Planning with a Known Model

(INF8250AE: Introduction to Reinforcement Learning)

Amir-massoud Farahmand

Polytechnique Montréal & Mila

Adaptive Agents Lab (Adage)

Table of Contents

Goal

We study some algorithms to solve the Planning problem: computing the (optimal) value function for a given MDP.

- Know the fundamental planning algorithms:
  - Value Iteration (VI)
  - Policy Iteration (PI)
  - Linear Programming
- Learn the intuition behind them
- Learn the mathematics justifying them

We refer to these frequently in studying and analyzing RL/Planning algorithms.

Learning Objectives

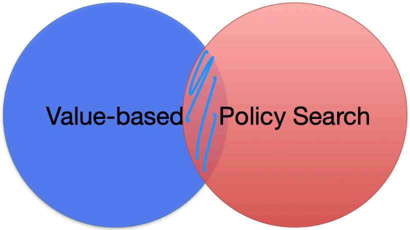
You need to

- Remember: Value Iteration, Policy Iteration, and LP-based algorithms
- Understand: Why VI, PI, LP work
- Apply: VI and PI to solve an MDP

How to Compute the Optimal Policy π^*?

- We have defined concepts and properties such as
  - Value function for a policy π (V^π) and optimal value function (V^*)
  - Relation between V^* (or Q^*) and π^* through the greedy policy
- Question: How can we find the optimal policy?
- Assumption: MDP is known, that is, we know $\mathcal{R}$ and $\mathcal{P}$.
- The assumption of knowing the MDP does not hold in the RL setting.
- But designing methods for finding the optimal policy with known model provides the foundation for developing methods for the RL setting.

Different Approaches to Find π^*



- Value-based: Compute Q^* (or V^*) and then $\pi^* \leftarrow \pi_g(Q^*)$.
- Direct policy search: Search in the space of policies without explicitly constructing the optimal value function.
- Hybrid: Explicitly construct value function to guide the search in the policy space.

Policy Evaluation vs. Control Problems

- Policy Evaluation (PE): Problem of computing the value function of a given policy π, i.e., V^π or Q^π.
  - Not the ultimate goal of an RL agent (finding the optimal policy is), but is often needed as an intermediate step in finding the optimal policy.
- Control: Problem of finding the optimal value function V^* or Q^* or optimal policy π^*.

Dynamic Programming (DP): Methods that benefit from the structure of the MDP, such as the recursive structure encoded in the Bellman equation, in order to compute the value function.

Policy Evaluation

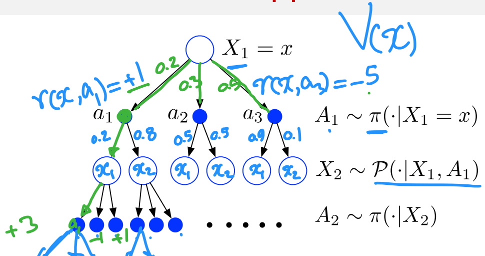
Problem Statement: Given an MDP $(\mathcal{X}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ and a policy π, we would like to compute V^π or Q^π.

$$V^\pi(x) = \mathbb{E}\left[\boxed{\sum_{t=1}^{\infty} \gamma^{t-1} R_t}\, | X_1 = x\right].$$

$V_T^\pi = E\left[\sum_{t=1}^{T}\right.$

G^π

Policy Evaluation: A Naive Approach



Idea: Expand the tree of all possible futures!

Example: the expected reward at time $t = 2$ is

$$\sum_{a,x',a'} \pi(a|x)\mathcal{P}(x'|x,a)\pi(a'|x')r(x',a').$$

Remark

- *This is inefficient. The size of the tree grows very fast.*
- *The sample-based approximate version of this idea works.*

Policy Evaluation: Linear System of Equations

Q: Can we improve the efficiency?

Key Idea: Benefit from the recursive structure of the value function

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

$$V(x) = r^{\pi}(x) + \gamma \sum_{x' \in \mathcal{X}} \mathcal{P}^{\pi}(x'|x) V(x'), \qquad \forall x \in \mathcal{X}$$

In the discrete state-action case:

- $n = |\mathcal{X}|$ equations
- $|\mathcal{X}|$ unknowns $(V(x_1), \ldots, V(x_n))$

Policy Evaluation: Linear System of Equations

We have n equations in the form of:

$$V(x) - \gamma \sum_{x' \in \mathcal{X}} \mathcal{P}^{\pi}(x'|x) V(x') = r^{\pi}(x),$$

More compactly in the matrix form:

$$(\mathbf{I} - \gamma \mathcal{P}^{\pi}) V = r^{\pi},$$

which is the same form of a generic linear system of equations:

$$A_{n \times n} x_{n \times 1} = b_{n \times 1}.$$

$O(n^{2.3?})$

$\lambda_i \geq 1-\gamma > 0$
$\gamma < 1$

Policy Evaluation: Linear System of Equations

Solving $A_{n\times n} x_{n\times 1} = b_{n\times 1}$:

- Compute A^{-1} and then calculate $A^{-1}b$.
- Better: Use various linear solvers.

$x = A^{-1} b$

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

Remark

To solve the control problem of finding V^*, we need to solve $V = T^* V$, i.e.,

$$V(x) = \max_{a\in\mathcal{A}} \left\{ r(x,a) + \gamma \sum_{x'\in\mathcal{X}} \mathcal{P}(x'|x,a) V(x') \right\}.$$

This is not a linear system of equations anymore!

Value Iteration

Value Iteration (PE)

Starting from $V_0 \in \mathcal{B}(\mathcal{X})$, we compute a sequence of $(V_k)_{k \geq 0}$ by

$$V_{k+1} \leftarrow T^{\pi} V_k [= r^{\pi} + \gamma \mathcal{P}^{\pi} V_k].$$

By the contraction property of the Bellman operator:

$$\lim_{k \to \infty} \|V_k - V^{\pi}\|_{\infty} = 0.$$

Remark

Similar procedure to compute Q^{π}, *i.e.,* $Q_{k+1} \leftarrow T^{\pi} Q_k$.

Value Iteration (Control)

$\to Q(x', \pi(x'))$

$$Q_{k+1}(x,a) \leftarrow r(x,a) + \gamma \sum_{x' \in X} P(x'|x,a) \max_{a'} Q(x',a')$$

$|X|$ $\quad |X|$ $\quad O(|A|)$

$\vdots$

$(x_{|X|})$

$|X| \times |A|$

$$(|X| \times |A|) \times |X| = |X|^2 \times |A|^2$$

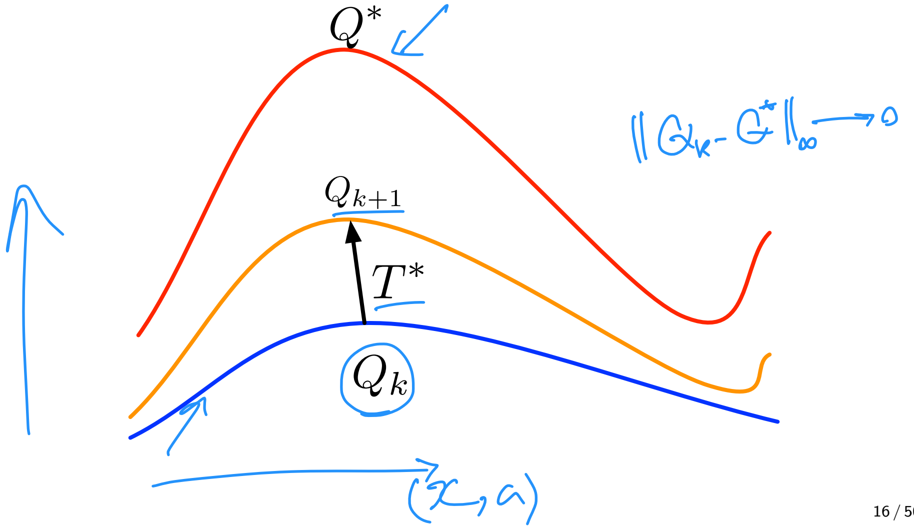
$$V_{k+1} \leftarrow T^* V_k,$$

$$Q_{k+1} \leftarrow T^* Q_k.$$

$|X| \times |A|$

By the contraction property of the Bellman optimality operator, it is guaranteed that $V_k \to V^*$ (or $Q_k \to Q^*$).

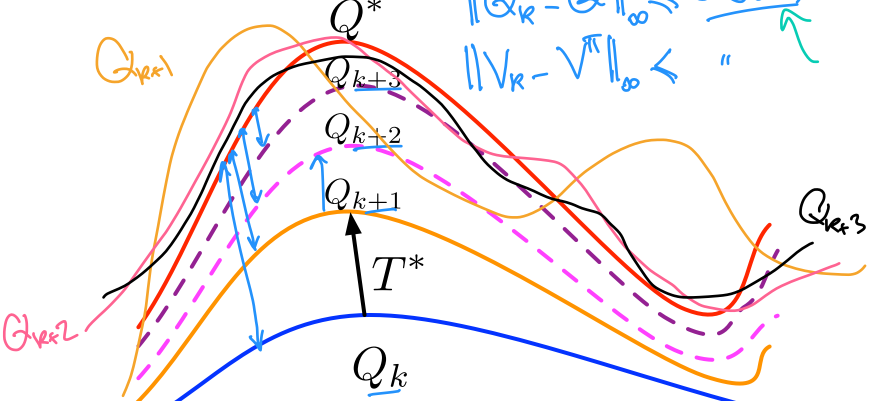
Value Iteration



Value Iteration

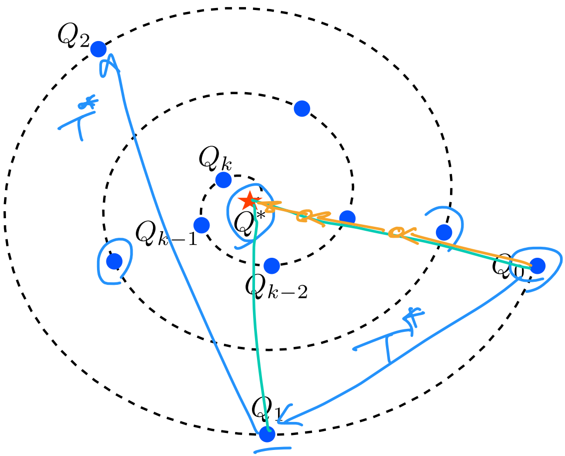
$\|Q_k - Q^*\|_\infty \leq O(\gamma^k)$

$\|V_k - V^\pi\|_\infty \leq$ "



Value Iteration

$|X| \times |A|$



Value Iteration

- The VI uses an existing approximation V_k of V^* (or V^π) to get a better approximation of V^* (or V^π). This idea is called bootstrapping in the in the RL literature.
  - Note: This is different from the bootstrapping method in statistics.
- VI is one of the fundamental algorithms for Planning.
- The current formulation of VI is a special case of an operator/matrix splitting-based formulation.
- Many RL algorithms are essentially the sample-based variants of VI too.

Mentimeter: $|X|$, $|A|$

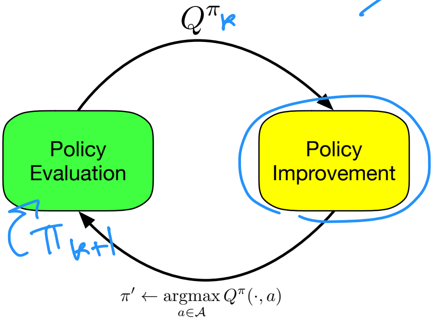
Policy Iteration

Policy Iteration

A different approach is based on the iterative application of the following two steps:

- (Policy Evaluation) Given a policy π_k, compute V^{π_k} (or Q^{π_k}).
- (Policy Improvement) Find a new policy π_{k+1} that is better than π_k, i.e., $V^{\pi_{k+1}} \geq V^{\pi_k}$ (with a strict inequality in at least one state, unless at convergence).

$>$



Policy Iteration

Q: How to perform Policy Evaluation and Policy Improvement?

- Policy Evaluation: This is clear. We can either solve a linear system of equations or even perform VI (PE) to compute the value of a policy π_k.
- Policy Improvement: Choose the greedy policy, i.e.,

$$\pi_{k+1}(x) \leftarrow \pi_g(x; Q^{\pi_k}) = \operatorname*{argmax}_{a \in \mathcal{A}} Q^{\pi_k}(x, a), \qquad \forall x \in \mathcal{X}.$$

Remark

The Policy Iteration (PI) algorithm refers to the specific case that we pick the new policy π_{k+1} *as* $\pi_g(Q^{\pi_k})$.

Why Greedy Policy for Policy Improvement? (Intuition)

Assume that at state x, we act according to $\pi_g(x; Q^{\pi_k})$, and afterwards, we follow π_k.
The value of this new policy is

$Q^{\pi}(x, a)$: First step a
Rest π

$$Q^{\pi_k}(x, \pi_g(x; Q^{\pi_k})) = Q^{\pi_k}(x, \operatorname*{argmax}_{a \in \mathcal{A}} Q^{\pi_k}(x, a)) = \max_{a \in \mathcal{A}} Q^{\pi_k}(x, a).$$

Comparing $\max_{a \in \mathcal{A}} Q^{\pi_k}(x, a)$ with $V^{\pi_k}(x) = Q^{\pi_k}(x, \pi_k(x))$, we see

(x,a) $=$

$$Q^{\pi_k}(x, \pi_g(x; Q^{\pi_k})) = \max_{a \in \mathcal{A}} Q^{\pi_k} \geq V^{\pi_k}(x).$$

So this new policy is equal to or better than π_k at state x.

Policy Iteration

greedy policy

Recall that:

- V^{π_k} is the unique fixed point of T^{π_k}.
- The greedy policy satisfies $T^{\pi_{k+1}} Q^{\pi_k} = T^* Q^{\pi_k}$.

We can summarize each iteration of the Policy Iteration algorithm:

- (Policy Evaluation) Given π_k, compute Q^{π_k}, i.e., find a Q that satisfies $Q = T^{\pi_k} Q$.
- (Policy Improvement) Obtain π_{k+1} as a policy that satisfies $T^{\pi_{k+1}} Q^{\pi_k} = T^* Q^{\pi_k}$.

Policy Iteration

Initialize π_0 arbitrarily
$k \leftarrow 0$
repeat
  $Q^{\pi_k} \leftarrow$ solution of $Q = T^{\pi_k} Q$ ▷ Policy Evaluation: compute Q^{π_k}
  $\pi_{k+1} \leftarrow$ policy s.t. $\boxed{T^{\pi_{k+1}} Q^{\pi_k} = T^* Q^{\pi_k}}$ ▷ Policy Improvement
  $k \leftarrow k + 1$
until $\pi_k = \pi_{k-1}$

$\pi_{k+1} \leftarrow \pi_g(\cdot\,; Q^{\pi_k})$

Approximate Policy Iteration

Approx. Value Iteration → $V_{k+1} = T V_k$
$V_{k+1} \simeq T V_k$

We also have approximate policy iteration algorithms too, where policy evaluation or improvement steps are performed approximately:

VI (PE) $k \to \infty$
$k = 10$

- $Q \approx T^{\pi_k} Q$
- $T^{\pi_{k+1}} Q^{\pi_k} \approx T^* Q^{\pi_k}$

We discuss this later when we get to function approximation.

Convergence of Policy Iteration

$\pi_k \longrightarrow \pi^*$

- The Policy Iteration algorithm converges to the optimal policy.
- For finite MDPs, the convergence happens in a finite number of iterations.

Policy Improvement Theorem

$\pi'(x) \stackrel{\text{def}}{=} \pi_g(Q^{\pi}) = \operatorname{argmax}_a Q^{\pi}(x,a)$

Theorem (Policy Improvement)

If for policies π and π', it holds that $T^{\pi'}Q^{\pi} = T^{*}Q^{\pi}$, we have that $Q^{\pi'} \geq Q^{\pi}$.

In other words, the greedy policy is a proper policy improvement step.

Policy Improvement Theorem (Proof)

We first show that $\boxed{T^{\pi'} Q^{\pi} \geq Q^{\pi}}$. Notice that $T^{\pi'} Q^{\pi} = T^{*} Q^{\pi}$, by the assumption.

We have $T^{*} Q^{\pi} \geq \boxed{T^{\pi} Q^{\pi} = Q^{\pi}}$ because for any $(x, a) \in \mathcal{X} \times \mathcal{A}$, it holds that

$$
\begin{aligned}
& r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x' | x, a) \max_{a' \in \mathcal{A}} Q^{\pi}(x', a') \geq \\
& r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x' | x, a) Q^{\pi}(x', \pi(x')) = T^{\pi} Q^{\pi}.
\end{aligned}
$$

Therefore, $T^{\pi'} Q^{\pi} = T^{*} Q^{\pi} \geq \underbrace{T^{\pi} Q^{\pi} = Q^{\pi}}_{\text{Bellman Eq}}$.

$T^{\pi'} Q^{\pi} \geq Q^{\pi}$

Policy Improvement Theorem (Proof)

The second step is to use $T^{\pi'}Q^{\pi} \geq Q^{\pi}$ to show that $Q^{\pi'} \geq Q^{\pi}$. Apply $T^{\pi'}$ to both sides of $T^{\pi'}Q^{\pi} \geq Q^{\pi}$, and use the monotonicity property of the Bellman operator to conclude

$$T^{\pi'}(T^{\pi'}Q^{\pi}) \geq T^{\pi'}Q^{\pi} = T^{*}Q^{\pi} \geq Q^{\pi}.$$

So we also have $(T^{\pi'})^2 Q^{\pi} \geq Q^{\pi}$.
By repeating this argument, we get that for any $m \geq 1$,

$$(T^{\pi'})^m Q^{\pi} \geq T^{*}Q^{\pi} \geq Q^{\pi}. \tag{1}$$

$$Q^{\pi'} \overset{?}{\geq} Q^{\pi}$$

Policy Improvement Theorem (Proof)

$$\boxed{(T^{\pi'})^m Q^\pi \geq T^* Q^\pi \geq Q^\pi.}$$

Take the limit of $m \to \infty$.
Because of the contraction property of the Bellman operator $T^{\pi'}$:

Q_0

$$\lim_{m\to\infty} (T^{\pi'})^m Q^\pi = Q^{\pi'}. \quad (2)$$

By combining (1) and (2), we get that

$$Q^{\pi'} = \lim_{m\to\infty} (T^{\pi'})^m Q^\pi \geq T^* Q^\pi \geq Q^\pi, \quad (3)$$

$Q^{\pi'} \geq Q^\pi$

Convergence of Policy Iteration

- The Policy Improvement theorem shows that if we are given π_k, the new policy π_{k+1} is at least as good as the previous one.
- We can show that the PI algorithm converges to an optimal policy. We shall prove this.
- If $|\mathcal{X} \times \mathcal{A}| < \infty$, this happens in a finite number of iterations.

Convergence of Policy Iteration

Theorem (Convergence of the Policy Iteration Algorithm)

Let $(\pi_k)_{k\geq 0}$ be the sequence generated by the PI algorithm.

- *For all k, we have that $V^{\pi_{k+1}} \geq V^{\pi_k}$, with equality if and only if $V^{\pi_k} = V^*$.*
- $\lim_{k\to\infty} \|V^{\pi_k} - V^*\|_\infty = 0.$
- *If the set of policies is finite, the PI algorithm converges in a finite number of iterations.*

Remark

We follow the line of proof of Proposition 2.4.1 of Bertsekas 2018.

Convergence of Policy Iteration (Proof)

The basic idea behind the proof is that either we can strictly improve the policy, or if we cannot, we are already at the optimal policy.

Convergence of Policy Iteration (Proof)

Proof of $V^{\pi_{k+1}} \geq V^{\pi_k}$:
By the Policy Improvement Theorem (Theorem 1), we have that $V^{\pi_{k+1}} \geq V^{\pi_k}$.
Proof of $V^{\pi_{k+1}} = V^{\pi_k} \Rightarrow V^{\pi_k} = V^*$:
Suppose that instead of a strict inequality, we have an equality of

$$V^{\pi_{k+1}} = V^{\pi_k}.$$

Apply $T^{\pi_{k+1}}$ to both side to get

$$T^{\pi_{k+1}} V^{\pi_{k+1}} = T^{\pi_{k+1}} V^{\pi_k}.$$

As $T^{\pi_{k+1}} V^{\pi_k} = T^* V^{\pi_k}$ by the definition of the PI algorithm, we get that

$$T^{\pi_{k+1}} V^{\pi_{k+1}} = T^{\pi_{k+1}} V^{\pi_k} = T^* V^{\pi_k} = T^* V^{\pi_{k+1}},$$

where in the last step we used $V^{\pi_{k+1}} = V^{\pi_k}$ again.

Convergence of Policy Iteration (Proof)

By these equalities, we have

$$\boxed{T^{\pi_{k+1}} V^{\pi_{k+1}}} = T^* V^{\pi_{k+1}}.$$

As $V^{\pi_{k+1}}$ is the value function of π_{k+1}, it satisfies the Bellman equation $T^{\pi_{k+1}} V^{\pi_{k+1}} = V^{\pi_{k+1}}$. Therefore, we also have

$$V^{\pi_{k+1}} = T^* V^{\pi_{k+1}}.$$

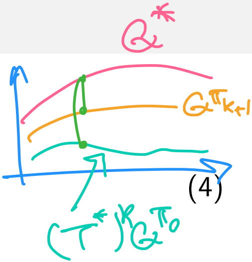
This means that $V^{\pi_{k+1}}$ is a fixed point of T^*.
But the fixed point of T^* is unique and is equal to V^*.
So we must have that

$$V^{\pi_{k+1}} = V^*.$$

Convergence of Policy Iteration (Proof)

Proof of $V^{\pi_k} = V^* \Rightarrow V^{\pi_{k+1}} = V^{\pi_k}$:
If $V^{\pi_k} = V^*$, then π_k is an optimal policy. The greedy policy of $V^{\pi_k} = V^*$ is still an optimal policy, hence $V^{\pi_{k+1}} = V^* = V^{\pi_k}$.

Convergence of Policy Iteration (Proof)



Proof of $\lim_{k\to\infty} \|V^{\pi_k} - V^*\|_\infty = 0$.
To prove the convergence, recall from (3) that

$$Q^{\pi_{k+1}} \geq T^* Q^{\pi_k} \geq Q^{\pi_k}. \tag{4}$$

By induction,

$$Q^{\pi_{k+1}} \geq T^* Q^{\pi_k} \geq T^*(T^* Q^{\pi_{k-1}}) \geq \cdots \geq (T^*)^k Q^{\pi_0}.$$

By the definition of the optimal policy, we have $\boxed{Q^\pi \leq Q^*}$ for any π, including all π_k generated during the iterations of the PI algorithm. So $Q^{\pi_{k+1}}$ is sandwiched between Q^* and $(T^*)^k Q^{\pi_0}$, i.e.,

$$Q^* \geq Q^{\pi_{k+1}} \geq (T^*)^k Q^{\pi_0}.$$

This entails that $\|Q^{\pi_{k+1}} - Q^*\|_\infty \leq \left\|(T^*)^k Q^{\pi_0} - Q^*\right\|_\infty$.

Convergence of Policy Iteration (Proof)

We show that the RHS of $\|Q^{\pi_{k+1}} - Q^*\|_\infty \leq \left\|(T^*)^k Q^{\pi_0} - Q^*\right\|_\infty$ converges to zero.

By the contraction property of the Bellman optimality operator, we have that

$$\lim_{k\to\infty} \left\|(T^*)^k Q^{\pi_0} - Q^*\right\|_\infty = 0.$$

Therefore,

$$\lim_{k\to\infty} \|Q^{\pi_k} - Q^*\|_\infty = 0.$$

This implies the convergence of V^{π_k} too.

Convergence of Policy Iteration (Proof)

Proof of finite convergence:
If the number of policies is finite, the number of times (4) can be a strict inequality is going to be finite too.

Convergence of Policy Iteration: Some Remarks

a_1 a_2 1 a_1 a_2 2 a_1 a_2 3

$2 \times 2 \times 2 = 2^3$ $\quad |A|^{|\mathcal{X}|}$

- The PI algorithm converges to the optimal policy in a finite number of iterations whenever the number of policies is finite.
- If the state space $\mathcal{X}$ and the action space $\mathcal{A}$ are finite, the number of policies are finite and is $|\mathcal{A}|^{|\mathcal{X}|}$.
- Even though finite, this can be very large.
  - Example: A 10×10 grid world problem with 4 actions at each state has $4^{100} \approx 1.6 \times 10^{60}$ possible policies.
- In practice, PI converges much faster.
- This suggest that the previous analysis might be crude.

a_1 a_2

$\pi(a_1|x_1) = P_1 \rightarrow \pi(a_2|x_1) = 1 - P_1$

$P_1 \in [0,1] \longrightarrow$ Deter $\rightarrow P_1 = 0, 1$

Fast Convergence of Policy Iteration

It can be shown that the PI algorithm converges in

$$\frac{400}{0.1} = 4000$$

100 × 4

$$O\left(\frac{|\mathcal{X}||\mathcal{A}|}{\underline{1-\gamma}}\log\left(\frac{1}{\underline{1-\gamma}}\right)\right)$$

1 - 0.9

iterations.

The proof is in the Foundations of Reinforcement Learning [Farahmand, 2025].

This is a significant quantitative improvement over the previous result.

Remark

This is a relatively recent result, which in various forms have been proven by Ye [2011]; Hansen et al. [2013]; Scherrer [2016].

Linear Programming

Linear Programming for Finding V^*

$V \in \mathcal{B}(\mathcal{X})$

We can find V^* by solving a Linear Program (LP) too.
Consider the set of all V that satisfy $V \geq T^*V$, i.e.,

$$C = \{ V : V \geq T^*V \}.$$

$V^* = T^* V^*$
$\geq V$
$V^* \in C$

Interesting property: For any $V \in C$, we have

$$V \geq T^*V \Rightarrow T^*V \geq T^*(T^*V) = (T^*)^2 V.$$

Repeating this argument, we get that for any $m \geq 1$,

$$V \geq (T^*)^m V.$$

Linear Programming for Finding V^*

$\checkmark\ V^\pi \leq V^* \quad \forall \pi$ $\qquad V^* = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$

$\exists V \in B(X),\ V \geq V^*$ $\qquad V = \begin{bmatrix} 20 \\ 6 \end{bmatrix}, \begin{bmatrix} 2 \\ 9 \end{bmatrix}$

From

$$V \geq (T^*)^m V.$$

$\forall V \in C,$ $V \geq V^*$

we get that

$$V \geq \lim_{m \to \infty} (T^*)^m V = V^*.$$

Interpretation:

- Any $V \in C$ is lower bounded by V^*.
- (OR) V^* is the function in C that is smaller or equal to any other function in C (pointwise sense).

$\sum V(x) = 9$

Linear Programming for Finding V^*

$V = [1, 5, 3]$ $\sum V^*(x) = 7$ $\mu = 1$

$V^* = [1, 5, 1]$

Choose a strictly positive vector $\mu > 0$ with the dimension of $\mathcal{X}$.
Solve

$$\min_{V \in C} \mu^\top V, = \sum_{x \in \mathcal{X}} \mu(x) V(x)$$

$V \leftarrow \min_{V \in C} \sum_x V(x)$

Can be written as

$$\begin{aligned} \min_V \quad & \mu^\top V, \\ \text{s.t.} \quad & V(x) \geq (T^* V)(x), \qquad \forall x \in \mathcal{X}. \end{aligned}$$

Linear objective; nonlinear constraints.

Linear Programming for Finding V^*

Each nonlinear constraint:

$V \geqslant TV$

$$V(x) \geq \max_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \sum_y \mathcal{P}(y|x,a) V(y) \right\}$$

is equivalent to

$$V(x) \geq r(x,a) + \gamma \sum_y \mathcal{P}(y|x,a) V(y), \quad \boxed{\forall a \in \mathcal{A}.}$$

Linear Programming for Finding V^*

$$\min_{x \in \mathbb{R}^d} c^\top x \qquad c \in \mathbb{R}^d$$
$$\text{s.t.} \quad A_{d\times d}\, x \le b_{d\times 1}$$

$$V^* = \arg\min_V \quad \mu^\top V,$$
$$\text{s.t.} \quad V(x) \geq r(x,a) + \gamma \sum_y \mathcal{P}(y|x,a) V(y), \qquad \forall (x,a) \in \mathcal{X} \times \mathcal{A}.$$

This is a linear program with $|\mathcal{X} \times \mathcal{A}|$ constraints.

Summary

ToDo: - Matchmaking for HWs & Proj

- Three methods for computing the optimal value function
  - Value Iteration
  - Policy Iteration
  - Linear Programming
- Established convergence of VI and PI
- These methods have variants for the RL setting.

References

Dimitri P. Bertsekas. *Abstract dynamic programming*. Athena Scientific Belmont, 2nd edition, 2018.

Amir-massoud Farahmand. *Foundations of Reinforcement Learning (draft)*. 2025.

Thomas Dueholm Hansen, Peter Bro Miltersen, and Uri Zwick. Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. *Journal of the ACM (JACM)*, 60(1):1–16, 2013.

Bruno Scherrer. Improved and generalized upper bounds on the complexity of policy iteration. *Mathematics of Operations Research*, 41(3):758–774, 2016.

Yinyu Ye. The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate. *Mathematics of Operations Research*, 36(4): 593–603, 2011.