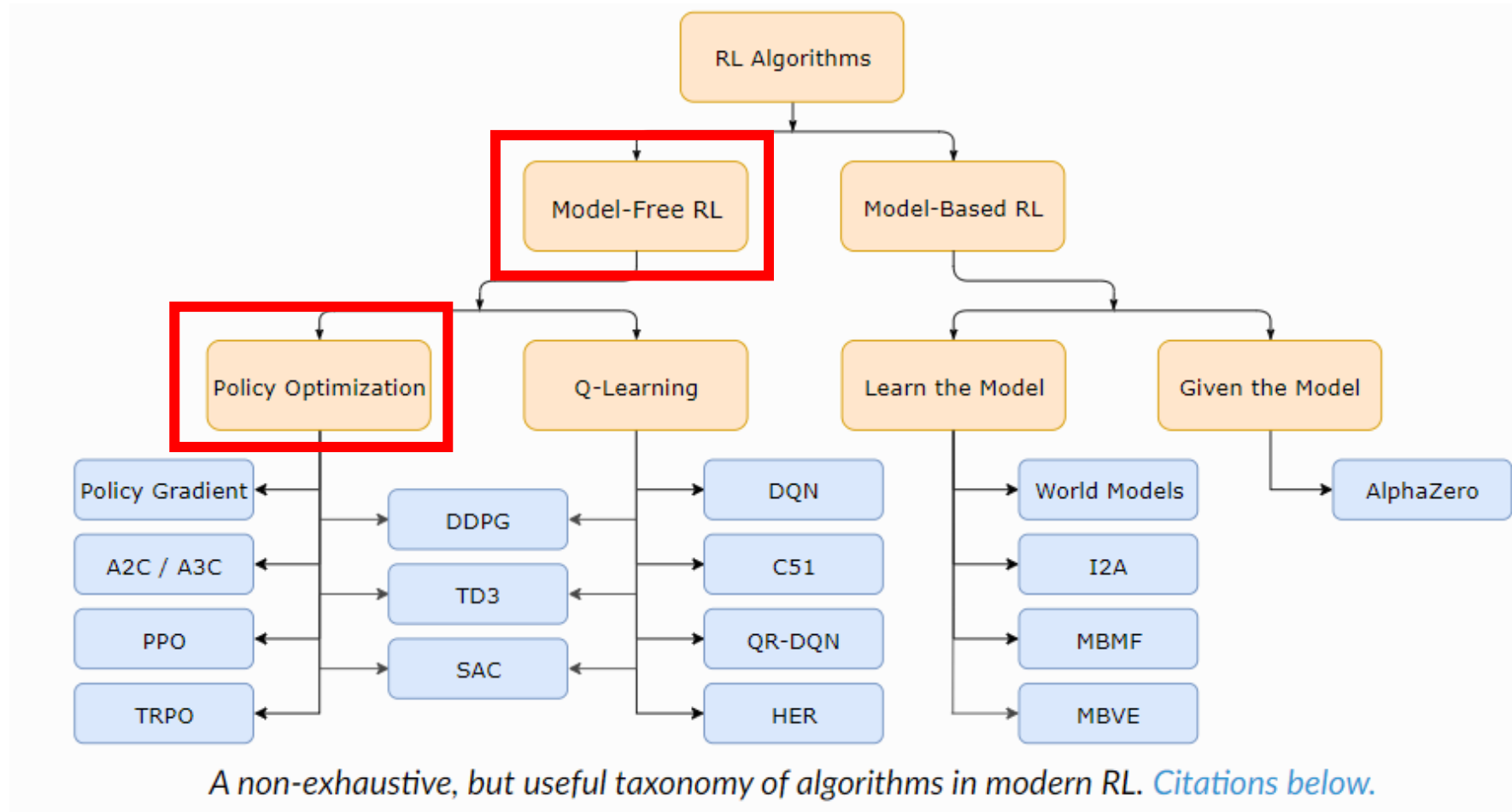


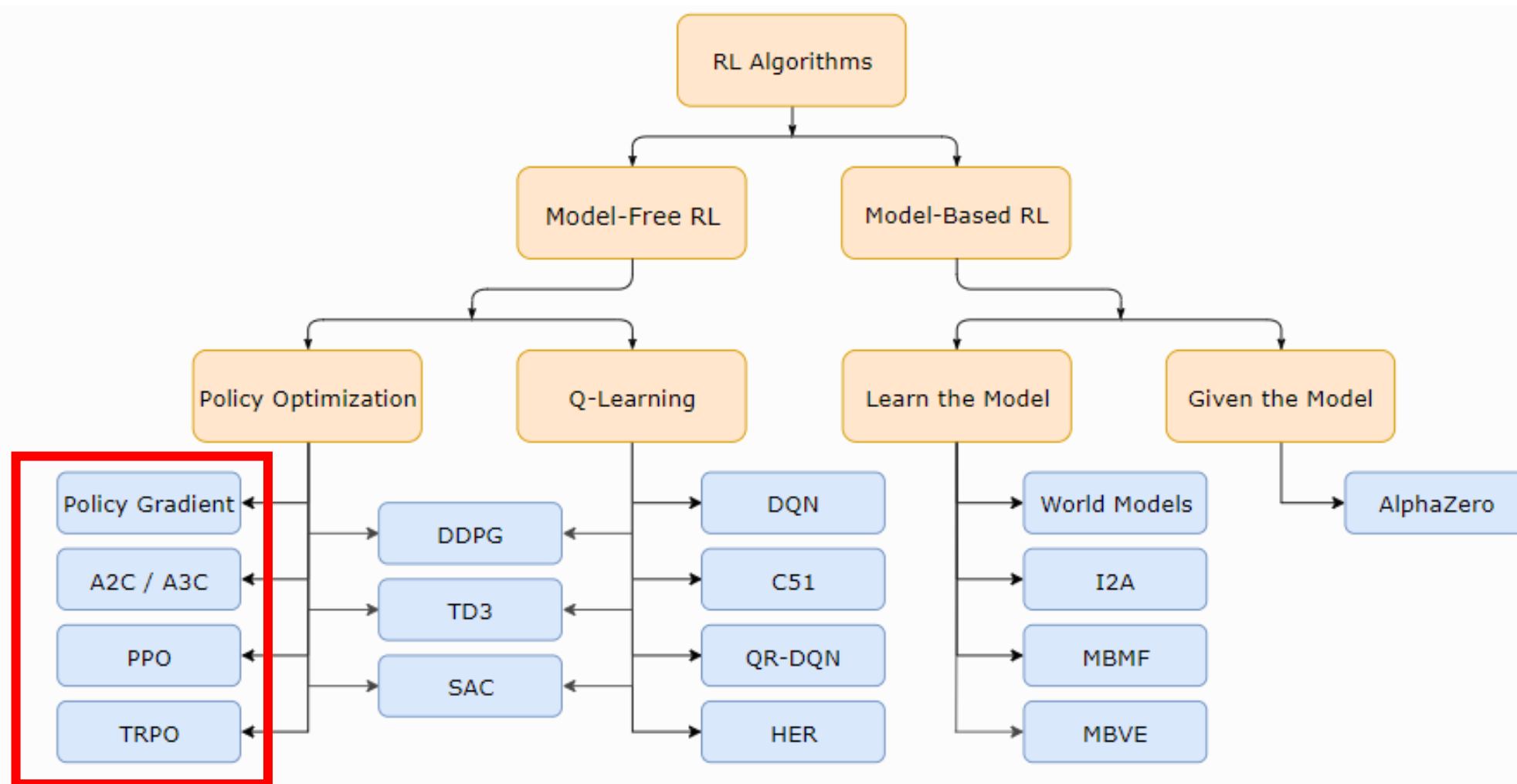
Policy optimization based RL

Model free RL algorithms include Policy Optimization and Q-Learning.



Policy optimization based RL

Popular policy optimization based RL algorithms include PG, A2C/A3C, TRPO, and PPO.



A non-exhaustive, but useful taxonomy of algorithms in modern RL. Citations below.

Recap – Key concepts

Policies	$a_t \sim \pi_\theta(s_t)$
Trajectories	$\tau = (s_0, a_0, s_1, a_1, \dots)$
Reward	$r_t = R(s_t, a_t, s_{t+1})$
Return	$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t$

$$P(\tau|\pi) = \rho_0(s_0) \cdot \pi(a_0|s_0) \cdot P(s_1|s_0, a_0) \cdot \pi(a_1|s_1) \cdot P(s_2|s_1, a_1) \cdot \dots$$

$$J(\pi) = E_{\tau \sim \pi} (R(\tau))$$

$$V^\pi(s) = E_{\tau \sim \pi} (R(\tau) | s_0 = s)$$

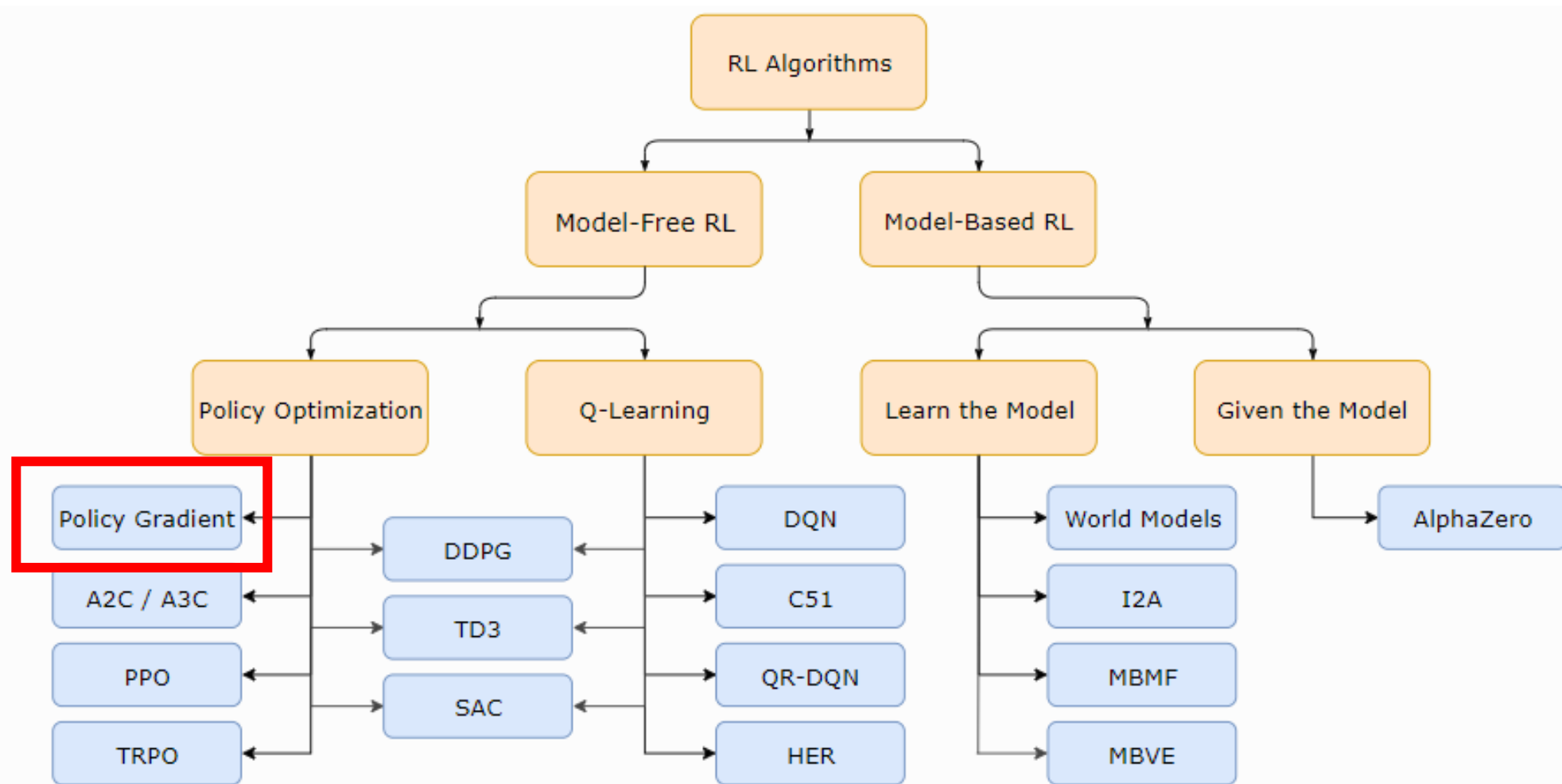
$$Q^\pi(s, a) = E_{\tau \sim \pi} (R(\tau) | s_0 = s, a_0 = a)$$

$$V^\pi(s) = E_{\substack{\tau \sim \pi \\ s' \sim P}} [r(s, a) + \gamma V^\pi(s')]$$

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

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

Policy gradient



A non-exhaustive, but useful taxonomy of algorithms in modern RL. Citations below.

Policy gradient

The screenshot shows a web browser window displaying the OpenAI Spinning Up documentation. The browser's address bar shows the URL `spinningup.openai.com/en/latest/spinningup/rl_intro3.html`. The page features the OpenAI Spinning Up logo and a search bar. A sidebar on the left contains a navigation menu with sections like 'USER DOCUMENTATION', 'INTRODUCTION TO RL', and 'Part 3: Intro to Policy Optimization'. The main content area is titled 'Part 3: Intro to Policy Optimization' and includes a 'Table of Contents' with links to various sub-topics. The page also has a 'Docs' breadcrumb and an 'Edit on GitHub' link.

Part 3: Intro to Policy Optimization

Docs » Part 3: Intro to Policy Optimization [Edit on GitHub](#)

Part 3: Intro to Policy Optimization

Table of Contents

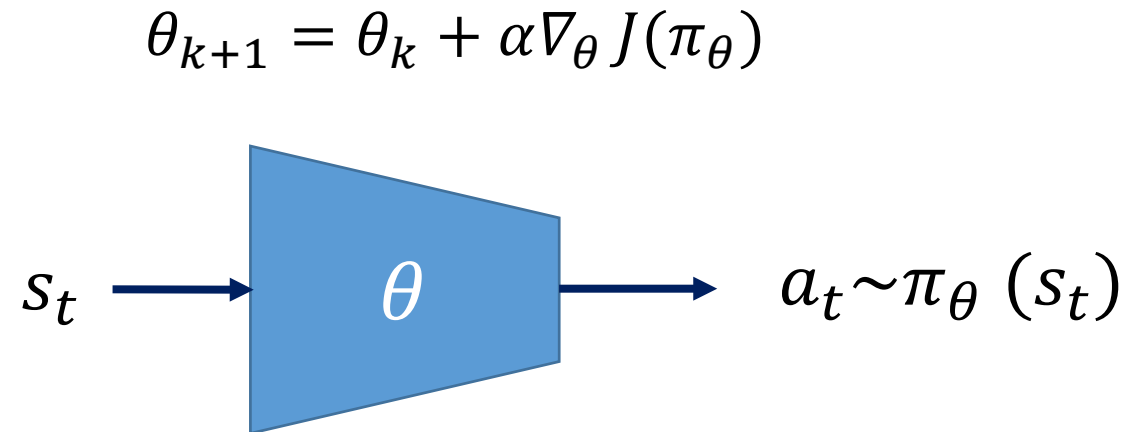
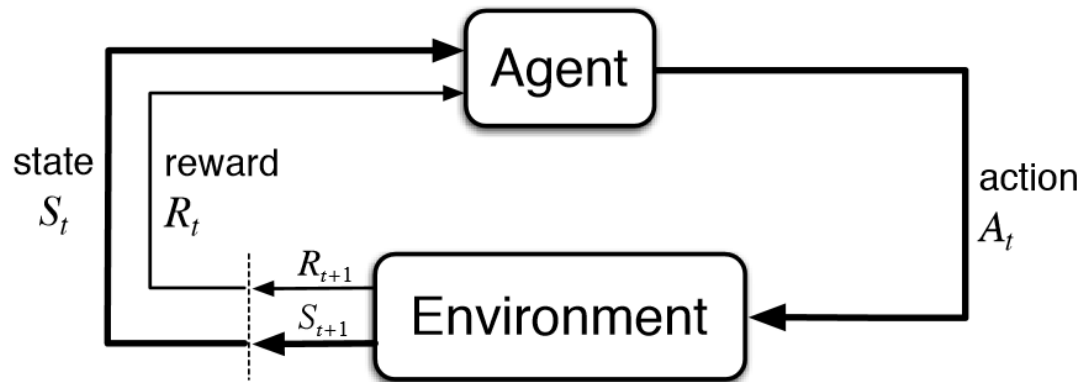
- [Part 3: Intro to Policy Optimization](#)
 - [Deriving the Simplest Policy Gradient](#)
 - [Implementing the Simplest Policy Gradient](#)
 - [Expected Grad-Log-Prob Lemma](#)
 - [Don't Let the Past Distract You](#)
 - [Implementing Reward-to-Go Policy Gradient](#)
 - [Baselines in Policy Gradients](#)
 - [Other Forms of the Policy Gradient](#)
 - [Recap](#)

In this section, we'll discuss the mathematical foundations of policy optimization algorithms, and connect the material to sample code. We will cover three key results in the theory of **policy gradients**:

- [the simplest equation](#) describing the gradient of policy performance with respect to policy parameters,
- a rule which allows us to [drop useless terms](#) from that expression,
- and a rule which allows us to [add useful terms](#) to that expression.

How to train NN to learn the best policy?

The NN should learn to generate actions that maximize cumulative rewards.



$$\max J(\pi) = E_{\tau \sim \pi} (R(\tau))$$

$$\tau = (s_0, a_0, s_1, a_1, \dots)$$

$$P(\tau|\pi) = \rho_0(s_0) \cdot \pi(a_0|s_0) \cdot P(s_1|s_0, a_0) \cdot \pi(a_1|s_1) \cdot P(s_2|s_1, a_1) \cdot \dots$$

How to train NN with maximize cumulative reward?

To find π_θ that has maximum $J(\pi)$, we need $\nabla J(\pi)$

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_\theta)$$

$$\nabla J(\pi) = \nabla E_{\tau \sim \pi} (R(\tau))$$

$$= \nabla \sum_{\tau \sim \pi} P(\tau|\pi) R(\tau)$$

$$= \sum_{\tau \sim \pi} \nabla P(\tau|\pi) R(\tau)$$

$$= \sum_{\tau \sim \pi} P(\tau|\pi) \nabla \log P(\tau|\pi) R(\tau) \quad \nabla f = f \nabla \log f$$

How to train NN with maximize cumulative reward?

$$\begin{aligned}
 \nabla J(\pi) &= \sum_{\tau \sim \pi} P(\tau|\pi) \nabla \log P(\tau|\pi) R(\tau) \\
 &= \sum_{\tau \sim \pi} P(\tau|\pi) [\nabla_{\theta} \log \pi(a_0|s_0) + \nabla_{\theta} \log \pi(a_1|s_1) + \dots + \nabla_{\theta} \log \pi(a_T|s_T)] R(\tau) \\
 &= \sum_{\tau \sim \pi} P(\tau|\pi) \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) \right] R(\tau) \\
 &= E_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) R(\tau) \right] \quad \tau = (s_0, a_0, s_1, a_1, \dots, s_T, a_T)
 \end{aligned}$$

$$P(\tau|\pi) = \rho_0(s_0) \cdot \pi(a_0|s_0) \cdot P(s_1|s_0, a_0) \cdot \pi(a_1|s_1) \cdot P(s_2|s_1, a_1) \cdot \dots \cdot \pi(a_T|s_T) \cdot P(s_{T+1}|s_T, a_T)$$

$$\log P(\tau|\pi) = \log \rho_0(s_0) + \log \pi(a_0|s_0) + \log P(s_1|s_0, a_0) + \log \pi(a_1|s_1) + \dots$$

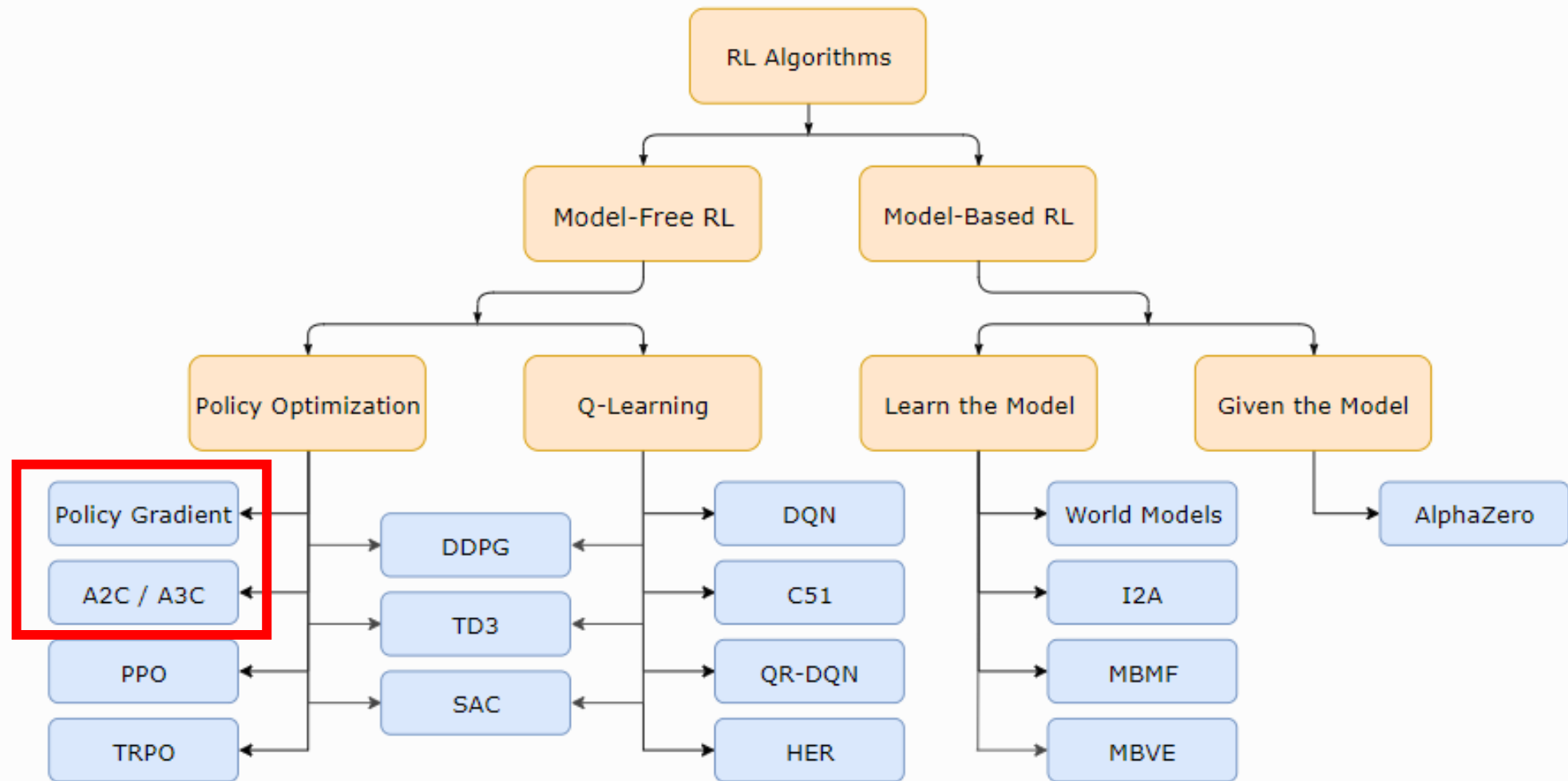
$$\nabla_{\theta} \log P(\tau|\pi) = \nabla_{\theta} \log \pi(a_0|s_0) + \nabla_{\theta} \log \pi(a_1|s_1) + \dots$$

Problem with policy gradients

$$\nabla J(\pi) = E_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) R(\tau) \right]$$

- Monte Carlo updates (i.e. taking random samples) will introduce high variability in log probabilities and cumulative reward values. This will cause unstable learning and/or the policy distribution skewing to a non-optimal direction
- Another problem occurs if we have trajectories whose cumulative reward is 0.
- These issues contribute to the instability and slow convergence of vanilla policy gradient methods.

Reducing sampling variance with actor-critic



A non-exhaustive, but useful taxonomy of algorithms in modern RL. Citations below.

Use expected value to reduce sampling variance

$$\nabla J(\pi) = E_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) R(\tau) \right]$$

REINFORCE (Monte Carlo PG)

$$= E_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) Q^{\pi}(s, a) \right]$$

Q Actor-Critic

$$= E_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) A^{\pi}(s, a) \right]$$

Advantage Actor-Critic

$$= E_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \delta \right]$$

TD Actor-Critic

Q actor-critic

$$\nabla_{\theta} J(\pi_{\theta}) = E_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \Phi_t \right] \quad \Phi_t = \sum_{t'=t}^T \gamma^{t'-t} r_{t'}$$

$$\nabla_{\theta} J(\pi_{\theta}) \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} \nabla \log p_{\theta}(a_t^n | s_t^n) \left(\sum_{t'}^{T_n} \gamma^{t'-t} r_{t'}^n \right)$$

$$G_t^n = \sum_{t'}^{T_n} \gamma^{t'-t} r_{t'}^n$$

unstable when sampling amount is not large enough

$$E[G_t^n] = Q^{\pi_{\theta}}(s_t^n, a_t^n)$$

By definition, the expected value of G_t^n is Q

Reducing sampling variance with a baseline

$$\nabla J(\pi) = E_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) (R(\tau) - b(s_t)) \right]$$

- Making the cumulative reward smaller by subtracting it with a baseline will make smaller gradients, and thus smaller and more stable updates..

Reducing sampling variance with a baseline

$$\begin{aligned}\nabla J(\pi) &= E_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t | s_t) R(\tau) \right] \\ &= E_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t | s_t) (R(\tau) - b(s_t)) \right]\end{aligned}$$

$$\begin{aligned}0 &= \nabla_\theta \int_x P_\theta(x) \\ &= \int_x \nabla_\theta P_\theta(x) \\ &= \int_x P_\theta(x) \nabla_\theta \log P_\theta(x) \\ \therefore 0 &= E_{x \sim P_\theta} [\nabla_\theta \log P_\theta(x)].\end{aligned}$$

$$E_{a_t \sim \pi_\theta} [\nabla_\theta \log \pi_\theta(a_t | s_t) b(s_t)] = 0$$

Advantage actor-critic

$$\nabla_{\theta} J(\pi_{\theta}) = E_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \Phi_t \right] \quad \Phi_t = \sum_{t'=t}^T \gamma^{t'-t} r_{t'} - b(s_t)$$

$$\nabla_{\theta} J(\pi_{\theta}) \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} \nabla \log p_{\theta}(a_t^n | s_t^n) \left(\underbrace{\sum_{t'}^{T_n} \gamma^{t'-t} r_{t'}^n}_{\substack{\downarrow \\ E[G_t^n] = Q^{\pi_{\theta}}(s_t^n, a_t^n)}} - \underbrace{b(s_t^n)}_{\substack{\downarrow \\ E[b(s_t^n)] = V^{\pi_{\theta}}(s_t^n)}} \right)$$

$$\nabla_{\theta} J(\pi_{\theta}) = E_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) (Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)) \right]$$

how much better it is to take a specific action compared to the average, general action at the given state.

Advantage actor-critic

$$\nabla_{\theta} J(\pi_{\theta}) = E_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) (Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)) \right]$$

$$Q^{\pi}(s_t, a_t) = E[r(s_t, a_t) + \gamma V^{\pi}(s_{t+1})]$$

$$\begin{aligned} A^{\pi}(s_t, a_t) &= Q^{\pi}(s_t, a_t) - V^{\pi}(s_t) \\ &= r(s_t, a_t) + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t) \end{aligned}$$

$$\nabla_{\theta} J(\pi_{\theta}) = E_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) (r(s_t, a_t) + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)) \right]$$

Advantage actor-critic

REINFORCE (Monte Carlo PG)

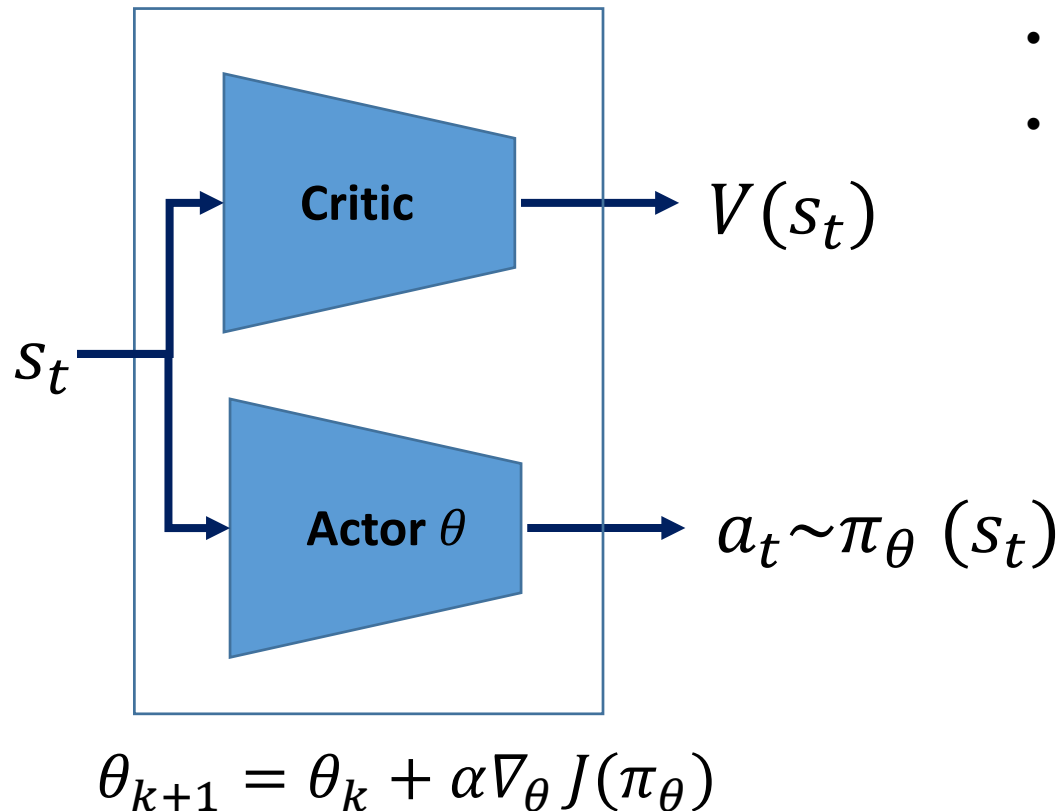
$$\nabla_{\theta} J(\pi_{\theta}) = E_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \Phi_t \right] \quad \Phi_t = \sum_{t'=t}^T \gamma^{t'-t} r_{t'}$$

Advantage Actor-Critic

$$\begin{aligned} \nabla_{\theta} J(\pi_{\theta}) &= E_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) (Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)) \right] \\ &= E_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) (r(s_t, a_t) + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)) \right] \end{aligned}$$

Advantage actor-critic (A2C)

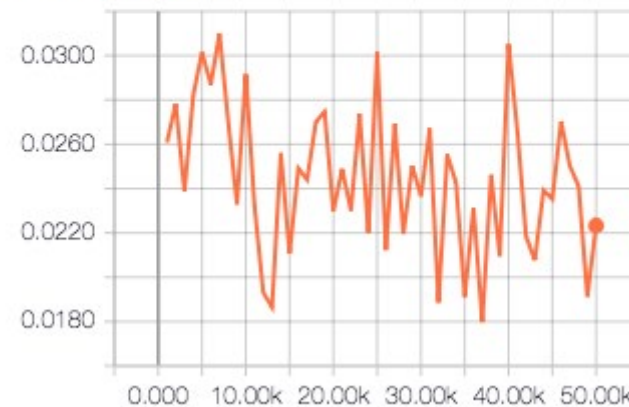
$$\nabla_{\theta} J(\pi_{\theta}) = E_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) (r(s_t, a_t) + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)) \right]$$



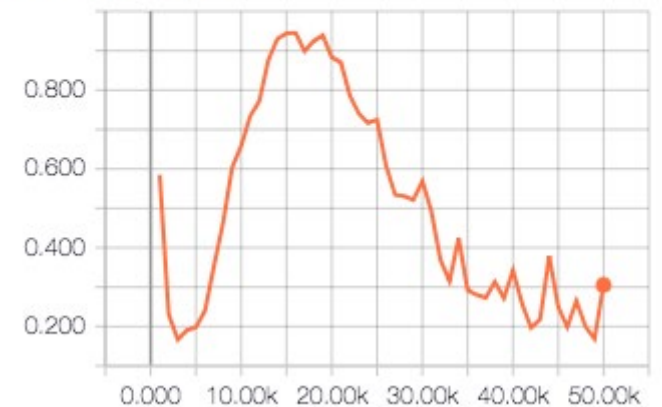
- The “Critic” estimates the value function.
- The “Actor” updates the policy distribution in the direction suggested by the Critic.

Losses

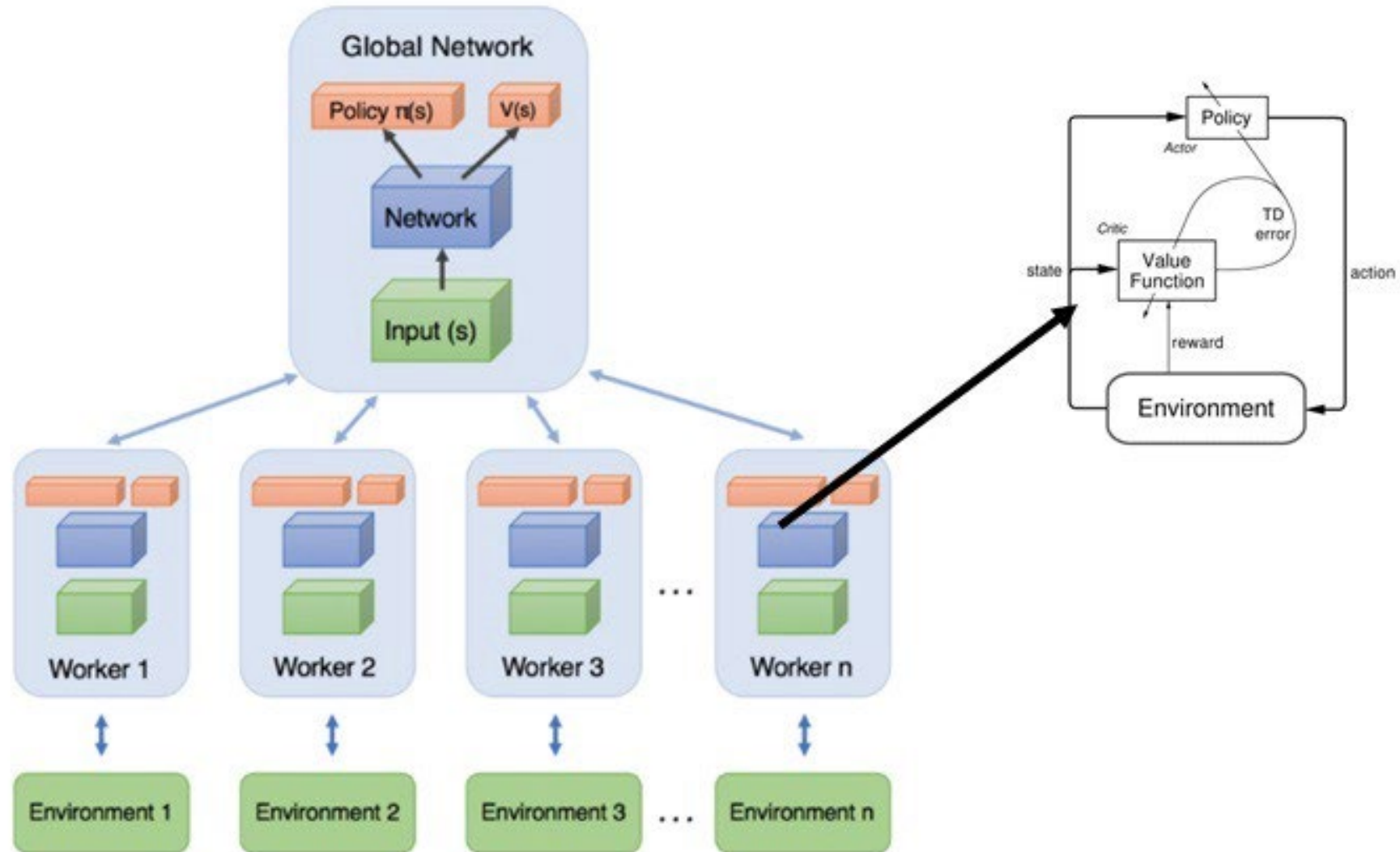
Losses/Policy Loss



Losses/Value Loss



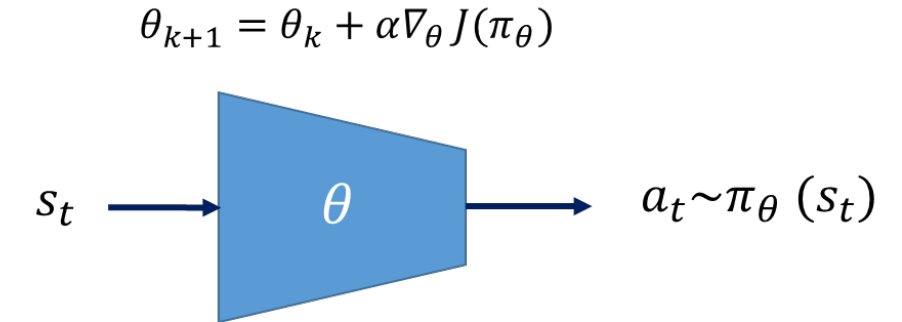
Asynchronous Advantage Actor Critic (A3C)



Sampling efficiency problem

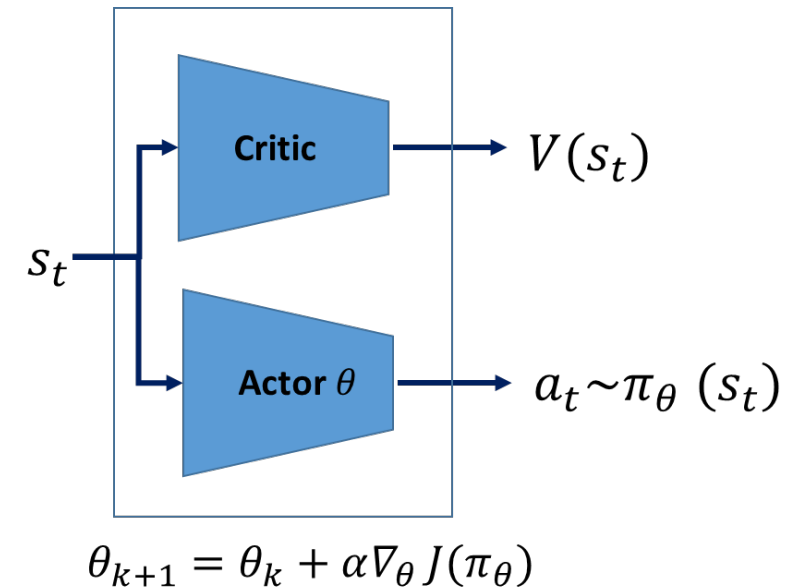
REINFORCE (Monte Carlo PG)

$$\nabla_{\theta} J(\pi_{\theta}) = E_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \Phi_t \right] \quad \Phi_t = \sum_{t'=t}^T \gamma^{t'-t} r_{t'}$$



Advantage Actor-Critic (A2C)

$$\begin{aligned} \nabla_{\theta} J(\pi_{\theta}) &= E_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) (Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)) \right] \\ &= E_{\tau \sim \pi_{\theta}} [\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) (r(s_t, a_t) + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t))] \end{aligned}$$



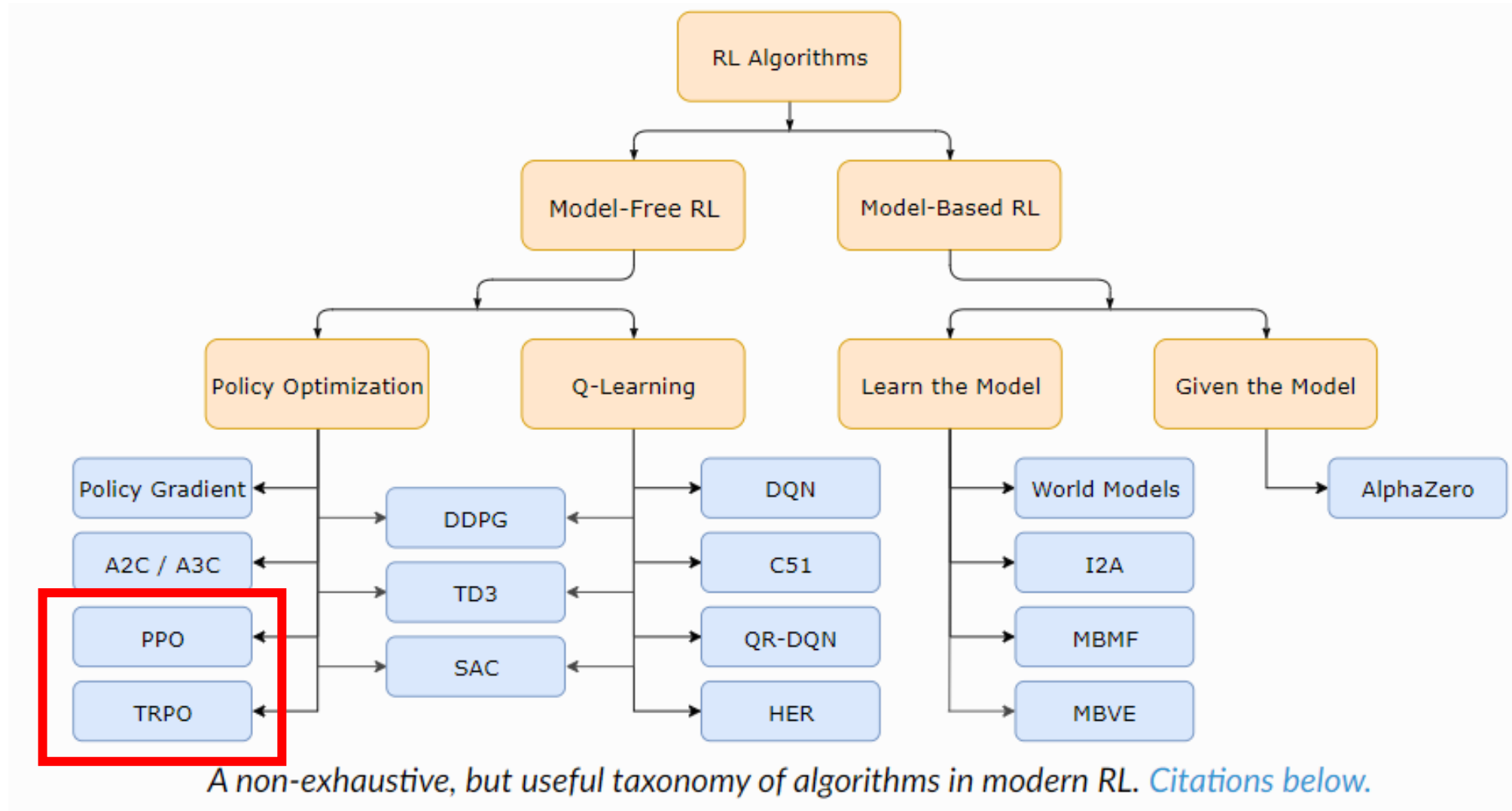
Important sampling

$$E_{x \sim p}[f(x)] = \int f(x)p(x)dx = \int f(x) \frac{p(x)}{q(x)} q(x)dx = E_{x \sim q} \left[f(x) \frac{p(x)}{q(x)} \right]$$

$$Var_{x \sim p}[f(x)] = E_{x \sim p}[f(x)^2] - (E_{x \sim p}[f(x)])^2 \quad \text{VAR}[X] = E(X^2) - (E[X])^2$$

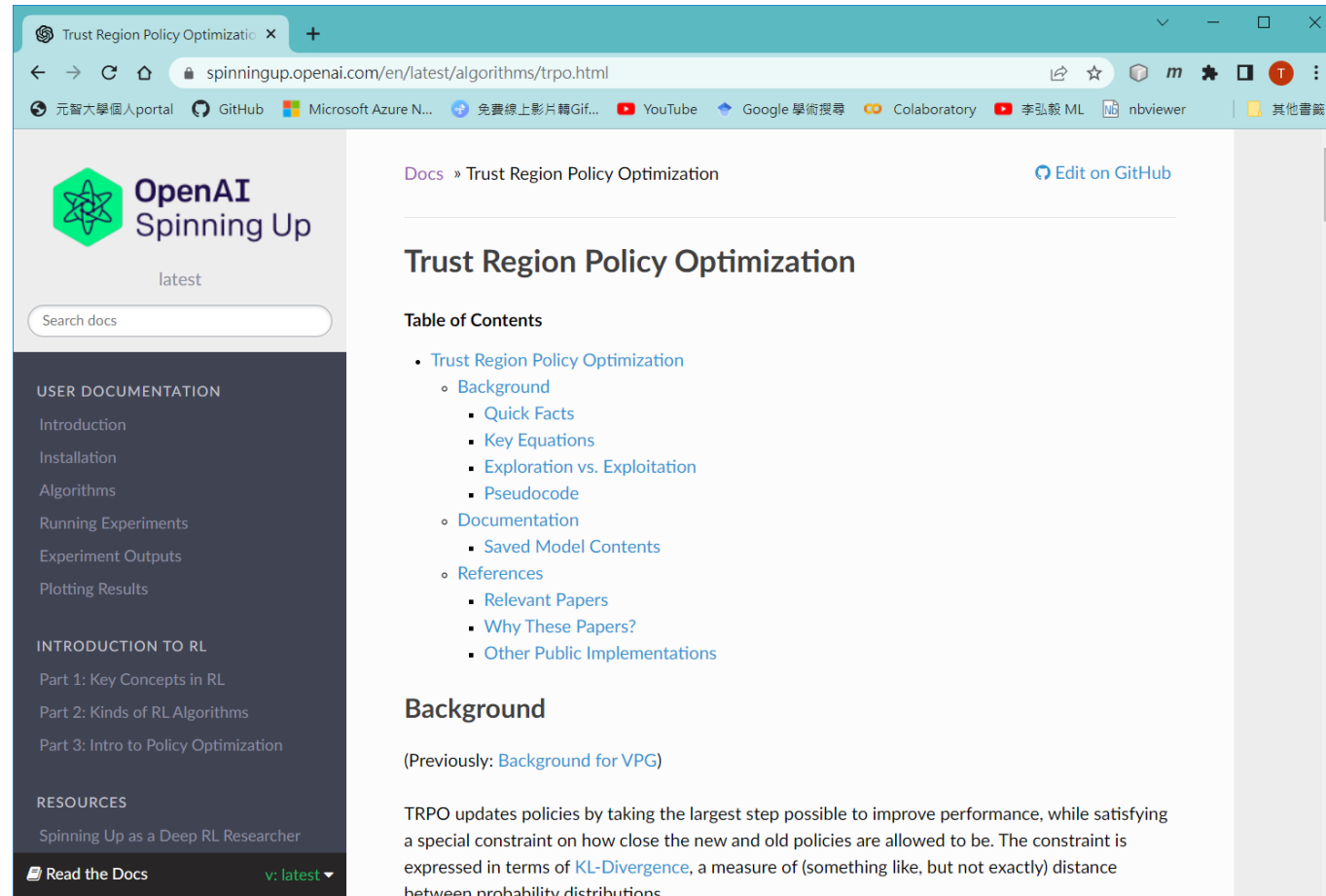
$$\begin{aligned} Var_{x \sim q} \left[f(x) \frac{p(x)}{q(x)} \right] &= E_{x \sim q} \left[\left(f(x) \frac{p(x)}{q(x)} \right)^2 \right] - \left(E_{x \sim q} \left[f(x) \frac{p(x)}{q(x)} \right] \right)^2 \\ &= E_{x \sim p} \left[f(x)^2 \frac{p(x)}{q(x)} \right] - (E_{x \sim p}[f(x)])^2 \end{aligned}$$

Solving sampling efficiency problem



TRPO

How can we make policy optimization more sampling efficient?



The screenshot shows a web browser window displaying the OpenAI Spinning Up documentation for Trust Region Policy Optimization (TRPO). The page is titled "Trust Region Policy Optimization" and includes a "Table of Contents" with links to various sections. The left sidebar contains a navigation menu with categories like "USER DOCUMENTATION", "INTRODUCTION TO RL", and "RESOURCES". The main content area shows the "Background" section, which explains that TRPO updates policies by taking the largest step possible to improve performance while satisfying a special constraint on how close the new and old policies are allowed to be. The constraint is expressed in terms of KL-Divergence, a measure of (something like, but not exactly) distance between probability distributions.

Trust Region Policy Optimization

Docs » Trust Region Policy Optimization [Edit on GitHub](#)

Trust Region Policy Optimization

Table of Contents

- Trust Region Policy Optimization
 - Background
 - Quick Facts
 - Key Equations
 - Exploration vs. Exploitation
 - Pseudocode
 - Documentation
 - Saved Model Contents
 - References
 - Relevant Papers
 - Why These Papers?
 - Other Public Implementations

Background

(Previously: [Background for VPG](#))

TRPO updates policies by taking the largest step possible to improve performance, while satisfying a special constraint on how close the new and old policies are allowed to be. The constraint is expressed in terms of [KL-Divergence](#), a measure of (something like, but not exactly) distance between probability distributions.

[Welcome to Spinning Up in Deep RL! — Spinning Up documentation \(openai.com\)](https://openai.com/spinningup/)

TRPO

$$\max J(\pi) = E_{\tau \sim \pi} (R(\tau))$$

$$\theta_{k+1} = \theta_k + \alpha \nabla_{\theta} J(\pi_{\theta})$$

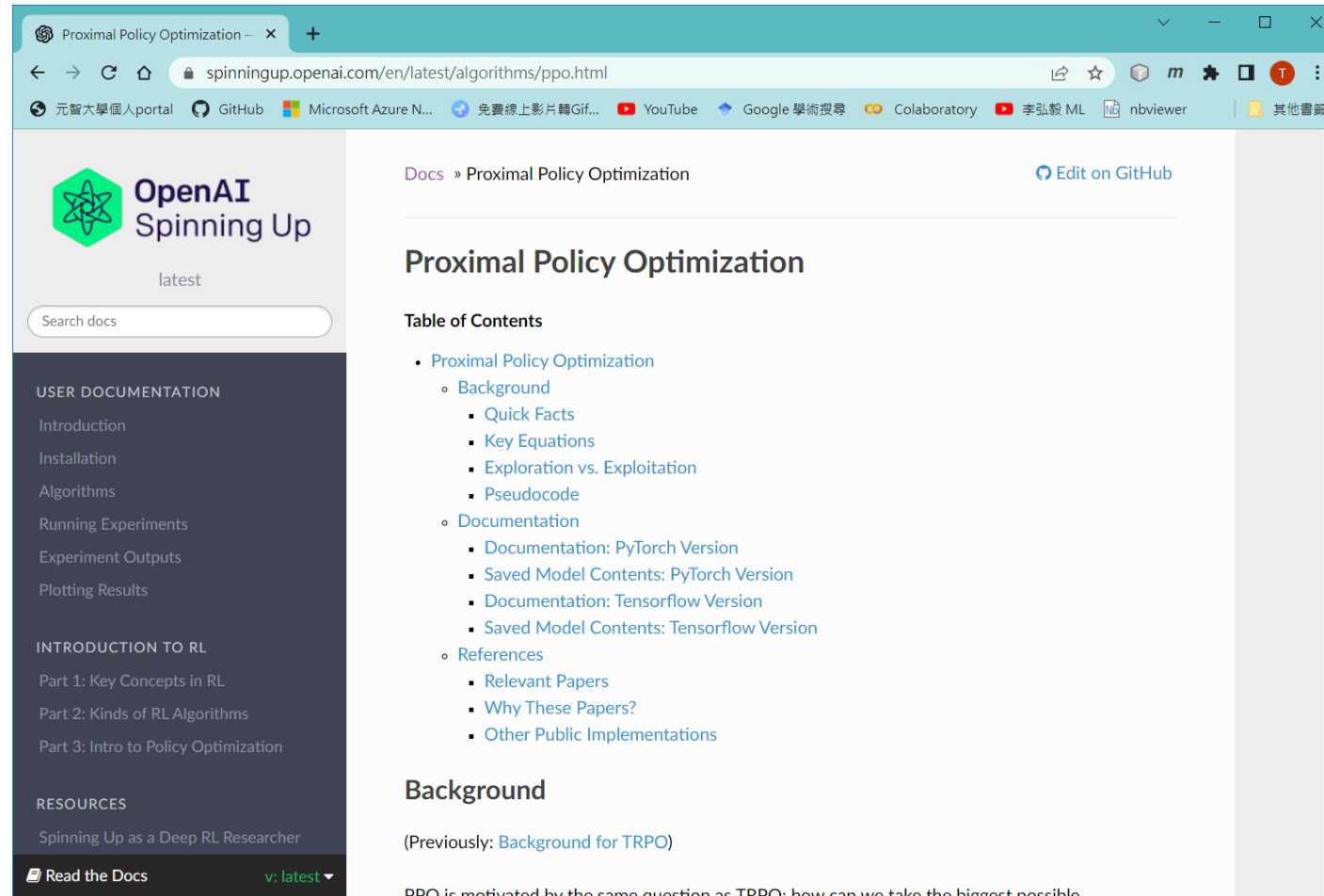
$$\begin{aligned} \theta_{k+1} &= \arg \max_{\theta} \mathcal{L}(\theta_k, \theta) \\ \text{s.t. } \bar{D}_{\text{KL}}(\theta \parallel \theta_k) &\leq \delta \end{aligned}$$

$$\mathcal{L}(\theta_k, \theta) = E_{s, a \sim \pi_{\theta_k}} \left[\frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta}}(s, a) \right]$$

$$\bar{D}_{\text{KL}}(\theta \parallel \theta_k) = E_{s \sim \pi_{\theta_k}} [D_{\text{KL}}(\pi_{\theta}(\cdot | s) \parallel \pi_{\theta_k}(\cdot | s))]$$

PPO

How can we make policy optimization more sampling efficient?



[Welcome to Spinning Up in Deep RL! — Spinning Up documentation \(openai.com\)](https://spinningup.openai.com/en/latest/algorithms/ppo.html)

$$\max J(\pi) = E_{\tau \sim \pi} (R(\tau))$$

$$\theta_{k+1} = \theta_k + \alpha \nabla_{\theta} J(\pi_{\theta})$$

$$\theta_{k+1} = \arg \max_{\theta} E_{s, a \sim \pi_{\theta_k}} [L(s, a, \theta_k, \theta)]$$

$$[L(s, a, \theta_k, \theta)] = \min \left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), \text{clip} \left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)}, 1 - \epsilon, 1 + \epsilon \right) A^{\pi_{\theta_k}}(s, a) \right)$$

HW 3

- Use PPO to train walker
- Change reward setting to let walker have different behavior