An Introduction to Deep Reinforcement Learning

Ehsan Abbasnejad

THE UNIVERSITY of ADELAIDE

Remember:
Supervised Learning

We have a set of sample observations, with **labels**

learn to predict the labels, given a new sample

cat

dog

Learn the function that associates a picture of a dog/cat with the label

Remember: supervised learning

We need thousands of samples

Samples have to be provided by experts

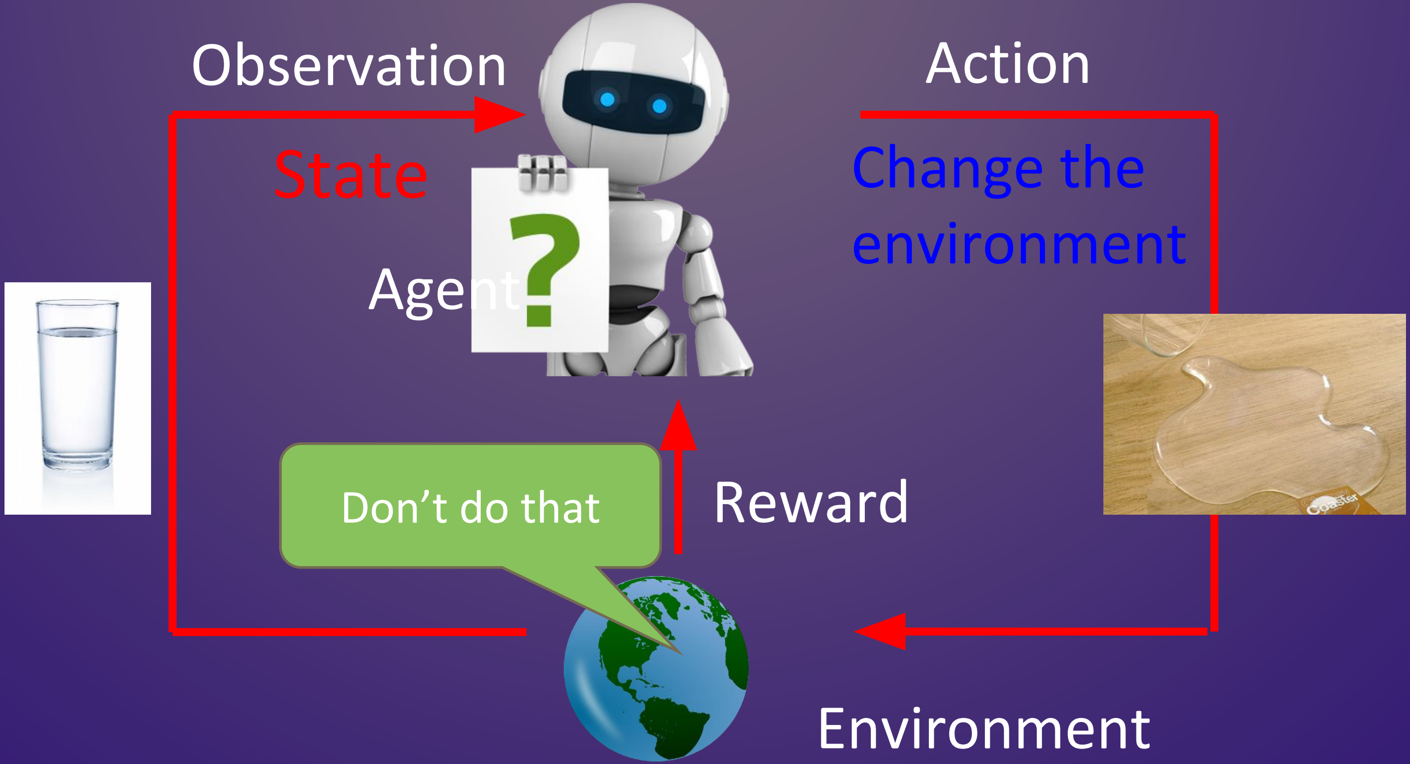
There are applications where

- We can't provide expert samples
- Expert examples are not what we mimic
- There is an interaction with the world

Deep Reinforcement Learning

AlphaGo

Scenario of Reinforcement Learning



Scenario of Reinforcem

Agent learns to take actions maximizing expected reward.

Observation

Action

State

Change the environment

Agent

Thank you.

Reward

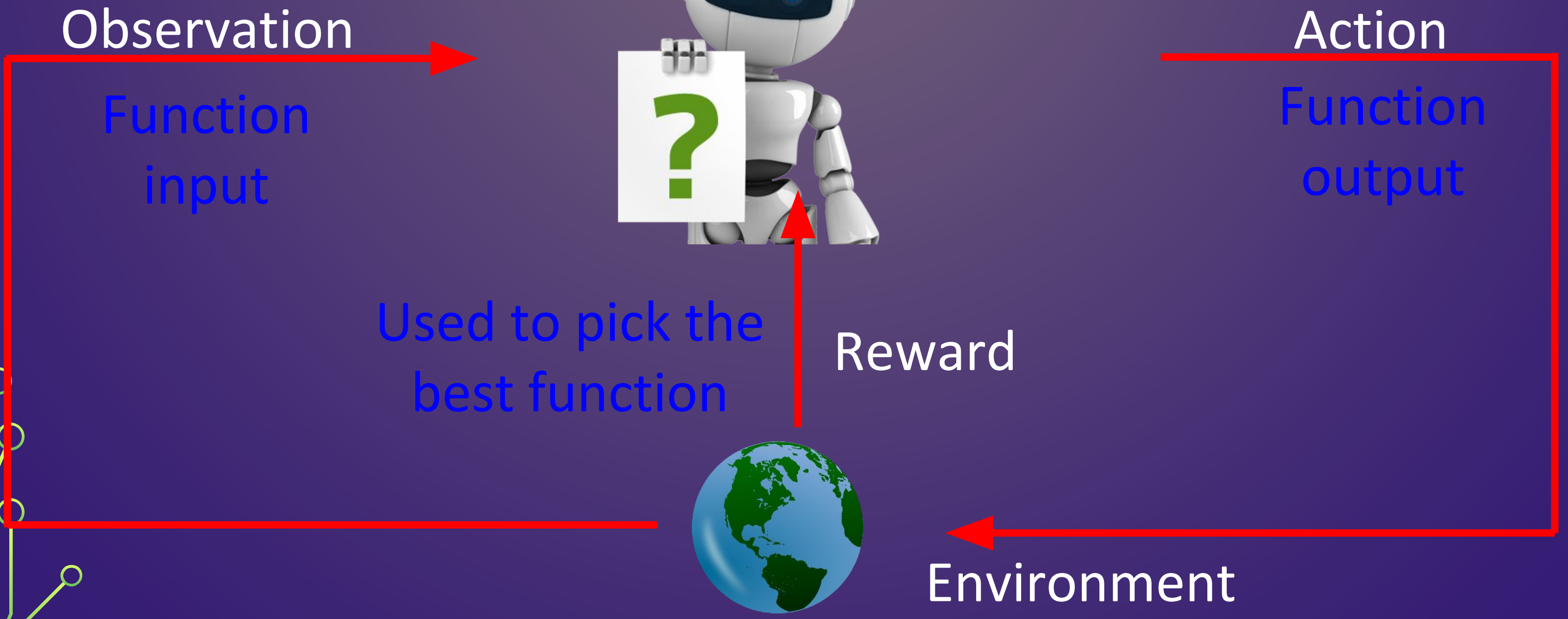
Environment

https://yoast.com/how-to-clean-site-structure/

Machine Learning
≈ Looking for a Function

Actor/Policy

Action = π(Observation)



Reinforcement Learning in a nutshell

RL is a general-purpose framework for decision-making

- RL is for an agent with the capacity to act
- Each action influences the agent's future state
- Success is measured by a scalar reward signal

Goal: select actions to maximise future reward

Deep Learning in a nutshell

DL is a general-purpose framework for representation learning

- Given an objective
- Learning representation that is required to achieve objective
- Directly from raw inputs
- Using minimal domain knowledge

Goal: Learn the representation that achieves the objective

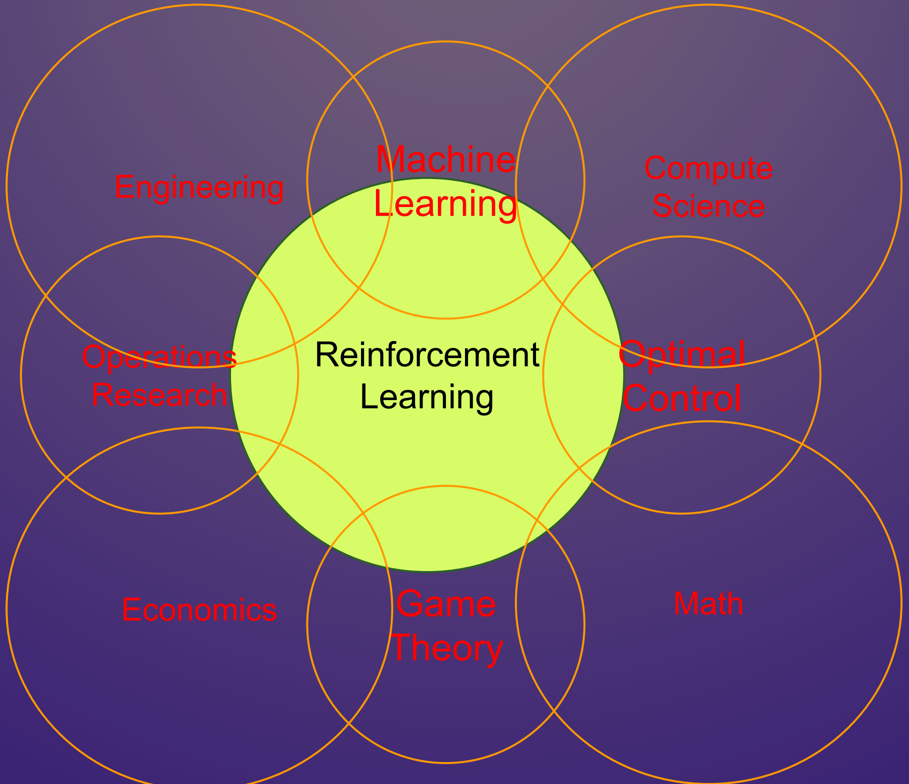
Deep Reinforcement Learning in a nutshell

A single agent that solves human level tasks

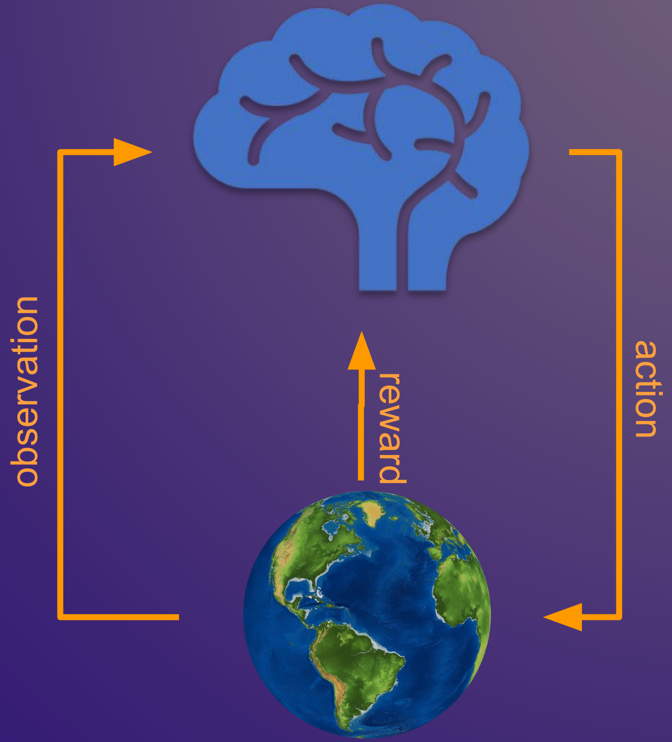
- RL defines the objective
- DL gives the mechanism and representation
- RL+DL=Deep reinforcement learning

This can lead to general intelligence

Reinforcement Learning is multi-disciplinary

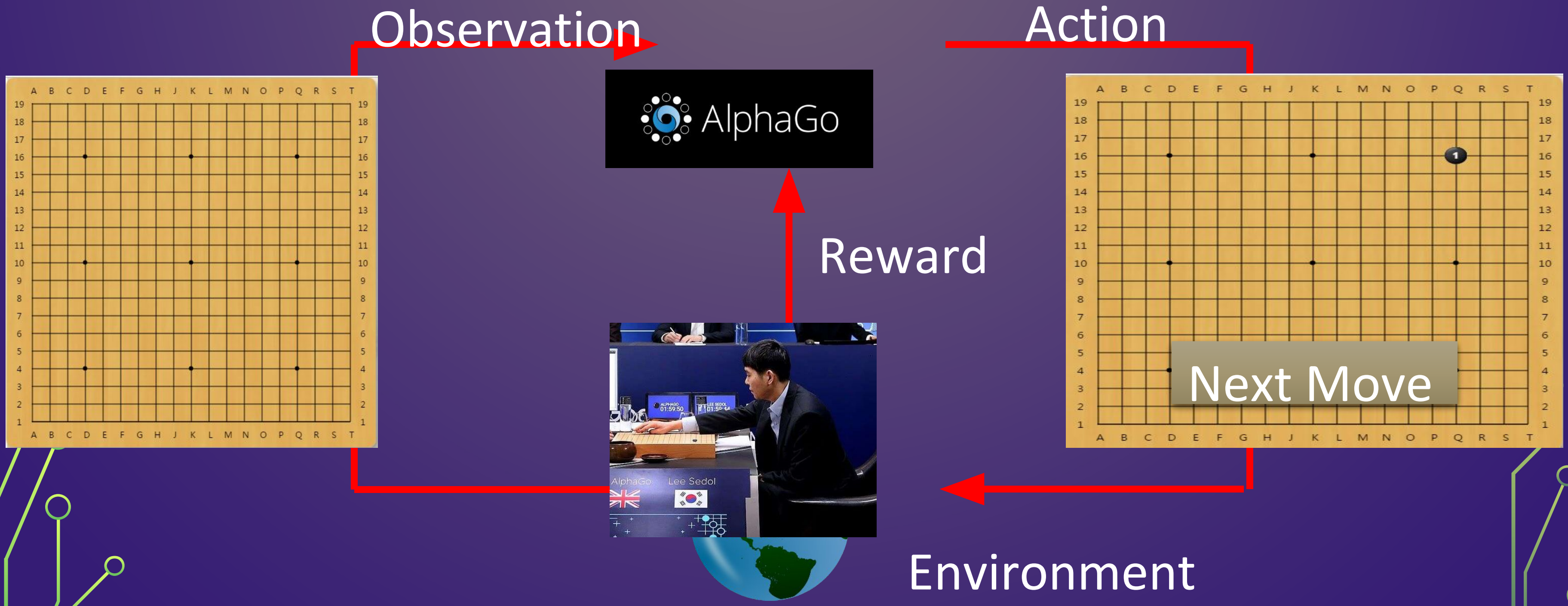


Agent and Environment



- At each step, the agent
    - Selects an action
    - Observes the environment
    - Receives reward
- The environment:
    - Receives action
    - Emits new observation
    - Emits reward for the agent

Learning to play Go



Learning to play Go

Agent learns to take actions maximizing expected reward.

Learning to play Go

- Supervised: **Learning from teacher**

Next move: "5-5"

Next move: "3-3"

- Reinforcement Learning **Learning from experience**

First move → …… many moves → Win!

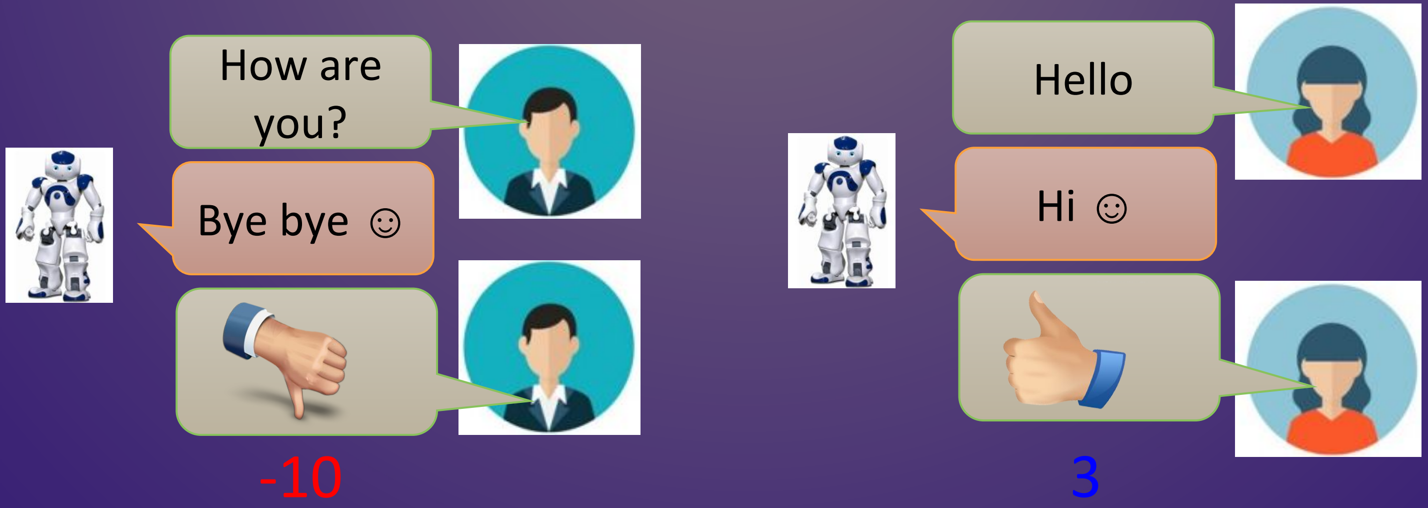
(Two agents play with each other.)

Alpha Go is supervised learning + reinforcement learning.

https://image.freepik.com/free-vector/variety-of-human-avatars_23-2147506285.jpg
http://www.freepik.com/free-vector/variety-of-human-avatars_766615.htm

Learning a chat-bot

- Machine obtains feedback from user



- Chat-bot learns to maximize the [*expected reward*]

Learning a chat-bot

- Let two agents talk to each other (sometimes generate good dialogue, sometimes bad)

How old are you?

See you.

See you.

See you.

How old are you?

I am 16.

I though you were 12.

What make you think so?

Learning a chat-bot

- By this approach, we can generate a lot of dialogues.
- Use some predefined rules to evaluate the goodness of a dialogue

Dialogue 1 | Dialogue 2 | Dialogue 3 | Dialogue 4

Dialogue 5 | Dialogue 6 | Dialogue 7 | Dialogue 8

Machine learns from the evaluation

Deep Reinforcement Learning for Dialogue Generation https://arxiv.org/pdf/1606.01541v3.pdf

Learning a chat-bot

- Supervised

"Hello" Say "Hi"

"Bye bye" Say "Good bye"

- Reinforcement

……. ……. …….

Hello ☺ ……

Agent Agent

Bad

More applications

- Flying Helicopter
  - https://www.youtube.com/watch?v=0JL04JJjocc
- Driving
  - https://www.youtube.com/watch?v=0xo1Ldx3L5Q
- Robot
  - https://www.youtube.com/watch?v=370cT-OAzzM
- Google Cuts Its Giant Electricity Bill With DeepMind-Powered AI
  - http://www.bloomberg.com/news/articles/2016-07-19/google-cuts-its-giant-electricity-bill-with-deepmind-powered-ai
- Text generation
  - https://www.youtube.com/watch?v=pbQ4qe8EwLo

Example: Playing Video Game

- Widely studies:
  - Gym: https://gym.openai.com/
  - Universe: https://openai.com/blog/universe/

Machine learns to play video games as human players

- ➢ What machine observes is pixels
- ➢ Machine learns to take proper action itself

Example: Playing Video Game

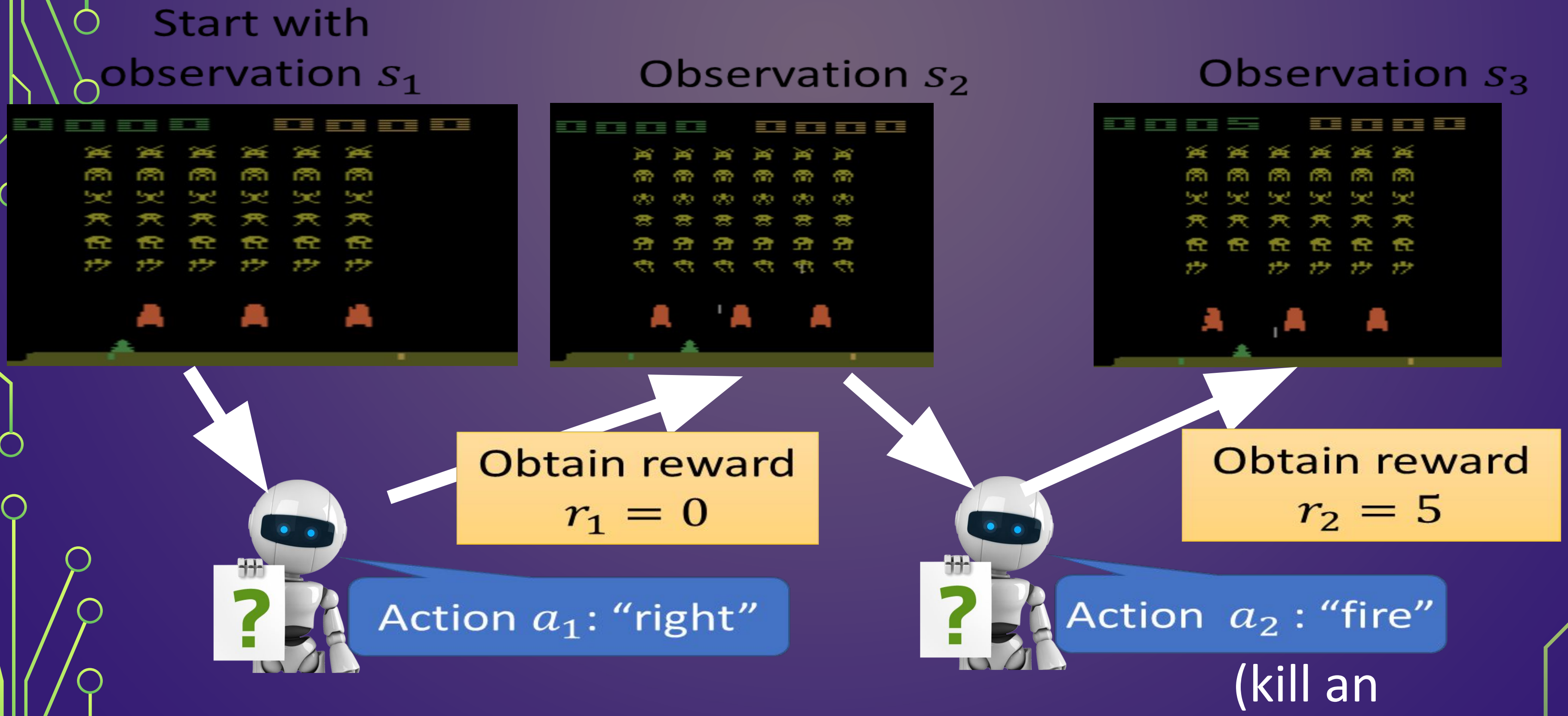
Termination: all the aliens are killed, or your spaceship is

- Space invader

Example: Playing Video Game

- Space invader
  - Play yourself: http://www.2600online.com/spaceinvaders.html
  - How about machine: https://gym.openai.com/evaluations/eval_Eduozx4HRyqgTCVk9ltw

Example: Playing Video Game

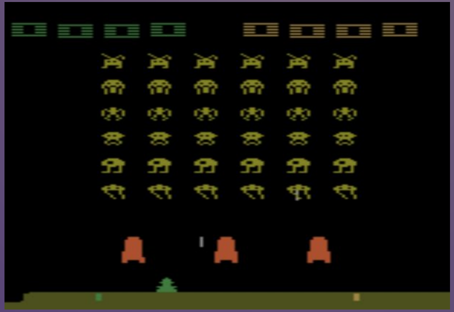


Usually there is some randomness in the environment

Example: Playing Video Game

Start with observation s_1

Observation s_2



Observation s_3

After many turns

Game Over
(spaceship destroyed)

Obtain reward r_T

Action a_T

This is an ***episode***.

Learn to maximize the expected cumulative reward per episode

Paradigm	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Objective	$p_\theta(y \mid x)$	$p_\theta(x)$	$\pi_\theta(a \mid s)$
Applications	$\rightarrow$ Classification $\rightarrow$ Regression	$\rightarrow$ Inference $\rightarrow$ Generation	$\rightarrow$ Prediction $\rightarrow$ Control

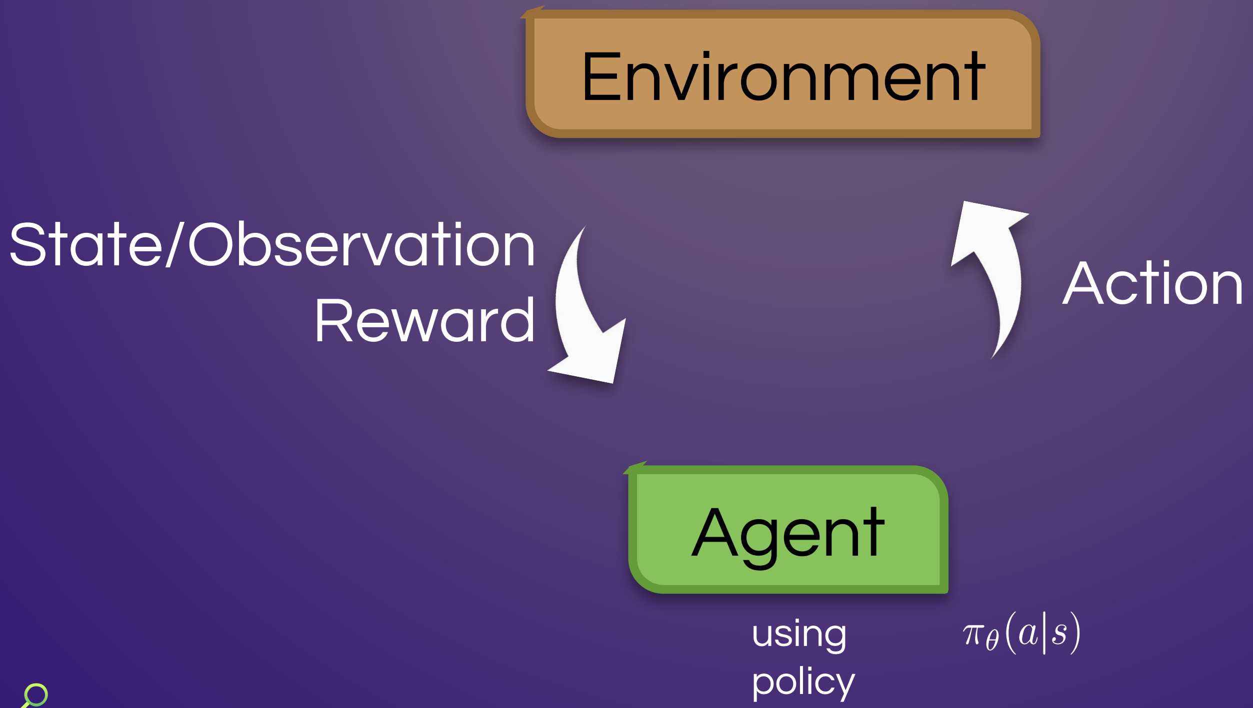
Prediction

$$x \rightarrow f_\theta \rightarrow y$$

<u>Control</u>

$$\begin{matrix} u \searrow & & \\ & f_\theta & \rightarrow y \\ x \nearrow & & \end{matrix}$$

SETTING


Environment
State/Observation
Reward
Action
Agent
using
policy
$\pi_\theta(a|s)$

MARKOV DECISION PROCESSES (MDP)

State space

Action space

Transition function

Reward function

- State: Markov property considers only the previous state
- Decision: agent takes actions, and those decisions have consequences
- Process: there is a transition function (dynamics of the system)
- Reward: depends on the state and action, often related to the state

Goal: maximise overall reward

Partially Observable MARKOV DECISION PROCESSES (POMDP)

State space

Action space

Transition function

Reward function

- State: Markov property considers only the previous state but the agent cannot directly observe the underlying state.
- Decision: agent takes actions, and those decisions have consequences
- Process: there is a transition function (dynamics of the system)
- Reward: depends on the state and action, often related to the state

Goal: maximise overall reward

MARKOV DECISION PROCESSES (MDP)

State space

$s_t \in \mathcal{S}$

Action space

$a_t \in \mathcal{A}$

Transition function

$\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$

$s_{t+1} \sim \mathcal{T}(\cdot | s_t, a_t)$

$s_0 \sim \mathcal{T}_0$

Reward function

$\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$

$r_t \sim \mathcal{R}(s_t, a_t)$

Computing Rewards

Episodic vs continuing: “Game over” after N steps

Additive rewards (can be infinite for continuing tasks)

Discounted rewards …

DISCOUNT FACTOR

→ We want to be greedy but not impulsive
→ Implicitly takes uncertainty in dynamics into account (we don't know the future)
→ Mathematically: $\gamma<1$ allows infinite horizon returns

Return: $$G(s_t, a_t) = \sum_{\tau=0}^{T} \gamma^{\tau} \mathcal{R}(s_{t+\tau}, a_{t+\tau})$$

SOLVING AN MDP

Objective:

$$J(\pi) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t), s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t), s_0 \sim \mathcal{T}_0} \left[\sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t) \right]$$

Goal:

$$\hat{\pi} = \arg\max_{\pi} J(\pi)$$

SOLVING AN MDP

- If the state and actions are discrete:
  - We have a table of state-action probabilities
  - Learning is filling this table: (dynamic programming)

Action

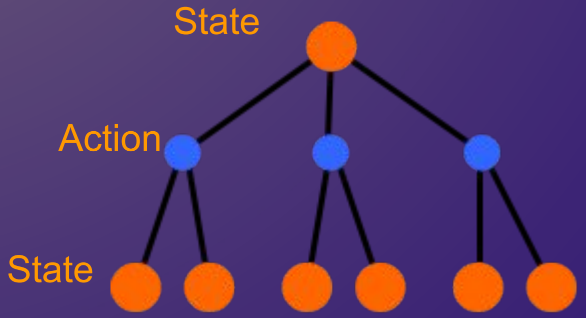
State

SOLVING AN MDP

- If the state and actions are discrete:
  - We have a table of state-action probabilities
  - Learning is filling this table: (dynamic programming)

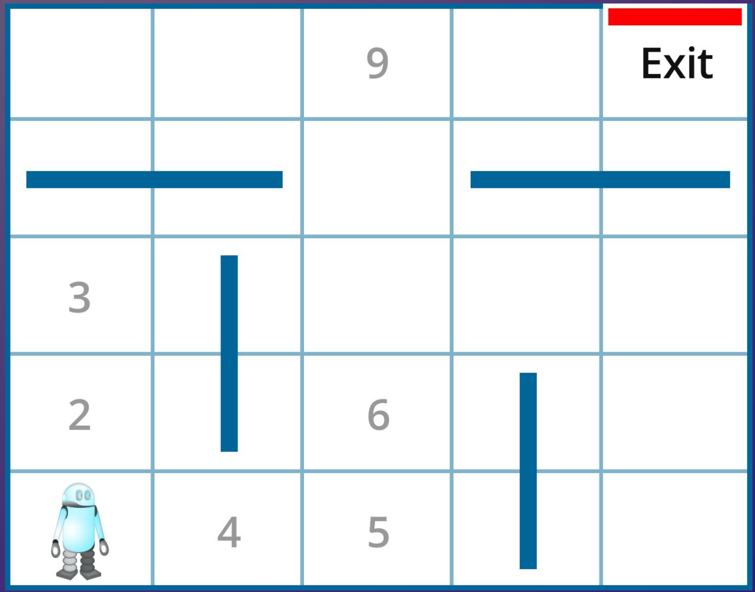
Action

State



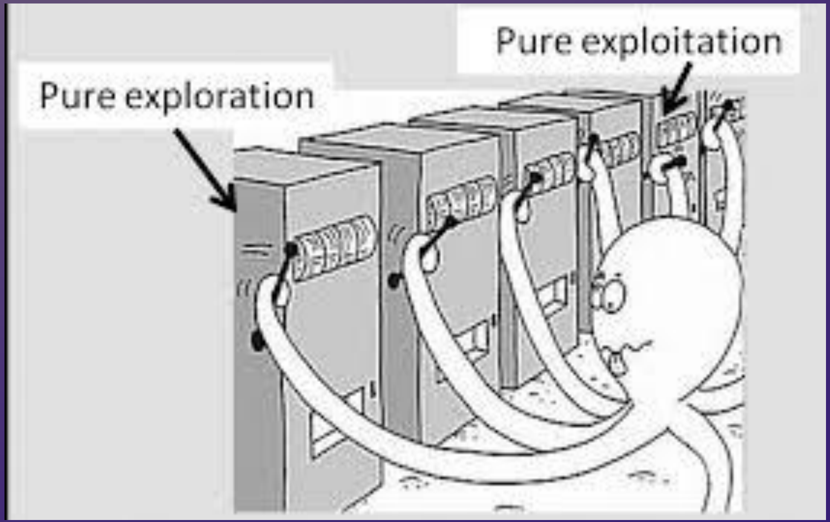
SOLVING AN MDP

- If the state and actions are discrete:
- Let's try different actions and see which one succeed



Exploration-Exploitation dilemma

Do we want to stick to action we think would be good or try something new



Choosing Actions

- Take the action with highest probability (Q-function): Greedy
- Proportionate by its probability: Sampling
- Greedy most times, with some probability random

VALUE FUNCTIONS

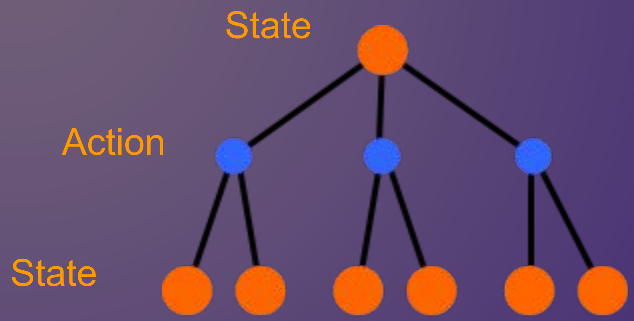
- → Value = expected gain of a state
- → Q function – action specific value function
- → Advantage function – how much more valuable is an action
- → Value depends on future rewards → depends on policy

$$V^{\pi}(s) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t)}[G(s_0, a_0)|s_0 = s]$$

$$Q^{\pi}(s, a) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t)}[G(s_0, a_0)|s_0 = s, a_0 = a]$$

$$A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$$

VALUE FUNCTIONS



$$V^{\pi}(s) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t)}[G(s_0, a_0)|s_0 = s]$$

$$Q^{\pi}(s, a) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t)}[G(s_0, a_0)|s_0 = s, a_0 = a]$$

$$A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$$

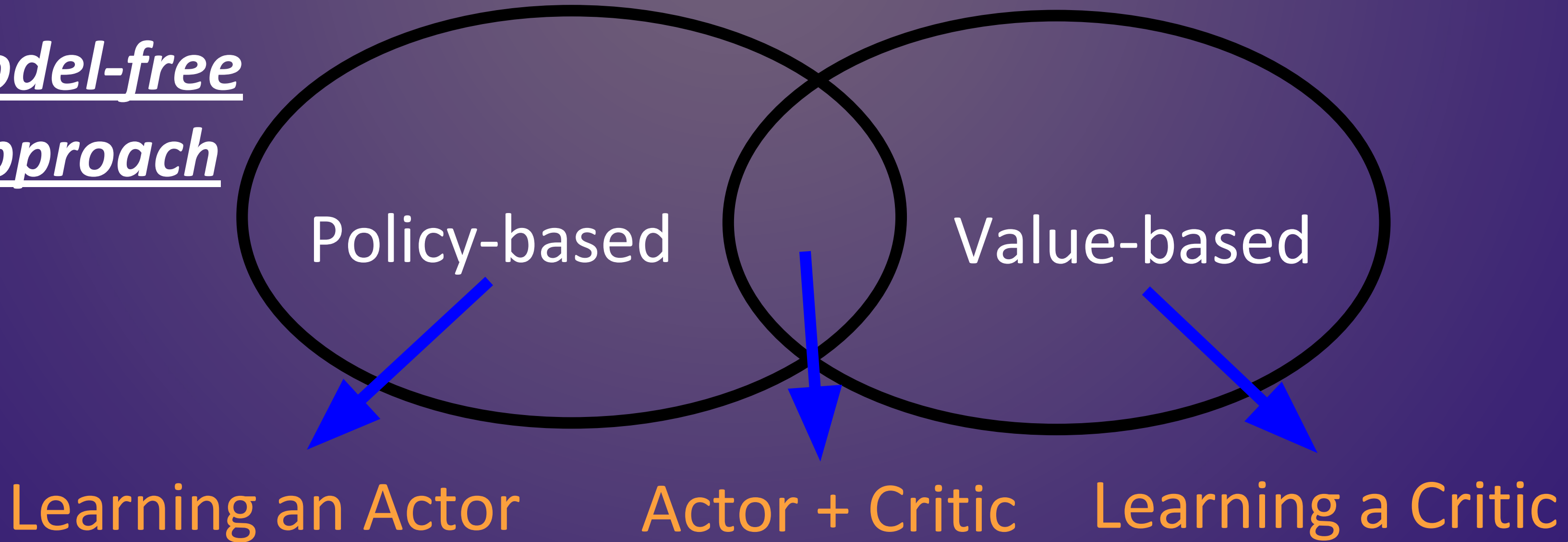
Solving Reinforcement Learning

- Model-based approaches:
  - We model the environment. Do we really need to model all the details of the world?

- Model free approaches:
  - We model the state-actions

Alpha Go: policy-based + value-based + model-based

Model-free Approach



Model-based Approach

POLICY ITERATION

Policy Evaluation

$$V^\pi(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a|s) Q^\pi(s,a)$$

$$Q^\pi(s,a) \leftarrow \pi(a|s) \left(\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s'|s,a) V^\pi(s') \right)$$

Policy Update

$$\pi(s) \leftarrow \arg\max_a Q^\pi(s,a)$$

Q-LEARNING

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\,(r_t + \gamma Q(s_{t+1}, \pi(s_{t+1})) - Q(s_t, a_t))$$

$$\pi(s) \leftarrow \arg\max_a Q(s, a)$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right)$$

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
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

Q-LEARNING

FUNCTION APPROXIMATION

Model: $Q_\theta(s_t, a_t)$

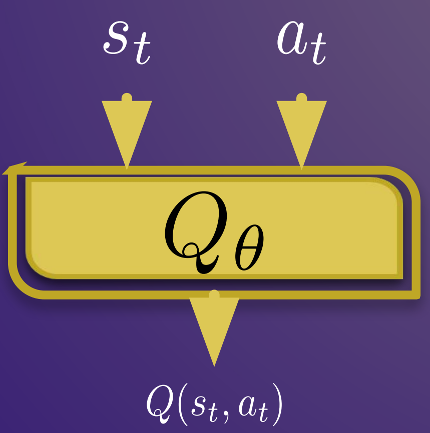
Training data: $\langle s_t, a_t, r_t, s_{t+1} \rangle$

Loss function: $\mathcal{L}(\theta) = ||y_t - Q_\theta(s_t, a_t)||_2^2$

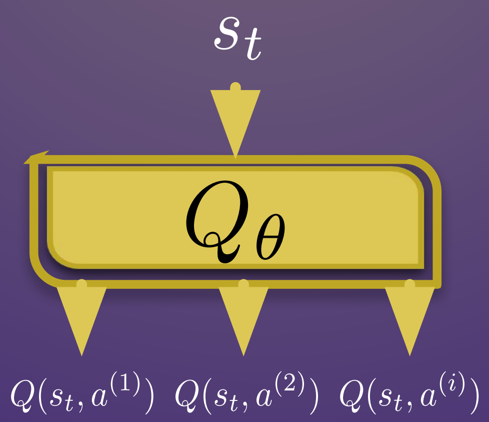
where $y_t = r_t + \gamma Q(s_{t+1}, \pi(s_{t+1}))$

IMPLEMENTATION

Action-in



Action-out



Off-Policy Learning

→ The target depends in part on our model → old observations are still useful

→ Use a Replay Buffer of most recent transitions as dataset

Properties of Reinforcement Learning

- **Reward delay**
  - In space invader, only “fire” obtains reward
    - Although the moving before “fire” is important
  - In Go playing, it may be better to sacrifice immediate reward to gain more long-term reward
- Agent’s actions **affect the subsequent data it receives**
  - E.g. Exploration

DQN ISSUES

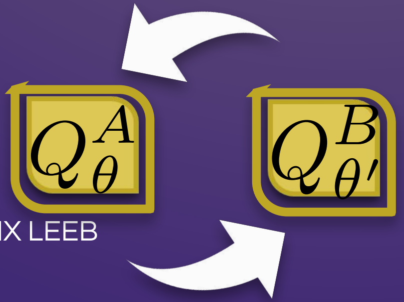
→ Convergence is not guaranteed – hope for deep magic!

Replay Buffer

Reward scaling

Using replicas

→ Double Q Learning – decouple action selection and value estimation



$$\theta_B \leftarrow \tau\theta_A + (1-\tau)\theta_B$$

POLICY GRADIENTS

- → Parameterize policy and update those parameters directly
- → Enables new kinds of policies: stochastic, continuous action spaces

$$\cancel{Q_\theta(s, a)} \qquad \pi(a|s) \rightarrow \pi_\theta(a|s)$$

- → On policy learning → learn directly from your actions

$$\hat{\theta} = \arg\max_{\theta} J(\theta)$$

$$J(\theta) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t;\theta), s_0 \sim \mathcal{T}_0} \left[G(s_0) \right]$$

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{a_t \sim \pi(\cdot|s_t;\theta)}\left[G(s_t, a_t)\right]$$

$$= \nabla_\theta \int \pi(a|s_t;\theta) G(s_t, a) \mathrm{da} = \int \mathrm{da}\ G(s_t, a) \nabla_\theta \pi(a|s_t;\theta)$$

$$= \int \mathrm{da}\ G(s_t, a) \frac{\pi(a|s_t;\theta)}{\pi(a|s_t;\theta)} \nabla_\theta \pi(a|s_t;\theta) = \int \mathrm{da}\ \pi(a|s_t;\theta) G(s_t, a) \frac{\nabla_\theta \pi(a|s_t;\theta)}{\pi(a|s_t;\theta)}$$

$$= \int \mathrm{da}\ \pi(a|s_t;\theta) G(s_t, a) \nabla_\theta \ln \pi(a|s_t;\theta)$$

$$= \mathbb{E}_{a_t \sim \pi(\cdot|s_t;\theta)}\left[G(s_t, a_t) \nabla_\theta \ln \pi(a_t|s_t;\theta)\right]$$

$\longrightarrow$ Approximate expectation value from samples

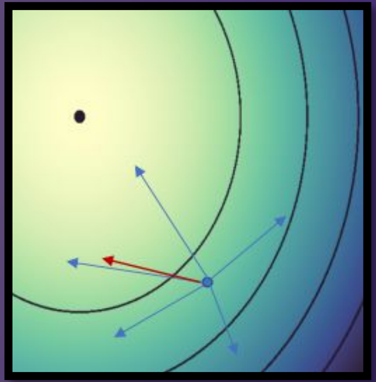
→ Constant offsets make it harder to differentiate the right direction

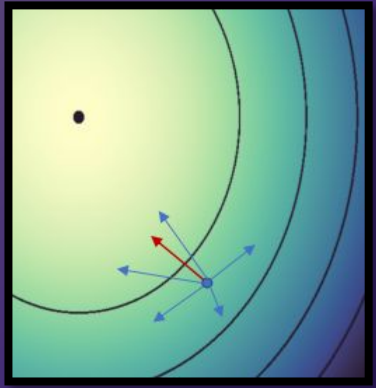
→ Remove offset → a priori value of each state

$$G(s_t, a_t) \approx Q(s_t, a_t)$$

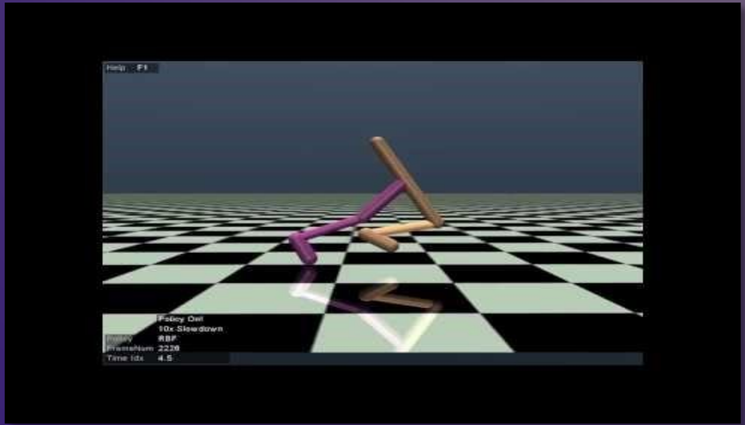
$$\nabla_\theta J(\theta) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t;\theta)} \left[(Q(s_t, a_t) - V(s_t)) \nabla_\theta \ln \pi(a_t|s_t;\theta) \right]$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t;\theta)} \left[A(s_t, a_t) \nabla_\theta \ln \pi(a_t|s_t;\theta) \right]$$

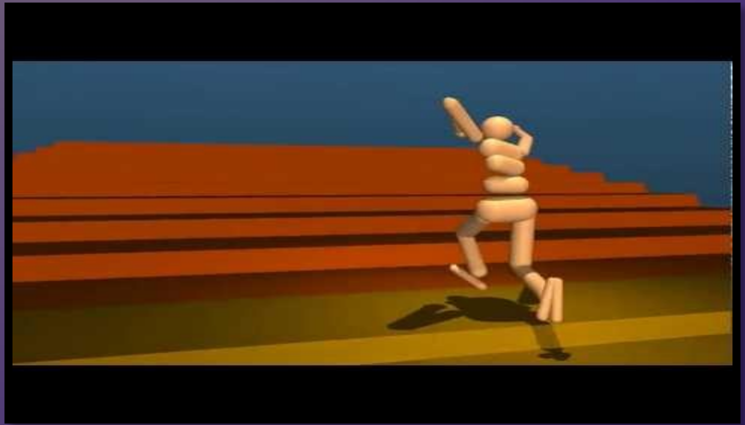




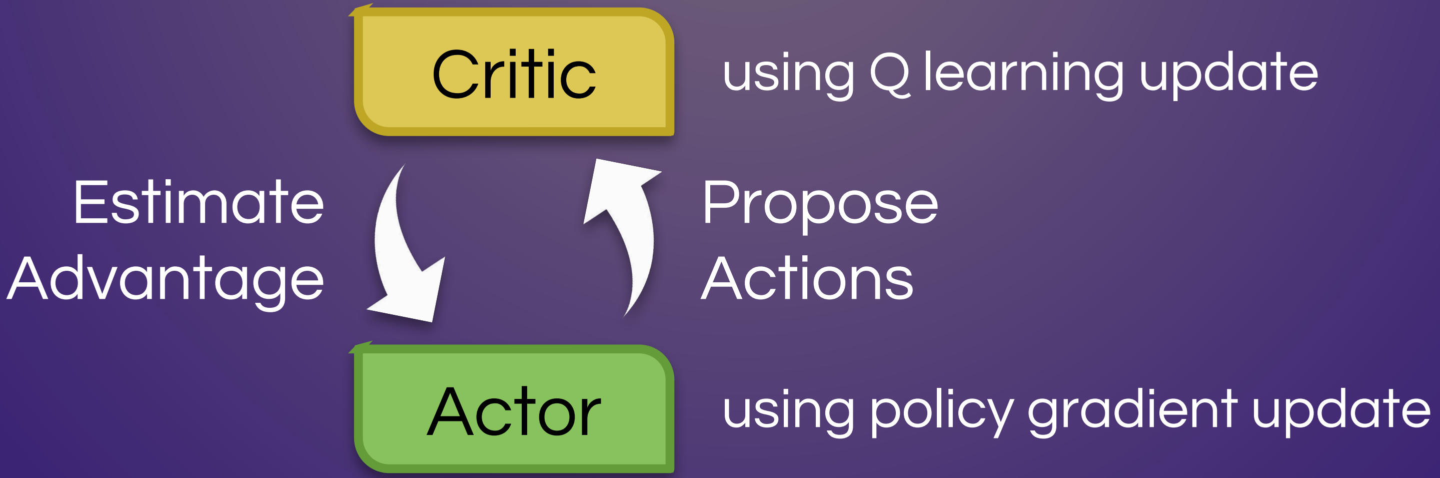
ADVANCED POLICY GRADIENT METHODS



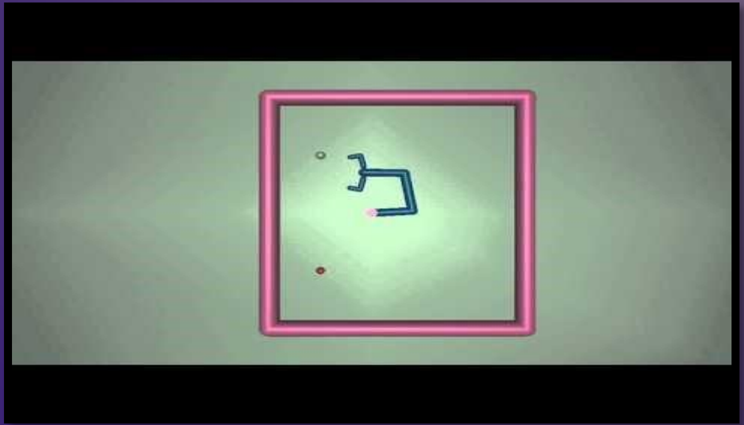
Rajeswaran et al.
(2017)



Heess et al.
(2017)


Critic
using Q learning update
Estimate
Advantage
Propose
Actions
Actor
using policy gradient update

ASYNC ADVANTAGE ACTOR-CRITIC (A3C)



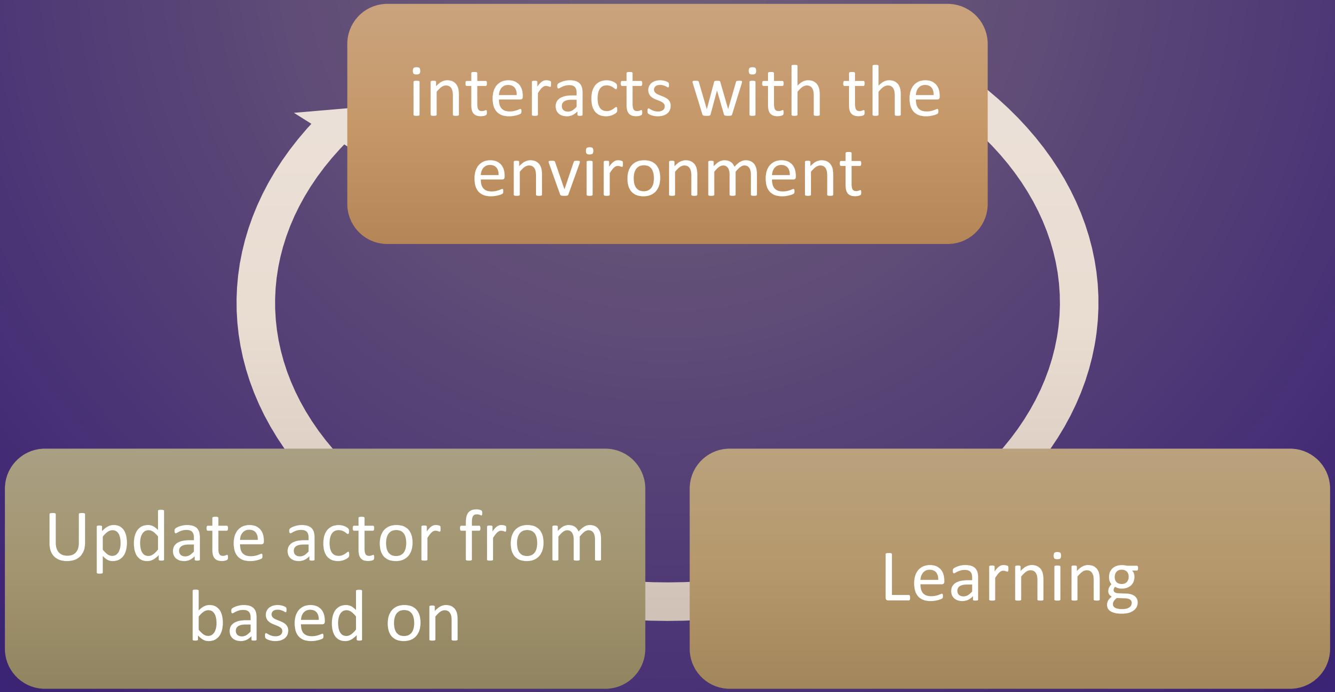
Mnih et al. (2016)

ASYNC ADVANTAGE ACTOR-CRITIC (A3C)

Deep Reinforcement Learning

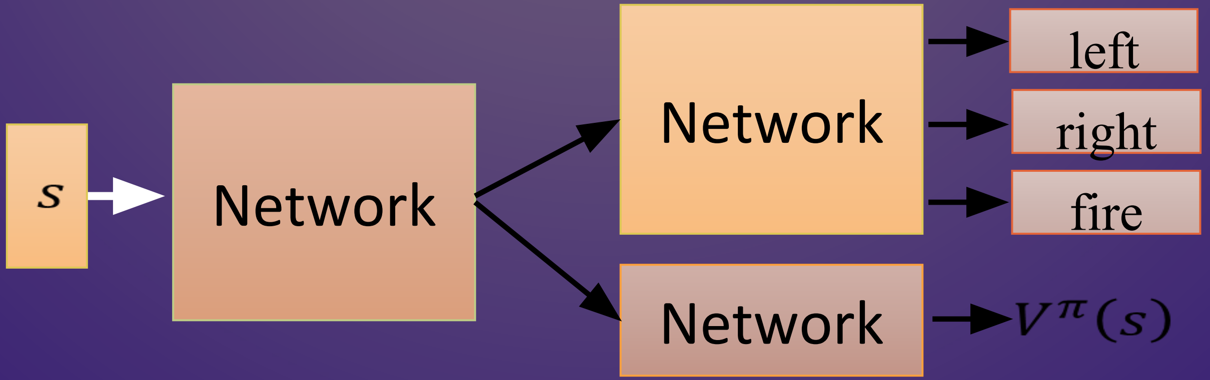
Actor-Critic

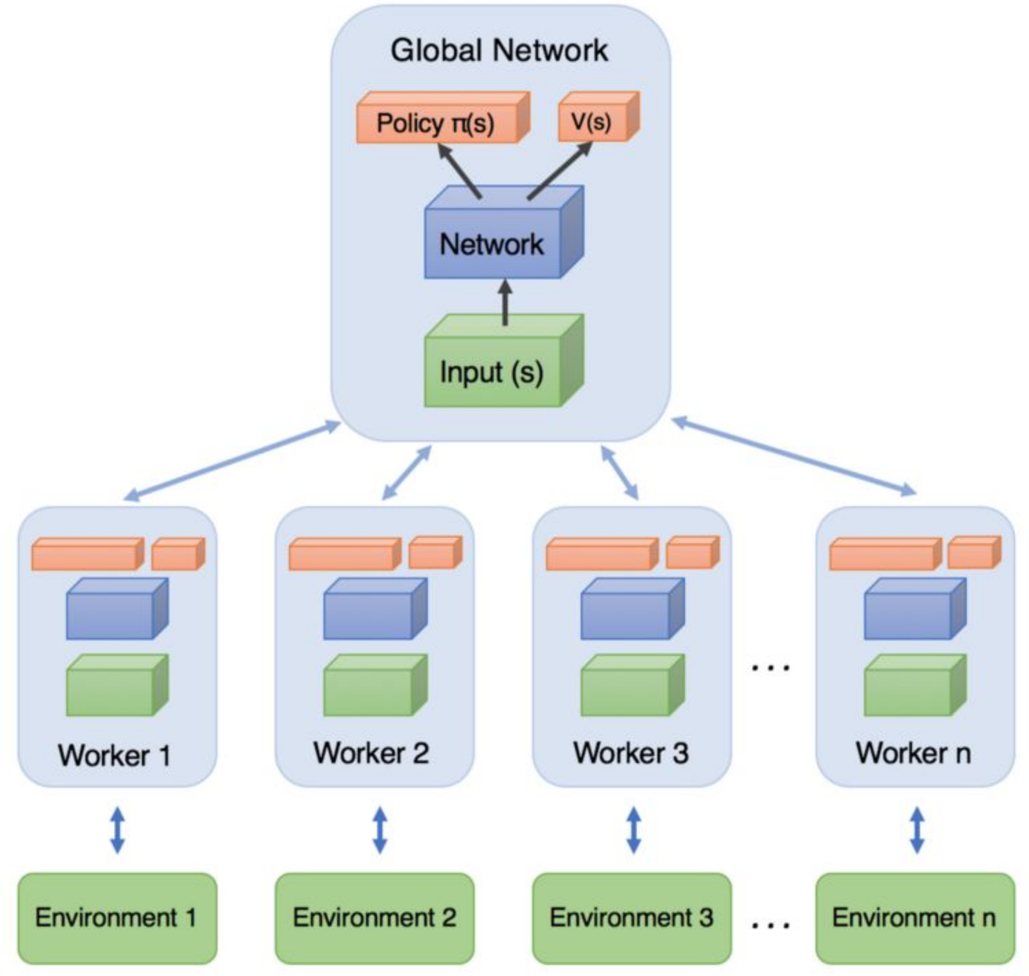
Actor-Critic



Actor-Critic

- Tips
  - The parameters of actor $\pi(s)$ and critic $V^\pi(s)$ can be shared




Global Network
Policy π(s)
V(s)
Network
Input (s)
Worker 1
Worker 2
Worker 3
...
Worker n
Environment 1
Environment 2
Environment 3
...
Environment n

Demo of A3C

- Visual Doom AI Competition @ CIG 2016
- https://www.youtube.com/watch?v=94EPSjQH38Y

Why is it challenging

- Exploration-exploitation dilemma
- How to reward the algorithm.
- How to learn when rewards are very sparse
- What representation do we need for states?
- How to update the policy
- How to incorporate the prior (or logic-based) knowledge
- How to learn for multiple tasks: **General Artificial Intelligence**

Reference

- Textbook: Reinforcement Learning: An Introduction
  - http://incompleteideas.net/sutton/book/the-book.html
- Lectures of David Silver
  - http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html (10 lectures, around 1:30 each)
  - http://videolectures.net/rldm2015_silver_reinforcement_learning/ (Deep Reinforcement Learning)
- Lectures of John Schulman
  - https://youtu.be/aUrX-rP_ss4