Taxonomy of RL Algorithms

Henry Zhang

School of Mathematicas
Sichuan University

March 11, 2025

Overview

1. Review on Reinforcement Learning (RL)
2. Clarification of Basic Concepts in RL
3. Classification of RL Algorithms

Reinforced Learning (RL)

The Reinforced Learning (RL) model includes agent(s) who take in environmental statess and rewardr from last action made by the agent(s)) and make an action *a* in order to maximize the long-term rewards.

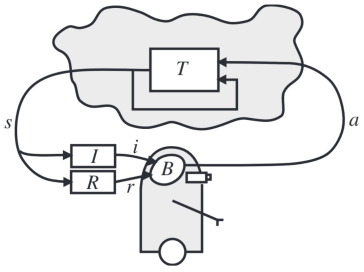


Figure: Standard RL model[a]

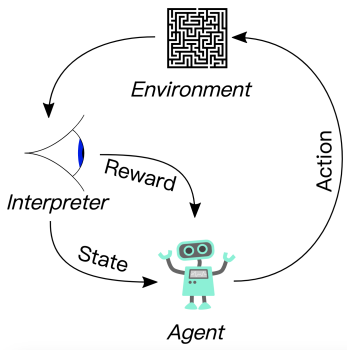


Figure: Interaction with environment

[a] Reinforcement Learning: A Survey (Kaelbling et al., 1996)

Markov Decision Process (MDP)

The Markov Decision Proces (MDP) consists of a state space $\mathcal{S}$, action space $\mathcal{A}$, horizon H, transition kernels P, where $P : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$ and known reward functions r, where $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. To interact with the MDP, we use policies of the form π s.t. $\pi : \mathcal{S} \mapsto \Delta(\mathcal{A})$, which takes actions given the current state at each step. The terminology Markov indicates that the state transitions are independent of any previous environment states or agent actions.

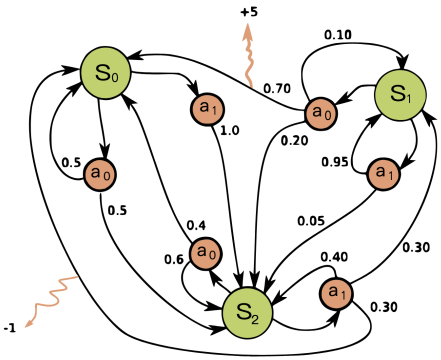


Figure: Example of a simple MDP with three states (green circles) and two actions (orange circles), with two rewards (orange arrows)

Finite-horizon vs. Infinite-horizon

- Finite-horizon model: Referring to a setting where the task has a fixed time limit or number of steps, after which the process ends (e.g., a game with finite moves [1] [2]).
  - Total reward: $\max_\pi \mathbb{E}\left(\sum_{h=1}^{H} r(s_h, a_h)\right)$

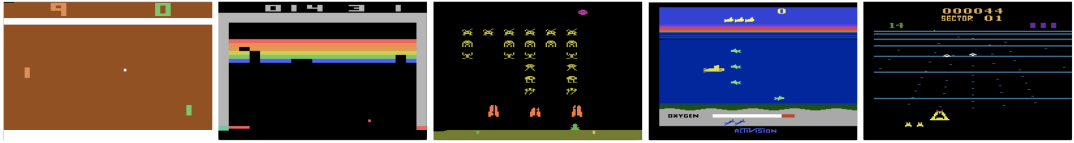


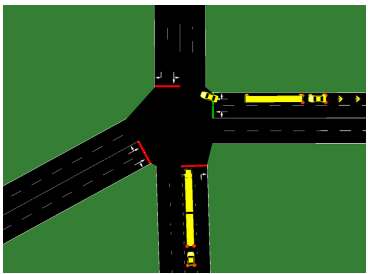
Figure: Screen shots from five Atari 2600 Games: (Left-to-right) Pong, Breakout, Space Invaders, Seaquest, Beam Rider

[1] A comparative study of urban traffic signal control with reinforcement learning and Adaptive Dynamic Programming (Mnih et al., 2013)

[2] Human-level control through deep reinforcement learning (Minh et al., Nature 2015)

Finite-horizon vs. Infinite-horizon

- Infinite-horizon model: Assuming the task continues indefinitely, with no fixed end (e.g., traffic signal control [3]).
  - Discounted reward: $\max_\pi \mathbb{E}\left(\sum_{h=1}^{\infty} \gamma^h r(s_h, a_h)\right)$
  - Averaged reward: $\max_\pi \left[\lim_{H\to\infty} \mathbb{E}\left(\frac{1}{H}\sum_{h=1}^{H} r(s_h, a_h)\right)\right]$



[3] A comparative study of urban traffic signal control with reinforcement learning and Adaptive Dynamic Programming (Dai et al., 2010)

Expectation Equation vs. Optimality Equation

Bellman Expectation Equation describes the expected value of a state (or state-action pair) under a specific policy, balancing immediate reward and future discounted value based on that policy: for the state-value function $V^{\pi}(s)$:

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

and for the action-value function $Q^{\pi}(s,a)$:

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

This equation recursively computes the expected reward for each state-action pair.

Expectation Equation vs. Optimality Equation[4]

Bellman Optimality Equation defines the optimal value of a state (or state-action pair), assuming the best possible actions are taken to maximize total reward, without following a fixed policy: for the optimal state-value function $V^\star(s)$:

$$V^\star(s) = \max_a \left[r(s,a) + \gamma \sum_{s'} P\left(s' \mid s, a\right) V^\star\left(s'\right) \right]$$

and for the optimal action-value function $Q^\star(s,a)$:

$$Q^\star(s,a) = r(s,a) + \gamma \sum_{s'} P\left(s' \mid s, a\right) \max_{a'} \left[Q^\star\left(s', a'\right) \right]$$

This equation recursively computes the optimal reward by selecting the best possible action at each time step. (Detailed proof can be found in MIT 6.7950 Lec3 pp.25-29)

[4] MIT 6.7950 Reinforcement Learning: Foundations and Methods (Cathy Wu, 2022 Fall)

Value Iteration vs. Policy Iteration

- **Value iteration**
  - Directly updates the **optimal state-value function** $V^\star$ by applying the Bellman optimality equation iteratively.

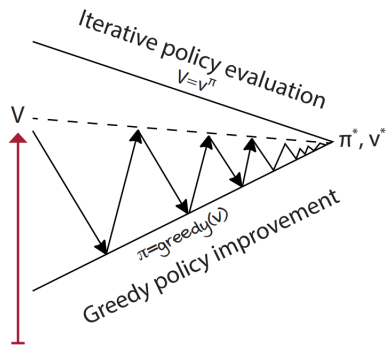
- **Policy iteration** alternates between two steps:
  - **Policy evaluation:** Computing the **state-value function** V^π for a given policy π via Bellman expectation equation.
  - **Policy improvement:** Updating the policy to be greedier with respect to the value function iteration.

Value Iteration vs. Policy Iteration[5]

1. During **iteration** 0, let $V_0(s)$ be any function $V_0 : \mathcal{S} \to \mathbb{R}$.
2. Apply the principle of optimality so that given V_i at iteration i, for all $s \in \mathcal{S}$ we compute

$$V_{i+1}(s) = \mathcal{T} V_i(s) \quad (\mathcal{T} : \text{Bellman Optimal Operator})$$
$$:= \max_{a \in \mathcal{A}} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[V_i(s') \right]$$

3. Terminate when V_i stops improving, e.g. when $\max_s |V_{i+1}(s) - V_i(s)|$ is small.
4. Return the greedy policy: $\pi_K(s) = \arg\max_{a \in \mathcal{A}} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} V_K(s')$



(a) Value iteration starts with an arbitrary value function and has a truncated policy evaluation step.

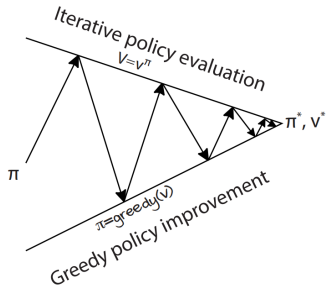
[5] Grokking Deep Reinforcement Learning (Morales, 2020)

Value Iteration vs. Policy Iteration[6]

1. Let π_0 be any stationary policy.
2. At each iteration $k = 1, 2, \ldots, K$
   - Policy evaluation: given π_k, compute V^{π_k}
   - Policy improvement: compute the greedy policy

$$\pi_{k+1}(s) \in \arg\max_{a \in \mathcal{A}} \left[r(s,a) + \gamma \sum_{s'} p\left(s' \mid s, a\right) V^{\pi_k}\left(s'\right) \right]$$

3. Stop if $V^{\pi_k} = V^{\pi_{k-1}}$.
4. Return the last policy π_K.



(i) Policy iteration consists of a full convergence of iterative policy evaluation alternating with greedy policy improvement.

[6] Grokking Deep Reinforcement Learning (Morales, 2020)

Algorithm Classification

- **Model-based:** Model-based methods rely on a model of the environment (e.g., transition probabilities and rewards) to plan and make decisions, learning or using this model to predict outcomes and optimize actions.
  - **Given Model** Assuming a known model (transition probabilities and reward functions) of the environment. (e.g. MCTS: AlphaGo & AlphaZero)
  - **Learned Model** (Supervised) Learning or approximating the environment model through interactions with the environment. (e.g. I2A/World Model)
- **Model-free:** Learning optimal policies directly from interactions with the environment without using a model of the environment.
  - **Value-Based:** Focusing on estimating the value (e.g. Q function) of states or state-action pairs to derive an optimal policy. (e.g. Q-Learning)
  - **Policy-based:** Directly optimizing the policy by adjusting the parameters of the policy function itself. (e.g. TRPO/DPO/PPO)

Algorithm Classification

- **Value-based:** Efficient in learning optimal policies from value functions but may struggle with continuous or large action spaces.
  - **On-policy** Updating the value function with each new interaction. (SARSA)
  - **Off-policy** Updating the value function based on a batch of data. (e.g. Q-learning, DQN)
  - **Distributional RL** Learning not just the expected value of returns but the entire probability distribution of returns. (e.g. Quantile Regression DQN, Categorical DQN)
  - **Non-Distributional RL** Learning the expected value of returns (the mean) without considering the full distribution. (e.g. Q-learning, SARSA)
- **Policy-based:** Suitable for complex action spaces, but tend to have higher variance and slower convergence.
  - **Gradient-free strageties:** Optimizing the policy through heuristic methods. (e.g. genetic algorithms)
  - **Gradient-based strageties:** Updating the policy parameters using the gradient of the objective function. (e.g. TRPO/DPO/PPO)

Model-based vs. Model-free

Here is an example[7]:

Figure: An illustration of model-based methods vs. model-free methods

[7] Behavioral Animation of Autonomous Virtual Agents Helped by Reinforcement Learning (Conde et al., 2003)

Model-based vs. Model-free

- **Model-based methods** rely on a *model of the environment* (e.g. transition probabilities and rewards function) to plan and make decisions.
  - Advantages: More sample-efficient (requires fewer interactions)[8];
  - Disadvantages: Requires an *accurate* model (errors can lead to poor performance); computationally expensive to build and use.
- **Model-free methods** learn directly from experience without modeling the environment, using trial-and-error.
  - Advantages: Simpler to implement; works well when environment is complex;
  - Disadvantages: Less sample-efficient (needs more data);

[8] Numerical Evidence for Sample Efficiency of Model-Based Over Model-Free Reinforcement Learning Control of Partial Differential Equations (Werner at al., 2024)

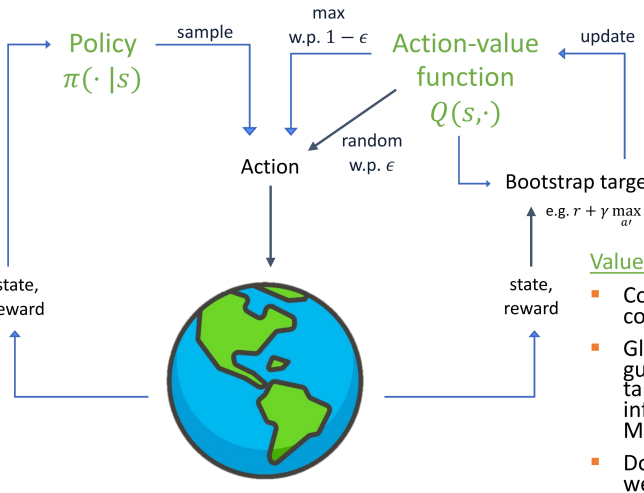
Model-based vs. Model-free[9]

horizon	algorithm	sample complexity	ε-range to attain sample optimality	type
infinite	VI-LCB (Rashidinejad et al., 2022)	$\frac{SC^\star}{(1-\gamma)^5\varepsilon^2}$	——	model-based
	Q-LCB (Yan et al., 2023)	$\frac{SC^\star}{(1-\gamma)^5\varepsilon^2}$	——	model-free
	VR-Q-LCB (Yan et al., 2023)	$\frac{SC^\star}{(1-\gamma)^3\varepsilon^2} + \frac{SC^\star}{(1-\gamma)^4\varepsilon}$	$(0, 1-\gamma]$	model-free
	VI-LCB (this paper: Theorem 1)	$\frac{SC^\star_{\text{clipped}}}{(1-\gamma)^3\varepsilon^2}$ $(\leq \frac{SC^\star}{(1-\gamma)^3\varepsilon^2})$	$\left(0, \frac{1}{1-\gamma}\right]$	model-based
	lower bound (this paper: Theorem 2)	$\frac{SC^\star_{\text{clipped}}}{(1-\gamma)^3\varepsilon^2}$	——	——

Figure: Comparisons with prior results (up to log terms) regarding finding an ε-optimal policy in offline RL. The ε-range stands for the range of accuracy level ε for which the derived sample complexity is optimal

[9] Settling the Sample Complexity of Model-Based Offline Reinforcement Learning (Li et al., 2022)

Value-based vs. Policy-based



Policy-based methods

- Sensible & simple ☺
- Local convergence guarantees only
- Also “works” for partial observation, nonstationary settings
- Doesn’t work that well (on its own) ☹

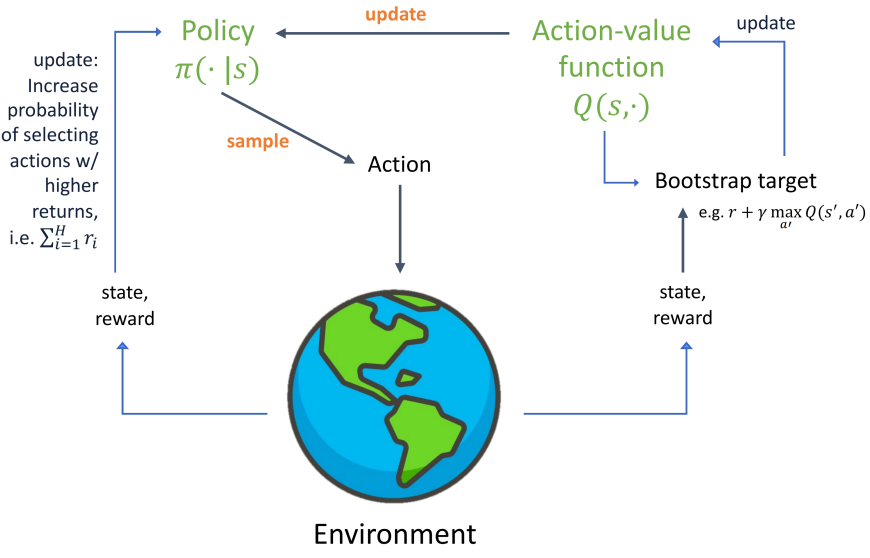
Value-based methods

- Conceptually more complicated
- Global convergence guarantees for tabular discounted infinite horizon MDPs ☺
- Doesn’t work that well outside of its comfort zone (on its own) ☹

Value-based vs. Policy-based

- **Value-based methods** estimate the value of states or actions (e.g., Q-values) and derive a policy by acting greedily w.r.t. these intermediate values.
  - Advantages: Effective in discrete action spaces; leverages Bellman equations for convergence;
  - Disadvantages: Struggles with continuous action spaces; prone to **overestimation** (an algorithm assigns unrealistically high values to actions or states, often due to noise, the max operator, or exploration strategies like ε-greedy).
- **Policy-based methods** directly learn a policy (e.g., a probability distribution over actions) without relying on value functions.
  - Advantages: Handles continuous action spaces well; more stable in high-dimensional settings;
  - Disadvantages: Can be sample-inefficient; may converge to suboptimal policies due to high variance.

Actor-Critic Methods



Actor-Critic Methods

- **Actor-Critic methods** integrate policy-based and value-based reinforcement learning. The *actor* learns a parameterized policy $\pi_\theta(a|s)$ (e.g., a neural network outputting action probabilities), deciding which actions to take in a given state. The *critic* estimates a value function, such as the state value $V_\phi(s)$ or action-value $Q_\phi(s, a)$, to assess the actor's performance and provide feedback for policy improvement.
  - Advantages: Reduces variance compared to pure policy gradient methods (e.g., REINFORCE) by using a critic; faster convergence than value-based methods alone; works in continuous action spaces;
  - Disadvantages: More complex to implement (two networks to train); sensitive to bias and may lead to overestimation.

Monte Carlo vs. Temporal Difference

- **Monte Carlo methods** estimate value functions (e.g., $V(s)$ or $Q(s, a)$) by averaging returns from *complete episodes*, using sampled full trajectories.
  - Advantages: Unbiased estimates since they use actual returns; simple to implement with no model assumptions;
  - Disadvantages: High variance due to reliance on full episode outcomes; sample-inefficient as updates require episode completion.
- **TD methods** update value estimates incrementally using bootstrapping (e.g., $V(s) \leftarrow V(s) + \alpha(r + \gamma V(s') - V(s))$), blending current rewards and future predictions.
  - Advantages: Lower variance by smoothing updates over time; more sample-efficient with step-by-step learning;
  - Disadvantages: Biased estimates due to reliance on initial value guesses; sensitive to inaccuracies in early value functions.

Monte Carlo vs. Temporal Difference

Here is an illustration[10] of these methods:

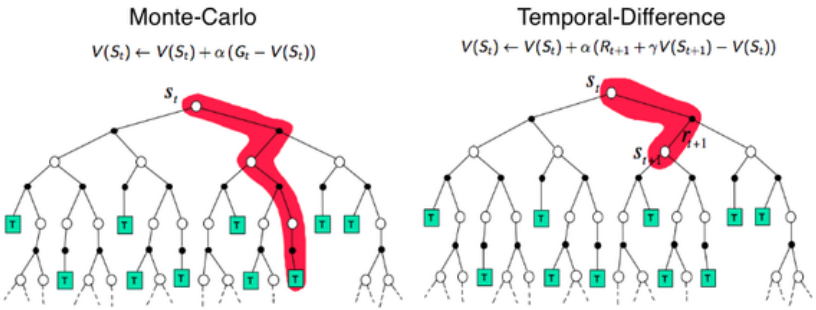


Figure: Comparison of the rollout search space (or backup diagrams) of Monte Carlo (MC) and Temporal Difference (TD) for state value functions (from David Silver's RL lecture). Additional notation: G_t is the final return, or expected future reward, $G_t = \sum_\tau \gamma^\tau r_{t+\tau+1}$.

[10] Reinforcement Learning and Bandits for Speech and Language Processing: Tutorial, Review and Outlook (Lin, 2022)

On-Policy vs. Off-Policy Methods

- **On-policy methods** evaluate and improve the policy that is used to make decisions, learning from data generated by the same policy.
  - Advantages: Simpler to implement; more stable with function approximation;
  - Disadvantages: Less sample-efficient (cannot reuse data from old policies); exploration is constrained by the current policy.
- **Off-policy methods** learn about a target policy using data from a different behavior policy, allowing for more flexible data sources.
  - Advantages: More sample-efficient (can learn from diverse data); enables separate exploration strategies;
  - Disadvantages: Prone to instability (especially with function approximation); more complex to implement (often requires importance sampling or experience replay).

On-Policy vs. Off-Policy Methods

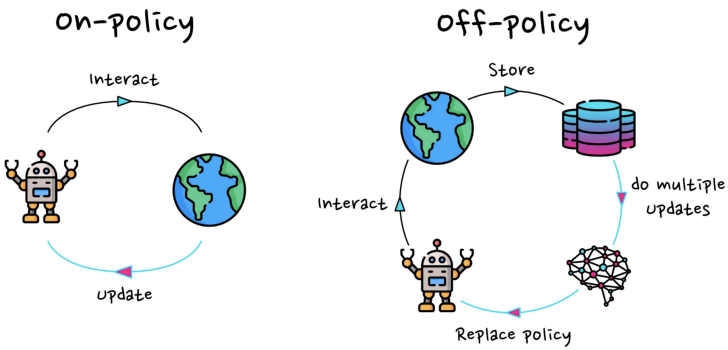


Figure: An illustration of on-policy RL and off-policy RL.

Thanks!