Reinforcement Learning

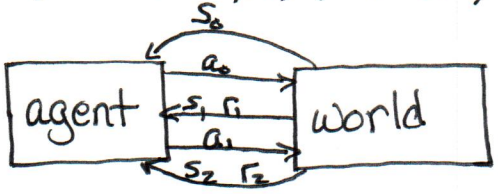
We initially explored MDPs from the assumption that our agent is provided with T & R (a "model of the world") at the start of its learning (not actually "learning") process.

What if T & R are <u>not</u> known?

Could an agent learn an optimal policy simply by knowing its current state identifier (but no state "meaning"), its allowed action identifiers (but no action "meaning"), and the numeric reward of its most recent action?

Yes, by exploring the world over time to obtain training tuples: $\langle s, a, s', r \rangle$ → "resulting state s' and reward r were obtained by taking action a from state s."

We plug the agent which uses $\langle s, a, s', r \rangle$ tuples ("experience") to learn $\pi^*(s) = a$, into the environment (world or simulation) which "knows" T & R (physics, rules, etc) or $f(s, a, T, R) = (s', r)$, and allow them to iteratively interact:



note: $s_1 = s'_0$

$s_2 = s'_1$

$s_3 = s'_2$

This process produces a series of experience tuples for the agent:

$\langle s_0, a_0, s'_0, r_0 \rangle$

$\langle s_1 = s'_0, a_1, s'_1, r_1 \rangle$

$\langle s_2 = s'_1, a_2, s'_2, r_2 \rangle$

$\vdots$

Model-Based RL

Collect many experience tuples by taking random actions or iterating over possible actions. Use them to construct a model of the world — an estimate of T & R.

Build $T(s,a,s') = P(s'|s,a)$ from the sample distribution.

If experiences:	Then T:
$\langle 12, 1, 12, 1 \rangle$	$T(12,1,12) = \frac{1}{2}$
$\langle 12, 1, 13, 1.5 \rangle$	$T(12,1,13) = \frac{1}{2}$
$\langle 12, 2, 12, 0 \rangle$	$T(12,2,12) = \frac{1}{3}$
$\langle 12, 2, 13, 1 \rangle$	$T(12,2,13) = \frac{2}{3}$
$\langle 12, 2, 13, 2 \rangle$	

And R:

$R(12) = 0.5$ $R(13) = 1.5$

OR

$R(12,1) = 1.25$ $R(12,2) = 1$

OR

$R(12,1,12) = 1$ $R(12,1,13) = 1.5$

$R(12,2,12) = 0$ $R(12,2,13) = 1.5$

Build $R(s)$, $R(s,a)$, or $R(s,a,s')$ from the appropriate mean reward.

With an estimated T & R, now solve using any MDP approach.

Model-Free RL

Forget about T & R and try to develop an optimal policy $\pi^*(s) \rightarrow a$ as directly as possible from the data.

The philosophy is a little like Value Iteration (compute precise state utilities then calculate policy) vs Policy Iteration (only compute utility when strictly necessary and then only for the "current" policy).

We don't <u>really</u> care about T & R, so why compute them?

Q-Learning

Popular form of model-free reinforcement learning.

Learns the "Q-function": $Q(s,a)$ = the total expected rewards (current and discounted future) of taking action a from state s and then continuing to act according to our optimal policy.

We treat S and A as discrete, thus Q can be a table/array/etc.

At any time, $\pi(s) = \text{argmax}_a\ Q[s,a]$.
- that is, the action with the highest expected total (including future) rewards

If we run an iterative improvement of Q (via exploration or experimentation) to convergence, we obtain Q^* and thus π^*.

How to get Q?

s = start state

Q = for all A, S, init to small uniform or normal distribution around zero.
(random values drawn from distribution)

loop
- $a = \text{argmax}_a\ Q[s,a]$
- observe s', r from environment
- improve Q using this single $<s,a,s',r>$ tuple
- $s = s'$
- if s is terminal state, s = start state

until Q stops changing (w/in some epsilon)

Improving Q[s,a]

$$Q'[s,a] = (1-\alpha)Q[s,a] + \alpha(r + \gamma \cdot Q[s', \text{argmax}_{a'}(Q[s',a'])])$$

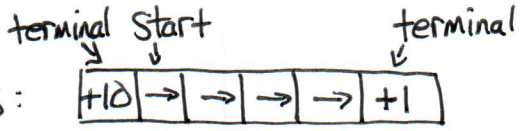
improved Q estimate

- current Q estimate
- learning rate $\alpha = [0,1]$
- discount rate $\gamma = [0,1]$
- value of best next action (from s') according to current policy, equivalent to: $\max_{a'}(Q[s',a'])$

Exploration - Exploitation Dilemma

If we always choose the optimal action (per our current knowledge), we may quickly converge to a suboptimal solution because we did not explore the state space enough. (e.g. our random initialization guided us to a small reward, but there was a big reward we never found)

That is, if our current policy is:



There is no reason our policy would ever change!

So, we force exploration by taking random actions. But not all the time! Because the goal of MDP/RL is still to maximize total rewards over time, including while learning, and "all random actions" certainly won't!

So we mix the two: Sometimes we explore (random action) and sometimes we exploit our learned knowledge (optimal action)

This is called epsilon-greedy, where ϵ is the probability of exploring during an action selection. Usually, ϵ is decreased over time, resulting in a transition from more exploration to more exploitation.

When properly structured, a Q-Learning agent can solve many different problems with no code changes. However, you must choose a state representation and reward function - not always easy!