

# Bipedal walking using deep reinforcement learning and proximal policy optimization

Jhon Charaja<sup>1</sup> and Luca Borgonovi<sup>1</sup>

<sup>1</sup>Universidade de São Paulo, Brasil

December 17, 2021

# Outline

1. Motivation

2. Objective

3. Scopes

4. Methodology

5. Results

6. Conclusions

# Motivation

Bipedal walking is difficult activity to describe due to nonlinear relationships.

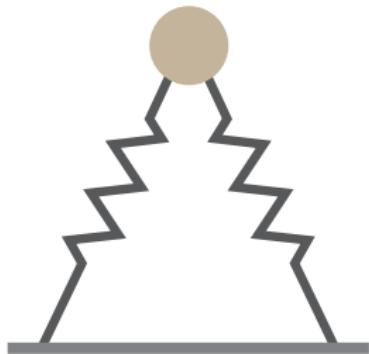


Figure: Spring-loaded inverted pendulum

- Physical representation of legs as springs<sup>1</sup>
- Motion of the trunk with center of mass<sup>1</sup>
- Recent works perform bipedal walking with machine learning methods<sup>2,3</sup>
- Deep reinforcement learning (DRL) does not require mathematical model or control formulation<sup>4</sup>
- DRL with proximal policy optimization is simple and efficient<sup>4</sup>

---

<sup>1</sup>H. Geyer (2006).

<sup>2</sup>T. Haarjona (2019).

<sup>3</sup>T. Li (2019).

<sup>4</sup>J. Schulman (2017)

# Motivation

Bipedal walking is difficult activity to describe due to nonlinear relationships.

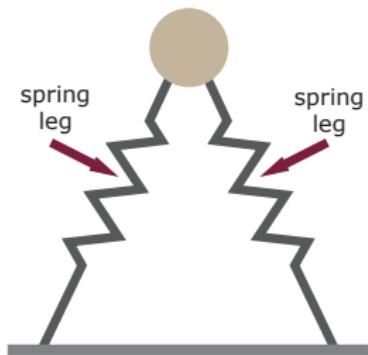


Figure: Spring-loaded inverted pendulum

- Physical representation of legs as springs<sup>1</sup>
- Motion of the trunk with center of mass<sup>1</sup>
- Recent works perform bipedal walking with machine learning methods<sup>2,3</sup>
- Deep reinforcement learning (DRL) does not require mathematical model or control formulation<sup>4</sup>
- DRL with proximal policy optimization is simple and efficient<sup>4</sup>

---

<sup>1</sup>H. Geyer (2006).

<sup>2</sup>T. Haarjona (2019).

<sup>3</sup>T. Li (2019).

<sup>4</sup>J. Schulman (2017)

# Motivation

Bipedal walking is difficult activity to describe due to nonlinear relationships.

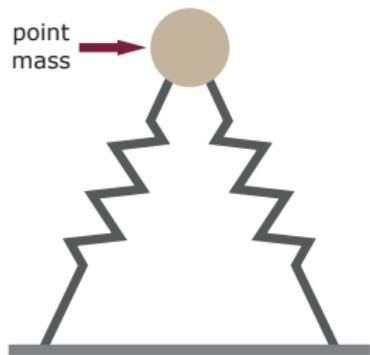


Figure: Spring-loaded inverted pendulum

- Physical representation of legs as springs<sup>1</sup>
- Motion of the trunk with center of mass<sup>1</sup>
- Recent works perform bipedal walking with machine learning methods<sup>2,3</sup>
- Deep reinforcement learning (DRL) does not require mathematical model or control formulation<sup>4</sup>
- DRL with proximal policy optimization is simple and efficient<sup>4</sup>

---

<sup>1</sup>H. Geyer (2006).

<sup>2</sup>T. Haarjona (2019).

<sup>3</sup>T. Li (2019).

<sup>4</sup>J. Schulman (2017)

# Motivation

Bipedal walking is difficult activity to describe due to nonlinear relationships.

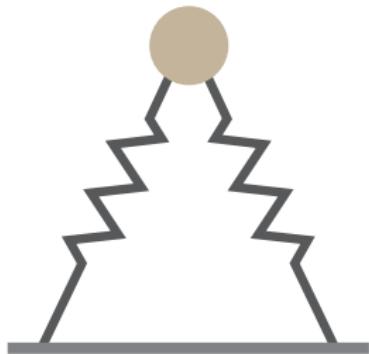


Figure: Spring-loaded inverted pendulum

- Physical representation of legs as springs<sup>1</sup>
- Motion of the trunk with center of mass<sup>1</sup>
- Recent works perform bipedal walking with machine learning methods<sup>2,3</sup>
- Deep reinforcement learning (DRL) does not require mathematical model or control formulation<sup>4</sup>
- DRL with proximal policy optimization is simple and efficient<sup>4</sup>

---

<sup>1</sup>H. Geyer (2006).

<sup>2</sup>T. Haarjona (2019).

<sup>3</sup>T. Li (2019).

<sup>4</sup>J. Schulman (2017)

# Outline

1. Motivation

2. Objective

3. Scopes

4. Methodology

5. Results

6. Conclusions

**Perform bipedal walking using deep reinforcement learning with proximal policy optimization algorithm**

# Outline

1. Motivation

2. Objective

3. Scopes

4. Methodology

5. Results

6. Conclusions

# Scopes

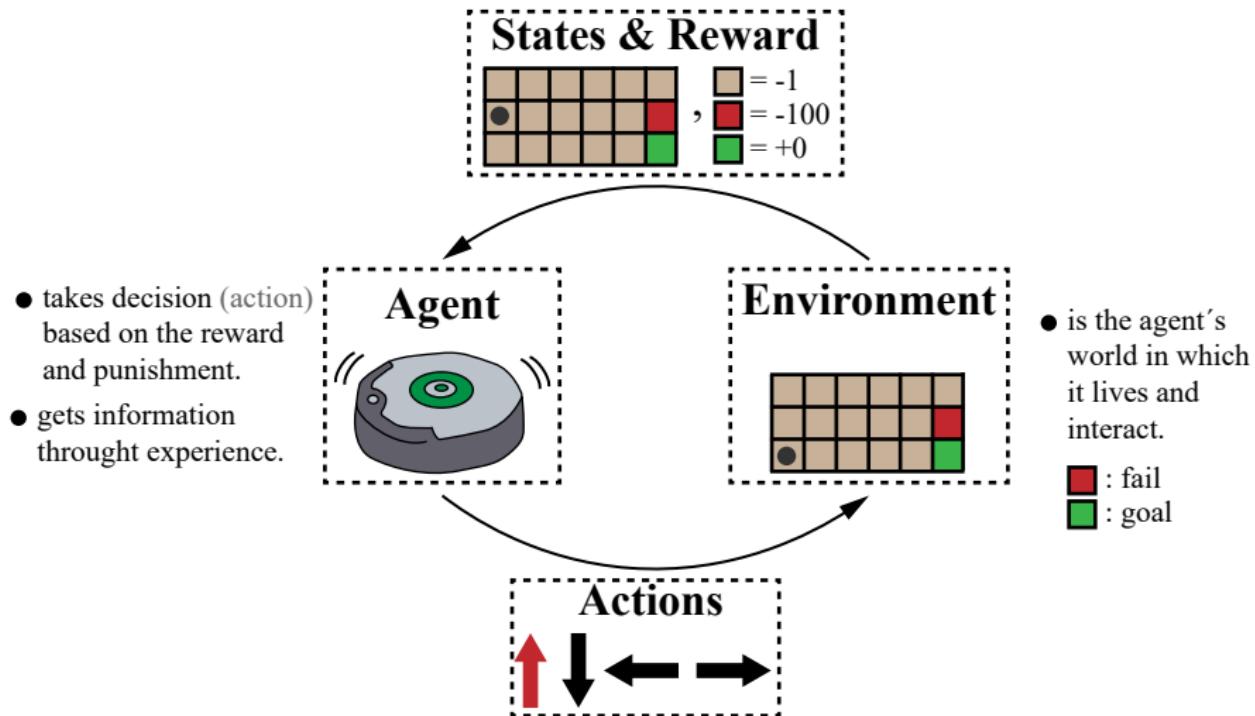
- The algorithm is focused on making the robot walk, but not on controlling the way it walks.
- Likewise, the algorithm is not focused on optimizing energy during walking.

# Outline

1. Motivation
2. Objective
3. Scopes
4. Methodology
5. Results
6. Conclusions

# What is reinforcement learning?

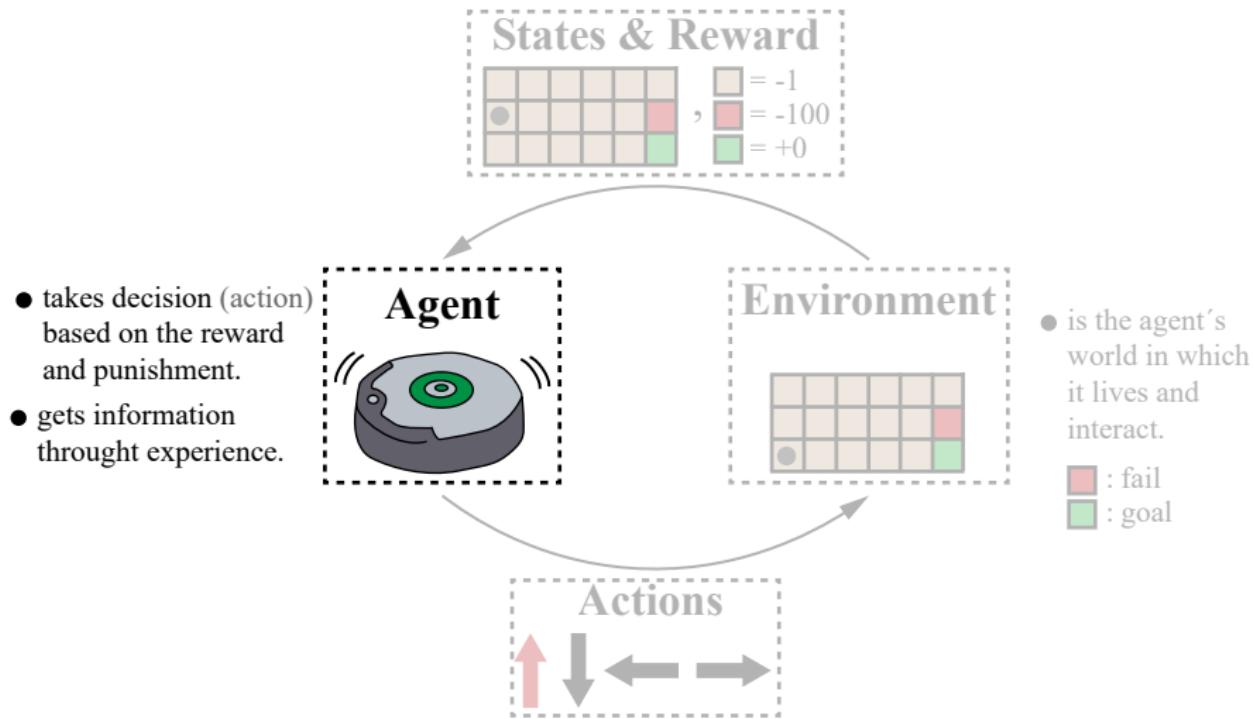
Reinforcement learning is a training technique of machine learning to teach models (agents) to make good (optimal) sequences of decisions under uncertainty<sup>1</sup>.



<sup>1</sup>R. S. Sutton (2018).

# What is reinforcement learning?

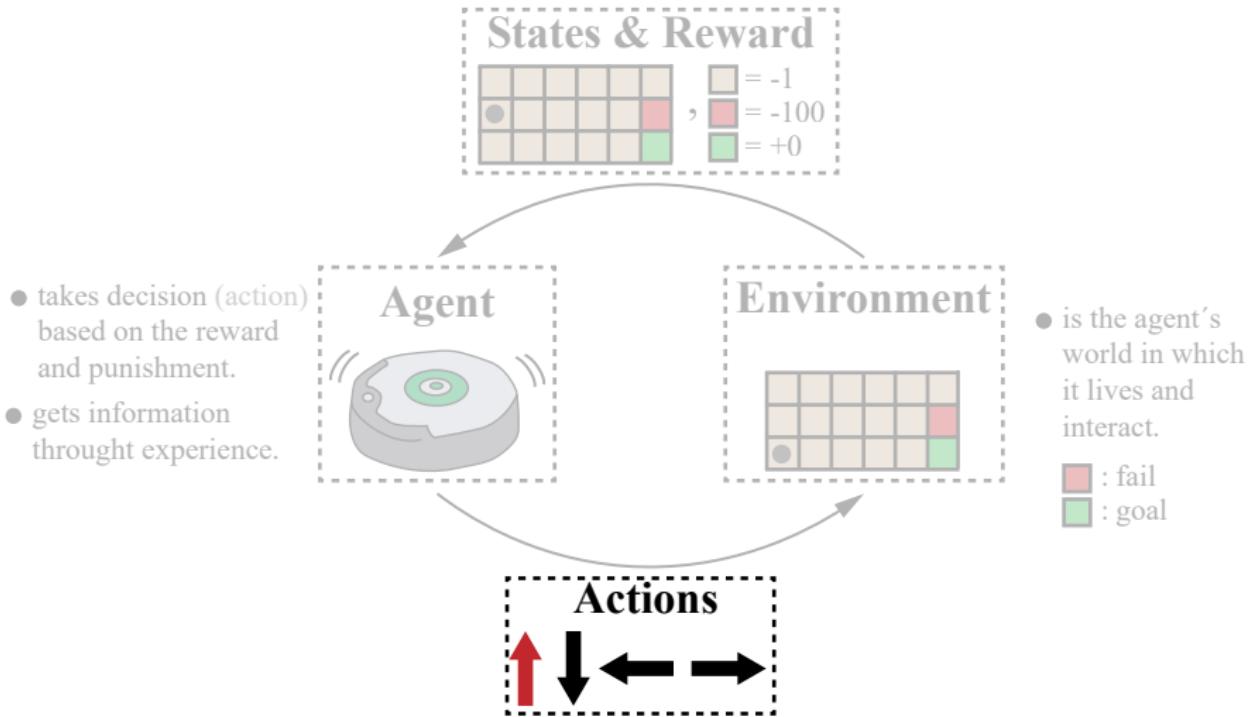
Reinforcement learning is a training technique of machine learning to teach models (agents) to make good (optimal) sequences of decisions under uncertainty<sup>1</sup>.



<sup>1</sup>R. S. Sutton (2018).

# What is reinforcement learning?

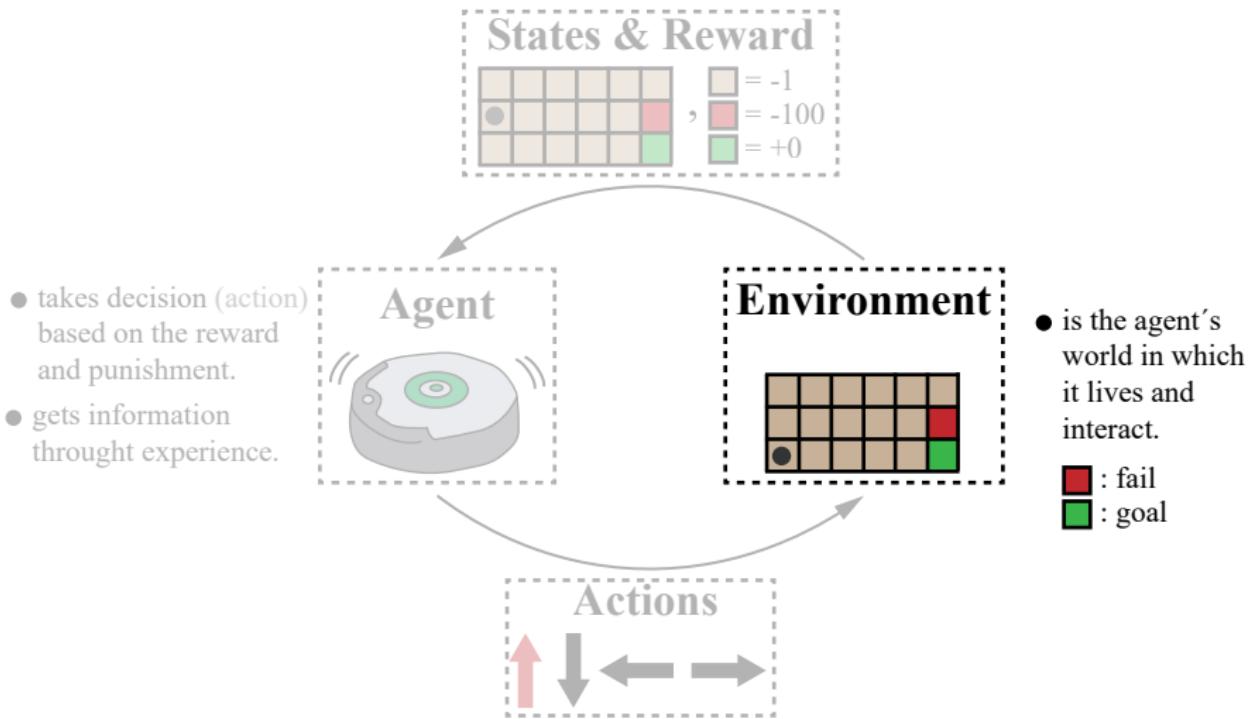
Reinforcement learning is a training technique of machine learning to teach models (agents) to make good (optimal) sequences of decisions under uncertainty<sup>1</sup>.



<sup>1</sup>R. S. Sutton (2018).

# What is reinforcement learning?

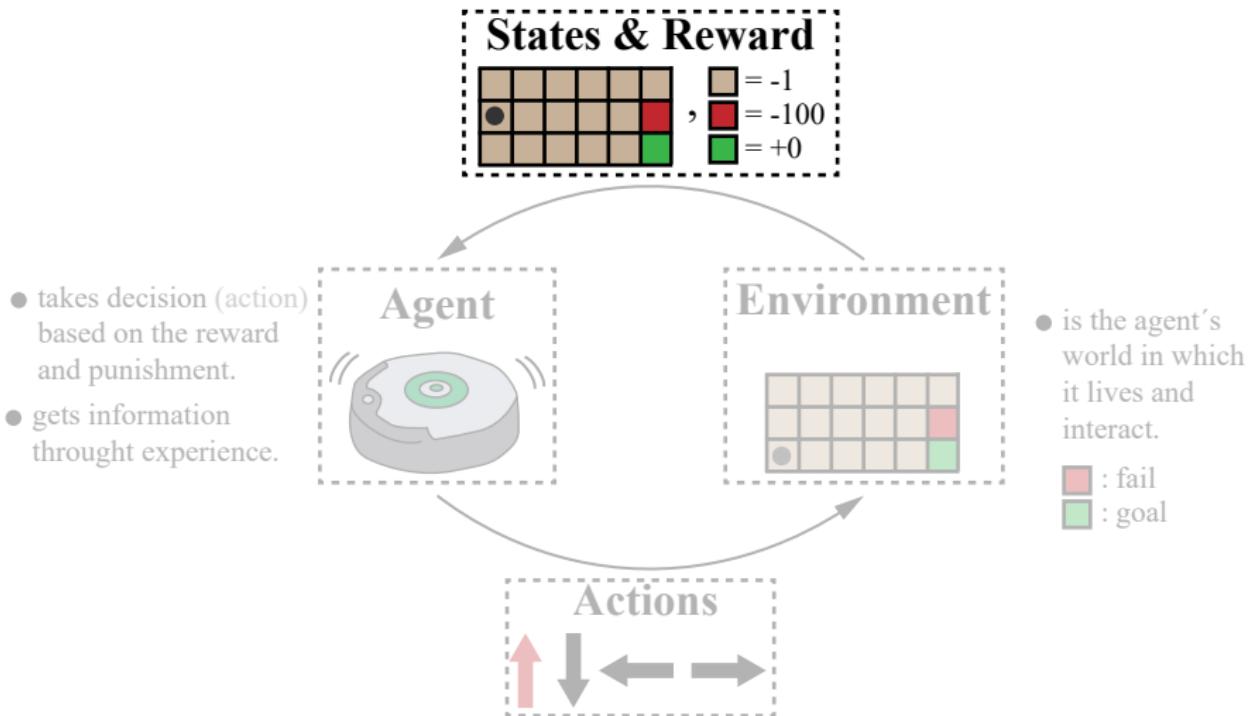
Reinforcement learning is a training technique of machine learning to teach models (agents) to make good (optimal) sequences of decisions under uncertainty<sup>1</sup>.



<sup>1</sup>R. S. Sutton (2018).

# What is reinforcement learning?

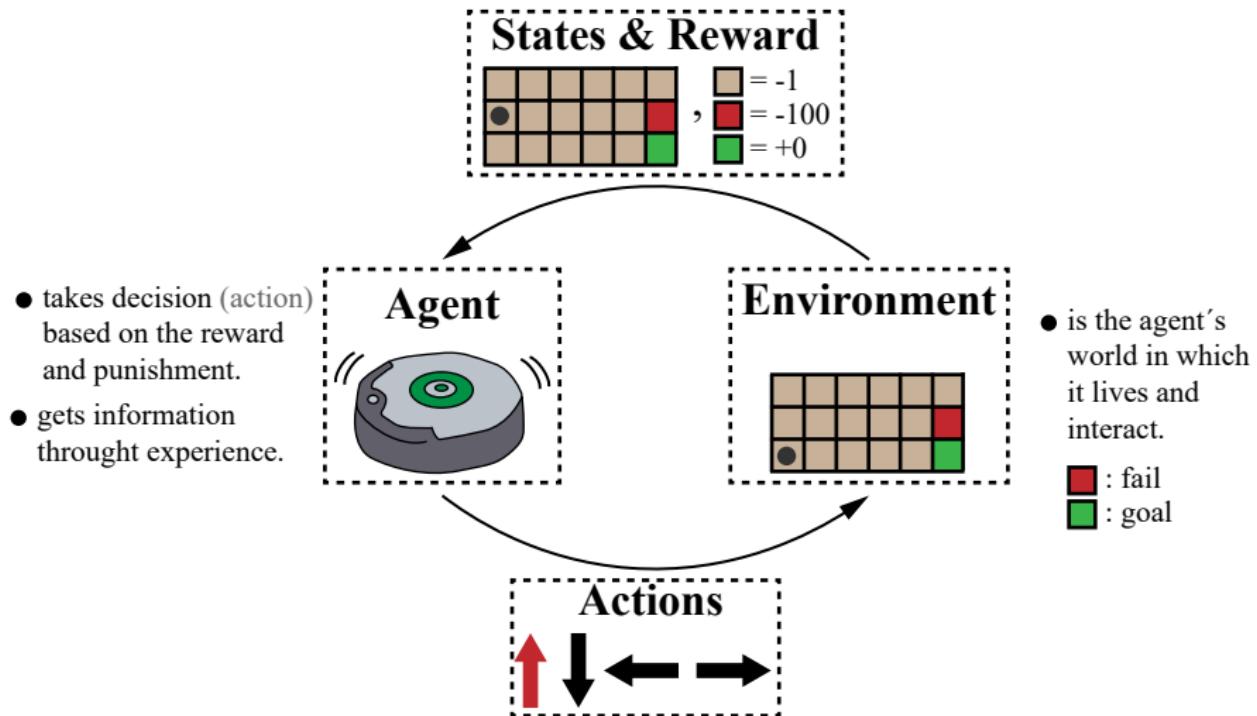
Reinforcement learning is a training technique of machine learning to teach models (agents) to make good (optimal) sequences of decisions under uncertainty<sup>1</sup>.



<sup>1</sup>R. S. Sutton (2018).

# What is reinforcement learning?

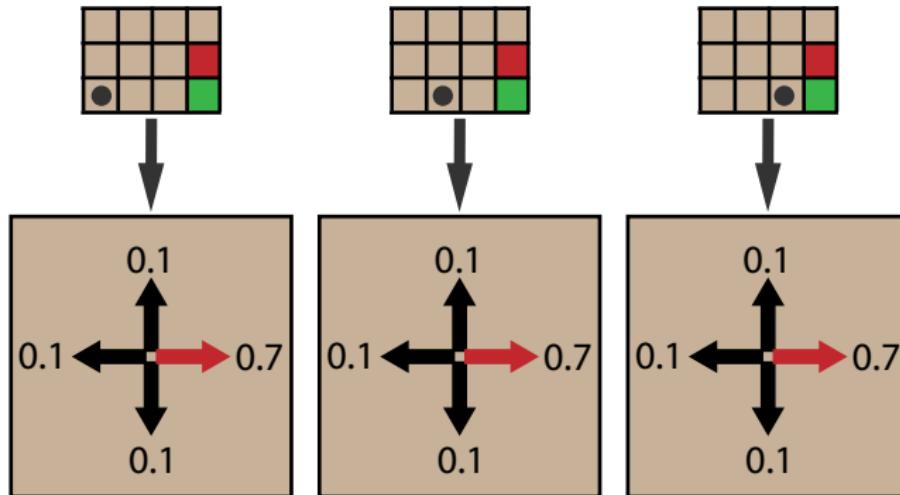
Reinforcement learning is a training technique of machine learning to teach models (agents) to make good (optimal) sequences of decisions under uncertainty<sup>1</sup>.



<sup>1</sup>R. S. Sutton (2018).

# Policy

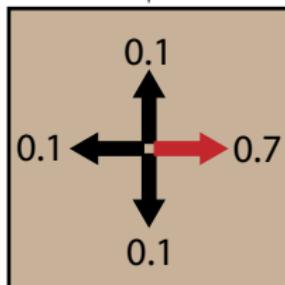
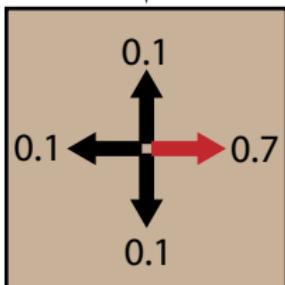
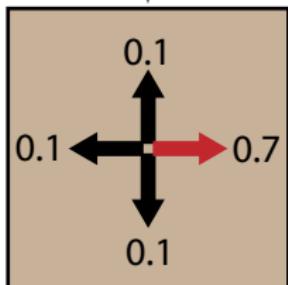
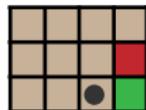
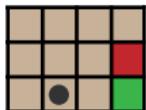
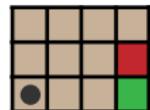
A policy ( $\pi$ ) indicates the decision (action) that the agent should take as a function of the agent's state to accomplish the task<sup>1</sup>.



<sup>1</sup>R. S. Sutton (2018).

# Policy

A policy ( $\pi$ ) indicates the decision (action) that the agent should take as a function of the agent's state to accomplish the task<sup>1</sup>.



The objective is find the optimal policy ( $\pi$ ) that maxime the reward

How quantify the performance of a policy ( $\pi$ )?

<sup>1</sup>R. S. Sutton (2018).

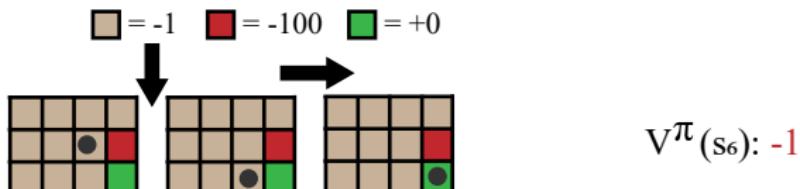
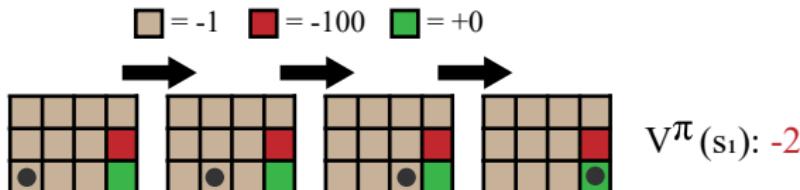
# Policy evaluation: value function

The value function ( $V^\pi(s)$ ) indicates the final return from being in a state when following a particular policy ( $\pi$ )<sup>1</sup>. It is given by

$$V^\pi(s_t = s) = E_\pi [r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots | s_t = s],$$

where  $\gamma$  is a discount factor and  $s_t$  is initial state.

Policy ( $\pi$ ): Considering:  $\gamma=1$



<sup>1</sup>R. S. Sutton (2018).

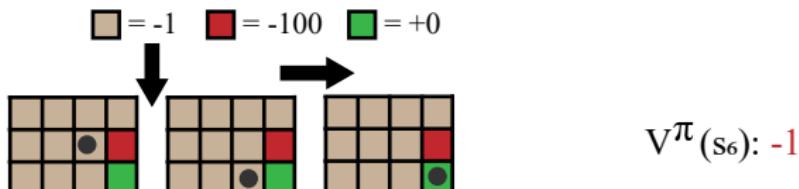
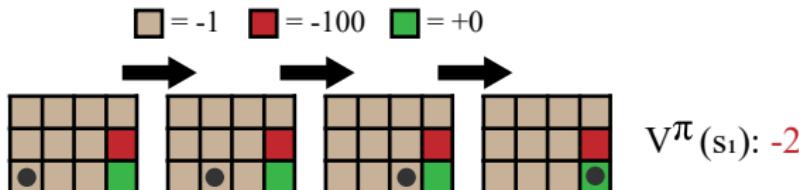
# Policy evaluation: value function

The value function ( $V^\pi(s)$ ) indicates the final return from being in a state when following a particular policy ( $\pi$ )<sup>1</sup>. It is given by

$$V^\pi(s_t = s) = E_\pi [r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots | s_t = s],$$

where  $\gamma$  is a discount factor and  $s_t$  is initial state.

Policy ( $\pi$ ): Considering:  $\gamma=1$

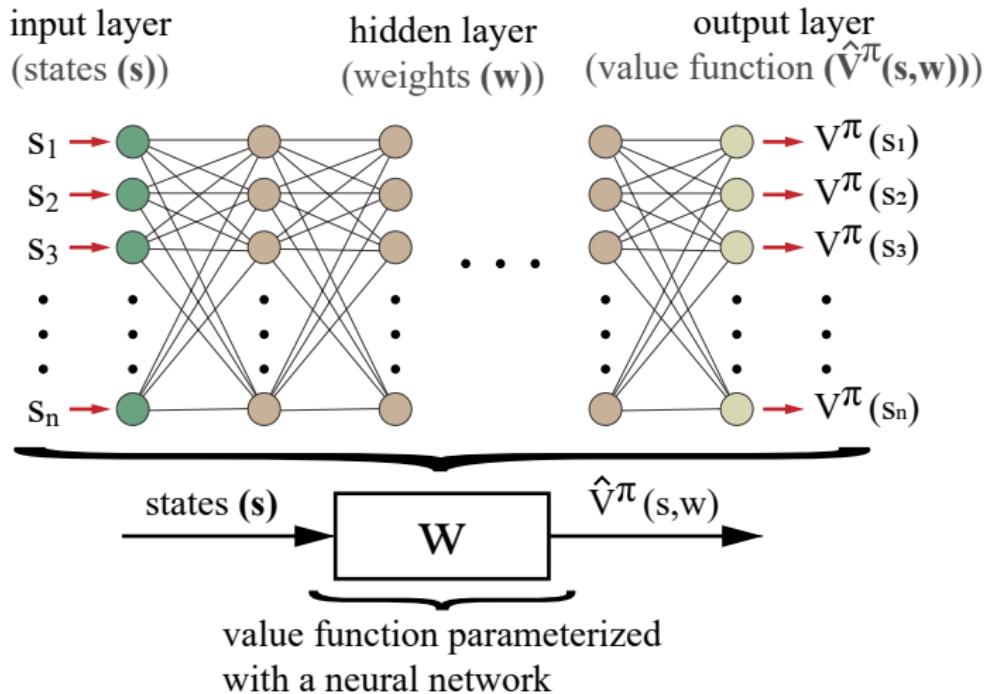


- $V^\pi(s)$  depends of policy ( $\pi$ ) and initial state ( $s_t$ ).
- Agent needs to start in  $(s_{3,4,5,7,8,9,10})$  to know its value  $V^\pi(s)$ .

<sup>1</sup>R. S. Sutton (2018).

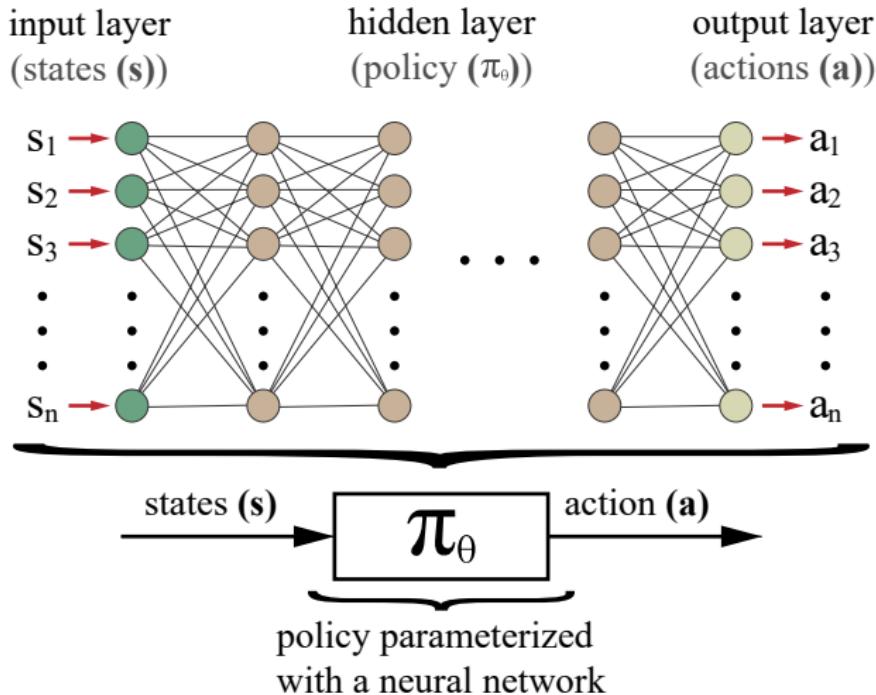
# Value function approximation

The value function could be approximated by a neural network



# Policy and neural networks

The policy could be represented by a neural network



# Policy gradient method

This method is focused on modify the neural network parameters ( $\theta$ ) to get a optimal (local) policy<sup>1</sup>.

The objective function (reward) is formulated as<sup>1</sup>

$$L(\theta) = \hat{\mathbb{E}}_t \left[ \log \pi_\theta(a|s) \hat{A}_t \right],$$

with,

$$\hat{A}_t = V_t^{\pi_\theta} - \hat{V}_t,$$

where,

- $\hat{A}_t$       advantage function
- $V_t^{\pi_\theta}$       final reward following policy ( $\pi_\theta$ )
- $\hat{V}_t$       expected final reward (baseline)

<sup>1</sup>R. S. Sutton (2018).

# Proximal policy optimization

PPO is one of the best algorithms for reinforcement learning due to simplicity and high performance<sup>1</sup>.

The main objective function (reward) is given by

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[ \min \left( r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right],$$

with,

$$r_t(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta,\text{old}}(a|s)},$$

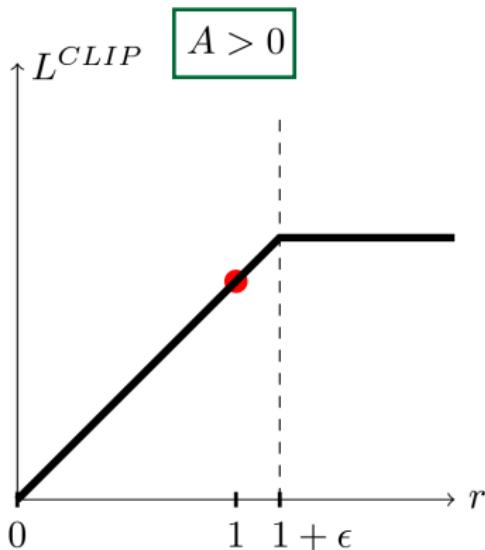
where,

- $r_t(\theta)$  probability ratio between policies
- $\epsilon$  hyperparameter

<sup>1</sup>J. Schulman (2017).

# Proximal policy optimization

the new action yielded  
**better** than expected return



the new action yielded  
**worse** than expected return

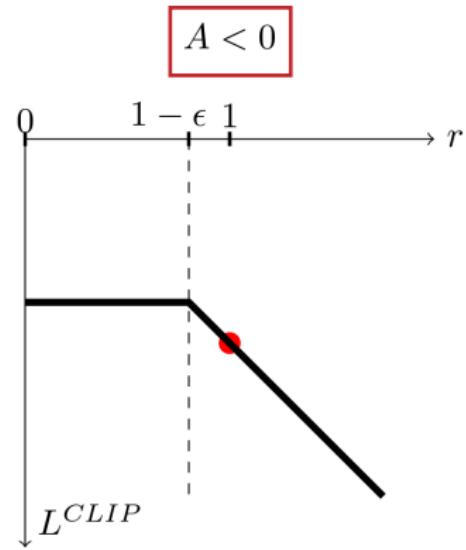


Figure: Image adapted from J. Schulman (2017)

# Proximal policy optimization

The final objective function (reward and exploration) is given by<sup>1</sup>

$$L^{\text{PPO}}(\theta) = \hat{\mathbb{E}}_t [L^{\text{CLIP}}(\theta) - c_1 L^{\text{VF}}(\theta) + c_2 S[\pi_\theta](s_t)] ,$$

with,

$$L^{\text{VF}}(\theta) = (V_\theta(s_t) - V_t^{\text{target}})^2 ,$$

where,

- $c_1, c_2$  weighting coefficients
- $S$  entropy bonus (exploration)
- $L^{\text{VF}}(\theta)$  square-error loss
- $V_t^{\text{target}}$  objective final reward

<sup>1</sup>J. Schulman (2017).

## MuJoCo

Advanced physics simulation

## OpenAI



## TensorFlow

# Bipedal robot: algorithm configuration

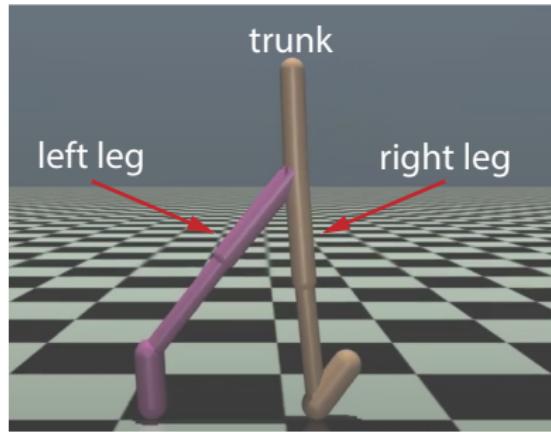


Figure: Bipedal robot in simulation environment

## States

- position: 6 (left, right) + 1 (trunk)
- velocity: 6 (left, right) + 1 (trunk)
- total: 14 states

## Reward

- fail when trunk angle  $> 50^\circ$
- +1 for each iteration alive
- + linear velocity

# Outline

1. Motivation

2. Objective

3. Scopes

4. Methodology

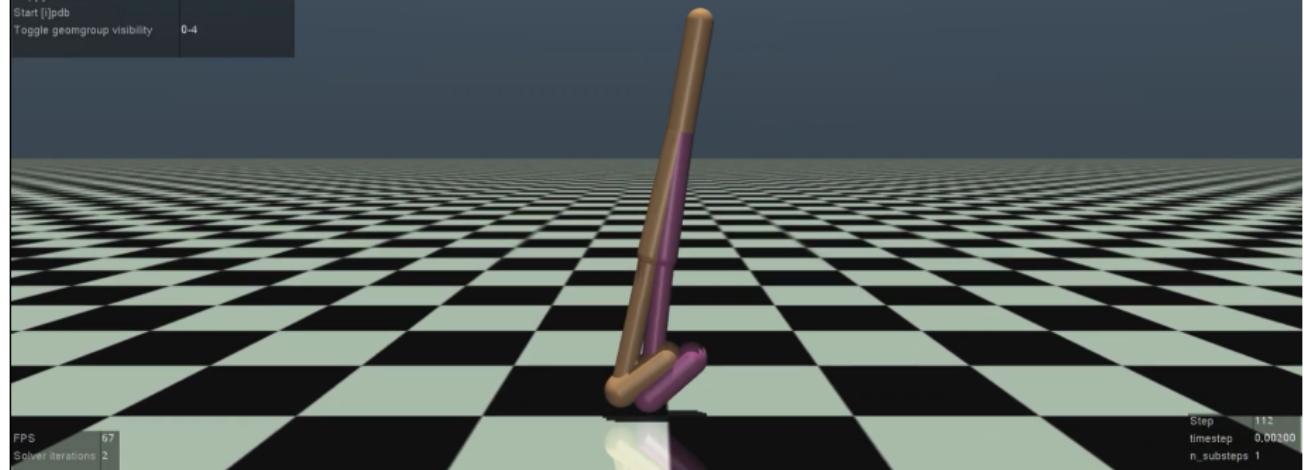
5. Results

6. Conclusions

# Results

Run speed = 0.125 x real time	[S]lower, [F]aster
Render every frame	On
Switch camera (#cams = 2)	[Tab] (camera ID = 0)
{C}ontact forces	On
Referenc[e] frames	On
Tr[ansparent	Off
Display [M]jocap bodies	On
Stop	[Space]
Advance simulation by one step	[Right arrow]
[H]ide Menu	
Record [V]ideo (Off)	
Cap[ture frame	
Start [J]pdb	
Toggle geomgroup visibility	0-4

State: 20M  
10 attempts per state



# Outline

1. Motivation

2. Objective

3. Scopes

4. Methodology

5. Results

6. Conclusions

# Conclusions