SARSA (High Values Ex. 0.1 Alpha or smaller than .005 don’t converge easily . need to repeat)

Alpha = 0.1

Q\_sarsa = sarsa(env, 5000, .01)

Chart

Description automatically generatedEstimated Optimal Policy (UP = 0, RIGHT = 1, DOWN = 2, LEFT = 3, N/A = -1):

[[ 2 3 0 1 3 2 3 1 0 3 1 2]

[ 1 1 1 2 1 1 2 0 1 2 1 2]

[ 1 1 1 1 1 1 1 1 1 1 1 2]

[ 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]]

Table

Description automatically generated

**Q\_sarsa = sarsa(env, 5000, .02)**

Chart

Description automatically generatedEstimated Optimal Policy (UP = 0, RIGHT = 1, DOWN = 2, LEFT = 3, N/A = -1):

[[ 3 1 1 1 1 2 1 2 2 2 3 2]

[ 0 2 1 1 2 1 1 1 1 2 2 2]

[ 1 1 1 1 1 1 1 1 1 1 1 2]

[ 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]]

Table

Description automatically generated

**Q\_sarsa = sarsa(env, 5000, .05)**

Chart

Description automatically generatedEstimated Optimal Policy (UP = 0, RIGHT = 1, DOWN = 2, LEFT = 3, N/A = -1):

[[ 0 2 1 1 1 1 2 2 1 2 1 2]

[ 1 1 1 2 1 2 2 1 0 1 2 2]

[ 1 1 1 1 1 1 1 1 1 1 1 2]

[ 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]]

Table

Description automatically generated

**Chart, line chart

Description automatically generatedQ\_sarsa = sarsa(env, 5000, .001) (EXERCISE DON’T CONVERGE)**

Estimated Optimal Policy (UP = 0, RIGHT = 1, DOWN = 2, LEFT = 3, N/A = -1):

[[ 1 2 0 3 2 3 1 2 2 3 3 1]

[ 0 0 3 3 3 1 0 2 1 1 1 3]

[ 3 0 3 1 3 0 0 1 1 1 0 2]

[ 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]]

**Table

Description automatically generated**

**Q\_sarsa = sarsa(env, 5000, .2) (EXERCISE DON’T CONVERGE)**

**A picture containing chart

Description automatically generated**Estimated Optimal Policy (UP = 0, RIGHT = 1, DOWN = 2, LEFT = 3, N/A = -1):

[[ 1 1 1 1 1 1 1 1 1 1 2 2]

[ 1 1 1 1 1 1 1 1 1 1 1 2]

[ 0 0 0 0 1 0 0 0 0 1 1 2]

[ 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]]

**Table

Description automatically generated**

**Q-Learning**

**Q\_sarsamax = q\_learning(env, 5000, .01)**

**A picture containing chart

Description automatically generated**

Estimated Optimal Policy (UP = 0, RIGHT = 1, DOWN = 2, LEFT = 3, N/A = -1):

[[ 1 0 1 1 2 1 2 1 0 1 1 2]

[ 1 1 2 1 0 1 1 1 1 2 1 2]

[ 1 1 1 1 1 1 1 1 1 1 1 2]

[ 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0]]

**Table

Description automatically generated**

**Q\_sarsamax = q\_learning(env, 5000, .02)**

**A picture containing chart

Description automatically generated**

[[ 0 1 2 2 3 1 1 2 3 3 2 2]

[ 0 2 0 1 1 2 1 1 0 2 1 2]

[ 1 1 1 1 1 1 1 1 1 1 1 2]

[ 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0]]

**Table

Description automatically generated**

**Q\_sarsamax = q\_learning(env, 5000, .05)**

**A picture containing chart

Description automatically generated**

Estimated Optimal Policy (UP = 0, RIGHT = 1, DOWN = 2, LEFT = 3, N/A = -1):

[[ 0 1 1 1 1 0 3 0 1 3 0 2]

[ 1 1 1 1 1 1 1 1 1 1 1 2]

[ 1 1 1 1 1 1 1 1 1 1 1 2]

[ 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0]]

**Table

Description automatically generated**

**Q\_sarsamax = q\_learning(env, 5000, .001) (EXERCISE DON’T CONVERGE)**

**Chart

Description automatically generated**

Estimated Optimal Policy (UP = 0, RIGHT = 1, DOWN = 2, LEFT = 3, N/A = -1):

[[ 1 2 2 0 1 2 2 2 0 3 3 0]

[ 3 0 0 1 1 0 3 0 2 3 3 2]

[ 1 0 1 0 1 3 1 0 1 0 3 2]

[ 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0]]

**Table

Description automatically generated**

**Q\_sarsamax = q\_learning(env, 5000, .2) (EXERCISE DON’T CONVERGE)**

**Chart

Description automatically generated with medium confidence**

Estimated Optimal Policy (UP = 0, RIGHT = 1, DOWN = 2, LEFT = 3, N/A = -1):

[[ 0 2 0 2 1 1 1 1 2 2 1 2]

[ 1 1 0 1 1 1 0 0 1 1 1 2]

[ 1 1 1 1 1 1 1 1 1 1 1 2]

[ 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0]]

**Table

Description automatically generated**

**TD Control: Expected Sarsa**

**Chart

Description automatically generatedQ\_expsarsa = expected\_sarsa(env, 10000, 1)**

Estimated Optimal Policy (UP = 0, RIGHT = 1, DOWN = 2, LEFT = 3, N/A = -1):

[[ 1 0 1 1 1 1 1 1 1 0 1 2]

[ 1 1 1 1 1 1 1 1 1 1 1 2]

[ 1 1 1 1 1 1 1 1 1 1 1 2]

[ 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0]]

**Table

Description automatically generated**

**Q\_expsarsa = expected\_sarsa(env, 10000, 0.01)**

**Chart

Description automatically generated with medium confidence**

Estimated Optimal Policy (UP = 0, RIGHT = 1, DOWN = 2, LEFT = 3, N/A = -1):

[[ 0 1 1 1 2 0 2 2 2 1 1 2]

[ 0 1 0 1 1 3 1 1 1 1 1 2]

[ 1 1 1 1 1 1 1 1 1 1 1 2]

[ 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0]]

**Table

Description automatically generated**

Q\_expsarsa = expected\_sarsa(env, 10000, 0.01)

Chart

Description automatically generated

Estimated Optimal Policy (UP = 0, RIGHT = 1, DOWN = 2, LEFT = 3, N/A = -1):

[[ 1 2 0 1 1 1 1 2 0 1 2 2]

[ 0 2 1 1 1 2 1 1 1 1 1 2]

[ 1 1 1 1 1 1 1 1 1 1 1 2]

[ 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0]]

**Table

Description automatically generated**

**Q\_expsarsa = expected\_sarsa(env, 10000, 0.001) (EXERCISE DON’T CONVERGE)**

**Chart

Description automatically generated**

Estimated Optimal Policy (UP = 0, RIGHT = 1, DOWN = 2, LEFT = 3, N/A = -1):

[[ 1 3 1 1 1 1 3 3 1 1 0 2]

[ 0 0 3 3 0 2 3 2 1 1 2 2]

[ 1 1 1 0 0 1 1 1 1 1 1 2]

[ 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0]]

**Table

Description automatically generated**

# Analysing Performance

You've learned about three different TD control methods in this lesson. So, what do they have in common, and how are they different?

## Similarities

All of the TD control methods we have examined (Sarsa, Sarsamax, Expected Sarsa) converge to the optimal action-value function q\_\**q*∗​ (and so yield the optimal policy \pi\_\**π*∗​) if:

1. the value of \epsilon*ϵ* decays in accordance with the GLIE conditions, and
2. the step-size parameter \alpha*α* is sufficiently small.

## Differences

The differences between these algorithms are summarized below:

* Sarsa and Expected Sarsa are both **on-policy** TD control algorithms. In this case, the same (\epsilon*ϵ*-greedy) policy that is evaluated and improved is also used to select actions.
* Sarsamax is an **off-policy** method, where the (greedy) policy that is evaluated and improved is different from the (\epsilon*ϵ*-greedy) policy that is used to select actions.
* On-policy TD control methods (like Expected Sarsa and Sarsa) have better online performance than off-policy TD control methods (like Sarsamax).
* Expected Sarsa generally achieves better performance than Sarsa.

If you would like to learn more, you are encouraged to read Chapter 6 of the [**textbook**](http://go.udacity.com/rl-textbook) (especially sections 6.4-6.6).

### Temporal-Difference Methods

* Whereas Monte Carlo (MC) prediction methods must wait until the end of an episode to update the value function estimate, temporal-difference (TD) methods update the value function after every time step.

### TD Control

* **Sarsa(0)** (or **Sarsa**) is an on-policy TD control method. It is guaranteed to converge to the optimal action-value function q\_\**q*∗​, as long as the step-size parameter \alpha*α* is sufficiently small and \epsilon*ϵ* is chosen to satisfy the **Greedy in the Limit with Infinite Exploration (GLIE)** conditions.
* **Sarsamax** (or **Q-Learning**) is an off-policy TD control method. It is guaranteed to converge to the optimal action value function q\_\**q*∗​, under the same conditions that guarantee convergence of the Sarsa control algorithm.
* **Expected Sarsa** is an on-policy TD control method. It is guaranteed to converge to the optimal action value function q\_\**q*∗​, under the same conditions that guarantee convergence of Sarsa and Sarsamax.

### Analyzing Performance

* On-policy TD control methods (like Expected Sarsa and Sarsa) have better online performance than off-policy TD control methods (like Q-learning).
* Expected Sarsa generally achieves better performance than Sarsa.