Reinforcement Learning

What's Wrong with Value/Policy Iteration?

MDP formulation reminder:

- **S: set of all possible states, S0: start state, SF: set of final states**
- **A: set of all possible actions**
- **T: transition matrix, P(s'|s,a)**
- **R: reward function, R(s), R(s,a), R(s,a,s')**

requires fully-observable environment

must know S, A, T, R up front!

Minimal Requirements to Learn a Policy

Could an agent learn an optimal policy knowing *only*:

- current state identifier (no “meaning”)
- allowed action identifiers (no “meaning”)
- numeric reward of most recent action

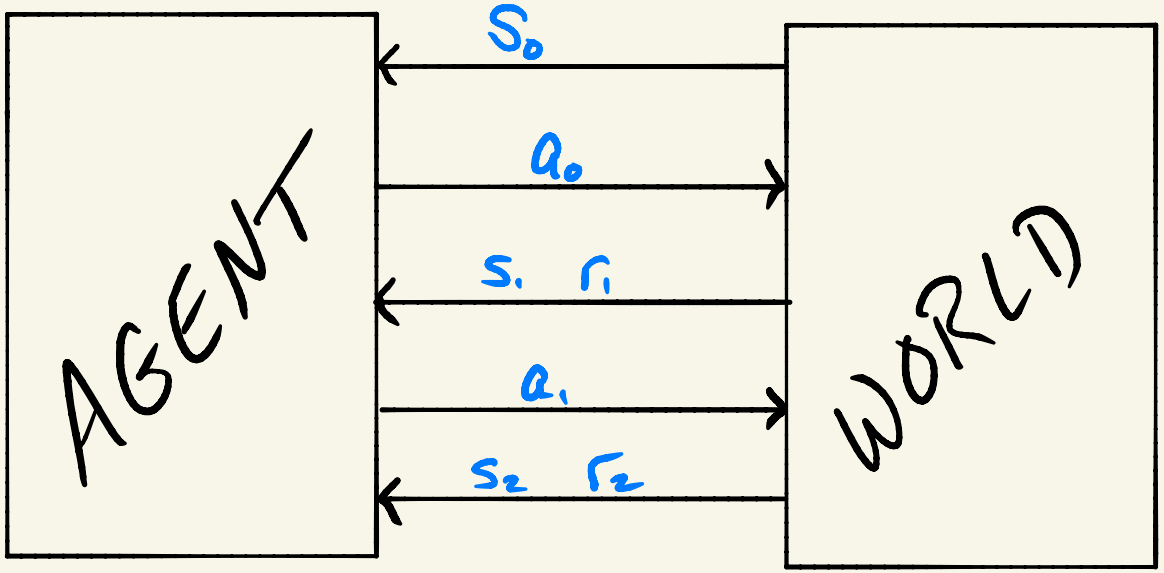
?

Yes! By iteratively exploring the world to obtain training tuples of a new type:

“This state and reward resulted from taking that action from that state.”

This is what we call the Reinforcement Learning problem.

Plugging that Agent into the World



NOTE:

$s_1 = s'_0$

$s_2 = s'_1$

$s_3 = s'_2$

Two interconnected models form the typical RL problem:

- **Agent explores world, uses experience to learn an optimal policy.**
- **Agent usually knows S, S0, A.**

- **World also knows T & R, or f(s, a) = (s', r)**

World tells agent what state it is in. Agent tells world what action it takes.
World tells agent its new state and an immediate reward.

Experience Tuples

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

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

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

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

$$\langle s_3 = s'_2, a_3, s'_3, r_3 \rangle$$

Model-Based Reinforcement Learning

Select random or directed actions to generate many learning tuples.

- **Use them to construct a model of the world (i.e. estimate T & R)**

- **Build T(s,a,s') = P(s'|s,a) from the sample distribution.**
- **Build R(s) or R(s,a) or R(s,a,s') from the mean sample reward.**

Example experiences:

```
< 12, 1, 12, 1   >
< 12, 1, 13, 1.5 >
< 12, 2, 12, 0   >
< 12, 2, 13, 1   >
< 12, 2, 13, 2   >
```

T(s,a,s')

T(12,1,12) = 0.5

T(12,1,13) = 0.5

T(12,2,12) = 0.33

T(12,2,13) = 0.66

R(s)

R(12) = 0.5

R(13) = 1.5

or R(s,a)

R(12,1) = 1.25

R(12,2) = 1

With an estimated T & R, now solve MDP as before!

Model-Free Reinforcement Learning

Solving RL by estimating T & R from many random samples is fine.

Why might that not be ideal?

1. Waste of effort? We only care about $\pi^*(s) \rightarrow a$
2. Incur real world cost/risk while collecting samples.

Remember the argument for Policy Iteration instead of Value Iteration?

We're doing it again.

We don't *actually* care about T & R. So why compute them?