Univ Gustave Eiffel - Cosys / Grettia

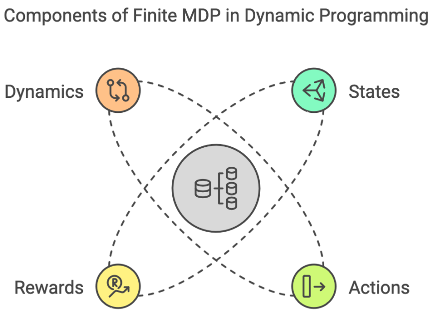
Reinforcement Learning and Optimal Control

Dynamic Programming

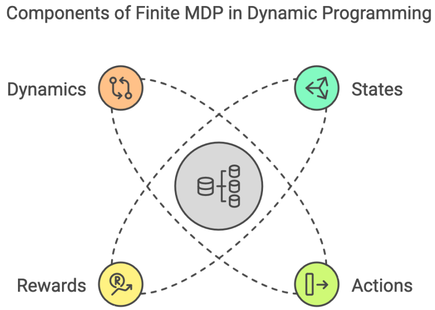
Nadir Farhi
chargé de recherche, UGE - Cosys/Grettia
nadir.farhi@univ-eiffel.fr

Uni Eiffel - 30 september 2024

MDP & Dynamic programming



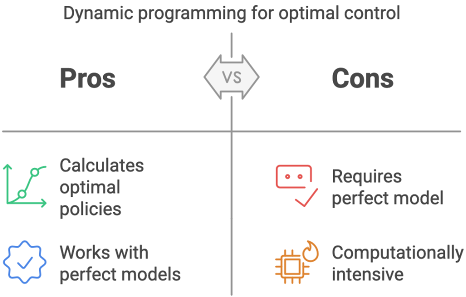
MDP & Dynamic programming



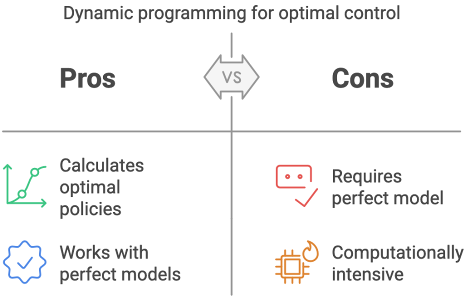
Dynamic programming solves optimal control problems :

- calculates optimal policies,
- given a perfect model of the environment (such as a MDP).

Dynamic programming (DP) vs Reinforcement Learning (RL)



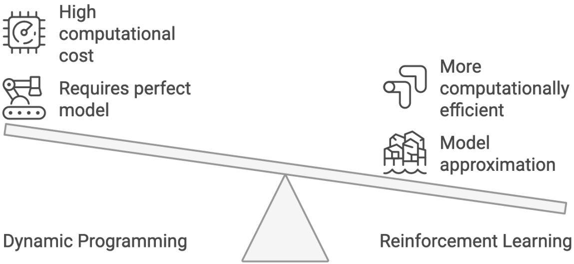
Dynamic programming (DP) vs Reinforcement Learning (RL)



Drawbacks of DP w.r.t. RL :

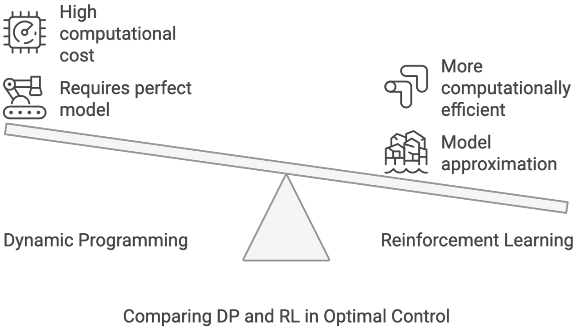
- Assumption of a perfect model.
- Great computational expenses (computation time & space memory).

RL approximates DP



Comparing DP and RL in Optimal Control

RL approximates DP



RL can be seen as approximation of Dynamic programming.

Dynamic programming

- Environment for DP : finite MDP :
  - finite sets $\mathcal{S}, \mathcal{A}, \mathcal{R}$ for states, actions and rewards.
  - the dynamics are given by $p(s', r \mid s, a)$.

Dynamic programming

- Environment for DP : finite MDP :
  - finite sets $\mathcal{S}, \mathcal{A}, \mathcal{R}$ for states, actions and rewards.
  - the dynamics are given by $p(s', r \mid s, a)$.
- Main idea : Use the value function :

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ for all } s \in \mathcal{S},$$

Dynamic programming

- Environment for DP : finite MDP :
  - finite sets $\mathcal{S}, \mathcal{A}, \mathcal{R}$ for states, actions and rewards.
  - the dynamics are given by $p(s', r \mid s, a)$.
- Main idea : Use the value function :

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ for all } s \in \mathcal{S},$$

or the action-value function :

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right].$$

Dynamic programming

- Bellman equation (state value function) :

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

Dynamic programming

- Bellman equation (state value function) :

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

- Bellman equation (state-action value function) :

$$
\begin{aligned}
q_*(s, a) &= \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \Bigm| S_t = s, A_t = a \Big] \\
&= \sum_{s',r} p(s', r \mid s, a) \Big[r + \gamma \max_{a'} q_*(s', a') \Big],
\end{aligned}
$$

Dynamic programming

Four algorithms

1. Policy evaluation (Prediction)
2. Policy improvement
3. Policy iteration & GPI
4. Value iteration

1 - Policy Evaluation (Prediction)

- Recall the Bellman equation :

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

1 - Policy Evaluation (Prediction)

- Recall the Bellman equation :

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

- Existence and uniqueness of v_π :
  - Either $\gamma < 1$,
  - Or termination is guaranteed from all states under the policy v_π.

1 - Policy Evaluation (Prediction)

- Recall the Bellman equation :

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

- Existence and uniqueness of v_π :
  - Either $\gamma < 1$,
  - Or termination is guaranteed from all states under the policy v_π.
- Known environment (dynamics) $\Rightarrow$ the Bellman equation is a linear system ($|\mathcal{S}|$ equations and $|\mathcal{S}|$ variables $v_\pi(s), s \in \mathcal{S}$)

1 - Policy Evaluation (Prediction)

- Iterative policy evaluation :

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

1 - Policy Evaluation (Prediction)

- Iterative policy evaluation :

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

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 $V(s)$ arbitrarily, for $s \in \mathcal{S}$, and $V(terminal)$ to 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 $\Delta < \theta$

2 - Policy Improvement

- For a given π, suppose we have v_π

2 - Policy Improvement

- For a given π, suppose we have v_π
- For a given s, what if we select a in s, and thereafter following policy π ?

2 - Policy Improvement

- For a given π, suppose we have v_π
- For a given s, what if we select a in s, and thereafter following policy π ?
- The value of this way of behaving is to compute

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

and compare it to $v_\pi(s)$.

2 - Policy Improvement

- For a given π, suppose we have v_π
- For a given s, what if we select a in s, and thereafter following policy π ?
- The value of this way of behaving is to compute

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

  and compare it to $v_\pi(s)$.
- If $q_\pi(s,a) \geq v_\pi(s)$, then, it will better to select a every time s is encountered.

2 - Policy Improvement

- For a given π, suppose we have v_π
- For a given s, what if we select a in s, and thereafter following policy π ?
- The value of this way of behaving is to compute

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

  and compare it to $v_\pi(s)$.
- If $q_\pi(s,a) \geq v_\pi(s)$, then, it will better to select a every time s is encountered.
- Then, the new policy is better than π.

2 - Policy Improvement

- Let π and π' a pair of deterministic policies, and $s \in \mathcal{S}$, such that :

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

2 - Policy Improvement

- Let π and π' a pair of deterministic policies, and $s \in \mathcal{S}$, such that :

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

- Then π' must be as good as, or better than π :

$$v_{\pi'}(s) \geq v_\pi(s), \forall s \in \mathcal{S}.$$

2 - Policy Improvement

- Let π and π' a pair of deterministic policies, and $s \in \mathcal{S}$, such that :

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

- Then π' must be as good as, or better than π :

$$v_{\pi'}(s) \geq v_\pi(s), \forall s \in \mathcal{S}.$$

- If $q_\pi(s, \pi'(s)) > v_\pi(s)$, then $v_{\pi'}(s) > v_\pi(s), \forall s \in \mathcal{S}$.

2 - Policy Improvement

- Let π and π' a pair of deterministic policies, and $s \in \mathcal{S}$, such that :

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

- Then π' must be as good as, or better than π :

$$v_{\pi'}(s) \geq v_\pi(s), \forall s \in \mathcal{S}.$$

- If $q_\pi(s, \pi'(s)) > v_\pi(s)$, then $v_{\pi'}(s) > v_\pi(s), \forall s \in \mathcal{S}$.
- Therefore π' improves π.

2 - Policy improvement

- we have :

$$q_\pi(s, \pi'(s)) \geq v_\pi(s) \Rightarrow v_{\pi'}(s) \geq v_\pi(s), \forall s \in \mathcal{S}.$$

2 - Policy improvement

- we have :

$$q_\pi(s, \pi'(s)) \geq v_\pi(s) \Rightarrow v_{\pi'}(s) \geq v_\pi(s), \forall s \in \mathcal{S}.$$

- Proof :

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

2 - Policy improvement

- Let us now consider changes at all states, i.e. selecting at each state the action that appears best according to $q_\pi(s, a)$.

2 - Policy improvement

- Let us now consider changes at all states, i.e. selecting at each state the action that appears best according to $q_\pi(s, a)$.
- That is to select the new greedy policy π' satisfying :

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

2 - Policy improvement

- Let us now consider changes at all states, i.e. selecting at each state the action that appears best according to $q_\pi(s, a)$.
- That is to select the new greedy policy π' satisfying :

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

- Then π' is as good as, or better than π.

2 - Policy improvement

- Let us now consider changes at all states, i.e. selecting at each state the action that appears best according to $q_\pi(s, a)$.
- That is to select the new greedy policy π' satisfying :

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

- Then π' is as good as, or better than π.
- Suppose now that π' is as good as, but not better than π, i.e. $v_{\pi'} = v_\pi$.

2 - Policy improvement

- Let us now consider changes at all states, i.e. selecting at each state the action that appears best according to $q_\pi(s, a)$.
- That is to select the new greedy policy π' satisfying :

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

- Then π' is as good as, or better than π.
- Suppose now that π' is as good as, but not better than π, i.e. $v_{\pi'} = v_\pi$.
- we then have :

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

which is the Bellman euation. Therefore π and π' are optimal.

2 - Policy improvement

- Applying the policy improvement to a given policy converges to the optimal policy.

2 - Policy improvement

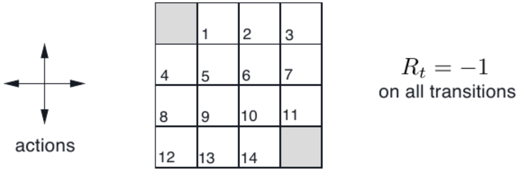
- Applying the policy improvement to a given policy converges to the optimal policy.
- We have also convergence with stochastic policies.

2 - Policy improvement

- Applying the policy improvement to a given policy converges to the optimal policy.
- We have also convergence with stochastic policies.

2 - Policy improvement

Example 4.1 Consider the 4×4 gridworld shown below.



The nonterminal states are $\mathcal{S} = \{1, 2, \ldots, 14\}$. There are four actions possible in each state, $\mathcal{A} = \{\texttt{up}, \texttt{down}, \texttt{right}, \texttt{left}\}$, which deterministically cause the corresponding state transitions, except that actions that would take the agent off the grid in fact leave the state unchanged. Thus, for instance, $p(6, -1|5, \texttt{right}) = 1$, $p(7, -1|7, \texttt{right}) = 1$, and $p(10, r|5, \texttt{right}) = 0$ for all $r \in \mathcal{R}$. This is an undiscounted, episodic task. The reward is -1 on all transitions until the terminal state is reached. The terminal state is shaded in the figure (although it is shown in two places, it is formally one state). The expected reward function is thus $r(s, a, s') = -1$ for all states s, s' and actions a. Suppose the agent follows the equiprobable random policy (all actions equally likely). The left side of Figure 4.1 shows the sequence of value functions $\{v_k\}$ computed by iterative policy evaluation. The final estimate is in fact v_π, which in this case gives for each state the negation of the expected number of steps from that state until termination. ■

2 - Policy improvement

v_k for the random policy

$k = 0$

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

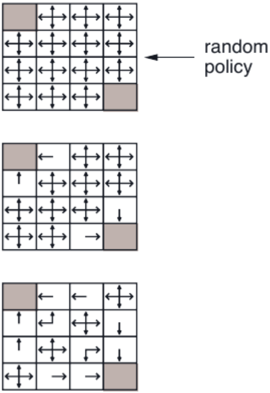
$k = 1$

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

greedy policy w.r.t. v_k



Iterative policy evaluation - Corresponding greedy policies

2 - Policy improvement

$k = 3$

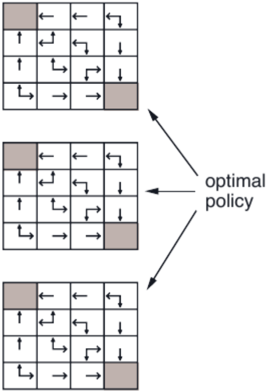
0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

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



Iterative policy evaluation - Corresponding greedy policies

3 - Policy iteration

Once a policy, π, has been improved using v_π to yield a better policy, π', we can then compute $v_{\pi'}$ and improve it again to yield an even better π''. We can thus obtain a sequence of monotonically improving policies and value functions:

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

3 - Policy iteration

Once a policy, π, has been improved using v_π to yield a better policy, π', we can then compute $v_{\pi'}$ and improve it again to yield an even better π''. We can thus obtain a sequence of monotonically improving policies and value functions:

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

Policy 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 $s \in \mathcal{S}$; $V(\mathit{terminal}) \doteq 0$

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 $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
   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

4 - Value iteration

Drawbacks of policy iteration :

- Each of its iterations involves policy evaluation.

4 - Value iteration

Drawbacks of policy iteration :

- Each of its iterations involves policy evaluation.
- If policy evaluation is done iteratively, convergence occurs only in the limit.

4 - Value iteration

Drawbacks of policy iteration :

- Each of its iterations involves policy evaluation.
- If policy evaluation is done iteratively, convergence occurs only in the limit.

Value iteration :

- Policy evaluation step of policy iteration can be truncated in several ways.

4 - Value iteration

Drawbacks of policy iteration :

- Each of its iterations involves policy evaluation.
- If policy evaluation is done iteratively, convergence occurs only in the limit.

Value iteration :

- Policy evaluation step of policy iteration can be truncated in several ways.
- One important special case is when policy evaluation is stopped after just one sweep (one update of each state).

4 - Value iteration

Drawbacks of policy iteration :

- Each of its iterations involves policy evaluation.
- If policy evaluation is done iteratively, convergence occurs only in the limit.

Value iteration :

- Policy evaluation step of policy iteration can be truncated in several ways.
- One important special case is when policy evaluation is stopped after just one sweep (one update of each state).
- In this case we call the algorithm : *value iteration*.

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

4 - Value iteration

Drawbacks of policy iteration :

- Each of its iterations involves policy evaluation.
- If policy evaluation is done iteratively, convergence occurs only in the limit.

Value iteration :

- Policy evaluation step of policy iteration can be truncated in several ways.
- One important special case is when policy evaluation is stopped after just one sweep (one update of each state).
- In this case we call the algorithm : *value iteration*.

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

- For arbitrary v_0, the sequence $\{v_k\}$ can be shown to converge to v_* (under the same conditions that guarantee the existence of v_*).

4 - Value iteration

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

- The value iteration is obtained simply by turning the Bellman optimality equation into an update rule (fixed ponit).

4 - Value iteration

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

- The value iteration is obtained simply by turning the Bellman optimality equation into an update rule (fixed ponit).
- The value iteration update is identical to the policy evaluation update, except that it requires the maximum to be taken over all actions.

4 - Value iteration

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

- The value iteration is obtained simply by turning the Bellman optimality equation into an update rule (fixed ponit).
- The value iteration update is identical to the policy evaluation update, except that it requires the maximum to be taken over all actions.
- Like policy evaluation, value iteration formally requires an infinite number of iterations to converge exactly to v_*.

4 - Value iteration

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

- The value iteration is obtained simply by turning the Bellman optimality equation into an update rule (fixed ponit).
- The value iteration update is identical to the policy evaluation update, except that it requires the maximum to be taken over all actions.
- Like policy evaluation, value iteration formally requires an infinite number of iterations to converge exactly to v_*.
- In practice, we stop once the value function changes by only a small amount in a sweep.

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

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

Asynchronous Dynamic Programming

Synchronous DP :

- Drawback to DP : It involve operations over the entire state set.

Asynchronous Dynamic Programming

Synchronous DP :

- Drawback to DP : It involve operations over the entire state set.
- If the state set is very large, then even a single sweep can be expensive.

Asynchronous Dynamic Programming

Synchronous DP :

- Drawback to DP : It involve operations over the entire state set.
- If the state set is very large, then even a single sweep can be expensive.

Asynchronous DP :

- The values of some states may be updated several times before the values of others are updated once.

Asynchronous Dynamic Programming

Synchronous DP :

- Drawback to DP : It involve operations over the entire state set.
- If the state set is very large, then even a single sweep can be expensive.

Asynchronous DP :

- The values of some states may be updated several times before the values of others are updated once.
- To converge correctly, an asynchronous algorithm can't ignore any state.

Asynchronous Dynamic Programming

Synchronous DP :

- Drawback to DP : It involve operations over the entire state set.
- If the state set is very large, then even a single sweep can be expensive.

Asynchronous DP :

- The values of some states may be updated several times before the values of others are updated once.
- To converge correctly, an asynchronous algorithm can't ignore any state.
- Asynchronous DP algorithms allow great flexibility in selecting states to update.

Asynchronous Dynamic Programming

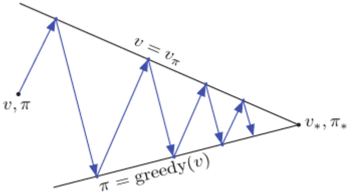
Synchronous DP :

- Drawback to DP : It involve operations over the entire state set.
- If the state set is very large, then even a single sweep can be expensive.

Asynchronous DP :

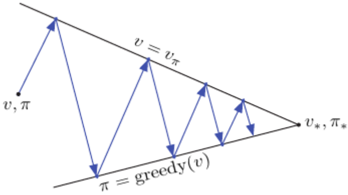
- The values of some states may be updated several times before the values of others are updated once.
- To converge correctly, an asynchronous algorithm can't ignore any state.
- Asynchronous DP algorithms allow great flexibility in selecting states to update.
- In the discounted case ($\gamma < 1$), the sequence could even be random, convergence is guaranteed.

Generalized Policy Iteration (GPI)



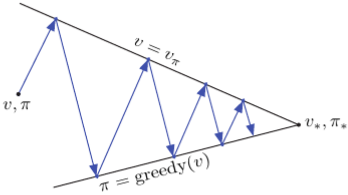
- GPI refers to the general idea of letting policy-evaluation and policy-improvement processes interact.

Generalized Policy Iteration (GPI)



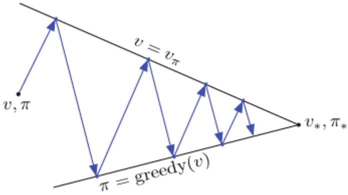
- GPI refers to the general idea of letting policy-evaluation and policy-improvement processes interact.
- Most of reinforcement learning methods are well described as GPI.

Generalized Policy Iteration (GPI)



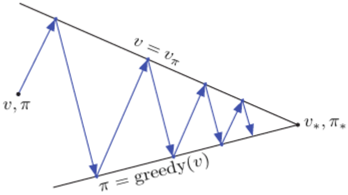
- GPI refers to the general idea of letting policy-evaluation and policy-improvement processes interact.
- Most of reinforcement learning methods are well described as GPI.
- The value function stabilizes only when it is consistent with the current policy.

Generalized Policy Iteration (GPI)



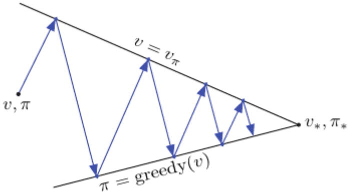
- GPI refers to the general idea of letting policy-evaluation and policy-improvement processes interact.
- Most of reinforcement learning methods are well described as GPI.
- The value function stabilizes only when it is consistent with the current policy.
- The policy stabilizes only when it is greedy w.r.t. the current value function.

Generalized Policy Iteration (GPI)



- GPI refers to the general idea of letting policy-evaluation and policy-improvement processes interact.
- Most of reinforcement learning methods are well described as GPI.
- The value function stabilizes only when it is consistent with the current policy.
- The policy stabilizes only when it is greedy w.r.t. the current value function.
- If both the value function and the policy stabilze, then the Bellman optimality equation holds.

Generalized Policy Iteration (GPI)



- GPI refers to the general idea of letting policy-evaluation and policy-improvement processes interact.
- Most of reinforcement learning methods are well described as GPI.
- The value function stabilizes only when it is consistent with the current policy.
- The policy stabilizes only when it is greedy w.r.t. the current value function.
- If both the value function and the policy stabilze, then the Bellman optimality equation holds.
- Thus the policy and value function are optimal.

Efficiency of Dynamic Programming

- In practice, DPcan be used with today's computers to solve MDPs with millions of states.

Efficiency of Dynamic Programming

- In practice, DPcan be used with today's computers to solve MDPs with millions of states.
- Both policy iteration and value iteration are widely used, and it is not clear which, if either, is better in general.

Efficiency of Dynamic Programming

- In practice, DPcan be used with today's computers to solve MDPs with millions of states.
- Both policy iteration and value iteration are widely used, and it is not clear which, if either, is better in general.
- In practice, these methods usually converge much faster than their theoretical worst-case run times
(particularly if they are started with good initial value functions or policies).

Efficiency of Dynamic Programming

- In practice, DPcan be used with today's computers to solve MDPs with millions of states.
- Both policy iteration and value iteration are widely used, and it is not clear which, if either, is better in general.
- In practice, these methods usually converge much faster than their theoretical worst-case run times
(particularly if they are started with good initial value functions or policies).
- With large state spaces, asynchronous DP methods are often preferred.

Efficiency of Dynamic Programming

- In practice, DPcan be used with today's computers to solve MDPs with millions of states.
- Both policy iteration and value iteration are widely used, and it is not clear which, if either, is better in general.
- In practice, these methods usually converge much faster than their theoretical worst-case run times
(particularly if they are started with good initial value functions or policies).
- With large state spaces, asynchronous DP methods are often preferred.
- Asynchronous methods and other variations of GPI can be applied in such cases.

Efficiency of Dynamic Programming

- In practice, DPcan be used with today's computers to solve MDPs with millions of states.
- Both policy iteration and value iteration are widely used, and it is not clear which, if either, is better in general.
- In practice, these methods usually converge much faster than their theoretical worst-case run times
  (particularly if they are started with good initial value functions or policies).
- With large state spaces, asynchronous DP methods are often preferred.
- Asynchronous methods and other variations of GPI can be applied in such cases.
- They may find good or optimal policies much faster than synchronous methods can.

Thank you !

nadir.farhi@univ-eiffel.fr