Introduction to Reinforcement Learning

David Quarel

September 8, 2026

What is Reinforcement Learning?

- Deep learning is usually presented in the supervised learning setting:
    1. We have access to labelled training data
    2. Agent only sees "real world" once it's already performing well
    3. Data is i.i.d between batches
    4. No planning required, future predictions don't depend on past predictions
- RL is vastly different: Agent takes actions in an interactive environment, receives scalar reward as feedback. This lends itself to several problems:
    1. **Sparse reward:** Sparse feedback to do good
    2. **Reward attribution:** Which action mattered?
    3. **No ground truth:** Optimal policy unknown
    4. **Explore vs. Exploit tradeoff:**
        1. **Exploration:** Learn how the world works
        2. **Exploitation:** Exploit best understanding to get reward
    5. **(often) Online only:** No clear distinction between training and testing.

Y. LeCun

How Much Information is the Machine Given during Learning?

- **“Pure” Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

 1.1: Deep Learning Hardware: Past, Present, & Future 59

Y. LeCun

How Much (cherry) is the Machine Given during Learning?

- "Pure" (cherry) Learning (cherry)
  - The machine predicts a scalar reward given once in a while.
  - Millions of (cherry) samples

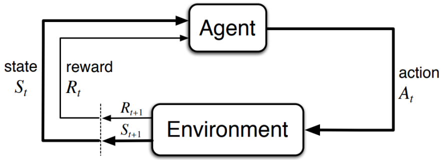
[redacted]

[redacted]

Agent-Environment Interaction Loop (MDPs)

- MDP is 4-tuple $(\mathcal{S}, T, \mathcal{A}, R)$:
  - *states* $\mathcal{S}$
  - *actions* $\mathcal{A}$
  - *transition distribution* $T : \mathcal{S} \times \mathcal{A} \to \Delta\mathcal{S}$.
    $T(s'|s, a) = \Pr(s_{t+1} = s'|s_t = s, a_t = a)$.
  - *reward function* $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$
- Agent has *policy* $\pi : \mathcal{S} \to \Delta\mathcal{A}$. $\pi(a|s) = \Pr(a_t = a|s_t = s)$.
- On timestep t,
  - Agent samples $a_t \sim \pi(\cdot|s_t)$, gives action a_t to environment.
  - Environment samples $s_{t+1} \sim T(\cdot \mid s_t, a_t)$, computes $r_{t+1} = R(s_t, a_t, s_{t+1})$, and gives *percept* (s_{t+1}, r_{t+1}) to agent.
- Generates an interaction history, or *trajectory*

$$s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, \ldots$$



Objective of the Agent

- At timestep t, the *return* G_t is the sum of all future rewards:

$$G_t = r_{t+1} + r_{t+2} + r_{t+3} + \ldots$$

- **Goal**: Maximise the return.
  - For episodic (finite length interaction) environments of maximum duration T, return $G_t = r_{t+1} + r_{t+2} + \ldots + r_T$ is well defined.
- **Problems:** (for continuing environments)
  - The return may diverge or be undefined
  - The agent might be lazy
  - The environment is stochastic, and the rewards are often up to chance. How to trade-off unlikely big rewards with likely small rewards?
  - May desire rewards now to be more valuable than rewards later: \$100 now? Or \$110 in a year?
- **Solutions:**
  - Add a discount factor $\gamma \in [0, 1)$ so rewards more imminent are worth more, and the return is always well defined.

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots$$

  - Want agent to choose actions to maximise the *expected* return.

Core Assumptions

These kinds of environments are called *Markov Decision Processes* (MDPs), and have the following “nice” properties

1. **Stationary:** The environmental distribution T is fixed and does not change over time
2. **Markovian:** The behaviour of the environment depends only on the current state s_t and action a_t.
3. **Fully Observable:** The state is a full description of the world
4. **Reward Hypothesis:**

   > *“That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward).” -Rich Sutton*

Value Function

- Want to define the "goodness" (*value*) of a state, so the agent can take actions to move towards "good" states, and away from "bad" states.
- The value of a state depends also on the current policy π chosen by the agent.

Value Function

$$V_\pi(s) = \mathbb{E}_\pi[G_t | s_t = s]$$

(Expectation is also with respect to the transition distribution T.)

Bellman Equation

We note that since

$$\begin{aligned} G_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots \\ &= r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \dots) \\ &= r_{t+1} + \gamma G_{t+1} \end{aligned}$$

we can then define the value function recursively,

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_\pi[G_t \mid s_t = s] \\ &= \mathbb{E}_\pi[r_{t+1} + \gamma G_{t+1} \mid s_t = s] \\ &= \mathbb{E}_\pi[r_{t+1} \mid s_t = s] + \gamma \mathbb{E}_\pi[G_{t+1} \mid s_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} T(s'|s,a) R(s,a,s') \\ &\quad + \gamma \sum_a \pi(a|s) \sum_{s'} T(s'|s,a) \mathbb{E}_\pi[G_{t+1} \mid s_{t+1} = s'] \\ &= \sum_a \pi(a|s) \sum_{s'} T(s'|s,a) \left(R(s,a,s') + \gamma V_\pi(s')\right) \end{aligned}$$

Policy Evaluation

This gives the **Bellman equation**

Bellman Equation

$$V_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} T(s'|s,a) \left(R(s,a,s') + \gamma V_\pi(s') \right)$$

- Equation is linear in $V_\pi(\cdot)$, giving a set of **linear** simultaneous equations.
- Given policy π, can now easily solve for $V_\pi(s_1), V_\pi(s_2), \dots$
- Computing V_π from π is called **policy evaluation**, denote $\pi \xrightarrow{E} V_\pi$.

Bellman Equation (stochastic policy)

$$V_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} T(s'|s,a)\left(R(s,a,s') + \gamma V_\pi(s')\right)$$

Assuming policy $\pi : S \to A$ is deterministic

Bellman Equation (deterministic policy)

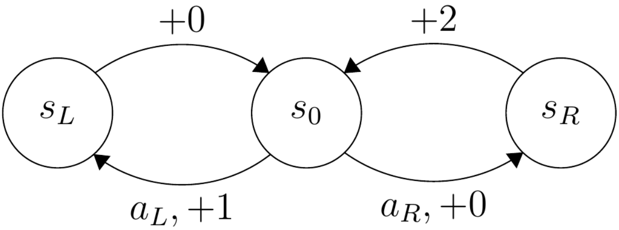
$$V_\pi(s) = \sum_{s'} T(s'|s,a)\left(R(s,a,s') + \gamma V_\pi(s')\right)$$

where $a = \pi(s)$.

Only need to sum over all states to find equation for $V_\pi(s)$ in terms of the value of all states $\{V_\pi(s')\}_{s' \in \mathcal{S}}$.

Example Environment

- States $\mathcal{S} = \{s_0, s_L, s_R\}$, actions $\mathcal{A} = \{a_L, a_R\}$, possible rewards $\mathcal{R} = \{0, 1, 2\}$.
- Each transition indicates if an action is taken, the reward returned and which state to transition to
- What is the best action from state s_0?



Exercise: Depends on the choice of γ! Verify that

$$V_{\pi_L}(s_0) = \frac{1}{1-\gamma^2} \quad V_{\pi_R}(s_0) = \frac{2\gamma}{1-\gamma^2}$$

so breakeven is at $\gamma = \frac{1}{2}$.

Optimal Bellman

- Policy π_1 is **better** than π_2 ($\pi_1 \geq \pi_2$) if $\forall s. V_{\pi_1}(s) \geq V_{\pi_2}(s)$. A policy is **optimal** if it is better than all other policies.
- **Theorem:** An optimal policy π^* always exists (may not be unique). Define optimal value function as

$$V_*(s) := \max_\pi V_\pi(s) \equiv V_{\pi^*}(s)$$

Optimal Bellman Equation

$$V_*(s) = \max_a \sum_{s'} T(s'|s,a)\left(R(s,a,s') + \gamma V_*(s')\right)$$

Gives a set of **non-linear** simultaneous equations with variables $V_*(s_1), V_*(s_2), \ldots$
Problem: No clear way to solve for $V_*(\cdot)$
Can't just compute V_π using policy evaluation for all π, as there are $|\mathcal{A}|^{|\mathcal{S}|}$ many to choose from.

Policy Improvement

- Obviously we have that

$$
\begin{aligned}
V_*(s) =& \max_a \sum_{s'} T(s'|s,a)\left(R(s,a,s') + \gamma V_*(s')\right) && (1)\\
\geq& \max_a \sum_{s'} T(s'|s,a)\left(R(s,a,s') + \gamma V_\pi(s')\right) && (2)\\
\geq& \sum_{s'} T(s'|s,\pi(s))\left(R(s,\pi(s),s') + \gamma V_\pi(s')\right) && (3)\\
=& V_\pi(s) && (4)
\end{aligned}
$$

- Given the value of a policy V_π, can feed it through the optimal Bellman equation (2) to get a better policy π'. Denote $V_\pi \xrightarrow{I} \pi'$.

Policy Improvement

$$\pi'(s) := \arg\max_a \sum_{s'} T(s'|s,a)\left(R(s,a,s') + \gamma V_\pi(s')\right)$$

Exercise: Prove $\pi' \geq \pi$.

Policy Iteration

Policy Improvement (I)

$$\pi'(s) \leftarrow \arg\max_a \sum_{s'} T(s'|s,a)\left(R(s,a,s') + \gamma V_\pi(s')\right)$$

Policy Evaluation (E)

Solve

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s'} T(s'|s,a)\left(R(s,a,s') + \gamma V_\pi(s')\right)$$

for $V_\pi(s_1), V_\pi(s_2), \ldots$.

- Start with arbitrary policy π_0.
- Note that π_* is fixed point of policy improvement.
- Alternate until policy is stable

$$\pi_0 \xrightarrow{E} V_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V_{\pi_1} \xrightarrow{E} \pi_2 \xrightarrow{I} V_{\pi_2} \xrightarrow{E} \ldots \xrightarrow{I} \pi_* \xrightarrow{E} V_{\pi^*} \xrightarrow{I} \pi_*$$

Theorem: Policy iteration converges to optimal policy in finitely many steps!

Slippery Gridworld

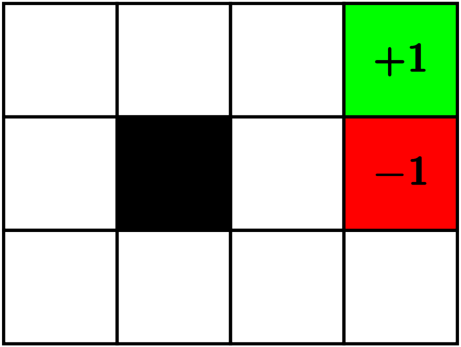
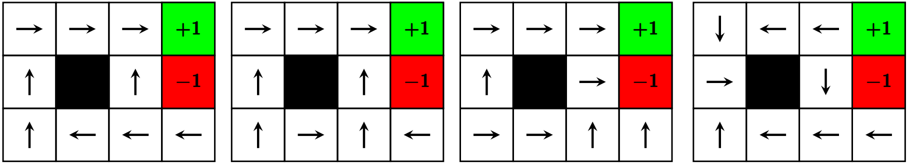


Figure: The slippery gridworld environment. The agent can move in any of the four cardinal directions. Receives reward (punishment?) $\epsilon < 0$ on each timestep. Movement is noisy, with a 70% chance of moving in the desired direction, and 10% of moving in the other three directions. Episode ends when the agent lands in **heaven** $\boxed{+1}$ (and gets reward $+1$) or in **hell** $\boxed{-1}$ (and gets reward -1). The wall is impassible; any movement that would walk into the wall, or off the grid, has no effect.

Optimal Policy for Slippery Gridworld



(a) $\varepsilon = -0.01$ (b) $\varepsilon = -0.1$ (c) $\varepsilon = -2$ (d) $\varepsilon = 0.01$

- (a) **Patient:** The penalty is slight, so the agent takes the long way around to get to heaven.
- (b) **Risky:** The penalty is moderate, so the agent takes the shortest path to heaven, risking hell along the way.
- (c) **Agony:** The penalty is severe, so the agent wants to end the episode as quickly as possible.
- (d) **Bliss:** The reward is positive on each timestep, so the agent runs away from heaven or hell to drag out the episode as long as possible.

Glossary

- **Transition** distribution $T : \mathcal{S} \times \mathcal{A} \to \Delta\mathcal{S}$, sample $s' \sim T(\cdot|s, a)$.
- **Reward** function $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, write **reward** $r_{t+1} := R(s_t, a_t, s_{t+1})$
- **Policy** $\pi : \mathcal{S} \to \Delta\mathcal{A}$ (stochastic) or $\pi : \mathcal{S} \to \mathcal{A}$ (deterministic), take $a_t = \pi(s_t)$ or sample $a_t \sim \pi(\cdot|s_t)$.
- **History/Trajectory** $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, \ldots$
- **Agent** is the algorithm that selects the *policy* based on the *history*.
- **Discount** $\gamma \in [0, 1)$. Adjusts how far-sighted agent is. $\gamma = 0$ is *myopic*. $\gamma \to 1$ is *far-sighted*.
- **Return** $G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = r_{t+1} + \gamma G_{t+1}$ is sum of *future discounted rewards*.
- **Goal** is to **maximise** G_t.
- **Value** function $V_\pi(s) = \mathbb{E}_\pi[G_t|s_t = s]$.
- **Bellman** equation $V_\pi(s) = \sum_{s' \in \mathcal{S}} T(s'|s, a)\,(R(s, a, s') + \gamma V_\pi(s'))$
- **Policy Improvement** $\pi'(s) \leftarrow \arg\max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} T(s'|s, a)\,(R(s, a, s') + \gamma V_\pi(s'))$
- **Policy Evaluation** Solve $V_\pi(s) = \sum_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} T(s'|s, a)\,(r + \gamma V_\pi(s'))$ for $V_\pi(s_1), V_\pi(s_2), \ldots$.

ΔX is the set of all probability vectors $[p_1, \ldots, p_n]$ over outcomes in X.

Problems with Policy Iteration

Policy Improvement (I)

$$\pi_{n+1}(s) \leftarrow \arg\max_a \sum_{s'} T(s'|s,a)\left(R(s,a,s') + \gamma V_{\pi_n}(s')\right)$$

Policy Evaluation (E)

Solve

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s'} T(s'|s,a)\left(R(s,a,s') + \gamma V_\pi(s')\right)$$

for $V_\pi(s_1), V_\pi(s_2), \ldots$.

- Requires white-box access to the environmental distribution T and reward function R.
- Only works for environments with few enough states and actions to sweep through.

For the moment, we weaken only the first assumption. Environment is black box, sample percepts (s', r) given state-action pair (s, a) from unknown T and R.

Temporal Difference Learning

- **Goal:** Perform policy evaluation without access to environmental distribution.
- **Motivation:** Consider once again the value function:

$$V_\pi(s) = \mathop{\mathbb{E}}_{\substack{a=\pi(s) \\ s' \sim T(\cdot|s,a)}} [R(s,a,s') + \gamma V_\pi(s')]$$

  On timestep t, this is *on average*, equal to the actual reward r_{t+1}, plus the discounted value of the actual next state s_{t+1}.

$$V_\pi(s_t) \approx r_{t+1} + \gamma V_\pi(s_{t+1})$$

We define the **TD-Error** as the difference

$$\delta_t := r_{t+1} + \gamma V_\pi(s_{t+1}) - V_\pi(s_t)$$

This then gives us an update rule to improve on our estimate $\hat{V}_\pi$ of V_π, similar to SGD, called *TD(0)*.

$$\hat{V}_\pi(s_t) \leftarrow \hat{V}_\pi(s_t) + \alpha\delta_t \equiv \hat{V}_\pi(s_t) + \alpha\left(r_{t+1} + \gamma \hat{V}_\pi(s_{t+1}) - \hat{V}_\pi(s_t)\right)$$

where $\alpha \in (0,1]$ is the **learning rate**.
Theorem: Given $\alpha \to 0$ in a nice way, TD(0) guaranteed to converge to V_π.

Q-Value

- **Q-value** is the expected return from state s, taking action a, and thereafter following policy π.

$$Q_\pi(s,a) = \mathbb{E}_\pi[G_t | s_t = s, a_t = a]$$

Contrast with the value function

$$V_\pi(s) = \mathbb{E}_\pi[G_t | s_t = s]$$

Q-value Bellman

$$Q_\pi(s,a) = \sum_{s'} T(s'|s,a)\left(R(s,a,s') + \gamma Q_\pi(s',a')\right)$$

where $a' = \pi(s')$

Optimal Q-value Bellman

$$Q_*(s,a) = \sum_{s'} T(s'|s,a)\left(R(s,a,s') + \max_{a'} Q_*(s',a')\right)$$

Q-value vs. Value

Exercise: Can state Q in terms of V, and vice-versa.

$$Q_\pi(s,a) = \sum_{s'} T(s'|s,a)\left(R(s,a,s') + \gamma V_\pi(s')\right)$$
$$V_\pi(s) = \sum_{s'} T(s'|s,\pi(s))\left(R(s,a,s') + \gamma Q_\pi(s',\pi(s'))\right)$$

$$Q_*(s,a) = \sum_{s'} T(s'|s,a)\left(R(s,a,s') + \gamma V_*(s')\right)$$
$$V_*(s) = \max_a \sum_{s'} T(s'|s,a)\left(R(s,a,s') + \gamma \max_{a'} Q_*(s',a')\right)$$

Motivation for Q-Value

- So far, we have been learning a policy π, and using π to compute V_π.
- Even if we were given V_* directly, can't recover π_* without white-box access to T and R (environment).

$$\pi_*(s) = \arg\max_a \sum_{s'} T(s'|s,a)\left(R(s,a,s') + \gamma V_*(s')\right)$$

- However, given Q_*, we can directly recover π_*

$$\pi_*(s) = \arg\max_a Q_*(s,a)$$

- **Idea:** Learn Q_* instead, argmax to recover policy π_*

SARSA: On-Policy TD Control

Apply same argument as TD(0) to the Q-Value

$$Q_*(s,a) = \mathop{\mathbb{E}}_{s' \sim T(\cdot|s,a)} \left[R(s,a,s') + \gamma Q_*(s', \pi_*(s'))\right]$$

On timestep t, this is "*on average*", equal to the actual reward r_{t+1}, plus the discounted Q-value of the actual next state-action pair s_{t+1}, a_{t+1}.

$$Q_*(s_t, a_t) \approx r_{t+1} + \gamma Q_*(s_{t+1}, a_{t+1})$$

Update depends on $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$.

SARSA Update Rule

$$\hat{Q}_*(s_t, a_t) \leftarrow \hat{Q}_*(s_t, a_t) + \alpha \left(r_{t+1} + \gamma \hat{Q}_*(s_{t+1}, a_{t+1}) - \hat{Q}_*(s_t, a_t) \right)$$

where $\alpha \in (0, 1]$ is the **learning rate**.

Actions drawn from ε-greedy strategy

$$\pi^{\varepsilon\text{-greedy}}(s) = \begin{cases} \text{do random action} & \text{prob } \varepsilon \\ \arg\max_a \hat{Q}_*(s,a) & \text{prob } 1 - \varepsilon \end{cases}$$

Theorem: Under "niceness" conditions SARSA guaranteed to converge to Q_*.

Q-Learning: Off-Policy TD Control

- Why learn from a_{t+1} when it was a random exploration action? Why not instead learn from the best[1] action $\arg\max_{a'} Q(s_{t+1}, a')$ that should have been taken?

Q-Learning Update Rule

$$\hat{Q}_*(s_t, a_t) \leftarrow \hat{Q}_*(s_t, a_t) + \alpha \left(r_{t+1} + \gamma \max_{a'} \hat{Q}(s_{t+1}, a') - \hat{Q}(s_t, a_t) \right)$$

Actions taken via ε-greedy strategy over $\hat{Q}_*(s, a)$.

Theorem: Under "niceness" conditions Q-learning guaranteed to converge to Q_*.

[1] according to the current Q-value estimates

SARSA vs. Q-Learning

SARSA Update Rule

$$\hat{Q}_*(s_t, a_t) \leftarrow \hat{Q}_*(s_t, a_t) + \alpha \left(r_{t+1} + \gamma \hat{Q}_*(s_{t+1}, a_{t+1}) - \hat{Q}_*(s_t, a_t) \right)$$

Q-Learning Update Rule

$$\hat{Q}_*(s_t, a_t) \leftarrow \hat{Q}_*(s_t, a_t) + \alpha \left(r_{t+1} + \gamma \max_{a'} \hat{Q}(s_{t+1}, a') - \hat{Q}(s_t, a_t) \right)$$

- Q-Learning (usually) tends to converge faster than SARSA, and chooses more aggressive/risky moves
- SARSA learns from the moves that were actually taken, including any exploration
- In "risky" environments, SARSA will learn to avoid getting near dangerous situations (to avoid accidentally taking a very bad exploratory move). Q-Learning will not.