Intro to Reinforcement Learning
强化学习纲要
Lecture 1: Overview

Bolei Zhou

https://github.com/zhoubolei/introRL

Outline

- Course overview
- Introduction to reinforcement learning
- Introduction to sequential decision making
- Experimenting with RL by coding

Course Overview

- Course objective: Democratize RL
- Host: Bolei Zhou
  - http://bzhou.ie.cuhk.edu.hk/
  - 知乎：周博磊（https://www.zhihu.com/people/zhou-bo-lei）
- Slides and resources will be uploaded to https://github.com/zhoubolei/introRL

Course Structure: Foundation

- Lecture 1: Overview
- Lecture 2: Markov Decision Process
- Lecture 3: Model-free Prediction and Control
- Lecture 4: On-policy and Off-policy Learning
- Lecture 5: Value Function Approximation
- Lecture 6: Policy Optimization: Foundation
- Lecture 7: Policy Optimization: State of the Art
- Lecture 8: Model-based RL

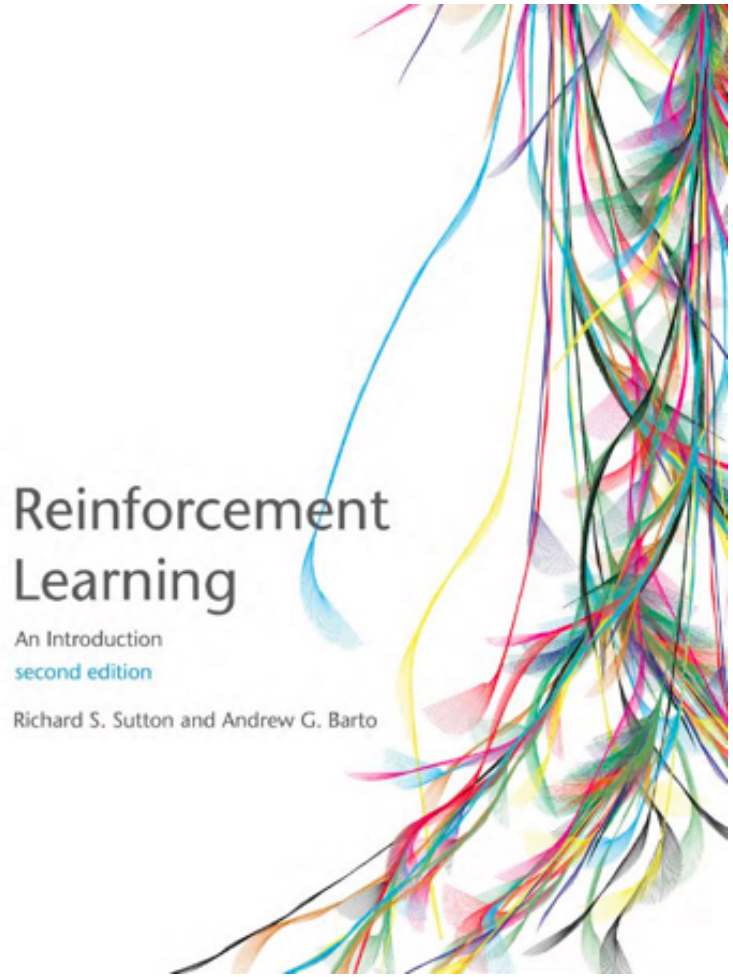
Value-based

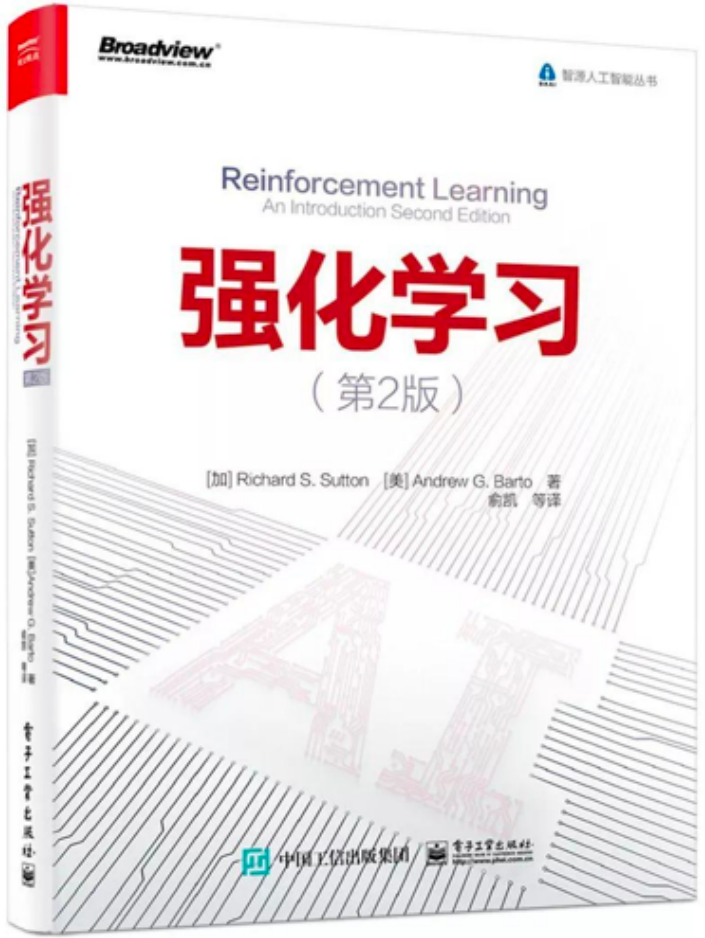
Policy-based

Course Structure: Advanced Topics

- Lecture 9: Case Study on AlphaGo series
- Lecture 10: Case Study on AlphaStar and OpenAI Five
- Lecture 11: Distributed computing and RL system design
- Lecture 12: Generative Modeling
- So on…

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

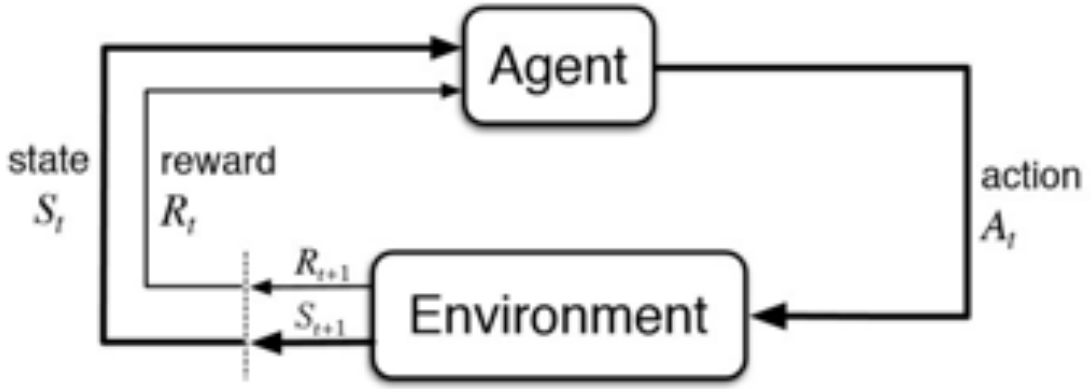




Prerequisites

- You should have some background in Linear Algebra course and Probability course, and one machine learning relevant course (data mining, pattern recognition, deep learning, etc).
- Programming experience: Python, PyTorch

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

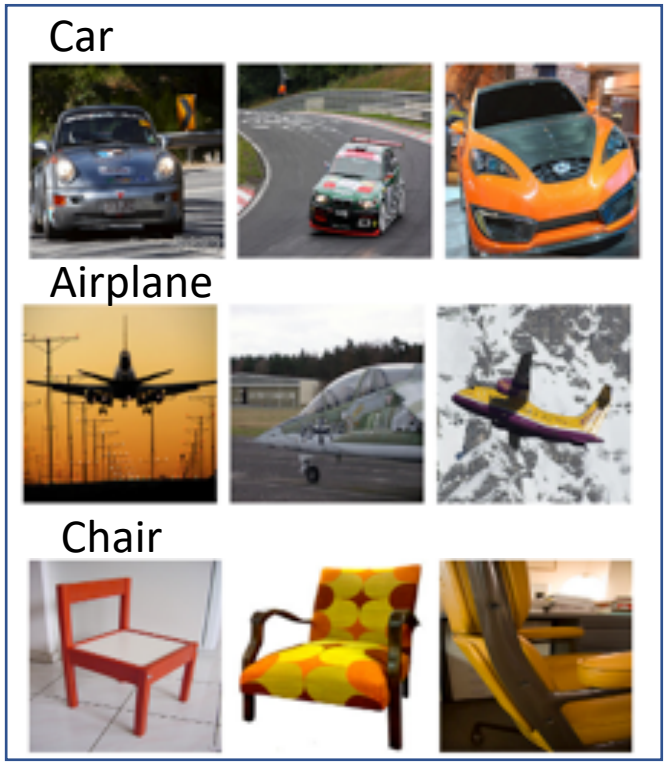
一个Agent怎么在一个不确定的环境下去最大化Reward

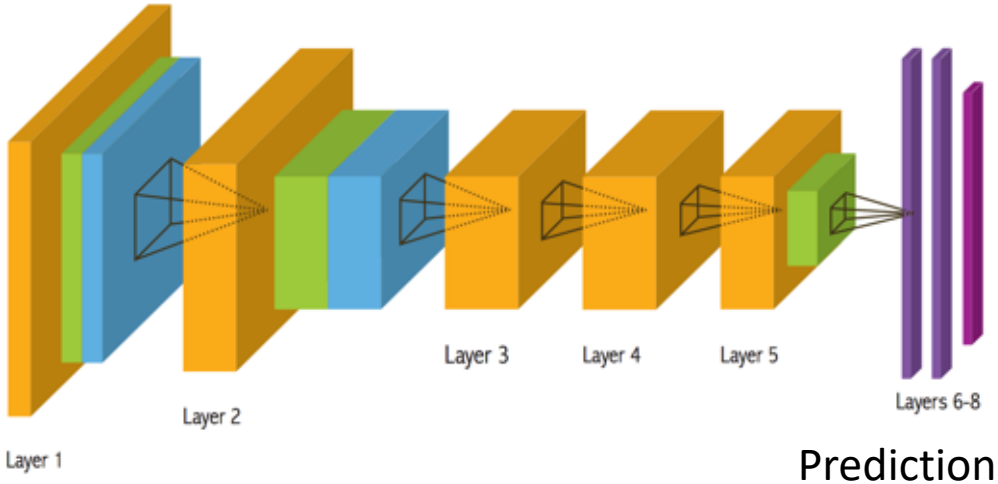
Supervised Learning: Image Classification

标注的数据是没有关联的　　独立同分布（iid，independently identically distribution）

- Annotated images, data follows i.i.d distribution.
- Learners are told what the labels are.　告诉学习器正确的标签是什么

Training annotated data





Prediction: Airplane

Prediction is wrong
Correct label: Car

Back-propagation

Reinforcement Learning: Playing Breakout

数据不是独立同分布的，而是一组时序数据

- Data are not i.i.d. Instead, a correlated time series data
- No instant feedback or label for correct action 做出Aciton后不能立即得到反馈

强化学习比带监督的深度学习困难

Action: Move LEFT or Right

Atari Game: Breakout

Reinforcement Learning: Playing Breakout

Training data 训练数据是“玩游戏的过程”（序列）

Step1

Step2

Step3

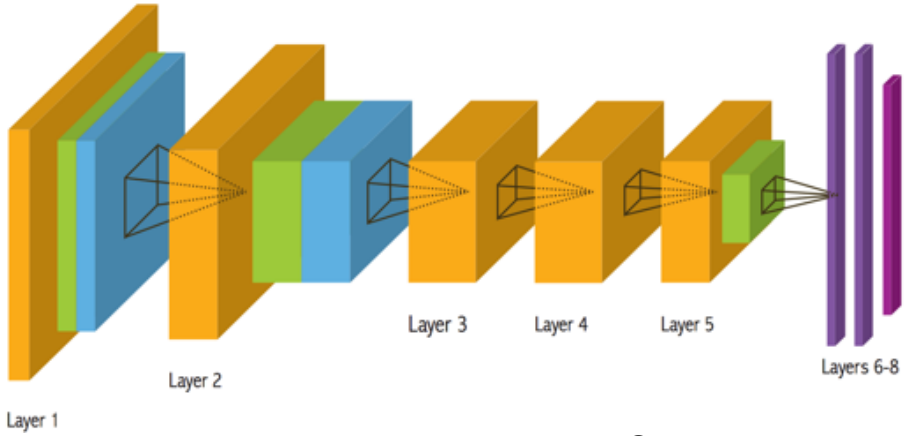
Step4

Step5

. . .

Human playing?

Step3



Action:
Move LEFT

因为没有标签，无法立即得知Action是正确还是错误。
所以这里会有奖励延迟（Delayed Reward）问题，只有在游戏结束之后才能知晓奖励。

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

Backpropagation?

强化学习 V.S. 监督学习

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)

  “不断试错”

  尝试新行为

  重复执行已经知道的策略
- There is no supervisor, only a reward signal, which is also delayed

  奖励信号是延迟知晓的

强化学习的特征

Features of Reinforcement Learning

- Trial-and-error exploration 需要在环境中不断探索和试错
- Delayed reward 获得的是延迟的奖励
- Time matters (sequential data, non i.i.d data) 时序起作用
- Agent's actions affect the subsequent data it receives (agent's action changes the environment)

  Agent当前的动作会影响到它随后获得的数据

Big deal: Able to Achieve Superhuman Performance

监督学习的数据标定是人为的，因此监督学习算法的效果是无法超过人为标定的效果的

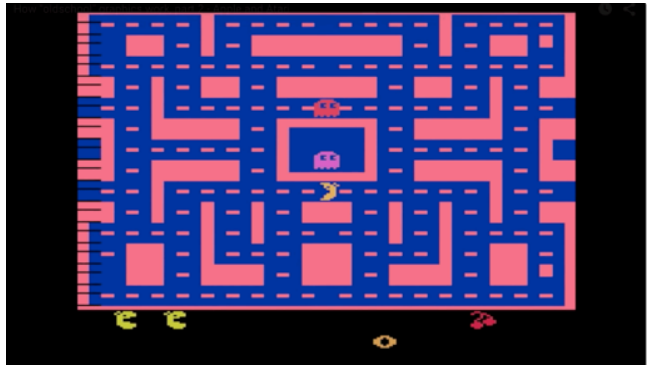
- Upper bound for supervised learning is human-performance.
- Upper bound for reinforcement learning?

而强化学习是有超越人的潜力的。

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

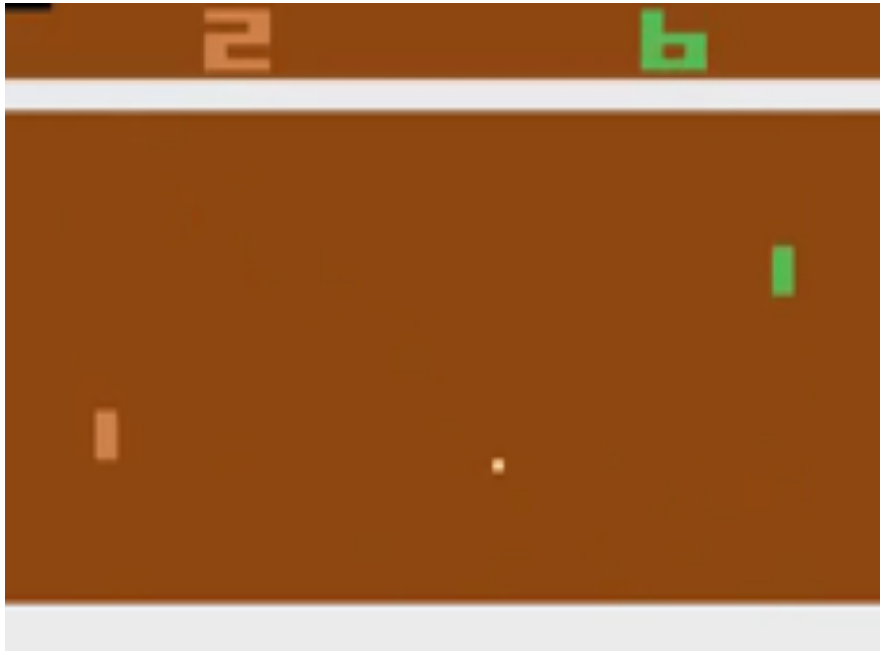
Examples of reinforcement learning

- 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 runs 36 kilometers per hour.
- Portfolio management.
- Playing Atari game



RL example: Pong

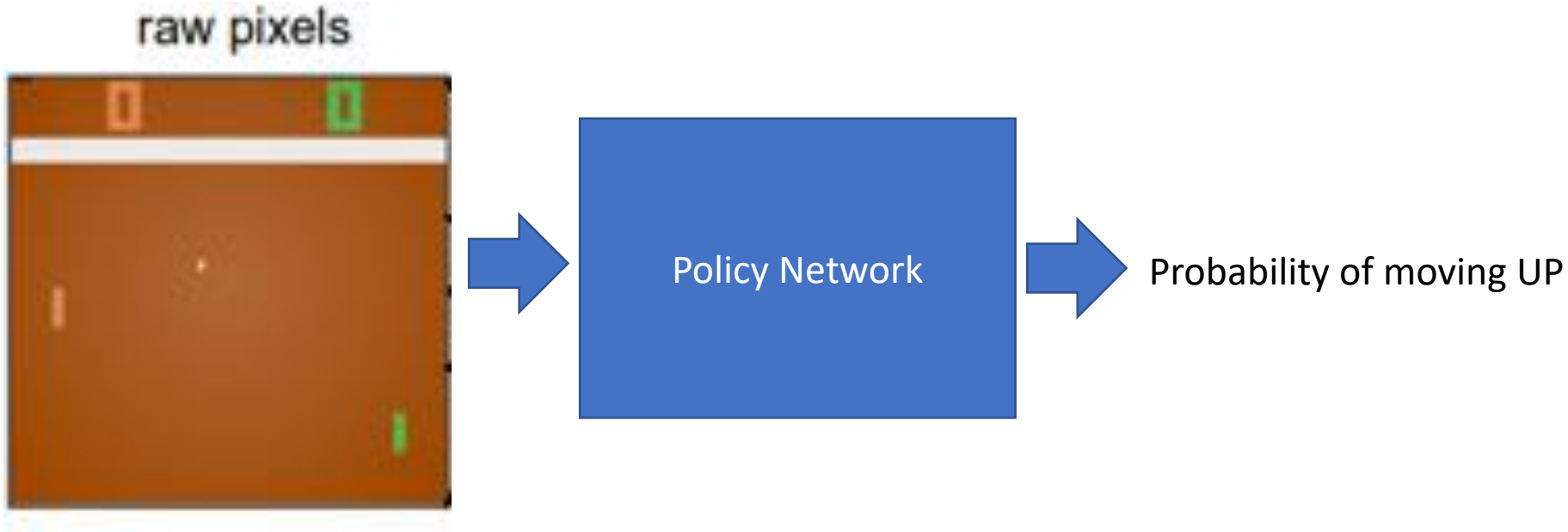
Action: move UP or DOWN



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

RL example: Pong

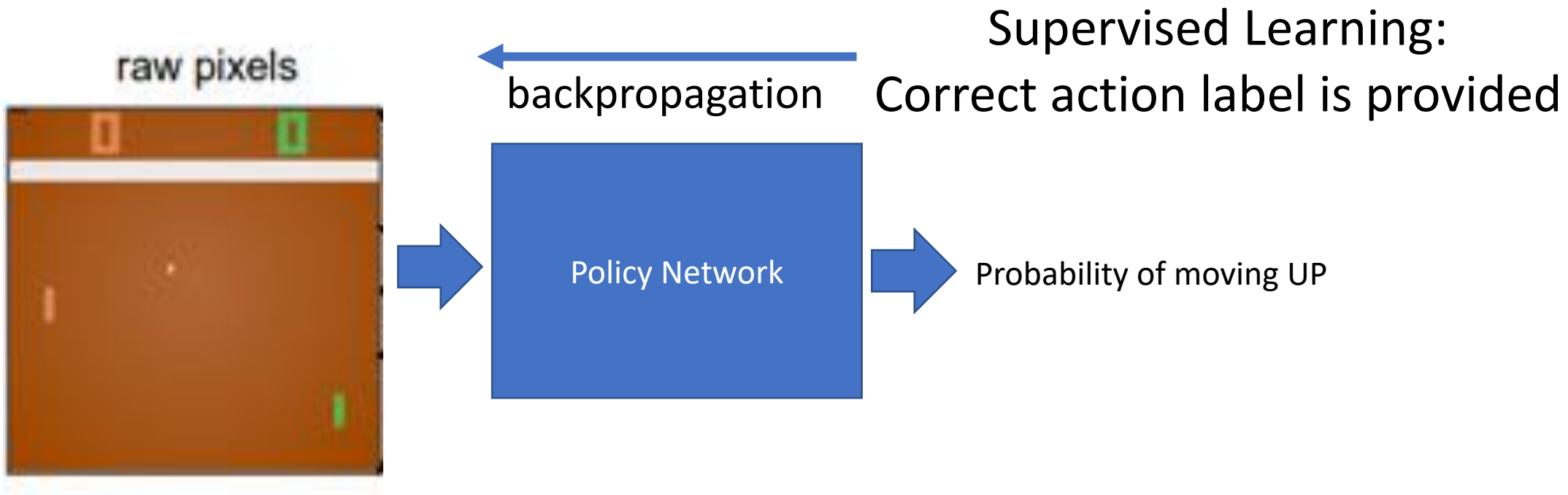
Action: move UP or DOWN



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

RL example: Pong

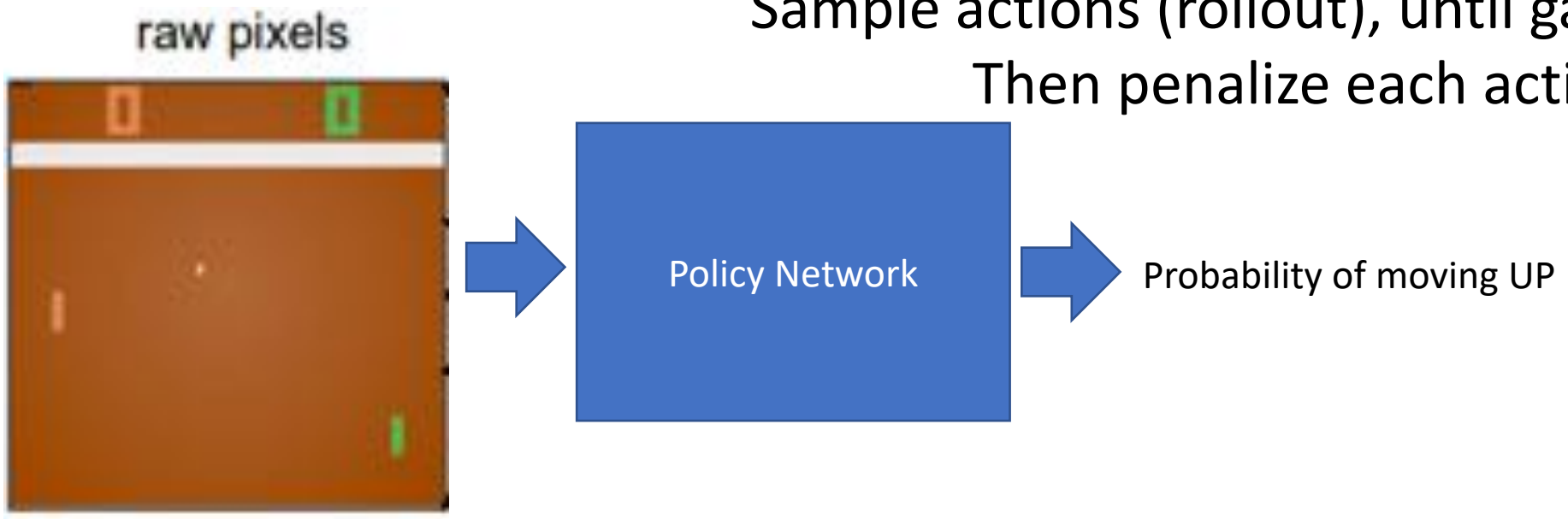
- Action: move UP or DOWN



RL example: Pong

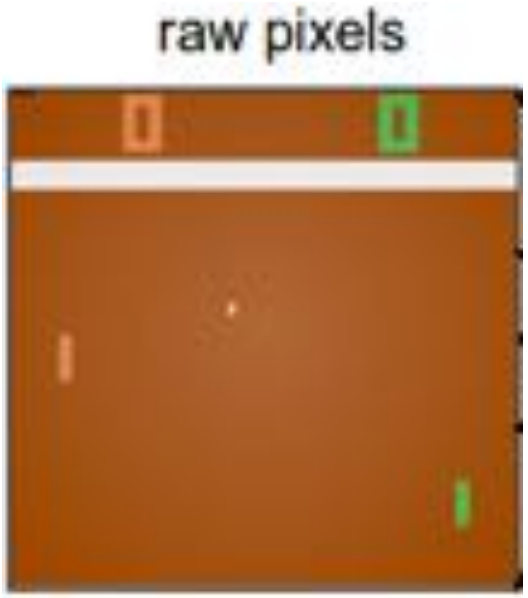
- Action: move UP or DOWN

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



RL example: Pong

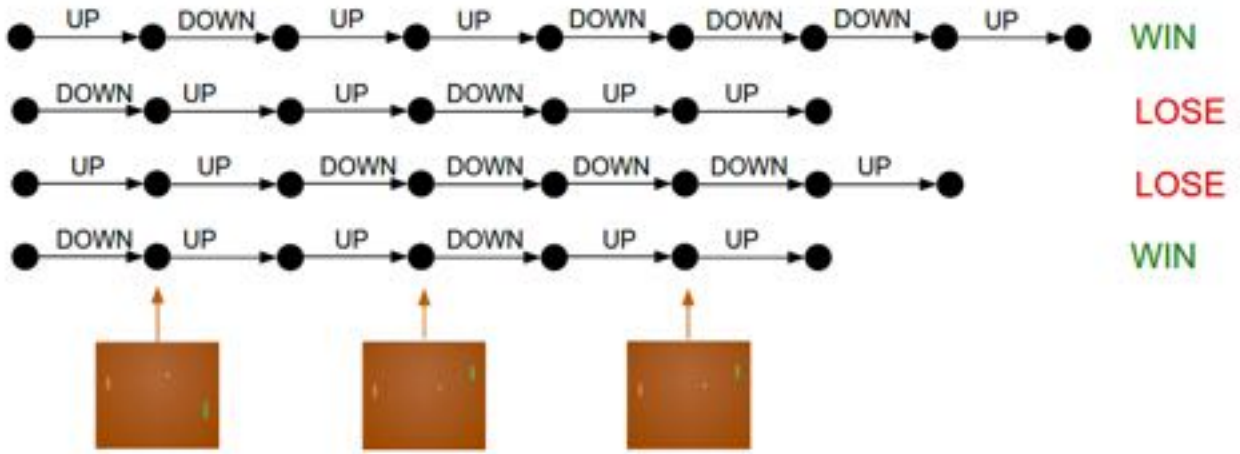
- Action: move UP or DOWN



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

Possible rollout sequence:

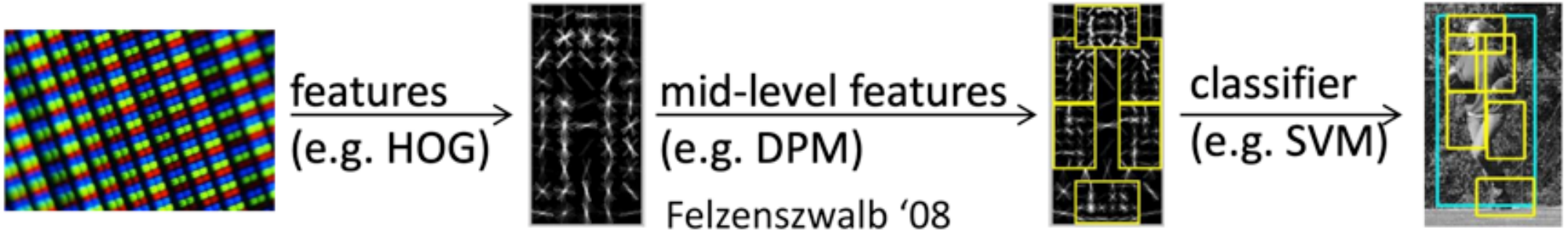
Eventual Reward:

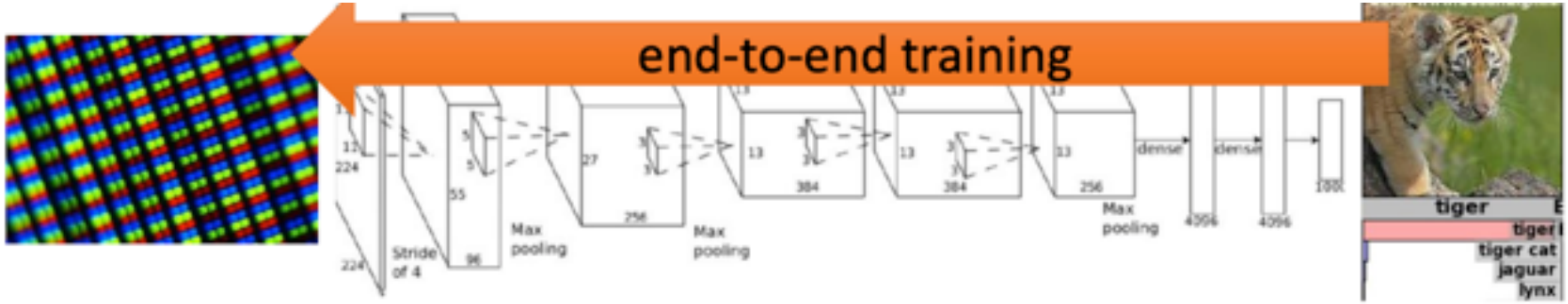


Rollout：从当前帧去生成很多局的游戏，得到采取不同动作的“轨迹”

Deep Reinforcement Learning: Deep Learning + Reinforcement Learning

- Analogy to traditional CV and deep CV



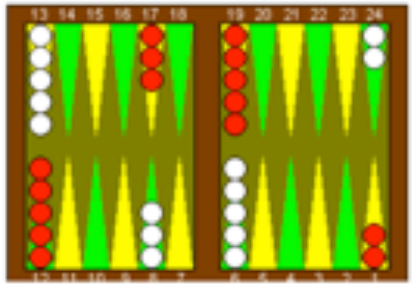


卷积神经网络实现了特征提取和分类的“合并”，实现了end–to–end

Deep Reinforcement Learning:
Deep Learning + Reinforcement Learning

- Standard RL and deep RL

TD-Gammon, 1995

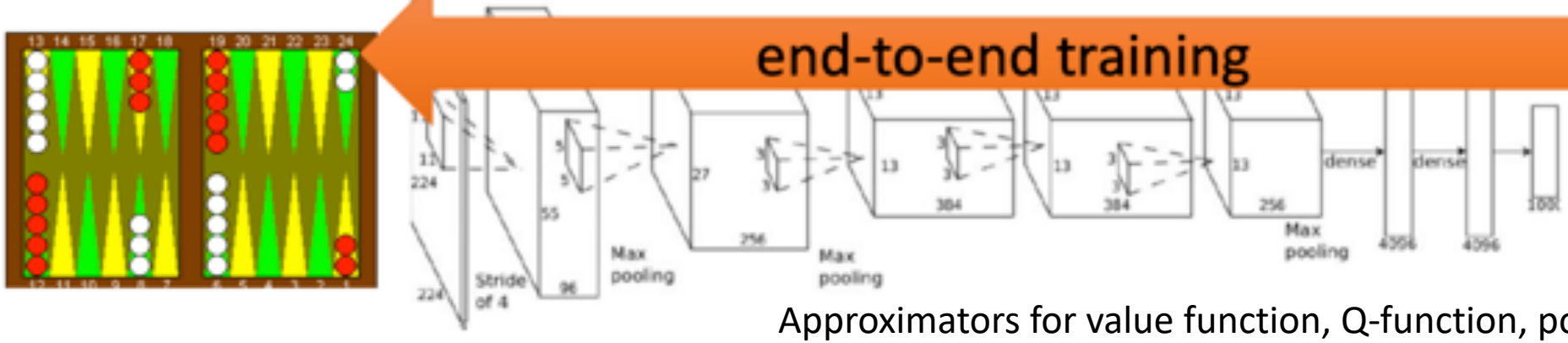


game of backgammon

•Standard RL

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

Deep RL:不需要手工去设计特征，用神经网络来拟合Action Value Func或者Policy

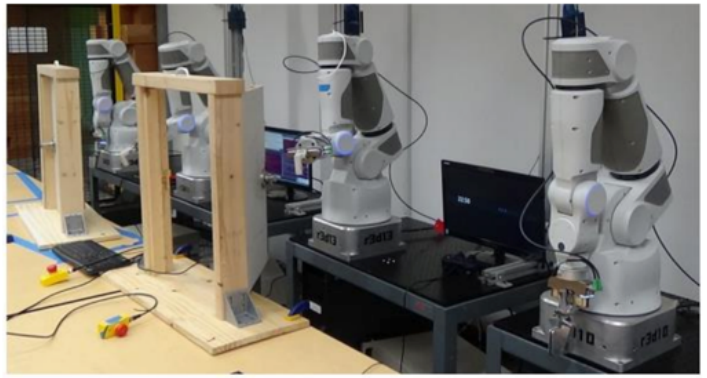


Approximators for value function, Q-function, policy networks

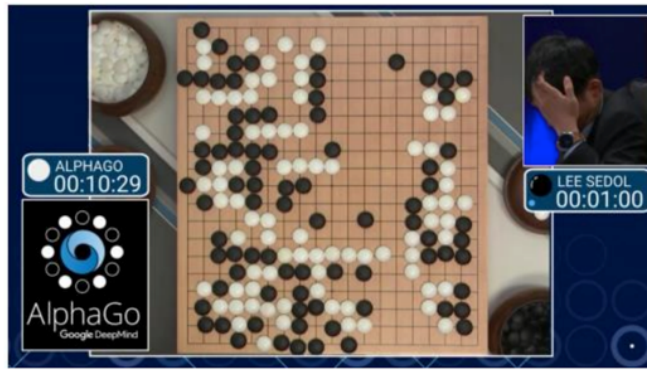
Why RL works now?

- 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



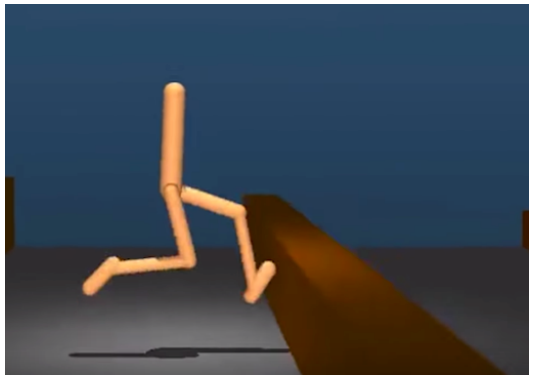
Robotics



Beating best human player

More Examples on RL

Learning to walk

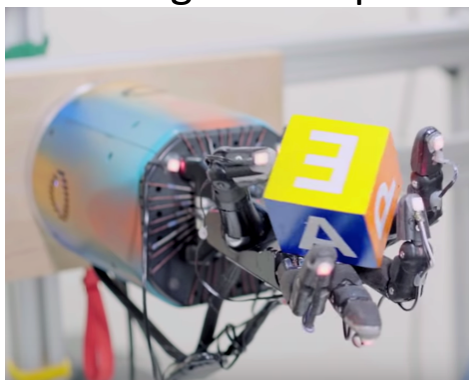


https://www.youtube.com/watch?v=gn4nRCC9TwQ

Learning to grasp

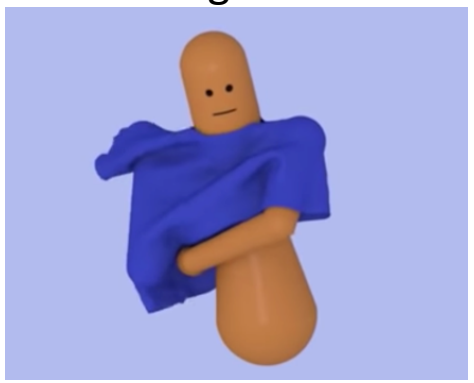
https://ai.googleblog.com/2016/03/deep-learning-for-robots-learning-from.html

Learning to manipulate



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

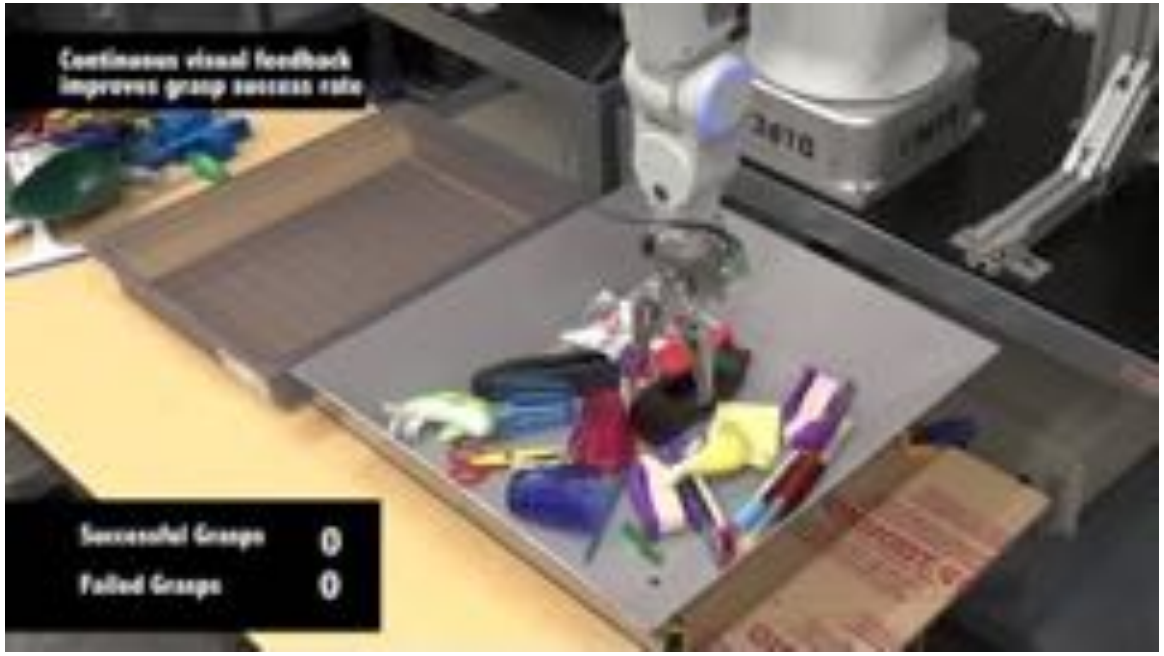
Learning to dress



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

Learning to walk

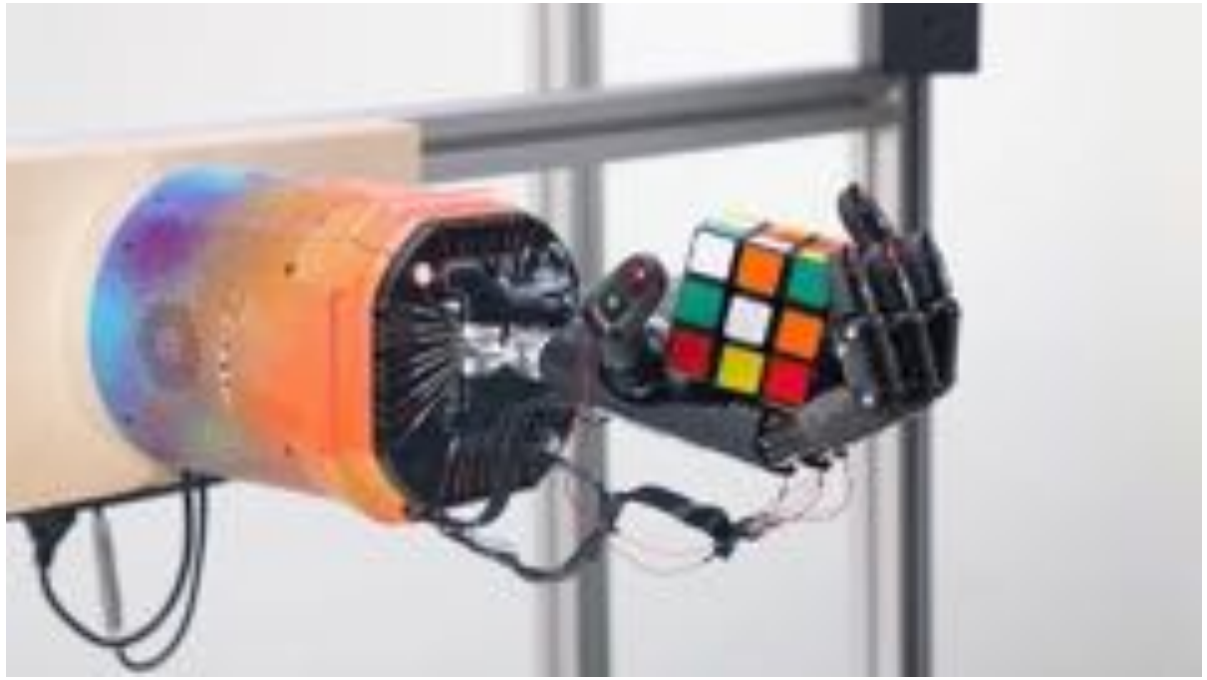
Learning to grasp



Learning to move fingers

2018

2019



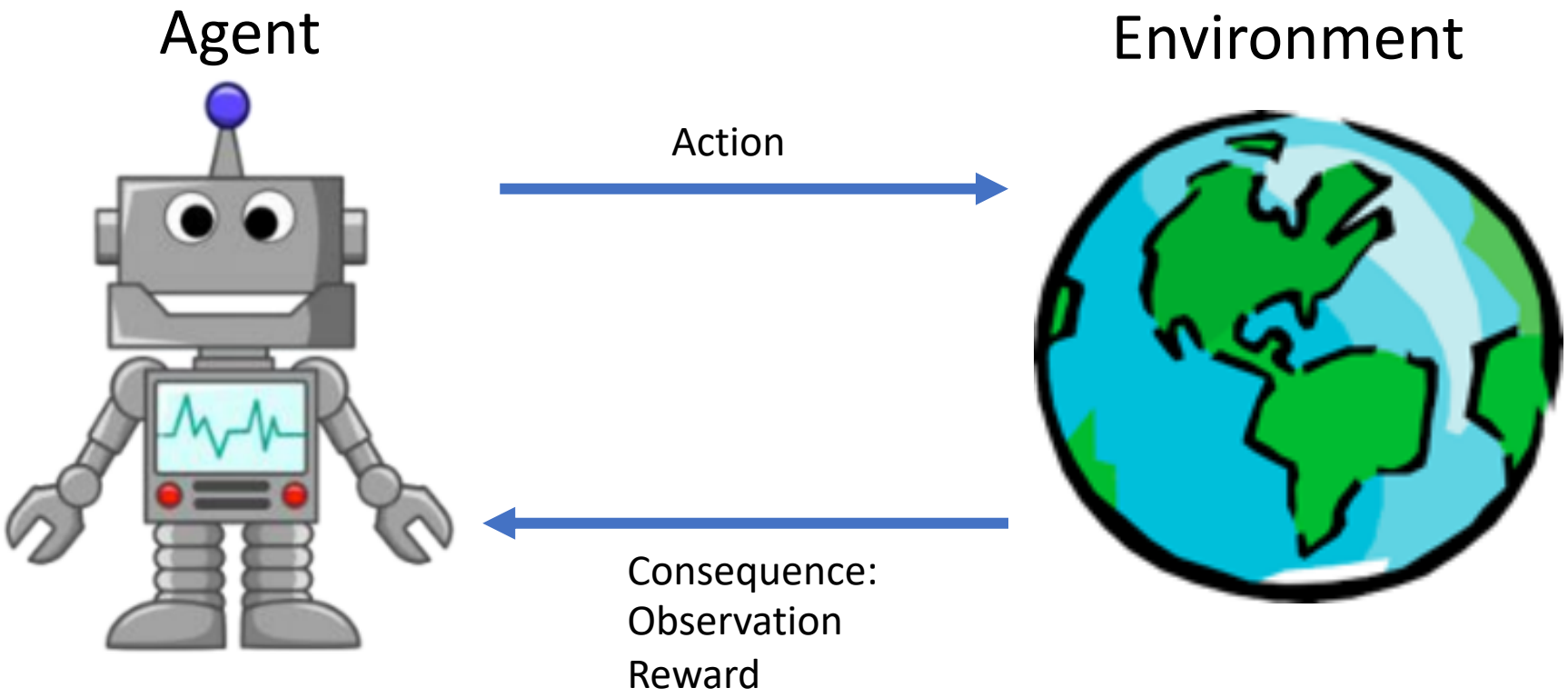
OpenAI

Learning to dress

Introduction to Sequential Decision Making

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

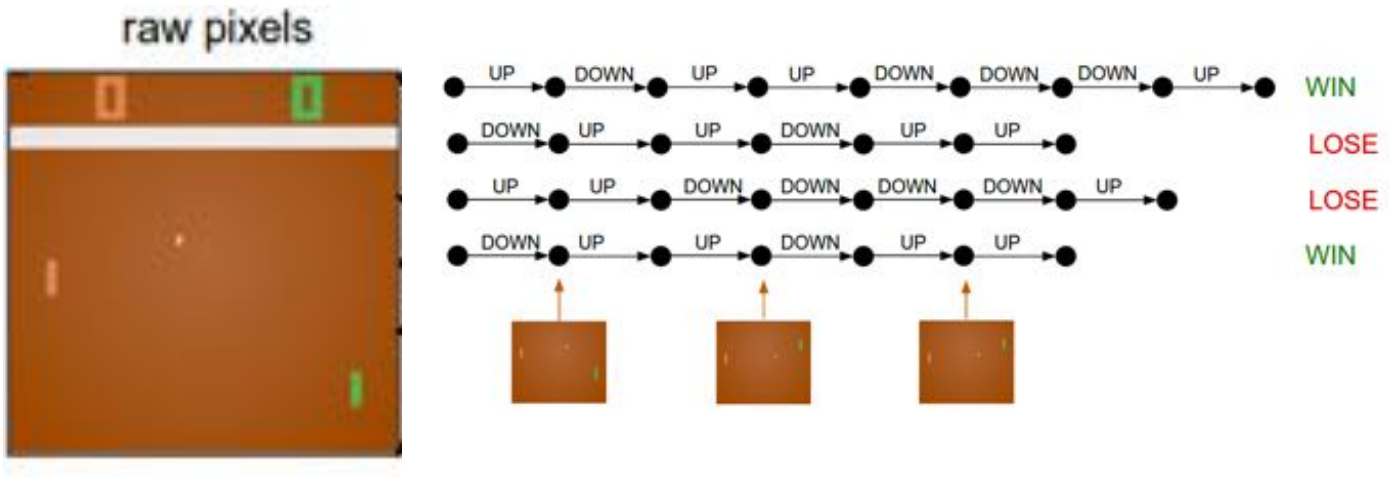
奖励的稀疏程度也会影响游戏的难度

强化学习过程中，Agent做的事情就是做出一系列的“Sequential Decision”

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

怎么让Agent取得更多的长期奖励?



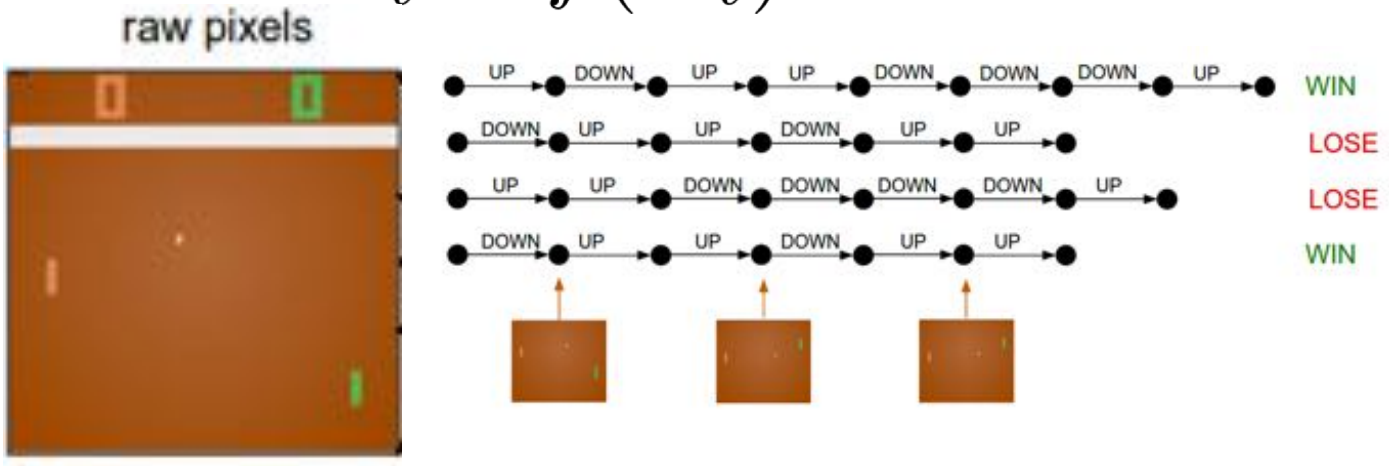
Sequential Decision Making

- 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

环境状态 $S_t^e = f^e(H_t)$ $S_t^a = f^a(H_t)$ Agent的状态

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

$$O_t = S_t^e = S_t^a$$ 环境状态等价于Agent状态时，称为是full observability

Sequential Decision Making

- Environment state and agent state

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

- **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,

Major Components of an RL Agent

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

- Policy: agent's behavior function 决策函数，被Agent用来决定下一步的Action
- Value function: how good is each state or action 价值函数，被Agent用来估计当前状态“多有利”
- Model: agent's state representation of the environment

  模型：Agent对环境的“理解”

Policy常用pie表示

Policy

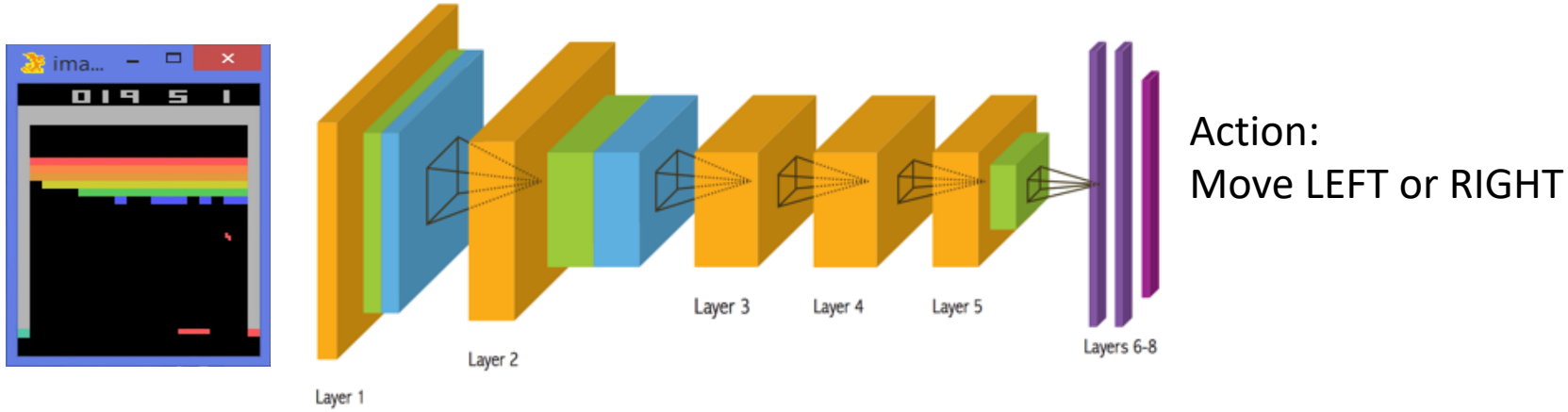
输入：状态
输出：动作

- A policy is the agent's behavior model
- It is a map function from state/observation to action.

Policy本质上是一个从状态/观察值映射到动作的函数

stochastic(随机的)

- Stochastic policy: Probabilistic sample $\pi(a|s) = P[A_t = a|S_t = s]$
- Deterministic policy: $a^* = \arg\max_a \pi(a|s)$



[强化学习论文简要--Policy Gradient - 知乎](https://zhuanlan.zhihu.com/p/43453846)

Value function

在Policy pi中预测未来能得到的奖励的“折扣”加和

- Value function: expected discounted sum of future rewards under a particular policy π

“十天之后给100块钱和现在给50块钱如何折算”

- Discount factor weights immediate vs future rewards
- Used to quantify goodness/badness of states and actions

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

Gt——奖励经过discount之后的值

- 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

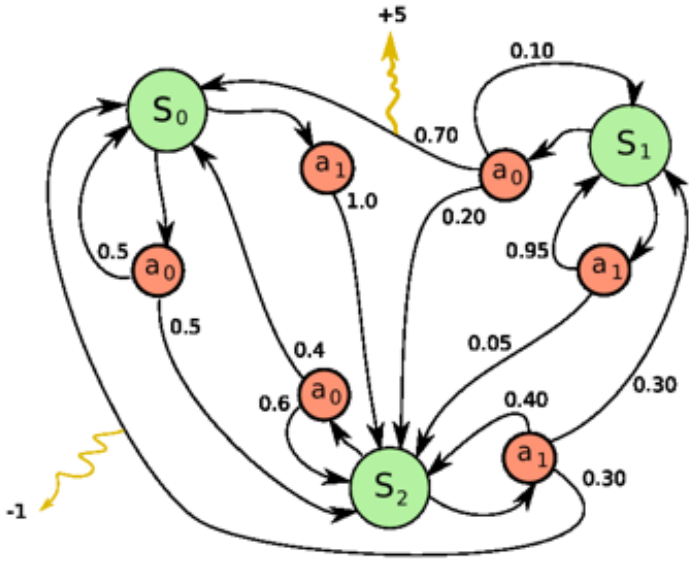
Predict the next state: $\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$

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

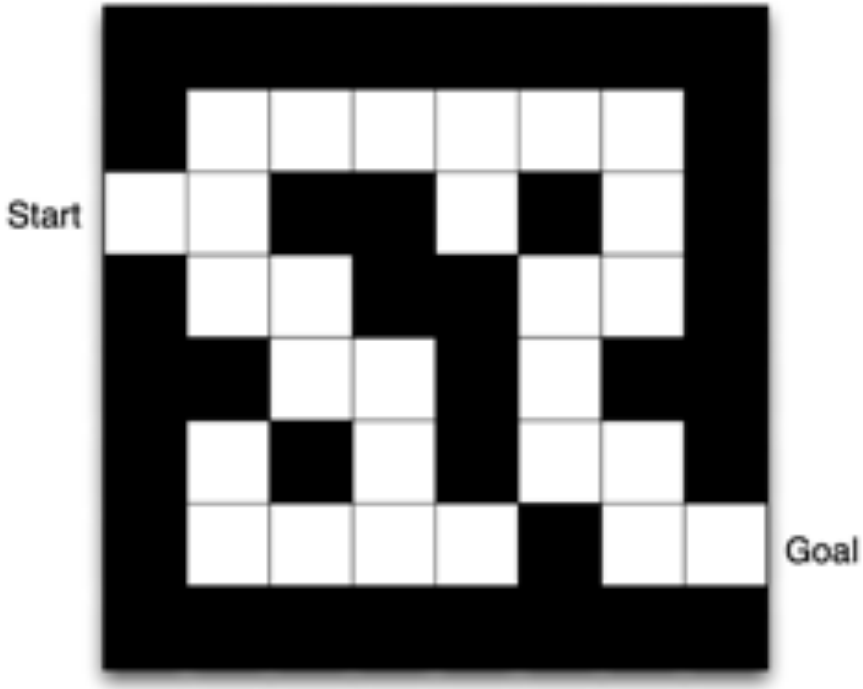
Markov Decision Processes (MDPs)

Definition of MDP

1. P^a is dynamics/transition model for each action
   $P(S_{t+1} = s' | S_t = s, A_t = a)$
2. $\mathcal{R}$ is a reward function $R(S_t = s, A_t = a) = \mathbb{E}[R_t | S_t = s, A_t = a]$
3. Discount factor $\gamma \in [0, 1]$



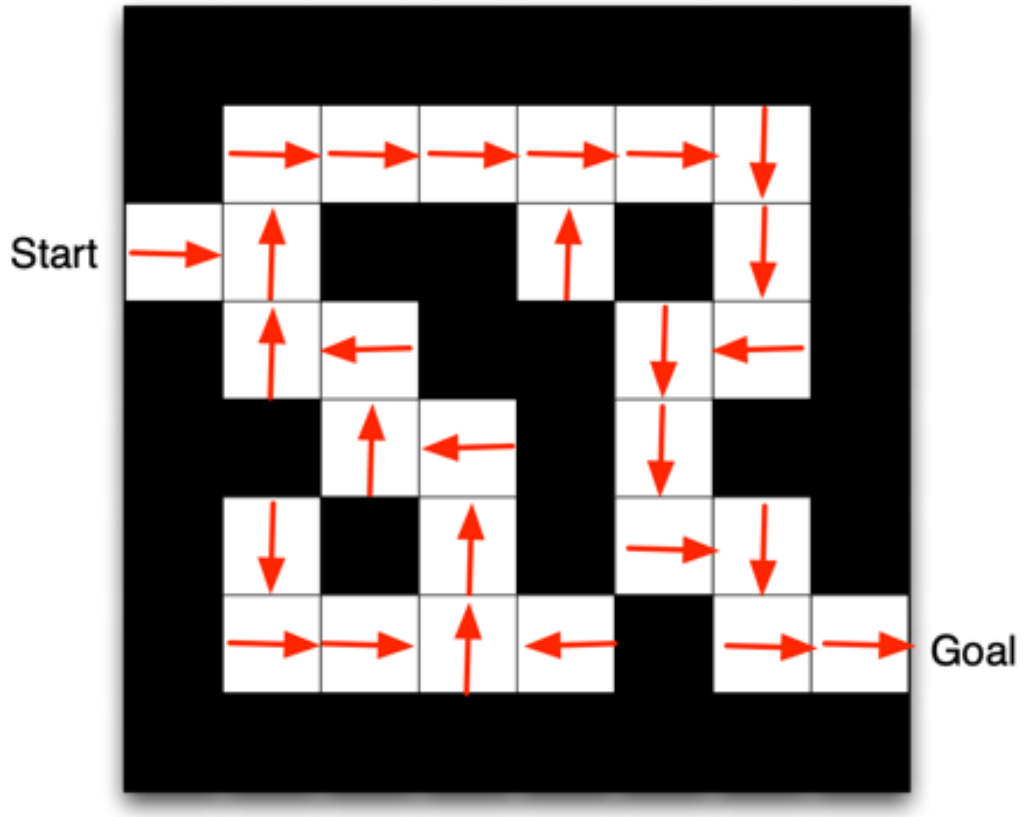
Maze Example



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

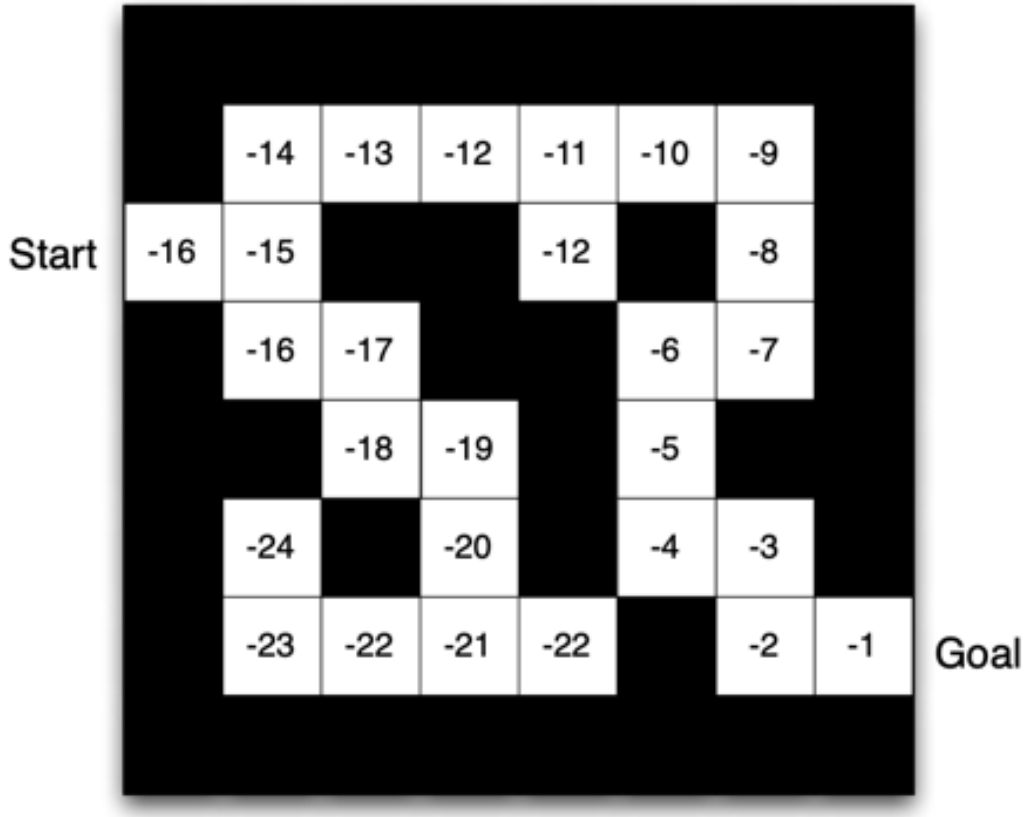
From David Silver Slide

Maze Example: Result from Policy-based RL



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

Maze Example: Result from Value-based RL



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

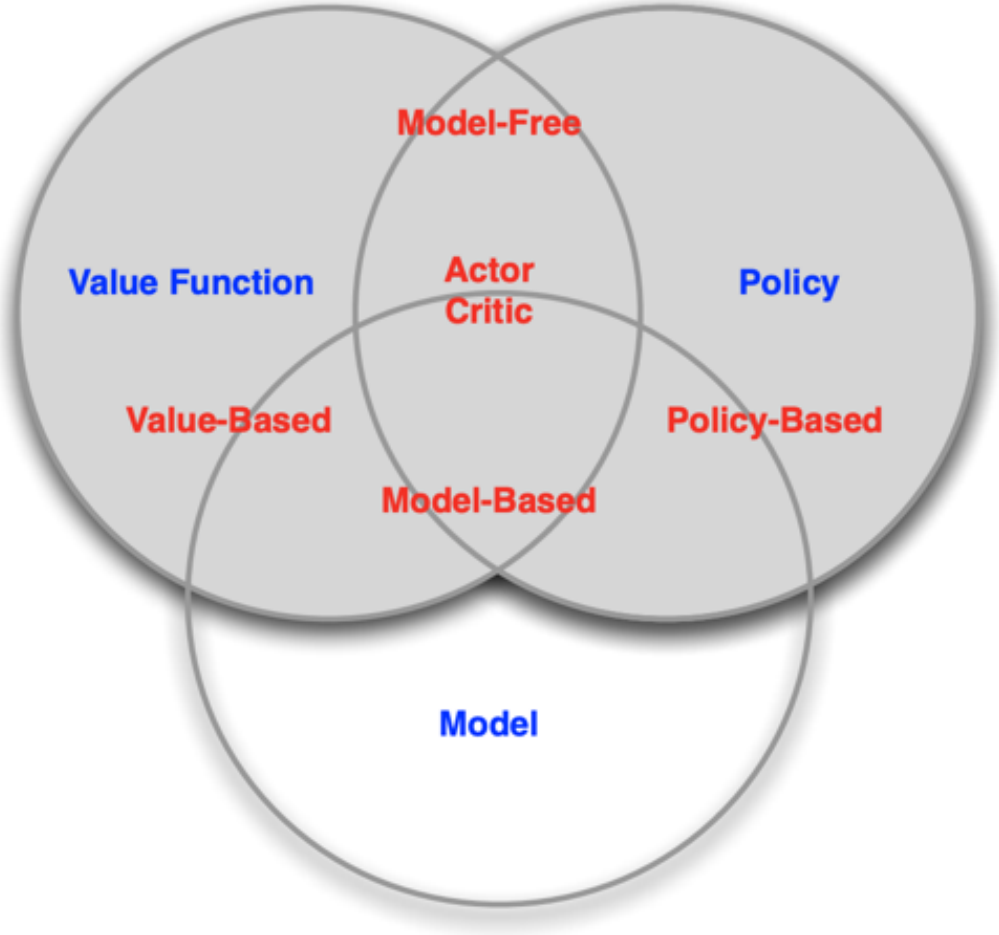
Types of RL Agents based on What the Agent Learns

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



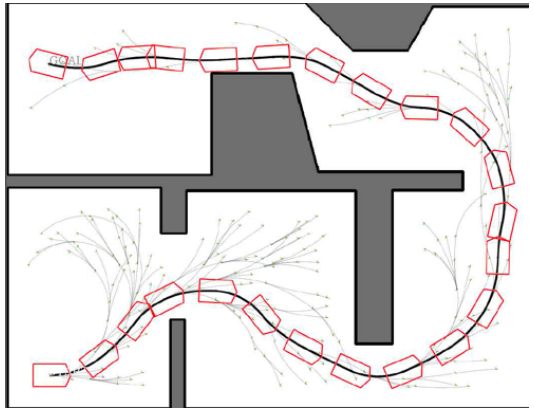
Credit: David Silver's slide

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



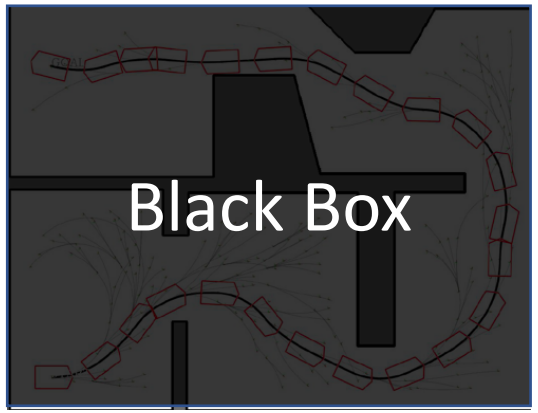
Map is known
All the rules of the vehicle are known
Planning algorithms: dynamic programming, A* search, tree search, …

Patience (单人纸牌游戏)

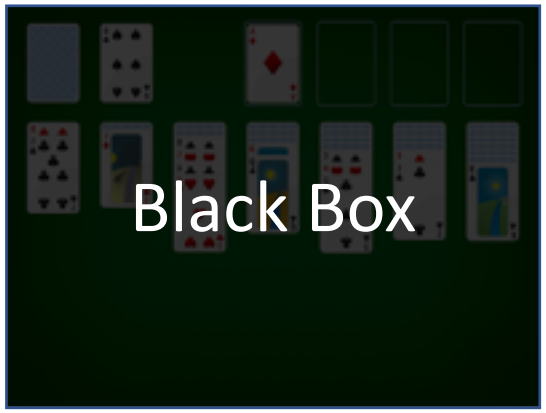
Rules of the game are known.
Planning algorithms: dynamic programming, tree search

Reinforcement Learning

Path planning



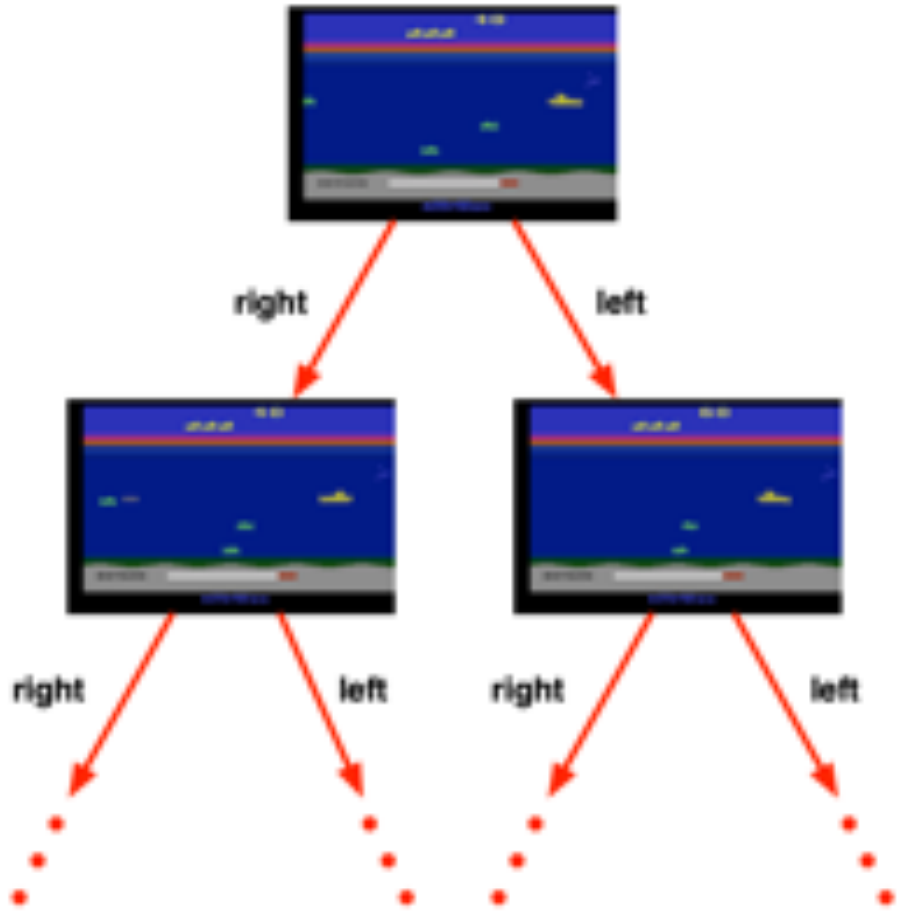
Patience (单人纸牌游戏)



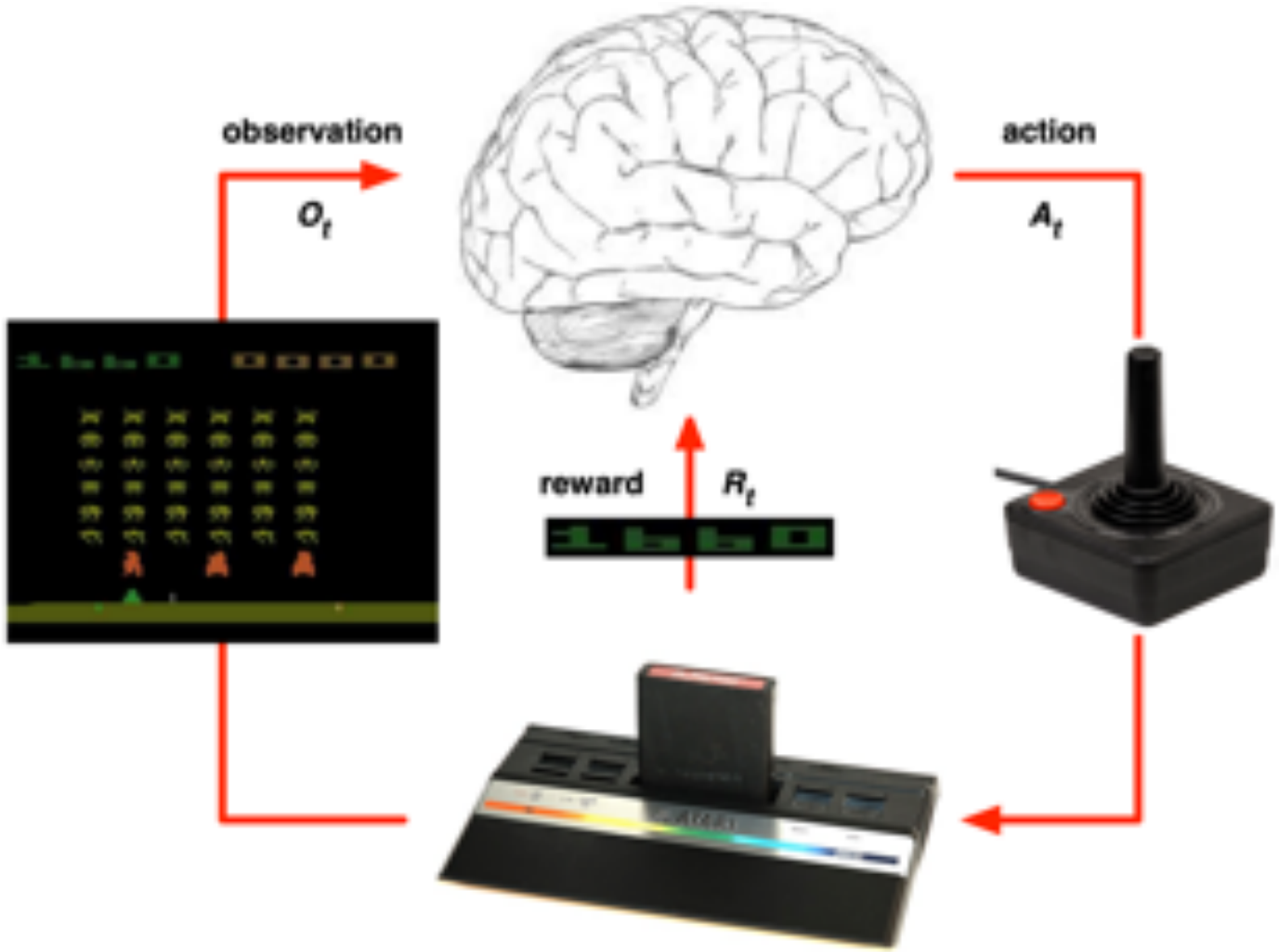
- No rules or knowledge about the environment.
- Learn directly by taking actions and seeing what happens.
- Try to find a good policy over time that yields high reward.
- Planning is needed in inference or forward pass.

Atari Example: 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



Atari Example: Reinforcement Learning



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

Exploration：探索环境得到策略

Exploitation：不探索新环境，采用既有策略来决定

Exploration and Exploitation

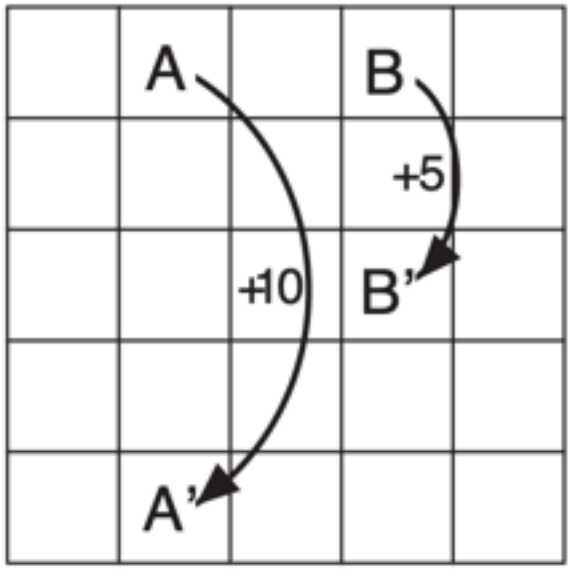
- Agent only experiences what happens for the actions it tries!
- How should an RL agent balance its actions?
  - Exploration: trying new things that might enable the agent to make better decisions in the future
  - Exploitation: choosing actions that are expected to yield good reward given the past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

Exploration and Exploitation

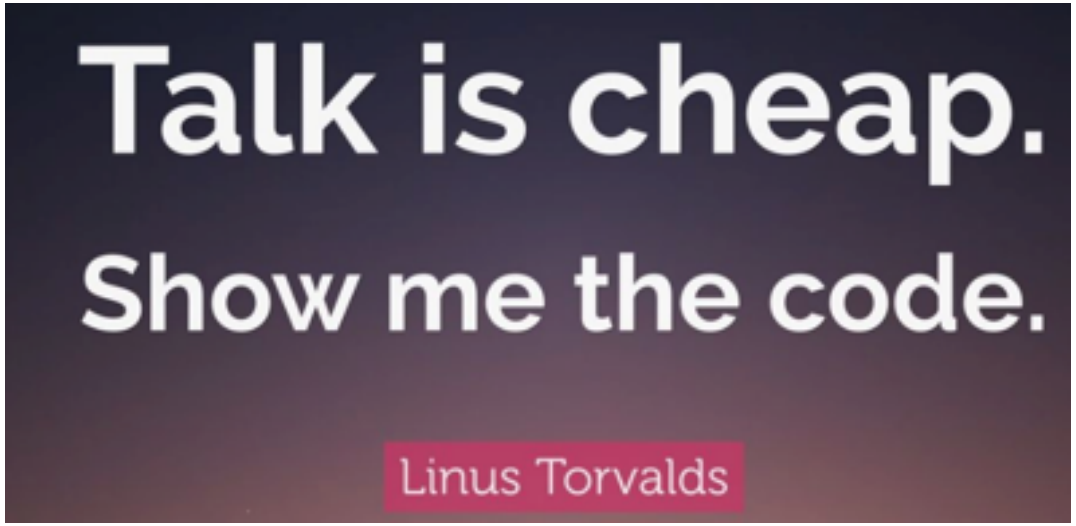
- Restaurant Selection
  - Exploitation: Go to your favourite restaurant
  - Exploration: Try a new restaurant
- Online Banner Advertisements
  - Exploitation: Show the most successful advert
  - Exploration: Show a different advert
- Oil Drilling
  - Exploitation: Drill at the best-known location
  - 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.



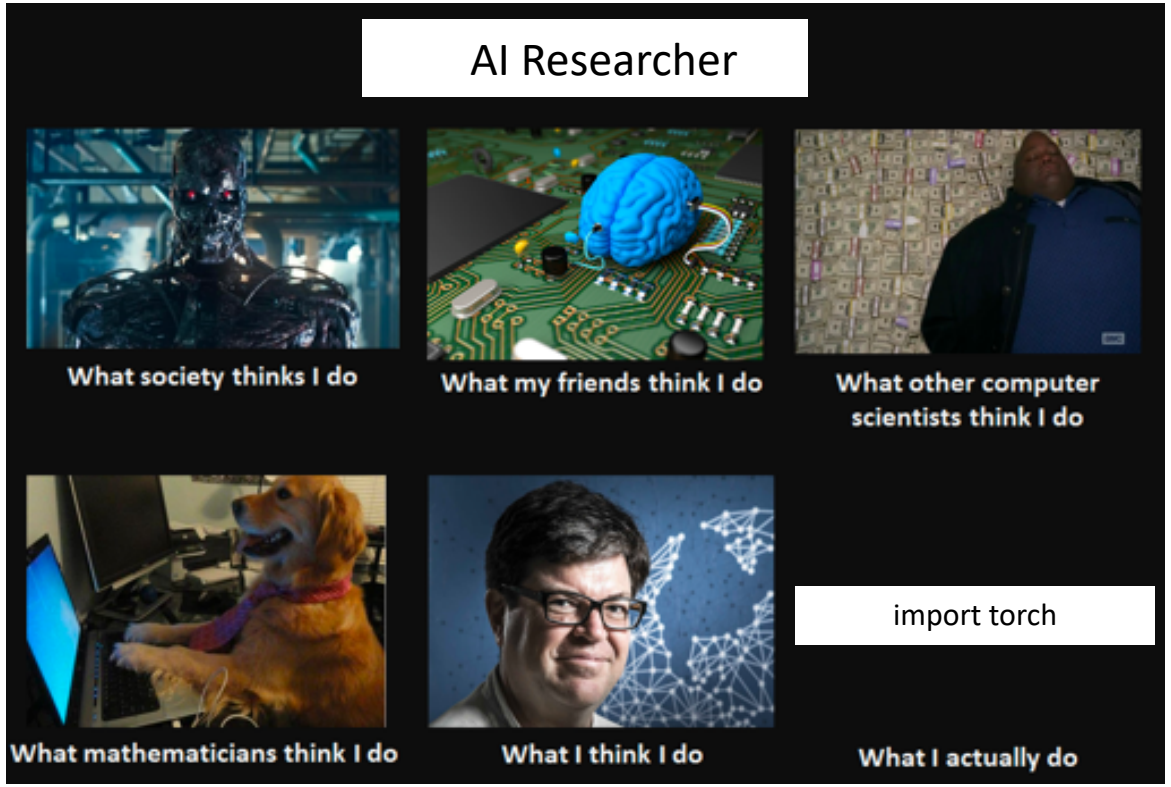
Experimenting with Reinforcement Learning



- Getting hand dirty on reinforcement learning is very important
- Deep learning and AI become more and more empirical
- Trial and error approach to learn reinforcement learning

Coding

- Python coding
- Deep learning libraries: PyTorch or TensorFlow
- https://github.com/metalbubble/RLexample

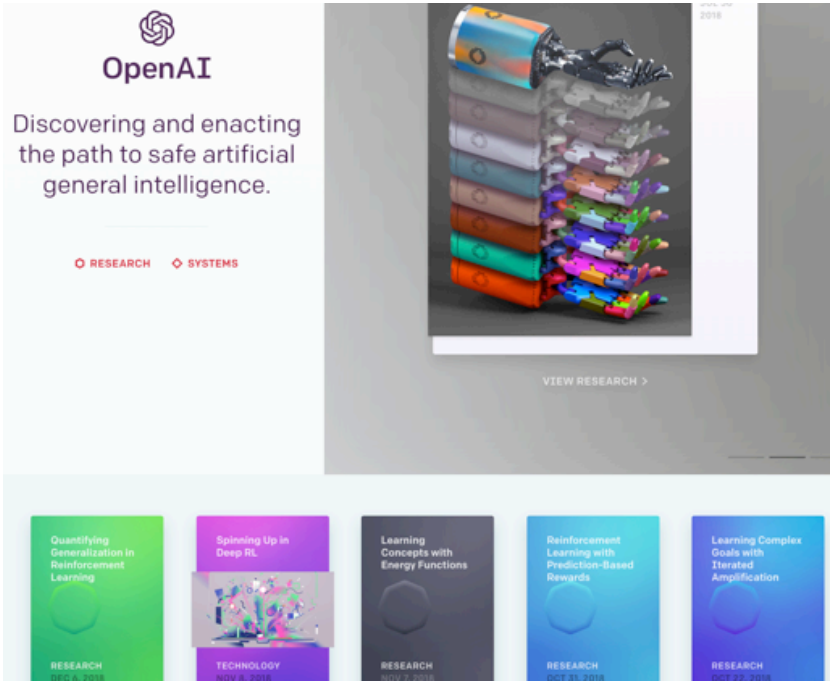


Reinventing Wheels? (造轮子？)

No. Start with existing libraries and pay more attentions to the specific algorithms

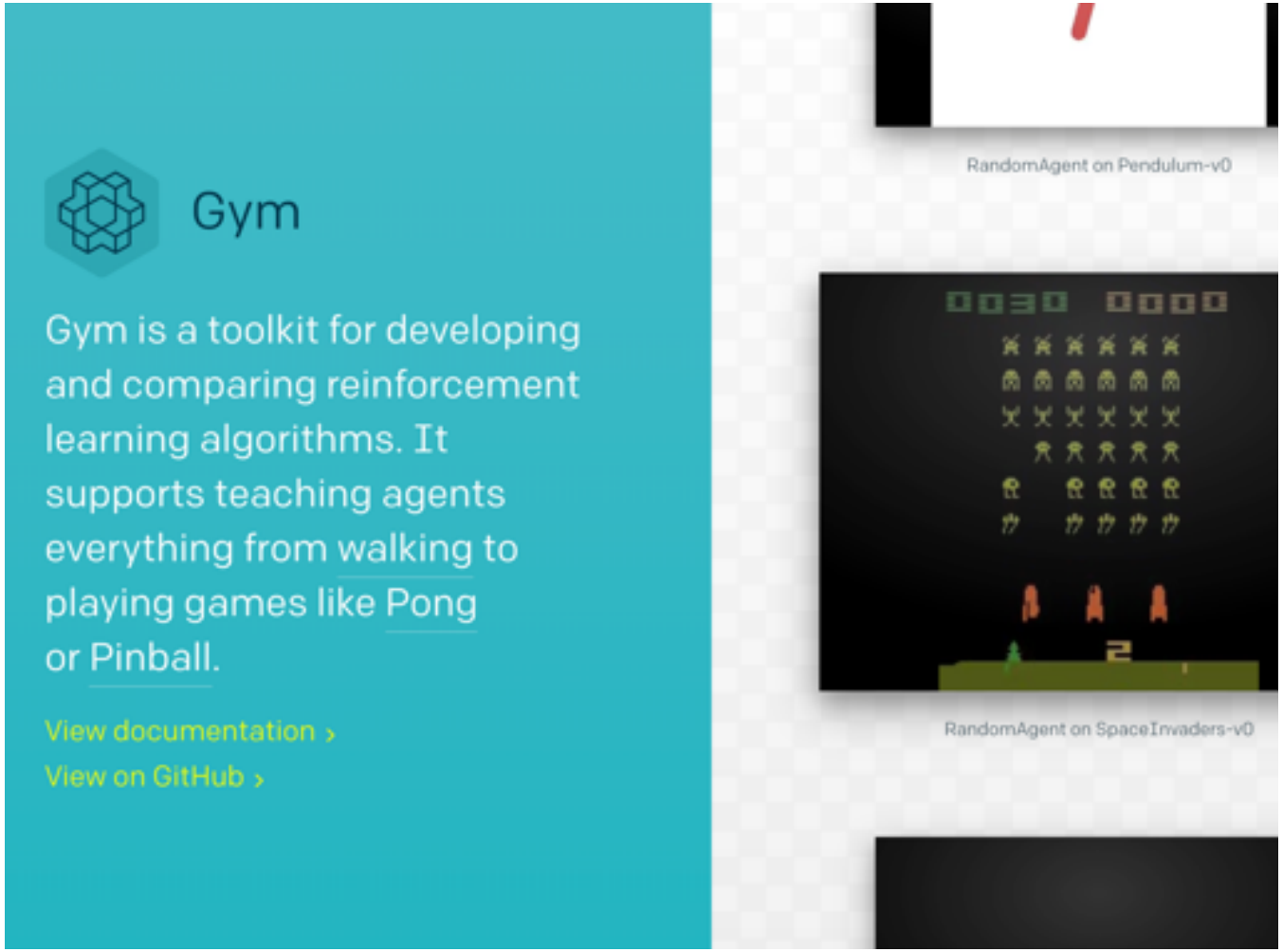
OpenAI: specialized in Reinforcement Learning

- https://openai.com/
- OpenAI is a non-profit AI research company, discovering and enacting the path to safe artificial general intelligence (AGI).



OpenAI gym library

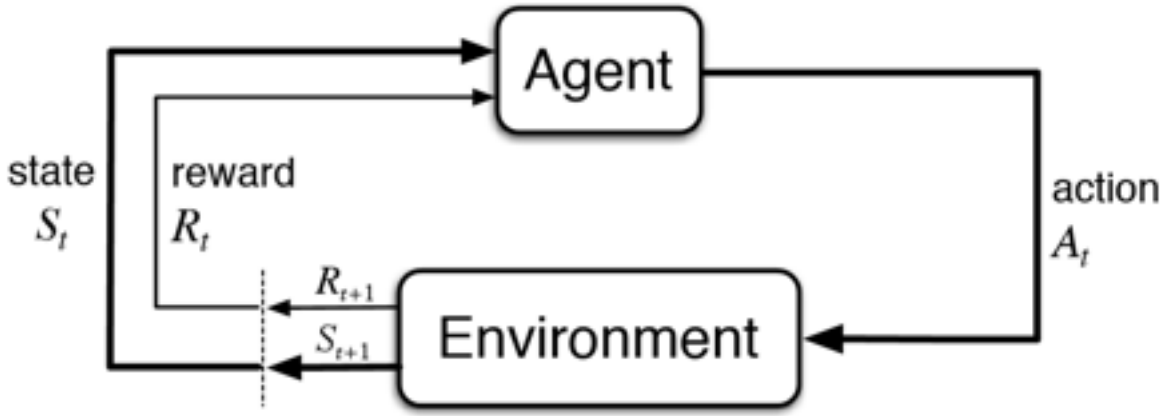
https://gym.openai.com/



https://github.com/openai/retro

We're releasing the full version of Gym Retro, a platform for reinforcement learning research on games. This brings our publicly-released game count from around 70 Atari games and 30 Sega games to over 1,000 games across a variety of backing emulators. We're also releasing the tool we use to add new games to the platform.

Algorithmic interface of reinforcement learning




```
import gym
env = gym.make("Taxi-v2")
observation = env.reset()
agent = load_agent()
for step in range(100):
    action = agent(observation)
    observation, reward, done, info = env.step(action)
```

Classic Control Problems

Acrobot-v1
Swing up a two-link robot.

CartPole-v1
Balance a pole on a cart.

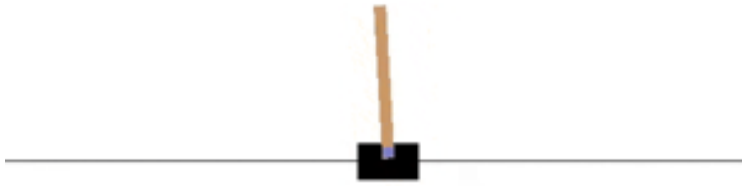
MountainCar-v0
Drive up a big hill.

MountainCarContinuous-v0
Drive up a big hill with continuous control.

Pendulum-v0
Swing up a pendulum.

https://gym.openai.com/envs/#classic_control

Example of CartPole-v0



Actions

Type: Discrete(2)

Num	Action
0	Push cart to the left
1	Push cart to the right

Observation

Type: Box(4)

Num	Observation	Min	Max
0	Cart Position	-2.4	2.4
1	Cart Velocity	-Inf	Inf
2	Pole Angle	~ -41.8°	~ 41.8°
3	Pole Velocity At Tip	-Inf	Inf

Reward

Reward is 1 for every step taken, including the termination step

Episode Termination

1. Pole Angle is more than ±12°
2. Cart Position is more than ±2.4 (center of the cart reaches the edge of the display)
3. Episode length is greater than 200

https://github.com/openai/gym/blob/master/gym/envs/classic_control/cartpole.py

Example code

```
import gym
env = gym.make('CartPole-v0’)
env.reset()
env.render() # display the rendered scene
action = env.action_space.sample()
observation, reward, done, info = env.step(action)
```

Example code: Random Agent

python my_random_agent.py CartPole-v0

python my_random_agent.py Pong-ram-v0

python my_random_agent.py Breakout-v0

What is the difference in the format of the observations?

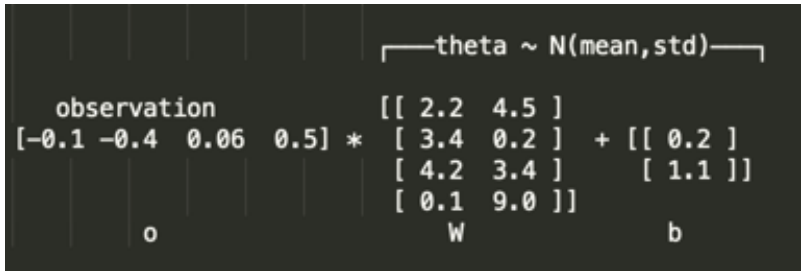
Example code: Naïve learnable RL agent

python my_random_agent.py CartPole-v0

python my_random_agent.py Acrobot-v1

python my_learning_agent.py CartPole-v0

python my_learning_agent.py Acrobot-v1



P_a = oW+b

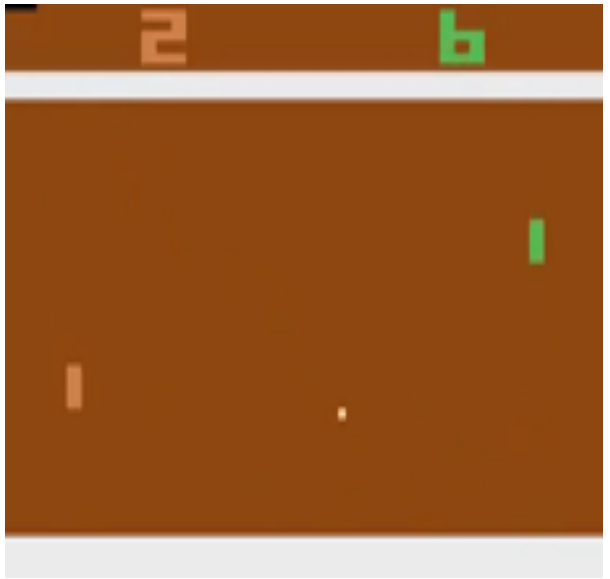
What is the algorithm?

Cross Entropy method (CEM)
https://gist.github.com/kashif/5dfa12d80402c559e060d567ea352c06

Deep Reinforcement Learning Example

- Pong example

```
import gym
env = gym.make('Pong-v0')
env.reset()
env.render() # display the rendered scene
```



```
python my_random_agent.py Pong-v0
```

Deep Reinforcement Learning Example

- Pong example

python pg-pong.py

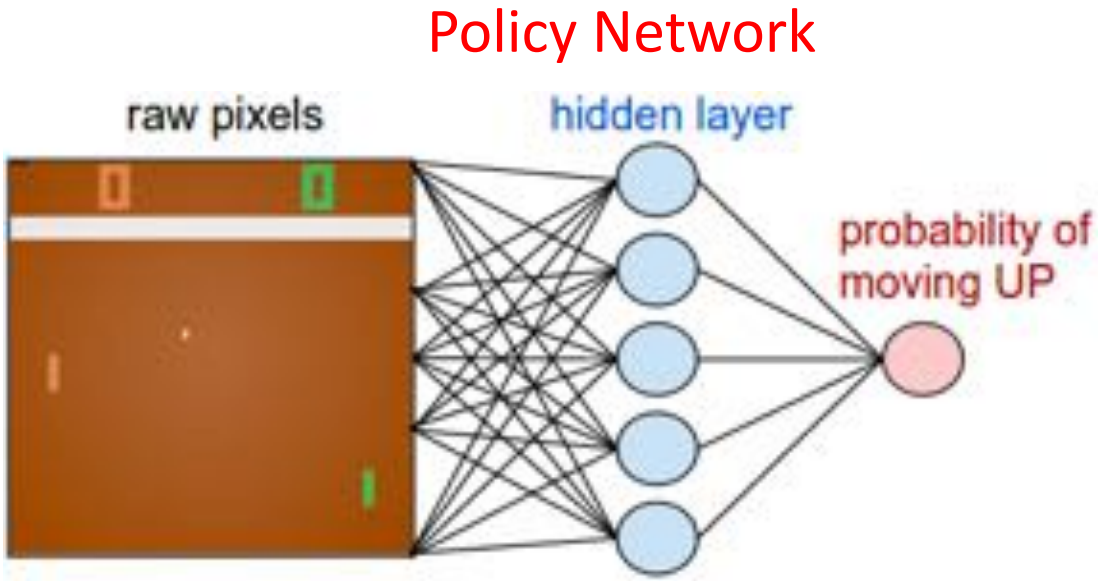
Loading weight: pong_bolei.p (model trained over night)

Deep Reinforcement Learning Example

- Look deeper into the code

```
observation = env.reset()

cur_x = prepro(observation)
x = cur_x - prev_x
prev_x = cur_x
aprob, h = policy_forward(x)
```



Randomized action:

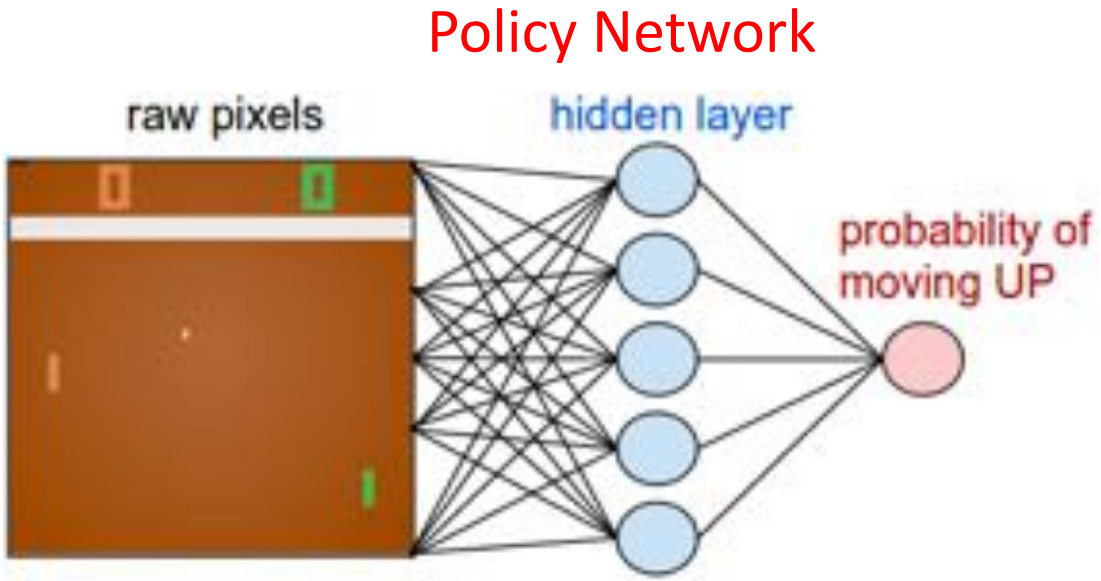
```
action = 2 if np.random.uniform() < aprob else 3 # roll the dice!
```

http://karpathy.github.io/2016/05/31/rl/

Deep Reinforcement Learning Example

- Look deeper into the code

```
h = np.dot(W1, x)
h[h<0] = 0 # ReLU nonlinearity: threshold at zero logp = np.dot(W2, h) # compute log probability of going up
p = 1.0 / (1.0 + np.exp(-logp)) # sigmoid function (gives probability of going up)
```



http://karpathy.github.io/2016/05/31/rl/

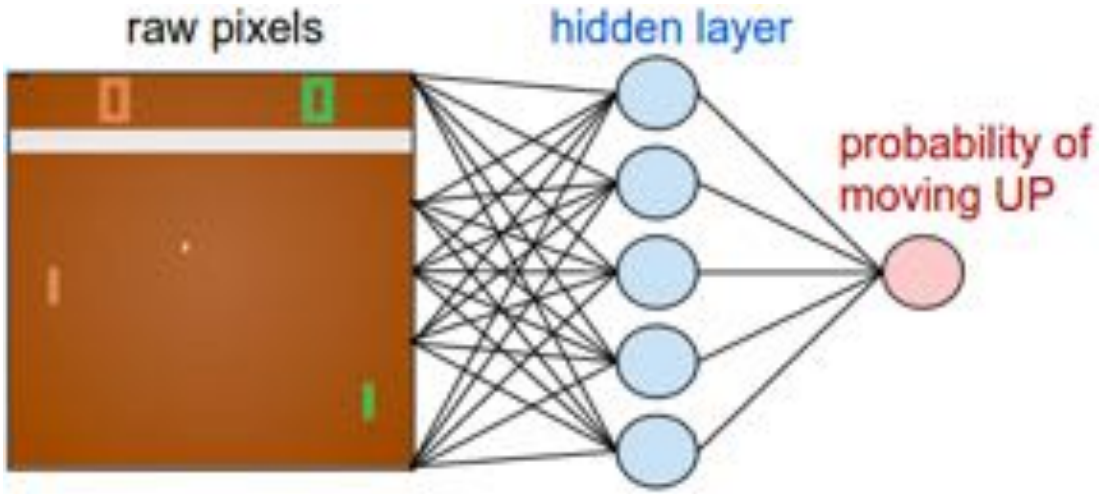
Deep Reinforcement Learning Example

- Look deeper into the code

How to optimize the W1 and W2?

Policy Gradient! (To be introduced in future lecture)

Policy Network



http://karpathy.github.io/2016/05/31/rl/

What could be the potential problems?

```
import gym
env = gym.make("Taxi-v2")
observation = env.reset()
agent = load_agent()
for step in range(100):
    action = agent(observation)
    observation, reward, done, info = env.step(action)
```

Speed, multiple agents, architecture of agent?

Homework and What's Next

- Play with OpenAI gym and the example code

https://github.com/cuhkrlcourse/RLexample

- Go through this blog in detail to understand pg-pong.py

http://karpathy.github.io/2016/05/31/rl/

- Next week: Markov Decision Process

Please read **Sutton and Barton: Chapter 1 and Chapter 3**

In case you need it

- Python tutorial: http://cs231n.github.io/python-numpy-tutorial/
- PyTorch tutorial: https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

Summary

- Course overview
- Introduction to reinforcement learning
- Introduction to sequential decision making
- Experimenting with RL by coding

Next

Lecture 2: Markov Decision Process