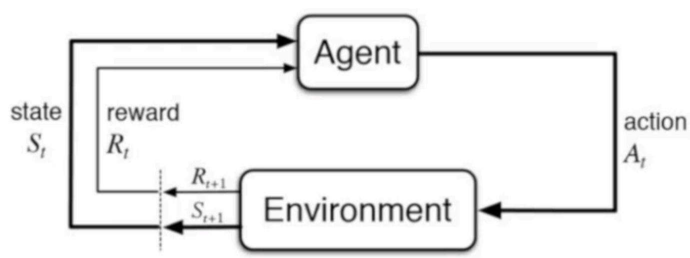
# Markov Decision Process

- Markov Chain→ Markov Reward Process (MRP)→ Markov Decision Processes (MDP)
- Policy evaluation in MDP
- Control in MDP: policy iteration and value iteration

### Markov Modules Define

- Markov Processes
- Markov Reward Processes(MRPs)
- Markov Decision Processes (MDPs)

Markov Decision Process (MDP)




1. Markov Decision Process can model a lot of real-world problem. It formally describes the framework of reinforcement learning
2. Under MDP, the environment is fully observable.
   1. Optimal control primarily deals with continuous MDPs
   2. Partially observable problems can be converted into MDPs

### Markov Property

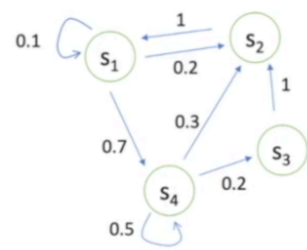
1. The history of states: $h_t = \{s_1, s_2, s_3, ..., s_t\}$
2. State $s_t$ is Markovian if and only if:

$$p(s_{t+1}|s_t) = p(s_{t+1}|h_t) \tag{1}$$
$$p(s_{t+1}|s_t, a_t) = p(s_{t+1}|h_t, a_t) \tag{2}$$

3. "The future is independent of the past given the present"

未来的转移和过去是独立的，只取决于现在

### Markov Process/Markov Chain




❶ State transition matrix P specifies $p(s_{t+1} = s'|s_t = s)$

$$P = \begin{bmatrix} P(s_1|s_1) & P(s_2|s_1) & \dots & P(s_N|s_1) \\ P(s_1|s_2) & P(s_2|s_2) & \dots & P(s_N|s_2) \\ \vdots & \vdots & \ddots & \vdots \\ P(s_1|s_N) & P(s_2|s_N) & \dots & P(s_N|s_N) \end{bmatrix}$$

### Markov Reward Processes(MRPs)

- Markov Chain + reward

  ❶ S is a (finite) set of states ($s \in S$)
  ❷ P is dynamics/transition model that specifies $P(S_{t+1} = s'|s_t = s)$
  ❸ R is a reward function $R(s_t = s) = \mathbb{E}[r_t|s_t = s]$
  ❹ Discount factor $\gamma \in [0, 1]$

- R can be a vector if finite
- Return and Value function
  - Horizon: 有限步数，可以无限
  - Return:

    Discounted sum of rewards from time step $t$ to horizon

    $$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \ldots + \gamma^{T-t-1} R_T$$

  - state value function Vt(s) for a MRP: for future value

    Expected return from $t$ in state $s$

    $$\begin{aligned} V_t(s) =& \mathbb{E}[G_t|s_t = s] \\ =& \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{T-t-1} R_T|s_t = s] \end{aligned}$$

  - Discount Factor γ
    - Avoids infinite returns in cyclic
    - Uncertainty in future
    - immediate rewards(human behavior)
    - undiscounted Markov reward processes (i.e. γ = 1, rewards all same)
    - γ = 0: Only care about the immediate reward

- bellman equation:

MRP value function satisfies the following **Bellman equation**:

$$V(s) = \underbrace{R(s)}_{\text{Immediate reward}} + \underbrace{\gamma \sum_{s' \in S} P(s'|s)V(s')}_{\text{Discounted sum of future reward}}$$

$$\begin{bmatrix} V(s_1) \\ V(s_2) \\ \vdots \\ V(s_N) \end{bmatrix} = \begin{bmatrix} R(s_1) \\ R(s_2) \\ \vdots \\ R(s_N) \end{bmatrix} + \gamma \begin{bmatrix} P(s_1|s_1) & P(s_2|s_1) & \dots & P(s_N|s_1) \\ P(s_1|s_2) & P(s_2|s_2) & \dots & P(s_N|s_2) \\ \vdots & \vdots & \ddots & \vdots \\ P(s_1|s_N) & P(s_2|s_N) & \dots & P(s_N|s_N) \end{bmatrix} \begin{bmatrix} V(s_1) \\ V(s_2) \\ \vdots \\ V(s_N) \end{bmatrix}$$

$$V = R + \gamma PV$$

  - Analytic solution for value of MRP: V = (I − γP)−1R
    1. Matrix inverse takes the complexity O(N**3**) for N states
    2. Only possible for a small MRPs
- Iterative methods for large MRPs:
  1. Dynamic Programming

     **Algorithm 2** Iterative algorithm to calculate MRP value function

     1: for all states $s \in S$, $V'(s) \leftarrow 0$, $V(s) \leftarrow \infty$
     2: **while** $||V - V'|| > \epsilon$ **do**
     3: $\quad V \leftarrow V'$
     4: $\quad$ For all states $s \in S$, $V'(s) = R(s) + \gamma \sum_{s' \in S} P(s'|s)V(s')$
     5: **end while**
     6: return $V'(s)$ for all $s \in S$

     - Bootstrapping

  2. Monte-Carlo evaluation

     **Algorithm 1** Monte Carlo simulation to calculate MRP value function

     1: $i \leftarrow 0, G_t \leftarrow 0$
     2: **while** $i \neq N$ **do**
     3: $\quad$ generate an episode, starting from state $s$ and time $t$
     4: $\quad$ Using the generated episode, calculate return $g = \sum_{i=t}^{H-1} \gamma^{i-t} r_i$
     5: $\quad G_t \leftarrow G_t + g, i \leftarrow i + 1$
     6: **end while**
     7: $V_t(s) \leftarrow G_t/N$

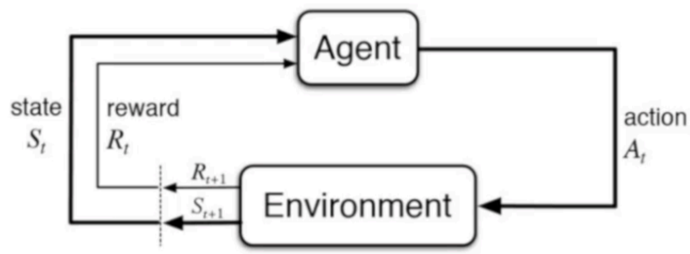
     1. For example: to calculate $V(s_4)$ we can generate a lot of trajectories then take the average of the returns:
        1. return for $s_4, s_5, s_6, s_7 : 0 + \frac{1}{2} \times 0 + \frac{1}{4} \times 0 + \frac{1}{8} \times 10 = 1.25$
        2. return for $s_4, s_3, s_2, s_1 : 0 + \frac{1}{2} \times 0 + \frac{1}{4} \times 0 + \frac{1}{8} \times 5 = 0.625$
        3. return $s_4, s_5, s_6, s_6 : = 0$
        4. more trajectories

3. Temporal-Difference learning

## Markov Decision Processes (MDPs)

Markov Reward Process with decisions. is a tuple.

### Markov Decision Process (MDP)




1. Markov Decision Process can model a lot of real-world problem. It formally describes the framework of reinforcement learning
2. Under MDP, the environment is fully observable.
    1. Optimal control primarily deals with continuous MDPs
    2. Partially observable problems can be converted into MDPs

- (S,A,P,R,γ):
    - S is a finite set of states
    - A is a finite set of actions
    - Pa is dynamics/transition model for each action P(st+1 =s' | st =s,at =a)
    - R is a reward function R(st = s,at = a) = E[rt | st = s,at = a]
    - Discount factor γ ∊ [0, 1]
- Policy π: stationary (time-independent)
    - At ~ π(a | s) = P(at = a | st = s) (any t>0)
    - $$P^\pi(s'|s) = \sum_{a \in A} \pi(a|s) P(s'|s,a)$$
      $$R^\pi(s) = \sum_{a \in A} \pi(a|s) R(s,a)$$
- action

## Difference between Policy Iteration and Value Iteration

- Policy iteration includes: policy evaluation + policy improvement, and the two are repeated iteratively until policy converges.
  - state-value function vπ(s)
  - action-value function qπ(s,a)

Bellman Expectation Equation for $V^\pi$ and $Q^\pi$

$$v^\pi(s) = \sum_{a \in A} \pi(a|s) q^\pi(s,a) \tag{8}$$

$$q^\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P(s'|s,a) v^\pi(s') \tag{9}$$

- Value iteration includes: finding optimal value function + one policy extraction. There is no repeat of the two because once the value function is optimal, then the policy out of it should also be optimal (i.e. converged).

https://github.com/cuhkrlcourse/RLexample/tree/master/MDP

## Prediction and Control in MDP

Table: Dynamic Programming Algorithms

| Problem | Bellman Equation | Algorithm |
|---|---|---|
| Prediction | Bellman Expectation Equation | Iterative Policy Evaluation |
| Control | Bellman Expectation Equation | Policy Iteration |
| Control | Bellman Optimality Equation | Value Iteration |