QL vs. SARSA: Mathematical Analysis

Joseph Parayil

March 2025

Claim

Let the training configuration C be defined as:

$$C = (\Omega, T, E, S, \alpha, \gamma, \epsilon)$$

where:

Ω is the environment.

T is the number of trials.

E is the number of episodes.

S is the maximum number of steps per episode.

α is the learning rate.

γ is the discount factor.

ϵ is the exploration probability at any given step.

Claim: There exists a training configuration C such that:

$$P_{QL}(C) > P_{SARSA}(C)$$

There exists a configuration where Q-learning's average convergence performance exceeds SARSA's.

Assumptions

CAUSAL CHAIN MODEL: The difference between the two algorithm's update rules snowballs into stark differences in performance through the following causal chain:

1. A difference in the update rule causes a difference in the Q-values
2. The maximum Q-value action for each state determines the policy π
3. The policy π determines which actions are taken by the agent and which states it then experiences
4. The current policy π and experienced states further affect how the Q-values are updated
5. Steps #1-4 repeat in a feedback loop for a set number of steps and episodes
6. The final average convergence values are averaged over T trials for average convergence score P.

Proof

Lemma #1:

SARSA's lower convergence is due to not reaching the goal state sufficiently.

From Gridworld #31:

$$P_{SARSA}(C) = 0.2$$

Since $P_{SARSA}(C) > 0$,

$$\exists i \in [0, n)[P_i > 0]$$

- Since the average performance of SARSA is greater than 0, there are at least some SARSA agents which manage to reach the goal state

Postulate: Performance results P_i from 0.7 to 1 inclusive are 'goal-seekers' and have found the path to the goal state. Those that are zero or below are 'corner-campers' who have not found the path to the goal state.

$$|P_{SARSA}(C) - 0| < |P_{SARSA}(C) - 0.7|$$

- The average performance of SARSA is closer to 0 than it is 0.7. Therefore, the majority of SARSA agents are 'corner-campers'.

Therefore, although some SARSA agents do manage to find the goal state, the majority of them are not able to. SARSA's lower convergence performance at gridworld #31 is due to not reaching the goal state sufficiently.

Lemma #2:

For every timestep, SARSA's update rule is exactly equal to QL's update rule with probability $1 - \epsilon$

First, I will prove that, when $\epsilon = 0$, SARSA's update rule is exactly idential to QL's.

QL update rule:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_{t+1} + \gamma \text{max}_a Q(s_{t+1}, a) - Q(s_t, a_t)\right)$$

SARSA update rule:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right)$$

In SARSA, when $\epsilon = 0$,

$$a_{t+1} = \text{argmax}(Q(s_t, a))$$

-The next action is always the maximum q-value action when there is no exploration probability. Substituting this value into the argument of the Q function:

$$Q(s_t, a_t) = Q(s_t, \text{argmax}(Q(s_t, a)))$$

-Based on the definition of argmax (returning index of maximum value) and max (returning maximum value), this can be simplified as follows:

$$Q(s_t, \text{argmax}(Q(s_t, a))) = \text{max}_a Q(s_{t+1}, a)$$

-Therefore:

$$Q(s_t, a_t) = \text{max}_a Q(s_{t+1}, a)$$

- Substituting this into the SARSA expression gives:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_{t+1} + \gamma \text{max}_a Q(s_{t+1}, a) - Q(s_t, a_t)\right)$$

which is identical to QL's expression. Therefore, when $\epsilon = 0$, SARSA's update rule equals QL's. This can then be extended to say that SARSA's update rule equals QL's with probability $1 - \epsilon$

Corollary:

$$\lim_{n \to \infty} \frac{1}{n} \sum_{n=0}^{\infty} Q(s_t, a_t) = \mathbb{E}\left[\sum_a \pi(S_{t+1}, a) Q(s_{t+1}, a)\right]$$

-When $\epsilon = 0.1$, SARSA's boostrap expression, on average, approaches the maximum q-value, discounted by 0.925, plus the 3 other possible actions each discounted by 0.25.

Lemma #3:

Insufficient exploration is the cause of SARSA not reaching the goal state.

Consider Gridworld #31:

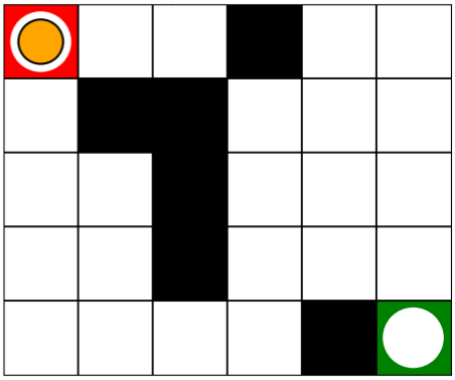

Figure 1: Gridworld #31

Feeding the SARSA's expected expression

$$\sum_{a} \pi(S_{t+1}, a) Q(s_{t+1}, a)$$

into the Value-Iteration algorithm gives us the following optimal policy:

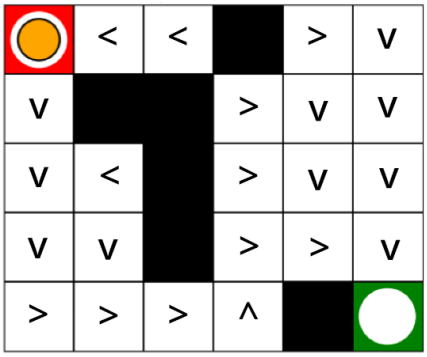

Figure 2: Gridworld #31's optimal policy using SARSA's expected boostrap expression

-A direct path from start to goal state is observed. -However, as stated in Lemma #1, the majority of SARSA agents did not obtain this policy.

-In Temporal-Differencing algorithms, convergence to the optimal* policy is guaranteed if sufficient exploration is done by the agent.

*optimal in this case means treating ϵ probability exploration as a transition probability part of the environment itself

Let a be a SARSA agent of one of the training runs for #31
Let E(a) mean that sufficient exploration was conducted during the training run
Let C(a) mean that the agent converged to the optimal policy
Let G(a) mean that the agent is a goal-seeker and finds the goal

1. $E(a) \implies C(a)$ [Premise]
2. $C(a) \implies G(a)$ [Premise]
3. $!G(a)$ [Premise]
4. $E(a) \implies G(a)$ [Hypothetical Syllogism on (1) and (2)]
5. $!G(a) \implies !E(a)$ [Contraposition on (4)]
6. $!E(a)$ [Modus Ponens on (3) and (5)]

- Therefore, SARSA does not explore the environment sufficiently enough in Gridworld #31, while QL does. Causal event #1 is not the sole reason for SARSA's lower convergence. It has to do with Causal event #4 as well.

Lemma #4:

Information about reward values propagate slowly through the gridworld. The speed at which they propagate is a function of their distance to the start state.

Every episode, an agent starts in the same state s_i, and follows an epsilon-greedy strategy for moving around a maze. This walking behavior can be loosely modeled as a random-walk.

Due to the random-walk principle, state visits to other states diminish in frequency based on their distance from the start.

Additionally, due to the nature of Temporal-Difference boostrapping, the Q-values of (s, a) pairs update based on neighboring (s, a) pairs only when the agent moves through those states.

Let $f(d)$ be the average frequency of visits to a state d spaces away from the start state. If we model the agent's movement as a 2D random-walk, we know that the probability distribution of the different states forms a bell-curve. $p(d)$, the average period between each visit to the state, would then be the inverse:

$$p(d) = \frac{1}{f(d)}$$

For the sake of argument, all we have to know is that, as d gets higher, the period increases dramatically.

The total number of timesteps needed for q-value information on a state d

spaces away to travel to the start state would be

$$t(d) = \sum_{n=1}^{d} [p(n)]$$

Therefore, q-values closer to the start state get propagated and accounted for much sooner than q-values farther away can.

Lemma #5:

SARSA's evaluation of going to the 'bottleneck' state in Gridworld 31 is a negative Q-value, whereas for QL it is a 0 Q-value. SARSA avoids going to the bottleneck again.

Consider Gridworld #31 again, specifically at state known as s_{27}:

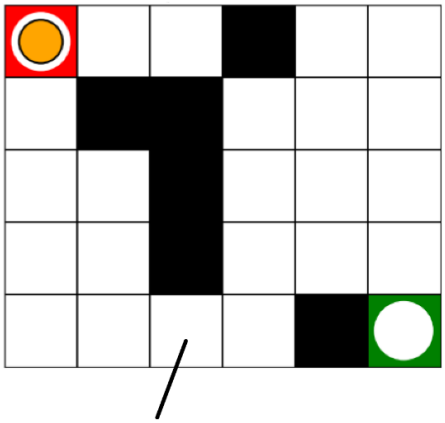

Figure 3: State s_{27} in Gridworld #31

Let's imagine a SARSA agent that is encountering this particular state for the first time. The maximum q-value action at that state is the rightward direction. The policy π would be:

$$\pi(\uparrow, s_{27}) = 2.5\%$$
$$\pi(\downarrow, s_{27}) = 2.5\%$$
$$\pi(\leftarrow, s_{27}) = 2.5\%$$
$$\pi(\rightarrow, s_{27}) = 92.5\%$$

Now imagine the same agent in the next episode, this time in the state s_{26} to the left:

$$Q(\rightarrow, s_{26}) \rightarrow \gamma \sum_{a} \pi(S_{27}, a) Q(s_{27}, a)$$

$$Q(\rightarrow, s_{26}) \rightarrow -0.025$$

and

$$Q(\leftarrow, s_{26}) \rightarrow \gamma \sum_{a} \pi(S_{25}, a) Q(s_{25}, a)$$

$$Q(\leftarrow, s_{26}) \rightarrow 0$$

Since $Q(\leftarrow, s_{26})) > Q(\rightarrow, s_{26})$, SARSA's chance of choosing right again at that state would be just 2.5%.

With QL, both evaluations would be zero due to it's use of the max function, and QL would be more able to explore the rest of the map by taking a right.

SARSA, on the other hand, stays in the left size area of the map, and does not ever discover the goal state, unless it gets several lucky explorations in a row many different times during training, which is unlikely.

It seems that QL's ignoration of the risks of epsilon-probility exploration made it explore more, leading it to gain more knowledge about the environment than SARSA ever could

QL can beat SARSA because SARSA's accounting of ϵ-probability exploration's costs prevents further exploration of the environment, causing it to converge at a suboptimal policy. □