Lecture 3: Dynamic Programming

Friday, November 12, 2021

Reinforcement Learning, Winter Term 2021/22

Joschka Boedecker, Gabriel Kalweit, Jasper Hoffmann

Neurorobotics Lab
University of Freiburg

Lecture Overview

Lecture Overview

Recap: Bellman Optimality Equations

Bellman Optimality Equation for v_*

The Bellman Equation for the optimal value function v_* is defined as:

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')].$$

Bellman Optimality Equation for q_*

The Bellman Equation for the optimal action-value function q_* is:

$$q_*(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} q_*(s', a')].$$

How can we turn these equations into practical algorithms to find optimal policies π_*?

Policy Iteration: Overview

Idea: Alternate **evaluating** the value function v_π and **improving** the policy π to convergence.

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

Policy Evaluation

Compute the state-value function v_π for an arbitrary policy π.
$\forall s \in S$:

$$v_\pi(s) \doteq \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) \left[r + \gamma v_\pi(s')\right]$$

If the environments dynamics are completely known, this is a system of $|\mathcal{S}|$ simultaneous linear equations in $|\mathcal{S}|$ unknowns. With the Bellman equation, we can iteratively update an initial approximation v_0:

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

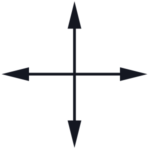
Policy Evaluation

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

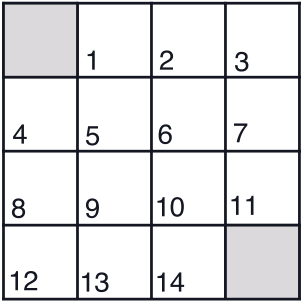
Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\mathit{terminal}) = 0$

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

Example: Gridworld



actions



$$R_t = -1$$

on all transitions

Policy Improvement

Once we have the value function for a policy, we consider which action a to select in a state s when we follow our old policy π afterwards. To decide this, we look at the Bellman equation of the state-action value function:

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

Policy improvement theorem

Let π and π' be any pair of deterministic policies. If, $\forall s \in S$,

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

then the policy π' must be as good as, or better than, π. It follows that, $\forall s \in S$:

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

Policy Improvement

To implement this, we compute $q_\pi(s,a)$ for *all* states and *all* actions, and consider the greedy policy:

$$\begin{aligned}
\pi'(s) &\doteq \arg\max_a q_\pi(s,a) \\
&= \arg\max_a \mathbb{E}\left[R_{t+1} + \gamma v_\pi(S_{t_1}) | S_t = s, A_t = a\right] \\
&= \arg\max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right]
\end{aligned}$$

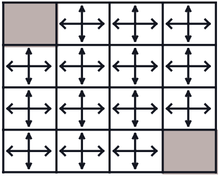
Example: Gridworld

v_k for the random policy

greedy policy w.r.t. v_k

$k = 0$

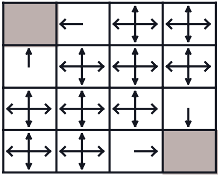
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0



random policy

$k = 1$

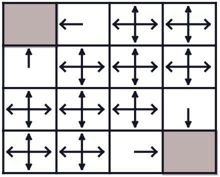
0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0



Example: Gridworld

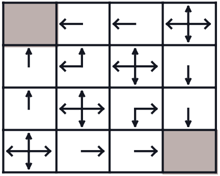
$k = 1$

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



$k = 2$

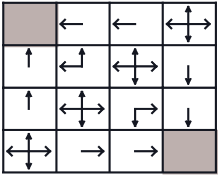
0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0



Example: Gridworld

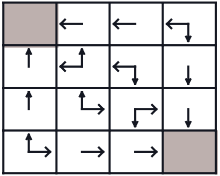
$k = 2$

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



Example: Gridworld

$k = 3$

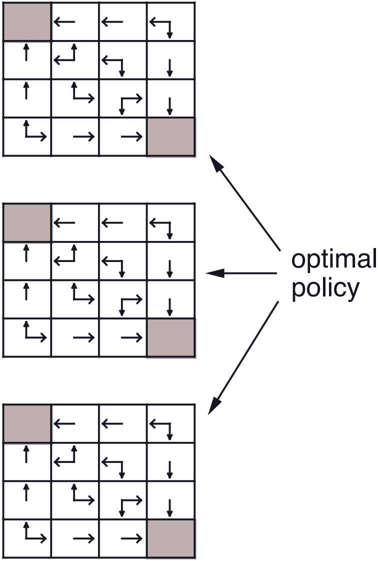
0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

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



Policy Iteration

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 $s \in \mathcal{S}$

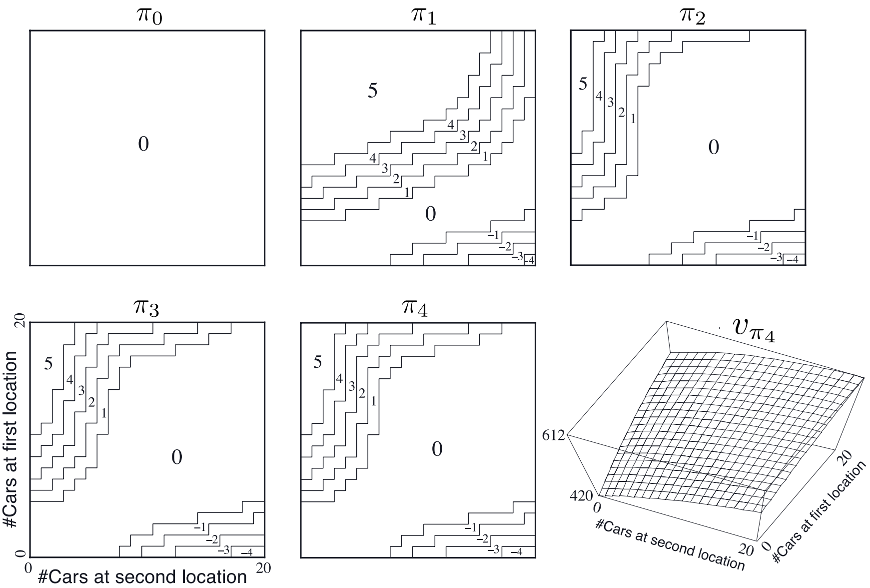
2. Policy Evaluation
   Loop:
   - $\Delta \leftarrow 0$
   - Loop for each $s \in \mathcal{S}$:
     - $v \leftarrow V(s)$
     - $V(s) \leftarrow \sum_{s',r} p(s', r \,|\, s, \pi(s)) \big[r + \gamma V(s') \big]$
     - $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

   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* $\leftarrow \pi(s)$
   - $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \,|\, s, a) \big[r + \gamma V(s') \big]$
   - 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

Example: Car rental



Lecture Overview

Value Iteration

Performing policy evaluation to convergence *in every iteration* is costly and often not necessary. A special case is to evaluate just once and combine it with the policy improvement step:

$$
\begin{aligned}
v_{k+1}(s) &\doteq \max_a \mathbb{E}\left[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s, A_t = a\right] \\
&= \max_a \sum_{s',r} p(s', r | s, a) \left[r + \gamma v_k(s')\right]
\end{aligned}
$$

Value Iteration

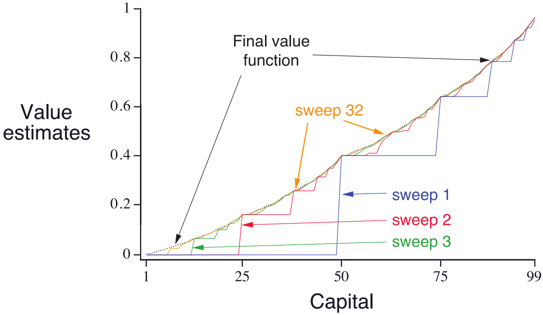
Value Iteration, for estimating $\pi \approx \pi_*$

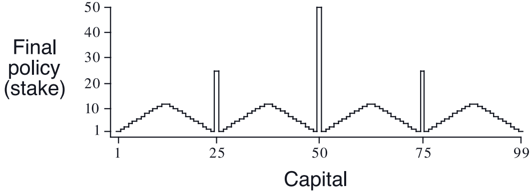
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

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

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

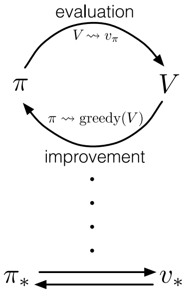
Gambler's Problem

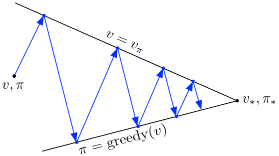




Additional topics

- Asynchronous Dynamic Programming
- Generalized Policy Iteration
- Efficiency of Dynamic Programming





Lecture Overview

Summary by Learning Goals

Having heard this lecture, you can now...

- formulate and apply Policy Iteration
- formulate and apply Value Iteration