OpenAI Spinning Up

[Welcome to Spinning Up in Deep RL! — Spinning Up documentation (openai.com)](openai.com)

Reinforcement Learning

- RL is an area of ML concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward.

- RL differs from supervised learning in not needing labelled input/output pairs, and in not needing sub-optimal actions to be explicitly corrected. Instead the focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).

| HW2 Train virtual human to walk | HW1, 2 Train regression and classification model |

https://en.wikipedia.org/wiki/Reinforcement_learning

Reinforcement Learning

- The environment is typically stated in the form of a Markov decision process (MDP), because many reinforcement learning algorithms for this context use dynamic programming techniques.[3] The main difference between the classical dynamic programming methods and RL algorithms is that the latter do not assume knowledge of an exact mathematical model of the MDP and they target large MDPs where exact methods become infeasible.

https://en.wikipedia.org/wiki/Reinforcement_learning

Markov decision process (MDP)

- In mathematics, a Markov decision process (MDP) is a discrete-time stochastic control process. It provides a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker. MDPs are useful for studying optimization problems solved via dynamic programming.

https://en.wikipedia.org/wiki/Markov_decision_process

Markov decision process (MDP)

- MDPs were known at least as early as the 1950s;[1] a core body of research on Markov decision processes resulted from Ronald Howard's 1960 book, Dynamic Programming and Markov Processes.[2] They are used in many disciplines, including robotics, automatic control, economics and manufacturing. The name of MDPs comes from the Russian mathematician Andrey Markov as they are an extension of Markov chains.

https://en.wikipedia.org/wiki/Markov_decision_process

Core RL algorithms

- Vanilla Policy Gradient (VPG)

- Trust Region Policy Optimization (TRPO)

- Proximal Policy Optimization (PPO)

- Deep Deterministic Policy Gradient (DDPG)

- Twin Delayed DDPG (TD3)

- Soft Actor-Critic (SAC)

They are all implemented with MLP (non-recurrent) actor-critics, making them suitable for fully-observed, non-image-based RL environments.

- These core deep RL algorithms reflect useful progressions of ideas from the recent history of the field, culminating in two algorithms in particular - PPO and SAC-which are close to state of the art on reliability and sample efficiency among policy-learning algorithms.

Algorithms — Spinning Up documentation (openai.com)

On-policy algorithms

- On-policy algorithms <u>don't use old data</u>, which makes them weaker on sample efficiency. But this is for a good reason: these algorithms directly optimize the objective you care about-policy performance-and it works out mathematically that you need on-policy data to calculate the updates. So, this family of algorithms <u>trades off sample efficiency in favor of stability</u>—but you can see the progression of techniques (from VPG to TRPO to PPO) working to make up the deficit on sample efficiency.

HW2 RL walker training (PPO) | num_epoch: 3

Off-policy algorithms

- Algorithms like DDPG and Q-Learning are off-policy, so they are able to reuse old data very efficiently. They gain this benefit by exploiting Bellman's equations for optimality, which a Q-function can be trained to satisfy using any environment interaction data (as long as there's enough experience from the high-reward areas in the environment).
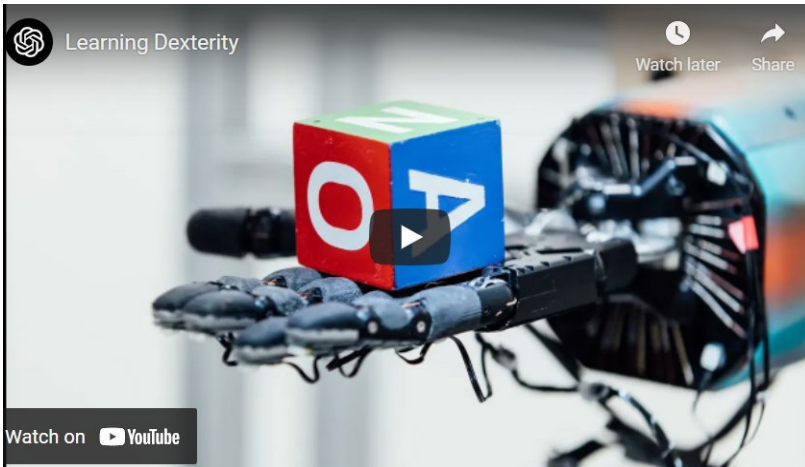
HW2 RL walker training (SAC)

batch_size: 1024
buffer_size: 2000000

Algorithms — Spinning Up documentation (openai.com)

Off-policy algorithms

- But problematically, there are no guarantees that <u>doing a good job of satisfying Bellman's equations leads to having great policy performance</u>. Empirically one can get great performance-and when it happens, the sample efficiency is wonderful-but the absence of guarantees makes algorithms in this class potentially brittle and unstable. TD3 and SAC are descendants of DDPG which make use of a variety of insights to mitigate these issues.

Algorithms — Spinning Up documentation (openai.com)

What RL can do?

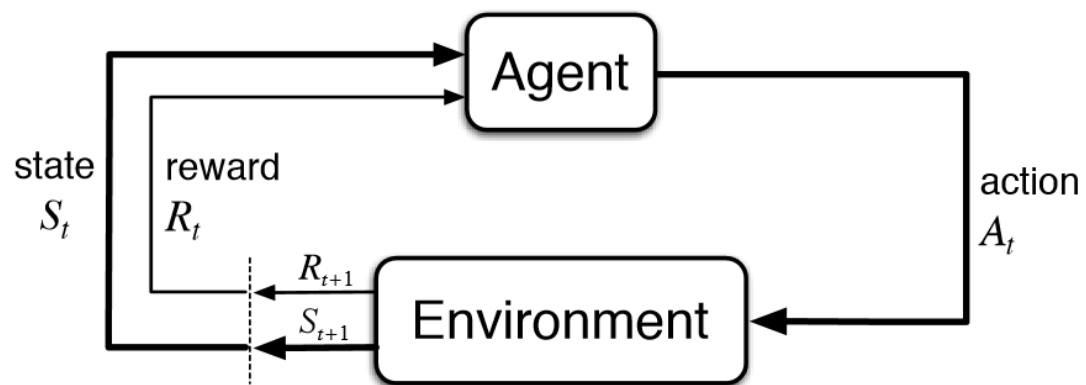https://youtu.be/5fVBeqpFqpw   https://youtu.be/vtp7AGEBbDM
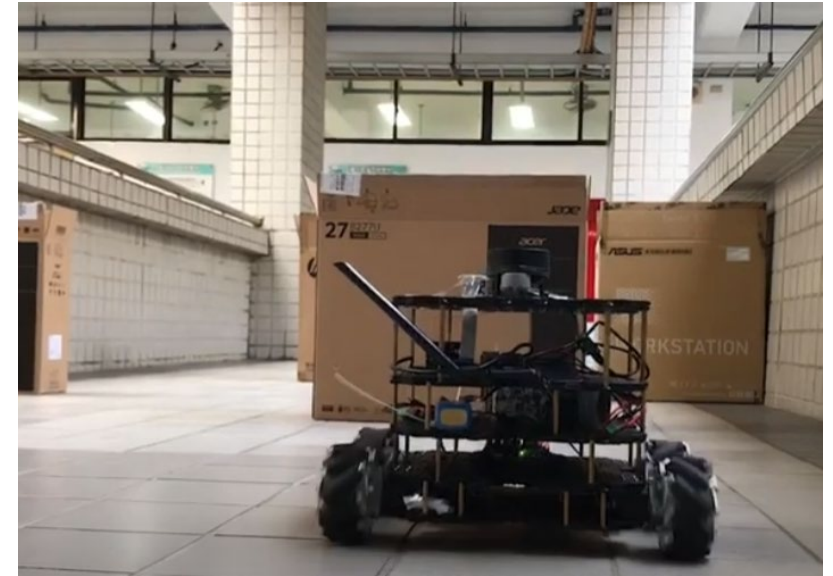
https://spinningup.openai.com/en/latest/spinningup/rl_intro.html

Agent-environment interaction loop

(Sutton and Barto, 1998)

Discuss the environment and agent of the Walker scene

- The main characters of RL are the <u>agent and the environment</u>. The environment is the world that the agent lives in and interacts with. At every step of interaction, the agent sees a (possibly partial) observation of <u>the state of the world</u>, and then decides on <u>an action to take</u>. The environment changes when the agent acts on it, but may also change on its own.

Part 1: Key Concepts in RL — Spinning Up documentation (openai.com)

Agent-environment interaction loop
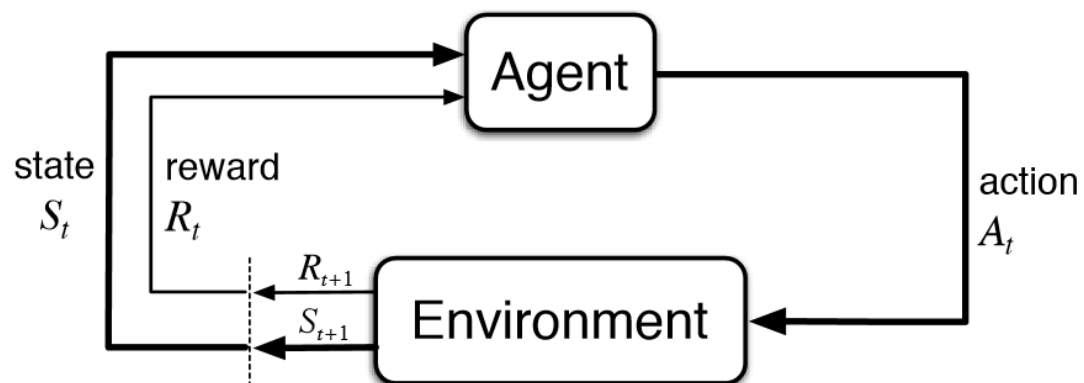
(Sutton and Barto, 1998)

- The agent also perceives a reward signal from the environment, a number that tells it how good or bad the current world state is. The goal of the agent is to maximize its cumulative reward, called return.

- Reinforcement learning methods are ways that the agent can learn behaviors to achieve its goal.

Part 1: Key Concepts in RL — Spinning Up documentation (openai.com)

States and Observations

- A state s is a complete description of the state of the world. There is no information about the world which is hidden from the state. An observation o is a partial description of a state, which may omit information.

- In deep RL, we almost always represent states and observations by a real-valued vector, matrix, or higher-order tensor. For instance, a visual observation could be represented by the RGB matrix of its pixel values; the state of a robot might be represented by its joint angles and velocities.

Discuss the state of the Walker scene

Part 1: Key Concepts in RL — Spinning Up documentation (openai.com)

Action spaces

- Different environments allow different kinds of actions. The set of all valid actions in a given environment is often called the action space. Some environments, like Atari and Go, have discrete action spaces, where only a finite number of moves are available to the agent. Other environments, like where the agent controls a robot in a physical world, have continuous action spaces. In continuous spaces, actions are real-valued vectors.

Discuss the actions of the Walker scene

Part 1: Key Concepts in RL — Spinning Up documentation (openai.com)

Policies

A **policy** is a rule used by an agent to decide what actions to take. It can be deterministic, in which case it is usually denoted by μ:

$$a_t = \mu(s_t),$$

μ: deterministic policy

or it may be stochastic, in which case it is usually denoted by π:

$$a_t \sim \pi(\cdot|s_t).$$

π: stochastic policy

Because the policy is essentially the agent's brain, it's not uncommon to substitute the word "policy" for "agent", eg saying "The policy is trying to maximize reward."

Discuss the policy of the Walker scene

Part 1: Key Concepts in RL — Spinning Up documentation (openai.com)

Policies

In deep RL, we deal with **parameterized policies**: policies whose outputs are computable functions that depend on a set of parameters (eg the weights and biases of a neural network) which we can adjust to change the behavior via some optimization algorithm.

We often denote the parameters of such a policy by θ or ϕ, and then write this as a subscript on the policy symbol to highlight the connection:

$$a_t = \mu_\theta(s_t)$$
$$a_t \sim \pi_\theta(\cdot|s_t).$$

The parameters of deterministic or stochastic policy are denoted as θ or ϕ

Discuss the policy NN of the Walker scene

Part 1: Key Concepts in RL — Spinning Up documentation (openai.com)

Stochastic Policies

- The two most common kinds of stochastic policies in deep RL are categorical policies (discrete actions) and diagonal Gaussian policies (continuous actions).

- Two key computations are centrally important for using and training stochastic policies:

- sampling actions from the policy,

- and computing log likelihoods of particular actions, $\log \pi_\theta(a|s)$.

Part 1: Key Concepts in RL — Spinning Up documentation (openai.com)

Trajectories

A trajectory τ is a sequence of states and actions in the world,

$$\tau = (s_0, a_0, s_1, a_1, ...).$$ τ: trajectory

Show a video that shows a trajectory of the Walker scene

The very first state of the world, s_0, is randomly sampled from the **start-state distribution**, sometimes denoted by ρ_0:

$$s_0 \sim \rho_0(\cdot).$$

State transitions (what happens to the world between the state at time t, s_t, and the state at $t+1$, s_{t+1}), are governed by the natural laws of the environment, and depend on only the most recent action, a_t. They can be either deterministic,

$$s_{t+1} = f(s_t, a_t)$$

or stochastic,

$$s_{t+1} \sim P(\cdot|s_t, a_t).$$

Part 1: Key Concepts in RL — Spinning Up documentation (openai.com)

Reward and Return

The reward function R is critically important in reinforcement learning. It depends on the current state of the world, the action just taken, and the next state of the world:

$$r_t = R(s_t, a_t, s_{t+1})$$

although frequently this is simplified to just a dependence on the current state, $r_t = R(s_t)$, or state-action pair $r_t = R(s_t, a_t)$.

r_t: reward at step t

Discuss the rewards of the Walker scene

Part 1: Key Concepts in RL — Spinning Up documentation (openai.com)

Reward and Return

The goal of the agent is to maximize some notion of cumulative reward over a trajectory, but this actually can mean a few things. We'll notate all of these cases with $R(\tau)$, and it will either be clear from context which case we mean, or it won't matter (because the same equations will apply to all cases).

One kind of return is the **finite-horizon undiscounted return**, which is just the sum of rewards obtained in a fixed window of steps:

$$R(\tau) = \sum_{t=0}^{T} r_t.$$

Another kind of return is the **infinite-horizon discounted return**, which is the sum of all rewards *ever* obtained by the agent, but discounted by how far off in the future they're obtained. This formulation of reward includes a discount factor $\gamma \in (0, 1)$:

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t.$$

Find the discount factor $\gamma \in (0,1)$

$R(\tau)$: discounted return of a trajectory τ

Part 1: Key Concepts in RL — Spinning Up documentation (openai.com)

Reward and Return

- Why would we ever want a discount factor, though? Don't we just want to get all rewards? We do, but the discount factor is both <u>intuitively appealing</u> and <u>mathematically convenient</u>.

- On an intuitive level: cash now is better than cash later. Mathematically: an infinite-horizon sum of rewards may not converge to a finite value, and is hard to deal with in equations. But with a discount factor and under reasonable conditions, the infinite sum converges.

Part 1: Key Concepts in RL — Spinning Up documentation (openai.com)

The RL problem

Whatever the choice of return measure (whether infinite-horizon discounted, or finite-horizon undiscounted), and whatever the choice of policy, the goal in RL is to select a policy which maximizes **expected return** when the agent acts according to it.

To talk about expected return, we first have to talk about probability distributions over trajectories.

Let's suppose that both the environment transitions and the policy are stochastic. In this case, the probability of a T-step trajectory is:

$$P(\tau|\pi) = \rho_0(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t).$$

Probability of obtaining a trajectory
τ under a stochastic policy π

Part 1: Key Concepts in RL — Spinning Up documentation (openai.com)

The RL problem

The expected return (for whichever measure), denoted by $J(\pi)$, is then:

$J(\pi)$: The expected value of a trajectory τ under a stochastic policy π

$$J(\pi) = \int_\tau P(\tau|\pi)R(\tau) = \operatorname*{E}_{\tau \sim \pi}\left[R(\tau)\right].$$

The central optimization problem in RL can then be expressed by

π^*: The optimal policy so that the expected value a trajectory τ is maximum

$$\pi^* = \arg\max_\pi J(\pi),$$

with π^* being the **optimal policy**.

What is the optimal policy for walker?

Part 1: Key Concepts in RL — Spinning Up documentation (openai.com)

Value functions

1. The **On-Policy Value Function**, $V^\pi(s)$, which gives the expected return if you start in state s and always act according to policy π:

$$V^\pi(s) = \mathop{E}_{\tau \sim \pi} [R(\tau)\,|\,s_0 = s]$$

$V^\pi(s)$: the expected value of returns if we start from s under a policy π

2. The **On-Policy Action-Value Function**, $Q^\pi(s, a)$, which gives the expected return if you start in state s, take an arbitrary action a (which may not have come from the policy), and then forever after act according to policy π:

$$Q^\pi(s, a) = \mathop{E}_{\tau \sim \pi} [R(\tau)\,|\,s_0 = s, a_0 = a] \quad Q^\pi(s, a)$$

3. The **Optimal Value Function**, $V^*(s)$, which gives the expected return if you start in state s and always act according to the *optimal* policy in the environment:

$$V^*(s) = \max_\pi \mathop{E}_{\tau \sim \pi} [R(\tau)\,|\,s_0 = s] \quad V^*(s)$$

4. The **Optimal Action-Value Function**, $Q^*(s, a)$, which gives the expected return if you start in state s, take an arbitrary action a, and then forever after act according to the *optimal* policy in the environment:

$$Q^*(s, a) = \max_\pi \mathop{E}_{\tau \sim \pi} [R(\tau)\,|\,s_0 = s, a_0 = a] \quad Q^*(s, a)$$

Bellman Equations

The Bellman equations for the on-policy value functions are

$$V^\pi(s) = \mathop{\mathrm{E}}_{\substack{a \sim \pi \\ s' \sim P}} \left[r(s, a) + \gamma V^\pi(s') \right],$$

$$Q^\pi(s, a) = \mathop{\mathrm{E}}_{s' \sim P} \left[r(s, a) + \gamma \mathop{\mathrm{E}}_{a' \sim \pi} \left[Q^\pi(s', a') \right] \right],$$

where $s' \sim P$ is shorthand for $s' \sim P(\cdot|s, a)$, indicating that the next state s' is sampled from the environment's transition rules; $a \sim \pi$ is shorthand for $a \sim \pi(\cdot|s)$; and $a' \sim \pi$ is shorthand for $a' \sim \pi(\cdot|s')$.

Part 1: Key Concepts in RL — Spinning Up documentation (openai.com)

Bellman Equations

The Bellman equations for the optimal value functions are

$$V^*(s) = \max_a \mathop{E}_{s' \sim P} \left[r(s, a) + \gamma V^*(s') \right],$$

$$Q^*(s, a) = \mathop{E}_{s' \sim P} \left[r(s, a) + \gamma \max_{a'} Q^*(s', a') \right].$$

The crucial difference between the Bellman equations for the on-policy value functions and the optimal value functions, is the absence or presence of the max over actions. Its inclusion reflects the fact that whenever the agent gets to choose its action, in order to act optimally, it has to pick whichever action leads to the highest value.

Part 1: Key Concepts in RL — Spinning Up documentation (openai.com)

Advantage Functions

- Sometimes in RL, we don't need to describe how good an action is in an absolute sense, but only how much better it is than others on average. That is to say, we want to know the relative advantage of that action. We make this concept precise with the advantage function.

$$A^\pi(s, a) = Q^*(s, a) - V^*(s)$$

Summary – Key concepts and terminology

- states and observations

- action spaces

- policies

- trajectories

- different formulations of return

- the RL optimization problem

- value functions

Part 1: Key Concepts in RL — Spinning Up documentation (openai.com)