UNIVERSITY OF MARYLAND

Drone Navigation with Reinforcement Learning

Formal Report

Project

Final Project

Author:
Akwasi A. Obeng

DirectoryID:
obenga01

December 08, 2019

Contents

1 Abstract

In recent times, the increase in the usage of automobiles has caused difficulty in transporting goods between two places. As a result, the alternative use of employing aerial robots such as drones in transporting goods has gained a lot of popularity due to the reduced traffic and the effiency and ease of transporting goods from one place to another. However, aerial robots face the disadvantage of navigating dynamic obstacles and being able to interact with the environment in an intelligent manner. Due to these difficulties, it is imperative that the drone be equipped with some form amount of intelligence. This paper focuses on how reinforcement learning can be applied to enable the drone navigate its environment intelligently and efficiently.

2 Introduction

Drones in recent times are being used for surveillance in military and for transportation purposes. Amazon currently is trying to commercialize the use of drones for transporting products from one place to another. The problem of operating a drone is that even if a drone has been programmed to follow a specified path using the most intelligent of algorithms, there are bound to be uncertainties. How well the drone responds to such situations is of paramount importance as it could mean a drone accidentaly hitting an obstacle, or detecting unreasonable things or venturing into really dangerous and costly situations.

Figure 1: Drone delivering kidney

This paper focuses on the first phase of the project which is training a drone to interact with its environment in an intelligent way devoid of any sensing capability. Suitable reinforcement learning algorithms are tried and tested to see how well the drone learns- that is, how much reward can be accumulated from experience.

3 Reinforcement Learning Introduction

Reinforcement learning algorithms differ from the traditional machine learning algorithms, supervised and unsupervised learning, in the sense that, the agent interacts with the environment and continous to learn from experience. Unlike supervised learning, where there is a target that is to be matched, in reinforcement Learning, the exact target or goal may not be known fully. The agent continuous to change its behavior in response to the amount of reward received for a particular action. RL also differs from unspervised learning because the goal is not just to find structure or similarity in the data, but rather actions or behaviors that maximize the reward in the short term or long term depending on the problem at hand.

4 Problem Description

The goal of the project is for the drone to navigate through the gateways as illustrated in the figure shown below. The rewards have been placed right in the middle of the gateways and the drone is supposed to orient and learn to generate a path that accumulates the most reward.

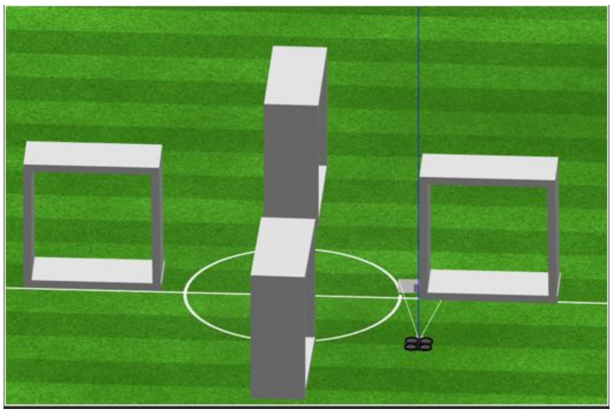

Figure 2: World created in ROS

4.1 Problem Characteristics

- **State Space** - Continuous State space $\rightarrow$ (Position p,Orientation o)
- **Action Space** - left, right,up,down, forward
- **Environment** - We focus on **static environment** for now.

4.2 Gazebo Parameters

Due to the huge workspace field, the robot is confined to a rectangular workspace with the dimensions given below.

$$min_x : -2.7 \quad max_x : 7$$
$$min_y : -2.5 \quad max_y : 7.3$$
$$min_z : 0.00 \quad max_z : 3.00$$

NB. Gazebo uses SI units. Therefore distances are in meters.

5 Approach

Both qlearning and deep qlearning where tested to see how well the drone learned. A brief summary and results using both approaches is given below.

5.1 Reward Structure

Reward structure	
Reward	Values
forward	1
turn	1.5
up	1
down	1.5
goal	2000

The drone gets a goal reward of 300, each time it passes through any of the centers of the gateways **(-1.6,2.2,2.0),(2.33,-1.14,2.0),(5.7,2.29,2.0),(2.0,6.1,2.0)** at some thresholded radius of 0.7m. To prevent the drone from circling about the same center in a gateway, the **goal reward** is set to **zero** after center has been reached and reset back to **300** after any of the other centers other than the previous one has been reached.

5.2 Qlearning

Q learning tries to maximize the action-value using the update rule given below.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \tag{1}$$

where,

$$\alpha \rightarrow \text{learning rate, A measure of movement for every time step}$$
$$\gamma \rightarrow \text{Discount factor, A measure of future or immediate rewards}$$
$$A \rightarrow \text{Action}$$
$$S \rightarrow \text{State}$$

Qlearning works with discrete state and action spaces. To apply qlearning to our problem, we discretize the continous state space apply the update rule shown above.

5.2.1 Hyperparameters

- **alpha**: 0.01 - Small learning rate to ensure convergence
- **gamma**: 0.6 - Fairly big discount factor as the drone can easily recover from bad orientations. No need for heavy dependence on future rewards.
- **epsilon**: 0.995 - More exploration in the beginning
- **epsilon_discount**: 0.998 - Exploration should last for several episodes. Need 1000 episodes to reach an epsilon value of about 0.1

5.2.2 Policy, Steps And Episodes

Qlearning was applied to the problem and the drone was trained using **epsilon-greedy policy** for about **1000 episodes** and **700 steps**.

5.2.3 Qlearning Results

Clearly, it can be seen from the graph that, the robot is not maximizing the amount reward that can be generated because, the state space is huge. Even after 1000 episodes, the drone is nowhere close to desired optimal solution. For qlearning to work, the resolution of the discretization can be increased-this results in huge state space- and the number of episodes can be increased. Better results can be achieved but the downside is that it would take a lot of time and space to achieve the desired result

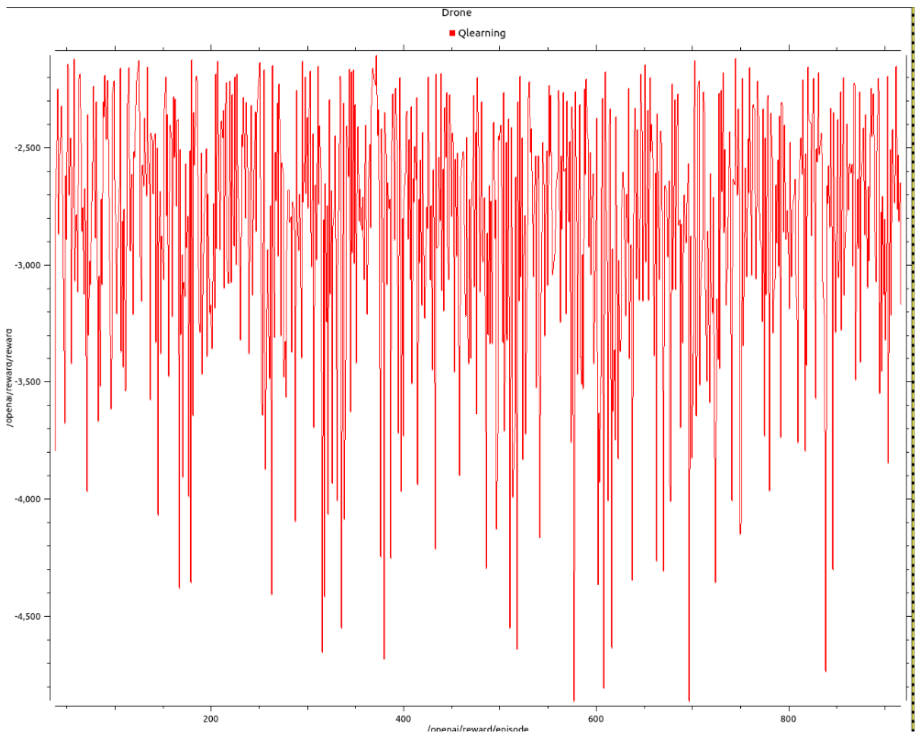

Figure 3: Qlearning Results

5.2.4 limitations

Qlearning is not an appropriate solution as:

- **State space is huge(continuous state space)**- The drone can be in particular position with some orientation.
- **Slow to learn** - Qlearning tries to learn the value of each state space. This is not ideal as state space close(suitable metric) together coudld have yield similar values. Therefore, a method which approximates the value function using a suitable function is more desirable.
- **Huge memory** - Qlearning learning requires a lot of memory space for this problem

5.3 Deep Qlearning

5.3.1 Function Approximation

Instead of storing all the possible states with their associated action-values, the values corresponding to the states are approximated. States that are really close together may tend to have the same values,or even yet, there may be an underlying structure or pattern to action-values associated to the different states. Therefore a Deep Network, is employed to approximate these values. to the states.

The goal then is to find parameters w, that minimizes the distance(suitable metric) between the actual value function $v_\pi(s)$ and the approximate value function $\bar{v}(s, w)$. The cost function then becomes,

$$J(w) = E_\pi[(v_\pi(S) - \bar{v}(S, w))^2] \tag{2}$$

Unlike in Supervised learning, where the target(actual value) is known, the actual value function(v_π) is not known and continuous to change. The update rule for action-value associated with a state is therefore given as

$$\Delta w = \alpha(q_\pi(S, A) - \bar{q}(S, A, w))\nabla_w \bar{q}(S, A, w) \tag{3}$$

5.3.2 Mini Batch Training Method

Mini batch update,with a **batch size of 100**, was used as the ordinary batch update is too slow and incremental changes using the stochastic update is susceptible to noise as it doesn't capture the pattern really well. In every iteration, 100 points(S,A,R,S') are sampled and used in updating the weights.

5.3.3 Architecture

The network was trained with two hidden units. The properties are tabulated below. The values were attained through trial and error.

Network Architecture		
Layers	Units	Activation Function
Input layer	state space dim	–
First layer	60	relu
Second layer	75	relu
Output layer	action space dim	linear
Other Properties		
Optimization Method	Adam	–
loss Function	MSE	–

5.3.4 Policy, Steps and Episodes

Qlearning was applied to the problem and the drone was trained using **epsilon-greedy policy** for about **6000 episodes** and **3000 steps**.

5.3.5 Hyperparameters

- **alpha**: 0.01 - Small learning rate to ensure convergence
- **gamma**: 0.6 - Fairly big discount factor as the drone can easily recover from bad orientations. No need for heavy dependence on future rewards.
- **epsilon**: 0.995 - More exploration in the beginning
- **epsilon_discount**: 0.998 - Exploration should last for several episodes. Need 1000 episodes to reach an epsilon value of about 0.1

5.4 Results

Shown below are the results obtained using deep-q learning

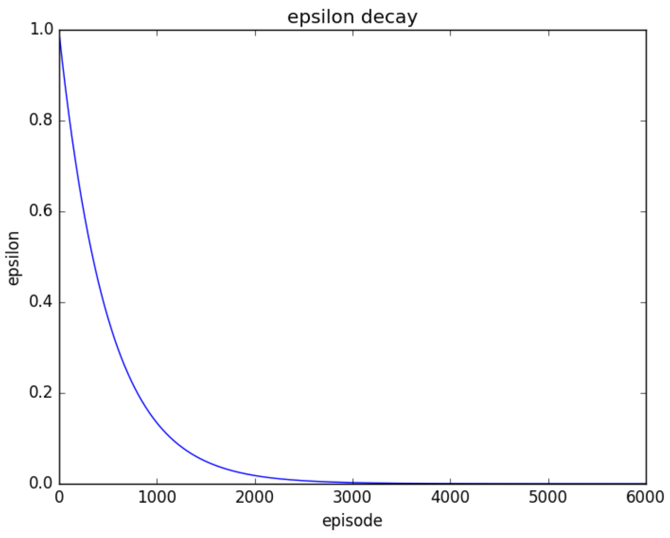

Figure 4: epsilon decay

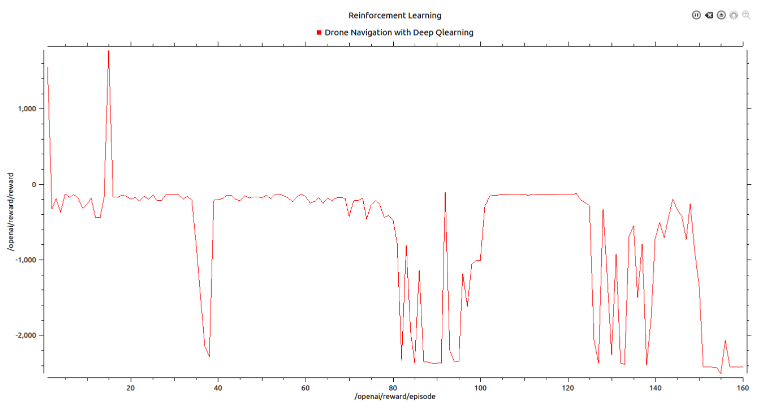

Figure 5: Deep-q Learning

5.5 Tools Used

- ROS -Kinetic
- Gazebo
- Openai-gym - The robot was trained using openai library
- Openai-ros - Openai ros makes interfacing robots in ros with easy. Refer to ros wiki for tutorials.
- Deep Neural Network - The code utilized keras which utilized tensor flow backend.

6 Analysis and Challenges

Clearly the graph shown above for deep qlearning is slightly better than qlearning. The reason for the steep negative rewards is that sometimes the drone gets stuck in some situations. For example, the drone sometimes getsstuck beneath the square hole and it takes a really long time to get out of that situation.

Several days was spent training the drone. Ros was extremely slow and coupled with how slow my machine was, this made the training process even much slower. The obvious decision was to train on the cloud but interfacing openai-gym with ros had a lot of dependencies and as a result couldn't train much longer.

7 Conclusions

Clearly, it can be noted that, deep qlearning outperforms qlearning when the state space is big and continuous. Qlearning is much more useful when both action space and state space are discrete and small. Depending on the type of problem, we can switch to different types of learning methods. For Example, if the drone is constrained to move in a plane with some predefined number of steps, then qlearning will be more suitable.

8 Future Work

We plan to continue working on the problem trying out different reinforcement learning architectures. From there, we incorporate dynamic obstacles and include sensing capability of the drone in the learning process- that is, camera and lidar. Then we create a suitable environment with a slightly more complex goal and examine how well the drone learns.

9 Bibliography

https://github.com/keon/deep-q-learning/blob/master/dqn.py
https://www.theconstructsim.com/how-to-launch-drone-simulation-locally/
https://www.theconstructsim.com/using-openai-ros/ http://wiki.ros.org/openai_ros

10 Appendix

Refer to the github link below for the code
https://github.com/mesneym/Dynamic-Obstacle-ML