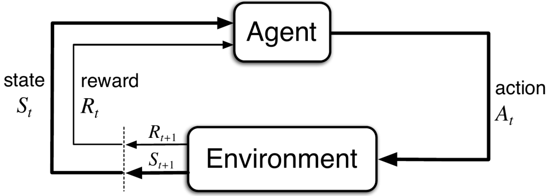
Reinforcement learning

Paulo Rauber

paulo@idsia.ch

November 14, 2019

Reinforcement learning



[1]

[1] Image from [Sutton and Barto, 2018]

Reinforcement learning

- An agent interacts with an environment during a sequence of discrete time steps $t = 0, 1, 2, \ldots$
- At each time step t, the agent receives some representation of the state $s_t \in \mathcal{S}$
- The agent then selects an action $a_t \in \mathcal{A}(s_t)$
- One time step later, the agent receives a reward $r_{t+1} \in \mathbb{R}$ and a new state $s_{t+1} \in \mathcal{S}$
- A policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ is a function such that $\pi(s, a)$ represents the probability that $a_t = a$ given that $s_t = s$

Discounted return

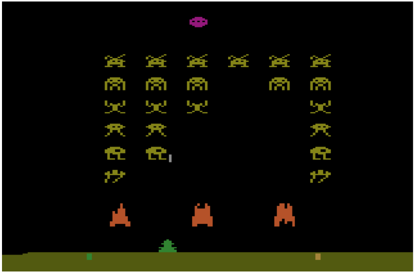
- The discounted return u_t after time step t is given by

$$u_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+1+k},$$

  where $0 \leq \gamma < 1$ is the discount factor
- A reward received k time steps into the future is only worth γ^{k-1} times what it would be worth if it were received on the next step
- If necessary, a state can transition only to itself and yield no rewards
- The objective of the agent is to maximize the discounted return

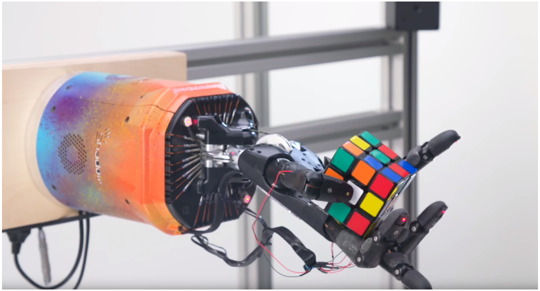
Applications: games

- Atari [Mnih et al., 2015], Dota 2 [Brockman et al., 2019a], chess and Go [Silver et al., 2018]



Applications: robotics

- Rubik's cube manipulation [Brockman et al., 2019b]

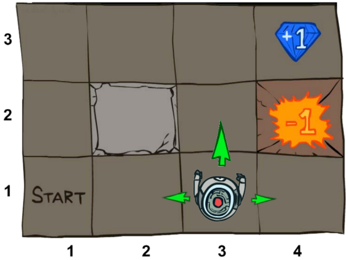


Applications: others

- Practical:
  - Logistics
  - Finance
  - Marketing
- Theoretical:
  - Every task with a computable description can be formulated as a reinforcement learning problem [Hutter, 2004].

Example: grid world

- States $\mathcal{S} = \{1, 2, \ldots, 12\}$, actions $\mathcal{A} = \{1, 2, 3, 4\}$
- Reward 1 on action at goal, reward -1 on action at trap, and reward 0 on action at other states
- Actions at goal and trap transition to an absorbing state
- Discount factor $\gamma = 0.9$



[2]

[2] Image from [Klein et al., 2019]

Markov decision process

- A state that summarizes the entire past with all that is relevant for decision making has the Markov property
- In a Markov decision process, for any sequence of states, action and rewards $s_t, a_t, r_t, \dots r_1, s_0, a_0$ (history) and all $s' \in \mathcal{S}, r' \in \mathbb{R}$,

$$P(S_{t+1} = s', R_{t+1} = r' \mid s_t, a_t, r_t, \dots, r_1, s_0, a_0) = P(S_{t+1} = s', R_{t+1} = r' \mid s_t, a_t)$$

- The conditional joint probability distribution over states and rewards on the right side defines the one-step dynamics of the problem

$\mathcal{P}^a_{ss'}$

- The probability $\mathcal{P}^a_{ss'}$ of transitioning from state s to state s' given action a is given by

$$\begin{aligned}\mathcal{P}^a_{ss'} &= P(S_{t+1} = s' \mid S_t = s, A_t = a) \\ &= \sum_{r'} P(S_{t+1} = s', R_{t+1} = r' \mid S_t = s, A_t = a).\end{aligned}$$

- Note that $\mathcal{P}^a_{ss'}$ is independent of the current time step

$\mathcal{R}^a_{ss'}$

- The expected reward on transitioning from state s to state s' given the action a is given by

$$\begin{aligned}\mathcal{R}^a_{ss'} &= \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a, S_{t+1} = s'] \\ &= \frac{1}{\mathcal{P}^a_{ss'}} \sum_{r'} r' P(S_{t+1} = s', R_{t+1} = r' \mid A_t = a, S_t = s)\end{aligned}$$

- Note that $\mathcal{R}^a_{ss'}$ is independent of the current time step

Value function V^{π}

- The value $V^{\pi}(s)$ of a state $s \in \mathcal{S}$ is the expected (discounted) return of starting in state s and following the policy π

$$V^{\pi}(s) = \mathbb{E}_{\pi}[U_t \mid S_t = s] = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right] \quad (1)$$

Action value function Q^π

- The value $Q^\pi(s, a)$ of taking an action $a \in \mathcal{A}$ when in state $s \in \mathcal{S}$ and afterwards following the policy π is given by

$$
\begin{aligned}
Q^\pi(s, a) &= \mathbb{E}_\pi[U_t \mid S_t = s, A_t = a] \\
&= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]
\end{aligned}
$$

Recursivity of the value function

Theorem (Recursivity of the value function)

For any policy π and state $s \in \mathcal{S}$,

$$V^{\pi}(s) = \sum_{a} \pi(s, a) \sum_{s'} \mathcal{P}^{a}_{ss'} [\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s')]. \tag{2}$$

Notation

- We denote random variables by upper case letters and assignments to these variables by corresponding lower case letters
- We omit the subscript that typically relates a probability function to random variables when there is no risk of ambiguity
- For example, let X and Y be discrete random variables. In the same context, we will let $p(x|y)$ denote $P(X = x|Y = y)$ and $p(y|x)$ denote $P(Y = y|X = x)$
- This notation where the arguments select between different probability functions is standard in machine learning

Recursivity of the value function

Proof.

Note that Equation 1 can be rewritten as

$$V^\pi(s) = \lim_{T \to \infty} \sum_{r_{t+1:T+t+1}} p(r_{t+1:T+t+1} \mid S_t = s) \sum_{k=0}^{T} \gamma^k r_{t+k+1},$$

where the dependency on the policy π becomes implicit. By marginalization,

$$V^\pi(s) = \lim_{T \to \infty} \sum_{r_{t+1:T+t+1}} \left[\sum_{a_t} \sum_{s_{t+1}} p(r_{t+1:T+t+1}, a_t, s_{t+1} \mid S_t = s) \right] \sum_{k=0}^{T} \gamma^k r_{t+k+1}.$$

Recursivity of the value function

Proof. (cont.)

By the chain rule of probability,

$$V^{\pi}(s) = \lim_{T \to \infty} \sum_{r_{t+1:T+t+1}} \sum_{a_t} \sum_{s_{t+1}} p(a_t \mid S_t = s) p(s_{t+1} \mid S_t = s, a_t) p(r_{t+1:T+t+1} \mid S_t = s, a_t, s_{t+1}) \sum_{k=0}^{T} \gamma^k r_{t+k+1}.$$

By the distributive property and reordering the three outermost summations,

$$V^{\pi}(s) = \sum_{a_t} p(a_t \mid S_t = s) \sum_{s_{t+1}} p(s_{t+1} \mid S_t = s, a_t) \lim_{T \to \infty} \sum_{r_{t+1:T+t+1}} p(r_{t+1:T+t+1} \mid S_t = s, a_t, s_{t+1}) \sum_{k=0}^{T} \gamma^k r_{t+k+1}. \tag{3}$$

Recursivity of the value function

Proof. (cont.)

Let E denote the limit in the previous equation, such that

$$E = \lim_{T \to \infty} \sum_{r_{t+1:T+t+1}} p(r_{t+1:T+t+1} \mid S_t = s, a_t, s_{t+1}) \sum_{k=0}^{T} \gamma^k r_{t+k+1}.$$

By isolating the first term in the innermost summation,

$$E = \lim_{T \to \infty} \sum_{r_{t+1:T+t+1}} p(r_{t+1:T+t+1} \mid S_t = s, a_t, s_{t+1}) \left[r_{t+1} + \sum_{k=1}^{T} \gamma^k r_{t+k+1} \right].$$

Recursivity of the value function

Proof. (cont.)

By the linearity of expectation, $E = E_1 + E_2$, where

$$\begin{aligned} E_1 &= \lim_{T \to \infty} \sum_{r_{t+1:T+t+1}} p(r_{t+1:T+t+1} \mid S_t = s, a_t, s_{t+1}) r_{t+1} \\ &= \mathbb{E}_\pi \left[R_{t+1} \mid S_t = s, a_t, s_{t+1} \right] = \mathcal{R}^{a_t}_{ss_{t+1}}, \end{aligned}$$

and

$$E_2 = \lim_{T \to \infty} \sum_{r_{t+1:T+t+1}} p(r_{t+1:T+t+1} \mid S_t = s, a_t, s_{t+1}) \sum_{k=1}^{T} \gamma^k r_{t+k+1}.$$

Recursivity of the value function

Proof. (cont.)

By changing the indices in the innermost summation,

$$E_2 = \lim_{T \to \infty} \sum_{r_{t+1:T+t+1}} p(r_{t+1:T+t+1} \mid S_t = s, a_t, s_{t+1}) \sum_{k=0}^{T-1} \gamma^{k+1} r_{t+k+2}.$$

By moving a constant factor of γ outside of the innermost summation,

$$E_2 = \gamma \lim_{T \to \infty} \sum_{r_{t+1:T+t+1}} p(r_{t+1:T+t+1} \mid S_t = s, a_t, s_{t+1}) \sum_{k=0}^{T-1} \gamma^{k} r_{t+k+2}.$$

Recursivity of the value function

Proof. (cont.)

Because $R_{t+2:T+t+1} \perp\!\!\!\perp S_t, A_t \mid S_{t+1}$ due to the Markov property,

$$E_2 = \gamma \lim_{T \to \infty} \mathbb{E}_\pi \left[\sum_{k=0}^{T-1} \gamma^k R_{t+k+2} \mid s_{t+1} \right] = \gamma V^\pi(s_{t+1}).$$

Returning to Equation 3,

$$V^\pi(s) = \sum_{a_t} p(a_t \mid S_t = s) \sum_{s_{t+1}} p(s_{t+1} \mid S_t = s, a_t) \left[\mathcal{R}^{a_t}_{ss_{t+1}} + \gamma V^\pi(s_{t+1}) \right].$$

By making the dependency on π explicit and renaming variables,

$$V^\pi(s) = \sum_a \pi(s, a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^\pi(s') \right],$$

$\square$

Recursivity of the action value function

Theorem (Recursivity of the action value function)

For any policy π, state $s \in \mathcal{S}$, and action $a \in \mathcal{S}$,

$$Q^{\pi}(s, a) = \sum_{s'} \mathcal{P}^{a}_{ss'}[\mathcal{R}^{a}_{ss'} + \gamma \sum_{a'} \pi(s', a') Q^{\pi}(s', a')].$$

Relationship between V^π and Q^π

Theorem (Relationship between V^π and Q^π)

For any policy π, state $s \in \mathcal{S}$, and action $a \in \mathcal{S}$,

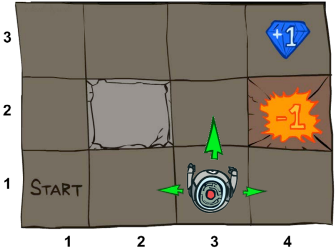
$$V^\pi(s) = \sum_a \pi(s, a) Q^\pi(s, a),$$

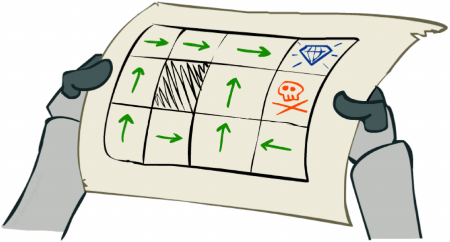
and

$$Q^\pi(s, a) = \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V(s')].$$

Optimal policies

- Let $\pi \geq \pi'$ if and only if $V^{\pi}(s) \geq V^{\pi'}(s)$ for all $s \in \mathcal{S}$.
- A policy π^* is optimal if $\pi^* \geq \pi$ for any policy π
- An optimal policy always exists, but is not necessarily unique




[3]

[3] Image from [Klein et al., 2019]

Optimal value functions

Theorem (Bellman optimality equations)

For any action $a \in \mathcal{A}$ and state $s \in \mathcal{S}$, the optimal state value function V^* and the optimal action value function Q^* are given by

$$V^*(s) \triangleq \max_{\pi} V^{\pi}(s) = \max_{a} \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V^*(s')],$$

and

$$Q^*(s, a) \triangleq \max_{\pi} Q^{\pi}(s, a) = \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q^*(s', a')].$$

Reinforcement learning algorithms

- Reinforcement learning algorithms aim to find an optimal policy π^* for a given environment
- For any state $s \in \mathcal{S}$, an optimal policy π^* can be found given either V^* or Q^*
- In the case of Q^*, for any $s \in \mathcal{S}$, it suffices to choose an a such that $Q^*(s, a)$ is maximal
- In the case of V^*, for any $s \in \mathcal{S}$, it suffices to choose one of the actions a that maximizes the right hand side of the Bellman optimality equation

Dynamic programming

- Dynamic programming algorithms can be used to compute optimal policies given a perfect model of the environment (one-step dynamics) when the sets of states and actions are finite
- The problem of finding the optimal value functions has optimal substructure: it can be solved by breaking it into sub-problems and then recursively finding the solutions to the sub-problems

Policy evaluation

- Policy evaluation is an iterative algorithm to compute the state value function V^π for an arbitrary policy π
- It relies on creating a sequence $V_0, V_1, \ldots$ of estimates of V^π given by

$$V_{k+1}(s) = \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V_k(s')],$$

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

- The initial value estimate V_0 can be arbitrary
- The sequence $V_0(s), V_1(s), \ldots$ converges to $V^\pi(s)$ for all $s \in \mathcal{S}$

In-place policy evaluation

- Instead of computing the new estimate V_{k+1} using the old estimate V_k, it is also possible to change a single estimate V in-place using

$$V(s) \leftarrow \sum_a \pi(s, a) \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V(s')],$$

  for all $s \in \mathcal{S}$
- The estimate $V(s)$ also converges to $V^\pi(s)$ for all $s \in \mathcal{S}$ after repeated passes over all states

In-place policy evaluation

Algorithm 1 Iterative policy evaluation (in-place)

Input: policy π, one-step dynamics functions $\mathcal{P}$ and $\mathcal{R}$, discount factor γ, tolerance θ.

Output: Value function $V = V^\pi$ when $\theta \to 0$.

```
for each s ∈ S do
    V(s) ← 0
end for
repeat
    Δ ← 0
    for each s ∈ S do
        v ← V(s)
        V(s) ← Σ_a π(s, a) Σ_{s'} P^a_{ss'} [R^a_{ss'} + γ V(s')]
        Δ ← max(Δ, |v − V(s)|)
    end for
until Δ < θ
```

for each $s \in \mathcal{S}$ **do**

    $V(s) \leftarrow 0$

end for

repeat

    $\Delta \leftarrow 0$

    for each $s \in \mathcal{S}$ **do**

        $v \leftarrow V(s)$

        $V(s) \leftarrow \sum_a \pi(s, a) \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V(s')]$

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

    end for

until $\Delta < \theta$

Convergence of iterative policy evaluation

Definition (Norm)

Consider a vector space Z over a field F. A function $||\cdot|| : Z \to [0, \infty)$ is a norm if

$$||\mathbf{u} + \mathbf{v}|| \leq ||\mathbf{u}|| + ||\mathbf{v}||,$$
$$||a\mathbf{v}|| = |a|||\mathbf{v}||,$$
$$||\mathbf{v}|| = 0 \implies \mathbf{v} = \mathbf{0},$$

for all $\mathbf{u}$, $\mathbf{v} \in Z$ and $a \in F$.

Convergence of iterative policy evaluation

Definition (Euclidean norm)

The Euclidean norm $||\cdot||_2 : \mathbb{R}^d \to [0, \infty)$ is given by

$$||\mathbf{v}||_2 = \sqrt{\sum_i v_i^2}.$$

Definition (Maximum norm)

The maximum norm $||\cdot||_\infty : \mathbb{R}^d \to [0, \infty)$ is given by

$$||\mathbf{v}||_\infty = \max_i |v_i|.$$

Convergence of iterative policy evaluation

Definition (Convergence of a sequence)

A sequence $(\mathbf{v}_n)_{n \geq 0} = \mathbf{v}_0, \mathbf{v}_1, \dots$ is said to converge to a vector $\mathbf{v}$ in the norm $|| \cdot ||$, denoted $\mathbf{v}_n \underset{||\cdot||}{\rightarrow} \mathbf{v}$, if

$$\lim_{n \to \infty} ||\mathbf{v}_n - \mathbf{v}|| = 0.$$

Convergence of iterative policy evaluation

Definition (Bellman operator)

Consider a reinforcement learning task with states $\mathcal{S} = \{1, 2, \dots, |\mathcal{S}|\}$, actions $\mathcal{A} = \{1, 2, \dots, |\mathcal{A}|\}$, and discount factor $\gamma < 1$. For any vector $\mathbf{v} \in \mathbb{R}^{|\mathcal{S}|}$ and state $s \in \mathcal{S}$, the Bellman operator $T^\pi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ for the policy π is given by

$$T^\pi(\mathbf{v})_s = \sum_a \pi(s, a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma v_{s'}\right].$$

Convergence of iterative policy evaluation

Theorem (Convergence of iterative policy evaluation)

Consider the Bellman operator $T^\pi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ *for the policy* π. *Given an arbitrary* $\mathbf{v}_0 \in \mathbb{R}^{|\mathcal{S}|}$, *consider also the sequence* $(\mathbf{v}_n)_{n \geq 0}$ *where* $\mathbf{v}_{k+1} = T^\pi(\mathbf{v}_k)$. *Finally, consider the vector* $\mathbf{v}^\pi \in \mathbb{R}^{|\mathcal{S}|}$ *such that* $v^\pi_s = V^\pi(s)$, *for any* $s \in \mathcal{S}$. *For any* $n \geq 0$,

$$\mathbf{v}_n \underset{||\cdot||_\infty}{\rightarrow} \mathbf{v}^\pi,$$

$$\mathbf{v}^\pi = T^\pi(\mathbf{v}^\pi),$$

$$||\mathbf{v}_n - \mathbf{v}^\pi||_\infty \leq \gamma^n ||\mathbf{v}_0 - \mathbf{v}^\pi||_\infty.$$

Convergence of iterative policy evaluation

Definition (Cauchy sequence)

Consider a normed vector space $(Z, ||\cdot||)$. A sequence $(\mathbf{v}_n)_{n\geq 0} = \mathbf{v}_0, \mathbf{v}_1, \dots$ of vectors in this space is Cauchy if

$$\lim_{n\to\infty} \sup_{m\geq n} ||\mathbf{v}_n - \mathbf{v}_m|| = 0.$$

In other words, if a sequence $(\mathbf{v}_n)_{n\geq 0}$ is Cauchy, then for every $\epsilon > 0$ there is an N such that for every $n \geq N$, we have $\sup_{m\geq n} ||\mathbf{v}_n - \mathbf{v}_m|| < \epsilon$.

Convergence of iterative policy evaluation

Definition (Banach space)

A Banach space is a normed vector space $(Z, ||\cdot||)$ where if $(\mathbf{v}_n)_{n \geq 0}$ is a Cauchy sequence then $\mathbf{v}_n \underset{||\cdot||}{\to} \mathbf{v}$ for some vector $\mathbf{v}$.

For any d, both $(\mathbb{R}^d, ||\cdot||_2)$ and $(\mathbb{R}^d, ||\cdot||_\infty)$ are Banach spaces, although we omit the corresponding proofs.

Convergence of iterative policy evaluation

Definition (L-contraction)

Consider a normed vector space $(Z, ||\cdot||)$ and a function $T : Z \to Z$. The function T is L-Lipschitz if

$$||T(\mathbf{u}) - T(\mathbf{v})|| \leq L||\mathbf{u} - \mathbf{v}||,$$

for all $\mathbf{u}, \mathbf{v} \in Z$. If $L < 1$, then T is also an L-contraction.

Convergence of iterative policy evaluation

Lemma

Consider a normed vector space $(Z, ||\cdot||)$, an L-Lipschitz function $T : Z \to Z$, and a sequence $(\mathbf{v}_n)_{n\geq 0} = \mathbf{v}_0, \mathbf{v}_1, \dots$ of vectors in this space. If $\mathbf{v}_n \underset{||\cdot||}{\to} \mathbf{v}$, then $T(\mathbf{v}_n) \underset{||\cdot||}{\to} T(\mathbf{v})$.

Convergence of iterative policy evaluation

Proof.

For any $n \geq 0$, by the definition of an L-Lipschitz function,

$$0 \leq ||T(\mathbf{v}_n) - T(\mathbf{v})|| \leq L||\mathbf{v}_n - \mathbf{v}||.$$

Since $\mathbf{v}_n \underset{||\cdot||}{\rightarrow} \mathbf{v}$,

$$\lim_{n \to \infty} L||\mathbf{v}_n - \mathbf{v}|| = L \lim_{n \to \infty} ||\mathbf{v}_n - \mathbf{v}|| = 0.$$

By the squeeze theorem,

$$\lim_{n \to \infty} ||T(\mathbf{v}_n) - T(\mathbf{v})|| = 0.$$

□

Convergence of iterative policy evaluation

Theorem (Banach's fixed point theorem)

If $(Z, ||\cdot||)$ is a Banach space and $T : Z \to Z$ is an L-contraction, then T has a unique fixed point $\mathbf{v}$. Furthermore, for any $\mathbf{v}_0 \in Z$, let $\mathbf{v}_{n+1} = T(\mathbf{v}_n)$. For any $n \geq 0$,

$$\mathbf{v}_n \underset{||\cdot||}{\rightarrow} \mathbf{v},$$

$$||\mathbf{v}_n - \mathbf{v}|| \leq L^n ||\mathbf{v}_0 - \mathbf{v}||.$$

Convergence of iterative policy evaluation

Proof.

We first show that the sequence $(\mathbf{v}_n)_{n \geq 0}$ is Cauchy, which guarantees that $\mathbf{v}_n \underset{||\cdot||}{\rightarrow} \mathbf{v}$ for some vector $\mathbf{v}$.

As a first step, we show that $||\mathbf{v}_{n+k} - \mathbf{v}_n|| \leq L^n ||\mathbf{v}_k - \mathbf{v}_0||$ for any $n, k \geq 0$. The case $n = 0$ is trivial. Suppose that the inductive hypothesis is true for some n, and consider the case $n + 1$:

$$\begin{aligned}
||\mathbf{v}_{n+k+1} - \mathbf{v}_{n+1}|| &= ||T(\mathbf{v}_{n+k}) - T(\mathbf{v}_n)|| && \text{(definition of the sequence)} \\
&\leq L||\mathbf{v}_{n+k} - \mathbf{v}_n|| && \text{(definition of } L\text{-contraction)} \\
&\leq L^{n+1}||\mathbf{v}_k - \mathbf{v}_0||, && \text{(inductive hypothesis)}
\end{aligned}$$

as we wanted to show.

Convergence of iterative policy evaluation

Proof. (cont.)

For $k \geq 1$, $||\mathbf{v}_k - \mathbf{v}_0|| = ||\mathbf{v}_k + (-\mathbf{v}_{k-1} + \mathbf{v}_{k-1}) + \ldots + (-v_1 + v_1) - \mathbf{v}_0||$. Therefore,

$$
\begin{aligned}
||\mathbf{v}_k - \mathbf{v}_0|| &= \left\| \sum_{i=1}^{k} \mathbf{v}_i - \mathbf{v}_{i-1} \right\| && \text{(reorganizing terms)} \\
&\leq \sum_{i=1}^{k} ||\mathbf{v}_i - \mathbf{v}_{i-1}|| && \text{(triangle inequality)} \\
&\leq \sum_{i=1}^{k} L^{i-1} ||\mathbf{v}_1 - \mathbf{v}_0|| && \text{(earlier result)} \\
&\leq \frac{||\mathbf{v}_1 - \mathbf{v}_0||}{1 - L}. && \left(\lim_{n \to \infty} \sum_{i=0}^{n} L^n = \frac{1}{1 - L} \right)
\end{aligned}
$$

Convergence of iterative policy evaluation

Proof. (cont.)

We are now close to showing that $(\mathbf{v}_n)_{n \geq 0}$ is Cauchy. For any $n, k \geq 0$, combining the previous two results,

$$0 \leq ||\mathbf{v}_{n+k} - \mathbf{v}_n|| \leq L^n ||\mathbf{v}_k - \mathbf{v}_0|| \leq L^n \frac{||\mathbf{v}_1 - \mathbf{v}_0||}{1 - L}.$$

Therefore, for any fixed $n \geq 0$,

$$0 \leq \sup_{k \geq 0} ||\mathbf{v}_{n+k} - \mathbf{v}_n|| \leq \sup_{k \geq 0} L^n \frac{||\mathbf{v}_1 - \mathbf{v}_0||}{1 - L} = L^n \frac{||\mathbf{v}_1 - \mathbf{v}_0||}{1 - L}.$$

Convergence of iterative policy evaluation

Proof. (cont.)

Because $0 \leq L < 1$,

$$\lim_{n \to \infty} L^n \frac{||\mathbf{v}_1 - \mathbf{v}_0||}{1 - L} = \frac{||\mathbf{v}_1 - \mathbf{v}_0||}{1 - L} \lim_{n \to \infty} L^n = 0.$$

Therefore, by the squeeze theorem,

$$\lim_{n \to \infty} \sup_{k \geq 0} ||\mathbf{v}_{n+k} - \mathbf{v}_n|| = 0,$$

which completes the proof that $(\mathbf{v}_n)_{n \geq 0}$ is Cauchy. Let $\mathbf{v}$ denote the vector such that $\mathbf{v}_n \underset{||\cdot||}{\to} \mathbf{v}$.

Convergence of iterative policy evaluation

Proof. (cont.)

Our next step is to show that $\mathbf{v}$ is a fixed point of T. For any n,

$$
\begin{aligned}
0 \leq ||T(\mathbf{v}) - \mathbf{v}|| &= ||T(\mathbf{v}) + (-T(\mathbf{v}_n) + T(\mathbf{v}_n)) - \mathbf{v}|| && \text{(introducing zeros)} \\
&\leq ||T(\mathbf{v}) - T(\mathbf{v}_n)|| + ||T(\mathbf{v}_n) - \mathbf{v}|| && \text{(triangle inequality)} \\
&\leq L||\mathbf{v} - \mathbf{v}_n|| + ||\mathbf{v}_{n+1} - \mathbf{v}|| && \text{(L-contraction)}
\end{aligned}
$$

Because $\mathbf{v}_n \underset{||\cdot||}{\rightarrow} \mathbf{v}$,

$$\lim_{n\to\infty} L||\mathbf{v} - \mathbf{v}_n|| + ||\mathbf{v}_{n+1} - \mathbf{v}|| = 0.$$

Convergence of iterative policy evaluation

Proof. (cont.)

Therefore, by the squeeze theorem

$$||T(\mathbf{v}) - \mathbf{v}|| = \lim_{n \to \infty} ||T(\mathbf{v}) - \mathbf{v}|| = 0.$$

By the definition of a norm, $T(\mathbf{v}) - \mathbf{v} = \mathbf{0}$, which implies $T(\mathbf{v}) = \mathbf{v}$, completing the proof.

Convergence of iterative policy evaluation

Proof. (cont.)

Our next step is to show that the fixed point of T is unique. Suppose that $T(\mathbf{u}) = \mathbf{u}$ and $T(\mathbf{v}) = \mathbf{v}$ for some vectors $\mathbf{u}$ and $\mathbf{v}$. In that case,

$$||\mathbf{u} - \mathbf{v}|| = ||T(\mathbf{u}) - T(\mathbf{v})|| \leq L||\mathbf{u} - \mathbf{v}||.$$

If we suppose that $||\mathbf{u} - \mathbf{v}|| > 0$, dividing the inequation by $||\mathbf{u} - \mathbf{v}||$ leads to the conclusion that $L \geq 1$. However, T is an L-contraction, contradicting our supposition. Therefore, $||\mathbf{u} - \mathbf{v}|| \leq 0$, which implies that $\mathbf{u} = \mathbf{v}$.

Convergence of iterative policy evaluation

Proof. (cont.)

Our last step is to show that $||\mathbf{v}_n - \mathbf{v}|| \leq L^n ||\mathbf{v}_0 - \mathbf{v}||$, for any n. The case $n = 0$ is trivial. Suppose that the inductive hypothesis is true for some n, and consider the case $n + 1$:

$$\begin{aligned} ||\mathbf{v}_{n+1} - \mathbf{v}|| &= ||T(\mathbf{v}_n) - T(\mathbf{v})|| && \text{(definition of fixed point)} \\ &\leq L||\mathbf{v}_n - \mathbf{v}|| \leq L^{n+1}||\mathbf{v}_0 - \mathbf{v}||, && \text{(inductive hypothesis)} \end{aligned}$$

as we wanted to show. $\square$

Convergence of iterative policy evaluation

Theorem (Convergence of iterative policy evaluation)

Consider the Bellman operator $T^\pi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ *for the policy* π. *Given an arbitrary* $\mathbf{v}_0 \in \mathbb{R}^{|\mathcal{S}|}$, *consider also the sequence* $(\mathbf{v}_n)_{n \geq 0}$ *where* $\mathbf{v}_{k+1} = T^\pi(\mathbf{v}_k)$. *Finally, consider the vector* $\mathbf{v}^\pi \in \mathbb{R}^{|\mathcal{S}|}$ *such that* $v_s^\pi = V^\pi(s)$, *for any* $s \in \mathcal{S}$. *For any* $n \geq 0$,

$$\mathbf{v}_n \underset{||\cdot||_\infty}{\to} \mathbf{v}^\pi,$$

$$\mathbf{v}^\pi = T^\pi(\mathbf{v}^\pi),$$

$$||\mathbf{v}_n - \mathbf{v}^\pi||_\infty \leq \gamma^n ||\mathbf{v}_0 - \mathbf{v}^\pi||_\infty.$$

Convergence of iterative policy evaluation

Proof.

As a first step, we show that $\mathbf{v}^\pi$ is a fixed point of T^π. For any $s \in \mathcal{S}$, by the definition of T^π and V^π,

$$T^\pi(\mathbf{v}^\pi)_s = \sum_a \pi(s, a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma v^\pi_{s'}\right] = v^\pi_s.$$

Our next step is to show that the Bellman operator $T^\pi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ is a γ-contraction. Because $(\mathbb{R}^{|\mathcal{S}|}, ||\cdot||_\infty)$ is a Banach space and $\mathbf{v}^\pi$ is a fixed point of T^π, the desired results follow from Banach's fixed point theorem.

Convergence of iterative policy evaluation

Proof. (cont.)

Note that, for any two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^{|\mathcal{S}|}$ and every state $s \in \mathcal{S}$,

$$\begin{aligned}
T^\pi(\mathbf{u})_s - T^\pi(\mathbf{v})_s &= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma u_{s'}\right] \\
&\quad - \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma v_{s'}\right] \\
&= \sum_a \sum_{s'} \pi(s,a) \mathcal{P}^a_{ss'} \gamma u_{s'} - \sum_a \sum_{s'} \pi(s,a) \mathcal{P}^a_{ss'} \gamma v_{s'} \\
&= \gamma \sum_a \sum_{s'} \pi(s,a) \mathcal{P}^a_{ss'} [u_{s'} - v_{s'}].
\end{aligned}$$

Convergence of iterative policy evaluation

Proof.

By the definition of maximum norm, for any two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^{|\mathcal{S}|}$,

$$
\begin{aligned}
&||T^\pi(\mathbf{u}) - T^\pi(\mathbf{v})||_\infty \\
&= \gamma \max_s \Big| \sum_a \sum_{s'} \pi(s,a) \mathcal{P}^a_{ss'} [u_{s'} - v_{s'}] \Big| && \text{(definition of maximum norm)} \\
&\leq \gamma \max_s \sum_a \sum_{s'} \Big| \pi(s,a) \mathcal{P}^a_{ss'} [u_{s'} - v_{s'}] \Big| && \text{(triangle inequality)} \\
&= \gamma \max_s \sum_a \sum_{s'} \pi(s,a) \mathcal{P}^a_{ss'} |u_{s'} - v_{s'}| && \text{(multiplicativity)} \\
&\leq \gamma \max_s \sum_a \sum_{s'} \pi(s,a) \mathcal{P}^a_{ss'} ||\mathbf{u} - \mathbf{v}||_\infty && \text{(definition of maximum norm)} \\
&= \gamma ||\mathbf{u} - \mathbf{v}||_\infty \max_s \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} && \text{(distributivity)} \\
&= \gamma ||\mathbf{u} - \mathbf{v}||_\infty && \text{(unit measure).}
\end{aligned}
$$

$\square$

Deterministic policies

- A deterministic policy π is one such that, for all $s \in \mathcal{S}$, $\pi(s, a) = 1$ for some $a \in \mathcal{A}$ and $\pi(s, b) = 0$ for all $b \neq a$
- In this case, we abuse notation and represent a policy by a function $\pi : \mathcal{S} \to \mathcal{A}$ from states to actions

Policy improvement

- Let π and π' be any pair of deterministic policies such that, for all $s \in \mathcal{S}$, $Q^\pi(s, \pi'(s)) \geq V^\pi(s)$
- The policy improvement theorem guarantees that $V^{\pi'}(s) \geq V^\pi(s)$ for all $s \in \mathcal{S}$
- For all $s \in \mathcal{S}$, a policy π may be improved to a policy π' by letting

$$\pi'(s) = \arg\max_a Q^\pi(s, a) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V^\pi(s')]$$

Policy iteration

- Policy evaluation and policy improvement can be interleaved
- This process produces the sequence

$$\pi_0, V^{\pi_0}, \pi_1, V^{\pi_1}, \pi_2, V^{\pi_2}, \dots$$

- If $\pi_t = \pi_{t+1}$, then π_t is optimal by the uniqueness of V^*
- The initial policy π_0 can be arbitrary

Value iteration

- A more efficient alternative iteratively improves the estimates for the value of each state under an optimal policy
- It relies on creating a sequence $V_0, V_1, \ldots$ of estimates given by:

$$V_{k+1}(s) = \max_a \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V_k(s')]$$

- The initial estimate V_0 can be arbitrary
- The sequence $V_0(s), V_1(s), \ldots$ converges to $V^*(s)$ for all $s \in \mathcal{S}$
- In-place value iteration has the same guarantees

Value iteration

Algorithm 2 Value iteration (in-place)

Input: one-step dynamics ($\mathcal{P}$ and $\mathcal{R}$), discount factor γ, and tolerance θ.
Output: optimal deterministic policy π when $\theta \to 0$.

for each $s \in \mathcal{S}$ **do**
  $V(s) \leftarrow 0$
end for
repeat
  $\Delta \leftarrow 0$
  for each $s \in \mathcal{S}$ **do**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V(s')]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  end for
until $\Delta < \theta$
for each $s \in \mathcal{S}$ **do**
  $\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V(s')]$
end for

Unknown one-step dynamics

- It is always possible to estimate the one-step dynamics by interacting with the environment
- For any $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$, the simplest (maximum likelihood) estimate for $\mathcal{P}^a_{ss'}$ is given by

$$\hat{\mathcal{P}}^a_{ss'} = \frac{N(s', s, a)}{N(s, a)},$$

  where $N(s, a) > 0$ is the number of times that action a was taken at state s, and $N(s', s, a)$ is the number of times that state s' was observed after action a was taken at state s
- The estimates $\hat{\mathcal{P}}^a_{ss'}$ and $\hat{\mathcal{R}}^a_{ss'}$ can be combined with value iteration
- Robust alternative: posterior sampling for reinforcement learning [Osband et al., 2013]

Exercises

1. Implement the environment described in p. 9. Note: a reward is obtained on going from (and not to) the trap/goal state to the absorbing state
2. Implement policy evaluation, policy improvement, policy iteration, and value iteration[4]
3. Consider an optimal policy for the environment described in p. 9. Using policy evaluation, manually compute the sequence $V_0, V_1, \dots, V_k$ for a small k. Let $V_0(s) = 0$ for all $s \in \mathcal{S}$. Compare this sequence to the sequence obtained by your implementation
4. Show that adding a constant to all the rewards in any given reinforcement learning problem simply adds a constant to the value of each state
5. (*) Show the recursivity of the action value function (Th. 2, p. 23)
6. (*) Show the relationship between V^π and Q^π (Th. 3, p. 24)
7. (*) Show the recursivity of the Bellman optimality equations (Th. 4, p. 26)

---

[4] You may want to study `https://github.com/paulorauber/rl`

Monte Carlo control

- Monte Carlo control methods find an optimal policy without estimating the one-step dynamics by interleaving policy evaluation and policy improvement
- These methods require an episodic problem, where there is a transition to an absorbing state after a finite number of time steps
- Policy evaluation for π consists of experiencing several episodes and averaging the returns that follow every possible state action pair (s, a) to obtain an estimate of $Q^\pi(s, a)$.
- In practice, "policy improvement" based on π is performed before a reliable estimate of Q^π is available

Exploration

- For a given state s, an ϵ-greedy policy with respect to an estimate Q of the action value function chooses a random action with probability ϵ, and an action $\arg\max_a Q(s, a)$ with probability $1 - \epsilon$
- Monte Carlo control typically relies on ϵ-greedy policies to ensure that the environment is explored sufficiently
- Exploration/exploitation trade-off: should the agent explore in order to learn about potentially new sources of reward or exploit the well-known sources of reward?

Algorithm 3 Monte Carlo control algorithm

Input: set of states $\mathcal{S}$, number of episodes N, probability of choosing random action ϵ.
Output: deterministic policy π, optimal when $N \to \infty$.

for each $s \in \mathcal{S}$ **do**
  for each action $a \in \mathcal{A}(s)$ **do**
    $Q(s, a) \leftarrow 0$
    $n(s, a) \leftarrow 0$
  end for
end for
for each i in $\{1, \dots, N\}$ **do**
  Experience a new episode e following an ϵ-greedy policy based on Q.
  for each state-action pair (s, a) in the episode e **do**
    $u \leftarrow$ return following (s, a) in the episode e.
    $n(s, a) \leftarrow n(s, a) + 1$
    $Q(s, a) \leftarrow Q(s, a) + \frac{1}{n(s,a)}[u - Q(s, a)]$
  end for
end for
for each state $s \in \mathcal{S}$ **do**
  $\pi(s) \leftarrow \arg\max_a Q(s, a)$
end for

Temporal difference

- Consider the tuple $h_t = (s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$ obtained by an agent using a policy π to interact with an environment
- Let Q denote an estimate of the action value function Q^π
- The one-step return based on h_t and Q is given by

$$r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$$

- The temporal difference for (s_t, a_t) based on h_t and Q is given by

$$r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$

- In other words, the difference between the immediate reward plus the (estimated) expected return from the next state and the (estimated) expected return for the current state

Sarsa control

- An algorithm bootstraps if it improves the estimate of the value of a state based on estimates of the values of other states
- Sarsa control is similar to Monte Carlo control, but it bootstraps based on temporal differences
- Sarsa control is comparatively more sample efficient, since it does not rely on the return that follows (s_t, a_t) after a single episode
- Given the tuple h_t and the estimate Q, Sarsa control updates its estimate of $Q(s_t, a_t)$ using

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)],$$

where α is the so-called learning rate

Sarsa control

Algorithm 4 Sarsa control algorithm

Input: set of states $\mathcal{S}$, number of episodes N, learning rate α, probability of random action ϵ, discount factor γ.

Output: deterministic policy π, optimal when $N \to \infty$ and α decays appropriately.

for each $(s, a) \in \mathcal{S} \times \mathcal{A}$ **do**
  $Q(s, a) \leftarrow 0$
end for
for each i in $\{1, \dots, N\}$ **do**
  $s \leftarrow$ initial state for episode i
  Select action a for state s according to an ϵ-greedy policy based on Q.
  while state s is not terminal **do**
    $r \leftarrow$ observed reward for action a at state s
    $s' \leftarrow$ observed next state for action a at state s
    Select action a' for state s' according to an ϵ-greedy policy based on Q.
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
    $s \leftarrow s'$
    $a \leftarrow a'$
  end while
end for
for each state $s \in \mathcal{S}$ **do**
  $\pi(s) \leftarrow \arg\max_a Q(s, a)$
end for

Q-learning

- An algorithm is off-policy if it learns about a policy that is different from the policy that it uses to act in the environment
- Q-learning learns about a greedy policy while acting using an ϵ-greedy policy
- Q-learning control is similar to Sarsa control: both algorithms bootstrap based on temporal differences
- Given the tuple h_t and the estimate Q, Q-learning control updates its estimate of $Q(s_t, a_t)$ using

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

where α is the so-called learning rate

Q-learning control

Algorithm 5 Q-learning control algorithm

Input: set of states $\mathcal{S}$, number of episodes N, learning rate α, probability of random action ϵ, discount factor γ.

Output: deterministic policy π, optimal when $N \to \infty$ and α decays appropriately.

for each $(s, a) \in \mathcal{S} \times \mathcal{A}$ **do**
    $Q(s, a) \leftarrow 0$
end for
for each i in $\{1, \dots, N\}$ **do**
    $s \leftarrow$ initial state for episode i
    while state s is not terminal **do**
        Select action a for state s according to an ϵ-greedy policy based on Q.
        $r \leftarrow$ observed reward for action a at state s
        $s' \leftarrow$ observed next state for action a at state s
        $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
        $s \leftarrow s'$
    end while
end for
for each state $s \in \mathcal{S}$ **do**
    $\pi(s) \leftarrow \arg\max_a Q(s, a)$
end for

Exercises

1. Implement an interactive version of the environment described in p. 9. This implementation must provide an initial state and respond to actions by transitioning between states and providing rewards.
2. Implement Monte Carlo control, Sarsa control, and Q-learning control [5]
3. Compare the results of these implementations to the results obtained by your implementations of policy iteration and value iteration
4. (*) Implement the "Cliff World" environment and compare your results to those of [Greydanus and Olah, 2019]

[5] You may want to study `https://github.com/paulorauber/rl`

Generalization

- In large state spaces, some states may be seen very rarely
- In these cases, the state or action value estimates should generalize across states
- Generalization relies on function approximation, which is studied extensively in machine learning
- The state value function $V^\pi : \mathcal{S} \to \mathbb{R}$ can be approximated by a parametric function $V : \mathcal{S} \times \mathbb{R}^m \to \mathbb{R}$
- The goal of policy evaluation becomes finding a $\boldsymbol{\theta}$ such that

$$V^\pi(s) \approx V(s; \boldsymbol{\theta}),$$

  for every $s \in \mathcal{S}$
- Changing $\boldsymbol{\theta}$ changes the value estimates of several states

Value regression

- For a given policy π, consider a dataset $\mathcal{D} = \{(s_i, V^\pi(s_i))\}_{i=1}^N$
- The mean squared error $J(\boldsymbol{\theta})$ is given by

$$J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} [V^\pi(s_i) - V(s_i; \boldsymbol{\theta})]^2$$

- Goal: finding a parameter vector $\boldsymbol{\theta}^*$ such that $J(\boldsymbol{\theta}^*) = \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

Stochastic gradient descent

- The procedure starts with an arbitrary estimate $\boldsymbol{\theta}_0$
- For any $t \geq 0$, a pair $(s_t, V^\pi(s_t))$ is drawn at random from $\mathcal{D}$, and the estimate $\boldsymbol{\theta}_{t+1}$ is obtained using

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{1}{2}\alpha \nabla_{\boldsymbol{\theta}} [V^\pi(s_t) - V(s_t; \boldsymbol{\theta}_t)]^2$$

  where α is the learning rate
- By the chain rule,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha [V^\pi(s_t) - V(s_t; \boldsymbol{\theta}_t)] \nabla_{\boldsymbol{\theta}} V(s_t; \boldsymbol{\theta}_t)$$

- If α decays appropriately, this procedure converges to a local optimum of J

Value regression from estimates

- If $V^{\pi}(s)$ were available for all states $s \in \mathcal{S}$, there would be no need for function approximation
- In practice, a dataset will be given by $\mathcal{D} = \{(s_i, v_i)\}_{i=1}^{N}$, where v_i is an estimate of the value of s_i under policy π
- Different estimates v_i may be considered, such as the empirical return or one-step return observed after state s_i
- For any $t \geq 0$, a pair (s_t, v_t) is drawn at random from $\mathcal{D}$, and the estimate $\boldsymbol{\theta}_{t+1}$ is obtained using

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha[v_t - V(s_t; \boldsymbol{\theta}_t)]\nabla_{\boldsymbol{\theta}} V(s_t; \boldsymbol{\theta}_t)$$

Gradient descent TD value estimation

Algorithm 6 Gradient descent TD value estimation algorithm

Input: policy π, number of episodes N, learning rate α, discount factor γ
Output: parameter vector $\boldsymbol{\theta}$

Initialize $\boldsymbol{\theta}$ arbitrarily
for each i in $\{1, \dots, N\}$ **do**
    $s \leftarrow$ initial state for episode i
    while state s is not terminal **do**
        $a \leftarrow \pi(s)$
        $r \leftarrow$ observed reward for action a at state s
        $s' \leftarrow$ observed next state for action a at state s
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha[r + \gamma V(s'; \boldsymbol{\theta}) - V(s; \boldsymbol{\theta})] \nabla_{\boldsymbol{\theta}} V(s; \boldsymbol{\theta})$
        $s \leftarrow s'$
    end while
end for

Linear value functions

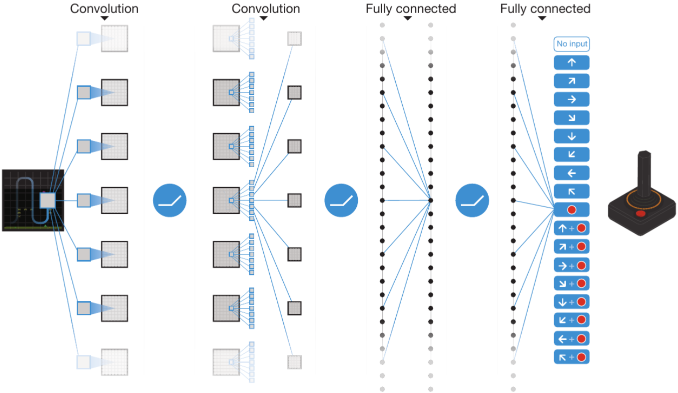
- Suppose that any state $s \in \mathcal{S}$ can be represented by a feature vector $\boldsymbol{\phi}(s) \in \mathbb{R}^m$, and that $V(s; \boldsymbol{\theta})$ is given by

$$V(s; \boldsymbol{\theta}) = \sum_{i=1}^{m} \theta_i \phi(s)_i$$

- Note that $\nabla_{\boldsymbol{\theta}} V(s; \boldsymbol{\theta}) = \boldsymbol{\phi}(s)$
- Several algorithms have strong convergence guarantees for this case

Deep Q-Networks

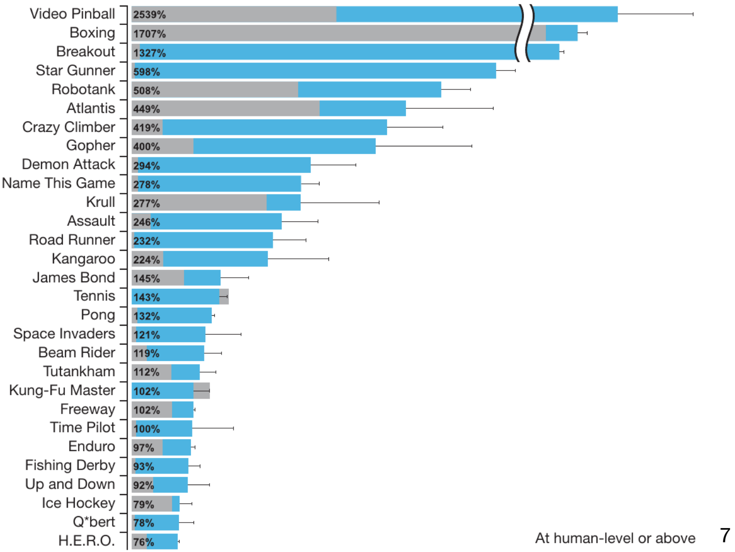
- $Q : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^m \to \mathbb{R}$ is represented by a neural network



[6]

[6] Image from [Mnih et al., 2015]

Deep Q-Networks



[7] Image from [Mnih et al., 2015]

Deep Q-Networks: preprocessing

- A sequence of images obtained from the emulator is preprocessed before being presented to the network
- Individually for each color channel, an elementwise maximum operation is employed between two consecutive images to reduce rendering artifacts
- Such $210 \times 160 \times 3$ preprocessed image is converted to grayscale, cropped, and rescaled into an 84×84 image $\mathbf{x}_k$
- A sequence of images $\mathbf{x}_{k-12}, \mathbf{x}_{k-8}, \mathbf{x}_{k-4}, \mathbf{x}_k$ obtained in this way is <u>stacked</u> into an $84 \times 84 \times 4$ image $\mathbf{s}$

Deep Q-Networks: architecture

- The image $\mathbf{s}_t$ is input to a neural network architecture given by:
  - Convolutional layer with 32 rectified linear filters (8×8, stride 4)
  - Convolutional layer with 64 rectified linear filters (4×4, stride 2)
  - Convolutional layer with 64 rectified linear filters (3×3, stride 1)
  - Fully-connected layer with 512 rectified linear units
  - Fully-connected layer with $|\mathcal{A}|$ <u>linear</u> units
- Each output unit represents $Q(\mathbf{s}_t, a; \boldsymbol{\theta})$ for a different action $a \in \mathcal{A}$

Deep Q-networks: algorithm

Algorithm 7 Deep Q-learning with experience replay

Input: replay buffer size M, number of episodes N, maximum episode length T, probability of random action ϵ, frame skip K, batch size B, learning rate α, number of episodes between target network updates C.
Output: estimate $Q(\cdot; \boldsymbol{\theta})$ of the optimal action value function Q^*

Initialize replay buffer $\mathcal{D}$, which stores at most M tuples
Initialize network parameters $\boldsymbol{\theta}$ randomly
$\boldsymbol{\theta}' = \boldsymbol{\theta}$
for each i in $\{1, \dots, N\}$ **do**
    $\mathbf{s}_0 \leftarrow$ initial state for episode i
    for each t in $\{0, \dots, T-1\}$ **do**
        if random() $< 1 - \epsilon$ **then** $a_t \leftarrow \arg\max_a Q(\mathbf{s}_t, a; \boldsymbol{\theta})$ **else** $a_t \leftarrow$ random action
        Obtain the next state $\mathbf{s}_{t+1}$ and reward r_{t+1} by repeating action a_t during K frames
        if the episode ends at step $t+1$ **then** $\Omega_{t+1} \leftarrow 1$ **else** $\Omega_{t+1} \leftarrow 0$
        Store the tuple $(\mathbf{s}_t, \mathbf{a}_t, r_{t+1}, \mathbf{s}_{t+1}, \Omega_{t+1})$ in the replay buffer $\mathcal{D}$
        Sample a subset $\mathcal{D}' \subset \mathcal{D}$ composed of B tuples
        Let $L(\boldsymbol{\theta}) = \sum_{(\mathbf{s}, a, r, \mathbf{s}', \Omega') \in \mathcal{D}'} (y - Q(\mathbf{s}, a; \boldsymbol{\theta}))^2$
        In the equation above, let $y = r + \gamma \max_{a'} Q(\mathbf{s}', a'; \boldsymbol{\theta}')$ if $\Omega' = 0$, and $y = r$ if $\Omega' = 1$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$, noting that $\boldsymbol{\theta}'$ is considered a constant with respect to $\boldsymbol{\theta}$
    end for
    if $i \mod C = 0$ **then**
        $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}$
    end if
end for

Policy gradient methods

- Consider an agent that interacts with its environment in a sequence of episodes, each of which lasts for exactly T time steps
- Let $\boldsymbol{\tau} = s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T, s_T$ denote a trajectory in a particular episode
- Under the Markov assumption, the probability $p(\boldsymbol{\tau} \mid \boldsymbol{\theta})$ of trajectory $\boldsymbol{\tau}$ given the policy parameters $\boldsymbol{\theta}$ is given by

$$p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) = p(s_0) \prod_{t=0}^{T-1} p(s_{t+1}, r_{t+1} \mid s_t, a_t) p(a_t \mid s_t, \boldsymbol{\theta}),$$

  where $p(a_t \mid s_t, \boldsymbol{\theta})$ is the probability of action a_t given state s_t and policy parameters $\boldsymbol{\theta}$

Policy gradient methods

- The expected return $J(\boldsymbol{\theta})$ of a policy parameterized by $\boldsymbol{\theta}$ is given by

$$J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=1}^{T} R_t \mid \boldsymbol{\theta}\right] = \sum_{t=1}^{T} \mathbb{E}\left[R_t \mid \boldsymbol{\theta}\right]$$

- Goal: finding a parameter vector $\boldsymbol{\theta}^*$ such that $J(\boldsymbol{\theta}^*) = \max_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

Policy gradient methods

Theorem (Policy gradient theorem)

The gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ of the expected return $J(\boldsymbol{\theta})$ is given by

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log p(A_t \mid S_t, \boldsymbol{\theta}) \sum_{t'=t+1}^{T} R_{t'} \mid \boldsymbol{\theta}\right].$$

Policy gradient methods

- The gradient of the expected return is a sum of expected values of random vectors that correspond to each time step
- In gradient ascent, the expected value for time step t weights a direction that locally increases the probability of each possible decision by its expected (positive or negative) outcome
- Positive expected outcomes contribute towards making the probability of a decision higher
- Negative expected outcomes contribute towards making the probability of a decision lower.

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log p(A_t \mid S_t, \boldsymbol{\theta}) \sum_{t'=t+1}^{T} R_{t'} \mid \boldsymbol{\theta}\right]$$

Policy gradient methods

- Consider a sequence $\boldsymbol{\tau}_1, \dots, \boldsymbol{\tau}_N$ of N trajectories obtained by following the policy parameterized by $\boldsymbol{\theta}$, and let

$$\boldsymbol{\tau}_i = s_{i,0}, a_{i,0}, r_{i,1}, s_{i,1}, a_{i,1}, r_{i,2}, \dots, s_{i,T-1}, a_{i,T-1}, r_{i,T}, s_{i,T}$$

- A Monte Carlo estimate $\hat{\mathbf{g}}(\boldsymbol{\theta})$ to $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is given by

$$\begin{aligned}
\hat{\mathbf{g}}(\boldsymbol{\theta}) &= \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log p(a_{i,t} \mid s_{i,t}, \boldsymbol{\theta}) \sum_{t'=t+1}^{T} r_{i,t'} \\
&= \nabla_{\boldsymbol{\theta}} \left[\frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \log p(a_{i,t} \mid s_{i,t}, \boldsymbol{\theta}) \sum_{t'=t+1}^{T} r_{i,t'} \right]
\end{aligned}$$

and may be used for gradient <u>ascent</u> on J.

Policy gradient theorem

Theorem (Policy gradient theorem)

The gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ of the expected return $J(\boldsymbol{\theta})$ is given by

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log p(A_t \mid S_t, \boldsymbol{\theta}) \sum_{t'=t+1}^{T} R_{t'} \mid \boldsymbol{\theta}\right].$$

Policy gradient theorem

Proof.

Using the law of the unconscious statistician,

$$J(\boldsymbol{\theta}) = \sum_{\boldsymbol{\tau}} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) \sum_{t=1}^{T} r_t = \sum_{t=1}^{T} \sum_{\boldsymbol{\tau}} r_t p(\boldsymbol{\tau} \mid \boldsymbol{\theta}).$$

Assuming $J(\boldsymbol{\theta})$ is differentiable with respect to $\boldsymbol{\theta}$, the partial derivative $\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$ of J with respect to θ_j at $\boldsymbol{\theta}$ is given by

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \sum_{t=1}^{T} \sum_{\boldsymbol{\tau}} r_t \frac{\partial}{\partial \theta_j} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}).$$

Policy gradient theorem

Proof.

Suppose that $p(\boldsymbol{\tau} \mid \boldsymbol{\theta})$ is positive for any $\boldsymbol{\tau}$ and $\boldsymbol{\theta}$. The so-called likelihood ratio trick uses the fact that

$$\frac{\partial}{\partial \theta_j} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) = p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) \frac{1}{p(\boldsymbol{\tau} \mid \boldsymbol{\theta})} \frac{\partial}{\partial \theta_j} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) = p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) \frac{\partial}{\partial \theta_j} \log p(\boldsymbol{\tau} \mid \boldsymbol{\theta}).$$

By using the previous expression for $\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$,

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \sum_{t=1}^{T} \sum_{\boldsymbol{\tau}} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) r_t \frac{\partial}{\partial \theta_j} \log p(\boldsymbol{\tau} \mid \boldsymbol{\theta}).$$

Policy gradient theorem

Proof.

Because we have already assumed that $p(\boldsymbol{\tau} \mid \boldsymbol{\theta})$ is positive for all $\boldsymbol{\tau}$ and $\boldsymbol{\theta}$,

$$\log p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) = \log p(s_0) + \sum_{t=0}^{T-1} \log p(s_{t+1}, r_{t+1} \mid s_t, a_t) + \sum_{t=0}^{T-1} \log p(a_t \mid s_t, \boldsymbol{\theta}).$$

Therefore,

$$\frac{\partial}{\partial \theta_j} \log p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) = \sum_{t'=0}^{T-1} \frac{\partial}{\partial \theta_j} \log p(a_{t'} \mid s_{t'}, \boldsymbol{\theta}).$$

Policy gradient theorem

Proof.

By using the previous expression for $\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$,

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \sum_{t=1}^{T} \sum_{\boldsymbol{\tau}} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) r_t \left[\sum_{t'=0}^{T-1} \frac{\partial}{\partial \theta_j} \log p(a_{t'} \mid s_{t'}, \boldsymbol{\theta}) \right].$$

It will be useful to split the innermost summation in the expression above into before and after t, leading to

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \sum_{t=1}^{T} \sum_{\boldsymbol{\tau}} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) \left[r_t \sum_{t'=0}^{t-1} \frac{\partial}{\partial \theta_j} \log p(a_{t'} \mid s_{t'}, \boldsymbol{\theta}) + r_t \sum_{t'=t}^{T-1} \frac{\partial}{\partial \theta_j} \log p(a_{t'} \mid s_{t'}, \boldsymbol{\theta}) \right].$$

Policy gradient theorem

Proof.

Alternatively, the expression above can be written as

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \sum_{t=1}^{T} \sum_{t'=0}^{t-1} \mathbb{E}\left[R_t \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid S_{t'}, \boldsymbol{\theta}) \mid \boldsymbol{\theta} \right] + \sum_{t=1}^{T} \sum_{t'=t}^{T-1} \mathbb{E}\left[R_t \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid S_{t'}, \boldsymbol{\theta}) \mid \boldsymbol{\theta} \right].$$

We will now show that the rightmost nested summations in the expression above can be dismissed.

Policy gradient theorem

Proof.

By representing the random variables involved in a trajectory using a Bayesian network, it can be seen that $A_{t'} \perp\!\!\!\perp R_t \mid S_{t'}, \boldsymbol{\theta}$ for $t' \geq t$. The analogous statement is not generally true for $t' < t$.
For $t' \geq t$, this independence leads to

$$\mathbb{E}\left[R_t \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid S_{t'}, \boldsymbol{\theta}) \mid \boldsymbol{\theta}\right] = \sum_{r_t} \sum_{a_{t'}} \sum_{s_{t'}} p(a_{t'} \mid s_{t'}, \boldsymbol{\theta}) p(r_t, s_{t'} \mid \boldsymbol{\theta}) r_t \frac{\partial}{\partial \theta_j} \log p(a_{t'} \mid s_{t'}, \boldsymbol{\theta}).$$

By reversing the likelihood-ratio trick,

$$\mathbb{E}\left[R_t \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid S_{t'}, \boldsymbol{\theta}) \mid \boldsymbol{\theta}\right] = \sum_{r_t} \sum_{a_{t'}} \sum_{s_{t'}} p(r_t, s_{t'} \mid \boldsymbol{\theta}) r_t \frac{\partial}{\partial \theta_j} p(a_{t'} \mid s_{t'}, \boldsymbol{\theta}).$$

Policy gradient theorem

Proof.

By changing the order of summations and pushing constants outside the innermost summation,

$$\mathbb{E}\left[R_t \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid S_{t'}, \boldsymbol{\theta}) \mid \boldsymbol{\theta}\right] = \sum_{r_t} \sum_{s_{t'}} p(r_t, s_{t'} \mid \boldsymbol{\theta}) r_t \sum_{a_{t'}} \frac{\partial}{\partial \theta_j} p(a_{t'} \mid s_{t'}, \boldsymbol{\theta}).$$

Finally, using the fact that $\frac{\partial}{\partial \theta_j} 1 = 0$,

$$\mathbb{E}\left[R_t \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid S_{t'}, \boldsymbol{\theta}) \mid \boldsymbol{\theta}\right] = \sum_{r_t} \sum_{s_{t'}} p(r_t, s_{t'} \mid \boldsymbol{\theta}) r_t \frac{\partial}{\partial \theta_j} \sum_{a_{t'}} p(a_{t'} \mid s_{t'}, \boldsymbol{\theta}) = 0.$$

Policy gradient theorem

Proof.

We may now remove the rightmost nested summations in the previous expression for $\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$, which gives

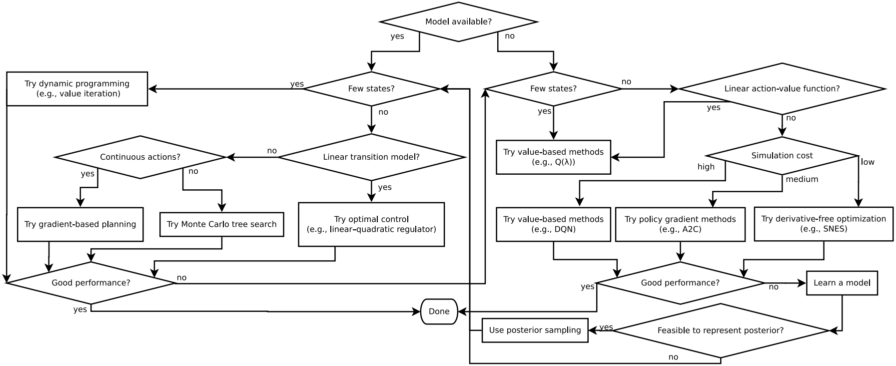
$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=1}^{T} R_t \sum_{t'=0}^{t-1} \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid S_{t'}, \boldsymbol{\theta}) \mid \boldsymbol{\theta}\right].$$

By reordering the summations, the expression above can be conveniently rewritten as

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{T-1} \frac{\partial}{\partial \theta_j} \log p(A_t \mid S_t, \boldsymbol{\theta}) \sum_{t'=t+1}^{T} R_{t'} \mid \boldsymbol{\theta}\right].$$

□

Solving a reinforcement learning problem



Additional reading

- Reinforcement learning: an introduction [Sutton and Barto, 2018]
- Algorithms for reinforcement learning [Szepesvári, 2010]
- Notes on reinforcement learning [Rauber, 2015]
- UCL course on reinforcement learning [Silver, 2015]
- Deep reinforcement learning [Levine, 2018]
- Stanford course on reinforcement learning [Ng, 2008]
- Deep reinforcement learning bootcamp [Abbeel et al., 2017]
- Stanford course on reinforcement learning [Brunskill, 2019]

References I

Abbeel, P. et al. (2017).
Deep reinforcement learning bootcamp.
https://sites.google.com/view/deep-rl-bootcamp/lectures.

Brockman, G. et al. (2019a).
OpenAI five.
https://openai.com/five/.

Brockman, G. et al. (2019b).
Solving rubik's cube with a robot hand.
https://openai.com/blog/solving-rubiks-cube/.

Brunskill, E. (2019).
CS234: Reinforcement learning.
http://web.stanford.edu/class/cs234/schedule.html.

References II

Greydanus, S. and Olah, C. (2019).
The paths perspective on value learning.
https://distill.pub/2019/paths-perspective-on-value-learning/.

Hutter, M. (2004).
Universal artificial intelligence: Sequential decisions based on algorithmic probability.
Springer Science & Business Media.

Klein, D., Abbeel, P., and Dragan, A. (2019).
CS 188: Introduction to artificial intelligence.
https://inst.eecs.berkeley.edu/~cs188/fa19/.

Levine, S. (2018).
Deep reinforcement learning.
http://rail.eecs.berkeley.edu/deeprlcourse-fa18/.

References III

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015).
Human-level control through deep reinforcement learning.
Nature, 518(7540):529.

Ng, A. (2008).
CS 229: Machine learning.
https://www.youtube.com/watch?v=RtxI449ZjSc.

Osband, I., Russo, D., and Van Roy, B. (2013).
(more) efficient reinforcement learning via posterior sampling.
In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, Advances in Neural Information Processing Systems 26, pages 3003–3011. Curran Associates, Inc.

References IV

Rauber, P. (2015).
Notes on reinforcement learning.
http://paulorauber.com/work.html.

Silver, D. (2015).
UCL course on reinforcement learning.
http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2018).
A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play.
Science, 362(6419):1140–1144.

References V

Sutton, R. S. and Barto, A. G. (2018).
Reinforcement Learning: An Introduction.
The MIT Press, second edition.

Szepesvári, C. (2010).
Algorithms for reinforcement learning.
Synthesis lectures on artificial intelligence and machine learning, 4(1):1–103.