Reinforcement Learning

RL vs. Supervised Learning

1. Sequential decision making

   - return vs. risk tradeoff
   - long term future consequences
   - greedy is not optimal

2. No or little human supervision available

3. Learn from designed reward fn

4. Can collect more data iteratively

Markov Decision Process

1 2 3 4 5 6 7 8 9 10

S: set of states (all possible configurations)
A: set of actions (A = {L, R})
Psa dynamics/transitions
State transition prob dumb

At state s, when applying action a
the distr. of next s'
Psa(s') = Pr[s' | s, a]
Psa is a dumb, Psa ∈ IR^|S|

∑ Psa(s') = 1
s'∈S
Example: L action succeeds w. prob 0.9
\[ P_{\text{L}}(6) = 0.9 \]
\[ P_{\text{L}}(7) = 0.1 \]
\[ P_{\text{L}}(s') = 0 \quad \forall \quad s' \neq 6, 7 \]
R action succeeds w. prob. 0.8
\[ P_{\text{R}}(8) = 0.8 \]
\[ P_{\text{R}}(7) = 0.2 \]
\[ P_{\text{R}}(s') = 0 \quad \forall \quad s' \neq 7, 8 \]

Once you reach 6, robot stays there
\[ P_{\text{L}}(6) = 1 \]
\[ P_{\text{L}}(s') = 0 \quad \forall \quad s' \neq 6 \]
\[ P_{\text{R}}(6) = 1 \]
\[ P_{\text{R}}(s') = 0 \quad \forall \quad s' \neq 6 \]

Sequential decision making process
- So initial state (given)
- algo chooses action \( a_0 \in A \)
- environment step: \( S_1 \sim P_{S_0} a_0 \)
- algo sees \( S_1 \), takes action \( a_1 \)
- environment: \( S_2 \sim P_{S_1} a_1 \)

\[ S_0 = 4 \]
\[ a_0 = L \]
\[ S_1 \sim P_{4,R} \]
\[ S_1 = 5 \]
\[ a_1 = R \]

Reward function
\[ R : S \rightarrow \mathbb{R} \]
\[ R(6) = 1.0 \]
\[ R(5) = -0.1 \quad \forall \quad s \neq 6 \]
\[ S_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \text{ trajectory / episode} \]

Total payoff:
\[ R(s_0) + R(s_1) + \ldots + R(s_T) \]

Discounted total payoff

Discount factor: \( \gamma < 1 \)
\[ R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots + \gamma^T R(s_T) + \ldots \]

Suppose \( R(s) \in [-M, M] \)
\[ \Rightarrow \text{discounted total payoff} \leq M + \gamma M + \gamma^2 M + \ldots \]
\[ \frac{-M}{1-\gamma} \leq \frac{M}{1-\gamma} \]

finite horizon: stop process at time \( T \)
infinite horizon

Expected payoff
\[ \mathbb{E}[R(s_0) + \gamma R(s_1) + \ldots + \gamma^T R(s_T) + \ldots] \]

Reward can be \( R(s, a) \)

MDP \( (S, A, \{P_{sa}\}_{s \in S, a \in A}, \gamma, R) \)

goal: maximize expected discounted payoff

Markov property:
at any time \( t \), given \( S_t \), all future states \( S_{t+j} \)
don't depend on past states \( S_t \)
\[ S_0, a_0, s_1, a_1 \ldots S_{t-1}, a_{t-1} \]
\[ \Rightarrow \text{optimal action at time } t \text{ does not depend on past state and actions} \]
Best strategy can be expressed as a policy

\[ \pi: S \rightarrow A \] policy

\[ a_t = \pi(S_t) \] optimal policy

\[ \pi(S) = R \quad \text{if} \quad S \leq 6 \]
\[ \pi(S) = L \quad \text{if} \quad S > 7 \] randomized policy \( \pi(a|s) \)

\[ \exists \text{ deterministic optimum policy} \]

**Value function**

\[ V^\pi: S \rightarrow \mathbb{R} \]

\[ V^\pi(S) = \text{Expected payoff of executing policy } \pi \text{ starting from state } S \]

\[ S_0, a_0 \ldots - a_t = \pi(S_t) \]

\[ V^\pi(S) = 1E[R(S_0) + \gamma R(S_1) + \ldots + \gamma^t R(S_t) + \ldots \mid S_0 = S] \]

How do we compute \( V^\pi(S) \) \( S_0 = 6 \) \( S \)

\[ V^\pi(6) = 1 + \gamma + \gamma^2 + \ldots \]

\[ = \frac{1}{1-\gamma} \]

\[ V^\pi(6) = 1 + \gamma \left(1 + \gamma + \gamma^2 + \ldots\right) \]

\[ = 1 + \gamma V^\pi(6) \]

\[ V^\pi(6) = \frac{1}{1-\gamma} \]

\[ V^\pi(S) = 1E[R(S_0) + \gamma R(S_1) + \ldots \mid S_0 = S] \]
\[ V^\pi(s) = R(s) + \gamma \mathbb{E}\left[ R(s) + \gamma R(s_2) + \ldots \mid s_0 = s \right] \]
\[ = R(s) + \gamma \mathbb{E}\left[ R(s) + \gamma R(s_2) + \ldots \right] \]
\[ = R(s) + \gamma \mathbb{E}\left[ R(s_1) + \gamma R(s) + \ldots \right] \]
\[ = R(s) + \gamma \mathbb{E}\left[ R(s_1) + \gamma R(s_2) + \ldots \right] \]
\[ V^\pi(s) = R(s) + \gamma \sum_{s' \in S} P_{s, \pi(s)}(s') \cdot V^\pi(s') \]

**Bellman Equation**

- \( V^\pi(s) \) as variables
- \(|S| \) variables, \(|S| \) equations
- Solving system of linear eqn. \( \Rightarrow V^\pi(s) \forall s \)

\[
\begin{align*}
\text{given } \pi, P_{s,a} & \quad \rightarrow \quad V^\pi(s), \forall s \\
\text{Next: } V^*(s) & = \max_{\pi} V^\pi(s) \\
\pi^* & = \arg\max_{\pi} V^\pi(s) \\
\text{Markov property} & \quad \Rightarrow \exists \pi^* \text{ that does not depend on } s \\
\pi^*: s & \rightarrow A
\end{align*}
\]

**Bellman equation for \( V^* \)**

\[
\begin{align*}
V^*(s) &= \max_{\pi} V^\pi(s) \\
&= \max_{\pi} \left( R(s) + \gamma \sum_{s'} P_{s, \pi(s)}(s') \cdot V^\pi(s') \right) \\
&= R(s) + \max_{\pi} \left( \gamma \sum_{s'} P_{s, \pi(s)}(s') \cdot V^\pi(s') \right) \\
&= R(s) + \max_{\pi} \left( \gamma \sum_{s'} P_{s, \pi(s)}(s') \max_{\pi} V^\pi(s') \right)
\end{align*}
\]
\[ V^*(s) = R(s) + \gamma \max_{a} \sum_{s'} P_{s,a}(s') V^*(s') \]

the best expected payoff after taking action \( a \)

\[ V^* \in \mathbb{R}^{|S|} \]

\[ \begin{bmatrix} V^*(1) \\ \vdots \\ V^*(|S|) \end{bmatrix} \]

B operator

\[ V^*(s) = B(V^*) \]

Value iteration

- Initialize \( V \in \mathbb{R}^{|S|} \), \( V = 0 \) \( V(s) = 0 \ \forall \ s \)
- Loop
  \[ V = B(V) \]
  \[ V(s) = R(s) + \gamma \max_{a} \sum_{s'} P_{s,a}(s') V(s') \]

\[ \pi^*(s) = \arg\max_{a} \sum_{s' \in S} P_{s,a}(s') V^*(s') \]

Policy iteration

- initialize policy \( \pi \)
- Loop
  - let \( V = V^{\pi} \) (obtained by solving system of eqs)
  \[ \pi(s) = \arg\max_{a} \sum_{s'} P_{s,a}(s') V(s') \]

\[ V^*, \pi^* \text{ is stationary point of algorithm} \]

\[ V^* = V^{\pi^*} \]
Common: $V^*, \pi^*$ are stationary points of algos.

**Policy Iteration:**
- **Con:** Each iteration requires solving linear system, expensive for large state space.
- **Pro:** Converges in finite time exactly.
  - Almost [A]fst policies.

Value Iteration always has some approximation.

**Practice:** Discrete state space
- Value iteration often slightly better.
- Continuous controls: Policy iteration (or some combo), because max operation in Value iteration is hard to do exactly.