Applied Reinforcement Learning

Hao Shen

Institute for Data Processing
Technische Universität München

January 16, 2017

A question to you

What makes you decide to visit this very lecture?

- Less study load during semester
- Fond of the subject
- Like robotics

A question to us

What makes us decide to offer this very lecture?

- We have to teach
- We like the subject
- And more, we believe in the future

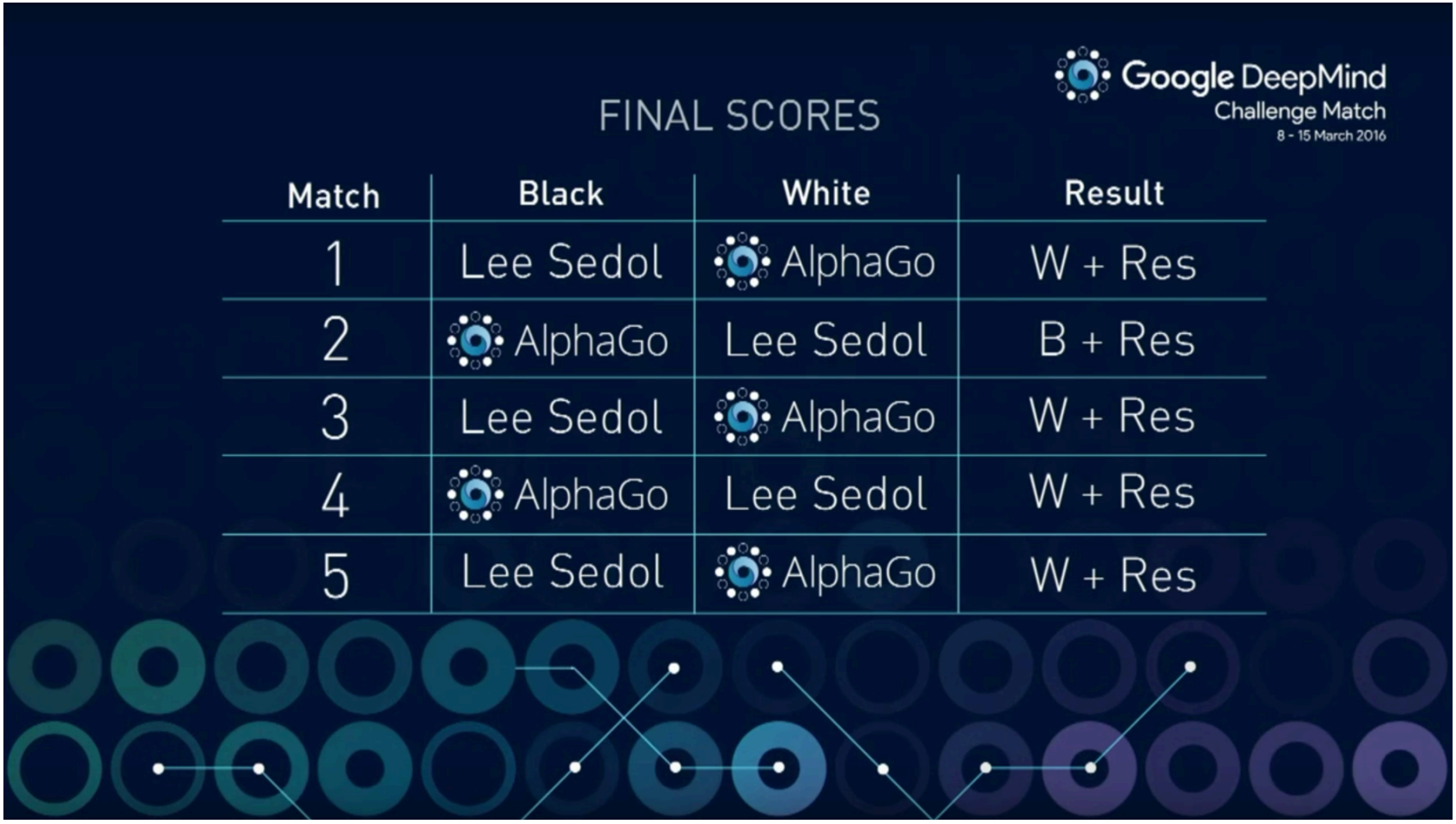
Autonomous cars

Smart cars

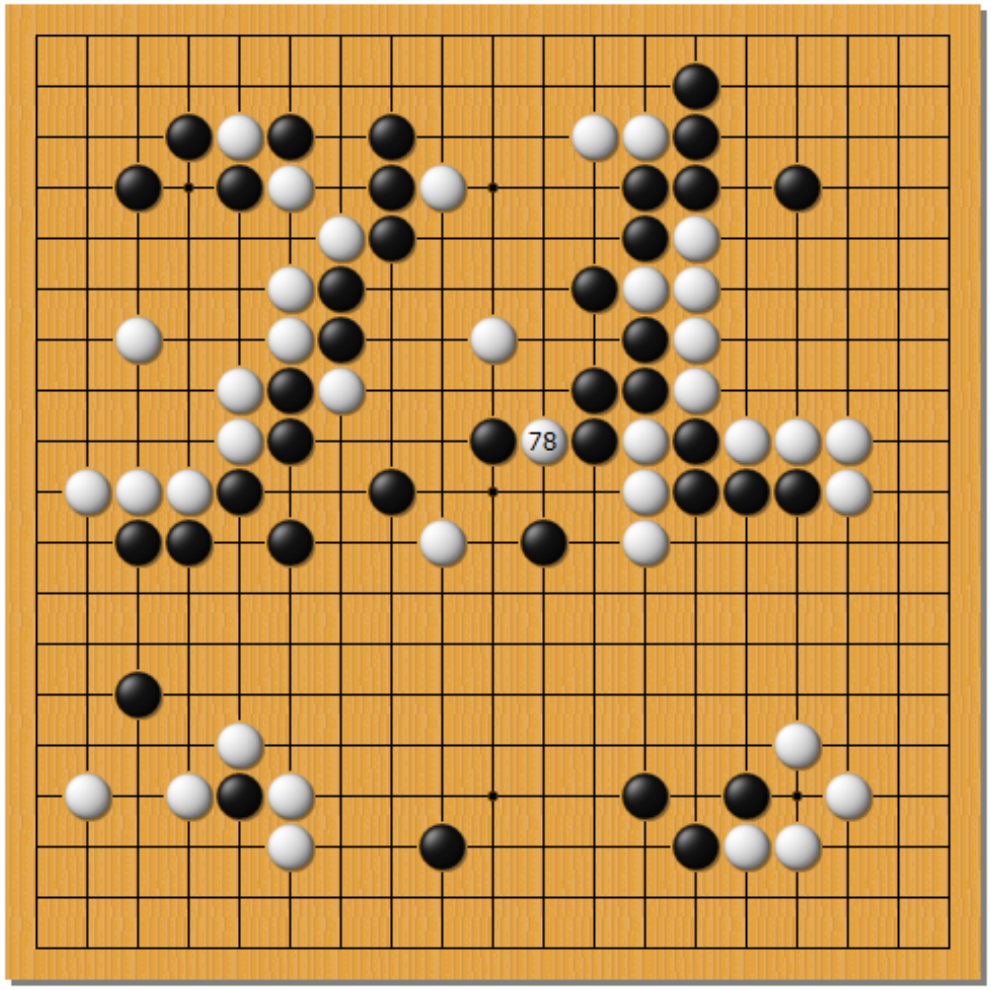

2016 MERCEDES E-CLASS
ACTIVE BRAKING ASSIST
PEDESTRIAN DETECTION
Gommeblog.it

AlphaGo vs. Lee Sedol: March 9-15, 2016

AlphaGo vs. Lee Sedol: final score



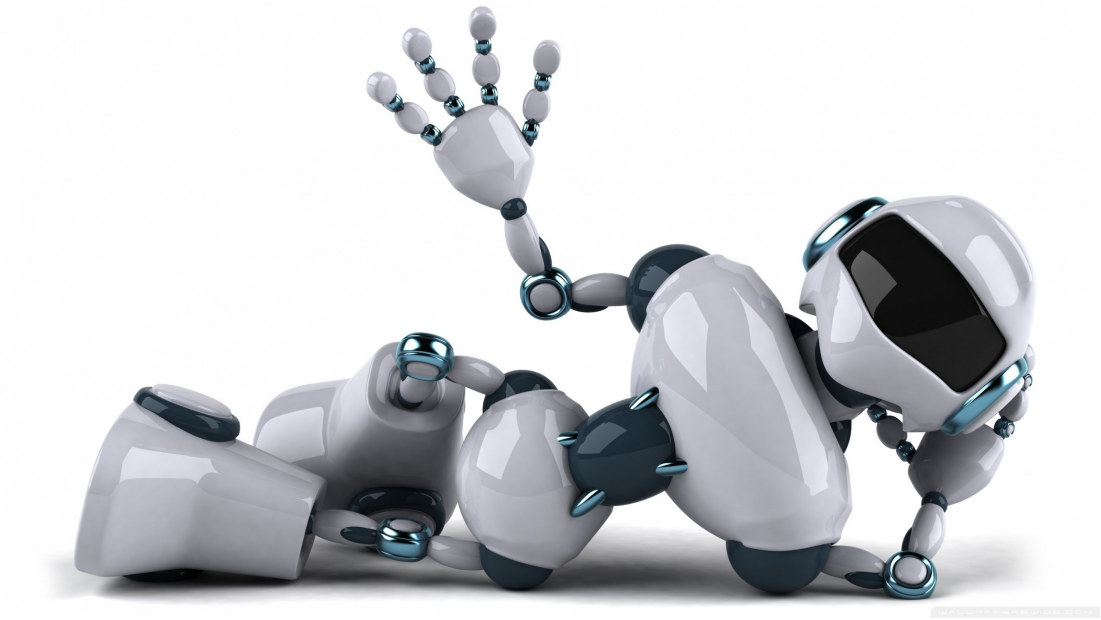
AlphaGo (black) vs. Lee Sedol (white): March 13, 2016



Future: version one

Future: version two

Future technology



Future technology might/should perform intelligent behavior.

Some facts

Key technical characteristics

- Adequate autonomy in learning, decision making
- Interactive temporal structure
- Strong uncertainties
- etc.

Learn from the nature!

What do we do?

What is reinforcement?

[Oxford Dictionaries]

The action or process of reinforcing or strengthening.

[Wikipedia]

A consequence that will strengthen an organism's future behavior whenever that behavior is preceded by a specific antecedent stimulus.

Reinforcement in nature

Animals learn to optimize their behavior in the face of rewards and punishments.

So is human

The big bang theory (S.3, E.3)

Example: robot bartender

About Shaker

1. Functionalities: recognize faces/places, pick up bottles, transfer virtual money to Mr. Babbage's account, etc
2. Problem: Shaker is a general purpose robot, knows nothing about any specific tasks initially, does not know when to use which skill
3. Difficulty: pre-programming Shaker is expensive
4. Solution: reinforcement learning is inexpensive, ready to use

About Shaker's RL module

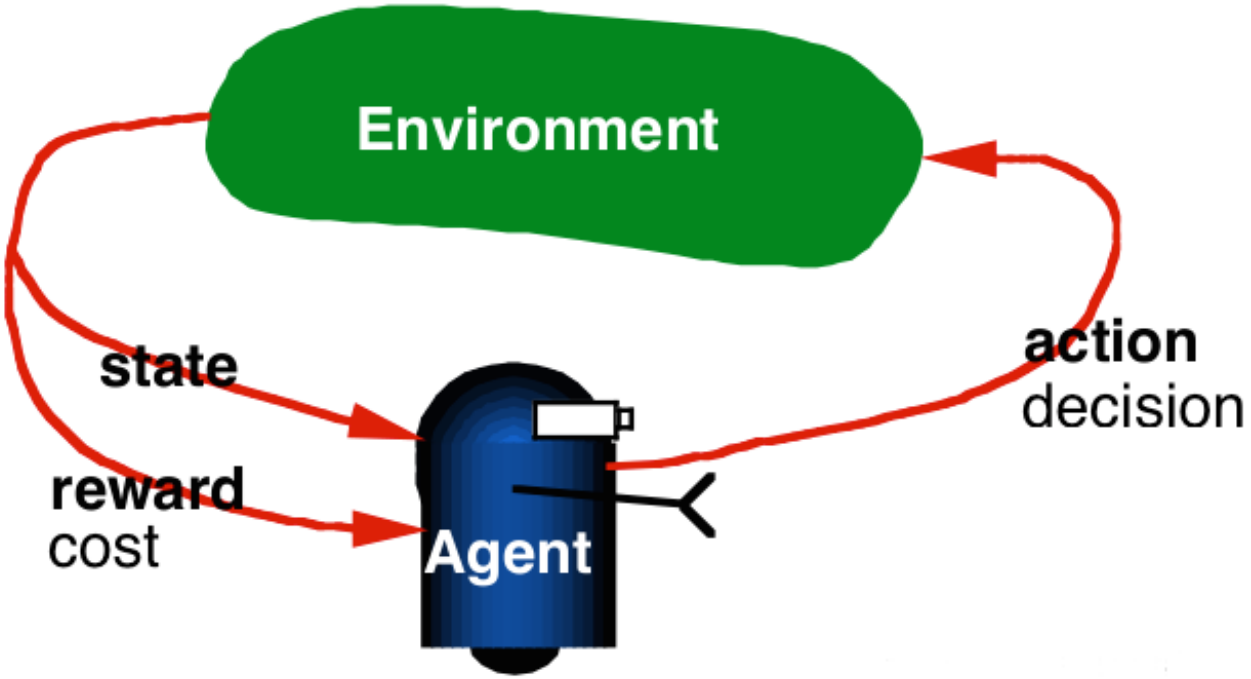
1. Goal: implicitly specified by numerical reward signals, wisely choose its actions to maximize the overall sum of rewards
2. Value function: future rewards that Shaker tries to maximize
3. Input: receive rewards or punishments based on the actions Shaker takes
4. Output: randomly select from the limited set of actions
5. Algorithm: *Q* learning

Characteristics of reinforcement learning

1. There is no supervisor, only reward signals
2. Feedback/reward can be delayed, not instantaneous
3. Trial-and-error approach
4. Exploration and exploitation
5. Time really matters (sequential, non i.i.d data)
6. Agent's actions affect the subsequent data it receives

An agent-system RL interface

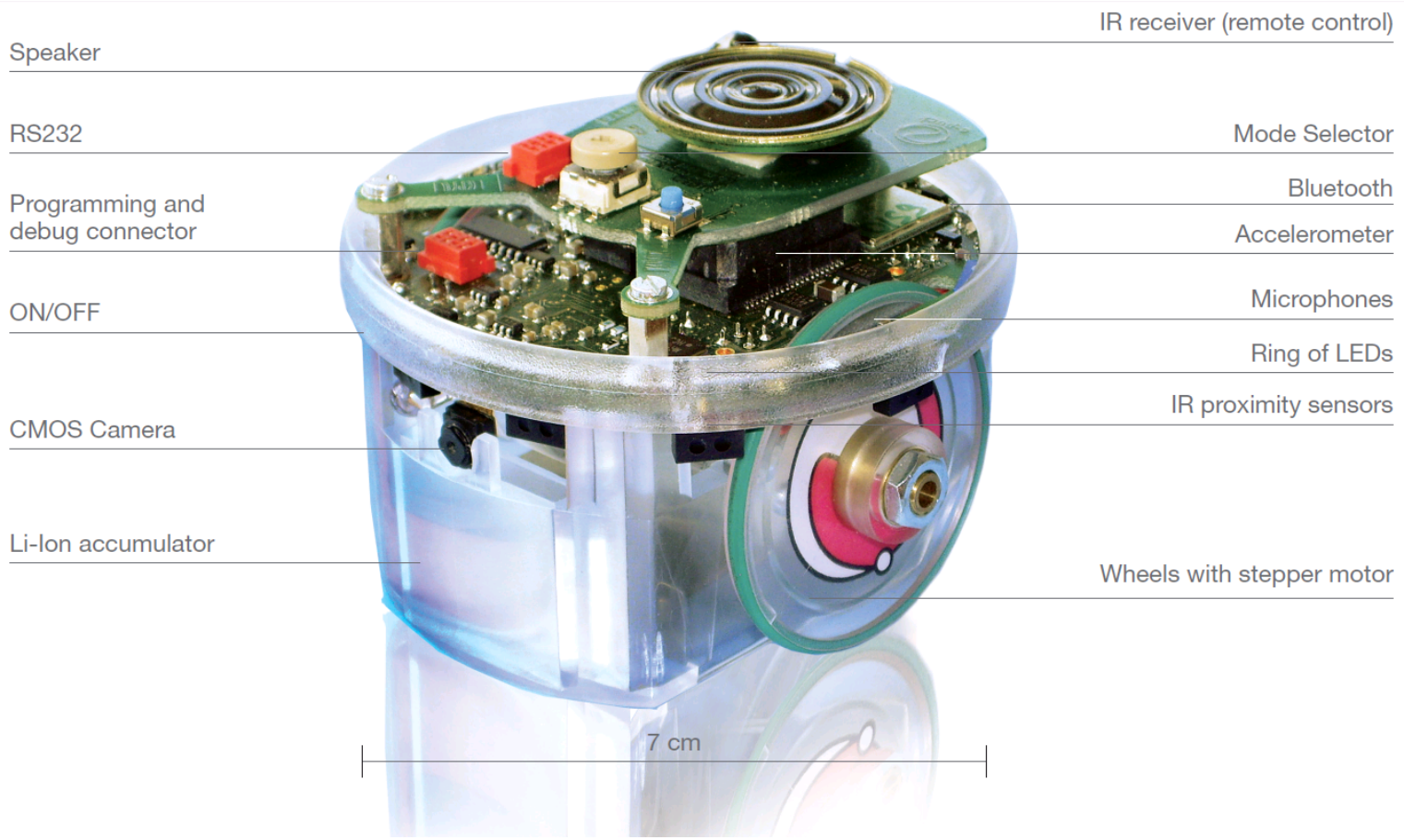
A class of learning problems in which an agent/algorithm interacts with a complex, stochastic, and incompletely known environment in order to achieve a goal.



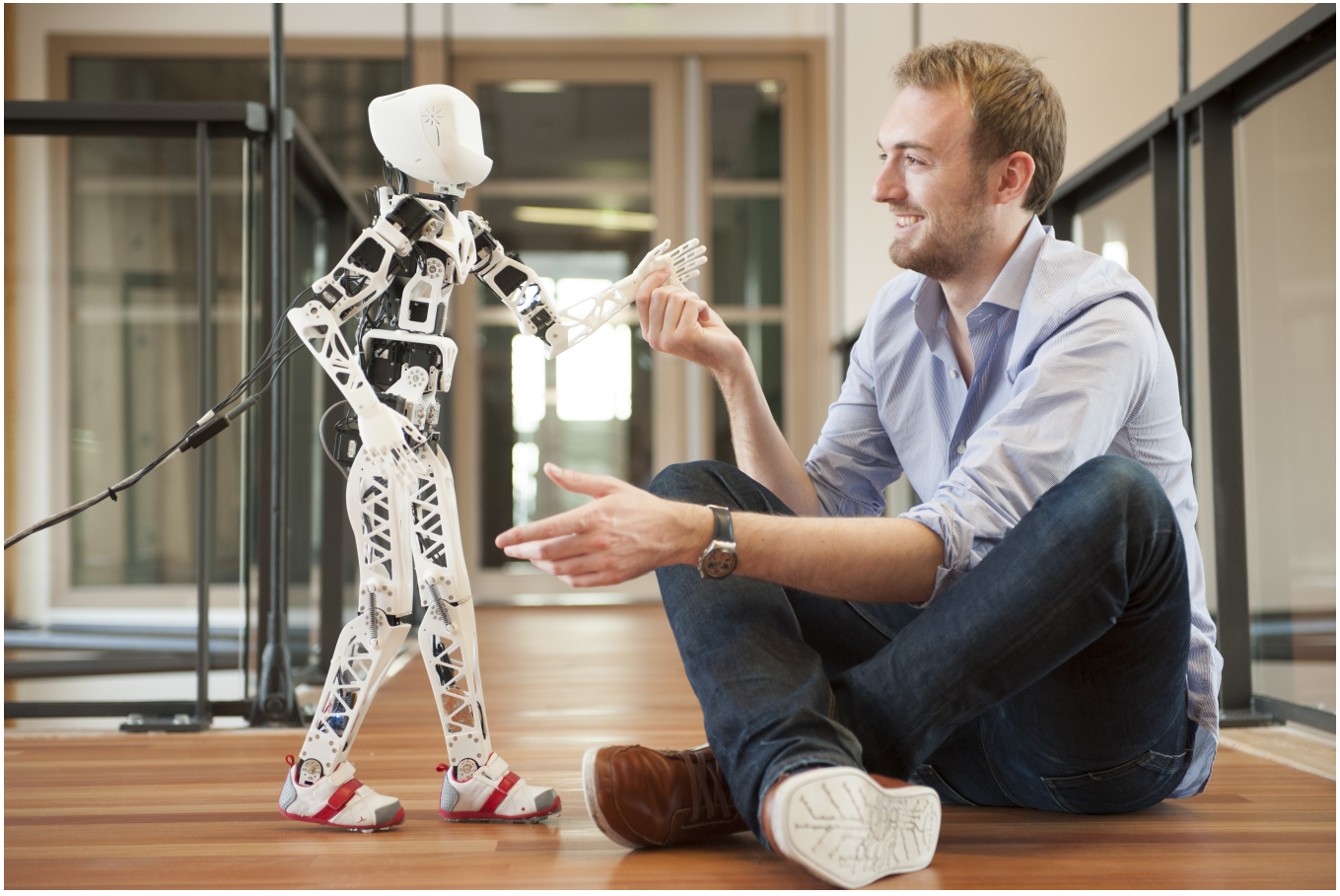
Structure of the course

- ECTS: 6
- Phase one: frontal teaching
  - Six days compact lectures
  - Six hours per day
  - Lecture & tutorial
- Phase two: group project (3-4 persons per group)
  - Two robot platforms (E-puck or Poppy)
  - Two hours group meeting per week
  - Six hours project work per week
  - One hour per week (volunteered consultation)

E-puck


IR receiver (remote control)
Speaker
RS232
Mode Selector
Bluetooth
Programming and
debug connector
Accelerometer
ON/OFF
Microphones
Ring of LEDs
IR proximity sensors
CMOS Camera
Li-Ion accumulator
Wheels with stepper motor
7 cm

Poppy



Assessment

- Self assessment: 45 min quiz on 01.04.2016 (no grading)
- Final grade:
  - Project proposal and intermediate reports (20%)
  - Codes and results (30%)
  - Final project demonstration and presentations (50%)

Schedule of phase one

Day 1 Introduction

Day 2 Dynamic Programming

Day 3 Sampling Based Methods

Day 4 Advanced TD Methods

Day 5 Approximate RL

Day 6 Numerical RL Methods

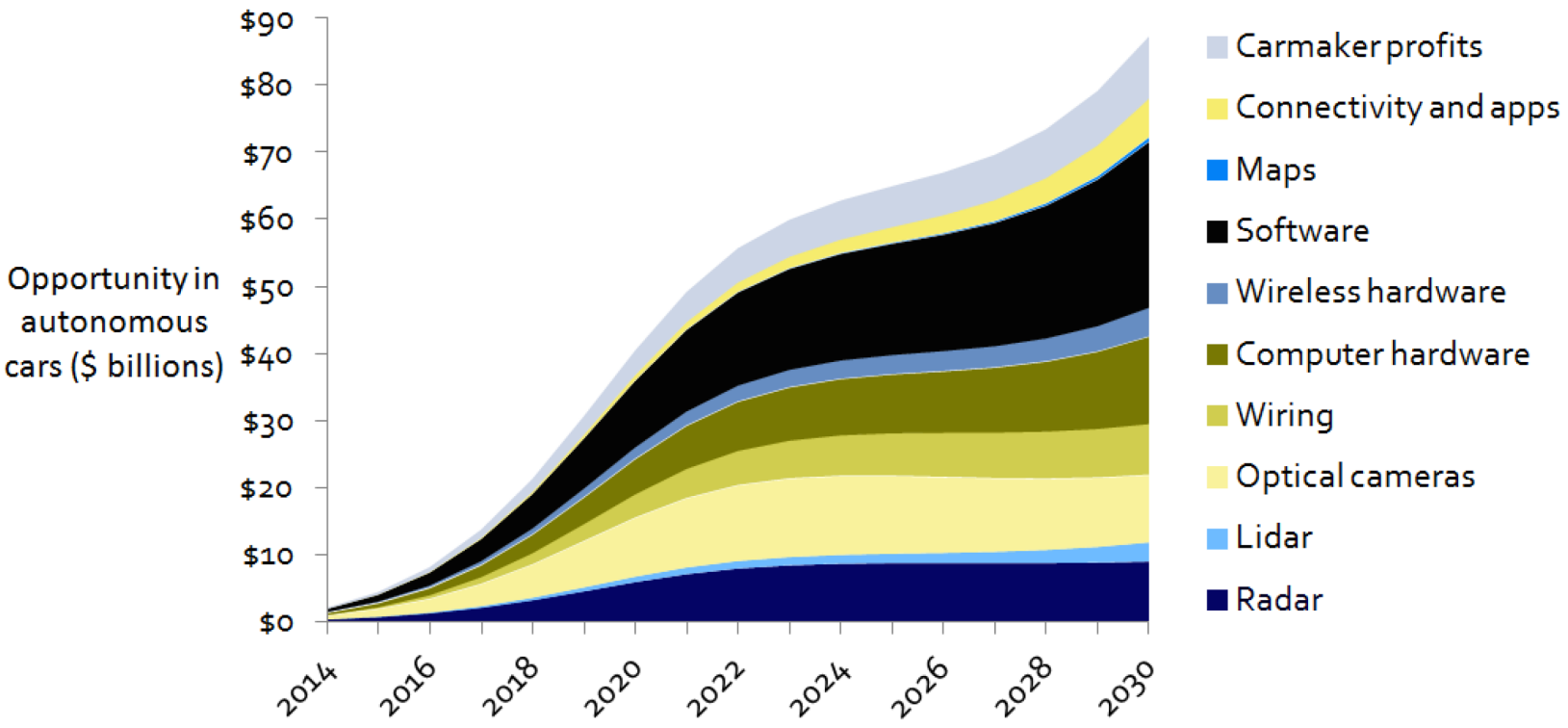
Literature


R. S. Sutton and A. G. Barto.
Reinforcement Learning: An Introduction.
The MIT Press, 1998.


D. P. Bertsekas and J. Tsitsiklis.
Neuro-dynamic programming.
Athena Scientific, 1996.


S. Szepesvári.
Algorithms for Reinforcement Learning.
Morgan & Claypool, 2010.

A coming wave in automobile industry



Source: Lux Research, Inc.
www.luxresearchinc.com

A remark

Groucho Marx

“While money can’t buy happiness, it certainly lets you choose your own form of misery.”

H. S.

“While grade can’t bring happiness, it certainly lets you choose your own form of misery.”

Part I

Day one: Introduction

Outline

Sequential Decision Making

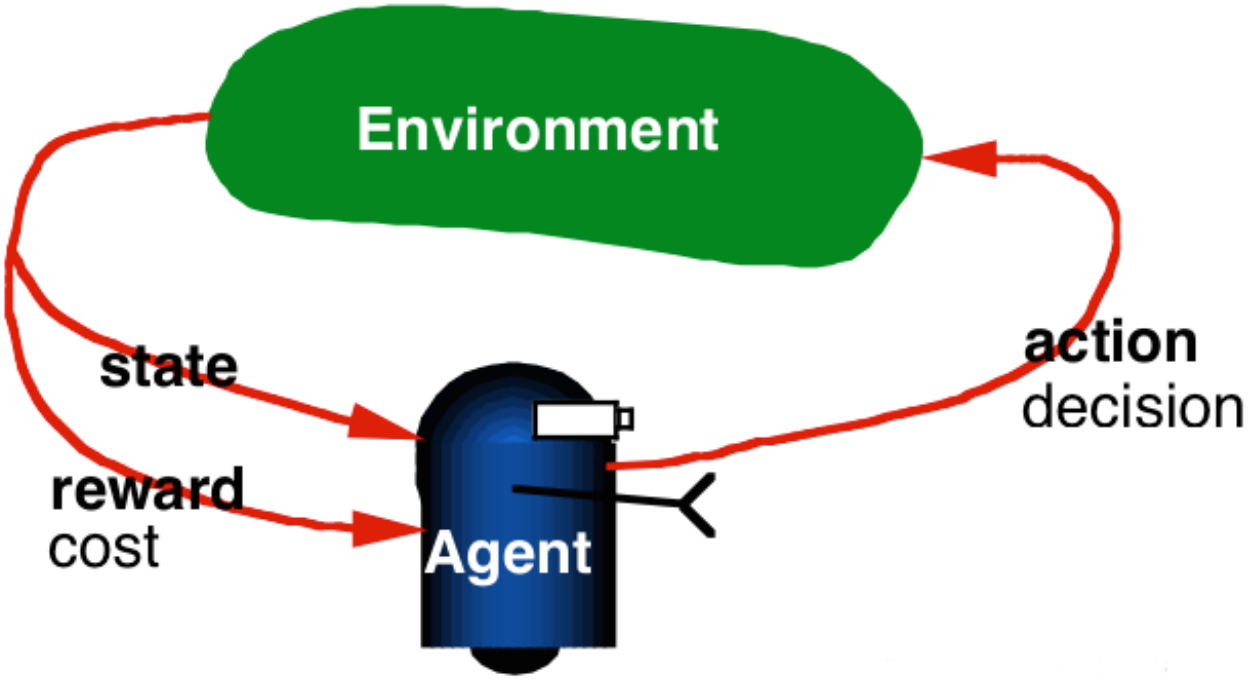
Markov Decision Processes

Simple Non-iterative Solutions

Outline

An agent-system RL interface

A class of learning problems in which an agent/algorithm interacts with a complex, stochastic, and incompletely known environment in order to achieve a goal.



On the degree of abstraction of RL

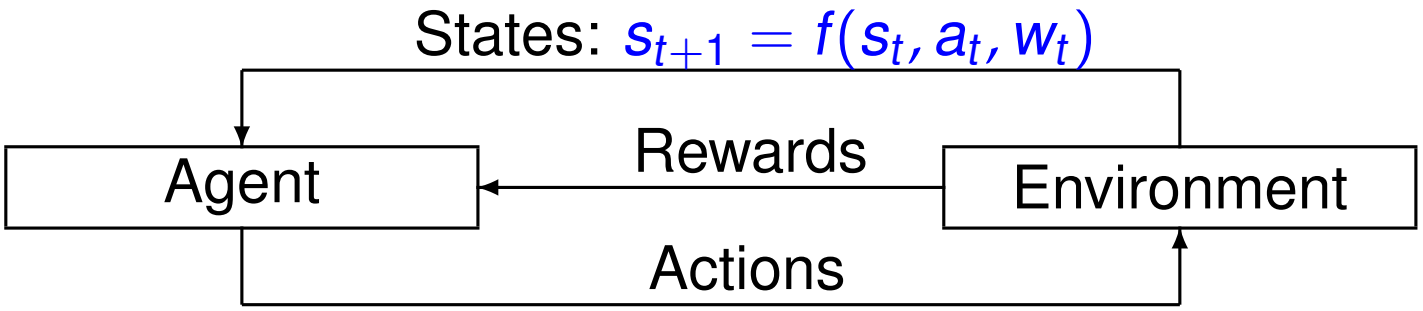
Agent: the learner and decision maker

Environment: everything outside the agent

Modelling goal-oriented learning problem

1. The action: the choices made by the agent
2. The states: the basis on which the choices are made
3. The rewards: one signal to define the agent's goal

State evolution

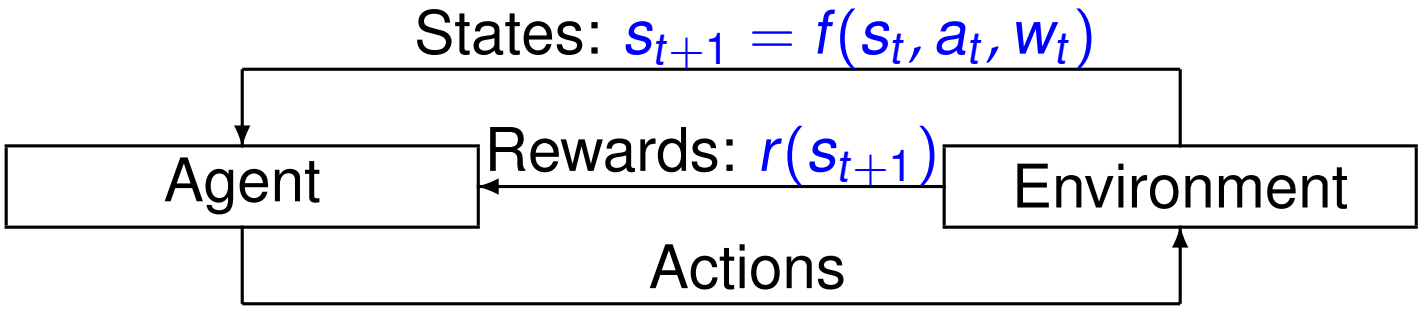


System equation

$$s_{t+1} = f(s_t, a_t, w_t)$$

1. State $s_t \in \mathcal{S}$: past information that is relevant for the future
2. Action $a_t \in \mathcal{A}(s_t) \subset \mathcal{A}$
3. Uncertainty w_t: random variable

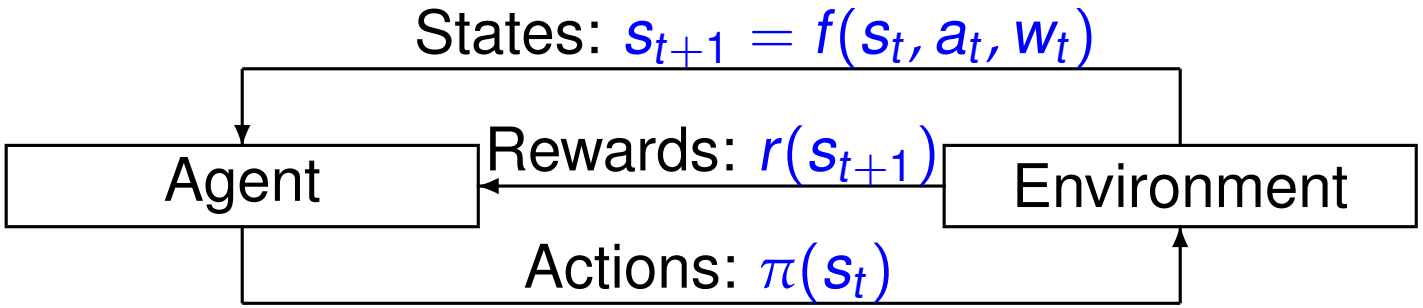
Reward function



State evaluation

$$r(s_{t+1}) := r(f(s_t, a_t, w_t)) \in \mathbb{R}$$

Policy: choice of actions



Definition

A policy is a mapping from states to actions $\pi : \mathcal{S} \to \mathcal{A}$.

1. Deterministic: $\pi(s) = a$
2. Stochastic: $\pi(s) \sim p(a|s)$

Rewards and goals

Question

Is a scalar reward signal an adequate notion of a goal?

- A goal should specify what we want to achieve, not how we want to achieve it
- A goal must be outside the agent's direct control – thus outside the agent
- The agent must be able to measure success: (1) explicitly; (2) frequently during its lifespan

From rewards to the goal

Question

Given a sequence of rewards as

$$\{r_0 := r(s_0, a_0, w_0), r_1, r_2, r_3, \ldots\},$$

how do we describe the goal?

Idea

Use the rewards to formalize an overall evaluation of the goal

$$R : \mathcal{S} \to \mathbb{R} \qquad s_0 \mapsto g(r_0, r_1, r_2, \ldots)$$

The return function: sum of rewards

Hypothesis (the *reward hypothesis*)

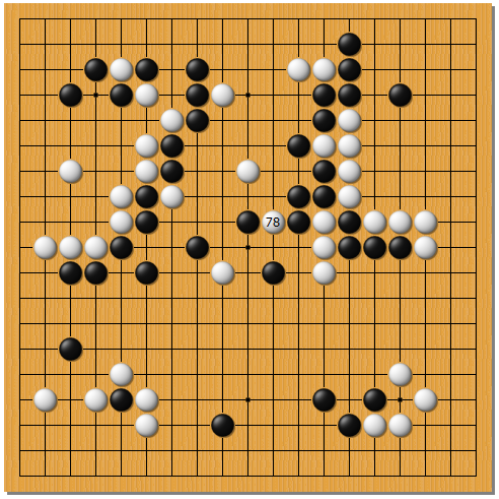
That all of what we mean by goals and purposes can be well thought of as the minimizsation of the cumulative sum of a received reward signal.

Definition

The return function for finite horizon problems is defined as

$$R(s_0) := \sum_{t=0}^{T} r(s_t, a_t, w_t)$$

Finite horizon problem: episodic



Interactions of system

$$s_k \overset{r_k}{\rightsquigarrow} a_k \to s_{k+1} \overset{r_{k+1}}{\rightsquigarrow} a_{k+1} \to s_{k+2} \ldots \to s_T$$

$$\uparrow w_k \qquad\qquad \uparrow w_{k+1}$$

Infinite horizon problem: continuing

Interactions of system

$$s_k \overset{r_k}{\curvearrowright} a_k \to s_{k+1} \overset{r_{k+1}}{\curvearrowright} a_{k+1} \to s_{k+2} \ldots \to s_\infty$$
$$\uparrow w_k \qquad\qquad \uparrow w_{k+1}$$

Discounted return function

Definition (infinite horizon)

The return function for infinite horizon problems is

$$R(s_0) := \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, w_t)$$

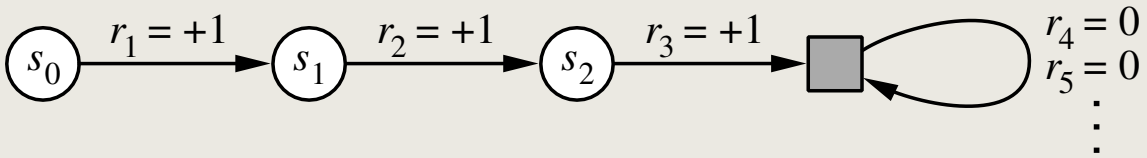
Why discount?

- Avoid infinite returns in infinite horizon problems
- Uncertainty about the future may not be fully represented

A unified notation

Idea

Think of each episode as ending in an absorbing state that always produces reward of zero:



Value function

A discounted return under a given policy π is

$$R^{\pi}(s_0) := \sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t), w_t)$$

The expected return (value function) of a given policy π is defined as

$$\begin{aligned} V^{\pi}(s_0) &:= \mathop{\mathbb{E}}_{w_t, t=0,\ldots,\infty} [R^{\pi}(s_0)] \\ &= \mathop{\mathbb{E}}_{w_t, t=0,\ldots,\infty} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t), w_t) \right] \end{aligned}$$

Sequential decision making problem

Definition (Control problem)

Let Π be the set of admissible policies, find a policy $\pi^* \in \Pi$, which solves

$$\pi^* := \underset{\pi \in \Pi}{\operatorname{argmin}} V^{\pi}(s), \qquad \text{for all } s \in \mathcal{S},$$

which is known as an optimal policy

Optimal value function

Definition

The *optimal value function* $V^*(s)$ is the minimum value function over all policies, i.e.

$$V^*(s) := \min_{\pi \in \Pi} V^\pi(s).$$

Definition

A policy π^* is called optimal, if it satisfies, for all $s \in \mathcal{S}$

$$V^{\pi^*}(s) = V^*(s)$$

Outline

Modelling state transitions

State transition as a stochastic process

Sequence of transitions $\{s_1, s_2, \ldots\}$ can be modeled as a *stochastic process*

$$s_1 \quad \rightarrow \quad s_2 \quad \ldots \quad \rightarrow \quad s_\infty$$

Definition

States s_t are *Markov* if and only if

$$P(s_{t+1}|s_1, \ldots, s_t) = P(s_{t+1}|s_t)$$

Alternative system description

System evolution

$$s_{t+1} = f(s_t, a_t, w_t)$$

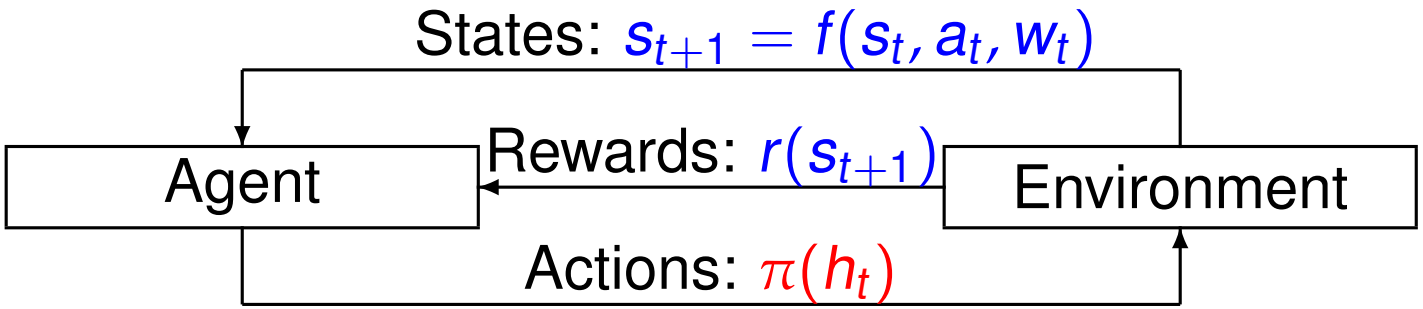
- System equation is deterministic
- Let $P(w_t|s_t, a_t)$, then $P(s_{t+1}|s_t, a_t) = P(w_t|s_t, a_t)$
- Let $w_t = s_{t+1}$, then $P(w_t|s_t, a_t) = P(s_{t+1}|s_t, a_t)$

Markov decision processes

A Markov decision process is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$

1. $\mathcal{S}$ is a finite set of states
2. $\mathcal{A}$ is a finite set of actions
3. $\mathcal{P}$ represents the state transition probability, $P(s'|s,a)$
4. r is a reward function, $r(s,a,s') \in \mathbb{R}$
5. $\gamma \in (0,1]$ is a discount factor

History-dependent policy



Definition

A history-depedent policy is a mapping from histories to actions $\pi : \mathcal{H} \to \mathcal{A}$.

1. Deterministic: $\pi(h) = a$
2. Stochastic: $\pi(h) \sim p(a|h)$

Adequacy of Markov policy

Theorem

Let $\pi_h \in \Pi$ be a stochastic, history-dependent policy. Then for each initial state $s \in \mathcal{S}$, there exists a stochastic Markov policy $\pi_m \in \Pi$ such that

$$V^{\pi_m} = V^{\pi_h}.$$

Adequacy of Markov policy: discussion

The value function corresponding to a stochastic history-dependent policy can be obtained by a stochastic Markov policy.

The performance of policies is still bounded by the modelling of the problme.

The Bellman equations

The value function of a given policy π is defined as

$$
\begin{aligned}
V^{\pi}(s_0) &= \mathop{\mathbb{E}}_{s_{t+1}\in\mathcal{S}}\Big[\sum_{t=0}^{\infty}\gamma^t r(s_t,\pi(s_t),s_{t+1})\Big] \\
&= \mathop{\mathbb{E}}_{s_{t+1}\in\mathcal{S}}\Big[r(s_0,\pi(s_0),s_1)+\sum_{t=1}^{\infty}\gamma^t r(s_t,\pi(s_t),s_{t+1})\Big] \\
&= \mathop{\mathbb{E}}_{s_{t+1}\in\mathcal{S}}\Big[r(s_0,\pi(s_0),s_1)+\gamma\sum_{t=1}^{\infty}\gamma^{t-1} r(s_t,\pi(s_t),s_t)\Big] \\
&= \mathop{\mathbb{E}}_{s_1\in\mathcal{S}}\Big[r(s_0,\pi(s_0),s_1)+\gamma V^{\pi}(s_1)\Big]
\end{aligned}
$$

Bellman equations

- The optimal value function is defined as

$$V^*(s_0) := \min_{a_t \in \mathcal{A}(s_t)} \mathop{\mathbb{E}}_{s_{t+1}, t=0,\ldots,\infty} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \right]$$

- Similar result also holds for the optimal value function as

$$V^*(s_0) := \min_{a_0 \in \mathcal{A}} \mathop{\mathbb{E}}_{s_1 \in \mathcal{S}} [r(s_0, a_0, s_1) + \gamma V^*(s_1)]$$

- Proof is much more subtle

Outline

Task of prediction

Question

Given two policies π and π', which one is better? Or

$$V^{\pi}(s) > V^{\pi'}(s)?$$

Policy prediction

Given a policy π, evaluate the value function $V^{\pi}(s)$.

Expected reward function

- Recall the Bellman equation, for all $s \in \mathcal{S}$,

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{s' \in \mathcal{S}} \Big[r(s, \pi(s), s') + \gamma V^{\pi}(s') \Big]$$
$$= \sum_{s' \in \mathcal{S}} P(s'|s) \Big(r(s, \pi(s), s') + \gamma V^{\pi}(s') \Big)$$

- Let K be the cardinality of the state space, i.e., $K = |\mathcal{S}|$
- Let $r^{\pi}(s) = \mathop{\mathbb{E}}_{s' \in \mathcal{S}} \Big[r(s, \pi(s), s') \Big]$, define

$$R^{\pi} := [r^{\pi}(s_1), \ldots, r^{\pi}(s_K)]^{\top} \in \mathbb{R}^K$$

State transition matrix

- State transition matrix P defines transition probabilities from all states s to all successor states s',

$$P := \begin{bmatrix} p_{11} & \cdots & p_{1K} \\ & \ddots & \\ p_{K1} & \cdots & p_{KK} \end{bmatrix}$$

  where each row of P sums to one.
- Known as Markov matrix.

Bellman equation in matrix form

- The Bellman expectation equations can be expressed as

$$V^\pi = R^\pi + \gamma P^\pi V^\pi$$

where $V^\pi \in \mathbb{R}^K$ with one entry per state, i.e.

$$\begin{bmatrix} V^\pi(s_1) \\ \vdots \\ V^\pi(s_K) \end{bmatrix} = \begin{bmatrix} R_1 \\ \vdots \\ R_K \end{bmatrix} + \gamma \begin{bmatrix} p_{11} & \dots & p_{1K} \\ & \ddots & \\ p_{K1} & \dots & p_{KK} \end{bmatrix} \begin{bmatrix} V^\pi(s_1) \\ \vdots \\ V^\pi(s_K) \end{bmatrix}$$

- Solution: solve the linear equation for $V_K \in \mathbb{R}^K$

$$(I_K - \gamma P^\pi)\, V = R^\pi$$

Closed form solution to prediction

- The solution is

$$V = (I_K - \gamma P^\pi)^{-1} R^\pi,$$

  with $I_K - \gamma P^\pi$ being invertible (why?)
- Computational complexity is $O(K^3)$ for K states
- Direct solution only possible for small problems

Solution to control problem

Fact

For the optimal value function V^*

$$V^*(s) := \min_{a \in \mathcal{A}} \mathop{\mathbb{E}}_{s' \in \mathcal{S}} [r(s,a,s') + \gamma V^*(s')]$$

Linear Programming solution

$$\min_{V \in \mathbb{R}^{|\mathcal{S}|}} \quad c^\top V, \qquad \text{s.t. } V(s) \leq \mathop{\mathbb{E}}_{s' \in \mathcal{S}} [r(s,a,s') + \gamma V(s')], \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}$$

with $c \in \mathbb{R}^{|\mathcal{S}|}$ is a probability distribution over $\mathcal{S}$

Part II

Day Two: Dynamic Programming

Outline

Principle of Optimality

Policy Iteration Algorithm

Value Iteration Algorithm

Asynchronous DP Methods

Outline

Origin of dynamic programming (DP)

Originally, the process of solving problems where one needs to find the best decisions one after another (R. Bellman, 1940s)

Generally, an approach of nesting smaller decision problems inside larger decisions (R. Bellman, 1953)

What is dynamic programming (DP)?

Bellman sought an impressive name to avoid confrontation

1. “it’s impossible to use dynamic in a pejorative sense.”
2. “something not even a congressman could object to” (Bellman, R. E., Eye of the Hurricane, An Autobiography).

dynamic: the time-varying aspect of the problems
programming: to find optimal “programs”

Tail subproblems

Interactions of finite horizon system

$$\ldots s_{k-1} \overset{r_{k-1}}{\rightsquigarrow} a_{k-1} \rightarrow s_k \overset{r_k}{\rightsquigarrow} a_k \rightarrow s_{k+1} \ldots \rightarrow s_T$$

$$\left| \text{tail problem:} k \rightarrow \infty \right.$$

Value function of tail problems

$$V^{\pi}(s_k) := \mathop{\mathbb{E}}_{s_{t+1}, t=0,\ldots,\infty} \left[\sum_{t=k}^{T} \gamma^{t-k} r(s_t, \pi(s_t), s_{t+1}) \right]$$

Principle of Optimality

Definition

Let $\pi^* \in \Pi$ be an optimal policy for the MDP problem, and assume that when using π^*, a given state s_t occurs at the time t with a positive probability. Then the truncated policy π^* from any time point k is optimal for this subproblem.

Application of optimal Bellman equation

- Bellman equation for tail problems

$$V^*(s_k) := \min_{a_k \in \mathcal{A}} \mathop{\mathbb{E}}_{s_{k+1} \in \mathcal{S}} [r(s_k, a_k, s_{k+1}) + \gamma V^*(s_{k+1})]$$

- At step T, $V^*(s_T) = r(s_T)$
- At step $T-1$, we have

$$\pi^*(s_{T-1}) := \operatorname*{argmin}_{a_{T-1} \in \mathcal{A}} \mathop{\mathbb{E}}_{s_T \in \mathcal{S}} [r(s_{T-1}, a_{T-1}, s_T) + \gamma V^*(s_T)]$$

- At step $T-2$, ...

Classic dynamic programming algorithm

Pseudocode

For $t = T - 1, \ldots, 1$, computes

$$V(s_t) = \min_{a_t \in \mathcal{A}(s_t)} \mathbb{E}_{s_{t+1} \in \mathcal{S}} \left[r(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1}) \right]$$

Convergence

A value function computed by the dynamic programming is optimal. Then the associated policy is optimal.

Question

How to deal with infinite horizon problems?

Outline

Bellman equation for policy π

- Deterministic policy $\pi\colon s \mapsto a$

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{s' \in \mathcal{S}} \Big[r(s, \pi(s), s') + \gamma V^{\pi}(s') \Big]$$

- Stochastic Markov policy $\pi\colon s \mapsto p(a|s)$

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{a \sim \pi(s), s' \in \mathcal{S}} \Big[r(s, a, s') + \gamma V^{\pi}(s') \Big]$$

- Let's consider $V^{\pi} \in \mathbb{R}^K$, Bellman equation serves as a map

$$\mathcal{T}_{\pi}\colon \mathbb{R}^K \to \mathbb{R}^K$$

Bellman operator for policy π

The Bellman operator $\mathcal{T}_\pi : \mathbb{R}^K \to \mathbb{R}^K$

Component-wise,

$$(\mathcal{T}_\pi V)(s) := \mathop{\mathbb{E}}_{a \sim \pi(s), s' \in \mathcal{S}} \Big[r(s, a, s') + \gamma V(s') \Big]$$

Lemma

For any policy π, the Bellman operator $\mathcal{T}_\pi$ is a contraction w.r.t. the max-norm, i.e. for any $V_1, V_2 \in \mathbb{R}^K$,

$$\|\mathcal{T}_\pi V_1 - \mathcal{T}_\pi V_2\|_\infty \leq \gamma \|V_1 - V_2\|_\infty, \qquad \gamma \in (0, 1)$$

Property of the Bellman operator

Question

What does the contraction property mean?

- Let us define $\mathcal{T}_\pi^k := \mathcal{T}_\pi \circ \mathcal{T}_\pi \circ \ldots \circ \mathcal{T}_\pi$ (k times)
- We have

$$\left\|\mathcal{T}_\pi^k V_1 - \mathcal{T}_\pi^k V_2\right\|_\infty \leq \gamma^k \|V_1 - V_2\|_\infty \xrightarrow{k\to\infty} 0$$

Properties of the Bellman operator

- Recall definition of Bellman equation

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{a \sim \pi(s), s' \in \mathcal{S}} \left[r(s, a, s') + \gamma V^{\pi}(s') \right]$$

- Fact: $\mathcal{T}_{\pi}(V^{\pi}) = V^{\pi}$

Theorem

For any policy π, the value function $V^{\pi} \in \mathbb{R}^{K}$ is a fixed point of the Bellman operator $\mathcal{T}_{\pi}$.

Iterative policy evaluation

Fact

$$\begin{aligned}\|V_k - V^\pi\|_\infty &= \|\mathcal{T}_\pi^k V_0 - \mathcal{T}_\pi^k V^\pi\|_\infty \\ &\leq \gamma^k \|V_0 - V^\pi\|_\infty \\ &\xrightarrow{k\to\infty} 0\end{aligned}$$

Abstract of IPE

Apply the Bellman operator iteratively

$$V_0 \to V_1 \to \ldots \underbrace{V_k \to V_{k+1}}_{\text{a sweep}} \ldots \to V^\pi$$

Convergence properties of IPE

Pseudocode (Policy Evaluation)

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

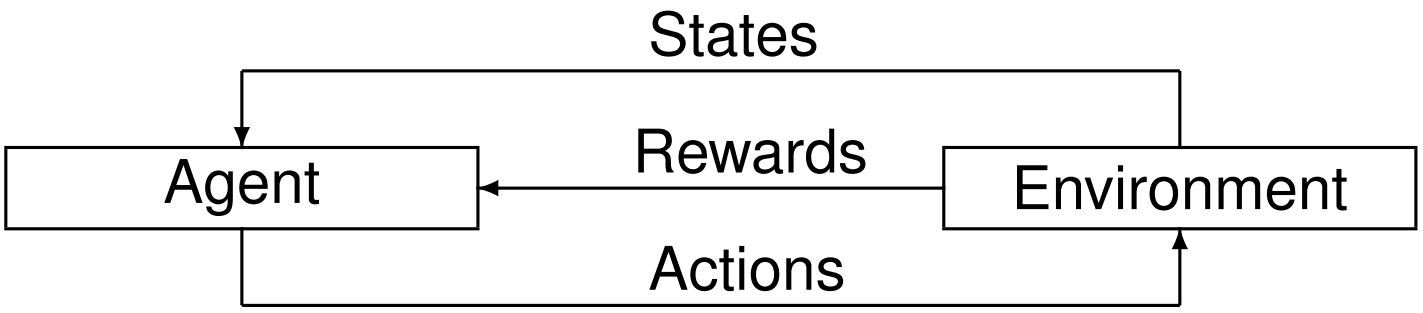
$$V_{k+1}(s) = \mathop{\mathbb{E}}_{a \sim \pi(s), s' \in \mathcal{S}} \left[r(s, a, s') + \gamma V_k(s') \right]$$

until converge

Lemma

For a given policy π *and an arbitrary* $V_0 \in \mathbb{R}^K$, *the IPE converges to the true value function* V^π.

Task of control



Ultimate goal

Find an optimal policy π^*

Idea

Given a policy π, we can solve the prediction task. Can we use value functions to structure the search for good policies?

Policy improvement

- The optimal policy is defined as

$$\pi^*(s) := \underset{a \in \mathcal{A}(s)}{\operatorname{argmin}} \underset{s' \in \mathcal{S}}{\mathbb{E}} \Big[r(s,a,s') + \gamma V^*(s') \Big]$$

- Given a policy π, instead of taking $a \sim \pi(s)$, we can improve the policy by acting greedily

$$\pi'(s) := \underset{a \in \mathcal{A}(s)}{\operatorname{argmin}} \underset{s' \in \mathcal{S}}{\mathbb{E}} \Big[r(s,a,s') + \gamma V^{\pi}(s') \Big]$$

- This is an improvement, because for any state s,

$$V^{\pi'}(s) \leq V^{\pi}(s)$$

Policy iteration

Idea

We can combine policy evaluation and improvement to obtain a sequence of monotonically improving policies and value functions

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \ldots \xrightarrow{I} \pi^* \xrightarrow{E} V^*$$

- $\xrightarrow{E}$: *policy evaluation*
- $\xrightarrow{I}$: *policy improvement ("greedification")*

Policy iteration (Pseudocode)

Pseudocode (PI)

Choose any initial policy π_0, iterate

1. *Policy evaluation: iterate until converge*

$$V^{\pi_i} = \mathcal{T}_{\pi_i}^k V \text{ with } k \to \infty$$

2. *Policy improvement: compute*

$$\pi_{i+1}(s) := \underset{a \in \mathcal{A}(s)}{\operatorname{argmin}} \underset{s' \in \mathcal{S}}{\mathbb{E}} \left[r(s, a, s') + \gamma V^{\pi_i}(s') \right]$$

until converge

Convergence of PI

- The sequence of $\{V^{\pi_k}\}_k$ is non-decreasing
- State and action spaces are finite

Theorem

Policy iteration generates a sequence of policies with increasing performance ($V^{\pi_{k+1}} \leq V^{\pi_k}$) and it terminates in a finite number of steps with the optimal policy π^*.

Policy iteration comments

- In each step of policy iteration
  - Policy evaluation involves solving a set of linear equations
  - Policy improvement: straightforward
- Each step of policy iteration is guaranteed to strictly improve the policy at some state, when improvement is possible

Outline

Optimal Bellman operator

- Recall the optimal Bellman equation

$$V^*(s) = \min_{a \in \mathcal{A}(s)} \mathbb{E}_{s' \in \mathcal{S}} \Big[r(s,a,s') + \gamma V^*(s') \Big]$$

The optimal Bellman operator $\mathcal{T}^* : \mathbb{R}^K \to \mathbb{R}^K$

Component-wise,

$$(\mathcal{T}^* V)(s) := \min_{a \in \mathcal{A}(s)} \mathbb{E}_{s' \in \mathcal{S}} \Big[r(s,a,s') + \gamma V(s') \Big]$$

Properties of the optimal Bellman operator

Lemma

The optimal Bellman operator $\mathcal{T}^*$ is a contraction in the max-norm, i.e. for any $V_1, V_2 \in \mathbb{R}^K$,

$$\|\mathcal{T}^* V_1 - \mathcal{T}^* V_2\|_\infty \leq \gamma \|V_1 - V_2\|_\infty$$

Theorem

The value function $V^* \in \mathbb{R}^K$ is a fixed point of the optimal Bellman operator $\mathcal{T}^*$.

Value iteration (Pseudocode)

Pseudocode (VI)

Iterate

$$V_{k+1} = \mathcal{T}^* V_k$$

until converge

Convergence properties of VI

Convergence

$$\|V_{k+1} - V^*\|_\infty \leq \gamma^{k+1} \|V_0 - V^*\|_\infty$$
$$\xrightarrow{k \to \infty} 0$$

Proposition

For an arbitrary $V_0 \in \mathbb{R}^K$, the VI algorithm converges to the optimal value function V^*.

Outline

Synchronous DP

Example: value iteration

For all $s \in \mathcal{S}$, iterate

$$V_{k+1} = \mathcal{T}^* V_k$$

until converge

Facts

- All states are updated simultaneously, i.e. slow convergence
- Difficult for parallel and distributed computation
- Hard for simulation-based implementations

Gauss-Seidel VI method (synchronous)

Pseudocode

Initialize V_0 and $i = 0$, then iterate

1. For $j = 1, \ldots, K$, do

$$V_i(s_j) \leftarrow (\mathcal{T}^* V_i)(s_j) := \min_{a \in \mathcal{A}(s_j)} \mathbb{E}_{s' \in \mathcal{S}} \Big[r(s_j, a, s') + \gamma V_i(s') \Big];$$

2. Set $i = i + 1$;

until converge

Convergence

For an arbitrary $V_0 \in \mathbb{R}^K$, the Gauss-Seidel VI algorithm converges to the optimal value function V^*.

Asynchronous DP

Idea

Instead going through a full sweep, i.e. for all $s \in \mathcal{S}$, we randomly pick state $s \in \mathcal{S}$ and then apply an appropriate update, PI or VI.

Asynchronous value iteration

Pseudocode

For arbitrarily chosen $s \in \mathcal{S}$, iterate

$$V(s) \leftarrow (\mathcal{T}^* V)(s) := \min_{a \in \mathcal{A}(s)} \mathbb{E}_{s' \in \mathcal{S}} \left[r(s, a, s') + \gamma V(s') \right]$$

until converge

Asynchronous policy iteration

Pseudocode (API)

Choose any initial policy π, iterate

1. *Local policy evaluation: for an arbitrary $s \in \mathcal{S}$*

$$V(s) = \mathop{\mathbb{E}}_{a \sim \pi(s), s' \in \mathcal{S}} \Big[r(s, a, s') + \gamma V(s') \Big]$$

2. *Local policy improvement: for an arbitrary $s \in \mathcal{S}$*

$$\pi(s) = \operatorname*{argmin}_{a \in \mathcal{A}(s)} \mathop{\mathbb{E}}_{s' \in \mathcal{S}} \Big[r(s, a, s') + \gamma V(s') \Big]$$

until converge

Something not right? – No full policy evaluation

Asynchronous policy iteration, corrected

Pseudocode (API)

Choose any initial policy π, iterate

1. *Local policy evaluation: for an arbitrary $s \in \mathcal{S}$*

$$V(s) = \mathop{\mathbb{E}}_{a \sim \pi(s), s' \in \mathcal{S}} \Big[r(s,a,s') + \gamma V(s') \Big]$$

2. *Local policy improvement: for an arbitrary $s \in \mathcal{S}$*

$$\pi(s) = \operatorname*{argmin}_{a \in \mathcal{A}(s)} \mathop{\mathbb{E}}_{s' \in \mathcal{S}} \Big[r(s,a,s') + \gamma V(s') \Big]$$

and

$$V(s) = \min_{a \in \mathcal{A}(s)} \mathop{\mathbb{E}}_{s' \in \mathcal{S}} \Big[r(s,a,s') + \gamma V(s') \Big]$$

until converge

Convergence conditions for Asynchronous DP

Synchronous convergence condition

- Let $\mathcal{V}_0 \subset \mathbb{R}^K$, we define

$$\mathcal{V}_{i+1} := \{\mathcal{T}^* V | \text{ for all } V \in \mathcal{V}_i\},$$

  and assume $\mathcal{V}_{i+1} \subset \mathcal{V}_i$
- Every sequence $\{V_t\}$ with $V_t \in \mathcal{V}_i$, for all $i = 0, 1, \ldots,$ converges pointwise to V^*

Box condition

For all i, $\mathcal{V}_k$ is a Cartesian product of the form

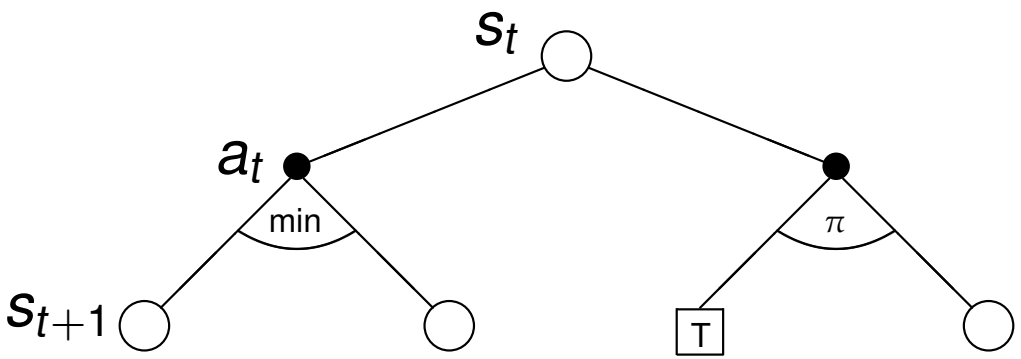
$$\mathcal{V}_i = \mathcal{V}_i^{(1)} \times \ldots \times \mathcal{V}_i^{(K)}$$

Abstract of RL

System transitions

$$\ldots s_k \overset{r_k}{\rightsquigarrow} a_k \to s_{k+1} \overset{r_{k+1}}{\rightsquigarrow} a_{k+1} \to s_{k+2} \ldots \to s_\infty$$

Update/backup diagram



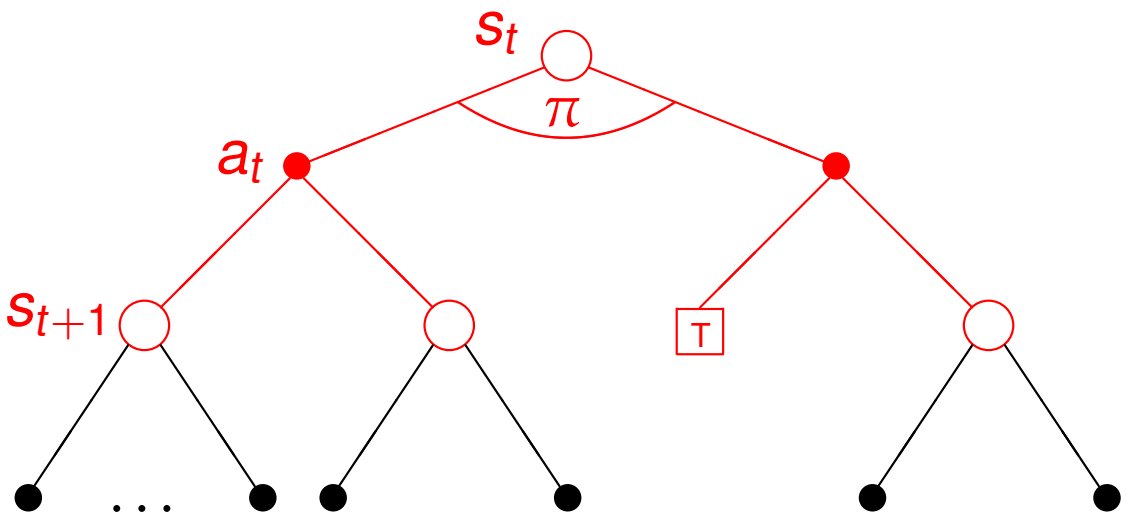
Concepts

state, terminal state, action, minimization, average under π

Iterative policy evaluation

Update rule

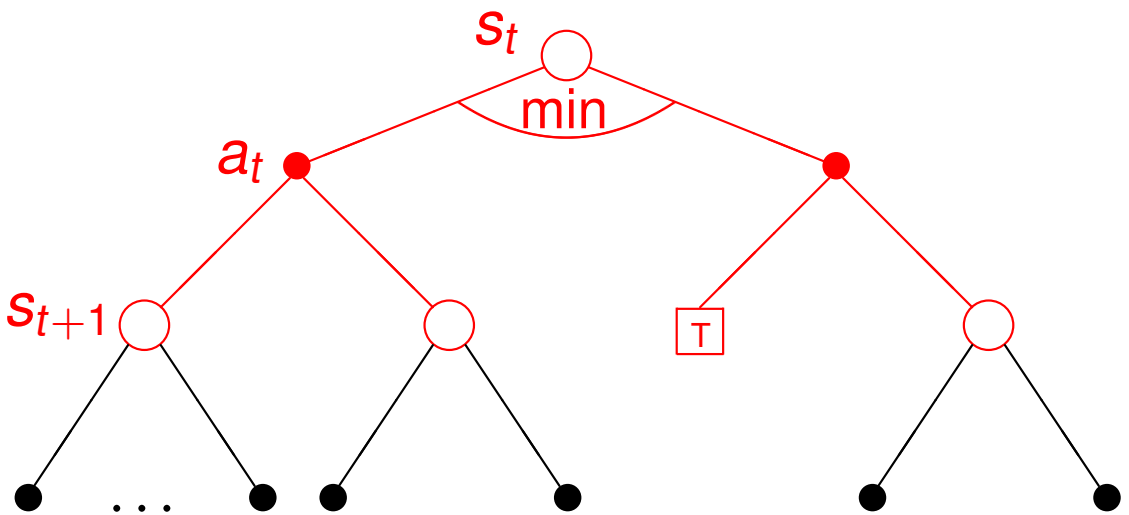
$$V(s_t) \leftarrow \mathop{\mathbb{E}}_{a_t \sim \pi(s_t), s_{t+1} \in \mathcal{S}} [r(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1})]$$



Value iteration

Update rule

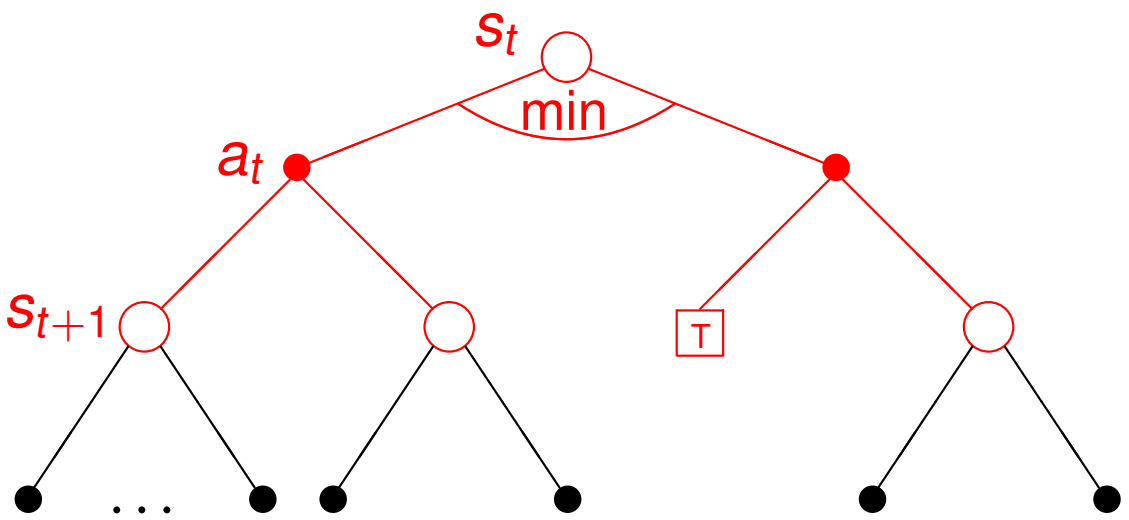
$$V(s_t) \leftarrow \min_{a_t \sim \mathcal{A}(s_t)} \mathop{\mathbb{E}}_{s_{t+1} \in \mathcal{S}} [r(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1})]$$



Part III

Day Three: Sampling Based Methods

Is DP really RL?



- No interaction with the environment, i.e. essentially offline
- Assume complete information about the environment

Outline

Monte Carlo Methods

TD Prediction Methods

On- and Off-policy Learning

Multi-armed bandits

Outline

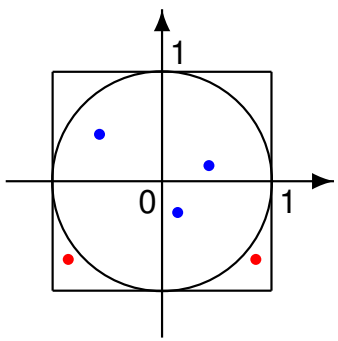
Curse of model?

What if $(\mathcal{S}, \mathcal{A}, r, \gamma)$?

Calculation of π

Idea

The value of π is the area of a circle with unit radius.



$$Pr(\text{hit in the circle}) = \frac{\text{area of the circle}}{\text{area of the square}} = \frac{\pi}{4}$$

1. Draw two random numbers (x, y) uniformly over $[-1, 1]$
2. Generate a random variable h

$$h = \begin{cases} 1, & \text{if } x^2 + y^2 \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

3. Compute the expectation

$$\mathbb{E}[h_i] = \frac{\pi}{4}$$

Basics of Monte Carlo methods

Law of large numbers (LNN)

Let $\{x_i\}_{i=1}^{\infty}$ be a sequence of independently identically distributed random variables such that $\mathbb{E}[x_i] = \mu$ and $\mathrm{Var}(x_i) < \infty$, then almost surely

$$\frac{1}{n}(x_1 + \ldots + x_n) \to \mu, \qquad \text{when } n \to \infty$$

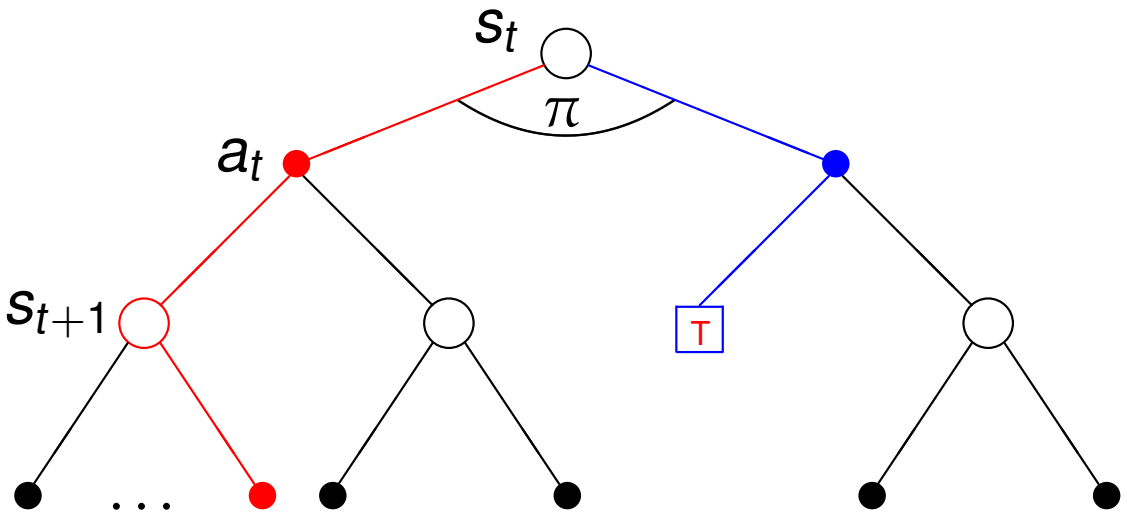
Monte Carlo methods

Any method that uses randomness and the LLN to compute a certain quantity which might have nothing to do with randomness is a *Monte Carlo* method.

Key components of Monte Carlo methods

1. Define a domain of possible inputs
2. Generate inputs randomly from a probability distribution over the domain
3. Perform a deterministic computation on the inputs
4. Aggregate the results

Overview



Task

Learn V^π from experience under π

MC policy evaluation

Fact

The value at s is the expected (discounted) return

$$V^\pi(s) := \mathbb{E}[R^\pi(s)]$$

Idea

Use empirical mean of return, instead of expected return

$$V^\pi(s) \approx \frac{1}{N} \sum_{i=1}^{N} \widehat{R}_i^\pi(s),$$

with $\widehat{R}_i^\pi(s)$ being the i-th realization of $R^\pi(s)$

Incremental calculation of empirical mean

- Let $x \in \mathbb{R}$ be a random variable, and the empirical mean

$$\mu_N = \frac{1}{N} \sum_{i=1}^{N} x_i$$

- With the $(N+1)$-th sample, we compute

$$\begin{aligned} \mu_{N+1} &= \frac{1}{N+1} \left(N\mu_N + x_{N+1} \right) \\ &= \mu_N + \frac{1}{N+1} \left(x_{N+1} - \mu_N \right) \end{aligned}$$

Incremental MC policy evaluation

Pseudocode

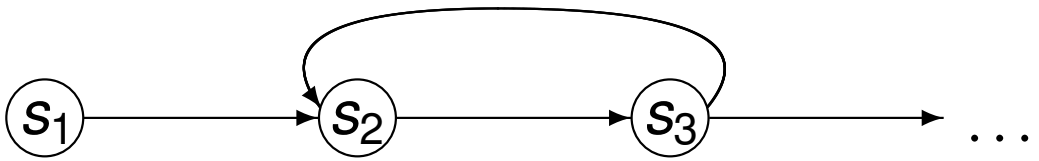
Iterate

1. Draw trajectory $\{s_0 = s, s_1, \ldots, s_T\}$, under policy π
2. Compute a realization $\widehat{R}(s)$
3. Set $N(s) = N(s) + 1$
4. Update $V(s) \leftarrow V(s) + \frac{1}{N(s)}(\widehat{R}(s) - V(s))$

4.* In non-stationary problems, update

$$V(s) \leftarrow V(s) + \alpha(\widehat{R}(s) - V(s)), \qquad \text{with } 0 < \alpha < 1$$

Trajectory with loops



Facts

- A sequence of states are visited
- A state can be visited multiple times

First-visit MC policy evaluation

Pseudocode

Iterate

1. Draw trajectory $\mathcal{T} := \{s_0 = s, s_1, \ldots, s_T\}$, under policy π
2. For all state s' appearing in $\mathcal{T}$:
   (i) Compute a realization $\widehat{R}(s')$, from the first appearance of s' until the run ends
   (ii) Set $N(s') = N(s') + 1$
   (iii) Update $V(s') \leftarrow V(s') + \frac{1}{N(s')}(\widehat{R}(s') - V(s'))$

Every-visit MC policy evaluation

Pseudocode

Iterate

1. Draw trajectory $\mathcal{T} := \{s_0 = s, s_1, \ldots, s_T\}$, under policy π
2. For $i = 0, 1, \ldots, T$:
   (i) Compute a realization $\widehat{R}(s_i)$, from the current appearance until the run ends
   (ii) Set $N(s_i) = N(s_i) + 1$
   (iii) Update $V(s') \leftarrow V(s_i) + \frac{1}{N(s_i)}(\widehat{R}(s_i) - V(s_i))$

Implementation technique

Idea

Exploring starts: *Explore many different start states*

In order to always explore, we need to have soft policies, e.g.

$$\pi(a|s) > 0 \text{ for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}(s)$$

i.e. every state-action pair has a non-zero probability of being the starting pair

MC-PI algorithms

General scheme

Combine MC policy evaluation with the standard PI algorithm

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots$$

- $\xrightarrow{E}$: MC policy evaluation
- $\xrightarrow{I}$: policy improvement (“greedification”)

Example: first visit MC-PI

Pseudocode

Repeat:

1. *Generate an episode under π, starting arbitrarily with $s \in \mathcal{S}$*
2. *For each s' in this episode*
   (i) *Let $\widehat{R}(s')$ be the return following the first occurrence of s'*
   (ii) *Update the values of s using incremental mean*

$$V(s') = V(s') + \frac{1}{N(s')+1}\left(\widehat{R}(s') - V(s')\right)$$

3. *For each s' in this episode (policy improvement)*

$$\pi(s') := \underset{a \in \mathcal{A}(s)}{\operatorname{argmax}} \underset{s'' \in \mathcal{S}}{\mathbb{E}}\left[r(s', a, s'') + \gamma V(s'')\right]$$

Convergence of MC control

- Policy improvement still works if evaluation is done with MC, by the policy improvement theorem
- Assumption: exploring starts and infinite number of episodes for MC policy evaluation

MC-VI methods?

- The optimal Bellman operator

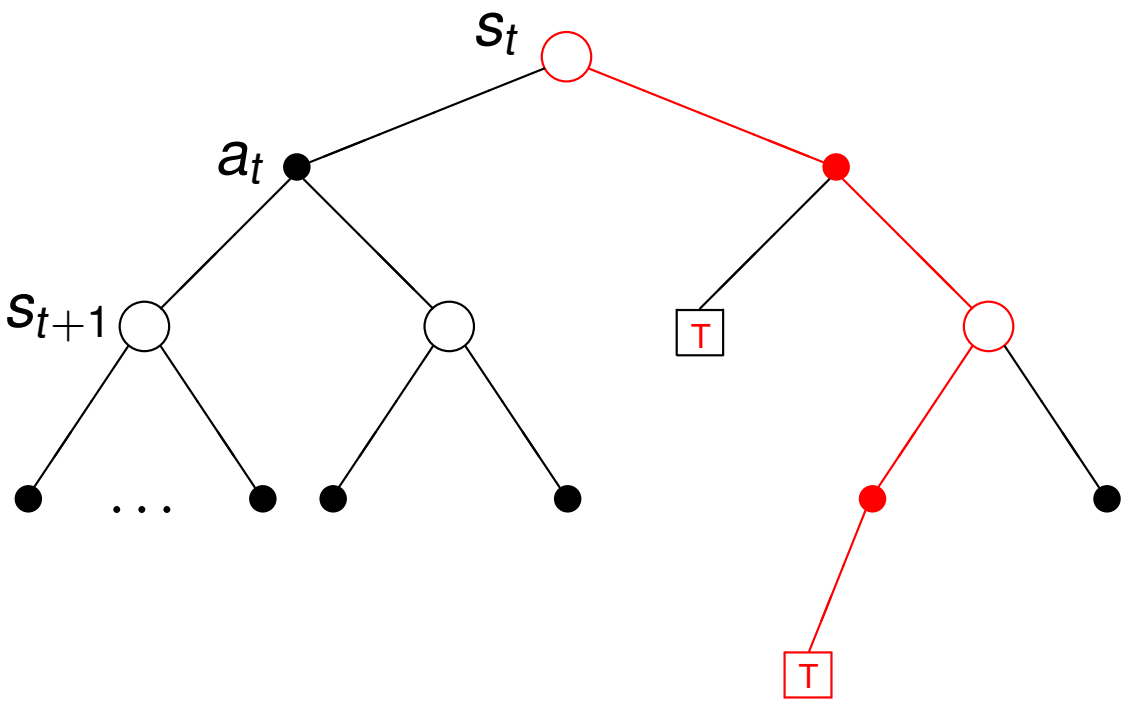
$$(\mathcal{T}^* V)(s_t) := \min_{a_t \sim \mathcal{A}(s_t)} \mathop{\mathbb{E}}_{s_{t+1} \in \mathcal{S}} [r(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1})]$$

- Impossible to sample minimization of expectation!

Policy evaluation in MC

Update rule

$$V(s_t) \leftarrow V(s_t) + \alpha\Big(R^{\pi}(s_t) - V(s_t)\Big)$$



Outline

Revisit to the value function

Motivation: Bellman equation

$$V^{\pi}(s) = \mathbb{E}_{\pi}\Big[R^{\pi}(s)\Big]$$
$$= \mathop{\mathbb{E}}_{a\sim\pi(s),s'\in\mathcal{S}}\Big[r(s,a,s') + \gamma V^{\pi}(s')\Big]$$

Approximating the return (*TD target*)

$$R(s) \approx r(s,a,s') + \gamma V(s')$$

TD policy evaluation

Key steps

1. Pick a start state $s \in \mathcal{S}$
2. Repeat for each step
   - (i) Take action a using policy π derived from V
   - (ii) Observe the next state s' and its reward $r(s, a, s')$
   - (iii) Set $V(s) \leftarrow V(s) + \alpha(r(s, a, s') + \gamma V(s') - V(s))$

Some facts of TD prediction

Temporal difference of value functions

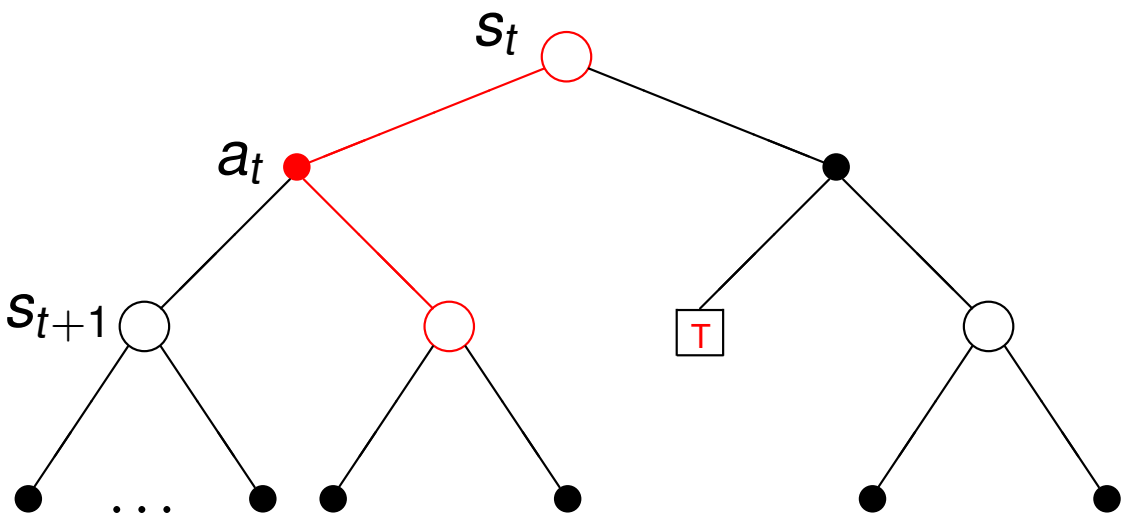
$$V'(s) - V(s) \propto r(s,a,s') + \gamma V(s') - V(s)$$

- The term $r(s,a,s') + \gamma V(s') - V(s)$ is known as *the TD error*
- Sensitive to initial values

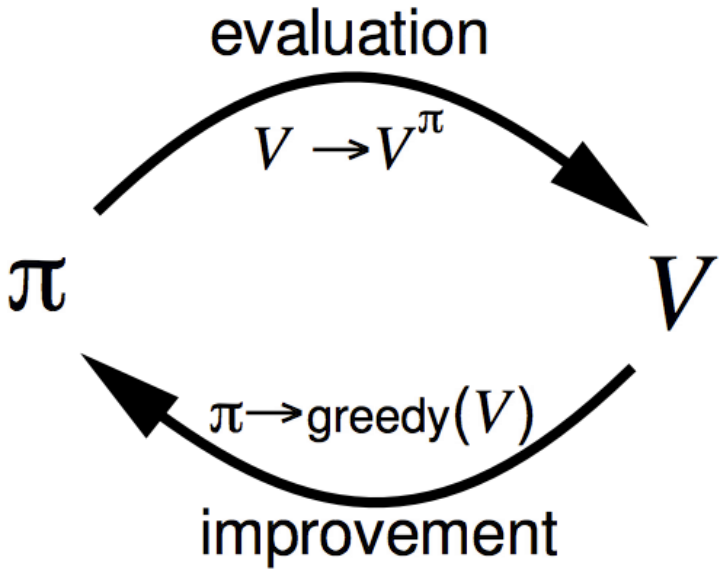
Policy evaluation in TD

Update rule

$$V(s_t) \leftarrow V(s_t) + \alpha\Big(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\Big)$$



Naïve TD control



1. Policy evaluation: TD prediction
2. Policy improvement: greedify with respect to approximate value function

Outline

Overview

Goal

Evaluate one policy while following another

- On-policy learning
  - "Learn on the job"
  - Learn about π from experience sampled from π
- Off-policy learning
  - "Look over someone's shoulder"
  - Learn about π from experience sampled from π'

Off-Policy RL methods

Notation

- π': Behavior policy, i.e. the policy being followed
- π: Estimation/target policy, i.e. the policy being learned about

Assumption

Every action taken under π is also taken under π', i.e.

$$\pi(a|s) > 0 \quad \Rightarrow \quad \pi'(a|s) > 0$$

Learning π while following π'

Question

How to estimate R^{π} from π', or $R^{\pi'}$?

Assumption

- Let $p(s)$ be the probability of the complete sequence following π and starting from s, so $p'(s)$ for π'
- The return on this experience for policy π is computed as

$$R^{\pi}(s) = \frac{p(s)}{p'(s)} R^{\pi'}(s)$$

Learning π while following π'

- Let experiences/sample episodes indexed by i
- The average value of following π from s is

$$V^{\pi}(s) \approx \sum_i \frac{p_i(s)}{p'_i(s)} \widehat{R}_i^{\pi'}(s) \Big/ \sum_i \frac{p_i(s)}{p'_i(s)}$$

Problem

The probability $p(s)$ of the path from s is not known.

Learning π while following π'

- The probability of the path from s is computed as

$$p(s) = \prod_{k=t_i}^{T_i(s)-1} \pi(a_k|s_k)P(s_{k+1}|s_k, a_k)$$

  where t_i is the first visit time to state s in episode i, and $T_i(s)$ is the time this episode ended
- Note, that $P(s_{k+1}|s_k, a_k)$ is unknown. But

$$\frac{p(s)}{p'(s)} = \frac{\prod_{k=t_i}^{T_i(s)-1} \pi(a_k|s_k)P(s_{k+1}|s_k,a_k)}{\prod_{k=t_i}^{T_i(s)-1} \pi'(a_k|s_k)P(s_{k+1}|s_k,a_k)} = \prod_{k=t_i}^{T_i(s)-1} \frac{\pi(a_k|s_k)}{\pi'(a_k|s_k)}$$

Off-Policy MC Control

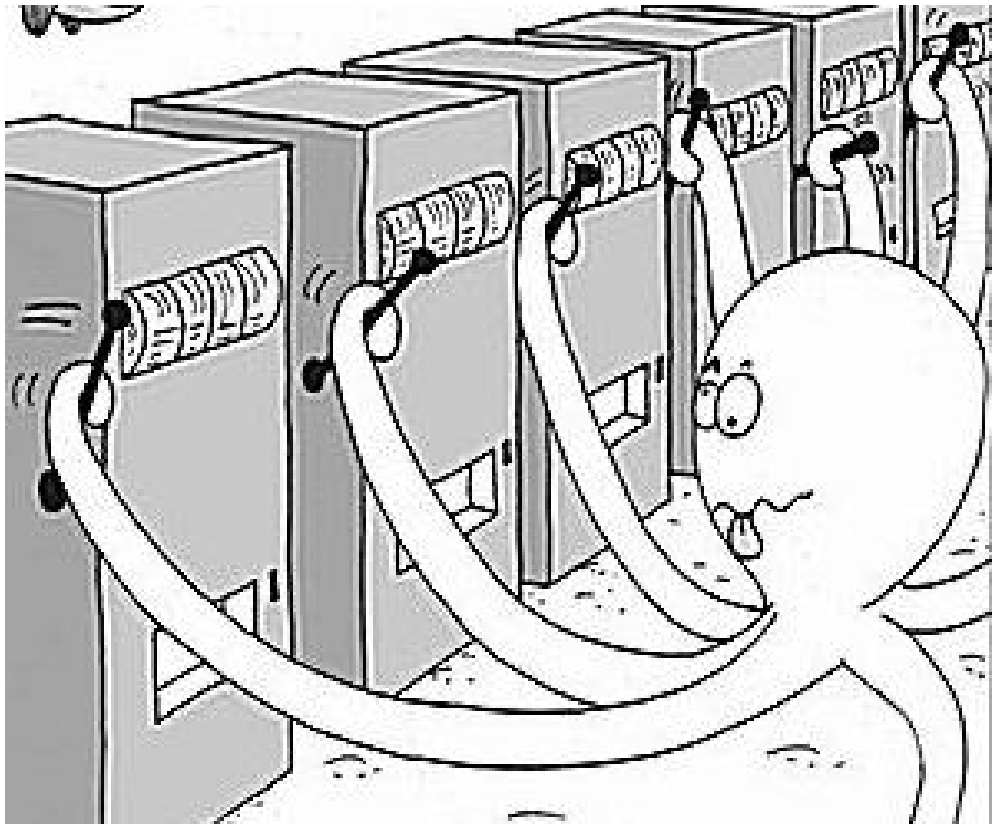
- The value function is estimated by

$$V^{\pi}(s) \approx \frac{\sum_i \prod_{k=t_i}^{T_i(s)-1} \frac{\pi(a_k|s_k)}{\pi'(a_k|s_k)} R^{\pi'}(s)}{\sum_i \prod_{k=t_i}^{T_i(s)-1} \frac{\pi(a_k|s_k)}{\pi'(a_k|s_k)}}$$

- Assume that the policy is deterministic, i.e. $\pi(a_k|s_k) = 1$

Outline

N-armed Bandits



Multi Armed Bandit, `http://www.research.microsoft.com/`

N-armed bandits

- Choose repeatedly from a set of N actions, each choice is denoted a play.
- Each play a_t ends up in a reward r_t.
- It is not known how the reward is distributed.
- Objective: Maximize reward in the long term.
- Way: Explore actions and exploit the best.

Action-value methods

- Method for estimating the values of actions.
- Estimation should satisfy $Q_t(a) \approx Q^*(a)$

- Playing k times
- Simple average: $Q_t(a) = \frac{R_1 + R_2 + ... + R_k}{k}$

$$\lim_{k \to \infty} Q_t(a) = Q^*(a)$$

- Do all rewards have to be stored?

Exploration/exploitation dilemma

- Best action: $a_t^* = \underset{a}{\operatorname{argmax}}\, Q_t(a)$
- Exploration: $a_t \neq a_t^*$
- Exploitation: $a_t = a_t^*$

- You can't exploit all the time, you can't explore all the time.
- What is the trade-off?
- But maybe you want to reduce exploring over the time.

ϵ-greedy action selection

- Greedy action selection:

$$a_t = a_t^* = \underset{a}{\text{argmax}}\, Q_t(a)$$

- Balance exploration and exploitation.
- Simple to control.

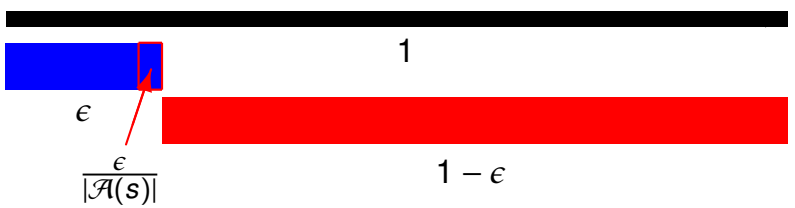
ϵ-greedy policy

- $$a_t = \begin{cases} a_t^* & \text{with } p = 1 - \epsilon \\ \text{random} & \text{with } p = \epsilon \end{cases}$$

- Random action with probability ϵ

ϵ-Greedy Policy

- Non-greedy action with probability $\frac{\epsilon}{|\mathcal{A}(s)|}$
- Greedy action with probability $1 - \epsilon + \frac{\epsilon}{|\mathcal{A}(s)|}$, i.e. $\pi(s,a) \geq \frac{\epsilon}{|\mathcal{A}(s)|}$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$



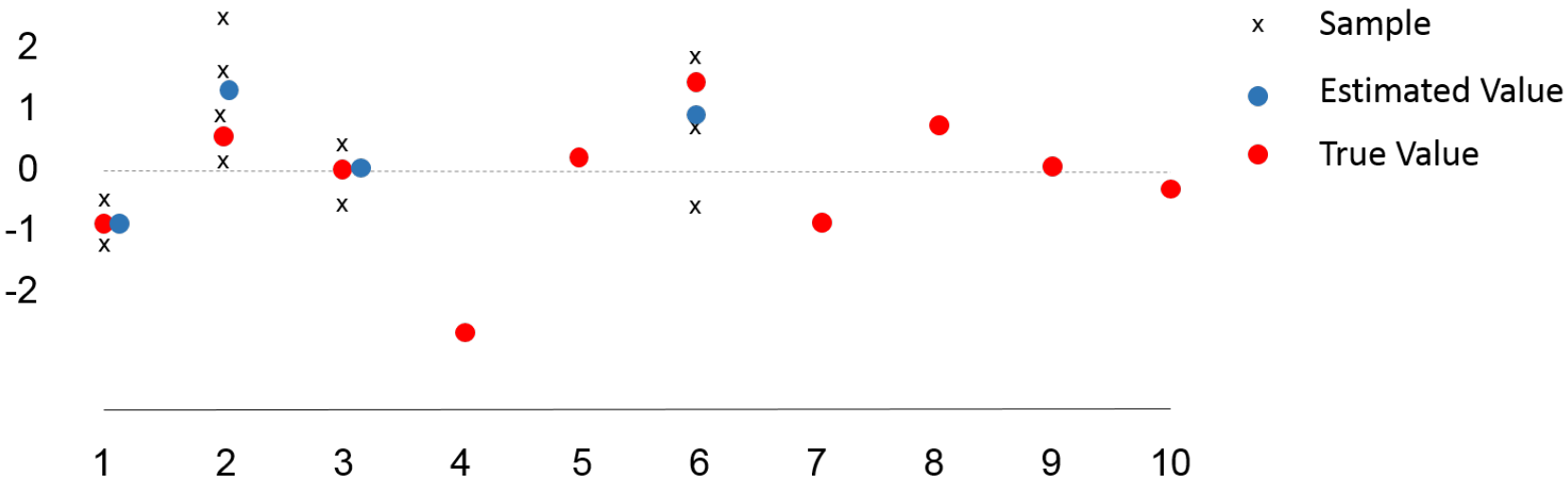
Multi-Armed Bandit 02

10-Armed Testbed

- n = 10 possible actions
- each $Q_*(a)$ is chosen randomly from $\mathcal{N}(0,1)$
- each r_t is normal distributed $r_t \approx \mathcal{N}(Q_*(a_t),1)$
- one run is 1000 plays
- average over 2000 runs

Is ϵ-Greedy the best solution?

- Is it reasonable to try each arm with equal probability?
- Suppose you see the following samples:



Softmax Action Selection

- Converts values into action probabilities
- A way to order the actions you want to explore
- The most common softmax uses Gibbs or Boltzmann distribution:

$$P(a) = \frac{e^{Q_t(a)/\tau}}{\sum_{b=1}^{n} e^{Q_t(b)/\tau}}$$

where τ is the *temperature parameter*

- If τ is high, all actions are (almost) equiprobable
- If τ is low, the policy more likely to be greedy

Multi-Armed Bandit 03

Gibbs Distribution

$$P(a) = \frac{e^{Q(a)/\tau}}{\sum_{b=1}^{n} e^{Q(b)/\tau}}$$

Show that in the case of two actions, the softmax operation using the Gibbs distribution becomes the logistic (or sigmoid) function. What effect does the temperature parameter have on the function? What does the function do to the action-values?

Multi-Armed Bandit 03: Gibbs Distribution (Solution)

Two actions: $n = 2$, therefore

$$P(a) = \frac{e^{Q(a)/\tau}}{e^{Q(a)/\tau} + e^{Q(b)/\tau}} = \frac{1}{1 + e^{Q(b)/\tau} \cdot e^{-Q(a)/\tau}} \quad (1)$$

$$= \frac{1}{1 + e^{(Q(b)-Q(a))/\tau}} \quad (2)$$

$$= \frac{1}{1 + e^{-t}}, \quad (3)$$

where $t = -\frac{Q(b)-Q(a)}{\tau}$.

Multi-Armed Bandits 04

Evaluation vs. Instruction

- binary bandit task
- rewards: success, failure
- supervised algorithm: infer from success or failure which was correct action
- choose action which was correct most

Multi-Armed Bandits 04

Evaluation vs. Instruction

L_{R-P} Algorithm

linear, reward penalty: d_t correct action, w_t failure action

$$\pi_{t+1}(d_t) = \pi_t(d_t) + \alpha[1 - \pi_t(d_t)]$$

$$\pi_{t+1}(w_t) = \pi_t(w_t) + \alpha[0 - \pi_t(w_t)]$$

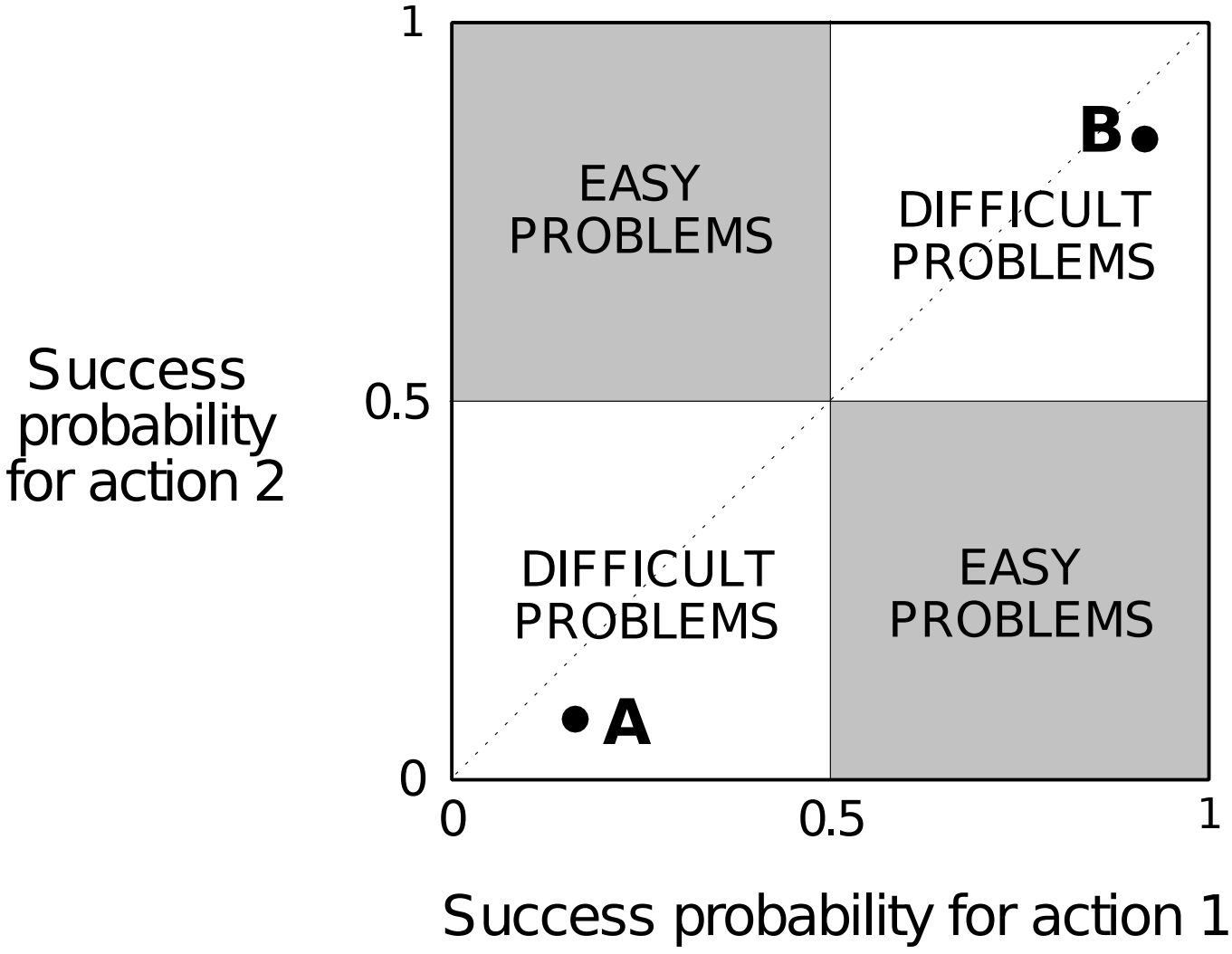
L_{R-I} Algorithm

linear, reward-inaction: update only upon success

$$\pi_{t+1}(d_t) = \pi_t(d_t) + \alpha[1 - \pi_t(d_t)]$$

Multi-Armed Bandits 04

Evaluation vs. Instruction



Part IV

Day Four: Advanced TD Methods

Outline

The Q Function

TD Methods in Q

N-step TD Prediction

Forward $TD(\lambda)$ Algorithm

Eligibility Trace / Backward $TD(\lambda)$ Algorithm

Outline

Recall: the state-value function

Definition

State-value function for policy π is defined as

$$V^{\pi}(s) := \mathbb{E}_{\pi}[R^{\pi}(s)]$$

$$= \mathop{\mathbb{E}}_{a_t \sim \pi(s_t), s_{t+1} \in \mathcal{S}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \middle| s_0 = s \right]$$

Fact

It is hard to compute the expectation under a policy without knowing the model of the environment.

Recall: the optimal policy

Definition

Given the optimal value function V^*. An optimal policy π^* is defined as

$$\pi^*(s) := \operatorname*{argmin}_{a \in \mathcal{A}} \mathop{\mathbb{E}}_{s' \in \mathcal{S}} \Big[r(s, a, s') + \gamma V^*(s') \Big]$$

Fact

It is hard to compute the expectation without knowing transition probability.

The action-value function

Definition

The action-value function for policy π is defined as

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

$$= \mathop{\mathbb{E}}_{a_t \sim \pi(s_t), s_{t+1} \in \mathcal{S}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \middle| s_0 = s, a_0 = a \right]$$

Connection to the state-value function

$$V^{\pi}(s) = \sum_{a \in \mathcal{A}(s)} \pi(s, a) Q^{\pi}(s, a)$$

Optimal Q function

Definition

The *optimal action-value function* $Q^*(s, a)$ is the minimum action-value function over all policies, i.e.

$$Q^*(s, a) := \min_\pi Q^\pi(s, a).$$

Connection to the optimal state-value function

$$V^*(s) = \min_{a \in \mathcal{A}(s)} Q^*(s, a)$$

Bellman equation in Q

Derivation

$$
\begin{aligned}
Q^{\pi}(\mathrm{s},\mathrm{a}) &:= \mathop{\mathbb{E}}_{a_t \sim \pi(s_t), s_{t+1} \in \mathcal{S}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \middle| s_0 = \mathrm{s}, a_0 = \mathrm{a} \right] \\
&= \mathop{\mathbb{E}}_{s_1 \in \mathcal{S}} \left[r(\mathrm{s},\mathrm{a},s_1) + \gamma \mathop{\mathbb{E}}_{a_t \sim \pi(s_t), s_{t+1} \in \mathcal{S}} \sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t, s_{t+1}) \right] \\
&= \mathop{\mathbb{E}}_{s_1 \in \mathcal{S}} \left[r(\mathrm{s},\mathrm{a},s_1) + \gamma \mathop{\mathbb{E}}_{a_1 \sim \pi(s_1)} Q^{\pi}(s_1, a_1) \right]
\end{aligned}
$$

Optimal Bellman equation in Q^*

Derivation

$$
\begin{aligned}
Q^*(s,a) &:= \min_{a_t \in \mathcal{A}(s_t)} \mathop{\mathbb{E}}_{s_{t+1} \in \mathcal{S}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \middle| s_0 = s, a_0 = a \right] \\
&= \min_{a_t \in \mathcal{A}(s_t)} \mathop{\mathbb{E}}_{s_1 \in \mathcal{S}} \left[r(s,a,s_1) + \gamma \mathop{\mathbb{E}}_{s_{t+1} \in \mathcal{S}} \sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t, s_{t+1}) \right] \\
&= \mathop{\mathbb{E}}_{s_1 \in \mathcal{S}} \left[r(s,a,s_1) + \gamma \min_{a_t \in \mathcal{A}(s_t)} \mathop{\mathbb{E}}_{s_{t+1} \in \mathcal{S}} \sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t, s_{t+1}) \right] \\
&= \mathop{\mathbb{E}}_{s_1 \in \mathcal{S}} \left[r(s,a,s_1) + \gamma \min_{a_1 \in \mathcal{A}(s_1)} Q^*(s_1, a_1) \right]
\end{aligned}
$$

Applications of Q functions

Retrieval of an optimal policy

Given the optimal state-action value function Q^*, an optimal policy π^* is constructed as

$$\pi^*(s) := \operatorname*{argmin}_{a \in \mathcal{A}} Q^*(s, a)$$

Construction of a greedy policy w.r.t. a given Q

Given a state-action value function Q, an optimal policy π is constructed as

$$\pi(s) := \operatorname*{argmin}_{a \in \mathcal{A}} Q(s, a)$$

Outline

Approximation of return in Q^π

Motivation

$$Q^\pi(s, a) = \mathbb{E}_\pi\Big[R^\pi(s, a)\Big]$$
$$= \mathop{\mathbb{E}}_{s' \in \mathcal{S}}\Big[r(s, a, s') + \gamma \mathop{\mathbb{E}}_{a' \sim \pi(s')} Q^\pi(s', a')\Big]$$

TD target

$$R^\pi(s, a) \approx r(s, a, s') + \gamma Q^\pi(s', a')$$

Learning the Q^π function

$$\ldots s_k \overset{r_k}{\curvearrowright} a_k \to s_{k+1} \overset{r_{k+1}}{\curvearrowright} a_{k+1} \to s_{k+2} \ldots \to s_\infty$$

One sweep

$$\ldots, s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}, \ldots$$

Goal

Estimate Q^π for the current behavior policy π

SARSA

Key steps

1. Pick a start state $s \in \mathcal{S}$
2. Repeat for each step in episode
   (i) Take action $a \sim \pi(s)$ reaching s', to observe $r(s,a,s')$
   (ii) Choose $a' \sim \pi(s')$
   (iii) Set $Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r(s,a,s') + \gamma Q(s',a') - Q(s,a)\Big)$

Properties of SARSA

Convergence

- By following an ϵ greedy policy, as long as all state-action pairs are visited an infinite number of times, SARSA converges with probability one to an optimal policy and action-value function
- The policy converges in the limit to the greedy policy

Question

Is SARSA on-policy or off-policy?

Approximation of return in Q^*

Observation

For an optimal policy π^*, we have

$$Q^{\pi^*}(s,a) := \mathbb{E}_{\pi^*}[R^{\pi^*}(s,a)]$$
$$= \mathop{\mathbb{E}}_{s' \in \mathcal{S}}\left[r(s,a,s') + \gamma \min_{a' \in \mathcal{S}(s')} Q^*(s',a') \right]$$

TD target

$$R^{\pi^*}(s,a) \approx r(s,a,s') + \gamma \min_{a' \in \mathcal{A}(s')} Q^*(s',a')$$

Q-Learning

Key steps

1. Pick a start state $s \in \mathcal{S}$
2. Repeat for each step in episode
   (i) Take action $a \sim \pi(s)$, using a greedy policy π derived from Q, reaching s' to observe $r(s,a,s')$
   (ii) Choose $a' \sim \pi(s')$ using a greedy policy π derived from Q
   (iii) Set

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r(s,a,s') + \gamma \min_{a' \in \mathcal{A}(s')} Q(s',a') - Q(s,a))$$

Properties of the Q learning

Convergence

Q learning converges with probability one to an optimal policy and action-value function, as long as all state-action pairs are visited enough

Question

Is Q-learning on-policy or off-policy?

On- or off-policy

Update rule [SARSA]

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r(s,a,s') + \gamma Q(s', \pi(s')) - Q(s,a)\Big)$$

Update rule [Q learning]

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r(s,a,s') + \gamma \min_{a' \in \mathcal{A}(s')} Q(s',a') - Q(s,a))$$

Discussion

Szepesvári

If the algorithm estimates the value function of the policy generating the samples, the method is called on-policy; otherwise off-policy.

Expected SARSA

Update rule

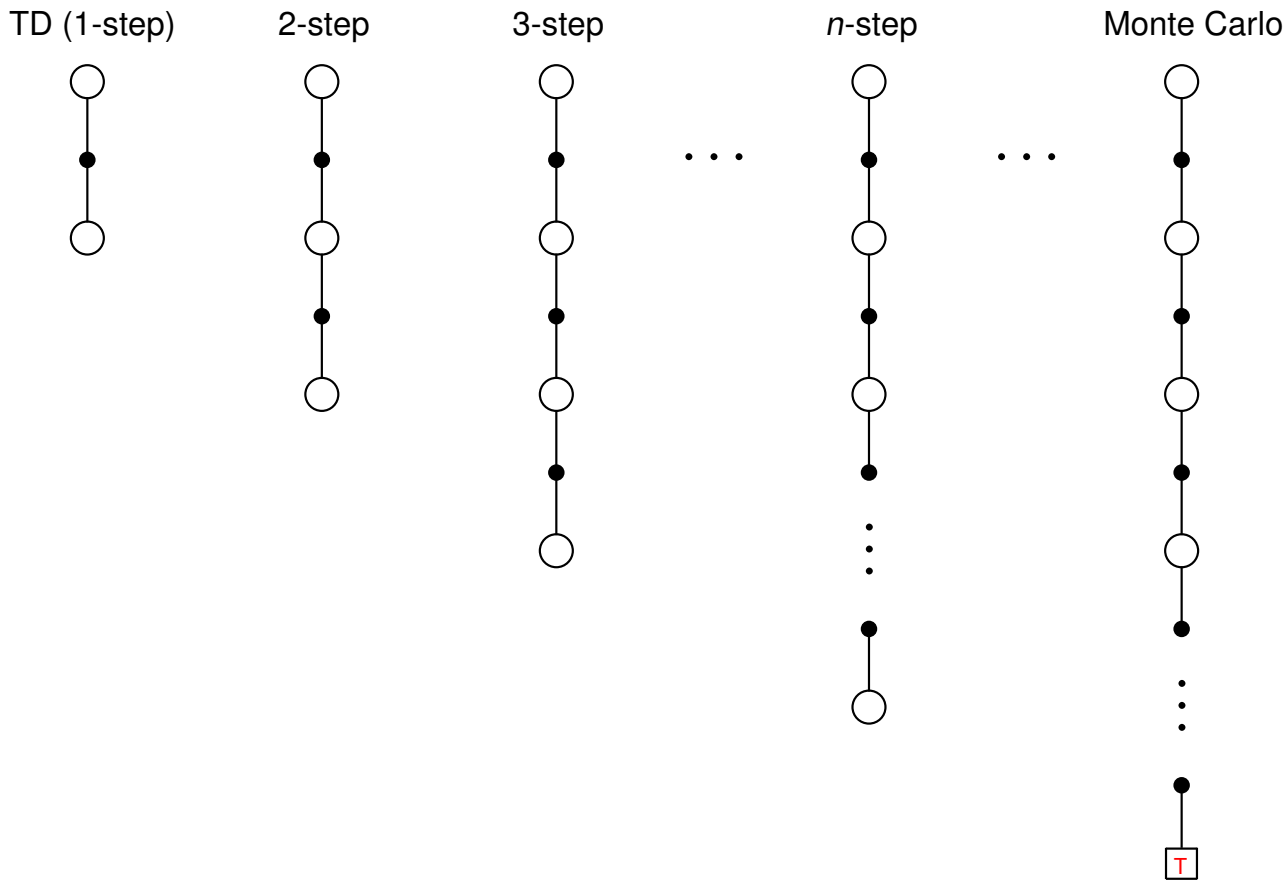
$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r(s,a,s') + \gamma \sum_{a' \sim \pi(s')} \pi(s',a')Q(s',a') - Q(s,a)\Big)$$

Question

Is expected SARSA on-policy or off-policy?

Outline

A pictorial overview



1-step return

The state-value function

$$V^{\pi}(s) := \mathbb{E}_{\pi}\Big[R^{\pi}(s)\Big]$$
$$= \mathop{\mathbb{E}}_{a\sim\pi(s),s'\in\mathcal{S}}\Big[r(s,a,s') + \gamma V^{\pi}(s')\Big]$$

Update rule (TD return)

$$R^{(1)}(s) \leftarrow r(s,a,s') + \gamma V(s')$$

2-step return

The state-value function

$$
\begin{aligned}
V^{\pi}(s) &:= \mathbb{E}_{\pi}\Big[R^{\pi}(s)\Big] \\
&= \mathop{\mathbb{E}}_{a\sim\pi(s),s'\in\mathcal{S}}\Big[r(s,a,s') + \gamma V^{\pi}(s')\Big] \\
&= \mathop{\mathbb{E}}_{\substack{a\sim\pi(s),s'\in\mathcal{S} \\ a'\sim\pi(s'),s''\in\mathcal{S}}}\Big[r(s,a,s') + \gamma r(s',a',s'') + \gamma^2 V^{\pi}(s'')\Big]
\end{aligned}
$$

Update rule (2-step return)

$$R^{(2)}(s) \leftarrow r(s,\pi(s),s') + \gamma r(s',\pi(s'),s'') + \gamma^2 V(s'')$$

n-step return

The state-value function

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{\substack{a_t \sim \pi(s_t), s_{t+1} \in \mathcal{S} \\ t=0,1,\ldots,n-1}} \left[\sum_{t=0}^{n-1} \gamma^t r(s_t, a_t, s_{t+1}) + \gamma^n V^{\pi}(s_n) \middle| s_0 = s \right]$$

Update rule (n-step return)

$$R^{(n)}(s) \leftarrow \underbrace{\sum_{t=0}^{n-1} \gamma^t r(s_t, \pi(s_t), s_{t+1})}_{\textit{data}} + \underbrace{\gamma^n V(s_n)}_{\textit{estimate}}$$

MC return

The state-value function

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{\substack{a_t \sim \pi(s_t), s_{t+1} \in \mathcal{S} \\ t=0,1,\ldots,T}} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t, s_{t+1}) \middle| s_0 = s \right]$$

Update rule (MC return)

$$R^{(MC)}(s) \leftarrow \sum_{t=0}^{T} \gamma^t r(s_t, a_t, s_{t+1})$$

n-step TD updates

Idea

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[R^{(n)}(s)\right]$$

Update rule

A naïve n-step update is given as

$$V(s) \leftarrow V(s) + \Delta^{(n)} V(s)$$

where $\Delta^{(n)} V(s) = \alpha\left(R^{(n)}(s) - V(s)\right)$

On-line vs. off-line

On-line

Update during episodes, as soon as the increments are computed

$$V(s) \leftarrow V(s) + \Delta^{(1)} V(s)$$

“Off-line”/Small batch on-line

Update, until the end of episode

$$V(s) \leftarrow V(s) + \sum_{t=1}^{T} \Delta^{(t)} V(s)$$

Note, the increments are not used to change value estimates until the end of the episode

Error reduction property of *n*-step backups

Result

For any given V, the expected value of the n-step return using V is guaranteed to be a better estimate of V^π than V is

$$\underbrace{\left\| \mathbb{E}_\pi \left[R^{(n)} \right] - V^\pi \right\|_\infty}_{\text{maximum error using N-step return}} \leq \underbrace{\gamma^n \left\| V - V^\pi \right\|_\infty}_{\text{Maximum error using } V}$$

Outline

Averaging *n*-step returns

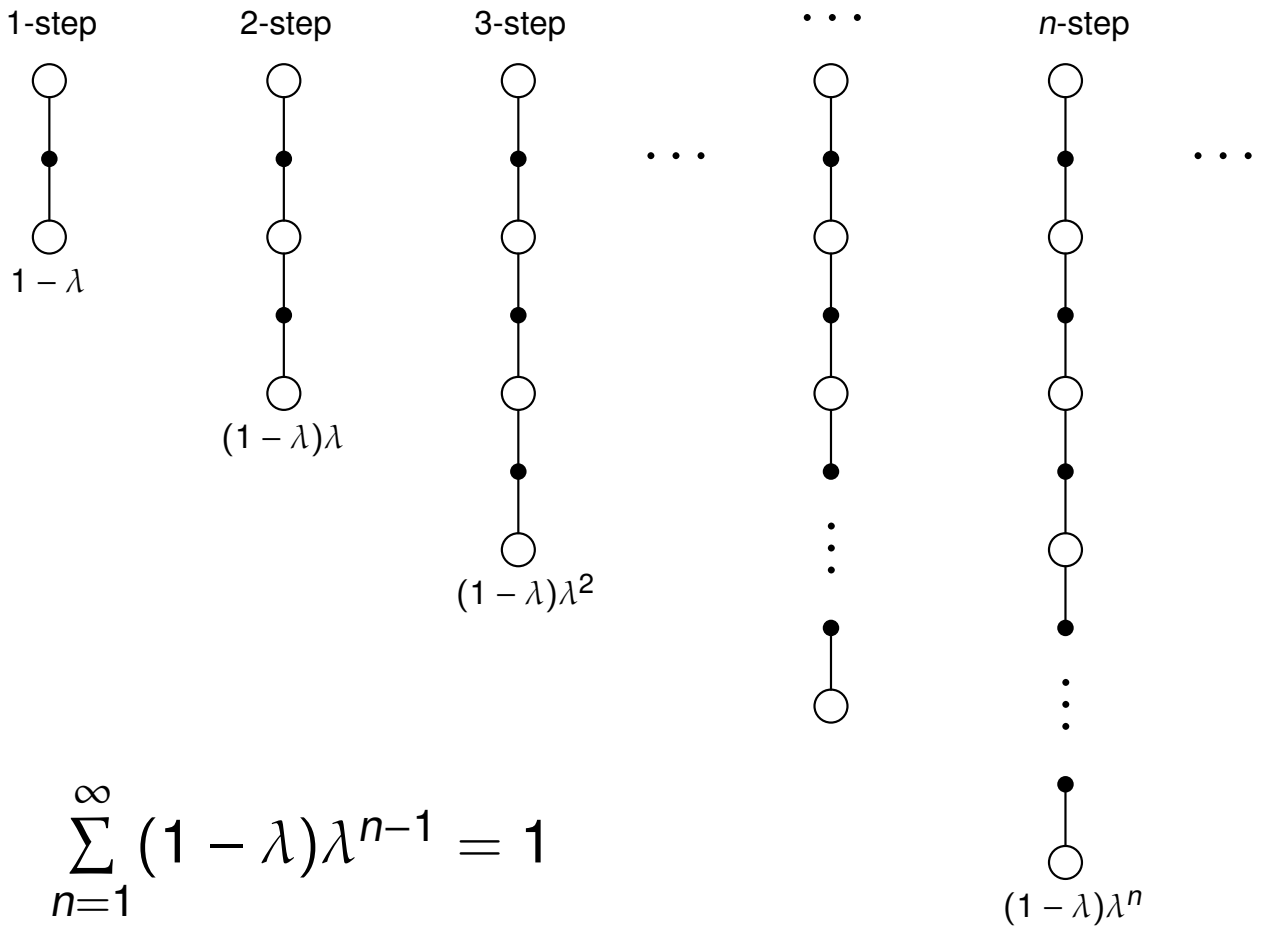
Idea

Backups can be done not just towards any n-step return, but towards any average of n-step returns, i.e.

$$R^{avg}(s) = \sum_{n=1}^{\infty} \lambda_n R^{(n)}(s)$$

with $\lambda_n > 0$ *and* $\sum_{n=1}^{\infty} \lambda_n = 1$

A pictorial overview of λ-return



$$\sum_{n=1}^{\infty} (1-\lambda)\lambda^{n-1} = 1$$

$TD(\lambda)$ backup

Idea

$TD(\lambda)$ is a particular method for averaging all n-step backups

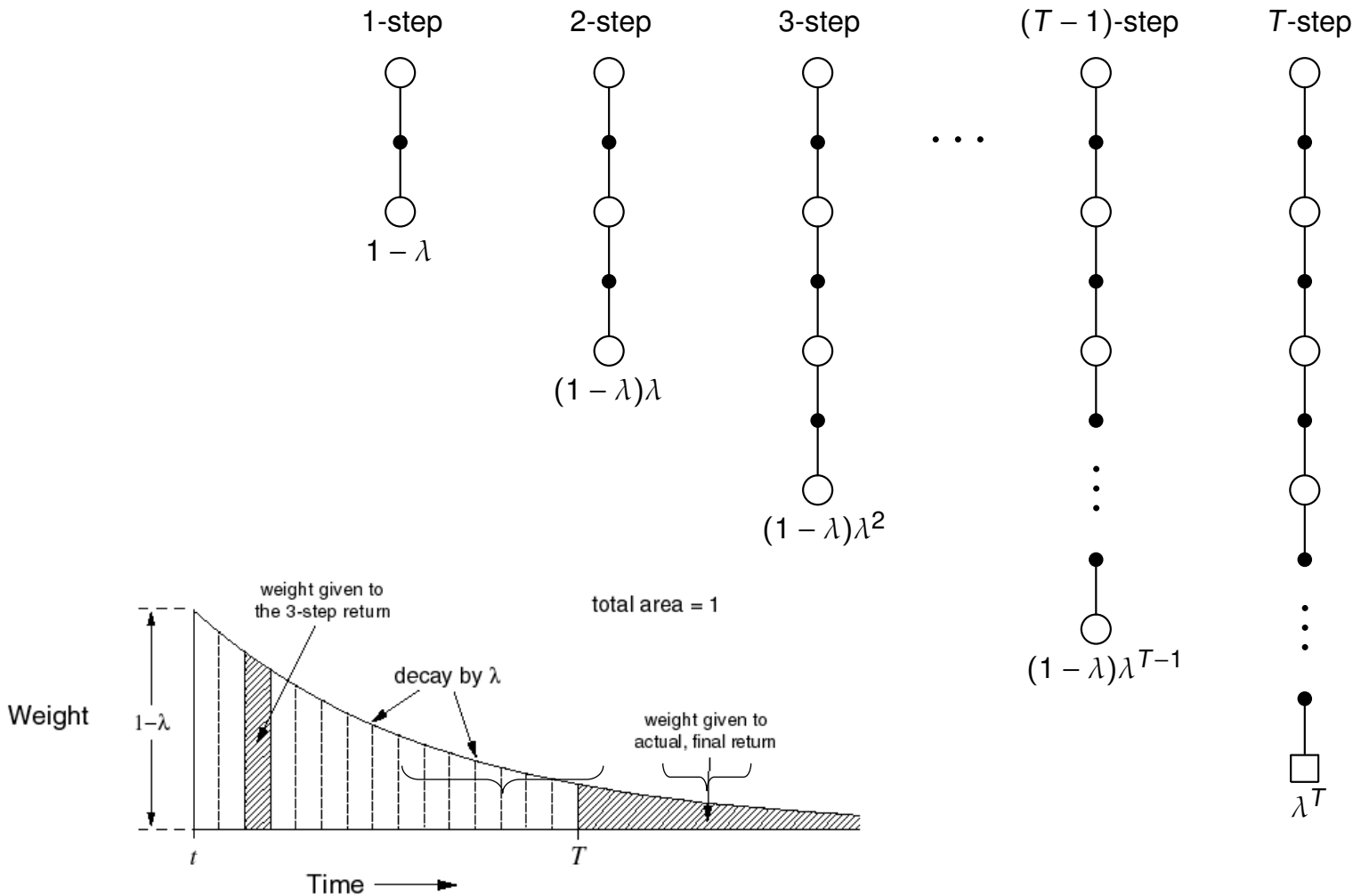
$$R^{[\lambda]}(s) := (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}R^{(n)}(s)$$

where $0 \leq \lambda \leq 1$ *with* $(1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1} = 1$

Backup using λ-return

$$\Delta V(s) = \alpha\left(R^{[\lambda]}(s) - V(s)\right)$$

A pictorial overview of λ-return



λ-return weighting function

Recall

$$R^{(n)}(s) = \sum_{t=0}^{n-1} \gamma^t r(s_t, \pi(s_t), s_{t+1}) + \gamma^n V(s_n)$$

Fact

After reaching the (virtual/algorithmic) terminal T, all subsequent T-step returns are equal to $R_T(s)$, i.e.

$$R^{(n)}(s) = R_T(s), \quad \text{for all } n \geq T$$

λ-return weighting function

Result

$$R^{[\lambda]}(s) = \underbrace{(1-\lambda)\sum_{n=1}^{T-1}\lambda^{n-1}R^{(n)}(s)}_{\text{until termination}} + \underbrace{\lambda^{T}R_{T}(s)}_{\text{after termination}}$$

Question

What if $\lambda = 0$? Or $\lambda = 1$?

Relation to $TD(0)$ and MC

$\lambda = 0$: $TD(0)$

$$R^{[0]}(s) = (1-0)\sum_{n=1}^{T-1} 0^{n-1} R^{(n)}(s) + 0^{T} R_T(s) = R^{(1)}(s)$$

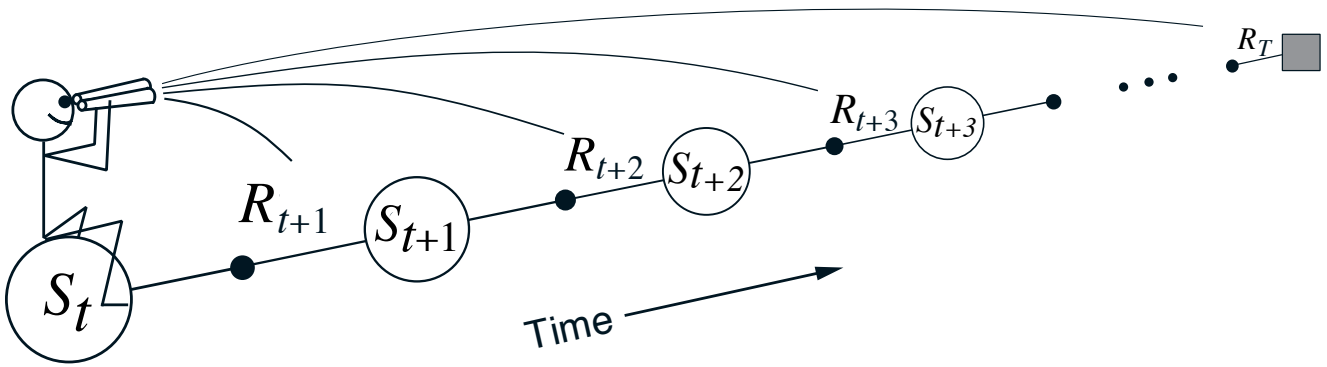
$\lambda = 1$: almost MC, but more (virtual terminal)

$$R^{[1]}(s) = (1-1)\sum_{n=1}^{T-1} 1^{n-1} R^{(n)}(s) + 1^{T} R_T(s) = R_T(s)$$

A forward $TD(\lambda)$

Summary

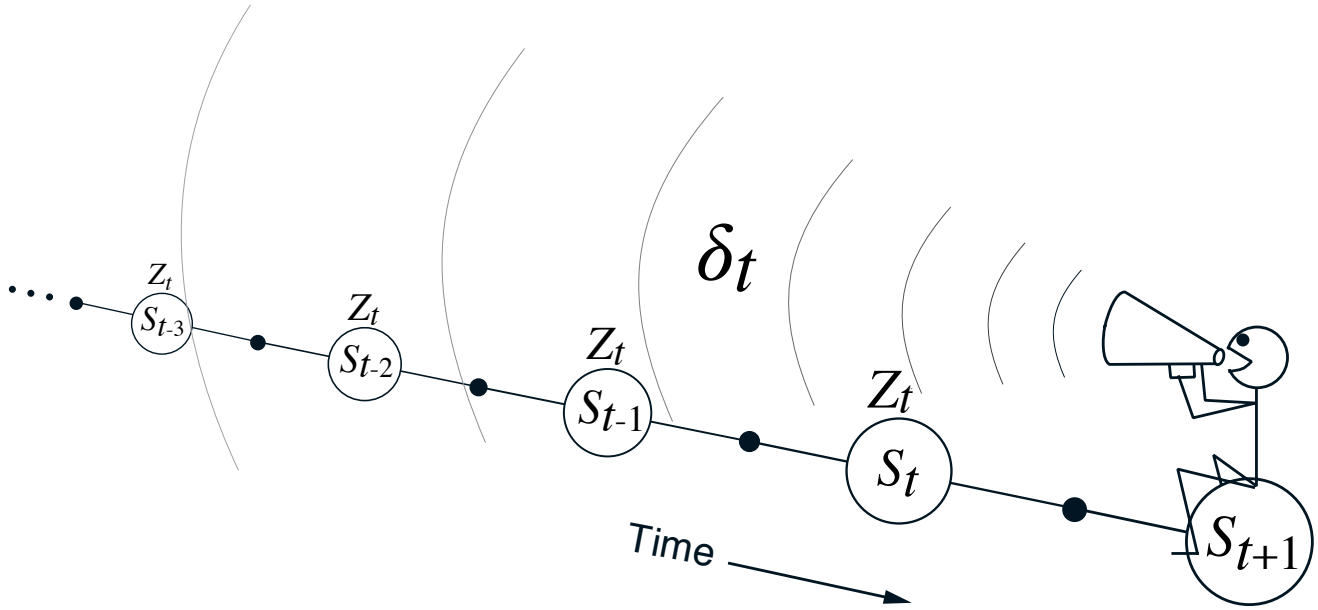
$TD(\lambda)$ averages all n-step backups. It looks forward from each state to determine update from future states and rewards



- The forward view is for theory, or off-line
- The forward view is not implementable for on-line

Outline

An alternative: propagate backwards



Idea

Shout the 1-step TD error backwards over time. The strength of the message decreases with temporal distance

Update with eligibility trace

- Let us denote an eligibility trace by $e_t(s) \geq 0$
- Let $\delta(s_t, a_t, s_{t+1})$ be the TD error

Update

1. $V(s_t) \leftarrow V(s_t) + \alpha\delta(s_t, a_t, s_{t+1})$
2. $V(s_{t-1}) \leftarrow V(s_{t-1}) + \alpha e_t(s_{t-1})\delta(s_t, a_t, s_{t+1})$
3. $V(s_{t-2}) \leftarrow V(s_{t-2}) + \alpha e_t(s_{t-2})\delta(s_t, a_t, s_{t+1})$
4. ...

Eligibility trace

Idea

Each state is associated with an additional memory variable, i.e. eligibility trace, defined by

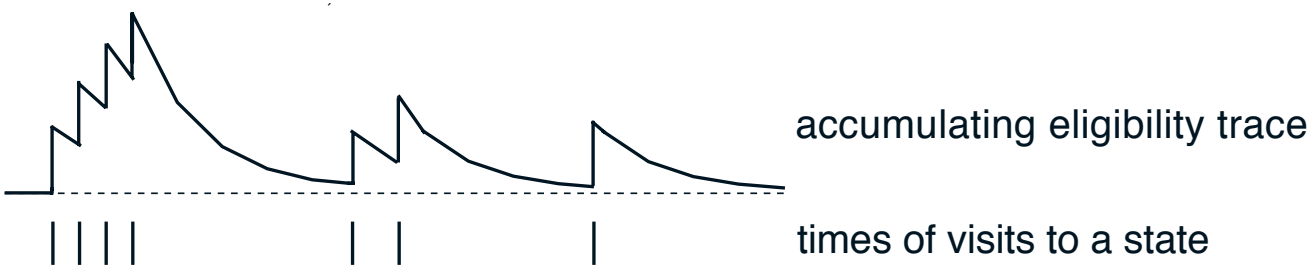
$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & \text{if } s_t \neq s \\ \gamma\lambda e_{t-1}(s) + 1 & \text{if } s_t = s \end{cases}$$

where $0 \leq \lambda \leq 1$ *is a trace-decay parameter .*

Comment

Different λ than the forward TD

Problem with accumulative traces



Observation

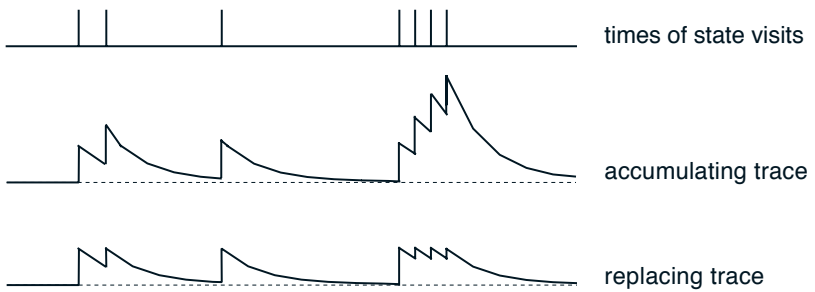
Using accumulative traces, frequently visited states can have eligibilities greater than one.

Replacing eligibility trace

Idea

Instead of adding one when you visit a state, set that trace to one, i.e.

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & \text{if } s_t \neq s \\ 1 & \text{if } s_t = s \end{cases}$$



Backward $TD(\lambda)$ prediction

Update rule

A backward $TD(\lambda)$ backup is given as

$$V(s) \leftarrow V(s) + \Delta V_t(s)$$

- Update $\Delta V_t(s) = \alpha e_t(s)\delta(s_t, a_t, s_{t+1})$
- The 1-step TD error

$$\delta(s_t, a_t, s_{t+1}) = r(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1}) - V(s_t)$$

Backward $TD(0)$

Eligibility trace at $\lambda = 0$

$$e_t(s) = \begin{cases} 0 & \text{if } s_t \neq s, \\ 1 & \text{if } s_t = s; \end{cases} \qquad \Delta V_t(s) = \begin{cases} 0 & \text{if } s_t \neq s \\ \alpha\delta_t & \text{if } s_t = s \end{cases}$$

Update rule

For $s_t = s$, update

$$V(s) \leftarrow V(s) + \alpha\Big(r(s,a,s') + \gamma V(s') - V(s)\Big)$$

Backward $TD(1)$

Eligibility trace at $\lambda = 1$

$$e_t(s) = \begin{cases} \gamma e_{t-1}(s) & \text{if } s_t \neq s \\ 1 & \text{if } s_t = s \end{cases}$$

Update rule

For all state s in the episode, update

$$V(s) \leftarrow V(s) + \alpha e_t(s) \delta(s_t, a_t, s_{t+1})$$

with $\delta(s_t, a_t, s_{t+1}) = r(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1}) - V(s_t)$

Forward $TD(\lambda)$ = Backward $TD(\lambda)$

Update rule of off-line $TD(\lambda)$

$$V(s) \leftarrow V(s) + \sum_{t=0}^{T-1} \Delta V^{\{b,f\}}(s)$$

Result (off-line)

$$\sum_{t=0}^{T-1} \Delta V_t^b(s) = \sum_{t=0}^{T-1} \Delta V_t^f(s)$$

where $\Delta V_t^b(s) = \alpha \delta_t e_t(s)$, and $\Delta V_t^f(s) = \alpha(R_t^{[\lambda]} - V_t(s)) I_{ss_t}$

with I_{ss_t} being the indicator function for $s_t = s$

SARSA(λ)

Idea

Save eligibility for state-action pairs instead of just states

- Eligibility trace for state-action pairs is defined as

$$e_t(s,a) = \begin{cases} 1 & \text{if } s_t = s \text{ and } a_t = a \\ \gamma\lambda e_{t-1}(s,a) & \text{otherwise} \end{cases}$$

- The Q-function update is given by

$$Q(s,a) = Q(s,a) + \alpha e_t(s,a)\delta(s_t, a_t, s_{t+1}, a_{t+1})$$

where

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

SARSA(λ)

Key steps

1. Take action $a \sim \pi(s)$ reaching s', and observe $r(s, a, s')$
2. Choose a' from s' using policy derived from Q
3. Compute $\delta = r(s, a, s') + \gamma Q(s', a') - Q(s, a)$
4. Compute $e(s, a) \leftarrow 1$
5. For all $(\hat{s}, \hat{a}) \in \mathcal{S} \times \mathcal{A}$
   $$Q(\hat{s}, \hat{a}) \leftarrow Q(\hat{s}, \hat{a}) + \alpha\delta e(\hat{s}, \hat{a})$$
   $$e(\hat{s}, \hat{a}) \leftarrow \lambda\gamma e(\hat{s}, \hat{a})$$
6. Update $s \leftarrow s'$ and $a \leftarrow a'$

TD $Q(\lambda)$ learning

Update rule

$$e_t(s,a) = \begin{cases} 1 & \text{if } (s_t, a_t) = (s,a) \text{ and} \\ & \quad a_t = \operatorname{argmin}_{a'} Q(s_t, a') \\ \gamma\lambda e_{t-1}(s,a) & \text{otherwise} \end{cases}$$

Problem

If every state-action pair is marked as eligible, the update is done over non-greedy policy

Watkins' idea

idea

Zero out eligibility trace after a non-greedy action.

Eligibility trace

$$e_t(s,a) = \begin{cases} 1 & \text{if } (s_t, a_t) = (s,a) \text{ and} \\ & \quad a_t = \operatorname{argmin}_{a'} Q(s_t, a') \\ 0 & \text{if } a_t \neq \operatorname{argmin}_{a'} Q(s_t, a') \\ \gamma\lambda e_{t-1}(s,a) & \text{otherwise} \end{cases}$$

Watkins' eligibility trace

Update rule [Q learning]

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r(s,a,s') + \gamma \min_{a' \in \mathcal{A}(s')} Q(s',a') - Q(s,a))$$

Illustration

	state s_t			
Greedy action a_t	0	0	0	0
	e_t	e_t	1	e_t
	0	0	0	0

Watkins' TD $Q(\lambda)$

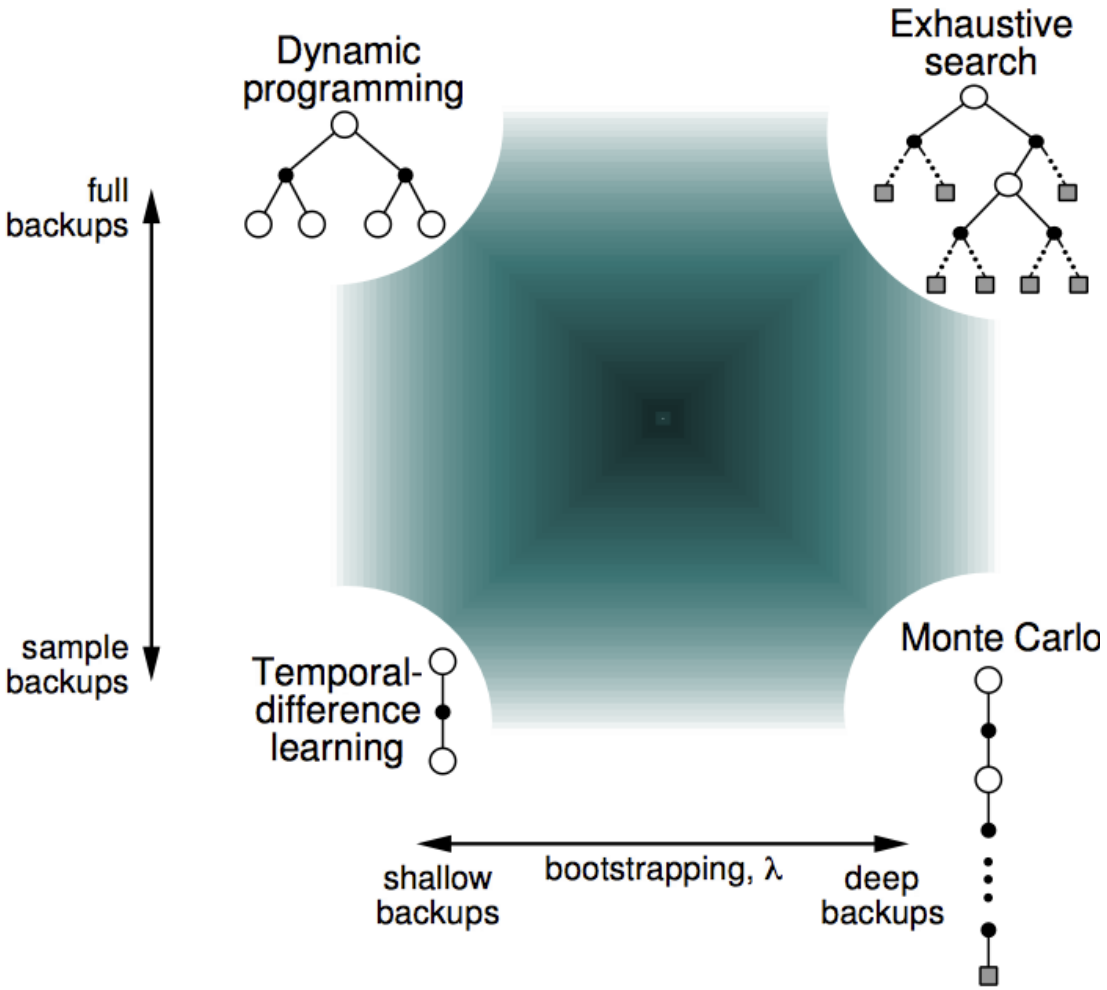
Key steps

1. Take action $a \sim \pi(s)$ reaching s', observe $r(s, a, s')$
2. Choose a' from s' using policy derived from Q
3. Choose $a^* = \underset{a''}{\operatorname{argmin}}\, Q(s', a'')$
4. Compute $\delta = r + \gamma Q(s', a^*) - Q(s, a)$
5. Compute $e(s, a) \leftarrow 1$
6. For all $(\hat{s}, \hat{a}) \in \mathcal{S} \times \mathcal{A}$
   $Q(\hat{s}, \hat{a}) \leftarrow Q(\hat{s}, \hat{a}) + \alpha\delta e(\hat{s}, \hat{a})$
   If $a' = a^*$, then $e(\hat{s}, \hat{a}) \leftarrow \lambda\gamma e(\hat{s}, \hat{a})$
   else $e(\hat{s}, \hat{a}) \leftarrow 0$
7. Update $s \leftarrow s'$ and $a \leftarrow a'$

TD does bootstrapping & sampling

- Bootstrapping: update involves an estimate of V or Q
  - MC does not bootstrap
  - DP bootstraps
  - TD bootstraps
- Sampling: update is based on one path through the state space
  - MC samples
  - DP does not sample
  - TD samples

A pictorial overview



Part V

Day Five: Approximate RL Methods

Large-scale RL problems

- Large-scale RL problems, e.g.
  1. Backgammon: $\sim 10^{20}$ states
  2. Game of Go: $\sim 10^{170}$ states
  3. Mountain car: continuous state space
- Value functions are represented by a lookup table
  1. Every state s has an entry $V(s)$
  2. Or every state-action pair (s, a) has an entry $Q(s, a)$

Problems with large-scale RL problems

Curse of dimensionality

- There are too many states and/or actions to store in memory
- It is too slow to learn the value of each state individually

Outline

Value Function Approximation

Approximate Dynamic Programming

Approximate RL with Linear Function Approximation

Outline

Value function approximation

Optimal policy

$$\pi^*(s) := \operatorname*{argmin}_{a \in \mathcal{A}} \mathop{\mathbb{E}}_{s' \in \mathcal{S}} [r(s, a, s') + \gamma V^*(s')]$$

General idea

Let $\mathcal{V} \subset \mathbb{R}^{|\mathcal{S}|}$ be some value function space. The goal is to find an approximation $V \in \mathcal{V}$ of the optimal value function V^*, which may not belong to $\mathcal{V}$.

Policy from value function approximation

Derived greedy policy

Let us define the policy π be greedy w.r.t. $V \in \mathcal{V}$ that is an approximation of the optimal value function V^*, i.e.

$$\pi(s) \in \operatorname*{argmin}_{a \in \mathcal{A}} \mathop{\mathbb{E}}_{s' \in \mathcal{S}} [r(s, a, s') + \gamma V(s')]$$

Question

How good is the approximate policy π?

Or, how good is V^π

Bound on the performance loss

Theorem 1

Let V be an approximation of V^*, and π be the policy greedy w.r.t. V. Then

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

Question

Is this result good?

Bellman residual

Definition

The Bellman residual of a function V is defined as

$$\mathcal{T}^* V - V$$

Fact

The Bellman residual of V^* is equal to zero, i.e.

$$\mathcal{T}^* V^* - V^* = 0$$

Bellman residual

Idea

We could minimize the residual $\mathcal{T}^* V - V$ for some norm

Question

If a function $V \in \mathcal{V}$ has a low $||\mathcal{T}^* V - V||_\infty$, then is V close to V^*?

Performance bound of the Bellman residual

Theorem 2.1

Let V be an approximation of V^*, then

$$\|V - V^*\|_\infty \leq \frac{1}{1-\gamma} \|\mathcal{T}^* V - V\|_\infty$$

Theorem 2.2

Let V be an approximation of V^* and π be the policy greedy w.r.t. V. Then

$$\|V^\pi - V^*\|_\infty \leq \frac{2}{1-\gamma} \|\mathcal{T}^* V - V\|_\infty$$

Discussion

Question

Is value function approximation really useful?

Minimizer of the Bellman residual

Definition

Given a value function space $\mathcal{V}$, we search for the function $V \in \mathcal{V}$ with minimum Bellman residual:

$$V_B := \underset{V \in \mathcal{V}}{\operatorname{argmin}} \|\mathcal{T}^* V - V\|_\infty$$

Property of Bellman residual Minimizer

Theorem 3

Let $V \in \mathcal{V}$ be an approximation of V^* and π be the policy greedy w.r.t. V. Then

$$\|V^{\pi_B} - V^*\|_\infty \leq \frac{2(1+\gamma)}{1-\gamma} \inf_{V \in \mathcal{V}} \|V - V^*\|_\infty$$

Comments

Minimizing the Bellman residual in $\mathcal{V}$ makes sense, only when $\mathcal{V}$ is rich enough.

Outline

Approximate policy iteration

Pseudocode (API)

Choose any initial policy π_0, iterate

1. *Policy evaluation:*

   Compute an approximation V_k of V^{π_k}

2. *Policy improvement:*

   Compute π_{k+1} greedy w.r.t. V_k, i.e.

$$\pi_{k+1}(s) \in \operatorname*{argmin}_{a \in \mathcal{A}} \mathbb{E}_{s' \in \mathcal{S}} [r(s, a, s') + \gamma V_k(s')]$$

Performance bound for API

Theorem 5

The performance loss $\|V^{\pi_n} - V^*\|_\infty$ resulting from using the policy π_n greedy w.r.t. V_n is bounded as

$$\limsup_{k\to\infty} \|V^{\pi_k} - V^*\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2} \limsup_{k\to\infty} \|V^{\pi_k} - V_k\|_\infty$$

Comments

If V^π can be well approximated at each iteration, then the performance of the resulting policies will be close to the optimum.

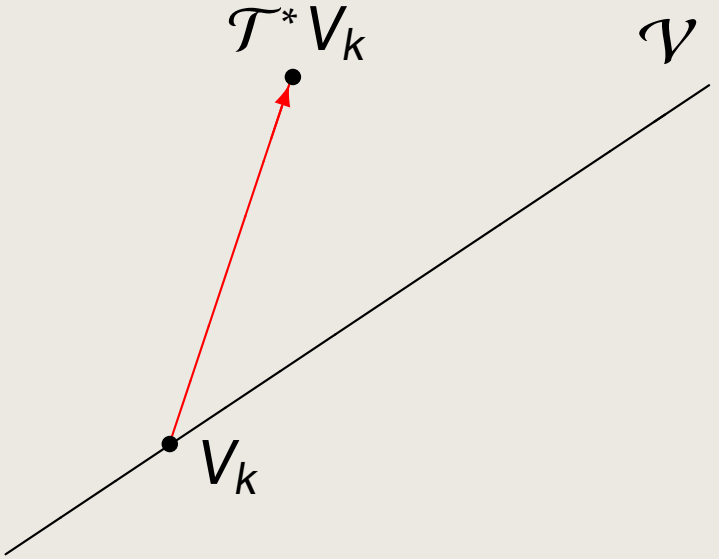
Approximate value iteration

Pseudocode (AVI)

Iterate

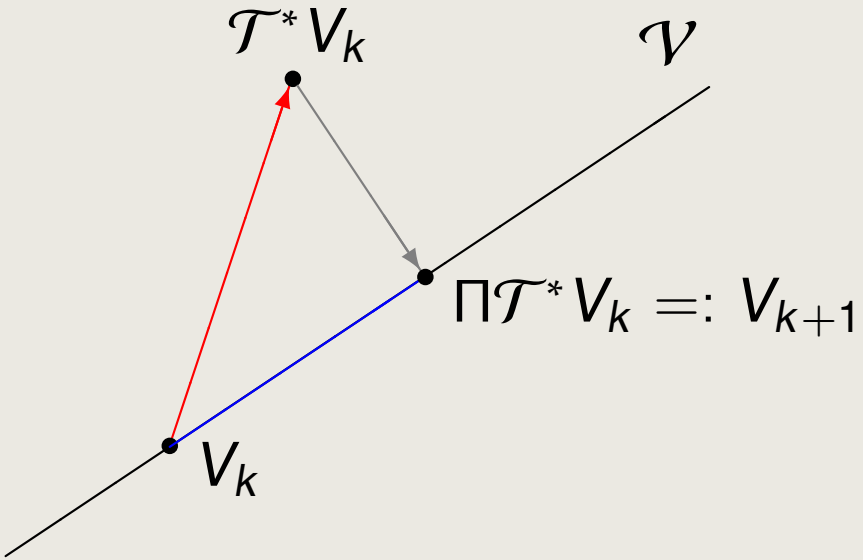
$$V_{k+1} = \mathcal{T}^* V_k$$

Illustration



Fitted approximate value iteration

Illustration



Pseudocode (FAVI)

Iterate $V_{k+1} = \Pi\mathcal{T}^* V_k$

Performance bound for FAVI

Theorem 4

Let us apply AVI for n iterations. Then the performance loss $\|V^{\pi_n} - V^*\|_\infty$ resulting from using the policy π_n greedy w.r.t. V_n is bounded as

$$\|V^{\pi_n} - V^*\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2} \min_{0 \leq k < n} \|\mathcal{T}^* V_k - V_{k+1}\|_\infty + \frac{2\gamma^{n+1}}{1-\gamma} \|V^* - V_0\|_\infty$$

Performance bound for FAVI

Theorem 5

Given an arbitrary $V_0 \in \mathcal{V}$, FAVI converges to $\widetilde{V}^*$, i.e.

$$\widetilde{V}^* = \Pi \mathcal{T}^* \widetilde{V}^*.$$

By denoting the policy $\widetilde{\pi}^*$ greedy w.r.t. $\widetilde{V}^*$, the performance loss $\|V^{\pi_n} - V^*\|_\infty$ is bounded as

$$\|V^{\widetilde{\pi}^*} - V^*\|_\infty \leq \frac{2}{(1-\gamma)^2} \inf_{V \in \mathcal{V}} \|V - V^*\|_\infty$$

Comments

FAVI works well, only when $\mathcal{V}$ is rich enough.

Outline

Parametric value function approximation

Idea

Estimate value functions with function approximation

$$V_\theta(s) \approx V^\pi(s)$$

$$\text{or } Q_\theta(s,a) \approx Q^\pi(s,a)$$

- Generalization from seen states to unseen states
- Only the parameter vector $\theta \in \mathbb{R}^m$ is to be learned

Linear value function approximation

Idea

- Represent states by a feature vector

$$\phi : \mathcal{S} \to \mathbb{R}^m$$
$$\phi(s) := (\phi_1(s), \ldots, \phi_m(s))^\top$$

- Represent value function by a linear combination of features

$$V_\theta(s) := (\phi(s))^\top \theta$$

Approximate linear value function space (ALVFS)

Definition

Let us denote $\Phi := [\phi(s_1), \ldots, \phi(s_{|\mathcal{S}|})] \in \mathbb{R}^{m \times |\mathcal{S}|}$, then we define the linear value function space by

$$\mathcal{V} := \left\{ \Phi^\top \theta \middle| \theta \in \mathbb{R}^m \right\}$$

Goal

Given samples $\{(s_t, a_t, r(s_t, a_t, s_{t+1}))\}_{t=1}^{\infty}$, *and a chosen ALVFS* $\mathcal{V}$, *find a parameter* $\theta \in \mathbb{R}^m$ *such that* $V_\theta \in \mathcal{V}$ *is as close as possible to* V^π

Tabular lookup features

- Lookup table is a special case of linear value function approximation
- Using tabular lookup features

$$\phi^{table}(s) := (\mathbf{1}_{s_1}(s), \ldots, \mathbf{1}_{s_n}(s))^\top$$

where the indicator function $\mathbf{1}_{s_i} : \mathcal{S} \to \{0, 1\}$ is defined as

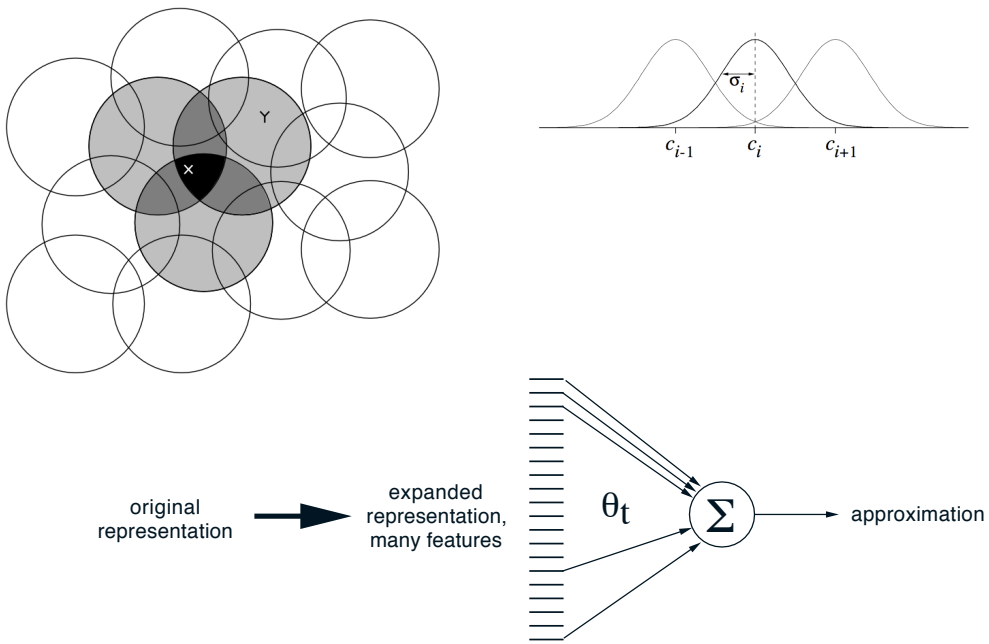
$$\mathbf{1}_{s_i}(s) := \begin{cases} 1, & \text{if } s = s_i \\ 0, & \text{otherwise.} \end{cases}$$

- Parameter vector θ gives value of each individual state

$$V_\theta(s) = \left(\phi^{table}(s)\right)^\top \theta$$

Coarse coding

- Coarse coding provides large feature vector $\phi(s)$
- Parameter vector θ gives a value to each feature



Performance measure

Mean-squared error (MSE)

$$J(\theta) := \frac{1}{2} \mathop{\mathbb{E}}_{s \in \mathcal{S}} \left[(V^{\pi}(s) - V_{\theta}(s))^2 \right]$$

Problem

We don't know the targets $V^{\pi}(s)$

Backups as training examples

Idea

Approximate by bootstrapping

$$V(s) \approx \mathbb{E}\left[R^{\ddagger}(s)\right]$$

where $\ddagger \in \left\{\textit{MC return}, \textit{n-step TD return}, \textit{TD}(\lambda)\textit{return}\right\}$

Mean-squared error (MSE)

$$J(\theta) := \frac{1}{2} \mathop{\mathbb{E}}_{s \in \mathcal{S}}\left[(V(s) - V_\theta(s))^2\right]$$

Gradient descent algorithm

- Let $f\colon \mathbb{R}^m \to \mathbb{R}$ be differentiable
- The gradient of f is defined as

$$\nabla f(\theta) = \left(\frac{\partial f(\theta)}{\partial \theta_1}, \ldots, \frac{\partial f(\theta)}{\partial \theta_m}\right)^\top$$

Result (gradient descent update)

Iterate

$$\theta \leftarrow \theta - \beta \nabla f(\theta)$$

where $\beta \in \mathbb{R}^+$ is a step size

Stochastic gradient descent

- Let $f\colon \mathbb{R}^m \to \mathbb{R}, \quad f(\theta) := \mathbb{E}_t[g(\theta, s_t)]$
- The gradient update is

$$\theta \leftarrow \theta - \beta \nabla f(\theta) = \theta - \beta \mathbb{E}_t[\nabla g(\theta, s_t)]$$

Idea

Use the sample gradient instead, i.e. given sample s_t,

$$\theta \leftarrow \theta - \beta_t \nabla g(\theta, s_t)$$

“Gradient descent” in RL

- Recall the MSE function as

$$J(\theta) := \frac{1}{2} \mathop{\mathbb{E}}_{s \in \mathcal{S}} \left[(V(s) - V_\theta(s))^2 \right]$$

- The gradient $\nabla J(\theta)$ is computed as

$$\nabla J(\theta) = - \mathop{\mathbb{E}}_{s \in \mathcal{S}} [(V(s) - V_\theta(s)) \nabla V_\theta(s)]$$

- The gradient descent update is given by

$$\theta \leftarrow \theta + \beta \mathop{\mathbb{E}}_{s \in \mathcal{S}} [(V(s) - V_\theta(s)) \nabla V_\theta(s)]$$

with $\nabla V_\theta(s) = \phi(s)$

“Stochastic gradient descent” in RL

- The stochastic gradient descent update is given by

$$\theta \leftarrow \theta + \beta_t (V(s_t) - \theta^\top \phi(s_t)) \phi(s_t)$$

- But $V(s) \approx \mathbb{E}\left[R^\ddagger(s)\right]$
- A more practical stochastic gradient descent update is

$$\theta \leftarrow \theta + \beta_t \left(R^\ddagger(s_t) - \theta^\top \phi(s_t)\right) \phi(s_t)$$

with $\ddagger \in \left\{\text{MC return}, n\text{-step TD return}, \text{TD}(\lambda)\textit{return}\right\}$

- Example: one-step TD

$$\theta \leftarrow \theta + \beta_t \left(r(s_t, a_t, s_{t+1}) + \gamma \theta^\top \phi(s_{t+1}) - \theta^\top \phi(s_t)\right) \phi(s_t)$$

Linear Q function approximation

- Represent pairs of state and action by a feature vector

$$\phi(s,a) = (\phi_1(s,a), \ldots, \phi_m(s,a))^\top$$

- Represent value function by a linear combination of features

$$Q_\theta(s,a) := \theta^\top \phi(s,a)$$

- Minimization of the mean-squared error

$$J(\theta) := \frac{1}{2} \mathop{\mathbb{E}}_{s \in \mathcal{S}, a \in \mathcal{A}} \left[(Q(s,a) - Q_\theta(s,a))^2 \right]$$

Control with value function approximation

Stochastic gradient descent update in Q

$$\theta \leftarrow \theta + \beta_t \left(R^{\ddagger}(s_t, a_t) - \theta^{\top} \phi(s_t, a_t) \right) \phi(s_t, a_t)$$

- For MC control

$$\theta \leftarrow \theta + \beta_t \left(R(s_t) - \theta^{\top} \phi(s_t, a_t) \right) \phi(s_t, a_t)$$

- For $TD(0)$ control

$$\theta \leftarrow \theta + \beta_t \left(r(s_t, a_t, s_{t+1}) + \gamma \theta^{\top} \phi(s_{t+1}, a_{t+1}) - \theta^{\top} \phi(s_t, a_t) \right) \phi(s_t, a_t)$$

- For forward $TD(\lambda)$, backward $TD(\lambda)$...

Properties of linear value function approximation

- $f(\theta)$ with linear function approximation is quadratic, i.e. convex
- Stochastic gradient descent algorithms converge to global optimum[?]
- Update rule is particularly simple, i.e. $\nabla V_\theta(s) = \phi(s)$

Convergence of prediction algorithms

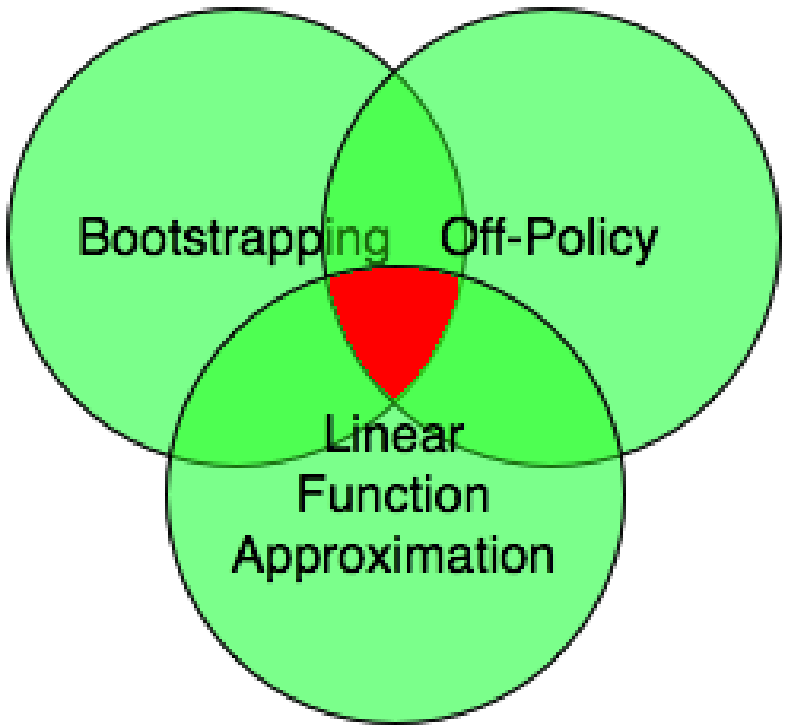
On/Off-Policy	Algorithm	Table Lookup	Linear	Non-Linear
On-Policy	MC	✓	✓	✓
	TD(0)	✓	✓	✗
	TD(λ)	✓	✓	✗
Off-Policy	MC	✓	✓	✓
	TD(0)	✓	✗	✗
	TD(λ)	✓	✗	✗

Convergence of control algorithms

Algorithm	Table Lookup	Linear
Monte-Carlo Control	✓	✓
Sarsa	✓	(✓)
Q-learning	✓	✗

(✓) = arrival at near-optimal value function

Outlook



Ongoing research

Part VI

Day Six: Numerical RL Methods

Outline

Bellman Residual Minimization

Least Squares TD

Gradient TD Learning

Outline

Bellman residual minimization (BRM)

Main idea

Search for a function $V \in \mathcal{V}$ that minimizes the Bellman residual for the policy π, i.e.

$$V_{BR} := \operatorname*{argmin}_{V \in \mathcal{V}} \left\| \mathcal{T}_\pi V - V \right\|$$

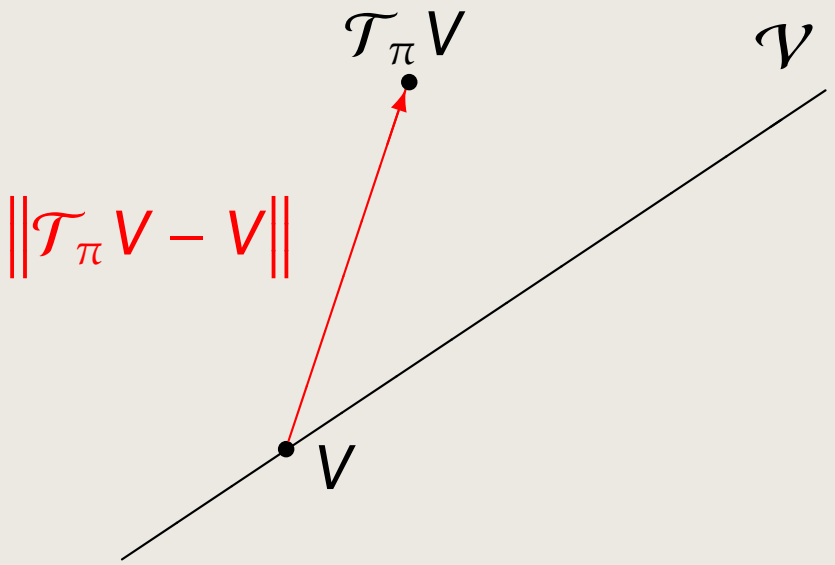
with some norm $\| \cdot \|$

Question

What norm then?

BRM

Illustration



Ergodic assumption

Assumption

Given an MDP model and a policy π, we assume the Markov chain induced by π is *irreducible*, or *ergodic*, namely, any state is reachable from any other states.

Fact

If the Markov chain induced by π is *ergodic*, then there exists a stationary distribution over states as, for arbitrary $\mathbf{s}, \mathbf{s}' \in \mathcal{S}$

$$\mu(\mathbf{s}') := \lim_{t \to \infty} p(s_t = \mathbf{s}' | s_0 = \mathbf{s}) > 0$$

Bellman operator

Let us define $\|x\|_\mu^2 := \langle x, x \rangle_\mu = x^\top \operatorname{diag}(\mu) x$

Theorem 1

The Bellman operator $\mathcal{T}_\pi$ is a contraction of modulus γ with respect to $\|\cdot\|_\mu$, i.e.

$$\left\| \mathcal{T}_\pi V - \mathcal{T}_\pi V' \right\|_\mu \leq \gamma \left\| V - V' \right\|_\mu .$$

The $\ell_2(\mu)$ BRM solution (1)

Problem formulation

Let μ be a stationary distribution for π and V_{BR} be the BRM solution using $\ell_2(\mu)$-norm, i.e.

$$V_{BR} := \operatorname*{argmin}_{V \in \mathcal{V}} \left\| \mathcal{T}_\pi V - V \right\|_\mu^2$$

Recall

The Bellman operator $\mathcal{T}_\pi V_\theta := r^\pi + \gamma P^\pi V_\theta$.

The $\ell_2(\mu)$ BRM solution (2)

Observation

The approximate BRM is quadratic in θ, i.e.

$$\begin{aligned}\left\|\mathcal{T}_\pi V_\theta - V_\theta\right\|_\mu^2 &= \left\|r^\pi + \gamma P^\pi \Phi^\top \theta - \Phi^\top \theta\right\|_\mu^2 \\ &= \left\|r^\pi + (\gamma P^\pi - I)\Phi^\top \theta\right\|_\mu^2\end{aligned}$$

Characterization of solutions

Global minimizer with closed form solution

The $\ell_2(\mu)$ BRM solution (3)

Derivation

The derivative of the BRM vanishes, i.e. for all states i

$$\left\langle r^\pi + \gamma P^\pi \Phi^\top \theta - \Phi^\top \theta, (\gamma P^\pi - I)\phi_i^\top \right\rangle_\mu = 0$$

i.e.

$$\left\langle r^\pi, (\gamma P^\pi - I)\phi_i^\top \right\rangle_\mu + \left\langle (\gamma P^\pi - I)\Phi^\top \theta, (\gamma P^\pi - I)\phi_i^\top \right\rangle_\mu = 0$$

Solution

Solve the linear system $A\theta = b$ for θ with

$$\begin{cases} A &= \Phi(I - \gamma P^\pi)^\top \operatorname{diag}(\mu)(I - \gamma P^\pi)\Phi^\top \\ b &= \Phi(I - \gamma P^\pi)^\top \operatorname{diag}(\mu) r^\pi \end{cases}$$

The $\ell_2(\mu)$ BRM solution (4)

Closed form solution

$$\theta = A^{-1}b$$

Solution

Solve the linear system $A\theta = b$ for θ with

$$\begin{cases} A = \mathbb{E}_\pi\left[(\phi(s) - \gamma\phi(s'))(\phi(s) - \gamma\phi(s'))^\top\right] \\ b = \mathbb{E}_\pi\left[r^\pi(s)(\phi(s) - \gamma\phi(s'))\right] \end{cases}$$

The $\ell_2(\mu)$ BRM online solution

The Sherman-Morrison formula

Let A be an invertible square matrix and u, v are column vectors. Assume that $1 + v^\top A^{-1} u \neq 0$. Then the inverse of the rank one update $A + uv^\top$ is given explicitly by

$$\left(A + uv^\top\right)^{-1} = A^{-1} - \frac{A^{-1} uv^\top A^{-1}}{1 + v^\top A^{-1} u}.$$

Performance bound for the BRM solution

Theorem 2

Let μ be the stationary distribution for π, then we get

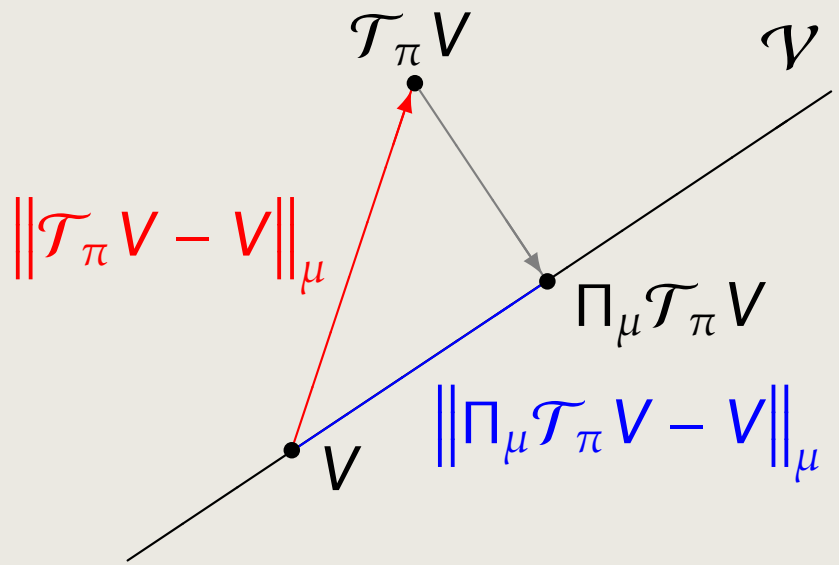
$$\left\| V^{\pi} - V_{BR} \right\|_{\mu} \leq \frac{1+\gamma}{1-\gamma} \inf_{V \in \mathcal{V}} \left\| V^{\pi} - V \right\|_{\mu}$$

Comments

The BRM solution is good, when $\mathcal{V}$ is rich enough to contain V^{π}.

Outline

LSTD (illustration)



Least squares temporal difference

Setting

Consider a linear space $\mathcal{V}$ and Π_μ the projection with norm $\ell_2(\mu)$, where μ is a distribution over s, i.e.

$$\Pi_\mu := \Phi(\Phi^\top \operatorname{diag}(\mu)\Phi)^{-1}\Phi^\top \operatorname{diag}(\mu)$$

Theorem 3

The projected Bellman operator $\Pi_\mu \mathcal{T}_\pi$ is a contraction of modulus γ with respect to $\|\cdot\|_\mu$, i.e.

$$\left\|\Pi_\mu \mathcal{T}_\pi V - \Pi_\mu \mathcal{T}_\pi V'\right\|_\mu \leq \gamma \left\|V - V'\right\|_\mu.$$

Least squares temporal difference

Formulation

Search for a function $V \in \mathcal{V}$ that minimizes the projected Bellman residual for the policy π, i.e.

$$V_{TD} := \operatorname*{argmin}_{V \in \mathcal{V}} \left\| \Pi_\mu \mathcal{T}_\pi V - V \right\|_\mu^2$$

The LSTD solution (1)

Derivation

The Bellman residual $\mathcal{T}_\pi V_{TD} - V_{TD}$ is orthogonal to the space $\mathcal{V}$, i.e. for all states $i = 1, \ldots, |\mathcal{S}|$

$$\left\langle r^\pi + \gamma P^\pi V_{TD} - V_{TD}, \phi_i \right\rangle_\mu = 0$$

i.e.

$$\left\langle r^\pi, \phi_i \right\rangle_\mu + \left\langle \gamma P^\pi \Phi^\top \theta_{TD} - \Phi^\top \theta_{TD}, \phi_i \right\rangle_\mu = 0$$

Solution

Solve the linear system $A\theta = b$ for θ with

$$\begin{cases} A &= \Phi \operatorname{diag}(\mu)(I - \gamma P^\pi)\Phi^\top \\ b &= \Phi \operatorname{diag}(\mu) r^\pi \end{cases}$$

The LSTD solution (2)

Solution

Solve the linear system $A\theta = b$ for θ with

$$\begin{cases} A &= \mathbb{E}_\pi \left[\phi(s)(\phi(s) - \gamma\phi(s'))^\top \right] \\ b &= \mathbb{E}_\pi \left[r^\pi(s)\phi(s) \right] \end{cases}$$

Online algorithm by applying the Sherman-Morrison formula...

Performance bound for the LSTD solution

Theorem 3

Consider μ to be the stationary distribution associated to π. Then $\mathcal{T}_\pi$ is a contraction mapping in $\ell_2(\mu)$ norm, thus $\Pi\mu\mathcal{T}_\pi$ is also a contraction, and there exists a unique LSTD solution V_{TD}. We have the approximation error bound

$$\left\|V^\pi - V_{TD}\right\|_\mu \leq \frac{1}{\sqrt{1-\gamma^2}} \inf_{V \in \mathcal{V}} \left\|V^\pi - V\right\|_\mu$$

Outline

Motivation (1)

Recall: the LSTD solution

Solve the linear system $A\theta = b$ for θ with

$$\begin{cases} A &= \Phi \operatorname{diag}(\mu)(I - \gamma P^{\pi})\Phi^{\top} \\ b &= \Phi \operatorname{diag}(\mu) r^{\pi} \end{cases}$$

Observation

We can rewrite

$$\begin{cases} A &= \mathbb{E}_{\mu}[\phi(\phi - \gamma\phi')^{\top}] \\ b &= \mathbb{E}_{\mu}[r\phi] \end{cases}$$

Motivation (2)

Derivation

We compute

$$\begin{aligned} b - A\theta &= \mathbb{E}[r\phi] - \mathbb{E}[\phi(\phi - \gamma\phi')^\top]\theta \\ &= \mathbb{E}[r\phi - \phi(\phi - \gamma\phi')^\top\theta] \\ &= \mathbb{E}[\delta_\theta\phi] \end{aligned}$$

with the one step TD error $\delta_\theta = \delta(\theta) := r + \theta^\top\left(\gamma\phi' - \phi\right)$.

Remark

Any θ solves the problem $\mathbb{E}[\delta_\theta\phi] = 0$ is a one-step linear TD solution, i.e. the vector $\mathbb{E}[\delta_\theta\phi]$ can be viewed as an error in the current solution θ

Cost function 1

The norm of expected TD update (NEU)

$$J_1 : \mathbb{R}^m \to \mathbb{R}, \qquad J_1(\theta) := \frac{1}{2} \left\| \mathbb{E}[\delta_\theta \phi] \right\|^2$$
$$= \frac{1}{2} \left(\mathbb{E}[\delta_\theta \phi] \right)^\top \mathbb{E}[\delta_\theta \phi],$$

Gradient descent

$$\nabla J_1(\theta) = \left(\nabla \mathbb{E}[\delta_\theta \phi] \right)^\top \mathbb{E}[\delta_\theta \phi]$$
$$= -\mathbb{E}[(\phi - \gamma \phi')\phi^\top] \mathbb{E}[\delta_\theta \phi]$$

ATD(0) algorithm

Naïve algorithm

$$\theta_{t+1} = \theta_t + \alpha_t \left(\frac{1}{t} \sum_{i=1}^{t} (\phi_i - \gamma \phi_i')\phi_i^\top \right) \delta_t \phi_t.$$

Remark

Complex is $\sim O(m^2)$

GTD(0) or GTD1 algorithm

Update rule

For each (s_t, a_t, r_{t+1}), update

$$\theta_{t+1} = \theta_t + \alpha_t(\phi_t - \gamma\phi'_t)\phi_t^\top u_t$$

with

$$u_{t+1} = u_t + \beta_t(\delta_t\phi_t - u_t)$$

Reference

Sutton, Szepesvári and Maei.

A Convergent $O(n)$ Algorithm for Off-policy Temporal-difference Learning with Linear Function Approximations.

pp. 1609–1616, NIPS 21, 2008.

Cost function 2

The Mean Squared Projected Bellman Error (MSPBE)

$$J_2 : \mathbb{R}^k \to \mathbb{R}, \qquad J_2(\theta) := \frac{1}{2} \left\| \Pi_\mu \mathcal{T}_\pi V_\theta - V_\theta \right\|_\mu^2$$
$$= \frac{1}{2} \mathbb{E}[\delta_\theta \phi]^\top \mathbb{E}[\phi \phi^\top]^{-1} \mathbb{E}[\delta_\theta \phi].$$

Gradient descent

$$\nabla J_2(\theta) = -\mathbb{E}[(\phi - \gamma \phi')\phi^\top] \Big(\mathbb{E}[\phi \phi^\top] \Big)^{-1} \mathbb{E}[\delta_\theta \phi]$$
$$\approx -\mathbb{E}[(\phi - \gamma \phi')\phi^\top w]$$

GTD2 algorithm

Update rule

For each (s_t, a_t, r_{t+1}), update

$$\theta_{t+1} = \theta_t + \alpha_t(\phi_t - \gamma\phi'_t)\phi_t^\top w_t$$

with

$$w_{t+1} = w_t + \beta_t(\delta_t - \phi_t^\top w_t)\phi_t.$$

Reference

Sutton, Maei, Precup, Bhatnagar, Silver, Szepesvári, Wiewiora.

Fast Gradient-Descent Methods for Temporal-Difference Learning with Linear Function Approximation.

pp. 993–1000, ICML 2009.

TDC algorithm

Gradient descent 2

$$\begin{aligned}\nabla J_2(\theta) &= -\mathbb{E}[(\phi - \gamma\phi')\phi^\top]\Big(\mathbb{E}[\phi\phi^\top]\Big)^{-1}\mathbb{E}[\delta_\theta\phi] \\ &= -\Big(\mathbb{E}[\phi\phi^\top] - \mathbb{E}[\gamma\phi'\phi^\top]\Big)\Big(\mathbb{E}[\phi\phi^\top]\Big)^{-1}\mathbb{E}[\delta_\theta\phi] \\ &\approx -\Big(\mathbb{E}[\delta_\theta\phi] - \gamma\mathbb{E}[\phi'\phi^\top]w\Big)\end{aligned}$$

Update rule

For each (s_t, a_t, r_{t+1}), update

$$\theta_{t+1} = \theta_t + \alpha_t(\delta_t\phi_t - \gamma\phi'_t\phi_t^\top w_t)$$

with

$$w_{t+1} = w_t + \beta_t(\delta_t - \phi_t^\top w_t)\phi_t$$