2019-2020 Spring IERG 6130

Reinforcement Learning and Beyond
Lecture 1: Course Overview and Introduction

Bolei Zhou

The Chinese University of Hong Kong

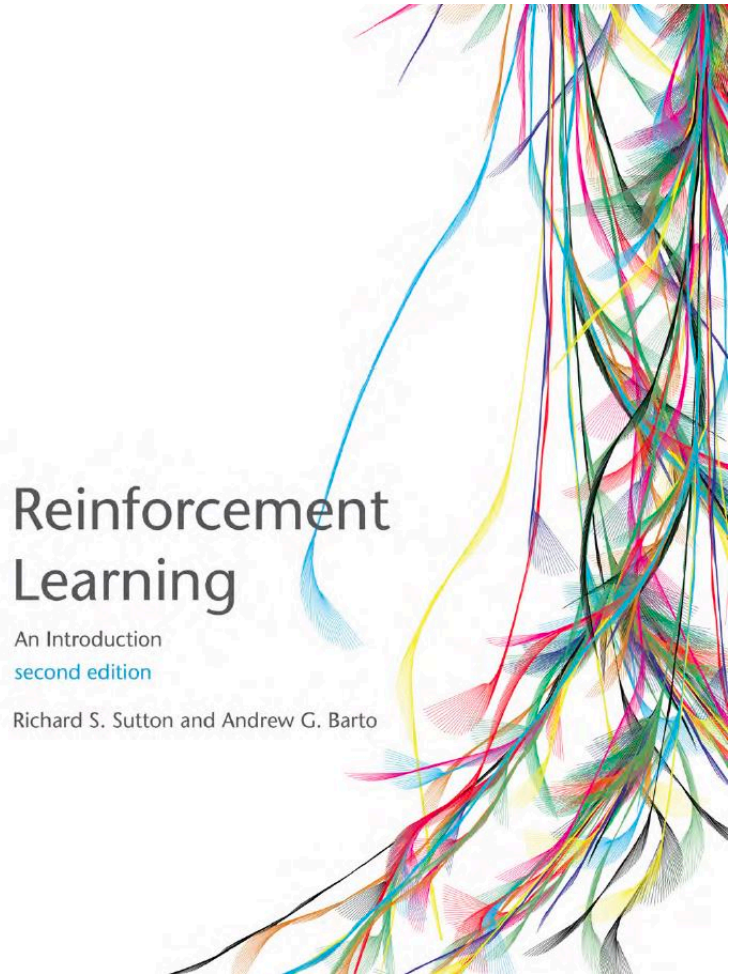
Outline

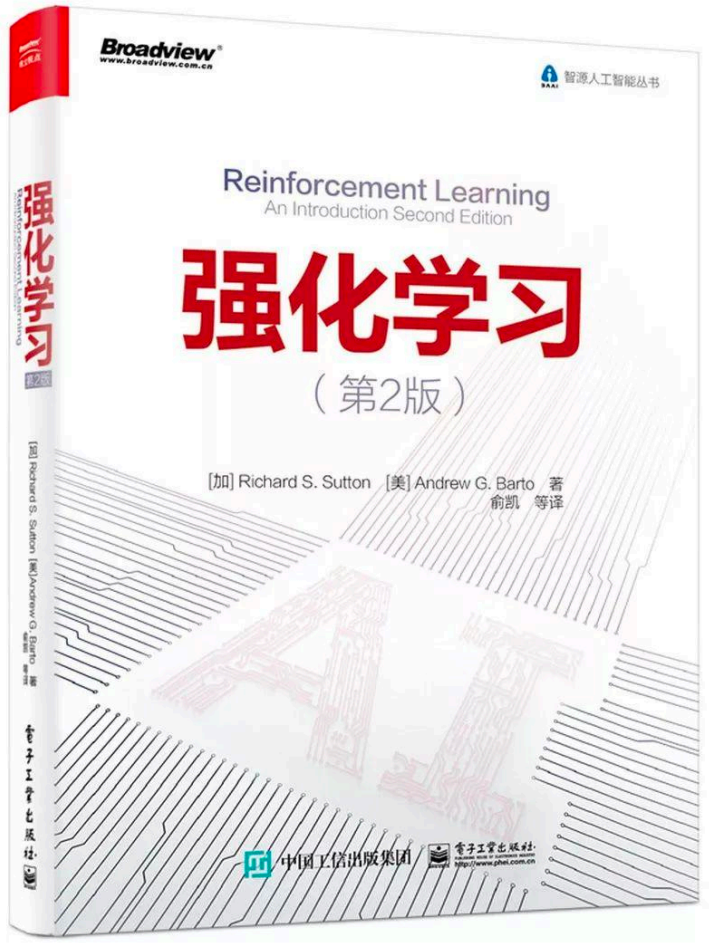
- Course logistics
- Overview of reinforcement learning
- Introduction to sequential decision making

Course Logistics

- Instructor: Bolei Zhou
- TA: Zhenghao Peng (pengzh@ie.cuhk.edu.hk)
- Time:
  - Tuesday (2:30pm – 4:10 pm): two 45-min sessions with a 10-min break
  - Wednesday (3:30pm – 4:15 pm): one 45-min session
  - Course website: https://course.ie.cuhk.edu.hk/~ierg6130/
- Office hour: Wednesday 4:30 pm – 5:30 pm at Room 717, SHB
- TA office hour: TBD

Textbook: Sutton and Barton: http://incompleteideas.net/book/the-book-2nd.html





Prerequisites & Enrollment

- This is IERG-6XXX course, which means it is a graduate level course
- All enrolled students must have taken the Linear Algebra course and Probability course, and one machine learning relevant course (data mining, pattern recognition, deep learning, etc).

- More general DL course: ELEG5491 Introduction to Deep Learning

What we will cover

- Key elements of RL: state, reward, action, Markov decision process, exploration and exploitation, etc.
- RL algorithms: Q-learning, policy gradients, actor-critic.
- Others: unsupervised learning, generative modeling
- Case studies: Projects in DeepMind, projects at OpenAI, and others.

Course Objective

- Know the difference between reinforcement learning, machine learning, and deep learning.
- Knowledge on the foundation and practice of RL
- Given your research problem (e.g. from computer vision, NLP, IoT, etc) decide if it should be formulated as a RL problem, if yes be able to define it formally (in terms of the state space, action space, dynamics and reward model), state what RL algorithm is best for addressing it, implement it!

Grading

- ~~Student-led Seminar (45-min presentation): 40%~~
- Attendance: 10%
- 4 assignments: 40%
- Course Project : 50%

Course project

- Relevant to RL or the application of RL
- Suggested projects will be posted in the coming week, but feel free to work on your own project relevant to RL.
- Expected workload of the project: 7 weeks.
- Course projects from last year: https://course.ie.cuhk.edu.hk/~ierg6130/2019/project.html
- Deliverables:
  - Proposal due (by the end of Week 3)
  - Mid-term presentation (Week 8)
  - A github repo containing your work (commits, readme, human-understandable code)
  - Final presentation (Week 14)
  - Course report (in NIPS LaTex Template), due by the end of semester

Course Schedule: https://course.ie.cuhk.edu.hk/~ierg6130/schedule.html

Part 1: Tabular methods, RL foundation

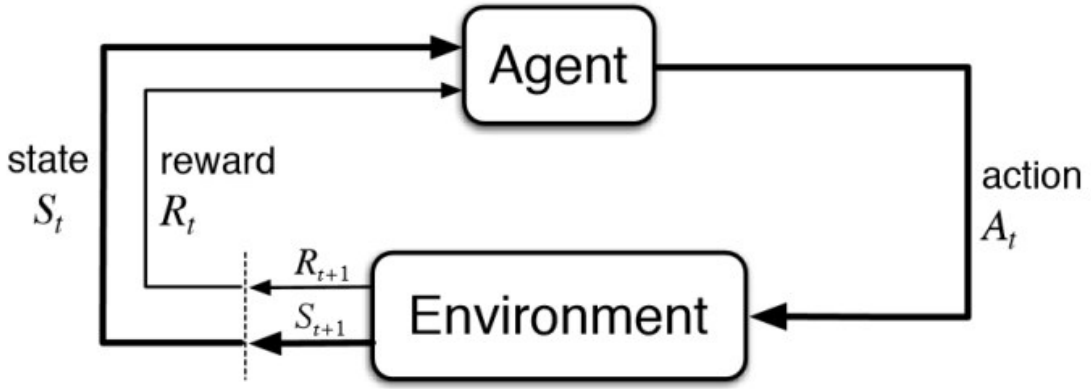
Part 2: Approximate methods

Part 3: Policy optimization

Part 4: Advanced topics

Week	Time	Topic	Materials
Week 1	Tu 2:30pm-4:15pm	Course overview and introduction of RL	
	Wed 3:30pm-4:15pm	Get hand dirty by coding	
Week 2	Tu 2:30pm-4:15pm	Markov decision process	
	Wed 3:30pm-4:15pm	Policy iteration and value iteration	HW1 out
Week 3	Tu 2:30pm-4:15pm	Model-free prediction	
	Wed 3:30pm-4:15pm	Model-free control	Project proposal due
Week 4	-	Chinese New Year Holiday	
	-	Chinese New Year Holiday	
Week 5	Tu 2:30pm-4:15pm	A recap and value function approximation	
	Wed 3:30pm-4:15pm	Deep Q Network	HW1 due, HW2 out
Week 6	Tu 2:30pm-4:15pm	Policy optimization I	
	Wed 3:30pm-4:15pm	Policy optimization II	
Week 7	Tu 2:30pm-4:15pm	Actor-critic and SOTA of Policy Gradient	
	Wed 3:30pm-4:15pm	Imitation learning	HW2 due, HW3 out
Week 8	Tu 2:30pm-4:15pm	**Student project mid-term presentation**	
	Wed 3:30pm-4:15pm	Model-based RL	
Week 9	Tu 2:30pm-4:15pm	Generative modeling	
	Wed 3:30pm-4:15pm	Guest lecture	HW3 due, HW4 out
Week 10	Tu 2:30pm-4:15pm	Exploitation and Exploration	
	Wed 3:30pm-4:15pm	Inverse reinforcement learning	
Week 11	Tu 2:30pm-4:15pm	GameAI, AlphaGo series, AlphaStar	
	Wed 3:30pm-4:15pm	Sample-efficiency	HW4 due
Week 12	Tu 2:30pm-4:15pm	Distributed computing and RL system design	
	Wed 3:30pm-4:15pm	RL interpretation	
Week 13	Tu 2:30pm-4:15pm	Course Summary	
	Wed 3:30pm-4:15pm	Course Summary	
Week 14	Tu 2:30pm-4:15pm	**Student project final presentation**	
	Wed 3:30pm-4:15pm	**Student project final presentation**	

What is reinforcement learning and why we care



a computational approach to learning whereby an agent tries to maximize the total amount of reward it receives while interacting with a complex and uncertain environment.

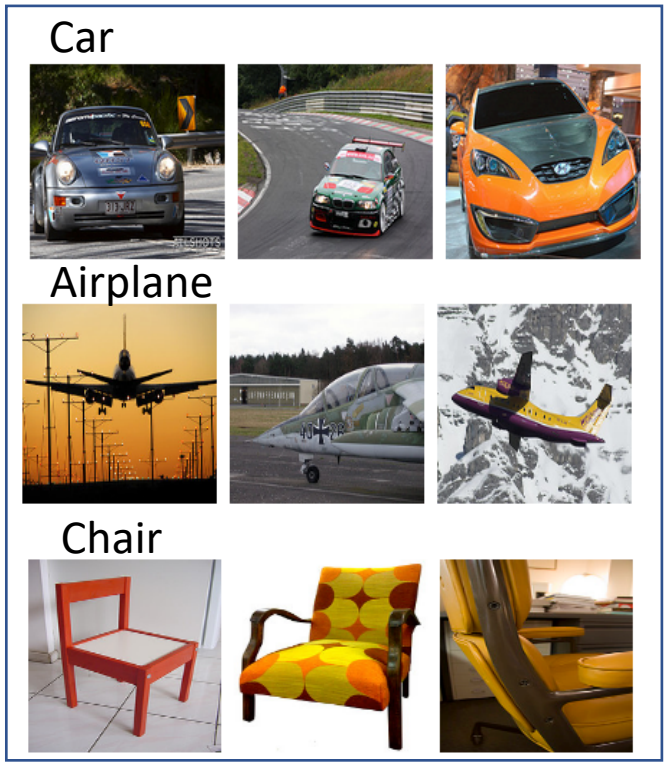
- Sutton and Barto

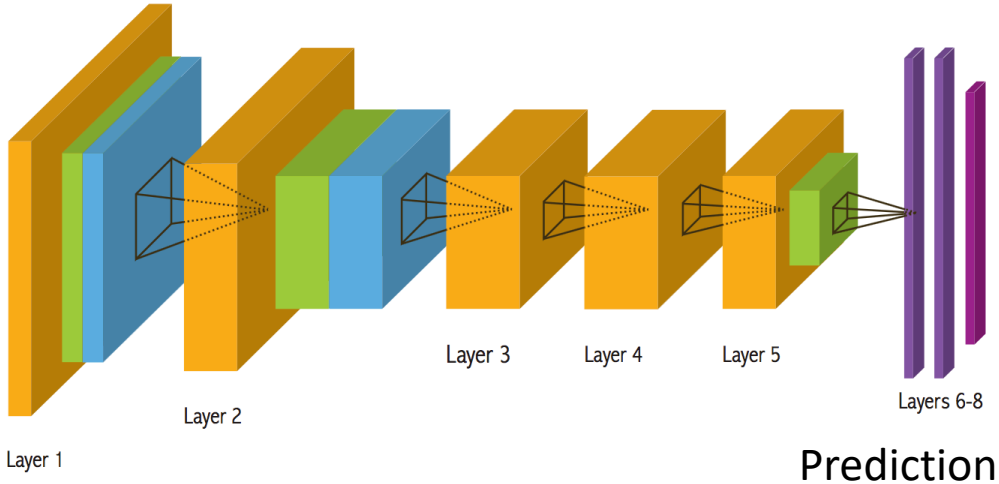
Supervised Learning

Learn to classify object

- Annotated images, data follows i.i.d distribution.
- Learners are told what the labels are.

Training annotated data





Prediction: Airplane

Prediction is wrong
Correct label: Car

Back-propagation

Reinforcement Learning

Learn to play Breakout

- Data are not i.i.d. Instead, a correlated time series data
- No instant feedback or label for correct action

Action: Move LEFT or Right

Atari Game: Breakout

Reinforcement Learning

Training data

Step1

Step2

Step3

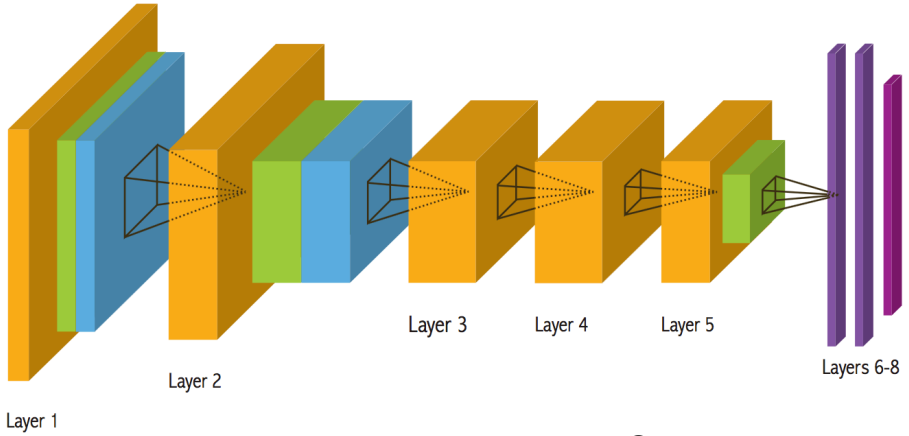
Step4

Step5

. . .

Human playing?

Step3



Action:
Move LEFT

Correct or Wrong???
Don't know for now, until game is over
Delayed reward

Backpropagation?

Difference between Reinforcement Learning and Supervised Learning

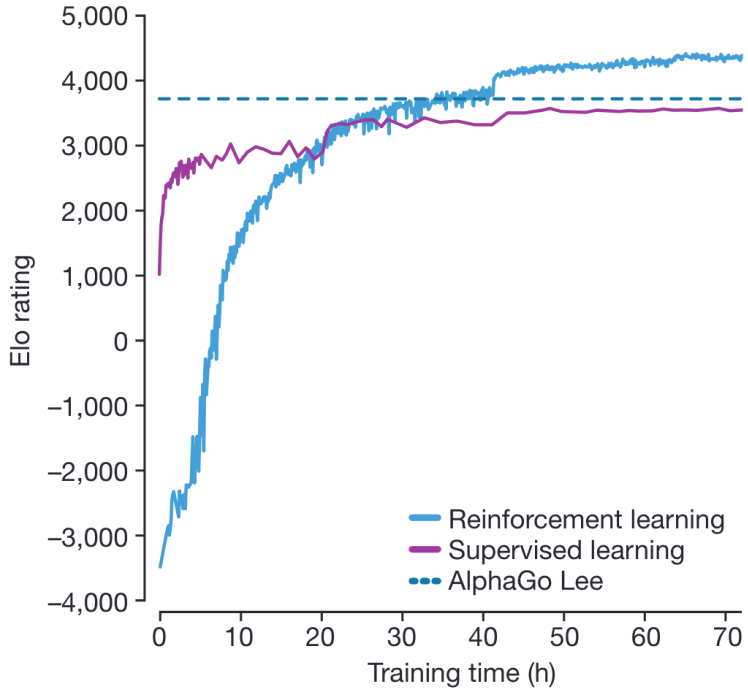
- Sequential data as input (not i.i.d)
- The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them.
- Trial-and-error exploration (balance between exploration and exploitation)
- There is no supervisor, only a reward signal, which is also delayed

Features of reinforcement learning

- Trial-and-error exploration
- Delayed reward
- Time matters (sequential data, non i.i.d data)
- Agent’s actions affect the subsequent data it receives (agent’s action changes the environment)

Big deal: May Achieve Superhuman AI

- Upper bound for supervised learning is human-performance.
- Upper bound for reinforcement learning?

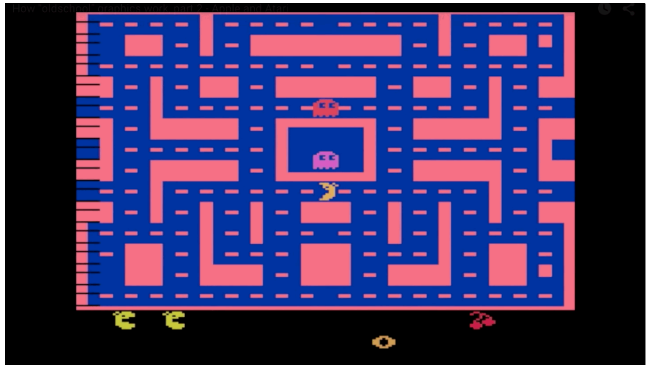


AlphaGo vs. AlphaGo Zero

Silver and Schrittwieser, et al. Mastering the game of Go without human knowledge, Nature'17

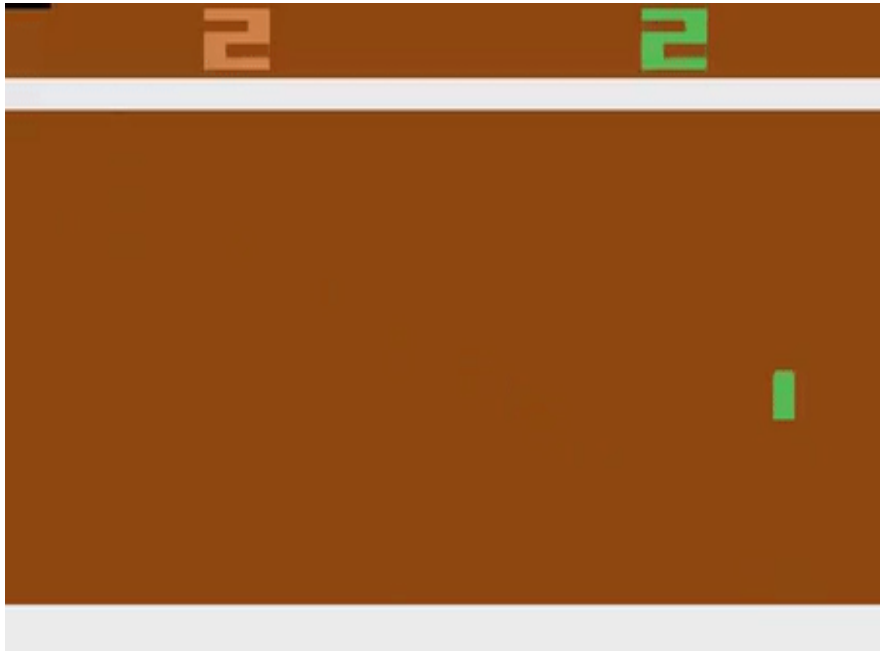
Examples of reinforcement learning

- A chess player makes a move: the choice is informed both by planning-anticipating possible replies and counterreplies.
- A gazelle calf struggles to stand, 30 min later it is able to run 36 kilometers per hour.
- Portfolio management.
- Playing Atari game



RL example: Pong

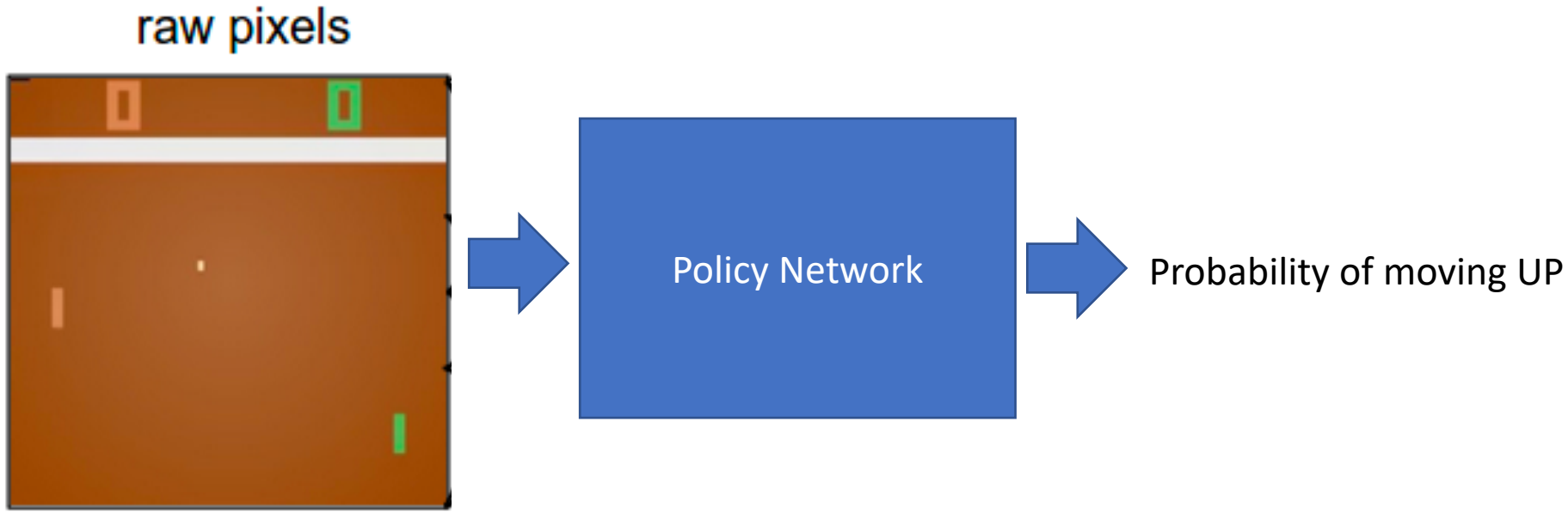
Action: move UP or DOWN



From Andrej Karpathy blog: http://karpathy.github.io/2016/05/31/rl/

RL example: Pong

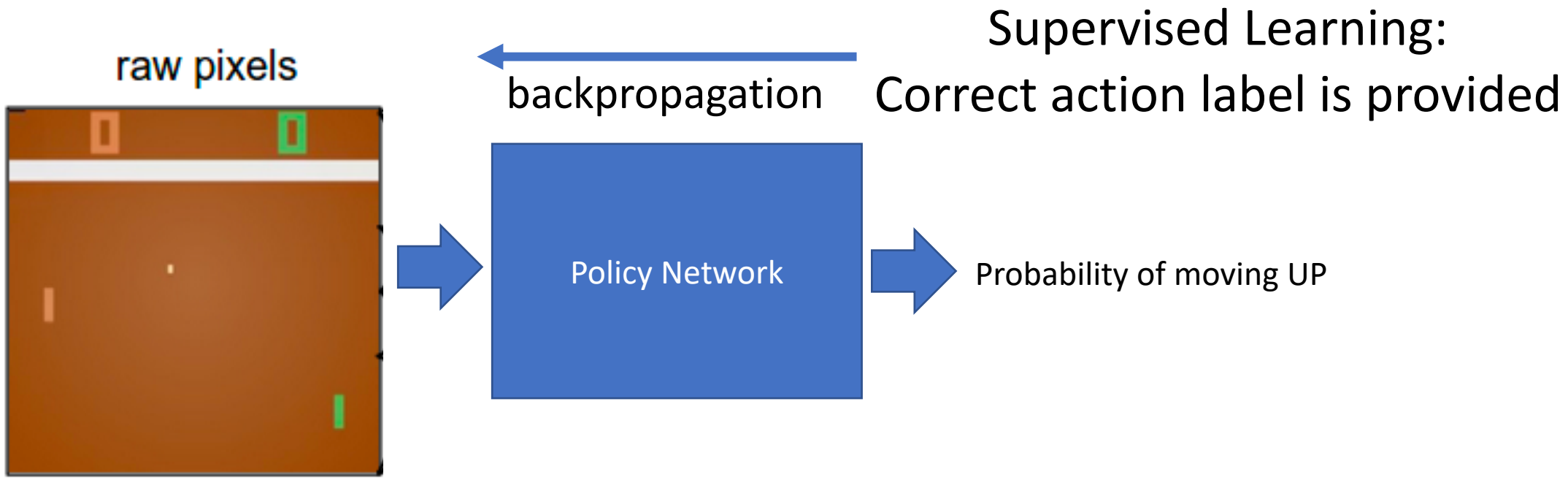
Action: move UP or DOWN



From Andrej Karpathy blog: http://karpathy.github.io/2016/05/31/rl/

RL example: Pong

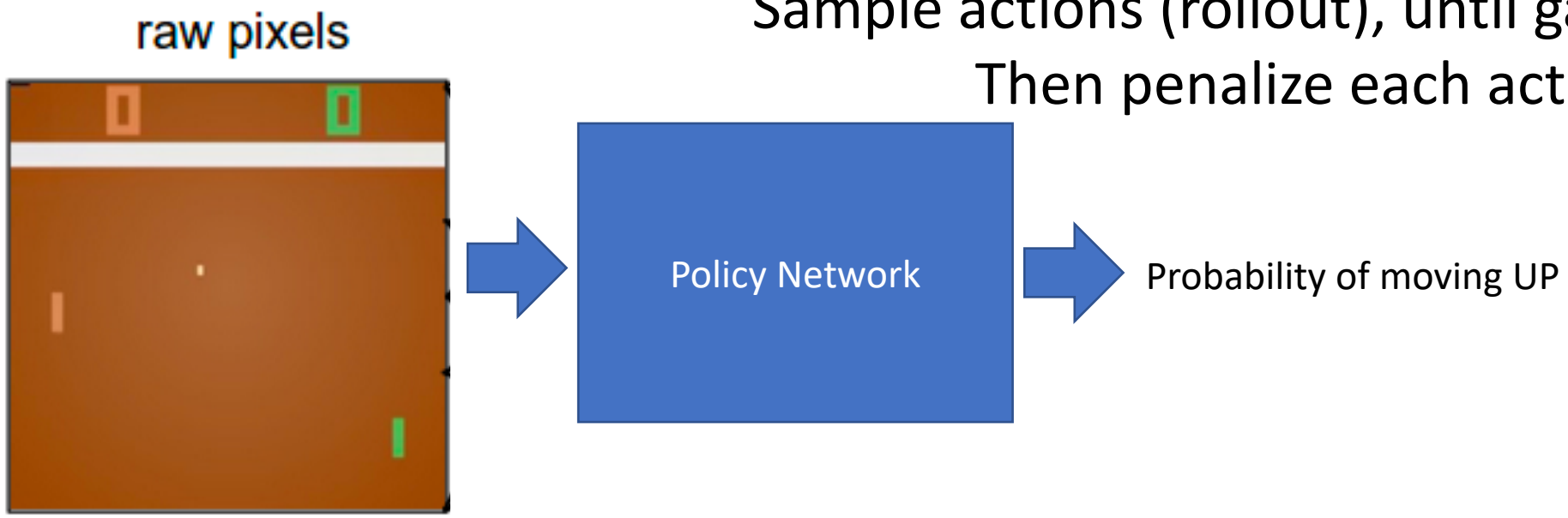
- Action: move UP or DOWN



RL example: Pong

- Action: move UP or DOWN

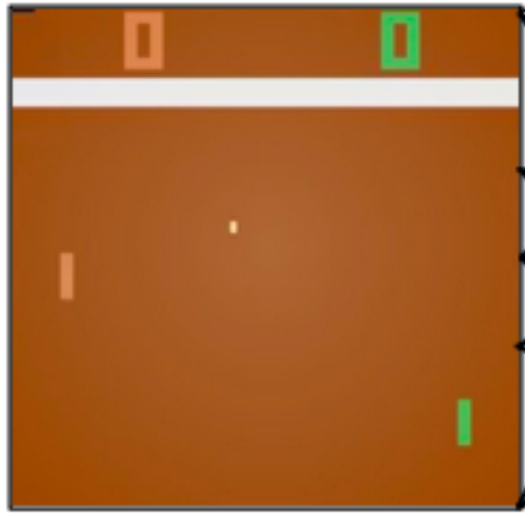
Reinforcement Learning:
Sample actions (rollout), until game is over,
Then penalize each action



RL example: Pong

- Action: move UP or DOWN

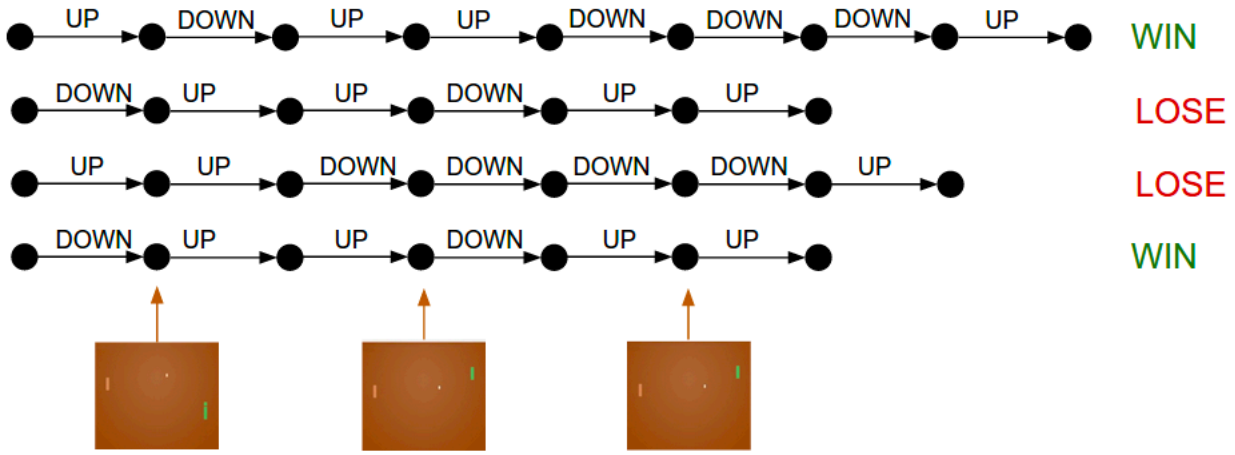
raw pixels



Reinforcement Learning:
Sample actions (rollout), until game is over,
Then penalize each action

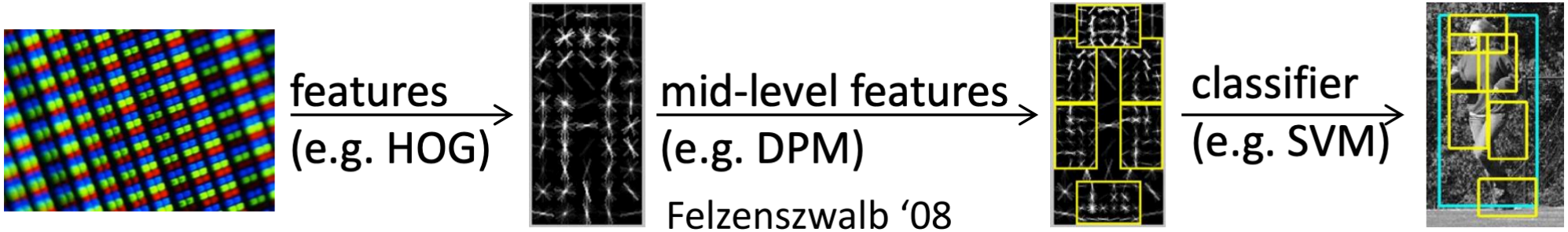
Possible rollout sequence:

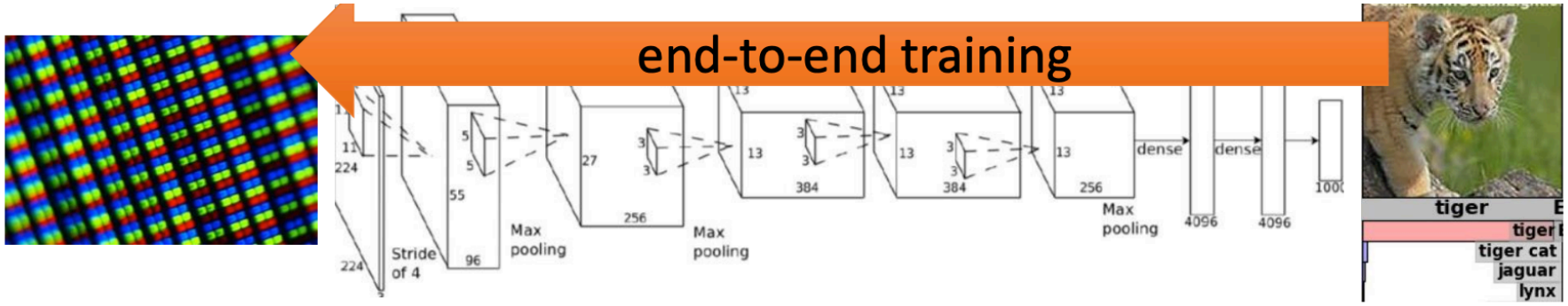
Eventual Reward:



Why deep reinforcement learning?

- Analogy to traditional CV and deep CV

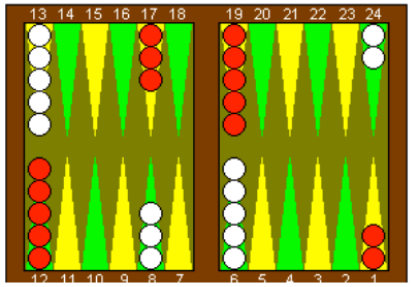




Why deep reinforcement learning?

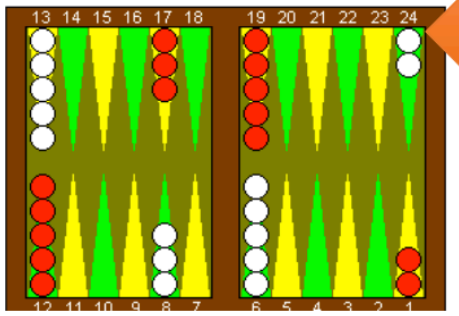
- Standard RL and deep RL

TD-Gammon, 1995

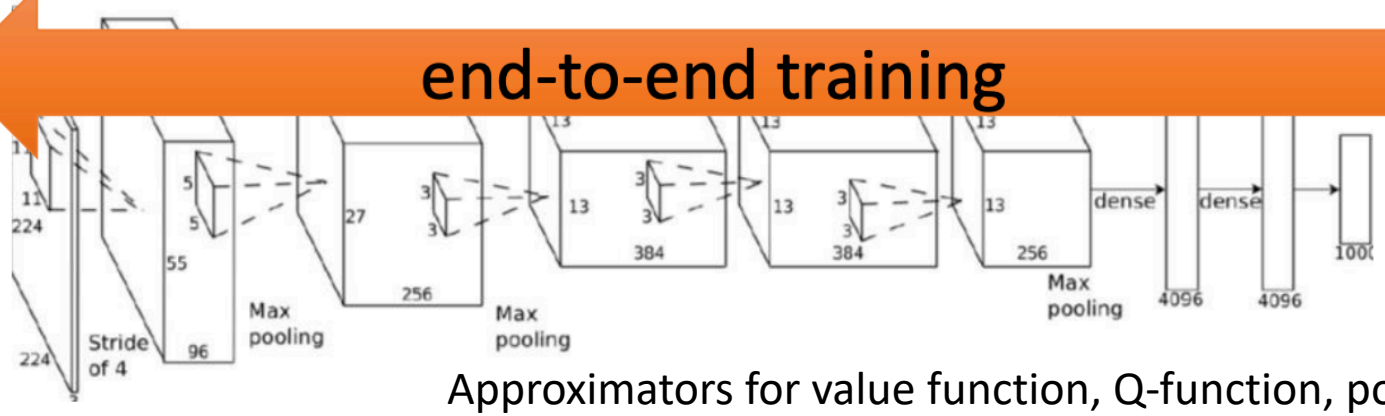


game of backgammon

features → ? → more features → ? → linear policy or value func. → action



end-to-end training



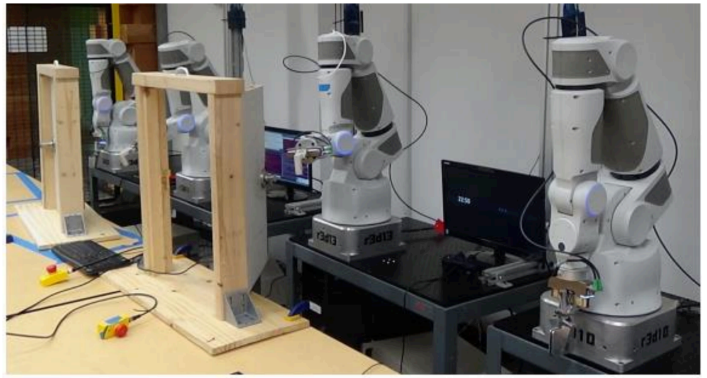
action

Approximators for value function, Q-function, policy networks

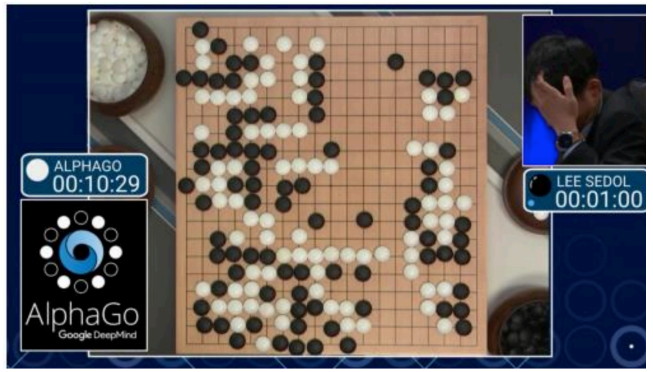
Why RL works now?

- One of the most exciting areas in machine learning

Game playing



Robotics



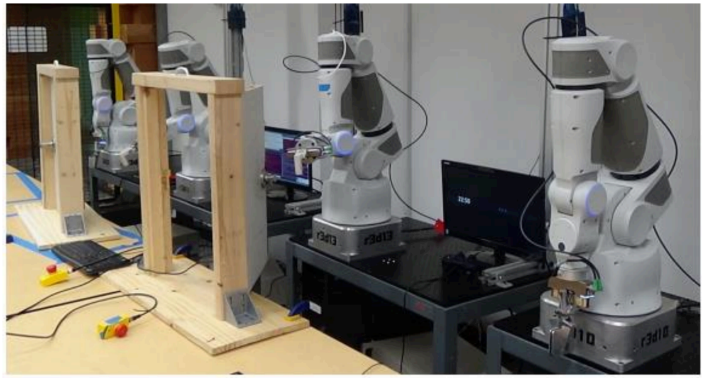
Beating best human player

[Playing Atari with Deep Reinforcement Learning]
[Mastering the game of Go without Human Knowledge]

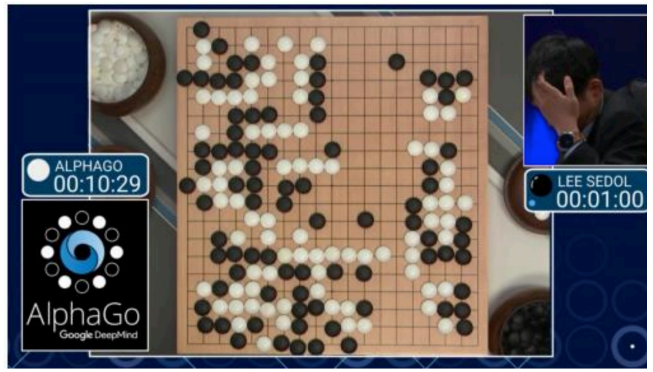
Why RL works now?

- Computation power: many GPUs to do trial-and-error rollout
- Acquire the high degree of proficiency in domains governed by simple, known rules
- End-to-end training, features and policy are jointly optimized toward the end goal.

Game playing



Robotics

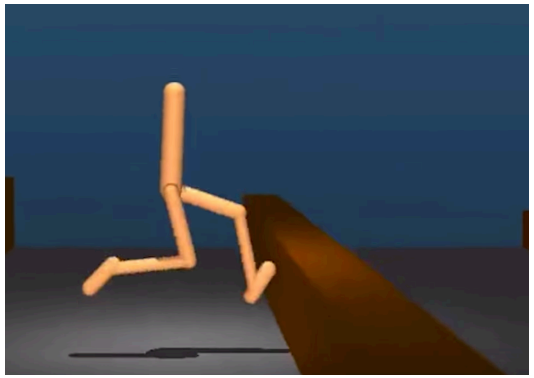


Beating best human player

What are the applications of RL?

Some interesting examples:

Learning to walk

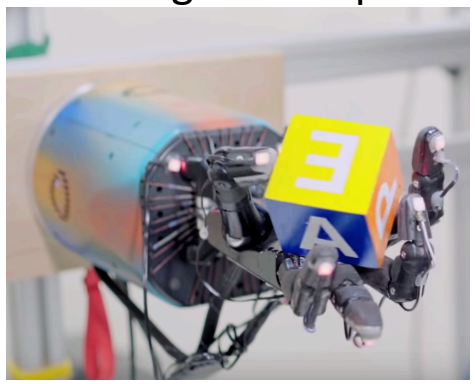


https://www.youtube.com/watch?v=gn4nRCC9TwQ

Learning to grasp

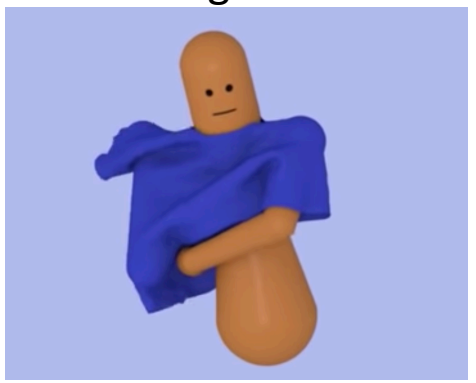
https://ai.googleblog.com/2016/03/deep-learning-for-robots-learning-from.html

Learning to manipulate



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

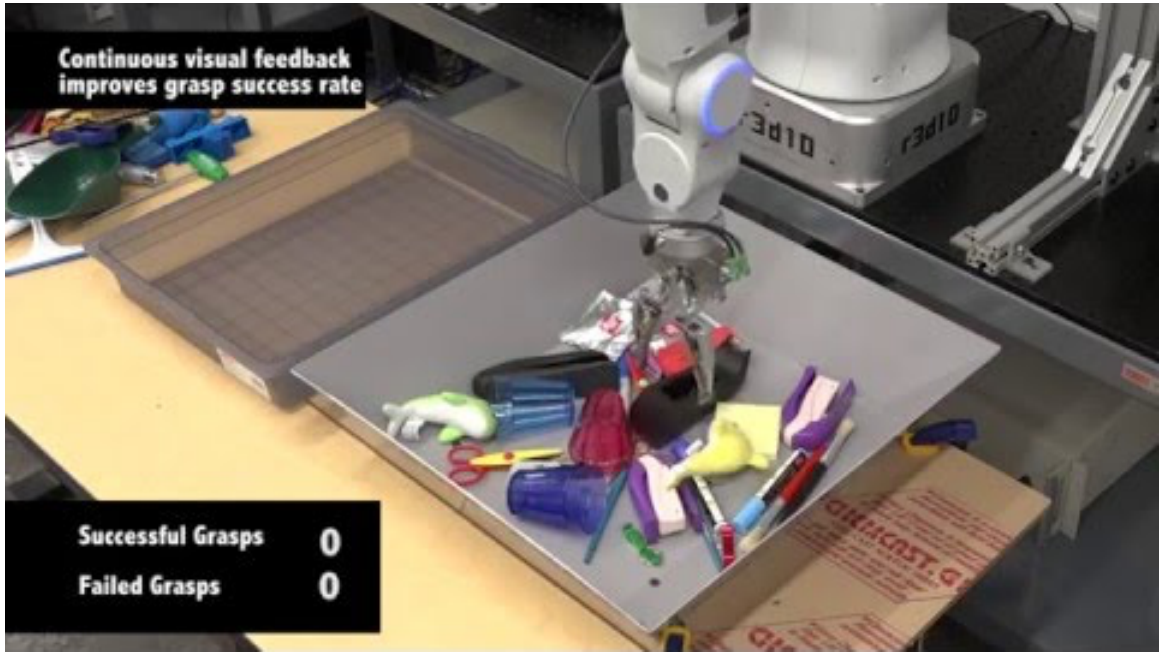
Learning to dress



https://www.youtube.com/watch?v=ixmE5nt2o88

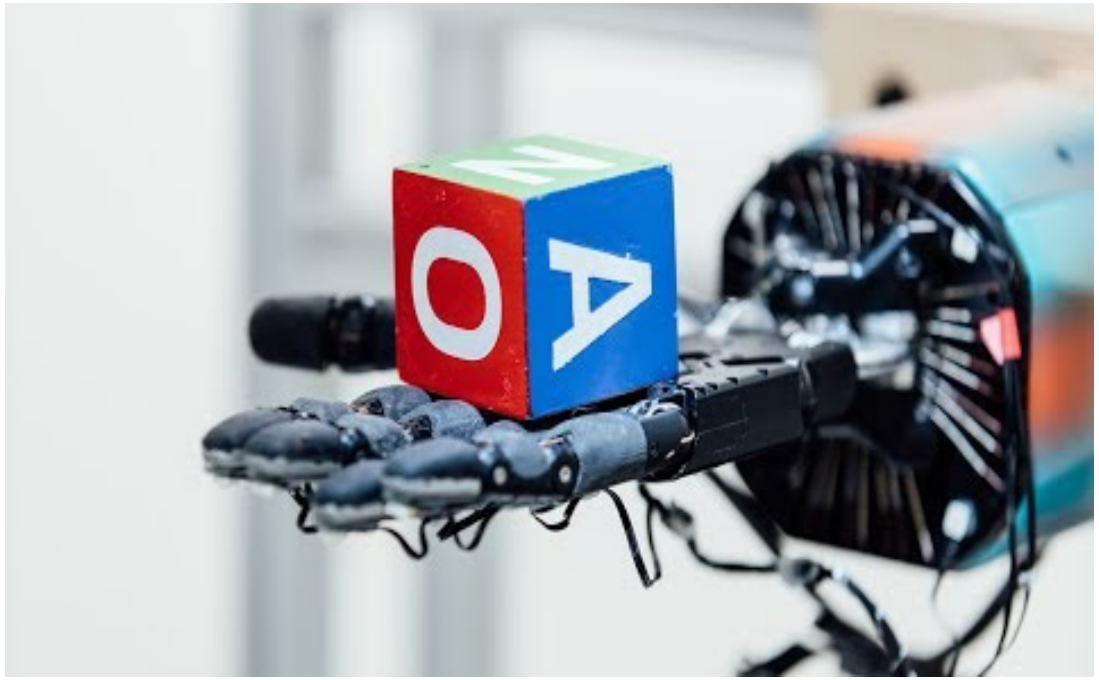
Learning to walk

Learning to grasp

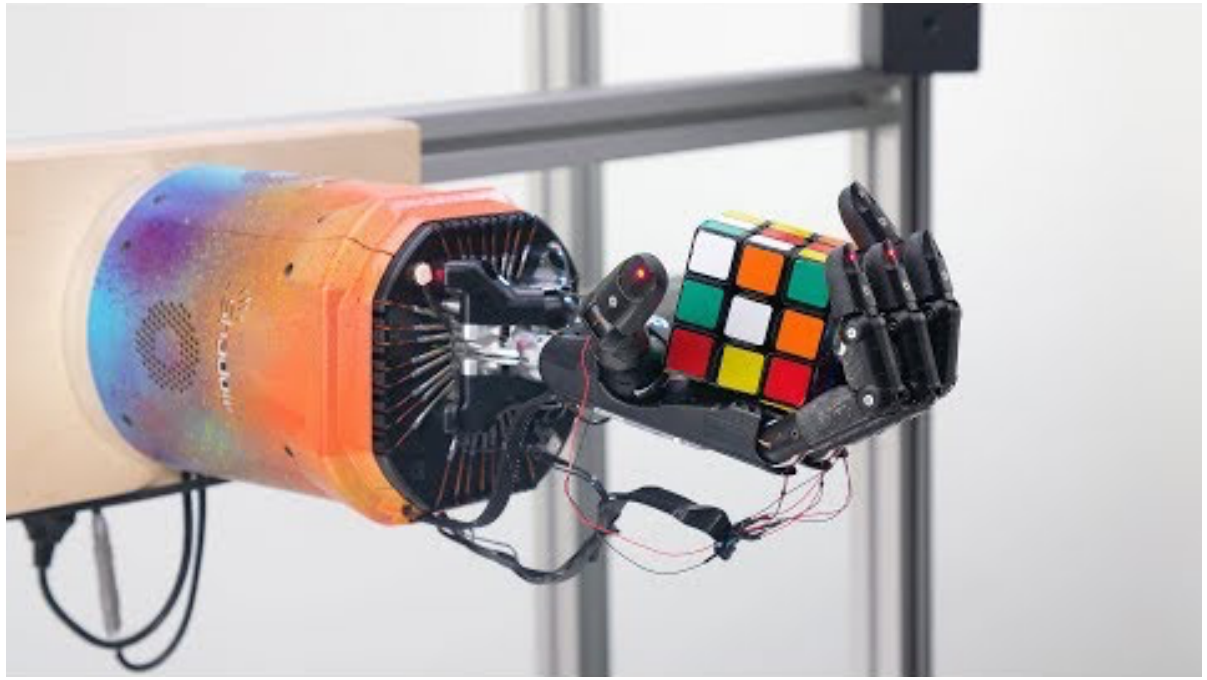


Learning to move fingers

2018



2019

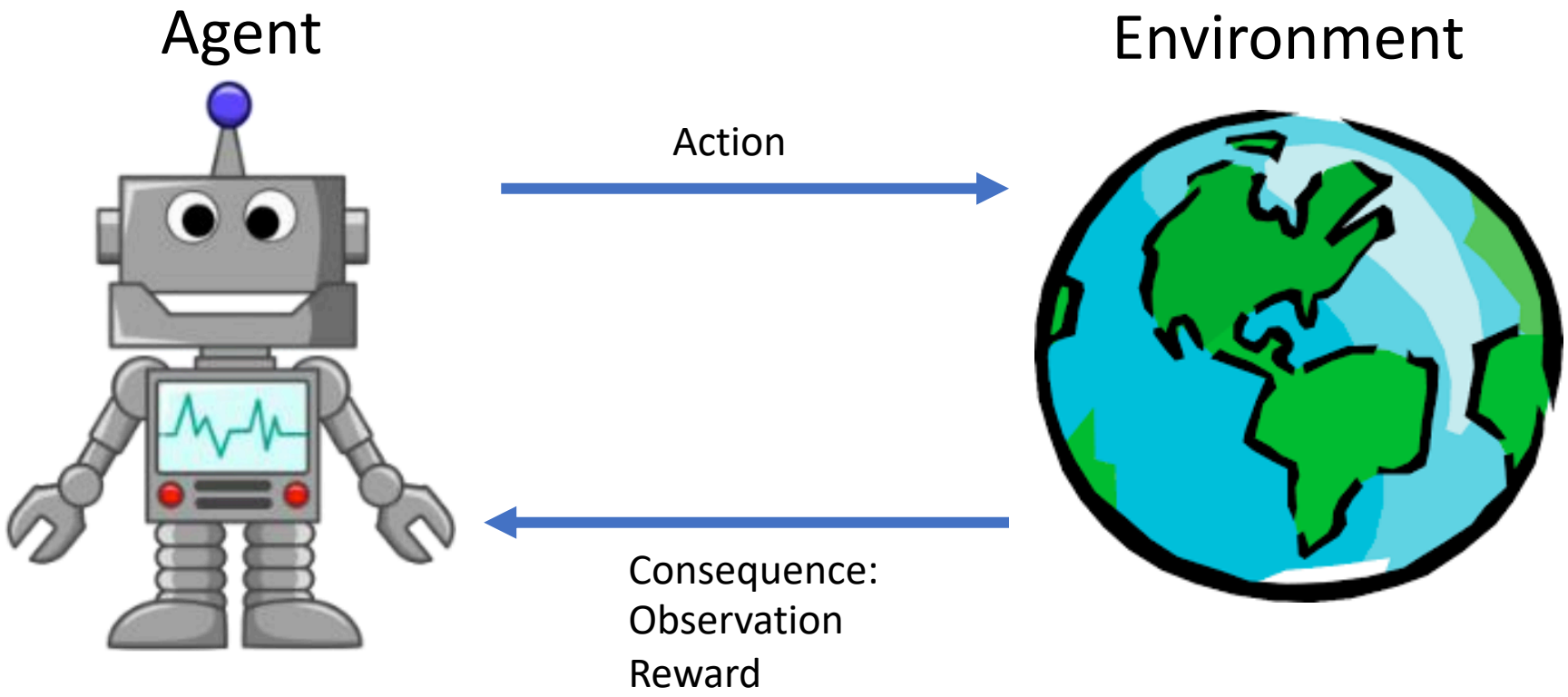


OpenAI

Learning to dress

Agent and Environment

- The agent learns to interact with the environment



Rewards

- A reward is a scalar feedback signal
- Indicate how well agent is doing at step t
- Reinforcement Learning is based on the maximization of rewards:

All goals of the agent can be described by the maximization of expected cumulative reward.

Examples of Rewards

- Chess players play to win:

+/- reward for wining or losing a game

- Gazelle calf struggles to stand:

+/- reward for running with its mom or being eaten

- Manage stock investment

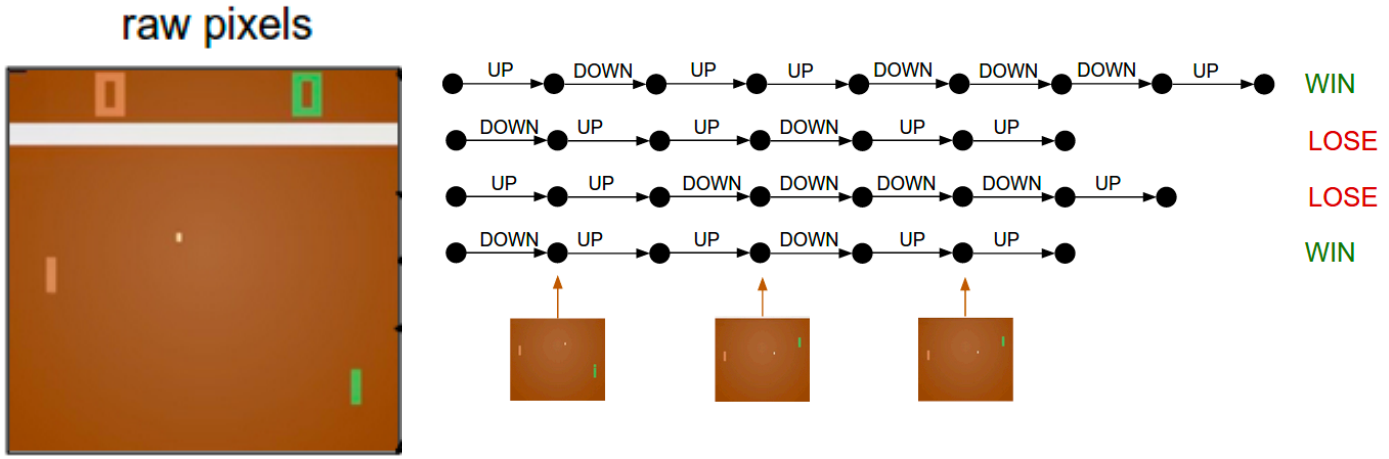
+/- reward for each profit or loss in $

- Play Atari games

+/- reward for increasing or decreasing scores

Sequential Decision Making

- Objective of the agent: select a series of actions to maximize total future rewards
- Actions may have long term consequences
- Reward may be delayed
- Trade-off between immediate reward and long-term reward



Sequential Decision Making

- The history is the sequence of observations, actions, rewards.

$$H_t = O_1, R_1, A_1, ..., A_{t-1}, O_t, R_t$$

- What happens next depends on the history
- State is the function used to determine what happens next

$$S_t = f(H_t)$$

Sequential Decision Making

- Environment state and agent state

$$S_t^e = f^e(H_t) \qquad\qquad S_t^a = f^a(H_t)$$

- **Full observability**: agent directly observes the environment state, formally as Markov decision process (MDP)

$$O_t = S_t^e = S_t^a$$

Sequential Decision Making

- Environment state and agent state

$$S_t^e = f^e(H_t) \qquad\qquad S_t^a = f^a(H_t)$$

- **Full observability**: agent directly observes the environment state, formally as Markov decision process (MDP)

$$O_t = S_t^e = S_t^a$$

- **Partial observability**: agent indirectly observes the environment, formally as partially observable Markov decision process (POMDP)
  - Black jack (only see public cards), Atari game with pixel observation,

Agent must construct its own state representation, as the beliefs of the environment state:

$$S_t^a = (P(S_t^e = s_1), ..., P(s_t^e = s_n))$$

$$S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$$

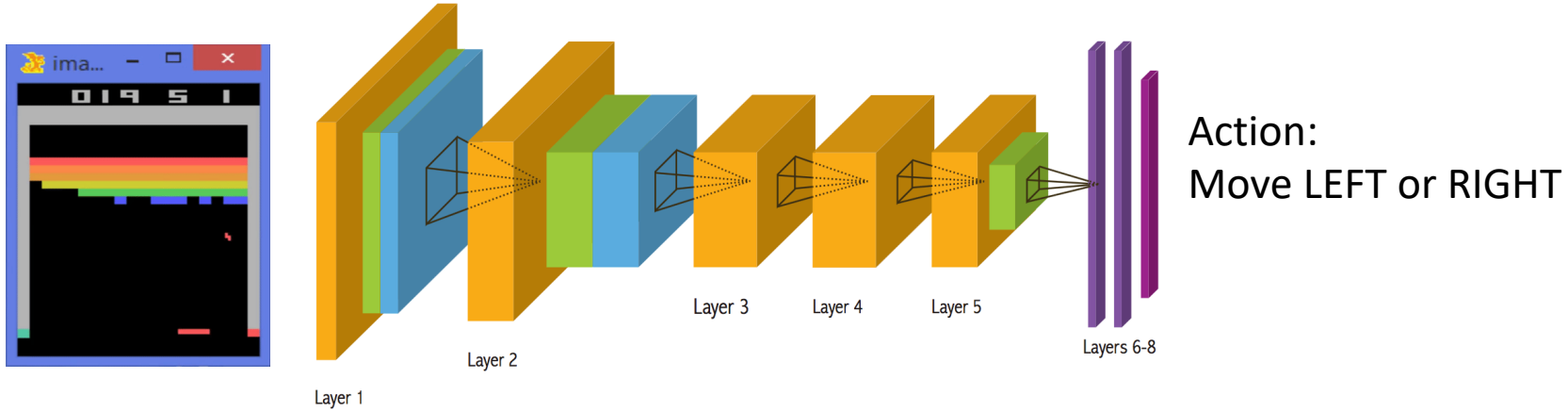
Major Components of an RL Agent

An RL agent may include one or more of these components:

- Policy: agent's behavior function
- Value function: how good is each state or action
- Model: agent's state representation of the environment

Policy

- A policy is the agent's behavior model
- It is a map function from state/observation to action.
- Stochastic policy: Probabilistic sample $\pi(a|s) = P[A_t = a|S_t = s]$
- Deterministic policy: $a^* = \arg\max_a \pi(a|s)$



Value function

- Value function: expected discounted sum of future rewards under a particular policy π
- Discount factor weights immediate vs future rewards
- Used to quantify goodness/badness of states and actions

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ for all } s \in \mathcal{S},$$

- Q-function (could be used to select among actions)

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right].$$

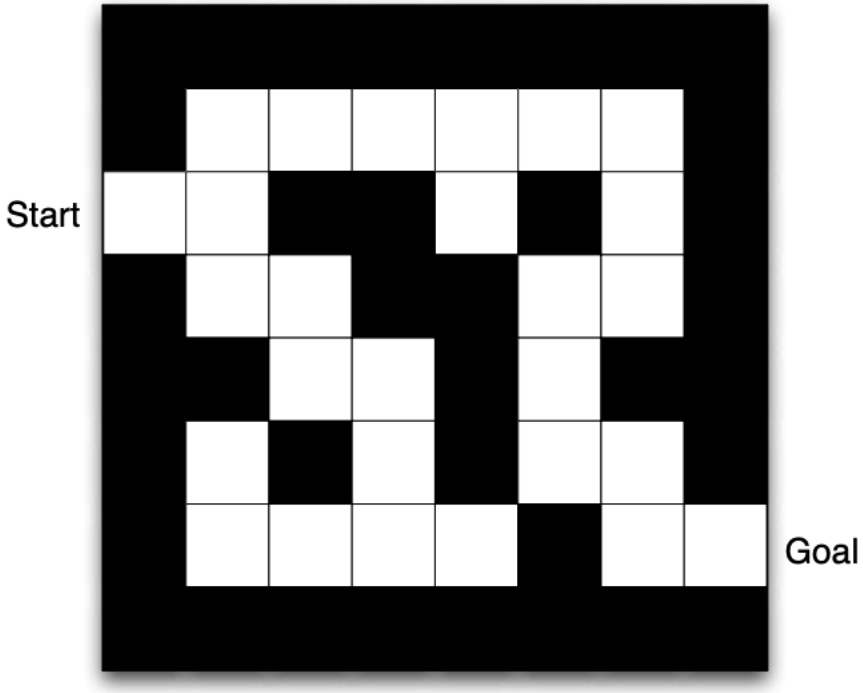
Model

A model predicts what the environment will do next

Predict the next state: $\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$

Predict the next reward: $\mathcal{R}^a_s = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$

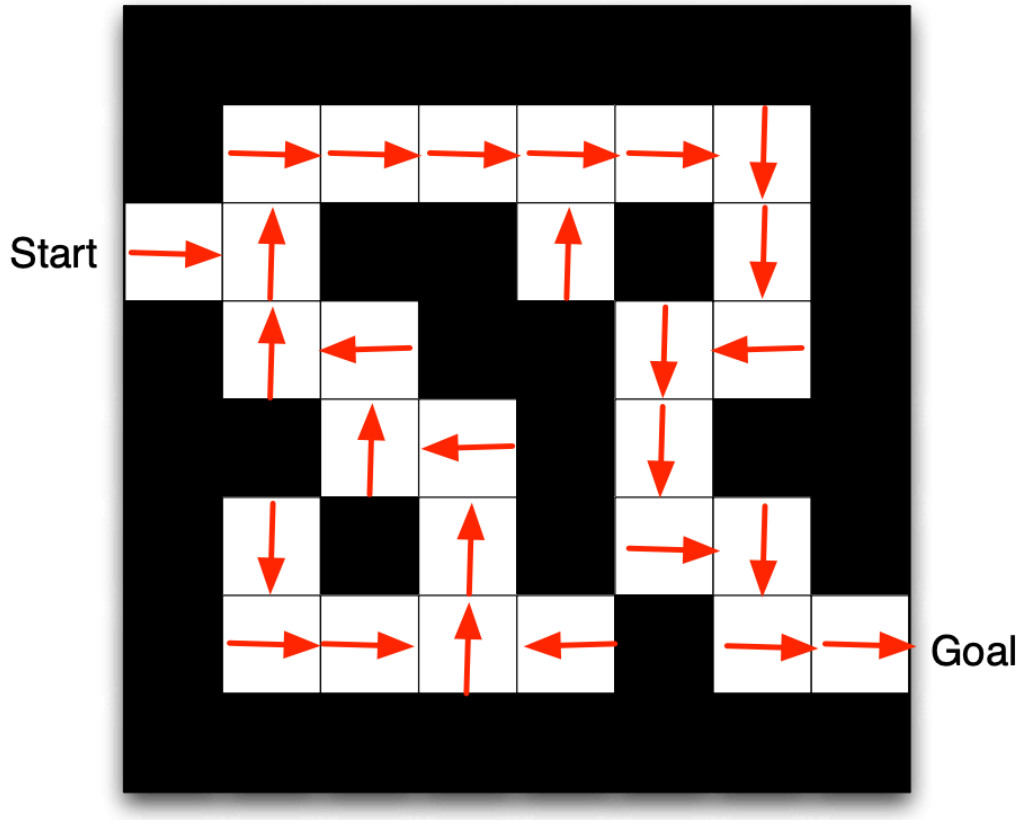
Maze Example



- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

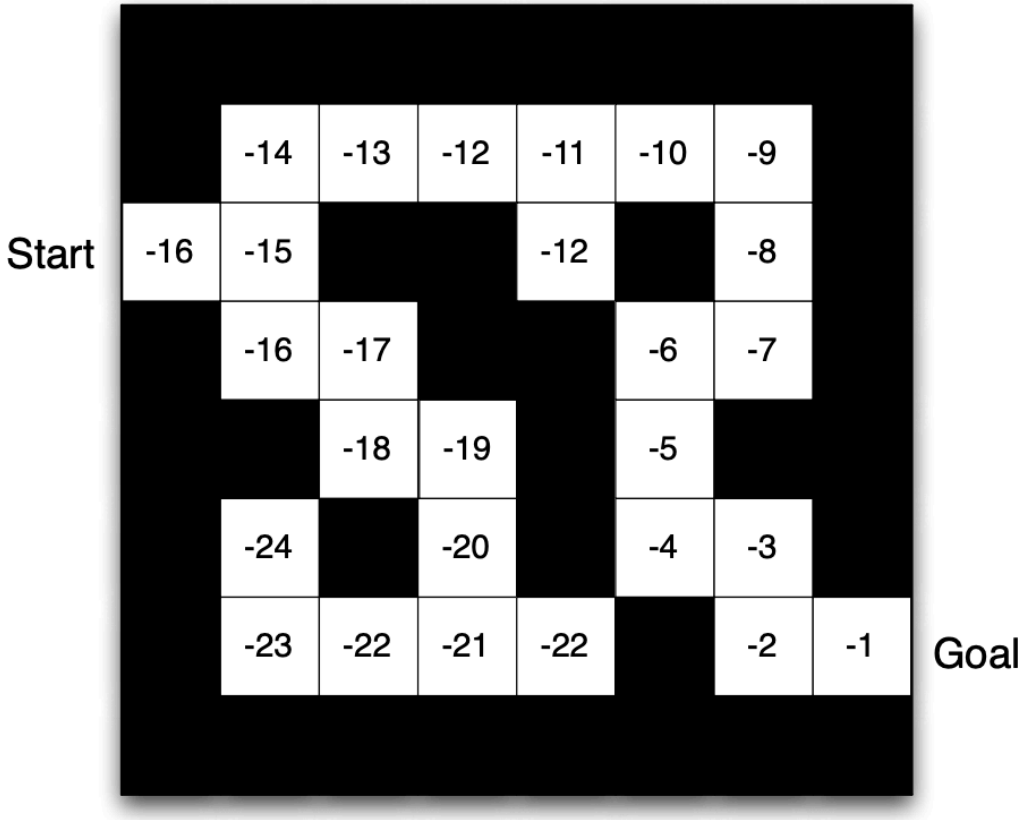
From David Silver Slide

Maze Example: Policy-based



- Arrows represent policy $\pi(s)$ for each state s

Maze Example: Value function-based



- Numbers represent value $v_\pi(s)$ of each state s

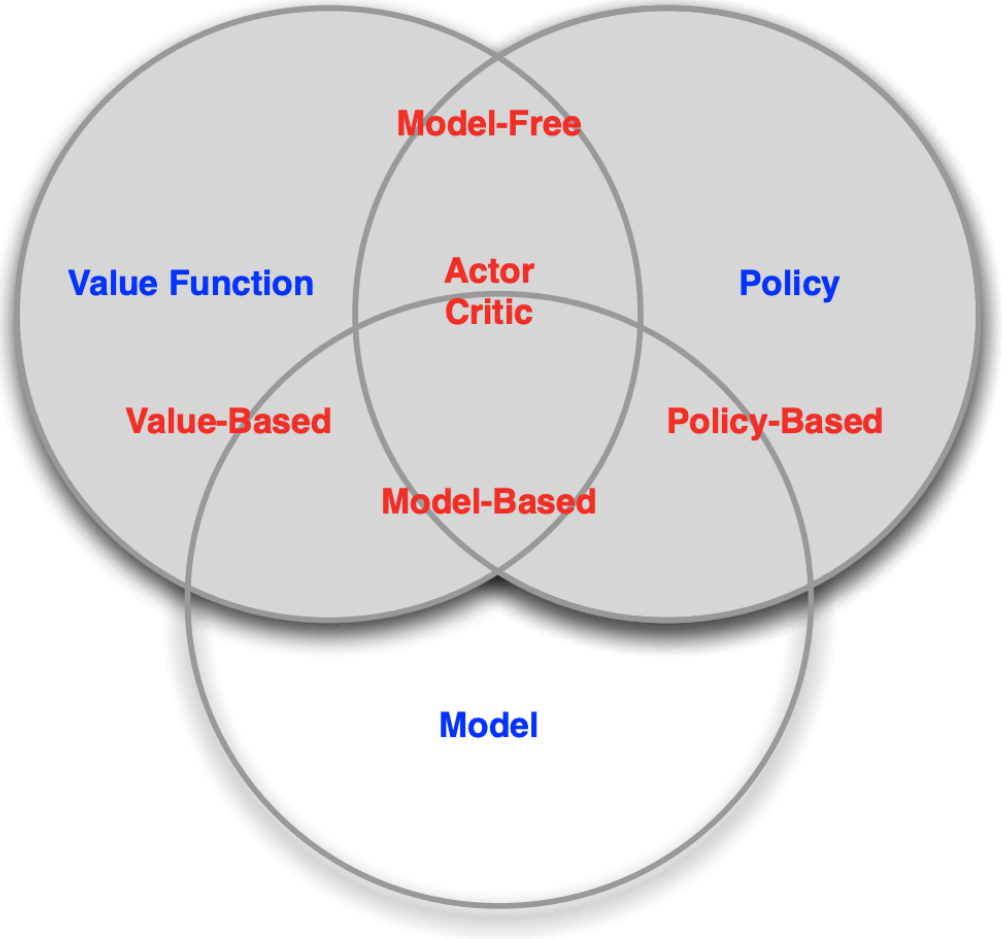
Types of RL Agents based on What the Agent Learns

- Value-based
  - Explicit: Value function
  - Implicit: Policy (can derive a policy from value function)
- Policy-based
  - Explicit: policy
  - No value function
- Actor-Critic:
  - Explicit: policy and value function

Types of RL Agents on if there is model

- Model-based
  - Explicit: model
  - May or may not have policy and/or value function
- Model-free
  - Explicit: value function and/or policy function
  - No model.

Types of RL Agents



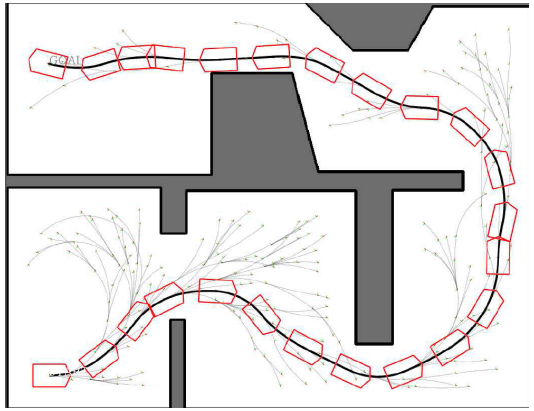
Credit: David Silver's slide

Two Fundamental Problems in Sequential Decision Making

- Planning
  - Given model about how the environment works.
  - Compute how to act to maximize expected reward without external interaction.
- Reinforcement learning
  - Agent doesn’t know how world works
  - Interacts with world to implicitly learn how world works
  - Agent improves policy (also involves planning)

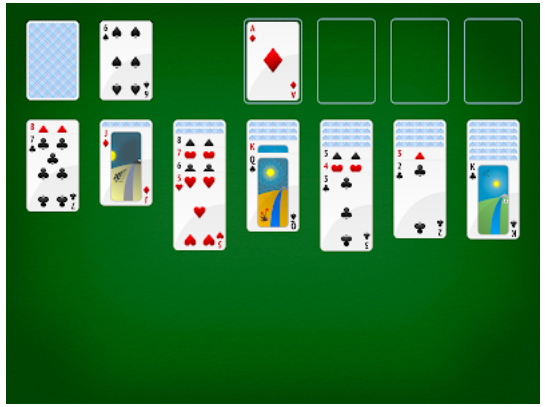
Planning

Path planning



Map is known
All the rules of the vehicle are known
Planning algorithms: dynamic programming, A* search, tree search, …

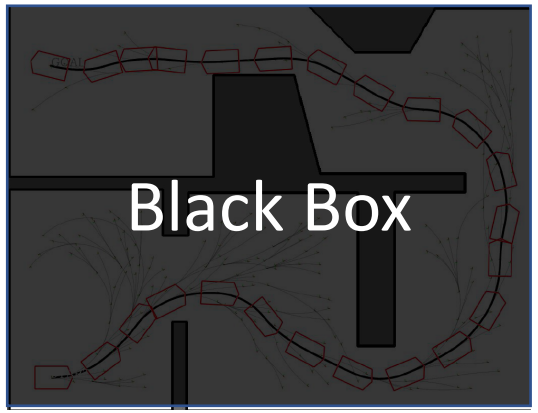
Patience (单人纸牌游戏)



Rules of the game are known.
Planning algorithms: dynamic programming, tree search

Reinforcement Learning

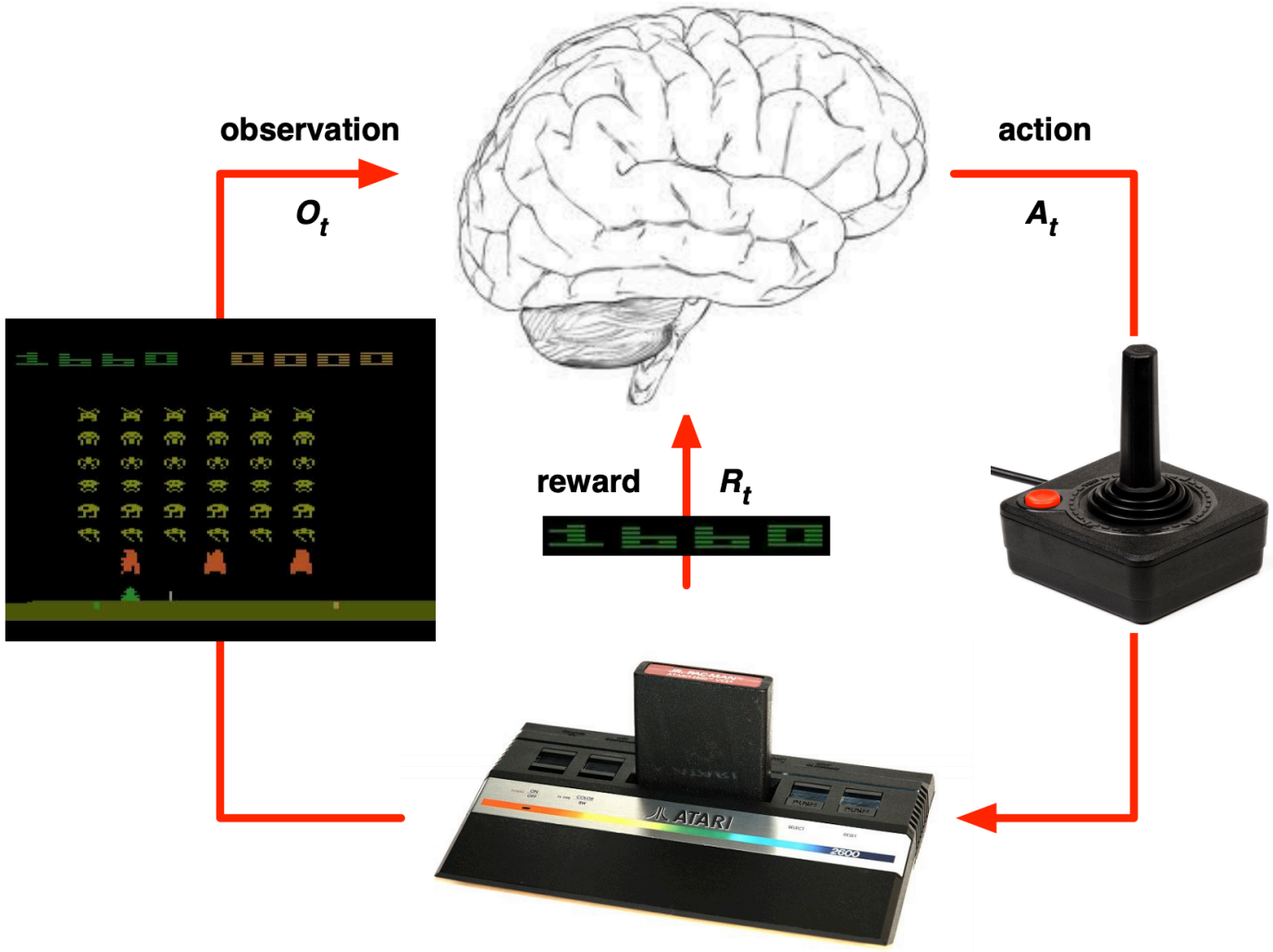
Path planning



Patience (单人纸牌游戏)

- No rules or knowledge about the environment.
- Learn directly by taking actions and seeing what happens.
- Try to find a good policy over time that yields high reward.
- Planning is needed in inference or forward pass.

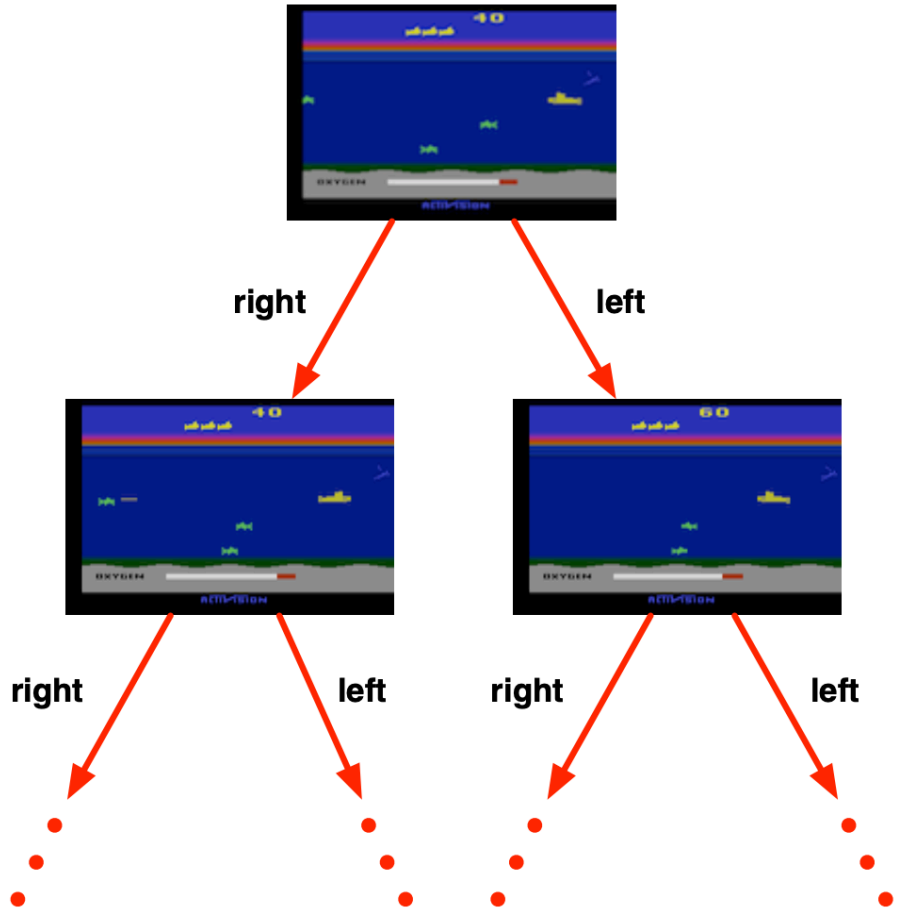
Atari Example: Reinforcement Learning



- Rules of the game are unknown
- Learn directly from interactive game-play
- Pick actions on joystick, see pixels and scores

Atari Example: Planning

- Rules of the game are known
- Can query emulator
  - perfect model inside agent's brain
- If I take action *a* from state *s*:
  - what would the next state be?
  - what would the score be?
- Plan ahead to find optimal policy
  - e.g. tree search



Exploration and Exploitation

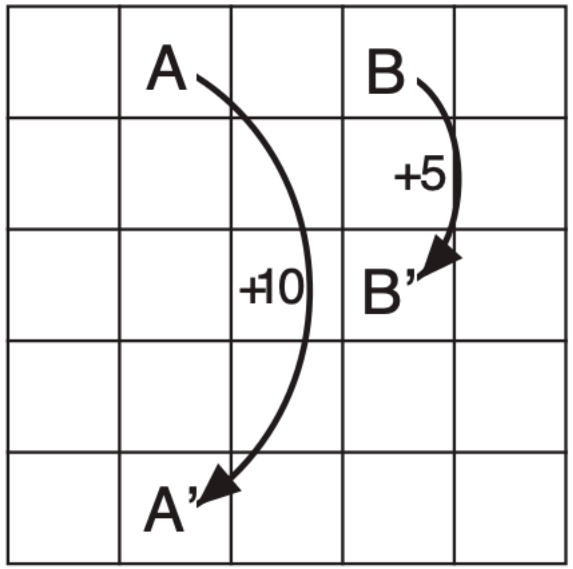
- Agent only experiences what happens for the actions it tries!
- How should an RL agent balance its actions?
  - Exploration: trying new things that might enable the agent to make better decisions in the future
  - Exploitation: choosing actions that are expected to yield good reward given past experience
- Often there may be an exploration-exploitation tradeoff
  - May have to sacrifice reward in order to explore & learn about potentially better policy

Exploration and Exploitation

- Restaurant Selection
  - Exploitation: Go to your favourite restaurant
  - Exploration: Try a new restaurant
- Online Banner Advertisements
  - Exploitation: Show the most successful advert
  - Exploration: Show a different advert
- Oil Drilling
  - Exploitation: Drill at the best known location
  - Exploration: Drill at a new location
- Game Playing
  - Exploitation: Play the move you believe is
  - Exploration: play an experimental move

One exercise (Gridworld example)

- Sutton & Barto: Example 3.5, Exercise 3.14 – Exercise 3.16, Example 3.8.



Summary

- Course logistics
- Overview of reinforcement learning
- Introduction to sequential decision making

Next

Lecture 2: Get hand dirty on coding RL