CPSC 340:

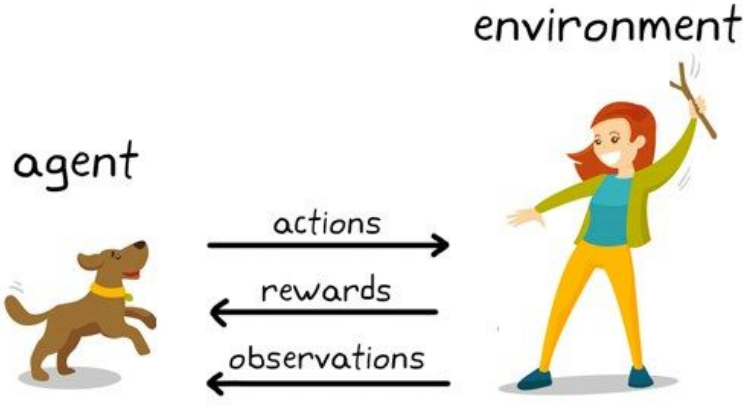
Machine Learning and Data Mining

Introduction to Reinforcement Learning -- Bonus Lecture

Helen Zhang
(slides adapted from Daniele Reda)
Fall 2021

Today’s Plan:

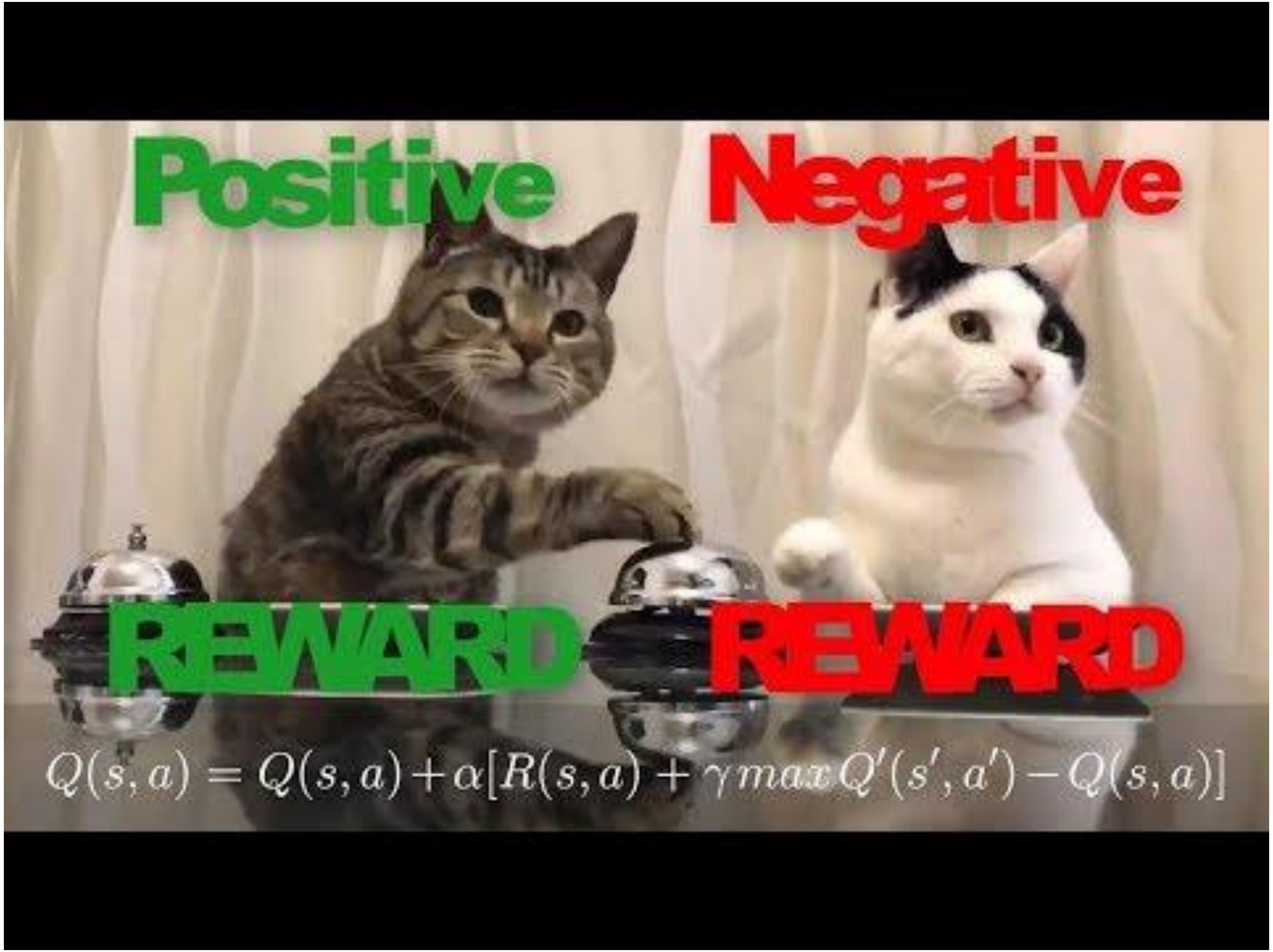
- What is RL
- Funny videos
- Q-learning, DQN
- Self-driving car



Law of Effect

"responses that produce a satisfying effect in a particular situation become more likely to occur again in that situation, and responses that produce a discomforting effect become less likely to occur again in that situation."

Edward Thorndike

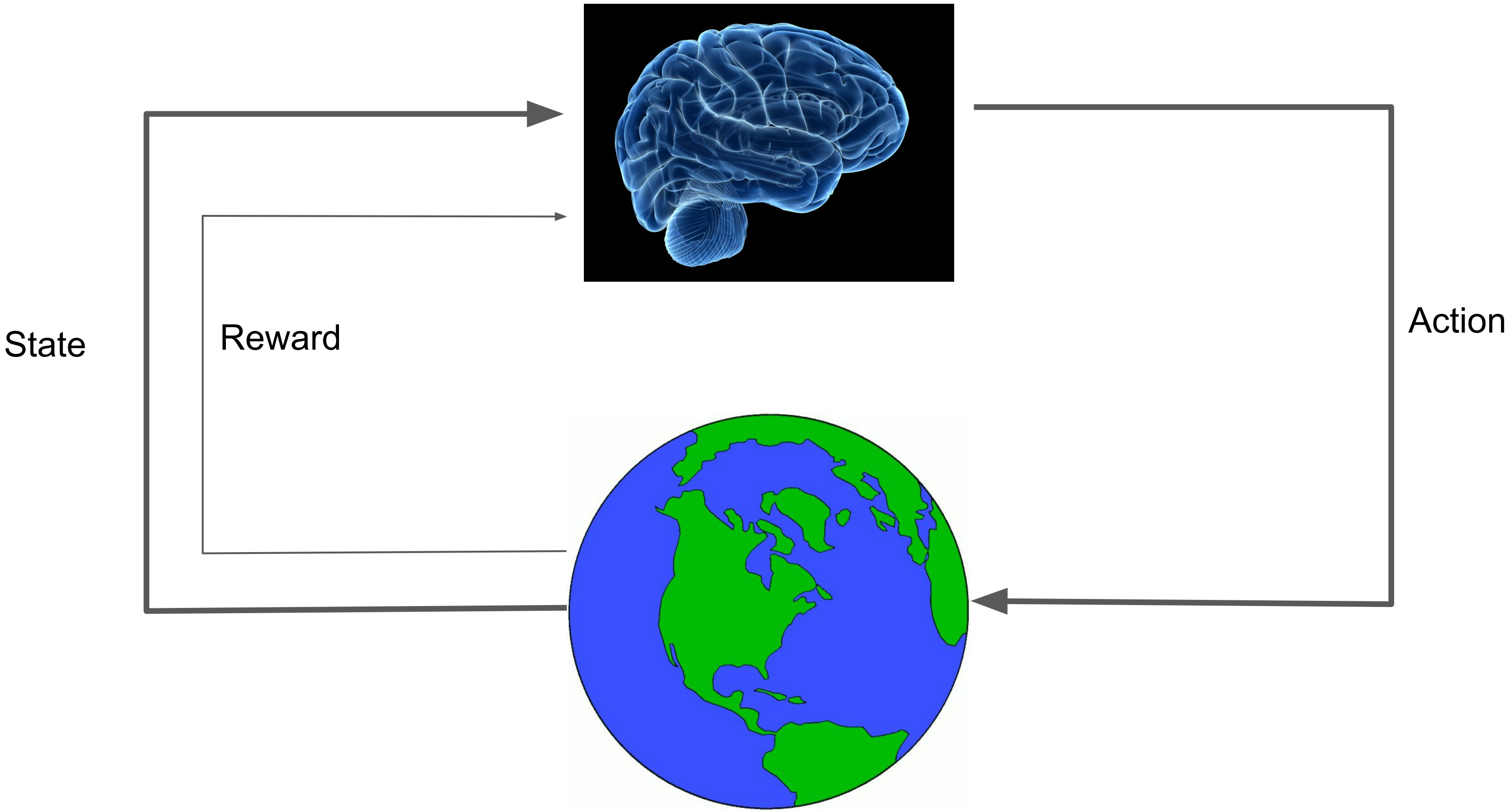


$$Q(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma max Q'(s',a') - Q(s,a)]$$

What is Reinforcement Learning?

- learning by trial and error
- learning by interacting with environment

Problem Setting

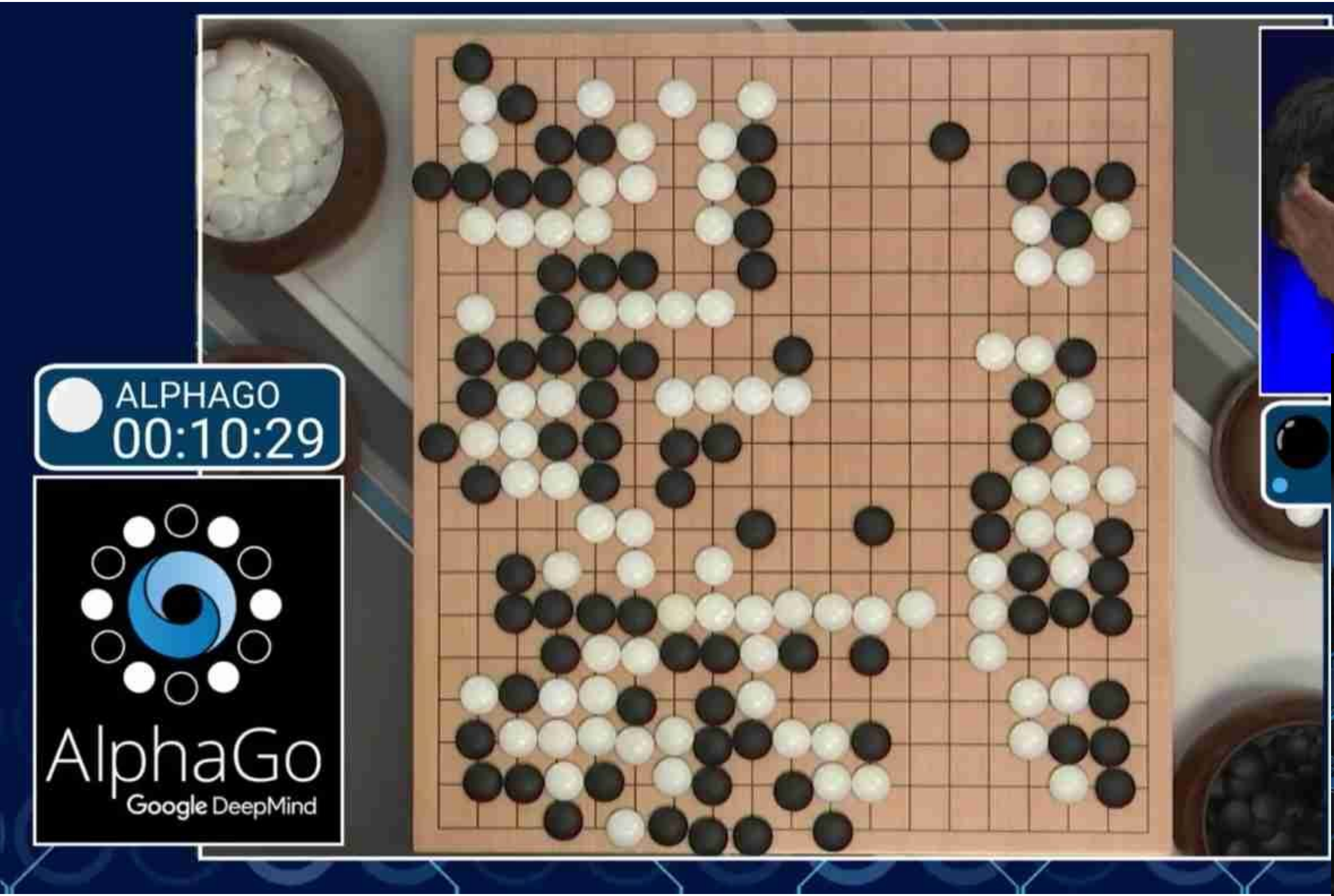


State

Reward

Action

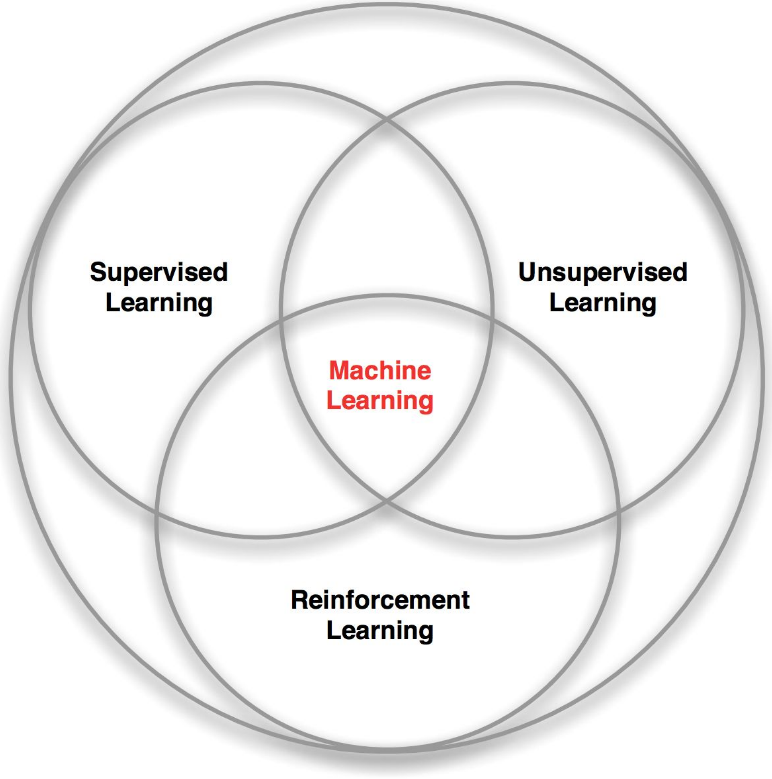
Alpha Go



nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
At last — a computer program that can beat a champion Go player PAGE 484
ALL SYSTEMS GO

Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search."

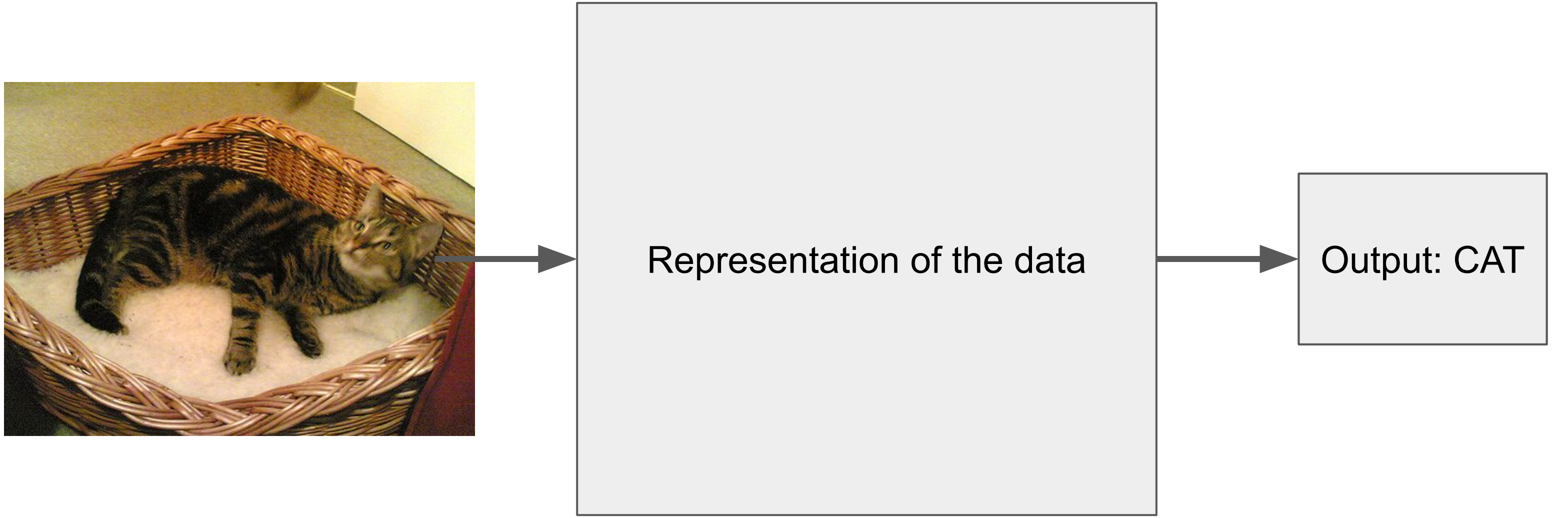
Branches of Machine Learning



Supervised Learning

Data	Label
X1	Y1
X2	Y2
X3	Y3
...	...

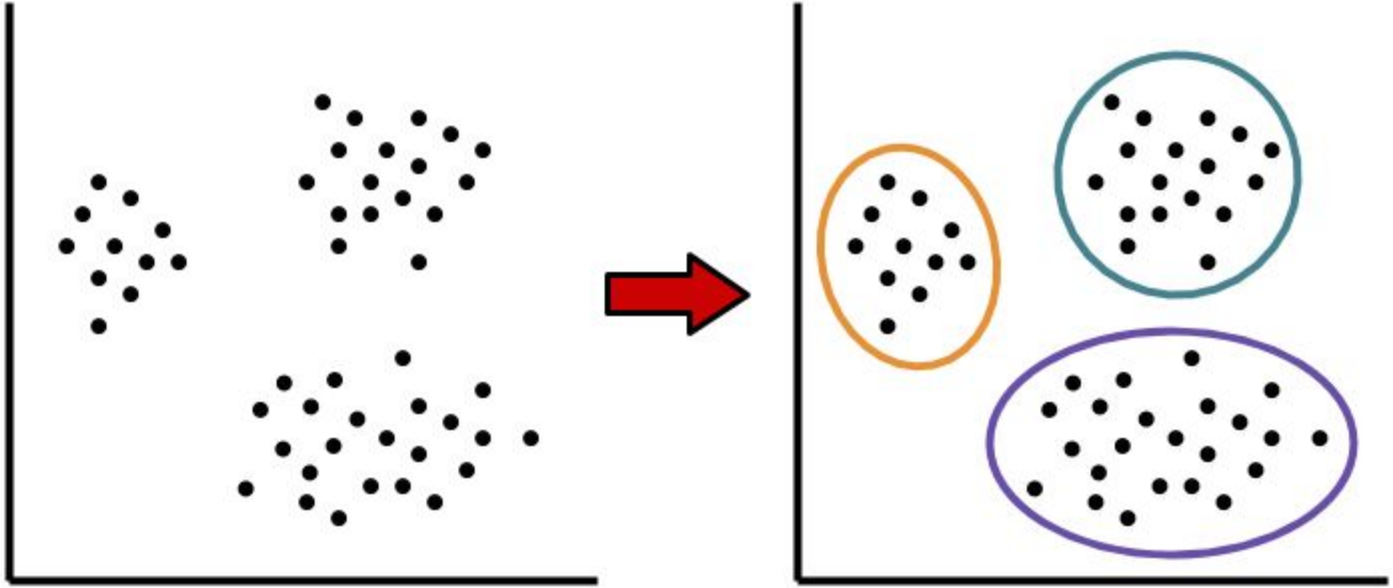
Supervised Learning



Representation of the data

Output: CAT

Unsupervised Learning

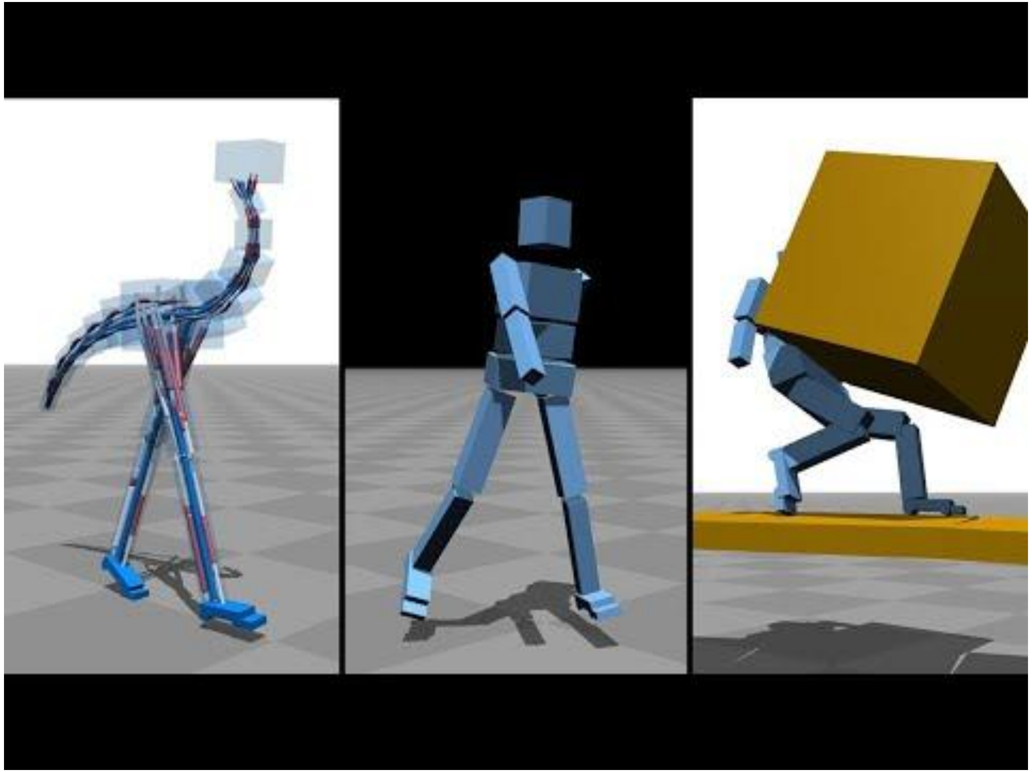


Characteristics

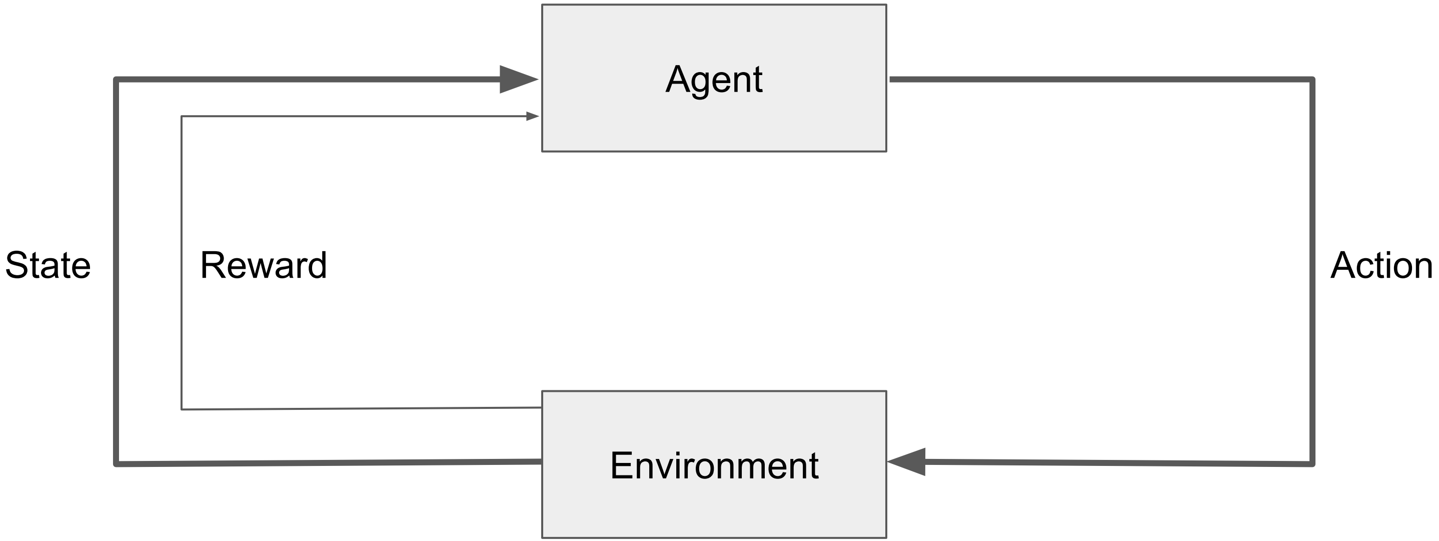
- no supervisor, only a reward signal
- feedback is delayed, not instantaneous
- process is iterative (time matters)
- agent’s actions affect subsequent data it receives

Flipping pancakes

Walking simulation



Problem Setting



Goal

- maximize total reward

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T$$

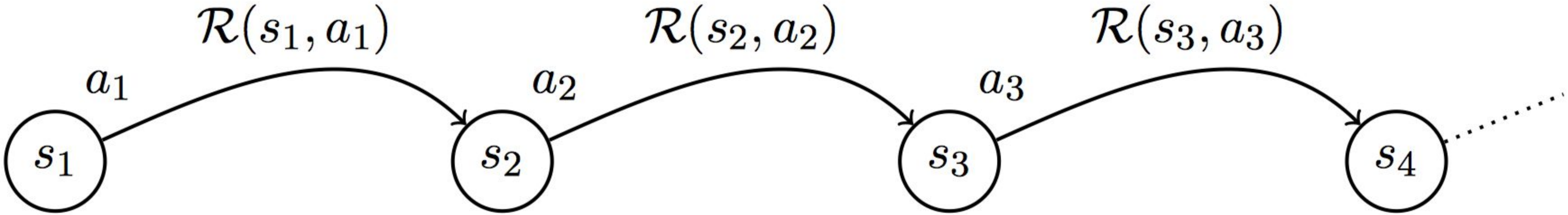
- T to infinity
- Why discount factor?
    - convergence
    - sooner rewards are usually more useful than later ones

Value Function

$$V^\star(S) = max_a \left[R(s,a) + \gamma \sum_{s'} p(s'|s,a) V^\star(s')\right]$$

- Problems:
  - it doesn’t tell which actions to take

Representation of the system



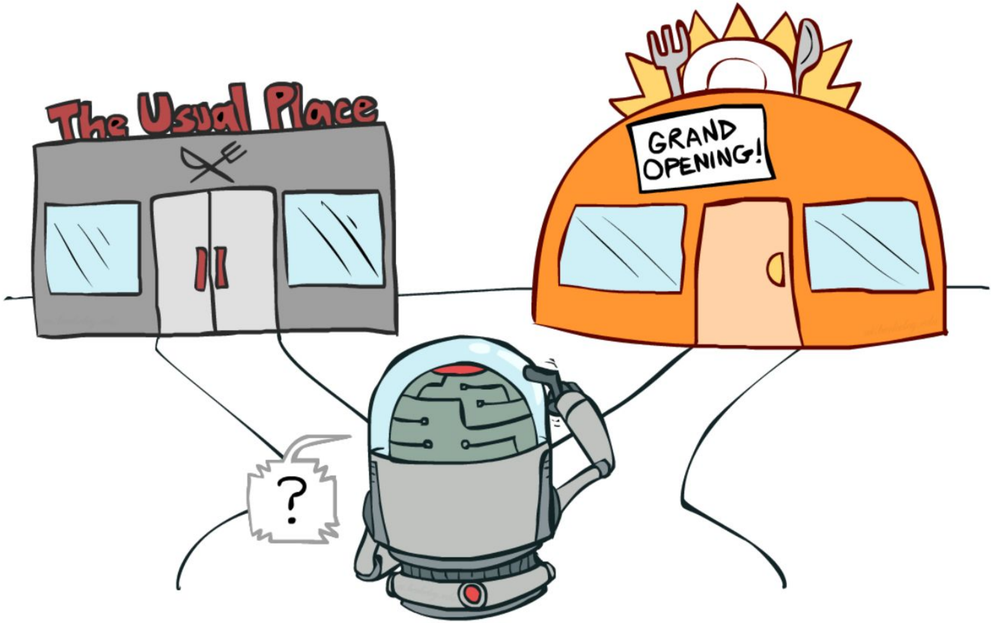
Q-Value Function

$$Q^{\star}(s, a) = R(s, a) + \gamma \textstyle\sum_{s'} p(s' \mid s, a) \max\limits_{\alpha} \left(Q^{\star}(s', \alpha)\right)$$

A very simple algorithm: Q-learning

		Actions					
		0	1	2	3	4	5
States	0	-1	-1	-1	-1	0	-1
	1	-1	-1	-1	0	1	100
	2	-1	-1	-1	0	-1	-1
	3	-1	0	0	-1	0	-1
	4	0	-1	-1	0	-1	100
	5	-1	0	-1	-1	0	100

Exploration vs Exploitation



One strategy: ε-greedy

Q-learning

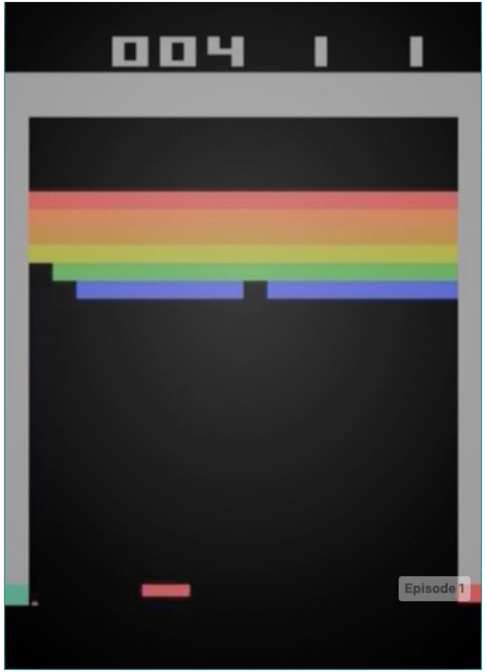
Initialize Q-table with random values.

1. Choose action a to perform in current state s. (ε-greedy)
2. Perform a and receive reward R(s,a).
3. Observe new state S(s,a).
4. Update Q-table.

$$Q'\left(s,a\right) \leftarrow \mathcal{R}\left(s,a\right) + \gamma \max_{\alpha} \left\{Q'\left(\mathcal{S}(s,a),\alpha\right)\right\}$$

PROBLEM:
TABLE CAN EASILY EXPLODE IN DIMENSIONS

Let’s look at an example: ATARI



State (the actual image)
84x84x4 pixels (gray-scale)

2 Actions
Left-Right

Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning.

Could we use a q-table?

Atari Breakout example:

State = raw pixels of last 4 frames (84x84) with 256 different possible values.

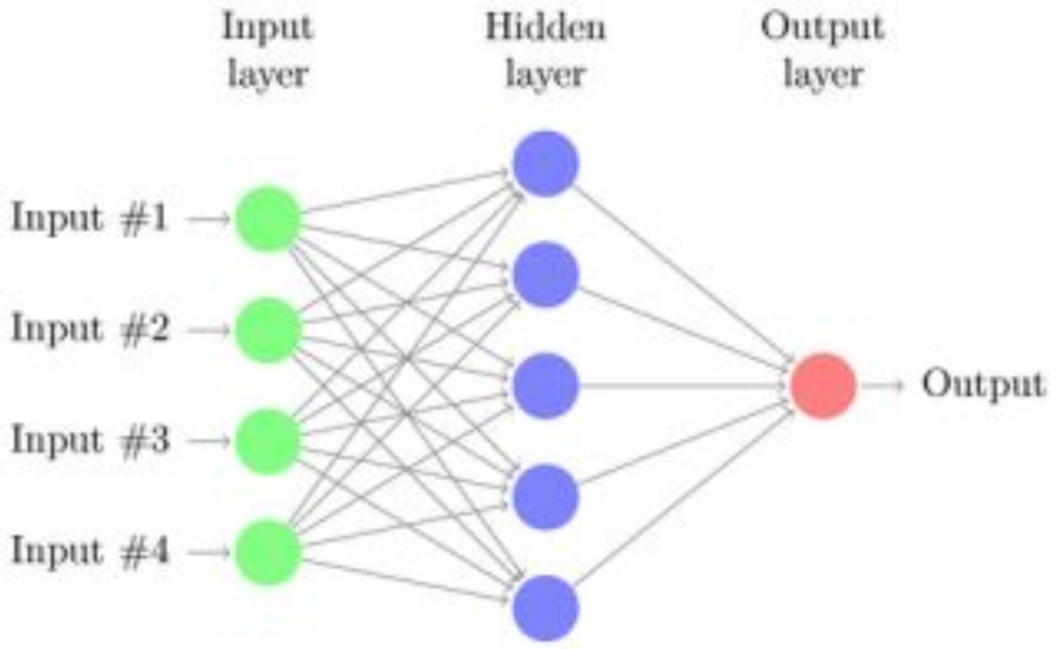
Actions = 2 actions available

(4x84x84x256)x2 = 14450688 different values

Solution

Neural networks!

Neural Networks

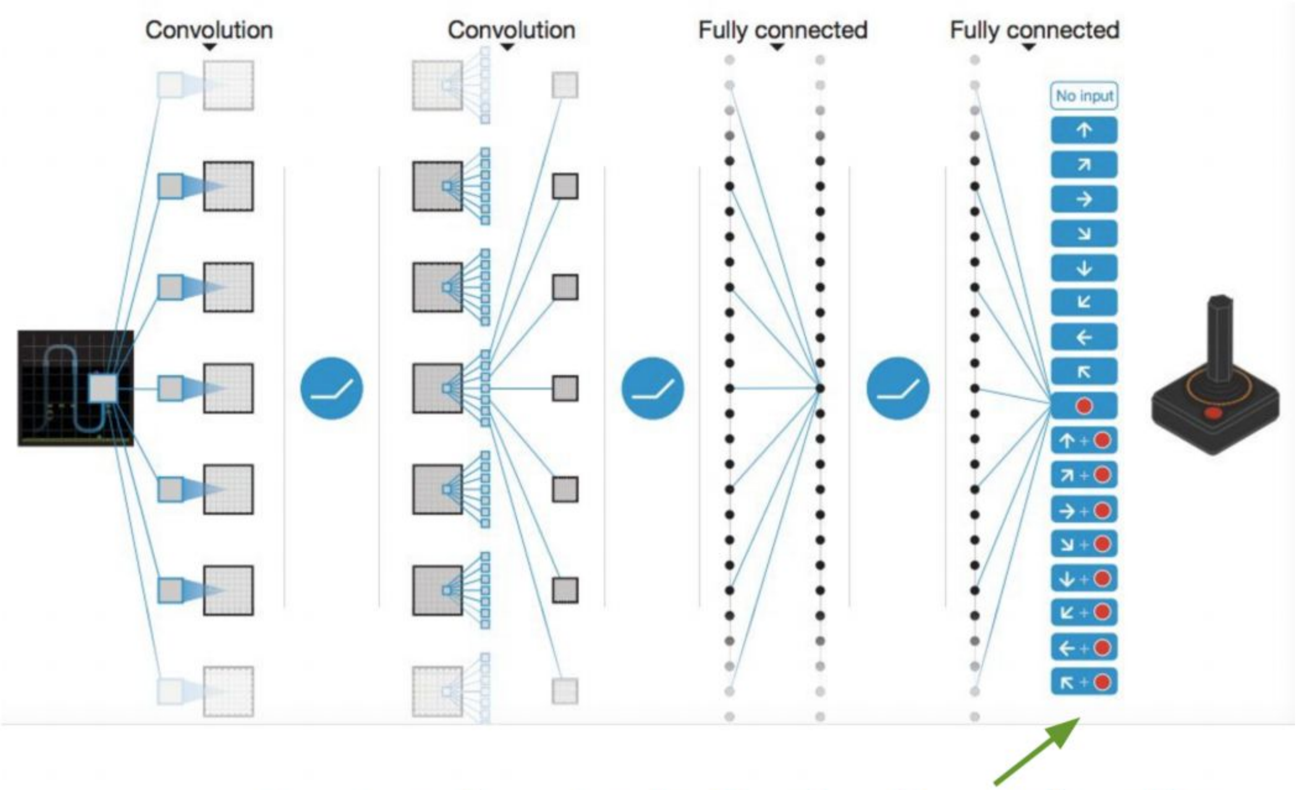


Neural Networks

$$loss = \left(\underbrace{\overset{\text{Reward}}{\mathrm{r}} + \overset{\text{Decay Rate}}{\gamma} \max_{a`} \hat{Q}(s,a`)}_{\text{Target}} - \underbrace{Q(s,a)}_{\text{Prediction}}\right)^2$$

$$Q'\left(s,a\right) \leftarrow \mathcal{R}\left(s,a\right) + \gamma \max_{\alpha} \left\{Q'\left(\mathcal{S}(s,a),\alpha\right)\right\}$$

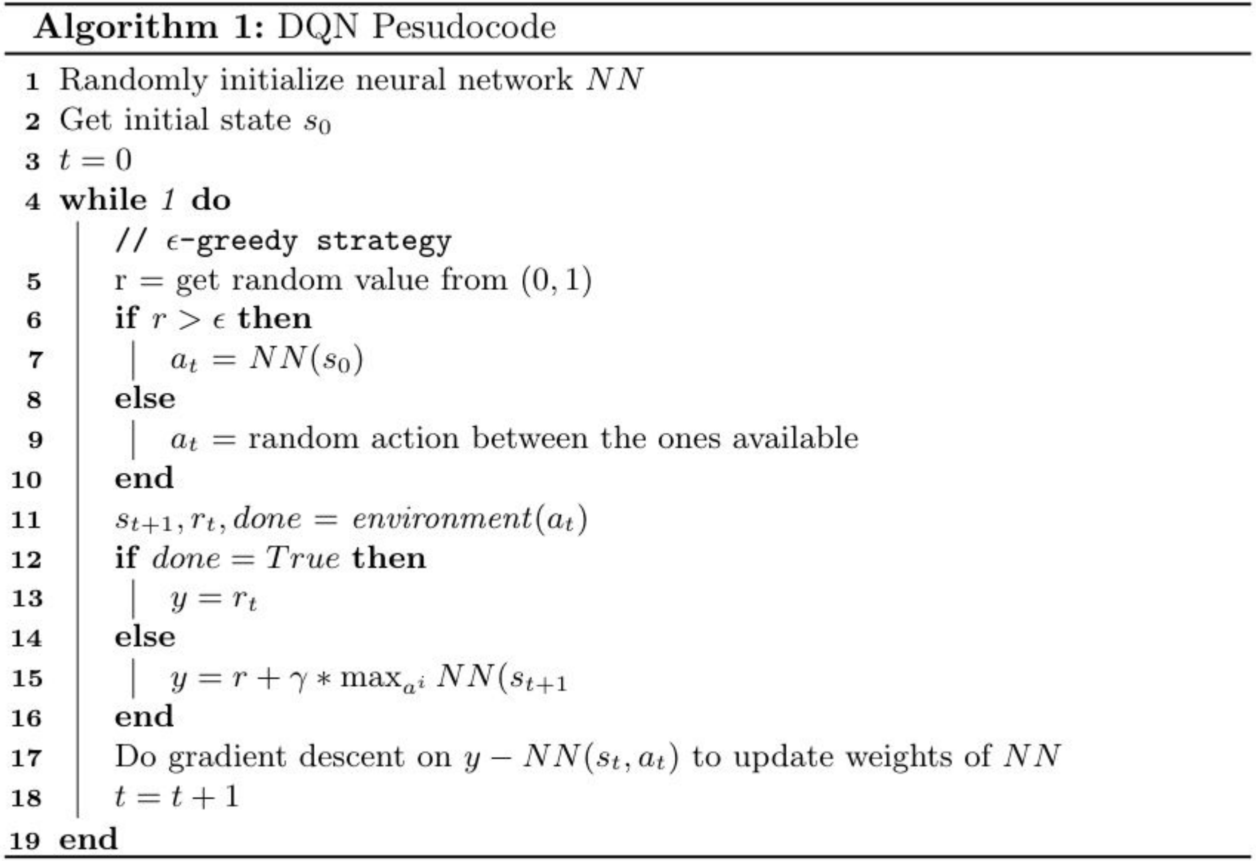
DQN Framework



1 network, outputs Q value for each action

Image from Emma Brunskill's RL course CS234, Stanford

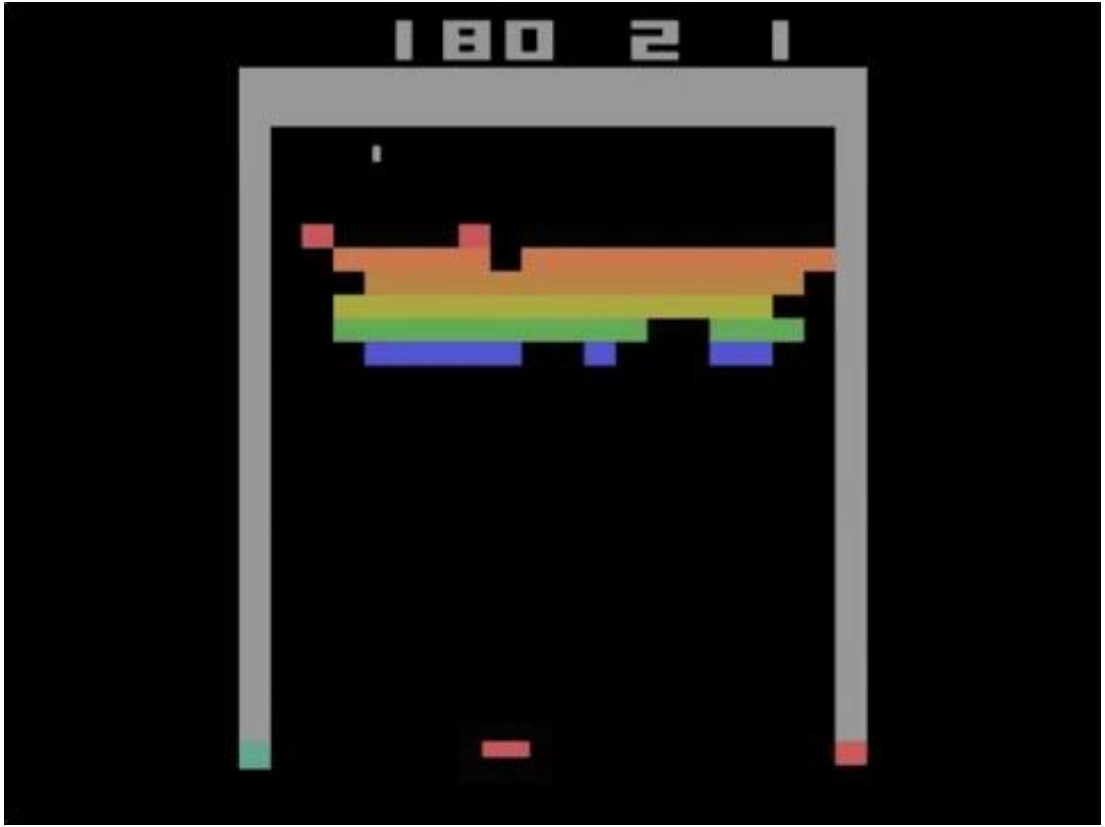
DQN Framework



Algorithm 1: DQN Pesudocode

```
Randomly initialize neural network $NN$
Get initial state $s_0$
$t = 0$
while 1 do
    // $\epsilon$-greedy strategy
    r = get random value from $(0, 1)$
    if $r > \epsilon$ then
        $a_t = NN(s_0)$
    else
        $a_t$ = random action between the ones available
    end
    $s_{t+1}, r_t, done = environment(a_t)$
    if $done = True$ then
        $y = r_t$
    else
        $y = r + \gamma * \max_{a^i} NN(s_{t+1}$
    end
    Do gradient descent on $y - NN(s_t, a_t)$ to update weights of $NN$
    $t = t + 1$
end
```

Famous successes of RL: Atari Breakout



Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning.

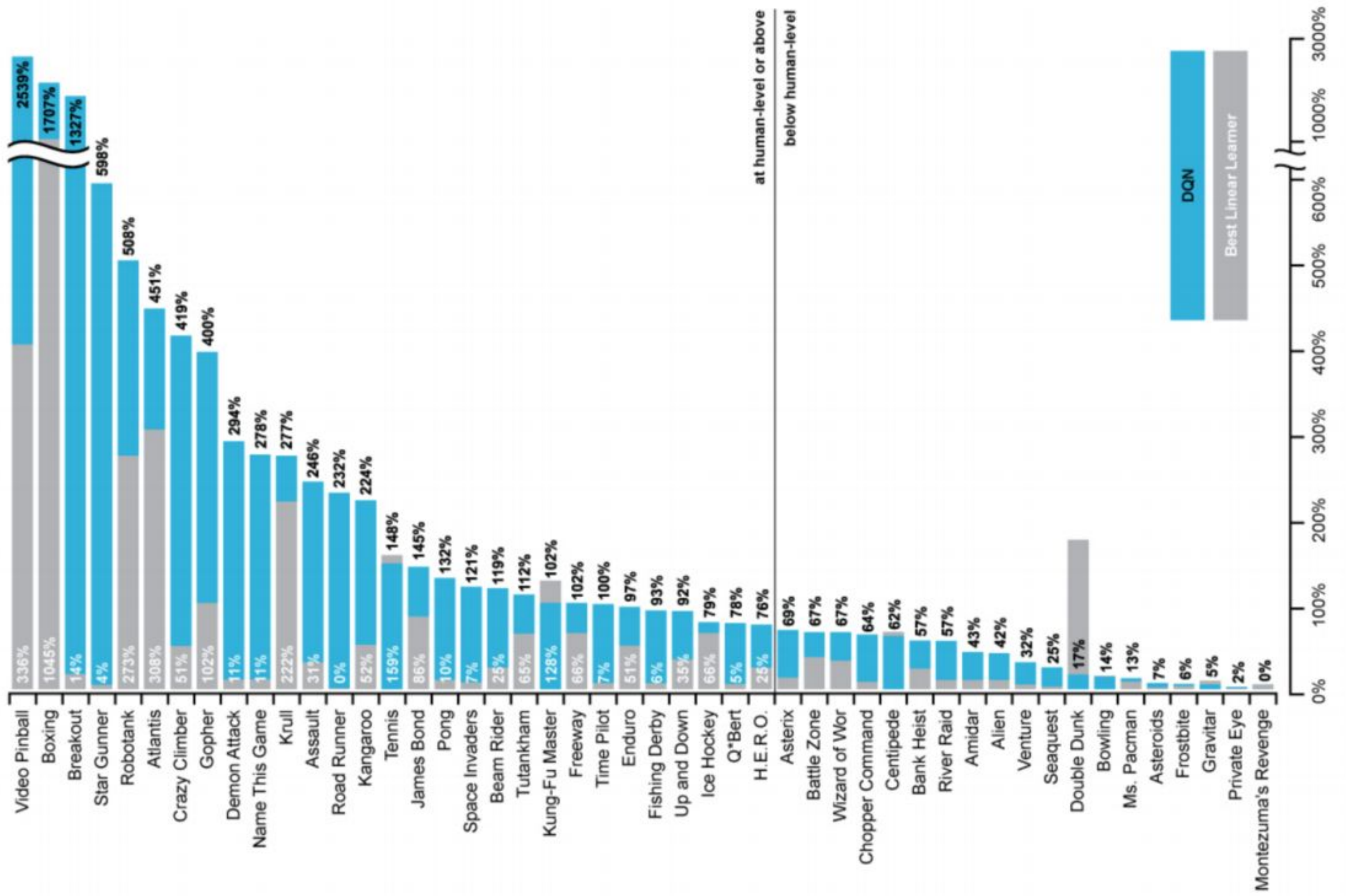
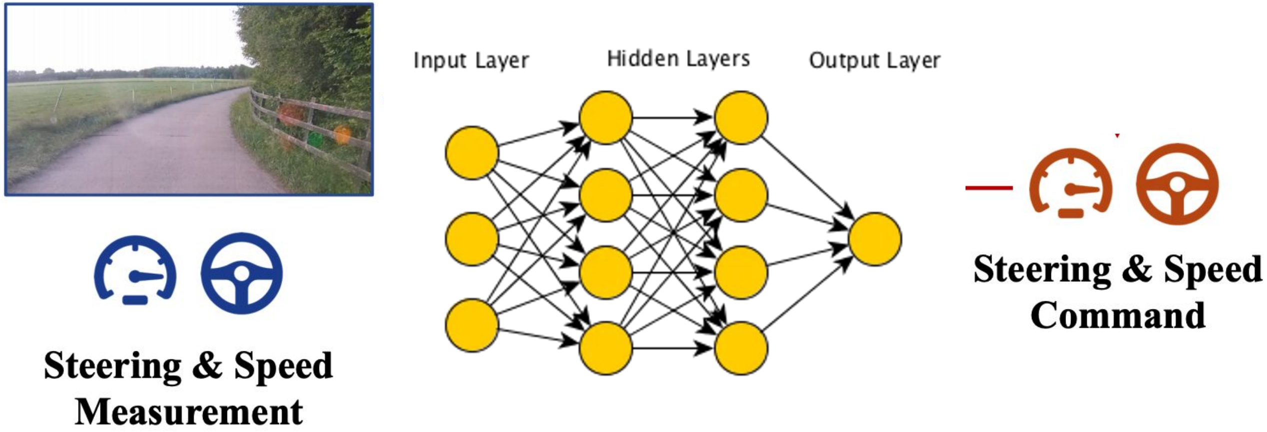


Image from Emma Brunskill's RL course CS234, Stanford

A real world example:
Learning to drive with Reinforcement Learning

Kendall, Alex, et al. "Learning to drive in a day."

Learning to drive with RL



Reward: forward distance

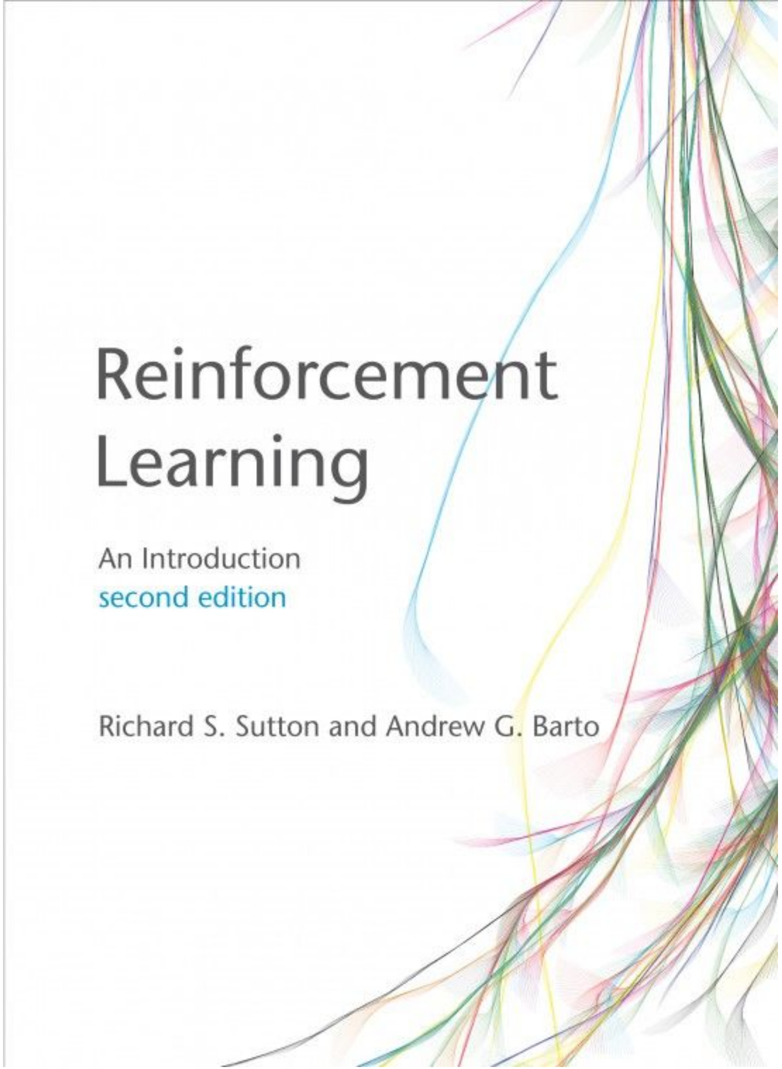
Terminate when it goes out of the lane

Kendall, Alex, et al. "Learning to drive in a day."

Kendall, Alex, et al. "Learning to drive in a day."

Where to go from here?

- openAI Spinning Up RL
- Sutton book
- David Silver UCL Course



Reinforcement Learning

An Introduction

second edition

Richard S. Sutton and Andrew G. Barto