Lecture 1: Introduction to RL

Chunlin Chen & Zhi Wang

Department of Control and Systems Engineering
Nanjing University

Nov. 5th, 2019

Table of Contents

What we'll cover

- Finite Markov decision processes
- Tabular RL algorithms:
  - Dynamic programming
  - Monte-Carlo methods
  - Temporal difference learning
- Deep RL algorithms:
  - Policy gradients
  - Actor-critics
  - Value function-based methods
- State-of-the-arts:
  - Transfer RL, meta-RL
  - Evolution strategies for RL
  - Model-based RL
  - Hierarchical RL
  - Inverse RL

Useful links and references

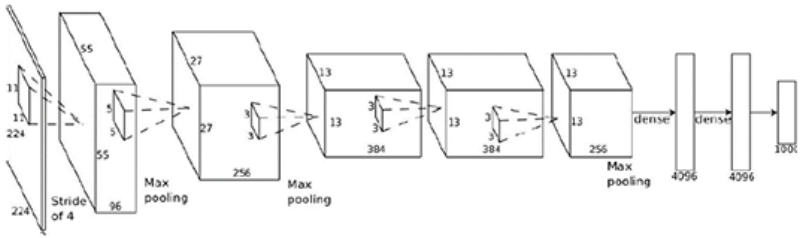
- Course website
  - `http://heyuanmingong.github.io/rl_lecture/index.html`
- Book
  - Richard S. Sutton, and Andrew G. Barto, *Reinforcement Learning: An Introduction.* 2nd Edition, `http://202.119.32.195/cache/2/03/incompleteideas.net/5b682cef88335157bc5542267cf3f442/RLbook2018.pdf`
- Famous open classes
  - CS 285 at UC Berkeley, Deep Reinforcement Learning, `http://rail.eecs.berkeley.edu/deeprlcourse/`
  - CS 234 at Stanford, Reinforcement Learning, `http://web.stanford.edu/class/cs234/`

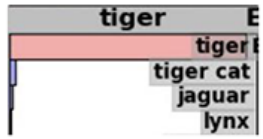
What is RL, and why we should care?

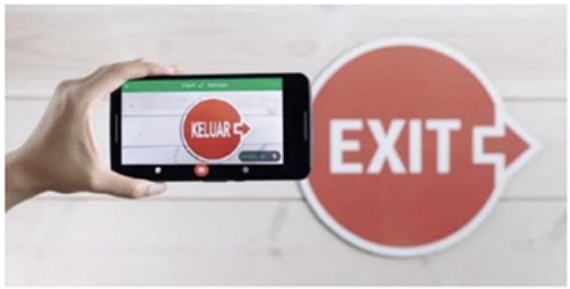
- Intelligent machines must able to adapt

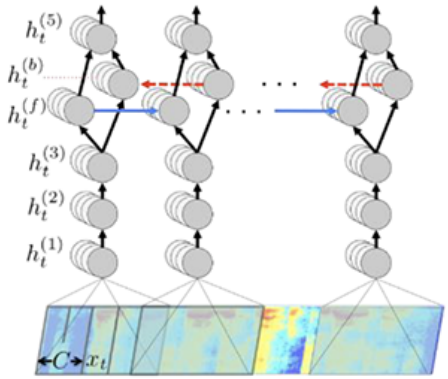
What is RL, and why we should care?

- Deep learning helps us handle *unstructured environments*



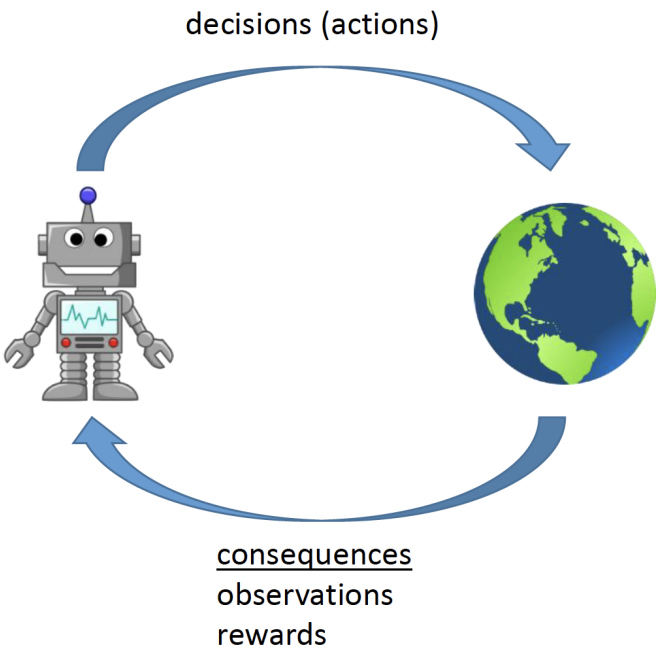


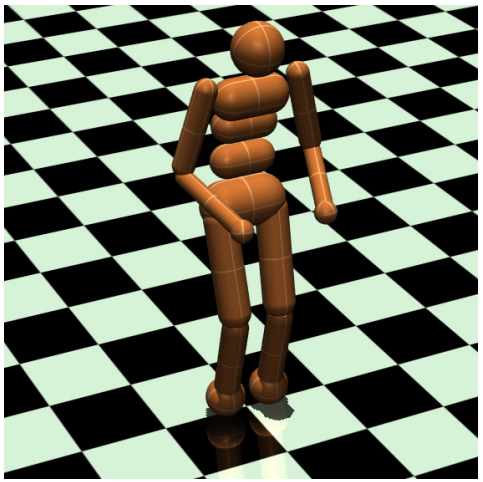


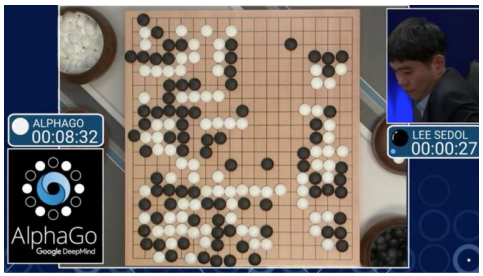


What is RL, and why we should care?

- RL provides a formalism for behavior, decision-making

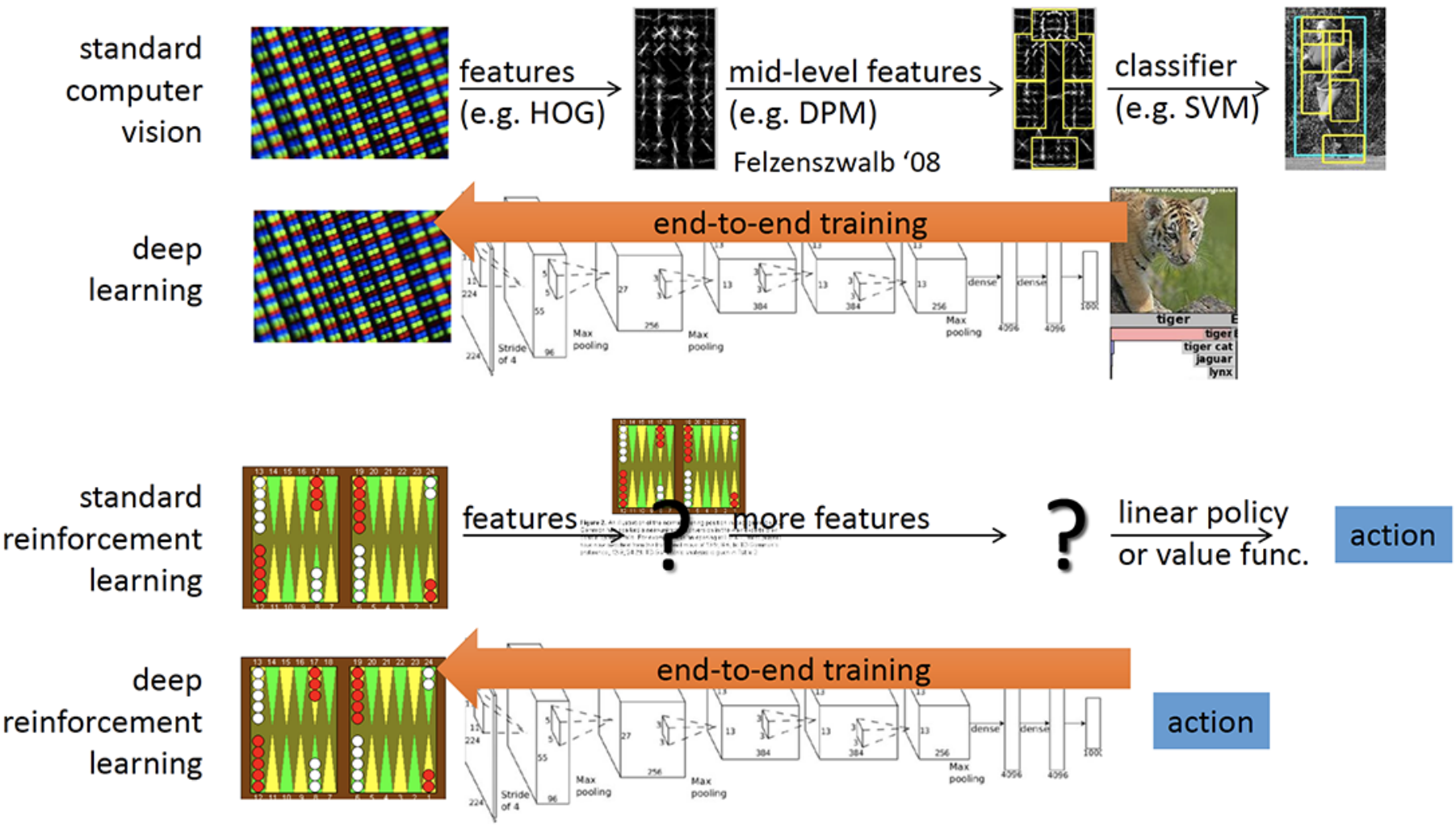






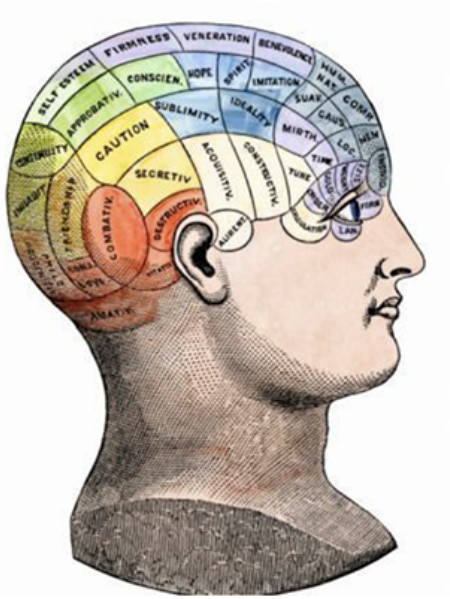
Space Invaders

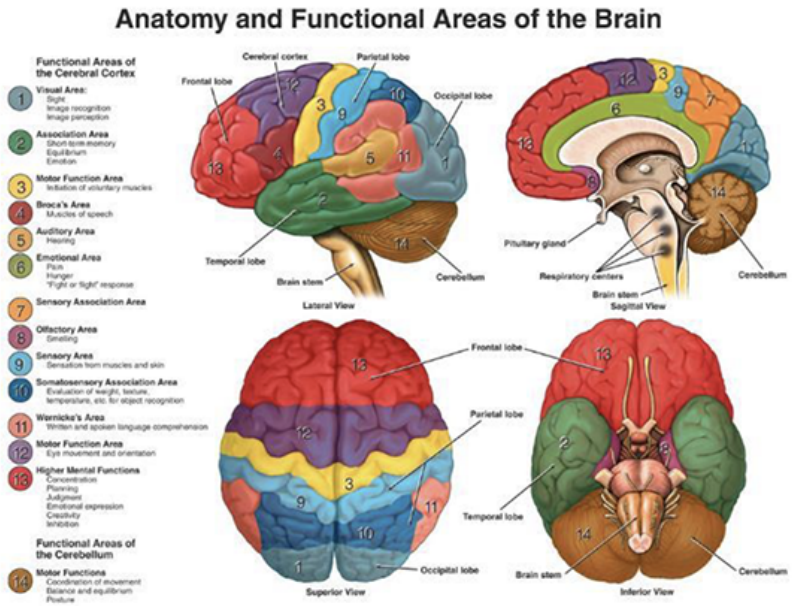
How to use RL?


standard computer vision
features (e.g. HOG)
mid-level features (e.g. DPM)
Felzenszwalb '08
classifier (e.g. SVM)
deep learning
end-to-end training
tiger
tiger cat
jaguar
lynx
standard reinforcement learning
features
more features
?
linear policy or value func.
action
deep reinforcement learning
end-to-end training
action

How do we build intelligent machines?

- Imagine you have to build an intelligent machine, where do you start?



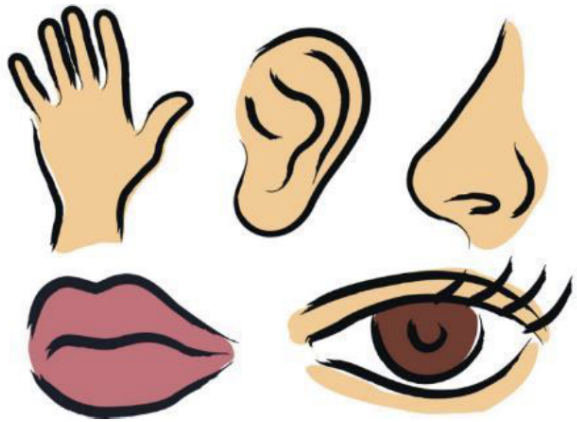


Learning as the basis of intelligence

- Some things we can all do (e.g. walking)
- Some things we can only learn (e.g. driving a car)
- We can learn a huge variety of things, including very difficult things
- Therefore our learning mechanism(s) are likely powerful enough to do everything we associate with intelligence
  - But it may still be very convenient to "hard code" a few really important bits

What must a single algorithm do?

- Interpret (rich) sensory inputs



- Choose (complex) actions

Some evidence for RL

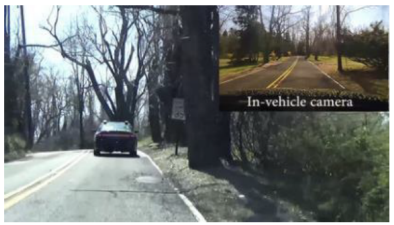
Reinforcement learning in the brain

Yael Niv
Psychology Department & Princeton Neuroscience Institute, Princeton University

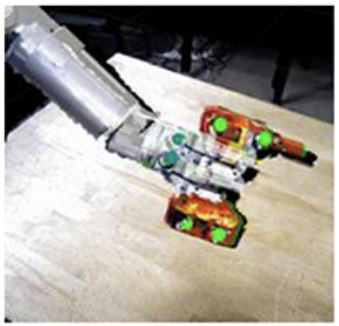
- Percepts that anticipate reward become associated with similar firing patterns as the reward itself
- Basal ganglia appears to be related to reward system
- Model free RL like adaptation is often a good fit for experimental data of animal adaptation (but not always)

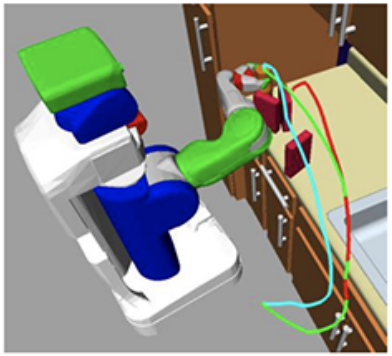
What can (deep) RL do?

- Acquire high degree of proficiency in domains governed by simple, known rules
- Learn simple skills with raw sensory inputs, given enough experience
- Learn from imitating enough human provided expert behavior



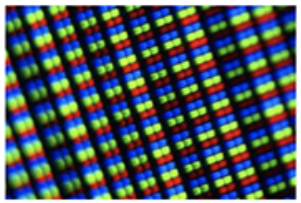
Example: Robotics

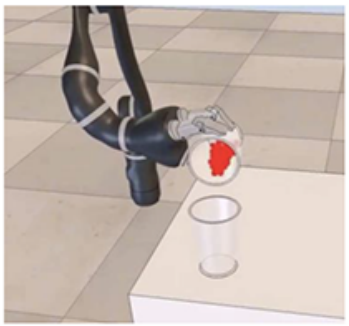


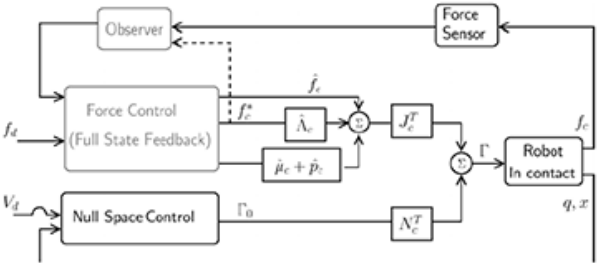


robotic control pipeline

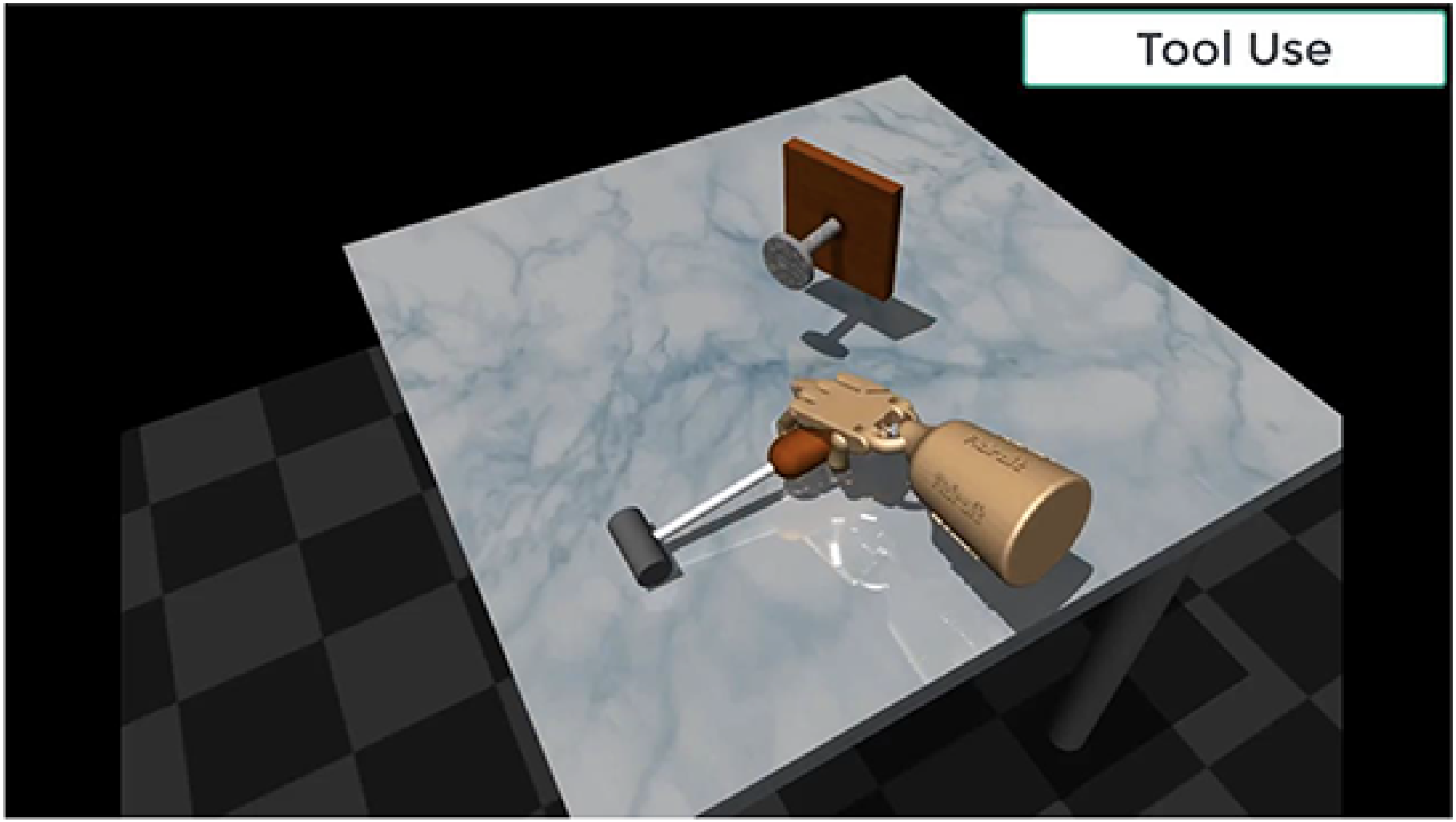
observations → state estimation (e.g. vision) → modeling & prediction → planning → low-level control → controls



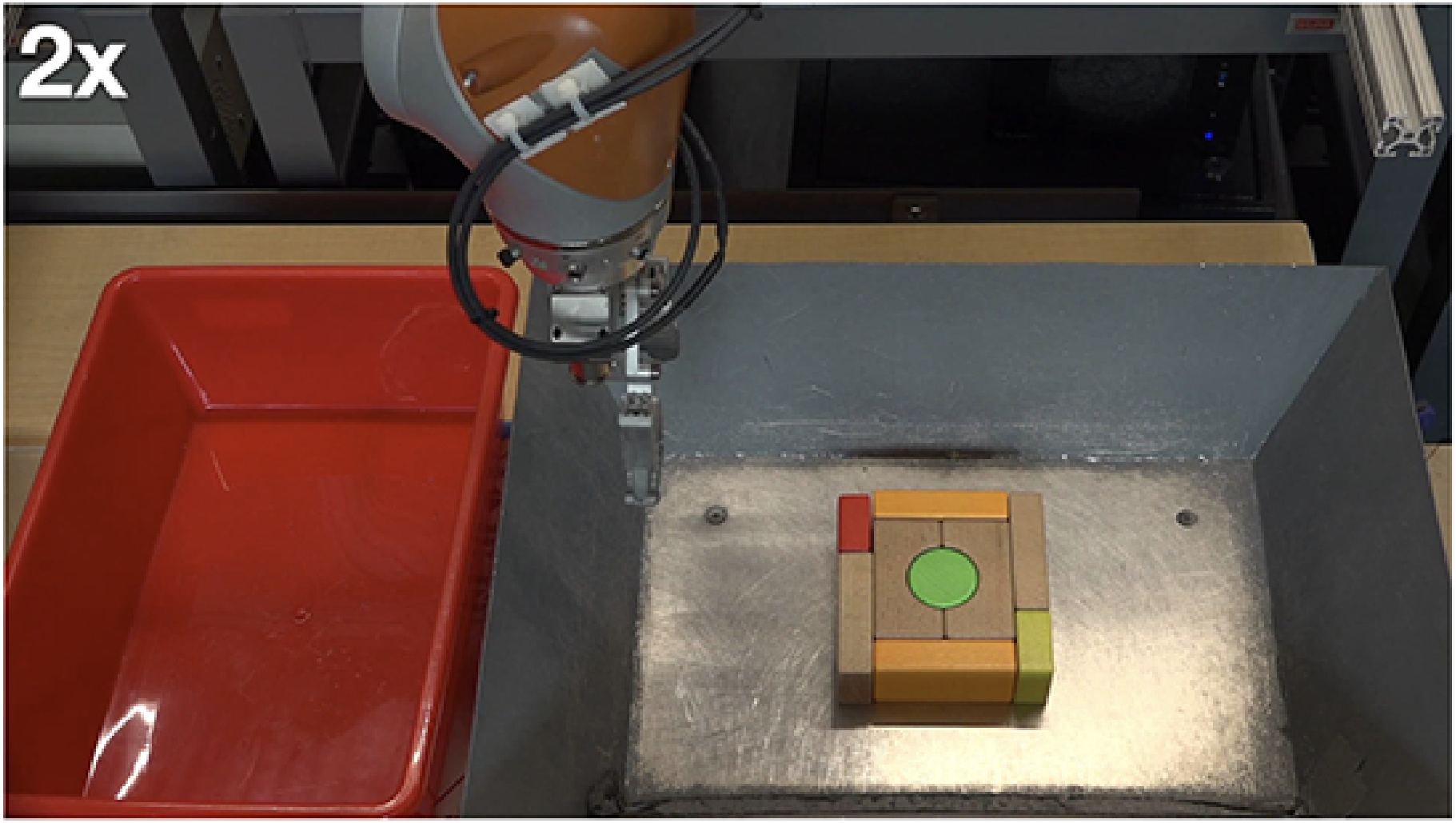




Example: Complex physical tasks



Example: Real-world scenarios



Why should we study this now?


Atari games:

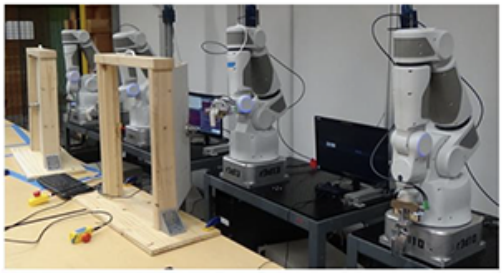
Q-learning:

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, et al. "Playing Atari with Deep Reinforcement Learning". (2013).

Policy gradients:

J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. "Trust Region Policy Optimization". (2015).

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, et al. "Asynchronous methods for deep reinforcement learning". (2016).



Real-world robots:

Guided policy search:

S. Levine*, C. Finn*, T. Darrell, P. Abbeel. "End-to-end training of deep visuomotor policies". (2015).

Q-learning:

D. Kalashnikov et al. "QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation". (2018).

Beating Go champions:

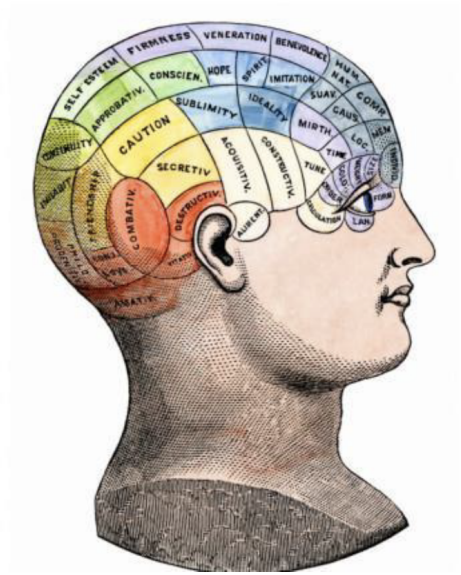
Supervised learning + policy gradients + value functions + Monte Carlo tree search:

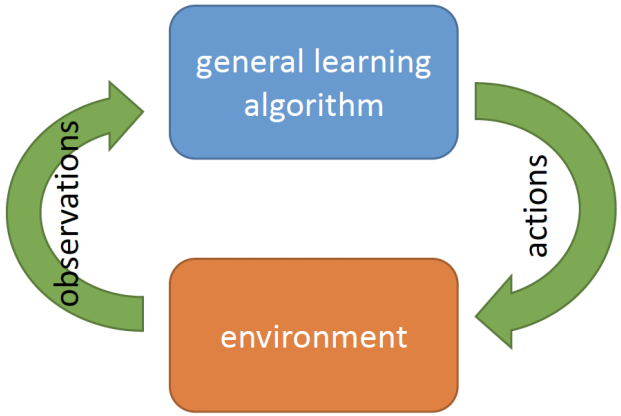
D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, et al. "Mastering the game of Go with deep neural networks and tree search". Nature (2016).

RL, decision making, human behavior

Instead of trying to produce a program to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain.

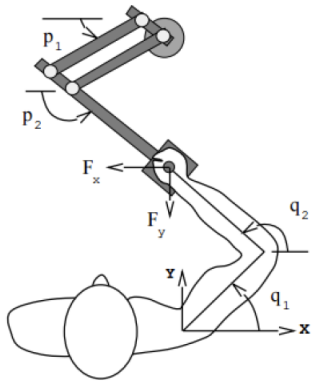
- Alan Turing

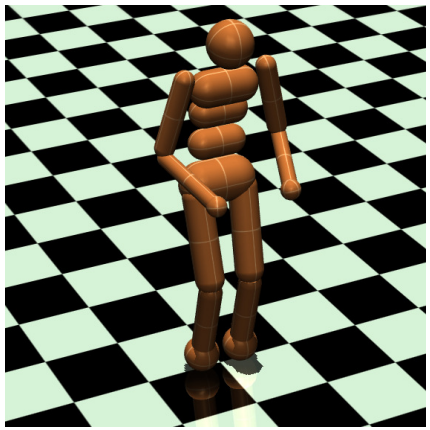




What has proven challenging so far?

- Humans can learn incredibly quickly
  - (Deep) RL methods are usually slow
- Humans can reuse past knowledge
  - Transfer learning in (deep) RL is an open problem
- Not clear what the reward function should be
- Not clear what the role of prediction should be





To enable real-world sequential decision making – Beyond learning from reward

- Basic RL deals with maximizing rewards
- This is not the only problem that matters for sequential decision making!
- More advanced topics
  - Transferring knowledge between domains (transfer learning, meta learning)
  - Learning to predict and using prediction to act (model-based RL)
  - Learning reward functions from example (inverse RL)

Table of Contents

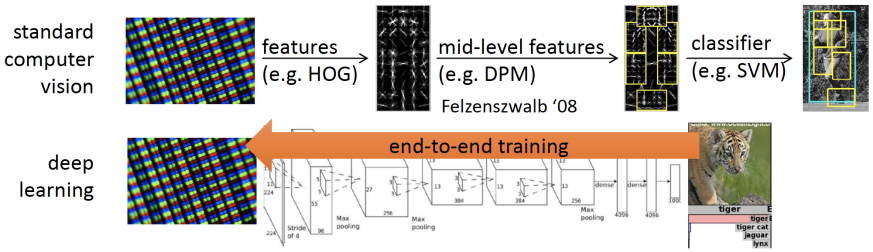
Markov Decision Process (MDP)

- A classical formalization of sequential decision making
  - Choosing different actions in different situations
  - Actions influence not just immediate rewards, but also subsequent situations through future rewards
  - Involve delayed reward and the need to tradeoff immediate and delayed reward
- A mathematically idealized form of the RL problem
  - Precise theoretical statements can be made
  - Key elements of the problem's mathematical structure, such as returns, value functions, Bellman equations, etc
  - A tension between breadth of applicability and mathematical tractability

Supervised learning vs. Sequential decision making

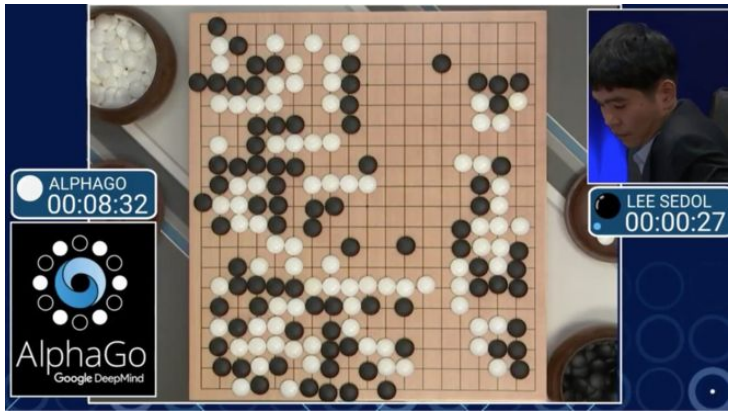
Supervised learning

- Samples are independent and identically distributed (i.i.d.)
- Given an input, map an optimal output

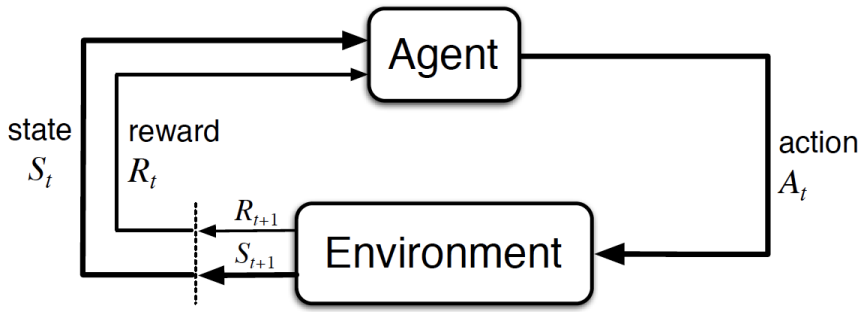


Reinforcement learning

- Samples are not i.i.d., temporally co-related
- Given an initial state, find a sequence of optimal actions



The agent-environment interface



- **Agent:** The learner, decision maker
- **Environment:** The thing it interact with, comprising everything outside the agent

At each time step $t = 0, 1, 2, ...,$ the agent...

- receives some representation of the environment's **state**, $S_t \in \mathcal{S}$
- on that basis, selects an **action**, $A_t \in \mathcal{A}(s)$
- one time step later, receives a numerical **reward**, $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$
- finds itself in a new state, S_{t+1}, (**transition function**)
- $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3...$

Dynamics of the MDP $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$

$$p(s', r|s, a) = Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$$

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) = 1, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$$

- The probabilities given by p completely characterize the environment's dynamics
- **Markov Property**
  - The probability of each possible value for S_t and R_t depends only on the immediately preceding state S_{t-1} and action A_{t-1}, not at all on earlier states and actions
  - $p(S_t, R_t|S_{t-1}, A_{t-1}) = p(S_t, R_t|S_{t-1}, A_{t-1}, S_{t-2}, A_{t-2}, S_{t-3}, A_{t-3}, ...)$
- Recall supervised learning $p(X_i|X_j) = 0$

Dynamics of the MDP $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$

- The probabilities given by p completely characterize the environment's dynamics

$$p(s', r|s, a) = Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$$

IEEE/ASME TRANSACTIONS ON MECHATRONICS, VOL. 24, NO. 2, APRIL 2019 621

Incremental Reinforcement Learning With Prioritized Sweeping for Dynamic Environments

Zhi Wang, Chunlin Chen, *Member, IEEE*, Han-Xiong Li, *Fellow, IEEE*, Daoyi Dong, *Senior Member, IEEE*, and Tzyh-Jong Tarn, *Life Fellow, IEEE*

III. Incremental Reinforcement Learning

As an example, imagine a search-and-rescue robot to detect injured people inside damaged buildings. The falling bricks or steels may block the shortest path to the wounded that the robot has already learned. Since relearning the new environment from scratch is too time-consuming, it is reasonable and effective to revise the optimal policy incrementally after learning the changed part first. In this type of problems, incremental learning can be achieved and the learning process can be accelerated to adapt to the dynamic environment. The concept of IRL has been initialized in our conference paper [37] and a formal framework for IRL is presented in detail with related techniques and specific algorithms in this section. In particular, we consider dynamic environments where the reward functions may change over time.

IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS 1

Incremental Reinforcement Learning in Continuous Spaces via Policy Relaxation and Importance Weighting

Zhi Wang, *Student Member, IEEE*, Han-Xiong Li, *Fellow, IEEE*, and Chunlin Chen, *Member, IEEE*

A. Problem Formulation

We consider the dynamic environment as a sequence of stationary tasks on a certain timescale where each task corresponds to the specific type of environment characteristics during the associated time period. Assume there is a space of MDPs, $\mathcal{M}$, and an infinite underlying distribution, $\mathcal{D}$, over time in $\mathcal{M}$. An RL agent interacts with the dynamic environment $\mathcal{D} = \{M_1, \ldots, M_{t-1}, M_t, \ldots\}$, where each $M_t \in \mathcal{M}$ denotes the specific MDP that is stationary during the tth time period. We assume, in this paper, that the environment changes only in the reward and state transition functions, but keeps the same state and action spaces.

Goals and rewards

- **Reward Hypothesis**
  - That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward).

- The agent's goal: maximize the total amount of reward it receives
  - maximize $J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1}\right]$
  - Maximize not immediate reward, but cumulative reward in the long run

Returns and episodes

- The subsequence of the agent-environment interaction, **episodes**
  - $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, ..., S_{T-1}, A_{T-1}, R_T, S_T$
  - Each episode ends in a special state called the terminal state, S_T
- We seek to maximize the **expected return**, G_t, the reward sequence
  - $G_t = R_{t+1} + R_{t+2} + ... + R_T$
  - Episodic tasks

Discount rate $\gamma \in [0, 1]$

- The expected **discounted** return
  - $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} ... = \sum_{k=1}^{\infty} \gamma^k R_{t+k+1}$
- The discount rate determines the present value of future rewards: a reward received k time steps in the future is worth only γ^{k-1} times what it would be worth if it were received immediately
- $\gamma \to 0$, the agent is "myopic", only maximizing immediate rewards
  - Akin to supervised learning that maximizes the log-likelihood of each sample, $\log p(y_i|x_i)$
- $\gamma \to 1$, the agent is "farsighted", taking future rewards into account
- Returns at successive time steps are related to each other

$$\begin{aligned} G_t &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + ...) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Policies and value functions

- **Value functions**: function of states or state-action pairs
  - Estimate how good it is for the agent to be in a given state, or how good it is to perform a given action in a given state
  - "How good": defined in expected future rewards, i.e., expected return
  - Depend on what actions to take, defined w.r.t. particular ways of acting, called **policies**
- **Policy**: $\pi(a|s)$, a mapping from states to probabilities of selecting each possible action
  - $\sum_a \pi(a|s) = 1$
  - e.g., for a given state s_1, four possible actions $a_1 = 0.1, a_2 = 0.3, a_3 = 0.2, a_4 = 4$
- RL = estimate value functions + estimate polices + estimate both

Policies and value functions

- v_π, the **state-value function** for policy π

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left[\sum_{k=1}^{\infty} \gamma^k R_{t+k+1}|S_t = s\right], \forall s \in \mathcal{S}$$

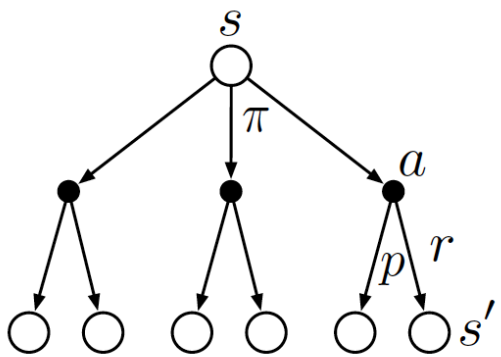
- Q_π, the **action-value function** for policy π

$$\begin{aligned} Q_\pi(s,a) &= \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \\ &= \mathbb{E}_\pi\left[\sum_{k=1}^{\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a\right], \forall s \in \mathcal{S}, a \in \mathcal{A}(s) \end{aligned}$$

A fundamental property: Bellman equation

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1} = s']] \\
&= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]
\end{aligned}
$$

- Average over all the possibilities, weighting each by its probability of occurring
- Express the relationship between the value of a state and the values of its successor states
- Transfer value information back to a state from its successor states



Backup diagram for v_π

Example: Gridworld

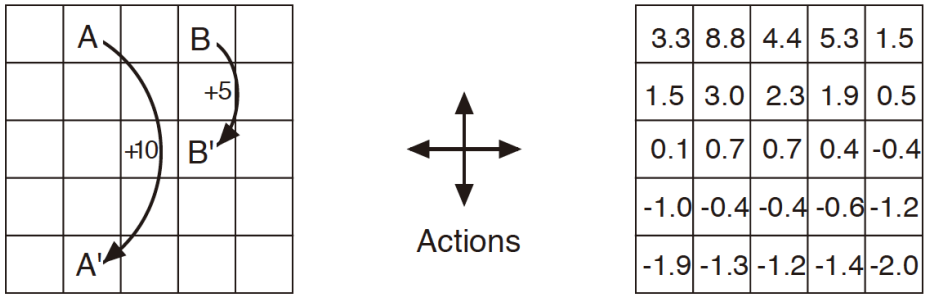


Figure 3.3: Gridworld example: exceptional reward dynamics (left) and state-value function for the equiprobable random policy (right).

$$v_\pi(s) = \sum_a \pi(a|s)[r + \gamma v_\pi(s')]$$

- Actions that would take the agent off the grid leave its location unchanged, but also result in $r = -1$, otherwise $r = 0$
- From state A, all four actions yield $r = 10$ and take the agent to A'
- From state B, all actions yield $r = 5$ and the agent to B'

Optimal policies and optimal value functions

- RL tasks: find a policy that achieves a lot of reward over the long run
- Value functions define a partial ordering over policies
  - $\pi \geq \pi'$ if and only if $v_\pi(s) \geq v_{\pi'}(s), \forall s \in \mathcal{S}$
- At least one policy that is better than or equal to all other policies, i.e., **optimal policy**
- Optimal policies, π^*, share the same **optimal value function**

$$v^*(s) = \max_\pi v_\pi(s), \quad \forall s \in \mathcal{S}$$

$$Q^*(s,a) = \max_\pi Q_\pi(s,a), \quad \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$$

$$Q^*(s,a) = \mathbb{E}[R_{t+1} + \gamma v^*(S_{t+1}) | S_t = s, A_t = a]$$

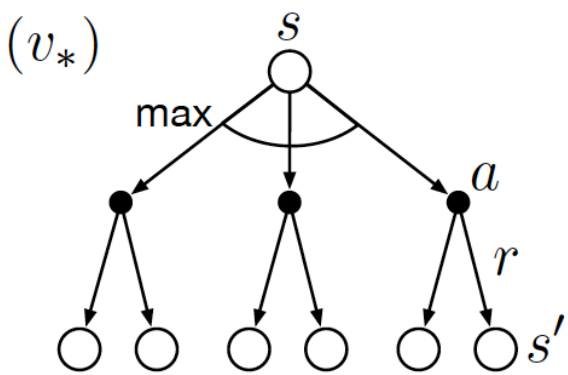
Bellman optimality equation

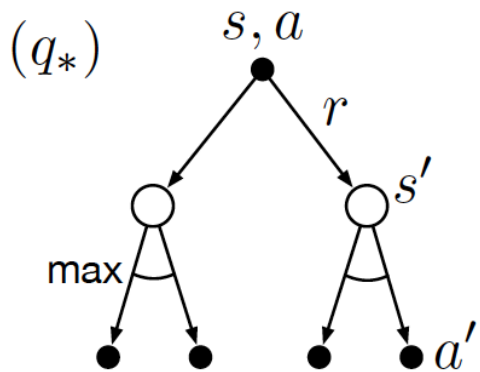
$$
\begin{aligned}
v^*(s) &= \max_a Q_{\pi^*}(s,a) \\
&= \max_a \mathbb{E}_{\pi^*}[G_t | S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi^*}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma v^*(S_{t+1}) | S_t = s, A_t = a] \\
&= \max_a \sum_{s',r} p(s', r | s, a)[r + \gamma v^*(s')]
\end{aligned}
$$

- The value of a state under an optimal policy must equal the expected return for the best action from that state

Bellman optimality equation

$$Q^*(s,a) = \mathbb{E}[R_{t+1} + \gamma \max_{a'} Q^*(S_{t+1}, a') | S_t = s, A_t = a]$$
$$= \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} Q^*(s', a')]$$



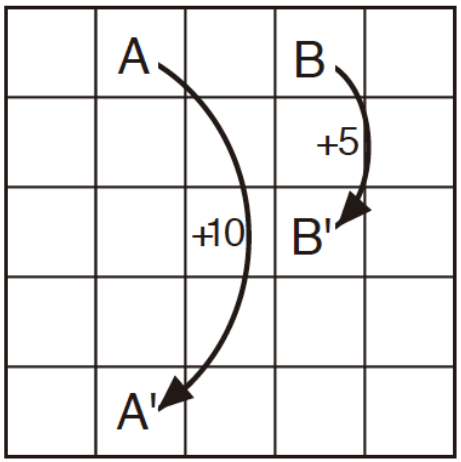


$$v^*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v^*(s')]$$

Determine optimal policies from optimal value functions

- For v^*: a one-step search
  - Actions that appear best after one-step search will be optimal actions
- For Q^*: no need to do a one-step-ahead search
  - $a^* = \arg\max_a Q^*(s, a)$
  - The optimal action-value function allows optimal actions to be selected without having to know anything about possible successor states and their values, i.e., without having to know anything about the environment's dynamics

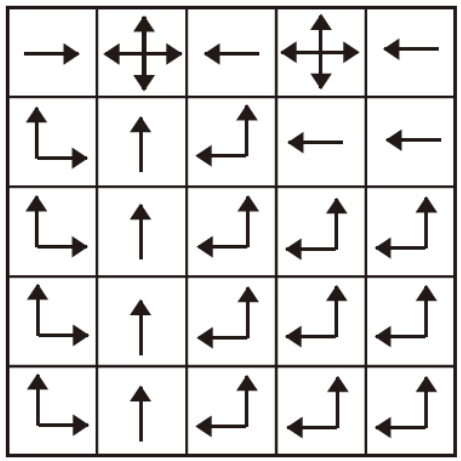
Example: Gridworld



Gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

v_*



π_*

Figure 3.6: Optimal solutions to the gridworld example.

Learning objectives of this lecture

You should be able to...

- Understand the basic concepts about sequential decision-making, the differences between supervised learning and RL
- How RL provides a formalism for behavior and decision-making

- Know elements of finite MDPs, the agent-environment interface
- (Discounted) returns and episodes, polices and value functions, Bellman (optimality) equation

References

- Lecture 1 & 2 of CS285 at UC Berkeley, *Deep Reinforcement Learning, Decision Making, and Control*
  - `http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-1.pdf`
  - `http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-2.pdf`
- Chapter 3: Finite Markov Decision Processes, *Reinforcement Learning: An Introduction*, 2nd Edition
  - `http://202.119.32.195/cache/2/03/incompleteideas.net/5b682cef88335157bc5542267cf3f442/RLbook2018.pdf`

THE END