lions@epfl

EE-568: REINFORCEMENT LEARNING
LABORATORY FOR INFORMATION AND INFERENCE SYSTEMS
SPRING 2024

EPFL

INSTRUCTOR: VOLKAN CEVHER

SCRIBE: MUTIAN HE

DYNAMIC PROGRAMMING

Index terms: Markov decision process, Bellman equation, dynamic programming

1 Preliminary

1.1 Markov Chain

Markov chains, a.k.a. Markov processes, are fundamental constructs in reinforcement learning. A (time-homogeneous) Markov chain is defined as a sequence of random variables $\{X_0, X_1, \ldots\}$ that take values in a countable set of states, formally described by the triple $\langle S, P, \mu \rangle$, where S is the set of all possible states, $P \in R^{|S| \times |S|}$ is the transition matrix with entries $P_{ss'} = P(s' \mid s)$, and $\mu \in R^{|S|}$ is the initial state distribution such that $P[X_0 = s] = \mu_s$. The key trait of a Markov chain is Markov property, which states that the probability of transitioning to the next state depends only on the current state but not the history. Mathematically, this is expressed as:

$$P[X_{t+1} = j \mid X_t = i, X_{t-1}, \ldots, X_0] = P[X_{t+1} = j \mid X_t = i] = P_{ij}.$$

Some Markov chains enjoy the characteristics like

- Irreducibility: A Markov chain is irreducible if it is possible to reach any state from any other state.
- Aperiodicity: The period of a state x is defined as $d(x) := \gcd\{n \in N^+ : P^n(x, x) > 0)\}$. A Markov chain is aperiodic if every state has a period of one, preventing the chain from getting stuck in cycles and ensuring thorough mixing.

For a Markov chain that is irreducible and aperiodic, and with a finite state space (thus ergodic), there exists a unique stationary distribution d^*, characterized by the convergence property of the chain:

$$\lim_{t \to \infty} [P^t]_{ij} = d^*_j, \forall i, j.$$

This can be also represented by the equation $d^* = d^* P$, indicating that d^* is the left principal eigenvector of the transition matrix P.

The convergence to a unique stationary distribution enables the estimation of long-term averages through simulation, while the actual efficiency of probabilistic modeling and simulations depends on its *mixing time*.

1.2 Markov Decision Process

1.2.1 Definition

We formulate the reinforcement learning task based on the key concept of Markov Decision Process (MDP), which is extended from a Markov chain. An (infinite horizon) MDP is defined by the tuple $(S, A, P, r, \mu, \gamma)$, where:

- S is the set of all possible states.
- A is the set of all possible actions.
- P^a represents the transition matrix for action a, with entries $P^a_{ss'} = P(s' \mid s, a)$.
- $r : S \times A \to \mathbb{R}$ is the reward function, typically bounded within $[0, 1]$.

- μ is the initial state distribution.
- $\gamma \in (0, 1)$ is a discount factor.

Unlike a standard Markov chain, the transition of state depends not only the current state, but also the action to take at the current state, which is determined by the policy. By taking an action at the current state, a reward is produced.

In a Gridworld example, the states represent the agent's position in a grid, and actions correspond to movements in the four directions. The reward function assigns values based on the agent's actions, i.e. penalizing moves outside the grid and rewarding movements to specific target positions. The transition of the state is non-stochastic in this case, determined by the agent's action.

1.2.2 Performance Criteria

We formulate Reinforcement Learning under the framework of MDP. Particularly, the goal is to find the optimal policy that maximizes the "total" reward, aggregated under some performance criteria. For finite horizon problems, cumulative or average reward is considered, while for infinite horizon problems, cumulative or discounted reward is used. Particularly, in this lecture we mainly focus on the case of discounted infinite horizon MDPs, with the criteria defined as:

$$J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 \sim \mu, \pi\right].$$

The factor $\gamma \in (0, 1)$ discounts the future rewards to come to the present, and ensures that the total reward remains finite. Otherwise a $\gamma = 1$ may lead to infinite total reward, e.g. when the process is cyclic.

1.2.3 Policy

In MDP, a policy π defines the action to be taken at each time, which can be deterministic or randomized:

- Deterministic Policy: A deterministic policy maps states to actions directly. For stationary policies, $a_t = \pi(s_t)$, in which π does not change over time. For Markov policies, $a_t = \pi_t(s_t)$. History-dependent policies, on the other hand, consider the entire history up to time t.
- Randomized Policy: A randomized policy specifies a probability distribution over actions. For stationary policies, actions are sampled from a distribution $\pi(\cdot \mid s_t)$, and similarly for Markov and history-dependent policies.

For infinite horizon problems, stationary deterministic policies are sufficient to maximize the objective. In contrast, finite horizon problems require non-stationary deterministic Markov policies.

1.2.4 Value Functions

Value functions are essential functions to the performance. There are two primary types of value functions: the state-value function V and the state-action value function Q.

Definition 1. *The state-value function $V^\pi(s)$ represents the expected return starting from state s and following policy π thereafter:*

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, \pi\right].$$

Then suppose when we take a specific action:

Definition 2. *The state-action value function $Q^\pi(s, a)$, also known as the quality function, represents the expected return starting from state s, taking action a, and following policy π thereafter:*

$$Q^\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a, \pi\right].$$

Therefore, the relationship between V^π and Q^π can be expressed as follows:

$$Q^\pi(s,a) = r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^\pi(s'),$$
$$V^\pi(s) = \sum_a \pi(a \mid s) Q^\pi(s,a).$$

1.2.5 Optimal Value Functions

The objective in RL is to find an optimal policy π^* that maximizes the value functions, i.e. achieving the optimal value functions.

Definition 3. *The optimal value functions V^* and Q^* are the value functions under the optimal policy.*

$$V^*(s) = \max_\pi V^\pi(s),$$
$$Q^*(s,a) = \max_\pi Q^\pi(s,a).$$

Hence their relationships are similarly given by:

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^*(s'),$$
$$V^*(s) = \max_a Q^*(s,a).$$

This can be simply derived by:

$$\begin{aligned}
Q^\pi(s,a) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t) \mid s_0 = s, a_0 = a, \pi\right] \\
&= r(s,a) + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r(s_t,a_t) \mid s_0 = s, a_0 = a, \pi\right] \\
&= r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t,a_t) \mid s_0 = s, s_1 = s', a_0 = a, \pi\right] \\
&= r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t) \mid s_0 = s', \pi\right] \quad \text{(Markov assumption)} \\
&= r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^\pi(s').
\end{aligned}$$

2 Solving MDP

The ultimate goal in reinforcement learning is to find an optimal policy π^* such that:

$$V^{\pi^*}(s) = V^*(s) = \max_\pi V^\pi(s), \quad \forall s \in S.$$

We first consider the case when the model (i.e. transition dynamics P) is pre-known, known as model-based RL. The key questions in this context are:

- How to evaluate the current policy π, i.e., how to compute $V^\pi(s)$? (policy evaluation)
- Does the optimal policy π^* exist? (existence of optimal policy)
- (How) can we improve the current policy π to find π^*? (policy improvement)

2.1 Policy Evaluation

For each given policy π, we would like to evaluate the value functions V^π.

Definition 4. *Bellman Consistency Equation (BCE): Following the definition of the value function, there is*

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[r(s,a) + \gamma \sum_{s'} P(s' \mid s,a)V^\pi(s')\right].$$

Alternatively, we can write it in a more compact form:

Definition 5. *Matrix form of BCE:*

$$V^\pi = R^\pi + \gamma P^\pi V^\pi,$$

where $V^\pi \in \mathbb{R}^{|S|} : V^\pi_s := V^\pi(s)$ is a vector of state values, $R^\pi \in \mathbb{R}^{|S|} : R^\pi_s := \sum_{a\in A}\pi(a|s)r(s,a)$ is the reward vector, and $P^\pi \in \mathbb{R}^{|S|\times|S|} : P^\pi_{s,s'} := \sum_{a\in A}\pi(a|s)P(s'|s,a)$ is the transition matrix under policy π.

Given this matrix form of the Bellman consistency equation, the closed-form solution for V^π can be obtained as:

$$V^\pi = (I - \gamma P^\pi)^{-1}R^\pi.$$

For $\gamma \in (0,1)$, the matrix $I - \gamma P^\pi$ is always invertible, and there is always a unique and exact solution. However, due to the inversion of the matrix $I - \gamma P^\pi$, it has a cubic time complexity of $O(|S|^3 + |S|^2|A|)$, which is computationally feasible only for small state spaces. Therefore, in practice we may instead use iterative methods for policy evaluation.

Definition 6. *The Bellman expectation operator T^π is an affine operator defined as:*

$$T^\pi V := R^\pi + \gamma P^\pi V.$$

By comparing with BCE, we can find several important properties:

Lemma 2.1. *The value function V^π is a fixed point of this operator*

$$T^\pi V^\pi = V^\pi.$$

Lemma 2.2. *T^π is a γ-contraction mapping.*

$$\|T^\pi V - T^\pi V'\|_\infty \le \gamma\|V - V'\|_\infty.$$

Proof: Since P^π is a stochastic matrix, the sum of each row is 1. Hence, for any vector x,

$$\|P^\pi x\|_\infty \le \|x\|_\infty.$$

Therefore, for any two value functions V and V', we have

$$\begin{aligned}\|T^\pi V - T^\pi V'\|_\infty &= \gamma\|P^\pi(V - V')\|_\infty \\ &\le \gamma\|V - V'\|_\infty\end{aligned}$$

Lemma 2.3. *T^π is component-wise monotonic:*

$$V' \le V \Rightarrow T^\pi V' \le T^\pi V$$

Proof:

$$\|T^\pi V' - T^\pi V\|_\infty = \gamma\|P^\pi(V - V')\|_\infty \le \gamma\|V - V'\|_\infty$$

Hence when $V' \le V$, $T^\pi V' \le T^\pi V$ also holds.

Unless otherwise stated, we use the infinity or max norm $\|\cdot\|_\infty$ in the following content. Given the two lemmas above and based on the fixed point theorem, there is

Theorem 2.4. *V^π is the unique fixed point of T^π, and for $V_{t+1} = T^\pi V_t$, it holds that*

$$\lim_{t\to\infty}(T^\pi)^t V_0 = V^\pi$$

Therefore, we can start with arbitrary V_0, and evaluate the policy π by iteratively applying T^π to the V_t until it converges to V^π, which is what we want.

2.2 Existence of Optimal Policy

The next topic is whether the optimal policy exists, and if so, how can we determine if a policy is optimal. This is revealed by the following theorems.

Theorem 2.5. *For an infinite horizon MDP, there exists a stationary and deterministic policy π^* such that:*

$$V^{\pi^*}(s) = V^*(s), \quad Q^{\pi^*}(s,a) = Q^*(s,a).$$

An optimal policy can be derived from Q^* as:

$$\pi^*(s) = \arg\max_{a \in A} Q^*(s,a).$$

Proof: We start with an initial state-action-reward-next-state tuple $(s_0, a_0, r_0, s_1) = (s, a, r, s')$.

First, we define the "offset" policy $\tilde{\pi}$ such that $\tilde{\pi}(a_t = a \mid h_t) := \pi(a_{t+1} = a \mid (s_0, a_0) = (s,a), h_t)$, i.e. a policy that is identical to π but with a time shift of one step ahead. By the Markov property, we have:

$$\mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r(s_t, a_t) \mid (s_0, a_0, r_0, s_1) = (s, a, r, s'), \pi\right] = \gamma V^{\tilde{\pi}}(s').$$

Given $(s_0, a_0, r_0) = (s, a, r)$, the set $\{\tilde{\pi} \mid \Pi\}$ is equivalent to the set Π itself.

Then, we can find that the optimal value function from s_1 onward is independent of $(s_0, a_0, r_0) = (s, a, r)$:

$$\max_{\pi \in \Pi} \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r(s_t, a_t) \mid (s_0, a_0, r_0, s_1) = (s, a, r, s'), \pi\right] = \gamma \max_{\pi \in \Pi} V^{\tilde{\pi}}(s') = \gamma \max_{\pi \in \Pi} V^{\pi}(s') = \gamma V^*(s').$$

Next, let $\pi(s) = \arg\max_{a \in A} \max_{\pi' \in \Pi} Q^{\pi'}(s,a)$. We can find that this deterministic and randomized policy is optimal:

$$\begin{aligned}
V^*(s_0) &= \max_{\pi' \in \Pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, \pi\right] \\
&= \max_{\pi' \in \Pi} \mathbb{E}\left[r(s_0, a_0) + \sum_{t=1}^{\infty} \gamma^t r(s_t, a_t) \mid \pi\right] \\
&= \max_{\pi' \in \Pi} \mathbb{E}\left[r(s_0, a_0) + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r(s_t, a_t) \mid (s_0, a_0, r_0, s_1), \pi\right]\right] \\
&\leq \max_{\pi' \in \Pi} \mathbb{E}\left[r(s_0, a_0) + \gamma V^*(s_1)\right] \quad \text{(Condition above)} \\
&= \max_{a_0, \pi' \in \Pi} \mathbb{E}\left[Q^{\pi'}(s_0, a_0)\right] \\
&= \mathbb{E}\left[r(s_0, a_0) + \gamma V^*(s_1) \mid \pi\right] \quad \text{(Definition of } \pi\text{)}
\end{aligned}$$

Finally, $V^*(s_0) \leq \mathbb{E}\left[r(s_0, a_0) + V^*(s_1) \mid \pi\right] \leq \mathbb{E}\left[r(s_0, a_0) + \gamma r(s_1, a_1) + \gamma^2 V^*(s_1) \mid \pi\right] \leq \ldots \leq V^{\pi}(s_0)$, and the equality holds if π is optimal. Therefore, we conclude that $V^{\pi} = V^*$, thus the proposed π is indeed optimal.

2.3 Policy Improvement

We answer the last question by taking the inspiration from the Bellman expectation operator. Intuitively, the optimal value function shall choose the best action for each state, which is decided by the reward and foreseeable future rewards (i.e. the value functions) of other states. Following this idea, we introduce a condition of the value function:

Definition 7. *Bellman optimality condition: A value function $\tilde{V}$ may satisfy the following equation*

$$\tilde{V}(s) = \max_{a \in A}\left[r(s,a) + \gamma \sum_{s'} P(s' \mid s, a)\tilde{V}(s')\right].$$

Similarly, as for the action-value function the condition becomes:

$$\tilde{Q}(s,a) = r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) \max_{a'} \tilde{Q}(s',a').$$

It is straightforward to see that for $\tilde{\pi}(s) = \arg\max_a \left\{ r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) \tilde{V}(s') \right\}$, $\tilde{V}$ satisfies the Bellman Expectation Condition. That is to say, $\tilde{V}$ is the valid value function corresponding to the policy $\tilde{\pi}$. We will then prove that $\tilde{V} = V^*$, i.e. it is the optimal value function, by first introducing the corresponding operator.

Definition 8. *The Bellman (optimality) operator T is an operator defined as:*

$$(TV)(s) = \max_{a \in A} \left[r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V(s') \right].$$

Similar to T^π, we can find several important properties.

Lemma 2.6. *T is a γ-contraction mapping:*

$$\|TV - TV'\|_\infty \leq \gamma \|V - V'\|_\infty.$$

Proof:

$$\begin{aligned}
|(TV')(s) - (TV)(s)| &= \left| \max_{a \in A} \left[r(s,a) + \gamma \sum_{s'} P(s'|s,a) V'(s') \right] - \max_{a' \in A} \left[r(s,a') + \gamma \sum_{s'} P(s'|s,a') V(s') \right] \right| \\
&\leq \max_{a \in A} \left| \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) V'(s') \right) - \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) V(s') \right) \right| \\
&= \max_{a \in A} \left| \gamma \sum_{s'} P(s'|s,a) \left(V'(s') - V(s') \right) \right| \\
&\leq \max_{a \in A} \gamma \sum_{s'} P(s'|s,a) \left| V'(s') - V(s') \right| \\
&\leq \gamma \max_{a \in A} \sum_{s'} P(s'|s,a) \|V' - V\|_\infty \\
&= \gamma \|V' - V\|_\infty.
\end{aligned}$$

Lemma 2.7. *T is component-wise monotonic:*

$$V \leq V' \Rightarrow TV \leq TV'$$

Proof: For $a^* = \arg\max_a \{ r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V(s') \}$, $a'^* = \arg\max_a \{ r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V'(s') \}$

$$\begin{aligned}
(TV)(s) &= \max_a \left\{ r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V(s') \right\} \\
&= r(s,a^*) + \gamma \sum_{s'} p(s' \mid s,a^*) V(s') \\
&\leq r(s,a'^*) + \gamma \sum_{s'} p(s' \mid s,a'^*) V(s') \\
&\leq r(s,a'^*) + \gamma \sum_{s'} p(s' \mid s,a'^*) V'(s') \\
&= (TV')(s)
\end{aligned}$$

Particularly, $V' < V \Rightarrow TV' < TV$. Then, we can show that:

Theorem 2.8. *$V^* = \tilde{V}$ is the unique fixed point of T^π that satisfies the Bellman optimality condition, and for $V_{t+1} = TV_t$, it holds that*

$$\lim_{t \to \infty} T^t V_0 = V^*$$

Proof: Given the fixed point theorem, $\tilde{V} = T\tilde{V}$ is the unique fixed point of T, and

$$\lim_{t\to\infty} T^t V = \tilde{V}$$

Furthermore, with V^* being the optimal value function (which always exists according to Theorem 2.5), suppose $\tilde{V} < V^*$, by the monotonicity, $T\tilde{V} = \tilde{V} < TV^*$. While by repeatedly applying T,

$$\tilde{V} < \lim_{t\to\infty} T^t V^* = \tilde{V}.$$

Therefore, $\tilde{V} = V^*$, i.e. V^* is the unique fixed point of T for arbitrary V.

2.4 Value Iteration

2.4.1 Value Iteration Algorithm

Value Iteration (VI) is an iterative method used to compute the optimal value function V^* by repeatedly applying the Bellman optimality operator T. The algorithm can be outlined as follows:

- Start with an initial guess V_0 (e.g., $V_0(s) = 0$ for all s).
- For each iteration t, update the value function using the Bellman operator:

$$V_{t+1} = TV_t.$$

Then according to Theorem 2.8, V_t will converge to V^*. It should be noticed that any V_t prior to convergence may not be a valid value function, i.e. there may not be any π satisfying the value function. However, once V^* is obtained through this iterative process, the optimal policy π^* can be derived by selecting the maximizing action, namely the greedy policy:

$$\pi^*(s) = \arg\max_a \left\{ r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V(s') \right\}$$

Alternatively, we may run the value iteration on Q function instead, following the Bellman operator on the Q function:

$$Q_{t+1}(s,a) = r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) \max_{a'} Q_t(s',a').$$

Then we may extract the policy

$$\pi^*(s) = \arg\max_{a\in A} Q^*(s,a).$$

One of the significant advantages of Q-value iteration is that it does not require explicit knowledge of the transition probabilities $P(s' \mid s,a)$, namely under model-free scenarios.

2.4.2 Convergence

Theorem 2.9. *The value iteration algorithm attains a linear convergence rate, i.e.:*

$$\|V_t - V^*\|_\infty \le \gamma^t \|V_0 - V^*\|_\infty.$$

Proof:

$$\begin{aligned} \|V_t - V^*\|_\infty &= \|TV_{t-1} - TV^*\|_\infty \\ &\le \gamma \|V_{t-1} - V^*\|_\infty \\ &\le \cdots \\ &\le \gamma^t \|V_0 - V^*\|_\infty. \end{aligned}$$

Therefore, the number of iterations required to achieve an ϵ-accurate solution is $T = O(\log \epsilon^{-1})$.

2.5 Policy Iteration

While value iteration focuses on finding V^* first, policy iteration directly searches for the optimal policy π^*. This is achieved through a two-step process involving policy evaluation and policy improvement.

2.5.1 Policy Improvement Theorem

The policy improvement theorem states that, under policy π, if we can take an alternative action $\pi'(s)$ at s that improves the results according to Q^π, then this π' also improves V. More formally,

Theorem 2.10. *Policy Improvement Theorem: if a deterministic policy π' satisfies*

$$Q^\pi(s, \pi'(s)) \geq V^\pi(s) \quad \forall s \in S,$$

then

$$V^{\pi'}(s) \geq V^\pi(s) \quad \forall s \in S.$$

Proof:

$$\begin{aligned}
V^\pi(s) &\leq Q^\pi(s, \pi'(s)) \\
&= \mathbb{E}_{\pi'}\left[r(s, \pi'(s)) + \gamma V^\pi(s_1) \mid s_0 = s\right] \\
&\leq \mathbb{E}_{\pi'}\left[r_0 + \gamma Q^\pi(s_1, \pi'(s_1)) \mid s_0 = s\right] \\
&\leq \mathbb{E}_{\pi'}\left[r_0 + \gamma r_1 + \gamma^2 V^\pi(s_2) \mid s_0 = s\right] \\
&\leq \cdots \\
&\leq \mathbb{E}_{\pi'}\left[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \mid s_0 = s\right] \\
&= V^{\pi'}(s).
\end{aligned}$$

This theorem provides the foundation for iteratively improving the policy. By updating the policy to choose actions that maximize the action-value function, we can ensure that the new policy performs at least as well as the current policy.

2.5.2 Policy Iteration Algorithm

The Policy Improvement Theorem gives a natural way of improving the current policy by making a greedy update

$$\pi_{t+1}(s) = \arg\max_a Q^{\pi_t}(s, a),$$

in which case $Q^{\pi_t}(s, \pi_{t+1}(s)) \geq Q^{\pi_t}(s, \pi_t(s)) = V^\pi(s)$.

The Policy Iteration (PI) algorithm can be described in the following steps:

1. Start with an initial policy guess π_0.
2. For each iteration t:
   (a) **Policy Evaluation**: Compute the value function V_t using either iterative application of the Bellman operator
   $$V_t \leftarrow T^{\pi_t} V_t$$
   or the closed-form solution:
   $$V)t = (I - \gamma P^{\pi_t})^{-1} R^{\pi_t}.$$
   (b) **Policy Improvement**: Update the policy by choosing the action that maximizes the expected return:
   $$\pi_{t+1}(s) = \arg\max_{a \in A} \left[r(s, a) + \gamma \sum_{s'} P(s' \mid s, a) V^{\pi_t}(s') \right].$$

This process continues until the policy converges, as $V^{\pi_{t+1}} > V^{\pi_t}$ as long as the policy is suboptimal, otherwise the Bellman optimality condition is already satisfied.

2.5.3 Convergence

Although there are only $|S||A|$ different policies, the policy may hop here and there with same V, so it will not simply converge in at most $|S|(|A|-1)$ steps. Therefore, we need to look at the convergence of V to decide the actual number of iterations. In fact,

Theorem 2.11. *The policy iteration algorithm converges in* $T = O(\frac{|S|(|A|-1)}{1-\gamma})$ *steps.*

To prove that, the basic idea is to show that

1. V has linear convergence, similar to Value Iteration;
2. Every K steps, we can eliminate a suboptimal action, i.e. $\pi_t(s) = a, \forall t' > t + K, \pi'_t(s) \neq a$. This K leads to the $1-\gamma$ factor in the result.

Lemma 2.12. *Linear convergence:*

$$\|V_t - V^*\|_\infty \leq \gamma^t \|V_0 - V^*\|_\infty.$$

Proof: For a policy π and its greedy update π', first we can note that

$$V^\pi = T^\pi V^\pi \leq TV^\pi = T^{\pi'} V^\pi$$

, i.e. greedification may improve V.

Then we utilize the induction: Given $i \geq 1$ such that $V^\pi \leq TV^\pi \leq (T^{\pi'})^i V^\pi$, by applying $T_{\pi'}$ on both sides, $T^{\pi'} V^\pi \leq (T^{\pi'})^{i+1} V^\pi$ due to monotonicity of T^π. Hence $V^\pi \leq TV^\pi \leq (T^{\pi'})^{i+1} V^\pi$, or $V^\pi \leq TV^\pi \leq V^{\pi'}$ when $i \to \infty$.

This indicates the V monotonically increases. Also, starting from π_0, we have $TV^{\pi_0} \leq V^{\pi_1}$, applying T on both sides give $T^2 V^{\pi_0} \leq TV^{\pi_1} \leq V^{\pi_2}$. Repeating k times, we obtain $T^k V^{\pi_0} \leq V^{\pi_k}$.

With $T^k V^{\pi_0} \leq V^{\pi_k} \leq V^*$, there is $V^* - V^{\pi_k} \leq V^* - T^k V_0^\pi$. Hence

$$\|V^* - V^{\pi_k}\|_\infty \leq \|V^* - T^k V^{\pi_0}\|_\infty = \|T^k V^* - T^k V^{\pi_0}\|_\infty \leq \gamma^k \|V^* - V^{\pi_0}\|_\infty,$$

as T is γ-contractive. Hence V has linear convergence.

Lemma 2.13. *Strict progress: For any* $k \geq k^* = \lceil H_{\gamma,1} \rceil + 1$, $\exists s_0, \pi_k(s_0) \neq \pi_0(s_0)$, *where* $H_{\gamma,\epsilon} = \ln(\epsilon(1-\gamma))/\ln(\gamma)$

Proof: first recall that $V^{\pi'} = (I - \gamma P_{\pi'})^{-1} r_{\pi'}$, we can find

$$\begin{aligned} V^{\pi'} - V^\pi &= (I - \gamma P_{\pi'})^{-1} [r_{\pi'} - (I - \gamma P_{\pi'}) V^\pi] \\ &= (I - \gamma P_{\pi'})^{-1} [T_{\pi'} V^\pi - V^\pi]. \end{aligned}$$

To represent the "advantage" of π' over π, we use $g(\pi', \pi) = T_{\pi'} V^\pi - V^\pi$, then the difference of value $V^{\pi'} - V^\pi = (I - \gamma P_{\pi'})^{-1} g(\pi', \pi)$.

Note that $g(\pi, \pi^*) \leq 0$, and can be used as a metric to track the progress of the policy iteration. More, $-g(\pi_k, \pi^*)(s_0) < -g(\pi_0, \pi^*)(s_0)$ indicates a change in action at s_0. Then,

$$-g(\pi_k, \pi^*) = (I - \gamma P_{\pi_k})(V^* - V^{\pi_k}) = (V^* - V^{\pi_k}) - \gamma P_{\pi_k}(V^* - V^{\pi_k}) \leq V^* - V^{\pi_k},$$

as since $V^* - V^{\pi_k} \geq 0$ and P is a probability matrix. This relates $g(\pi, \pi^*)$ to the difference of value $V^* - V^\pi$. Then by using the linear convergence lemma that bounds the difference of V, and by taking the infinity norm on both sides

$$\begin{aligned} \|g(\pi_k, \pi^*)\|_\infty &\leq \|V^* - V^{\pi_k}\|_\infty \\ &\leq \gamma^k \|V^* - V^{\pi_0}\|_\infty = \gamma^k \|(I - \gamma P_{\pi_0})^{-1}(-g(\pi_0, \pi^*))\|_\infty \\ &\leq \frac{\gamma^k}{1-\gamma} \|g(\pi_0, \pi^*)\|_\infty \end{aligned}$$

The last step is because $(I - \gamma P_{\pi_0})^{-1} = \sum_{i\geq 0} \gamma^i P_{\pi_0}^i$ (von Neumann expansion, similar to the formula of the sum of geometric series). Then by triangle inequality

$$\|(I - \gamma P_{\pi_0})^{-1} x\|_\infty = \| \sum_{i\geq 0} \gamma^i P_{\pi_0}^i x\|_\infty \leq \sum_{i\geq 0} \gamma^i P_{\pi_0}^i \|x\|_\infty \leq \sum_{i\geq 0} \gamma^i \|x\|_\infty,$$

as P is a transition that will not increase the infinity norm.

Now, find the s_0 corresponding to the norm, i.e. $0 \leq -g(\pi_0, \pi^*)(s_0) = \|g(\pi_0, \pi^*)(s_0)\|_\infty$. Considering that $0 \leq -g(\pi_k, \pi^*)(s_0) \leq \|g(\pi_k, \pi^*)\|_\infty$, there is

$$-g(\pi_k, \pi^*)(s_0) \leq \|g(\pi_k, \pi^*)\|_\infty \leq \frac{\gamma^k}{1-\gamma}(-g(\pi_0, \pi^*)(s_0)).$$

For sufficiently large $k = \tilde{O}(1/(1-\gamma))$, there is $g(\pi_k, \pi^*)(s_0) < g(\pi_0, \pi^*)(s_0)$, i.e. a change of action, and the early action will not revive anymore. This leads to convergence theorem above.

2.6 Summary

Value Iteration (VI) and Policy Iteration (PI) are two fundamental methods for solving MDPs, broadly used in model-based reinforcement learning when P is known. The differences are summarized below.

Algorithm	Per iteration cost	Number of iterations	Output
Value Iteration (VI)	$O(\|S\|^2\|A\|)$	$T = O\left(\frac{\log(\epsilon(1-\gamma))}{\log \gamma}\right)$	V_T such that $\|V_T - V^*\| \leq \epsilon$
Policy Iteration (PI)	$O(\|S\|^3 + \|S\|^2\|A\|)$	$T = O\left(\frac{\|S\|(\|A\|-1)}{1-\gamma}\right)$	V^* and π^*

3 Model-free scenarios

3.1 Introduction

3.1.1 Challenges

As we transition from simple dynamic programming-based planning methods to more realistic scenarios of reinforcement learning (RL), we encounter several fundamental challenges that must be addressed to effectively solve large-scale Markov Decision Processes (MDPs).

1. PI and VI require full knowledge of the transition model P and the reward function r. This requirement can be a significant limitation in many real-world scenarios where the transition dynamics and reward structures are not known a priori. To overcome this challenge, we need sampling approaches that can learn these models from interactions with the environment.

2. The computation and memory cost associated with dynamic programming methods can be very expensive for large-scale MDP problems when there is enormous or even infinite (continuous) space of state and actions, which necessitates the development of non-tabular approaches to represent the state and action spaces and the corresponding functions (e.g. policy, value, quality, etc.).

3.1.2 Types of RL approaches

Reinforcement learning offers several methodologies to address the challenges posed by dynamic programming. The primary approaches can be categorized as follows, based on their goals:

- Value-based RL: To learn the optimal value functions, V^* or Q^*, from which the policy can be decided.
- Policy-based RL: To directly learn the optimal policy π^*.
- Model-based: To first decide the model P and r, and then do planning based on that.

Model-based methods make full use of "experiences" and can reason about model uncertainty. They are sample efficient for easy dynamics but may be complex to implement. While model-free methods are direct and simple, not affected by poor model estimation. However, they are not sample efficient, often requiring a large amount of data to learn effectively. In the following part of this section, we introduce several value-based approaches to cope with the two challenges above.

There are also several different scenarios to be considered:

- Online RL: The agent collects data by interacting with the environment. This approach requires balancing the tradeoff between exploitation (using known information to maximize reward) and exploration (gathering new information).
- Offline RL: We use only previously collected data. The data is static, and no online data collection occurs. This approach allows the agent to learn from a fixed dataset, which can be beneficial in scenarios where interacting with the environment is costly or impractical.
- On-policy RL algorithms learn based on data collected from the current policy. These algorithms are always online, continually updating the policy based on new experiences.
- Off-policy RL algorithms learn from data collected from other policies. These algorithms can be either online or offline, providing flexibility in how data is gathered and used for learning.

3.1.3 Representation Learning

Large or continuous state and action spaces makes it infeasible to completely represent functions including V, Q, or π in a tabular way, e.g. by writing down all $Q(s, a)$ in a table. Instead, it becomes essential to approximate those functions by some other (possibly parameterized) functions, such as

- Linear Function Approximation (LFA): The value function can be represented by a standard linear regression

  $$V_\theta(s) = \phi(s)^T \theta$$

  where ϕ is the fixed feature vector and θ is the weight/parameter vector. Particularly, as in standard kernel regression, ϕ can be based on a kernel function, or even neural tanget kernels.
- Nonlinear Function Approximation: We may parameterize the function by non-linear models, such as modern deep neural networks.

Particularly, in a d-dimensional continuous state space, an alternative way is to discretize the state space and represent the function in a tabular way, in which each cell is represented independently. However, this leads to an exponential growth of table size w.r.t. d. Instead, alternative representations above allow for efficient representation and computation. Particularly, LFAs can exploit similarities between states, treating similar states in a correlated manner.

In more complicated applications like robotics, linear function approximations are not expressive enough. For instance, in MuJoCo, a three-layer neural network might be used to parameterize the value function. The input to the network is typically a vector containing the positions and velocities of the system's joints.

3.2 Monte Carlo Approach

Monte Carlo methods in reinforcement learning involve using sampling to estimate value functions V^π and optimize policies when the model of the environment is not fully known.

3.2.1 Objective

Consider a state distribution $\rho \in \Delta_S$, for which we take the initial state distribution $\rho = \mu$. We typically define the task of estimating V^π as solving the optimization problem

$$\min_V L(V)$$

where the loss or objective function

$$L(V) = \frac{1}{2}\|V^\pi - V\|_\rho^2$$

is a ρ-weighted norm loss $\|v\|_\rho^2 = v^T \cdot \text{diag}(\rho) \cdot v$. It is a metric if ρ is non-zero everywhere. In this case, the gradient is simply $\nabla L(V) = \text{diag}(\rho)(V - V^\pi)$, which is not directly computable as we don't know V^π, and has to be solved in a stochastic or Monte Carlo way.

3.2.2 Algorithm

We use Stochastic Gradient Descent (SGD) to solve the optimization task where the gradient is not directly tractable, by taking samples from the environment. The key ingredients of the algorithm are:

- An unbiased, bounded estimate g_t of the gradient of the loss function, i.e., $\mathbb{E}[g_t] = \nabla L(V)$.
- A step-size (learning rate) η_t.
- A simple iteration invariant: $V_{t+1} = V_t - \eta_t g_t$.

Having a gradient estimate is the most important prerequisite for SGD. As for estimating V^π, we take a random vector $G^\pi \in \mathbb{R}^{|S|}$ with entries

$$G^\pi(s_0) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \quad \text{with } s_0 \sim \rho, \text{ given } \pi.$$

This can be interpreted as follow: starting from an s_0 sampled from ρ, the agent walks through a trajectory of states and actions using π. If we record the total reward in this trajectory, it becomes $G^\pi(s_0)$.

By definition, $\mathbb{E}[G^\pi(s_0)] = V^\pi(s_0)$. Therefore, G^π is an unbiased estimator of V^π. Let $e_s \in \{0,1\}^{|S|}$ be the vector such that the s-th entry equals 1, and is zero elsewhere. Then for any V, $g_t = (V(s_0) - \mathbb{E}[G^\pi(s_0)])e_{s_0}$ is an unbiased gradient estimator:

$$\begin{aligned}\mathbb{E}_{s_0\sim\rho}[(V(s_0) - \mathbb{E}[G^\pi(s_0)])e_{s_0}] &= \mathbb{E}_{s_0\sim\rho}[(V(s_0) - V^\pi(s_0))e_{s_0}] \\ &= \text{diag}(\rho)(V - V^\pi) \\ &= \nabla L(V).\end{aligned}$$

Therefore, we can use SGD updates as follows. We first s ample an initial state $s_0 \sim \rho$ and sample a trajectory or episode $\{s_0, a_0, r_0, s_1, \ldots\}$, and then update the value function estimate by

$$V_{t+1}(s_0) = V_t(s_0) - \eta_t(V_t(s_0) - G^\pi(s_0)),$$

where the step-size η_t must be chosen appropriately.

In practice, the algorithm is implemented following the pseudo code below. A difference should be noticed: in the second loop, we update V^π of all states in a single step altogether. In this way, the correlation between value updates at different states are ignored.

Algorithm 1 Monte Carlo Algorithm

```
for t = 1, ..., T do
    Collect an episode τ = {s_0, a_0, r_0, s_1, ..., s_T, a_T, r_T} generated following π.
    for each state s_h do
        Compute return G^π(s_t) = r_t + γ r_{t+1} + ···
        Update V_{t+1}(s_h) ← V_t(s_h) + η_t(G^π(s_h) − V_t(s_h))
    end for
end for
```

3.2.3 Convergence Analysis

Note that for $r \in [0, 1]$, g_t has a bounded variance:

$$\mathbb{E}[\|g_t\|^2] = (G^\pi(s_0) - V^\pi(s_0))^2 \leq \frac{1}{(1-\gamma)^2}.$$

Furthermore, the loss function $L(V)$ is $(\min_s \rho(s))$-strongly convex as the Hessian $\nabla^2 L(v) = \text{diag}(\rho)$. By plugging into the SGD bound for strongly convex smooth functions with smoothness $L = 1$, gradient variance $\sigma^2 = \frac{1}{(1-\gamma)^2}$, strong convexity $\min_s \rho(s)$, and $\eta_t = \frac{1}{\mu t}$, we obtain the following guarantee:

Theorem 3.1. *Convergence of Monte Carlo estimation of the value function:*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|V^\pi - V_t\|_\rho^2] \leq \frac{2|S|\log T}{T \min_s \rho(s)(1-\gamma)^2}.$$

By Jensen's inequality, the above guarantee holds true for the average iterate $\bar{V}_T = \frac{1}{T}\sum_{t=1}^{T} V_t$ as well.

The situation is nevertheless more complicated if some $\rho(s) = 0$.

As for LFA, we may further look into the converged point in the misspecified scenario, when V^π can not be identified by a linear model, i.e.

$$\min_\theta \|\Phi\theta - V^\pi\|_\rho = \epsilon_{\text{approx}} > 0.$$

Starting from the stationarity of the solution, as

$$\nabla_\theta L(\theta^*_{\text{MC}}) = 0$$

or

$$\Phi^T \text{diag}(\rho)(V_{\theta^*_{\text{MC}}} - V^\pi) = 0,$$

there is

$$\Phi^T \text{diag}(\rho)\Phi\theta^*_{\text{MC}} = \Phi^T \text{diag}(\rho)V^\pi,$$

which means that θ^*_{MC} is the projection of V^π in the feature span.

3.2.4 Extension: LFA

We can parameterize the value function using linear function approximation, i.e., $V_\theta = \Phi\theta$ with $\theta \in \mathbb{R}^d$, $\Phi \in \mathbb{R}^{|S|\times d}$, in which typically $d \ll |S|$. In this case, the optimization problem becomes:

$$\min_\theta L(\theta)$$

with

$$L(\theta) = \frac{1}{2}\|V^\pi - \Phi\theta\|_\rho^2.$$

Then, the gradient of the loss becomes $\nabla L(\theta) = \Phi^T \text{diag}(\rho)(\Phi\theta - V^\pi)$. However, we can still solve the problem in a similar way: Let $s \sim \rho$ and let $\phi(s)$ denote the sth row of Φ. Then, the SGD updates are as follows:

$$\theta_{t+1} = \theta_t - \eta_t \phi(s)((\Phi\theta_t)(s) - G^\pi(s)),$$

where $\phi(s)((\Phi\theta_t)(s) - G^\pi(s))$ is an unbiased gradient estimate, and η_t has to be chosen appropriately.

By using the SGD convergence theorem on L-smooth functions, we can obtain a similar convergence bound.

Theorem 3.2. *Let θ^* be the minimizer of the LFA loss function and assume realizability, i.e., $V_{\theta^*} = V^\pi$. If we run Monte Carlo for T iterations with step size $\eta_t = \sqrt{\frac{d(1-\gamma)}{t}}$, then it holds that*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|V_{\theta_t}(s) - V^\pi(s)\|_\rho^2] \leq \frac{2\sqrt{d\log T}}{(1-\gamma)\sqrt{T}}.$$

By Jensen's inequality, the above guarantee holds true for the average iterate $\bar{V}_T = \frac{1}{T}\sum_{t=1}^{T} V_{\theta_t}$ as well.

3.2.5 Limitations

Monte Carlo methods require updates only after reaching the end of an episode, which is inefficient for long episodes. On the other hand, for infinite horizon problems, we cannot compute $G^\pi(s_0) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$. We need to artificially introduce a finite horizon truncating the trajectories. The truncation introduces bias, making the above analysis invalid. These problems can be overcome by the temporal difference method, which will be discussed next.

3.3 Temporal Difference learning

Unlike Monte Carlo methods above, Temporal Difference (TD) learning updates the value function estimates based on incomplete episodes, making it suitable for non-terminating environments.

3.3.1 Overview

Consider the case when we parameterize $V = V_\theta$ (e.g. by LFA), instead of directly solving V^π, we leverage the fact that there is a unique fixed point $T^\pi V^\pi = V^\pi$, and estimate $V^\pi$by solving the following problem

$$\min_{\theta \in \mathbb{R}^d} L(\theta),$$

where the objective function $L(\theta)$ is defined as:

$$L(\theta) = \frac{1}{2} \|V_\theta - T^\pi V_\theta\|_\rho^2.$$

This objective measures the violation of the fixed-point condition.

Unlike MC where $\rho = \mu$ is the initial distribution, we take ρ to be the limit distribution that represents the "final" distribution after infinite steps, defined as:

$$\rho(s) = \lim_{t \to \infty} P^\pi[s_t = s].$$

Then, the gradient of the loss function $L(\theta)$ can be written as:

$$\begin{aligned}
\nabla_\theta L(\theta) &= \nabla_\theta (V_\theta - T^\pi V_\theta) \operatorname{diag}(\rho) (V_\theta - T^\pi V_\theta) \\
&= \sum_{s \in S} \rho(s) \left(V_\theta(s) - \sum_{a \in A} \pi(a|s) \left(r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[V_\theta(s')] \right) \right) \left(\nabla_\theta V_\theta(s) - \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[\nabla_\theta V_\theta(s')] \right)
\end{aligned}$$

3.3.2 Gradient estimation

In a model-free scenario, we have to use samples to estimate the gradient above. Intuitively, we can sample $S \sim \rho$, $A \sim \pi(\cdot|S)$, and $S' \sim P(\cdot|S, A)$, and propose the estimator:

$$g_t = \left(V_\theta(S) - r(S, A) - \gamma V_\theta(S') \right) \left(\nabla_\theta V_\theta(S) - \gamma \nabla_\theta V_\theta(S') \right).$$

In the formula of $\nabla_\theta L(\theta)$, the terms of $s' \sim P(\cdot|s, a)$ appear twice independently. However, unless we are under a rather unrealistic case (e.g. in a simulator), when we are at state S, we can only sample the next state S' once. If we use a single S' as the sample to estimate the whole formula as in g_t, the estimator becomes biased because the expectation does not distribute over products:

$$\mathbb{E}_{s' \sim P(\cdot|S,A)}[V_\theta(S') \nabla_\theta V_\theta(S')] \neq \mathbb{E}_{s' \sim P(\cdot|S,A)}[V_\theta(S')] \mathbb{E}_{s' \sim P(\cdot|S,A)}[\nabla_\theta V_\theta(S')].$$

Therefore, we have to drop the sampling in either term. In TD, we judiciously ignore the sampling in the second term depending on θ, and define:

$$g_t = \left(V_\theta(S) - r(S, A) - \gamma V_\theta(S') \right) \nabla_\theta V_\theta(S).$$

Note that the first term is actually the difference between the estimated value at the current state and the one estimated by looking ahead for one step. This explains the name of "temporal difference", and a gradient descent based on this

estimator pushes the parameter to reduce the difference. Actually, g_t is an unbiased estimator for the partial computation of the TD gradient:

$$F(\theta) = \sum_{s\in S} \rho(s) \left(V_\theta(s) - \sum_{a\in A} \pi(a|s) \left(r(s,a) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)}[V_\theta(s')] \right) \right) \nabla_\theta V_\theta(s).$$

Although $F(\theta)$ is not the true gradient (i.e. the steepest descent direction), it can be shown that $F(\theta)$ is a descent direction. Specifically, it holds that:

$$\langle F(\theta), \theta^* - \theta \rangle \geq (1-\gamma) \|V_\theta - V_{\theta^*}\|_\rho^2,$$

where θ^* is the optimal parameter vector. This means that the expected direction of the update forms an acute angle with the vector pointing to θ^*. Additionally, the descent will not go too far as $F(\theta)$ satisfies:

$$\|F(\theta)\|^2 \leq 2\|V_\theta - V_{\theta^*}\|_\rho.$$

3.3.3 Algorithm

The algorithm to estimate V^π under the TD update rule is shown below.

Algorithm 2 TD Learning / TD(0)

1: **for** $t = 1$ to T **do**
2: **for** each step of an episode $\tau = \{s_0, a_0, r_0, s_1, \ldots, s_T, a_T, r_T\}$ following π **do**
3: Compute update direction $g_t = (V(s_t) - r_t - \gamma V(s_{t+1})) \nabla_\theta V_{\theta_t}(s_t)$
4: Update $\theta_{t+1} \leftarrow \theta_t - \eta_t g_t$
5: **end for**
6: **end for**

Similar to Monte Carlo methods, TD learning learns directly from episodes of experiences without needing the full MDP model. However, TD learning can learn from incomplete episodes, making it applicable to non-terminating environments, and enjoys lower variance. The above implementation does not sample from ρ, but it has been proven that this only introduces a small additive bias in the convergence guarantee. In addition to this TD(0) algorithm that looks ahead of 1 step, there are also more advanced methods of to look ahead for k-steps. Particularly, the TD(∞) algorithm is the Monte Carlo algorithm above.

In a tabular case, we can notice that the update rule is actually $V(s_t) \leftarrow (1-\eta_t)V(s_t) + \eta_t(r_t + \gamma V(s_{t+1}))$. Therefore, TD(0) can be also interpreted as local updates of V using the Bellman operator in a moving average manner.

3.3.4 Convergence Analysis

Running TD with a step-size $\eta_t = \frac{1-\gamma}{4\sqrt{t}}$, the following finite time bound holds [1]:

Theorem 3.3. *Convergence of the Temporal Difference estimation of the value function:*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|V_{\theta_t} - V_{\theta^*}\|_\rho^2] \leq \frac{9d\log T}{2(1-\gamma)^2\sqrt{T}},$$

where θ^* is the minimizer of the loss function.

This result shows that while TD learning has slightly worse performance than Monte Carlo (MC) methods, it is much easier to implement.

As for the misspecified case, note that at the stationary point

$$\nabla_\theta F(\theta) = \Phi^T \mathrm{diag}(\rho)(\Phi\theta^*_{\mathrm{TD}} - T^\pi \Phi\theta^*_{\mathrm{TD}}) = 0,$$

which implies:

$$\Phi^T \mathrm{diag}(\rho)\Phi\theta^*_{\mathrm{TD}} = \Phi^T \mathrm{diag}(\rho) T^\pi \Phi\theta^*_{\mathrm{TD}}.$$

or

$$\theta^*_{\mathrm{TD}} = (\Phi^T \mathrm{diag}(\rho)\Phi)^{-1}\Phi^T \mathrm{diag}(\rho) T^\pi \Phi\theta^*_{\mathrm{TD}}.$$

Hence, θ^*_{TD} is not directly related to V^π but is the fixed point of a projected Bellman equation, and may not be equal to the Monte Carlo solution. Nevertheless, previous research [2] has shown that:

$$\|V_{\theta^*_{\text{TD}}} - V^\pi\|_\rho \leq \frac{1}{\sqrt{1-\gamma^2}} \|\Phi\theta^*_{\text{MC}} - V^\pi\|_\rho.$$

That is to say, approximation error of TD is bounded but TD may suffer from a larger error, with an inflation factor of $\frac{1}{\sqrt{1-\gamma^2}}$ in the bound, compared to the Monte Carlo approach above.

3.4 Quality Estimation

3.4.1 SARSA algorithm

Instead of estimating the V^π function, we may also turn to Q^π function, which simplifies the process to derive the policy from the estimated values for the controling process. A well-known approach is the State-Action-Reward-State-Action (SARSA) algorithm, which solves the following optimization problem:

$$\min_Q L(Q) := \frac{1}{2}\|Q - T^\pi Q\|^2_\rho,$$

where T^π is the Bellman operator for Q-value functions, defined as:

$$(T^\pi Q)(s,a) = r(s,a) + \gamma \sum_{s',a'} P(s'|s,a)\pi(a'|s')Q(s',a').$$

This is essentially TD(0) but applied to the Q-function, and ρ also represents the stationary distribution. It can be solved by SGD in a similar way with similar convergence analysis, as shown below:

Algorithm 3 SARSA

for $t = 1$ to T **do**
  for each step of an episode $\tau = \{s_0, a_0, r_0, s_1, \ldots, s_T, a_T, r_T\}$ following π **do**
    Update $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
  end for
end for

3.4.2 Q-learning

In the methods above, we evaluate the Q or V function under the current policy, while applying the policy for the control process to produce new samples of trajectories is under a separate phase. While we may also consider the policy from the current Q-value estimation, though it can be different from the policy we use to produce the sample, i.e. being off-policy. This leads to the Q-learning algorithm that mix the control and learning, and directly learn the optimal Q^*:

Algorithm 4 Q-Learning

for $t = 1$ to T **do**
  for each step of an episode $\tau = \{s_0, a_0, r_0, s_1, \ldots, s_T, a_T, r_T\}$ **do**
    Update $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t(r_t + \gamma \max_{b \in A} Q(s_{t+1}, b) - Q(s_t, a_t))$
  end for
end for

4 Summary

In summary, we may use policy iteration to solve the MDP problem, with or without the model, as shown below:

Algorithm 5 Policy Iteration (PI)

for each iteration t **do**

1. **Policy Evaluation:** Compute V^{π_t}:
   - **Option 1:** Iteratively apply policy value iteration, $V_t \leftarrow T^{\pi_t} V_t$, until convergence (Exact).
   - **Option 2:** Use the closed-form solution: $V^{\pi_t} = (I - \gamma P^{\pi_t})^{-1} R^{\pi_t}$ (Exact).
   - **Option 3:** Approximate V^{π_t} with Monte Carlo (Inexact).
   - **Option 4:** Approximate V^{π_t} with Temporal Differences (Inexact).
2. **Policy Improvement:** Update the current policy π_t by the greedy policy:

$$\pi_{t+1}(s) = \arg\max_{a \in A} \left[r(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi_t}(s') \right].$$

end for

The advantage of Options 3 and 4 is that they do not require knowledge of the transition probabilities P. However, the value functions computed using these methods are inexact. This approach to policy iteration with inexact value functions is also known as Least Squares Policy Iteration (LSPI). Alternatively, we may use SARSA or Q-learning instead, which simplifies the policy improvement stage.

As for this algorithm, when we use inexact approximation of values, what we really care about is not the error of evaluating V^{π}, but the final error towards to the optimal function V^{*}. Fortunately, it can be also bounded as follows [3]:

Theorem 4.1. *Convergence of Policy Iteration with approximated value function:*

$$\sum_{s \in S} \rho(s)(V^{*}(s) - V^{\pi_k}(s)) \leq O\left(\frac{1}{(1-\gamma)^2} \max_{1 \leq j \leq k} \|V_{\theta_j} - V^{\pi_j}\|_{\rho} \right) + O(\gamma^k).$$

Therefore, the model-free approaches can be reliably used to obtain the optimal value function.

References

[1] Bhandari, Jalaj, Daniel Russo, and Raghav Singal. "A finite time analysis of temporal difference learning with linear function approximation." Conference on learning theory. PMLR, 2018.

[2] Tsitsiklis, John, and Benjamin Van Roy. "Analysis of temporal-diffference learning with function approximation." Advances in neural information processing systems 9 (1996).

[3] Scherrer, Bruno, et al. "Approximate modified policy iteration." arXiv preprint arXiv:1205.3054 (2012).