Exact Methods : Value and Policy Iteration

Easwar Subramanian

TCS Innovation Labs, Hyderabad

Email : easwar.subramanian@tcs.com / cs5500.2020@iith.ac.in

September 6, 2021

Overview

1. Review
2. Value Iteration
3. Policy Iteration

Review

Optimiality in Policies

- A partial ordering over policies is given by

$$\pi \geq \pi', \quad \text{if} \quad V^{\pi}(s) \geq V^{\pi'}(s), \quad \forall s \in \mathcal{S}$$

- Given an MDP, under mild assumptions, there exists an optimal policy π_* that is better than or equal to all other policies.
- All optimal policies achieve the optimal state value function $V_*(s) = V^{\pi_*}(s)$ and optimal action value function $Q_*(s,a) = Q^{\pi_*}(s,a)$
- Solving an MDP means finding a policy π_* such that

$$\pi_* = \arg\max_{\pi} \left[\mathbb{E}_{\pi} \left(\sum_{t=0}^{\infty} \gamma^t r_{t+1} \right) \right]$$

- Solving an MDP means finding an **optimal value function** V_* or **optimal action value function** Q_* or **optimal policy** π_*

Bellman's Optimality Equations

- Optimality equation for state value function

$$V_*(s) = \max_a Q_*(s,a) = \max_a \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V_*(s') \right) \right]$$

- Optimality equation for action value function

$$Q_*(s,a) = \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q_*(s',a') \right) \right]$$

- Optimality equations are non-linear system of equations with n unknowns and n non-linear constraints (i.e., the max operator).
- Iterative methods are typically used to solve for optimal state and action value functions

Dynamic Programming

A method to solve complex problem by

- Break the complex problem into sub-problems
- Solve sub-problems
- Combine solutions of sub-problems

Problems with **Optimal substructures** and **Overlapping sub-problems** can be solved using dynamic programming

- The recursive decomposition given by Bellman equations of (action) value functions pave way to solve an MDP using dynamic programming

Slide Credit: David Silver's UCL course

Value Iteration

Value Iteration

- Value iteration is an iterative algorithm for finding $V_*(s),\ \forall s \in \mathcal{S}$
- Once $V_*(s),\ \forall s \in \mathcal{S}$ is found, then optimal policy π_* can be found using the greedy evaluation of the $V_*(s)$

Value Iteration : Algorithm

TCS Research & Innovation

Algorithm Value Iteration

1: Start with an initial value function $V_1(\cdot)$;
2: **for** $k = 1, 2, \cdots, K$ **do**
3: **for** $s \in \mathcal{S}$ **do**
4: Calculate

$$V_{k+1}(s) \leftarrow \max_a \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V_k(s') \right) \right]$$

5: **end for**
6: **end for**

TATA CONSULTANCY SERVICES

Value Iteration : Example

No noise and discount factor $\gamma = 1$

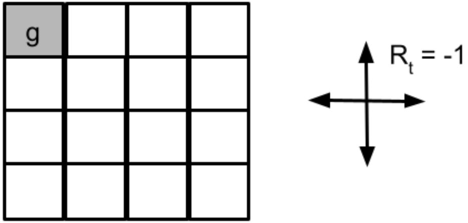


Figure Source: David Silver's UCL Course

Value Iteration : Example

g			

Problem

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

V_1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1

V_2

0	-1	-2	-2
-1	-2	-2	-2
-2	-2	-2	-2
-2	-2	-2	-2

V_3

0	-1	-2	-3
-1	-2	-3	-3
-2	-3	-3	-3
-3	-3	-3	-3

V_4

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-4
-3	-4	-4	-4

V_5

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-5

V_6

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-6

V_7

Value Iteration : Remarks

- The sequence of value functions $\{V_1, V_2, \cdots, \}$ converge
- It converges to V_*
- Convergence is independent of the choice of V_1.
- Intermediate value functions need not correspond to a policy in the sense of satisfying the Bellman Evaluation Equation

Iterative Procedure for Optimal Action Value Function

There is a recursive formulation for $Q_*(\cdot, \cdot)$

$$Q_*(s,a) = \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q_*(s', a')\right)\right]$$

One could similarly conceive an iterative algorithm to compute optimal Q_* using the above recursive formulation !!

Iterative Procedure for Bellman Evaluation Equation

The Bellman Evaluation Equation for an MDP with policy π

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

One could conceive a similar iterative update rule and arrive at a sequence of value functions $\{V_1^{\pi}, V_2^{\pi}, \cdots, \}$ that converges to V^{π}

Policy Iteration

Policy Iteration : Problem

Question : Is there a way to arrive at π_* starting from an arbitrary policy π ?

Answer : Policy Iteration

To describe policy iteration, we first need to know to evaluate a policy π using value functions.

Policy Evaluation

- **Problem** : Evaluate a given policy π
- Compute $V^{\pi}(s) = \mathbb{E}_{\pi}(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s)$
- **Solution 1** : Solve a system of linear equations using any solver
- **Solution 2** : Iterative application of Bellman Evaluation Equation
- Iterative update rule :

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{a} \pi(a|s) \sum_{s'} \mathcal{P}_{ss'}^{a} \left[\mathcal{R}_{ss'}^{a} + \gamma V_{k}^{\pi}(s')\right]$$

- The sequence of value functions $\{V_1^{\pi}, V_2^{\pi}, \cdots, \}$ converge to V^{π}

Policy Improvement

Suppose we know V^π

Question :

How can we improve policy π ? Is there a way to come up with a policy that is better than or equal to policy π ?

Answer :

$$\pi' = \text{greedy}(V^\pi)$$

This is the **policy improvement step** :

Greedy Policy

For a given $Q^\pi(\cdot,\cdot)$, define $\pi'(s)$ as follows

$$\pi'(s) = \text{greedy}(Q) = \begin{cases} 1 & \text{if } a = \arg\max_{a\in\mathcal{A}} Q^\pi(s,a) \\ 0 & \text{Otherwise} \end{cases}$$

For a given $V^\pi(\cdot)$, define $\pi'(s)$ as follows

$$\pi'(s) = \text{greedy}(V) = \begin{cases} 1 & \text{if } a = \arg\max_{a\in\mathcal{A}} \left[\sum_{s'\in\mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V^\pi(s')\right)\right] \\ 0 & \text{Otherwise} \end{cases}$$

Greedy Policy

Greedy policy with respect to optimal (action) value function is an optimal policy

An optimal policy can be found by maximising over $Q_*(s, a)$

$$\pi_*(s) = \begin{cases} 1 & \text{if } a = \arg\max_{a \in \mathcal{A}} Q_*(s, a) \\ 0 & \text{Otherwise} \end{cases}$$

Policy Improvement

- Consider the policy $\pi' = \textbf{greedy}(V^\pi)$.
- Then, $\pi' \geq \pi$. That is, $V^{\pi'}(s) \geq V^\pi(s)$ for all $s \in \mathcal{S}$.
- By defintion of π', at state s, the action chosen by policy π' is given by the greedy operator

$$\pi'(s) = \arg\max_a Q^\pi(s,a)$$

- This improves the value from any state s over one step

$$Q^\pi(s, \pi'(s)) = \max_a Q^\pi(s,a) \geq Q^\pi(s, \pi(s)) = V^\pi(s)$$

- It therefore improves the value function, $V^{\pi'}(s) \geq V^\pi(s)$

$$V^\pi(s) \leq Q^\pi(s, \pi'(s)) \leq V^{\pi'}(s)$$

- Policy π' is at least as good as policy π

Figure Source: Refer to David Silver Lecture 3 slides for a more detailed proof

Policy Improvement

- If improvements stop,

$$Q^{\pi}(s, \pi'(s)) = \max_a Q^{\pi}(s, a) = Q^{\pi}(s, \pi(s)) = V^{\pi}(s)$$

- Bellman optimality equation is satisfied as,

$$V^{\pi}(s) = \max_a Q^{\pi}(s, a)$$

- The policy π for which the improvement stops is the optimal policy.

$$V^{\pi}(s) = V_*(s) \quad \forall s \in \mathcal{S}$$

Policy Iteration : Algorithm

Algorithm Policy Iteration

1: Start with an initial policy π_1
2: **for** $i = 1, 2, \cdots, N$ **do**
3: Evaluate $V^{\pi_i}(s) \quad \forall s \in \mathcal{S}$. That is,
4: **for** $k = 1, 2, \cdots, K$ **do**
5: For all $s \in \mathcal{S}$ calculate

$$V_{k+1}^{\pi_i}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} \mathcal{P}_{ss'}^a \left[\mathcal{R}_{ss'}^a + \gamma V_k^{\pi_i}(s')\right]$$

6: **end for**
7: Perform policy Improvement

$$\pi_{i+1} = \text{greedy}(V^{\pi_i})$$

8: **end for**

Policy Iteration : Example

Update Rule :

$$V_{k+1}^{\pi_i}(s) \leftarrow \sum_{a} \pi(a|s) \sum_{s'} \mathcal{P}_{ss'}^{a} \left[\mathcal{R}_{ss'}^{a} + \gamma V_{k}^{\pi_i}(s')\right]$$

v_k for the random policy

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

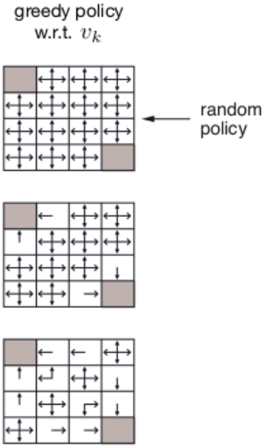


Figure Source: David Silver's UCL course

Policy Iteration : Example

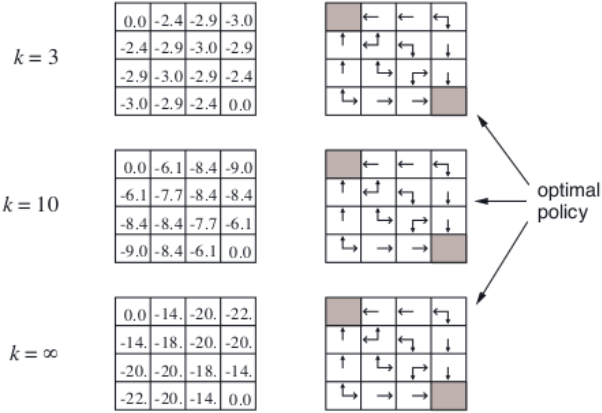
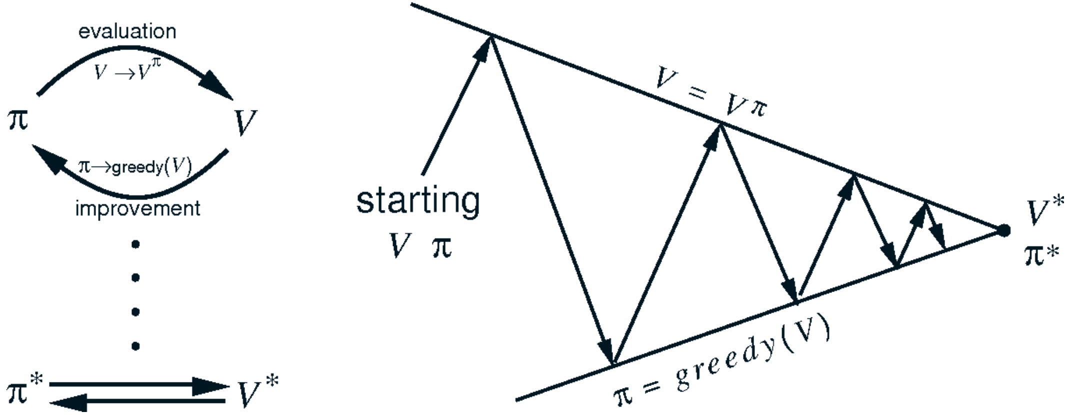


Figure Source: David Silver's UCL course

Policy Iteration : Schematic Representation



- The sequence $\{\pi_1, \pi_2, \cdots,\}$ is guaranteed to converge.
- At convergence, both current policy and the value function associated with the policy are optimal.

Figure Source: David Silver's UCL course

Modified Policy Iteration

Can we computationally simplify policy iteration process ?

- We need not wait for policy evaluation to converge to V^{π}
- We can have a stopping criterion like ϵ-convergence of value function evaluation or K iterations of policy evaluation
- Extreme case of $K = 1$ is value iteration. We update the policy every iteration