Capstone Report
Arise and walk, by Deep Deterministic Policy Gradient algorithm

Guitard Alan

June 1, 2018

1 Definition of the problem

1.1 Project Overview

With this project, I want to tackle an important task of robotics: the ability of a humanoid to walk. This is a really interesting problem since we, humans, learn to walk without any feedback from another agent, just by a training process. The parents doesn't teach anything to the kids, the kids just learns. I think this is very interesting to build such model of artificial intelligence in order to to understand our brain a little bit more.
The anthropologic reason is not the only reason that task is important, it is also important to build autonomous walking robots able to progress on every fields. There are several areas of actions: In war, such a robot will be able to reach wounded soldier to give medical treatment or bring the soldier back to a safe place. It will avoid a medic human to risk his life in order to do that. In hospital, we will gain time by giving task to the walking robot so doctors and nurses will have more time to give to the patient. In every day life, we will be able to get near from the future described by movies like "I-Robot". Even if it questions the place of the robot in human society, I think making this kind of robot is still a goal to reach in order to send the robot in places human cannot or with difficulty go, like to another planet.
I am talking about walking but a neural network able to make a robot walk sure will be able to tackle another difficult task. The difficulty in those task is their continuous observation spaces. Indeed, in order to make a robot walk, we have two possibilities. The first one is to learn over pixels of the environment. I didn't choose this one because I don't think a human learn to walk from its sight but rather from its body. That why I chose to watch the position of its joints. With pixels, we will able to train the humanoid to watch its steps but only after it is able to walk.

1.2 Problem Statement

The goal of this project, precisely, is to train a 3D avatar to walk forward on a plane fields. I will use OpenAI Gym ([1]) as interface to the environment because it gives a simple and general way to use all kind of environment. For 3-dimensional avatar, it provides a binding for Mujoco library but since this library is paid, I will use a free similar one called Roboschool[1] which propose few environments among the one I will use, RoboschoolHumanoid-v1.
I am planning to use Reinforcemen t Learning to tackle this task with the Deep Deterministic Policy Gradient algorithm (Lillicrap et al.), an actor-critic algorithm.

Actor-critic In Reinforcement Learning, many algorithms uses an action-value function. It is a function which returns the value of an action a from a state s following a policy μ. It is defined like this:

$$Q^{\mu}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}[r(s_t, a_t) + \gamma Q^{\mu}(s_{t+1}, \mu(s_{t+1}))] \tag{1}$$

$$R = \sum_{i=t}^{T} \gamma^{(i-t)} r(s_i, a_i) \tag{2}$$

1.3 Metrics

2 Analysis

2.1 Data Exploration

2.1.1 Observation space

This environment has an observation space of 44 float values in the range [-5, 5] which is a concatenation a three vectors described as follows:

- **more**: It is a vector of 8 values defined as follows:
  - The distance between the last position of the body and the current one.
  - The sinus of the angle to the target.
  - The cosinus of the angle to the target.
  - The three next values is the X, Y and Z values of the matrix multiplication between
    - $$\begin{pmatrix} \cos(-yaw) & -\sin(-yaw) & 0 \\ \sin(-yaw) & \cos(yaw) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
    - The speed vector of the body.

[1]https://github.com/openai/roboschool

    - The roll value of the body
    - The pitch value of the body

- **j**: This is the current relative position of the joint described earlier and their current speed. The position is in the even position, and the speed in the odds (34 values).
- **feet_contact**: Boolean values, 0 or 1, for left and right feet, indicating if the respective feet is touching the ground or not.

2.1.2 Action space

The action space is a vector of 17 float values in the range [-1, 1]. Each value corresponds to the joints of the avatar by this order XML:

- 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

At each step, these values are applied to all the joints of the body by the code

```
for n,j in enumerate(self.ordered_joints):
    j.set_motor_torque( self.power*j.power_coef \
                        *float(np.clip(a[n], -1, +1)) )
```

in the `apply_action` function in the class which extends the `gym.Env` class (`RoboschoolMujocoXmlEnv`) to set the torque value into the respective motor.

2.2 Exploratory Visualization

2.3 Algorithm and Techniques

Reward The reward is a sum of 5 computed values:

- **alive**: -1 or +1 wether is on the ground or not
- **progress**: potential minus the old potential. The potential is defined by the speed multiplied by the distance to target point, to the negative.

- **electricity_cost**: The amount of energy needed for the last action
- **joints_at_limit_cost**: The amount of collision between joints of body during the last action
- **feet_collsion_cost**: The amount of feet collision taken during the last action

2.4 Benchmark

3 Methodology

3.1 Data Preprocessing

3.2 Implementation

3.3 Refinement

4 Results

4.1 Model Evaluation and Validation

4.2 Justification

5 Conclusion

5.1 Free-Form Visualization

5.2 Reflection

5.3 Improvement

References

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[2] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015. URL `http://dblp.uni-trier.de/db/journals/corr/corr1509.html#LillicrapHPHETS15`.