Speedy Q-Learning

Akif Khan

Reinforcement Learning Course Project
Indian Institute of Science

April, 2024

Q-Learning is Slow

- We know Q-Learning converges to the optimal value function with sample complexity $\tilde{O}\left(\frac{|S||A|}{(1-\gamma)^5\epsilon^2}\right)$

Q-Learning is Slow

- We know Q-Learning converges to the optimal value function with sample complexity $\tilde{O}\left(\frac{|S||A|}{(1-\gamma)^5\epsilon^2}\right)$
- But it is slow!

Q-Learning is Slow

- We know Q-Learning converges to the optimal value function with sample complexity $\tilde{O}\left(\frac{|S||A|}{(1-\gamma)^5\epsilon^2}\right)$
- But it is slow!
  - Sample inefficient due to decaying learning rate in the stochastic approximation. For ex. for $\alpha_k = \frac{1}{k+1}$, we have an overly pessimistic step size for large k

Q-Learning is Slow

- We know Q-Learning converges to the optimal value function with sample complexity $\tilde{O}\left(\frac{|S||A|}{(1-\gamma)^5\epsilon^2}\right)$
- But it is slow!
  - Sample inefficient due to decaying learning rate in the stochastic approximation. For ex. for $\alpha_k = \frac{1}{k+1}$, we have an overly pessimistic step size for large k
  - Propagation of the Bellman Operator throughout the space for discount factors close to 1

Q-Learning is Slow

- We know Q-Learning converges to the optimal value function with sample complexity $\tilde{O}\left(\frac{|S||A|}{(1-\gamma)^5\epsilon^2}\right)$
- But it is slow!
  - Sample inefficient due to decaying learning rate in the stochastic approximation. For ex. for $\alpha_k = \frac{1}{k+1}$, we have an overly pessimistic step size for large k
  - Propagation of the Bellman Operator throughout the space for discount factors close to 1
  - These issues can be resolved with Speedy Q-Learning

Notations Used

- X, A are finite sets and $Z = |X||A|$
- The immediate rewards $r(x, a)$ are bounded, i.e. $|r(x, a)| \leq R_{max}$
- Let γ be the discount factor and define $\beta = \frac{1}{1-\gamma}$ and $V_{max} = \beta R_{max}$
- T_k and T are the Bellman operator and the Bellman optimality operator
- $Q(x, a)$ is the Q value function and define $(MQ)(x) = \max_{a \in A} Q(x, a)$, $\forall x \in X$
- α_k be the step size

Pseudocode for Speedy Q-Learning

Algorithm 1: Synchronous Speedy Q-Learning (SQL)

Input: Initial action-value function Q_0, discount factor γ, and number of iteration T

```
Q_{-1} := Q_0;                                        // Initialization
for k := 0, 1, 2, 3, ..., T − 1 do                    // Main loop
    α_k := 1/(k+1);
    for each (x, a) ∈ Z do
        Generate the next state sample y_k ~ P(·|x, a);
        T_k Q_{k−1}(x, a) := r(x, a) + γ M Q_{k−1}(y_k);
        T_k Q_k(x, a) := r(x, a) + γ M Q_k(y_k);      // Empirical Bellman operator
        Q_{k+1}(x, a) := Q_k(x, a) + α_k (T_k Q_{k−1}(x, a) − Q_k(x, a)) + (1 − α_k)(T_k Q_k(x, a) − T_k Q_{k−1}(x, a));
                                                      // SQL update rule
    end
end
return Q_T
```

Pseudocode for Speedy Q-Learning

Algorithm 1 Synchronous Speedy Q-Learning(SQL)

Initialize Q_0, γ, Number of Iterations, T

$Q_{-1} \leftarrow Q_0$

for $k = 0, 1, ..., T-1$ **do**

  $\alpha_k = \frac{1}{k+1}$

  for *each* $(x, a) \in Z$ **do**

    Generate the next sample y_k $P(.|x, a)$

    $T_k Q_{k-1}(x, a) \leftarrow r(x, a) + \gamma(MQ_{k-1})(y_k)$

    $T_k Q_k(x, a) \leftarrow r(x, a) + \gamma(MQ_k)(y_k)$

    $Q_{k+1}(x, a) \leftarrow Q_k(x, a) + \alpha_k(T_k Q_{k-1}(x, a) - Q_k)(x, a)) +$
    $(1 - \alpha_k)(T_k Q_k(x, a) - T_k Q_{k-1}(x, a))$

  end

end

return Q_T

An Intuitive Explanation

- The update rule for Q-Learning is given by

$$Q_{k+1}(x,a) = Q_k(x,a) + \alpha_k(T_k Q_k(x,a) - Q_k(x,a))$$

An Intuitive Explanation

- The update rule for Q-Learning is given by

$$Q_{k+1}(x,a) = Q_k(x,a) + \alpha_k(T_kQ_k(x,a) - Q_k(x,a))$$

- Adding and subtracting $T_kQ_{k-1}(x,a)$ for the term inside the bracket gives us

$$\begin{aligned} Q_{k+1}(x,a) = Q_k(x,a) &+ \alpha_k(T_kQ_{k-1}(x,a) - Q_k(x,a)) + \\ &\alpha_k(T_kQ_k(x,a) - T_kQ_{k-1}(x,a)) \end{aligned}$$

An Intuitive Explanation

- The update rule for Q-Learning is given by

$$Q_{k+1}(x,a) = Q_k(x,a) + \alpha_k(T_k Q_k(x,a) - Q_k(x,a))$$

- Adding and subtracting $T_k Q_{k-1}(x,a)$ for the term inside the bracket gives us

$$\begin{aligned} Q_{k+1}(x,a) = Q_k(x,a) + &\alpha_k(T_k Q_{k-1}(x,a) - Q_k(x,a)) + \\ &\alpha_k(T_k Q_k(x,a) - T_k Q_{k-1}(x,a)) \end{aligned}$$

- The last term on the RHS goes to zero as $Q_k \to Q^*$, so a more aggressive step-size, $1 - \alpha_k = \frac{k}{k+1}$ can be used for this term

1 Motivation for Speedy Q-Learning

2 Algorithm

3 Simulations

4 Proposed Modification to SQL

5 Results

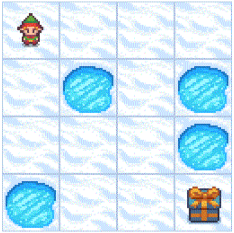
6 Questions

Environment and Libraries

Frozen-Lake environment:

- $4 \times 4 = 16$ states.
  S:Start, H:Hole, G:Goal, F: Frozen
- 4 actions.
  0:Left, 1:Down, 2: Right, 3:Up
- Rewards.
  Reach Goal:+1, Reach Hole: 0, Reach Frozen: 0
- Slipping.
  Slips from target direction to either side, with equal probabilities. Eg: If moving left, it goes to left with $\frac{1}{3}$ probability and slips top/bottom with probability $\frac{1}{3}$.

```
SFFF
FHFH
FFFH
HFFG
```



Libraries Used: gym, matplotlib, IPython.display and numpy.

Numerical Experiments for $\gamma = 0.99$

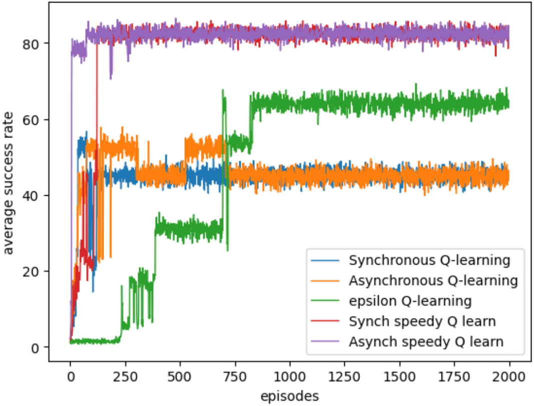


Figure 1: Discount factor $\gamma = 0.99$. Success Rate[1] vs Episodes[2]

[1] Success is percentage of the trials reached to Goal.
[2] Number of Iterations

Numerical Experiments for $\gamma = 0.9$

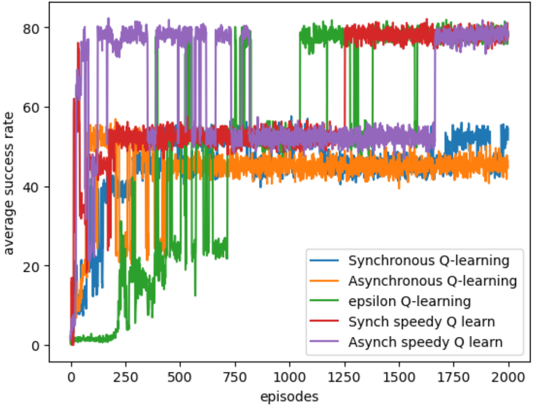


Figure 2: Discount factor $\gamma = 0.9$. Success Rate[3] vs Episodes[4]

[3] Success is percentage of the trials reached to Goal.
[4] Number of Iterations

Numerical Experiments for $\gamma = 0.8$

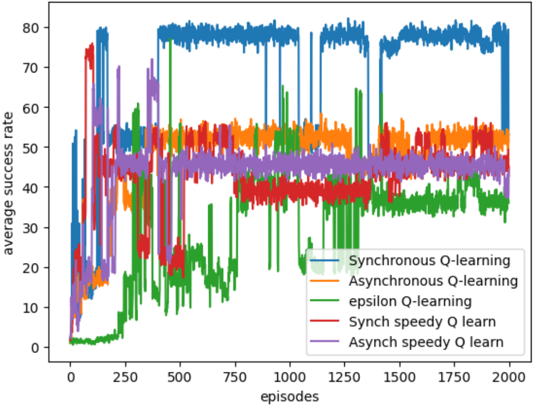


Figure 3: Discount factor $\gamma = 0.8$. Success Rate[5] vs Episodes[6]

[5] Success is percentage of the trials reached to Goal.

[6] Number of Iterations

Numerical Experiments for $\gamma = 0.7$

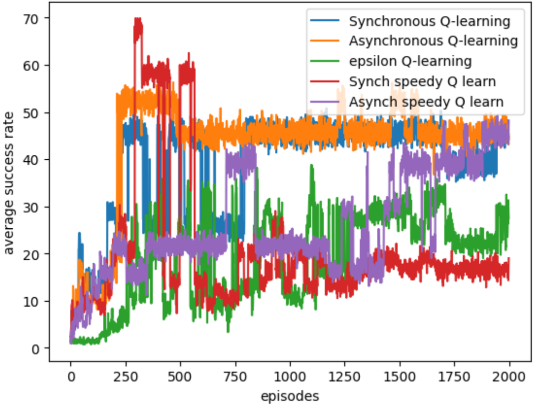


Figure 4: Discount factor $\gamma = 0.7$. Success Rate[7] vs Episodes[8]

[7] Success is percentage of the trials reached to Goal.

[8] Number of Iterations

1. Motivation for Speedy Q-Learning
2. Algorithm
3. Simulations
4. Proposed Modification to SQL
5. Results
6. Questions

A Proposed Modification to SQL

- The Speedy Q-Learning update rule is

$$\begin{aligned} Q_{k+1}(x,a) := Q_k(x,a) + \alpha_k\big(T_k Q_{k-1}(x,a) - Q_k(x,a)\big) \\ + (1-\alpha_k)\big(T_k Q_k(x,a) - T_k Q_{k-1}(x,a)\big) \end{aligned}$$

A Proposed Modification to SQL

- The Speedy Q-Learning update rule is

$$\begin{aligned} Q_{k+1}(x,a) := Q_k(x,a) + \alpha_k\big(T_k Q_{k-1}(x,a) - Q_k(x,a)\big) \\ + (1-\alpha_k)\big(T_k Q_k(x,a) - T_k Q_{k-1}(x,a)\big) \end{aligned}$$

- Why to stop at one buffer, $Q_{k-1}(x,a)$? We can have more than one buffer like $Q_{k-2}(x,a)$, $Q_{k-3}(x,a)$ and so on. We can use a convex combination of $\alpha_k' s$ such that the step-size gets more aggressive for the terms that go to zero faster

$$\begin{aligned} Q_{k+1}(x,a) := Q_k(x,a) + \alpha_k\big(T_k Q_{k-1}(x,a) - Q_k(x,a)\big) \\ + \frac{2}{3}(1-\alpha_k)\big(T_k Q_k(x,a) - T_k Q_{k-1}(x,a)\big) \\ + \frac{1}{3}(1-\alpha_k)\big(T_k Q_{k-1}(x,a) - T_k Q_{k-2}(x,a)\big) \end{aligned}$$

A Proposed Modification to SQL

- The Speedy Q-Learning update rule is

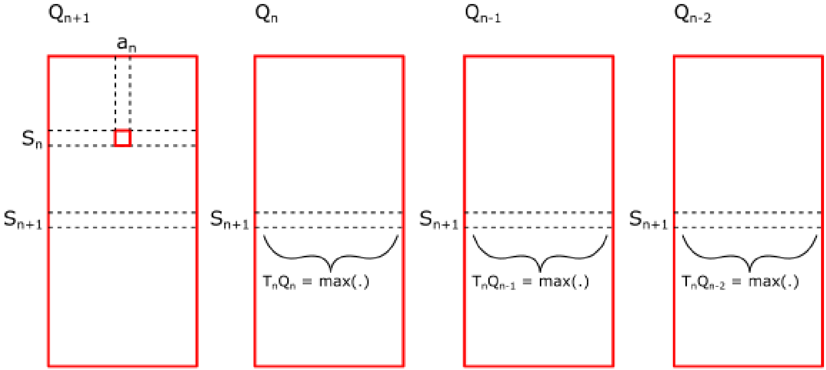
$$
\begin{aligned}
Q_{k+1}(x,a) := Q_k(x,a) &+ \alpha_k\big(T_k Q_{k-1}(x,a) - Q_k(x,a)\big) \\
&+ (1-\alpha_k)\big(T_k Q_k(x,a) - T_k Q_{k-1}(x,a)\big)
\end{aligned}
$$

- Why to stop at one buffer, $Q_{k-1}(x,a)$? We can have more than one buffer like $Q_{k-2}(x,a)$, $Q_{k-3}(x,a)$ and so on. We can use a convex combination of $\alpha_k' s$ such that the step-size gets more aggressive for the terms that go to zero faster

$$
\begin{aligned}
Q_{k+1}(x,a) := Q_k(x,a) &+ \alpha_k\big(T_k Q_{k-1}(x,a) - Q_k(x,a)\big) \\
&+ \frac{2}{3}(1-\alpha_k)\big(T_k Q_k(x,a) - T_k Q_{k-1}(x,a)\big) \\
&+ \frac{1}{3}(1-\alpha_k)\big(T_k Q_{k-1}(x,a) - T_k Q_{k-2}(x,a)\big)
\end{aligned}
$$

- **Observation:** It seems to outperform the Speedy Q learning

A Proposed Modification to SQL



$$
\begin{aligned}
Q_{k+1}(x,a) := Q_k(x,a) &+ \alpha_k\big(T_k Q_{k-1}(x,a) - Q_k(x,a)\big) \\
&+ \frac{2}{3}(1-\alpha_k)\big(T_k Q_k(x,a) - T_k Q_{k-1}(x,a)\big) \\
&+ \frac{1}{3}(1-\alpha_k)\big(T_k Q_{k-1}(x,a) - T_k Q_{k-2}(x,a)\big)
\end{aligned}
$$

Numerical results for Modified SQL(1 of 4)

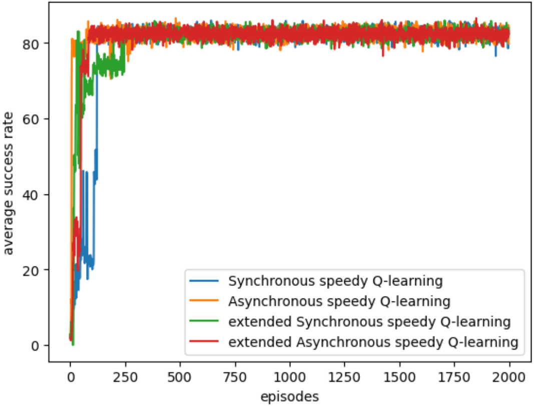


Figure 6: Discount factor $\gamma = 0.99$. Success Rate[9] vs Episodes[10]

[9] Success is percentage of the trials reached to Goal.

[10] Number of Iterations

Numerical results for Modified SQL(2 of 4)

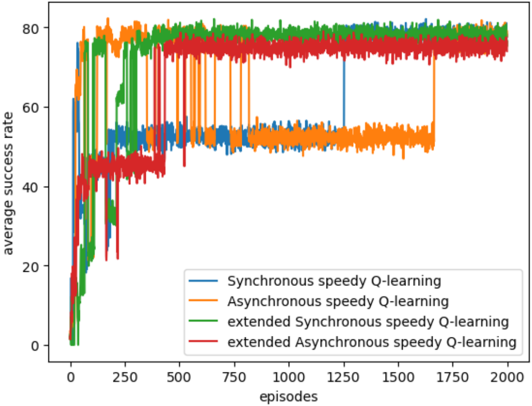


Figure 7: Discount factor $\gamma = 0.9$. Success Rate[11] vs Episodes[12]

[11] Success is percentage of the trials reached to Goal.

[12] Number of Iterations

Numerical results for Modified SQL(3 of 4)

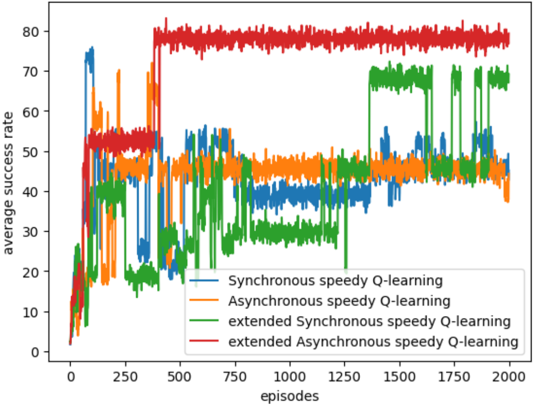


Figure 8: Discount factor $\gamma = 0.8$. Success Rate[13] vs Episodes[14]

[13] Success is percentage of the trials reached to Goal.

[14] Number of Iterations

Numerical results for Modified SQL(4 of 4)

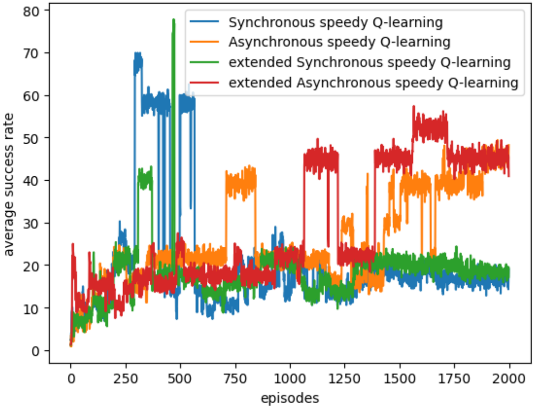


Figure 9: Discount factor $\gamma = 0.7$. Success Rate[15] vs Episodes[16]

[15] Success is percentage of the trials reached to Goal.

[16] Number of Iterations

1. Motivation for Speedy Q-Learning
2. Algorithm
3. Simulations
4. Proposed Modification to SQL
5. **Results**
   - Main Theoretical Result
   - Comparison of Q-Learning vs Speedy Q-Learning
6. Questions

1. Motivation for Speedy Q-Learning
2. Algorithm
3. Simulations
4. Proposed Modification to SQL
5. Results
    - Main Theoretical Result
    - Comparison of Q-Learning vs Speedy Q-Learning
6. Questions

- *Let Assumption 1 holds and T be a positive integer. Then, at iteration T of SQL with probability at least $1 - \delta$, we have*

$$||Q^* - Q_k|| \leq 2\beta^2 R_{max} \left[\frac{\gamma}{T} + \sqrt{\frac{2\log\frac{2n}{\delta}}{T}} \right]$$

- *Let Assumption 1 holds and T be a positive integer. Then, at iteration T of SQL with probability at least $1-\delta$, we have*

$$||Q^* - Q_k|| \leq 2\beta^2 R_{max} \left[\frac{\gamma}{T} + \sqrt{\frac{2\log\frac{2n}{\delta}}{T}} \right]$$

- This shows that $Q^* \xrightarrow{a.s.} Q$ with rate $\sqrt{\frac{1}{T}}$

- *Let Assumption 1 holds and T be a positive integer. Then, at iteration T of SQL with probability at least $1-\delta$, we have*

$$||Q^* - Q_k|| \leq 2\beta^2 R_{max} \left[\frac{\gamma}{T} + \sqrt{\frac{2\log\frac{2n}{\delta}}{T}} \right]$$

- This shows that $Q^* \xrightarrow{\text{a.s.}} Q$ with rate $\sqrt{\frac{1}{T}}$
- Additionally, *for any $\epsilon > 0$, after $T = \frac{11.66\beta^4 R_{max}^2 \log\frac{2n}{\delta}}{\epsilon^2}$ steps of SQL, $||Q^* - Q_k|| \leq \epsilon$, with probability at least $1-\delta$*

- It has been shown for Q-Learning the finite time PAC bound for a step size, $\alpha_k = \frac{1}{(k+1)^{\omega}}$, where $\omega \in (0.5, 1)$, the ϵ-optimal performance w.p at least $1 - \delta$ after

$$T = O\left(\left[\frac{\beta^4 R_{max}^2 \log \frac{n\beta R_{max}}{\delta\epsilon}}{\epsilon^2} \right]^{\frac{1}{\omega}} + \left[\beta \log \frac{\beta R_{max}}{\epsilon} \right]^{\frac{1}{1-\omega}} \right)$$

- It has been shown for Q-Learning the finite time PAC bound for a step size, $\alpha_k = \frac{1}{(k+1)^\omega}$, where $\omega \in (0.5, 1)$, the ϵ-optimal performance w.p at least $1-\delta$ after

$$T = O\left(\left[\frac{\beta^4 R_{max}^2 \log \frac{n\beta R_{max}}{\delta\epsilon}}{\epsilon^2}\right]^{\frac{1}{\omega}} + \left[\beta \log \frac{\beta R_{max}}{\epsilon}\right]^{\frac{1}{1-\omega}}\right)$$

- We see when $\gamma \approx 1$, and we have $O\left(\beta^{\frac{4}{\omega}} + \beta^{\frac{1}{1-\omega}}\right)$ dominates the performance and the optimal bound. Which will be optimized when the exponents are equal, giving us $O(\frac{\beta^5}{\epsilon^{2.5}})$

- Summary of Q-Learning vs Speedy Q-Learning

Method	SQL	Q-learning
SC	$\tilde{O}\left(\frac{n\beta^4}{\epsilon^2}\right)$	$\tilde{O}\left(\frac{n\beta^5}{\epsilon^{2.5}}\right)$
CC	$\tilde{O}\left(\frac{n\beta^4}{\epsilon^2}\right)$	$\tilde{O}\left(\frac{n\beta^5}{\epsilon^{2.5}}\right)$
SPC	$\Theta(n)$	$\Theta(n)$

Questions?