PA230: Reinforcement Learning

Petr Novotný

"Good and evil, reward and punishment, are the only motives to a rational creature; these are the spur and reins whereby all mankind are set on work and guided."

John Locke, *Some Thoughts Concerning Education* (1693)

Organizational Information

General

- Lecture: Thursdays 2-3:40p.m.

General

- Lecture: Thursdays 2-3:40p.m.
- Homework: see the interactive syllabus in IS
  - mainly binary classification (accepted/not accepted)
  - all your homeworks need to be marked as passed to proceed to exam
  - can (but do not have to) be done in pairs (pairs can differ across the individual assignments)
  - for those who passed, the teacher will receive feedback on the general quality of the solutions for each student - can be taken into account when determining the final grade (typically in students' favor)

General

- Lecture: Thursdays 2-3:40p.m.
- Homework: see the interactive syllabus in IS
  - mainly binary classification (accepted/not accepted)
  - all your homeworks need to be marked as passed to proceed to exam
  - can (but do not have to) be done in pairs (pairs can differ across the individual assignments)
  - for those who passed, the teacher will receive feedback on the general quality of the solutions for each student - can be taken into account when determining the final grade (typically in students' favor)
- Exam:
  - oral
  - each attempt counts ? (unlike the Brázdil system)
  - in general, knowledge of anything mentioned on the slides can be required, unless explicitly marked with "nex" (like the Brázdil system)

Team

- Lecturer: Petr Novotný

- HW team:

Martin Kurečka

Václav Nevyhoštěný

Vít Unčovský

Communication

Official discord server:

`https://discord.gg/9mxTgYhcdB`

- Official communication forum of the course: falls under the university ethical guidelines.
- Use your real name for posting (you can set-up an account under your IS email if necessary).

Reading

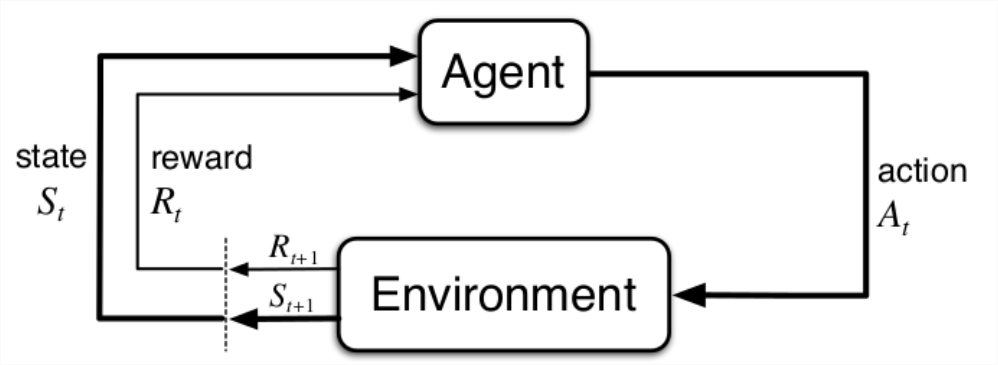
- Compulsory:
  - these slides,
  - material explicitly prescribed by these slides (not much).
- Recommended:
  - Sutton & Barto: Reinforcement Learning: An Introduction (2nd ed.), available at `https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf`
    - henceforth referenced as "S&B"
  - slides by David Silver `https://www.davidsilver.uk/teaching/`
  - CMU slides `https://www.andrew.cmu.edu/course/10-703/`
  - more specific literature recommendations will be given for each topic later

Reinforcement Learning: What, Why, When, How, & Other Questions

Types of machine learning

- unsupervised
  - spot "useful" patterns in data
- supervised
  - given labeled data, predict labels on unlabeled data
- reinforcement
  - agents and decision-making
  - agency = "the ability to take action or to choose what action to take" (Cambridge Dictionary)

General RL scheme



source: Sutton&Barto, p. 48

Keywords: sequential, dynamic, subject to uncertainty

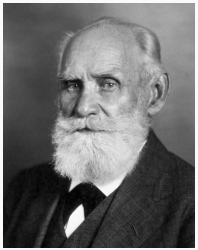
RL: Objective and approach

- Objective: Design a decision policy (= agent behavior) which prescribes to the agent how to act in different situations (states), typically so as to achieve some goal.

RL: Objective and approach

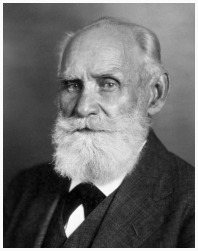
- Objective: Design a decision policy (= agent behavior) which prescribes to the agent how to act in different situations (states), typically so as to achieve some goal.
- Approach: Start with ($\pm$ random) behavior and adapt it based on past experience via the law of effect:
  - actions with good/bad consequences for the agent are more/less likely to be repeated by the agent (within the same context)

RL in psychology (nex)

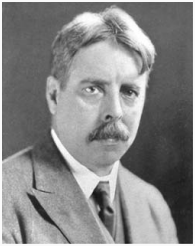


I.P. Pavlov
(1849-1936)
classical conditioning

RL in psychology (nex)

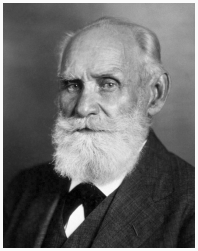


I.P. Pavlov
(1849-1936)
classical conditioning

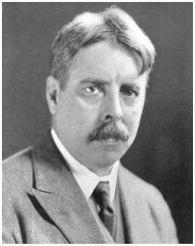


E. Thorndike
(1874-1949)
law of effect

RL in psychology (nex)



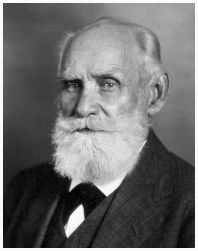
I.P. Pavlov
(1849-1936)
classical conditioning



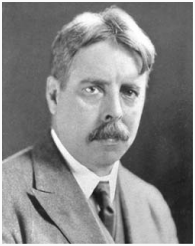
E. Thorndike
(1874-1949)
law of effect

J.B. Watson
(1878-1958)
behaviorist manifesto

RL in psychology (nex)

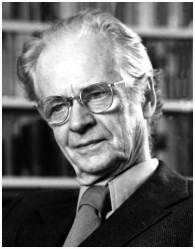


I.P. Pavlov
(1849-1936)
classical conditioning



E. Thorndike
(1874-1949)
law of effect

J.B. Watson
(1878-1958)
behaviorist manifesto



B.F. Skinner
(1904-1990)
radical behaviorism,
reinforcement,
rewards

Learning by trying & XX dilemma

Underlying the RL approach is the idea of learning by trying:

- first, act more or less randomly (exploration)
  - integral part of early human development
- continually adapt behavior according to experience and feedback from the environment (exploitation)
  - strength of feedback $\approx$ strength of behavior adaptation

Balancing exploration and exploitation (XX) is a recurring theme in RL.

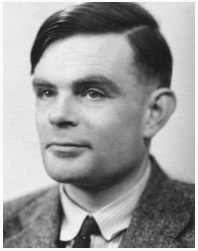
Incomplete history of RL in computer science I

"Learning by trying" machines and software, ad hoc approaches:

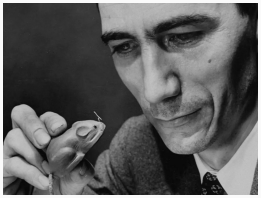
A. Turing
(1912-1954)
1948: theoretical
"pleasure & pain" system
to train computers

Incomplete history of RL in computer science I

"Learning by trying" machines and software, ad hoc approaches:



A. Turing
(1912-1954)
1948: theoretical
"pleasure & pain" system
to train computers

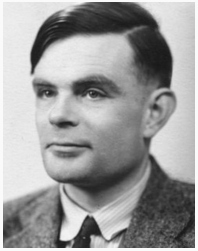


C. Shannon
(1916-2001)
1950: Theseus
maze-solving mouse

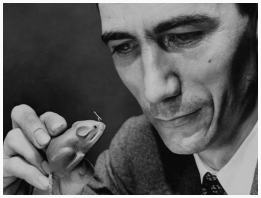
M. Minsky
(1927-2016)
1950s: analog neural net
machines (SNARCS)

Incomplete history of RL in computer science I

"Learning by trying" machines and software, ad hoc approaches:



A. Turing
(1912-1954)
1948: theoretical
"pleasure & pain" system
to train computers



C. Shannon
(1916-2001)
1950: Theseus
maze-solving mouse

M. Minsky
(1927-2016)
1950s: analog neural net
machines (SNARCS)

And many more...
Recommended: S&B: Sec. 1.7.

Incomplete history of RL in computer science II

Mathematical foundations of sequential decision making:

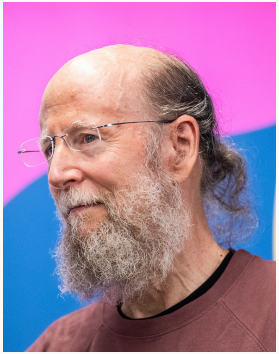
R. Bellman
(1920-1984)

R. Howard
(b. 1934)

- Formalization via Markov decision processes (MDPs)
- value iteration
  (attributed to Bellman, 1957)
- policy iteration
  (attributed to Howard, 1960)

Incomplete history of RL in computer science III

Since late 1980's: synthesis – learning by trial in MDPs



R. Sutton

A. Barto

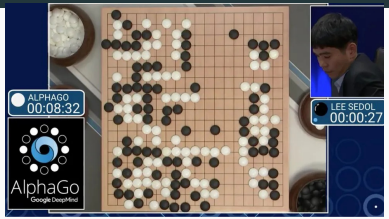
C. Watkins

Temporal difference learning

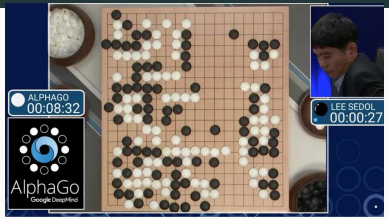
Q-learning

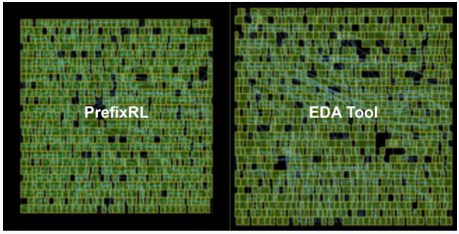
. . . and many more.

Successes of RL (nex)

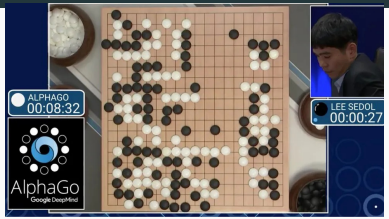


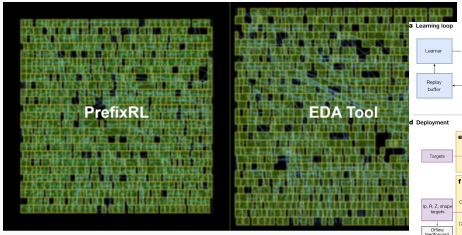
Successes of RL (nex)

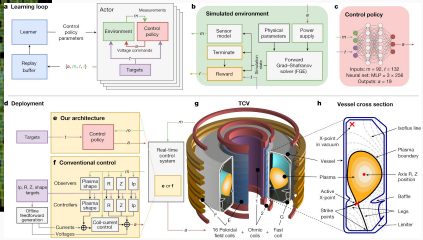




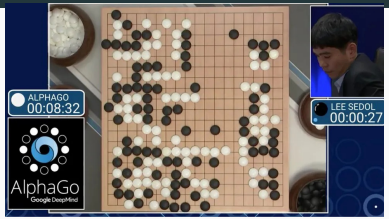
Successes of RL (nex)

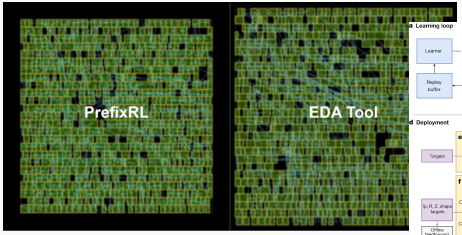


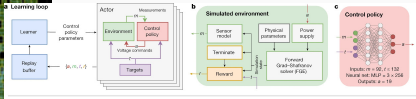


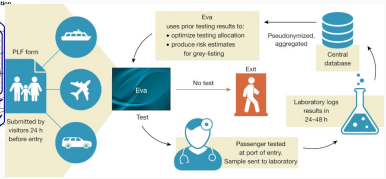


Successes of RL (nex)

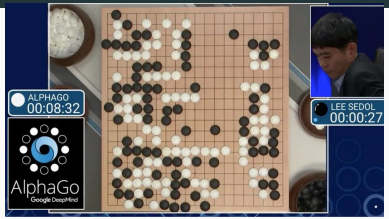


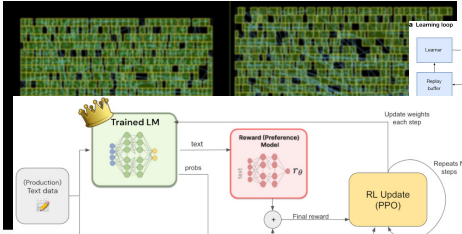


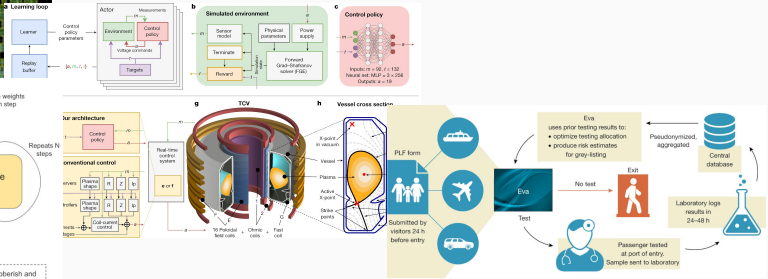




Successes of RL (nex)







Words of caution (and controversy) (nex)

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

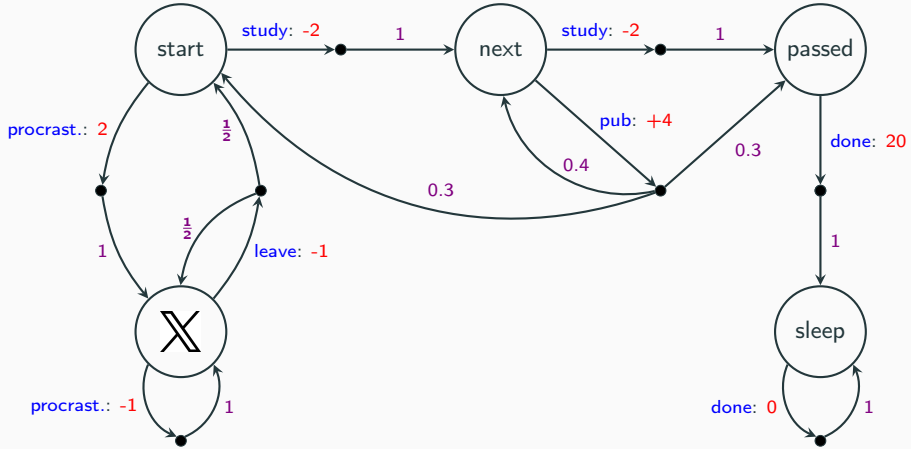
Words of caution (and controversy) (nex)

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

Mathematical Foundations of Sequential Decision-Making

MDP Example

MDP with actions, rewards and transition probabilities.



Markov Decision Process

Given a set X, we denote $\mathcal{D}(X)$ the set of all probability distributions over X.

Definition 1

A Markov decision process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, p, r)$ where

- $\mathcal{S}$ is a set of states,
- $\mathcal{A}$ is a set of actions,
- $p\colon \mathcal{S} \times \mathcal{A} \to \mathcal{D}(\mathcal{S})$ is a probabilistic transition function,
- $r\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function.

We will shorten $p(s, a)(s')$ to $p(s' \mid s, a)$.

Markov Decision Process

Given a set X, we denote $\mathcal{D}(X)$ the set of all probability distributions over X.

Definition 1

A Markov decision process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, p, r)$ where

- $\mathcal{S}$ is a set of states,
- $\mathcal{A}$ is a set of actions,
- $p\colon \mathcal{S} \times \mathcal{A} \to \mathcal{D}(\mathcal{S})$ is a probabilistic transition function,
- $r\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function.

We will shorten $p(s,a)(s')$ to $p(s' \mid s, a)$.

The p, r can be partial functions: action a is enabled in state s if both $p(s,a)$ and $r(s,a)$ are defined. We denote by $\mathcal{A}(s)$ the set of all actions enabled in s.

Dynamics of MDPs

- start in some initial state s_0

Dynamics of MDPs

- start in some initial state s_0
- MDP evolves in discrete time steps $t = 0, 1, 2, 3, \ldots$

Dynamics of MDPs

- start in some initial state s_0
- MDP evolves in discrete time steps $t = 0, 1, 2, 3, \ldots$
- in each time step t, let s_t be the current state; then:
  - agent selects action $a_t \in \mathcal{A}(s_t)$
  - the environment responds with next state $s_{t+1} \sim p(s_t, a_t)$ and with immediate reward $r_{t+1} = r(s_t, a_t)$
  - t is incremented and the process repeats in the same fashion forever

Thus, the agent produces a trajectory $\tau = s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, \ldots$.

Dynamics of MDPs

- start in some initial state s_0
- MDP evolves in discrete time steps $t = 0, 1, 2, 3, \ldots$
- in each time step t, let s_t be the current state; then:
  - agent selects action $a_t \in \mathcal{A}(s_t)$
  - the environment responds with next state $s_{t+1} \sim p(s_t, a_t)$ and with immediate reward $r_{t+1} = r(s_t, a_t)$
  - t is incremented and the process repeats in the same fashion forever

Thus, the agent produces a trajectory $\tau = s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, \ldots$.

τ is produced randomly (due to p and possibly also agent choices being probabilistic): it is a random variable and so are its components: we define random variables

- S_t = state at time step t
- A_t = action at time step t
- R_t = reward received just before entering S_t

Policies

Definition 2

A history is a finite prefix of a trajectory ending in a state, i.e., an object of type

$$s_0, a_0, r_1, s_1, a_1, r_2, \ldots, a_{t-1}, r_t, s_t \in (\mathcal{S} \cdot \mathcal{A} \cdot \mathbb{R})^* \mathcal{S}.$$

we denote by $last(h)$ the last state of a history h.

Policies

Definition 2

A history is a finite prefix of a trajectory ending in a state, i.e., an object of type

$$s_0, a_0, r_1, s_1, a_1, r_2, \ldots, a_{t-1}, r_t, s_t \in (\mathcal{S} \cdot \mathcal{A} \cdot \mathbb{R})^* \mathcal{S}.$$

we denote by $last(h)$ the last state of a history h.

Definition 3

A policy is a function $\pi \colon (\mathcal{S} \cdot \mathcal{A} \cdot \mathbb{R})^* \mathcal{S} \to \mathcal{D}(\mathcal{A})$ which to each history h assigns a probability distribution over $\mathcal{A}(last(h))$.

A policy is by definition an infinite object!

MD policies

Definition 4

A policy π is:

- memoryless if $\pi(h) = \pi(h')$ whenever $last(h) = last(h')$ (we can view memoryless policies as objects of type $\pi\colon \mathcal{S} \to \mathcal{D}(\mathcal{A})$);
- deterministic if $\pi(h)$ always assigns probability 1 to one action, and zero to all others (we can view det. policies of objects of type $\pi\colon (\mathcal{S} \cdot \mathcal{A} \cdot \mathbb{R})^* \mathcal{S} \to \mathcal{A}$).

MD policies

Definition 4

A policy π is:

- memoryless if $\pi(h) = \pi(h')$ whenever $last(h) = last(h')$ (we can view memoryless policies as objects of type $\pi \colon \mathcal{S} \to \mathcal{D}(\mathcal{A})$);
- deterministic if $\pi(h)$ always assigns probability 1 to one action, and zero to all others (we can view det. policies of objects of type $\pi \colon (\mathcal{S} \cdot \mathcal{A} \cdot \mathbb{R})^* \mathcal{S} \to \mathcal{A}$).

Definition 5

A policy π is MD (memoryless deterministic) if it is both memoryless and deterministic.

Dynamics of MDPs (more precise)

Given a distribution $\mathcal{I}$ of initial states and a policy π

- start in some initial state $s_0 \sim \mathcal{I}$

Dynamics of MDPs (more precise)

Given a distribution $\mathcal{I}$ of initial states and a policy π

- start in some initial state $s_0 \sim \mathcal{I}$
- MDP evolves in discrete time steps $t = 0, 1, 2, 3, \ldots$

Dynamics of MDPs (more precise)

Given a distribution $\mathcal{I}$ of initial states and a policy π

- start in some initial state $s_0 \sim \mathcal{I}$
- MDP evolves in discrete time steps $t = 0, 1, 2, 3, \ldots$
- in each time step t, let h_t be the history produced so far; then:
  - agent selects action $a_t \in \mathcal{A}(s_t)$ according to π, i.e. $a_t \sim \pi(h_t)$
  - the environment responds with next state $s_{t+1} \sim p(s_t, a_t)$ and with immediate reward $r_{t+1} = r(s_t, a_t)$, the history is extended by a_t, r_t, s_{t+1},
  - t is incremented and the process repeats in the same fashion forever

Probability space induced by a policy

In particular, each policy π together with a distribution $\mathcal{I}$ of initial states induce a probability measure $\mathbb{P}^\pi$ over the trajectories of the MDP.[1]

[1] $\mathcal{I}$ is typically known from the context and hence omitted from the notation

Probability space induced by a policy

In particular, each policy π together with a distribution $\mathcal{I}$ of initial states induce a probability measure $\mathbb{P}^\pi$ over the trajectories of the MDP.[1]

We denote by $\mathbb{E}^\pi$ the associated expected value (expectation) operator.

We denote by $\mathbb{P}^\pi[E \mid S_0 = s]$ the probability of event E provided that the initial state is fixed to s (and similarly for expectations).

[1] $\mathcal{I}$ is typically known from the context and hence omitted from the notation

Probability space induced by a policy

In particular, each policy π together with a distribution $\mathcal{I}$ of initial states induce a probability measure $\mathbb{P}^\pi$ over the trajectories of the MDP.[1]

We denote by $\mathbb{E}^\pi$ the associated expected value (expectation) operator.

We denote by $\mathbb{P}^\pi[E \mid S_0 = s]$ the probability of event E provided that the initial state is fixed to s (and similarly for expectations).

Exercise 6

In the "study" MDP, consider an MD policy π s.t. $\pi(\mathit{start}) = \mathit{study}$ and $\pi(\mathit{next}) = \mathit{pub}$. Compute the following quantities:

[1] $\mathcal{I}$ is typically known from the context and hence omitted from the notation

Probability space induced by a policy

In particular, each policy π together with a distribution $\mathcal{I}$ of initial states induce a probability measure $\mathbb{P}^\pi$ over the trajectories of the MDP.[1]

We denote by $\mathbb{E}^\pi$ the associated expected value (expectation) operator.

We denote by $\mathbb{P}^\pi[E \mid S_0 = s]$ the probability of event E provided that the initial state is fixed to s (and similarly for expectations).

Exercise 6

In the "study" MDP, consider an MD policy π s.t. $\pi(\mathit{start}) = \mathit{study}$ and $\pi(\mathit{next}) = \mathit{pub}$. Compute the following quantities:

- $\mathbb{P}^\pi[\text{visit pub at least twice}' \mid S_0 = \mathit{start}'']$

[1] $\mathcal{I}$ is typically known from the context and hence omitted from the notation

Probability space induced by a policy

In particular, each policy π together with a distribution $\mathcal{I}$ of initial states induce a probability measure $\mathbb{P}^\pi$ over the trajectories of the MDP.[1]

We denote by $\mathbb{E}^\pi$ the associated expected value (expectation) operator.

We denote by $\mathbb{P}^\pi[E \mid S_0 = s]$ the probability of event E provided that the initial state is fixed to s (and similarly for expectations).

Exercise 6

In the "study" MDP, consider an MD policy π s.t. $\pi(\mathit{start}) = \mathit{study}$ and $\pi(\mathit{next}) = \mathit{pub}$. Compute the following quantities:

- $\mathbb{P}^\pi[\text{visit pub at least twice}' \mid S_0 = \mathit{start}'']$
- $\mathbb{P}^\pi[\text{visit pub at exactly twice} \mid S_0 = \mathit{start}'']$

[1] $\mathcal{I}$ is typically known from the context and hence omitted from the notation

Probability space induced by a policy

In particular, each policy π together with a distribution $\mathcal{I}$ of initial states induce a probability measure $\mathbb{P}^\pi$ over the trajectories of the MDP.[1]

We denote by $\mathbb{E}^\pi$ the associated expected value (expectation) operator.

We denote by $\mathbb{P}^\pi[E \mid S_0 = s]$ the probability of event E provided that the initial state is fixed to s (and similarly for expectations).

Exercise 6

In the "study" MDP, consider an MD policy π s.t. $\pi(\mathit{start}) = \mathit{study}$ and $\pi(\mathit{next}) = \mathit{pub}$. Compute the following quantities:

- $\mathbb{P}^\pi[\text{visit pub at least twice}' \mid S_0 = \mathit{start}'']$
- $\mathbb{P}^\pi[\text{visit pub at exactly twice} \mid S_0 = \mathit{start}'']$
- $\mathbb{E}^\pi[R_1]$

[1] $\mathcal{I}$ is typically known from the context and hence omitted from the notation

Probability space induced by a policy

In particular, each policy π together with a distribution $\mathcal{I}$ of initial states induce a probability measure $\mathbb{P}^\pi$ over the trajectories of the MDP.[1]

We denote by $\mathbb{E}^\pi$ the associated expected value (expectation) operator.

We denote by $\mathbb{P}^\pi[E \mid S_0 = s]$ the probability of event E provided that the initial state is fixed to s (and similarly for expectations).

Exercise 6

In the "study" MDP, consider an MD policy π s.t. $\pi(\mathit{start}) = \mathit{study}$ and $\pi(\mathit{next}) = \mathit{pub}$. Compute the following quantities:

- $\mathbb{P}^\pi[\text{visit pub at least twice}' \mid S_0 = \mathit{start}'']$
- $\mathbb{P}^\pi[\text{visit pub at exactly twice} \mid S_0 = \mathit{start}'']$
- $\mathbb{E}^\pi[R_1]$
- $\mathbb{E}^\pi[R_3]$

Memorylessness

In this course, we will almost exclusively focus on memoryless policies. Hence, from now on, policy = memoryless policy. General policies will be referred to as history-dependent policies should the need arise.

Memorylessness

In this course, we will almost exclusively focus on memoryless policies. Hence, from now on, policy = memoryless policy. General policies will be referred to as history-dependent policies should the need arise.

Why memoryless?

Intuition: Markov property of MDPs: next step depends only on the current state and on action performed in the current step. Hence, intuitively there is no need for a policy to remember the past so as to "play well".

The sufficiency of memoryless policies does not extended to more general/complex decision-making settings (not covered in this course), such as:

- partially observable MDPs
- non-stationary environments
- quantile/risk-aware MDPs, etc.

Returns (payoffs)

Returns (payoffs)

Definition 7

Let $\gamma \in [0, 1)$ be a discount factor.
For a trajectory $\tau = s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, \ldots$ we define the discounted return (or payoff) of τ to be the quantity

$$G(\tau) = r_1 + \gamma \cdot r_2 + \gamma^2 \cdot r^3 + \cdots \gamma^3 \cdot r^4 = \sum_{i=0}^{\infty} \gamma^i \cdot r_{i+1}.$$

Equivalently

$$G = \sum_{i=0}^{\infty} \gamma^i \cdot R_{i+1}.$$

Returns (variants)

- Finite horizon (FH): additionally, we are given a finite decision horizon $H \in \mathbb{N} \cup \{\infty\}$. The return is that counted only up to step H:

$$G^H = \sum_{i=0}^{H-1} \gamma^i \cdot R_{i+1}$$

For finite H, the discount factor can be 1. $H = \infty$ corresponds to the original definition.

Returns (variants)

- Finite horizon (FH): additionally, we are given a finite decision horizon $H \in \mathbb{N} \cup \{\infty\}$. The return is that counted only up to step H:

$$G^H = \sum_{i=0}^{H-1} \gamma^i \cdot R_{i+1}$$

  For finite H, the discount factor can be 1. $H = \infty$ corresponds to the original definition.
- Episodic returns: In episodic tasks, there is a distinguished set $Term \subseteq \mathcal{S}$ of terminal states which is guaranteed to be reached with probability 1 under any policy. We denote by T a random variable denoting the first point in time when we hit a terminal state. We count rewards only up to that time:

$$G^T = \sum_{i=1}^{T-1} \gamma^i \cdot R_{i+1}$$

  Can be modeled under original definition by "sink" states.

Types of returns: discussion

- We will typically omit the superscripts since the type of task considered will be known from the context.
- We have $G^H \to G$ (pointwise) as $H \to \infty$. I.e., finite-horizon returns with high enough H approximate the standard (infinite-horizon) case.
- In real world, we typically deal with FH or episodic tasks: we cannot wait infinite time to learn something from a trajectory. However, the infinite-horizon case can be viewed as a neat mathematical abstraction of the FH&episodic tasks, and the classical sequential decision-making theory is most developed for the infinite horizon case.

Policy and state values

Definition 8

Let π be a policy and s a state. The value of π in state s is the quantity

$$v^\pi(s) = \mathbb{E}^\pi[G \mid S_0 = s].$$

Exercise 9

Discuss the values of MD policies in our running example.

Policy and state values

Definition 8

Let π be a policy and s a state. The value of π in state s is the quantity

$$v^\pi(s) = \mathbb{E}^\pi[G \mid S_0 = s].$$

Exercise 9

Discuss the values of MD policies in our running example.

Definition 10

The (optimal) value of state s is the quantity

$$v^*(s) = \sup_\pi v^\pi(s).$$

Optimality

Definition 11

Let π be a policy and $\varepsilon > 0$. We say that π is ε-optimal in state s if

$$v^{\pi}(s) \geq v^{*}(s) - \varepsilon.$$

We say that π is optimal in s is it is 0-optimal in s, i.e. if

$$v^{\pi}(s) = v^{*}(s).$$

A policy is (ε-)optimal if it is (ε-)optimal in every state.

Existence of optimal policies

Theorem 12: (Classical result, not formally proven here)

Let $\mathcal{M}$ be a finite MDP (i.e., the state and action sets are finite) with infinite-horizon returns. Then there exists an optimal MD policy. Moreover, an optimal MD policy can be computed in polynomial time.

Existence of optimal policies

Theorem 12: (Classical result, not formally proven here)

Let $\mathcal{M}$ be a finite MDP (i.e., the state and action sets are finite) with infinite-horizon returns. Then there exists an optimal MD policy. Moreover, an optimal MD policy can be computed in polynomial time.

Agent control solved?

Existence of optimal policies

Theorem 12: (Classical result, not formally proven here)

Let $\mathcal{M}$ be a finite MDP (i.e., the state and action sets are finite) with infinite-horizon returns. Then there exists an optimal MD policy. Moreover, an optimal MD policy can be computed in polynomial time.

Agent control solved? NO! "Only" works if you can actually construct the MDP model of your environment and fit it into a computer. Otherwise, we use reinforcement learning.

Exact Planning with Known Model: Value & Policy Iteration

Goal of this lecture

Algorithms that compute the optimal value vector v^* and some optimal MD policy π^* given a full knowledge of an MDP $\mathcal{M}$.

Polynomial-time algorithm

MDPs can be solved by linear programming (LP)

$$\begin{aligned} &\text{maximize } \vec{c} \cdot \vec{x} \\ &\text{subject to } A \cdot \vec{x} \leq \vec{b} \end{aligned}$$

- LP can be solved in polynomial time by so-called interior-point algorithms.
- However, we typically use other, MDP-specific algorithms: value iteration (VI) and policy iteration (PI). These are not polynomial-time in general, but typically faster on practical instances.
- Moreover, most truly RL algorithms can be seen as approximate generalizations of VI or PI (or both).

Example: Policy evaluation

Exercise 13

Consider all four MD policies in our running "pub or study" example. Compute the values of these policies in the initial state *start*.

Policy evaluation equations

Theorem 14

For any memoryless policy π and any state s it holds:

$$v^\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) \cdot \underbrace{\left[r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s'|s,a) \cdot v^\pi(s') \right]}_{\stackrel{\text{def}}{=} q^\pi(s,a)}.$$

Bellman optimality equations

Theorem 15

The following holds for any state s:

$$v^*(s) = \max_{a \in \mathcal{A}(s)} \underbrace{\left[r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s'|s,a) \cdot v^*(s') \right]}_{\stackrel{\text{def}}{=} q^*(s,a)}$$

Bellman optimality equations

Theorem 15

The following holds for any state s:

$$v^*(s) = \max_{a \in \mathcal{A}(s)} \underbrace{\left[r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s'|s,a) \cdot v^*(s') \right]}_{\stackrel{\text{def}}{=} q^*(s,a)}$$

- Note: the policy evaluation equations are a special case of the Bellman ones: given a policy π, we can consider an MDP $\mathcal{M}^\pi$ in which there is a single action $*$ enabled in each state and the probability of transition $s \stackrel{*}{\to} s'$ equals $\sum_{a \in \mathcal{A}(s)} \pi(a|s) \cdot p(s'|s,a)$. Then $\mathcal{M}^\pi$ mimics the behavior of π in $\mathcal{M}$ and Bellman eq's in $\mathcal{M}^\pi$ = evaluation equations for π in $\mathcal{M}$.

Bellman optimality equations

Theorem 15

The following holds for any state s:

$$v^*(s) = \max_{a \in \mathcal{A}(s)} \underbrace{\left[r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s'|s,a) \cdot v^*(s') \right]}_{\stackrel{\text{def}}{=} q^*(s,a)}$$

- Note: the policy evaluation equations are a special case of the Bellman ones: given a policy π, we can consider an MDP $\mathcal{M}^\pi$ in which there is a single action $*$ enabled in each state and the probability of transition $s \xrightarrow{*} s'$ equals $\sum_{a \in \mathcal{A}(s)} \pi(a|s) \cdot p(s'|s,a)$. Then $\mathcal{M}^\pi$ mimics the behavior of π in $\mathcal{M}$ and Bellman eq's in $\mathcal{M}^\pi$ = evaluation equations for π in $\mathcal{M}$.
- But these equations are no longer linear! How do we solve them? Is the solution even unique?

Bellman update operator

The right-hand-side (RHS) of the Bellman equations can be viewed as an operator $\Phi\colon \mathbb{R}^{\mathcal{S}} \to \mathbb{R}^{\mathcal{S}}$: for any $\vec{x} \in \mathbb{R}^{\mathcal{S}}$, $\Phi(\vec{x})$ is a vector such that for any state s:

$$\Phi(\vec{x})(s) \stackrel{\text{def}}{=} \max_{a \in \mathcal{A}(s)} \left[r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s'|s,a) \cdot \vec{x}(s') \right]$$

Bellman update operator

The right-hand-side (RHS) of the Bellman equations can be viewed as an operator $\Phi\colon \mathbb{R}^{\mathcal{S}} \to \mathbb{R}^{\mathcal{S}}$: for any $\vec{x} \in \mathbb{R}^{\mathcal{S}}$, $\Phi(\vec{x})$ is a vector such that for any state s:

$$\Phi(\vec{x})(s) \stackrel{\text{def}}{=} \max_{a \in \mathcal{A}(s)} \left[r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s'|s,a) \cdot \vec{x}(s') \right]$$

Exercise 16

In our running example, compute $\Phi(\vec{0})$.

Bellman update operator

The right-hand-side (RHS) of the Bellman equations can be viewed as an operator $\Phi\colon \mathbb{R}^{\mathcal{S}} \to \mathbb{R}^{\mathcal{S}}$: for any $\vec{x} \in \mathbb{R}^{\mathcal{S}}$, $\Phi(\vec{x})$ is a vector such that for any state s:

$$\Phi(\vec{x})(s) \stackrel{\text{def}}{=} \max_{a \in \mathcal{A}(s)} \left[r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s'|s,a) \cdot \vec{x}(s') \right]$$

Exercise 16

In our running example, compute $\Phi(\vec{0})$.

Theorem 15 says that the optimal value vector v^* is a fixed point of Φ:

$$v^* = \Phi(v^*).$$

Lemma 17: (not proven here)

For any discount factor $\gamma \in [0, 1)$, the Bellman operator Φ is a contraction, i.e. for any pair of vectors $\vec{x}, \vec{y}$ it holds

$$\|\vec{x} - \vec{y}\|_\infty \leq \gamma \cdot \|\Phi(\vec{x}) - \Phi(\vec{y})\|_\infty.$$

Lemma 17: (not proven here)

For any discount factor $\gamma \in [0, 1)$, the Bellman operator Φ is a contraction, i.e. for any pair of vectors $\vec{x}, \vec{y}$ it holds

$$\|\vec{x} - \vec{y}\|_\infty \leq \gamma \cdot \|\Phi(\vec{x}) - \Phi(\vec{y})\|_\infty.$$

Theorem 18: Banach fixed point theorem (classical calculus, not proven here)

A contraction mapping from a complete metric space (in particular, $\mathbb{R}^n$) to itself has a unique fixed point.

Exact characterization of v^*

Corollary 19

The optimal value vector is a unique solution of the Bellman optimality equations.

Exact characterization of v^*

Corollary 19

The optimal value vector is a unique solution of the Bellman optimality equations.

In particular, also the policy evaluation equations have a unique solution, equal to v^π. Since the policy evaluation equations are linear, their solution can be computed by Gaussian elimination.

But the general Bellman equations are not linear. How can ve solve them?

Banach fixpoint theorem (full)

Theorem 20: Banach fixed point theorem (full version, not proven here)

A contraction mapping Φ from a complete metric space (in particular, $\mathbb{R}^n$) to itself has a unique fixed point $\vec{z}$.

Moreover, $\vec{z}$ is the limit of iterative applications of Φ on any initial vector. I.e., for any $\vec{x_0} \in \mathbb{R}^n$, the sequence $\vec{x_0}, \Phi(\vec{x_0}), \Phi(\Phi(\vec{x_0})), \Phi^{(3)}(\vec{x_0}), \ldots$ converges to $\vec{z}$:

$$z = \lim_{i \to \infty} \Phi^{(i)}(\vec{x_0})$$

Value iteration (VI; Bellman, 1957)

Algorithm 1: Value iteration

Input: MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$

Output: Approximation $\tilde{v}$ of v^*

$x \leftarrow$ any vector from $\mathbb{R}^{|\mathcal{S}|}$; `// typically` $\vec{0}$

$next \leftarrow x$;

repeat

  foreach $s \in \mathcal{S}$ **do**

    $next(s) \leftarrow \underbrace{\max_{a \in \mathcal{A}(s)} [r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s'|s,a) \cdot \vec{x}(s')]}_{\Phi(x)(s)}$;

  $x \leftarrow next$

until *termination condition*;

Typical term. conditions:

- after a fixed no. of iterations (i.e., use for-loop with a fixed bound)
- after each component of x changes less then some given ε

How to use VI

By the Banach fixpoint theorem (and Lemma 17), the value of variable x VI converges to v^*. Can we recognize when is x "close enough" to $\vec{x}$?

How to use VI

By the Banach fixpoint theorem (and Lemma 17), the value of variable x VI converges to v^*. Can we recognize when is x "close enough" to $\vec{x}$?

In the following couple of theorems, let $\vec{x}_0, \vec{x}_1, \vec{x}_2, \ldots$ be the sequence of vectors computed by VI, i.e. $\vec{x}_0$ is arbitrary and $\vec{x}_{i+1} = \Phi(\vec{x}_i)$ for all $i \geq 0$.

Theorem 21: Stopping condition (not proven here)

For any $\varepsilon > 0$: if

$$\|\vec{x}_{i+1} - \vec{x}_i\|_\infty \leq \varepsilon \cdot \frac{1-\gamma}{\gamma},$$

then

$$\|\vec{x}_{i+1} - v^*\|_\infty \leq \varepsilon$$

How to use VI (2)

How fast can we get to the point where we are close enough?

Theorem 22: Speed of convergence (not proven here)

For all $i \geq 0$ it holds

$$\|\vec{x}^n - v^*\|_\infty \leq \frac{\gamma^n}{1-\gamma} \cdot \|\vec{x}_1 - \vec{x}_0\|_\infty.$$

In particular, if we terminate VI after

$$i = \left\lceil \frac{\log(\varepsilon) + \log\left(\frac{1-\gamma}{\|\vec{x}_1 - \vec{x}_0\|_\infty}\right)}{\log(\gamma)} \right\rceil$$

steps, then its output x_i will be an ε-approximation of v^*.

How to use VI (3)

Can we actually get some optimal values instead of approximations?

How to use VI (3)

Can we actually get some optimal values instead of approximations? First, note that VI computes optimal finite-horizon values:

Let $v^i = \sup_\pi \mathbb{E}^\pi[\sum_{i=1}^{H} \gamma^{i-1} \cdot R_i]$. The supremum is over all (i.e., history dependent) policies, since in the FH problem an optimal policy needs to track the number of elapsed (and thus remaining) steps: memory is needed for that.

Theorem 23: (Easy but important exercise)

If $\vec{x}_0 = \vec{0}$, then $\vec{x}_H = v^H$ for all $H \geq 0$.

How to use VI (3)

Can we actually get some optimal values instead of approximations? First, note that VI computes optimal finite-horizon values:

Let $v^i = \sup_\pi \mathbb{E}^\pi[\sum_{i=1}^H \gamma^{i-1} \cdot R_i]$. The supremum is over all (i.e., history dependent) policies, since in the FH problem an optimal policy needs to track the number of elapsed (and thus remaining) steps: memory is needed for that.

Theorem 23: (Easy but important exercise)

If $\vec{x}_0 = \vec{0}$, then $\vec{x}_H = v^H$ for all $H \geq 0$.
Moreover, let π^H be a deterministic history-dependent policy such that for all $1 \leq i \leq H$, whenever there are i steps remaining till the horizon, the policy π^H selects in state s an action a s.t.

$$a = \arg\max_{a \in \mathcal{A}(s)} [r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s'|s,a) \cdot \vec{x}_{i-1}(s')]$$

(with ties broken arbitrarily). Then π^H is an optimal H-step policy.

Greedy policies

Can we actually get optimal policy for the inf. horizon problem?

Definition 24: $\vec{x}$-greedy policy (very important)

Let $\vec{x} \in \mathbb{R}^{\mathcal{S}}$ be any vector. A $\vec{x}$-greedy policy is an MD policy π such that in any state s:

$$\pi(s) = \arg\max_{a \in \mathcal{A}(s)} [r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s'|s,a) \cdot \vec{x}(s')].$$

How to use VI (4)

Theorem 25: Optimal inf.-horizon policy from VI (not proven here)

There is a number N polynomial in size of the MDP and exponential in the binary encoding size of γ such that a policy π that is $\vec{x}_N$-greedy is optimal in every state, i.e. $v^\pi = v^*$.

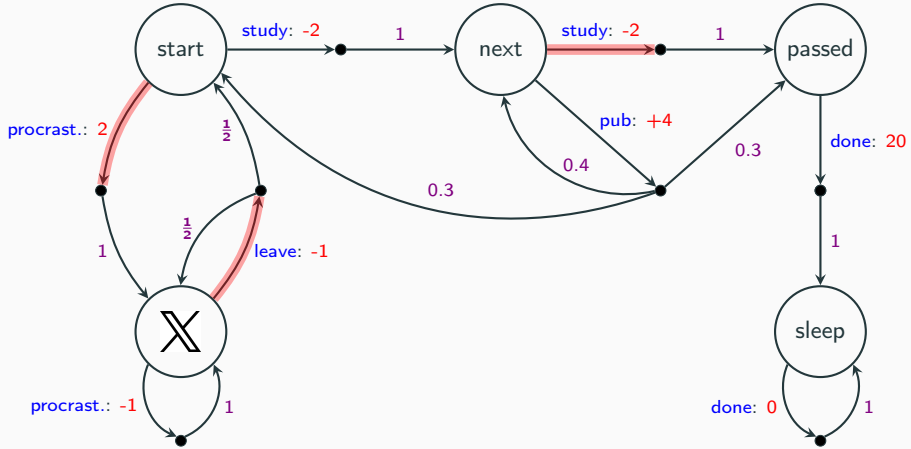
Note that once π is computed, v^π can be computed in polynomial time via policy evaluation equations.

Hence, VI can be said to solve MDPs in exponential time (and in polynomial time if the discount factor is assumed to be a fixed constant instead of an input parameter), though the approximate version is typically used in practice.

Note: the fact that some policy π is $\vec{x}$-greedy does not mean that $v^\pi \geq \vec{x}$! Homework: find a counterexample and post it to Discord.

However, for VI it can be shown that if $\|\vec{x}_{i+1} - \vec{x}_i\|_\infty \leq \varepsilon \cdot \frac{1-\gamma}{\gamma}$ (stopping condition from Theorem 21), then an $\vec{x}_{i+1}$-greedy policy is ε-optimal.

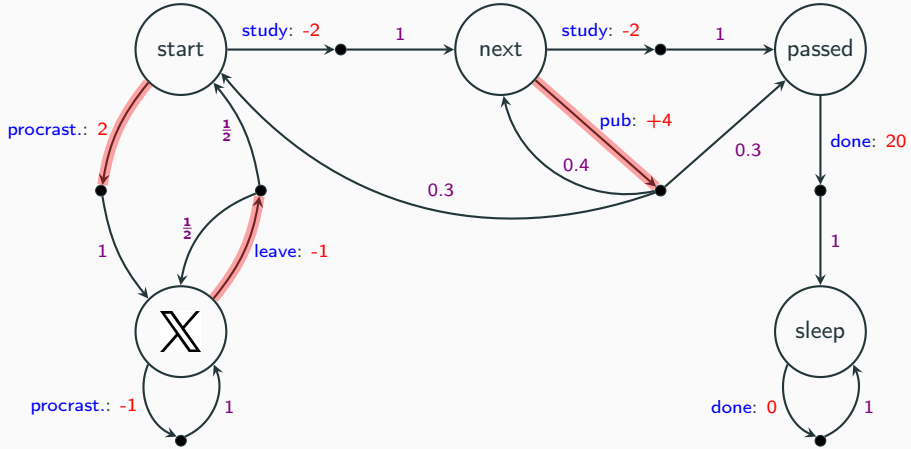
Policy improvement



$v^\pi = ($ $)$

let π' be v^π-greedy

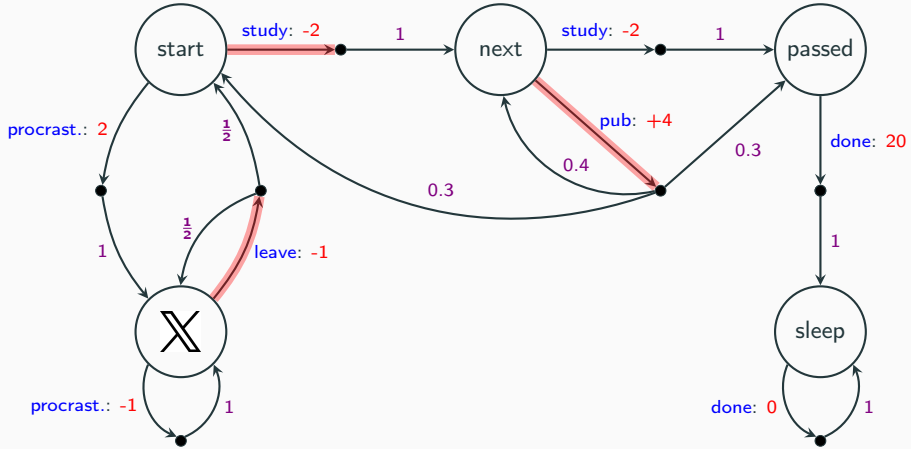
Policy improvement



$v^{\pi'} = ($ $)$

let π'' be $v^{\pi'}$-greedy

Policy improvement



Policy improvement theorem

Theorem 26: Policy improvement

Let π be a policy. If $\Phi(v^\pi) \geq v^\pi$, then any v^π-greedy policy π_g is at least as good as π, i.e. $\forall s \in \mathcal{S} : v^{\pi_g}(s) \geq v^\pi(s)$.
Moreover, if $\Phi(v^\pi)(s) > v^\pi(s)$ for some state s, then also $v^{\pi_g}(s') > v^\pi(s')$ for some state s'.

Returns from a given time step

For the proof of PIT and also many times later, we will need the following notation:

Definition 27: Important!

Let $\tau = s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, \dots$ be a trajectory and $t \in \mathbb{N}$ a time step. We define

$$G_t(\tau) = \sum_{i=t}^{H-1} \gamma^{i-t} \cdot r_{i+1} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots,$$

where $H \in \mathbb{N} \cup \{\infty\}$ or $H = T$ for episodic tasks.
We similarly define, for any policy π:

$$G_t^\pi = \mathbb{E}^\pi[G_t] = \mathbb{E}^\pi[\sum_{i=t}^{H-1} \gamma^{i-t} \cdot R_{i+1}].$$

Proof of PIT (setup)

We will define a sequence of policies $\pi_0, \pi_1, \pi_2, \ldots$ s.t.:

- $\pi_0 = \pi$
- π_i behaves as π_g (i.e., selects the same actions in same states) for the first i steps, then "switches" back to behave as π:
- we also define $\pi_\infty = \pi_g$

Proof of PIT (setup)

We will define a sequence of policies $\pi_0, \pi_1, \pi_2, \ldots$ s.t.:

- $\pi_0 = \pi$
- π_i behaves as π_g (i.e., selects the same actions in same states) for the first i steps, then "switches" back to behave as π:

- we also define $\pi_\infty = \pi_g$

We want: $v^{\pi_\infty}(s) \geq v^{\pi}(s)$ for all s.

Proof of PIT (setup)

We will define a sequence of policies $\pi_0, \pi_1, \pi_2, \ldots$ s.t.:

- $\pi_0 = \pi$
- π_i behaves as π_g (i.e., selects the same actions in same states) for the first i steps, then "switches" back to behave as π:

- we also define $\pi_\infty = \pi_g$

We want: $v^{\pi_\infty}(s) \geq v^{\pi}(s)$ for all s.

Not hard to see: $v^{\pi_i} \to v^{\pi_\infty}$ as $i \to \infty$ (π_i behaves as π_∞ for longer and longer as i increases + discounting).
It suffices to show: $v^{\pi_i}(s) \geq v^{\pi}(s)$ for all $i \in \mathbb{N}$ and all $s \in \mathcal{S}$.

Proof of PIT (induction)

$v^{\pi_i}(s) \geq v^{\pi}(s)$ for all $i \in \mathbb{N}$ and all $s \in \mathcal{S}$

- $i = 0$: clear

Proof of PIT (induction)

$v^{\pi_i}(s) \geq v^{\pi}(s)$ for all $i \in \mathbb{N}$ and all $s \in \mathcal{S}$

- $i = 0$: clear
- $i > 0$:

$$v^{\pi_i}(s) = \mathbb{E}^{\pi_i}[R_1 + \gamma R_2 + \cdots + \gamma^{i-1} R_i + \gamma^i R_{i+1} + \cdots \mid S_0 = s]$$

Proof of PIT (induction)

$v^{\pi_i}(s) \geq v^{\pi}(s)$ for all $i \in \mathbb{N}$ and all $s \in \mathcal{S}$

- $i = 0$: clear
- $i > 0$:

$$
\begin{aligned}
v^{\pi_i}(s) &= \mathbb{E}^{\pi_i}[R_1 + \gamma R_2 + \cdots + \gamma^{i-1} R_i + \gamma^i R_{i+1} + \cdots \mid S_0 = s] \\
&= \mathbb{E}^{\pi_i}[R_1 + \gamma R_2 + \cdots + \gamma^{i-2} R_{i-1} \mid S_0 = s] + \mathbb{E}^{\pi_i}[\gamma^{i-1} R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]
\end{aligned}
$$

Proof of PIT (induction)

$v^{\pi_i}(s) \geq v^{\pi}(s)$ for all $i \in \mathbb{N}$ and all $s \in \mathcal{S}$

- $i = 0$: clear
- $i > 0$:

$$
\begin{aligned}
v^{\pi_i}(s) &= \mathbb{E}^{\pi_i}[R_1 + \gamma R_2 + \cdots + \gamma^{i-1} R_i + \gamma^i R_{i+1} + \cdots \mid S_0 = s] \\
&= \mathbb{E}^{\pi_i}[R_1 + \gamma R_2 + \cdots + \gamma^{i-2} R_{i-1} \mid S_0 = s] + \mathbb{E}^{\pi_i}[\gamma^{i-1} R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] \\
&= \mathbb{E}^{\pi_{i-1}}[R_1 + \gamma R_2 + \cdots + \gamma^{i-2} R_{i-1} \mid S_0 = s] + \mathbb{E}^{\pi_i}[\gamma^{i-1} R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]
\end{aligned}
$$

Proof of PIT (induction)

$v^{\pi_i}(s) \geq v^{\pi}(s)$ for all $i \in \mathbb{N}$ and all $s \in \mathcal{S}$

- $i = 0$: clear
- $i > 0$:

$$
\begin{aligned}
v^{\pi_i}(s) &= \mathbb{E}^{\pi_i}[R_1 + \gamma R_2 + \cdots + \gamma^{i-1} R_i + \gamma^i R_{i+1} + \cdots \mid S_0 = s] \\
&= \mathbb{E}^{\pi_i}[R_1 + \gamma R_2 + \cdots + \gamma^{i-2} R_{i-1} \mid S_0 = s] + \mathbb{E}^{\pi_i}[\gamma^{i-1} R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] \\
&= \mathbb{E}^{\pi_{i-1}}[R_1 + \gamma R_2 + \cdots + \gamma^{i-2} R_{i-1} \mid S_0 = s] + \underbrace{\mathbb{E}^{\pi_i}[\gamma^{i-1} R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]}_{\text{Suppose we prove } \geq \mathbb{E}^{\pi_{i-1}}[\gamma^{i-1} R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]}
\end{aligned}
$$

Proof of PIT (induction)

$v^{\pi_i}(s) \geq v^{\pi}(s)$ for all $i \in \mathbb{N}$ and all $s \in \mathcal{S}$

- $i = 0$: clear
- $i > 0$:

$$
\begin{aligned}
v^{\pi_i}(s) &= \mathbb{E}^{\pi_i}[R_1 + \gamma R_2 + \cdots + \gamma^{i-1} R_i + \gamma^i R_{i+1} + \cdots \mid S_0 = s] \\
&= \mathbb{E}^{\pi_i}[R_1 + \gamma R_2 + \cdots + \gamma^{i-2} R_{i-1} \mid S_0 = s] + \mathbb{E}^{\pi_i}[\gamma^{i-1} R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] \\
&= \mathbb{E}^{\pi_{i-1}}[R_1 + \gamma R_2 + \cdots + \gamma^{i-2} R_{i-1} \mid S_0 = s] + \underbrace{\mathbb{E}^{\pi_i}[\gamma^{i-1} R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]}_{\text{Suppose we prove } \geq \mathbb{E}^{\pi_{i-1}}[\gamma^{i-1} R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]} \\
&\geq \mathbb{E}^{\pi_{i-1}}[R_1 + \gamma R_2 + \cdots + \gamma^{i-2} R_{i-1} + \gamma^{i-1} R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]
\end{aligned}
$$

Proof of PIT (induction)

$v^{\pi_i}(s) \geq v^{\pi}(s)$ for all $i \in \mathbb{N}$ and all $s \in \mathcal{S}$

- $i = 0$: clear
- $i > 0$:

$$
\begin{aligned}
v^{\pi_i}(s) &= \mathbb{E}^{\pi_i}[R_1 + \gamma R_2 + \cdots + \gamma^{i-1} R_i + \gamma^i R_{i+1} + \cdots \mid S_0 = s] \\
&= \mathbb{E}^{\pi_i}[R_1 + \gamma R_2 + \cdots + \gamma^{i-2} R_{i-1} \mid S_0 = s] + \mathbb{E}^{\pi_i}[\gamma^{i-1} R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] \\
&= \mathbb{E}^{\pi_{i-1}}[R_1 + \gamma R_2 + \cdots + \gamma^{i-2} R_{i-1} \mid S_0 = s] + \underbrace{\mathbb{E}^{\pi_i}[\gamma^{i-1} R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]}_{\text{Suppose we prove } \geq \mathbb{E}^{\pi_{i-1}}[\gamma^{i-1} R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]}
\end{aligned}
$$

$$
\begin{aligned}
&\geq \mathbb{E}^{\pi_{i-1}}[R_1 + \gamma R_2 + \cdots + \gamma^{i-2} R_{i-1} + \gamma^{i-1} R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] \\
&\overset{IH}{\geq} v^{\pi}(s)
\end{aligned}
$$

Proof of PIT (induction, behavior at "reset")

We need: $\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] \geq \mathbb{E}^{\pi_{i-1}}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]$.

Proof of PIT (induction, behavior at "reset")

We need: $\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] \geq \mathbb{E}^{\pi_{i-1}}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]$.

$\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]$

Proof of PIT (induction, behavior at "reset")

We need: $\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] \geq \mathbb{E}^{\pi_{i-1}}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]$.

$\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] = \gamma^{i-1} \cdot \mathbb{E}^{\pi_i}[R_i + \gamma R_{i+1} \cdots \mid S_0 = s]$

Proof of PIT (induction, behavior at "reset")

We need: $\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] \geq \mathbb{E}^{\pi_{i-1}}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]$.

$$\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] = \gamma^{i-1} \cdot \mathbb{E}^{\pi_i}[R_i + \gamma R_{i+1} \cdots \mid S_0 = s]$$
$$= \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_i}[S_{i-1} = s' \mid S_0 = s] \cdot \Big(r(s', \pi_i(s')) + \gamma \cdot \sum_{s''} p(s'' \mid s', \pi_i(s')) \cdot \mathbb{E}^{\pi_i}[G_i \mid S_i = s''] \Big)$$

Proof of PIT (induction, behavior at "reset")

We need: $\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] \geq \mathbb{E}^{\pi_{i-1}}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]$.

$$\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] = \gamma^{i-1} \cdot \mathbb{E}^{\pi_i}[R_i + \gamma R_{i+1} \cdots \mid S_0 = s]$$

$$= \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_i}[S_{i-1} = s' \mid S_0 = s] \cdot \Big(r(s', \pi_i(s')) + \gamma \cdot \sum_{s''} p(s'' \mid s', \pi_i(s')) \cdot \mathbb{E}^{\pi_i}[G_i \mid S_i = s'']\Big)$$

$$= \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_g}[S_{i-1} = s' \mid S_0 = s] \cdot \Big(r(s', \pi_g(s')) + \gamma \cdot \sum_{s''} p(s'' \mid s', \pi_g(s')) \cdot \mathbb{E}^{\pi}[G_i \mid S_i = s'']\Big)$$

Proof of PIT (induction, behavior at "reset")

We need: $\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] \geq \mathbb{E}^{\pi_{i-1}}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]$.

$$\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] = \gamma^{i-1} \cdot \mathbb{E}^{\pi_i}[R_i + \gamma R_{i+1} \cdots \mid S_0 = s]$$

$$= \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_i}[S_{i-1} = s' \mid S_0 = s] \cdot \Big(r(s', \pi_i(s')) + \gamma \cdot \sum_{s''} p(s'' \mid s', \pi_i(s')) \cdot \mathbb{E}^{\pi_i}[G_i \mid S_i = s''] \Big)$$

$$= \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_g}[S_{i-1} = s' \mid S_0 = s] \cdot \Big(r(s', \pi_g(s')) + \gamma \cdot \sum_{s''} p(s'' \mid s', \pi_g(s')) \cdot \underbrace{\mathbb{E}^{\pi}[G_i \mid S_i = s'']}_{v^{\pi}(s'')} \Big)$$

Proof of PIT (induction, behavior at "reset")

We need: $\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] \geq \mathbb{E}^{\pi_{i-1}}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]$.

$$\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] = \gamma^{i-1} \cdot \mathbb{E}^{\pi_i}[R_i + \gamma R_{i+1} \cdots \mid S_0 = s]$$

$$= \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_i}[S_{i-1} = s' \mid S_0 = s] \cdot \Big(r(s', \pi_i(s')) + \gamma \cdot \sum_{s''} p(s'' \mid s', \pi_i(s')) \cdot \mathbb{E}^{\pi_i}[G_i \mid S_i = s''] \Big)$$

$$= \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_g}[S_{i-1} = s' \mid S_0 = s] \cdot \underbrace{\Big(r(s', \pi_g(s')) + \gamma \cdot \sum_{s''} p(s'' \mid s', \pi_g(s')) \cdot \underbrace{\mathbb{E}^{\pi}[G_i \mid S_i = s'']}_{v^{\pi}(s'')} \Big)}_{= \Phi(v^{\pi}), \text{ since } \pi_g \text{ is } v^{\pi}\text{-greedy}}$$

Proof of PIT (induction, behavior at "reset")

We need: $\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] \geq \mathbb{E}^{\pi_{i-1}}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]$.

$$\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] = \gamma^{i-1} \cdot \mathbb{E}^{\pi_i}[R_i + \gamma R_{i+1} \cdots \mid S_0 = s]$$

$$= \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_i}[S_{i-1} = s' \mid S_0 = s] \cdot \Big(r(s', \pi_i(s')) + \gamma \cdot \sum_{s''} p(s'' \mid s', \pi_i(s')) \cdot \mathbb{E}^{\pi_i}[G_i \mid S_i = s''] \Big)$$

$$= \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_g}[S_{i-1} = s' \mid S_0 = s] \cdot \underbrace{\underbrace{\Big(r(s', \pi_g(s')) + \gamma \cdot \sum_{s''} p(s'' \mid s', \pi_g(s')) \cdot \underbrace{\mathbb{E}^{\pi}[G_i \mid S_i = s'']}_{v^\pi(s'')} \Big)}_{= \Phi(v^\pi)(s'), \text{ since } \pi_g \text{ is } v^\pi\text{-greedy}}}_{\geq v^\pi(s'), \text{ since } \Phi(v^\pi) \geq v^\pi \text{ by PIT assumption.}}$$

Proof of PIT (induction, behavior at "reset")

We need: $\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] \geq \mathbb{E}^{\pi_{i-1}}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]$.

$$\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] = \gamma^{i-1} \cdot \mathbb{E}^{\pi_i}[R_i + \gamma R_{i+1} \cdots \mid S_0 = s]$$

$$= \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_i}[S_{i-1} = s' \mid S_0 = s] \cdot \Big(r(s', \pi_i(s')) + \gamma \cdot \sum_{s''} p(s'' \mid s', \pi_i(s')) \cdot \mathbb{E}^{\pi_i}[G_i \mid S_i = s''] \Big)$$

$$= \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_g}[S_{i-1} = s' \mid S_0 = s] \cdot \underbrace{\underbrace{\Big(r(s', \pi_g(s')) + \gamma \cdot \sum_{s''} p(s'' \mid s', \pi_g(s')) \cdot \underbrace{\mathbb{E}^{\pi}[G_i \mid S_i = s'']}_{v^{\pi}(s'')} \Big)}_{= \Phi(v^{\pi})(s'), \text{ since } \pi_g \text{ is } v^{\pi}\text{-greedy}}}_{\geq v^{\pi}(s'), \text{ since } \Phi(v^{\pi}) \geq v^{\pi} \text{ by PIT assumption.}}$$

$$\geq \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_g}[S_{i-1} = s' \mid S_0 = s] \cdot v^{\pi}(s')$$

Proof of PIT (induction, behavior at "reset")

We need: $\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] \geq \mathbb{E}^{\pi_{i-1}}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]$.

$$\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] = \gamma^{i-1} \cdot \mathbb{E}^{\pi_i}[R_i + \gamma R_{i+1} \cdots \mid S_0 = s]$$

$$= \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_i}[S_{i-1} = s' \mid S_0 = s] \cdot \Big(r(s', \pi_i(s')) + \gamma \cdot \sum_{s''} p(s'' \mid s', \pi_i(s')) \cdot \mathbb{E}^{\pi_i}[G_i \mid S_i = s''] \Big)$$

$$= \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_g}[S_{i-1} = s' \mid S_0 = s] \cdot \underbrace{\underbrace{\Big(r(s', \pi_g(s')) + \gamma \cdot \sum_{s''} p(s'' \mid s', \pi_g(s')) \cdot \underbrace{\mathbb{E}^{\pi}[G_i \mid S_i = s'']}_{v^{\pi}(s'')} \Big)}_{= \Phi(v^{\pi})(s'), \text{ since } \pi_g \text{ is } v^{\pi}\text{-greedy}}}_{\geq v^{\pi}(s'), \text{ since } \Phi(v^{\pi}) \geq v^{\pi} \text{ by PIT assumption.}}$$

$$\geq \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_g}[S_{i-1} = s' \mid S_0 = s] \cdot v^{\pi}(s') = \gamma^{i-1} \cdot \mathbb{E}^{\pi_g}[G_{i-1} \mid S_0 = s] = \mathbb{E}^{\pi_g}[\gamma^{i-1}G_{i-1} \mid S_0 = s]$$

Proof of PIT (induction, behavior at "reset")

We need: $\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] \geq \mathbb{E}^{\pi_{i-1}}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]$.

$$\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] = \gamma^{i-1} \cdot \mathbb{E}^{\pi_i}[R_i + \gamma R_{i+1} \cdots \mid S_0 = s]$$

$$= \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_i}[S_{i-1} = s' \mid S_0 = s] \cdot \Big(r(s', \pi_i(s')) + \gamma \cdot \sum_{s''} p(s'' \mid s', \pi_i(s')) \cdot \mathbb{E}^{\pi_i}[G_i \mid S_i = s''] \Big)$$

$$= \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_g}[S_{i-1} = s' \mid S_0 = s] \cdot \underbrace{\underbrace{\Big(r(s', \pi_g(s')) + \gamma \cdot \sum_{s''} p(s'' \mid s', \pi_g(s')) \cdot \underbrace{\mathbb{E}^{\pi}[G_i \mid S_i = s'']}_{v^{\pi}(s'')} \Big)}_{= \Phi(v^{\pi})(s'), \text{ since } \pi_g \text{ is } v^{\pi}\text{-greedy}}}_{\geq v^{\pi}(s'), \text{ since } \Phi(v^{\pi}) \geq v^{\pi} \text{ by PIT assumption.}}$$

$$\geq \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_g}[S_{i-1} = s' \mid S_0 = s] \cdot v^{\pi}(s') = \gamma^{i-1} \cdot \mathbb{E}^{\pi_g}[G_{i-1} \mid S_0 = s] = \mathbb{E}^{\pi_g}[\gamma^{i-1}G_{i-1} \mid S_0 = s]$$

$$= \mathbb{E}^{\pi_g}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]$$

Proof of PIT (induction, behavior at "reset")

We need: $\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] \geq \mathbb{E}^{\pi_{i-1}}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]$.

$$\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] = \gamma^{i-1} \cdot \mathbb{E}^{\pi_i}[R_i + \gamma R_{i+1} \cdots \mid S_0 = s]$$

$$= \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_i}[S_{i-1} = s' \mid S_0 = s] \cdot \Big(r(s', \pi_i(s')) + \gamma \cdot \sum_{s''} p(s'' \mid s', \pi_i(s')) \cdot \mathbb{E}^{\pi_i}[G_i \mid S_i = s''] \Big)$$

$$= \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_g}[S_{i-1} = s' \mid S_0 = s] \cdot \underbrace{\underbrace{\Big(r(s', \pi_g(s')) + \gamma \cdot \sum_{s''} p(s'' \mid s', \pi_g(s')) \cdot \underbrace{\mathbb{E}^{\pi}[G_i \mid S_i = s'']}_{v^{\pi}(s'')} \Big)}_{= \Phi(v^{\pi})(s'), \text{ since } \pi_g \text{ is } v^{\pi}\text{-greedy}}}_{\geq v^{\pi}(s'), \text{ since } \Phi(v^{\pi}) \geq v^{\pi} \text{ by PIT assumption.}}$$

$$\geq \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_g}[S_{i-1} = s' \mid S_0 = s] \cdot v^{\pi}(s') = \gamma^{i-1} \cdot \mathbb{E}^{\pi_g}[G_{i-1} \mid S_0 = s] = \mathbb{E}^{\pi_g}[\gamma^{i-1}G_{i-1} \mid S_0 = s]$$

$$= \mathbb{E}^{\pi_g}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] = \mathbb{E}^{\pi_{i-1}}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]$$

Proof of PIT (induction, behavior at "reset")

We need: $\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] \geq \mathbb{E}^{\pi_{i-1}}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s]$.

$$\mathbb{E}^{\pi_i}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] = \gamma^{i-1} \cdot \mathbb{E}^{\pi_i}[R_i + \gamma R_{i+1} \cdots \mid S_0 = s]$$

$$= \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_i}[S_{i-1} = s' \mid S_0 = s] \cdot \Big(r(s', \pi_i(s')) + \gamma \cdot \sum_{s''} p(s'' \mid s', \pi_i(s')) \cdot \mathbb{E}^{\pi_i}[G_i \mid S_i = s''] \Big)$$

$$= \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_g}[S_{i-1} = s' \mid S_0 = s] \cdot \underbrace{\underbrace{\Big(r(s', \pi_g(s')) + \gamma \cdot \sum_{s''} p(s'' \mid s', \pi_g(s')) \cdot \underbrace{\mathbb{E}^{\pi}[G_i \mid S_i = s'']}_{v^\pi(s'')} \Big)}_{= \Phi(v^\pi)(s'), \text{ since } \pi_g \text{ is } v^\pi\text{-greedy}}}_{\geq v^\pi(s'), \text{ since } \Phi(v^\pi) \geq v^\pi \text{ by PIT assumption.}}$$

$$\geq \gamma^{i-1} \cdot \sum_{s' \in \mathcal{S}} \mathbb{P}^{\pi_g}[S_{i-1} = s' \mid S_0 = s] \cdot v^\pi(s') = \gamma^{i-1} \cdot \mathbb{E}^{\pi_g}[G_{i-1} \mid S_0 = s] = \mathbb{E}^{\pi_g}[\gamma^{i-1} G_{i-1} \mid S_0 = s]$$

$$= \mathbb{E}^{\pi_g}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] = \mathbb{E}^{\pi_{i-1}}[\gamma^{i-1}R_i + \gamma^i R_{i+1} \cdots \mid S_0 = s] \quad \text{Strictness?}$$

Policy iteration (PI; Howard, 1960)

Algorithm 2: Policy iteration

Input: MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$

Output: Optimal MD policy π^* for $\mathcal{M}$, its value vector v^*

$\pi \leftarrow$ arbitrary MD policy ;

$v \leftarrow v^\pi$; `// e.g. by solving linear policy evaluation equations`

while $\Phi(v) \neq v$ **do**

  foreach $s \in \mathcal{S}$ **do**

    $\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}(s)} [r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s'|s,a) \cdot v(s')]$

  $v \leftarrow v^\pi$

return π, v

PI correctness & complexity

Theorem 28

Policy iteration terminates after at most exponentially many iterations. Upon termination, it returns an optimal MD policy.

PI correctness & complexity

Theorem 28

Policy iteration terminates after at most exponentially many iterations. Upon termination, it returns an optimal MD policy.

Proof:

- Optimal upon termination: v^π is a fixpoint of Φ when terminating: optimality follows from Corollary 19.

PI correctness & complexity

Theorem 28

Policy iteration terminates after at most exponentially many iterations. Upon termination, it returns an optimal MD policy.

Proof:

- Optimal upon termination: v^π is a fixpoint of Φ when terminating: optimality follows from Corollary 19.
- Terminates: π always stores an MD policy and there are finitely many of these. We will show that no single MD policy appears in more than one iteration of PI.

Theorem 28

Policy iteration terminates after at most exponentially many iterations. Upon termination, it returns an optimal MD policy.

Proof:

- Optimal upon termination: v^π is a fixpoint of Φ when terminating: optimality follows from Corollary 19.
- Terminates: π always stores an MD policy and there are finitely many of these. We will show that no single MD policy appears in more than one iteration of PI.
  Consider any iteration and let v, v' be the contents of variable v before and after the iteration. We will show that unless $\Phi(v) = v$, it holds $v' > v$, i.e. $v' \geq v$ componentwise with strict inequality in some component. Hence, $v = v^\pi$ strictly increases during PI, so no π can appear twice.

PI: correctness proof

$v' \geq v$:

PI: correctness proof

$v' \geq v$:

We verify assumptions of PIT: $\Phi(v) \geq v$. Recall $v = v^\pi$. For all $s \in \mathcal{S}$:

$$\Phi(v)(s) = \max_{a \in \mathcal{A}(s)} [r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s,a) \cdot v(s')]$$

By PIT, $v' = v^{\pi'} \geq v^\pi = v$ (here π' is the v-greedy policy).

PI: correctness proof

$v' \geq v$:

We verify assumptions of PIT: $\Phi(v) \geq v$. Recall $v = v^\pi$. For all $s \in \mathcal{S}$:

$$\begin{aligned}\Phi(v)(s) &= \max_{a \in \mathcal{A}(s)} [r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s,a) \cdot v(s')] \\ &\geq r(s,\pi(s)) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s,\pi(s)) \cdot v^\pi(s')\end{aligned}$$

By PIT, $v' = v^{\pi'} \geq v^\pi = v$ (here π' is the v-greedy policy).

PI: correctness proof

$v' \geq v$:

We verify assumptions of PIT: $\Phi(v) \geq v$. Recall $v = v^\pi$. For all $s \in \mathcal{S}$:

$$\begin{aligned}
\Phi(v)(s) &= \max_{a \in \mathcal{A}(s)} [r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s,a) \cdot v(s')] \\
&\geq r(s, \pi(s)) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, \pi(s)) \cdot v^\pi(s') \\
&= v^\pi(s) = v(s).
\end{aligned}$$

By PIT, $v' = v^{\pi'} \geq v^\pi = v$ (here π' is the v-greedy policy).

PI: correctness proof II

It remains to prove that $v' > v$ or PI terminates. Assume that $v' = v$. Then for all $s \in \mathcal{S}$:

,

PI: correctness proof II

It remains to prove that $v' > v$ or PI terminates. Assume that $v' = v$. Then for all $s \in \mathcal{S}$:

$$v'(s) = r(s, \pi'(s)) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, \pi'(s)) \cdot v^{\pi'}(s')$$

,

PI: correctness proof II

It remains to prove that $v' > v$ or PI terminates. Assume that $v' = v$. Then for all $s \in \mathcal{S}$:

$$
\begin{aligned}
v'(s) &= r(s, \pi'(s)) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, \pi'(s)) \cdot v^{\pi'}(s') \\
&= r(s, \pi'(s)) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, \pi'(s)) \cdot v^{\pi}(s') \quad \text{(assumption)}
\end{aligned}
$$

,

PI: correctness proof II

It remains to prove that $v' > v$ or PI terminates. Assume that $v' = v$. Then for all $s \in \mathcal{S}$:

$$
\begin{aligned}
v'(s) &= r(s, \pi'(s)) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, \pi'(s)) \cdot v^{\pi'}(s') \\
&= r(s, \pi'(s)) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, \pi'(s)) \cdot v^{\pi}(s') \quad \text{(assumption)} \\
&= \max_{a \in \mathcal{A}(s)} [r(s, a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, a) \cdot v^{\pi}(s')] \quad (\pi' \text{ is } v = v^{\pi}\text{-greedy})
\end{aligned}
$$

,

PI: correctness proof II

It remains to prove that $v' > v$ or PI terminates. Assume that $v' = v$. Then for all $s \in \mathcal{S}$:

$$
\begin{aligned}
v'(s) &= r(s, \pi'(s)) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, \pi'(s)) \cdot v^{\pi'}(s') \\
&= r(s, \pi'(s)) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, \pi'(s)) \cdot v^{\pi}(s') \quad \text{(assumption)} \\
&= \max_{a \in \mathcal{A}(s)} [r(s, a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, a) \cdot v^{\pi}(s')] \quad (\pi' \text{ is } v = v^{\pi}\text{-greedy}) \\
&= \max_{a \in \mathcal{A}(s)} [r(s, a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, a) \cdot v^{\pi'}(s')] \quad \text{(assumption)}
\end{aligned}
$$

,

PI: correctness proof II

It remains to prove that $v' > v$ or PI terminates. Assume that $v' = v$. Then for all $s \in \mathcal{S}$:

$$
\begin{aligned}
v'(s) &= r(s, \pi'(s)) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, \pi'(s)) \cdot v^{\pi'}(s') \\
&= r(s, \pi'(s)) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, \pi'(s)) \cdot v^{\pi}(s') \quad \text{(assumption)} \\
&= \max_{a \in \mathcal{A}(s)} [r(s, a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, a) \cdot v^{\pi}(s')] \quad (\pi' \text{ is } v = v^{\pi}\text{-greedy}) \\
&= \max_{a \in \mathcal{A}(s)} [r(s, a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, a) \cdot v^{\pi'}(s')] \quad \text{(assumption)} \\
&= \Phi(v')(s),
\end{aligned}
$$

so PI terminates at this point.

PI: correctness proof II

It remains to prove that $v' > v$ or PI terminates. Assume that $v' = v$. Then for all $s \in \mathcal{S}$:

$$
\begin{aligned}
v'(s) &= r(s, \pi'(s)) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, \pi'(s)) \cdot v^{\pi'}(s') \\
&= r(s, \pi'(s)) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, \pi'(s)) \cdot v^{\pi}(s') \quad \text{(assumption)} \\
&= \max_{a \in \mathcal{A}(s)} [r(s, a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, a) \cdot v^{\pi}(s')] \quad (\pi' \text{ is } v = v^{\pi}\text{-greedy}) \\
&= \max_{a \in \mathcal{A}(s)} [r(s, a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, a) \cdot v^{\pi'}(s')] \quad \text{(assumption)} \\
&= \Phi(v')(s),
\end{aligned}
$$

so PI terminates at this point. Complexity?

PI&VI: discussion

- We know that MDPs have a linear programming (LP) formulation. PI is basically a variant of a simplex method for this LP, using a special pivoting rule.
- PI typicaly requires less iterations to converge than VI, though each iteration is more expensive (policy eval.)
- Both PI and VI typically work well in practice for MDPs whose explicit transition table fits inside a computer. Which of the two is faster is rather domain-specific.

PI variants

Can we get rid of the expensive policy evaluation by linear system solving?

PI variants

Can we get rid of the expensive policy evaluation by linear system solving?

Yes: we can approximate the value of the current policy π by applying VI on the MDP $\mathcal{M}^\pi$, for either fixed number of steps or until v does not change much. Often appearing in RL textbooks:

Policy iteration with approximate evaluation

$\pi \leftarrow$ arbitrary MD policy; $v \leftarrow$ arbitrary vector;

repeat

  $v \leftarrow \texttt{Eval}(\pi, v)$;

  foreach $s \in \mathcal{S}$ **do**

    $\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}(s)} [r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s'|s,a) \cdot v(s')]$

until π *has not changed*;

return π, v

Function $\texttt{Eval}(\pi, v)$**:**

  $v' \leftarrow v$;

  repeat

    foreach $s \in \mathcal{S}$ **do**

      $v(s) \leftarrow v'(s)$;

      $v'(s) \leftarrow r(s, \pi(s)) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s' \mid s, \pi'(s)) \cdot v(s')$

  until $\|v - v'\|_\infty \leq \varepsilon$;

return v'

Convergence of PI variants

- The algorithm on previous slide still converges to an optimal policy provided that ε is small enough.
- If we replaced the "π not changed condition" with the original "$\Phi(v) = v$" condition, the algorithm might not terminate, since the VI is only guaranteed to reach a true fixpoint in the limit. However, v would still converge to v^* and thus π would eventually become equal to an optimal policy.
- The previous point holds even in the very degenerate case when we do just one iteration of VI per policy evaluation! See next slide.

Curiously looking approximate PI

$v \leftarrow$ arbitrary vector;

$\pi \leftarrow v$-greedy MD policy ;

repeat

  foreach $s \in \mathcal{S}$ **do**

    $v'(s) \leftarrow r(s, \pi(s)) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s'|s, \pi(s)) \cdot v(s')$;

  $v \leftarrow v'$;

  foreach $s \in \mathcal{S}$ **do**

    $\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}(s)} [r(s, a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s'|s, a) \cdot v(s')]$

until $\Phi(v) = v$;

return π, v

Curiously looking approximate PI

$v \leftarrow$ arbitrary vector;

$\pi \leftarrow v$-greedy MD policy ;

repeat

  foreach $s \in \mathcal{S}$ **do**

    $v'(s) \leftarrow r(s, \pi(s)) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s'|s, \pi(s)) \cdot v(s')$;

  $v \leftarrow v'$;

  foreach $s \in \mathcal{S}$ **do**

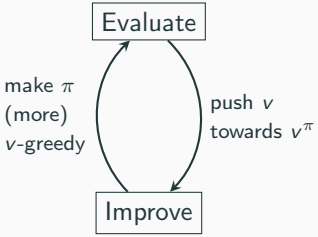
    $\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}(s)} [r(s, a) + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s'|s, a) \cdot v(s')]$

until $\Phi(v) = v$;

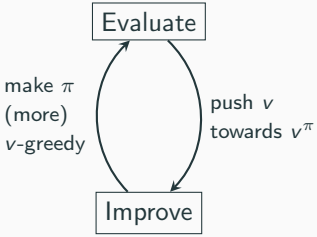
return π, v

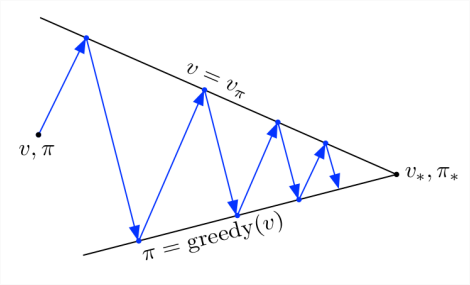
This is just VI in disguise!

Generalized policy iteration



Generalized policy iteration





Source: Sutton&Barto, p. 87

Tabular Methods for Model-Free Reinforcement Learning

Model-free

Model-free

We will still be working with MDPs. But for a bunch of the following lectures, we will not (necessarily) have access to, e.g.:

- a table containing explicit enumeration of all states/actions
- a table containing the description of p or r
- the ability to compute the probability vector $\delta(s, a)$ or the reward signal $r(s, a)$ given s and a (having this = gray-box model of the MDP)

Sampling from MDP

But the MDP is still there "behind the scene". In particular, we:

Sampling from MDP

But the MDP is still there "behind the scene". In particular, we:

- know how the states of the MDP look like
  - (e.g. robot state = all possible output values of its sensors)

Sampling from MDP

But the MDP is still there "behind the scene". In particular, we:

- know how the states of the MDP look like
  - (e.g. robot state = all possible output values of its sensors)
- know how the actions of the MDP look like
  - (e.g. robot = all possible signals that can be sent to the actuators)

Sampling from MDP

But the MDP is still there "behind the scene". In particular, we:

- know how the states of the MDP look like
  - (e.g. robot state = all possible output values of its sensors)
- know how the actions of the MDP look like
  - (e.g. robot = all possible signals that can be sent to the actuators)
- can, for any $s \in \mathcal{S}$, enumerate $\mathcal{A}(s)$
  - could be weakened, but simplifies things

Sampling from MDP

But the MDP is still there "behind the scene". In particular, we:

- know how the states of the MDP look like
  - (e.g. robot state = all possible output values of its sensors)
- know how the actions of the MDP look like
  - (e.g. robot = all possible signals that can be sent to the actuators)
- can, for any $s \in \mathcal{S}$, enumerate $\mathcal{A}(s)$
  - could be weakened, but simplifies things
- given $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$, we can sample the next state $s' \sim p(s, a)$ and receive the reward $r(s, a)$.

Sampling from a policy

Given an effective representation of a policy π, we can sample a trajectory $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, \ldots$ by performing, for each $t \in \{0, \ldots, T\}$:

- sample $a_t \sim \pi(s_t)$
- query the environment for $s_{t+1} \sim p(s_t, a_t)$ and $r_{t+1} = r(s_t, a_t)$
- increment t

Sampling from a policy

Given an effective representation of a policy π, we can sample a trajectory $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, \ldots$ by performing, for each $t \in \{0, \ldots, T\}$:

- sample $a_t \sim \pi(s_t)$
- query the environment for $s_{t+1} \sim p(s_t, a_t)$ and $r_{t+1} = r(s_t, a_t)$
- increment t

Tabular = value estimates and policies represented as tables (e.g. $Q(s, a)$ for each state s and action a used in s – explicit representation might only be needed for states/actions actually encountered).

Basic classification of (tabular) RL algorithms

Three independent axes:

problem	on/off	updates
policy evaluation (value prediction)	on-policy	Monte Carlo
vs.	vs.	
control	off-policy	temporal difference

Assumptions: successor-dependent rewards & episodic tasks

Since we do no longer have the knowledge of the transition dynamics p, we cannot freely interchange MDPs with rewards functions of type $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ and $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$ via the equation $r(s, a) = \sum_{s' \in \mathcal{S}} p(s' \mid s, a) \cdot r(s, a, s')$.
Hence, to maintain generality (and correspondence to e.g. Gymnasium environments) we will assume reward functions of type $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$.

We will assume episodic returns: each trajectory terminates with probability 1 at some (possibly random) time step T. Termination can be defined e.g. by reaching some terminal state or by running out of some fixed decision horizon (in Gymnasium, this is sometimes called truncation):

$$G = \sum_{i=0}^{T-1} \gamma^i \cdot R_{i+1}.$$

Episode = one high-level iteration of an RL algorithm, corresponding of sampling a single trajectory from some policy.

Monte Carlo Methods

MC evaluation

Policy evaluation: given an effective representation of a policy π, estimate v^π (or q^π).

MC evaluation

Policy evaluation: given an effective representation of a policy π, estimate v^π (or q^π).

Naive Monte Carlo: Sample from π: if $\{\tau_1, \tau_2, \ldots, \tau_n\}$ are trajectories (episodes) independently sampled under π from the same initial state s, then $\frac{1}{n}\sum_{i=1}^{n} G(\tau_i) \to v^\pi(s)$ as $n \to \infty$ due to law of large numbers (LLN).

MC evaluation

Policy evaluation: given an effective representation of a policy π, estimate v^π (or q^π).

Naive Monte Carlo: Sample from π: if $\{\tau_1, \tau_2, \ldots, \tau_n\}$ are trajectories (episodes) independently sampled under π from the same initial state s, then $\frac{1}{n}\sum_{i=1}^{n} G(\tau_i) \to v^\pi(s)$ as $n \to \infty$ due to law of large numbers (LLN).

But this throws away a lot of valuable information! E.g. what if we want to estimate the whole v^π?

First-visit MC

For each s, we estimate $v^\pi(s)$ as an average of sample returns $Ret(s)$ which is formed as follows:

- initially, $Ret(s) = \emptyset$ for all s
- we then sample trajectories until timeout:
  - for each sampled trajectory τ and each state s, we identify the first occurrence of s on τ: let this be at timestep t; we add $G_t(\tau)$ to $Ret(s)$

First-visit MC

For each s, we estimate $v^\pi(s)$ as an average of sample returns $Ret(s)$ which is formed as follows:

- initially, $Ret(s) = \emptyset$ for all s
- we then sample trajectories until timeout:
  - for each sampled trajectory τ and each state s, we identify the first occurrence of s on τ: let this be at timestep t; we add $G_t(\tau)$ to $Ret(s)$

Sub-trajectory starting at the first appearance of s can be seen as a trajectory sampled from π when s is the initial state! (Since we consider memoryless π.)

Theorem 29

As $|Ret(s)| \to \infty$, the average of $Ret(s)$ converges to $v^\pi(s)$. Moreover, the average of $Ret(s)$ is an unbiased estimate of $v^\pi(s)$ (as long as $Ret(s) \neq \emptyset$).

First-visit MC (pseudocode)

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

Append G to $Returns(S_t)$

$V(S_t) \leftarrow$ average$(Returns(S_t))$

Source: Sutton&Barto, p.92

Every-visit MC

For each s, we estimate $v^\pi(s)$ as an average of sample returns $Ret(s)$ which is formed as follows:

- initially, $Ret(s) = \emptyset$ for all s
- we then sample trajectories until timeout:
  - for each sampled trajectory τ and each state s, and each t such that $S_t(\tau) = s$ we add $G_t(\tau)$ to $Ret(s)$

Every-visit MC

For each s, we estimate $v^\pi(s)$ as an average of sample returns $Ret(s)$ which is formed as follows:

- initially, $Ret(s) = \emptyset$ for all s
- we then sample trajectories until timeout:
  - for each sampled trajectory τ and each state s, and each t such that $S_t(\tau) = s$ we add $G_t(\tau)$ to $Ret(s)$

The sample returns added to $Ret(s)$ within the same episode are not independent! Hence, the estimate is biased, though the bias vanishes in the limit:

Theorem 30

As $|Ret(s)| \to \infty$, the average of $Ret(s)$ converges to $v^\pi(s)$.

MC convergence

Optional reading: More on MC estimate bias, variance, and convergence in:

Singh, S.P. and Sutton, R.S.: Reinforcement Learning with Replacing Eligibility Traces. In *Machine Learning* 22:123–158. Kluwer, 1996.
(Section 3, particularly 3.3 and onwards, you can skip Theorem 4.)

Towards MC control

Control = computation of "good" policy for a given environment. (Ideally, the policy should get closer to the optimal policy the more episodes we sample.)

Towards MC control

Control = computation of "good" policy for a given environment. (Ideally, the policy should get closer to the optimal policy the more episodes we sample.)

We know (PIT): given a policy π a v^π-greedy policy is at least as good as π:

$$\pi_g(s) = \arg\max_{a \in \mathcal{A}(s)} \Big[\sum_{s' \in \mathcal{S}} p(s' \mid s, a) \cdot \big(r(s, a, s') + \gamma \cdot v^\pi(s') \big) \Big]$$

Do we have an algo? There is an issue:

MC control with q-values

Recall:

$$q^{\pi}(s,a) = \sum_{s' \in \mathcal{S}} p(s' \mid s,a) \cdot \overset{\text{def}}{\big(} r(s,a,s') + \gamma \cdot v^{\pi}(s') \,.\big)$$

Thus, the v^{π}-greedy policy π_g can be defined as:

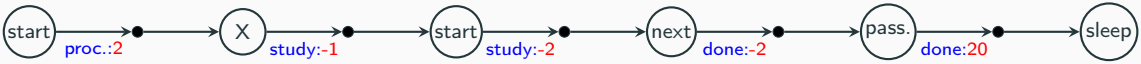
$$\pi_g(s) = \arg\max_{a \in \mathcal{A}(s)} \underbrace{q^{\pi}(s,a)}_{\text{Estimate by MC.}}$$

MC for q-value estimation

Analogous to value estimation, e.g. first-visit:

For each s, a, we estimate $q^\pi(s, a)$ as an average of sample returns $Ret(s, a)$ which is formed as follows:

- initially, $Ret(s, a) = \emptyset$ for all s
- we then sample trajectories until timeout:
  - for each sampled trajectory τ and each state-action pair (s, a), we identify the first t such that $S_t(\tau) = s \wedge A_t(\tau) = a$; we add $G_t(\tau)$ to $Ret(s)$



Similarly for every visit. Convergence guarantees the same as for state values.

Infinite exploration and exploring starts

Issue: MC only estimates $q^\pi(s, a)$ if:

- s guaranteed to be visited with positive probability in each episode
- $\pi(a \mid s) > 0$.

Infinite exploration and exploring starts

Issue: MC only estimates $q^\pi(s, a)$ if:

- s guaranteed to be visited with positive probability in each episode
- $\pi(a \mid s) > 0$.

Definition 31: Infinite exploration

A RL algorithm has infinite exploration (IE) if, during the infinite execution of the algorithm, each state-action pair (s, a) is visited infinitely often with probability 1.

Infinite exploration and exploring starts

Issue: MC only estimates $q^\pi(s, a)$ if:

- s guaranteed to be visited with positive probability in each episode
- $\pi(a \mid s) > 0$.

Definition 31: Infinite exploration

A RL algorithm has infinite exploration (IE) if, during the infinite execution of the algorithm, each state-action pair (s, a) is visited infinitely often with probability 1.

One way of achieving IE is through exploring starts (ES): each episode begins with (typically uniformly) randomly selected s_0 and a_0. This is achievable when training, e.g., in simulated environments but might be difficult/impossible in real-world environments.

MC control with exploring starts

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):

Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0

Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

Source: Sutton&Barto, p.99

IE through ε-soft policies

Exploring starts are not always feasible. Alternative: make the sampled policy itself exploratory.

Definition 32: ε-soft policy

A policy π is ε-soft if for every $s \in \mathcal{S}$ and every $a \in \mathcal{A}(s)$ it holds $\pi(a|s) \geq \frac{\varepsilon}{|\mathcal{A}(s)|}$.

IE through ε-soft policies

Exploring starts are not always feasible. Alternative: make the sampled policy itself exploratory.

Definition 32: ε-soft policy

A policy π is ε-soft if for every $s \in \mathcal{S}$ and every $a \in \mathcal{A}(s)$ it holds $\pi(a|s) \geq \frac{\varepsilon}{|\mathcal{A}(s)|}$.

Definition 33: ε-greedy policy

Let $v \in \mathbb{R}^{\mathcal{S}}$ be a value vector. A policy π is v-ε-greedy if for every state $s \in \mathcal{S}$ there is action $a^* = \arg\max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}} p(s' \mid s, a) \cdot \big(r(s, a, s') + \gamma \cdot v(s')\big)$ such that for any action $a \in \mathcal{A}(s)$ it holds:

$$\pi(a|s) = \begin{cases} \frac{\varepsilon}{\mathcal{A}(s)} & \text{if } a \neq a^* \\ 1 - \varepsilon + \frac{\varepsilon}{\mathcal{A}(s)} & \text{if } a = a^*. \end{cases}$$

Interpretation: with prob. ε: play uniformly at random; with prob. $1 - \varepsilon$: play greedily.

ε-softing

Definition 34

Let π be a policy. An ε-softing of π is a policy π_ε defined as follows: in each state s

- with probability ε, π_ε selects an action uniformly at random;
- with probability $1-\varepsilon$, π_ε selects $a \sim \pi(s)$.

I.e., an ε-greedy policy can be alternatively defined as ε-softing of a greedy policy.

MC control with ε-greedy policies

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ε-soft policy

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

source: Sutton&Barto, p. 101

Policy iteration for ε-soft policies

Theorem 35

Let π be an ε-soft policy and let π' be a v^{π}-ε-greedy policy. Than $v^{\pi'} \geq v^{\pi}$ (componentwise). Moreover, the two value vectors are equal if and only if bot π and π' are optimal among all ε-soft policies; i.e. if, for every state s:

$$v^{\pi}(s) = \sup_{\bar{\pi} \text{ that is } \varepsilon\text{-soft}} v^{\bar{\pi}}(s).$$

Proof: Required reading: Sutton&Barto, p.101-103.

Incremental computing of averages

Given a sample $\{n_1, n_2, \ldots, n_{k+1}\}$ and average $A = avg(\{n_1, n_2, \ldots, n_k\})$, how to compute $A' = avg(\{n_1, n_2, \ldots, n_k, n_{k+1}\})$ without recomputing the average of the whole sample?

$$A' =$$

Incremental computing of averages

Given a sample $\{n_1, n_2, \ldots, n_{k+1}\}$ and average $A = avg(\{n_1, n_2, \ldots, n_k\})$, how to compute $A' = avg(\{n_1, n_2, \ldots, n_k, n_{k+1}\})$ without recomputing the average of the whole sample?

$$A' = \frac{k}{k+1} \cdot A + \frac{n_{k+1}}{k+1}.$$

On-policy vs. off-policy

- On-policy algorithms: track one "policy variable" π; the policy stored in π is used to interact with the environment (i.e., to sample episodes) and at the same time we learn something about it (e.g. its value vector).
  - Corresponds to the generalized policy iteration scheme.
  - All the MC algos we have seen so far.

On-policy vs. off-policy

- On-policy algorithms: track one "policy variable" π; the policy stored in π is used to interact with the environment (i.e., to sample episodes) and at the same time we learn something about it (e.g. its value vector).
  - Corresponds to the generalized policy iteration scheme.
  - All the MC algos we have seen so far.
- Off-policy algorithms: track more (typically two) different policy variables:
  - behavior policy: used to sample episodes
  - target policy: which we want to learn about

Off-policy evaluation

We are given effective representations of:

- a behavior policy β,
- a target policy π.

The task is to estimate v^π by sampling episodes from β . We cannot sample from π! (E.g. π too risky or expensive to sample from.)

Off-policy evaluation

We are given effective representations of:

- a behavior policy β,
- a target policy π.

The task is to estimate v^π by sampling episodes from β . We cannot sample from π! (E.g. π too risky or expensive to sample from.)

Assumptions:

- given (s, a), we can effectively compute $\pi(a|s)$ and $\beta(a|s)$ (or at least estimate via sampling)
- coverage: $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$: if $\pi(a|s) > 0$, then also $\beta(a|s) > 0$

Definition 36: Importance ratio

Let $\tau = s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, \ldots$ be a trajectory. The importance-sampling ratio of τ is the quantity

$$\begin{aligned}\rho(\tau) &\overset{def}{=} \frac{\mathbb{P}^{\pi}[\tau \mid S_0 = s_0]}{\mathbb{P}^{\beta}[\tau \mid S_0 = s_0]} \\ &= \frac{\mathbb{P}^{\pi}[A_0 = a_0, S_1 = s_1, A_1 = a_1, \ldots, A_{T-1} = a_{T-1}, S_T = s_T \mid S_0 = s_0]}{\mathbb{P}^{\beta}[A_0 = a_0, S_1 = s_1, A_1 = a_1, \ldots, A_{T-1} = a_{T-1}, S_T = s_T \mid S_0 = s_0]}.\end{aligned}$$

Importance ration from time t

Definition 37

Let $\tau = s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, \ldots$ be a trajectory. By $\tau_{i..j}$ we denote the sub-trajectory of τ starting in time step i and ending in timestep j. By $\tau_{i..}$ we denote the suffix of $s_i, a_i, r_{i+1}, s_{i+1}, a_{i+1}, \ldots$.

Importance ration from time t

Definition 37

Let $\tau = s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, \ldots$ be a trajectory. By $\tau_{i..j}$ we denote the sub-trajectory of τ starting in time step i and ending in timestep j. By $\tau_{i..}$ we denote the suffix of $s_i, a_i, r_{i+1}, s_{i+1}, a_{i+1}, \ldots$.

Definition 38: Importance ratio from time t

Let $\tau = s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, \ldots$ be a trajectory and t a time step. The importance-sampling ratio of τ from t is the quantity

$$\begin{aligned}\rho_t(\tau) &\stackrel{def}{=} \frac{\mathbb{P}^{\pi}[\tau_{t..} \mid S_0 = s_t]}{\mathbb{P}^{\beta}[\tau_{t..} \mid S_0 = s_t]} \\ &= \frac{\mathbb{P}^{\pi}[A_0 = a_t, S_1 = s_{t+1}, A_1 = a_{t+1}, \ldots, A_{T-1-t} = a_{T-1}, S_{T-t} = s_T \mid S_0 = s_t]}{\mathbb{P}^{\beta}[A_0 = a_t, S_1 = s_{t+1}, A_1 = a_{t+1}, \ldots, A_{T-1-t} = a_{T-1}, S_{T-t} = s_T \mid S_0 = s_t]}.\end{aligned}$$

Off-policy evaluation with importance sampling

Theorem 39

For any $s \in \mathcal{S}$ it holds:

$$\mathbb{E}^{\beta}[\rho \cdot G \mid S_0 = s] = v^{\pi}(s).$$

Off-policy evaluation with importance sampling

Theorem 39

For any $s \in \mathcal{S}$ it holds:

$$\mathbb{E}^{\beta}[\rho \cdot G \mid S_0 = s] = v^{\pi}(s).$$

Proof:

$$\mathbb{E}^{\beta}[\rho \cdot G \mid S_0 = s] = \sum_{\tau} \mathbb{P}^{\beta}[\tau \mid S_0 = s] \cdot \rho(\tau) \cdot G(\tau)$$

Off-policy evaluation with importance sampling

Theorem 39

For any $s \in \mathcal{S}$ it holds:

$$\mathbb{E}^{\beta}[\rho \cdot G \mid S_0 = s] = v^{\pi}(s).$$

Proof:

$$\begin{aligned}\mathbb{E}^{\beta}[\rho \cdot G \mid S_0 = s] &= \sum_{\tau} \mathbb{P}^{\beta}[\tau \mid S_0 = s] \cdot \rho(\tau) \cdot G(\tau) \\ &= \sum_{\tau} \mathbb{P}^{\beta}[\tau \mid S_0 = s] \cdot \frac{\mathbb{P}^{\pi}[\tau \mid S_0 = s]}{\mathbb{P}^{\beta}[\tau \mid S_0 = s]} \cdot G(\tau)\end{aligned}$$

Off-policy evaluation with importance sampling

Theorem 39

For any $s \in \mathcal{S}$ it holds:

$$\mathbb{E}^{\beta}[\rho \cdot G \mid S_0 = s] = v^{\pi}(s).$$

Proof:

$$\begin{aligned}
\mathbb{E}^{\beta}[\rho \cdot G \mid S_0 = s] &= \sum_{\tau} \mathbb{P}^{\beta}[\tau \mid S_0 = s] \cdot \rho(\tau) \cdot G(\tau) \\
&= \sum_{\tau} \mathbb{P}^{\beta}[\tau \mid S_0 = s] \cdot \frac{\mathbb{P}^{\pi}[\tau \mid S_0 = s]}{\mathbb{P}^{\beta}[\tau \mid S_0 = s]} \cdot G(\tau) \\
&= \sum_{\tau} \mathbb{P}^{\pi}[\tau \mid S_0 = s] \cdot G(\tau) = \mathbb{E}^{\pi}[G \mid S_0 = s] = v^{\pi}(s).
\end{aligned}$$

.

Off-policy evaluation with importance sampling

Theorem 39

For any $s \in \mathcal{S}$ it holds:

$$\mathbb{E}^{\beta}[\rho \cdot G \mid S_0 = s] = v^{\pi}(s).$$

Proof:

$$\begin{aligned}
\mathbb{E}^{\beta}[\rho \cdot G \mid S_0 = s] &= \sum_{\tau} \mathbb{P}^{\beta}[\tau \mid S_0 = s] \cdot \rho(\tau) \cdot G(\tau) \\
&= \sum_{\tau} \mathbb{P}^{\beta}[\tau \mid S_0 = s] \cdot \frac{\mathbb{P}^{\pi}[\tau \mid S_0 = s]}{\mathbb{P}^{\beta}[\tau \mid S_0 = s]} \cdot G(\tau) \\
&= \sum_{\tau} \mathbb{P}^{\pi}[\tau \mid S_0 = s] \cdot G(\tau) = \mathbb{E}^{\pi}[G \mid S_0 = s] = v^{\pi}(s).
\end{aligned}$$

Easily integrates into both first-visit and every visit MC: sample from β and store $\rho_t(\tau) \cdot G_t(\tau)$ in $Ret(s_t)$.

Weighted importance sampling

First-visit variant: for each state s, we keep a set of samples $Sam(s)$. Each sample is a tuple (τ, t) – trajectory and time step.

- initially, $Sam(s) = \emptyset$ for all s
- we then sample trajectories until timeout:
  - for each sampled trajectory τ and each state s, and the smallest t such that $S_t(\tau) = s$ we add (τ, t) to $Sam(s)$

Throughout the algorithm, the value of state s is estimated as

$$WIS(s) = \frac{\sum_{(\tau,t)\in Sam(s)} \rho_t(\tau) \cdot G_t(\tau)}{\sum_{(\tau,t)\in Sam(s)} \rho_t(\tau)}$$

Weighted importance sampling

First-visit variant: for each state s, we keep a set of samples $Sam(s)$. Each sample is a tuple (τ, t) – trajectory and time step.

- initially, $Sam(s) = \emptyset$ for all s
- we then sample trajectories until timeout:
  - for each sampled trajectory τ and each state s, and the smallest t such that $S_t(\tau) = s$ we add (τ, t) to $Sam(s)$

Throughout the algorithm, the value of state s is estimated as

$$WIS(s) = \frac{\sum_{(\tau,t)\in Sam(s)} \rho_t(\tau) \cdot G_t(\tau)}{\sum_{(\tau,t)\in Sam(s)} \rho_t(\tau)}$$

Exercise 40

Compare ordinary/weighted importance sampling after single sample.

Weighted importance sampling – correctness

The weighted sampling is clearly a biased estimator. However, the bias vanishes in the limit:

Theorem 41

With probability 1: as $|Sam(s)| \to \infty$, we have that $WIS(s) \to v^{\pi}(s)$.

Proof:

Ordinary vs. weighted sampling

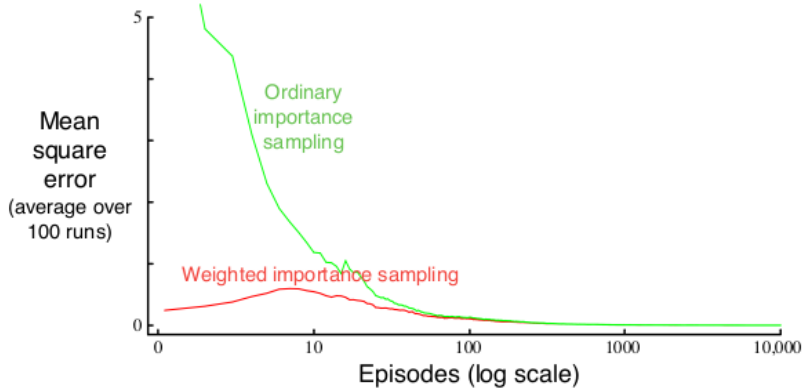


Figure 5.3: Weighted importance sampling produces lower error estimates of the value of a single blackjack state from off-policy episodes. ■

source: Sutton&Barto, p. 106

Importance sampling: summary

But ordinary and weighted importance sampling can be adapted to every-visit MC.

Bias & Convergence:

- First visit:
  - ordinary IS: unbiased, i.e. also converges
  - weighted IS: biased, but converges in the limit
- Every visit:
  - both ordinary and weighted: biased (due to EV), but converges in the limit

Weighted IS: incremental implementation

Instead of recomputing the weighted average for each new sample, $WIS(s)$ can be updated by keeping keep just two variables:

- V – current value of $WIS(s)$, initially arbitrary
- C – the sum of importance ratios, initially 0

Upon arrival of new sample (τ', t'), we update V, C into new values V', C' by setting:

$$C' = C + \rho_{t'}(\tau')$$
$$V' = V + \frac{\rho_{t'}(\tau')}{C'} \cdot (G_{t'}(\tau') - V)\,.$$

Off-policy evaluation with weighted IS

Off-policy MC prediction (policy evaluation) for estimating $Q \approx q_\pi$

Input: an arbitrary target policy π
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s,a) \in \mathbb{R}$ (arbitrarily)
    $C(s,a) \leftarrow 0$

Loop forever (for each episode):
    $b \leftarrow$ any policy with coverage of π
    Generate an episode following b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
    $W \leftarrow 1$
    Loop for each step of episode, $t = T-1, T-2, \ldots, 0$, while $W \neq 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
        $W \leftarrow W \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$

Off-policy control with weighted IS

Required reading: Sutton&Barto, Section 5.7.

Temporal Difference Methods

TD: Motivation

Let us first focus on policy evaluation. TBD

MC: zero bias (at least in the limit), but potentially high variance: many samples needed to converge. Also, to update estimates, it must wait till the end of each episode.

TD methods retain the focus on sampling but combine it with bootstrapping.

Incremental update notation

Definition 42: Notation for updates

In the context of RL algorithms will denote by $V^n(s)$ (resp. $Q^n(s,a)$) the algorithm's estimate of $v^\pi(s)$ (resp. $q^\pi(s,a)$) after n-th update of this estimate.

MC vs. TD(0) update

On-policy MC (incremental) update using sampled trajectory τ:

$$V^{n+1}(s_t) \leftarrow (1-\alpha_n)V^n(s_t) + \alpha_n G_t(\tau) = V^n(s_t) + \alpha_n \cdot \Big[\underbrace{G_t(\tau)}_{\text{update target}} - V^n(s_t) \Big]_{\text{update error}},$$

where $\alpha_n = n/(n+1)$.

MC vs. TD(0) update

On-policy MC (incremental) update using sampled trajectory τ:

$$V^{n+1}(s_t) \leftarrow (1-\alpha_n)V^n(s_t) + \alpha_n G_t(\tau) = V^n(s_t) + \alpha_n \cdot \big[\underbrace{G_t(\tau)}_{\text{update target}} - V^n(s_t)\big],$$

update error

where $\alpha_n = n/(n+1)$.

TD(0) update in the same situation, with α_n "suitably chosen" (possibly constant):

$$V^{n+1}(s_t) \leftarrow V^n(s_t) + \alpha_n \cdot \big[R_{t+1}(\tau) + \gamma \cdot \underbrace{V^n(S_{t+1}(\tau))}_{\text{bootstrap}} - V^n(s_t)\big]$$

Policy evaluation with TD(0)

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\mathit{terminal}) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
    $S \leftarrow S'$
  until S is terminal

source: Sutton&Barto, p. 120

How can it even work?

Really "just" a very asynchronous, sample-based, and "α-dampened" version of value iteration.

$$\mathbb{E}^\pi[G_t|S_t = s] = \mathbb{E}^\pi[R_{t+1} + \gamma \cdot G_{t+1} \mid S_t = s] = \mathbb{E}^\pi[R_{t+1} \mid S_t = s] + \gamma \cdot \underbrace{\mathbb{E}^\pi[G_{t+1} \mid S_t = s]}_{v^\pi(S_{t+1})}.$$

In expectation, the TD(0) update is the same as VI update in $\mathcal{M}^\pi$. Thanks to the contractivity of the Bellman operator, VI possesses an error reduction property: after each update, the error of the estimate decreases. Hence, in expectation, the same is true for the TD(0) update.

Formal proof of correctness in optional reading:

> Sutton, R.S.: Learning to Predict by Methods of Temporal Differences. In *Machine Learning* 3:9–44. Kluwer, 1988. (For MDPs with function approximation.)

Why TD is natural (Sutton&Barto, p. 122-123)

State	*Elapsed Time (minutes)*	*Predicted Time to Go*	*Predicted Total Time*
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

Why TD is natural (Sutton&Barto, p. 122-123)

State	*Elapsed Time (minutes)*	*Predicted Time to Go*	*Predicted Total Time*
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43