Q-Learning

Reading

- Sutton, Richard S., and Barto, Andrew G. Reinforcement learning: An introduction. MIT press, 2018.
  - http://www.incompleteideas.net/book/the-book-2nd.html
  - Chapters 6.5-6.9
- David Silver lecture on Model-free Control
  - https://www.youtube.com/watch?v=0g4j2k_Ggc4
- Smith, James E., and Robert L. Winkler. "The optimizer’s curse: Skepticism and postdecision surprise in decision analysis." *Management Science* 52.3 (2006): 311-322.
  - Mostly just to motivate maximization bias

Overview

- Q-learning is the most popular algorithm in RL
  - It is essentially off-policy TD learning
  - Similar to other off-policy methods, it is less stable but may find better policies
  - A lot of stabilization techniques have been developed over the years
- Most modern deep RL algorithms are in large part based on the standard Q-learning algorithm
  - Main difference is that Q-learning is essentially search, since it still only works for finite-state MDPs
  - Over the next few weeks, we'll start relaxing that assumption

On-Policy vs Off-Policy Control

- Recall the SARSA Q-value recursion

$$Q'(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

- Why is this on-policy?
  - Need to wait for next action A_{t+1}, selected by current π
- What action can we choose instead?
  - What would be the best given what we know from π?
  - Think policy improvement theorem
  - How about the action that maximizes the Q value?

$$Q'(S_t, A_t) = Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right]$$

- This is Q-learning

Off-policy TD Control: Q-learning

- Similar to on-policy, but try to estimate q_* directly

$$Q'(S_t, A_t) = Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right]$$

- May require less exploration as it "takes" the optimal action
- Guaranteed to converge as long as all state-action pairs are continually updated
  – In some sense, this assumption is unavoidable – guarantees sufficient exploration

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 $Q(terminal, \cdot) = 0$

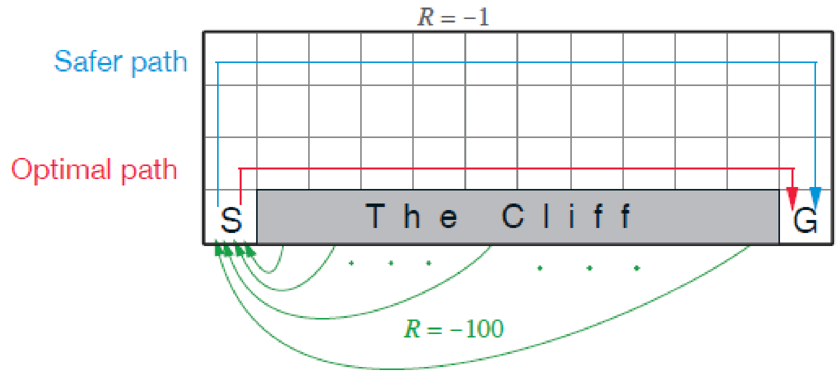
Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Q-learning Exploration

- Exploration is crucial in any RL algorithm
- Q-learning enforces exploration through ϵ-greedy policies
  - i.e., start from your current deterministic policy π and make it ϵ-greedy
  - Next iteration, π' will be deterministic again, so make it ϵ-greedy once more
- This exploration is OK, but it's quite limited
  - Why?
  - All exploration is slight deviation from current policy
  - May not explore much, especially if π changes slowly
- We'll talk about better ways to explore later on

Comparison between on-policy and off-policy

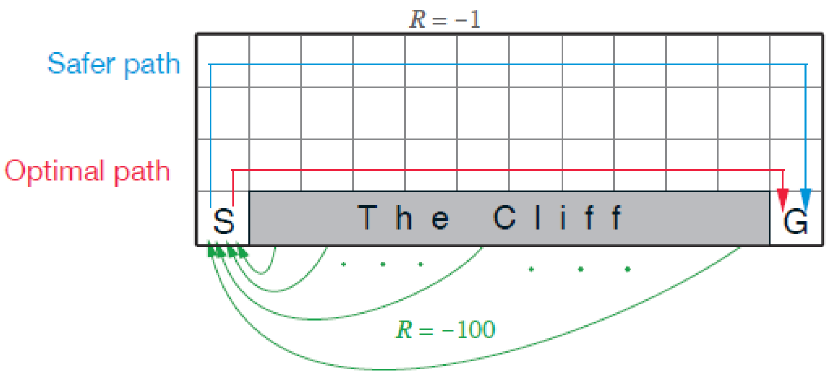
- Consider the following environment

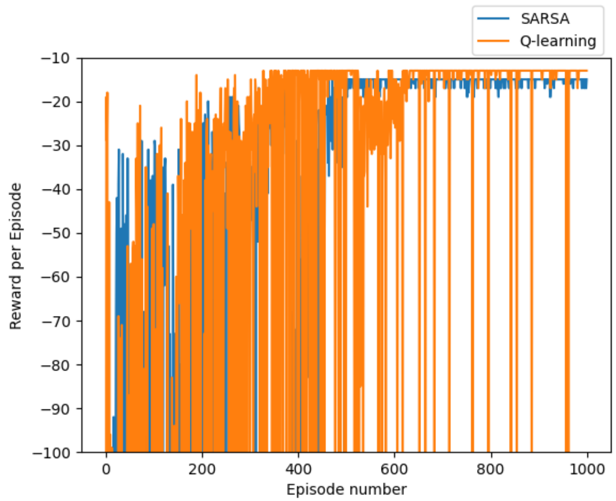


- Goal is to reach G from S
- Actions are up, down, left, right
- Reward of -1 after each step
- Reward of -100 if you fall of The Cliff
- Goal is a sink state (so no more negative reward at that point)

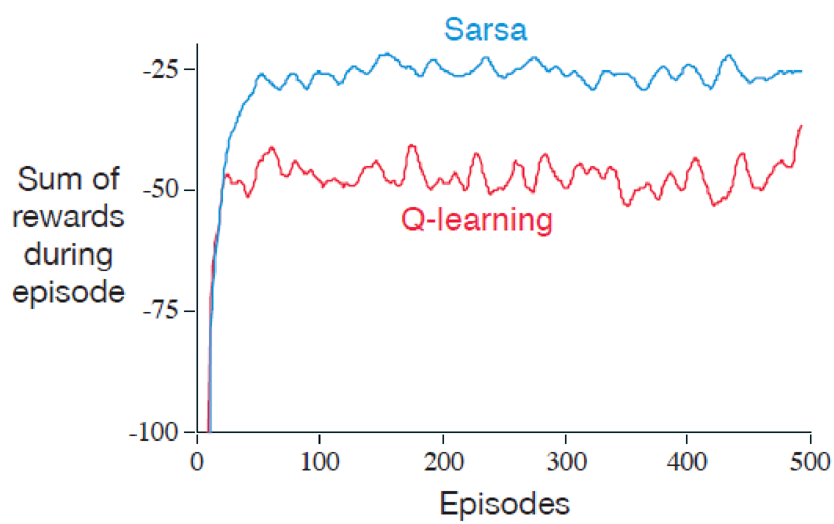
Comparison between on-policy and off-policy, cont'd

- Consider the following environment





- Q-learning learns the optimal path but is less safe due to ϵ-greedy policy
- If ϵ-greediness is gradually removed, both would converge to the optimal



Convergence of Q-learning

- Proof is fairly technical[1,2]
- Q-learning is guaranteed to converge if the following are true
  - All state-action pairs are visited infinitely often
  - $\sum_i \alpha_i = \infty$
  - $\sum_i \alpha_i^2 < \infty$
- The learning rates must converge to 0 but not too quickly
- One of the strongest theoretical results in RL
  – Uses the fact that the Bellman operator is a contractive map

[1] Watkins, Christopher JCH, and Peter Dayan. "Q-learning." Machine learning 8.3 (1992): 279-292.
[2] Tsitsiklis, John N. "Asynchronous stochastic approximation and Q-learning." *Machine learning* 16.3 (1994): 185-202.

Convergence of Q-learning

- Let H denote the Bellman operator, i.e., (for a given q function)

$$Hq(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q\,(S_{t+1}, a')|S_t = s, A_t = a\right]$$
$$= \sum_{s'} P(s', a, s)\left[R(s,a,s') + \gamma \max_{a'} q\,(s', a')\right]$$

- One can show that for any q_1, q_2:

$$\left|\left|H\boldsymbol{q}_1 - H\boldsymbol{q}_2\right|\right|_\infty \leq \gamma \left|\left|\boldsymbol{q}_1 - \boldsymbol{q}_2\right|\right|_\infty$$

  – Where the each q function is interpreted as a vector

$$\boldsymbol{q} = \left[q(s_1, a_1)\;\; q(s_1, a_2)\;\; \ldots\;\; q(s_N, a_1)\;\; \ldots\;\; q\left(s_N, a_p\right)\right]^T$$

  – And the infinity norm is the just the max element

$$\left|\left|\boldsymbol{x}\right|\right|_\infty = \max_i |x_i|$$

Convergence of Q-learning, cont'd

$$\left|\left|H\boldsymbol{q}_1 - H\boldsymbol{q}_2\right|\right|_\infty =$$

$$= \max_{s,a} \left| \sum_{s'} P(s', a, s) \left[R(s, a, s') + \gamma \max_{a'} q_1(s', a') - R(s, a, s') - \gamma \max_{b'} q_2(s', b') \right] \right|$$

$$= \gamma \max_{s,a} \left| \sum_{s'} P(s', a, s) \left[\max_{a'} q_1(s', a') - \max_{b'} q_2(s', b') \right] \right|$$

$$\leq \gamma \max_{s,a} \sum_{s'} P(s', a, s) \left| \max_{a'} q_1(s', a') - \max_{b'} q_2(s', b') \right|$$

Inequality true because $|ax + by| \leq a|x| + b|y|$ for $a, b > 0$

Convergence of Q-learning, cont'd

$$\left|\left|H\boldsymbol{q}_1 - H\boldsymbol{q}_2\right|\right|_\infty \leq \gamma \max_{s,a} \sum_{s'} P(s', a, s) \left|\max_{a'} q_1(s', a') - \max_{b'} q_2(s', b')\right|$$

$$\leq \gamma \max_{s,a} \sum_{s'} P(s', a, s) \max_{a'} |q_1(s', a') - q_2(s', a')|$$

- For second inequality, need to analyze each case:
  - Case 1: suppose $\max_{a'} q_1(s', a') - \max_{b'} q_2(s', b') \geq 0$, i.e.,

$$\left|\max_{a'} q_1(s', a') - \max_{b'} q_2(s', b')\right| = \max_{a'} q_1(s', a') - \max_{b'} q_2(s', b')$$

    - Let $a^* = arg \max_{a'} q_1(s', a')$. Then

$$\max_{a'} q_1(s', a') = q_1(s', a^*)$$
$$\max_{b'} q_2(s', b') \geq q_2(s', a^*)$$

    - i.e.,

$$\max_{a'} q_1(s', a') - \max_{b'} q_2(s', b') \leq q_1(s', a^*) - q_2(s', a^*)$$
$$\leq \max_{a'} |q_1(s', a') - q_2(s', a')|$$

Convergence of Q-learning, cont'd

$$\left||H\boldsymbol{q}_1 - H\boldsymbol{q}_2|\right|_\infty \leq \gamma \max_{s,a} \sum_{s'} P(s', a, s) \left| \max_{a'} q_1(s', a') - \max_{b'} q_2(s', b') \right|$$

$$\leq \gamma \max_{s,a} \sum_{s'} P(s', a, s) \max_{a'} |q_1(s', a') - q_2(s', a')|$$

- For second inequality, need to analyze each case:
  - Case 2: suppose $\max_{a'} q_1(s', a') - \max_{b'} q_2(s', b') < 0$, i.e.,

    $\left| \max_{a'} q_1(s', a') - \max_{b'} q_2(s', b') \right| = \max_{b'} q_2(s', b') - \max_{a'} q_1(s', a')$

    - Let $a^* = arg \max_{a'} q_2(s', a')$. Then

      $$\max_{a'} q_1(s', a') \geq q_1(s', a^*)$$
      $$\max_{b'} q_2(s', b') = q_2(s', a^*)$$

    - i.e.,

    $$\max_{b'} q_2(s', b') - \max_{a'} q_1(s', a') \leq q_2(s', a^*) - q_1(s', a^*)$$
    $$\leq \max_{a'} |q_1(s', a') - q_2(s', a')|$$

Convergence of Q-learning, cont'd

$$\left|\left|H\boldsymbol{q}_1 - H\boldsymbol{q}_2\right|\right|_\infty \leq \gamma \max_{s,a} \sum\nolimits_{s'} P(s', a, s) \left|\max_{a'} q_1(s', a') - \max_{b'} q_2(s', b')\right|$$

$$\leq \gamma \max_{s,a} \sum_{s'} P(s', a, s) \max_{a'} |q_1(s', a') - q_2(s', a')|$$

$$\leq \gamma \max_{s,a} \sum_{s'} P(s', a, s) \max_{s'', a'} |q_1(s'', a') - q_2(s'', a')|$$

$$= \gamma \max_{s,a} \sum_{s'} P(s', a, s) \left|\left|\boldsymbol{q}_1 - \boldsymbol{q}_2\right|\right|_\infty$$

$$= \gamma \left|\left|\boldsymbol{q}_1 - \boldsymbol{q}_2\right|\right|_\infty$$

Convergence of Q-learning

- Let H denote the Bellman operator, i.e., (for a given q function)

$$Hq(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q\left(S_{t+1}, a'\right) | S_t = s, A_t = a\right]$$
$$= \sum_{s'} P(s', a, s)\left[R(s, a, s') + \gamma \max_{a'} q\left(s', a'\right)\right]$$

- One can show that for any $\boldsymbol{q}_1, \boldsymbol{q}_2$:

$$\left|\left|H\boldsymbol{q}_1 - H\boldsymbol{q}_2\right|\right|_\infty \leq \gamma \left|\left|\boldsymbol{q}_1 - \boldsymbol{q}_2\right|\right|_\infty$$

- In particular, the Bellman optimality equation tells us that

$$H\boldsymbol{q}_* = \boldsymbol{q}_*$$

- So applying the Bellman operator multiple times gets us closer to the optimal
  - Policy improvement theorem!

Maximization Bias

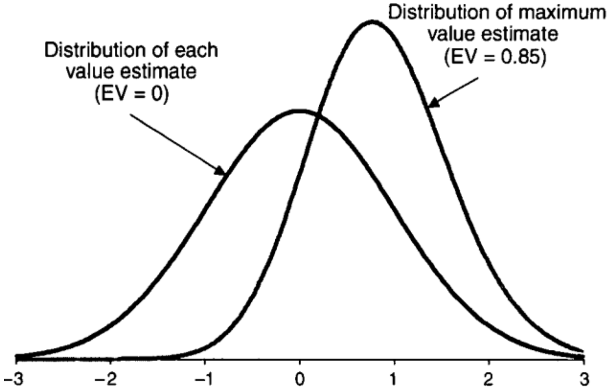
- Turns out taking the max over running averages is biased
  - In essence, the Q-learning actions are based on too "optimistic" estimates of the max
  - Leads to much slower convergence in some cases

Maximization Bias, cont'd

- Let X_1, X_2, X_3 be IID standard normal distributions

$$\mathbb{E}[X_i] = 0, \forall i$$

  - Therefore, $\max_i \mathbb{E}[X_i] = 0$
- Suppose we have running averages for each X_i
  - i.e., $S_i = \frac{1}{n_i}\sum_j x_{ij}$, where x_{ij} are realizations of X_i
- If we estimate $\max_i \mathbb{E}[X_i]$ using $\max_i S_i$, estimate is biased
- Figure shows distributions for 1 sample per X_i
- Gets even worse with more X_i
  - But improves with more samples

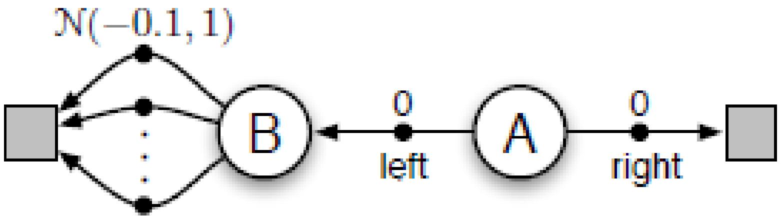


Smith, James E., and Robert L. Winkler. "The optimizer's curse: Skepticism and postdecision surprise in decision analysis." Management Science 52.3 (2006): 311-322.

Maximization Bias, cont'd

Rensselaer

- Same phenomenon occurs when estimating Q values
- Consider this MDP from the book



- Start from A
  - If you go right, you terminate with reward of 0
  - If you go left, you take one of many actions, where each reward is distributed normal with mean -0.1
- Going left has expected reward of -0.1
  - But Q estimate may be positive initially, due to the maximization bias
  - Will significantly slow down learning

Double Q-Learning

- Intuitively, the bias comes from the fact that we're using the same estimator both to estimate Q values and the max
  - How do we improve this?
  - Two independent Q estimators!
- Suppose Q_1 is used to determine the max Q value, i.e.,
$$A^* = \text{argmax}_a\, Q_1(a)$$
- And Q_2 is used to get the actual value of A^*, i.e.,
$$Q_2(A^*) = Q_2\left(\underset{a}{\text{argmax}}\, Q_1(a)\right)$$
- Now it can be shown that this is unbiased, i.e.,
$$\mathbb{E}[Q_2(A^*)] = q(A^*)$$
- Can do the same for $Q_1\left(\underset{a}{\text{argmax}}\, Q_2(a)\right)$

Double Q-Learning, cont'd

- To ensure Q_1 and Q_2 are independent, train on different data
- After every step, update one or the other
  - Can flip a coin to decide which one
  - Crucially, use $Q_2\left(S', arg\max_a Q_1(S', a)\right)$ to remove bias
    - Or vice versa, depending on which one you update

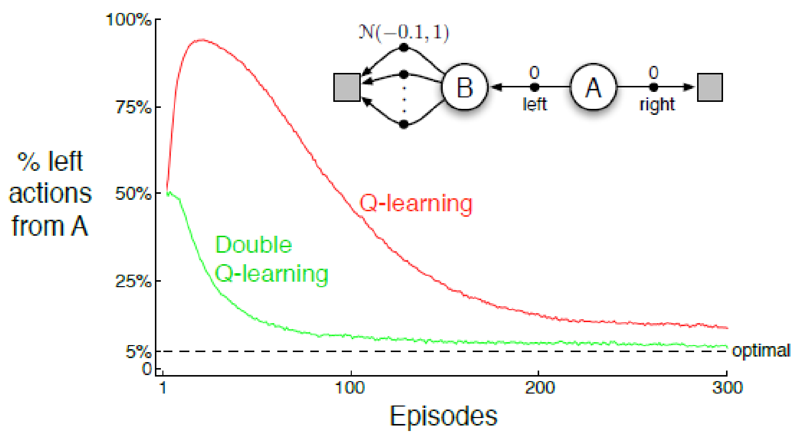
Double Q-learning, for estimating $Q_1 \approx Q_2 \approx q_*$

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 $Q(terminal, \cdot) = 0$

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:
      $Q_1(S, A) \leftarrow Q_1(S, A) + \alpha\Big(R + \gamma Q_2\big(S', \arg\max_a Q_1(S', a)\big) - Q_1(S, A)\Big)$
    else:
      $Q_2(S, A) \leftarrow Q_2(S, A) + \alpha\Big(R + \gamma Q_1\big(S', \arg\max_a Q_2(S', a)\big) - Q_2(S, A)\Big)$
    $S \leftarrow S'$
  until S is terminal

Benefit of Double Q-learning

- Double Q-learning may significantly speed up learning
  - Takes a while until Q-learning bias is reduced
- Double Q-learning is also used in modern RL
  - Often helps with neural nets, but it's not a silver bullet
  - Estimation bias smaller when actions bring significantly different rewards (i.e., identifying the max is easier)



Deficiencies of standard Q-learning

- The assumption that all state-action pairs be visited infinitely often is quite strong
  – Hard to ensure in high-dimensional settings or in infinite-state MDPs (which are more realistic)
  – Training may be very slow if we have a high-dimensional state-space, if we wait for the algorithm to visit all pairs
- MDP transition distribution needs to be stationary
  – i.e., does not change over time
  – May not be very realistic for most systems, e.g., partially observable MDPs with changing sensor noise
  – Stationarity not an issue per se as long as the MDP does not change too quickly

n-step off-policy learning

- Similar to n-step TD learning
- Instead of updating values every step, wait for n steps
- A combination between Q-learning and off-policy MC control
- Recall that off-policy MC requires us to know the relationship between behavior policy b and target policy π:

$$v_\pi(s) = \mathbb{E}_b[\rho_{t:T-1} G_t | S_t = s]$$

  – where $\rho_{t:T} = \prod_{k=t}^{T} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$

- The rest is almost the same as n-step SARSA

n-step off-policy learning, cont'd

Rensselaer

- Return after n steps is $G_{t:t+n}$
- Q update then becomes
  $Q'(S_t, A_t) = Q(S_t, A_t) + \alpha\rho_{t+1:t+n}[G_{t:t+n} + \gamma^n Q(S_{t+n}, A_{t+n}) - Q(S_t, A_t)]$
- Same as n-step SARSA, with the addition of ρ
  - Note that ρ starts at $t + 1$
    - Don't weight first action, A_t, similar to standard Q-learning
- Note that this is different from Q-learning as it doesn't select the maximizing action at time $t + n$
  - All of the exploration is outsourced to the behavior policy
  - Still might be better than on-policy SARSA since the behavior policy might explore aggressively