

Deep Reinforcement Learning

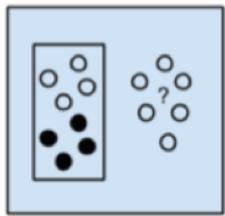
MIT 6.SI91

Alexander Amini
January 30, 2019

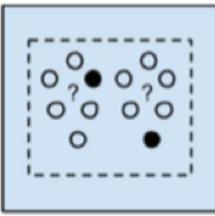


Play Video@ 01:00

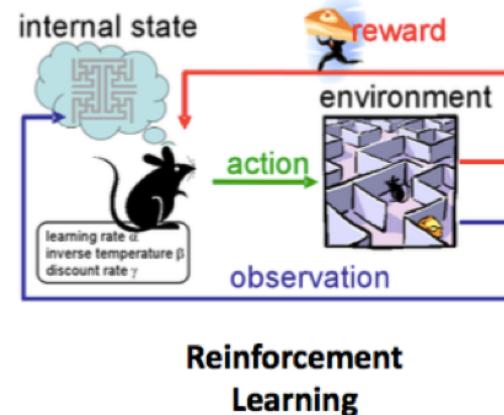
Types of Deep Learning



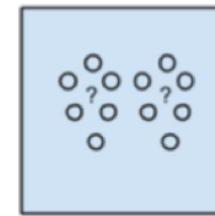
Supervised
Learning



Semi-Supervised
Learning



Reinforcement
Learning



Unsupervised
Learning



Classes of Learning Problems

Supervised Learning

Data: (x, y)

x is data, y is label

Goal: Learn function to map

$$x \rightarrow y$$

Apple example:



This thing is an apple.

Classes of Learning Problems

Supervised Learning

Data: (x, y)

x is data, y is label

Goal: Learn function to map
 $x \rightarrow y$

Apple example:



This thing is an apple.

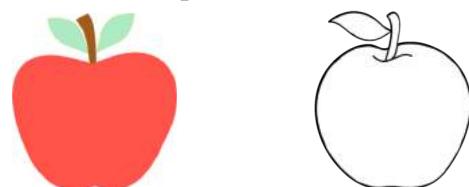
Unsupervised Learning

Data: x

x is data, no labels!

Goal: Learn underlying
structure

Apple example:



This thing is like
the other thing.

Classes of Learning Problems

Supervised Learning

Data: (x, y)

x is data, y is label

Goal: Learn function to map
 $x \rightarrow y$

Apple example:



This thing is an apple.

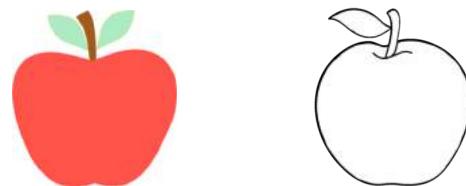
Unsupervised Learning

Data: x

x is data, no labels!

Goal: Learn underlying
structure

Apple example:



This thing is like
the other thing.

Reinforcement Learning

Data: state-action pairs

Goal: Maximize future rewards
over many time steps

Apple example:



Eat this thing because it
will keep you alive.

Classes of Learning Problems

Supervised Learning

Data: (x, y)

x is data, y is label

Goal: Learn function mapping
 $x \rightarrow y$

Apple example:



This is an apple.

Unsupervised Learning

Data: x

x is data, no labels!

Goal: Learn underlying
structure

Apple example:



This thing is like
the other thing.

Reinforcement Learning

Data: state-action pairs

Goal: Maximize future rewards
over many time steps

Apple example:



Eat this thing because it
will keep you alive.

Reinforcement Learning (RL): Key Concepts



AGENT

Agent: takes actions.

Reinforcement Learning (RL): Key Concepts



AGENT



ENVIRONMENT

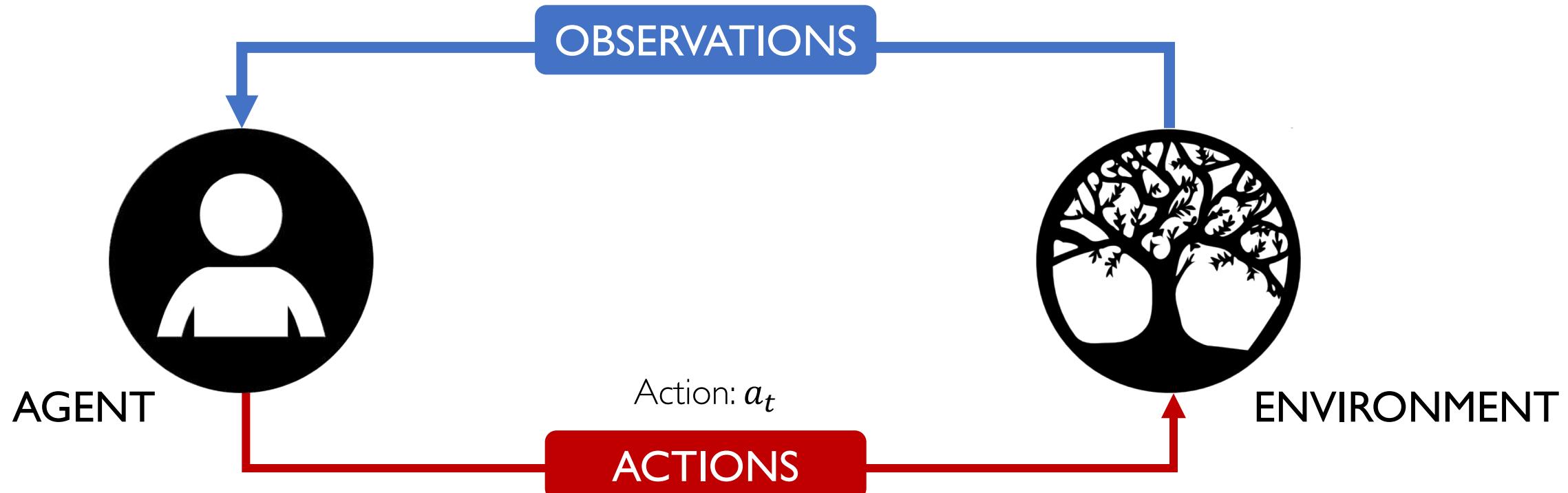
Environment: the world in which the agent exists and operates.

Reinforcement Learning (RL): Key Concepts



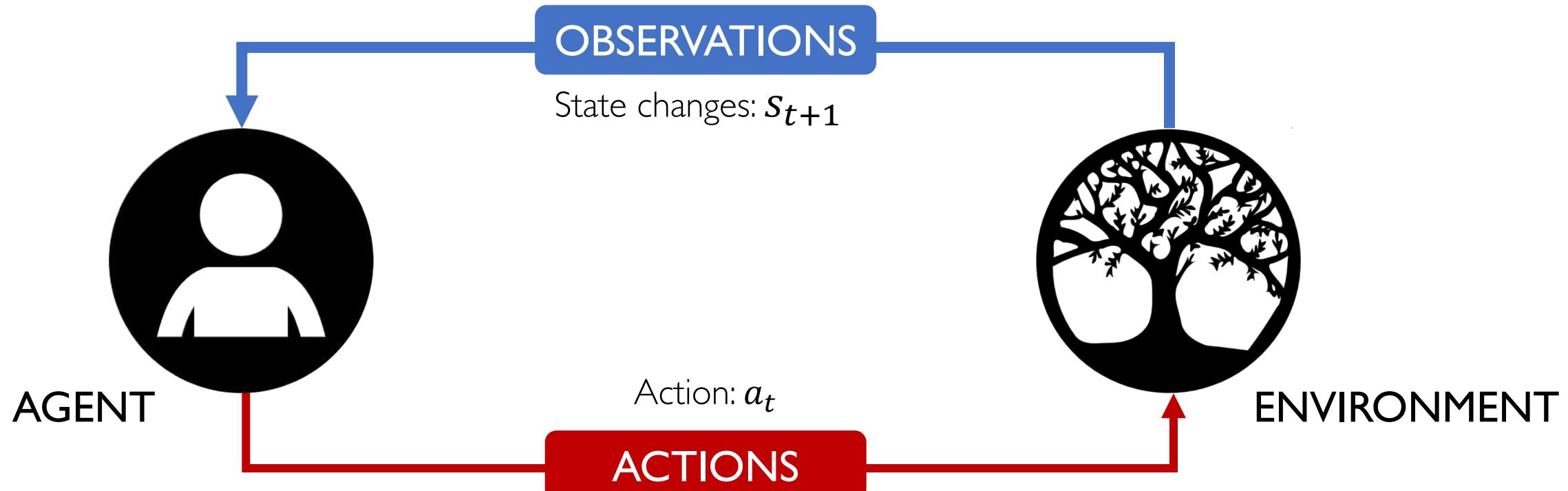
Action: a move the agent can make in the environment.

Reinforcement Learning (RL): Key Concepts



Observations: of the environment after taking actions.

Reinforcement Learning (RL): Key Concepts



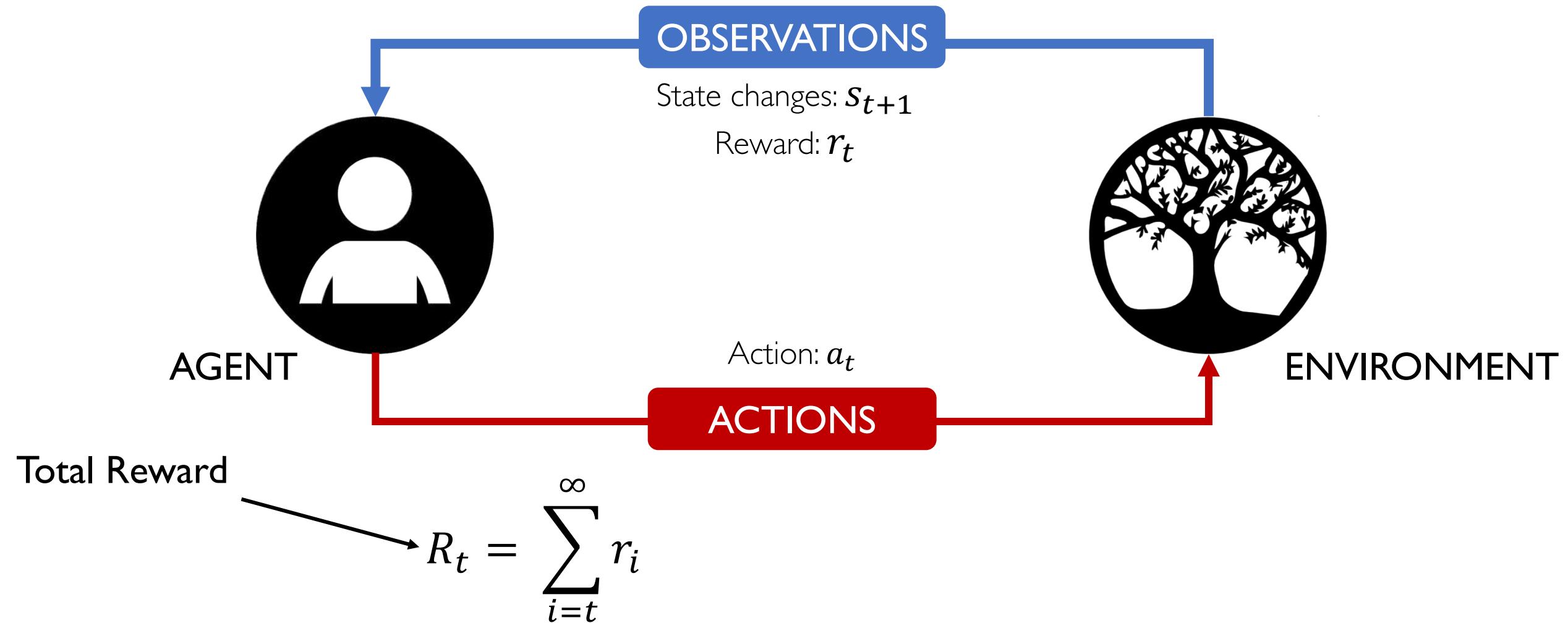
State: a situation which the agent perceives.

Reinforcement Learning (RL): Key Concepts

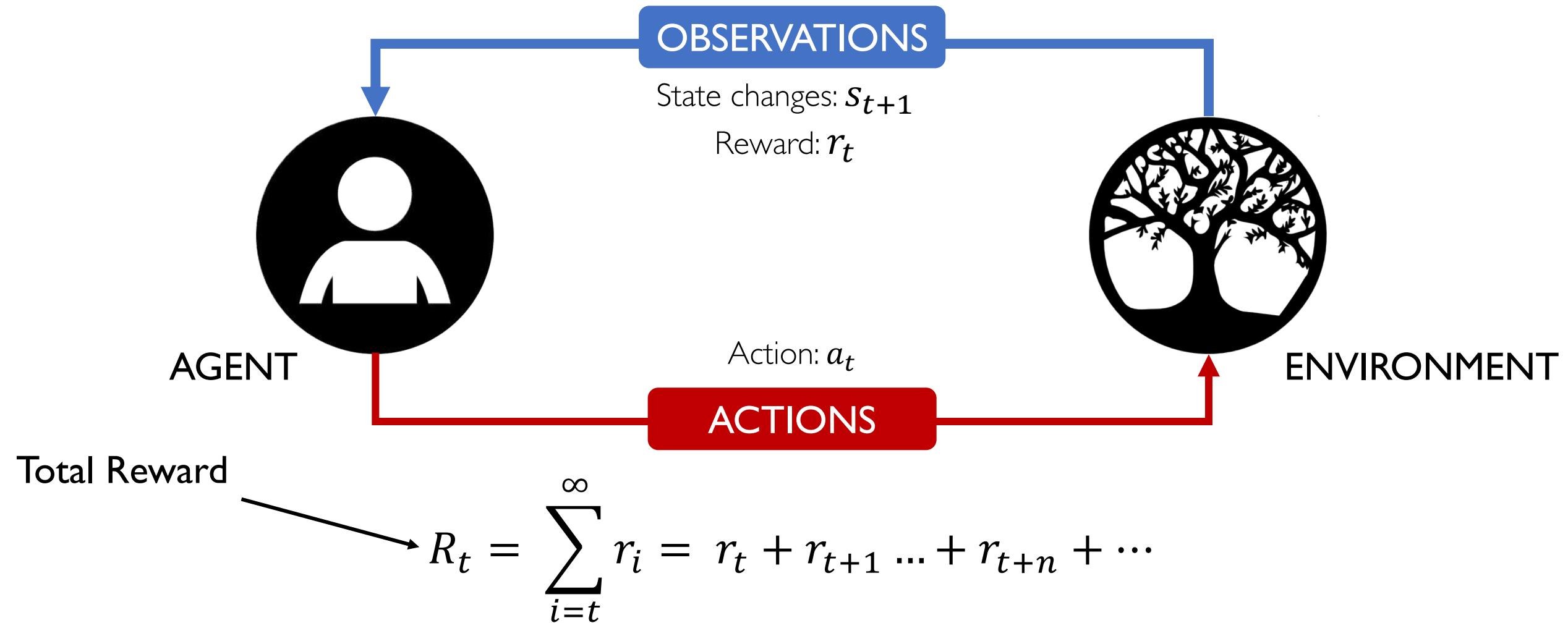


Reward: feedback that measures the success or failure of the agent's action.

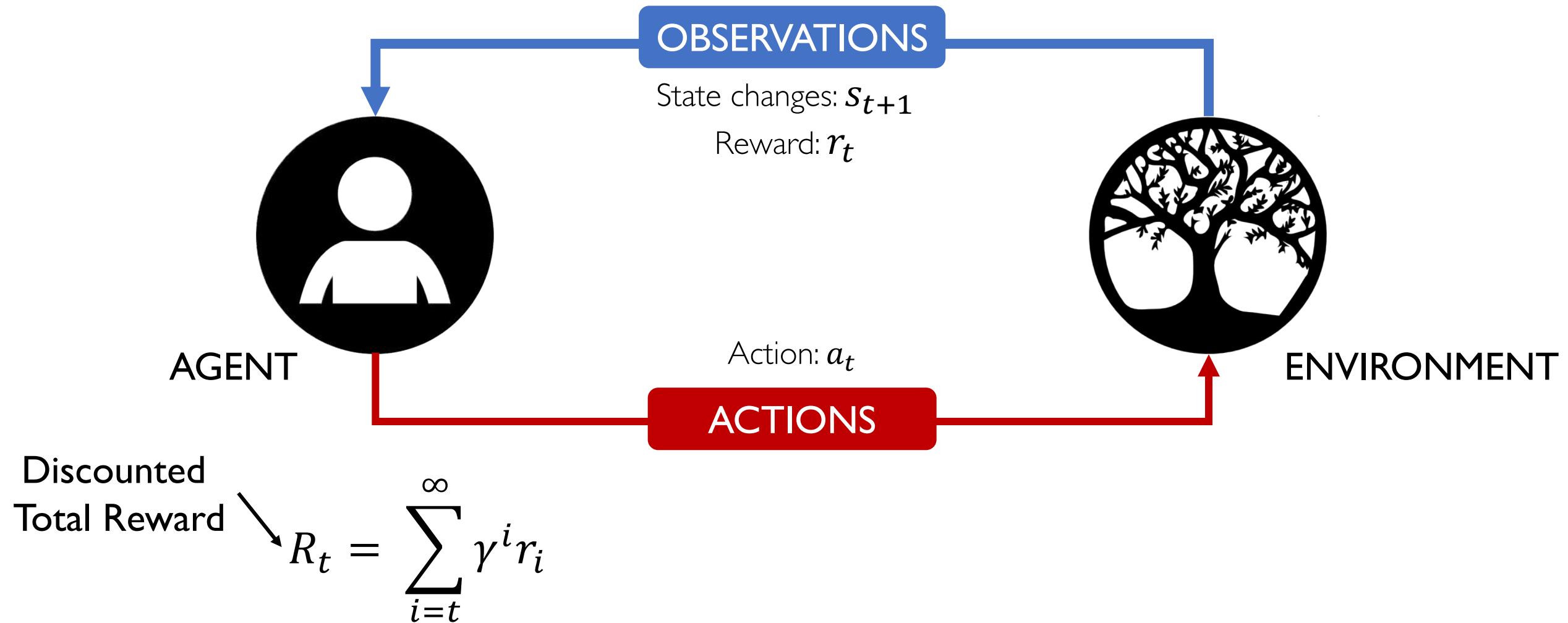
Reinforcement Learning (RL): Key Concepts



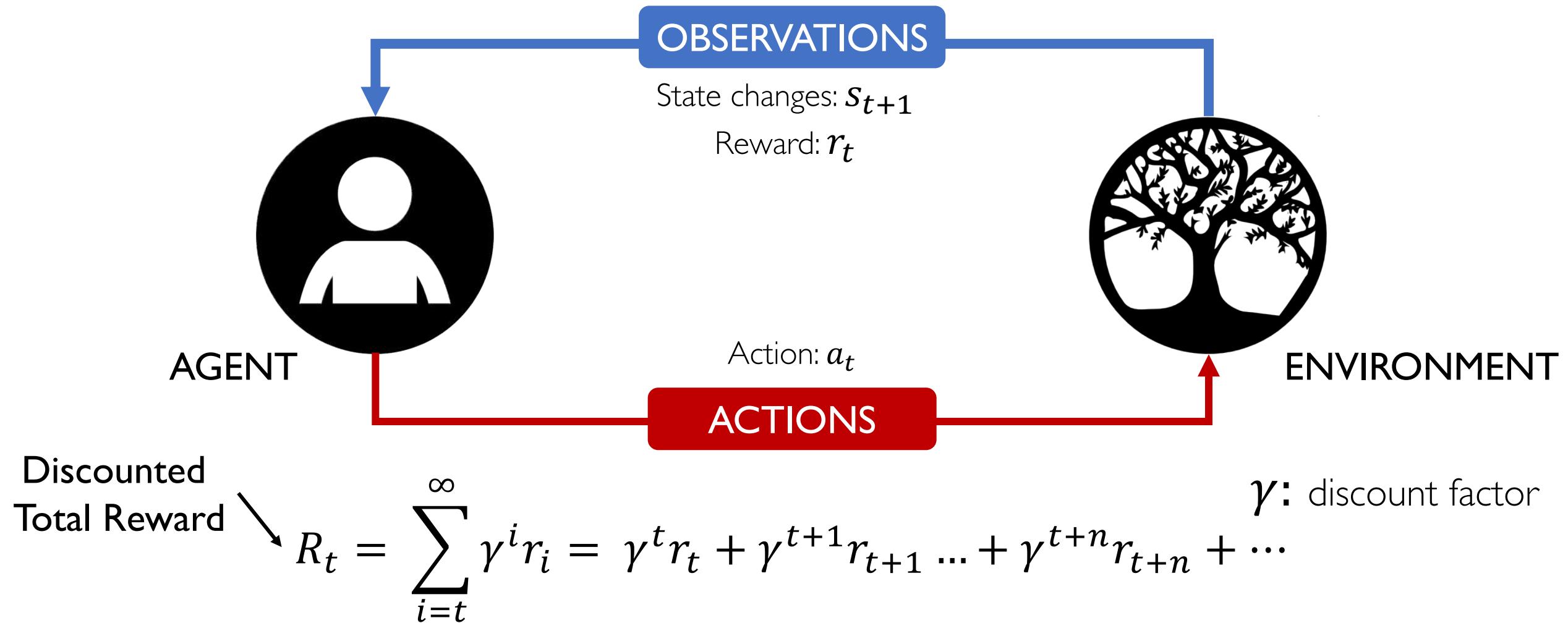
Reinforcement Learning (RL): Key Concepts



Reinforcement Learning (RL): Key Concepts



Reinforcement Learning (RL): Key Concepts



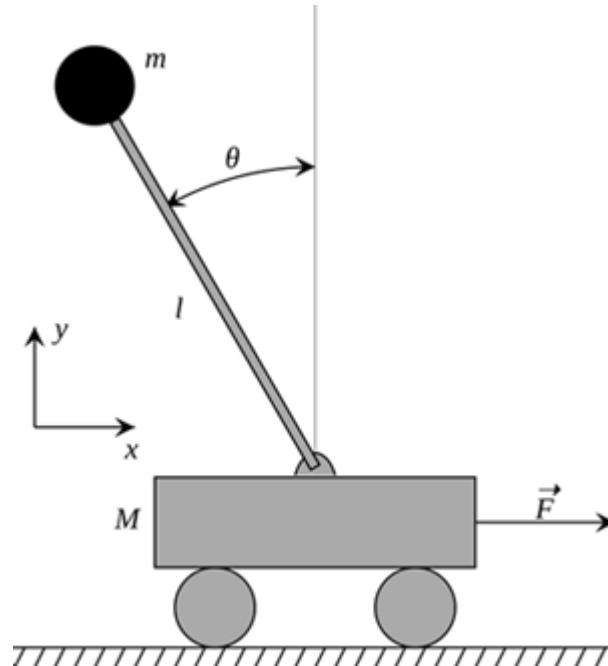
Examples of Reinforcement Learning

Reinforcement learning is a general-purpose framework for decision-making:

- An agent operates in an environment: **Atari Breakout**
- An agent has the capacity to **act**
- Each action influences the agent's **future state**
- Success is measured by a **reward** signal
- **Goal** is to select actions to **maximize future reward**



Examples of Reinforcement Learning



Cart-Pole Balancing

- **Goal** — Balance the pole on top of a moving cart
- **State** — Pole angle, angular speed. Cart position, horizontal velocity.
- **Actions** — horizontal force to the cart
- **Reward** — 1 at each time step if the pole is upright

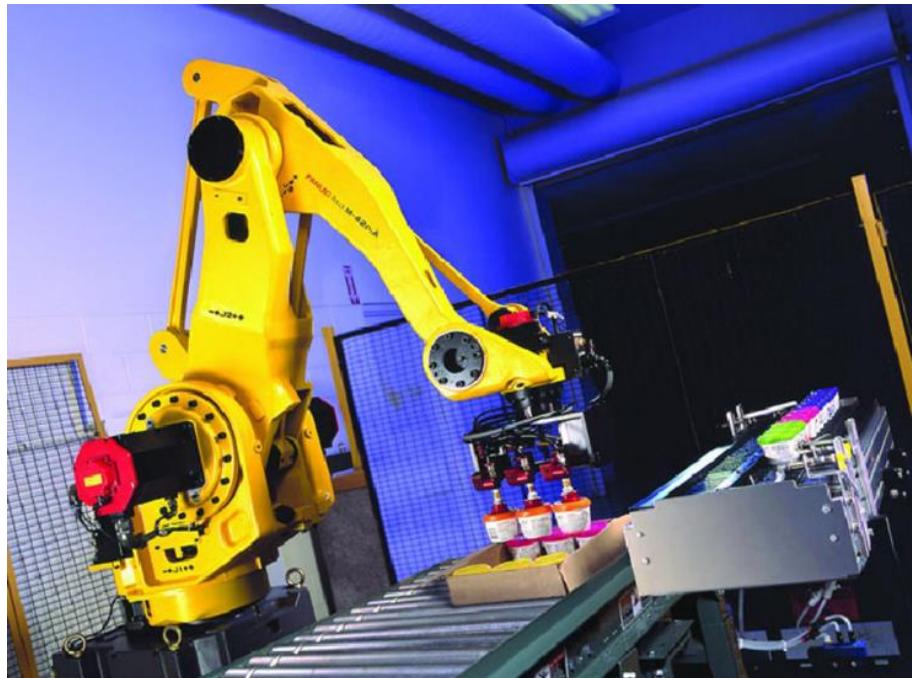
Examples of Reinforcement Learning



Doom

- **Goal** — Eliminate all opponents
- **State** — Raw game pixels of the game
- **Actions** — Up, Down, Left, Right etc
- **Reward** — Positive when eliminating an opponent, negative when the agent is eliminated

Examples of Reinforcement Learning



Bin Packing

- **Goal** - Pick a device from a box and put it into a container
- **State** - Raw pixels of the real world
- **Actions** - Possible actions of the robot
- **Reward** - Positive when placing a device successfully, negative otherwise

Defining the Q-function

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots$$

Total reward, R_t , is the discounted sum of all rewards obtained from time t

$$Q(s, a) = \mathbb{E}[R_t]$$

The Q-function captures the **expected total future reward** an agent in state, s , can receive by executing a certain action, a

How to take actions given a Q-function?

$$Q(s, a) = \mathbb{E}[R_t]$$

↑ ↑
(state, action)

Ultimately, the agent needs a **policy** $\pi(s)$, to infer the **best action to take** at its state, s

Strategy: the policy should choose an action that maximizes future reward

$$\pi^*(s) = \operatorname{argmax}_a Q(s, a)$$

Deep Reinforcement Learning Algorithms

Value Learning

Find $Q(s, a)$

$$a = \underset{a}{\operatorname{argmax}} Q(s, a)$$

Policy Learning

Find $\pi(s)$

Sample $a \sim \pi(s)$

Deep Reinforcement Learning Algorithms

Value Learning

Find $Q(s, a)$

$$a = \underset{a}{\operatorname{argmax}} Q(s, a)$$

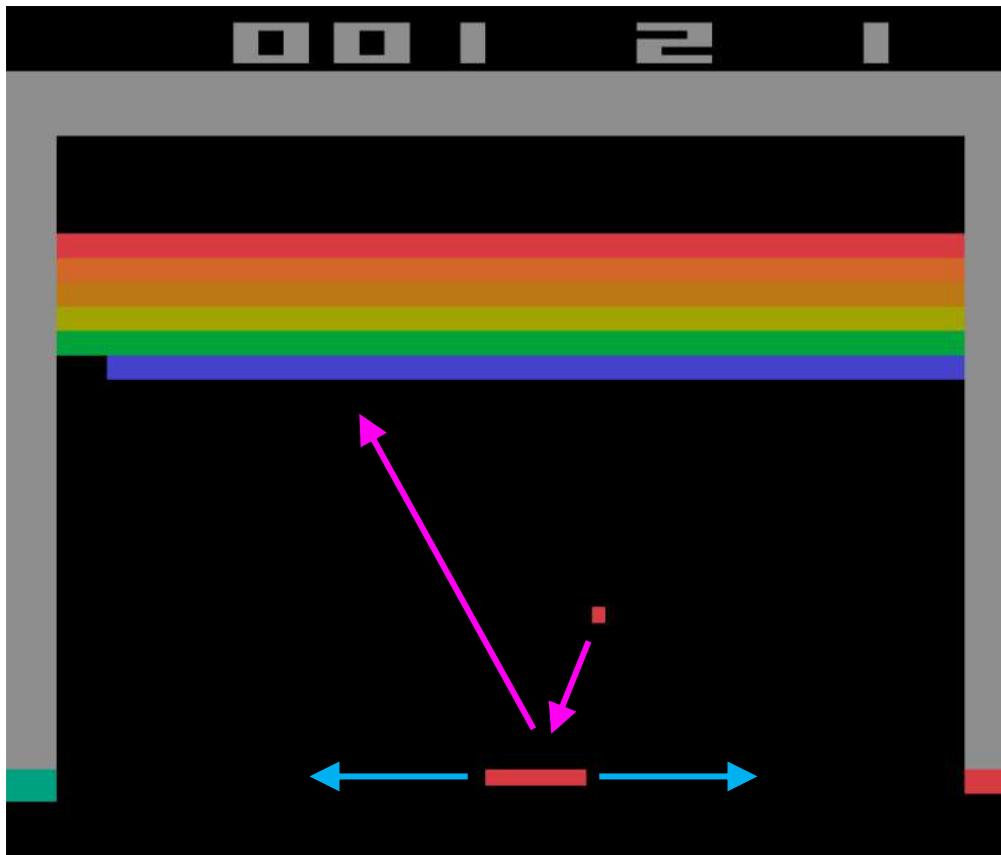
Policy Learning

Find $\pi(s)$

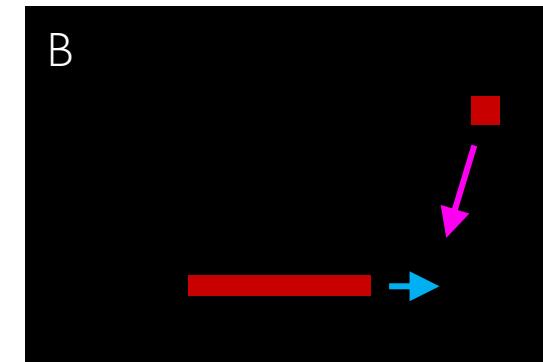
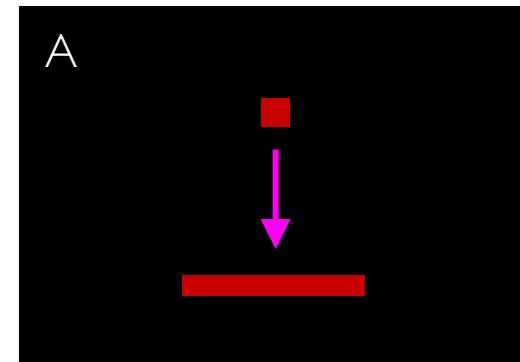
Sample $a \sim \pi(s)$

Digging deeper into the Q-function

Example: Atari Breakout



It can be very difficult for humans to accurately estimate Q-values

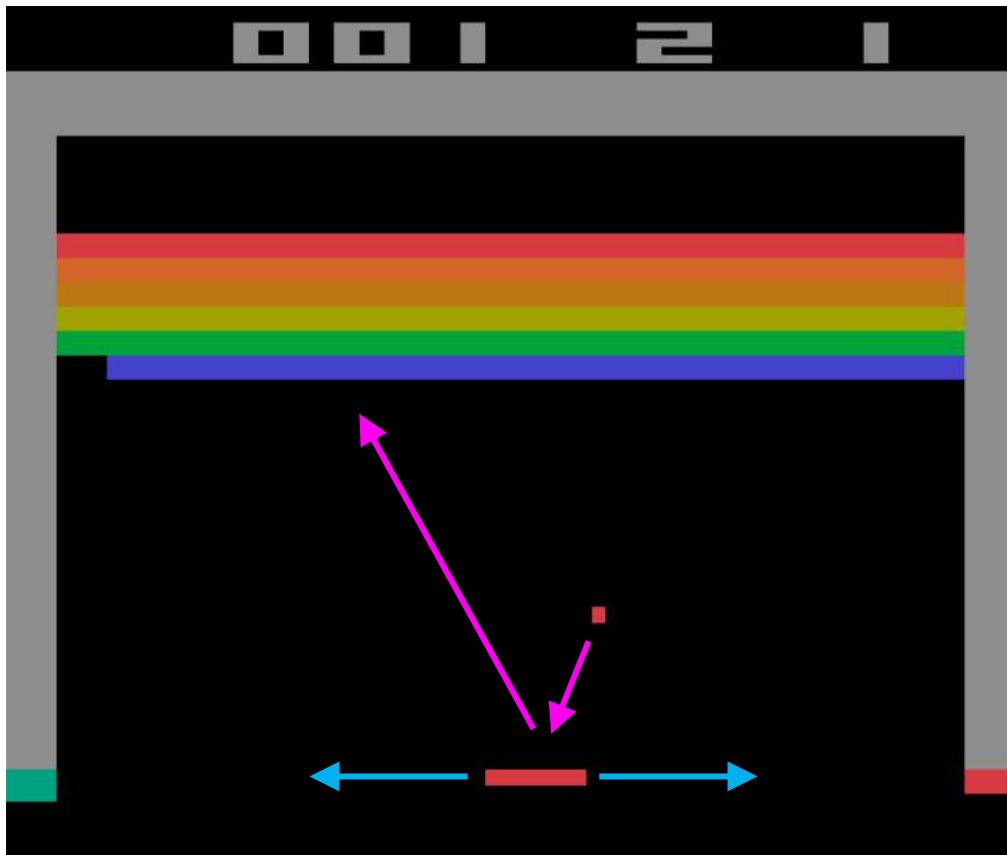


Which (s, a) pair has a higher Q-value?

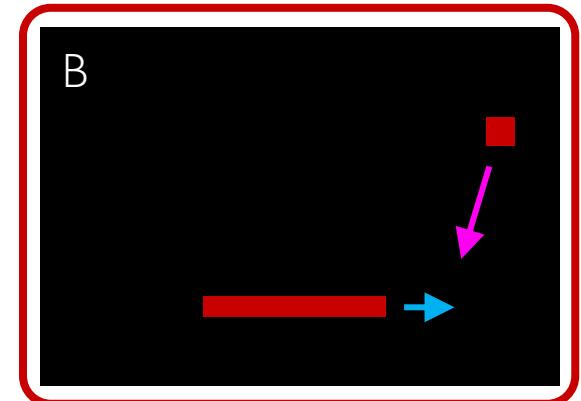
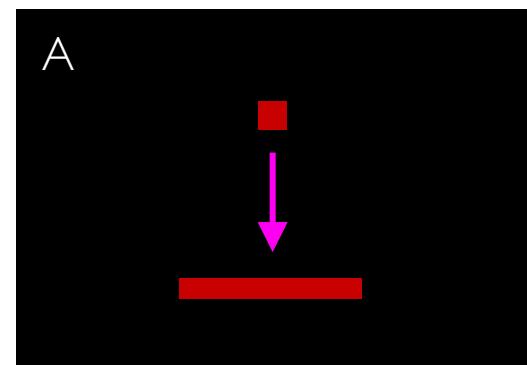


Digging deeper into the Q-function

Example: Atari Breakout



It can be very difficult for humans to accurately estimate Q-values

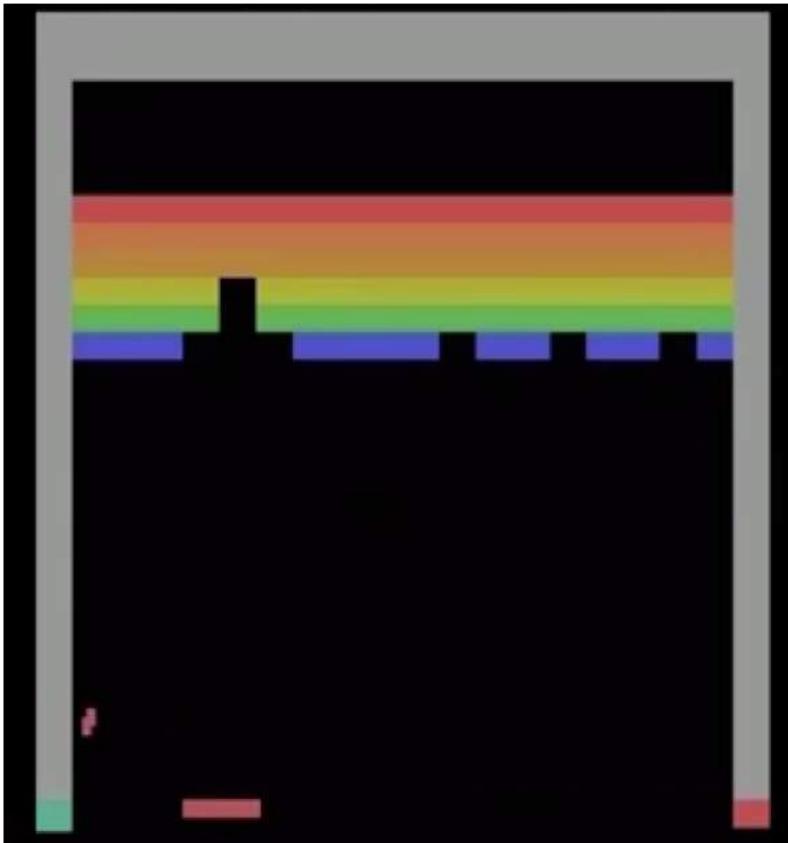


Which (s, a) pair has a higher Q-value?

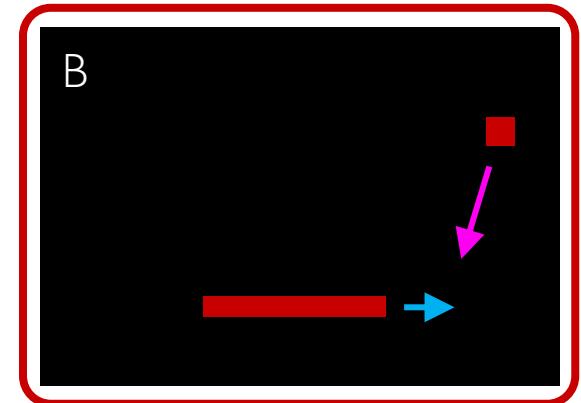
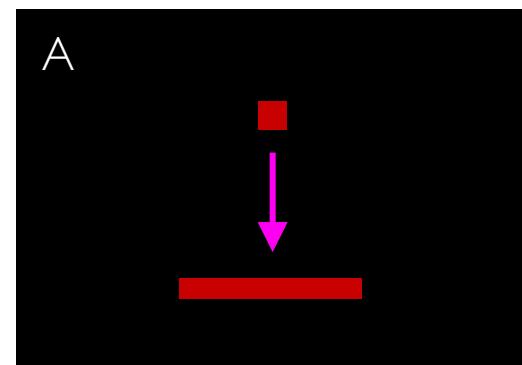


Digging deeper into the Q-function

Example: Atari Breakout - Middle



It can be very difficult for humans to accurately estimate Q-values



Which (s, a) pair has a higher Q-value?

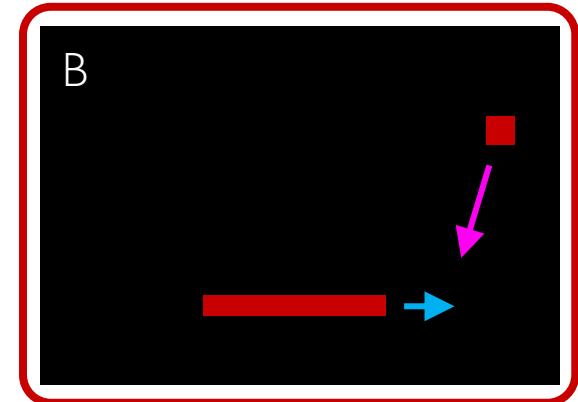
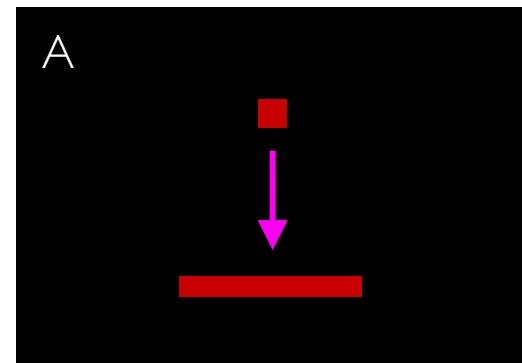


Digging deeper into the Q-function

Example: Atari Breakout - Side



It can be very difficult for humans to accurately estimate Q-values



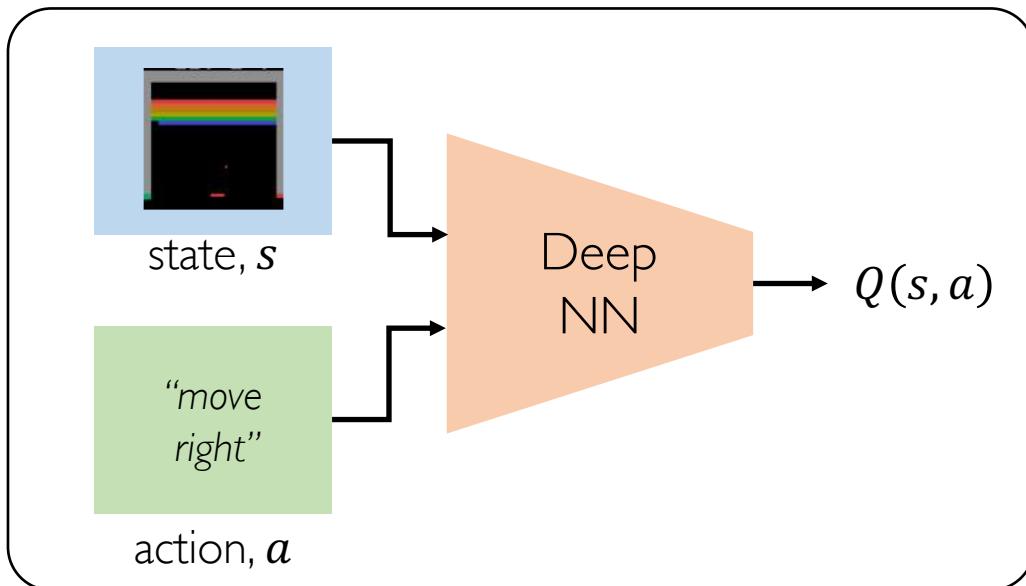
Which (s, a) pair has a higher Q-value?



Play Video @19:48

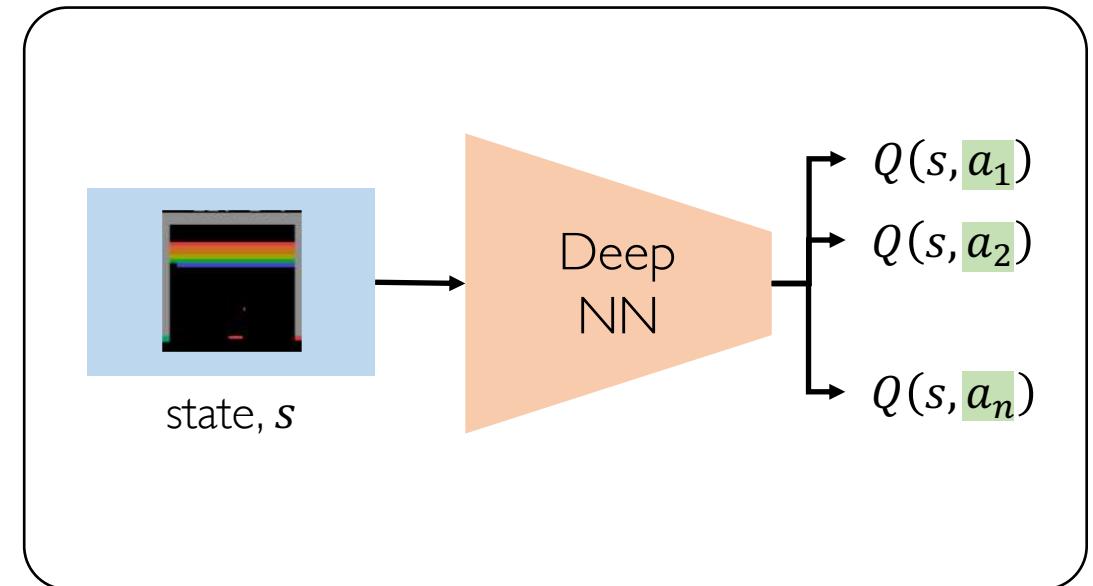
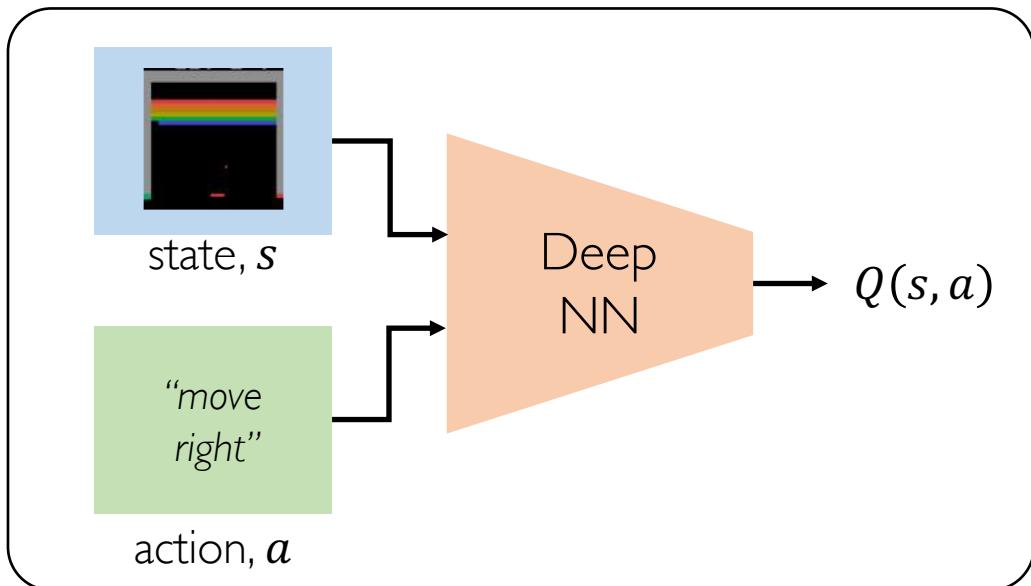
Deep Q Networks (DQN)

How can we use deep neural networks to model Q-functions?



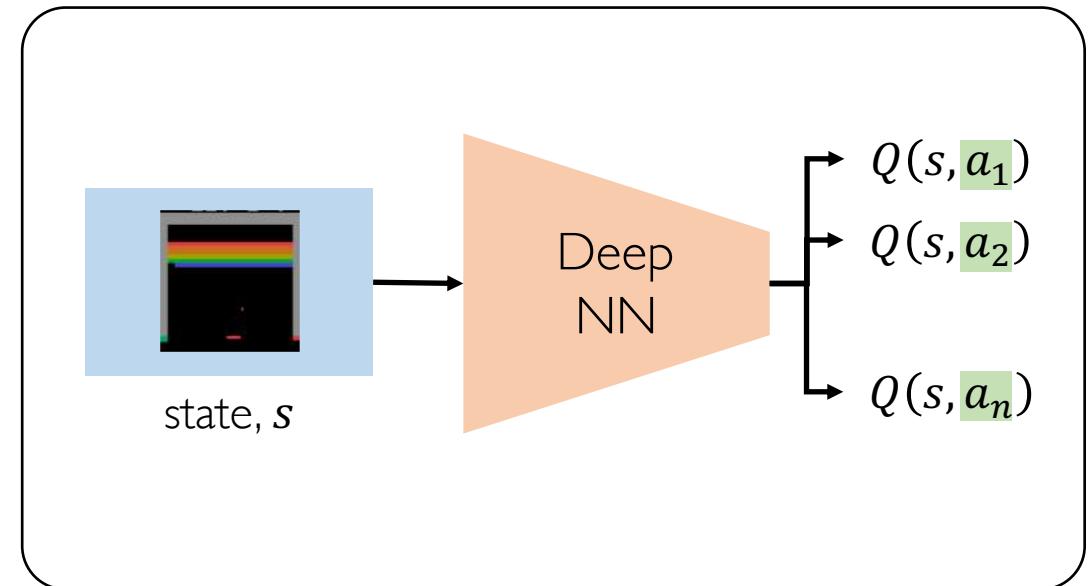
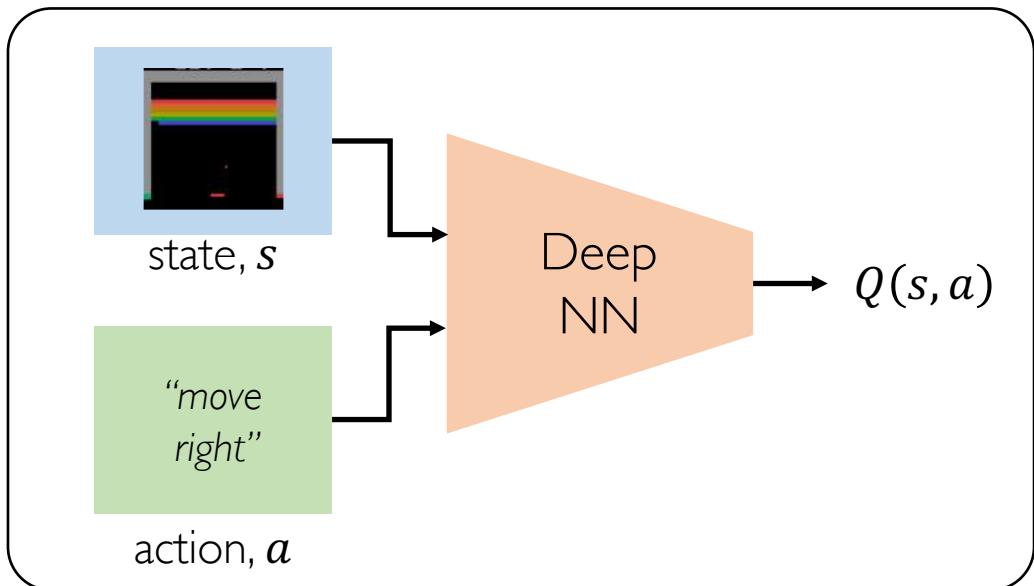
Deep Q Networks (DQN)

How can we use deep neural networks to model Q-functions?



Deep Q Networks (DQN): Training

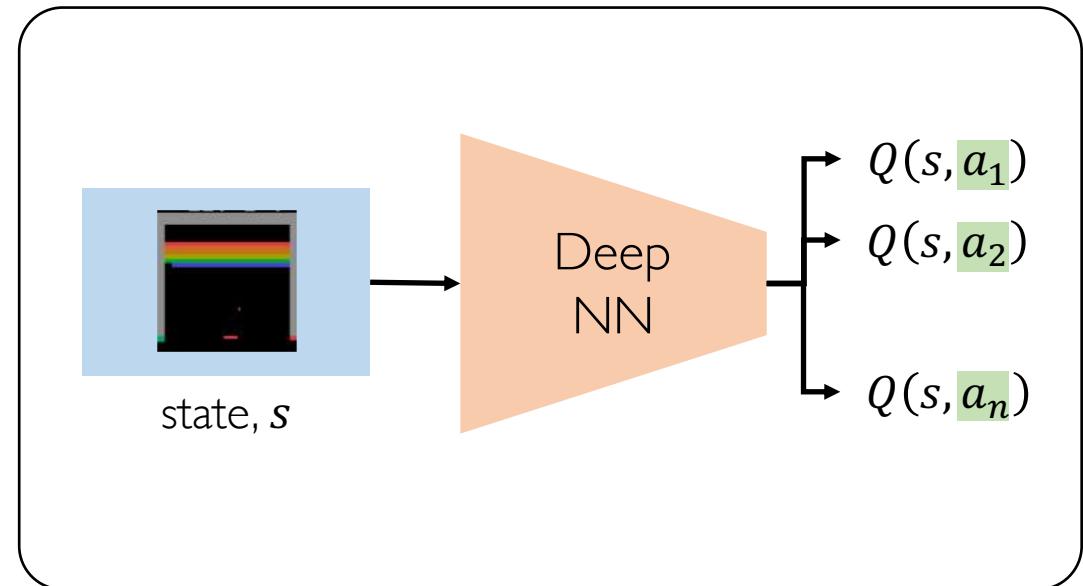
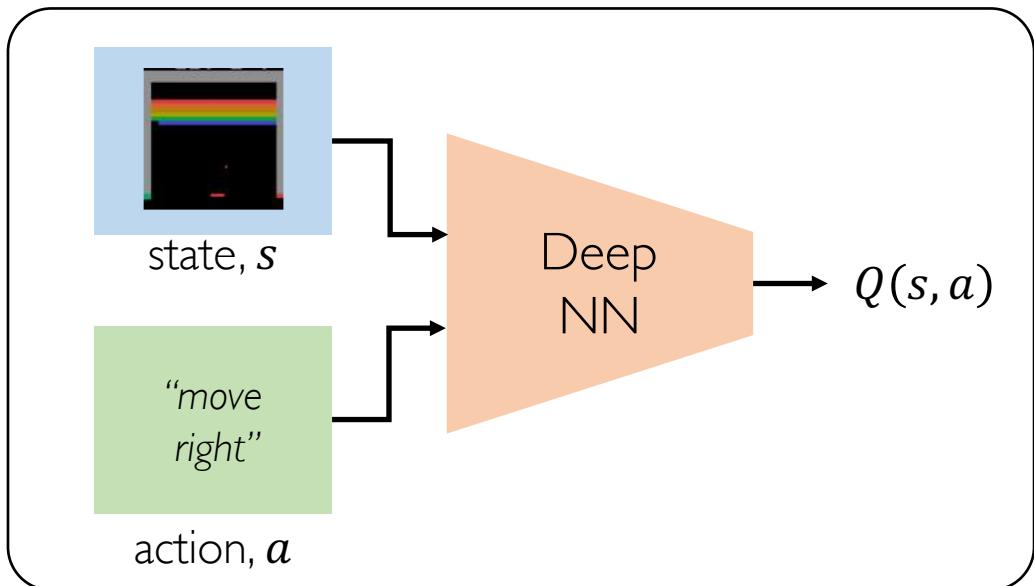
How can we use deep neural networks to model Q-functions?



$$\mathcal{L} = \mathbb{E} \left[\left\| \left(r + \gamma \max_{a'} Q(s', a') \right) - Q(s, a) \right\|^2 \right]$$

Deep Q Networks (DQN): Training

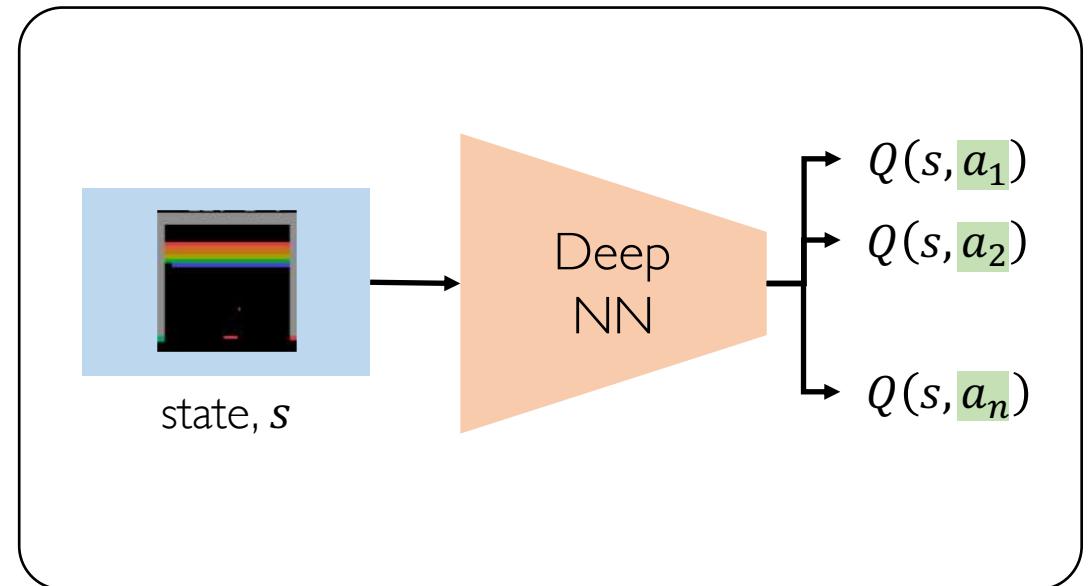
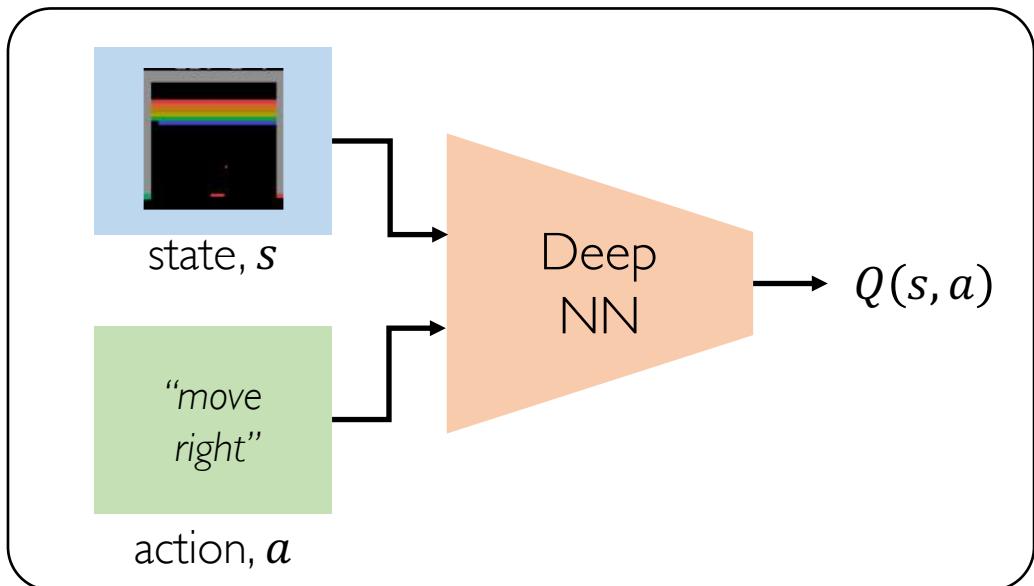
How can we use deep neural networks to model Q-functions?



$$\mathcal{L} = \mathbb{E} \left[\left\| \underbrace{\left(r + \gamma \max_{a'} Q(s', a') \right)}_{\text{target}} - \underbrace{Q(s, a)}_{\text{predicted}} \right\|^2 \right]$$

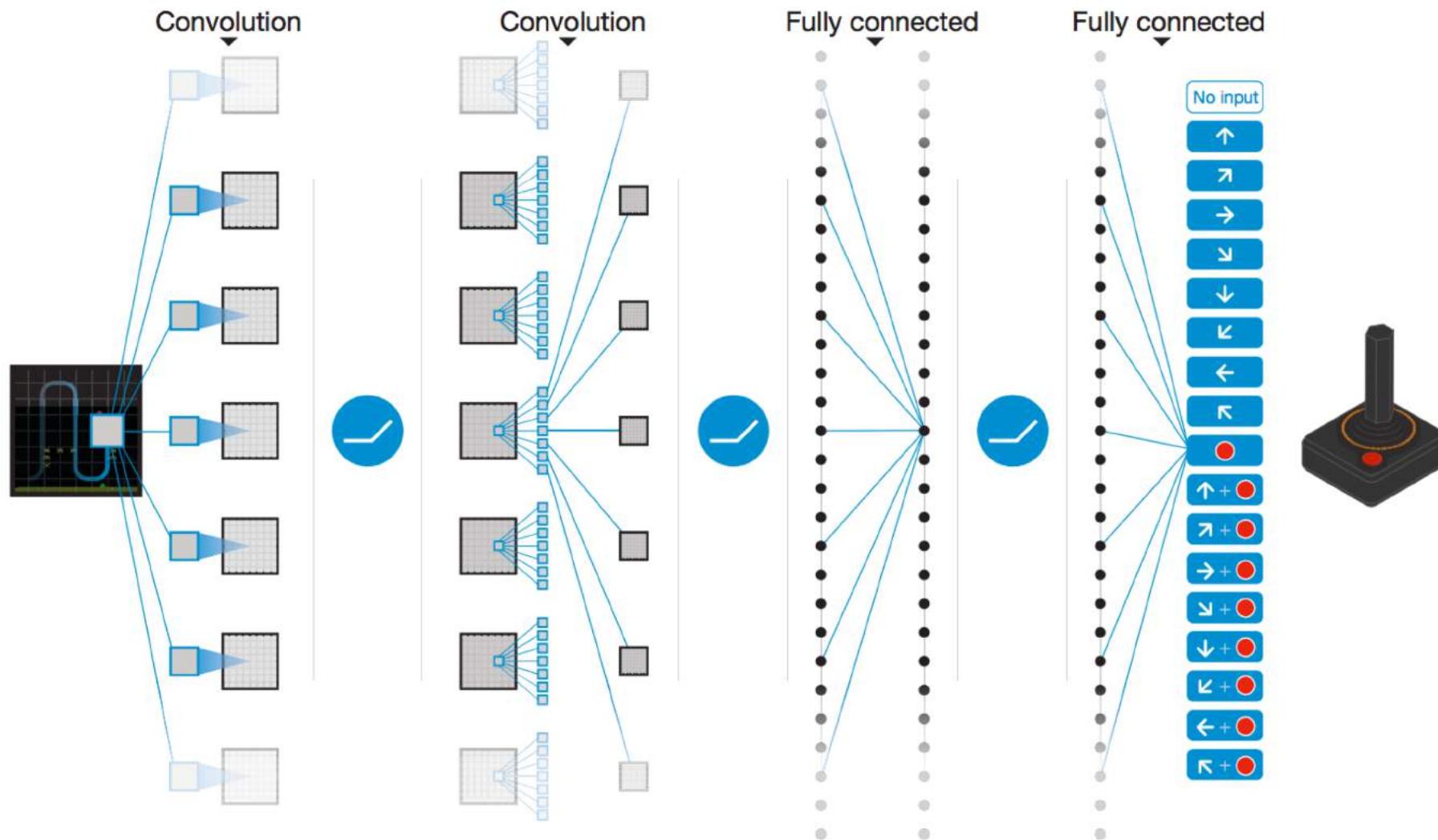
Deep Q Networks (DQN): Training

How can we use deep neural networks to model Q-functions?

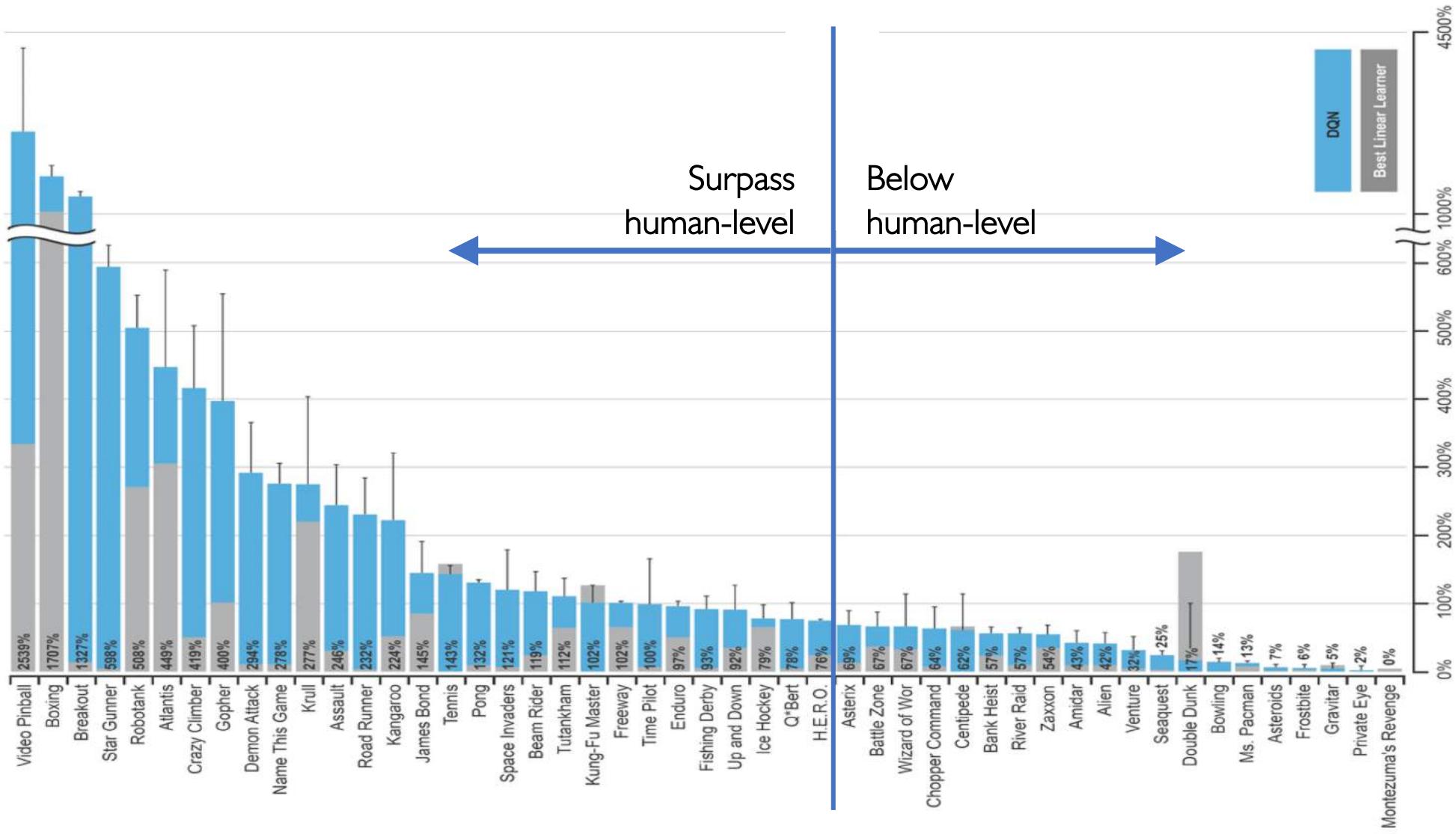


$$\mathcal{L} = \mathbb{E} \left[\left\| \underbrace{\left(r + \gamma \max_{a'} Q(s', a') \right)}_{\text{target}} - \underbrace{Q(s, a)}_{\text{predicted}} \right\|^2 \right]$$

DQN Atari Results



DQN Atari Results



Downsides of Q-learning

Complexity:

- Can model scenarios where the action space is discrete and small
- Cannot handle continuous action spaces

IMPORTANT:
Imagine you want to predict
steering wheel angle of a car!

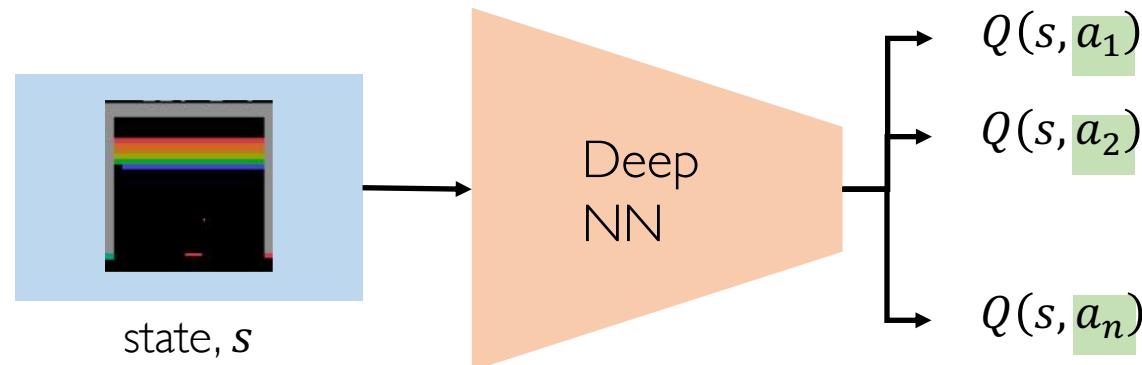
Flexibility:

- Cannot learn stochastic policies since policy is deterministically computed from the Q function

**To overcome, consider a new class of RL training algorithms:
Policy gradient methods**

Policy Gradient (PG) : Key Idea

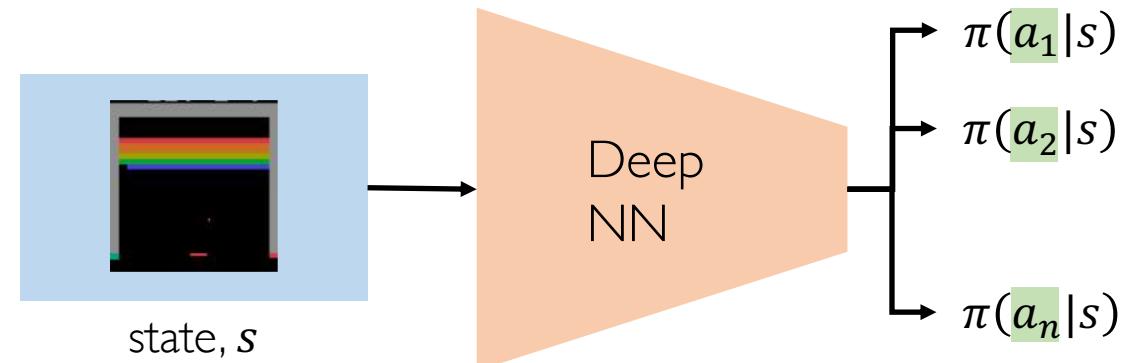
DQN (before): Approximating Q and inferring the optimal policy,



Policy Gradient (PG): Key Idea

DQN (before): Approximating Q and inferring the optimal policy,

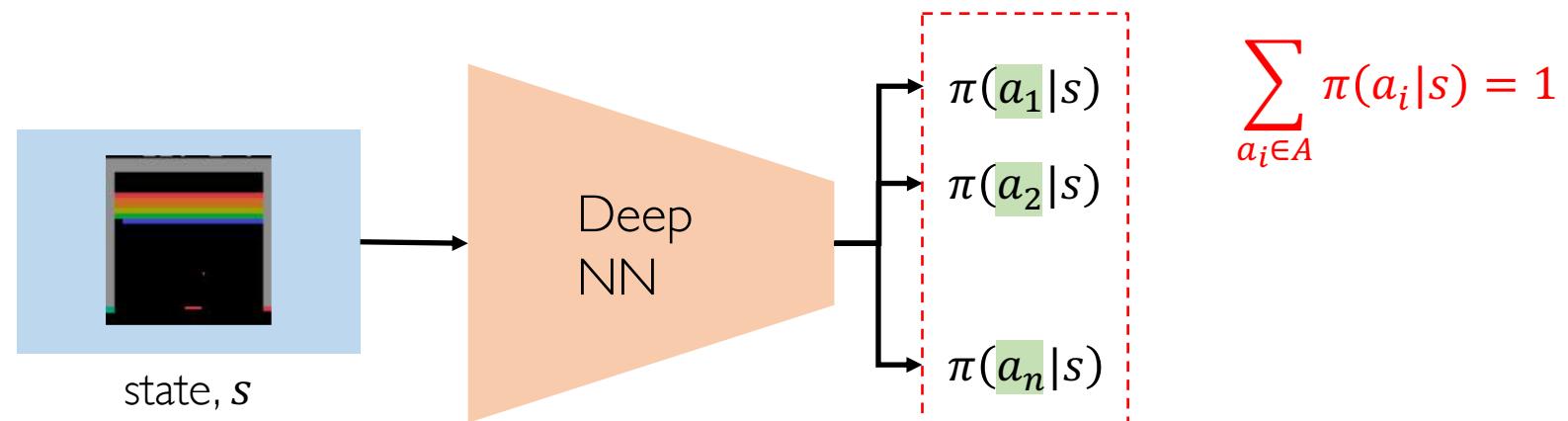
Policy Gradient: Directly optimize the policy!



Policy Gradient (PG): Key Idea

DQN (before): Approximating Q and inferring the optimal policy,

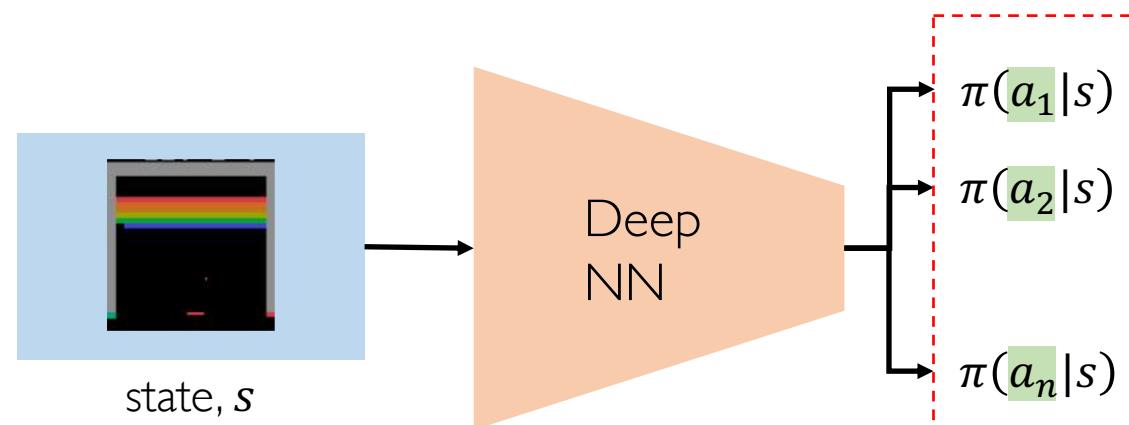
Policy Gradient: Directly optimize the policy!



Policy Gradient (PG): Key Idea

DQN (before): Approximating Q and inferring the optimal policy,

Policy Gradient: Directly optimize the policy!



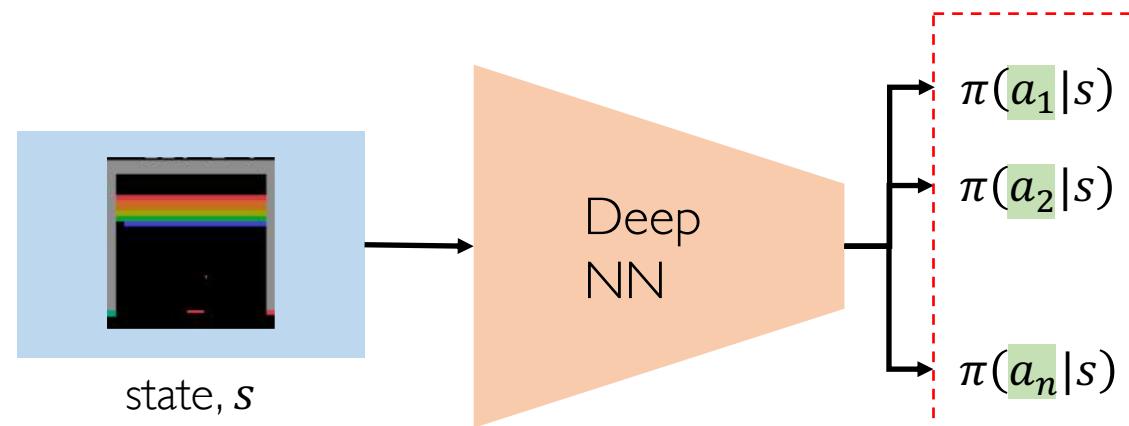
$$\sum_{a_i \in A} \pi(a_i|s) = 1$$

$$\pi(a|s) = P(\text{action}|s)$$

Policy Gradient (PG): Key Idea

DQN (before): Approximating Q and inferring the optimal policy,

Policy Gradient: Directly optimize the policy!

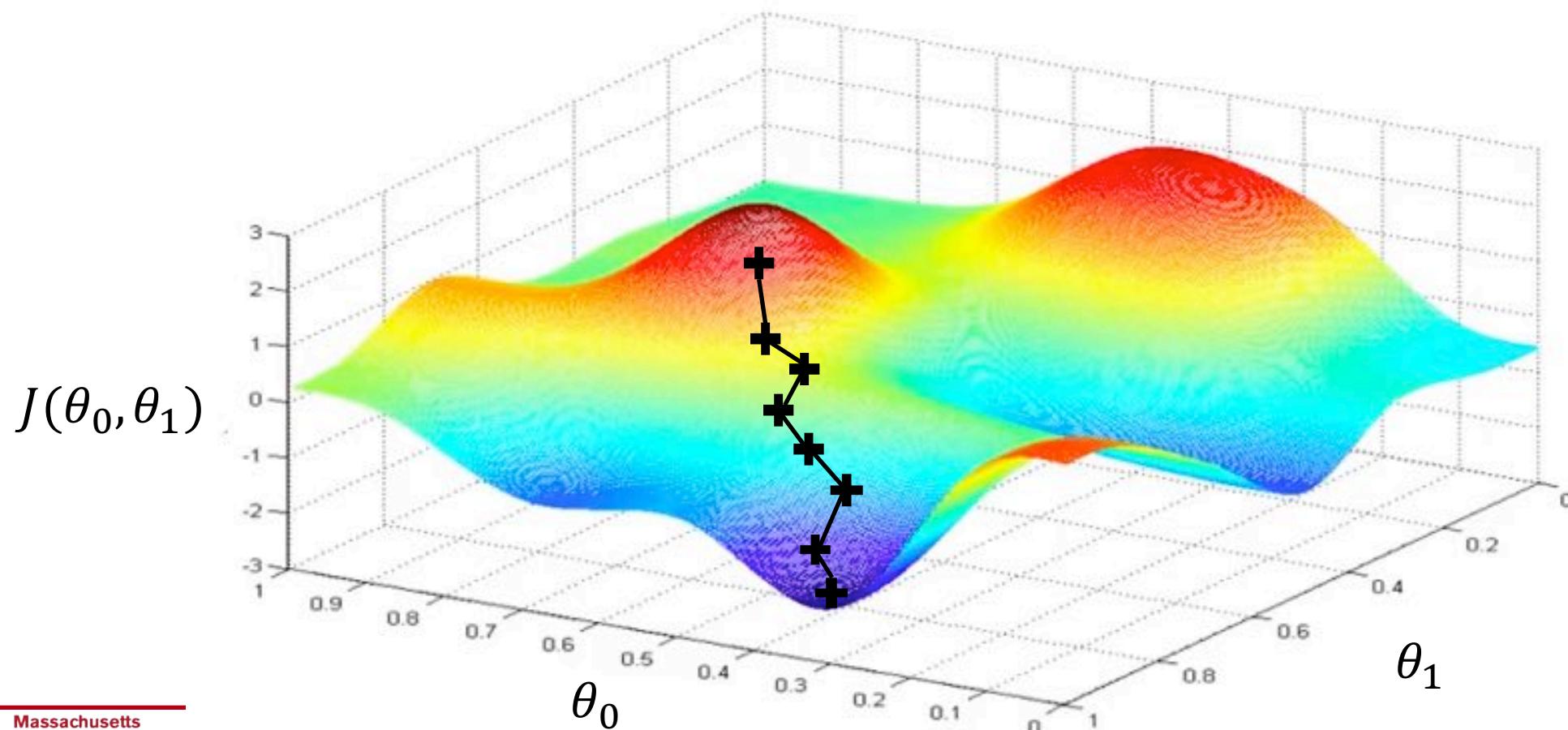


$$\sum_{a_i \in A} \pi(a_i|s) = 1$$

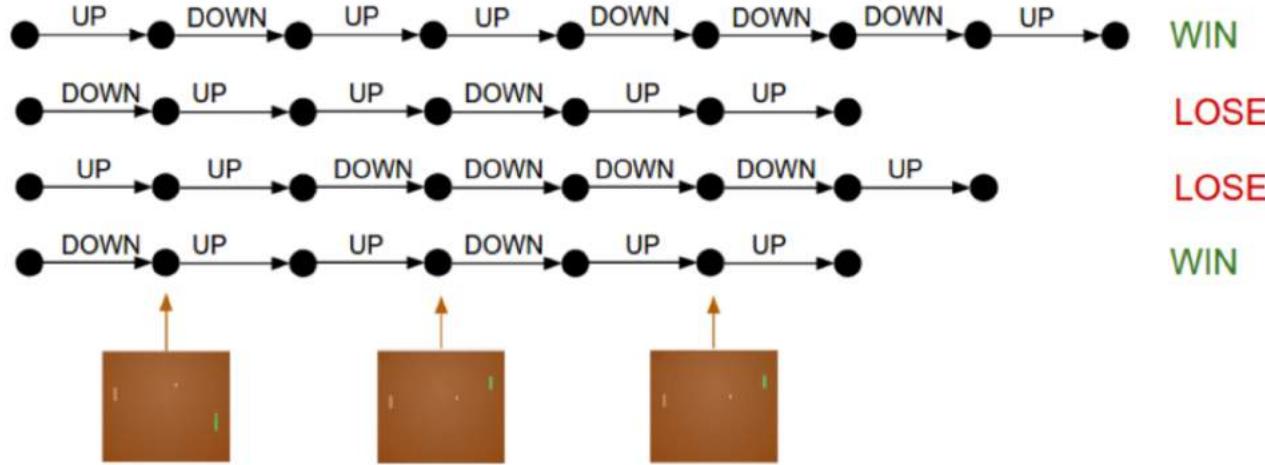
$$\pi(a|s) = P(\text{action}|s)$$

Gradient Descent in Deep Neural Network

Repeat until convergence



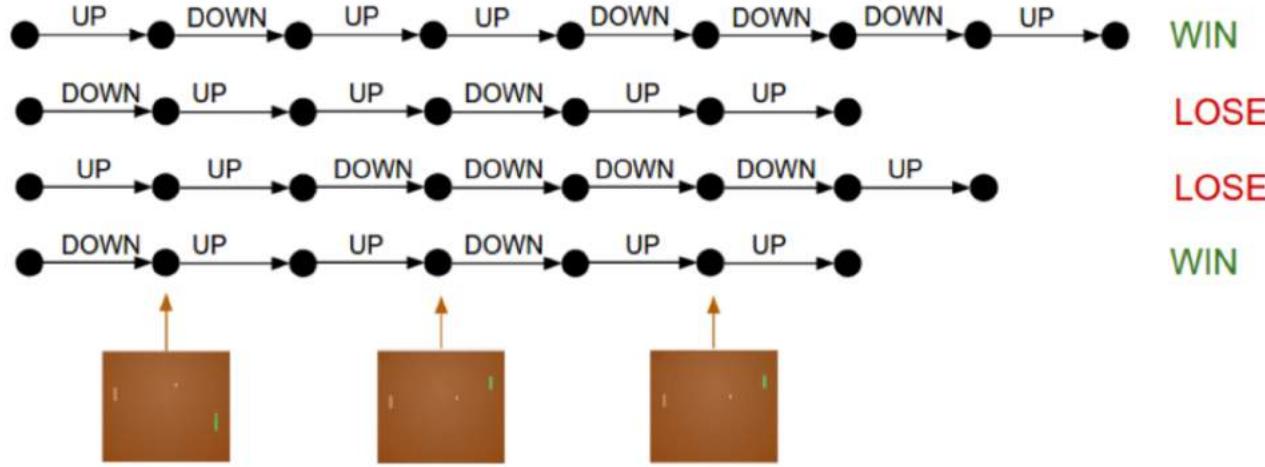
Policy Gradient (PG): Training



1. Run a policy for a while
2. Increase probability of actions that lead to high rewards
3. Decrease probability of actions that lead to low/no rewards

```
function REINFORCE
    Initialize  $\theta$ 
    for episode  $\sim \pi_\theta$ 
         $\{s_i, a_i, r_i\}_{i=1}^{T-1} \leftarrow \text{episode}$ 
        for  $t = 1$  to  $T-1$ 
             $\nabla \leftarrow \nabla_\theta \log \pi_\theta(a_t | s_t) R_t$ 
             $\theta \leftarrow \theta + \alpha \nabla$ 
    return  $\theta$ 
```

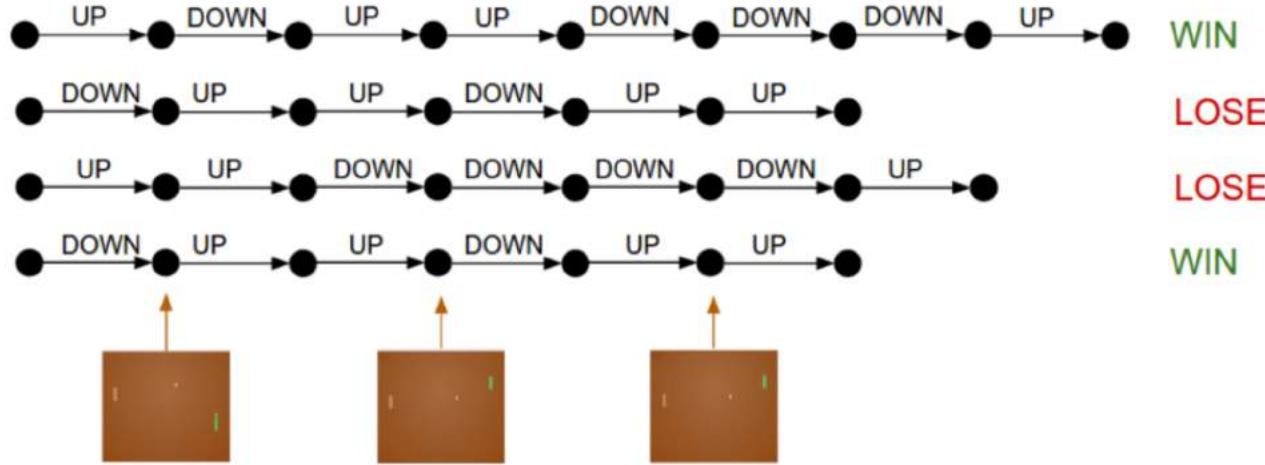
Policy Gradient (PG): Training



1. Run a policy for a while
2. Increase probability of actions that lead to high rewards
3. Decrease probability of actions that lead to low/no rewards

```
function REINFORCE
    Initialize  $\theta$ 
    for episode  $\sim \pi_\theta$ 
         $\{s_i, a_i, r_i\}_{i=1}^{T-1} \leftarrow \text{episode}$ 
        for  $t = 1$  to  $T-1$ 
             $\nabla \leftarrow \nabla_\theta \log \pi_\theta(a_t | s_t) R_t$ 
             $\theta \leftarrow \theta + \alpha \nabla$ 
    return  $\theta$ 
```

Policy Gradient (PG): Training



1. Run a policy for a while
2. Increase probability of actions that lead to high rewards
3. Decrease probability of actions that lead to low/no rewards

Calculate derivative to obtain the direction that gives a higher log-likelihood of a_t action at s_t

```
function REINFORCE
    Initialize  $\theta$ 
    for episode  $\sim \pi_\theta$ 
         $\{s_i, a_i, r_i\}_{i=1}^{T-1} \leftarrow \text{episode}$ 
        for  $t = 1$  to  $T-1$ 
             $\nabla \leftarrow \nabla_\theta \log \pi_\theta(a_t | s_t) R_t$ 
             $\theta \leftarrow \theta + \alpha \nabla$ 
    return  $\theta$ 
```

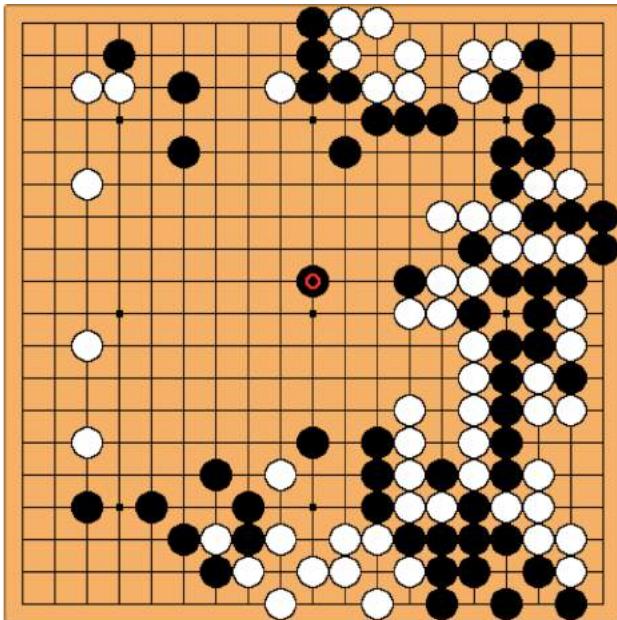
log-likelihood of action

$\nabla_\theta \log \pi_\theta(a_t | s_t) R_t$

reward

The Game of Go

Aim: Get more board territory than your opponent.



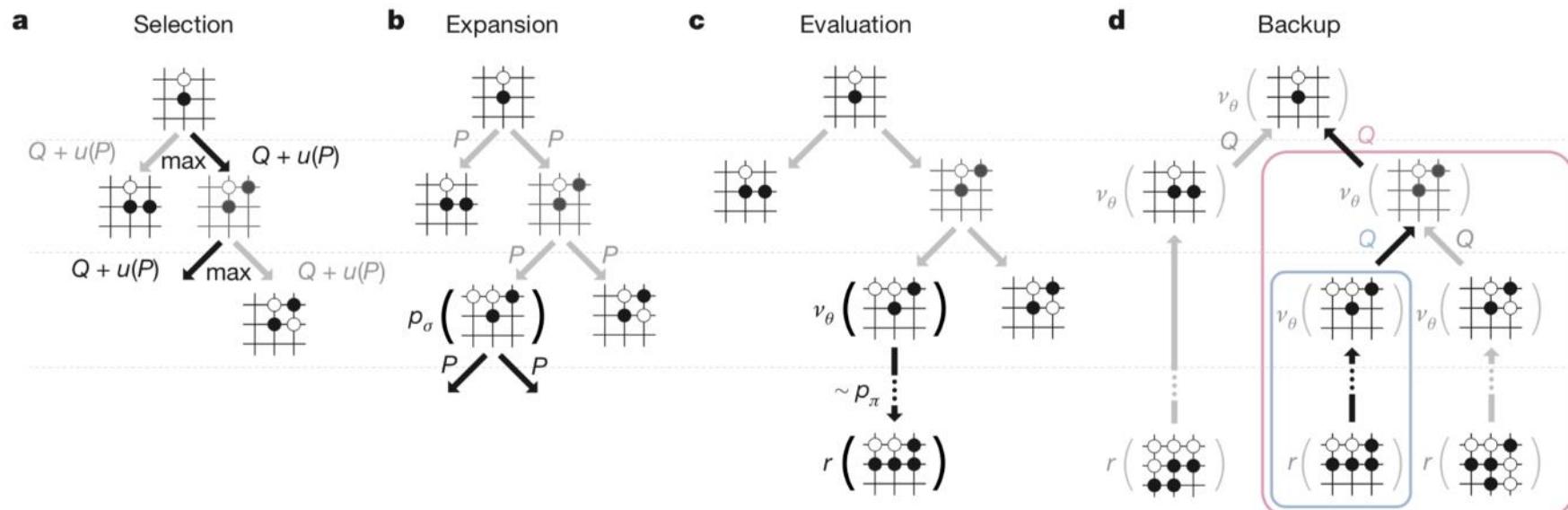
Board Size $n \times n$	Positions 3^{n^2}	% Legal	Legal Positions
1×1	3	33.33%	1
2×2	81	70.37%	57
3×3	19,683	64.40%	12,675
4×4	43,046,721	56.49%	24,318,165
5×5	847,288,609,443	48.90%	414,295,148,741
9×9	$4.434264882 \times 10^{38}$	23.44%	$1.03919148791 \times 10^{38}$
13×13	$4.300233593 \times 10^{80}$	8.66%	$3.72497923077 \times 10^{79}$
19×19	$1.740896506 \times 10^{172}$	1.20%	$2.08168199382 \times 10^{170}$

Greater number of legal board positions than atoms in the universe.

Source: Wikipedia.

AlphaGo Approach

- Monte Carlo Tree Search (MCTS)
 - Balance exploitation/exploration (going deep on promising positions or exploring new underplayed positions)
- Use a neural network as “intuition” for which positions to expand as part of MCTS



Upper confidence bound (UCB)

Pick each node with probability proportional to:

$$v_i + C \times \sqrt{\frac{\ln(N)}{n_i}}$$

Diagram illustrating the UCB formula components:

- v_i (value estimate) is highlighted in blue.
- C (tunable parameter) is highlighted in green.
- $\sqrt{\frac{\ln(N)}{n_i}}$ is highlighted in red.
- n_i (number of visits) is highlighted in purple.

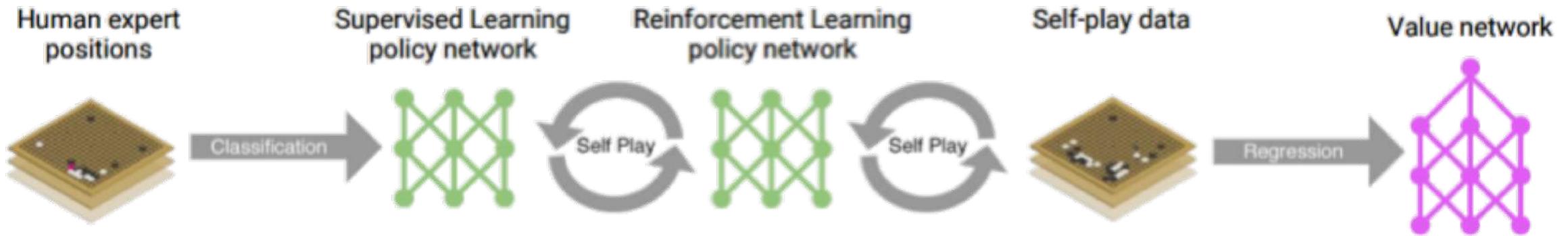
- probability is decreasing in the number of visits (explore)
- probability is increasing in a node's value (exploit)
- always tries every option once

UCB1 Formula (Auer et al 2002)

- Name UCB stands for Upper Confidence Bound
- Policy:
 1. First, try each arm once
 2. Then, at each time step:
 - choose arm i that maximizes the *UCB1 formula* for the upper confidence bound:

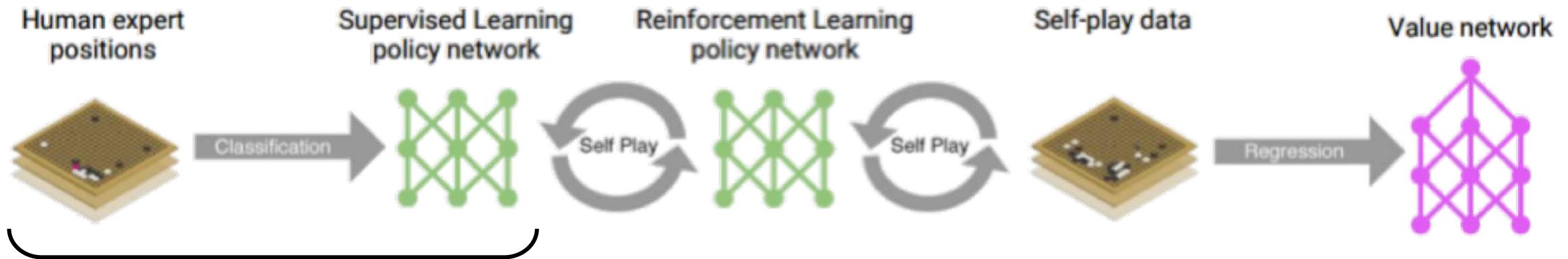
$$\bar{x}_i + \sqrt{\frac{2 \ln(n)}{n_i}}$$

AlphaGo Beats Top Human Player at Go (2016)



Silver et al., Nature 2016.

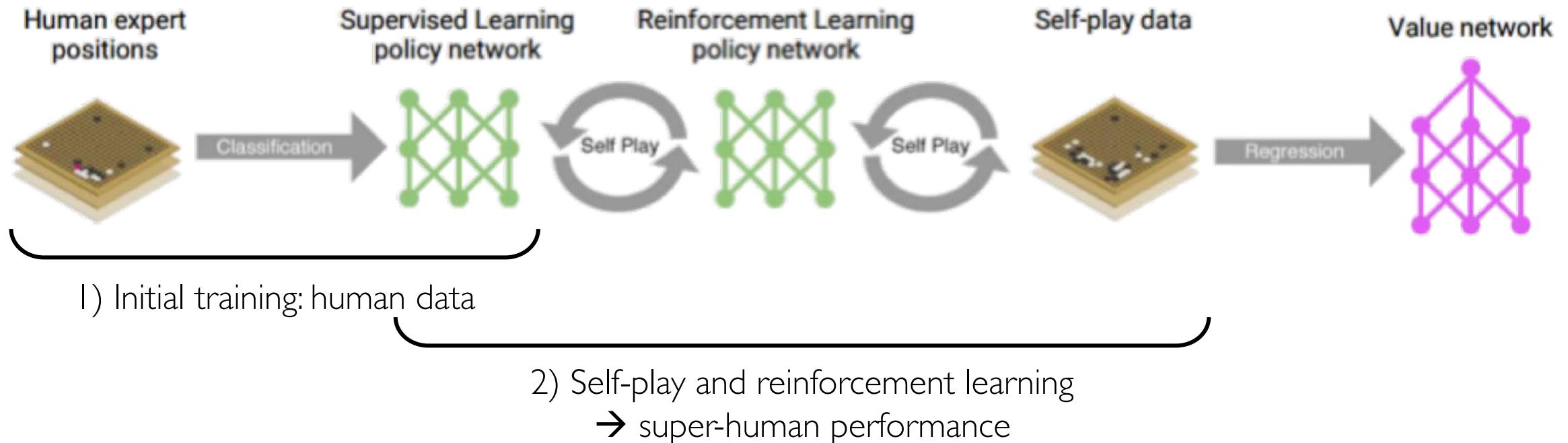
AlphaGo Beats Top Human Player at Go (2016)



I) Initial training: human data

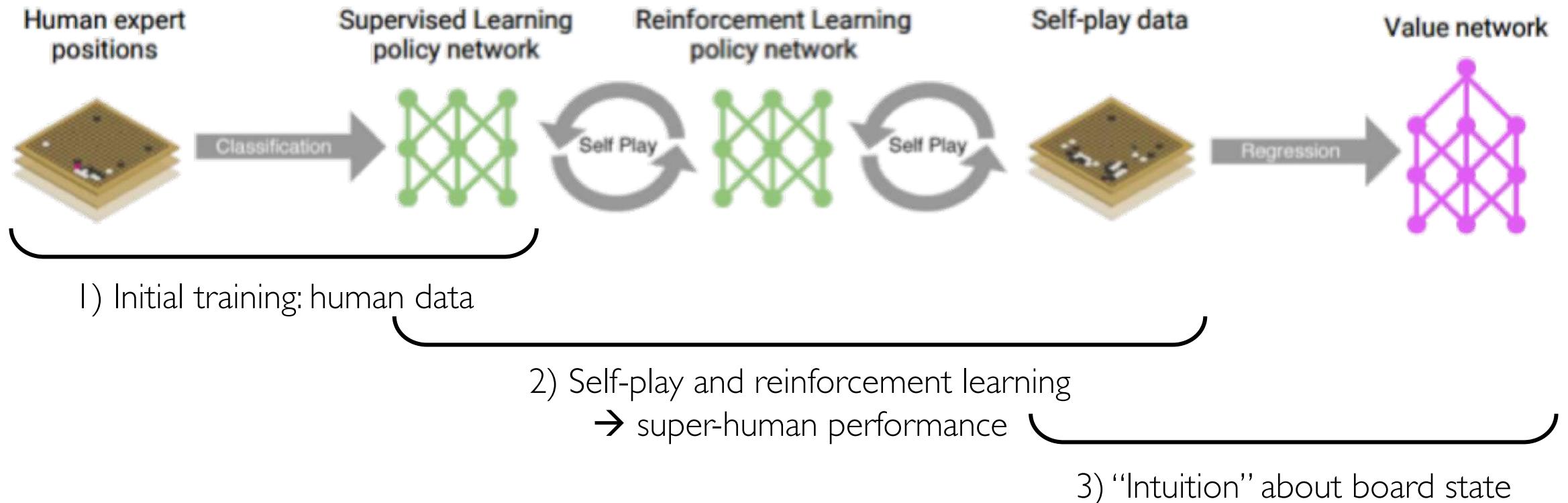
Silver et al., Nature 2016.

AlphaGo Beats Top Human Player at Go (2016)



Silver et al., Nature 2016.

AlphaGo Beats Top Human Player at Go (2016)



Silver et al., Nature 2016.

AlphaGo Beats Top Human Player at Go (2016)



Silver et al., Nature 2016.

Questions?