Deep Backward Policy Gradient: A Model-free Framework for Nonlinear Optimal Control

Anonymous Author(s)
Affiliation
Address
email

Abstract: In this paper, we propose a learning-based method for searching the optimal feedback control policy of a class of nonlinear systems. This method is implemented in a model-free framework, thus is more applicable in real control problems compared with the naive deep forward-backward stochastic optimal controller, which is model-based due to the explicit computation using model dynamics. Specifically, by introducing some exploration noise, the original control problem is reformulated to solving decoupled forward-backward stochastic differential equations whose forward solution can be estimated with samples. In addition, the optimal control policy is represented with a sequence to sequence model in deep learning, and a new loss function derived from dynamic programming is utilized to perform policy improvement purely with samples.

Keywords: optimal control, forward-backward stochastic differential equations, nonlinear systems, dynamic programming

1 Introduction

Optimal control (OC) and Stochastic Optimal Control (SOC) problems widely exist in real-world applications. It is well known that the dynamic programming method can represent the optimal feedback control with the solution of Hamiltonian-Jacobi-Bellman (HJB) equation. In practical, the HJB equation is a second order nonlinear partial differential equation (PDE), which is difficult to solve either analytically and numerically. With the rapid development of deep learning, there are more and more works dedicated to developing a method that can numerically solve HJB equation while overcoming the curse of dimensionality encountered in traditional methods.

The Deep Galerkin Method (DGM) and Deep Backward Stochastic Differential Equation (Deep BSDE) method are two parallel numerical frameworks for solving general PDEs (not limited to HJB equation) and have attracted great attention in applied mathematics, physics and finance [1], [2]. Some works adopt these techniques and combine some special characteristics of HJB equation inherited from optimal control problems [3], [4], [5]. While these methods do successfully overcome the curse of dimensionality, they are all model-based and require analytically description of underlying dynamics. Recently, reinforcement learning (RL) algorithms become popular in applications because of their model-free feature. Although RL treats optimal control problems in discrete-time, one of their starting point is solving Bellman equation derived from dynamic programming principle as well [6]. Indeed, as discussed later, some RL algorithms can be regarded as a model-free extension of DGM by replacing integral operator with Monte Carlo approximator.

Our method embeds the Deep BSDE in a model-free framework on the other hand. By further exploiting characteristics of optimal control problems, the proposed method possess following application-friendly features besides the model-free benefit: 1. robust to control noise. 2. no need to estimate value function on the entire domain. 3. natural for parallelizing.
2 A common pattern in OC and RL

Despite the different description of the underlying dynamics, the ultimate goal in optimal control or reinforcement learning is minimizing or maximizing some feedback signals along system trajectories. Thereby the dynamic programming principle is a common pattern to check policy optimality for these problems. This leads people in OC to seek a suitable PDE solver for HJB equation.

In applied mathematics, DGM and Deep BSDE are such methods that can treat the nonlinear and high-dimensional properties of HJB equation. DGM regards solving a PDE as a variational optimization problem, where the optimized objective function is composed of the PDE itself and boundary conditions. By parameterizing a trial solution of the PDE with a neural network, some gradient optimization method is used to updating the neural network so that the deviation from the differential equation and boundary condition is reduced. Deep BSDE differs from DGM in their formulation of the variational optimization problem. By applying the Feynman-Kac lemma, Deep BSDE obtains forward-backward stochastic differential equations (FBSDEs), which is further reformulated to a minimizing deviation from terminal condition subjected to that FBSDEs. One drawback of DGM is the high computational complexity of loss function, which often includes a high order differential operator in the PDE to be solved. The loss function used by Deep BSDE involves only simple function evaluation, thereby Deep BSDE can iterate faster. However, because of the Feynman-Kac formula used for deriving FBSDEs, Deep BSDE method can only handle a certain type of PDEs compared with DGM. To some extent, [5] and [3] can be viewed as the corresponding advanced works of DGM and Deep BSDE in optimal control problems.

On the other hand, it is worth noting that a model-free extension of DGM has been around for a long time in RL, which is actually the Soft Actor-Critic (SAC) framework [7]. SAC rewrites the Bellman equation as two first order integral equations and represents solutions with two separately networks. Same as DGM, SAC uses the deviation of these two integral equation to update trial solutions. The key to achieve model-free is to replace the integral operation with a Monte Carlo approximation, which can be evaluated merely with samples. However, there are characteristics in common optimal control problems that closely relate the solutions of these integral equations. Our method embeds the Deep BSDE in a model-free framework by further exploiting the characteristics of optimal control and thereby manage to maintain a single policy network and avoid unpredictable phenomena arisen in updating two separate networks.

3 Theory Walk-through

In this section, a brief theory walk-through of DBPG is presented. Due to page limits, only the most essential treatments are mentioned here and readers are encouraged to consult existed works for details (e.g., [2], [3], [8]). We directly starts by the following optimal control problem with the control term disturbed by a Gaussian noise,

\[
\min_{u(t)} J(u(t)) := \mathbb{E} \left[ \int_0^T Q(t, x(t)) + \frac{1}{2} u(t)^T R u(t) \, dt + \phi(x(T)) \right], \tag{1}
\]

s. t. \quad \dot{x}(t) = F(t, x(t)) + G(t, x(t))[u(t) + u_\epsilon], \quad x(0) = x_0, \tag{2}

where \( u_\epsilon dt := \Sigma(t, x(t)) dw \) is Gaussian noise with mean zero and covariance matrix \( \Sigma \Sigma^T \), and \( w(t) \) is a standard \( n \)-dimensional Brownian motion. We should emphasize that this control noise is not necessarily the intrinsic system noise but also can be enforced manually during searching an acceptable policy. For simplicity, \( \Sigma(t, x) \) is chosen such that \( \Sigma^{-1} \) is always well defined. According to dynamic programming theory, the HJB equation associated with this SOC problem is (let \( \tilde{\Sigma} := G \Sigma \))

\[
\dot{v}_t + \frac{1}{2} \text{tr} \left( v_x \Sigma \Sigma^T \right) + Q + v_x^T (F + G \bar{u}) - \frac{1}{2} v_x^T \tilde{\Sigma} (\Sigma^T R \Sigma)^{-1} \Sigma^T v_x - v_x^T \Sigma \Sigma^{-1} \bar{u} = 0, \tag{3}
\]

along with a terminal condition \( v(T, x) = \phi(x) \). The optimal feedback control is obtained by taking the gradient of value function \( u^* = -R^{-1}G^T v_x \) [9]. This connection between the optimal control \( u^* \) and the solution of the HJB equation \( v(t, x) \) relates the SOC problem to a Cauchy problem (3).
Notice that \( v^T \Sigma \hat{u} - v^T \Sigma \Sigma^{-1} \hat{u} \equiv 0 \), which means \( \hat{u}(t, x) \) can be freely chosen as any appropriate function. Apply Feynman-Kac formula to Eq. (3) to derive the associated FBSDEs

\[
dX_t = \left[ F + G\hat{u} \right] dt + \tilde{\Sigma} dw, \quad X_s = \xi \tag{4}
\]

\[
dY_t = \left[ -Q(t, X_t) + \frac{1}{2} Z_t^T (\Sigma^T R \Sigma)^{-1} Z_t + Z_t^T \Sigma^{-1} \hat{u} \right] dt + Z_t^T dw, \quad Y_T = \phi(X_T), \tag{5}
\]

where \( v(t, X_t) = Y(t; \xi) \), \( \bar{S}^T \Sigma v_x(t, X_t) = Z(t; \xi) \) hold for any \((\xi, \hat{z}) \in [0, T] \times \mathbb{R}^n\). The optimal control can be represented as well \( \hat{u}^* (t, X_t) = -R^{-1}(Z_t \Sigma^{-1})^T \). Therefore, once the solution \( (X_t, Y_t, \Sigma_t) \) of (4), (5) is obtained, the SOC of (1), (2) is obtained directly. Define \( U_t : = -R^{-1}(Z_t \Sigma^{-1})^T \) and recall that \( \Sigma = G \Sigma_t \), the above FBSDEs can be rewritten as

\[
dX_t = \left[ F + G(\hat{u} + u_a) \right] dt, \quad X_s = \xi, \tag{6}
\]

\[
dY_t = \left[ -Q(t, X_t) + \frac{1}{2} U_t^T R U_t + U_t^T R (u_a - u_i) \right] dt, \quad Y_T = \phi(X_T), \tag{7}
\]

where \( v(t, X_t) = Y(t; \xi) \), \( u^*(t, X_t) = U(t; \xi) \) hold as well. Eq. (6) means that if the system starts at state \( \xi \) and state \( \xi \), then the state trajectory sampled from that system is in fact a realization of (4) as long as the control is set to \( \hat{u} = u_a \), which is termed as exploration policy in RL context.

The BSDE (7) is actually a necessary condition of optimal policy \( u^* \) derived from HJB equation, which inspires us to throw the following constrained variational optimization problem

\[
\min_{y_0, \{U_t\}_{t \leq T}} \mathbb{E} \left[ y_0 + \int_0^T dY_t - \phi(X_T) \right], \tag{8}
\]

where \( \{X_t\}_{t \leq T}, \{Y_t\}_{t \leq T} \) are subjected to FBSDEs (6), (7) (let \( \xi = x_0 \)). Under certain assumptions, it can be proved that \( y_0 = v(0, x_0), U_t = u^*(t, X_t) \) is a minimizer of (8) [8]. At this point, we successfully transformed the original optimal control problem to a constrained variational optimization problem whose minimizer is the optimal feedback control policy.

### 4 Deep Backward Policy Gradient

DBPG takes the idea of policy iteration, which starts by a initial guess and then improved it repeatedly. To be specific, the first step is representing the solution of BSDE (7) \( U_t \) with a suitable network. Considering the adaptiveness requirement, a RNN based sequence to sequence (seq2seq) architecture is introduced here. The main advantage of using a seq2seq model is that it can accept variable length sequences as inputs and be evaluated not necessarily under the exact same settings in training, which gives the flexibility to deploy it on various situations. The second step is determining a rule for updating networks. The objective function (8) is already a criterion for it except some minor modifications to be done. Suppose \( u_0 \) is a network fitting the solution \( U_t \), and there are \( M \) trajectories \( \{X^{(i)}\}_{1 \leq i \leq M}, \{u^{(i)}_a\}_{1 \leq i \leq M} \) sampled from the environment that starts at \( s = 0 \) and \( \xi = x_0 \), then the loss of this batch of data is written as follows:

\[
\ell(\theta) : = \mathbb{E}_t \left[ \phi(X^{(i)}_t) - \hat{y}_0 - \int_0^T dY^{(i)}_t \right], \quad \hat{y}_0 : = \mathbb{E}_i \left[ \hat{y}^{(i)}_0 \right] : = \mathbb{E}_i \left[ \phi(X^{(i)}_T) - \int_0^T dY^{(i)}_t \right], \tag{9}
\]

where \( dY \) equals the right hand of (7) and the integral is discretized with the sampling frequency.

There is a trick that we avoid optimizing a standalone variable \( \hat{y}_0 \) in (8) and replace it with a Monte Carlo estimation \( \mathbb{E}_i \left[ \hat{y}^{(i)}_0 \right] \). This replacement does not change the fact that the optimal policy minimizing the loss function (9). This trick also provide a convenient way to compute loss by the variance \( \mathbb{E}_i \left[ \left[ \hat{y}_0 - \hat{y}^{(i)}_0 \right]^2 \right] \). The final step is to choose a appropriate exploration policy \( u_a \). In principle, it can be any Gaussian policy providing the covariance matrix is invertible. To keep it simple, we assign \( \hat{u} \) with the last iteration result and choose \( \Sigma = \sigma I \) (\( \sigma > 0 \)). We summarize the training procedure and network architecture in Figure 1.

Numerical experiments on three nonlinear systems suggest that the proposed method convergent quickly and is robust to control noise. See Appendix for details.
5 Conclusions

In this paper, we propose a model-free framework for nonlinear optimal control by studying its probabilistic representation. We also point out the popular soft actor-critic framework can be regarded as a natural extension of DGM in reinforcement learning, while ours is an extension of Deep BSDE in optimal control. Those methods are all related by the HJB equation derived from dynamic programming principle. We hope this work could share some insights of the relationship between applied mathematics, optimal control and reinforcement learning.

References

Appendix

We test DBPG on a classical problem of controlling an inverted pendulum. To illustrate the generalization of the proposed method, we evaluate on three pendulum systems in 2-, 4- and 6-dimensional cases and keep most hyperparameters consistent. The 2D and 4D inverted pendulum systems are known as Pendulum and CartPole and 6D system is chosen as two-wheeled inverted pendulum (TWIP). The general purpose is to swing the pendulum up and stabilize it. We set initial angle is pointed down and the target angle is pointed up. The standard deviation of exploration noise is set to \( \sqrt{2\alpha/\omega} \), where \( \omega = 0.01 \text{ s}^{-1} \) is the sampling frequency and \( \alpha \) is 0.33 for TWIP and 1.0 for the other two systems. Some quantities in cost functional is chosen as follows

\[
Q(t, x) = (x - x^*)^T \text{diag}\{q_1, q_2, \ldots, q_n\}(x - x^*), \quad \phi \equiv 0, \quad (10)
\]

where \( q_i \) respected to the controlled angle is set to 1.01. Figure. 2 shows the controlled angle and control output on three systems. Cost along these trajectories are also plotted in Figure 2. It can be seen that even for the 6D dimensional system, the cost functional decreases to a acceptable level within 1000 gradient steps (where batch size is equal to 64). All these networks are trained with time horizon \( T = 2.0 \text{ s} \) but evaluated under time interval \([0, 3.0 \text{ s}]\). Results in Figure. 2 suggest the network is trained to be able to generalize not only on states but also in time space. Figure. 3 shows the generalized policy is particular robust to control noise and can recover almost same trajectory with considerably large control fluctuations.

![Figure 2](image1.png)

**Figure 2**: controlled angle (a), control outputs (b) and cost (c). For TWIP system, the control is 2-dimensional but only one component is plotted here for clarity. And vertical axes units are removed in (b), (c) because their absolute values are meaningless on different systems.

![Figure 3](image2.png)

**Figure 3**: The trajectories when noises are added to control outputs. There are 10 lines in each subplots (though some are almost completely coincide).