4 Dynamic Programming for RL

Melih Kandemir

Özyeğin University
Computer Science Department
melih.kandemir@ozyegin.edu.tr

17 Oct 2017

Dynamic programming (DP)

- **Dynamic:** sequential (temporal)
- **Programming** optimizing a program
  (a sequence of operation steps)

DP is applicable to problems that consist of

- optimal substructure
  - there exists a notion of optimality that can be proven
  - optimal solution can be decomposed into subproblems
- overlapping subproblems
  - subproblems recur many times
  - solutions can be cached and reused

DP for MDPs

DP suits perfectly for solving MDPs

- **optimal substructure:** Bellman equation decomposes recursively (optimality principle yet to come!)
- **overlapping subproblems:** Value function stores and reuses solutions

Dynamic programming in RL

- Assumes full knowledge of the MDP, which is rarely feasible, even more rarely tractable in real applications.
- Still worthwhile stuyding to understand finer RL algorithms.
- The main point in DP is to exploit the value function concept to learn good policies.

Bellman optimality equation recap

Given v_* and q_*, finding π_* is trivial

$$v_*(s) = \underset{a}{\operatorname{argmax}}\, \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a],$$
$$= \underset{a}{\operatorname{argmax}} \sum_{s',r'} p(s', r | s, a) \Big[r + \gamma v_*(s') \Big],$$

or

$$q_*(s, a) = \mathbb{E}\Big[R_{t+1} + \gamma\, \underset{a'}{\operatorname{argmax}}\, q_*(S_{t+1}, a') \Big| S_t = s, A_t = a \Big],$$
$$= \sum_{s',r'} p(s', r | s, a) \Big[r + \gamma\, \underset{a'}{\operatorname{argmax}}\, q_*(s', a') \Big].$$

Policy evaluation

Given π, compute the related state-value function (a.k.a. *the prediction problem*).

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

Existence and uniqueness of v_π is guaranteed if either $\gamma < 1$ or the task is episodic.

Iterative policy evaluation

Use Bellman equation as an **update rule**

Input: π
$V(s) \leftarrow 0, \ \forall s \in \mathcal{S}$
repeat
    $\Delta \leftarrow 0$
    for all $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\Big[r + \gamma V(s')\Big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \epsilon$
Output: $V \approx v_\pi$

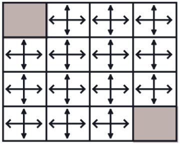
Example: GridWorld

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



$k = 1$

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

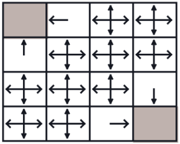


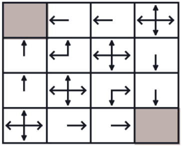
Figure. Sutton and Barto, MIT Press, 2017

Example: GridWorld

All policies are optimal from $K = 3$ on.

$k = 2$

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

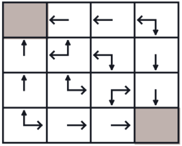


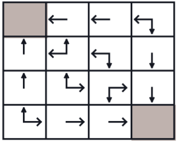
Figure. Sutton and Barto, MIT Press, 2017

Example: GridWorld

All policies are optimal from $K = 3$ on.

$k = 10$

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

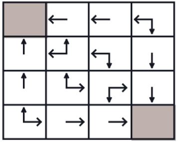


Figure. Sutton and Barto, MIT Press, 2017

Policy improvement theorem

Given a pair of *deterministic* policies π and π', such that $\forall s \in \mathcal{S}$

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

then $\pi' \geq \pi$.

Proof

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{T+1}) | S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(S_{T+1}, \pi'(S_{t+1})) | S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}_{\pi'}[R_{t+2} + \gamma v_\pi(S_{t+2})] | S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) | S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_\pi(S_{t+3}) | S_t = s] \\
&\vdots \\
&\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots | S_t = s] \\
&= v_{\pi'}(s) \qquad \blacksquare
\end{aligned}
$$

Then improve the policy by updating as

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

(the updated π' is called the *greedy* policy)

What if $\pi' = \pi$?

Then $v_\pi = v_{\pi'}$, leading to

$$v_{\pi'}(s) = \underset{a}{\operatorname{argmax}} \sum_{s',r} p(s', r|s, a)\Big[r + \gamma v_{\pi'}(s')\Big],$$

which is the very Bellman optimality equation. Hence, $v_{\pi'} = v_*$ should hold, meaning both π and π' are optimal policies.

Policy improvement update should bring a strictly better policy, otherwise the policy is already optimal!

If multiple actions maximize the state-value function at an update, assign a non-zero probability to each and zero to all other actions. The resultant stochastic policy is also optimal.

Policy iteration

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

$\xrightarrow{E}$: Policy evaluation $\qquad$ $\xrightarrow{I}$: Policy improvement

The policy iteration algorithm

1. Initialize $V(s) \in \mathrm{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$.
2. **repeat** (Policy evaluation)

   $\Delta \leftarrow 0$

   for all $s \in \mathcal{S}$:

   $v \leftarrow V(s)$

   $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\Big[r + \gamma V(s')\Big]$

   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

   until $\Delta < \epsilon$
3. $policyStable \leftarrow true$ (Policy improvement)

   for all $s \in \mathcal{S}$:

   $oldAction \leftarrow \pi(s)$

   $\pi(s) \leftarrow \underset{a}{\operatorname{argmax}} \sum_{s',r} p(s', r|s, a)\Big[r + \gamma V(s')\Big]$

   if $oldAction \neq \pi(s)$, **then** $policyStable \leftarrow false$

   if $policyStable$, **then** stop and return $V \approx v_*$ and $\pi \approx \pi_*$,

   else go to 2.

Value iteration

- Policy iteration requires policy evaluation in each iteration, which itself takes multiple iterations to converge.
- Instead, *truncate* policy evaluation after one sweep.
- Combine policy improvement and truncated policy steps

$$v_{k+1}(s) \triangleq \operatorname*{argmax}_a \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_k(s')\Big].$$

The value iteration algorithm

Initialize $V(s) \in \mathrm{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$.

repeat

$\Delta \leftarrow 0$

for all $s \in \mathcal{S}$:

$v \leftarrow V(s)$

$V(s) \leftarrow \underset{a}{\operatorname{argmax}} \sum_{s',r} p(s', r|s, a)\Big[r + \gamma V(s')\Big]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \epsilon$

Output: A deterministic policy $\pi \approx \pi_*$, such that

$$\pi(s) = \underset{a}{\operatorname{argmax}} \sum_{s',r} p(s', r|s, a)\Big[r + \gamma V(s')\Big]$$

Synchronous DP

$v_{new} \leftarrow v_{old}$
for $s \in \mathcal{S}$:
    $v_{new}(s) \leftarrow Bellman(v_{old})$
$v_{old} \leftarrow v_{new}$

- Two copies of the values are stored
- All states are updated synchronously

Asynchronous DP

- It is often the case that even a single sweep is prohibitive.
- **Asynch DP:** Update the states *in-place* in arbitrary order.
- Convergence to v_* still guaranteed if $0 \leq \gamma < 1$ and all states occur infinitely many times asymptotically.
- We can even arrange the back-up order to boost up convergence. Back up tricky states more often than easy ones, skip irrelevant ones entirely.
- Back up states as the agent *visits* them (i.e. as the case occurs).
- Interleave evaluation and improvement steps on single state basis.

Asynchronous DP

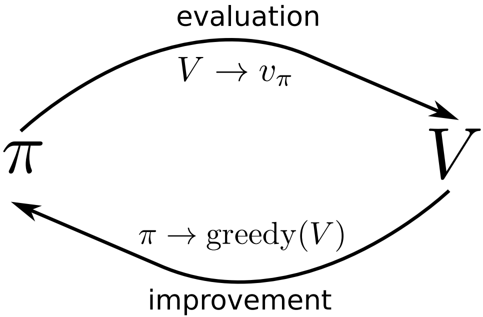
- *In-place* DP
  for $s \in \mathcal{S}$:
  $v(s) \leftarrow Bellman(v)$
- Prioritized sweeping
  Sort states by the *Bellman error*

  $$\left|\operatorname*{argmax}_a \sum_a \pi(a|s)[r + \gamma \sum_{s',r} p(s',r|s,a)v(s')] - v(s)\right|$$

  Update only the highest-priority states
- Real-time DP
  Update only the state being *experienced*

Generalized Policy Iteration (GPI)

- Let the policy evaluation and policy improvement goals *interact*, no matter in what granularity.
- Value function stabilizes when it is consistent with the current policy.
- Policy stabilizes when it is greedy wrt the current value function.



Generalized Policy Iteration (GPI)

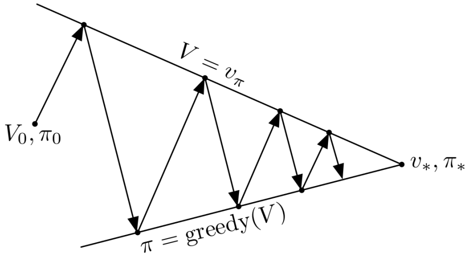


Figure. Sutton and Barto, MIT Press, 2017

Policy evaluation and improvement goals

- both compete (zig-zag)
- and co-operate (converge to the same point).

Scalability of DP

- Uses *full-width* backups
  - Considers every successor state
- Scales up to medium-sized problems ($\propto 1M$ states)
- Suffers from *Bellman's curse of dimensionality*
  - Number of states grows exponentially with the number of state variables