MDPs: epsilon-greedy

Exploration

Algorithm: reinforcement learning template

For $t = 1, 2, 3, \ldots$
  Choose action $a_t = \pi_{\text{act}}(s_{t-1})$ (**how?**)
  Receive reward r_t and observe new state s_t
  Update parameters (**how?**)

$$s_0; a_1, r_1, s_1; a_2, r_2, s_2; a_3, r_3, s_3; \ldots; a_n, r_n, s_n$$

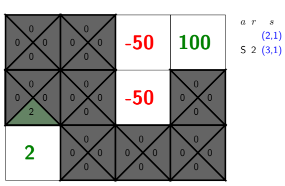
Which **exploration policy** π_{act} to use?

- We have so far given many algorithms for updating parameters (i.e., $\hat{Q}_\pi(s,a)$ or $\hat{Q}_{\text{opt}}(s,a)$). If we were doing supervised learning, we'd be done, but in reinforcement learning, we need to actually determine our **exploration policy** π_{act} to collect data for learning. Recall that we need to somehow make sure we get information about each (s,a).
- We will discuss two complementary ways to get this information: (i) explicitly explore (s,a) or (ii) explore (s,a) implicitly by actually exploring (s',a') with similar features and generalizing.
- These two ideas apply to many RL algorithms, but let us specialize to Q-learning.

No exploration, all exploitation

Attempt 1: Set $\pi_{\text{act}}(s) = \arg\max_{a \in \text{Actions}(s)} \hat{Q}_{\text{opt}}(s,a)$

Run (or press ctrl-enter)



Average (lifetime) utility: 2

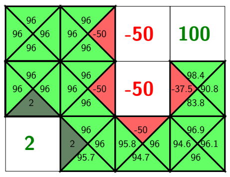
Problem: $\hat{Q}_{\text{opt}}(s,a)$ estimates are inaccurate, **too greedy**!

- The naive solution is to explore using the optimal policy according to the estimated Q-value $\hat{Q}_{\text{opt}}(s,a)$.
- But this fails horribly. In the example, once the agent discovers that there is a reward of 2 to be gotten by going south that becomes its optimal policy and it will not try any other action. The problem is that the agent is being too greedy.
- In the demo, if multiple actions have the same maximum Q-value, we choose randomly. Try clicking "Run" a few times, and you'll end up with minor variations.
- Even if you increase `numEpisodes` to 10000, nothing new gets learned.

No exploitation, all exploration

Attempt 2: Set $\pi_{\text{act}}(s) =$ random from Actions(s)

Run (or press ctrl-enter)



a r s
(2,1)
S 2 (3,1)

Average (lifetime) utility: -19.25

Problem: average utility is low because exploration is **not guided**

- We can go to the other extreme and use an exploration policy that always chooses a random action. It will do a much better job of exploration, but it doesn't exploit what it learns and ends up with a very low utility.
- It is interesting to note that the value (average over utilities across all the episodes) can be quite small and yet the Q-values can be quite accurate. Recall that this is possible because Q-learning is an off-policy algorithm.

Exploration/exploitation tradeoff

Key idea: balance

Need to balance **exploration** and **exploitation**.

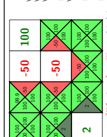
Examples from life: restaurants, routes, research

Epsilon-greedy

Algorithm: epsilon-greedy policy

$$\pi_{\text{act}}(s) = \begin{cases} \arg\max_{a \in \text{Actions}} \hat{Q}_{\text{opt}}(s,a) & \text{probability } 1-\epsilon, \\ \text{random from Actions}(s) & \text{probability } \epsilon. \end{cases}$$

Run (or press ctrl-enter)



Average (lifetime) utility: 31.91

- The natural thing to do when you have two extremes is to interpolate between the two. The result is the **epsilon-greedy** algorithm which explores with probability ϵ and exploits with probability $1-\epsilon$.
- It is natural to let ϵ decrease over time. When you're young, you want to explore a lot ($\epsilon = 1$). After a certain point, when you feel like you've seen all there is to see, then you start exploiting ($\epsilon = 0$).
- For example, we let $\epsilon = 1$ for the first third of the episodes, $\epsilon = 0.5$ for the second third, and $\epsilon = 0$ for the final third. This is not the optimal schedule. Try playing around with other schedules to see if you can do better.