• How do you make sure you explore all states if using model-free approaches?

• How do you get your user simulation to be in an arbitrary state when it only knows how to advance from one state to another following interaction with the system?

- Actually, you only have to explore over possible states

- Impossible to have low confidence for a slot when the slot is not filled in

- Only really possible if using model-based approach

To make sure all state-action pairs get Q values, Levin started a dialogue

Exploring Starts

Exploring Starts

Overview

SARSA and Q-Learning

Alpha Factor

Epochs
Epsilon-Greedy

- Every once in a while, pick an action other than policy action
  - With probability $\epsilon$, pick the non-policy action
- Problem: might pick several random actions in dialogue
- Only after last random action do we strictly follow policy
- But then it is difficult to cover all state-action pairs
- So, should only update till that point
- Instead with Monte Carlo
- Typically lower epsilon as training progresses
  - With probability $\epsilon$, pick the non-policy action
  - Must start dialogue simulations at the beginning
  - Problem: initially pick several random actions in dialogue
- Every once in a while, pick an action other than policy action
- But how do we get $Q$ scores for all state-action pairs for the current policy?
Levin updated the policy after each sweep through the state.

Don't throw away $Q$ and counts.

Instead, use epochs of 100 runs or so, and update the policy.

Not possible anymore with Epsilon Greedy.
- She also reset the $Q$ and counts.
- Action pairs.

Epochs
But, after each run through all the state action pairs, she might reset the 

\[ Q(s, a) = C(s, a) \]

Each time to update the policy, you want to use your previous 

\[ Q(s, a) = C(s, a) \]

But, the policy is improving estimates, rather than learn from scratch. 

Each time to update the policy, you want to use your previous 

\[ Q(s, a) = C(s, a) \]

How much to remember past experiences, perhaps with an older policy 

\[ Q(s, a) = C(s, a) \]

But, the policy is improving older experiences are based on bad policies 

\[ Q(s, a) = C(s, a) \]

Each time to update the policy, you want to use your previous 

\[ Q(s, a) = C(s, a) \]

I-\[ \]

\[ Q(s, a) = C(s, a) \]

How much to remember past experiences, perhaps with an older policy 

\[ Q(s, a) = C(s, a) \]

But, the policy is improving older experiences are based on bad policies 

\[ Q(s, a) = C(s, a) \]
Overview

SARSA and Q-Learning

- Alpha Factor
- Epochs
- Exploring States

Alternatives

- Weight each experience at a constant value
  \( \alpha = \frac{1}{n(s,a)} \)
  - Due to randomness, might go into a suboptimal policy
  - Averaging over the last 500 might not be enough
  - Also, is 500 large enough?
  - But, still weight the first 500 equally
  \( \alpha = \frac{1}{n(s,a)} \)

- Weight each experience at a constant value
  \( \alpha = \frac{1}{500} \)
  - But, too much weight given to the initial value

- Weight each experience at a constant value
SARSA and Q-Learning

- SARSA and Q-learning both bootstrap estimates of the Q score of the following state.
- SARSA and Q-learning both bootstrap estimates of the Q score of the following state.
- SARSA and Q-learning both bootstrap estimates of the Q score of the following state.

More sensitive to MDP violations
- SARSA and Q-learning both bootstrap estimates of the Q score of the following state.
- SARSA and Q-learning both bootstrap estimates of the Q score of the following state.
- SARSA and Q-learning both bootstrap estimates of the Q score of the following state.

With Q-learning (and SARSA), don't need to lower epsilon
- SARSA and Q-learning both bootstrap estimates of the Q score of the following state.
- SARSA and Q-learning both bootstrap estimates of the Q score of the following state.
- SARSA and Q-learning both bootstrap estimates of the Q score of the following state.

Updating Q values
- SARSA and Q-learning both bootstrap estimates of the Q score of the following state.
- SARSA and Q-learning both bootstrap estimates of the Q score of the following state.
- SARSA and Q-learning both bootstrap estimates of the Q score of the following state.

Monte-Carlo:
- SARSA and Q-learning both bootstrap estimates of the Q score of the following state.
- SARSA and Q-learning both bootstrap estimates of the Q score of the following state.
- SARSA and Q-learning both bootstrap estimates of the Q score of the following state.

\[
Q(s, a) = (c(s, a) + Q(s', a')) \alpha + Q(s, a) (1 - \alpha)
\]

\[
Q(s', a') = (c(s', a') + Q(s', a')) \alpha + Q(s', a') (1 - \alpha)
\]

\[
Q(s', a') = (c(s', a') + Q(s', a')) \alpha + Q(s', a') (1 - \alpha)
\]