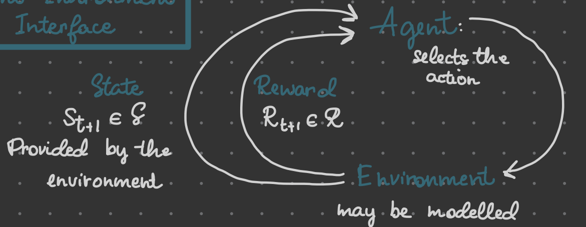
Reinforcement Learning Cheat Sheet

Fundamentals of RL

Agent-Environment Interface



Action: affects immediate reward + future states
$A_t \in \mathcal{A}(S_t)$

$S_t \longrightarrow A_t \longrightarrow S_{t+1}$

Goal: learn policy $\pi_t(a|s)$: maps state $\mapsto$ action taken, to maximize returns
May be deterministic: $\pi(s) = a$; or stochastic: $\pi(a|s) \geq 0$, $\sum_a \pi(a|s) = 1$.

Markov Decision Process (MDP)

: general framework for sequential decision making, 5-tuple

$(\mathcal{S}, \mathcal{A}, \mathcal{R}, \gamma, p)$

- $\mathcal{S}$: Finite space of states
- $\mathcal{A}$: Finite space of actions
- $\mathcal{R}$: Set of rewards $\mathcal{R} \subseteq \mathbb{R}$
- $p: \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \longrightarrow [0,1]$

$$\sum_{s',r} \sum_{s,a} p(s',r|s,a) = 1$$

Return: $G_t \equiv \sum_{k=0}^{H} \gamma^k R_{t+k+1}$

Horizon may be ∞ ; γ: Discount factor

- If H finite: episodic task.
  - ↳ terminal state.
- If $H = +\infty$: continuing task.
  - ↳ no terminal state.

$G_t = R_{t+1} + \gamma G_{t+1}$

State-value function: $v_\pi(s) \equiv \mathbb{E}_\pi[G_t | S_t = s]$

Action-value function: $q_\pi(s,a) \equiv \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$$

Bellman's Equations

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) \times [r + \gamma v_\pi(s')]$$

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a) \times [r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s',a')]$$

Theorem: π_1 better than π_2 i.o.i $v_{\pi_1}(s) \geq v_{\pi_2}(s)$ $\forall s \in \mathcal{S}$.
There is always at least one optimal policy π_* (i.e. as good or better than any other).

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

$\Rightarrow$ $q_*(s,a) = \max_\pi q_\pi(s,a)$, $v_*(s) = \max_a q_*(s,a)$

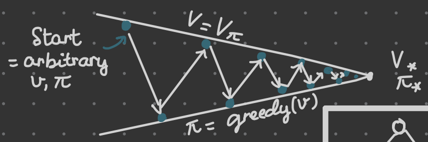
Dynamic Programming (DP)

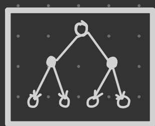
- **Bootstrapping**: use at least one estimated value in the update state for the same kind of expected value
- Requires a model.

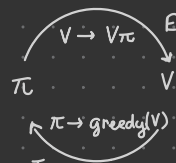
Policy Evaluation (Prediction): Determine a policy's value function.
Iteration (Control): Find a policy to maximize value functions.
DP: do both tasks iteratively.

Bellman's equations $\mapsto$ update rule. $\underset{\text{updated value}}{V'(s)} = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\,[r + \gamma \underset{\text{old estimate}}{V(s')}]$

Policy Improvement Theorem: $\forall s,\ q_\pi(s, \pi'(s)) \geq q_\pi(s, \pi(s)) \Rightarrow \pi' \geq \pi$
AND if $\exists s$ such that $q_\pi(s, \pi'(s)) > q_\pi(s, \pi(s)) \Rightarrow \pi' > \pi$.
Greedified Policy = strict improvement unless already optimal.




DP



Evaluation: sweep until $|V'(s) - V(s)| \leq \theta$ (threshold)

Improvement: greedify π w.r.t V. If π is stable, $\pi \simeq \pi_*$, $V \simeq V_*$

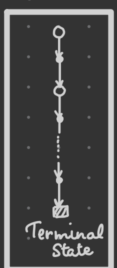
General Policy Iteration (GPI): all algorithms iteratively performing evaluation + improvement.

Sample-based Learning Methods

Monte Carlo methods (MC)

- No bootstrapping
- Model-free.

Relies on repeated random sampling $\mapsto$ averages sampled returns.



MC

- Generate an episode according to π, recording $S_0, A_0, R_1, S_1, A_1, \dots S_{T-1}, A_{T-1}, R_T$
- From $T-1 \to 0$: $G = \gamma G + R_{t+1}$. Keep track of multiple observed returns.

  $V(S_t) = \underset{\text{over generated episodes where } S_t \text{ was visited}}{\text{average}}[G]$ or $Q(S_t, A_t) = \underset{\text{over generated episodes where } S_t, A_t \text{ was visited}}{\text{average}}[G]$

  For non-stationary problems: $V'(S_t) = V(S_t) + \underset{\text{learning rate}}{\alpha}[G_t - V(S_t)]$
- To maintain exploration:
  - Exploring starts: A_0, S_0 picked at random.
  - ϵ-greedy: $\pi(a|S_t) = \begin{cases} 1 - \epsilon + \dfrac{\epsilon}{|A(S_t)|} & \text{if } a = \arg\max_a Q(S_t, a); \\ \dfrac{\epsilon}{|A(S_t)|} & \text{otherwise.} \end{cases}$

Off-Policy Learning

Use two policies: → target $\pi(a|s)$ – evaluated policy.
→ behaviour $b(a|s)$ – used to generate episodes.

Introduce importance sampling ratio $\rho_{t:T-1} = \prod_{k=t}^{T-1} \dfrac{\pi(A_k|S_k)}{b(A_k|S_k)}$.

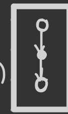
$v_\pi(s) = \mathbb{E}_b[\rho_{t:T-1} G_t | S_t = s]$.

Temporal Difference Learning (TD)

- Bootstrapping.
- Model-free.

$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$ MC: uses samples to estimate $\mathbb{E}_\pi[\,]$.

$= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$ TD: uses samples to estimate $\mathbb{E}_\pi[\,]$ & $v_\pi(S_{t+1})$

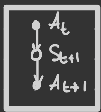


TD

TD update:

$$V'(S_t) = V(S_t) + \alpha[\underbrace{R_{t+1} + \gamma V(S_{t+1}) - V(S_t)}_{\text{TD error } \delta_t}]$$

Sarsa:



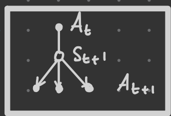
SARSA

$$Q'(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

On-policy

A_{t+1} chosen from π before updating.

Expected Sarsa



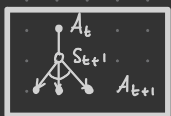
Expected SARSA

$$Q'(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \sum_{a'} \pi(a' \mid S_{t+1}) Q(S_{t+1}, a') - Q(S_t, A_t)]$$

Lower variance, more computationally expensive.

Can be used off-policy.

Q-learning



Q-learning

$$Q'(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

Special case of expected SARSA

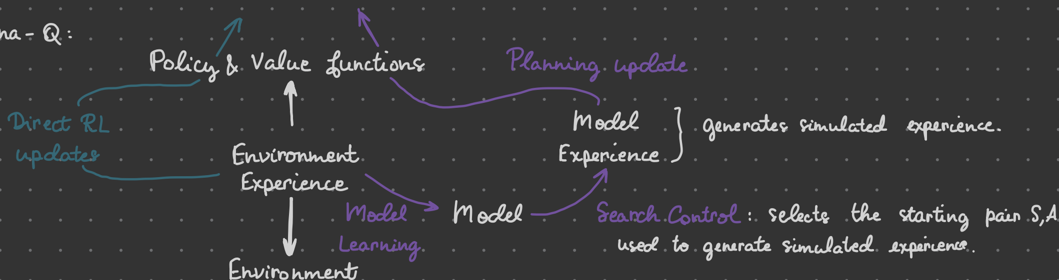
Planning

using a model to improve policy.

Q-planning:

1. Sampling: (S, A) picked at random, use sample model for S', R.
2. Q-learning update for (S, A)
3. Greedify π w.r.t $Q(S, A)$

Dyna-Q:



To force exploration: $R \to R + k\sqrt{\tau}$

k: hyperparameter; τ: time since transition was last visited

Amélie Chatelain