Solving a Type of HJB Equation by the Reinforcement Learning approach

Jiamin Jian Jungang Bu Yiyang Che

Abstract

In this report, firstly we introduce the HJB equation and give a benchmark problem. To solve this problem, we apply the UFD scheme and the CFD scheme to get the Markov decision process and then use the reinforcement approach such as value iteration. We find that our method is effective for low-dimensional HJB equation, and we can get a accurate numerical solution for the benchmark problem. Compared with the UFD scheme, we find that the CFD scheme is more accurate.

1 Introduction

Stochastic control theory has a long history and it has been widely used in various fields. Generally we introduce the corresponding Hamilton-Jacobi-Bellman equations when we want to solve the stochastic control problem. As the regularity of the system is often not known a prior due to the control process, and since the solutions of the HJB equations are usually highly nonlinear, closed-form solution are rarely obtainable. Thus sometimes it is more convenient and efficient to consider the numerical solution of HJB equation. Numerical methods such as finite element method, finite difference method and spectral method are often used to get the numerical solution of partial differential equations.

Nowadays, with the strengthening of computer's computing ability and the continuous development of machine learning and especially deep learning, we have an effective tool to solve some problems. Artificial neural networks can sometimes get unexpected great results in solving high dimensional problems. And there are increasing interests of using the reinforcement learning and deep learning technologies to solve the PDE problems. In this project we want to apply the reinforcement learning and deep learning approach based on the Markov chain approximation methods to solve a type a HJB equations. It involves two steps. Firstly, we approximate the solutions of the underlying stochastic control problems and recast them into discrete time Markov decision

processes. Then, we solve the MDPs by using reinforcement learning and deep learning approaches.

1.1 Hamilton-Jacobi-Bellman Equation

In optimal control theory, the Hamilton–Jacobi–Bellman (HJB) equation gives a necessary and sufficient condition for optimality of a control with respect to a loss function. It is, in general, a nonlinear partial differential equation in the value function, which means its solution is the value function itself. Once the solution is known, it can be used to obtain the optimal control by taking the maximizer or minimizer of the Hamiltonian involved in the HJB equation.

Generally, for an elliptic nonlinear HJB equation with the d-dimensional Euclidean space $\mathbb{R}^d$ and the domain $\mathcal{O}$, we denote

$$F(x, Du(x), D^2u(x)) = 0, \forall x \in \mathcal{O}$$

with its boundary condition

$$u(x) = g(x), \forall x \in \mathcal{O}^c,$$

where $F : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^{d\times d} \mapsto \mathbb{R}$ is a given continuous function of the type

$$F(x, p, q) = \otimes_a\{b(x, a) \cdot p + \frac{1}{2}\mathbf{tr}(\sigma(x, a)\sigma^{'}(x, a)q) + \ell(x, a)\},$$

and $\mathbf{tr}(\cdot)$ is the trace of a matrix. In the above, the $\otimes$ is an operator as defined in [1] that summarizes values over actions as a function of the state satisfying the following three conditions:

- $\otimes_a[c\phi_1(x, a) + \phi_2(x)] = c \otimes_a [\phi_1(x, a)] + \phi_2(x)$
- $\otimes_a\phi_1(x, a) \leq \otimes_a\phi_2(x, a)$, whenever $\phi_1 \leq \phi_2$
- $|\otimes_a \phi_1(x, a) - \otimes_a\phi_2(x, a)| \leq K \max_a |\phi_1(x, a) - \phi_2(x, a)|$,

where ϕ_1, ϕ_2 are real valued functions and c, K are constant. Many frequently used operators satisfy the aforementioned conditions. For instance, $\max_a \phi(x, a)$ and $\min_a \phi(x, a)$ are related to standard HJB equations and stochastic control problems.

1.2 Markov decision process

A Markov decision process (MDP) is a discrete time stochastic control process. It provides a mathematical framework for modeling decision making in situations where

outcomes are partly random and partly under the control of a decision maker. MDPs are useful for studying optimization problems solved via dynamic programming and reinforcement learning. A Markov decision process is a 4-tuple (S, A, P_a, R_a), where

- S is a finite set of states,
- A is a finite set of actions (alternatively, A_s is the finite set of actions available from state s),
- $P_a(s, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability that action a in state s at time t will lead to state s' at time $t + 1$,
- $R_a(s, s')$ is the immediate reward (or expected immediate reward) received after transitioning from state s to state s', due to action a.

For example, the Figure 1 give a MDP with three states (green circles), two actions (orange circles), and two rewards(orange arrows):

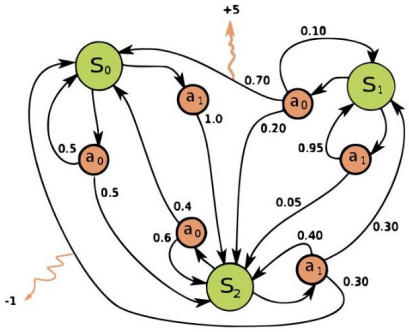

Figure 1: Example of a simple MDP

The core problem of MDPs is to find an optimal policy for the decision maker: a function π that specifies the action $\pi(s)$ that the decision maker will choose when in state s. Once a Markov decision process is combined with a policy in this way, this fixes the action for each state and the resulting combination behaves like a Markov chain, since the action chosen in state s is completely determined by $\pi(s)$ and $\Pr(s_{t+1} = s' | s_t = s, a_t = a)$.

The goal is to choose a policy π that will maximize some cumulative function of the random rewards, typically the expected discounted sum over a potentially infinite horizon:

$$\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1})]$$

where we choose $a_t = \pi(s_t)$, i.e. actions given by the policy and γ is the discount factor satisfying $0 \leq \gamma \leq 1$. And the expectation is taken over

$$s_{t+1} \sim P_{a_t}(s_t, s_{t+1}).$$

1.3 Reinforcement learning

Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize the notion of cumulative reward. Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning. The environment is typically stated in the form of a Markov decision process (MDP), because many reinforcement learning algorithms for this context utilize dynamic programming techniques. The main difference between the classical dynamic programming methods and reinforcement learning algorithms is that the latter do not assume knowledge of an exact mathematical model of the MDP and they target large MDPs where exact methods become infeasible.

2 Problem Setup

In this section we consider a type of HJB equation and give a benchmark problem for the demonstration of the computational method.

2.1 HJB equation

Let's consider a type of d-dimension HJB as follows.

- Domain:
$$\mathcal{O} = \{x \in \mathbb{R}^d : 0 < x_i < 1, i = 1, 2, \ldots, d\}.$$
- Equation on $\mathcal{O}$:
$$(\frac{1}{2}\Delta - \lambda)u(x) + \inf_a(\sum_{i=1}^{d} b_i(x, a)\frac{\partial u(x)}{\partial x_i} + \ell(x, a)) = 0. \tag{1}$$
- Dirichlet data on the boundary $\partial\mathcal{O}$:
$$u(x) = g(x).$$

And all of the deductions in our report are based on this type of HJB equation. We will give the deduction of how to get the Markov decision process by the central finite scheme and the upwind finite scheme.

2.2 Benchmark problem

To illustrate the effect of our numerical method, we consider a benchmark problem, on which we can compare the numerical solutions of differential equations with the exact ones. We consider the following d-dimensional HJB equation:

- Domain:
$$\mathcal{O} = \{x \in \mathbb{R}^d : 0 < x_i < 1, i = 1, 2, \ldots, d\}.$$
- Equation on $\mathcal{O}$:
$$\frac{1}{2}\Delta u + \inf_{|a|\leq 3}(a \cdot \nabla u + d + 2|x|^2 + \frac{1}{2}|a|^2) = 0, . \tag{2}$$
- Dirichlet data on the boundary $\partial\mathcal{O}$:
$$u(x) = -|x|^2.$$

Comparing the formula (2) with the general form of HJB equation (1), we know that the parameters in the benchmark problem is: $\lambda = 0, b_i(x, a) = a$ and $\ell(x, a) = d + 2|x|^2 + \frac{1}{2}|a|^2$.

2.2.1 Exact solution of the benchmark problem

To get the exact solution to the above problem, firstly we take the derivative of $a \cdot \nabla u + \ell(x, a)$ with respect to a, we can get that

$$\frac{\partial u}{\partial x} + a = 0,$$

so we have $a = -\frac{\partial u}{\partial x}$. Then substitute it into the equation on $\mathcal{O}$, we have

$$\begin{aligned} \frac{1}{2}\Delta u + (-\frac{\partial u}{\partial x}) \cdot \frac{\partial u}{\partial x} + d + 2|x|^2 + \frac{1}{2}(\frac{\partial u}{\partial x})^2 = 0 \\ \frac{1}{2}\Delta u - \frac{1}{2}|\frac{\partial u}{\partial x}|^2 + d + 2|x|^2 = 0 \end{aligned}$$

Therefore, the exact solution of the above benchmark problem is $u(x) = -|x|^2$ with $a = 2x$, and F can be written as

$$F(x, p, q) = \inf_{|a|\leq 3}\Big\{\frac{1}{2}\mathbf{tr}(q) + a \cdot p + \ell(x, a)\Big\}.$$

3 Methods of Approximation

First, we will introduce some notions of finite difference operators. Commonly used first order finite difference operators are Forward Finite Difference (FFD), Backward Finite Difference (BFD), Central Finite Difference (CFD) and Upwind Finite Difference (UFD). Here, to approximate the solution to the HJB equation, we mainly use CFD and UFD operators.

The CFD operate for the $\frac{\partial}{\partial x_i}u(x)$ is given below:

$$\frac{\partial}{\partial x_i}u(x) \approx \delta_{\pm he_i}u(x) := \frac{1}{2}(\delta_{-he_i} + \delta_{he_i})u(x) = \frac{u(x+he_i) - u(x-he_i)}{2h},$$

and the UFD operate for the $\frac{\partial}{\partial x_i}u(x)$ is given as follows:

$$\frac{\partial}{\partial x_i}u(x) \approx \begin{cases} \delta_{he_i}u(x) = \dfrac{u(x+he_i) - u(x)}{h}, & b(x) < 0 \\ \delta_{-he_i}u(x) = \dfrac{u(x) - u(x-he_i)}{h}, & b(x) \geq 0 \end{cases}$$

where $b(\cdot)$ is the coefficient function of $\frac{\partial}{\partial x_i}u(x)$.

The Second order finite difference operators in both the CFD scheme and the UFD scheme are as the following:

$$\frac{\partial^2}{\partial x_i^2}u(x) \approx \delta_{-he_i}\delta_{he_i}u(x) = \frac{u(x+he_i) - 2u(x) + u(x-he_i)}{h^2},$$

and if $i \neq j$, we use

$$\begin{aligned}\frac{\partial^2}{\partial x_i \partial x_j}u(x) &\approx \delta_{\pm he_i}\delta_{\pm he_j}u(x) \\ &= \frac{u(x+he_i+he_j) - u(x+he_i-he_j) - u(x-he_i+he_j) + u(x-he_i-he_j)}{4h^2}.\end{aligned}$$

3.1 CFD on HJB

Now, implement the CFD scheme on HJB equation (1) for approximation, we substitute

$$\frac{\partial}{\partial x_i}u(x) \leftarrow \delta_{\pm he_i}u(x)$$

and

$$\frac{\partial^2}{\partial x_i^2}u(x) \leftarrow \delta_{-he_i}\delta_{he_i}u(x).$$

Thus we can get the equations as follows,

$$\frac{1}{2}\sum_{i=1}^{d}\frac{v_i^+ - 2v + v_i^-}{h^2} - \lambda v + \inf_a\Big\{\sum_{i=1}^{d} b_i(x,a)\frac{v_i^+ - v_i^-}{2h} + \ell(x,a)\Big\} = 0,$$

and then we have

$$\inf_a\Big\{\sum_{i=1}^{d}[v_i^+(\frac{1}{2h^2} + \frac{b_i(x,a)}{2h}) + v_i^-(\frac{1}{2h^2} - \frac{b_i(x,a)}{2h})] + \ell(x,a)\Big\} = v(x)(\frac{d}{h^2} + \lambda).$$

Hence we can get that

$$v(x) = \frac{d}{d + h^2\lambda}\inf_a\Big\{\frac{h^2}{d}\ell(x,a) + \sum_{i=1}^{d}[v_i^+(\frac{1 + hb_i(x,a)}{2d}) + v_i^-(\frac{1 - hb_i(x,a)}{2d})]\Big\},$$

where, $v_i^+,\ v_i^-$ are the abbrevations of $v(x + he_i),\ v(x - he_i)$ respectively.

For simplicity, if we set

$$\gamma = \frac{d}{d + h^2\lambda},\ p^h(x \pm he_i|x,a) = \frac{1}{2d}(1 \pm hb_i(x,a)),\ \ell^h(x,a) = \frac{h^2\ell(x,a)}{d},$$

then it yields Markov decision process (MDP)

$$v(x) = \gamma\inf_a\Big\{\ell^h(x,a) + \sum_{i=1}^{d}\big[p^h(x + he_i|x,a)v(x + he_i) + p^h(x - he_i|x,a)v(x - he_i)\big]\Big\}.$$

In our benchmark problem, we have

$$\gamma = 1,\ p^h(x \pm he_i|x,a) = \frac{1}{2d}(1 \pm ha),\ \ell(x,a) = d + 2|x|^2 + \frac{1}{2}|a|^2.$$

3.2 UFD on HJB

If choose to implement the UFD scheme on HJB, we replace $\frac{\partial}{\partial x_i}u(x)$ by the equation mentioned before and

$$\frac{\partial^2}{\partial x_i^2}u(x) \leftarrow \delta_{-he_i}\delta_{he_i}u(x).$$

Then we can get the equation

$$\frac{1}{2}\sum_{i=1}^{d}\frac{v_i^+ - 2v + v_i^-}{h^2} - \lambda v + \inf_a\Big\{\sum_{i=1}^{d}(b_i^+\frac{v - v_i^-}{h} + b_i^-\frac{v_i^+ - v}{h})\ell(x,a)\Big\} = 0,$$

and then we have

$$\inf_a\Big\{\sum_{i=1}^{d}\frac{1 - 2hb_i^-}{2h^2}v_i^+ - (\frac{d}{h^2} + \lambda)v + \sum_{i=1}^{d}\frac{|b_i|}{h}v + \sum_{i=1}^{d}\frac{1 - 2hb_i^+}{2h^2}v_i^- + \ell(x,a)\Big\} = 0.$$

Move $v(x)$ to one side of this equation and denote $c = d - h\sum_{i=1}^{d}|b_i|$, we will get

$$\inf_a \{\frac{c+h^2\lambda}{h^2}\}v(x) = \inf_a \Big\{ \sum_{i=1}^{d} (\frac{1-2hb_i^-}{2h^2}v_i^+ + \frac{1-2hb_i^+}{2h^2}v_i^-) + \ell(x,a)\Big\}.$$

Thus we have

$$\begin{aligned} v(x) &= \inf_a \Big\{\frac{h^2}{c+h^2\lambda}\big[\sum_{i=1}^{d}(\frac{1-2hb_i^-}{2h^2}v_i^+ + \frac{1-2hb_i^+}{2h^2}v_i^-) + \ell(x,a)\big]\Big\} \\ &= \inf_a \Big\{\frac{c}{c+h^2\lambda}\big[\frac{h^2}{c}\ell(x,a) + \sum_{i=1}^{d}\big(\frac{1-2hb_i^-}{2c}v_i^+ + \frac{1-2hb_i^+}{2c}v_i^-\big)\big]\Big\}, \end{aligned}$$

where $b_i^+,\ b_i^-$ mean the positive part and the negative one of $b(x,a)$ respectively. The positive part is defined by the formula

$$b_i^+(x,a) = \max\{b_i(x,a),0\} = \begin{cases} b_i(x,a), & b_i(x,a)\geq 0 \\ 0, & \text{otherwise} \end{cases}$$

similarly, the negative part is defined as

$$b_i^-(x,a) = \max\{-b_i(x,a),0\} = \begin{cases} -b_i(x,a), & b_i(x,a)\leq 0 \\ 0, & \text{otherwise} \end{cases}$$

In a similar way as before, for simplicity we set

$$\gamma = \frac{c}{c+h^2\lambda},\ p^h(x\pm he_i|x,a) = \frac{1}{2c}(1-2hb_i^{\mp}(x,a)),\ \ell^h(x,a) = \frac{h^2\ell(x,a)}{c},$$

then it also yields MDP

$$v(x) = \inf_a \gamma\Big\{\ell^h(x,a) + \sum_{i=1}^{d}\big[p^h(x+he_i|x,a)v(x+he_i) + p^h(x-he_i|x,a)v(x-he_i)\big]\Big\}.$$

In our benchmark problem, we have

$$\gamma = 1,\ p^h(x\pm he_i|x,a) = \frac{1}{2c}(1-2ha^{\mp}),\ \ell(x,a) = d + 2|x|^2 + \frac{1}{2}|a|^2.$$

4 Numerical Results and Conclusion

We implement the CFD scheme and the UFD scheme to our benchmark problem with the first 3 dimension domain and want to solve this type of HJB equation by the value iteration method. We first implement the CFD scheme on one-dimensional benchmark

problem with the mesh size $h = 1/8$. From Figure 2, it can be seen that the approximate solution is sufficiently close to the exact solution. And we also apply the UFD scheme on one-dimensional benchmark problem with the mesh size $h = 1/8$, and the result is showed as the Figure 3. We can see that the UFD scheme can also get a close estimation to the exact solution of the benchmark problem. But the CFD scheme can get a better result than the UFD scheme.

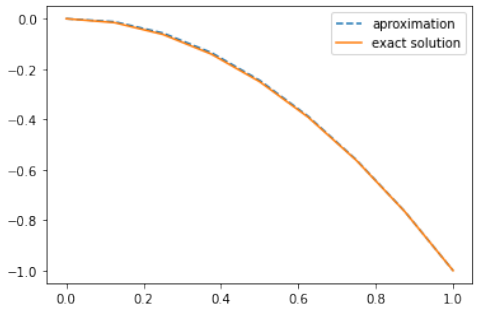

Figure 2: the CFD scheme, $d = 1$

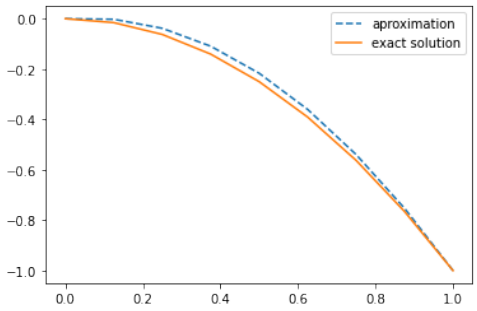

Figure 3: the UFD scheme, $d = 1$

For the $d = 2$, by the same way as the above and we can the numerical solutions of the benchmark problem, which showed as the Figure 4 and Figure 5. We can see that the numerical simulation results obtained by these two methods are very similar.

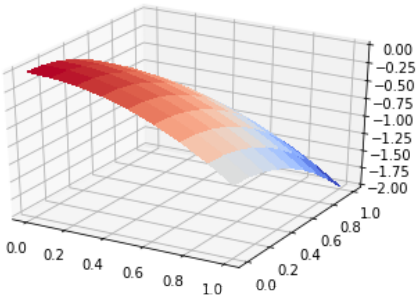

Figure 4: the CFD scheme, $d = 2$

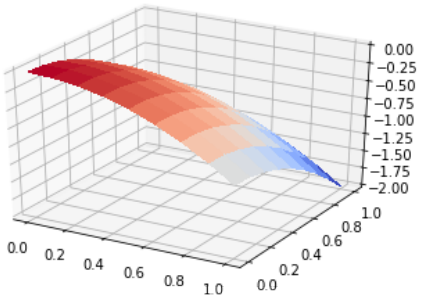

Figure 5: the UFD scheme, $d = 2$

To get a better evaluation about the simulation results of these methods on different dimension problem, we introduce the L^2 error between the exact value and the numerical solution of the benchmark problem. And the definition of the L^2 error in our report is

$$\text{error} = \frac{1}{N}\sum_{i=1}^{N}(v(x_i) - \hat{v}(x_i))^2,$$

where x_i is the point in the domain and N is the total number of the points we used to get the numerical solutions. And our experiment result is showed as the table 1. For

the case $d = 3$, the mesh size we used is $h = 1/4$. We can see that compared with the UFD scheme, the CFD scheme is more accurate to get the numerical solution of our benchmark problem. And we find that when the dimension goes up, it will spends more time to get the numerical result. So our method is not efficient for the high dimensional HJB equation. To get an efficient estimation in high dimensional problem, we may need apply the neural network approach.

Table 1: The L^2 error of each scheme in different dimension problems

Dimension	**CFD scheme**	**UFD scheme**
$d = 1$	0.0037	0.0224
$d = 2$	0.0015	0.0172
$d = 3$	0.0001	0.0225

References

[1] Q. Song. Convergence of markov chain approximation on generalized HJB equation and its applications, Automatica, 44(3):761–766, 2008.

[2] Wenhao Qiu, Qingshuo Song, George Yin. Solving Elliptic Hamilton-Jacobi-Bellman Equations in A Value Space.