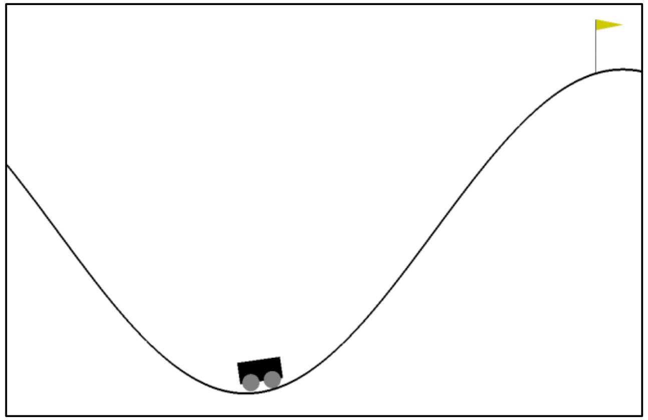
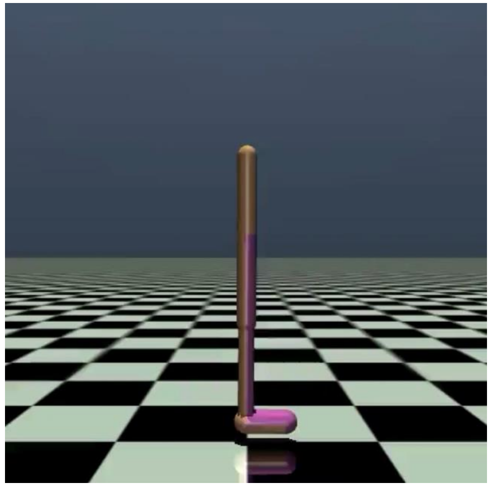
ETH zürich

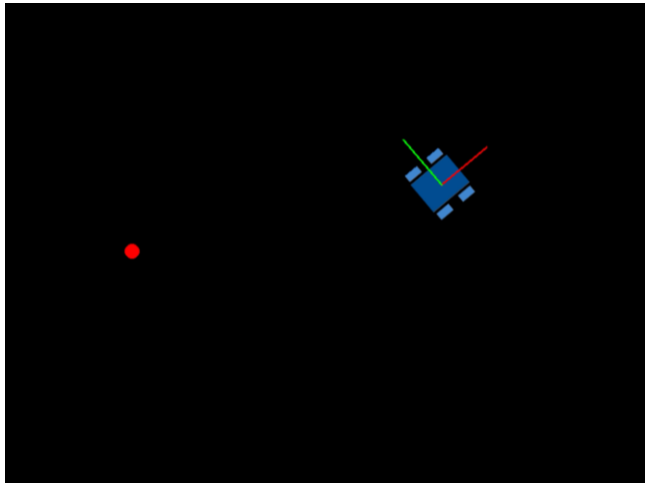
Deep Reinforcement Learning

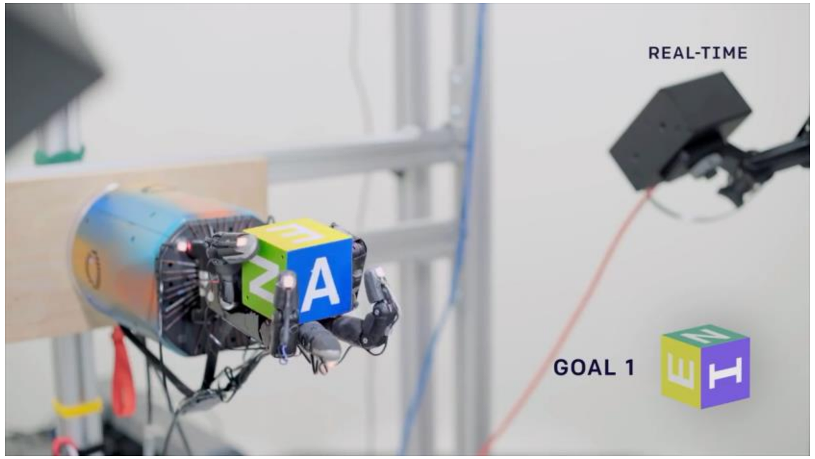
Núria Armengol Urpí

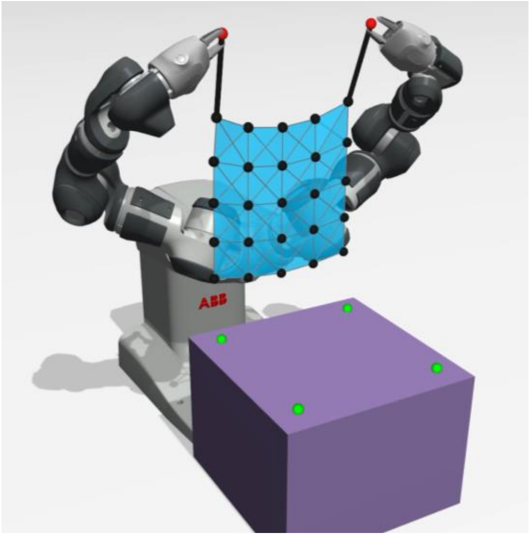
19.05.2021








REAL-TIME
GOAL 1



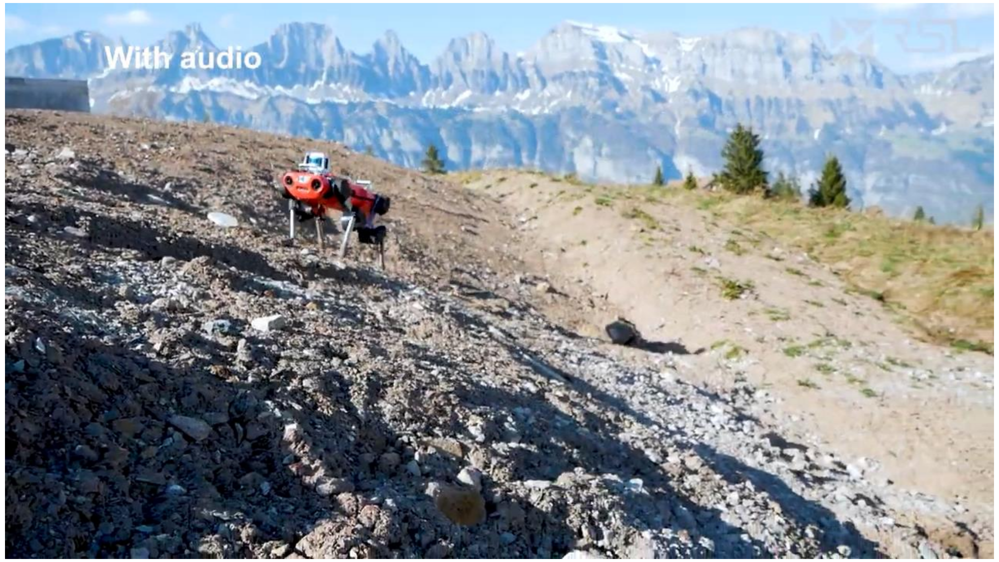
ABB


ALPHAGO
00:05:30
Google DeepMind
Challenge Match
LEE SEDOL
00:28:28

Examples

Learning from interaction

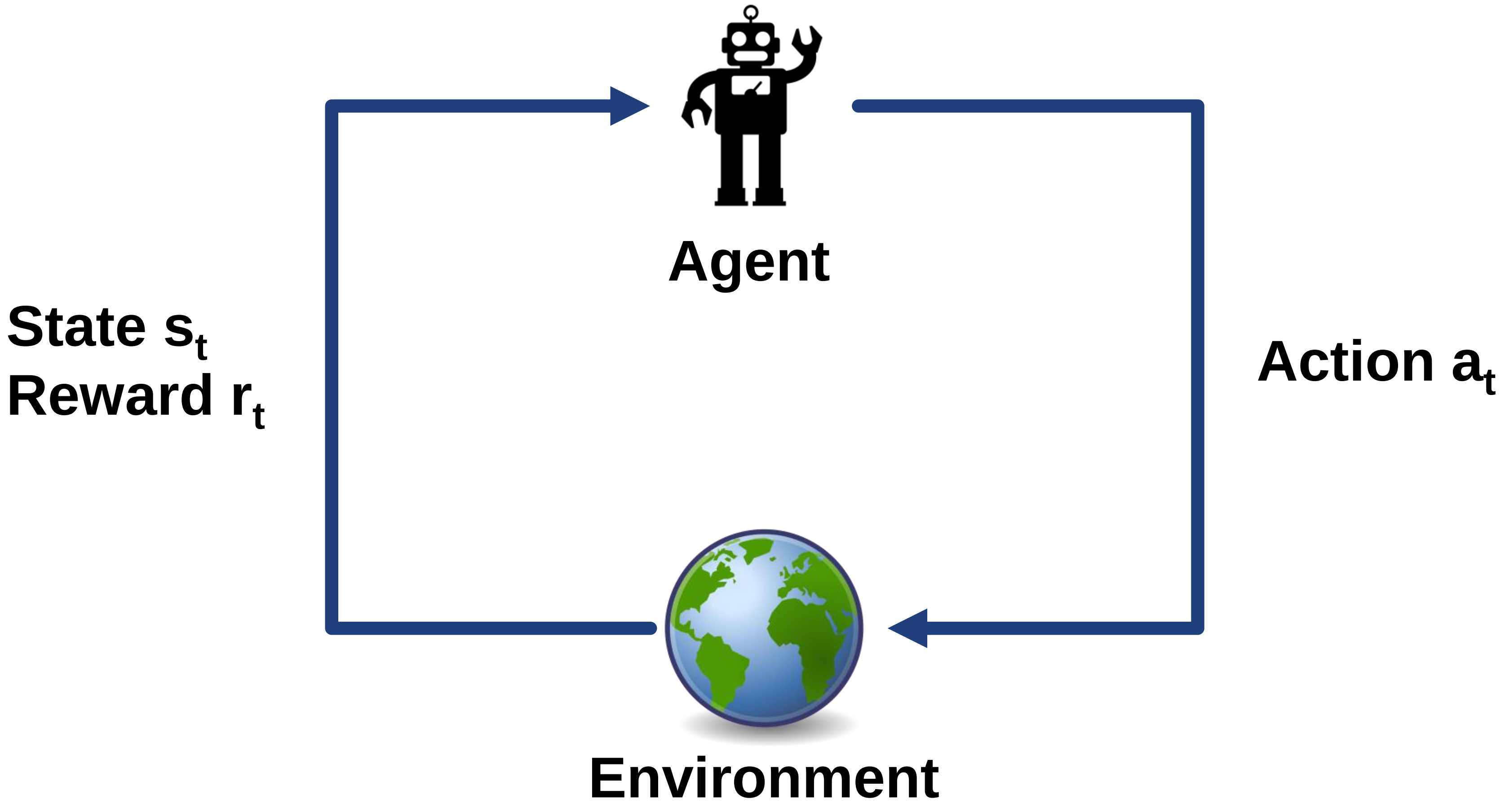
When do we want to use Reinforcement Learning (RL)?



- Sequential decision making problem
- Do **not** know OPTIMAL behaviour yet
- Can evaluate whether behaviours are ‘good’ or ‘bad’

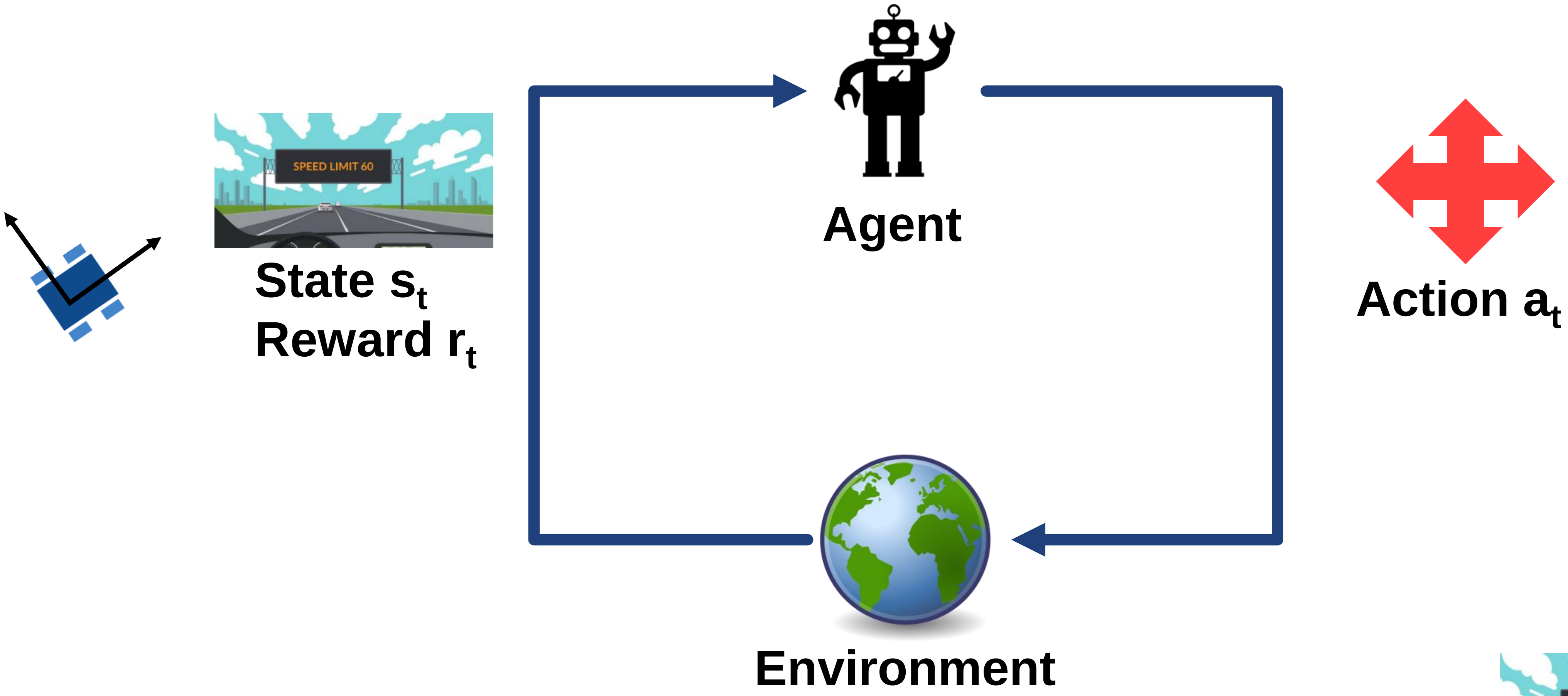
RL is useful when evaluating behaviours is easier than generating them.

Reinforcement Learning basics



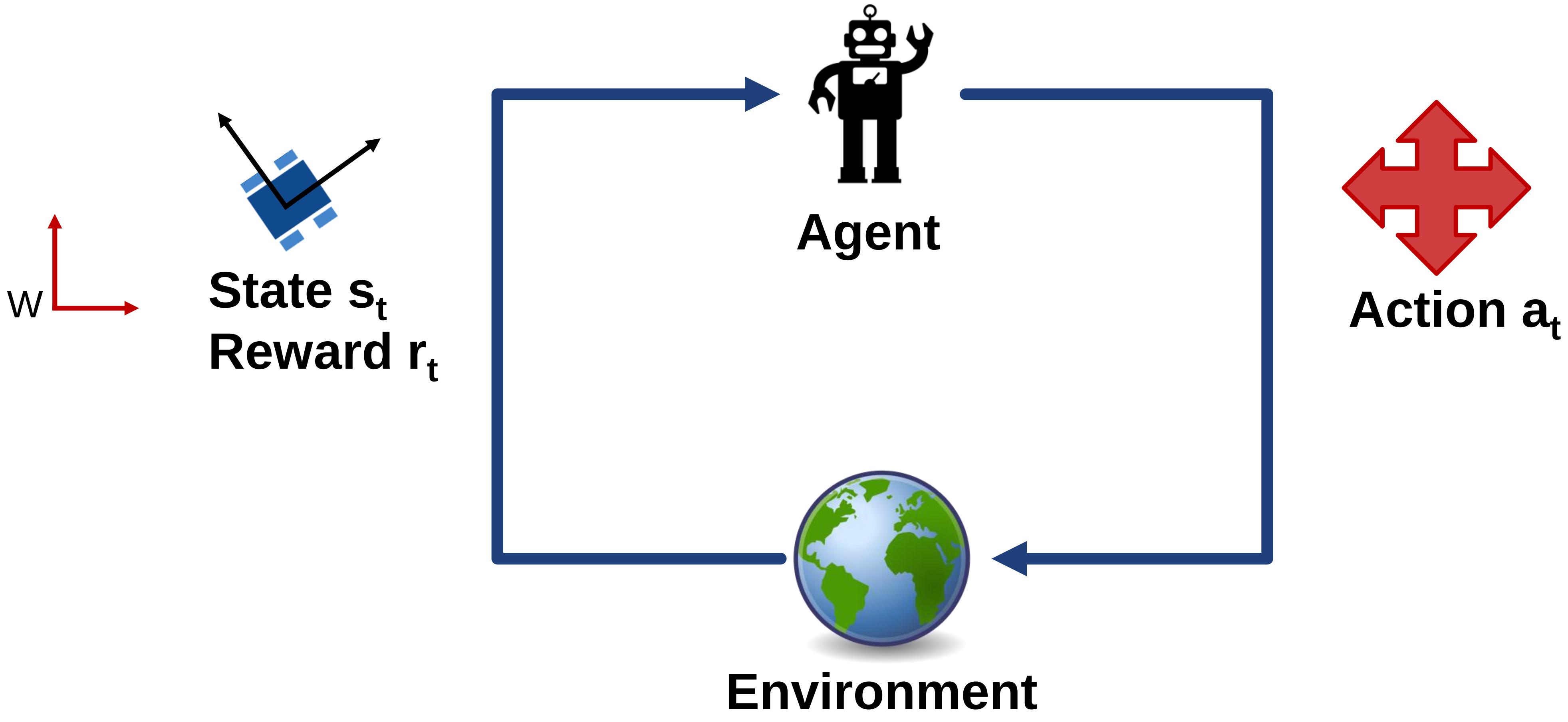
RL interaction loop

Reinforcement Learning basics



Observation: what the agent sees about the current state of the world. Ex: bunch of pixels

Reinforcement Learning basics

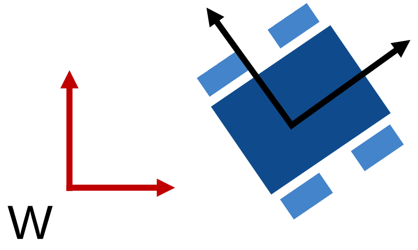


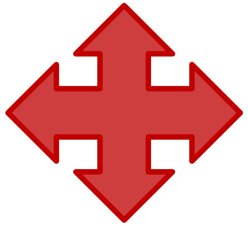
Observation: what the agent sees about the current state of the world

State: complete description of the world. Ex: $s_t : [x_t, y_t, v_{x_t}, v_{y_t}, \alpha_t]$

The policy

$$s_t : [x, y, v_x, v_y, \alpha]$$





Action a_t

Policy π: Rule for selecting actions

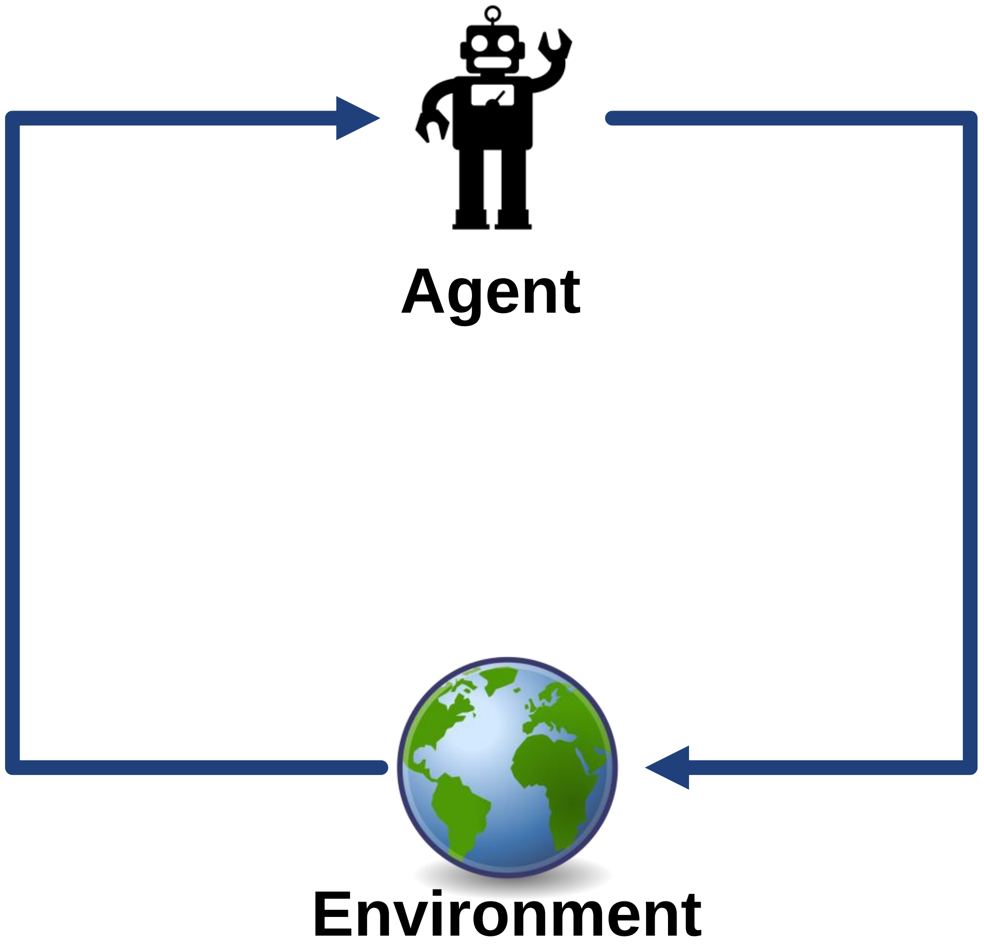
Deterministic $a_t = \pi(s_t)$

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

The reward

State s_t

Reward r_t

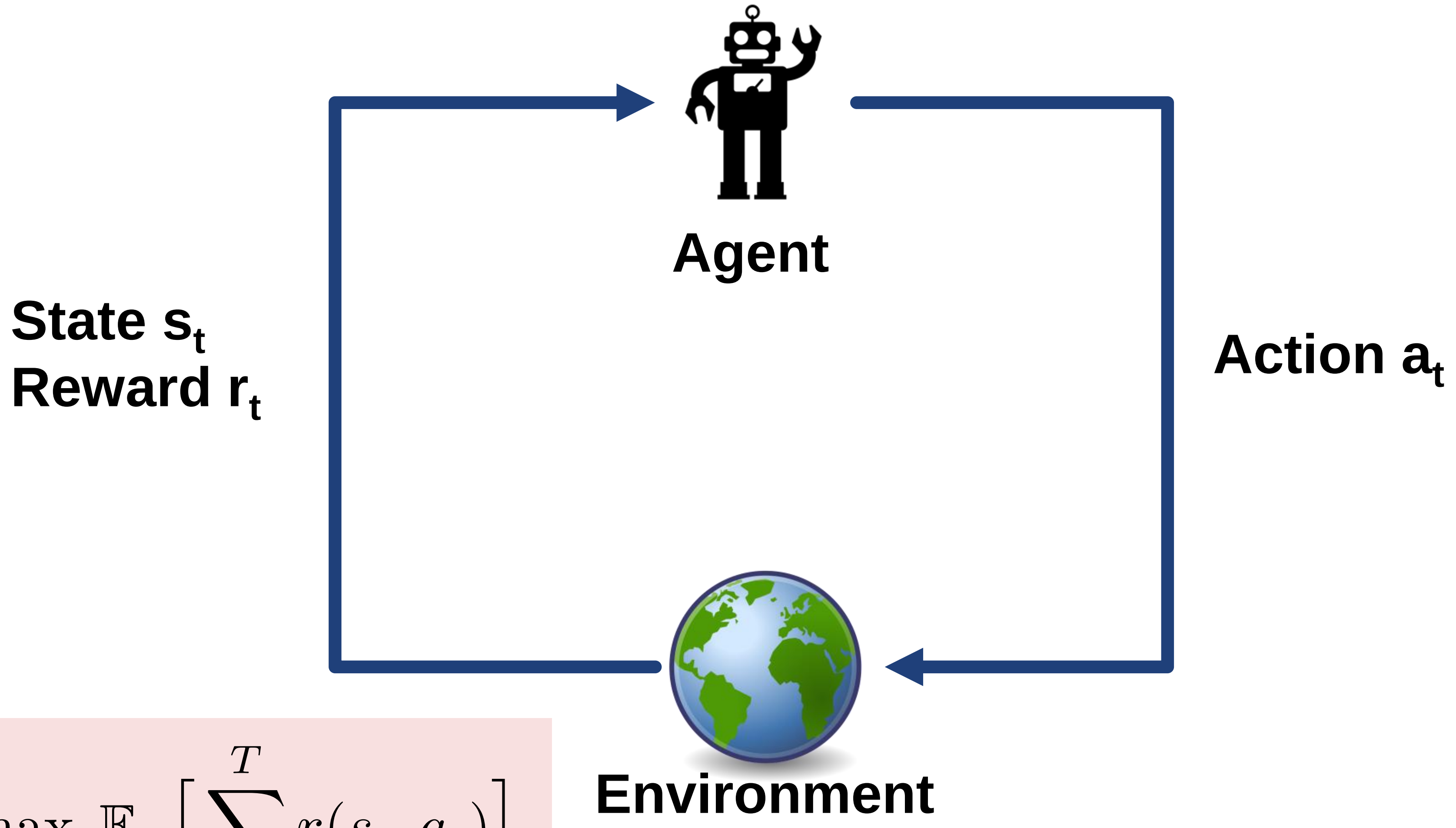


Action a_t

$$r \in \mathcal{R}$$

$$\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$$

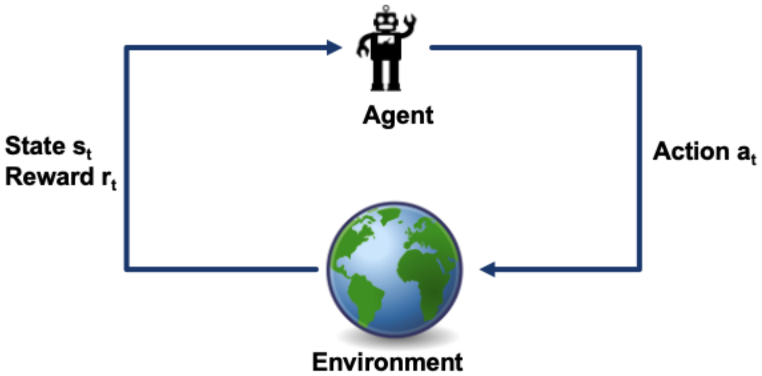
The RL objective



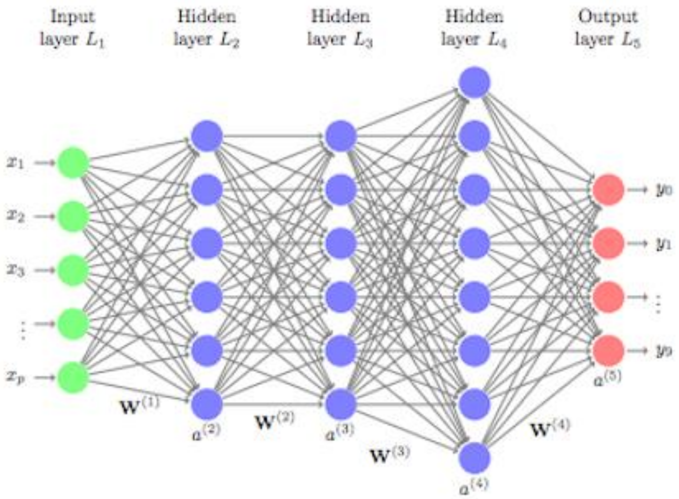
$$\pi^* = \arg\max_{\pi} \; \mathbb{E}_{\pi}\left[\sum_{t=1}^{T} r(s_t, a_t)\right]$$

What is deep RL?

- Combination of Reinforcement Learning (RL) with deep learning



RL interaction loop
Solve a sequential decision making task by interaction with the environment



Deep neural network (Deep NN)

Deep RL: Train a NN to solve a sequential decision-making task by interacting with the environment.

CRL

When do we want to use deep learning?

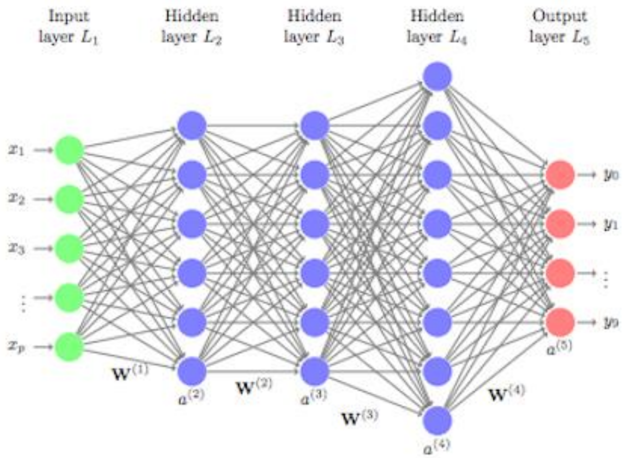
- ‘Deep’ refers to using function composition as the building block for the model

- Represent the model as a function of parameters

    For ex. for a 2 layer feed-forward NN:

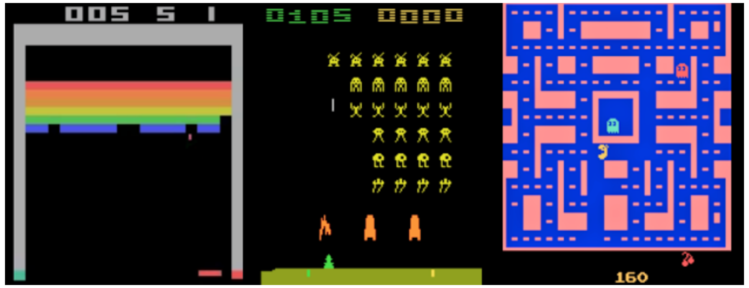
$$y(x;\theta) = W_2\sigma(W_1 x + b_1) + b_2 \qquad \theta = \{W_1, W_2, b_1, b_2\}$$

Non-linearity



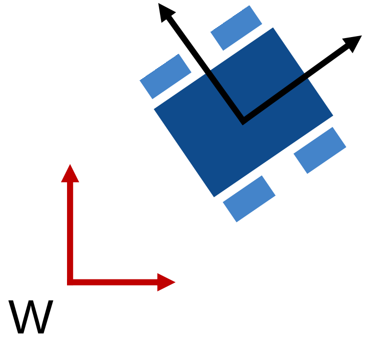
- Approximate a complex function

- Inputs and/or outputs are high-dimensional

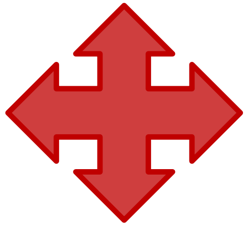


The policy

$$s_t : [x, y, v_x, v_y, \alpha] \longrightarrow$$



$\longrightarrow$



Action a_t

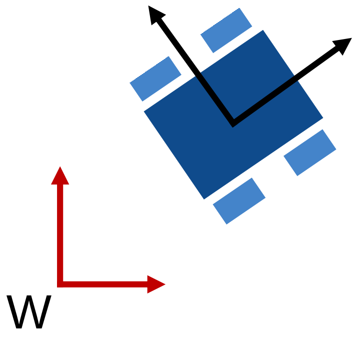
Policy π: Rule for selecting actions

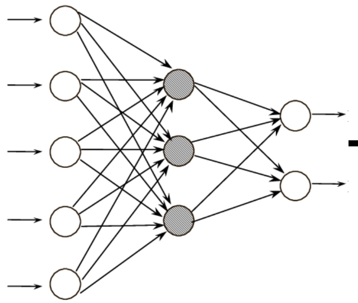
Deterministic $a_t = \pi(s_t)$

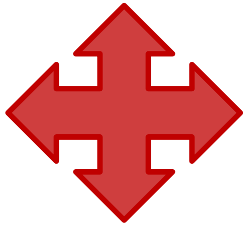
Stochastic $a_t \sim \pi(\cdot|s_t)$

Reinforcement learning basics

$$s_t : [x, y, v_x, v_y, \alpha]$$







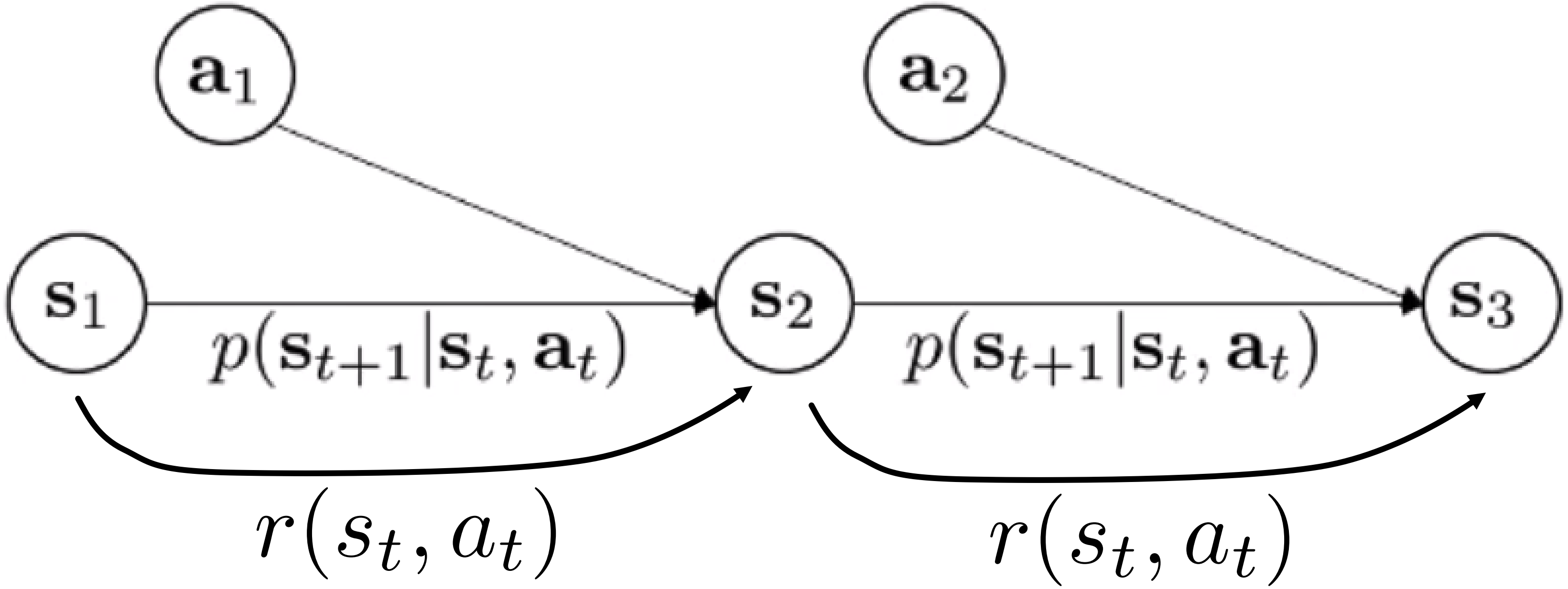
Action a_t

Policy π_θ: Rule for selecting actions

Deterministic $a_t = \pi_\theta(s_t)$

Stochastic $a_t \sim \pi_\theta(\cdot|s_t)$

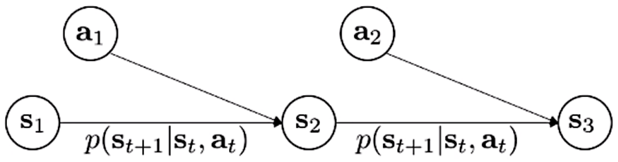
Markov decision processes (MDPs)



- Mathematical formulation of the agent-environment interaction
- Discrete-time stochastic control process

CRL

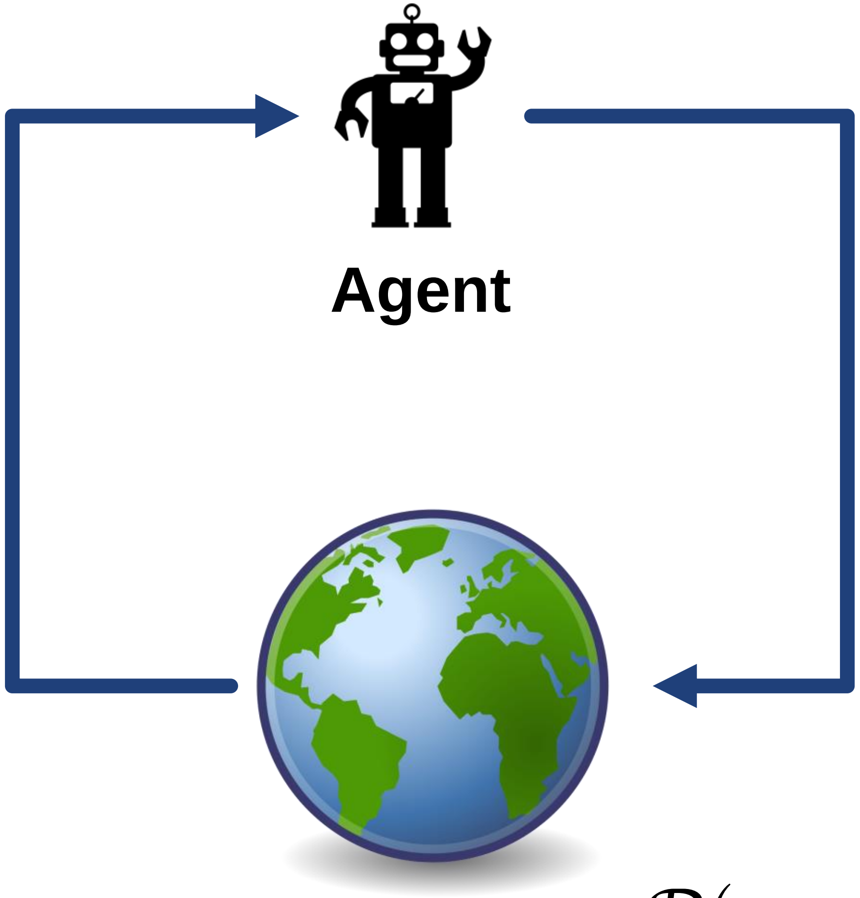
Markov decision process



- Markov decision process $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, d_1\}$
- $\mathcal{S}$ – state space Set of all valid states $s \in \mathcal{S}$ (discrete or continuous)
- $\mathcal{A}$ – action space Set of valid actions actions $a \in \mathcal{A}$ (discrete or continuous)
- $\mathcal{P}$ – Transition operator Describes the dynamics of the system $\mathcal{P}(s_{t+1}|s_t, a_t)$
- $\mathcal{R}$ – reward function Describes a a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$
- d_1 - Initial state distribution

System obeys the **Markov property:** transitions only depend on the most recent state and action, and no prior history.

Reinforcement Learning basics

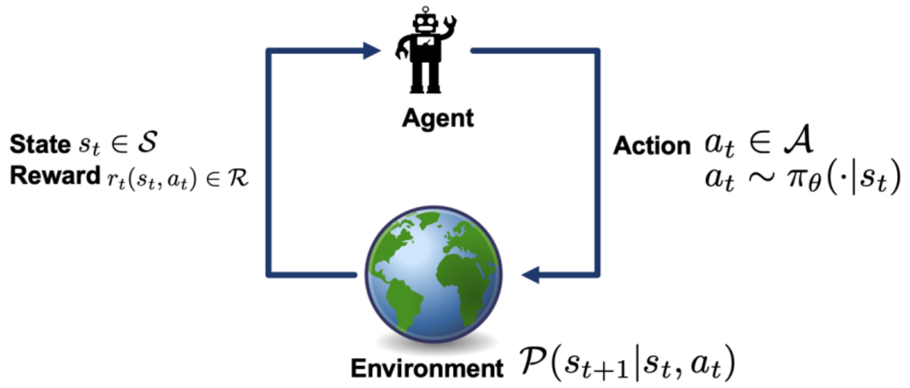


State $s_t \in \mathcal{S}$
Reward $r_t(s_t, a_t) \in \mathcal{R}$

Action $a_t \in \mathcal{A}$
$a_t \sim \pi_\theta(\cdot|s_t)$

Environment $\mathcal{P}(s_{t+1}|s_t, a_t)$

The RL objective



Trajectory $\tau = (s_1, a_1, \ldots, s_T, a_T)$

$$p_{\pi_\theta}(\tau) = d_1(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

$$\theta^* = \arg\max_\theta \mathbb{E}_{\tau \sim p_{\pi_\theta}(\tau)} \underbrace{\left[\sum_{t=1}^{T} r(s_t, a_t) \right]}_{r(\tau)}$$

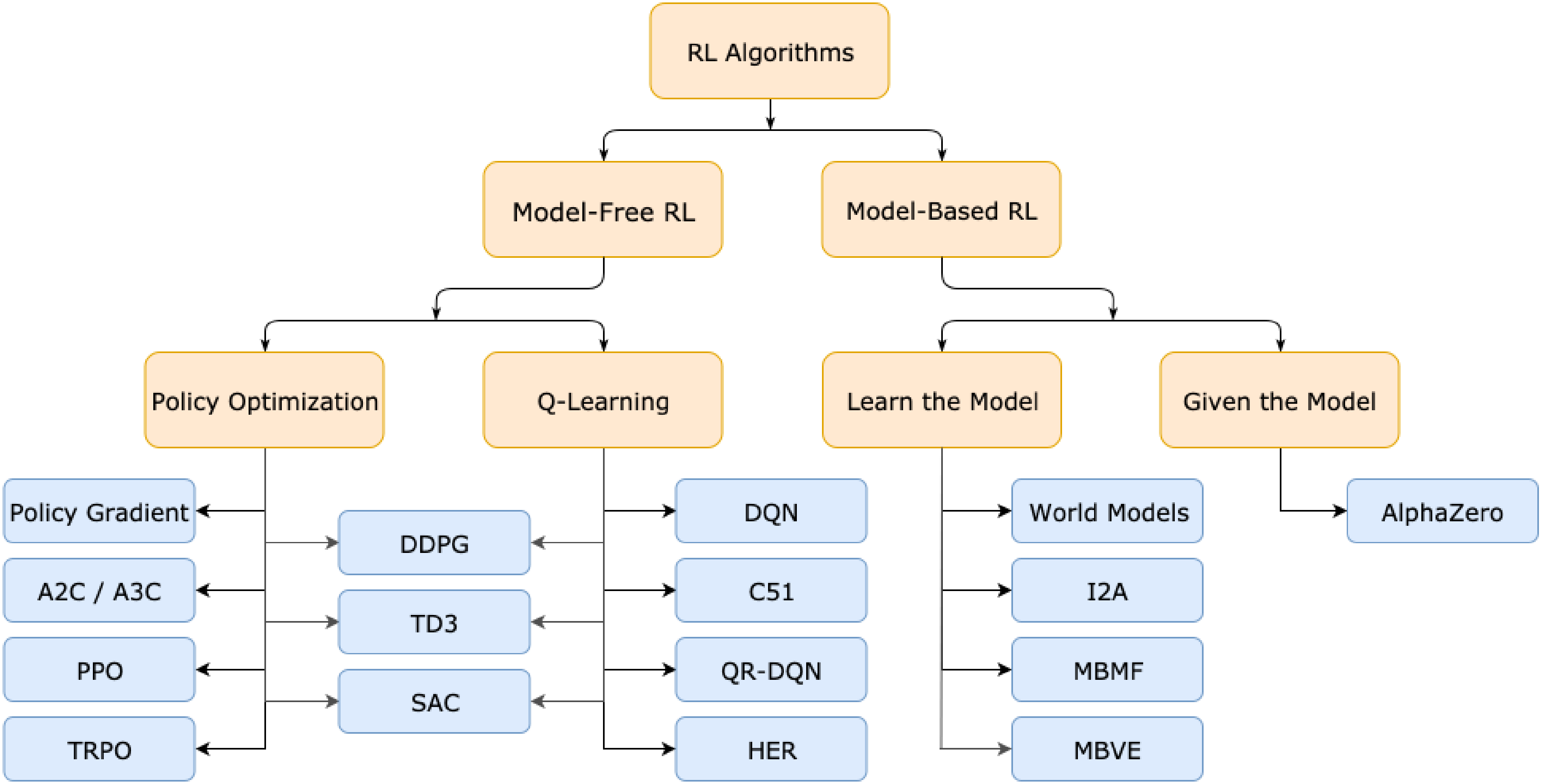
$E(\theta)$

RL objective

: Return or Cumulative reward

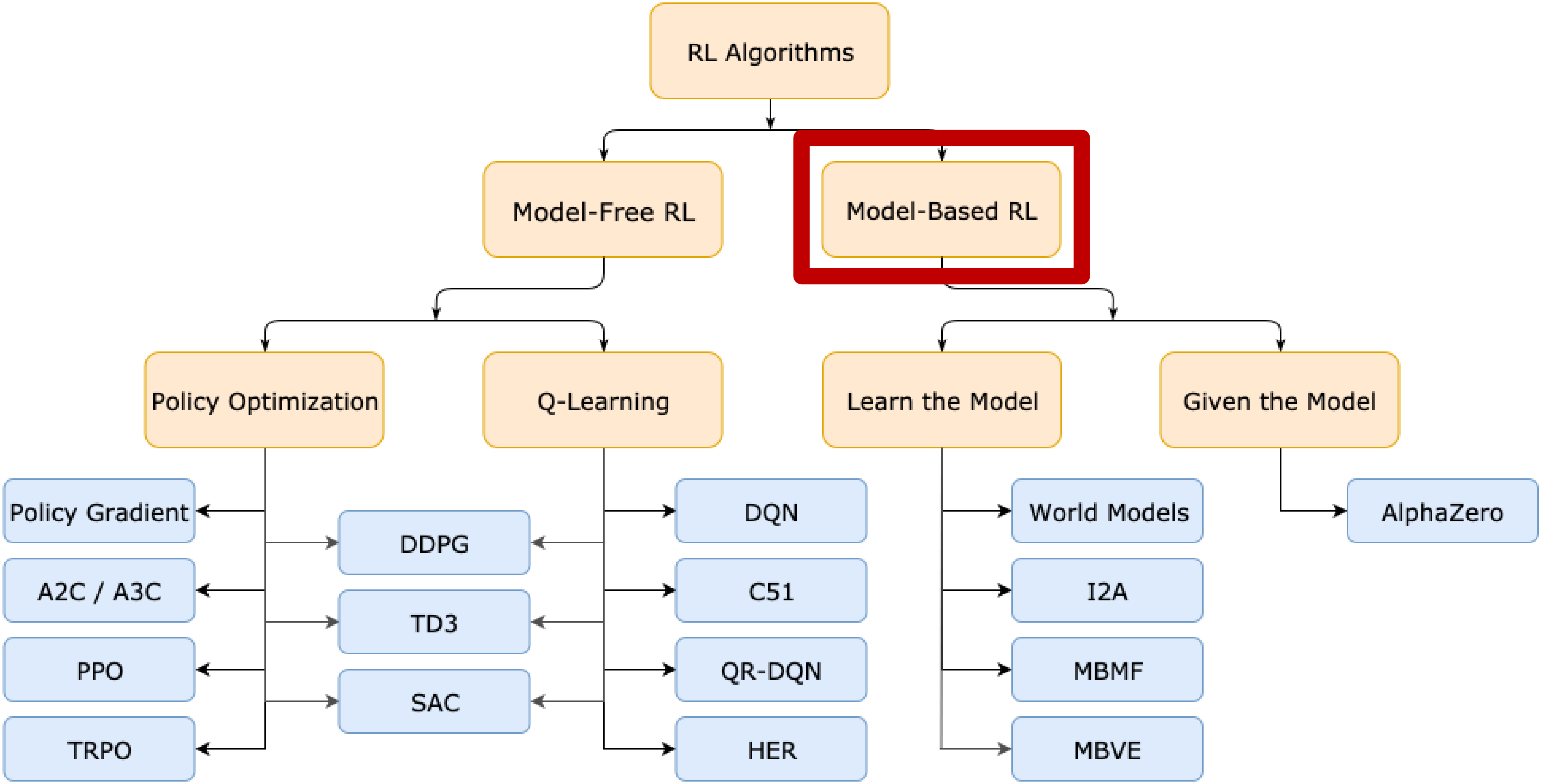
CRL

A Taxonomy of RL algorithms

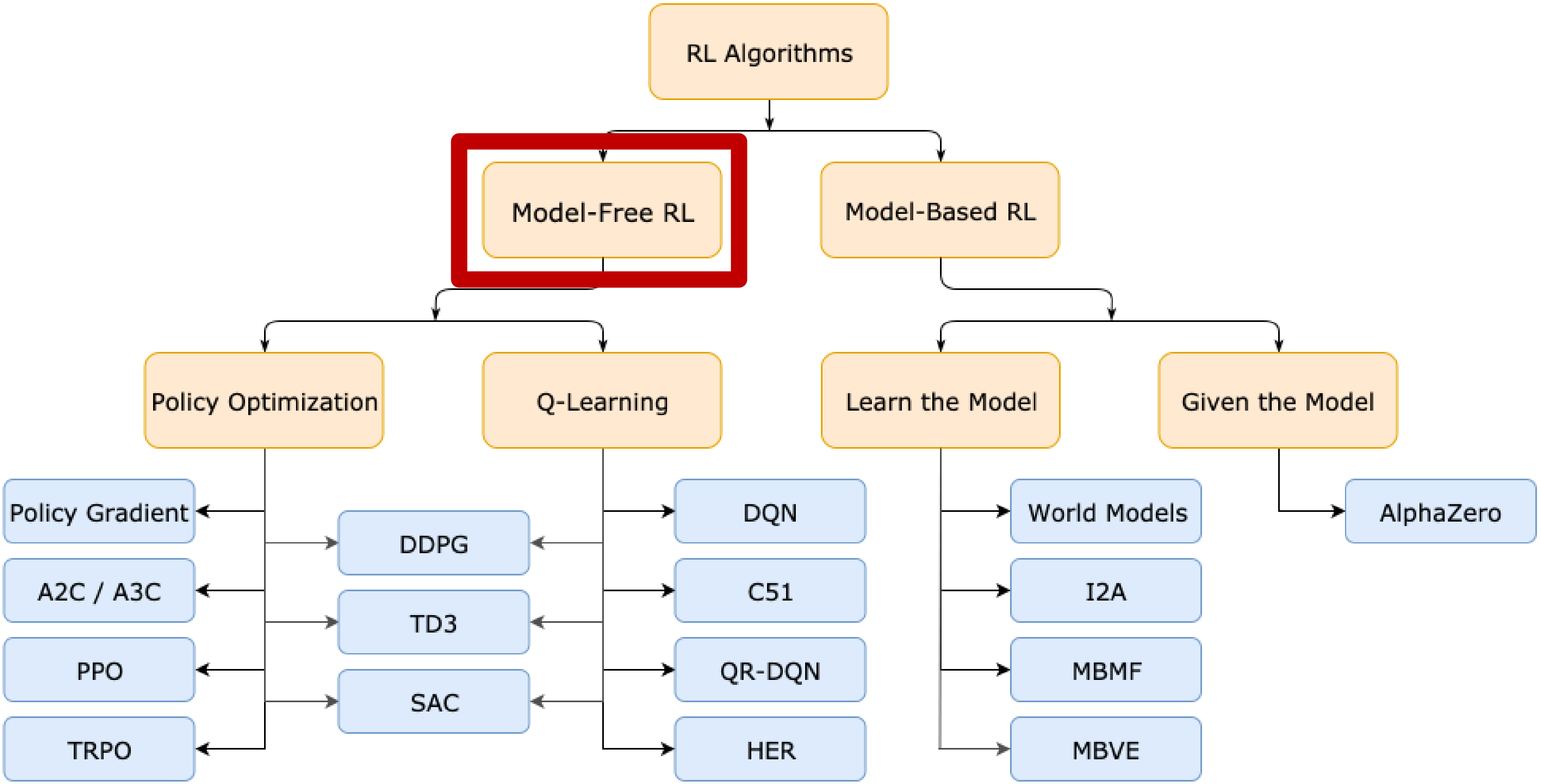


Source: Spinning up Documentation.
https://spinningup.openai.com/

A Taxonomy of RL algorithms



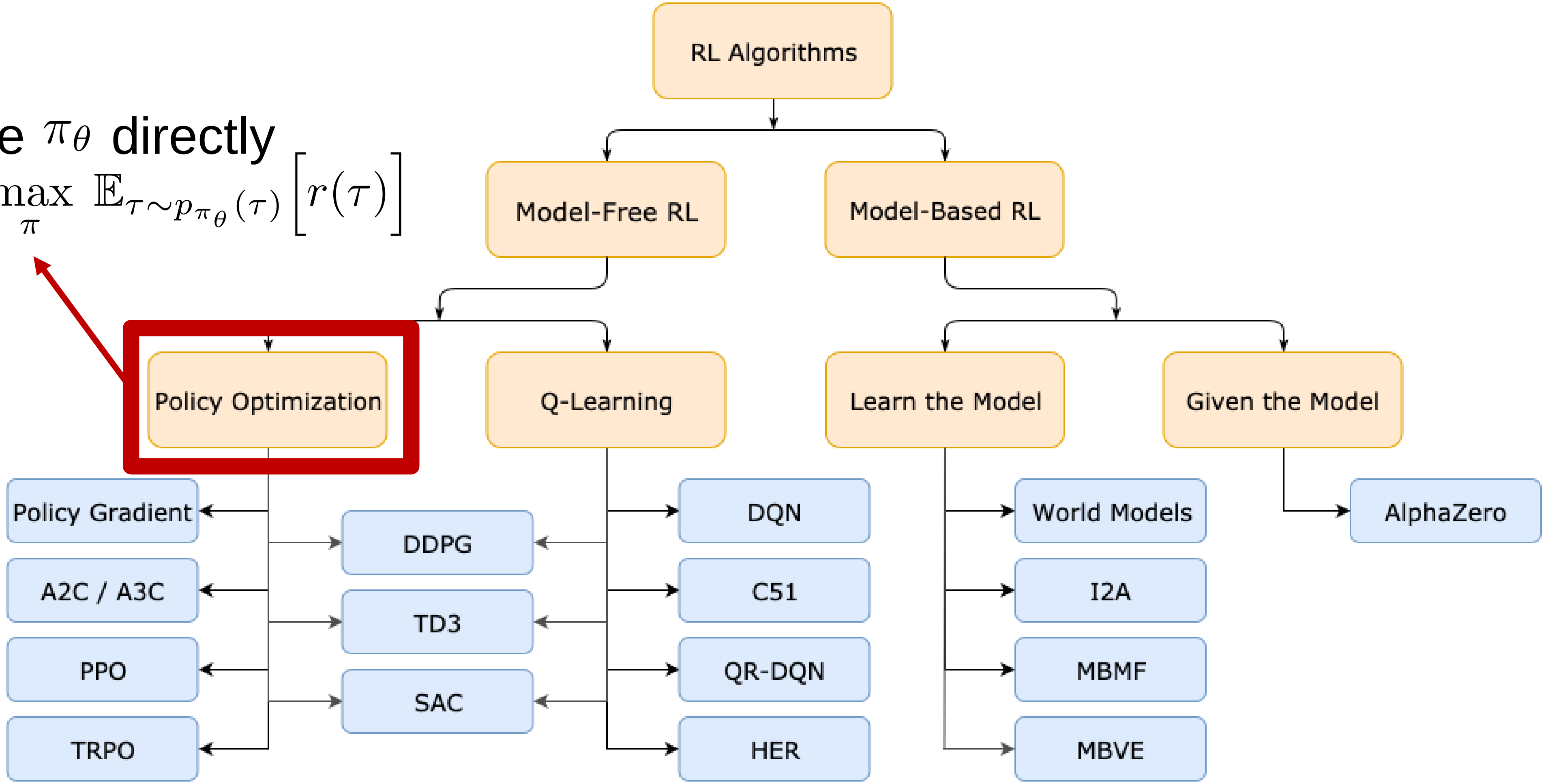
A Taxonomy of RL algorithms



A Taxonomy of RL algorithms

Optimize π_θ directly

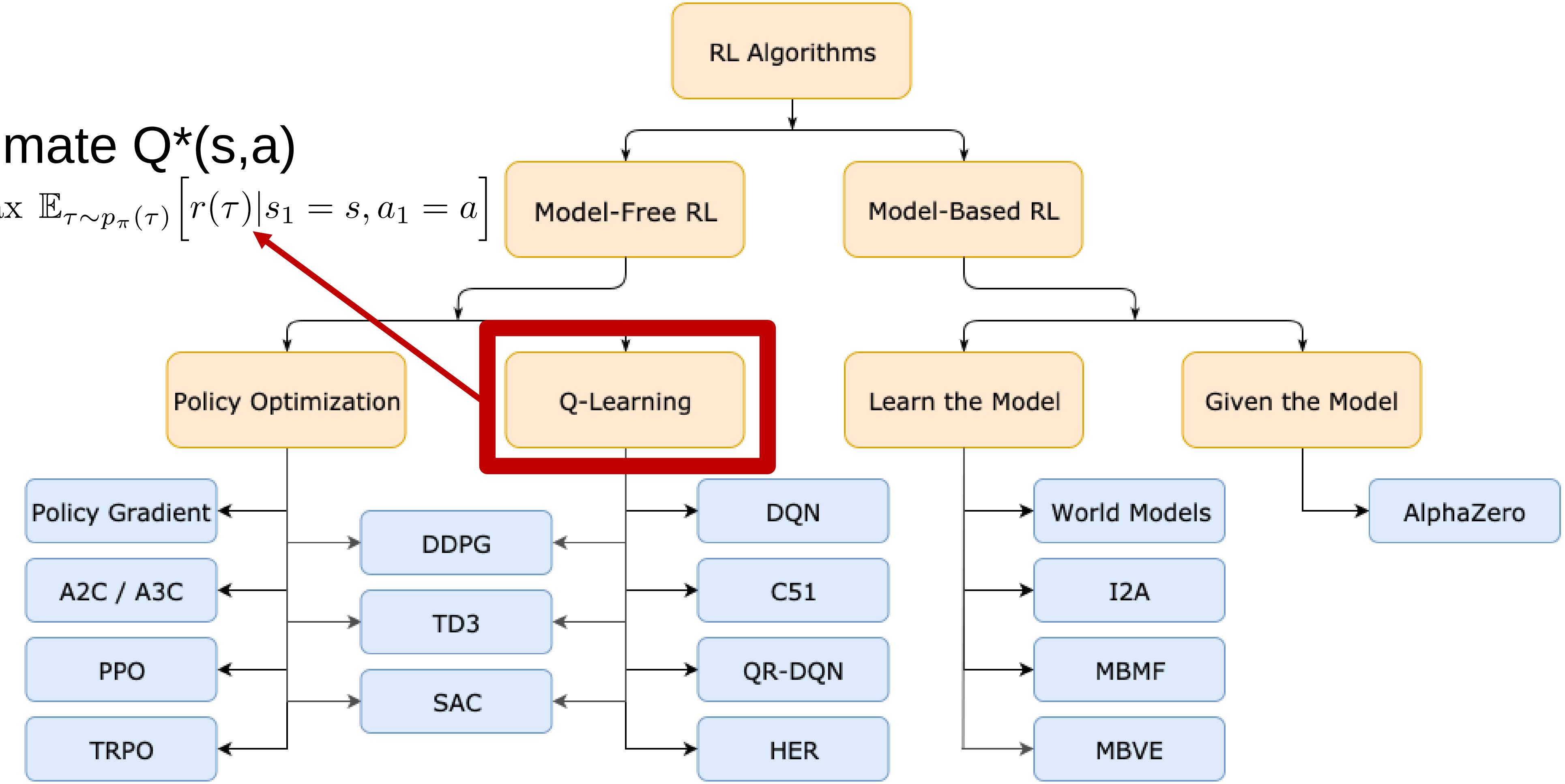
$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau \sim p_{\pi_\theta}(\tau)} \Big[r(\tau) \Big]$$



A Taxonomy of RL algorithms

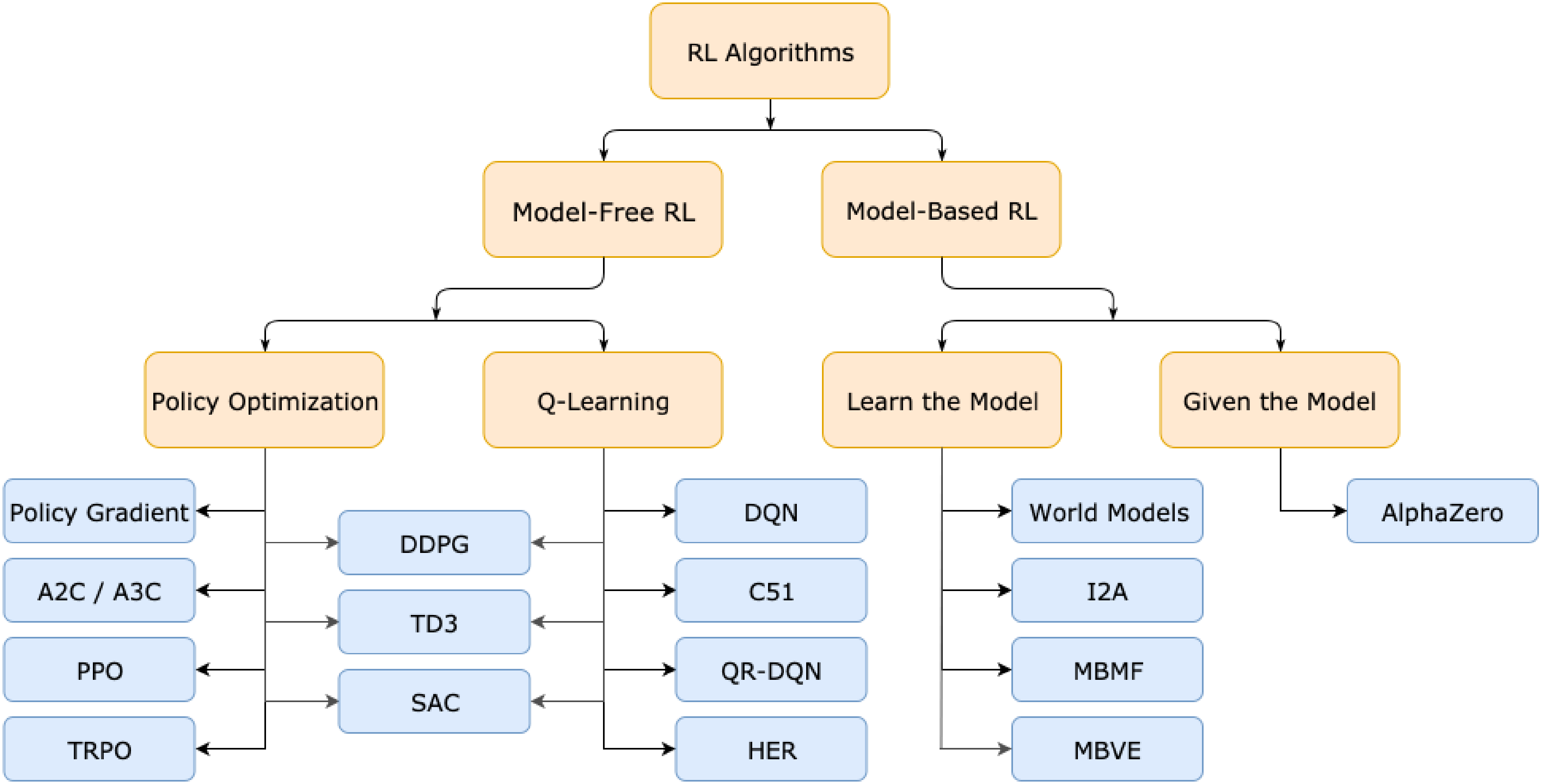
Estimate Q*(s,a)

$$Q^*(s,a) = \max_{\pi} \mathbb{E}_{\tau \sim p_\pi(\tau)} \Big[r(\tau) | s_1 = s, a_1 = a \Big]$$



CRL

A Taxonomy of RL algorithms



Policy gradients

Evaluating the objective

$r(\tau)$ **: Return or Cumulative reward**

$$\theta^* = \arg\max_\theta \underbrace{\mathbb{E}_{\tau \sim p_{\pi_\theta}(\tau)} \Big[\overbrace{\sum_{t=1}^{T} r(s_t, a_t)}^{r(\tau)} \Big]}_{E(\theta)}$$

$$\theta_{k+1} \leftarrow \theta_k + \alpha \nabla_\theta E(\theta)$$

Let’s compute the gradient!

Direct policy differentiation

$$E(\theta) = \mathbb{E}_{\tau \sim p_{\pi_\theta}(\tau)}\Big[r(\tau)\Big] = \int p_{\pi_\theta}(\tau) r(\tau) d\tau$$

$$\nabla_\theta E(\theta) = \int \nabla_\theta p_{\pi_\theta}(\tau) r(\tau) d\tau$$

$$\nabla_\theta E(\theta) = \mathbb{E}_{\tau \sim p_{\pi_\theta}(\tau)}\Big[\nabla_\theta \log p_{\pi_\theta}(\tau) r(\tau)\Big]$$

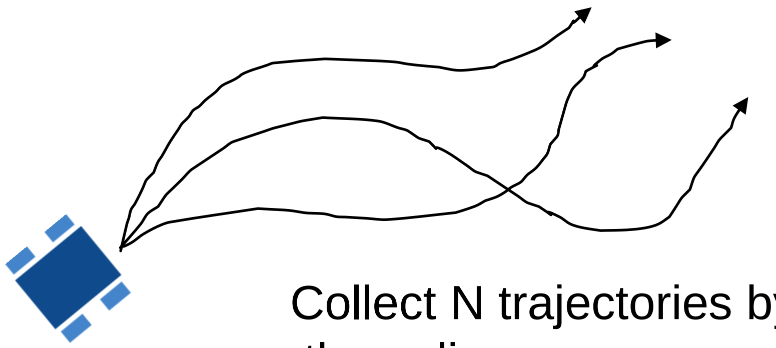
$$\nabla_\theta E(\theta) = \mathbb{E}_{\tau \sim p_{\pi_\theta}(\tau)}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\right) r(\tau)\right]$$

The policy gradient

$$\nabla_\theta E(\theta) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)\right)\left(\sum_{t=1}^{T} r(s_t,a_t)\right)\right]$$

$$\nabla_\theta E(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\right)\left(\sum_{t=1}^{T} r(s_{i,t},a_{i,t})\right)$$

$$\theta_{k+1} \leftarrow \theta_k + \alpha\nabla_\theta E(\theta)$$



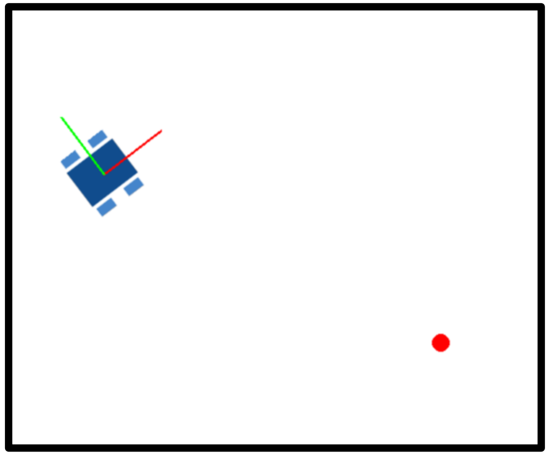
Collect N trajectories by running the policy

REINFORCE: A policy gradient algorithm

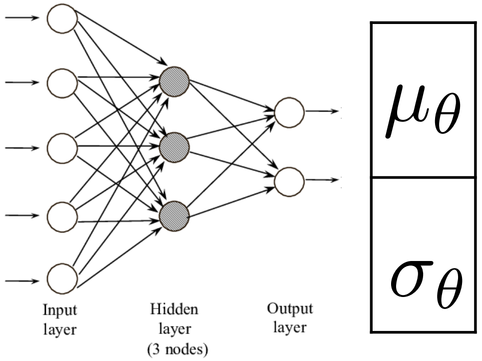
- Ronald J. Williams 1992.
- 3 steps:

1. Generate samples by running the current policy π_{θ_k} on the environment
2. Evaluate the gradient $\nabla_\theta E(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\right)\left(\sum_{t=1}^{T} r(s_{i,t}, a_{i,t})\right)$
3. Do a gradient ascent step $\theta_{k+1} \leftarrow \theta_k + \alpha \nabla_\theta E(\theta)$

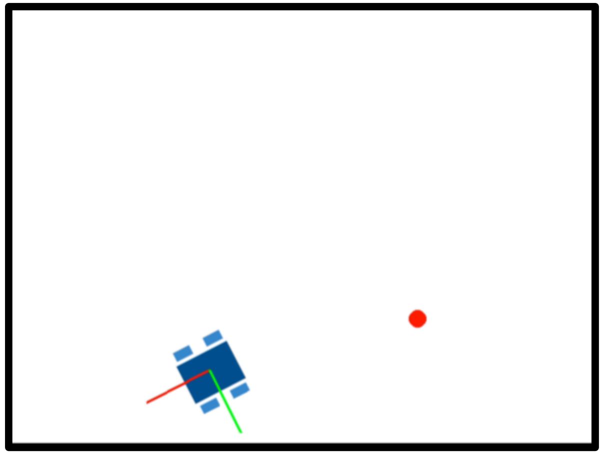
Intuition behind PG



$$s_t : [x, y, v_x, v_y, \alpha]$$

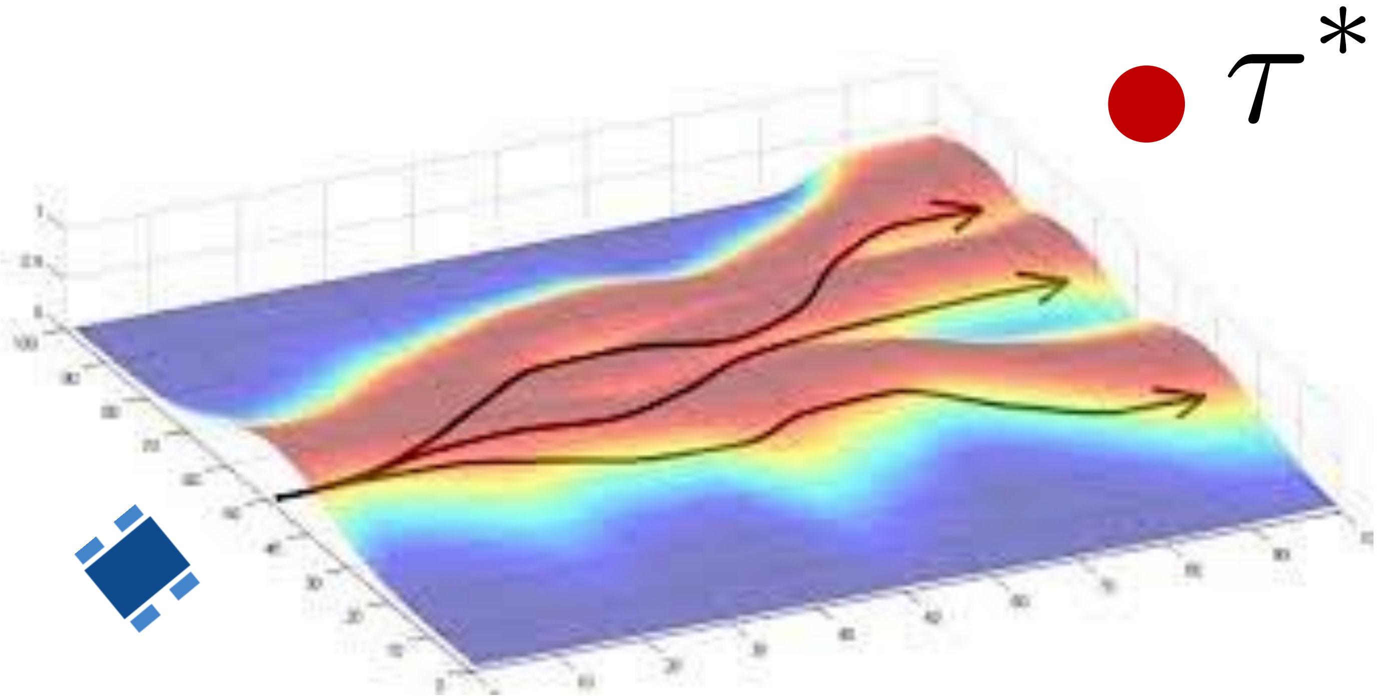


$$\pi_\theta(a_t | s_t)$$



$$a_t : [acc, \dot{\alpha}]$$

Intuition behind PG

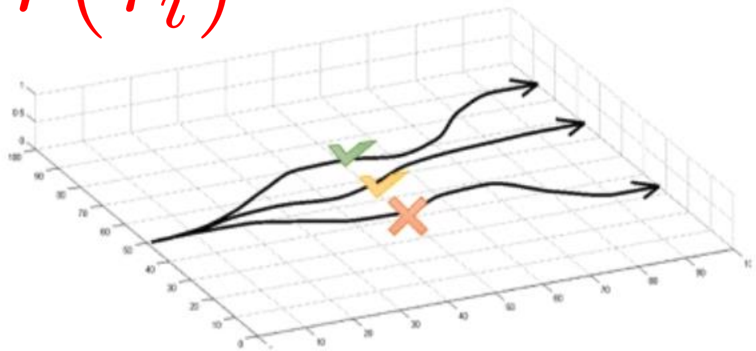


$$\theta_{k+1} \leftarrow \theta_k + \alpha \nabla_\theta \log p_{\pi_\theta}(\tau^*)$$

Update θ in the direction so as to *increase* the value of $\pi_\theta(a_t^*|s_t)$ the fastest

Intuition behind PG

$$\theta_{k+1} \leftarrow \theta_k + \alpha \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log p_{\pi_\theta}(\tau_i) r(\tau_i)$$



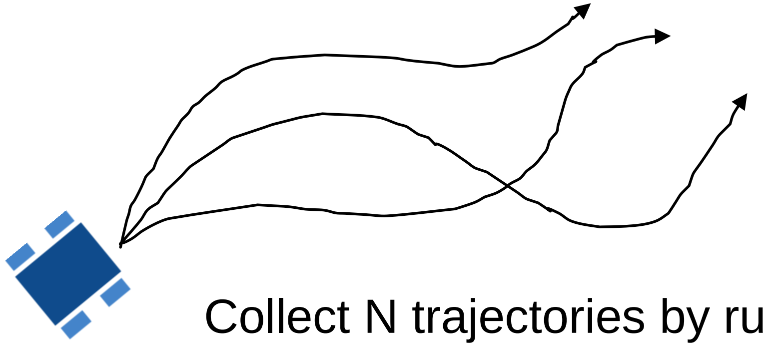
Increase probability of trajectories with positive returns

Decrease probability of trajectories with negative returns

Improving Policy gradients: Some tricks

Reducing variance of the PG estimator

$$\nabla_\theta E(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\right)\left(\sum_{t=1}^{T} r(s_{i,t},a_{i,t})\right)$$



Collect N trajectories by running the policy

1. Enforcing causality

$$\nabla_\theta E(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left(\sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right)$$

We are not accounting for the temporal structure of the problem.

Future actions ($a_{t'}$) cannot affect past rewards (r_t when $t < t'$).

1. Enforcing causality

$$\nabla_\theta E(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) \underbrace{\left(\sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'}) \right)}$$

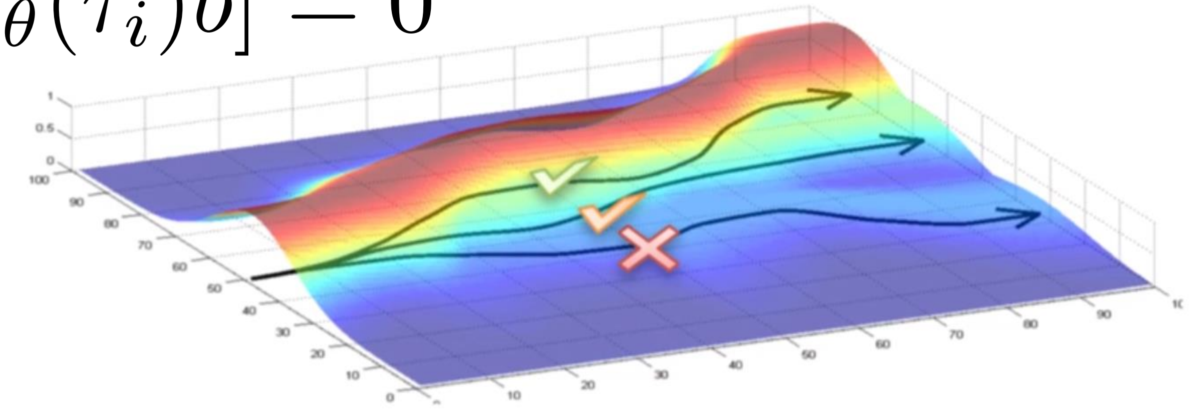
Reward-to-go

$$\hat{Q}^{\pi_\theta}(s_t, a_t)$$

2. Introducing baselines

$$\nabla_\theta E(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau_i)[r(\tau_i) - b]$$

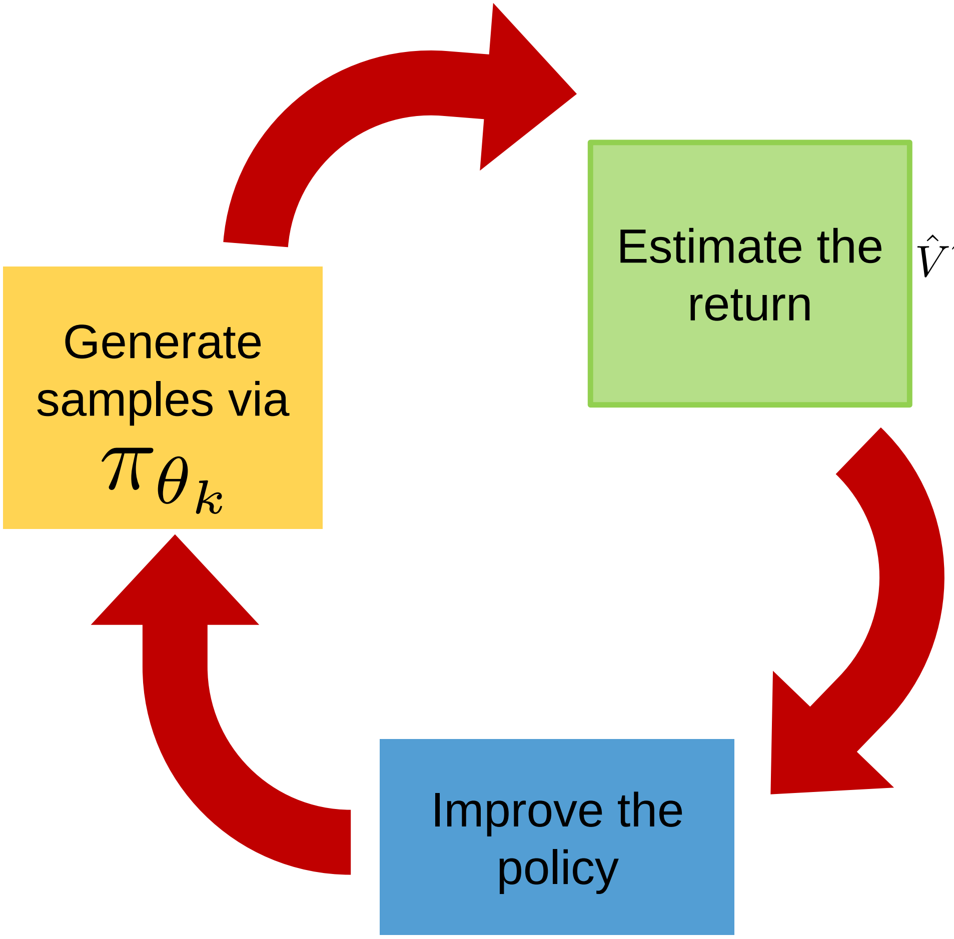
$$b = \frac{1}{N}\sum_{i=1}^{N} r(\tau_i) \qquad \mathbb{E}_{\tau \sim p_{\pi_\theta}}[\nabla_\theta \log \pi_\theta(\tau_i) b] = 0$$



Subtracting a baseline gives us an *unbiased* estimate

Reduces the variance of the gradient estimator

Policy gradient loop



$$\hat{V}^{\pi_\theta}(s) = \mathbb{E}[r(\tau)|s_0 = s] \approx \frac{1}{N}\sum_i^N \sum_t^T r(s_t, a_t)$$

$$\theta_{k+1} \leftarrow \theta_k + \alpha \nabla_\theta E(\theta)$$

CRL

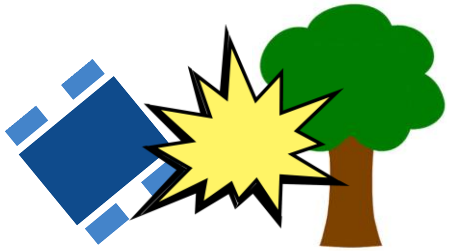
Limitations of vanilla policy gradient

Policy gradient is an **on-policy** algorithm.

$$\nabla_\theta E(\theta) = \mathbb{E}_{\underline{\tau \sim \pi_\theta}(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau) r(\tau)\right]$$

1. Extremely inefficient in terms of number of samples

2. Very risky for real-world problems



Policy gradient loop

$$\cancel{\hat{V}^{\pi_\theta}(s) = \mathbb{E}[r(\tau)|s_0 = s] \approx \frac{1}{N}\sum_i^N \sum_t^T r(s_t, a_t)}$$

Fit a model to estimate the return

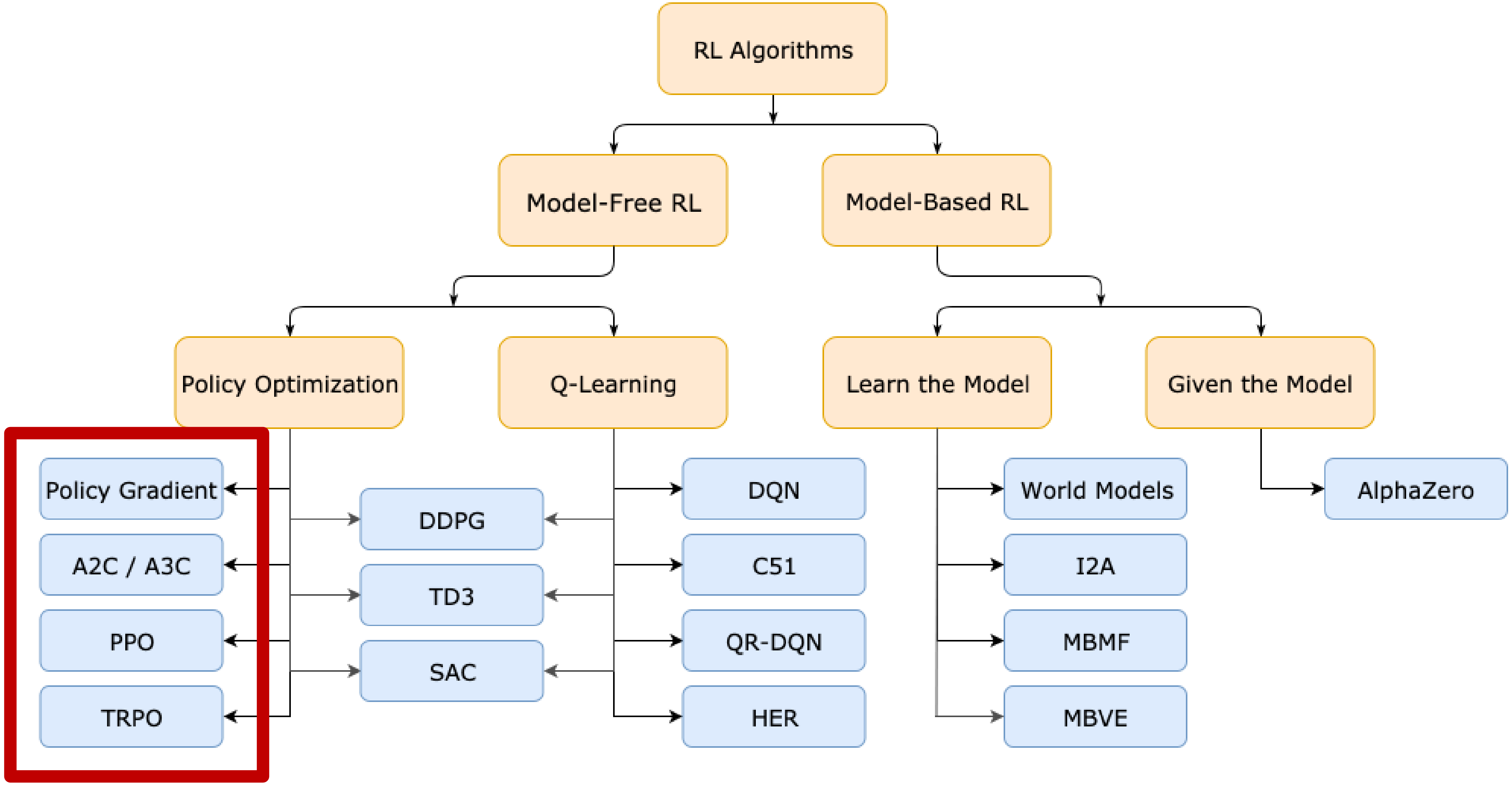
$$\hat{V}^{\pi}_{\phi}(s_t)$$

Generate samples via π_{θ_k}

Improve the policy

$$\theta_{k+1} \leftarrow \theta_k + \alpha \nabla_\theta E(\theta)$$

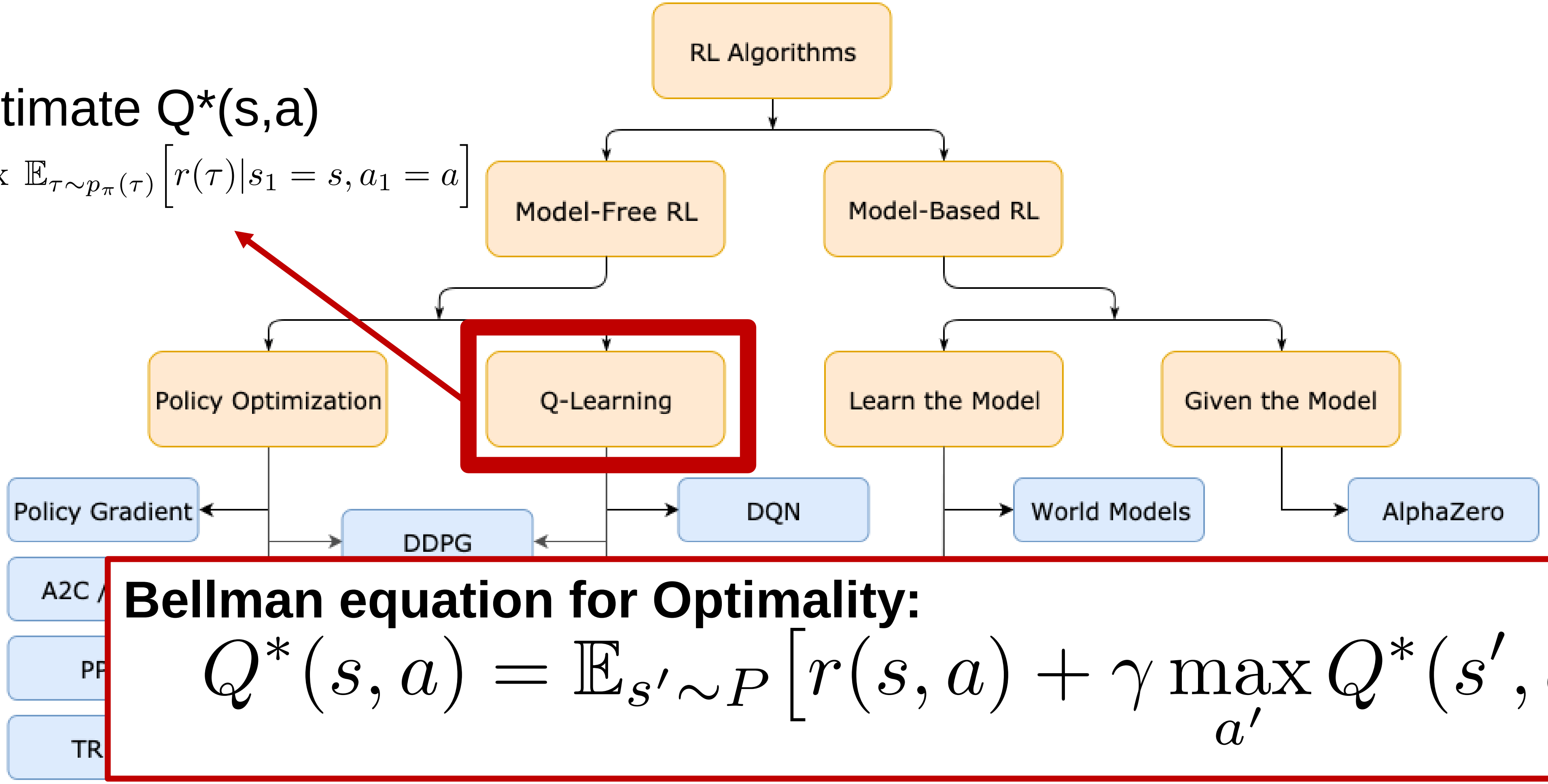
A Taxonomy of RL algorithms



A Taxonomy of RL algorithms

Estimate Q*(s,a)

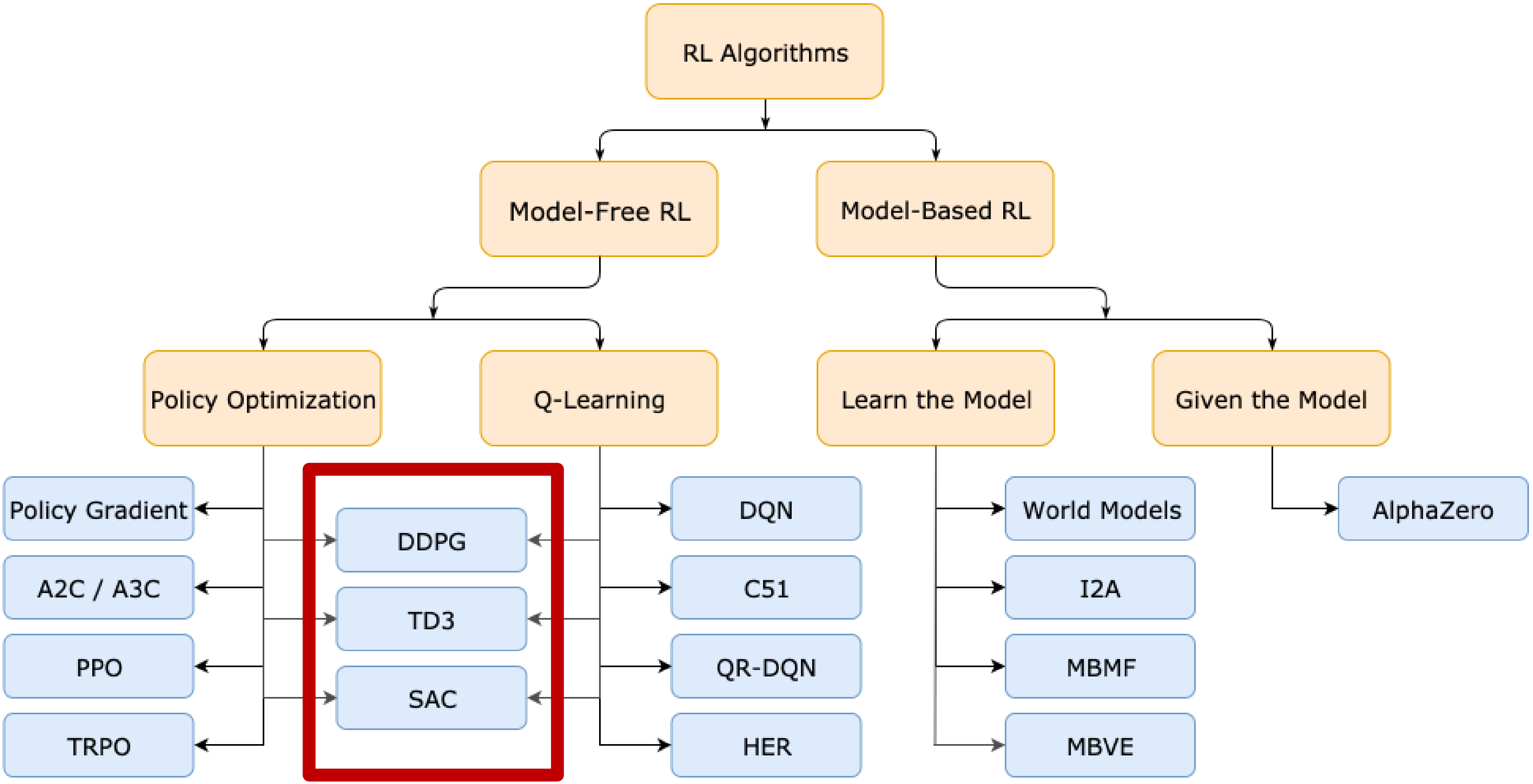
$$Q^*(s,a) = \max_{\pi} \mathbb{E}_{\tau \sim p_\pi(\tau)} \Big[r(\tau) | s_1 = s, a_1 = a \Big]$$



Bellman equation for Optimality:

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

A Taxonomy of RL algorithms



Deep learning. *Autumn Semester.*

Probabilistic Artificial Intelligence. *Autumn Semester.*

Dynamic Programming and Optimal Control. *Autumn Semester*

Recommended Courses

Lectures for UC Berkeley CS 182: Deep Learning.

Spinning up in Deep RL. *Open AI.*

Sources

Appendix

Computing policy gradients

Direct policy differentiation

$$E(\theta) = \mathbb{E}_{\tau \sim p_{\pi_\theta}(\tau)}\Big[r(\tau)\Big] \int p_{\pi_\theta}(\tau) r(\tau) d\tau$$

$$\nabla_\theta E(\theta) = \int \nabla_\theta p_{\pi_\theta}(\tau) r(\tau) d\tau =$$

Convenient identity (the log-derivative trick):

$$\nabla_\theta p_{\pi_\theta}(\tau) = p_{\pi_\theta}(\tau) \frac{\nabla_\theta p_{\pi_\theta}(\tau)}{p_{\pi_\theta}(\tau)} = p_{\pi_\theta}(\tau) \nabla_\theta \log p_{\pi_\theta}(\tau)$$

$$= \int p_{\pi_\theta}(\tau) \nabla_\theta \log p_{\pi_\theta}(\tau) r(\tau) d\tau =$$

We can express the integral as an expected value under the trajectory distribution $p_{\pi_\theta}(\tau)$ now ☺

$$\nabla_\theta E(\theta) = \mathbb{E}_{\tau \sim p_{\pi_\theta}(\tau)}\Big[\nabla_\theta \log p_{\pi_\theta}(\tau) r(\tau)\Big]$$

How to compute this term?

Direct policy differentiation

$$p_{\pi_\theta}(\tau) = d_1(s_1)\prod_{t=1}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t)$$

$$\log p_{\pi_\theta}(\tau) = \log d_1(s_1)\prod_{t=1}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t) =$$

$$= \log d_1(s_1) + \sum_{t=1}^{T}\log\pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t,a_t)$$

No dependency on θ

$$\nabla_\theta E(\theta) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\nabla_\theta \underbrace{\left[\cancel{\log p(s_1)} + \sum_{t=1}^{T}\log\pi_\theta(a_t|s_t) + \cancel{\log p(s_{t+1}|s_t,a_t)}\right]}_{\log p_{\pi_\theta}(\tau)} r(\tau)\right]$$

CRL

The policy gradient

$$\nabla_\theta E(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)\left(\sum_{t=1}^{T} r(s_t, a_t)\right)\right]$$

$$\nabla_\theta E(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\right)\left(\sum_{t=1}^{T} r(s_{i,t}, a_{i,t})\right)$$

$$\theta_{k+1} \leftarrow \theta_k + \alpha \nabla_\theta E(\theta)$$

CRL