**Analyzing Performance**

All of the TD control algorithms 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.

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).

As an optional exercise to deepen your understanding, you are encouraged to reproduce Figure 6.4. (Note that this exercise is optional!)

The figure shows the performance of Sarsa and Q-learning on the cliff walking environment for constant \epsilon = 0.1*ϵ*=0.1. As described in the textbook, in this case,

* Q-learning achieves worse online performance (where the agent collects less reward on average in each episode), but learns the optimal policy, and
* Sarsa achieves better online performance, but learns a sub-optimal "safe" policy.

You should be able to reproduce the figure by making only small modifications to your existing code.

# **Summary**

**[The cliff-walking task (Sutton and Barto, 2017)](https://classroom.udacity.com/nanodegrees/nd009t/parts/ac12e0fe-e54e-40d5-b0f8-136dbdd1987b/modules/f87db1ea-a332-4007-9f37-5e641d80c92a/lessons/d2de57a0-cd89-40bd-b87f-ec0298b425cf/concepts/7d2dafe6-e522-4a8d-beb0-e9dd6eadddfc)**

**TD Prediction: TD(0)**

* 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.
* For any fixed policy, **one-step TD** (or **TD(0)**) is guaranteed to converge to the true state-value function, as long as the step-size parameter \alpha*α* is sufficiently small.
* In practice, TD prediction converges faster than MC prediction.

**TD Prediction: Action Values**

* (In this concept, we discussed a TD prediction algorithm for estimating action values. Similar to TD(0), this algorithm is guaranteed to converge to the true action-value function, as long as the step-size parameter \alpha*α* is sufficiently small.)

**TD Control: Sarsa(0)**

* **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.

**TD Control: Sarsamax**

* **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.

**TD Control: Expected Sarsa**

* **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.