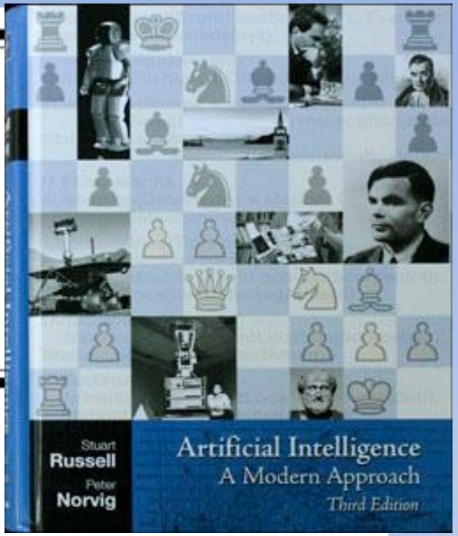
21 REINFORCEMENT LEARNING

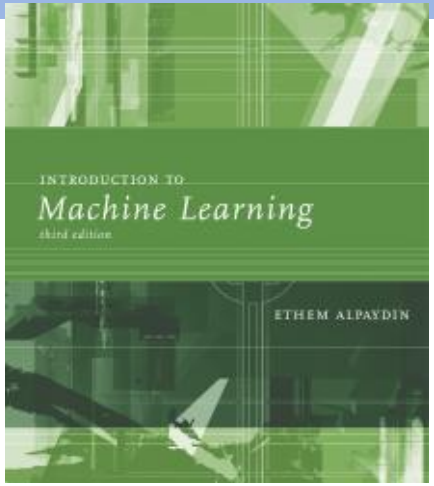


In which we examine how an agent can learn from success and failure, from reward and punishment.

- Introduce Reinforcement Learning
- Understand relationship to MDP
- Introduce MDP

Reinforcement Learning

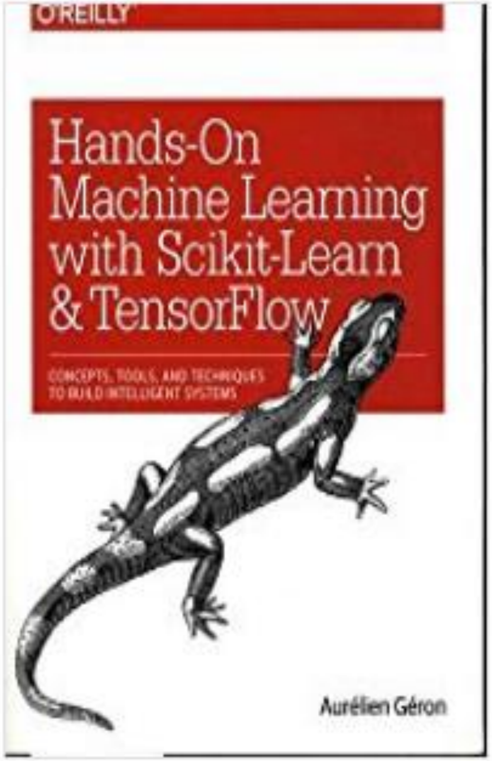
18 *Reinforcement Learning*



In reinforcement learning, the learner is a decision-making agent that takes actions in an environment and receives reward (or penalty) for its actions in trying to solve a problem. After a set of trial-and-error runs, it should learn the best policy, which is the sequence of actions that maximize the total reward.

18 *Reinforcement Learning* 517

16. Reinforcement Learning



Chapter 16. Reinforcement Learning

Reinforcement Learning (RL) is one of the most exciting fields of Machine Learning today, and also one of the oldest. It has been around since the 1950s, producing many interesting applications over the years, [1] in particular in games (e.g., *TD-Gammon*, a *Backgammon* playing program) and in machine control, but seldom making the headline news. But a revolution took place in 2013 when researchers from an English startup called DeepMind demonstrated a system that could learn to play just about any Atari game from scratch, [2] eventually outperforming humans [3] in most of them, using only raw pixels as inputs and without any prior knowledge of the rules of the games. [4] This was the first of a series of amazing feats, culminating in May 2017 with the victory of their system AlphaGo against Ke Jie, the world champion of the game of *Go*. No program had ever come close to beating a master of this game, let alone the world champion. Today the whole field of RL is boiling with new ideas, with a wide range of applications. DeepMind was bought by Google for over 500 million dollars in 2014.

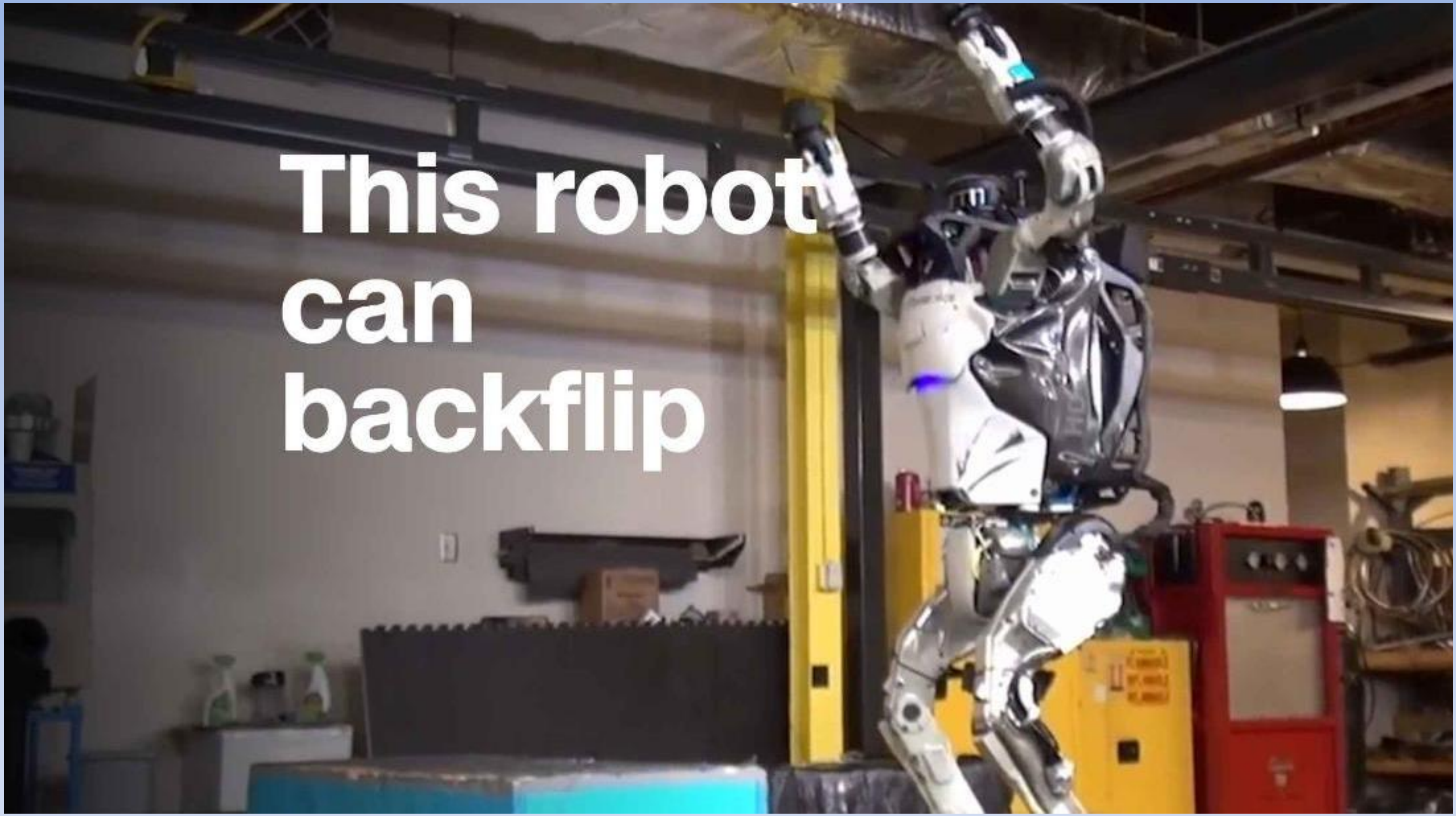
CS 188: Artificial Intelligence

Reinforcement Learning

Instructors: Dan Klein and Pieter Abbeel

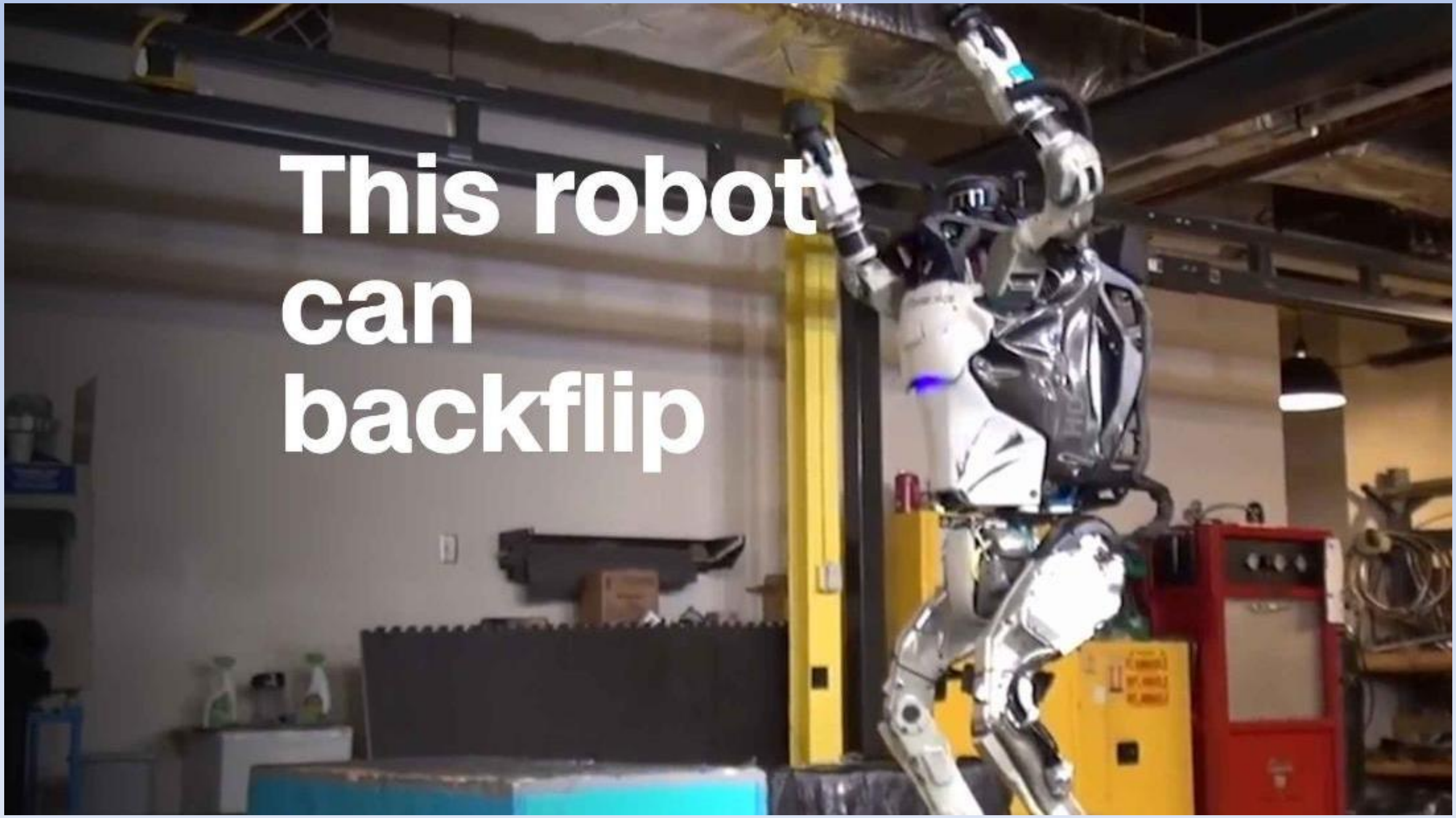
University of California, Berkeley

[These slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.]


This robot
can
backflip

https://youtu.be/fRj34o4hN4I

- https://youtu.be/fRj34o4hN4I

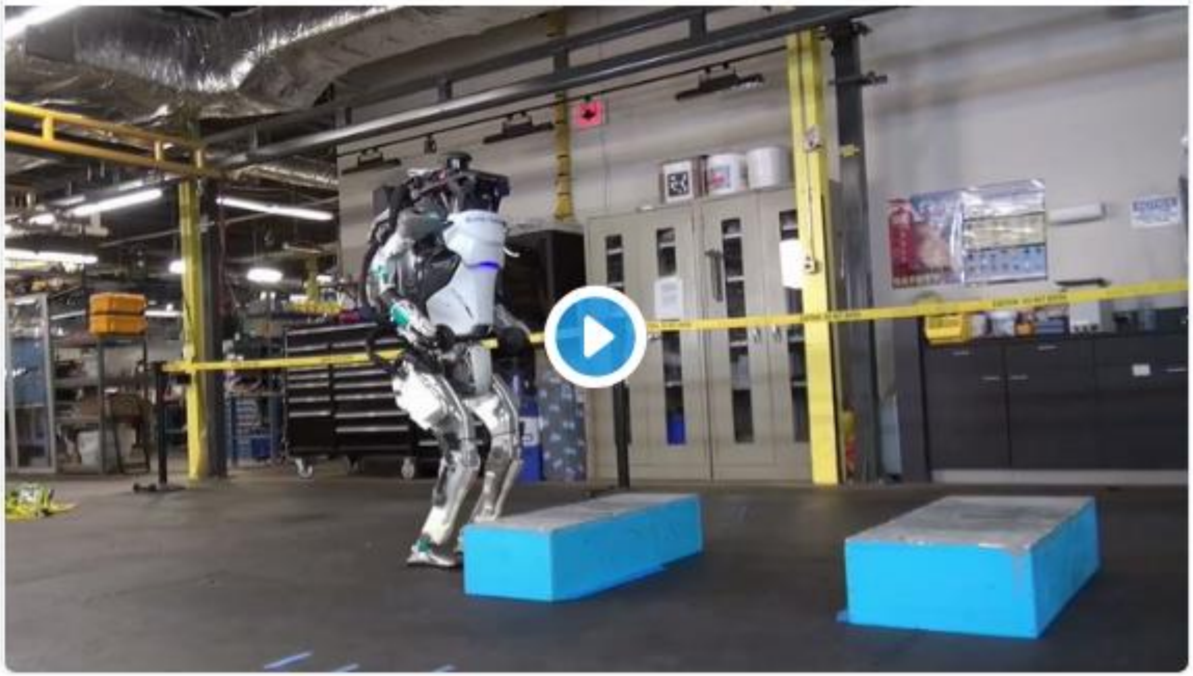


Elon Musk @elonmusk

This is nothing. In a few years, that bot will move so fast you'll need a strobe light to see it. Sweet dreams…

alex medina @mrmedina

we dead



1:54 PM - Nov 26, 2017

275K 96.8K people are talking about this


a

Learning to Walk

- Robocup -- Robot Soccer

Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih **Koray Kavukcuoglu** **David Silver** **Alex Graves** **Ioannis Antonoglou**

Daan Wierstra **Martin Riedmiller**

DeepMind Technologies

{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com

Abstract

We present the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards. We apply our method to seven Atari 2600 games from the Arcade Learning Environment, with no adjustment of the architecture or learning algorithm. We find that it outperforms all previous approaches on six of the games and surpasses a human expert on three of them.

Playing Atari with Deep Reinforcement Learning

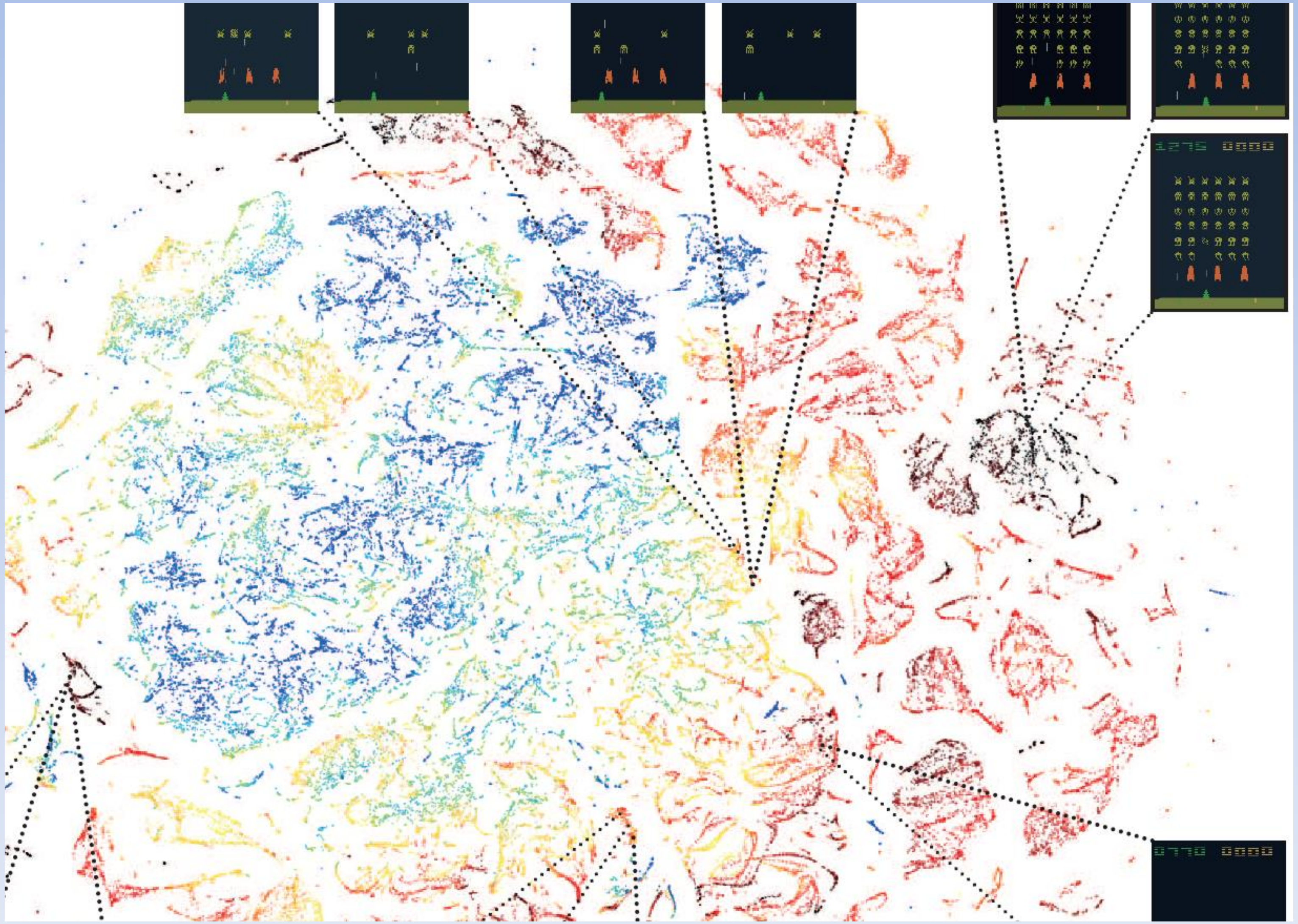
Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller

(Submitted on 19 Dec 2013)

We present the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards. We apply our method to seven Atari 2600 games from the Arcade Learning Environment, with no adjustment of the architecture or learning algorithm. We find that it outperforms all previous approaches on six of the games and surpasses a human expert on three of them.

Comments: NIPS Deep Learning Workshop 2013
Subjects: **Machine Learning (cs.LG)**
Cite as: **arXiv:1312.5602 [cs.LG]**
(or **arXiv:1312.5602v1 [cs.LG]** for this version)

DeepMind – Atari Games w/ RL



DeepMind Video

- https://youtu.be/xN1d3qHMIEQ

The company

DeepMind was founded in London in 2010 and backed by some of the most successful technology entrepreneurs in the world. Having been acquired by Google in 2014, we are now part of the Alphabet group. We continue to be based in our hometown of London, with additional research centres in Edmonton and Montreal, Canada, and a DeepMind Applied team in Mountain View, California.

https://deepmind.com/about/

Co-Founder & CEO, DeepMind

Demis is a former child chess prodigy, who finished his A-levels two years early before coding the multi-million selling simulation game Theme Park aged 17. Following graduation from Cambridge University with a Double First in Computer Science he founded the pioneering videogames company Elixir Studios producing award winning games for global publishers such as Vivendi Universal. After a decade of experience leading successful technology startups, Demis returned to academia to complete a PhD in cognitive neuroscience at UCL, followed by postdocs at MIT and Harvard, before founding DeepMind. His research into the neural mechanisms underlying imagination and planning was listed in the top ten scientific breakthroughs of 2007 by the journal Science. Demis is a 5-times World Games Champion, a Fellow of the Royal Society of Arts, and the recipient of the Royal Society's Mullard Award and the Royal Academy of Engineering's Silver Medal.

Dr Demis Hassabis

https://deepmind.com/about/

Mustafa Suleyman

Co-Founder & Head of Applied AI

Mustafa Suleyman is co-founder and Head of Applied AI at DeepMind, where he is responsible for the application of DeepMind's technology to real-world problems, as part of DeepMind's commitment to use intelligence to make the world a better place. In February 2016 he launched DeepMind Health, which builds clinician-led technology in the NHS. Mustafa was Chief Product Officer before DeepMind was bought in 2014 by Google in their largest European acquisition to date. At 19, Mustafa dropped out of Oxford University to help set up a telephone counselling service, building it to become one of the largest mental health support services of its kind in the UK, and then worked as policy officer for then Mayor of London, Ken Livingstone. He went on to help start Reos Partners, a consultancy with seven offices across four continents specializing in designing and facilitating large-scale multi-stakeholder 'Change Labs' aimed at navigating complex problems. As a skilled negotiator and facilitator Mustafa has worked across the world for a wide range of clients such as the UN, the Dutch Government and WWF.

https://deepmind.com/about/

Co-Founder & Chief Scientist, DeepMind

Shane obtained his PhD from IDSIA in Switzerland, supervised by Prof. Marcus Hutter, the leading authority on theoretical models of super intelligent machines. His thesis proposed a formal definition of machine intelligence, for which he was awarded the $10,000 Canadian Singularity Institute research prize. He spent a post doctoral year at the Swiss Finance Institute building models of human decision making, followed by two years at the Gatsby Computational Neuroscience Unit at UCL studying the algorithmic organisation of the brain. In 2010 he co-founded DeepMind Technologies with Demis Hassabis and Mustafa Suleyman. After three years of rapid growth and a number of research breakthroughs, DeepMind was acquired by Google.

@ShaneLegg

Dr Shane Legg

Reinforcement Learning

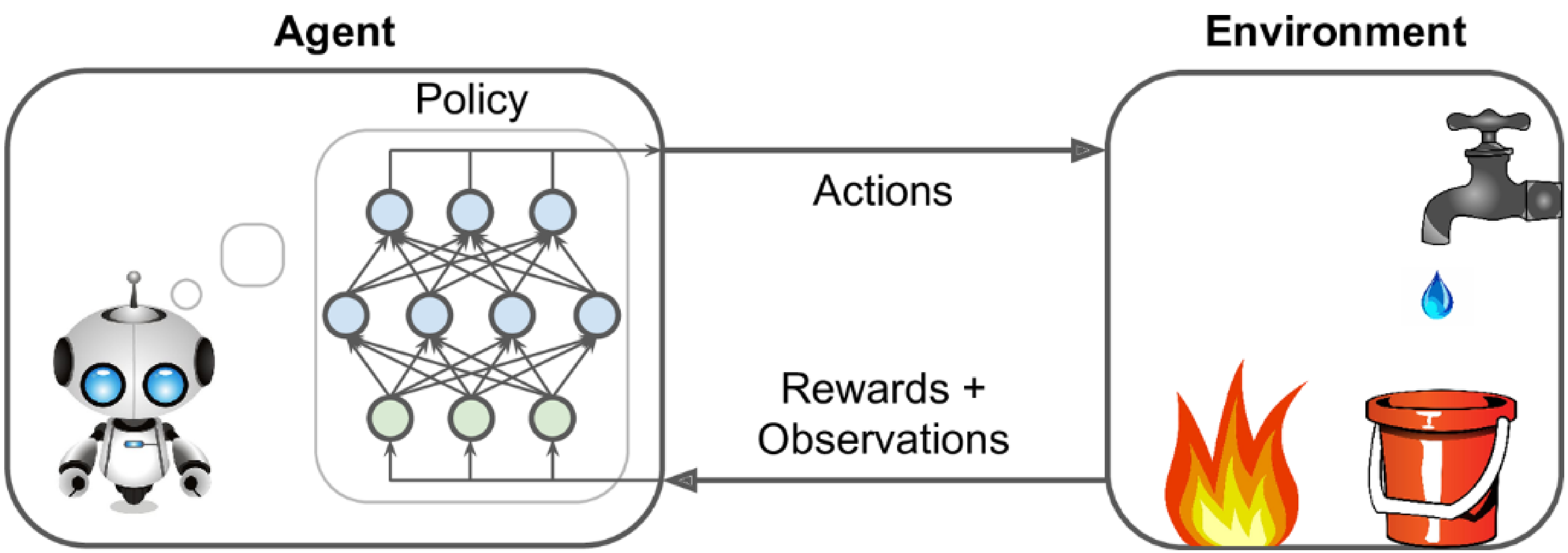
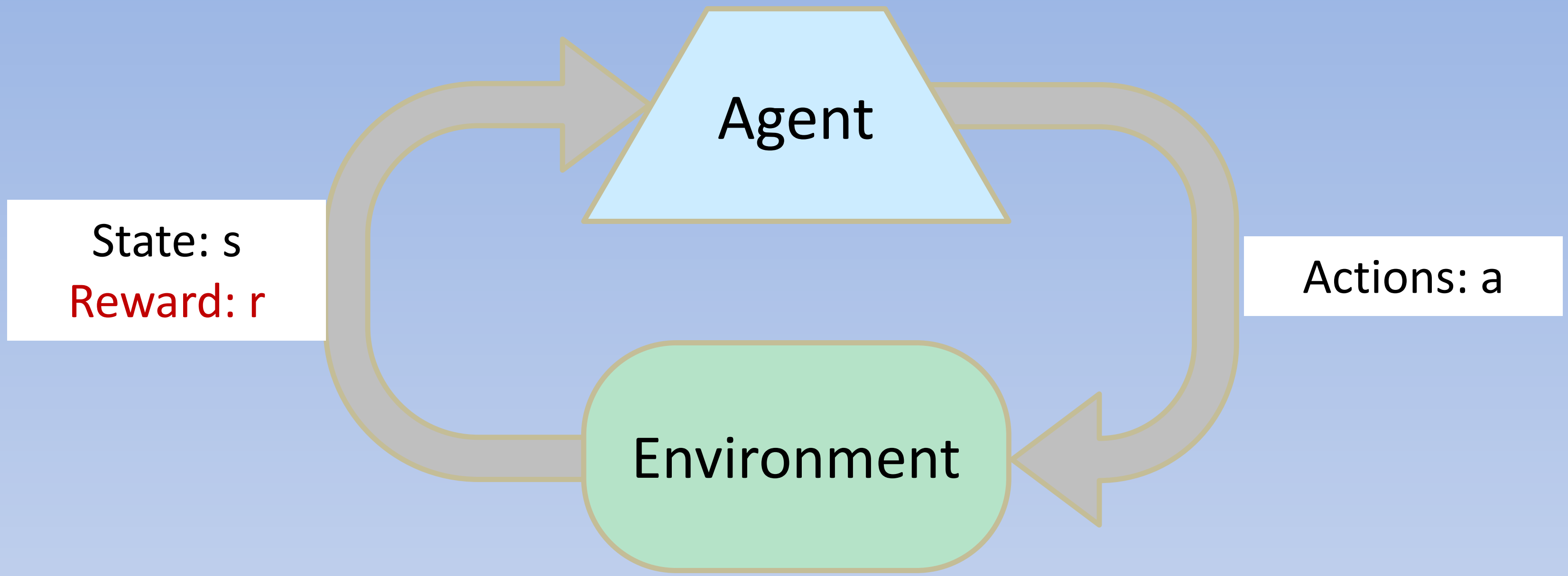


Figure 16-2. Reinforcement Learning using a neural network policy



- Basic idea:
  - Receive feedback in the form of rewards
  - Agent’s utility is defined by the reward function
  - Must (learn to) act so as to maximize expected rewards
  - All learning is based on observed samples of outcomes!

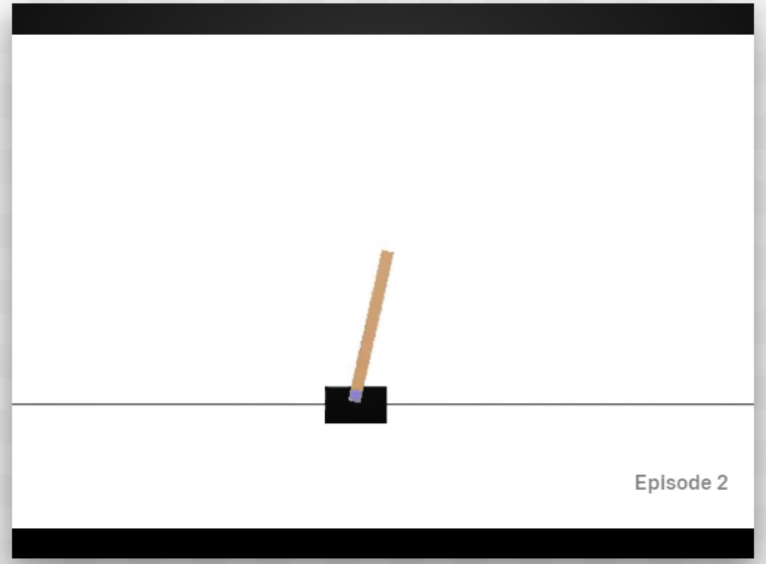
Gym

Gym is a toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like Pong or Pinball.

View documentation ›

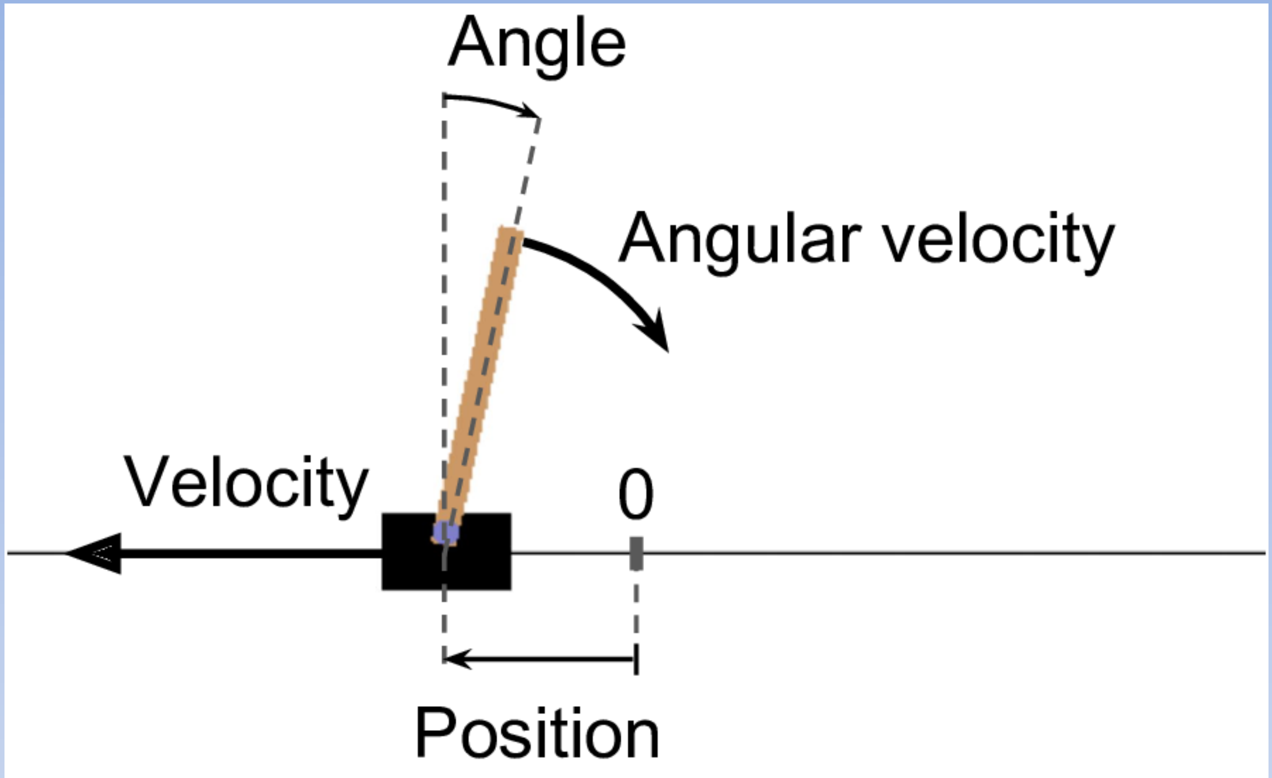
View on GitHub ›

RandomAgent on Ant-v2



RandomAgent on CartPole-v1

Cart-Pole



- Balance The Pole
- Moves: Left, Right
- State:
  - Cart horizontal position
  - Cart velocity
  - Angle of the pole (0 = vertical)
  - Angular velocity.

Cart-Pole

In [2]:
```
import gym
```

Cart-Pole

In [3]:
```
env = gym.make("CartPole-v0")
obs = env.reset()
print(obs)
```

```
WARN: gym.spaces.Box autodetected dtype as <type 'numpy.float32'>. Please provide explicit dtype.
[ 0.03201594 -0.02388707  0.0477543  -0.04951689]
```

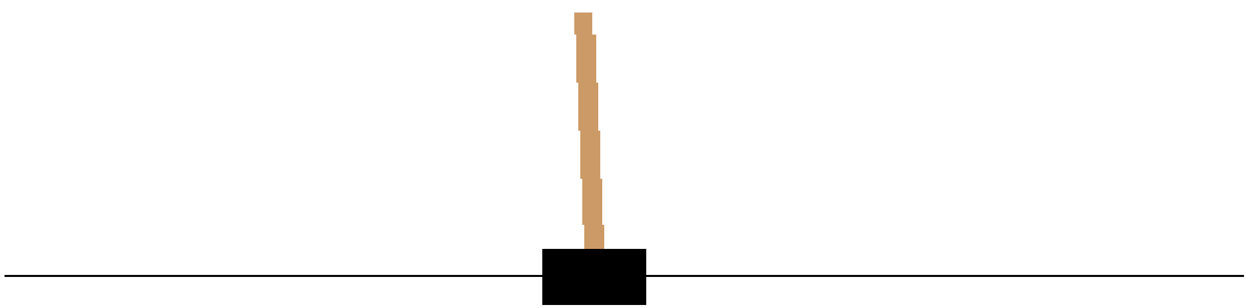
Bit.ly/RubyBoard

- http://bit.ly/c166f20rl

```
WARN: gym.spaces.Box autodetected dtype as <type 'numpy.float32'>. Please provide explicit dtype.
[-0.048696    0.01398668 -0.03688122  0.01554001]
```

Figure 1

```
In [7]: env = gym.make("CartPole-v0")
        obs = env.reset()
        print(obs)
        plot_cart_pole(env, obs)
```



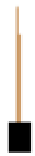
Environment Test

```
[68]: obs = env.reset()
      print(obs)
      plot_cart_pole(env, obs)

      obs = env.reset()
      while True:
          obs, reward, done, info = env.step(1)  #right
          plot_cart_pole(env, obs)
          print(obs);print(reward);print(done)
          if done:
              break
```

```
[-0.02885786  0.16162679  0.02435226 -0.24038935]
1.0
False
```

```
[-0.02562533  0.35639256  0.01954448 -0.52529252]
1.0
False
```

```
[-0.01849748  0.5512341   0.00903863 -0.81175331]
1.0
False
```

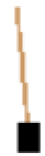


```
[-0.00747279  0.74623108 -0.00719644 -1.1015795 ]
1.0
False
```

```
[ 0.00745183  0.94144697 -0.02922803 -1.3965115 ]
1.0
False
```

```
[ 0.02628077  1.13692004 -0.05715826 -1.69818768]
1.0
False
```

How should we learn?



```
[ 0.04901917  1.33265247 -0.09112201 -2.00810224]
1.0
False
```

```
[ 0.07567222  1.52859732 -0.13128406 -2.32755294]
1.0
False
```

```
[ 0.10624416  1.7246428  -0.17783512 -2.65757732]
1.0
False
```

```
[ 0.14073702  1.92059414 -0.23098666 -2.99887718]
1.0
True
```

First Idea

- Hand encode some rules:
  - Accelerate left when pole is leaning left
  - Accelerate right when pole is leaning right

- Let’s Test It….
  - We’ll throw our agent into environment as see what happens…

Testing Agent In Environment

- Back to OpenAi Gym

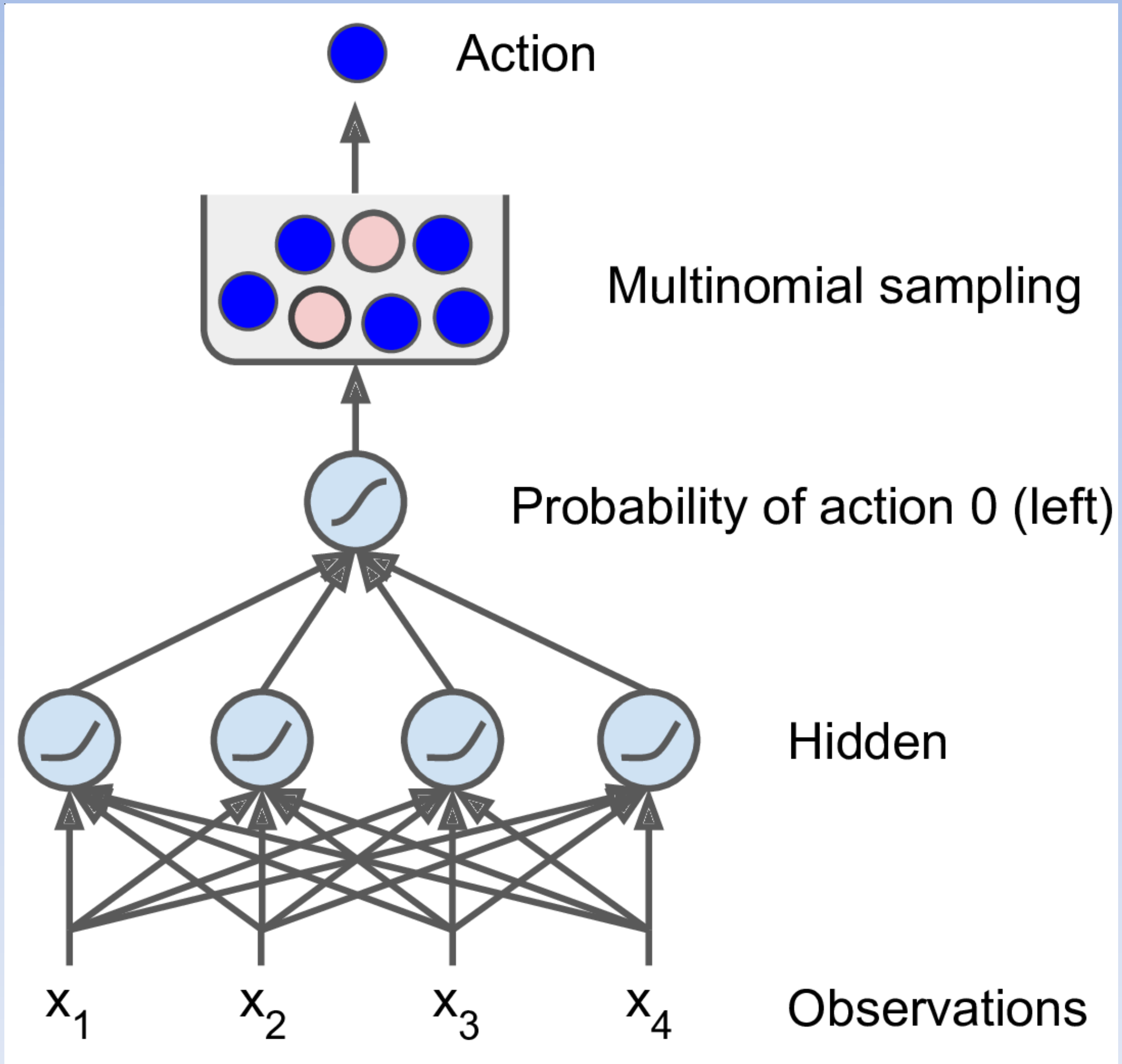
```
In [13]: def basic_policy(obs):
             angle = obs[2]
             return 0 if angle < 0 else 1

         totals = []
         for episode in range(500):
             episode_rewards = 0
             obs = env.reset()
             for step in range(1000): # 1000 steps max, we don't want to run forever
                 action = basic_policy(obs)
                 obs, reward, done, info = env.step(action)
                 episode_rewards += reward
                 if done:
                     break
             totals.append(episode_rewards)
```

Results

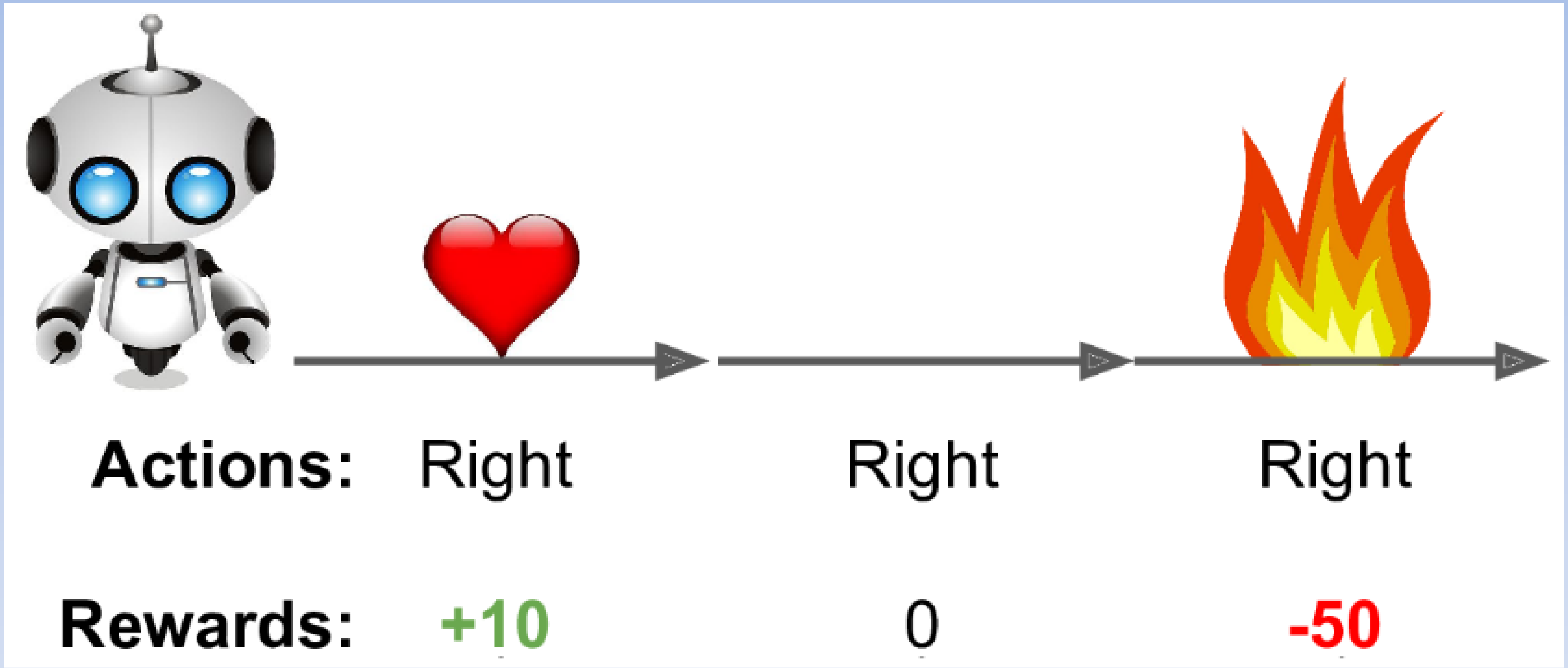
- np.mean(totals)= 42.125999999999998
- np.std(totals) = 9.1237121830974033,
- np.min(totals) = 24.0
- np.max(totals) = 68.0

Maybe Learning?

- How do we know which moves are good/bad ?



Credit Assignment Problem



Step Back

- Agent learns to estimate the expected sum of discounted future rewards for each state
- Agent learns to estimate the expected sum of discounted future rewards for each action in each state
- Then agent uses this knowledge to decide how to act.
- To understand these algorithms, we must first introduce Markov decision processes (MDP).

Reinforcement Learning to MDP

- Assume a Markov decision process (MDP):
  - A set of states $s \in S$
  - A set of actions (per state) A
  - A model $P(s'|s,a)$ or $T(s,a,s')$
  - A reward function $R(s)$ or $R(s,a,s')$

Model

518 18 *Reinforcement Learning*

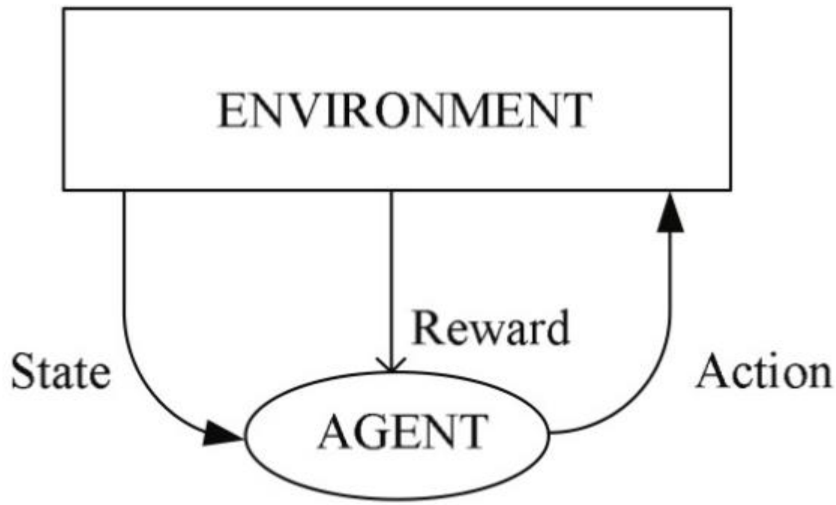
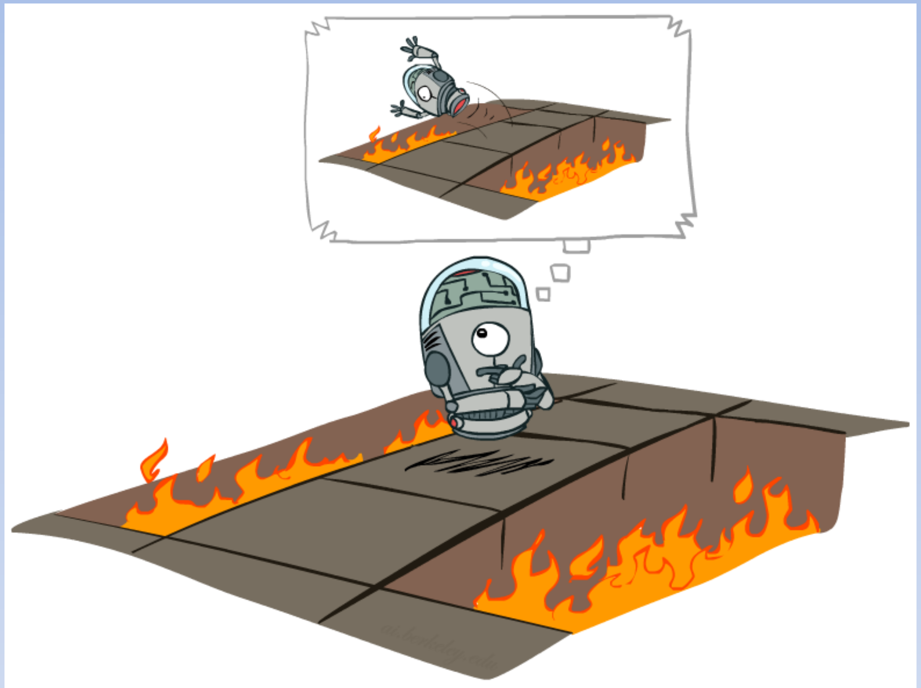
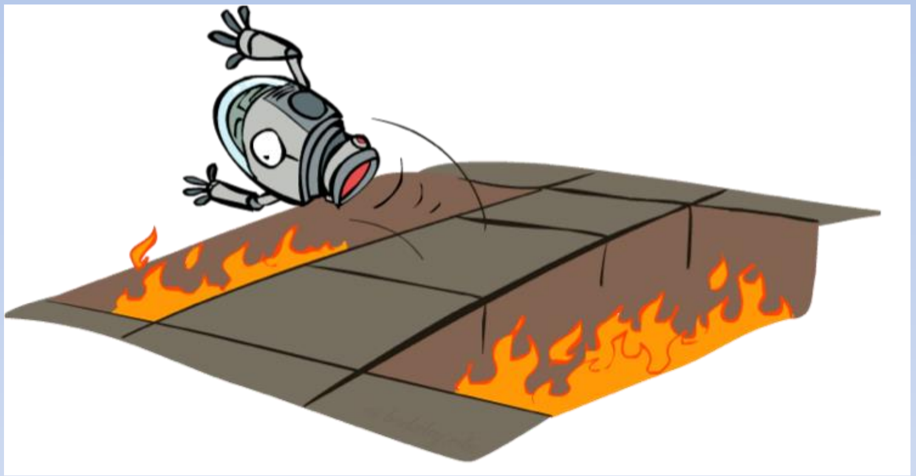


Figure 18.1 The agent interacts with an environment. At any state of the environment, the agent takes an action that changes the state and returns a reward.

Offline (MDPs) vs. Online (RL)



Offline Solution



Online Learning