DQN and REINFORCE (Finite Action Space)

Hao Su

(slides prepared by Shuang Liu)

Spring, 2021

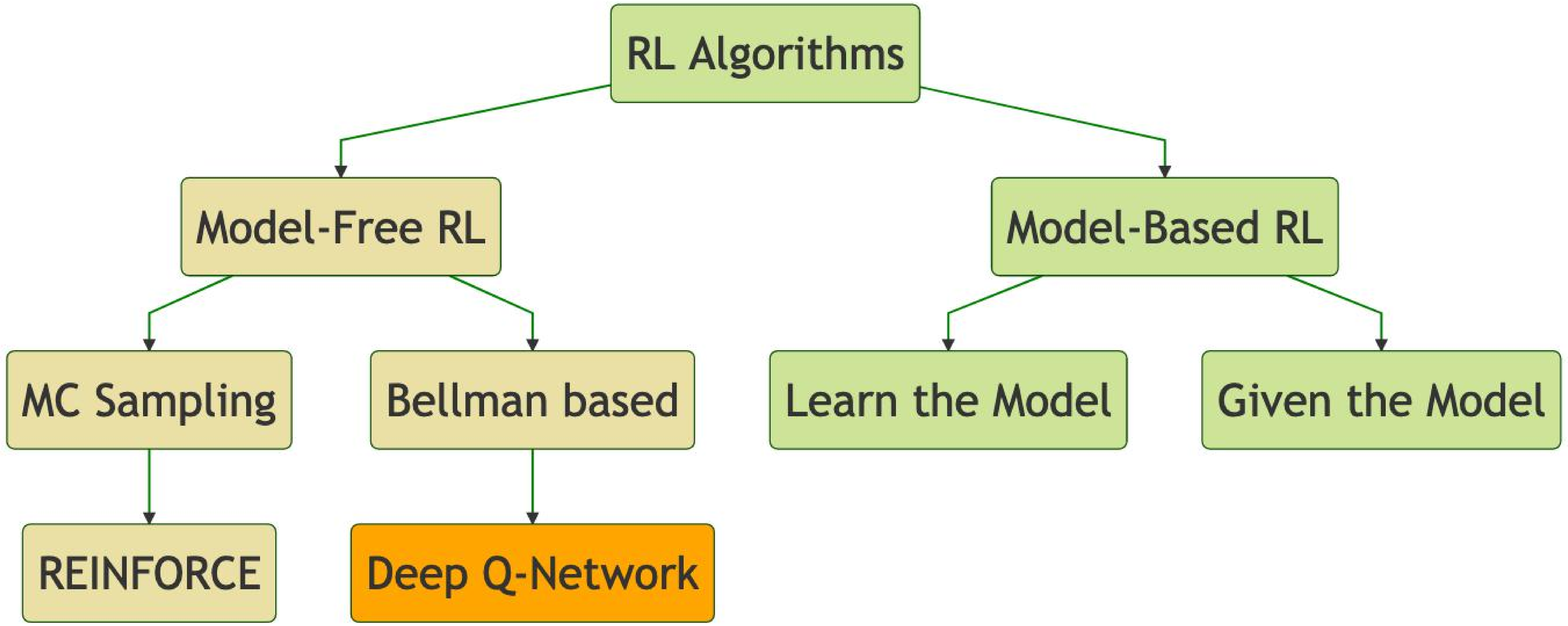
Agenda

- Deep Q-Learning
- Unbiased Policy Gradient Estimation (REINFORCE)
- Practical First-Order Policy Optimization
- Case Study: Proximal Policy Optimization (PPO)

click to jump to the section.

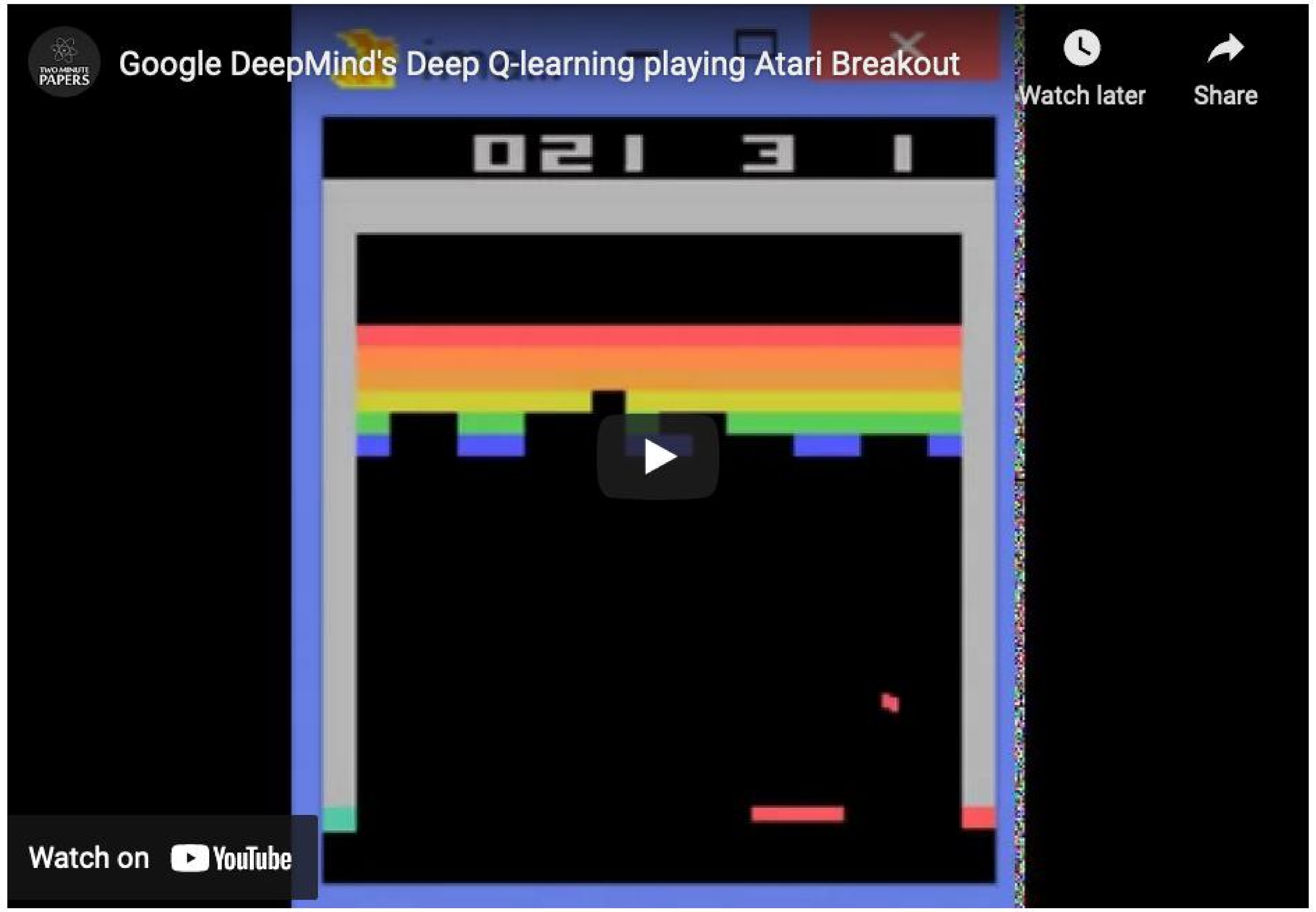
Deep Q-Learning

Taxonomy of RL Algorithms and Examples



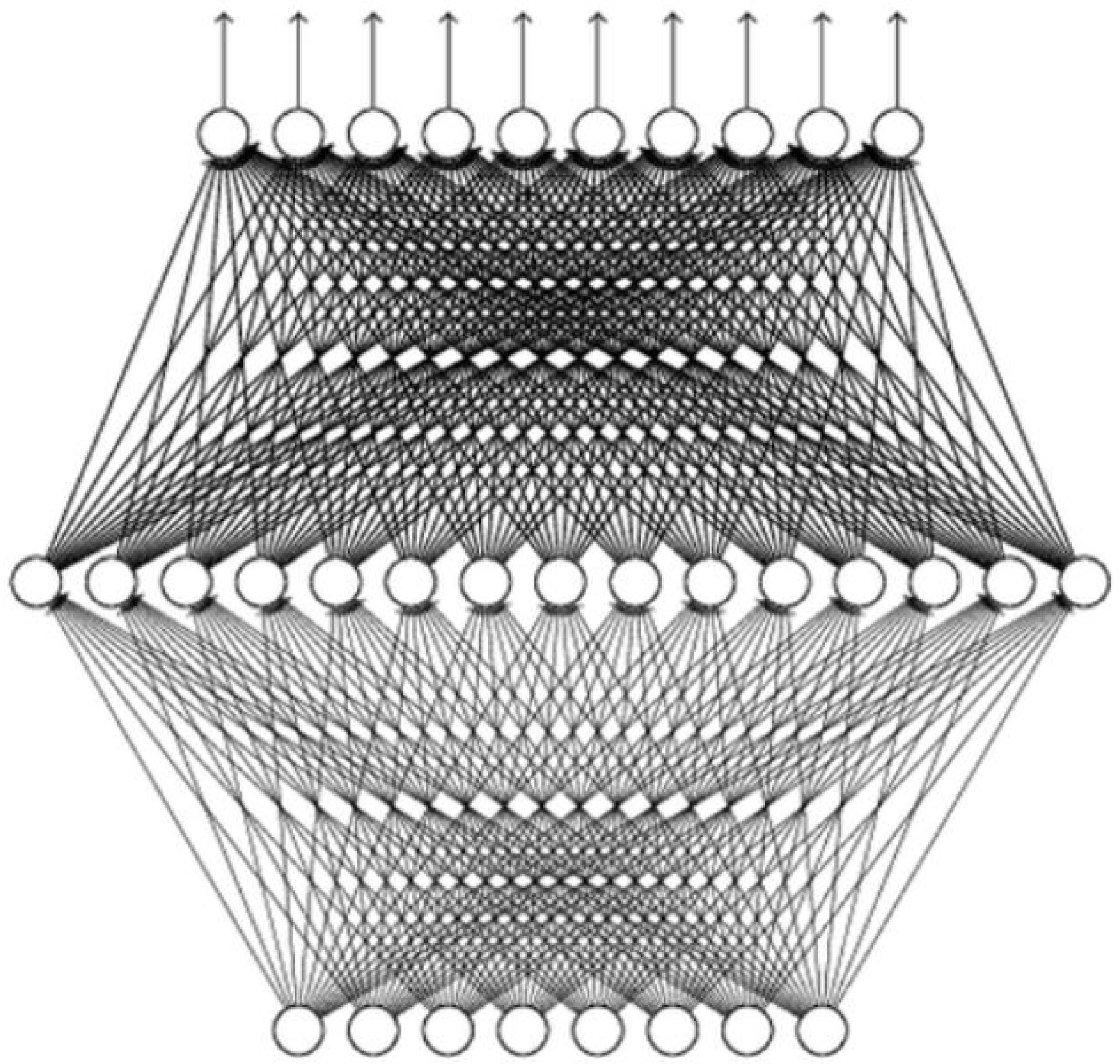
Challenge of Representing Q

- How do we represent $Q(s, a)$?
- Maze has a discrete and small *state space* that we can deal with by an array.
- However, for many cases the state space is continuous, or discrete but huge, array does not work.



Deep Value Network

- Use a neural network to parameterize Q:
  - Input: state $s \in \mathbb{R}^n$
  - Output: each dimension for the value of an action $Q(s, a; \theta)$



Training Deep Q Network

- Recall the Bellman optimality equation for action-value function:

$$Q^*(s,a) = \mathbb{E}[R_{t+1} + \gamma \max_{a'} Q^*(S_{t+1}, a')|S_t = s, A_t = a]$$

- It is natural to build an *optimization problem*:

$$L(\theta) = \mathbb{E}_{(s,a,s')\sim Env}[TD_\theta(s,a,s')] \tag{TD loss}$$

  where $TD_\theta(s,a,s') = \|Q_\theta(s,a) - [R(s,a,s') + \gamma \max_{a'} Q_\theta(s',a')]\|^2$.
- Note: How to obtain the Env distribution has many options!
  - It does not necessarily sample from the optimal policy.
  - A suboptimal, or even bad policy (e.g., random policy), may allow us to learn a good Q.
  - It is a cutting-edge research topic of studying how well we can do for non-optimal Env distribution.

Replay Buffer

- As in the previous Q-learning, we consider a routine that we take turns to
  - Sample certain transitions using the current Q_θ
  - Update Q_θ by minimizing the TD loss
- **Exploration:**
  - We use ϵ-greedy strategy to sample transitions, and add (s, a, s', r) in a **replay buffer** (e.g., maintained by FIFO).
- **Exploitation:**
  - We sample a batch of transitions and train the network by gradient descent:

$$\nabla_\theta L(\theta) = \mathbb{E}_{(s,a,s') \sim \text{ReplayBuffer}}[\nabla_\theta TD_\theta(s, a, s')]$$

Deep Q-Learning Algorithm

- Initialize the replay buffer D and Q network Q_θ.
- For every episode:
  - Sample the initial state $s_0 \sim P(s_0)$
  - Repeat until the episode is over
    - Let s be the current state
    - With prob. ϵ sample a random action a. Otherwise select $a = \arg\max_a Q_\theta(s, a)$
    - Execute a in the environment, and receive the reward r and the next state s'
    - Add transitions (s, a, s') in D
    - Sample a random batch from D and build the batch TD loss
    - Perform one or a few gradient descent steps on the TD loss

Some Engineering Concerns about Deep Q-Learning

- States and value network architecture
  - First of all, a good computer vision problem worth research.
  - Need to ensure that states are sufficient statistics for decision-making
    - Common practice: Stack a fixed number of frames (e.g., 4 frames) and pass through ConvNet
    - If long-term history is important, may use LSTM/GRU/Transformer/... to aggregate past history
  - May add other computing structures that are effective for video analysis, e.g., optical flow map
  - Not all vision layers can be applied without verification (e.g., batch normalization layer may be harmful)
- Replay buffer
  - Replay buffer size matters.
  - When sampling from the replay buffer, relatively large batch size helps stabilizing training.

Some Theoretical Concerns about Q-Learning

- Behavior/Target Network: Recall that $TD_\theta(s,a,s') = \|Q_\theta(s,a) - [R(s,a,s') + \gamma \max_{a'} Q_\theta(s',a')]\|^2$. We keep two Q networks in practice. We only update the blue network by gradient descent and use it to sample new trajectories. Every few episodes we replace the red one by the blue one. The reason is that the blue one changes too fast. The red one is called *target network* (to build target), and the blue one is called *behavior network* (to sample actions).
- Value overestimation: Note that the TD loss takes the maximal a for each $Q(s,\cdot)$. Since TD loss is not unbiased, the max operator will cause the Q-value to be overestimated! There are methods to mitigate (e.g., double Q-learning) or work around (e.g., advantage function) the issue.
- Uncertainty of Q estimation: Obviously, the Q value at some (s,a) are estimated from more samples, and should be more trustable. Those high Q value with low confidence are quite detrimental to performance. Distributional Q-Learning quantifies the confidence of Q and leverages the confidence to recalibrate target values and conduct exploration.
- Theoretically, Q-learning (more precisely, a variation of it) is an **optimal online learning algorithm** for tabular RL.

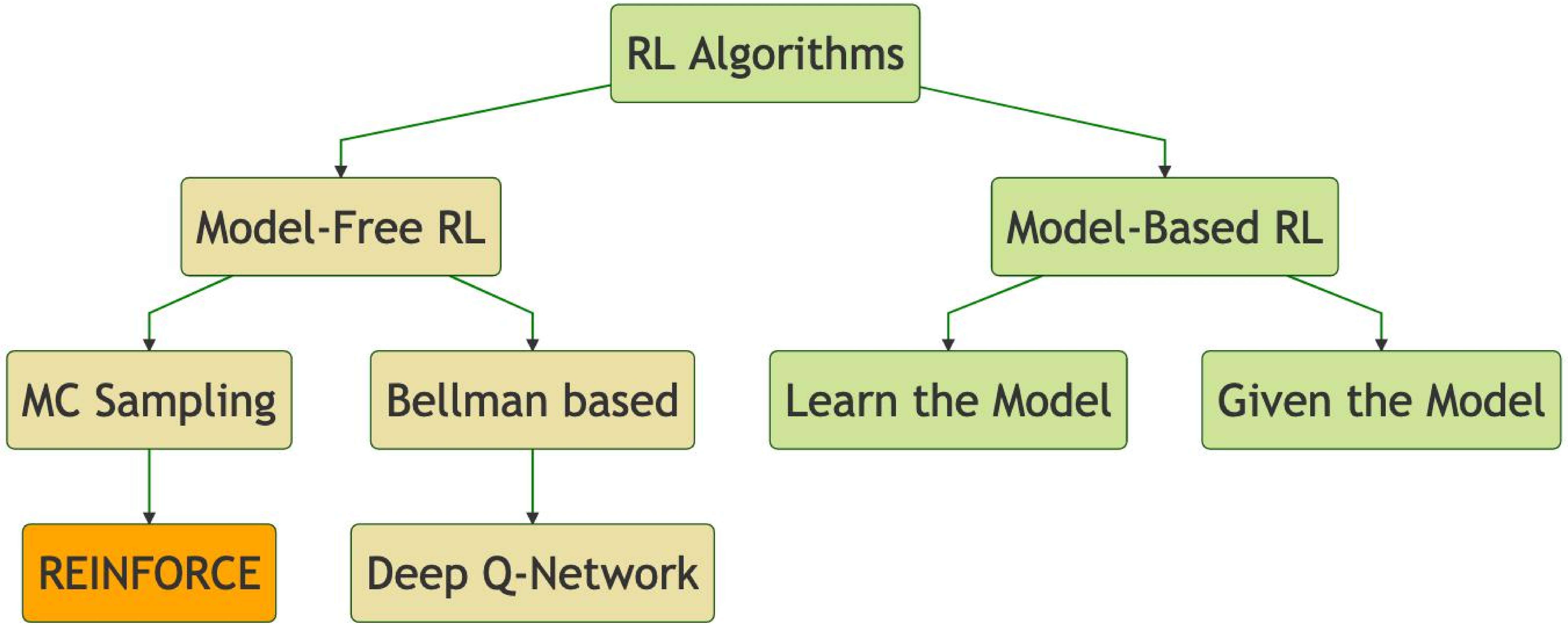
Convergence of Reinforcement Learning Algorithms

We state the facts without proof:

- Q-Learning:
  - Tabular setup: Guaranteed convergence to the optimal solution. A simple proof (using contraction mapping).
  - Value network setup: No convergence guarantee due to the approximation nature of networks.

Unbiased Policy Gradient Estimation (REINFORCE)

Taxonomy of RL Algorithms and Examples



First-Order Policy Optimization

- Recall that a policy π (assume its independent of step t) is just a function that maps from a state to a distribution over the action space.
  - The quality of π is determined by $V^{\pi}(s_0)$, where s_0 is the initial state
    - Q: What if the initial state is a distribution?
  - We can parameterize π by π_θ, e.g.,
    - a neural network
    - a categorical distribution

First-Order Policy Optimization

- Now we can formulate policy optimization as

$$\underset{\theta\in\Theta}{\text{maximize}} \qquad V^{\pi_\theta}(s_0).$$

First-Order Policy Optimization

- Now we can formulate policy optimization as

$$\underset{\theta \in \Theta}{\text{maximize}} \qquad V^{\pi_\theta}(s_0).$$

- Sampling allows us to estimate values.
- If sampling also allows us to estimate the gradient of values w.r.t. policy parameters $\frac{\partial V^{\pi_\theta}(s_0)}{\partial \theta}$, we can improve our policies!

First-Order Policy Optimization

- Now we can formulate policy optimization as

$$\underset{\theta \in \Theta}{\text{maximize}} \qquad V^{\pi_\theta}(s_0).$$

- Sampling allows us to estimate values.
- If sampling also allows us to estimate the gradient of values w.r.t. policy parameters $\frac{\partial V^{\pi_\theta}(s_0)}{\partial \theta}$, we can improve our policies!

Goal: Derivate a way to directly estimate the gradient $\frac{\partial V^{\pi_\theta}(s_0)}{\partial \theta}$ from samples.

Policy Gradient Theorem (Undiscounted)

- By Bellman expectation equation,

$$\begin{aligned} V^{\pi_\theta}(s) &= \mathbb{E}_{\pi_\theta}[R_{t+1} + \gamma V_\pi(S_{t+1}) | S_t = s] \\ &= \sum_a \pi_\theta(s,a) \cdot \mathbb{E}_{s' \sim T(s,a)} \left[r(s,a) + V^{\pi_\theta}(s') \right]. \end{aligned}$$

- How to calculate $\nabla_\theta V^{\pi_\theta}(s_0)$? Note that,

$$\begin{aligned} \nabla_\theta V^{\pi_\theta}(s_0) &= \sum_{a_0} \nabla_\theta \left\{ \pi_\theta(s_0,a_0) \cdot \mathbb{E}_{s_1 \sim T(s_0,a_0)} \left[r(s_0,a_0,s_1) + V^{\pi_\theta}(s_1) \right] \right\} \\ \text{(product rule)} &= \sum_{a_0} \left\{ \nabla_\theta \pi_\theta(s_0,a_0) \cdot Q^{\pi_\theta}(s_0,a_0) + \pi_\theta(s_0,a_0) \cdot \mathbb{E}_{s_1 \sim T(s_0,a_0)} \left[\nabla_\theta V^{\pi_\theta}(s_1) \right] \right\} \\ &= \sum_{a_0} \left\{ \nabla_\theta \pi_\theta(s_0,a_0) \cdot Q^{\pi_\theta}(s_0,a_0) + \pi_\theta(s_0,a_0) \cdot \mathbb{E}_{s_1 \sim T(s_0,a_0)} \left[\sum_{a_1} \{ \nabla_\theta \pi(s_1,a_1) Q^{\pi_\theta}(s_1,a_1) + \pi_\theta(s_1,a_1) \mathbb{E}[\nabla_\theta V^{\pi_\theta}(s_2)] \} \right] \right\} \\ &= \left\{ \sum_{a_0} \nabla_\theta \pi_\theta(s_0,a_0) \cdot Q^{\pi_\theta}(s_0,a_0) \right\} + \left\{ \sum_{a_0} \pi_\theta(s_0,a_0) \cdot \mathbb{E}_{s_1 \sim T(s_0,a_0)} \left[\sum_{a_1} \nabla_\theta \pi(s_1,a_1) Q^{\pi_\theta}(s_1,a_1) \right] \right\} + \dots \\ \text{(recursively repeat above)} &= \sum_{t=0}^{\infty} \sum_s \mu_t(s; s_0) \sum_a \nabla_\theta \pi_\theta(s,a) \cdot Q^{\pi_\theta}(s,a) = \sum_s \sum_t^{\infty} \mu_t(s; s_0) \sum_a \nabla_\theta \pi_\theta(s,a) \cdot Q^{\pi_\theta}(s,a) \end{aligned}$$

Q: What does $\mu_t(s; s_0)$ mean?

Policy Gradient Theorem (Undiscounted)

- Policy Gradient Theorem (Undiscounted):

$$\nabla_\theta V^{\pi_\theta}(s_0) = \sum_s \sum_t^{\infty} \mu_t(s; s_0) \sum_a \nabla_\theta \pi_\theta(s, a) \cdot Q^{\pi_\theta}(s, a)$$

where $\mu_t(s; s_0)$ is the average visitation frequency of the state s in step k, and $\sum_s \sum_t \mu_t(s; s_0) = 1$.

Policy Gradient Theorem (Undiscounted)

- Policy Gradient Theorem (Undiscounted):

$$\nabla_\theta V^{\pi_\theta}(s_0) = \sum_s \sum_t^\infty \mu_t(s; s_0) \sum_a \nabla_\theta \pi_\theta(s,a) \cdot Q^{\pi_\theta}(s,a)$$

  where $\mu_t(s; s_0)$ is the average visitation frequency of the state s in step k, and $\sum_s \sum_t \mu_t(s; s_0) = 1$.
- Let $d^{\pi_\theta}(s; s_0) = \sum_t \mu_t(s; s_0)$, and $d^{\pi_\theta}(s; s_0)$ is the *stationary distribution* of state visitation under π. The theorem is also sometimes stated as

$$\nabla_\theta V^{\pi_\theta}(s_0) = \sum_s d^{\pi_\theta}(s; s_0) \sum_a \nabla_\theta \pi_\theta(s,a) \cdot Q^{\pi_\theta}(s,a)$$

Policy Gradient Theorem (Undiscounted)

$$\nabla_\theta V^{\pi_\theta}(s_0) = \sum_s d^{\pi_\theta}(s; s_0) \sum_a \nabla_\theta \pi_\theta(s,a) \cdot Q^{\pi_\theta}(s,a)$$

- Q: What is the intuitive interpretation from this equation?

Policy Gradient Theorem (Undiscounted)

$$\nabla_\theta V^{\pi_\theta}(s_0) = \sum_s d^{\pi_\theta}(s; s_0) \sum_a \nabla_\theta \pi_\theta(s, a) \cdot Q^{\pi_\theta}(s, a)$$

- Q: What is the intuitive interpretation from this equation?
  - Weighted sum of (log) policy gradients for all steps.
  - Higher weights for states with higher frequency.
  - Earlier steps has higher Q, thus higher weights.

Policy Gradient Theorem (Discounted)

- Policy Gradient Theorem (Undiscounted):

$$\nabla_\theta V^{\pi_{\theta,\gamma}}(s_0) = \sum_s \sum_{t=0}^{\infty} \gamma^t \mu_t(s; s_0) \sum_a \nabla_\theta \pi_\theta(s,a) \cdot Q^{\pi_\theta,\gamma}(s,a).$$

  $\mu_t(s; s_0)$ is the average visitation frequency of the state s in step k.
- Can you guess the influence of γ in this result?

We will assume the discounted setting from now on.

Creating an Unbiased Estimate for PG

$$\nabla_\theta V^{\pi_\theta,\gamma}(s_0) = \sum_{t=0}^{\infty} \sum_s \gamma^t \mu_t(s; s_0) \sum_a \nabla_\theta \pi_\theta(s,a) \cdot Q^{\pi_\theta,\gamma}(s,a)$$

Let's say we have used π_θ to collect a rollout trajectory $\{(s_t, a_t, r_t)\}_{t=0}^{\infty}$, where s_t, a_t, r_t are random variables.
Note that $\nabla_\theta \ln \pi_\theta = \frac{\nabla \pi_\theta}{\pi_\theta} \Rightarrow \nabla \pi_\theta = \nabla_\theta \ln(\pi_\theta) \cdot \pi_\theta$

$$\begin{aligned}
\nabla_\theta V^{\pi_\theta,\gamma}(s_0) &= \sum_s \sum_{t=0}^{\infty} \gamma^t \mu_t(s; s_0) \sum_a \nabla_\theta \ln(\pi_\theta(s,a)) \cdot \pi_\theta(s,a) Q^{\pi_\theta,\gamma}(s,a) \\
\text{(absorb randomness of env. in } \mu_t) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \sum_a \nabla_\theta \ln(\pi_\theta(s_t,a)) \cdot \pi_\theta(s_t,a) Q^{\pi_\theta,\gamma}(s_t,a)\right] \\
\text{(absorb randomness of action in } \pi_\theta) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \ln(\pi_\theta(s_t,a_t)) \cdot Q^{\pi_\theta,\gamma}(s_t,a_t)\right] \\
&= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \ln(\pi_\theta(s_t,a_t)) \cdot \sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i\right]
\end{aligned}$$

Creating an Unbiased Estimate for PG (Cont'd)

We have shown that

$$\nabla_\theta V^{\pi_\theta,\gamma}(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \ln(\pi_\theta(s_t, a_t)) \cdot \sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i\right] = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \ln(\pi_\theta(s_t, a_t)) \cdot G_t\right]$$

Q: Does this result look reasonable?

Creating an Unbiased Estimate for PG (Cont'd)

We have shown that

$$\nabla_\theta V^{\pi_\theta,\gamma}(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \ln(\pi_\theta(s_t, a_t)) \cdot \sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i\right] = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \ln(\pi_\theta(s_t, a_t)) \cdot G_t\right]$$

Q: Does this result look reasonable?

- Using more trajectories, we can get more accurate gradient estimate (smaller variance)
- Since the unbiased estimate is a summation, we can sample from the individual terms to do batched gradient descent

We have established an MC sampling based method to estimate the gradient of value w.r.t. policy parameters! This estimate is *unbiased*.

- In literature, this MC-sampling based policy gradient method is called **REINFORCE**.

Practical First-Order Policy Optimization

Advanced Value Estimates

We have seen that we can use $\sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i$ as an unbiased estimate for $Q^{\pi_\theta,\gamma}(s_t, a_t)$.

We can also have a value network $v_\omega(s)$ to try to **memorize** (the estimates of) $V^{\pi_\theta,\gamma}(s)$ during the training. This way, whenever we need an estimate of $Q^{\pi_\theta,h}(s_t, a_t)$, we can use

Advanced Value Estimates

We have seen that we can use $\sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i$ as an unbiased estimate for $Q^{\pi_\theta,\gamma}(s_t, a_t)$.

We can also have a value network $v_\omega(s)$ to try to **memorize** (the estimates of) $V^{\pi_\theta,\gamma}(s)$ during the training. This way, whenever we need an estimate of $Q^{\pi_\theta,h}(s_t, a_t)$, we can use

- $e_\infty = \sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i$, which is unbiased but has high variance.

Advanced Value Estimates

We have seen that we can use $\sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i$ as an unbiased estimate for $Q^{\pi_\theta,\gamma}(s_t, a_t)$.

We can also have a value network $v_\omega(s)$ to try to **memorize** (the estimates of) $V^{\pi_\theta,\gamma}(s)$ during the training. This way, whenever we need an estimate of $Q^{\pi_\theta,h}(s_t, a_t)$, we can use

- $e_\infty = \sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i$, which is unbiased but has high variance.
- $e_{t,0} = r_t + \gamma \cdot v_\omega(s_{t+1})$, which is biased but possibly has lower variance.

Advanced Value Estimates

We have seen that we can use $\sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i$ as an unbiased estimate for $Q^{\pi_\theta,\gamma}(s_t, a_t)$.

We can also have a value network $v_\omega(s)$ to try to **memorize** (the estimates of) $V^{\pi_\theta,\gamma}(s)$ during the training. This way, whenever we need an estimate of $Q^{\pi_\theta,h}(s_t, a_t)$, we can use

- $e_\infty = \sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i$, which is unbiased but has high variance.
- $e_{t,0} = r_t + \gamma \cdot v_\omega(s_{t+1})$, which is biased but possibly has lower variance.
- $e_{t,h} = \sum_{i=t}^{t+h} \gamma^{i-t} \cdot r_i + \gamma^{h+1} \cdot v_\omega(s_{t+h+1})$, which has a trade-off between the first two, depending on the choice of h.

Advanced Value Estimates

We have seen that we can use $\sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i$ as an unbiased estimate for $Q^{\pi_\theta,\gamma}(s_t, a_t)$.

We can also have a value network $v_\omega(s)$ to try to **memorize** (the estimates of) $V^{\pi_\theta,\gamma}(s)$ during the training. This way, whenever we need an estimate of $Q^{\pi_\theta,h}(s_t, a_t)$, we can use

- $e_\infty = \sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i$, which is unbiased but has high variance.
- $e_{t,0} = r_t + \gamma \cdot v_\omega(s_{t+1})$, which is biased but possibly has lower variance.
- $e_{t,h} = \sum_{i=t}^{t+h} \gamma^{i-t} \cdot r_i + \gamma^{h+1} \cdot v_\omega(s_{t+h+1})$, which has a trade-off between the first two, depending on the choice of h.
- $\sum_{h=0}^{\infty} \alpha_h e_{t,h}$, further combines different e_h's with tunable weights α_h's that summing to 1.

Advantage

- We introduce another statistics trick to reduce the variance of the value estimation.

Advantage

- We introduce another statistics trick to reduce the variance of the value estimation.

- Suppose that X and Y are two random variables.
- Recall that, if $Z = X - Y$, then

$$\mathrm{Var}[Z] = \mathrm{Var}[X] + \mathrm{Var}[Y] - 2\mathrm{Cov}(X, Y)$$

- If X and Y are strongly correlated, then $\mathrm{Var}[Z]$ is smaller than $\mathrm{Var}[X]$ and $\mathrm{Var}[Y]$.
  - For example, $X, Y \sim \mathbb{N}(0, 1)$, then $\mathrm{Var}[Z] = 2 - 2\rho$, where ρ is the Pearson correlation coefficient and $-1 \leq \rho \leq 1$. For highly correlated X and Y, $\rho \approx 1$, and $\mathrm{Var}[Z] \approx 0$.

Advantage

- We introduce another statistics trick to reduce the variance of the value estimation.

- Suppose that X and Y are two random variables.
- Recall that, if $Z = X - Y$, then

$$\mathrm{Var}[Z] = \mathrm{Var}[X] + \mathrm{Var}[Y] - 2\mathrm{Cov}(X, Y)$$

- If X and Y are strongly correlated, then $\mathrm{Var}[Z]$ is smaller than $\mathrm{Var}[X]$ and $\mathrm{Var}[Y]$.
  - For example, $X, Y \sim \mathbb{N}(0, 1)$, then $\mathrm{Var}[Z] = 2 - 2\rho$, where ρ is the Pearson correlation coefficient and $-1 \leq \rho \leq 1$. For highly correlated X and Y, $\rho \approx 1$, and $\mathrm{Var}[Z] \approx 0$.

- For this reason, we introduce the function $A^{\pi_\theta,\gamma}(s, a) = Q^{\pi_\theta,\gamma}(s, a) - V^{\pi_\theta,\gamma}(s)$, which is called **advantage**.
- Our next goal is to relate $\nabla V^{\pi_\theta,\gamma}(s, a)$ with $A^{\pi_\theta,\gamma}(s, a)$, which has smaller variance than estimating through $Q^{\pi_\theta,\gamma}(s, a)$.

Advantage Estimates

$$
\begin{aligned}
\nabla_\theta V^{\pi_\theta,\gamma}(s_0) &= \sum_s \sum_{t=0}^{\infty} \gamma^t \mu_t(s; s_0) \left(\sum_a \nabla_\theta \pi_\theta(s,a) \cdot Q^{\pi_\theta,\gamma}(s,a) - 0 \right) \\
\text{(why?)} &= \sum_s \sum_{t=0}^{\infty} \gamma^t \mu_t(s; s_0) \left(\sum_a \nabla_\theta \pi_\theta(s,a) \cdot Q^{\pi_\theta,\gamma}(s,a) - \sum_a \nabla_\theta \pi_\theta(s,a) \cdot V^{\pi_\theta,\gamma}(s) \right) \\
&= \sum_s \sum_{t=0}^{\infty} \gamma^t \mu_t(s; s_0) \sum_a \nabla_\theta \pi_\theta(s,a) \cdot \left(Q^{\pi_\theta,\gamma}(s,a) - V^{\pi_\theta,\gamma}(s) \right). \\
&= \sum_s \sum_{t=0}^{\infty} \gamma^t \mu_t(s; s_0) \sum_a \nabla_\theta \ln(\pi_\theta(s,a)) \cdot \pi_\theta(s,a) A^{\pi_\theta,\gamma}(s,a) \\
&= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \ln(\pi_\theta(s_t,a_t)) \cdot A^{\pi_\theta,\gamma}(s_t,a_t) \right]
\end{aligned}
$$

- Q: Does this form look reasonable? Compare it with the update for REINFORCE.

Advantage Estimates (Cont'd)

- Recall that we said $Q^{\pi_\theta,\gamma}(s_t, a_t)$ can be estimated by $\sum_{h=0}^{\infty} \alpha_h e_{t,h}$ in general, where

$$\begin{cases} e_{t,h} = \sum_{i=t}^{t+h} \gamma^{i-t} \cdot r_i + \gamma^{h+1} \cdot v_\omega(s_{t+h+1}) \\ \sum_{i=0}^{\infty} \alpha_i = 1 \end{cases}$$

- The very popular **General Advantage Estimate** (GAE) estimates the advantage in the same fashion and it chooses α_i to be proportional to λ^i, where $\lambda \in [0, 1]$.

Advantage Estimates (Cont'd)

- Define $\delta_{t,h} = e_{t,h} - v_\omega(s_t) = \sum_{i=t}^{t+h} \gamma^{i-t} \cdot r_i + \gamma^{h+1} \cdot v_\omega(s_{t+h+1}) - v_\omega(s_t)$
- The General Advantage Estimate (GAE) estimates $A^{\pi_\theta,\gamma}(s_t, a_t)$ by

$$\hat{A}^{\pi_\theta,\gamma}_{\mathrm{GAE}(\lambda)}(s_t, a_t) = (1-\lambda) \sum_{h=0}^{\infty} \lambda^h \delta_{t,h}$$

$$\text{(calculation omitted, HW)} = \sum_{h=0}^{\infty} (\gamma\lambda)^h \delta_{t+h,0}$$

- Define $0^0 = 1$, $\hat{A}^{\pi_\theta,\gamma}_{\mathrm{GAE}(0)}(s_t, a_t) = r_t + \gamma v_\omega(s_{t+1}) - v_\omega(s_t)$.
- We also have $\hat{A}^{\pi_\theta,\gamma}_{\mathrm{GAE}(1)}(s_t, a_t) = \sum_{h=0}^{\infty} \gamma^h r_{t+h} - v_\omega(s_t)$.
- We leave the proofs to homework.
- Q: How to interpret the role of λ?

Incremental Monte Carlo
Value Function Estimation

- Recall that our plan was to have a value network $v_\omega(s)$ to **memorize** the estimates of $V^{\pi_\theta,\gamma}(s)$ during the training.
- In practice, certain variants of the incremental Monte-Carlo method will be used to update $v_\omega(s)$
  - e.g., in Sec 5 of High-Dimensional Continuous Control Using Generalized Advantage Estimation,

$$\begin{aligned} \text{minimize}_\omega \quad & \sum_{n=1}^{N} \|v_\omega(s_n) - \hat{V}_n\|^2 \\ \text{subject to} \quad & \frac{1}{N}\sum_{n=1}^{N} \frac{\|v_\omega(s_n) - v_{\omega_{old}}(s_n)\|^2}{2\sigma^2} \leq \epsilon \end{aligned}$$

  where $\hat{V}_t = \sum_{h=0}^{\infty} \gamma^h r_{t+h}$ is the discounted sum of rewards, and n indexes over all timestamps in a batch of trajectories.
- Note that we do not need a replay buffer to estimate $v_\omega(s)$.

Some Additional Information

Given any advantage estimate $\hat{A}^{\pi_\theta,\gamma}(s_t, a_t)$, we can estimate the policy gradient by

$$\hat{\nabla}_\theta V^{\pi_\theta,\gamma}(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \ln(\pi_\theta(s_t, a_t)) \cdot \hat{A}^{\pi_\theta,\gamma}(s_t, a_t)\right].$$

However, in most implementations, people simply use

$$\hat{\nabla}_\theta V^{\pi_\theta,\gamma}(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \nabla_\theta \ln(\pi_\theta(s_t, a_t)) \cdot \hat{A}^{\pi_\theta,\gamma}(s_t, a_t)\right].$$

Escaping Local Optima

- Because policy gradient is essentially first-order optimization of a non-convex function, no convergence is guaranteed.
- People use regularization (exploration term) to encourage escaping local optima.
- For example, instead of taking the gradient of

$$\theta' \mapsto \mathbb{E}\left[\sum_{t=0}^{\infty} \ln(\pi_{\theta'}(s_t, a_t)) \cdot \hat{A}^{\pi_\theta,\gamma}(s_t, a_t)\right]$$

at θ, people instead take the gradient of

$$\theta' \mapsto \mathbb{E}\left[\sum_{t=0}^{\infty} \ln(\pi_{\theta'}(s_t, a_t)) \cdot \hat{A}^{\pi_\theta,\gamma}(s_t, a_t) + \eta \cdot \text{ entropy}(\pi_{\theta'}(s_t, \cdot))\right]$$

at θ.

Case Study: Proximal Policy Optimization (PPO)

More Stable Policy Optimization

Now that we know how to estimate policy gradients, any method/trick that can be applied to general first-order optimization can in principle to be used for policy optimization.

Naive gradient descent does not work very well in practice: rollouts are expensive, so we typically do multiple gradient descents on the same set of rollouts, these descents will become more and more biased. Some remedies:

- TRPO (Trust Region Policy Optimization): encourages the updated policy to be close to the previous policy in terms of $\mathrm{KL}(\pi_{\mathrm{new}}(s,\cdot)||\pi_{\mathrm{old}}(s,\cdot))$, w.r.t. some distribution over s.
- PPO (Proximal Policy Optimization): encourages the updated policy to be close to the previous policy in terms of $\left|\frac{\pi_{\mathrm{new}}(s,\cdot)}{\pi_{\mathrm{old}}(s,\cdot)} - 1\right|$, w.r.t. some distribution over s.

PPO

PPO solves a local optimization problem at each step, and the solution to each problem forms an updated policy π. It repeats the following procedure:

- Sample multiple (say, 128) trajectories of certain length (say, 128) to get a batch of state-actions pairs (say, 128 * 128 (s, a) pairs)
- Estimate the advantage of each (s, a) pair using GAE (say GAE(0.95)) and the corresponding trajectory
- Do (mini-batch) gradient descent w.r.t to an objective function multiple times (say, mini-batch of size 128 * 32, 16 gradient descents)

PPO - Objective Function

Suppose the current batch is generated using π_θ, the regular (non-proximal) gradient update uses the following objective function at θ (note the gradient of the $\ln(\pi_{\theta'}(s,a))$ is expanded):

$$\theta' \mapsto \mathbb{E}_{(s,a)\sim\text{batch}} \left[\frac{\pi_{\theta'}(s,a)}{\pi_\theta(s,a)} \cdot \hat{A}^{\pi_\theta,\gamma}(s,a) + \eta \cdot \text{ entropy}(\pi_{\theta'}(s,\cdot))\right].$$

PPO use the following objective function for gradient descents at θ:

$$\theta' \mapsto \mathbb{E}_{(s,a)\sim\text{batch}} \left[\min\left(\frac{\pi_{\theta'}(s,a)}{\pi_\theta(s,a)} \cdot \hat{A}^{\pi_\theta,\gamma}(s,a), \text{clip}\left(\frac{\pi_{\theta'}(s,a)}{\pi_\theta(s,a)}, 1-\epsilon, 1+\epsilon\right) \cdot \hat{A}^{\pi_\theta,\gamma}(s,a)\right) + \eta \cdot \text{ entropy}(\pi_{\theta'}(s,\cdot))\right].$$

End