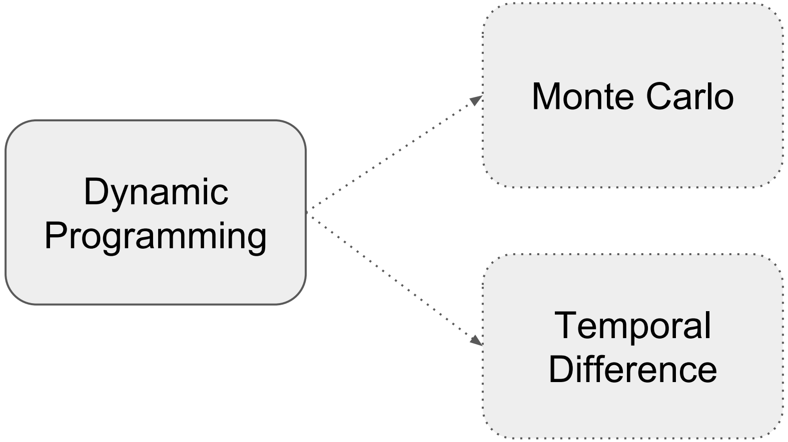
Chapter 4: Dynamic Programming

Seungjae Ryan Lee

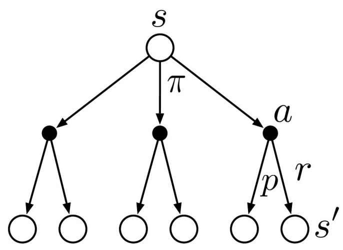
endtoend.ai

Dynamic Programming

- Algorithms to compute optimal policies with a **perfect model of environment**
- Use value functions to structure searching for good policies
- Foundation of all methods hereafter



Policy Evaluation (Prediction)



- Compute state-value $v_\pi(s)$ for some policy π
- Use the Bellman Equation:

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \mid s, a) \Big[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s'] \Big] \\
&= \sum_a \pi(a|s) \sum_{s', r} p(s', r \mid s, a) \Big[r + \gamma v_\pi(s') \Big], \quad \text{for all } s \in \mathcal{S},
\end{aligned}
$$

endtoend.ai

Iterative Policy Evaluation

- Solving linear systems is tedious → Use iterative methods
- Define sequence of approximate value functions $v_0, v_1, v_2 \ldots$
- *Expected update* using the Bellman equation:
  - Update based on *expectation of all possible next states*

$$
\begin{aligned}
v_{k+1}(s) &\doteq \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v_k(s')\Big],
\end{aligned}
$$

endtoend.ai

Iterative Policy Evaluation in Practice

- In-place methods usually converge faster than keeping two arrays
- Terminate policy evaluation when $\max_s |v_{k+1}(s) - v_k(s)|$ is sufficiently small

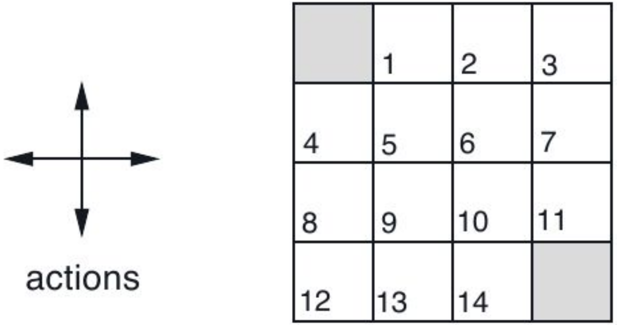
Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
    $\Delta \leftarrow 0$
    Loop for each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Gridworld Example

- Deterministic state transition
- Off-the-grid actions leave the state unchanged
- Undiscounted, episodic task



$R_t = -1$
on all transitions

endtoend.ai

Policy Evaluation in Gridworld

- Random policy π

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$k = 0$

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

$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

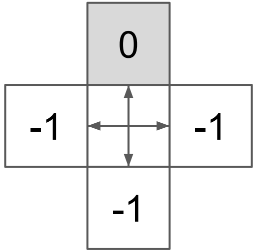
$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

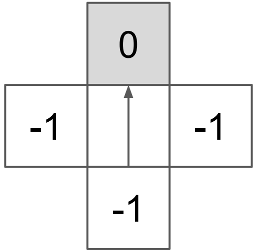
endtoend.ai

Policy Improvement - One state

- Suppose we know v_π for some policy π
- For a state s, see if there is a better action $a \neq \pi(s)$
- Check if $q_\pi(s, a) \geq v_\pi(s)$
  - If true, greedily selecting a is better than $\pi(s)$
  - Special case of *Policy Improvement Theorem*



$\pi(s)$



a

endtoend.ai

Policy Improvement Theorem

For policies π, π' , if for all state $s \in \mathcal{S}$,

$$q_\pi(s, \pi'(s)) \geq v_\pi(s)$$

Then, π' is at least as good a policy as π.

$$v_{\pi'}(s) \geq v_\pi(s)$$

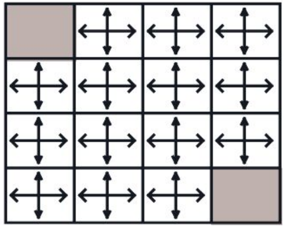
(Strict inequality if $q_\pi(s, \pi(s')) > v_\pi(s)$)

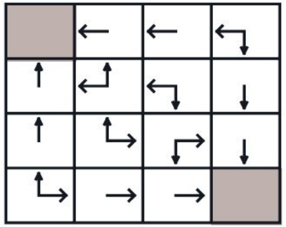
endtoend.ai

Policy Improvement

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

- Find better policies with the computed value function
- Use a new *greedy* policy π'
- Satisfies the conditions of Policy Improvement Theorem





$$
\begin{aligned}
\pi'(s) &\doteq \arg\max_a q_\pi(s,a) \\
&= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \arg\max_a \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v_\pi(s')\Big],
\end{aligned}
$$

Guarantees of Policy Improvement

- If $v_\pi = v_{\pi'}$, then the Bellman Optimality Equation holds.

$$\begin{aligned} v_{\pi'}(s) &= \max_a \mathbb{E}[R_{t+1} + \gamma v_{\pi'}(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v_{\pi'}(s')\Big]. \end{aligned}$$

→ Policy Improvement always returns a better policy unless already optimal

Policy Iteration

- Repeat *Policy Evaluation* and *Policy Improvement*
- Guaranteed improvement for each policy
- Guaranteed convergence in finite number of steps for finite MDPs

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*,$$

Policy Iteration in Practice

- Initialize $v_{\pi_{t+1}}$ with v_{π_t} for quicker policy evaluation
- Often converges in surprisingly few iterations

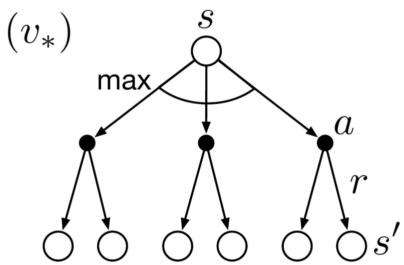
Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
      $\Delta \leftarrow 0$
      Loop for each $s \in \mathcal{S}$:
         $v \leftarrow V(s)$
         $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \big[r + \gamma V(s')\big]$
         $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
      old-action $\leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a) \big[r + \gamma V(s')\big]$
      If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Value Iteration



- "Truncate" policy evaluation
  - Don't wait until $\max_s |v_{k+1}(s) - v_k(s)|$ is sufficiently small
  - Update state values **once** for each state
- *Evaluation* and *improvement* can be simplified to one update operation
  - Bellman optimality equation turned into an update rule

$$\begin{aligned} v_{k+1}(s) &= \sum_a \pi_k(a, s) \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_k(s') \right] \\ &= \max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_k(s') \right] \end{aligned}$$

endtoend.ai

Value Iteration in Practice

- Terminate when $\max_s |v_{k+1}(s) - v_k(s)|$ is sufficiently small

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$

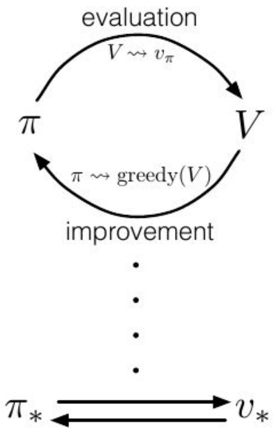
endtoend.ai

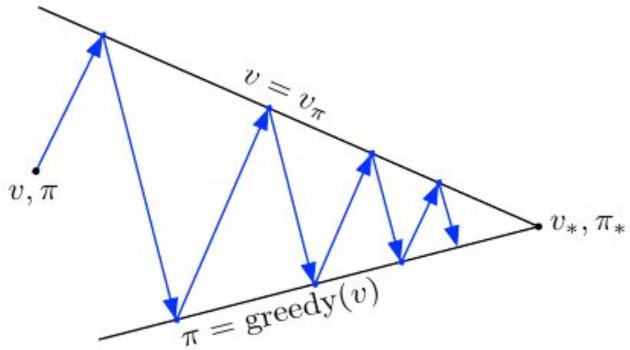
Asynchronous Dynamic Programming

- Don’t sweep over the entire state set systematically
  - Some states are updated multiple times before other state is updated once
  - Order/skip states to propagate information efficiently
- Can intermix with real-time interaction
  - Update states according to the agent’s experience
  - Allow focusing updates to relevant states
- To converge, all states must be continuously updated

Generalized Policy Iteration

- Idea of interaction between policy evaluation and policy improvement
  - Policy improved w.r.t. value function
  - Value function updated for new policy
- Describes most RL methods
- Stabilized process guarantees optimal policy





endtoend.ai

Efficiency of Dynamic Programming

- Polynomial in $|\mathcal{S}|$ and $|\mathcal{A}|$
  - Exponentially faster than direct search in policy space $|\mathcal{A}|^{|\mathcal{S}|}$
- More practical than linear programming methods in larger problems
  - Asynchronous DP preferred for large state spaces
- Typically converge faster than their worst-case guarantee
  - Initial values can help faster convergence

Thank you!

Original content from

- Reinforcement Learning: An Introduction by Sutton and Barto

You can find more content in

- github.com/seungjaeryanlee
- www.endtoend.ai