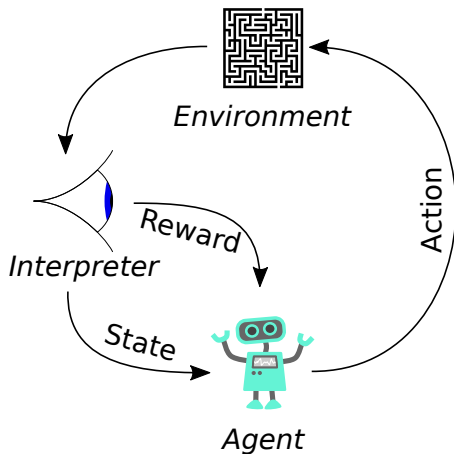


Deep he(a)p, big feat

[arXiv:1707.06887](#) A Distributional Perspective on Reinforcement Learning
[arXiv:1702.08165](#) Reinforcement Learning with Deep Energy-Based Policies

Reinforcement Learning



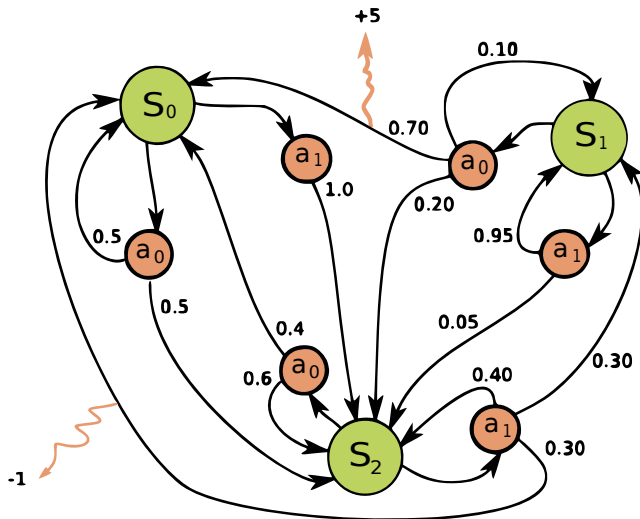
Reinforcement Learning

A framework for modeling intelligent agents. An agent takes an **action** depending on its **state** to change the **environment** with the goal of maximizing their **reward**.

Reinforcement Learning

- ▶ bandits / Markov decision process (MDP)
- ▶ episodes and discounts
- ▶ model-based RL / model-free RL
- ▶ single-agent / multi-agent
- ▶ tabular RL / Deep RL (parameterized policies)
- ▶ discrete / continuous
- ▶ on-policy / off-policy learning
- ▶ policy gradients / Q-learning

Markov decision process (MDP)



Markov decision process (MDP)

- ▶ states $s \in \mathcal{S}$
- ▶ actions $a \in \mathcal{A}$
- ▶ transition probability $p(s'|s, a)$
- ▶ rewards $r(s)$, $r(s, a)$, or $r(s, a, s')$

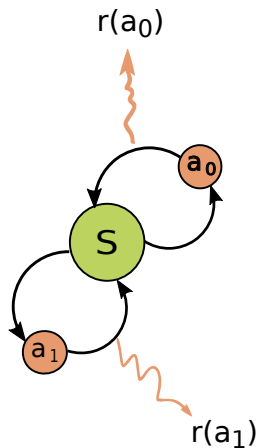
It's *Markov* because the transition $s_t \rightarrow s_{t+1}$ only depends on s_t .
It's a *decision process* because it depends on a .

Goal is to find policy $\pi(a|s)$ that maximizes reward over time.

Multi-armed bandits



Multi-armed bandits



Want to learn $p(r|a)$ and maximize $\langle r \rangle$. Tradeoff between *exploit* and *explore*.

Episodic RL

Agent either acts until a terminal state is reached.

$$s_0 \sim \mu(s_0)$$

$$a_0 \sim \pi(a_0|s_0)$$

$$r_0 = r(s_0, a_0)$$

$$s_1 \sim p(s_1|, s_0, a_0)$$

...

$$a_{T-1} \sim \pi(a_{T-1}|s_{T-1})$$

$$r_{T-1} = r(s_{T-1}, a_{T-1})$$

$$s_T \sim p(s_{T-1}|, s_{T-1}, a_{T-1})$$

The goal is to maximize total rewards

$$\eta(\pi) = E[r_0 + r_1 + \dots + r_{T-1}]$$

Discount factor

If there are no terminal states, the episode lasts “forever” and the agent takes “infinite” actions. In this case, we maximize discounted total rewards

$$\eta(\pi) = \eta(\pi) = E[r_0 + \gamma r_1 + \gamma^2 r_2 \cdots + \gamma^{T-1} r_{T-1}]$$

with discount $\gamma = [0, 1]$.

Without γ ,

- ▶ the agent has no incentive to do anything *now*.
- ▶ η will diverge.

This means that the agent has an effective time-horizon








$$t_h \sim 1/(1 - \gamma)$$

Model-based vs. Model-free

In model-based RL, we try to learn the transition function $p(s'|s, a)$. This lets us predict the expected next state s_{t+1} given state s_t and action a_t . *This means that the agent can think ahead and plan future actions.*

In model-free RL, we either try to learn $\pi(a|s)$ directly (policy gradient methods), or we learn a function $Q(s, a)$ that tells us the value of taking action a when in state s , which implies a $\pi(a|s)$. *This means that the agent has no “understanding” of the process and is essentially a lookup table.*

Multi-agent RL

	SELLER  COOPERATE DEFECT	
BUYER COOPERATE		
 DEFECT		

Parameterized policies / Deep RL

If the total number of states is small, then Monte Carlo or dynamic programming techniques can be used to find $\pi(a|s)$ or $Q(s, a)$. These are sometimes referred to as *tabular methods*.

In many cases, this is intractable. Instead, we need to use a function approximator, such as a neural network, to represent these functions

$$\pi(a|s) \rightarrow \pi(a|s, \theta), \quad Q(s, a) \rightarrow Q(s, a|\theta)$$

This takes advantage of the fact that in similar states we should take similar actions.

Discrete vs. continuous action spaces

Similarly, agents can either select from a discrete set of actions (i.e. left vs. right) or a continuum (steer the boat to heading 136 degrees). I'm not sure why people make a big deal out of the difference

- ▶ discrete: $\pi(a|s)$ is a discrete probability distribution.
- ▶ continuous: $\pi(a|s)$ is (just about always) Gaussian.

On-policy vs. off-policy

If our current best policy is $\pi(a|s)$, do we sample from $\pi(a|s)$ or do we sample from a different policy $\pi'(a|s)$?

- ▶ on-policy: Learn from $\pi(a|s)$, then update based on what worked well / didn't work well.
- ▶ off-policy: Learn from $\pi'(a|s)$ but update $\pi(a|s)$, letting us explore areas of state-action space that aren't likely to come up with our policy. *Can also learn from old experience*

Policy gradients

In which we just go for it and maximize the policy directly. Define

$$R[s(T), a(T)] \equiv \sum_{t=0}^T \gamma^t r(s(t), a(t))$$

We want to maximize $R(t)$, which depends on the trajectory

$$\begin{aligned} \nabla_{\theta} \eta(\theta) &= \nabla_{\theta} E[R] \\ &= \nabla_{\theta} \sum p(R|\theta) R \\ &= \sum R \nabla_{\theta} p(R|\theta) \\ &= \sum R p(R|\theta) \nabla_{\theta} \log p(R|\theta) \\ &= E[R \nabla_{\theta} \log p(R|\theta)] \end{aligned}$$

Policy gradients

The probability of a trajectory is

$$p(R|\theta) = \mu(s_0) \prod_{t=0}^{T-1} \pi(a_t|s_t, \theta) p(s_{t+1}|s_t, a_t)$$

which means that the derivative of it's log doesn't depend on the unknown transition function. This is model-free.

$$\begin{aligned} \nabla_{\theta} \log p(R|\theta) &= \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi(a_t|s_t) \\ \nabla_{\theta} \eta(\theta) &= E \left[R \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi(a_t|s_t) \right] \end{aligned}$$

Policy gradients

Expressing the gradient as an expectation value means we can sample trajectories

$$E \left[R \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi(a_t | s_t) \right] \rightarrow \frac{1}{N} \sum_{i=1}^N \left[R \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi(a_t | s_t) \right]$$

and then do gradient descent on the policy

$$\theta \rightarrow \theta - \alpha \nabla_{\theta} \eta(\theta)$$

Since the gradient update is derived explicitly from trajectories sampled from $\pi(a|s)$, clearly this method is on-policy.

Policy gradients

$$\begin{aligned}\nabla_{\theta}\eta(\theta) &= E\left[R \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi(a_t|s_t)\right] \\ &= E\left[\sum_{t=0}^{T-1} \nabla_{\theta} \log \pi(a_t|s_t) \sum_{t'=t}^{T-1} \gamma^{t'-t} r(s_{t'}, a_{t'})\right] \\ &= E\left[\sum_{t=0}^{T-1} \nabla_{\theta} \log \pi(a_t|s_t) Q_{\pi}(s_t, a_t)\right] \\ &= E\left[\sum_{t=0}^{T-1} \nabla_{\theta} \log \pi(a_t|s_t) \left(Q_{\pi}(s_t, a_t) - V_{\pi}(s_t)\right)\right]\end{aligned}$$

$$Q_{\pi}(s_t, a_t) \equiv \sum_{t'=t}^{T-1} \gamma^{t'-t} r(s_{t'}, a_{t'}), \quad V_{\pi}(s_t) \equiv \sum_{a_t} Q_{\pi}(s_t, a_t) \pi(a_t|s_t)$$

Q-learning

What if we instead learn the Q -function or state-action value function associated with the optimal policy?

$$a_* = \arg \max_a Q_*(s, a)$$

Q-learning is model free

Just knowing the value function $V(s)$ of the state for a policy isn't enough to pick actions because we would need to know the transition function $p(s'|s, a)$.

Q-learning is off-policy

Expanding the definition of $Q(s_t, a_t)$, we see

$$Q_\pi(s_t, a_t) = E[r_t + \gamma V_\pi(s_{t+1})]$$

$$Q_\pi(s_t, a_t) = E[r_t + \gamma E[Q_\pi(s_{t+1}, a_{t+1})]]$$

This is known as *temporal difference learning*.

Now, let's find the optimal Q -function

$$Q_*(s_t, a_t) = E[r_t + \gamma \max_a [Q_\pi(s_{t+1}, a)]]$$

This is Q-learning.

If we have too many states, we instead minimize the loss

$$L(\theta) = \sum_t |r_t + \gamma \max_{a_{t+1}} [Q_\pi(s_{t+1}, a_{t+1}) - Q_\theta(s_t, a_t)]|^2$$

via gradient descent

$$\theta \rightarrow \theta - \alpha \nabla_\theta L(\theta)$$

Q learning II, the SQL

Define

$$\text{soft max}_x f(x) \equiv \log \int dx e^{f(x)}$$

Then, soft Q-learning is

$$Q_*(s_t, a_t) = E[r_t + \gamma \text{soft max}_a Q(s_{t+1}, a)]$$

which has optimal policy

$$\pi(a|s) \propto \exp Q(s, t).$$

Trade-off between optimality and entropy. Allows transfer learning by letting policies compose.

A Distributional Perspective on Reinforcement Learning

Learn a distribution over Q -values. Let $Z(s_t, a_t)$ have an expectation value that is $Q(s, a)$. Then we learn

$$Z(s_t, a_t) = r_t + \gamma Z(s_{t+1}, a_{t+1})$$