Deep Learning-Based Algorithms for High-Dimensional PDEs and Control Problems

Weinan E

Department of Mathematics and PACM,
Princeton University
Joint work with Jiequn Han and Arnulf Jentzen

October 9, 2019

Table of Contents

Outline

Deep learning-based algorithms for

- stochastic control problems in high dimension

  Jiequn Han and Weinan E, "Deep learning approximation for stochastic control problems", NIPS Workshop on Deep Reinforcement Learning (2016)

- high dimensional nonlinear PDEs, including the HJB equation, based on stochastic control formulation

  Weinan E, Jiequn Han and Arnulf Jentzen, "Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations", Communications in Mathematics and Statistics (2017)

  Jiequn Han, Arnulf Jentzen and Weinan E, "Solving high-dimensional partial differential equations using deep learning", Proceedings of the National Academy of Sciences (2018)

Motivating Examples

- The Hamilton-Jacobi-Bellman equation in stochastic control

$$v_t + \max_a \left\{ \frac{1}{2} \mathrm{Tr}\Big(\sigma\sigma^{\mathrm{T}} (\mathsf{Hess}_x v) \Big) + \nabla v \cdot b + f \right\} = 0,$$
$$v(x_t) = \max_a \left\{ f(x_t, a) + \gamma \mathbb{E} v(x_{t+1}) \right\}.$$

- The Black-Scholes equation for pricing financial derivatives,

$$v_t + \frac{1}{2} \Delta v + r \nabla v \cdot x - rv + f = 0.$$

- Reinforcement learning (model-based)

$$\max_\pi \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right].$$

Curse of Dimensionality

- The dimension can be easily large in practice.

Equation	Dimension
Black-Scholes equation	# of underlying financial assets
HJB equation	the same as the state space

- A key computational challenge is the curse of dimensionality: the complexity is exponential in dimension d for finite difference/element method – usually unavailable for $d \geq 4$.

Related Work in High-dimensional Case

- Linear parabolic PDEs: Monte Carlo methods based on the Feynman-Kac formula
- Semilinear parabolic PDEs:
  1. branching diffusion approach (Henry-Labordère 2012, Henry-Labordère et al. 2014)
  2. multilevel Picard approximation (E and Jentzen et al. 2015)
- Hamilton-Jacobi PDEs: using Hopf formula and fast convex/nonconvex optimization methods (Darbon & Osher 2016, Chow et al. 2017)
- Deep reinforcement learning

Table of Contents

Linear Parabolic PDE and Feynman-Kac Formula

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

Terminal condition $u(T,x) = g(x)$.

Let

$$dX_t = \mu(t,X_t)\,dt + \sigma(t,X_t)\,dW_t,$$

Feynman-Kac formula:

$$u(t,x) = \mathbb{E}[g(X_T) + \textstyle\int_t^T f(s,X_s)ds | X_t = x].$$

Compute the solution of PDE using Monte Carlo, overcoming the curse of dimensionality.

Semilinear Parabolic PDE

$$\frac{\partial u}{\partial t}(t,x) + \frac{1}{2}\mathrm{Tr}\Big(\sigma\sigma^{\mathrm{T}}(t,x)(\mathsf{Hess}_x u)(t,x)\Big) + \nabla u(t,x)\cdot\mu(t,x) + f\big(t,x,u(t,x),\sigma^{\mathrm{T}}(t,x)\nabla u(t,x)\big) = 0.$$

Terminal condition $u(T,x) = g(x)$.

Let

$$X_t = \xi + \int_0^t \mu(s,X_s)\,ds + \int_0^t \sigma(s,X_s)\,dW_s.$$

Itô's lemma:

$$\begin{aligned} & u(t,X_t) - u(0,X_0) \\ = & -\int_0^t f\big(s,X_s,u(s,X_s),\sigma^{\mathrm{T}}(s,X_s)\,\nabla u(s,X_s)\big)\,ds \\ & + \int_0^t [\nabla u(s,X_s)]^{\mathrm{T}}\,\sigma(s,X_s)\,dW_s. \end{aligned}$$

Connection between PDE and BSDE

- BSDEs give a nonlinear Feynman-Kac representation of some nonlinear parabolic PDEs. (Pardoux & Peng 1992, El Karoui et al. 1997, etc).

- Consider the following BSDE

$$\begin{cases} X_t = \xi + \int_0^t \mu(s, X_s)\, ds + \int_0^t \sigma(s, X_s)\, dW_s, \\ Y_t = g(X_T) + \int_t^T f(s, X_s, Y_s, Z_s)\, ds - \int_t^T (Z_s)^{\mathrm{T}}\, dW_s, \end{cases}$$

  The solution is an (unique) adapted process $\{(X_t, Y_t, Z_t)\}_{t\in[0,T]}$ with values in $\mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d$.

- This is the Pontryagin maximum principle for stochastic control.

- This is also the method of characteristics for nonlinear parabolic PDEs.

Reformulating the PDE problem

- Connection between BSDE and PDE

$$Y_t = u(t, X_t) \qquad \text{and} \qquad Z_t = \sigma^{\mathrm{T}}(t, X_t)\, \nabla u(t, X_t).$$

- In other words, consider the following variational problem

$$\begin{aligned} &\inf_{Y_0, \{Z_t\}_{0 \le t \le T}} \mathbb{E}|g(X_T) - Y_T|^2, \\ s.t. \quad &X_t = \xi + \textstyle\int_0^t \mu(s, X_s)\, ds + \int_0^t \Sigma(s, X_s)\, dW_s, \\ &Y_t = Y_0 - \textstyle\int_0^t h(s, X_s, Y_s, Z_s)\, ds + \int_0^t (Z_s)^{\mathrm{T}}\, dW_s. \end{aligned}$$

The unique minimizer is the solution to the PDE.

Table of Contents

Deep BSDE Method

- Key step: approximate the unknown functions

$$X_0 \mapsto u(0, Y_0) \qquad \text{and} \qquad X_t \mapsto \sigma^{\mathrm{T}}(t, X_t)\, \nabla u(t, X_t)$$

  by feedforward neural networks ψ and ϕ.

- work with variational formulation, discretize time using Euler scheme on a grid $0 = t_0 < t_1 < \ldots < t_N = T$:

$$\begin{aligned}
&\inf_{\psi_0, \{\phi_n\}_{n=0}^{N-1}} \mathbb{E}|g(X_T) - Y_T|^2, \\
s.t. \quad & X_0 = \xi, \quad Y_0 = \psi_0(\xi), \\
& X_{t_{n+1}} = X_{t_i} + \mu(t_n, X_{t_n})\Delta t + \sigma(t_n, X_{t_n})\Delta W_n, \\
& Z_{t_n} = \phi_n(X_{t_n}), \\
& Y_{t_{n+1}} = Y_{t_n} - f(t_n, X_{t_n}, Y_{t_n}, Z_{t_n})\Delta t + (Z_{t_n})^{\mathrm{T}} \Delta W_n.
\end{aligned}$$

- there is a subnetwork at each time t_j
- Observation: we can stack all the subnetworks together to form a deep neural network (DNN) as a whole

Network Architecture

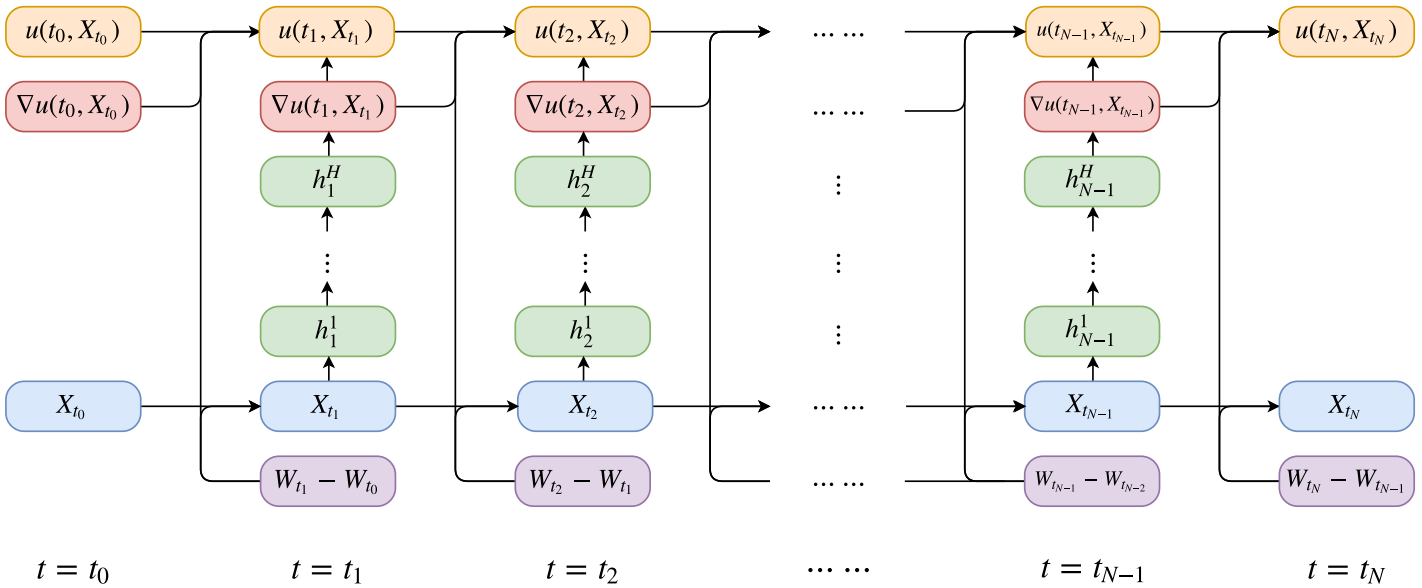

Figure: Network architecture for solving parabolic PDEs. Each column corresponds to a subnetwork at time $t = t_n$. The whole network has $(H + 1)(N - 1)$ layers in total that involve free parameters to be optimized simultaneously.

Optimization

- This network takes the paths $\{X_{t_n}\}_{0\leq n\leq N}$ and $\{W_{t_n}\}_{0\leq n\leq N}$ as the input data and gives the final output, denoted by $\hat{u}(\{X_{t_n}\}_{0\leq n\leq N}, \{W_{t_n}\}_{0\leq n\leq N})$, as an approximation to $u(t_N, X_{t_N})$.

- The error in the matching of given terminal condition defines the expected loss function

$$l(\theta) = \mathbb{E}\Big[\big|g(X_{t_N}) - \hat{u}\big(\{X_{t_n}\}_{0\leq n\leq N}, \{W_{t_n}\}_{0\leq n\leq N}\big)\big|^2\Big].$$

- The paths can be simulated easily. Therefore the commonly used SGD algorithm fits this problem well.

- We call the introduced methodology deep BSDE method since we use the BSDE and DNN as essential tools.

Why such deep networks can be trained?

Intuition: there are skip connections between different subnetworks

$$\begin{aligned} & u(t_{n+1}, X_{t_{n+1}}) - u(t_n, X_{t_n}) \\ \approx & - f\big(t_n, X_{t_n}, u(t_n, X_{t_n}), \phi_n(X_{t_n})\big)\, \Delta t_n + (\phi_n(X_{t_n}))^{\mathrm{T}} \Delta W_n \end{aligned}$$

resemble residual networks (fully connected deep NN are unstable!)

Table of Contents

Implementation

- Each subnetwork has 4 layers, with 1 input layer (d-dimensional), 2 hidden layers (both $d + 10$-dimensional), and 1 output layer (d-dimensional).
- Choose the rectifier function (ReLU) as the activation function and optimize with Adam method.
- The means and the standard deviations of the relative errors are approximated by 5 independent runs of the algorithm with different random seeds.
- Implement in Tensorflow and reported examples are all run on a Macbook Pro.
- Github: https://github.com/frankhan91/DeepBSDE

LQG (linear quadratic Gaussian) Example for d=100

$$dX_t = 2\sqrt{\lambda}\, m_t\, dt + \sqrt{2}\, dW_t,$$

Cost functional: $J(\{m_t\}_{0\le t\le T}) = \mathbb{E}\big[\int_0^T \|m_t\|_2^2\, dt + g(X_T)\big]$.

HJB equation:

$$\frac{\partial u}{\partial t} + \Delta u - \lambda \|\nabla u\|_2^2 = 0$$

$$u(t,x) = -\frac{1}{\lambda}\ln\Big(\mathbb{E}\Big[\exp\Big(-\lambda g(x+\sqrt{2}W_{T-t})\Big)\Big]\Big).$$

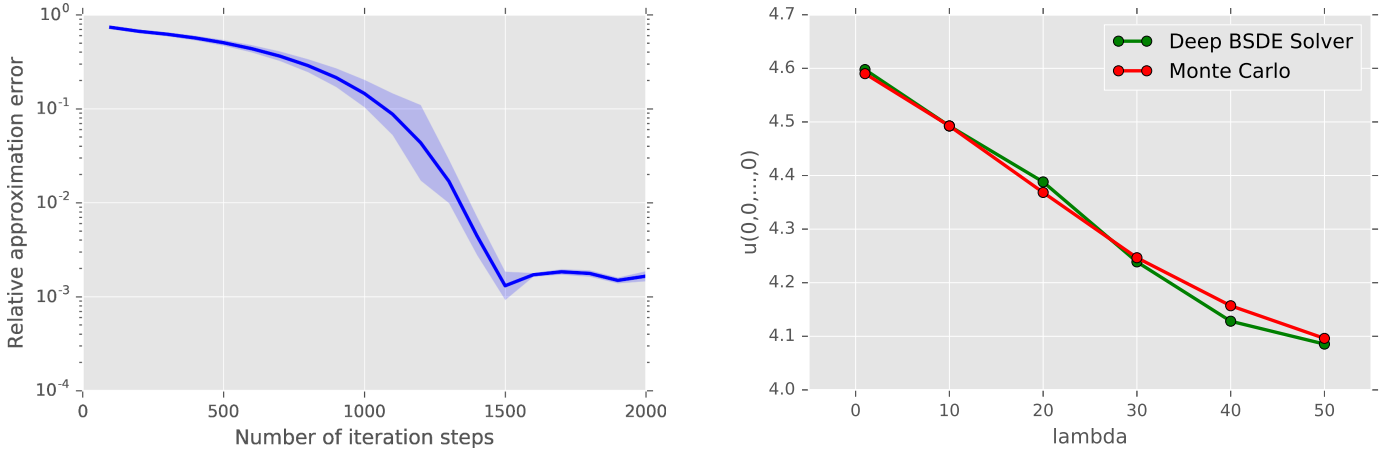

Figure: Left: Relative error of the deep BSDE method for $u(t{=}0, x{=}(0,\dots,0))$ when $\lambda = 1$, which achieves 0.17% in a runtime of 330 seconds. Right: Optimal cost $u(t{=}0, x{=}(0,\dots,0))$ against different λ.

Black-Scholes Equation with Default Risk

- The classical Black-Scholes model can and should be augmented by some important factors in real markets, including defaultable securities, transactions costs, uncertainties in the model parameters, etc.

- Ideally the pricing models should take into account the whole basket of financial derivative underlyings, resulting in high-dimensional nonlinear PDEs.

- To test the deep BSDE method, we study a special case of the recursive valuation model with default risk (Duffie et al. 1996, Bender et al. 2015).

Black-Scholes Equation with Default Risk

- Consider the fair price of a European claim based on 100 underlying assets conditional on no default having occurred yet.

- The underlying asset price moves as a geometric Brownian motion and the possible default is modeled by the first jump time of a Poisson process.

- The claim value is modeled by a parabolic PDE with the nonlinear function

$$\begin{aligned} & f\big(t, x, u(t,x), \sigma^{\mathrm{T}}(t,x) \nabla u(t,x)\big) \\ = & -(1-\delta)\, Q(u(t,x))\, u(t,x) - R\, u(t,x). \end{aligned}$$

Black-Scholes Equation with Default Risk

The unknown "exact" solution at $t = 0$ $x = (100, \ldots, 100)$ is computed by the **multilevel Picard method**.

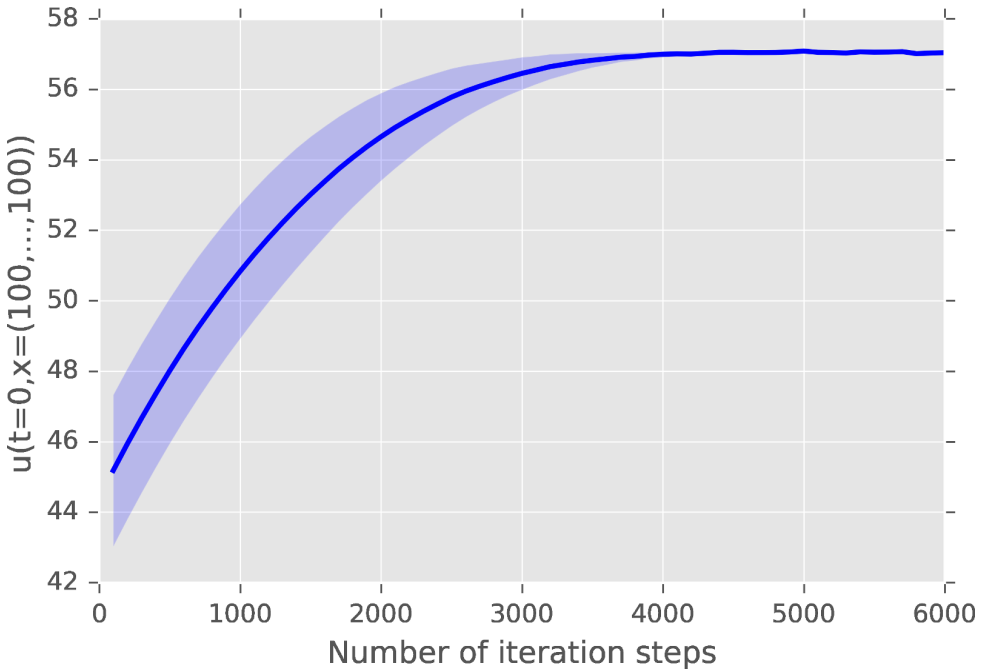

Figure: Approximation of $u(t{=}0, x{=}(100, \ldots, 100))$ against number of iteration steps. The deep BSDE method achieves a relative error of size 0.46% in a runtime of 617 seconds.

Allen-Cahn Equation

The Allen-Cahn equation is a reaction-diffusion equation for the modeling of phase separation and transition in physics. Here we consider a typical Allen-Cahn equation with the "double-well potential" in 100-dimensional space:

$$\frac{\partial u}{\partial t}(t,x) = \Delta u(t,x) + u(t,x) - \left[u(t,x)\right]^3,$$

with initial condition $u(0,x) = g(x)$.

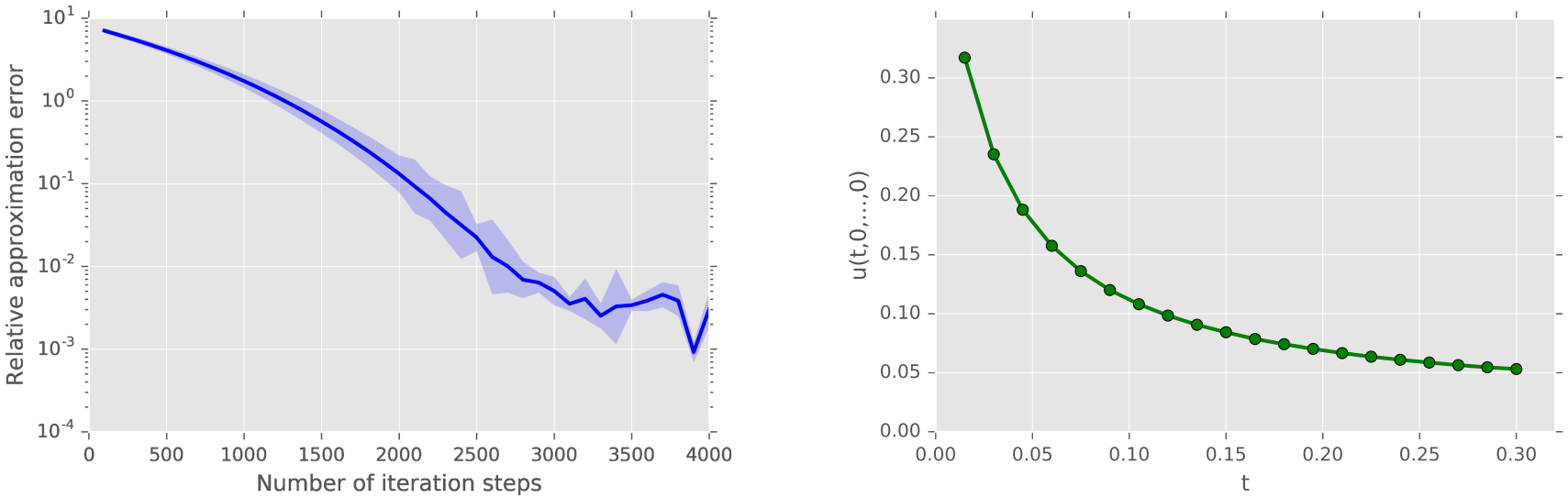

Figure: Left: relative error of the deep BSDE method for $u(t{=}0.3, x{=}(0,\ldots,0))$, which achieves 0.30% in a runtime of 647 seconds. Right: time evolution of $u(t, x{=}(0,\ldots,0))$ for $t \in [0, 0.3]$, computed by means of the deep BSDE method.

An Example with Oscillating Explicit Solution

We consider an example studied for the numerical methods of PDE in literature (Gobet & Turkedjiev 2017). We set $d = 100$ instead of $d = 2$.

The PDE is constructed artificially in a form

$$\frac{\partial u}{\partial t}(t,x) + \frac{1}{2}\Delta u(t,x) + \min\Big\{1, \big(u(t,x) - u^*(t,x)\big)^2\Big\} = 0,$$

in which $u^*(t,x)$ is the explicit oscillating solution

$$u^*(t,x) = \kappa + \sin\big(\lambda \, \Sigma_{i=1}^{d} \, x_i\big) \exp\big(\tfrac{\lambda^2 d(t-T)}{2}\big).$$

Ablation Study

Number of layers†	29	58	87	116	145
Mean of relative error	2.29%	0.90%	0.60%	0.56%	0.53%
Std. of relative error	0.0026	0.0016	0.0017	0.0017	0.0014

Table: The mean and standard deviation (std.) of the relative error for the above PDE, obtained by the deep BSDE method with different number of hidden layers. † We only count the layers that have free parameters to be optimized.

	Nonlinear BS	LQG	Allen-Cahn
ReLU	0.46% (0.0008)	0.17% (0.0004)	0.30% (0.0021)
Tanh	0.44% (0.0006)	0.17% (0.0005)	0.28% (0.0024)
Sigmoid	0.46% (0.0004)	0.19% (0.0008)	0.38% (0.0026)
Softplus	0.45% (0.0007)	0.17% (0.0004)	0.18% (0.0017)

Table: The mean and standard deviation (in parenthesis) of relative error obtained by the deep BSDE method with different activation functions, for the nonlinear Black-Scholes equation, the Hamilton-Jacobi-Bellman equation, and the Allen-Cahn equation.

References and Follow-up Works

- References:
  - Han, Jentzen, and E, Solving high-dimensional partial differential equations using deep learning, *Proceedings of the National Academy of Sciences* (2018)
  - E, Han, and Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, *Communications in Mathematics and Statistics* (2017)
- Follow-up works:
  - Beck et al. 2017 (deep 2BSDE method), Henry-Labordère 2017 (deep primal-dual for BSDEs), Fujii et al. 2017 (deep BSDE with asymptotic expansion), Becker et al. 2018 (deep optimal stopping), Raissi 2018, Beck et al. 2018, Chan-Wai-Nam et al. 2018, Huré et al. 2019

Table of Contents

Formulation of Stochastic Control

Model dynamics:

$$s_{t+1} = s_t + b_t(s_t, a_t) + \xi_{t+1},$$

s_t is state, a_t is control, ξ_t is randomness. Consider objective:

$$\min_{\{a_t\}_{t=0}^{T-1}} \mathbb{E}\Big\{ \sum_{t=0}^{T-1} c_t(s_t, a_t(s_t)) + c_T(s_T) \mid s_0 \Big\},$$

We look for a feedback control:

$$a_t = a_t(s_t).$$

- Neural network approximation:

  $$a_t(s_t) \approx a_t(s_t|\theta_t),$$

  Solve directly the approximate optimization problem

  $$\min_{\{\theta_t\}_{t=0}^{T-1}} \mathbb{E}\Big\{ \sum_{t=0}^{T-1} c_t(s_t, a_t(s_t|\theta_t)) + c_T(s_T) \Big\},$$

  rather than dynamic programming principle.

Network Architecture

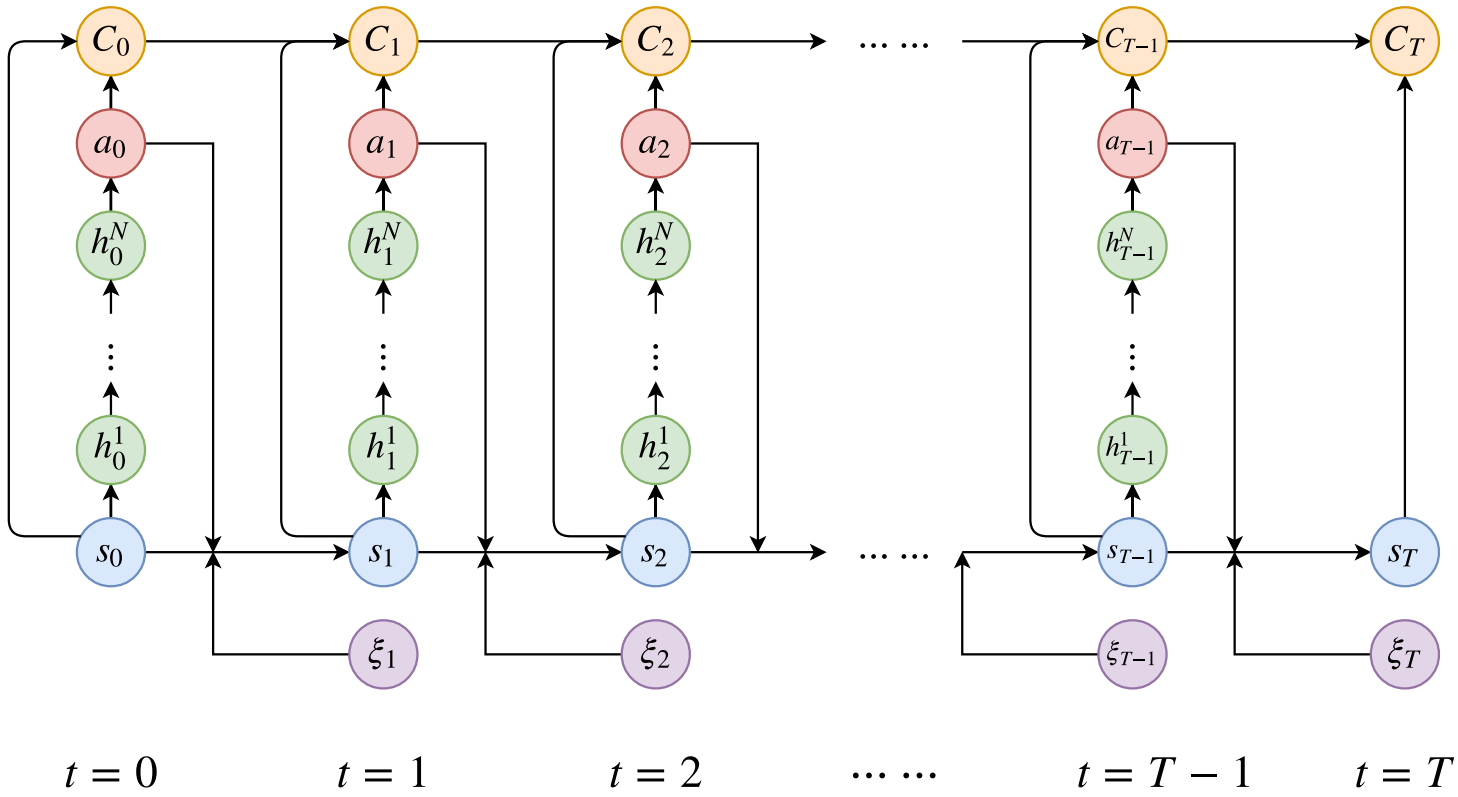

Figure: Network architecture for solving stochastic control in discrete time. The whole network has $(N + 1)T$ layers in total that involve free parameters to be optimized simultaneously. Each column (except ξ_t) corresponds to a sub-network at t.

Example in Optimal Execution of Portfolios

The goal is to minimize the expected cost for trading multiple stocks over a fixed time horizon:

$$\min_{\{a_t\}_{t=0}^{T-1}} \mathbb{E} \sum_{t=0}^{T-1} p_t^{\mathrm{T}} a_t,$$

subject to $\Sigma_{t=0}^{T-1} a_t = \bar{a} \in \mathbb{R}^n$. The execution price is influenced by the amount we buy at each time and the stochastic market conditions.

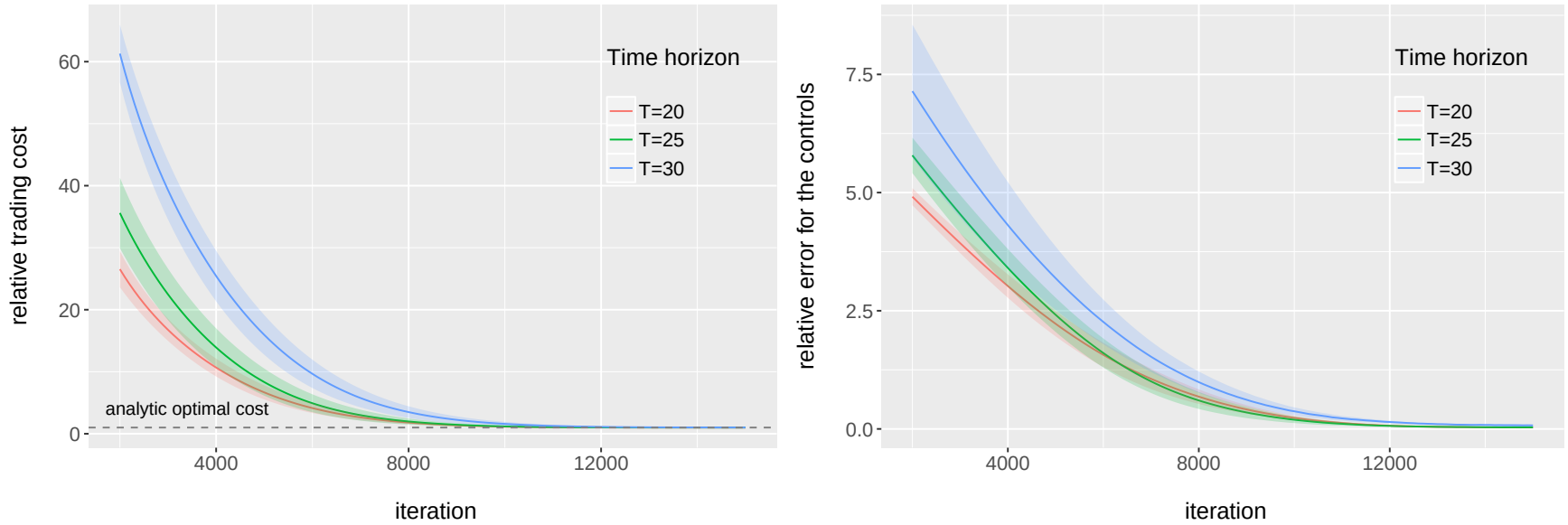

Figure: Learning curves of relative trading cost (left) and relative error for the controls (right). The space of control function is $\mathbb{R}^{23} \to \mathbb{R}^{10}$.

Example in Energy Storage with a Single Device

The goal is to maximize revenues from an energy storage device and a renewable wind energy source while satisfying stochastic electricity demand.

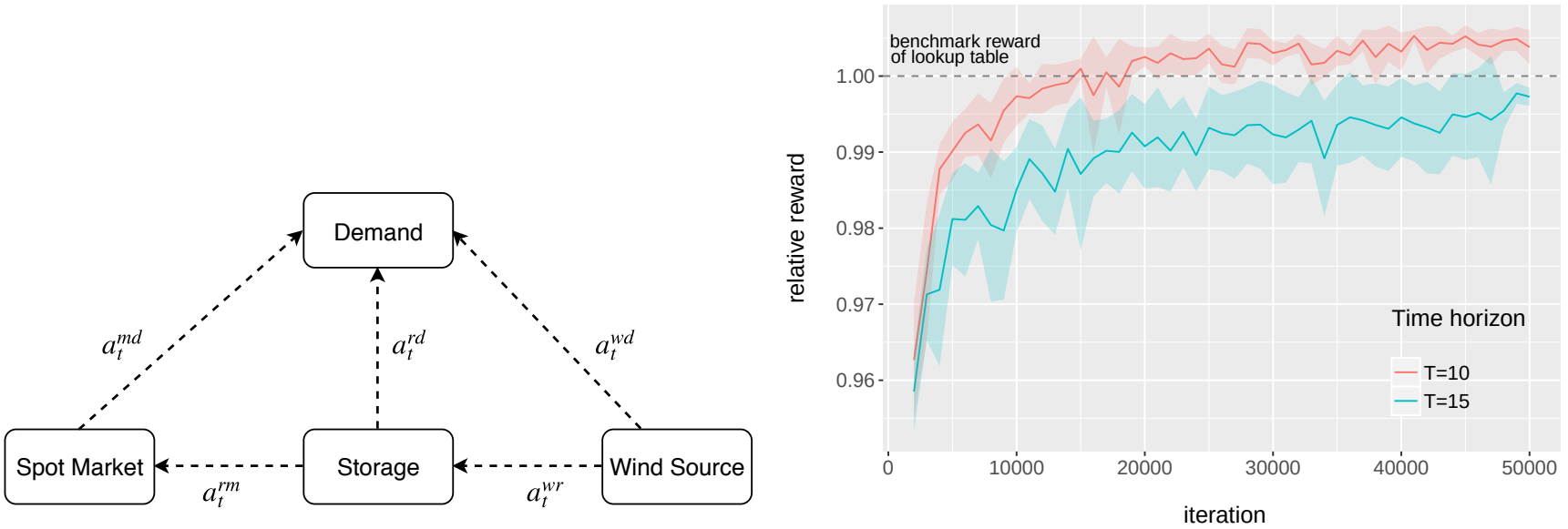

Figure: Left: network diagram of energy; Right: learning curves of relative reward. The space of control function is $\mathbb{R}^4 \to \mathbb{R}^5$ with multiple equality and inequality constrains.

Example in Energy Storage with Multiple Devices

The setting is similar to the above but now there are multiple devices, in which we do not find any other available solution for comparison.

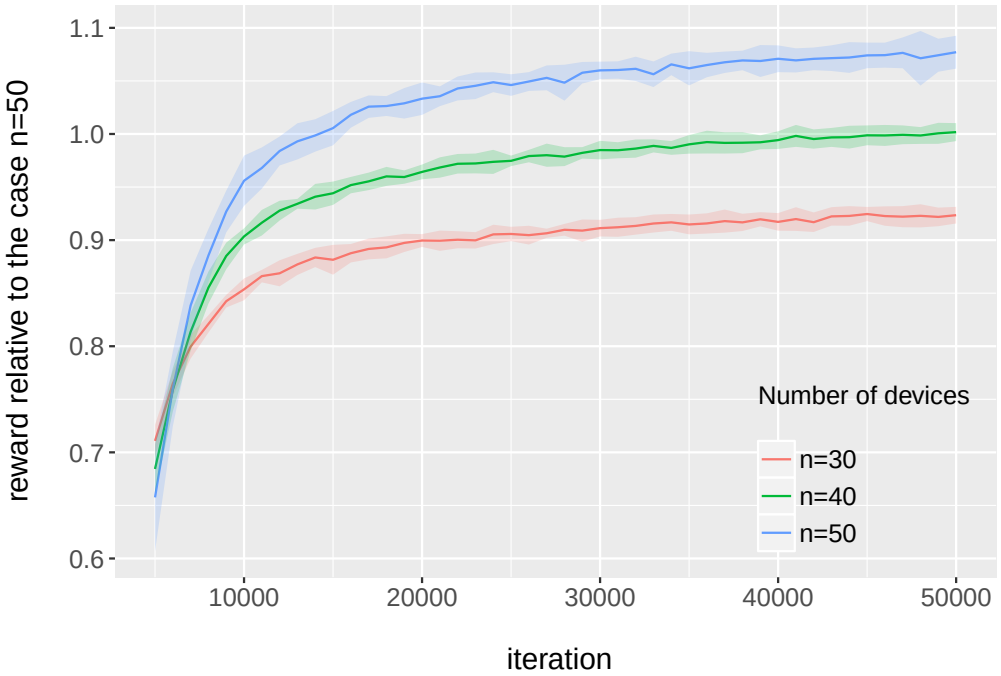

Figure: Relative reward to the case $n = 50$ (with controls satisfying constraints strictly). The space of control function is $\mathbb{R}^{n+2} \to \mathbb{R}^{3n}$ for $n = 30, 40, 50$, with multiple equality and inequality constrains.

Table of Contents

Theorem (A Posteriori Estimates (Han and Long))

Under some assumptions, there exists a constant C, independent of h, d, and m, such that for sufficiently small h,

$$\begin{aligned}&\sup_{t\in[0,T]}(\mathbb{E}|X_t-\hat{X}_t^\pi|^2+\mathbb{E}|Y_t-\hat{Y}_t^\pi|^2)+\int_0^T\mathbb{E}|Z_t-\hat{Z}_t^\pi|^2\,\mathrm{d}t\\ \leq& C[h+\mathbb{E}|g(X_T^\pi)-Y_T^\pi|^2],\end{aligned}$$

where $\hat{X}_t^\pi = X_{t_i}^\pi$, $\hat{Y}_t^\pi = Y_{t_i}^\pi$, $\hat{Z}_t^\pi = Z_{t_i}^\pi$ *for* $t \in [t_i, t_{i+1})$.

Theorem (Upper Bound of Optimal Loss (Han and Long))

Under some assumptions, there exists a constant C, independent of h , d and m, such that for sufficiently small h,

$$\begin{aligned}&\mathbb{E}|g(X_T^\pi) - Y_T^\pi|^2 \\ &\leq C\Big\{h + \mathbb{E}|Y_0 - \mu_0^\pi(\xi)|^2 + \sum_{i=0}^{N-1} \mathbb{E}|\mathbb{E}[\tilde{Z}_{t_i}|X_{t_i}^\pi, Y_{t_i}^\pi] - \phi_i^\pi(X_{t_i}^\pi, Y_{t_i}^\pi)|^2 h\Big\},\end{aligned}$$

where $\tilde{Z}_{t_i} = h^{-1}\mathbb{E}[\int_{t_i}^{t_{i+1}} Z_t \,\mathrm{d}t|\mathcal{F}_{t_i}]$. *If* b *and* σ *are independent of* Y, *the term* $\mathbb{E}[\tilde{Z}_{t_i}|X_{t_i}^\pi, Y_{t_i}^\pi]$ *can be replaced with* $\mathbb{E}[\tilde{Z}_{t_i}|X_{t_i}^\pi]$.

Remark

Similar bounds can be derived for the stochastic control problem (in both continuous time and discrete time)

Table of Contents

Summary

- Deep learning are providing us powerful tools to overcome the curse of dimensionality in high-dimensional parabolic PDEs and control problems.
- For general nonlinear parabolic PDEs, the deep BSDE method reformulate it into a variational problem based on BSDEs and approximate the unknown gradients by neural networks.
- Similar methodology can be applied to solve model based stochastic control problems, in which the optimal policies are approximated by neural networks.
- Numerical results validate the proposed algorithm in high dimensions, in terms of both accuracy and speed.
- This opens up new possibilities in various disciplines, including economics, finance, operational research, and physics.

DL for Other High-Dimensional Problems

- Moment closure for kinetic equations: a machine learning based hydrodynamic model with uniform accuracy[1]
- Solving many-electron Schrödinger equation using deep neural networks[2]
- Large-scale simulation of molecular dynamics: Deep Potential[3] model provides potential energy, forces, or even coarse-grained models based on neural networks with quantum accuracy

Thank you for your attention!

[1] J. Han, C. Ma, Z. Ma, W. E, *PNAS* (2019)

[2] J. Han, L. Zhang, and W. E, *JCP, 399, 108929* (2019)

[3] J. Han, L. Zhang, R. Car, and W. E, *CiCP, 23, 629–639* (2018); L. Zhang, J. Han, H. Wang, R. Car, and W. E, *PRL, 120(10), 143001* (2018)