Markov Decision Processes

Johanni Brea

5 March 2024

Artificial Neural Networks/Reinforcement Learning CS-456

EPFL

Introduction

Many RL papers contain a background section like the following one:

The Option-Critic Architecture

Pierre-Luc Bacon, Jean Harb, Doina Precup
Reasoning and Learning Lab, School of Computer Science
McGill University
{pbacon, jharb, dprecup}@cs.mcgill.ca

Preliminaries and Notation

A Markov Decision Process consists of a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a transition function $\mathrm{P} : \mathcal{S} \times \mathcal{A} \to (\mathcal{S} \to [0, 1])$ and a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. For convenience, we develop our ideas assuming discrete state and action sets. However, our results extend to continuous spaces using usual measure-theoretic assumptions (some of our empirical results are in continuous tasks). A (Markovian stationary) *policy* is a probability distribution over actions conditioned on states, $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. In discounted problems, the value function of a policy π is defined as the expected return: $V_\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s_0 = s\right]$ and its action-value function as $Q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s_0 = s, a_0 = a\right]$, where $\gamma \in [0, 1)$ is the *discount factor*. A policy π is *greedy* with respect to a given action-value function Q if

In this lecture you will learn

1. what a Markov Decision Process is.
2. how MDPs can be solved with dynamic programming or linear programming.
3. how future discounted MDPs can be solved with value iteration or policy iteration.

Recommended reading:
Sutton & Barto, Chapters 3 & 4
Algorithms of Reinforcement Learning
`http://www.ualberta.ca/~szepesva/papers/RLAlgsInMDPs.pdf`

Table of Contents

Markov Decision Processes

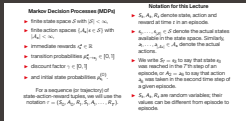
Markov Decision Processes (MDPs)

- finite state space $\mathcal{S}$ with $|\mathcal{S}| < \infty$,
- finite action spaces $\{\mathcal{A}_s | s \in \mathcal{S}\}$ with $|\mathcal{A}_s| < \infty$,
- immediate rewards $r_s^a \in \mathbb{R}$
- transition probabilities $p^a_{s_i \to s_j} \in [0, 1]$
- discount factor $\gamma \in [0, 1]$
- and initial state probabilities $p^{(0)}_{s_i}$.

For a sequence (or trajectory) of state-action-reward tuples, we will use the notation $\tau = (S_0, A_0, R_1, S_1, A_1, \ldots, R_T)$.

Notation for this Lecture

- S_t, A_t, R_t denote state, action and reward at time t in an episode.
- $s_1, \ldots, s_{|\mathcal{S}|} \in \mathcal{S}$ denote the actual states available in the state space. Similarly, $a_1, \ldots, a_{|\mathcal{A}_s|} \in \mathcal{A}_s$ denote the actual actions.
- We write $S_7 = s_3$ to say that state s_3 was reached in the 7'th step of an episode, or $A_2 = a_6$ to say that action a_6 was taken in the second time step of a given episode.
- S_t, A_t, R_t are random variables; their values can be different from episode to episode.

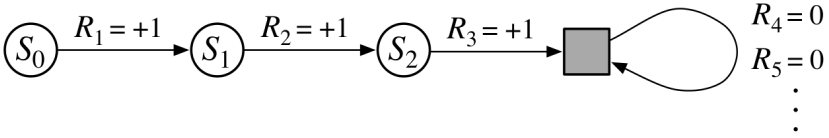


Notes

- The Markov Decision Processes can also be defined for continuous state and action spaces, but we restrict ourselves here to finite (and thus discrete) state and action spaces.
- In general, the available actions can depend on the state (it is not possible to advance when standing in front of a wall). Sometimes the action spaces are independent of the state; in this case we just write $\mathcal{A}$ for the action space.
- The transition probabilities have the property $\sum_{s_j \in \mathcal{S}} p^a_{s_i \to s_j} = 1, \forall a \in \mathcal{A}_{s_i}, s_i \in \mathcal{S}$.
- Sometimes rewards are considered stochastic or dependent on the next state $R^a_{s_i \to s_j}$. In this case one can define the immediate rewards as the expected immediate rewards $r^a_{s_i} = \mathrm{E}[R^a_{s_i \to s_j}]$.
- The initial state probabilities have the property $\sum_{s_i \in \mathcal{S}} p^{(0)}_{s_i} = 1$.
- Different authors use different conventions to define MDPs; some include only the state space, action space, transition probabilities and rewards, others include also the discount factor or the initial state probabilities.

MDP Example 1

Terminal or Absorbing States

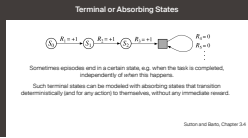


Sometimes episodes end in a certain state, e.g. when the task is completed, independently of *when* this happens.

Such terminal states can be modeled with absorbing states that transition deterministically (and for any action) to themselves, without any immediate reward.

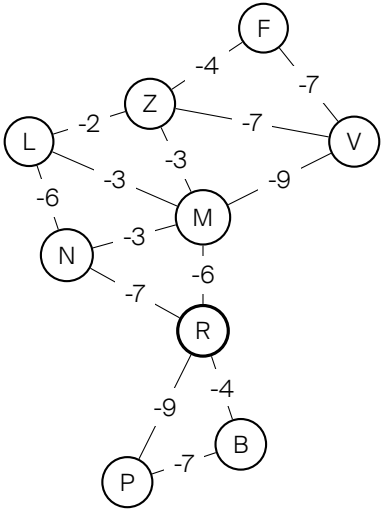
Sutton and Barto, Chapter 3.4

Notes



The planning example in the simplified map of some cities in Europe can be modeled by introducing a self-transition with zero reward for the goal city and removing all out-going transitions, e.g. for Rome. If we want to use the same map to solve another planning problem, we would re-insert the out-going transitions and remove the self-transition for Rome and apply these changes to the new goal city. This modification of the MDP results effectively in fixing the value for the goal city at 0.

MDP Example 2: Travel to Rome



- $\mathcal{S} = \{\mathsf{F}, \mathsf{Z}, \mathsf{V}, \mathsf{L}, \mathsf{M}, \mathsf{N}, \mathsf{R}, \mathsf{P}, \mathsf{B}\}$
- $\mathcal{A}_L = \{\mathsf{Z}, \mathsf{M}, \mathsf{N}\}$
- Lines indicate deterministic bi-directional connections, e.g. $p^{\mathsf{N}}_{\mathsf{L}\to\mathsf{N}} = 1$ and $p^{\mathsf{L}}_{\mathsf{N}\to\mathsf{L}} = 1$.
- The number on the lines indicate the "reward" r^a_s, i.e. the cost (distance) of traveling along this line.
- If we want to travel to R, we define R as an absorbing state.

Policies, Value Functions and Objectives

The goal is to find a **policy** $\pi^{(t)}(a|s) \in [0,1]$ (i.e. probability of taking action a in state s and time point t) that maximizes some objective. We use the notation π to denote the policy for all states and time points. We define the horizon-T **value function**

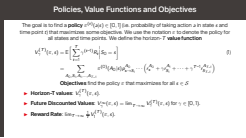
$$V_\gamma^{(T)}(\pi, s) = \mathrm{E}\left[\sum_{t=1}^{T} \gamma^{(t-1)} R_t \middle| S_0 = s\right] \tag{1}$$

$$= \sum_{A_0, S_1, A_1, \ldots, A_{T-1}} \pi^{(0)}(A_0|s) p_{s \to S_1}^{A_0} \cdots \left(r_s^{A_0} + \gamma r_{S_1}^{A_1} + \cdots + \gamma^{T-1} r_{S_{T-1}}^{A_{T-1}}\right)$$

Objectives find the policy π that maximizes for all $s \in \mathcal{S}$

- **Horizon-T values**: $V_\gamma^{(T)}(\pi, s)$.
- **Future Discounted Values**: $V_\gamma^{\infty}(\pi, s) = \lim_{T \to \infty} V_\gamma^{(T)}(\pi, s)$ for $\gamma \in [0,1)$.
- **Reward Rate**: $\lim_{T \to \infty} \frac{1}{T} V_1^{(T)}(\pi, s)$.

Notes



- In general, the policy can depend on the time point, but in some cases it is independent of time and we can drop the upper index (t).
- On the second line of the definition of the value function we write out explicitly the expectation by summing over all possible actions and states (up to horizon T), weighted by the probabilities of taking those actions $\pi^{(t)}(A_t, S_t)$ (given by the policy) and the transition probabilities $p^{A_t}_{S_t \to S_{t+1}}$.

Comments

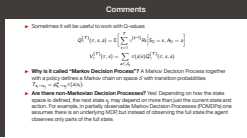
- Sometimes it will be useful to work with Q-values

$$Q_\gamma^{(T)}(\pi, s, a) = \mathrm{E}\left[\sum_{t=1}^{T} \gamma^{(t-1)} R_t \middle| S_0 = s, A_0 = a\right]$$

$$V_\gamma^{(T)}(\pi, s) = \sum_{a \in \mathcal{A}_s} \pi(a|s) Q_\gamma^{(T)}(\pi, s, a)$$

- **Why is it called "Markov Decision Process"?** A Markov Decision Process together with a policy defines a Markov chain on space $\mathcal{S}$ with transition probabilities $T_{s_i \to s_j} = p^a_{s_i \to s_j} \pi(a|s_i)$.
- **Are there non-Markovian Decision Processes?** Yes! Depending on how the state space is defined, the next state s_j may depend on more than just the current state and action. For example, in partially observable Markov Decision Processes (POMDPs) one assumes there is an underlying MDP, but instead of observing the full state the agent observes only parts of the full state.

Notes



A famous, academic example of a partially observable Markov Decision Process is the so-called Tiger Problem (`https://people.csail.mit.edu/lpk/papers/aij98-pomdp.pdf`), where a tiger is behind one door and a large reward is behind the other door. The agent can either listen, or open the left or the right door. When the agent listens, it observes a roar either behind the left or the right door, but the observation is not always accurate; with a small probability the agent may observe a roar behind the left door, even when the tiger is behind the right door and vice versa.

All non-Markovian Decision Processes could in principle be turned into Markov Decision Processes by augmenting the state space (for example with perfect knowledge about the actual position of the tiger) but in practice it may be difficult or impossible to do this augmentation.

Our world is usually partially observable: as long as the door of the fridge is closed we do not directly observe the content of the fridge; the full state of mind of another person is usually unobservable to us.

What is the Relationship with Reinforcement Learning?

MDP

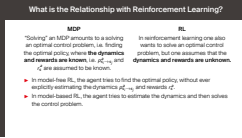
"Solving" an MDP amounts to a solving an optimal control problem, i.e. finding the optimal policy, where **the dynamics and rewards are known**, i.e. $p^a_{s_i \to s_j}$ and r^a_s are assumed to be known.

RL

In reinforcement learning one also wants to solve an optimal control problem, but one assumes that the **dynamics and rewards are unknown.**

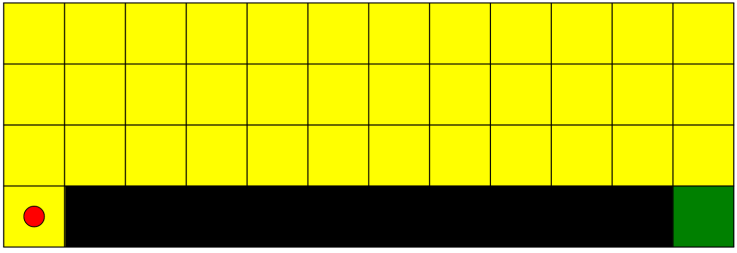
- In model-free RL, the agent tries to find the optimal policy, without ever explicitly estimating the dynamics $p^a_{s_i \to s_j}$ and rewards r^a_s.
- In model-based RL, the agent tries to estimate the dynamics and then solves the control problem.

Notes



- There is no estimation problem involved in solving an MDP, but solving the optimal control problem is still a non-trivial problem itself.
- Model-based reinforcement learning solves explicitly an estimation problem *and* an optimal control problem. Model-free reinforcement learning solves the estimation and the optimal control problem implicitly.
- You have already seen an examples of model-free reinforcement learning: Q-learning!
- You will see examples of model-based RL later in the semester.
- The exploration-exploitation trade-off exists only in reinforcement learning, but not when solving MDPs. One could say, solving MDPs is solving an exploitation problem. The exploration part in reinforcement learning is needed to tackle the estimation problem.

Exercise: The Cliff-Walking MDP



1. How many states are needed (at least) to represent this environment as an MDP?
2. What is the size of the action space for the initial state?
3. Does this MDP have any absorbing states?
4. What is the optimal policy?
5. What is the value $V_1^{\infty}(\pi^*, \texttt{initialstate})$ of the initial state for the optimal policy?

Notes

In the cliff-walking environment, the agent (red dot) starts each episode at the bottom left (where the agent is in the image); the agent can move to neighboring squares with the up, down, left, right actions (unless there is a wall). An episode ends, when falling into the cliff (black area) or when landing in the green state at the bottom right. Each normal step costs $r = -1$, falling into the cliff costs $r = -100$. There are 12 yellow states in the top row.

Table of Contents

The Optimal Fixed Horizon Policy

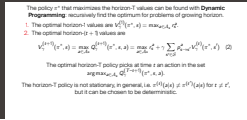
The policy π^* that maximizes the horizon-T values can be found with **Dynamic Programming**: recursively find the optimum for problems of growing horizon.

1. The optimal horizon-1 values are $V_\gamma^{(1)}(\pi^*, s) = \max_{a \in \mathcal{A}_s} r_s^a$.
2. The optimal horizon-$(t+1)$ values are

$$V_\gamma^{(t+1)}(\pi^*, s) = \max_{a \in \mathcal{A}_s} Q_\gamma^{(t+1)}(\pi^*, s, a) = \max_{a \in \mathcal{A}_s} r_s^a + \gamma \sum_{s' \in \mathcal{S}} p_{s \to s'}^a V_\gamma^{(t)}(\pi^*, s') \quad (2)$$

The optimal horizon-T policy picks at time t an action in the set $\arg\max_{a \in \mathcal{A}_s} Q_\gamma^{(T-t+1)}(\pi^*, s, a)$.

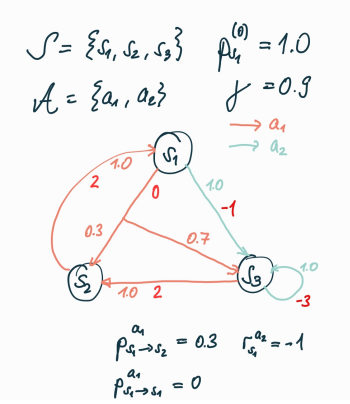
The horizon-T policy is not stationary, in general, i.e. $\pi^{(t)}(a|s) \neq \pi^{(t')}(a|s)$ for $t \neq t'$, but it can be chosen to be deterministic.



Notes

- The **dynamic programming** method breaks decision problems into smaller subproblems. **Bellman's principle of optimality** describes how to do this: An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. (See Bellman, 1957, Chap. III.3.)
- The $\arg\max_{a \in \mathcal{A}_s} Q_\gamma^{(T-t+1)}(\pi^*, s, a)$ may contain multiple actions. In this case one can randomly break ties and select any action that maximizes the Q-values.
- For a horizon-T problem, one finds first the (set of) optimal action(s) for the last time step (the solution for the horizon-1 problem), for example $\pi^{(T)}(a|s) = 1$ if $a = \text{first}(\arg\max_{a \in \mathcal{A}_s} r_s^a)$ and $\pi^{(T)}(a'|s) = 0$ for all $a' \neq a$.
- Then one finds the (set of) optimal action(s) for the second to last time step (the solution for the horizon-2 problem), e.g. $\pi^{(T-1)}(a|s) = 1$ if $a = \text{first}(\arg\max_{a \in \mathcal{A}_s} Q^{(2)}(\pi^*, s, a))$ and $\pi^{(T-1)}(a'|s) = 0$ for all $a' \neq a$, etc.
- In the exercises you will construct an example to show that the horizon-T policy can be non-stationary.

The Optimal 3-step Policy for the MDP in Example 1



Fixed-Point Iterations and Banach's Fixed Point Theorem

Some equations of the form $x = T(x)$ can be solved with a fixed point iteration:

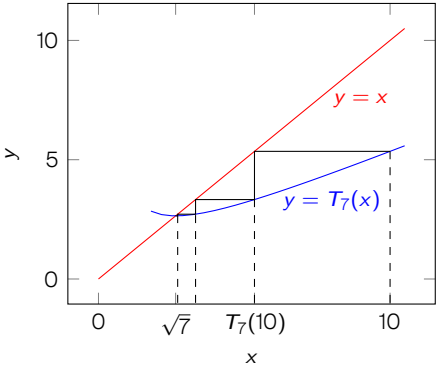
Start with $x^{(0)}$ and compute

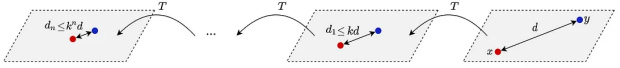
$$x^{(k)} = T(x^{(k-1)})$$

until $x^{(k)} \approx x^{(k-1)}$.

Example: Heron's method for computing the square root of a

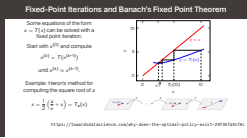
$$x = \frac{1}{2}\left(\frac{a}{x} + x\right) = T_a(x)$$





https://towardsdatascience.com/why-does-the-optimal-policy-exist-29f30fd51f8c

Notes



For the interested students, here is the mathematical background
(see also Appendix A of `http://www.ualberta.ca/~szepesva/papers/RLAlgsInMDPs.pdf`):

Let (X, d) be a complete metric space. Then a map $T : X \to X$ is called a **contraction mapping** on X if there exists $q \in [0, 1)$ such that $d(T(x), T(y)) \leq q d(x, y)$ for all $x, y \in X$.

Banach Fixed Point Theorem. Let (X, d) be a non-empty complete metric space with a contraction mapping $T : X \to X$. Then T admits a unique fixed-point x^* in X (i.e. $T(x^*) = x^*$). Furthermore, x^* can be found as follows: start with an arbitrary element $x_0 \in X$ and define a sequence $(x_n)_{n \in \mathbb{N}}$ by $x_n = T(x_{n-1})$ for $n \geq 1$. Then $\lim_{n \to \infty} x_n = x^*$.

For Heron's method for computing the square root, we can take the distance $d(x, y) = |x - y|$ and the contraction mapping $T : [\sqrt{a/2}, \infty) \to [\sqrt{a/2}, \infty), x \to T_a(x)$ for $a > 0$. It is easy to show that $\left|\frac{df_a}{dx}(x)\right| \leq \frac{1}{2}$ on $[\sqrt{a/2}, \infty)$ and therefore (by the mean value theorem) $|f_a(x) - f_a(y)| \leq \frac{1}{2}|x - y|$.

Maximizing Future Discounted Values with Dynamic Programming

Let us define the mapping (sometimes called **Bellman operator**)

$$T_\gamma : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|},\ T_\gamma(X)_s = \max_{a \in \mathcal{A}_s} \left(r_s^a + \gamma \sum_{s' \in \mathcal{S}} p^a_{s \to s'} X_{s'} \right). \tag{3}$$

- One can show that the mapping T_γ is a contraction mapping and Banach's fixed point theorem can be applied. Hence, there is a unique fixed point $X^* = T_\gamma(X^*)$.
- Note that $X^* = T_\gamma(X^*)$ is exactly the same equation we want the optimal horizon-∞ values to satisfy (c.f. Eq. 2).
- Therefore, this fixed point is the solution $V_\gamma^\infty(\pi^*, s) = X_s^*$.
- The optimal policy is to choose actions in $\arg\max_{a \in \mathcal{A}_s} Q_\gamma^\infty(\pi^*, s, a)$.
- This policy is stationary, i.e. $\pi^{(t)}(a|s) = \pi^{(t')}(a|s)$ for $t \neq t'$, and it can be chosen to be deterministic!

Let us define the mapping (sometimes called **Bellman operator**)

$$\mathcal{T}_\gamma : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|},\ \mathcal{T}_\gamma(X)_s = \max_{\mathcal{A}_s \neq \emptyset} \left(\mathcal{C}_s^{\mathcal{A}_s} + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s \to s'}^{\mathcal{A}_s} X_{s'} \right) \quad (3)$$

- One can show that the mapping $\mathcal{T}_\gamma$ is a contraction mapping and Banach's fixed point theorem can be applied. Hence, there is a unique fixed point $X^* = \mathcal{T}_\gamma(X^*)$.
- Note that $X^* = \mathcal{T}_\gamma(X^*)$ is exactly the same equation we want the optimal horizon-∞ values to satisfy (c.f. Eq. 2).
- Therefore, this fixed point is the solution $V_s^{\infty}(s^0, s) = X_s^*$.
- The optimal policy is to choose actions in $\arg\max_{\mathcal{A}_s \neq \emptyset} \mathcal{A}_s^T \mathcal{C}_s^{\mathcal{A}_s}(s^0, s, s)$.
- This policy is stationary, i.e. $\pi^{(k)}(s|s) = \pi^{(k')}(s|s)$ for $t \neq t'$, and it can be chosen to be deterministic!

Notes

X is a vector in $\mathbb{R}^{|\mathcal{S}|}$. The Bellman operator thus maps vectors to vectors.

Value Iteration

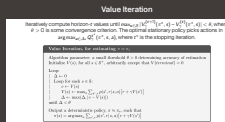
Iteratively compute horizon-t values until $\max_{s \in \mathcal{S}} |V_\gamma^{(t+1)}(\pi^*, s) - V_\gamma^{(t)}(\pi^*, s)| < \theta$, where $\theta > 0$ is some convergence criterion. The optimal stationary policy picks actions in $\arg\max_{a \in \mathcal{A}_s} Q_\gamma^{t^*}(\pi^*, s, a)$, where t^* is the stopping 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 $s \in \mathcal{S}$:
| $\quad v \leftarrow V(s)$
| $\quad V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
| $\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

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



Notes

Value iteration is a naive application of the convergence property of the Bellman operator: just apply it until some stopping criterion is reached.

It is not unreasonable to start the fixed-point iteration with $X_s = \max_{a \in \mathcal{A}_s} r_s^a$ (as for the horizon-T solution), but value iteration would also converge, if one initialized X randomly (thanks to the contraction mapping and Banach's fixed point theorem).

Having to define some stopping criterion makes value iteration a bit unattractive.

Policy Improvement Theorem

For deterministic policies $\pi(a|s) = 1$ for some a and $\pi(a'|s) = 0, \forall a' \neq a$, we will use here the notation $a = \pi(s)$.

Policy Improvement Theorem

Let π and π' be a pair of deterministic policies such that $\forall s$

$$Q_\gamma(\pi, s, \pi'(s)) \geq V_\gamma(\pi, s)\,.$$

Then the policy π' must be as good as or better than π, i.e.

$$V_\gamma(\pi', s) \geq V_\gamma(\pi, s)\,.$$

Policy Iteration

Policy Evaluation

$$V_\gamma^\infty(\pi, s) = r_s^{\pi(s)} + \gamma \sum_{s' \in \mathcal{S}} p_{s \to s'}^{\pi(s)} V_\gamma^\infty(\pi, s')$$

$$V_\gamma^\infty(\pi) = (I - \gamma P)^{-1} r$$

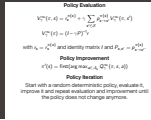
with $r_s = r_s^{\pi(s)}$ and identity matrix I and $P_{s,s'} = p_{s \to s'}^{\pi(s)}$.

Policy Improvement

$$\pi'(s) = \text{first}(\arg\max_{a \in \mathcal{A}_s} Q_\gamma^\infty(\pi, s, a))$$

Policy Iteration

Start with a random deterministic policy, evaluate it, improve it and repeat evaluation and improvement until the policy does not change anymore.



Notes

- $V_\gamma^\infty(\pi)$ and r are vectors in $\mathbb{R}^{|\mathcal{S}|}$, I is the $|\mathcal{S}|$-dimensional identity matrix.
- The policy evaluation can be done by explicitly inverting the matrix $(I - \gamma P)$ or by approximating the Neumann series $(I - \gamma P)^{-1} = \sum_{k=0}^{\infty} \gamma^k P^k$ as in the example of iterative policy evaluation in the pseudocode on the right.

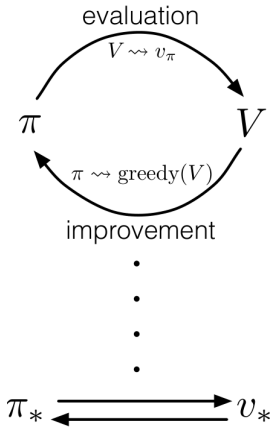
Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

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

Generalized Policy Iteration



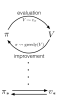
Policy iteration consists of two simultaneous, interacting processes, one making the value function consistent with the current policy (policy evaluation), and the other making the policy greedy with respect to the current value function (policy improvement). In policy iteration, these two processes alternate, each completing before the other begins, but this is not really necessary. In value iteration, for example, only a single iteration of policy evaluation is performed in between each policy improvement. In asynchronous dynamic programming methods, the evaluation and improvement processes are interleaved at an even finer grain. In some cases a single state is updated in one process before returning to the other. As long as both processes continue to update all states, the ultimate result is typically the same—convergence to the optimal value function and an optimal policy.

Sutton and Barto, Chapter 4.6

Notes

Standard Q-learning can be seen as an example of asynchronous dynamic programming, where in every step the policy is evaluated only for the current state action pair and the policy is only improved (if necessary) for the current state.

Generalized Policy Iteration



Policy iteration consists of two simultaneous, interacting processes, one making the value function consistent with the current policy (policy evaluation), and the other making the policy greedy with respect to the current value function (policy improvement). In policy iteration, these two processes alternate, each completing before the other begins, but this is not really necessary. In value iteration, for example, only a single iteration of policy evaluation is performed in between each policy improvement. In asynchronous dynamic programming methods, the evaluation and improvement processes are interleaved at an even finer grain. In some cases a single state is updated in one process before returning to the other. As long as both processes continue to update all states, the ultimate result is typically the same—convergence to the optimal value function and an optimal policy.

Sutton and Barto, Chapter 4.6

Summary

Table of Contents

Maximizing Future Discounted Values with Linear Programming

As an alternative to dynamic programming, one can define the problem of finding optimal values as a linear program.

Let us use the notation $v_s = V_\gamma^\infty(\pi^*, s)$, for the optimal policy π^*. We have

$$v_s = \max_a \left[r_s^a + \gamma \sum_{s' \in \mathcal{S}} p_{s \to s'}^a v_{s'} \right] \geq r_s^a + \gamma \sum_{s' \in \mathcal{S}} p_{s \to s'}^a v_{s'}, \forall a \in \mathcal{A}_s, s \in \mathcal{S}\,. \tag{4}$$

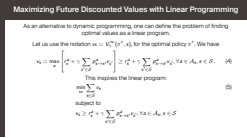
This inspires the linear program:

$$\min_{v_s} \sum_{s \in \mathcal{S}} v_s \tag{5}$$

subject to

$$v_s \geq r_s^a + \gamma \sum_{s' \in \mathcal{S}} p_{s \to s'}^a v_{s'}, \forall a \in \mathcal{A}_s, s \in \mathcal{S}$$

Notes



Linear Programming is a technique for the optimization of a linear objective function, subject to linear equality and linear inequality constraints. There exist efficient solvers for linear programming problems.

We will not discuss linear programming solutions of MDPs in details, but you should know that MDPs can be solved with linear programming and you should get an idea how MDP problems can be described as a linear programming problems.

All techniques to solve MDPs can give us inspirations for solving reinforcement learning problems (see for example `http://proceedings.mlr.press/v130/bas-serrano21a.html`).

Example

Maximizing the Reward Rate with Linear Programming

Let us define the Markov chain with transition probabilities $p^{\pi}_{s\to s'} = \sum_{a\in\mathcal{A}_s} \pi(a|s) p^a_{s\to s'}$ and let us assume that this Markov chain is irreducible and aperiodic for all π. Then there exists the stationary distribution ρ^{π} Note that this is a strong assumption, which, however, is e.g. fulfilled when $p^a_{s\to s'}$ is positive for all a, s, s'. Let us define the reward rate

$$\bar{r} = \sum_{s\in\mathcal{S}, a\in\mathcal{A}_s} \pi^{\pi}_s \pi(a|s) r^a_s$$

and introduce the variables $c^a_s = \rho^{\pi}_s \pi(a|s)$. Then we can define the linear program

$$\max_{c^a_s} \sum_{s\in\mathcal{S}, a\in\mathcal{A}_s} r^a_s c^a_s \tag{6}$$

subject to

$$\sum_{s\in\mathcal{S}, a\in\mathcal{A}_s} c^a_s = 1 \text{ and } \sum_{a\in\mathcal{A}_{s'}} c^a_{s'} = \sum_{s\in\mathcal{S}, a\in\mathcal{A}_s} c^a_s p^a_{s\to s'} \tag{7}$$