4. Reinforcement Learning Notes: Policy Iteration

Rahul Sawhney

March 29, 2025

Contents

1 Introduction to Policy Iteration

Policy iteration is a method for finding an optimal policy in a Markov Decision Process (MDP). It works by alternating between two main steps:

1. **Policy Evaluation:** Computing the state-value function v_π for a given policy π.
2. **Policy Improvement:** Improving the policy by making it greedy with respect to the current value function.

This process continues until the policy converges to an optimal policy, denoted as π^*. Policy iteration is at the core of many reinforcement learning methods as it provides a fundamental approach to solving MDPs when the model (transition probabilities and rewards) is known.

2 Value Functions

2.1 State-Value Function

The state-value function $v_\pi(s)$ represents how good it is for an agent to be in a particular state s when following a policy π. Formally, it is defined as the expected return (expected cumulative discounted reward) when starting in state s and following policy π thereafter:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s] \tag{1}$$

Notation Overview

- $v_\pi(s)$: State-value function for state s under policy π
- $\mathbb{E}_\pi$: Expected value when following policy π
- G_t: Return (cumulative discounted reward) starting from time step t
- S_t: State at time step t
- $\doteq$: Defined as (definitional equality)

Utilizing the recursive property of the return, we can also express this as:

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \tag{2}$$

Notation Overview

- $v_\pi(s)$: State-value function for state s under policy π
- $\mathbb{E}_\pi$: Expected value when following policy π
- R_{t+1}: Reward received after taking action in state S_t
- γ: Discount factor, $0 \leq \gamma \leq 1$, determines the present value of future rewards
- G_{t+1}: Return starting from time step $t+1$
- S_t: State at time step t

2.2 Action-Value Function

The action-value function $q_\pi(s, a)$ represents how good it is for an agent to take a specific action a in state s and then follow policy π thereafter. Formally, it is defined as:

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \tag{3}$$

Notation Overview

- $q_\pi(s, a)$: Action-value function for taking action a in state s under policy π
- $\mathbb{E}_\pi$: Expected value when following policy π
- G_t: Return (cumulative discounted reward) starting from time step t
- S_t: State at time step t
- A_t: Action taken at time step t

2.3 Bellman Equations for Value Functions

The Bellman equations provide a recursive definition of value functions, establishing a fundamental relationship between the value of a state and the values of its successor states.

For the state-value function, the Bellman equation is:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \tag{4}$$

Notation Overview

- $v_\pi(s)$: State-value function for state s under policy π
- $\pi(a|s)$: Probability of taking action a in state s under policy π
- $p(s', r|s, a)$: Probability of transitioning to state s' and receiving reward r when taking action a in state s
- γ: Discount factor
- s': Next state
- r: Reward

For the action-value function, the Bellman equation is:

$$q_\pi(s, a) = \sum_{s', r} p(s', r|s, a)[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s', a')] \tag{5}$$

Notation Overview

- $q_\pi(s, a)$: Action-value function for taking action a in state s under policy π
- $p(s', r|s, a)$: Probability of transitioning to state s' and receiving reward r when taking action a in state s
- $\pi(a'|s')$: Probability of taking action a' in state s' under policy π
- γ: Discount factor
- s': Next state
- r: Reward
- a': Next action

2.4 Relationship Between Value Functions

The state-value function and action-value function are closely related. The relationship can be expressed as:

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s, a) \tag{6}$$

Notation Overview

- $v_\pi(s)$: State-value function for state s under policy π
- $q_\pi(s, a)$: Action-value function for taking action a in state s under policy π
- $\pi(a|s)$: Probability of taking action a in state s under policy π

And conversely:

$$q_\pi(s, a) = \sum_{s', r} p(s', r|s, a)[r + \gamma v_\pi(s')] \tag{7}$$

Notation Overview

- $q_\pi(s, a)$: Action-value function for taking action a in state s under policy π
- $p(s', r|s, a)$: Probability of transitioning to state s' and receiving reward r when taking action a in state s
- γ: Discount factor
- $v_\pi(s')$: State-value function for state s' under policy π
- s': Next state
- r: Reward

3 Optimal Value Functions

3.1 Definitions and Properties

An optimal policy is a policy that achieves the maximum possible expected return from all states. The optimal state-value function, denoted as $v_*(s)$, gives the maximum value achievable for each state:

$$v_*(s) \doteq \max_\pi v_\pi(s) \tag{8}$$

Notation Overview

- $v_*(s)$: Optimal state-value function for state s
- $\max_\pi$: Maximum over all possible policies π
- $v_\pi(s)$: State-value function for state s under policy π

Similarly, the optimal action-value function, denoted as $q_*(s, a)$, gives the maximum value achievable for each state-action pair:

$$q_*(s, a) \doteq \max_\pi q_\pi(s, a) \tag{9}$$

Notation Overview

- $q_*(s, a)$: Optimal action-value function for taking action a in state s
- $\max_\pi$: Maximum over all possible policies π
- $q_\pi(s, a)$: Action-value function for taking action a in state s under policy π

3.2 Bellman Optimality Equations

The Bellman optimality equation for the state-value function is:

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \tag{10}$$

Notation Overview

- $v_*(s)$: Optimal state-value function for state s
- $\max_a$: Maximum over all possible actions a
- $p(s', r|s, a)$: Probability of transitioning to state s' and receiving reward r when taking action a in state s
- γ: Discount factor
- s': Next state
- r: Reward

The Bellman optimality equation for the action-value function is:

$$q_*(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} q_*(s', a')] \tag{11}$$

Notation Overview

- $q_*(s, a)$: Optimal action-value function for taking action a in state s
- $p(s', r|s, a)$: Probability of transitioning to state s' and receiving reward r when taking action a in state s
- $\max_{a'}$: Maximum over all possible next actions a'
- γ: Discount factor
- s': Next state
- r: Reward
- a': Next action

3.3 Relationship Between Optimal Value Functions

The relationship between the optimal state-value function and the optimal action-value function can be expressed as:

$$v_*(s) = \max_a q_*(s, a) \tag{12}$$

Notation Overview

- $v_*(s)$: Optimal state-value function for state s
- $q_*(s, a)$: Optimal action-value function for taking action a in state s
- $\max_a$: Maximum over all possible actions a

This shows that the optimal value of a state is the maximum of the optimal action-values for that state, which corresponds to taking the best possible action in that state.

4 Policy Evaluation

Policy evaluation is the process of computing the state-value function v_π for a given policy π. This is a crucial step in policy iteration.

4.1 Iterative Policy Evaluation

The Bellman equation for the state-value function provides a system of linear equations with one equation for each state. For a finite MDP, this system can be

solved directly. However, iterative methods are often more practical, especially for large state spaces.

The iterative policy evaluation algorithm updates the value function as follows:

$$v_{k+1}(s) = \sum_{a} \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_k(s')] \tag{13}$$

Notation Overview

- $v_{k+1}(s)$: Estimated state-value of state s after $k+1$ iterations
- $v_k(s)$: Estimated state-value of state s after k iterations
- $\pi(a|s)$: Probability of taking action a in state s under policy π
- $p(s', r|s, a)$: Probability of transitioning to state s' and receiving reward r when taking action a in state s
- γ: Discount factor
- s': Next state
- r: Reward

This is a form of dynamic programming update, which converges to the true value function v_π as the number of iterations approaches infinity.

4.2 Iterative Policy Evaluation Algorithm

Algorithm 1 Iterative Policy Evaluation (for estimating $V \approx v_\pi$)

Input: A policy π to be evaluated
Parameter: A small threshold $\theta > 0$ determining the accuracy of estimation
Initialize $V(s) = 0$, for all $s \in \mathcal{S}^+$ (except for terminal states where $V(terminal) = 0$)
repeat
    $\Delta \leftarrow 0$
    for each $s \in \mathcal{S}$ **do**
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
    end for
until $\Delta < \theta$
return $V \approx v_\pi$

Notation Overview

- $V(s)$: Estimated state-value function
- $\pi(a|s)$: Probability of taking action a in state s under policy π
- $p(s', r|s, a)$: Probability of transitioning to state s' and receiving reward r when taking action a in state s
- γ: Discount factor
- $\mathcal{S}$: Set of all non-terminal states
- $\mathcal{S}^+$: Set of all states, including terminal states
- Δ: Maximum change in value function
- θ: Threshold for convergence

5 Policy Improvement

Policy improvement is the process of making a policy better by making it greedy with respect to the current value function.

5.1 Policy Improvement Theorem

The policy improvement theorem states that if we have two policies π and π' such that, for all states $s \in \mathcal{S}$:

$$q_\pi(s, \pi'(s)) \geq v_\pi(s) \tag{14}$$

Notation Overview

- $q_\pi(s, \pi'(s))$: Action-value of taking action prescribed by policy π' in state s, then following policy π
- $v_\pi(s)$: State-value of state s under policy π
- $\pi'(s)$: Action prescribed by policy π' in state s
- $\mathcal{S}$: Set of all non-terminal states

Then policy π' is at least as good as π, meaning that $v_{\pi'}(s) \geq v_\pi(s)$ for all $s \in \mathcal{S}$. Furthermore, if there exists at least one state where $q_\pi(s, \pi'(s)) > v_\pi(s)$, then π' is strictly better than π in at least one state.

The proof of this theorem is based on the recursive nature of value functions. If we start from any state s where $q_\pi(s, \pi'(s)) > v_\pi(s)$, we can show that following π' will lead to higher expected returns than following π.

5.2 Greedy Policy Improvement

A natural way to improve a policy is to make it greedy with respect to the current value function. For any deterministic policy π, the improved policy π' is:

$$\pi'(s) = \arg\max_a q_\pi(s,a) = \arg\max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')] \tag{15}$$

Notation Overview

- $\pi'(s)$: Action prescribed by the improved policy in state s
- $\arg\max_a$: The action a that maximizes the following expression
- $q_\pi(s,a)$: Action-value of taking action a in state s, then following policy π
- $p(s',r|s,a)$: Probability of transitioning to state s' and receiving reward r when taking action a in state s
- γ: Discount factor
- $v_\pi(s')$: State-value of state s' under policy π

The policy improvement theorem guarantees that this new policy π' will be strictly better than π unless π is already optimal.

6 Policy Iteration Algorithm

Policy iteration combines policy evaluation and policy improvement in an iterative process, which converges to an optimal policy.

Algorithm 2 Policy Iteration (for estimating $\pi \approx \pi_*$)

Initialize: $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$
repeat
    // Policy Evaluation
    repeat
        $\Delta \leftarrow 0$
        for each $s \in \mathcal{S}$ **do**
            $v \leftarrow V(s)$
            $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
            $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
        end for
    until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)
    // Policy Improvement
    policy-stable $\leftarrow$ true
    for each $s \in \mathcal{S}$ **do**
        old-action $\leftarrow \pi(s)$
        $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
        if old-action $\neq \pi(s)$ **then**
            policy-stable $\leftarrow$ false
        end if
    end for
until policy-stable
return $V \approx v_*$ and $\pi \approx \pi_*$

Notation Overview

- $V(s)$: Estimated state-value function
- $\pi(s)$: Current policy (deterministic)
- $\mathcal{S}$: Set of all non-terminal states
- $\mathcal{A}(s)$: Set of actions available in state s
- $p(s', r|s, a)$: Probability of transitioning to state s' and receiving reward r when taking action a in state s
- γ: Discount factor
- θ: Threshold for convergence in policy evaluation
- Δ: Maximum change in value function
- policy-stable: Boolean indicating whether the policy has changed
- old-action: Action prescribed by the old policy
- $\arg\max_a$: The action a that maximizes the following expression

6.1 Convergence of Policy Iteration

Policy iteration is guaranteed to converge to the optimal policy π_* and the optimal value function v_* in a finite number of iterations for any finite MDP. This is because:

1. The set of policies for a finite MDP is finite.
2. Each policy improvement step produces a strictly better policy unless the current policy is already optimal.
3. Value functions are bounded for finite MDPs with a discount factor $\gamma < 1$.

Therefore, policy iteration must converge to an optimal policy in a finite number of iterations.

7 Value Iteration

Value iteration is a variant of policy iteration that combines the policy evaluation and policy improvement steps into a single update. Instead of waiting for policy evaluation to converge completely before policy improvement, value iteration performs a single sweep of policy evaluation followed immediately by policy improvement.

The update rule for value iteration is:

$$v_{k+1}(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_k(s')] \tag{16}$$

Notation Overview

- $v_{k+1}(s)$: Estimated state-value of state s after $k+1$ iterations
- $v_k(s)$: Estimated state-value of state s after k iterations
- $\max_a$: Maximum over all possible actions a
- $p(s', r|s, a)$: Probability of transitioning to state s' and receiving reward r when taking action a in state s
- γ: Discount factor
- s': Next state
- r: Reward

Algorithm 3 Value Iteration (for estimating $\pi \approx \pi_*$)

Initialize: $V(s) = 0$, for all $s \in \mathcal{S}^+$
repeat
    $\Delta \leftarrow 0$
    for each $s \in \mathcal{S}$ **do**
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
    end for
until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)
// Output a deterministic policy
for each $s \in \mathcal{S}$ **do**
    $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
end for
return $\pi \approx \pi_*$

Notation Overview

- $V(s)$: Estimated state-value function
- $\mathcal{S}$: Set of all non-terminal states
- $\mathcal{S}^+$: Set of all states, including terminal states
- $\max_a$: Maximum over all possible actions a
- $\arg\max_a$: The action a that maximizes the following expression
- $p(s', r|s, a)$: Probability of transitioning to state s' and receiving reward r when taking action a in state s
- γ: Discount factor
- θ: Threshold for convergence
- Δ: Maximum change in value function
- $\pi(s)$: Deterministic optimal policy

8 Example: Grid World

Let's consider a simple grid world example to illustrate policy iteration. The grid world is a 4x4 grid with states labeled from (0,0) to (3,3). The agent can move in four directions: up, down, left, and right. If the agent attempts to move off the grid, it remains in the same state. The agent receives a reward of -1 for each move, except when it reaches the goal state (3,3), where it receives a reward of +10 and the episode terminates.

			Goal
(0,3)	(1,3)	(2,3)	(3,3)
(0,2)	(1,2)	(2,2)	(3,2)
(0,1)	(1,1)	(2,1)	(3,1)
(0,0)	(1,0)	(2,0)	(3,0)

Figure 1: 4x4 Grid World

8.1 Initial Policy and Value Function

We start with an arbitrary policy, such as moving right in all states, and initialize the value function to zero for all states (except terminal states).

8.2 Policy Evaluation

Using the policy evaluation algorithm, we iteratively update the value function for the current policy. For example, for state (0,0), following the "move right" policy:

$$\begin{aligned} v_\pi(0,0) &= \sum_a \pi(a|(0,0)) \sum_{s',r} p(s',r|(0,0),a)[r + \gamma v_\pi(s')] \\ &= 1 \cdot \sum_{s',r} p(s',r|(0,0),\text{right})[r + \gamma v_\pi(s')] \\ &= 1 \cdot [(-1) + 0.9 \cdot v_\pi(1,0)] \end{aligned}$$

Assuming $\gamma = 0.9$, and continuing this process for all states, we eventually converge to the value function for the initial policy.

8.3 Policy Improvement

Once we have the value function for the current policy, we improve the policy by making it greedy with respect to the value function:

$$\pi'(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$$

For example, for state (0,0), we would consider all four actions (up, down, left, right) and choose the one that gives the highest expected return. After improving the policy for all states, we check if the policy has changed. If it has, we go back to the policy evaluation step with the new policy.

8.4 Convergence

We repeat the policy evaluation and policy improvement steps until the policy no longer changes. At this point, we have found an optimal policy for the grid world, which tells the agent the best action to take in each state to maximize the expected return.

9 Generalized Policy Iteration

Generalized Policy Iteration (GPI) is a term that refers to the general idea of letting policy evaluation and policy improvement processes interact, regardless

of how they are implemented. Most reinforcement learning methods can be viewed as approximations to GPI.

In GPI, the value function is repeatedly pushed toward the value function for the current policy (policy evaluation), while the policy is repeatedly improved with respect to the current value function (policy improvement). These two processes work together, both pushing toward the optimal value function and optimal policy.

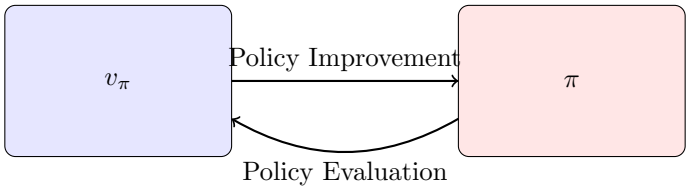

Figure 2: Generalized Policy Iteration

10 Efficiency of Policy Iteration vs Value Iteration

Both policy iteration and value iteration are guaranteed to converge to an optimal policy, but they have different computational characteristics:

- **Policy Iteration**: Requires fewer iterations but each iteration is more computationally expensive due to the complete policy evaluation step.
- **Value Iteration**: Requires more iterations but each iteration is less computationally expensive due to the single backup operation.

In practice, a modified version of policy iteration is often used, where the policy evaluation step is truncated after a fixed number of iterations or when a certain convergence criterion is met. This approach, known as truncated policy iteration, combines the advantages of both policy iteration and value iteration.

11 Theoretical Guarantees

Policy iteration is guaranteed to converge to an optimal policy and value function in a finite number of iterations for any finite MDP. This is because:

1. The policy improvement step always yields a strictly better policy unless the current policy is already optimal.
2. There are only a finite number of deterministic policies for a finite MDP.

Similarly, value iteration is also guaranteed to converge to the optimal value function for any finite MDP, as long as the discount factor $\gamma < 1$ (for continuing tasks) or all episodes terminate (for episodic tasks).

12 Conclusion

Policy iteration is a fundamental algorithm in reinforcement learning for finding optimal policies in MDPs. It consists of two main steps: policy evaluation, which computes the value function for a given policy, and policy improvement, which makes the policy greedy with respect to the computed value function. The algorithm alternates between these two steps until convergence.

Value iteration is a variant of policy iteration that combines the policy evaluation and policy improvement steps into a single update, often resulting in faster convergence for large state spaces.

Both policy iteration and value iteration are instances of the more general concept of Generalized Policy Iteration (GPI), which forms the foundation of many reinforcement learning algorithms.

When the MDP model (transition probabilities and rewards) is known, policy iteration and value iteration provide powerful tools for solving reinforcement learning problems. However, in many practical scenarios, the model is not known, leading to the development of model-free reinforcement learning methods that estimate value functions directly from experience.