AsyncQVI: Asynchronous Randomized Q-Value Iteration For Reinforcement Learning

Fei Feng

joint work with Yibo Zeng and Wotao Yin

INFORMS 2019

Motivation: Accelerate Learning

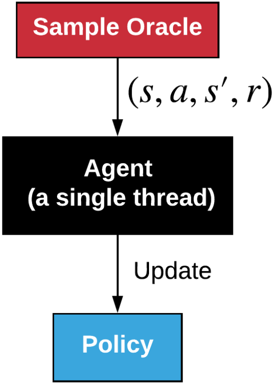


Figure: Paradigm for RL algorithms with a single computing agent.

Motivation: Accelerate Learning

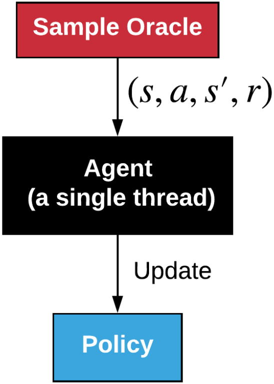


Figure: Paradigm for RL algorithms with a single computing agent.

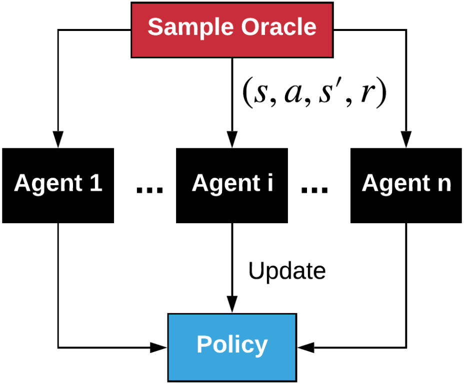
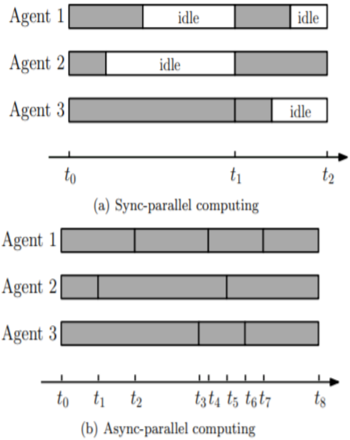


Figure: Paradigm for RL algorithms with multiple computing agents.

Technique: Asynchronous Parallel



s

Figure: Pictures from Peng et al. 2016

Sync-parallel:

- probably long idle time;
- little tolerant to communication glitches;
- keeps information consistent.

Async-parallel:

- saves idle time
- more tolerant to communication glitches;
- easier to incorporate new agents;
- information is delayed or inconsistent.

Challenges and Solution

Error sources:

- randomization
- delayed and inconsistent information

Challenges and Solution

Error sources:

- randomization
- delayed and inconsistent information

Assumptions:

- delay is uniformly bounded by B_1.
- the time interval between consecutive updates for each coordinate is uniformly bounded by B_2.
- a generative model.

Challenges and Solution

Error sources:

- randomization
- delayed and inconsistent information

Assumptions:

- delay is uniformly bounded by B_1.
- the time interval between consecutive updates for each coordinate is uniformly bounded by B_2.
- a generative model.

Leverage: The contraction property of the Bellman operator.

Key Idea:

Key Idea:

1. Q-value iteration:

$$Q_{s,a}(t+1) = \sum_{s'} p_{ss'}^{a} r_{ss'}^{a} + \gamma \sum_{s'} p_{ss'}^{a} \max_{a'} Q_{s',a'}(t), \ \forall \ s, a$$

Key Idea:

1. Q-value iteration:

$$Q_{s,a}(t+1) = \sum_{s'} p^a_{ss'} r^a_{ss'} + \gamma \sum_{s'} p^a_{ss'} \max_{a'} Q_{s',a'}(t), \ \forall \ s, a$$

2. Revise the former step to coordinate update

$$Q_{s,a}(t+1) = \begin{cases} \sum_{s'} p^a_{ss'} r^a_{ss'} + \gamma \sum_{s'} p^a_{ss'} \max_{a'} Q(t)_{s',a'}, & (s_{t+1}, a_{t+1}); \\ Q_{s,a}(t), & \text{o.w.} \end{cases}$$

Key Idea:

1. Q-value iteration:

$$Q_{s,a}(t+1) = \sum_{s'} p^a_{ss'} r^a_{ss'} + \gamma \sum_{s'} p^a_{ss'} \max_{a'} Q_{s',a'}(t), \ \forall \ s, a$$

2. Revise the former step to coordinate update

$$Q_{s,a}(t+1) = \begin{cases} \sum_{s'} p^a_{ss'} r^a_{ss'} + \gamma \sum_{s'} p^a_{ss'} \max_{a'} Q(t)_{s',a'}, & (s_{t+1}, a_{t+1}); \\ Q_{s,a}(t), & \text{o.w.} \end{cases}$$

3. implement in an asynchronous parallel manner:

$$Q_{s,a}(t+1) = \begin{cases} \sum_{s'} p^a_{ss'} r^a_{ss'} + \gamma \sum_{s'} p^a_{ss'} \max_{a'} \hat{Q}_{s',a'}, & (s_{t+1}, a_{t+1}); \\ Q_{s,a}(t), & \text{o.w.} \end{cases}$$

Key Idea:

1. Q-value iteration:

$$Q_{s,a}(t+1) = \sum_{s'} p^a_{ss'} r^a_{ss'} + \gamma \sum_{s'} p^a_{ss'} \max_{a'} Q_{s',a'}(t), \ \forall\ s, a$$

2. Revise the former step to coordinate update

$$Q_{s,a}(t+1) = \begin{cases} \sum_{s'} p^a_{ss'} r^a_{ss'} + \gamma \sum_{s'} p^a_{ss'} \max_{a'} Q(t)_{s',a'}, & (s_{t+1}, a_{t+1}); \\ Q_{s,a}(t), & \text{o.w.} \end{cases}$$

3. implement in an asynchronous parallel manner:

$$Q_{s,a}(t+1) = \begin{cases} \sum_{s'} p^a_{ss'} r^a_{ss'} + \gamma \sum_{s'} p^a_{ss'} \max_{a'} \hat{Q}_{s',a'}, & (s_{t+1}, a_{t+1}); \\ Q_{s,a}(t), & \text{o.w.} \end{cases}$$

4. Further revise to a randomized fashion with active sampling:

$$Q_{s,a}(t+1) = \begin{cases} \frac{1}{K} \sum_k r_k + \gamma \frac{1}{K} \sum_k \max_{a'} \hat{Q}_{s'_k,a'} - \frac{(1-\gamma)\varepsilon}{4}, & (s_{t+1}, a_{t+1}); \\ Q_{s,a}(t), & \text{o.w.} \end{cases}$$

Algorithm

Algorithm 1: AsyncQVI: Asynchronous-Parallel Q-value Iteration

Input: $\varepsilon \in (0, (1-\gamma)^{-1})$, $\delta \in (0,1)$, L, K;
Shared variables: $\mathbf{v} \leftarrow \mathbf{0}$, $\pi \leftarrow \mathbf{0}$, $t \leftarrow 0$;
Private variables: $\hat{\mathbf{v}}, r, S, q$;
while $t < L$, *every agent asynchronously* **do**
  select state $i_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$;
  copy shared variable to local memory $\hat{\mathbf{v}} \leftarrow \mathbf{v}$;
  call $\mathtt{GM}(s_t, a_t)$ K times and collect samples $\{s'_1, \ldots, s'_K\}$ and $r_1, \ldots, r_K$;
  $q \leftarrow \frac{1}{K}\sum_{k=1}^{K} r_k + \gamma \frac{1}{K}\sum_{k=1}^{K} \hat{v}_{s'_k} - \frac{(1-\gamma)\varepsilon}{4}$;
  if $q > v_{i_t}$ **then**
    mutex lock;
    $v_{i_t} \leftarrow q$, $\pi_{i_t} \leftarrow a_t$;
    mutex unlock;
  increment the global counter $t \leftarrow t+1$;
return π

Sample Complexity

Theorem 1 (Zeng, Feng, and Yin 2018)

Under the assumptions, AsyncQVI returns an ε-optimal policy π with probability at least $1-\delta$ at the sample complexity

$$\tilde{\mathcal{O}}\Big(\frac{B_1+B_2}{(1-\gamma)^5\varepsilon^2}\log(\frac{1}{\delta})\Big).$$

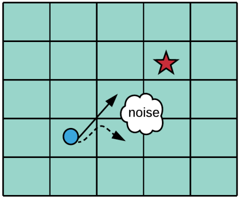
In [Azar, Munos, and Kappen 2013], it shows that the optimal sample complexity with a generative model is:

$$\mathcal{O}\Big(\frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^3\varepsilon^2}\log(\frac{|\mathcal{S}||\mathcal{A}|}{\delta})\Big).$$

Related Algorithms

Algorithms	Async	Sample Complexity	Memory	References
Variance-Reduced VI	$\times$	$\tilde{O}\left(\frac{\|\mathcal{S}\|\|\mathcal{A}\|}{(1-\gamma)^4\varepsilon^2}\log(\frac{1}{\delta})\right)$	$\mathcal{O}(\|\mathcal{S}\|\|\mathcal{A}\|)$	Sidford, Wang, Wu, and Ye 2018
Variance-Reduced QVI	$\times$	$\tilde{O}\left(\frac{\|\mathcal{S}\|\|\mathcal{A}\|}{(1-\gamma)^3\varepsilon^2}\log(\frac{1}{\delta})\right)$	$\mathcal{O}(\|\mathcal{S}\|\|\mathcal{A}\|)$	Sidford, Wang, Wu, Yang, et al. 2018
Async Q-learning	$\surd$	–	$\mathcal{O}(\|\mathcal{S}\|\|\mathcal{A}\|)$	Tsitsiklis 1994
AsyncQVI	$\surd$	$\tilde{O}\left(\frac{\|\mathcal{S}\|\|\mathcal{A}\|}{(1-\gamma)^5\varepsilon^2}\log(\frac{1}{\delta})\right)$	$\mathcal{O}(\|\mathcal{S}\|)$	This Work

Numerical Test: Sailing Problem



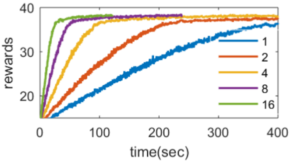


Figure: Parallel speedup

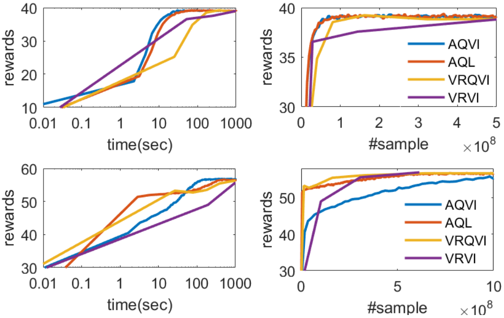


Figure: Performance with 20 parallel threads and different noises.

Conclusion and Future Work

Conclusion:

- We propose an asynchronous algorithm AsyncQVI for RL with explicit sample complexity.
- AsyncQVI trades a little more samples for less time and memory.
- AsyncQVI has linear parallel speedup empirically.

Conclusion and Future Work

Conclusion:

- We propose an asynchronous algorithm AsyncQVI for RL with explicit sample complexity.
- AsyncQVI trades a little more samples for less time and memory.
- AsyncQVI has linear parallel speedup empirically.

Future work:

- Add variance reduction trick to achieve a better sample complexity result;
- Relax generative model to exploration policy.

Thank you!

References

Azar, Mohammad Gheshlaghi, Rémi Munos, and Hilbert J Kappen (2013). "Minimax PAC bounds on the sample complexity of reinforcement learning with a generative model". In: *Machine learning* 91.3, pp. 325–349.

Peng, Zhimin et al. (2016). "Arock: an algorithmic framework for asynchronous parallel coordinate updates". In: *SIAM Journal on Scientific Computing* 38.5, A2851–A2879.

Sidford, Aaron, Mengdi Wang, Xian Wu, Lin Yang, et al. (2018). "Near-Optimal Time and Sample Complexities for Solving Markov Decision Processes with a Generative Model". In: *Advances in Neural Information Processing Systems*, pp. 5192–5202.

Sidford, Aaron, Mengdi Wang, Xian Wu, and Yinyu Ye (2018). "Variance reduced value iteration and faster algorithms for solving markov decision processes". In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, pp. 770–787.

Tsitsiklis, John N. (Sept. 1994). "Asynchronous stochastic approximation and Q-learning". In: *Machine Learning* 16.3, pp. 185–202. ISSN: 1573-0565.

Zeng, Yibo, Fei Feng, and Wotao Yin (2018). "AsyncQVI: Asynchronous-Parallel Q-Value Iteration for Reinforcement Learning with Near-Optimal Sample