ORCS 4529: Reinforcement Learning

Shipra Agrawal

Columbia University
Industrial Engineering and Operations Research

Contents I

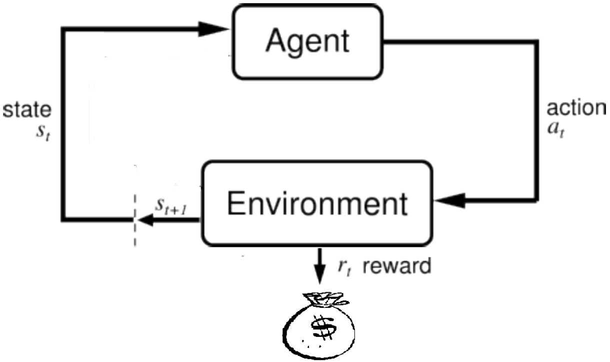
Markov Decision Process (MDP) definition

State space, Action space, Reward model, Transition model

$$(\mathcal{S}, \mathcal{A}, R, P)$$

Starting state s_1 or a distribution of starting state
Finite horizon H or inifinte horizon

- At $t = 1, 2, \ldots$, observe s_t, take action a_t, observe reward r_t and next state s_{t+1}.



Markov Property of MDP

- At $t = 1, 2, \ldots$, observe s_t, take action a_t, observe reward r_{t+1} and next state s_{t+1}.
- Markov Property:

Solution Concept: Policy

- Markovian Policy vs. History Dependent policy
- Deterministic Policy vs. Randomized Policy
- Stationary vs. Non-stationary policy

Stationary Policy π

$$\pi : \mathcal{S} \to \mathcal{A} \text{ or } \pi : \mathcal{S} \to \Delta^{\mathcal{A}}$$

- Markov reward process (compare to Markov chains)
- Stationary distribution d^π of a stationary policy π

Goal of an MDP: Finite Horizon

Find (possibly non-stationary) policy $\pi = (\pi_1, \ldots, \pi_H)$ that maximizes 'Value' starting from state s_1.

- Episodic or finite horizon setting.

$$V^\pi(s_1) = \mathbb{E}[\sum_{t=1}^{H} \gamma^{t-1} r_t | s_1; a_t = \pi_t(s_t)]$$

Optimal policy and value depend critically on s_1, H.
$0 \leq \gamma \leq 1$.

Goal of an MDP: Infinite Horizon

Find (possibly non-stationary) policy $\pi = (\pi_1, \pi_2 \ldots, \pi_t, \ldots,)$ that maximizes 'Gain' or 'Value' starting from state s_1.

- Infinite horizon expected total reward (Value).

$$V^\pi(s_1) = \lim_{T\to\infty} \mathbb{E}[\sum_{t=1}^{T} \gamma^{t-1} r_t | s_1; a_t = \pi_t(s_t)]$$

- Infinite horizon discounted sum of rewards (Value).

$$V^\pi(s_1) = \lim_{T\to\infty} \mathbb{E}[\sum_{t=1}^{T} \gamma^{t-1} r_t | s_1; a_t = \pi_t(s_t)]$$

- Infinite horizon average reward (gain):

$$\rho^\pi(s_1) = \lim_{T\to\infty} \mathbb{E}[\frac{1}{T} \sum_{t=1}^{T} r_t | s_1; a_t = \pi_t(s_t)]$$

Optimal Policy

A policy that maximizes the gain or value starting from the starting state.

Is optimal policy Markovian? Stationary?

Optimal Policy

A policy that maximizes the gain or value starting from the starting state.

Is optimal policy Markovian? Stationary?

Assume finite or countable states and actions.

- If an optimal policy exists, there always exists a **Markovian policy** that is optimal.
- In all three **infinite horizon settings**, if an optimal policy exists, there always exists a **stationary policy** that is optimal.

Reference to proofs available in lecture notes.

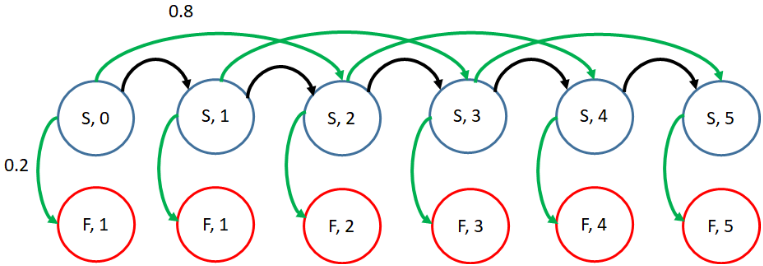
MDP formulation Example 1

Robot learning to move on a line

- Three actions: walk or run or stay.
- On walking: the robot to move one step without falling.
- On running: robot might move two steps forward (80% chance), or fall (20% chance). Once the robot falls, it cannot get up.
- Once a target position (say 5 steps away from the starting position) is reached, the robot stays there.
- Aim: move forward on the line, quickly and without falling, and reach the target position.

MDP formulation Example 1

Robot learning to walk



Reward model? Goal? Policies?

Example 2: Inventory control MDP

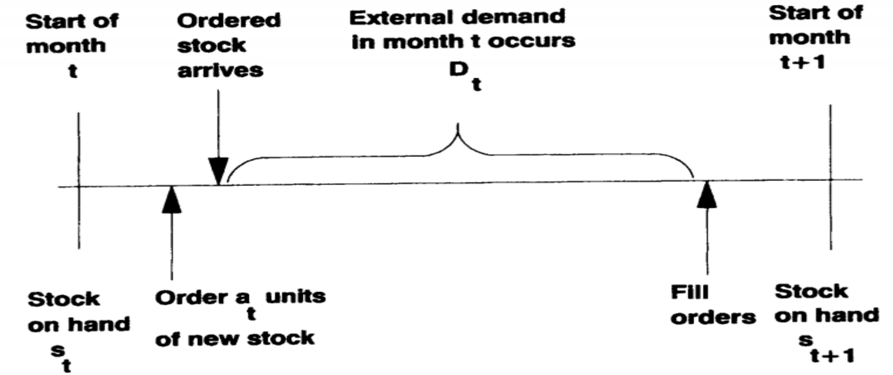


Figure: Timing of events in an inventory problem (Figure taken from Puterman:1994.)

MDP formulation

Example 3: Tabular MDP

Robot learning to walk

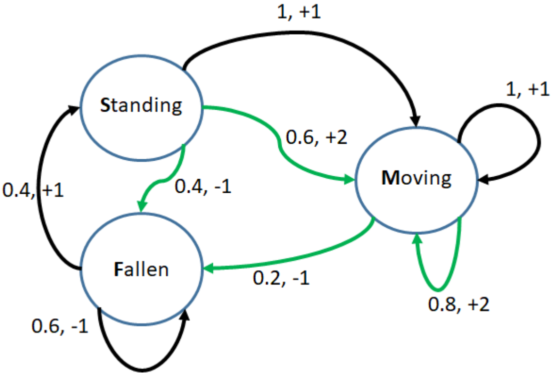


Figure: A simple MDP for the robot toy example

MDP formulation

Contents I

Solving an MDP: Finite horizon

$$\max_{\pi} \mathbb{E}[\sum_{t=1}^{H} \gamma^{t-1} r_t | s_1; a_t = \pi_t(s_t)]$$

where maximum is taken over all (non-stationary) policies $\pi = (\pi_1, \ldots, \pi_k)$

Solving an MDP: Finite horizon

Dynamic programming algorithm using optimal substructure property

Define for all $s \in \mathcal{S}$, $k = 1, \ldots, H$,

$$V_k^*(s) = \max_{\pi=\{\pi_t\}} \mathbb{E}[\sum_{t=1}^{k} \gamma^{t-1} r_t | s_1 = s]$$

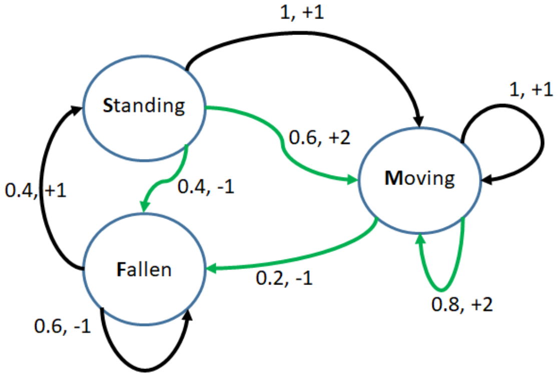
Then, optimal substructure property:

$$V_k^*(s) = \max_a R(s,a) + \gamma \sum_{s'} P(s,a,s') V_{k-1}^*(s')$$

Dynamic programming uses this property backwards starting from $V_1^*(\cdot)$ to finally compute $V_H^*(\cdot)$, the optimal value for horizon H.

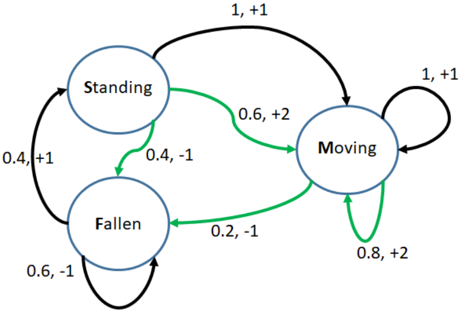
Proof

Solve the Robot MDP



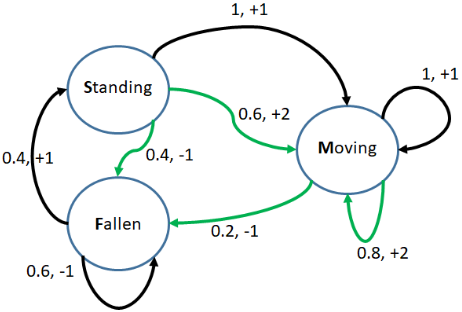
Let's optimize the expected sum of rewards ($\gamma = 1$) for horizon $H = 4$.

Dynamic Programming



$$R = \begin{bmatrix} -0.2 & 0 \\ 1 & 0.8 \\ 1 & 1.4 \end{bmatrix}$$

Dynamic Programming

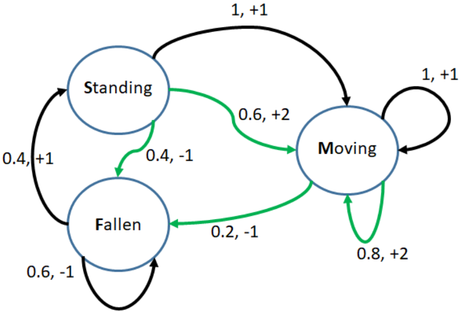


$$R = \begin{bmatrix} -0.2 & 0 \\ 1 & 0.8 \\ 1 & 1.4 \end{bmatrix}$$

$V_1^*(\cdot)$ is simply immediate reward maximization,

$$\begin{aligned} V_1^*(F) &= 0(\text{fast action/do nothing}) \\ V_1^*(S) &= 1(\text{slow action}) \\ V_1^*(M) &= 1.4(\text{fast action}) \end{aligned}$$

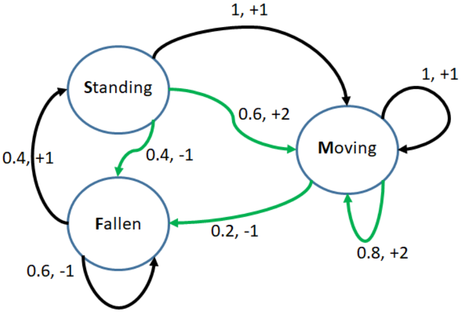
Dynamic Programming



$$R = \begin{bmatrix} -0.2 & 0 \\ 1 & 0.8 \\ 1 & 1.4 \end{bmatrix}$$

$$\begin{aligned} V_2^*(F) &= \max\{-0.2 + 0.4 \times 1 + 0.6 \times 0, 0 + 0\} = 0.2(\text{slow action}) \\ V_2^*(S) &= \max\{1 + 1.4, 0.8 + 0.6 \times 1.4 + 0.4 \times 0\} = 2.4(\text{slow action}) \\ V_2^*(M) &= \max\{1 + 1.4, 1.4 + 0.8 \times 1.4 + 0.2 \times 0\} = 2.56(\text{fast action}) \end{aligned}$$

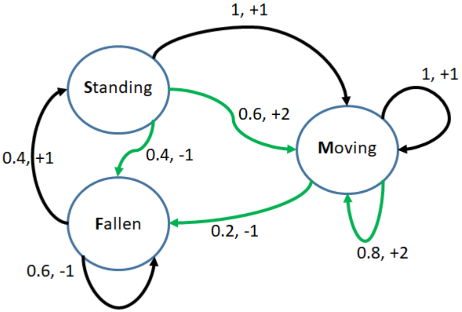
Dynamic Programming



$$R = \begin{bmatrix} -0.2 & 0 \\ 1 & 0.8 \\ 1 & 1.4 \end{bmatrix}$$

$$\begin{aligned} V_3^*(F) &= \max\{-0.2 + 0.4 \times 2.4 + 0.6 \times 0.2, 0 + 0.2\} = 0.88(\text{slow action}) \\ V_3^*(S) &= \max\{1 + 2.56, 0.8 + 0.6 \times 2.56 + 0.4 \times 0.2\} = 3.56(\text{slow action}) \\ V_3^*(M) &= \max\{1 + 2.56, 1.4 + 0.8 \times 2.56 + 0.2 \times 0.2\} = \max\{3.56, 3.488\} = 3.56(\text{slow}) \end{aligned}$$

Dynamic Programming



$$R = \begin{bmatrix} -0.2 & 0 \\ 1 & 0.8 \\ 1 & 1.4 \end{bmatrix}$$

$$\begin{aligned}
V_4^*(F) &= \max\{-0.2 + 0.4 \times 3.56 + 0.6 \times 0.88, 0 + 0.88\} = \max\{1.752, 0.88\} = 1.752(\text{slo} \\
V_4^*(S) &= \max\{1 + 3.56, 0.8 + 0.6 \times 3.56 + 0.4 \times 0.88\} = \max\{4.56, 3.24\} = 4.56(\text{slow}) \\
V_4^*(M) &= \max\{1 + 3.56, 1.4 + 0.8 \times 3.56 + 0.2 \times 0.88\} = \max\{4.56, 4.4\} = 4.56(\text{slow})
\end{aligned}$$

Contents I

Infinite horizon settings: Bellman optimality equations

We still use the memoization idea but fixed point equations instead of recursive equations.

- Memoization idea in finite horizon: Given remaining horizon k, the optimal value from a state s is fixed irrespective of how you arrived there.
- Memoization in infinite horizon: ~~Given remaining horizon k,~~ the optimal value from a state s is fixed irrespective of how you arrived there.

Bellman equations for value of a stationary policy

Infinite horizon discounted reward setting

Value of stationary policy π from state s (discount factor $\gamma < 1$):

$$V_\gamma^\pi(s) := \lim_{T\to\infty} \mathbb{E}[\sum_{t=1}^{T} \gamma^{t-1} r_t | s_1 = s; a_t = \pi(s_t)]$$

Bellman equations for value of a stationary policy

Infinite horizon discounted reward setting

Value of stationary policy π from state s (discount factor $\gamma < 1$):

$$V_\gamma^\pi(s) := \lim_{T\to\infty} \mathbb{E}[\sum_{t=1}^{T} \gamma^{t-1} r_t | s_1 = s; a_t = \pi(s_t)]$$

Bellman equations

$$V_\gamma^\pi(s) = \mathbb{E}_{a\sim\pi(s), s'\sim P(s,a)} \left[R(s,a,s') + \gamma V_\gamma^\pi(s')\right]$$

Vector form for finite state space

$$V_\gamma^\pi = \mathbf{R}^\pi + \gamma P^\pi V_\gamma^\pi$$

Proof

Bellman Optimality Equations

Infinite horizon discounted reward setting

Define optimal value:

$$V^*_\gamma(s) \quad = \quad \sup_{\pi=\{\pi_t\}} \mathbb{E}[r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \ldots | s_1 = s]$$

Bellman optimality equations:

$$V^*_\gamma(s) = \max_a R(s,a) + \gamma \sum_{s'} P(s,a,s') V^*_\gamma(s')$$

More generally,

$$V^*_\gamma(s) = \max_a R(s,a) + \gamma \mathbb{E}_{s' \sim P(s,a)}[V^*_\gamma(s')]$$

Proof

Optimal Policy

Infinite horizon discounted reward setting

$$\pi^*(s) := \arg\max_a R(s,a) + \gamma \mathbb{E}_{s' \sim P(s,a)}[V_\gamma^*(s')]$$

Value of optimal policy can also be computed as

$$V_\gamma^* = (I - \gamma P^{\pi^*})^{-1} R^{\pi^*}$$

Inverse exists for $\gamma < 1$

Finding an optimal policy

Solving the Bellman equations

Assume finite state space and action space (aka 'tabular' setting).

- Linear programming
- Iterative algorithms

LP for solving Bellman Equations

LP for solving Bellman Equations

The fixed point of Bellman optimality equations can be found by solving the following linear program.

$$
\begin{aligned}
\min_{\mathbf{v} \in \mathbb{R}^S} \quad & \textstyle\sum_s v_s \\
\text{subject to} \quad & v_s \geq R(s,a) + \gamma P(s,a)^\top \mathbf{v} \quad \forall a, s
\end{aligned}
$$

Proof

Iterative algorithms

Finite state space and action space

- Value iteration: Iteratively improve estimate of optimal value vector $[V^*(1), \dots, V^*(S)]$.
- Q-value iteration: Iteratively improve the estimate of Q-values $[Q^*(s, a), s \in \mathcal{S}, a \in \mathcal{A}]$. (to be defined)
- Policy iteration: Iteratively improve estimate of optimal policy $[\pi^*(1), \dots, \pi^*(S)]$.

Iterative algorithms: Value Iteration

Infinite horizon discounted reward setting

Estimate the optimal value vector

1. Start with an arbitrary initialization $\mathbf{v}^0$. Specify $\epsilon > 0$
2. **Repeat** for $k = 1, 2, \ldots$ **until** :
   - Update value vector estimate $\mathbf{v}^k$
3. Output a near-optimal policy

Iterative algorithms: Value Iteration

Infinite horizon discounted reward setting

Estimate the optimal value vector

1. Start with an arbitrary initialization $\mathbf{v}^0$. Specify $\epsilon > 0$
2. **Repeat** for $k = 1, 2, \ldots$ **until** :
   - for every $s \in S$, improve the value vector as:

$$\mathbf{v}^k(s) = \max_{a \in A} R(s,a) + \gamma \sum_{s'} P(s,a,s')\mathbf{v}^{k-1}(s'), \tag{1}$$

3. Output a near-optimal policy

Iterative algorithms: Value Iteration

Infinite horizon discounted reward setting

Estimate the optimal value vector

1. Start with an arbitrary initialization $\mathbf{v}^0$. Specify $\epsilon > 0$
2. **Repeat** for $k = 1, 2, \ldots$ **until** $\|\mathbf{v}^k - \mathbf{v}^{k-1}\|_\infty \leq \epsilon\frac{(1-\gamma)}{2\gamma}$:
   - for every $s \in S$, improve the value vector as:

$$\mathbf{v}^k(s) = \max_{a \in A} R(s,a) + \gamma \sum_{s'} P(s,a,s')\mathbf{v}^{k-1}(s'), \qquad (1)$$

3. Output a near-optimal policy

Iterative algorithms: Value Iteration

Infinite horizon discounted reward setting

Estimate the optimal value vector

1. Start with an arbitrary initialization $\mathbf{v}^0$. Specify $\epsilon > 0$
2. **Repeat** for $k = 1, 2, \ldots$ **until** $\|\mathbf{v}^k - \mathbf{v}^{k-1}\|_\infty \leq \epsilon \frac{(1-\gamma)}{2\gamma}$:
   - for every $s \in S$, improve the value vector as:

$$\mathbf{v}^k(s) = \max_{a \in A} R(s,a) + \gamma \sum_{s'} P(s,a,s')\mathbf{v}^{k-1}(s'), \qquad (1)$$

3. Output a near-optimal policy

$$\pi(s) \in \arg\max_a R(s,a) + \gamma P(s,a)^\top \mathbf{v}^k \qquad (2)$$

Bellman operator

$L, L^\pi : \mathbb{R}^S \to \mathbb{R}^S.$

Bellman operator

$L, L^\pi : \mathbb{R}^S \to \mathbb{R}^S$.

$$
\begin{aligned}
[LV](s) &:= \max_{a \in A} R(s,a) + \gamma \sum_{s'} P(s,a,s') V(s') \\
[L^\pi V](s) &:= \mathbb{E}_{a \in \pi(s)} [R(s,a) + \gamma \sum_{s'} P(s,a,s') V^\pi(s')]
\end{aligned}
$$

Iterative algorithms: Value Iteration

Infinite horizon discounted reward setting

Estimate the optimal value vector

1. Start with an arbitrary initialization $\mathbf{v}^0$. Specify $\epsilon > 0$
2. **Repeat** for $k = 1, 2, \ldots$ **until** $\|\mathbf{v}^k - \mathbf{v}^{k-1}\|_\infty \leq \epsilon\frac{(1-\gamma)}{2\gamma}$:

$$\mathbf{v}^k = L\mathbf{v}^{k-1}$$

3. Output a near-optimal policy

$$\pi(s) \in \arg\max_a R(s,a) + \gamma P(s,a)^\top \mathbf{v}^k$$

Analysis

Theorem (Theorem 6.3.3, Section 6.3.2 in Puterman:1994)

The convergence rate of the above algorithm is linear at rate γ. Specifically,

$$\|\mathbf{v}^k - V^*\|_\infty \leq \frac{\gamma^k}{1-\gamma}\|v^1 - v^0\|_\infty$$

Further, let π^k be the arg max policy defined by v^k. Then,

$$\|V^{\pi^k} - V^*\|_\infty \leq \frac{2\gamma^k}{1-\gamma}\|v^1 - v^0\|_\infty$$

Contraction property of L-operator

$$\|Lv - Lu\|_\infty \leq \gamma \|v - u\|_\infty.$$

$$\|L^\pi v - L^\pi u\|_\infty \leq \gamma \|v - u\|_\infty.$$

Proof of the value iteration convergence theorem

Q-values and Q-value-iteration

Q-values are defined as values after fixing the first action

- $Q^*(s, a)$ is defined the expected utility on taking action a in state s, and thereafter acting optimally.

$$Q^*(s, a) := R(s, a) + \sum_{s' \in \mathcal{S}} P(s, a, s') V^*(s')$$

$$V^*(s) := \max_a Q^*(s, a)$$

- Bellman Optimality Equation

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s, a, s') \left(\max_{a'} Q^*(s', a') \right)$$

Q-values and Q-value-iteration

Q-values are defined as values after fixing the first action

- $Q^*(s, a)$ is defined the expected utility on taking action a in state s, and thereafter acting optimally.

$$Q^*(s, a) := R(s, a) + \sum_{s' \in \mathcal{S}} P(s, a, s') V^*(s')$$

$$V^*(s) := \max_a Q^*(s, a)$$

- Bellman Optimality Equation

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s, a, s') \left(\max_{a'} Q^*(s', a') \right)$$

- $Q^\pi(s, a)$ is defined the expected utility on taking action a in state s, and thereafter playing policy π.

$$Q^\pi(s, a) := R(s, a) + \sum_{s' \in \mathcal{S}} P(s, a, s') V^\pi(s')$$

$$V^\pi(s) = \mathbb{E}_{a \in \pi(s)}[Q^\pi(s, a)]$$

Q-values and Q-value iteration

Q-value iteration estimates $Q^*(s, a)$ s for all s, a.
(instead of $V^*(s)$ for all s in value iteration)

Why?

Q-values and Q-value iteration

Q-value iteration estimates $Q^*(s, a)$ s for all s, a.
(instead of $V^*(s)$ for all s in value iteration)

Why?
No need to know the MDP model to compute optimal policy:

$$\pi^*(s) = \arg\max_a Q^*(s, a)$$

Important in the learning setting when we don't know the model.

Q-value iteration

1. Start with an arbitrary initialization $\mathbf{Q}^0 \in \mathbb{R}^{S\times A}$.
2. In every iteration k, improve the Q-value vector as:

$$\mathbf{Q}^k(s,a) = R(s,a) + \gamma \sum_{s'} P(s,a,s') \left(\max_{a'} Q^{k-1}(s',a') \right), \forall s, a$$

3. Stop if $\|Q^k - Q^{k-1}\|_\infty$ is small.
4. Output policy π^k defined as $\pi^k(s) = \arg\max_a Q^k(s,a)$

Q-value iteration

1. Start with an arbitrary initialization $\mathbf{Q}^0 \in \mathbb{R}^{S \times A}$.
2. In every iteration k, improve the Q-value vector as:

$$\mathbf{Q}^k(s,a) = R(s,a) + \gamma \sum_{s'} P(s,a,s') \left(\max_{a'} Q^{k-1}(s',a') \right), \forall s,a$$

3. Stop if $\|Q^k - Q^{k-1}\|_\infty$ is small.
4. Output policy π^k defined as $\pi^k(s) = \arg\max_a Q^k(s,a)$

Convergence proof follows from value iteration convergence.

Policy iteration

Directly estimate the optimal policy π^*.
Use that at π^*

$$\pi^*(s) = \arg\max_a R(s, a) + \gamma \sum_{s'} P(s, a, s') V^{\pi^*(s')}, \forall s$$

Greedy update should give back π^*

Policy iteration using values

1. Initialize policy π^0.
2. In every iteration $k = 0, 1, \ldots,$,
   - (Policy evaluation) Compute value $V^{\pi^k}(s)$, the value of policy π^k for every state s.
   - (Greedy Policy improvement) Compute new policy

$$\pi^{k+1}(s) := \arg\max_a R(s,a) + \gamma \sum_{s'} P(s,a,s') V^{\pi^k}(s'), \forall s$$

3. Stop when $\pi^{k+1} = \pi^k$.

Relaxed stopping criteria: not much change in policy or its value.

Policy iteration using Q-values

1. Initialize policy π^0.
2. In every iteration $k = 0, 1, 2, \ldots,$
   - (Policy evaluation) Compute value $Q^{\pi^k}(s, a)$, the Q-values of policy π^k for all s, a.
   - (Greedy Policy improvement) Compute new policy

$$\pi^{k+1}(s) := \arg\max_a Q^{\pi_k}(s, a), \forall s$$

3. Stop when $\pi^{k+1} = \pi^k$.

Relaxed stopping criteria: not much change in policy or its value.

Policy-iteration vs. Value-iteration

+ separate the policy evaluation (learning) and the improvement (optimization) steps
+ can actually be faster if estimating the performance of a fixed policy is much easier than finding an optimal policy
+ can warm start if there is a known good initial policy to start from and improve upon
+ always maintains a good policy
- parameterizing a policy (function) can be more difficult/complex than parameterizing a value (vector)
- If policy evaluation is done by an iterative method like value iteration then policy iteration is slower.

Policy iteration convergence proof

Theorem

For the policy π^k computed in the k^{th} iteration of policy iteration, we have

$$\|V^{\pi^k} - V^*\|_\infty \leq \gamma^k \|V^{\pi_0} - V^*\|_\infty$$

- Compare it to the value iteration convergence.
- Why does this look much better? Is it really much better?

Policy iteration converence

How much does the policy improve in one step?

Lemma

$$V^{\pi^{k+1}} \geq LV^{\pi^k}$$

Once we can prove the above, the proof is similar to value iteration.

Why is this not trivial (unlike value iteration)?

Contents I

Infinite horizon Average reward goal

Find a policy π that maximizes the average reward under that policy

$$\rho^\pi(s_1) = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}[\sum_{t=1}^{T} r_t | s_1 = s; a_t = \pi(s)]$$

Connections to finite horizon reward

$$\rho^\pi(s) = \lim_{T\to\infty} \frac{1}{T} V_T^\pi(s)$$

Connections to discounted reward

$$\rho^\pi(s) = \lim_{\gamma\to 1} (1-\gamma) V_\gamma^\pi(s)$$

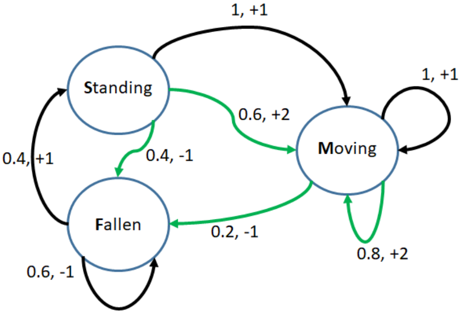
Bias of a policy

For average reward case, an important quantity is bias of a policy π from state s is defined as

$$h^\pi(s) = \lim_{T\to\infty} \mathbb{E}[\sum_{t=1}^{T}(r_t - \rho^\pi(s_t))|s_1 = s; a_t = \pi(s_t)]$$

The limit in above is Cesaro limit which exists. More details in Section 8.2 of Puterman:1994.

Example



For each state, compute bias of the policy that plays slow action in all states.

Connection of Bias and value

Under a policy π, if two states s, s' are in the same irreducible class (i.e., can be reached from each other in finite expected time) then

- Finite time value:

$$h^\pi(s) - h^\pi(s') = \lim_{T\to\infty}(V_T^\pi(s) - V_T^\pi(s'))$$

  where $V_T^\pi(s) = \mathbb{E}[\sum_{t=1}^T r_t | s_1 = s]$
- Discounted value:

$$h^\pi(s) - h^\pi(s') = \lim_{\gamma\to 1}(V_\gamma^\pi(s) - V_\gamma^\pi(s'))$$

When comparing outcomes from two different states, bias behaves like the value function in the other settings.

Bellman equations in average reward case

Given a policy π such that all s, s' are reachable from each other in finite time.

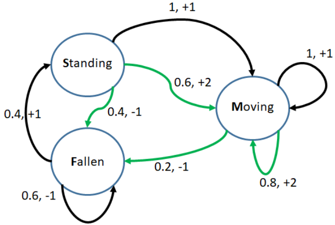
$$\rho^\pi(s) + h^\pi(s) = \mathbb{E}_{a\sim\pi(s), s'\sim P(s,a)}\left[R(s,a,s') + h^\pi(s')\right], \forall s$$

Or, in compact notation:

$$\mathbf{h}^\pi + \rho^\pi = \mathbf{R}^\pi + P^\pi \mathbf{h}^\pi$$

Example

Consider the robot example. Check that the bias and average reward (aka gain) of the policy that always plays slow actions satisfy the Bellman equations stated above.



$$R^\pi = \begin{bmatrix} -0.2 \\ 1 \\ 1 \end{bmatrix}, P^\pi = \begin{bmatrix} 0.6 & 0.4 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Bellman optimality equations

Assume communicating MDP.

Definition

An MDP is called **communicating** if for any two states s, s', there exists a policy such that the expected number of steps to reach s' from s is finite.

Theorem

For communicating MDP, for optimal gain policy $\rho^*(s) = \rho^*(s') = \rho^*$, i.e., optimal average infinite horizon reward does not depend on the starting state.

Bellman Optimality Equations for average reward case

Assuming communicating MDP, gain and bias ρ, h of optimal policy satisfies the following equations:

$$\rho + h(s) = \max_a R(s, a) + \sum_{s' \in \mathcal{S}} P(s, a, s')h(s'), \forall s$$

Bellman Optimality Equations for average reward case

Assuming communicating MDP, gain and bias ρ, h of optimal policy satisfies the following equations:

$$\rho + h(s) = \max_a R(s,a) + \sum_{s' \in \mathcal{S}} P(s,a,s')h(s'), \forall s$$

Also, for any feasible solution (ρ, h) to the above equations, we can get an optimal policy π^* defined as

$$\pi^*(s) \in \arg\max_a R(s,a) + \sum_{s'} P(s,a,s')h(s'),$$

with $\rho = \rho^{\pi^*}$ and $h = h^{\pi^*} + c\mathbf{e}$ for some constant c.

- Note that to compute the optimal policy we need to just know the bias vector h that satisfies Bellman equations.

Solving Bellman equations: Linear Program

$$\begin{array}{ll} \min\limits_{\rho \in R, \mathbf{h} \in \mathbb{R}^S} & \rho \\ \text{subject to} & \rho \geq R(s,a) + \sum_{s'} P(s,a,s') h_{s'} - h_s \quad \forall a, s \end{array}$$

Solving Bellman equations: Linear Program

$$
\begin{aligned}
\min_{\rho \in R, \mathbf{h} \in \mathbb{R}^S} \quad & \rho \\
\text{subject to} \quad & \rho \geq R(s,a) + \sum_{s'} P(s,a,s') h_{s'} - h_s \quad \forall a, s
\end{aligned}
$$

Write the dual LP for better interpretation:

$$
\begin{aligned}
\max_{q} \quad & \sum_{s,a} q(s,a) R(s,a) \\
& \sum_{s,a} P(s,a,s') q(s,a) - \sum_{a} q(s',a) = 0 \quad \forall s' \\
& \sum_{s,a} q(s,a) = 1 \\
& q(s,a) \geq 0 \quad \forall s, a
\end{aligned}
$$

That is, find the stationary distribution $q(s,a)$ that maximizes the expected reward.

Solving Bellman equations: value iteration/policy iteration

- Same algorithm but with $\gamma = 1$.
- Instead of estimating the value vector, we are updating and estimating bias.
- linear convergence with rate γ is not guaranteed since $\gamma = 1$. The convergence rate depends on the properties of the transition matrix.
- a sufficient condition for linear convergence is

$$\alpha := \max_{s,s',a,a'} \sum_{j \in S} \min\{P(s,a,j), P(s',a',j)\} > 0$$

then linear convergence with rate $1 - \alpha$. .

Contents I

Reinforcement Learning algorithms

$$\begin{aligned} \text{RL} &== \text{MDP + unknown model} \\ &== \text{Value/Policy iteration + sampling} \\ &\quad\ \text{OR} \\ &\quad\ \text{Direct function optimization from samples} \end{aligned}$$

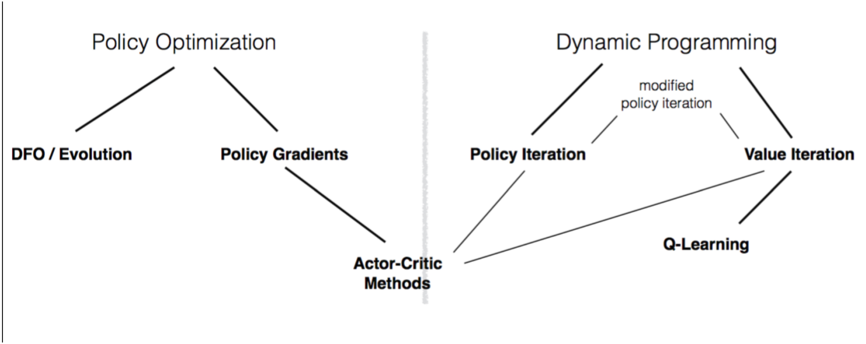


Figure: Algorithms for RL (Drawing taken from Pieter Abbeel's slides)