Recall! Policy Iteration

General policy improvement framework:

1. Start with a policy $\pi$
2. Learn $Q_\pi$
3. Improve $\pi$
   a. $\pi(s) = \max_a Q(s, a), \forall s$

Repeat

This is known as **policy iteration**.
It is guaranteed to converge to the optimal policy.

Steps 2 and 3 can be interleaved as rapidly as you like.
Usually, perform 3a every *time step*.
Sarsa

Sarsa: very simple algorithm

1. Initialize $Q(s, a)$
2. For $n$ episodes
   - observe transition $(s, a, r, s', a')$
   - compute TD error $\delta = r + \gamma Q(s', a') - Q(s, a)$
   - update $Q$: $Q(s, a) = Q(s, a) + \alpha \delta$
   - select and execute action based on $Q$
Sarsa Demo ...
Q-Learning

Alternative to Sarsa

- Don't use the transition you experienced
- Use the greedy transition

\[ Q(s, a) = Q(s, a) + \alpha \left[ Q(s, a) - (r + \gamma \max_{a'} Q(s', a')) \right] \]
Q-Learning

1. Initialize $Q(s, a)$
2. For $n$ episodes
   - observe transition $(s, a, r, s')$
   - compute TD error $\delta = r + \gamma \max_{a'} Q(s', a') - Q(s, a)$
   - update $Q$: $Q(s, a) = Q(s, a) + \alpha \delta$
   - select and execute action based on $Q$
Off-Policy

This is off-policy:

• Learning $Q$ for a policy you are not executing.
• Why might you want to do this?

Example: \textit{epsilon greedy up to a point, then you switch \textit{epsilon} off.}

Off policy algorithms allow you to use one policy to gather samples, and learn $V/Q$ for another policy.
Off-Policy

Why might you not want to do this …
Recall: TD($\lambda$)

Weighted sum:

\[
R^{(1)} = r_0 + \gamma V(s_1) \\
R^{(2)} = r_0 + \gamma r_1 + \gamma^2 V(s_2) \\
\cdots \\
R^{(n)} = \sum_{i=0}^{n-1} \gamma^i r_i + \gamma^n V(s_n)
\]

Estimator:

\[
R^\lambda_{st} = (1 - \lambda) \sum_{n=0}^{\infty} \lambda^n R^{(n+1)}_{st}
\]
**TD(\(\lambda\)): Implementation**

Each state has eligibility trace \(e(s)\).

At time \(t\):
\[
e(s_t) = 1 \quad (\text{replacing traces})
\]
\[
e(s) = \gamma \lambda e(s), \text{ for all other } s.
\]

When updating:
- Compute \(\delta\) as before
- \(Q(s, a) = Q(s, a) + \alpha \delta e(s)\)
Sarsa(\(\lambda\)) Demo ...