SmartCab

Implement a Basic Driving Agent

- *Observe what you see with the agent's behavior as it takes random actions. Does the smartcab eventually make it to the destination? Are there any other interesting observations to note?*

  Yes, it reaches the destination, but not all the times within the hardline of 100. It does run some red lights and it doesn’t give the right of way all the time but in general it does. In red lights it tends to turn right and in many it stays put in red lights, it may be because it doesn’t get a penalty because of that.

Inform the Driving Agent

- *What states have you identified that are appropriate for modeling the smartcab and environment? Why do you believe each of these states to be appropriate for this problem?*

  I have identified states based only on the light, left, oncoming and next. These states are appropriate because they cover all the possible restrictions to traffic so I consider them to be significantly more relevant than all the others.

  light: the color of the light is the main input for defining the penalties or rewards.
  next_waypoint: the next state is critical in order to know in which direction de agent is heading.
  left: if the right is red it matters if there’s a car coming from the left as it restricts the agent crossing to the right without getting a penalty.
  oncoming: if the light is green it matters if there’s a car coming onward as it restricts the agent crossing to the left without getting a penalty.
  right: whether the light is red or light it makes no difference if there’s a car coming for the right as it doesn’t restricts the ability of the agent crossing in any direction without receiving a penalty.
  deadline: even though knowing that the deadline is close could increase the willingness of the agent to break traffic rules, I decided not consider that option into my approach in order to reduce the number of inputs to consider.

- *How many states in total exist for the smartcab in this environment? Does this number seem reasonable given that the goal of Q-Learning is to learn and make informed decisions about each state? Why or why not?*

  There are 128 possible states, all the possible combinations of the selected inputs (light, left, oncoming and next). It seems reasonable because it significantly reduces the number of states that would result if all the inputs were considered.

Implement a Q-Learning Driving Agent

- *What changes do you notice in the agent's behavior when compared to the basic driving agent when random actions were always taken? Why is this behavior occurring?*

  The success rates improve from almost zero to 8%-9%, this is basically because information is being recorded and the model can said to be learning, it seems to obey traffic rules and seems to move toward the destination.

Improve the Q-Learning Driving Agent

- *Report the different values for the parameters tuned in your basic implementation of Q-Learning. For which set of parameters does the agent perform best? How well does the final driving agent perform?*

  The following table describes the performance of the algorithm in terms of success rates based on different combinations of alpha, gamma and epsilon values. The average is calculated based on five 100 trials runs.

Alpha	Gamma	Epsilon	Success Rate (Avg.)
0.9	0.9	0.75	93.8%
0.1	0.1	0.75	94.8%
0.9	0.1	0.5	94%
0.1	0.9	0.5	86%
0.1	0.1	0.1	95%
0.9	0.9	0.1	92%

  Based on the table it seems that ability of the agent to explore does affect the performance of the agents with lower values of alpha and gamma also improving the success rate.

  Also the algorithm takes into consideration when the reported Q values are not significant enough to be taken into consideration in which case a random action is taken, by doing this learned actions are taken only if the registered Q values are larger than zero, otherwise it assumes that so far only wrong actions were taken in the past for that specific state and goes for a random action.

- *Does your agent get close to finding an optimal policy, i.e. reach the destination in the minimum possible time, and not incur any penalties? How would you describe an optimal policy for this problem?*

  My model gets close to find an optimal policy (i.e. obeying traffic rules and move in the direction of the destination) within the enforced deadline, doing very well at obeying traffic laws but not so well at going always in the right direction. My model -even though it reaches the goal a majority of the times0 it usually commit infractions at beginning, mostly going through red lights, and it will keep being penalized for going in the wrong direction even in the latest trials.