Convergence of Policy & Value Iteration in Markov Decision Processes

Navneet Kashyap

1 Introduction

Markov Decision Processes (MDPs) provide a mathematical framework for sequential decision-making in stochastic environment i.e when the outcome is not completely determined and has a probsbility distribution over the possible next states. Two fundamental algorithms for solving MDPs are:

- **Policy Iteration (PI)**: Alternates between policy evaluation and policy improvement
- **Value Iteration (VI)**: Directly uses the Bellman optimality equation

Both algorithms converge to the optimal policy π^* and optimal value function V^* under certain conditions.

2 Intuition Behind the Algorithms

2.1 Policy Iteration

- **Policy Evaluation**: Given a policy π, compute its value function V^π by solving:

$$V^\pi(s) = \sum_a \pi(a|s) \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^\pi(s') \right]$$

  This is a linear system that can be solved exactly or iteratively.

- **Policy Improvement**: Update the policy greedily with respect to V^π:

$$\pi'(s) = \arg\max_a \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^\pi(s') \right]$$

  The new policy π' is strictly better unless π is already optimal.

Why it works: The general idea is that we start off with a random policy. Policy evaluation finds the value function associated for all the states. Then in policy improvement we check if performing an action and then following the policy π gives a better value than following π from the start. If that is so then we change our policy so that whenever we choose that state then we perform that action. We do this for all the states and thus generate a new policy. We then again start over with evaluation and keep repeating. If we try to draw a very crude parallel between gradient descent and policy iteration then policy evaluation is finding the value of the function and policy improvement is taking a step in the opposite direction of the gradient. Of course the style of working is entirely different but the idea of starting off at a random solution and then bit by bit building the correct solution is same.

2.2 Value Iteration

- Direct application of the Bellman optimality operator:

$$V_{k+1}(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_k(s') \right]$$

- The contraction mapping ensures $V_k \to V^*$ as $k \to \infty$

Why it works: The idea here is that instead of improving policy and then evaluating it and building both policy and value function to the optimal case here we only focus on finding the optimal value function without caring about the policy until the very end. Once we have found the optimal value function then we extract what should be the optimal policy from there.

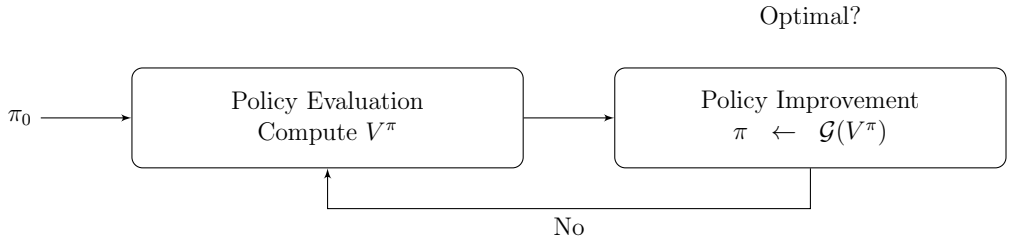

Figure 1: Policy Iteration Flow

3 Generalized Policy Iteration (GPI)

Definition 3.1. GPI is the conceptual framework where *any* combination of policy evaluation and policy improvement steps interact to reach the optimal policy and value function. PI and VI are special cases:

- **Policy Iteration**: Full policy evaluation + greedy improvement
- **Value Iteration**: One evaluation sweep + improvement
- **Modified PI**: Partial evaluation (e.g., k sweeps) + improvement

Theorem 3.1 (GPI Convergence)**.** Any GPI method converges to π^* and V^* if:

1. Evaluation establishes V^π or its approximation
2. Improvement generates $\pi' \geq \pi$ (monotonic improvement)
3. The MDP is finite and $\gamma < 1$

4 Convergence Proofs

4.1 Value Iteration Convergence

Theorem 4.1. For any initial V_0, VI converges to V^* in discounted MDPs ($\gamma < 1$).

Proof. Define the Bellman optimality operator $\mathcal{T}$:

$$(\mathcal{T}V)(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a)V(s') \right]$$

This operator takes in a function and returns a new function $V(s)$

1. **Contraction**: A contraction is a function that brings things closer together. That means, if you apply it to two value functions V and U, the new outputs are closer than the original inputs.

$$\|\mathcal{T}V - \mathcal{T}U\|_\infty \leq \gamma \|V - U\|_\infty$$

The metric used here is the max-norm. Metric is a way of generalizing distance functions. There is euclidean distance, minkowski distance and many more. What we are trying to prove is that applying this operator brings the max distance between the two functions closer by atleast gamma times.

A side note : The Bellman operator is not guaranteed to be a contraction under all norms

$$|\mathcal{T}V(s) - \mathcal{T}U(s)| \leq \max_a \gamma \sum_{s'} P(s'|s,a)|V(s') - U(s')|$$
$$\leq \gamma \|V - U\|_\infty$$

This shows that if we keep applying $\mathcal{T}$ then the value function keeps getting closer and closer to each other

2. **Fixed Point**: $\mathcal{T}V^* = V^*$ (by Bellman optimality)

   This just means that V^* is a stable solution of the Bellman update and we can't improve it more.

3. By Banach Fixed-Point Theorem, iterating $\mathcal{T}$ converges to V^* at rate γ:

$$\|V_k - V^*\|_\infty \leq \frac{\gamma^k}{1-\gamma} \|V_1 - V_0\|_\infty$$

   Banach Fixed-Point theorem is a powerful theorem saying that if you have a contraction function (like $\mathcal{T}$) and you apply it repeatedly to any starting point, it will eventually converge to the unique fixed point (here, V^*).

□

4.2 Policy Iteration Convergence

Theorem 4.2. PI converges to π^* in finite MDPs ($|\mathcal{S}| < \infty$, $|\mathcal{A}| < \infty$) in finite iterations.

Proof. 1. **Monotonic Improvement**: $\pi_{k+1} \geq \pi_k$ (componentwise)

$$V^{\pi_{k+1}}(s) \geq V^{\pi_k}(s) \quad \forall s$$

   with strict inequality if π_k is suboptimal.

   This is guaranteed by the policy improvement theorem.

2. **Finiteness**: There are at most $|\mathcal{A}|^{|\mathcal{S}|}$ policies

3. **Termination**: Algorithm stops when $\pi_{k+1} = \pi_k$, which implies:

$$V^{\pi_k} = \mathcal{T}V^{\pi_k} \implies \pi_k = \pi^*$$

The intuitive idea of the proof is that if your algorithm always improves and there is only finite many policies, and the algorithm will stop only when it has reached the optimal policy, all of these pretty much guarantee that the algo will eventually reach the optimal conditions □

5 Complexity Analysis

Table 1: Complexity Comparison ($n = |\mathcal{S}|$, $m = |\mathcal{A}|$)

Algorithm	**Time per Iteration**	**Memory**
Policy Iteration	$\mathcal{O}(mn^2 + n^3)$	$\mathcal{O}(n)$
- Evaluation (direct)	$\mathcal{O}(n^3)$	
- Evaluation (iterative)	$\mathcal{O}(mn^2)$	
- Improvement	$\mathcal{O}(mn^2)$	
Value Iteration	$\mathcal{O}(mn^2)$	$\mathcal{O}(n)$

5.1 Convergence Rates

- **Value Iteration**: Linear convergence with factor γ

$$\|V_k - V^*\|_\infty \leq \frac{\gamma^k}{1-\gamma}\|V_1 - V_0\|_\infty$$

- **Policy Iteration**: faster than linear FTL :) convergence in practice (finite iterations)

6 Applications and Trade-offs

Algorithm	**When to Use**
Policy Iteration	• Small state spaces ($n < 10^3$) • When exact solution is required • When policy evaluation can use fast linear solvers
Value Iteration	• Large state spaces ($n > 10^3$) • When approximate solutions suffice • When $P(s'\|s, a)$ is sparse • Preferred in reinforcement learning

6.1 Practical Considerations

- **VI Advantages**: Simple implementation, low per-iteration cost

- **PI Advantages**: Faster convergence when policy evaluation is efficient
- **Hybrid Approaches**: Use VI in policy evaluation step (modified PI)
- **Stopping Criteria**: $\|V_k - V_{k-1}\|_\infty < \epsilon$ (VI) or $\pi_k = \pi_{k-1}$ (PI)

We formally proved the convergence of both VI and PI and discussed what is happening intuitively here.