential equations (FBSDEs for short) can be dated back to Douglas et al. [7] and Ma et al. [17], who proposed the four steps scheme to get the relation between the FBSDEs and their corresponding parabolic PDEs. Their work is based on the nonlinear Feynman-kac formulation (see [20, 25]). Bouchard and Touzi [3] as well as Delarue and Menozzi [6] studied a time-space discretization scheme for FBSDEs and provided an efficient probabilistic representation of this type of equations. Bender and Zhang [2] proved the convergence through a time discretization and a Markovian iteration. Cvitanic and Zhang [5] transformed the FBSDE to a control problem and proposed the steepest descent method to solve the latter one. The Fourier method to solve quite general FBSDEs with second-order accuracy was also developed (see [14, 22]).

There are numerous studies (see [7, 18, 27, 30–32]) on the convergence of neural networks, but few of them deal with the convergence of neural networks compound with stochastic system especially BSDEs. To the best of our knowledge, [12] is the only paper dealing with the convergence of deep BSDE method, but the work is quite different from us. They proved that as long as the objective function is optimized to be close to zero under fine time discretization, the approximate solution is close to the true solution. We focus on the convergence through the deep learning update steps. In other words, they proved that the deep BSDE method has abilities to get the true solution, and we obtain that the deep BSDERSG algorithm (we propose in Section 4) can get gradient convergence in the actual update method. On the other side, the mainstream research direction of neural networks' convergence remains on the gradient convergence, only several papers for example Zou et al. [29] study the global convergence

of neural networks. But their result depends on large number of neurons, which is far from using in practice. The main contribution of this work is as follows: We give the gradient convergence of the model raised by E, Han and Jentzen [8] of deep learning-based numerical methods for BSDEs, and we take the stochastic gradient decent method, quadratic loss function, and sigmoid activation function in our neural network settings. For sake of the nonconvexity of neural network, we get the gradient convergence, which can give some theoretical directions for this method, such as the choice of learning rate and how many iterative steps we need.

The rest of the paper is organized as the following five sections.

In Section 2 we give a brief introduction to the deep learning-based numerical method. In Section 3 we introduce the stochastic approximation type methods to solve the nonconvex stochastic programming problems with randomized stochastic gradient (RSG for short) method. In Section 4, we give the proof of the gradient convergence on the neural network first, and then prove the main result (Theorem 4.2) of this paper. In Section 5 we give the numerical experiment to explain our gradient convergence. Section 6 is the Appendix.

2 Deep Learning-Based Algorithm

This section focuses on giving the details of deep learning-based algorithm for a fairly general class of nonlinear parabolic PDEs, which was introduced by E et al. [8]. To get a better understanding of the proof, we state the deep learning-based algorithm in its general case.

The main steps of the algorithm are as follows: Through the nonlinear Feynman-Kac formula, we can formulate the PDEs associated to the FBSDEs. The FBSDE is viewed as a stochastic control problem with the gradient of the solution being the policy function (control). The policy function can then be approximated by a deep neural network.

We consider the setup of a system of parabolic PDEs with terminal conditions since this facilitates making connections with BSDEs. Terminal value problems can obviously be transformed into initial value problems.

2.1 The formulation of the problem

First, we introduce the formulation of the nonlinear parabolic PDEs. Let $T \in (0,\infty), d, r, q \in \mathbb{N}$, the functions $f\colon [0,T]\times\mathbb{R}^q\times\mathbb{R}^r\times\mathbb{R}^{r\times d}\to\mathbb{R}^r$ and $g\colon \mathbb{R}^q\to\mathbb{R}^r$ are continuous, and $(b,\sigma)\colon [0,T]\times\mathbb{R}^q\to\mathbb{R}^q\times\mathbb{R}^{q\times d}$ is continuously differentiable for all variables. Let $u\in C^{1,2}([0,T]\times\mathbb{R}^q,\mathbb{R}^r)$ satisfy that $u(T,x)=g(x)$ and

$$\begin{aligned}&\frac{\partial u^i}{\partial t}(t,x)+\frac{1}{2}\mathrm{Trace}((\Delta_x u^i)(t,x)\sigma(t,x)[\sigma(t,x)]^T)+\langle b(t,x),(\nabla_x u^i)(t,x)\rangle\\&+f^i(t,x,u(t,x),(\nabla_x u)(t,x)\sigma(t,x))=0\end{aligned}\tag{2.1}$$

for all $(t,x)\in([0,T]\times\mathbb{R}^q)$ and $i=1,2,\cdots,r$, where u^i and f^i are the i-th component of u and f respectively.

Then, through the nonlinear Feynman-Kac formula, we can formulate the PDE associated to the FBSDE. Let $(\Omega,\mathcal{F},\mathbb{P})$ be a probability space and W be a d-dimensional standard Brownian motion on $(\Omega,\mathcal{F},\mathbb{P})$ with $\mathbb{F}=(\mathbb{F}_t)_{t\in[0,T]}$ being the natural filtration on $(\Omega,\mathcal{F},\mathbb{P})$ generated by W. Let $Y\colon [0,T]\times\Omega\to\mathbb{R}^r$ and $Z\colon [0,T]\times\Omega\to\mathbb{R}^{r\times d}$ be $\mathbb{F}$-adapted continuous stochastic processes such that for all $t\in[0,T]$, it holds $\mathbb{P}$-a.s. that

$$\begin{cases}X_t=x_0+\displaystyle\int_0^t b(s,X_s)\mathrm{d}s+\int_0^t\sigma(s,X_s)\mathrm{d}W_s,\\ Y_t=g(X_T)+\displaystyle\int_t^T f(s,X_s,Y_s,Z_s)\mathrm{d}s-\int_t^T Z_s\mathrm{d}W_s.\end{cases}\tag{2.2}$$

Under suitable additional regularity assumptions on the nonlinear function f, the nonlinear PDE (2.1) is associated to the FBSDE (2.2) in the following sense: For all $t \in [0,T]$, it holds $\mathbb{P}$-a.s. that

$$Y_t = u(t, X_t), \quad Z_t = (\nabla_x u)\sigma(t,x). \tag{2.3}$$

We now solve the PDE via solution of a suitable stochastic control problem, and associate the solution $u \in C^{1,2}([0,T]\times\mathbb{R}^q, \mathbb{R}^r)$ of (2.1) and its gradient to the solution of a stochastic control problem associated with (2.2). More precisely, under suitable regularity on the nonlinear function f and $\xi \in \mathbb{R}^q$, the pair $(u(0,x_0), (\nabla_x u)\sigma(t,x_t)_{t\in[0,T]})$ is (up to indistinguishability) the unique global minimum of the function

$$\mathbb{R}\times\mathcal{A} \ni (y,Z) \mapsto \mathbb{E}[|Y_T^{y,Z} - g(X_T)|^2] \in [0,\infty]. \tag{2.4}$$

The usual method of discretization for decoupled FBSDEs is to discretize the SDE and BSDE, separately.

For the SDE part, let $N \in \mathbb{N}$ and let $t_0, t_1, \cdots, t_N \in [0,T]$ be real numbers which satisfy $0 = t_0 < t_1 < \cdots < t_N = T$ (we often take $t_n - t_{n-1} = \Delta t$ for $n = 1, 2, \cdots, N$),

$$X_{t_{n+1}} \approx X_{t_n} + b(t_n, X_{t_n})(t_{n+1} - t_n) + \langle \sigma(t_n, X_{t_n}), W_{t_{n+1}} - W_{t_n}\rangle. \tag{2.5}$$

For the BSDE part,

$$\begin{aligned} Y_{t_{n+1}} \approx\ & Y_{t_n} - f(t_n, X_{t_n}, Y_{t_n}, (\nabla_x u)\sigma(t_n, X_{t_n}))(t_{n+1} - t_n) \\ & + \langle (\nabla_x u)(t_n, X_{t_n})\sigma(t_n, X_{t_n}), W_{t_{n+1}} - W_{t_n}\rangle. \end{aligned} \tag{2.6}$$

2.2 Formulation of the algorithm

Let Υ: $[0,T]^2 \times \mathbb{R}^q \times \mathbb{R}^d \to \mathbb{R}^q$ be a function and $\xi \in \mathbb{R}^q$, let W^k, $k \in \mathbb{N}_0$, be independent d-dimensional standard Brownian motions on $(\Omega, \mathcal{F}, \mathbb{P})$. For every $\theta \in \mathbb{R}^\rho$, let $\mathcal{U}^\theta \in \mathbb{R}^r$, and $\mathcal{V}_n^\theta$: $\mathbb{R}^q \to \mathbb{R}^{r\times d}$, $n \in \{0, 1, \cdots, N-1\}$ be functions. For every $k \in \mathbb{N}_0$, let $\mathcal{X}^k$: $\{0, 1, \cdots, N\} \times \Omega \to \mathbb{R}^q$, and $\mathcal{Y}^{\theta,k}$: $\{0, 1, \cdots, N\} \times \Omega \to \mathbb{R}^k$ be stochastic processes which satisfy for all $\theta \in \mathbb{R}^\rho$, $n \in \{0, 1, \cdots, N-1\}$ that

$$\mathcal{X}_0^k = \xi, \quad \mathcal{Y}_0^{\theta,k} = \mathcal{U}^\theta, \quad \mathcal{X}_{n+1}^k = \Upsilon(t_n, t_{n+1}, \mathcal{X}_n^k, W_{t_{n+1}}^k - W_{t_n}^k) \tag{2.7}$$

and

$$\mathcal{Y}_{n+1}^{\theta,k} = \mathcal{Y}_n^{\theta,k} - f(t_n, \mathcal{X}_n^k, \mathcal{Y}_n^{\theta,k}, \mathcal{V}_n^\theta(\mathcal{X}_n^k))(t_{n+1} - t_n) + \mathcal{V}_n^\theta(\mathcal{X}_n^k)(W_{t_{n+1}}^k - W_{t_n}^k). \tag{2.8}$$

For every $k \in \mathbb{N}_0$, define the function (ϕ^k, Φ^k): $\mathbb{R}^\rho \times \Omega \to \mathbb{R} \times \mathbb{R}^\rho$ as

$$\phi^k(\theta,\omega) = \|\mathcal{Y}_N^{\theta,k}(\omega) - g(\mathcal{X}_N^k(\omega))\|, \tag{2.9}$$

$$\Phi^k(\theta,\omega) = (\nabla_\theta \phi^k)(\theta,\omega) \tag{2.10}$$

for all $(\theta,\omega) \in \mathbb{R}^\rho \times \Omega$.

Define Θ: $\mathbb{N}_0 \times \Omega \to \mathbb{R}^\rho$ be a stochastic process which satisfies for all $k \in \mathbb{N}$ that

$$\Theta_k = \Theta_{k-1} - \gamma_k \cdot \Phi^k(\Theta_{k-1}), \tag{2.11}$$

where Θ_0 is given randomly and γ_k is called the learning rate at the k-th step.

Figure 1 shows the main steps of the algorithm (to simplify Figure 1, we replace $(\nabla_x u)(t_n, X_{t_n})\sigma(t_n, X_{t_n})$ with $(\nabla_x u)(t_n, X_{t_n})$).

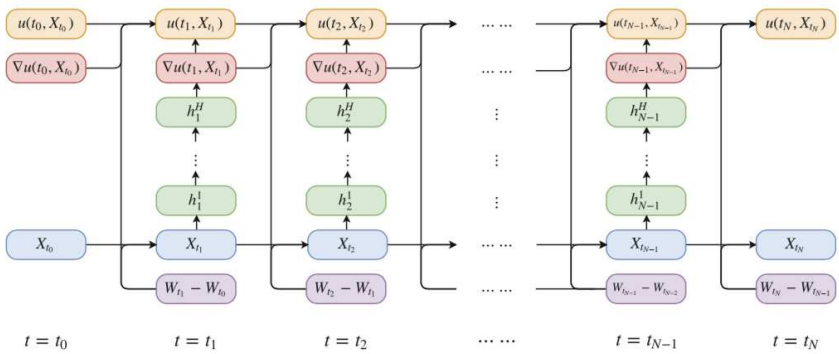

Figure 1 The sketch of deep BSDE algorithm (see [11, p.358]).

(1) $X_{t_n} \to h_n^1 \to h_n^2 \to \cdots \to h_n^H \to (\nabla_x u)(t_n, X_{t_n})$ is the multilayer feedforward neural network approximating the control at time t_n, where X_{t_n} is the input, the hidden layers are given by h_n^l for $l = 1, 2, \cdots, H$, and the output is $(\nabla_x u)(t_n, X_{t_n})$. The weights θ of this subnetwork are the parameters we aim to optimize.

(2) $(X_{t_n}, (\nabla_x u)(t_n, X_{t_n}), u(t_n, X_{t_n})) \to u(t_{n+1}, X_{t_{n+1}})$, where $u(t_{n+1}, X_{t_{n+1}})$ is the direct contribution to the final output of the network. Their functional form is determined by the BSDE (2.2). There are no parameters to be optimized in this type of connection.

(3) $(X_{t_n}, W_{t_n} - W_{t_{n-1}}) \to X_{t_{n+1}}$ is completely characterized by the SDE (2.2). There are also no parameters to be optimized in this type of connection.

3 The Nonconvex Stochastic Programming

As we know, the deep networks generally relate to a highly nonconvex optimization problem. Before handling the deep BSDE algorithm, we first consider the stochastic approximation type methods for solving an important class of nonconvex stochastic programming problems, which was introduced by Ghadimi and Lan [11]. More specifically, they studied the classical unconstrained nonlinear programming problem, which is given in the form of

$$f^* := \inf_{x \in \mathbb{R}^n} f(x), \tag{3.1}$$

where $f\colon \mathbb{R}^n \to \mathbb{R}$ is differentiable (usually nonconvex) and is bounded from below, and its gradient $\nabla f(\cdot)$ is uniformly Lipschitz continuous with constant $L \geq 0$:

$$\|\nabla f(y) - \nabla f(x)\| \leq L\|y - x\|, \quad \forall\, x, y \in \mathbb{R}^n. \tag{3.2}$$

In the standard nonlinear programming problem, the problem (3.1) can be solved by iterative algorithms which acquire the gradients of f. However, in our stochastic case, we have no full access to f, so we assume that we can only get noisy function values or gradients about the objective function f in (3.1). That is, at the k-th iteration of the algorithm, we can output a stochastic gradient $G(x_k, \xi_k)$ by making use of the known x_k, where ξ_k, $k \geq 1$, are random variables whose distributions P_k are supported on $\Xi_k \subset R^d$. The following assumptions are made for the Borel functions $G(x_k, \xi_k)$.

Assumptions A For any $k \geq 1$, we have

$$\text{(a) } \mathbb{E}[G(x_k, \xi_k)] = \nabla f(x_k), \tag{3.3}$$

$$\text{(b) } \mathbb{E}[\|G(x_k,\xi_k)-\nabla f(x_k)\|^2]\le\sigma^2 \tag{3.4}$$

for some parameter $\sigma\ge 0$.

Observe that $G(x_k,\xi_k)$ is an unbiased estimator of $\nabla f(x_k)$ and the variance of the random variable $G(x_k,\xi_k)-\nabla f(x_k)$ is bounded. In the standard setting for stochastic programming, we can always assume that the random vectors $\xi_k,\ k=1,2,\cdots$, are independent of each other (and also of x_k).

To solve the aforementioned nonconvex stochastic programming problem, Ghadimi and Lan gave the following RSG method in [11].

Algorithm 0: RSG Method

Input: Initial point x_1, iteration limit N, learning rates $\gamma_k, k\ge 1$, probability mass function $P_R(\cdot)$ supported on $\{1,\cdots,N\}$.

Step 0 Let R be a random variable with probability mass function P_R.

Step k. Get the noisy gradients function $G(x_k,\xi_k)$ and set

$$x_{k+1}=x_k-\gamma_k G(x_k,\xi_k),\ k=1,\cdots,R.$$

Output: x_R.

Theorem 3.1 *Suppose that the learning rates $\{\gamma_k\}$ and the probability mass function $P_R(\cdot)$ in the RSG method are chosen such that $\gamma_k\le\frac{2}{L}$ and*

$$P_R(k):=\mathrm{Prob}\{R=k\}=\frac{2\gamma_k-L\gamma_k^2}{\sum\limits_{k=1}^{N}(2\gamma_k-L\gamma_k^2)},\quad k=1,\cdots,N. \tag{3.5}$$

Then, under Assumption A *and the Lipschitz continuity of ∇f : For any $N\ge 1$, we have*

$$\frac{1}{L}\mathbb{E}[\|\nabla f(x_R)\|^2]\le\frac{D_f^2+\sigma^2\sum\limits_{k=1}^{N}\gamma_k^2}{\sum\limits_{k=1}^{N}(2\gamma_k-L\gamma_k^2)}, \tag{3.6}$$

where the expectation is taken with respect to R and $\xi_{[N]}:=(\xi_1,\cdots,\xi_N)$.

$$D_f:=\Big[\frac{2(f(x_1)-f^*)}{L}\Big]^{\frac{1}{2}}, \tag{3.7}$$

where f^* denotes the optimal value of problem (3.1).

The proof of Theorem 3.1 is given in the Appendix.

Remark 3.1 Suppose that the learning rates $\{\gamma_k\}$ are set to

$$\gamma_k=\min\Big\{\frac{1}{L},\frac{\widetilde{D}}{\sigma\sqrt{N}}\Big\},\quad k=1,\cdots,N \tag{3.8}$$

for some $\widetilde{D}>0$. Under the same assumption in Theorem 3.1, we have

$$\frac{1}{L}\mathbb{E}[\|\nabla f(x_R)\|^2]\le\frac{LD_f^2}{N}+\Big(\widetilde{D}+\frac{D_f^2}{\widetilde{D}}\Big)\frac{\sigma}{\sqrt{N}}. \tag{3.9}$$

So, this allows us to obtain the $O(\sqrt{N}^{-1})$ rate of convergence in terms of $\mathbb{E}[\|\nabla f(x_R)\|^2]$.

If, in addition, problem (3.1) is convex and x^* is an optimal solution, then, we further have the convergence of $\mathbb{E}[f(x_R)-f^*]$, and the details are referred to Ghadimi and Lan [11] and Nemirovski et al. [19]. Due to the nonconvexity of our neural network, the convergence of ∇f will be discussed in the next section.

4 Gradient Convergence on Deep Learning-Based Algorithm

We shall give the gradient convergence result for the deep learning-based algorithm in this section. First, note that BSDEs are solved from terminal time T to 0. However, in the underlying algorithm, the initial value y_0 is to be chosen as a parameter of the corresponding forward SDE, so that the terminal value will be satisfied. In this way, the BSDE is solved as an SDE, which is easy to be approximated and can also be updated by the back-propagate algorithm (the update algorithm in neural network). In order to make our proof more briefly, we take y_0 (or $u(t_0, X_{t_0})$) as an output of neural network. In summary, as one system, the inputs of the neural network are $X_{t_0}, X_{t_1}, \cdots, X_{t_N}$ (that is, $X_{t_0}, W_{t_1} - W_{t_0}, W_{t_2} - W_{t_1}, \cdots, W_{t_N} - W_{t_{N-1}}$ with functions b and σ), and the outputs are

$$u(t_0, X_{t_0}),\ (\nabla_x u)\sigma(t_0, X_{t_0}),\ (\nabla_x u)\sigma(t_1, X_{t_1}),\ \cdots,\ (\nabla_x u)\sigma(t_N, X_{t_N}).$$

Our main idea follows the nonconvex stochastic approximation methods which are introduced in Section 3. In the next subsection, we give the setting and proof only for a simple neural network.

4.1 Convergence on neural network

Denote by the pair $(x, y) \in \mathcal{X} \subset \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$ the input feature vector and the corresponding output (or label), respectively. The mapping from the input unit x to the output unit y after L hidden layers $h^1, \cdots, h^L$ is called L-layer neural network (L-NN for short). The dimensions of these $L + 2$ layers are $d_0 = d_x, d_1, d_2, \cdots, d_L, d_{L+1} = d_y$, respectively.

The transformation between two adjacent layers is the composition of a non-linear mapping (in general, nonconvex and not necessarily point-wise) and an affine transform. That is, the input and hidden layers are given by $h^0 := x,\ h^l = \sigma(W_l, h^{l-1})$ for $l = 1, \cdots, L$. The output layer is $y = W_{L+1}h^L$, where $W_l \in \mathbb{R}^{d_l \times d_{l-1}}$ for $l = 1, \cdots, L + 1$ are parameters. Notice that the output layer is just a linear transformation which makes sure that the output is not restrict on some area (especially for some bounded σ where $\sigma(\cdot)$ represents the non-linear mapping between layers).

Consider the case with two hidden layers. We shall restrict ourselves to the point-wise sigmoid function, i.e.,

$$\sigma(v) = \frac{1}{1 + \exp(-v)} \quad \text{for any } v \in \mathbb{R}^d.$$

In the proof of BSDE deep learning-based methods, we only consider the case of sigmoid function (bounded derivative in general).

Consider the following unconstrained minimization problem

$$\min_W f(W) := \mathbb{E}_{x,y}\mathcal{L}(x, y; W), \tag{4.1}$$

where $\mathcal{L}(\cdot)$ is the loss function parameterized by W and (x, y) is the sample of data. In this network, we choose quadratic loss function,

$$h^0 = x, \quad h^l = \sigma(W_l h^{l-1}), \quad \mathcal{L}(x, y; W) = \|y - W_{L+1}h^L\|^2, \tag{4.2}$$

where $\|\cdot\|$ denotes the Euclidean norm.

In the 2-NN, we can simply denote as

$$\mathcal{L}(x, y; W) = \|y - W_2(\sigma(W_1\sigma(W_0 x)))\|^2. \tag{4.3}$$

The stochastic gradient (update using B samples $\eta^1, \cdots, \eta^B$) is $W \leftarrow W - \gamma G(\eta; W)$, where γ is the learning rate, and $G(\eta; W)$ computed at W is given by

$$G(\eta; W) = \frac{1}{B} \sum_{i=1}^{B} \nabla_W \mathcal{L}(\eta^i; W). \tag{4.4}$$

Consider a 2-NN with the corresponding loss function (4.3). We use the random stopping stochastic gradients ($R \sim \mathbb{P}_R(k),\ k = 1, \cdots, N$). This learning procedure is summarized in Algorithm 1. W^1 and W^R are the initial and final estimates respectively, and γ^k is the learning rate at the k-th iteration. Then we have the following theorem.

Algorithm 1: Two Layers Neural Network Randomized Stochastic Gradients

Input: $d_x, d_y, B, N, \gamma^k, \mathbb{P}_R(\cdot), \mathcal{X}, W^1$.
$R \sim \mathbb{P}_R(\cdot), \quad \mathcal{I} = 1_{d_k \times d_v}$,
for $k = 1, \cdots, R-1$, **do**
$(x^i, y^i) \sim \mathcal{X}, i = 1, \cdots, B,\ \eta^i := (x^i, y^i)$,
$W^{k+1} \leftarrow W^k - \mathcal{I} * \Big(\frac{\gamma^k}{B} \sum\limits_{i=1}^{B} \nabla_W \mathcal{L}(\eta^i; W^k)\Big)$,
end for.
Output: $W^R \in \mathbb{R}^{d_y \times d_x}$.

Theorem 4.1 *Consider a 2-NN with constant learning rate $\gamma^k = \gamma, \forall\, k$. Let $e^s_\gamma = \big(1 - \frac{13}{16}\gamma\big)$, $e^s = \frac{13 d_x d_y}{256}$. We assume that x, y, W are bounded by M (without loss of generality we set $M = 1$). The expected gradients are estimated as follows*:

$$\mathbb{E}_{R,\eta}(\|\nabla_W f(W^R)\|^2) \le \frac{1}{e^s_\gamma}\Big(\frac{D_f}{N\gamma} + \frac{e^s \gamma}{B}\Big), \tag{4.5}$$

where $D_f := f(W^1) - f^*$, and the optimal constant learning rate is $\gamma_o = \sqrt{\frac{B f(W^1)}{e^s N}}$.

The proof is sketched as follows (the details are given in the Appendix).

Step One Verify that $\nabla_W f(W)$ satisfies Lipschitz condition, i.e.,

$$\|\nabla_W f(W) - \nabla_W f(\widehat{W})\| \le L \|W - \widehat{W}\|. \tag{4.6}$$

Step Two Define $\delta := G(\eta; W) - \nabla_W f(W)$. Then we have $\mathbb{E}_\eta \delta = 0$, and it remains to prove $\mathbb{E}_\eta \delta^2 \le K$.

Step Three Following the methods used in Theorem 3.1, we can get the convergence of $\mathbb{E}_{R,\eta} \|\nabla_W f(W^k)\|^2$.

4.2 Convergence on deep learning-based algorithm

In this subsection, we prove the convergence of deep learning-based algorithm, under the formulation of the algorithm introduced in Subsection 2.2. We first propose our BSDE Randomized Stochastic Gradients algorithm.

Algorithm 2: BSDE Randomized Stochastic Gradients

Input: $T \in (0,\infty), d, \rho, K \in \mathbb{N}, \xi \in \mathbb{R}^q, \Theta^0$,
$0 = t_0 < t_1 < \cdots < t_N = T, P_R(\cdot)$ supported on $\{1, 2, \cdots, K\}$,
$\mathcal{X}_0^k = \xi, \quad \mathcal{Y}_0^{\theta,k} = \mathcal{U}^\theta, \quad \mathcal{X}_{n+1}^m = \Upsilon(t_n, t_{n+1}, \mathcal{X}_n^k, W_{t_{n+1}}^k - W_{t_n}^k)$,
$R \sim P_R(\cdot)$.

for $k = 1, \cdots, R$, **do**
$\mathcal{Y}_{n+1}^{\theta,k} = \mathcal{Y}_n^{\theta,k} - f(t_n, \mathcal{X}_n^k, \mathcal{Y}_n^{\theta,k}, \mathcal{V}_n^\theta(\mathcal{X}_n^k))(t_{n+1} - t_n) + \mathcal{V}_n^\theta(\mathcal{X}_n^k)(W_{t_{n+1}}^k - W_{t_n}^k)$,
$\phi^k(\theta,\omega) = \|\mathcal{Y}_N^{\theta,k}(\omega) - g(\mathcal{X}_N^k(\omega))\|^2, \quad \Phi^k(\theta,\omega) = (\nabla_\theta \phi^k)(\theta,\omega)$,
$\Theta_k = \Theta_{k-1} - \gamma_k \cdot \Phi^k(\Theta_{k-1})$,
end for.

Output: Θ_R.

Then we have the following convergence result of this algorithm. The main idea of the proof is as follows: We consider the whole problem as a nonconvex problem, and if we can prove our problem satisfies Assumptions A and condition (3.2), then using Theorem 3.1, we can finish the proof. Since the algorithm is combining stochastic differential equations with neural network, the results we get in Subsection 4.1 and the Euler scheme of SDE will help us.

We first give our assumptions.

Assumptions B

(i) The SDE (2.2) has a unique strong solution.

(ii) The derivative of f in BSDE (2.2) is bounded with respect to Y, Z.

(iii) Function f is Lipschitz continuous with respect to all parameters, i.e.,

$$|f(t,x,y,z) - f(\overline{t},\overline{x},\overline{y},\overline{z})| \le L(|t-\overline{t}| + |x-\overline{x}| + |y-\overline{y}| + |z-\overline{z}|).$$

(iv) The parameters θ (the layer-wise transformations) in neural network are bounded. This is a common assumption for the proof of the convergence on neural network.

(v) The learning rates $\{\gamma_k\}$ and the probability mass function $P_R(\cdot)$ in the RSG method are chosen such that $\gamma_k \le \frac{2}{M}$ (where M is the Lipschitz constant of $\nabla_\theta \psi(\theta^R)$ given in Lemma 4.2) and

$$P_R(k) := \text{Prob}\{R = k\} = \frac{2\gamma_k - M\gamma_k^2}{\sum_{k=1}^N (2\gamma_k - M\gamma_k^2)}, \quad k = 1, \cdots, K.$$

Before giving the main theorem, we need two more lemmas.

By definition $\phi(\theta,\omega) = \|y_N^\theta - g(x_N)\|^2$, let

$$\psi(\theta) = \mathbb{E}[\phi(\theta,\omega)] = \mathbb{E}\|y_N^\theta - g(x_N)\|^2, \quad \Phi(\theta,\omega) = \nabla_\theta \phi(\theta,\omega t) = 2\|y_N^\theta - g(x_N)\|\nabla_\theta y_N^\theta.$$

Lemma 4.1 *We have $\mathbb{E}\delta_k^2 \le D$, where $\delta_k := \nabla_\theta \psi(\theta_k) - \Phi(\theta_k,\omega)$, and D is a given constant.*

Proof Notice that $\mathbb{E}\delta_k = 0$, so

$$\mathbb{E}\delta_k^2 = \mathbb{E}\|\Phi^2(\theta_k,\omega)\| - (\mathbb{E}\|\Phi(\theta_k,\omega)\|)^2. \tag{4.7}$$

By Cauchy-Schwarz inequality,

$$\begin{aligned} \mathbb{E}\|\Phi^2(\theta_k,\omega)\| &= 4\mathbb{E}\|(y_N^\theta - g(x_N)^2(\nabla_\theta y_N^\theta)^2\| \\ &\le 4\mathbb{E}(\|y_N^\theta - g(x_N)\|^4)^{\frac{1}{2}}\mathbb{E}(\|\nabla_\theta y_N^\theta\|^4)^{\frac{1}{2}}, \\ \mathbb{E}\|\Phi(\theta_k,\omega)\|^2 &= 4\mathbb{E}\|(y_N^\theta - g(x_N))\nabla_\theta y_N^\theta\|^2 \\ &\le 4\mathbb{E}\|y_N^\theta - g(x_N)\|^2\mathbb{E}\|\nabla_\theta y_N^\theta\|^2. \end{aligned} \tag{4.8}$$

We split the proof into two steps.

Step One First consider the terms $\mathbb{E}\|y_N^\theta - g(x_N)\|^2$ and $\mathbb{E}\|y_N^\theta - g(x_N)\|^4$.

Denote $f_1 = \nabla_Y f(t, X, Y, Z)$ and $f_2 = \nabla_Z f(t, X, Y, Z)$. By Assumptions B(ii), f_1 and f_2 are bounded by C_1.

Recall that y_0^θ and $V_n^\theta(X_n), n = 1, \cdots, N$, are the outputs of neural network, for the activation function is bounded and has bounded derivative. Then under Assumptions B(iv) and (6.10) in Theorem 4.1, we have that y_0^θ, $V_n^\theta(X_n)$, $\mathbb{E}\|\nabla_\theta y_0^\theta\|^2$ and $\mathbb{E}\|\nabla_\theta V_n^\theta(X_n)\|^2, n = 1, \cdots, N$, are also bounded by C_2, where C_1 and C_2 are some given constants.

The discrete iterative equation is as (2.8). Taking the derivative with respect to θ, we have

$$\begin{aligned}
\nabla_\theta y_N^\theta &= \nabla_\theta y_{N-1}^\theta - (\nabla_\theta y_{N-1}^\theta f_1 \Delta t + \nabla_\theta V_{N-1}^\theta(X_{N-1}) f_2 \Delta t) + \nabla_\theta V_{N-1}^\theta(X_{N-1}) f_2 \Delta W_{t_N} \\
&= \cdots\cdots \quad \text{(iterate through } \nabla_\theta y_N, \cdots, \nabla_\theta y_1) \\
&= \nabla_\theta y_0^\theta - \Big(f_1 \Delta t (1 + f_1 \Delta t)^N \nabla_\theta y_0^\theta + \sum_{n=1}^N \frac{1 - (f_1 \Delta t)^{N-n+1}}{1 - f_1 \Delta t} f_2 \Delta t \nabla_\theta V_n^\theta(X_n) \Big) \\
&\quad + \sum_{n=1}^N \frac{1 - (f_1 \Delta t t)^{N-n+1}}{1 - f_1 \Delta t} \nabla_\theta V_n^\theta(X_n) \Delta W_{t_n} \\
&\le \nabla_\theta y_0^\theta - C_1 C_2 \frac{T}{N} + C_1 \sum_{n=1}^N \frac{T}{N} \nabla_\theta V_n^\theta(X_n) + C_1 \sum_{n=1}^N \nabla_\theta V_n^\theta(X_n) \Delta W_{t_n} \\
&\le C + \nabla_\theta y_0^\theta + C_1 \sum_{n=1}^N \nabla_\theta V_n^\theta(X_n) \Delta W_{t_n}, \qquad (4.9)
\end{aligned}$$

where C is only depend on C_1, C_2, T.

With $\mathbb{E}\big\| \sum_{n=1}^N \nabla_\theta V_n^\theta(X_n) \Delta W_{t_n} \big\|^2 \le N\big(C_2 \frac{T}{N}\big) = C_2 T$, then it is easy to know that $\mathbb{E}\|\nabla_\theta y_N^\theta\|^2$ is bounded and so is $\mathbb{E}\|\nabla_\theta y_N^\theta\|^4$.

Step Two We assert that $\mathbb{E}\|y_N^\theta - g(x_N)\|^2$ and $\mathbb{E}\|y_N^\theta - g(x_N)\|^4$ are bounded. From (2.8), we have

$$\begin{aligned}
y_{k+1}^\theta = y_k^\theta &- f(t_k, X_k, y_k^\theta, V_k^\theta(X_k))(t_{k+1} - t_k) \\
&+ V_k^\theta(X_k)(W_{t_{k+1}} - W_{t_k}), \quad k = 0, 1, \cdots, N-1, \qquad (4.10)
\end{aligned}$$

then we can generate $\{y_t\}$ from $\{y_k\}$ by

$$\begin{aligned}
y_t^\theta &= y_k^\theta - f(t_k, X_k, y_k^\theta, V_k^\theta(X_k))(t - t_k) + V_k^\theta(X_k)(W_t - W_{t_k}) \\
&= y_k^\theta - \int_{t_k}^t f(t_k, X_k, y_k^\theta, V_k^\theta(X_k)) \mathrm{d}s + \int_{t_k}^t V_k^\theta(X_k) \mathrm{d}W_s, \quad t \in (t_k, t_{k+1}]. \qquad (4.11)
\end{aligned}$$

Denote $n_t = \max\{k \mid t_k \le t\}$ and $\pi(t) = t_{n_t}$. So

$$y_T^\theta = y_0^\theta - \int_0^T f(\pi(s), X_{\pi(s)}, y_{\pi(s)}^\theta, V_{\pi(s)}^\theta(X_{\pi(s)})) \mathrm{d}s + \int_0^T V_{\pi(s)}^\theta(X_{\pi(s)}) \mathrm{d}W_s. \qquad (4.12)$$

We transform the BSDE (2.2) from backward to forward and replace Z_t with $V_t^*(X_t)$, and denote y_0^* as the initial value. Then we have

$$g(X_N) = y_0^* - \int_0^T f(s, X_s, y_s, V_s^*(X_s)) \mathrm{d}s + \int_0^T V_s^*(X_s) \mathrm{d}W_s. \qquad (4.13)$$

Its Euler scheme is

$$\overline{y_T} = y_0^* - \int_0^T f(\pi(s), X_{\pi(s)}, \overline{y}_{\pi(s)}, V^*_{\pi(s)}(X_{\pi(s)}))\mathrm{d}s + \int_0^T V^*_{\pi(s)}(X_{\pi(s)})\mathrm{d}W_s. \tag{4.14}$$

So $\|y_N^\theta - g(x_N)\| = \|y_T^\theta - \overline{y_T} + \overline{y_T} - g(x_N)\|$ (notice that $y_N = y_T$). Using Lipschitz continuity of f, we have

$$\begin{aligned}
|\overline{y_T} - g(x_N)| &= \int_0^T |f(s, X_s, y_s, V_s^*(X_s)) - f(\pi(s), X_{\pi(s)}, \overline{y}_{\pi(s)}, V^*_{\pi(s)}(X_{\pi(s)}))|\mathrm{d}s \\
&\quad + \int_0^T |V^*_{\pi(s)}(X_{\pi(s)}) - V_s^*(X_s)|\mathrm{d}W_s \\
&\le \int_0^T L|y_s - \overline{y}_{\pi(s)}|\mathrm{d}s + \int_0^T |V_s^*(X_s) - V^*_{\pi(s)}(X_{\pi(s)})|\mathrm{d}W_s \\
&\quad + \int_0^T L(|s - \pi(s)| + |X_s - X_{\pi(s)}| + |V_s^*(X_s) - V^*_{\pi(s)}(X_{\pi(s)})|)\mathrm{d}s \\
&\le \int_0^T L|y_s - \overline{y}_s|\mathrm{d}s + \int_0^T CL\Delta t\mathrm{d}s + \int_0^T C\Delta t\mathrm{d}W_s.
\end{aligned} \tag{4.15}$$

Squaring and taking expectation on each side and by BDG inequality, we have

$$\mathbb{E}\|\overline{y_T} - g(x_N)\|^2 \le 2TL\int_0^T \mathbb{E}\|\|y_s - \overline{y}_s|\|\mathrm{d}s + 2C. \tag{4.16}$$

Using Gronwall's inequality, we get the boundedness of $\mathbb{E}\|\overline{y_T} - g(x_N)\|^2$. In the same way, we have the bound of $\mathbb{E}\|\overline{y_T} - g(x_N)\|^4$.

With the Lipschitz continuity of f, we have

$$\begin{aligned}
y_T^\theta - \overline{y}_T &= (y_0 - \overline{y}_0) + \int_0^T (V^\theta_{\pi(s)}(X_{\pi(s)}) - V^*_{\pi(s)}(X_{\pi(s)}))\mathrm{d}W_s \\
&\quad - \int_0^T (f(\pi(s), X_{\pi(s)}, y^\theta_{\pi(s)}, V^\theta_{\pi(s)}(X_{\pi(s)})) \\
&\quad - f(\pi(s), X_{\pi(s)}, \overline{y}_{\pi(s)}, V^*_{\pi(s)}(X_{\pi(s)})))\mathrm{d}s.
\end{aligned} \tag{4.17}$$

Following the same method, we can get the boundedness of $\mathbb{E}\|y_T^\theta - \overline{y_T}\|^2$ and $\mathbb{E}\|y_T^\theta - \overline{y_T}t\|^4$. Combining Steps One, Two and (4.8), we have proved that $\mathbb{E}\delta_k^2 \le D$.

Lemma 4.2 *We assert that $\nabla_\theta\psi(\theta,\omega) := \mathbb{E}[\nabla_\theta\phi(\theta,\omega)]$ is Lipschitz continuous with respect to θ, i.e.,*

$$\|\nabla_\theta\psi(\widehat{\theta},\omega) - \nabla_\theta\psi(\theta,\omega)\| \le M\|\widehat{\theta} - \theta\|, \quad \forall\, \theta, \widehat{\theta} \in \mathbb{R}^n.$$

Proof Since $\mathbb{E}\|\nabla_\theta\phi(\theta,\omega)\| = 2\mathbb{E}\|(y_N^\theta - g(x_N))\nabla_\theta y_N^\theta\|$, we need to prove the Lipschitz continuity of $\mathbb{E}[y_N^\theta]$ and $\mathbb{E}[\nabla_\theta y_N^\theta]$:

$$\begin{aligned}
|y_T^\theta - y_T^{\widehat{\theta}}| &= |y_0^\theta - y_0^{\widehat{\theta}}| + \int_0^T |V^\theta_{\pi(s)}(X_{\pi(s)}) - V^{\widehat{\theta}}_{\pi(s)}(X_{\pi(s)})|\mathrm{d}W_s \\
&\quad + \int_0^T |f(\pi(s), X_{\pi(s)}, y^\theta_{\pi(s)}, V^\theta_{\pi(s)}(X_{\pi(s)})) - f(\pi(s), X_{\pi(s)}, y^{\widehat{\theta}}_{\pi(s)}, V^{\widehat{\theta}}_{\pi(s)}(X_{\pi(s)}))|\mathrm{d}s \\
&\le |y_0^\theta - y_0^{\widehat{\theta}}| + \int_0^T L|y^\theta_{\pi(s)} - y^{\widehat{\theta}}_{\pi(s)}|\mathrm{d}s + \int_0^T L|V^\theta_{\pi(s)}(X_{\pi(s)}) - V^{\widehat{\theta}}_{\pi(s)}(X_{\pi(s)})|\mathrm{d}s
\end{aligned}$$

$$+\int_0^T |V^{\theta}_{\pi(s)}(X_{\pi(s)}) - V^{\widehat{\theta}}_{\pi(s)}(X_{\pi(s)})|\mathrm{d}W_s. \tag{4.18}$$

Since y^{θ}_0, and $V^{\theta}_n(X_n),\ n = 1, \cdots, N$, are the outputs of neural network, using the Lipschitz continuity of $V^{\theta}_{\pi(s)}(X_{\pi(s)})$ in Theorem 4.1, and taking expectation on both sides of the last inequality, we have

$$\mathbb{E}|y^{\theta}_T - y^{\widehat{\theta}}_T| \le \int_0^T L\mathbb{E}|y^{\theta}_{\pi(s)} - y^{\widehat{\theta}}_{\pi(s)}|\mathrm{d}s + D|\theta - \widehat{\theta}| + LDT|\theta - \widehat{\theta}|. \tag{4.19}$$

Further more, using Gronwall's inequality, we have

$$\mathbb{E}|y^{\theta}_T - y^{\widehat{\theta}}_T| \le M|\theta - \widehat{\theta}|. \tag{4.20}$$

On the other side, from the third equation in (4.9), we can also get the Lipschitz continuity of $\mathbb{E}[\nabla_\theta y^{\theta}_N]$. Combining the boundedness of $\mathbb{E}\|(y^{\theta}_N - g(x_N))\|$ and $\mathbb{E}\|\nabla_\theta y^{\theta}_N\|$, we finally get the Lipschitz continuity of $\nabla_\theta\psi(\theta)$.

With this two lemmas, we can finally give the proof of Theorem 4.2.

Theorem 4.2 *Following the algorithm introduced in Subsection* 2.2*, we use stochastic gradient decent method, quadratic loss function and sigmoid function as activation function in our neural network. Under Assumptions B and Algorithm* 2*, we have that*

$$\mathbb{E}_R[\|\nabla_\theta\psi(\theta^R)\|^2] \le \frac{M}{\sum\limits_{k=1}^{N}(2\gamma_k - M\gamma_k^2)}\Big[\frac{2(\psi(\theta^1) - \psi^*)}{M} + D\sum_{k=1}^{N}\gamma_k^2\Big],$$

where D is the bound of $\mathbb{E}\delta_k^2$ introduced in Lemma 4.1*.*

Proof Using the Lipschitz continuity of $\nabla\psi(\theta)$, we have

$$\begin{aligned}
\psi(\theta^{k+1}) &\le \psi(\theta^k) + \langle\nabla_\theta\psi(\theta^k), \theta_{k+1} - \theta_k\rangle + \frac{M}{2}\gamma_k^2\|\Phi(\theta_k, \omega)\|^2\\
&= \psi(\theta^k) - \gamma_k\langle\nabla_\theta\psi(\theta^k), \Phi(\theta_k, \omega)\rangle + \frac{M}{2}\gamma_k^2\|\Phi(\theta_k, \omega)\|^2\\
&= \psi(\theta^k) - \gamma_k\|\nabla_\theta\psi(\theta^k)\|^2 + \gamma_k\langle\nabla_\theta\psi(\theta^k), \delta_k\rangle\\
&\quad + \frac{M}{2}\gamma_k^2[\|\nabla_\theta\psi(\theta^k)\|^2 + 2\langle\nabla_\theta\psi(\theta^k), \delta_k\rangle + \|\delta_k\|^2]\\
&= \psi(\theta^k) - \Big(\gamma_k - \frac{M}{2}\gamma_k^2\Big)\|\nabla_\theta\psi(\theta^k)\|^2\\
&\quad - (\gamma_k - M\gamma_k^2)\langle\nabla_\theta\psi(\theta^k), \delta_k\rangle + \frac{M}{2}\gamma_k^2\|\delta_k\|^2.
\end{aligned} \tag{4.21}$$

Further, we obtain

$$\begin{aligned}
&\sum_{k=1}^{N}\Big(\gamma_k - \frac{M}{2}\gamma_k^2\Big)\|\nabla_\theta\psi(\theta^k)\|^2\\
&\le \psi(\theta^1) - \psi(\theta^{N+1}) - \sum_{k=1}^{N}(\gamma_k - M\gamma_k^2)\langle\nabla_\theta\psi(\theta^k), \delta_k\rangle + \frac{M}{2}\sum_{k=1}^{N}\gamma_k^2\|\delta_k\|^2\\
&\le \psi(\theta^1) - \psi^* - \sum_{k=1}^{N}(\gamma_k - M\gamma_k^2)\langle\nabla_\theta\psi(\theta^k), \delta_k\rangle + \frac{M}{2}\sum_{k=1}^{N}\gamma_k^2\|\delta_k\|^2,
\end{aligned} \tag{4.22}$$

where ψ^* is the lower bound of ψ. Then from $\mathbb{E}\delta_k = 0$ and $\mathbb{E}\delta_k^2 \le D$, we obtain

$$\sum_{k=1}^{N}\Big(\gamma_k - \frac{M}{2}\gamma_k^2\Big)\mathbb{E}\|\nabla_\theta\psi(\theta^k)\|^2 \le \psi(\theta^1) - \psi^* + \frac{MD}{2}\sum_{k=1}^{N}\gamma_k^2. \tag{4.23}$$

Dividing both sides of the above inequality by $M\sum\limits_{k=1}^{N}\big(\gamma_k - M\frac{\gamma_k^2}{2}\big)$ and noting that

$$\mathbb{E}_R[\|\nabla_\theta\psi(\theta^R)\|^2] = \frac{\sum\limits_{k=1}^{N}(2\gamma_k - M\gamma_k^2)\mathbb{E}\|\nabla_\theta\psi(\theta^k)\|^2}{\sum\limits_{k=1}^{N}(2\gamma_k - M\gamma_k^2)}, \tag{4.24}$$

we conclude

$$\mathbb{E}_R[\|\nabla_\theta\psi(\theta^R)\|^2] \le \frac{M}{\sum\limits_{k=1}^{N}(2\gamma_k - M\gamma_k^2)}\Big[\frac{2(\psi(\theta^1)-\psi^*)}{M} + D\sum_{k=1}^{K}\gamma_k^2\Big]. \tag{4.25}$$

Remark 4.1 Suppose that the learning rates $\{\gamma_k\}$ are set to

$$\gamma_k = \min\Big\{\frac{1}{M}, \frac{C}{\sqrt{DMK}}\Big\}, \quad k = 1, \cdots, K \tag{4.26}$$

for some $C > 0$. Then, we have

$$\frac{1}{M}\mathbb{E}_R[\|\nabla_\theta\psi(\theta^R)\|^2] \le \frac{MA}{K} + \Big(C + \frac{A}{C}\Big)\frac{\sqrt{DM}}{\sqrt{K}}, \tag{4.27}$$

where $A = \frac{2(\psi(\theta^1-\psi^*))}{M}$.

So, we have arrived at the $O(\sqrt{K}^{-1})$ rate of convergence in terms of $\mathbb{E}_R[\|\nabla_\theta\psi(\theta^R t)\|^2]$.

We have get the gradient convergence result. First, since our problem is nonconvex, we can only get the gradient convergence results even if in simple nonconvex stochastic programming problem. Second, in the numerical experiment, we can observe that when we change hidden layers from two to three, the experiment result (see in Section 5) may not convergence.

5 Numerical Experiment

We now give a numerical example to explain why we can only get the gradient convergence for deep BSDE algorithm.

We test the deep BSDE algorithm in the case of Allen-Cahn PDE (see (5.1)). Let $x, w \in \mathbb{R}^d$, $y \in \mathbb{R}$, $z \in \mathbb{R}^{1\times d}$, $k \in \mathbb{N}$ that $\gamma_k = 5\cdot 10^{-4}$, $d = 100$, $T = \frac{3}{10}$, $N = 20$, $\mu(t,x) = 0$, $\sigma(t,x)w = \sqrt{2}w$, $\xi = (0,0,\cdots,0) \in \mathbb{R}^d$, $f(t,x,y,z) = y - y^3$, and $g(x) = \big[2 + \frac{2}{5}\|x\|^2_{\mathbb{R}^d}\big]^{-1}$. Let $u \in C^{1,2}([0,T]\times\mathbb{R}^d, \mathbb{R})$, satisfy that $u(T,x) = g(x)$ and

$$\frac{\partial u}{\partial t}(t,x) + u(t,x) - [u(t,x)]^3 + (\Delta_x u)(t,x) = 0 \tag{5.1}$$

for all $(t,x) \in ([0,T]\times\mathbb{R}^d)$.

The following results (see in Figure 2) show the difference between the two and three hidden layers. The results show that the approximation error and the gradient approximation error are convergence when the number of hidden layers is two. But when the number of hidden layers changes to three, the approximation error fails to converge while the gradient approximation error still does.

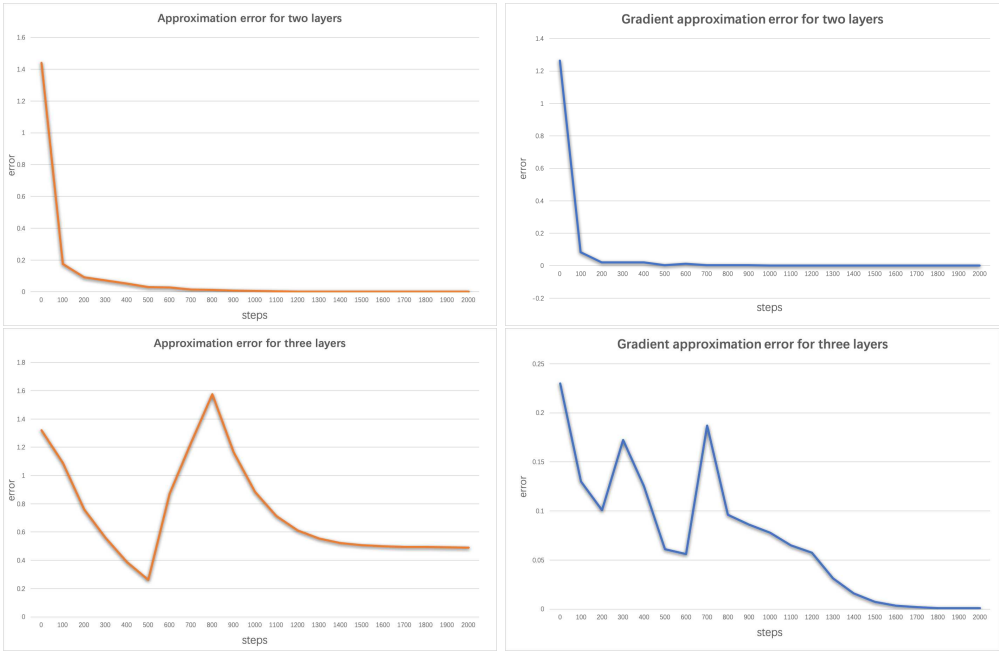

Figure 2 Numerical results for Allen-Cahn PDE (5.1).

6 Appendix

We first give the proof of Theorem 3.1 (see [14, Theorem 2.1]).

Proof of Theorem 3.1 First, from the Lipschitz continuity of ∇f, we have

$$|f(y) - f(x) - \langle \nabla f(x), y - x\rangle| \le \frac{L}{2}\|y - x\|^2, \quad \forall\, x, y \in \mathbb{R}^n. \tag{6.1}$$

Define $\delta_k \equiv G(x_k, \xi_k) - \nabla f(x_k),\ k \ge 1$. We have for any $k = 1, \cdots, N$,

$$\begin{aligned} f(x_{k+1}) &\le f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k\rangle + \frac{L}{2}\gamma_k^2\|G(x_k, \xi_k)\|^2 \\ &= f(x_k) - \gamma_k\langle \nabla f(x_k), G(x_k, \xi_k)\rangle + \frac{L}{2}\gamma_k^2\|G(x_k, \xi_k)\|^2 \\ &= f(x_k) - \gamma_k\|\nabla f(x_k)\|^2 - \gamma_k\langle \nabla f(x_k), \delta_k\rangle \\ &\quad + \frac{L}{2}\gamma_k^2[\|\nabla f(x_k)\|^2 + 2\langle \nabla f(x_k), \delta_k\rangle + \|\delta_k\|^2] \\ &= f(x_k) - \Big(\gamma_k - \frac{L}{2}\gamma_k^2\Big)\|\nabla f(x_k)\|^2 - (\gamma_k - L\gamma_k^2)\langle \nabla f(x_k), \delta_k\rangle + \frac{L}{2}\gamma_k^2\|\delta_k\|^2. \end{aligned} \tag{6.2}$$

Further, we obtain

$$\begin{aligned} &\sum_{k=1}^{N}\Big(\gamma_k - \frac{L}{2}\gamma_k^2\Big)\|\nabla f(x_k)\|^2 \\ &\le f(x_1) - f(x_{N+1}) - \sum_{k=1}^{N}(\gamma_k - L\gamma_k^2)\langle \nabla f(x_k), \delta_k\rangle + \frac{L}{2}\sum_{k=1}^{N}\gamma_k^2\|\delta_k\|^2 \\ &\le f(x_1) - f^* - \sum_{k=1}^{N}(\gamma_k - L\gamma_k^2)\langle \nabla f(x_k), \delta_k\rangle + \frac{L}{2}\sum_{k=1}^{N}\gamma_k^2\|\delta_k\|^2, \end{aligned} \tag{6.3}$$

where the last inequality follows from the fact that $f(x_{N+1}) \geq f^*$. Note that the search point x_k is random because of depending on the generated random process by $\xi_{[k-1]}$. Under Assumption A, $\mathbb{E}[\|\delta_k\|^2] \leq \sigma^2$, and

$$\mathbb{E}[\langle \nabla f(x_k), \delta_k \rangle | \xi_{[k-1]}] = 0. \tag{6.4}$$

We take expectations (with respect to $\xi_{[N]}$) on both sides of (6.3), and obtain

$$\sum_{k=1}^{N} \Big(\gamma_k - \frac{L}{2}\gamma_k^2\Big)\mathbb{E}_{\xi_{[N]}}\|\nabla f(x_k)\|^2 \leq f(x_1) - f^* + \frac{L\sigma^2}{2}\sum_{k=1}^{N}\gamma_k^2. \tag{6.5}$$

Dividing both sides of the last inequality by $L\sum\limits_{k=1}^{N}\big(\gamma_k - \frac{L\gamma_k^2}{2}\big)$,

$$\mathbb{E}[\|\nabla f(x_R)\|^2] = \mathbb{E}_{R,\xi_{[N]}}[\|\nabla f(x_R)\|^2] = \frac{\sum\limits_{k=1}^{N}(2\gamma_k - L\gamma_k^2)\mathbb{E}_{\xi_{[N]}}\|\nabla f(x_k)\|^2}{\sum\limits_{k=1}^{N}(2\gamma_k - L\gamma_k^2)}. \tag{6.6}$$

We conclude

$$\frac{1}{L}\mathbb{E}[\|\nabla f(x_R)\|^2] \leq \frac{1}{\sum\limits_{k=1}^{N}(2\gamma_k - L\gamma_k^2)}\Big[\frac{2(f(x_1) - f^*)}{L} + \sigma^2\sum_{k=1}^{N}\gamma_k^2\Big], \tag{6.7}$$

which together with (3.7) implies (3.6).

In the proof of Theorem 4.1, we adapt the proof of [15], from the case of no hidden layer to our case of two hidden layers.

Proof of Theorem 4.1 Let $\delta = G(\eta; W) - \nabla_W f(W)$, by definition

$$f(W) = \mathbb{E}_\eta \mathcal{L}(\eta; W), \quad G(\eta; W) = \frac{1}{B}\sum_{i=1}^{B} g(\eta; W) \quad \text{with} \quad g(\eta; W) = \nabla_W \mathcal{L}(\eta; W). \tag{6.8}$$

It is easy to know that the Lipschitz constant of the activation function $\sigma(\cdot)$ is $\frac{1}{4}$. Denote δ to be the vector with length $d_x * d_y$, and each component δ_{ij} is $g^{ij}(\eta; W) - \nabla_{W_{ij}} f(W)$, where $i = 1, \cdots, d_y$ and $j = 1, \cdots, d_x$. Hence

$$\mathbb{E}_\eta\|\delta\|^2 = \mathbb{E}_\eta \sum_{ij} |\delta_{ij}|^2 = \sum_{ij} \mathbb{E}_\eta (g^{ij}(\eta; W) - \nabla_{W_{ij}} f(W))^2, \tag{6.9}$$

where $g^{ij}(\eta; W)$ are the noisy gradients with respect to W_{ij}.

$$g^{ij}(\eta; W) = \nabla_W^{ij}\mathcal{L}(\eta; W) = -2W_2(y_i - W_2 A)(A + W_2(A(1-A)(B + W_1 W_0 B(1-B))))x_j,$$

where $A = \sigma_i(W_1\sigma_i(W_0 x))$ and $B = \sigma_i(W_0 x)$.

Since x, y, W are bounded and $|g^{ij}(\eta; W)| \leq 1$,

$$\begin{aligned}\mathrm{Var}_\eta(g^{ij}(\eta; W)) &= \mathbb{E}_\eta(g^{ij}(\eta; W))^2 - (\mathbb{E}_\eta g^{ij}(\eta; W))^2 \\ &\leq \mathbb{E}_\eta g^{ij}(\eta; W)\Big(\frac{1}{2} - \mathbb{E}_\eta g^{ij}(\eta; W)\Big) \leq \frac{1}{16} \;\Rightarrow\; \mathbb{E}_\eta\|\delta\|^2 \leq \frac{d_x d_y}{16},\end{aligned} \tag{6.10}$$

and we also have

$$|\nabla_W^{ij}\mathcal{L}(\eta;W)-\nabla_W^{ij}\mathcal{L}(\eta;\widehat{W})|\le\frac{25}{2}|\sigma_i(Wx)-\sigma_i(\widehat{W}x)|\le\frac{25}{8}|W_{ij}-\widehat{W}_{ij}|. \tag{6.11}$$

Using $\nabla_W f(W)=\mathbb{E}_\eta\nabla_W\mathcal{L}(\eta;W)$, we have

$$\|\nabla_W f(W)-\nabla_W f(\widehat{W})\|\le\sqrt{\sum_{ij}\Big(\frac{25}{8}\Big)^2|W_{ij}-\widehat{W}_{ij}|^2}=\frac{25}{8}\|W-\widehat{W}\|. \tag{6.12}$$

Following the method used in Theorem 3.1, we have

$$\begin{aligned}
f(W^{k+1})&\le f(W^k)-\gamma^k\langle\nabla_W f(W^k),G(\eta^k;W^k)\rangle+\frac{25}{16}(\gamma^k)^2\|G(\eta^k;W^k)\|^2\\
&\le f(W^k)-\frac{\gamma^k}{B}\sum_{b=1}^B\langle\nabla_W f(W^k),g(\eta^{b,k};W^k)\rangle+\frac{25}{16B^2}(\gamma^k)^2\Big\|\sum_{b=1}^B g(\eta^{b,k};W^k)\Big\|^2\\
&=f(W^k)-\Big(\gamma^k-\frac{25}{16}(\gamma^k)^2\Big)\|\nabla_W f(W^k)\|^2\\
&\quad-\frac{1}{B}\Big(\gamma^k-\frac{25}{8}(\gamma^k)^2\Big)\sum_{b=1}^B\langle\nabla_W f(W^k),\delta^{b,k}\rangle+\frac{25}{16B^2}(\gamma^k)^2\Big\|\sum_{b=1}^B\delta^{b,k}\Big\|^2,
\end{aligned} \tag{6.13}$$

where $\eta^{b,k}$ denotes the sample used for the b-th noisy gradient computation at the k-th iteration. Adding up the above inequalities over N iterations

$$\begin{aligned}
&\sum_{k=1}^N\Big(\gamma^k-\frac{25}{16}(\gamma^k)^2\Big)\|\nabla_W f(W^k)\|^2\\
&\le f(W^1)-f^*-\frac{1}{B}\sum_{k=1}^N\sum_{b=1}^B\Big(\gamma^k-\frac{25}{8}(\gamma^k)^2\Big)\langle\nabla_W f(W^k),\delta^{b,k}\rangle\\
&\quad+\frac{25}{16B^2}\sum_{k=1}^N(\gamma^k)^2\Big\|\sum_{b=1}^B\delta^{b,k}\Big\|^2.
\end{aligned} \tag{6.14}$$

Taking expectations with respect to the random samples η, the second term from the right hand side will be 0. And it is easy to know

$$\mathbb{E}_\eta\Big\|\sum_{b=1}^B\delta^{b,k}\Big\|^2=\mathbb{E}_\eta\Big\|\sum_{b=1}^B\delta^{b,k}\Big\|^2+\mathbb{E}_\eta\|\delta^{B,k}\|^2=\sum_{b=1}^B\mathbb{E}_\eta\|\delta^{b,k}\|^2. \tag{6.15}$$

This becomes

$$\mathbb{E}_\eta\Big[\frac{25}{16B^2}\sum_{k=1}^N(\gamma^k)^2\Big\|\sum_{b=1}^B\delta^{b,k}\Big\|^2\Big]=\frac{25}{16B^2}\sum_{k=1}^N(\gamma^k)^2\mathbb{E}_\eta\Big\|\sum_{b=1}^B\delta^{b,k}\Big\|^2\le\frac{25d_xd_y}{256B}\sum_{k=1}^N(\gamma^k)^2. \tag{6.16}$$

Combining (6.13)–(6.14) and (6.16), we get

$$\sum_{k=1}^N\Big(\gamma^k-\frac{25}{16}(\gamma^k)^2\Big)\mathbb{E}_\eta\|\nabla_W f(W^k)\|^2\le f(W^1)-f^*+\frac{e^s}{B}\sum_{k=1}^N(\gamma^k)^2. \tag{6.17}$$

Let $p_R^k = \gamma^k - \frac{25}{16}(\gamma^k)^2$ and $\mathbb{E}_{R,\eta}\|\nabla_W f(W^k)\|^2 := \frac{\sum_{k=1}^N p_R^k(\mathbb{E}_\eta\|\nabla_W f(W^k)\|^2)}{\sum_{k=1}^N p_R^k}$. So we have

$$\mathbb{E}_{R,\eta}\|\nabla_W f(W^k)\|^2 \le \frac{f(W^1) - f^* + \frac{e^s}{B}\sum_{k=1}^N (\gamma^k)^2}{\sum_{k=1}^N \left(\gamma^k - \frac{25}{16}(\gamma^k)^2\right)}. \tag{6.18}$$

Let $\gamma^k = \gamma$,

$$\mathbb{E}_{R,\eta}\|\nabla_W f(W^k)\|^2 := \frac{1}{N\gamma e_\gamma^s}\Big(f(W^1) - f^* + \frac{e^s N\gamma^2}{B}\Big) \le \frac{1}{e_\gamma^s}\Big(\frac{D_f}{N\gamma} + \frac{e^s\gamma}{B}\Big). \tag{6.19}$$

Balancing the two terms in the bound, the optimal constant learning rate is

$$\gamma_o = \sqrt{\frac{BD_f}{e^s N}} \approx \sqrt{\frac{Bf(W^1)}{e^s N}}. \tag{6.20}$$

Acknowledgement The authors would like to thank the anonymous reviewers for their careful work and many useful comments.

References

[1] Beck, C., Becker, S., Grohs, P., et al., Solving stochastic differential equations and Kolmogorov equations by means of deep learning. arXiv: 1806.00421, 2018

[2] Bender, C. and Zhang, J., Time discretization and Markovian iteration for coupled FBSDEs, *The Annals of Applied Probability*, **18**(1), 2008, 143–177.

[3] Bouchard, B. and Touzi, N., Discrete-time approximation and Monte-Carlo simulation of backward stochastic differential equations, *Stochastic Processes and their applications*, **111**(2), 2004, 175–206.

[4] Carreira-Perpinan, M. and Wang, W., Distributed optimization of deeply nested systems, Appearing in Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS) 2014, Reykjavik, Iceland. JMLR: W&CP volume 33.

[5] Cvitanic, J. and Zhang, J., The steepest descent method for forward-backward SDEs, *Electronic Journal of Probability*, **10**, 2005, 1468–1495.

[6] Delarue, F. and Menozzi, S., A forward-backward stochastic algorithm for quasi-linear PDEs, *The Annals of Applied Probability*, **16**(1), 2006, 140–184.

[7] Douglas, J., Ma, J. and Protter, P., Numerical methods for forward-backward stochastic differential equations, *The Annals of Applied Probability*, **6**(3), 1996, 940–968.

[8] E, W., Han, J. and Jentzen A., Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, *Communications in Mathematics and Statistics*, **5**(4), 2017, 349–380.

[9] E, W., Ma, C. and Wu, L., A priori estimates of the generalization error for two-layer neural networks. arXiv:1810.06397, 2018

[10] E, W., A proposal on machine learning via dynamical systems, *Communications in Mathematics and Statistics*, **5**(1), 2017, 1–11.

[11] Ghadimi, S. and Lan, G., Stochastic first- and zeroth-order methods for nonconvex stochastic programming, *SIAM Journal on Optimization*, **23**(4), 2013, 2341–2368.

[12] Han, J. and Long, J., Convergence of the deep BSDE method for coupled FBSDEs. arXiv: 1811.01165, 2018

[13] Han, J. and E, W., Deep learning approximation for stochastic control problems. arXiv: 1611.07422, 2016

[14] Huijskens, T. P., Ruijter, M. J. and Oosterlee, C. W., Efficient numerical Fourier methods for coupled forward-backward SDEs, *Journal of Computational and Applied Mathematics*, **296**, 2016, 593–612.

[15] Ithapu, V. K., Ravi, S. N. and Singh, V., On the interplay of network structure and gradient convergence in deep learning, 2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton), IEEE, 2016, 488–495

[16] Li, Q., Chen, L., Tai, C. and E, W., Maximum principle based algorithms for deep learning, *Journal of Machine Learning Research*, **18**(165), 2017, 1–29.

[17] Ma, J., Shen, J. and Zhao, Y., On numerical approximations of forward-backward stochastic differential equations, *SIAM Journal on Numerical Analysis*, **46**(5), 2008, 2636–2661.

[18] Malek, A. and Beidokhti, R., Numerical solution for high order differential equations using a hybrid neural network-optimization method, *Appl. Math. Comput.*, **183**(1), 2006 ,260–271.

[19] Nemirovski, A., Juditsky, A., Lan, G. and Shapiro, A., Robust stochastic approximation approach to stochastic programming, *SIAM Journal on Optimization*, **19**(4), 2009, 1574–1609.

[20] Pardoux, E. and Peng, S., Backward stochastic differential equations and quasilinear parabolic partial differential equations, Stochastic Partial Differential Equations and Their Applications, Springer-Verlag, Berlin, Heidelberg, 1992, 200–217.

[21] Rudd, K., Solving Partial Differential Equations Using Artificial Neural Networks, Ph.D. Thesis, Duke University, 2013.

[22] Ruijter, M. J. and Oosterlee, C. W., Numerical Fourier method and second-order Taylor scheme for backward SDEs in finance, *Applied Numerical Mathematics*, **103**, 2016, 1–26.

[23] Shao, H. and Zheng, G., Convergence analysis of a back-propagation algorithm with adaptive momentum, *Neurocomputing*, **74**(5), 2011, 749–752.

[24] Sirignano, J. and Spiliopoulos, K., DGM: A deep learning algorithm for solving partial differential equations, *Journal of Computational Physics*, **375**, 2018, 1339–1364.

[25] Pardoux, E. and Tang, S., Forward-backward stochastic differential equations and quasilinear parabolic PDEs, *Probability Theory and Related Fields*, **114**(2), 1999, 123–150.

[26] Xu, Y. and Yin, W., A globally convergent algorithm for nonconvex optimization based on block coordinate update, *Journal of Scientific Computing*, **72**(2), 2017, 700–734.

[27] Zeng, J., Ouyang, S., Lau, T. T. K., et al., Global convergence in deep learning with variable splitting via the Kurdyka-łojasiewicz property. arXiv: 1803.00225, 2018

[28] Zhang, X. and Zhang, N., A study on the convergence of gradient method with momentum for sigma-pi-sigma neural networks, *Journal of Applied Mathematics and Physics*, **6**(04), 2018, 880–887.

[29] Zou, D., Cao, Y., Zhou, D. and Gu, Q., Stochastic gradient descent optimizes over-parameterized deep ReLU networks. arXiv: 1811.08888, 2018