Deep Galerkin Method for Mean Field Control Problem

Jingruo Sun [*] William Hofgard [*] Asaf Cohen [**]

[*]Department of Management Science & Engineering, Stanford University [**]Department of Mathematics, University of Michigan

Introduction

The N-agent optimal control problem considers a cooperative game of N weakly interacting players over a finite horizon. Each player aims to minimize its average running cost by selecting the transition rates at each step. Their actions influence the empirical distribution of states and thus the costs of others.

However, as N grows large, the problem becomes technically intractable. Hence, we approximate the N-agent optimal control problem by a Mean Field Control Problem (MFCP). We consider a continuum of players traversing among multiple states in continuous time and replace the empirical distribution by a flow of measures. Therefore, we define it as an optimization problem over a McKean-Vlasov process.

We solve MFCP numerically by an approximation method inspired from deep learning. Our experiments demonstrate the evolution of the cost function and the trend of its training loss. We prove that such numerical solution converges to the true solution as the number of samples goes to infinity.

N-agent Optimal Control Problem

The N-agent optimal control problem involves a decision process with multiple interacting agents. Each agent is associated with an individual cost function and a strategy set. The players are targeting at minimizing the average cost.

We formulate the process (X_t) with state space $[\![d]\!]$ on a simplex S_d. Denote μ_t^N as the empirical distribution of the N agents at time t.

- **Transition dynamics:**

$$\mathbb{P}(X_{t+h}^k = j \mid \mathbf{X}_t = \mathbf{x}) = \beta^k(t,\mathbf{x};j)h + o(h) \quad \text{as } h \to 0^+$$

- **Common costs:**

$$J^N(m_0^N, \beta) = \frac{1}{N}\sum_{k=1}^{N} \mathbb{E}\left[\int_0^T f(t, X_t^k, \beta^k(t, \mathbf{X}_t; \cdot), \mu_t^N)dt + g(X_T^k, \mu_T^N)\right]$$

We choose feedback controls $\beta = (\beta^1, \ldots, \beta^N)$ to minimize the common cost, where f is the running cost of the stochastic problem and g is the terminal cost. We derive the HJB equation for N-agent control problem as a first-order, non-linear PDE on the $(d-1)$-dimensional simplex.

However, as d increases, the "Curse of Dimensionality" prevents standard numerical schemes, such as Monte Carlo methods, mesh-based algorithms, etc, from solving the HJB equation in a tractable manner.

Mean Field Control Problem

We formulate the mean field control problem to approximate the collective behavior in the N-agent optimal control problem. The strategy of each player is only affected by the average density of other players but not by a particular stochastic configuration of the system. Therefore, we reduce the computational complexity of the problem significantly.

We formulate the process $(X_t)_{t\in[0,T]}$ as

$$\mathcal{A}^d = [0,M]^d \text{ be the action space, } \mu_t = \mathrm{Law}(X_t) \text{ be the distribution,}$$

$$S_d = \left\{(\mu_1, \ldots, \mu_d) \in \mathbb{R}^d : \mu_i \geq 0, \sum_{i=1}^{d} \mu_i = 1\right\} \text{ be the simplex,}$$

$$\alpha : [0,T] \to \mathcal{A}^d \text{ be the control as a deterministic measurable function}$$

Therefore, we derive:

1. **Transition:** follows the McKean-Vlasov dynamics

$$\mathbb{P}(X_{t+h} = j \mid X_t = i, \mu_t = m) = \alpha_j(t)h + o(h) \quad \text{as } h \to 0^+$$

2. **Cost:**

$$J(\alpha) = \mathbb{E}\left[\int_0^T f(t, X_t, \alpha(t, X_t), \mathrm{Law}(X_t))dt + g(X_T, \mathrm{Law}(X_T))\right]$$

3. **Value function:** the optimal cost converges to V^N as:

$$V = \inf_{\alpha\in\mathcal{A}^d} J(\alpha), \quad V \to V^N \in \mathcal{O}(1/\sqrt{N})$$

4. **Hamiltonian:** $H(t,m,z) = \sum_{i\in[\![d]\!]} m_i H^i(t,m,z)$, with

$$H^i(t,m,z) = \sup_{a\in[0,M],\, j\in[\![d]\!]\setminus\{i\}} \Big(-\sum_{j\in[\![d]\!]\setminus\{i\}} \alpha_j(t) z_j - f(t,i,a,m)\Big)$$

5. **HJB equation:** we only explore the derivative along the simplex in the directions $D^i = (\partial_{e_j - e_i})_{j\in[\![d]\!]}$:

$$-\partial_t V(t,m) + \sum_{i\in[\![d]\!]} m_i H^i(t,m,D^i V(t,m)) = 0,$$

$$V(T,m) = \sum_{i\in[\![d]\!]} m_i g^i(m)$$

We assume the Lipschitz continuity for $\alpha, \nabla_\alpha f$, and the uniform convexity for f, g. Therefore, we characterize the value function V as the unique viscosity solution of the HJB equation on the simplex.

Deep Galerkin Method

Deep Galerkin Method (DGM) is a mesh-free method merging Galerkin methods and machine learning. DGM approximates the solution with a deep neural network which is trained to satisfy the differential operator, initial condition, and boundary conditions at randomly sampled spatial points. In this way, we convert the PDE problem into a machine learning problem.

1. **Neural network approximators:** parameterized with θ: with total layers L, activation function σ , weight W_i, bias vectors c_i, and scalar weight α.

$$\varphi(t,m;\theta) := \sigma(W_L \ldots \sigma(W_1 m + \alpha t + c_1) \ldots + c_L)$$

2. **Loss function:** We generate samples $s_i = \{(t_i, m_i), (\tau_i, z_i)\}$ uniformly, where (t_i, m_i) are from $[0,T] \times S_d$ and (τ_i, z_i) are from $\{T\} \times S_d$.

$$L(s_i,\theta) = \underbrace{\left\| -\partial_t\varphi(t_i,m_i;\theta) + \sum_{i\in[\![d]\!]} m_i H^i(t_i,m_i,D^i\varphi)\right\|^2}_{\text{differential operator}} + \underbrace{\left\|\varphi(T,z_i;\theta) - \sum_{i\in[\![d]\!]} z_i g^i(z_i)\right\|^2}_{\text{terminal condition}}$$

3. **DGM algorithm:**

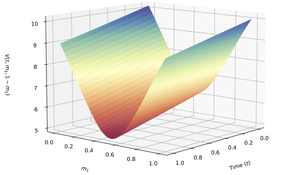
Initialize: parameters $\theta^{(0)}$, tolerance $\delta \in (0,1)$, and sample size B
Initialize: $i = 0$ and $G(\theta^{(0)}) = \infty$
while $G(\theta^{(i)}) \geq \delta$ **do**
  Sample a batch $s_i^{(j)} = \{(t_i^{(j)}, m_i^{(j)}), (\tau_i^{(j)}, z_i^{(j)})\}_{j=1}^B$
  $G(\theta^{(i)}) = \frac{1}{B}\sum_{j=1}^B L(s_i^{(j)}, \theta^{(i)})$
  $\theta^{(i+1)} \leftarrow \theta^{(i)} - \alpha^{(i)} \nabla_\theta G(\theta^{(i)})$
  $i \leftarrow i + 1$
end while

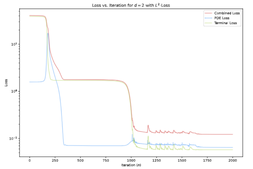
Numerical Results

We consider a quadratic running cost and a linear terminal condition. We apply SGD optimizer to train the DGM network for 200 epochs with sample size 10000. We observe a roughly linear increase in runtime with $d > 50$.

d	Training Time (s)	Total Loss	Running Loss	Terminal Loss
2	47.5	1.2134	0.5452	0.6682
10	62	0.7200	0.1190	0.6010
50	76	0.0588	0.0093	0.0494
100	100	0.0217	0.0031	0.0186
200	221	0.0070	0.0011	0.0059

We also illustrate the evolution of value function and training loss for dimension $d = 2$. We observe the global minimum is achieved at $(0.5, 0.5)$ and the loss continues decreasing along the training time.





Thus, we conclude that our DGM algorithm is able to fully learn the MFCP and achieves to generate a robust numerical solution for the PDE.

Theory Results

We establish a theoretical scheme for DGM by proving, as iterations $\to \infty$:

1. The training loss of DGM converges to 0

For every $\varepsilon > 0$, there exists a constant $K > 0$, which depends on the Lipschitz constant of Hamiltonian, such that for all φ with bounded σ, the DGM loss function satisfies $L(\theta) \leq K\varepsilon$.

2. The numerical solution of DGM converges to the true value function

The neural network approximators $\varphi(\theta)$ converges uniformly to the unique classical solution of HJB equation V as

$$\sup_{(t,m)\in[0,T]\times S_d} |\varphi(t,m;\theta) - V(t,m)| \to 0 \quad \text{as } n \to \infty$$

Conclusion

We establish a novel deep learning method for solving high-dimensional HJB equations on the simplex arising from MFCP. Our DGM algorithm achieves a well-performed approximation to the optimal solution. We provide a theoretical scheme to support its convergence to the true value.

For future work, we want to explore the Backward Stochastic Differential Equations (BSDE), which is another deep learning method for solving high-dimensional PDE, for its application on MFCP. We aim to relax the regularity assumptions for the HJB equation and investigate the convergence.

Email: jingruo@stanford.edu