Kansas State University Undergraduate Research Conference

Spring 2019

DQN: Deep Q-Learning for Autonomous Navigation

Leonardo Garrido Alvarez

DQN: Deep Q-Learning for Autonomous Navigation

Leonardo Garrido, Carlos Aguirre, Cameron Thompson, M. F. De La Torre, V. Behzadan, W. Hsu

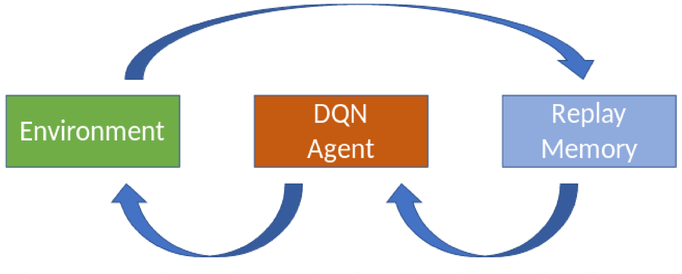
KDD Research Lab | Department of Computer Science | Electrical Engineering

Introduction

This project deals with autonomous mobile robots trained using reinforcement learning, a branch of machine learning (the science of improving problem-solving performance based on experience) based on choosing actions to maximize rewards from various environments. This is a form of behavioral learning that is observed in nature and thus more biologically plausible than cognitive models based on labeled data provided by a teacher (supervised learning). We developed an experimental test bed by implementing Deep Q-Networks (DQN), a form of reinforcement learning, for goal-oriented navigation and obstacle avoidance tasks using a *TurtleBot3* Burger robot and in the *gazebo* simulation environment for behavior learning in autonomous agents. To achieve the goal of avoiding obstacles, the DQN Agent provides a positive reward to the robot whenever it gets closer to its goal and a negative reward when it is farther from its goal. The *TurtleBot3* Burger requires a large number of training iterations before it achieves the goal and successfully avoids obstacles. Future work involves extending the reward functions so that DQN can be used to learn to solve fully autonomous exploration and mapping tasks, where the robot does not know the exact location of the goal.

Problem Statement

Our job is to train a DQN agent that learns an optimal policy to navigate the robot from point a to point b with minimum effort. How does a DQN agent works?



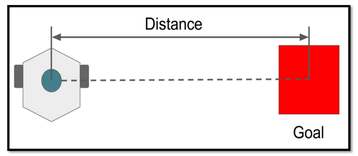
- The agent selects the action by Q-value and reflects it in the environment.
- It stores the reward obtained in an environment in replay memory.
- The agent is updated by randomly sampling the stored samples from the replay memory.
- The agent's goal is to get the *Turtlebot3* to a target state.

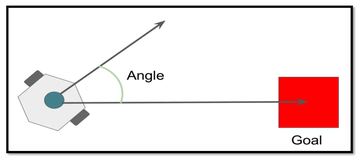
State

State: is an observation of environment and describes the current situation. This is vital for the agent because it would calculate and act depending on the state. The state size is 26 and 24 LDS (Laser Distance Sensor) values. The other two are distances to goal, and angle to goal. A mathematical approach for this is as follow:

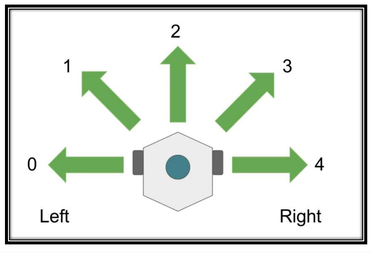
State = *LDS* (24 values) + *Distance* (1) + *Angle* (1)

LDS denotes the (24) values that the lidar sensor emits. *Distance* represents the distance to the goal and *Angle* is the angle between the robot heading and vector to the goal.





Actions (Degrees of Freedom): The robot has five actions which can act on depending on the type of state. In here, the robot has a fixed linear velocity of .15m/s and the angular velocity is determined by the state.



Action	Angular velocity(rad/s)
0	-1.5
1	-0.75
2	0
3	0.75
4	1.5

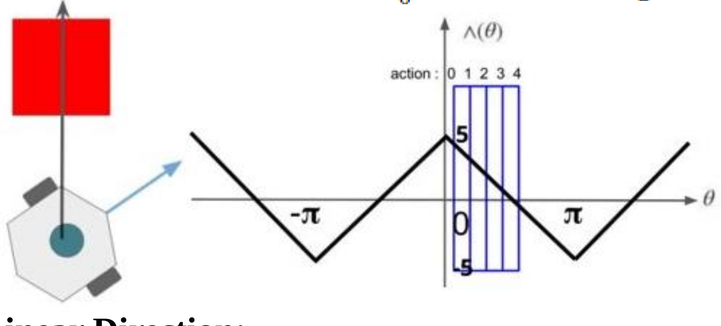
Experiment Setup

Angular Direction:

$\theta = \frac{\pi}{2} + action * \frac{\pi}{8} + \phi$

$R_\theta = 5 * 1^{-\theta}$

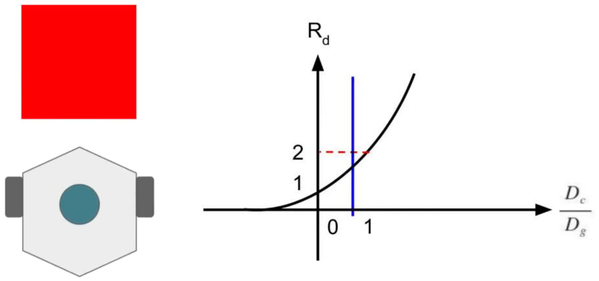
ϕ : Yaw of *TurtleBot3*
θ : Angle from Goal
R_θ : Reward from angle



Linear Direction:

$R_d = 2^{\frac{D_c}{D_g}}$

D_c : Current distance from Goal
D_g : Absolute distance from Goal
R_d : Reward from distance



Reward Function:

if $-\frac{1}{2}\pi < \theta < \frac{1}{2}\pi$, $R_\theta \geq 0$
else $R_\theta < 0$
if $D_c < D_g$, $R_d > 2$
else $1 < R_d \leq 2$
$R = R_d * R_\theta$

θ : Angle from *TurtleBot3* to Goal
D_c : Current distance from Goal
D_g : Absolute distance from Goal
R_θ : Reward from θ
R_d : Reward from distance
R : Reward

Progress to Date

- **Experiment design:** The simulation completes when the *TurtleBot3* either collides with an obstacle, reaches goal, or after time-step limit is exceeded.
- **Measure of quality:** The reward function executes when the robot interacts with obstacle or reaches a goal by providing feedback.
- **Implementation:** Built maze environment (**video**).
- **Desired capability, plan:** **Learn** state/action by maximizing reward function through (100-1000+) simulations.

Future Work

- **Scientific challenge:** Develop new reward functions to facilitate evolution of different behaviors including adding new features to the environment.
- **Hypotheses:** DQN with lookahead parameters can learn better than plain DQN (current work).
- **Test beds:** Develop/test other real-world environments. Design a higher quality robot for challenging environment.

References

- Garrido, Alvarez L. (2019). *ROBOTIS e-Manual*. [online] ROBOTIS e-Manual. Available at: http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/ [Accessed 1 Apr. 2019].
- Garrido Alvarez, L. (2019). *Documentation - ROS Wiki*. [online] Wiki.ros.org. Available at: http://wiki.ros.org/Documentation [Accessed 1 Apr. 2019].