Bipedal robot walking using reinforcement learning

Authors: J. Charaja and L. Borgonovi

Jhon Charaja[1] and Luca Borgonovi[1]

[1]Universidade de São Paulo, Brasil

December 9, 2021

Outline

Motivation

Outline

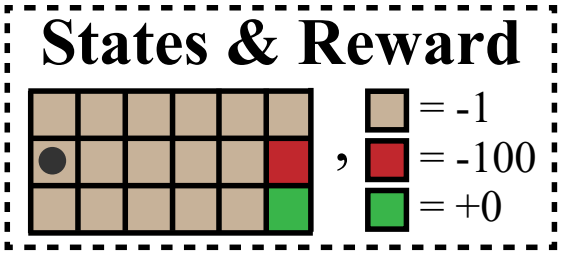
Scopes

- 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.

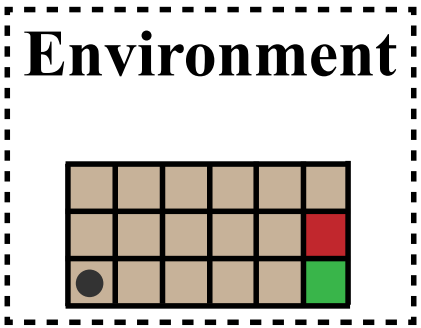
Outline

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 [1].

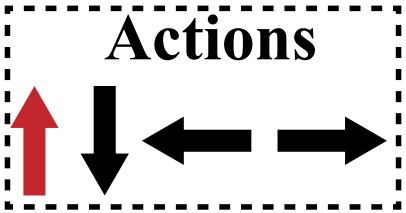


- takes decision (action) based on the reward and punishment.
- gets information throught experience.



- is the agent´s world in which it lives and interact.

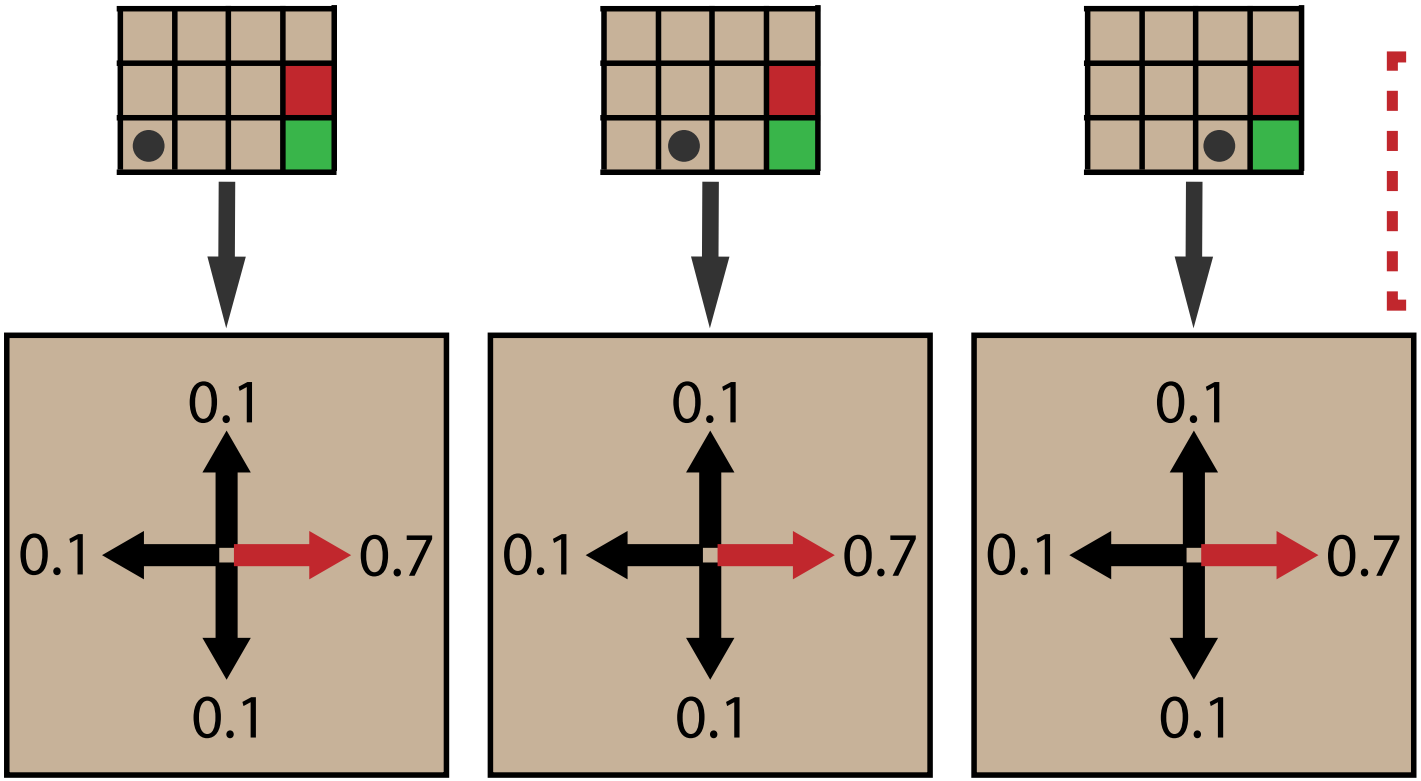
: fail
: goal



[1] R. S. Sutton (2018).

Policy

A policy (π) indicates the decision (action) that the agent should take as a function of the agent's state to accomplish the task[1].



How quantify the performance of a policy (π)?

[1] R. S. Sutton (2018).

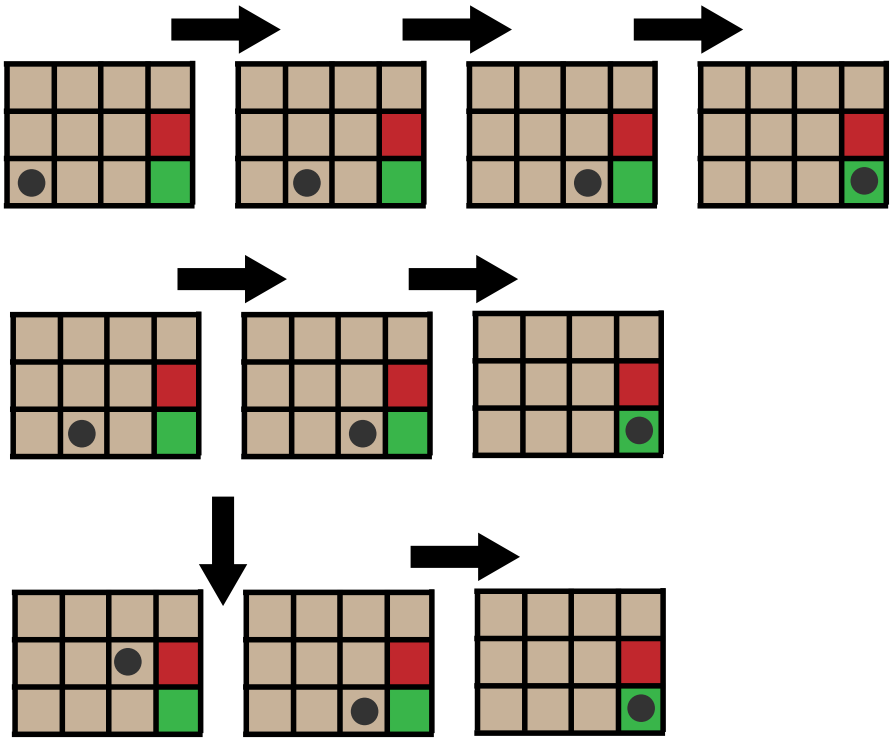
Policy evaluation: value function

The value function ($V^{\pi}(s)$) indicates the final return from being in a state when following a particular policy (π)[1]. It is given by

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

where γ is a discount factor and s_t is initial (current) state.

Policy (π): Considering:γ=1



$V^{\pi}(s_1)$: -2

$V^{\pi}(s_2)$: -1

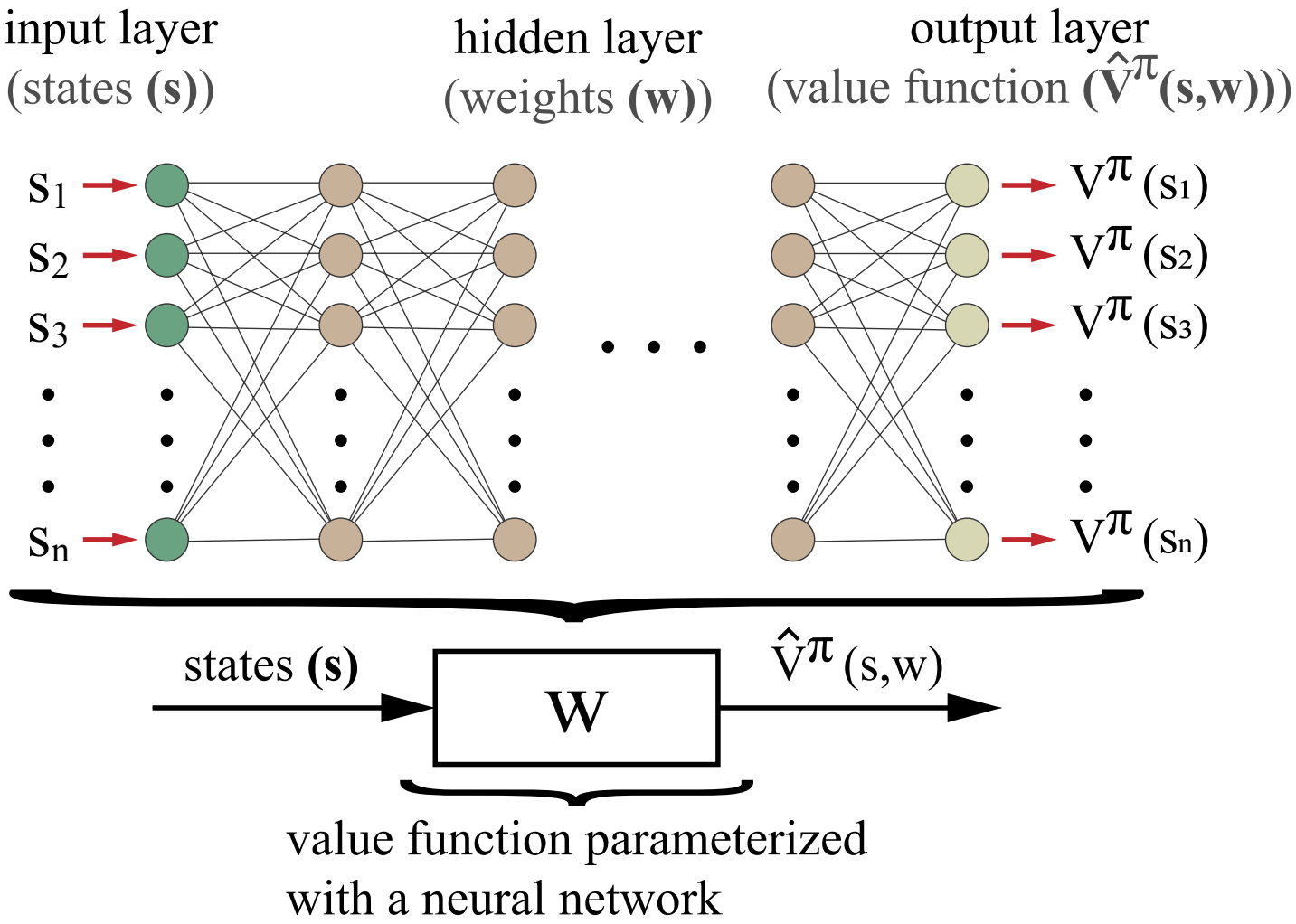
$V^{\pi}(s_6)$: -1

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

[1] R. S. Sutton (2018).

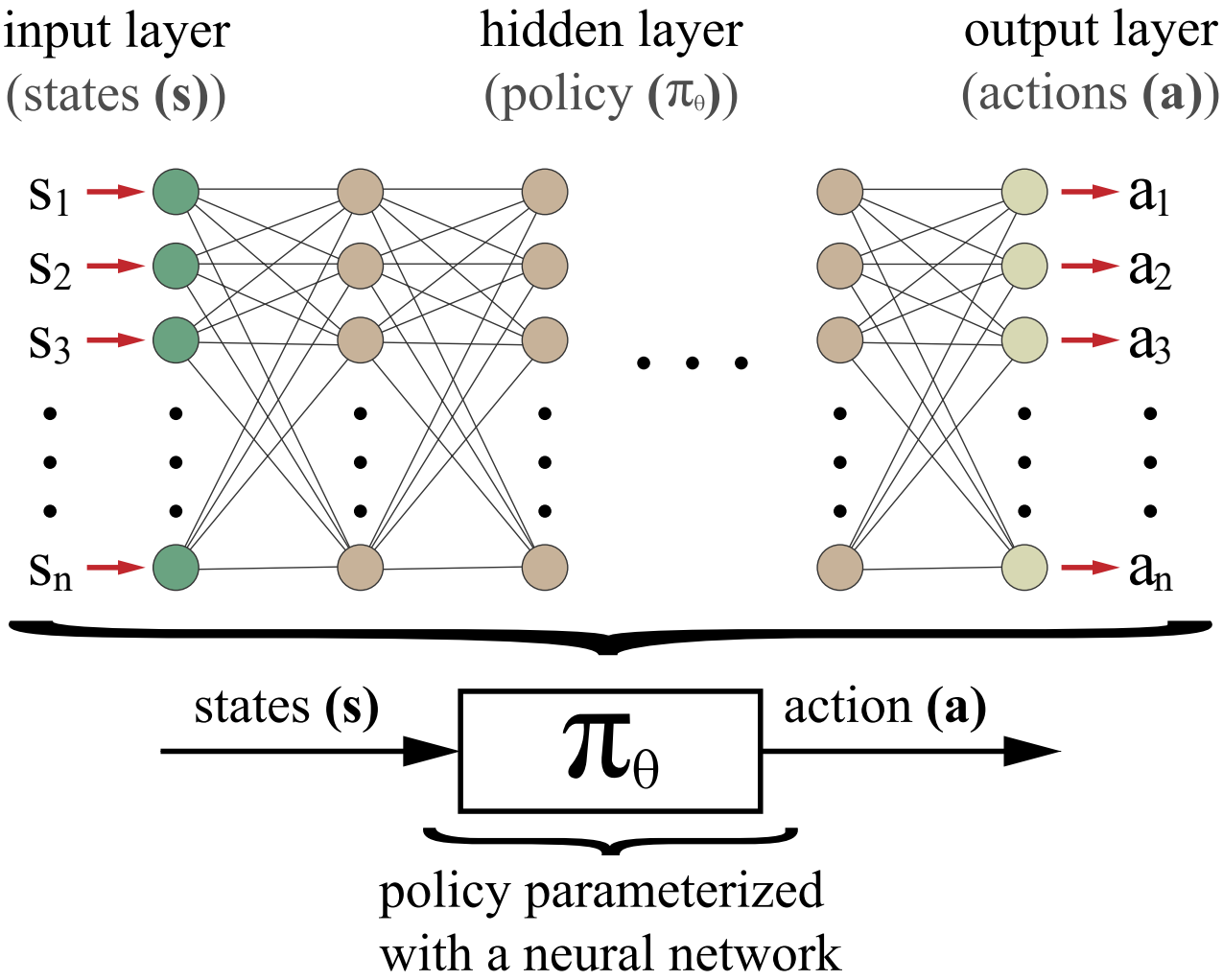
Value function approximation

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 (θ) to get a optimal (local) policy[1].

The objective function (reward) is formulated as[1]

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

with,

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

where,

- $r_t(\theta)$ probability ratio between policies
- $\hat{A}_t$ advantage function
- $V_t^{\pi_\theta}$ final reward following policy (π_θ)
- $\hat{V}_t$ expected final reward (baseline)

[1] R. S. Sutton (2018).

Proximal policy optimization

PPO is one of the best algorithms for reinforcement learning due to simplicity and high performance[1].

The main objective function (reward) is given by

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

with,

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

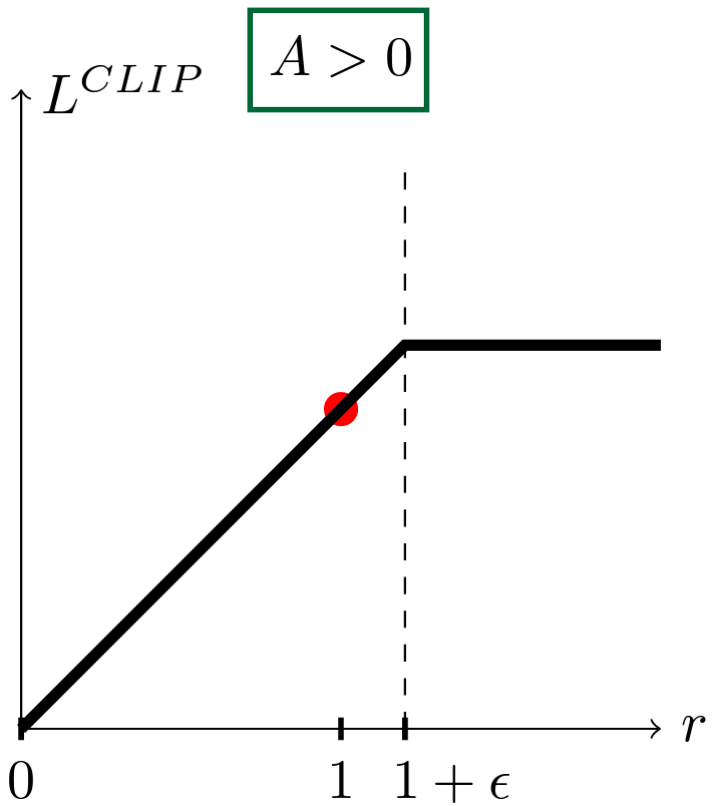
where,

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

[1] J. Schulman (2017).

Proximal policy optimization

the new action yielded
better than expected return



the new action yielded
worse than expected return

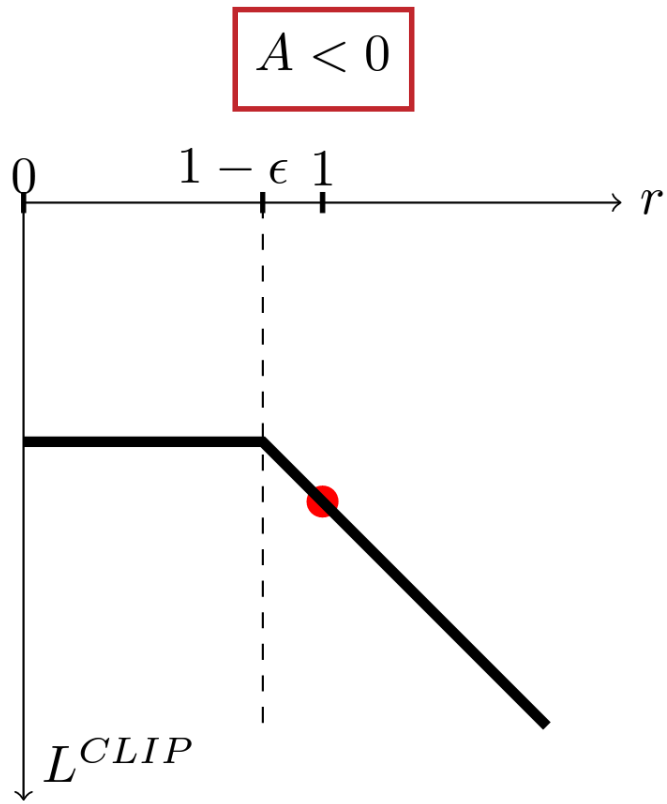


Figure: Image adapted from J. Schulman (2017)

Proximal policy optimization

The final objective function (reward and exploration) is given by[1]

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

with,

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

where,

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

[1] J. Schulman (2017).

Simulation environment and libraries

MuJoCo

Advanced physics simulation

OpenAI

TensorFlow

Bipedal robot: algorithm configuration

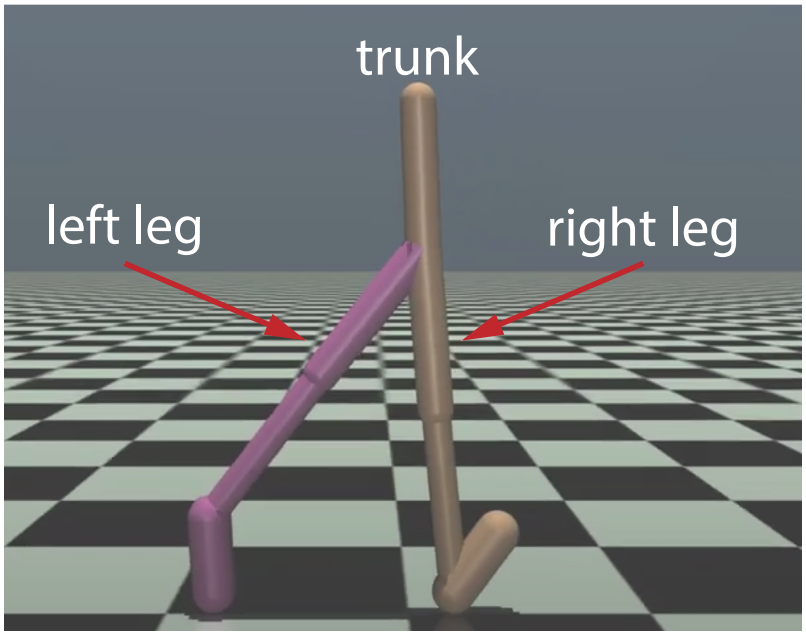


Figure: Bipedal robot in simulation environment

States

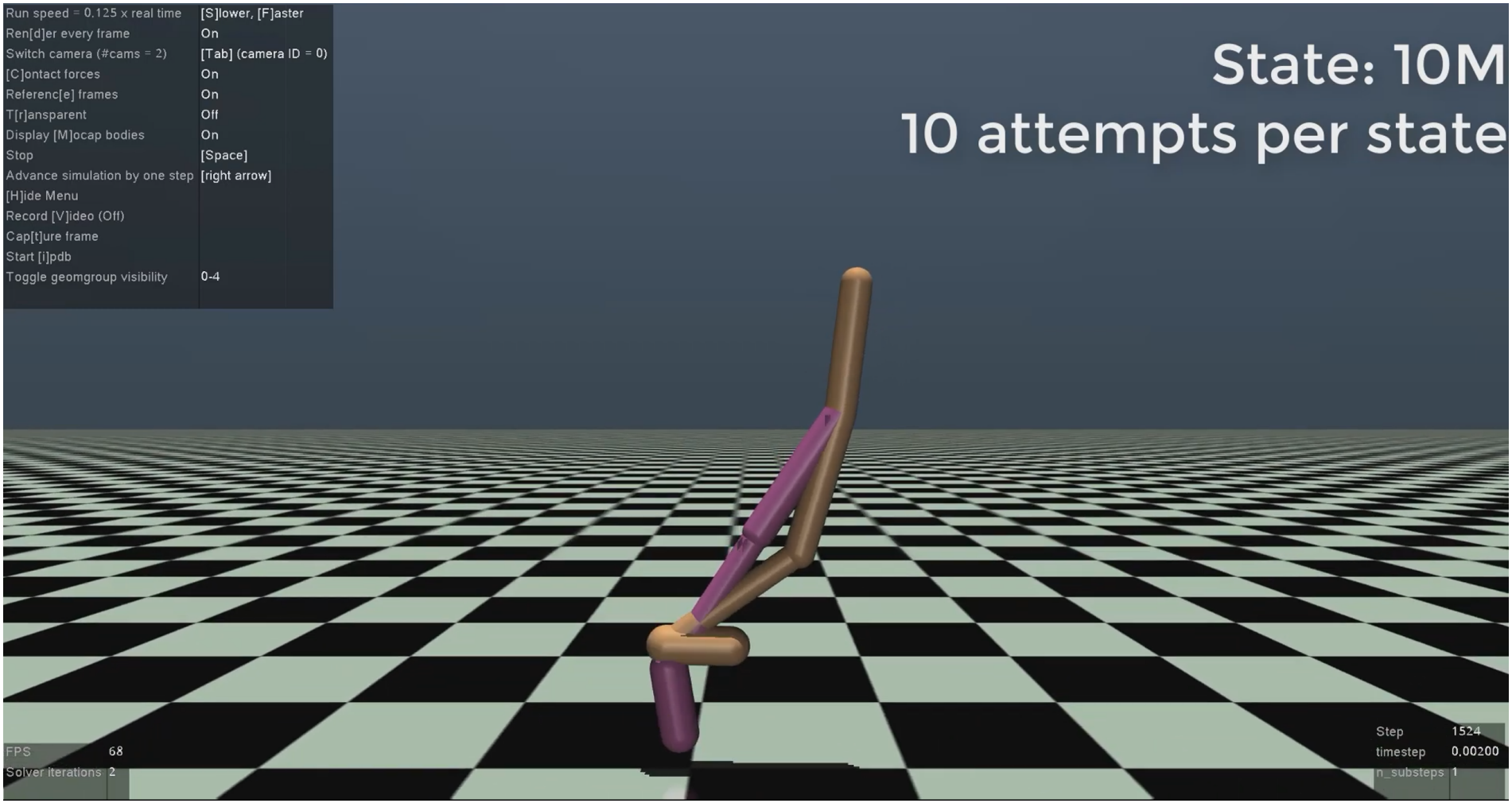
- position: 6 (left, right) + 1 (trunk)
- velocity: 6 (left, right) + 1 (trunk)
- total: 14 states

Reward

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

Outline

Results


Run speed = 0.125 x real time [S]lower, [F]aster
Ren[d]er every frame On
Switch camera (#cams = 2) [Tab] (camera ID = 0)
[C]ontact forces On
Referenc[e] frames On
T[r]ansparent Off
Display [M]ocap bodies On
Stop [Space]
Advance simulation by one step [right arrow]
[H]ide Menu
Record [V]ideo (Off)
Cap[t]ure frame
Start [i]pdb
Toggle geomgroup visibility 0-4
State: 10M
10 attempts per state
FPS 68
Solver iterations 2
Step 1524
timestep 0.00200
n_substeps 1

Outline

Conclusions