HumanoidStandup V2

Fabrice Mulotti

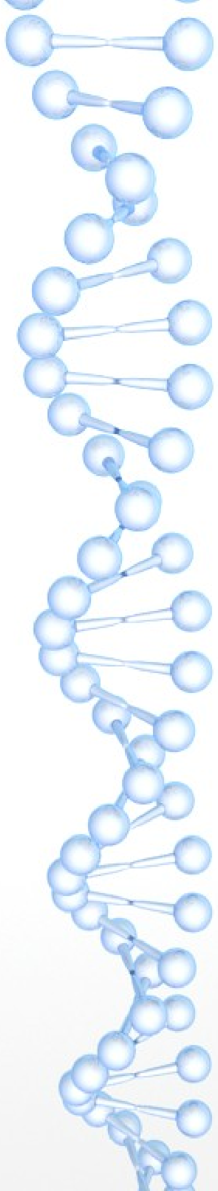
Cours 420 A59 – M. Swawola

Apprentissage par renforcement

git https://github.com/zolympe/humanoidstandup.git

Plan

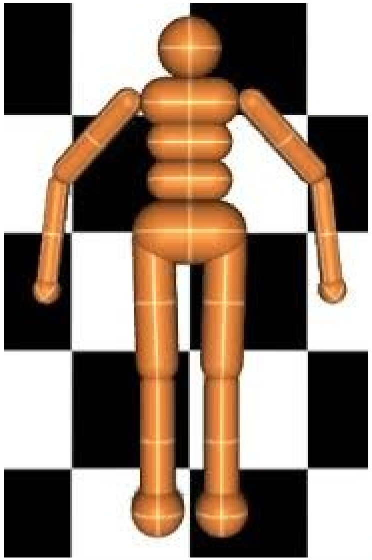
- Introduction
- Description de l’Humanoid
- Environnement Technologique
- Choix de l’algorithme
- Expérimentation
- Conclusions



Introduction

Expectations vs Reality

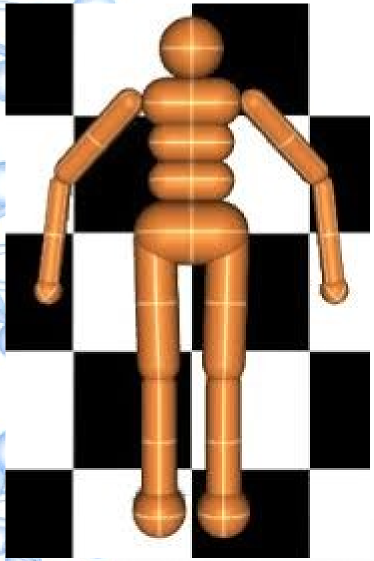
Description de l’humanoid



Objectif

Faire tenir notre humanoid debout si possible en équilibre

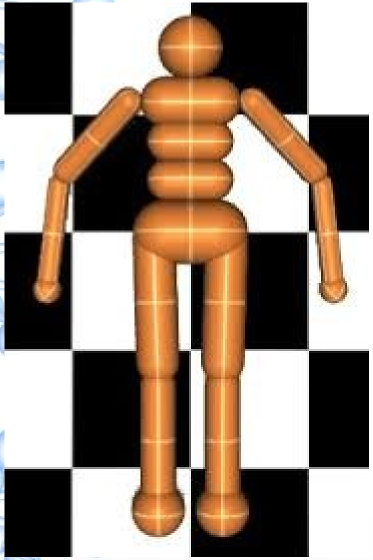
HumanoidStandup



Définissons notre ami Humanoid

→ Le monde
→ Les muscles (actions)
→ Les observations
→ La récompense

HumanoidStandup



Humanoidstandup.xml

```
<joint armature="0.02" axis="0 0 1" damping="5" name="abdomen_z" pos="0 0 0.065"
       range="-45 45" stiffness="20" type="hinge"/>
```

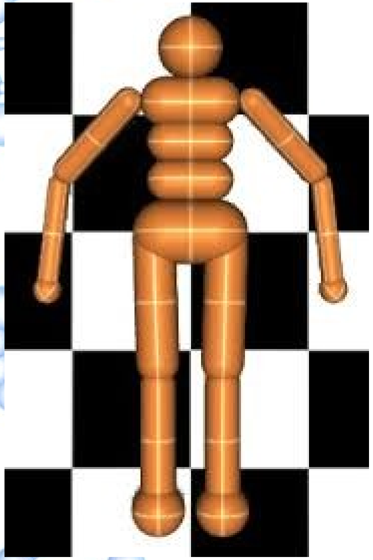
13 éléments du corps
Avec pour caractéristiques :

- Nom,
- Les positions,
- Les longueurs (calcul du moment des forces),
- ….

Monde

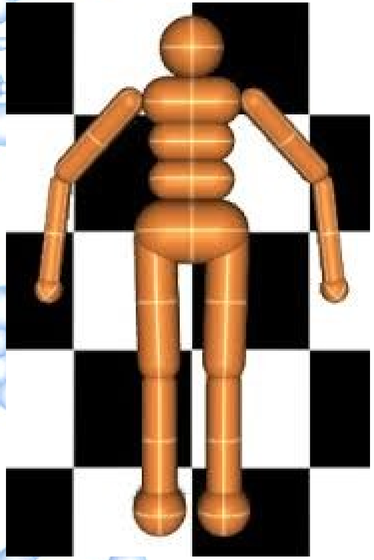
HumanoidStandup

17 Effecteurs



```xml
<motor gear="100" joint="abdomen_y" name="abdomen_y"/>
<motor gear="100" joint="abdomen_z" name="abdomen_z"/>
<motor gear="100" joint="abdomen_x" name="abdomen_x"/>
<motor gear="100" joint="right_hip_x" name="right_hip_x"/>
<motor gear="100" joint="right_hip_z" name="right_hip_z"/>
<motor gear="300" joint="right_hip_y" name="right_hip_y"/>
<motor gear="200" joint="right_knee" name="right_knee"/>
<motor gear="100" joint="left_hip_x" name="left_hip_x"/>
<motor gear="100" joint="left_hip_z" name="left_hip_z"/>
<motor gear="300" joint="left_hip_y" name="left_hip_y"/>
<motor gear="200" joint="left_knee" name="left_knee"/>
<motor gear="25" joint="right_shoulder1" name="right_shoulder1"/>
<motor gear="25" joint="right_shoulder2" name="right_shoulder2"/>
<motor gear="25" joint="right_elbow" name="right_elbow"/>
<motor gear="25" joint="left_shoulder1" name="left_shoulder1"/>
<motor gear="25" joint="left_shoulder2" name="left_shoulder2"/>
<motor gear="25" joint="left_elbow" name="left_elbow"/>
```

HumanoidStandup

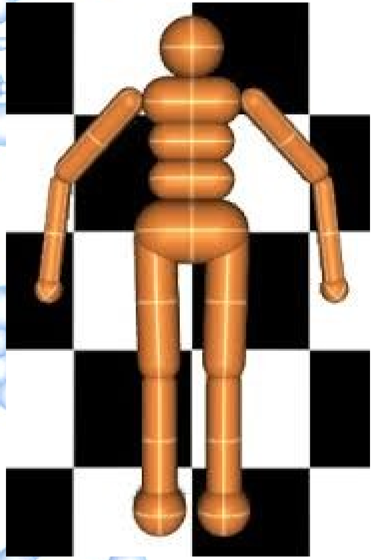


Valeurs Continues!

- 0,4

0,4

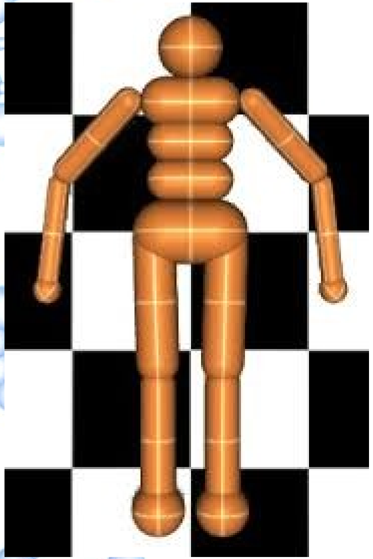
HumanoidStandup



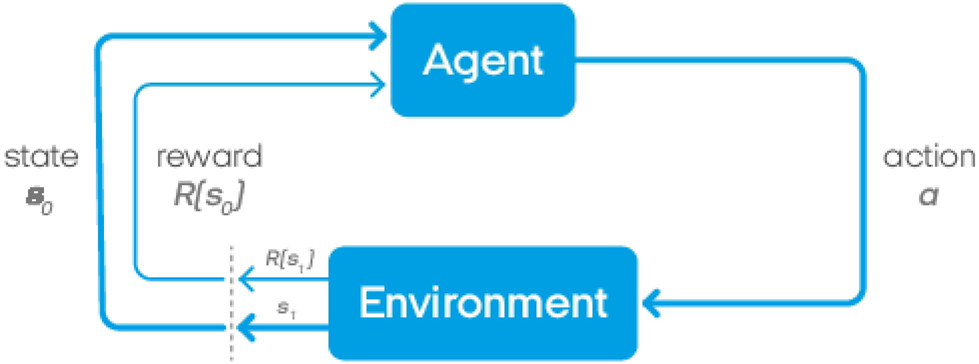
Reward :
Le récompense basée sur la position, diminuant
Avec le temps et minorée des forces appliquées

```
pos_after = self.sim.data.qpos[2]
data = self.sim.data
uph_cost = (pos_after - 0) / self.model.opt.timestep

quad_ctrl_cost = 0.1 * np.square(data.ctrl).sum()
quad_impact_cost = .5e-6 * np.square(data.cfrc_ext).sum()
quad_impact_cost = min(quad_impact_cost, 10)
reward = uph_cost - quad_ctrl_cost - quad_impact_cost + 1
```

HumanoidStandup



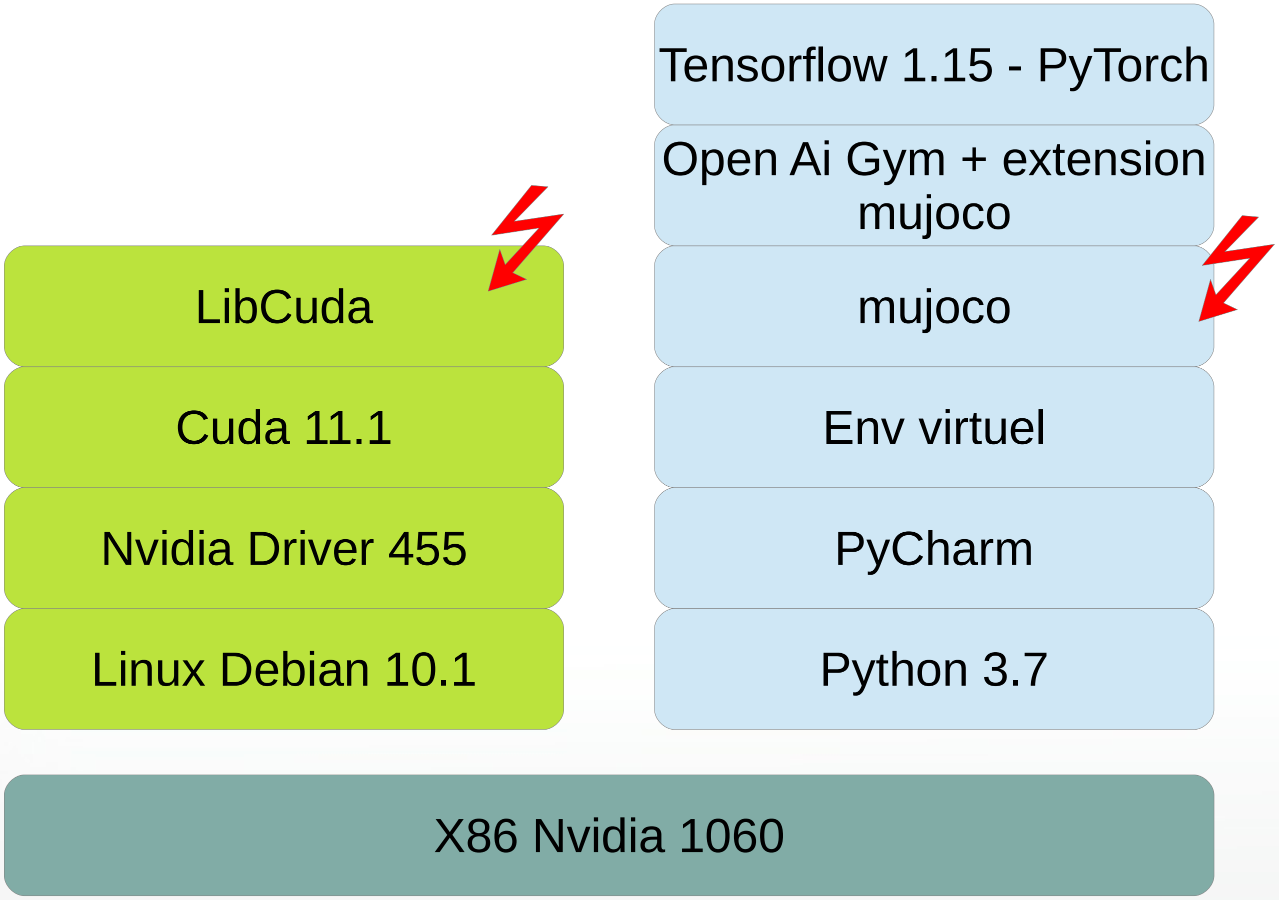
Reward
1 valeur continue positive



Action
17 valeurs
continues

States
393 valeurs
continues décrivant, les angles, position, vitesse des
Éléments du corps

Environnement technologique



PyTorch

PyTorch Build	Stable (1.7.0)	Preview (Nightly)			
Your OS	Linux	Mac	Windows		
Package	Conda	Pip	LibTorch	Source	
Language	Python	C++ / Java			
CUDA	9.2	10.1	10.2	11.0	None
Run this Command:					

```
pip install torch==1.7.0+cu110 torchvision==0.8.1+cu110 torchaudio===0.7.0 -f h
ttps://download.pytorch.org/whl/torch_stable.html
```

AlgorithmeS

DQN

Reinforce

DDPG

+?

DQN

- DQN n’est pas adapté aux valeurs continues
- Néanmoins cela m’a permis d’explorer le fonctionnement d’OpenAI et de mujoco

Reinforce 1/3

Avantages/Caractéristiques

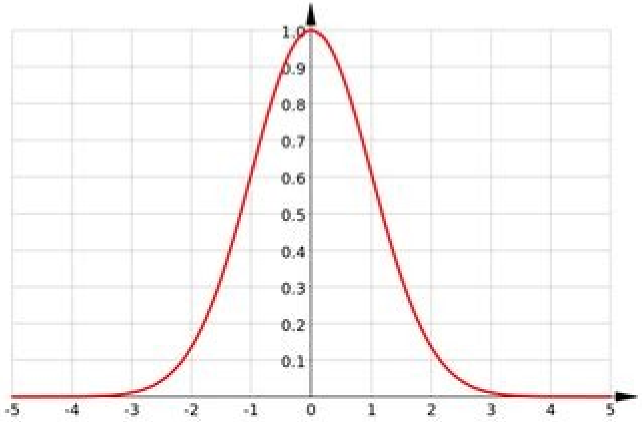
- ✔ Apprentissage de la politique
- ✔ Pas de fonction de valeur
- ✔ Efficace avec des valeurs continues

Inconvénient

- ✗ Converge vers des minima locaux
- ✗ Variance élevée

Reinforce 2/3

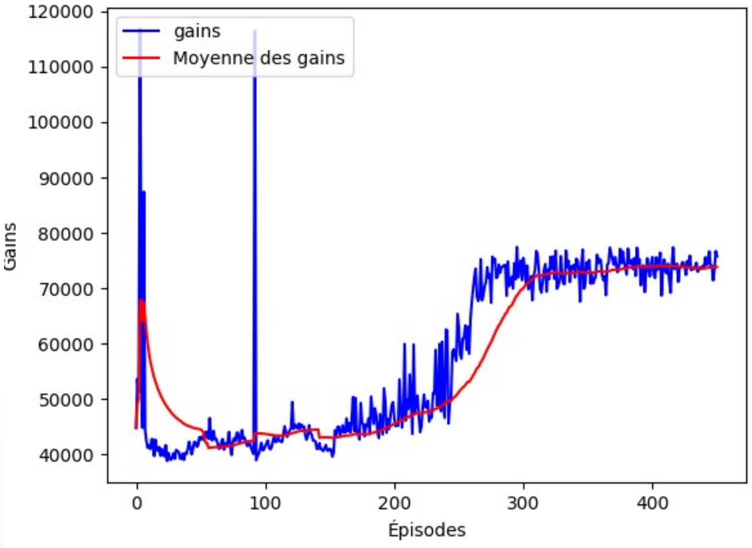
```python
def act(self, state):
    """
    Sélection d'une action suivant la sortie du réseau de neurones
    """
    state=state[np.newaxis, :] # .reshape(1, state.shape[0])

    prob=self.predict.predict(state,batch_size=self.batch_size)
    # prob=prob/np.max(prob)*self.motorRange
    # print(prob)
    #print(f"Prob {np.round(prob,2)}")
    action=np.zeros(self.num_actions)
    for i in range(self.num_actions):
       action[i]=np.random.normal(prob[0][i],scale=(self.et))
       if action[i]>self.rangeMax:
           action[i]=self.rangeMax
       if action[i]<self.rangeMin:
            action[i]=self.rangeMin
    #print(f"Action {np.round(action,2)}")
    # action=np.random.choice(self.num_actions, 1, p=prob)[0]
    # action=action.reshape(-1,1)
    return(action)
```

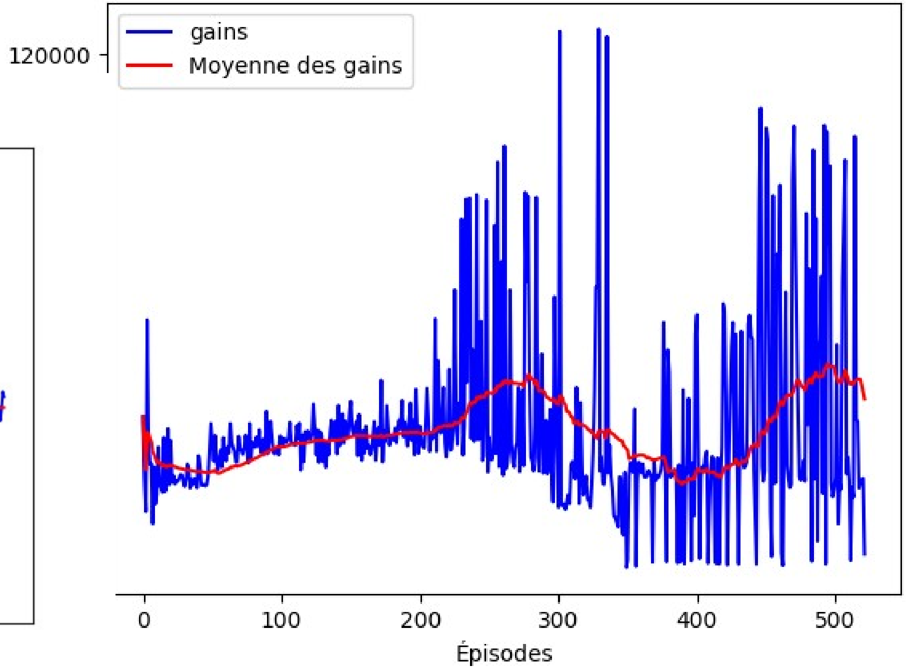


Utilisation d’une fonction random selon une loi normale centrée sur la valeur proposée par l’algorithme avec un écart type en hyperparamétre

Reinforce 3/3

Résultats décevants….





DDPG

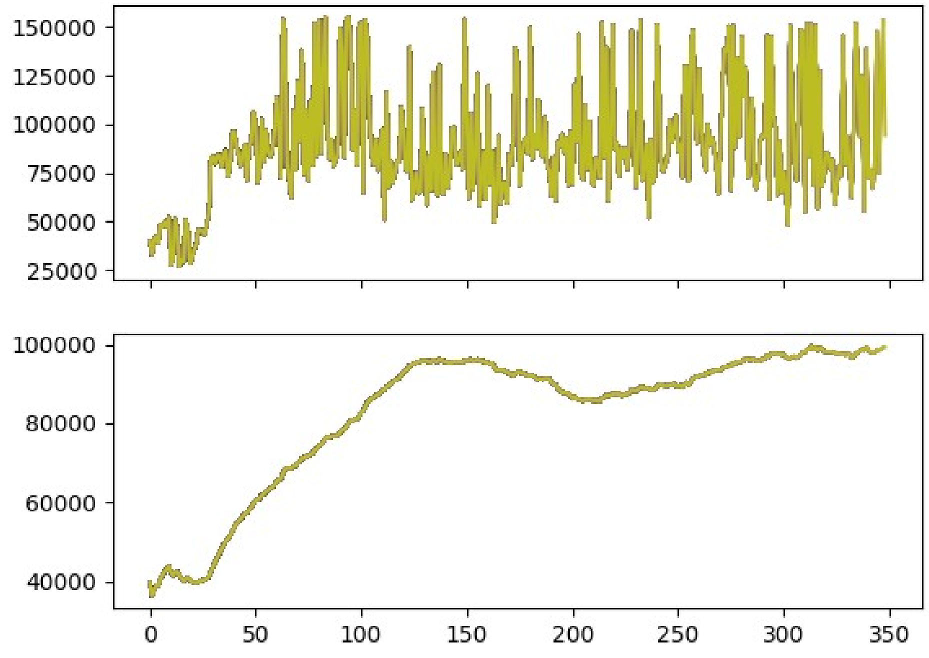
- DDPG se prête aux valeurs continues et aux grandes dimensions.
- DDPG est une approche off-policy, proposant un bon compromis biais variance.
- Basé sur la méthode Actor Critic, DDPG recherche l’apprentissage de la fonction de valeur (critic) et de politique (actor)

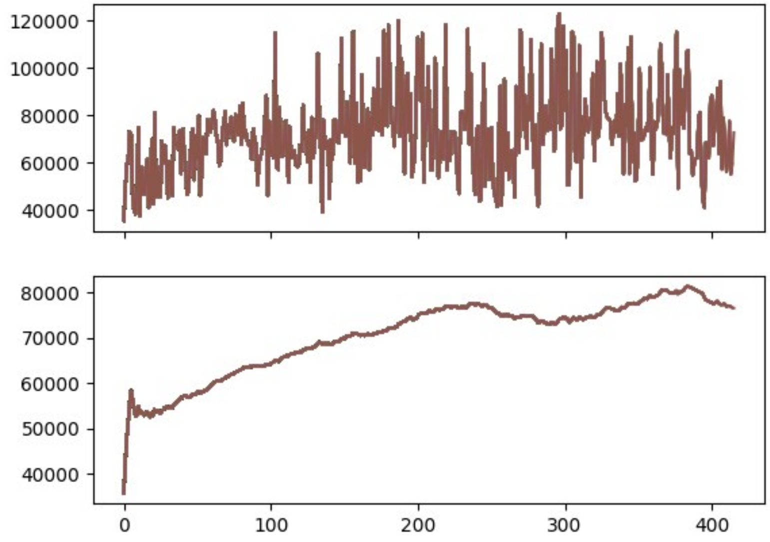
DDPG Hyperparamètres

- Nombre de neurones
- Nombre de couches (2 ou 3)
- Alpha et beta : learning rate (actor et critic)
- Tau : coefficient pour la recopie des poids
- Gamma : dévaluation

```
env = gym.make('HumanoidStandup-v2')
agent = Agent(alpha=0.0005, beta=0.005,
                input_dims=env.observation_space.shape, tau=0.001,
                batch_size=64, fc1_dims=1000, fc2_dims=300,  # b was 64 fc1 was 400 fc 2 was 300
                n_actions=env.action_space.shape[0],gamma=0.99)
```

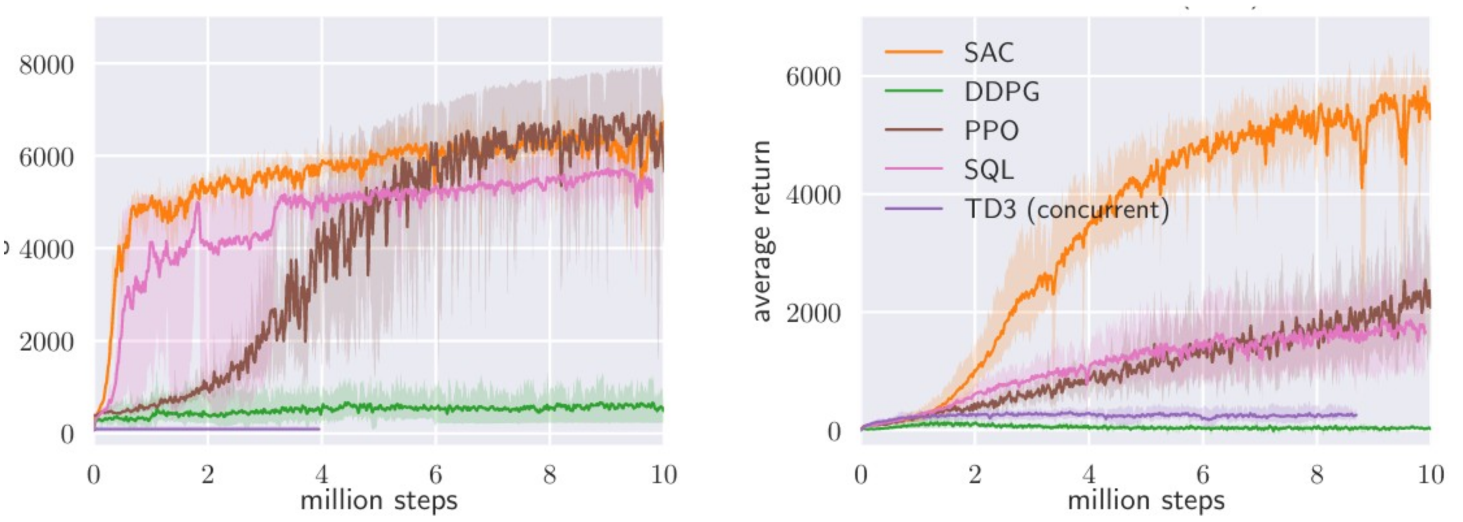
Résultats DDPG





Mais...

(b) Walker2d-v1 (c) HalfCheetah-v1



(e) Humanoid-v1 (f) Humanoid (rllab)

SAC semble montrer une performance à court terme plus élevée

https://arxiv.org/pdf/1801.01290.pdf

SAC

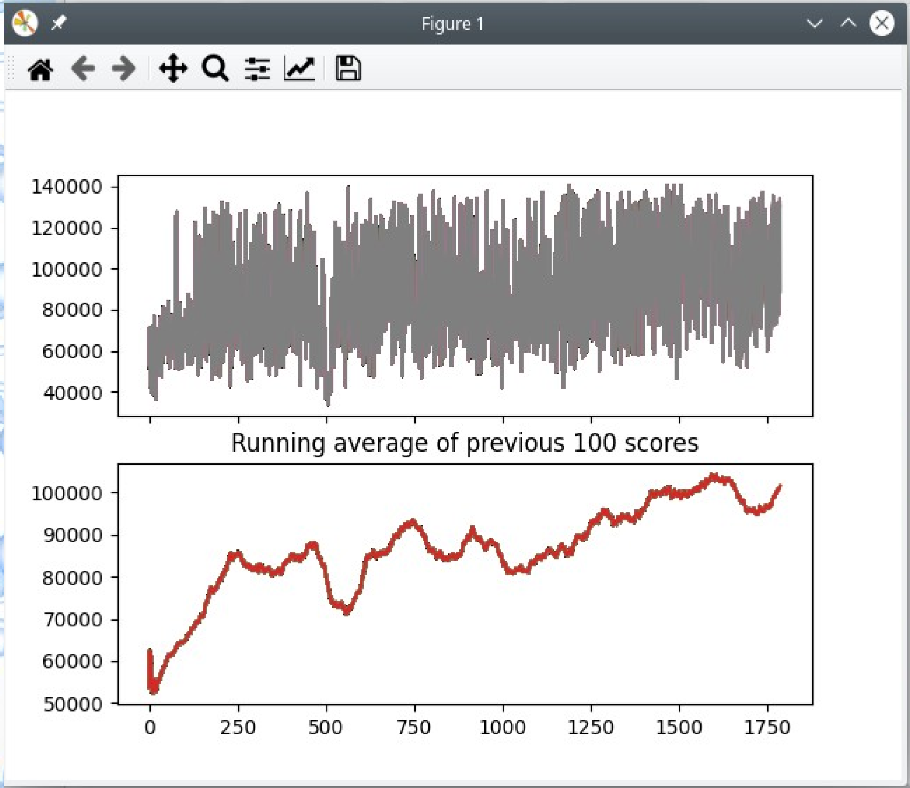
- SAC est un descendant de l’algorithme Soft Q Learning et se base sur le double Q Learning à l’instar de TD3.
- SAC se différencie par la recherche d’un compromis entre la récompense attendue et l’entropie qui est une mesure du caractère aléatoire de la police.

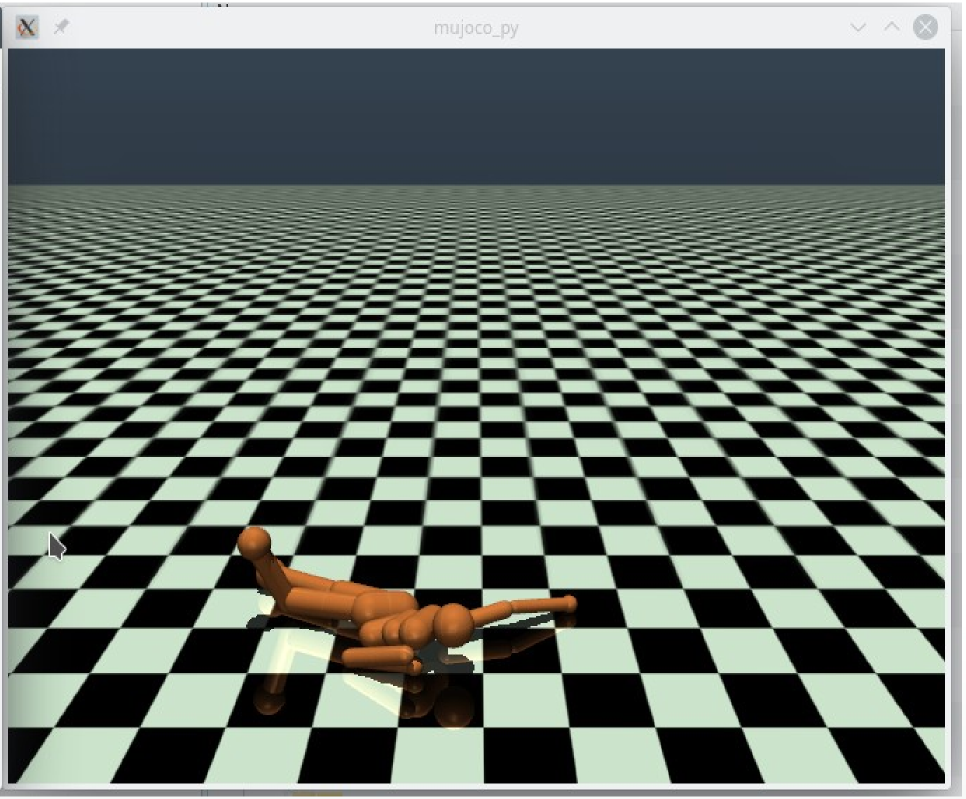
Sac Hyperparamètres

- Cf DDPG

```
env_id = 'HumanoidStandup-v2'
env = gym.make(env_id)
agent = Agent(alpha=0.0003, beta=0.003, reward_scale=2, env_id=env_id,
            input_dims=env.observation_space.shape, tau=0.005,
            env=env, batch_size=256, layer1_size=256, layer2_size=256,
            n_actions=env.action_space.shape[0])
```

Résultats SAC





Expérimentations

- Env.observation_space → 393
- Env.action_space.n → 17
- Env.action_space.high (et low) → 0.4 et -0.4
- Env.model.body_names
- Env.model.actuator_names

('world',
'torso',
'lwaist',
'pelvis',
'right_thigh',
'right_shin',
'right_foot',
'left_thigh',
'left_shin',
'left_foot',
'right_upper_arm',
'right_lower_arm',
'left_upper_arm',
'left_lower_arm')

('abdomen_y',
'abdomen_z',
'abdomen_x',
'right_hip_x',
'right_hip_z',
'right_hip_y',
'right_knee',
'left_hip_x',
'left_hip_z',
'left_hip_y',
'left_knee',
'right_shoulder1',
'right_shoulder2',
'right_elbow',
'left_shoulder1',
'left_shoulder2',
'left_elbow')

Lancement

```python
import gym
import numpy as np
from ddpg_torch import Agent
from utils import plot_learning_curve
import matplotlib.pyplot as plt
import os

if __name__ == '__main__':
    env = gym.make('HumanoidStandup-v2')
    agent = Agent(alpha=0.0005, beta=0.005,
                  input_dims=env.observation_space.shape, tau=0.001,
                  batch_size=64, fc1_dims=1000, fc2_dims=300,  # b was 64 fc1 was 400 fc 2 was 300
                  n_actions=env.action_space.shape[0],gamma=0.99)
    showBot_episode=True
    showBot_turn=False
    freshStart=True
    n_games = 5000
```

Expérimentation

- Video

Références

- Cours de Mikael pour une grande partie du code
- https://arxiv.org/pdf/1801.01290.pdf
- http://proceedings.mlr.press/v32/silver14.pdf
- https://gym.openai.com/docs/