RL: Agent-oriented learning—learning by interacting with an environment to achieve a goal; Learning by trial and error, with only delayed evaluative feedback (reward).

- No supervisor, only a reward signal
- Feedback is delayed, not instantaneous
- Time matters, i.e., sequential decision making
- Agent's actions affect the subsequent data/ feedback from the environment

RL Problem is a considerable abstraction of the problem of goal-directed learning from interaction with the environment. **RL methods/solutions** specify how the agent changes its policy as a result of its experience. The agent's **goal** is to maximize the total amount of reward it receives over the long run.

Bandit Problem (repeatedly take a choice, reward from stationary prob dist, maximize total reward)

Action-Value Methods

$$Q_t(a) = \frac{R_1 + R_2 + \cdots + R_{K_a}}{K_a}$$

Policies: Greedy policy, ϵ-greedy policy. The greedy method performs significantly worse in the long run because it often gets stuck performing suboptimal actions. (Exploitation vs. exploration dilemma)

Softmax (choose action a at t-th play with prob:)

$$\frac{e^{Q_t(a)/\tau}}{\sum_{i=1}^{n} e^{Q_t(i)/\tau}}$$

τ is a positive parameter called the temperature

- Large temp: all (nearly) equiprobable selection
- Small temp: greedy action selection

Algorithm Implementation

$$Q_{k+1} = \frac{1}{k}\left(R_k + kQ_k - Q_k\right)$$

$$= Q_k + \frac{1}{k}\left[R_k - Q_k\right]$$

Choose constant step size for non-stationary problem. Weight recent reward more heavily.

Markov Property: the future is independent of the past, given the present. A Markov Process is a seq of random states with Markov prop, a tuple <S,P>.

State Transition Probability Matrix

$$\mathcal{P} = \begin{matrix} \\ from \end{matrix} \overset{\begin{matrix} to \end{matrix}}{\begin{bmatrix} \mathcal{P}_{11} & \cdots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \cdots & \mathcal{P}_{nn} \end{bmatrix}}$$

Markov Reward Process (MRP): a Markov process with reward values, a tuple $< S, P, R, \gamma >$

$$\mathcal{R}_s = \mathbb{E}\left[R_{t+1} \,|\, S_t = s\right]$$

Return: the total discounted reward from time t

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- γ close to 0 leads to short-sighted evaluation
- γ close to 1 leads to far-sighted evaluation

State-Value Function: expected return starting from state s

$$v(s) = \mathbb{E}\left[G_t \,|\, S_t = s\right]$$

Bellman Equation for MRP

$$v(s) = \mathbb{E}\left[R_{t+1} + \gamma\, v\left(S_{t+1}\right) \,|\, S_t = s\right]$$

$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v\left(s'\right)$$

Bellman Equation in Matrix Form

$$v = \mathcal{R} + \gamma \mathcal{P} v$$

$$v = (I - \gamma \mathcal{P})^{-1} \mathcal{R}$$

Markov Decision Process (MDP): MRP with decisions, a tuple $< S, A, P, R, \gamma >$

Policies: a distribution over actions given states

$$\pi(a \,|\, s) = \mathbb{P}\left[A_t = a \,|\, S_t = s\right]$$

$$\mathcal{P}^{\pi}_{s,s'} = \sum_{a \in \mathcal{A}} \pi(a \,|\, s) \mathcal{P}^{a}_{ss'}$$

$$\mathcal{R}^{\pi}_{s} = \sum_{a \in \mathcal{A}} \pi(a \,|\, s) \mathcal{R}^{a}_{s}$$

Value Functions for MDP

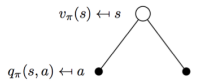
$$v_\pi(s) = \mathbb{E}_\pi\left[G_t \,|\, S_t = s\right]$$

$$q_\pi(s,a) = \mathbb{E}_\pi\left[G_t \,|\, S_t = s, A_t = a\right]$$

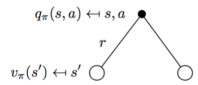
Bellman Expectation Equation for MDP

$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma\, v_\pi\left(S_{t+1}\right) \,|\, S_t = s\right]$$

$$q_\pi(s,a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma q_\pi\left(S_{t+1}, A_{t+1}\right) \,|\, S_t = s, A_t = a\right]$$



$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \,|\, s) q_\pi(s,a)$$



$$q_\pi(s,a) = \mathcal{R}^{a}_{s} + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^{a}_{ss'} v_\pi\left(s'\right)$$

Bellman Expectation Equation in Matrix Form

$$v_\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_\pi$$

$$v_\pi = \left(I - \gamma \mathcal{P}^\pi\right)^{-1} \mathcal{R}^\pi$$

Optimal Value Function

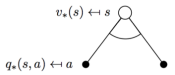
$$V_*(s) = \max_\pi V_\pi(s) \quad q_*(s,a) = \max_\pi q_\pi(s,a)$$

Optimal value specify the best performance in MDP

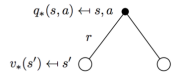
Optimal Policy

$$\pi_*(a \mid s) = \begin{cases} 1 \text{ if } a = \operatorname{argmax}_{a \in \mathcal{A}} q_*(s,a) \\ 0 \text{ otherwise} \end{cases}$$

Bellman Optimality Equation



$$V_*(s) = \max_a q_*(s,a)$$



$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

Model-based RL

Dynamic Programming (DP) (Full Backup)

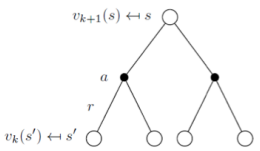
DP refers to a collection of algorithms that simplify a complicated problem by breaking it down into simpler sub-problems in a recursive manner.

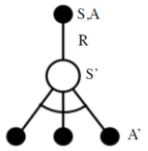
Policy Evaluation / Prediction

Problem: evaluate a given policy π

Solution: iterative application of Bellman expectation backup $v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_\pi$

Using synchronous backup: update $v_{k+1}(s)$ from $v_k(s')$





$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_k(s') \right)$$

Policy improvement

$$\pi' = \text{greedy}\left(v_\pi\right)$$

Policy Iteration / Control

Iterative policy evaluation + Greedy policy improvement. Converge to v_*, π_*.

Generalized Policy Iteration (GPI)

Any policy evaluation + Any policy improvement.

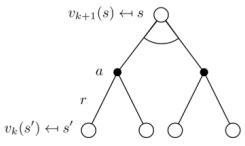
Converge to v_*, π_*.

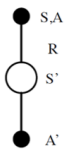
Value Iteration / Control

Problem: find optimal policy π

Solution: iterative application of Bellman expectation backup $v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_*$

Using synchronous backup: update $v_{k+1}(s)$ from $v_k(s')$





$$v_{k+1}(s) = \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_k(s') \right)$$

Asynchronous DP: (In-place DP; Prioritized sweeping; Real-time dynamic programming)

Model-free RL

Monte Carlo RL (MC)

MC Policy Evaluation / Prediction

Monte-Carlo policy evaluation uses empirical mean return instead of expected return.

First Visit MC Policy Evaluation, Every Visit MC Policy Evaluation. (in an episode)

$$N(s) \leftarrow N(s) + 1 \qquad S(s) \leftarrow S(s) + G_t$$

$$V(s) = S(s)/N(s)$$

Incremental MC Updates: Update V(s) incrementally after each episode. Useful in nonstationary problem

$$N\left(S_t\right) \leftarrow N\left(S_t\right) + 1$$

$$V\left(S_t\right) \leftarrow V\left(S_t\right) + \frac{1}{N\left(S_t\right)} \left(G_t - V\left(S_t\right) \right)$$

MC Policy Improvement / Control

Model Free GPI with MC Evaluation

Greedy policy improvement over V(s) needs model, over Q(s,a) is model free.

$$\pi'(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q(s,a)$$

MC Policy Iteration

MC policy evaluation, $Q = q_\pi$ + ϵ-greedy policy improvement. Converge to q_*, π_*

Greedy in the Limit with Infinite Exploration (GLIE) Policy Iteration

All state-action pairs are explored **infinite** times.

Sample k-th episode: $\left\{S_1, A_1, R_2, \ldots, S_T\right\} \sim \pi$

Evaluation:

$$N\left(S_t, A_t\right) \leftarrow N\left(S_t, A_t\right) + 1$$

$$Q\left(S_t, A_t\right) \leftarrow Q\left(S_t, A_t\right) + \frac{1}{N\left(S_t, A_t\right)} \left(G_t - Q\left(S_t, A_t\right) \right)$$

Improvement:

$$\epsilon \leftarrow 1/k$$

$$\pi \leftarrow \epsilon - \text{greedy}(Q)$$

Converge to $q_*(s,a)$

On policy: learn π from experience sampled from π

Off policy: learn π from experience sampled from μ

Temporal Difference (TD) (Sample Backup)

TD Prediction

$$V\left(S_t\right) \leftarrow V\left(S_t\right) + \alpha \left(R_{t+1} + \gamma V\left(S_{t+1}\right) - V\left(S_t\right) \right)$$

Learn online **every step**. Low variance, some bias.

$$\hat{\mathcal{P}}_{s,s'}^a = N(s,a,s')/N(s,a) \quad \hat{\mathcal{R}}_s^a = \sum r_s^a / N(s,a)$$

TD Control: Prediction + e-greedy

Sarsa Prediction

$$Q(S,A) \leftarrow Q(S,A) + \alpha \left(R + \gamma Q\left(S', A'\right) - Q(S,A) \right)$$

Q-Learning (Off-Policy) Prediction

$$Q(S,A) \leftarrow Q(S,A) + \alpha \left(R + \gamma \max_{a'} Q\left(S', a'\right) - Q(S,A) \right)$$