Fall 2023

Introduction

What is sequential decision making?

Cathy Wu

6.7920: Reinforcement Learning: Foundations and Methods

Announcements

- HW0: Out on Piazza. Not due. Purpose is to check your understanding.
- HW1: Released tomorrow

- First recitation: Tomorrow at 10am (Room 32-155)
  - Preview for next week
  - Topic: Finite-horizon inventory control

- Registration questions? Ask during break or after class.

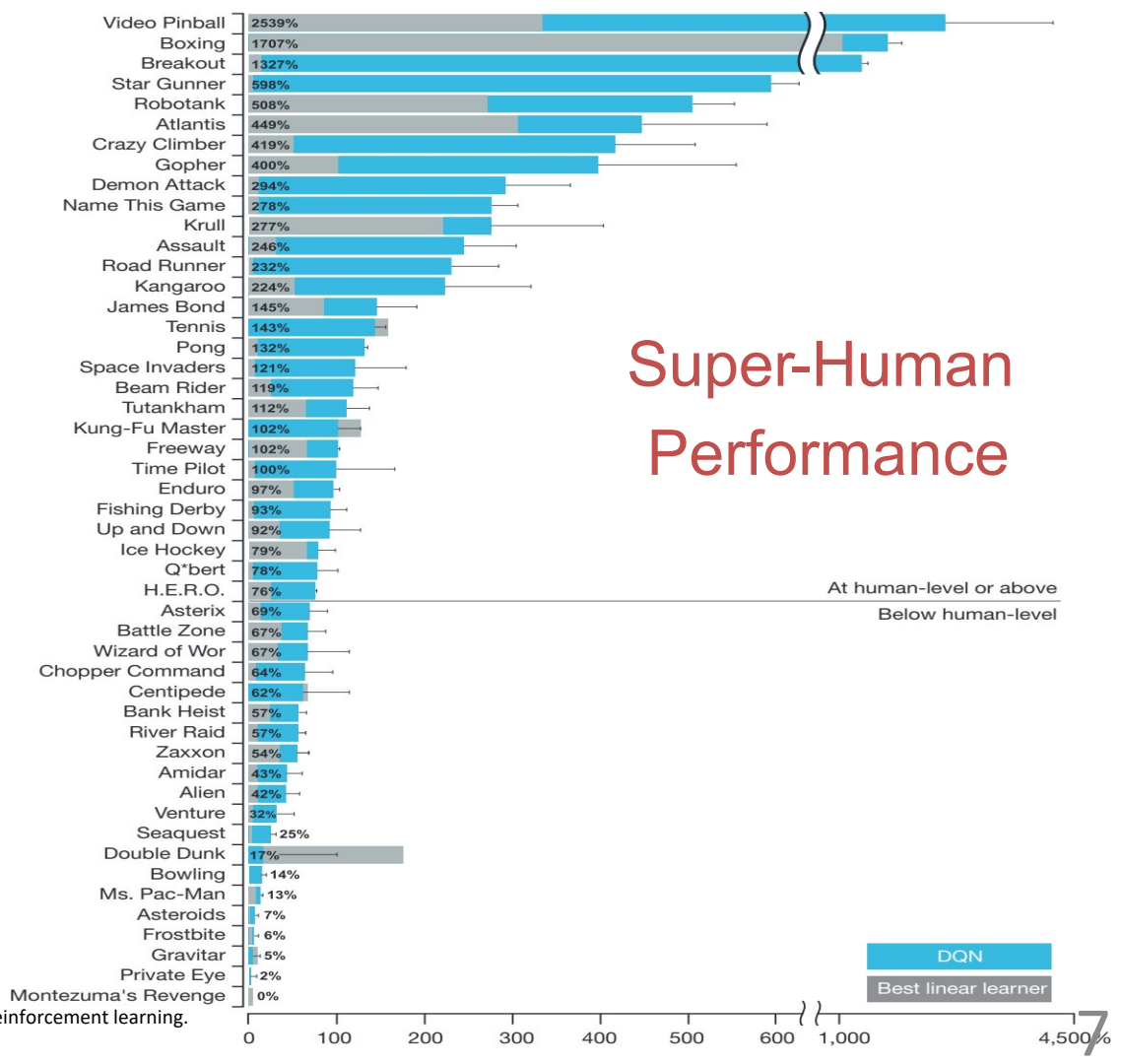
Readings

1. 6.231 Sp22 Lecture 1-3 notes [N] N1 §3, N2 §1-3, N3 §1
2. Dynamic Programming and Optimal Control vol 1 [DPOC] 1.1-1.3, 2.1
3. (Optional) Reinforcement Learning: An introduction [SB] Ch1

Outline

1. Reinforcement learning to solve sequential decision problems
2. Formulation of finite-horizon decision problems
3. Course overview

2015:

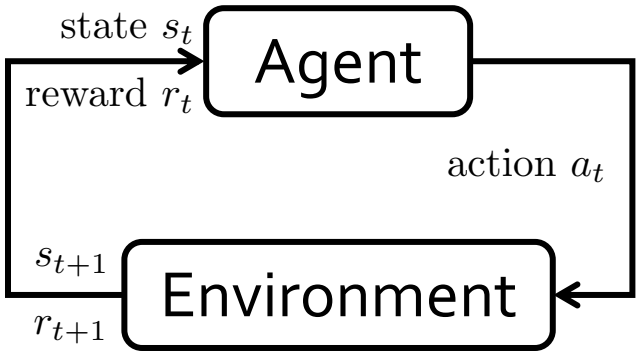


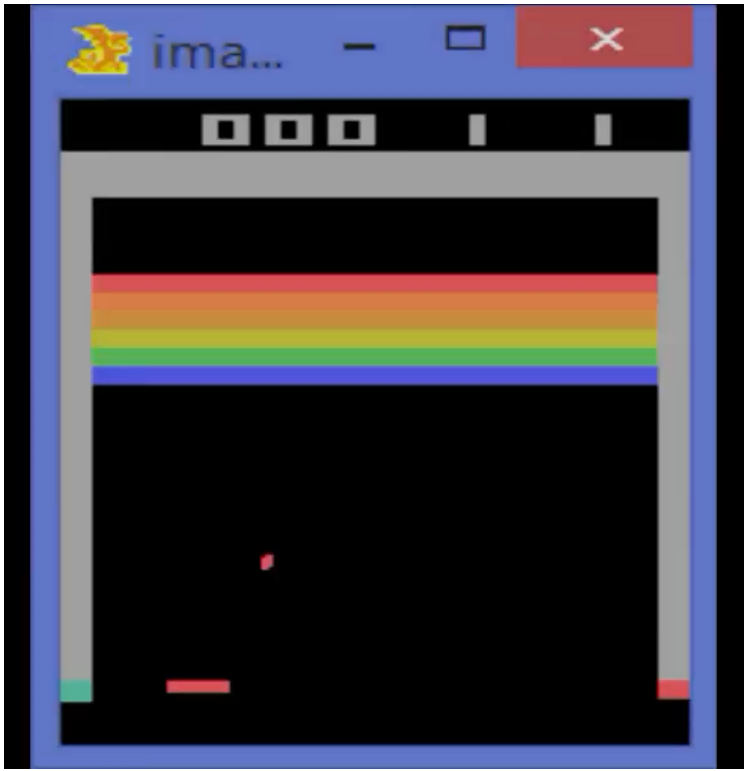
Super-Human Performance

Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature 518, 529–533 (2015). https://doi.org/10.1038/nature14236

Deep reinforcement learning (RL)

- Solve complex decision & control problems through trial and error
- Model-free: no need for an explicit model

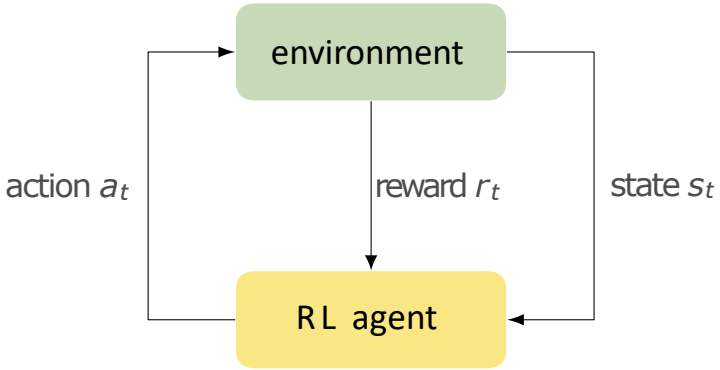




V. Mnih *et al.*, **Human-level control through deep reinforcement learning**, *Nature*, 2015.

What: Reinforcement Learning

Also known as *approximate dynamic programming* (ADP). We will use these terms more-or-less interchangeably.



"**Reinforcement learning** is learning how to map states to actions so as to **maximize** a numerical **reward** signal in an unknown and **uncertain** environment.

In the most interesting and challenging cases, **actions** affect not only the immediate reward but also the **next situation** and all subsequent rewards (**delayed reward**).

The agent is not told which actions to take but it must discover which actions yield the most reward by trying them (**trial-and-error**)."

— Sutton and Barto (1998)

"No simple yet reasonable evaluation function will ever be found for Go."

-- 2002, Martin Müller
(winner of 2009 Go program competition)

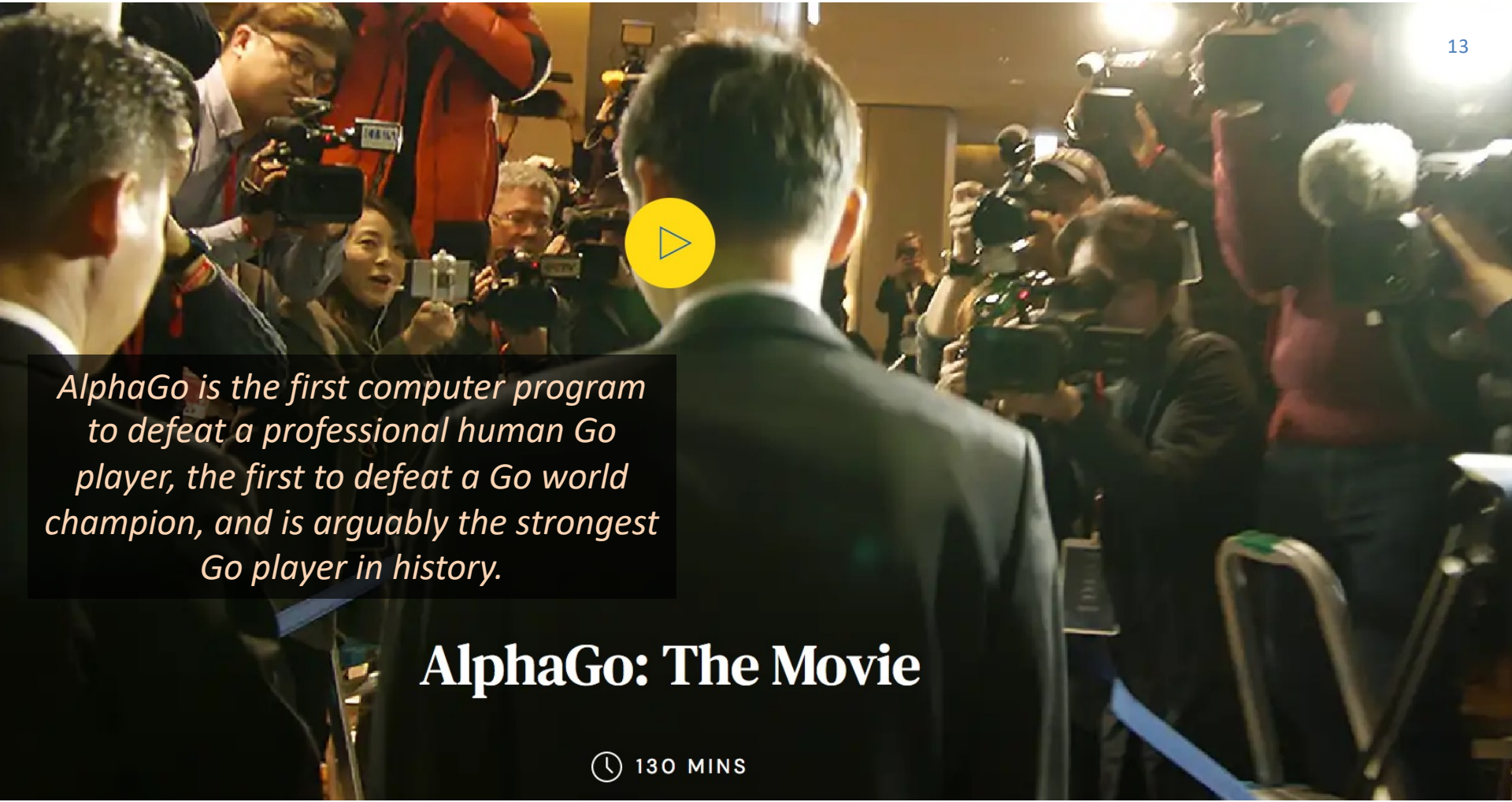
2016:

ARTICLE

doi:10.1038/nature16961

Mastering the game of Go with deep neural networks and tree search

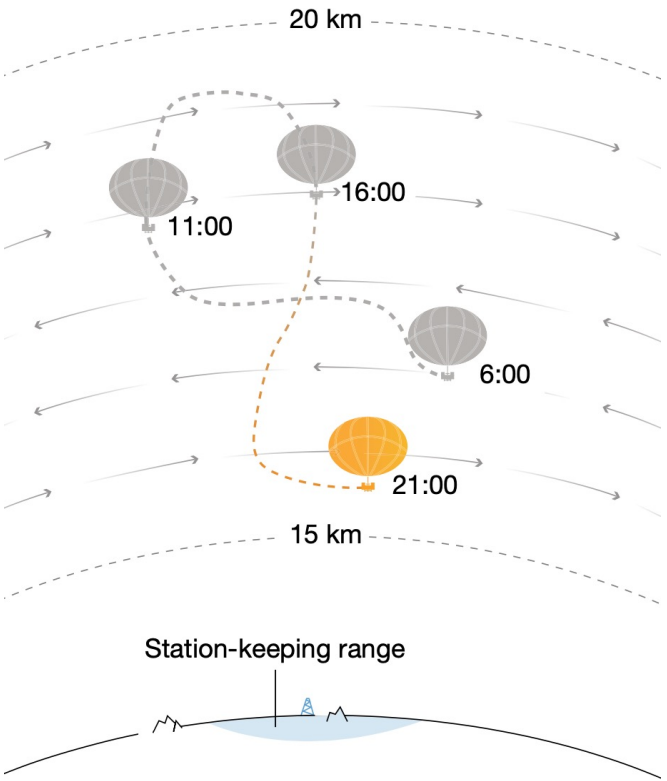
David Silver[1]*, Aja Huang[1]*, Chris J. Maddison[1], Arthur Guez[1], Laurent Sifre[1], George van den Driessche[1], Julian Schrittwieser[1], Ioannis Antonoglou[1], Veda Panneershelvam[1], Marc Lanctot[1], Sander Dieleman[1], Dominik Grewe[1], John Nham[2], Nal Kalchbrenner[1], Ilya Sutskever[2], Timothy Lillicrap[1], Madeleine Leach[1], Koray Kavukcuoglu[1], Thore Graepel[1] & Demis Hassabis[1]



AlphaGo is the first computer program to defeat a professional human Go player, the first to defeat a Go world champion, and is arguably the strongest Go player in history.

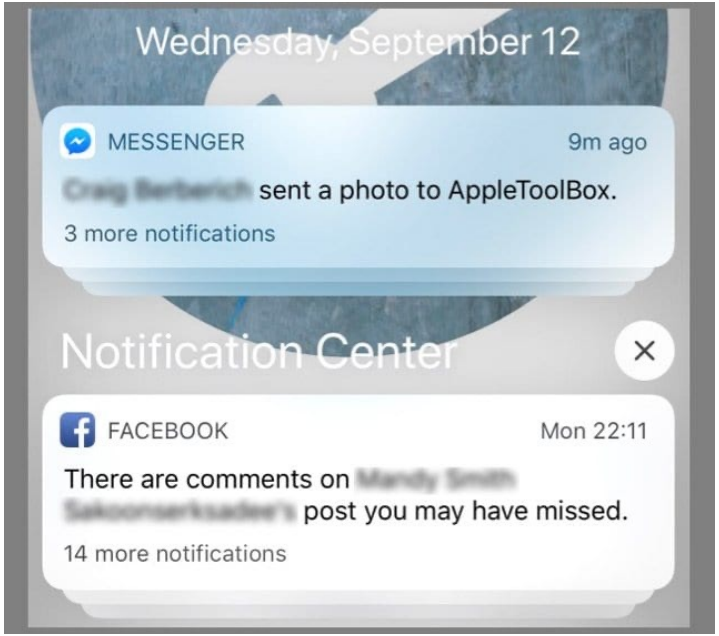
https://www.youtube.com/watch?v=WXuK6gekU1Y

High-altitude balloons (2020)



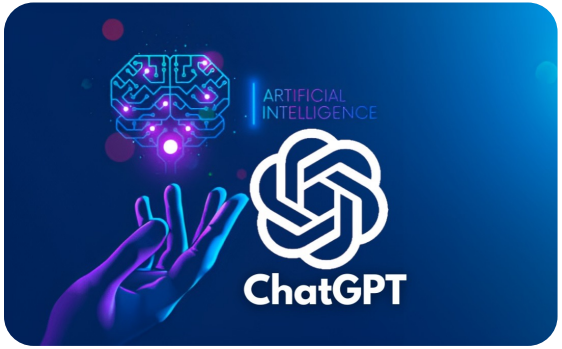
Bellemare et al., “Autonomous navigation of stratospheric balloons using reinforcement learning” Nature, 2020.

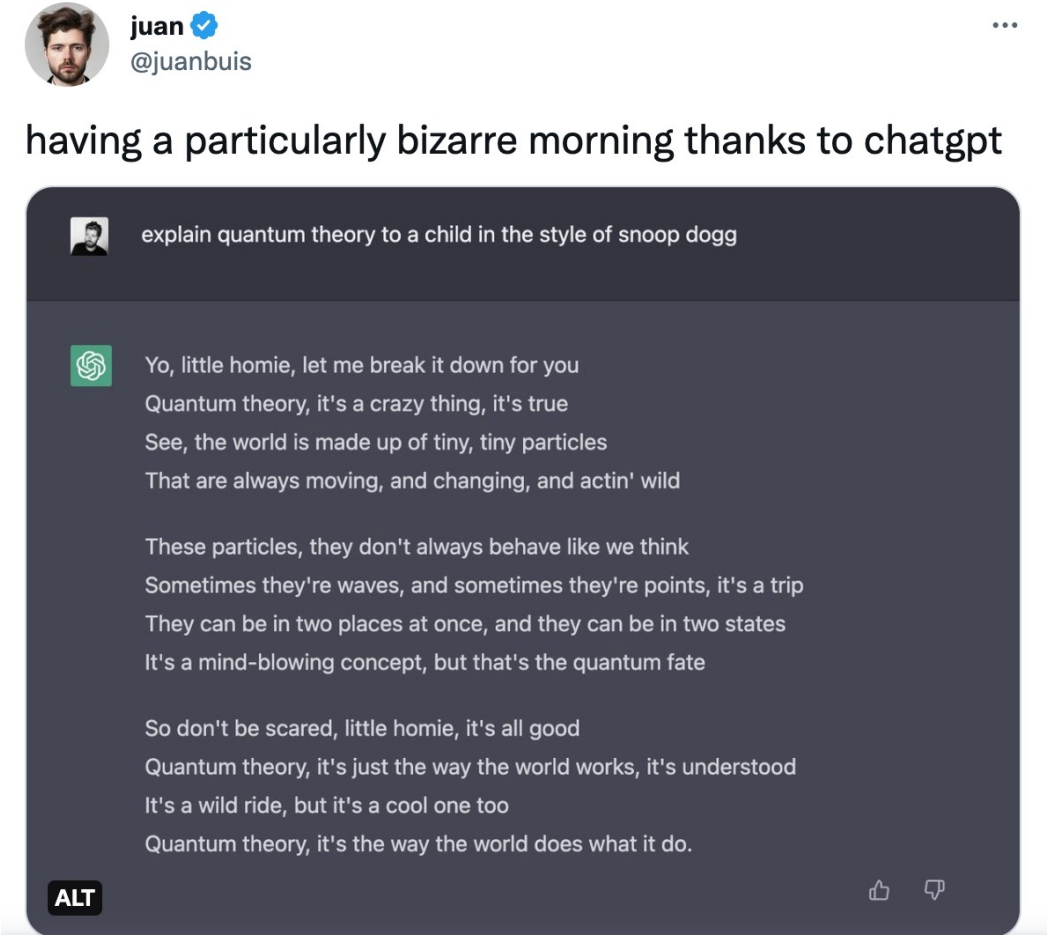
Push notifications (2020)



Gauci, et al., “Horizon: Facebook's Open Source Applied Reinforcement Learning Platform - Facebook Research” (2020)

ChatGPT (2022)

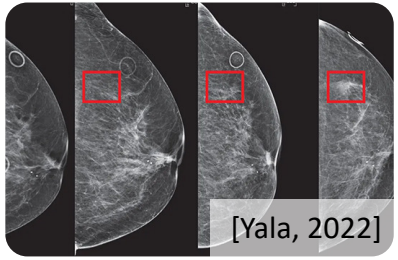




Reinforcement learning for increasingly complex decision making


[Bellemare, 2020]

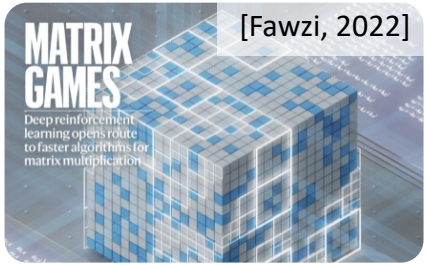
Google Loon


[Yala, 2022]

Breast cancer screening

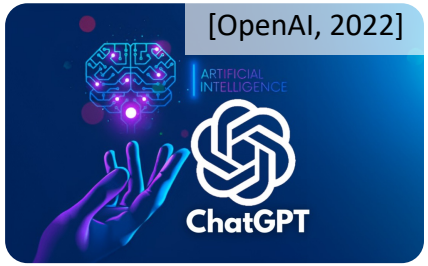
[Feng, 2023]

AV safety validation



[Fawzi, 2022]

AlphaTensor



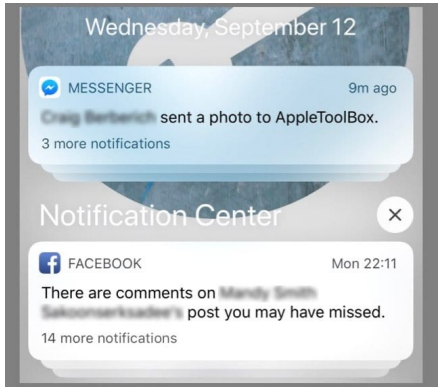
[OpenAI, 2022]

ChatGPT

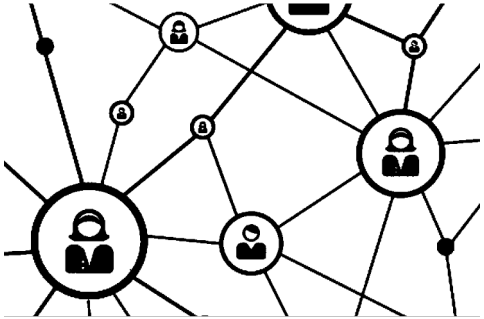

[Kaufmann, 2023]

Drone racing

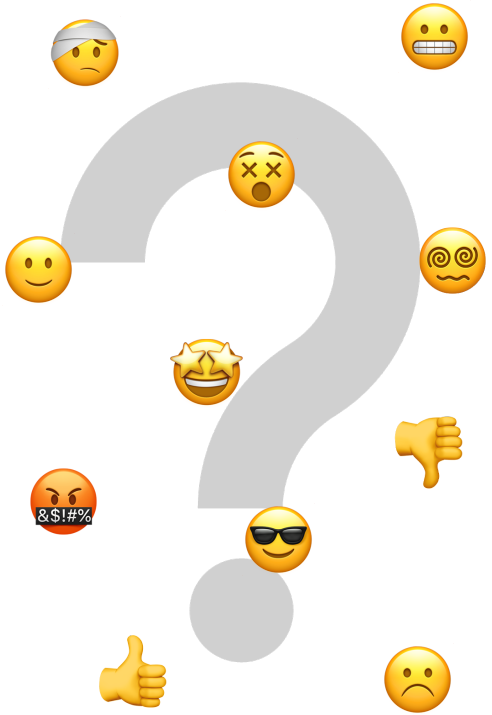
RL as a scientific tool to study automated decision making



+



=



+

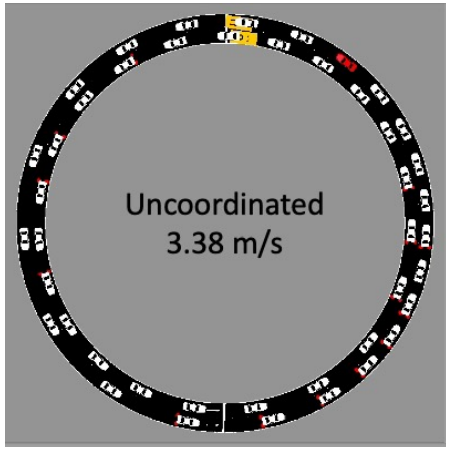
=

Agent **Environment** **Reward**

Impact of autonomous vehicles on urban traffic

Mixed autonomy is the decades-long regime between no autonomy (0% AVs) and full autonomy (100% AVs).

No autonomy

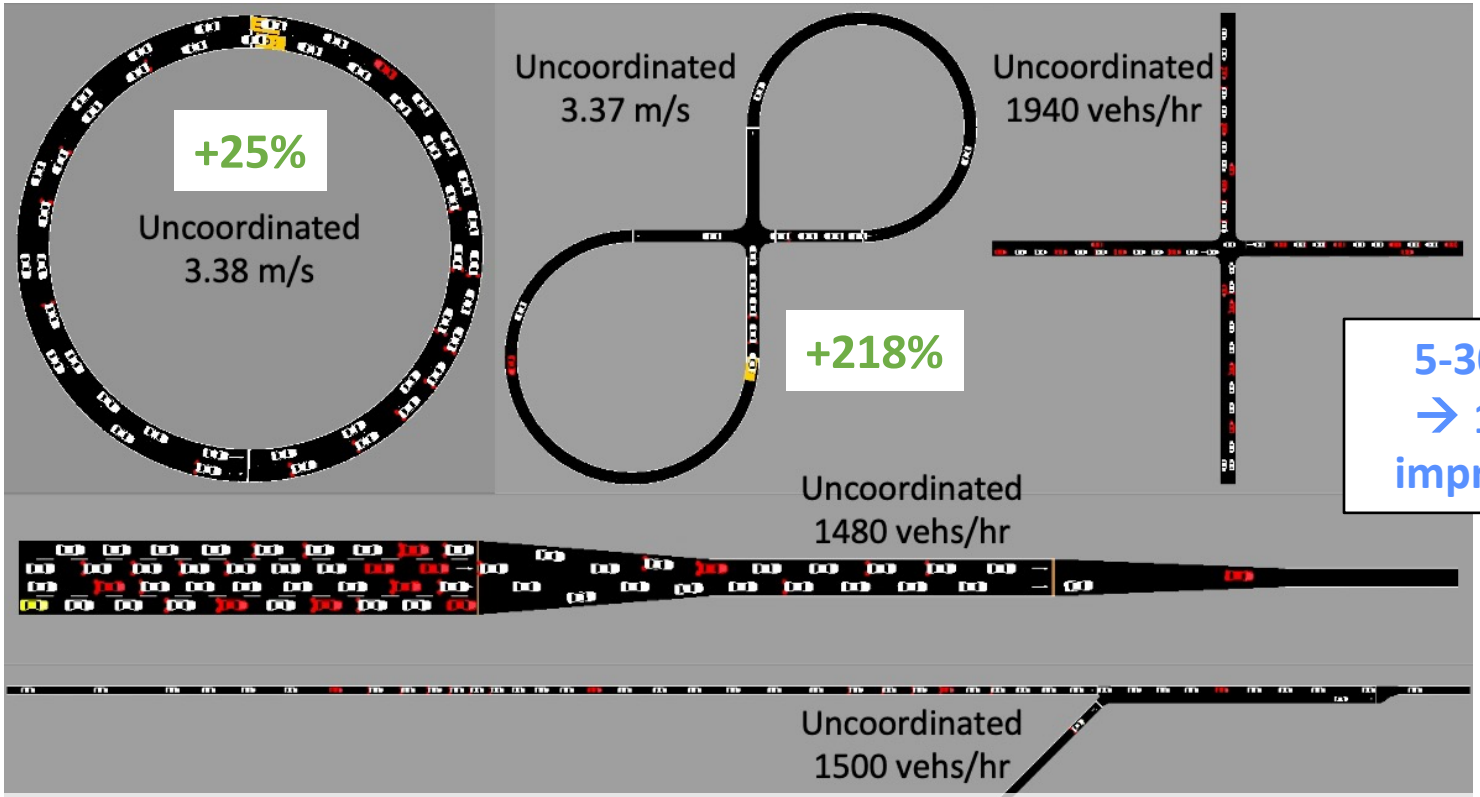


Mixed autonomy

Full autonomy

Wu, Kreidieh, Vinitsky, Bayen, **Emergent behaviors in mixed-autonomy traffic**, in *1st Annual Conference on Robot Learning (CoRL)*, PMLR, 2017.
Wu, Kreidieh, Parvate, Vinitsky, Bayen, **Flow: A modular learning framework for mixed autonomy traffic**, *IEEE Transactions on Robotics (T-RO)*, 2021.
Yan, Kreidieh, Vinitsky, Bayen, Wu, **Unified automatic control of vehicular systems with reinforcement learning**, *IEEE Transactions on Automation Science and Engineering (T-ASE)*, 2022.

Autonomous vehicles can boost traffic flow (2016–2022)



Wu, Kreidieh, Vinitsky, Bayen, **Emergent behaviors in mixed-autonomy traffic**, in *1st Annual Conference on Robot Learning (CoRL)*, PMLR, 2017.
Wu, Kreidieh, Parvate, Vinitsky, Bayen, **Flow: A modular learning framework for mixed autonomy traffic**, *IEEE Transactions on Robotics (T-RO)*, 2021.
Vinitsky, et al. Wu, Bayen. **Benchmarks for reinforcement learning in mixed-autonomy traffic**, in *2nd Annual Conference on Robot Learning (CoRL)*, PMLR, 2018.
Yan, Kreidieh, Vinitsky, Bayen, Wu, **Unified automatic control of vehicular systems with reinforcement learning**, *IEEE Transactions on Automation Science and Engineering (T-ASE)*, 2022.

Q: What applications are you excited about?

But it’s not that easy

It’s hard to specify what you want

RL algorithms are complex

RL implementations are complex

Complex problems are … complex

Dario Amodei, Jack Clark, “Faulty reward functions in the wild,” OpenAI Blog, 2016.

Precise recommendations for training RL models

What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, Olivier Bachem

Google Research, Brain Team

Abstract

In recent years, on-policy reinforcement learning (RL) has been successfully applied to many different continuous control tasks. While RL algorithms are often conceptually simple, their state-of-the-art implementations take numerous low- and high-level design decisions that strongly affect the performance of the resulting agents. Those choices are usually not extensively discussed in the literature, leading to discrepancy between published descriptions of algorithms and their implementations. This makes it hard to attribute progress in RL and slows down overall progress [27]. As a step towards filling that gap, we implement >50 such "choices" in a unified on-policy RL framework, allowing us to investigate their impact in a large-scale empirical study. We train over 250'000 agents in five continuous control environments of different complexity and provide insights and practical recommendations for on-policy training of RL agents.

Andrychowicz, *et al.*, "What matters in on-policy reinforcement learning? A large-scale empirical study," International conference on learning representations (ICLR), 2021.

Precise recommendations for training RL models

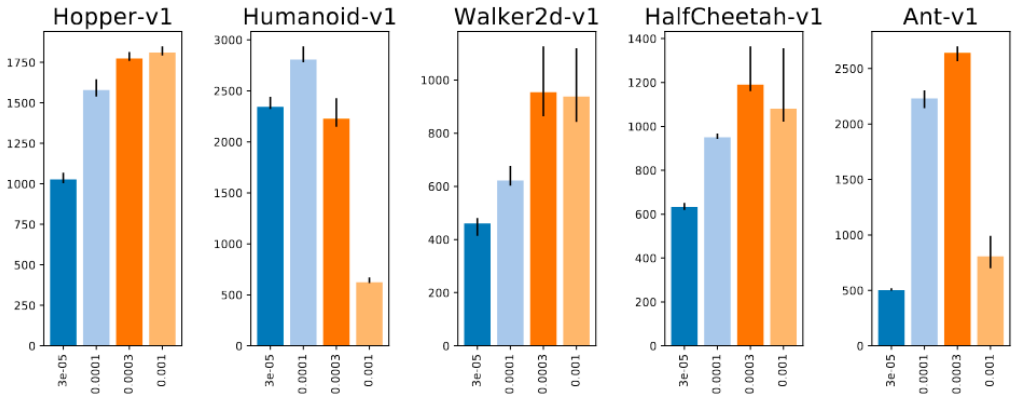
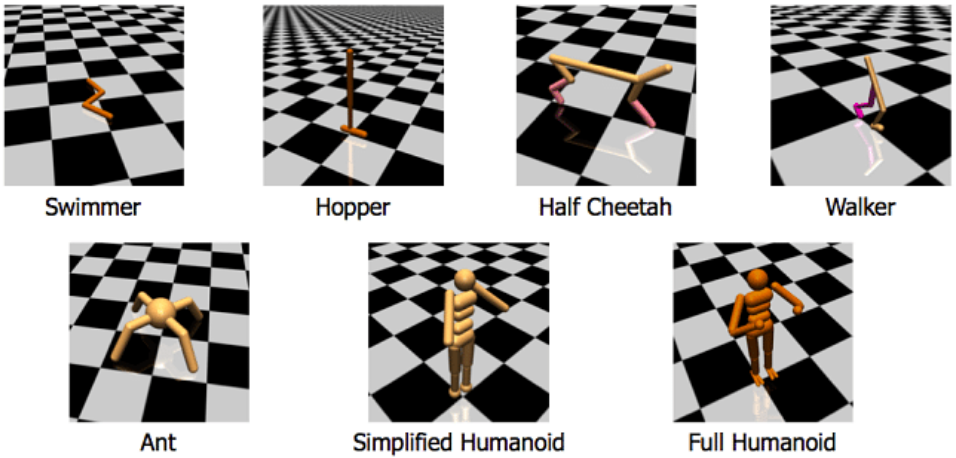


Figure 69: Analysis of choice `Adam learning rate`

Recommendation. Use Adam [8] optimizer with momentum $\beta_1 = 0.9$ and a tuned learning rate (0.0003 is a safe default). Linearly decaying the learning rate may slightly improve performance but is of secondary importance.

Andrychowicz, *et al.*, "What matters in on-policy reinforcement learning? A large-scale empirical study," International conference on learning representations (ICLR), 2021.

Recommendations overfit to Mujoco environments



Mujoco environments

Abstract

In recent years, on-policy reinforcement learning (RL) has been successfully applied to many different continuous control tasks. While RL algorithms are often conceptually simple, their state-of-the-art implementations take numerous low- and high-level design decisions that strongly affect the performance of the resulting agents. Those choices are usually not extensively discussed in the literature, leading to discrepancy between published descriptions of algorithms and their implementations. This makes it hard to attribute progress in RL and slows down overall progress [27]. As a step towards filling that gap, we implement >50 such "choices" in a unified on-policy RL framework, allowing us to investigate their impact in a large-scale empirical study. We train over 250'000 agents in five continuous control environments of different complexity and provide insights and practical recommendations for on-policy training of RL agents.

Andrychowicz, *et al.*, "What matters in on-policy reinforcement learning? A large-scale empirical study," International conference on learning representations (ICLR), 2021.

The antidote? Seek foundations

This class is for students seeking a foundational understanding of reinforcement learning, in order to:

- *Systematically apply* reinforcement learning to a problem of your choice and understand when *not* to
- *Develop reliable methods* for reinforcement learning
- *Derive insights* into either the methods or problems

How? We will discuss later [Course overview]

Outline

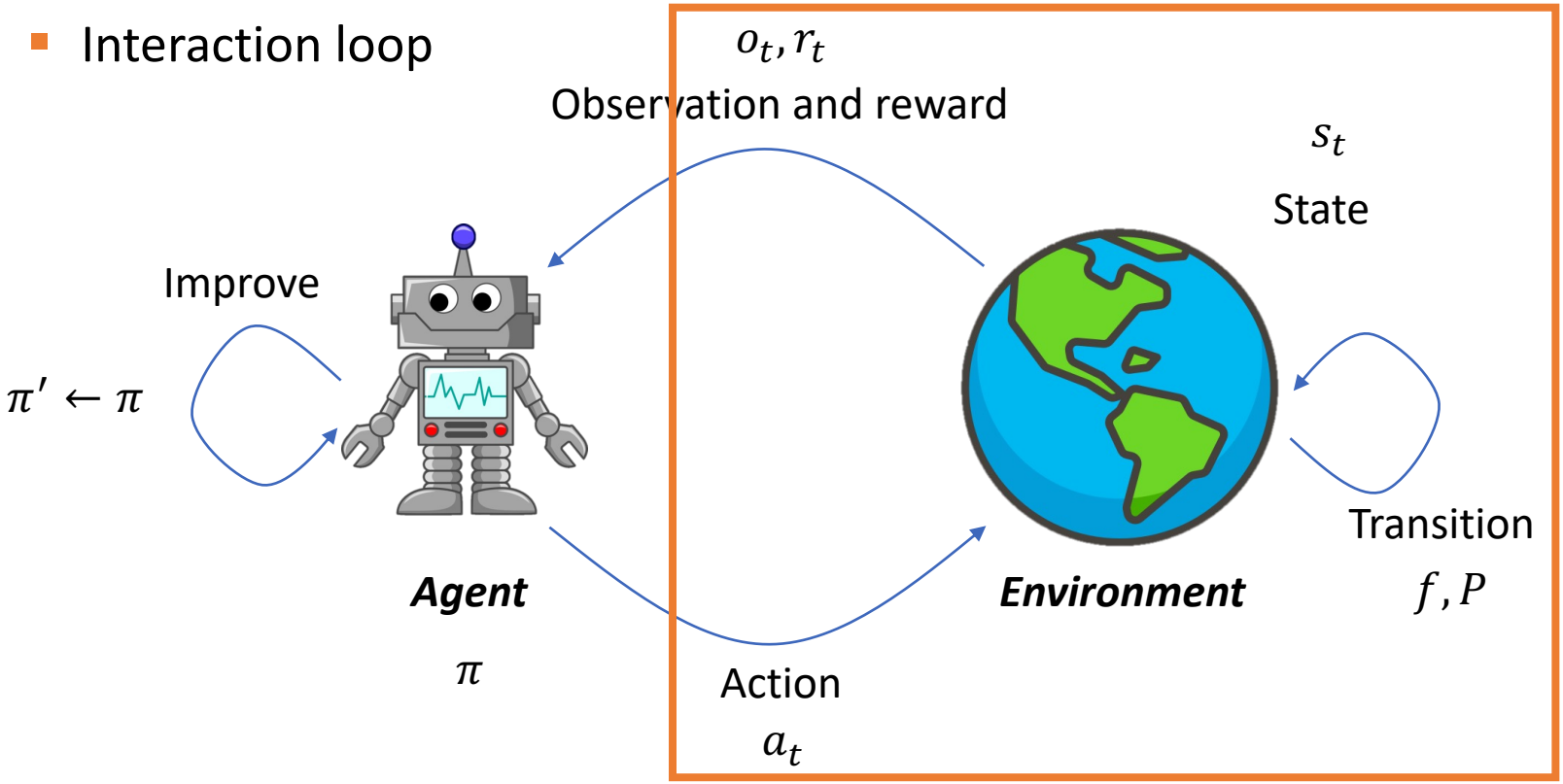
1. Reinforcement learning to solve sequential decision problems
2. **Formulation of finite-horizon decision problems**
   a. Optimization objective: Value function
   b. Constraints: Markov Decision Process (MDP)
   c. Variables: Policy
3. Course overview

Introduce the characters*

M

Markov Decision Process (MDP)

- Interaction loop



Goal: maximize reward over time (returns, cumulative reward)

* pun intended

Assume for now: finite horizon problems, i.e. $T < \infty$

Used when: there is an intrinsic deadline to meet.

Later: infinite horizon

The optimization problem of reinforcement learning

- Call V^{π} the value function, which indicates the expected cumulative reward using policy π.

$$V^{\pi}(t,s) = \mathbb{E}[\sum_{\tau=t}^{T-1} r(s_\tau, \pi(s_\tau)) + R(s_T)|s_t = s; \pi, M]$$

  where R is a value function for the final state, and M is a Markov Decision Process.
- Shorthand: $V_t^{\pi}(s)$ or simply V_t^{π} (think: vector of size $|S|$)
- Our goal: achieve the best value, i.e., max value-to-go (min cost-to-go)

Definition (Optimal policy and optimal value function)

The solution to a Markov Decision Process M is an optimal policy π^* satisfying

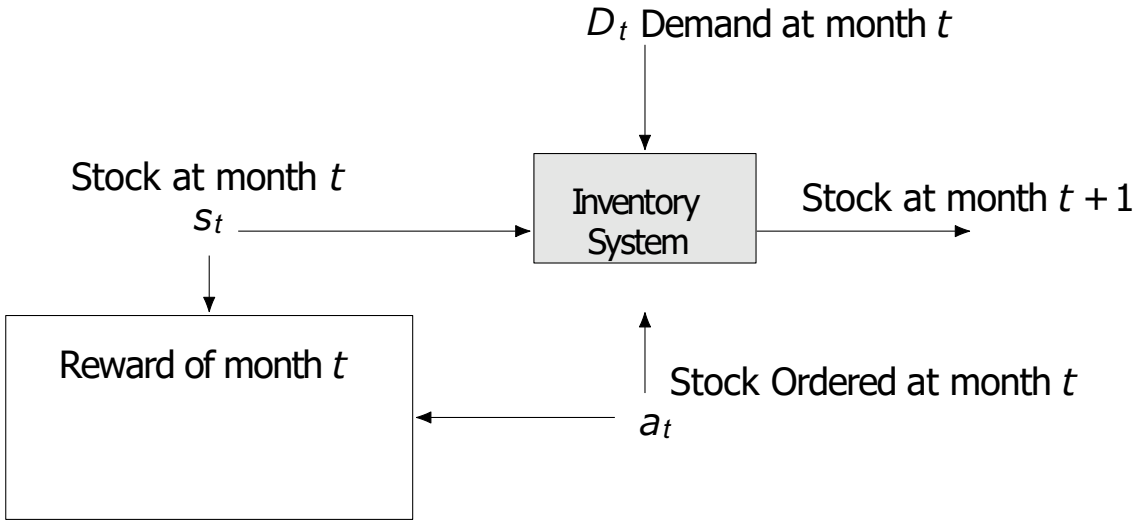
$$\pi^* \in \arg\max_{\pi \in \Pi} V_0^{\pi}$$

where Π is some policy set of interest.

The corresponding value function is the optimal value function

$$V^* = V_0^{\pi^*}$$

Example: The Amazing Goods Company Example



Example: The Amazing Goods Company Example

- *Description.* At each month t, a warehouse contains s_t *items* of a specific goods and the demand for that goods is D (stochastic). At the end of each month the manager of the warehouse can *order* a_t more items from the supplier.

- The cost of maintaining an inventory of s is $h(s)$.
- The cost to order a items is $C(a)$.
- The income for selling q items if $f(q)$.
- If the demand $d \sim D$ is bigger than the available inventory s, customers that cannot be served leave.
- The value of the remaining inventory at the end of the year is $g(s)$.
- Constraint: the store has a maximum capacity C.

Markov Decision Process

Definition (Markov decision process)

A **Markov decision process** (MDP) is defined as a tuple $M = (S, A, P \text{ or } f, r, T)$ where

- S is the *state* space,

Example: The Amazing Goods Company

- State space: $s \in S = \{0, 1, \ldots, C\}$.

Markov Decision Process

Definition (Markov decision process)

A **Markov decision process** (MDP) is defined as a tuple $M = (S, A, P \text{ or } f, r, T)$ where

- S is the *state* space,
- A is the *action* space,

Example: The Amazing Goods Company

- Action space: it is not possible to order more items than the capacity of the store, so the action space should depend on the current state. Formally, at state s, $a \in A(s) = \{0, 1, \ldots, C - s\}$.

Markov Decision Process

Definition (Markov decision process)

A **Markov decision process** (MDP) is defined as a tuple $M = (S, A, P \text{ or } f, r, T)$ where

- S is the *state* space,
- A is the *action* space,

} often simplified to finite

- $P(s'|s,a)$ is the transition probability with

$$P(s'|s,a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a)$$

transition equation
$s' = f_t(s, a, w_t)$
where $w_t \sim W_t$
(some random variables)

Example: The Amazing Goods Company

- Dynamics: $s_{t+1} = [s_t + a_t - d_t]^+$.
- The demand d_t is stochastic and time-independent. Formally, $d_t \overset{\text{i.i.d.}}{\sim} D$.

Recall: Markov Chains

Definition (Markov chain)

Let the *state space S* be a subset of the Euclidean space, the discrete-time dynamic system $(s_t)_{t\in\mathbb{N}} \in S$ is a Markov chain if it satisfies the *Markov property*

$$P(s_t + 1 = s | s_t, s_t - 1, \ldots, s_0) = P(s_t + 1 = s | s_t),$$

Given an initial state $s_0 \in S$, a Markov chain is defined by the *transition probability p*

$$p(s'|s) = P(s_t + 1 = s' | s_t = s).$$

Markov Decision Process

Definition (Markov decision process)

A **Markov decision process** (MDP) is defined as a tuple $M = (S, A, P \text{ or } f, r, T)$ where

- S is the *state* space,
- A is the *action* space,

} often simplified to finite

- $P(s'|s,a)$ is the transition probability with

$$P(s'|s,a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a)$$

- $r(s,a,s')$ is the immediate reward at state s upon taking action a,

} sometimes simply $r(s)$

- H is the horizon.

☞ In general, a non-Markovian decision process's transitions could depend on much more information:

$$\mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a, s_{t-1}, a_{t-1}, \dots, s_0, a_0),$$

Markov Decision Process

Definition (Markov decision process)

A **Markov decision process** (MDP) is defined as a tuple $M = (S, A, P \text{ or } f, r, T)$ where

- S is the *state* space,
- A is the *action* space,

often simplified to finite

- $P(s'|s,a)$ is the transition probability with

$$P(s'|s,a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a)$$

- $r(s,a,s')$ is the immediate reward at state s upon taking action a,

sometimes simply $r(s)$, assumed to be bounded

Example: The Amazing Goods Company

- Reward: $r_t = -C(a_t) - h(s_t + a_t) + f([s_t + a_t - s_{t+1}]^+)$. This corresponds to a purchasing cost, a cost for excess stock (storage, maintenance), and a reward for fulfilling orders.

Markov Decision Process

Definition (Markov decision process)

A **Markov decision process** (MDP) is defined as a tuple $M = (S, A, P \text{ or } f, r, T)$ where

- S is the *state* space,
- A is the *action* space,

} often simplified to finite

- $P(s'|s,a)$ is the transition probability with

$$P(s'|s,a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a)$$

- $r(s,a,s')$ is the immediate reward at state s upon taking action a,

} sometimes simply $r(s)$

- T is the horizon.

Example: The Amazing Goods Company

- The horizon of the problem is 12 (12 months in 1 year).

Markov Decision Process (infinite horizon preview)

Definition (Markov decision process)

A **Markov decision process** (MDP) is defined as a tuple $M = (S, A, P \text{ or } f, r, \gamma)$ where

- S is the *state* space,
- A is the *action* space,

often simplified to finite

- $P(s'|s,a)$ is the transition probability with

$$P(s'|s,a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a)$$

- $r(s,a,s')$ is the immediate reward at state s upon taking action a,

sometimes simply $r(s)$

- $\gamma \in [0,1)$ is the discount factor.

Example: The Amazing Goods Company

- Discount: $\gamma = 0.91667$. A dollar today is worth more than a dollar tomorrow.
- The effective horizon of the problem is 12 (12 months in 1 year), i.e. $\mathrm{T} \approx \frac{1}{1-\gamma}$.

Markov Decision Process

Definition (Markov decision process)

A **Markov decision process** (MDP) is defined as a tuple $M = (S, A, P \text{ or } f, r, T)$ where

- S is the *state* space,
- A is the *action* space,

} often simplified to finite

- $P(s'|s,a)$ is the transition probability with

$$P(s'|s,a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a)$$

- $r(s,a,s')$ is the immediate reward at state s upon taking action a,

} sometimes simply $r(s)$

- T is the horizon.

☞ The process generates trajectories $\tau_t = (s_0, a_0, \ldots, s_{t-1}, a_{t-1}, s_t)$, with $s_{t+1} \sim P(\cdot|s_t, a_t)$

Markov Decision Process

Definition (Markov decision process)

A **Markov decision process** (MDP) is defined as a tuple $M = (S, A, P \text{ or } f, r, T)$ where

- S is the *state* space,
- A is the *action* space,

} often simplified to finite

- $P(s'|s,a)$ is the transition probability with

$$P(s'|s,a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a)$$

- $r(s,a,s')$ is the immediate reward at state s upon taking action a,

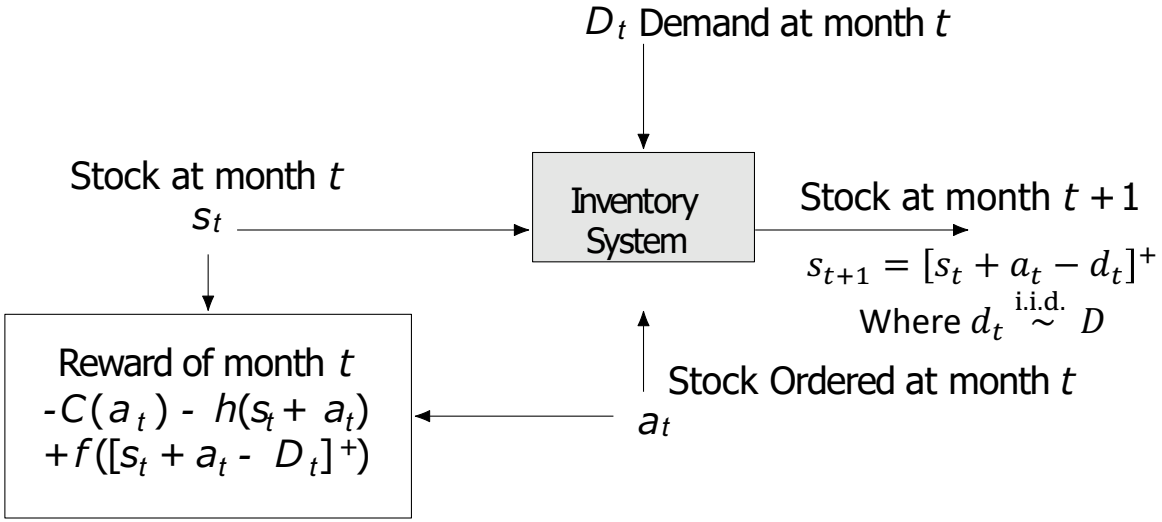
} sometimes simply $r(s)$

- T is the horizon.

Example: The Amazing Goods Company

- Objective: $V(s_0; a_0, \dots) = \mathbb{E}[\sum_{t=0}^{T-1} r_t + r_T \,|s_0 = s_0; a_0, \dots; M]$, where $r_{12} = g(s_{12})$. This corresponds to the cumulative reward, including the value of the remaining inventory at "the end."

Example: The Amazing Goods Company Example



- State space: $s \in S = \{0, 1, \ldots, C\}$.
- Action space: it is not possible to order more items than the capacity of the store, so the action space should depend on the current state. Formally, at state s, $a \in A(s) = \{0, 1, \ldots, C - s\}$.
- Objective: $V(s_0; a_0, \ldots) = \mathbb{E}[\sum_{t=0}^{T-1} r_t + r_T \,|s_0 = s_0; a_0, \ldots; M]$, where $T = 12$ and $r_{12} = g(s_{12})$

Expectations

- Technical note: the expectations refer to all possible stochastic trajectories.
- A (possibly non-stationary stochastic) policy π applied from state s_0 returns

$$(s_0, r_0, s_1, r_1, s_2, r_2, \dots)$$

- Where $r_t = r(s_t, a_t)$ and $s_{t+1} \sim p(\cdot\,|s_t, a_t = \pi_t(s_t))$ are random realizations.

- The value function is

$$V^{\pi}(t,s) = \mathbb{E}_{(s_1, s_2, \dots)}\left[\sum_{\tau=t}^{T-1} r(s_\tau, \pi(s_\tau)) + R(s_T)|s_t = s; \pi\right]$$

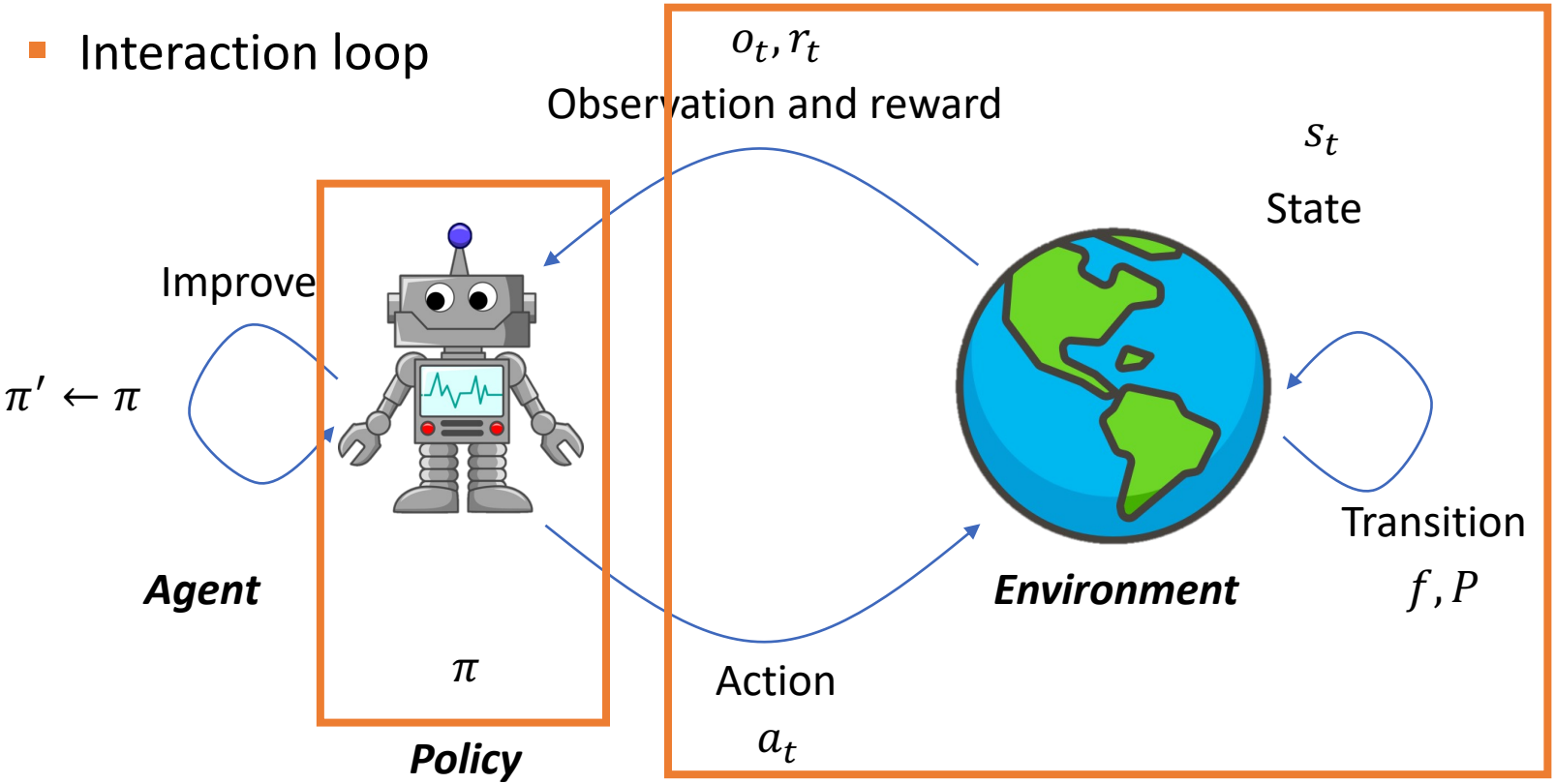
- More generally, for stochastic policies:

$$V^{\pi}(t,s) = \mathbb{E}_{(a_0, s_1, a_1, s_2, \dots)}\left[\sum_{\tau=t}^{T-1} r(s_\tau, \pi(s_\tau)) + R(s_T)|s_t = s; \pi\right]$$

Recall: the characters*

Markov Decision Process (MDP) M

- Interaction loop



Goal: maximize reward over time (returns, cumulative reward)

* pun intended

Policy

Definition (Policy)

A decision rule d can be

- Deterministic: $d: S \to A$,
- Stochastic: $d: S \to \Delta(A)$,
- History-dependent: $d: H_t \to A$,
- Markov: $d: S \to \Delta(A)$,

A policy (strategy, plan) can be

- Stationary: $\pi = (d, d, d, \dots)$,
- (More generally) Non-stationary: $\pi = (d_0, d_1, d_2, \dots)$

☞ For simplicity, we will typically write π instead of d for stationary policies, and π_t instead of d_t for non-stationary policies.

The Amazing Goods Company Example

- *Description.* At each month t, a warehouse contains s_t *items* of a specific goods and the demand for that goods is D (stochastic). At the end of each month the manager of the warehouse can *order* a_t more items from the supplier.

- The cost of maintaining an inventory of s is $h(s)$.
- The cost to order a items is $C(a)$.
- The income for selling q items if $f(q)$.
- If the demand $d \sim D$ is bigger than the available inventory s, customers that cannot be served leave.
- The value of the remaining inventory at the end of the year is $g(s)$.
- Constraint: the store has a maximum capacity C.

Stationary policy composed of deterministic Markov decision rules

$$\pi(s) = \begin{cases} C - s & \text{if } s < M/4 \\ 0 & \text{otherwise} \end{cases}$$

The Amazing Goods Company Example

- *Description.* At each month t, a warehouse contains s_t *items* of a specific goods and the demand for that goods is D (stochastic). At the end of each month the manager of the warehouse can *order* a_t more items from the supplier.

- The cost of maintaining an inventory of s is $h(s)$.
- The cost to order a items is $C(a)$.
- The income for selling q items if $f(q)$.
- If the demand $d \sim D$ is bigger than the available inventory s, customers that cannot be served leave.
- The value of the remaining inventory at the end of the year is $g(s)$.
- Constraint: the store has a maximum capacity C.

Stationary policy composed of stochastic history-dependent decision rules

$$\pi(s_t) = \begin{cases} U(C - s_{t-1}, C - s_{t-1} + 10) & \text{if } s_t < s_{t-1}/2 \\ 0 & \text{otherwise} \end{cases}$$

Summary & takeaways

- Sequential decision problems are those where selected actions affect future states.
  - Sequential decision problems are found everywhere.
  - Deterministic examples include routing, combinatorial optimization, linear quadratic control, inventory management.
  - Stochastic problems are needed to represent uncertainty in the environment and in the policy.
- Markov Decision Processes (MDPs) represent a general class of stochastic sequential decision problems, for which reinforcement learning methods are commonly designed.
  - The Markovian property means that the next state is fully characterized by the current state and action.
  - The generality of MDPs facilitates discussion of model-free learning (later lectures).

Outline

1. Reinforcement learning to solve sequential decision problems
2. Formulation of finite-horizon decision problems
3. **Course overview**
   a. **Course structure**
   b. Administrivia

Philosophy + aims of the course

- What is an appropriate **foundational** course to **advance research and practice** in sequential decision making?
- Context

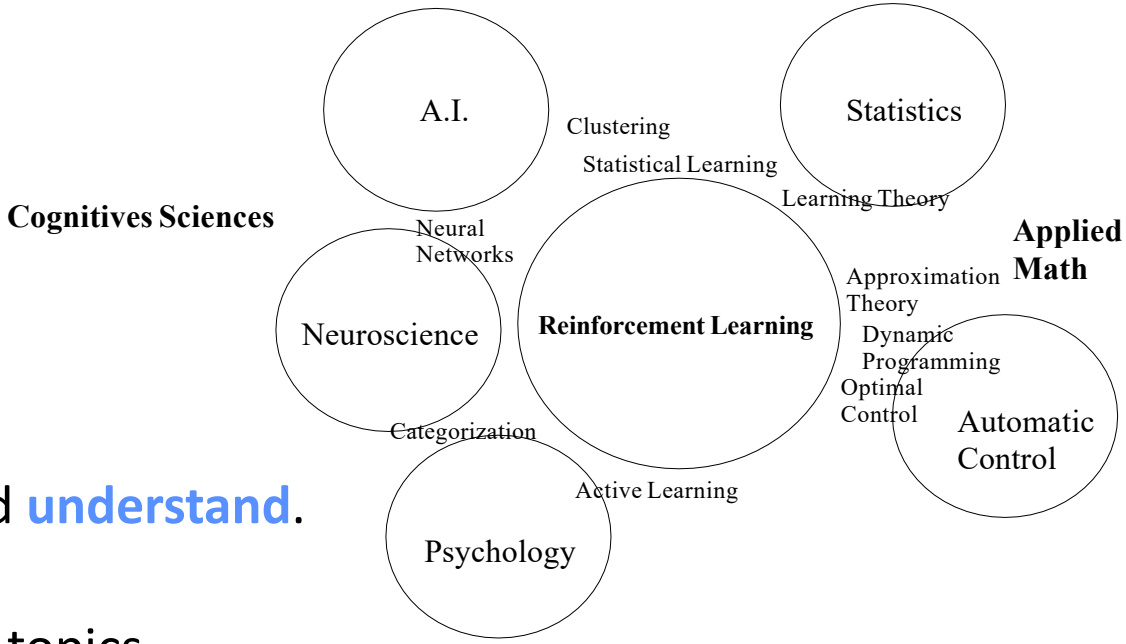


Figure: Note: circles may not be to scale.
Credit: Alessandro Lazaric

Design

- (2/3 **Exploit**)
  Teach what we **know** and **understand**.
- (1/3 **Explore**)
  Selected **up-and-coming** topics.

What: the Highlights of the Course

How to *model* DP & RL problems

- *What:* problem space, deterministic vs Markov decision process, imperfect information
- *Tools:* probability, processes, Markov chain

What: the Highlights of the Course

How to *model* DP & RL problems

How to solve *exactly* DP & RL problems

- *What:* Bellman equations, dynamic programming algorithms
- *Tools:* induction, optimality principle, fixed point theory

What: the Highlights of the Course

How to *model* DP & RL problems

How to solve *exactly* DP & RL problems

How to solve *incrementally* DP & RL problems

- *What:* Monte Carlo, temporal difference (TD), Q-learning
- *Tools:* stochastic approximation theory

What: the Highlights of the Course

How to *model* DP & RL problems

How to solve *exactly* DP & RL problems

How to solve *incrementally* DP & RL problems

How to solve *approximately* DP & RL problems

- *What:* approximate RL (TD-based methods, policy space methods, deep RL)
- *Tools:* function approximation, Lyapunov function analysis, deep learning, variance reduction

What: the Highlights of the Course

How to *model* DP & RL problems

How to solve *exactly* DP & RL problems

How to solve *incrementally* DP & RL problems

How to solve *approximately* DP & RL problems

With examples from *resource optimization*, *control systems*, *computer games*, *and beyond*.

Special topics (tentative)

- Empirical rigor in RL – How to not fool yourself when doing RL
- Learning for discrete optimization – Solving NP-hard problems
- Recommendation systems – Bandits in practice
- Monte Carlo Tree Search – Superhuman Go
- Learning-based control
- Applications
  - Case studies
  - Healthcare
  - Robotics
- Recent theoretical results

Outline

1. Reinforcement learning to solve sequential decision problems
2. Formulation of finite-horizon decision problems
3. Solving finite-horizon decision problems
   a. Example: shortest path routing
   b. Dynamic programming algorithm
   c. Sequential decision making as shortest path
   d. Forward DP
4. **Course overview**
   a. Course structure
   b. Administrivia

How: Textbooks and readings

Useful references (recommended but not required)

(a) **Dynamic Programming and Optimal Control** (2007), Vol. I, 4th Edition, ISBN-13: 978-1-886529-43-4 by Dimitri P. Bertsekas. [**DPOC**]

(b) The second volume of the text is a useful and comprehensive reference. [**DPOC2**]

(c) **Neuro Dynamic Programming** (1996) by Dimitri P. Bertsekas and John N. Tsitsiklis. [**NDP**]

Readings: We will give pointers to these references. Some additional readings / notes may be posted.

A note on notation. We will be using contemporary notation (e.g. s, a, V), which differs from notation from these texts (e.g. x, u, J). We will be maximizing instead of minimizing, etc.

How: Pre-requisites

(a) Solid knowledge of undergraduate probability (6.041A & 6.041B)

(b) Mathematical maturity and the ability to write down precise and rigorous arguments

(c) Python programming

We will issue a HW0 (not graded) to help you gauge your level of familiarity with the pre-requisite material and useful concepts (hints for HW).

When/What/Where

- Lecture: TR 2:30-4pm (4-237)
  - Lecture recordings will be made available for review
  - We cannot guarantee the quality of the recordings (extra incentive to come to class)
- Instructor
  - Cathy Wu <cathywu@mit.edu>
  - Office Hours: TR 4-4:30pm (4-237)
- Teaching assistants
  - Chanwoo Park <cpark97@mit.edu>
  - Gilhyun Ryou <ghryou@mit.edu>
  - Office hours: Check website
- Recitations: TBD (check website)
  - First recitation: 10am tomorrow
- Staff list: <6-7920-staff@mit.edu>
  - Please include “[6.7920]" in your email subject line

Course pointers

- **web.mit.edu/6.7920/www**
- Website: lecture materials & general info
- Piazza: announcements, collab, HW, recitation, solutions, readings
- Gradescope: submit HW
- Psetpartners: find pset partners

Grading

- 5 homework assignments (30%)
  - More at the beginning, sparser later
  - Mix of theoretical and computational problems
  - **Best advice: start early**
- 1 in-class quiz (25%)
  - Coverage: first 13 lectures
- Class project (35%)
  - Research-level project of your choice.
  - Form groups of 1-2 students, you're welcome to start early!
  - Class presentation + final report
- Class participation (10%)
  - Participation during lecture; answering questions on Piazza; attending office hours and recitation
- Late policy: 4 late days across all homeworks
  - Solutions for homework will be released shortly after the deadline (late submitters must abide by honor code)

References

1. Some slides adapted from Alessandro Lazaric (FAIR/INRIA)
2. DPOC vol 1, 1.1-1.3, 2.1