Self-Driving Agent

Summary

The intention of this exercise is to teach a self-driving car (Agent) to automatically drive to reach the final destination maximizing rewards. Q learning algorithm is implemented.

Key Factors:

Rewards:

Any time the agent acts, a reward is reached, the possible rewards are: - 10: Agent reaches destination - 2: Agent action is valid and equals the next_waypoint() policy - 1: Agent action is None - 0.5: Agent action is valid but is not the next_waypoint() policy - -1: Agent action is not valid

Planner:

Agent uses a planner to reach destination, **this planner identifies the next waypoints** agent should follow to reach destination in a grid environment. However planner is limited since the process to find the optimun waypoints firstly checks for optimal waypoints in the North-South direction and, in case None is found, then in the East-West direction. As a consequence, in cases where there are two optimal waypoints, only the waypoint in the North-South direction is reported.

States:

Agent' state includes the following information: - Street light color - Oncoming traffic - traffic on the left - traffic on the right - Next waypoint

Since rewards are related to actions taken and environment rules defined by traffic conditions (Oncoming, left, right), street light colors and next waypoint, agent' state need to include this information to learn which action maximizes rewards for the current conditions (traffic + street lights + next waypoint).

Q-Learning:

In order to provide the ability to the agent of learning from previous states, a Q-matrix is generated. Each element of this Q matrix is an agent state&action which includes the following information: - Street light color - Oncoming traffic - traffic on the left - traffic on the right - Next waypoint - Action taken

Each state&action in the the Q-matrix has a particular value that is updated accordingly to the reward obtained. At the very begining of the game, all possible pairs of states and actions had been assigned to a value of 0 but as the agent walks through the different states and takes different actions received rewards are used to update values. Values are updated with the received reward of the current state plus the maximun Q-value for the next state among all possible actions.

Q-learning agent:

Agent target is to reach destination while maximizing rewards received, for such reason q-learning agent follows the Planner policy but "senses" enviroment states (traffic + street light + Next waypoint) to decide which action take. The way the agent moves around the environment is the following:

1. Sense current state (state1)
2. Choose an action (action1) that maximizes Q-equation from this current state
3. Take the selected action and move agent to the next state (state2) and get a reward
4. Sense the new state and update the Q-equation for (state1,action1) using the reward just received and the (state2,action2), being action2 the action that maximizes Q-equation for state2

Difference between Basic Agent and q-learning Agent:

Basic agent is not "sensing" the environment conditions and simply follows planner actions defined to reach final destination. As a consequence it performs actions that are negatively rewarded, in the other hand q-learning agent "senses" the environment anytime it needs to take an action and so understand the environment conditions, although q-learning agent also follows planner actions, at every step the final decision it takes is the one that maximizes rewards, independenly of the planner action required. In practice, basically the main difference between agents is that basic agent never takes an action = 'None' while q-learning agent takes it every time the planner action has associated a negative regard for the current state conditions.

Learning Rate (alpha) and Discount Factor (gamma) tuning:

The parameters used in the Q-value update process are:

- the learning rate (alpha), set between 0 and 1. Setting it to 0 means that the Q-values are never updated, hence nothing is learned. Setting a high value such as 0.9 means that learning can occur quickly.
- discount factor (gamma), also set between 0 and 1. This models the fact that future rewards are worth less than immediate rewards. Mathematically, the discount factor needs to be set less than 0 for the algorithm to converge.

Gamma = 0.6

Rewards	alpha 0.2	alpha 0.4	alpha 0.6	alpha 0.8
-1	17	17	16	18
0.5	4	363	3	239
1	635	10	623	282
2	489	949	507	766
12	100	98	100	95
TOTAL	2798	3248.5	2822.5	3055.5

Gamma = 0.2

Rewards	alpha 0.2	alpha 0.4	alpha 0.6	alpha 0.8
-1	12	14	13	9
1	546	652	591	644
2	505	492	523	513
12	100	100	99	100
TOTAL	2744	2822	2812	2861

The selected alpha and gamma values are 0.4 and 0.6 respectively since these values maximizes rewards obtained (1364).

Results:

In the following graph it is represented the total number of different rewards for a simulation of 100 trials for an "intelligent" and "non intelligent" agent. The intelligent agent uses the Q equation to identify the optimum actions while non-intelligent agent simply follows those actions towards the next waypoint indicated by the Planner.

In the way rewards are defined, since an action=None has a rewards of 1, while an action that is valid but not to the next_waypoint is rewarded with 0.5, it is not expected the intelligent agent to be faster than the non-trained agent reaching destination, but an overall increase of the rewards obtained, as the following graph shows:

Conclussion:

Q-learning agent behaves optimally following the planner when possible avoiding negative rewards. However planner is limited since in most occasions there are two valid actions but it only returns one. As a consequence, q-learning agent does not reaches destination more quickly than the basic agent does, however it maximizes rewards along the way. To avoid such shortcomming the planner should return two optimum actions when possible and let the agent to decide which action take, doing so q-learning object would not just maximize regrads, but also reach destination faster than basic agent.