Reinforcement Learning

3. Dynamic programming

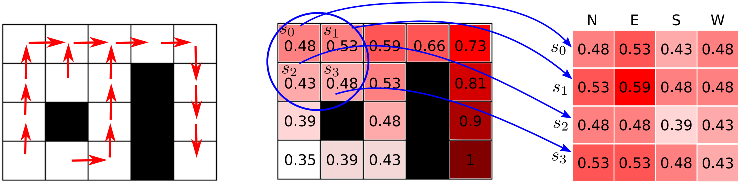
Olivier Sigaud

Sorbonne Université
http://people.isir.upmc.fr/sigaud

Introduction

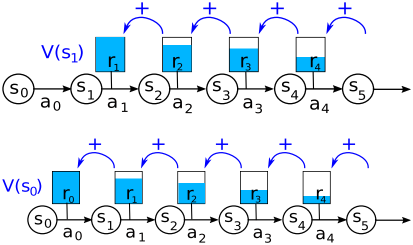
- Once we have defined an MDP
- Dynamic programming methods can find the optimal policy
- Assuming they know everything about the MDP tranzition and reward function
- Reinforcement Learning applies when the transition and reward functions are unknown
- To define dynamic programming methods, we need value functions

Value and action value functions



- The value function $V^\pi : S \rightarrow \mathbb{R}$ records the agregation of reward on the long run for each state (following policy π). It is a vector with one entry per state
- The action value function $Q^\pi : S \times A \rightarrow \mathbb{R}$ records the agregation of reward on the long run for doing each action in each state (and then following policy π). It is a matrix with one entry per state and per action
- In the remainder, we focus on V, trivial to transpose to Q

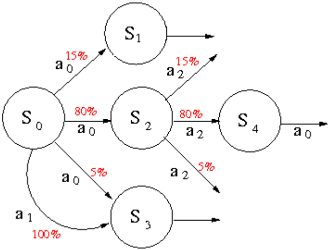
Bellman equation over a Markov chain: recursion



Given the discounted reward agregation criterion:

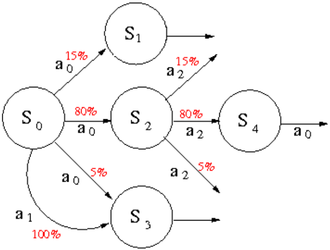
- $V(s_0) = r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + ...$
- $V(s_0) = r_0 + \gamma(r_1 + \gamma r_2 + \gamma^2 r_3 + ...)$
- $V(s_0) = r_0 + \gamma V(s_1)$
- More generally $V(s_t) = r_t + \gamma V(s_{t+1})$

Bellman equation: general case



- Generalisation of $V(s_t) = r_t + \gamma V(s_{t+1})$ over all possible trajectories
- The expectation of a random variable is the sum of the realizations weighted by their probabilities
- The realizations are the next states
- Deterministic π: $V^{\pi}(s) = r(s, \pi(s)) + \gamma \sum_{s'} p(s'|s, \pi(s)) V^{\pi}(s')$

Bellman equation: general case



- Generalisation of $V(s_t) = r_t + \gamma V(s_{t+1})$ over all possible trajectories
- The expectation of a random variable is the sum of the realizations weighted by their probabilities
- The realizations are the next states
- Stochastic π: $V^{\pi}(s) = \sum_a \pi(a|s)[r(s,a) + \gamma \sum_{s'} p(s'|s,a)V^{\pi}(s')]$

Recursive operators and convergence

- If we define an operator T such that $X_{n+1} \leftarrow TX_n$
- It T is contractive, then through repeated application of T, X_n will converge to some fixed point
- For instance, if T divides by 2, X_n converges to 0

The Bellman optimality operator (Value Iteration)

- We call Bellman optimality operator (noted T^*) the application

$$V_{n+1}(s) \leftarrow \max_{a \in A} \Big[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_n(s') \Big]$$

- If $\gamma < 1$, T^* is contractive
- By iterating, computes the value of the current policy
- The optimal value function is the fixed-point of T^*: $V^* = T^* V^*$
- Value iteration: $V_{n+1} \leftarrow T^* V_n$

Puterman, M. L. (2014) *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

The Bellman operator (Policy Iteration)

- We call Bellman operator (noted T^π) the application

$$V^\pi_{n+1}(s) \leftarrow r(s,\pi(s)) + \gamma \sum_{s'} p(s'|s,\pi(s)) V^\pi_n(s')$$

- If $\gamma < 1$, T is contractive
- Converges to optimal value and policy
- Policy Iteration:
  - Policy evaluation:
    $V^\pi_{n+1} \leftarrow T^\pi V^\pi_n$
  - Policy improvement:
    $\forall s \in S, \pi'(s) \leftarrow \arg\max_{a \in A} \sum_{s'} p(s'|s,a)[r(s,a) + \gamma V^\pi_n(s')]$
    or
    $\forall s \in S, \pi'(s) \leftarrow \arg\max_{a \in A} [r(s,a) + \gamma \sum_{s'} p(s'|s,a) V^\pi_n(s')]$
- Note: $\sum_{s',r} p(s',r|s,a)[r + \gamma V(s')] = r + \gamma \sum_{s'} p(s'|s,a) V(s')$

Value Iteration: the algorithm 1st method of dynamic programming

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

- Taken from Sutton & Barto, 2018, p. 83
- Reminder: $\sum_{s',r} p(s', r|s, a)[r + \gamma V(s')] = r + \gamma \sum_{s'} p(s'|s, a)V(s')$

Value Iteration in practice

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		1

$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} \left[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i(s') \right]$$

Value Iteration in practice

0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0		0.0
0.0		0.0		0.9
0.0	0.0	0.0		1

$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} \Big[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i(s') \Big]$$

Value Iteration in practice

0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0		0.81
0.0		0.0		0.9
0.0	0.0	0.0		1

$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} \Big[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i(s') \Big]$$

Value Iteration in practice

0.0	0.0	0.0	0.0	0.73
0.0	0.0	0.0		0.81
0.0		0.0		0.9
0.0	0.0	0.0		1

0.9*0.81

0.9*0.9

?

$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A}\Big[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i(s')\Big]$$

Value Iteration in practice

0.0	0.0	0.0	0.66	0.73
0.0	0.0	0.0		0.81
0.0		0.0		0.9
0.0	0.0	0.0		1

$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A}\Big[r(s,a) + \gamma \sum_{s'} p(s'|s,a)V_i(s')\Big]$$

Value Iteration in practice

0.0	0.0	0.59	0.66	0.73
0.0	0.0	0.0		0.81
0.0		0.0		0.9
0.0	0.0	0.0		1

$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A} \Big[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i(s') \Big]$$

Value Iteration in practice

0.0	0.53	0.59	0.66	0.73
0.0	0.0	0.53		0.81
0.0		0.0		0.9
0.0	0.0	0.0		1

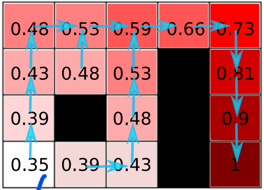
$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A}\left[r(s,a) + \gamma \sum_{s'} p(s'|s,a)V_i(s')\right]$$

Value Iteration in practice

0.48	0.53	0.59	0.66	0.73
0.43	0.48	0.53		0.81
0.39		0.48		0.9
0.35	0.39	0.43		1

$$\forall s \in S, V_{i+1}(s) \leftarrow \max_{a \in A}\left[r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i(s')\right]$$

Value Iteration in practice



Pourquoi en haut et pas a droite?

We have iterated on values, and determined a policy out of it (without necessarily representing it if using $Q(s, a)$)

Policy Iteration: the algorithm 2nd method for dynamic programming

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

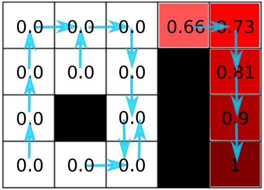
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

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

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $\quad$ *old-action* $\leftarrow \pi(s)$
   $\quad \pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
   $\quad$ 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

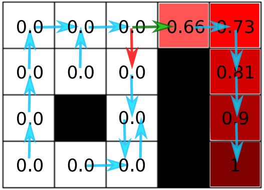
- Taken from Sutton & Barto, 2018, p. 80
- Note: $\sum_{s',r} p(s', r|s, a)[r + \gamma V(s')] = r + \gamma \sum_{s'} p(s'|s, a)V(s')$

Policy Iteration in practice



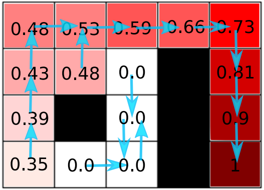
$$\forall s \in S, V_i(s) \leftarrow evaluate(\pi_i(s))$$

Policy Iteration in practice



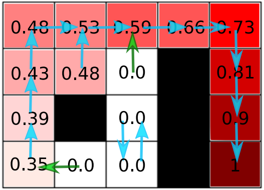
$$\forall s \in S, \pi_{i+1}(s) \leftarrow improve(\pi_i(s), V_i(s))$$

Policy Iteration in practice



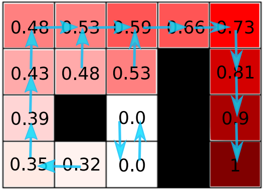
$$\forall s \in S, V_i(s) \leftarrow evaluate(\pi_i(s))$$

Policy Iteration in practice



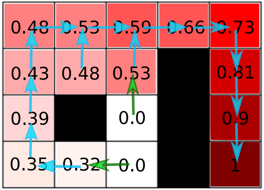
$$\forall s \in S, \pi_{i+1}(s) \leftarrow improve(\pi_i(s), V_i(s))$$

Policy Iteration in practice



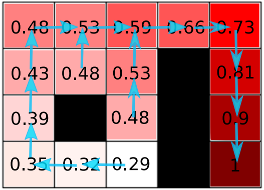
$$\forall s \in S, V_i(s) \leftarrow evaluate(\pi_i(s))$$

Policy Iteration in practice



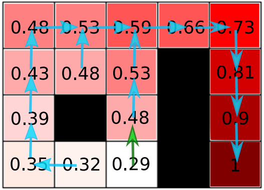
$$\forall s \in S, \pi_{i+1}(s) \leftarrow improve(\pi_i(s), V_i(s))$$

Policy Iteration in practice



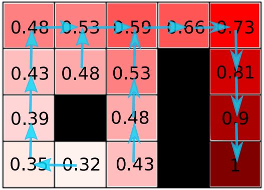
$$\forall s \in S, V_i(s) \leftarrow evaluate(\pi_i(s))$$

Policy Iteration in practice



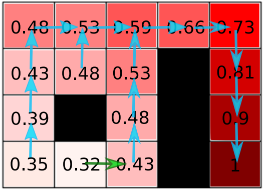
$$\forall s \in S, \pi_{i+1}(s) \leftarrow improve(\pi_i(s), V_i(s))$$

Policy Iteration in practice



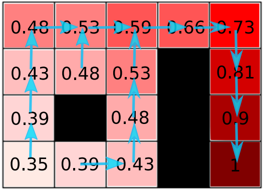
$$\forall s \in S, V_i(s) \leftarrow evaluate(\pi_i(s))$$

Policy Iteration in practice



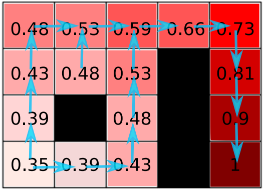
$$\forall s \in S, \pi_{i+1}(s) \leftarrow improve(\pi_i(s), V_i(s))$$

Policy Iteration in practice



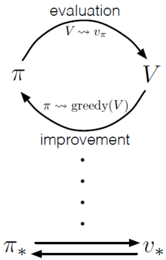
$$\forall s \in S, V_i(s) \leftarrow evaluate(\pi_i(s))$$

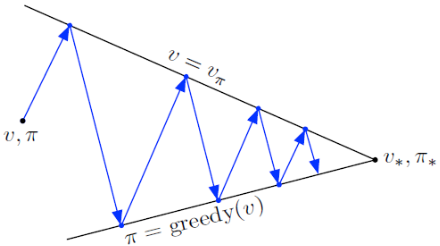
Policy Iteration in practice



Here we have managed a policy and a value representations at all steps

Generalized Policy Iteration



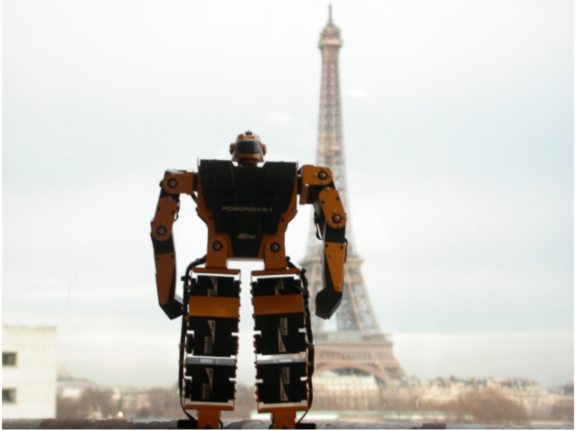


- Policy iteration evaluates each intermediate policy up to convergence. This is slow.
- Instead, evaluate the policy for N iterations, or even not for all states.
- Asynchronous dynamics programming: decoupling policy evaluation and improvement
- Taken from Sutton & Barto, 2018

Corresponding labs

- Implement value iteration with the V and the Q functions
- Implement policy iteration with the V and the Q functions
- Compare them
- Optional: study "Generalized policy iteration"

Any question?



Send mail to: Olivier.Sigaud@upmc.fr

Puterman, M. L.
Markov decision processes: discrete stochastic dynamic programming.
John Wiley & Sons, 2014.