Markov Decision Processes and Reinforcement Learning

Day 3 — Computing Optimal Value Functions and Policies

Luiz Felipe Martins

Department of Mathematics and Statistics
Cleveland State University

January 28, 2026

Outline

Optimal Bellman Equation

- The **optimal Bellman qquation is**:

$$V^\star(s) = \max_{a \in \mathscr{A}(s)} \sum_{s' \in \mathscr{S}} p(s' \mid s, a)[r(s, s', a) + \gamma V^\star(s')]$$

- **Verification Theorem**: If $V^\star$ is a solution of the optimal Bellman equation, the *we know it is the optimal value function*
- In this case, an optimal deterministic policy can be obtained by **one-step lookahead**:

$$\pi^\star(s) = \arg \max_{a \in \mathscr{A}(s)} \sum_{s' \in \mathscr{S}} p(s' \mid s, a)[r(s, s', a) + \gamma V^\star(s')]$$

Value iteration algorithm

- Let $V_0(s)$ be initialized arbitrarily for $s \in \mathscr{S}$.
- Iterate:

$$V_{n+1}(s) = \max_{a \in \mathscr{A}(s)} \left\{ \sum_{s' \in \mathscr{S}} p(s, s' \mid a) \left[r(s, s', a) + \gamma V_n(s') \right] \right\},$$

except that we let $V_{n+1}(s) = 0$ if s is a terminal state.

Policy Iteration Algorithm

- Let π_0 be a randomized initial policy, with corresponding value function $V_0^{\pi}(\cdot)$.
- Iterate for $n = 0, 1, 2, \ldots$:
  - Compute a new policy:

$$\pi_{n+1}(s) = \arg \max_{a \in \mathscr{A}(s)} \sum_{s' \in \mathscr{S}} p(s' \mid s, a) \left[r(s, s', a) + \gamma V_n^{\pi}(s') \right]$$

  - Compute the value function of π_{n+1}, $V^{\pi_{n+1}}(\cdot)$
- Stopping criteria:
  - $\pi_{n+1}(s) = \pi_n(s)$ for all $s \in \mathscr{S}$
  - $\max_{s \in \mathscr{S}} |V^{\pi_{n+1}}(s) - V^{\pi_n}(s)| < \epsilon_{\text{tol}}$
- A policy computed as above is called **greedy**.

Policy Iteration example

n	$\pi_n(0)$	$\pi_n(1)$	$V_n(0)$	$V_n(1)$
0	b	a	33.121	34.515
1	a	b	71.250	63.571
2	a	b	71.250	63.571

- In general, policy iteration converges much faster than value iteration.
- In fact, for a finite MDP, policy improvement will “converge” in a finite number of steps, since the total number of deterministic policies is finite.
- This assumes that:
  - Transition probabilities are known.
  - The whole state is observed.
  - Rewards are a discrete subset of $\mathbb{R}$.
  - $V_n(s)$ is computed exactly.

Approximate Policy Iteration

- Policy iteration starts with a policies π_n that are far from optimal.
- It is wasteful to compute $V^{\pi_n}(s)$ exactly in the beginning.
- Instead, we compute only an *approximation* $\tilde{V}^{\pi_n}(s)$ in each step. For example, use a Jacobi or Gauss-Seidel iteration with a small number of relaxations.
- We don't even need a good approximation!
- Stopping condition:
  - Policy stabilizes: $\pi_{n+1}(s) = \pi_n(s)$ for all states s *and*
  - Value function estimates stabilize: $\tilde{V}^{\pi_{n+1}}(s) \approx \tilde{V}^{\pi_n}(s)$
- Compute a high-precision approximation of $V^{\pi_n}(s)$ for the final policy.
- Optionally, do one more policy improvement step to guarantee we have an optimal policy.

Approximate Policy Iteration Example

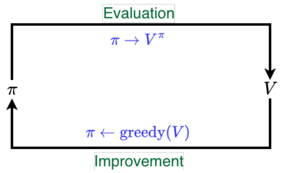
Approximate Policy Iteration for Tri-state MDP with 10 Gauss-Seidel relaxations for each policy improvement step:

n	$\pi_n(0$	$\pi_n(1)$	$\tilde{V}_n(0)$	$\tilde{V}_n(1)$	$\max_s \lvert\tilde{V}_n(s) - \tilde{V}_{n-1}(s)\rvert$
0	b	a	32.59054893	34.02505034	–
1	a	b	70.18751040	62.82240404	3.760×10^{1}
2	a	b	71.22266853	63.55216067	1.035×10^{0}
3	a	b	71.24929693	63.57093292	2.663×10^{-2}
4	a	b	71.24998191	63.57141582	6.850×10^{-4}

High precision value function for final policy, obtained with 200 Gauss-Seidel relaxations:

$$V(0) = 71.25000000, \quad V(1) = 63.57142857$$

Generalized Policy Iteration



- Policy iteration alternates two stages:
    - **Evaluation**: Given a policy π, evaluate to V^π.
    - **Improvement**: Given a value function V, compute a policy π greedy with respect to V.
- **Generalized policy iteration** (GPI) algorithm: interleaves these two processes arbitrarily.
    - Value iteration: policy improvement step is performed after $V(s)$ is updated for each state.
    - Asynchronous DP: the two processes are carried out in parallel.

Why we need more general evaluation methods?

- From now on, we assume that we have a policy π (it may be deterministic or randomized), and we want to estimate its state value function $V(s)$.
- The methods seen so far are *model based*: the transition probabilities and rewards are known, and can't change.
- We want methods that can handle:
  - **Non-model based problems**: transitions and rewards are not known.
  - **Dynamic environments**: transitions and rewards can change with time.
- We need agents that can **learn** by interacting with the environment.

Monte Carlo methods

- **Monte Carlo** methods are used to analyze complex systems using computer-based simulations. They find applications in many areas of science.
- In many applications, we want to estimate the *expected value* of a random variable.
- We simulate a large number of realizations of the random variable, and use the average of the outcomes as an estimate for the expected value.
- Legend has that they are called "Monte Carlo methods" because of the famous casino in Monaco.

Monte Carlo Monaco

Monte Carlo Recife

Estimating the value function for a policy — Naive algorithm

```
• Input:
    • $\pi$, a policy to be evaluated.
• Initialization:
    • $V(s)$ initialized arbitrarily.
• For each state $s$:
    • Let returns $= [\,]$
    • Let $S_0 = s$
    • Repeat a large enough number of times:
        • Generate an episode following $\pi$: $(S_0, R_0, A_0), \ldots, (S_T, R_T, A_t)$
        • Let $G = \sum_{t=0}^{T} \gamma^t R_{t+1}$
        • Append $G$ to returns
    • Let $V(s) = \mathrm{average}(G)$
```

Example

- MDP has states $\{0, 1, 2, \ldots, \mathtt{f}\}$ and actions $\{a, b\}$. State $\mathtt{f}$ is terminal.
- The following are the results of three runs of the MDP ($\gamma = 1$):

Run 1:	$(0, a, -), (4, b, 12), (2, a, 5), (\mathtt{f}, -, 3)$	Total return: 20
Run 2:	$(0, b, -), (0, b, 5), (\mathtt{f}, -, 5)$	Total return: 10
Run 1:	$(0, a, -), (3, a, 11), (1, a, 2), (\mathtt{f}, -, 2)$	Total return: 15

- From this we get the estimate:

$$V(0) \approx \frac{1}{3}(20 + 10 + 15) = 15$$

- Repeat this procedure for all possible initial states.

Being more efficient

- To make things simpler, from now on we will consider only *episodic tasks*.
- Suppose we have a run of an episode of length n starting in the state S_0:
$$(S_0, -), (S_1, R_1), (S_2, R_2), \ldots, (S_n, R_n)$$
- From this we get one value of the return for the initial state S_0, which is *one data point* that is used in the estimate of $V(S_0)$.
- However, we can consider a run that starts at time $t = 1$:
$$(S_1, R_1), (S_2, R_2), \ldots, (S_n, R_n)$$
- This gives me a data point for estimating $V(S_1)$, if $S_1 \neq S_0$!
- Be careful adjusting the discount factor!

First-visit MC method

- Input:
    - π, a policy to be evaluated.
- Initialization.
    - $V(s)$, initialized arbitrarily.
    - $\texttt{returns}(s)$, an empty list for each state s.
- Repeat a large enough number of times:
    - Generate an episode following π: $(S_0, A_0, R_0), \ldots, (S_T, A_T, R_T)$
    - Let $G = 0$
    - Loop for $t = T-1, T-2, \ldots, 0:$
        - $G = \gamma G + R_{t+1}$
        - If S_t does not appear in $S_0, S_2, \ldots, S_{t-1}$, append G to $\texttt{returns}(S_t)$
- Let $V(s) = \texttt{average}(\texttt{returns}(s))$ for $s \in \mathscr{S}$

First-visit MC method — Example

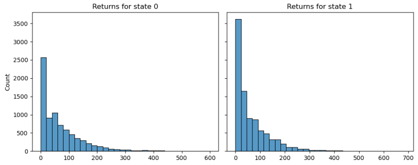
- For the Tri-state MDP, with policy $\pi(0) = a$, $\pi(1) = b$, simulating 10000 episodes we get the estimates:

$$V(0) = 71.18310, \quad V(1) = 63.25460$$

- Actual values:

$$V(0) = 71.25000, \quad V(1) = 63.57143$$

- Distribution of returns:



- $\mathrm{std}[V(0)] = 74.77556$, $\mathrm{std}[V(1)] = 74.56148$

Every-visit MC method

- Input:
    - π, an admissible policy to be evaluated.
- Initialization.
    - $V(s)$, initialized arbitrarily.
    - $\texttt{returns}(s)$, an empty list for each state s.
- Repeat a large enough number of times:
    - Generate an episode following π: $(S_0, A_0, R_0), \ldots, (S_T, A_T, R_T)$
    - Let $G = 0$
    - Loop for $t = T-1, T-2, \ldots, 0$:
        - $G = \gamma G + R_{t+1}$
        - Append G to $\texttt{returns}(S_t)$
- Let $V(s) = \texttt{average}(\texttt{returns}(s))$ for $s \in \mathscr{S}$

Every-visit MC method — Example

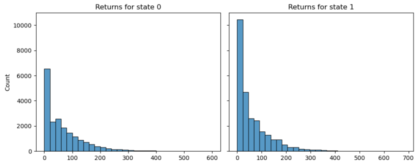
- For the Tri-state MDP, with policy $\pi(0) = a$, $\pi(1) = b$, simulating 10000 episodes we get the estimates:

$$V(0) = 69.84248, \quad V(1) = 62.18433$$

- Actual values:

$$V(0) = 71.25000, \quad V(1) = 63.57143$$

- Distribution of returns:



- $\mathrm{std}[V(0)] = 70.86432$, $\mathrm{std}[V(1)] = 73.64984$

Dynamic update of averages

- Suppose that we have a simulation that gives us a sequence return values $G_1, G_2, \ldots, G_N$. We use the average to estimate the expected return:

$$V_N = \frac{1}{N} \sum_{j=1}^{N} G_N$$

- Now, we compute a new value, G_{N+1}. How should we adjust the average? A simple computation shows that:

$$V_{N+1} = V_N + \frac{1}{N+1} [G_{N+1} - V_N]$$

- In control theory, this process is called *tracking*: The values of V_N are updated in a way that they "track" the values of G_N.
- Engineers like this because *the dynamics of the process G_N may change*, and the values of V_N can still try to "follow" it.

First-visit Monte Carlo with dynamic updates

- Input:
  - π, a policy to be evaluated.
- Initialization.
  - $V(s)$, initialized arbitrarily.
  - $\texttt{visits}(s) = 0$ for each state s.
- Repeat a large enough number of times:
  - Generate an episode following π: $(S_0, A_0, R_0), \dots, (S_T, A_T, R_T)$
  - Let $G = 0$
  - Loop for $t = T-1, T-2, \dots, 0$:
    - $G = \gamma G + R_{t+1}$
    - If S_t does not appear in $S_0, S_2, \dots, S_{t-1}$, increment $\texttt{visits}(s)$ by one and update:

$$V(s) \leftarrow V(s) + \frac{1}{\texttt{visits}(s)}[G - V(s)]$$

Target tracking

- The update rule:

$$V_{N+1} \leftarrow V_N + \frac{1}{N+1}[G_{N+1} - V_N]$$

  is an example of *tracking* a non-stationary sequence G_N.
- The general form of this update rule is:

$$\texttt{new\_estimate} = \texttt{old\_estimate} + \texttt{step\_size}[\texttt{target} - \texttt{old\_estimate}]$$

- A wide variety of learning algorithms fit this pattern.

TD Learning, Monte Carlo and DP

- TD Learning is a combination of Dynamic Programming and Monte Carlo methods.
- Monte Carlo aspect: learn from experience, does not need a model for the environment dynamics.
- DP aspect: update estimates based on other learned estimates (bootstrap).
- TD is the basis for many current RL algorithms.

Framework for policy evaluation

- Suppose we have:
  - An estimate $V(s)$ for the state value function for a policy π.
  - An episode $S_0, S_1, \ldots, S_T$ generated with the policy π.
- Denote by G_t the total discounted return for this episode after time t:

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

- For every visited state S_t, we update $V(S_t)$ to move in the direction of G_t.
- Different update strategies lead to distinct algorithms: Monte Carlo, TD(0), etc.

From Monte Carlo TD(0)

- Monte Carlo update rule:

$$V(S_t) \leftarrow V(S_t) + \frac{1}{\texttt{visits}(S_t)}[G_t - V(S_t)]$$

- Tracking with constant stepsize:

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

- TD(0): Instead of waiting for a whole episode to end, we use a one-step lookahead estimate of the value function:

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

TD relationship to to DP

- Bellman Equation for state value function V of a policy π:

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

  The policy π may be stochastic or deterministic.

- Value iteration is a *fixed point iteration* for the exact value of $V(s)$:

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

- TD(0) is a *stochastic fixed point iteration* that tracks a one-step lookahead:

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

  where $S_0, S_1, \ldots, S_T$ is an episode generated following policy π.

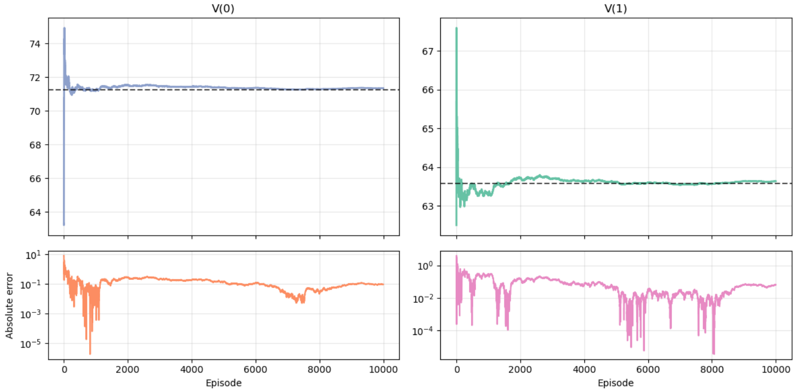
TD(0), Monte Carlo and DP

$$V_{\text{new}}(s) = V_{\text{old}}(S_t) + \alpha[r + \gamma V_{\text{old}}(s') - V_{\text{old}}(s)]$$

- Relationship to Monte Carlo:
  - Can learn directly from experience, independently on a model.
  - Monte Carlo algorithms have no bias and high variance.
  - TD(0) has bias controlled by the step size α and low variance.
- Relationship to DP
- TD methods update estimates based on already learned estimates (bootstrap)
- The TD(0) update rule is a form of *stochastic fixed point iteration*

Estimation by the Monte Carlo method

- Monte Carlo estimates for the Tri-state MDP with policy $\pi(0) = \text{a}$, $\pi(1) = \text{b}$ with 10000 episodes.
- Do 100 runs of the method and average.

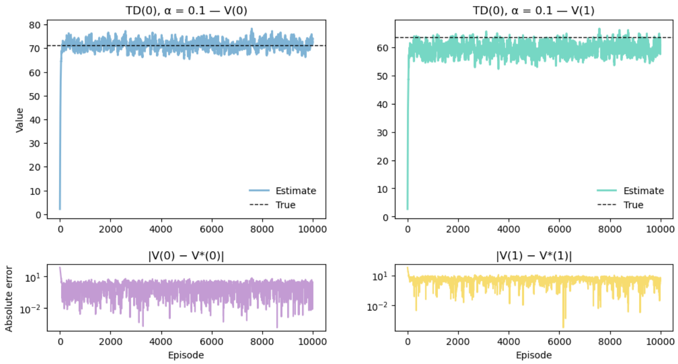


Interpretation of results for MC

- The value function estimates exhibit large transient oscillations, but stabilize around the true values. This behavior is consistent with an unbiased estimator.
- The absolute errors stabilize around 10^{-1}.
- The errors do not converge to zero at finite time due to the intrinsic variance of Monte Carlo returns and the finite number of effective samples per state.
- The magnitude of the error is governed by the standard deviation of the return distribution (empirically on the order of 70 in this model), with error scaling approximately as $\sigma_G/\sqrt{N}$.
- The sharp "dips" in the error correspond to episode indices where the mean estimate of $V(s)$ happens to be very close to the true value; this reflects bias cancellation rather than simultaneous accuracy across runs and is expected behavior.

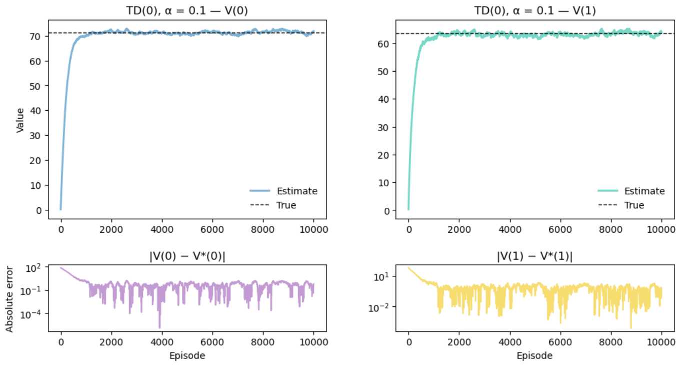
Estimation using TD(0) — $\alpha = 0.1$

- TD(0) estimates for the Tri-state MDP with policy $\pi(0) = \mathsf{a}$, $\pi(1) = \mathsf{b}$ with $\alpha = 0.1$ and 10000 episodes.
- Do 100 runs of the method and average.



Estimation using TD(0) — $\alpha = 0.01$

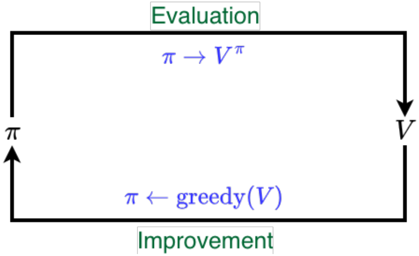
- TD(0) estimates for the Tri-state MDP with policy $\pi(0) = a$, $\pi(1) = b$ with $\alpha = 0.01$ and 10000 episodes.
- Do 100 runs of the method and average.



Interpretation of results for TD(0)

- TD(0) converges rapidly to a neighborhood of the true value function.
- With a fixed step-size, TD(0) does not converge pointwise, but to a stationary distribution around a biased fixed point.
- The stationary bias is *state-dependent* and need not be the same for all states.
- States with lower visitation frequency or closer to termination exhibit both higher variance and larger apparent bias.
- The variance of the estimator does not vanish and is controlled by the step-size: larger step-sizes yield faster learning but higher steady-state variance.

From estimation to control



We have not explored the "Evaluation" branch of the cycle. Next time we will see how this can be incorporated in a policy optimization algorithm.