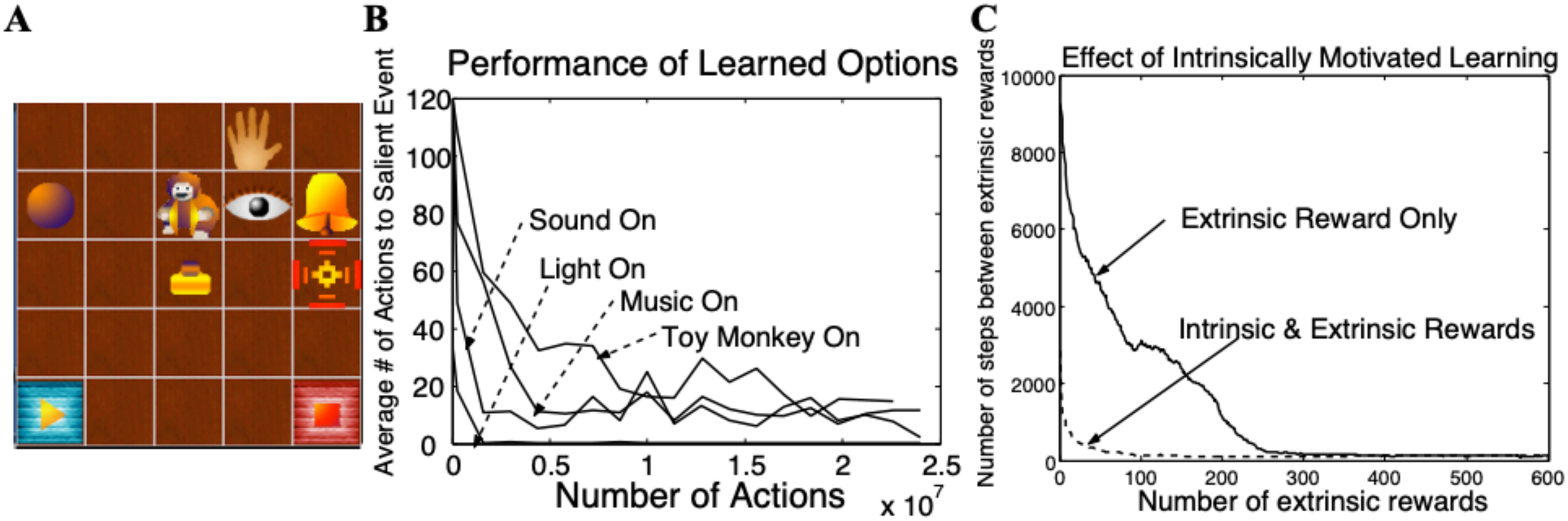
Temporal Difference Learning for Control

CMPUT 655
Fall 2022

A learning system that cares about learning tiny details

Intrinsically motivated RL



Why is Q-learning so popular?

GPI with TD(0)

- We want to find near-optimal policies: control
- Generalized policy iteration, with TD(0) taking care of the *policy evaluation* part
- Again, we will learn state-action value functions: Q(s,a)

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

Exercise

Modify the Tabular TD(0) algorithm for estimating v_π, to estimate q_π.

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\mathit{terminal}) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S'
        $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
        $S \leftarrow S'$
    until S is terminal

Sarsa: On-Policy TD Control

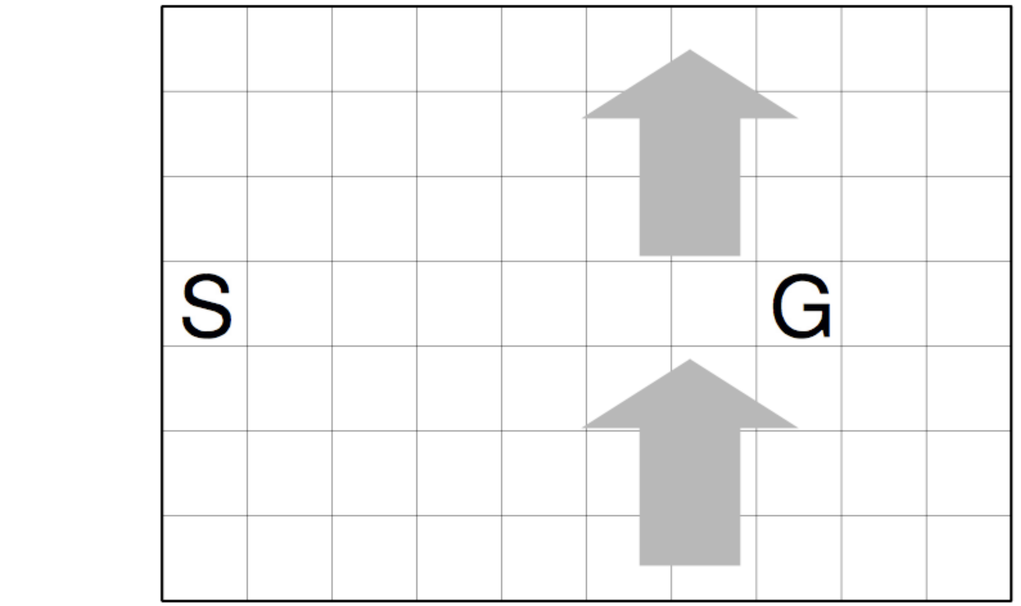
Turn this into a control method by always updating the policy to be greedy with respect to the current estimate:

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

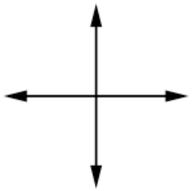
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

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

Windy Gridworld



Wind: 0 0 0 1 1 1 2 2 1 0

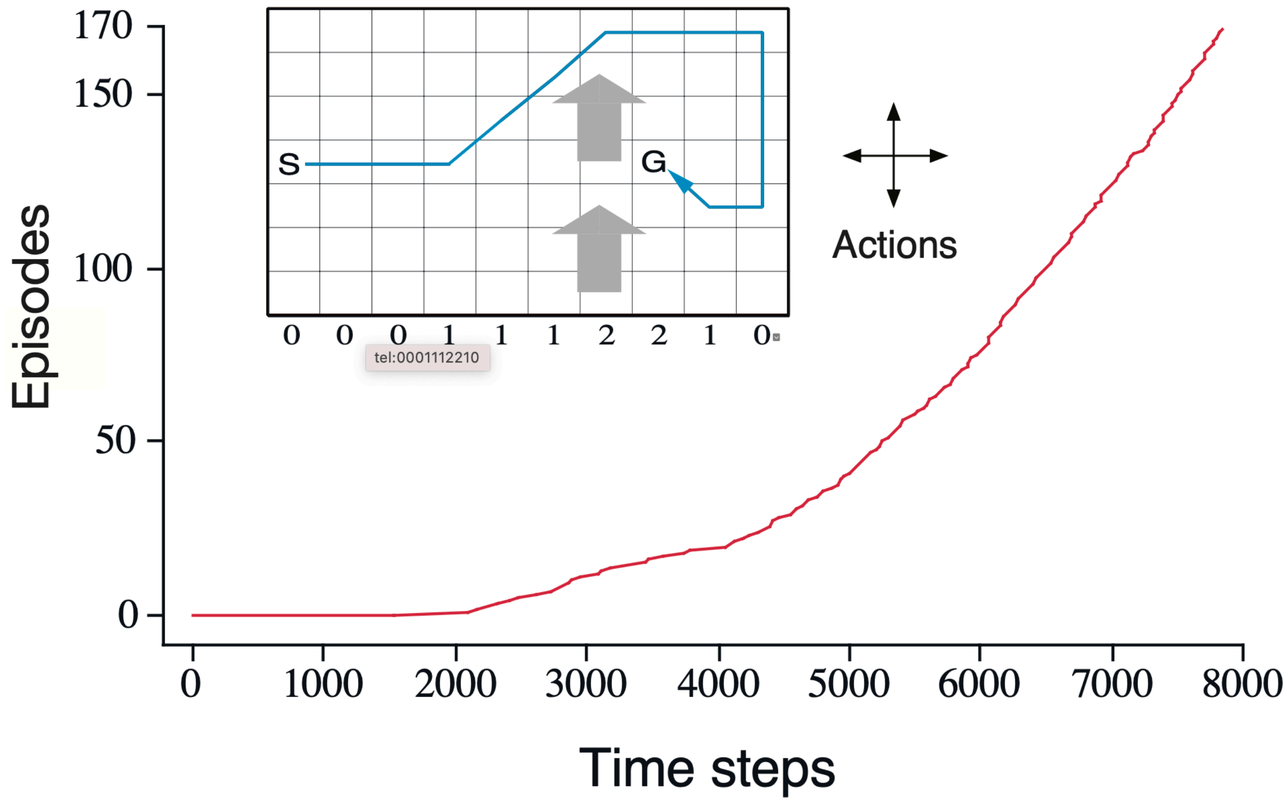


standard moves

undiscounted, episodic, reward = –1 until goal

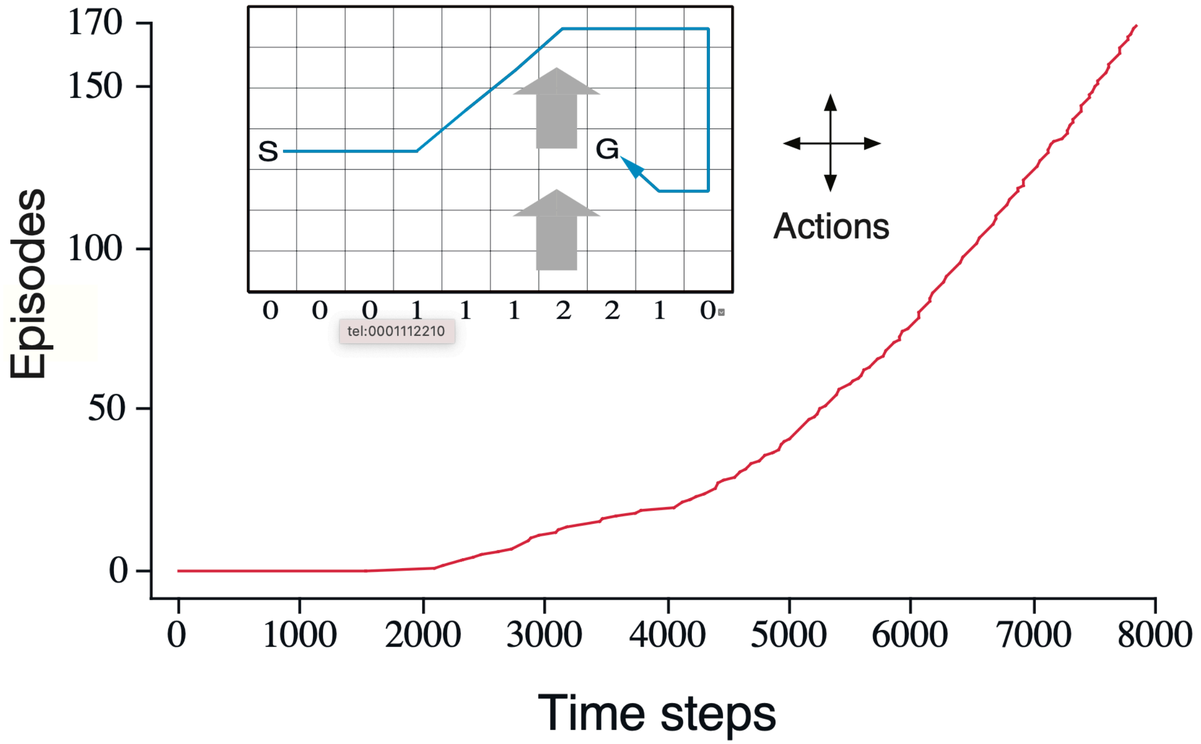
New type of learning curve

- *How can we tell that the agent is learning and getting better?*



- *What would the plot look like if we plotted steps (y-axis) per episode (x-axis)?*

- *ϵ = 0.1; Q(s,a) = 0 => optimistic!*



- *The optimal path is 15 steps, Sarsa reaches about 17. Why? MC methods are challenging to implement on this task. Why?*

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

Q-learning:

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

Q-learning

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

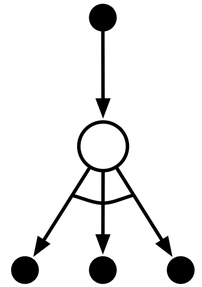
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big]$$

- Q-learning directly approximates q*, independent of the policy being followed
  - the behavior must satisfy coverage
  - Q-learning learns π_* not an ϵ-soft policy
  - Q-learning is thus off-policy

Q-Learning: Off-Policy TD Control

One-step Q-learning:

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



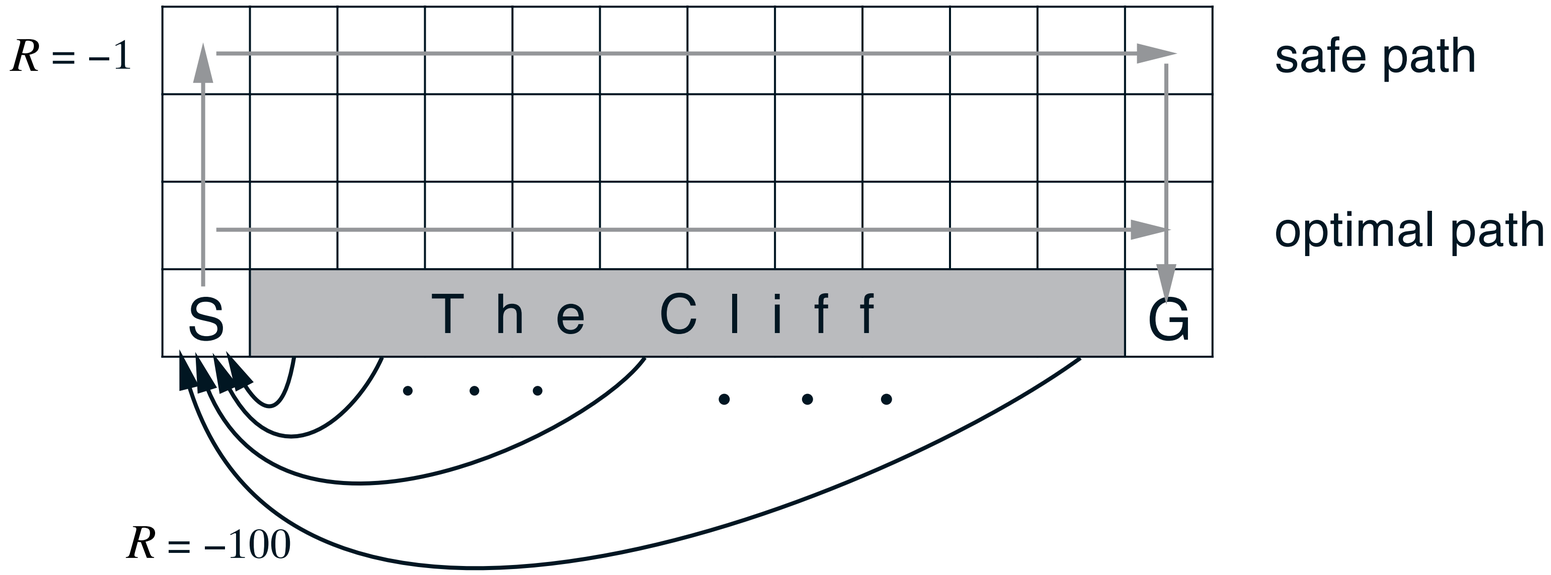
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

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 $Q(\mathit{terminal}, \cdot) = 0$

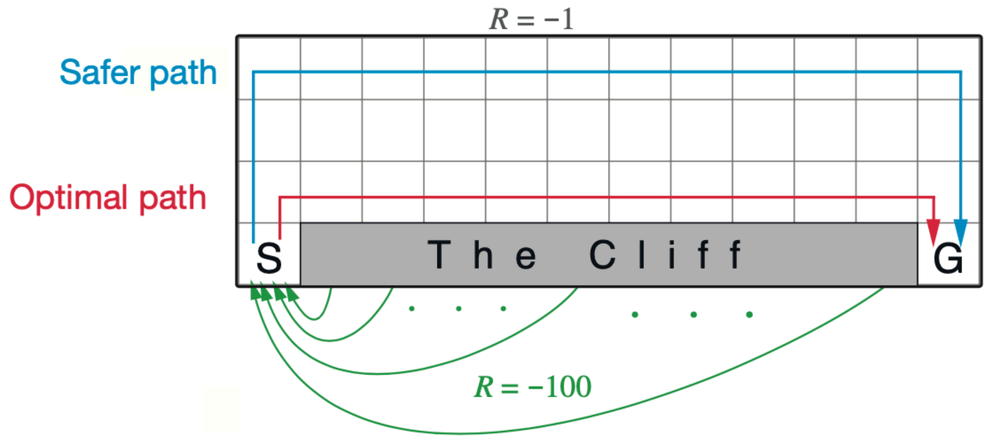
Loop for each episode:
    Initialize S
    Loop for each step of episode:
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$
 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

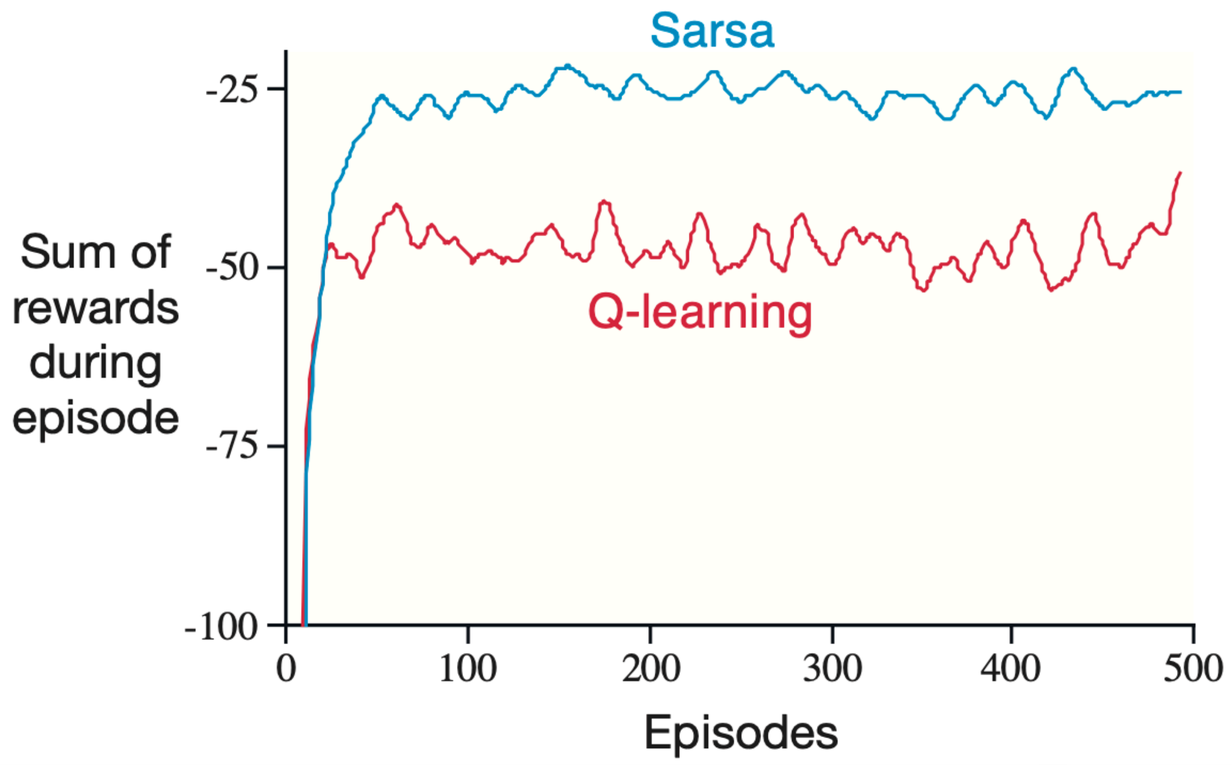
Comparing Q-learning and Sarsa

- Imagine a grid world with rewards of -1 per step, and -100 per step if we fall of the cliff, then teleport but **no termination**
- Let's try Sarsa and Q-learning with ϵ-greedy policies (ϵ=0.1)



- *Why is Q-learning worse here?*





Expected Sarsa

- In Q-learning, we take the **max** over state-action pairs in the update
- In Sarsa, we take the **sample** value of the next state-action pair in the update
- We could also use the **expected value** in the update
  - weighting each action-value based on how likely the action is under the current (target) policy

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

Expected Sarsa

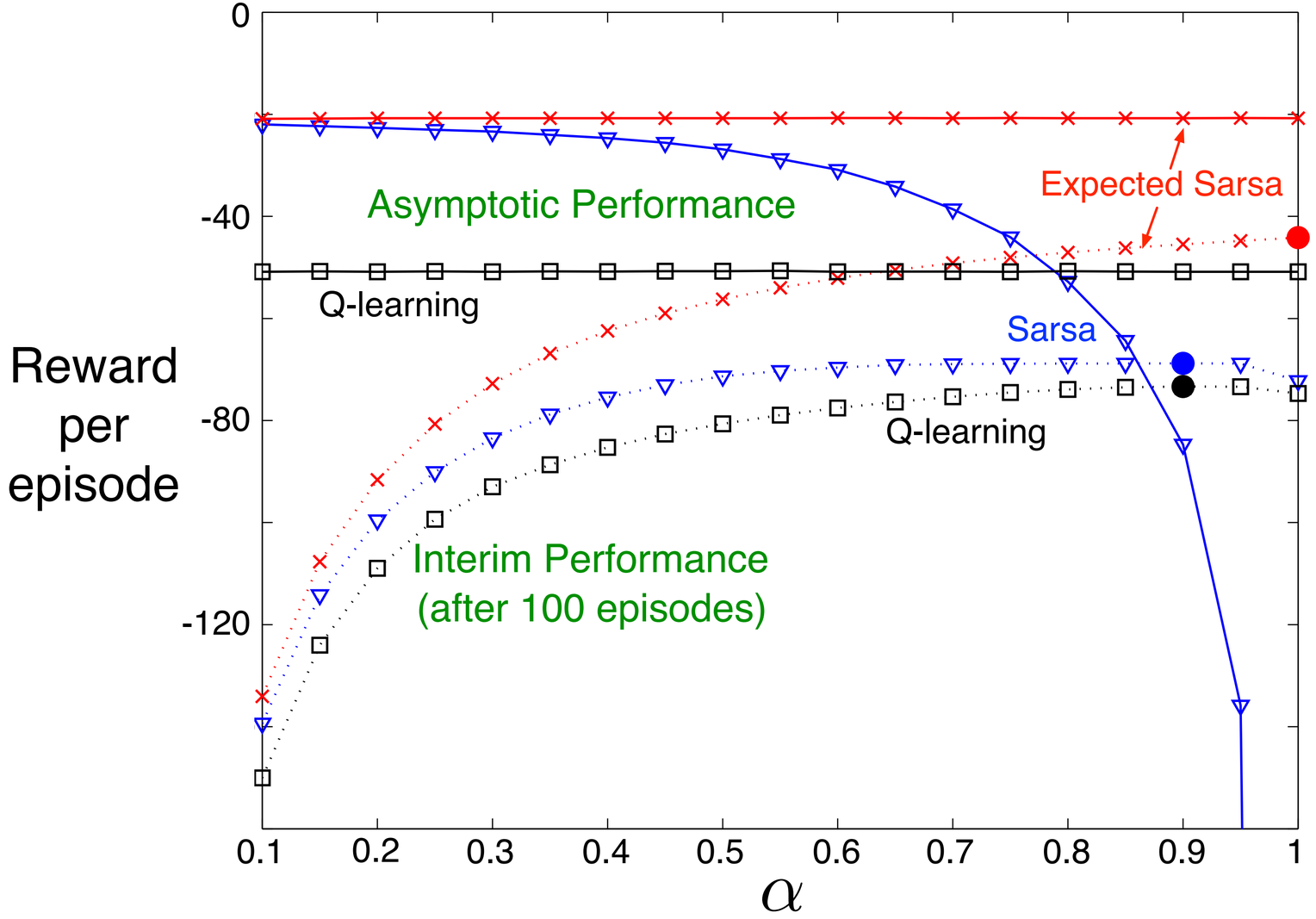
$$
\begin{aligned}
Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \mathbb{E}[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t)\Big] \\
&\leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)\Big]
\end{aligned}
$$

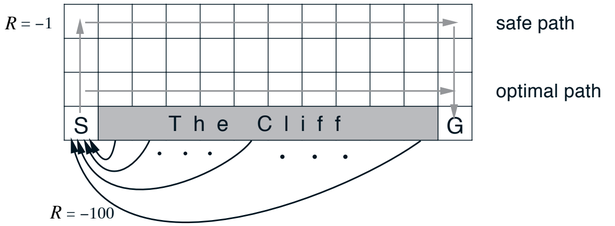
Q-learning Expected Sarsa

- Expected Sarsa's performs better than Sarsa (but costs more)

van Seijen, van Hasselt, Whiteson, & Wiering 2009

Performance on the Cliff-walking Task





General Expected Sarsa

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

- What if π was greedy? and μ was exploratory
  - Expected Sarsa becomes exactly Q-learning!
- Expected Sarsa generalizes Q-learning
  - and improves on Sarsa!

So many policies

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

- Imagine Expected Sarsa with the following policies:
  - \pi: \epsilon-greedy wrt Q, with \epsilon=0.01
  - b: \epsilon-greedy wrt Q, with \epsilon=0.1
  - Why do these choices make sense?

So many policies

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

- We are learning an estimate of q_\pi
  - We are selecting actions according to b. We don’t select actions according to \pi
- Policy b determines S_t, A_t in the update. The environment determines R_{t+1} and S_{t+1} in the update. \pi only determine the little ‘a’ in the update.
- As our estimate of q_\pi gets better and better, so does \pi and b

So many policies

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

- A very sensible choice is to make \pi greedy wrt to Q
  - That's Q-learning
  - We want to learn the optimal policy, AND, we are not following \pi…no harm
- Other off-policy algorithms (e.g., greedy GQ) do exactly this:
  - \pi = greedy
  - b = some exploratory policy

Terminology Review

- TD methods we have learned about are **tabular, one-step, model-free** learning algorithms

- **Tabular:** we store the value function in a table. One entry in the table per value, so each value is stored independently of the others. We are implicitly assuming the state-space ($\mathcal{S}$) is small

- **One-step**: we update a single state or state-action value on each time-step. Only the value of Q(S,A) from S -- A --->S',R. We never update more than one value per learning step

- **Model-free**: we don't assume access to or make use of a model of the world. All learning is driven by sample experience. Data generated by the agent interacting with the environment

Discussion posts

Stuff

- Does Q-learning converge?
- Is Q-learning faster than Expected Sarsa?
- Is the extra sum in Expected Sarsa a big deal computationally?
- Can we think of experience replay used in DQN as a model?
  - https://arxiv.org/abs/1806.04624; https://arxiv.org/abs/1906.05243

Maximization Bias

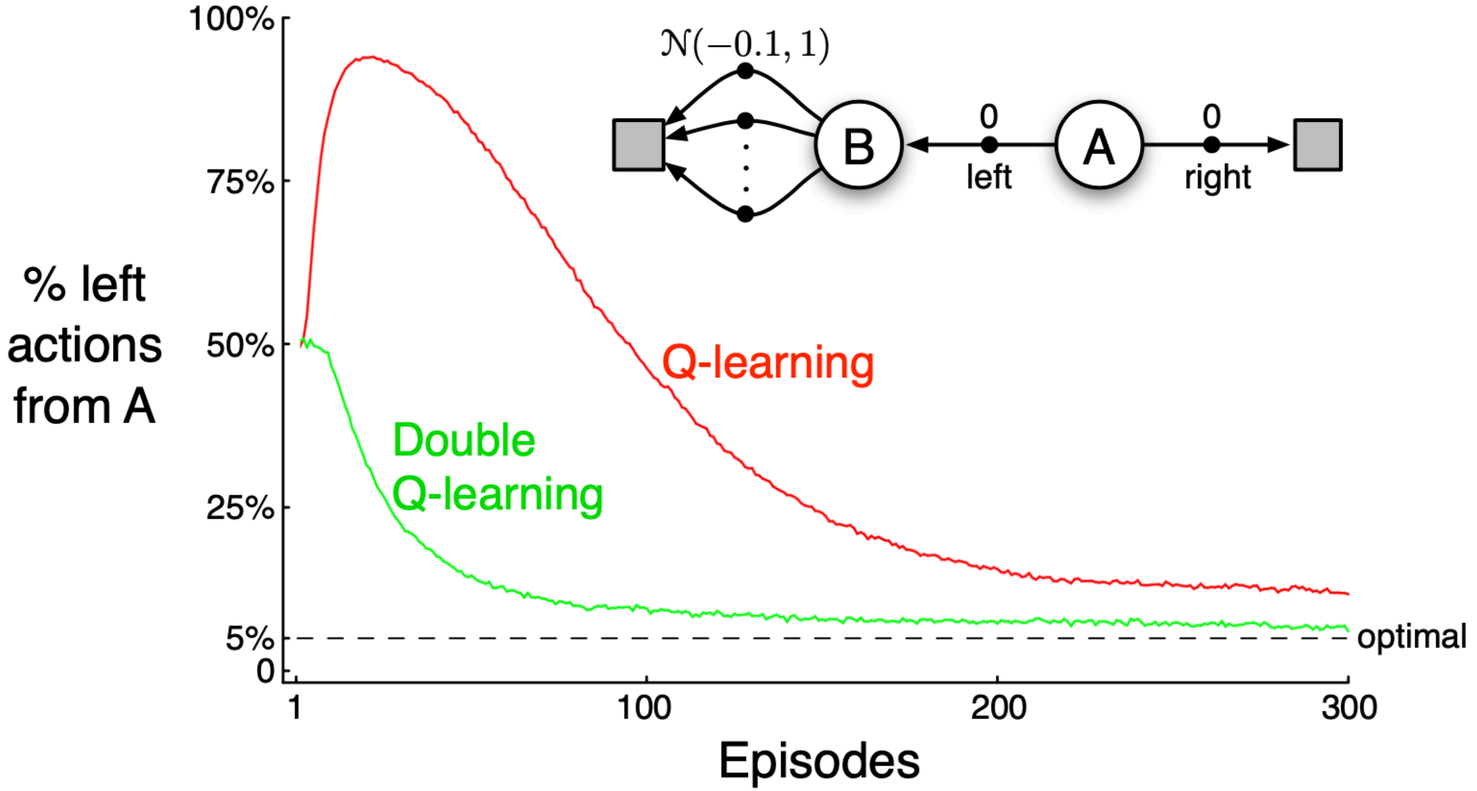
- Consider two random variables X_1, X_2, with distribution p, with $\mu_1 = E[X_1]$, and $\mu_2 = E[X_2]$
- We estimate the expected value of each, with two estimators Y_1 and Y_2, and they are unbiased: $E[Y_1] = \mu_1$, and $E[Y_2] = \mu_2$
- **We want to estimate $v^* = \max(\mu_1, \mu_2)$**
- Given a finite number of samples, Y_1 and Y_2 will not have converged to μ_1 and μ_2
  - one is likely larger than the expected value it is estimating
- If we take the max of these two estimators, then that max is likely to be greater than v^*
- Given *m* random variables one can show:
  $E[\max(Y_1, Y_2, \ldots, Y_m)] >= \max(\mu_1, \mu_2, \ldots, \mu_m)$.
  $E[\max(Y_1, Y_2, \ldots, Y_m)] >= \max(E[Y_1], E[Y_2], \ldots, E[Y_m])$

Maximization Bias

- Q-learning updates towards the max value in the next state
- But we are taking the maximum over estimated values
  - more likely to select overestimated values
  - less likely to select underestimated values
- Consider a single state with many actions, and $q_\pi(s,a) = 0$ for all actions—the true values are zero.
  - our estimates Q(s,a) are uncertain and thus distributed above and below zero
  - max_a $q_\pi(s,a)$!= max_a Q(s,a) => positive bias

Stuff

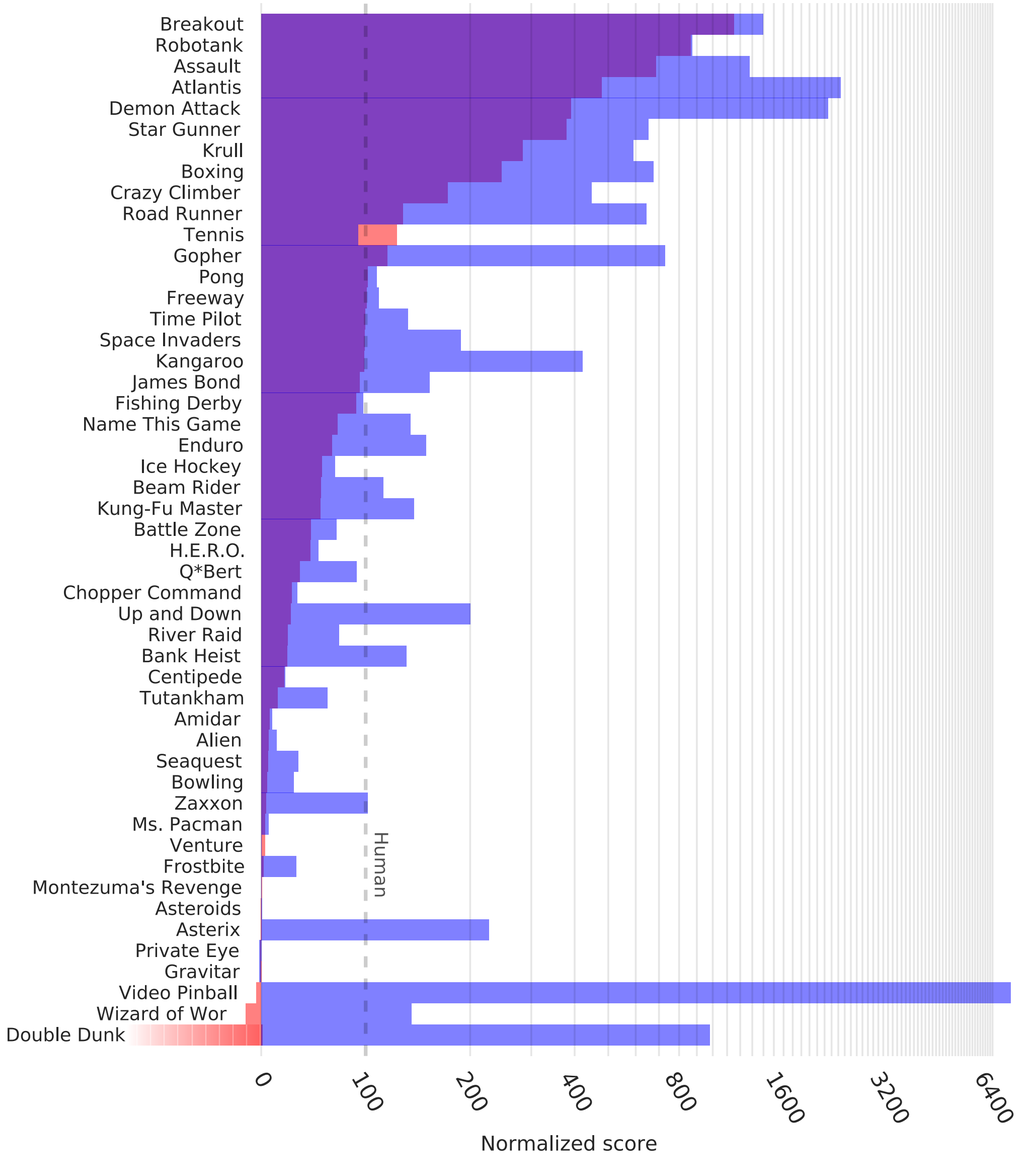
- The over-estimation problem. Should we be worried about it?



Double DQN on Atari

DQN

Double DQN



Hado van Hasselt 2016

Stuff

- Imagine creating off-policy TD with importance sampling. How would we do it?
  - What about off-policy TD with importance sampling for estimating Q(S,A)?
- Why is there no Deep Sarsa or Deep Expected Sarsa?
  - From the book: The Sarsa algorithm was introduced by Rummery and Niranjan (1994). They explored it in conjunction with artificial neural networks and called it “Modified Connectionist Q-learning”.

Stuff

- [Question about Experimentation in RL] In the cliff walking example, we see two ways of presenting the results. Comparing the two algorithms by just looking at learning curves suggests SARSA has a better performance, whereas when we look at environment-based metrics like the actual policy, we see Q learning takes a shorter route / fewer timesteps generally. Are situations like these common in empirical/experimental RL? And has there been any study into what sort of metrics need to be presented together to paint the full picture?
    - Online vs off-line evaluatuion
- Stochastic transitions vs policy stochasticity

Alg development

- Looking at the Bellmen equation, we have three summations one over the next states, one over rewards, and one over actions. We can choose which summations to sample from and which to calculate its expectations.
  - For example, if we calculate the expectations for all of them, we have DP.
  - If we sample from all of them, we have TD methods.
  - We can also have more variations, like sampling the next state and reward, but calculate the expectations over actions like in Expected Sarsa.
  - We can also see Q-learning in the same perspective but using the optimal Bellmen equation instead.