DDA4300 Final Project Report
Value Iteration Methods on Zero-sum Games

Xu, Yuan (SID: 121090658)
Chinese University of Hong Kong, Shenzhen
yuanxu1@link.cuhk.edu.cn

Lin, Wenda (SID: 121090323)
Chinese University of Hong Kong, Shenzhen
wendalin@link.cuhk.edu.cn

Chen, Yanzhen (SID: 121090065)
Chinese University of Hong Kong, Shenzhen
yanzhenchen@link.cuhk.edu.cn

Wei, Jiaxing (SID: 121090594)
Chinese University of Hong Kong, Shenzhen
jiaxingwei@link.cuhk.edu.cn

Abstract

1. Introduction

[1. Basic background of RL methods in various environments and games. 2. Mention that Value Iteration is a basic but very fundamental method, with various improvement and application. 3. Our motivation for the project.]

2. Related Work

[2 parts. First: The ideas that improve Value Iteration methods. Second: The works that apply Value Iteration in zero-sum games.]

3. Methodologies

In our project, we made experiments on various environments with different variants of Value Iteration methods.

3.1. Standard Value Iteration (VI)

For a given Markov Decision Process (MDP), S and A denote the state space and action space, respectively. $T(s'|s,a)$ is the transition function and $R(s,a)$ is the reward function. The scheme to update the optimal value function V^* is applied synchronously across all states $s \in S$ synchronously:

$$\mathrm{V_{k+1}(s)} = \max_{a\in A}[R(s,a) + \gamma \sum_{s'} T(s'|s,a)V_k(s')]$$

where $||V^*_{k+1} - V^*_k||_1 \leq \epsilon$, guaranteeing convergence to the optimal value function $V*$ at a geometric rate proportional to the discount factor $\gamma \in [0,1)$ [6] below a threshold ϵ. Each iteration requires $O(|S|^2|A|)$. [4]

3.2. Convergence Proof of Value Iteration

[Answer the questions in the original project document.]

3.3. Random Value Iteration (RVI)

RVI introduces a sampling probability $\rho \in (0,1]$ to govern state updates. [1] At each iteration k, this algorithm selects a random subset of $S_k \in S$ where each state $s \in S$ has independent probability ρ of being included.

$$Q_k(s,a) = R(s,a) + \gamma \sum_{s'} T(s,a,s')V_k(s')$$

$$\mathrm{V_{k+1}(s)} = \begin{cases} Q_k(s,a) & \text{w.p. } \rho \\ V_k(s) & \text{w.p. } 1-\rho \end{cases}$$

According to Tsitsiklis [7], this approach achieves an expected per-iteration complexity to $O(\rho|S|^2|A|)$. The sampling probability ρ brings a trade-off between computational savings and convergence rate, with smaller values accelerating iterations but potentially requiring more total updates, and vice versa.

3.4. RVI on Reduced State Space (RSRVI)

The method operates on an MDP where each state s' maintains a set of predecessor states that s' can be reached in a single transition, i.e., $\exists a \in A, T(s'|s,a) > 0$. The algorithm maintains a priority queue that ranks states according to their Bellman error:

$$\delta(s) = |V(s) - \max_{a\in A}[R(s,a) + \gamma \sum_{s'} T(s'|s,a)V_k(s')]|$$

After updating s, all predecessors s_{pred} have their Bellman errors recomputed and priorities adjusted. Each maintenance requires $O(|A||S| + log|S|)$. [5]

3.5. Cyclic Value Iteration (CVI)

For a given MDP, CVI utilizes the Gauss-Seidel scheme to update the optimal function value by incorporating immediate updates of state values during each iteration:

$$V_k(s_i) = \max_{a \in A} \Big[R(s_i, a) + \gamma \Big(\sum_{j=1}^{i-1} T(s_j|s_i,a) V_k(s_j) + \sum_{j=i}^{|S|} T(s_j|s_i,a) V_{k-1}(s_j) \Big) \Big]$$

CVI utilizes updated value $V_k(s_j)$ for predecessor states $j < i$ and previous value $V_{k-1}(s_j)$ for $j \geq i$. The scheme maintains the same $O(|S|^2|A|)$ per iteration as the Standard VI. [2]

3.6. Random Permutation CVI (RPCVI)

Contrasted with fixed-sequence CVI, the algorithm updates sequence V_k through state ordering randomization [3]:

$$V_k(s_{\sigma_k(i)}) = \max_{a \in A} \left[R(s_{\sigma_k(i)}, a) + \gamma \sum_{s'} T(s'|s_{\sigma_k(i)}, a) \tilde{V}(s') \right]$$

$$\tilde{V}(s') = \begin{cases} V_k(s') & \text{if } \sigma_k^{-1}(s') < i \\ V_{k-1}(s') & \text{otherwise} \end{cases}$$

The random reordering σ_k regenerates every iteration to prevent directional bias in value propagation.

4. Implementations

[Here write some details in implementation and interface of our implementation.]

5. Experiments

5.1. Convergence Performance of VI methods

OpenAI Gym: [Here write a brief introduction to the OpenAI Gym environments as a RL environment. We also need to cite OpenAI Gym.]

VI variants	FrozenLake-v1		Taxi-v3	
Metrics	# Iteration	CPU Time	# Iteration	CPU time
Standard (VI)[3.1]	1372	0.2625	4004	29.8630
Random (RVI)[3.3]	592	0.0549	3508	1.8483
RSRVI[3.4]	225	0.0069	289	0.1948
Cyclic (CVI)[3.5]	990	0.1747	2063	15.3803
RPCVI[3.6]	1055	0.1925	2684	20.5360

Table 1. Convergence Iterations and Execution Time (unit: seconds) for Convergence of Different Variants of Value Iteration on *FrozenLake-v1* and *Taxi-v3* environments.

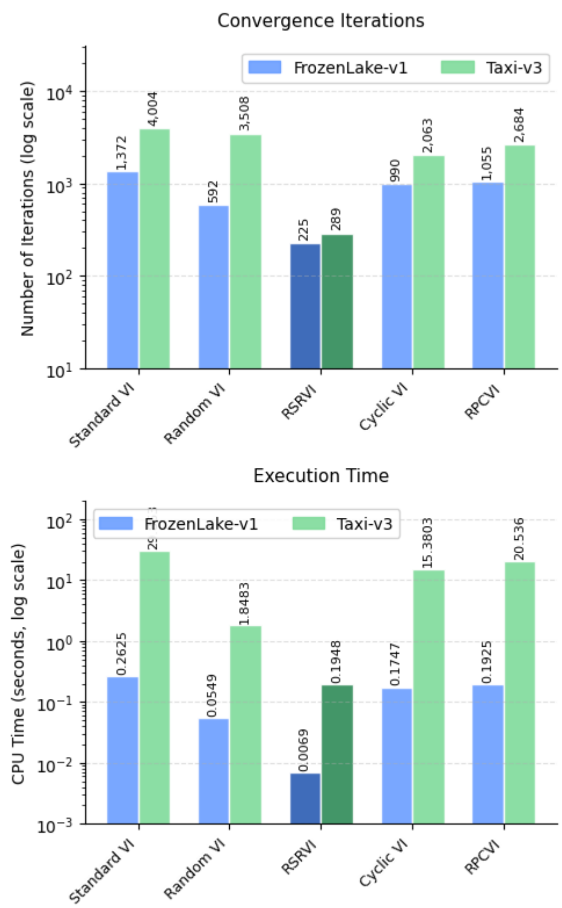

Figure 1. Plots of Iterations and Execution Time

5.2. Performance of Cyclic Value Iteration

[Plot the adjustment of parameters.]

5.3. Performance of Random Value Iteration

[Plot the adjustment of parameters.]

6. Applications

6.1. Grid World (Optional)

[Build a fundamental grid world environment or find another good example. (zero-sum game can be better!)]

6.2. Zero-sum Game

6.3. Tic-Tac-Toe Game

7. Conclusion

[Our project first implement the value iteration and its four variants. The experiment results demonstrates that ... We also apply value iteration agents on zero-sum games. The agents perform ...]

8. Contribution Distribution

The group had good organization and collaboration to complete this project about Value Iteration methods. The contribution of each group members are listed in the table 2.

Tasks	Xu	Lin	Chen	Wei
Datasets Collection	✓†			✓†
Method Reproduction	✓†			
Method Application	✓†	✓†		
Literature Review			✓†	
Paper Writing	✓		✓	✓†

Table 2. Table of Contribution Distribution. ✓ represents participation in the part, and ✓† represents making the major contribution to the part.

References

[1] Dimitri P. Bertsekas. Neuro-dynamic programming: An overview. *Laboratory for Information and Decision Systems, Massachusetts Institute of Technology*, 1995. Technical Report. 1

[2] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, third edition, 2017. 2

[3] Eyal Even-Dar and Yishay Mansour. Learning rates for q-learning. *Journal of Machine Learning Research*, 5:1–25, 2003. 2

[4] Michael L Littman, Thomas L Dean, and Leslie Pack Kaelbling. On the complexity of solving markov decision problems. *arXiv preprint arXiv:1302.4971*, 2013. 1

[5] Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less real time. *MIT Artificial Intelligence Laboratory*, 1993. Technical Report. 1

[6] Aaron Sidford, Mengdi Wang, Xian Wu, and Yinyu Ye. Variance reduced value iteration and faster algorithms for solving markov decision processes. *Naval Research Logistics (NRL)*, 70(5):423–442, 2023. 1

[7] John N. Tsitsiklis. Asynchronous stochastic approximation and q-learning. *Machine Learning*, 16:185–202, 1994. Manufactured in The Netherlands. 1