Practical Machine Learning

Reinforcement Learning

https://bit.ly/2USYqjZ

Today we cover

Reinforcement learning

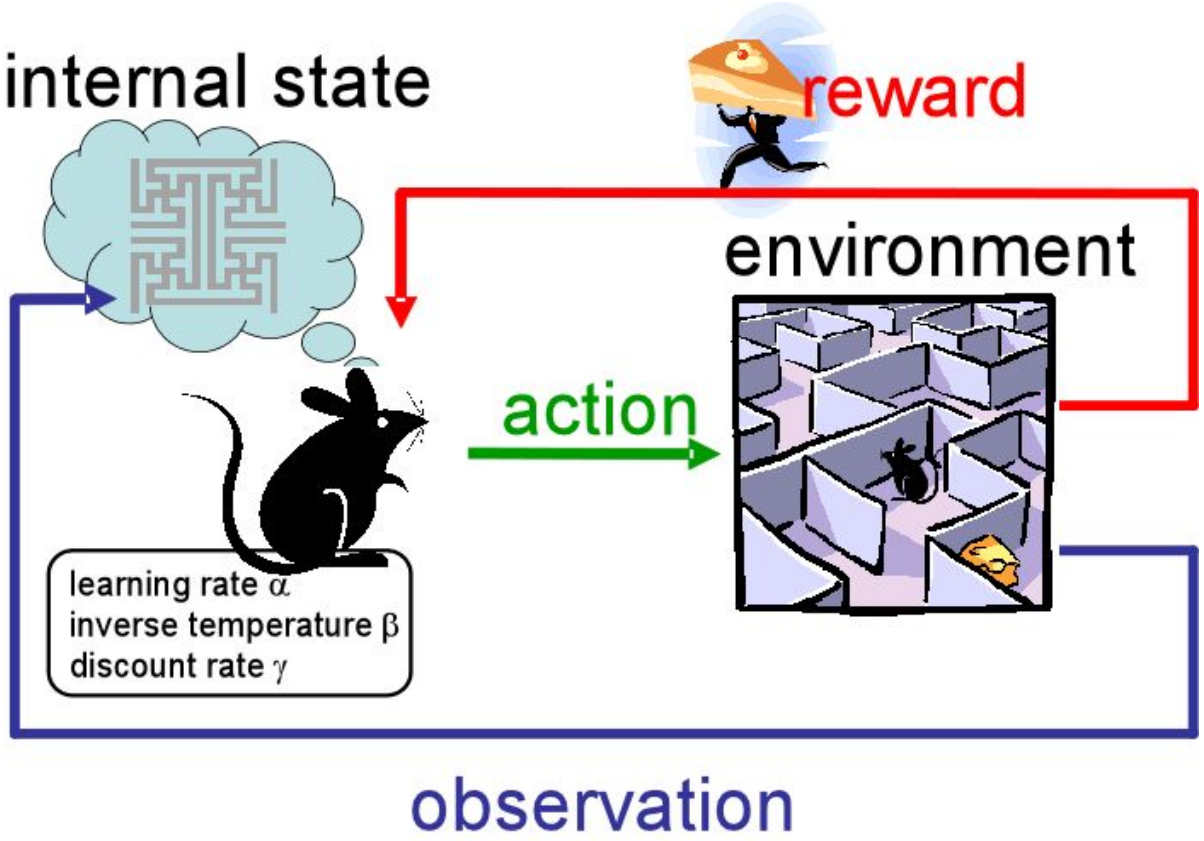
Open AI Gym

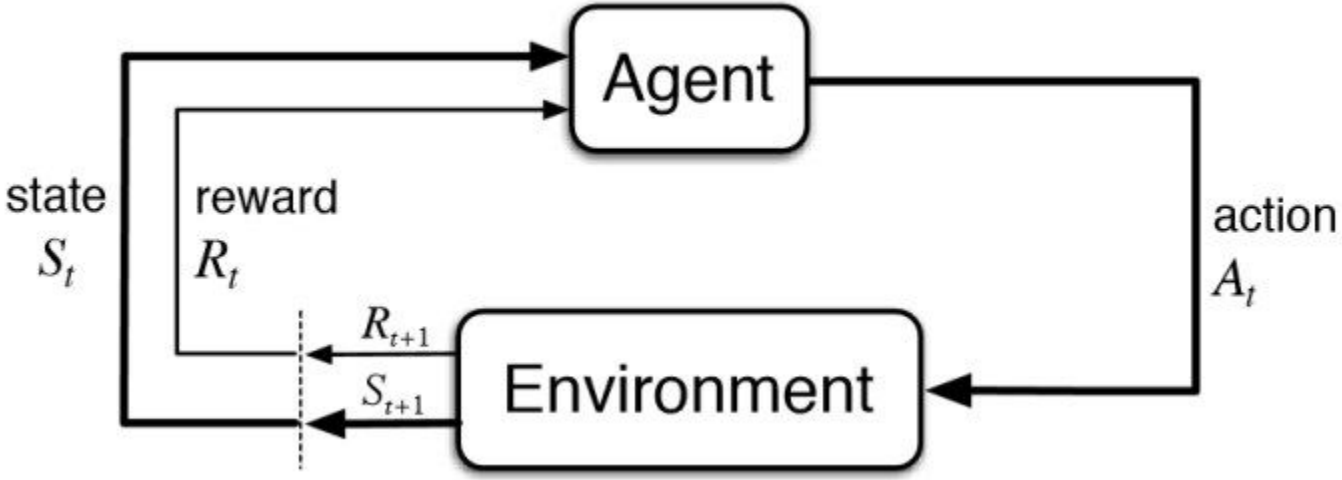
Building an agent

Reinforcement Learning

Learning to act in complex environments

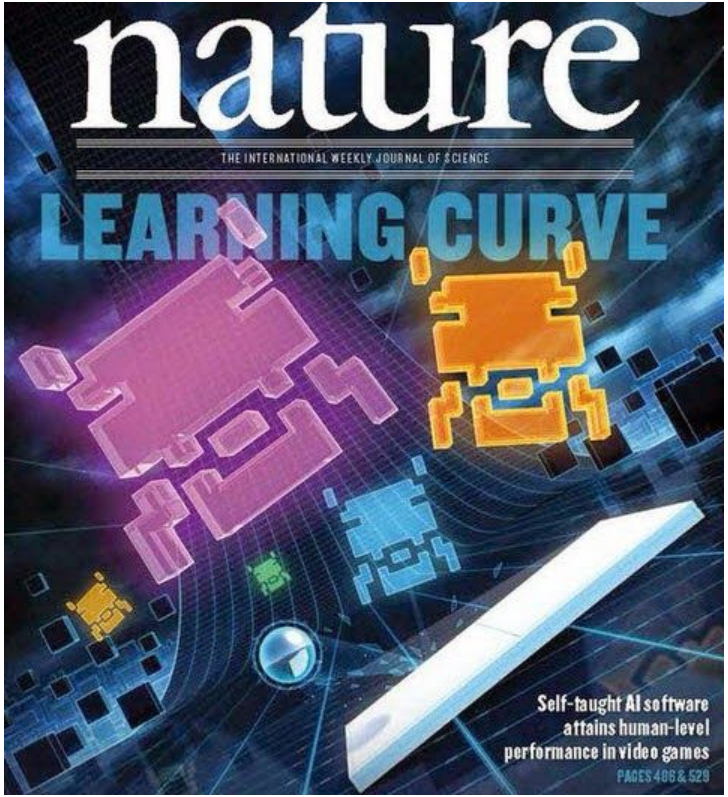
Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward.


internal state
reward
environment
action
learning rate α
inverse temperature β
discount rate γ
observation


Agent
state
S_t
reward
R_t
action
A_t
R_{t+1}
S_{t+1}
Environment

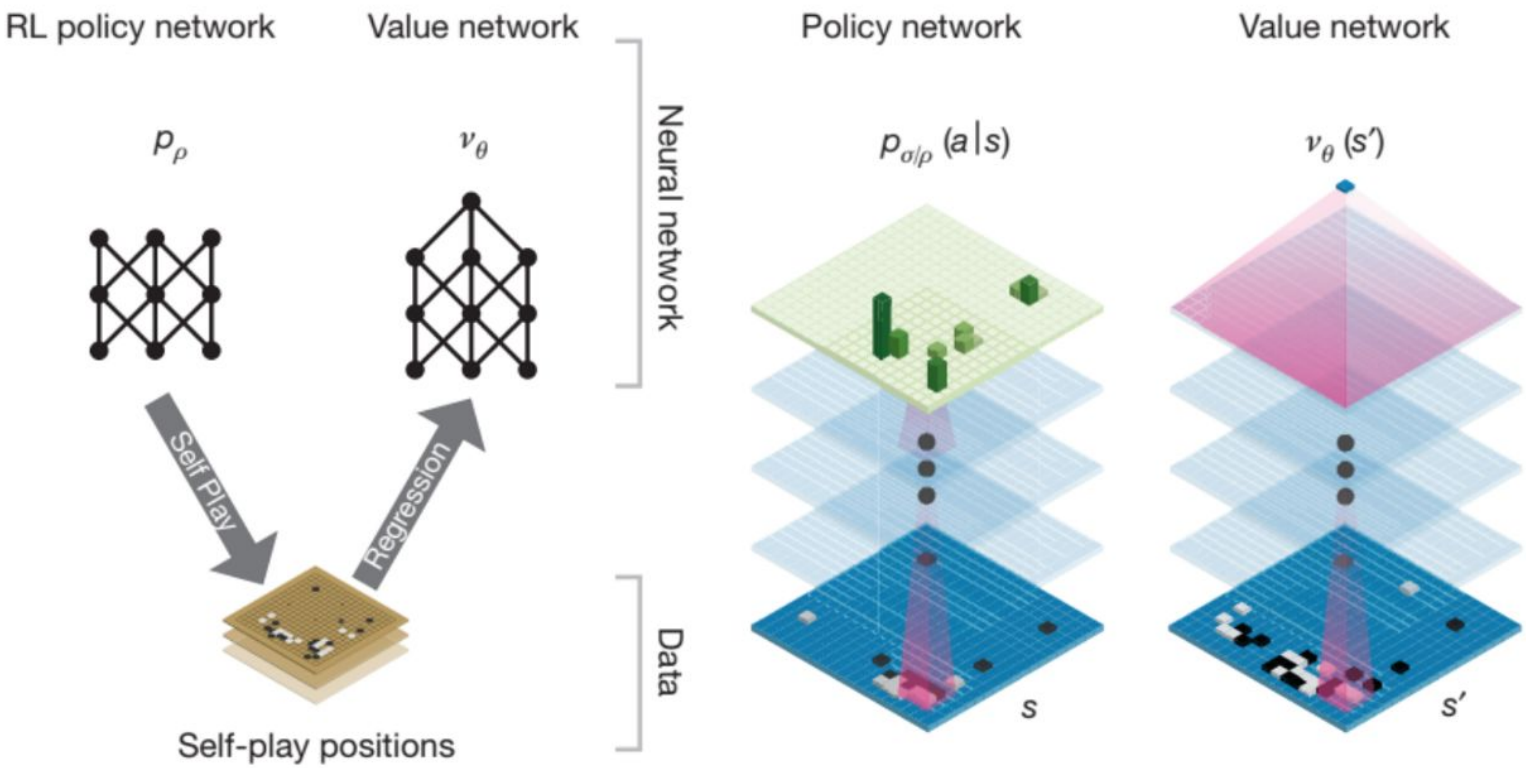
Why is it difficult?

- No immediate label
- Reward could be from a combination of actions
- I need to explore the space
- I need to find what was useful from my actions


nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
LEARNING CURVE
Self-taught AI software attains human-level performance in video games
PAGES 406 & 529


180 2 1


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

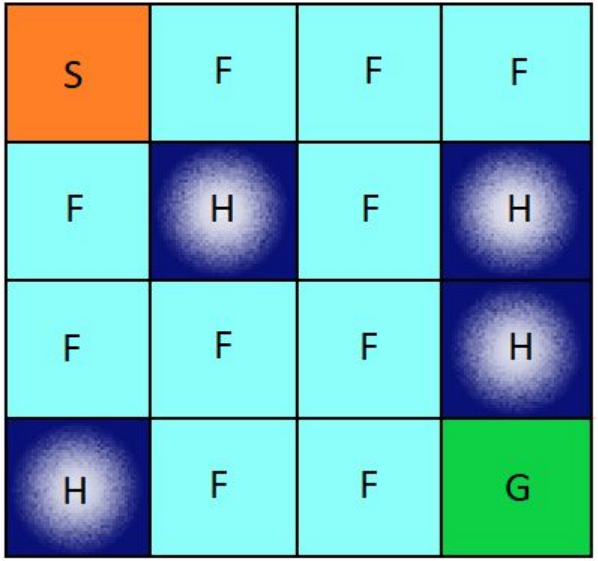

RL policy network
Value network
p_ρ
v_θ
Self Play
Regression
Self-play positions
Neural network
Data
Policy network
Value network
$p_{\sigma/\rho}(a|s)$
$v_\theta(s')$
s
s'


Reinforcement Learning
First trial...

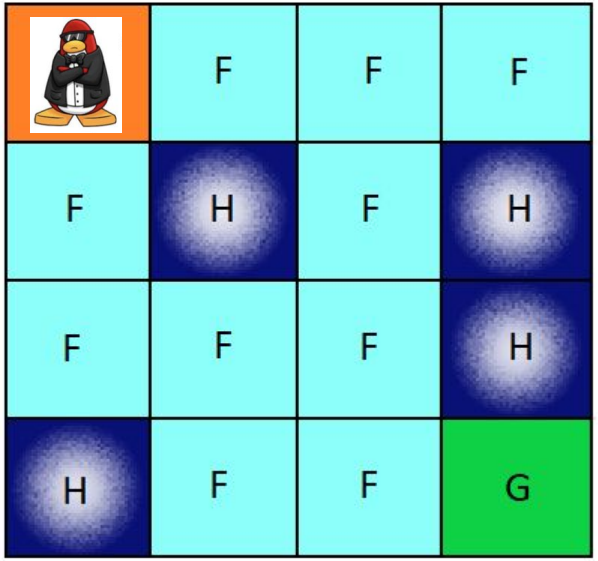
Key concepts

- Environment
- Agent
- State / observation
- Action
- Reward
- Discount factor
- Policy

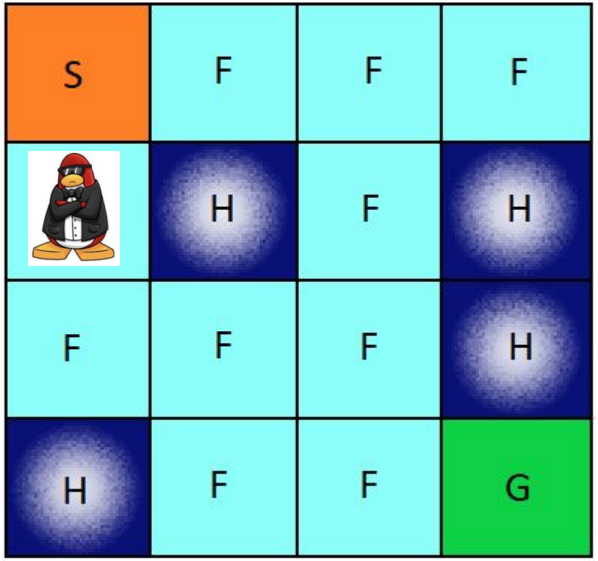
Environment



Agent

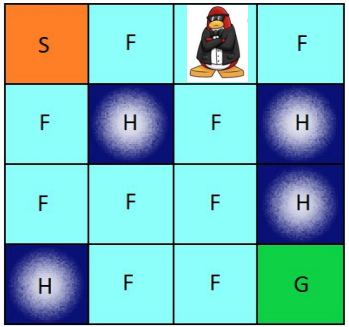


Action

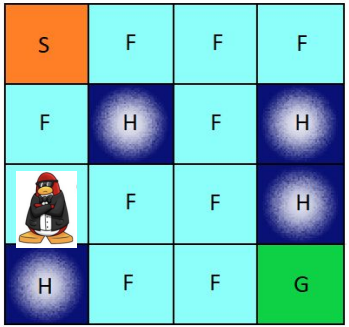


Left / Right / Up / Down

State

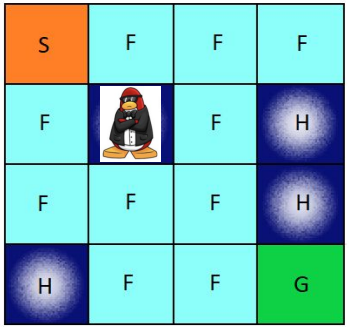


State = 2

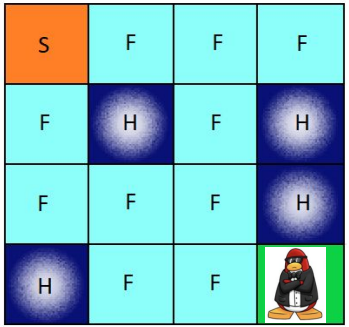


State = 8

Reward



0 if falls



1 if goal reached

Return

$$R = \sum_{t=0}^{\infty} \gamma^t r_t,$$

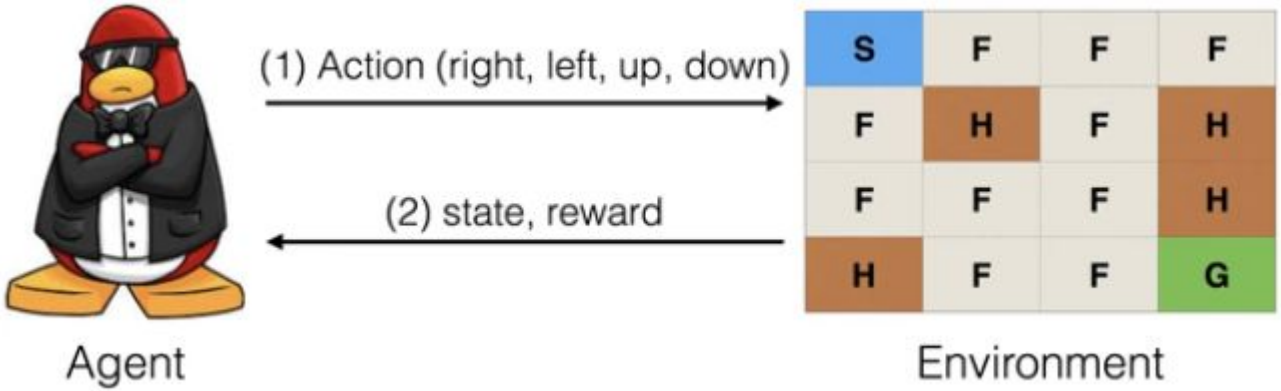
Reward at time t

Discount rate modulated by time

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

Discount rate expresses my preference for immediate rewards.

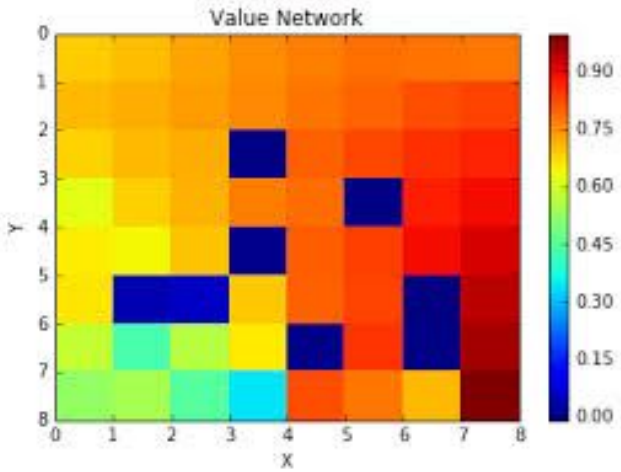
Policy



$$\pi : S \times A \rightarrow [0, 1]$$

$$\pi(a|s) = Pr(a_t = a|s_t = s)$$

Value of a policy



$$V_\pi(s) = E[R] = E[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s],$$

Optimal policy

The optimal policy guarantees the best expected returns

Star indicates optimal

Select the best policy

Value of a given policy

$$V^*(s) = \max_{\pi} V^{\pi}(s).$$

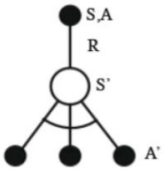
Reward at time t

Q-Value

$$Q : S \times A \to \mathbb{R}\,.$$

$$Q^{\pi}(s, a) = E[R|s, a, \pi],$$

Q-learning (SARSA Max)



Learning rate

Starting state

$$Q(S,A) = Q(S,A) + \alpha[R + \gamma \underset{a}{argmax}\, Q(s', a) - Q(S,A)]$$

Action taken

Next state

Algorithm:

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(terminal\text{-}state, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$;
  until S is terminal

Code

- Gym API
  - Env
  - Step
  - Observation
  - Reward
  - IsDone

```
[ ]  !pip install gym
```

```
[ ]  import gym
```

```
env = gym.make('FrozenLake-v0')
obs = env.reset()
for _ in range(10):
    env.render()
    obs, rewaerd, is_done, info = env.step(0)
env.close()
```