Recall

- Agent
- Environment
- State (s)
- Action (a)
- Reward (r)
- Policy (π)
- Value Function (V)

The objective of Reinforcement Learning is to learn an optimal policy that maximizes the expected cumulative reward over time.

$$\pi_\theta(a|s)$$

Recall

- Value Function (V)

It estimates how good a state or action is in terms of expected future rewards.

Types:

1. State Value Function (V(s)) – Expected return starting from state *s* and following a policy.
2. Action Value Function (Q(s, a)) – Expected return starting from state *s*, taking action *a*, and then following a policy.

Recall

- Value Function (V)

It estimates how good a state or action is in terms of expected future rewards.

Types:

1. State Value Function ($V(s)$) – Expected return starting from state *s* and following a policy.
2. Action Value Function ($Q(s, a)$) – Expected return starting from state *s*, taking action *a*, and then following a policy.

Recall

The main classes of RL algorithms used to estimate value functions and optimize policies

1. Dynamic Programming
2. Monte Carlo Estimation Methods
3. Temporal Difference Learning
4. Policy Gradient–Based Methods

Recall

Concepts of Grid World to Understand RL:

1. Environment – A grid of discrete states representing the world.
2. Agent – The entity that moves through the grid to achieve a goal.
3. State – Each cell or position in the grid.
4. Action – Possible moves (up, down, left, right) the agent can take.
5. Reward – Feedback given for entering a state (e.g., +1 for goal, -1 for trap).
6. Policy – Strategy defining which action to take in each state.
7. Value Function – Expected long-term reward from each state under a policy.
8. Goal State – Target location the agent tries to reach.
9. Episode – A complete sequence of states, actions, and rewards until the goal or termination.
10. Exploration vs. Exploitation – Balancing trying new moves vs. using known good

Recall

Concepts of Grid World to Understand RL:

1. Environment – A grid of discrete states representing the world.
2. Agent – The entity that moves through the grid to achieve a goal.
3. State – Each cell or position in the grid.
4. Action – Possible moves (up, down, left, right) the agent can take.
5. Reward – Feedback given for entering a state (e.g., +1 for goal, -1 for trap).
6. Policy – Strategy defining which action to take in each state.
7. Value Function – Expected long-term reward from each state under a policy.
8. Goal State – Target location the agent tries to reach.
9. Episode – A complete sequence of states, actions, and rewards until the goal or termination.
10. Exploration vs. Exploitation – Balancing trying new moves vs. using known good

Recall

Sequence of Action

- Set initial state — place the agent in a starting grid cell.
- Initialize policy — choose a starting policy (e.g., random actions).
- Explore by generating episodes — run episodes following the current policy to collect (state, action, reward) trajectories.
- Record returns — following each visited state, compute the episode return (sum of rewards).
- Estimate state values — update V(s) (e.g., using TD) from collected returns.
- Improve policy — derive a new policy that prefers actions leading to higher estimated values (policy improvement).
- Check for convergence — if policy (or value) changed little, stop; otherwise continue.
- Exploit learned policy — use the (near-)optimal policy to act greedily for best performance.
- (Optional) Continue exploration — occasionally explore to avoid local optima and further refine the policy.

Recall

```
+-----------+----------+-----------+
| (0,0) S   | (0,1) 0  | (0,2) -1T |
+-----------+----------+-----------+
| (1,0) 0   | (1,1) 0  | (1,2) 0   |
+-----------+----------+-----------+
| (2,0) 0   | (2,1) 0  | (2,2) +1G |
+-----------+----------+-----------+
```

- S (Start) → (0,0), reward = 0
- T (Trap) → (0,2), reward = -1
- G (Goal) → (2,2), reward = +1
- 0 → Neutral states, reward = 0

Recall

```
+----------+----------+----------+
| (0,0) S  | (0,1) 0  | (0,2) -1T |
+----------+----------+----------+
| (1,0) 0  | (1,1) 0  | (1,2) 0   |
+----------+----------+----------+
| (2,0) 0  | (2,1) 0  | (2,2) +1G |
+----------+----------+----------+
```

Step-by-step intuition:

- Observe the current state S_t.
- Take an action and move to the next state S_{t+1}.
- Compare what you expected $V(S_t)$ with what actually seems better $V(S_{t+1})$.
- Update your estimate of $V(S_t)$a little bit toward $V(S_{t+1})$,using a learning rate α.

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[V(S_{t+1}) - V(S_t)\Big]$$

Recall

```
+-----------+----------+-----------+
| (0,0) S   | (0,1) 0  | (0,2) -1T |
+-----------+----------+-----------+
| (1,0) 0   | (1,1) 0  | (1,2) 0   |
+-----------+----------+-----------+
| (2,0) 0   | (2,1) 0  | (2,2) +1G |
+-----------+----------+-----------+
```

1) Initial values (before episode)

```
+--------------------+--------------------+--------------------+
| (0,0) V=0  (S)     | (0,1) V=0          | (0,2) V=0  (T:-1)  |
+--------------------+--------------------+--------------------+
| (1,0) V=0          | (1,1) V=0          | (1,2) V=0          |
+--------------------+--------------------+--------------------+
| (2,0) V=0          | (2,1) V=0          | (2,2) V=+1  (G)    |
+--------------------+--------------------+--------------------+
```

1) Episode

$$(0,0) \rightarrow (0,1) \rightarrow (0,2) \rightarrow \textit{terminate (trap)}$$

Recall

```
+-----------+-----------+-----------+
| (0,0) S   | (0,1) 0   | (0,2) -1T |
+-----------+-----------+-----------+
| (1,0) 0   | (1,1) 0   | (1,2) 0   |
+-----------+-----------+-----------+
| (2,0) 0   | (2,1) 0   | (2,2) +1G |
+-----------+-----------+-----------+
```

1) Initial values (before episode)

```
+--------------------+--------------------+--------------------+
| (0,0) V=0  (S)     | (0,1) V=0          | (0,2) V=0  (T:-1)  |
+--------------------+--------------------+--------------------+
| (1,0) V=0          | (1,1) V=0          | (1,2) V=0          |
+--------------------+--------------------+--------------------+
| (2,0) V=0          | (2,1) V=0          | (2,2) V=+1  (G)    |
+--------------------+--------------------+--------------------+
```

1) Episode

$$(0,0) \rightarrow (0,1) \rightarrow (0,2) \rightarrow \textit{terminate (trap)}$$

1) Episode (given)

$$(0,0) \rightarrow (0,1) \rightarrow (0,2) \rightarrow \text{terminate (trap)}$$

We update online after each transition.

Step 1 — transition (0,0) → (0,1):

Current: $V(0,0) = 0,\ V(0,1) = 0.$

Apply TDL:

$$V(0,0) \leftarrow 0 + 0.5\big[V(0,1) - V(0,0)\big] = 0 + 0.5(0-0) = 0.$$

So $V(0,0)$ stays **0**.

Step 2 — transition (0,1) → (0,2) and termination:

On arrival at trap set terminal value $V(0,2) = -1$. Now update predecessor (0,1):

$$V(0,1) \leftarrow 0 + 0.5\big[V(0,2) - V(0,1)\big] = 0.5(-1-0) = -0.5.$$

Recall

```
+---------------------+---------------------+---------------------+
| (0,0) V=0  (S)      | (0,1) V=0           | (0,2) V=0  (T:-1)   |
+---------------------+---------------------+---------------------+
| (1,0) V=0           | (1,1) V=0           | (1,2) V=0           |
+---------------------+---------------------+---------------------+
| (2,0) V=0           | (2,1) V=0           | (2,2) V=+1  (G)     |
+---------------------+---------------------+---------------------+
```

2) State values after first episode

```
+---------------------+---------------------+---------------------+
| (0,0) V=0           | (0,1) V=-0.5        | (0,2) V=-1  (T)     |
+---------------------+---------------------+---------------------+
| (1,0) V=0           | (1,1) V=0           | (1,2) V=0           |
+---------------------+---------------------+---------------------+
| (2,0) V=0           | (2,1) V=0           | (2,2) V=+1  (G)     |
+---------------------+---------------------+---------------------+
```

1) Episode

$$(0,0) \rightarrow (0,1) \rightarrow (0,2) \rightarrow \textit{terminate (trap)}$$

3) Greedy policy arrows (choose neighbour with highest V)

Tie-breaking: when neighbors tie, I show one plausible choice (downwards preferred).

- **(0,0)** neighbors: (0,1) V=−0.5, (1,0) V=0 → prefer **down** to (1,0).
- **(0,1)** neighbors: (0,0)=0, (0,2)=−1, (1,1)=0 → best are (0,0) or (1,1) (tie). Pick **down** to (1,1) (illustrative).
- **(0,2)** trap — terminal (no outgoing).
- Other cells are 0 and have ties → shown as `?`.

```
+------------------------+------------------------+------------------------
| (0,0) V=0    ↓         | (0,1) V=-0.5  ↓ (tie)  | (0,2) V=-1  (T) [term]
+------------------------+------------------------+------------------------
| (1,0) V=0    ?         | (1,1) V=0     ?        | (1,2) V=0     ?
+------------------------+------------------------+------------------------
| (2,0) V=0    ?         | (2,1) V=0     ?        | (2,2) V=+1  (G)
+------------------------+------------------------+------------------------
```