

Robotics Software Engineer Nanodegree: Deep RL Arm Manipulation

Manuel Huertas López

Abstract—Reinforcement learning is an area of machine learning. It instructs an agent to take actions in an environment in order to maximize a cumulative reward. In this project a DQN algorithm will be used to guide a robotic arm to touch a cylindrical object. The idea behind the algorithm is to find an action value function; this function will give the cumulative expected reward for every state and action pair values. Once this function is obtained the action for every state will be the one that maximizes the reward. DQN is a variation of the algorithm where this function is replaced with a Deep Neural Network.

Index Terms—Robot, Udacity, Deep Reinforcement Learning.

1 INTRODUCTION

In this project a Robotic Arm with three degree of freedom will be taught in order to touch a cylindrical shape in the scene. Rather than using the forward and backward kinematic to calculate the position of the arm with respect to the joint angles or to move the arm to a goal position sending demands to the joints, the robot will try some random movements and it will receive a positive or negative reward. A video camera will be recording constantly the robot movement and this information is used to train a neural network, the network will try to reproduce the sequence of movement that maximise the cumulative reward. As the final goal the robot will find the combination of joint movement, without a model, to approximate and finally touch the object in the scene.

Every Episodic finalized when the robot: touch the ground, touch the cylinder or the maximum number of iteration is reached. The final reward will be positive just in case the robot touch the cylinder, and there are some intermediate positive reward bases on smooth movement and when the robot is closer to the cylinder. The problem can be defined as a Markov Decision Process where:

- a finite set of states: the image from the video camera and all the pixel value combinations.
- a finite set of actions: the movement every joint.
- a finite set of rewards: the positive and negative reward depending on the robot behaviour.
- the one-step dynamics of the environment: given the current state and the action, the expected new state and new reward. The neural network will obtain this during the training process.
- a discount rate between $[0,1]$: the discount rate.

2 BACKGROUND / FORMULATION

Our goal is to find a function that for every state, the image in this case, tell use which action to take, the movement of the joint. This problem is not easy since there are a big number of states, pixel combination of the image, and sequences of movements. In this project an algorithm based

in DQN have been used. In DQN a Deep Neural Network will build a function where the input will be the pixel of the image and the output the expected cumulative reward for every possible action. Selecting the action with the highest cumulative reward will give us the desired action. In addition, a Long Short Term Memory network have been used to use the information of the sequence of multiple images, instead of a single frame.

2.1 Deep Q Network

The Q-Learning algorithm is a model-free algorithm. The values for each state-action pair are estimated based on observation of the environment. The Q-Learning algorithm estimate the state action function using the following equation:

$$Q(s_t, a_t) = (1 - \alpha) \cdot Q(s_t, a_t) + \alpha \cdot (r_t + \gamma \max_a Q(s_{t+1}, a))$$

The meaning of this parameter are:

- s : state value at time t , from the observation.
- a : action value a time t .
- α : learning rate, the way in which new values replaces old ones.
- r : reward at time t .
- γ : discount factor.

In addition, the greedy policy is used to select between a current best action, exploit, or to select a random action, explore. If during the learning process we always select the action with currently the best reward we are never going to explore new solutions whose can give a better discounted future reward. On the other one, we need to select the current best action in order to use the knowledge we are acquiring iteratively. One way to accomplish this is to explore more at the beginning of the learning process, and gradually reduce the exploration favouring the exploitation of the current knowledge. For doing this the following parameters are defined:

- EPS_START: starting greedy value.
- EPS_END: ending greedy value.
- EPS_DECAY: greedy decay rate

Finally, the state action function is replaced with a deep neural network. The input of the network is the image from the video camera, and the output the reward or every action. The loss function, for the gradient descent algorithm, is defined as:

$$\text{Loss} = 1/2 \cdot [r_t + \max_{a_{t+1}} \cdot Q(s_{t+a}, a_{t+1}; \theta_{t-1}) - Q(s, a; \theta)]^2$$

Two extra techniques are included to make the neural network converge to the final solution. Experience replay, where a memory of experience is used to store previous values; mini-batches of these values are used as input of the neural network during learning. This technique helps to break the temporal dependencies of the learning process. Since the target of the loss function and the predictions have parameters in common, the process of learning can not converge to the correct solution, fixed Q targets is used to update the Q function after several iterations, in this way the small changes in the Q function approximation are not affecting the learning process.

The following extra parameters are defined.

- INPUT_WIDTH: Image from the camera, width
- INPUT_HEIGHT: Image from the camera, height
- OPTIMIZER: Optimizer for the gradient descent algorithm
- LEARNING_RATE: Neural Network learning rate.
- REPLAY_MEMORY: Size of the memory for the experience replay technique.
- BATCH_SIZE: Batch size for the experience replay technique.

2.2 Long Short Term Memory

LSTM is a special Recurrent Neural Network, capable of learning long-term dependencies. The Long short-term memory adds memory to the DQN to remember the sequence of movements of the joint to find and touch the object in the scene.

3 ROBOTIC ARM

In a reinforcement learning problem, a temporary series of interactions are produced until the episodic ends: $S_0, A_0, R_1, S_1, A_1, R_2, \dots, R_T, S_T$

In this project, an episode ends when: the robot's arm touches the ground, a maximum of iterations is reached, or the robot touches the object in the scene; the states are the image, captured by the video camera, of the scene; the actions can be to move the joints at random using position or speed control; finally, the reward strategy is defined as a negative number when the robot finishes the episodic without touching the object, the positive number when it touches the object and an incremental or decremental reward when the arm approaches the target or not.

Different reward strategies were tested, the final choose solution was: a big number positive number when the robot touches the object; a negative number when the robot touch the ground or the maximum iterations are reached, to this negative number a 1% of the distance to the goal is added, in order to decrease the negative reward depending on the closest to the goal; a smooth reward when the robot is searching the object in function of the smoothest of the movement and the distance to the goal.

3.1 Rewards

The positive reward when the object is touched is 1000000.0. The negative base reward is -1.0. When the episodic ends without success to the negative base reward is added 1% the distance to the goal. Finally when the Robot is searching a smooth average goal delta is added to the distance to the goal as follow:

$$\text{AverageGoalDelta} = (\text{AverageGoalDelta} * 0.2) + (\text{DistanceDelta} * (1-0.2));$$

$$\text{Reward Searching} = \text{AverageGoalDelta} + 0.01 * \exp(-\text{pow}(\text{DistanceGoal}, 2));$$

The same rewards were used for both objectives.

3.2 Hiper Paramters

Finally the hiper-parameters selecting were:

- VELOCITY_CONTROL[false]: The position control was used.
- NUMBER_ACTIONS[6]: The number of actions was calculated as number of DOF x 2.
- ALLOW_RANDOM[true]: The learning process allows random decisions to explore new solutions.
- GAMMA[0.9]: discount factor, selected was 0.9.
- EPS_START[0.9]: Starting greedy value.
- EPS_END[0.01] Ending greedy value.
- EPS_DECAY[200]: Greedy decay rate
- INPUT_WIDTH[64]: Image from the camera width
- INPUT_HEIGHT[64]: Image from the camera height
- OPTIMIZER[RMSprop]: Optimizer for the gradient descent algorithm
- LEARNING_RATE[0.01]: Neural Network learning rate.
- REPLAY_MEMORY[10000]: Size of the memory for the experience replay technique.
- BATCH_SIZE[64]: Batch size for the experience replay technique.
- USE_LSTM[true]: Uses Long Short Memory.
- LSTM_SIZE[512]: Sizes of the LSTM network.

The same Hiper Parameters were used for both objectives.

4 RESULTS

In order to find the reward and hiper-parameters a trial and error strategy was used. At the beginning velocity control was used, but the desired accuracy of 80% of success touching the object with the gripper base was not achieve, for that reason the position control was selected. The reward strategy consists in give a high positive number when the robot touches the object, this big number was selected because the robot could accumulate small rewards during the searching process and it was important to stimulate the robot not to keep searching but to move toward the goal. During the searching process a small amount of the distance to goal was added to the reward to indicate the robot's arm that the action should move the arm in the object direction. Also, a factor depending on of the delta movement, was used to increase, when the movement was directing the robot to the goal, or decrease when the movement was in the other direction. This searching strategy was used

The screenshot shows a Gazebo simulation environment. The main window displays a 3D scene with a robotic arm (a white and black articulated arm) positioned over a green cube on a red laser line. The background is a stylized, abstract pattern. A terminal window is open in the foreground, displaying ROS logs. The logs show the current accuracy of the simulation and the issuing of rewards to the 'ArmPlugin'.

Manuel Huertas Lopez

Gazebo

Current Acc: 0.9609 (123 of 128) (reward=100000.00 WtN)

ArmPlugin - Issuing reward 0.116798, EOE=False POS+

ArmPlugin - Issuing reward 0.144083, EOE=False POS+

ArmPlugin - Issuing reward 0.154037, EOE=False POS+

ArmPlugin - Issuing reward 0.159982, EOE=False POS+

ArmPlugin - Issuing reward 0.175370, EOE=False POS+

Real Time Factor: 0.76 Sim Time: 00:00:45.478 Real Time: 00:0

Manuel Huertas López

Fig. 1: Robot's Arm

[1] Robotics Software Engineer Nanodegree program <https://eu.udacity.com>

[2] Human-level control through deep reinforcement learning.

[3] Understanding LSTM Networks <http://colah.github.io/posts/2015-08-Understanding-LSTMs>