2021-10-27

Value iteration

Solving infinite horizon problems

Cathy Wu

1.041/1.200/11.544 Transportation: Foundations and Methods

References

1. With many slides adapted from Alessandro Lazaric and Matteo Pirotta.
2. Dimitri P. Bertsekas. Dynamic Programming and Optimal Control. Volume 2. 4th Edition. (2012). Chapters 1-2: Discounted Problems.
3. R. E. Bellman. Dynamic Programming. Princeton University Press, Princeton, N.J., 1957.

Outline

1. **Dynamic programming iteration for infinite horizon problems**
2. Value iteration
3. Policy iteration

Remark

The dynamic programming iteration is valid for infinite horizon problems, too!

$$V_k^*(s) = \max_a r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} [V_{k+1}^*(s')]$$

Where $V_k^*(s) = \max_\pi \mathbb{E}\left[\sum_{k'=0}^{\infty} \gamma^{k'} r\left(s_{k'+k}, \pi(s_{k'+k})\right) \middle| \pi, s_k = s\right]$

Proof: Dynamic programming (infinite horizon)

Consider any step k:

$$V_k^*(s) = \max_\pi \mathbb{E}\left[\sum_{k'=0}^{\infty} \gamma^{k'} r\left(s_{k'+k}, \pi(s_{k'+k})\right) \middle| \pi, s_k = s\right]$$

Decouple the first term of the sum from the remainder of the sum.

$$= \max_\pi r\left(s, \pi(s)\right) + \mathbb{E}\left[\sum_{k'=1}^{\infty} \gamma^{k'} r\left(s_{k'+k}, \pi(s_{k'+k})\right) \middle| \pi, s_k = s\right]$$

Expand expectation and pull out a γ factor

$$= \max_\pi r\left(s, \pi(s)\right) + \gamma \sum_{s'} P(s_{k+1} = s' | s_k = s; \pi(s))$$
$$\mathbb{E}\left[\sum_{k'=1}^{\infty} \gamma^{k'-1} r\left(s_{k'+k}, \pi(s_{k'+k})\right) \middle| \pi, s_{k+1} = s'\right]$$

Proof: Dynamic programming (infinite horizon)

$$= \max_{\pi} r(s, \pi(s)) + \gamma \sum_{s'} P(s_{k+1} = s' | s_k = s; \pi(s))$$
$$\mathbb{E}\left[\sum_{k'=1}^{\infty} \gamma^{k'-1} r\left(s_{k'+k}, \pi(s_{k'+k})\right) \middle| \pi, s_{k+1} = s'\right]$$

Decomposition of policy $\pi = (a, \pi')$, rewrite expectation, and change of variables $k'' = k' - 1$.

$$= \max_{(a,\pi')} r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, a)}$$
$$\left[\mathbb{E}\left[\sum_{k''=0}^{\infty} \gamma^{k''} r\left(s_{k''+1+k}, \pi'(s_{k''+1+k})\right) \middle| \pi', s_{k+1} = s'\right]\right]$$

Proof: Dynamic programming (infinite horizon)

$$= \max_{(a,\pi')} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[\mathbb{E}\left[\sum_{k''=0}^{\infty} \gamma^{k''} r\left(s_{k''+1+k}, \pi'(s_{k''+1+k})\right) \middle| \pi', s_{k+1} = s' \right] \right]$$

- Basic inequality

$$\max_{\pi'} \sum_{s'} P(s'|s,a) V^{\pi'}(s') \le \sum_{s'} P(s'|s,a) \max_{\pi'} V^{\pi'}(s')$$

- Let $\bar{\pi}(s') = \arg\max_{\pi'} V^{\pi'}(s')$. Then,

$$\sum_{s'} P(s'|s,a) \max_{\pi'} V^{\pi'}(s') = \sum_{s'} P(s'|s,a) V^{\bar{\pi}}(s') \le \max_{\pi'} \sum_{s'} P(s'|s,a) V^{\pi'}(s')$$

- Thus, the max and expectation can be exchanged:

$$= \max_{a} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[\max_{\pi'} \mathbb{E}\left[\sum_{k''=0}^{\infty} \gamma^{k''} r\left(s_{k''+1+k}, \pi'(s_{k''+1+k})\right) \middle| \pi', s_{k+1} = s' \right] \right]$$

Proof: Dynamic programming (infinite horizon)

$$= \max_{a} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}$$
$$\left[\max_{\pi'} \mathbb{E}\left[\sum_{k''=0}^{\infty} \gamma^{k''} r\left(s_{k''+1+k}, \pi'(s_{k''+1+k})\right) \middle| \pi', s_{k+1} = s'\right]\right]$$

Follows from definition of optimal k-stage value function.

$$V_k^*(s) = \max_{a} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} [V_{k+1}^*(s')]$$

■

Next time:

- But that was the induction step. What about the base case?
- $V_k^*(s)$ vs $V_{k+1}^*(s)$ vs $V^*(s)$
- Optimal Bellman equation (no k subscripts!):

$$V^*(s) = \max_{a \in A} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} V^*(s')$$

How to solve infinite horizon problems?

Recall:

$$V_k^*(s) = \max_a r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} [V_{k+1}^*(s')]$$

Where $V_k^*(s) = \max_\pi \mathbb{E}\left[\sum_{k'=0}^{\infty} \gamma^{k'} r\left(s_{k'+k}, \pi(s_{k'+k})\right) \middle| \pi, s_k = s\right]$

Outline

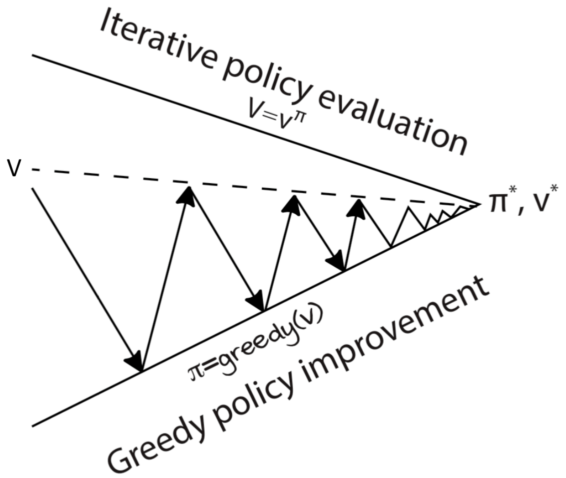
1. Dynamic programming iteration for infinite horizon problems
2. **Value iteration**
   a. Bellman equation & operators
   b. Convergence analysis
   c. Example: grid world parking
   d. Computational demo
3. Policy iteration

Value iteration algorithm

1. Let $V_0(s)$ be any function $V_0 : S \to \mathbb{R}$. [Note: not stage 0, but iteration 0.]
2. Apply the principle of optimality so that given V_i at iteration i, we compute
$$V_{i+1}(s) = \mathcal{T}V_i(s) = \max_{a \in A} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} [V_i(s')] \quad \text{for all } s$$
3. Terminate when V_i stops improving, e.g. when $\max_s |V_{i+1}(s) - V_i(s)|$ is small.
4. Return the greedy policy: $\pi_K(s) = \arg\max_{a \in A} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} V_K(s')$

☞ A key result: $V_i \to V^*$, as $i \to \infty$.

☞ Helpful properties
- Markov process
- Contraction in max-norm
- Cauchy sequences
- Fixed point



Adapted from Morales, Grokking Deep Reinforcement Learning, 2020.

Value iteration algorithm

1. Let $V_0(s)$ be any function $V_0 : S \to \mathbb{R}$. [Note: not stage 0, but iteration 0.]
2. Apply the principle of optimality so that given V_i at iteration i, we compute
$$V_{i+1}(s) = \mathcal{T}V_i(s) = \max_{a \in A} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} [V_i(s')] \quad \text{for all } s$$
3. Terminate when V_i stops improving, e.g. when $\max_s |V_{i+1}(s) - V_i(s)|$ is small.
4. Return the greedy policy: $\pi_K(s) = \arg\max_{a \in A} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} V_K(s')$

Definition (Optimal Bellman operator)

For any $W \in \mathbb{R}^{|S|}$, the optimal Bellman operator is defined as
$$\mathcal{T}W(s) = \max_{a \in A} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} W(s') \quad \text{for all } s$$

☞ Then we can write the algorithm step 2 concisely:
$$V_{i+1}(s) = \mathcal{T}V_i(s) \quad \text{for all } s$$

Key question: Does $V_i \to V^*$?

Properties of Bellman Operators

Proposition

1. Contraction in L_∞-norm: for any $W_1, W_2 \in \mathbb{R}^N$

$$\|\mathcal{T}W_1 - \mathcal{T}W_2\|_\infty \le \gamma \|W_1 - W_2\|_\infty$$

- Norms give a size for a multi-dimensional object.
  L_p-norms for a vector $v \in \mathbb{R}^d$:

$$\|v\|_p = \left(\sum_{i=1}^{d} |v_i|^p\right)^{\frac{1}{p}}$$

Most common: L_2, L_1, L_∞. L_∞ is also called the max norm for good reason:

$$\|v\|_\infty = \max_{1\le i\le d} |v_i|$$

Properties of Bellman Operators

Proposition

1. Contraction in L_∞-norm: for any $W_1, W_2 \in \mathbb{R}^N$

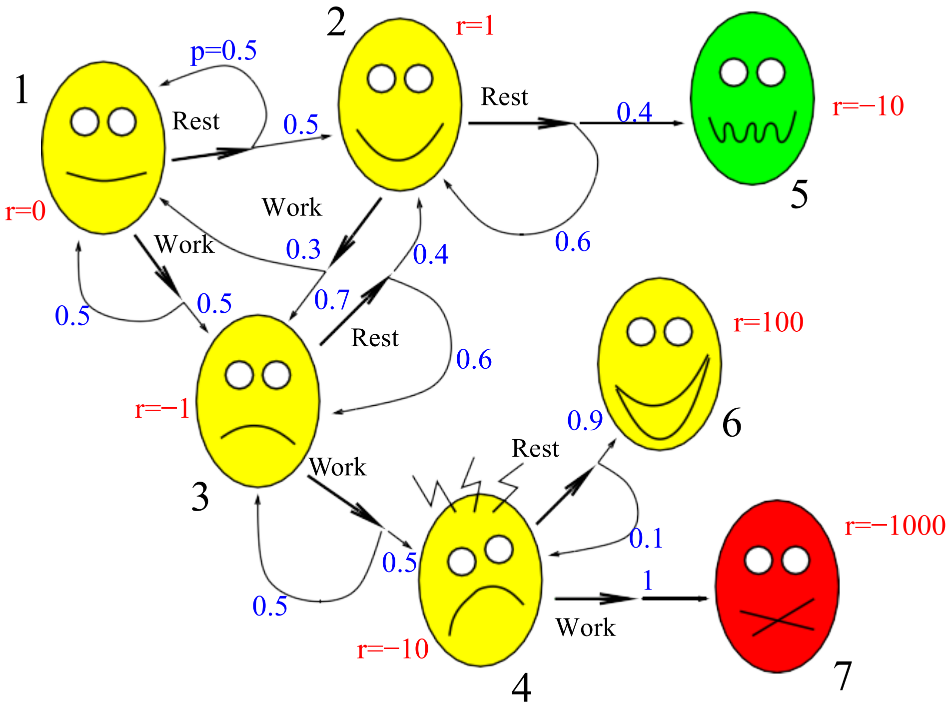
$$\|\mathcal{T}W_1 - \mathcal{T}W_2\|_\infty \leq \gamma \|W_1 - W_2\|_\infty$$

For instance, how big is the following vector?

$$x = \begin{bmatrix} 1 \\ 0 \\ 5 \\ 3 \\ -10 \end{bmatrix}$$

The student dilemma

- *Model*: all the transitions are Markov, states s_5, s_6, s_7 are terminal.
- *Setting*: infinite horizon with terminal states.
- *Objective*: find the policy that maximizes the expected sum of rewards before achieving a terminal state.
- *Notice*: Not a discounted infinite horizon setting. But the Bellman equations hold unchanged.



The Optimal Bellman Equation

Bellman's Principle of Optimality (Bellman (1957)):

"An optimal policy has the property that, whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."

The Optimal Bellman Equation

Theorem (Optimal Bellman Equation)

The optimal value function V^* (i.e. $V^* = \max_\pi V^\pi$) is the solution to the optimal Bellman equation:

$$V^*(s) = \max_{a \in A} \left[r(s,a) + \gamma \sum_{s'} p\,(s'|s,a)\, V^*(s') \right]$$

And any optimal policy is such that:

$$\pi^*(a|s) \geq 0 \Leftrightarrow a \in \arg\max_{a' \in A} \left[r(s,a') + \gamma \sum_{s'} p\,(s'|s,a)\, V^*(s') \right]$$

Or, for short: $V^* = \mathcal{T} V^*$

☞ There is always a deterministic policy (see: Puterman, 2005, Chapter 7)

Proof: The Optimal Bellman Equation

For any policy $\pi = (a, \pi')$ (possibly non-stationary),

$$V^*(s) = \max_{\pi} \mathbb{E}\left[\sum_{t\geq 0}^{\infty} \gamma^t r\big(s_t, \pi(s_t)\big) \mid s_0 = s; \pi\right]$$

[value function]

$$= \max_{(a,\pi')}\left[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V^{\pi'}(s')\right]$$

[Markov property & change of "time"]

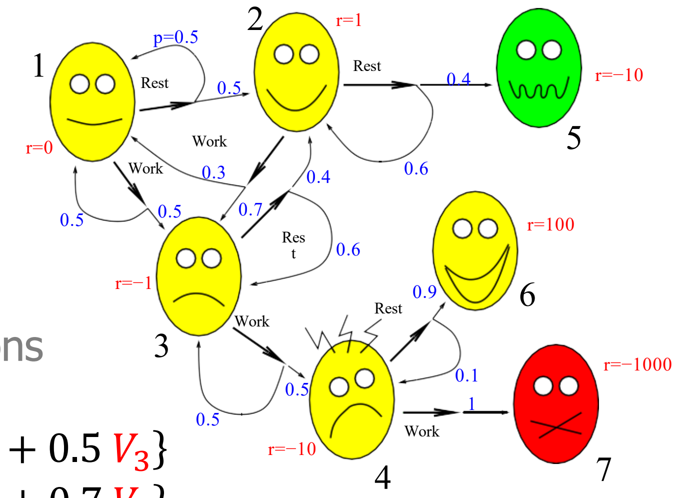
$$= \max_{a}\left[r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{\pi'} V^{\pi'}(s')\right]$$

$$= \max_{a}\left[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V^*(s')\right]$$

[value function]

The student dilemma

$$V^*(s) = \max_{a \in A} \left[r(x,a) + \gamma \sum_y p\,(y|x,a)\, V^*(y) \right]$$



System of equations

$$\begin{cases} V_1 = & \max\{0 + 0.5\,V_1 + 0.5\,V_2; 0 + 0.5\,V_1 + 0.5\,V_3\} \\ V_2 = & \max\{0 + 0.4\,V_5 + 0.6\,V_2; 0 + 0.3\,V_1 + 0.7\,V_3\} \\ V_3 = & \max\{-1 + 0.4\,V_2 + 0.6\,V_3; -1 + 0.5\,V_4 + 0.5\,V_3\} \\ V_4 = & \max\{-10 + 0.9\,V_6 + 0.1\,V_4; -10 + V_7\} \\ V_5 = & -10 \\ V_6 = & 100 \\ V_7 = & -1000 \end{cases}$$

Discuss: How to solve this system of equations?

System of Equations

The optimal Bellman equation:

$$V^*(s) = \max_{a \in A}\left[r(s,a) + \gamma \sum_{s'} p\,(s'|s,a)\,V^*(s')\right]$$

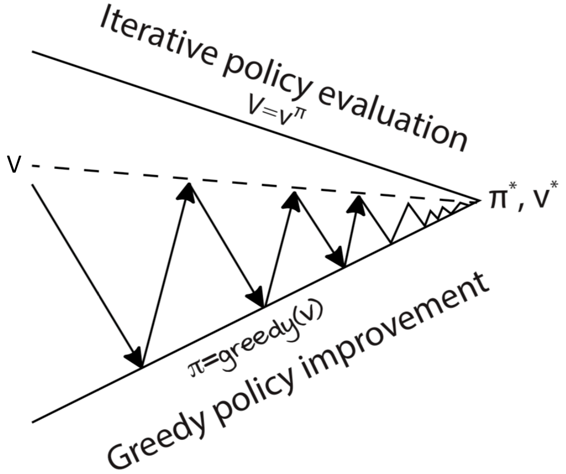
Is a non-linear system of equations with N unknowns and N non-linear constraints (i.e. the max operator).

Value iteration algorithm

1. Let $V_0(s)$ be any function $V_0: S \to \mathbb{R}$. [Note: not stage 0, but iteration 0.]
2. Apply the principle of optimality so that given V_i at iteration i, we compute
$$V_{i+1}(s) = \mathcal{T}V_i(s) = \max_{a \in A} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} [V_i(s')] \quad \text{for all } s$$
3. Terminate when V_i stops improving, e.g. when $\max_s |V_{i+1}(s) - V_i(s)|$ is small.
4. Return the greedy policy: $\pi_K(s) = \arg\max_{a \in A} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} V_K(s')$

☞ A key result: $V_i \to V^*$, as $i \to \infty$.

☞ Helpful properties
- Markov process
- Contraction in max-norm
- Cauchy sequences
- Fixed point



Adapted from Morales, Grokking Deep Reinforcement Learning, 2020.

Properties of Bellman Operators

Proposition

1. Contraction in L_∞-norm: for any $W_1, W_2, \in \mathbb{R}^N$

$$\|\mathcal{T}W_1 - \mathcal{T}W_2\|_\infty \leq \gamma \|W_1 - W_2\|_\infty$$

2. Fixed point: V^* is the unique fixed point of $\mathcal{T}$, i.e. $V^* = \mathcal{T}V^*$.

Proof: value iteration

- From contraction property of $\mathcal{T}$, $V_k = \mathcal{T}\mathrm{V}_{\mathrm{k}-1}$, and optimal value function $V^* = \mathcal{T}V^*$:

$$\|V^* - V_{k+1}\|_\infty$$

$$= \|\mathcal{T}V^* - \mathcal{T}V_k\|_\infty \quad \text{[value iteration and optimal Bellman eq.]}$$

$$\leq \gamma \|V^* - V_k\|_\infty \quad \text{[contraction]}$$

$$\leq \gamma^{k+1} \|V^* - V_0\|_\infty \quad \text{[recursion]}$$

$$\to 0$$

$$V_k \to V^* \quad \text{[fixed point]}$$

Properties of Bellman Operators

Proposition

1. Contraction in L_∞-norm: for any $W_1, W_2, \in \mathbb{R}^N$

$$\|\mathcal{T}W_1 - \mathcal{T}W_2\|_\infty \leq \gamma \|W_1 - W_2\|_\infty$$

2. Fixed point: V^* is the unique fixed point of $\mathcal{T}$, i.e. $V^* = \mathcal{T}V^*$.

Proof: value iteration

- Convergence rate. Let $\epsilon > 0$ and $\|r\|_\infty \leq r_{\max}$, then after at most

$$\|V^* - V_{k+1}\|_\infty \leq \gamma^{k+1} \|V^* - V_0\|_\infty < \epsilon \implies K \geq \frac{\log\left(\frac{r_{\max}}{(1-\gamma)\epsilon}\right)}{\log(\frac{1}{\gamma})}$$

Proof: Contraction of the Bellman Operator

For any $s \in S$

$$|\mathcal{T}W_1(s) - \mathcal{T}W_2(s)|$$

$$= \left| \max_a \left[r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, W_1(s') \right] - \max_{a'} \left[r(s,a') + \gamma \sum_{s'} p(s'|s,a')\, W_2(s') \right] \right|$$

$$\leq \max_a \left| \left[r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, W_1(s') \right] - \left[r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, W_2(s') \right] \right|$$

$$= \gamma \max_a \sum_{s'} p(s'|s,a)\, |W_1(s') - W_2(s')|$$

$$\leq \gamma \|W_1 - W_2\|_\infty \max_a \sum_{s'} p(s'|s,a) \;= \gamma \|W_1 - W_2\|_\infty$$

$\blacksquare$

$$\max_x f(x) - \max_{x'} g(x') \leq \max_x (f(x) - g(x))$$

Value Iteration: the Complexity

Time complexity

- Each iteration takes on the order of S^2A operations.

$$V_{k+1}(s) = \mathcal{T}V_k(s) = \max_{a\in A}\left[r(s,a) + \gamma \sum_{s'} p(s'|s,a)V_k(s')\right]$$

- The computation of the greedy policy takes on the order of S^2A operations.

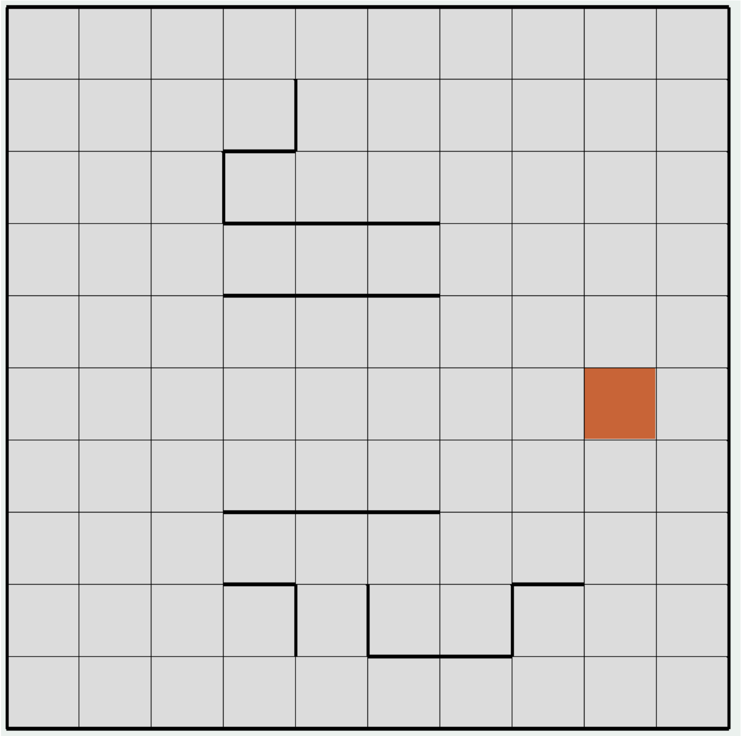
$$\pi_K(s) \in \arg\max_{a\in A}\left[r(s,a) + \gamma \sum_{s'} p(s'|s,a)V_K(s')\right]$$

- Total time complexity on the order of KS^2A.

Space complexity

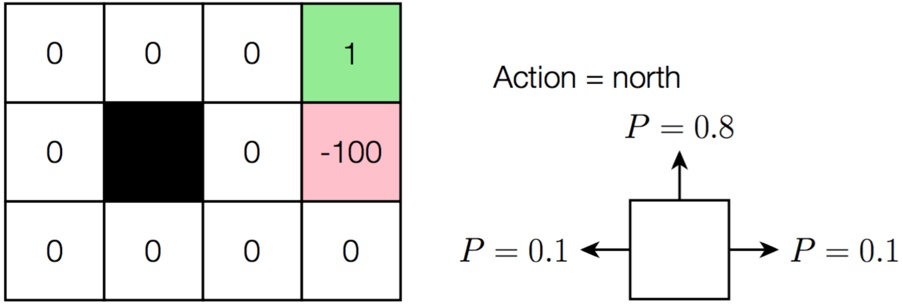
- Storing the MDP: dynamics on the order of S^2A and reward on the order of SA.
- Storing the value function and the optimal policy on the order of S.

The Grid-World Problem



Example: Winter parking (with ice and potholes)

- Simple grid world with a *goal state* (green, desired parking spot) with reward (+1), a *"bad state"* (red, pothole) with reward (-100), and all other states neural (+0).
- *Omnidirectional vehicle (agent)* can head in any direction. Actions move in the desired direction with probably 0.8, in one of the perpendicular directions with.
- Taking an action that would bump into a wall leaves agent where it is.



[Source: adapted from Kolter, 2016]

Example: value iteration

Running value iteration with $\gamma = 0.9$

0	0	0	1
0		0	-100
0	0	0	0

Original reward function

(a)

Recall value iteration algorithm:

$$\mathrm{V}_{\mathrm{i+1}}(s) = \max_{a \in A} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} V_i(s') \quad \text{for all } s$$

Let's arbitrarily initialize V_0 as the reward function, since it can be any function.

Example update (red state):

$$\begin{aligned}\mathrm{V}_1(\text{red}) = -100 + \gamma \max\{ \ & 0.8V_0(\text{green}) + 0.1V_0(\text{red}) + 0, && \text{[up]} \\ & 0 + 0.1V_0(\text{red}) + 0, && \text{[down]} \\ & 0 + 0.1V_0(\text{green}) + 0, && \text{[left]} \\ & 0.8V_0(\text{red}) + 0.1V_0(\text{green}) + 1 \ \} && \text{[right]}\end{aligned}$$

$= -100 + 0.9(0.1 * 1) = -99.91$ [best: go left]

Example: value iteration

Running value iteration with $\gamma = 0.9$

0	0	0	1
0		0	-100
0	0	0	0

Original reward function

(a)

Recall value iteration algorithm:

$$V_{i+1}(s) = \max_{a \in A} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} V_i(s') \quad \text{for all } s$$

Let's arbitrarily initialize V_0 as the reward function, since it can be any function.

Wu

Example update (green state):

$$V_1(\text{red}) = 1 + \gamma \max\{ \begin{array}{ll} 0.8V_0(\text{green}) + 0.1V_0(\text{green}), & \text{[up]} \\ 0.8V_0(\text{red}) + 0.1V_0(\text{green}), & \text{[down]} \\ 0 + 0.1V_0(\text{green}) + 0.1V_0(\text{red}), & \text{[left]} \\ 0.8V_0(\text{red}) + 0.1V_0(\text{green}) + 0 \} & \text{[right]} \end{array}$$

$= 1 + 0.9(0.9 * 1) = 1.81$ [best: go up]

Example: value iteration

Running value iteration with $\gamma = 0.9$

0	0	0	1
0		0	-100
0	0	0	0

(a)

Original reward function

Running value iteration with $\gamma = 0.9$

0	0	0.72	1.81
0		0	-99.91
0	0	0	0

(b)

$\hat{V}$ at one iteration

Recall value iteration algorithm:

$$V_{i+1}(s) = \max_{a \in A} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} V_i(s') \quad \text{for all } s$$

Let's arbitrarily initialize V_0 as the reward function, since it can be any function.

Need to also do this for all the "unnamed" states, too.

Wu

Example: value iteration

Running value iteration with $\gamma = 0.9$

0	0	0	1
0		0	-100
0	0	0	0

Original reward function

(a)

Running value iteration with $\gamma = 0.9$

0	0	0.72	1.81
0		0	-99.91
0	0	0	0

$\hat{V}$ at one iteration

(b)

Running value iteration with $\gamma = 0.9$

0.809	1.598	2.475	3.745
0.268		0.302	-99.59
0	0.034	0.122	0.004

$\hat{V}$ at five iterations

(c)

Running value iteration with $\gamma = 0.9$

2.686	3.527	4.402	5.812
2.021		1.095	-98.82
1.390	0.903	0.738	0.123

$\hat{V}$ at 10 iterations

(d)

Running value iteration with $\gamma = 0.9$

5.470	6.313	7.190	8.669
4.802		3.347	-96.67
4.161	3.654	3.222	1.526

$\hat{V}$ at 1000 iterations

(e)

Running value iteration with $\gamma = 0.9$

→	→	→	↑
↑		←	←
↑	←	←	↓

Resulting policy after 1000 iterations

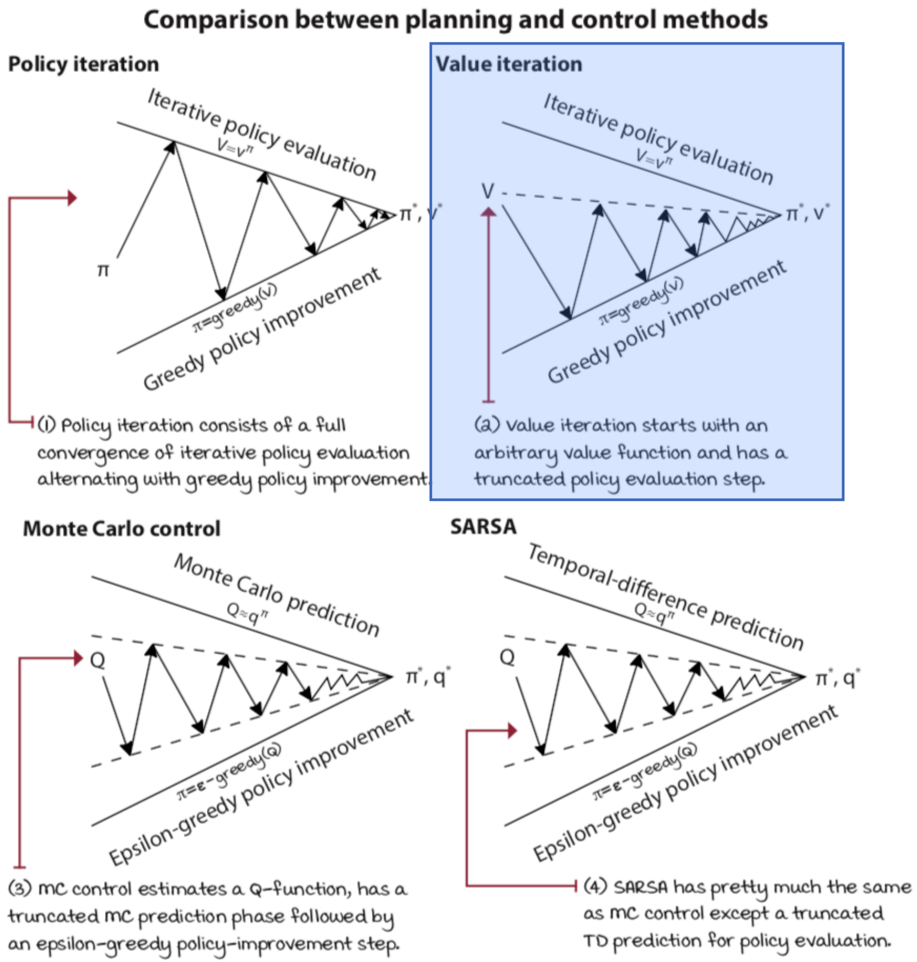
(f) **Wu**

Wu

Outline

1. Dynamic programming iteration for infinite horizon problems
2. Value iteration
3. **Policy iteration**

Numerous variations



Adapted from Morales, Grokking Deep Reinforcement Learning, 2020.

More generally…

Value iteration:

1. $V_{i+1}(s) = \max_{a \in A} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[V_i(s')]$ for all s
2. $\pi_K(s) = \arg\max_{a \in A} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} V_K(s')$

Related Operations:

- Policy evaluation: $V_{i+1}(s) = r(s, \pi_i(s)) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,\pi_i(s))}[V_i(s')]$ for all s
- Policy improvement: $\pi_i(s) = \arg\max_{a \in A} r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} V_i(s')$

☞ **Generalized Policy Iteration:**

- Repeat:
  1. Policy evaluation for N steps
  2. Policy improvement

- Value iteration: $N = 1$; Policy iteration: $N = \infty$

Policy Iteration: the Idea

1. Let π_0 be any stationary policy

2. At each iteration $k = 1, 2, \ldots., K$
   - Policy evaluation: given π_k, compute V^{π_k}
   - Policy improvement: compute the greedy policy

$$\pi_{k+1}(s) \in \arg\max_{a \in A} \left[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V^{\pi_k}(s') \right]$$

3. Stop if $V^{\pi_k} = V^{\pi_{k-1}}$

4. Return the last policy π_K

Policy Iteration: the Guarantees

Proposition

The policy iteration algorithm generates a sequence of policies with non-decreasing performance

$$V^{\pi_{k+1}} \geq V^{\pi_k}$$

and it converges to π^* in a finite number of iterations.

The Bellman Equation

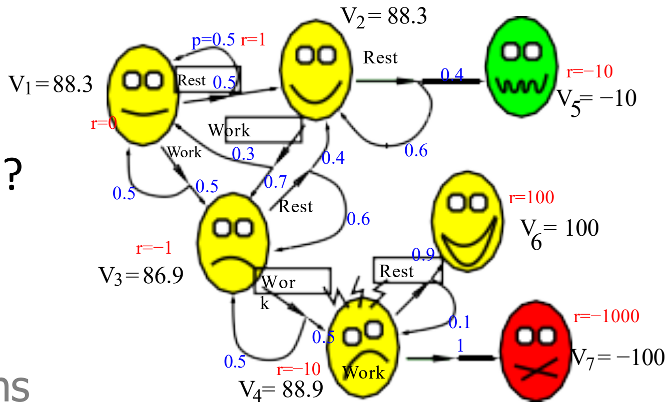
Theorem (Bellman equation)

For any stationary policy $\pi = (\pi, \pi, \ldots)$, at any state $s \in S$, the state value function satisfies the Bellman equation:

$$V^{\pi}(s) = r\big(s, \pi(s)\big) + \gamma \sum_{s' \in S} p(s' | s, \pi(s)) V^{\pi}(s')$$

The student dilemma

- **Discuss**: How to solve this system of equations?



$$V^{\pi}(x) = r\big(x, \pi(x)\big) + \gamma \sum_{y} p(y|x, \pi(x)) V^{\pi}(y)$$

System of equations

$$\begin{cases} V_1 = & 0 + 0.5\, V_1 + 0.5\, V_2 \\ V_2 = & 1 + 0.3\, V_1 + 0.7\, V_3 \\ V_3 = & -1 + 0.5\, V_4 + 0.5\, V_3 \\ V_4 = & -10 + 0.9 V_6 + 0.1\, V_4 \\ V_5 = & -10 \\ V_6 = & 100 \\ V_7 = & -1000 \end{cases} \Longrightarrow$$

$$V, R \in \mathbb{R}^7, P^{\pi} \in \mathbb{R}^{7\times 7}$$

$$V = R + PV$$

$$\Downarrow$$

$$V = (I - P)^{-1} R$$

Recap: The Bellman Operators

Notation. w.l.o.g. a discrete state space $|S| = N$ and $V^{\pi} \in \mathbb{R}^N$
(analysis extends to include $N \to \infty$)

Definition

For any $W \in \mathbb{R}^N$, the Bellman operator $\mathrm{T}^{\pi}: \mathbb{R}^N \to \mathbb{R}^N$ is

$$\mathrm{T}^{\pi} W(s) = r\big(s, \pi(s)\big) + \gamma \sum_{s'} p(s'|s, \pi(s)) W(s')$$

And the optimal Bellman operator (or dynamic programming operator) is

$$\mathrm{T} W(s) = \max_{a \in A} \left[r(s,a) + \gamma \sum_{s'} p(s'|s,a) W(s) \right]$$

The Bellman Operators

Proposition

Properties of the Bellman operators

1. Monotonicity: For any $W_1, W_2 \in \mathbb{R}^N$, if $W_1 \leq$ \W$_2$ component-wise, then

$$\mathrm{T}^{\pi} W_1 \leq \mathrm{T}^{\pi} W_2$$

$$\mathrm{T} W_1 \leq \mathrm{T} W_2$$

2. Offset: For any scalar $c \in \mathbb{R}$,

$$\mathrm{T}^{\pi}(W - cI_N) = \mathrm{T}^{\pi} W + \gamma c I_N$$

$$\mathrm{T}(W - cI_N) = \mathrm{T} W + \gamma c I_N$$

The Bellman Operators

Proposition

3. Contraction in L_∞-norm: For any $W_1, W_2 \in \mathbb{R}^N$

$$\|\mathrm{T}^\pi W_1 - \mathrm{T}^\pi W_2\|_\infty \le \gamma \|W_1 - W_2\|_\infty$$

$$\|\mathrm{T} W_1 - \mathrm{T} W_2\|_\infty \le \gamma \|W_1 - W_2\|_\infty$$

4. Fixed point: For any policy π,

V^π is the unique fixed point of T^π

V^* is the unique fixed point of T

- For any $W \in \mathbb{R}^N$ and any stationary policy π

$$\lim_{k\to\infty} (\mathrm{T}^\pi)^k W = V^\pi$$

$$\lim_{k\to\infty} (\mathrm{T})^k W = V^*$$

Policy Iteration: the Idea

1. Let π_0 be any stationary policy

2. At each iteration $k = 1, 2, \dots, K$
   - Policy evaluation: given π_k, compute V^{π_k}
   - Policy improvement: compute the greedy policy

$$\pi_{k+1}(s) \in \arg\max_{a \in A} \left[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V^{\pi_k}(s') \right]$$

3. Stop if $V^{\pi_k} = V^{\pi_{k-1}}$

4. Return the last policy π_K

Policy Iteration: the Guarantees

Proposition

The policy iteration algorithm generates a sequence of policies with non-decreasing performance

$$V^{\pi_{k+1}} \geq V^{\pi_k}$$

and it converges to π^* in a finite number of iterations.

Proof: Policy Iteration

From the definition of the Bellman operators and the greedy policy π_{k+1}

$$V^{\pi_k} = \mathcal{T}^{\pi_k} V^{\pi_k} \le \mathcal{T} V^{\pi_k} = \mathcal{T}^{\pi_{k+1}} V^{\pi_k} \tag{1}$$

and from the monotonicity property of $\mathcal{T}^{\pi_{k+1}}$, it follows that

$$V^{\pi_k} \le \mathcal{T}^{\pi_{k+1}} V^{\pi_k}$$

$$\mathcal{T}^{\pi_{k+1}} V^{\pi_k} \le (\mathcal{T}^{\pi_{k+1}})^2 V^{\pi_k}$$

$$\ldots$$

$$(\mathcal{T}^{\pi_{k+1}})^{n-1} V^{\pi_k} \le (\mathcal{T}^{\pi_{k+1}})^n V^{\pi_k}$$

$$\ldots$$

Joining all inequalities in the chain, we obtain

$$V^{\pi_k} \le \lim_{n\to\infty} (\mathcal{T}^{\pi_{k+1}})^n V^{\pi_k} = V^{\pi_{k+1}}$$

Then $(V^{\pi_k})_k$ is a non-decreasing sequence.

Policy Iteration: the Guarantees

Since a finite MDP admits a finite number of policies, then the termination condition is eventually met for a specific k.

Thus eq. 1 holds with an equality and we obtain

$$V^{\pi_k} = \mathcal{T}V^{\pi_k}$$

and $V^{\pi_k} = V^*$ which implies that π_k is an optimal policy.

∎

Policy Iteration: Complexity

- Policy Improvement Step
  - Complexity $O(S^2A)$
- Number of Iterations
  - At most $O\left(\frac{SA}{1-\gamma}\log\left(\frac{1}{1-\gamma}\right)\right)$
  - Other results exist that do not depend on γ

Wu

Comparison between Value and Policy Iteration

- Value Iteration
  - Pros: each iteration is very computationally efficient.
  - Cons: convergence is only asymptotic.

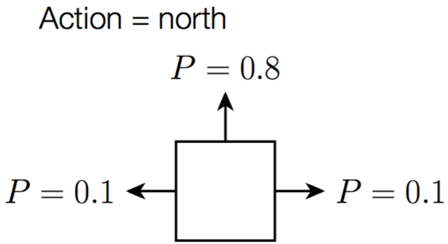
- Policy Iteration
  - Pros: converge in a finite number of iterations (often small in practice).
  - Cons: each iteration requires a full policy evaluation and it might be expensive.

Wu

Example: Winter parking (with ice and potholes)

- Simple grid world with a *goal state* (green, desired parking spot) with reward (+1), a *"bad state"* (red, pothole) with reward (-100), and all other states neural (+0).
- *Omnidirectional vehicle (agent)* can head in any direction. Actions move in the desired direction with probably 0.8, in one of the perpendicular directions with.
- Taking an action that would bump into a wall leaves agent where it is.

0	0	0	1
0		0	-100
0	0	0	0



[Source: adapted from Kolter, 2016]

Example: value iteration

Running value iteration with $\gamma = 0.9$

0	0	0	1
0		0	-100
0	0	0	0

Original reward function

(a)

Running value iteration with $\gamma = 0.9$

0	0	0.72	1.81
0		0	-99.91
0	0	0	0

$\hat{V}$ at one iteration

(b)

Running value iteration with $\gamma = 0.9$

0.809	1.598	2.475	3.745
0.268		0.302	-99.59
0	0.034	0.122	0.004

$\hat{V}$ at five iterations

(c)

Running value iteration with $\gamma = 0.9$

2.686	3.527	4.402	5.812
2.021		1.095	-98.82
1.390	0.903	0.738	0.123

$\hat{V}$ at 10 iterations

(d)

Running value iteration with $\gamma = 0.9$

5.470	6.313	7.190	8.669
4.802		3.347	-96.67
4.161	3.654	3.222	1.526

$\hat{V}$ at 1000 iterations

(e)

Running value iteration with $\gamma = 0.9$

→	→	→	↑
↑		←	←
↑	←	←	↓

Resulting policy after 1000 iterations

(f) Wu

Wu

Example: policy iteration

Running policy iteration with $\gamma = 0.9$, initialized with policy $\pi(s) =$ North

0	0	0	1
0		0	-100
0	0	0	0

Original reward function

(a)

Running policy iteration with $\gamma = 0.9$, initialized with policy $\pi(s) =$ North

0.418	0.884	2.331	6.367
0.367		-8.610	-105.7
-0.168	-4.641	-14.27	-85.05

V^π at one iteration

(b)

Running policy iteration with $\gamma = 0.9$, initialized with policy $\pi(s) =$ North

5.414	6.248	7.116	8.634
4.753		2.881	-102.7
2.251	1.977	1.849	-8.701

V^π at two iterations

(c)

Running policy iteration with $\gamma = 0.9$, initialized with policy $\pi(s) =$ North

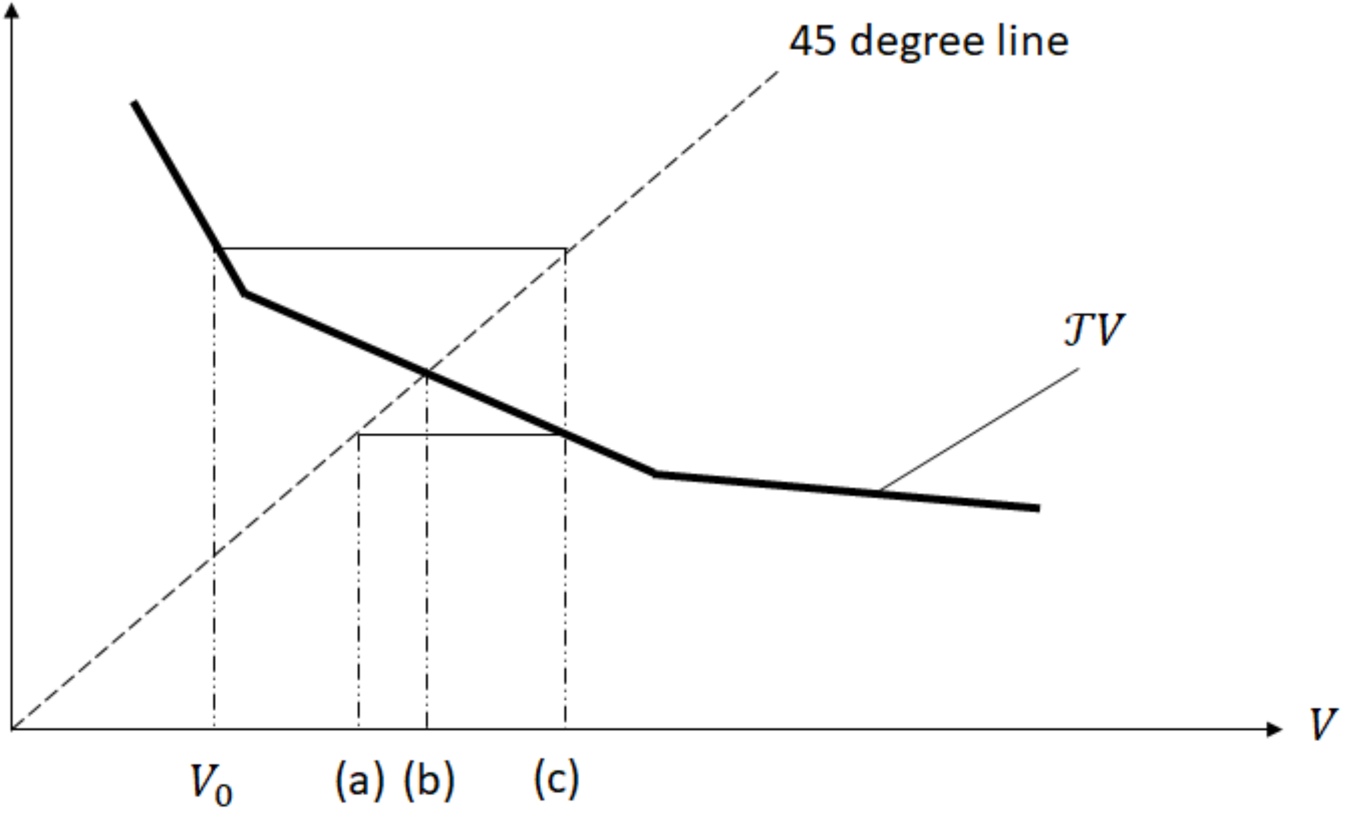
5.470	6.313	7.190	8.669
4.803		3.347	-96.67
4.161	3.654	3.222	1.526

V^π at three iterations (converged)

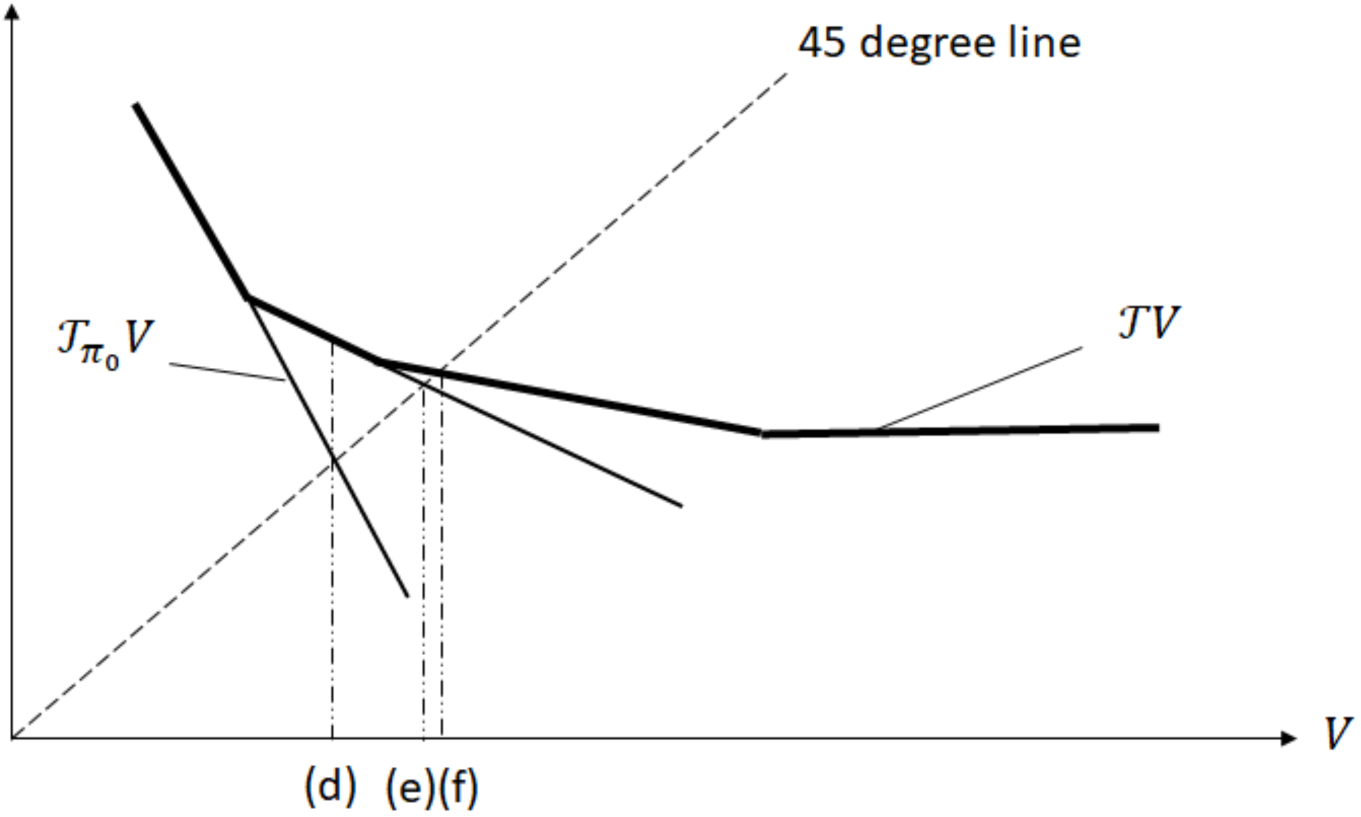
(d)

Wu

Value iteration: geometric Interpretation



Policy iteration: geometric Interpretation



Summary & Takeaways

- The ideas from **dynamic programming**, namely the **principle of optimality**, carry over to infinite horizon problems.
- The **value iteration** algorithm solves discounted infinite horizon MDP problems by leveraging results of **Bellman operators**, namely the **optimal Bellman equation**, **contractions**, and **fixed points**.
- **Generalized policy iteration** methods include policy iteration and value iteration.
- **Policy iteration** algorithm additionally leverages **monotonicity** and **Bellman equation**.
- The update mechanism for VI and PI differ and thus their convergence in practice depends on the **geometric structure** of the optimal value function.