Collision Avoidance using Deep Q-Network

Robot Learning Project

Chethan Mysore Parameshwara

Outline

> When to use DQN ? - Problem Formulation

> Why DQN? Why not Q-Learning ?

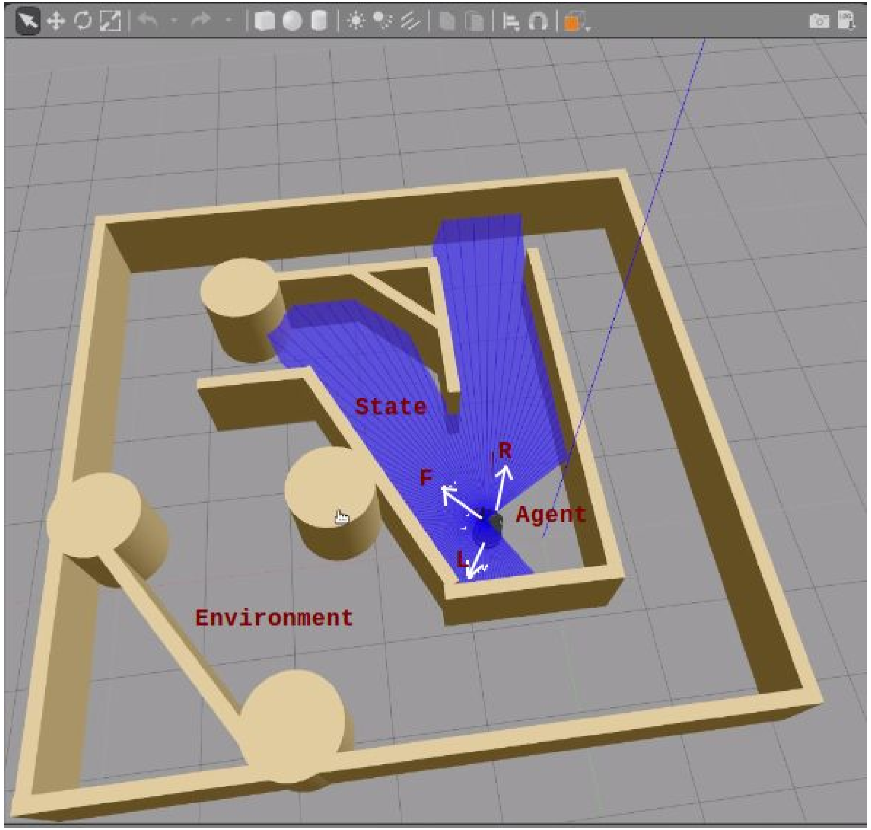
> What is DQN ?

> How to train DQN ?

> Does it really work ? Demo

> Who helped me ? References

When to use DQN ? - Problem Formulation



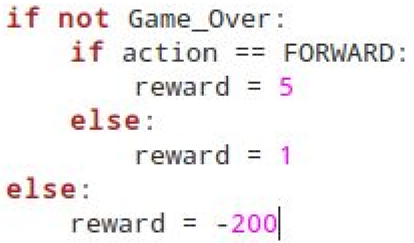
> Markov Random Process

probability of the next state s_{i+1} depends only on current state s_i and action a_i, but not on preceding states or actions.

> Episode - $s_0, a_0, r_1, s_1, a_1, r_2, s_2, \ldots, s_{n-1}, a_{n-1}, r_n, s_n$

> Policy - $\pi = P(a/s)$

> Rewards -



```
if not Game_Over:
    if action == FORWARD:
        reward = 5
    else:
        reward = 1
else:
    reward = -200
```

Why DQN ? Why not Q-Learning ?

We know that

$$R_t = r_t + \gamma(r_{t+1} + \gamma(r_{t+2} + \ldots)) = r_t + \gamma R_{t+1}$$

$$Q(s_t, a_t) = max\ R_{t+1}$$

$$\pi(s)\ = argmax_a\ Q(s,a)$$

We iteratively approximate the Q-function using the Bellman equation

$$Q(s,a) = r\ + \gamma max_{a'} Q(s', a')$$

Q - Learning

_Q_[_s_,_a_] = _Q_[_s_,_a_] + α (r+γ _max Q_[_s_,_a_] - _Q_[_s’_,_a’_])

Q - Learning Limitations

> Multi-dimensional state

> Multiple action

> Complexity in searching Q - table

What is DQN ?

> Approximate Q-function with a neural network

> Extracts features from multidimensional data

> Loss Function

$$L = \frac{1}{2}[\underbrace{r + max_{a'} Q(s', a')}_{\text{target}} - \underbrace{Q(s, a)}_{\text{prediction}}]^2$$

Problems with DQN

> correlations present in the sequence of observations

> correlations between the action-values (Q) and the target values (r + max (Q))

Workaround

> Experience Replay

> Fixed Target Q-Network

How to train DQN ?

Network Architecture

- Input Size - 100
- Output Size - 3
- Network = 3 Fully Connected Layer
- Activation Function - ReLU
- Optimization Solver- RMSprop

Network Parameters

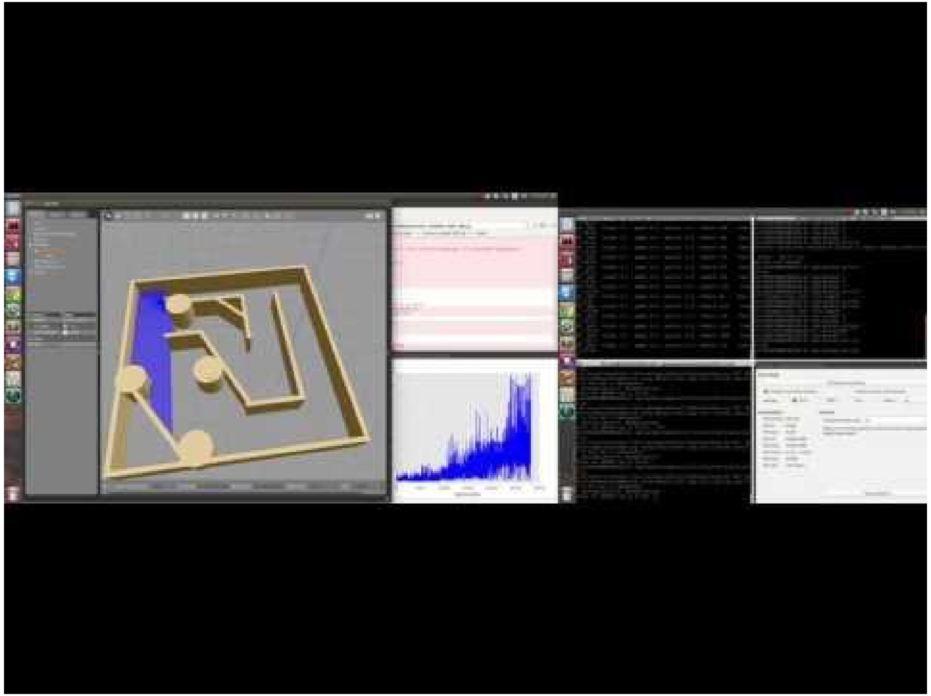
- Max_Episodes = 15000
- Max_steps_episode = 1000
- Exploration_Rate = 1 (adaptive)
- Minibatch_size = 64
- Learning_Rate = 0.8
- Discount_Factor = 0.99

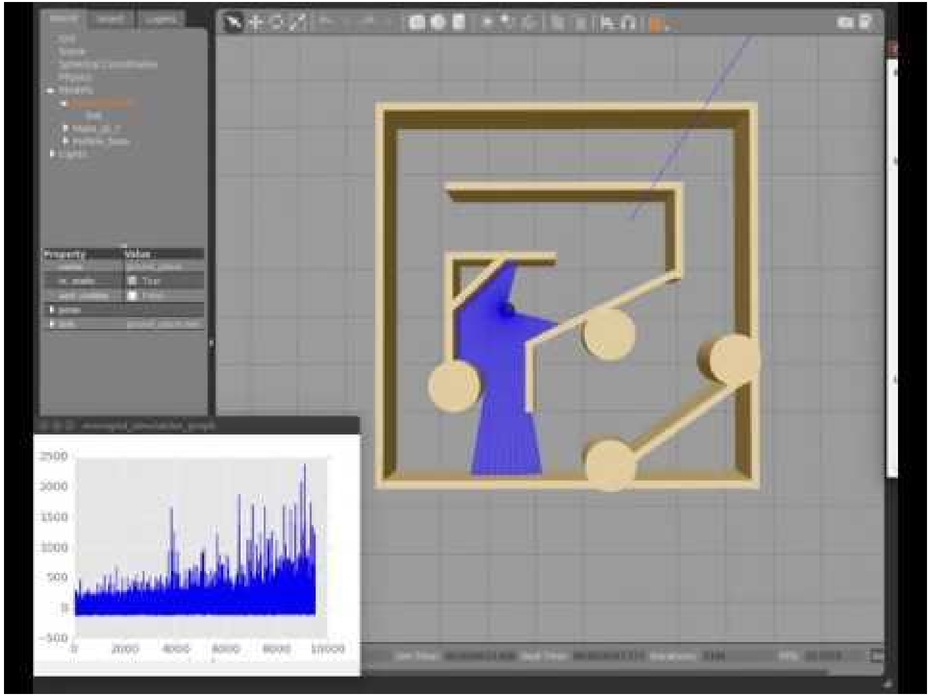
Hardware - NVIDIA GF106GL [Quadro 2000] GPU (Outdated !!)

Software - Keras, ROS, Gazebo

Programming Language - Python

Does it really work ? Demo





Source Code **-** https://github.com/analogicalnexus/gym-gazebo

Who helped me ? References

- Zamora, Iker, et al. "Extending the OpenAI Gym for robotics: a toolkit for reinforcement learning using ROS and Gazebo." *arXiv preprint arXiv:1608.05742* (2016).
- Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *Nature* 518.7540 (2015): 529-533.
- https://www.nervanasys.com/demystifying-deep-reinforcement-learning/

Thank you