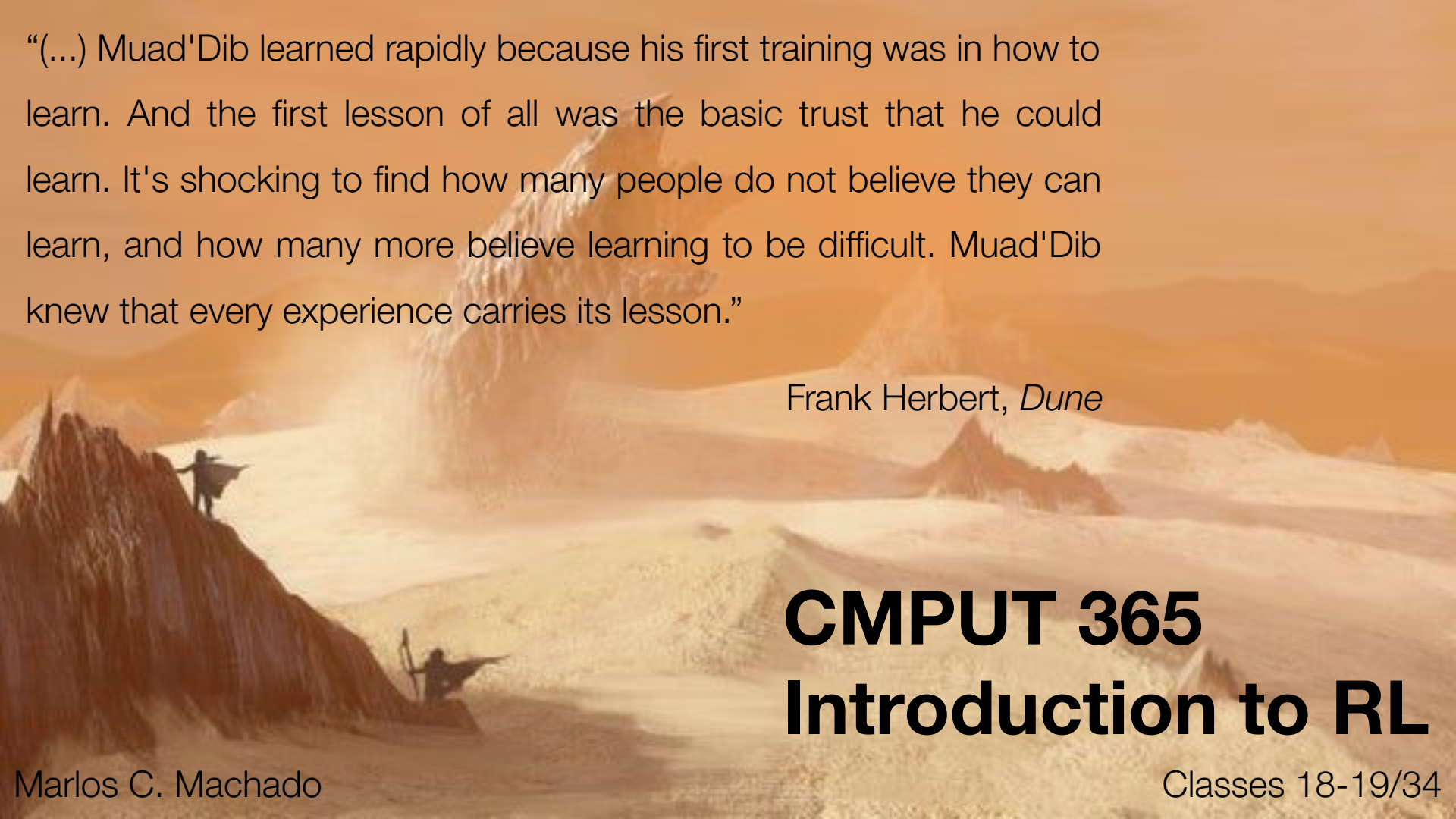

“(...) Muad'Dib learned rapidly because his first training was in how to learn. And the first lesson of all was the basic trust that he could learn. It's shocking to find how many people do not believe they can learn, and how many more believe learning to be difficult. Muad'Dib knew that every experience carries its lesson.”
Frank Herbert, *Dune*
CMPUT 365
Introduction to RL
Marlos C. Machado
Classes 18-19/34

Coursera Reminder

You **should be enrolled in the private session** we created in Coursera for CMPUT 365.

I **cannot** use marks from the public repository for your course marks. You **need** to **check**, **every time**, if you are in the private session and if you are submitting quizzes and assignments to the private section.

At the end of the term, I **will not port grades** from the public session in Coursera.

If you have any questions or concerns, **talk with the TAs** or email us `cmput365@ualberta.ca`.

Reminders and Notes

- Exam viewing:
  - ~~Mon 16:30 - 18:30 @ UCOMM 3-136~~
  - Thur 10:00 - 12:00 @ UCOMM 3-136
- What I plan to do today:
  - TD Learning for Control (Second half of Chapter 6 of the textbook).
- Practice quiz and programming assignment are due on Friday.
- The midterm is *Friday, Oct 30.*

Please, interrupt me at any time!



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Reminder

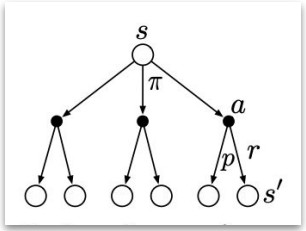
Temporal-difference Learning – Why?

“If one had to identify one idea as central and novel to reinforcement learning, it would undoubtedly be temporal-difference (TD) learning.”

Last “Week”: Prediction with Temporal-difference Learning

- Dynamic programming update:

$$\begin{aligned} v_{k+1}(s) &\doteq \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r \,|\, s, a) \Big[r + \gamma v_k(s') \Big] \end{aligned}$$



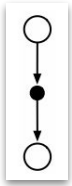
- Monte Carlo update:

$$V(S_t) \leftarrow V(S_t) + \alpha \Big[G_t - V(S_t) \Big]$$



- TD update:

$$V(S_t) \leftarrow V(S_t) + \alpha \Big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \Big]$$



Control

Control with Temporal-difference Learning – Why?

The ultimate goal of reinforcement learning algorithms is to learn how to maximize rewards ¯_(ツ)_/¯

… and in AI courses, when people have to choose a single thing to teach students, they teach them Q–Learning!

Many high-profile stories in RL are due to control with TD

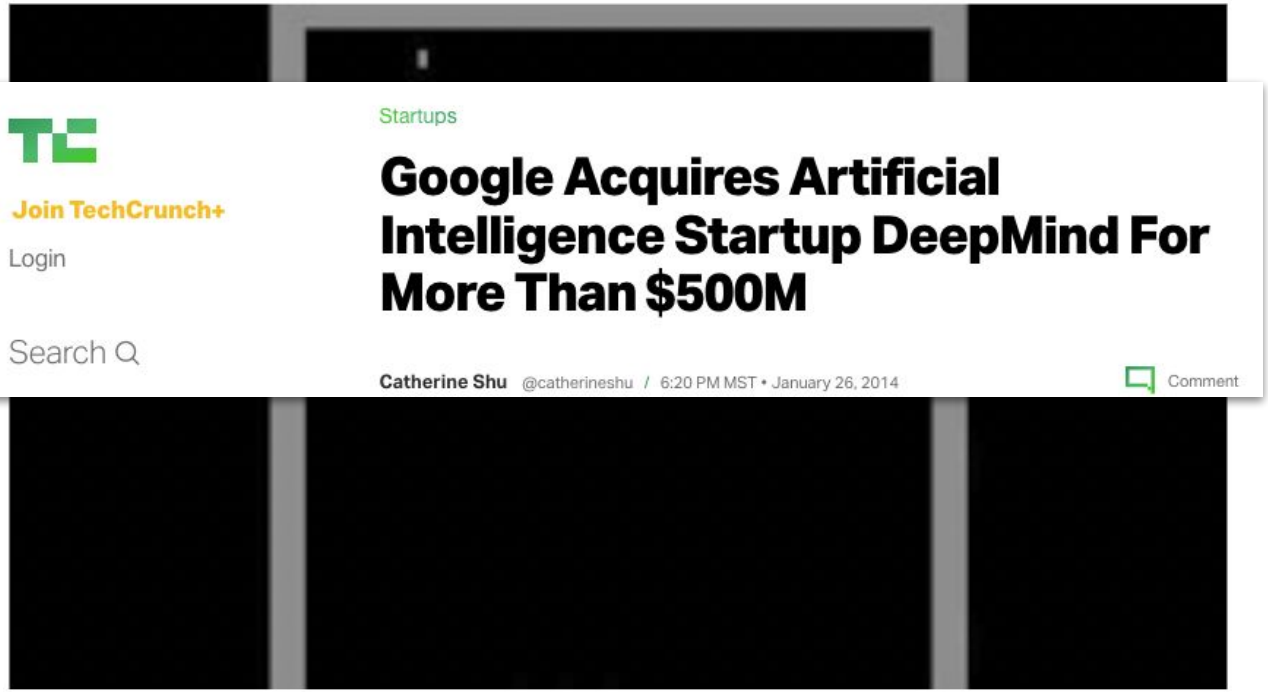




Human-level control through deep reinforcement learning

Volodymyr Mnih, Koray Kavukcuoglu ✉, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare,

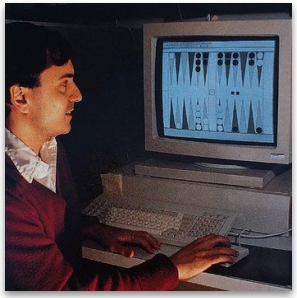
Many high-profile stories in RL are due to control with TD





Many high-profile stories in RL are due to control with TD

[Tesauro, 1994]



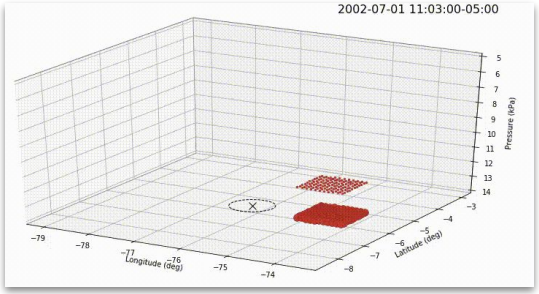
[Mnih et al., 2015]



Watch this talk in case you are interested in the history of RL and Games.

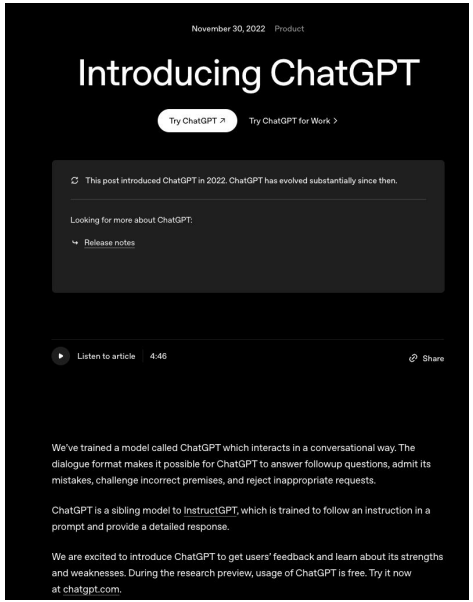


[Vinyals et al., 2019]



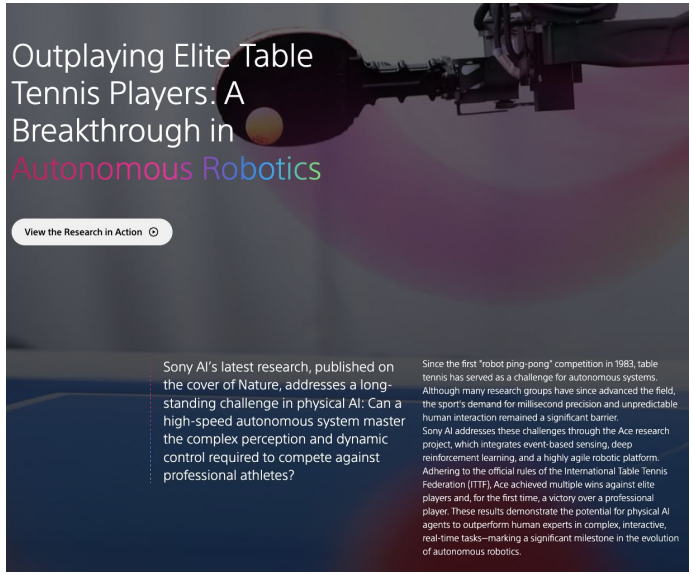
[Bellemare et al., 2020]

And this continues to be the case, of course



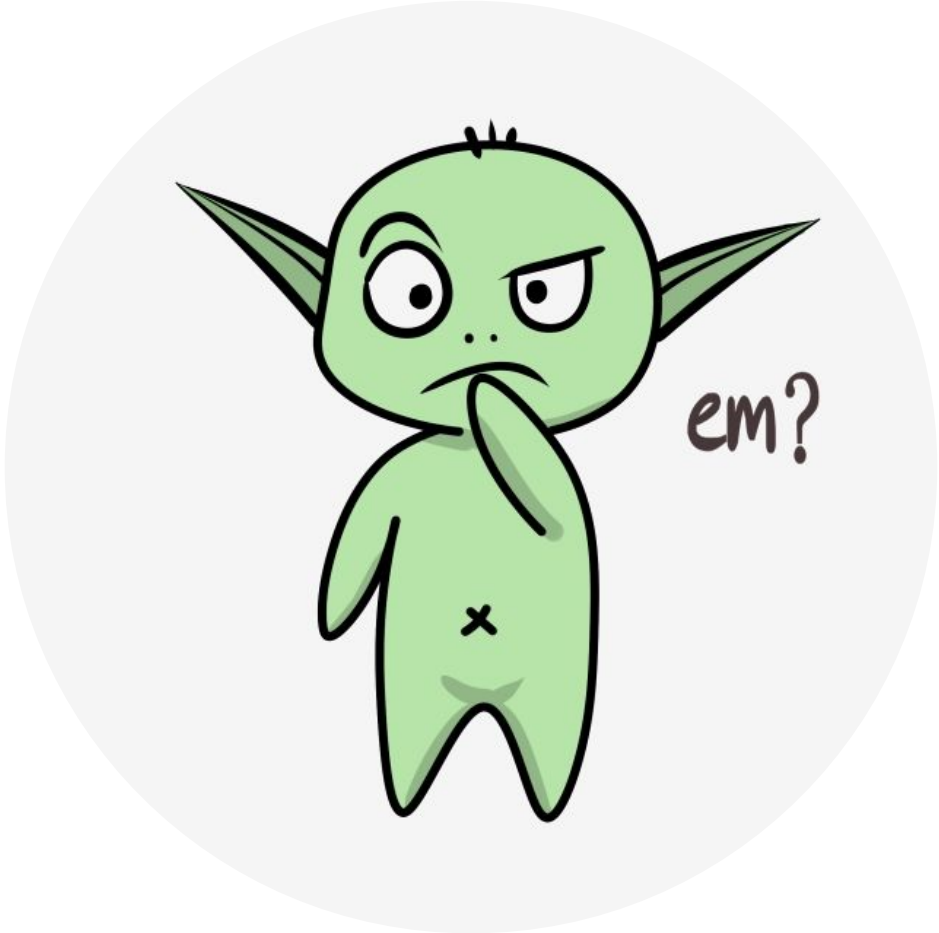
[OpenAI, 2022]

[Dürr et al., 2026]



And this continues to be the case, of course

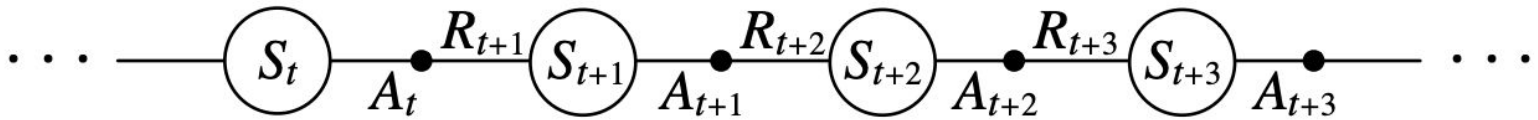




https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Sarsa: On-policy Control

- We again use generalized policy iteration (GPI), but now using TD for evaluation.
- We need to learn an action-value function instead of a state-value function. We can do this!



$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \Big]$$

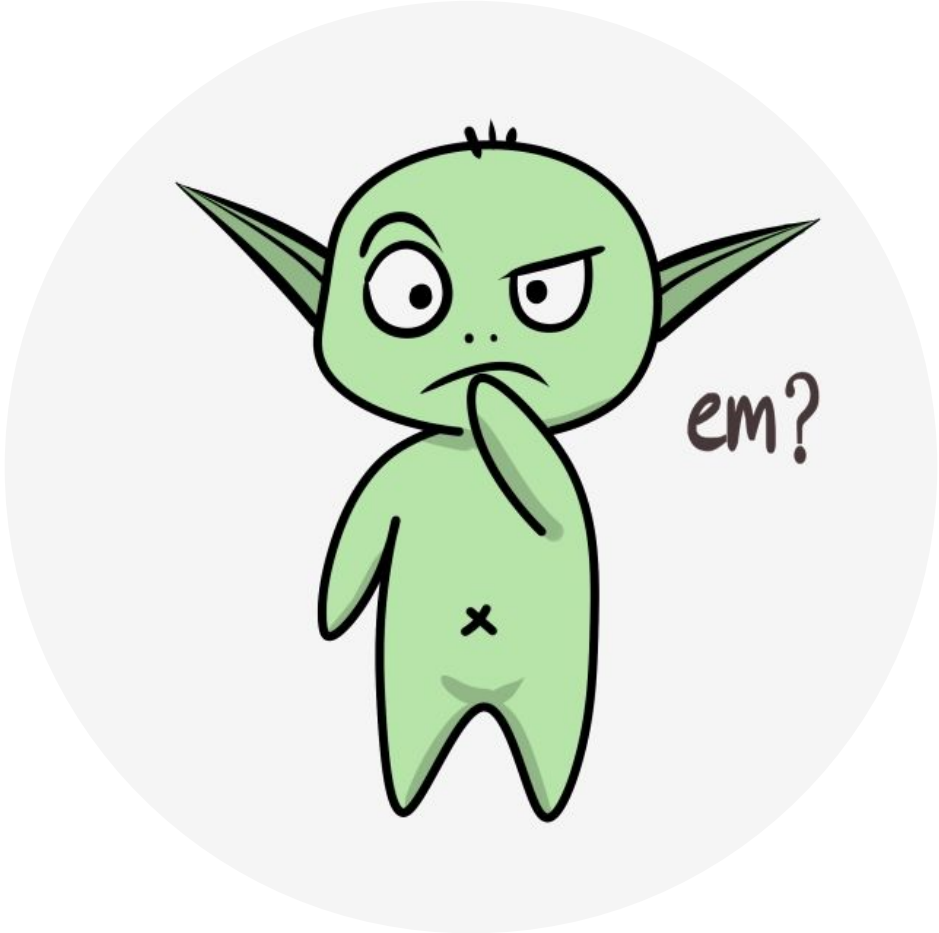
Sarsa: On-policy Control

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

We need to explore!



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Q-Learning: Off-Policy Control

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \Big]$$

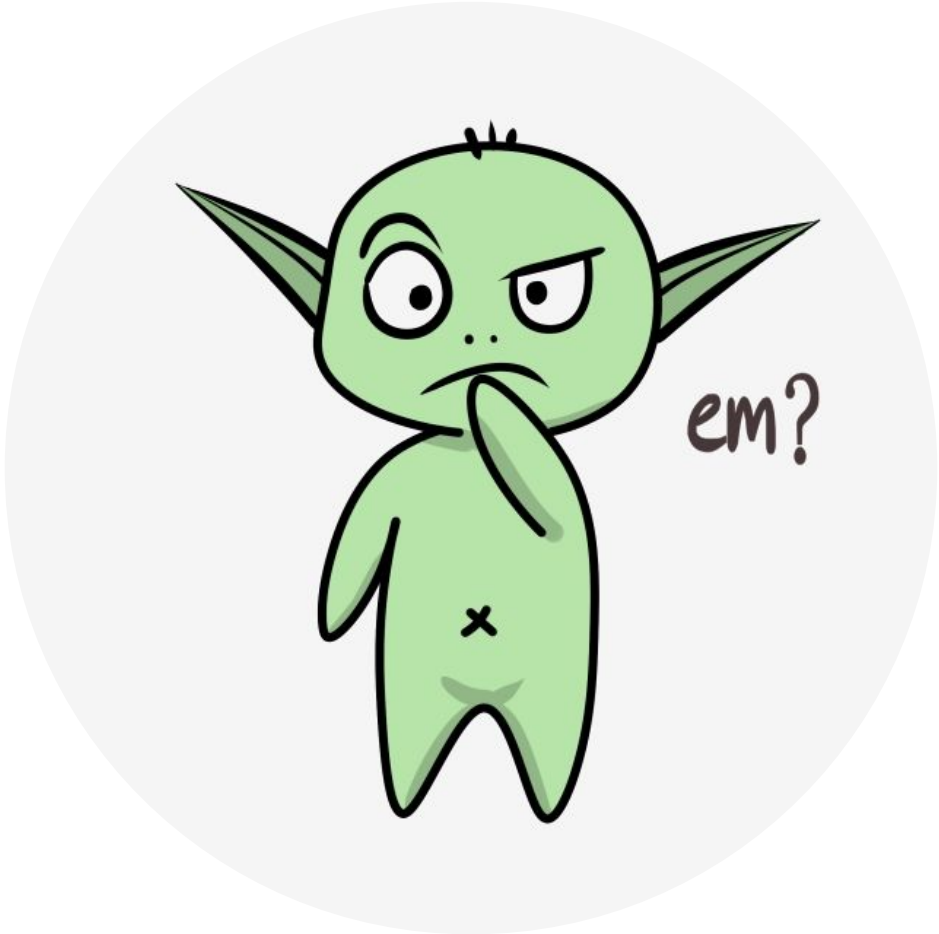
- Q directly approximates q_*, regardless of the policy being followed.
- Notice we do not need importance sampling. We are updating a state–action pair. We do not have to care how likely we were to select the action; now that we have selected it we want to learn fully from what happens.

Q-Learning: Off-Policy Control

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

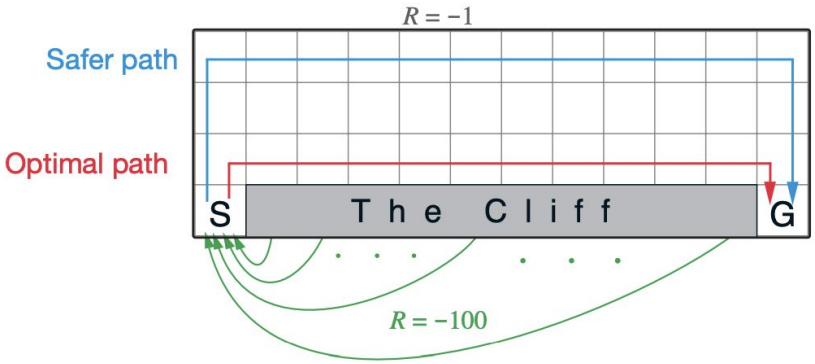
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

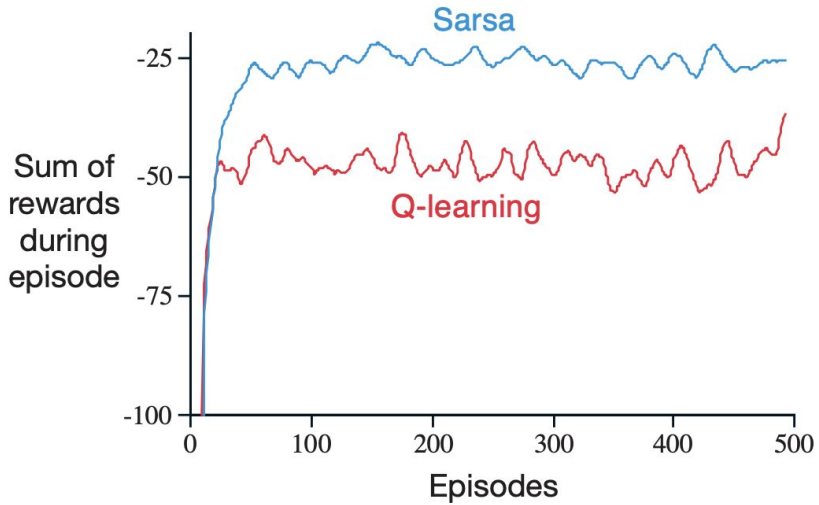
Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Example – Q-Learning vs Sarsa



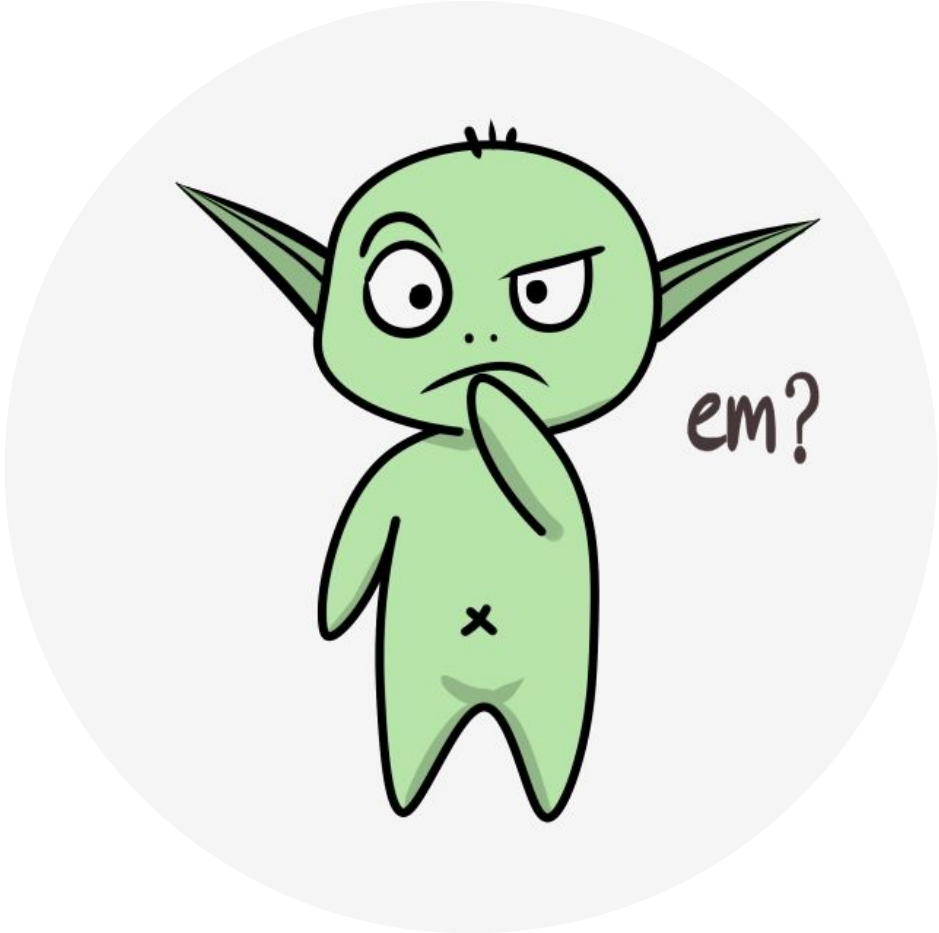


Discussion

Exercise 6.12. Suppose action selection is greedy. Is Q-learning then exactly the same algorithm as Sarsa? Will they make exactly the same action selections and weight updates?

$$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$$

$$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$$



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Expected Sarsa

What if instead of the maximum over next state-action pairs we used the expected value, taking into account how likely each action is under the current policy?

$$
\begin{aligned}
Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \mathbb{E}_\pi [Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t) \Big] \\
&= Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \Big],
\end{aligned}
$$

Expected Sarsa

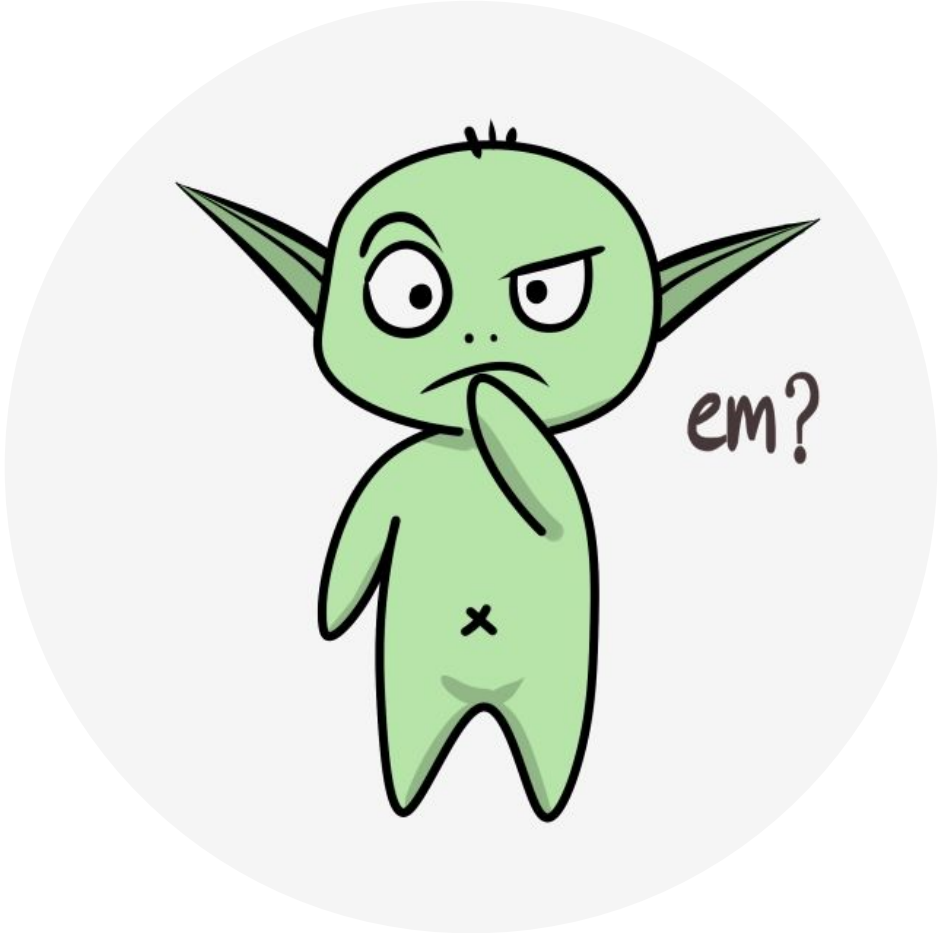
What if instead of the maximum over next state-action pairs we used the expected value, taking into account how likely each action is under the current policy?

$$
\begin{aligned}
Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \mathbb{E}_\pi [Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t) \Big] \\
&= Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \Big],
\end{aligned}
$$

Expected Sarsa is more computationally expensive than Sarsa but, in return, it eliminates the variance due to the random selection of A_{t+1}.

Is Expected Sarsa on-policy or off-policy?

Expected can use a policy different from the target policy π to generate behavior (thus, it can be off-policy; although one can use it on-policy as well).



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Maximization Bias

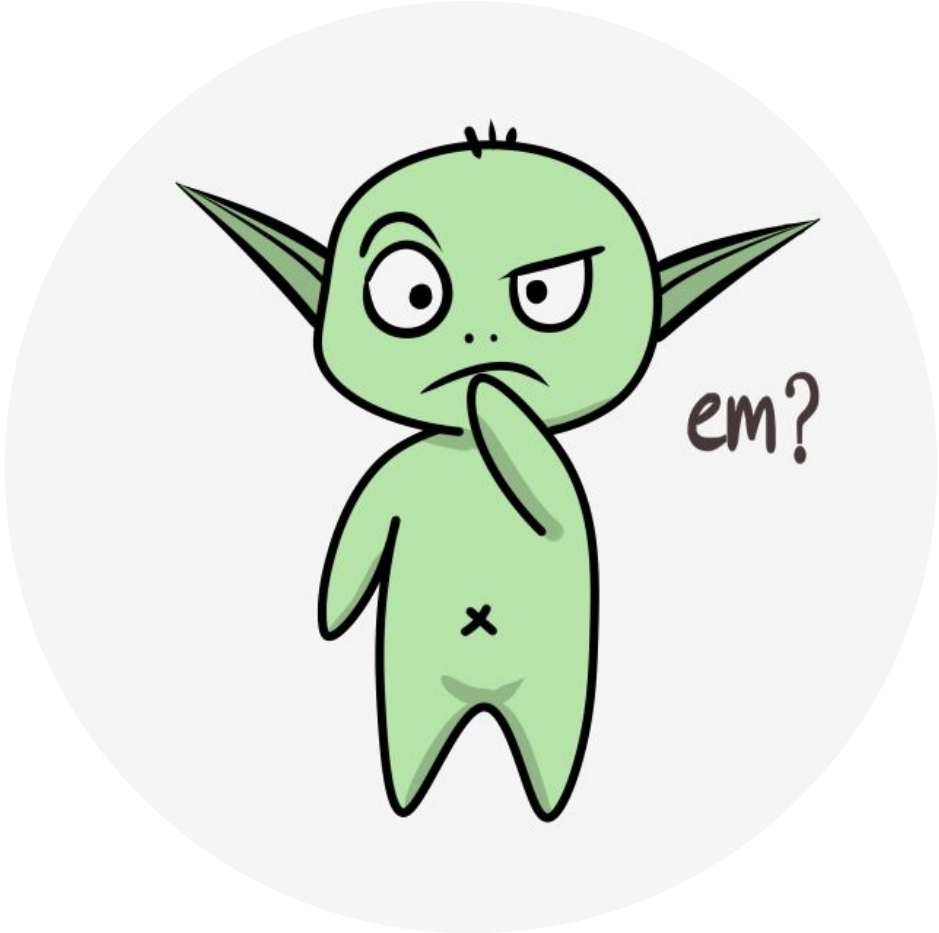
- The control algorithms we discussed so far use a maximization to get their target policies (either a max/greedy policy or an ε-greedy policy).
- *Maximization bias:* A maximum over estimated values is used implicitly as an estimate of the maximum value, which can lead to a significant positive bias.

Double Learning

- The issue is that we use the same samples to determine the maximizing action and to estimate its value.
- In Bandits:
  - Split the data, learn $Q_1(a)$ and $Q_2(a)$ to estimate $q(a)$.
  - Choose actions according to one estimate and get estimate from the other:
    $A^* = \text{argmax}_a\ Q_1(a)$ $\quad\quad$ $Q_2(A^*) = Q_2(\text{argmax}_a\ Q_1(a))$
  - This leads to unbiased estimates, that is: $\mathbb{E}[Q_2(A^*)] = q(A^*)$



$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q_2\left(S_{t+1}, \arg\max_a Q_1(S_{t+1}, a)\right) - Q_1(S_t, A_t) \right]$$



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Double Q-Learning

Double Q-learning, for estimating $Q_1 \approx Q_2 \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q_1(s, a)$ and $Q_2(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, such that $Q(\textit{terminal}, \cdot) = 0$

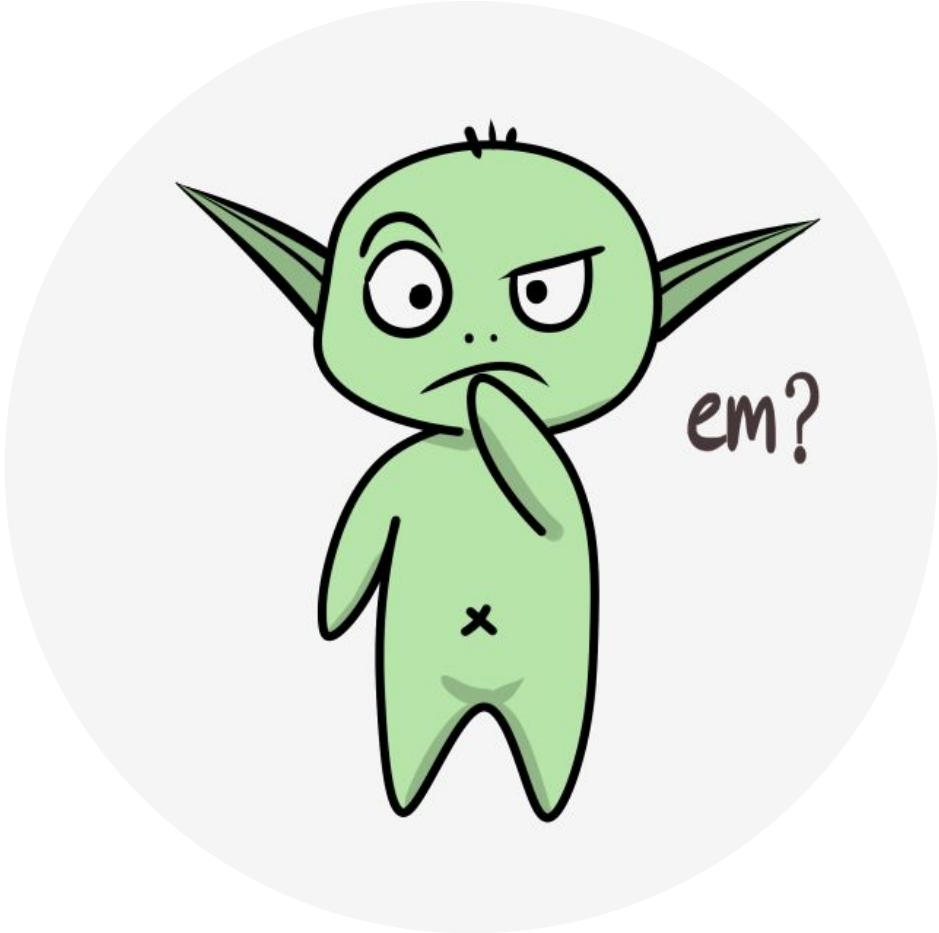
Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using the policy ε-greedy in $Q_1 + Q_2$
    Take action A, observe R, S'
    With 0.5 probabilility:

$$Q_1(S, A) \leftarrow Q_1(S, A) + \alpha\Big(R + \gamma Q_2\big(S', \operatorname{argmax}_a Q_1(S', a)\big) - Q_1(S, A)\Big)$$

    else:

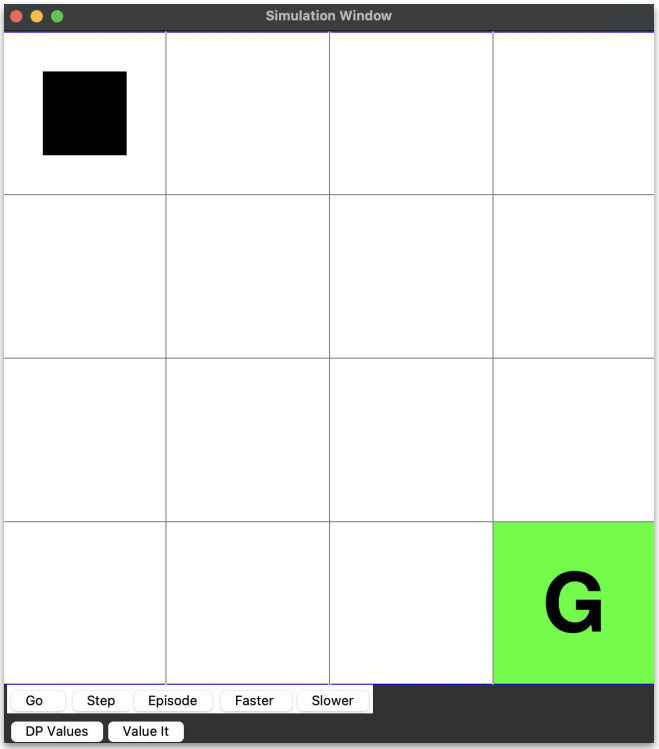
$$Q_2(S, A) \leftarrow Q_2(S, A) + \alpha\Big(R + \gamma Q_1\big(S', \operatorname{argmax}_a Q_2(S', a)\big) - Q_2(S, A)\Big)$$

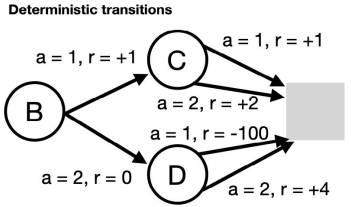
    $S \leftarrow S'$
  until S is terminal



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Demo


Simulation Window
G
Go
Step
Episode
Faster
Slower
DP Values
Value It



Exercise

Consider the following MDP, with three states B, C, and D ($\mathcal{S}$ = {B, C, D}), and 2 actions ($\mathcal{A}$ = {1, 2}), with γ = 1.0. Assume the action values are initialized Q(s, a) = 0 $\forall$ s $\in \mathcal{S}$. The agent takes actions according to an ε-greedy policy with ε = 0.1.

1. What is the optimal policy for this MDP and what are the action-values corresponding to the optimal policy: q*(s, a)?
2. Imagine the agent experienced a single episode, and the following experience: S_0 = B, A_0 = 2, R_1 = 0, S_1 = D, A_1 = 2, R_2 = 4. What are the Sarsa updates during this episode, assuming α = 0.1? Start with state B, and perform the Sarsa update, then update the value of state D.
3. Using the sample episode above, compute the updates Q-learning would make, with α = 0.1? Again, start with state B, and then state D.
4. Let's consider one more episode: S_0 = B, A_0 = 2, R_1 = 0, S_1 = D, A_1 = 1, R_2 = -100. What would the Sarsa updates be? And what would the Q-learning updates be?
5. Assume you see one more episode, and it's the same on as in 4. Once more update the action values, for Sarsa and Q-learning. What do you notice?